

**Compatible con Arduino**

# **POP-BOT**

**Kit de Robótica**

**Manual de Actividades**

# Créditos

---

Módulo POP-168, la tarjeta RBX-168 que controla el robot, son marcas comerciales de **Innovative Experiment Co. Ltd.**

POP-BOT, POP-BOT logotipo, INEX, y el logo son marcas registradas de INEX **Innovative Experiment Co. Ltd.**

AVR, Atmel, Atmel logotipo, AVR Studio son marcas comerciales registradas de **Atmel Corporation.**

WinAVR es una marca registrada de **SourceForge. Inc.**

AVR-GCC es el copyright del Software libre **Foundation. Inc.**

Arduino es un proyecto de código abierto con mucho soporte. El equipo Arduino se compone de Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, y Mellis David. Nicolás Zambetti ha contribuido desde el principio. Yaniv Steiner y Giorgio Olivero han estado apoyando el proyecto y están trabajando en el uso con la plataforma de sopa instantánea. La plataforma Arduino utiliza para incluir la cadena de herramientas avr-gcc, uisp, y el Procyon AVR-LIB de Pascal Stang. La sintaxis del lenguaje Arduino se basa en cableado por Hernando Barragán. El entorno de Arduino está basada en Procesamiento por Ben Fry y Casey Reas. Gracias a todas las personas que están apoyando a Arduino.

FTDI es una marca comercial de los futuros dispositivos de **Tecnólogi International Ltd.**

I2C es una marca registrada de **Philips Semiconductors.**

Microsoft, Windows son marcas comerciales registradas de **Microsoft Corporation.**

Windows 2K, Windows XP y Windows Vista son marcas comerciales registradas de **Microsoft Corporation.**

Todos los demás nombres de productos y servicios aquí mencionados son marcas registradas de sus respectivos dueños.

# Índice

---

1: POP-BOT Contenido	4
2: Creación de POP-BOT robot Móvil	17
3: Introducción a Arduino ID	23
4: Pop-Bot desarrollo del programa Arduino	29
5: POP-BOT Activación de movimientos	39
6: POP-BOT Pantalla serial LCD	50
7: POP-BOT línea de rastreo	59
8: POP-BOT detección de bordes	88
9: POP-BOT evitar contacto con objetos	93
10: POP-BOT actividad del servomotor	105
11: POP-BOT capacidad de búsqueda de objetos	115

# 1: POP-BOT Contenido

---

## 1.1 La lista de partes del kit de POP-BOT robot móvil

1. POP-168 El módulo Arduino Mini micro controlador compatibles
2. RBX-168 de control del robot tablero con 4-AA soporte de la batería
3. Cambiar módulo con cable JST (2 juegos)
4. Reflector con placa de infrarrojos por cable JST (2 juegos)
5. GP2D120 sensor de distancia por infrarrojos con cable JST
6. Serial LCD de 16 caracteres por 2 módulos líneas con retroiluminación LED y el cable
7. 48:1 ratio de 4,5 V DC motor con caja de cambios de cable de IDC (2 juegos)
8. Servo motor estándar (voltaje de funcionamiento es de 4,8 a 7.2Vdc)
9. Juego de rueda circular, rueda de con goma estriada y tornillos con rosca de 2mm. (2 juegos)
10. 80x60 cm. y 80x80 cm. Rejilla de plástico juego de placas (2 juegos)
11. Círculo de base con ruedas de bolas de inactividad
12. Carpintería de plástico y piezas de ensamble (60 piezas de carpintería, 3 diferentes tipos de colores de plástico, 4 piezas de cada uno de los agujeros 3/5/12)
13. Eje de metal en ángulo recto (4 piezas de cada uno de 1x2, 2x2, 2x5)
14. Tuercas y tornillos de fijación
15. Plano para demo de seguimiento
16. UCON-4 Cable convertidor de serie a USB para descarga y la comunicación
17. CD-ROM con software, código fuente y documentación

## 1.2 Información componentes Micro controladores

### 1.2.1 POP-168 microcontrolador módulo

El Arduino POP-168 es una tarjeta flexible, sin componentes ocultos que permite el pleno desarrollo de sus funciones con herramientas AVR estándar como el IAR como C / C ++, MikroElektronika Mikro BASIC / MikroPascal para AVR, y también la herramienta de código abierto WinAVR: AVRGCC para Windows, etc.

**Arduino POP-168** utiliza el microcontrolador AVR ATMega168 de Atmel ([www.atmel.com](http://www.atmel.com)). **Arduino POP-168** es un módulo similar al **BASIC Stamp** de **Parallax** ([www.parallax.com](http://www.parallax.com)). Incluye el RS-232 circuito de comunicación puerto serial para la descarga y la comunicación de datos con la computadora. Arduino POP-168, hardware compatible con el Proyecto Arduino ([www.arduino.cc/es](http://www.arduino.cc/es)).

El diagrama esquemático completo de Arduino POP-168 módulo se muestra en la figura 1-1. El resumen de las características de POP-168 módulo es como sigue:

- ATMega168 a bordo con ADC de 10 bits del convertidor, 16KB de memoria Flash, EEPROM de 512 bytes, 1 KB de RAM, reloj de 16MHz
- Capacidad de la interfaz RS-232 para la Comunicación
- Código de inmediato la subida con el gestor de arranque integrado
- Botón de restablecimiento de la capacidad de restablecimiento de
- Factor de forma pequeño para el desarrollo de tamaño compacto
- ISP de puerto para la programación con el dispositivo PX-400/PX-4000
- SMD LED para las indicaciones
- Totalmente compatible con el Proyecto Arduinio
- 16 I / O ASIGNACIÓN pin compatible con el módulo i-Stamp/i-Stamp2P24
- Rango de Voltaje de alimentación 3.3 a 5 V 50mA

### 1.2.2 RBX-168 Robot tarjeta controladora para Arduino POP-168

RBX-168 es la tarjeta controladora, es una completa plataforma de bajo costo de desarrollo, diseñada para aquellos interesados en el aprendizaje y el uso de Arduino POP-168, en el módulo de aplicaciones de robótica. Su tamaño compacto, características convenientes y precios más bajos la convierten en una herramienta ideal para el estudiante y el educador. La Figura 1-2 muestra la disposición de los RBX-168 bordo y esquema completo de la que se muestra en la figura 1-3. La característica resumida técnica de la RBX-168 placa es como sigue:

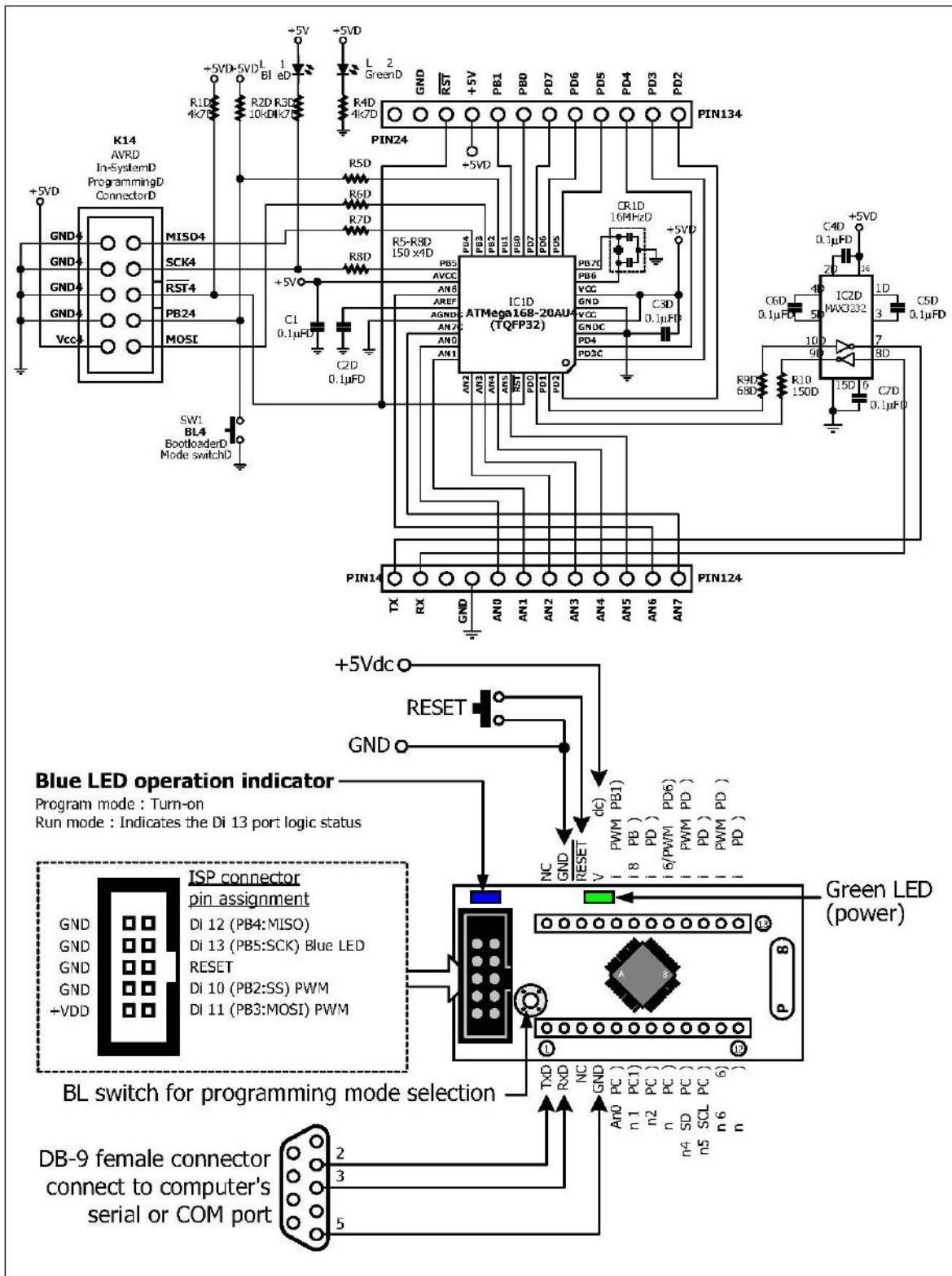


Figure 1-1 Schematic diagram, Pin assignment and simple connection of POP-168 microcontroller module

- Bloque para las conexiones de la batería. Es compatible con +4,8 a +12 VDC y tiene un interruptor de energía a bordo.
- Regulador de voltaje de +5 VDC de alimentación. Regula la tensión de alimentación para el módulo del POP-168 y todos los puertos de los sensores.
- 2 interruptores se conectan con el puerto digital 2 (ED2) y 4 (DI4). También se conectan con LED para el indicador de la operación.
- 5 Puertos universales para la función de soporte de entrada analógica y entrada / salida digital; An1 (DI15) a AN5 (Di19)
- 2 puertos de entrada analógica; AN6 y AN7
- Puerto I2C bus; AN4 (SDA) y AN5 (SCL)
- RS-232 interfaz de puerto serial.
- 2ch. Motor de corriente continua con indicadores. Voltaje del motor de 2,5 a 13,5Vdc.
- 2 Salidas para servo motores; conectar con el puerto digital 7 (ED7) y 8 (DI8).
- Las conexiones de los altavoces piezoelectricos (no se muestran en la figura 1.2, que se fija en el tablero de circuitos inferior RBX-1689, placa que se conecta mediante el protocolo POP-168 An0/Di14 pin.

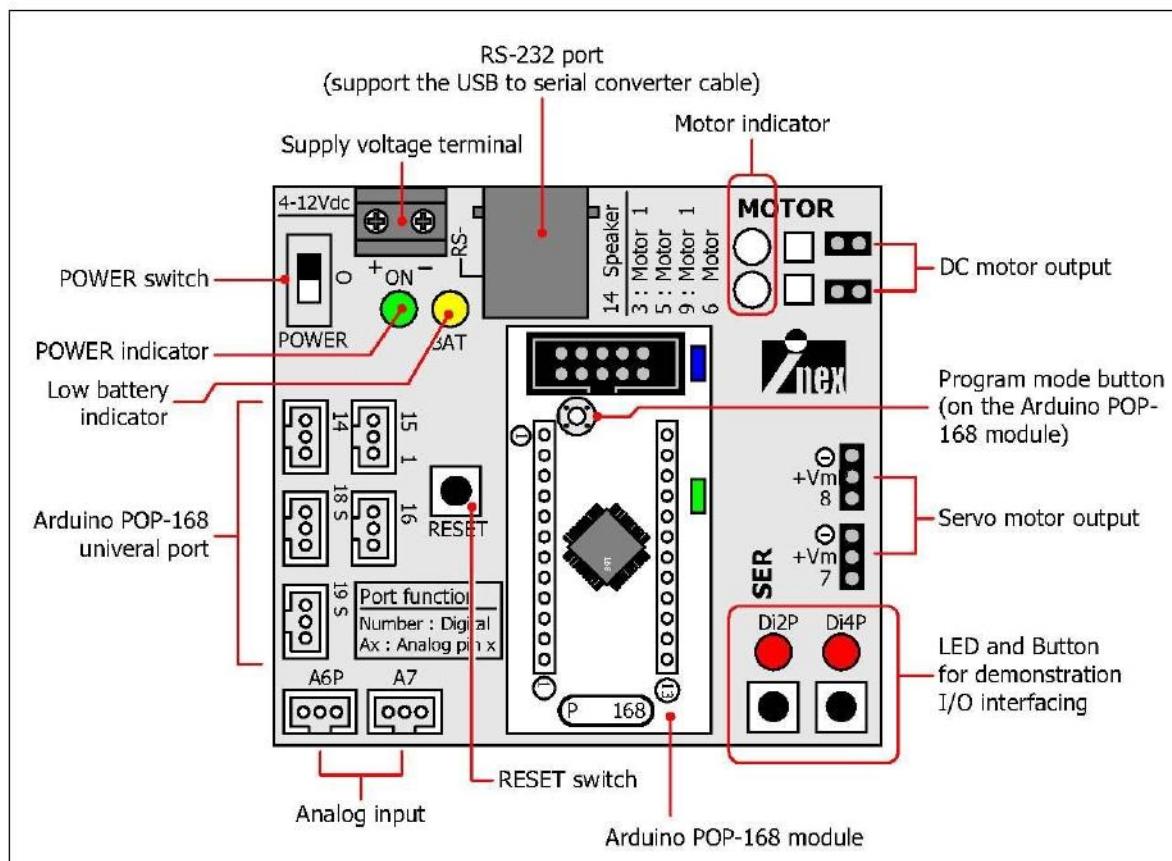


Figura 1-2 Diseño del controlador, placa X-168

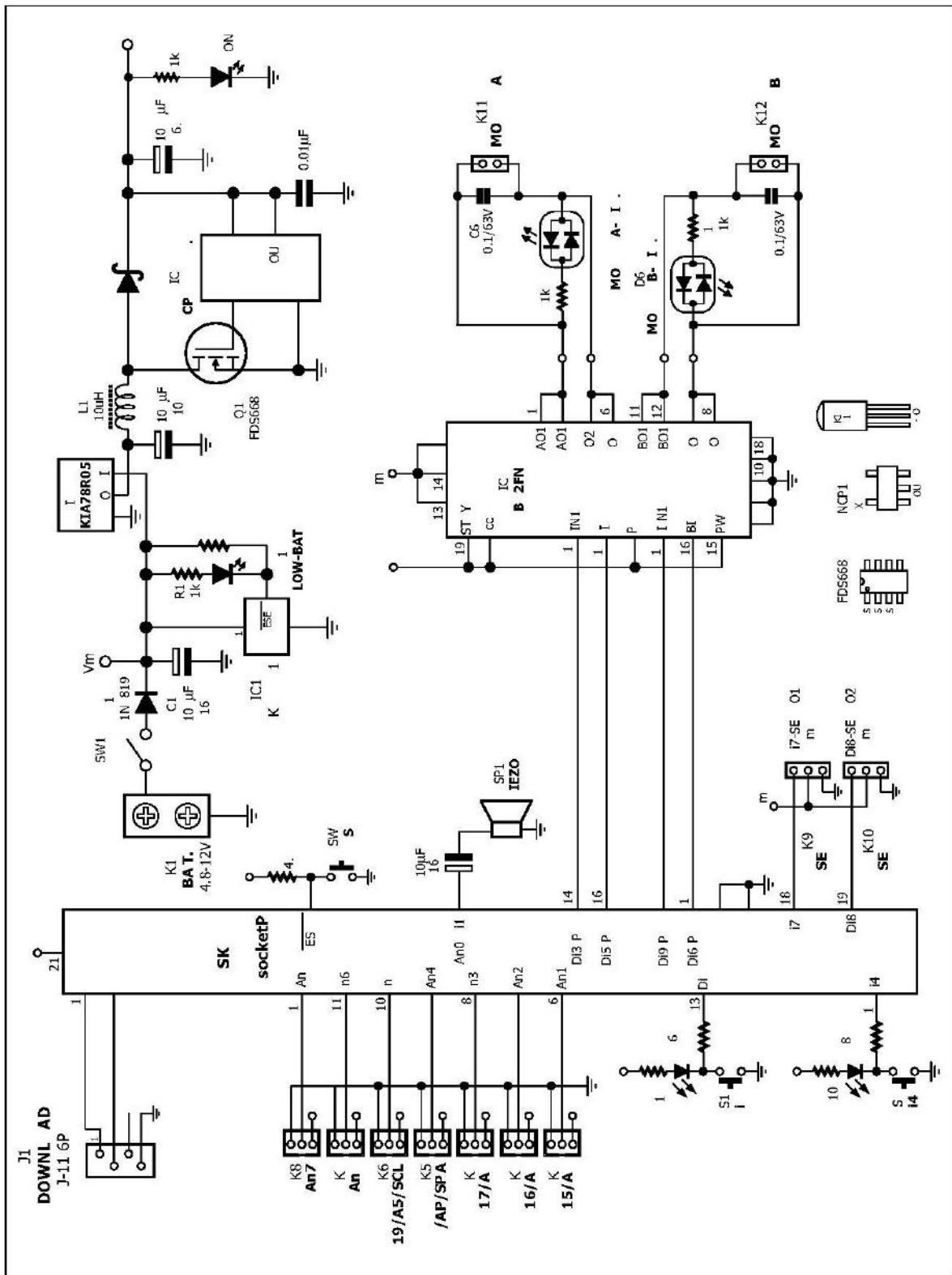
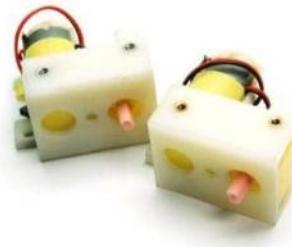


Figura 1-3 Esquema completo del diagrama de la placa RBX-168

## **1.3 Características del dispositivo de salida**

### **1.3.1 Caja de Velocidades del motor de corriente continua**

Este kit de robot ofrece una relación de 48:1 con la caja de velocidades del motor, modelo BO-2 con cable conector IDC. Sus características son:



- Voltaje de trabajo de +3 a +9Vdc
- Consumo de corriente 130mA (+6Vdc y sin carga)
- La velocidad media es de 170 a 250 RPM
- El peso es de 30 gramos
- Torque mínimo, es de 0,5 kg.cm.
- El montaje se realiza con 5 tuercas de inserción
- Dimensión: 42,0 x 45,0 x 22,7mm.

### **1.3.2 Servomotor estándar**

El servo estándar es ideal para la robótica y proyectos básicos de movimiento. Estos servos permiten un rango de movimiento de 0 a 180 grados. El eje de salida del engranaje del servo es un estándar de configuración de Futaba. Especificaciones técnicas son las siguientes:



- Tensión de trabajo máxima es 6Vdc.
- Velocidad de 0 ° a 180 ° en 1,5 segundos en promedio.
- Peso 45,0 grams
- Torque 3,40 kg-cm/47oz-in
- Tamaño 40.5x20.0x38.0mm

### 1.3.3 SLCD16x2: 16 caracteres 2 líneas del módulo LCD

El módulo LCD de 16x2 proporciona una forma sencilla de mostrar los datos del micro-controlador. El módulo sólo requiere un pin I/O, +5 V y tierra para funcionar. Simples datos en serie de comandos se puede utilizar en el módulo Arduino POP-168 para comunicarse con el módulo en 2400 y 9600 baudios.

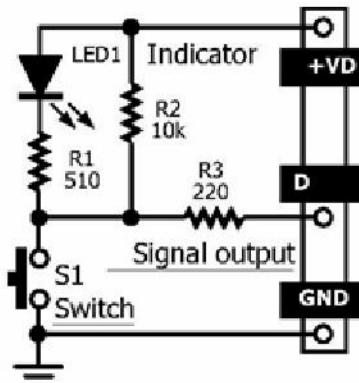
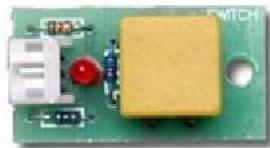


Características del módulo de serie 16x2 LCD:

- Entrada serial TTL con nivel de logica inversión/no inversion
- 1/8 o 1/16 de servicio pueden ser seleccionados por un puente.
- Scott Edwardsís LCD serial Backpack, además compatible con los comandos extendido que hacen más fácil el control de LCD.
- Funcionamiento con +5Vdc
- SLCD16x2 proporciona un ajuste de brillo con resistencia variable en la posición BRILLO.
- Interfaz de conector tiene 3 pines: +5V de alimentación, la entrada de datos en serie (S) y tierra (G).

## 1.4 Características del sensor del módulo

### 1.4.1 Módulo comutador / Sensor de Tacto

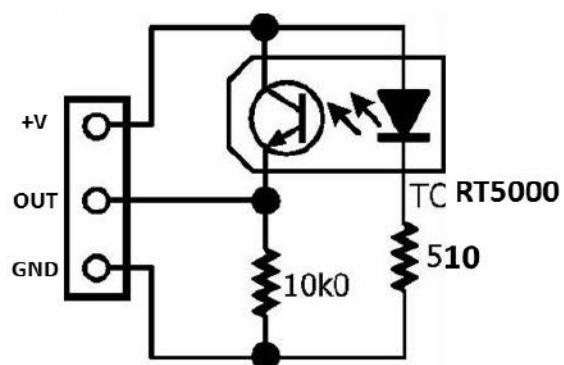
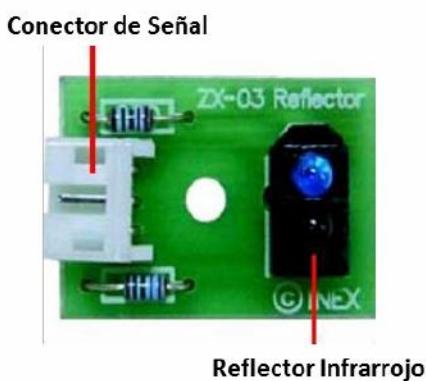


La entrada del interruptor se utiliza para detectar colisión en  $\text{I}^{\circ}\text{O}$  lógica. Dos conjuntos junto con el cable de conexión se proporcionan.

### 1.4.2 ZX-03: sensor de reflector infrarrojo

El corazón de este sensor es el TCRT5000, sensor de objetos reflectantes. Está diseñado para proximidad y detección por medio de infrarrojos (IR). Tiene un diodo infrarrojo detrás de la ventana transparente azul y un transistor de infrarrojos detrás de su ventana en negro. Cuando la infrarroja emitida por el diodo se refleja en una superficie y vuelve a la ventana en negro, que golpea la base del transistor infrarrojo, haciendo que pase corriente. Cuando se utiliza como un sensor analógico, el ZX-03 puede detectar tonos de gris en el papel y las distancias en un intervalo corto si la luz en la habitación se mantiene constante.

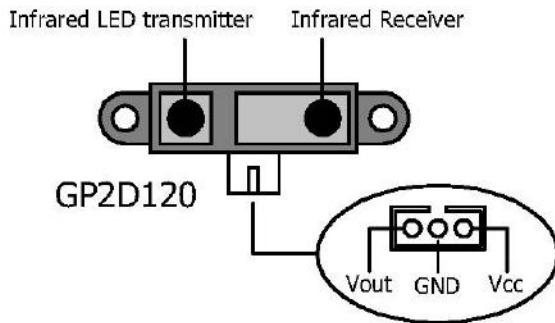
La distancia adecuada desde el sensor a la línea o en el suelo es de 3 a 8mm.



El Voltaje de salida es de entre 0,1 a 4,8 V y el valor digital de ellos es de 10 bits; convertidor A / D es de 20 a 1.000. Por lo tanto, ZX-03 será adecuado para su aplicación a la línea del sensor de seguimiento.

#### 1.4.3 GP2D120 sensor de distancia por infrarrojos

Uno de los sensores especiales en la robótica es el GP2D120. Es un sensor de distancia por infrarrojos. Algunas personas lo llaman el Ranger IR. Con el módulo de GP2D120, agrega la medición de distancia y detección de obstáculos utilizando la función de luz infrarroja para su robot. El robot MicroCamp puede evitar los obstáculos sin tener que hacer ningún contacto físico.



Características del módulo GP2D120

- Utiliza la reflexión de luz infrarroja para medir el alcance
- Se puede medir un rango de 4 a 30 cm.
- 4,5 a 5 V de alimentación y 33mA corriente eléctrica
- El rango de tensión de salida es de 0,4 a 2,4 V como fuente de +5 V

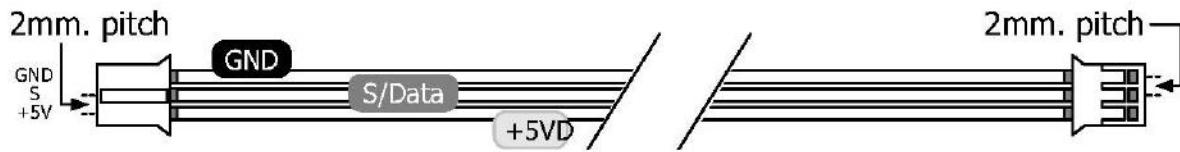
El módulo de infrarrojos GP2D120 de seguimiento tiene 3 terminales: entrada de alimentación (Vcc), Tierra (GND) y salida de tensión (salida). Para leer los valores de tensión de la GP2D120, debe esperar hasta después del período de reconocimiento que es de alrededor de 32 a 52,9 ms. La tensión de salida del GP2D120 en un rango de 30 cm y 5 de suministro de potencia V es entre 0,25 a 0,55V, con los medios de 0,4V siendo. En el rango de 4 cm., La tensión de salida va a cambiar en 2.25V a  $\pm 0.3V$ .

## 1.5 POP-BOT Cable de información

El kit de POP-BOT robot móvil incluye cables de señal para la interconexión entre la tarjeta del controlador, el módulo de sensor y la computadora. Entre ellos se incluyen los JST3AA-8 cable para la interconexión con el módulo de sensor, UCON-4 es el USB a RS-232 cable conversor para la interfaz con la computadora.

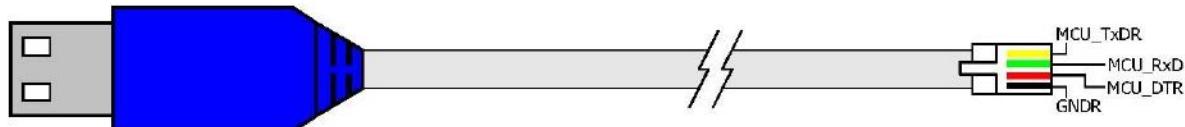
### 1.5.1 JST3AA-8 cable

Este es un cable de INEX estándar, 3-hilos combinados, de 2mm. Conector JST a cada extremo. 20cm de largo. Se utiliza para conectar entre el microcontrolador y todos los módulos de sensores en el kit del robot POP-BOT. La asignación de alambre se muestra en el diagrama siguiente.



### 1.5.2 UCON-4 USB cable convertidor de puerto serial

Este se utiliza para conectar el puerto USB del computador y el RBX-168 panel de control. El extremo del cable utiliza un enchufe modular RJ-11 6P4C (6-pins forma y 4 contactos) Su longitud es de 1,5 metros. La asignación de cable se muestra en el diagrama siguiente.



Este cable requiere +5V desde el puerto USB (1,0 o 2,0). El usuario puede configurar la velocidad de transmisión de hasta 115.200 bits por segundo. Requiere la instalación del driver antes de usar.

## **1.6 Características mecánicas**

### **1.6.1 Círculo de ruedas y neumáticos conjunto**

Incluye 2 de ruedas de rodadura de caucho. Fijar la rueda con un eje a la caja de velocidades de 2 mm y atornillar.



### **1.6.2 Juego de rejillas de plástico**

Incluye 2 rejilla de 80x60mm y de 80x80mm. Cada placa proporciona 3mm de diámetro con 5 mm terreno de juego.



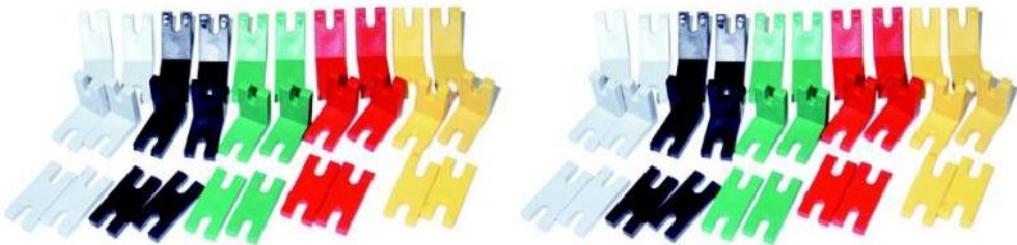
### **1.6.3 Base Circular**

Esta base es de plástico ABS. Cuenta con 2 ruedas de bolas en ambos lados. Cuenta con agujeros para la fijación de la tarjeta controladora, sensores y otros componentes mecánicos.



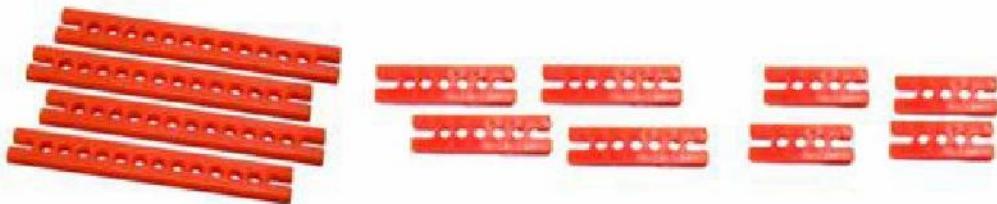
#### **1.6.4 Carpinteros de plástico**

60 piezas de colores variados carpinteros de PVC de plástico. Ellos pueden ser conectados entre sí o mediante el uso de tornillos y tuercas de 3mm en la instalación. Hay 4 tipos: de ángulo recto, obtuso, carpintería recta y recta del agujero carpintero.



#### **1.6.5 Franja de carpinteros**

Están hechas de plástico. Cada carpintero se puede unir para la expansión de longitud. Son 4 unidades de 3 tamaños, de 3, 5 y 12 agujeros. Total de 12 piezas.



#### **1.6.6 cuadro titular**

Caja de plástico inyectado ABS para apoyar el panel de control RBX-168. Tiene agujeros para la fijación con cualquier plataforma.



### **1.6.7 Ángulos rectos de Metal**

Son de 7,5mm de ancho, de ángulo recto, de metal. Cada eje tiene 3mm de agujero para la inserción del tornillo para fijar con otras estructuras. El juego incluye 4 piezas de 1x2, 2x2 y 2x5 agujeros.



### **1.6.8 Tornillos y tuercas del conjunto**

Incluye 2 tornillos roscalata de 2mm; 4 de 3x8mm, 30 Tornillos M3 de 3x10mm; 4 tornillos M3 de 3x15 mm; 4 Tornillos M3 de 3x40mm; 10 Tornillos M3 de 3x8mm; 2 tornillos de cabeza plana y 30 Tuercas M3 de 3mm.



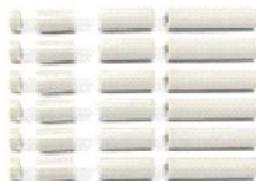
### **1.6.9 Espaciadores de metal**

Son piezas metálicas para apoyar la placa y la placa de los sensores. Ellos están hechos de metal chapado nikle. Incluye 6 de 33mm. Cada enfrentamiento tiene 3mm e hilo a través de los hoyos.



### **1.6.10 Separador de plástico**

Son apoyos de algunas partes mecánicas a la placa y la placa del sensor. Este kit incluye 4 piezas de separadores de plástico (3mm., 10mm., 15mm. Y 25mm.)

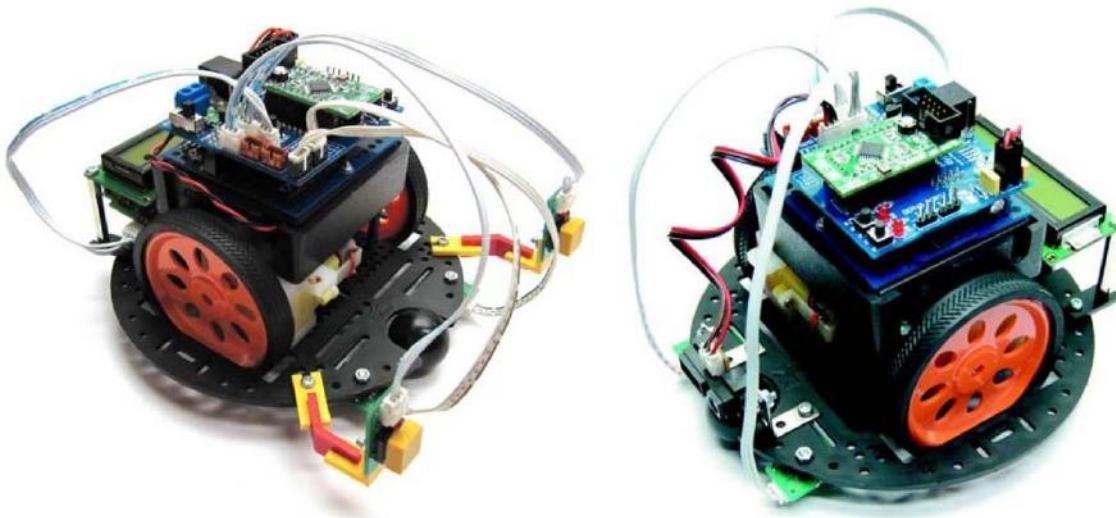




## 2: Cr eacion d  POP-BOT robot movil

---

### 2.1 POP-BOT caracter sticas



- Controlados por el m dulo del microcontrolador Arduino POP-168. Basado en ATMega168 microcontrolador.
- Programable a trav s del puerto serie y apoyar el convertidor de USB a puerto serial.
- Apoyar la variedad de sensores, tales como reflectores infrarrojos, el sensor t ctil para evitar objetos, seguidor infrarrojo o un sensor de distancia evitar objetos.
- Apoyarse con el control remoto con cable incluye PlayStation controlador.
- Apoyar la red inal mbrica de datos en serie del m dulo de comunicaci n SUHC como Xbee, XBee Pro y Bluetooth.
- Incluye m dulo de 16x2 LCD de serie para el seguimiento y estado de la pantalla Otorgamiento.
- 2 servo motores controladores de puerto. Apoyar a las peque as RC servo motor 4,8 a 6V.
- Requiere 4 pilas AA.

## 2.2 La lista de partes



Circle base x 1



RBX-168 board with POP-168 x1



Box holder x 1



Circle wheel and tire x 2



DC motor gearbox x 2



33mm. metal spacer x 4



Serial LCD  
module x 1



Infared reflector x 2



GP2D120 distance sensor x 1



Plastic joiners



Right angled metal  
shaft 2x2 x 2



2mm. self-tapping screw x 2



Plastic spacer set



Screw and Nut set

## 2.3 Crear Procedimientos

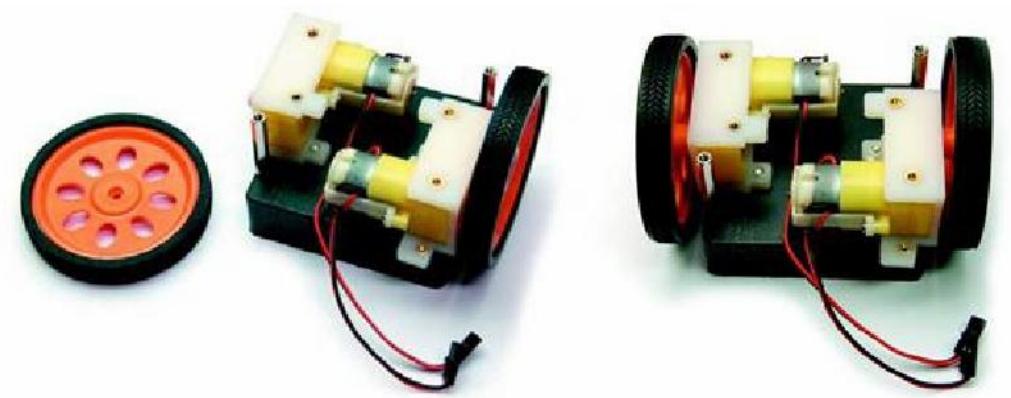
(1) Agregado de dos cajas de velocidad del motor con el titular de la Caja de 3x8mm tornillos planos.



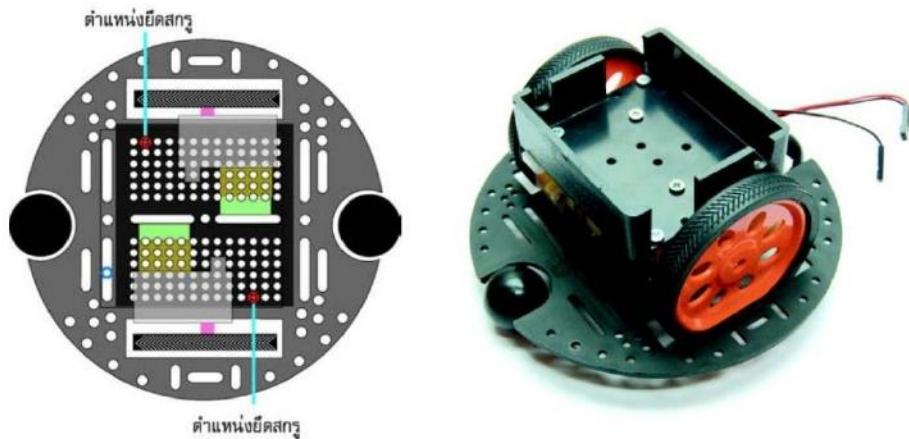
(2) Agregar los 2 separadores de metal de 33mm con el titular de la Caja de 3x8mm, con los tornillos.



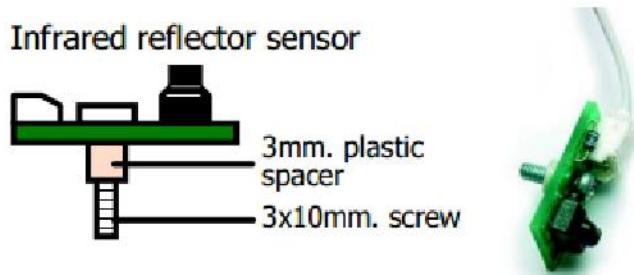
(3) Inserte la rueda con los neumáticos al eje del motor de corriente continua y fijar con tornillos de 2mm de roscalata.



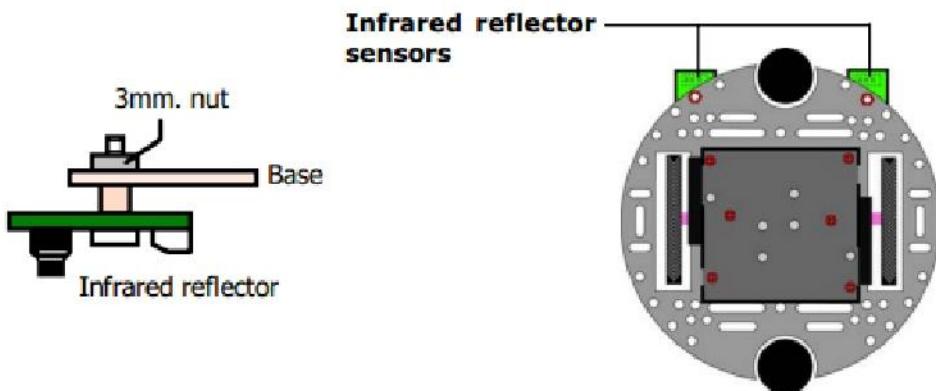
(4) Fijar la estructura de la caja de velocidades, como se muestra en el paso (3) con la base del círculo mediante el uso de tornillos de 3x6mm. Véase la posición de las ruedas en el centro de la base.



(5) Inserte el tornillo de 3x10mm por el agujero del reflector infrarrojo y los separadores de plástico de 3mm. Haz 2 series.



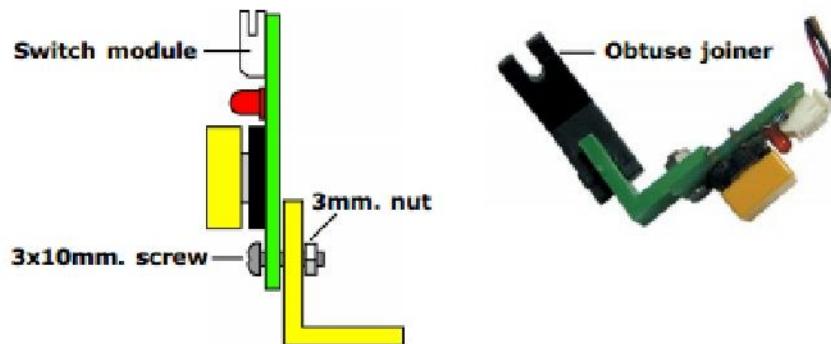
(6) Fijar las estructuras reflectoras de infrarrojos desde el paso (5) en la base del robot en el frente de ambos lados mediante el uso de tuercas de 3mm.



(7) Ahora el chasis POP-BOT con sensores infrarrojos reflector está listo.



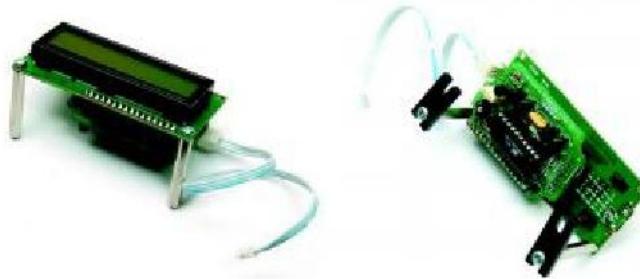
(8) Conecte el derecho carpintería ángulo con el módulo de conmutador mediante el uso de 3x10mm. tornillo y 3 mm. siguiente tuerca de conectar la carpintería obtuso en el extremo derecho de la carpintería ángulo. Haz 2 series.



(9) Fix 2 piezas de carpintería recta en la parte frontal de chasis del robot mediante el uso de 3x6mm. tornillos y 3 mm. frutos secos. A continuación, conecte las estructuras de conmutación de la etapa (9) al final de carpinteros rectas.



(10) Coloque 2 separadores de metal de 33mm con el módulo SLCD16x2, mediante el uso de tornillos de 3x6m. A continuación, fijar la recta carpintero en el extremo del espaciador mediante el tornillo de 3x10mm.

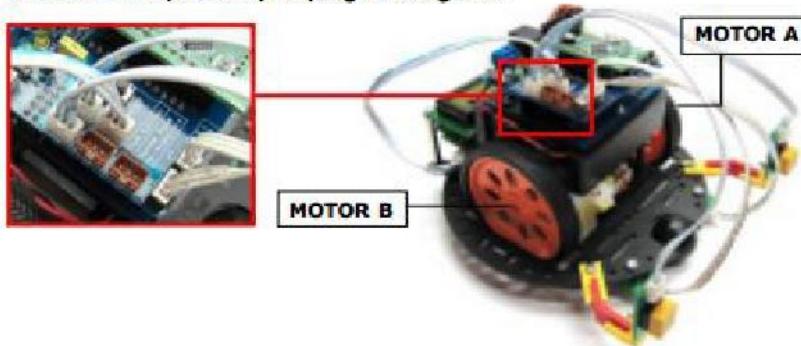


(11) Fijar la estructura SLCD16x2 desde el paso (10) en la parte posterior del chasis del robot después; de la foto de abajo mediante el uso de los tornillos de 3x10mm y tuercas de 3 mm.



(12) Coloque la placa RBX-168 en el soporte de caja. Conecte todos los cables. Comience con el cable del motor izquierdo para operar la salida del cable del motor derecho de la salida MOTOR B. Conecte el cable de la izquierda reflector de infrarrojos al pin A7, el cable adecuado reflector de infrarrojos a la clavija A6. A continuación, conecte el interruptor de la izquierda con el cable del módulo de pin 15/A1, el cable del interruptor derecho a 17/A3 y conectar a la SLCD16x2 pin 16/A2.

*The POP-BOT is just ready for programming now.*



# 3 : Introducción a Arduino IDE

Arduino es una plataforma de código abierto basado en prototipos de electrónica flexible y fácil de usar hardware y software. Destinados a artistas, diseñadores, aficionados y cualquier persona interesada en la creación de objetos o entornos interactivos. \*

En este capítulo se describen sobre la introducción a Arduino. Comience con la instalación, explicará acerca de los componentes del IDE Arduino y los detalles de la barra de menús.

## 3.1 Instalación del software

(1) Inserte el POP-BOT de CD-ROM a la unidad de CD del ordenador.

(2) Entra en la carpeta del software de Arduino. Encuentra el ArduinoSetup.exe clic y doble, La instalación se iniciará.



POP-BOT CD-ROM contiene el software de Arduino V15, todos los códigos de ejemplo para las actividades de POP-BOT y los archivos de la biblioteca son necesarios. Puede obtener la última versión de Arduino en [www.arduino.cc](http://www.arduino.cc). Sin embargo es necesario asegurarse de la ruta correcta de la biblioteca de POP-BOT después de actualizar la nueva versión de Arduino IDE.

\* Introducción es el párrafo de la página web de Arduino ([www.arduino.cc](http://www.arduino.cc))

### 3.2 Arduino medio ambiente

Después de iniciar Arduino IDE, la ventana principal aparecerá como shwon en la figura 3.1. El Arduino incluye el entorno de la siguiente manera.

- **Menu:** Seleccione el comando de la operación.
- **Toolbar:** Incluye todos los botones de comando más.
- **Tabs:** Permite gestionar bocetos con más de un archivo (cada uno de ellos aparece en su propia pestaña).
- **Text editor:** área de Editor de texto para crear el boceto.
- **Message area:** Muestra el estado de la operación del programa, tales como la compilación de resultados.
- **Text área:** El espacio muestra la compilación de información y de la ventana de serie de terminales de datos si habilitar.

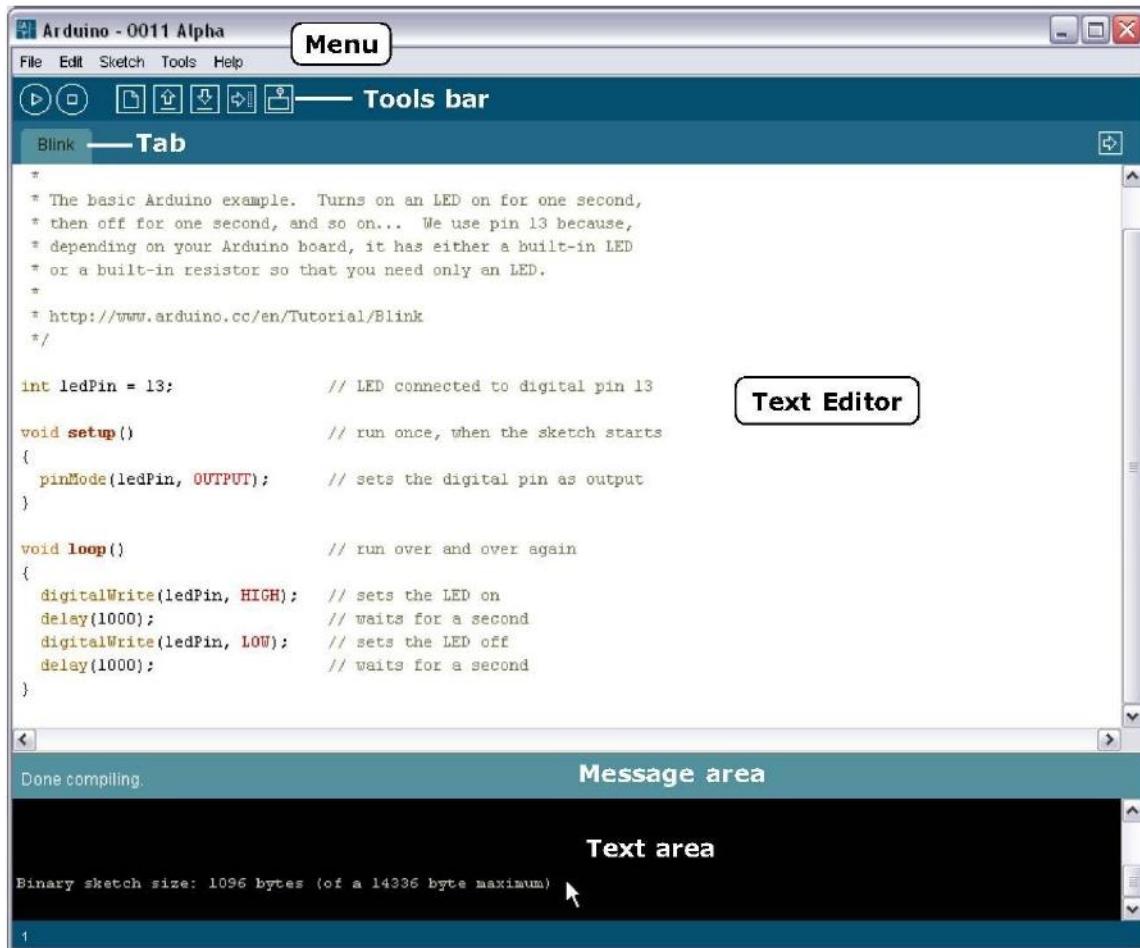


Figure 3-1 Arduino environment

### 3.3 Barra de menús

#### 3.3.1 Archivo

El Arduino llama al código como **Sketch**. Este menú contiene muchos comandos como abrir, guardar y cerrar el esquema de la siguiente manera:

- New:** Crea un nuevo dibujo, el nombre es el formato de fecha actual "sketch\_YMMDDa".
- Sketchbook**
  - Open:** el esbozo existe.
  - Example:** Abra el dibujo existente.
- Save:** Guarda el dibujo actual.
- Save as:** Guarda el dibujo actual como otro nombre.
- Upload to I/O board:** Contenido de su código para el Arduino I / O (POP →168 del módulo). Asegúrese de guardar o verificar su esquema antes de subirlo.
- Preference:** Establecer algún tipo de preferencia del entorno Arduino.
- Quit:** Salir de la IDE de Arduino.

#### 3.3.2 Edición (Edit)

El menú Edición ofrece una serie de comandos para la edición de los archivos de Arduino.

- Undo:** Invierte el último comando o la última entrada escrita.
- Redo:** Invierte la acción del último comando Deshacer. Esta opción sólo está disponible, si ya ha habido una acción de deshacer.
- Cut:** Elimina el texto seleccionado y copia en el portapapeles.
- Copy:** Copia el texto seleccionado al portapapeles.
- Paste:** Inserta el contenido del portapapeles en la ubicación del cursor y remplaza el texto seleccionado.
- Select all:** Selecciona todo el texto en el archivo que está actualmente abierto en el editor de texto.
- Find:** busca una ocurrencia de una cadena de texto en el archivo abierto en el editor de texto y da la opción de sustituirlo por un texto diferente.
- Find Next:** Busca la siguiente ocurrencia de una cadena de texto en el archivo abierto en el editor de texto.

### 3.3.3 Croquis

Este menú ofrece una serie de comandos para compilar el código y gestionar la biblioteca.

- Verify /Compile:** Verificar y compila el código.
- Stop:** Detiene la actividad actual.
- Add File:** abre un navegador de archivos. Seleccione un archivos de código para agregarlo a los bocetos directorio "data".
- Import Library:** Importar la biblioteca de la adición.
- Scow Sketch Folder:** Abre el directorio para el dibujo actual.

### 3.3.4 Herramientas

Este menú proporciona comandos sobre las herramientas para el desarrollo del boceto Arduino y la configuración del hardware de Arduino.

- Auto Format:** Los intentos de formatear el código en un más legible para el diseño. Formato automático que antes se llamaba embellecer.
- Archive Sketch:** Comprimir el boceto actual en el archivo zip.
- Export Folder:** Abra la carpeta que contiene el boceto curretn.
- Board:** Elige el hardware Arduino. Para POP-BOT, elija POP-168 o el Mini Arduino
- Serial Port:** Permite seleccionar el puerto serie a usar por defecto para el código de subir a
- la Arduino I / O o los datos del monitor que viene de él. Los datos procedentes de la Arduino I / O se imprime en formato de caracteres en la región del área de texto de la consola.

### 3.3.5 Ayuda

Este menú contiene muchas informaciones en formato HTML para el apoyo a los usuarios de Arduino.

- . Primeros pasos: Abre el ¿Cómo empezar a Arduino.
- . Solución de problemas: Proponer la solución cuando se resuelva el problema, en el Arduino.
- . Medio Ambiente: Describir acerca de los entornos Arduino
- . Referencia: Se abre la referencia en el navegador web predeterminado. Incluye referencia para el lenguaje, entorno de programación, las bibliotecas, así como una comparación del lenguaje.
- . Preguntas más frecuentes: Vea la pregunta popular y respuesta acerca de Arduino.
- . Visita [www.arduino.cc](http://www.arduino.cc): Abre el navegador web por defecto para la página de Arduino.
- . Acerca de Arduino: Abre un panel de información concisa sobre el software.

### 3.4 La barra de herramientas



**Verificar / Compilar:** Comprueba el código de los errores.



**Stop:** Detiene el monitor de serie, o eliminar el realce de los otros botones.



**Nuevo:** Crea un nuevo dibujo.



**Abierto:** Presenta un menú con todos los bocetos en su cuaderno de dibujo.



**Guardar:** Guarda tu dibujo.



**Subir a la I/O:** Contenido de su código para el Arduino I / O (POP-168 módulo). Asegúrese de guardar o verificar su esquema antes de subirlo.



**Serial Monitor:** Muestra los datos de serie que se envían a la placa Arduino (USB o Junta de serie). Para enviar datos a la tarjeta, introducir texto y haga clic en el botón "enviar" o pulse Intro. Elija la velocidad de transmisión de la lista desplegable que coincida con la tasa pasó a Serial.begin en su dibujo. Tenga en cuenta que en Mac o Linux, la placa Arduino se restablecerá (volver a ejecutar su esquema desde el principio) cuando se conecta con el monitor de serie.

### 3.5 Noticias de referencia del programa Arduino

Este libro de actividades no se describen sobre la programación de Arduino. Usted puede leer y hacer que el conocimiento acerca de la sintaxis y la referencia de programación de Arduino en el menú Ayuda o aprender desde el sitio web de Arduino en [www.arduino.cc](http://www.arduino.cc).

Además, usted puede aprender de las 40 páginas del cuaderno de programación de Arduino. Descargue también rom página web de Arduino en la página de juegos.



## 4: POP-BOT desarrollo del programa Arduino

El desarrollo del programa POP-BOT puede resumir en el siguiente diagrama de la figura 4-1

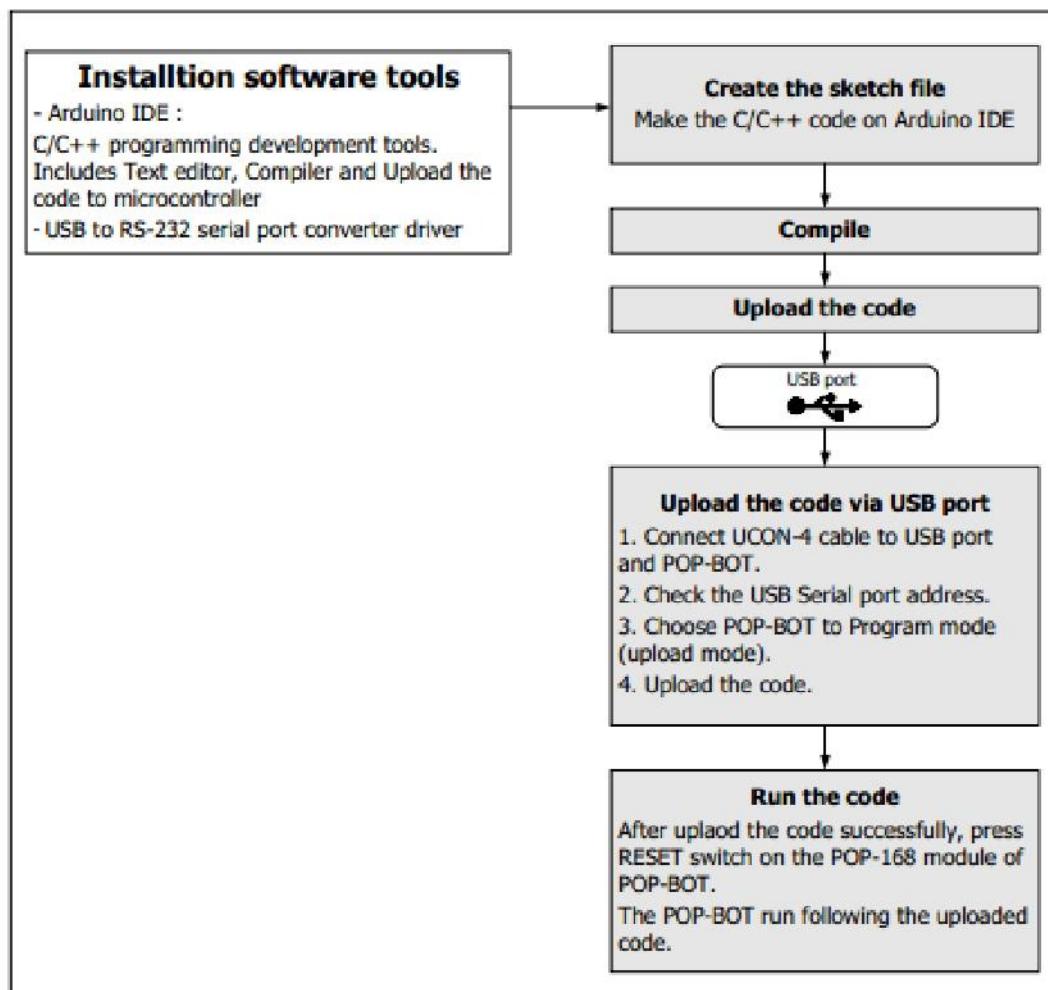


Figure 4-1 Programming development diagram of POP-BOT by using Arduino IDE

## 4.1 Preparación del UCON-4, USB a RS-232 puerto serial del convertidor de cable

El POP-BOT requiere interfaz de la computadora y Arduino para cargar el código. Normalmente se utiliza puerto RS-232 serial o puerto COM. Para la computadora moderna proporciona el puerto USB como interfaz principal. Por tanto, el USB a RS-232serial convertidor de puerto se requiere. En THER POP-BOT kit preaparares la UCON- 4 cable para este propósito.

Antes de utilizar el cable de UCON-4, debe instalar el controlador adecuado y comprobar algunas configuraciones.

### 4.1.1 Instalación del controlador

Haga doble clic en el archivo USBDriverInstallerV2.0.0.exe de POP-BOT CD-ROM para iniciar la instalación del controlador. La instalación de la caja de diálogo aparecerá a continuación.

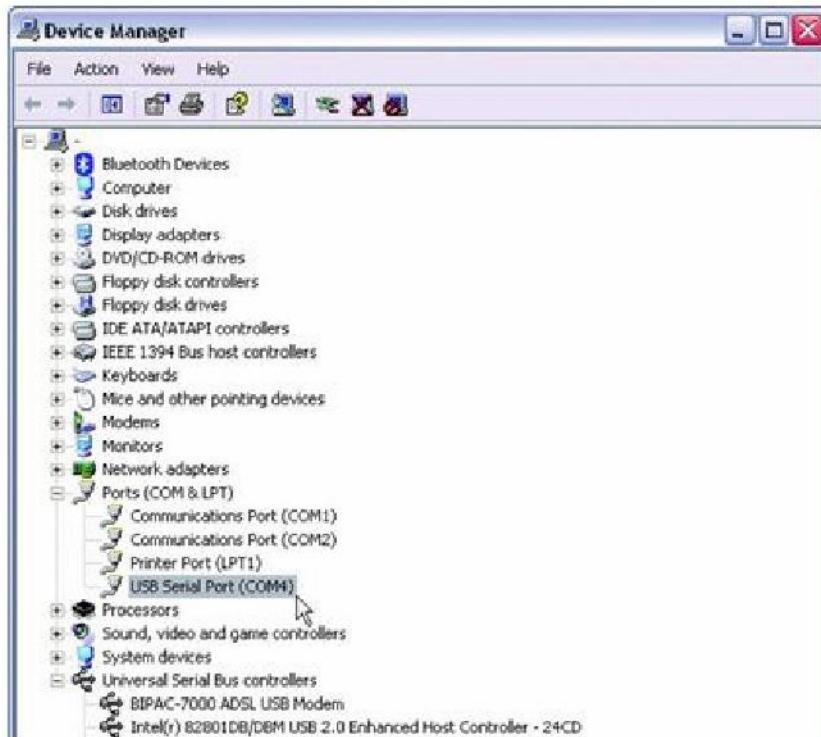


### 4.1.2 Compruebe la dirección del puerto USB de serie

- (1) Conecte el cable USB al puerto USB y POP-BOT panel de control. Espere un momento.
- (2) Compruebe el puerto COM virtual o la dirección del puerto USB de serie, haga clic en Inicio → Panel de Control → Sistema → hardware → Administrador de dispositivos.



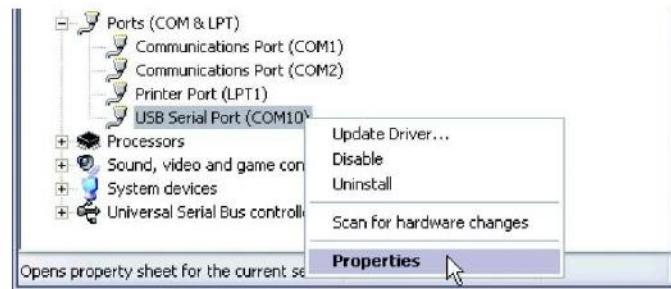
(3) Véase la lista de puerto serie USB y recordar la dirección del puerto COM para trabajar con UCON-4 por cable. Normalmente se creará COM3 o superior. En este ejemplo es COM4.



#### 4.1.3 UCON-4 por cable con aviso de operación Arduino

Normalmente el software de Arduino puede interactuar con el puerto COM no mayores que COM9. De este modo, el usuario debe asegurarse de que la dirección del puerto USB de serie no mayores que COM9. Si es mayor, por favor, realice el procedimiento siguiente.

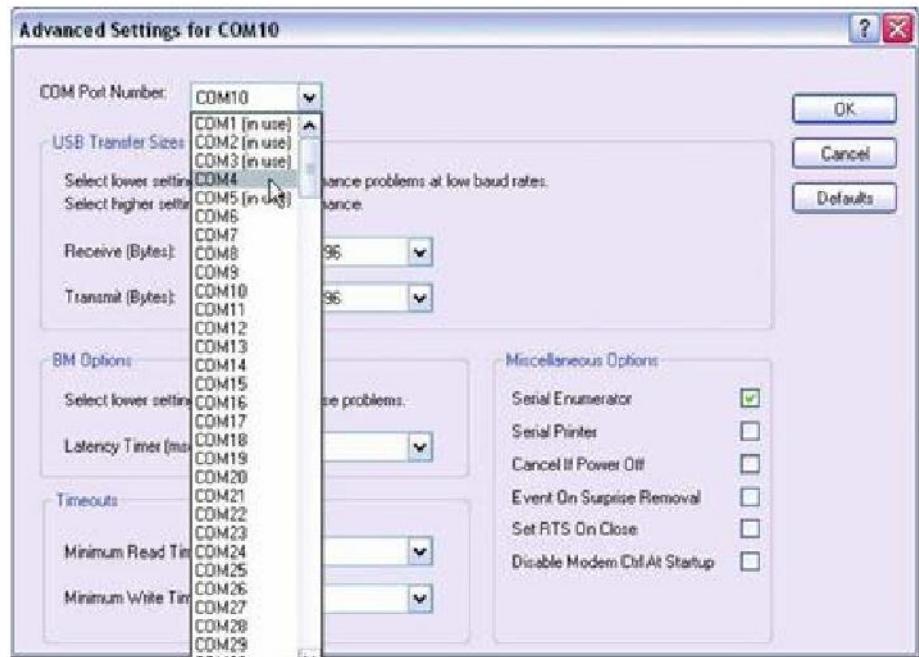
- (1) Conecte el cable de UCON-4 para puerto USB del ordenador.
- (2) Compruebe la dirección del puerto COM, haga clic en Inicio → Panel de control → Sistema
- (3) Seleccione la pestaña Hardware y haga clic en el botón Administrador de dispositivos.
- (4) Compruebe la lista de hardware. En la lista Puerto, se encuentra el puerto USB de serie (COM x). Si el puerto COM es mayor que COM9 (este ejemplo es COM10), haz clic en el botón derecho del ratón y seleccione Propiedades.



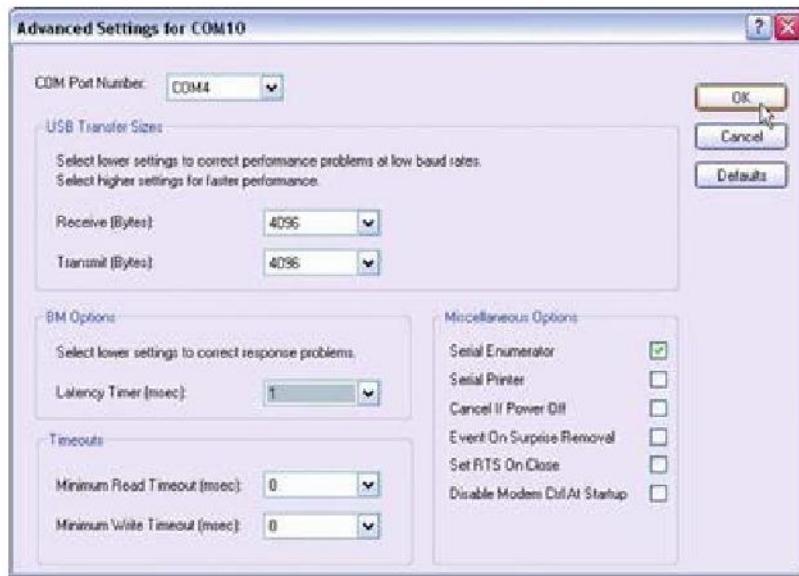
(5) El USB Serial Port (COM10) aparecerá la ventana Propiedades. Seleccione la pestaña Configuración de puerto y establecer todos los valores tras la figura de abajo y haga clic en el botón de avance.



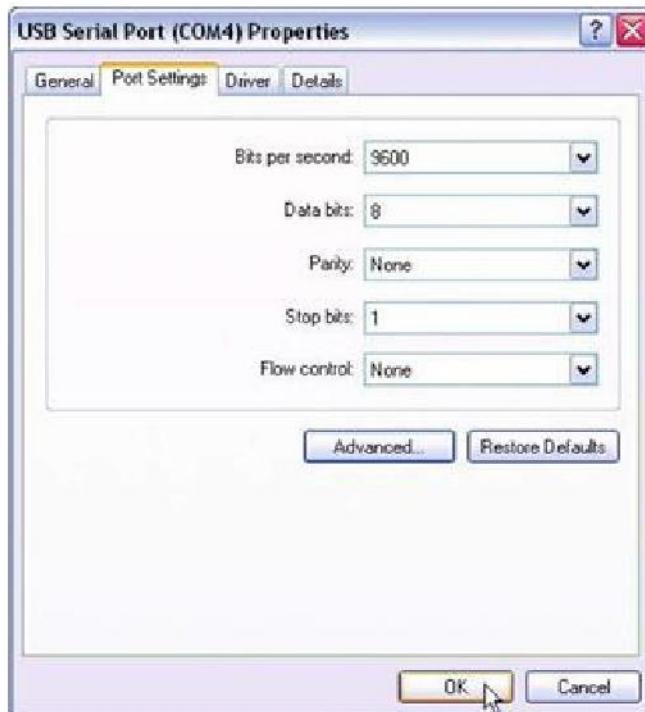
(6) La opción avanzada para COM10 aparecerá. Haga clic en la casilla Número de puerto COM para cambiar a otro puerto COM4 o en el rango de COM1 a COM9.



(7) Establecer el valor que sigue la siguiente figura. Especialmente en el temporizador de latencia (ms) sugirió que el valor 1 y marque la casilla en el Enumerador de serie. Haga clic en el botón Aceptar.



(8) Volver a las propiedades del puerto serie USB. Ahora el número de puerto COM en la barra de título cambiará a COM4. Haga clic en el botón Aceptar.



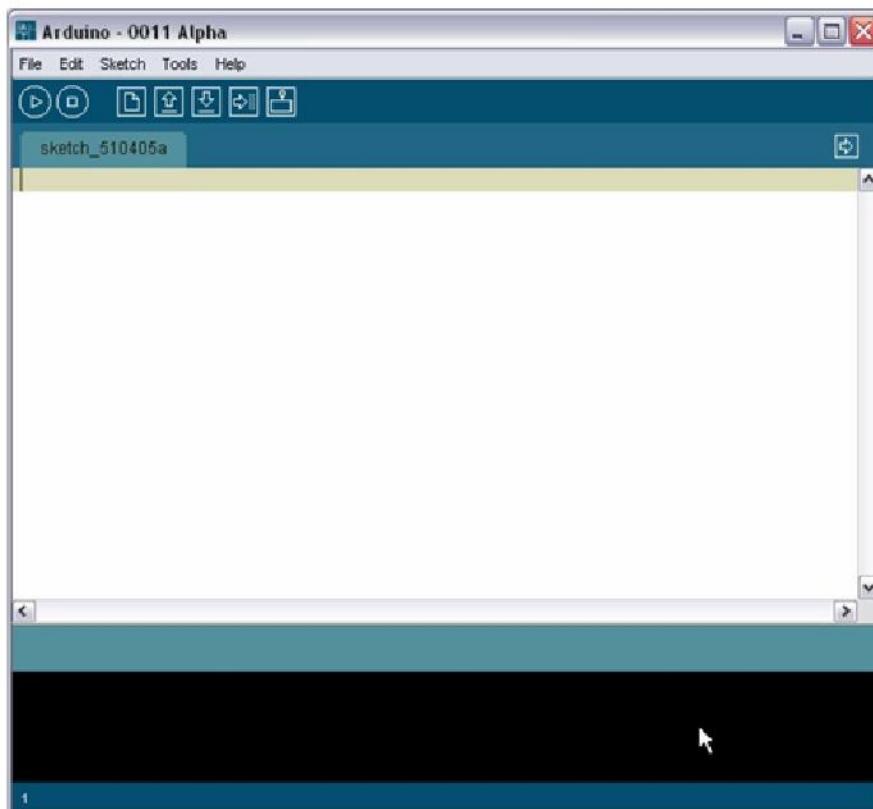
(9) Desconecte el cable de UCON-4 desde el puerto USB y vuelva a conectar de nuevo. Compruebe la dirección del puerto USB de serie. La nueva dirección debe ser COM4. Ahora el cable UCON-4 listo para usar con el software de Arduino IDE.

## 4.2 Obtención de inicio POP-BOT con Arduino

Ejecutar Arduino IDE clicking el menú Inicio - Todos los programas - POP-168 Paquete de Software

→ Arduino

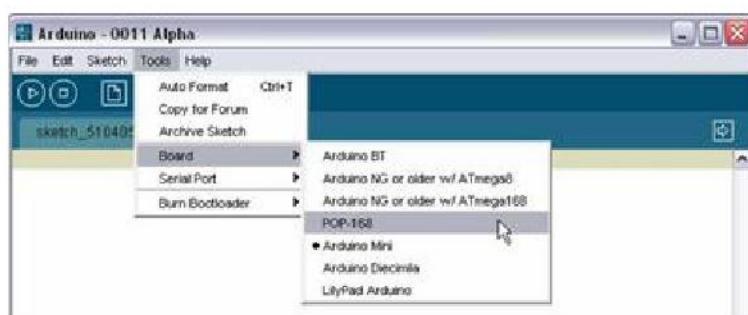
El primer lanzamiento de Arduino mostrará la pantalla de abajo.



### 4.2.1 Arduino POP-168 configuración de hardware

#### 4.2.1.1 Selección del chip microcontrolador

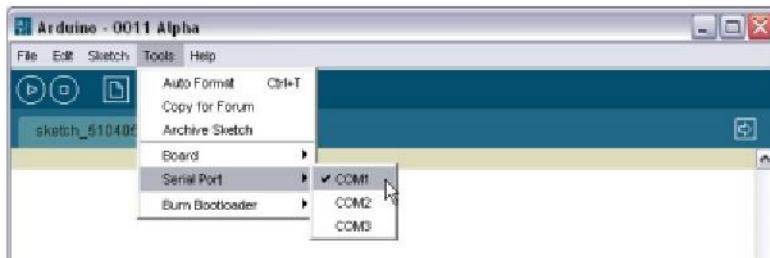
Selección por el menú Herramientas - Junta - POP-168 o el Arduino Mini (puede usar ambas versiones)



#### 4.2.1.2 Seleccione el puerto COM

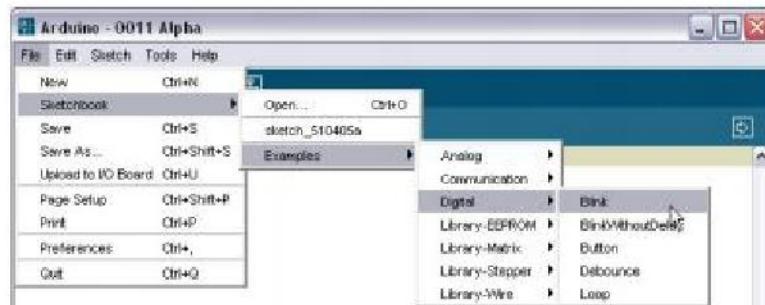
Cargar el boceto de Arduino IDE para POP-168 módulo requiere la comunicación del puerto serie. Se puede trabajar con el puerto COM virtual que crea a partir de convertidor de USB a puerto serie.

Seleccione el menú **Tools → Serial Port**. Usted puede seleccionar el puerto COM de destino.



#### 4.2.2 Abra el dibujo de ejemplo

Seleccione menú **File → Sketchbook → Example → Digital → Blink**



El código de ejemplo; **Blink.pde** van a aparecer en el área de edición de texto.



```
/*
 * Blink
 *
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

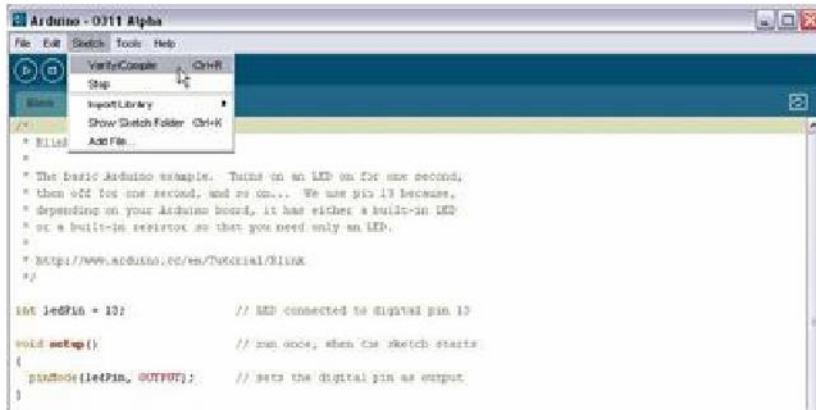
int ledPin = 13; // LED connected to digital pin 13

void setup() // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() // run over and over again
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for a second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000); // waits for a second
}
```

#### 4.2.3 Compilar el esquema

Después de abrir el archivo de dibujo y edición de listas, se puede compilar el boceto mediante la selección de **Sketch → Verify/Compile** menú o haga clic en el botón .

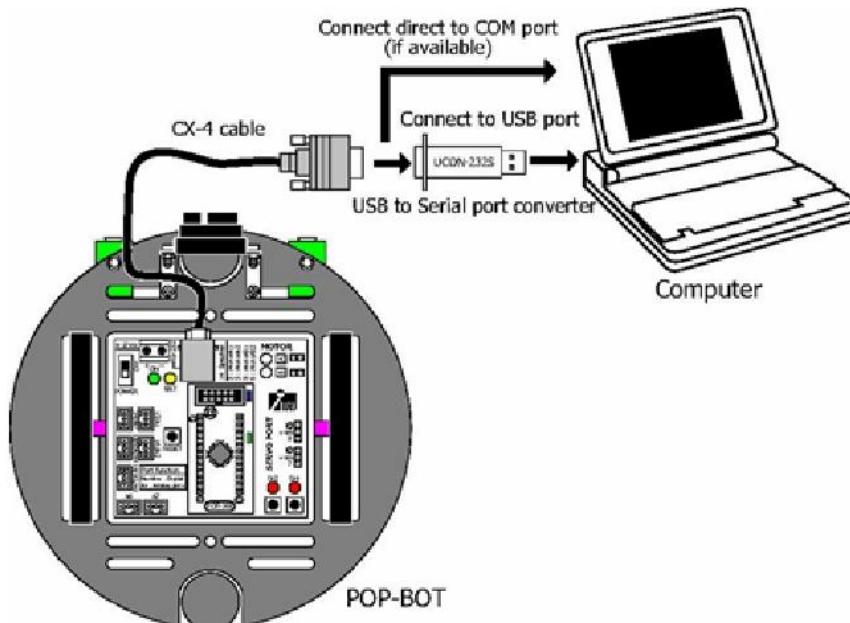


La barra de estado en la parte inferior de la pantalla principal se mostrará el estado de la compilación. Si compila sin error, informa la compilación Hecho y el área de texto mostrará un mensaje de tamaño boceto binario.

#### 4.2.4 Cargar el esquema de POP-168 del módulo

La descarga del código de la máquina de la compilación de hardware de Arduino se llama Carga. Debe preparar el hardware Aruino listos para ser descargados mediante el establecimiento de la COP-168 a modo de gestor de arranque. El procedimiento es:

- (1) Conecte el POP-BOT con el puerto COM de la computadora por CX-4 a través de cable o convertidor de USB a puerto serial.



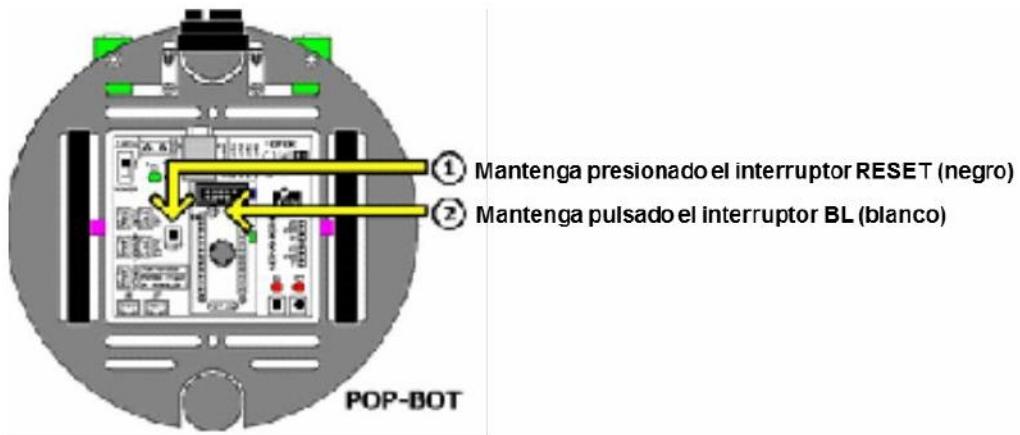
(2) Ajuste POP-168 del módulo al modo de programa. Tiene 2 opciones.

**(2.1) Use el interruptor de RESET en RBX-168 y el interruptor de placa BL en el POP-168 del módulo.**

(2.1.1) Activar el POP-BOT

(2.1.2) Mantenga pulsado el interruptor RESET de la placa controladora RBX-168.

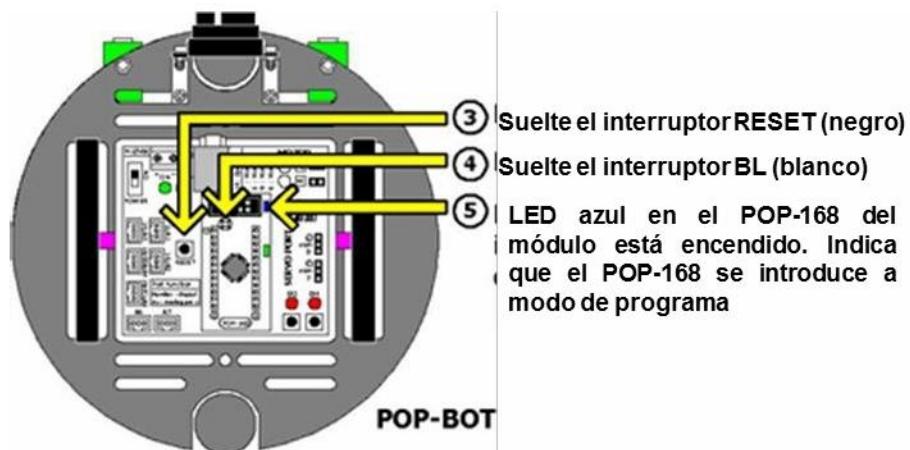
(2.1.3) Mantenga pulsado el interruptor BL en el módulo de POP-168.



(2.1.4) Suelte el botón RESET.

(2.1.5) Suelte el siguiente interruptor BL.

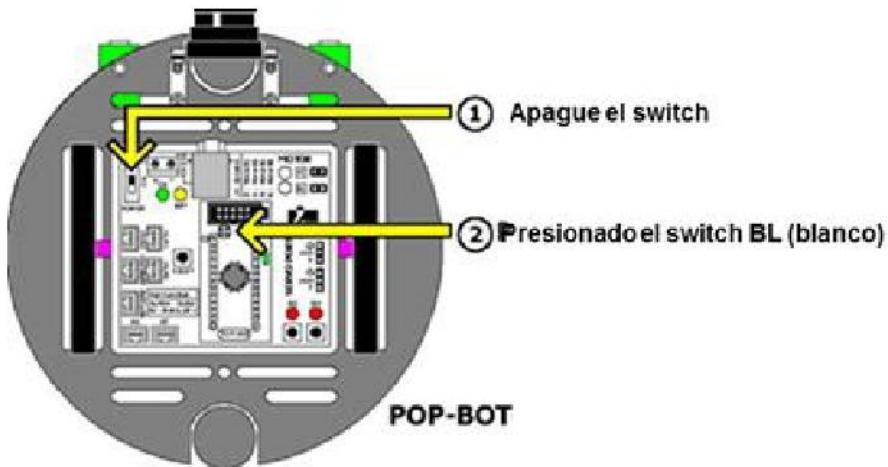
Si la luz azul en el POP-168 se enciende y no parpadea, entonces el POP-168 a entrado en modo de arranque y listo para cargarle datos.



## (2.2) CONSUMO DE ENERGÍA T swtich el RBX-168 y el interruptor de placa BL en el POP-168 del módulo

(2.2.1) Apague el POP-BOT.

(2.2.2) Mantenga pulsado el interruptor BL.



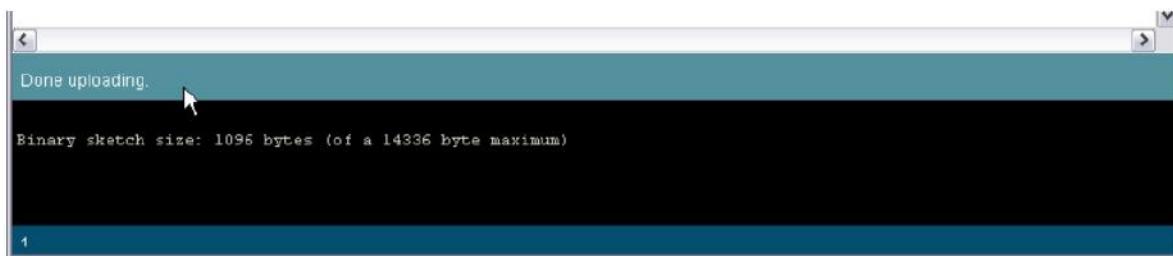
(2.2.3) en mano en el interruptor POWER

(2.2.4) Suelte el interruptor BL

Si la luz azul en el POP-168 se enciende y no parpadea, el POP-168 habrá entrado en modo de arranque y listo para cargarle datos.

(3) En Arduino IDE, seleccione el menú **File → Upload y I/O Board**. Espere a que cargue la información.

(4) Cuando se complete la carga, la barra de estado en la parte inferior de la pantalla principal aparecerá **Done Uploading**.



(5) Después de que la carga haya terminado, presione RESET. El esquema se ejecutará en el POP-BOT.

*El LED en Di113 (LED azul) en el POP-168 parpadeará 1 de segundo.*

# 5: POP-BOT actividades de movimiento

En esta sección se describe cómo conducir motor de corriente continua con PWM (Pulse width modulation) y la forma de generar la señal PWM de Arduino POP-168 en el microcontrolador de programación C.

## 5.1 Funcionamiento básico de la conducción del motor DC con PWM

Al cambiar (modulación) la anchura del pulso aplicado al motor de corriente continua se puede aumentar o disminuir la cantidad de energía suministrada al motor, lo que aumenta o disminuye la velocidad del motor. Nótese que, aunque el voltaje tiene una amplitud fija, tiene un ciclo de trabajo variable. Eso significa que mientras más ancho el pulso, mayor será la velocidad.

Consulte la Figura 5-1. La velocidad depende del tiempo Ton (tiempo de encendido del motor). En este momento, el motor de corriente continua recibe el voltaje completo; Vm. Si la anchura es más Ton, el motor de corriente continua recibe más tensión. La relación entre el tiempo Ton en porcentaje con el período (T) se llama ciclo de trabajo. Se puede calcular de la siguiente manera:

$$\% \text{ duty cycle} = 100 \times \frac{T_{on}}{T_{on} + T_{off}} \quad \dots \dots \dots \quad (5.1)$$

$$\text{PWM frequency} = \frac{1}{T_{on} + T_{off}} = \frac{1}{T} \quad \dots \dots \dots \quad (5.2)$$

$$\text{Average DC motor voltage drop} = \text{Supply voltage} \times \text{duty cycle (\%)} \quad \dots \dots \dots \quad (5.3)$$

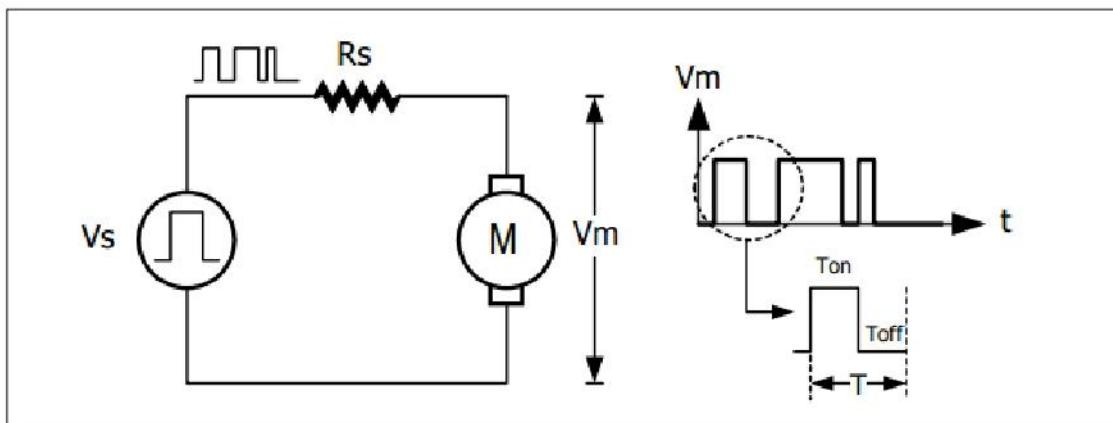
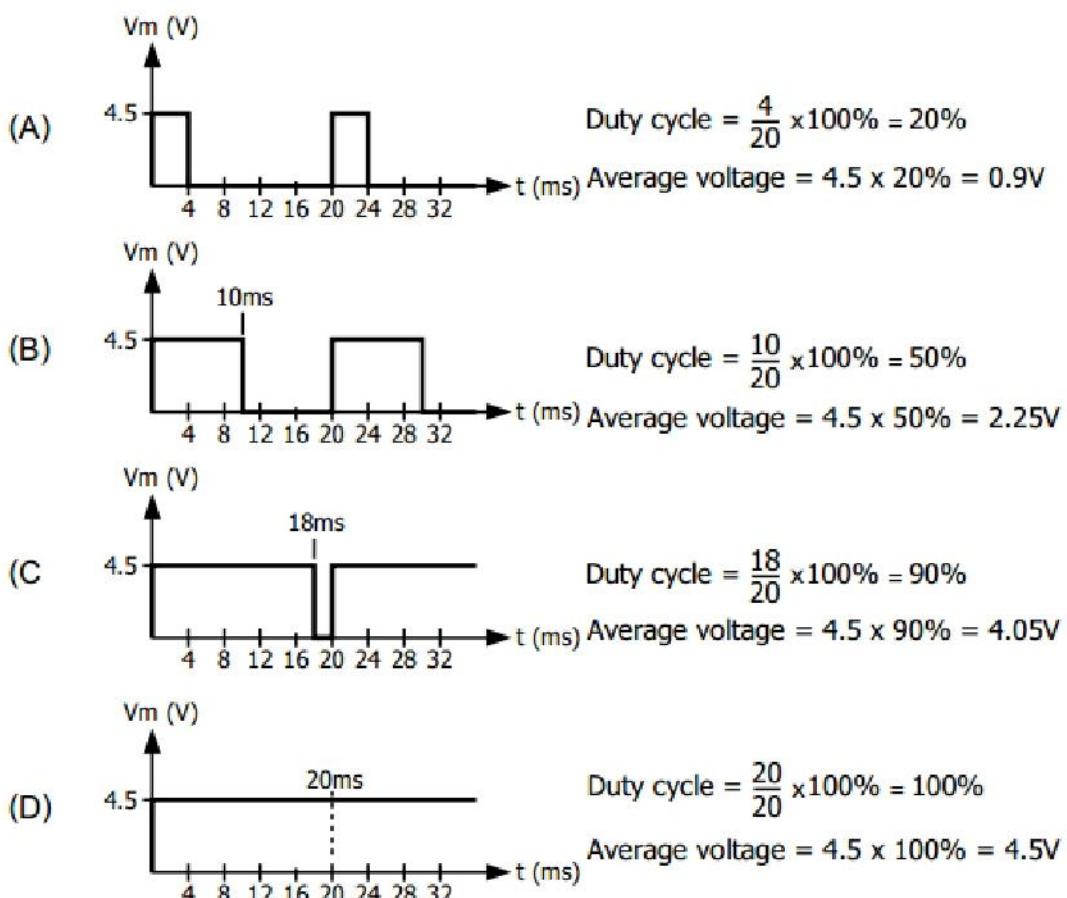


Figura 5-1



**Figura 5-2 : Muestra la relación entre el ciclo de trabajo diferente y la tensión del motor DC**

## 5.2 Arduino con PWM

Arduino tiene una función especial para generar la señal PWM y salidas a los pines digitales. Es **analogWrite()**. El usuario puede ajustar el ciclo de trabajo PWM 0 a 100% con un valor entre 0 y 255.

En valor = 0, no hay señal PWM. Salida de tensión en 0V.

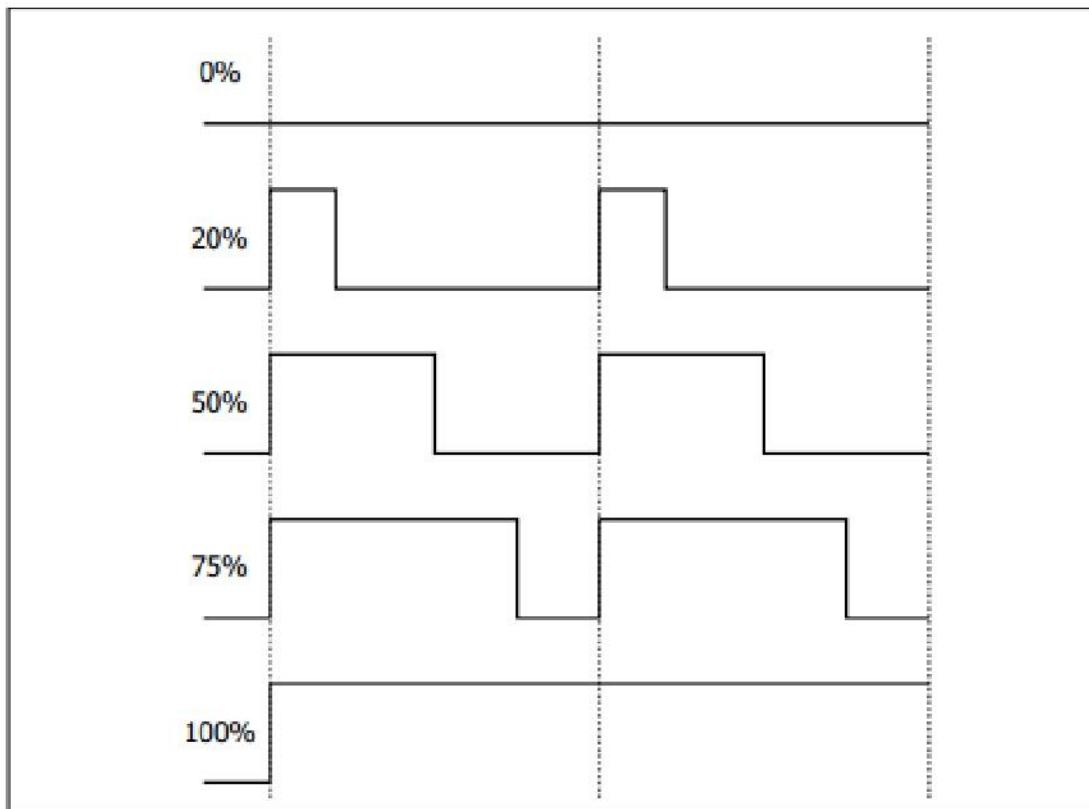
En valor = 51, La señal PWM tiene un ancho de pulso positivo del 20% del período. El ciclo de trabajo es igual a 20%.

En el valor = 127, La señal PWM tiene la mitad de ancho de pulso positivo. El ciclo de trabajo es igual a 50%.

En el valor = 191, La señal PWM tiene un ancho de pulso positivo del 75% del período. El ciclo de trabajo es igual a 75%.

En el valor = 255, La señal PWM tiene ancho de pulso positivo. El ciclo de trabajo es igual a 100%.

La figura 5-2 muestra la señal PWM en cualquier ciclo de trabajo.



**Figura 5-2 anchos de señal PWM en cualquier ciclo de trabajo**

La tensión de salida de señal PWM es el valor promedio relacionado con el ciclo de trabajo. Puede calcularse a partir de la relación siguiente:

$$\text{Outout\_voltage} = (\text{on\_time} / \text{off\_time}) * \text{max\_voltage}$$

Podemos utilizar la señal PWM de **analogWrite()** para ajustar el brillo del LED o amplificar para impulsar el motor de corriente continua. El pasador que Arduino asigna da a la salida PWM el PWM continuará hasta que haga la función de la analogWrite() en el nuevo período o **digitalRead** excecute y funtion **digitalWrite()** en el mismo pin.

Arduino POP-168 del módulo cuenta con 4 pines de salida analógicas, incluye el pin 3, 5, 6 y 9 (Di3, Di5, Di6 y Di9).

La función analogWrite es:

**AnalogWrite (pin, valor);**

Por lo tanto, pines como puerto de El Arduino el pin 3, 5, 6 y 9

De valor como valor de Ciclo de 0 a 255.

## Actividad 1: POP-BOT movimiento básico

### Actividad 1-1 adelante y atrás el movimiento

A1.1 Abrir el IDE de Arduino y crear el código de A1-1.

A1.2 Poner el POP-BOT en modo de Programa. Sube el boceto al robot.

A1.3 Apague el robot y retire el cable de descarga.

A1.4 Asegúrese de que el robot esté en una superficie plana. Prenda el robot y observe la operación.

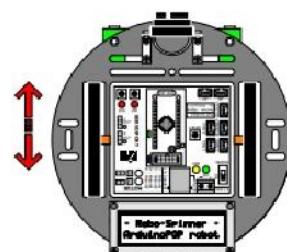
*El POP-BOT se moverá hacia adelante con la luz LED indicadora verde. Después de 1 segundo, ambos indicadores cambian de color a rojo y el robot se mueve hacia atrás.*

*Si esto no es correcto usted tendrá que volver a conectar el cable del motor en su polaridad opuesta. Haga esto hasta que el robot se mueve correctamente. Una vez que el proceso termine, utilice esta configuración de puerto de motor para todas sus actividades de programación. El robot se moverá hacia adelante y hacia atrás continuamente hasta que lo apague.*

---

```
*****
* POP-BOT V1.0
* Running Forward/Backward Full Speed
*****
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
}
void Forward(){                  // Robo-Spinner Go Forward Rountine
    digitalWrite(3,HIGH);
    digitalWrite(5,LOW);
    digitalWrite(6,HIGH);
    digitalWrite(9,LOW);
}
void Backward(){                // Robo-Spinner Go Backward Rountine
    digitalWrite(3,LOW);
    digitalWrite(5,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(9,HIGH);
}
void loop(){
    Forward();
    delay(1000);
    Backward();
    delay(1000);
}
```

---



Listado A1-1: El archivo **Forward\_Backwardr.pde**, de boceto Arduino para la conducción de POP-BOT para avanzar y hacia atrás.

## Actividad 1-2 Controlar el movimiento en forma de Círculo

Con el ajuste de la velocidad diferente para cada motor (rueda), se logra hacer que el robot se mueva en círculo. Puedes probar con este procedimiento de la siguiente manera:

A1.5 Crear un archivo para un nuevo dibujo y escribir los códigos en C. Se muestran en el listado A1-2.

A1.6 Establecer el POP-BOT en modo de Programa. Sube el programa para el robot.

A1.7 Apague y retire el cable de descarga.

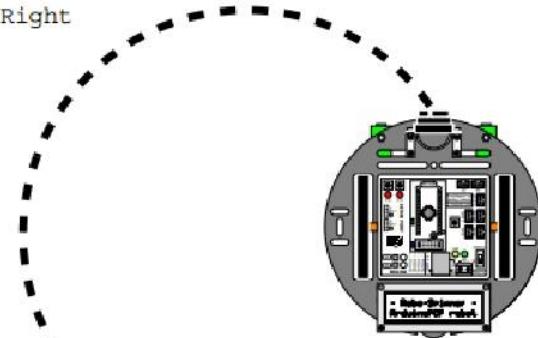
A1.8 Asegúrese de que el robot está en una superficie plana. Prenda el robot y observar.

*El robot se mueve en círculo en forma continua hasta que pulse el interruptor del pin D14 de la tarjeta de conexión POP-BOT para detener el movimiento del robot.*

---

```
*****  
* POP-BOT V1.0  
* Filename : MotorSpeedControl.pde  
* Left Motor Lowspeed and Right Motor Highspeed Robo-Spinner Run in Circle  
*****  
void setup(){  
    pinMode(3,OUTPUT);           // Motor A1  
    pinMode(5,OUTPUT);           // Motor A2  
    pinMode(6,OUTPUT);           // Motor B2  
    pinMode(9,OUTPUT);           // Motor B1  
    pinMode(2,INPUT);            // Switch Left  
    pinMode(4,INPUT);            // Switch Right  
}  
void Forward(int Lspeed,int Rspeed){  
    analogWrite(3,Lspeed);  
    digitalWrite(5,LOW);  
    analogWrite(6,Rspeed);  
    digitalWrite(9,LOW);  
}  
void Motor_Stop(){  
    digitalWrite(5,LOW);  
    digitalWrite(3,LOW);  
    digitalWrite(6,LOW);  
    digitalWrite(9,LOW);  
}  
void loop(){  
    Forward(80,255);           // Circle running  
    if(digitalRead(4)==0){      // if Switch Press  
        Motor_Stop();          // Stop  
        while(1);  
    }  
}  
*****
```

---



**Listado A1-2: Archivo MotorSpeedControlrp.pdee; archivo Arduino para movimiento en círculo de POP-BOT.**

## Actividad 1-3 Controlar el movimiento en forma de Rectángulo

A1.9 Crear un archivo nuevo de dibujo y escribir los códigos en C. Se muestran en el listado A1-3.

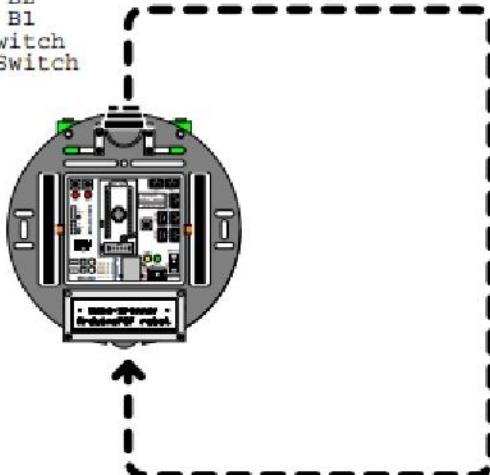
A1.10 Sube el boceto para el robot. Gire a la potencia de despegue y quite el cable de descarga.

A1.11 Prenda y observar el robot.

El robot se activará si SW1 o SW2 está siendo presionado. Si se pulsa SW1, el robot se moverá hacia adelante y girar a la izquierda continuamente, haciendo un cuadrado. Si se presiona SW2, la operación es a la inversa.

---

```
/*
 * Robo-Spinner V1.0
 * Filename : Rectangle_Running.pde
 * Running 90 Degree Turnleft And Turnright
 */
void setup() {
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
    pinMode(2,INPUT);            // LeftSwitch
    pinMode(4,INPUT);            // RightSwitch
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void loop(){
    if (digitalRead(2)==0){      // Switch Di2 Press
        while(1){
            Forward(125);
            delay(900);
            Spin_Left(125);       // Turnleft 90 degree
            delay(400);
        }
    }
    if (digitalRead(4)==0){      // Switch Di4 Press
        while(1){
            Forward(125);
            delay(900);
            Spin_Right(125);      // Turnright 90 degree
            delay(400);
        }
    }
}
```



Listado A1: Archivo en Arduino para el movimiento en forma de Rectángulo

## Actividad 2: POP-BOT Con Parachoques

### Actividad 2-1 Detector de Colisión

Programa para detectar colisiones, con los sensores frontales del POP-BOT.

Después de una colisión que se encuentre, el robot se moverá hacia atrás y cambiar la dirección del movimiento.

A2.1 Abrir el IDE de Arduino y crear código de la lista A2-1.

A2.2 Poner a POP-BOT en modo programa. Subir el código al robot.

A2.3 Apagar el robot y desconectar el cable de descarga.

A2.4 Preparar el área de demostración, mediante la colocación y fijación de cajas u objetos en la superficie.

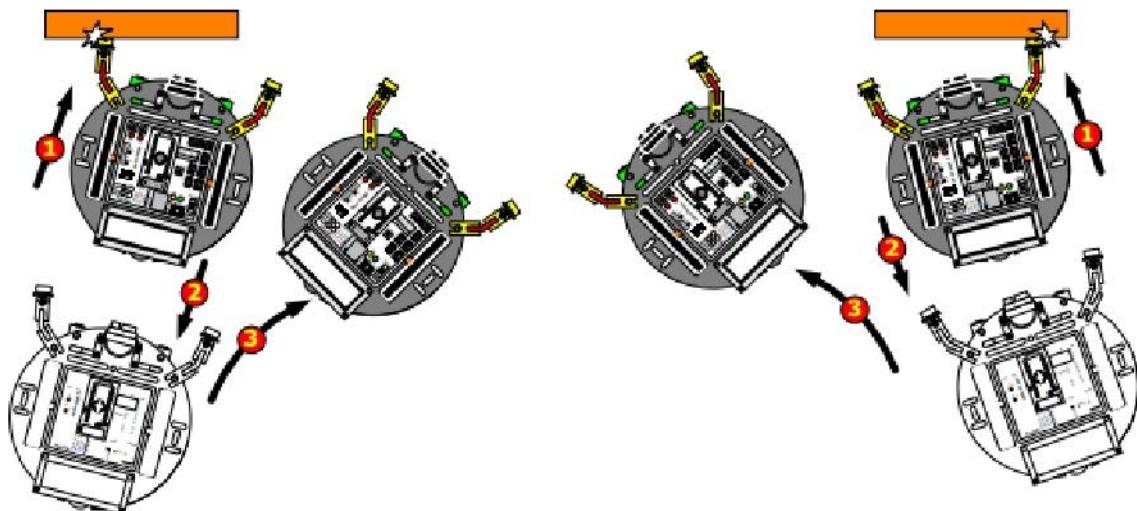
A2.5 colocar el robot en el área de demostración. Prender y observar al robot.

*El POP-BOT con los switch en los puertos 15/A1 y 17/A3 pulsa los objetos.*

*En un operación normal, el robot se moverá hacia delante de forma continua.*

*Si el módulo de comutador izquierda choca contra algún objeto, el robot se moverá hacia atrás y cambiará la dirección a la derecha para evitar el objeto.*

*Si el módulo de comutador de la derecha choca contra algún objeto, el robot se moverá hacia atrás y cambiará su dirección a la izquierda para evitar el objeto.*



*El Robot ataca por la izquierda*

*El Robot ataca por la derecha*

---

```

/*****
* POP-BOT V1.0
* Filename : BumperRobot.pde
* POP-BOT with bumper sensor
*****/
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1

    pinMode(15,INPUT);           // Left Switch
    pinMode(17,INPUT);           // Right Switch
}

void loop(){
    Forward(150);
    if (digitalRead(15)==0){      // Test Bumper Switch From Left
        Backward(150);delay(500);
        Spin_Right(200);delay(400);
    }
    if (digitalRead(17)==0){      // Test Bumper Switch From Right
        Backward(150);delay(400);
        Spin_Left(200);delay(400);
    }
}
/*****
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
*****/

```

---

**Listado A2-1: Archivo BumperRobot.pde, código Arduino para activar Antigolpes en el POP-BOT.**

## Actividad 2.2 Atrapado en una situación de esquina

Cuando el POP-BOT esta en una esquina y se encuentra atrapado en el medio, donde a la izquierda y a la derecha tiene pared. Esto hace que golpee de forma continua las paredes y así queda atrapado en el rincón. La solución es modificar el código de salida C del listado A2-1, la que se muestra en el listado A2-2.

A2.6 Abrir el IDE de Arduino y crear el código que aparece en el Listado de A2-2.

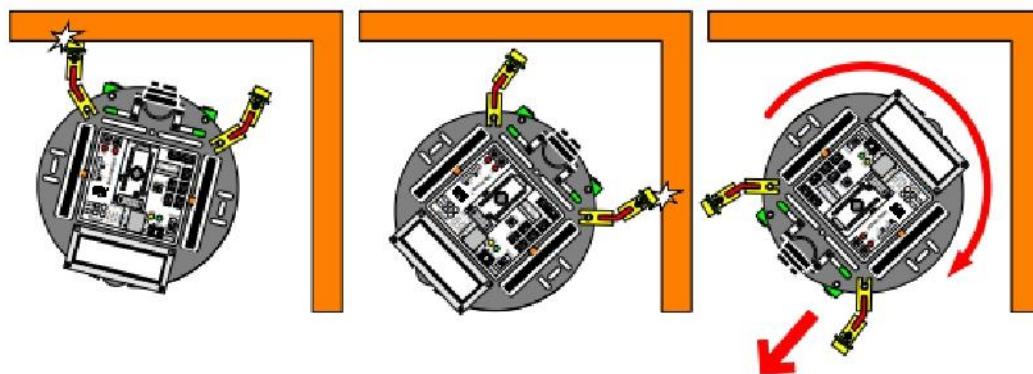
A2.7 Establecer el POP-BOT en el modo de Programa. Subir el código para el robot.

A2.8 Apagarlo y retirar el cable de descarga.

A2.9 Elaborar el área de demostración, colocando y asegurar las cajas o los objetos en la misma superficie de la Actividad 2.1.

A2.10 Coloque el robot en el área de demostración. Prender y observar el robot.

*El robot se moverá hacia adelante y comprobará la colisión. Si esto ocurre más de 5 veces consecutivas, el robot girará 180 grados para cambiar su dirección.*



---

```
/*
 * POP-BOT V1.0
 * Filename : CornerEscape.pde
 * Robot escapes from corner
 */
int Count=0;
int Flag_=0;

void setup(){
    pinMode(3,OUTPUT);          // Motor A1
    pinMode(5,OUTPUT);          // Motor A2
    pinMode(6,OUTPUT);          // Motor B2
    pinMode(9,OUTPUT);          // Motor B1

    pinMode(15,INPUT);          // Left Switch
    pinMode(17,INPUT);          // Right Switch
}
void loop(){
    Forward(150);
    if(Flag_==5){
        Turn();
        Flag_=0;
    }
    if(Count>5){
        Turn();
        Count=0;
    }
    if((digitalRead(15)==1)&(digitalRead(17)==1)){
        Count++;
    }
}
```

```

if (Count>5){                                // Trapped in a corner more than 5 times ?
    Count=0;
    Backward(150);
    delay(2000);
    Spin_Right(200);
    delay(800);
}

if (digitalRead(15)==0){                      // Test left switch
    if(Flag_==1){                            // Check previous state
        Count++;
    }
    else{
        Count=0;
    }
    Flag =0;
    Backward(150);                         // Normal operate
    delay(500);
    Spin_Right(200);
    delay(400);
}

if (digitalRead(17)==0){                      // Test right switch
    if(Flag_==0){                            // Check previous state
        Count++;
    }
    else{
        Count=0;
    }
    Flag =1;
    Backward(150);                         // Normal operate
    delay(400);
    Spin_Left(200);
    delay(400);
}
}*****void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
}*****

```

---

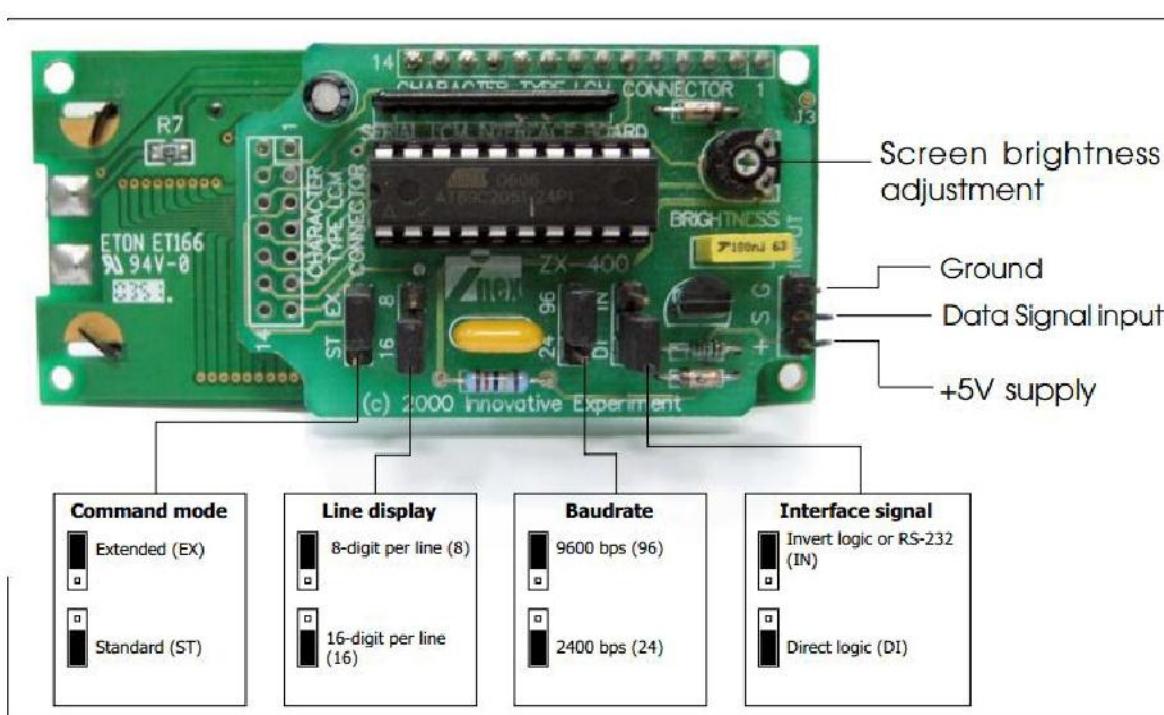
# 6: POP-BOT Pantalla serial LCD

La pantalla LCD serial, modelo SLCD16x2, es de 16 caracteres y 2 líneas. La pantalla LCD se comunica por puerto serial. Recibe datos en serie y se muestran en la pantalla LCD. Acepta datos de serie en 2400 o 9600 baudios. Normalmente la interfaz LCD requiere un mínimo de 6 cables, pero la **SLCD16x2 sólo necesita un cable de señal**. Este módulo es ideal para el robot POP-BOT.

## 6.1 Información de la SLCD16x2

### 8.1.1 Características

- Entrada de serial o Invertir / No invierta nivel lógico TTL / CMOS.
- 1/8 o 1/16 de servicio pueden ser seleccionados por un puente.
- Cuenta con una fácil interfaz con el microcontrolador.
- Funcionamiento con +5 vdc.



### 6.1.2 Configuración de

En la figura 6-1, se muestra el detalle de la parte trasera del SLCD16x2. El usuario verá 4 puentes configurados de la siguiente manera:

**(1) Puente de modo comando:** Selecciona los modos de comando. La SLCD16x2 cuenta con 2 modos. Uno es el comando estándar (ST). Otro modo es el modo extendido de comandos (EX). **Para las actividades de POP-BOT seleccionar el modo de comando estándar (ST).**

**(2) Puente de Líneas:** Selecciona las líneas de pantallas entre valores de 1/8 y 1/16. 1/8 Significa que son 8 dígitos por línea. 1/16 Significa mostrar 16 líneas por cada dígito o más. **Normalmente se establece en 1/16.**

**(3) Transmisión de datos:** 2 selecciones de puente, 2400 y 9600 bps (bits por segundo) con el formato de datos 8N1 (8 bits de datos, sin bit de paridad y 1 stop bit). **Normalmente establecer el POP-BOT en 9600 bps.**

**(4) Interfaz de puente de la señal:** Son 2 señales lógicas, TTL/CMOS de nivel (IN) y la lógica TTL/CMOS de nivel (DI). **Establecer a POP-BOT en DI.**

La SLCD16x2 proporciona un ajuste de brillo con resistencia variable en la posición **BRIGHTNESS**.

El conector de interfaz consta de 3 pinos: +5V, la introducción de datos seriales (S), y la conexión a tierra (G).

### 6.1.3 Interfaz SLCD16x2 con POP-BOT

El cable JST3AA-8 se requiere para la conexión entre la SLCD y la tarjeta controladora del POP-BOT. Este cable conector se muestra a continuación.



El cable JST3AA-8 tiene los dos extremos de 2,00 mm. Se conecta al conector JST de cualquier puerto de la tarjeta controladora del POP-BOT y al conector de entrada de la SLCD16x2.

Después de conectar, configurar todos los puentes de la siguiente manera:

- Seleccione el modo de comando en modo estándar (ST).
- Seleccionar las líneas de la pantalla para 16 dígitos por línea (16).
- Seleccione la velocidad de transmisión de 9600 bps (96).

- Seleccione la señal de la interfaz de directa (DI).

#### 6.1.4 Los datos y el envío de comandos

Una vez que la SLCD16x2 esté correctamente conectada y configurada, los datos y los comandos pueden ser enviados en serie. Para envío de datos, usted puede enviar cualquier mensaje como ¡Hello! a través del serial I/O de forma directa, el mensaje de ¡Hello! se mostrará en la pantalla LCD.

Para el envío de comandos, usted puede enviar instrucciones en la norma establecida LCD (vea la Figura 8-2) y anteponerle el carácter de prefijo de instrucción, ASCII 254 (0FE hexadecimal o binario 11111110). En la pantalla se trata el byte inmediatamente después de un prefijo como una instrucción, a continuación, vuelve automáticamente al modo de datos.

Un ejemplo: Para borrar la pantalla LCD, la instrucción clara es 00000001 binario (o ASCII 1), enviar [254] y [1] para SLCD16x2 (en paréntesis () significan los símbolos bytes individuales establecido en estos valores).

COMMAND\DATA BIT	D7	D6	D5	D4	D3	D2	D1	D0	*	Don't care bit
1. Initial LCD	0	0	0	0	0	0	0	0		
2. Clear LCD	0	0	0	0	0	0	0	1		
3. Return Home	0	0	0	0	0	0	1	*		
4. Entry Mode Setting	0	0	0	0	0	1	I / D	S		
5. Display Setting	0	0	0	0	1	D	C	B		
6. Shift Display	0	0	0	1	S / C	R / L	*	*		
7. Function Setting	0	0	1	*	N	F	*	*		
8. Set CGRAM Address	0	1	A5	A4	A3	A2	A1	A0		
9. Set DDRAM Address	1	A6	A5	A4	A3	A2	A1	A0		

**Standard instruction command set summary**  
(except Initial LCD is addition command.  
Initialize make I/D=1, S=0, D=1, C=0, B=0, N=1, F=0, DDRAM Address=00  
A0 to A7 are CGRAM or DDRAM Address

**Serial input timing diagram**

SERIAL INPUT      Start D0 D1 D2 D3 D4 D5 D6 D7 Stop ← T<sub>p</sub> (Processing time) → T<sub>p MIN</sub> = 5 ms.      Start D0 D1 D2 ...

Figura 6-2 de la SLCD16x2: Resumen de los comandos y diagrama

### 6.1.5 Caracteres LCD

La mayoría de los caracteres del LCD (Figura E) no se pueden cambiar ya que se almacenan en la ROM. Sin embargo, los ocho primeros símbolos, que corresponden a ASCII 0 a 7, se almacenan en la memoria RAM. Al escribir nuevos valores a la RAM de caracteres-generador (CGRAM), puede modificar los que quieras, en el tamaño de 5x8 puntos.

HEX	00h	20h	30h	40h	50h	60h	70h	A0h	B0h	C0h	D0h	E0h	F0h
DEC	0	32	40	48	56	64	72	80	88	96	104	112	120
	160	168	176	184	192	200	208	216	224	232	240	248	
0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
2	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
3	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
4	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
5	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
6	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
7	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

NOTE: Custom characters occupy ASCII 0-7  
ASCII 8-15 repeat the custom characters  
ASCII 128-159 are used for Extended Mode Command only

### Set de caracteres de la LCD. (Construido en HD44780A o SED1278F0A)

Para crear los símbolos señalando la ubicación CGRAM, a continuación, escribir la primera línea cuyos bits forman el patrón deseado, y el punto a la dirección de CGRAM próxima a escribir trozos más tarde. Repita este procedimiento hasta 8 veces (para un símbolo), el símbolo estará listo para usar ahora. CGRAM 0 se encuentra en 0x00 CGRAM Dirección de 0x07, 0x08 CGRAM 1 en la que 0x0F, CGRAM 2 de 0x10 a 0x17, ... hasta CGRAM 7 de 0x38 a 0x3F. Vea la siguiente figura.

Bitmap Layout					Byte Values		
	bit 4	bit 3	bit 2	bit 1	bit 0	binary	decimal
byte 0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	xxx00000	0
byte 1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	xxx00100	4
byte 2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	xxx00010	2
byte 3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	xxx11111	31
byte 4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	xxx00010	2
byte 5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	xxx00100	4
byte 6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	xxx00000	0
byte 7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	xxx00000	0

#### Defining custom symbols.

Example: Load arrow symbol on CGRAM 3, a program would send the following bytes to the SLCD controller.

```
[254], [01011000 b], [0],
[254], [01011001 b], [4],
[254], [01011010 b], [2],
[254], [01011011 b], [31],
[254], [01011100 b], [2],
[254], [01011101 b], [4],
[254], [01011110 b], [0],
[254], [01011111 b], [0]
```

## Instrucción estándar para el conjunto de LCD

Sólo el registro de instrucción (IR) y el registro de datos (DR) de la pantalla LCD puede ser controlado por la MCU. Antes de iniciar el funcionamiento interno de la pantalla LCD, el control de la información se almacena temporalmente en estos registros para permitir la interconexión con diversas MCU, que operan a velocidades diferentes, o varios dispositivos de control de periféricos. El funcionamiento interno de la pantalla LCD está determinada por las señales enviadas desde la MCU. Estas señales, que incluyen el registro de la señal de selección (RS), de lectura / escritura de la señal (R / W), y el bus de datos (DB0 a DB7), constituyen las instrucciones del LCD (Tabla 3). Hay cuatro categorías de instrucciones que:

- Designar funciones a la LCD, como el formato de visualización, la longitud de datos, etc.
- Dirigir el Juego de memoria RAM interna.
- Llevar a cabo la transferencia de datos con la RAM interna
- Realizar funciones auxiliares

Mirando la tabla usted puede hacer sus propios comandos y ponerlos a prueba. A continuación se muestra una breve lista de comandos útiles que se utilizan con frecuencia mientras se trabaja en la pantalla LCD.

Instruction	Hex	Decimal
Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
Entry Mode	0x06	6
Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
Display on Cursor on	0x0E	14
Display on Cursor off	0x0C	12
Display on Cursor blinking	0x0F	15
Shift entire display left	0x18	24
Shift entire display right	0x1C	30
Move cursor left by one character	0x10	16
Move cursor right by one character	0x14	20
Clear Display (also clear DDRAM content)	0x01	1
Set DDRAM address or cursor position on display	0x80+add*	128+add*
Set CGRAM address or set pointer to CGRAM location	0x40+add**	64+add**

\* DDRAM address given in LCD basics section

\*\* CGRAM address from 0x00 to 0x3F, 0x00 to 0x07 for char1 and so on..

## 6.2 Cosas que debe saber sobre la interfaz Arduino con SLCD16x2

El procedimiento de Arduino POP-168 con interfaz Serial LCD es la siguiente:

- (1) Incluye la biblioteca **SoftwareSerial.h** con el comando #include.
- (2) Definir el puerto del POP-168 con el comando #define de la siguiente manera.

```
#define rxPin 3      // Set Di 3 as serial receiver pin or rxPin
#define txPin 2      // Set Di 2 as serial transmitter pin or txPin

SoftwareSerial mySerial = SoftwareSerial (rxPin, txPin);
```

- (3) En el **setup ()**, es necesario configurar el pin para transmitir a la High logic, retrasar y establecer la velocidad en 9600 baudios con mySerial.begin(9600); de comandos. El código de configuración de ejemplo se muestra a continuación.

```
digitalWrite (txPin, HIGH); // set txPin high (as recommended) delay(1000);

// define pin modes for tx, rx pins:
pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
```

```
mySerial.begin(9600); // set baudrate
```

Se recomienda el código de configuración de interfaz con Arduino, con el serie del módulo del LCD o SLCD16x2.

Los comandos importantes para la interfaz con SLCD16x2 del módulo del POP-BOT son los siguientes:

(1) se inicializa:

```
mySerial.print (0xFE,BYTE);  
mySerial.print (Command,BYTE);
```

(2) Limpiar pantalla LCD:

```
mySerial.print (BYTE 0x01,);
```

(3) Mover el cursor a la posición de arriba a la izquierda o la posición de CASA:

```
mySerial.print (0x80, BYTE);
```

(4) Mueve el cursor hacia la posición izquierda de la línea inferior:

```
mySerial.print (0xC0, BYTE);
```

(5) Para escribir el mensaje, el usuario debe poner el mensaje y cubierto por " ".

```
mySerial.print ("Hola"); // Muestra "Hola"
```

### **Actividad 3: SLCD16x2 programación sencilla**

A3.1 Abrir el IDE Arduino y crear el código de boceto de Listado de A3-1.

A3.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto al robot.

A3.3 Restablecer el POP-BOT y observar el funcionamiento SLCD16x2.

El SLCD16x2 muestra el mensaje siguiente:

# POP-BOT

## Hello World !

---

```
/*
 * POP-BOT V1.0
 * Filename : SimpleLCD.pde
 * Show message on SLCD
 */
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);

void setup(){
    digitalWrite(txPin,HIGH);
    delay(1000);
    pinMode(txPin,OUTPUT);
    MySerial.begin(9600);
    delay(1000);
}
void loop(){
    MySerial.print(0xFE,BYTE);
    MySerial.print(0x80,BYTE);
    MySerial.print("POP-BOT");
    MySerial.print(0xFE,BYTE);
    MySerial.print(0xC0,BYTE);
    MySerial.print("Hello World !");
    while(1);
}
```

---

**Listado A3-1: archivo SimpleLCD.pde; el archivo de boceto Arduino para la demostración de la simple operación de la pantalla LCD de serie con POP-BOT**

## Actividad 4: Control de la SLCD16x2 con comandos

Usted puede controlar muchas de las operaciones de visualización de SLCD16x2 como ajustar la visualización de la línea, borrar la pantalla, seleccione el formato de visualización, etc mediante el envío de los comandos de control a SLCD16x2. Para el modo de comandos estándar, inicie byte debe comenzar con 0xFE y después dé la orden. El usuario puede ver el símbolo del LCD en el tema de la información SLCD en este capítulo.

A4.1 Abra el IDE de Arduino y cree el código de boceto de Listado de A4-1.

A4.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A4.3 Restablecer el POP-BOT y observar el funcionamiento SLCD16x2.

---

```
/*
 * POP-BOT V1.0
 * Filename : SLCDrunningText.pde
 * Show running text and number on Serial LCD
 */
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);

void setup() {
    digitalWrite(txPin,HIGH);
    delay(1000);
    pinMode(txPin,OUTPUT);
    MySerial.begin(9600);
    delay(1000);
}
void LCD_CMD(int Command){
    MySerial.print(0xFE,BYTE);           // Command
    MySerial.print(Command,BYTE);
}
void loop(){
    int i;
    LCD_CMD(0x80);                    // First Line
    MySerial.print("POP-BOT");
    LCD_CMD(0xC0);                    // Second Line
    MySerial.print("Hello World !");
    delay(2000);

    LCD_CMD(0x01);                   // Clear Screen Command
    LCD_CMD(0x85);                   // ROW 1,COL 5
    MySerial.print("From");
    delay(500);

    LCD_CMD(0x07);                   // Shift left text
    for(i=0;i<9;i++){
        MySerial.print(" ");
        delay(200);
    }
}
```

```

LCD_CMD(0x05);                                // Shift right text
for(i=0;i<9;i++){
    MySerial.print(" ");
    delay(200);
}

for(i=0;i<9;i++){                            // Blinking text
    LCD_CMD(0x08);
    delay(200);
    LCD_CMD(0x0C);
    delay(200);
}

LCD_CMD(0x00);                                // Show text
MySerial.print("Innovative");                  // Line 1
LCD_CMD(0xC0);
MySerial.print("Experiment");                  // Line 2
delay(5000);
i=0;

// Show Number
LCD_CMD(0x01);                                // Clear screen
MySerial.print("Counter");                     // Line 1
while(1){
    i++;
    LCD_CMD(0xC5);                            // Line 2
    MySerial.print(i,DEC);
    delay(100);
}
}

```

### Descripción del Programa

**Parte 1** iniciar la comunicación del microcontrolador y SLCD16x2.

**Parte 2** Seleccione la línea de meta para mostrar. La línea superior (0x80) está configurado para mostrar .....mensaje. El resultado final (0xC0) está configurado para mostrar el mensaje **POP-BOT**.

**Parte 3** Enviar el comando a la pantalla (0x01) y definir la primera letra en el 5º dígito de la línea superior de la pantalla LCD (0x85) para mostrar el mensaje **FROM**.

**Parte 4** Enviar el comando de desplazamiento a la izquierda (0x07) y el bucle para cambiar el mensaje **FROM** hacia la izquierda.

**Parte 5** Enviar el cambio de comando de la derecha (0x05) y el bucle para cambiar el mensaje **FROM** de vuelta a empezar.

**Parte 6** bucle para enviar el comando de la pantalla Turn-off (0x08) y Turn-on comando de salida (0x0C) y de intercambio. Es hacer que el mensaje **FROM** abra y cierre.

**Parte 7** E la operación 2, cambiar el mensaje de la línea superior **INNOVATIVE** de la línea de fondo, por **EXPERIMENT**.

# 7: POP-BOT línéa dé rastréo

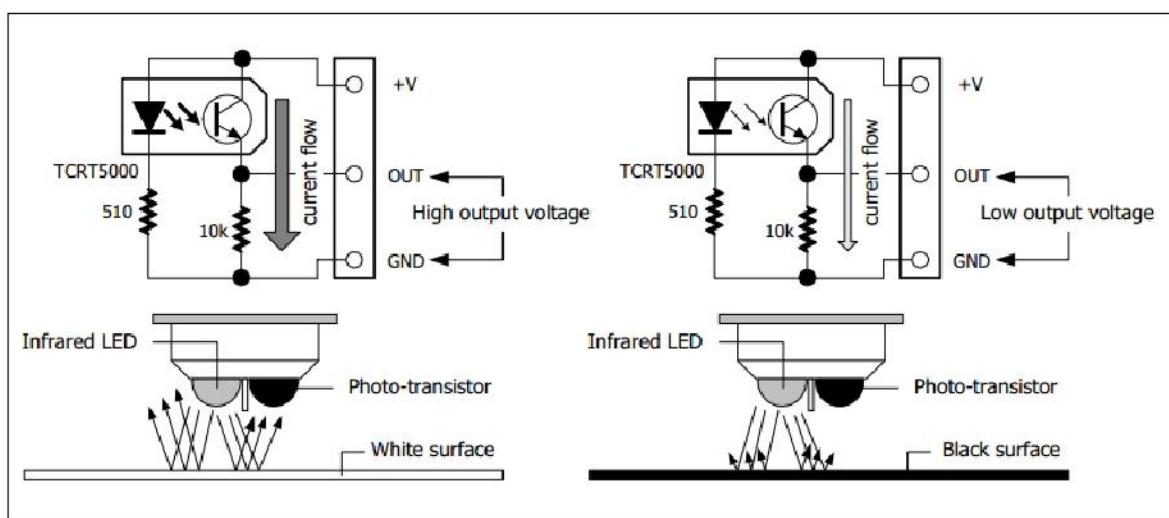
Línea siguiente o el seguimiento de la línea es una actividad muy popular y común en el aprendizaje de la robótica. El propósito de esta actividad es aprender acerca de cómo conectar sensores analógicos. El POP-BOT, tiene un par de sensor reflector de infrarrojos para esta actividad. Dos sensores de reflector de infrarrojos se instalarán en la parte inferior del POP-BOT de manera que pueda detectar ambas líneas blancas y negras.

## 7.1 ZX-03: sensor de infrarrojos del reflector

El corazón de este sensor es TCRT5000 sensor de objeto reflectante. Está diseñado para de detección de proximidad con el infrarrojo (IR). Cuenta con un diodo de infrarrojos detrás de su ventana transparente azul y un transistor de infrarrojos detrás de su ventana en negro. Cuando la infrarroja emitida por el diodo se refleja en una superficie y vuelve a la ventana en negro golpea la base del transistor infrarrojo, haciendo que conduzca corriente. La figura 7-1 muestra el funcionamiento del sensor ZX-03.

Cuando se utiliza como un sensor analógico, el ZX-03 puede detectar tonos de gris en el papel y las distancias en un intervalo corto si la luz en la habitación se mantiene constante.

La distancia adecuada desde el sensor a la línea o en el suelo es de entre 3 a 8 mm. La tensión de salida es de entre 0,1 a 4,8 V y el valor digital de ellos es 10 bits convertidor A/D es de 20 a 1.000. Por lo tanto, el ZX-03 será adecuado para su aplicación a la línea del sensor de seguimiento.



## **7.2 Preparación de la línea de seguimiento de la actividad**

### **7.2.1 Preparación y demostración en el campo de los componentes**

Todas las actividades que se describen en este capítulo utilizan el hacer su propia demostración. Se incluye la superficie blanca y línea negra; y superficie de color negro con el campo de la línea blanca. Usted debe hacer su propio campo con los siguientes elementos (no se incluye en este paquete):

1. Tablero blanco y una hoja de Negro. El tamaño es 90 x 60 cm. Sin embargo, el tamaño puede variar dependiendo de las aplicaciones y recursos.
2. 2 Rollos de sinta aislante, una blanca y la otra negra; de ancho 1 cm.
3. Tijeras o un cortador.

### **7.2.2 Ajuste del valor de referencia para la actividad del seguimiento de la línea con la función analogRead ()**

POP-BOT puede detectar la diferencia entre las líneas y la superficie mediante la lectura del reflector de infrarrojos valor sensores a través de puertos de entrada analógicos. POP-BOT uso de programación **analogRead ()** de Arduino para leer el puerto sensor analógico.

POP-BOT lee la línea de negro y los datos de superficie de bajo valor (menos de 400 y el mínimo es 0) y lee la línea blanca y los datos de superficie de alto valor (superior a 500 y el máximo es 1023). El valor de referencia para tomar la decisión acerca de la línea o superficie es el valor medio de la suma de la superficie del blanco y negro de la siguiente manera:

$$\text{Reference value} = (\text{Valor de la superficie blanca} + \text{valor de la superficie de negro}) / 2$$

La actividad 5 muestra el detalle del valor de referencia para esta actividad de seguimiento de la línea.

## Actividad 5: Área de pruebas en blanco y negro

El robot de POP-BOT está unido con 2 módulos de reflector de infrarrojos en la parte inferior de la base del robot. Por lo tanto, esta actividad sólo voy a detenerme en la programación. Antes de desarrollar el robot para rastrear la línea, los desarrolladores deben programar el robot para detectar la diferencia entre la superficie de blanco y negro.

A5.1 Abrir el IDE Arduino y crear el código de boceto de Listado de A5-1.

A5.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A5.3 Desconecte el cable de carga.

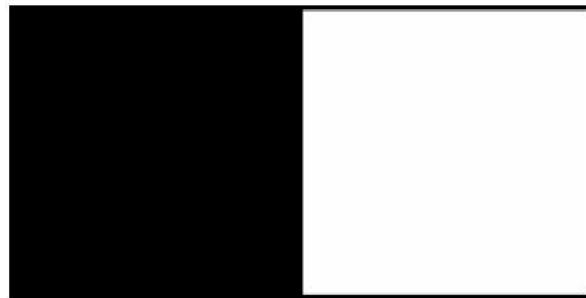
---

```
*****  
* POP-BOT V1.0  
* Filename : AnalogRead.pde  
* Read analog signal from Infrared reflector sensor to show on SLCD  
*****  
#include <SoftwareSerial.h>  
#define rxPin 16  
#define txPin 16  
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);  
int LeftSensor,  
int RightSensor;  
  
void setup(){  
    digitalWrite(txPin,HIGH);  
    pinMode(txPin,OUTPUT);  
    MySerial.begin(9600);  
    delay(1000);  
}  
void LCD_CMD(int Command){  
    MySerial.print(0xFE,BYTE);           // Command  
    MySerial.print(Command,BYTE);  
}  
void loop(){  
    LeftSensor = analogRead(7);          // Read value from left sensor  
    RightSensor = analogRead(6);         // Read value from right sensor  
    LCD_CMD(0x80);                     // Set display to first Line  
    MySerial.print("L Sensor=      ");   // Show left sensor value on 1st line  
    LCD_CMD(0x8A);  
    MySerial.print(LeftSensor,DEC);  
  
    LCD_CMD(0xC0);                     // Set display to second Line  
    MySerial.print("R Sensor=      ");   // Show right sensor value on 2nd line  
    LCD_CMD(0xCA);  
    MySerial.print(RightSensor,DEC);  
    delay(200);  
}
```

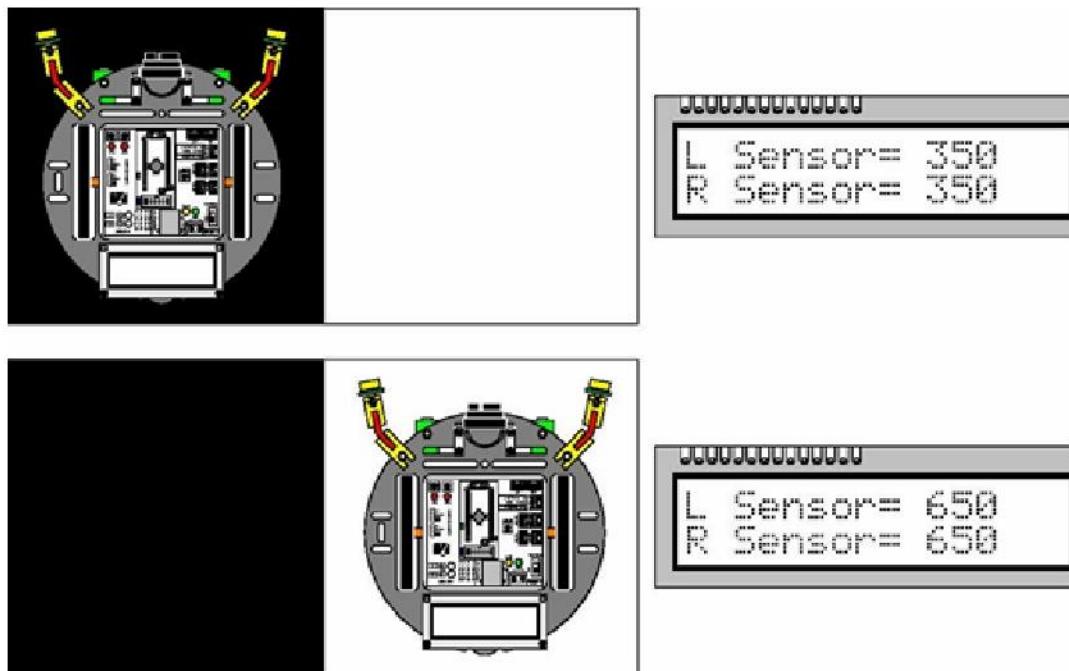
---

**Listado de AS-1:Fila AnalogRead.pde; el programa Arduino es para la lectura del sensor de infrarrojos del display Serial LED del robot POP-BOT**

A5.4 Hacer la hoja de prueba en blanco y negro similar a la ilustración, como se muestra a continuación. El área de superficie blanca es de 30 x 30 cm. y la superficie negra es 30 x 30cm. (recomendado).



A5.5 Coloque el POP-BOT que está programado ya desde el paso A5.3 sobre la superficie blanca de la tabla de prueba. Encienda el robot. Ver el valor de lectura en la pantalla SLCD y grabarlo. Después de eso, leer el valor de la superficie de color negro y registrar el valor también.



El resultado es:

El valor superficie blanca se sitúa entre 500 y 950

El valor superficie negro es entre 100 y 400

El valor de referencia de ejemplo para la detección de la línea es  $(650 + 350) / 2 = 500$ .

## Actividad 6: POP-BOT circulación transfronteriza

Después de conocer los valores de las superficies en blanco y negro, en la siguiente actividad toca seguir con el POP-BOT en el borde negro.

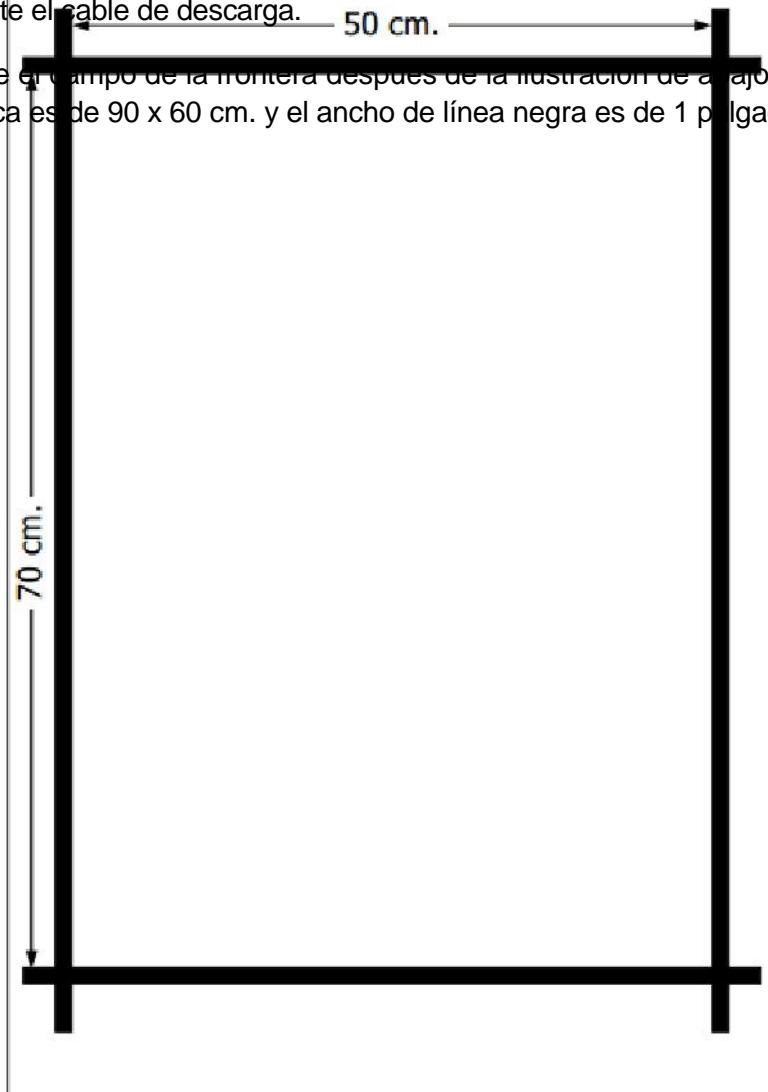
A6.1 Abrir el IDE Arduino y crear el código del listado esbozo A6-1

A6.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A6.3 Desconecte el cable de descarga.

50 cm.

A6.4 Hacer que el campo de la frontera después de la ilustración de abajo. El área de superficie blanca es de 90 x 60 cm. y el ancho de línea negra es de 1 pulgada (2.5 cm.).



---

```

*****
* POP-BOT V1.0
* Filename : BlackBorderMove.pde
* POP-BOT moves in the black border line
*****
int Ref=500;
int Left,Right;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
}
void loop(){
    Left = analogRead(7);          // Read value from the left ZX-03 sensor
    Right = analogRead(6);         // Read value from the right ZX-03 sensor
    if (Left>Ref && Right>Ref){   // Both sensors detect the white surface
        Forward(150);
    }
    else if (Left<Ref && Right<Ref){ // Both sensors detect the black line
        Backward(150);delay(100);
    }
    else if (Left<Ref) {           // Only left sensor detects the black line
        Backward(150);delay(300);
        Spin_Right(150);delay(500);
    }
    else if (Right<Ref){          // Only right sensor detects the black line
        Backward(150);delay(300);
        Spin_Left(150);delay(400);
    }
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}

```

---

**Listado A6 -1: Fila BlackBorderMove.pde, en el boceto Arduino se usapara que POP-BOT se mueva dentro del frontera negra.**

A6.5 Coloque el POP-BOT en el campo de borde negro. Encienda el robot. Observar el movimiento del robot.

POP-BOT se mueve hacia delante sobre la superficie blanca hasta que un sensor detecta el borde negro. Este es el comportamiento del robot:

**Si el sensor detecta la línea negra:** POP-BOT se moverá hacia atrás durante un tiempo corto y luego de nuevo hacia delante.

**Si el sensor detecta a la izquierda la línea negra:** POP-BOT se moverá hacia por un corto tiempo, girara a la derecha y luego de nuevo hacia delante.

**Si el sensor detecta a la derecha la línea de negra:** POP-BOT se moverse hacia atrás durante un corto tiempo, girara a la izquierda y luego hacia delante otra vez.

Por último, POP-BOT se mueve en el borde negro continuamente.

### **Actividad 7: POP-BOT movimiento de ping-pong**

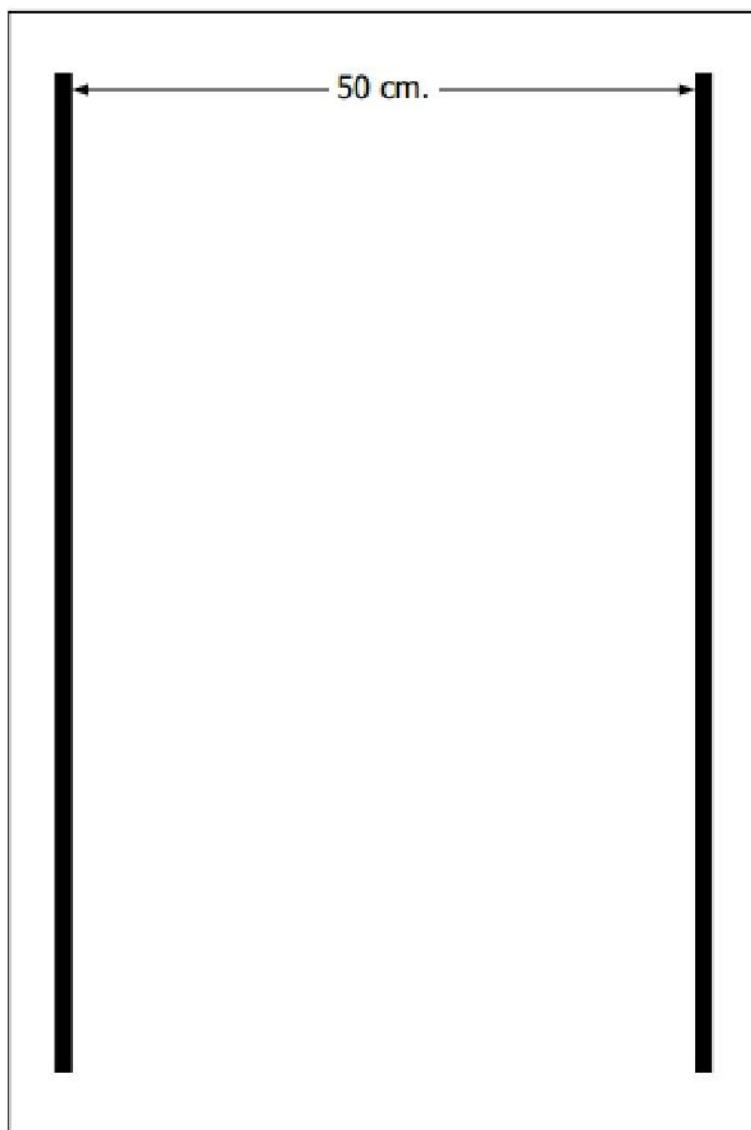
aqui, POP-BOT se mueve en un formato de ZIG-ZAG. La línea de negro es el punto de inflexión. El POP-BOT se mueve hacia adelante hasta encontrar la línea de negro para cambiar la dirección. Se repite esta operación siempre. De este modo, el robot se mueve en el espacio entre las dos líneas negras.

A7.1 Abrir el IDE Arduino y crear el código del listado esbozo A6-1

A7.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A7.3 Desconecte el cable de descarga.

A7.4 El campo de ping-pong, de la ilustración de abajo. El área de superficie blanca es de 90 x 60 cm. y el ancho de línea negra es de 1 pulgada (2.5 cm.).



---

```

/*
 * POP-BOTV1.0
 * Filename : PingPong.pde
 * POP-BOT moves zigzag with 2 parallel lines
 */
int Ref=500;
int Left,Right;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
}
void loop(){
    Left = analogRead(7);         // Read value from the left ZX-03 sensor
    Right = analogRead(6);        // Read value from the right ZX-03 sensor
    if (Left>Ref && Right>Ref){   // Both sensors detect the white surface
        Forward(150);
    }
    else if (Left<Ref && Right<Ref){ // Both sensors detect the black line
        Backward(150),delay(100);
    }
    else if (Left<Ref) {          // Only the left sensor detects the line
        Spin_Right(150);
        delay(420);
    }
    else if (Right<Ref) {         // Only the right sensor detects the line
        Spin_Left(150);
        delay(420);
    }
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}

```

---

**Listado A7-1: Fila PingPong.pde, el boceto Arduino de ping-pong del POP-BOT**

## **Programming concept**

The steps of programming in this activity are :

(1) Find the value between the white surface and black line.

(2) Read the sensor values and store to compare

(3) Check the condition as follows :

**Case #1** : Both sensors detect the white surface

**Action** : Robot moves forward.

**Case #2** : Only the left sensor detects the black line.

**Action** : Robot spins right to change direction for moving to the black lines opposite side.

**Case #3** : Only the right sensor detects the black line.

**Action** : Robot spins left to change direction for moving to the black lines opposite side.

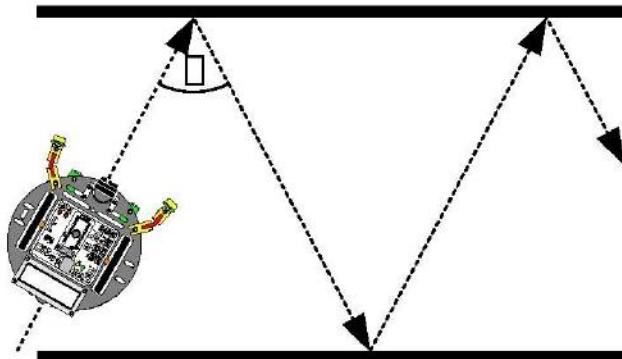
**Case #4** : Out of 3 conditions above

**Action** : Up to programmer purpose.

(4) Back to step (2).

A7.5 Place the POP-BOT on the Ping-pong field. Turn on the robot. Observe the robot movement.

*In this program, determine the turning time to 150 millisecond approximation. It cause the turning angle  $\phi$  (see the illustration below). Programmer can adjust the time value to make the suitable angle for control the movement path correctly.*

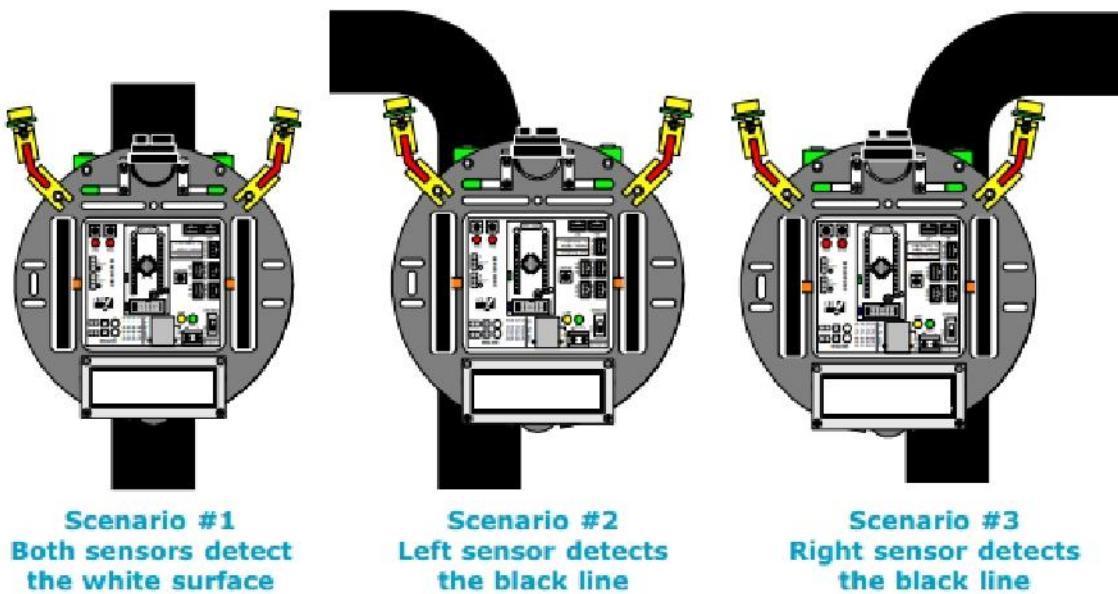


*If the time value is more, the  $\phi$  angle is narrow. It cause the robot moves back to same path. In the other hand, the time value is less, the robot should move out of the line or possible to moves parallel the lines with do not detect the lines*

## Actividad 8: El robot se mueve a lo largo de la línea de negro

El robot se mueve a lo largo de la línea, puede estar en 3 escenarios diferentes.

- (1) **Los sensores leerán los valores que son blanco:** El robot se moverá hacia adelante. Por lo tanto, este programa está escrito para que el robot se mueva hacia adelante con normalidad.
- (2) **El sensor detecta la izquierda de la línea de negro:** Esto ocurre cuando el robot está ligeramente inclinado hacia la derecha. Así, el programa está escrito para que el robot se mueva de nuevo a la izquierda para continuar su camino normal.
- (3) **El sensor detecta la derecha de la línea de negro:** Esto ocurre cuando el robot está ligeramente inclinada hacia la izquierda. Así, el programa está escrito para que el robot se mueva hacia la derecha para continuar su camino normal.



De todos los escenarios, puede hacer que el programa en C de la siguiente manera en el listado A8-1

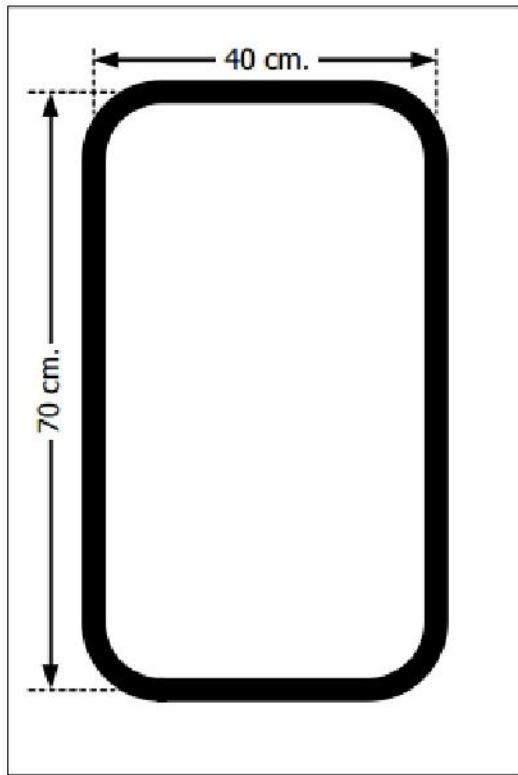
A8.1 Abrir el IDE Arduino y crear el código del listado esbozo A8-1

A8.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto al robot.

A8.3 Desconecte el cable de descarga.

```
*****
* POP-BOT V1.0
* Filename : SimpleLineTracking.pde
* POP-BOT tracks the black line
*****
int Ref=500;
int Left,Right;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
}
void loop(){
    Left = analogRead(7);         // Read value from left sensor
    Right = analogRead(6);        // Read value from right sensor
    if (Left>Ref && Right>Ref){ // Both sensors detect white surface
        Forward(150);
    }
    else if (Left<Ref) {         // Left sensor detects black line
        Spin_Left(250);
    }
    else if (Right<Ref){         // Rightt sensor detects black line
        Spin_Right(250);
    }
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
```

A8.4 Hacer un campo simple de líneas de negro, como el de la ilustración de abajo. El área de superficie blanca es de 90 x 60 cm. y el ancho de línea negra es 1 pulgada (2.5 cm.)



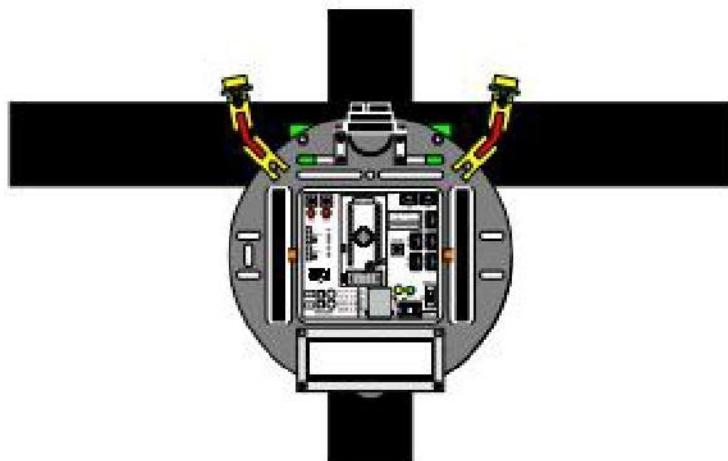
A8.5 Coloque el POP-BOT en el campo de la línea Negra. Encienda el robot. Observar el movimiento del robot.

*El POP-BOT se moverá a lo largo de la línea de color negro. Es posible que el robot se mueva fuera de la línea. Puede mejorar la precisión mediante la edición del programa con el ajuste del valor de referencia del sensor y ajuste a la posición de los dos sensores reflectores infrarrojos.*

## Actividad 9: Detección de la línea de cruce

De la actividad 8, usted puede mejorar el POP-BOT para que se mueva a lo largo de la línea de negro y detecte la unión o la línea 2 con los mismos sensores. Todo lo que tienes que hacer es editar el código del programa.

Cuando el robot se mueve a la línea de negro cruce en T, los dos sensores detectan la línea de color negro. Debe agregar el programa de apoyo a este escenario. La mejora de programa C se muestra en el listado A9-1.



```
*****
* POP-BOT V1.0
* Filename : CrossingLineDetect.pde
* POP-BOT tracks the black line and beep when detect the crossing line
*****
int Ref=700;
int Left,Right;
int Cnt=0,j;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
    pinMode(14,OUTPUT);          // PIEZO Speaker
}
void loop(){
    Left = analogRead(7);        // Read value from left sensor
    Right = analogRead(6);       // Read Value from right sensor
    if (Left>Ref && Right>Ref){ // Both sensors detect the white surface
        Forward(150);
    }
    else if (Left<Ref && Right<Ref){ // Both sensors detect the black line.
        // It's mean crossing line.
        Cnt++;
        Motor_Stop();
        for (j=0;j<Cnt;j++){
            tone(14,500);
            delay(100);
        }
    }
}
```

```

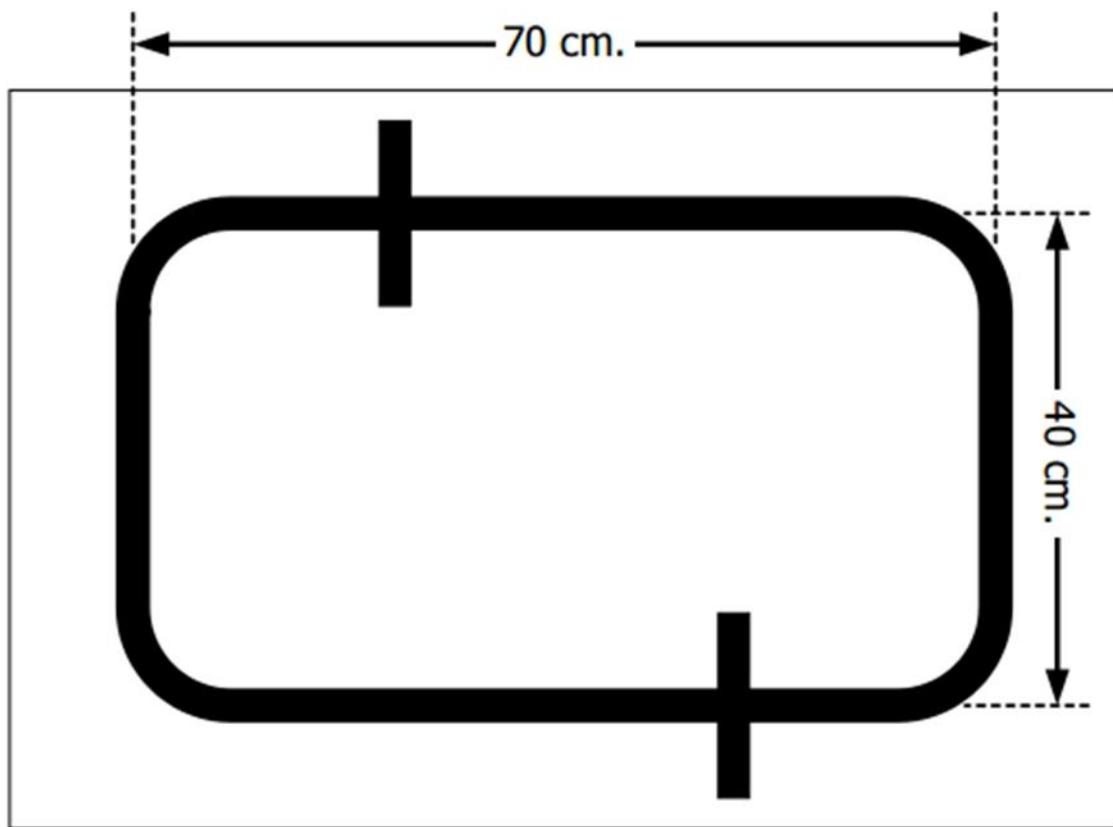
        Beep();
        delay(100);
    }
    Forward(150);
    delay(100);
}
else if (Left<Ref) {           // Left sensor detects the black line
    Spin_Left(255);
}
else if (Right<Ref){          // Right sensor detects the black linek
    Spin_Right(255);
}
}
void Beep(){                   // Beep routine
int i;
for (i=0;i<600;i++){
    digitalWrite(14,HIGH);
    delayMicroseconds(150);
    digitalWrite(14,LOW);
    delayMicroseconds(150);
}
}
void sound(int freq ,int duration){
unsigned long us;
int duration_i;
us=(1000000/(freq*2));
duration_ = (duration/(us*2));
for (i=0;i<duration_i;i++){
    digitalWrite(14,HIGH);
    delayMicroseconds(us);
    digitalWrite(14,LOW);
    delayMicroseconds(us);
}
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Motor_Stop(){
    digitalWrite(3,LOW);
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}

```

---

**Listado de A -1: Fila CrossingLineDetect.pde; el boceto Arduino para POP-BOT se mueve a lo largo del negro y detecta el cruce de líneas**

A9.1 Mejorar el campo de la línea simple de negro de la Actividad 8. Agregue algunas líneas cruzadas. Agregar cuantos cruces desee. Sin embargo, asegúrese de que al menos sean dos robots de ancho de separación cada cruce.



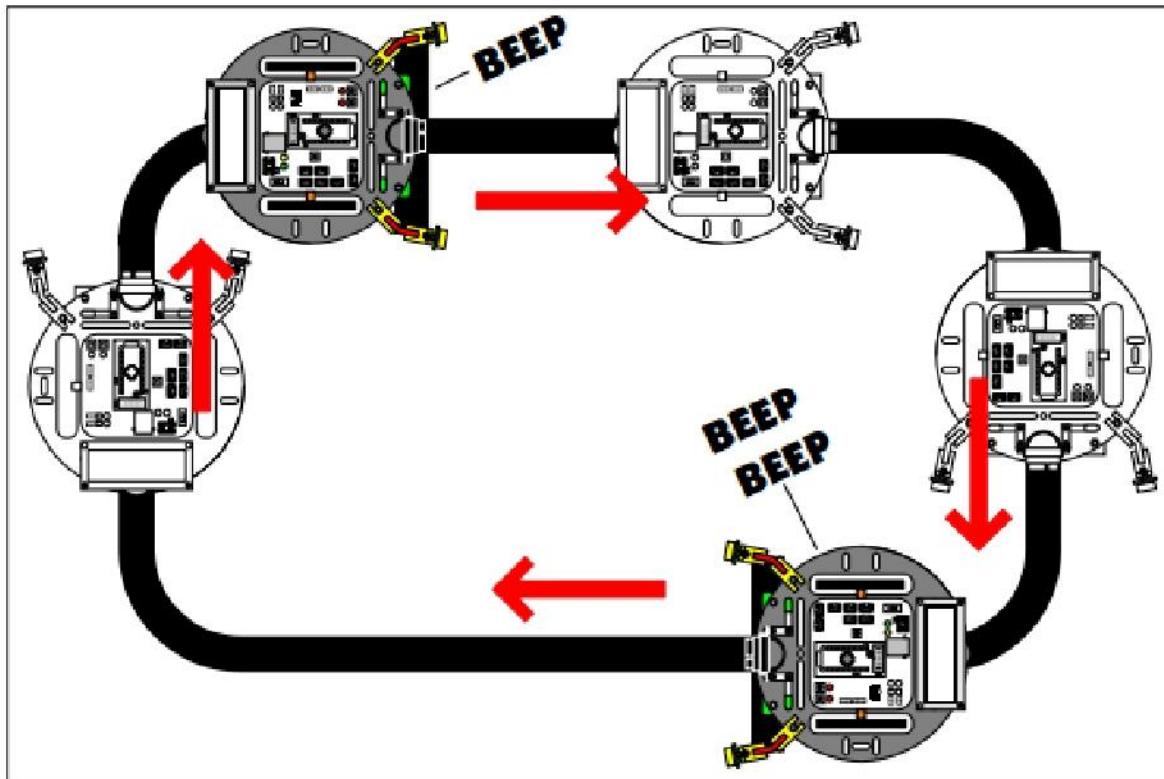
A9.2 Abrir el IDE Arduino y crear el código del listado esbozo A9-1.

A9.2 Establecer el POP-BOT en el modo de Programa. Sube el boceto al robot.

A9.3 Desconecte el cable de carga.

A9.3 Coloque el robot en el campo. Encienda la alimentación. Observar el movimiento del robot.

El robot se moverá a lo largo de la línea de color negro. Cuando el robot detecta la unión, lo frenará y un sonará pitido. Cuando encuentra el segundo cruce, sonará dos veces y esto aumentará las uniones posteriores.



**Nota:** En el funcionamiento del freno de motor, el robot se detendrá y bloqueará el eje motriz inmediatamente. Pero a veces, esto no es suficiente. Usted debe programar el robot para desplazarse hacia atrás por un corto tiempo. Esto hará que el robot pare en su posición.

## Actividad 10: POP-BOT, con 90 grados de inflexión de seguimiento de la línea

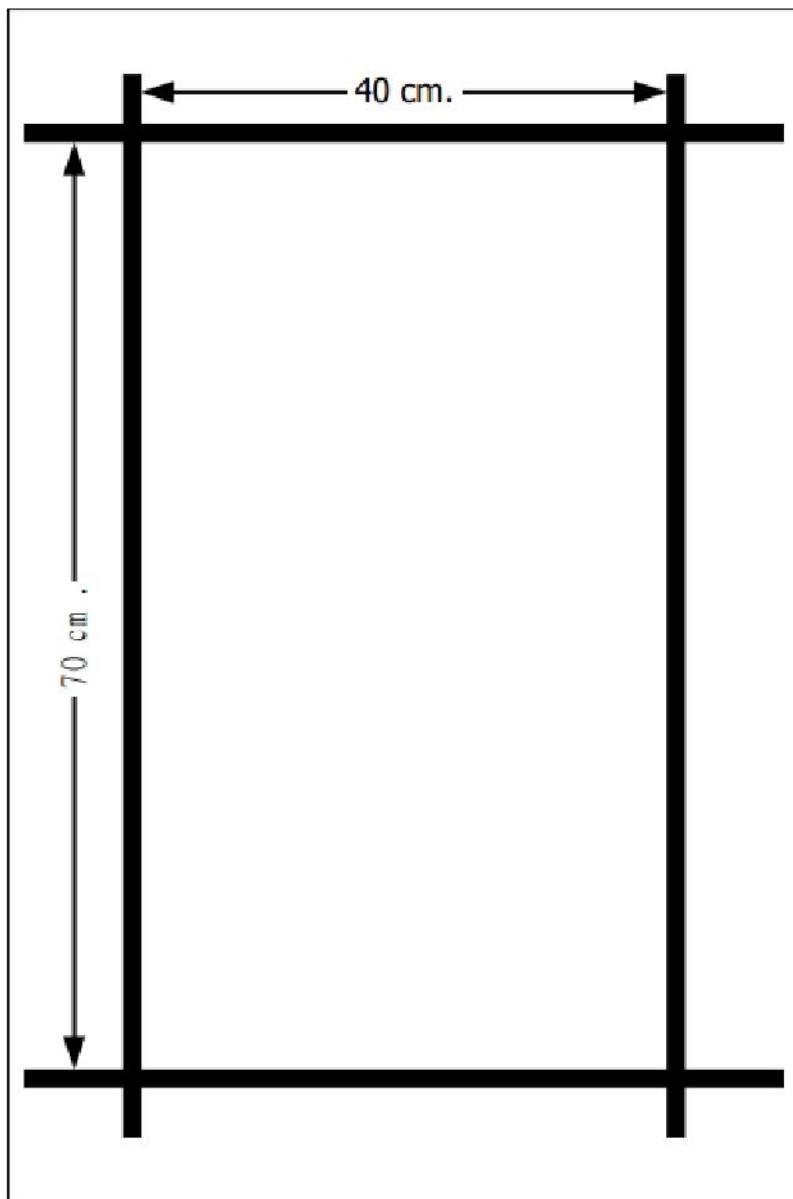
Esta actividad cuenta con alrededor de 90 grados de inflexión cuando el robot detecta el cruce o paso fronterizo. Esta técnica es muy importante en los desafíos robóticos. El robot debe detectar y moverse con precisión suficiente para mantener su movimiento estable.

A10.1 abrir el IDE de Arduino y crear el código del listado esbozo A10-1

A10.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto al robot.

A10.3 Desconecte el cable de descarga.

A10.4 Hacer el campo de la ilustración de abajo. El área de superficie blanca es de 90 x 60 cm. y el ancho de línea negra es 1 pulgada (2.5 cm.).



---

```

*****
* POP-BOT V1.0
* Filename : RightTurnLineTracking.pde
* POP-BOT move following the line and turns right 90° when detect the crossing line
*****
int Ref=700;
int Left,Right;
void setup(){
    pinMode(3,OUTPUT);                      // Motor A1
    pinMode(5,OUTPUT);                      // Motor A2
    pinMode(6,OUTPUT);                      // Motor B2
    pinMode(9,OUTPUT);                      // Motor B1
    pinMode(14,OUTPUT);                     // PIEZO Speaker
}
void loop(){
    Left = analogRead(7);                  // Read value from the left ZX-03 sensor
    Right = analogRead(6);                 // Read value from the right ZX-03 sensor
    if ((Left<Ref) && (Right>Ref)){     // Detect the crossing line
        Right90();
    }
    else if ((Left>Ref) && (Right>Ref)){ // Over the line
        Forward(150);
    }
    else if (Left<Ref) {                // Only the left sensor detects the black line
        Spin_Left(150);
    }
    else if (Right<Ref){               // Only the right sensor detects the black line
        Spin_Right(150);
    }
}
/*Turn right 90 degree function */
void Right90(){
    Forward(150);
    delay(50);
    Spin_Right(200);
    delay(100);
    while(analogRead(6)>Ref);
    delay(50);
}
/*Movement function*/
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}

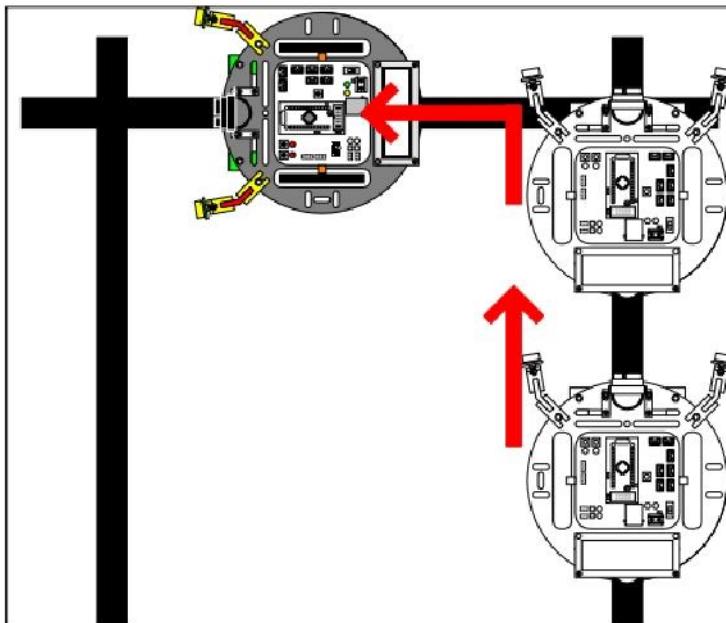
```

---

**Listado A10-1: Fila CrossingLineDetect.pde; el boceto Arduino para el POP-BOT lo mueve en la línea negra y detecta la línea de cruce**

#### A10.5 Coloque el POP-BOT sobre la línea. Encienda y observar el movimiento del robot.

POP-BOT se mueve a lo largo de la línea. Cada robot detecta la línea de cruce y gira a la derecha con un ángulo de 90 grados y continúa siguiendo la línea.



El factor más importante de esta actividad es la función **Right90**. Se trata de la función para Arduino C/C++ . Podemos describir la operación de la función como sigue:

1. Después de detectar el sensor la unión, POP-BOT debe avanzar 0,05 segundos para ajustar la posición en el centro de cruzar la línea.
1. Haga girar a la derecha y retrasar 0.1 segundos
2. En el cruce lee el valor del sensor de la derecha y regresa hasta que detecta la línea de color negro.
3. Retraso 0,05 segundos antes de volver al cruce principal.

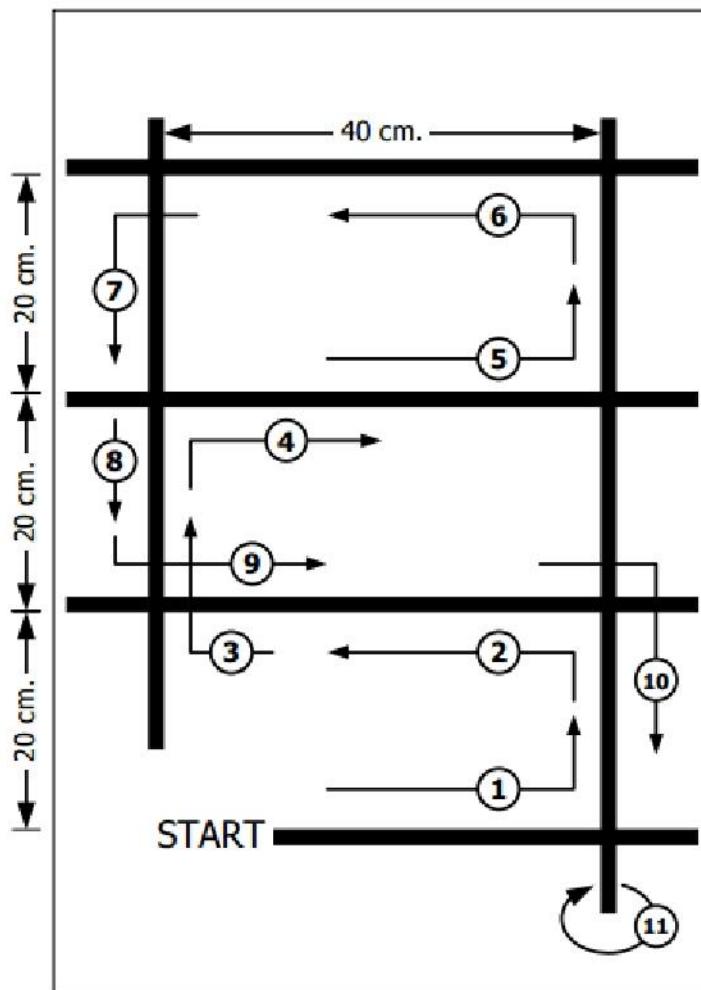
El código fuente de esta función se muestra a continuación.

```
/*Turn right 90 degree function */
void Right90(){
    Forward(150);
    delay(50);
    Spin_Right(200);
    delay(100);
    while(analogRead(6) >Ref) {
        delay(50);
    }
}
```

## Actividad 11: Misión de Multi-cruce de línea

Esta actividad se ha tornado muy popular, se usa en muchos juegos robóticos. Muchos de cruzar la línea se incluyen el campo. El robot debe moverse siguiendo la línea y detectar todos los puntos de cruce o de conexiones y tomar la decisión de girar a la izquierda o a la derecha o hacia adelante o hacia atrás.

Con el código de ejemplo en la actividad de 8 a 10, usted puede combinar todo para que el código complete la solución de la misión de múltiples líneas de cruce. El campo de ejemplo y trayectoria de movimiento se muestra a continuación.



A11.1 abrir el IDE de Arduino y crear el código del listado esbozo A11-1

A11.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A11.3 Desconecte el cable de descarga.

```
*****
 * POP-BOT V1.0
 * Filename : MultiCrossingLineDetect.pde
 * POP-BOT move following the line and check the crossing line to do complex movement
 ****
int Ref=700;
int Left,Right;
int Cnt=0;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
    pinMode(14,OUTPUT);          // PIEZO Speaker
}
void loop(){
    while(Cnt<11){
        Left = analogRead(7);      // Read value from the left ZX-03 sensor
        Right = analogRead(6);     // Read value from the right ZX-03 sensor
        if ((Left<Ref) && (Right>Ref)){ // Found crossing point
            Cross();
        }
        else if ((Left>Ref) && (Right<Ref)){ // Move across the line a short while
            Forward(150);
        }
        else if (Left<Ref){         // Only left sensor detects the black
line
            Spin_Left(150);         // Spin left a short while
        }
        else if (Right<Ref){        // Only right sensor detects the black
line
            Spin_Right(150);        // Spin right a short while
        }
        Forward(200);
        delay(200);
        Spin_Left(200);           // Turn around
        delay(2000);
        Motor_Stop();
        Beep();
        while(1);
    }
    void Cross(){
        Cnt++;
        if (Cnt==11){
            Motor_Stop();
        }
        else if (Cnt==8){          // Check for Forward
            Forward(200);
            delay(300);
        }
        else if(Cnt==3 || Cnt==4 || Cnt==10 ){ // Check for Turn right 90 degree
            Right90();
        }
        else{                      // else Turn Left
            Left90();
        }
    }
}
```

```

/*Turn 90 degree function */
void Right90(){
    Forward(150);
    delay(50);
    Spin_Right(200);
    delay(100);
    while(analogRead(6)>Ref);
    delay(50);
}
/*Turn left 90 degree function
void Left90(){
    Forward(150);
    delay(50);
    Spin_Left(200);
    delay(100);
    while(analogRead(7)>Ref);
    delay(50);
}
void Beep(){
    int i;
    for (i=0;i<600;i++) {
        digitalWrite(14,HIGH);
        delayMicroseconds(150);
        digitalWrite(14,LOW);
        delayMicroseconds(150);
    }
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Motor_Stop(){
    digitalWrite(3,LOW);
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    digitalWrite(9,LOW);
}

```

A11.4 Coloque el POP-BOT en el punto inicial (véase la ilustración de campo). Encienda y observar el movimiento del robot.

POP-BOT se mueve a lo largo de la línea que sigue la ruta de movimiento que se muestra en la ilustración sobre el terreno. POP-BOT hará 3 movimientos cuando se detecta un cruce:

Escenario 1: Se desplaza hacia adelante después de detectar el cruce

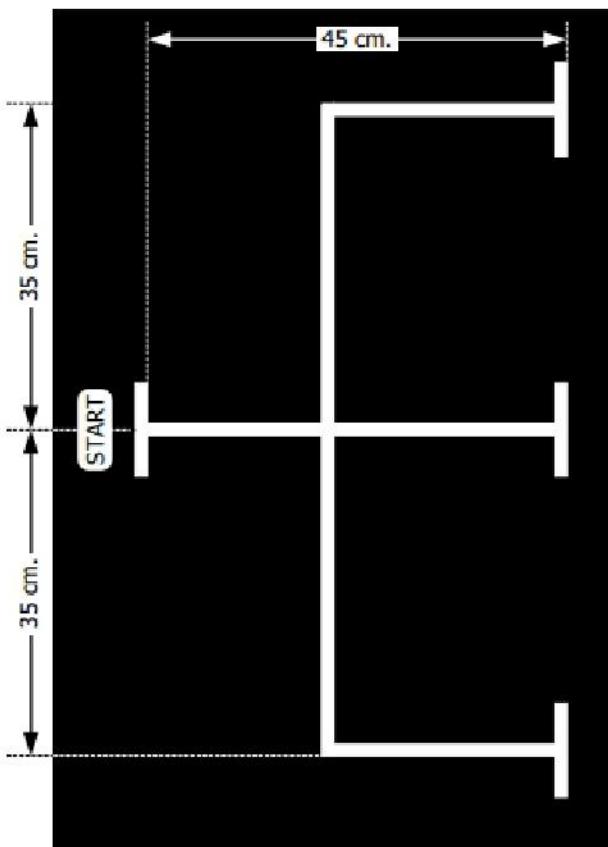
Escenario 2: Gira a la izquierda después de detectar el cruce

Escenario 3: Gira a la derecha después de detectar el cruce de la ruta

Cuando tiene un sólo escenario una posición en la salida 8th de Escenario 2 a 3 posiciones en la 3 rd-4h-10th. POP-BOT hará Escenario 3 para el cruce de descanso. Después de POP-BOT se mueve pasar el último cruce (11), POP-BOT girará en torno a 2 segundos y se detendrá.

### Actividad 12: Desafío con la línea blanca

La diferencia con la Actividad 11 son color de la línea y la ruta de movimiento. Esta actividad cambia el color de la línea en blanco en la superficie de color negro. Vea la ilustración de campo de abajo. La misión es realizar un seguimiento de la línea desde el inicio hasta las 3 líneas de término o destino. Usted puede colocar un globo en cada extremo de la línea. Robot debe perforar el globo. Jugador que puede penetrar todos los globos más rápido es el ganador.



A12.1 abrir el IDE de Arduino y crear el código del listado esbozo A12-1

A12.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A12.3 Desconecte el cable de descarga.

```

int Ref=400;
int Left,Right;
int Cnt=0;

void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
    pinMode(14,OUTPUT);          // PIEZO Speaker
}

void loop(){
    while(Cnt<12){
        Left = analogRead(7);      // Read value from the left ZX-03 sensor
        Right = analogRead(6);     // Read value from the right ZX-03 sensor
        if ((Left<Ref) && (Right>Ref)){ // Detect the black crossing line
            Cross();
        }
        else if ((Left>Ref) && (Right>Ref)){ // Detect the white line
            Forward(150);
        }
        else if (Left<Ref){         // Only the left sensor detects the line
            Spin_Right(150);        // Turn right
        }
        else if (Right<Ref){        // Only the right sensor detects the line
            Spin_Left(150);         // Turn left
        }
    }
    while(1);                      // Endless loop
}
void Cross(){
    Cnt++;
    if (Cnt==12){
        Motor_Stop();
    }
    else if (Cnt==2 || Cnt==10 ){   // Check for turning right 90 deg.
        Right90();
    }
    else if(Cnt==3 || Cnt==6 || Cnt==9 ){ // Check for turning left 180 deg.
        Left180();
    }
    else{                           // else turn left 90 deg.
        Left90();
    }
}

```

---

```

/*Turn right 90 degree function */
void Right90(){
    Forward(150);
    delay(50);
    Spin_Right(200);
    delay(100);
    while(analogRead(6)<Ref);
    delay(50);
}

/*Turn left 90 degree function */
void Left90(){
    Forward(150);
    delay(50);
    Spin_Left(200);
    delay(100);
    while(analogRead(7)<Ref);
    delay(50);
}

/*Turn left 180 degree function */
void Left180(){
    Spin_Left(200);
    delay(300);
    while(analogRead(7)<Ref);
    delay(50);
}

/*Movement function*/
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Motor_Stop(){
    digitalWrite(3,LOW);
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    digitalWrite(9,LOW);
}

```

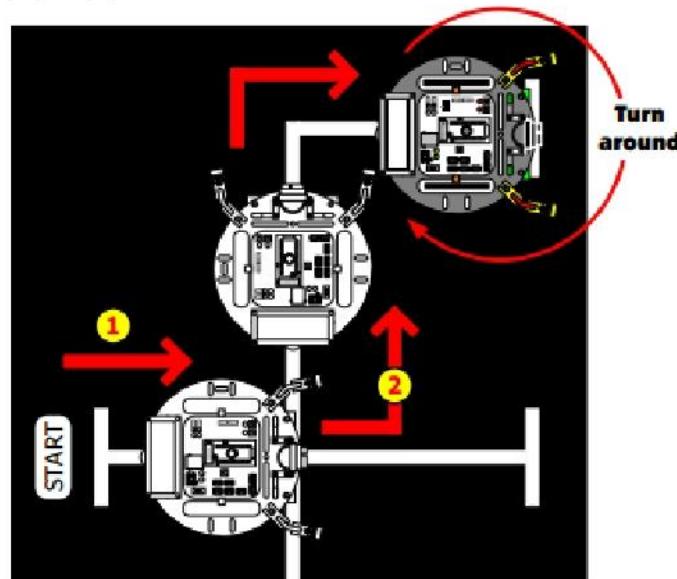
---

**Listado A12-1: Fila WhiteLineDetect.pde**

A12.4 Coloque el POP-BOT en el punto inicial (véase la ilustración de campo). Encienda y observar el movimiento del robot.

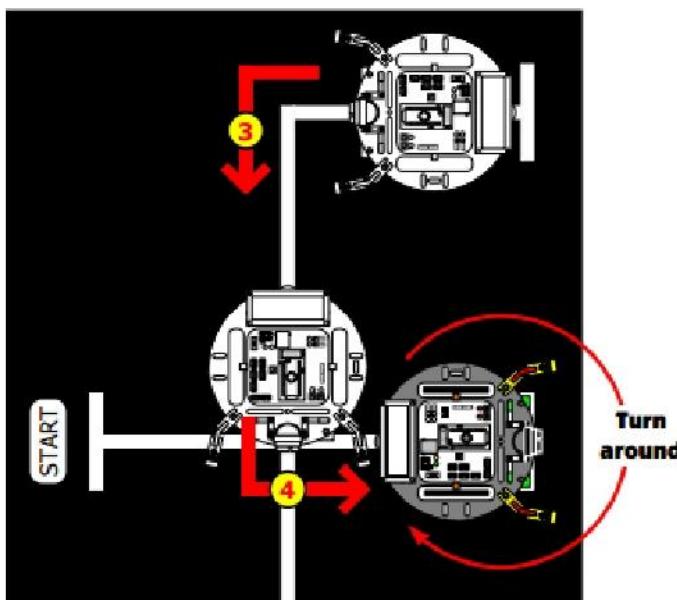
Paso 1: POP-BOT se mueve a lo largo de la línea desde un punto inicial

Paso 2: POP-BOT detecta el primer cruce y gira a la izquierda, se mueve hacia el primer destino en el lado izquierdo del campo.



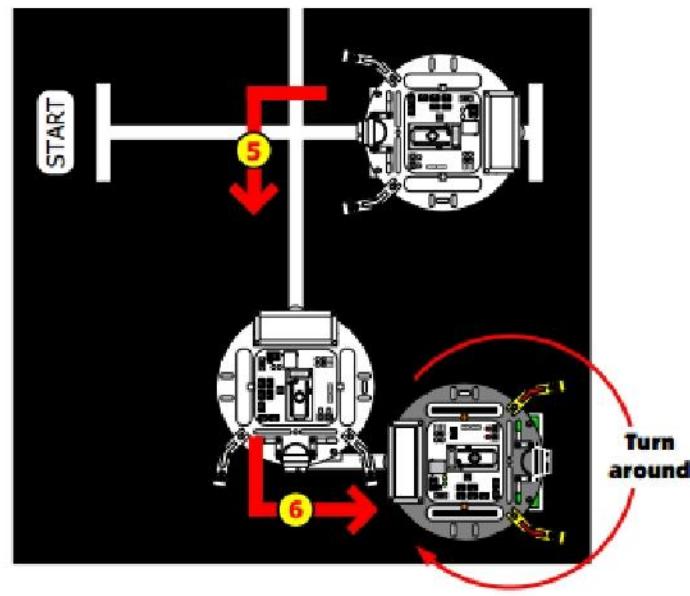
Paso 3: POP-BOT se mueve hacia atrás desde el primer destino y pasa el cruce de nuevo.

Paso 4: POP-BOT detecta el tiempo de unión segundo y gira a la izquierda, se mueve hacia el segundo destino en el centro del campo.



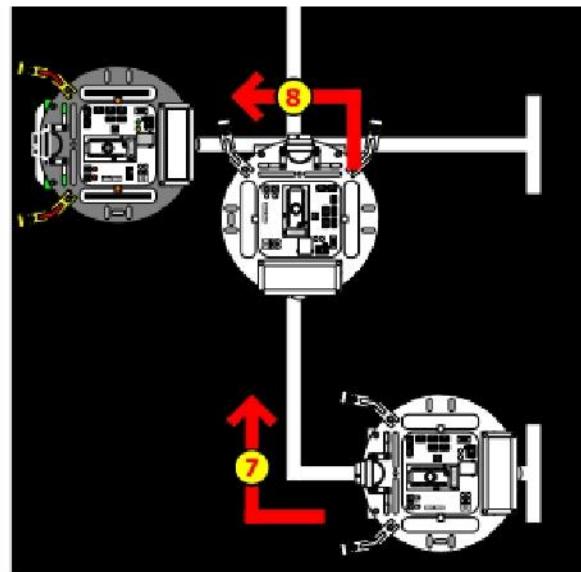
Paso 5: POP-BOT se mueve hacia atrás desde el segundo destino, pasa el cruce de nuevo.

Paso 6: POP-BOT detecta el tiempo de cruce de la tercera y gira a la izquierda, se mueve hacia el destino último en el lado derecho del campo.



Paso 7: POP-BOT se mueve hacia atrás desde el último destino superando el cruce de nuevo.

Paso 8: POP-BOT detecta el tiempo de la unión anterior y se mueve hacia adelante para volver al punto de partida para terminar la misión.



# 8: POP-BOT détection de Bordés

En el capítulo 7, se utilizan los sensores reflectores infrarrojos para detectar las líneas. ¿Sabe usted que estos sensores son capaces de hacer más? En este capítulo se presenta una actividad sobre el uso de sensores infrarrojos para la detección de la superficie, con el reflector se controlará el robot, para que se mueva sobre la mesa y no caer en el borde de la mesa.

Con un simple cambio de la posición de los sensores y un programa simple, usted puede adaptar el POP-BOT para la detección de bordes. Comenzar a ensamblar las piezas mecánicas, colocar los sensores en la posición correcta y crear el boceto Arduino para la prueba de superficie de la mesa.

## 8.1 Lista de piezas adicionales

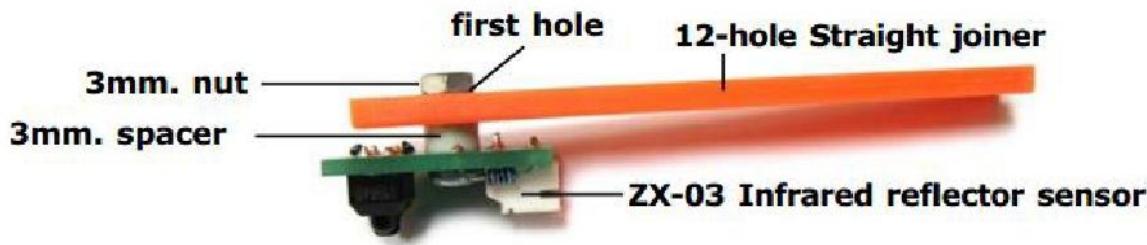


## 8.2 Modificar el procedimiento

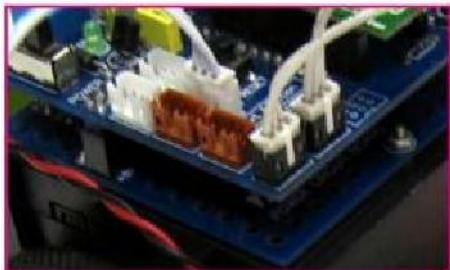
- (1) Eliminar todos los sensores táctiles y sensores de línea de rastreo desde el chasis POP-BOT. Ahora tenemos la forma más simple del robot POP-BOT móvil.



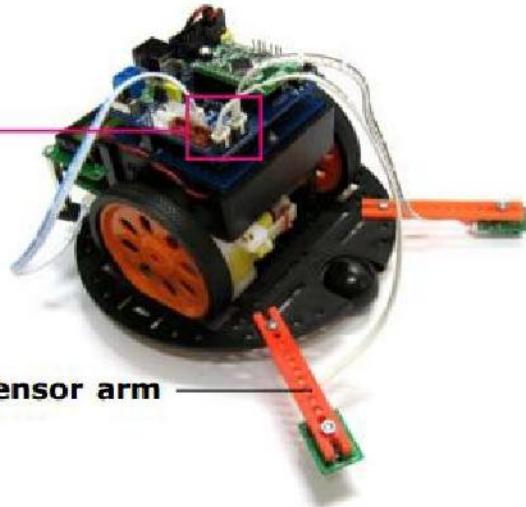
- (2) Monte el sensor reflector de ZX-03 infrarrojo con la recta de 12 hoyos carpintería en el primer agujero mediante el uso del tornillo de 3x10mm y espaciador de 3mm plástico y tuerca de 3mm. La foto muestra la continuación. Haz 2 series de los mismos.



- (3) Fijar ambas estructuras de sensores de la etapa (2) en el lado izquierdo y derecho de la parte delantera del chasis POP-BOT mediante tornillos de 3x10mm y tuercas de 3mm, como la foto de abajo. A continuación, conecte el cable del sensor izquierdo al puerto A7 y el sensor de la derecha al puerto A6. Puede ajustar la posición del brazo del sensor a la condición de la habitación.



**Photo shows the sensor connections on POP-BOT controller board**



**Sensor arm** —————

### Actividad 13: POP-BOT detección de bordes

Esta actividad POP-BOT se mueve sobre la mesa y nunca se cae fuera de la mesa. Mediante el uso de dos de los reflectores de infrarrojos que se fijan en la parte delantera del robot, puede detectar la zona exterior de la tabla. Similar a la línea de código de seguimiento. Si los sensores detectan la superficie, se dará un conjunto de datos superiores. Una vez que los sensores están fuera de la mesa, no hay un rayo infrarrojo reflejado desde la superficie al sensor y por lo tanto el valor de retorno desde el sensor será baja o casi cero.

Puede utilizar este comportamiento para que el código pueda controlar el POP-BOT para moverse sobre la mesa y detectar el borde.

A13.1 Abrir el IDE de Arduino y crear el código del listado esbozo A13-1.

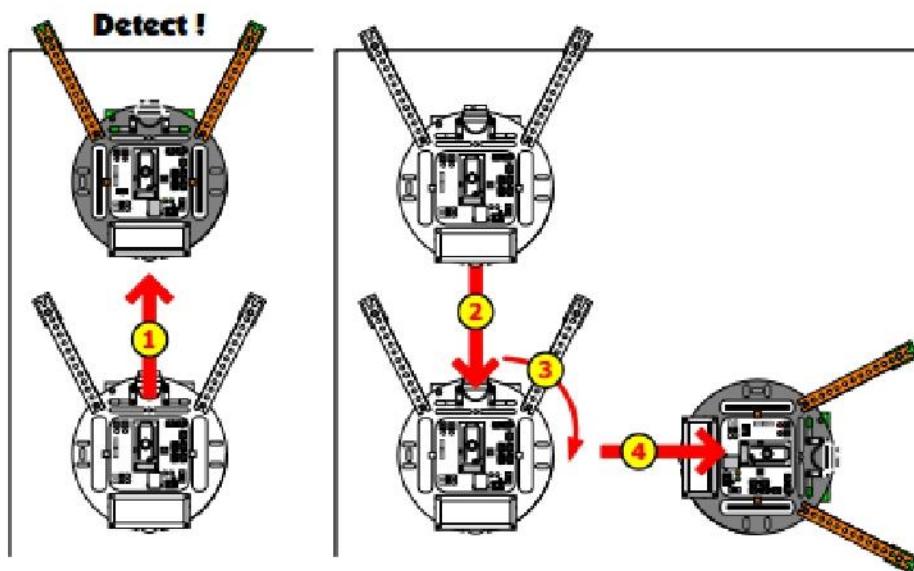
A13.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A13.3 Desconecte el cable de descarga.

A13.4 Coloque el POP-BOT sobre la mesa. Debe eliminar todos los objetos de la mesa. Encienda el POP-BOT y observar el movimiento del Robot.

*POP-BOT se mueve hacia adelante hasta que el sensor llegue al borde de la tabla. POP-BOT va a cambiar la dirección del movimiento siguiendo los siguientes escenarios:*

1. Ambos sensores están fuera del borde de la mesa de: POP-BOT se mueve hacia atrás y girar en ese momento se mueve hacia adelante de nuevo.



---

```

/*
 * POP-BOT V1.0
 * Filename : EdgeDetect.pde
 */
int Ref=300;
int Left,Right;
void setup(){
    pinMode(3,OUTPUT);           // Motor A1
    pinMode(5,OUTPUT);           // Motor A2
    pinMode(6,OUTPUT);           // Motor B2
    pinMode(9,OUTPUT);           // Motor B1
}
void loop(){
    Left = analogRead(7);         // Read value from the left ZX-03 sensor
    Right = analogRead(6);        // Read value from the right ZX-03 sensor
    if (Left>Ref && Right>Ref){ // Both sensors are on the table
        Forward(150);
    }
    else if (Left<Ref && Right<Ref){ // Both sensors out of the table
        Backward(150);
        delay(200);
        Spin_Right(150);
        delay(500);
    }
    else if (Left<Ref) {          // Only the left sensor outs of the table
        Backward(150);
        delay(300);
        Spin_Right(150);
        delay(400);
    }
    else if (Right<Ref){          // Only the right sensor outs of the table
        Backward(150);
        delay(300);
        Spin_Left(150);
        delay(300);
    }
}

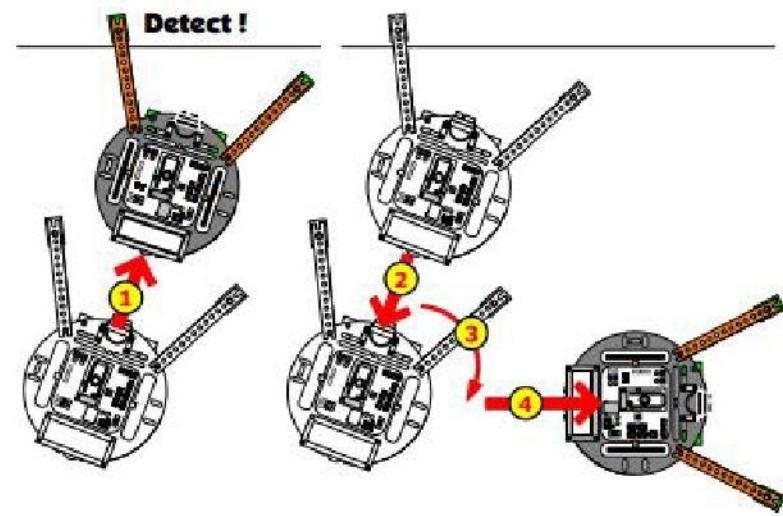
/*Movement function*/
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Spin_Right(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}

```

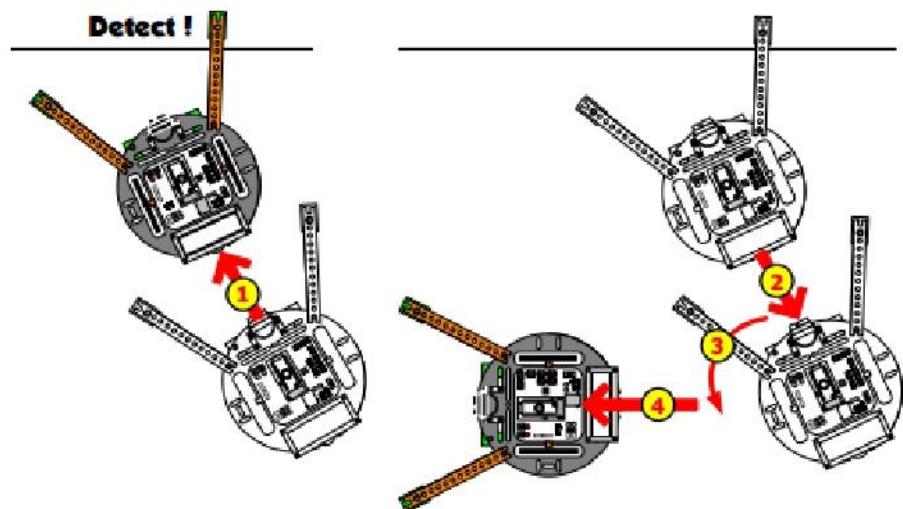
---

**Listing A13-1 : Fila EdgeDetect.pde**

2. El sensor de la izquierda está fuera del borde de la mesa: POP-BOT se mueve hacia atrás y girar en ese momento, se mueve hacia adelante de nuevo.



3. El sensor de la derecha está fuera del borde de la mesa: POP-BOT se mueve hacia atrás y girar luego a la izquierda se mueve hacia adelante de nuevo.



# 9: POP-BOT Evita contacto con objetos

## 9.1 GP2D120: de 4 a 30 cm. Sensor de distancia por infrarrojos

Desde el capítulo 7, tenemos muchos ejemplos acerca de la interconexión de los sensores infrarrojos. Ellos son un tipo de sensor analógico. En este capítulo nos centraremos en Interacción con otros sensores analógicos. Se trata de sensor de distancia por infrarrojos o explorador infrarrojo; GP2D120. Tenemos algunos de ejemplo sobre el uso de este sensor y aplicaciones.

Uno de los sensores especiales en la robótica es el sensor de distancia por infrarrojos. Algunas personas lo llaman el Buscador o Ranger IR. El módulo de GP2D120, le da al POP-BOT la capacidad para la medición de la distancia y la detección de obstáculos mediante una luz infrarroja. Su POP-BOT puede evitar los obstáculos sin tener que hacer ningún contacto físico.

### 9.1.1 GP2D120 características

- Utiliza la reflexión de luz infrarroja para medir la distancia.
- Se puede medir un rango de 4 a 30 cm.
- 4,5 a 5V de alimentación y 33mA corriente eléctrica.
- El rango de tensión de salida es de 0,4 a 2,4V; como fuente, usa +5V.

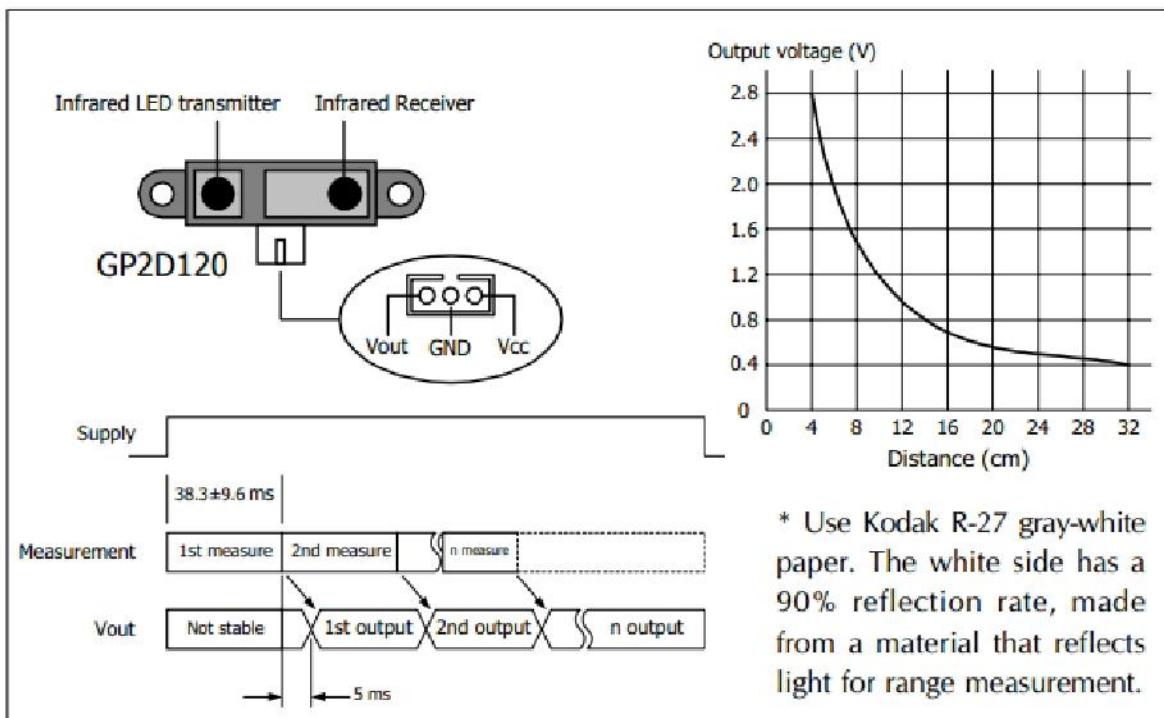


Figura 9-1: GP2D120, asignación de Pin's, operación y características



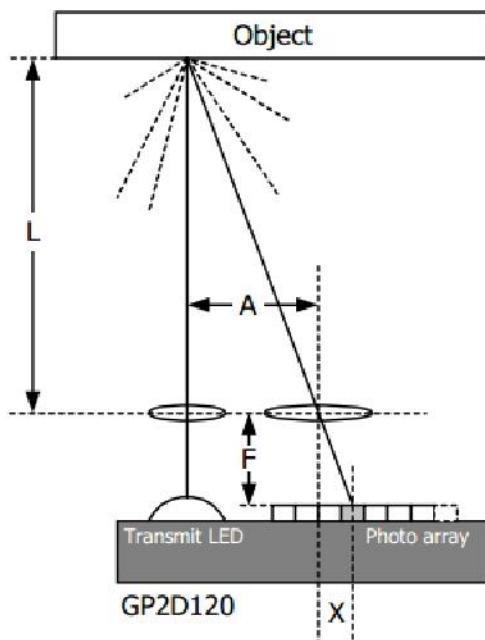
GP2D120 es el módulo de infrarrojos de seguimiento o Ranger, tiene 3 terminales: la entrada de alimentación (Vcc), Tierra (GND) y Voltaje de salida (Vout). Para leer los valores de Voltaje de la GP2D120, usted debe esperar hasta después del período de reconocimiento que es de alrededor de 32 a 52,9 ms.

La tensión de salida de GP2D120 tiene un rango de 30 cm. y +5V de alimentación y entre 0,25 a 0.55V, con los medios de 0.4V. En el rango de 4 cm., La tensión de salida va a cambiar en  $2.25V \pm 0.3V$ .

### 9.1.2 Cómo funciona el módulo IR guardabosques trabaja

Rango de medición se puede hacer de muchas maneras. La forma más fácil de entender es a través de ultrasónico donde las ondas de sonido se envían al objeto y el tiempo que tarda para reflejar de vuelta se mide. Esto es porque las ondas de sonidos no viajan rápido, y puede ser medida por el equipo de nuestros días. Sin embargo, en el caso de la luz infrarroja, el tiempo que tarda en golpear un obstáculo y reflejan de nuevo no se puede medir porque la luz infrarroja viaja rápido. Ningún equipo de medición está disponible todavía. Por lo tanto, la siguiente teoría debe ser utilizada.

La luz infrarroja se envía desde un transmisor al objeto en el frente, pasando a través de una lente de condensación de modo que la intensidad de la luz se enfoca en un punto determinado. La refracción se produce una vez que la luz llega a la superficie del objeto. Parte de la luz refractada será enviado de nuevo al lado del receptor, donde se combinan estas luces y detecta el punto de impacto. La luz pasa a una serie de foto-transistores. Según la posición en la que la luz cae, se puede utilizar para calcular la distancia (L) desde el transmisor hasta el obstáculo utilizando la siguiente fórmula:



$$\frac{L}{A} = \frac{F}{X}$$

Por lo tanto, L es igual

$$L = \frac{F \times A}{X}$$

Así, el valor de la distancia desde los fototransistores se enviará al módulo de señal, evaluando antes de que se cambie la tensión, lo que resulta en un cambio de tensión de acuerdo con la distancia medida.

### 9.1.3 Lectura GP2D120 con convertidor A/D

El voltaje de salida GP2D120 cambiará de acuerdo con la distancia de detección. Por ejemplo, 0.5V Vout es igual a 26cm. de distancia y 2V Vout es de 6cm de distancia. La tabla 9-1 muestra el resumen de Vout GP2D120 y relación de distancia.

Para la conexión con un módulo convertidor A/D dentro de un microcontrolador, el resultado es de datos es en bruto de la conversión A/D. El usuario tendrá que utilizar el software para convertir los datos en bruto a la distancia exacta. Se puede calcular la distancia aproximada de la siguiente fórmula.

$$R = \frac{2914}{V + 5} - 1$$

Así, R como Distancia, V Centímetro unidad como datos digitales de la conversión A/D

Por ejemplo, ver la Tabla 9-1. Los datos en bruto de la conversión es de 307. Se trata de 8 cm de distancia.

### **Advertencia para el cable de señal de la GP2D120**

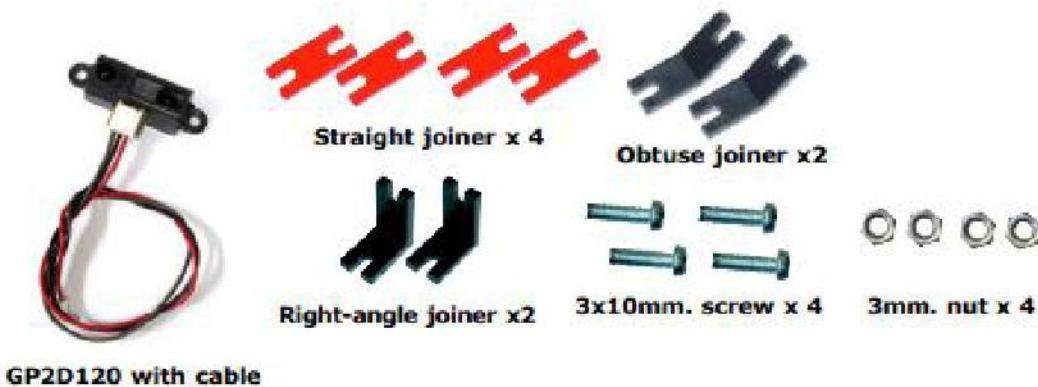
*El módulo tiene una GP2D120 disposición de las patillas diferente a la de la tarjeta del controlador POP-BOT, a pesar de que tiene una apariencia similar. Por lo tanto, un cable de señal especial ya se ha conectado al módulo GP2D120. el usuario sólo tiene que conectar el otro extremo del cable a los puntos de conexión de la placa controladora POP-BOT. NO quite el cable del módulo, y no sustituirlo por los cables de señal a partir de módulos de sensores de otros.*

<b>GP2D120 output voltage (V)</b>	<b>10-bit A/D converter result</b>	<b>Distance (cm.)</b>
0.4	82	32
0.5	102	26
0.6	123	22
0.7	143	19
0.8	164	16
0.9	184	14
1.0	205	13
1.1	225	12
1.2	246	11
1.3	266	10
1.4	287	9
1.5	307	8
1.6	328	8
1.7	348	7
1.8	369	7
1.9	389	6
2.0	410	6
2.1	430	6
2.2	451	5
2.3	471	5
2.4	492	5
2.5	512	5
2.6	532	4

**Tabla 9-1: La relación de tensión de salida del GP2D120, resultado del convertidor de AJD y la distancia medida**

## 9.2 POP-BOT modificación para GP2D120

### 9.2.1 Lista de piezas adicionales



### 9.2.2 Modificar el procedimiento

- (1) Eliminar todos los sensores del chasis POP-BOT. Ahora tenemos la forma más simple del POP-BOT.



- (2) Coloque 2 piezas rectas en el agujero del módulo GP2D120 mediante el uso de tornillos de 3x10mm y tuercas de 3 mm. A continuación, conecte las piezas obtusas al final de cada pieza recta, después las piezas en ángulo.



(3) Fijar estructura GP2D120 desde el paso (2) en la parte delantera del chasis POP-BOT mediante el uso de tornillos de 3x10mm y tuercas de 3 mm en la posición indicada en la foto de abajo. Conecte el cable de GP2D120 19/SCL/A5 de la tarjeta controladora de POP-BOT. Ahora, el POP-BOT con IR está listo para programarse.



### 9.3 Cómo leer los datos de GP2D120 de POP-BOT

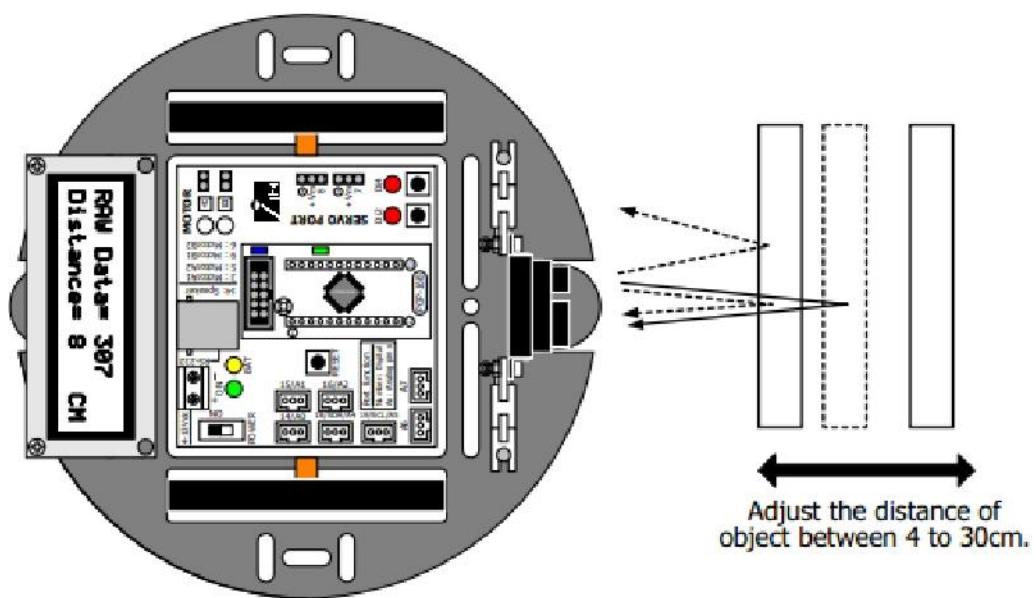
POP-BOT tiene el POP-168, módulo microcontrolador. Funciona con el software de Arduino. Arduino tiene una función especial para leer el valor del puerto analógico. Es la función **analogRead()**. El valor entre corchetes es el número de entrada analógica (0 a 7). Para POP-BOT proporciona sólo 3 a 7. El analogRead() es devolver los datos enteros desde 0 a 1023. Es de 10-bit Un resultado convertidor A/D.

Usted puede calcular los datos en bruto en la unidad de voltios (V) a raíz de esta fórmula

$$\text{volt} = \text{Los datos brutos} \times 5/1023$$

## Actividad 14: Lectura de los datos GP2DI20

- A14.1 Abrir el IDE de Arduino y crear el código del listado esbozo A14-1.
- A14.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.
- A14.3 Desconecte el cable de carga.
- A14.4 Coloque el POP-BOT sobre la mesa. Coloque un objeto frente a la GP2D120. Encienda el POP-BOT. Trate de mover un objeto dentro y fuera de la GP2D120 sensor. Observe el resultado en la pantalla SLCD.



---

```

/*
 * POP-BOT V1.0
 * Filename : GP2D120withSLCD.pde
 * Show GP2D120 data on SLCD16x2 module
 */
#include <SoftwareSerial.h>
#define rxPin 16           // SLCD pin
#define txPin 16           // Same pin
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int gp2;
float distance;
void setup(){
  pinMode(txPin,OUTPUT);
  MySerial.begin(9600);
  delay(1000);
}
void LCD_CMD(int Command){
  MySerial.print(0xFE,BYTE);           // Command
  MySerial.print(Command,BYTE);
}
void loop(){
  gp2=analogRead(5);
  distance=(2914/(gp2+5))-1;          // Convert to Centimetre unit
  LCD_CMD(0x80);                     // Select LCD first Line
  MySerial.print("RAW Data=      ");   // Show raw data
  LCD_CMD(0x8A);
  MySerial.print(gp2,DEC);

  LCD_CMD(0xC0);                     // Select LCD second Line
  MySerial.print("Distance=      ");   // Show distance in Centimetre unit
  LCD_CMD(0xCA);
  MySerial.print(distance,DEC);
  LCD_CMD(0xCE);
  MySerial.print("CM");
  delay(200);
}

```

### Programa descripción

- (1) la comunicación inicial de los datos en serie. Establecer el pin 16/A2 al puerto serie.
- (2) Bucle para leer la señal analógica en el puerto An5 de la tarjeta de conexión POP-BOT y ver en la pantalla SLCD.
- (3) Convertir los datos en bruto a los datos de distancia de un centímetro en la unidad mediante el uso de este formulario con  $cm = (2914 / (gp2 + 5)) - 1$
- (4) Convertir el resultado a formato ASCII y enviarlo al módulo SLCD16x2 para mostrar.
- (5) Bucle para obtener los datos de GP2D120 cada 0,2 segundos.

**Listado A14-1: Archivo GP2D120\_LCD.pde; archivo grabado en Arduino para la lectura con GP2D120 al POP-BOT**

## Actividad 15: Evitar objetos sin contacto

Con el módulo GP2D120, agrega la medición de distancia y detección de obstáculos utilizando la función de luz infrarroja para su robot. Su POP-BOT puede evitar los obstáculos sin tener que hacer ningún contacto físico.

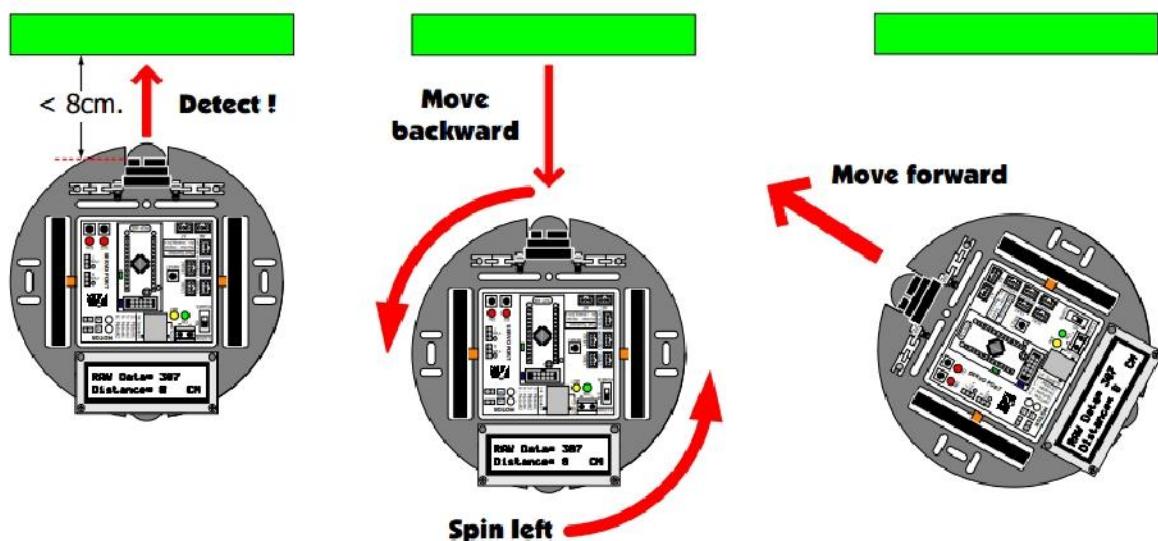
A15.1 abrir el IDE de Arduino y crear el código del listado esbozo A15-1.

A15.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para al robot.

A15.3 Desconecte el cable de descarga.

A15.4 Coloque el POP-BOT. Trate de colocar cualquier objeto en la parte delantera del robot y ver su funcionamiento.

*El robot compruebe la distancia del objeto con un rango de 8cm. Si no existe ningún obstáculo, el robot se moverá hacia adelante continuamente. Si encuentra el objeto, éste se moverá hacia atrás, girará a la izquierda y avanzará de nuevo.*



---

```

/*
 * POP-BOT V1.0
 * Filename : TouchlessObjectRobot.pde
 */
int gp2;
void setup(){
    pinMode(3,OUTPUT); // sets the digital pin as output
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
}
void Forward(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(6,speed);
    digitalWrite(9,LOW);
}
void Backward(int speed){
    analogWrite(5,speed);
    digitalWrite(3,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void Spin_Left(int speed){
    analogWrite(3,speed);
    digitalWrite(5,LOW);
    analogWrite(9,speed);
    digitalWrite(6,LOW);
}
void loop(){
    int i;
    for (i=0;i<5;i++){ // Loop 5 times for noise filter
        gp2=(gp2+analogRead(5));
    }
    gp2=gp2/5;
    if (gp2>290){ // Found object
        Backward(200); // Move backward to change direction
        delay(300);
        Spin_Left(200); delay(350); // Change direction
    }
    else{
        Forward(200); // Moving forward
    }
}

```

### Descripción del programa

- (1) Con el arranque, POP-BOT hace un pitido. Puede utilizar esta señal para comprobar el estado de RESET del robot cuando su batería está baja. Si el pitido del robot, durante el movimiento, bajo o no suena, significa que la batería del robot es baja.
- (2) Lectura del valor de la GP2D120 para almacenar en la GP2 variables 5 veces. Calcula para obtener el promedio valor para la protección del error al leer desde el movimiento.
- (3) Verificar si el valor de la GP2 es más de 290 o no? Si es así, significa que ahora el robot está a menos de 8 cm. (aproximación). Programa controlar el robot se mueva hacia atrás 0,25 segundos y el giro a la izquierda 0,5 segundos para cambiar la dirección para evitar el objeto.
- (4) Sien la GP2 el valor es menor que 290, el robot sigue mueviendoce hacia adelante.

### Listado A15-1: Archivo Robot\_Survey.pde

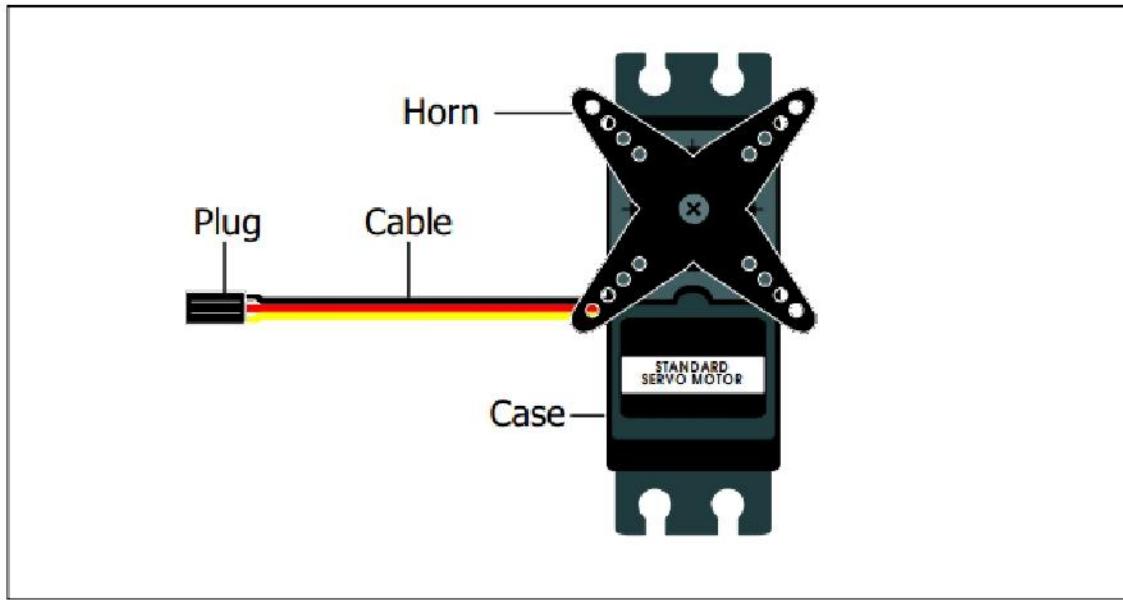
# 10 : POP-BOT actividad del SERVOMOTOR

POP-BOT características del servo motor. POP-BOT puede controlar a dos de los pequeños motores servo simultáneamente. El usuario no requiere baterías adicionales para el motor servo. Se trata de las características importantes de POP-BOT. POP-BOT puede manejar 4 motores, dos de los motores de corriente continua y 2 de los servomotores.

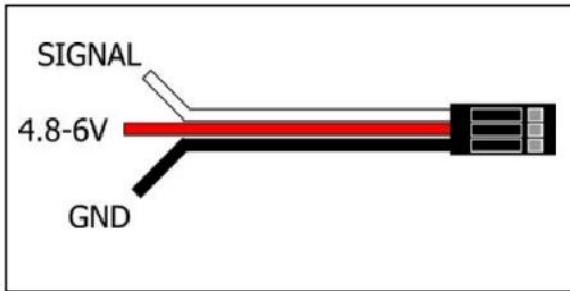
## 10.1 Introducción del Servomotor

La Figura 10-1 muestra un dibujo de un servo estándar. El enchufe se utiliza para conectar el servo motor a una fuente de alimentación (Vdd y Vss) y una fuente de señal (una microcontrolador pin I/O). El cable lleva los Vdd, Vss y la línea de señal desde el conector en el servomotor. El cuerno (Horn) es la parte del servo que se parece a una estrella de cuatro puntas. Cuando el servo está en marcha, el cuerno es la parte que se mueve y controla el microcontrolador. La caja contiene los circuitos de control Servos, un motor de corriente continua y engranajes. Estas partes trabajan juntas para trabajar con altas / bajas señales del microcontrolador y los convierten en posiciones mantenidas por el brazo del servo.

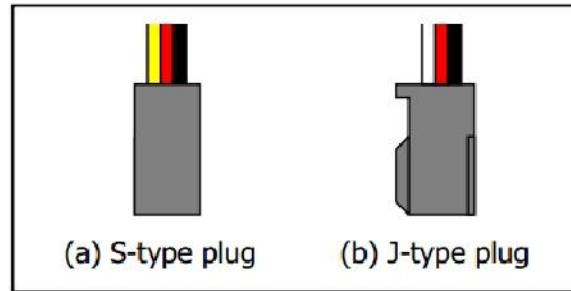
Figura 10-2 muestra la asignación del cable del servomotor. Cuenta con 3 cables con diferencia de color, el negro para GND o el polo Vss o negativo; rojo para la tensión de alimentación del motor Vdd o servo y amarillo (a veces es de color marrón) cable de señal.



**Figura 10\_1: Vista del Servomotor**



**Figura 10-2: Cable del Servomotor**



**Figura 10-3: Tipos enchufes Servomotor**

El servomotor cuenta con 2 tipos de enchufe, S y J, se muestra en la figura 10-3. Los servos se controlan por medio de pulsos. El impulso positivo da longitud de 1 a 2ms, que se repiten de 50 a 60 veces por segundo. Puedes consultar los detalles en la figura 4-10. Comience con un pulso de 20 milisegundos y ajusta el ancho de pulso positivo a 1 milisegundo. El cuerno del servomotor se mueve a la izquierda. El ancho de pulso de 1,5 milisegundos mueve el brazo del servo hacia el centro. El ancho de pulso de 2 milisegundos mueve el servo a la derecha.

Las especificaciones importantes del servomotor son 2, velocidad de giro del servo o el tiempo de tránsito y de par. La velocidad de giro de servo o tiempo de tránsito, se utiliza para determinar la velocidad de rotación del servo. Esta es la cantidad de tiempo que tarda el servo para moverse 60 grados. Por ejemplo, suponga que tiene un servo con un tiempo de tránsito de 0,17sec / 60 grados sin carga. Esto significa que se necesitarían cerca de la mitad de un segundo para girar una totalidad de 180 grados. Más si la servo está bajo una carga. Esta información es muy importante si la respuesta del servo necesita una alta velocidad.

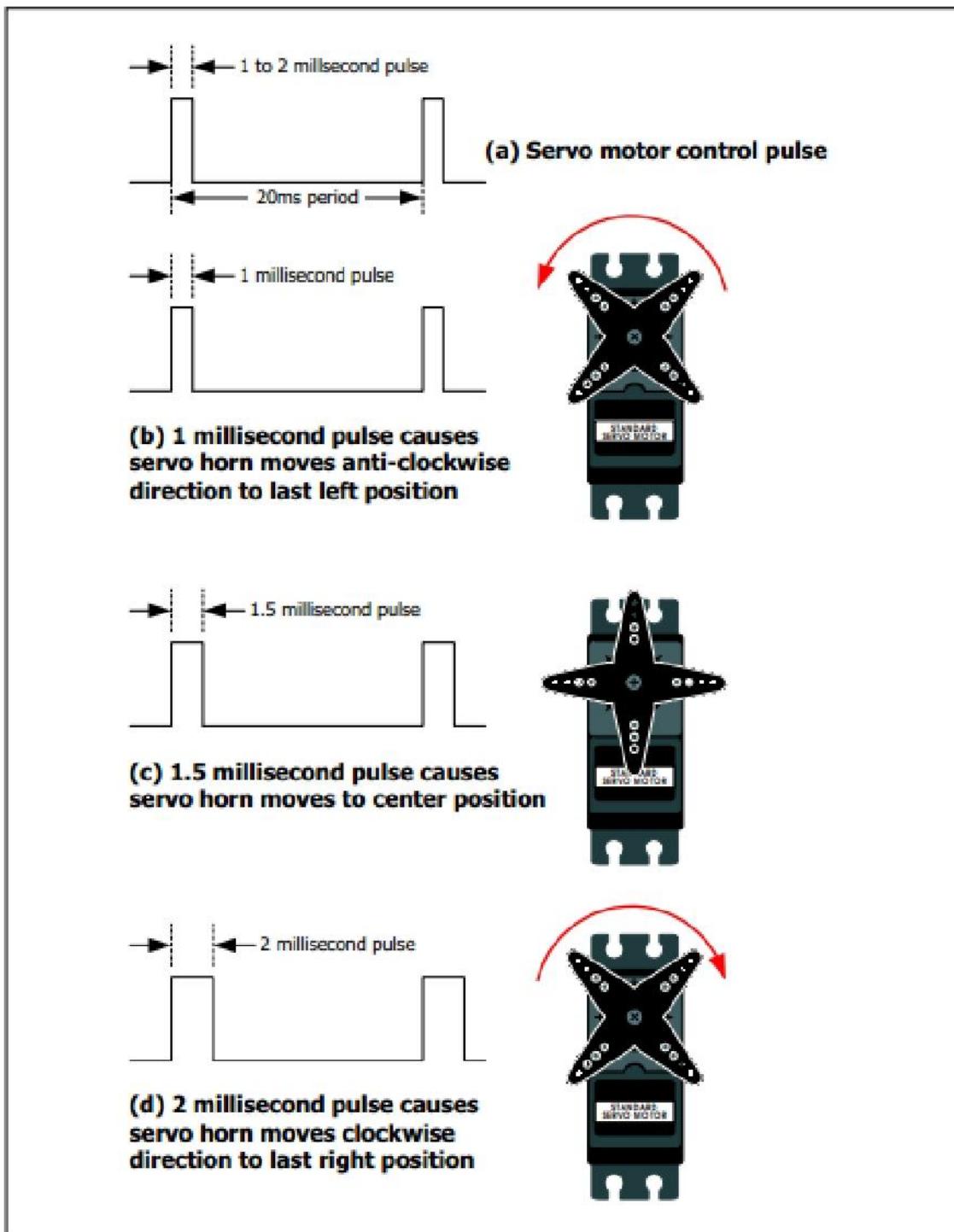


Figura 10-4 Diagrama de tiempos del Servomotor

## 10.2 Control del servomotor con Arduino

La biblioteca Arduino POP-168 para el control del servomotor es la **SoftwareServo**.

Debido a que el hardware de POP-BOT no utiliza el pin PWM para hacer la salida del servo motor, utilizamos el puerto de propósito general a la salida del servo; ED7 y ED8.

La biblioteca SoftwareServo puede manejar servos en todos los pasadores de forma simultánea. La API sigue el modelo de la biblioteca del servo wiring.org pero el código es diferente. Usted no está limitado a 8 servos, pero debe llamar a la **SoftwareServo :: refresh()** al menos una vez cada 50ms o menos para mantener a sus servos actualizados.

### 10.2.1 Método Estándar

#### **attach(int)**

Gire el controlador de servo. Llama pinMode. Devuelve 0 en caso de fallo.

#### **detach()**

Suelte un alfiler de la conducción del servo.

#### **write(int)**

Ajuste el ángulo del servo en grados, de 0 a 180.

#### **read()**

Devolver ese valor ajustado con la última escritura () .

#### **attached()**

Devuelve 1 si el servo está conectado actualmente.

### 10.2.2 Métodos adicionales

#### **refrech()**

Usted debe llamar a esto por lo menos una vez cada 50 ms para mantener actualizados los servos. Se le puede llamar tantas veces como quieras, no va a disparar más de una vez cada 20 ms. Cuando lo hace se llevará a partir de 0.5 a 2.5 milisegundos para completar, pero no hará deshabilitar las interrupciones.

#### **setMinimumPulse (uint16\_t)**

Ajustar la duración del pulso a 0 grados en microsegundos. (Valor mínimo predeterminado es de 544 microsegundos)

#### **setMaximumPulse (uint16\_t)**

Ajustar la duración del pulso de 180 grados en microsegundos. (Máximo valor por defecto es 2400 pulsos por microsegundo)

\* <Http://www.arduino.cc/playground/ComponentLib/Servo>

#### **10.2.3 Para tomar en cuenta**

A pesar de que se conecte un servo, no recibirá ninguna señal de control hasta que usted envíe su primera posición con el método **write()**, esto es para evitar que salte a un valor arbitrario impar.

#### **10.2.4 Tamaño**

La biblioteca tiene alrededor de 850 bytes de flash y 6+(8 x servos) bytes de SRAM.

#### **10.2.5 Limitaciones**

Esta biblioteca no impide interrupciones, por lo que **millis()** guardará el trabajo y usted no perderá entradas de datos en serie, pero, un fin de pulso puede ser de longitud máxima de interrupción, puede causar un pequeño fallo en la posición del servo . Si usted tiene un gran número de servos habrá una ligera distorsión (1 a 3 grados) de la posición de los que tienen los más bajos valores angulares.

#### **10.2.6 Un ejemplo**

```

#include <SoftwareServo.h>
SoftwareServo myservo; // create servo object to control a servo
int potpin = 2; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin
void setup() {
    myservo.attach(7); // attaches the servo on pin 7 to the servo object
}

void loop() {
    val = analogRead(potpin);
    // reads the value of the potentiometer (value between 0 and 1023)
    val = map(val, 0, 1023, 0, 179);
    // scale it to use it with the servo (value between 0 and 180)
    myservo.write(val);
    // sets the servo position according to the scaled value
    delay(15); // waits for the servo to get there

    SoftwareServo::refresh();
}

```

### Actividad 16: POP-BOT controles del servomotor

Esta actividad demuestra el ejemplo sencillo sobre el control del servomotor estándar con la tarjeta de conexión POP-BOT.

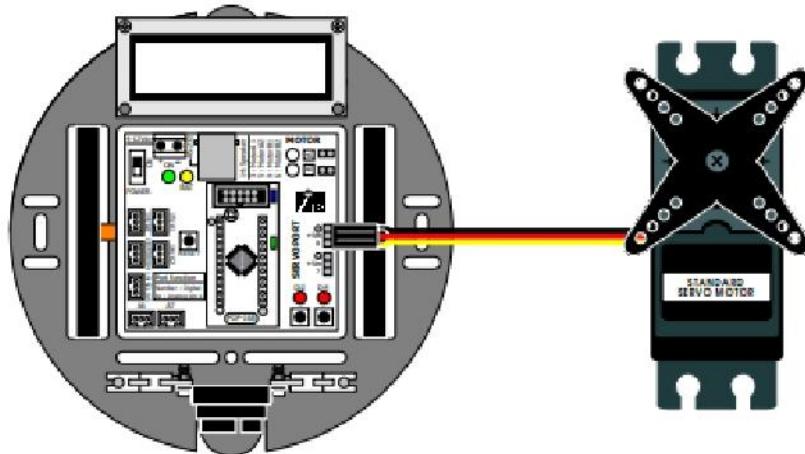
#### Actividad 16-1 simple control servomotor

A16.1 abrir el IDE de Arduino y crear el código del listado esbozo A16-1.

A16.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A16.3 Desconecte el cable de descarga.

A16.4 Conecte el motor estándar RC servo PUERTO SERVO 7 u 8 de la tarjeta controladora de POP-BOT.



A16.5 Encienda el POP-BOT. Vea su funcionamiento.

*Tras el encendido, el servomotor es impulsado por POP-BOT. El movimiento lo hace desde la izquierda a la última posición de la derecha y la última posición de la izquierda de nuevo.*

---

```
*****  
* POP-BOT V1.0  
* Filename : SimpleServo.pde  
* Simple servo motor controlling  
*****  
int i;  
  
void setup(){  
//--- Servo Motor ---//  
    pinMode(8,OUTPUT);           // Servo Motor  
    pinMode(7,OUTPUT);           // Servo Motor  
}  
void loop(){  
    for (i=0;i<100;i++){  
        digitalWrite(7, HIGH);      // Set Servo Di7  
        digitalWrite(8, HIGH);      // Set Servo Di8  
        delayMicroseconds(500);     // Positive Delay  
        digitalWrite(7,LOW);  
        digitalWrite(8,LOW);  
        delay(20);                 // Negative delay  
    }  
    for (i=0;i<100;i++){  
        digitalWrite(7, HIGH);      // Set Servo Di7  
        digitalWrite(8, HIGH);      // Set Servo Di8  
        delayMicroseconds(2300);    // Positive delay  
        digitalWrite(7,LOW);  
        digitalWrite(8,LOW);  
        delay(20);                 // Negative delay  
    }  
}
```

---

**Listado A16-1 Archivo SimpleServo.pde, el archivo de boceto para la demostración Arduino POP-BOT de control de servomotores**

## Actividad 16-2: POP-BOT botón de control del servomotor

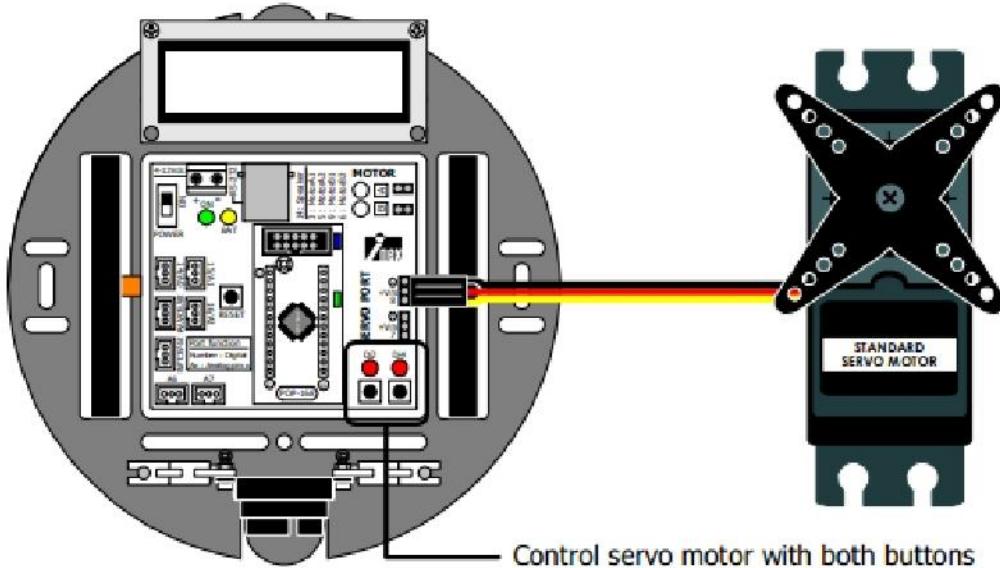
Esta actividad añadir más códigos para mejorar el servomotor de control por botones en placa controladora POP-BOT.

A16.6 abrir el IDE de Arduino y crear el código del listado esbozo A16-2.

A16.7 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.

A16.8 Desconecte el cable de descarga.

A16.9 Conecte el motor estándar RC servo PUERTO SERVO 7 u 8 de la tarjeta controladora de POP-BOT.



A16.10 Encienda el POP-BOT. Pulse el botón en Di2 y Di4 y ver el funcionamiento del motor servo.

Di2 botón se utiliza para controlar el motor servo mueve a la posición derecha último.

Di4 botón se utiliza para controlar el motor servo mueve a la posición izquierda pasado.

*Cuando se mueve el brazo del servo a la conclusión última, POP-BOT sonará un sonido para informar al usuario saber la posición final.*

*Usted puede presionar y mantener o presione y suelte para controlar la posición del servo.*

---

```

/*
 * POP-BOT V1.0
 * Filename : SwitchControlServo.pde
 * Control a servo motor with 2 button switches at D12 and D14. Show on SLCD
 */
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int Old_i,i=1500,j=0,k;

void setup(){
    pinMode(8,OUTPUT);           // Servo Motor
    pinMode(7,OUTPUT);           // Servo Motor
    pinMode(14,OUTPUT);          // PIRZO Speaker
    pinMode(2,INPUT);            // Left Switch
    pinMode(4,INPUT);            // Right Switch
    pinMode(txPin,OUTPUT);       // SLCD pin
    MySerial.begin(9600);        // Communicate With SLCD
    delay(1000);
    i=1500;                      // Centre value for Servo Motor
}
void loop(){
    if(digitalRead(2)==0){       // Press an Increment switch
        if(i<2500){              // Check the maximum value
            i+=20;                // If less than, increase the variable value
        }
        else {Beep();}
    }
    if(digitalRead(4)==0){       // Press a Decrement switch
        if(i>400){               // Check thre minimum value
            i-=20;                // If more than, decrease the variable value
        }
        else {Beep();}
    }
    if (i!=Old_i){
        MySerial.print(0xFE,BYTE); // Clear Screen
        MySerial.print(0x01,BYTE);
        MySerial.print(i,DEC);     // Show position on LCD
        Old_i =i;
    }
    digitalWrite(7, HIGH);        // Set Servo D17
    digitalWrite(8, HIGH);        // Set Servo D18
    delayMicroseconds(i);         // Positive pulse delay
    digitalWrite(7,LOW);
    digitalWrite(8,LOW);
    delay(20);                  // Negative pulse delay
}
void Beep(){
    int i;
    for (i=0;i<600;i++){
        digitalWrite(14,HIGH);
        delayMicroseconds(150);
        digitalWrite(14,LOW);
        delayMicroseconds(150);
    }
}

```

---

### Listado A16-2 Archivo SwitchControlServo.pde; para Arduino POP-BOT servocontrol, demostración de los botones

## **Programación de la operación**

Propósito de esta actividad es mostrar la posición del servomotor que controla pulsando los botones. Tenemos que añadir 2 botones para cambiar la posición del motor servo y enviar el valor de posición para mostrar en la pantalla LCD de serie del POP-BOT. Puede utilizar este valor de referencia para el control de servo motor.

El código se activa el botón de presionar tanto Di2 y el puerto Di4. Si el botón se pulsa en el Di2, se incrementará el valor a cada 20. Si el botón se pulsa en el Di4, variara disminuyendo el valor de cada 20. El valor de la variable se utiliza para definir la anchura del impulso de control de servomotor.

Cuando se cambia el valor a la última posición final (máximo y mínimo), la función de sonido se ejecutará para conducir una señal acústica para informar desarrollador conocer la operación.

# 11: POP-BOT Capacidad de buscar objetos

Desde el capítulo 10, nos enteramos acerca de cómo controlar el servomotor con nuestro POP-BOT. El factor importante es la biblioteca SoftwareServo. En este capítulo se centrará sobre el control del servomotor y la aplicación de lectura del sensor. El sensor que se utiliza en este capítulo es GP2D120. Vamos a modificar POP-BOT a la búsqueda del objeto robot móvil.

## 11.1 POP-BOT modificación a la búsqueda del objeto robot móvil.

### 11.1.1 Lista de piezas adicionales

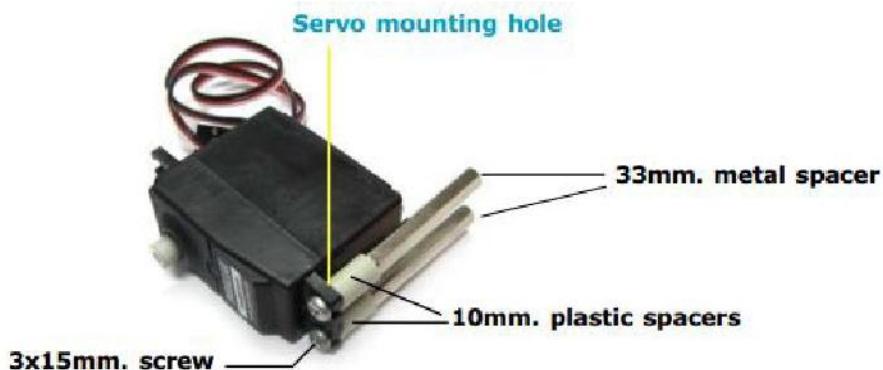


### 11.1.2 Modificar el procedimiento

- (1) Eliminar todos los sensores de chasis POP-BOT. Ahora tenemos la forma más simple del robot POP-BOT móvil.



- (2) Retire el brazo del servo. Coloque 2 espaciadores metálicos de 33mm y separadores de plástico de 10 mm, con dos agujeros de montaje del servomotor mediante el uso de tornillos 3x15mm; como en la foto que aparece a continuación.



- (3) Monte el servomotor que se adjunta separadores de paso (2) con el chasis del robot en la parte frontal mediante tornillos de 3x10mm. Ajuste los tornillos en la parte inferior.



(4) Coloque 2 piezas de carpintería recto con el agujero de la GP2D120 módulo mediante el uso de tornillos de 3x10mm y tuercas de 3mm.



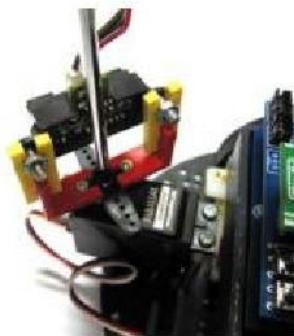
(5) Coloque 2 piezas de ángulo recto con el brazo del servo en la posición del agujero interior mediante el uso de tornillos autorroscantes de 2mm.



(6) Conecte la estructura GP2D120 desde el paso (4) al final con la pieza de ángulo recto.



(7) Conecte el GP2D120 y la estructura de servo del paso (6) con el eje del servo.  
Apretar con un tornillo de servo.



(8) Conecte el enchufe del servomotor servo en el puerto del, puerto de salida 7. Asegúrese de que el enchufe del motor servo esté conectado correctamente. Por último, conecte GP2D120 de 19/SCL/A5 puerto de POP-BOT.

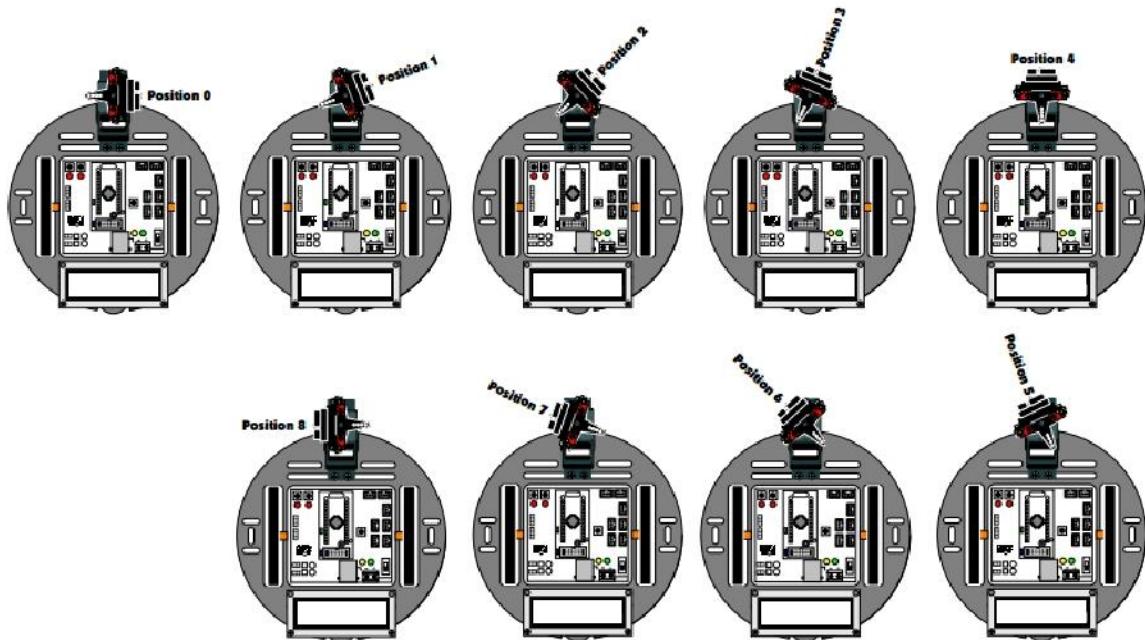


(9) Ahora bien, el POP-BOT esta modificado y listo para ser programado.



## Actividad 17: POP-BOT objeto la búsqueda de

Esta actividad demuestra cómo buscar un objeto por mover el servomotor. El POP-BOT que se adjunta al GP2D120 con servo cuerno de motor se moverá el servo y compruebe la distancia entre el sensor y el objeto de destino. Hay 9 pasos en el movimiento y la comprobación después de la ilustración de abajo.



POP-BOT leerá el valor de la detección del sensor en cada paso y mostrar en la pantalla SLCD16x2. Después de comprobar todas las posiciones 9, el controlador selecciona el valor más alto como resultado. Debido a que el sensor proporciona el valor más alto en la posición objetivo. Así, el resultado de esta actividad es POP-BOT puede detectar la dirección correcta de objeto de destino.

- A17.1 abrir el IDE de Arduino y crear el código del listado esbozo A17-1.
- A17.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para el robot.
- A17.3 Desconecte el cable de descarga.
- A17.4 Establece la posición de objeto de destino. Por ejemplo, 67,5 grados de ángulo y 15 cm. lejos de POP-BOT.

```

***** POP-BOT V1.0 *****
* Filename : SeekingObject.pde
* Moves servo and seek object position to show the result on SLCD
*****
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int PosValue[] = {460,610,760,1050,1340,1500,1660,1940,2220}; // Servo position value
int GP2[9];
int j,Maximum,MAX_Point;

void setup(){
//--- Servo Motor ---//
pinMode(7,OUTPUT); // Servo Motor
pinMode(14,OUTPUT); // PIR Speaker
pinMode(txPin,OUTPUT); // SLCD pin
MySerial.begin(9600); // Communication With SLCD
delay(1000);
LCD_Clear(); // LCD Clear Screen
}
void loop(){
for(j=0;j<9;j++){ // Check 9 positions
Servo_Move(PosValue[j]); // Move servo motor
GP2[j]=analogRead(5); // Read value from GP2D120
LCD_Clear();
LCD_Show_Text(0x80,"Position");
LCD_Show(0x89,j);
LCD_Show_Text(0x8A,":= ");
LCD_Show(0x8D,GP2[j]); // Show value on SLCD
delay(1000);
}
MAX_Point = GET_Point();
Servo_Move(PosValue[MAX_Point]); // Move servo to target position.
// It is selected from the most of 9 sensor values
LCD_Clear();
LCD_Show_Text(0x80,"Selected :"); // Show maximum value
LCD_Show(0x8A,MAX_Point);
LCD_Show_Text(0xC0,"Value = ");
LCD_Show(0xC8,Maximum);
Beep();delay(5000);
LCD_Clear(); // Clear LCD screen
}

/** Calculate target position ***/
int GET_Point(){
int i,Old=0,max_;
for(i=0;i<9;i++){
if(GP2[i]>Old){
Old=GP2[i];
max_=i;
}
}
Maximum=Old;
return(max_);
}

```

```

/** Servo position control */
void Servo_Move(int val){
    int i;
    for(i=0;i<20;i++){
        digitalWrite(7, HIGH);           // Set port 7 to servo port
        delayMicroseconds(val);         // Positive pulse delay
        digitalWrite(7,LOW);
        delay(20);                     // Negative pulse delay
    }
}

/** Beep function */
void Beep(){
    int i;
    for (i=0;i<600;i++){
        digitalWrite(14,HIGH);
        delayMicroseconds(150);
        digitalWrite(14,LOW);
        delayMicroseconds(150);
    }
}

/** SLCD Function */
void LCD_Show(int Position,int x){
    MySerial.print(0xFE,BYTE);          // Clear Screen
    MySerial.print(Position,BYTE);
    MySerial.print(x,DEC);             // Show Data in Decimal
}
void LCD_Show_Text(int Position,char* x){
    MySerial.print(0xFE,BYTE);          // Clear Screen
    MySerial.print(Position,BYTE);
    MySerial.print(x);                // Show text
}
void LCD_Clear(){
    MySerial.print(0xFE,BYTE);          // Clear Screen
    MySerial.print(0x01,BYTE);
}
//*********************************************************************

```

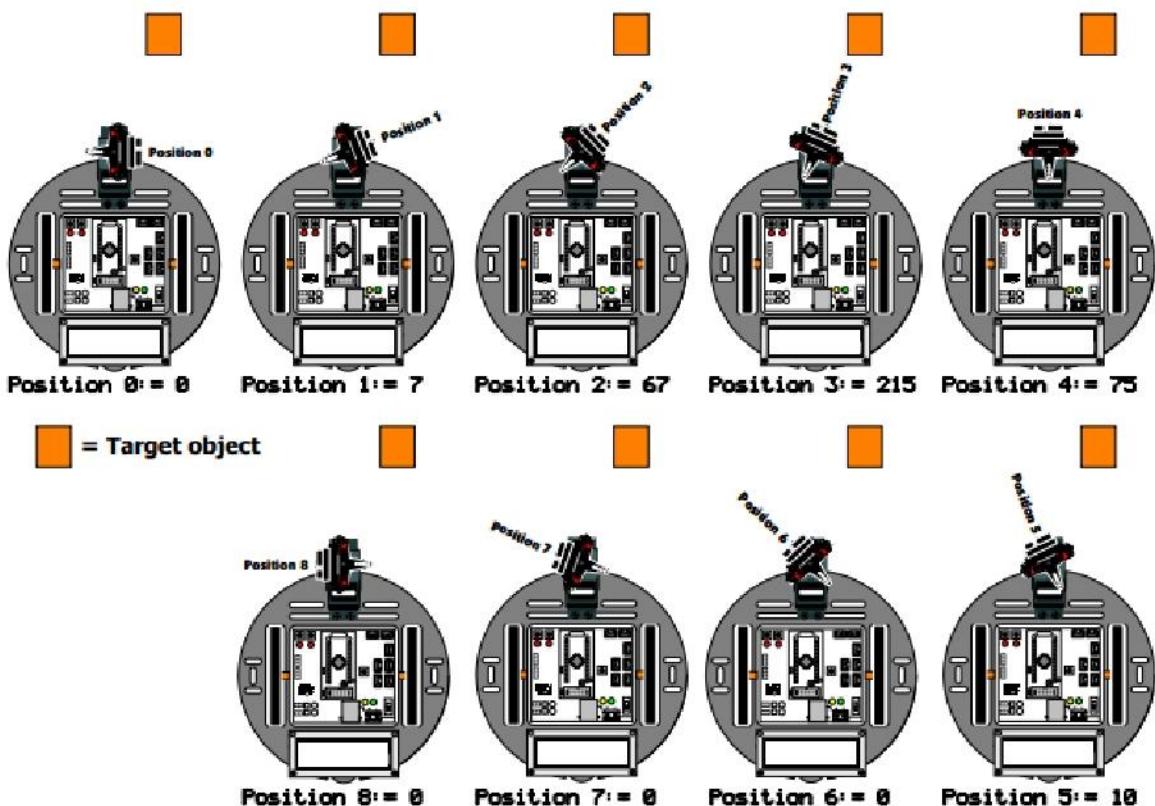
### Archivo A17-1 : Archivo SeekingObjectTest.pde

### A17.5 Encienda el POP-BOT. Vea su funcionamiento.

Después de encender el POP-BOT impulsará servo para mover el GP2D120 a la posición de último derecho, la posición 0. Es 0 grados de ángulo. POP-BOT controlador lee los datos de GP2D120 y mostrará en la pantalla SLCD16x2 de la siguiente manera:

**POSITION 0 := 0 (valor puede cambiar en cualquier robot)**

A continuación, POP-BOT impulsa la estructura GP2D120 a la posición 1 (22,5 grados de ángulo) y se lee el valor del sensor y la muestra en la pantalla SLCD. El robot va a hacer igual hasta Position8.



Después de eso, el controlador seleccionará la posición de más alto valor a la muestra en la pantalla SLCD de la siguiente manera:

**SELECTED : 3**

**VALUE = 215**

Esto significa POP-BOT detecta el objeto en la posición 3. El ángulo es de unos 67,5 grados.

## Actividad 18: POP-BOT buscador de una pelota

Esta actividad se ha modificado de la Actividad 17. Se aplica el código para el mundo real. El POP-BOT se mueve y busca el objeto de destino, la pelota. La misión estará completa cuando POP-BOT se mueve a la posición de la bola, parada y sonido.

A18-1 abrir el IDE de Arduino y crear el código del listado esbozo A18-1.

A18.2 Ajuste el POP-BOT en el modo de Programa. Sube el boceto para al robot.

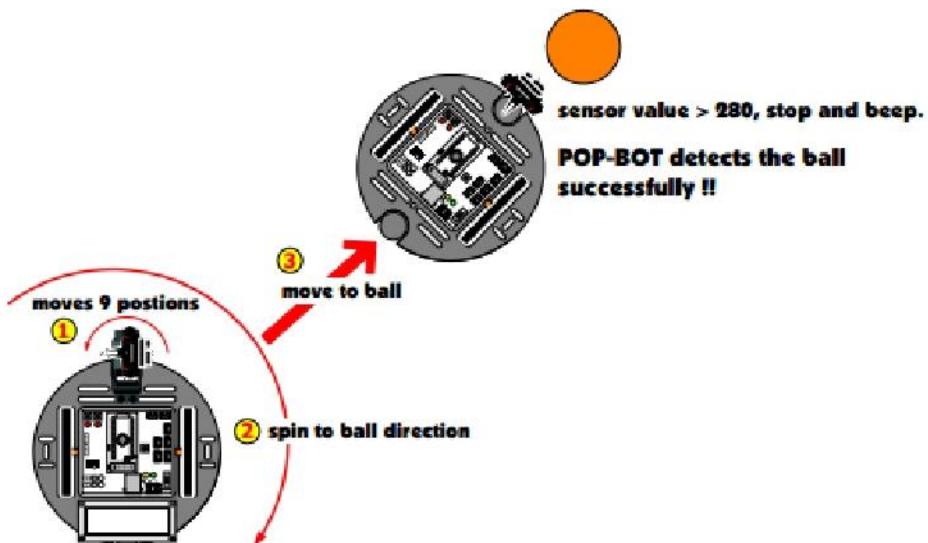
A18.3 Desconecte el cable de carga.

A18.4 Establece la posición de objeto de destino libremente en el campo. Coloque el POP-BOT en el campo. Vuelta y observar el funcionamiento.

*POP-BOT se inicia con la conducción del servomotor para buscar la pelota. La búsqueda es similar al funcionamiento de la Actividad 17, pero más rápido y no muestra el valor en el SLCD. POP-BOT se moverá a la dirección que da el máximo valor del sensor.*

*Si el valor del sensor es inferior a 20, significa no hay objeto en esta dirección. POP-BOT se dará la vuelta para cambiar la dirección.*

*Además POP-BOT compara el valor del sensor ya que más de 28, la es media POP-BOT mantener el balón listo. Debido a que el valor más de 280 es la distancia más cerca de POP-BOT de la pelota. Si no lo hace el valor del sensor llega a 280, el POP-BOT todavía buscará el balón con la misma operación.*



```
*****
* POP-BOT V1.0
* Filename : BallSeekerRobot.pde
* Seek ball and move to it.
*****
int PosValue[] = {460,610,760,1050,1340,1500,1660,1940,2220}; // Servo position value
int GP2[9];
int j,Maximum,Position;

void setup(){
    pinMode(3,OUTPUT);                      // Motor A1
    pinMode(5,OUTPUT);                      // Motor A2
    pinMode(6,OUTPUT);                      // Motor B2
    pinMode(9,OUTPUT);                      // Motor B1
    pinMode(7,OUTPUT);                      // Servo Motor
    pinMode(14,OUTPUT);                     // PIEZO Speaker
    Beep();
    delay(2000);
}

void loop(){
    Motor_Stop();
    Servo_Home();
    for(j=0;j<9;j++){                      // Set the seeking point 9 positions
        Servo_Move(PosValue[j]);            // Move servo motor
        GP2[j]=analogRead(5);              // Read valei from GP2D120
    }
    Position=MAX_Point();                  // Check lost object condition
    if(Maximum<20){
        Spin_Right(150);delay(600);      // Turn around if value < 20.
    }
    else if(Maximum>280){                // Check ball position
        Servo_Move(PosValue[MAX_Point()]);
        Beep();                          // Beep to finish
        while(1);
    }
    else{
        switch(Position){              // Seeking routine
            case 0: Spin_Right(200);
                        delay(320);
                        Forward(200);
                        delay(400-Maximum);
                        break;
            case 1: Spin_Right(200);
                        delay(240);
                        Forward(200);
                        delay(400-Maximum);
                        break;
            case 2: Spin_Right(200);
                        delay(160);
                        Forward(200);
                        delay(400-Maximum);
                        break;
            case 3: Spin_Right(200);
                        delay(80);
                        Forward(200);
                        delay(400-Maximum);
                        break;
        }
    }
}
```

```

        case 4: Forward(200);
                   delay(400-Maximum);
                   break;
        case 5: Spin_Left(200);
                   delay(80);
                   Forward(200);
                   delay(400-Maximum);
                   break;
        case 6: Spin_Left(200);
                   delay(160);
                   Forward(200);
                   delay(400-Maximum);
                   break;
        case 7: Spin_Left(200);
                   delay(240);
                   Forward(200);
                   delay(400-Maximum);
                   break;
        case 8: Spin_Left(200);
                   delay(320);
                   Forward(200);
                   delay(400-Maximum);
               }
           }
}

/** Calculate the selected position **/
int MAX_Point(){
    int i,Old=0,max_;
    for(i=0;i<9;i++){
        if(GP2[i]>Old){
            Old=GP2[i];
            max_=i;
        }
    }
    Maximum=Old;
    return(max_);
}

/** Servo position control **/
void Servo_Home(){
    Servo_Move(460);
    Servo_Move(460);
    Servo_Move(460);
    Servo_Move(460);
}
void Servo_Move(int val){
    int i;
    for(i=0;i<5;i++){
        digitalWrite(7, HIGH);          // Set port 7 to Servo output port
        delayMicroseconds(val);        // Positive pulse delay
        digitalWrite(7,LOW);
        delay(20);                    // Negative pulse delay
    }
}

```

```
void Forward(int speed){  
    analogWrite(3,speed);  
    digitalWrite(5,LOW);  
    analogWrite(6,speed);  
    digitalWrite(9,LOW);  
}  
void Motor_Stop(){  
    digitalWrite(3,LOW);  
    digitalWrite(5,LOW);  
    digitalWrite(6,LOW);  
    digitalWrite(9,LOW);  
}  
void Spin_Left(int speed){  
    analogWrite(5,speed);  
    digitalWrite(3,LOW);  
    analogWrite(6,speed);  
    digitalWrite(9,LOW);  
}  
void Spin_Right(int speed){  
    analogWrite(3,speed);  
    digitalWrite(5,LOW);  
    analogWrite(9,speed);  
    digitalWrite(6,LOW);  
}  
  
/** Beep function **/  
void Beep(){  
    int i;  
    for (i=0;i<600;i++){  
        digitalWrite(14,HIGH);  
        delayMicroseconds(150);  
        digitalWrite(14,LOW);  
        delayMicroseconds(150);  
    }  
}
```

---

**Listado A18-1: Archivo BallSeekerRobot.pde, el archivo de boceto Arduino para POP-BOT para buscar y atrapar la pelota**