CIMM 2011
V Congreso Internacional de Ingeniería Mecánica y III de Ingeniería Mecatrónica

PLATAFORMAS ABIERTAS HARDWARE/SOFTWARE PARA APLICACIONES EN ROBOTICA

Carlos Camargo*
*Universidad Nacional de Colombia,
cicamargoba@unal.edu.co

RESUMEN

La robótica móvil es un campo de investigación que crece rápidamente; en la actualidad existe un gran número de grupos que trabajan en diferentes áreas [1], [2], [3], [4], de sus experiencias, se concluye, que para desarrollar algoritmos aplicables en robótica, es necesario contar con plataformas físicas que permitan validar los algoritmos y modelos computacionales propuestos. El Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia ha venido trabajando en la creación y adaptación de metodologías de diseño y técnicas de fabricación de sistemas digitales [5], [6], [7], como resultado, se diseñaron y fabricaron plataformas de desarrollo abiertas que permiten ser estudiadas, utilizadas y modificadas (incluso con fines comerciales), ahorrando años de investigación a los usuarios (cualquier persona interesada) de estos conocimientos. Esto hace parte de una metodología para la transferencia tecnológica en el área de diseño de sistemas embebidos [8] que gira entorno al conocimiento como un bien común, por lo que el acceso a dicho bien es un derecho y no existe ninguna restricción para acceder a sus beneficios; este concepto novedoso va contra el modelo tradicional de considerar el conocimiento como un recurso por el que se debe pagar. En este trabajo se presentan dos plataformas de desarrollo abiertas para el estudio de sistemas embebidos con aplicaciones en robótica móvil.

Research on mobile robotics has been rapidly growing , there are many research groups currently working in this area; the experiences gained by these projects concludedes that for algorithmic development in robotics, is necessary to have hardware and software platforms to validate the proposed algorithms and computational models. The Department of Electrical and Electronic Engineering, of the National University of Colombia has been working on the development and adaptation of methodologies for design and manufacturing techniques of digital systems in the local industry, as result of this work, have been released some open hardware platforms that allow to be studied, used and modified (even commercially), saving years of research to anyone interested. This, is part of a new methodology for technology transfer in embedded system design based on the knowledge as a commons, so, the access to that resource is a right and is garanted, this new concept goes against the traditional model of knowledge as an economic resource. This paper discusses two open hardware platforms for the study of embedded systems with applications in mobile robotics.

Palabras Clave: Diseño digital (digital design), robótica (robotics), sistemas embebidos (embedded systems), transferencia tecnológica (technology transfer), conocimiento como bien común (knowledge as a commons)

1 INTRODUCCIÓN

La utilización de la robótica en la educación básica y media ha venido en aumento, y su uso adecuado permite el desarrollo de habilidades asociadas al constructivismo (aprender haciendo), y al aprendizaje significativo (aprendizaje relacionado con necesidades, intereses propios) haciendo que el estudiante aprenda el valor de la nueva tecnología y como esta puede ser utilizada para dar solución a problemas comunes. Existen dos formas de utilizar la robótica en la educación [10]: La robótica como objeto de aprendizaje: enfocado a aspectos relacionados con el robot como construcción, programación e inteligencia artificial y la robótica como herramienta de aprendizaje: visto como proyecto interdisciplinario que involucra ciencias, matemáticas, tecnologías de la información y comunicaciones.

En la mayoría de los estudios en robótica realizados en el país se utilizan plataformas comerciales provenientes del extranjero para la implementación de los algoritmos de control, esta práctica genera dependencia tecnológica hacia economías más desarrolladas y limita el campo de acción de nuestros profesionales a la integración de software y hardware. Por otro lado, los grupos de investigación están compuestos por un solo tipo de profesionales (ing. eléctrico/electrónico, de sistemas, mecánico o mecatrónico) lo que ocasiona que los problemas sean atacados desde un solo punto de vista, descuidando los otros factores: los electrónicos dedican mayor esfuerzo a la arquitectura de la plataforma electrónica que implementa la funcionalidad; los mecánicos se ocupan de la parte física; y los ingenieros de sistemas se preocupan por la programación eficiente de la funcionalidad.

Este artículo muestra un trabajo interdisciplinario realizado por ingenieros de sistemas, electrónicos y mecatrónicos; cuyo resultado es el diseño de dos plataformas robóticas abiertas fáciles de programar con gran capacidad de computo y de comunicaciones: ECBOT y SIEBOT; las cuales permiten estudiar su funcionamiento, realizar modificaciones y desarrollar aplicaciones comerciales en otros campos diferentes a la robótica; para que esto sea posible: se suministran los archivos necesarios para su fabricación, es decir, los esquemáticos y los archivos de la placa de circuito impreso (utilizando herramientas abiertas como Kicad o geda); se proporciona la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; se proporcionar el código fuente de: el programa que inicializa la plataforma (*bootloader*), la herramienta que carga dicho programa en la memoria no volátil (*usbboot*), el sistema de archivos y aplicaciones (*openwrt*); se suministra la documentación completa que indica como fue diseñada, construida, como utilizarla; y se permite el acceso a aplicaciones y tutoriales que explican el funcionamiento de los diferentes componentes y finamente se cuenta con la posibilidad de fabricación y montaje. Estas características definen el concepto *hardware copyleft, el cual, a*l ser inspirado en el movimiento FOSS, comparte su filosofía [8] y son su complemento perfecto.

En la actualidad existen plataformas comerciales que proporcionan la documentación necesaria para realizar la programación completa de sus dispositivos, pero no existen muchas que permitan realizar modificaciones en su hardware o crear nuevos productos a partir de ellas. La plataforma más completa que se acerca a los requerimientos del hardware copyleft fué desarrollada por el EPFL [3] y proporciona los archivos gerber con los que se puede reproducir la placa de circuito impreso, pero no suministra los archivos de diseño necesarios para modificarla.

2 MATERIALES Y MÉTODOS

La metodología utilizada para adquirir y absorber los conocimientos necesarios para el diseño y fabricación de estas plataformas robóticas se basa en el uso de herramientas abiertas para el diseño de las placas de circuito impreso y para la creación de la aplicación software; las plataformas de desarrollo son liberadas bajo el esquema de licencias *Creative Commons CC: BY-SA*, lo que permite su distribución, uso, reproducción y modificación incuso para fines comerciales con los requisitos de dar crédito al autor del trabajo original y que los trabajos derivados posean el mismo esquema de licencias. Esto se hace con el fin de crear una comunidad que aporte conocimientos que pueden ser utilizados sin restricción alguna por quien esté interesado. A continuación se listan las actividades que se realizaron para adquirir, absorber, asimilar y aplicar los conocimientos necesarios para el diseño, implementación y programación de las plataformas robóticas[8]:

• Elección: Identificación del estado de la plataforma tecnológica existente para identificar facilidades y necesidades; identificación de niveles de complejidad de la tecnología; selección de una alternativa que pueda implementarse y de resultados a mediano y corto plazo con no muy altas inversiones de capital.

• Adquisicion: Adquisición de plataformas adecuadas de desarrollo HW y SW; identificación de herramientas de desarrollo HW y SW y su origen (abierto/cerrado).

• Adopción: Utilización de plataformas de desarrollo para estudio de metodologías de diseño; uso de ingeniería inversa para entender la arquitectura, funcionamiento y programación de productos comerciales; utilización de productos comerciales para adaptarlos a problemas locales (Integración); tomar conciencia de la importancia del uso de esta tecnología.

• Absorción:Desarrollo o adaptación de metodologías de diseño y procesos de fabricación; desarrollo de productos tecnológicos propios; enseñanza de metodologías de diseño y procesos de fabricación en centros de educación consolidados.

• Aplicación: Desarrollo de soluciones a problemas locales; uso de metodologías de diseño en la concepción, diseño e implementación de sistemas digitales utilizando la tecnología; utilización de procesos de fabricación adaptados al entorno local; desarrollo de proyectos académicos utilizando esta tecnología.

• Difusión : Vinculación de la academia para incluir los conocimientos generados en los programas académicos de las carreras relacionadas; capacitación a la industria local sobre el uso de la tecnología, las metodologías de diseño y procesos de producción; creación de una comunidad que utilice, mejore y aumente el conocimiento generado; hacer que el conocimiento generado en los pasos anteriores este disponible a todos los interesados; dar a conocer los procesos, productos y conocimientos creados a los generadores de políticas de estado.

• Desarrollo: Aumento de la demanda de productos, bienes y servicios relacionados; compra de maquinaria que permita la fabricación masiva de forma local; diseño de nuevos componentes (Circuitos Integrados, IPs, software CAD); creación de políticas gubernamentales que protejan la producción local; participación activa de la academia en la solución de problemas y en la formulación de políticas de gobierno relacionadas.

3 RESULTADOS

El resultado de este proceso de transferencia tecnológica y de conocimientos en el área de diseño digital aplicado a la robótica, fue el desarrollo de las plataformas hardware copyleft ECBOT, SIE y dos interfaces de programación que permiten su uso a cualquier nivel de formación; la información necesaria para reproducir y modificar dichas plataformas se puede encontrar en el portal web *linuxencaja* (http://www.linuxencaja.com); adicionalmente, este portal proporciona dos canales de comunicación: listas de correo y wiki que pueden ser utilizadas por cualquier persona para solicitar soporte o para publicar proyectos abiertos relacionados con el diseño digital.

3.1 ECBOT

3.1.1 Arquitectura Software

La columna vertebral del componente Software es el proyecto Player/Stage, creado por el grupo de robótica de la University of Southern California. Este proyecto esta dividido en dos partes:

1) Player: Servidor que proporciona una interfaz flexible a una gran variedad de sensores y actuadores, utiliza un modelo cliente/servidor basado en sockets TCP lo que permitie que los programas de control del robot sean escritos en cualquier lenguaje y puedan ser ejecutados en cualquier plataforma (cliente) que posea una conexión de red con el robot (servidor). Además, soporta múltiples conexiones concurrentes de clientes, lo cual es muy útil en estudios de control descentralizado.

2) Stage: simulador escalable, trabaja con robots que se mueven, realizan operaciones de sensado en un entorno de dos dimensiones y son controlados por Player; proporciona robots virtuales que pueden interactuar con dispositivos físicos simulados. Una de las características más atractivas del proyecto Player-Stage es que permite la creación de sensores y actuadores que simulan el comportamiento de los dispositivos reales teniendo en cuenta aspectos físicos como forma, peso y material.

3.1.2 Arquitectura Hardware

Uno de los requerimientos iniciales fue la capacidad de ejecutar de forma nativa el cliente/servidor Player; para esto, es necesario que ECBOT pueda correr aplicaciones Linux; por esta razón, se diseñó la plataforma abierta ECB AT91 [5], la cual fué la base del desarrollo de ECBOT.

3.1.3 Especificaciones de ECBOT

1) Sensores y Actuadores: ECBOT cuenta con (ver Figura 1) un bus I2C encargado de realizar la interfaz con el mundo real (ya que el procesador central AT91RM9200 no posee conversores A/D), 2 microcontroladores de 8 bits (AVR de ATmel) permiten el manejo de: seis sensores infra-rojos de proximidad y luz de ambiente; 2 motores DC utilizados en tareas de evasión de obstáculos y 8 LEDs (RGB) que representan el estado interno del Robot. Para reducir el costo de los componentes ECBOT implementa un control de posición y velocidad de motores utilizando un método basado en la fuerza electromotriz que no requiere partes mecánicas. Adicionalmente,

ECBOT dispone de una cámara digital que permite la implementación de algoritmos de seguimiento de color, una FPGA se encarga de: controlar el sensor de imagen y de la comunicación con el procesador; proporcionar 10 señales digitales de propósito general que pueden ser utilizadad para el manejo de servo motores u otros actuadores.

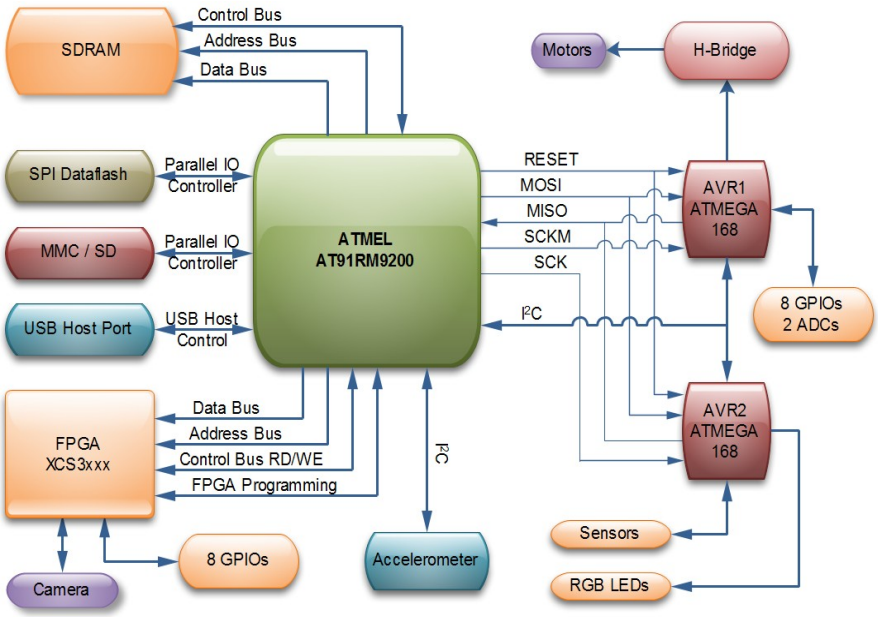

Figura 1: Diagrama de bloques de la plataforma robótica ECBOT.

Los microcontroladores de 8 bits y la FPGA son programados/configurada por el procesador central a través de pines de entrada/salida de propósito general (GPIOs), los archivos de programación/configuración son almacenados en el sistema de archivos de la plataforma y son transmitidos utilizando un enlace inalámbrico (WiFi) y pueden ser modificados en cualquier momento sin necesidad de conectores adicionales, ni de conexiones físicas con la plataforma; esto hace que ECBOT sea autónomo y que pueda ser programada “en caliente” y de forma remota.

2) Comunicaciones: Se dispone de tres canales de comunicación: un puerto USB host que permite la conexión de cualquier dispositivo *USB device*, como adaptadores USB-WiFi 802.11, modems 3G, EDGE, GSM, tranceivers ZigBee, etc. permitiendo el control y reconfiguración de forma remota; un puerto serie que en las primeras etapas del desarrollo se utiliza para cargar las imágenes del *loader*, *bootloader* e imagen del *kernel*, y se puede utilizar para conectar un GPS o un sensor con protocolo serial asíncrono; y un puerto I2C que puede ser utilizado para adicionar cualquier sensor que cumpla con este protocolo.

3) Memorias y sistema de archivos: Se cuenta con una memoria flash serial de 2 Mbytes donde se almacenan las imágenes del loader, *bootloader* y *kernel*, esta memoria puede ser modificada utilizando un *loader* que se ejecuta al inicializar la plataforma, o puede modificarse desde Linux a través de un dispositivo MTD (Memory Technology Device). Permite la utilización de una memoria RAM tipo SDRAM de hasta 64 MBytes, suficiente para ejecutar una gran variedad de aplicaciones.

ECBOT utiliza la distribución de Linux *buildroot*, basada en la librería de C *uClibc*, concebida para trabajar con sistemas embebidos, es altamente configurable y proporciona una gran variedad de aplicaciones que pueden ser instaladas siguiendo las instrucciones de un script de instalación, lo que facilita la adición de aplicaciones no soportadas por la distribución oficial; en nuestro caso se agregaron 3 aplicaciones: el servidor *player* portado a la plataforma ECBOT, el programador de microcontroladores para microcontroladores AVR *uisp* y el programador para FPGA *xc3sprog*. El sistema de archivos generado por *buildroot* y las aplicaciones necesarias para el funcionamiento de ECBOT son almacenadas en una memoria SD.

4) Unidad Central de Procesamiento: El cerebro de ECBOT es un procesador de 32 Bits de la familia ARM de ATMEL el AT91RM9200, que corre a 180 MHz. Este procesador goza de gran popularidad dentro del grupo de desarrolladores de drivers para Linux, por lo que casi todos sus periféricos están soportados.

3.2 SIEBOT

SIEBOT integra la plataforma de desarrollo para co-diseño Hardware/Software SIE [11] con una interfaz que permite el control de sensores y actuadores. La Figura 2 muestra su diagrama de bloques, en ella, podemos encontrar un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, un LCD a color de 3 pulgadas, 2 entradas y salidas de audio estéreo, 2 entradas análogas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor análogo digital de 8 canales. Existen dos canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (eliminando la necesidad de cables de programación); y otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA. El procesador utilizado es un *Ingenic JZ4725* (XBURST - MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

1) Arquitectura: Como ya se mencionó la FPGA de SIE utiliza los buses de datos, dirección y control para comunicarse con el procesador, por lo que es posible implementar un número indeterminado de periféricos en ella como, controladores de sensores, controladores de motores servo, generadores de señales PWM, interfaces de comunicaciones seriales, I2C o SPI.

2) Comunicaciones: SIE proporciona tres canales de comunicación: un puerto USB device, permite la comunicación con un computador personal a través de un puerto USB device, este canal permite la programación de la memoria NAND no volátil con el lanzador de Linux *u-boot*, la imágen del kernel y el sistema de archivos (openwrt), adicionalmente, permite el inicio de consolas remotas utilizando el protocolo *ssh*; dos puertos seriales, con niveles de voltaje RS232 uno conectado al procesador (actualmente se conecta un GPS) y otro a la FPGA 3; un puerto I2C que puede ser utilizado para conectar una amplia gama de sensores que proporcionan su salida en este formato.

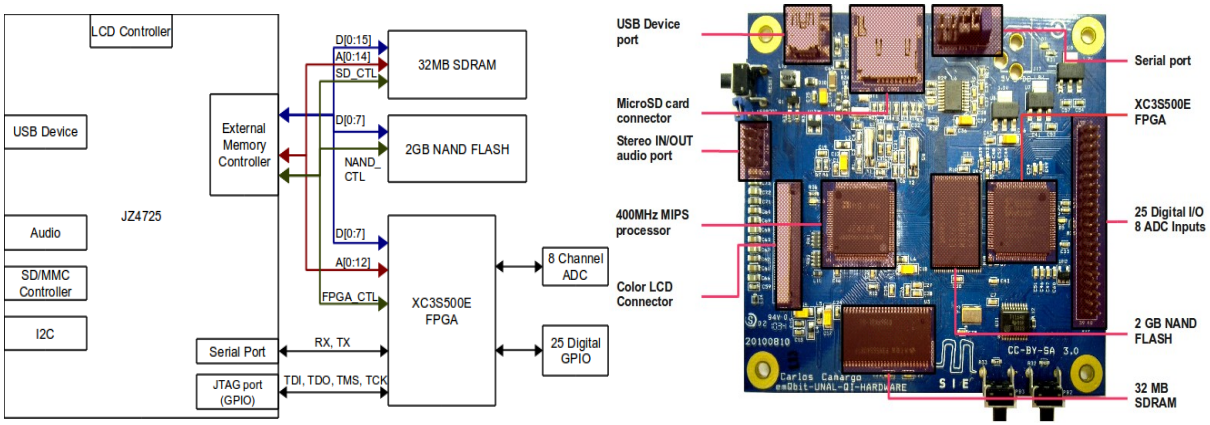

Figura 2: Diagrama de bloques de la plataforma robótica SIEBOT.

3) Sensores y actuadores: La tarjeta de sensores y actuadores soporta: un controlador puente-h para manejar dos motores de 6.8V, 2 encoders de cuadratura, 2 sensores de proximidad, una etapa de amplificación para 4 servomotores y 8 entradas análogas.

3.3 Interfaces Para Programación de Aplicaciones

Se desarrollaron dos entornos gráficos de programación abiertos con el fin de abstraer las tediosas operaciones de programación a bajo nivel y configuración de las plataformas robóticas: SIE-Blocks y SIE-CodeGenerator, desarrollados en Jave y QT respectivamente, lo que facilita su portabilidad. Estos entornos permiten la programación de aplicaciones utilizando lenguajes gráficos y su posterior descarga a las plataformas, sus simplicidad permite que sean utilizadas en diferentes niveles de formación, desde la educación básica primaria hasta la formación profesional y de esta forma difundir el uso de esta tecnología y concienciar a la sociedad de su importancia.

3.3.1 SIE Blocks:

SIE Blocks está basado en el proyecto openblocks [12] desarrollado en el MIT, el cual a su vez sigue el trabajo en entornos de programación gráficos Starlogo [13], los que buscan reducir las barreras que presentan los lenguajes de programación tradicionales al ser utilizados por principiantes. La aproximación a la programación gráfica del MIT recibe el nombre de *block programming* y en ella, los usuarios manipulan y conectan piezas de rompecabezas que representan objetos para construir programas; la silueta de estos bloques representa implícitamente la sintaxis del lenguaje y estos solo pueden conectarse con bloques con siluetas complementarias, eliminando la necesidad de memorizar reglas complejas y de esta forma ayudar a reducir la curva de aprendizaje. Esta aproximación ha demostrado ser muy efectiva aplicaciones comerciales como LEGO Mindstorm, Cricket LOGO y Labview.

En la Figura 3 se muestra una captura de pantalla de la ejecución de SIE Blocks, en ella podemos observar un menú que contiene una serie de bloques constructores separados según su función: *GPIOs*, *Functions*, *Movement*, *Loops*, *Data*, *Display* y *Math*; en la zona central se pueden colocar los diferentes bloques constructores para formar un determinado algoritmo. En la parte superior, se encuentra un menú con las operaciones que puede realizar SIEBlocks. *Load/save*:

permite almacenar y cargar diferentes algoritmos en formato XML; *connect with SIE*: establece una comunicación con la plataforma SIEBOT utilizando el protocolo ssh; y *Upload and compile*: Convierte el algoritmo a un formato XML para después ser convertido en un script que ejecutará un intérprete *Lua* que reside en SIEBOT.

Figura 3: Entorno de programación SIEBlocks.

3.3.2 SIECG

SIE Code Generator permite la creación de nuevos elementos y librerías en tiempo de ejecución, no utiliza ningún lenguaje intermedio tipo Lua o Python, genera código en C a partir de la entrada gráfica y lo compila para la arquitectura XBURST - MIPS; adicionalmente, proporciona una serie de utilidades para transferir los archivos de configuración de la FPGA y el ejecutable generado por el proceso de compilación a la plataforma SIEBOT. En la Figura 4 se puede ver una captura de pantalla.

SIE CD permite la creación de componentes y librerías que pueden ser utilizados como bloques constructores de aplicaciones; existen dos tipos de bloques básicos: software - que ejecutan una función determinada (secuencia de instrucciones) en el procesador de la plataforma, o software-hardware - que controla una tarea hardware implementada en la FPGA (periféricos dedicados); en la descripción del componente es necesario incluir el segmento de código en C que implementa la función deseada, SIE CG se encarga de unir estos segmentos de código en un solo código fuente y generar la aplicación que debe ser ejecutada en la plataforma robótica para obtener la funcionalidad requerida. En la figura 4 podemos observar: el componente software *while* equivalente a la sentencia *while (1)* de C; el componente *Frame Buffer Print Line* imprime dos mensajes de texto en la pantalla utilizando el buffer de video de Linux /dev/fb0 "*Executing Code... y CHA[0]*"; el segundo componente *ADC Single Channel* controla un periférico que maneja un

conversor serial de ocho canales, el resultado de la conversión del canal 0 se imprime en consola y se muestra en la pantalla de SIEBOT en forma numérica y en una gráfica de barras.

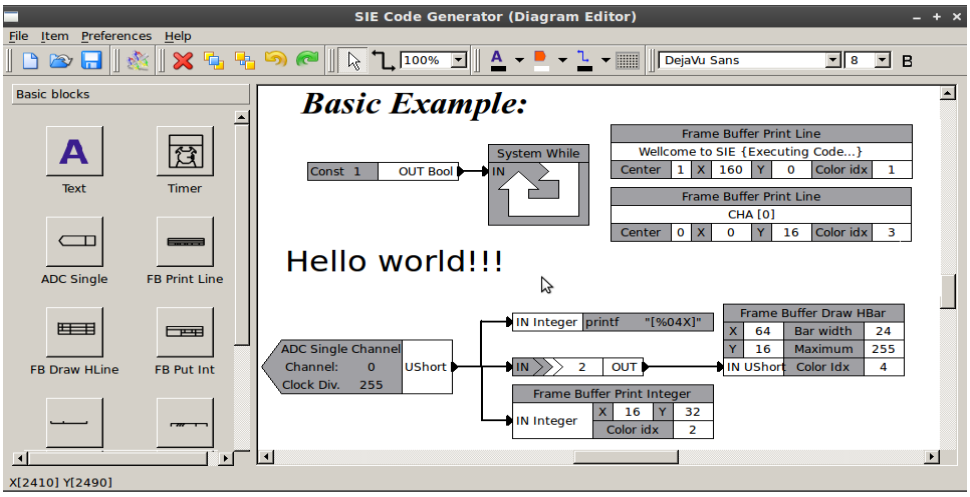

Figura 4: Entorno de programación SIECG.

4 DISCUSIÓN

Este artículo presenta dos plataformas robóticas *copyleft hardware* diseñadas, construidas y programadas como parte de un proyecto de investigación que busca difundir el diseño y construcción de sistemas digitales en el país; la información necesaria para reproducirlas y modificarlas se encuentra disponible en un servidor público y quien esté interesado puede utilizarlas en centros de formación de varios niveles, o como base de desarrollos comerciales.

Se presentaron dos entornos de programación gráfico que facilitan el proceso de aprendizaje, ayudando a adquirir habilidades en programadores principiantes relacionadas con algoritmia y pensamiento estructurado; abstrayendo la complejidad de la sintaxis en una interfaz gráfica de alto nivel.

Con estas dos plataformas se demuestra que en el país es posible realizar productos que utilizan tecnologías de punta y se encuentran en el mismo nivel de complejidad de productos diseñados y construidos en países desarrollados.

Las plataformas hardware copyleft son una herramienta poderosa en el proceso de transferencia tecnológica ya que permiten entender, adaptar, absorber y asimilar el conocimiento asociado a la tecnología necesaria para diseñar y construir sistemas digitales.

En este momento ambas plataformas se encuentran en proceso de producción en masa, y se está trabajando en el contenido de los programas académicos que se utilizarán en educación básica y media. El primer piloto se implementará en el Instituto Técnico Central 2 , en donde, se presentan los tres ciclos de interés: enseñanza media, técnica y profesional.

REFERENCIAS

[1] F. Mondada, G. C. Pettinaro, et al. SWARMBOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour, pages 307–312, University of Zurich.

[2] C. Jones and M. Mataric. Adaptive Division of Labor in Large Scale Minimalist Multi-Robot Systems. Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS). Las Vegas, Nevada, 2003.

[3] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, Floreano, D. and Martinoli, A. (2009) The e-puck, a Robot Designed for Education in Engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions.

[4] J. McLurkin. Speaking Swarmish. AAAI Spring Symposium, 2006.

[5] C. Camargo. ECBOT y ECB_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-diseño HW-SW, VIII Jornadas de Computación Reconfigurable y Aplicaciones, 2008.

[6] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos, Simposio Argentino de Sistemas Embebidos 2011, Buenos Aires Argentina.

[7] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles, IEEE Colombian Workshop on Circuits and Systems, Bogotá 2007.

[8] C. Camargo. Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft, VI Congreso Internacional de la Red de Investigación y Docencia en Innovación Tecnológica RIDIT 2011.

[9] R. M. Stallman. The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution. O'Reilly and Associates, 1999.

[10] D. Alimisis and C. Kynigos. Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods. School of Pedagogical and Technological Education (ASPETE), 2009.

[11] C. Camargo. SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales. XVII Workshop de Iberchip, Bogotá Colombia, February 2011.

[12] Ricarose Vallarta Roque. OpenBlocks : an extendable framework for graphical block programming systems. Master's thesis, MIT, 2007.

[13] M. Resnick. Starlogo: An Environment for Decentralized Modeling and Decentralized Thinking. Conference Companion on Human Factors in Computing Systems. New York, NY, USA: ACM Press, 1996.