

ECBOT & ECB_AT91 Open Development Platforms for Robotics, Embedded Systems Design, and HW-SW co-design

Carlos Iván Camargo^{1*}

Departamento de Ingeniería Eléctrica y Electrónica, cicamargoba@unal.edu.co,
Bogotá, Colombia

Abstract. This paper presents two *open* Development platforms for Embedded Systems Design. ECBOT: A Development board for Mobile Robotics applications, using HW-SW co-design, and ECB_AT91: An Embedded System Development Platform. These platforms allow that them will be used as tool for teaching: Digital Systems Design methodologies, operating systems, and any course that requiere digital processing like control, instrumentation, communications, etc. Thanks to his architecture ECBOT allow create applications like: test bench for Multi-Robot System (MRS) control algorithms, test bench for imagen processing algorithms, etc. ECB_AT91 can be adapted to a lot of applications that require network conection, USB devices. All the source code, schematics, and PCB's fabrication files, are available for downloading at <http://www.emqbit.net/ECBOT>.

1 Introducción

Según Campusano [1], en 1996 el New York Times estimó que el Americano promedio estaba en contacto con cerca de 60 microprocesadores cada día, y año tras año esta cifra va en aumento; esto nos indica, que los sistemas digitales están invadiendo las actividades humanas y hoy en día es muy difícil encontrar una situación en la que no se utilice un dispositivo digital. Existe un gran mercado que mueve miles de millones de dólares al año, el tamaño de este mercado es tal, que el número de procesadores vendidos para este fin en el 2000, según Venture Development Corporation¹ fué de 6.4 Billones y se estima que en el 2010 llegarán a 16 Billones.

Por otro lado, la robótica móvil es un campo de investigación que crece rápidamente y en la actualidad existe un gran número de grupos de investigación que trabajan en el desarrollo de técnicas de auto-organización para sistemas Multi-Robot, entre los que se encuentran: El proyecto Swarm-bots [2], [3], Interaction Labs (USC) [4] [5],

* Founded by Facultad de Ingeniería UNAL

¹ <http://www.vdc-corp.com/>

Autonomous System Lab (EPFL), MIT Computer Science and Artificial Intelligence Laboratory [6]. Los robots utilizados por estos y otros grupos no son ajenos a la anteriormente mencionada invasión digital, aunque los primeros robots se construyeron con unidades de procesamiento basadas en microncontroladores, las nuevas plataformas disponibles presentan una gran variedad de ayudas al desarrollo de los algoritmos de control: poseen procesadores de 32 bits con una gran capacidad de computo, mecanismos “robustos” de comunicación inalámbricos, Sistemas Operativos con capacidades de Tiempo Real.

ECBOT fue desarrollado en el Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia, con el objetivo de proporcionar una plataforma Hardware y Software **abierto** de bajo costo para el desarrollo de Sistemas Embebidos utilizando co-diseño HW/SW, con aplicación en Sistemas Multi-Robot (MRS). Gracias a que ECBOT utiliza el sistema operativo Linux, existe una gran cantidad de aplicaciones y librerías que pueden ser utilizadas en el desarrollo de algoritmos de control. Se eligió el proyecto *Player/Stage* [7], como plataforma de desarrollo de dichos algoritmos, *player* suministra funciones de alto nivel que encapsulan el manejo de bajo nivel de los sensores y actuadores, haciendo posible que la programación del Robot se realice en un lenguaje de programación de alto nivel como Java, Lisp, Python, C/C++, etc.

Gracias a su carácter abierto, ECBOT y ECB_AT91 pueden ser construidas por cualquier persona, en cualquier lugar del mundo, los esquemáticos y los archivos de fabricación de la placa de circuito impreso se encuentran disponibles, así como el código fuente de las aplicaciones necesarias para obtener un dispositivo funcional. ECBOT permite la adopción de nuevos periféricos a través de 7 señales análogas disponibles, 10 pines de Entrada/Salida digitales y un Puerto I2C, controlado por el procesador central. Los drivers necesarios para controlar estos puertos análogos y digitales de expansión también se encuentran disponibles.

2 Arquitectura Global de ECBOT

ECBOT está compuesto por un procesador Central ARM920T de 32 bits, encargado de las comunicaciones (WiFi, serial), el procesamiento de la información, y la configuración/programación de la FPGA y de 5 microncontroladores AVR de 8 bits; los cuales se comunican con el procesador central a través de una interfaz I2C. Adicionalmente ECBOT posee un acelerómetro que también se comunica por el puerto I2C. Se escogió el puerto I2C ya que este permite la conexión de hasta 128 diferentes dispositivos y utiliza solo dos líneas para controlarlos, lo cual reduce la interconexión y la complejidad de la placa de circuito impreso.

2.1 Arquitectura Software

Control de Sensores y Actuadores Como se mencionó anteriormente, ECBOT posee 6 microcontroladores de 8 bits (AVR de Atmel) que realizan el control de los sensores y actuadores (Motores DC, Sensores Infra-rojos y LEDs), Cada uno de estos microcontroladores implementa el protocolo I2C esclavo y obedece a una única dirección (entre 0 y 127), adicionalmente implementa el control completo de su respectivo sensor o actuador.

Linux Embebido Uno de los requerimientos de ECBOT fué la capacidad de ejecución en forma nativa del sistema operativo Linux. Esto debido a que Linux ya posee una gran trayectoria y miles de programadores han creado aplicaciones en una gran variedad de áreas del conocimiento, es un sistema operativo abierto que permite la creación de drivers para manejar dispositivos Hardware y es de libre distribución. Por esta razón, el primer paso en el desarrollo de ECBOT fue el estudio de plataformas que soportaran Linux, como resultado de este estudio se construyó la plataforma abierta: ECB_AT91 [8], la cual fué la base del desarrollo que dio como resultado ECBOT.

Durante el desarrollo de ECB_AT91 y de ECBOT se realizó el *port* de Linux a esta familia de plataformas, y este esfuerzo está incluido en la distribución “oficial” de los procesadores AT91 de Atmel². Así mismo se realizaron variaciones sobre el loader de Linux u-boot³, y una serie de aplicaciones que se utilizan para programar y configurar los microcontroladores y la FPGA.

Se adaptaron 3 nuevos paquetes a las distribuciones Buildroot y OpenEmbedded: El servidor Player, el programador de microcontroladores para AVR: *uisp*⁴ y el programador para FPGAs: *xc3sprog*⁵. Esto con el fin de hacer que ECBOT no requiera ser conectado a un computador personal para realizar cambios de programación en alguno de sus componentes, los microcontroladores de 8 bits y la FPGA, son programados por el procesador central a través de pines de Entrada/Salida de propósito general, un driver emula el comportamiento de un puerto paralelo, los archivos de programación/configuración (transferidos de forma inalámbrica) son almacenados en la plataforma y pueden ser modificados en cualquier momento; esto hace que ECBOT sea una plataforma independiente y que pueda ser programada “en caliente”.

² El patch oficial para la Arquitectura AT91 de Atmel puede ser descargada de:
http://maxim.org.za/at91_26.html

³ <http://sourceforge.net/projects/u-boot>

⁴ <http://www.nongnu.org/uisp/>

⁵ <http://www.rogerstech.co.uk/xc3sprog/>

2.2 Arquitectura Hardware

Uno de los criterios de diseño de ECBOT fué la capacidad de expansión; su arquitectura debe permitir la adición de nuevos sensores y actuadores de forma fácil; por este motivo, se escogió el puerto I2C como protocolo de comunicación entre la Unidad de procesamiento Central y los sensores y actuadores. En la actualidad, la mayoría de los microcontroladores disponibles en el mercado implementan este protocolo, lo cual permite adaptar cualquier sensor a ECBOT. Las características de ECBOT se listan a continuación:

- Procesador: ARM 920 AT91RM9200 @ 200MHz
- Movimiento: 2 Servo motores DC
- Sensores, Entrada/Salida:
 - 8 sensores Infrarojos de proximidad y luz de ambiente
 - 10 I/O Digitales, 8 Analógicas
 - 8 Leds Programables
 - Sensor de Imágen de hasta 640x480 pixels.
- Comunicación:
 - Puerto serie Standard
 - Comunicación USB
 - Ethernet Inalámbrico
- Simuladores: Player/Stage
- Herramientas de Desarrollo: GNU TOOLS
- Costo: 400 USD

Como puede verse en La Figura 1, seis microcontroladores de 8 bits (AVR de Atmel) permiten el manejo de los sensores y actuadores a través del bus común I2C. Cuatro microcontroladores de 8 bits, manejan cada uno de ellos: dos sensores infrarojos de proximidad y luz de ambiente, utilizados en tareas de evasión de obstáculos y 6 LEDs (2-rojos, 2-verdes, 2-azules) que representan el estado interno del Robot; el quinto microcontrolador de 8 bits se encarga de manejar la velocidad y posición de 2 motores DC, 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 [9], [10]; el sexto procesador se encuentra disponible para que el usuario utilice sus entradas analógicas.

Adicionalmente ECBOT cuenta con una FPGA que comparte el bus de Datos, dirección y control con el procesador central, lo que permite, la creación de periféricos dedicados y de aplicaciones HW/SW. En la placa se encuentran dos conectores que permiten que el estudiante tenga acceso a 26 pines (divididos en 16 y 10 señales) de Entrada/Salida de propósito General (GPIOs). En la plataforma robótica, se utilizan 16 GPIOs para manejar un sensor de imagen (KAC9628), y los 10 restantes para controlar 10 servo-motores.

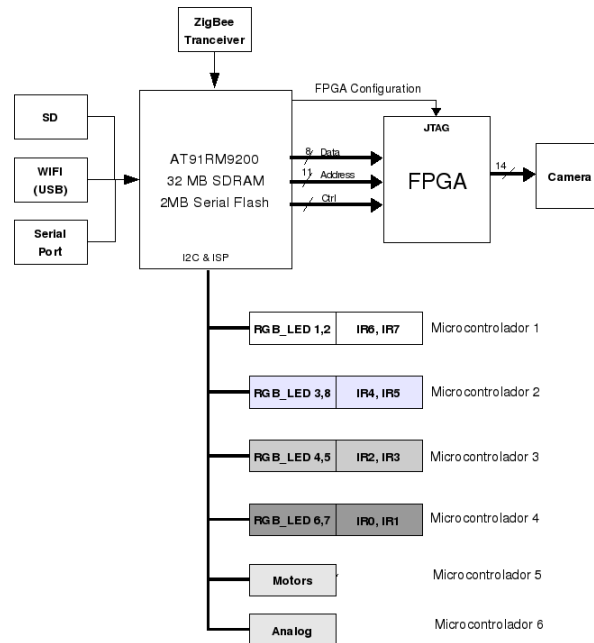


Fig. 1. Diagrama de Bloques del componente Hardware de ECBOT.

Comunicaciones ECBOT proporciona 2 formas de comunicación:

- Módulo WiFi: ECBOT permite la conexión de un adaptador USB 802.11, en la actualidad solo los adaptadores basados en el chip ZD1211 están soportados.
- Puerto Serie: En las primeras etapas del desarrollo es necesario utilizar el puerto serie para cargar las imágenes del loader, bootloader y kernel.

Memorias y Dispositivos de Almacenamiento 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). ECBOT permite la utilización de una memoria SDRAM de hasta 64 MBytes, la cual es utilizada como memoria de propósito general para las aplicaciones que corren bajo Linux. Adicionalmente se proporcionan dos conectores para memoria SD, uno para una SD standard y el otro para una tarjeta micro-SD.

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. Esto facilita la creación del soporte necesario para la board; la información necesaria sobre el *port* de Linux a la plataforma se puede encontrar en [11].

Fotografías de la tarjeta principal de ECBOT En las Figuras 2 y 3 se muestra el lado de componentes y el lado de soldadura de la tarjeta principal de ECBOT; en ellas podemos observar la localización de los componentes, la cual está fuertemente influenciada por el robot *e-puck* [12] del EPFL.

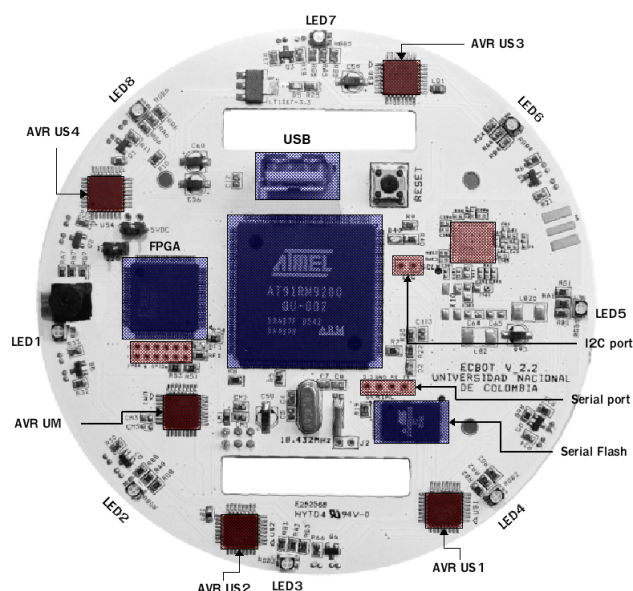


Fig. 2. Lado de Componentes de la tarjeta principal de ECBOT.

3 Aplicaciones de la plataforma ECBOT

3.1 Robótica

Resumiendo lo visto hasta el momento: Contamos con una plataforma HW que posee una serie de recursos para el desarrollo de aplicaciones en robótica móvil, tenemos una serie de sensores y actuadores que se comunican por el puerto I2C con el procesador

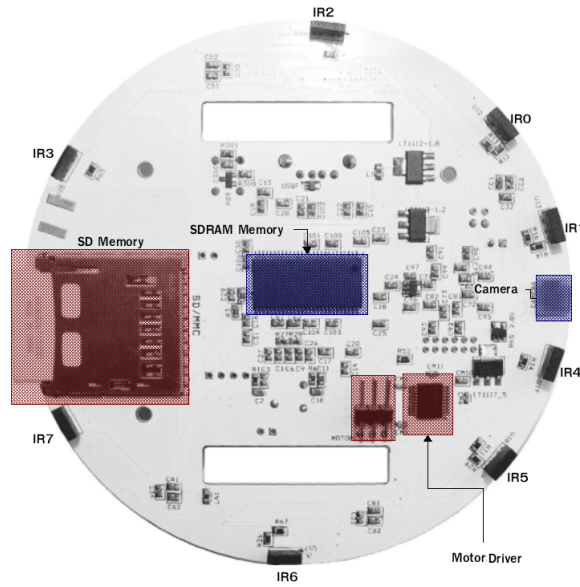


Fig. 3. Lado de soldadura de la tarjeta principal de ECBOT.

central de dicha plataforma. Solo falta crear aplicaciones que utilicen la información proveniente de los sensores para generar acciones en los actuadores. Esto se puede realizar de dos formas: Utilizar los mecanismos de Entrada/Salida de los drivers de Linux, lo cual, aunque no es difícil es un poco engorroso y provoca que el programador se pierda entre los diferentes llamados a funciones, no recuerde con qué formato debe comunicarse con cada dispositivo, o qué información suministran, etc. La otra alternativa es utilizar un API que se encargue de este manejo tedioso y suministre al programador funciones de fácil utilización. Aquí es donde entra en acción el servidor Player:

Servidor Player El proyecto Player/Stage, es el resultado de un estudio del Grupo de Investigación en Robótica de la *University of Southern California* [7], y esta dividido en tres partes:

1. **Player** Es un servidor que proporciona una interfaz de red flexible a una gran variedad de sensores y actuadores. Como puede verse en la Figura 4 utiliza un modelo cliente/servidor basado en sockets TCP, su principal característica es el manejo de los sensores y actuadores a través de una interfaz de programación de alto nivel, lo cual permite que los programas de control del robot sean escritos en cualquier lenguaje y puedan ser ejecutados en cualquier plataforma⁶ (cliente)

⁶ También es posible que el cliente y el servidor se ejecuten en el mismo robot.

que posea una conexión de red con el robot (servidor). Además, Player soporta múltiples conexiones concurrentes de clientes, lo que es muy útil en estudios de control descentralizado.

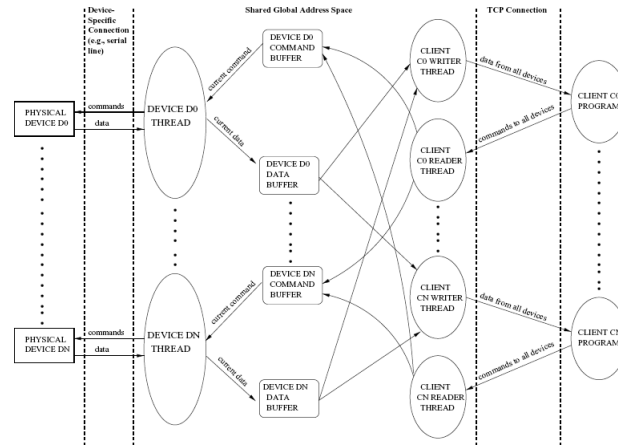


Fig. 4. Arquitectura de Player. [13]

2. **Stage** Es un simulador multi-robot; que simula robots que se mueven y realizan operaciones de sensado en un entorno de dos dimensiones, estos robots son controlados por Player. Stage proporciona robots virtuales los cuales interactúan con dispositivos simulados. En la Figura 5 se puede observar una simulación multi-robot utilizando las librerías de Stage; una de las características más atractivas del proyecto Player-Stage es la creación de sensores y actuadores que simulan el comportamiento de los dispositivos reales.
3. **Gazebo** Es un simulador multi-robot 3D. A diferencia de Stage que fué diseñado para simular grupos numerosos de robots, gazebo simula el comportamiento de poblaciones pequeñas de robots (menos de 10).

Adicionalmente, una de las características más atractivas del proyecto Player/Stage es su carácter libre y que su código fuente está disponible, gracias a esto, fué posible escribir un driver que soportara a ECBOT. Este driver permite controlar:

1. El dispositivo de seguimiento de color implementado en la FPGA.
2. La velocidad y posición de 2 motores.
3. El color de los LEDs.
4. La activación y la lectura de los sensores InfraRojos de proximidad.

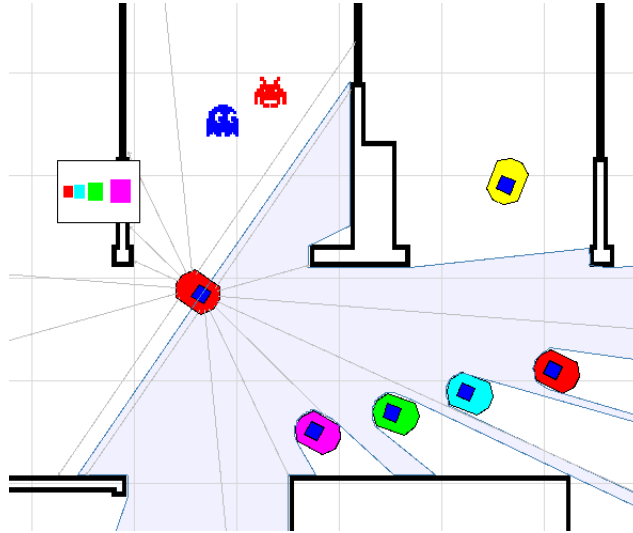


Fig. 5. Captura del simulador Stage.

La figura 6 muestra gráficamente la arquitectura del servidor Player de ECBOT, como puede verse, el servidor está encargado de comunicarse con los sensores (cámara y sensores InfraRojos), procesar esta información y enviarla al cliente, donde el algoritmo de control por intermedio de un generador de comandos del servidor controla los actuadores (LEDs y motores).

4 Arquitectura General de ECB_AT91

4.1 Componente Hardware

Como se mencionó anteriormente ECB_AT91⁷ fué la plataforma que se diseñó para dar inicio al estudio de Sistemas Embebidos con Linux como sistema operativo, su importancia radica en ser el primer SBC fabricado en Colombia. Y gracias a que los archivos de fabricación y el código fuente de las aplicaciones requeridas para su funcionamiento se encuentran disponibles; ha sido utilizada en varias universidades en diferentes países de mundo; se tienen reportes de la utilización de ECB_AT91 en: Perú, Venezuela, Portugal, USA, Bélgica, España, Italia, Alemania y Brasil. Nuestra empresa ⁸ recibe a diario muchos reportes y preguntas relacionadas con el diseño y funcionamiento de esta placa.

⁷ <http://wiki.emqbit.com/free-ecb-at91>

⁸ <http://www.emqbit.com>

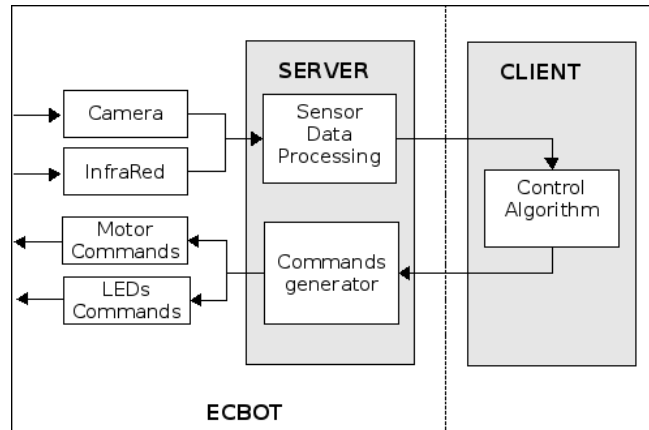


Fig. 6. Servidor Player de ECBOT.

Las Características de esta placa son (favor ver Figuras 7 y 8:

- procesador ARM9 de 180 MHz (Atmel AT91RM9200).
- 2 MBytes de memoria flash serial.
- Hasta 64 MBytes de memoria SDRAM (Soporta 8M/16M/32M/64M).
- 1 slot SD/MMC.
- 1 puerto host USB 2.0.
- 1 puerto I2C.
- 1 Interfaz Ethernet 10/100.
- 4 Interfaces SPI.
- 2 Interfaces seriales (RS232).
- Soporte JTAG.
- Bus de Datos (16 bits), Dirección (7 bits) y Control Disponibles.
- 8 Pines de Entrada/Salida de Propósito General Disponibles.

4.2 Aplicaciones Realizadas con ECB_AT91

ECB_AT91 ha sido utilizada como base para el desarrollo de diversas aplicaciones, dentro de las que se destacan:

1. Sistema de Recolección, Procesamiento y Transferencia de Información para molinos de Campo Eléctrico.
2. Sistema de Automatización para Tornos Industriales.
3. Sistema de seguimiento Vehicular Utilizando la Red Celular.
4. Sistema de Monitoreo de Temperatura.
5. Sistema de Medición de Signos Vitales.
6. Gateway Para Red de Sensores Inalámbricos.

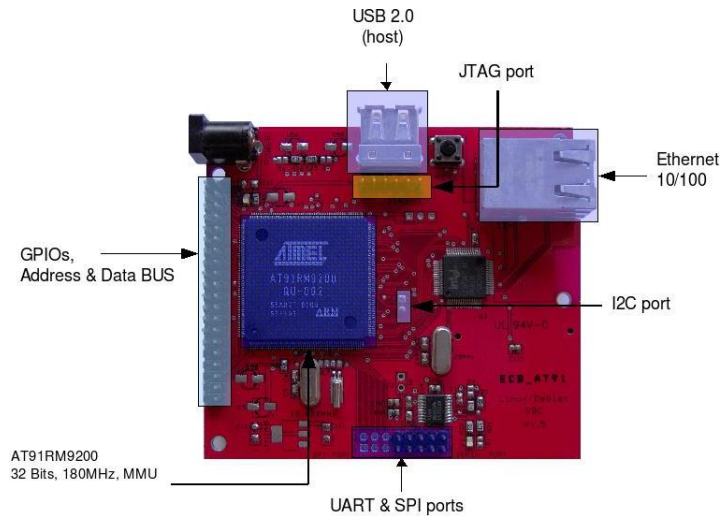


Fig. 7. Lado de Componentes de la placa ECB_AT91.

5 Conclusiones y Trabajo Futuro

Se presentaron dos plataformas abiertas HW/SW para el estudio de sistemas embebidos y sistemas multi-robot, los archivos necesarios para la fabricación de las placas de circuito impreso, esquemáticos, el firmware de los microcontroladores y su respectivo código fuente, cambios requeridos al kernel de Linux se encuentran disponibles.

El estudio del diseño de Sistemas Embebidos es muy importante en el mundo de hoy, ya que existe una gran demanda en constante aumento por parte de la sociedad.

Linux ha demostrado ser una plataforma ideal para el desarrollo de este tipo de aplicaciones ya que gracias a la gran cantidad de aplicaciones y librerías disponibles reduce el tiempo requerido para la implementación de las aplicaciones.

ECBOT y ECB_AT91 son dos herramientas poderosas en la enseñanza de sistemas digitales, ya que pueden ser utilizadas en diferentes niveles y en diferentes contextos (tecnología, ingeniería).

Referencias

1. R. Camposano and W. Wolf. *Design Automation for Embedded Systems*. Springer, January 1996.
2. 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

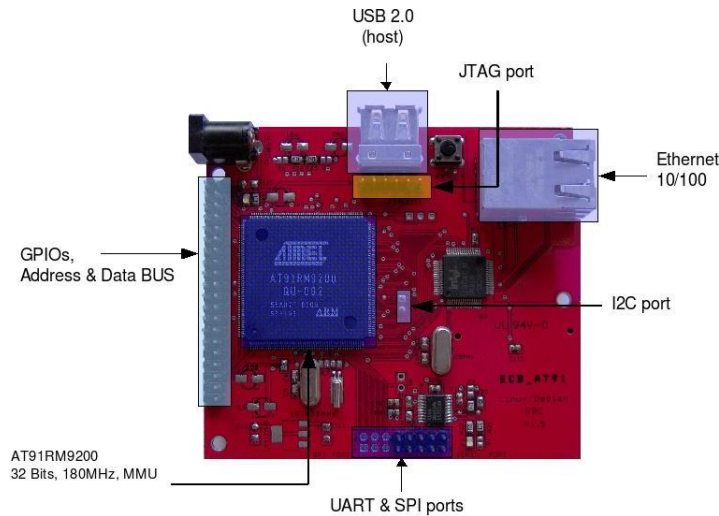


Fig. 8. Lado de Soldadura de la placa ECB.AT91.

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.

3. 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.
4. 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.
5. 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.
6. J. McLurkin. Speaking Swarmish. *AAAI Spring Symposium*, 2006.
7. B.P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for Multi-robot and Distributed Sensor Systems,. *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 317–323, 2003.
8. C. Camargo. First Colombian Linux SBC runs Debian. <http://www.linuxdevices.com/news/NS6698241512.html>, 24 April 2006.
9. Abu Zaharin and Mohd Nasir. A Study On The DC Motor Speed Control By Using Back-EMF Voltage. *ASIAN CONFERENCE ON SENSORS*, July 2003.
10. R. LeGrand. Closed-Loop Motion Control for Mobile Robotics. *Circuit Cellar Ink*, August 2004.
11. C. Camargo, N. Castillo, and A. Calderón. Free ECB.AT91. <http://wiki.emqbit.com/wiki>.
12. EPFL. e-puck EPFL Education robot. <http://www.e-puck.org/>.
13. B. Gerkey, K. Stoy, and R. T. Vaughan. Player Robot Server. Technical report, USC Robotics Labs, 22 November 2000.