Plataformas Roboticas Abiertas Como Herramienta de Difusión Tecnológica

Carlos Iván Camargo – Universidad Nacional de Colombia
cicamargoba@unal.edu.co

***Abstract*— La robótica móvil es un área que atrae muchos interesados con diferentes niveles de formación e intereses a lo largo del mundo; países en vía de desarrollo como Colombia, no son la excepción y muchos grupos de investigación y centros de formación a nivel básico, medio y profesional realizan actividades con estos dispositivos para aplicaciones en diferentes áreas del conocimiento. Sin embargo, en estos países existen muy pocos desarrollos de plataformas propias y la mayor parte es importada de países más industrializados. En este artículo se presentan dos plataformas robóticas abiertas que pueden ser utilizadas en diferentes niveles de formación como herramientas pedagógicas en diferentes áreas y como punto de partida de desarrollos comerciales; están compuestas por un componete hardware que permite la ejecución de aplicaciones de libre distribución como el sistema operativo Linux y un componente software que le permite al usuario trabajar en la solución de un problema determinado a alto nivel ocultando el tedioso manejo a bajo nivel de los sensores, actuadores y protocolos de comunicación asociados a las plataformas robóticas. El interés principal al liberar estas herramientas es la difusión del conocimiento adquirido durante su realización y de esta forma motivar al mayor número de personas posible a que construyan dispositivos digitales que den solución a problemas locales.**

***Index Terms*— Sistemas Embebidos, Robótica Móvil, Linux, apropiación y difusión tecnológica.**

I. Introducción

La robótica Móvil representa un campo de investigación que crece rápidamente; en la actualidad existe un gran número de grupos que trabajan en el desarrollo de aplicaciones en diversas áreas, [1], [2], [3], [4], [5], [6], [7]. En países en vía de desarrollo como Colombia, una gran parte de estas investigaciones utilizan productos comerciales o integran componentes electrónicos y mecánicos genéricos para construir plataformas de experimentación; evitando el desarrollo local de hardware especializado; esto en gran parte se debe a la carencia de una plataforma tecnológica que permita el desarrollo de aplicaciones propias y al atraso que presenta a industria electrónica en dichos paises.

Gracias a los avances de la industria de los semiconductores, a la invasión originada por los dispositivos digitales en las actividades humanas, y a las facilidades que ofrece la industria manufacturera asiática, se ha cambiado de forma dramática el escenario mundial en lo relacionado con el diseño y desarrollo de aplicaciones digitales, pasando de un modelo de total protección y confidencialidad a uno donde se proporciona toda la información y las herramientas necesarias para el desarrollo de dichas aplicaciones. Esto unido a la disponibilidad de aplicaciones software de libre distribución, disminuye de forma considerable la inversión necesaria para crear empresas de base tecnológica y de esta forma reducir la dependencia de paises en vía de desarrollo del mercado asiático.

Un estudio reciente [8] desarrollado en la Universidad Nacional de Colombia, mostró que una de las causas del atraso tecnológico en Colombia es la utilización de metodologías de diseño y tecnologías obsoletas por parte de la industria electrónica, originada en gran parte por la falta de profesionales con las habilidades necesarias para desarrollar productos comercializables que utilicen tecnología de punta, la fuerte dependencia hacia el mercado asiático, y la falta de políticas gubernamentales que protejan las industrias locales; todo esto afecta de forma considerable el mercado laboral de profesionales en el área y los obliga a aceptar salarios muy bajos o desempeñarse en actividades para las cuales no fueron fromados. Sin embargo, es posible realizar una transferencia tecnológica exitosa en el diseño de sistemas digitales, desarrollando una serie de actividades que ayuden a difundir los conocimientos asociados a dicha tecnología a diferentes niveles de formación. En [9] se muestra la experiencia al modificar la forma tradicional de enseñanza de diseño de sistemas digitales, (utilizando tecnologías obsoletas y metodologías de diseño basadas en la experiencia del diseñador sin ayuda de herramentas CAD) introduciendo metodologías de diseño aceptadas internacionalmente [10], herramientas hardware y software copyleft [11] (las que proporcionan toda la información necesaria para reproducirlas, modificarlas y programarlas utiizando herramientas abiertas) y una metodología basada en el proceso de diseño de un dispositivo digital que da solución a un problema local [9]. Como resultado, se presentaron trabajos finales que superan en calidad a los presentados antes de realizar estos cambios y se percibe en los estudiantes una necesidad de crear productos que puedan convertirse en desarrollos comerciales, lo que puede ser la base de generación de empresas de base tecnológica.

A. La robótica en la Educación

La utilización de la robótica en la educación básica y media ha venido en aumento, y su uso adecuado permite el desarrollo de habilidades asociados al constructivismo (aprender haciendo), 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 en dar solución a problemas comunes. Existen dos formas de utilizar la robótica en la educación [12]: 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 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; ya que para que esto sea posible, es necesario suministrar los archivos de diseño de la placa de circuito impreso, la lista de materiales y utilizar componentes que se puedan adquirir sin firma de acuerdos de confidencialidad, así como el código fuente del software necesario para su funcionamiento. La plataformas más completas que puede ser considerada como *copyleft* hardware fué desarrollada por el EPFL [6] 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.

En este artículo presentamos ECBOT [13] [14], (concebida para uso universitario) y SIEBOT [11] (concebida para educación básica, media y profesional) como plataformas robóticas para la educación; su carácter abierto permite estudiar su funcionamiento, realizar modificaciones y desarrollar aplicaciones comerciales en otros campos diferentes a la robótica. Adicionalmente, pueden ser utilizada en la enseñanza de metodologías de diseño de sistemas digitales [11] [9] proporcionando una herramienta que permita la transferencia de conocimiento en esta área. Estas plataformas encajan en la definición del hardware *copyleft* y suministran toda la información necesaria [1] para que puedan ser estudiadas, programadas, y modificadas (incluso para fines comerciales) por cualquier interesado.

II. ECBOT

ECBOT está compuesto por el componente hardware y el componente software, el primero constituye el dispositivo físico mediante el cual se interactúa con el entorno y sobre el cual el componente software implementa un determinado algoritmo.

A. Arquitectura Software

La columna vertebral del componente Software es el proyecto Player/Stage, el cual, es el resultado de un proyecto de investigación del Grupo de Investigación en Robótica de la University of Southern California. Este proyecto esta dividido en tres partes:

1) **Player:** Servidor que proporciona una interfaz flexible a una gran variedad de sensores y actuadores, utilizando un modelo cliente/servidor basado en sockets TCP; permitiendo su uso a trevés de una interfaz de programación de alto nivel, lo que permite 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, Player 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 y realizan operaciones de sensado en un entorno de dos dimensiones y son controlados por Player; proporciona robots virtuales los cuales interactúan con dispositivos físicos simulados. Una de las caracteristicas 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.

[1] http://wiki.linuxencaja.net

B. Arquitectura Hardware

Uno de los requerimientos iniciales fué 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* [15], la cual fué la base del desarrollo de ECBOT.

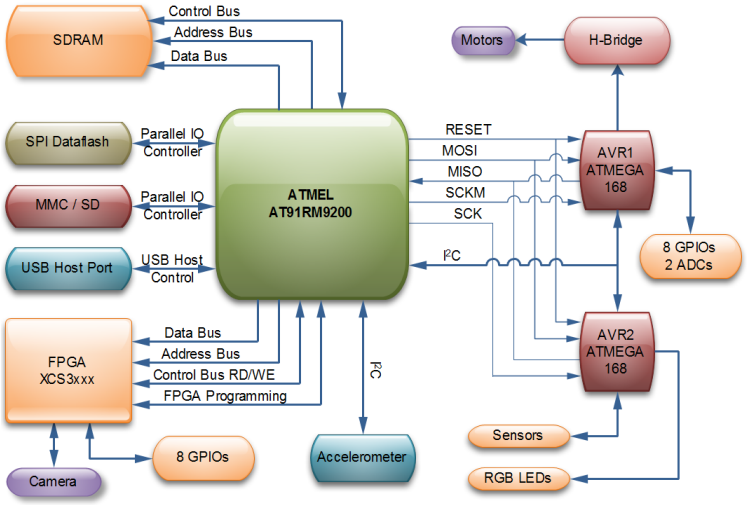


Fig. 1. Diagrama de Bloques del componente Hardware de ECBOT.

C. Especificaciones de ECBOT

1) Sensores y Actuadores: ECBOT cuenta con (ver figura 1) un bus I2C encargado de realizar la interfaz con el mundo análogo (ya que el procesador central AT91RM9200 de ATmel no posee conversores A/D), 2 microcontroladores de 8 bits (AVR de ATmel) permiten el manejo de: seis sensores infrarojos de proximidad y luz de ambiente, y 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 de medición de la velocidad basado en la fuerza electromotriz [16], [17]. Adicionalmente, ECBOT dispone de una cámara digital que permite la implementación de algoritmos de seguimiento de color, una FPGA se encarga del manejo del sensor de imagen y de la comunicación con el procesador. Adicionalmente, la FPGA proporciona 10 señales de propósito

general que pueden ser utilizadas como señales digitales de propósito general o para el manejo de servo motores.

Los microcontroladores de 8 bits y la FPGA son programados por el procesador central a través de pines de Entrada/Salida de propósito general, los archivos de programación/configuración son almacenados en la plataforma, pueden ser transmitidos utilizando un enlace inalámbrico y pueden ser modificados en cualquier momento sin necesidad de conectores adicionales; esto hace que ECBOT sea una plataforma independiente y que pueda ser programada "en caliente" y de forma remota.

2) Comunicaciones: Se dispone de tres canales de comunicación: *1 - puerto USB host* - es posible conectar cualquier dispositivo USB, como adaptadores USB-WiFi 802.11, modems 3G, EDGE, GSM, tranceivers ZigBee, etc permitiendo el control y reconfiguración de forma remota; *2 - puerto Serie* En las primeras etapas del desarrollo se utiliza para cargar las imágenes del loader, bootloader y kernel, y se puede utilizar para conectar un GPS o un sensor con protocolo serial asíncrono; y *puerto I2C* El bus I2C se encuentra disponible y puede ser utilizado para adicionar cualquier sensor que cumpla con su 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*, la cual esta basada en la librería de C *uClibc*, concebida para trabajar con sistemas embebidos; es altamente configurable y proporciona una gran variedad de aplicaciones, las 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 programas: el servidor Player portado a la plataforma ECBOT, el programador de microcontroladores para AVR: *uisp* y el programador para FPGAs: *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 la totalidad de sus periféricos están soportados. La información necesaria sobre el *port* de Linux a la plataforma se puede encontrar en [18]

III. SIEBOT

SIEBOT integra la plataforma de desarrollo para codiseño HW/SW 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 stereo, 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, permitiendo la ejecución de una gran variedad de aplicaciones Linux.

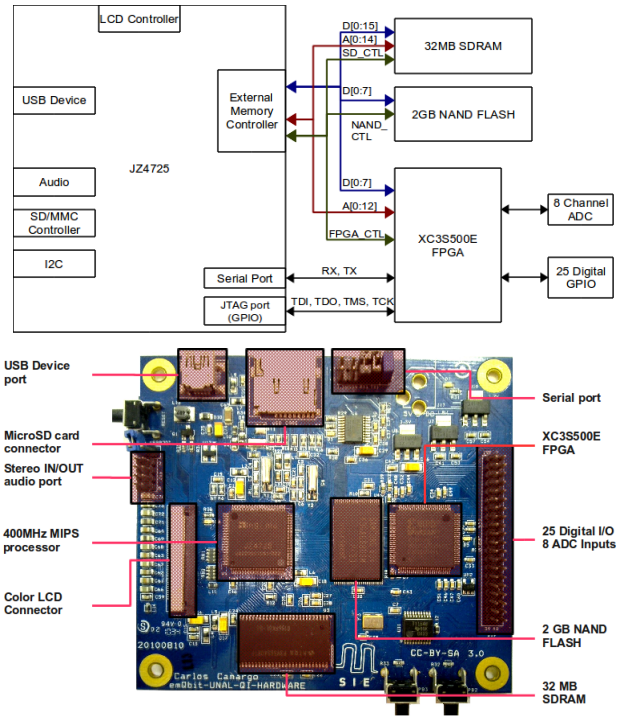


Fig. 2. Estructura de la plataforma de desarrollo SIE

5) 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 perifericos en ella, los cuales pueden ser consultados en cualquier momento por el procesador con operaciones de lectura y escritura a un rango de memoria determinado por el diseñador. Esto permite la utilización de una gran cantidad de periféricos útiles en aplicaciones en robótica como sensores (que utilizan puerto serie o I2C), controladores de motores servo, generadores de señales PWM para control de motores, y otros actuadores, etc.

6) Comunicaciones: SIE proporciona tres canales de comunicación: *1 - puerto USB device* - permite la comunicación con un computador personal a través de un puerto serial 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*; *2 - puertos serie* - se dispone de dos puertos seriales con niveles de voltaje RS232 uno conectado al procesador (actualmente se conecta un GPS) y otro a la FPGA *3 - puerto I2C* - puede ser utilizado para conectar una amplia gama de sensores que proporcionan su salida en este formato.

7) 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.

IV. Interfaces para Programación de Aplicaciones

En esta sección se presentarán dos entornos de programación abiertos, desarrolladas con el fín de abstraer las tediosas operaciones de programación a bajo nivel y configuración de la plataforma robótica SIEBOT (también es posible utilizarlas como herramientas de desarrollo de aplicaciones para ECBOT). *SIE Blocks* y *SIE Code Generator* permiten la realización de aplicaciones utilizando lenguajes gráficos y permiten descargarlas a la plataforma de forma automática utilizando como canal de comunicación una interfaz de red USB; dejando al usuario la programación de la funcionalidad. Las dos aplicaciones requieren el archivo de configuración de la FPGA en formato .bit y generado por las herramientas de síntesis de Xilinx, con la implementación de los periféricos necesarios para manejar los sensores y actuadores; dependiendo del nivel académico y de la aplicación se pueden suministrar diferentes archivos de configuración para la FPGA; sin embargo, el código fuente en lenguaje de descripción de hardware se encuentra disponible junto con un tutorial explicando su principio de funcionamiento, para que usuarios más avanzados los puedan ajustara sus requerimientos.

A. SIE Blocks

SIE Blocks está basado en el proyecto openblocks [19] desarrollado en el MIT, el cual a su vez sigue el trabajo en entornos de programación gráficos *Starlogo* [20] los cuales buscan reducir las barreras que presentan los lenguajes de programación tradicionales al ser utilizados en 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.

1) Principio de Funcionamiento: En la Figura 3 se muestra una captura de pantalla de la ejecución de SIE Blocks, en ella podemos obervar un menú que contiene una serie de bloques constructores separados según su función: *GPIOs, Functions, Movement, Loops, Data, Display y Math*, (es posible modificar estos bloques básicos ya que se suministra el código fuente de la aplicación [19]); en la zona central se pueden colocar los diferentes bloques constructores para formar un determinado algoritmo.

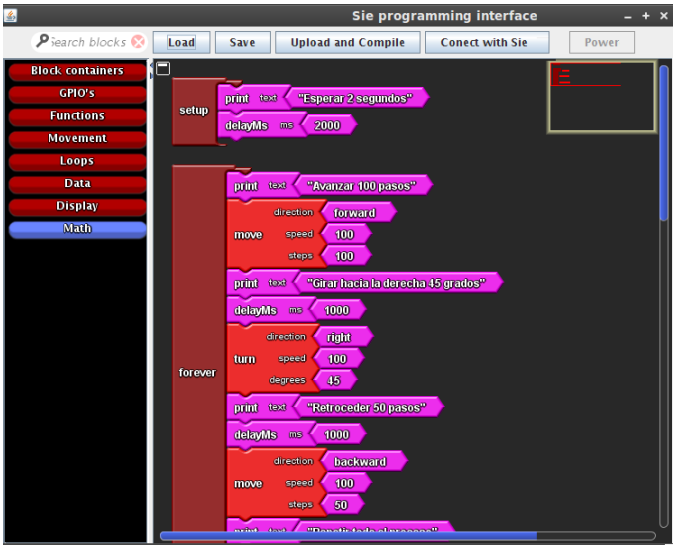


Fig. 3. Interfaz gráfica de SIE Blocks

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. La figura 4 resume el flujo de diseño al utilizar SIEBlocks.

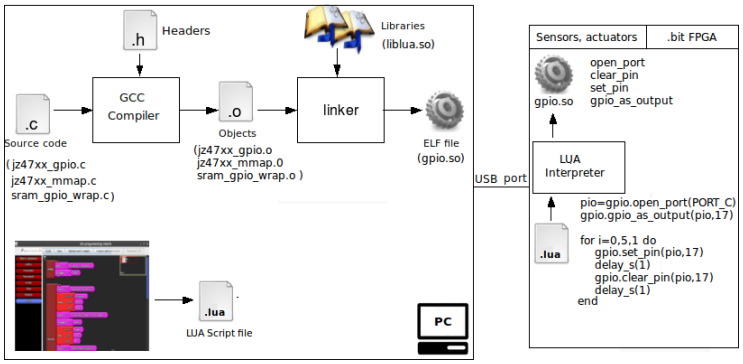


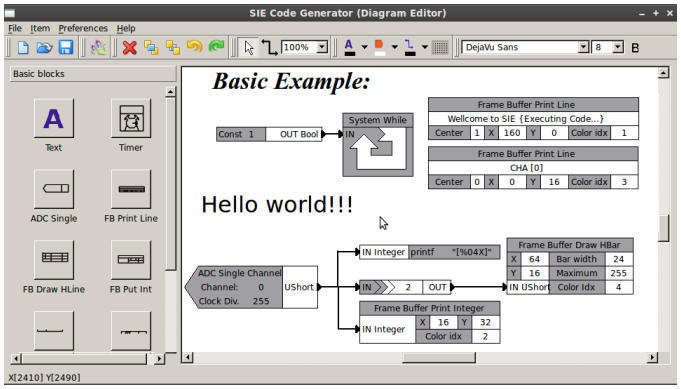
Fig. 4. Flujo de diseño con SIEBlocks

El intérprete de Lua se encuentra en SIEBOT y recibe como entradas un script donde se indican las instrucciones a ejecutarse; para realizar operaciones sobre los sensores y actuadores se proporciona una librería (*gpio.so* en este ejemplo) escrita en C, que contiene las funciones necesarias para manejo de memoria (comunicación con periféricos *jz47xx_mmap.c*) y control de pines de entrada/salida de propósito general

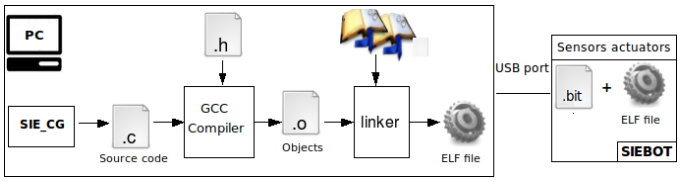
(*jz47xx_gpio.c*), así como un archivo que encapsula estas funciones y permite su uso por parte del intérprete de Lua (*sram_gpio_wrap.c*) en este archivo se encuentran el nombre de las funciones que se utilizarán en el script de Lua (*open_port*, *clear_pin*, *set_pin*, y *gpio_as_output*).

B. SIE_CG

SIE Code Generator es desarrollado en QT y está inspirado en el simulador *qucs*; 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 figura5(a) se puede ver una captura de pantalla y en la 5(b) se muestra el flujo de diseño hasta obtener el ejecutable final.



(a) Interfaz gráfica de SIE_CG



(b) Flujo de diseño con SIE_CG

1) Principio de funcionamiento: 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 (genérico) en C que implementa la función deseada, SIE_CG se encarga de unir estos segmentos de código en un solo código fuente (ver figura 5(b)) y generar la aplicación que debe ser ejecutada en la plataforma robótica para obtener la funcionalidad deseada. En la figura 5(b) podemos obervar 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.

V. Conclusiones y Trabajo Futuro

Este artículo presenta dos plataformas robóticas *copyleft hardware* que utilizan el sistema operativo Linux y pueden ser utilizadas 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 sintáxis en una interfaz gráfica de alto nivel.

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.

Este tipo de trabajos ayuda a difundir el conocimiento adquirido en años de investigación en procesos de fabricació y metodologías de diseño en sistemas digitales, proporcionando un ahorro de tiempo y dinero considerable a quien lo utilice.

VI. Bibliografía

References

[1] F. Mondada, G. C. Pettinaro, I. Kwee, A. Guignard, L. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. SWARM-BOT: 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, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.

[2] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviours. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour*, pages 11–22, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.

[3] 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.

[4] Jones and Maja J Mataric. *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, chapter Behavior-Based Coordination in Multi-Robot Systems. Marcel Dekker, Inc, 2005.

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

[6] W. Liu and A. Winfield. Open-hardware e-puck Linux Extension Board for Experimental Swarm Robotics Research. Microprocessors and Microsystems. Microprocessors and Microsystems, In Press, 2010.

[7] C. Camargo. Control de Sistemas Paralelos Inspirado en la Naturaleza. *Colombian Workshop on Circuits and Systems*, 2009.

[8] C. Camargo. Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft. *XVII Workshop de Iberchip, Bogotá Colombia*, February 2011.

[9] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Simposio Argentino de Sistemas Embebidos*, 2011.

[10] Luis Alejandro Cortés. *Verification and Scheduling Techniques for Real-Time Embedded Systems*. PhD thesis, Linköpings universitet Institute of Technology, 2005.

[2] http://www.itc.edu.co

[11] C. Camargo. SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales. *XVII Workshop de Iberchip, Bogotá Colombia*, February 2011.
[12] D. Alimisis and C. Kynigos. *Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods*. School of Pedagogical and Technological Education (ASPETE), 2009.
[13] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *IEEE Colombian Workshop on Circuits and Systems*, 2007.
[14] 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.
[15] C. Camargo. First Colombian Linux SBC runs Debian. URL: http://www.linuxfordevices.com/c/a/News/First-Colombian-Linux-SBC-runs-Debian/, 2006.
[16] Abu Zaharin and Mohd Nasir. A Study On The DC Motor Speed Control By Using Back-EMFVoltage. *ASIAN CONFERENCE ON SENSORS*, July 2003.
[17] R. LeGrand. Closed-Loop Motion Control for Mobile Robotics. *Circuit Cellar Ink*, August 2004.
[18] C. Camargo. Linux como herramienta de Desarrollo de Sistemas Embebidos. *XII Workshop de Iberchip, San Jose*, 2006.
[19] Ricarose Vallarta Roque. OpenBlocks : an extendable framework for graphical block programming systems. Master's thesis, MIT, 2007.
[20] 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.