Aplicación de Técnicas de Robótica e Inteligencia Artificial sobre un robot móvil utilizando el entorno de programación Visual Studio.Net

Directores: **Ing. Claudio Verrastro, Ing. Roberto Barneda**
Revisión: **Ing. Juan Carlos Gómez**
www.secyt.frba.utn.edu.ar/gia/

Módulo 2: "Aplicando un algoritmo para evasión de obstáculos"

Autores: **Carlos Fernando Carmona, Damián De Biase, Cesar Foti, Sebastián Verrastro y Daniel Lopez Amado**

Índice general

Figuras

1 Introducción

Se presenta un conjunto de prácticas orientadas a la aplicación de técnicas de Inteligencia Artificial en Robótica.
Para el desarrollo de estas prácticas se utiliza la plataforma de programación *Visual Studio.NET* de *Microsoft â* y un robot *ER1* de *Evolution Robotics â*.
(http://www.evolution.com/er1/) (http://msdn.microsoft.com/vstudio/)
Se trabaja con una PC portátil (notebook), montada sobre el robot, con un módulo de comunicación inalámbrico (Wi-Fi). La ejecución, sobre la PC portatil, del programa desarrollado en *Visual Studio.NET*, llamado *ServidorER1*, permite controlar todas las funciones del robot y sus accesorios: cámaras y pinzas. Este control puede realizarse en forma local o remota, vía Internet, a traves de otra computadora utilizando comunicación interproceso (**Socket**). Esta última es la modalidad elegida para el desarrollo de las prácticas. Se crean entonces aplicaciones (clientes) propias que ejecutadas en otras PCs permiten acceder al control del robot.

SOCKET:
Un socket es un identificador para un servicio concreto en un nodo concreto de la red. El socket consta de una dirección de nodo y de un número de puerto que identifica al servicio

2 Objetivos

El objetivo de este módulo es aplicar un **algoritmo** de búsqueda **heurística** (A*) para que el robot pueda ir desde un punto inicial hasta otro final esquivando obstáculos de dos formas posibles, realizando el camino más corto o efectuándolo de forma más rápida. Como complemento de esta, se realizarán una serie de prácticas previas, que consisten en:

- Reemplazo de los drivers del robot ER1.
- Reemplazo del programa del robot por uno realizado en **VB.NET** (Servidor ER-1).
- Ampliación y mejora del software cliente.

Estas prácticas se acompañan con ejemplos y ejercicios, además se vinculan de manera tal que al final de este módulo se obtenga un programa integrado.

ALGORITMO:
Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

HEURÍSTICA:
Técnica de la indagación y del descubrimiento. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

3 Requisitos fundamentales para la realización de las prácticas

Para la realización de las prácticas que se presentan a continuación se debe contar con el siguiente equipamiento:

- Al menos un robot ER1 armado.
- Una PC portátil[1], la cual será montada sobre el robot, dotada de 2 puertos USB (como mínimo) y tecnología Wi-Fi, Blue tooth u otra.
- Un ordenador con acceso a Internet, desde el cual se controlará al robot.
- El robot debe ser armado siguiendo las instrucciones del manual que viene con el equipo.
- Se debe controlar que las baterías de la PC y del robot se encuentren cargadas.

[1] NOTA: La PC portátil (notebook, tablet PC...) maneja su propia batería, la del robot es un módulo metálico con interruptor y terminal de carga.

4 PRÁCTICA Nº 1

Autores: **Daniel J. López Amado, Sebastián Verrastro**

4.1 Driver para el ER1

4.1.1 Introducción

El software "ER1 Robot Control Center" de Evolution Robotics permite manejar al Robot ER1 desde una PC remota mediante el envío de comandos. Los mismos son ejecutados en forma secuencial, es decir, una vez que finaliza la ejecución de un comando el Robot comienza con la ejecución del siguiente. Por esta razón no es posible generar una trayectoria curva dado que no se puede avanzar y girar al mismo tiempo.

Para entender la forma de solucionar este problema se dará una breve explicación del hardware del ER1

El diagrama en bloques del módulo de hardware del ER1 (Figura 1) está compuesto por un bloque encargado de la comunicación con la PC a través del puerto USB y por un segundo bloque encargado de controlar los dos motores paso a paso del Robot.

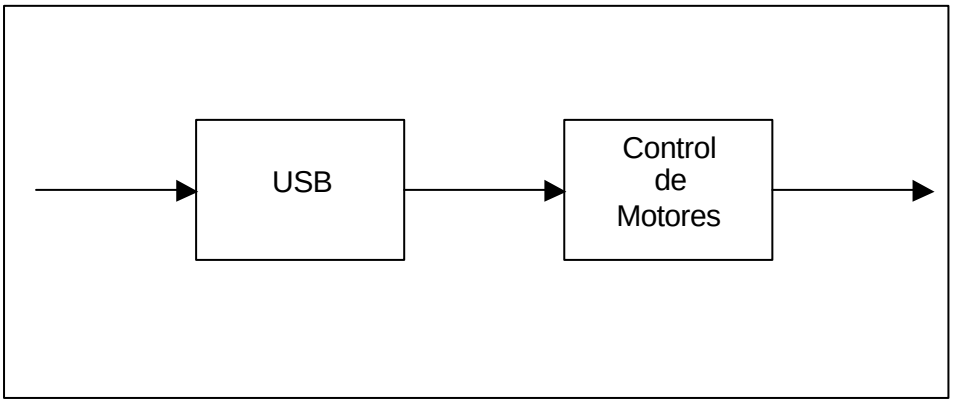

Figura 1 Diagrama en bloques del hardware del ER1

El bloque de comunicación está implementado con el chip FT232 que es un conversor USB-SERIE. Es necesario instalar los drivers provistos por Evolution Robotics en la Notebook donde se conectará el módulo de hardware para que en la misma aparezca un COM virtual.

El bloque de control está implementado con el chip MC3410 que es un controlador de motores paso a paso. Este chip permite la recepción de comandos para realizar distintas acciones.

4.1.2 Implementación del driver

Se generó un driver que permite enviarle comandos a través del COM virtual directamente al controlador de motores paso a paso del ER1.

Tabla de Comandos permitidos

ClearInterrupt	0xAC	AdjustActualPostion	0xF5
ClearPositionError	0x47	GetActivityStatus	0xA6
GetActualVelocity	0xAD	GetCaptureValue	0x36
GetChecksum	0xF8	GetCommandedAcceleration	0xA7
GetCommandedPostion	0x1D	GetCommandedVelocity	0x1E
GetCurrentMotorCommand	0x3A	GetDerivative	0x9B
GetEventStatus	0x31	GetHostIOError	0xA5
GetIntegral	0x9A	GetInterruptAxis	0xE1
GetPhaseCommand	0xEA	GetPositionError	0x99
GetSignalStatus	0xA4	GetTime	0x3E
GetTraceCount	0xBB	GetTraceStatus	0xBA
GetVersion	0x8F	InitializPhase	0x7A
NoOperation	0x00	ReadAnalog	0xEF
ReadBuffer	0xC9	ReadIO	0x83
Reset	0x39	ResetEventStatus	0x34
SetAcceleration	0x90	GetAcceleration	0x4C
SetActualPostion	0x4D	GetActualPostion	0x37
SetActualPostionUnits	0xBE	GetActualPostionUnits	0xBF
SetAutoStopMode	0xD2	GetAutoStopMode	0xD3
SetAxisMode	0x87	GetAxisMode	0x88
SetAxisOutSource	0xED	GetAxisOutSource	0xEE
SetBreakpoint	0xD4	GetBreakpoint	0xD5
SetBreakpointValue	0xD6	GetBreakpointValue	0xD7
SetBufferFunction	0xCA	GetBufferFunction	0xCB
SetbufferLength	0xC2	GetBufferLength	0xC3
SetBufferReadIndex	0xC6	GetBufferReadIndex	0xC7
SetBufferStart	0xC0	GetBufferStart	0xC1
SetBufferWriteIndex	0xC4	GetBufferWriteIndex	0xC5
SetCaptureSource	0xD8	GetCaptureSource	0xD9
SetCommutationMode	0xE2	GetCommutationMode	0xE3
SetDeceleration	0x91	GetDeceleration	0x92
SetDerivativeTime	0x9C	GetDerivativeTime	0x9D
SetDiagnosticPortMode	0x89	GetDiagnosticPortMode	0x8A
SetEncoderModulus	0x8D	GetEncoderModulus	0x8E
SetEncoderSource	0xDA	GetEncoderSource	0xDB
SetEncoderToStepRatio	0xDE	GetEncoderToStepRatio	0xDF
SetIntegrationLimit	0x95	GetIntegrationLimit	0x96
SetInterruptMask	0x2F	GetInterruptMask	0x56
SetJerk	0x13	GetJerk	0x58
SetKaff	0x93	GetKaff	0x94
SetKd	0x27	GetKd	0x52
SetKi	0x26	GetKi	0x51
SetKout	0x9E	GetKout	0x9F
SetKp	0x25	GetKp	0x50

SetKvff	0x2B	GetKvff	0x54
SetLimitSwitchMode	0x80	GetLimitSwitchMode	0x81
SetMotionCompleteMode	0xEB	GetMotionCompleteMode	0xEC
SetMotorBias	0x0F	GetMotorBias	0x2D
SetMotorCommand	0x77	GetMotorCommand	0x69
SetMotorLimit	0x06	GetMotorLimit	0x07
SetMotorMode	0xDC	GetMotorMode	0xDD
SetNumberPhases	0x85	GetNumberPhases	0x86
SetOutputMode	0xE0	GetOutputMode	0x6E
SetPhaseAngle	0x84	GetPhaseAngle	0x2C
SetPhaseCorrectionMode	0xE8	GetPhaseCorrectionMode	0xE9
SetPhaseCounts	0x75	GetPhaseCounts	0x7D
SetPhaseInitializeMode	0xE4	GetPhaseInitializeMode	0xE5
SetPhaseInitializeTime	0x72	GetPhaseInitializeTime	0x7C
SetPhaseOffset	0x76	GetPhaseOffset	0x7B
SetPhasePrescale	0xE6	GetPhasePrescale	0xE7
SetPostion	0x10	GetPostion	0x4A
SetPostionErrorLimit	0x97	GetPostionErrorLimit	0x98
SetProfileMode	0xA0	GetProfileMode	0xA1
SetSampleTime	0x38	GetSampleTime	0x61
SetSerialPortMode	0x8B	GetSerialPortMode	0x8C
SetSettleTime	0xAA	GetSettleTime	0xAB
SetSettleWindow	0xBC	GetSettleWindow	0xBD
SetSignalSense	0xA2	GetSignalSense	0xA3
SetStartMode	0xCC	GetStartMode	0xCD
SetStartVelocity	0x6A	GetStartVelocity	0x6B
SetStopMode	0xD0	GetStopMode	0xD1
SetSynchronizationMode	0xF2	GetSynchronizationMode	0xF3
SetTraceMode	0xB0	GetTraceMode	0xB1
SetTracePeriod	0xB8	GetTracePeriod	0xB9
SetTraceStart	0xB2	GetTraceStart	0xB3
SetTraceStop	0xB4	GetTraceStop	0xB5
SetTraceVariable	0xB6	GetTraceVariable	0xB7
SetTrackingWindow	0xA8	GetTrackingWindow	0xA9
SetVelocity	0x11	GetVelocity	0xAB
Update	0x1A	WriteBuffer	0xC8
WriteIO	0x82		

4.1.3 Explicación de la Clase RCMotion

Estructuras utilizadas por los métodos

```
typedef struct PWDCmd_struct {
      byte address;
      byte checksum;
      byte axis;  // initialized by constructor, always 0
      byte code;
      byte data[6];
      byte size;  // meta data, should be not be send.
}PWDCmd;
```

```
typedef struct PWDReply_struct {
public:
      byte data[PWMREPLYSIZE];
      DWORD dwSize;
      DWORD Status();
      DWORD CheckChecksum();
}PWDReply;
```

Métodos

```
/*    RCMInit: Inicializa el puerto de comunicación

      Parámetros:
            CP: Número de Puerto de Comunicación

      Retorno:
      ERR_RCM_NO_ERROR: Si fue correctamente inicializado.
      ERR_RCM_INVALID_PORT: Si no lo fue.
*/
int RCMotion::RCMInit(int CP);

/*    SetCmd: Arma un commando compatible con el protocolo
      que utiliza el controlador de motores paso a paso.

      Parámetros:
            paddres: Dirección
            pcode: Código de Comando
            wData1: Dato número 1
            wData2: Dato número 2

      Retorno: Comando armado
*/
PWDCmd RCMotion::SetCmd(byte paddress, byte pcode, WORD
wData1, WORD wData2);

/*    SendCmd: Envia por el Puerto de comunicación un
      comando.

      Parámetros:
            cmd: Comando a enviar
            Retorno: Respuesta al comando enviado
*/
PWDReply RCMotion::SendCmd(PWDCmd *cmd);

/*    WheelUpdate: Hace efectivo el commando enviado con la
      función SendCmd.
*/
void RCMotion::WheelUpdate()

/*    SetPower: Setea la potencia de los motores.
      Parámetros:
      wValue: Potencia de los motores.
*/
void RCMotion::WheelSetPower(WORD wValue);
```

```
/*    WheelInit: Inicializa el controlador de motores
*/
void RCMotion::WheelInit()

/*    WheelSetVelocity: Setea la velocidad de las avance
      Parámetros:
      dwSpeed: Velocidad.
*/
void RCMotion::WheelSetVelocity(long dwSpeed)

/*    WheelSetTurn: Setea la velocidad de giro
      Parámetros:
      dwSpeed: Velocidad.
*/
void RCMotion::WheelSetTurn(long dwSpeed)
```

4.2 Ejemplo del uso del driver

Se desarrolló un software de ejemplo de utilización del driver (ControlER1_V1.01_Version2005). El mismo cuenta con 5 botones que activan los dos motores simultáneamente y 6 botones que los activan individualmente.

Botones de activación simultánea de los dos motores

Adelante: Los dos motores giran en sentido inverso para lograr que el Robot avance hacia delante.

Izquierda: Los dos motores giran en el mismo sentido para lograr que el Robot gire hacia la izquierda.

Derecha: Los dos motores giran en el mismo sentido para lograr que el Robot gire hacia la derecha.

Atras: Los dos motores giran en sentido inverso para lograr que el Robot avance hacia atrás.

Parar: Detiene a los dos motores.

Botones de activación individual de los motores

Motor_Izq_Adelante: El motor de la izquierda gira hacia adelante.

Motor_Izq_Atras: El motor de la izquierda gira hacia atrás.

Motor_Izq_Parar: Detiene al motor de la izquierda.

Motor_Der_Adelante: El motor de la derecha gira hacia adelante.

Motor_Der_Atras: El motor de la derecha gira hacia atrás.

Motor_Der_Parar: Detiene al motor de la derecha.

4.3 Sumario

Se desarrolló un driver para el manejo del hardware del robot ER1 a través del COM. Si bien este driver puede utilizarse en cualquier aplicación que sea ejecutada en la PC que va sobre el robot, utilizado desde el Servidor ER1 de la práctica siguiente, permite mucha más versatilidad que la obtenida inicialmente con el ER1 Robot Control Center (RCC).

Aún cuando este driver no sea una aplicación directa de inteligencia artificial, objetivo de estos módulos, se consideró necesaria su inclusión para permitir en las sucesivas prácticas un mejor empleo del robot y mayores posibilidades en los algoritmos de control.

5 PRÁCTICA Nº 2

Autores: **Carlos Fernando Carmona, Damián De Biase, Cesar Foti**

5.1 Descripción del Servidor ER-1

5.1.1 Introducción

SERVIDOR:
Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

CLIENTE:
Es un programa que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota.

El **Servidor** ER-1 es un software desarrollado en **Visual Studio.NET** que sirve para controlar el robot ER-1 en forma local y aceptar la conexión de programas **clientes** para su control en forma remota. Este software reemplaza al programa original RCC del robot y se ejecuta en la PC portátil que debe ir montada sobre el mismo.

5.1.2 Descripción general

Como se puede observar en la Figura 2 la pantalla del Servidor se divide en cuatro zonas, cada una de la cuales corresponde a: visualización de la imagen recibida desde la cámara, pantalla de estado, configuración y botones de control.

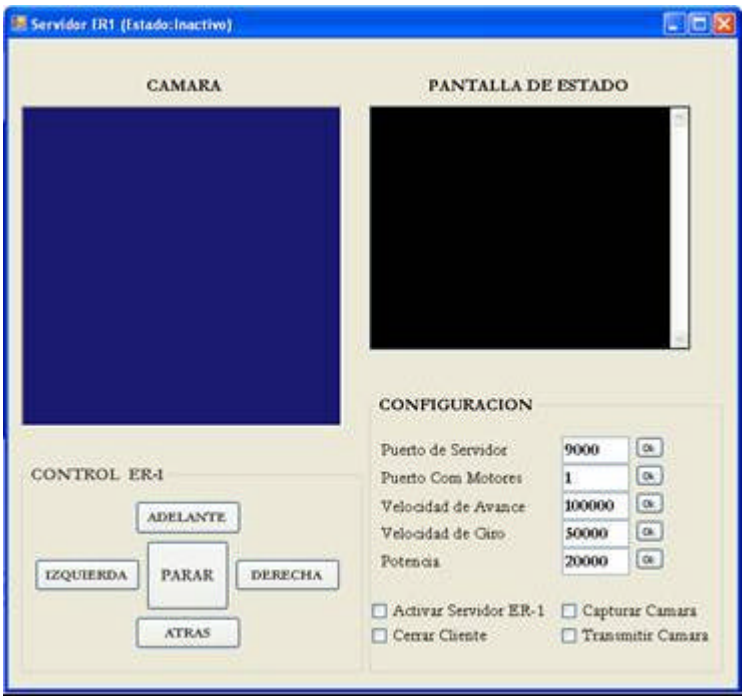

Figura 2 Servidor ER-1

Zona cámara: se visualiza la imagen que capta la cámara del robot.
Zona pantalla de estado: se visualizan los comandos recibidos desde el cliente y se informa de las acciones que ejecuta el servidor.
Zona Control ER-1: botones para el manejo local del robot.
Zona configuración: configuración general del servidor y el robot.

Las características principales del Servidor ER-1 son:

- Manejo local del robot.
- Visualización de la imagen recibida desde la cámara.
- Configuración de los motores (velocidad, potencia, etc.)
- Recepción de comandos enviados desde un cliente.
- Utiliza el mismo protocolo de comunicación que el software original RCC.

A continuación se describe en forma detallada la utilización del Servidor.

5.1.3 Configuración.

La configuración se realiza desde la agrupación "Configuración" la cual se ve en la Figura 3.

Figura 3 Control del ER-1 y cámara

Puerto del servidor: por default el puerto en el cual el servidor inicia la escucha es el 9000.

Puerto Com Motores: es el puerto que identifica el USB al cual se conecta el módulo del robot. Ver PRÁCTICA Nº 1.

Velocidad de avance: la velocidad lineal en m/s.

Velocidad de giro: la velocidad angular en º/s.

Potencia: un número entero que representa la potencia de las ruedas.

Activar servidor ER-1: activa el servidor en el puerto de escucha ya definido. Es imprescindible setear el puerto antes de activar el servidor.

Capturar Cámara: captura la imagen de la cámara del robot. La cámara debe estar conectada a la máquina, de lo contrario el programa mostrará un mensaje de error.

Transmitir Cámara: transmite la imagen de la cámara al cliente que esté conectado. La cámara debe estar conectada a la máquina.

Cerrar cliente: cierra la conexión con el cliente actual, sin desactivar el servidor.

5.1.4 Manejo local del robot.

Antes de ejecutar el Servidor ER-1 se deben conectar el robot y la cámara a los USB de la PC portátil y encender la batería del mismo. Luego, en la parte de configuración se setea el puerto COM que identifica el USB al cual se conecta el robot y se presiona el botón Ok. Además, es necesario setear las velocidades con las cuales el robot se va a mover.
En la parte inferior izquierda de la pantalla se observa una serie de botones para el manejo del robot en forma local como se muestra en la Figura 4.

Figura 4 Control del ER-1

Los botones tienen una etiqueta que indica el tipo de movimiento a efectuar, como ser: avanzar, retroceder o girar a los lados. Para detener el movimiento debe presionar el botón parar.

5.1.5 Manejo remoto del robot.

Para controlar el robot en forma remota, es decir desde un programa cliente, debemos iniciar la escucha del Servidor ER-1. Los pasos a seguir son:

a) Ejecutar el Servidor ER-1 en la PC portátil conectada al robot.
b) Ingresar el puerto COM de comunicación con el robot.
c) Ingresar el puerto de escucha.
d) Activar la casilla de verificación "Activar Servidor ER-1".

De esta forma se está en condiciones de controlar el ER-1 en forma remota, a través de una red privada o Internet, mediante un programa del tipo cliente. En la práctica 3 se detalla el uso del robot remotamente.

5.1.6 Protocolo de comunicación.

Como ya fue mencionado, el protocolo de comunicación que utiliza el Servidor ER-1 es el mismo que utiliza el software original RCC (Robot Control Center). Esto permite una gran flexibilidad, ya que un programa cliente desarrollado para comunicarse con el RCC, puede comunicarse con el Servidor ER-1 sin modificación alguna. Además, el Servidor ER-1 también acepta la comunicación desde el Telnet de Windows.
El protocolo es de la siguiente forma:

"cadena de texto"+caracter 13+caracter 10

Donde la cadena de texto es el comando que se quiere enviar. Los caracteres 13 y 10 representan el retorno de carro y el salto de línea. Como ejemplo citamos el comando para establecer la velocidad lineal, el mismo es:

set(espacio)**linear**(espacio)**velocity**(espacio)valor

Donde valor es un entero de 32 bits. El comando representa la cadena de texto del protocolo, entonces para enviar el comando al servidor sería:

"set linear velocity 150000"+chr(13)+chr(10)

Cuando se envía un comando al servidor, este responde de la siguiente forma al recibir el comando correctamente:

"Ok"+chr(13)+chr(10)

Lo cual sirve para avisar al cliente que el comando llegó de manera correcta.

5.1.7 Lista de comandos nuevos.

El Servidor ER-1 conserva los comandos de movimiento y configuración que tenía el RCC, los cuales se listan a continuación:

- move <distancia> <unidad>
- set linear velocity <velocidad>
- set angular velocity <velocidad>
- set power moving <potencia>
- stop

Además, se agregaron los siguientes comandos:

COMANDO	DESCRIPCIÓN
adelante	El robot avanza
atrás	El robot retrocede
izquierda	El robot rota en sentido antihorario
derecha	El robot rota en sentido horario

5.2 Sumario

Se reemplazó el software original del ER-1 por uno realizado en **Visual Studio.NET,** lo cual hubiese sido imposible sin la reescritura de los drivers (PRÁCTICA Nº 1). El Servidor ER-1 permite el manejo local del robot, la visualización de la cámara, la configuración de los motores (velocidad, aceleración, potencia), la comunicación con un cliente para el manejo en forma remota y la transmisión de la imagen al cliente. Se conservan los mismos comandos que tiene el RCC y se adicionan nuevos.

6 PRÁCTICA Nº 3

Autores: **Carlos Fernando Carmona, Damián De Biase, Cesar Foti**

6.1 Descripción del Cliente ER-1

6.1.1 Introducción

El Cliente ER-1 es un software desarrollado en **Visual Studio.NET** que tiene como función establecer una comunicación con el Servidor ER-1 mediante un socket, y de esta manera tener control absoluto del robot.

6.1.2 Pantalla principal

En la Figura 5 se puede observar la pantalla principal del Cliente ER-1.

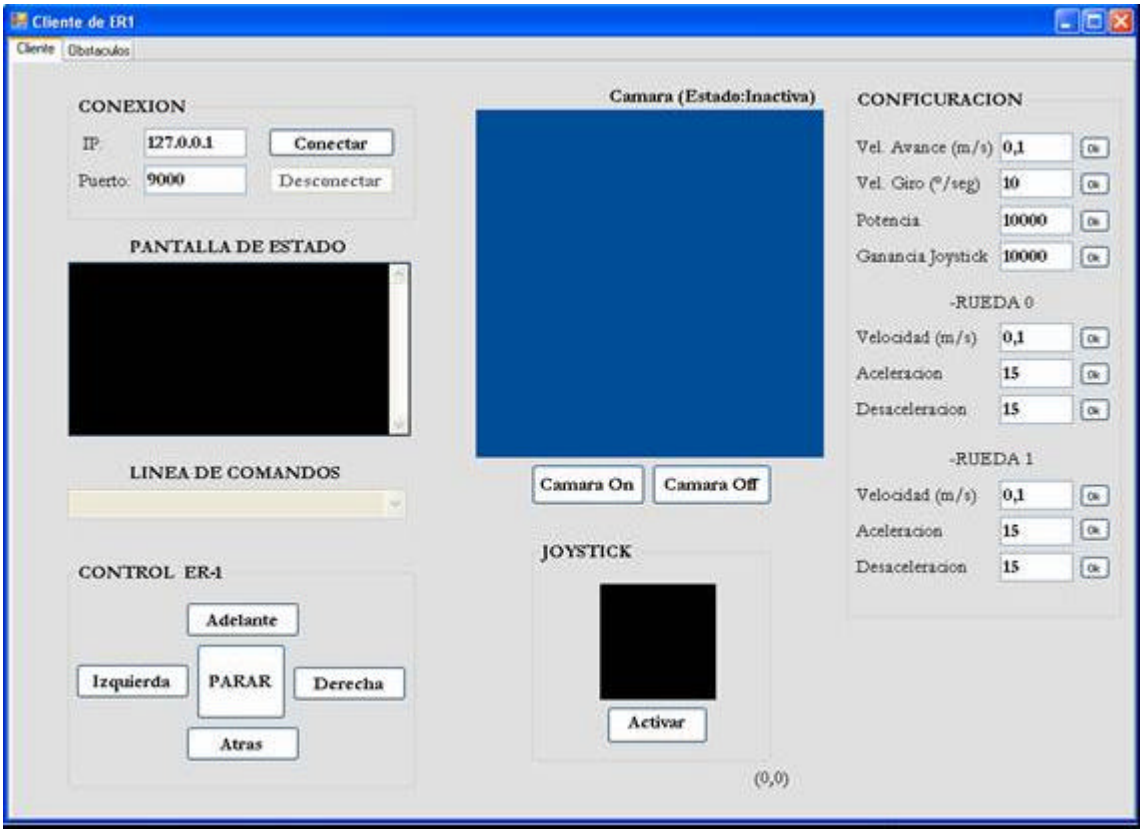

Figura 5 Pantalla principal

6.1.3 Conexión

La conexión con el Servidor ER-1 se realiza ingresando la dirección de **IP** de la PC portátil a bordo del robot y el número de **puerto** en que se encuentra "escuchando" el Servidor ER-1, luego presionar Conectar. Si la conexión se realiza con éxito en la pantalla de estado aparecerá la siguiente cadena de texto:

"Cliente conectado en IP Puerto"

IP y **Puerto** serán los correspondientes a la PC del robot.
En caso de no concretar la conexión con éxito se mostrará un aviso indicando el problema

6.1.4 Pantalla de estado.

En la pantalla de estado se reflejan todos los comandos enviados al Servidor ER-1 y las respuestas de este, como así también cada cambio en la configuración. Esta pantalla es de solo lectura.

6.1.5 Línea de comandos

La línea de comandos se activa sólo cuando estamos conectados con el servidor. Es un combo desplegable en el cual figuran todos los comandos en formato general que se pueden enviar al servidor. De esta manera, para enviar un comando al servidor se escribe el comando o se selecciona del combo y se reemplazan los argumentos de cada comando por los valores correspondientes. Para enviar pulsar la tecla ENTER.

6.1.6 Configuración

A continuación se describe cada ítem de la agrupación configuración:

Velocidad avance: es un número decimal que representa la velocidad lineal del robot en m/s.
Velocidad de giro: es un número entero que representa la velocidad angular del robot en °/s.
Potencia: representa la potencia de los motores.
Ganancia Joystick: representa cuanto varía la velocidad del robot en función de la distancia del puntero del joystick al centro del cuadrado en el que esta contenido.
Rueda 0: los valores ingresados en los campos contenidos en Rueda 0 afectan solo a esta rueda.

- Velocidad: ídem velocidad lineal pero para la rueda cero. Solo tiene efecto en el uso del comando **move**.

- Aceleración: valor de la pendiente de la recta de aceleración. Solo tiene efecto en el uso del comando **move**.
- Desaceleración: valor de la pendiente de la recta de desaceleración. Solo tiene efecto en el uso del comando **move**.

Rueda 1: ídem rueda 0 pero se aplica a la rueda 1.

6.1.7 Control ER-1.

En la parte inferior izquierda de la Figura 5 se puede observar una serie de botones para el control del ER-1. Cada vez que se presiona un botón el robot ejecuta el movimiento correspondiente con las velocidades de avance y de giro seteadas previamente. Para finalizar el movimiento se presiona Parar.

6.1.8 Joystick.

Con el joystick es posible controlar la dirección del movimiento y la velocidad del mismo en forma simultanea y continua. Para su utilización se presiona el botón Activar, instante en que aparece un circulo rojo en el centro del mismo como indica la Figura 6.

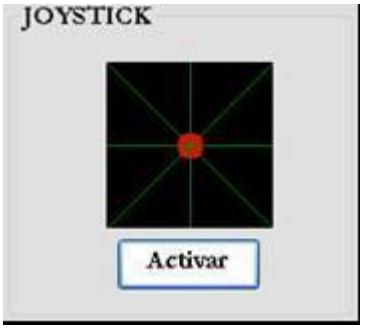

Figura 6 Joystick

Para navegar el robot, hacer clic en el centro del círculo rojo y mantener apretado el botón del mouse mientras desplaza el círculo en cualquier dirección.

- Apuntando el joystick en línea recta hacia la parte superior o inferior, el robot se mueve hacia el frente o hacia atrás respectivamente.
- Apuntando el joystick en línea recta hacia la parte derecha o izquierda, el robot gira en el lugar en sentido antihorario u horario respectivamente.
- Apuntando el joystick en cualquier otra dirección el robot se mueve describiendo una curva en la dirección que corresponda.

El robot finaliza el movimiento cuando se deja de presionar el botón del mouse.
La velocidad del robot varía en función de la distancia del círculo rojo al centro del joystick. A medida que el círculo se aleja del centro aumenta la velocidad. La proporción del cambio de velocidad en función de la distancia se ajusta desde la casilla Ganancia joystick.

6.1.9 Cámara.

En la pantalla Cámara se visualiza la imagen que se recibe desde el Servidor ER-1 y que a su vez, este la capta desde la webcam del robot. Para poder visualizar la imagen debe asegurarse que la webcam este conectada al la PC portátil ubicada en el robot y que en el Servidor ER-1 esté activa la casilla de verificación “Capturar Cámara”. Luego, presionar el botón “Camara On” para visualizar. El botón “Camara Off” corta la transmisión.

6.2 Sumario.

El Cliente ER-1 permite comandar el robot de tres formas posibles: mediante el uso del Joystick, el uso de los botones o la línea de comando. Además se puede visualizar la imagen de la cámara que está en el robot y configurar parámetros como velocidad y aceleración de las ruedas.

7 PRÁTICA Nº 4

Autores: **Carlos Fernando Carmona, Damián De Biase, Cesar Foti**

7.1 Implementación de un algoritmo de búsquedas heurísticas para evasión de obstáculos

7.1.1 Introducción

El problema que se plantea consiste en ir de un punto inicial a otro final esquivando los obstáculos que pueda haber entre estos puntos, de dos formas diferentes: realizando el camino más corto u obteniendo el camino en el menor tiempo posible (este último no necesariamente será el más corto). Para la resolución se utiliza el algoritmo A* (A estrella)[2]

7.1.2 Área de búsqueda.

El área de búsqueda esta representada en forma de grilla (Figura 7), en la cual el círculo verde representa el punto inicial y el rojo el punto final (donde debe de llegar el robot). Los círculos azules representan obstáculos. De esta forma se puede pensar en el área de búsqueda como una matriz, y cada elemento de esta es un cuadrado que estará identificado como transitable o intransitable. A partir de ahora los cuadrados de la matriz se llamaran nodos.

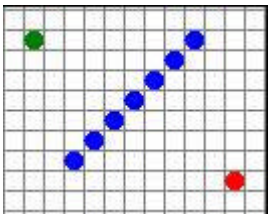

Figura 7 Área de búsqueda

[2] [Francisco Escolano Ruiz, Miguel Angel Cazorola Quevedo. Inteligencia Artificial: modelos, tecnicas y areas de aplicación, Thompson, 2003] [Aplicando un algoritmo de búsqueda: http://www.msdnaacr.net/curriculum/pfv.aspx?ID=16200665725AM] [Stuart Russell, Peter Norving. Inteligencia Artificial: un enfoque moderno, Prentice Hall, 2°edición]

7.1.3 Formato general del algoritmo

El algoritmo consiste básicamente en un *loop* que contiene las siguientes acciones:

Lista cerrada: la lista cerrada contiene los nodos que no nesecitan (por el momento) ser comprobados.

Lista abierta: la lista abierta contiene los nodos que pueden o no ser parte del camino. Son los puntos que se deben explorar.

Lista de nodos padres: proporciona los padres de los nodos que están en la lista cerrada.

Nodo Explorado: un nodo fue explorado cuando todos su nodos vecinos están en la lista abierta o en la lista cerrada

1. Se añade el punto inicial a una "**lista cerrada**".
2. Se agregan los nodos adyacentes al punto inicial a una "**lista abierta**", excepto los que están en diagonal a este, y los que fuesen obstáculos (los diagonales podrían incluirse pero en este caso el algoritmo solo tiene en cuenta los nodos a la derecha, izquierda, superior e inferior). A cada uno de estos se le asigna como nodo "padre" el nodo inicial, y se guarda esta información en una **lista de nodos padres**. Esta operación implica que el nodo inicial fue explorado (también llamada, expansión del nodo en sus sucesores).
3. En este momento, se debe quitar de la lista abierta el nodo que tenga el menor costo y añadirlo a la lista cerrada. El costo de los nodos esta determinado por una **función de evaluación.**
4. Luego se repite el proceso a partir del punto 2, pero esta vez se remplaza el nodo inicial por el último nodo guardado en la lista cerrada[3].

A continuación se describe la función de evaluación para luego entrar en detalle con la implementación del algoritmo.

7.1.4 Función de evaluación

La función de evaluación (f) es la suma de dos términos. El primero (g) evalúa la distancia recorrida desde el nodo inicial hasta el nodo actual, y el segundo (h) la distancia que queda por recorrer desde el nodo actual hasta el nodo final. De esta manera, f es función de los nodos y viene dada por:

$$f(n) = g(n) + h(n)$$

El término h(n) es la *heurística*, llamada así porque es una estimación, ya que no se conoce tal distancia hasta que no se encuentre el camino. El método usado en esta práctica para su evaluación se llama distancia Manhattan, que consiste en sumar la distancia horizontal y vertical, contando los nodos, que hay desde el nodo actual hasta el final sin tener en cuenta los obstáculos[4].

[3] Esto no es exactamente así, como se verá mas adelante, sólo se intenta dar una idea general.

[4] Para más información sobre la heurística ver Inteligencia Artificial de Stuart Russell, Peter Norving. Ed. Prentice Hall.

Resumiendo:

- g(n): se conoce, es el costo de movimiento (distancia) para ir desde el nodo inicial hasta un nodo del mapa, siguiendo el camino utilizado para llegar hasta él.
- h(n): es una estimación y representa el costo de movimiento para ir desde el nodo actual hasta el nodo final.

En esta práctica el costo para ir de un nodo a otro vecino (en forma horizontal o vertical) vale 1.

7.1.5 Descripción del algoritmo A*

El algoritmo consta de dos etapas: Inicio y Loop.

Etapa Inicio.

1) Se inicializan todas las listas y variables del algoritmo.
2) Se calculan los costos h(n) de todos los nodos de la grilla.
3) Se toma como nodo actual al nodo inicial.
4) Se guarda el nodo inicial en la lista cerrada.

Etapa Loop.

5) Se comprueban los nodos vecinos al nodo actual, esto es:
   IF no esta en la lista abierta **AND** no esta en la lista cerrada **AND** no es un obstáculo **THEN** se agrega a la lista abierta. Luego a cada uno de estos nodos se le asigna como nodo padre el nodo actual. Por último, se calcula el costo g(n) y f(n) de cada uno de estos nodos.
6) Si alguno de los nodos vecinos ya está en la lista abierta se debe comprobar si el costo g(n) del nodo es más bajo que el del camino que se esta usando para llegar a él. Si esto se cumple no se modifica nada, en caso contrario significa que el costo g(n) del camino actual es más bajo, entonces se cambia el padre que tenía el nodo por el nodo actual y se recalculan los costos f y g del nodo.
7) Se comprueba que la lista abierta no esté vacía, ya que en ese caso no hay más nodos por explorar, lo que significa que no existe camino posible. Si no está vacía se quita de la misma el nodo que tenga menor costo f, se lo agrega a la lista cerrada y se selecciona como nodo actual.
8) Comprobar si el nodo actual coincide con el nodo de llegada, lo cual significa que se encontró el camino y se debe salir del loop. En caso contrario, se vuelve al punto 5.
9) Si se encontró el camino, el mismo queda determinado por los padres de los nodos partiendo del nodo final.

7.1.6 Implementación del algoritmo

El algoritmo esta implementado en la clase Astar contenida en el software Cliente ER1. A continuación se describen brevemente los campos, procedimientos y funciones más importantes de la clase, sin entrar en detalle ya que el código fuente esta lo suficientemente comentado para su comprensión.

Estructura XYnodo: esta estructura permite manejar con gran facilidad todas las variables que se refieran a los nodos de la grilla.

```
Public Structure XYnodo
      Dim x As Integer
      Dim y As Integer
End Structure
```

Donde x e y representan las coordenadas del nodo.

Inicialización: procedimiento que debe llamarse antes de resolver el camino. Inicializa todos los campos y listas del algoritmo en función de las dimensiones de la grilla.

```
Public Sub Inicializacion(ByVal Ancho As Integer, ByVal
Alto As Integer)
```

Ancho y Alto representan las dimensiones de la grilla en unidades de nodos.

Camino: es un vector del tipo XYnodo el cual contiene el camino encontrado.

```
Public Camino() As XYnodo
```

ResolverCamino: es una función que resuelve el camino utilizando el algoritmo A*. Devuelve un string indicando la cantidad de ciclos que se realizaron para encontrar el camino.

```
Public Function ResolverCamino() As String
```

7.1.7 Utilización de la pantalla obstaculos del software Cliente ER1

La pantalla obstáculos se puede observar en la Figura 8.

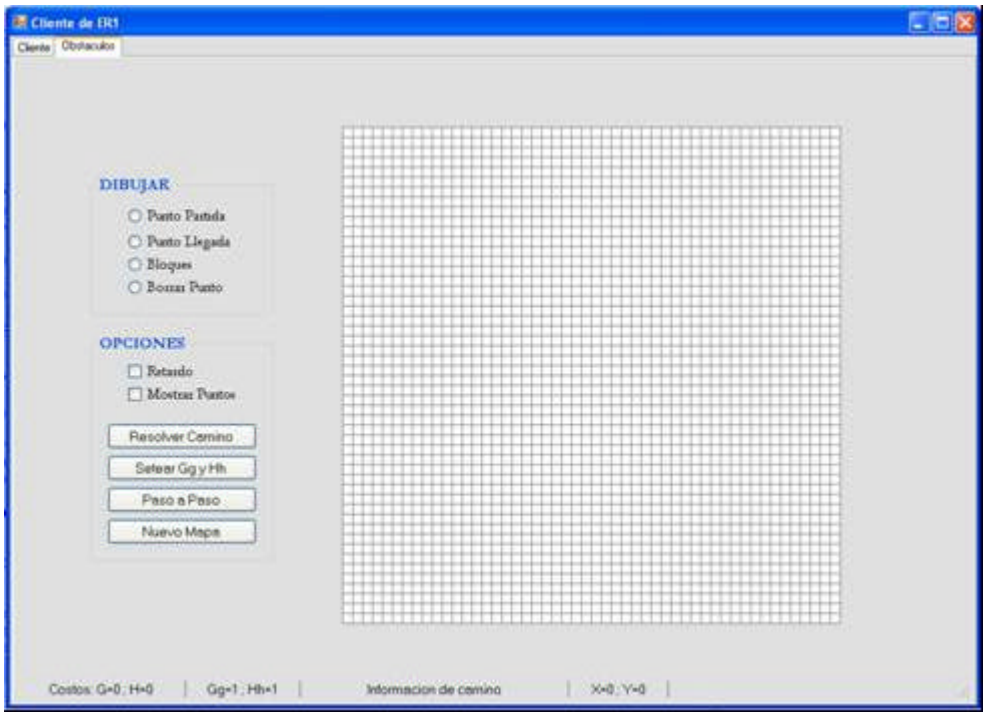

Figura 8 Pantalla obstáculos

Para dibujar se debe seleccionar desde la agrupación “Dibujar” el punto correspondiente. Luego dirigirse a la grilla y hacer click en el cuadrado que desee dibujar el punto previamente seleccionado.
En el caso de los puntos inicial y final, solo podrá dibujar uno de cada uno, por lo cual si intenta dibujar dos veces el mismo punto, el programa cambiara la posición del primer punto por la posición en la que quiso dibujar el segundo. Por otro lado, los bloques (que representan los obstáculos) pueden ser dibujados de dos formas: haciendo clic en un cuadrado y arrastrando el mouse para dibujar varios puntos o hacer click y soltar el botón para un solo punto. Para borrar bloques debe seleccionar “Borrar punto” y hacer clic sobre los bloques que desee borrar.
La casilla de verificación “Retardo” genera un retardo en la exploración del camino para poder visualizar como progresa el algoritmo a medida que este se ejecuta. También es posible mostrar los puntos que se van explorando mediante la casilla “Mostrar Puntos”.
Una vez que está cargado el mapa se presiona “Resolver Camino”, y el programa ejecuta el algoritmo en forma continúa hasta encontrar la solución (si es que existe). Por el contrario, si se presiona el botón “Paso a Paso”, el programa ejecuta un solo ciclo del algoritmo y para, hasta que

se vuelva a presionar el botón, instante en que se ejecuta otro ciclo y así sucesivamente. Esto permite al usuario observar cada movimiento del algoritmo, cuales son los puntos que se exploran, los que se guardan en la lista abierta y en la cerrada, de manera de obtener una total comprensión de su funcionamiento.
El algoritmo tiene dos formas de resolución. Una consiste en encontrar el camino óptimo, esto significa recorriendo la mínima distancia entre el punto inicial y la meta (punto final), lo cual requiere explorar una mayor cantidad de nodos (por lo tanto mas tiempo). La otra consiste en resolver el camino en el menor tiempo posible, pero como resultado la mayoría de las veces el camino encontrado no es el óptimo.
Esto se logra mediante la configuración de dos variables, Gg y Hh. Según a cual de las dos se le dé más peso, el algoritmo encontrará el camino óptimo (si Gg es mayor) o lo resolverá en menos tiempo (Hh mayor a Gg).

7.2 Ejemplo de resolución

A continuación se presenta un ejemplo detallado para la resolución de un mapa.

1. Ingresar el punto de partida del robot. Seleccionar punto de partida en la agrupación dibujar y hacer click en la coordenada (12,7) de la grilla.(Figura 9).

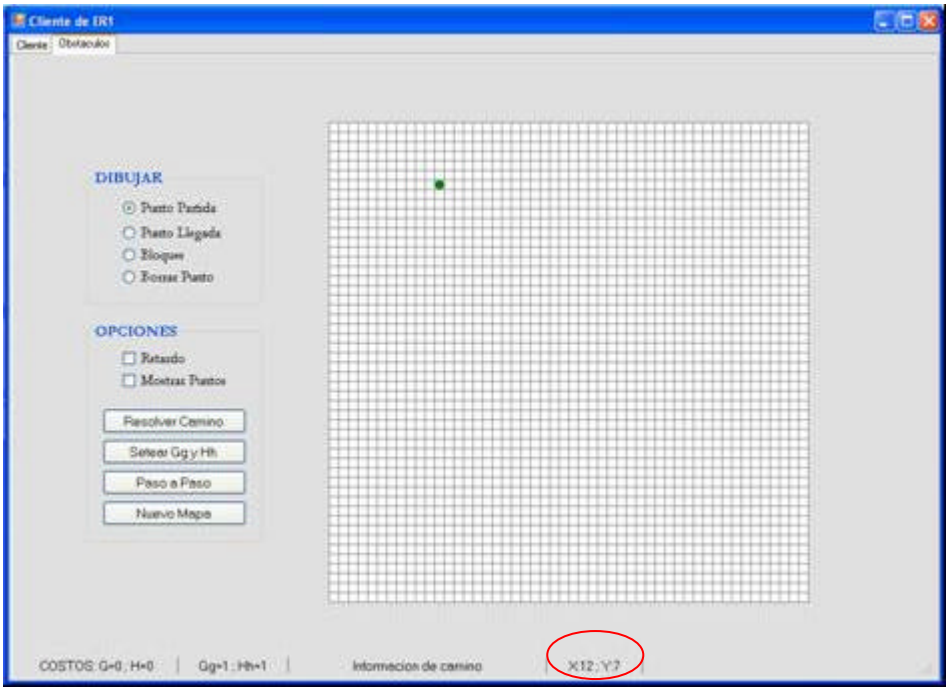

Figura 9 Punto de partida

2. Ingresar el punto de llegada del robot. Seleccionar “Punto Llegada” y hacer clic en la coordenada (43,43) de la grilla. (Figura 10)

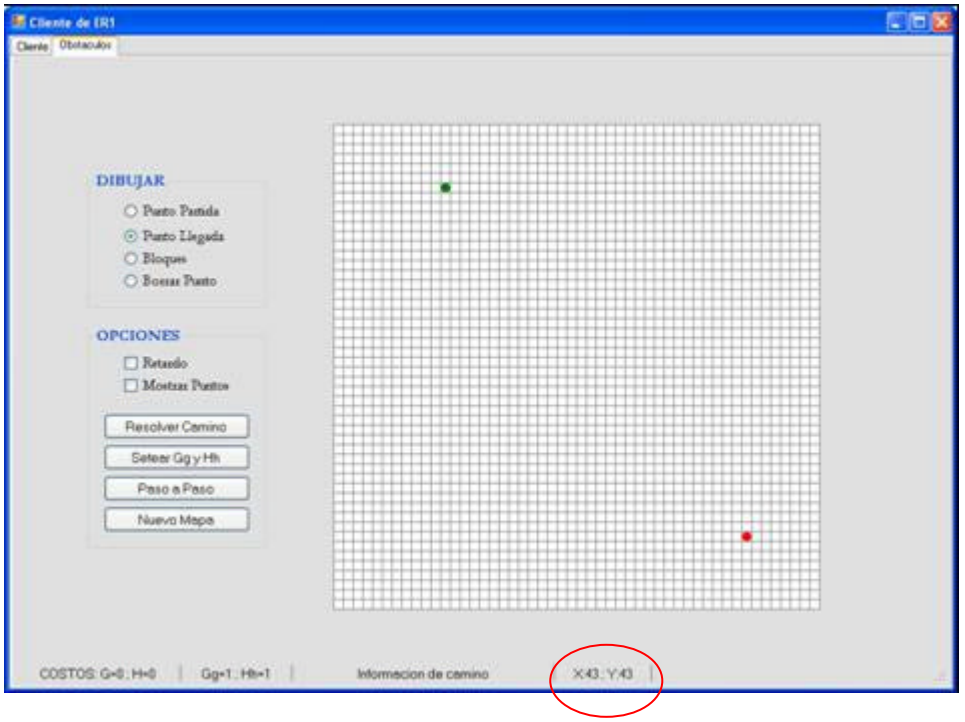

Figura 10 Punto de llegada

3. Ubicar obstáculos. (Figura 11)

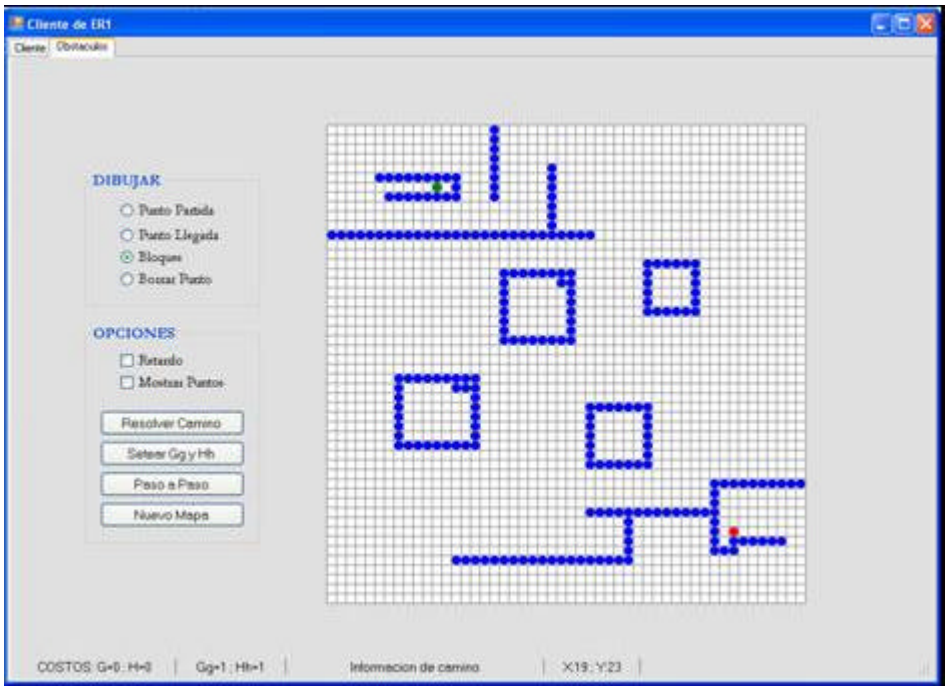

Figura 11 Ubicación de obstáculos

4. Configurar las variables Gg y Hh desde el botón “Setear Gg y Hh”, aparecerá una pantalla para ingresar los valores. En este caso elegimos 10 para Gg y 1 para Hh. Luego presionar “Resolver Camino”. La pantalla queda como muestra la Figura 12.

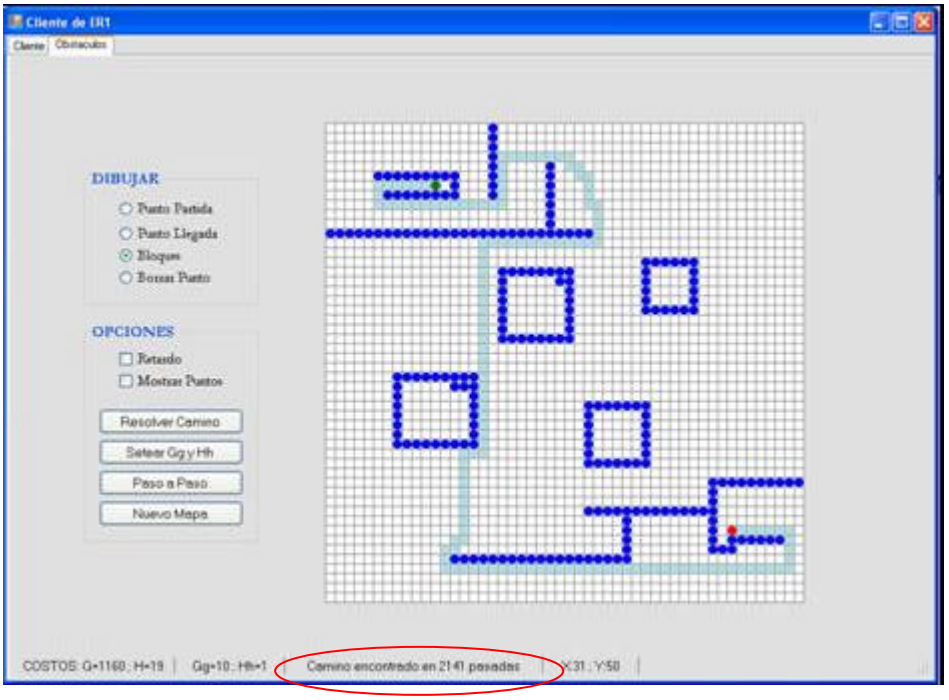

Figura 12 Camino resuelto (el más corto)

Se puede observar, enmarcado en rojo en la figura, que al resolver el camino el programa muestra en pantalla la cantidad de ciclos que ejecuto el algoritmo para la resolución. En este caso al darle mas peso a Gg, se obtuvo el camino más corto.
Para visualizar la cantidad de nodos explorados, seleccionar “Mostrar nodos”. (Figura 13)

En este ejemplo, se ve claramente que el algoritmo debió explorar casi todos los nodos de la grilla para encontrar el camino optimo.

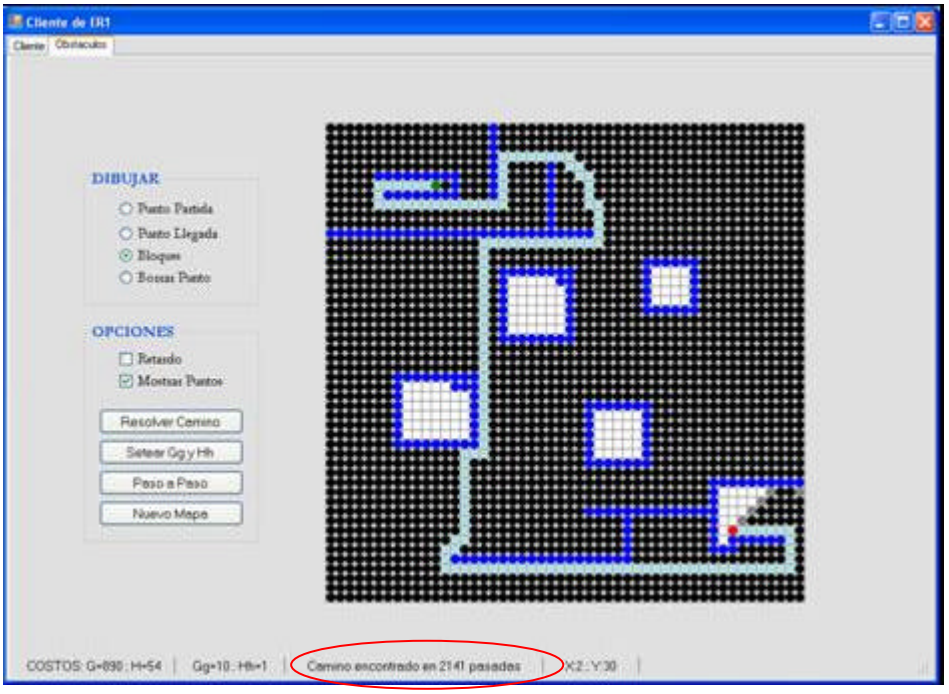

Figura 13 Nodos expandidos (para el camino más corto)

Ahora repetimos el mismo procedimiento pero esta vez se configuran Gg y Hh con lo siguientes valores: Gg=1 ; Hh=10. El camino resuelto se muestra en la Figura 14.

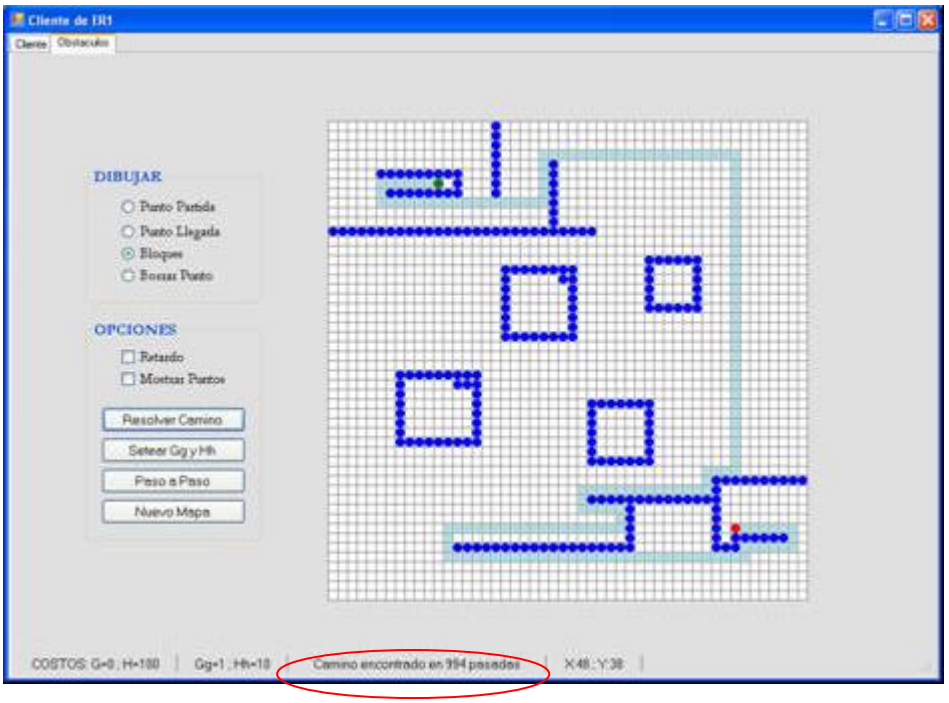

Figura 14 Camino resuelto (**no** el más corto)

Se puede ver claramente que en este caso el camino **no** es el más corto. Pero si se muestran los nodos explorados notará que la cantidad es mucho menor a la del ejemplo anterior (ver Figura 15). Además, se puede apreciar en la parte inferior de la pantalla que la cantidad de ciclos ejecutados por el algoritmo es casi 2,5 veces menor.

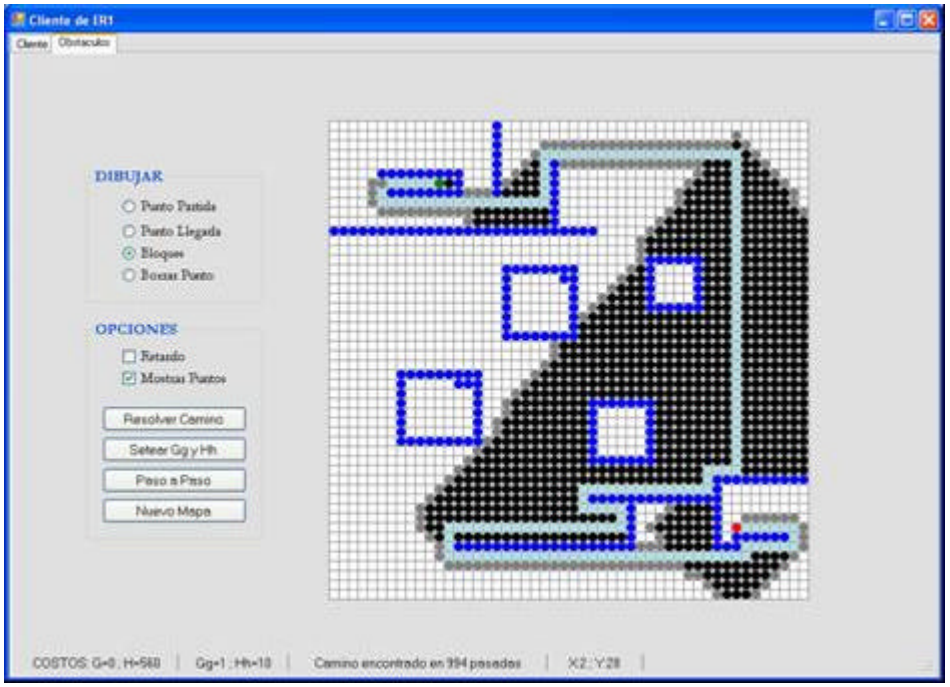

Figura 15 Nodos expandidos (**no** por el camino más corto)

7.3 Sumario

Se ha presentado hasta aquí, la implementación del algoritmo de búsqueda denominado A* y de un entorno gráfico que permite establecer, en un campo de trabajo, las posiciones de los puntos de partida y llegada así como la de diferentes obstáculos. Además se puede observar el desarrollo de la búsqueda en función del tiempo y cambiar los parámetros de la misma (Gg y Hh). En efecto, se muestra como llegar, con mucho esfuerzo de búsqueda y mayor empleo de tiempo por el camino más corto ó simplemente llegar más rápido, con menos esfuerzo de búsqueda, por un camino no óptimo.