Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Controlador Empotrado para Plataformas Neumáticas de Dos Grados de Libertad empleando Arquitectura ARM9 y Sistema Operativo Linux

Autor: Enrique Eugenio Cepero Morfa

Tutor: M.Sc. Alleiny Machado Sosa

Santa Clara

2011

"Año 53 de la Revolución"

Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Controlador Empotrado para Plataformas Neumáticas de Dos Grados de Libertad empleando Arquitectura ARM9 y Sistema Operativo Linux

TRABAJO DE DIPLOMA

Autor: Enrique Eugenio Cepero Morfa
email: ecepero@uclv.edu.cu

Tutor: M.Sc. Alleiny Machado Sosa Prof. Asistente
Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV
email: alleini@uclv.edu.cu

Santa Clara

2011

"Año 53 de la Revolución"

Hago constar que el presente TRABAJO DE DIPLOMA fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

Enrique Eugenio Cepero Morfa — Fecha
Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Enrique Eugenio Cepero Morfa — Fecha
Autor

Boris Luis Martínez Jiménez, Dr.C — Fecha
Jefe del Departmento

Responsable ICT o J' de Carrera, (Dr.C., M.Sc. o Ing.) — Fecha
Responsable de Información Científico-Técnica

PENSAMIENTO

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado.
Un esfuerzo total es una victoria completa."

Mahatma Gandhi

AGRADECIMIENTOS

A todos mis compañeros y mi claustro de profesores por su aporte a mi formación profesional y humana.

A veces sentimos que lo que hacemos es tan solo una gota en el mar, pero el mar sería menos si le faltara una gota.

A mi tutor Alleiny Machado por su perseverancia y ayuda incondicional en este proyecto.

Me lo contaron y lo olvidé. Lo vi y lo entendí. Lo hice y lo aprendí.

A mi familia por ser el pedestal sobre el que me he desarrollado y sobre todas las cosas a mis padres por su consejo y sacrificio.

Lo que usted planta, usted cosechará.

SÍNTESIS

Desde hace varios años el Grupo de Automatización, Robótica y Percepción (GARP) ha venido trabajando en el desarrollo de algoritmos de control para robots paralelos accionados neumáticamente. Estos algoritmos, debido a la naturaleza altamente no lineal de los actuadores neumáticos, resultan ser de gran complejidad por lo que es necesario implementarlos físicamente en hardware de elevada capacidad de cómputo. Desde el punto de vista de la obtención del algoritmo de control se han alcanzado muy buenos resultados, no obstante, su implementación física no ha tenido aun una solución definitiva. En este trabajo se propone la implementación de dichos algoritmos en la tarjeta MBTI2440 (CPU) de fabricación nacional basada en el potente microcontrolador S3C2440 de SAMSUNG empleando Sistema Operativo GNU/Linux.

TABLA DE CONTENIDO

Página

Índice de cuadros

Cuadro Página

Índice de figuras

Figura Página

INTRODUCCIÓN

El desarrollo vertiginoso de la ciencia y la tecnología que se muestra hoy en día proporciona infinidad de productos y servicios en constante cambio a la sociedad, los cuales tienden a ser cada vez más competitivos en cuanto a precios y prestaciones. El desarrollo tecnológico en las ramas de la electrónica y la computación ha permitido el desarrollo de videojuegos, simuladores de vuelo y conducción entre otros.

Los simuladores de conducción han tenido mayor aceptación en los últimos tiempos puesto que permiten el aprendizaje de técnicas de conducción sin los riesgos que implicaría la práctica real. También permiten el aprendizaje minimizando los costos y reproducen de forma fiel las sensaciones de un vehículo real.

Para recrear estos mundos virtuales y hacerlos parecer reales se utilizan robots paralelos de varios grados de libertad, que pueden ser eléctricos, hidráulicos o neumáticos. En nuestro país el Centro de Investigación y Desarrollo de Simuladores CIDSIM (SIMPRO) ha construido plataformas neumáticas que requieren de un adecuado control para su eficiente explotación. La utilización de las mismas con actuadores electro-neumáticos aminora considerablemente el costo del simulador pero trae consigo el inconveniente de que controlar los actuadores neumáticos resulte ser más complicado por la compresibilidad del aire, fricciones y variaciones en la fuente de presión que da como resultado un modelo dinámico altamente no lineal. Esta compleja dinámica del sistema presupone la utilización de algoritmos de control no convencionales que obligan a utilizar hardware de elevada capacidad de cómputo.

Pese a la complejidad de los algoritmos de control para accionamientos neumáticos esta tecnología nos proporciona ventajas tales como altas razones de carga contra volumen, carga contra peso, grandes velocidades y fuerzas.

Anteriormente ya se han realizado trabajos relacionados con la implementación de controladores empotrados para plataformas neumáticas. Se desarrollaron controladores que utilizaron hardware como Flashlite (Machado, 2007) y dsPIC (Sosa, 2007). La implementación del controlador empotrado con Flashlite no cumplió con la exigencia de tiempo requerida puesto que no respondía con la inmediatez necesaria. El desarrollo del controlador empotrado que se hizo sobre dsPIC si respondía satisfactoriamente ante las exigencias de la plataforma pero esta tecnología no está actualmente a disposición del usuario final.

Tras un análisis de lo mencionado anteriormente podemos concretar de manera general nuestro problema. ¿Cómo implementar un controlador empotrado para simuladores de conducción con plataformas de dos grados de libertad con altas frecuencias de muestreo, algoritmos no convencionales de control y hardware a nuestro alcance?

Como autor del trabajo asumo como hipótesis que es posible implementar un controlador empotrado para simuladores de conducción con plataformas de dos grados de libertad con altas frecuencias de muestreo, algoritmos no convencionales de control y hardware de producción nacional.

En este trabajo se establece como objetivo de estudio la investigación de las diferentes tendencias del hardware empotrado para el control de plataformas de dos grados de libertad haciendo énfasis en la tarjeta MBTI2440 con arquitectura ARM9 de fabricación nacional y la utilización de Sistema Operativo (SO) GNU/Linux.

Por consiguiente el objetivo general de este trabajo es apoyar el desarrollo de los simuladores comercializados por SIMPRO elevando sus prestaciones y potenciando su desempeño al incorporar un controlador empotrado de alta capacidad de cómputo. Como objetivo específico en este trabajo planteamos el desarrollo de un controlador empotrado basado en la tarjeta MBTI2440 con arquitectura ARM9 y Sistema Operativo Linux que permita ejecutar algoritmos no convencionales para el control de plataformas neumáticas.

Para la realización del presente proyecto tendremos en cuenta la siguiente metodología de trabajo:

- Primeramente se realizará una revisión de la bibliografía especializada con el fin de disponer de elementos de juicio para decidir las vertientes en las que encaminaremos el trabajo.
- Se realizará una recopilación de textos técnicos y programas necesarios con el objetivo de poner a punto la tarjeta MBTI2440 para su explotación como controlador de ejes empotrado en plataformas neumáticas de simuladores de conducción.
- Posteriormente se verificará a partir de los resultados obtenidos el desempeño de la tarjeta.

En este trabajo para la concreción de nuestros objetivos nos planteamos las siguientes tareas:

- Comunicación de la tarjeta MBTI2440 con interfaz de entrada-salida analógicas (Multi I/O Expansion Board).
- Compilación del kernel de Linux instalado en la tarjeta para activar funcionalidades no presentes.
- Utilización de las interrupciones del Sistema Operativo (SO) Linux para lograr las exigencias de tiempo real del algoritmo a implementar.
- Programación de interfaz gráfica para test del sistema.

El presente trabajo contará con la siguiente organización:

En el capítulo uno se realizará una revisión bibliográfica sobre las plataformas o robots paralelos desde su surgimiento hasta las plataformas de fabricación nacional. Posteriormente tocaremos el tema de los algoritmos de control para plataformas neumáticas prestando especial atención al algoritmo de Rubio (Rubio, 2006). Finalmente hablaremos de las distintas variantes de hardware utilizados para el control de las plataformas explicando las ventajas que brinda la utilización de hardware empotrado para el control de esta.

En el capítulo dos se brindarán detalles de las tarjetas MBTI2440 y Multi I/O Expansion Board, justificaremos la necesidad de implementar un hardware adicional del que daremos información. Finalmente abordaremos la temática del SO GNU/Linux y dejaremos definida la configuración de nuestro hardware.

En el capítulo tres explicaremos el procedimiento para la compilación e instalación del kernel de Linux en la tarjeta MBTI2440. Mostraremos el hardware adicional implementado así como su circuito impreso. Para concluir expondremos los resultados del trabajo, haremos un breve análisis económico y recomendaremos algunas tareas sobre las cuales se debe seguir trabajando.

Capítulo 1
MARCO TEÓRICO

1.1. Introducción.

En este primer capítulo realizaremos una revisión bibliográfica relacionada con las plataformas destinadas a simuladores en sentido general, Fig. 1–1 así como sus aplicaciones y bondades a la sociedad. Abordaremos los diferentes tipos de plataformas con el fin de desarrollar simuladores de conducción y profundizaremos en las plataformas construidas por SIMPRO que son las que nos ocupan en el presente trabajo. Posteriormente realizaremos un breve bosquejo relacionado con el modelo matemático de estas plataformas y sus posibles estrategias de control refiriéndonos principalmente a la desarrollada por Rubio (Rubio, 2006). Por último daremos una mirada al hardware con el que trabajaremos y haremos una revisión de los que anteriormente se utilizaron en nuestro grupo investigativo GARP.

1.1.1. Definiciones

Un simulador de conducción cuenta con un mundo virtual y mandos reales con los que interactúa el usuario. Estos mandos a su vez interactúan con la plataforma sobre la que se ha colocado la cabina del simulador. Esta plataforma puede ser catalogada como un robot paralelo. Un robot en sentido general es una máquina integrada por 3 componentes: posee una parte mecánica, una parte electrónica y una parte de procesamiento (Florez, 2007). Un robot paralelo Fig. 1–2 es definido como aquella estructura en la cual, el efector final está unido a la base por más de una cadena cinemática (Silva, 2005). Siendo las cadenas cinemáticas en el caso de las plataformas pistones neumáticos o hidráulicos. El número de

Figura 1–1: Simulador de Conducción.

pistones define los grados de libertad de la plataforma, siendo un diseño muy popular el Stewart (Stewart, 1965-66) de seis grados de libertad.

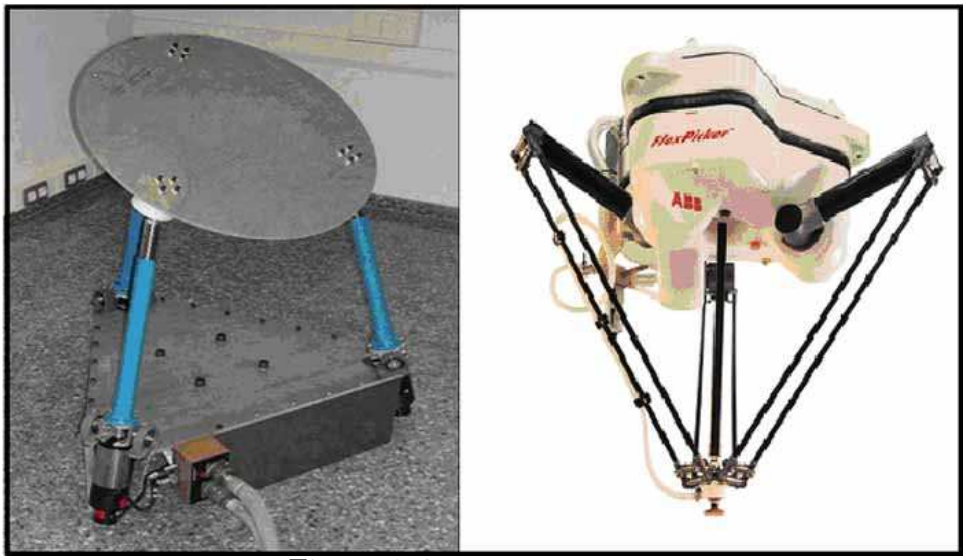

Figura 1–2: Robots Paralelos.

1.1.2. Arquitectura y su Evolución

La ya mencionada plataforma de Stewart Fig. 1–3 fue propuesta en 1965 para el desarrollo de simuladores de vuelo. Esta contaba con una base triangular que mediante articulaciones esféricas se unían a pistones los cuales se fijaban a la parte superior de la

plataforma también triangular. Posteriormente esta plataforma fue modificándose pasando de una base triangular a una hexagonal y luego la base y la parte superior de la plataforma fueron hexagonales, manteniendo siempre los seis grados de libertad.

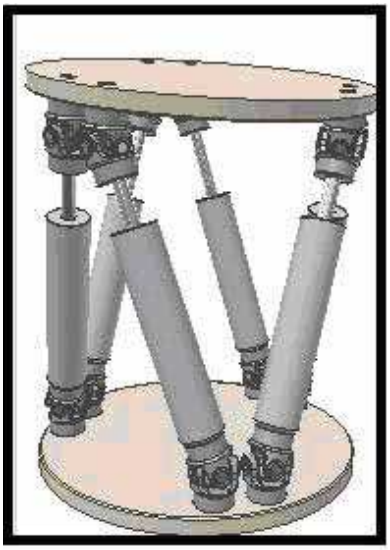

Figura 1–3: Plataforma Stewart.

Ya en la década del 80 comienzan a desarrollarse proyectos para plataformas de dos y tres grados de libertad las cuales tenían como ventajas un mayor espacio de trabajo, un menor coste, un diseño mecánico más sencillo y se podían controlar más fácilmente. Las plataformas de 4 y 5 grados de libertad no vinieron a desarrollarse hasta la década del 90 puesto que su cons- trucción no puede realizarse con cadenas cinemáticas iguales.

Como una variante de las plataformas anteriormente mencionadas tenemos las desarrolladas por SIMPRO, plataformas de 2 y 3 grados de libertad que ofrecen una solución económica y eficaz para el desarrollo de simuladores de conducción. La plataforma de dos grados de libertad Fig. 1–4 posee un pivote ubicado en el centro de la plataforma, consta además de dos pistones ubicados perpendiculares al pivote y con una amplitud de 90 grados entre sí.

El origen de coordenadas para las medidas de longitud y ubicación del centro de masa (CM) se establecen en el pivote central. Cada articulación electro-neumática está formada por un cilindro FESTO DNC-100-400 gobernado por una válvula proporcional de flujo

Figura 1–4: Plataforma de Dos Grados de Libertad desarrollada por Simpro.

FESTO MPYE-5-3/8 y cuya posición se mide con un potenciómetro lineal FESTO MLO-POT-450. Los datos mecánicos más importantes de la plataforma del simulador se ofrecen en el cuadro 1–1.

Cuadro 1–1: **Datos Mecánicos de la Plataforma de Dos Grados de Libertad.**

Masa Total de la Cabina	510	Kg
Posición del CM en Z	480	mm
Posición del CM en X	100	mm
Posición del CM en Y	60	mm
Distancia del Origen a cada Cilindro	500	mm
Elongación de los Cilindros	150	mm
Ángulos de Ladeo y Cabeceo	0.26	rad

Esta plataforma desarrollada por SIMPRO requiere de un adecuado algoritmo de control de movimiento y de un hardware capaz de satisfacer las exigencias requeridas por el simulador.

1.2. Análisis de métodos de Control para Actuadores Neumáticos.

En el mundo actual cada vez los actuadores neumáticos ganan más terreno frente a actuadores eléctricos e hidráulicos por sus bajos costes, buena relación fuerza-peso, altas velocidades y ya se pueden aplicar algoritmos de control que logran controlar de forma eficiente estos actuadores de compleja dinámica.

La complejidad del control de estos actuadores se debe a fricciones, al cambio de la dinámica del actuador según su posición y carga que soporta. Estos factores conspiran contra la dinámica de los pistones presentándose un modelo altamente no lineal que exige de la implementación de algoritmos de control complejos, obviándose las posibilidades de gobernar estos actuadores con algoritmos PID relativamente sencillos.

1.2.1. Revisión bibliográfica.

Algunos de los algoritmos que podemos citar son (Brun, 2000) donde a partir de lograr estructurar parte del comportamiento del sistema se plantea una estrategia de control no lineal. Otro algoritmo es el de (Uchikado, 2000) que propone un control por ubicación de polos adaptables con linealizador neuronal. También hay algoritmos no lineales que de una forma u otra realimentan aceleración y su derivada garantizando robustez, (Schulte, 2003)(Pandian, 2002)(Sakamoto, 2006). También se encuentra el método planteado en Igor y German (Igor, 2006) donde se propone la ubicación de polos realimentando posición, velocidad y aceleración. Tenemos también el método planteado por Rubio en el cual se aplica un algoritmo de control lineal por ubicación de polos donde se busca compensar constantemente los polos complejos conjugados que dependen de la posición del actuador en la función de transferencia de lazo abierto de la planta. Este método planteado por Rubio (Rubio, 2006) fue puesto en práctica en las plataformas de 2 y 3 grados de libertad producidas por SIMPRO arrojando resultados satisfactorios, esta variante tiene a su favor que no realimenta presión, haciéndola viable en su aplicación por su sencillez y robustez.

1.3. Hardware para control de Plataformas Robóticas.

Como hemos visto en el epígrafe anterior existe una variada gama de algoritmos que permiten controlar plataformas de simulación de conducción neumáticas, que van perfeccionándose con relativa agilidad si lo comparamos con el hardware que se utiliza para el control de estas plataformas. Ya sea en plataformas con actuadores eléctricos o hidráulicos las tendencias generales son de colocar una PC para el cálculo del mundo virtual y una PC industrial para el cálculo de la cinemática inversa de la plataforma Fig. 1–5. Existen casos en los que la PC industrial es la encargada de convertir los datos analógicos en digitales y otros en los que se utilizan conversores en las entradas-salidas de la PC para aligerar su carga de trabajo ya que tiene que realizar los cálculos dinámicos y dar salidas en tiempo real. Existe también la posibilidad de utilizar dispositivos de hardware potentes y compactos con gran capacidad de cómputo tales como micro-controladores capaces de realizar el control de los actuadores electro-neumáticos. De esta forma se logra implementar una variante comercial para estas plataformas y aminorar los costes pues se elimina la necesidad de una PC industrial.

Anteriormente para las plataformas desarrolladas por SIMPRO nuestro grupo investigativo GARP ha realizado investigaciones en busca de un hardware capaz de controlar satisfactoriamente las plataformas. Primeramente se hicieron pruebas utilizando el algoritmo de Rubio sobre tarjetas Flashlite 186 (Machado, 2007) Fig. 1–6 los cuales no brindaban los resultados deseados puesto que la respuesta del microprocesador no presentaba la agilidad necesaria y no se ofrecían sensaciones cercanas a la realidad al conductor del simulador.

Luego se implementó un hardware empotrado usando dsPIC (Sosa, 2007) Fig. 1–7 el cual cumplió satisfactoriamente los requerimientos de tiempo real de la plataforma además de ser una variante económica no obstante esta solución no fue definitiva debido a la carencia por parte de SIMPRO de los componentes implicados.

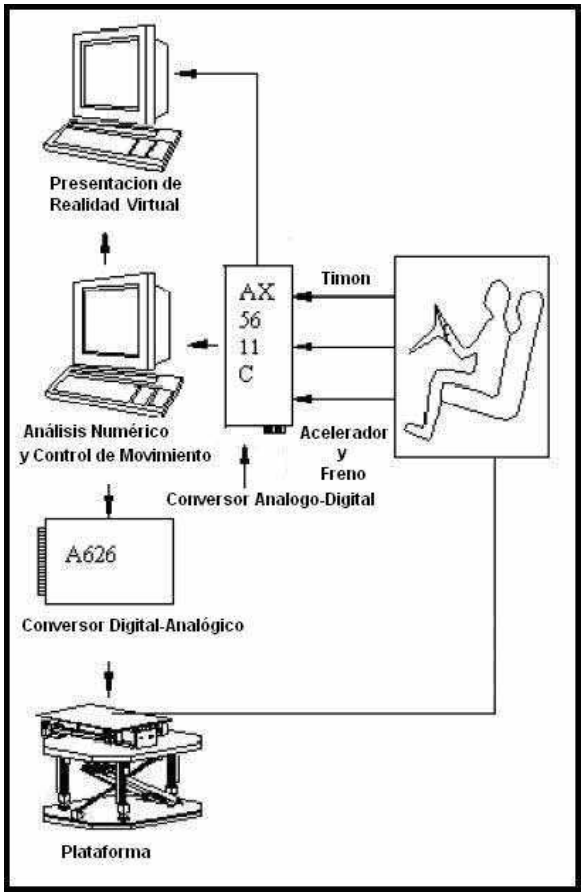

Figura 1–5: Controlador Empotrado basado en CAD y PC Industrial.

La variante propuesta en este trabajo además de superar en potencia y prestaciones a la solución con dsPIC, es de fabricación nacional lo que la hace fácilmente asequible al usuario final (SIMPRO).

1.4. Consideraciones Finales del Capítulo

Las plataformas de dos grados de libertad presentadas por SIMPRO han sido utilizadas anteriormente en el desarrollo de simuladores de conducción, con buenos resultados. Para el control de estas se ha utilizado el algoritmo de Rubio, el cual obliga a utilizar dispositivos de cómputo potentes. Nosotros proponemos la implementación de un hardware empotrado para el cálculo del control de la plataforma, siendo esta una solución económica y factible puesto que el hardware propuesto, la tarjeta MBTI2440, es de fabricación nacional.

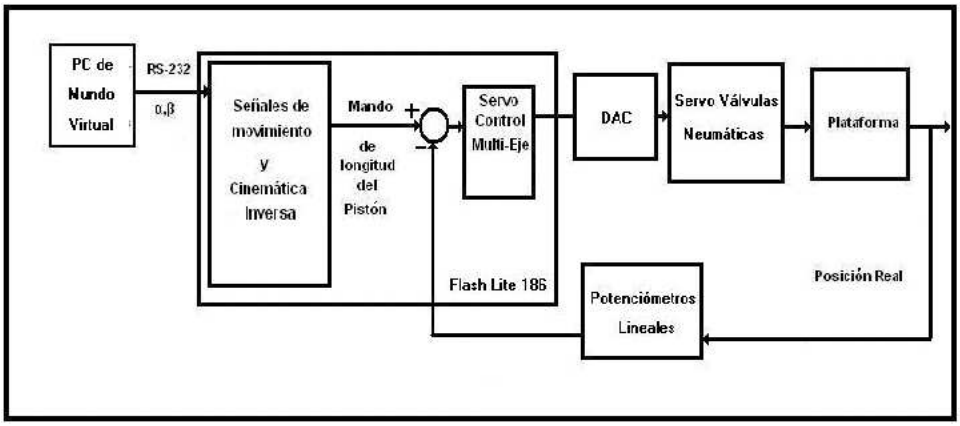

Figura 1–6: Controlador Empotrado basado en Flashlite 186.

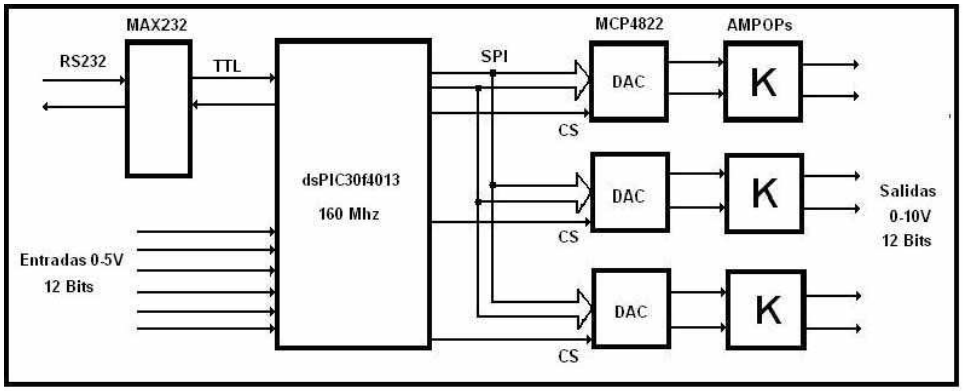

Figura 1–7: Controlador Empotrado basado en dsPIC 30f4013.

Capítulo 2
HARDWARE Y SOFTWARE EMPLEADO

2.1. Introducción.

En este capítulo profundizaremos en los elementos de hardware y software implicados en el desarrollo de este trabajo. También abordaremos detalles acerca de las estrategias usadas para enfrentar algunas dificultades que se presentaron en la resolución de la tareas.

Primeramente mostramos las características técnicas de la tarjeta MBTI2440 y la Multi I/O Expansion Board. Seguidamente mostramos la interfaz necesaria para la comunicación entre la MBTI2440 y las entradas-salidas analógicas. Finalmente abordaremos lo referente al Sistema Operativo empleado.

2.2. Tarjeta MBTI2440

La tarjeta MBTI2440, es una computadora compacta con grandes prestaciones, poco peso, bajo consumo de potencia (400 mA trabajando con 5 VDC) y con dimensiones de 104 x 73 mm, ideal para ser utilizada como hardware empotrado en cualquier proyecto. Este hardware brinda múltiples ventajas sobre los utilizados anteriormente, con este fin el Cuadro 2–1 muestra una comparación entre nuestra tarjeta, el dsPIC 30f4013 y la Flashlite 186.

La tarjeta MBTI2440 es considerada como una pequeña PC puesto que cuenta con el procesador SAMSUNG S3C2440A con arquitectura ARM9 (Sam, 2004) el cual opera a una frecuencia de trabajo de 400 MHz. La arquitectura ARM posee un conjunto de instrucciones simple pero eficientes que permiten un tamaño de silicio compacto y ofrece alta velocidad de ejecución a bajo consumo. Además son microcontroladores que ofrecen una inmejorable relación precio-prestaciones (CAPEL, 2008). El procesador S3C2440A soporta múltiples interfaces tales como: 4 puertos usb host, un puerto usb device, 3 puertos serie, posee interfaces para tarjeta SD/MMC, pantalla táctil, LCD con resolución hasta 1024x768 bpp entre otros. (ICI, n.d.). Anexo A.

Cuadro 2–1: **Comparación entre distintos tipos de Hardware Empotrados .**

	dsPIC 30f4013	Flashlite 186	MBTI2440
Frecuencia de Trabajo	120 MHz	33 MHz	400 MHz
Memoria ROM	1 KB	512 KB	64 MB
Memoria RAM	2 KB	512 KB	32 MB
Sistema Operativo	no utiliza	X-DOS	Linux

En las Fig. 2–1, 2–2 se pueden ver los conectores de la tarjeta MBTI2440 y a continuación explicamos sus funciones. El puerto CON2 ubicado en la parte superior de la tarjeta posee la conexión para cámara, el CON3 es el conector por el cual se realiza la conexión para programar las memorias flash de nuestra tarjeta. Los puertos CON5 y

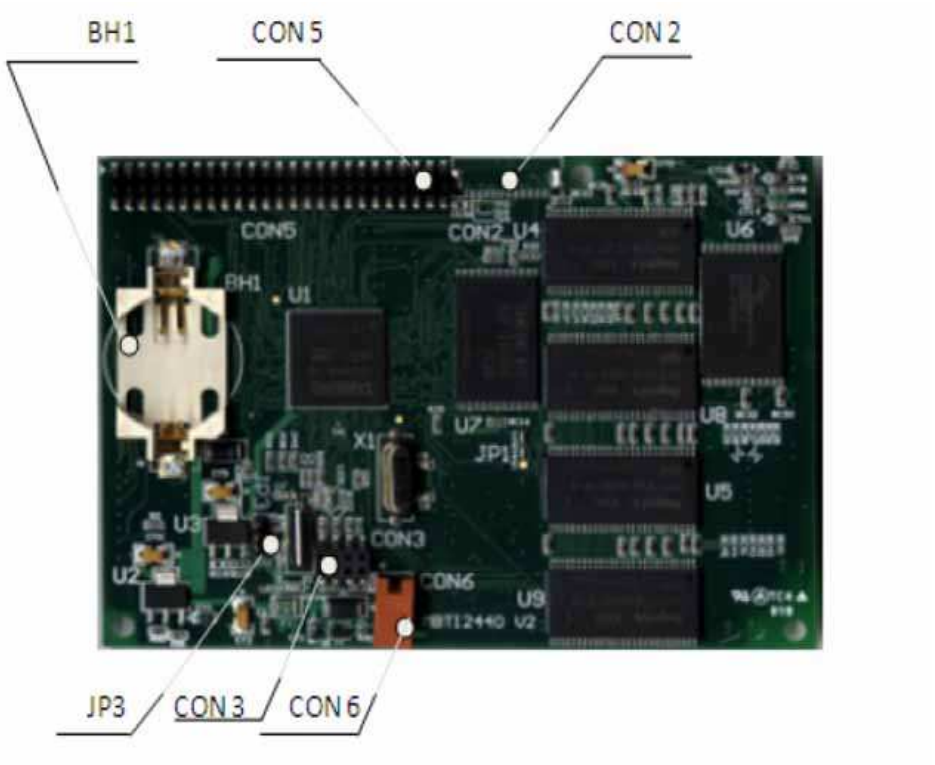

Figura 2–1: Tarjeta MBTI2440 conectores cara superior.

CON5a son los que permiten la salida de video y poseen la interfaz para pantalla táctil. El conector CON6 es el que permite la alimentación de la tarjeta, también tenemos el conector BH1 que porta la batería de tipo CR2032 que da sustento al reloj de tiempo real. La tarjeta también cuenta con un conector JP3 que determina que memoria se va a emplear, cuando JP3 está abierto se accede a la memoria NOR Flash y cuando está cerrado se accede a la memoria NAND Flash.

El puerto CON1 contiene las líneas que se encargan del manejo del audio, la red, la memoria SD y los puertos usb. El terminal CON4 es una extensión de entradas-salidas digitales con 50 pines (2x25) y es el que utilizaremos en nuestro proyecto para comunicarnos con el simulador.

Esta tarjeta se puede conectar de manera sencilla a nuestra PC a través del puerto serie y usb, de esta manera podemos leer o escribir en ella utilizando el software DNW lo cual es una ventaja a la hora de transferirle programas o información haciéndose este procedimiento de forma ágil y segura. También el hecho de que sobre esta tarjeta podamos instalar un SO en nuestro caso GNU/Linux brinda la posibilidad de implementar una

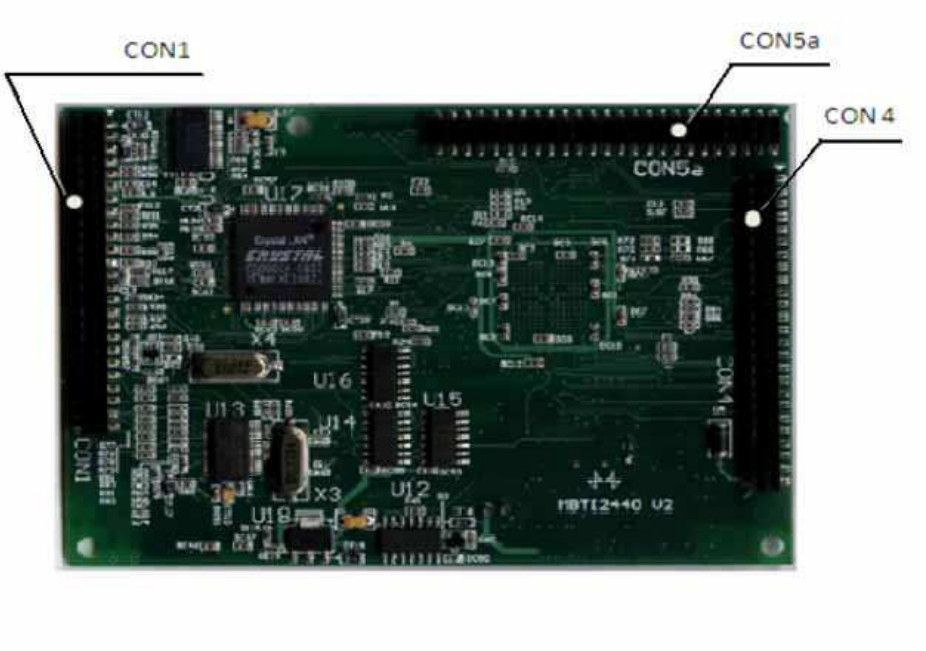

Figura 2–2: Tarjeta MBTI2440 conectores cara inferior.

interfaz gráfica, introducir información directamente desde el teclado sobre ella, en fin las potencialidades de nuestro hardware lo hacen notablemente superior a los hardware utilizados anteriormente para los simuladores de conducción que nos ocupan.

La MBTI2440 está acoplada a una tarjeta base de desarrollo Fig. 2–3 la cual posee los componentes físicos que permiten las conexiones, esta tarjeta posee los conectores usb, puerto serie, memoria SD, Ethernet, audio in, audio out y alimentación.

Figura 2–3: Tarjeta Base de la MBTI2440.

2.3. Hardware Adicional

Los requerimientos de hardware de este proyecto hacen necesario la utilización de dos entradas analógicas de 12 bits y dos salidas analógicas también de 12 bits y 0-10V. Estas entradas-salidas no están presentes en la MBTI2440 y por tal razón proponemos la utilización de la tarjeta Multi I/O Expansion Board Fig. 2–4 para un primer prototipo, que será sustituida por un diseño nacional de aceptarse este trabajo como solución definitiva al problema del control articular de las plataformas neumáticas de los simuladores de conducción fabricados por SIMPRO.

Esta tarjeta dispone de un UART (Unidad Sincrónica de Transmisión y Recepción de datos.) 4 Conversores Análogo-Digitales (CAD) con 12 bits de resolución, 2 Conversores Dígito-Analógicos (CDA) igualmente con 12 bits de resolución y un voltaje máximo a la salida de 4.095V. Esta tarjeta está dotada del circuito integrado BBADS7841 que es un CAD, el integrado AD5320 que es un CDA y el componente MAX3100 que es el encargado

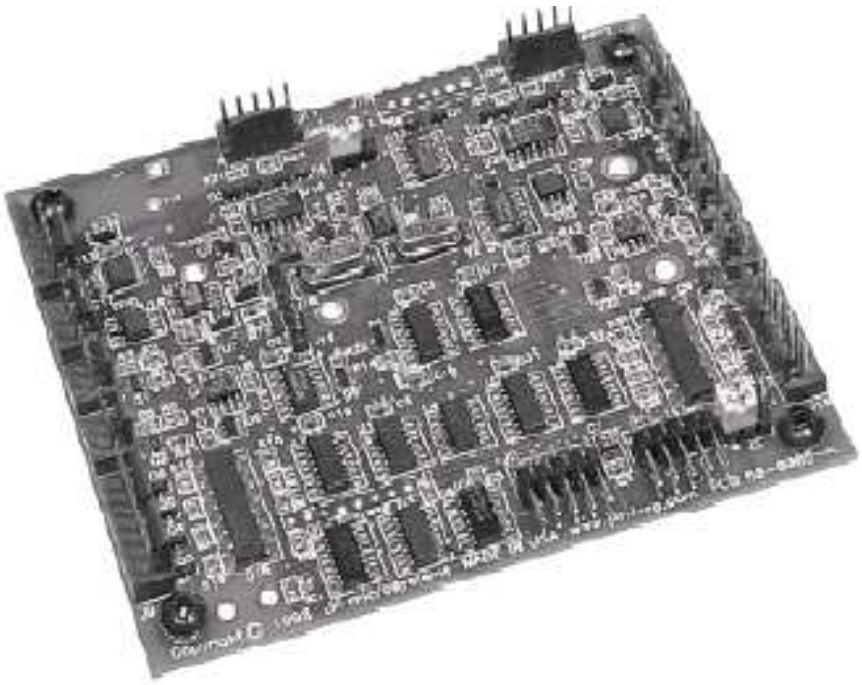

Figura 2–4: Tarjeta Multi I/O Expansion Board.

de garantizar la comunicación UART.(JK , n.d.) Utilizaremos esta tarjeta porque que la teníamos disponible en nuestro grupo investigativo y garantiza con sus 12 bits de resolución una precisión aceptable para las señales de intercambio entre los actuadores y nuestro hardware.

2.3.1. Componentes Electrónicos Adicionales

Para poder comunicar correctamente la tarjeta MBTI2440 con las entradas-salidas analógicas presentes en la tarjeta Multi I/O Expansion Board es necesario diseñar y construir un circuito de interfaz puesto que los niveles de tensión no son compatibles (3.3V en MBTI2440 y 5V en la Multi I/O Expansion Board). Para resolver esta dificultad utilizaremos el circuito integrado (CI) conversor de niveles 74LS244. Además implementaremos un amplificador DC-DC para acoplar los niveles de tensión entre la salida de Multi I/O Expansion Board (4.095V) y las electroválvulas (10V). Con este fin utilizaremos el circuito integrado LM324 el cual contiene 4 amplificadores operacionales de los cuales solamente trabajaremos con dos. Estos CI los tenemos disponibles en nuestro grupo investigativo por lo que no serán un obstáculo en la realización de nuestro proyecto.

Una vez los niveles de tensión (3.3V CPU y 5V Multi I/O Expansion Board) han sido corregidos hay que proceder a comunicar correctamente las tarjetas MBTI2440 y Multi I/O Expansion Board. Para comunicar con éxito las tarjetas MBTI2440 y Multi I/O Expansion Board es necesario transmitir las señales de Dato, Clock (CLK), Reset y Chip Selec (CS), puesto que la tarjeta Multi I/O Expansion Board necesita todas estas señales para su correcto funcionamiento. Las señales se transmitirán hacia el puerto J2 de la Multi I/O Expansion Board utilizando los pines del uno al seis ya que además se le conectan los potenciales de VCC y GND. Cuadro 2–2.

Cuadro 2–2: **Señales enviadas al conector J2 de la Multi I/O Expansion Board.**

Señal	Pin	Pin	Señal
Dato	1	2	CLK
Reset	3	4	CS
VCC	5	6	GND

2.4. Sistema Operativo GNU/Linux

En esta tarjeta instalaremos SO GNU/LINUX con la distribución 2.6.29 del kernel. El núcleo o kernel GNU/LINUX se encarga del arranque del sistema y una vez este ya es utilizable por los programas y los usuarios, gestiona los recursos de la PC en forma de gestión de la memoria, sistemas de ficheros, entrada-salida, procesos e intercomunicación de procesos (Jorba, 10). Este SO es distribuido gratuitamente y está disponible su código fuente gracias a su creador Linus Torvalds que lo hizo público en 1991, el núcleo expuesto en aquel entonces ha sido ampliado y mejorado hasta obtener lo que tenemos hoy en día, Fig. 2–5 un SO eficiente y competente capaz de hacerle frente a sistemas propietarios como Macintosh y Windows. La versión 2.6.29 del kernel que utilizaremos está disponible gratuitamente en el sitio FriendlyARM en el apartado de descargas, también están disponibles todas las herramientas que utilizaremos, el compilador cruzado arm-linux-gcc-4.3.2 que nos permitirá compilar para la plataforma ARM y el software DNW que nos permite comunicarnos con nuestro hardware. La elección de SO GNU/Linux para este proyecto se debe a que como ya se dijo es un SO gratuito disponible en múltiples sitios en la web, además su código fuente es conocido y configurable, lo que permite adaptarlo a nuestros propósitos.

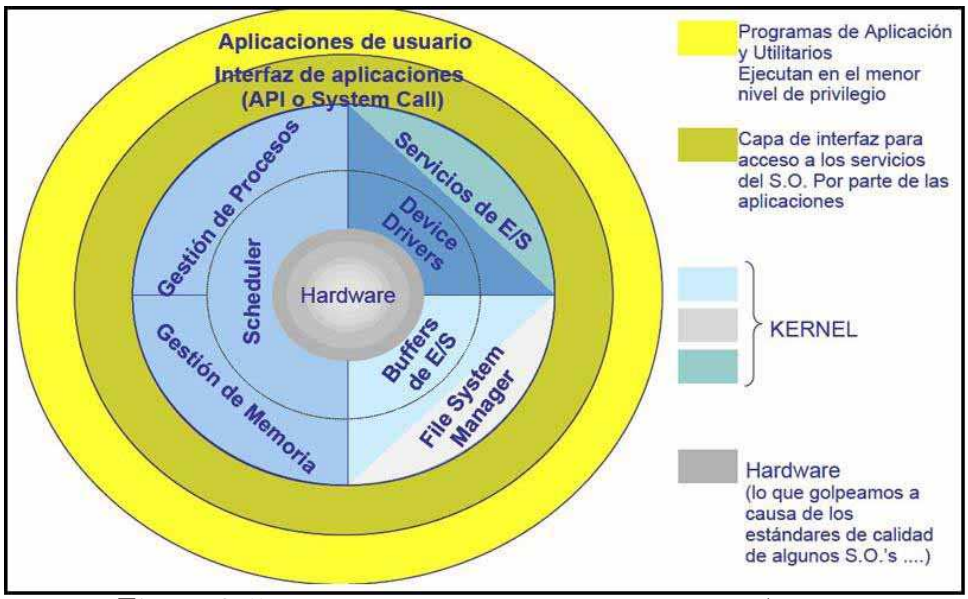

Figura 2–5: Esbozo del Funcionamiento SO GNU/Linux.

Los sistemas GNU/Linux son muy estables y pueden leer ficheros de casi cualquier sistema de ficheros, además pueden comunicarse por red con muchos otros SO, para recibir

o brindar servicios. Estos SO están además ampliamente soportados y existen infinidad de parches y actualizaciones para estos sistemas. También hay variedad de foros a los que acudir ante un problema determinado. Actualmente ya hay importantes fabricantes como HP, Epson, NVIDIA, ATI que están cediendo sus drivers para incorporarlos a estos SO.

Otra de las razones por la que nos inclinamos a trabajar con este SO es que los kernel 2.6.xx permiten el trabajo con Tiempo Real (RT) con resoluciones de hasta 1 ms lo cual es imprescindible en este proyecto (Lucero, 2006). RT del inglés Real Time se refiere a las exigencias que impone una tarea determinada para su correcto funcionamiento. Un sistema informático en tiempo real es aquel en que la corrección del resultado depende tanto de su validez lógica como del instante en el que se produce (Munoz, 2002).

Una de las premisas de este proyecto es lograr el trabajo en RT del hardware que se pretende implementar, puesto que para lograr una sensación semejante a la real en el simulador de conducción se debe cerrar el lazo de control con la mayor prontitud posible.

2.5. Consideraciones Finales del Capítulo

En este capítulo hemos detallado el software y hardware que utilizaremos en nuestro trabajo, de manera general se puede decir que cumplen con las exigencias que imponen las características del proyecto. Todos los componentes de hardware expuestos anteriormente los tenemos disponibles por lo que no hay ninguna traba que nos impida llevar a cabo nuestro trabajo. De manera general nuestro hardware tendrá la siguiente configuración Fig.2–6.

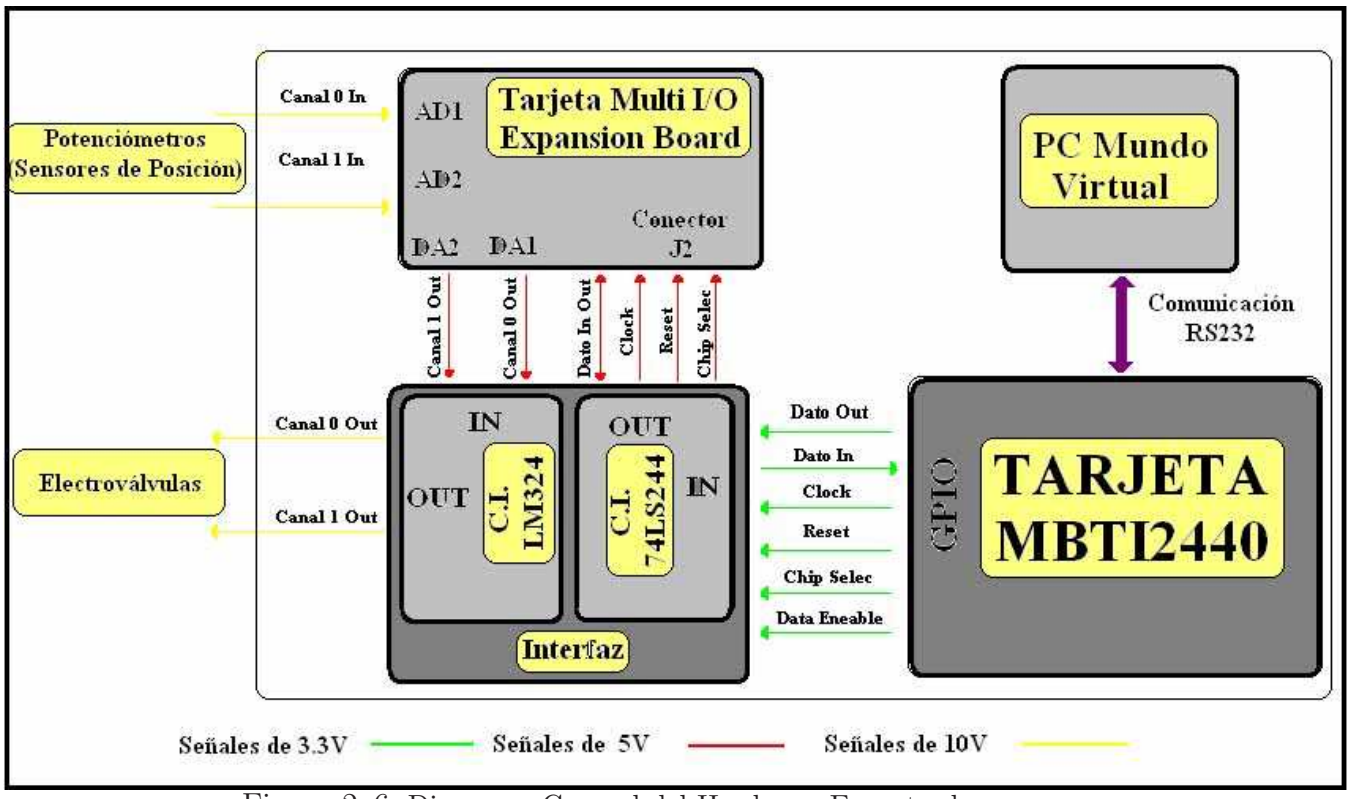

Figura 2–6: Diagrama General del Hardware Empotrado.

Capítulo 3
IMPLEMENTACIÓN DEL SISTEMA

3.1. Introducción

En los capítulos anteriores hemos brindado información a cerca de los hardware empotrados, los robots paralelos, y los algoritmos de control de plataformas de pistones neumáticos, quedando como opciones asequibles y óptimas la utilización de la tarjeta MBTI2440 con el algoritmo de Rubio (Rubio, 2006). También realizamos un esbozo de los métodos que emplearemos para lograr la funcionalidad del hardware, complementándolo con la tarjeta Multi I/O Expansion Board y otros elementos de hardware de los cuales se brindaron sus características.

Finalmente en este capítulo plasmaremos como se llevaron a cabo las tareas propuestas y los resultados que obtuvimos con nuestro controlador.

3.2. Hardware del Sistema

La Fig. 3–1 muestra el sistema final luego de interconectar los elementos de hardware mencionados en el capítulo anterior.

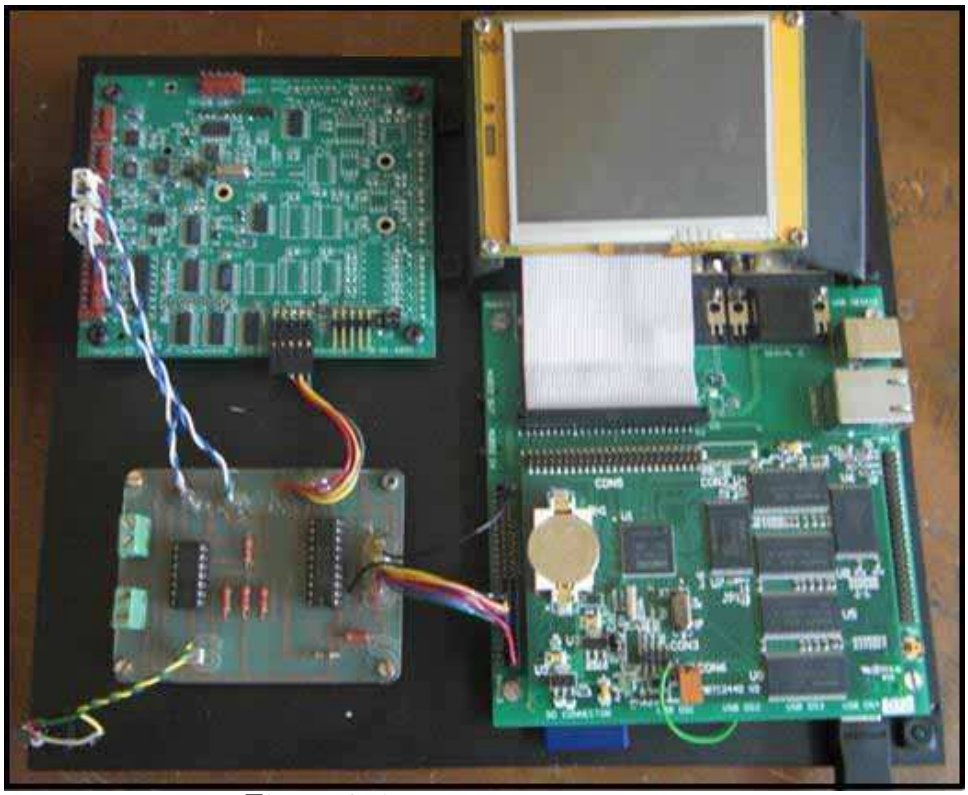

Figura 3–1: Controlador Empotrado.

La conexión entre la CPU y las entradas-salidas analógicas se realizó utilizando los puertos F y G presentes en el conector CON4 de la tarjeta MBTI2440. El siguiente cuadro 3–1 muestra como se utilizaron dichos pines.

Cuadro 3–1: **Función y Ubicación de los Pines del CON4 para Transmisión y Recepción de Datos.**

Función	Posición Lógica	Número del Pin
Dato-Out	Bit 0 Puerto F	30
Dato-In	Bit 5 Puerto F	31
CLK	Bit 6 Puerto F	32
Reset	Bit 7 Puerto F	33
CS	Bit 2 Puerto F	34
Dato-Eneable	Bit 3 Puerto G	35

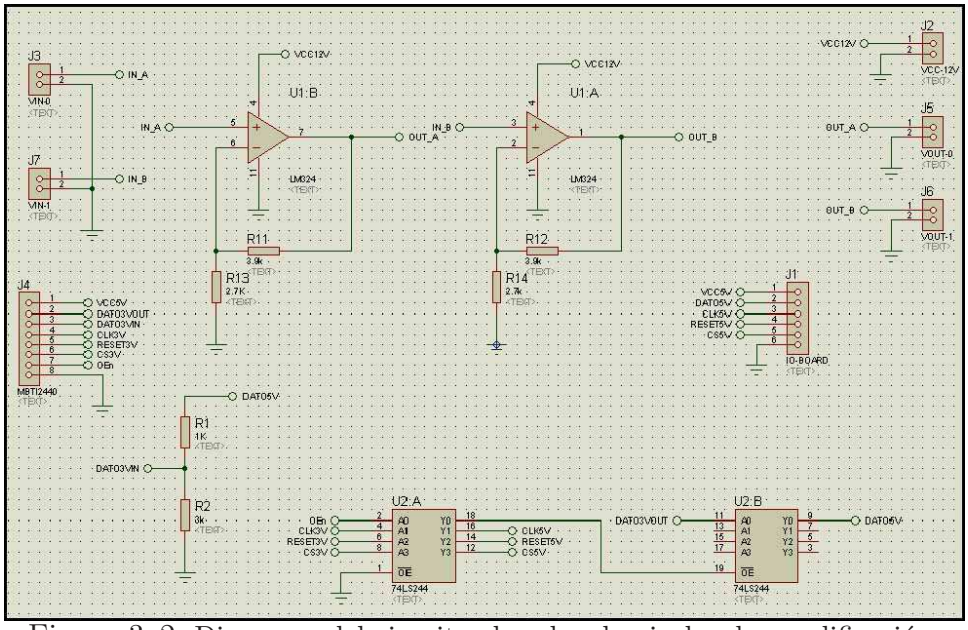

Figura 3–2: Diagrama del circuito elevador de nivel y de amplificación.

Estas señales presentan niveles de tensión de 3.3V por lo que fue necesario diseñar una interfaz que excitara correctamente la tarjeta Multi I/O Expansion Board la cual utiliza niveles de tensión de 5V, esto se logró con el CI 74LS244. Las figuras siguientes muestran el circuito diseñado Fig. 3–2, el circuito impreso Fig. 3–3 y el resultado final respectivamente Fig. 3–4. El diseño de la interfaz fue realizado utilizando el Proteus Profesional de Labcenter Electronics. Esta interfaz incluye además dos amplificadores basados en el amplificador operacional LM324 necesarios para aumentar el rango de salida de los conversores dígito-analógicos ya que estos alcanzan 4.095V máximo y las electroválvulas operan de 0-10V.

Las señales que controlan la Multi I/O Expansion Board son:

- Dato, esta señal para la Multi I/O Expansion Board es bidireccional. La señal debe ser acondicionada para poder ser manejada por la CPU. Para esto se utilizan, en un sentido (CPU hacia I/O), uno de los buffers presentes en la 74LS244 y el divisor de tensión formado por R1 y R2 (I/O hacia la CPU). Fig. 3–2.
- Clk (reloj de datos serie), es la encargada de sincronizar la transmisión y recepción de datos por parte de la CPU. También es amplificada por uno de los buffers presentes en la 74LS244.

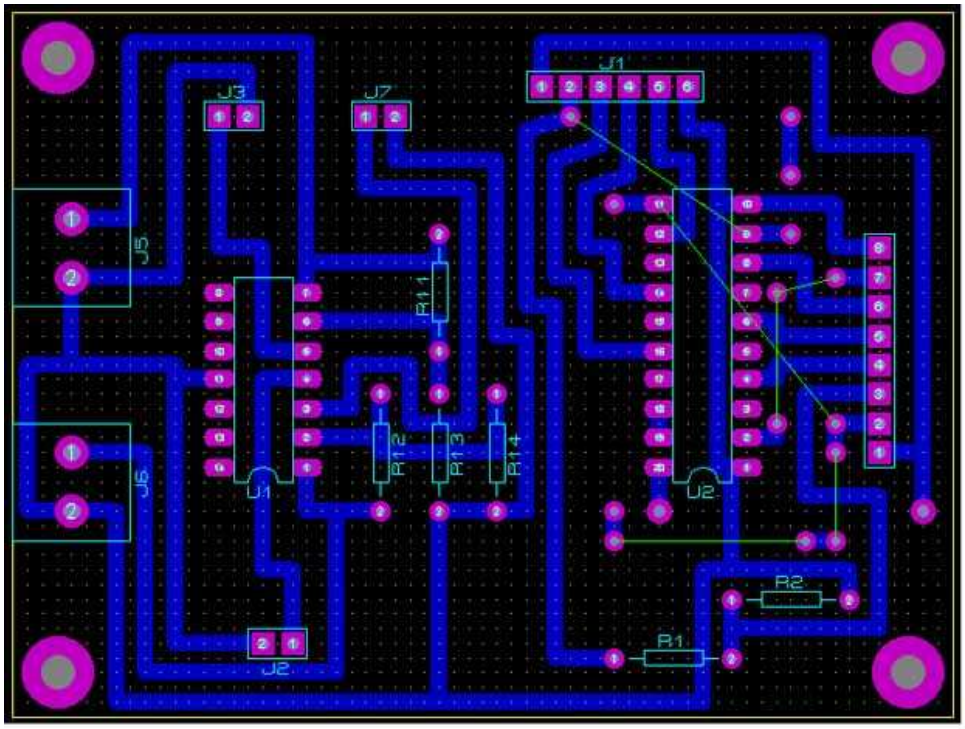

Figura 3–3: Diseño del circuito impreso.

- Reset, esta señal, activa en bajo, es necesaria para el comienzo de cualquier transferencia de datos entre la CPU y la Multi I/O Expansion Board. También pasa a través de uno de los buffers presentes en la 74LS244.
- CS (Chip Select), esta señal, activa en bajo, se utiliza para seleccionar la Multi I/O Expansion Board. También pasa a través de uno de los buffers presentes en la 74LS244.
- Data-Eneable, esta señal, activa en bajo, es la encargada de liberar la linea de datos por parte de la CPU para que su control sea tomado por la Multi I/O Expansion Board. En otras palabras, como la Multi I/O Expansion Board presenta una sola linea de datos que es bidireccional y la MBTI2440 presenta dos líneas de datos (dato-in y dato-out) es necesario que la linea de datos de salida de la CPU sea desactivada (puesta en alta impedancia) para evitar corto circuito cuando la Multi I/O Expansion Board este enviando datos hacia la CPU.

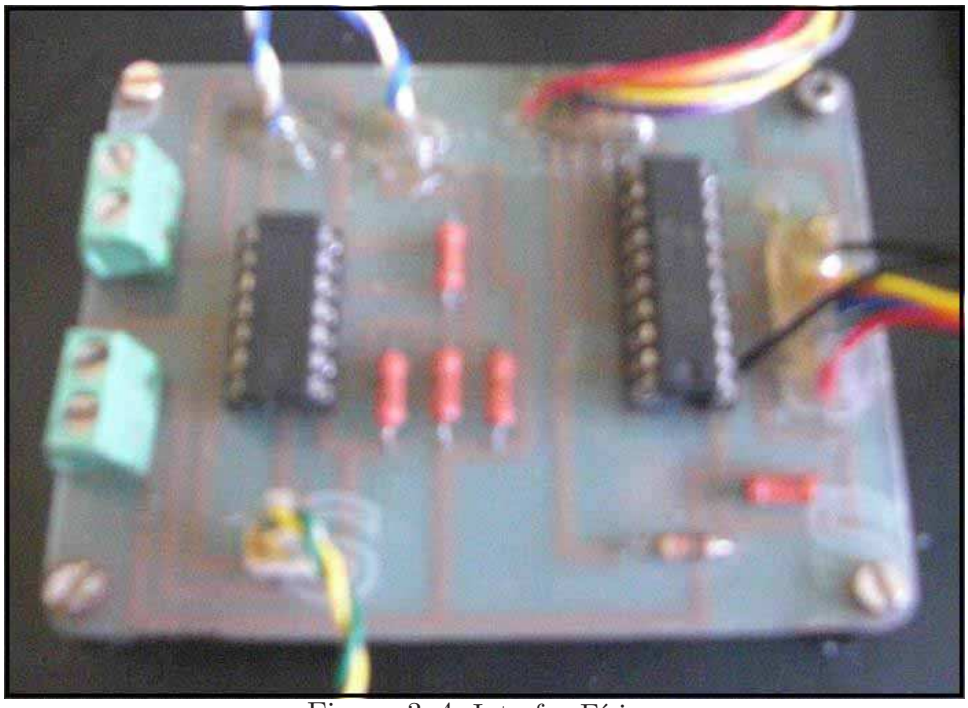

Figura 3–4: Interfaz Física.

3.3. Software del Sistema

3.3.1. Compilación del Kernel de Linux

En la versión del kernel de Linux instalada previamente en la tarjeta MBTI2440 no estaban disponibles las Entradas Salidas de Propósito General (GPIO) necesarias para la comunicación con la Multi I/O Expansion Board. Esta funcionalidad solo es accesible luego de una recompilación del kernel. Para lograr nuestro objetivo nos apoyamos en documentos disponibles en la web principalmente en el tutorial (emeb, 2010). Básicamente es necesario el código fuente del kernel que se pretende compilar, en este caso contamos con la distribución 2.6.29 suministrada por el fabricante de la tarjeta, quien también suministra el compilador cruzado arm-linux-gcc-4.3.2 con el cual se obtiene el nuevo kernel y además se compilan las demás aplicaciones (algoritmo de control). En este proyecto se utiliza además el software ncurses que permite configurar el kernel mediante una interfaz gráfica. Es necesario aclarar que el trabajo de cross-compilación del núcleo se realiza sobre SO GNU/Linux en este caso utilizamos la versión 6 de Debian. Para realizar la instalación del compilador cruzado se debe añadir su dirección al PATH del sistema, lo cual se realiza con el siguiente comando:

export PATH=/home/..(dirección del compilador)/:$PATH

El PATH es una variable de entorno que contiene una serie de rutas de directorios separados por dos puntos (Vialfa, 2008). Esta variable contiene las direcciones donde se realiza la búsqueda cada vez que se pretende ejecutar una aplicación. Para la instalación del programa ncurses bastó con escribir el comando **apt-get install ncurses 5.6.** Una vez instalados el compilador y el ncurses en la PC nos situamos en la carpeta donde se encuentra el kernel, desde un Terminal GNOME del SO introducimos el comando **make menuconfig**. Este comando ejecuta el programa ncurses el cual carga un fichero de configuración genérico o definido por el usuario en el cual se agregan o desactivan funcionalidades del SO quedando de esta manera el kernel acorde a nuestras necesidades.

El Terminal GNOME emula un terminal de aplicaciones sobre el cual se puede acceder al shell de Unix el cual interpreta y ejecuta comandos introducidos por el usuario. También se pueden ejecutar aplicaciones desde este terminal (Cutler, 2009).

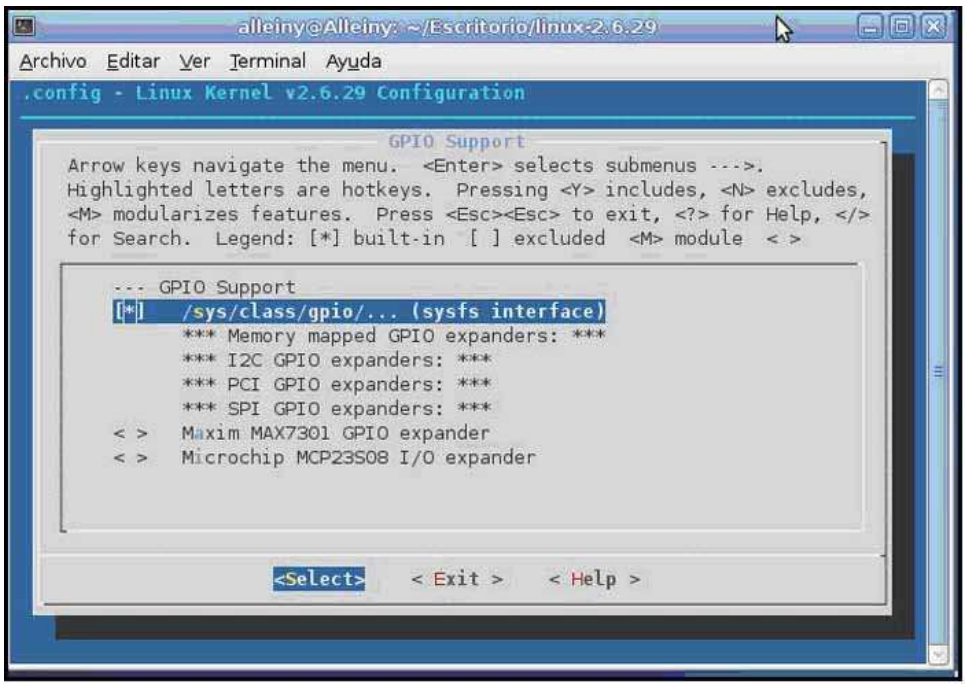

Figura 3–5: Interfaz del Ncurses para añadir GPIO.

En este trabajo solo incluimos los controladores de GPIO Fig. 3–5 para ser posteriormente compilados tras ejecutar el comando **make zImage**. Una vez terminado el proceso de compilación ya tenemos lista la imagen del kernel que debemos cargar en la tarjeta

MBTI2440. Para cargar la imagen del kernel recompilado en nuestra tarjeta es necesario realizar los siguientes pasos. Primeramente habilitamos la memoria NOR Flash de la tarjeta MBTI2440, en esta memoria se ejecuta un programa que nos permite almacenar en memoria RAM la imagen del kernel. A partir de este momento estaremos trabajando sobre SO Windows en este caso XP. Necesitamos también los programas DNW y supervivi, el DNW es el software que se comunica con la tarjeta MBTI2440 y el supervivi es el responsable de cargar el kernel para el arranque del sistema. Con el ejecutable WRITE.BIN podemos escribir en la memoria NAND Flash el software supervivi y la imagen del kernel. Para realizar estas operaciones necesitamos establecer 2 conexiones con la tarjeta MBTI2440 Fig. 3–6, una conexión serie (master) para ejecutar instrucciones y una conexión usb (slave) para la copia de los ficheros en la memoria NAND Flash.

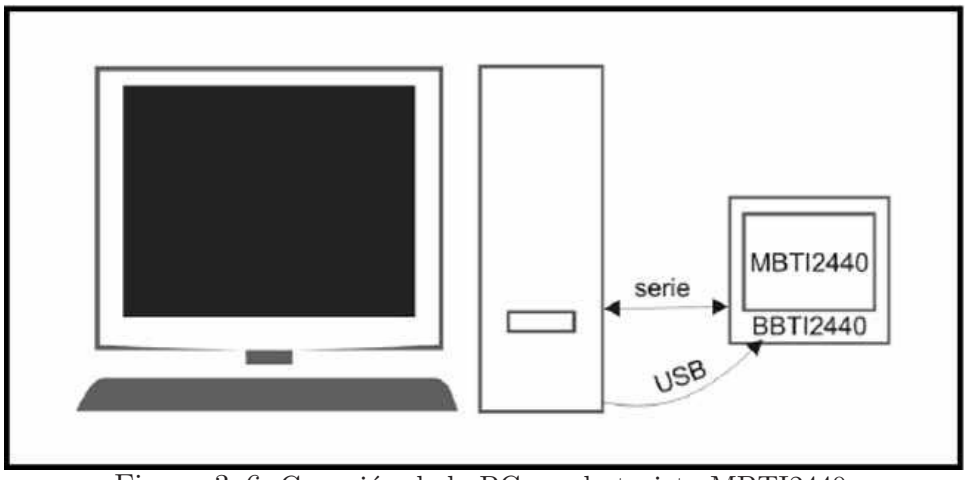

Figura 3–6: Conexión de la PC con la tarjeta MBTI2440.

El primer paso para dejar funcional nuestro kernel después de compilado es borrar la memoria NAND Flash de nuestra tarjeta, luego se le graba el programa supervivi en la dirección **30100000** y posteriormente se le graba la imagen del kernel en la dirección **30130000**.

Una vez realizadas estas operaciones ya podemos arrancar nuestra tarjeta con los driver de GPIO incorporados a nuestro SO no sin antes habilitar nuevamente la memoria NAND Flash, lo cual se realiza volviendo a colocar en su lugar el jumper resaltado en la Fig. 3–7.

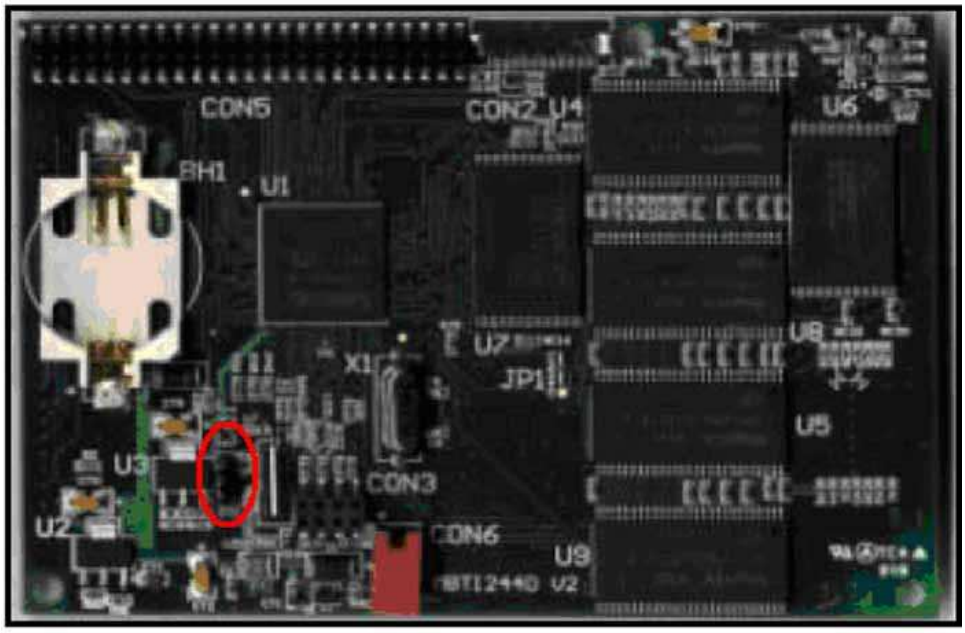

Figura 3–7: Selección de la Memoria a Utilizar.

Una vez realizados estos procedimientos el SO funciona correctamente en nuestra tarjeta y ya tiene operables las entradas-salidas digitales necesarias para implementar posteriormente la comunicación y el control previstos en este proyecto.

3.3.2. Software Controlador de Ejes

El software de control de ejes implementado consta básicamente de 5 ficheros fuentes los cuales están presentes en el Anexo B.

- io-board.c, en este fichero se implementan todas las funciones de entrada-salida. En primer lugar se exportan todos los pines necesarios para comunicación con la Multi I/O Expansion Board. Seguidamente se implementan las funciones que escriben y leen directamente en los pines y finalmente se implementan las funciones capaces de leer las entradas analógicas y escribir en las salidas analógicas. Estas funciones fueron programadas basándonos en la hoja de datos de la Multi I/O Expansion Board (JK , n.d.).
- timer-int.c, en este fichero se implementa la función que genera la interrupción de tiempo (Corbet, 2005), necesaria para producir el período de muestreo del sistema.
- control.c, en este fichero se implementa el algoritmo de control. La Fig. 3–8 muestra el diagrama de bloque del algoritmo implementado (Rubio, 2006). Vale destacar que como

la plataforma es de dos grados de libertad se programó el mismo algoritmo para cada eje (ladeo y cabeceo).

- win-stile-form.cxx, en este fichero se implementó la interfaz gráfica para la versión de prueba.
- main.c, este es el fichero principal. Aquí se inicializa la Multi I/O Expansion Board, se arranca la interrupción de tiempo y se inicializan las variables del controlador.

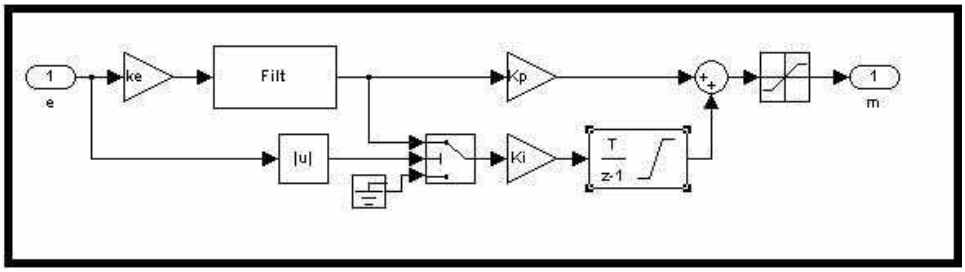

Figura 3–8: Esquema del Algoritmo de Control.

3.4. Resultados Obtenidos

En este proyecto hemos obtenido un controlador empotrado con elevada capacidad de cómputo para el control de plataformas neumáticas de dos grados de libertad. Debido a problemas ajenos a nuestra voluntad el sistema obtenido no ha podido ser probado aun en la plataforma y por tal razón nos dimos a la tarea de crear una aplicación que muestre las características y prestaciones del mismo. La Fig. 3–9 muestra el tiempo de ejecución del lazo de control. Como se puede observar este tiempo es de 5 ms pudiéndose reducir a 1 ms según referencias del fabricante.

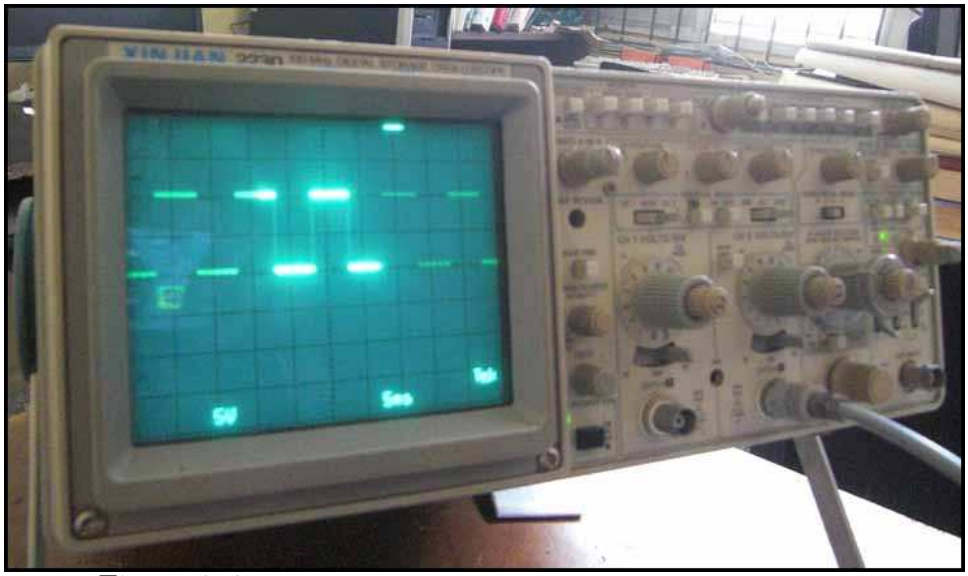

Figura 3–9: Tiempo de Ejecución del Algoritmo de Control.

La Fig. 3–10 muestra el funcionamiento de las entradas-salidas analógicas, pudiéndose observar que la medición del multímetro se corresponde con el valor mostrado por el slider referente al canal cero.

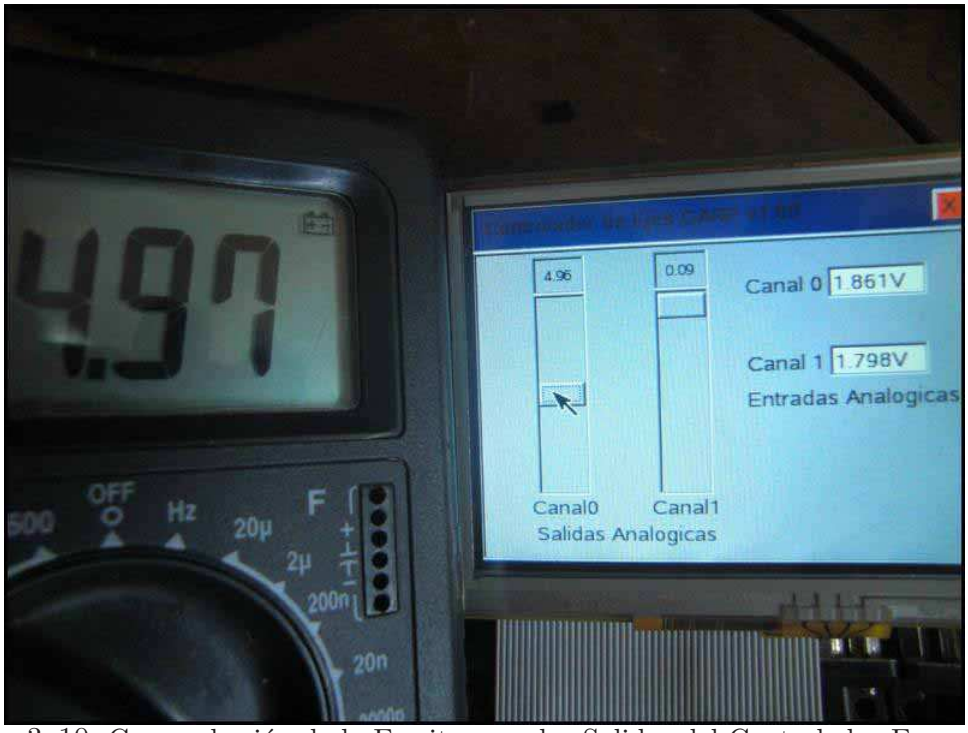

Figura 3–10: Comprobación de la Escritura en las Salidas del Controlador Empotrado.

3.5. Análisis Económico

El controlador empotrado obtenido presenta un bajo coste en relación con su capacidad de cómputo, la cual es importante tomando en consideración que nos encontramos en presencia del control de plataformas neumáticas, las cuales exigen de algoritmos de control potentes. Comparado con controladores implementados anteriormente en nuestro grupo de investigación como el de Rubio (Rubio, 2006) que disponía de una PC de escritorio y una tarjeta de adquisición de datos con un coste aproximado de 1500 USD es un controlador barato. En Machado (Machado, 2007) se presenta un controlador empotrado con un coste aproximado de 200 USD lo cual es un bajo precio pero el controlador basado en Flashlite 186 presenta un desempeño menor que nuestra tarjeta. En el caso de Sosa (Sosa, 2007) presentó un hardware para el control de plataformas con un coste de 100 USD aproximadamente pero su hardware presentaba mucho menos potencialidades que el nuestro pues él utilizó dsPIC para su controlador. Nuestro hardware básico, la tarjeta MBTI2440 con su touch-panel y kit de programación posee un coste que se estima en 250 USD. La tarjeta Multi I/O Expansion Board se estima en 50 USD aunque puede ser reemplazada por un hardware más óptimo con el objetivo de reducir gastos innecesarios

pues esta tarjeta presenta potencialidades que no son necesarias. Los gastos que pudieran calcularse para los circuitos integrados 74LS244, LM324 y el circuito impreso se pudieran despreciar incluyéndolos si se quiere dentro del coste de la tarjeta Multi I/O expansion Board, por lo que nuestro hardware tiene un costo aproximado de 300 USD. Este coste resulta pequeño si observamos sus prestaciones y comparamos con los controladores implementados anteriormente. Ver cuadro 3–2.

Cuadro 3–2: **Comparación entre Controladores Empotrados según su coste.**

	MBTI2440	PC Escritorio	Flashlite	dsPIC 30f4013
Frecuencia de Trabajo	400 MHz	2 GHz	33 MHz	120 MHz
Coste	300 USD	1500 USD	200 USD	100 USD

3.6. Consideraciones Finales del Capítulo

En este capítulo hemos expuesto las características del hardware empotrado obtenido basado en la tarjeta MBTI2440, los cuales no son todo lo que desearíamos pero aun así son buenos resultados. A través del software de prueba y el osciloscopio XINJIAN 2230 expusimos el buen funcionamiento de nuestro controlador empotrado que muestra buena precisión y desempeño.

CONCLUSIONES

Con el presente trabajo se logró desarrollar un sistema empotrado para el control de la plataforma desarrollada por SIMPRO la cual posee dos grados de libertad con actuadores electro-neumáticos. Dicha plataforma cumple con los requerimientos básicos para el entrenamiento del personal en la conducción de vehículos y su efectividad ha sido probada en las aplicaciones reales desarrolladas por SIMPRO. La tecnología empotrada utilizada permite la ejecución del algoritmo de control desarrollado por (Rubio, 2006) el cual es complejo y no convencional, pero que había sido probado con buenos resultados en varias configuraciones de hardware pero no definitivos para el usuario final (SIMPRO). La selección de MBTI2440 como regulador empotrado, a partir de las posibilidades empresariales de SIMPRO, garantiza las exigencias prácticas de ejecución de los algoritmos de control para ambas articulaciones con una frecuencia de muestreo de 5 ms disminuible a 1 ms .

El camino escogido de usar controladores empotrados no está limitado solamente para plataformas de dos grados de libertad, tiene capacidad de ser utilizada en plataformas con seis grados de libertad, controlando seis ejes y resolviendo la cinemática inversa.

RECOMENDACIONES

Como recomendaciones del trabajo podemos plantear que el controlador empotrado utilizando MBTI2440, ejecutando el algoritmo de control de Rubio (Rubio, 2006) sea introducido en los simuladores que desarrolla SIMPRO y que sea evaluada la conveniencia de incorporar esta tecnología de control en las plataformas en funcionamiento en este momento. Recomendamos además continuar trabajando en la disminución del período de muestreo así como el diseño a la medida de una tarjeta de entrada-salida que incluya lectura de encoders y adicione varias entradas-salidas analógicas. Como última recomendación está la de crear una aplicación que incluya además del control de ejes, la identificación de la plataforma y el ajuste del controlador automáticamente.

REFERENCIAS BIBLIOGRÁFICAS

Brun (2000). Control of an electropneumatic actuator, comparison between some linear and nonlinear control laws.. *Journal of Systems and Control Engineering*.

CAPEL, Dpto Técnico (2008). Microcontroladores arm: Estructuras y herramientas de desarrollo. pp. 66–70.

Corbet, Jonathan (2005). *Linux Device Drivers*. 3ra ed.

Cutler, Paul (2009). *GNOME Terminal Manual*.

emeb (2010). Friendly arm mini 2440. Página Web.

Florez, J. (2007). Análisis de lenguajes y herramientas de especificación, implementación y validación de sistemas en robótica móvil.

ICI (n.d.). *MBTI2440 Computadora en una Tarjeta*. 1.31 ed.

Igor, G. (2006). Pneumatic avtuating systems for automatic equipment, structure and desing.

JK (n.d.). *Multi I/O Expansion Board*.

Jorba, J. (10). El kernel.

Lucero, A. (2006). Linux en sistemas empotrados.

Machado, Alleiny (2007). Controlador empotrado para plataforma neunática de simulador de conducción. Master's thesis. UNIVERSIDAD CENTRAL "MARTA ABREU"DE LAS VILLAS.

Munoz, J. (2002). Sistemas informáticos en tiempo real para sistemas de control. *Anales de Mecánica y Electricidad*.

Pandian (2002). Pressure observer-controller design for pneumatic cylinder actuators. *EEE/ASME Transactions on Mechatronics*.

Rubio, E. (2006). Modelado, identificaciÓn y control de actuadores lineales electroneumÁticos. aplicaciÓn en plataforma de dos grados de libertad.. *CEA-IFAC*.

Sakamoto (2006). Virtual and remote laboratory for robot manipulator control study. *International Journal of Engineering Education 22*.

Sam (2004). *S3C2440A 32-BIT CMOS Microcontroller*. 1ra ed.

Schulte, H. (2003). Fuzzy state feedback gain scheduling control of servo-pneumatic actuators. *Journal on Control Engineering Practice*.

Silva, L. (2005). Control visual de robots paralelos. anÁlisis, desarrollo y aplicaciÓn a la plataforma robotenis.. Master's thesis. Universisdad Politecnica de Madrid.

Sosa, Y. (2007). CONTROLADOR COMPACTO MULTI EJE DE DESPLAZAMIENTO LINEAL CON DSPIC30F4013. Tesis de grado. Universidad Central "Martha Abreu"de las Villas.

Stewart (1965-66). Platform with six degrees of freedom.. *UK Institution of Mechanical Engineers Proceedings*.

Uchikado, Yamada. T. (2000). Adaptive pole-placement control with multi-rate type neural network for pneumatic servo system. *IEEE Proceedings of the International Conference on Control Applications.*

Vialfa, C. (2008). Bash, la variable de entrono path. Pàgina Web.

Apéndice A
ANEXO 1

MBTI2440 computadora en una tarjeta. Manual de Usuario:

Versión 1.31

Instituto Central de Investigación Digital (ICID) Calla 202 #1704, Siboney, Playa CP 11600, La Habana, CUBA

Teléfonos 271-5345 / 271-5054

1. INTRODUCCIÓN

La MBTI2440 es una computadora en una tarjeta basada en el procesador S3C2440A de SAMSUNG.

Fig. 1.1: Vista superior de la MBTI2440

Figura A–1:

MBTI2440 fue diseñada con el objetivo de obtener una tarjeta de procesamiento de bajo costo y pequeño tamaño, bajo consumo de potencia, con gran variedad de dispositivos de interfaz y elevada conectividad y una capacidad de procesamiento adecuada a los requerimientos de los procesadores empleados en equipos médicos de complejidad media desarrollados por el ICID como son monitores de parámetros vitales, desfibriladores y electrocardiógrafos entre otros, sobre la cual se pudieran montar los sistemas operativos LINUX y Windows CE. La tarjeta MBTI2440 no posee los conectores estándar de interconexión con los periféricos y dispositivos de interfaz. Todas las líneas de entrada salida de la MBTI2440 se están conectadas a tres conectores de 50x2 contactos. Para emplear la tarjeta la de procesamiento los usuarios deben desarrollar una tarjeta base que se conectará a MBTI2440, y en la cual se colocarán los conectores requeridos en las posiciones apropiadas para cada aplicación.

2. PARTES QUE SE SUMINISTRAN

Cuando se suministra la tarjeta MBTI2440 con las partes requeridas para trabajarla como sistema de desarrollo, se puede suministrar además la tarjeta MBTI2440, las siguientes partes:

- TARJETA BASE DE DESARROLLO
- TARJETA MBTI2440
- MEMORIA FLASH SD 2GB
- MODULO DE CRISTAL LIQUIDO TFT EN COLORES DE 320X240 PUNTOS CON CABLE DE CONEXIÓN
- CABLE DE COMUNICACIÓN USB A-B
- CABLE DE COMUNICACIÓN SERIE (adaptador macho/hembra)
- CABLES PARA PROGRAMACIÓN JTAG
- CD CON INFORMACIÓN TÉCNICA Y SOFTWARE
- ADAPTADOR DE CORRIENTE ALTERNA A DIRECTA (5V/1.0A)

Para facilitar el desarrollo de las tarjetas base que se requieren en cada aplicación, el presente Manual incluye en el capítulo 8 los esquemas eléctricos y la lista de partes de la tarjeta base de desarrollo que se suministra junto con la tarjeta de procesamiento.

3. CARACTERÍSTICAS TÉCNICAS

CARACTERÍSTICAS ELÉCTRICAS Y MECÁNICAS

- Alimentación: 5 V 400 mA (máximo).
- Dimensiones: 104 x73 mm.

CARACTERÍSTICAS DEL PROCESADOR

- - Velocidad del procesador: hasta 400MHz.
- - Velocidad del Bus: Hasta 133MHz.
- - Memoria
    1. SDRAM 32, 64, 128, 256 Mbyte.
    2. NOR FLASH: 1Mbyte.
    3. NAND FLASH: 64 Mbyte.
- - Interfaz para LCD (STN o TFT) resolución hasta 1024 x 768 con 24 bpp.
- - Interfaz para pantalla táctil (touch screen) de 4 hilos.
- - Interfaz Ethernet 10 Mbit/s (incluye los LEDs de link y LAN).
- - Entrada y salida de audio estéreo.
- - 3 puertos serie (niveles TTL).
- - 1 puerto USB device.
- - 4 puertos USB host.
- - Reloj de tiempo real con respaldo de batería.
- - Interfaz para tarjeta de memoria SD/MMC.
- - 1 LED de estado.
- - 1 LED de indicación de alimentación.
- - Interfaz JTAG.
- - Interfaz para cámara.

- - 2 puertos SPI.
- - 1 puerto IIC.
- - Hasta 94 líneas de entrada/salida digital (11 sin comprometer otras funciones, 83 con otra función primaria).
- - 4 entradas analógicas a un conversor A/D de 10 bits.
- - Entradas y salidas de señal digital LVTTL (3.3 V).

CARACTERÍSTICAS DEL SOFTWARE La tarjeta MBTI2440 es suministrada con Linux 2.6.13 embebido para procesadores ARM. Para emplear la tarjeta con Windows CE contactar con el fabricante. La tarjeta se suministra con los siguientes módulos:

sbc_vivi (no se suministra con el código fuente) Boot Start up del sistema Xmodem Soporta el protocolo de transmisión Xmodem USB Le adiciona funciones de descarga USB (downloading) al vivi, suportando descargas/actualizaciones de imágenes vía USB. Kernel Parameter Proporciona el establecimiento de parámetros del kernel Partition Proporciona establecimiento de particiones

Kernel Versión Linux kernel 2.6.13
File system ROM/CRAM/EXT2/FAT32/NFS/YAFFS

Drivers
Interrupt & Timer Interrupciones y Timer del Sistema
Serial device Tres puertos serie
10M Ethernet CS8900
10M/100M EtherneT DM9000
RTC Reloj de Tiempo Real
USB Host USB mouse, USB keyboard, U-disk
LEDS Manejo de LEDs
Buttons Manejo de botones de usuario
Language Soporte Multi-lenguaje

LCD Manejo de pantallas de LCD

Frame Buffer Frame Buffer

Touch panel

SD/MMC card

UDA1341

IDE No se suministra el código fuente

Embedded GUI

Qt/Embedded

Protocolo de redes y aplicaciones

TCP/IP TCP/IP protocol

File transfer (FTP client/server)

Remote login

Apéndice B
ANEXO 2

Código Fuente.

[io-board.c]

```
////////////////////////////////////////////////////////////////////////////
//Codigo fuente de interfaz entre MBTI2440 y Multi IO Board de Flash lite.     //
//Autores: MSc. Alleiny Machado Sosa (GARP 2011)                              //
//       : Enrique Cepero Morfa                                               //
////////////////////////////////////////////////////////////////////////////


#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int i;
FILE *fp;
FILE *fp0;
FILE *fp2;
FILE *fp3;
FILE *fp4;
FILE *fp5;
////////////////////////////////////////////////////////////////////////////
// Prototipo de funciones                                                   //
```

```
////////////////////////////////////////////////////////////////////////////

void io_board_init(void);
void set_out_en(int valor);
void set_dato(int valor);
void set_reset(int valor);
void set_clk(int valor);
void set_cs(int valor);
int get_dato(void);
void analog_out(int canal, int dato);
unsigned int analog_in(int canal);
void mydelay(void);
////////////////////////////////////////////////////////////////////////////
// Implementacion de funciones                                            //
////////////////////////////////////////////////////////////////////////////


void io_board_init(void)
{

 char set_value[4];

////////////////////////////////////////////////////////////////////////////
 if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 195, Bit 3
{
 exit(1);
}
 rewind(fp); //inicio del fichero
```

```
 fwrite("195", sizeof(char), 3, fp);
 fclose(fp);

 if ((fp = fopen("/sys/class/gpio/gpio195/direction", "rb+")) == NULL) //direccion c
  {
    exit(1);
  }
 rewind(fp);
 fwrite("out", sizeof(char), 3, fp);
 fclose(fp); //pin 195 exportado y como salida (Output_Enable)

 if ((fp5 = fopen("/sys/class/gpio/gpio195/value", "rb+")) == NULL) //pin abierto y
   {
     exit(1);
   }

 ////////////////////////////////////////////////////////////////////////////////
 if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 160, Bit C
{
 exit(1);
}
 rewind(fp); //inicio del fichero
 fwrite("160", sizeof(char), 3, fp);
 fclose(fp);

 if ((fp = fopen("/sys/class/gpio/gpio160/direction", "rb+")) == NULL) //direccion c
  {
    exit(1);
```

```
 }
rewind(fp);
fwrite("out", sizeof(char), 3, fp);
fclose(fp); //pin 160 exportado y como salida (dato_out)

if ((fp0 = fopen("/sys/class/gpio/gpio160/value", "rb+")) == NULL) //pin abierto y
  {
    exit(1);
  }



////////////////////////////////////////////////////////////////////////////////

if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 165, Bit 5
{
exit(1);
}
rewind(fp); //inicio del fichero
fwrite("165", sizeof(char), 3, fp);
fclose(fp);

if ((fp = fopen("/sys/class/gpio/gpio165/direction", "rb+")) == NULL) //direccion d
 {
   exit(1);
 }
rewind(fp);
fwrite("in", sizeof(char), 2, fp);
fclose(fp); //pin 165 exportado y como entrada (dato_in) // cada vez que se quiera
```

```
 ////////////////////////////////////////////////////////////////////////////

 if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 166, Bit 6
{
 exit(1);
}
 rewind(fp); //inicio del fichero
 fwrite("166", sizeof(char), 3, fp);
 fclose(fp);

 if ((fp = fopen("/sys/class/gpio/gpio166/direction", "rb+")) == NULL) //direccion d
  {
    exit(1);
  }
 rewind(fp);
 fwrite("out", sizeof(char), 3, fp);
 fclose(fp); //pin 166 exportado y como Salida (clk)

  if ((fp2 = fopen("/sys/class/gpio/gpio166/value", "rb+")) == NULL) //pin abierto y
   {
     exit(1);
   }

 ////////////////////////////////////////////////////////////////////////////

 if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 167, Bit 7
{
```

```
 exit(1);
}
 rewind(fp); //inicio del fichero
 fwrite("167", sizeof(char), 3, fp);
 fclose(fp);

 if ((fp = fopen("/sys/class/gpio/gpio167/direction", "rb+")) == NULL) //direccion c
  {

    exit(1);
  }
 rewind(fp);
 fwrite(&"out", sizeof(char), 3, fp);
 fclose(fp); //pin 167 exportado y como Salida (Reset)

   if ((fp3 = fopen("/sys/class/gpio/gpio167/value", "rb+")) == NULL) //pin abierto
   {
     exit(1);
   }
 ////////////////////////////////////////////////////////////////////////////////

 if ((fp = fopen("/sys/class/gpio/export", "ab")) == NULL)  //Exporta Pin 162, Bit 2
{
 exit(1);
}
 rewind(fp); //inicio del fichero
 fwrite("162", sizeof(char), 3, fp);
 fclose(fp);
```

```
 if ((fp = fopen("/sys/class/gpio/gpio162/direction", "rb+")) == NULL) //direccion c
  {
    exit(1);
  }
 rewind(fp);
 fwrite("out", sizeof(char), 3, fp);
 fclose(fp); //pin 162 exportado y como Salida (CS)

  if ((fp4 = fopen("/sys/class/gpio/gpio162/value", "rb+")) == NULL) //pin abierto y
   {
     exit(1);
   }

////////////////////////////////////////////////////////////////////////////////
// Inicializacion de las señales de la io_board
////////////////////////////////////////////////////////////////////////////////
set_out_en(1);  //set Output Enable en 1

set_reset(1);  //reset a 0

set_cs(1);     //chip select a 1

set_clk(0);    // reloj a 0

set_dato(0);   // dato a 0

analog_out(0,0);
```

```
analog_out(1,0);
}



void set_out_en(int valor)
{

 if(valor)
   {
     rewind(fp5);
     fwrite("1", sizeof(char), 1, fp5);
     rewind(fp5);
     fwrite("1", sizeof(char), 1, fp5);
   }
  else
   {
     rewind(fp5);
     fwrite("0", sizeof(char), 1, fp5);
     rewind(fp5);
     fwrite("0", sizeof(char), 1, fp5);
   }
 }

void set_dato(int valor)
{

 if(valor)
```

```
   {
     rewind(fp0);
     fwrite("1", sizeof(char), 1, fp0);
     rewind(fp0);
     fwrite("1", sizeof(char), 1, fp0);
   }
  else
   {
     rewind(fp0);
     fwrite("0", sizeof(char), 1, fp0);
     rewind(fp0);
     fwrite("0", sizeof(char), 1, fp0);
   }

 }

void set_clk(int valor)
{
 if(valor)
   {
     rewind(fp2);
     fwrite("1", sizeof(char), 1, fp2);
     rewind(fp2);
     fwrite("1", sizeof(char), 1, fp2);
   }
  else
   {
     rewind(fp2);
```

```
     fwrite("0", sizeof(char), 1, fp2);
     rewind(fp2);
     fwrite("0", sizeof(char), 1, fp2);
   }

}

void set_reset(int valor)
{
  if(valor)
   {
     rewind(fp3);
     fwrite("1", sizeof(char), 1, fp3);
     rewind(fp3);
     fwrite("1", sizeof(char), 1, fp3);
   }
  else
   {
     rewind(fp3);
     fwrite("0", sizeof(char), 1, fp3);
     rewind(fp3);
     fwrite("0", sizeof(char), 1, fp3);
   }

}

void set_cs(int valor)
{
```

```
  if(valor)
   {
     rewind(fp4); FILE *fp1;
     fwrite("1", sizeof(char), 1, fp4);
     rewind(fp4);
     fwrite("1", sizeof(char), 1, fp4);
   }
  else
   {
     rewind(fp4);
     fwrite("0", sizeof(char), 1, fp4);
     rewind(fp4);
     fwrite("0", sizeof(char), 1, fp4);
   }

}




int get_dato(void)
{
 char in_value[4];
 int dato;
 FILE *fp1;
  if ((fp1 = fopen("/sys/class/gpio/gpio165/value", "rb+")) == NULL)
    {
      exit(1);
    }
```

```
  rewind(fp1);
  fread(in_value, sizeof(char), 1, fp1);
  fclose(fp1);

  dato = in_value[0] - 48;
  return dato;
}

void analog_out(int canal, int dato)
{
  int i;
  int buf[12];

  for(i = 0; i < 12; i++)
   {
     buf[i] = (dato >> i) & 0x0001;
   }

  // seleccion del DAC
   set_reset(0); // pulso de reset
   set_reset(1);

   set_cs(0); // cs a 0

   set_out_en(0);

   set_dato(1); // start bit board
   set_clk(1);  // pulso de reloj 1
```

```
set_clk(0);

switch(canal)
{
  case 0:{   //canal 1
           set_dato(0); // direcion DAC1 bit 2
           set_clk(1);  // pulso de reloj 2
           set_clk(0);

           set_dato(1); // direcion DAC1 bit 1
           set_clk(1);  // pulso de reloj 3
           set_clk(0);

           set_dato(1); // direcion DAC1 bit 0
           set_clk(1);  // pulso de reloj 4
           set_clk(0);
           break;
         }
   case 1:{   //canal 2
           set_dato(1); // direcion DAC2 bit 2
           set_clk(1);  // pulso de reloj 2
           set_clk(0);

           set_dato(0); // direcion DAC2 bit 1
           set_clk(1);  // pulso de reloj 3
           set_clk(0);

           set_dato(0); // direcion DAC2 bit 0
```

```
            set_clk(1);  // pulso de reloj 4
            set_clk(0);
            break;
          }
   }

   // DAC selecionado
   for(i = 0; i < 4; i++)
     {
      set_dato(0);  // operacion normal
      set_clk(1);  // idle
      set_clk(0);
     }

   for(i = 0; i < 12; i++)
    {
      set_dato(buf[11 - i]);  // MSB primero
      set_clk(1);  // pulsos de reloj para escritura en DAC
      set_clk(0);
    }

   set_out_en(1);
   set_cs(1); //deselecciono io_board
 }

unsigned int analog_in(int canal) //modo diferencial
{
  int dato[12];
```

```
int i,out,dat;
 // seleccion del ADC
 set_reset(0); // pulso de reset
 set_reset(1);

 set_cs(0); // cs a 0

 set_out_en(0);  // tomo control de la linea de dato

 set_dato(1); // start bit board
 set_clk(1);  // pulso de reloj 1
 set_clk(0);

 set_dato(0); // direcion ADC bit 2
 set_clk(1);  // pulso de reloj 2
 set_clk(0);

 set_dato(0); // direcion ADC bit 1
 set_clk(1);  // pulso de reloj 3
 set_clk(0);

 set_dato(0); // direcion ADC bit 0
 set_clk(1);  // pulso de reloj 4
 set_clk(0);

 // ADC activo

 set_dato(1); // start bit ADC
```

```
set_clk(1);  // pulso de reloj 1 ADC
set_clk(0);

set_dato(0); // A2 bit
set_clk(1);  // pulso de reloj 2 ADC
set_clk(0);

if(canal)//si canal = 1
 {
   set_dato(1); // A1 para canal 1
   set_clk(1);  // pulso de reloj 3 ADC
   set_clk(0);

   set_dato(0); // A0 para canal 1
   set_clk(1);  // pulso de reloj 4 ADC
   set_clk(0);
 }
else
 {
   set_dato(0); // A1 para canal 0
   set_clk(1);  // pulso de reloj 3 ADC
   set_clk(0);

   set_dato(1); // A0 para canal 0
   set_clk(1);  // pulso de reloj 4 ADC
   set_clk(0);
 }
```

```
    set_dato(0); // Modo 12 bit
    set_clk(1);  // pulso de reloj 5 ADC
    set_clk(0);

    set_dato(0); // Modo diferencial
    set_clk(1);  // pulso de reloj 6 ADC
    set_clk(0);

    set_dato(1); // no power down
    set_clk(1);  // pulso de reloj 7 ADC
    set_clk(0);

    set_dato(1); // no power down
    set_clk(1);  // pulso de reloj 8 ADC
    set_clk(0);

    set_out_en(1); // libero linea de datos para que el dac tome el control de la li

    // Dato dentro se pasa a lectura del dato
    for(i = 0; i < 12; i++)
    {
      set_clk(1);  // pulsos de reloj  para lectura de ADC  (busy a 0)
      set_clk(0);  //dato listo
      dato[11-i]= get_dato(); //leo dato
    }
   for(i = 0; i < 4; i++)
     {
      set_clk(1);  // idle
```

```
      set_clk(0);
     }
   set_cs(1); //deselecciono io_board

   out = dato[0];

   for(i = 1; i < 12; i++)  // recomposicion del dato
     {
       out = out + (dato[i] << i);
     }


  return out;

}
```

timer-int.c]

```
  ////////////////////////////////////////////////////////////////////////
//Código de inicialización del timer para interrupción de tiempo          //
//Autores: MSc. Alleiny Machado Sosa (GARP 2011)                          //
//       : Enrique Cepero Morfa                                           //
////////////////////////////////////////////////////////////////////////

#include <time.h>
#include <stdio.h>
```

```
#include <signal.h>
#include <pthread.h>
#include <unistd.h>
#include "control.c"


int estado=0;

int timer_interrupt_init(void); // inicializa timer
void timer_isr(union sigval val); // rutina de control




void timer_isr(union sigval val)
    {
      control(); // llama a algoritmo de control en control.c
    }


int timer_interrupt_init(void)
{
  int Ret;

  pthread_attr_t attr;
  pthread_attr_init( &attr );

  struct sched_param parm;
  parm.sched_priority = 255;
```

```
  pthread_attr_setschedparam(&attr, &parm);

  struct sigevent sig;
  sig.sigev_notify = SIGEV_THREAD;
  sig.sigev_notify_function = timer_isr;
  sig.sigev_value.sival_int = 1;
  sig.sigev_notify_attributes = &attr;

  timer_t timerid;   //crea timer nuevo.
  Ret = timer_create(CLOCK_REALTIME, &sig, &timerid);
  if (Ret == 0)
   {
     struct itimerspec in, out;
     in.it_value.tv_sec = 0;
     in.it_value.tv_nsec = 1;
     in.it_interval.tv_sec = 0;
     in.it_interval.tv_nsec = 500000; // 5 ms

     Ret = timer_settime(timerid, 0, &in, &out);
     if(Ret == 1)
      {
        timer_delete(timerid);
      }
   }

   return Ret;
}
```

control.c]

```
////////////////////////////////////////////////////////////////////////////
//Algoritmo de control para cilindros neumáticos de plataforma de conducción //
//Autores: MSc. Alleiny Machado Sosa (GARP 2011)                           //
//                                                                         //
////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <stdlib.h>

#define ke 1.0
#define kp 1.0

#define kilad  2.07446968976845  //coeficientes del controlador (roll)
#define a1lad -1.55464680122260
#define a2lad  0.60423166913791
#define b0lad  0.79794500153352
#define b1lad -1.57714023761965
#define b2lad  0.78379323120734

#define kicab  2.07446968976845 //coeficientes del controlador (pitch)
#define a1cab -1.55464680122260
#define a2cab  0.60423166913791
#define b0cab  0.79794500153352
#define b1cab -1.57714023761965
```

```
#define b2cab  0.78379323120734

#define mmx 5.0   //en volt
#define mmn -5.0  // en volt
#define emx 4.0   //en mm

char dato[4];
char cab_ctrl,lad_ctrl;

unsigned int  xposd,yposd,lad_init;     //valores deseados (entero 12 bits)
unsigned int  xposr,yposr,cab_init;     //valores real     (entero 12 bits)



  en volt
#define emx 4.0   //en mm

char dato[4];
char cab_ctrl,lad_ctrl;

unsigned int  xposd,yposd,lad_init;     //valores deseados (entero 12 bits)
unsigned int  xposr,yposr,cab_init;     //valores real     (entero 12 bits)

float mando;
float fexk,fexk1,fexk2;        //entrada al filtro en x
float feyk,feyk1,feyk2;        //entrada al filtro en y
float fsxk,fsxk1,fsxk2;        //salida del filtro  en x
float fsyk,fsyk1,fsyk2;        //salida del filtro  en y
```

```
float pfexk,pfexk1,pfexk2;     //entrada al prefiltro en x
float pfeyk,pfeyk1,pfeyk2;     //entrada al prefiltro en y
float pfsxk,pfsxk1,pfsxk2;     //salida del prefiltro  en x
float pfsyk,pfsyk1,pfsyk2;     //salida del prefiltro  en y

float iexk,iexk1;        //entrada del integrdor en x
float ieyk,ieyk1;        //entrada del integrdor en y
float isxk,isxk1;        //salida del integrdor en x
float isyk,isyk1;        //salida del integrdor en y

float  exk;              //errores xy
float  eyk;

unsigned int mxk;         //mandos xy
unsigned int myk;



void control(void)
{
  double tmpx, tmpy;
 //********************************** Salida mando anterior *************//

   analog_out(0,mxk);    // Ladeo
   analog_out(1,myk);    // Cabeceo

//********************************** Entrada de posición real  **********//

   xposr = analog_in(0);  //posición real ladeo
```

```
   yposr = analog_in(1);   //posición real cabeceo

//****************************************** Calculo control Ladeo *******//
if(lad_ctrl)
   {
    xposd = (dato[0] << 8) + dato[1]; //posición deseada ladeo
    exk = xposd - xposr;    //error de posición ladeo
    fexk2 = fexk1;          //filtro x
    fexk1 = fexk;
    fexk =  (ke * exk);
    fsxk2 = fsxk1;
    fsxk1 = fsxk;
    fsxk = b0lad * fexk + b1lad * fexk1 + b2lad * fexk2 - a1lad
    * fsxk1 - a2lad * fsxk2;


    iexk1 = iexk;    //integrador x
    isxk1 = isxk;

    tmpx = fsxk * kilad;

    iexk = (abs(exk) > emx)? tmpx:0;             //anti windup
    isxk = (0x51 * iexk + 0x51 * iexk1 + isxk1);
    isxk = (isxk > mmx)? mmx:isxk;
    isxk = (isxk < mmn)? mmn:isxk;

    mando = fsxk + isxk;
```

```
    mando = (mando > mmx)? mmx:mando;
    mando = (mando < mmn)? mmn:mando;

    mxk = 0x7FF + (mando / 0x10);  //mando x
 }
  else
    {
     if(xposr > lad_init)
      {
lad_ctrl = 1;
dato[0] = lad_init >> 8;     //vaor deseado de ladeo inicial
dato[1] = lad_init & 0xFF;
      }
   }

//************ Calculo control cabeceo ************************//
if(cab_ctrl)
   {
    yposd = (dato[2] << 8) + dato[3]; //posición deseada cabeceo

    eyk = yposd - yposr;   //error posición cabeceo
    feyk2 = feyk1;      //filtro y
    feyk1 = feyk;
    feyk  = (ke * eyk);
    fsyk2 = fsyk1;
    fsyk1 = fsyk;
    fsyk  = b0cab*feyk + b1cab*feyk1 + b2cab*feyk2 - a1cab*fsyk1
     - a2cab*fsyk2;
```

```
    ieyk1 = ieyk;    //integrador y
    isyk1 = isyk;

    tmpy = fsyk * kicab;

    ieyk = (abs(eyk) > emx)? tmpy:0;           //anti windup
    isyk = 0x51 * ieyk + 0x51 * ieyk1 + isyk1;
    isyk = (isyk > mmx)? mmx:isyk;
    isyk = (isyk < mmn)? mmn:isyk;

    mando = fsyk + isyk;
    mando = (mando > mmx)? mmx:mando;
    mando = (mando < mmn)? mmn:mando;

    myk =  0x7FF + (mando / 0x10); //mando y
  }
else
 {
   if(yposr > cab_init)
    {
cab_ctrl = 1;
dato[2] = lad_init >> 8;
dato[3] = lad_init & 0xFF;
    }
 }
}
```

win-stile-form.cxx]

```
////////////////////////////////////////////////////////////////////////////
// Forms al estilo de windows                                             //
//Autores: MSc. Alleiny Machado Sosa (GARP 2011)                          //
//       : Enrique Cepero Morfa                                           //
////////////////////////////////////////////////////////////////////////////

#include "FL/forms.H"

static int border = 1;

char texta[5] = {'1','0','.','0','V'};
char textb[5] = {'1','0','.','0','V'};
//char val[2];

#include "pixmaps/srs.xbm"

FL_FORM *form;

Fl_Widget *exit_button, *texto;
Fl_Widget *roll_slider,*pitch_slider, *inputa, *inputb;

void fl_win_stile_form(int x, int y, int w, int h,const char *l)
{
```

```
    //esta función debe ser llamada en: create_the_forms()

    Fl_Widget *up_box, *down_box, *title_form;

    form = fl_bgn_form(FL_NO_BOX, 320, 240); //forma

    up_box = fl_add_box(FL_DOWN_BOX, x, y, w, 30, ""); //box superior
    fl_set_object_color(up_box, FL_DARK_BLUE, FL_COL1);     //azul

    title_form = fl_add_text(FL_NORMAL_TEXT, x+5, y+5, w-35, 15, l); // texto de la
    fl_set_object_color(title_form, FL_YELLOW, FL_COL1); //color blanco

    exit_button = fl_add_button(FL_NORMAL_BUTTON, w-25, y+5, 20, 20, "X"); //botón c
    fl_set_object_color(exit_button, FL_RED, FL_COL1); // color rojo


    down_box = fl_add_box(FL_DOWN_BOX, x, y+30, w, h-30, ""); //box de abajo;
    fl_set_object_color(down_box, FL_COL1, FL_COL1); // forma gris

 }



void create_main_forms (void)
{
  Fl_Widget *analog_out_text, *analog_input_text;

  fl_win_stile_form(0,0,320,240,"Controlador de Ejes GARP v1.00");
```

```
  analog_out_text = fl_add_text(FL_NORMAL_TEXT, 35, 213, 100, 15, "Salidas Analógica
  analog_input_text = fl_add_text(FL_NORMAL_TEXT,175 , 125, 100, 15, "Entradas Analo

  roll_slider = fl_add_valslider(FL_VERT_SLIDER, 40, 35, 35, 160, "Canal0");
  fl_set_slider_bounds(roll_slider,0,10);
  fl_set_slider_value(roll_slider,0);

  pitch_slider = fl_add_valslider(FL_VERT_SLIDER, 120, 35, 35, 160, "Canal1");
  fl_set_slider_bounds(pitch_slider,0,10);
  fl_set_slider_value(pitch_slider,0);

  texto = fl_add_box(FL_NO_BOX, 220, 150,60,20,"");
  inputa = fl_add_input (FL_NORMAL_INPUT, 230,50,60,20, "Canal 0");

  inputb = fl_add_input (FL_NORMAL_INPUT, 230,100,60,20, "Canal 1");

  fl_end_form ();
  fl_show_form (form, FL_PLACE_MOUSE, border, "");
  //while (fl_do_forms () != exit_button);
}

void on_roll(Fl_Widget *,void *)
{
  double value;
  int dato;
  value = fl_get_slider_value(roll_slider);
  dato = value * 409.5;
```

```
  analog_out(0,dato);
}

void on_pitch(Fl_Widget *,void *)
{
  double value;
  int dato;
  value = fl_get_slider_value(pitch_slider);
  dato = value * 409.5;
  analog_out(1,dato);
}
```

main.cxx]

```
////////////////////////////////////////////////////////////////
//Programa Principal para MBTI2440 como controladora de ejes   //
//Autores: MSc. Alleiny Machado Sosa (GARP 2011)               //
//       : Enrique Cepero Morfa                                //
////////////////////////////////////////////////////////////////

#include "FL/forms.H"
#include "io_board.c"
#include "timer_int.c"
#include "win_stile_form.cxx"
#include <time.h>
#include <unistd.h>
```

```
int main (int argc, char *argv[])
{
  dato[0] = 0;   //inicialización de variables del controlador
  dato[1] = 0;
  dato[2] = 0;
  dato[3] = 0;

  xposd = yposd = xposr = yposr = 0;
  fexk = fexk1 = fexk2 = 0;
  feyk = feyk1 = feyk2 = 0;
  fsxk = fsxk1 = fsxk2 = 0;
  fsyk = fsyk1 = fsyk2 = 0;

  iexk = iexk1 = 0;
  ieyk = ieyk1 = 0;
  isxk = isxk1 = 0;
  isyk = isyk1 = 0;

  exk = eyk = 0;
  mxk = myk = 0;

  lad_ctrl = cab_ctrl = 0;   // parqueo o control
  lad_init = cab_init = 0x7FF; //posición de parqueo

   io_board_init();  // inicialización de tarjeta IO
```

```
   analog_out(1,0x8CC);  // 5.5 Volt... abre ligeramente las válvulas para parqueo
   analog_out(2,0x8CC);  // 5.5 Volt... abre ligeramente las válvulas para parqueo


   timer_interrupt_init(); //inicialización y arranque de
                           //timer para interrupción de tiempo
  fl_initialize(&argc, argv, "FormDemo", 0, 0); //aplicación gráfica
  create_main_forms ();
  roll_slider -> callback(on_roll);
  pitch_slider -> callback(on_pitch);
  while (fl_do_forms () != exit_button)
   {
    sleep(1000);
   }
}
```