Sobre el sistema de control de un robot cartesiano

Propuesta de actualización del hardware con una beagleboard y un microprocesador

Andrés Mauro
Laboratorio de Robótica, Depto. de Electrónica
Facultad de Ingeniería de la Universidad de Buenos Aires
Buenos Aires, Argentina
amauro@fi.uba.ar

Mauricio Anigstein
Laboratorio de Robótica, Depto. de Ing. Mecánica
Facultad de Ingeniería de la Universidad de Buenos Aires
Buenos Aires, Argentina
manigst@fi.uba.ar

Resumen— **El Laboratorio de Robótica de la FIUBA cuenta con un robot industrial de 6 ejes de la firma ABB y un robot didáctico para tareas planas construido en el Laboratorio de Robótica por alumnos y docentes investigadores de la Cátedra: el Gantry UBA de 3 ejes + 1 eje de dos posiciones. El software de control del Gantry está distribuido en varios procesos organizados jerárquicamente que realizan sus tareas de manera cooperativa y concurrente utilizando un único procesador y bajo la plataforma del sistema operativo de tiempo real QNX. En este artículo se propone un nuevo diseño del hardware del sistema de control del Gantry UBA con el objetivo de mejorarlo y actualizarlo, reemplazando los dispositivos y sistemas electrónicos que han quedado obsoletos debido a los avances tecnológicos de los últimos años, especialmente en el área de la electrónica y los sistemas embebidos.**

Palabras Calve—robótica; controlador; tiempo real; sistemas embebidos.

I. INTRODUCCIÓN

El robot Gantry UBA fue construido en el Laboratorio de Robótica por alumnos y docentes investigadores de la cátedra [1], [2], [3], [4] y es el resultado de sucesivas tesinas de graduación relacionadas. Es un robot tipo pórtico (Gantry), con 3 ejes servocontrolados que permiten posicionar y orientar la herramienta en forma arbitraria en un plano. Un cuarto eje de dos posiciones permite el desplazamiento vertical de la herramienta, que consiste en una pinza accionada por un solenoide que puede tomar pequeñas piezas.

En el desarrollo de un manipulador robótico confluyen temas provenientes de distintas áreas del conocimiento científico-tecnológico: la mecánica, la teoría de los sistemas de control, los dispositivos electromecánicos y electrónicos, y los sistemas informáticos.

El software del controlador del robot Gantry UBA se desarrolló en C++ y lo ejecuta una PC utilizando como software de base el sistema operativo de tiempo real QNX [5]. QNX permitió desarrollar un controlador en tiempo real distribuido en varios procesos organizados jerárquicamente que realizan sus tareas de manera cooperativa y concurrente utilizando un único procesador.

Actualmente el hardware de control está compuesto por dos placas encargadas de adquirir los datos provenientes del robot. Una placa interior a la PC y conectada a un BUS ISA de 8 bits y una placa exterior. Estas tarjetas constituyen la interfase entre la PC y el robot Gantry [3].

Debido a que su controlador es abierto, el robot Gantry UBA es una máquina sobre la cual se pueden generar actividades de investigación y desarrollo sobre cada uno de los subsistemas que lo componen para mejorar su funcionamiento y/o actualizar alguna de sus partes.

En el presente trabajo se propone el diseño de un nuevo hardware del sistema de control, que reemplace a la PC y a las placas de adquisición de datos por un sistema embebido. El objetivo es mejorar y actualizar el hardware del controlador, sin hacer grandes modificaciones sobre el código del software. En la siguiente sección se hace una descripción mecánica del robot. Luego se detalla la arquitectura y el software de control. Seguidamente se describe el hardware actual del controlador, y finalmente se expondrán los motivos por los cuales conviene actualizar el hardware, cuáles son las modificaciones propuestas, cómo sería aproximadamente la nueva arquitectura y qué ventajas tendría.

II. DESCRIPCIÓN GENERAL DEL ROBOT

Se trata de un robot cartesiano tipo pórtico (Gantry), Fig. 1, con 3 ejes servocontrolados que permiten posicionar y orientar la herramienta en forma arbitraria en un plano, dentro de un área de trabajo de 450x400 mm2. Un cuarto eje prismático permite el desplazamiento vertical de la herramienta, para acercarla y alejarla del plano de trabajo. Los actuadores de los 3 primeros ejes son motores de corriente continua de imán permanente con codificadores ópticos incrementales montados sobre sus ejes. El primer motor tiene además un tacogenerador.

Los dos primeros ejes son prismáticos, y permiten el desplazamiento del cabezal en un plano horizontal, mediante transmisiones motor-tornillo sinfín. El tercer eje es vertical y de rotación y controla la orientación de la herramienta mediante un motor con un reductor planetario (1:200). La base o mesa es una estructura prismática hueca de aluminio. La herramienta es una pinza cuya apertura es accionada con un solenoide y su cierre se logra con un resorte.

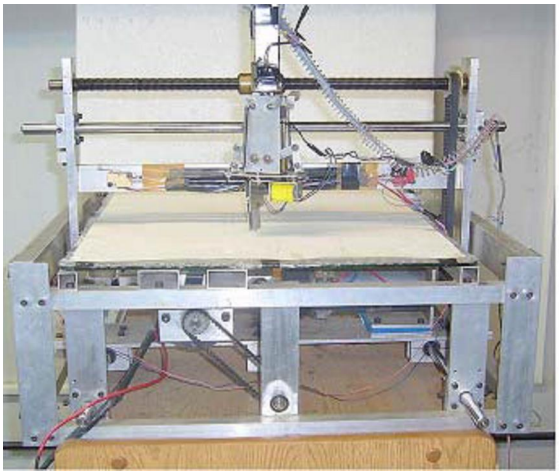

Fig. 1. Fotografía del Robot Gantry UBA

La Fig. 2 corresponde a los últimos dos eslabones del robot, en ella puede observarse el diseño de la pinza, el montaje del solenoide para el desplazamiento vertical de la herramienta, el mecanismo de palancas para multiplicar la carrera de descenso y los resortes utilizados para el ascenso.

El sistema cuenta, además, con:

- Entradas digitales: 12 x 8 bits
- Salidas digitales: 12 x 8bits
- Entradas analógicas: 6
- Salidas analógicas: 4

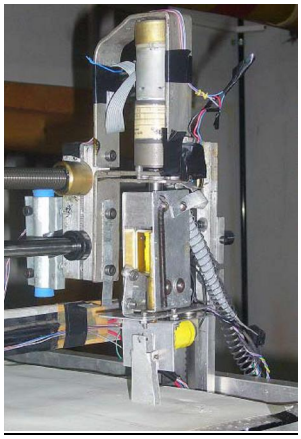

Fig. 2. Los dos últimos eslabones del robot Gantry

III. FUNCIONAMIENTO

Un robot industrial es un manipulador mecánico controlado automáticamente, flexible y de propósito general, cuyos movimientos pueden ser programados para realizar una amplia variedad de tareas. La ejecución de una tarea involucra el seguimiento de una trayectoria cuya generación usualmente está determinada por eventos externos que pueden provenir de sensores, dispositivos u otras máquinas que formen parte del ambiente de trabajo y con los cuales el robot debe interactuar. La ocurrencia de estos eventos no se puede predecir y por lo tanto las trayectorias no pueden ser planificadas sino que deben ser generadas en tiempo real a medida que se ejecuta el movimiento. Por otro lado, el controlador debe asegurar los tiempos de ejecución de las distintas tareas para poder realizar los movimientos recorriendo las trayectorias en forma y tiempo, cumpliendo con los requerimientos tanto de posición como de velocidad.

La generación automática de la trayectoria hace la descripción del movimiento fácil para el usuario, dejando que el sistema realice los cálculos necesarios [6]. Para programar una tarea el operador especifica una secuencia de objetivos (posición y orientación) por los cuales la herramienta del robot deberá acercarse. Para pasar exactamente por un objetivo deberá detenerse en el mismo [7]. El algoritmo generador de trayectoria calcula los puntos intermedios a seguir entre los objetivos ingresados por el operador. Estos puntos intermedios son las referencias para el algoritmo que realiza el proceso de control de los actuadores de los ejes. La Fig. 3 muestra un conjunto de posiciones en el espacio de coordenadas para una variable genérica θ en función del tiempo, unidas por una función lineal por tramos. Por la estructura cinemática del Gantry, linealidad en el espacio de los ejes significa linealidad en el espacio cartesiano y no existen singularidades. En cada uno de los segmentos la velocidad es constante, sin embargo los cambios bruscos de velocidad que se muestran no son admisibles, requerirían aceleraciones (torques en los actuadores) infinitas.

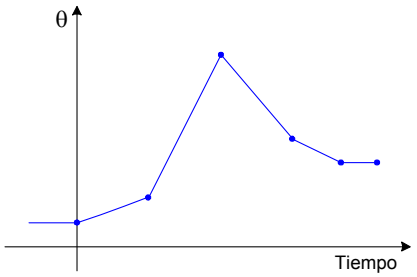

Fig. 3. Función lineal por tramos

Con el fin de garantizar la continuidad en la posición y la velocidad, los sucesivos segmentos lineales de la trayectoria se unen definiendo zonas alrededor de cada uno de los puntos de transición, en las cuales la posición será una función polinómica del tiempo, ya que dentro de estas zonas la aceleración no será nula. Hay que tener en cuenta el tiempo necesario para acelerar al manipulador desde el reposo hasta alcanzar su máxima velocidad, tacc. Las zonas de aceleración tendrán una duración de 2tacc. En un instante de tiempo tacc anterior al tiempo necesario para alcanzar el punto B, el manipulador estará moviéndose a velocidad constante y pasando por el punto definido como A, como se muestra en la Fig. 4.

Recién en ese instante es necesario conocer las coordenadas del punto C para empezar a evaluar el movimiento desde el punto B hacia el C.

Es también necesario que el robot mueva todos sus ejes al mismo tiempo y que todos ellos lleguen simultáneamente a la posición de destino, es decir, que el movimiento sea coordinado. Por lo tanto, conociendo el valor de las coordenadas de cada eje (i) correspondientes a los puntos B y C, y la máxima velocidad permitida de cada eje, se calcula T1 = tiempo total para recorrer el segmento B – C como el máximo valor entre: los tiempos que tarda cada uno de los ejes en recorrer el segmento, dos veces el tiempo de aceleración y eventualmente un tiempo proporcionado por el usuario (que requiera el proceso) (1):

$$T_1 = \max\left\{ \frac{|\theta_{Ci} - \theta_{Bi}|}{v_{i\,max}} \; ; \; 2\, t_{acc} \; ; \; T_{usuario} \right\} \qquad (1)$$

Obsérvese que no es necesario conocer las coordenadas del punto objetivo siguiente D, hasta tanto el tiempo transcurrido t sea igual a T1-tacc. Cuando esto ocurre, se inicia la transición al siguiente segmento calculando T2 con la Ecuación (1) (tiempo total para recorrer el segmento C – D), y haciendo las siguientes asignaciones:

T1 = T2

A = X Posición actual.

B = C

t = -tacc Resetear el tiempo.

IV. Arquitectura Del Software De Control

El controlador del robot Gantry UBA se desarrolló en C++ y se ejecuta en una PC utilizando como software de base el sistema operativo de tiempo real QNX [5]. Además cuenta con una interfase gráfica de usuario diseñada y programada utilizando como herramienta el Photon Application Builder (PhAB) [8].

Está compuesto por cinco procesos organizados jerárquicamente que realizan sus tareas de manera cooperativa y concurrente utilizando un único procesador. Dos de ellos son esenciales para el funcionamiento del robot: el proceso que genera en tiempo real la trayectoria que debe realizar la herramienta, y el proceso encargado de realizar el control de los ejes interactuando con el hardware. El resto realiza tareas auxiliares de configuración, monitoreo y supervisión.

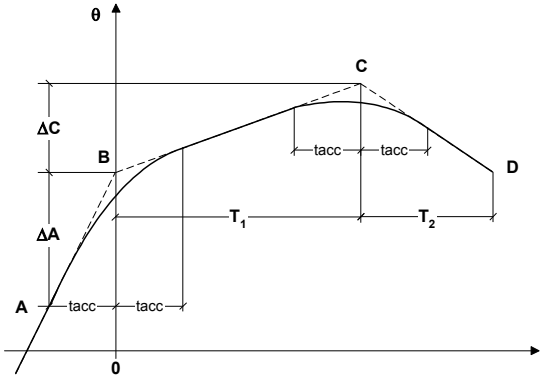

Fig. 4. Transición al siguiente segmento

A. *El proceso que genera en tiempo real la trayectoria que debe realizar la herramienta*

El proceso generador de trayectoria lee e interpreta una instrucción, resuelve el problema inverso, calcula el tiempo T1 necesario para alcanzar la siguiente posición y genera las sucesivas referencias que serán enviadas al proceso que realiza el control. Para que el sistema funcione en tiempo real, se debe asegurar el cumplimento de las metas de tiempo para las diferentes tareas. En particular, el generador de trayectoria no sólo debe generar una nueva referencia a intervalos regulares de tiempo, sino que debe garantizar que esas referencias se correspondan con el tiempo realmente transcurrido. Finalmente, al cabo de un tiempo t igual a T1– tacc, debe ir en busca de la siguiente instrucción y así repetir el ciclo hasta la última instrucción.

B. *El proceso encargado de realizar el control de los ejes*

El proceso de control, recibe las referencias del generador de trayectoria, las compara con las posiciones reales de los ejes leyendo los sensores de posición y calcula y envía la tensión de control para los actuadores. Este proceso debe funcionar en todo momento, aun cuando el robot no se encuentre en movimiento (no reciba nuevas referencias del generador de trayectoria). Debe rechazar ruidos, perturbaciones externas sobre el manipulador y eventualmente esfuerzos provocados por el peso propio. Su objetivo no es cumplir tiempos de movimiento sino seguir las referencias del generador estableciendo un período de muestreo que asegure la estabilidad del sistema.

- Si hay nuevas referencias, las actualiza. Si no, realiza el control con las últimas referencias recibidas
- Lee los contadores.
- Calcula las posiciones.
- Calcula los errores.
- Aplica las actuaciones.

Debido a la configuración cinemática cartesiana del robot, es suficiente para lograr estabilidad y una respuesta aceptable, utilizar una estrategia de control lineal y de ejes independientes. En la Fig. 5 se muestra el diagrama en bloques del sistema de servo control digital implementado actualmente para el control de los motores del robot.

Los procesos adicionales del software del controlador, realizan tareas auxiliares de configuración, monitoreo y supervisión.

El uso de un sistema operativo de tiempo real como QNX, la cooperación entre procesos y la organización del software permitió desarrollar un controlador para que el robot Gantry UBA pueda ejecutar trayectorias no sólo con la forma deseada sino también en el tiempo deseado [9].

V. Descripción Del Hardware Actual

El hardware que actualmente está en funcionamiento, se compone de dos placas que constituyen la interfase entre el software del controlador que se ejecuta en la PC y el robot

Gantry. En la Fig. 6 se puede ver un diagrama en bloques general del hardware actualmente implementado.

La PC es una pentium 1 con 16 Mbytes de memoria total, 133Mhz de velocidad del núcleo y 66Mhz de velocidad en el bus. No dispone de puertos USB ni conexión a red.

A. Placa Interior

Esta placa está conectada a un BUS ISA de 8 bits y contiene los bloques que se muestran en la Fig. 7.

Estos bloques interactúan entre sí y con el medio externo. La lógica de entrada intercambia datos entre el BUS ISA y la unidad central, memoriza la posición de memoria base de la placa para que la PC pueda trabajar con ella y genera la señal que bloquea a la PC hasta que su pedido sea procesado.

La unidad central genera todas las señales de control para las siguientes etapas, separa los buses de datos y direcciones y almacena toda la información de entrada proveniente de la placa exterior.

Las restantes etapas actúan independientes entre sí, sólo son controladas por la unidad central. Las dimensiones y el consumo de esta placa se resumen en la tabla 1.

TABLA 1. CONSUMO Y DIMENSIONES DE LA PLACA INTERIOR

Placa Interior				
Dimensiones				***Consumo típico***
Largo	***Ancho***	***Alto***	***Peso***	
226 mm	117 mm	18 mm	180 gr	350 mA (±5V) - 100 mA (±12V)

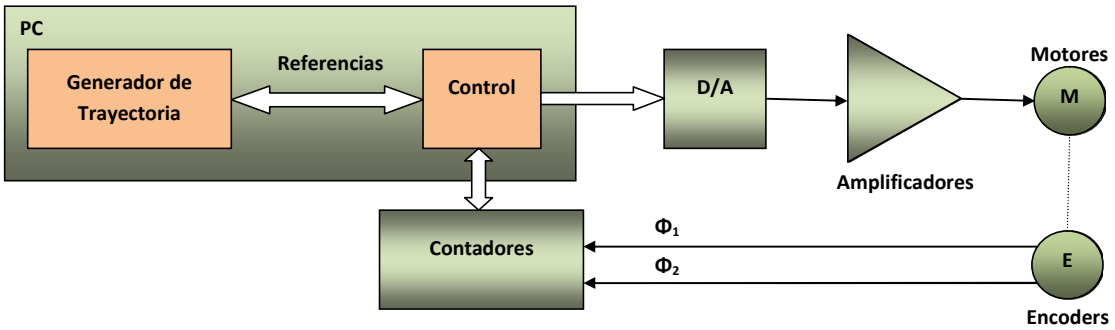

Fig. 5. Diagrama en bloques del sistema de servo control actual

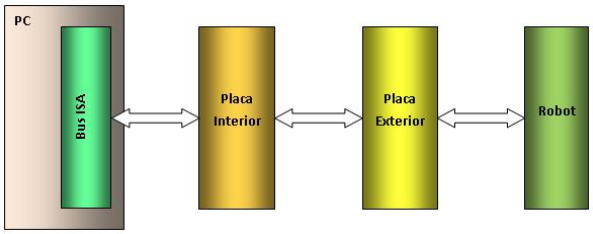

Fig. 6. Diagrama en bloques general del hardware actual

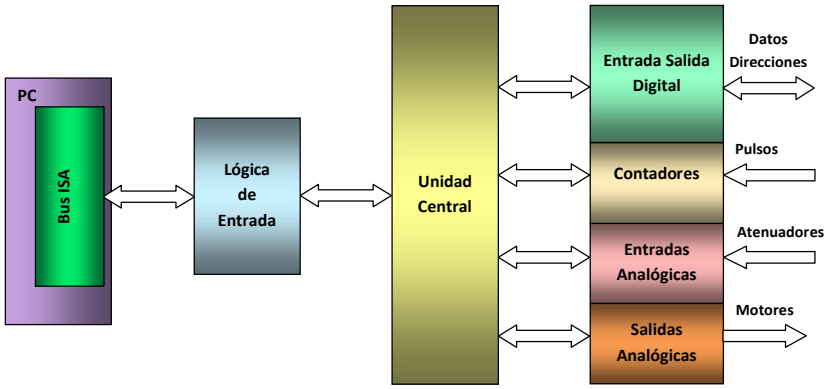

Fig. 7. Diagrama en bloques de la placa interior

B. *Placa Exterior*

En esta placa se encuentra el bloque que demultiplexa las salidas y multiplexa las entradas digitales, una etapa de atenuadores, una de potencia para el motor 3 y un bloque para el control de la mano que maneja los relés de apertura de la pinza y descenso de la herramienta, como muestra la Fig. 8.

La etapa de rectificación y regulación genera las tensiones de alimentación para el control de la mano, la etapa de potencia del motor 3 y los atenuadores de las entradas analógicas.

Ambas placas se conectan entre sí con un cable plano de 37 vías y 2 metros de longitud.

La tabla 2. resume las dimensiones y el consumo de la placa exterior.

TABLA 2. CONSUMO Y DIMENSIONES DE LA PLACA EXTERIOR

Placa Exterior				
Dimensiones				***Consumo máximo***
Largo	***Ancho***	***Alto***	***Peso***	
207 mm	102 mm	33 mm	170 gr	1200 mA

C. *Tecnología Del Hardware Actual*

La electrónica de los diferentes bloques está implementada utilizando componentes COTS (commercial off-the-shelf), circuitos integrados MSI de tecnología CMOS 74HCxx. Memorias PAL, buffers, transceivers, latches, comparadores, contadores, un ADC de 12 bits, un DAC de 8 bits y otro de 13, amplificadores operacionales, reguladores de voltaje, optoacopladores, transistores y diodos. Un microcontrolador AT89C51de 8 bits, CMOS, con 4 kbytes de memoria flash y 20 Mhz de frecuencia de reloj.

VI. PROPUESTA PARA UN NUEVO HARDWARE

El hardware que se presenta en esta sección, todavía no está en funcionamiento, se trata de una propuesta de actualización sobre la cual se está trabajando. La modificación propone el reemplazo de la PC por una Single Board Computer (SBC) que ejecutará los procesos de control, generación de trayectoria, supervisión y monitoreo; con una interfaz gráfica de usuario y un módulo de enseñanza/aprendizaje a través de una pantalla táctil. También se propone la sustitución de los diferentes bloques del hardware que actualmente están distribuidos entre las dos placas de adquisición, por un microcontrolador apropiado para aplicaciones embebidas que requieran un alto nivel de integración.

Para la selección de la SBC del hardware se establecieron dos requisitos fundamentalmente: que disponga de un BSP (Board Support Packages) para embeber el sistema operativo QNX Neutrino RTS, y que sea una plataforma de hardware abierta con foro actualizado de soporte y consulta. Además, se tienen en cuenta diversos factores como: disponibilidad en el mercado, costo, puertos de I/O, memoria RAM, memoria Flash, puertos USB, puertos Ethernet, puerto JTAG, entre otros.

Considerando estos requerimientos, la SBC que se propone en este trabajo es la BeagleBoard-xM [10]. Ésta es una placa de bajo costo y bajo consumo. Toda la información de diseño del hardware se encuentra disponible. Utiliza un procesador DM3730 de Texas Instruments con un núcleo ARM Coretx-A8 1 GHz como procesador de propósito general (GPP), un procesador DSP TMS320C64x+ 800 MHz como procesador de propósito específico, 512 MB de RAM con soporte para el almacenamiento por bloques en memoria SD. Además es posible incorporar pequeños módulos para conectar cámaras para la captura y el procesamiento de imágenes. Además, la BeagleBoard permite embeber el sistema operativo QNX Neutrino RTOS [11].

El microcontrolador elegido para implementar los bloques del hardware es el LPC1769 de la firma NXP [12]. Esta línea de microcontroladores tiene un núcleo ARM Coretx-M3 de 32 bits [13], opera a una frecuencia de CPU de 120 MHz, 512 KB de memoria Flash, 64 KB de memoria RAM, conversor DAC de 10 bits, un conversor ADC de 12 bits con entrada multiplexada en 8 canales, un PWM para el control de motores, cuatro timers de propósito general, seis salidas PWM de propósito general, interfaces Ethernet y USB, cuatro UARTS, CAN, I2C y SPI, entre otros. Es decir, cuenta con la cantidad de dispositivos periféricos que permitirán integrar los bloques necesarios para el control de los motores, la lectura de los encoders, las entradas y salidas digitales y analógicas necesarias. Además, las distintas aplicaciones se pueden programar utilizando un sistema operativo de tiempo real como FreeRTOS [14].

En la Fig. 9 se muestra el diagrama en bloques del sistema de servo control digital propuesto como alternativo al diseño actual. La conversión digital-analógica se efectuará utilizando modulación por ancho de pulso. El filtrado y la conversión de las señales de los codificadores incrementales se realizará utilizando un dispositivo PLD externo cuyo circuito ya fue diseñado utilizando el lenguaje VHDL. Las salidas serán etapas de potencia conmutada para incrementar el nivel de las señales PWM.

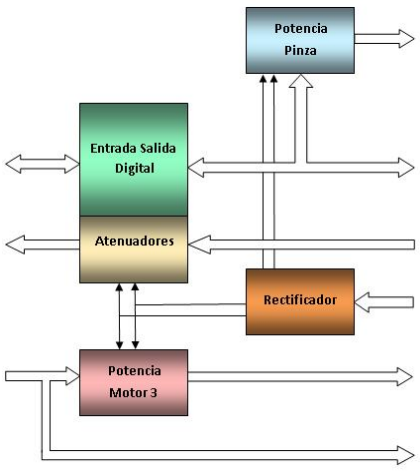

Fig. 8. Diagrama en bloques de la placa exterior

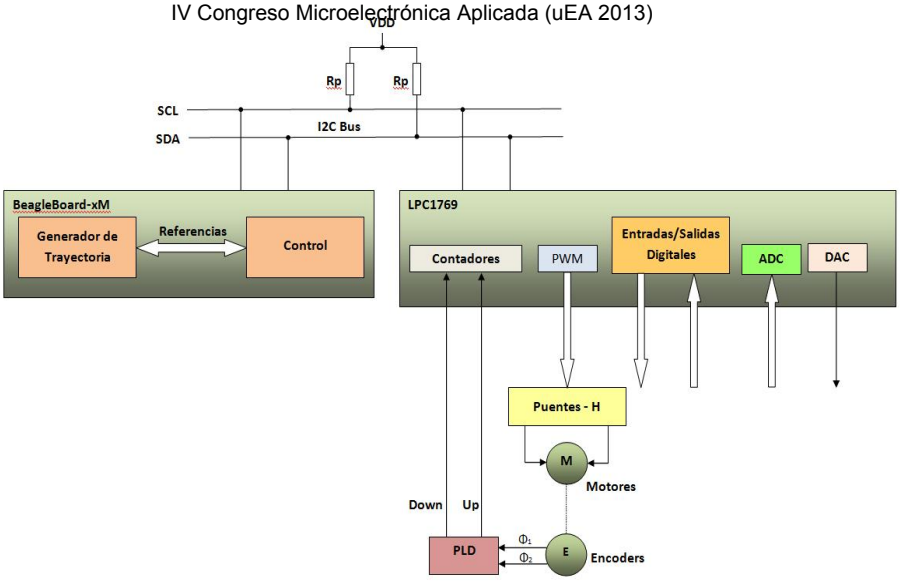

Fig. 9. Diagrama en bloques del sistema de servo control propuesto en este artículo

Con este nuevo diseño del hardware, el Gantry aumentará las posibilidades de comunicación con otros dispositivos, sistemas y sensores. Se podrá incorporar un sistema de visión que permita obtener información para la localización de objetos en tiempo real, mejorando su capacidad para adaptarse a distintas tareas y potenciando de esta manera su flexibilidad.

Además, el diseño propuesto reducirá el consumo de energía a la vez que la integración de componentes disminuirá los costos y los tiempos de fabricación y reparación, y aumentará la confiabilidad del sistema.

VII. CONCLUSIONES

En el presente trabajo se describe el hardware y software del controlador de un robot cartesiano didáctico, que utiliza QNX para obtener un control distribuido de tiempo real que cumple con las metas de tiempo y asegura la estabilidad del sistema. Se explica cómo es el proceso de generación automática de la trayectoria y se expusieron las razones por las cuales ésta debe ser generada en tiempo real a medida que se ejecuta el movimiento. Se justifican las razones por las cuales se propone un nuevo diseño para el hardware del controlador.

La conveniencia de conservar el sistema operativo de tiempo real QNX en el hardware propuesto permite aprovechar la experiencia adquirida en trabajos anteriores y cumplir con el objetivo de actualizar y mejorar el controlador.

REFERENCIAS

[1] R. Ramos, J. Yamamoto, F. Rendo, M. Anigstein, "Interfase de programación orientada a objetos para el control de ejes robóticos," XIV Simposio Nacional de Control Automático AADECA, Bs As, 1994.

[2] R. Ramos, C. Costas, C. S. Kang, D. S. Son, M. Anigstein, "Procesamiento Distribuido en tiempo Real. Aplicación a un controlador robotico," Rpic ´97 San Juan, 224-229, 1997.

[3] R. Ramos, E. Panciera, M. Lehmann, "Hardware de Control para la Plataforma del Robot Gantry-UBA," XVIII Congr. Argentino Control Automático, AADECA 2002, # 6, Buenos Aires, 2002.

[4] A. Mauro, R. Ramos, M. Anigstein, "Nueva Interfase Hombre Máquina para la Plataforma del Robot Gantry-UBA," Primer Congreso Argentino de Ingeniería Mecánica, I CAIM 2008. 1-3 Oct 2008, # 058 D. Bahía Blanca 2008.

[5] QNX RTOS 4.25 System Architecture. Note Control Systems Magazine, Vol 19, number 3, June 1999.

[6] J. J. Craig, Introduction to Robotics: Mechanics and Control, Pearson, Prentice-Hall, México. 2004.

[7] R. P. Paul, Robot Manipulators: Mathematics, Programming, and Control, The MIT Press. Cambridge Massachussets, 1981.

[8] Photon microGUI 1.13 Programmer's Guide.

[9] A. Mauro, M. Anigstein, "Sobre los sistemas de tiempo real y el cumplimiento de los tiempos de las tareas en un manipulador robótico," Segundo Congreso Argentino de Ingeniería Mecánica, II CAIM 2010. 16-19 Nov 2010, # 191 D. San Juan 2010.

[10] Beagleboard.org. (2013) Beagleboard-xm product details. [Online]. Available: http://beagleboard.org/hardware-xM

[11] QNX Neutrino Realtime Operating System *Building Embedded Systems*, Electronic edition published 2010, http://www.qnx.com/

[12] "NXP LPC1769." [Online]. Available: http://ics.nxp.com/lpcxpresso/~LPC1769

[13] J. Yiu. (2010). Cortex-M3, The definitive guide to the ARM. Newnes for Elsevier.

[14] R. Barry, Using the FreeRTOS Real Time Kernel - a Practical Guide - NXP LPC17xx Edition, www.freertos.org.