*Cuadernos de Ingeniería. Nueva Serie.*Publicaciones Académicas
Fac. Ingeniería, Universidad Católica de Salta, vol. 9, 2016: 107-119

Diseño de un taller de introducción a la programación de robots[1]

M. Lorena Talamé - Norberto Aramayo - Ana E. Gardel - M. Alicia Pérez Abelleira[2]

Resumen

El interés por la robótica entre alumnos de la carrera de Ingeniería en Informática y la ausencia de temas directamente relacionados con la misma en el currículum, junto con la disponibilidad cada vez mayor de robots a costos accesibles, ha motivado el diseño de un taller de introducción a la programación de robots. Dicho taller permite a los alumnos incursionar en la disciplina usando tanto un robot real como un simulador, e integrar algunas habilidades ya adquiridas a lo largo de la carrera, con la motivación añadida de ver resultados al aplicarlas al control del robot. El presente trabajo describe las herramientas utilizadas y los contenidos, eminentemente prácticos del taller.

Palabras Clave: Robots móviles - Robótica educativa - Microsoft Robotics Development Studio- iRobot Create

Abstract

The increasing interest of informatics engineering students in learning about robotic and the lack of contents of this subject in currently courses, motivated the organization of the workshop. This activity let pupils to go inside the discipline, making use of a real robots and a simulator; at the same time, they can integrate knowledge acquired up to this time.
This report explains the tools used and the characteristics of the workshop, which has a primarily practical point of view.

Keywords: Mobil robots - educational robotics - Microsoft Robotics Development Studio - iRobot Create

---

[1] Este trabajo fue presentado en 2° Congreso Nacional de Ingeniería Informática/ Sistemas de Información, CoNaIISI 2014, San Luis, 13-14 Nov 2014. Publicado en *Anales CoNaIISI* 2014, eds: D. Riesco et al, Universidad Nacional de San Luis, ISSN: 2346-9927

[2] Facultad de Ingeniería e IESIING - Universidad Católica de Salta.

1. Introducción y motivación

En nuestra experiencia docente en la carrera de Ingeniería en Informática de la Universidad Católica de Salta hemos venido observando el interés de muchos alumnos por la robótica, algo atractivo pero prácticamente inaccesible en nuestro contexto. Es por eso que comenzamos a explorar la posibilidad de introducir el tema de manera extracurricular para los alumnos de ingeniería y para alumnos de colegios secundarios como elemento motivador. Dado el perfil de nuestra carrera, el énfasis estaría en la programación de robots, en lo posible de robots reales pero también, aprovechando la disponibilidad, de simuladores.

No es casualidad este interés por la robótica. Esta disciplina, que ciertamente no es nueva, se va haciendo cada vez más cercana por la extensión de las aplicaciones robóticas al ámbito del hogar y otras tareas de la vida diaria. Ya en 2007 Bill Gates predijo en un artículo en *Scientific American* titulado *Un robot en cada hogar* (gates, 2007), que la robótica sería la próxima área destacada de la informática. Por su interés, reproducimos aquí el primer párrafo de este artículo:

> Imagínense que están presentes en el nacimiento de una nueva industria. Es una industria basada en tecnologías innovadoras, en la que un puñado de corporaciones bien establecidas vende dispositivos altamente especializados para usos industriales, y un número cada vez mayor de *startups* producen juguetes innovadores, gadgets para aficionados y otros productos interesantes. Pero también es una industria muy fragmentada, con pocos estándares o plataformas. Los proyectos son complejos, el progreso es lento, y las aplicaciones prácticas son relativamente raras. De hecho, a pesar de toda la excitación y la promesa, nadie puede decir con certeza cuándo – o si – esta industria llegará a tener masa crítica. Si lo consigue, seguramente podrá cambiar el mundo.

Al decir de Gates, este párrafo bien podría haberse referido a la industria de los PCs hace ya unas décadas, pero sin embargo está hablando de la industria robótica.

Según la *International Robotics Federation* (IRF, 2013), solo en 2012 se vendieron unos 3 millones de robots para uso personal y doméstico. La mayoría de ellos, 1,96 millones, se utilizan en tareas domésticas (limpieza del suelo, corte del pasto, etc.), con un incremento del 53% sobre 2011 y ventas de US$697 millones. Los restantes son principalmente para entretenimiento (robots que son juguetes o para aficionados), educación e investigación. Las previsiones para el período 2013-2016 indicaban que se venderían unos 22 millones de unidades para uso personal, de los cuales 15 serían para tareas domésticas con un valor estimado de US$5600 millones; unos 3,5 millones, para entretenimiento, unos 3 millones para educación e investigación, y unas 6.400 unidades para asistencia a personas ancianas y discapacitadas, un mercado, este último que aumentará sustancialmente en los próximos 20 años.

Así, cada vez está más cerca un futuro en que los robots y dispositivos relacionados formen parte de la vida diaria, gracias a tecnologías como la computación distribuida, el reconocimiento y síntesis de voz, el procesamiento de imágenes, y la conectividad inalámbrica de banda cada vez más ancha. Este futuro ciertamente motiva a muchos de nuestros alumnos, aun cuando no está exento de preocupaciones éticas.

Una empresa, iRobot, iniciada por investigadores del MIT, ha fabricado y vendido desde su

comienzo más de 10 millones de robots para uso doméstico, entre ellos su conocido modelo de robot aspiradora, Roomba (iRobot; 2014). Gracias a su éxito comercial, iRobot ha podido poner en el mercado un robot con la misma arquitectura que el Roomba para tareas de investigación y educación, el Create (Figura 1). Se trata de una variante del Roomba sin los accesorios de aspirar y cepillar. El precio del mismo de US$150 (más US$100 para una batería recargable), es un costo al alcance de las posibilidades de muchos interesados en la robótica, específicamente en la programación de robots.

Figura 1. Robot Create de iRobot con cámara inalámbrica utilizado en el taller.

Es importante destacar que la programación de los robots se está convirtiendo en el principal desafío, más que el hardware. Algunos investigadores de prestigio, como Red Whittaker y Sebastian Thrun, han afirmado que a pesar de las dificultades para que el hardware funcione adecuadamente, el avance en este campo es significativo y los escollos superables; ahora la dificultad está en el software que resuelva el problema (Gates, 2007; Thrun, 2006).

La existencia del Create, por su precio y calidad, nos motivó a poner en práctica el deseo de ayudar a nuestros alumnos a incursionar en el mundo de la robótica. Tras explorar el estado de la enseñanza de la robótica y las fortalezas y debilidades de nuestro equipo y contexto, y tras conseguir una unidad de este robot, avanzamos en la posibilidad de dictar talleres de introducción a la robótica, dejando la posibilidad abierta para conseguir en el futuro un número mayor de robots, en particular del tipo Lego Mindstorms, con mayor versatilidad desde su arquitectura física y variedad de sensores y actuadores. El Create viene listo para usarse, con una variedad de sensores, lo cual nos ha permitido centrarnos en la programación más que en el hardware. No obstante como plataforma el robot permite muchas posibilidades de extensión también desde el hardware y la electrónica.

Parte de nuestra exploración fue la búsqueda de alternativas para desarrollar software para robots, y específicamente para el Create y el Mindstorms. En base a ella decidimos inclinarnos por el Microsoft Robotics Development Studio (MRDS). Los aspectos en que basamos la decisión fueron la disponibilidad de un simulador de robots en diversos contextos;la posibilidad de controlar diversos robots usando la misma interfaz junto con la existencia de librerías para los dos robots mencionados, que nos permitieron comenzar a usar el Create de inmediato; la posibilidad de programar tanto con un lenguaje visual como con un lenguaje más tradicional, C#, conocido por buena parte de nuestros alumnos; y por último la disponibilidad del entorno en forma gratuita.

La introducción a la robótica, además de ofrecer a los estudiantes un acercamiento a la robótica, les permite aplicar conceptos y técnicas de programación que han ido adquiriendo durante la carrera, y ampliar sus conocimientos en otras que tal vez solo hayan visto desde la teoría. En particular, uno de los problemas más complejos de la programación en robótica es gestionar en simultáneo todos los datos provenientes de una diversidad de sensores, y emitir los comandos apropiados a los actuadores. En un enfoque de programación convencional un programa con un solo hilo (*thread*) iría sucesivamente leyendo datos de los sensores, después los procesaría y finalmente decidiría el comportamiento del robot, para volver a comenzar el ciclo de nuevo. Este enfoque tiene obvias limitaciones para la capacidad de respuesta ante imprevistos. Así, el alumno se enfrenta con el problema y gestión de la concurrencia, un desafío que se extiende más allá de la robótica a aplicaciones que precisen orquestar de manera eficiente un código que se ejecute, por ejemplo, en diferentes servidores o procesadores al mismo tiempo.

En resumen, hemos desarrollado el taller de introducción a la robótica, y más concretamente a la programación de robots, que describimos en este trabajo, con la intención de ayudar a los alumnos a acercarse al gran potencial, presente y futuro, de la robótica, combinando técnicas ya aprendidas por ellos a lo largo de su carrera y exponiéndolos a elementos teóricos y prácticos novedosos en esta interesante y excitante tecnología emergente.

En las dos próximas secciones describimos el entorno de desarrollo elegido y el enfoque y elementos del taller propuesto, para terminar este trabajo con un análisis de las ventajas e inconvenientes de la propuesta, líneas de trabajo futuro y algunas conclusiones.

2. MRDS

Microsoft Robotics Developer Studio (MRDS) es un entorno de programación basado en .NET para construir aplicaciones robóticas. En esta sección se describen los cuatro componentes principales de MRDS: CCR (*Concurrency and Coordination Runtime*), DSS (*Decentralized Software Services*), el lenguaje de programación visual VPL, y el entorno de simulación.

Como se ha mencionado, la concurrencia es un aspecto esencial en la programación robótica. Un enfoque para gestionar la concurrencia es escribir programas multihilo, siendo esta, una de las tareas más complejas en programación. MRDS esconde esta complejidad tras una librería llamada *Concurrency and Coordination Runtime* (CCR) para gestionar tareas asíncronas ejecutadas en paralelo, diseñada inicialmente para ayudar a los programadores a aprovechar las ventajas de sistemas multinúcleo y multiprocesador (Microsoft, 2010.a). Está escrita en C# y basada en .NET. CCR incluye una clase *Dispatcher* que implementa un pool de hilos, que

pueden ejecutarse simultáneamente. Cada despachador (*dispatcher*) consta de una cola de *delegados*, cada uno de los cuales representa el punto de entrada de una tarea que puede ejecutarse asíncronamente. El despachador distribuye esas tareas entre los hilos. Cada elemento de la cola del despachador tiene un puerto, que es a su vez una cola en que se coloca la ejecución de la tarea. La tarea puede tener asociada un objeto *ReceiverTask* que consume el resultado para procesamiento posterior. Un árbitro gestiona este consumidor y lo invoca cuando el resultado esperado está disponible en el puerto.

Además de CCR, MRDS añade otro nivel denominado *Decentralized Software Services* (DSS) para combinar aplicaciones concurrentes en CCR que se denominan servicios, permitiendo además que los servicios corran en diferentes computadoras en una red. Se trata de un modelo de servicios *loosely coupled*. DSS organiza, crea y manipula servicios facilitando la creación de aplicaciones concurrentes y distribuidas. Los servicios – por ejemplo el código que lee un sensor o controla un motor – operan como procesos separados que pueden ser orquestados por otro servicio. El diseño de DSS sigue una arquitectura orientada a servicios web, en analogía con la forma en que texto, imágenes o información de diversos servidores se agrega en una sola página web. DSS utiliza el protocolo DSSP, que es similar a SOAP (Simple Object Access Protocol), utilizado para servicios web. La diferencia entre ambos es que mientras que el estado de un servicio web está oculto, el estado de un robot debe ser visible (el estado, por ejemplo, incluye la lectura de los sensores del robot, o la imagen capturada por su cámara). Así el estado de un servicio puede visualizarse u modificarse mediante un navegador web. DSSP separa el estado del servicio de su comportamiento u operaciones.

DSS permite que los servicios corran aisladamente, por lo que si un componente del robot falla, puede ser apagado o reiniciado desde la misma aplicación. Los servicios de un robot pueden también monitorearse o puede cambiarse su estado desde una ubicación remota usando la misma interfaz web.

Cuando se programa en MRDS la aplicación no reside en el robot, sino en una computadora (o distribuida entre varias). Así el robot puede ser un dispositivo relativamente sencillo que delega las tareas de procesamiento complejo (ej. visión, navegación y planificación de caminos con algoritmos sofisticados, orquestación de las tareas al nivel más alto) al hardware mucho más potente de la PC. Es posible incluso manejar grupos de robots que colaboren para realizar tareas complejas.

Las aplicaciones en C# pueden crearse utilizando la IDE Visual Studio, que tiene plantillas para crear un servicio que incluyenmuchas de las directivas y elementos necesarios para el servicio. Hemos de decir que a pesar de estas ayudas la programación en este entorno es bastante compleja. Esta complejidad queda oculta en buena parte cuando se usa el lenguaje visual VPL, que se describe más adelante.

La vinculación entre el software y el hardware se define mediante archivos *manifest*; una misma aplicaciónpuede correr en diferentes plataformas hardware cambiando simplemente el archivo *manifest*. Por ejemplo, un usuario podría programar usando comandos genéricos el movimiento de un dispositivo con tracción diferencial, luego y reemplazar el archivo *manifest* con el de cualquier robot, real o simulado, que utilice tracción diferencial. El servicio convierte automáticamente los comandos de alto nivel en instrucciones de bajo nivel específicas para ese tipo de robot. O también un servicio que precise una cámara, puede utilizar indistintamente

una cámara web, una cámara IP o la cámara del simulador simplemente cambiando el *manifest* y no el código del servicio. Esta es una ventaja de utilizar MRDS: la posibilidad de cambiar el hardware sin cambiar el código [6].

El tercer componente a destacar es un lenguaje de programación visual (VPL) y su correspondiente entorno de desarrollo. Está basado en un modelo de programación gráfica para el paradigma de flujo de datos, que modela un programa como un grafo dirigido de datos que fluyen entre operaciones. Un programa, más que una serie de comandos imperativos que se ejecutan en secuencia, es como un conjunto de trabajadores en una línea de ensamblado, que hacen su tarea tan pronto como reciben los componentes, en este caso los datos. Por ello VPL se adapta para escenarios de procesamiento concurrente y distribuido. Está dirigido a programadores principiantes, aunque otros más veteranos lo pueden aprovechar para el desarrollo rápido y prueba de prototipos de servicios. El código generado puede convertirse automáticamente en código en C# (Microsoft, 2010)

Por último, MRDS incluye un simulador, en que los usuarios pueden recrear entornos complejos y colocar en ellas robots. Hay una serie de escenas y de robots disponibles de antemano y un lenguaje para crear nuevas escenas y dispositivos simulados. El motor del simulador es capaz de simular colisiones, gravedad y otras fuerzas de manera muy aproximada a como los objetos se comportan en el mundo real. El uso del simulador facilita el acceso a la robótica a estudiantes aunque no se disponga de robots físicos suficientes.

La versión de MRDS que utilizamos es la R4 (2012) que precisa del sistema operativo Windows 7 u 8 y Visual Studio Express 2010 o superior. Con Windows XP solo puede utilizarsela versión R3 (2008). No existen versiones de MRDS para otros sistemas operativos. Todo el software está disponible de manera gratuita en la web o a través de la iniciativa académica de Microsoft, lo cual es una innegable ventaja en nuestro contexto. Como se indicó anteriormente, MRDS incluye librerías para utilizar el Create y el Lego Mindstorms, tanto con el robot real como en simulación.

3. Enfoque del taller

El taller diseñado consta de contenidos teóricos y prácticos, con marcado énfasis en este segundo aspecto. Surge como investigación de cátedra en el área de la inteligencia artificial (IA) y esto marca la orientación de la parte teórica hacia la IA y los robots inteligentes. El material introductorio de las clases teóricas está basado en el trabajo de Housten (2008) en cuanto a los elementos básicos de un robot, y en el de Murphy (2000) en cuanto a los diversos paradigmas de programación robótica (jerárquico, reactivo, e híbrido) y el proceso de diseño para construir robots inteligentes. Aunque no son obras muy recientes, son adecuadas para el nivel introductorio del taller. Para el trabajo de programación del Create con MRDS, hemos tomado como referencia la obra de Johns y Taylor (2008).

El núcleo del taller son los trabajos prácticos que los alumnos deben realizar, y que se irán describiendo en las secciones que siguen. En cada sesión se propone una actividad de programación facilitando a los alumnos un tutorial que indica los pasos a seguir; previo a ello se desarrolla un tema teórico vinculado con el práctico. En la elección de temas hemos adaptado parte de los propuestos por Warkman y Elzer (2008).

La mayor parte de los prácticos están diseñados para utilizar tanto el simulador como el robot real. MRDS permite esto con pequeños cambios en los programas creados por los alumnos, en particular con el intercambio de archivos *manifest*. Así se puede aprovechar mejor el recurso escaso que es el único robot Create disponible y los alumnos pueden continuar las prácticas en otro horario y lugar. En general se sugiere a los alumnos que trabajen en parejas, tanto por la disponibilidad del robot como por la ayuda que se pueden prestar en el aprendizaje, ya que algunas de las tareas son difíciles de implementar y de hacer funcionar en VPL, especialmente para principiantes; esto hace que el trabajo sea en ocasiones muy satisfactorio y en otras bastante frustrante.

Aunque nuestros alumnos tienen experiencia en programar en diversos lenguajes de .NET, incluyendo C#, la curva de aprendizaje de CCR y DSS es demasiado pronunciada para un taller de las características que nos proponemos. Por ello en el taller se utiliza el lenguaje VPL, aunque al final se propone a los alumnos el desafío de programar servicios en C# y se les proporcionan algunos materiales a modo de tutorial.

El hardware disponible para los prácticos incluye además del robot Create con su batería recargable, una cámara web inalámbrica, y un módulo BAM para conectar el robot mediante Bluetooth a las computadoras en que trabajan los alumnos (estas deben disponer de conexión Bluetooth o utilizar el *dongle* USB y software del que dispone el taller). También se dispone de una pared virtual diseñada específicamente para el robot Create.

3.1. Práctico I: Introducción a la robótica, al MRDS y al robot Create

El primer práctico pide a los alumnos que realicen los tres primeros tutoriales proporcionados por los creadores del MRDS y disponibles en la web (traducidos por nosotros). El primer tutorial introduce el concepto de servicios usando VPL. Los servicios son el núcleo básico de la programación robótica en nuestro entorno. Un servicio es una entidad (implementada con orientación a objetos en C#) que controla un componente hardware o software particular. Como se indicó, una aplicación robótica consiste en orquestar una serie de servicios, y combinando servicios muy sencillos se puede obtener un comportamiento complejo. En el segundo tutorial se combinan varios servicios, en particular un sensor de impacto y la tracción diferencial del robot. Los alumnos aprenden a encender y apagar los motores del robot mediante un botón, y pueden extender ese comportamiento para controlar la velocidad del robot, por ejemplo.

El tercer tutorial añade algo más de funcionalidad con los mismos elementos, haciendo que el robot deambule y, si choca con un obstáculo retroceda una pequeña distancia, gire aleatoriamente por un corto periodo de tiempo, y luego avance en la nueva dirección. Además se propone construir una nueva actividad en VPL (análoga a una subrutina) que nuclea este comportamiento. Como ya se indicó, el mismo código puede ser utilizado tanto en el simulador (para lo cual proporcionamos un escenario con un robot y diversos obstáculos, mostrado en la Figura 2) como con el robot real, simplemente modificando el archivo *manifest* utilizado. El objetivo de este práctico es familiarizarse con el funcionamiento de MRDS y VPL, en particular crear un proyecto, acceder a los servicios, y conectarlos entre sí en el paradigma de flujo de datos que sigue VPL. En la introducción a los tutoriales se explica brevemente la diferencia entre la programación basada en flujo de datos y la basada en flujo de control, que es a la cual los alumnos

están acostumbrados. Este es un aspecto que suele causar dificultades conceptuales en cuanto la complejidad de los programas va aumentando.

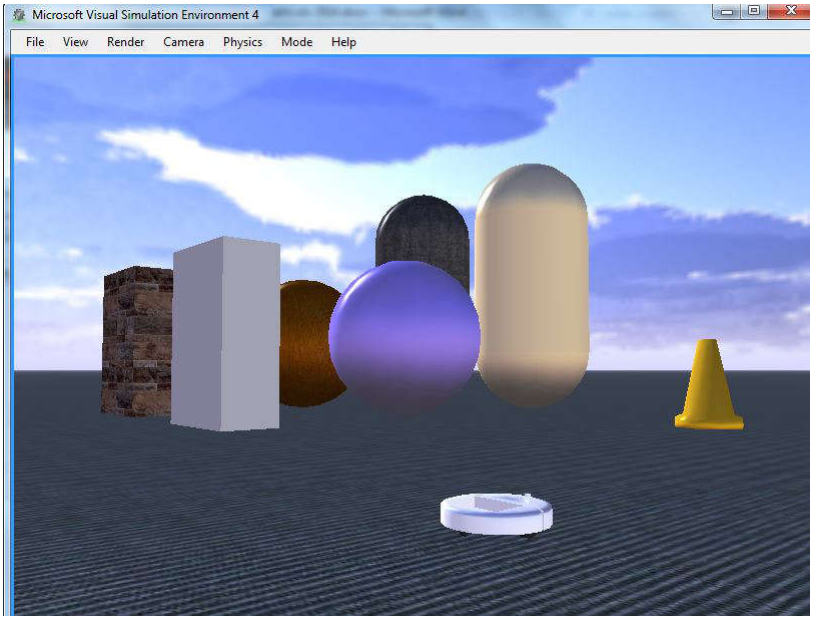

Figura 2. Escenario de simulación por defecto para el robot Create.

El práctico en sí no tiene mayor dificultad ya que los alumnos van siguiendo los pasos especificados en los tutoriales. No obstante, en base a nuestra experiencia con estas herramientas, este trabajo puede llevar tiempo considerable y encontrar dificultades complejas y sutiles para hacer que el código funcione.

3.2. Práctico II: Teleoperación

El segundo práctico toma como punto de partida un tutorial sobre teleoperación también disponible en MRDS, en el que además se debe aplicar lo aprendido en el primer práctico. El material teórico cubre el tema de teleoperación. Aunque nuestro énfasis es en robots autónomos, este es un tema importante, un punto intermedio necesario entre la mayoría de los robots industriales, programados para realizar tareas de precisión en entornos predecibles y no cambiantes, y los robots inteligentes que deben desenvolverse en entornos desconocidos y cambiantes y responder adecuadamente, sin poder ser programados de antemano para responder y adaptarse a todas las posibles contingencias (Murphy, 2000). El puente entre ambos enfoques tradicionalmente han sido mecanismos que permiten que una persona controle de manera remota el robot, o partes del mismo. A este conjunto de mecanismos, locales o remotos, suele llamarse teleoperación. Además la teleoperación forma parte de muchas de las nuevas aplicaciones de robots para la telepresencia, un conjunto de tecnologías que permiten a una persona

sentir como si estuviera presente en un lugar remoto, o dar la impresión a otros de que lo está (Lehrbaum, 2013).

En este práctico los alumnos crean una interfaz para controlar el robot remotamente y visualizan la imagen devuelta por la cámara ubicada en el robot. Así navegan un recorrido con obstáculos, pero no viendo el robot sino la imagen obtenida por la cámara montada sobre el mismo, como suele ocurrir en la mayoría de las aplicaciones de teleoperación. El práctico incluye una competencia para ver qué equipo de alumnos es más hábil en teledirigir el robot y evitar los obstáculos, algo más difícil de lo que parece.

El primer práctico constaba de sencillos ejemplos. En este se construye en base a ellos una aplicación práctica. El carácter competitivo añade motivación para que los alumnos vayan un poco más allá de lo propuesto en el tutorial, especialmente en cuanto a la interfaz de teleoperación.

3.3. Práctico III: Deambular y cubrir un recinto

Una de las tareas frecuentes de un robot es deambular recorriendo su entorno. En la instancia teórica de esta actividad, se presenta a los estudiantes la importancia de este comportamiento y ciertos enfoques al mismo (Murphy, 2000) y luego, en la práctica, los mismos pueden implementar sus propios algoritmos, en particular para conseguir que el Create cubra el suelo de la habitación de la manera más eficiente posible. Después deben añadir comportamiento que haga que el robot evite obstáculos usando los sensores disponibles en el Create (sensor de impacto y sensor de infrarrojo que detecta una pared cercana). Así pueden comprobar la efectividad, ventajas e inconvenientes de diversos tipos de sensores para una tarea específica. No existe un sensor perfecto, sino que deben aprovecharse los sensores disponibles en el robot. Tampoco puede decirse que los enfoques más complejos siempre realicen la tarea mejor que los comportamientos muy sencillos.

Hay una relación entre este práctico y una aplicación real, ya que la tarea propuesta es la básica del robot aspirador Roomba, mellizo de nuestro robot Create, con los mismos sensores y actuadores. El éxito del Roomba está en gran medida basado en la efectividad de sus algoritmos para deambular y recorrer un recinto.

3.4. Práctico IV: Ejemplo del paradigma reactivo

La tarea de este práctico está inspirada en los comportamientos biológicos para buscar comida y evitar peligros, tales como depredadores. Con esta idea se pretende mostrar los fundamentos del paradigma reactivo en robótica. Los alumnos tienen libertad para elegir cómo implementar esos comportamientos. Para ello deben utilizar la cámara web inalámbrica disponible o el simulador junto con un servicio que realiza un procesado de imágenes sencillo para detectar «comida» - pelotas de color azul - y «depredadores» - pelotas de color rojo (ver Figura 3). Este servicio ha sido desarrollado por nosotros usando AForge.NET, un framework en C# para el desarrollo de aplicaciones de procesamiento de imágenes, redes neuronales, algoritmos genéticos, robótica, etc (*www.aforgenet.com*).

Dado que nuestro robot no tiene un brazo, lo que los alumnos pueden realizar con comida y depredadores es bastante limitado. No obstante hay una variedad de respuestas que pueden

implementar, por ejemplo alejarse inmediatamente del peligro o tratar de rodearlo como si fuera un obstáculo.

Figura 3. Entorno para el práctico IV.

3.5. Práctico V: Seguidor de línea

En este práctico los alumnos enfrentan un ejemplo típico de robótica: seguir una línea marcada en el suelo. En nuestro caso se utilizan los sensores de abismo del Create, cuyo uso original es detectar desniveles, tales como escaleras. Estos sensores utilizan infrarrojos, por lo que pueden aprovechar las diferencias en las propiedades de reflexión de distintas superficies para detectar la ausencia o presencia de la línea. Así los alumnos aprecian la versatilidad, ventajas e inconvenientes de diversos sensores. Este práctico permite el trabajo tanto con el robot real como con el simulado; en el segundo caso se aprovechan los servicios y los escenarios del simulador proporcionados por Lehrbaum (2013).

3.6. Práctico VI: Contar objetos

Este último práctico, cuyo objetivo es contar los objetos existentes en un entorno, es opcional ; depende de cuánto cada alumno o equipo avance con los prácticos anteriores, ya que los conocimientos adquiridos facilitan esta tarea, que por otro lado no está exenta de dificultades, como la de saber qué objetos ya han sido contados.

Una alternativa es dejar libertad a los alumnos para atacar un problema que sea de su interés después de todas las experiencias anteriores. Aunque el hardware disponible, especialmente la variedad de sensores, es limitado, hay muchas posibilidades de temas que no son abordados en el tiempo disponible. Así los integrantes del taller, pueden integrar los conceptos anteriores y explorar algo de su propio interés.

4. Ventajas e inconvenientes del diseño del taller

En el diseño del taller se intentó hacer corresponder temas básicos teóricos sobre robótica con los trabajos prácticos. Así los alumnos pueden comprender algunos de los problemas principales de la robótica, especialmente desde el punto de vista de la programación, y aplicar en el laboratorio alguno de los principios básicos para resolverlos.

Por tratarse de un taller opcional para alumnos interesados, y no un curso formal del currículum, no se puso el énfasis en diseñar actividades de evaluación. Es difícil cuantificar la creatividad y el ingenio a la hora de resolver alguno de los prácticos. Además no siempre los esfuerzos se traducen en el comportamiento adecuado o hasta deseado del robot, lo cual puede ser resultado de las limitaciones del hardware (ej. precisión de los sensores o actuadores) y del software disponible (ej. para procesado de imágenes, o las características del lenguaje y entorno VPL).

Desde el comienzo MRDS pareció una alternativa adecuada para un taller de estas características. En cierto modo proporciona un puente entre lo que los alumnos ya conocen (sintaxis de lenguajes de programación, o hasta el entorno Visual Studio y .NET) y los elementos semánticos nuevos propios de la robótica. Otra ventaja clara fue la posibilidad de comenzar a trabajar inmediatamente con el robot disponible y con un simulador gracias al soporte y librerías incluidas en la herramienta. Esta es una ventaja importante dado que nuestro énfasis, como ya se indicó, es en la programación robótica y no tanto en la electrónica u otros aspectos de hardware de bajo nivel. La calidad y realismo del simulador, la variedad de escenarios proporcionados y la posibilidad de diseñar nuevos escenarios con relativa facilidad son aspectos muy atractivos. La disponibilidad de tutoriales y otros materiales para iniciarse en el entorno MRDS es otra ventaja adicional. La disponibilidad gratuita de las herramientas necesarias fue también un factor determinante.

Sin embargo la curva de aprendizaje para nuestro equipo ha sido bastante pronunciada. Una de las tareas arduas de la preparación del taller ha sido familiarizarnos lo suficiente con el software para poder transmitir conceptos y tecnología de manera adecuada. En el proceso hemos encontrado muchas dificultades, tales como que algunos aspectos del entorno no funcionaban como se esperaba, errores del software, falta de material en castellano y la necesidad de aprender diversos tópicos necesarios, desde la programación orientada a servicios con CCR y DSS, hasta las dificultades propias de los servicios concretos para controlar el robot.

Microsoft lanzó la iniciativa de MRDS como una propuesta de una plataforma estándar para programar robots; la versión más reciente es de marzo de 2012 e incluye amplio soporte para el dispositivo Kinect. Sin embargo no está claro el futuro de esta herramienta, el sitio web no ha variado sustancialmente desde entonces, y el interés y soporte en los foros disponibles parece ir disminuyendo (aunque las preguntas que realizamos en los mismos fueron contestadas relati-

vamente pronto por los desarrolladores de la herramienta). Uno de los factores que podrían causar el desinterés de su uso, es su complejidad, que es notable cuando se trata de extender la herramienta a otros sensores o robots diferentes de los muchos disponibles inicialmente. En teoría parece una tarea sencilla dada la cantidad de ejemplos y de implementaciones que se pueden adaptar a nuevos robots; en la práctica la complejidad es grande.

No obstante, el escenario de trabajo hardware y software que hemos descripto nos ha dado ya la posibilidad de introducir la robótica de manera accesible en nuestro contexto y con capacidad de entusiasmar a nuestros alumnos y aumentar su interés y curiosidad.

5. Trabajo futuro

A la hora de redactar este trabajo estamos a punto de ofrecer este taller por primera vez a un grupo de alumnos. La evaluación de esta experiencia, que esperamos repetir regularmente, nos dará información para trazar el camino a seguir en el futuro. Idealmente podremos aumentar el número y variedad de robots disponibles. Como se mencionó, una de las posibilidades es utilizar el Lego Mindstorms. El taller no precisaría grandes modificaciones, pero los sensores disponibles (ej. sónar) y las posibilidades de configuración del robot (por ej. como un brazo robótico) amplían el rango de posibilidades de lo que los asistentes al taller podrían llegar a diseñar y construir a lo largo del mismo.

Varios de los alumnos están comenzado proyectos de grado con la tecnología presentada y aumentándola en diversas direcciones. Uno de los aspectos a explorar es el uso de otras herramientas software que permitan el desarrollo de proyectos más avanzados, incluso en el ámbito de la investigación, y que puedan reemplazar el uso de MRDS, especialmente si éste cae en desuso o se torna obsoleto.

6. Conclusiones

La disponibilidad de un robot y especialmente de un equipo de personas interesadas en la robótica ha generado muchas expectativas ente los alumnos y profesores de la Facultad. Aunque el taller propuesto puede mejorarse en diversos aspectos, cumple las expectativas de motivar a nuestros alumnos a incursionar en la robótica y de exponerlos a esta tecnología tan innovadora en nuestro contexto pero de tanta importancia para el futuro de sus carreras y hasta de su estilo de vida.

La disponibilidad de robots de bajo costo, como el Create de iRobot, y de software de acceso gratuito para la programación de robots han hecho posible el acceso de nuestros alumnos a esta tecnología. El campo está abierto además para que los alumnos puedan llevar a cabo trabajos finales e investigaciones a nivel de cátedra, lo que genera entre nosotros grandes expectativas.

Referencias

Gates,B. «A robot in every home» *Scientific American,* Enero 2007: 58-65.

Housten, D. «Learning Roomba». MSc Thesis, Drexel University, 2008.

iRobot, «Our History,» 2014. Disponible en http://www.irobot.com/us/Company/About/Our_History.aspx. [Último acceso: 11 07 2014].

Johns, K. y T. Taylor. «Professional Microsoft Robotics Developer Studio». Wiley, 2008

Lehrbaum, R. «Attack of the Telepresence Robots!,» *Information Week,* pp. Disponible en http://www.informationweek.com/applications/attack-of-the-telepresence-robots!/d/d-id/1108137, 1 11 2013.

Microsoft. «Microsoft Robotics User Guide» 2010. [En línea]. Disponible en http://msdn.microsoft.com/en-us/library/dd936006.aspx.

Microsoft. «Visual Programming Language,» 2010a. Disponible en http://msdn.microsoft.com/en-us/library/bb964572.aspx..

Murphy, R. «Introduction to AI robotics». Cambridge, Massachusetts: The MIT Press, 2000.

RF.«World Robotics - Service Robots 2013. Executive Summary,» Disponible en http://www.worldrobotics.org/uploads/tx_zeifr/Executive_Summary_WR_2013_01.pdf, 2013.

Thrun, S. «The great robot race,» *NOVA - PBS,* p. Disponible en http://www.pbs.org/wgbh/nova/transcripts/3308_darpa.html, 28 Marzo 2006.

Workman, K. and S. Elzer. «Creating An Upper-Level Undergraduate Robotics Elective in Computer Science» de *Proceedings of the 23rd Annual Conference of the Pennsylvania Computer and Information Science Educators (PACISE)*, Kutztown, PA, 2008.

.