Uzi: Máquina virtual sobre plataforma Arduino para robótica educativa

Resultados de la primer etapa

Ricardo Moran, Gonzalo Zabala, Matías Teragni, Sebastián Blanco

Centro de Altos Estudios en Tecnología Informática
Facultad de Tecnología Informática
Universidad Abierta Interamericana
Av. Montes de Oca 745, Ciudad Autónoma de Buenos Aires, República Argentina
(+54 11) 4301-5323; 4301-5240; 4301-5248
{Ricardo.Moran, Gonzalo.Zabala, Matias.Teragni, Sebastian.Blanco}@uai.edu.ar

Resumen

El objetivo de este proyecto es el desarrollo de una máquina virtual para Arduino que ejecute un set de instrucciones definido especialmente para facilitar la enseñanza de robótica en las escuelas.

Palabras clave: Physical Etoys, Arduino, máquina virtual, lenguaje de programación, robótica educativa

Contexto

El presente proyecto será radicado en el Centro de Altos Estudios en Tecnología Informática (CAETI), dependiente de la Facultad de Tecnología Informática de la Universidad Abierta Interamericana. El mismo se encuentra inserto en la línea de investigación "Sociedad del conocimiento y Tecnologías aplicadas a la Educación". El financiamiento está dado por la misma Universidad Abierta Interamericana

Introducción

En los últimos años, la aparición de kits de robótica orientados a usuarios no expertos fomentó el desarrollo de un conjunto significativo de proyectos educativos usando robots en diferentes niveles de educación, desde jardín de infantes hasta educación de grado. La plataforma de hardware Arduino, siendo abierta y de bajo costo, se ha popularizado muy rápidamente a nivel mundial para proyectos de este estilo.

Arduino provee un entorno de desarrollo simplificado (basado en el lenguaje de programación C++) en el que muchos conceptos avanzados de programación de microcontroladores están "escondidos" al usuario. Sin embargo, este entorno es todavía demasiado complejo para algunos de los usuarios menos experimentados, sobre todo los niños más pequeños. Esto limita la complejidad de los problemas que estos usuarios pueden atacar.

Por estas razones, y aprovechando el carácter abierto de Arduino, han surgido

múltiples intentos de proveer un entorno de programación más adecuado para principiantes. Uno de estos intentos es Physical Etoys, un ambiente de programación visual para chicos diseñado para proveer mecanismos de comunicación con distintas plataformas de hardware, incluyendo Arduino. Usando Physical Etoys, un alumno puede programar robots de manera completamente gráfica, simplemente arrastrando y soltando instrucciones en la pantalla.

Sin embargo, Physical Etoys tiene un problema: dado que los programas se ejecutan en la computadora y las órdenes son transmitidas hacia el robot, Physical Etoys requiere que los robots estén continuamente conectados a la computadora para poder funcionar. Si bien la interactividad que este modelo de programación provee es muy importante (sobre todo para los alumnos que recién están comenzando), la falta de autonomía es un limitante muy importante. Por un lado, la comunicación constante con la computadora no está permitida en algunas competencias. Por otro lado, la cantidad de proyectos que pueden realizarse de esta forma es limitada (por ejemplo, quedan fuera algunas actividades de robótica situada que requieran la toma de decisiones en un entorno complejo y dinámico). Otros entornos de programación similares a Physical Etoys presentan el mismo problema (por ejemplo, Scratch 4 Arduino).

Una posible solución sería compilar los programas generados visualmente a un código nativo que pueda ejecutar el robot sin necesidad de una conexión con la computadora. Esta opción es la que eligieron otros entornos de programación (por ejemplo, Minibloq). Sin embargo, esta solución tiene sus propios problemas. Separar el "tiempo de compilación" del "tiempo de ejecución" imposibilita la forma de trabajo interactiva que caracteriza a ambientes como Physical Etoys.

Líneas de Investigación, Desarrollo e Innovación

No existe un entorno de programación visual para robótica que resuelva estos problemas, por lo tanto, creemos necesario un enfoque diferente. Los siguientes requerimientos deben cumplirse:

1. La ejecución de los programas debe hacerse directamente en el Arduino sin necesidad de interacción con la computadora (a excepción de la transmisión del programa).

2. En caso que el Arduino estuviera conectado con la computadora, todas las posibilidades de interacción que ofrece Physical Etoys deben mantenerse.

3. Para el usuario final debe ser transparente si el arduino funciona en modo "autónomo" o "interactivo".

Otros requerimientos importantes (aunque no esenciales):

4. El entorno de programación debe ofrecer mecanismos de abstracción que permitan ocultar los detalles de algunos conceptos avanzados de programación de microcontroladores (por ejemplo: timers, interrupciones,

modelos de concurrencia, etc.) Estos conceptos pueden luego ser introducidos a un ritmo compatible con las necesidades de los alumnos, pero desconocerlos no debería ser un limitante.

5. El entorno de ejecución debe ofrecer mecanismos para encontrar y arreglar errores (debugging). Particularmente, debe ser posible detener en cualquier momento la ejecución, observar el estado interno del programa, y continuar ejecutando paso a paso de forma interactiva.

Nuestra propuesta para cumplir con estos requisitos es la utilización de una máquina virtual que ejecute en el Arduino un set de instrucciones sencillo diseñado especialmente para robótica educativa. Esta máquina virtual será transmitida (compilada) al Arduino una sola vez, y a partir de ese momento las herramientas de desarrollo diseñadas especialmente para este propósito podrán comunicarse con la máquina virtual para transmitir instrucciones individuales o programas enteros a través del puerto serie.

Esta máquina virtual, además de ejecutar los programas, será la encargada de abstraer todo mecanismo de comunicación y de detectar cuando el Arduino está conectado con la computadora, en cuyo caso podrá comunicarse con la misma en forma interactiva (ya sea enviando información de los sensores y estado interno del motor de ejecución, o recibiendo comandos que modifiquen el estado de los actuadores).

Si bien no se conocen entornos que cumplan con los requisitos antes planteados, el desarrollo de máquinas virtuales y lenguajes de programación de alto nivel para microcontroladores como Arduino no es nuevo. La mayoría de los desarrollos relevados se basan en lenguajes de programación de propósito general como Java, Scheme, o Python.

Resultados y Objetivos

En esta primera etapa hemos avanzado en el diseño de un conjunto de instrucciones para robótica educativa así como el desarrollo de un prototipo de la máquina virtual encargada de ejecutar estas instrucciones. Tomando como base este desarrollo hemos definido un lenguaje de programación de alto nivel con una sintaxis inspirada en C para el cual hemos desarrollado un compilador que permite generar las instrucciones que puede entender la máquina virtual. Para facilitar el desarrollo hemos desarrollado además un editor de código simplificado.

Los objetivos para la siguiente etapa serán:

1. Validar el diseño del lenguaje y del conjunto de instrucciones usando la máquina virtual prototípica.
2. Optimizar el uso de recursos tanto de la máquina virtual como del compilador.
3. Avanzar en el desarrollo del editor para incluir herramientas de depuración de código.
4. Integrar el desarrollo con Physical Etoys

Formación de Recursos Humanos

El equipo de trabajo está conformado por un investigador adjunto del Centro de Altos Estudios en Tecnología Informática (CAETI) quien ejerce el rol de director del proyecto, dos doctorandos, y un

ayudante alumno de la Facultad de Tecnología Informática de la Universidad Abierta Interamericana.

Bibliografía

- J.E. Smith, R. Nair: “Virtual Machines: Versatile Platforms for Systems and Processes” in The Morgan Kaufmann Series in Computer Architecture and Design Series, Morgan Kaufmann Publishers (2005)
- C. Schlegel, ‘‘Communication patterns as key towards component interoperability,’’ in Software Engineering for Experimental Robotics (Series STAR, vol. 30), D. Brugali, Ed. Berlin, Heidelberg: Springer-Verlag, , pp. 183–210. Smartsoftware (2007)
- Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., Yoon, W.K.: RT-middleware: Distributed component middleware for RT (robot technology). In: International Conference on Intelligent Robots and Systems 2005 (IROS 2005), pp. 3933–3938 (2005)
- Brooks, A., Kaupp, T., Makarenko, A., Oreback, A., Williams, S.: Towards component-based robotics. In: Proc. of 2005 IEEE/RSJ Int. Conf. on Intellegent Robots and Systems (IROS’05), pp. 163–168. Alberta, Canada (2005)
- Christian Schlegel, Thomas Haßler, Alex Lotz and Andreas Steck. Robotic Software Systems: From Code-Driven to Model-Driven Designs. In procs. Of ICAR 2009. International Conference on Advanced Robotics. IEEE Press (2009)
- Bert Freudenberg, Yoshiki Ohshima, and Scott Wallace, "Etoys for one laptop per child," in 7th International Conference on Creating, Connecting and Collaborating through Computing - C5 2009, Kyoto, 2009, pp. 57-64.
- R. Rahul, A. Whitchurch, and M. Rao, "An open source graphical robot programming environment in introductory programming curriculum for undergraduates," in 2014 IEEE International Conference on MOOCs, Innovation and Technology in Education, IEEE MITE 2014, Patiala, 2014, pp. 96-100.
- A. Rao, "The application of LeJOS, Lego Mindstorms Robotics, in an LMS environment to teach children Java programming and technology at an early age," in 5th IEEE Integrated STEM Education Conference, ISEC 2015, 2015, pp. 121-122.
- M. Elizabeth and C. Hull, "Occam-A programming language for multiprocessor systems," Computer Languages, vol. 12, no. 1, pp. 27-37, 1987.
- C. L. Jacobsen, M. C. Jadud, O. Kilic, and A. T. Sampson, "Concurrent event-driven programming in occam-π for the Arduino," Concurrent Systems Engineering Series, vol. 68, pp. 177-193, 2011.
- Ryan Suchocki and Sara Kalvala, "Microscheme: Functional programming for the Arduino," in Scheme and Functional Programming Workshop, Washington, D.C., 2014, pp. 21-29.