ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"CONTROL DE MOTORES USANDO PLATAFORMA NIOS II"

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN

ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentada por:

JOSÉ ANTONIO ZAMBRANO MOLINA.

ANGEL ALFREDO LARA CANDELARIO.

GUAYAQUIL – ECUADOR

AÑO: 2014

AGRADECIMIENTO

Agradezco a Dios por sobre todas las cosas por guiarme en medio de la oscuridad cuando lo único que se puede hacer es abrir los ojos buscando la luz y aun si la luz no aparece avanzar confiando en el corazón.

Agradezco a mi familia, mis profesores y compañeros por el apoyo brindado durante estos años llenos de sacrificios y dificultades en los cuales sus consejos y ayuda me hicieron saber que no estaba solo.

José Antonio Zambrano Molina.

DEDICATORIA

A la memoria de mi padre que creyó en mi hasta el último día de su vida y quien siempre tuvo fe en que pasaría por este punto como un escalón de una meta mucho mayor y cuyos consejos divagan en mi mente todos los días y desde siempre.

José Antonio Zambrano Molina.

AGRADECIMIENTO

Agradeciéndole en primer lugar a Dios por encaminarme correctamente en mi vida y darme las fuerzas necesarias para seguir adelante.

Agradezco también a mi familia por cuidarme, protegerme y guiarme a mis Profesores, Amigos y mi enamorada por el apoyo dado durante mis estudios y por depositar su confianza en mí.

Angel Alfredo Lara Candelario.

DEDICATORIA

Dedicado a mi Padre quien siempre confió en mí, que me brindó su apoyo confianza y siempre velo por mí para poder seguir adelante.

Angel Alfredo Lara Candelario.

TRIBUNAL DE SUSTENTACIÓN

.....

M.Sc. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

.....

M.Sc. Damián Larco

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesina nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL).

.....

José Antonio Zambrano Molina.

.....

Angel Alfredo Lara Candelario.

RESUMEN

Este proyecto presenta una solución alternativa al uso de drivers industriales, Por medio de una FPGA Cyclone II de Altera.

El presente trabajo se lo ha estructurado en cinco capítulos.

El capítulo 1 se presenta los objetivos del proyecto, cual es la ventaja con referencia a otros sistemas de control de motores en la industria y la metodología a seguir en el proyecto.

El capítulo 2 incluye el fundamento teórico sobre el que se basa esta tesina. Se revisan conceptos fundamentales de procesadores embebidos configurables y FPGA. Incluye información sobre los diferentes tipos de motor que serán usados en el presente proyecto así como también el método que se usara para controlarlo y de todo lo necesario para tener claras las limitaciones y alcances de la aplicación.

En el capítulo 3 se describe cada una de las partes que componen el proyecto, separados por hardware embebido, hardware físico y software a implementar. El hardware embebido está comprendido principalmente por el FPGA, el hardware físico son los circuitos que deben adicionarse la tarjeta para que esta conste de la parte de control (hardware embebido) y la interface de potencia para operar los motores. Se presenta además un detalle de la secuencia de configuración del módulo. En esta parte podremos

apreciar las capacidades del sistema de control así como la interface que se usara con el operador.

El software del proyecto se lo desarrolla en lenguaje C para el núcleo de procesador NIOS II que esta embebido en el FPGA configurado anteriormente.

El capítulo 4 presenta un exhaustivo análisis del funcionamiento del sistema en condiciones reales donde se verifica su utilidad y limitaciones

Finalmente en el capítulo 5 se presentan las pruebas realizadas al circuito, imágenes del funcionamiento y datos adquiridos en las pruebas.

ÍNDICE GENERAL

ABREVIATURAS

FPGA	Field Programmable Gate Array
DE2	Development and Education Board 2
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
ADC	Analog-to-Digital Converter
TTL	Transistor-Transistor Logic
HDL	Language Description Hardware
SD	Secure Data
E/S	Entrada y Salida
UART	Universal Asynchronous Receiver-Transmitter
DSP	Digital Signal Processing
USB	Universal Serial Bus
JTAG	Joint Test Action Group
VHDL	Very High Speed Integrated Circuit hardware (Description Language)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN.

El avance tecnológico cada día se supera más y nos damos cuenta que desde el inicio de la electrónica con los dispositivos básicos luego prosiguiendo con los circuitos integrados, continuando con memorias de programables llegando a la era de los Microcontroladores y al final tarjetas específicas como lo son CPLD’s y FPGA’s, la cual nos permite diseñar el comportamiento del hardware según nuestros requerimientos, para después poder implementar los programas que realizamos para una aplicación específica.

Adicional a ello el control de motores es también una de las líneas tecnológicas que ha avanzado dentro de la industria y demás aplicaciones llegando a conectarse de una manera óptima y eficaz a través de estos circuitos electrónicos como lo son la FPGA.

Por medio del presente trabajo escrito, se ha mostrado todo el análisis necesario para diseñar una computadora la cual nos permite utilizar los recursos para controlar los diversos motores en una maqueta que se basa en desembarcar harina desde un contenedor en un buque hacia una tolva y después de ello depositarlo en diversos autos los cuales van a distribuir el producto. Este proyecto se basa en el análisis del control de motores, con la posibilidad de incrementar el número de motores por expansión modular.

Nuestro análisis va a comenzar con el requerimiento del hardware para el control de dichos motores, después de ello se realiza el software en un código en C, el cual es implementado en la FPGA para poder efectuar nuestro propósito.

En conclusión nuestro proyecto va a mostrar por medio de una pantalla LCD el tipo de control a realizar de manera automática o manera manual usando diferentes tipos de sensores ubicados en la maqueta, mediante un teclado para manual. Todo esto brevemente programado para poder extender o suprimir la cantidad de motores y sensores de la maqueta.

CAPÍTULO 1

1. GENERALIDADES

Este documento presenta cada objetivo alcanzado en el proceso de este estudio, los problemas en los cuales es presentado como una solución alternativa y bajo que metodología se ha desarrollado esta solución.

1.1. OBJETIVOS

1.1.1. Objetivos Generales

Aplicar los conocimientos y habilidades adquiridas durante nuestro desarrollo en el pregrado usando las experiencias dadas en materias previas como laboratorio de digitales, electrónica de potencia, enfatizando la aplicación del presente proyecto a la carrera de automatización industrial a la cual pertenecemos.

Adicionando a nuestra formación lo aprendido durante el seminario de Microcontroladores Embebidos Configurables. Lo

cual ha facilitado y simplificado en gran parte el diseño del presente controlador de motores.

La idea general del proyecto muestra hacia donde apuntaron nuestras aspiraciones a lo largo de la documentación de la presente tesis. Cuyas partes hemos ido concluyendo a lo largo de las pruebas e ideas adaptadas a nuestra aplicación.

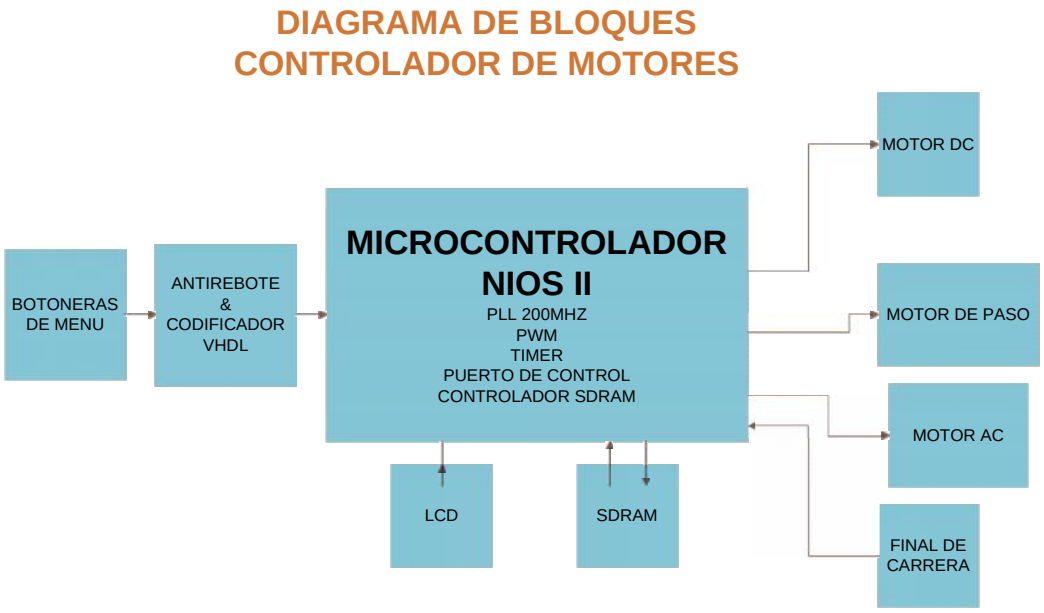


Figura 1.1 Diagrama de bloques controlador de motores

1.1.2. Objetivos Específicos

- Investigar los diferentes tipos de control de motores de corriente continua y alterna usados principalmente en la industria y en algunos sistemas robotizados los cuales

requieren de alta precisión como motores de paso, motores de corriente alterna, motores DC de magnetos permanentes entre otros.

- Diseñar de manera eficiente un Microcontrolador embebido en un FPGA permitiendo usar solo los recursos necesarios para el control de motores específicos.
- Medir la cantidad de recursos usados en el proyecto y proponer las posibles mejoras al proyecto en un futuro.
- Tener una perspectiva más afín con las posibilidades futuras de los sistemas embebidos en los diseños electrónicos inteligentes.

1.2. IDENTIFICACIÓN DEL PROBLEMA

Hoy en día, la electrónica ha avanzado con mucha rapidez desde el descubrimiento del transistor que nos llevó a la era de las computadoras, hoy por hoy hay sistemas electrónicos para todo tipo de aplicaciones, la electrónica se ha convertido en la herramienta del día a día.

Desde un sistema neuronal que podemos ver en el industria hasta el celular que llevamos con nosotros para poder comunicarnos, estos sistemas electrónicos han dejado de cumplir simplemente la función

para la que fueron creados, ahora casi todo es multifunción y con accesos no solo manual, sino también automático, online por medio de internet, conexión bluetooth, etc.

La intención de los ingenieros es hacer estos sistemas más eficientes, sistemas que puedan manejar una gran cantidad de datos y realizar cálculos complejos a alta velocidad con una interfaz fluida entre el Usuario y la Maquina, que sea portable para facilitar su traslado y que genere el menor consumo de energía para que sea amigable con el medio ambiente.

Todas estas características pueden ser solventadas por los FPGA gracias a su gran capacidad para albergar gran cantidad de dispositivos embebidos como Microcontroladores y dispositivos creados en VHDL, lo cual nos permite reducir mucho un circuito electrónico además de que posee una gran velocidad de procesamiento y bajo consumo de energía.

1.3. METODOLOGÍA

Este proyecto parte de un proceso industrial en el cual se controlaran tres tipos de motores eléctricos diferentes, la idea es que los módulos para agregar motores sean montables por programación en el dispositivo y así cada vez que se agregue un nuevo motor al sistema,

el controlador tenga la capacidad de con un pequeño cambio adaptar al nuevo motor.

La selección de motores para este propósito será motor de paso, motor AC y motor DC.

Primero se procede a convenir los métodos de control de tales motores, como se trabajara con respecto a las cargas y las acciones que realizara en el proceso a simular. Se emplearan diagramas de bloques para explicar mejor la estructura del sistema de control de cada motor.

Segundo se realiza la estructura global sobre la cual se definirá como trabajara el Microcontrolador con referente a los motores y se seleccionara el medio de interface con el usuario.

Tercero se crea la maquina embebida en Qsys y luego mediante diagrama de bloques procederemos a realizar las conexiones con dispositivos en VHDL en Quartus II de la versión 12.1, una vez habiendo creado el hardware, se procederá a programar el software en Nios II.

Cuarto se construyelos circuitos de potencia, los cuales estarán conectados directamente a los motores y también al circuito de control (FPGA) mediante una interface aislada.

En la última fase del proyecto se realizan las pruebas del controlador de motores ya en la maqueta, junto con la presentación del proceso industrial elegido.

CAPÍTULO 2

2. MARCO TEÓRICO

En este capítulo damos a conocer el funcionamiento general de los componentes fundamentales de este proyecto debido que en ellos se basa el diseño del proyecto, los principales componente aquí presentados son el FPGA, motor de paso, motor AC, el motor DC, tarjetas optoacopladoras, tarjetas de fuerzas y Sensores Magnéticos.

2.1. FPGA

2.1.1. Definición y características

El FPGA es un dispositivo semiconductor que posee bloques de lógica digital cuya interconexión puede ser configurable mediante un leguaje de descripción de hardware HDL mediante el cual, el usuario puede crear dentro del dispositivo nuevos componentes con funcionalidades específicas. Los alcances de este leguaje

van desde crear simples puertas lógicas, sistemas combinatoriales, secuenciales, hasta sistemas tan complejos como computadores que manejan sistemas operativos lite.

La ventaja de los FPGA es que son reprogramables y esto da una gran flexibilidad en el flujo del diseño además que da la posibilidad de mejora sin tener que modificar el hardware físico. Permitiendo aminorar los costos de desarrollo y tiempo.

Las características de los FPGA's

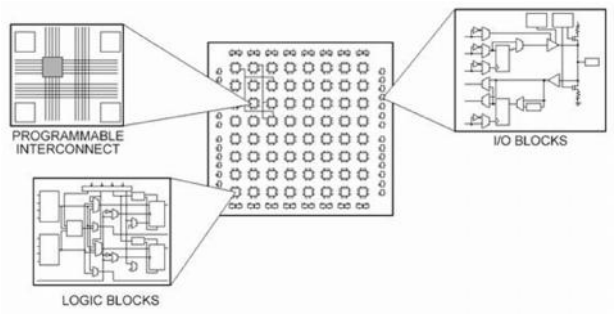


Figura 2.1 Partes de un FPGA

Lo que destaca en estos dispositivos es su jerarquía de bloques lógicos que pueden ser interconectados según la necesidad del diseñador, estos dispositivos pueden ser programados en la maquina aun después de un proceso de manufacturación, así el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido la de combinar los bloques lógicos de los FPGA con microcontroladores y periféricos para realizar sistemas complejos de un chip programable (Minicomputadoras

con utilidad especifica). Esto es la tendencia actual hacia los sistemas reconfigurables o actualizables.

La configuración de estos dispositivos se logra mediante lenguaje de descripción de hardware (HDL), los lenguajes más utilizados para estos dispositivos son:

- VHDL
- Verilog
- Abel

Por último, National Instruments propone una interfaz de programación grafica para aminorar el tiempo y la complejidad en el desarrollo de sistemas en FPGA.

2.1.2. Familia Cyclone II

Figura 2.2 Cyclone II

Se utiliza Cyclone II FPGA para mejorar la relación precio-rendimiento y para el procesamiento de señales digitales (DSP) dentro de las aplicaciones.

Se puede implementar los sistemas de alto rendimiento en DSP a bajo costo con las siguientes características del ciclón II dentro de su soporte de diseño:

- Hasta 300 multiplicadores embebidos en el chip
- Hasta 1152000 bits de memoria integrados en el chip
- Hasta 68416 elementos logicos
- Procesador de señales digitales para núcleos (IP).
- Interfaz de diseño para procesadores de señales digitales con plataforma de entorno para los programas Simulink y Matlab
- Kit de desarrollo de DSP, el ciclón II Edición

2.1.2.1. Tarjeta de desarrollo DE2

Figura 2.3 DE2

La DE2 es la tarjeta que se ha definido seleccionar para el proyecto del controlador de motores, se la ha elegidodebido a que la tarjeta DE2 tiene características que permiten al usuario implementar una amplia gama de circuitos diseñados para diversos proyectos multimedia. [7]

Entre las características de alimentación esta tarjeta puede alimentarse mediante una línea de 9 Voltios DCque puede adquirirse desde una fuente externa o una batería.

La tarjeta posee un FPGA Cyclone II EP2C35F672C6 CON EPCS16 16-Mbit con un dispositivo de configuración serial EPCS64 64-Mbit, una memoria SDRAM de 8 MB, una 512KB SRAM, además de 4MB de memoria flash y un ranura para una memoria SD externa.

Entre los dispositivos de maniobra que posee la tarjeta están

- Línea entrada/ salida de micrófono de 24 bits
- Salida de video VGA con convertidor de analógico a digital de 10 bits.
- Entrada de video en NTSC/PAL/ Multi-formato
- Puerto infrarrojo
- Puerto PS/2 para mouse o Keyboard
- Puerto Ethernet 10/100

- Puertos USB tipo Ay tipo B 2.0
- 2 puertos de 40 pines de expansión
- 8 displays de 7-segmentos
- Display LCD 16x2
- 18 Switches de dos posiciones
- 4 pulsadores
- 18 Leds rojos
- 9 leds verdes

De los cuales usaremos algunos en el proyecto y pruebas por separado del control de los motores.

2.2. Motor

2.2.1. Motor DC

Es una máquina que con diferencia sobre las demás (maquina síncrona y maquina asíncrona), permite variar la velocidad mediante métodos de variación de parámetros realmente sencillos. Este es el motivo por el cual, pese a sus inconvenientes como el costo, mantenimiento y la necesidad de una corriente continua, siguen utilizándose en accionamientos eléctricos de velocidad variable y control de torque.

Los motores de corriente continua de aplicación industriales son de excitación separada.

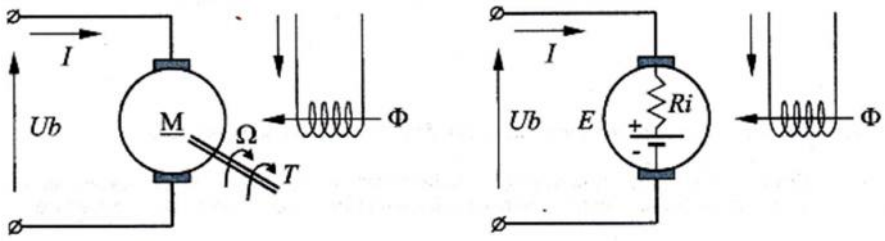


Figura 2.4 Esquema de un motor DC de excitación separada

Los parámetros del motor DC son:

- ϕ: flujo por polo
- I: corriente de inducido
- Ub: tensión en colector de motor
- R_i: resistencia del inducido
- w: velocidad de giro
- T: par desarrollado por motor
- k_a: constante de maquina

Las ecuaciones que gobiernan el comportamiento del motor DC

$$E = K_a \varphi w$$

$$T = k_a \varphi I$$

$$Ub = E + R_i I$$

En muchos casos de los motores DC se encuentran con cajas reductoras que no es más que un sistema de engranajes que varían el eje disminuyendo la cantidad de vueltas que se proporcionan pero aumentando el torque.

2.2.1.1. Aplicación en la industria del Motor DC

Los motores de corriente continua son usados para aplicaciones de control y automatización de procesos gracias a su fácil control de posición, par y velocidad.

Actualmente se utilizan en aplicaciones que poseen frenado regenerativo mediante chopper

Podemos encontrar motores de corriente continua, en sistemas troceados de voltaje para control de corriente en montacargas.

2.2.2. Motor de Paso

Los motores paso a paso es un dispositivo electromecánico que puede controlarse mediante pulsos eléctricos los cuales son convertidos en movimientos angulares, es decir con cada serie de pulsos puede girar su rotor una cantidad determinadade grados o también llamado paso, este motor trabaja como un convertidos digital/analógico y puede ser controlado por impulsos lógicos.

Su principal ventaja es su alta precisión y su repetitividad con respecto al posicionamiento.

Estos motores tienen dos bobinas A y B, es decir cuatro terminales. Su secuencia de trabajo seria la siguiente. [8]

PASO	A1	A2	B1	B2
1	+VCC	+VCC	GND	GND
2	GND	+VCC	+VCC	GND
3	GND	GND	+VCC	+VCC
4	+VCC	GND	GND	+VCC

Tabla 1 Secuencia de polaridades por paso

Para realizar esta secuencia, normalmente se utilizan dos integrados, los cuales son el Microcontrolador el cual realiza la función de generador de pulsos y el L298N el cual es un driver de potencia.

2.2.2.1. Aplicación en la industria del Motor de Paso

Entre las aplicaciones de los motores de paso está la de posicionador, se lo puede encontrar dentro de los discos duros de computadoras, impresoras y lectoras de CD, se lo usa en la robótica además de aplicaciones de rodillo, por ejemplo en la impresión de etiquetas, fecha o codificado en rollos de material de envase de polietileno el cual es utilizado en la industria alimenticia.

2.2.3. Motor de corriente alterna

La mayoría de las maquinas utilizadas en la industria son movidas por motores asíncronos alimentados por corriente alterna, el cual funciona debido a la interacción de campos magnéticos producidos por corrientes eléctricas producidas por el fenómeno de inducción electromagnética. Estos motores pueden ser monofásicos, bifásicos o trifásicos, cuando este tipo de motores entra en operación se desarrolla un campo giratorio, pero antes de que inicie la rotación, el estator genera un campo magnético pulsante.

En nuestro proyecto utilizamos un motor monofásico el cual para producir un campo giratorio y un par de arranque, debe tener un devanado auxiliar desfasado 90 grados con respecto al devanado principal. [9]

2.2.3.1. Características de los motores AC

- Excelente par de arranque
- Facilidad de conexión
- Bajo costos
- De poco Mantenimiento
- Elevado desempeño

- Robustez

2.2.3.2. Aplicación en la industria de los Motores AC.

De los motores de corriente alterna podemos nosotros hallarlo dentro de las industrias muy comúnmente en:

- Bombas centrífugas.
- Bombas de desplazamiento alternativo.
- Bandas transportadoras.
- Trituradoras.
- Ventiladores.
- Máquinas herramientas.
- Embotelladoras.
- Compresoras de arranque sin carga.
- Hiladoras.
- Voladoras garrotillo
- Desmenuzadoras de alimentos

2.3. LCD

Para el presente proyecto se ha seleccionado una LCD como interfaz de usuario, en la cual el operario puede visualizar las velocidades de los motores.

Las LCD son placas delgadas y planas, constituida por un número de pixeles de color o monocromático los cuales son sometidos a una fuente de luz.

El display a utilizar fue un monocromático de 2 líneas por 16 caracteres por línea en el cual la primera línea indica que dato se está visualizando, la segunda línea presenta al usuario el valor del parámetro medido.

CAPÍTULO 3

3. Diseño e implementación

En el siguiente capítulo se va a redactar en detalle el diseño e implementación y la descripción del proyecto y la maqueta para dar a conocer el cómo funcionan los diferentes elementos y componentes mencionados el capítulo 2. Además de ello se hace una descripción de hardware y software utilizados, así como su interacción, en la maqueta.

Comenzando con el hardware este se encuentra compuesto por la configuración de la FPGA (Controlador), motores (Carga) como motores DC, motor de paso y motores AC. De la tarjeta DE2 se va usar los elementos, LEDs rojos, Pines de Expansión (Expansion Header JP) y el display LCD de 2x16 los cuales se los va usar como herramienta de interacción y verificación para el funcionamiento del sistema hacía el operador, para ello primero debemos de configurar el hardware del FPGA

y usando el software Quartus II, con la herramienta SOPC Builder donde escogemos los diferentes drives para interactuar con los hardware externos.

Para el software del sistema lo va a constituir la programación que se realice sobre el núcleo de procesador NIOS II, el cual se va a desarrollar en lenguaje C usando el entorno de programación Eclipse. Con esta programación vamos a manejar los Controladores que configuramos en el hardware de la FPGA. En la figura 6 se muestra el diagrama general utilizado en el proyecto.

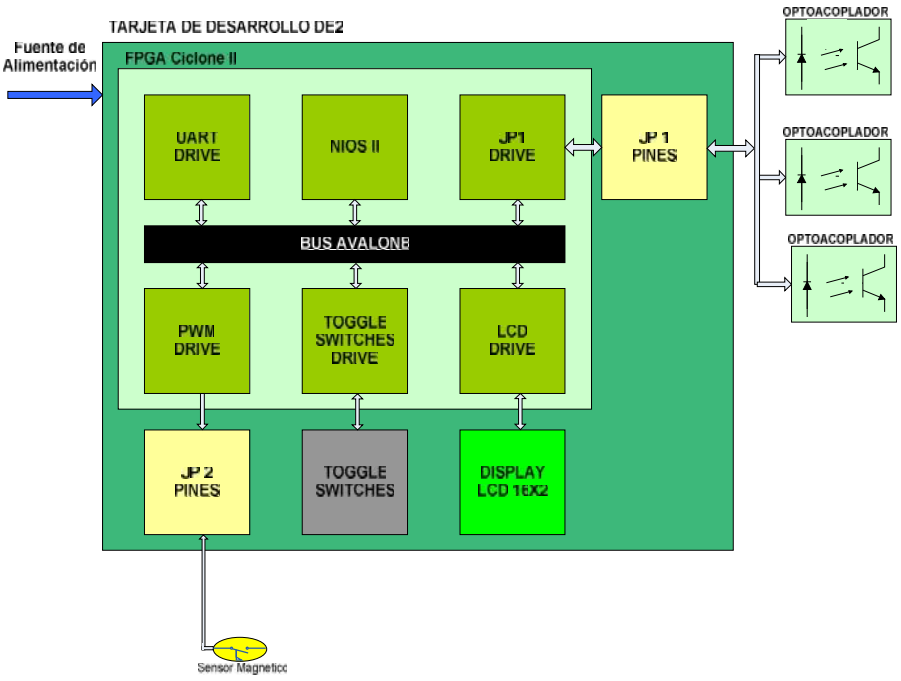


Figura 3.1 Diagrama de Bloques General de Sistema

3.1. Hardware

3.1.1. Componentes en la tarjeta de desarrollo

Dentro de un correcto funcionamiento del sistema para la tarjeta de desarrollo DE2 es obligatorio utilizar los elementos necesarios tanto para la comunicación como para la programación y adicionalmente demás elementos los cuales van a interactuar con el hardware externo (figura 7).

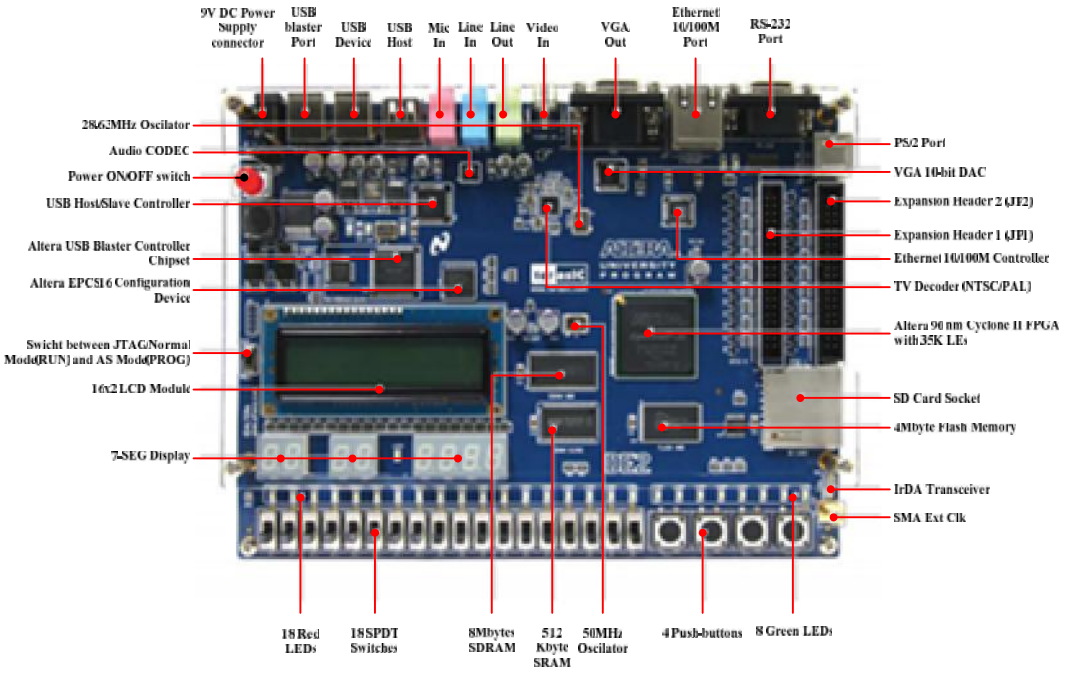


Figura 3.2 Componentes utilizados de la tarjeta DE2

- Interruptores de palanca (Toggle Switches): Interruptor con lógica 0 en posición abajo (cerca del borde de la tarjeta DE2) y con lógica 1 en la posición de arriba (alejados del borde de la tarjeta DE2).
- LEDs Rojos: Componente optoelectrónico pasivo el cual emite luz y dependiente de su estado indica una acción pasada o actual de un proceso previamente desarrollado en el software del NIOS II.
- Pines de Expansión (ExpansionHeader JP):72 Cyclone II I/O pins, así como 8 líneas de poder y tierra, se presenta como 40 pines de salida en forma de un conector de expansión, la cabecera de 40 pines está diseñado para aceptar un cable plano de 40 pines estándar que se utiliza para los discos duros IDE que además cuenta con Diodos y resistencias de Protección, estos pines interactúan con los tarjetas externas como tarjetas con optoacopladores, o tarjeta con troceadores, además interactúa con dispositivos de accionamiento directo Sensores Magnéticos, etc.
- LCD de 2x16: Donde se Presentan los mensajes indicando el estado del proceso con las respectivas acciones.

3.1.2. FPGA

La configuración del hardware dentro de la FPGA se lo realiza en base del programa Quartus II, el cual tiene como herramienta SOPC Builder con la que se configura cada uno de los módulos.

Para nuestro proyecto van a ser necesarios los siguientes módulos:

- CPU: Procesador NIOS II.
- SDRAM: Controlador RAM de 8Mbyte para el procesamiento del código.
- SRAM: Controlador SRAM de 512 kbyte para el procesamiento del código.
- Expansion_JP: Controlador par uso de los puertos JP1 y JP2.
- Char_LCD_16x2: Controlador para display de 16x2.
- DIP_Switches: Controlador para interface de los Interruptores de palanca (Toggle Switches).
- Green_LEDS: Controlador para interface de los LEDS.

Cuando se utiliza configuraciones por defecto como por ejemplo DE2_Media_Computer.sop, se le debe agregar módulos como el PWM el cual fue debidamente explicado en el seminario de "Microprocesadores Embebidos Configurables" del cual previamente se debe revisar la configuración de los pines asignados, para que no exista algún conflicto con los módulos generales.

Con el programa Quartus II dentro de la herramienta SOPCBuilder, se utiliza una interfaz amigable y confiable para realizar esta tarea con programas previos ya existentes en el cual nosotros podemos realizar cambios, para ingresar a esta herramienta se escoge el icono SOPCBuilder, en donde se abre una nueva ventana con el listado de módulos disponibles en el lado izquierdo y en el lado derecho los módulos que vamos a configurar, para escoger uno de estos modelos disponibles están sólo necesario dar doble clic sobre el nombre para poderlos ubicar en el lado derecho, en el caso de no existir algún módulo como es el caso del PWM se lo agrega.

3.1.3. Motores DC

Los Motores DC a utilizar tienen un accionamiento por frecuencia PWM los cuales por medio una tarjeta que va a optoacoplar el sistema de la tarjeta de desarrollo DE2 con la tarjeta de fuerza se van accionar los motores dependiente de la frecuencia debidamente programada para ello dentro de la tarjeta de fuerza vamos a utilizar el integrado L298N con el cual cuando nuestro programa emita la onda respectiva esta tarjeta nos genera el movimiento requerido para ello nos basamos en la tabla 2 para generar el movimiento.

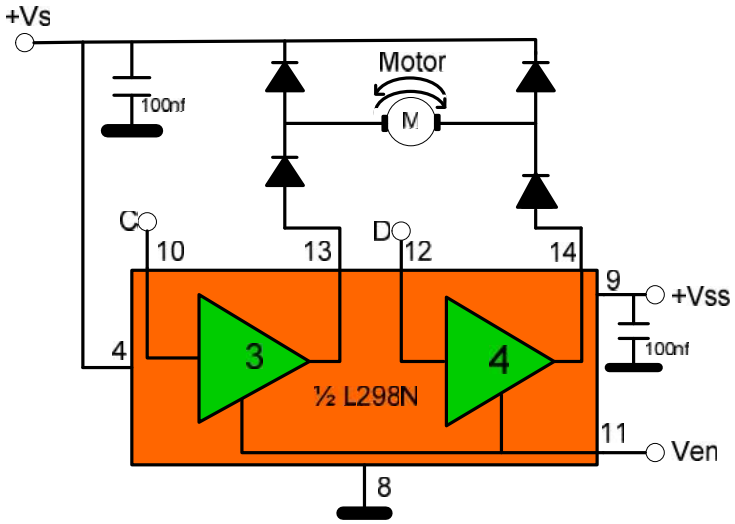


Figura 3.3 Descripción del uso del módulo de fuerza L298N para PWM

Ven	C	D	M1
H	H	L	Girar Hacia Adelante
H	L	H	Girar Hacia atrás
H	C=D		Frenada Rápida del Motor
L	X	X	Libre Corrida del Motor o Freno
L= Low		H= High	X= Don´tcare

Tabla 2 Descripción de entradas salidas de la mitad del integrado L298N

3.1.4. Motores AC

Con respecto a los motores AC van a ser utilizados para las bombas las cuales van a transportar el líquido de los tanques hacía el tanque de mezclado, para ello utilizaremos el método de arranque directo con un relé de estado sólido que active las bombas según el diagrama de bloques. Las protecciones que vamos a utilizar para estos motores van a ser por medio de fusibles. Como se muestra en la figura 9.

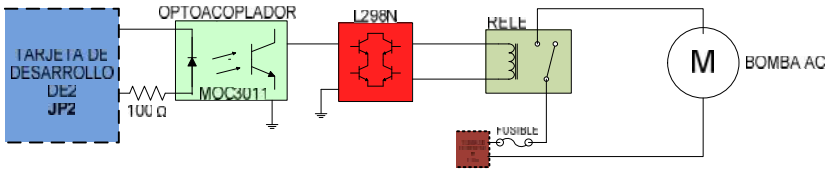


Figura 3.4 Diagrama de conexión entre la Tarjeta DE2 y el motor AC

3.1.5. Motor de Paso

Los motores de paso se han abierto gran campo de la industria y uno de sus mayores utilidades son las bandas transportadoras o platos giratorios y de esa misma manera vamos a utilizar para nuestro proyecto, el motor de paso se va a encargar de desplazar nuestros recipientes los cuales van a recibir el producto finalizado, para la parte de fuerza vamos a usar el integrado L298N, el cual nos va a permitir desplazar lentamente los

recipientes en un plato giratorio y mediante los finales de carrera vamos a detectar la localización del recipiente y llenarlo de forma temporizada. Como lo muestra la figura 10.

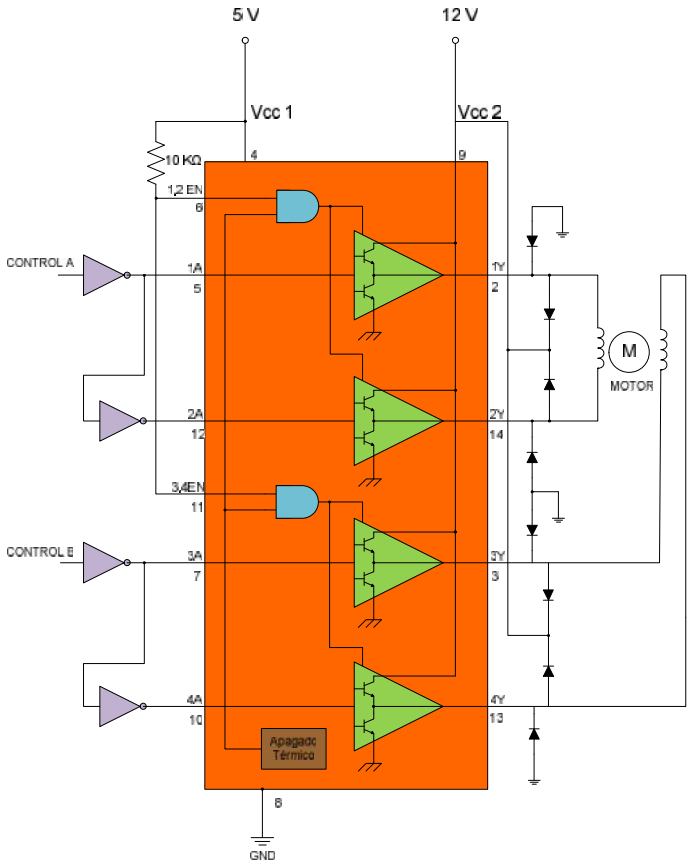


Figura 3.5 Descripción de conexión del módulo de fuerza L298N para Motores de paso.

3.1.6. Sensor Magnético

Los Sensores magnéticos tienen como objetivo detectar la posición de los recipientes dentro del plato giratorio, la conexión con la tarjeta de control DE2 es de forma directa. Figura 11

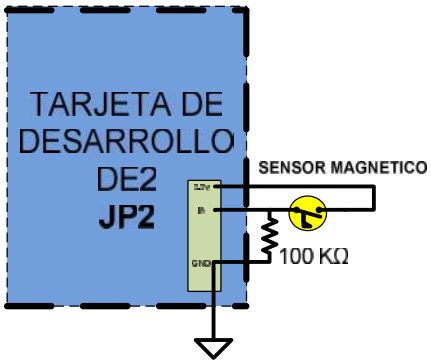


Figura 3.6 Diagrama de conexión entre la Tarjeta DE2 y el Sensor Magnético

3.2. Programación del procesador

Cuando el hardware escogido es configurado se procede con la programación del núcleo dentro del procesador NIOS II el cual se encuentra dentro de la FPGA Cyclone II. Para ello se utilizó el programa NIOS II basado en el entorno Eclipse.

3.2.1. Entorno de Programación

Desde el entorno Eclipse se vuelve más simplificada la programación debido a que se puede realizar una programación

mixta con el Lenguaje C, y el Lenguaje Ensamblador. Sin embargo vamos realizar una programación dentro del lenguaje de programación C, para ello vamos a usar la plantilla con el archivo DE2_Media_Computer.sopcinfo, antes especificado, en el cual se describe el Hardware dentro del FPGA, de esta manera se pueden utilizar las librerías mencionadas.

La librería especial a utilizarse son altera_up_avalon_character_lcd.c, altera_up_avalon_character_lcd.h,altera_up_avalon_character_lcd_regs.h, las cuales nos permiten realizar un correcto manejo de las funciones delChar_LCD_16x2, y de las libreríaaltera_avalon_pio_regs.h, es para el uso de los DIP_Switches y Green_LEDS, y para los Expansion_JP, se realiza una codificación hacia sus punteros bases.

Cuando las librerías se encuentran correctamente definidas, se revisan los ejemplos de su uso en la documentación disponible para utilizarlas correctamente en el proyecto.

Es ventajoso el uso del Lenguaje C, debido a su facilidad de programación con sintaxis y secuencias conocidas. En la figura 12 se observa el entorno de programación.

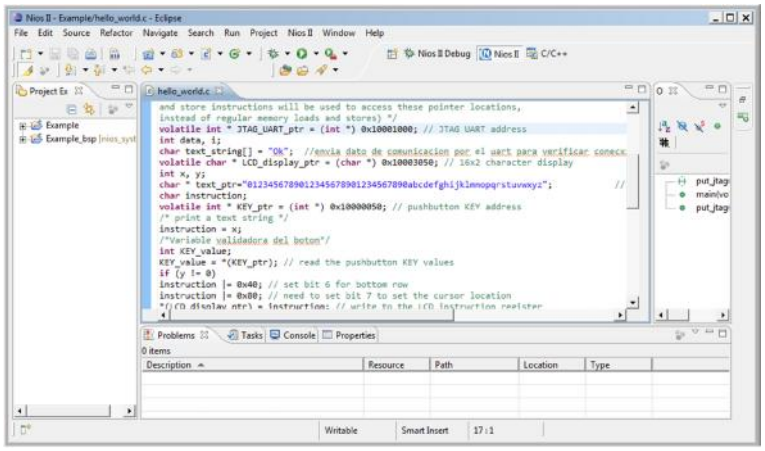


Figura 3.7 Entorno de programación Nios II 12.1 Software Build Tools for Eclipse.

CAPÍTULO 4

4. Análisis de funcionalidad del sistema

Para la funcionalidad del sistema hemos realizado una maqueta, como lo muestra la figura 13, con lo cual vamos a realizar nuestro proceso que lo vamos a indicar por medio de un diagrama de flujo.

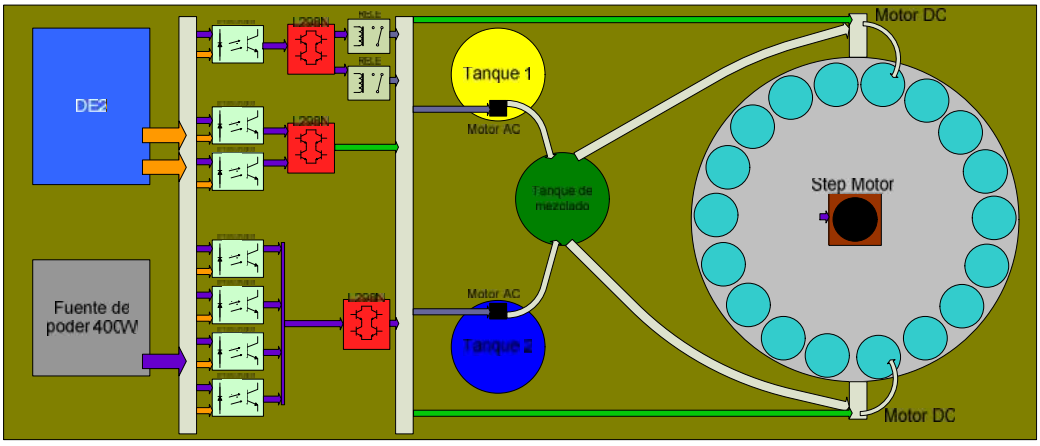


Figura 4.1 Diagrama Esquemático de la maqueta a diseñar

Como podemos observar en la maqueta se puede apreciar que se encuentra claramente separada la parte de las fuente de poder (color gris), tarjetas de control (Color Rojo), las Tarjetas de optoacoplación (color verde claro) y los tanques de materia prima (color amarrillo y azul), tanque de Mezclado (color verde) motores del proceso, por esquema del diagrama no mostramos los sensores magnéticos que se encuentran en el tanque de mezclado y en el exterior del plato giratorio (color gris claro).

4.1. Especificación de funcionalidad del Programa.

El siguiente programa va a funcionar de una forma sincronizada usando sensores y actuadores, en primer lugar nuestra tarjeta DE2 va a iniciar dándole el valor de cero a nuestra variables para no comenzar con ningún dato erróneo, adicional se cargan las librerías como por ejemplo las de la LCD y de E/S, A continuación vamos a presentar mediante la LCD propia de la tarjeta, un mensaje de bienvenida.

A continuación nuestro programa va a detectar si existe algún recipiente para poder depositar nuestra mezcla en caso de no hacerlo nuestro plato se va a mover hasta ubicar el recipiente por medio del motor de pasos, una vez que nuestro recipiente se encuentre la posición indicada detectada por el sensor magnético nuestra DE2 se

va a preguntar si existen mezcla dentro del tanque de no ser así se va a realizar la mezcla usando un temporizador y moviendo nuestro motor DC con caja reductora.

Una vez que ya se haya depositado la cantidad predeterminada en el recipiente nuestro motor DC encargado de depositarlo se va a frenar e inmediatamente se mueve el frasco a una nueva posición para continuar con el proceso.

Un segundo sensor magnético se encuentra en el recipiente para poder detectar si existe líquido mezclado de no ser así se activan las bombas AC para poder depositar el líquido y generar la mezcla.

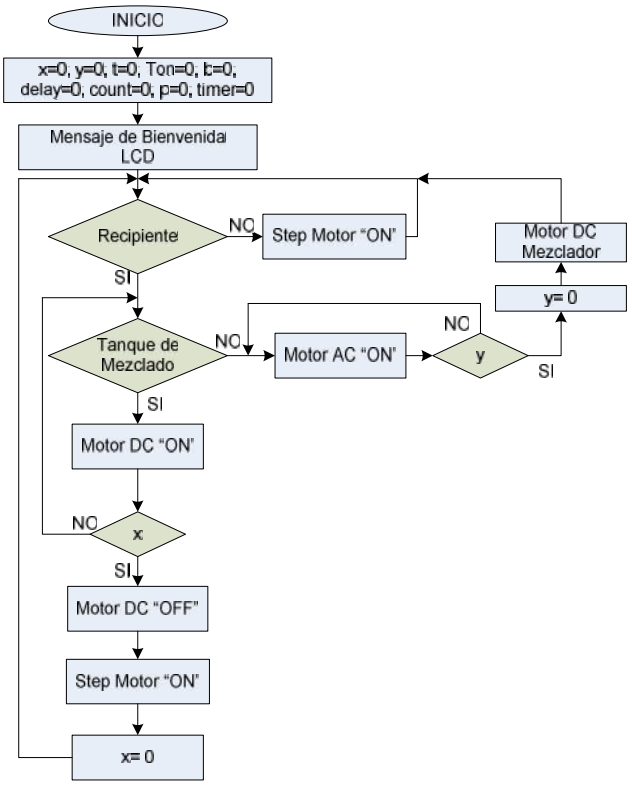


Figura 4.2 Diagrama de Flujo del Programa.

CAPÍTULO 5

5. Pruebas de aplicación

Las pruebas de aplicación fueron realizadas por separado, Primero se probó la fuente de poder la cual va a generar la energía para toda la maqueta figura 15,

A continuación se probó los códigos de manera simulada dentro de la DE2, para verificar la modulación del ancho de pulso y el proceso respectivo usando los LEDs, Además de los mensajes en la LCD figura 15.

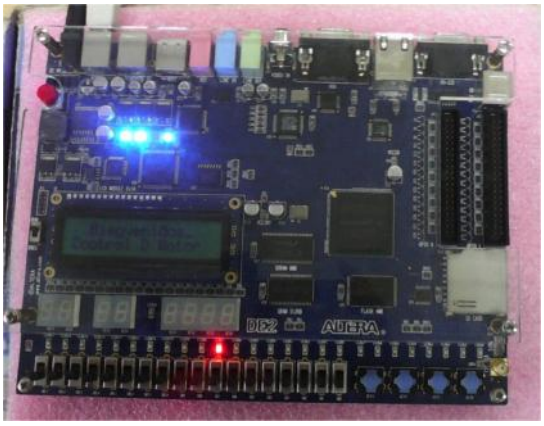


Figura 5.1 Tarjeta DE2 con códigos de prueba

Por separado se realizó pruebas de funcionamiento a la tarjeta optoacoplada, y a la de fuerza, como se muestra en la figura 16.

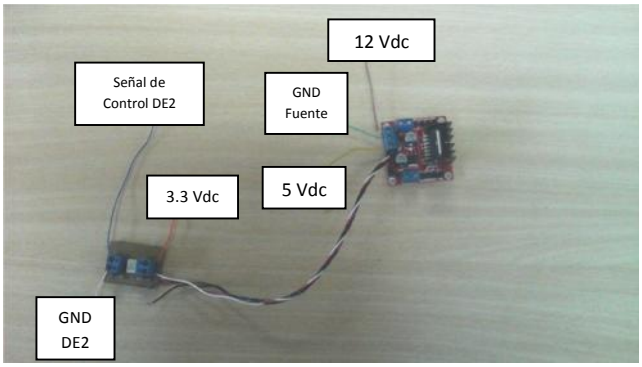


Figura 5.2 Conexión de tarjetas para pruebas

Se demostró además de su funcionalidad, el voltaje correspondiente al ingreso y a la salida de las tarjetas de fuerza.

5.1. Prueba de los motores AC

Para realizar la prueba de motores AC, se siguieron los siguientes pasos:

Primero se realizó un programa en NIOS II, el cual mediante un DIP_Switch simularía el encendido y apagado del motor conectándolo como lo muestra la figura 17.

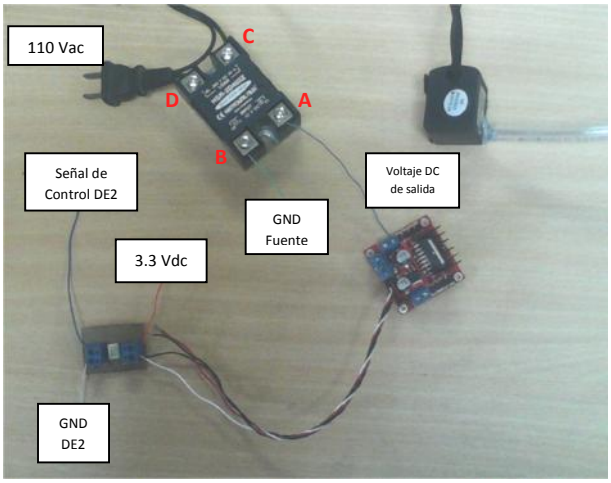


Figura 5.3 Conexión para pruebas del motor AC

Debido a que es un arranque directo sólo se tomó mediciones de voltaje y de corriente en los puntos especificados (punto A, B, C, D).

5.1.1. Datos motor AC

Los datos adquiridos en la prueba del motor AC se van indicar en la siguiente tabla.

DIP_Switch	Red LED	Voltaje AB	Voltaje CD
down	Off	0.41 mV dc	0.35 mV Ac
up	On	11.7 V dc	115.2 V Ac

Tabla 3 Datos de voltaje de la prueba del motor AC

5.2. Prueba de los motores DC

Para la prueba del motor DC al igual que el motor AC se realizó un programa en NIOS II, el cual mediante un DIP_Switch simularía el encendido y apagado del motor conectándolo como lo muestra la figura 18.

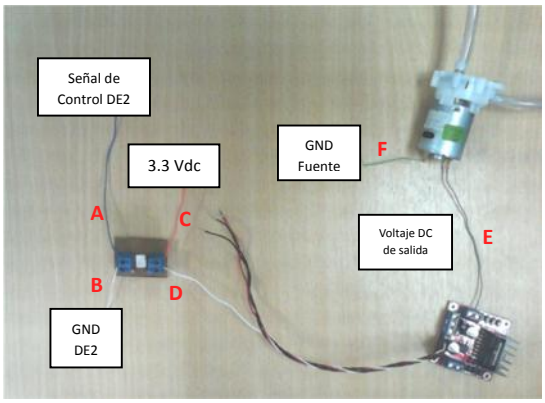


Figura 5.4 Conexión para pruebas del motor DC

Como podemos observar en la figura 18 el motor DC se encuentra conectado mediante la tarjeta optoacoplada y la tarjeta de fuerza, y se va a realizar las pruebas correspondientes en los puntos especificados (puntos A, B, C, D, E, F).

5.2.1. Datos motor DC

Los datos adquiridos en la prueba del motor DC se van a indicar en la tabla 4.

DIP_Switch	Voltaje AB	Voltaje CD	Voltaje EF	Capacidad de llenado ml/10useg
0	0.01mV dc	0.20 mVdc	0.00 mVdc	2.5 ml
1	0.52 V dc	0.48 Vdc	3.22 Vdc	7.5 ml
2	1.15 V dc	1.10 Vdc	6.10 Vdc	15 ml
3	2.3 V dc	2.2 Vdc	9.40 Vdc	22.5 ml
4	3.1 V dc	3.3 Vdc	11.82 Vdc	30 ml

Tabla 4 Datos de voltaje de la prueba del motor DC

5.3. Prueba del motor de pasos

Las pruebas del motor de pasos también se realizaron usando un código para poderlo conectar mediante la tarjeta optoacoplada y la tarjeta de fuerza como lo muestra la siguiente figura 19.

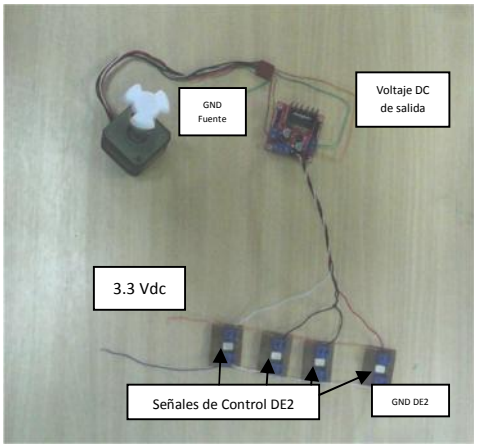


Figura 5.5 Conexión para pruebas del motor de pasos

Usando la secuencia ya antes mencionada en el capítulo III, podemos verificar la posición del motor de pasos.

CONCLUSIONES

1. Hemos concluido que la versatilidad de los FPGA de Altera nos ha permitido generar un sistema embebido adecuado a nuestra aplicación la cual se puede extender más en sistemas de control de lazo cerrado con PID, permitiéndonos diseñar procesadores de señales digitales de muy bajo costo, aminorando el tiempo de diseño ya que el microcontrolador embebido puede ser conectado a compuertas lógicas, contadores y demás. Que también estarían embebidos en el chip. Logrando además un ahorro de espacio físico.
2. Debido a la conclusión anterior y a lo experimentado durante el proyecto durante la construcción de la maqueta concluimos también que el uso en la industrial es viable ya que desde el FPGA se puede controlar arranques de motores, brazos robóticos, activación de solenoides, electroválvulas, bombas, etc. Las cuales son muy comunes en maquinarias y equipos industriales.
   En la industria se usa mucho la comunicación Ethernet, la misma que presenta la FPGA como interfaz de comunicación permitiéndole así ser apto para formar parte de una red neuronal industrial dando como único inconveniente que ninguna de las tarjetas de desarrollo viene encapsulada en una caja hermética con protección internacional (IP) entre 65 y 67.

3. La capacidad de este FPGA de trabajar con una frecuencia de 50 MHz y teniendo la posibilidad de usar un PLL (Phase-Locked Loop o Lazo de seguimiento de fase) propio del FPGA que le da la posibilidad de incrementar la frecuencia de trabajo hasta 200 MHz le permite generar una interfaz fluida entre la máquina y el usuario mediante un sistema grafico la cual puede ser una VGA o una pantalla táctil, aprovechando la salida y el controlador integrado a la tarjeta de desarrollo DE2. Se puede diseñar un sistema SCADA (supervisor de control y adquisición de datos) cubriendo así todos los niveles de control y supervisión exige la industria hoy en día a muy bajos costos económicos pero de gran valor en propiedad intelectual.

RECOMENDACIONES

1. Se recomienda opto aislar todas las salidas de la tarjeta de desarrollo que tengan algún conexión con los sistemas a controlar, ya que la tarjeta tiene salidas análogas y discretas, en nuestro proyecto hemos usado el integrado MOC3011 para las salidas discretas, en el caso de usar las salidas análogas recomendamos el LM300 el cual basado en experiencias previas ha presentado excelentes resultados para opto acoplamiento de señales variables en DC y AC, al realizar el opto aislamiento garantizamos mantener a la tarjeta fuera de daños físicos causador por fallos de voltaje, señales parasitas y demás que puedan ser ocasionados por las cargas de fuerza.
2. En la página Web de Altera, se encuentran toda la información de la estructura de la tarjeta de desarrollo así como también los diagramas realizado en la herramienta de diseño Altium, para realizar la presentación de la LCD, esta tarjeta tiene un controlador HD44780 el cual genera la presentación en la LCD, la secuencia de control debe ser tomada en cuenta a la hora de trabajar con tarjetas de desarrollo que carezcan del controlador de LCD y se desee trabajar con una para interfaz de usuario, así como la DE0-NANO que fue la tarjeta de desarrollo que utilizamos en el seminario y la cual carece de una LCD incorporada.

Anexos

ANEXO 1

Librería altera_avalon_pio_regs.h

```
//altera_avalon_pio_regs.h


#ifndef __ALTERA_AVALON_PIO_REGS_H__
#define __ALTERA_AVALON_PIO_REGS_H__

#include<io.h>

#defineIOADDR_ALTERA_AVALON_PIO_DATA(base)
__IO_CALC_ADDRESS_NATIVE(base, 0)
#defineIORD_ALTERA_AVALON_PIO_DATA(base)            IORD(base, 0)
#defineIOWR_ALTERA_AVALON_PIO_DATA(base, data)      IOWR(base, 0, data)

#defineIOADDR_ALTERA_AVALON_PIO_DIRECTION(base)
__IO_CALC_ADDRESS_NATIVE(base, 1)
#defineIORD_ALTERA_AVALON_PIO_DIRECTION(base)       IORD(base, 1)
#defineIOWR_ALTERA_AVALON_PIO_DIRECTION(base, data)  IOWR(base, 1, data)

#defineIOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base)
__IO_CALC_ADDRESS_NATIVE(base, 2)
#defineIORD_ALTERA_AVALON_PIO_IRQ_MASK(base)        IORD(base, 2)
#defineIOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data)   IOWR(base, 2, data)

#defineIOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base)
__IO_CALC_ADDRESS_NATIVE(base, 3)
#defineIORD_ALTERA_AVALON_PIO_EDGE_CAP(base)        IORD(base, 3)
#defineIOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data)   IOWR(base, 3, data)

#endif/* __ALTERA_AVALON_PIO_REGS_H__ */
```

Libreríaaltera_up_avalon_character_lcd.c

```
/******************************************************************************
*                                                                             *
* License Agreement                                                           *
*                                                                             *
* Copyright (c) 2006 Altera Corporation, San Jose, California, USA.           *
* All rights reserved.                                                        *
*                                                                             *
* Permission is hereby granted, free of charge, to any person obtaining a     *
* copy of this software and associated documentation files (the "Software"),  *
* to deal in the Software without restriction, including without limitation   *
* the rights to use, copy, modify, merge, publish, distribute, sublicense,    *
* and/or sell copies of the Software, and to permit persons to whom the       *
* Software is furnished to do so, subject to the following conditions:        *
*                                                                             *
* The above copyright notice and this permission notice shall be included in  *
* all copies or substantial portions of the Software.                         *
*                                                                             *
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  *
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,    *
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE *
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER      *
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     *
* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER         *
* DEALINGS IN THE SOFTWARE.                                                   *
*                                                                             *
* This agreement shall be governed in all respects by the laws of the State   *
* of California and by the laws of the United States of America.              *
*                                                                             *
******************************************************************************/

#include<string.h>
#include<errno.h>
#include<limits.h>

#include<priv/alt_file.h>

#include"altera_up_avalon_character_lcd.h"
#include"altera_up_avalon_character_lcd_regs.h"

////////////////////////////////////////////////////////////////////////////////
/*
 * InternalUtilityFunctions:
* these functions are not supposed to be called directly by the user so they
 * are not defined in the header file
```

```
*/
//////////////////////////////////////////////////////////////////////////////

/**
* @brief calculate the DDRAM address according to the provided x and y coordinates
*
* @paramx_pos  x coordinate ( 0 to 15, from left to right )
 * @paramy_pos  y coordinate ( 0 for the top row, 1 for the bottom row )
*
* @return the converted address
 * @sa the datasheet for the LCD Display Controller on the DE2 Board
 * @note the function requires that the input are in the valid range
*
 **/
unsignedcharget_DDRAM_addr(unsignedx_pos, unsignedy_pos)
{
        //assumevalid inputs
        unsignedcharaddr = 0x00000000;
        if (y_pos == 0)
        {
                addr |= x_pos;
        }
        else
        {
                addr |= x_pos;
                addr |= 0x00000040;
        }
        // b_7 is always 1 for DDRAM address, see datasheet
        return (addr | 0x00000080);
}

/**
* @brief send a command to the Instruction Register of the LCD controller
*
* @paramlcd -- struct for the LCD Controller device
 * @paramcmd -- the command bits
 *
* @returnnothing
 **/
voidalt_up_character_lcd_send_cmd(alt_up_character_lcd_dev *lcd, unsignedcharcmd)
{
        // NOTE: We use the term Instruction Register and Control Register interchangeably
        IOWR_ALT_UP_CHARACTER_LCD_COMMAND(lcd->base, cmd);
}

//////////////////////////////////////////////////////////////////////////////
/*
 * Interface Functions :
 * The interface of these functions are provided to the user. See the header
 * file for a detailed description of each function
*/
//////////////////////////////////////////////////////////////////////////////
```

```
void alt_up_character_lcd_init(alt_up_character_lcd_dev *lcd)
{
	IOWR_ALT_UP_CHARACTER_LCD_COMMAND(lcd->base,
ALT_UP_CHARACTER_LCD_COMM_CLEAR_DISPLAY);
	// register the device
	// see "Developing Device Drivers for the HAL" in "Nios II Software Developer's
Handbook"
}

alt_up_character_lcd_dev* alt_up_character_lcd_open_dev(constchar* name)
{
// find the device from the device list
// (see altera_hal/HAL/inc/priv/alt_file.h
// and altera_hal/HAL/src/alt_find_dev.c
// for details)
alt_up_character_lcd_dev	*dev	=	(alt_up_character_lcd_dev*)alt_find_dev(name,
&alt_dev_list);

return dev;
}

void alt_up_character_lcd_write(alt_up_character_lcd_dev	*dev,	constchar	*ptr,
unsigned int len)
{
	unsigned int i;
	for (i = 0; i<len; i++)
	{
		IOWR_ALT_UP_CHARACTER_LCD_DATA(dev->base, *(ptr+i));
	}
}

void alt_up_character_lcd_string(alt_up_character_lcd_dev *dev, constchar *ptr)
{
	while ( *ptr )
	{
		IOWR_ALT_UP_CHARACTER_LCD_DATA(dev->base, *(ptr));
		++ptr;
	}
}

// this function isn't used, and is included for future upgrades
int alt_up_character_lcd_write_fd(alt_fd *fd, constchar *ptr, int len)
{
	alt_up_character_lcd_write( (alt_up_character_lcd_dev *) fd->dev, ptr, (unsigned int)
len);
	return 0;
}

int alt_up_character_lcd_set_cursor_pos(alt_up_character_lcd_dev *lcd, unsigned x_pos,
	unsigned y_pos)
{
	//boundarycheck
	if (x_pos> 39 || y_pos> 1 )
```

```
                // invalid argument
                return -1;
        // calculate address
        unsigned char addr = get_DDRAM_addr(x_pos, y_pos);
        // set the cursor
        alt_up_character_lcd_send_cmd(lcd, addr);
        return 0;
}

void alt_up_character_lcd_shift_cursor(alt_up_character_lcd_dev *lcd,
int x_right_shift_offset)
{
        if (x_right_shift_offset == 0)
                // don't ask me to do nothing
                return;

        // see shift right or left
        unsigned char shift_cmd = (x_right_shift_offset> 0) ?
                ALT_UP_CHARACTER_LCD_COMM_CURSOR_SHIFT_RIGHT
:ALT_UP_CHARACTER_LCD_COMM_CURSOR_SHIFT_LEFT;
        // see how many to shift
        unsigned char num_offset = (x_right_shift_offset> 0) ? x_right_shift_offset :
                -x_right_shift_offset;
        // do the shift
        while (num_offset-- > 0)
                alt_up_character_lcd_send_cmd(lcd, shift_cmd);
}

void alt_up_character_lcd_shift_display(alt_up_character_lcd_dev *lcd,
int x_right_shift_offset)
{
        if (x_right_shift_offset == 0)
                // don't ask me to do nothing
                return;

        // see shift right or left
        unsigned char shift_cmd = (x_right_shift_offset> 0) ?
                ALT_UP_CHARACTER_LCD_COMM_DISPLAY_SHIFT_RIGHT
:ALT_UP_CHARACTER_LCD_COMM_DISPLAY_SHIFT_LEFT;
        // see how many to shift
        unsigned char num_offset = (x_right_shift_offset> 0) ? x_right_shift_offset :
                -x_right_shift_offset;
        // do the shift
        while (num_offset-- > 0)
                alt_up_character_lcd_send_cmd(lcd, shift_cmd);
}

int alt_up_character_lcd_erase_pos(alt_up_character_lcd_dev *lcd, unsigned x_pos,
unsigned y_pos)
{
        // boundary check
        if (x_pos> 39 || y_pos> 1 )
                return -1;
```

```
	// get address
	unsignedcharaddr = get_DDRAM_addr(x_pos, y_pos);
	// set cursor to dest point
	alt_up_character_lcd_send_cmd(lcd, addr);
	//send an empty char as erase (refer to the Character Generator ROM part of the Datasheet)
	IOWR_ALT_UP_CHARACTER_LCD_DATA(lcd->base, (0x00000002) );
	return 0;
}

voidalt_up_character_lcd_cursor_off(alt_up_character_lcd_dev *lcd)
{
	alt_up_character_lcd_send_cmd(lcd,
ALT_UP_CHARACTER_LCD_COMM_CURSOR_OFF);
}

voidalt_up_character_lcd_cursor_blink_on(alt_up_character_lcd_dev *lcd)
{
	alt_up_character_lcd_send_cmd(lcd,
ALT_UP_CHARACTER_LCD_COMM_CURSOR_BLINK_ON);
}
```

Librería altera_up_avalon_character_lcd.h

```
#ifndef __ALTERA_UP_AVALON_CHARACTER_LCD_H__
#define __ALTERA_UP_AVALON_CHARACTER_LCD_H__

#include<stddef.h>

#include"sys/alt_dev.h"
#include"sys/alt_alarm.h"
#include"sys/alt_warning.h"

#ifdef __cplusplus
extern"C"
{
#endif/* __cplusplus */

/*
* Device structure definition. Each instance of the driver uses one
* of these structures to hold its associated state.
*/
typedefstructalt_up_character_lcd_dev {
/// @brief character mode device structure
/// @sa Developing Device Drivers for the HAL in Nios II Software Developer's Handbook
alt_devdev;
/// @brief the base address of the device
unsignedintbase;
} alt_up_character_lcd_dev;

// system functions
/**
* @brief Initialize the LCD by clearing its display
*
* @paramlcd -- struct for the LCD Controller device
*
**/
voidalt_up_character_lcd_init(alt_up_character_lcd_dev *lcd);

// file-like operation functions
intalt_up_character_lcd_write_fd(alt_fd *fd, constchar *ptr, unsignedintlen);

// direct operation functions
/**
* @brief Open the character LCD device specified by <em> name </em>
*
* @param name -- the character LCD name. For example, if the character LCD name in
SOPC Builder is "character_lcd_0", then <em> name </em> should be
"/dev/character_lcd_0"
*
* @return The corresponding device structure, or NULL if the device is not found
**/
alt_up_character_lcd_dev* alt_up_character_lcd_open_dev(constchar* name);
```

```
/**
* @brief Write the characters in the buffer pointed to by <em>ptr</em> to
* the LCD, starting from where the current cursor points to
*
* @paramlcd -- struct for the LCD Controller device
* @paramptr -- the pointer to the char buffer
* @paramlen -- the length of the char buffer
*
* @return 0 forsuccess
**/
intalt_up_character_lcd_write(alt_up_character_lcd_dev *lcd, constchar *ptr,
unsignedintlen);

/**
* @brief Set the cursor position
*
* @paramlcd -- struct for the LCD Controller device
* @paramx_pos -- x coordinate ( 0 to 15, from left to right )
* @paramy_pos -- y coordinate ( 1 for the first row, 2 for the second row )
*
* @return 0 forsuccess
**/
intalt_up_character_lcd_set_cursor_pos(alt_up_character_lcd_dev *lcd, unsignedx_pos,
unsignedy_pos);

/**
* @brief Shift the cursor to left or right
*
* @paramlcd -- struct for the LCD Controller device
* @paramx_right_shift_offset -- the number of spaces to shift to the right.Ifthe offset is
* negative, then the cursor shifts to the left.
*
* @return 0 forsuccess
**/
intalt_up_character_lcd_shift_cursor(alt_up_character_lcd_dev *lcd,
intx_right_shift_offset);

/**
* @brief Shift the entire display to left or right
*
* @paramlcd -- struct for the LCD Controller device
* @paramx_right_shift_offset -- the number of spaces to shift to the right.Ifthe offset is
* negative, then the display shifts to the left.
*
* @return 0 forsuccess
**/
intalt_up_character_lcd_shift_display(alt_up_character_lcd_dev *lcd,
intx_right_shift_offset);

/**
* @brief Erase the character at the specified coordinate
*
```

```
* @paramlcd -- struct for the LCD Controller device
* @paramx_pos -- x coordinate ( 0 to 15, from left to right )
* @paramy_pos -- y coordinate ( 1 for the first row, 2 for the second row )
*
* @return 0 forsuccess
**/
intalt_up_character_lcd_erase_pos(alt_up_character_lcd_dev *lcd, unsignedx_pos,
unsignedy_pos);

/*
* Macros used by alt_sys_init
*/
#defineALTERA_UP_AVALON_CHARACTER_LCD_INSTANCE(name, device) \
staticalt_up_character_lcd_dev device = \
{ \
{ \
ALT_LLIST_ENTRY, \
name##_NAME, \
NULL, /* open */ \
NULL, /* close */ \
NULL, /* read */ \
alt_up_character_lcd_write_fd, \
NULL, /* lseek */ \
NULL, /* fstat */ \
NULL, /* ioctl */ \
}, \
name##_BASE, \
}

#defineALTERA_UP_AVALON_CHARACTER_LCD_INIT(name, device) \
{ \
alt_up_character_lcd_init(&device); \
alt_dev_reg(&device.dev);          \
}


#ifdef __cplusplus
}
#endif/* __cplusplus */

#endif/* __ALTERA_UP_AVALON_CHARACTER_LCD_H__ */
```

Librería altera_up_avalon_character_lcd_regs.h

```
/*****************************************************************************
*                                                                            *
* License Agreement                                                          *
*                                                                            *
* Copyright (c) 2003 Altera Corporation, San Jose, California, USA.          *
* All rights reserved.                                                       *
*                                                                            *
* Permission is hereby granted, free of charge, to any person obtaining a    *
* copy of this software and associated documentation files (the "Software"), *
* to deal in the Software without restriction, including without limitation  *
* the rights to use, copy, modify, merge, publish, distribute, sublicense,   *
* and/or sell copies of the Software, and to permit persons to whom the      *
* Software is furnished to do so, subject to the following conditions:       *
*                                                                            *
* The above copyright notice and this permission notice shall be included in *
* all copies or substantial portions of the Software.                        *
*                                                                            *
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR *
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,   *
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE *
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER     *
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING    *
* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER        *
* DEALINGS IN THE SOFTWARE.                                                  *
*                                                                            *
* This agreement shall be governed in all respects by the laws of the State  *
* of California and by the laws of the United States of America.             *
*                                                                            *
*****************************************************************************/

#ifndef __ALT_UP_CHARACTER_LCD_REGS_H__
#define __ALT_UP_CHARACTER_LCD_REGS_H__

#include<io.h>

/*
 * Control Register (When RS = 0)
 * (In the document, we used the name Instruction Register)
 */
#defineALT_UP_CHARACTER_LCD_CONTROL_REG              0
#defineIOADDR_ALT_UP_CHARACTER_LCD_CONTROL(base)     \
      __IO_CALC_ADDRESS_NATIVE(base,
ALT_UP_CHARACTER_LCD_CONTROL_REG)
#defineIORD_ALT_UP_CHARACTER_LCD_CONTROL(base)       \
IORD(base, ALT_UP_CHARACTER_LCD_CONTROL_REG)
```

```
#defineIOWR_ALT_UP_CHARACTER_LCD_CONTROL(base, data)  \
IOWR(base, ALT_UP_CHARACTER_LCD_CONTROL_REG, data)

//#define ALT_UP_CHARACTER_LCD_CONTROL_MSK
(0x000000FF)
//#define ALT_UP_CHARACTER_LCD_CONTROL_OFST
(0x00000000)

// Clear Display
#defineALT_UP_CHARACTER_LCD_CTRL_CLEAR_DISPLAY
        (0x00000001)
// Return Home
#defineALT_UP_CHARACTER_LCD_CTRL_RETURN_HOME
        (0x00000002)
//Entry Mode
#defineALT_UP_CHARACTER_LCD_CTRL_ENTRY_DIR_RIGHT
        (0x00000006)
#defineALT_UP_CHARACTER_LCD_CTRL_ENTRY_DIR_LEFT
        (0x00000004)
#defineALT_UP_CHARACTER_LCD_CTRL_ENTRY_SHIFT_ENABLE (0x00000005)
#defineALT_UP_CHARACTER_LCD_CTRL_ENTRY_SHIFT_DISABLE
        (0x00000004)

// Display ON/OFF Control
#defineALT_UP_CHARACTER_LCD_CTRL_DISPLAY_ON
        (0x0000000C)
#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_ON
        (0x0000000E)
#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_BLINK_ON
        (0x0000000F)

#defineALT_UP_CHARACTER_LCD_CTRL_DISPLAY_OFF
        (0x00000008)

#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_OFF
        (0x0000000C) //equivalent to ALT_UP_CHARACTER_LCD_CTRL_DISPLAY_ON
#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_BLINK_OFF
        (0x0000000E) //equivalent to ALT_UP_CHARACTER_LCD_CTRL_CURSOR_ON

// Cursor/Display Shift
// cause the entire display to move to left or right (the origin of the display is also changed)
#defineALT_UP_CHARACTER_LCD_CTRL_DISPLAY_SHIFT_RIGHT (0x0000001C)
#defineALT_UP_CHARACTER_LCD_CTRL_DISPLAY_SHIFT_LEFT   (0x00000018)
// move the cursor to left or right
#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_SHIFT_RIGHT (0x00000014)
#defineALT_UP_CHARACTER_LCD_CTRL_CURSOR_SHIFT_LEFT
        (0x00000010)

/*
 * Data Register (When RS = 1)
 */
#defineALT_UP_CHARACTER_LCD_DATA_REG              1
#defineIOADDR_ALT_UP_CHARACTER_LCD_DATA(base)       \
```

```
      __IO_CALC_ADDRESS_NATIVE(base, ALT_UP_CHARACTER_LCD_DATA_REG)
#defineIORD_ALT_UP_CHARACTER_LCD_DATA(base)          \
IORD(base, ALT_UP_CHARACTER_LCD_DATA_REG)
#defineIOWR_ALT_UP_CHARACTER_LCD_DATA(base, data)    \
IOWR(base, ALT_UP_CHARACTER_LCD_DATA_REG, data)

//#define ALT_UP_CHARACTER_LCD_DATA_MSK      (0x000000FF)
//#define ALT_UP_CHARACTER_LCD_DATA_OFST     (0)


#endif/* __ALT_UP_CHARACTER_LCD_REGS_H__ */
```

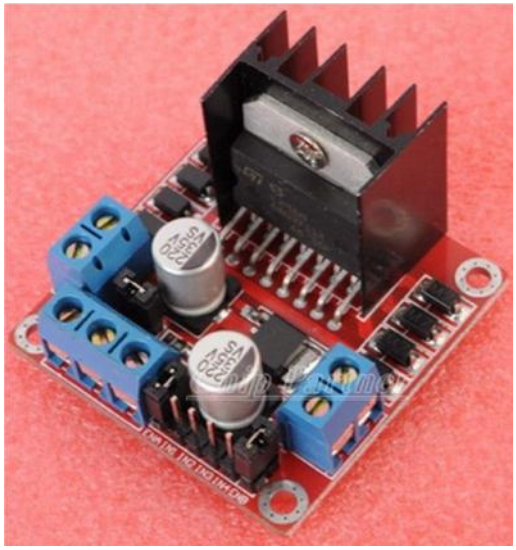
ProgramaCompletohelloworld.c

```
/* function prototypes */
# include<stdio.h>
# include"system.h"
# include"altera_avalon_pio_regs.h"
# include"altera_up_avalon_character_lcd.h"
# include"altera_up_avalon_character_lcd_regs.h"
# define  ADDR_JP1PORT (( volatilechar *) 0x10000060)
# define  ADDR_JP2PORT ((volatilechar *) 0x10000070)

intmain(void)
{
        volatileint *psw = SLIDER_SWITCHES_BASE;
         *(ADDR_JP2PORT+4) = 0; //set every port1B bit direction to input
         *(ADDR_JP1PORT+4) = 0xffffffff; //set every port1A bit dir to output
        alt_up_character_lcd_dev * lcd;
        lcd = alt_up_character_lcd_open_dev(CHAR_LCD_16X2_NAME);

        intlen=15;
        int count = 0;
        int delay = 0;
        int x = 0;
        int y = 0;
        int b = 0;
        intswitchp = *psw;
        intsel = *ADDR_JP2PORT;

                                alt_up_character_lcd_set_cursor_pos(lcd, 0, 0);
                                alt_up_character_lcd_write(lcd, "                ", len);
                                alt_up_character_lcd_set_cursor_pos(lcd, 0, 1);
                                alt_up_character_lcd_write(lcd, "                ", len);
                                //resetea las bombas
                                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE,
0xf0);

        while(b <20000)
        {
                alt_up_character_lcd_set_cursor_pos(lcd, 0, 0);
                alt_up_character_lcd_write(lcd, "Bienvenidos          ", len);
                b++;
         }
        alt_up_character_lcd_set_cursor_pos(lcd, 0, 0);
        alt_up_character_lcd_write(lcd, "                ", len);
        alt_up_character_lcd_set_cursor_pos(lcd, 0, 1);
        alt_up_character_lcd_write(lcd, "                ", len);
        //resetea las bombas
        IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, 0x00);
        switchp = *psw;
        sel = *ADDR_JP2PORT;

        while(1)
         {
```

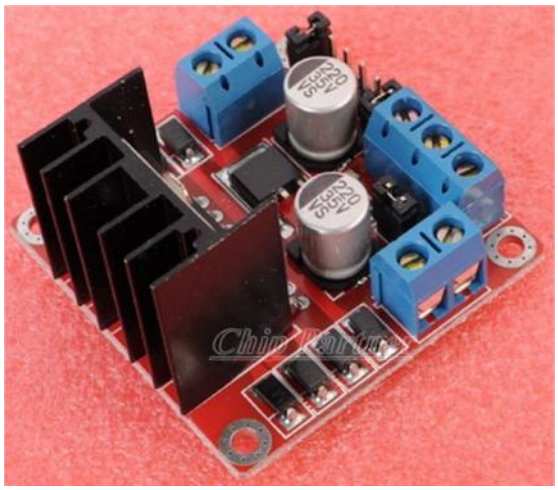
```
                while((switchp& 0x07) == 0x03)
                 {
                        sel = *ADDR_JP2PORT;
                         *ADDR_JP1PORT = sel;
                        IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, 0x03);
                        switchp = *psw;
                 }
                switchp = *psw;
                 *ADDR_JP1PORT = 0x0f;
                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, 0x01);
        }
}
```

Anexo 2

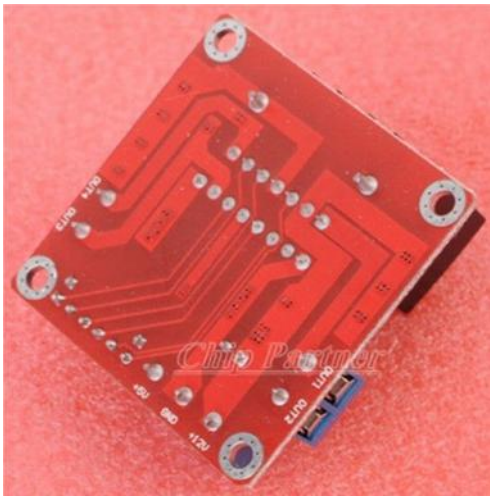
Módulo L298N



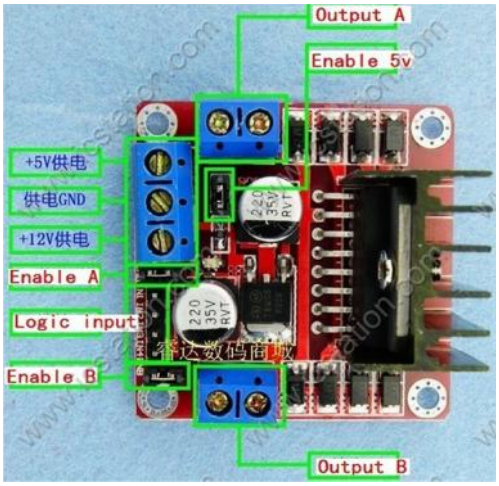
Vista diagonal frontal de tarjeta Driver



Vista diagonal posterior de tarjeta Driver



Vista de las pistas de la tarjeta Driver



Entradas, Salidas e indicadores de tarjeta Driver

Atributos básicos			
nombre	Doble puente H para modulo driver	Modo de trabajo	Controlado por el doble puente H
Chip de control	L298N (ST)		
Voltaje	5V	Voltaje de motor	5V-35V
Corriente	0mA-36mA	Corriente de motor	2A Por puente H
temperatura	-20-(+135) grados Celcius	Potencia máxima	25W
Peso	30g	Dimensiones	43*43*27mm

Casos de aplicación

Control para motor de paso

stepper motor	signal input	step 1	step 2	step 3	step 4	teturn to step1
corotation	IN1	0	1	1	1	return
	IN2	1	0	1	1	return
	IN3	1	1	0	1	return
	IN4	1	1	1	0	return
reversal	IN1	1	1	1	0	return
	IN2	1	1	0	1	return
	IN3	1	0	1	1	return
	IN4	0	1	1	1	return

Tabla de control para step motor

Control para motor DC

DC motor	rotate	IN1	IN2	IN3	IN4	speed adjust PWM signal	
						end	end
M1	corotation	high	low	/	/	high	/
	reversal	low	high	/	/	high	/
	stop	low	low	/	/	high	/
M2	corotation	/	/	high	low	/	high
	reversal	/	/	low	high	/	high
	stop	/	/	low	low	/	high

Tabla de control para motor DC

Anexo 3

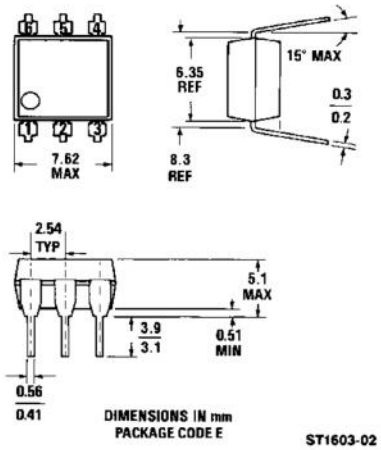
Datasheet de Optoacoplador MOC3011

QT OPTOELECTRONICS

NON-ZERO-CROSSING TRIACS

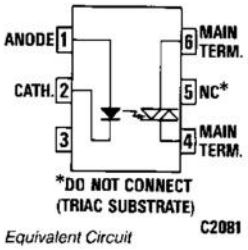
MOC3009 MOC3010
MOC3011 MOC3012

PACKAGE DIMENSIONS



DIMENSIONS IN mm
PACKAGE CODE E

ST1603-02



Equivalent Circuit

C2081

DESCRIPTION

The MOC3009, MOC3010, MOC3011 and MOC3012 are optically isolated triac driver devices. These devices contain a GaAs infrared emitting diode and a light activated silicon bilateral switch, which functions like a triac. This series is designed for interfacing between electronic controls and power triacs to control resistive and inductive loads for 120 VAC operations.

FEATURES

- Low input current required (typically 5mA—MOC3011)
- High isolation voltage—minimum 7500 VAC peak
- Underwriters Laboratory (UL) recognized—File E90700

APPLICATIONS

- Triac driver
- Industrial controls
- Traffic lights
- Vending machines
- Motor control
- Solid state relay

ABSOLUTE MAXIMUM RATINGS

TOTAL PACKAGE

Storage temperature ..... −55°C to 150°C
Operating temperature ..... −40°C to 100°C
Lead temperature
(soldering 10 sec) ..... 260°C
Withstand test voltage ... 7500 VAC Peak (50-60 Hz)

INPUT DIODE

Forward DC current ..... 50 mA
Reverse voltage ..... 3 V
Peak forward current
(1 μs pulse, 300 pps) ..... 3.0 A
Power dissipation (25°C ambient) ..... 100 mW
Derate linearly (above 25°C) ..... 1.33 mW/°C

OUTPUT DRIVER

Off-state output terminal voltage ..... 250 volts
On-state RMS current T_A=25°C ..... 100 mA
(Full cycle, 50 to 60 Hz) T_A=70°C ..... 50 mA
Peak nonrepetitive surge current ..... 1.2 A
(PW=10 ms, DC=10%)
Total power dissipation @ T_A=25°C ..... 300 mW
Derate above 25°C ..... 4.0 mW/°C

QT OPTOELECTRONICS

ELECTRO-OPTICAL CHARACTERISTICS (25°C Temperature Unless Otherwise Specified)

INDIVIDUAL COMPONENT CHARACTERISTICS

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
INPUT DIODE						
Forward voltage	V_F		1.2	1.50	V	I_F=10 mA
Junction capacitance	C_J		50		pF	V_F=0 V, f=1 MHz
Reverse leakage current	I_R			100	μA	V_R=3.0 V
OUTPUT DETECTOR						
Peak blocking current, either direction	I_{DRM}	—		100	nA	V_{DRM}=250 V, Note 1
Peak on-state voltage, either direction	V_{TM}	—	2.0	3.0	Volts	I_{TM}=100 mA Peak

Note 1. Test voltage must be applied within dv/dt rating.

TRANSFER CHARACTERISTICS

DC CHARACTERISTICS		SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
LED trigger current (current required to latch output)	MOC3009	I_{FT}	—	15.0	30	mA	Main terminal voltage=3.0 V, R_L = 150Ω
	MOC3010	I_{FT}	—	10.0	15	mA	
	MOC3011	I_{FT}	—	5	10	mA	
	MOC3012	I_{FT}	—	—	5	mA	
Holding current		I_H	—	100	—	μA	Either direction

TRANSFER CHARACTERISTICS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
AC dv/dt RATING Critical rate of rise of off-state voltage	dv/dt	—	12.0	—	V/μs	Static dv/dt (see Fig. 4)
Critical rate of rise of commutating voltage	dv/dt	—	0.2	—	V/μS	Commutating dv/dt I_{LOAD}=15 mA (see Fig. 4)

ISOLATION CHARACTERISTICS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
Isolation voltage	V_{iso}	5300			V_{AC}RMS	$I_{I-O} \leq 1$ μA, 1 Minute
	V_{iso}	7500			V_{AC}PEAK	$I_{I-O} \leq 1$ μA, 1 Minute
Isolation resistance	R_{iso}	10^{11}			ohms	V_{I-O}=500 VDC
Isolation capacitance	C_{iso}		0.5		pF	f=1 MHz

Anexo 4

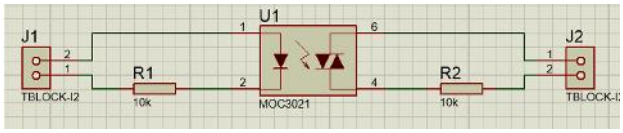
Tarjetas de Opto acoplado

Descripción.-

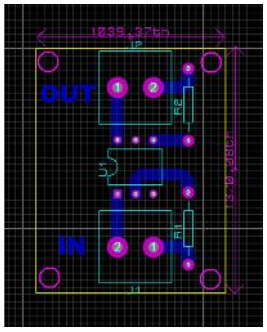
La tarjeta de opto acoplado fue construida para aislar la señal de 3.3V de la tarjeta de desarrollo DE2 la cual es utilizada como el voltaje de control con el voltaje de fuerza al cual sería sometido los periféricos del circuito, ya que en este proyecto se tienen además 3.3V, 5V y 12V provenientes de la fuente y la cual se encarga de alimentar a todos los motores que posee la maqueta.

La tarjeta de opto acoplado es un diseño sencillo en el cual se usa el MOC3011 para hacer el aislamiento de señales.

Se utilizó el programa Proteus 8.0 para realizar el diseño



Se procedió a usar el programa Ares para realizar las pistas



Se construyeron 8 tarjetas las cuales trabajan dentro de la maqueta que opto aíslan señales de la DE2 para el control del motor de paso, motor DC y motor AC.

Anexo 5

Rele de estado Solido

HFS15(JGX-1505FB) SOLID STATE RELAY

c UL us
File No: E133481

File No.: B050453286001(D-240 type)

CQC
File No: CQC02001001936

Features

- 4000V dielectric strength
- Photo isolation
- Removable finger proof cover available
- Built-in snubber
- Zero cross or random turn-on
- TRIAC AC output
- Panel mount
- DC or AC control
- With LED indicator or not
- RoHS compliant

INPUT

Control voltage range (DC input)	3 to 32VDC (Without LED) 4 to 32VDC (With LED)
Control voltage range (AC input)	85 to 132VAC (110V input) 175 to 264VAC (220V input) 19.2 to 28.8VAC (24V input)
Must operate voltage (DC input)	Max. 3VDC
Must operate voltage (AC input)	85VAC (110V input) 175VAC (220V input) 19.2VAC (24V input)
Must release voltage (DC input)	Min. 1.0VDC
Must release voltage (AC input)	10VAC (110V, 220V input) 2VAC (24V input)
Max. input current	25mA (DC input) 15mA (AC input)
Max. reverse protection voltage (DC input)	-32VDC

OUTPUT

Type	D-□A10□	D-□A15□	D-□A20□	D-□A25□	D-□A40□
Load voltage range (at 47-63HZ)	D240	48 to 280VAC			
	D380	48 to 400VAC			
Transient overvoltage	D240	600Vpk			
	D380	800Vpk			
Load current range(A)	0.1 to 10	0.1 to 15	0.1 to 20	0.1 to 25	0.1 to 40
Max.I^2t for fusing(10ms, A^2s)	78	144	312	312	880
Max. surge current(10ms)	100Apk	150Apk	200Apk	250Apk	400Apk
Max. leakage current	5mA	5mA	5mA	5mA	5mA
Max. on-state voltage drop	1.5VAC				
Max. turn-on time	Zero cross turn on: 1/2 cycle+1ms Random turn-on: 1ms				
Max. turn-off time	1/2 cycle+1ms				
Min. off-state (dv/dt)	200V/μs				
Min. power factor	0.5				

GENERAL

Type		D-□A10□	D-□A15□	D-□A20□	D-□A25□	D-□A40□
Dielectric strength (input to output)		4000VAC, 50/60Hz, 1min.				
Insulation resistance		1000MΩ (at 500VDC)				
Ambient temperature	Operating	-30°C to +80°C				
	Storage	-30°C to +100°C				
Unit weight		Typ. 88g				

Notes: All parameters at 25°C.

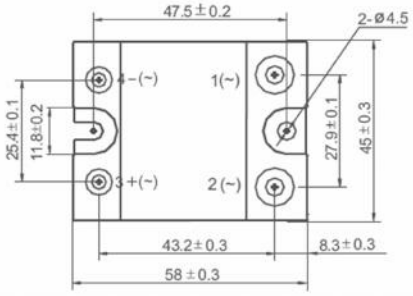
DESCRIPTION

The HFS15 offer 3-32VDC, 24VAC, 110VAC or 220VAC input control, with outputs rated at 10A, 15A, 20A, 25A or 40A. All models include an internal snubber. The relays provide 4000VAC opto-isolation, between input and output. Outline dimension is 58.4mmX45.7mmX22.9mm.

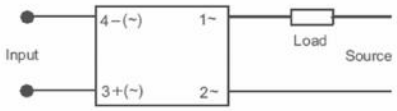
PRECAUTIONS

1. When choosing a SSR, please notice the actual load current and working ambient temperature. To use the SSR correctly, please refer to CHARACTERISTIC DATA and make sure the heat sink size when it works in full load current.
2. Apply heat-radiation silicon grease of a heat conductive sheet between the SSR and heat sink. There will be a space between the SSR and heat sink Attached to the SSR. Therefore, the generated heat of the SSR cannot be radiated properly without the grease. As a result, the SSR may be overheated and damaged or deteriorated.
3. Tighten the SSR terminal screws properly. If the screws are not tight, the SSR will be Damaged by heat generated when the power in ON. Perform wiring using the tightening torque shown in the following table.

Screw size	Recommended tightened torque
M3	0.58 to 0.98 N·m
M4	0.98 to 1.37 N·m

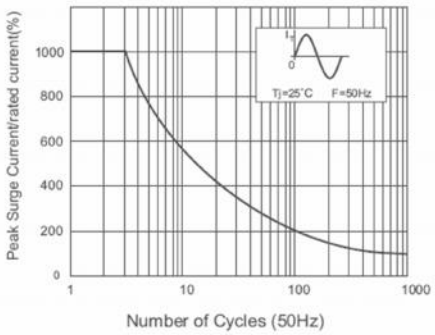


Wiring Diagram

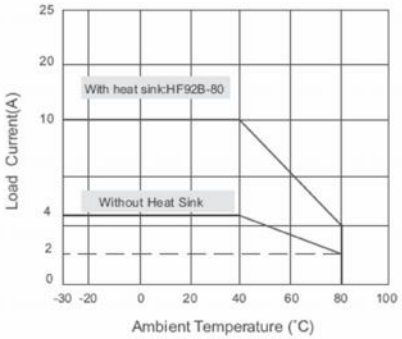


CHARACTERISTIC CURVE

Max.Surge Current vs.Number of Cycles



Max. Load Current vs. Ambient Temp. (10A)



ORDERING INFORMATION

Type	HFS15 /	D-	240	A	10	Z	-L XXX

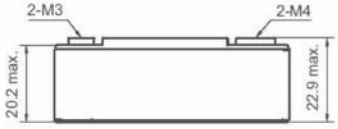
Input voltage	**D**: 3 to 32VDC (Without LED) 4 to 32VDC (With LED) **24A**: 24VAC **110A**: 110VAC **220A**: 220VAC
Load voltage	**240**: 240V **380**: 380V
Load voltage form	**A**: AC
Load current	**10**: 10A **15**: 15A **20**: 20A **25**: 25A **40**: 40A
Zero cross function	**Z**: Zero cross turn-on **P**: Random turn-on
LED indicator	**L**: With LED **Nil**: Without LED
Special request code	(Only for special requirements, e.g. 555 stand for RoHS compliant)

Notes: HFS15 is an environmental friendly product, please mark special code (555) when order.

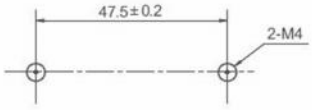
OUTLINE DIMENSIONS, WIRING DIAGRAM AND MOUNTING HOLES

Unit: mm

Outline Dimensions



Mounting Hole Layout



Anexo 6

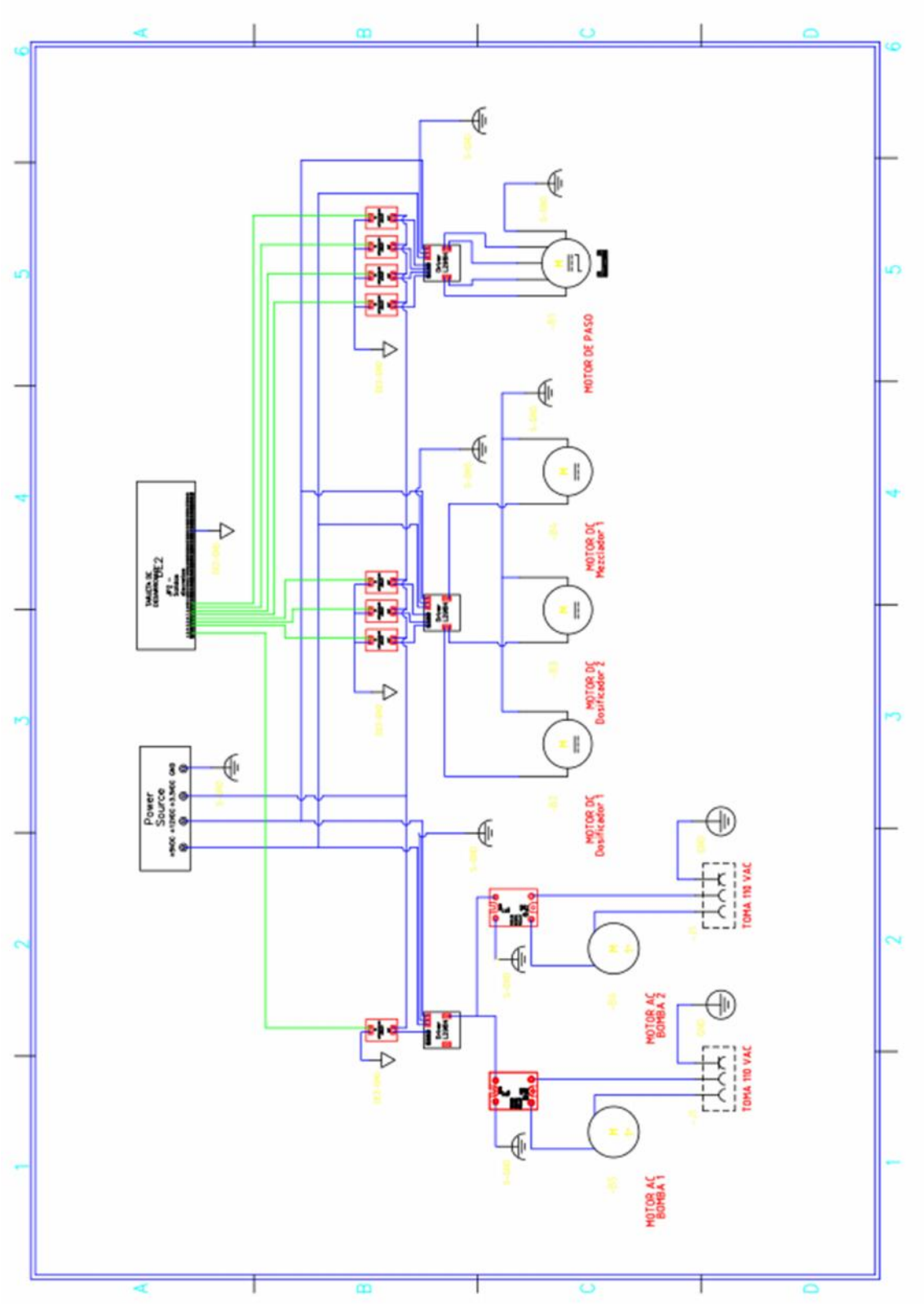
Diagrama esquemático del Proyecto

En el diagrama esquemático están todas las conexiones que se realizaron en la maqueta de presentación de la tesis, este esquemático se realizó en Autocad Electrical, y se ha adjuntado una copia del archivo en el CD de la tesis.

Como lo muestra el esquemático a continuación, con la DE2 se controlaron 6 motores para el proceso simulado, entre los cuales podemos contar 3 motores DC, 1 motor de paso y dos motores de corriente alterna.

Se utilizaron 8 salidas de la tarjeta de desarrollo para controlar dichos motores.

Los sistemas de optoacoplado se utilizaron para aislar el voltaje entregado por las salidas de la tarjeta de desarrollo y el circuito de fuerza de los motores.


MOTOR DE PASO
MOTOR DC Mezclador 1
MOTOR DC Dosificador 2
MOTOR DC Dosificador 1
MOTOR AC BOMBA 2
MOTOR AC BOMBA 1
TOMA 110 VAC
TOMA 110 VAC
Power Source

Asignación de señales DE2 salida JP2		
JP2	**Señal**	**Descripción**
2	CBAC	Bit de control de bombas AC
4	CDDC1	Bit de control del dosificador 1
5	CDDC2	Bit de control del dosificador 2
6	CMDC1	Bit de control del mezclador 1
7	CStep1	Bit 1 de control de motor de paso
8	CStep 2	Bit 2 de control de motor de paso
9	CStep 3	Bit 3 de control de motor de paso
10	CStep 4	Bit 4 de control de motor de paso
30	DE2-GND	Referencia a tierra de la tarjeta de desarrollo

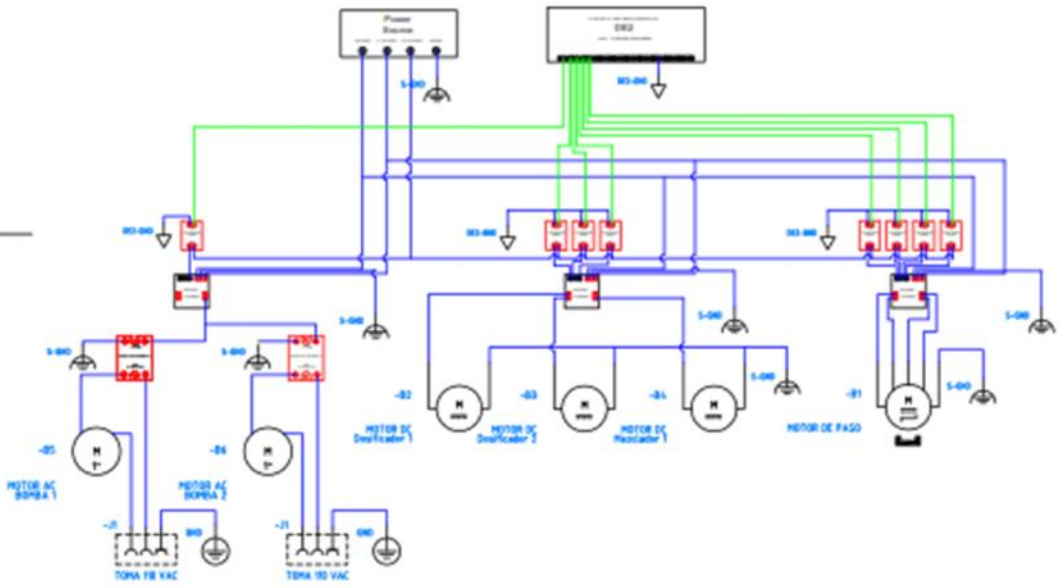
Otras señales	
Nombre	**Descripción**
S-GND	Referencia de fuente de alimentación
GND	Puesta a tierra de tomacorriente de 110VAC
+5VDC	Alimentación de fuente de 5 voltios
+12VDC	Alimentación de fuente de 12 voltios
+3.3VDC	Alimentación de fuente de 3.3 voltios

Escuela Superior Politecnica del Litoral

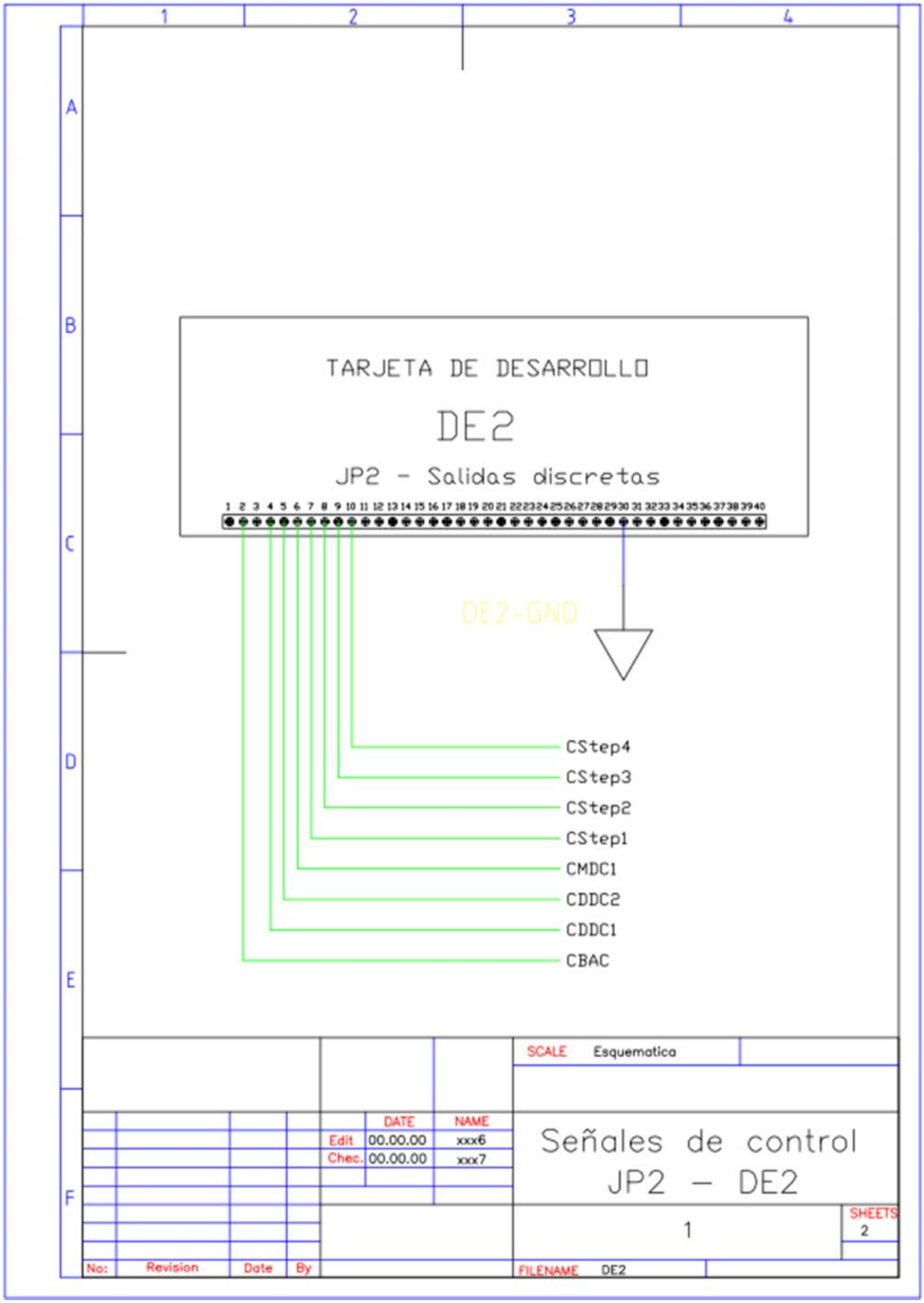
Facultad de Ingenieria Electrica y Computacion

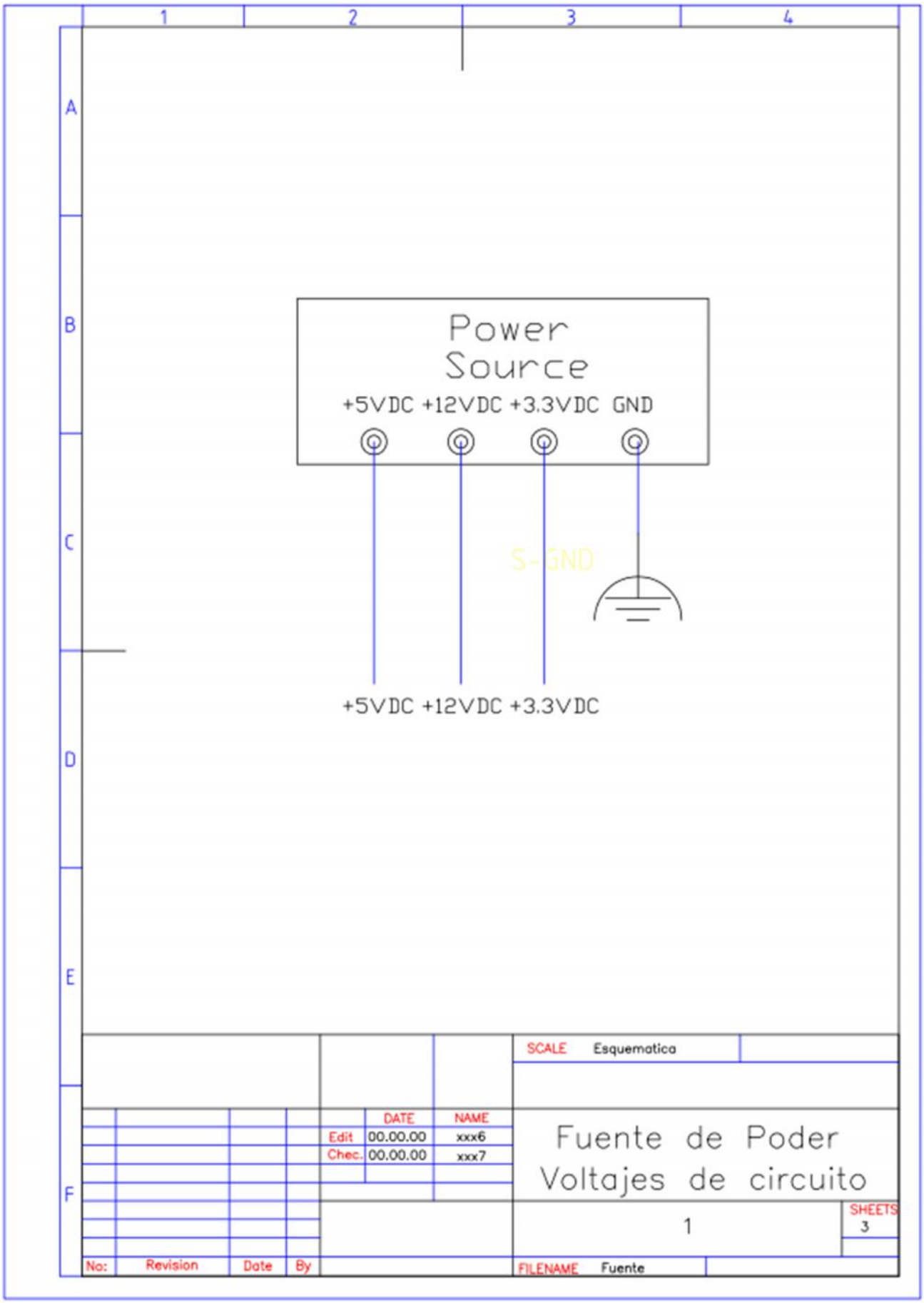
Control de Motores usando Plataforma NIOS II

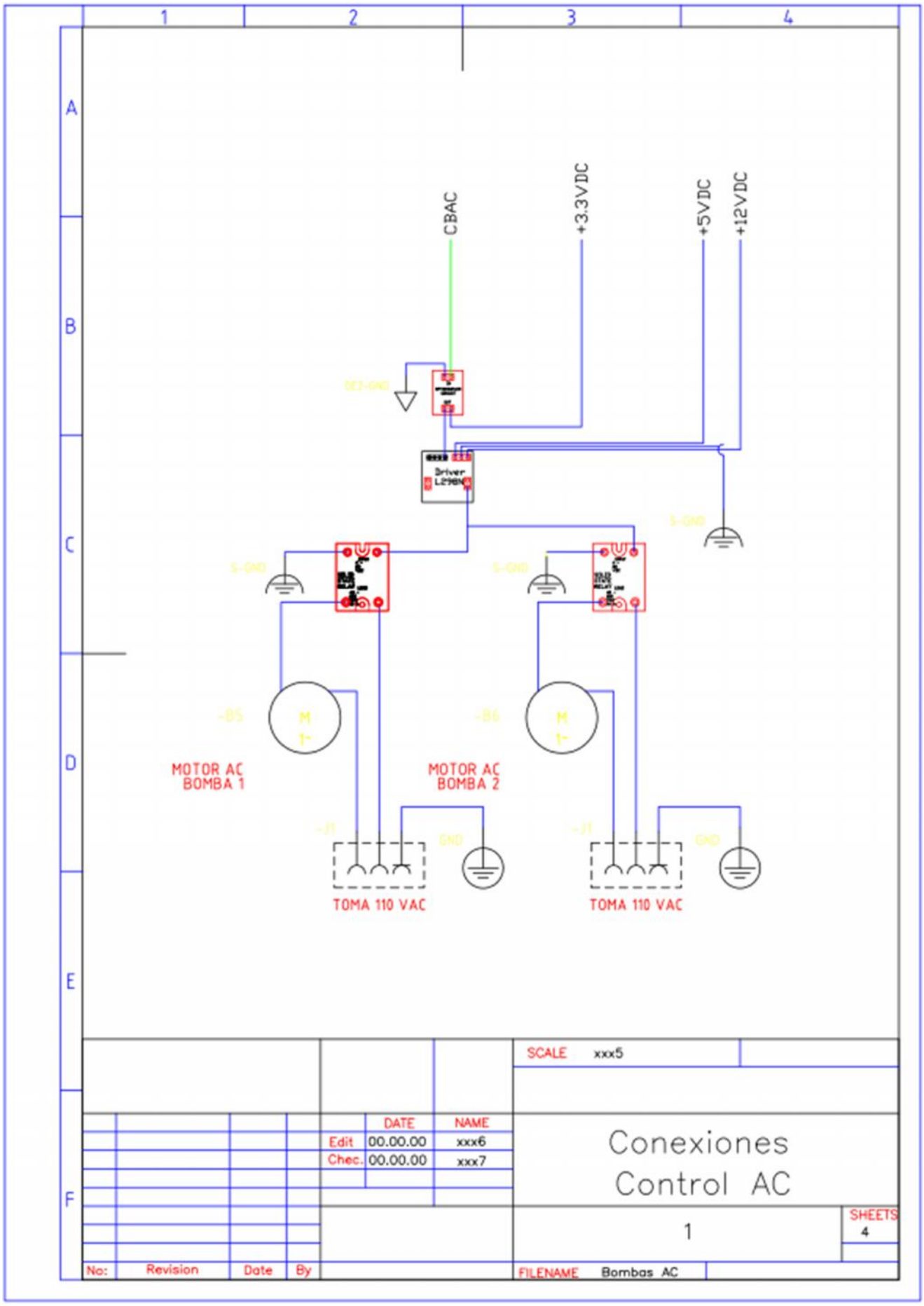
Diagrama Esquematico
Conexiones Realizadas

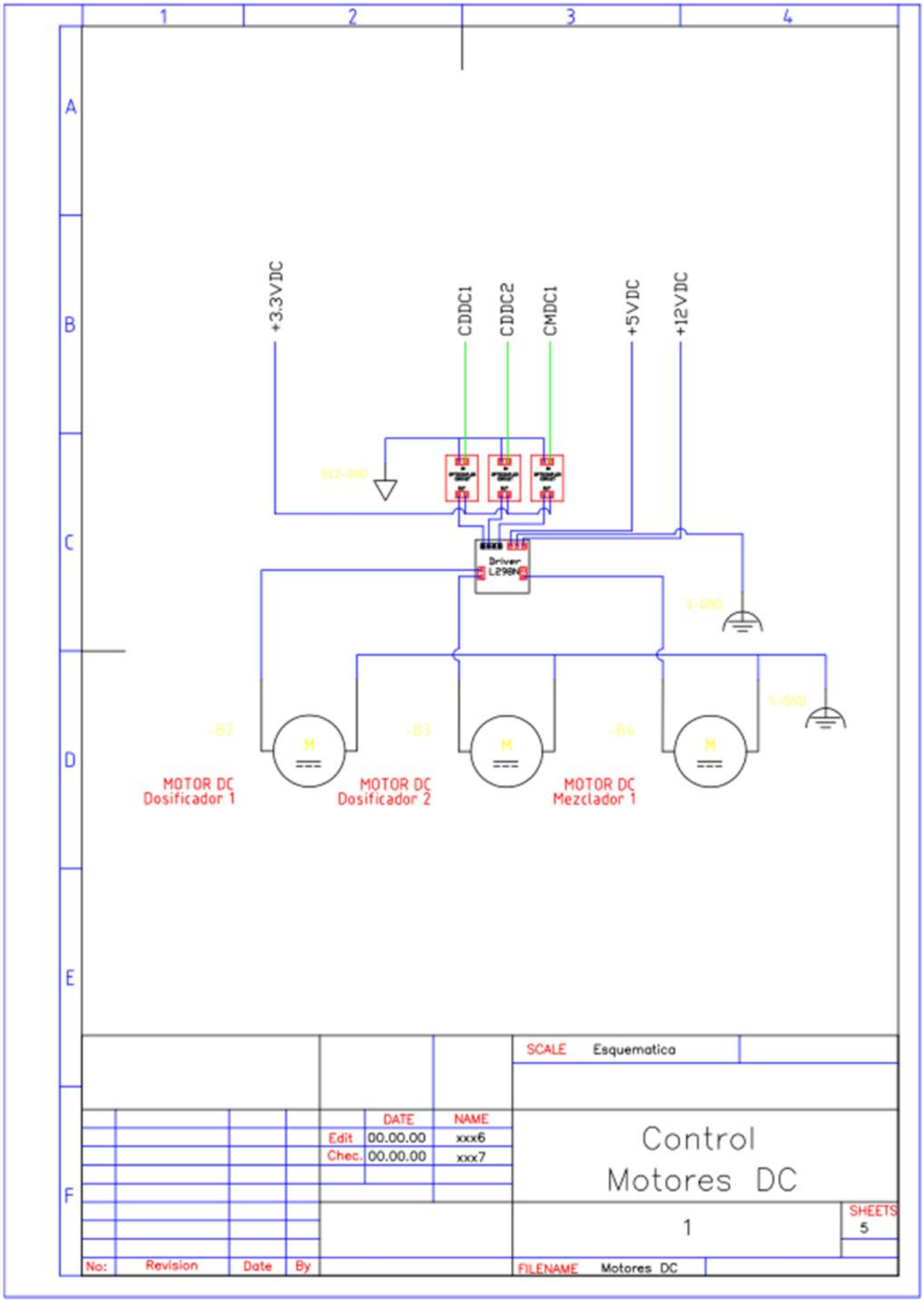


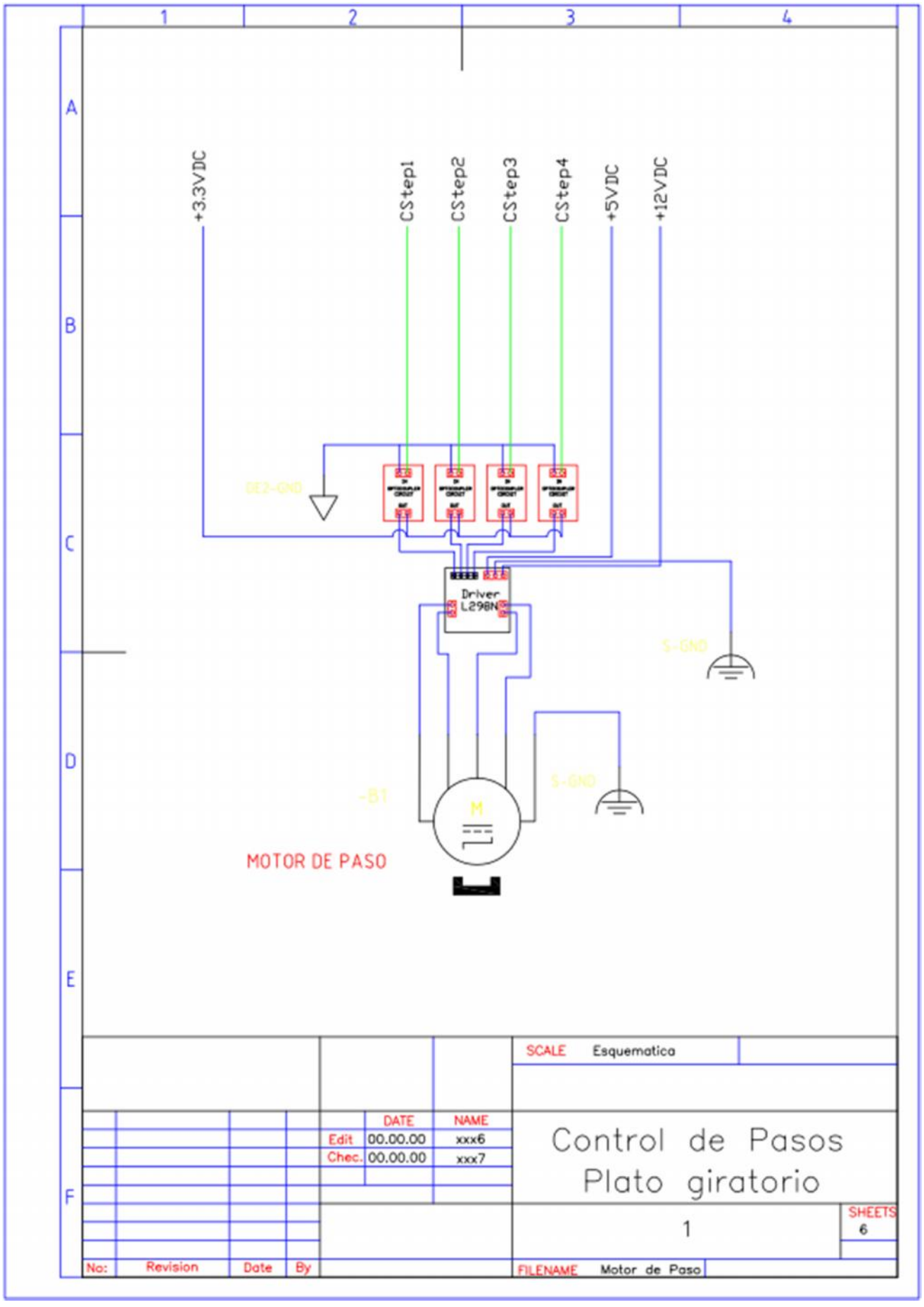
SCALE	Esquematica		
		DATE	NAME
Edit		00.00.00	xxx6
Chec.		00.00.00	xxx7
Controlador Motores			
1		SHEETS	1
No:	Revision	Date	By
FILENAME	Conexiones		


1
2
3
4
A
B
C
D
E
F
TARJETA DE DESARROLLO
DE2
JP2 - Salidas discretas
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
DE2-GND
CStep4
CStep3
CStep2
CStep1
CMDC1
CDDC2
CDDC1
CBAC
SCALE Esquematica
DATE
NAME
Edit 00.00.00 xxx6
Chec. 00.00.00 xxx7
Señales de control
JP2 – DE2
1
SHEETS
2
No:
Revision
Date
By
FILENAME DE2


1
2
3
4
A
B
C
D
E
F
Power
Source
+5VDC +12VDC +3.3VDC GND
S-GND
+5VDC +12VDC +3.3VDC
SCALE Esquematica
DATE NAME
Edit 00.00.00 xxx6
Chec. 00.00.00 xxx7
Fuente de Poder
Voltajes de circuito
1
SHEETS
3
No: Revision Date By
FILENAME Fuente


1
2
3
4
A
B
C
D
E
F
CBAC
+3.3VDC
+5VDC
+12VDC
S-GND
S-GND
S-GND
-B5
-B6
M
1~
MOTOR AC
BOMBA 1
MOTOR AC
BOMBA 2
-J1
-J1
GND
GND
TOMA 110 VAC
TOMA 110 VAC
SCALE xxx5
DATE
NAME
Edit 00.00.00 xxx6
Chec. 00.00.00 xxx7
Conexiones
Control AC
1
SHEETS
4
No: Revision Date By
FILENAME Bombas AC


1
2
3
4
A
B
C
D
E
F
+3.3VDC
CDDC1
CDDC2
CMDC1
+5VDC
+12VDC
Driver
L298N
S-GND
S-GND
-B2
-B3
-B4
M
M
M
MOTOR DC
Dosificador 1
MOTOR DC
Dosificador 2
MOTOR DC
Mezclador 1
SCALE Esquematica
DATE
NAME
Edit 00.00.00 xxx6
Chec. 00.00.00 xxx7
Control
Motores DC
1
SHEETS
5
No: Revision Date By
FILENAME Motores DC


1
2
3
4
A
B
C
D
E
F
+3.3VDC
CStep1
CStep2
CStep3
CStep4
+5VDC
+12VDC
GE2-GND
Driver
L298N
S-GND
S-GND
-B1
M
MOTOR DE PASO
SCALE Esquematica
DATE
NAME
Edit 00.00.00 xxx6
Chec. 00.00.00 xxx7
Control de Pasos
Plato giratorio
1
SHEETS
6
No: Revision Date By
FILENAME Motor de Paso

BIBLIOGRAFÍA

[1] National Instrument, FPGAs a Fondo

http://www.ni.com/white-paper/6983/es ,2 de Enero del 2014

[2] Altera, Cyclone II FPGAs at Cost That Rivals ASICs

http://www.altera.com/devices/fpga/cyclone2/cy2-index.jsp ,2 de Enero del 2014

[3] Altera, DE2 Development and Education Board

http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html ,2 de Enero del 2014

[4] Universidad de Toronto, GPIO Puerto 1 y 2

http://www-ug.eecg.toronto.edu/desl/nios_devices/dev_pit.html ,2 de Enero del 2014

[5] Ayuda de manuales de la compañía Altera,

ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

http://www.altera.com/literature/lit-cyc2.jsp, 5 de Enero del 2014

[6] Datasheet de los integrados

http://www.datasheetcatalog.com/, 5 de Enero del 2014

[7] Datasheet DE2

http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html, 5 de Enero del 2014

[8] Secuencia de Motores de Paso

Es.m.wikipedia.org/wiki/Motor_paso_a_paso, 5 de Enero del 2014

[9] Motores de Corriente Alterna

http://platea.pntic.mec.es/ ̃jgarrigo/SAP/archivos/1eva/introducción_motores_ca., 5 de Enero del 2014