“Interfaz Visual para el manejo de un mini-robot educativo”

Germán Osella Masa[1], Esteban García[1], Hugo Ramón[3], Fernando Romero[2].

Laboratorio de Investigación y Desarrollo en Informática[4]
Facultad de Informática
Universidad Nacional de La Plata

1. Resumen

Se presenta el desarrollo de una interfaz visual para el manejo de un robot micromouse, realizado en la cátedra Software de Tiempo Real (4to. Año, Licenciatura en Informática, UNLP).

Esta interfaz gráfica permite la generación y carga de programas en el PIC del robot que es el encargado de producir las señales de control de movimiento (accionamiento de los servos). Estas señales pueden tener en cuenta el estado de los sensores del robot, así como las reglas de software que se hayan especificado (por ejemplo límite máximo de recorrido en un eje, número máximo de pasos sucesivos en una dirección, etc).

El objetivo didáctico esencial es separar al alumno de la programación de alto nivel (funciones a realizar por el robot) del control de bajo nivel (drivers de los motores), desarrollando un ambiente esencialmente portable para diferentes modelos de robot.

La programación que el alumno puede especificar consta del diseño de recorridos (Avanzar, Doblar a Derecha, Detenerse, etc) y en el ambiente se puede depurar los programas desarrollados mediante un simulador de movimiento, antes de transferir el código al PIC del robot. (Luego éste actúa en forma autónoma de la PC donde reside el ambiente).
Por otra parte, dado que el robot utilizado permite sensar obstáculos y seguir líneas identificatorias del recorrido, se pueden desarrollar algoritmos elaborados que simulan tareas en tiempo real de robots industriales.

Palabras Clave
Sistemas de Tiempo Real. Ambientes gráficos. Control de Robots. Simulación.

[1] Alumnos avanzados de la Licenciatura en Informática. Facultad de Informática, UNLP.
[2] Jefe de Trabajos Prácticos Ded. Excl., LIDI. Facultad de Informática, UNLP. E-mail: fromero@info.unlp.edu.ar
[3] Profesor Adjunto Ded. Excl., LIDI. Facultad de Informática, UNLP. E-mail: hramon@info.unlp.edu.ar
[4] Calle 50 y 115 Primer Piso, (1900) La Plata, Argentina, Teléfono 54-221-4227707 *WEB: lidi.info.unlp.edu.ar*

2. Objetivos

Dado que el programa que ejecuta el robot es construido en un lenguaje de programación, el usuario que quiera obtener algún efecto en dicho dispositivo debe tener alguna noción de programación y conocer el lenguaje provisto. Además, el programa para compilar y cargar el programa en la memoria del PIC provee un simple editor de texto para ingresar el código.

Entonces nuestro objetivo princpal es el de proveer una interface visual, agradable e intuitiva para que cualquier tipo de usuario pueda obtener algún resultado traducido en movimientos del robot.

Un objetivo secundario es proveer una simulación gráfica del robot que permita al usuario comprobar el programa en forma virtual antes de ser cargado en el robot.

3. Tipo de robot utilizado

El Micromouse consiste en un robot que se moviliza mediante dos motores que hacen girar independientemente y ambos sentidos dos ruedas situadas en ambos laterales del dispositivo, pudiendo de esta manera avanzar, retroceder y girar a ambos lados.

Como funciones adicionales proporciona un circuito para la detección de obstáculos y otro para el seguimiento de material reflector de luz infrarroja.

Como control central dispone de un microcontrolador (el **PIC 16C56**), el cual dispone de una memoria de tipo **EEPROM** en la cual se guarda el programa que debe ejecutar, el cual se proporciona exteriormente mediante una entrada apropiadamente acondicionada al puerto paralelo de una PC. Dicho programa se escribe en el lenguaje **STAMP BASIC I,** cuyo compilador y cargador están incluidos en el paquete del microcontrolador. El intérprete del lenguaje se encuentra en la memoria interna del PIC.

4. Estrategia y resolución

El primer aspecto planteado fue el sistema operativo sobre el cual se iba a desarrollar. La decisión no fue difícil ya que Windows, dadas sus características gráficas, facilita la tarea de una interface visual, además de ser bastante común en las PC de hoy.

Luego, nos dedicamos a explorar el robot en cuanto a su funcionamiento y explotar sus capacidades al máximo para ver con que posibilidades contábamos. Podemos ver en esta parte tres aspectos perfectamente diferenciados:

- **Movimientos**

  Se estudiaron los parámetros para que el robot avance, retroceda y gire a voluntad nuestra, cotejando dichos parámetros con alguna unidad de medida (Centímetros en el caso del avance y retroceso y grados en el caso de los giros).

- **Detección de obstáculos**

  Se trabajó en el método utilizado para identificar obstáculos. Se buscó optimizar y ajustar al máximo la detección, de manera que se realice en una distancia no muy

lejana ni muy cercana. Además se buscó identificar cuando “ve” un obstáculo al frente, cuando a la derecha y cuando a la izquierda.

- **Seguimiento de material reflector de luz infrarroja**

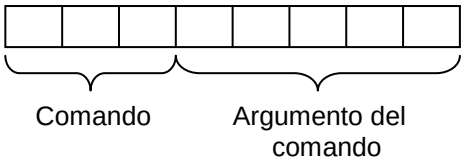
  Este seguimiento se hace en base a sensores situados en el frente del robot. Se buscó la manera que el dispositivo se mueva siguiendo un recorrido definido por una cinta blanca de un ancho razonable pegada en una mesa cuya pintura no poseía grandes cualidades reflectantes. De esta manera se analizó las diferentes situaciones para poder establecer políticas a seguir cuando se “pierde el rastro” y poder retomarlo lo antes posible.

En un primer bosquejo, la idea era proveer una interface que permitiera construir de manera visual un algoritmo insertando libremente instrucciones para movilizar el robot, detección de obstáculos y seguimiento de material reflector de luz infrarroja pero en la exploración del robot encontramos un obstáculo: **la escasa capacidad de la memoria EEPROM del PIC (sólo 256 bytes)**.

Este contratiempo nos obligó a condicionar la estructura algorítmica del programa y a limitar la cantidad de movimientos que pueden programarse.

El problema se resolvió estableciendo un programa que es fijo y que lee constantemente y uno tras otro bytes almacenados en el final de la memoria. Cada byte representa un movimiento del robot donde los tres primeros bits representan el tipo de movimiento (adelante, atrás, giro izquierda y giro derecha) y los otros cinco bits

Forma en que se codifica una orden en un byte



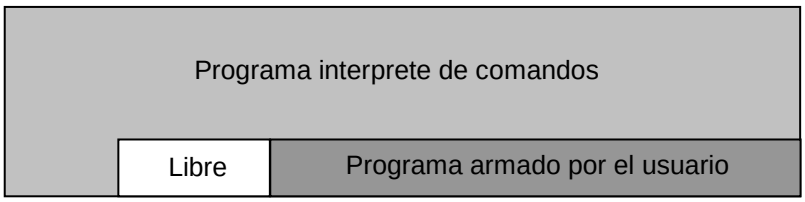
representan la cantidad, siendo centímetros para avance y retroceso, y grados para los giros.

El programa fijo tiene una estructura algorítmica en la cual se ejecutan los movimientos de un programa principal. Luego de cada movimiento del programa principal, se chequea la presencia de obstáculos. Si no hay, sigue con el próximo movimiento. Si hay obstáculo, dependiendo si está al frente, a la izquierda o a la derecha se ejecutará un subprograma para resolver la situación.

El programa principal y los subprogramas alternativos para los obstáculos son los “programables”. El programa principal y los subprogramas terminan con un byte que puede indicar si terminó o si se repite nuevamente el programa.

Los bytes correspondientes a cada programa se almacenan en el final de la memoria del PIC.

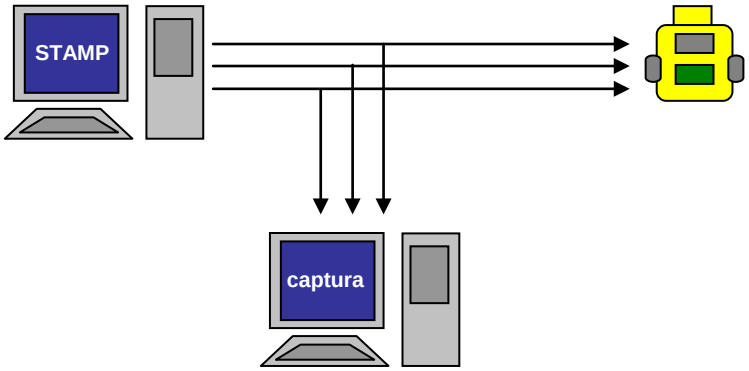
De esta manera, en la interface para programar el micromouse se pueden distinguir cuatro secciones, una para el programa principal y otro para las subrutinas para resolver la presencia de obstáculos.



Una vez resuelto el problema y construida la aplicación, que a través de una interface visual permite programar el micromouse, se podía obtener un código fuente en STAMP BASIC I, que podía ser compilado y enviado al robot mediante el editor que incluye el paquete.

Sin embargo, nos habíamos propuesto enviar el código compilado directamente desde nuestra aplicación hasta el dispositivo. Como primera medida, intentamos determinar como era el código enviado, cosa que se pudo lograr a través de una sentencia provista por el STAMP BASIC I mediante la cual creaba un archivo cuyo contenido eran los bytes que enviaba.

Luego, quisimos identificar que política utilizaba la memoria EEPROM provista para el PIC para ser cargada. Esto se realizó conectando una PC que capturaba lo que otra PC enviaba hacia la memoria EEPROM a través de su puerto paralelo.



En una PC se corrió el STAMP BASIC I con un código generado por nuestra aplicación y en la otra se corrió un programa que capturaba lo que entraba por el puerto paralelo, como si fuera la EEPROM del micromouse. De esta manera se pudo ver que alternaba pulsos de sincronismo con pulsos que eran bits del código que se transmitía.

Al llegar a este punto, determinamos que se podía alargar demasiado el tiempo de terminación de este trabajo ya que había que identificar los tiempos de sincronismo y también como trabajar con el puerto paralelo desde Windows, cosa que no es del todo trivial, ya que se requería un acceso exclusivo y además, que se respetaran los tiempos de ejecución, cosa poco probable en un sistema multitarea.

Por estos motivos, se acordó delegar el desarrollo de esta funcionalidad en otro grupo de manera que sólo se centraran en esto ya que es todo un tema y no tengan que empezar de cero a investigar sobre el robot, cosa que ya habíamos realizado nosotros

5. Especificación de la interfaz

Micromouse View ofrece una interface visual para la edición de programas que se ejecutan sobre el Micromouse de Lynxmotion. Mediante una interface orientada a ventanas se pueden programar movimientos sobre el micromouse y hasta crear rutinas que se activan cuando los sensores de proximidad detectan obstáculos.

5.1. Funciones Básicas

Como la mayoría de los programas bajo Windows, **Micromouse View**, ofrece las funciones de **crear nuevos programas, guardarlos y abrirlos**, funciones que se encuentran en el menú **Archivo**.

Estas funciones también están en la barra de herramientas en los botones  Para **abrir y guardar** se ofrece la típica interface de Windows para seleccionar archivos. Si el programa que se está editando es un programa nuevo que nunca se guardó, al guardarlo se ofrecerá la posibilidad de cambiar el nombre. Si no era un programa nuevo se guardará con el nombre original con el que se abrió. En este caso, si se quiere cambiar el nombre, habrá que seleccionar la opción **Guardar Como** en el menú **Archivo**. Los programas que edita **Micromouse View** tienen por defecto extensión "mmp" (Micromouse program).

5.2. Edición de Programas

Una vez que se abrió o se creó un programa (ver Funciones Básicas.) o bien al iniciarse **Micromouse View**, se podrán ver cuatro cuadrantes, tres de los cuales de igual tamaño y uno de tamaño mayor. El de mayor tamaño mostrará el programa principal, el cuál se ejecutará normalmente en caso de no encontrar ningún obstáculo. Los de menor tamaño corresponden a rutinas que se ejecutan al encontrar un obstáculo, al frente, a la derecha o a la izquierda. Cuando el Micromouse encuentra un obstáculo, ejecutará la rutina correspondiente y retornará al programa principal en donde lo interrumpió.

Edición

Posicionando el mouse sobre el programa que se quiere editar y ejecutando el pop-up menú (botón derecho del mouse) se ofrecen las posibilidades de edición para el programa.

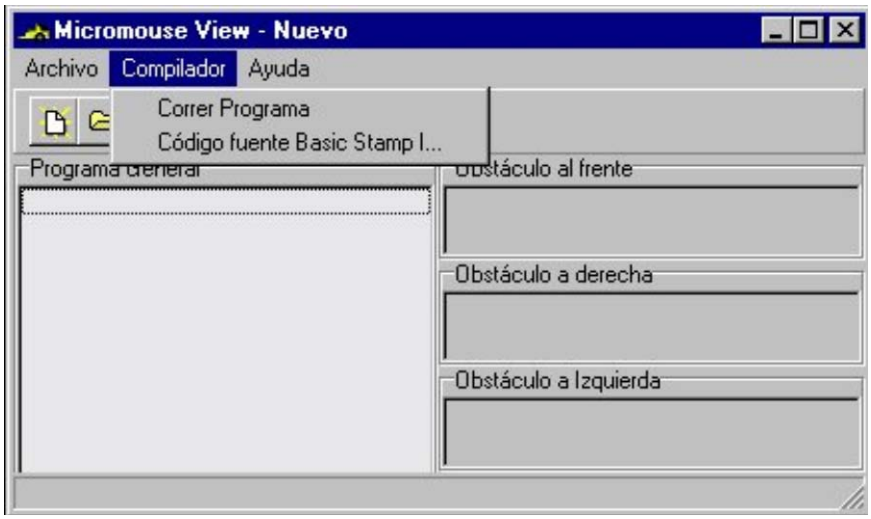
:

- **Agregar Movimiento:** Agrega un movimiento al final.
- **Insertar Movimiento:** Inserta un movimiento en la posición seleccionada.
- **Modificar Movimiento:** Inserta el movimiento seleccionado.
- **Eliminar Movimiento:** Elimina el movimiento seleccionado.
- **Repetición:** Habilita o deshabilita un ciclo infinito de ejecución para el programa en edición. También se puede habilitar o deshabilitar con el botón  de la barra de herramientas, en cuyo caso, se aplicará al programa activo (programa cuyo cuadrante está de color blanco).

Las opciones de insertar, modificar o eliminar aparecerán deshabilitadas cuando el programa no tenga movimientos o no haya ningún movimiento seleccionado.
En las opciones de agregar, insertar y modificar, se mostrará una ventana en la cual hay que elegir el tipo de movimiento (**Avanzar, Retroceder, Girar izquierda, Girar derecha**) y el parámetro (entre 1 y 30). En el caso de los movimientos de avance y retroceso, el parámetro representa centímetros. En el caso de los giros, una unidad del parámetro equivale a un giro de 45º en la dirección seleccionada.

5.3. Compilación

Ya editado el programa (ver Edición de Programas) o vez abierto un programa editado, se debe trasladar el programa a la memoria del Micromouse.

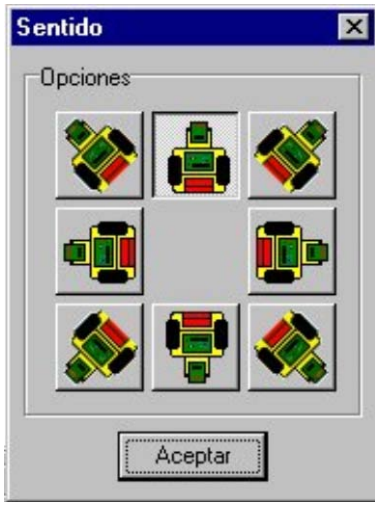


Hay dos opciones
- A través de la opción **Correr programa** en el menú **Compilador**, la cual envía el programa al Micromouse el cual debe estar conectado y con el PIC encendido.
- La opción **Código fuente Basic Stamp I** en el mismo menú, permite en un archivo seleccionado por el usuario, almacenar el código fuente en Basic Stamp I correspondiente al programa editado. Esta opción permite mayor elasticidad ya que permitiría editar el código fuente y luego enviarlo mediante el programa **Stamp.exe** al Micromouse, debiendo estar éste en las mismas condiciones que la opción anterior.
Una vez enviado el programa por una vía u otra, sólo resta proporcional alimentación eléctrica a los servos del Micromouse.

5.4. Funciones Especiales

A continuación se detallarán una serie de funciones que no modifican la edición y tienen un carácter adicional:
- **Presentación preliminar**
Esta función permite tener una pre visualización de los movimientos que hará el Micromouse ejecutar el programa Se puede acceder a dicha opción a través del botón  en la barra de herramientas en la barra de herramientas.
Se mostrará una pantalla donde se podrá configurar la velocidad de la ejecución y si se quiere o no repetición continua de la ejecución. Además se pueden dibujar algunos obstáculos, siempre y cuando la pre visualización esté detenida, para ver el efecto de las rutinas para esquivar obstáculos. Para esto se debe hacer clic en el botón **Obstáculos** y con el movimiento del mouse y presionando el botón izquierdo, dibujar mediante líneas los obstáculos. Luego, haciendo clic en el botón **Listo Obstáculos,** se deja de dibujar.
Una vez configurados estos parámetros, haciendo clic en el botón **Iniciar** se pedirá que se seleccione en el cuadrante principal, la posición desde donde comenzará la ejecución.

Luego de la selección de la posición se mostrará una interface que permite elegir la orientación del Micromouse en la simulación.



Una vez hecho este proceso comienza la simulación, que se puede parar haciendo clic en el botón **Detener** y luego seguir haciendo clic en **Continuar.**

- Line Tracker

Esta opción permite obtener en lugar de un programa tradicional, un programa en el cual el recorrido efectuado por el Micromouse depende del seguimiento de una línea que refleja luz ultravioleta. Al seleccionar esta opción, desaparecen los programas de los cuadrantes, los cuales quedan inhabilitados y no se podrán editar. Los programas no se pierden, se recuperan al deshabilitar la opción **Line Tracker** que se habilita o desabilita con el botón 

situado en la barra de herramientas.

6. Resultados obtenidos

En definitiva, el resultado final fue una aplicación que ofrece una interfaz amigable para poder construir un programa que controlara al micromouse, creando un archivo que contiene el código fuente para ser levantado por el editor del STAMP BASIC I y enviado desde allí al micromouse. Además, la aplicación permite ver una simulación del comportamiento del robot antes de correrlo en él, lo cual en este caso no es de gran importancia ya que no hay riesgos en caso de equivocaciones, pero somos conscientes que en aplicaciones de tiempo real verdaderas, siempre es mejor tener una vista previa de las acciones a realizar, aunque sea una aproximación, ya que puede estar en juego tecnología muy costosa y hasta vidas humanas. Creemos que el objetivo principal fue alcanzado, ya que no es necesario saber de programación ni del lenguaje para poder controlar el robot.

Otro objetivo alcanzado fue el de explorar a fondo las posibilidades del micromouse y la especificación de éstas para su utilización directa. A continuación se muestra esta especificación y el manual de usuario de la aplicación.

7. Conclusiones y líneas de trabajo actuales

La principal limitación que se encontró en el robot utilizado es su escasa capacidad de procesamiento. El PIC con que cuenta el robot tiene una capacidad de memoria limitada a 512 bytes, de los cuales 256 bytes se utilizan en el interprete, lo cual hace que los programas posibles de implementar sean de pequeño tamaño. Por otro lado, el potencial de posibles tareas a realizar por el robot puede ser facilmente ampliado por el agregado de sensores de otro tipo. Todo lleva a la conclusión de que para futuros trabajos debiera contarse con un PIC de mayor capacidad. Ello permitiría optimizar recorridos a través de algoritmos mas sofisticados, como también implementar recorridos de mayor complejidad. Otra posible vía de enriquecer esta herramienta educativa sería poder suministrarle información del ambiente a través de algún vínculo de comunicación (ultrasonido, rayos infrarrojos o RF).

8. Bibliografía

Technical Service & Solutions. Manual “Robotic Arm Kit, 1977.

“Robótica básica”. Edson de Paula Ferreira. ed. V EBAI, 1991.

“Real-Time Systems and Programing Languajes”. A. Burns & A. Willings. Addison Weley ISBN 0-201-40365-X.