

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**Facultad de Ingeniería en Mecánica y Ciencias de la  
Producción**

“Actualización del Sistema Operativo, Manual de Operación y  
Guía de Prácticas para el Sistema Didáctico Robótico Móvil del  
Laboratorio de Mecatrónica de la FIMCP-ESPOL”

**TESIS DE GRADO**

Previo a la obtención del Título de:

**INGENIERO MECÁNICO**

Presentado por:

**OSMAR GEOVANNI LEMA CAICEDO**

**GUAYAQUIL – ECUADOR**

**2013**

## **AGRADECIMIENTO**

A Dios, en primer lugar, por otorgarme la existencia y permitirme llegar donde estoy.

A mi madre, por ser una guía a lo largo de mi vida y por toda la paciencia y abnegación que me tiene.

A mi padre por todos sus sacrificios, dedicación y atención hacia la familia.

Y por último a todas las personas que de una u otra forma colaboraron con la realización de este trabajo, en especial a mi Director de Tesis Ing. Msc. Eduardo Orces Pareja, por su invaluable ayuda.

# DEDICATORIA

A Mis Padres.

A Mi Hermano.

A Mi Novia.

## **TRIBUNAL DE GRADUACIÓN**

---

**Dr. Kléber Barcia V., Ph.D.  
DECANO DE LA FIMCP  
PRESIDENTE**

---

**Ing. Eduardo Orcés P., M.Sc.  
DIRECTOR DE TESIS**

---

**Ing. Jorge Roca G.  
VOCAL**

## **DECLARACIÓN EXPRESA**

La responsabilidad del contenido de esta Tesis de Grado me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL.

(Reglamento de Graduación de la ESPOL)

---

Osmar Geovanni Lema Caicedo.

## RESUMEN

El propósito de esta tesis es elaborar nuevos manuales de uso y guías de práctica para el sistema didáctico robótico móvil, debido a que las anteriores quedaron obsoletas por una actualización del sistema operativo que recibió la unidad que forma parte del laboratorio de Mecatrónica de la Facultad de Ingeniería en Mecánica y Ciencias de la Producción (FIMCP-ESPOL).

También en el presente trabajo se expone las generalidades del sistema robótico de nombre *Robotino® fabricado por Festo®*, su uso y manipulación.

Además se explica la programación para el sistema en Robotino®view 2 y Matlab® 7

Por último se incluye las guías actualizadas y reformuladas en Robotino®view 2 y las prácticas nuevas programando en MatLab®, para el laboratorio.

## ÍNDICE GENERAL

RESUMEN.....	VI
ÍNDICE GENERAL.....	VII
ABREVIATURAS .....	X
SIMBOLOGÍA .....	XI
ÍNDICE DE PLANOS Y FIGURAS.....	XII
ÍNDICE DE TABLAS .....	XVII
INTRODUCCIÓN.....	1
CAPÍTULO 1:.....	4
1 MARCO TEÓRICO.....	4
1.1 Generalidades .....	4
1.2 Sistema didáctico robótico móvil.....	7
1.3 Reconocimiento de los componentes del sistema didáctico robótico móvil.....	10
1.4 Uso, control y puesta en marcha del sistema didáctico robótico móvil. ....	25
1.5 Actualización sistema operativo del sistema didáctico robótico móvil. ....	29
CAPÍTULO 2:.....	35
2 SOFTWARE DE COMUNICACIÓN Y CONTROL DEL SISTEMA DIDÁCTICO ROBÓTICO MÓVIL.....	35
2.1 Robotino®view 2.....	36
2.1.1 Generalidades de Robotino®view 2.....	37
2.1.2 Introducción a la programación del sistema didáctico robótico móvil en Robotino®view 2 .....	42

2.1.2.1	Ejemplos de aplicación .....	45
2.2	MatLab® 7 para el sistema didáctico robótico móvil.....	56
2.2.1	Generalidades de MatLab® 7 y sus controladores para el sistema didáctico robótico móvil.....	56
2.2.2	Introducción a la programación del sistema didáctico robótico móvil en MatLab® 7.....	60
2.2.2.1	Ejemplos de aplicación .....	67
CAPÍTULO 3:.....		70
3	DESARROLLO DE NUEVAS PRÁCTICAS DE LABORATORIO CONSIDERADAS.....	70
3.1	Robotino®view 2: Control de las comunicaciones, puesta en funcionamiento y programación de movimientos lineales en sentidos indistintos del sistema robótico .....	71
3.2	Sensores de reflexión directa: programación del movimiento guiado del sistema robótico.....	78
3.3	Detector analógico inductivo: comportamiento frente a diversos tipos de materiales y programación del movimiento guiado del sistema robótico. ....	85
3.4	Detectores de distancias infrarrojos: línea característica (curva de calibración), avance en función de distancias precisas y mantenerlas, bordear o evitar obstáculos con el sistema robótico. ....	94
3.5	Motores: control, calibración, movimiento lineales y posicionamiento del sistema robótico.....	108
3.6	Cámara digital: detección de una pieza de diferentes colores y ejercicios de aplicación.....	116
3.7	Movimiento guiado del sistema robótico mediante el uso de una cámara digital .....	128
3.8	MatLab® 7: control de comunicaciones, puesta en funcionamiento y programación de tareas básicas del sistema robótico.....	132
3.9	Detectores de distancia infrarrojos: programación de sistema de exploración a través de MatLab® 7.....	137










3.10	Detector analógico inductivo: programación de movimiento guiado en MatLab® 7.....	142
3.11	Sensores de reflexión directa: programación de movimiento guiado en MatLab® 7 mediante bloques en Simulink. ....	147
CAPÍTULO 4:.....		156
4	CONCLUSIONES Y RECOMENDACIONES.....	156
APÉNDICE		
BIBLIOGRAFÍA		

## ABREVIATURAS

- FIMCP:** Facultad De Ingeniería En Mecánica Y Ciencias De La Producción.
- GRAFCE:** (GRAphe Fonctionel de Commande Etape Transition), es un grafo o diagrama funcional normalizado, que permite hacer un modelo de un proceso a automatizar, contemplando entradas, acciones a realizar, y los procesos intermedios que provocan estas acciones
- GUI:** graphical user interface (interfaz gráfica del usuario)
- HSV:** Del inglés Hue, Saturation, Value que significa Matiz, Saturación, Valor
- IDE:** integrated development environment (entorno de desarrollo integrado)
- P.I.D.:** Control “proporcional, integral y derivativo”, es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso.
- S.D.Ro.M.:** Sistema didáctico robótico móvil
- S.O.:** Sistema Operativo (O.S. operative system)

## SIMBOLOGÍA

	Tierra (símbolo eléctrico)
	Diodo luminoso (símbolo eléctrico)
	Símbolo de Robotino®
	Establecer conexión (Robotino®view)
	Abrir nuevo proyecto (Robotino®view)
	Abrir nuevo subprograma (Robotino®view)
	Abrir proyecto existente (Robotino®view)
	Insertar ramal alternativo (Robotino®view)
	Insertar paso/transición (Robotino®view)
	Iniciar programa principal (Robotino®view)
	Iniciar subprograma actual (Robotino®view)

## ÍNDICE DE PLANOS Y FIGURAS

FIGURA 1.1: RAMAS QUE CONFORMAN LA INGENIERÍA MECATRÓNICA.....	5
FIGURA 1.2: ALAN TURING Y SU MÁQUINA "ENIGMA" (MÁQUINA DE CIFRADO).....	5
FIGURA 1.3: SISTEMA DIDÁCTICO ROBÓTICO MÓVIL.....	7
FIGURA 1.4: PRINCIPALES COMPONENTES DEL S.D.RO.M. ROBOTINO® .....	10
FIGURA 1.5: CÁMARA DIGITAL USADA POR S.D.RO.M. ROBOTINO® ....	11
FIGURA 1.6: "PUENTE DE MANDO" DE ROBOTINO® .....	12
FIGURA 1.7: UNIDAD DE ACCIONAMIENTO DE ROBOTINO® .....	13
FIGURA 1.8: CARACTERÍSTICAS DE UN ENCODER.....	14
FIGURA 1.9: RODILLOS OMNIDIRECCIONALES.....	16
FIGURA 1.10: VISTA DE SECCIÓN DEL LISTÓN PROTECTOR.....	17
FIGURA 1.11: FUNCIONAMIENTO DETECTOR INFRARROJO (IZQ.); DETECTOR PSD (DER.).....	19
FIGURA 1.12 SENSOR DE DISTANCIA; EMISOR, RECEPTOR Y SISTEMA DE PROCESAMIENTO ESTÁN INCLUIDOS EN LA MISMA UNIDAD.....	20
FIGURA 1.13: SENSOR ÓPTICO Y UNIDAD DE FIBRA ÓPTICA.....	21
FIGURA 1.14: FUNCIONAMIENTO DEL SENSOR INDUCTIVO.....	22
FIGURA 1.15: DETECTOR INDUCTIVO USADO POR EL S.D.RO.M.....	23
FIGURA 1.16: CHASIS DELS.D.RO.M.....	24
FIGURA 1.17: BATERÍAS PLOMO-GEL USADAS POR S.D.RO.M.....	24
FIGURA 1.18: CONTROL DE MANDO MANUAL DE ROBOTINO®.....	25
FIGURA 1.19: (A) PANTALLA PRINCIPAL, (B) PANTALLA MENÚ PRINCIPAL, (C) SUBMENÚ DEMOS.....	26
FIGURA 1.20: RED INALÁMBRICA DE ROBOTINO® DETECTADA POR EL COMPUTADOR.....	27
FIGURA 1.21: VENTANA "SIMBOLO DEL SISTEMA", PARA VERIFICAR LA CORRECTA CONEXIÓN.....	28
FIGURA 1.22: INCORRECTA CONEXIÓN A ROBOTINO®.....	28
FIGURA 1.23: WINDOWS7® CORRIENDO LA CONSOLA VIRTUAL "XP MODE".....	31
FIGURA 1.24: ARCHIVO NECESARIOS PARA ACTUALIZACIÓN.....	32
FIGURA 1.25: PASOS A SEGUIR POR EL PROGRAMA DE ACTUALIZACIÓN.....	33

FIGURA 2.1: INTERFACE DE USUARIO DE ROBOTINO®VIEW: (1) BARRA DE TÍTULO; (2) BARRA DE MENÚ; (3) BARRA DE HERRAMIENTAS; (4) PESTAÑAS DE SELECCIÓN DE PROGRAMAS; (5) ESPACIO DE TRABAJO DEL SUBPROGRAMA; (6) LIBRERÍA DE BLOQUES DE FUNCIONES; (7) BARRA DE ESTADO.....	38
FIGURA 2.2: VENTANA PARA VISUALIZAR Y CARGAR PROYECTOS A ROBOTINO®.....	41
FIGURA 2.3: VENTANA DE ACTUALIZACIÓN.....	42
FIGURA 2.4: BLOQUES DE FUNCIÓN.....	44
FIGURA 2.5: VALORES DE CONSIGNA Y PRESENTACIÓN DE VALORES ACTUALES DEL BLOQUE DE FUNCIÓN.....	44
FIGURA 2.6: VARIABLES GLOBALES EN ROBOTINO®VIEW.....	45
FIGURA 2.7: CONTENIDO DE LA PESTAÑA "PRINCIPAL".....	46
FIGURA 2.8: CONTENIDO DE LA PESTAÑA PRINCIPAL MODIFICANDO EL NOMBRE DEL SUBPROGRAMA.....	47
FIGURA 2.9: EJEMPLO, USO DE BLOQUES DE FUNCIONES Y ESPACIO DE TRABAJO.....	48
FIGURA 2.10: CREACIÓN DE REDES O CONEXIONES Y ASIGNACIÓN DE VALORES A BLOQUES DE FUNCIONES PARA EJEMPLO DE APLICACIÓN.....	49
FIGURA 2.11: CREACIÓN DE SUBPROGRAMAS PARA EJEMPLO DE APLICACIÓN.....	50
FIGURA 2.12: CREACIÓN DE PASOS Y CONDICIONES DE TRANSICIÓN PARA EL EJEMPLO DE APLICACIÓN.....	51
FIGURA 2.13: PROGRAMA PRINCIPAL TERMINADO PARA EL EJEMPLO DE APLICACIÓN.....	52
FIGURA 2.14: BLOQUE ALEATORIO DEL EJEMPLO 2.....	53
FIGURA 2.15: CREACIÓN DE RAMALES ALTERNATIVOS Y/O PARALELOS EJEMPLO 2: (A) LÍNEAS DE SECUENCIA ENTRE PASOS; (B) OPCIONES DE RAMALES.....	54
FIGURA 2.16: EJEMPLO 2 (A) RAMALES ALTERNATIVOS (B) PASOS DE PROGRAMA EN PARALELO.....	55
FIGURA 2.17: SECUENCIA TERMINADA DEL PROGRAMA PRINCIPAL DEL EJEMPLO 2.....	56
FIGURA 2.18: CONFIGURACIÓN DE MATLAB® PARA ROBOTINO® CONTROL EN TIEMPO REAL.....	59

FIGURA 2.19: CONFIGURACIÓN DE MATLAB® PARA ROBOTINO® (COMPILADOR).....	60
FIGURA 2.20: DIAGRAMA DE FLUJO.....	61
FIGURA 2.21: CONDICIONALES EN LA PROGRAMACIÓN.....	63
FIGURA 2.22: ESTRUCTURA "IF" .....	63
FIGURA 2.23: ESTRUCTURA DE LOS LAZOS O BUCLES EN MATLAB®.....	65
FIGURA 3.1: CÁLCULO DE RECORRIDO SIN USAR BLOQUE DE OMNIDIRECCIONAMIENTO .....	72
FIGURA 3.2: PROGRAMACIÓN DEL PASO 4 PARA COMPROBAR COMPONENTES, PRÁCTICA 1.....	75
FIGURA 3.3: PROGRAMA PARA PASO 5, PRÁCTICA 1.....	75
FIGURA 3.4: PROGRAMA PARA PASO 7, PRÁCTICA 1.....	76
FIGURA 3.5: PARTES CONSTITUTIVAS DE LOS DETECTORES ÓPTICOS DEL S.D.RO.M. ROBOTINO®.....	80
FIGURA 3.6: ESQUEMA DE CONEXIÓN DE LOS DETECTORES ÓPTICOS.....	82
FIGURA 3.7: AYUDA EN LA CALIBRACIÓN DE LOS DETECTORES ÓPTICO. ....	83
FIGURA 3.8: PROGRAMACIÓN BÁSICA PARA EL MOVIMIENTO GUIADO USANDO LOS DETECTORES ÓPTICOS; PASO 5 PRÁCTICA 2.....	83
FIGURA 3.9: CONEXIÓN DEL SENSOR INDUCTIVO A ROBOTINO®; PASO 2 PRÁCTICA 3.....	88
FIGURA 3.10: PROGRAMACIÓN DE AYUDA PARA LA COMPROBACIÓN Y CALIBRACIÓN DE LOS DETECTORES INDUCTIVOS; PASO 3 PRÁCTICA 3.....	89
FIGURA 3.11: PROGRAMACIÓN PARA DIFERENCIACIÓN DE MATERIALES; PASO 4 PRÁCTICA 3 .....	90
FIGURA 3.12: COMPORTAMIENTO DEL SENSOR ANTE MATERIALES METÁLICOS; PASO 4 PRÁCTICA 3.....	90
FIGURA 3.13: ESQUEMA DE LA ESTRATEGIA TOMADA PARA LA PROGRAMACIÓN DEL MOVIMIENTO GUIADO A TRAVÉS DE UN TRAYECTO CON UN SENSOR INDUCTIVO; PASO 5 PRÁCTICA 3.....	92
FIGURA 3.14: PROGRAMACIÓN BÁSICA DEL MOVIMIENTO GUIADO DE ROBOTINO® CON UN SENSOR INDUCTIVO; PASO 5 PRÁCTICA 3.....	93

FIGURA 3.15: ESQUEMA PARA IDENTIFICAR CADA SENSOR DE DISTANCIA DE ROBOTINO®; PASO 1 PRÁCTICA 4. ....	97
FIGURA 3.16: PROGRAMA PARA IDENTIFICAR Y VERIFICAR LOS SENSORES DE DISTANCIA INFRARROJOS; PASO 1 PRÁCTICA 4. ....	98
FIGURA 3.17: ESQUEMA PARA REGISTRAR LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4. ....	99
FIGURA 3.18: CURVA DE CALIBRACIÓN DE UN SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4. ....	100
FIGURA 3.19: LINEALIZACIÓN PARA LA CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4. ....	101
FIGURA 3.20: PROGRAMA BÁSICO PARA AVANCES EN FUNCIÓN DE DISTANCIAS PRECISAS; PASO 3 PRÁCTICA 4. ....	102
FIGURA 3.21: SECUENCIA ADICIONAL PARA EL AVANCE A LO LARGO DE UNA PARED; PASO 4 PRÁCTICA 4. ....	103
FIGURA 3.22: ESQUEMA 1 DE LA ESTRATEGIA TOMADA PARA QUE LA UNIDAD RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4. .	104
FIGURA 3.23: ESQUEMA 2 DE LA ESTRATEGIA TOMADA PARA QUE LA UNIDAD RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4. .	105
FIGURA 3.24: SECUENCIA ADICIONAL PARA LOGRAR QUE ROBOTINO® RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4. ....	106
FIGURA 3.25: PROCESO CON RESPUESTA OSCILATORIA MANTENIDA ANTE UNA PERTURBACIÓN DE ENTRADA ESCALÓN; MÉTODO DE SINTONIZACIÓN PID ZIEGLER-NICHOLS EN LAZO CERRADO. ....	111
FIGURA 3.26: SINTONIZACION PID: AJUSTE DE KP. ....	111
FIGURA 3.27: PROGRAMA PARA SINTONIZACIÓN PID DE ZIEGLER-NICHOLS; PASO 1 PRÁCTICA 5. ....	113
FIGURA 3.28: PROGRAMA PARA ANÁLISIS ANTE ENTRADA "ESCALON"; PASO 5 PRÁCTICA 5. ....	114
FIGURA 3.29: ESQUEMA PARA EL RECONOCIMIENTO DE PATRONES E IMAGENES. ....	118
FIGURA 3.30: ESPACIO DE COLOR "HSV" . ....	120
FIGURA 3.31: SECUENCIA PARA RECNOCIMIENTO DE IMÁGENES; PASO 1 PRÁCTICA 6. ....	122

FIGURA 3.32: USO DEL BLOQUE "BÚSQUEDA DE GAMAS DE COLORES", (A)SE VISUALIZA LA ENTRADA (B)SE CALCULA LOS VALORES MÍNIMO Y MÁXIMO "HSV" (C)REPRESENTACIÓN VISUAL DE LA GAMA DE COLORES SELECCIONADA; PASO 1 PRÁCTICA 6. ....	123
FIGURA 3.33: RESULTADO DEL PROCESAMIENTO DE IMÁGENES ELABORADO POR EL BLOQUE DE FUNCIONES "BÚSQUEDA DE GAMAS DE COLORES"; PASO 1 PRÁCTICA 6. ....	124
FIGURA 3.34: USO DE BLOQUE "RASTREADOR DE SEGMENTOS"; PASO 1 PRÁCTICA 6. ....	125
FIGURA 3.35: PRIMERA SECUENCIA DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 1 PRÁCTICA 6. ....	126
FIGURA 3.36: SEGUNDA SECUENCIA DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 2 PRÁCTICA 6. ....	126
FIGURA 3.37: PROGRAMA PRINCIPAL DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 2 PRÁCTICA 6. ....	127
FIGURA 3.38: VENTANA DESPLEGADA POR EL BLOQUE "DETECTOR DE LÍNEAS"; PASO 1 PRÁCTICA 7. ....	130
FIGURA 3.39: PROGRAMACIÓN BÁSICA PARA EL MOVIMIENTO GUIADO DE ROBOTINO® CON LA AYUDA DE UNA CÁMARA DIGITAL; PASO 2 PRÁCTICA 7. ....	131
FIGURA 3.40: LIBRERÍA DE BLOQUES PARA SIMULINK®; PRÁCTICA 11. ....	150
FIGURA 3.41: SUB-MÁSCARA DEL BLOQUE "DIGITALINPUT" .....	151
FIGURA 3.42: LIBRERÍAS DE SIMULINK; PASO 1 PRÁCTICA 11. ....	152
FIGURA 3.43: BLOQUE "COM" DE SIMULINK®. ....	153
FIGURA 3.44: BLOQUE "OMNIDRIVE" DE SIMULINK®; PASO 1 PRÁCTICA 11. ....	153
FIGURA 3.45: BLOQUE "MOTOR" DE SIMULINK®; PASO 1 PRÁCTICA 11. ....	154
FIGURA 3.46: BLOQUE "DIGITALINPUT" DE SIMULINK®; PASO 1 PRÁCTICA 11. ....	154
FIGURA 3.47: PROGRAMACIÓN DEL MOVIMIENTO GUIADO DEL S.D.RO.M. EN SIMULINK®; PASO 1 PRÁCTICA 11. ....	155



## ÍNDICE DE TABLAS

TABLA 1.1: MOTOR DC Y ESPECIFICACIONES TÉCNICAS.....	13
TABLA 1.2: CARACTERÍSTICAS TÉCNICAS DEL REDUCTOR PLANETARIO .....	16
TABLA 2.1: OPERACIONES DE COMPARACIÓN PARA PROGRAMAR. ..	62
TABLA 3.1: OBSERVACIÓN DEL RECORRIDO REAL DEL S.D.RO.M., PRÁCTICA 1 .....	76
TABLA 3.2: VERIFICACIÓN DE LOS DETECTORES DE DISTANCIA; PASO 1 PRÁCTICA 4. ....	98
TABLA 3.3: REGISTRO DE LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4. ....	99
TABLA 3.4: TABLA DE ZIEGLER- NICHOLS PARA EL CÁLCULO DE LAS CONSTANTES PID.....	112

# INTRODUCCIÓN

Con el advenimiento de la nueva carrera en la Escuela Superior Politécnica del Litoral, Ingeniería en Mecatrónica, la Facultad de Ingeniería en Mecánica y Ciencias de la Producción (FIMCP) implementó un nuevo laboratorio para esta rama.

El laboratorio de Mecatrónica, se inició con la adquisición de nuevos equipos entre los que se encuentran: equipos de computación, un brazo robótico, estaciones de automatización y el sistema didáctico robótico móvil (*Robotino® de Festo®*).

Esta tesis se centra en el sistema didáctico robótico móvil, que después de ser actualizado con un nuevo sistema operativo, nace la necesidad de realizar nuevos manuales de uso y prácticas de laboratorio, debido a que las anteriores quedaron obsoletas

Con estas nuevas guías de laboratorio se espera suplir las necesidades académicas del estudiante y que logre asimilar los conocimientos adquiridos en el aula mediante análisis y pruebas.

Este trabajo, en primer lugar, presenta una descripción completa de sistema didáctico robótico móvil y sus componentes, denotando sus características y especificaciones técnicas e indicando también el procedimiento usado para su actualización.

Seguidamente, se da una introducción al nuevo software de programación de la unidad: Robotino®view 2 y otro software alternativo “MatLab® 7”, que posteriormente será complementada con las guías prácticas.

A continuación, se expone cada una de las nuevas prácticas que se implementaran para comprensión y uso de la unidad robótica.

Las prácticas planeadas para este trabajo son:

- Robotino®view 2: Control de las comunicaciones, puesta en funcionamiento y programación de movimientos lineales en sentidos indistintos del sistema robótico
- Sensores de reflexión directa: programación del movimiento guiado del sistema robótico.
- Detector analógico inductivo: comportamiento frente a diversos tipos de materiales y programación del movimiento guiado del sistema robótico.
- Detectores de distancias infrarrojos: línea característica (curva de calibración), avance en función de distancias precisas y mantenerlas, bordear o evitar obstáculos con el sistema robótico.

- Motores: control, calibración movimiento lineales y posicionamiento del sistema robótico.
- Cámara digital: detección de una pieza de diferentes colores y ejercicios de aplicación.
- Movimiento guiado del sistema robótico mediante el uso de una cámara digital
- MatLab® 7: control de comunicaciones, puesta en funcionamiento y programación de tareas básicas del sistema robótico.
- Detectores de distancia infrarrojos: programación de sistema de exploración a través de MatLab® 7.
- Detector analógico inductivo: programación de movimiento guiado en MatLab® 7.
- Sensores de reflexión directa: programación de movimiento guiado en MatLab® 7 mediante bloques en Simulink.

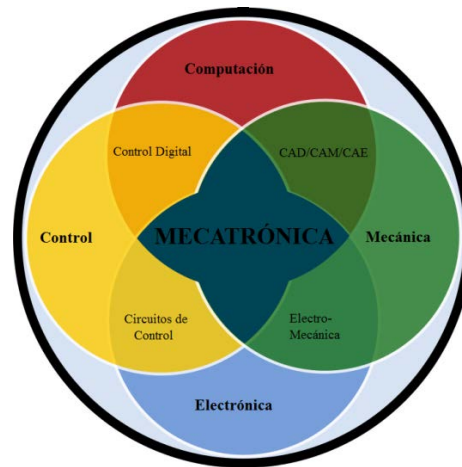
Por último, se espera llevar a cada estudiante a la comprensión total y familiarización de cada uno de los componentes y sensores contenidos en la unidad robótica

# CAPÍTULO 1:

## 1 MARCO TEÓRICO.

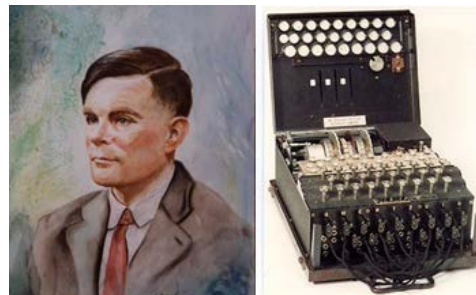
### 1.1 Generalidades

La mecatrónica y su ingeniería está compuesta por la unión de las disciplinas: mecánica, electrónica, control e informática (*Figura 1.1*); enfocándose en el diseño y desarrollo de productos que involucren estas áreas. Con lo que se busca crear máquinas más complejas y precisas para facilitar las actividades del ser humano



**FIGURA 1.1: RAMAS QUE CONFORMAN LA INGENIERÍA MECATRÓNICA.**

Esta rama tiene sus inicios en los estudios realizados por Alan Mathison Turing (*Figura 1.2*) en el área de la cibernética en 1936, quien es considerado como uno de los padres de la ciencia de la computación y precursor de la informática moderna, proporcionando una influyente formalización de los conceptos de algoritmo y computación.



**FIGURA 1.2: ALAN TURING Y SU MÁQUINA "ENIGMA" (MÁQUINA DE CIFRADO)**

Seguido por las máquinas de control numérico desarrolladas inicialmente en 1946 por Devol y autómatas programables, desarrollados por Bedford Associates en 1968.

En 1969 un ingeniero de la empresa japonesa Yaskawa Electric Co., Tetsuro Mori, acuña el término Mecatrónica, recibiendo el derecho de marca en 1971 liberándolo 1982.

Para los años 70 la mecatrónica se debía principalmente a la tecnología de servomecanismos, usados en puertas automáticas, cámaras autoenfoco, máquinas de autoservicio, naciendo una orientación hacia métodos avanzados de control.

Hacia los años 80 con la tecnología de la información, se comenzaron a incluir microprocesadores en sistemas mecánicos para mejorar su desempeño, logrando que las máquinas de control numérico y robots se hicieran más compactas y las aplicaciones en el área automotriz se extendieran.

En los años 90 con el apareamiento de tecnología en comunicaciones, se crearon nuevas máquinas capaces de conectarse a grandes redes, haciendo posible la operación remota de las mismas.

En la actualidad se están usando micro-sensores y micro-actuadores, los cuales han servido para miniaturizar aún más las máquinas modernas, volviéndolas más versátiles y portables con bajo costo de energía.

## 1.2 Sistema didáctico robótico móvil.

La Facultad de Ingeniería en Mecánica y Ciencias de la Producción (FIMCP-ESPOL) en el equipamiento de su nuevo laboratorio de mecatrónica, adquirió, un sistema robótico denominado Robotino® perteneciente a la marca FESTO® (veáse figura 1.3), el cual ha sido desarrollado netamente con el propósito de la formación y perfeccionamiento profesional en el área de automatización, robótica y tecnología.



**FIGURA 1.3: SISTEMA DIDÁCTICO ROBÓTICO MÓVIL**

Con este sistema, se persigue que cada estudiante se familiarice con las diferentes clases de sensores y actuadores que éste contiene. Al igual de su uso y aplicaciones prácticas que se pueden hacer con cada uno de ellos.



Igualmente se busca que el estudiante aprenda y perfeccione tareas profesionales como:

- Puesta en funcionamiento y regulación de un sistema mecatrónico.
- Adquisición, factores de escala y análisis de datos ofrecidos por los detectores.
- Accionamiento eléctrico de motores.
- Técnicas de calibración de actuadores eléctricos.
- Programación de un sistema robótico móvil.
- Introducción al tema del procesamiento de imágenes.
- Control a través de redes inalámbricas.
- Programación para navegación autónoma.

Además el equipo cumple con ciertos criterios técnicos y características para los robots móviles como son:

- Poseer sistemas propios de navegación, orientación, detección y elusión de obstáculos.
- Una fuente de energía propia o independiente.
- Contener detectores y actuadores propios.

Así, cumpliendo con estas condiciones, el sistema permite acogerse a numerosos temas relacionados con la robótica móvil.

Robotino® se propone como un sistema amigable y abierto, donde cada alumno o usuario tiene acceso a sus componentes y tecnología, para ser modificados o dispuesto de la manera más conveniente o requerida, volviéndose divertido. También, al promover la competencia o concurso para establecer maneras más óptimas y/o eficientes en su programación y control, obteniendo soluciones semejantes a las que se utilizan en la industria, ya que está compuesto por elementos que se emplean en la misma.

Cabe destacar que a pesar de constar con componentes vistos en la industria, es compacto, versátil y fácil de transportar, ocupando poco espacio de almacenamiento.

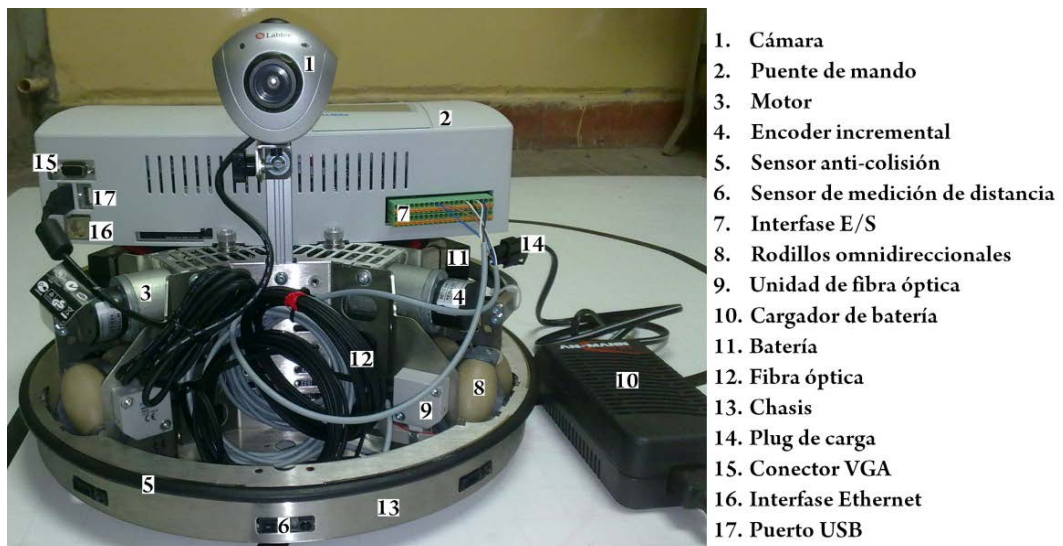
Del mismo modo, promueve el entendimiento y raciocinio del funcionamiento real de cada uno de los componentes y sus posibles aplicaciones en diferentes áreas mediante la experimentación, aplicando la teoría relacionada con sistemas mecatrónicos.

### 1.3 Reconocimiento de los componentes del sistema didáctico robótico móvil

El S.D.Ro.M. Robotino® al igual que robots industriales incluye diversos sistemas parciales o componentes que tienen funciones diversas e independientes.

En esta sección se dará una breve descripción de las partes principales del S.D.Ro.M. Robotino®, su funcionamiento y aplicaciones posibles en la industria.

Para empezar en la *Figura 1.4* se muestra los componentes del S.D.Ro.M. Robotino® y donde están situados para su rápido reconocimiento.



**FIGURA 1.4: PRINCIPALES COMPONENTES DEL S.D.RO.M. ROBOTINO®**

### **Cámara (webcam)**

Esta es una pequeña cámara digital conectada al S.D.Ro.M. Robotino® por medio de una conexión USB, la cual puede capturar imágenes y transmitir las en tiempo real. También su altura puede ser regulada al igual que su inclinación

El S.D.Ro.M. Robotino® usa una cámara (*Figura 1.5*) Labtec® WebCam Pro con un sensor VGA CMOS para videos y fotos de alta resolución, captura videos de hasta 640x480 píxeles hasta 30 fps, lente de enfoque manual y micrófono.



**FIGURA 1.5: CÁMARA DIGITAL USADA POR S.D.RO.M. ROBOTINO®**

Esta será utilizada para las tareas de reconocimiento de colores y procesamiento de imágenes. De la misma manera los resultados pueden utilizarse para señalar objetos con precisión, así como para el seguimiento de rutas trazadas y cosas.

### **Puente de mando o unidad de control**

Este componente también puede ser considerado como el “cerebro electrónico” del robot (*Figura 1.6*) y es aquí donde se encuentra los componentes más sensibles del sistema, controladores, el módulo E/S, interfaces y el módulo de memoria (tarjeta compact flash) en la cual está su sistema operativo Linux con kernel en tiempo real sistema y APIC++ para controlar a Robotino®.



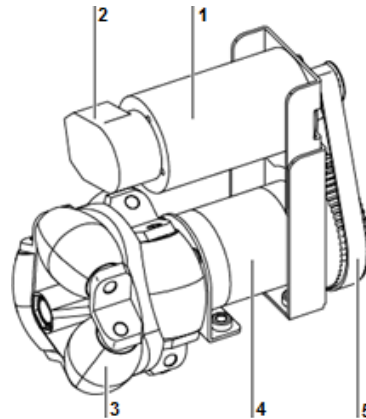
**FIGURA 1.6: "PUENTE DE MANDO" DE ROBOTINO®**

La unidad de control consta de un procesador PC 104, compatible con MOPS1cdVE, de 300Mhz, punto de acceso LAN inalámbrico y las interfaces: Ethernet, 2 USB, y VGA; a las que se les puede conectar un teclado, un ratón y una pantalla, para acceder al sistema operativo y a la librería C++ sin necesidad de una PC

### **Unidades de accionamiento**

Robotino® tres unidades de accionamiento “omnidireccional” independientes, montados de manera radial formando un ángulo de 120º entre sí. Estas unidades (*Figura 1.7*) están compuestas de: (1) motor DC,

(2)encoder incremental, (3) rodillos omnidireccionales, (4) Reductor (relación 16:1), y (5) una correa dentada.



**FIGURA 1.7: UNIDAD DE ACCIONAMIENTO DE ROBOTINO®**

- (1) Motor DC:

El motor DC de escobillas, tiene un sistema electrónico de regulación integrado y sus características se presentan en la tabla 1.1:

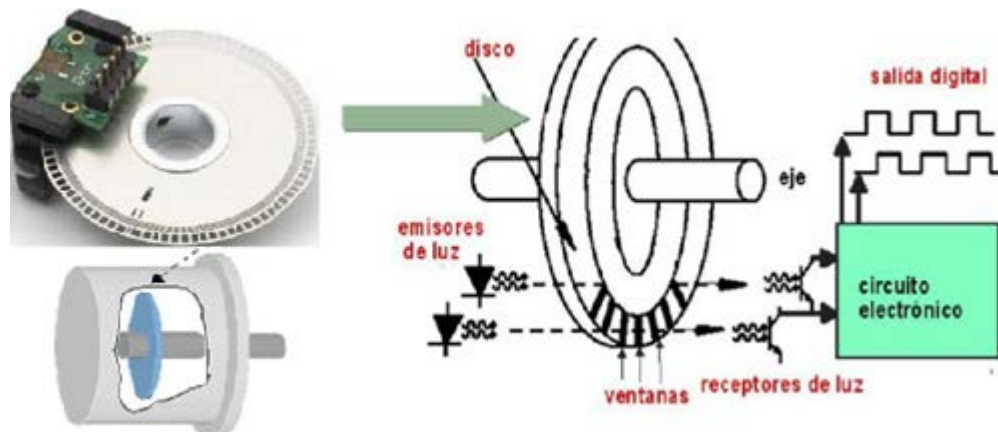
**TABLA 1.1:  
MOTOR DC Y ESPECIFICACIONES TÉCNICAS.**

Motor DC (GR 42x25)	Unidad de medida	
Tensión nominal	V DC	24
Velocidad nominal	RPM	3600
Par nominal	Ncm	3,8
Intensidad nominal	A	0,9
Par de arranque	Ncm	20
Intensidad de arranque	A	4
Velocidad sin carga	RPM	4200
Intensidad sin carga	A	0,17
Intensidad de desmagnetización	A	6,5
Momento de inercia de la masa	gcm <sup>2</sup>	71
Peso del motor	gr.	390

- (2) Encoder incremental:

Los Encoders son sensores que generan señales digitales en respuesta al movimiento. Están disponibles en dos tipos, uno que responde a la rotación, y el otro al movimiento lineal. Cuando son usados en conjunto con dispositivos mecánicos tales como engranajes, ruedas de medición o flechas de motores, estos pueden ser utilizados para medir movimientos lineales, velocidad y posición.

Robotino® usa un encoder (*Figura 1.8*) de tipo incremental caracterizado porque determina su posición, contando el número de impulsos que se generan cuando un rayo de luz, es atravesado por marcas opacas en la superficie de un disco unido al eje.



**FIGURA 1.8: CARACTERÍSTICAS DE UN ENCODER.**

En el estator hay como mínimo dos pares de foto receptores ópticos, escalados un número entero de pasos más  $\frac{1}{4}$  de paso. Al girar el rotor

genera una señal cuadrada, el escalado hace que las señales tengan un desfase de  $\frac{1}{4}$  de periodo si el rotor gira en un sentido y de  $\frac{3}{4}$  si gira en el sentido contrario, lo que se utiliza para discriminar el sentido de giro y la resolución del encoder depende del número de impulsos por revolución.

Estos sensores actúan como: transductores de retroalimentación para el control de velocidad en motores, sensores para medición, de corte y de posición, entrada para velocidad y controles de rango, entre otros.

Y en la industria estos sensores son encontrados en: dispositivos de control de puertas, robótica, plotter, soldadura ultrasónica, máquinas de ensamblaje, máquinas etiquetadoras, posicionamientos x/y, máquinas taladradoras, equipo médico, entre otro.

- (3) Rodillos omnidireccionales.

Las ruedas omnidireccionales (*Figura 1.9*), además de los movimientos básicos de atrás y adelante, permiten movimientos complicados: movimiento en todas las direcciones, diagonalmente, lateralmente e incluso rotar sobre sí mismo en  $360^\circ$ .





**FIGURA 1.9: RODILLOS OMNIDIRECCIONALES.**

Las ruedas omnidireccionales (de 80 mm de diámetro para esta unidad) nos proporcionan dos grandes ventajas: podemos tener una tracción compuesta por únicamente 3 ruedas y con estas podemos movernos a cualquier parte del campo sin cambiar nuestra orientación

- (4) Reductor (relación 16:1).

Reductor planetario de alta eficiencia y compacto, de acero todos sus componentes internos, cuyas características se presentan en la tabla 1.2

**TABLA 1.2:  
CARACTERÍSTICAS TÉCNICAS DEL REDUCTOR PLANETARIO**

<b>Reductor planetario (PLG 42 S)</b>	
De una sola etapa, Nm	3,5
De una sola etapa, i	4:1 - 8:1
2-etapas, Nm	6
2-etapas, i	16:1 - 64:1
3-etapas, Nm	14
3-etapas, i	100:1 - 512:1

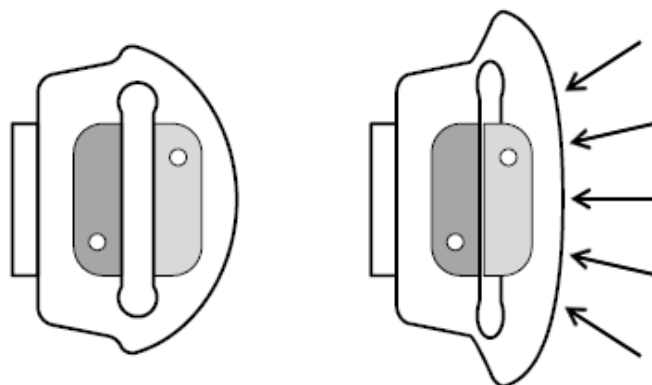
- (5) Correa dentada.

Correa de caucho de dientes que evitan que esta deslice durante la transmisión del movimiento, asegurando así una relación de transmisión constante.

### **Sensor anti-colisión**

Este sensor, que tiene forma de listón, está localizado en la periferia del chasis, es capaz de detectar cuando el S.D.Ro.M. Robotino® ha colisionado contra un objeto u obstáculo.

Consta de un listón (*Figura 1.10*) perfilado de material sintético flexible provisto internamente de una cámara de conmutación, donde se encuentran dos zonas conductoras separadas entre sí.



**FIGURA 1.10: VISTA DE SECCIÓN DEL LISTÓN PROTECTOR**

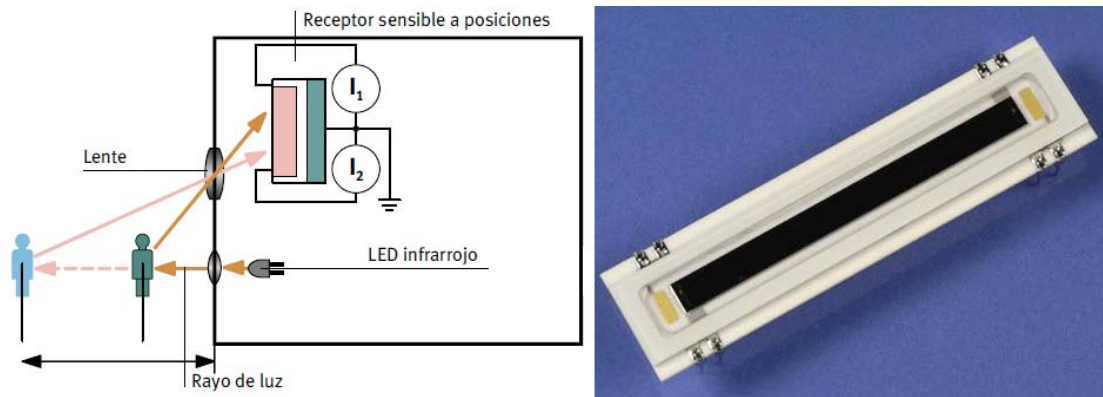
Al aplicar presión en el listón se produce un corto-circuito, emitiendo una señal que es enviada a la unidad de procesamiento. Además el listón

funciona según el principio de corriente de reposo, en el cual una corriente eléctrica fluye constantemente en un circuito eléctrico, pudiendo así detectar una ruptura del cable o destrucción del listón, si la corriente de reposo deja de fluir.

### **Sensor de medición de distancia (Detectores de rayos infrarrojos)**

Este tipo de sensor (*Figura 1.11*) consta de un emisor un receptor y una parte electrónica de procesamiento. El emisor proyecta un haz de luz infrarroja el cual topa un objeto y es reflejado hacia el receptor, considerando que el receptor y el emisor están juntos, se observa que entre rayo emitido, el reflejado y la distancia entre el receptor y el emisor, se forma un triángulo, que posteriormente se utiliza para el cálculo de las distancias

Para esto el receptor es un detector sensible a las posiciones (PSD), que es capaz de diferenciar distintos puntos de incidencia de luz.



**FIGURA 1.11: FUNCIONAMIENTO DETECTOR INFRARROJO (IZQ.);  
DETECTOR PSD (DER.)**

El PSD es un fotodiodo, que está compuesto por una capa sensible a la luz y por una capa metálica, donde se encuentran electrodos metálicos a los extremos. Si un rayo de luz incide en un punto de la capa sensible, se liberan electrones en ese punto, generándose un flujo de corriente hacia los electrodos, además las partes no iluminadas hacen las veces de resistencia.

De esta manera la relación recíproca de la intensidad en los electrodos depende de la relación del punto de incidencia sobre la capa sensible de la luz, siendo todo esto es procesado por la parte electrónica de procesamiento

La relación de las intensidades entre sí es independiente de la cantidad de luz, por lo que la medición de distancias no depende de la capacidad de reflexión y del material del objeto.

Para evitar la interferencia que podría ocasionar la luz dispersa o la luz diurna, el emisor envía los rayos infrarrojos por ciclos, por lo tanto solo se procesarán las señales que correspondan a esos ciclos, ignorando las demás señales recogidas.



**FIGURA 1.12 SENSOR DE DISTANCIA; EMISOR, RECEPTOR Y SISTEMA DE PROCESAMIENTO ESTÁN INCLUIDOS EN LA MISMA UNIDAD.**

Este tipo de sensores (*Figura 1.12*) además de ser usado por el S.D.Ro.M., tienen aplicaciones: en sistemas de asistencia para parqueo, medición de distancia, posición y proximidad, puertas y lavaderos automáticos, etc.

### **Detectores ópticos**

Estos sensores trabajan con el principio de reflexión de luz (roja o infrarroja) por el medio del cual pueden detectar objetos; estos se componen de un emisor y un receptor y normalmente usan diodos luminosos como fuente de luz, debido a que son pequeños, robustos, de gran duración y

permiten una sencilla modulación. Como elementos receptores se utiliza fotodiodos o transistores fotosensibles. Adicionalmente se puede hacer uso de reflectores y fibras ópticas dependiendo de su construcción y aplicación.

Los sensores ópticos (Figura 1.13) del S.D.Ro.M., usan luz roja que tiene la ventaja de poder ser detectados a simple vista, lo que es importante al efectuar el ajuste de los ejes ópticos de los detectores. Además estos sensores hacen del uso de fibra óptica de polímero dándoles mayor versatilidad, si los detectores: ocupan demasiado espacio, se requiere detectar en lugares de difícil acceso, detectar de modo muy preciso la posición de objetos muy pequeños o detectar en zonas con peligro de explosión.

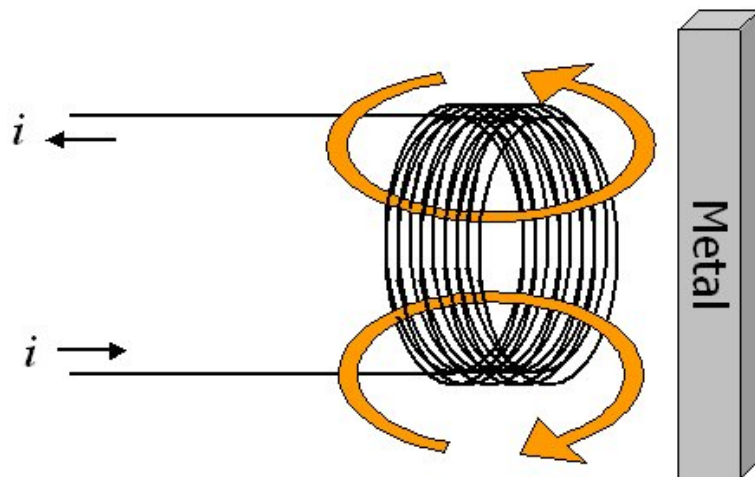


**FIGURA 1.13: SENSOR ÓPTICO Y UNIDAD DE FIBRA ÓPTICA**

### **Sensor inductivo análogo**

Son un tipo especial de sensores que sirven para detectar materiales metálicos ferrosos.

Estos están constituidos por una bobina con un núcleo férnico y capacitor que forman un circuito oscilante. Cuando una corriente circula por el mismo, un campo magnético es generado (en sentido de la regla de la mano derecha). Cuando un metal (*Figura 1. 14*) es acercado al campo magnético generado por el sensor, este es detectado debido a que el devanado induce corrientes de EDDY en el material por detectar. Estas, a su vez, generan un campo magnético que se opone a la bobina del sensor, causando una reducción en la inductancia de la misma, la cual trae aparejado una disminución en la impedancia de esta.



**FIGURA 1.14: FUNCIONAMIENTO DEL SENSOR INDUCTIVO.**

El circuito oscilante podrá generar nuevamente el campo magnético con su amplitud normal. Es en este momento en que el circuito detector

nuevamente detecta este cambio de impedancia y envía una señal a la amplificador de salida para que éste sea quién, restituya el estado de salida del sensor, nuevamente.

El detector inductivo (Figura 1.15) usado por el S.D.Ro.M. se utiliza en los siguientes casos:

- Diferenciación entre tipos de materiales metálicos.
- Medición de distancias.
- Control de operaciones de montaje.

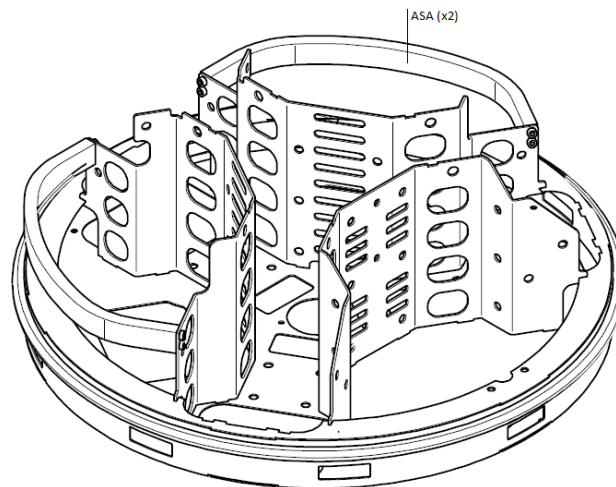


**FIGURA 1.15: DETECTOR INDUCTIVO USADO POR EL S.D.RO.M.**

### **Chasis**

El chasis (*Figura 1.16*) consiste en una plataforma de acero soldada con láser. Es donde se hallan montados las unidades de accionamiento, las baterías recargables, sensores de medición de distancia y el sensor anti-colisión. Y además ofrece un espacio adicional y opciones de montaje para otros añadidos, sensores y / o actuadores.





**FIGURA 1.16: CHASIS DEL S.D.RO.M.**

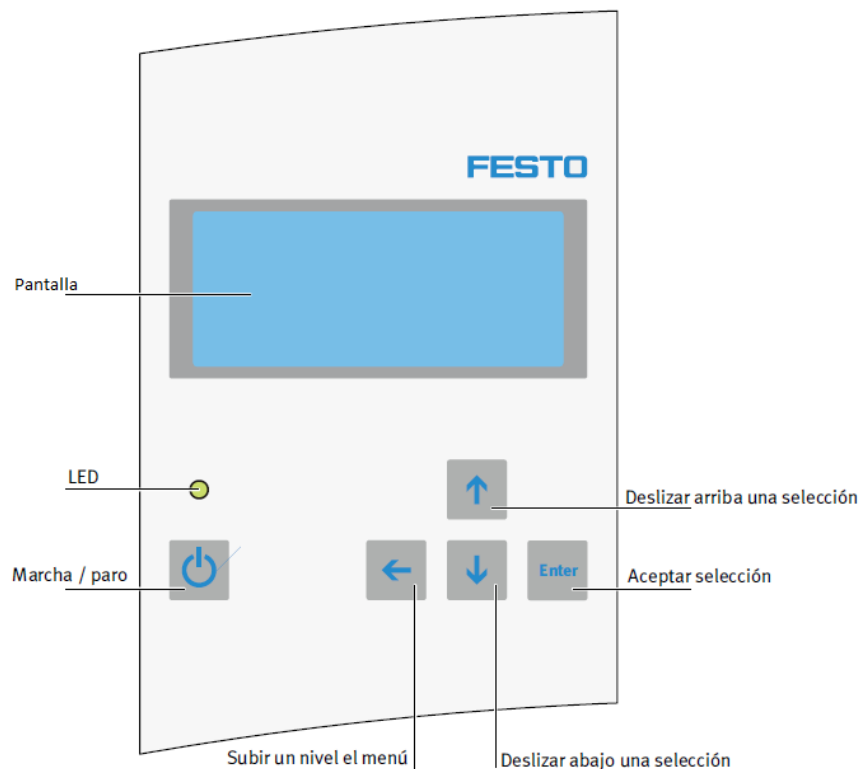
Las baterías (*Figura 1.17*) de S.D.Ro.M. montadas en el chasis son de tipo plomo-gel de 12 voltios de 5 Amperios/hora. Que se cargan de dos en dos, en serie con su respectivo cargador de 24 voltios



**FIGURA 1.17: BATERÍAS PLOMO-GEL USADAS POR S.D.RO.M.**

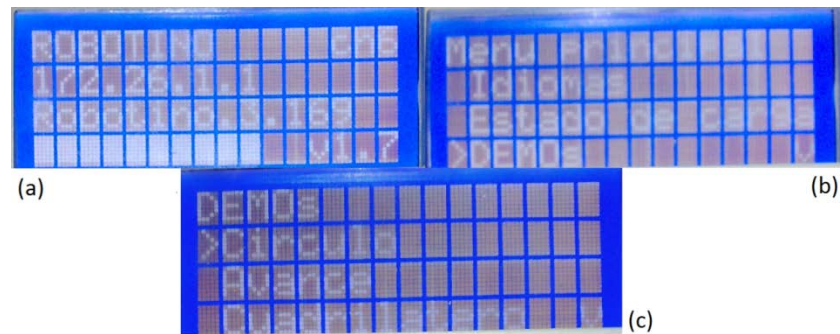
#### 1.4 Uso, control y puesta en marcha del sistema didáctico robótico móvil.

El S.D.Ro.M. Robotino® dispone en el puente de mando, de un teclado de membrana y una pantalla de diodos luminosos (Figura 1.18), donde es posible manejar y revisar varios parámetros del mismo; como: el estado de carga de las baterías, dirección IP de Robotino®, versión de sistema operativo, entre otros.



**FIGURA 1.18: CONTROL DE MANDO MANUAL DE ROBOTINO®**

Además se puede acceder y controlar: programas que hayan sido grabados en la memoria, programas predeterminados y movimientos básicos de Robotino®. Lo que servirá para comprobar el correcto funcionamiento del S.D.Ro.M.



**FIGURA 1.19: (A) PANTALLA PRINCIPAL, (B) PANTALLA MENÚ PRINCIPAL, (C) SUBMENÚ DEMOS**

Para encender el S.D.Ro.M. se debe mantener presionado el botón On/Off hasta que el led encienda. Después de esto se esperará hasta la unidad arranque completamente, la pantalla (*Figura 1.19: (a)*) se encenderá y mostrará: el canal al que se encuentra conectado, su dirección IP, nombre de la red, estado de carga de las baterías y la versión del robot.

Para ingresar a los diferentes sub-menús (*Figura 1.19: (b)*) se debe presionar la tecla Enter y desplazarse con las teclas Arriba/Abajo hasta la selección deseada y presionar Enter de nuevo para ingresar o activar la opción seleccionada (*Figura 1.19: (c)*).

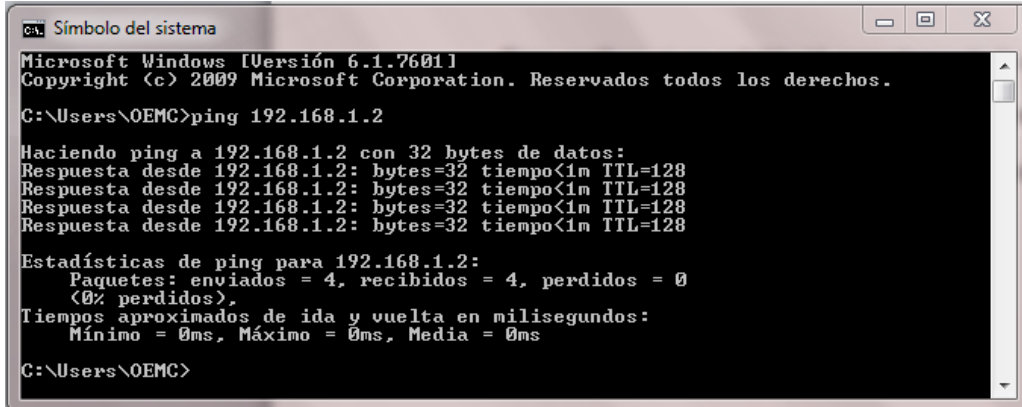
El S.D.Ro.M. activa una red inalámbrica “Wi-fi” una vez encendido, lo que permitirá una conexión remota entre un computador y el robot. Esta conexión se establece yendo al listado de redes inalámbricas detectadas por el

computador y conectándose al red de Robotino® que tendrá el mismo nombre que es presentado en la pantalla del S.D.Ro.M.



**FIGURA 1.20: RED INALÁMBRICA DE ROBOTINO® DETECTADA POR EL COMPUTADOR.**

Para verificar si se estableció correctamente la conexión, para Windows se puede abrir una ventana “Símbolo de Sistema”, escribiendo el comando “ping” seguido de la por un espacio y la dirección IP de Robotino, luego presionar “ENTER”; en una correcta conexión la pantalla se mostrará como en la *Figura 1.21*



```

C:\> Símbolo del sistema
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\OEMC>ping 192.168.1.2

Haciendo ping a 192.168.1.2 con 32 bytes de datos:
Respuesta desde 192.168.1.2: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo<1m TTL=128

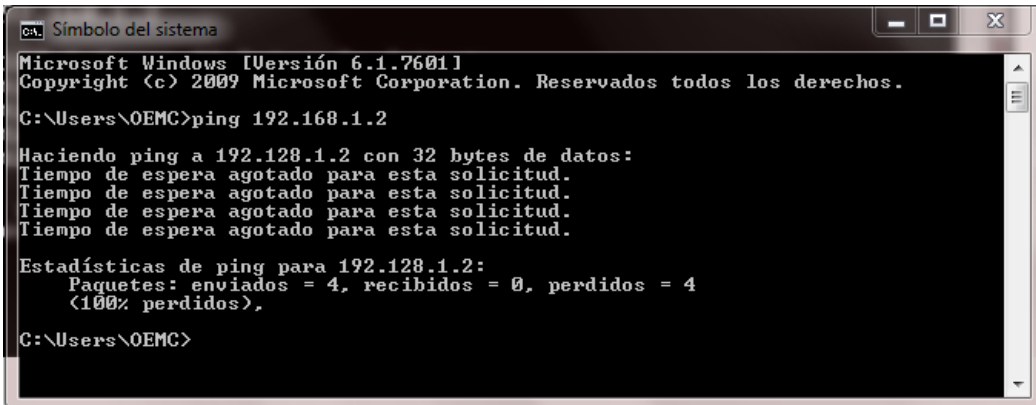
Estadísticas de ping para 192.168.1.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 0ms, Máximo = 0ms, Media = 0ms

C:\Users\OEMC>

```

**FIGURA 1.21: VENTANA "SIMBOLO DEL SISTEMA", PARA VERIFICAR LA CORRECTA CONEXIÓN.**

De no estar correcta la conexión la pantalla se mostrará como en la *Figura 1.22*



```

C:\> Símbolo del sistema
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\OEMC>ping 192.168.1.2

Haciendo ping a 192.128.1.2 con 32 bytes de datos:
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.

Estadísticas de ping para 192.128.1.2:
    Paquetes: enviados = 4, recibidos = 0, perdidos = 4
    (100% perdidos),

C:\Users\OEMC>

```

**FIGURA 1.22: INCORRECTA CONEXIÓN A ROBOTINO®**

También Robotino® presenta dos modos de conexión: AP (Access Point) donde se pueden controlar hasta de 1 a 4 unidades independientemente. Y el modo "Client", que permite trabajar con varias unidades en red, asignándole una dirección IP a cada uno.

Una vez que se ha establecido la conexión, es posible proceder con el uso de los diferentes programas de control y programación del S.D.Ro.M.

### **1.5 Actualización sistema operativo del sistema didáctico robótico móvil.**

Cuando la FIMCP adquirió el S.D.Ro.M. Robotino®, vino con la versión 1.7 en su sistema operativo y con las continuas mejoras en software que se dan con el transcurso del tiempo, Festo®didactic la empresa que fabrica los Robotinos® ofreció una nueva versión (2.xx) del S.O. por lo tanto se comenzó a trabajar en esta actualización, que no solo involucraba el cambio de sistema operativo sino también la modificación de las prácticas establecidas al nuevo programa de control del S.D.Ro.M

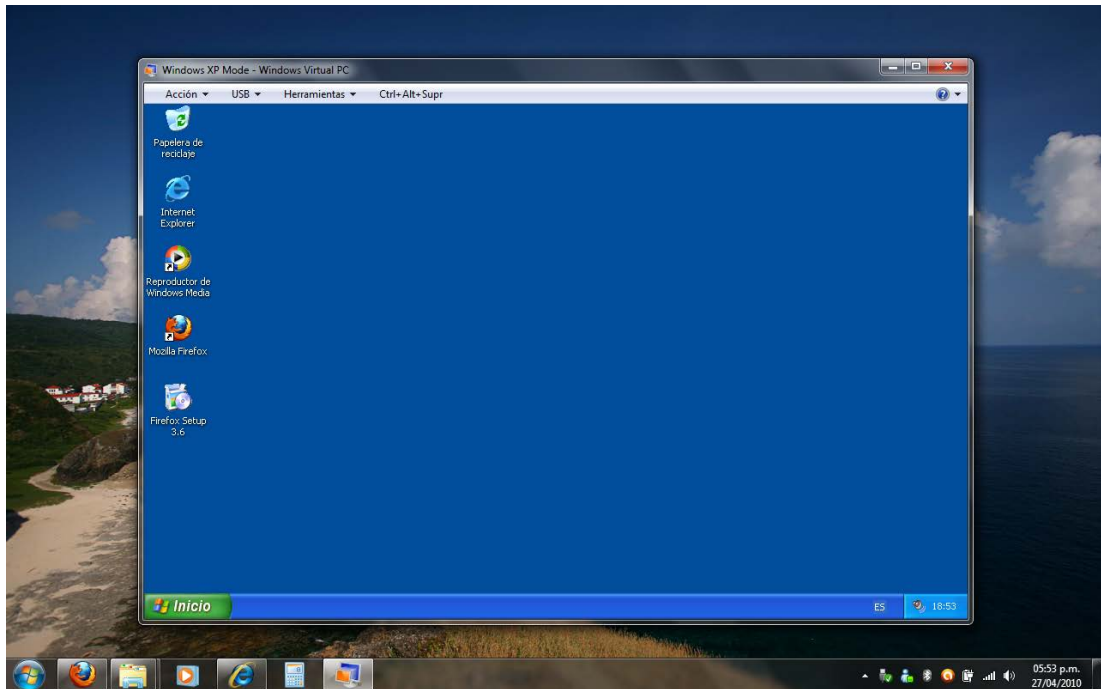
Al comenzar con la actualización se debió cumplir con algunos requisitos antes, entre ellos están:

- Windows XP (no compatible con Win. Vista).
- Nociones en uso de DOS o la ventana de comandos “símbolo del sistema”.
- Tarjeta compact Flash (CF-card, mínimo 256Mb preferible 1Gb), esta tarjeta es incluida con el Robotino® pero si se desea tener los dos sistemas operativos, se puede adquirir otra.
- Un lector de tarjetas CF-card a USB, adquirido para este trabajo
- Descargar el archivo de “imagen” del sistema operativo, actualizada.

- El programa “dd-removable.exe”, que servirá para crear una partición en la CF-card y para transferir el archivo imagen a la CF-card

Como recomendación importante, al momento de hacer la actualización se debe solo conectar el lector de CF-card al computador, además “dd-removable.exe” es programa gratuito con licencia GNU, con lo que NO se garantiza que trabaje correctamente y al momento de que se haga el formato se puede perder información.

Debido a que, en las computadoras del laboratorio se trabaja con Windows7®, antes de que se realice la actualización, se tuvo que activar el “XP Mode” o Modo XP (*Figura 1.23*), en el cual se corre una consola virtual de con Windows XP®, para cumplir con unos de los requisitos antes mencionados.



**FIGURA 1.23: WINDOWS7® CORRIENDO LA CONSOLA VIRTUAL "XP MODE"**

Para acceder al Modo XP lo primero que se necesita es asegurarse que la “Tecnología de Virtualización” esté activada, lo que significa que el CPU debería soportar la virtualización de hardware. Esto puede comprobarse usando la “herramienta de identificación del procesador Intel” o el “AMD-V Technology and Microsoft Hyper-V Compatibility Check”. Si la PC está habilitada para la virtualización de hardware, se puede asegurar de ello usando las opciones de la BIOS.

Otros requisitos para el XP Mode:

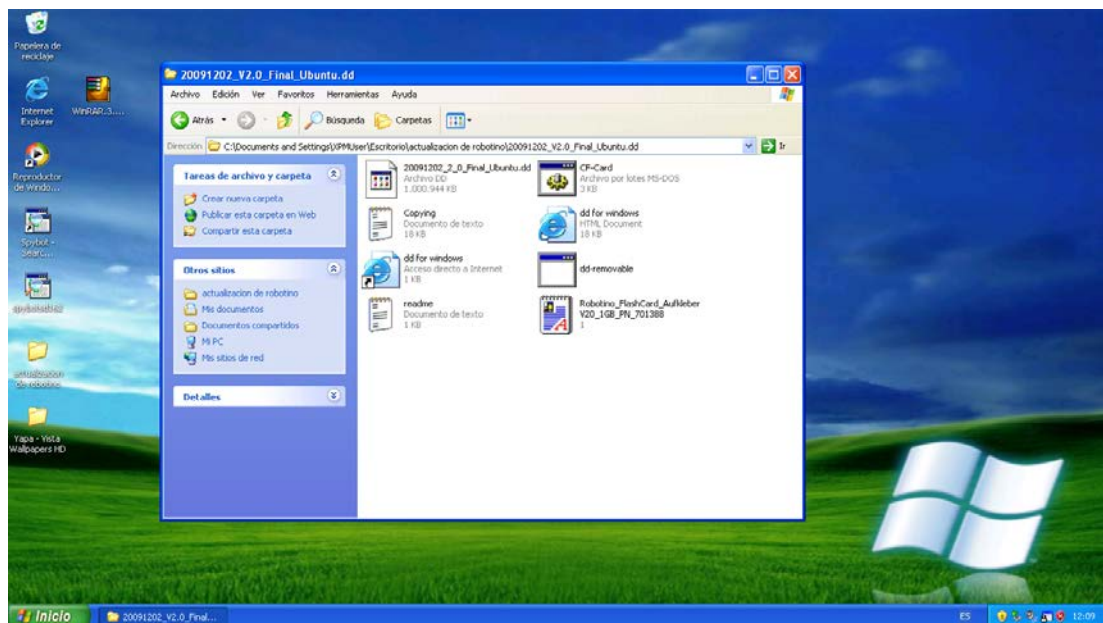
- Procesador 1 GHz 32-bit / 64-bit
- Memoria (RAM) - 1.25 GB requerido, 2 GB memoria recomendada.



- Recomendado 1.5 GB de espacio en el disco duro para el entorno virtual de Windows.
- Windows XP Mode solo está disponible en Windows 7 Enterprise, Windows 7 Professional, y Windows 7 Ultimate.

Una vez que se ha cumplido con los requisitos, se descarga el archivo comprimido que ofrece Festo® para la actualización (<http://www.festo-didactic.com/int-en/services/robotino/robotino-cf-card.htm>) donde se encuentra los archivos necesarios para el proceso.

Para este caso, primero se debe iniciar la consola virtual y correr Windows XP y luego se procede a descomprimir el archivo de actualización (Figura 1.24)



**FIGURA 1.24: ARCHIVO NECESARIOS PARA ACTUALIZACIÓN.**

Se corre el archivo CF-card.bat y se sigue los pasos en el programa (Figura 1.25)

```

C:\WINDOWS\system32\cmd.exe
Copier for Robotino(CD) CompactFlash Card nhe1@de.festo.com 17.11.09
To avoid loss of data read all instructions carefully!
!! disconnect all USB drives, only connect the (USB)-Cardreader !!
press CTRL/C to abort
Insert source CF-card to USB or PCMCIA-cardreader and press any key
Presione una tecla para continuar . . .

C:\WINDOWS\system32\cmd.exe
vawrite dd for windows version 0.5.
Written by John Neuhigin (j@nit.swin.edu.au)
This program is covered by the GPL. See copying.txt for details

NT Block Device Objects
  \Device\Harddisk1\Partition0
  link to \Device\Harddisk1\DR10
  Removable media other than Floppy. Block size = 512
  size is 1024966656 bytes

above list shows all removable drives connected to your PC.
the cf-card in your Cardreader / PCMCIA Cards should show as
"Removable media other than Floppy. Block size = 512"

Enter correct Harddisknumber (1/2/3 or 4)

C:\WINDOWS\system32\cmd.exe
the cf-card in your Cardreader / PCMCIA Cards should show as
"Removable media other than floppy. Block size = 512"

Enter correct Harddisknumber (1/2/3 or 4)

writing process takes about 4 minutes - please wait -
counting up to 1,024,966,656bytes / "244+1 records"

starting time:
12:15
vawrite dd for windows version 0.5.
Written by John Neuhigin (j@nit.swin.edu.au)
This program is covered by the GPL. See copying.txt for details
Output file does not match device filter removable
dd will not continue
//
//
DONE, please remove CF-Card and label it with 20091202_2_0_Final_Ubuntu.dd
Presione una tecla para continuar . . .

C:\WINDOWS\system32\cmd.exe
vawrite dd for windows version 0.5.
Written by John Neuhigin (j@nit.swin.edu.au)
This program is covered by the GPL. See copying.txt for details

NT Block Device Objects
  \Device\Harddisk1\Partition0
  link to \Device\Harddisk1\DR20
  Removable media other than Floppy. Block size = 512
  size is 1024966656 bytes

above list shows all removable drives connected to your PC.
the cf-card in your Cardreader / PCMCIA Cards should show as
"Removable media other than Floppy. Block size = 512"

Enter correct Harddisknumber (1/2/3 or 4)

writing process takes about 4 minutes - please wait -
counting up to 1,024,966,656bytes / "244+1 records"

starting time:
12:20
vawrite dd for windows version 0.5.
Written by John Neuhigin (j@nit.swin.edu.au)
This program is covered by the GPL. See copying.txt for details
33.554.432
  
```

**FIGURA 1.25: PASOS A SEGUIR POR EL PROGRAMA DE ACTUALIZACIÓN.**

Ya terminados todos los procesos del programa la actualización fue hecha y se puede contar con las diferentes mejoras, entre las que encontramos:

- Tarjeta Compact Flash con Ubuntu Linux.
- Soporte adicional para el uso de varias cámaras Web USB- (driver UVC/Video4Linux2)
- Soporte del sensor odométrico "Giroscopio" y Sensor NorthStar NS2
- Acceso a través de www / telnet / ftp / ssh / RobotinoView.
- Opción de seguridad "Para Choques on / off" en el menú Robotino® en caso de choque, los motores se detienen por 1seg. La velocidad de los motores está limitada a un máximo de 2000 U/m.

- Función odométrica para la medición de distancia.
- Posibilidad de elegir entre dirección IP estática o DHCP (dirección IP recibida del router)
- Ejecución de programas de Robotino-View directamente desde Robotino®.
- RobotinoSIM: ponga a prueba su programa de simulación.

También cambia el programa Robotinoview de su versión 1.x a la 2.x, manejándose una manera distinta de programación, la cual será explicada en detalle en el capítulo 2 de este trabajo.

## **CAPÍTULO 2:**

# **2 SOFTWARE DE COMUNICACIÓN Y CONTROL DEL SISTEMA DIDÁCTICO ROBÓTICO MÓVIL.**

El S.D.Ro.M. Robotino®, trae consigo un software llamado Robotinoview®, mediante el cual puede ser programado y controlado, pero

para usuarios más experimentados existen varios lenguajes de programación y software, como son: C, C++, Java, .NET, MatLab®, Simulink, Labview y Microsoft Robotics Developer Studio, con los que también, se puede lograr la comunicación y control de la unidad robótica

Este trabajo apuntará a enseñar el uso general y programación en Robotinoview® (versión 2.x) y Matlab® 7, que es un programa ampliamente usado por la ESPOL

## **2.1 Robotino®view 2.**

La actualización del sistema operativo del S.D.Ro.M., trajo consigo una nueva versión del programa de comunicación y control, la cual se denomina Robotino®view 2

Esta nueva versión contiene cualidades importantes como: actualizaciones automáticas, ejecución de programas en esquemas Grafset, control simultáneo de varios Robotinos, nuevos bloques funcionales, ejecución de programas de Robotino®view 2 directamente desde el S.D.Ro.M. Robotino®

Además el programa de control de secuencia es reemplazado por un programa de control “real”, conocido de la programación de PLC según DIN EN 61131.

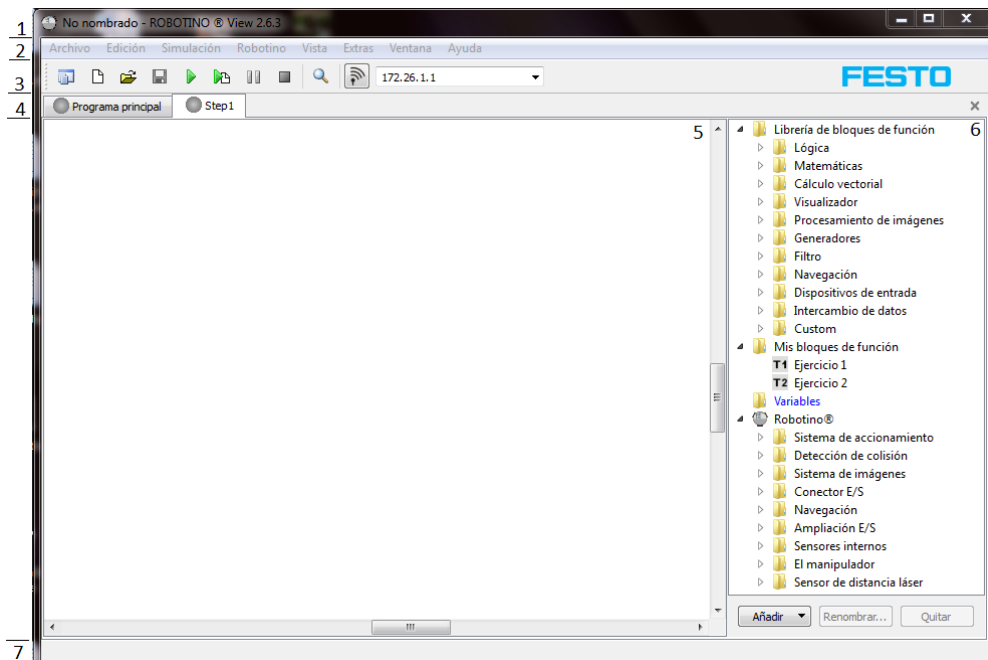
Los subprogramas pueden ser reutilizados dentro del programa principal y pueden ser importados desde diferentes proyectos.

También se puede diseñar e implementar bloques de función personalizados que se hayan cargado en la librería de bloques de función, así como dispositivos personalizados y cargarlos como plugin en el administrador de dispositivos.

### **2.1.1 Generalidades de Robotino®view 2.**

Robotino®view ofrece una interface o espacio de trabajo amigable con el usuario, que permite una rápida familiarización del mismo.

Al correr el programa, éste abre un espacio vacío para iniciar un proyecto nuevo, listo para comenzar a trabajar en la programación y control de la unidad. La *Figura 2.1* muestra una impresión de pantalla de Robotino®view



**FIGURA 2.1: INTERFACE DE USUARIO DE ROBOTINO®VIEW: (1) BARRA DE TÍTULO; (2) BARRA DE MENÚ; (3) BARRA DE HERRAMIENTAS; (4) PESTAÑAS DE SELECCIÓN DE PROGRAMAS; (5) ESPACIO DE TRABAJO DEL SUBPROGRAMA; (6) LIBRERÍA DE BLOQUES DE FUNCIONES; (7) BARRA DE ESTADO**

Para una mejor utilización de Robotino®view 2 es importante definir la siguiente terminología:

Boques de función: Unidad funcional mínima de la que puede estar formado un subprograma. Conectando varios bloques de función, es posible obtener movimientos complejos del Robotino®.

Subprograma: Los bloques de función están interconectados en un subprograma.

Programa principal: Programa de control escrito como diagrama de bloques secuencial que une los subprogramas.

Proyecto: Un proyecto consiste en un programa principal y varios subprogramas. Los proyectos pueden guardarse y cargarse.

Red: Los bloques de función se unen mediante una o varias redes (conexiones o líneas de conexión).

Punto de red o nodo: Los puntos de red permiten la estructuración y representación gráfica de una red. De un punto de red puede iniciarse una nueva sub-red.


Una vez comprendidos y asimilados estos términos se podrá adentrar a la utilización del programa en sí. En esta sección se aprenderán las nociones básicas de Robotino®view, la cual será una ayuda a la familiarización de la programación de Robotino®, el uso del programa y la comunicación entre el PC y la unidad.

### **Establecer conexión entre Robotino®view y el S.D.Ro.M.**

Como se explicó en la sección 1.4 es posible establecer una conexión Wi-Fi entre el S.D.Ro.M. y un Pc , por la cual el programa Robotino®view, hace uso de la misma para el control de la unidad.



Para realizar esto hay que dirigirse a la “Barra de Herramientas” (*Figura 2.1 (3)*) y en la barra de direcciones se escribe la dirección IP del Robotino® que



se desea controlar luego se dará “click” al icono  de conexión, el cual mostrara 3 estados: gris: sin conexión, anaranjado: si el Robotino® que se quiere controlar ya está conectado a otro PC o tiene problemas de conexión y verde cuando la conexión se realizó con éxito

### **Crear un proyecto nuevo o cargar uno ya existente**

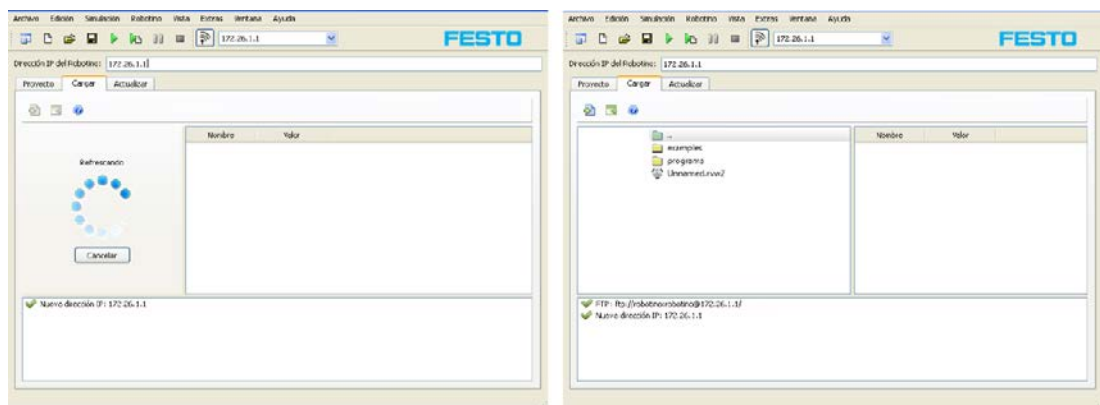
Para crear un proyecto nuevo simplemente habrá que dirigirse a la ficha archivo en la barra de menú (*Figura 2.1 (2)*) y dar click a la opción “nuevo”; de igual manera para abrir un proyecto ya existente, se irá a la ficha archivo y se elige la opción “abrir”

También se puede acceder a estas opciones por medio de la barra de herramientas (*Figura 2.1 (3)*), dando click al icono  para crear un nuevo proyecto, o al icono  para abrir uno ya existente. Cada proyecto que se crea, tendrá la extensión “.rvw2”

### **Exploración de ROBOTINO®**

Esta nueva versión del S.D.Ro.M. ROBOTINO® trae consigo instalados un servidor Telnet y FTP en el sistema Linux. Los cuales son usados para visualizar los archivos de Robotino® y para cargar o subir

proyectos en la unidad. Para acceder a esta función se va a la “Barra de menú”, haciendo “click” en la ficha “Robotino” y se elige la alternativa “Cargar”. Se abrirá una ventana (Figura 2.2) de exploración donde se podrán ver los archivos que se encuentran guardados en el S.D.Ro.M.

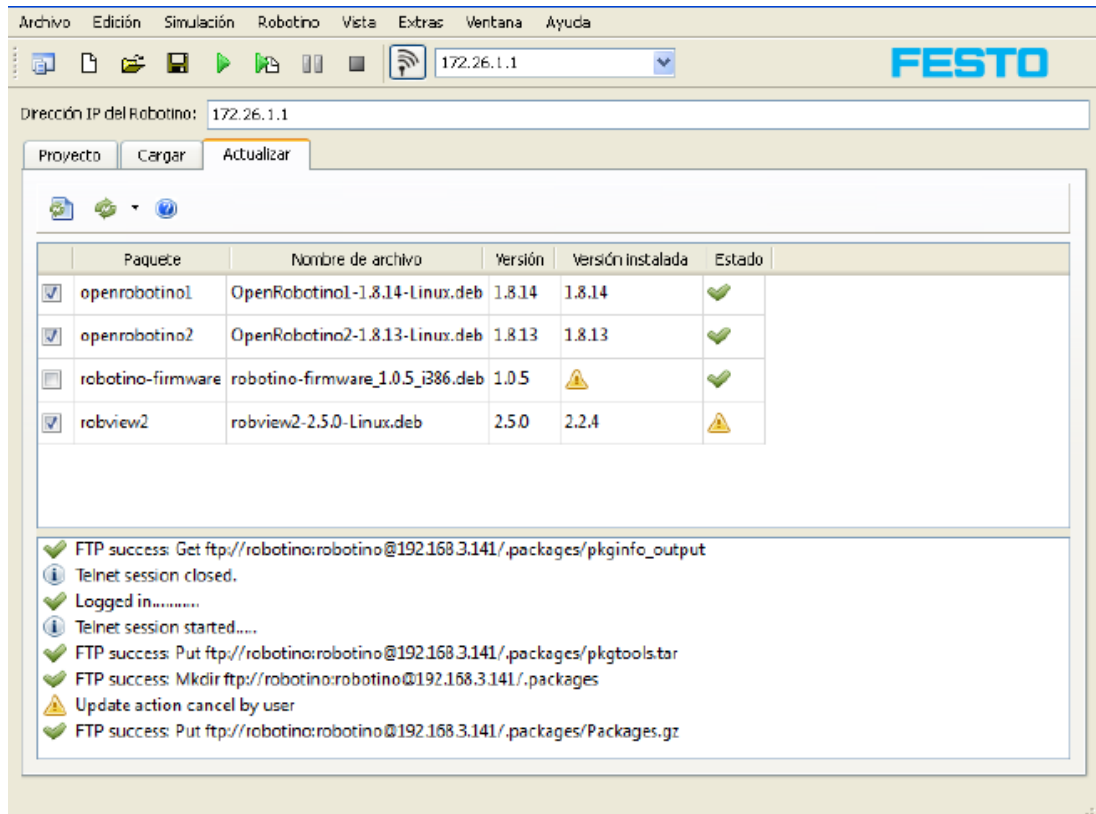


**FIGURA 2.2: VENTANA PARA VISUALIZAR Y CARGAR PROYECTOS A ROBOTINO®**

Desde aquí, puede guardar y ejecutar un proyecto directamente desde Robotino®

### **Actualización de Robotino®**

Otras de las mejoras en Robotino®View, es la posibilidad de actualizar los paquetes de Linux instalados en Robotino® desde el mismo programa. Esta función la se encuentra en: Barra de menú→ Robotino→ Actualizar Software. Se abrirá una pantalla de actualización con todos los paquetes disponibles para actualizar el programa y/o la unidad



**FIGURA 2.3: VENTANA DE ACTUALIZACIÓN**

## 2.1.2 Introducción a la programación del sistema didáctico robótico móvil en Robotino®view 2

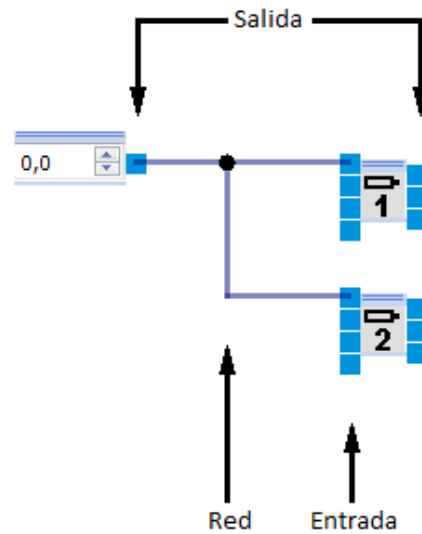
Tras crear un proyecto nuevo o cargar uno ya existente se puede elaborar un programa de control propio o modificarlo.

La programación en Robotino®view 2 es gráfica, usa diagramas de flujo y bloques de función para ingresar los comandos y dar las diferentes instrucciones para el control de la unidad

Estos bloques de función se encuentran en la “Librería de bloques de funciones” (*Figura 2.1 (6)*), situada a la derecha de la ventana del programa, además están enlistados y organizados en directorios y subdirectorios según su aplicación.

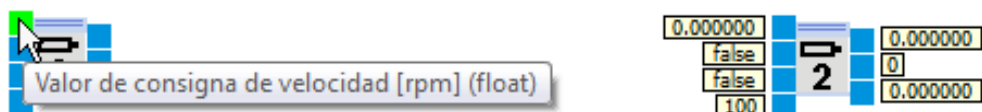
Al trabajar en un proyecto, se crean subprogramas donde se ingresan los bloques de funciones antes mencionados. Para esto, y según sea la necesidad, habrá que ir a la librería y elegir entre los directorios, el bloque de función deseado. Teniendo la elección hecha, simplemente con “click” sostenido se arrastra el bloque hasta el espacio de trabajo del subprograma (*Figura 2.1 (5)*) y estará listo para ser utilizado.

Dependiendo del bloque de función, estos podrán tener “datos de entrada” y/o “datos de salida”, los cuales pueden interconectarse a través de líneas llamadas “red (network)”. Una red siempre está conectada a la salida de un bloque y por lo menos a una entrada de otro.



**FIGURA 2.4: BLOQUES DE FUNCIÓN**

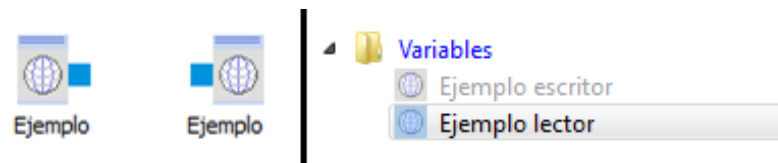
Si se desea eliminar una línea solo se da “click” encima de ella y se pulsa la tecla “suprimir” o para borrar un punto de red o nodo, se da “click derecho” y se elige eliminar. Además para conocer que representa cada una de las entradas y salidas se pasa el puntero del ratón (mouse) sobre lo que se desea conocer y se activará el cuadro de ayuda y para conocer el valor actual de esa variable se pulsa la combinación “Control+D” (Figura 2.5).



**FIGURA 2.5: VALORES DE CONSIGNA Y PRESENTACIÓN DE VALORES ACTUALES DEL BLOQUE DE FUNCIÓN**

El programa también hace uso de variables globales (Figura 2.6) que pueden leerse y escribirse en todos los subprogramas de un proyecto y

además en el programa principal pueden utilizarse como condiciones de transición. En la vista del programa principal, el administrador de variables se halla en el lado derecho (*Figura 2. 6*). Allí es posible añadir, quitar y renombrar variables, así como asignarles valores iniciales. Estas variables, por el momento, solo almacenan números reales (float). En la vista de subprograma para crearla debemos ir a la librería y buscamos el directorio “variables”, damos “click derecho”, seleccionamos la opción “añadir”, ingresamos el nombre de la variable y ya estará lista para ser usada como un bloque de función. Después de creadas, se dispone de un bloque escritor y bloques lectores de esta variable en la librería de bloques de función.



**FIGURA 2.6: VARIABLES GLOBALES EN ROBOTINO®VIEW**

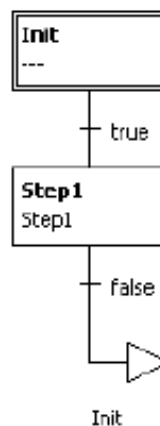
### 2.1.2.1 Ejemplos de aplicación

En esta sección se realizarán varios programas de control sencillos con ramales alternativos y se afianzará aún más la programación con Robotino®view, la cual será reforzada con las prácticas de laboratorio que se expondrán en el siguiente capítulo

### Ejemplo 1

El primer ejemplo que es tratado en esta sección tendrá como objetivo que los tres motores giren uno a continuación del otro, a una velocidad de 100 r.p.m. por 10 segundos cada vez, y luego se detendrá el programa.

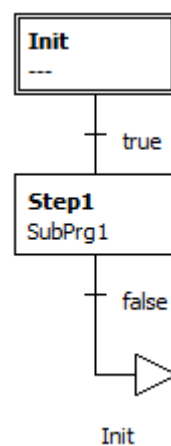
Primero se abre un nuevo proyecto y se observa que ya se encuentran abiertas dos pestañas de programas (*Figura 2.1 (4)*): “Programa principal” y la otra “Step1” o paso uno.



**FIGURA 2.7: CONTENIDO DE LA PESTAÑA "PRINCIPAL"**

En la pestaña “Principal” se presenta un resumen y la secuencia de instrucciones completa del programa que se esté creando, en esta pestaña están el principio y final del mismo.

Como se ve en la Figura 2.7 se muestra el nombre de la pestaña que se abrió con el nuevo proyecto “Step 1” en el primer paso (step 1) que dará el programa. Con esto último se debe tener cuidado, pues se puede prestar para confusiones, se debe interpretar de maneras diferentes: el nombre del subprograma y paso de programa principal; y es por esta razón que se cambiará el nombre del subprograma, situándose sobre la pestaña “Step1” y dando “click derecho”, se escogerá la opción renombrar y se colocará el nombre que se desee, en el ejemplo se lo cambiará a “SubPrg1”, quedando el contenido de la pestaña “Pestaña principal” de la siguiente manera:



**FIGURA 2.8: CONTENIDO DE LA PESTAÑA PRINCIPAL MODIFICANDO EL NOMBRE DEL SUBPROGRAMA**

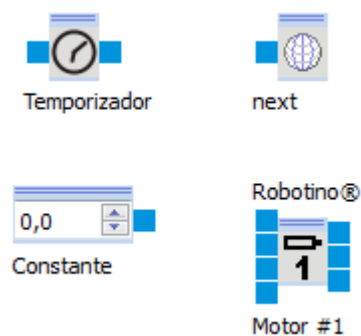
Una vez entendido lo anterior se comenzará con la creación del programa. Damos “click” en la pestaña “SubPrg1”, para abrir el espacio de trabajo, y halamos los bloques de función que se necesiten.

Para este programa se usará:



- Bloque: “Motor 1” encontrado en el directorio: Robotino®→Sistema de accionamiento.
- Bloque: “Constante”; directorio: Librería de bloques de función→ Generadores
- Bloque: “Temporizador”, directorio: Librería de bloques de función→ Generadores
- Además se necesitará crear una variable de tipo “float” escritor, la cual se nombrará como “next”

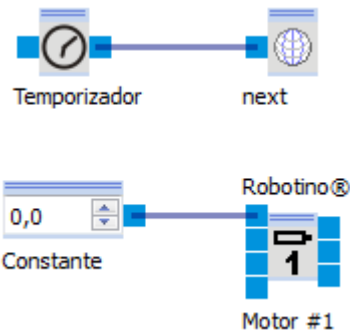
Al final el espacio de trabajo quedará de la siguiente manera:



**FIGURA 2.9: EJEMPLO, USO DE BLOQUES DE FUNCIONES Y ESPACIO DE TRABAJO.**

Después se crearán las conexiones o redes entre los bloques, esto se hace uniéndolo la salida de un bloque a la entrada de otro, lo que servirá para asignar valores iniciales a los bloques de función.


En el ejemplo se unirá la salida del Bloque “constante” con la entrada del Bloque “motor 1” para definir el valor de velocidad en r.p.m. del motor. Lo mismo se hará con los bloques “Temporizador” y “Next”

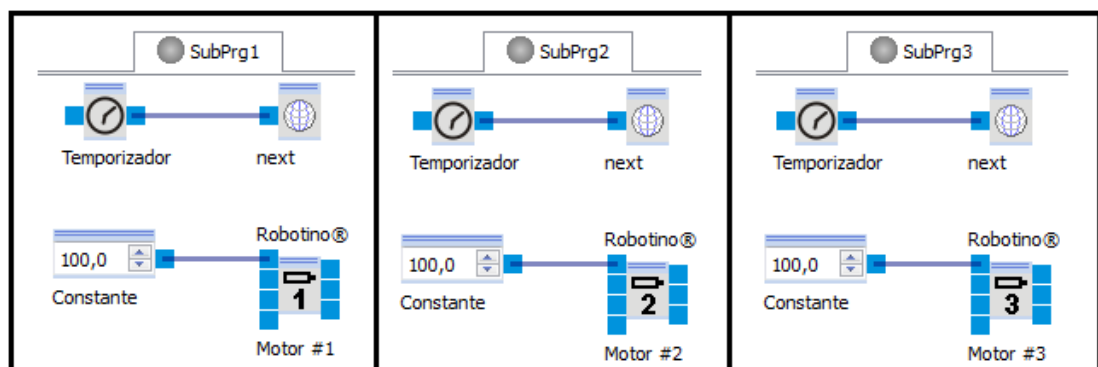


**FIGURA 2.10: CREACIÓN DE REDES O CONEXIONES Y ASIGNACIÓN DE VALORES A BLOQUES DE FUNCIONES PARA EJEMPLO DE APLICACIÓN**


El bloque temporizador contará el tiempo transcurrido en milisegundos durante la ejecución del subprograma y asignará el valor numérico de éste a la variable “next” continuamente, así mediante la variable global se podrá establecer condiciones para dar la continuación a un siguiente subprograma o terminación del programa principal.

Como se dijo al iniciar este ejemplo los requisitos son que cada motor girará a 100 rpm por 10 segundos uno a continuación del otro, por lo tanto se irá a la pestaña “SubPrg1” y se ingresa el valor de 100 en el bloque “Constante” y se cumplirá con el primer requisito.

También se crearán dos subprogramas más, haciendo click en el icono  con los nombres: “SubPrg2” y “SubPrg3”, donde se colocará el mismo contenido de “SubPrg1” con la excepción del bloque Motor que será reemplazado por los restantes, quedando como se muestra a continuación:

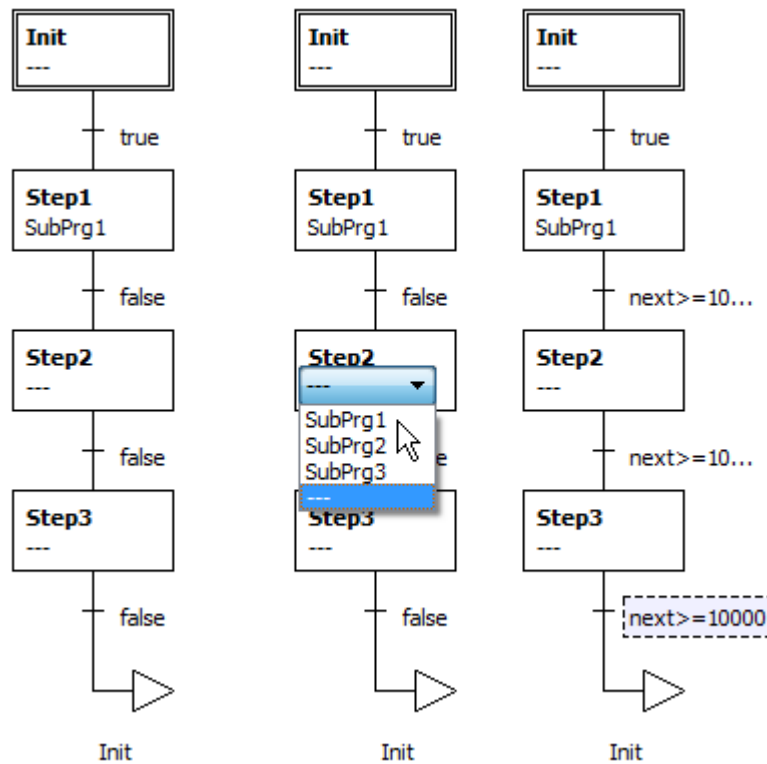


**FIGURA 2.11: CREACIÓN DE SUBPROGRAMAS PARA EJEMPLO DE APLICACIÓN**

Lo siguiente que se hará, es ir al Programa principal, se crearán dos pasos adicionales seleccionando el recuadro del “Step1” o primer paso y después daremos click en el icono  para crear más pasos. Luego se designará un subprograma a cada paso dando click en “---” que aparece dentro de cada recuadro de los pasos

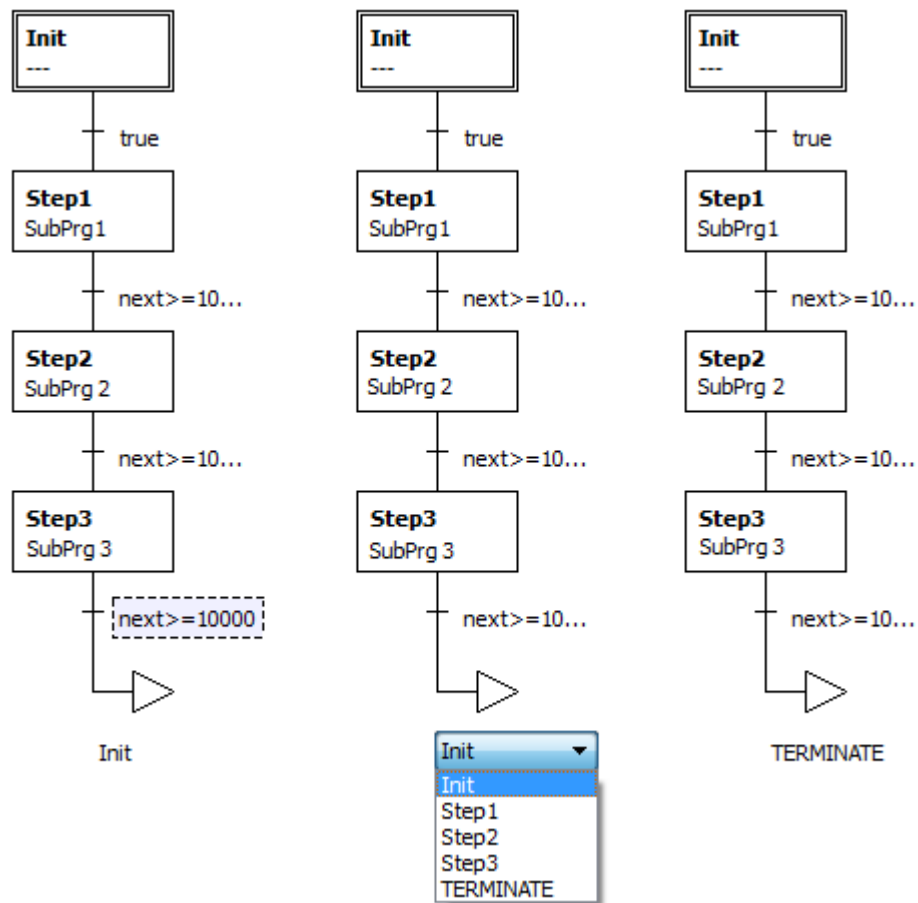
Además se establecerán las condiciones de transición para cada uno de los subprogramas haciendo click en la palabra “false” que se encuentra debajo de cada recuadro de los pasos y escribiremos “next >=10000”, la última

instrucción indica que cuando la variable “next” llegue o supere el valor de 10000 (1s=1000ms) dará paso al siguiente subprograma.



**FIGURA 2.12: CREACIÓN DE PASOS Y CONDICIONES DE TRANSICIÓN PARA EL EJEMPLO DE APLICACIÓN**

Con esto, el programa está casi listo, solo que si se lo deja de esa manera el programa será de tipo cíclico; fijándose en la *Figura 2.12* al final de la programación se encuentra la palabra “Init”, por lo tanto dando click en la misma se podrá elegir la continuación o final del programa, en este ejemplo lo que se desea es que, el programa se detenga, por lo cual se selecciona “TERMINATE”: Quedando finalmente como se muestra en la *Figura 2.13*:



**FIGURA 2.13: PROGRAMA PRINCIPAL TERMINADO PARA EL EJEMPLO DE APLICACIÓN**

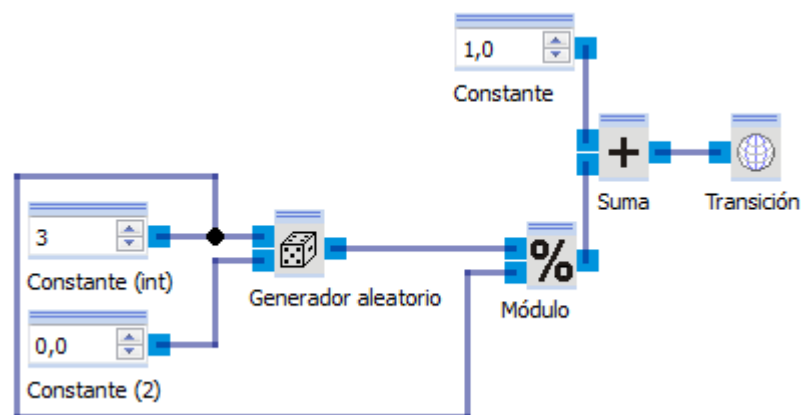
## Ejemplo 2

Como ya se sabe las variables globales se pueden usar para dar paso al siguiente subprograma, en este ejemplo se verá como crear ramales paralelos o alternativos y definir condiciones para trabajar con los mismos.

Para este ejercicio se usará la misma estructura del ejemplo anterior en los subprogramas, pero con la nueva condición de que los motores giren independiente a 100 r.p.m. y de manera aleatoria.

Utilizando la misma estructura de los subprogramas del ejemplo uno (*Figura 2.11*), solo se cambiará la secuencia del programa principal y se adicionará un subprograma de generación de variables aleatorias.

Para crear este subprograma adicional, se dará click en “nuevo subprograma”, se le pondrá de nombre “Aleatorio”, se irá a la librería de funciones y se halarán los bloques de función: tres de “Constante” asignándole valores “0, 1 y 3”, “Generador aleatorio”, “Módulo”, “Suma” y se crea otra variable global con el nombre “transición”. Quedando como se muestra en la figura 2.14:

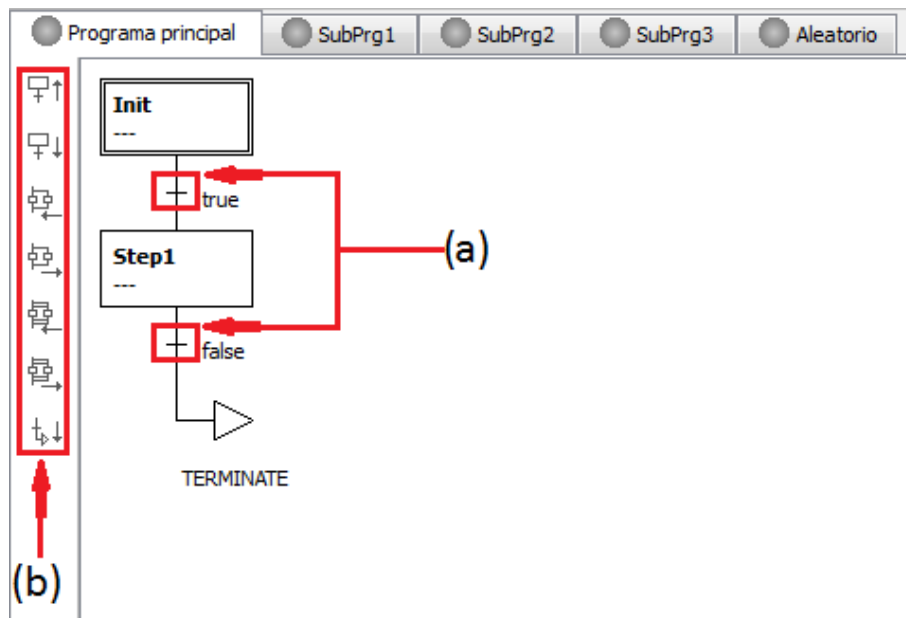


**FIGURA 2.14: BLOQUE ALEATORIO DEL EJEMPLO 2**


Para cumplir con la condición del ejercicio de que los motores girarán de manera aleatoria e independientemente, en el programa principal se crearán

ramales paralelos para que el subprograma “aleatorio”, asigne con que paso se continúa.

Esto se realiza yendo a la pestaña “Programa principal”, dando “click” en las líneas horizontales entre de cada paso (*Figura 2.15(a)*), para que se activen las opciones del ramal (*Figura 2.15 (b)*), que se encuentran dispuestas en lado derecho la ventana.

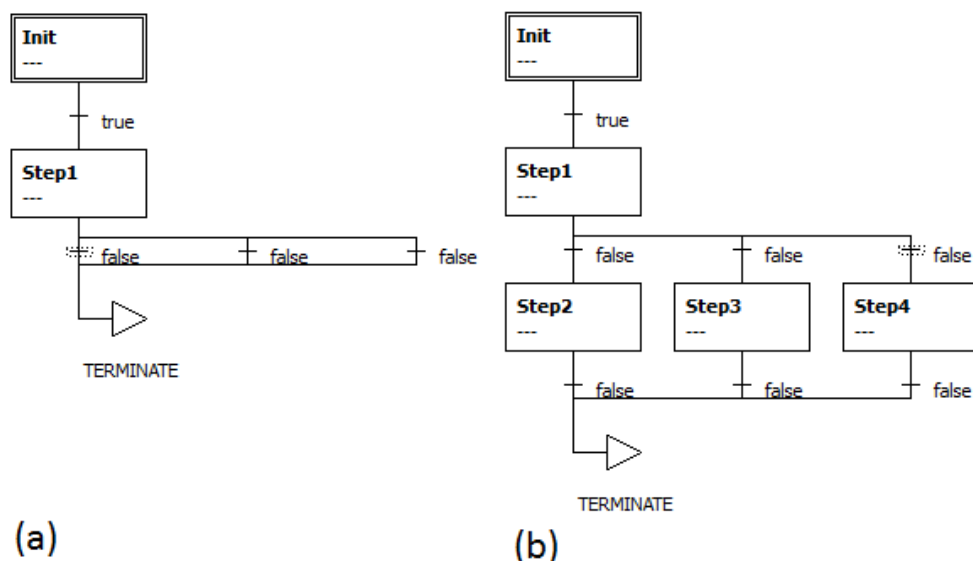


**FIGURA 2.15: CREACIÓN DE RAMALES ALTERNATIVOS Y/O PARALELOS EJEMPLO 2: (A) LÍNEAS DE SECUENCIA ENTRE PASOS; (B) OPCIONES DE RAMALES.**

Seguidamente se dará “click” en la opción “insertar ramal alternativo” , creando ramales alternativos al lado de la línea de secuencia seleccionada (*Figura 2.16 (a)*). Para este ejemplo se insertarán dos ramales adicionales y

se tendrán tres en total, uno para cada subprograma que controla los motores.

Después se dará “click” en cada línea de secuencia nueva y se irá a la opción “insertar paso/transición”  $\square \downarrow$ , para crear un nuevo paso después de cada línea de secuencia (Figura 2.16 (b)).

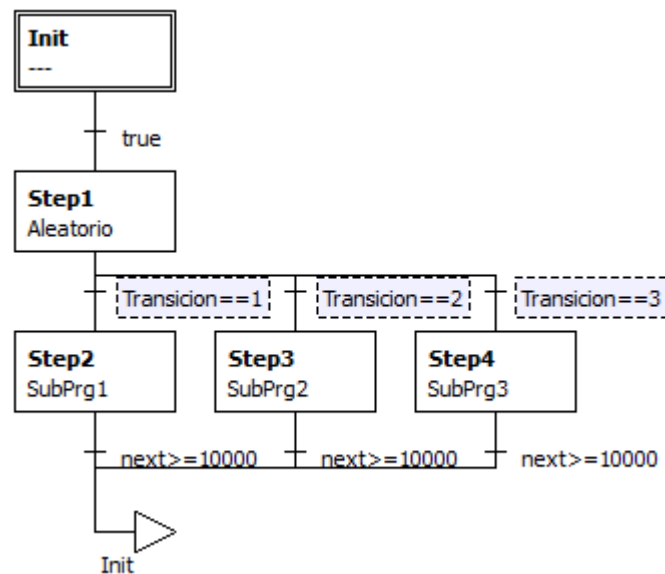


**FIGURA 2.16: EJEMPLO 2 (A) RAMALES ALTERNATIVOS (B) PASOS DE PROGRAMA EN PARALELO**

Ahora se asignará un subprograma a cada paso del programa principal y se establecerán las condiciones de transición en el mismo quedando finalmente como se muestra en la figura 2.17.

Además notar que al final de la secuencia del ejemplo se ha elegido la opción “Init” para hacer que el programa entre en un ciclo para asegurarse que los motores se muevan aleatoriamente, uno a la vez.





**FIGURA 2.17: SECUENCIA TERMINADA DEL PROGRAMA PRINCIPAL DEL EJEMPLO 2**

## 2.2 MatLab® 7 para el sistema didáctico robótico móvil.

Uno de los objetivos de este trabajo de tesis será introducir al lector al uso y control del S.D.Ro.M mediante otro programa y/o lenguaje de programación. Por lo tanto, se hablará de MatLab® 7, un programa ampliamente usado en la Escuela Politécnica del Litoral y en muchas otras universidades a nivel nacional y mundial en ámbitos académicos y de igual manera, por personas en diferentes áreas empresariales.

### 2.2.1 Generalidades de MatLab® 7 y sus controladores para el sistema didáctico robótico móvil.

MatLab® 7 derivado de la abreviatura MATrix LABoratory (Laboratorio de matrices) es un software matemático con lenguaje de programación

propio (lenguaje M), ofreciendo un entorno de desarrollo integrado o IDE por sus siglas en inglés, lo que significa que es un programa informático compuesto por un conjunto de herramientas de programación, que consisten en un “editor de código”, “un compilador”, “un depurador” y un “constructor de interfaz gráfica (GUI)”. y está disponible para las plataformas Unix Windows y Mac OS

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuarios y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. Dispone también de dos herramientas que expanden sus prestaciones: “Simulink” (plataforma de simulación multidominio) y “GUIDE” (editor de interfaces de usuario). Además se puede ampliar sus capacidades con cajas de herramientas “Toolboxes” o paquetes de bloques “blocksets” para Simulink.

Y dada a la importancia que tiene MatLab®, se han creado los controladores que permiten establecer la comunicación y operar la unidad por medio del mismo. Para lo cual se debe hacer los siguientes pasos:

- 1) Sobre la plataforma Windows 7 x86 (versión de 32 bits) instalar MatLab® 7.
- 2) Hecho esto, los drivers necesitan tener un compilador, para que MatLab® los pueda usar, normalmente se usa el Microsoft Visual Studio c/c++, por lo que eso será lo siguiente a instalar, para este trabajo se ha usado Microsoft Visual Studio® 2010 Ultimate (es necesario reconocer la versión del compilador para asegurarse de descargar los drivers correctos).
- 3) Después de instalar el compilador, habrá que dirigirse a la web: [openrobotino.org](http://wiki.openrobotino.org/index.php?title=Downloads#Matlab) para descargar los archivos que se necesitaran (<http://wiki.openrobotino.org/index.php?title=Downloads#Matlab>), de ahí se elige: “OpenRobotino API” según sea la versión del compilador, y el paquete de versión más actualizada de “MatLab®” para Robotino® que se encuentra en la misma página
- 4) Se instalará primero “OpenRobotino API” seguido de los drivers Robotino® para “MatLab®” (para este trabajo se ha usado “API 1.8.31” y “MatLab® 2.1.1”), siguiendo las instrucciones de instalación de los mismos.

Después de terminados estos pasos abrimos MatLab® y habrá que seleccionar como “espacio de trabajo” la carpeta de instalación de los

controladores 'ROBOTINOMATLAB\_DIR'. Además se ejecutan los comandos:

```
>>ddpath( strcat( getenv('ROBOTINOMATLAB_DIR'), '/blockset' ) );
```

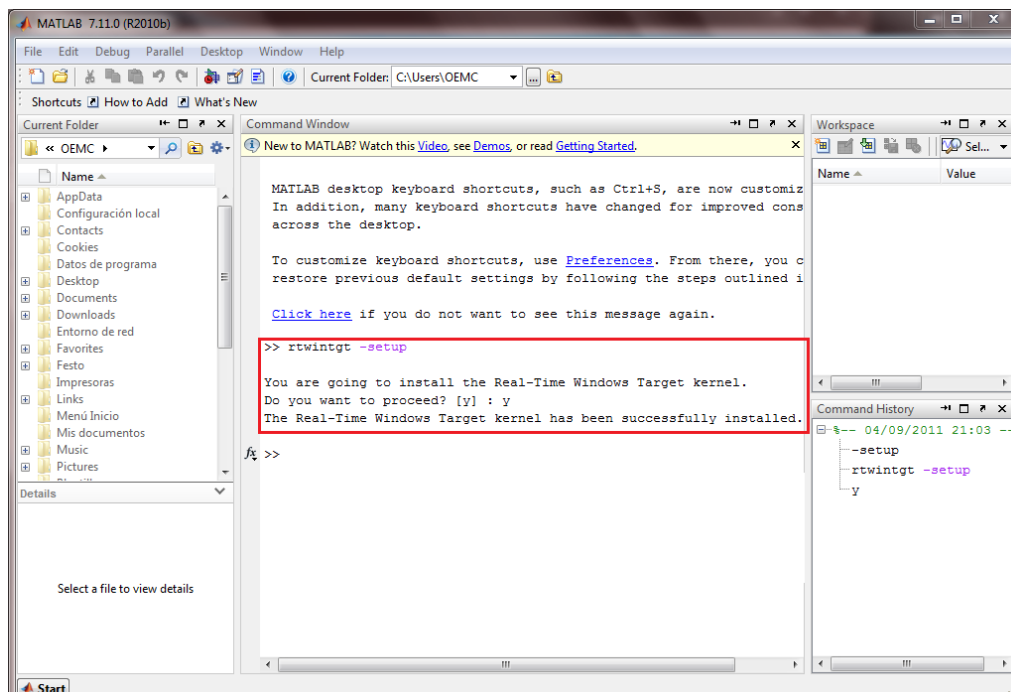
```
>>addpath( strcat( getenv('ROBOTINOMATLAB_DIR'), '/toolbox' ) );
```

o también se puede correr el programa "startup.m" que está dentro de la misma carpeta para designar los paquetes de bloques y las cajas de herramientas que MatLab® usará.

Además se activarán las herramientas de control en tiempo real de MatLab® escribiendo el comando:

```
>>rtwintgt -setup ;
```

quedando como se muestra en la *Figura 2.18*



**FIGURA 2.18: CONFIGURACIÓN DE MATLAB® PARA ROBOTINO® CONTROL EN TIEMPO REAL**

Luego debemos asignar el compilador que MatLab® usará para los archivos “.m”; Para esto se escribe el siguiente comando:

>>mbuilt **-setup** ; seleccionado la opción de Microsoft Visual C++, como compilador de MatLab®. Resultando como se ve en la figura 2.19

```

MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\OEMC
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> mbuilt -setup
Please choose your compiler for building standalone MATLAB applications:

Would you like mbuilt to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2010b\sys\lcc
[2] Microsoft Visual C++ 2010 in c:\Program Files\Microsoft Visual Studio 10.0

[0] None

Compiler: 2

Please verify your choices:

Compiler: Microsoft Visual C++ 2010
Location: c:\Program Files\Microsoft Visual Studio 10.0

Are these correct [y]/n? y

*****
Warning: Applications/components generated using Microsoft Visual C++
2010 require that the Microsoft Visual Studio 2010 run-time
libraries be available on the computer used for deployment.
To redistribute your applications/components, be sure that the
deployment machine has these run-time libraries.
*****

Trying to update options file: C:\Users\OEMC\AppData\Roaming\MathWorks\MATLAB\R2010b\compropts.bat
From template: C:\PROGRA~1\MATLAB\R2010b\bin\win32\mbuildopts\msvc100comp.bat

Done . . .

```

**FIGURA 2.19: CONFIGURACIÓN DE MATLAB® PARA ROBOTINO® (COMPILADOR)**

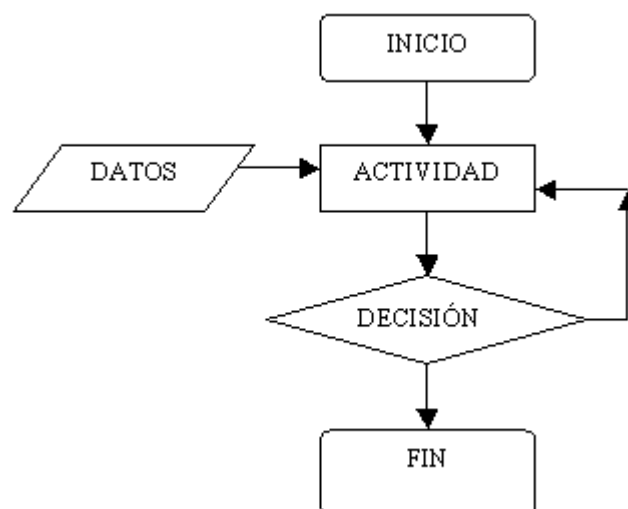
## 2.2.2 Introducción a la programación del sistema didáctico robótico móvil en MatLab® 7

Debido a que MatLab® es ampliamente usado solamente se recordarán las nociones básicas para programar en el mismo.

MATLAB® es un programa de cálculo numérico orientado a matrices. Por tanto, para su sintaxis, será más eficiente si se diseñan los algoritmos en términos de matrices y vectores.

Para elaborar un programa en MatLab® bastará con abrir un archivo de texto y escribir la instrucciones que se deseen y guardar el documento con la extensión “.m”, MatLab® lo reconocerá automáticamente y para ejecutarlo se escribirá el nombre del programa en la ventana de comandos.

Al momento de comenzar a escribir un programa es siempre recomendable empezar por elaborar un diagrama de flujo (*Figura 2.20*) para establecer correctamente lo que se desea hacer, en especial si el mismo es largo o complejo.



**FIGURA 2.20: DIAGRAMA DE FLUJO**

### Comparativos o expresiones relacionales

Al iniciar con la programación, son muy útiles las expresiones que permitan comparar variables entre sí; entre las más usadas se tiene:

**TABLA 2.1:  
OPERACIONES DE COMPARACIÓN PARA PROGRAMAR.**

>	Mayor que
<	Menor que
==	Igual que
<=	Menor o igual que
>=	Mayor o igual que
~=	Diferente que

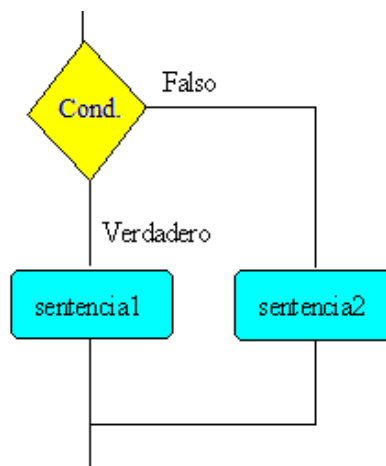
Estos comparativos solo admiten dos respuestas: “verdadero” o “falso” (1/0). Por lo tanto si se quiere preguntar: ¿Es “x” menor o igual a “y”?, en MatLab® lo expresaríamos como:  $x \leq y$ . También hay que notar que el operador de comparación “igual que” de la *Tabla 2.2 1* se construye en MatLab® como “==” con lo que se pregunta: ¿es igual a?, a diferencia de que si se escribiera como “=”, significaría que un valor se asigna a una variable; por ejemplo, si se escribe “x==10” se entiende como “¿x es igual a 10?”, pero si por el contrario, se escribe “x=10” se entiende que “x es igual 10”, independientemente de lo que x valiera en antes.

Además se debe saber que la “negación” de una sentencia en MatLab® se construye con el símbolo “~”, esto nos será útil para trabajar con lógica inversa.

Para resumir los comparativos o expresiones relacionales permiten tomar una decisión a partir de preguntas de respuesta univoca (verdadero o falso).

### Condicionales o ramificaciones

Las ramificaciones son el producto de establecer una condición y dependiendo de la misma permite tomar una decisión, es decir, si la condición es verdadera o falsa, se ejecutan determinadas sentencias.



**FIGURA 2.21: CONDICIONALES EN LA PROGRAMACIÓN**

La ramificación "IF" cuya estructura se muestra en la Figura 2.22

<pre> if (condición)   sentencia end </pre>	<pre> if (condición)   sentencias A else   sentencias B end </pre>
---	--

**FIGURA 2.22: ESTRUCTURA "IF"**

Permitirá realizar una o dos acciones si se cumple la condición.



En efecto si se escribe:

```
a=input('ingrese un número real: ');
if(a>5)
    disp('el número que ingreso es mayor a cinco');
end
```

El programa evaluará si el número ingresado es mayor a cinco, mostrando la frase “el número que ingreso es mayor a cinco” si se cumple la condición y no mostrará nada si no se cumple.

Si por el contrario ahora se escribe:

```
a=input('ingrese un número real: ');
if(a>5)
    disp('el número que ingreso es mayor a cinco');
else
    disp('el número que ingreso es menor o igual a cinco');
end
```

Esta instrucción hará lo mismo pero si no se cumple la condición mostrará la frase: “el número que ingreso es menor o igual a cinco”. También hay que recordar que es posible “anidar” ramificaciones o sea introducir condiciones dentro de otra condición, volviéndose más complejo el código.

Otra posibilidad de ramificación múltiple la ofrece la construcción “switch”. La sintaxis es:

```
switch variable
case valor1,
    sentencias A
case valor2,
    sentencias B
case ...
    ...
end
```

En la ramificación “switch” al llegar a la expresión “switch *variable*”, si *variable* tiene el valor “*valor1*” se ejecutan las “*sentencias A*”; si *variable* toma el valor “*valor2*”, las *sentencias B*; y así sucesivamente.

Es importante notar que la variable sólo debe tomar unos pocos valores: *valor1*, *valor2*, etc. para que el programa se ramifique en unas pocas ramas. No tiene sentido intentar una ramificación “switch” con una variable que pueda tomar un número infinito de valores.

### Lazos o bucles

Los bucles son necesarios cuando se desea que varias secuencias se ejecuten repetidas veces cambiando algunos detalles.

Entre los lazos que se usan se tienen:

<pre>for contador=inicio:paso:fin;     sentencias; end</pre>		<pre>while(condición);     sentencias; end</pre>
--	--	--

**FIGURA 2.23: ESTRUCTURA DE LOS LAZOS O BUCLES EN MATLAB®**

Dado el caso donde se quiere conocer el valor de “y” en la ecuación:  $y=3.5x+(1/4)x^3+1.18^x$  para valores de “x” entre 1 y 25 a intervalos de uno en

uno y adicionalmente se desea saber la tendencia de la ecuación en una gráfica “XY”

Para este ejemplo el programa con la estructura “FOR” quedaría de la siguiente manera:

```
for x=1:1:25
    y=3.5*x+(1/4)*(x^3)+1.18^x;
    X(x,1)=x;
    Y(x,1)=y;
end
plot(X,Y, '*')
```

y con la estructura “WHILE” el programa queda como se muestra a continuación:

```
x=1;
while x<=25
    y=3.5*x+(1/4)*(x^3)+1.18^x;
    X(x,1)=x;
    Y(x,1)=y;
    x=x+1;
end
plot(X,Y, '*')
```

Los dos programas son equivalentes pero cabe notar que en la estructura “WHILE” es necesario iniciar un “contador” para cada variable que se evalué (“x=1”) y definir el paso del mismo (“x=x+1”), lo cual no sucede con el lazo “FOR” donde el “contador” es parte de su estructura (“x=1:1:25”).

Además el bucle “WHILE” es usado cuando no se sabe de antemano cuantas repeticiones se necesiten para cumplir una acción, por lo tanto el lazo “WHILE” seguirá haciendo las repeticiones hasta que se cumpla una determinada condición (“x<=25”).

### 2.2.2.1 Ejemplos de aplicación

Al igual que en la sección “2.1.2.1” se realizarán varios programas de control sencillos para familiarizarse con el uso de los controladores y cajas de herramientas.

#### Ejemplo 1

Como primer ejemplo se aprenderá a establecer una conexión entre MatLab® y el S.D.Ro.M. Robotino.

Para este ejercicio, primero se debe asegurar que MatLab® se está ejecutando con derechos de administrador y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente, el cual, normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

Después se escribirá directamente en la “ventana de comandos” o se abrirá un nuevo archivo “.m” y escribiremos:

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
```

Una vez escritos los comandos se ejecuta el archivo “.m” que se había creado y nos dará las siguientes respuestas:

```
comid =
0
ans =
1
ans =
1
```

Además la pantalla LCD del S.D.Ro.M. se iluminará y presentará el mensaje de que se ha establecido una conexión, confirmado que los comandos se han ejecutado con éxito.

## Ejemplo 2

En el ejemplo a continuación se usarán las cajas de herramientas o “Blocksets” que se instalaron junto con los controladores, buscando activar los motores del S.D.Ro.M. mediante MatLab®.

A igual que el ejemplo anterior como primer paso se establecerá la conexión entre MatLab® el S.D.Ro.M. y a continuación se abrirá un archivo nuevo “.m” y escribiendo las siguientes instrucciones:

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
omniDriveId = OmniDrive_construct
OmniDrive_setComId( omniDriveId, comid )
OmniDrive_setVelocity( omniDriveId, 100, 50, 25 )
```

y nos dará las siguientes respuestas:

```
rec_robotino_com_c.dll loaded.
comid =
0
ans =
1
ans =
1
omniDriveId =
0
ans =
1
ans =
1
```

Y se verán que los tres motores se mueven, dándole al sistema un movimiento combinado de 100mm/s en “x”, 50mm/s en “y” y una rotación de 25Grados/s.

Todas las cajas de herramientas están ubicadas en el directorio de instalación de “Robotino-MatLab”, donde también se podrá acceder a un archivo en “.txt” de ayuda donde se indica la función de cada una de las instrucciones y comandos dedicados al Robotino®.

## **CAPÍTULO 3:**

### **3 DESARROLLO DE NUEVAS PRÁCTICAS DE LABORATORIO CONSIDERADAS.**

### **3.1 Robotino®view 2: Control de las comunicaciones, puesta en funcionamiento y programación de movimientos lineales en sentidos indistintos del sistema robótico**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°1

#### **TÍTULO**

---

Robotino®view 2: Control de las comunicaciones, puesta en funcionamiento y programación de movimientos lineales en sentidos indistintos del sistema robótico

#### **OBJETIVOS**

---

- ✓ Conocer los componentes más importantes de un Sistema Didáctico Robótico Móvil (S.D.Ro.M.).
- ✓ Poner en funcionamiento, probar y explicar los movimientos que ejecuta el S.D.Ro.M. Robotino® y los grados de libertad que posee, mediante Robotino®View 2.
- ✓ Describir y programar movimientos sencillos con bloques de funciones y secuencias en Robotino®View 2; tomando en cuenta aspectos de seguridad en el caso de una colisión del S.D.Ro.M.

#### **MARCO TEÓRICO**

---

##### **Identificación de componentes del S.D.Ro.M. Robotino®**

Todo lo concerniente a los componentes del S.D.Ro.M. fué tratado en la sección 1.3 de este trabajo



## Control de recepción y envío de datos de la unidad robótica Robotino

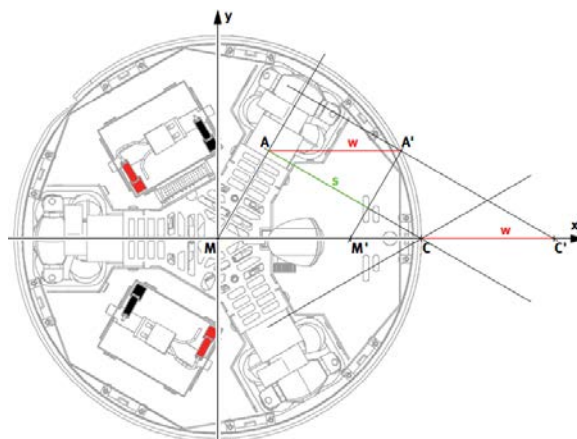
Referirse a la sección 1.4 de este escrito.

### Movimientos lineales y posicionamiento de un sistema robótico

Como ya se ha hablado anteriormente el S.D.Ro.M. Robotino® posee un sistema accionado por 3 ejes distribuidos a 360 grados equitativamente.

Este tipo de distribución da la ventaja de tener grados de libertad independientes en nuestro sistema robótico, pero se presenta otro problema a resolver, que será: determinar la distancia recorrida por la unidad, en relación al número de giros que den los motores

Considerando que Robotino® ejecuta un movimiento hacia delante y recorre un tramo “w” y sus ruedas recorren un tramo “s”



**FIGURA 3.1: CÁLCULO DE RECORRIDO SIN USAR BLOQUE DE OMNIDIRECCIONAMIENTO**

Analizando la imagen se obtiene:

$$(1) s = w \cdot \sin(60^\circ)$$

además se sabe que  $P = d \cdot \pi$  (donde  $d = 80\text{mm}$  es el diámetro de la ruedas dado en la especificaciones ) por lo que se puede inferir la cantidad de giros que hace la rueda para un tramo “s” como:

$$(2) Gi = \frac{s}{P}$$

Reemplazando (1) en (2) finalmente se tiene que:

$$(3) \quad Gi = \frac{w \cdot \sin(60^\circ)}{d \cdot \pi}$$

Gi: cantidad de giros.

w: tramo recorrido por la unidad.

d: diámetro de las ruedas.

Se debe recordar que las unidades de accionamiento del S.D.Ro.M. tiene un contador de pulsos o incrementos (encoder) y consultando la documentación técnica dice que se dan 2048 incrementos por cada giro del motor eléctrico. Además cada motor tiene reductor cuya relación es de 1:16, con lo que se puede determinar la *cantidad de incrementos* (Ci) que deben de dar los motores para que el sistema recorra un tramo “w”:

$$(4) \quad Ci = \frac{w \cdot \sin(60^\circ)}{d \cdot \pi} \times 2048 \times 16$$

## EQUIPOS Y MATERIALES UTILIZADOS

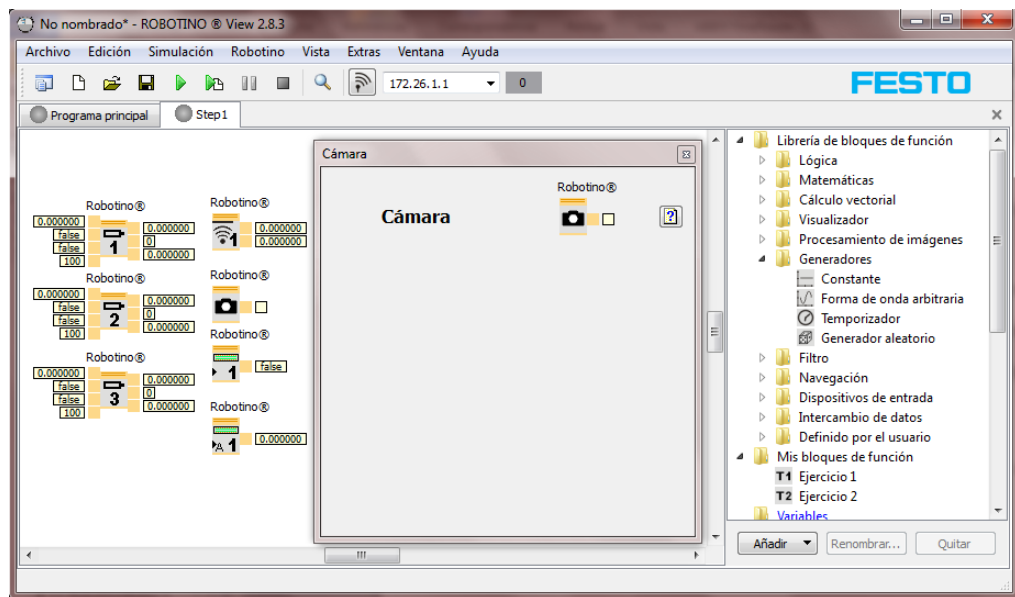
- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.

## PROCEDIMIENTO EXPERIMENTAL

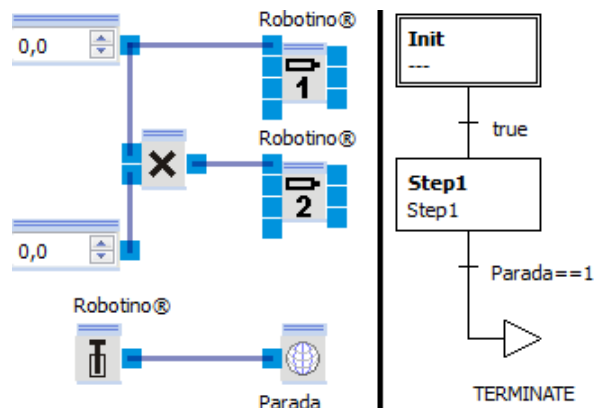
1. Elaborar una tabla de comprobación y control visual del sistema completo
2. Comprobar el funcionamiento de los componentes, observando que la pantalla de la unidad de mando se muestre correctamente y verificando el estado de carga del S.D.Ro.M.
3. Comprobar los movimientos ejecutados por Robotino®, activando las aplicaciones de prueba.
4. En el programa Robotino®View confeccionar un programa con los bloques de función de los componentes y observar su comportamiento.
5. Confeccione otro programa para que Robotino® avance hacia adelante, adicionando una función de protección en caso de colisiones.
6. Determine la cantidad de giros e incrementos que deben efectuar las ruedas para que la unidad avance un tramo de 1m.
7. Confeccione un programa para probar los resultados del punto 6, mida la distancia de 1m y explique por qué se obtienen resultados diferentes al resultado nominal.

8. Pruebe el programa anterior a diferentes velocidades y tipos de suelo

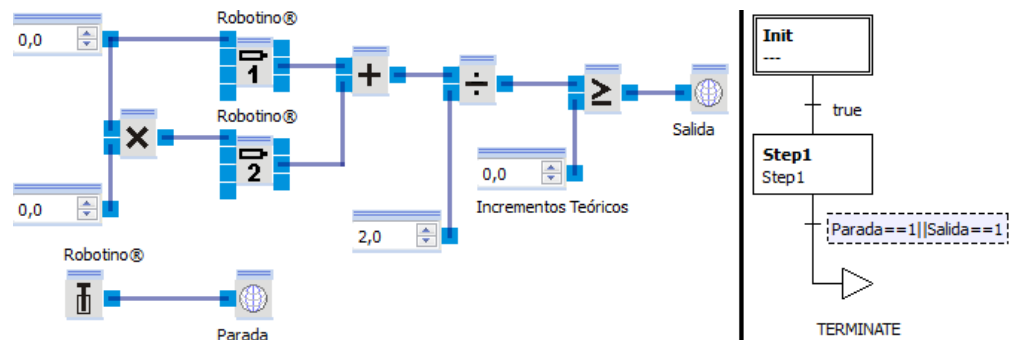
**Solución:** A continuación se muestra las impresiones de pantalla para realizar procedimiento:



**FIGURA 3.2: PROGRAMACIÓN DEL PASO 4 PARA COMPROBAR COMPONENTES, PRÁCTICA 1**



**FIGURA 3.3: PROGRAMA PARA PASO 5, PRÁCTICA 1**



**FIGURA 3.4: PROGRAMA PARA PASO 7, PRÁCTICA 1**

### OBSERVACIONES

Anote las observaciones hechas

Además use la siguiente tabla para apoyar sus observaciones y calcule la desviación nominal del recorrido real del S.D.Ro.M.

**TABLA 3.1:  
OBSERVACIÓN DEL RECORRIDO REAL DEL S.D.RO.M., PRÁCTICA 1**

Trayecto recorrido en metros	Desviación del valor nominal (1m)
$\bar{X} =$ n=10	$\bar{X} =$ n=10

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

## CUESTIONARIO

---

- ¿Qué relación existe entre el movimiento de las ruedas y el comportamiento de Robotino® cuando ejecuta los movimientos?
- ¿Qué motores deben accionarse para que Robotino® avance hacia delante? Y explique por qué, al avanzar hacia delante, los motores deben girar en sentido contrario
- Describa los movimientos e indique los grados de libertad observados (movimientos posibles de los cuerpos y el sistema completo).
- Explique por qué surgen desviaciones en relación con el trayecto nominal de 1m y genere un concepto para optimizar el programa.

### **3.2 Sensores de reflexión directa: programación del movimiento guiado del sistema robótico.**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°2

#### **TÍTULO**

---

Sensores de reflexión directa: programación del movimiento guiado del sistema robótico.

#### **OBJETIVOS**

---

- ✓ Aprender sobre el montaje y conexión de los sensores de reflexión directa.
- ✓ Evaluar y calibrar las señales de un sensor de reflexión directa.
- ✓ Desarrollar una estrategia para la ejecución de movimientos guiados de un sistema de transporte sin conductor, elaborando un programa para aplicar, probar y optimizar las ideas planteadas

#### **MARCO TEÓRICO**

---

##### **Detectores ópticos**

Se puede observar la teoría acerca de “detectores ópticos” en la sección 1.3 de este trabajo.

## **Calibración de los detectores ópticos**

Cada uno de los detectores ópticos está provisto de potenciómetros que regulan la sensibilidad de los detectores.

Esta sensibilidad está basada en la diferencia de reflexión ocasionada por el objeto y por el trasfondo. Si el contraste es débil, es posible ajustar el umbral de respuesta mediante el potenciómetro, de modo que sea posible detectar el objeto bajo las condiciones más difíciles.

Para efectuar un ajuste correcto, deberá preverse un margen de tolerancia considerando posibles cambios en las características de las superficies, ensuciamiento del detector y/o presencia de contaminantes en el ambiente.

## **EQUIPOS Y MATERIALES UTILIZADOS**

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Hoja de papel blanco con marca “negra”
- ✓ Cinta adhesiva negra.

## **PROCEDIMIENTO EXPERIMENTAL**

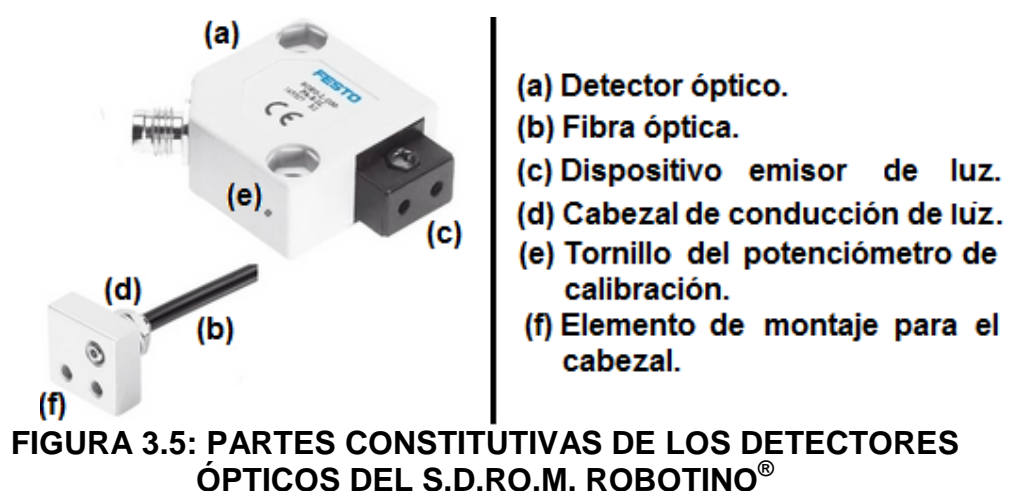
1. Montar los detectores ópticos de reflexión directa en los lugares previstos del chasis de Robotino®.



2. Conectar los detectores a la interface E/S, identificando cual será “derecho o izquierdo”
3. Calibrar los sensores mediante el ajuste del potenciómetro y la ayuda del programa Robotino®View.
4. Medite y analice con sus compañeros una forma para lograr que el S.D.Ro.M. recorra un trayecto fijado, de manera autónoma con el uso de los detectores ópticos de reflexión directa.
5. Elabore un programa en Robotino®View aplicando las ideas derivadas del punto anterior.

**Solución:** Para el montaje especificado en el punto 1, primero se debe identificar las partes constituyentes de los detectores ópticos del S.D.Ro.M.

En la figura 3.5 se muestran dichas partes



Además se necesitarán las siguientes herramientas:

- ✓ Juego de llaves hexagonales.
- ✓ Destornillador plano (1mm).
- ✓ Juego de llaves de boca o llave inglesa pequeñas.
- ✓ Tenaza para cortar fibra óptica.

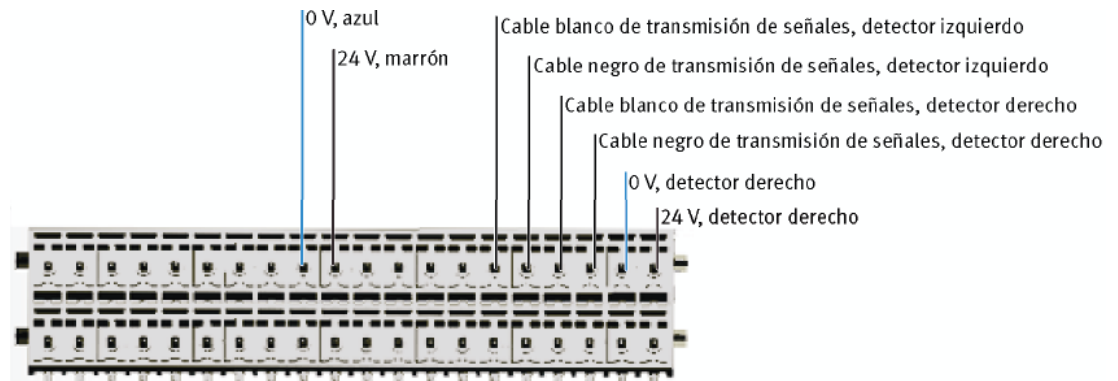
A continuación, con la ayuda de la llaves hexagonales, se fijará el dispositivo conductor de luz a la chapa de montaje de Robotino®.

Luego, atornillar el cabezal de conducción de luz a su elemento de montaje y fijarlo a la ranura destinada en la base de Robotino®

Después, ayudándose con la tenaza, cortar la fibra óptica para obtener la longitud necesaria desde el cabezal hasta el dispositivo emisor de luz. Teniendo especial cuidado en el corte, el cual debe ser liso y perpendicular a la fibra.

Luego, conectar el extremo libre de la fibra óptica al dispositivo emisor de luz.

Finalmente se conectan los cables de los sensores a la interface E/S siguiendo el diagrama de la Figura 3.6.

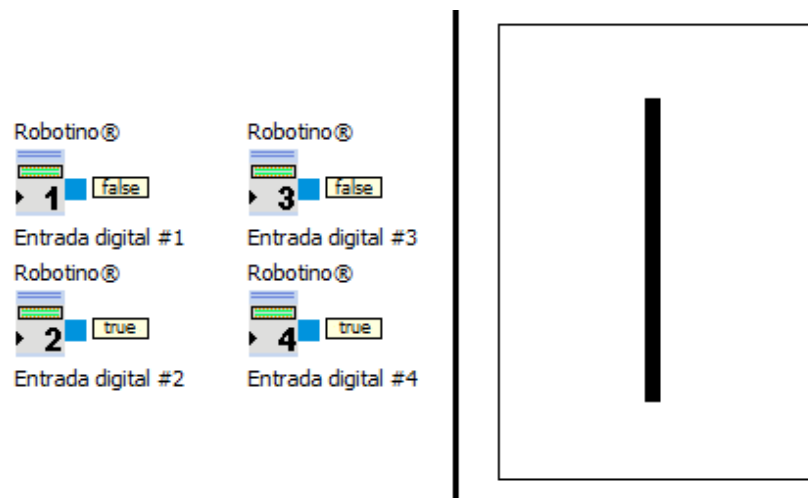


**FIGURA 3.6: ESQUEMA DE CONEXIÓN DE LOS DETECTORES ÓPTICOS**

Para la calibración de los sensores, requerida en el punto 3, se usará ayuda del programa Robotino®View y una hoja blanca con una marca negra (*Figura 3. 7*).

Para cada sensor se interpondrá la hoja con la marca y se observará la reacción que tienen los bloques de función en el programa Robotino®View.

A cada detector le corresponde dos entradas digitales que funcionan de manera opuesta entre sí, según lo que se esté detectando la marca negro o la parte blanca; “1 y 2” sensor derecho, “3 y 4” sensor izquierdo. Estos deben cambiar su estado cuando se les interpone la marca negra, pero si esto no sucede se deberá girar el tornillo de calibración (potenciómetro) hasta entrar al rango de detección de la marca.

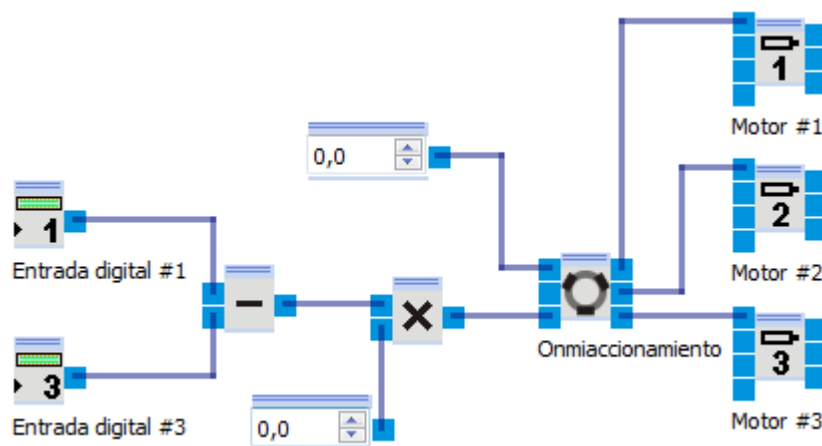


**FIGURA 3.7: AYUDA EN LA CALIBRACIÓN DE LOS DETECTORES ÓPTICO.**

Una vez realizada la calibración se analizará sobre la estrategia tomada hacer que el S.D.Ro.M. recorra un trayecto trazado.

Luego, trazar la pista o trayecto a ser recorrido con la cinta adhesiva negra

Y finalmente se realizará un programa para aplicar todas las ideas surgidas y probarlas.



**FIGURA 3.8: PROGRAMACIÓN BÁSICA PARA EL MOVIMIENTO GUIADO USANDO LOS DETECTORES ÓPTICOS; PASO 5 PRÁCTICA 2**

## OBSERVACIONES

---

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

## CUESTIONARIO

---

- Piense que situación se produce cuando Robotino® ejecuta un giro en sentido anti-horario. En ese caso, ¿Qué detector deberá consultarse?
- ¿Qué puede suceder si los dos sensores quedan encima de la marca negra? ¿Cómo se puede solucionar este problema?

### **3.3 Detector analógico inductivo: comportamiento frente a diversos tipos de materiales y programación del movimiento guiado del sistema robótico.**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°3

#### TÍTULO

Detector analógico inductivo: comportamiento frente a diversos tipos de materiales y programación del movimiento guiado del sistema robótico.

#### OBJETIVOS

- ✓ Aprender sobre el montaje y conexión de un detector analógico inductivo.
- ✓ Evaluar y calibrar las señales de un detector analógico inductivo.
- ✓ Desarrollar un método para reconocer la diferencia entre tipos de materiales metálicos
- ✓ Desarrollar una estrategia para la ejecución de movimientos guiados de un sistema de transporte sin conductor.
- ✓ Elaborar un programa secuencial para aplicar, probar y optimizar las ideas planteadas en el punto anterior.

#### MARCO TEÓRICO

##### **Detector analógico inductivo**

Referirse a la sección 1.3 donde se trata la teoría acerca de sensores inductivos.

### **Calibración de los sensores analógicos inductivos**

Al igual que los detectores ópticos cada sensor inductivo está provisto de potenciómetros que regulan su sensibilidad.

Por lo que la calibración es parecida a la de los sensores ópticos y dependerá de la tarea que se desee realizar:

- Diferenciación entre tipos de materiales metálicos.
- Medición de distancias.

En la diferenciación de materiales se deberá tener el material “Patrón” que se desea detectar y así definir el rango de reacción del sensor.

Por otra parte, para la medición de distancias se supone que el material detectado no se altera, por lo que la variabilidad de las mediciones en el sensor se deberá al cambio de posición relativa entre ellos. En este caso, aparte de la calibración, también habrá que establecer una relación “distancia Vs voltaje” y quedará a criterio del lector, el método más eficiente a utilizar.

### **EQUIPOS Y MATERIALES UTILIZADOS**

---

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.

- ✓ Elemento metálico.
- ✓ Cinta de aluminio ( $\geq 5\text{cm}$ ).

## PROCEDIMIENTO EXPERIMENTAL

1. Montar el detector análogo inductivo en los lugares previstos del chasis de Robotino®.
2. Conectar el detector a la interface E/S
3. Comprobar el correcto funcionamiento del sensor con la ayuda del programa Robotino®View.
4. Considere la manera y elabore un programa para la diferenciación de materiales.
5. Medite y analice con sus compañeros una forma para lograr que el S.D.Ro.M. recorra un trayecto fijado, de manera autónoma con el uso del detector

**Solución:** Para el montaje especificado en el punto 1, primero se debe determinar las conexiones necesarias para el sensor, según su ficha técnica.

Para el caso de Robotino® el sensor inductivo consta de 4 cables: dos de alimentación y dos para la transmisión de señales;

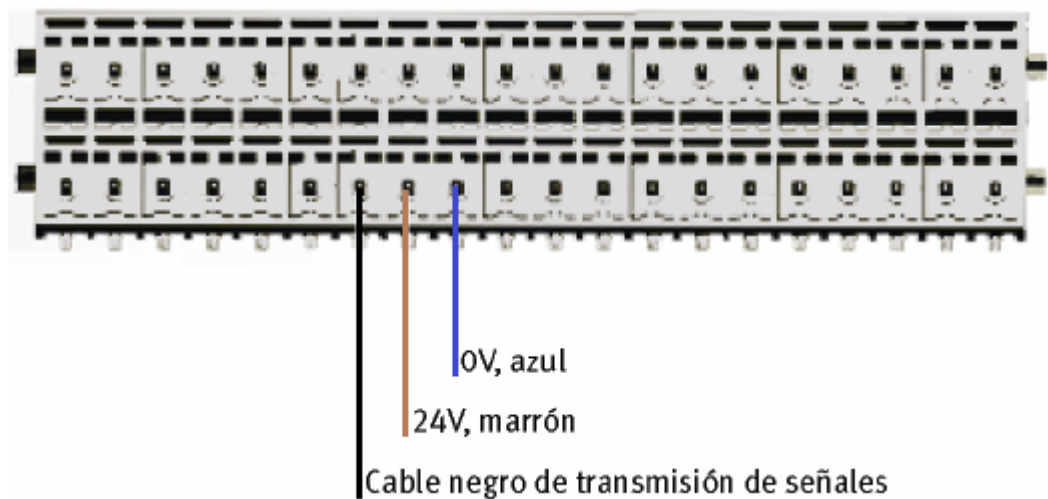
- ✓ BN = marrón; 24V.
- ✓ BU = azul; 0 V.



- ✓ BK = Negro; Transmisión de señales (Valor de tensión).
- ✓ WE = Blanco; Transmisión de señales (Valor de intensidad).

La interface E/S del S.D.Ro.M. únicamente procesa valores de tensión, por lo que se debe escoger el cable negro para la conexión del sensor.

En el esquema de la figura 3.9 se puede observar la conexión del sensor:



**FIGURA 3.9: CONEXIÓN DEL SENSOR INDUCTIVO A ROBOTINO®;  
PASO 2 PRÁCTICA 3**

Identificadas las conexiones se puede proceder a hacer el montaje; para lo cual el sensor viene provisto de una base donde se enrosca y se sujeta con pernos al chasis de Robotino® (*referirse a la figura 1.15*).

A continuación, en la figura 3.10, se muestra la programación que será de ayuda para la comprobación y calibración del detector.

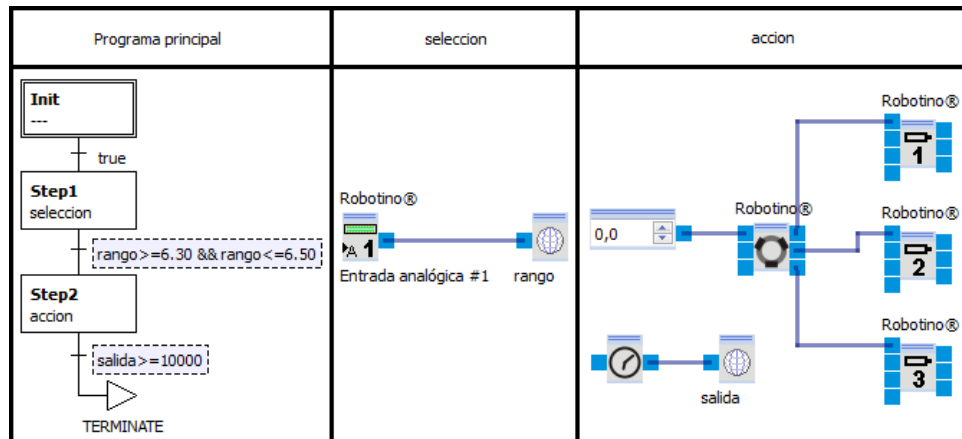


**FIGURA 3.10: PROGRAMACIÓN DE AYUDA PARA LA COMPROBACIÓN Y CALIBRACIÓN DE LOS DETECTORES INDUCTIVOS; PASO 3 PRÁCTICA 3**

Para el paso 4 de esta práctica se considerará un programa en el cual Robotino detecta el material metálico y realizará una acción para confirmar que se ha detectado el objeto correctamente

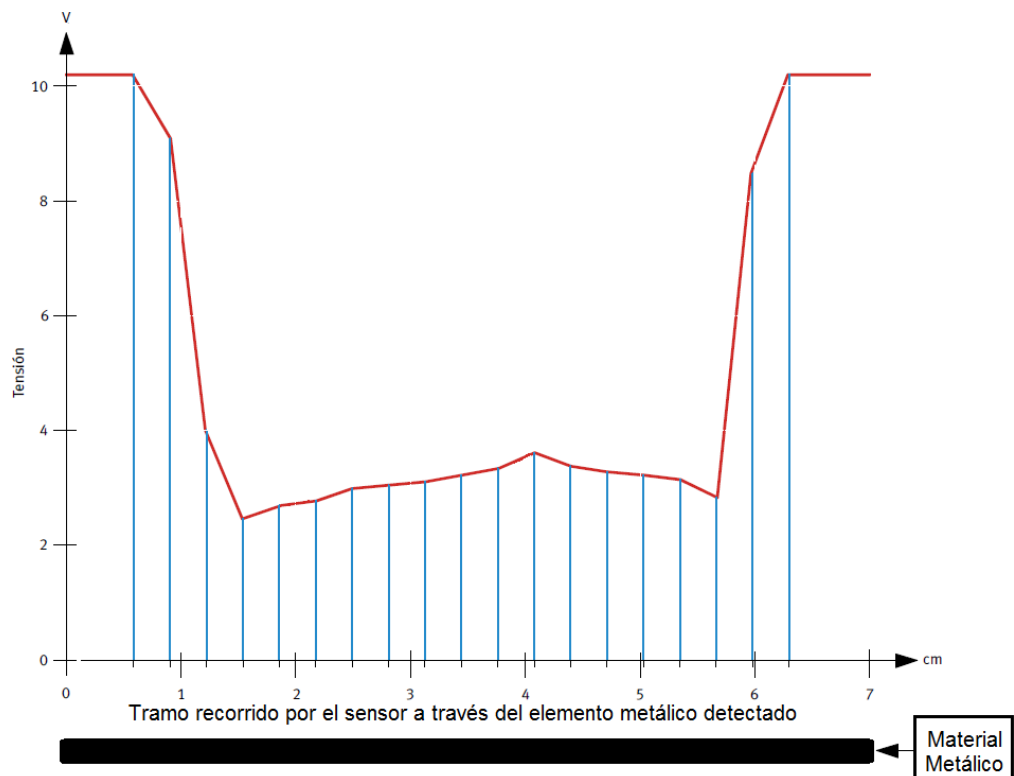
Este programa tendrá un determinado rango de reconocimiento para que el S.D.Ro.M. realice la acción, solo con el tipo de material que deseemos.

En la figura 3.11: se muestra la programación básica para la diferenciación de materiales donde se presenta varios subprogramas que se corren en secuencia para formar el programa principal:



**FIGURA 3.11: PROGRAMACIÓN PARA DIFERENCIACIÓN DE MATERIALES; PASO 4 PRÁCTICA 3**

Siguiendo con el paso 5, se determinará el comportamiento del sensor usando la cinta de aluminio y el programa que se presentó en la Figura 3.10 obteniendo el siguiente resultado:



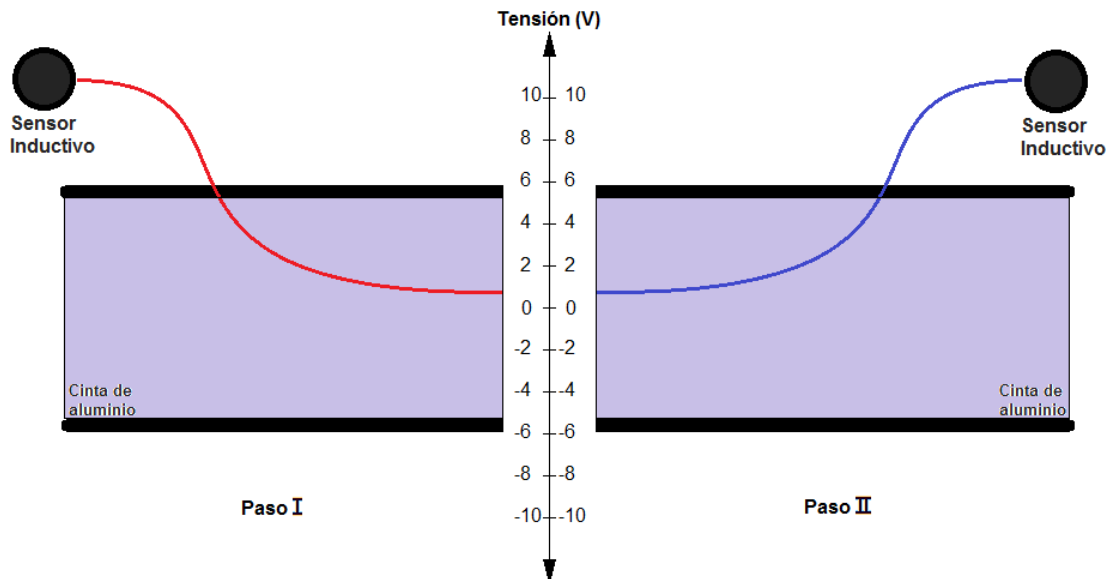
**FIGURA 3.12: COMPORTAMIENTO DEL SENSOR ANTE MATERIALES METÁLICOS; PASO 4 PRÁCTICA 3**

En la gráfica de la Figura 3.12, se observa que el sensor presenta una caída de los valores de tensión mostrados, cuando éste se encuentra en la zona central del objeto y aumenta la tensión conforme se acerca a la periferia del mismo.

Con lo cual se puede inferir una estrategia para lograr que el S.D.Ro.M. recorra un trayecto previamente marcado con la cinta de aluminio.

Para esto se elaborará un programa que constará de dos “pasos” o subprogramas, en los cuales, con la ayuda del bloque de función “Omnidireccionamiento”, se hará que el S.D.Ro.M. realice un movimiento lineal más un movimiento circular, teniendo como resultado un movimiento curvilíneo, que provocará que el detector se desplace de esta misma manera, dentro del ancho de la cinta metálica.

El primer subprograma tendrá el objetivo, que el sistema se desplace desde la zona central hacia la periferia y el segundo hará lo contrario, el sistema se desplazará desde la periferia hasta la zona central de la cinta de aluminio.



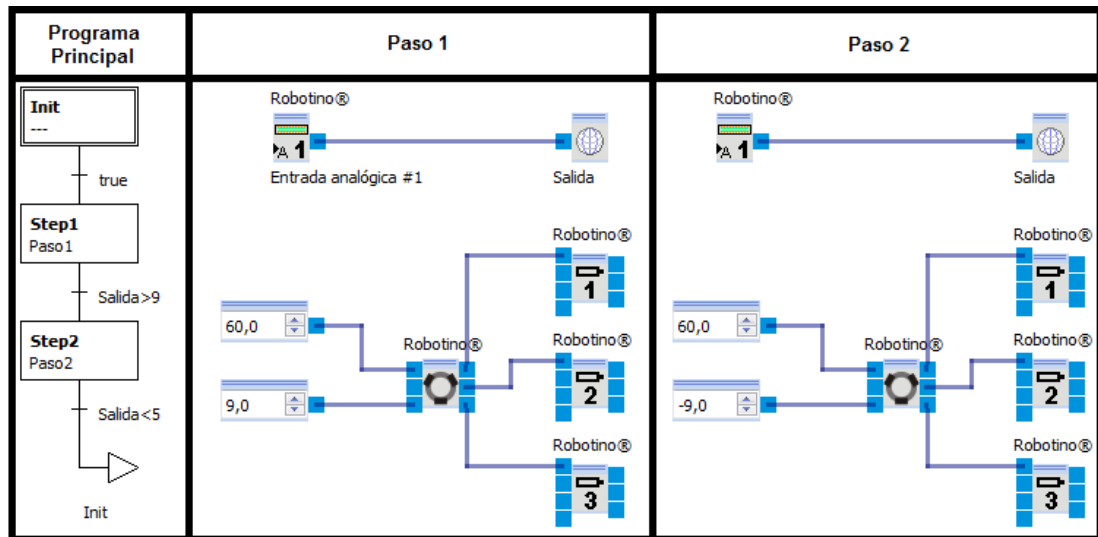
**FIGURA 3.13: ESQUEMA DE LA ESTRATEGIA TOMADA PARA LA PROGRAMACIÓN DEL MOVIMIENTO GUIADO A TRAVÉS DE UN TRAYECTO CON UN SENSOR INDUCTIVO; PASO 5 PRÁCTICA 3**

Cada subprograma se alternará para lograr que el S.D.Ro.M. avance secuencialmente a lo largo del trayecto.

En la Figura 3.13 se observa cómo se da la variación de los valores de tensión del sensor, conforme cambia la posición del mismo respecto a la cinta en cada paso o subprograma.

Luego se hará pasar al sensor a través de todo el trayecto; primero por la zona central y luego por la periferia de la cinta para determinar los valores promedio en cada zona y establecer los límites que darán paso al siguiente subprograma.

Teniendo como resultado el programa que se presenta a continuación en la figura 3.14:



**FIGURA 3.14: PROGRAMACIÓN BÁSICA DEL MOVIMIENTO GUIADO DE ROBOTINO® CON UN SENSOR INDUCTIVO; PASO 5 PRÁCTICA 3.**

### OBSERVACIONES

---

Anote las observaciones hechas

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

### CUESTIONARIO

---

- Para el montaje del detector, ¿Qué debe tenerse en cuenta?
- ¿Cómo se optimizaría el programa de tal manera que los movimientos sean más fluidos?

**3.4 Detectores de distancias infrarrojos: línea característica (curva de calibración), avance en función de distancias precisas y mantenerlas, bordear o evitar obstáculos con el sistema robótico.**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°4

TÍTULO

Detectores de distancias infrarrojos: línea característica (curva de calibración), avance en función de distancias precisas y mantenerlas, bordear o evitar obstáculos con el sistema robótico.

OBJETIVOS

- ✓ Conocer el comportamiento y el funcionamiento de los detectores de distancia por luz infrarroja del S.D.Ro.M.
- ✓ Determinar y registrar la línea de calibración de los detectores de distancia y saber cómo usarla para relacionarla con la medición distancias.
- ✓ Elaborar un programa de regulación donde la unidad sea capaz de acercarse a un obstáculo y mantener una distancia definida respecto al mismo.
- ✓ Elaborar varias estrategias para que Robotino® mantenga distancias ante un obstáculo y/o lo bordee

## MARCO TEÓRICO

---

### **Detectores de distancias infrarrojos.**

Referirse a la sección 1.3 donde se trata la teoría acerca de sensores de medición de distancia (Detectores de rayos infrarrojos).

### **Curva de calibración de los sensores de distancia infrarrojos**

A diferencia de los sensores ópticos y de inducción, estos detectores normalmente no poseen tornillo de calibración pero es posible obtener una curva característica o de calibración. La que nos ayudará establecer una relación entre el valor mostrado (Voltaje) por el sensor y las distancias (centímetros) que tenemos como objetivo medir.

Para establecer esta relación primero se tiene que escoger un método de linealización de ajuste de curva, el cual va a depender de la aplicación que se le dé al sensor, con lo que será potestad del estudiante u operador elegir el método que más se acomode y que sea más eficiente para cumplir con las necesidades que requiera.



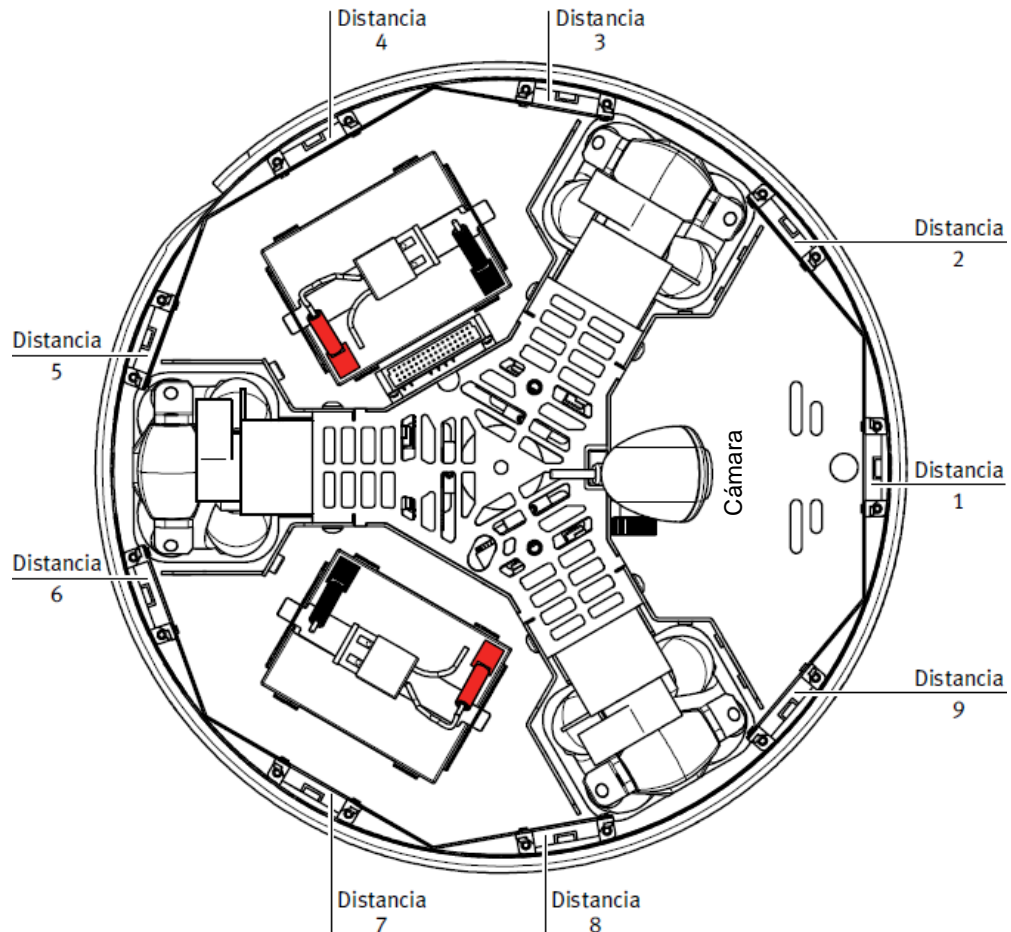
## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Marcador de pizarra acrílica.
- ✓ Cinta métrica o flexómetro.
- ✓ Pliego de papel o pizarra acrílica.
- ✓ Cualquier objeto que servirá de obstáculo


## PROCEDIMIENTO EXPERIMENTAL

1. Identificar cada uno de los sensores que tiene Robotino® y verificar su correcto funcionamiento.
2. Registrar la línea característica del sensor de distancia y establecer relación distancia vs tensión, linealizándola.
3. Elaborar la estrategia a tomar para que el S.D.Ro.M. avance desde una posición y se detenga ante un obstáculo, a una distancia especificada .Adicionalmente genere un programa para realizar esta acción.
4. Realizar otro programa para que Robotino® se desplace a lo largo de una pared manteniendo una distancia específica y únalo en secuencia con el programa anterior.
5. Elaborar una nueva estrategia y programa para lograr que la unidad robótica gire o bordeé un obstáculo de forma irregular.

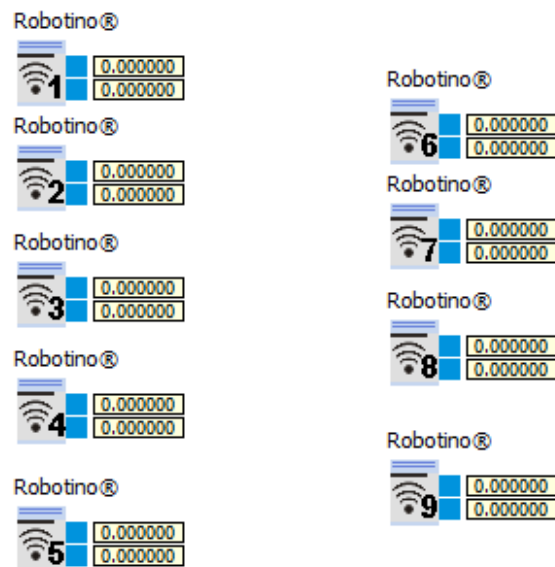
**Solución:** Para realizar el punto 1, posicionamos a Robotino® sobre su base e identificamos sus sensores con la ayuda de la Figura 3.15



**FIGURA 3.15: ESQUEMA PARA IDENTIFICAR CADA SENSOR DE DISTANCIA DE ROBOTINO®; PASO 1 PRÁCTICA 4.**

Luego en el programa Robotino®view previamente conectado a Robotino®, se abre un nuevo proyecto y se halan los bloques de función que representa cada uno de los sensores de distancia (*Figura 3.16*), luego se presionan las teclas “ctrl+d” para visualizar los valores y se da al “click” al botón  para correr el subprograma y se muestren

los valores reales de cada uno de los sensores y para constatar el correcto funcionamiento de los detectores, se les acerca y aleja un objeto haciendo que el sensor cambie el valor mostrado.

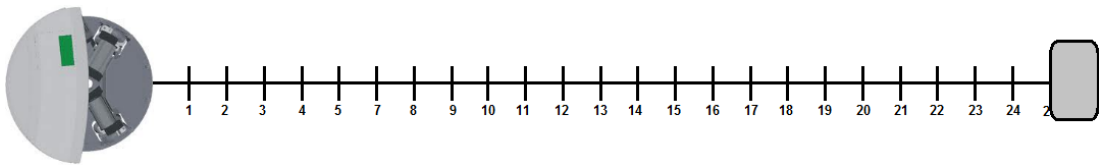


**FIGURA 3.16: PROGRAMA PARA IDENTIFICAR Y VERIFICAR LOS SENSORES DE DISTANCIA INFRARROJOS; PASO 1 PRÁCTICA 4.**

**TABLA 3.2:  
VERIFICACIÓN DE LOS DETECTORES DE DISTANCIA; PASO 1  
PRÁCTICA 4.**

Detector	Funcionamiento OK	Detector	Funcionamiento OK
Distancia 1	<input type="checkbox"/>	Distancia 6	<input type="checkbox"/>
Distancia 2	<input type="checkbox"/>	Distancia 7	<input type="checkbox"/>
Distancia 3	<input type="checkbox"/>	Distancia 8	<input type="checkbox"/>
Distancia 4	<input type="checkbox"/>	Distancia 9	<input type="checkbox"/>
Distancia 5	<input type="checkbox"/>		

Para registrar la línea característica del sensor del paso 2 (Figura 3.17), se tenderá la pizarra acrílica o el pliego de papel sobre el piso y se trazan marcas a 1 cm de espaciamiento a lo largo de un tramo de 25 a 30cm.



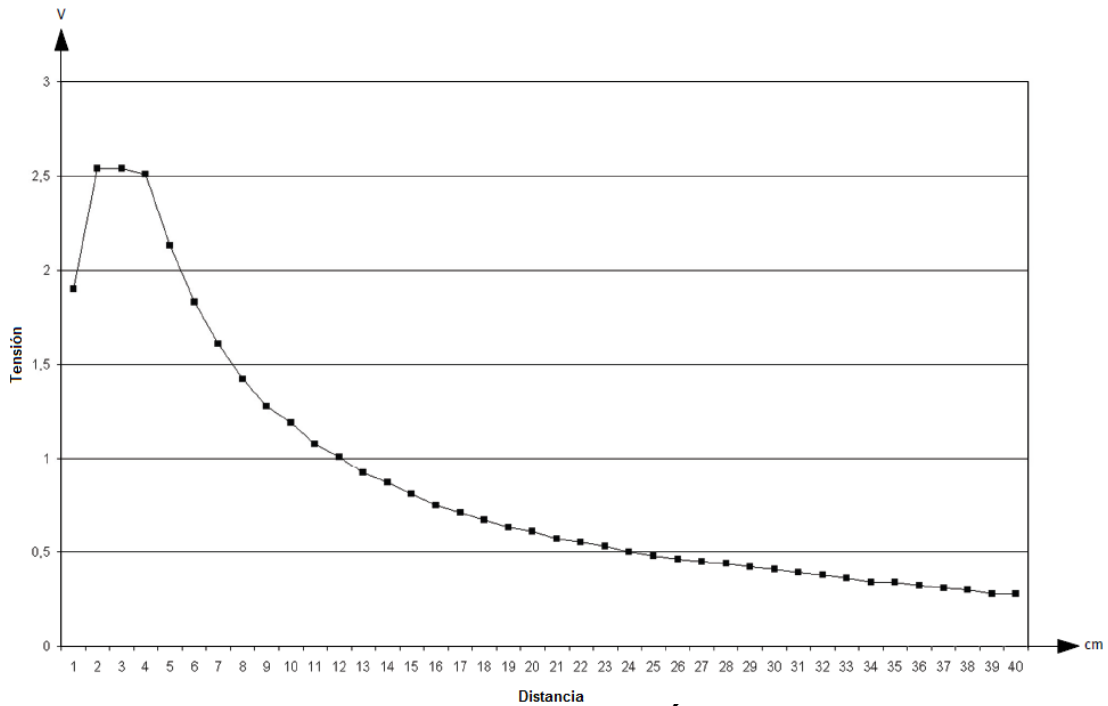
**FIGURA 3.17: ESQUEMA PARA REGISTRAR LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4.**

Hecho esto, se coloca el S.D.Ro.M. sobre el papel o pizarra al inicio de las marcas y un objeto al final de las mismas y con la ayuda del programa de la Figura 3.16, se acerca el objeto hacia Robotino® centímetro a centímetro, anotando cada valor que muestre el sensor del cual se registra la curva característica.

**TABLA 3.3:  
REGISTRO DE LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4.**

Distancia (cm)	Tensión (V)	Distancia (cm)	Tensión (V)	Distancia (cm)	Tensión (V)
1		10		18	
2		11		19	
3		12		20	
4		13		21	
5		14		22	
6		15		23	
7		16		24	
8		17		25	
9					

Con la ayuda de la tabla 3.3 se elaborará la gráfica Distancia vs Tensión (Figura 3.18), la cual se usará para la linealización y calibración del sensor



**FIGURA 3.18: CURVA DE CALIBRACIÓN DE UN SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4**

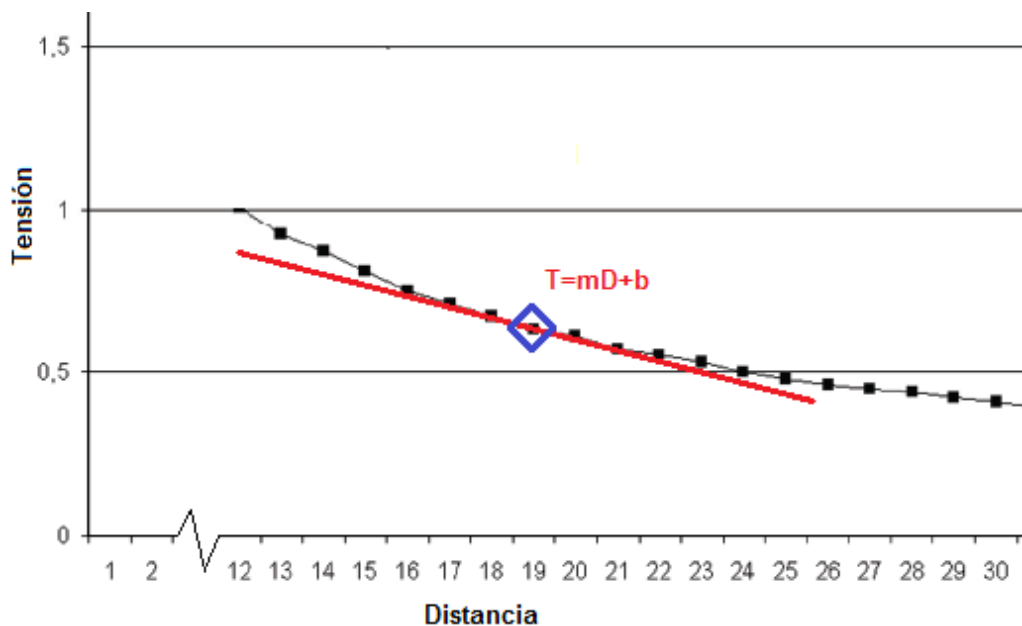
Para los objetivos que se plantean en esta práctica, se debe linealizar la curva presentada en la Figura 3.18, donde se pide realizar una acción a una distancia especificada. Por lo tanto para la linealización, se aproximará una recta alrededor de un punto, que corresponderá a la distancia que se establezca.

En consecuencia, eligiendo dos puntos " $x_o, y_o$ " & " $x_f, y_f$ ", ya sean estos tabulados o estimados y que se encuentren alrededor de la coordenada

que se especificó para la distancia deseada, se calculará la pendiente

$$m = \frac{y_f - y_o}{x_f - x_o}, \text{ y el intersepto } b = y_o - mx_o, \text{ para encontrar nuestra recta}$$

del tipo “ $y=mx+b$ ”, que establecerá la relación Distancia vs Tensión, que se desea: “ $T=mD+b$ ”



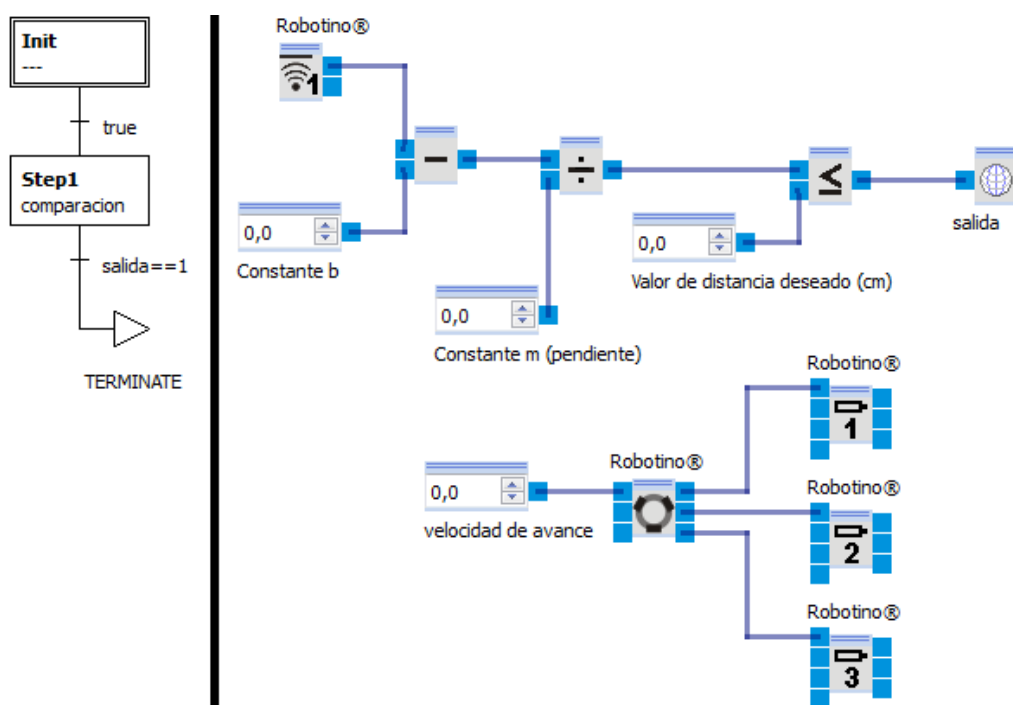
**FIGURA 3.19: LINEALIZACIÓN PARA LA CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4**

Que en términos de distancia tendremos  $D = \frac{T - b}{m}$ , relación que se aplicará, más adelante en la programación.

El siguiente paso se establece que el S.D.Ro.M. avance desde una posición y se detenga ante un obstáculo, a una distancia especificada. Para lograrlo, en el programa Robotino@view, se transformará el valor

mostrado (Tensión) por el bloque de función “distancia 1” en longitud, con la relación que se estableció anteriormente, la cual representará los valores más exactos alrededor del punto que se especifique.

A continuación en la Figura 3.20 se presenta la programación más básica para lograr este objetivo.

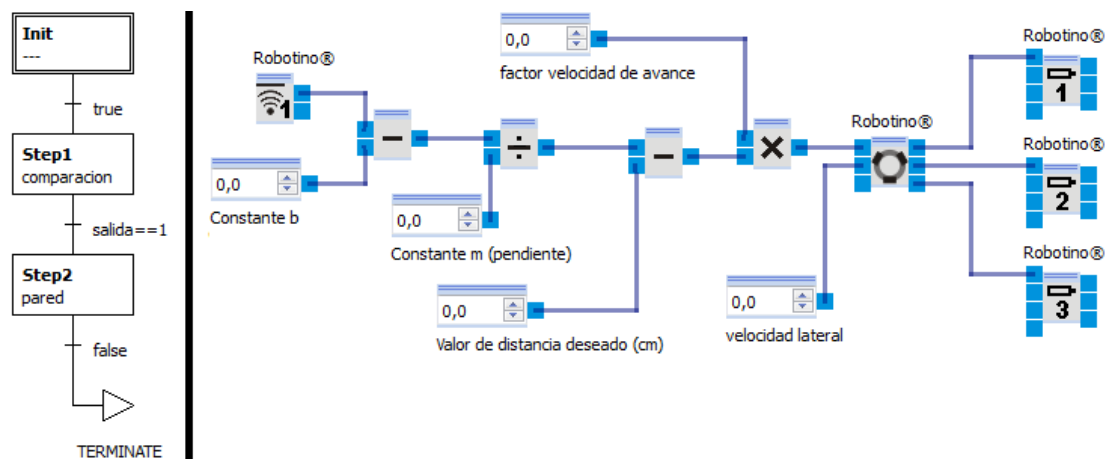


**FIGURA 3.20: PROGRAMA BÁSICO PARA AVANCES EN FUNCIÓN DE DISTANCIAS PRECISAS; PASO 3 PRÁCTICA 4**

En el programa anterior se transforma el valor del sensor en distancia mediante la relación que se obtuvo y luego este resultado se lo compara con nuestro valor deseado, donde se da la acción de que Robotino® se acerque hacia un objeto. Para esta estrategia se deberá

considerar valores de avance pequeños para así mejorar la precisión en la medición de distancia. También podrá ajustar este programa para que haga correcciones en caso de errores de medición.

A continuación para lograr el objetivo del paso 4 se añade una secuencia o subprograma al programa presentado en la Figura 3.20, que hará que el S.D.Ro.M. se desplace a lo largo de una pared, manteniendo una determinada distancia entre ella.



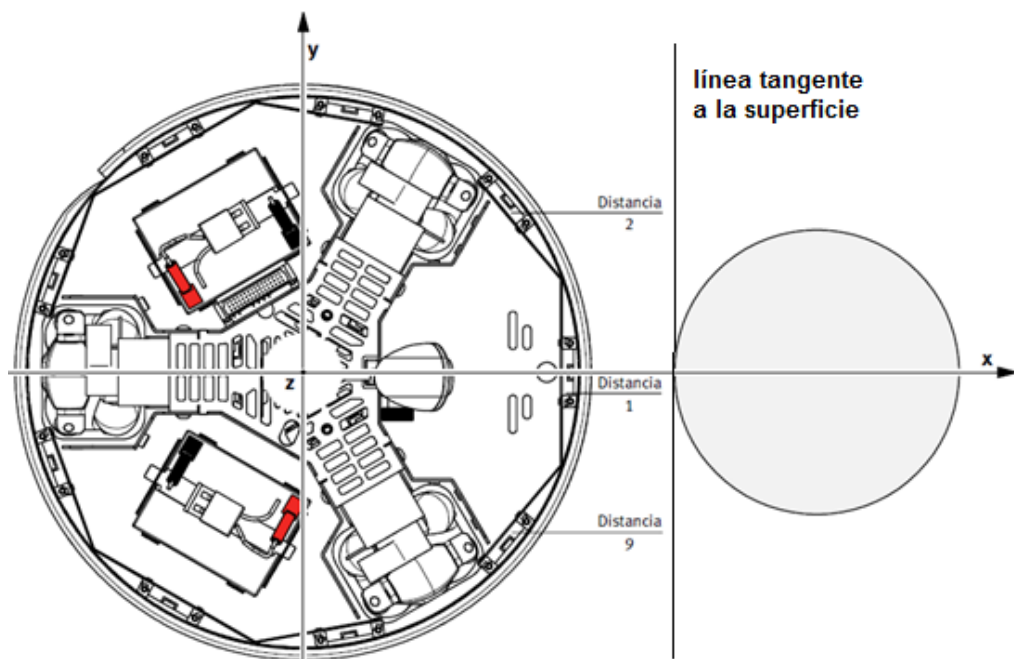
**FIGURA 3.21: SECUENCIA ADICIONAL PARA EL AVANCE A LO LARGO DE UNA PARED; PASO 4 PRÁCTICA 4.**

En la Figura 3.21, se muestra la secuencia que se adiciona para lograr el avance a lo largo de una pared y como queda el programa principal al final.

Finalmente en el paso 5 se tiene como objetivo que la unidad robótica avance, hasta un obstáculo y lo rodee. Para esto, usaremos como



apoyo los sensores de distancia (*Figura 3.22*), que se encuentran a los lados del sensor de distancia 1 (“distancia 2” y “distancia 9”), los cuales servirán para establecer el giro de la unidad robótica, en combinación con los programas vistos en los puntos “3” y “4” del procedimiento.

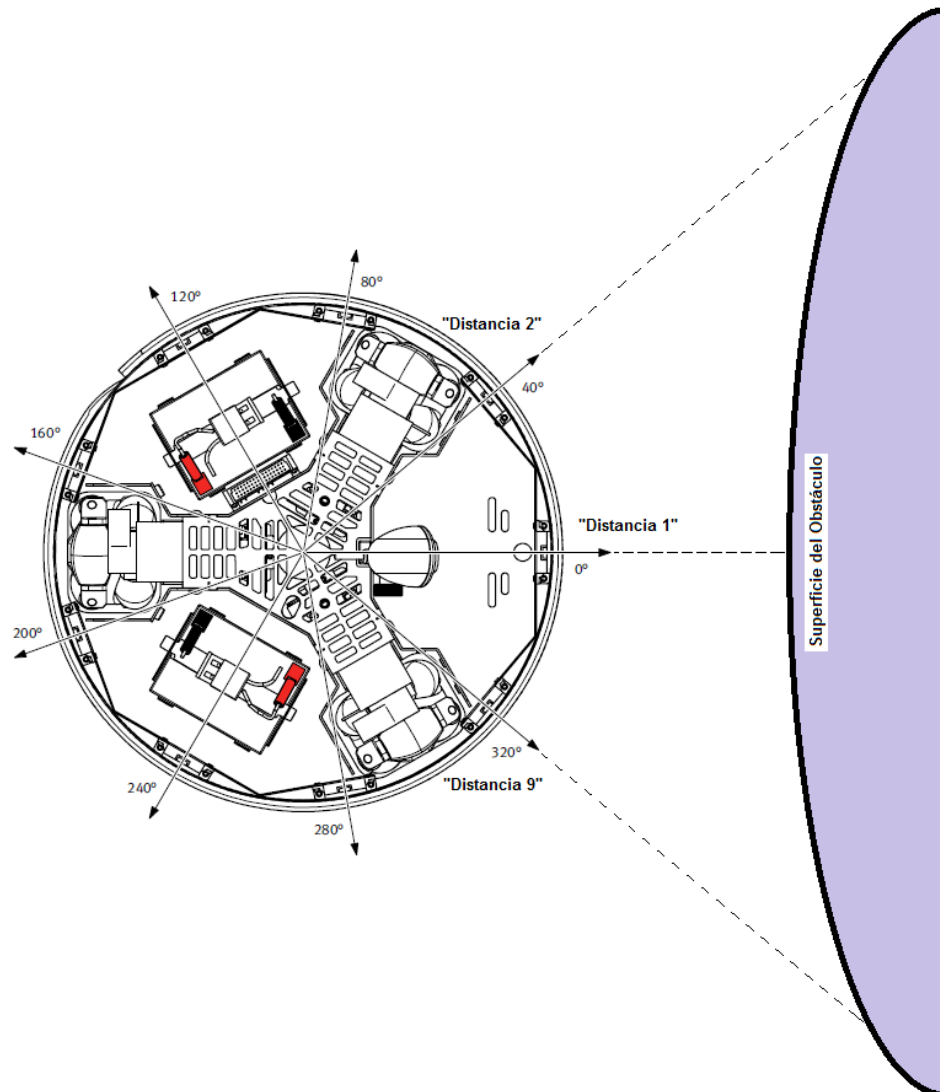


**FIGURA 3.22: ESQUEMA 1 DE LA ESTRATEGIA TOMADA PARA QUE LA UNIDAD RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4.**

Este último, también será adjuntado como una secuencia adicional al programa presentado en la Figura 3.21

La estrategia a tomar, buscará mantener la perpendicularidad entre la “línea tangente” a la superficie de un obstáculo cualquiera y eje “x” de la unidad robótica (*Figura 3.22*). Para esto se usarán los sensores de

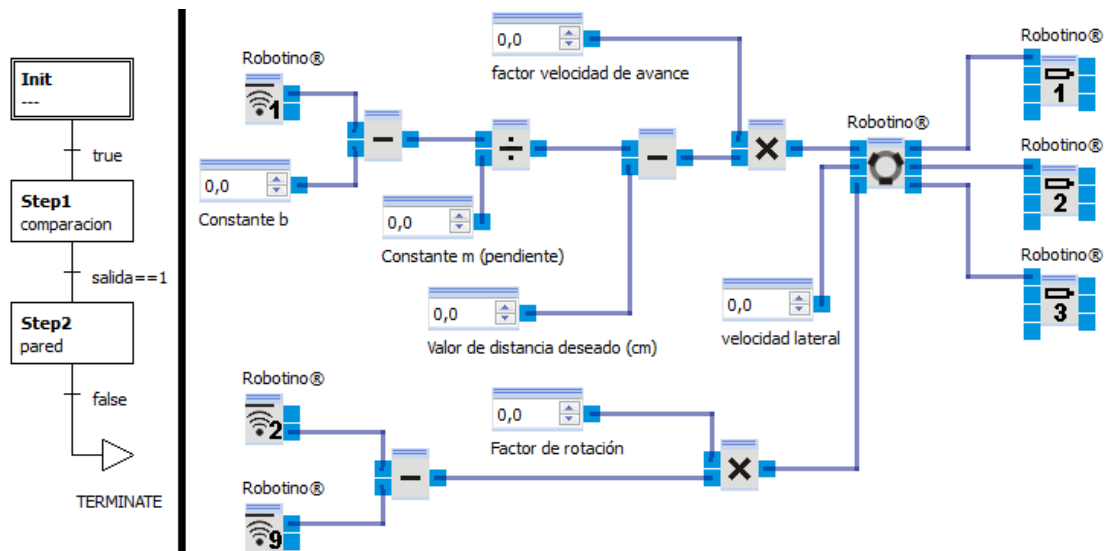
distancia adyacentes, los que medirán distancias de manera oblicua a la superficie del obstáculo (Figura 3.23).



**FIGURA 3.23: ESQUEMA 2 DE LA ESTRATEGIA TOMADA PARA QUE LA UNIDAD RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4.**

Los valores mostrados por los sensores adyacentes deberán ser iguales para asegurar la perpendicularidad entre el detector "Distancia1" y la superficie del obstáculo.

Con lo dicho anteriormente se realizará el programa mostrado en la Figura 3.24:



**FIGURA 3.24: SECUENCIA ADICIONAL PARA LOGRAR QUE ROBOTINO® RODEE UN OBSTÁCULO; PASO 5 PRÁCTICA 4.**

En la Figura 3.24 se agrega una secuencia donde el valor de los sensores adyacentes, de tal manera que: con diferencia “cero” la unidad es perpendicular a la superficie, caso contrario esta diferencia servirá para corregir, con la rotación, la perpendicularidad del S.D.Ro.M.

#### OBSERVACIONES

---

Anote las observaciones hechas

#### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

## CUESTIONARIO

---

- ¿Cómo puede mejorarse el efecto del regulador?
- Piense cómo puede obtener un movimiento circular con la ayuda de bloque de funciones de accionamiento omnidireccional. ¿Qué sucede y porque?

### **3.5 Motores: control, calibración movimiento, lineales y posicionamiento del sistema robótico**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°5

#### **TÍTULO**

---

Motores: control, calibración, movimiento lineales y posicionamiento del sistema robótico

#### **OBJETIVOS**

---

- ✓ Obtener conocimientos básicos acerca de la técnica de regulación de motores eléctricos.
- ✓ Ajustar y optimizar los parámetros de un regulador P.I.D. mediante el software correspondiente.
- ✓ Describir y evaluar los efectos que tiene el ajuste de las constantes del control P.I.D. en los movimientos ejecutados por el robot.

#### **MARCO TEÓRICO**

---

##### **Unidades de accionamiento de Robotino®**

Todo lo concerniente a los motores o unidades de accionamiento se trató en la sección 1.3 de este trabajo.

##### **Método para ajuste de las constantes PID:**

## **Método De Ziegler-Nichols**

En las primeras aplicaciones del control PID el ajuste se basaba únicamente en la experiencia del operador de planta. En procesos lentos cada prueba de sintonía puede llevar horas e incluso días.

Para solucionar estos problemas Ziegler y Nichols (1942) propusieron técnicas empíricas para el ajuste de PID no interactivos, obtenidas tras numerosas pruebas y sin presuponer ningún conocimiento de la planta a controlar.

Existen dos métodos de Ziegler-Nichols

- Ziegler-Nichols en lazo abierto
- Ziegler-Nichols en lazo cerrado

En ambos métodos Ziegler-Nichols, el objetivo es conseguir que: el valor del Máximo sobre-impulso sea menor del 25% para una entrada en escalón

## **Sintonización de controladores PID**

Debido a que casi todos los controladores PID se ajustan en el sitio, en la literatura se han propuesto muchos tipos diferentes de sintonización

delicada y fina de los controladores PID en el sitio. Asimismo, se han desarrollado métodos automáticos de sintonización y algunos de los controladores PID poseen capacidad de sintonización en línea.

### **Control PID de plantas:**

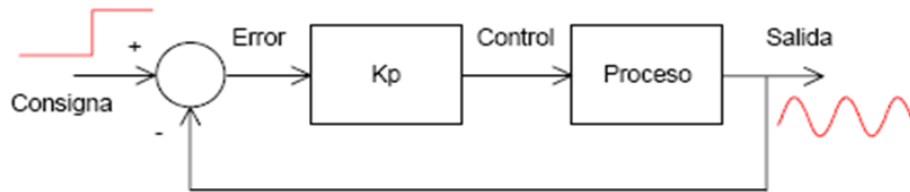
Si se puede obtener un modelo matemático de la planta, es posible aplicar diversas técnicas de diseño con el fin de determinar los parámetros del controlador que cumpla las especificaciones en estado transitorio y en estado estable del sistema en lazo cerrado. Sin embargo, si la planta es tan complicada que no es fácil obtener su modelo matemático, tampoco es posible un enfoque analítico para el diseño de un controlador PID. En este caso se debe recurrir a los enfoques experimentales para la sintonización de los controladores PID.

### **Método de Ziegler-Nichols en lazo cerrado:**

Este método se basa en que la mayoría de los procesos pueden oscilar de forma mantenida bajo control proporcional con una ganancia adecuada (Figura 3.25):

- Ganancia crítica ( $K_{cr}$ )

- Periodo de oscilación mantenida

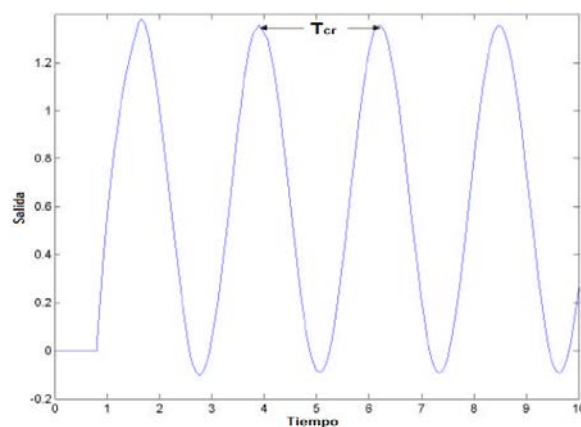


**FIGURA 3.25: PROCESO CON RESPUESTA OSCILATORIA MANTENIDA ANTE UNA PERTURBACIÓN DE ENTRADA ESCALÓN; MÉTODO DE SINTONIZACIÓN PID ZIEGLER-NICHOLS EN LAZO CERRADO.**

Esta técnica de respuesta en frecuencia es un método alternativo de sintonización de PIDs que puede describirse como sigue:

En primer lugar es necesario ajustar las ganancias integral y derivativa a cero, esto es  $K_i = 0$  y  $K_d = 0$

A continuación, partiendo de un valor bajo de la ganancia proporcional,  $K_p$ , se va aumentando ésta gradualmente hasta conseguir un comportamiento oscilatorio mantenido en la respuesta del sistema tal como muestra la Figura 3.26. A esta ganancia se la identificará  $K_{cr}$ .



**FIGURA 3.26: SINTONIZACIÓN PID: AJUSTE DE  $K_p$**



Se llama ganancia crítica porque se aumenta hasta llegar al borde de la inestabilidad en el sistema

El otro parámetro que hace falta es el periodo de oscilación del sistema para esta ganancia, que se identificará como  $T_{cr}$ , y que se calcula gráficamente por la Figura 3.26.

Con los valores de  $K_{cr}$  y  $T_{cr}$  entra a la tabla en la tabla 3.4 de Ziegler-Nichols y se calcula los parámetros correspondientes.

**TABLA 3.4:  
TABLA DE ZIEGLER- NICHOLS PARA EL CÁLCULO DE LAS  
CONSTANTES PID**

Tipo de controlador	$K_p$	$K_i$	$K_d$
<b>P</b>	$0.5 \cdot K_{cr}$	$\infty$	0
<b>PI</b>	$0.45 \cdot K_{cr}$	$1/1.2 \cdot T_{cr}$	0
<b>PID</b>	$0.6 \cdot K_{cr}$	$0.5 \cdot T_{cr}$	$0.125 \cdot T_{cr}$

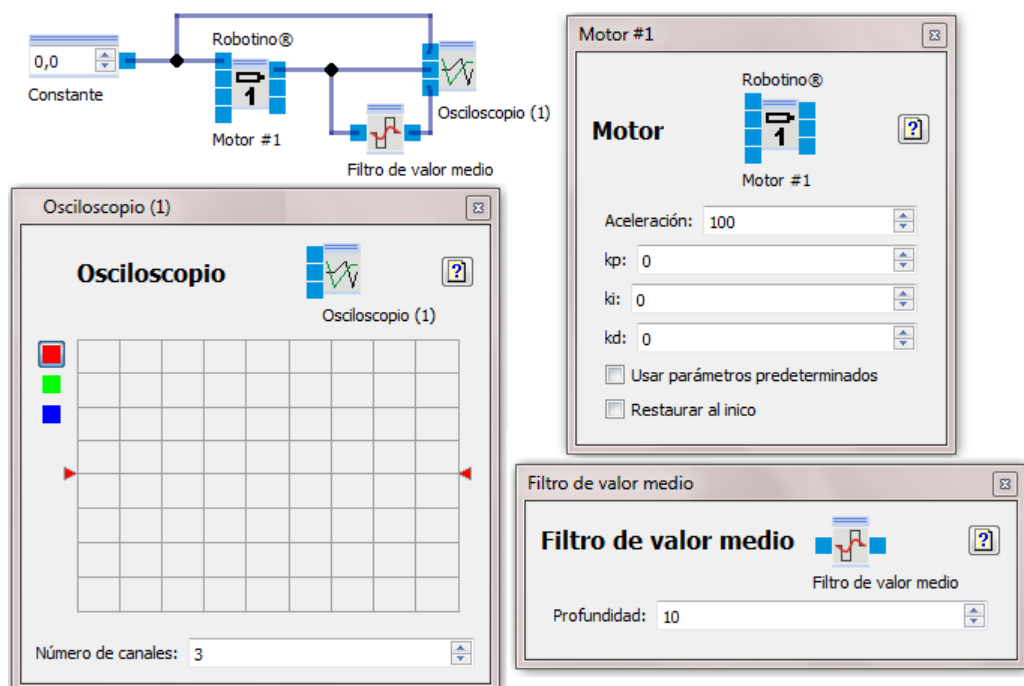
#### EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Base de Robotino®

## PROCEDIMIENTO EXPERIMENTAL

1. Confeccionar un programa o diagrama de bloques en Robotino®View 2 para realizar el ajuste de las constantes PID.
2. Colocar la unidad sobre su base y realizar el procedimiento de sintonización de Ziegler-Nichols con cada uno de los motores.
3. Repetir el procedimiento de sintonización, asentando la unidad sobre una superficie y dejando que recorra libremente.
4. Comparar ambos resultados.
5. Analizar el sistema ante una entrada “escalón”

**Solución:** A continuación se muestra las impresiones de pantalla para realizar el punto 1 del procedimiento:

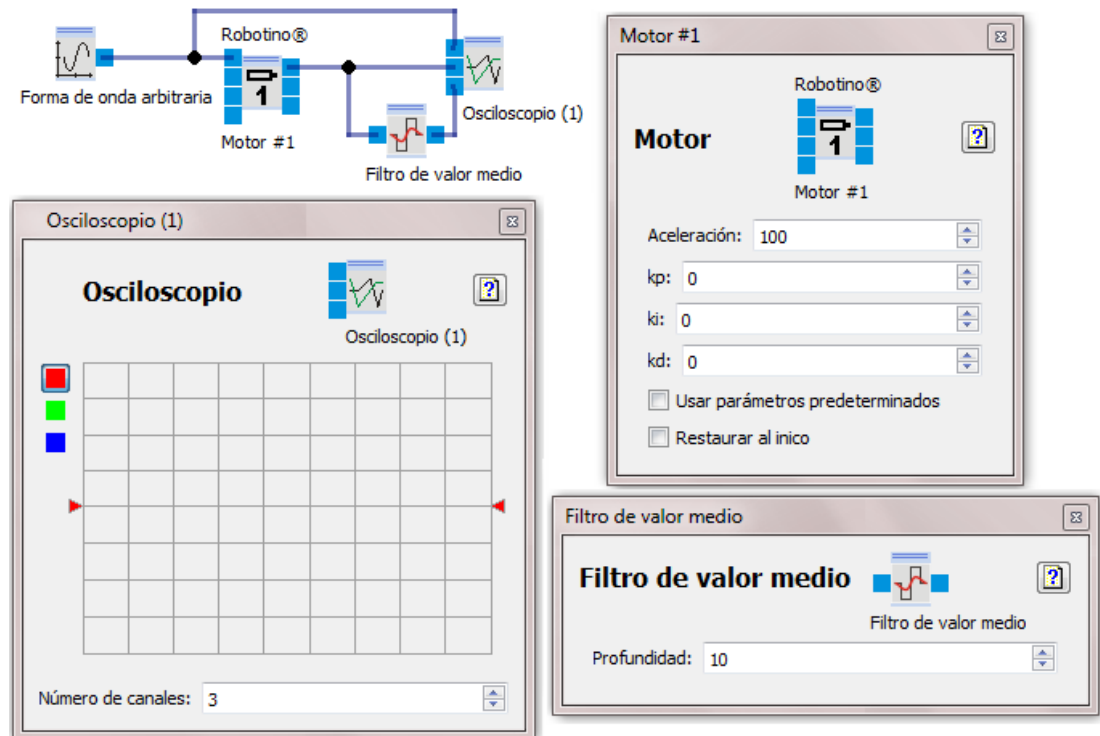


**FIGURA 3.27: PROGRAMA PARA SINTONIZACIÓN PID DE ZIEGLER-NICHOLS; PASO 1 PRÁCTICA 5.**

En la Figura 3.27 se hace uso del osciloscopio para observar la reacción del sistema cuando se varían las constantes del controlador. Además se usa el bloque “filtro de valor medio” como ayuda visual y comprobación de las oscilaciones del sistema.

También se usa el generador de constante para causar la perturbación del sistema, es por esta razón que se pide en el paso 5 que se analice el sistema ante una entrada “escalón” y así observar la reacción de la curva en su estado transitorio (“transient”)

Para eso se cambiará el Bloque “Constante” por uno “Generador de Onda Arbitrario” (Figura 3.28) seleccionando la opción “Rectángulo”



**FIGURA 3.28: PROGRAMA PARA ANÁLISIS ANTE ENTRADA "ESCALON"; PASO 5 PRÁCTICA 5.**

## OBSERVACIONES

---

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

## CUESTIONARIO

---

- En la práctica observe el comportamiento de la rueda omnidireccional. Describa y explique dicho comportamiento.
- En el ajuste de los parámetros del regulador PID. Describa el efecto que tiene en los movimientos. Consulte lo que significan los parámetros P, I y D, ¿Cuáles son sus efectos?
- Considere el margen de valores para los parámetros de regulación, dentro del que los movimientos son aceptables.
- ¿Qué influencia tiene en los movimientos las oscilaciones grandes?
- Explique por qué existen diferencias entre las dos curvas (con Robotino® elevado y sobre el suelo).

### 3.6 Cámara digital: detección de una pieza de diferentes colores y ejercicios de aplicación.

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°6

#### TÍTULO

---

Cámara digital: detección de una pieza de diferentes colores y ejercicios de aplicación.

#### OBJETIVOS

---

- ✓ Conocer el funcionamiento y utilización de la cámara de Robotino®.
- ✓ Configurar las funciones de detección de colores de Robotino®View y saber usarlas.
- ✓ Conocer las limitaciones y condiciones que se tienen la detección de colores y objetos.

#### MARCO TEÓRICO

---

##### **Cámara de Robotino® (Webcam)**

Leer las referencias acerca de “Cámara” en la sección 1.3 de este trabajo.

### **Reconocimiento de patrones (colores o sonido)**

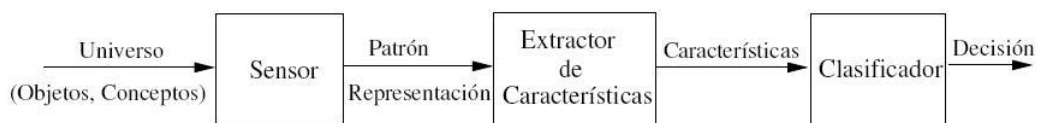
El reconocimiento de patrones es la ciencia que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados con objetos físicos o abstractos, con el propósito de extraer información que permita establecer propiedades de entre conjuntos de dichos objetos.

Los patrones se obtienen a partir de los procesos de segmentación, extracción de características y descripción dónde cada objeto queda representado por una colección de descriptores. El sistema de reconocimiento debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que las diferencia del resto). Para poder reconocer los patrones (*Figura 3.29*) se siguen los siguientes procesos:

- adquisición de datos
- extracción de características
- toma de decisiones

El punto esencial del reconocimiento de patrones es la clasificación: se requiere clasificar una señal dependiendo de sus características.

Estas señales, características y clases pueden ser de cualquiera forma, por ejemplo se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiendo de sus píxeles o se puede clasificar ruidos de cantos de los pájaros en clases de órdenes aviares dependiendo de las frecuencias.



**FIGURA 3.29: ESQUEMA PARA EL RECONOCIMIENTO DE PATRONES E IMAGENES**

### Aplicaciones

Los sistemas de reconocimiento de patrones tienen diversas aplicaciones. Algunas de las más relevantes y utilizadas actualmente son:

- **Previsión meteorológica:** poder clasificar todos los datos meteorológicos según diversos patrones, y con el conocimiento a priori que tenemos de las diferentes situaciones que pueden aparecer nos permite crear mapas de predicción automática.
- **Reconocimiento de caracteres escritos a mano o a máquina:** es una de las utilidades más populares de los sistemas de reconocimiento de patrones ya que los símbolos de escritura son fácilmente identificables.

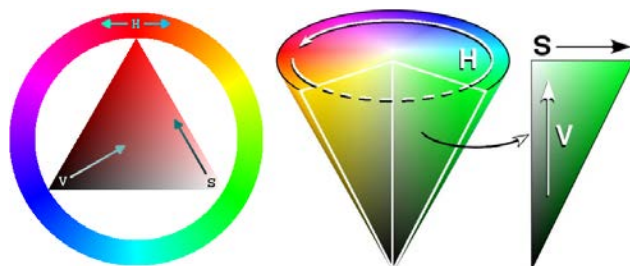
- **Reconocimiento de voz:** el análisis de la señal de voz se utiliza actualmente en muchas aplicaciones, un ejemplo claro son los teleoperadores informáticos.
- **Aplicaciones en medicina:** análisis de biorritmos, detección de irregularidades en imágenes de rayos-x, detección de células infectadas, marcas en la piel.
- **Reconocimiento de huellas dactilares:** utilizado y conocido por la gran mayoría, mediante las huellas dactilares todos somos identificables y con programas que detectan y clasifican las coincidencias, resulta sencillo encontrar correspondencias.
- **Reconocimiento de caras:** utilizado para contar asistentes en una manifestación o simplemente para detectar una sonrisa, ya hay diferentes cámaras en el mercado con esta opción disponible.
- **Interpretación de fotografías aéreas y de satélite:** gran utilidad para propuestas militares o civiles, como la agricultura, geología, geografía, planificación urbana.
- **Reconocimiento de objetos:** con importantes aplicaciones para personas con discapacidad visual.
- **Reconocimiento de música:** identificar el tipo de música o la canción concreta que suena.



### Modelo de color “HSV”

El modelo HSV (Hue, Saturation, Value – Matiz, Saturación, Valor), también llamado HSB (Hue, Saturation, Brightness – Matiz, Saturación, Brillo), define un modelo de color en términos de sus componentes. Tratándose de una transformación no lineal del espacio de color RGB, y se puede usar en progresiones de color.

Es común que se desee elegir un color adecuado para alguna de nuestras aplicaciones, cuando es así, resulta muy útil usar la ruleta de color HSV (Figura 3.30). En ella el matiz se representa por una región circular; una región triangular separada, puede ser usada para representar la saturación y el valor del color. Normalmente, el eje horizontal del triángulo denota la saturación, mientras que el eje vertical corresponde al valor del color. De este modo, un color puede ser elegido al tomar primero el matiz de una región circular, y después seleccionar la saturación y el valor del color deseados de la región triangular.



**FIGURA 3.30: ESPACIO DE COLOR “HSV”**

**Matiz**

Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100%). Cada valor corresponde a un color. Ejemplos: 0 es rojo, 60 es amarillo y 120 es verde.

**Saturación**

Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100%. A este parámetro también se le suele llamar "pureza" por la analogía con la pureza de excitación y la pureza colorimétrica de la colorimetría. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. Por eso es útil definir la insaturación como la inversa cualitativa de la saturación.

**Valor**

Representa la altura en el eje blanco-negro. Los valores posibles van del 0 al 100%. 0 siempre es negro. Dependiendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

## EQUIPOS Y MATERIALES UTILIZADOS

---

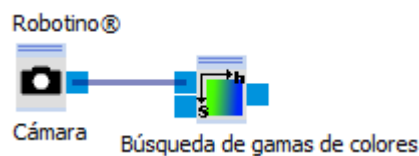
- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Cualquier objeto que servirá para el reconocimiento del patrón

## PROCEDIMIENTO EXPERIMENTAL

---

1. Elaborar un programa para el reconocimiento de patrones de colores.
2. Elaborar un programa para que Robotino® busque y se acerque a un objeto.

**Solución:** Para realizar el primer paso del procedimiento, antes se identifican los bloques de funciones que se usarán y como se los utiliza según el esquema de la Figura 3.29. Abrimos un nuevo proyecto en Robotino®View 2 y se elabora la siguiente secuencia.

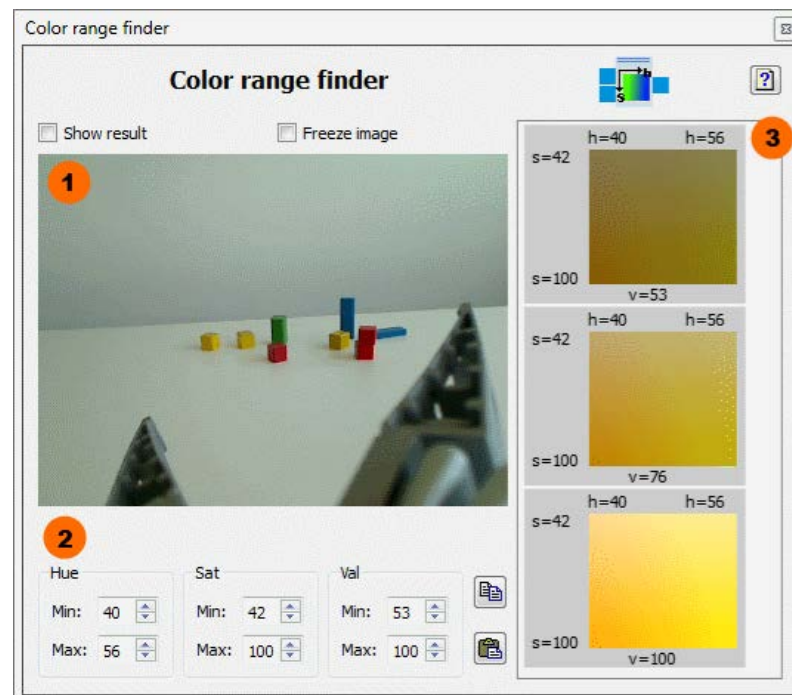


**FIGURA 3.31: SECUENCIA PARA RECONOCIMIENTO DE IMÁGENES; PASO 1 PRÁCTICA 6.**

En la Figura 3.31, se muestra el “sensor” encargado de la adquisición de datos (“cámara”), el cual se lo conecta al bloque “Búsqueda de gamas de colores”, encargado de la extracción de las características y clasificar las imágenes captadas por la cámara.

### Bloque “Búsqueda de gamas de colores”

Este bloque recibe las imágenes captadas por la cámara donde se puede seleccionar una gama haciendo clic sobre ella y arrastrándola con el ratón. Tras la selección de una gama, se calculan los valores máximo y mínimo de todos los píxeles del patrón seleccionado por cada canal de las imágenes. Utilizando para ello una foto convertida al modelo de color HSV (Hue, Saturation, Value)

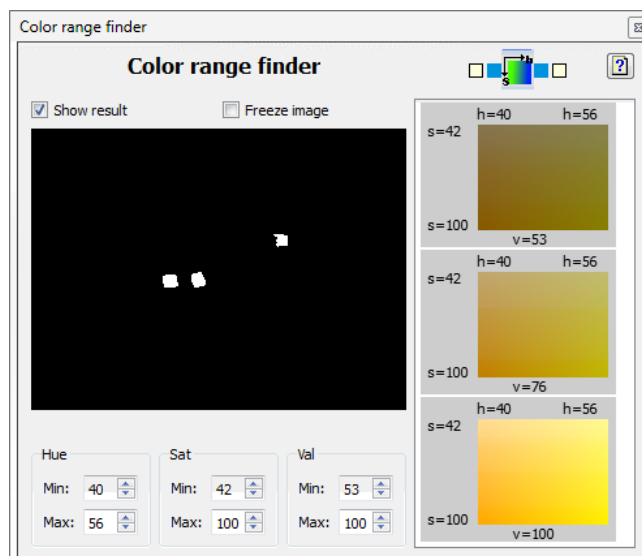


**FIGURA 3.32: USO DEL BLOQUE "BÚSQUEDA DE GAMAS DE COLORES", (A)SE VISUALIZA LA ENTRADA (B)SE CALCULA LOS VALORES MÍNIMO Y MÁXIMO "HSV" (C)REPRESENTACIÓN VISUAL DE LA GAMA DE COLORES SELECCIONADA; PASO 1 PRÁCTICA 6.**

En el bloque de funciones “Búsqueda de gama de colores”, se da “doble click” y se abrirá una pantalla (Figura 3.32) donde se podrá

visualizar la imagen captada por la cámara (*Figura 3.32(a)*), seguidamente tildaremos la opción “congelar imagen”, seleccionaremos el patrón de color de un objeto, a continuación de manera automática el bloque hará el cálculo de los valores máximo y mínimo “HSV” (*Figura 3.32(b)*) y se podrá observar representación visual de la gama de colores seleccionada (*Figura 3.32(c)*).

Finalmente elegimos la opción “Mostrar resultado” y el bloque mostrará una imagen en blanco y negro (*Figura 3.33*) mostrando el resultado del reconocimiento del patrón elegido.



**FIGURA 3.33: RESULTADO DEL PROCESAMIENTO DE IMÁGENES ELABORADO POR EL BLOQUE DE FUNCIONES "BÚSQUEDA DE GAMAS DE COLORES"; PASO 1 PRÁCTICA 6**

A continuación para que se cumpla el paso 1 del procedimiento, se usará la ayuda de la secuencia mostrada en la Figura 3.31,

adicionándole el bloque “Rastreador de segmentos” (Figura 3.33), que ubicará el patrón previamente procesado por el bloque “Búsqueda de gama de colores”

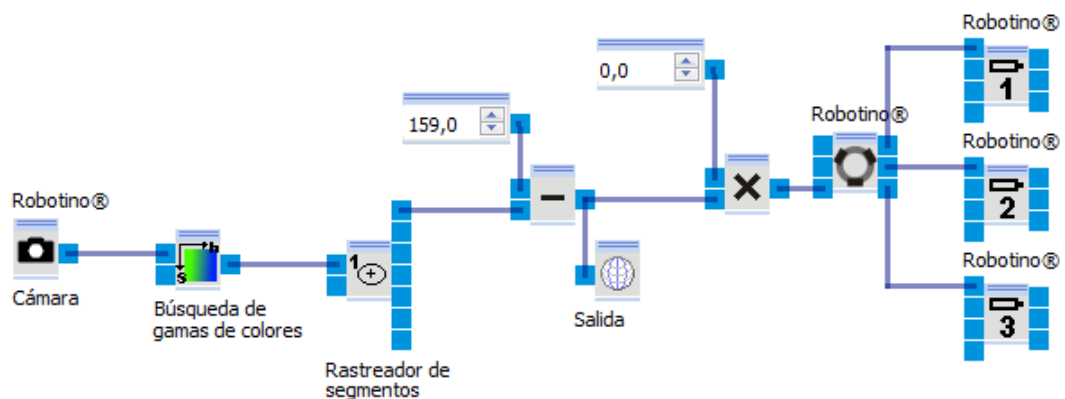


**FIGURA 3.34: USO DE BLOQUE “RASTREADOR DE SEGMENTOS”; PASO 1 PRÁCTICA 6.**

El rastreador de segmentos mide la distancia por pixeles recorridos en X e Y, de la imagen transformada en blanco y negro, por el bloque “búsqueda de gama de colores”. Por lo tanto, observando los valores mostrados por el rastreador es posible determinar, cuando el objeto se encuentra en la mitad de la imagen mostrada por la cámara y se podrá elaborar una estrategia para lograr lo que se pide en el punto 2 del procedimiento.

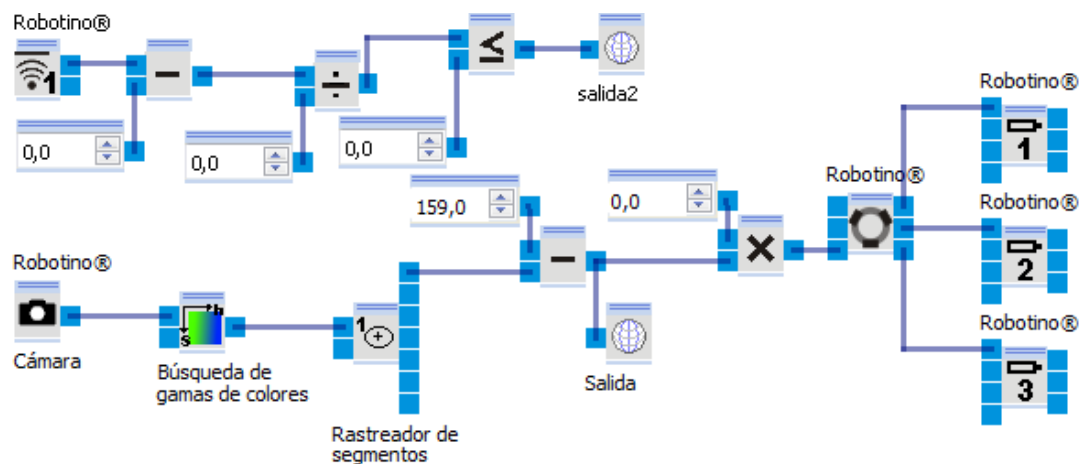
Para hacer que Robotino® busque un objeto y se acerque a él, se elaborará un programa con dos pasos.

El primero, hará que Robotino® gire hasta encontrar el objeto (Figura 3.35).



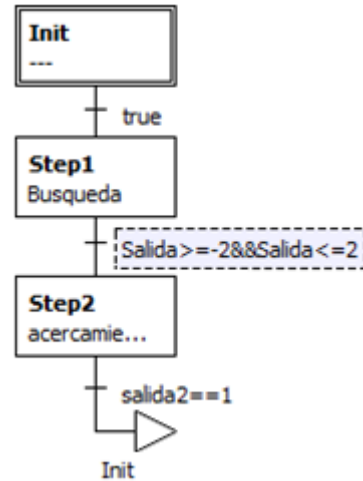
**FIGURA 3.35: PRIMERA SECUENCIA DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 1 PRÁCTICA 6**

El segundo paso será parecido al anterior pero se añadirá, velocidad de avance y la secuencia vista en la Figura 3.21, quedando como se muestra en la Figura 3.36:



**FIGURA 3.36: SEGUNDA SECUENCIA DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 2 PRÁCTICA 6**

Y el programa principal queda como se muestra en la Figura 3.37




**FIGURA 3.37: PROGRAMA PRINCIPAL DEL PROGRAMA DE BÚSQUEDA DE UN OBJETO Y ACERCAMIENTO AL MISMO; PASO 2 PRÁCTICA 6.**

#### OBSERVACIONES

Anote las observaciones hechas

#### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

#### CUESTIONARIO

- ¿Qué sugerencia puede dar, para evitar fallos por el cambio de iluminación?
-  Con referencia a la figura anterior, ¿Qué problemas se pueden dar en el modelo HSV cuando el objeto presenta matices de colores y no un solo color? ¿Qué problema se puede dar específicamente con la figura anterior donde se empieza con violeta y se termina en amarillo?



### **3.7 Movimiento guiado del sistema robótico mediante el uso de una cámara digital**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°7

#### **TÍTULO**

---

Movimiento guiado del sistema robótico mediante el uso de una cámara digital

#### **OBJETIVOS**

---

- ✓ Conocer el funcionamiento y utilización de la cámara de Robotino®.
- ✓ Configurar y usar el bloque de función “Detector de líneas”.
- ✓ Lograr movimientos guiados usando la cámara de Robotino®

#### **MARCO TEÓRICO**

---

##### **Cámara de Robotino® (Webcam)**

Leer las referencias acerca de “Cámara” en la sección 1.3 de este trabajo.

##### **Reconocimiento de línea por medio de la cámara de Robotino®.**

Esto se logra con el bloque de funciones “Detector de líneas”, el cual aplica el mismo método de reconocimiento de patrones explicado en la

sección 3.6, con la diferencia que este bloque es totalmente dedicado a la detección o reconocimiento de líneas en trayectos marcados.

#### EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Pista previamente marcada para el recorrido.

#### PROCEDIMIENTO EXPERIMENTAL

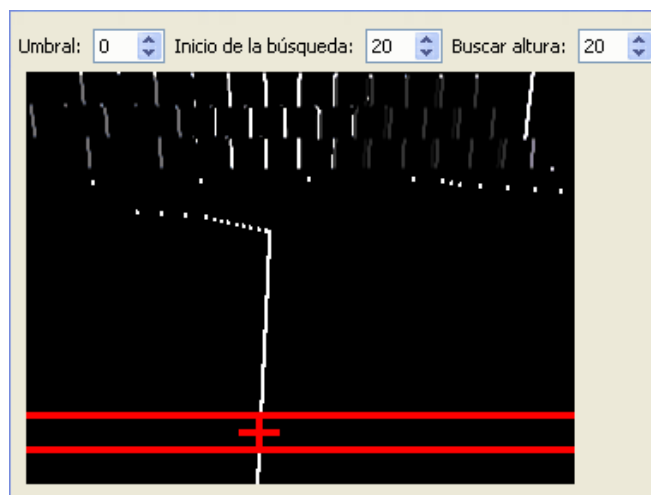
1. Establecer una estrategia para el reconocer y el seguir de una línea.
2. Elaborar un programa para el movimiento guiado mediante el uso de la cámara de Robotino®

**Solución:** Para realizar el punto 1 del procedimiento primero se deberá aprender el uso del bloque de funciones encargado de reconocer la línea trazada.

#### **Bloque “Rastreo de líneas”**

Este bloque entrega directamente la imagen recibida de la cámara al modelo “HSV”. Al dar doble “click” sobre él (Figura 3.38), se abre una ventana presentando la imagen en tiempo real ya transformada.

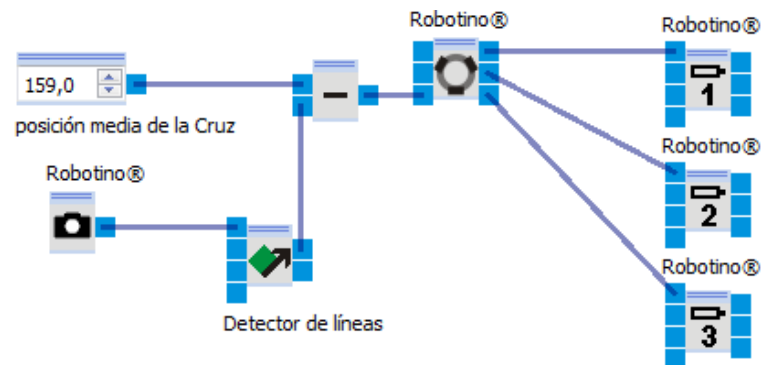
Además el bloque presenta dos líneas y una cruz. La línea inferior señala el "Inicio de la búsqueda" y la superior indica "Inicio de la búsqueda" más la "Altura de búsqueda" y la cruz roja "+" marca el borde de oscuro a claro de la línea vista de izquierda a derecha.



**FIGURA 3.38: VENTANA DESPLEGADA POR EL BLOQUE "DETECTOR DE LÍNEAS"; PASO 1 PRÁCTICA 7.**

Al igual que el "rastreador de segmentos" el "detector de líneas", muestra un valor de "X" medido en pixeles de derecha a izquierda para indicar la posición de la "cruz roja" en la pantalla, por lo que se puede determinar fácilmente en qué valor, la cruz está en el centro de la pantalla y así se podrá elaborar una estrategia para lograr lo que se pide en el punto 2 del procedimiento.

Finalmente, el programa pedido en el punto 2 del procedimiento se muestra en la Figura 3.39



**FIGURA 3.39: PROGRAMACIÓN BÁSICA PARA EL MOVIMIENTO GUIADO DE ROBOTINO® CON LA AYUDA DE UNA CÁMARA DIGITAL; PASO 2 PRÁCTICA 7.**

#### OBSERVACIONES

---

Anote las observaciones hechas

#### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

#### CUESTIONARIO

---

- ¿Qué ventajas y desventajas tiene el movimiento guiado por una cámara contra el movimiento guiado por los sensores ópticos y el de inducción?
- ¿Cómo debe estar orientada la cámara para que pueda detectar la línea de guía delante de Robotino®?
- Explique la influencia que tienen diversos colores en la capacidad de la cámara de detectar la línea
- ¿Qué puede hacerse para minimizar las influencias que interfieren en la detección correcta de la línea de guía?

### **3.8 MatLab® 7: control de comunicaciones, puesta en funcionamiento y programación de tareas básicas del sistema robótico.**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°8

#### TÍTULO

MatLab® 7: control de comunicaciones, puesta en funcionamiento y programación de tareas básicas del sistema robótico.

#### OBJETIVOS

- ✓ Establecer la conexión y el control de comunicaciones de Robotino® mediante MatLab®7
- ✓ Poner en funcionamiento, probar y explicar los movimientos que ejecuta el S.D.Ro.M. Robotino® y los grados de libertad que posee, mediante MatLab®7.
- ✓ Describir y programar movimientos sencillos con las cajas de herramientas "ToolBoxes" en MatLab®7.

#### MARCO TEÓRICO

##### **MatLab® 7 para del sistema didáctico robótico móvil.**

Referirse a la sección 2.2 para aprender acerca de las generalidades de MatLab® 7 aplicado a Robotino®

#### EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.

- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.

## PROCEDIMIENTO EXPERIMENTAL

1. Establecer la conexión y el control de comunicaciones entre Robotino® y MatLab® 7.
2. Escribir una secuencia para realizar movimientos básicos con la unidad, usando las cajas de herramientas que se instalaron junto con los controladores de Robotino® para MatLab®

**Solución:** Para realizar el primer paso del procedimiento, primero hay que asegurarse que MatLab® se está ejecutando con derechos de administrador (Windows 7) y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente y que normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

Después se abre un nuevo archivo “.m” o se escribe directamente en la ventana de comandos lo siguiente:

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
```

Donde la línea “comid = Com\_construct” nos servirá para asignar un puerto o interface de comunicaciones para Robotino®. La siguiente

línea “Com\_setAddress( comid, '172.26.1.1' )” nos servirá para establecer la dirección ip de Robotino®, a través del cual se comunica la interface asignada. Y la última línea “Com\_connect( comid )” establece la comunicación y conexión entre MatLab® y Robotino®

Ejecutando estos comandos nos devolverá la siguiente respuesta:

```
comid =
0
ans =
1
ans =
1
```

Lo que indicará que la conexión ha sido establecida.

Para el segundo paso, se programaran movimientos en “x” (adelante y atrás), en “y” (derecha e izquierda), circulares “ω” (horario y antihorario) o combinaciones de los mismos, mediante la siguiente secuencia:

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
omniDriveId = OmniDrive_construct
OmniDrive_setComId( omniDriveId, comid )
OmniDrive_setVelocity( omniDriveId, 0, 0, 0 )
```

Como ya se sabe las tres primeras líneas establecen la conexión de Robotino® con MatLab®, la cuarta línea construye e identifica el “objeto” de omnidireccionamiento, la quinta línea asocia el “objeto” de omnidireccionamiento con la interface de comunicación y asigna el mismo a Robotino®. Finalmente la sexta línea servirá para asignar los

valores de velocidad en “x” ( $v_x$ ), en “y” ( $v_y$ ) y a velocidad angular ( $\omega$ ), con el siguiente formato:

```
OmniDrive_setVelocity( omniDriveId, v_x, v_y, \omega )
```

Por lo tanto si se desea programar un movimiento solo en “x”, la secuencia quedará así:

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
omniDriveId = OmniDrive_construct
OmniDrive_setComId( omniDriveId, comid )
OmniDrive_setVelocity( omniDriveId, 100, 0, 0 )
```

Se observará que la unidad se moverá en el sentido del eje “x”, y si se quiere cambiar de sentido, simplemente se cambia el signo del valor ingresado

De la misma manera programar un movimiento en sentido del eje “y” o una rotación alrededor del eje “z”, solo se tiene que asignar el valor correspondiente de “ $v_y$ ” u “ $\omega$ ” y los demás valores dejarlo en cero (0).

Por otro lado si se desea combinaciones de movimientos, se asignarán al mismo tiempo valores de “ $v_x$ ”, “ $v_y$ ” u “ $\omega$ ”

```
comid = Com_construct
Com_setAddress( comid, '172.26.1.1' )
Com_connect( comid )
omniDriveId = OmniDrive_construct
OmniDrive_setComId( omniDriveId, comid )
OmniDrive_setVelocity( omniDriveId, 200, 300, 30 )
```



Finalmente, después de que se ejecute el movimiento se devolverán las siguientes respuestas:

```
rec_robotino_com_c.dll loaded.  
comid =  
0  
ans =  
1  
ans =  
1  
omniDriveId =  
0  
ans =  
1  
ans =  
1
```

### OBSERVACIONES

---

Anote las observaciones hechas

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

### CUESTIONARIO

---

- ¿Es posible manejar varias unidades con MatLab® 7? Si su respuesta es positiva describa ¿cómo hacerlo?

### 3.9 Detectores de distancia infrarrojos: programación de sistema de exploración a través de MatLab® 7.

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°9

#### TÍTULO

---

Detectores de distancia infrarrojos: programación de sistema de exploración a través de MatLab® 7.

#### OBJETIVOS

---

- ✓ Conocer la programación y utilización de los sensores de distancia de Robotino® en MatLab® 7.
- ✓ Configurar y usar las cajas de herramientas “DistanceSensor”.
- ✓ Elaborar un programa de exploración donde Robotino® se acerque a los obstáculos y los evite.

#### MARCO TEÓRICO

---

##### **Cajas de herramientas “DistanceSensor”.**

**DistanceSensor\_construct:** construye el objeto “DistanceSensor”, parecido al bloque de funciones visto en Robotino®View.

**DistanceSensor\_setComId:** asocia el objeto “DistanceSensor” a la interface de comunicación. Además lo liga a un Robotino® específico.

**DistanceSensor\_voltage:** es una función que retorna el valor de voltaje de un sensor de distancia específico.

Ejemplo: `[ voltage ] = DistanceSensor_voltage(DistanceSensorId)`

Donde DistanceSensorId se refiere a un sensor en específico.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cualquier objeto que sirva de obstáculo.

## PROCEDIMIENTO EXPERIMENTAL

1. Escriba un código de exploración para que Robotino®, ante un obstáculo de un ambiente cualquiera, gire para evitarlo en MatLab®.

**Solución:** Para realizar el primer paso del procedimiento, primero se debe asegurar que MatLab® se está ejecutando con derechos de administrador (Windows 7) y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente y que normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

El programa siguiente, hará que la unidad explore y cuando se encuentre ante un obstáculo gire para tomar otro rumbo. Aparte tendrá una duración de 60 segundos.

Para este programa hay que recordar que Robotino® tiene 9 sensores de distancia y en MatLab® se identifican desde el 0 al 8.

Se usarán las lecturas de los detectores de distancia para saber si la unidad se ha acercado o no a un obstáculo. Para esto se seleccionan los tres detectores frontales, haciendo que Robotino® avance hasta un obstáculo, se detenga y rote hasta que los sensores dejen de detectarlo.

Primero se necesitará construir todos los objetos que se requieran en el programa:

```
ComId = Com_construct;  
OmniDriveId = OmniDrive_construct;  
DistanceSensor0Id = DistanceSensor_construct(0);  
DistanceSensor1Id = DistanceSensor_construct(1);  
DistanceSensor8Id = DistanceSensor_construct(8);  
BumperId = Bumper_construct;
```

Para el caso de los sensores de distancia también se necesitará especificar el número del sensor como se ve en las líneas 3, 4, 5 de la secuencia anterior.

Después de la creación de los objetos, el programa retorna una identificación para cada objeto. Esto se utilizará posteriormente para la

comunicación con Robotino®. A continuación se establecerá la conexión de Robotino® con su dirección ip:

```
Com_setAddress(ComId, '172.26.1.1');
Com_connect(ComId);
```

Luego de establecer la conexión, se ligará cada objeto con la interface de comunicación creada para Robotino®:

```
OmniDrive_setComId(OmniDriveId, ComId);
DistanceSensor_setComId(DistanceSensor0Id, ComId);
DistanceSensor_setComId(DistanceSensor1Id, ComId);
DistanceSensor_setComId(DistanceSensor8Id, ComId);
Bumper_setComId(BumperId, ComId);
```

Ahora se iniciará el contador de MatLab:

```
tStart = tic;
```

Seguidamente se iniciará un lazo basado en la condición que el sensor de colisión no está activo y dentro del mismo se compara las lecturas de los sensores para saber si el sistema se ha acercado o no a un obstáculo, también se usa la función “OmniDrive\_setVelocity” para detener y rotar al S.D.Ro.M., en caso que se encuentre con un objeto y dejar continúe cuando los sensores ya no detecten nada.

```
while (Bumper_value(BumperId) ~= 1)
    tElapsed = toc(tStart);
    % si los 60 segundos se cumplen sale del lazo
    if(tElapsed >= 60 )
        break;
    end;
```

```

value0 = DistanceSensor_voltage(DistanceSensor0Id);
value1 = DistanceSensor_voltage(DistanceSensor1Id);
value8 = DistanceSensor_voltage(DistanceSensor8Id);

if((0.7 <= value0)|(0.7 <=value1)|(0.7 <= value8))
    % Acercándose a un obstáculo
    OmniDrive_setVelocity(OmniDriveId, 0, 0 ,100);
else
    % Obstáculo no encontrado
    OmniDrive_setVelocity(OmniDriveId, 500, 0 ,0);
end;

end;

```

Una vez los 60 segundos se haya terminado, el lazo se terminará y finalmente se termina con la desconexión de la unidad:

```

Com_disconnect(ComId);
clear

```

## OBSERVACIONES

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

## CUESTIONARIO

- Indique las utilidades que se pueden dar a un robot Explorador.

### **3.10 Detector analógico inductivo: programación de movimiento guiado en MatLab® 7.**

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°10

#### **TÍTULO**

---

Detector analógico inductivo: programación de movimiento guiado en MatLab® 7.

#### **OBJETIVOS**

---

- ✓ Conocer la programación y utilización de los sensores de inducción de Robotino® en MatLab® 7.
- ✓ Configurar y usar las cajas de herramientas “AnalogInput”.
- ✓ Elaborar un programa de exploración donde Robotino® se acerque a los obstáculos y los evite rodeándolos.

#### **MARCO TEÓRICO**

---

##### **Cajas de herramientas “AnalogInput”.**

Al igual que en Robotino®View la entrada análoga se refiere al sensor de inducción, por este motivo al referirse a este detector se lo hará con las cajas de herramientas “AnalogInput”

**AnalogInput\_construct:** construye el objeto “AnalogInput”, parecido al bloque de funciones visto en Robotino®View.

**AnalogInput\_setComId:** asocia el objeto “AnalogInput” a la interface de comunicación. Además lo liga a un Robotino® específico.

**AnalogInput\_value:** es una función que retorna el valor de un sensor de inducción específico.

Ejemplo: `[ value ] = AnalogInput_value (AnalogInputId)`

Donde AnalogInputId se refiere a un sensor en específico.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cinta de aluminio (≥5cm)

## PROCEDIMIENTO EXPERIMENTAL

1. Escriba un código para que Robotino®, recorra un trayecto marcado con una cinta metálica usando el sensor de inducción en MatLab®.



**Solución:** Para realizar el primer paso del procedimiento, primero se debe asegurar que MatLab® se está ejecutando con derechos de administrador (Windows 7) y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente y que normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

El programa siguiente, ilustra el uso de las entradas análogas que tiene Robotino®, a las cuales pueden ser conectados los sensores inductivos y tener la capacidad de detectar una línea metálica en el piso y seguirla. El S.D.Ro.M. tiene 8 entradas analógicas que en MatLab se cuentan desde el 0 hasta el 7

La unidad tiene un detector inductivo conectado cuyas lecturas tendrán un máximo cuando este fuera de la cinta metálica y un mínimo cuando se encuentre en el centro de la cinta. Por lo tanto para controlar el giro del S.D.Ro.M. usaremos el valor que muestre el sensor de inducción restándole el valor medio ( $Valormedio = \frac{V_{max} + V_{min}}{2}$ ) y finalmente se multiplicará por un factor para aumentar la ganancia en el giro. Este programa tendrá una duración de 60 segundos y tendrá una velocidad de avance constante que se combinara con la velocidad de giro.

Primero se necesitará construir todos los objetos que se requieran en el programa:

```
ComId = Com_construct;
OmniDriveId = OmniDrive_construct;
AnalogInputId = AnalogInput_construct(0);
BumperId = Bumper_construct;
```

Después de la creación de los objetos, el programa retorna una identificación para cada objeto. Esto se utilizará posteriormente para la comunicación con Robotino®. A continuación se establecerá la conexión de Robotino® con su dirección ip:

```
Com_setAddress(ComId, '172.26.1.1');
Com_connect(ComId);
```

Luego de establecer la conexión, se ligará cada objeto con la interface de comunicación creada para Robotino®:

```
OmniDrive_setComId(OmniDriveId, ComId);
AnalogInput_setComId(AnalogInputId, ComId);
Bumper_setComId(BumperId, ComId);
```

Ahora se prepara el contador de MatLab:

```
tStart = tic;
```

Y se iniciará un lazo, poniendo como condición que el sensor de colisión no este activado, y se obtendrá el valor de lectura proveniente de la entrada analógica para realizar las operaciones matemáticas descritas anteriormente:

```

vm=input('ingrese el valor medio: ');
a=input('ingrese factor de giro: ');

while (Bumper_value(BumperId) ~= 1)
    tElapsed = toc(tStart);
    % si se cumplen los 60 segundos el lazo terminará
    if(tElapsed >= 60 )
        break;
    end;

    value = AnalogInput_value(AnalogInputId);
    % se resta el promedio y se multiplica por el factor
    value = (value - vm) * 3;
    OmniDrive_setVelocity(OmniDriveId, 100, 0 ,value);
end;

```

Una vez los 60 segundos se haya terminado, el lazo se terminará y finalmente se termina con la desconexión de la unidad:

```

Com_disconnect(ComId);
clear

```

## OBSERVACIONES

---

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

## CUESTIONARIO

---

- Indique las utilidades que se pueden dar a los sensores inductivos.
- Anote las ventajas y desventajas entre la programación de movimiento guiado de Robotino®View y la de MatLab®7

### 3.11 Sensores de reflexión directa: programación de movimiento guiado en MatLab® 7 mediante bloques en Simulink.

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°11

#### TÍTULO

---

Sensores de reflexión directa: programación de movimiento guiado en MatLab® 7 mediante bloques en Simulink.

#### OBJETIVOS

---

- ✓ Conocer la programación y utilización de bloques en Simulink para los sensores de distancia de Robotino® en MatLab® 7.
- ✓ Configurar y usar los bloques de Simulink “DigitalInput0”.
- ✓ Elaborar un programa de exploración donde Robotino® siga el trayecto de una línea negra previamente marcada.

#### MARCO TEÓRICO

---

##### **Simulink.**

Simulink® es un entorno para la simulación multidominio y el diseño basado en modelos para sistemas dinámicos y embebidos. Presenta un entorno gráfico interactivo y un conjunto personalizable de bibliotecas de bloques que permiten diseñar, simular, implementar y probar una serie de sistemas variables con el tiempo, como

comunicaciones, controles, procesamiento de señales, procesamiento de vídeo y procesamiento de imágenes..

También, vendría a ser una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Se hace hincapié en el análisis de sucesos, a través de la concepción de sistemas (cajas negras que realizan alguna operación).

Se emplea arduamente en Ingeniería Electrónica en temas relacionados con el procesamiento digital de señales (DSP), involucrando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros. También es muy utilizado en Ingeniería de Control y Robótica.

#### EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cinta adhesiva negra.

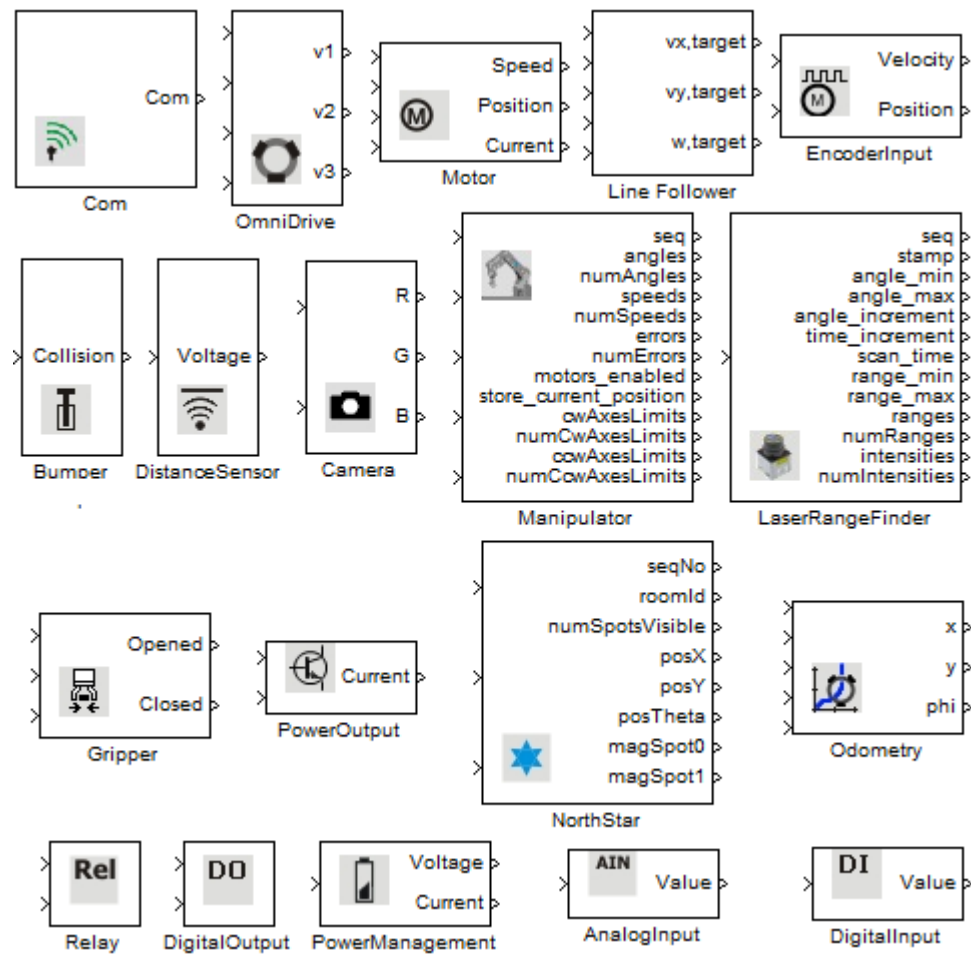
## PROCEDIMIENTO EXPERIMENTAL

---

1. Escriba un código para que Robotino®, recorra un trayecto marcado con una cinta usando el sensor óptico en Simulink®.

**Solución:** Para realizar el primer paso del procedimiento, primero se debe asegurar que MatLab® se está ejecutando con derechos de administrador (Windows 7) y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente y que normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

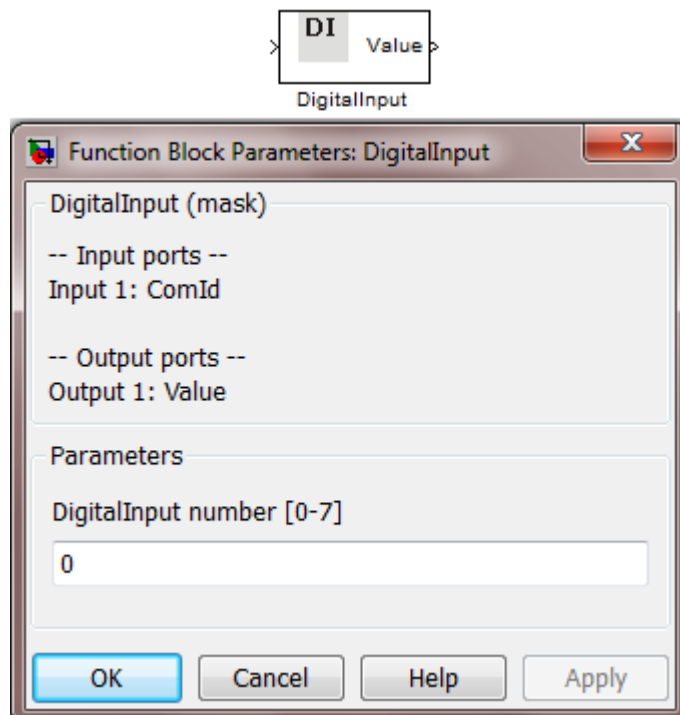
El siguiente programa demuestra la capacidad de programar mediante el entorno Simulink®, gracias a los “blocksets” que se colocaron junto con la instalación de los controladores de Robotino® para MatLab®7.



**FIGURA 3.40: LIBRERÍA DE BLOQUES PARA SIMULINK®; PRÁCTICA 11**

En Simulink® cada uno de estos bloques se les puede agregar una “Sub-máscara” para ingresar los “valores iniciales” al bloque. Para acceder a esta función simplemente se da “doble click” sobre el bloque y se abrirá una ventana con la información del bloque los datos que se pueden ingresar en él

Así es como, si damos doble click sobre el bloque "DigitalInput" se abrirá la siguiente ventana presentada en la Figura 3.41:



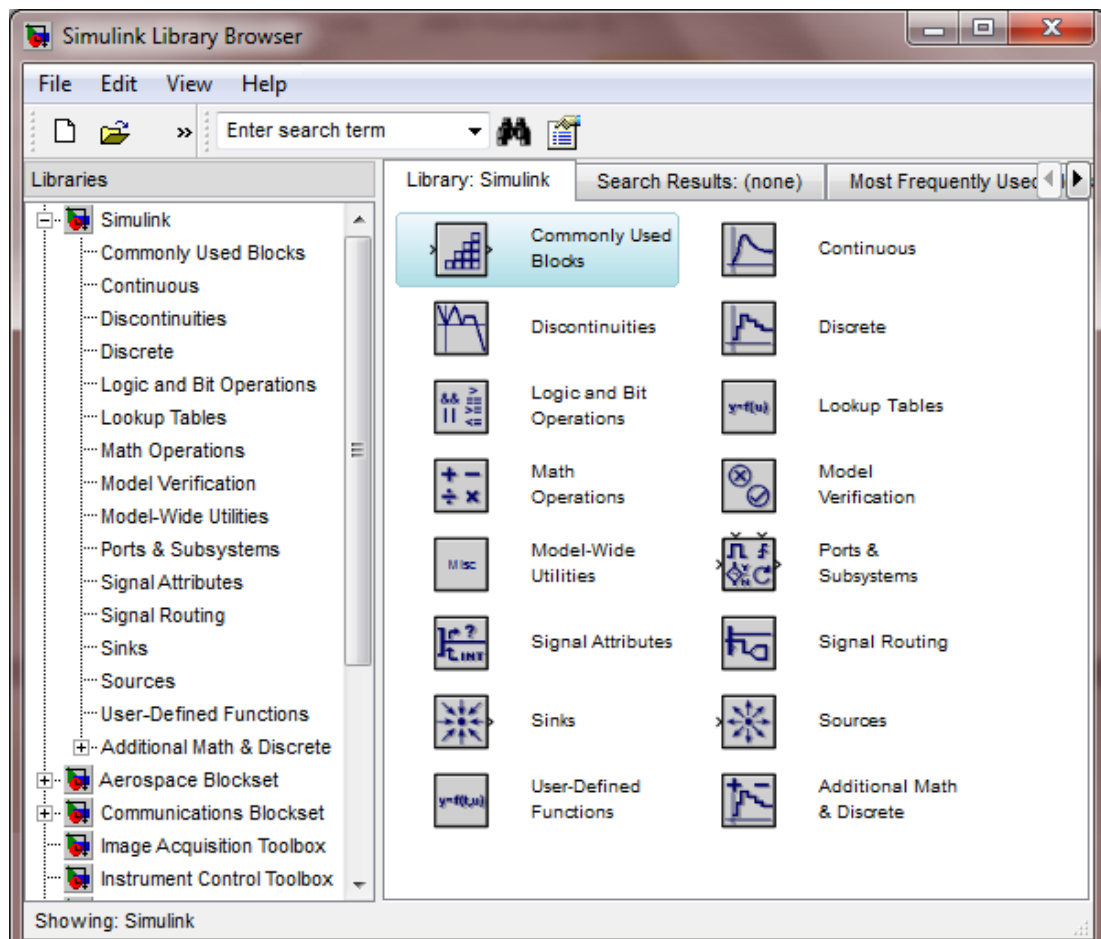
**FIGURA 3.41: SUB-MÁSCARA DEL BLOQUE "DIGITALINPUT"**

En la ventana anterior se muestra la información de entradas, salidas y los parámetros que se pueden modificar, que para este caso es la identificación de la entrada digital a la que está conectado el sensor óptico. Además cada bloque se puede unir con líneas


Teniendo claro estas generalidades se comenzará con la programación:



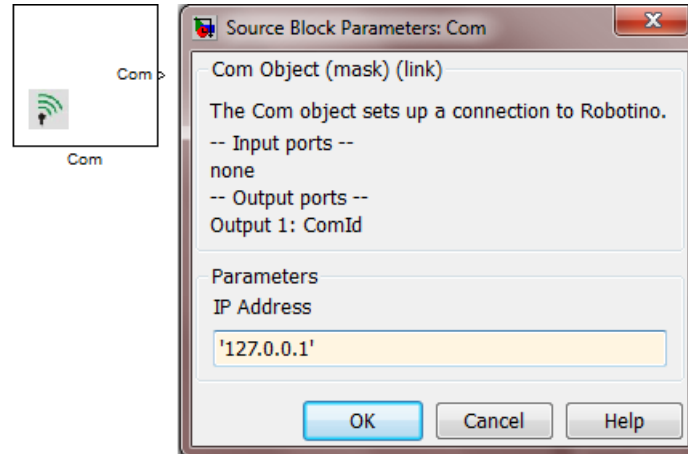
Primero, se escribe “Simulink” en la ventana de comandos de matlab® y se abrirá la ventana de la Figura 3.42 donde se muestran las librerías para programar en Simulink®:



**FIGURA 3.42: LIBRERÍAS DE SIMULINK; PASO 1 PRÁCTICA 11**

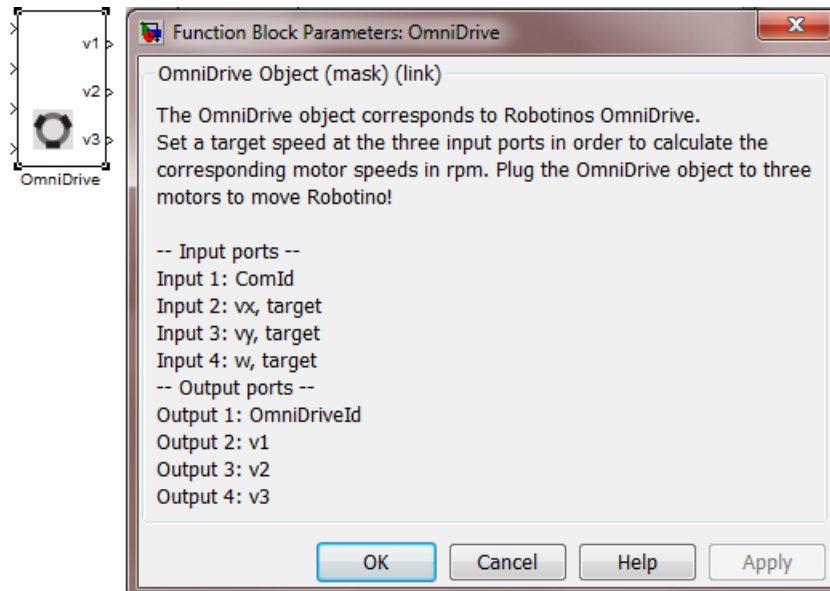
Segundo, para abrir un nuevo proyecto se dará “click” en el icono “”, se abrirá una ventana en blanco y se añadirán los bloques necesarios para este programa:

**Bloque “Com”**: configura la conexión hacia Robotino® (Figura 3.43)



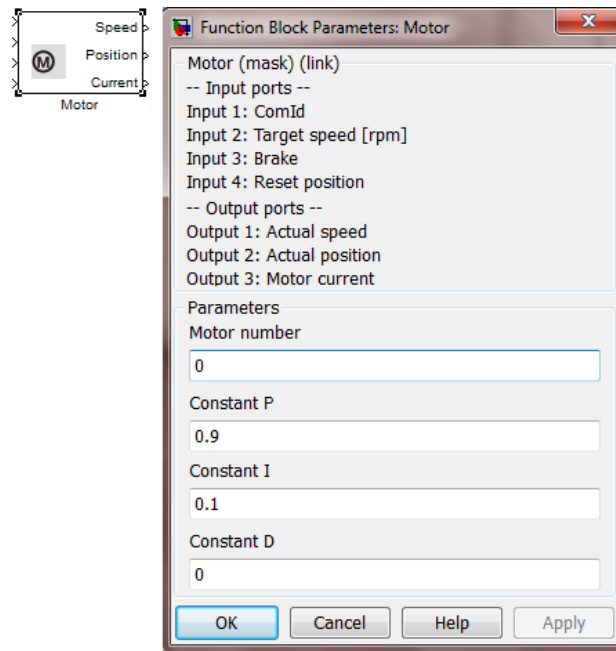
**FIGURA 3.43: BLOQUE "COM" DE SIMULINK®**

**Bloque “OmniDrive”**: este bloque administra los tres motores de la unidad dependiendo de las velocidades que se le ingresen (Figura 3.44)



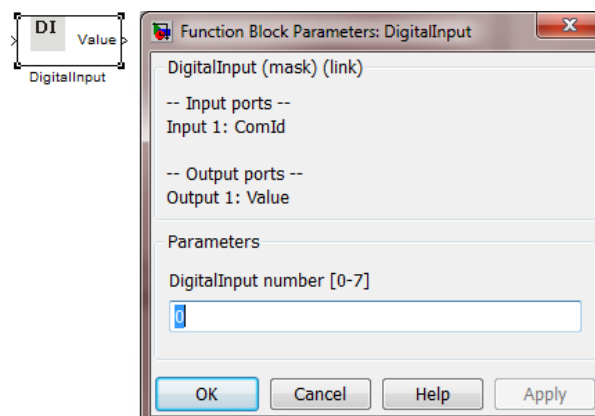
**FIGURA 3.44: BLOQUE "OMNIDRIVE" DE SIMULINK®; PASO 1 PRÁCTICA 11**

**Bloque “Motor”:** administra directamente unos de los motores de Robotino® (Figura 3.45)



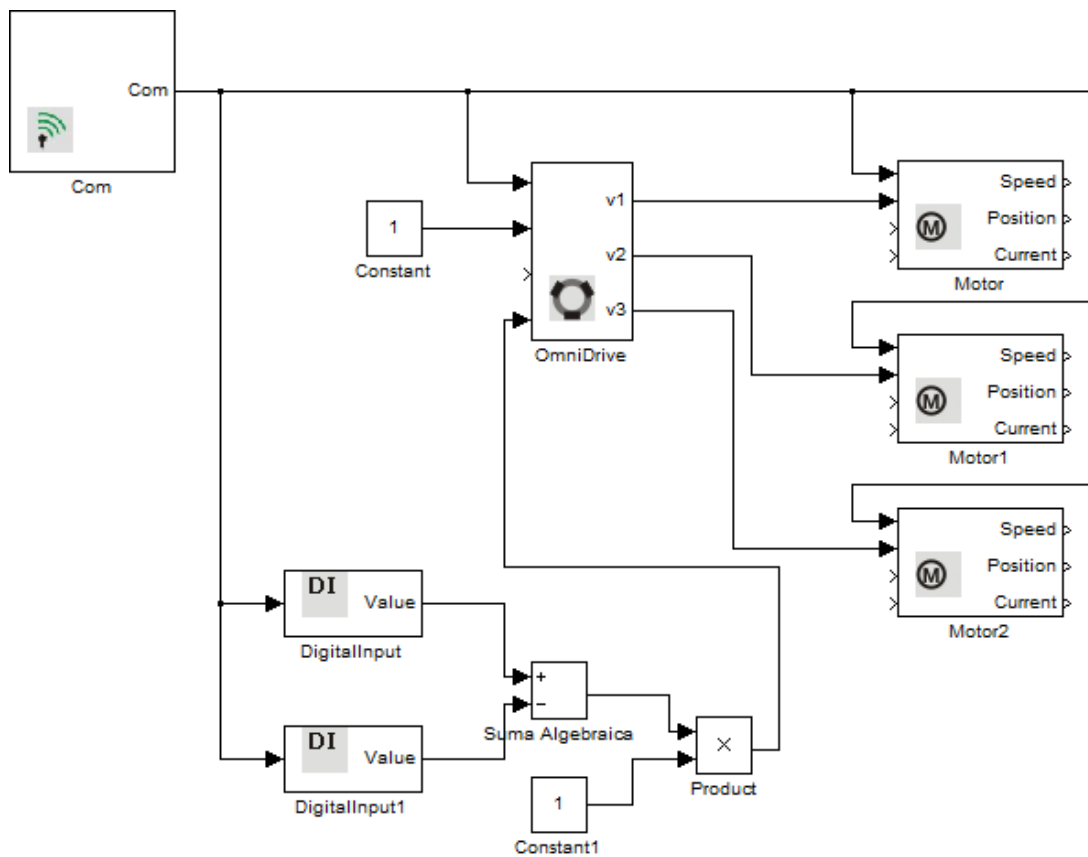
**FIGURA 3.45: BLOQUE "MOTOR" DE SIMULINK®; PASO 1 PRÁCTICA 11**

**Bloque “DigitalInput”:** administra directamente una de las entradas digitales de Robotino® donde estarán conectados los sensores ópticos (Figura 3.46)



**FIGURA 3.46: BLOQUE "DIGITALINPUT" DE SIMULINK®; PASO 1 PRÁCTICA 11**

Finalmente el programa (Figura 3.47) quedará así:



**FIGURA 3.47: PROGRAMACIÓN DEL MOVIMIENTO GUIADO DEL S.D.RO.M. EN SIMULINK®; PASO 1 PRÁCTICA 11**

### OBSERVACIONES

---

Anote las observaciones hechas

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

### CUESTIONARIO

---

- Indique las ventajas y desventajas entre la programación en pseudocódigo y la programación en un entorno visual.

## **CAPÍTULO 4:**

# **4 CONCLUSIONES Y RECOMENDACIONES.**

En conclusión, esta tesis deja como resultado, guías de laboratorio, ya probadas por estudiantes de la asignatura “Sistemas de Control” para un mayor entendimiento y afianzamiento de ideas, de lo dictado en clases.

Además se desea promover que el alumno se motive con el autoaprendizaje, a buscar mayores conocimientos y desarrollos para su carrera

Verá como varias áreas de estudio, pueden compenetrarse en un proceso multidisciplinario, dando como resultado las mejoras de las tecnologías aplicadas a su carrera.

Comprenderá como la implementación de sensores, en elementos mecánicos y/o eléctricos, puede aumentar ampliamente la eficiencia de los mismos y del proceso o sistema en general al que son aplicados.

Se aplicará los conocimientos adquiridos por el educando, para comprender el comportamiento, control y regulación de sistemas o procesos mediante sensores; además de los fenómenos que se puedan producir, de cómo evitarlos, reducirlos y/o regularlos.

Las siete primeras prácticas, buscan que, el estudiante se familiarice con la unidad robótica, cada uno de sus componentes y su programa de control, realizando tareas de comunicaciones y programando diferente tipos

de movimientos o ejecutando acciones. Además se da ejemplos de cómo los sensores pueden ser aplicados de la misma manera u otras formas, en diferentes proyectos que se presenten o se desee realizar.

Se analizan los comportamientos, rangos y fluctuaciones de cada uno de los detectores; que tiene como consecuencia la indagación de varias maneras de regulación, para que cumplan con una tarea propuesta.

Asimismo, se insta al alumno a encontrar y/o intentar formas en las que una máquina o robot, pueda realizar tareas de manera autónoma, que antes eran hechas por seres humanos; buscando aumentar la eficiencia, incrementar la productividad, mejorar la calidad, evitar riesgos, entre otros.

En las últimas cuatro prácticas se da una introducción del manejo de la unidad Robótica a través de MatLab® 7 y cómo es posible expandir las posibilidades, con el uso de una herramienta de control diferente.

MatLab® con la administración de archivos “.m” nos da la posibilidad de examinar internamente en cada uno de los objetos que representan los

componentes y bloques de control del S.D.Ro.M., para así, verificar, mejorar y/o corregir el código de los mismos.

Como se explicó anteriormente, se escoge MatLab®, porque está relacionado directamente en varias de las asignaturas dictadas en la carrera, por este motivo, los alumnos tienen conocimientos previos al manejo de este programa y se considera que el educando se adaptará rápidamente al uso de Robotino® con este software de control y su programación.

Como conclusión final, se debe aclarar que las guías expuestas en este trabajo, ya han sido probadas y se deja a la Facultad de Ingeniería en Mecánica y Ciencias de la Producción de la Escuela Politécnica del Litoral, un manual para las prácticas del laboratorio de Mecatrónica aplicados al a unidad didáctica robótica móvil Robotino®, que pueden ser realizadas con total facilidad, agilidad y aplicando conocimiento ingenieriles.

Se recomienda seguir, el orden que se ha establecido en las guías, debido a que van acorde a un ritmo escalado de complejidad; además que, siguen el plan actual de lo que se dicta en clases de Sistema de Control.



Robotino® es un dispositivo robusto, fabricado de materiales de alta resistencia, pero de todas formas es necesario extremar sus cuidados; que por ser un equipo electrónico, poseedor de elementos sensibles y delicados, se aconseja que antes de su manipulación, se dedique un tiempo a examinar la unidad, identificar sus componentes y proponer acciones a tomar frente a los mismos.

De esta manera se recomienda que la FIMCP proporcione repuestos o equipos que sean necesarios en adquisición o remplazo, por culminación de su tiempo de vida o desperfectos.

El responsable del laboratorio, deberá establecer un programa de mantenimiento para el sistema y sus componentes, incluyendo las baterías de tipo plomo-gel que tiene el S.D.Ro.M. las cuales son productos perecibles en el tiempo y deberá recibir una atención especial evitando que entren en “descarga profunda”, recordando darles ciclo de carga pertinentes para alargar su vida útil. Un buen mantenimiento de las mismas influirá en la calidad de ejecución de tareas realizadas por la unidad robótica. Los demás componentes, tendrán un mantenimiento preventivo, revisando

periódicamente su estado, conexiones, contactos, entre otros; tomando en cuenta su limpieza para evitar que cualquier suciedad intervenga en la toma de datos del sensor.

Igualmente se sugiere al laboratorista que tenga todos los equipos e insumos preparados antes de cada práctica para evitar contratiempos en la misma.

Finalmente, esta tesis fue hecha para introducir al alumno al uso y control de un sistema robótico , por lo que se recomienda, explotar al máximo todas las posibilidades que tiene la unidad, realizando nuevas prácticas a manera de proyectos o combinando las ya existentes, para ampliar o complementar las tareas y aplicaciones que se le puede dar la unidad.

# APÉNDICE

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°1

**TÍTULO**

---

Robotino®view 2: Control de las comunicaciones, puesta en funcionamiento y programación de movimientos lineales en sentidos indistintos del sistema robótico

**OBJETIVOS**

---

- ✓ Conocer los componentes más importantes de un Sistema Didáctico Robótico Móvil (S.D.Ro.M.).
- ✓ Poner en funcionamiento, probar y explicar los movimientos que ejecuta el S.D.Ro.M. Robotino® y los grados de libertad que posee, mediante Robotino®View 2.
- ✓ Describir y programar movimientos sencillos con bloques de funciones y secuencias en Robotino®View 2; tomando en cuenta aspectos de seguridad en el caso de una colisión del S.D.Ro.M.

**MARCO TEÓRICO**

---

**Identificación de componentes del S.D.Ro.M. Robotino®**

Llenar con lo explicado en clases y complementar con una breve investigación de cada uno de los componentes.

**Control de recepción y envío de datos de la unidad robótica**

**Robotino**

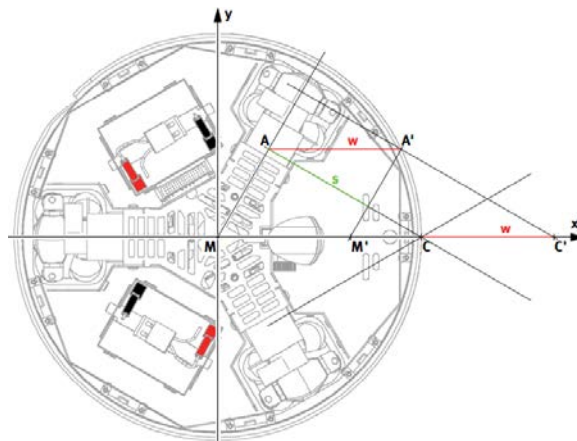
Escriba sus apuntes de lo visto en clase del laboratorio.

## Movimientos lineales y posicionamiento de un sistema robótico

Como ya se ha hablado anteriormente el S.D.Ro.M. Robotino® posee un sistema accionado por 3 ejes distribuidos a 360 grados equitativamente.

Este tipo de distribución da la ventaja de tener grados de libertad independientes en nuestro sistema robótico, pero se presenta otro problema a resolver, que será: determinar la distancia recorrida por la unidad, en relación al número de giros que den los motores

Considerando que Robotino® ejecuta un movimiento hacia delante y recorre un tramo “w” y sus ruedas recorren un tramo “s”



### ILUSTRACIÓN 1: CÁLCULO DE RECORRIDO SIN USAR BLOQUE DE OMNIDIRECCIONAMIENTO

Analizando la imagen se obtiene:

$$(5) s=w \cdot \sin(60^{\circ})$$

Además se sabe que  $P=d \cdot \pi$  (donde  $d=80\text{mm}$  es el diámetro de las ruedas dadas en las especificaciones) por lo que se puede inferir la cantidad de giros que hace la rueda para un tramo “s” como:

$$(6) \quad G_i = \frac{s}{P}$$

Reemplazando (1) en (2) finalmente se tiene que:

$$(7) \quad G_i = \frac{w \cdot \sin(60^\circ)}{d \cdot \pi}$$

$G_i$ : cantidad de giros.

$w$ : tramo recorrido por la unidad.

$d$ : diámetro de las ruedas.

Se debe recordar que las unidades de accionamiento del S.D.Ro.M. tiene un contador de pulsos o incrementos (encoder) y consultando la documentación técnica dice que se dan 2048 incrementos por cada giro del motor eléctrico. Además cada motor tiene reductor cuya relación es de 1:16, con lo que se puede determinar la *cantidad de incrementos* ( $C_i$ ) que deben de dar los motores para que el sistema recorra un tramo “w”:

$$(8) \quad C_i = \frac{w \cdot \sin(60^\circ)}{d \cdot \pi} \times 2048 \times 16$$

## EQUIPOS Y MATERIALES UTILIZADOS

✓ S.D.Ro.M. Robotino®.

- ✓ Computadora con capacidad de conexión inalámbrica WiFi.

## PROCEDIMIENTO EXPERIMENTAL

1. Elaborar una tabla de comprobación y control visual del sistema completo
2. Comprobar el funcionamiento de los componentes, observando que la pantalla de la unidad de mando se muestre correctamente y verificando el estado de carga del S.D.Ro.M.
3. Comprobar los movimientos ejecutados por Robotino®, activando las aplicaciones de prueba.
4. En el programa Robotino®View confeccionar un programa con los bloques de función de los componentes y observar su comportamiento.
5. Confeccione otro programa para que Robotino® avance hacia adelante, adicionando una función de protección en caso de colisiones.
6. Determine la cantidad de giros e incrementos que deben efectuar las ruedas para que la unidad avance un tramo de 1m.
7. Confeccione un programa para probar los resultados del punto 6, mida la distancia de 1m y explique por qué se obtienen resultados diferentes al resultado nominal.
8. Pruebe el programa anterior a diferentes velocidades y tipos de suelo

## OBSERVACIONES

---

Anote las observaciones hechas

Además use la siguiente tabla para apoyar sus observaciones y calcule la desviación nominal del recorrido real del S.D.Ro.M.

Trayecto recorrido en metros	Desviación del valor nominal (1m)
$\bar{X} =$ n=10	$\bar{X} =$ n=10

**TABLA: OBSERVACIÓN DEL RECORRIDO REAL DEL S.D.RO.M.,  
PRÁCTICA 1**

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

---

- ¿Qué relación existe entre el movimiento de las ruedas y el comportamiento de Robotino® cuando ejecuta los movimientos?



- ¿Qué motores deben accionarse para que Robotino® avance hacia delante? Y explique por qué, al avanzar hacia delante, los motores deben girar en sentido contrario
- Describa los movimientos e indique los grados de libertad observados (movimientos posibles de los cuerpos y el sistema completo).
- Explique por qué surgen desviaciones en relación con el trayecto nominal de 1m y genere un concepto para optimizar el programa.

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°2

TÍTULO

---

Sensores de reflexión directa: programación del movimiento guiado del sistema robótico.

OBJETIVOS

---

- ✓ Aprender sobre el montaje y conexión de los sensores de reflexión directa.
- ✓ Evaluar y calibrar las señales de un sensor de reflexión directa.
- ✓ Desarrollar una estrategia para la ejecución de movimientos guiados de un sistema de transporte sin conductor, elaborando un programa para aplicar, probar y optimizar las ideas planteadas

MARCO TEÓRICO

---

**Detectores ópticos**

Llenar con lo explicado en clases y complementar con una breve investigación acerca de los detectores ópticos.

**Calibración de los detectores ópticos**

Cada uno de los detectores ópticos está provisto de potenciómetros que regulan la sensibilidad de los detectores.

Esta sensibilidad está basada en la diferencia de reflexión ocasionada por el objeto y por el trasfondo. Si el contraste es débil, es posible ajustar el umbral de respuesta mediante el potenciómetro, de modo que sea posible detectar el objeto bajo las condiciones más difíciles.

Para efectuar un ajuste correcto, deberá preverse un margen de tolerancia considerando posibles cambios en las características de las superficies, ensuciamiento del detector y/o presencia de contaminantes en el ambiente.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Hoja de papel blanco con marca “negra”
- ✓ Cinta adhesiva negra.

## PROCEDIMIENTO EXPERIMENTAL

1. Montar los detectores ópticos de reflexión directa en los lugares previstos del chasis de Robotino®.
2. Conectar los detectores a la interface E/S, identificando cual será “derecho o izquierdo”

3. Calibrar los sensores mediante el ajuste del potenciómetro y la ayuda del programa Robotino®View.
4. Medite y analice con sus compañeros una forma para lograr que el S.D.Ro.M. recorra un trayecto fijado, de manera autónoma con el uso de los detectores ópticos de reflexión directa.
5. Elabore un programa en Robotino®View aplicando las ideas derivadas del punto anterior.

#### OBSERVACIONES

Anote las observaciones hechas

#### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Analice y concluya, sobre sus resultados.

#### CUESTIONARIO

- Piense que situación se produce cuando Robotino® ejecuta un giro en sentido anti-horario. En ese caso, ¿Qué detector deberá consultarse?
- ¿Qué puede suceder si los dos sensores quedan encima de la marca negra? ¿Cómo se puede solucionar este problema?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°3

**TÍTULO**

---

Detector analógico inductivo: comportamiento frente a diversos tipos de materiales y programación del movimiento guiado del sistema robótico.

**OBJETIVOS**

---

- ✓ Aprender sobre el montaje y conexión de un detector analógico inductivo.
- ✓ Evaluar y calibrar las señales de un detector analógico inductivo.
- ✓ Desarrollar un método para reconocer la diferencia entre tipos de materiales metálicos
- ✓ Desarrollar una estrategia para la ejecución de movimientos guiados de un sistema de transporte sin conductor.
- ✓ Elaborar un programa secuencial para aplicar, probar y optimizar las ideas planteadas en el punto anterior.

**MARCO TEÓRICO**

---

**Detector analógico inductivo**

Llenar con lo explicado en clases y complementar con una breve investigación sobre detectores inductivos.

**Calibración de los sensores analógicos inductivos**

Al igual que los detectores ópticos cada sensor inductivo está provisto de potenciómetros que regulan su sensibilidad.

Por lo que la calibración es parecida a la de los sensores ópticos y dependerá de la tarea que se desee realizar:

- Diferenciación entre tipos de materiales metálicos.
- Medición de distancias.

En la diferenciación de materiales se deberá tener el material “Patrón” que se desea detectar y así definir el rango de reacción del sensor.

Por otra parte, para la medición de distancias se supone que el material detectado no se altera, por lo que la variabilidad de las mediciones en el sensor se deberá al cambio de posición relativa entre ellos. En este caso, aparte de la calibración, también habrá que establecer una relación “distancia Vs voltaje” y quedará a criterio del lector, el método más eficiente a utilizar.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Elemento metálico.
- ✓ Cinta de aluminio ( $\geq 5\text{cm}$ ).

## PROCEDIMIENTO EXPERIMENTAL

1. Montar el detector análogo inductivo en los lugares previstos del chasis de Robotino®.
2. Conectar el detector a la interface E/S
3. Comprobar el correcto funcionamiento del sensor con la ayuda del programa Robotino®View.
4. Considere la manera y elabore un programa para la diferenciación de materiales.
5. Medite y analice con sus compañeros una forma para lograr que el S.D.Ro.M. recorra un trayecto fijado, de manera autónoma con el uso del detector

### OBSERVACIONES

Anote las observaciones hechas

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Analice y concluya, sobre sus resultados.

### CUESTIONARIO

- Para el montaje del detector, ¿Qué debe tenerse en cuenta?
- ¿Cómo se optimizaría el programa de tal manera que los movimientos sean más fluidos?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°4

TÍTULO

Detectores de distancias infrarrojos: línea característica (curva de calibración), avance en función de distancias precisas y mantenerlas, bordear o evitar obstáculos con el sistema robótico.

OBJETIVOS

- ✓ Conocer el comportamiento y el funcionamiento de los detectores de distancia por luz infrarroja del S.D.Ro.M.
- ✓ Determinar y registrar la línea de calibración de los detectores de distancia y saber cómo usarla para relacionarla con la medición distancias.
- ✓ Elaborar un programa de regulación donde la unidad sea capaz de acercarse a un obstáculo y mantener una distancia definida respecto al mismo.
- ✓ Elaborar varias estrategias para que Robotino® mantenga distancias ante un obstáculo y/o lo bordee

MARCO TEÓRICO

**Detectores de distancias infrarrojos.**



Llenar con lo explicado en clases y complementar con una breve investigación sensores de distancia infrarrojos.

### **Curva de calibración de los sensores de distancia infrarrojos**

A diferencia de los sensores ópticos y de inducción, estos detectores normalmente no poseen tornillo de calibración pero es posible obtener una curva característica o de calibración. La que nos ayudará establecer una relación entre el valor mostrado (Voltaje) por el sensor y las distancias (centímetros) que tenemos como objetivo medir.

Para establecer esta relación primero se tiene que escoger un método de linealización de ajuste de curva, el cual va a depender de la aplicación que se le dé al sensor, con lo que será potestad del estudiante u operador elegir el método que más se acomode y que sea más eficiente para cumplir con las necesidades que requiera.

### **EQUIPOS Y MATERIALES UTILIZADOS**

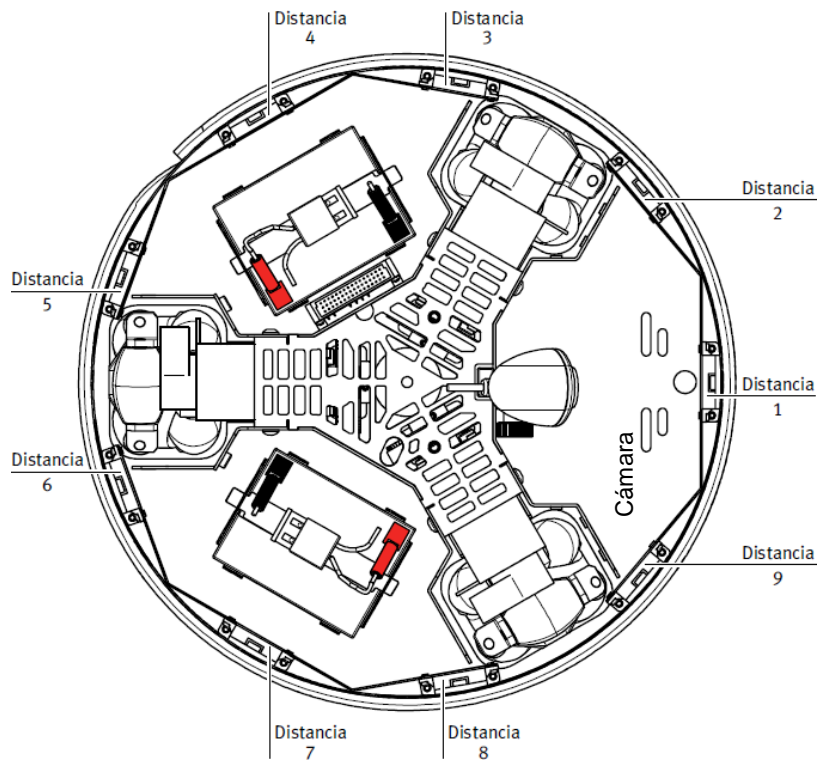
✓ S.D.Ro.M. Robotino®.

- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Marcador de pizarra acrílica.
- ✓ Cinta métrica o flexómetro.
- ✓ Pliego de papel o pizarra acrílica.
- ✓ Cualquier objeto que servirá de obstáculo

## PROCEDIMIENTO EXPERIMENTAL

1. Identificar cada uno de los sensores que tiene Robotino® y verificar su correcto funcionamiento.
2. Registrar la línea característica del sensor de distancia y establecer relación distancia vs tensión, linealizándola.
3. Elaborar la estrategia a tomar para que el S.D.Ro.M. avance desde una posición y se detenga ante un obstáculo, a una distancia especificada .Adicionalmente genere un programa para realizar esta acción.
4. Realizar otro programa para que Robotino® se desplace a lo largo de una pared manteniendo una distancia específica y únalo en secuencia con el programa anterior.
5. Elaborar una nueva estrategia y programa para lograr que la unidad robótica gire o bordee un obstáculo de forma irregular.

**Solución:** Para realizar el punto 1, se posiciona a Robotino® sobre su base y se identifica sus sensores con la ayuda de la ilustración 2.



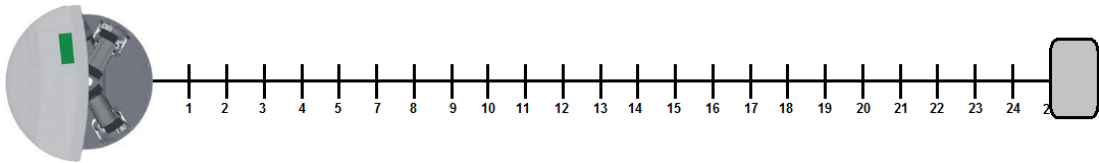
**ILUSTRACIÓN 2: ESQUEMA PARA IDENTIFICAR CADA SENSOR DE DISTANCIA DE ROBOTINO®; PASO 1 PRÁCTICA 4.**

Verifique el funcionamiento de los sensores con la siguiente tabla

Detector	Funcionamiento OK	Detector	Funcionamiento OK
Distancia 1	<input type="checkbox"/>	Distancia 6	<input type="checkbox"/>
Distancia 2	<input type="checkbox"/>	Distancia 7	<input type="checkbox"/>
Distancia 3	<input type="checkbox"/>	Distancia 8	<input type="checkbox"/>
Distancia 4	<input type="checkbox"/>	Distancia 9	<input type="checkbox"/>
Distancia 5	<input type="checkbox"/>		

**TABLA: TABLA PARA VERIFICACIÓN DE LOS DETECTORES DE DISTANCIA; ; PASO 1 PRÁCTICA 4.**

Para registrar la línea característica del sensor del paso 2 (*Ilustración 3*), se tenderá la pizarra acrílica o el pliego de papel sobre el piso y se trazan marcas a 1 cm de espaciamiento a lo largo de un tramo de 25 a 30cm.



**ILUSTRACIÓN 3: ESQUEMA PARA REGISTRAR LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4.**

Y nos ayudaremos con la tabla siguiente:

Distancia (cm)	Tensión (V)	Distancia (cm)	Tensión (V)
1		14	
2		15	
3		16	
4		17	
5		18	
6		19	
7		20	
8		21	
9		22	
10		23	
11		24	
12		25	
13			

**TABLA: REGISTRO DE LA CURVA DE CALIBRACIÓN DEL SENSOR DE DISTANCIA; PASO 2 PRÁCTICA 4.**

**OBSERVACIONES**

---

Anote las observaciones hechas

**ANÁLISIS DE RESULTADOS Y CONCLUSIONES**

---

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

---

- ¿Cómo puede mejorarse el efecto del regulador?
- Piense cómo puede obtener un movimiento circular con la ayuda de bloque de funciones de accionamiento omnidireccional. ¿Qué sucede y porque?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°5

## TÍTULO

---

Motores: control, calibración movimiento lineales y posicionamiento del sistema robótico

## OBJETIVOS

---

- ✓ Obtener conocimientos básicos acerca de la técnica de regulación de motores eléctricos.
- ✓ Ajustar y optimizar los parámetros de un regulador P.I.D. mediante el software correspondiente.
- ✓ Describir y evaluar los efectos que tiene el ajuste de las constantes del control P.I.D. en los movimientos ejecutados por el robot.

## MARCO TEÓRICO

---

### **Unidades de accionamiento de Robotino®**

Llenar con lo explicado en clases y complementar con una breve investigación, motores “de paso” (Brushless), y ruedas omnidireccionales.

### **Método para ajuste de las constantes PID:**

### **Método De Ziegler-Nichols**

En las primeras aplicaciones del control PID el ajuste se basaba únicamente en la experiencia del operador de planta.

En procesos lentos cada prueba de sintonía puede llevar horas e incluso días.

Para solucionar estos problemas Ziegler y Nichols (1942) propusieron técnicas empíricas para el ajuste de PID no interactivos, obtenidas tras numerosas pruebas y sin presuponer ningún conocimiento de la planta a controlar.

Existen dos métodos de Ziegler-Nichols

- Ziegler-Nichols en lazo abierto
- Ziegler-Nichols en lazo cerrado

En ambos métodos Ziegler-Nichols, el objetivo es conseguir que: el valor del Máximo sobre-impulso sea menor del 25% para una entrada en escalón

### **Sintonización de controladores PID**

Debido a que casi todos los controladores PID se ajustan en el sitio, en la literatura se han propuesto muchos tipos diferentes de sintonización delicada y fina de los controladores PID en el sitio. Asimismo, se han

desarrollado métodos automáticos de sintonización y algunos de los controladores PID poseen capacidad de sintonización en línea.

### **Control PID de plantas:**

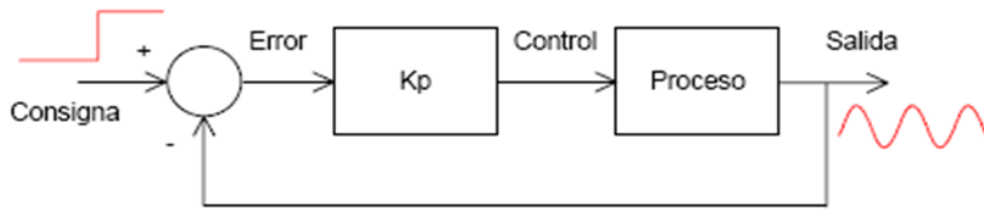
Si se puede obtener un modelo matemático de la planta, es posible aplicar diversas técnicas de diseño con el fin de determinar los parámetros del controlador que cumpla las especificaciones en estado transitorio y en estado estable del sistema en lazo cerrado. Sin embargo, si la planta es tan complicada que no es fácil obtener su modelo matemático, tampoco es posible un enfoque analítico para el diseño de un controlador PID. En este caso se debe recurrir a los enfoques experimentales para la sintonización de los controladores PID.

### **Método de Ziegler-Nichols en lazo cerrado:**

Este método se basa en que la mayoría de los procesos pueden oscilar de forma mantenida bajo control proporcional con una ganancia adecuada (*Ilustración 4*):

- Ganancia crítica ( $K_{cr}$ )
- Periodo de oscilación mantenida



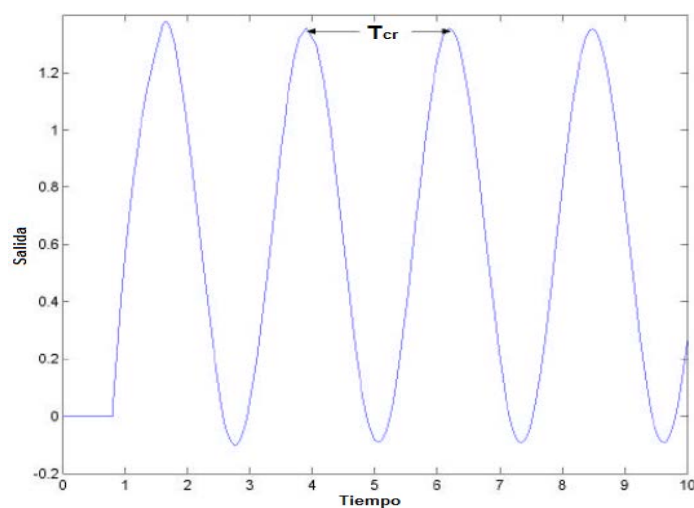


**ILUSTRACIÓN 4: PROCESO CON RESPUESTA OSCILATORIA MANTENIDA ANTE UNA PERTURBACIÓN DE ENTRADA ESCALÓN; MÉTODO DE SINTONIZACIÓN PID ZIEGLER-NICHOLS EN LAZO CERRADO.**

Esta técnica de respuesta en frecuencia es un método alternativo de sintonización de PID que puede describirse como sigue:

En primer lugar es necesario ajustar las ganancias integral y derivativa a cero, esto es  $K_i = 0$  y  $K_d = 0$

A continuación, partiendo de un valor bajo de la ganancia proporcional,  $K_p$ , se va aumentando ésta gradualmente hasta conseguir un comportamiento oscilatorio mantenido en la respuesta del sistema tal como muestra la Ilustración 5. A esta ganancia se la identificará  $K_{cr}$ .



**Ilustración 5: SINTONIZACIÓN PID: AJUSTE DE  $K_p$**

Se llama ganancia crítica porque se aumenta hasta llegar al borde de la inestabilidad en el sistema

El otro parámetro que hace falta es el periodo de oscilación del sistema para esta ganancia, que se identificará como  $T_{cr}$ , y que se calcula gráficamente por la Ilustración 5.

Con los valores de  $K_{cr}$  y  $T_{cr}$  se entra a la tabla a continuación de Ziegler-Nichols y se calcula los parámetros correspondientes.

Tipo de controlador	$K_p$	$K_i$	$K_d$
P	$0.5 \cdot K_{cr}$	$\infty$	0
PI	$0.45 \cdot K_{cr}$	$1/1.2 \cdot T_{cr}$	0
PID	$0.6 \cdot K_{cr}$	$0.5 \cdot T_{cr}$	$0.125 \cdot T_{cr}$

**TABLA: DE ZIEGLER- NICHOLS PARA EL CÁLCULO DE LAS CONSTANTES PID**

#### EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Base de Robotino®

## PROCEDIMIENTO EXPERIMENTAL

---

1. Confeccionar un programa o diagrama de bloques en Robotino®View 2 para realizar el ajuste de las constantes PID.
2. Colocar la unidad sobre su base y realizar el procedimiento de sintonización de Ziegler-Nichols con cada uno de los motores.
3. Repetir el procedimiento de sintonización, asentando la unidad sobre una superficie y dejando que recorra libremente.
4. Comparar ambos resultados.
5. Analizar el sistema ante una entrada “escalón”

## OBSERVACIONES

---

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

## CUESTIONARIO

---

- En la práctica observe el comportamiento de la rueda omnidireccional. Describa y explique dicho comportamiento.
- En el ajuste de los parámetros del regulador PID. Describa el efecto que tiene en los movimientos. Consulte lo que significan los parámetros P, I y D, ¿Cuáles son sus efectos?
- Considere el margen de valores para los parámetros de regulación, dentro del que los movimientos son aceptables.

- ¿Qué influencia tiene en los movimientos las oscilaciones grandes?
- Explique por qué existen diferencias entre las dos curvas (con Robotino® elevado y sobre el suelo).

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°6

**TÍTULO**

---

Cámara digital: detección de una pieza de diferentes colores y ejercicios de aplicación.

**OBJETIVOS**

---

- ✓ Conocer el funcionamiento y utilización de la cámara de Robotino®.
- ✓ Configurar las funciones de detección de colores de Robotino®View y saber usarlas.
- ✓ Conocer las limitaciones y condiciones que se tienen la detección de colores y objetos.

**MARCO TEÓRICO**

---

**Cámara de Robotino® (Webcam)**

Llenar con lo explicado en clases y se desea complementarse con una breve investigación acerca de cámaras y uso en la robótica.

**Reconocimiento de patrones (colores o sonido)**

El reconocimiento de patrones es la ciencia que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados

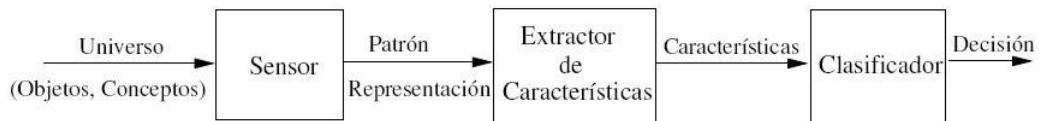
con objetos físicos o abstractos, con el propósito de extraer información que permita establecer propiedades de entre conjuntos de dichos objetos.

Los patrones se obtienen a partir de los procesos de segmentación, extracción de características y descripción dónde cada objeto queda representado por una colección de descriptores. El sistema de reconocimiento debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que las diferencia del resto). Para poder reconocer los patrones (*Ilustración 6*) se siguen los siguientes procesos:

- adquisición de datos
- extracción de características
- toma de decisiones

El punto esencial del reconocimiento de patrones es la clasificación: se requiere clasificar una señal dependiendo de sus características.

Estas señales, características y clases pueden ser de cualquiera forma, por ejemplo se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiendo de sus píxeles o se puede clasificar ruidos de cantos de los pájaros en clases de órdenes aviares dependiendo de las frecuencias.



**ILUSTRACIÓN 6: ESQUEMA PARA EL RECONOCIMIENTO DE PATRONES E IMÁGENES**

### **Aplicaciones**

Los sistemas de reconocimiento de patrones tienen diversas aplicaciones. Algunas de las más relevantes y utilizadas actualmente son:

- **Previsión meteorológica:** poder clasificar todos los datos meteorológicos según diversos patrones, y con el conocimiento a priori que tenemos de las diferentes situaciones que pueden aparecer nos permite crear mapas de predicción automática.
- **Reconocimiento de caracteres escritos a mano o a máquina:** es una de las utilidades más populares de los sistemas de reconocimiento de patrones ya que los símbolos de escritura son fácilmente identificables.
- **Reconocimiento de voz:** el análisis de la señal de voz se utiliza actualmente en muchas aplicaciones, un ejemplo claro son los teleoperadores informáticos.

- **Aplicaciones en medicina:** análisis de biorritmos, detección de irregularidades en imágenes de rayos-x, detección de células infectadas, marcas en la piel.
- **Reconocimiento de huellas dactilares:** utilizado y conocido por la gran mayoría, mediante las huellas dactilares todos somos identificables y con programas que detectan y clasifican las coincidencias, resulta sencillo encontrar correspondencias.
- **Reconocimiento de caras:** utilizado para contar asistentes en una manifestación o simplemente para detectar una sonrisa, ya hay diferentes cámaras en el mercado con esta opción disponible.
- **Interpretación de fotografías aéreas y de satélite:** gran utilidad para propuestas militares o civiles, como la agricultura, geología, geografía, planificación urbana.
- **Reconocimiento de objetos:** con importantes aplicaciones para personas con discapacidad visual.
- **Reconocimiento de música:** identificar el tipo de música o la canción concreta que suena.

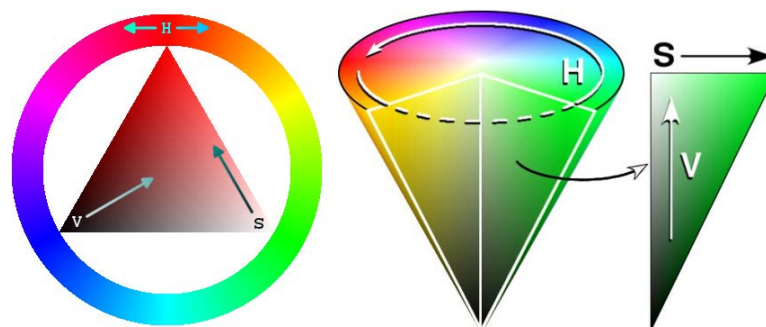
### **Modelo de color “HSV”**

El modelo HSV (Hue, Saturation, Value – Matiz, Saturación, Valor), también llamado HSB (Hue, Saturation, Brightness – Matiz, Saturación, Brillo), define un modelo de color en términos de sus componentes.



Tratándose de una transformación no lineal del espacio de color RGB, y se puede usar en progresiones de color.

Es común que se desee elegir un color adecuado para alguna de nuestras aplicaciones, cuando es así, resulta muy útil usar la ruleta de color HSV (*Ilustración 7*). En ella el matiz se representa por una región circular; una región triangular separada, puede ser usada para representar la saturación y el valor del color. Normalmente, el eje horizontal del triángulo denota la saturación, mientras que el eje vertical corresponde al valor del color. De este modo, un color puede ser elegido al tomar primero el matiz de una región circular, y después seleccionar la saturación y el valor del color deseados de la región triangular.



**Ilustración 7: ESPACIO DE COLOR “HSV”**

### **Matiz**

Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al

100%). Cada valor corresponde a un color. Ejemplos: 0 es rojo, 60 es amarillo y 120 es verde.

### **Saturación**

Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100%. A este parámetro también se le suele llamar "pureza" por la analogía con la pureza de excitación y la pureza colorimétrica de la colorimetría. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. Por eso es útil definir la insaturación como la inversa cualitativa de la saturación.

### **Valor**

Representa la altura en el eje blanco-negro. Los valores posibles van del 0 al 100%. 0 siempre es negro. Dependiendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

### **EQUIPOS Y MATERIALES UTILIZADOS**

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi.

- ✓ Cualquier objeto que servirá para el reconocimiento del patrón

## PROCEDIMIENTO EXPERIMENTAL

1. Elaborar un programa para el reconocimiento de patrones de colores.
2. Elaborar un programa para que Robotino® busque y se acerque a un objeto.


## OBSERVACIONES

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

- ¿Qué sugerencia puede dar, para evitar fallos por el cambio de iluminación?
-  Con referencia a la figura anterior, ¿Qué problemas se pueden dar en el modelo HSV cuando el objeto presenta matices de colores y no un solo color? ¿Qué problema se puede dar específicamente con la figura anterior donde se empieza con violeta y se termina en amarillo?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°7

TÍTULO

---

Movimiento guiado del sistema robótico mediante el uso de una cámara digital

OBJETIVOS

---

- ✓ Conocer el funcionamiento y utilización de la cámara de Robotino®.
- ✓ Configurar y usar el bloque de función “Detector de líneas”.
- ✓ Lograr movimientos guiados usando la cámara de Robotino®

MARCO TEÓRICO

---

**Cámara de Robotino® (Webcam)**

Llenar con lo explicado en clases

**Reconocimiento de línea por medio de la cámara de Robotino®.**

Esto se logra con el bloque de funciones “Detector de líneas”, el cual aplica el mismo método de reconocimiento de patrones explicado en la sección 3.6, con la diferencia que este bloque es totalmente dedicado a la detección o reconocimiento de líneas en trayectos marcados.

EQUIPOS Y MATERIALES UTILIZADOS

---

- ✓ S.D.Ro.M. Robotino®.

- ✓ Computadora con capacidad de conexión inalámbrica WiFi.
- ✓ Pista previamente marcada para el recorrido.

## PROCEDIMIENTO EXPERIMENTAL

1. Establecer una estrategia para el reconocer y el seguir de una línea.
2. Elaborar un programa para el movimiento guiado mediante el uso de la cámara de Robotino®

## OBSERVACIONES

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

## CUESTIONARIO

- ¿Qué ventajas y desventajas tiene el movimiento guiado por una cámara contra el movimiento guiado por los sensores ópticos y el de inducción?
- ¿Cómo debe estar orientada la cámara para que pueda detectar la línea de guía delante de Robotino®?
- Explique la influencia que tienen diversos colores en la capacidad de la cámara de detectar la línea
- ¿Qué puede hacerse para minimizar las influencias que interfieren en la detección correcta de la línea de guía?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°8

**TÍTULO**

---

MatLab® 7: control de comunicaciones, puesta en funcionamiento y programación de tareas básicas del sistema robótico.

**OBJETIVOS**

---

- ✓ Establecer la conexión y el control de comunicaciones de Robotino® mediante MatLab®7
- ✓ Poner en funcionamiento, probar y explicar los movimientos que ejecuta el S.D.Ro.M. Robotino® y los grados de libertad que posee, mediante MatLab®7.
- ✓ Describir y programar movimientos sencillos con las cajas de herramientas "ToolBoxes" en MatLab®7.

**MARCO TEÓRICO**

---

**MatLab® 7 para del sistema didáctico robótico móvil.**

Llenar con lo explicado en clases y complementar con una breve introducción a la programación en MatLab®7.

**EQUIPOS Y MATERIALES UTILIZADOS**

---

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.

## PROCEDIMIENTO EXPERIMENTAL

---

1. Establecer la conexión y el control de comunicaciones entre Robotino® y MatLab® 7.
2. Escribir una secuencia para realizar movimientos básicos con la unidad, usando las cajas de herramientas que se instalaron junto con los controladores de Robotino® para MatLab®

## OBSERVACIONES

---

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

---

- ¿Es posible manejar varias unidades con MatLab® 7? Si su respuesta es positiva describa ¿cómo hacerlo?

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°9

## TÍTULO

---

Detectores de distancia infrarrojos: programación de sistema de exploración a través de MatLab® 7.

## OBJETIVOS

---

- ✓ Conocer la programación y utilización de los sensores de distancia de Robotino® en MatLab® 7.
- ✓ Configurar y usar las cajas de herramientas “DistanceSensor”.
- ✓ Elaborar un programa de exploración donde Robotino® se acerque a los obstáculos y los evite.

## MARCO TEÓRICO

---

### **Cajas de herramientas “DistanceSensor”.**

**DistanceSensor\_construct:** construye el objeto “DistanceSensor”, parecido al bloque de funciones visto en Robotino®View.

**DistanceSensor\_setComId:** asocia el objeto “DistanceSensor” a la interface de comunicación. Además lo liga a un Robotino® específico.

**DistanceSensor\_voltage:** es una función que retorna el valor de voltaje de un sensor de distancia específico.



Ejemplo: `[ voltage ] = DistanceSensor_voltage(DistanceSensorId)`

Donde DistanceSensorId se refiere a un sensor en específico.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cualquier objeto que sirva de obstáculo.

## PROCEDIMIENTO EXPERIMENTAL

1. Escriba un código de exploración para que Robotino®, ante un obstáculo de un ambiente cualquiera, gire para evitarlo en MatLab®.

## OBSERVACIONES

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

- Indique las utilidades que se pueden dar a un robot Explorador.

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°10

## TÍTULO

---

Detector analógico inductivo: programación de movimiento guiado en MatLab® 7.

## OBJETIVOS

---

- ✓ Conocer la programación y utilización de los sensores de inducción de Robotino® en MatLab® 7.
- ✓ Configurar y usar las cajas de herramientas “AnalogInput”.
- ✓ Elaborar un programa de exploración donde Robotino® se acerque a los obstáculos y los evite rodeándolos.

## MARCO TEÓRICO

---

### **Cajas de herramientas “AnalogInput”.**

Al igual que en Robotino®View la entrada análoga se refiere al sensor de inducción, por este motivo al referirse a este detector se lo hará con las cajas de herramientas “AnalogInput”

**AnalogInput\_construct:** construye el objeto “AnalogInput”, parecido al bloque de funciones visto en Robotino®View.

**AnalogInput\_setComId:** asocia el objeto “AnalogInput” a la interface de comunicación. Además lo liga a un Robotino® específico.

**AnalogInput\_value:** es una función que retorna el valor de un sensor de inducción específico.

Ejemplo: `[ value ] = AnalogInput_value (AnalogInputId)`

Donde AnalogInputId se refiere a un sensor en específico.

## EQUIPOS Y MATERIALES UTILIZADOS

- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cinta de aluminio (≥5cm)

## PROCEDIMIENTO EXPERIMENTAL

1. Escriba un código para que Robotino®, recorra un trayecto marcado con una cinta metálica usando el sensor de inducción en MatLab®.

## OBSERVACIONES

Anote las observaciones hechas

## ANÁLISIS DE RESULTADOS Y CONCLUSIONES

---

Analice y concluya, sobre sus resultados.

## CUESTIONARIO

---

- Indique las utilidades que se pueden dar a los sensores inductivos.
- Anote las ventajas y desventajas entre la programación de movimiento guiado de Robotino®View y la de MatLab®7

FIMCP  
LABORATORIO DE MECATRÓNICA  
GUÍA DE PRÁCTICA N°11

**TÍTULO**

---

Sensores de reflexión directa: programación de movimiento guiado en MatLab® 7 mediante bloques en Simulink.

**OBJETIVOS**

---

- ✓ Conocer la programación y utilización de bloques en Simulink para los sensores de distancia de Robotino® en MatLab® 7.
- ✓ Configurar y usar los bloques de Simulink “DigitalInput0”.
- ✓ Elaborar un programa de exploración donde Robotino® siga el trayecto de una línea negra previamente marcada.

**MARCO TEÓRICO**

---

**Simulink.**

Simulink® es un entorno para la simulación multidominio y el diseño basado en modelos para sistemas dinámicos y embebidos. Presenta un entorno gráfico interactivo y un conjunto personalizable de bibliotecas de bloques que permiten diseñar, simular, implementar y probar una serie de sistemas variables con el tiempo, como comunicaciones, controles, procesamiento de señales, procesamiento de vídeo y procesamiento de imágenes..

También, vendría a ser una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Se hace hincapié en el análisis de sucesos, a través de la concepción de sistemas (cajas negras que realizan alguna operación).

Se emplea arduamente en Ingeniería Electrónica en temas relacionados con el procesamiento digital de señales (DSP), involucrando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros. También es muy utilizado en Ingeniería de Control y Robótica.

#### EQUIPOS Y MATERIALES UTILIZADOS

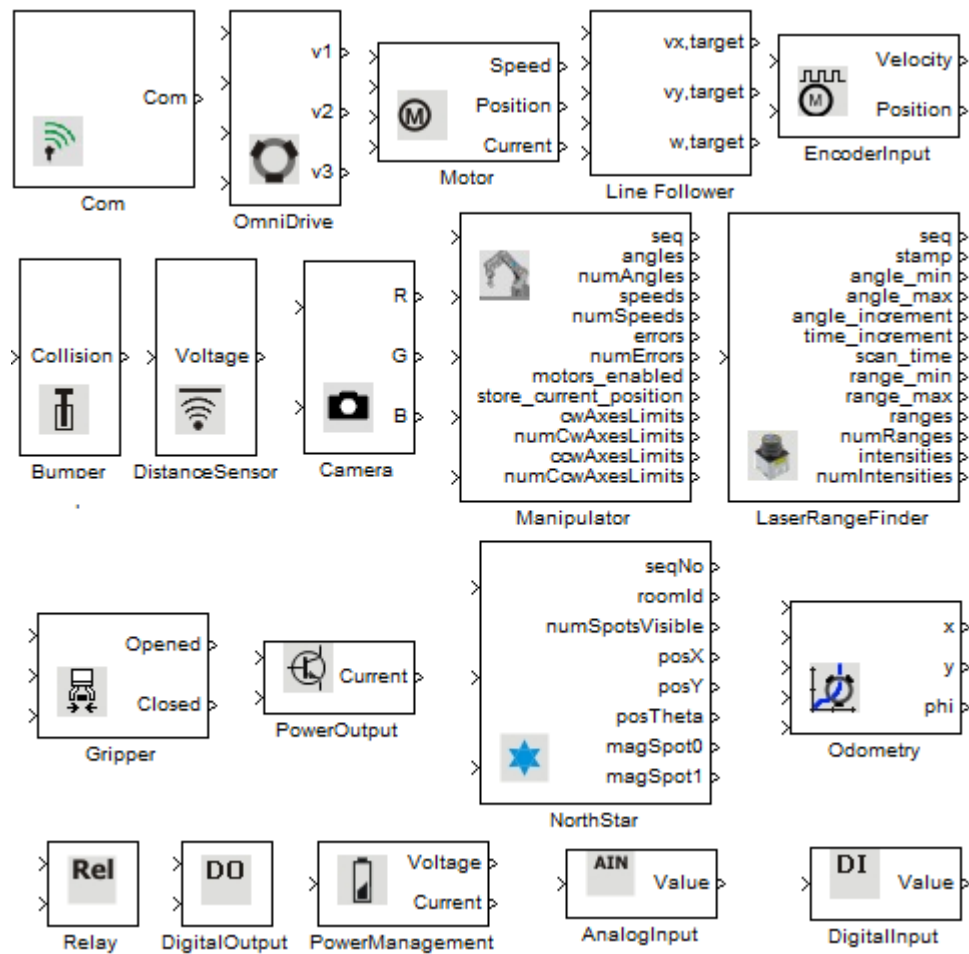
- ✓ S.D.Ro.M. Robotino®.
- ✓ Computadora con capacidad de conexión inalámbrica WiFi y software de MatLab®7 instalado.
- ✓ Cinta adhesiva negra.

#### PROCEDIMIENTO EXPERIMENTAL

1. Escriba un código para que Robotino®, recorra un trayecto marcado con una cinta usando el sensor óptico en Simulink®.

**Solución:** Para realizar el primer paso del procedimiento, primero se debe asegurar que MatLab® se está ejecutando con derechos de administrador (Windows 7) y se está trabajando con el directorio de instalación “Robotino-MatLab” que se instaló anteriormente y que normalmente se ubica en la ruta: C:\Archivos de programa\Festo\RobotinoMatlab.

El siguiente programa demuestra la capacidad de programar mediante el entorno Simulink®, gracias a los “blocksets” que se colocaron junto con la instalación de los controladores de Robotino® para MatLab®7.

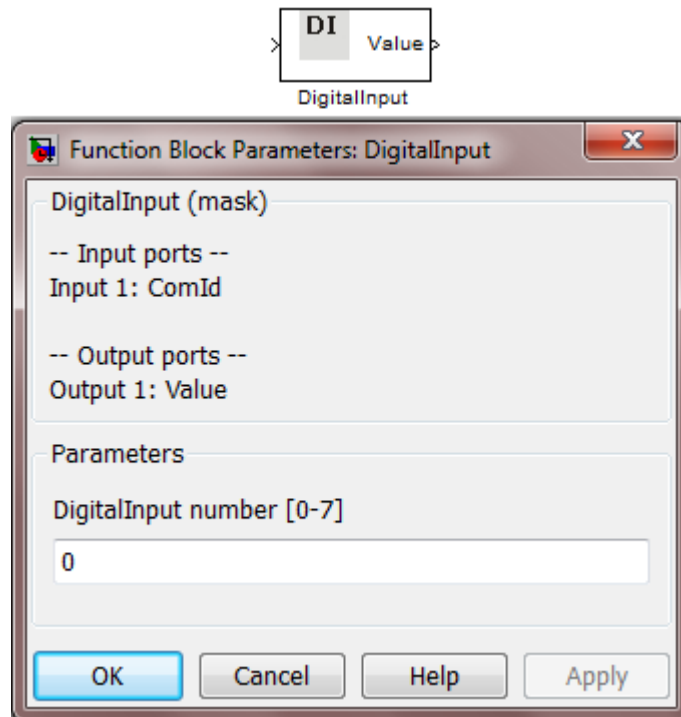


**ILUSTRACIÓN 8: LIBRERÍA DE BLOQUES PARA SIMULINK®;  
PRÁCTICA 11**

En Simulink® cada uno de estos bloques se les puede agregar una “Sub-máscara” para ingresar los “valores iniciales” al bloque. Para acceder a esta función simplemente se da “doble click” sobre el bloque y se abrirá una ventana con la información del bloque los datos que se pueden ingresar en él



Así es como, si damos doble click sobre el bloque "DigitalInput" se abrirá la siguiente ventana presentada en la ilustración 9:

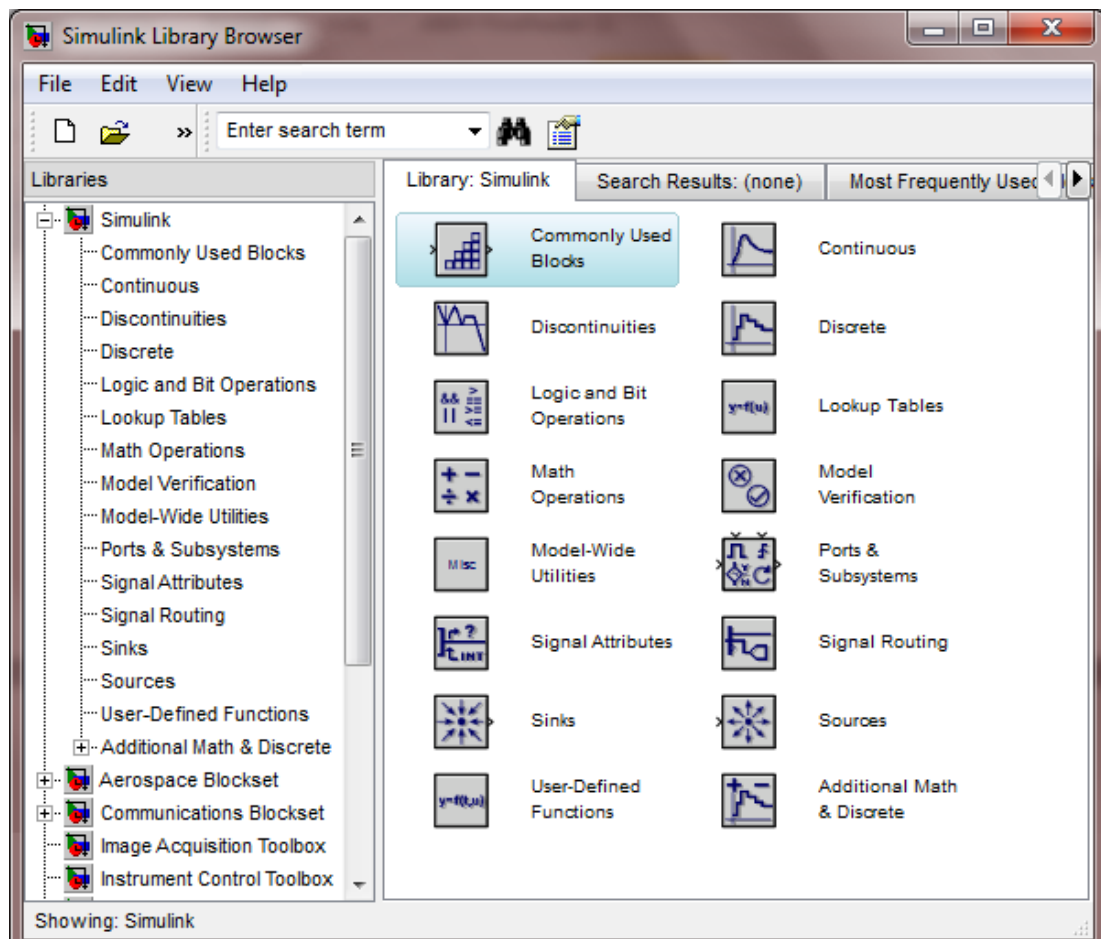


**ILUSTRACIÓN 9: SUB-MÁSCARA DEL BLOQUE "DIGITALLINPUT"**

En la ventana anterior se muestra la información de entradas, salidas y los parámetros que se pueden modificar, que para este caso es la identificación de la entrada digital a la que está conectado el sensor óptico. Además cada bloque se puede unir con líneas

Teniendo claro estas generalidades se comenzará con la programación:

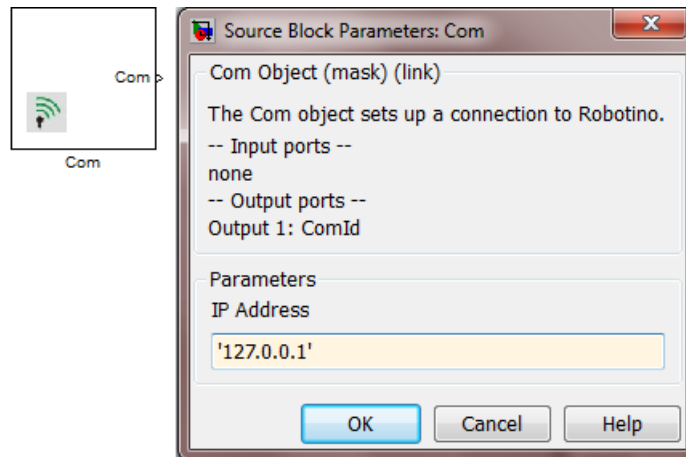
Primero, se escribe “Simulink” en la ventana de comandos de matlab® y se abrirá la ventana de la Ilustración 10 donde se muestran las librerías para programar en Simulink®:



**ILUSTRACIÓN 10: LIBRERÍAS DE SIMULINK; PASO 1 PRÁCTICA 11**

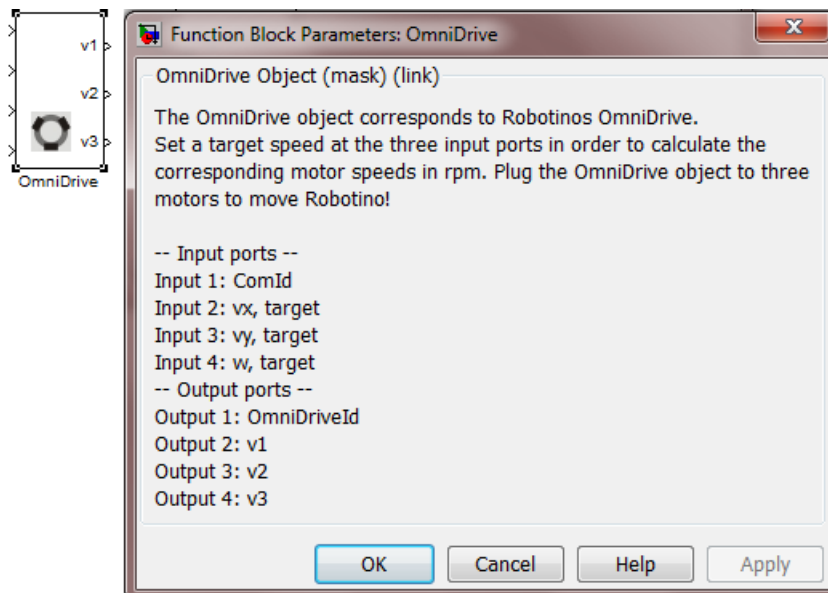
Segundo, para abrir un nuevo proyecto se dará “click” en el icono “□”, se abrirá una ventana en blanco y se añadirán los bloques necesarios para este programa:

**Bloque “Com”**: configura la conexión hacia Robotino® (Ilustración 11)



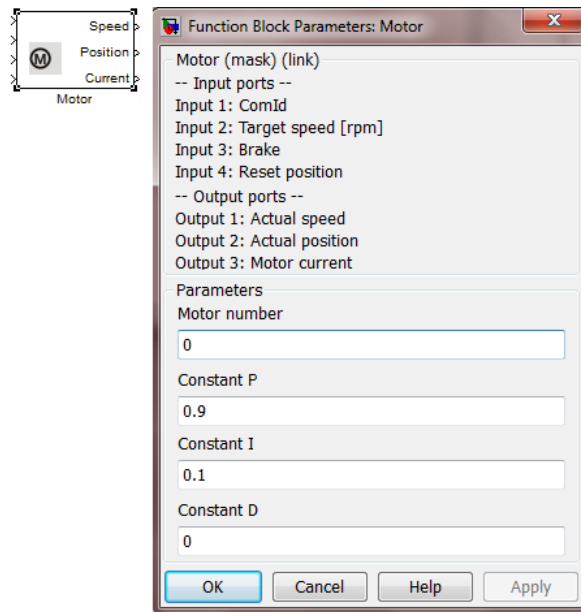
**ILUSTRACIÓN 11: BLOQUE "COM" DE SIMULINK®**

**Bloque “OmniDrive”**: este bloque administra los tres motores de la unidad dependiendo de las velocidades que se le ingresen (Ilustración 12)



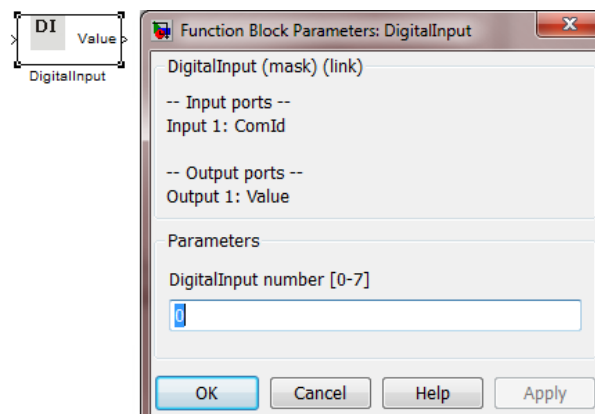
**ILUSTRACIÓN 12: BLOQUE "OMNIDRIVE" DE SIMULINK®; PASO 1 PRÁCTICA 11**

**Bloque “Motor”:** administra directamente unos de los motores de Robotino® (Ilustración 13)



**ILUSTRACIÓN 13: BLOQUE "MOTOR" DE SIMULINK®; PASO 1 PRÁCTICA 11**

**Bloque “DigitalInput”:** administra directamente una de las entradas digitales de Robotino® donde estarán conectados los sensores ópticos (Ilustración 14)



**ILUSTRACIÓN 14: BLOQUE "DIGITALINPUT" DE SIMULINK®; PASO 1 PRÁCTICA 11**

Finalmente elabore el programa antes expuesto:

### OBSERVACIONES

Anote las observaciones hechas

### ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Analice y concluya, sobre sus resultados.

### CUESTIONARIO

- Indique las ventajas y desventajas entre la programación en pseudocódigo y la programación en un entorno visual.

# BIBLIOGRAFÍA

- (1) RUIZ ROJAS, PAOLA ANDREA. “Mecatrónica, Revolución para el siglo XXI”

[http://www.metalactual.com/revista/8/tecnologia\\_mecatronica.pdf](http://www.metalactual.com/revista/8/tecnologia_mecatronica.pdf)

- (2) ARBELÁEZ SALAZAR; MENDOZA VARGAS, OSIEL JAIRO, “La ingeniería Mecatrónica por ciclos en Colombia”, 16 de Julio de 2007.

<http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/122935421-426.pdf>

- (3) C. TEUSCHER, Alan Turing: “Life and Legacy of a Great Thinker”, Springer-Verlag, 2004

(4) ALFONSO ROMERO BARCOJO, "Control y robótica"

Sección: Sensores, IES Departamento de Tecnología,

Unidad Didáctica:

<http://cmapspublic2.ihmc.us/rid=1H2F1807L-JP0SG2-J1J/encoder.pdf>

(5) <http://wiki.openrobotino.org/>

(6) GARCIA MARTÍN ÁNGEL, "Introducción a la programación

con MatLab", Departamento de Física y Matemática

Aplicada

<http://fisica.unav.es/~angel/matlab/matlab1.html>

(7) (FRSF-UTN) DPTO. INGENIERÍA MECÁNICA, "Ajuste

empírico de controladores PID: método de Ziegler-Nichols"

(.pptx), Área: electrónica y sistemas de control -;

(8) O. DUDA RICHARD, E. HART PETER, G. STORK DAVID,

"Pattern classification" (2ª edición), Wiley New York 2001.

(9) DIETRICH PAULUS, JOACHIM HORNEGGER, "Applied

Pattern Recognition" (2ª edición), Vieweg, 1998.