Programación de vehículos móviles bajo plataforma Pololu 3π

Antonio BENITEZ[1], Leonardo OROZCO[2], Jorge DE LA CALLEJA[1] y Ma. MEDINA[1]

[1] Universidad Politécnica de Puebla
Tercer Carril del Ejido Serrano S/N, San Mateo Cuanalá Juan C. Bonilla C.P. 72640, México
abenitez@uppuebla.edu.mx

[2] ESIME-Z Instituto Politécnico Nacional
leo_4_oro@hotmail.com

Resumen

En este artículo se describe la implementación de algunos comportamientos para una un robot móvil. Este proyecto se enfoca a la comprensión y manejo de sistemas robóticos, en particular uno soportado en la plataforma *Pololu 3π*, el cual es un pequeño robot móvil de alto rendimiento, diseñado para competencias de seguimiento de línea y resolución de laberintos, entre otras cosas. Es así que la parte medular de este trabajo se enfoca a la implementación de programas que permitan desarrollar la dinámica del robot *3π*. Se presentan algoritmos para programar la velocidad de los motores, para programar y utilizar los sensores de luz y se describe una aplicación para un algoritmo de solución de laberintos.

Palabras Clave: *Robótica Móvil, Programación de Sensores.*

1. Introducción

El robot *Pololu 3π* es alimentado por 4 pilas AAA y un único sistema de tracción para los motores que trabaja a 9.25V. En lo sucesivo haremos referencia al robot *Pololu 3π* como *3π*. El *3π* es capaz de desarrollar velocidades por encima de los 100cm/s mientras realiza vueltas precisas y cambios de sentido que no varían con el voltaje de las baterías. El robot está totalmente ensamblado con dos micromotores de metal para las ruedas, cinco sensores de reflexión, una pantalla LCD, tres pulsadores y más, todo ello conectado a un microcontrolador programable. El *3π* mide aproximadamente 9,5 cm de diámetro y pesa alrededor de 83 gr. (sin baterías)

El 3π (Figura 1.) contiene un microcontrolador Atmel ATmega328 a 20 MHz con 32KB de memoria flash, 2KB de RAM, y 1KB de EEPROM. Las herramientas gratuitas de desarrollo en C y C++ así como un extenso paquete de librerías que pueden trabajar con el hardware que lleva integrado están disponibles. También hemos diseñado programas que muestran cómo trabajan los diferentes componentes del *3π*.

2. Software para trabajar con la arquitectura Pololu 3π.

La programación del robot se realiza a través del microcontrolador Atmel ATmega328. Para esto es necesario instalar varios elementos como compiladores y controladores de hardware.

2.1 Controlador USB

Es necesario instalar el controlador que hace posible la comunicación entre el robot y la computadora, para ello, hay que usar "***PololuUSBInstaller***" que podemos obtener en la página del *3π* [1], el instalador logrará que sea posible usar el dispositivo de USB como medio para poder cargar programas al robot a través de la PC.

Figura. 1 Robot Pololu 3π.

2.2 AVR Studio

AVR Studio es un IDE (Entorno de desarrollo integral) compatible con el robot 3pi. Es recomendable trabajar con AVR-Studio4, sin embargo, es muy importante que se instale primero WinAVR, de lo contrario AVR-Studio presentará problemas; ambos programas se encuentran también en la página del *3π*. La Figura 2. Muestra los IDEs de instalación para estas herramientas [1] [2].

Figura. 2. Instaladores para herramientas de programación

Por otro lado, es necesario instalar las librerías del *3π* de manera independiente al AVR-Studio. Esta librería también se obtiene de la página de *3π* descargando la carpeta **"libpololu-avr".**

Cuando se desea crear un proyecto para este robot, es necesario seleccionar como opción de compilación **AVR GCC.** Además, es necesario prestar atención al modelo del microcontrolador que usa el *3π*, en este caso es **ATmega328p** y se selecciona bajo la plataforma (Debug platform) AVR simulator.

3. Desarrollo de Programas.

El lenguaje C/C++, ofrece diversas herramientas básicas para la solución de problemas, tales como bucles, operadores lógicos, operadores aritméticos, etc.

Debido a que el microcontrolador del robot utiliza puertos, AVR-Studio ha configurado instrucciones (palabras reservadas) particulares que sirven para controlar dichos puertos sin necesidad de que el programador esté obligado a conocer comandos especiales para poder activar algún puerto [3]. Por ejemplo en la tabla 1, podemos encontrar comandos que activan a los motores. Sin embargo, estos comandos pueden ser sustituidos por la función ***set_motors (a,b).***

Tabla 1:Comomandos reconocidos por AVR-Studio

Byte Comando	Comando	Descripción
0xC1	M1 forward	Motor M1 gira adelante a una velocidad de 0 (paro) hasta 127 (máximo avance).
0xC2	M1 backward	Motor M1 gira atrás con una velocidad entre 0 (paro) hasta 127 (máximo retroceso).
0xC5	M2 forward	Motor M2 gira adelante a una velocidad de 0 (paro) hasta 127 (máximo avance).
0xC6	M2 backward	Motor M2 gira atrás con una velocidad entre 0 (paro) hasta 127 (máximo retroceso).

De la misma forma existen otras funciones y palabras reservadas que permiten manipular en su totalidad al 3π. Cabe mencionar que para la comprensión de las funciones es posible consultar directamente las librerías de *AVR-Studio*, en conjunto con la tabla completa de comandos que ofrece la guía de usuario del 3π .

3.1 Descarga de un Programa hacia el 3π

Los pasos para guardar un programa en la memoria flash del 3π son los siguientes:

A: Verificar que el programa a descargar no tenga errores de sintaxis.

B: Conectar el programador a la PC y éste al *ISP port* del 3π (el programador debe estar instalado correctamente antes de usarlo)**.**

D: Cargar en la memoria flash el archivo *test.hex* que se ha generado después de que la compilación del programa ha sido exitosa.

3.2 Inicialización y Programación de Sensores

Se han implementado varias funciones que nos permiten controlar de manera básica algunas de las funcionalidades que la plataforma 3π ofrece. A continuación se describen algunas de las funciones implementadas.

Initialize():establece la frecuencia de 20MHz, inicializa los sensores de luz, prepara la pantalla LCD para mostrar un histograma que presenta el nivel de cada sensor, recupera y muestra el nivel de voltaje de las baterías, muestra mensajes de bienvenida y reproduce tonos musicales.

sensor_motor(): Activa la lectura de los sensores y obtiene un valor para cada uno de ellos (0 - superficie blanca, 2000 - superficie oscura).

A partir de estas funciones se han desarrollado varios algoritmos que le permiten al 3π desarrollar varias tareas.

3.2.1 Programa de Control de Sensores de Luz

Este programa utiliza los sensores que están en la parte inferior del *3π* para reconocer el color de una superficie, en particular el *3π* se moverá sólo cuando se encuentre sobre superficies oscuras y redireccionará su movimiento en caso de encontrar una superficie blanca, buscando encontrar una superficie obscura para seguirla[4].

Este programa hace uso de la función *initialize()*. Sin embargo, la parte medular de este programa está en la función *sensor_motor()*. Dicha función se encarga de reconocer el color de la superficie y realizar los movimientos pertinentes.

```
//********** FUNCIÓN PRINCIPAL (MAIN) *********

int main()
{
        initialize();
        while(1)
        {
                sensor_motor();
        }
}
```

3.2.1 Programa Sigue Linea

El problema a resolver con este algoritmo consiste en matener al robot sobre la linea obscura, de tal manera que pueda desplazarse sobre ella. Para que este algoritmo funcione correctamente es necesario establecer una línea oscura sobre una superficie blanca para que sirva de guía al 3π, como se muestra en la figura 3.

Es posible aprovechar los sensores y hacer que los motores actúen de acuerdo a lo que detectan. Debido a que el 3π cuenta con 5 sensores. La figura 4. Presenta la parte inferior del robot, en ella podemos observar la posición en la que se encuentran los sensores de luz, así como otros elementos de hardware importantes.

Figura. 3 El 3π es capaz de moverse tomando como guía la línea obscura.

Para dar solución a este probelama, se consideran los siguientes 3 casos:

Cuando el sensor PC0 está sobre la línea. En este caso, el robot debe moverse ligeramente hacia la derecha, por lo que el motor izquierdo debe girar más rápido que el derecho hasta posicionar el centro del 3π sobre la línea.

Cuando los sensores PC1, PC2 ó PC3 están sobre la línea. Estos sensores son los centrales y por ello los motores deben girar a la misma velocidad, para que el robot ejecute un movimiento rectilíneo.

Cuando la línea esta sobre el sensor PC4. En este caso el robot debe moverse ligeramente hacia la izquierda, por lo que el motor derecho debe girar más rápido que el izquierdo. Hasta posicionar el centro del 3π sobre la línea.

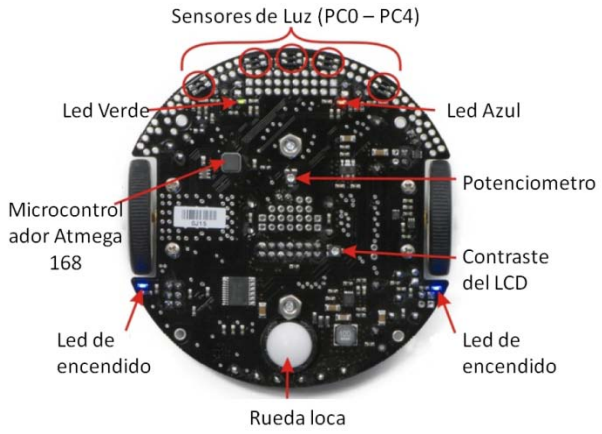

Figura. 4 Distribución frontal de los sensores de luz, leds, control de contraste para el LCD, entre otros elementos.

El programa que ejecuta estas acciones es el siguiente:

```
int main()
{
  unsigned int sensors[5]; // Un arreglo para los 5 sensores
  initialize();
  while(1)
        {
    //obtiene un valor de 0-4000,
    //si hay una línea oscura sobre
    // el sensor 0  el valor es 0,
    //si la línea esta sobre el
    //sensor 1 el valor es 1000, etc.
    unsigned int position = read_line(sensors, IR_EMITTERS_ON);
    if(position < 1000) {
        //Vamos hacia la izquierda.
        set_motors(0,100);
```

```
                left_led(1);
                right_led(0);
            }
            else if(position < 3000)  {
                //Ejecutar un movimiento rectilineo
                set_motors(50,50);
                left_led(1);
                right_led(1);
        }
        else      {
            //Vamos hacia laderecha.
                set_motors(100,0);
                left_led(0);
                right_led(1);
        }
    }
}
```

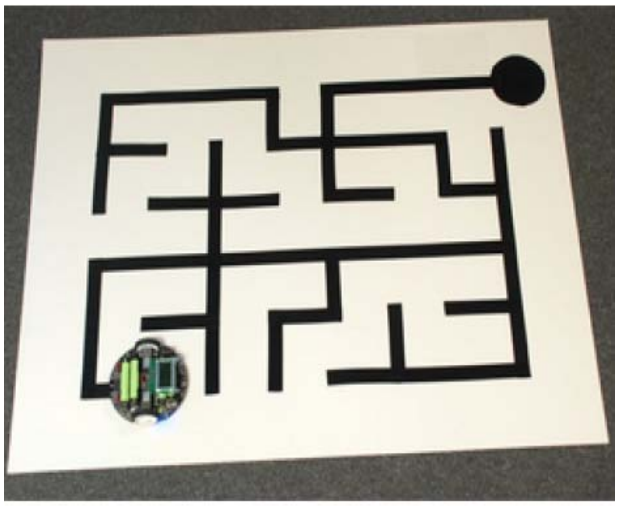

Figura. 5 Ambiente con un ejemplo de laberinto, el robot encontrará la salida cuando alcance la zona circular.

3.2.1 Programa de Solución de Laberintos

Es necesario crear una pista en forma de laberinto a partir de líneas negras y una superficie blanca (Figura 5). Es muy importante que las líneas sean lo suficientemente largas; porque, si las líneas terminales son muy cortas, el 3π reconocerá erróneamente el camino y habrá problemas para guardar la solución eficiente del laberinto.

En esta ocasión se usará la estrategia de la mano izquierda sobre el muro. Este algoritmo asegura que si el robot mantiene su mano lado izquierdo pegado al muro, el 3π encontrará la salida. El código es bastante amplio y conviene que el lector consulte directamente el programa. Sin embargo, describiremos la forma en que trabajan las funciones involucradas. Las funciones siguientes son utilizadas para resolver el problema de encontrar una salida al laberinto.

follow_segment(). Esta función se encarga de asignarle valores a los motores utilizando velocidades apropiadas para hacer movimientos suaves en curvas, la finalidad de esta función es seguir un tramo de línea y deja de ejecutarse en el momento en que se encuentra una intersección.

Se ha declarado un arreglo de caracteres llamado ***path***, en el se guardarán letras que designan el giro correspondiente:

- 'L'-Left
- 'R'-Right
- 'B'-Back
- 'S'-Straight

turn (char dir). Esta función recibe un carácter (L,R,S,B,) y de acuerdo a él envía un valor a los motores para que realicen el movimiento correspondiente.

select_turn (*unsigned char found_left, unsigned char found_straight, unsigned char found_right***).** Los parámetros que recibe esta función indican la existencia ó la inexistencia de una intersección.

Select_turn devuelve un carácter; debido a que se está usando la estrategia de la mano izquierda, la dirección a la que le dará prioridad es a la izquierda.

display_path (). Muestra en el LCD todos los caracteres que contiene el arreglo path**.**

simplify_path(). La primera vez que el 3π recorre el laberinto, registra el camino dando prioridad de giro hacia la izquierda, aunque el movimiento sea erróneo para llegar a la meta, sin embargo esta función se va a encargar de simplificar la solución del laberinto, es decir va a eliminar giros innecesarios.

Antes que cualquier cosa, se verifica que la cantidad de letras que contiene el arreglo sea mayor o igual a tres y además es necesario que exista una letra 'B' en medio de dos caracteres que pueden ser 'L' , 'R' ó 'S', por ejemplo:

Si se tiene la secuencia LBL (left, back, left) quiere decir que el robot hizo movimientos innecesarios que se deben simplificar.

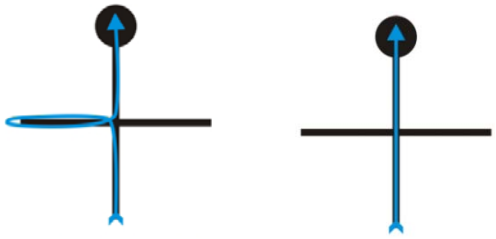

Figura. 6. Ejecucion de comandos Left +Back +Left (lado izquierdo), simplificación de comandos ejecutando S (lado derecho.

En este caso el 3π viene por la parte inferior y se dirige a la parte superior, en el centro de la cruz detecta la existencia de intersecciones en los tres sentidos, pero hace el giro a la izquierda (I) y sigue su curso hasta que detecta el fin de la línea y tiene que dar una vuelta en U (B), después de avanzar se encuentra en la misma intersección y vuelve a optar por la izquierda (L) en donde encuentra la meta. Hacer este proceso equivale a llegar a la intersección y optar por ir hacia enfrente (S), ver Figura 6.

Desde el punto de vista geométrico definimos un ángulo para cada giro:

- 'S' - 0°
- 'R' - 90°
- 'B' - 180°
- 'L' - 270°

Si sumamos la secuencia LBL obtenemos 270°+180°+270° lo que es igual a 720°, aplicando el operador módulo (%) a este valor sobre 360° (Normalizamos el ángulo a 360°), obtenemos un ángulo de 0°. Por lo tanto LBL (720°) equivale a S (0°).

Este método de suma de ángulos es válido siempre y cuando exista un carácter 'B' en medio de dos caracteres (S, L, R).

maze_solve (). Esta es la función principal que resuelve el laberinto utilizando las funciones anteriores.

4. Conclusiones

Este proyecto se desarrolló como parte del programa de verano científico del 2010. Un estudiante del ESIME-Z Instituto Politécnico Nacional trabajó con este proyecto durante su estancia de verano.

En este artículo se presentan algunos de los resultados obtenidos, como son la implementación satisfactoria de un algoritmo que a través del uso de sensores de luz es capaz de seguir una línea. Por otro lado, se implementó también un algoritmo que resuelve el problema de encontrar la salida de un laberinto. Sin embargo, el proyecto fue un poco más ambicioso, dotando al *3π* con sensores de proximidad para desarrollar tareas de detección de objetos y reconocimiento de ambientes, mismas que se describirán en próximas publicaciones.

Reconocimientos

Este proyecto se lleva a cabo gracias al apoyo del PROMEP (Programa de Mejoramiento al profesorado) como parte de apoyo al fomento a la generación y aplicación innovadora del conocimiento. Además se contó con la colaboración del estudiante Leonardo Orozco del IPN.

References

[1] http://www.pololu.com/docs/0J21/6.a
http://www.pololu.com/file/download/libpololu-avr-090605.zip?file_id=0J200
[2] http://www.pololu.com/resources
[3] http:// www.pololu.com/ resources/software
[4]http://www.pololu.com/resources/pololuprojects