CONTROL DE MOTORES DE CORRIENTE CONTINUA A TRAVÉS DEL PUERTO USB, USANDO UNA INTERFAZ WEB

CARLOS ARTURO OROZCO RAMIREZ
FABIO ALEXANDER CANO ZULUAGA

UNIVERSIDAD DE MANIZALES
FACULTAD DE CIENCIAS E INGENIERÍA
TECNOLOGÍA INFORMÁTICA
MANIZALES
2015

CONTROL DE MOTORES DE CORRIENTE CONTINUA A TRAVÉS DEL PUERTO USB, USANDO UNA INTERFAZ WEB

CARLOS ARTURO OROZCO RAMIREZ
FABIO ALEXANDER CANO ZULUAGA

Trabajo de Grado presentado como opción parcial para optar
al título de Tecnólogos Informáticos

Presidente
LUIS CARLOS CORREA ORTIZ
Ingeniero Electrónico

UNIVERSIDAD DE MANIZALES
FACULTAD DE CIENCIAS E INGENIERÍA
TECNOLOGÍA INFORMÁTICA
MANIZALES
2015

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del Jurado

_____
Firma del Jurado

Manizales, abril de 2015

AGRADECIMIENTOS

A nuestros padres, familiares y amigos más cercanos…
Mil bendiciones por su apoyo incondicional

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

El presente documento sintetiza el proceso para realizar una aplicación web para el control de motores de corriente continua convencionales a través del puerto USB de la computadora para el envío de las señales de control de giro y velocidad, las cuales llegan a una tarjeta electrónica construida con base en un microcontrolador el cual está programado para recibir las señales de la aplicación, interpretarlas y transmitirlas hacia los diferentes mecanismos actuadores que en este caso son los motores CC convencionales.

PALABRAS CLAVE: Microcontrolador, Puerto USB, Motores de corriente continua, Aplicación web.

ABSTRACT

This paper summarizes the process for a web application to control conventional DC motors via the USB port for sending control and speed of rotation signals, which reach an electronic card built based on a microcontroller which is programmed to receive signals from the application, to interpret and transmit the various actuator mechanisms in this case are the conventional DC motor.

KEY WORDS: Microcontrollers, USB port, DC motors, web application.

INTRODUCCIÓN

Las tecnologías de la información y las comunicaciones, TIC, están presentes en la mayoría de las labores emprendidas por el ser humano hoy en día. Una de ellas es, sin duda, la automatización industrial. Ésta pretende, mediante el uso de sistemas de control unidos con la informática, disminuir la intervención humana en los diferentes procesos industriales; lo cual permite la realización de procesos repetitivos y de alto riesgo para la integridad física de las personas; logrando sistemas de producción más eficientes, de mayor calidad y con menor desperdicio en tiempo y material, obteniendo acabados de precisión en los diferentes productos permitiendo el mejoramiento de la calidad en los sistemas de producción.

Es importante tener en cuenta que la tecnología informática, en combinación con procesos y mecanismos industriales, se ha convertido en una herramienta importante en el diseño, implementación y monitoreo de sistemas de control. Muchos de los sistemas de control consisten en aplicaciones informáticas que utilizando interfaces de comunicación controlan dispositivos electrónicos los cuales se han convertido en controles lógicos programables (PLC), redes neuronales artificiales (ANN), sistemas de control distribuidos (DCS), controles de supervisión y adquisición de datos (SCADA), instrumentación, robótica y control de movimiento; por mencionar algunos ejemplos de sistemas de automatización y control.

Este documento es una propuesta para el desarrollo de un proyecto como opción de grado en el programa Tecnología Informática de la Facultad de Ciencias e Ingeniería de la Universidad de Manizales; el cual consiste en el desarrollo de una aplicación web utilizando como base fundamental el lenguaje de programación JAVA, para el control de motores de corriente continua convencionales, por medio de una computadora empleando como interface de comunicación el puerto USB, a través de un microcontrolador PIC de Microchip.

1. ÁREA PROBLEMÁTICA

Con el constante avance de la tecnología en el campo de las computadoras se ve que el uso de los puertos serial y paralelo ha quedado relegado por completo, dando paso a la tecnología de los puertos USB; este cambio significativo implica que muchas aplicaciones, sobre todo aquellas que mediante el uso de microcontroladores que permitían una comunicación desde la computadora a través de los puertos serial y paralelo; tengan que adaptarse a esta nueva realidad.

Durante muchos años las aplicaciones desarrolladas para el control de dispositivos eléctricos y electrónicos fueron realizadas con lenguajes de programación de alto nivel como el lenguaje C; el cual se popularizó como herramienta de desarrollo de estas aplicaciones; pero no se puede desconocer el hecho de que el paradigma de la programación orientada a objetos brinda una herramienta excepcional en el desarrollo de aplicaciones para el control de dichos dispositivos. Sin embargo, lenguajes tan potentes como JAVA, el cual fue desarrollado bajo el modelo de programación orientada a objetos, se ha visto relegado en el desarrollo de estas aplicaciones dadas las dificultades que este lenguaje presenta al momento de trabajar con los puertos de la computadora.

Se puede decir que en el ámbito nacional en el área de la automatización y el control industrial se pueden encontrar pocas aplicaciones web elaboradas con el lenguaje de programación JAVA, que realicen el control de motores de corriente continua convencionales. Esa es una falencia generalizada de la industria colombiana toda vez que se consume e importa la tecnología proveniente de países industrializados y se produce muy poca tecnología vital para la competitividad del país, lo cual implica elevados costos de producción para las empresas nacionales.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar una aplicación que permita el control de velocidad y giro de motores de corriente continua convencionales a través del puerto USB.

2.2 OBJETIVOS ESPECÍCOS

- Desarrollar una aplicación web que permita el control de motores de corriente continua convencionales.

- Diseñar e implementar el hardware necesario, desde la simulación empleando CAD hasta el montaje del circuito impreso.

- Desarrollar el firmware necesario para la interfaz física.

3. JUSTIFICACIÓN

Debido al cambio en la tecnología de las interfaces en los computadores y con el auge del puerto USB el cual se ha convertido en el estándar de la industria de los sistemas computacionales, dejando en la obsolescencia los puertos seriales (RS232) y el puerto paralelo ; y en vista de que la mayoría de las aplicaciones desarrolladas para el control de sistemas electrónicos embebidos, han sido desarrollados para la utilización de los antiguos puertos anteriormente señalados; se hace necesario plantear el desarrollo de aplicaciones que se puedan comunicar y enviar señales a través de la nueva interfaz USB.

Una de las características más importantes es la velocidad de transferencia de los datos, lo cual hace que el puerto USB de la computadora sea ideal para la comunicación en el control de componentes electrónicos, los cuales requieren tiempos de transferencia de la información muy cortos con el fin de brindar un mayor control de los mecanismos actuadores, que para el caso en particular son los motores de corriente continua, los cuales responden a pulsos eléctricos que son enviados desde la computadora y controlados por medio de una aplicación desarrollada en algún lenguaje de programación; por lo tanto la transferencia de la información requiere rapidez para que el tiempo de respuesta sea menor.

El puerto USB al convertirse en el estándar de la industria de la computación obliga a que los desarrollos tecnológicos que impliquen una comunicación entre la computadora y dispositivos periféricos, con el fin de establecer un control en tiempo real; sean desarrollados teniendo en cuenta la utilización de dicho puerto, es así como surge la necesidad de desarrollar aplicaciones que cumplan con estos requerimientos.

4. MARCO TEÓRICO

4.1 ANTECEDENTES:

Desde el desarrollo del puerto Universal Serial Bus (USB) en la década de 1990 la industria computacional y el diseño e implementación de dispositivos electrónicos controlados por computadora han adoptado este estándar de la industria. Es así como prácticamente todos los dispositivos periféricos que hacen parte de un sistema computacional y prácticamente cualquier dispositivo electrónico; en la actualidad han incorporado en sus diseños he implementado esta nueva tecnología.

4.1.1 SITUACIÓN MUNDIAL:

La tecnología USB ha sido diseñada bajo el concepto de estandarización de componentes periféricos computacionales tales como teclados, mouse, impresoras, joysticks, escáneres, cámaras digitales, módems, tarjetas de red, disco duros externos, unidad de CD/DVD externas; este cambio tecnológico genero un cambio en el diseño de estos dispositivos permitiendo a su vez que tecnologías anteriores a la USB quedaran obsoletas, es así como los antiguos puertos serial, paralelo, PS/2 y los distintos dispositivos que funcionaban con estas tecnologías, fueron quedando en el olvido.

Esta nueva tecnología se ha extendido prácticamente a cualquier dispositivo electrónico como los radios de los automóviles que se han convertido en reproductores de dispositivos USB, minicomponentes, televisores, dispositivos de comunicación como los celulares modernos, tablets, PDAS y video juegos. En la actualidad para el diseño de dispositivos electrónicos es necesario tener en cuenta su conectividad con otros dispositivos y los medios de comunicación empleados, que para este caso es la tecnología de los puertos USB. Se dice que desde el año 2004 existen unos 6 mil millones de dispositivos electrónicos que usan esta tecnología y que cada año se venden alrededor de 2 mil millones de aparatos electrónicos con puertos USB a nivel mundial.

Una de las principales características de esta tecnología es su velocidad de transferencia de información que paso de los 480 Mbit/s a los 4,8 Gbit/s (600 MB/s) lo que supone un aumento de 10 veces en su velocidad; igualmente este puerto tiene la capacidad de detectar la inactividad de algún dispositivo, generando así un bajo consumo de energía. A su vez la intensidad de la corriente ha sido incrementa y paso de 500 a 900 miliamperios lo cual permite por ejemplo: alimentar un teléfono móvil o un reproductor MP3 o MP4, en un menor tiempo. La transmisión de datos de este puerto paso de 3 líneas a 5 lo cual permite el aumento de la velocidad en la transferencia de información ya que utiliza dos líneas para enviar y dos líneas para recibir datos, y una quinta línea que se encargar de suministrar la corriente, generando así un trafico bidireccional (Full Dúplex).

En marzo del año 1998 apareció el primer computador que incluyo un puerto USB de forma estándar y fue el IMac de Apple, cuyo puerto fue utilizado inicialmente para conectar tanto el mouse como el teclado. El uso de este puerto se empezó a generalizar cuando la compañía Microsoft introdujo los controladores necesarios para el funcionamiento de este dispositivo en la versión OSR 2.1 de Windows 95, esto para los computadores personales o de escritorio y en el área de los servidores los controladores USB fueron incorporados en Windows 2000.

Entre los primeros dispositivos diferentes del mouse y el teclado que ya funcionaban con esta tecnología, se encuentran las cámaras de video-conferencia, a partir del año 2005 este tipo de puerto se puede encontrar en prácticamente cualquier dispositivo electrónico.

En la actualidad han tomado un gran auge en el campo de la instrumentación virtual que utiliza una computadora como instrumento de medición y control, por medio de una tarjeta electrónica que sirve como interfaz, la cual va conectada al puerto USB; anteriormente se usaba mucho las tarjetas que iban conectadas al bus de la computadora lo que las hacia muy costosas y difíciles de programar, actualmente y con el uso de la tecnología USB estos dispositivos se han vuelto dispositivos económicos, de fácil conexión y fácil programación.

Estos dispositivos electrónicos permiten la medición y el control de magnitudes físicas como lo son, la corriente, temperatura, humedad, fuerza, distancia, presión, rotación y desplazamiento, con la utilización de sensores que permiten medir las diferentes magnitudes, o actuadores como los motores de corriente continua tanto convencionales como los paso a paso y los servomotores.

Muchos dispositivos utilizan los microcontroladores PIC de la empresa de Microchip ya que muchos de estos dispositivos incorporan registros que permiten el manejo del protocolo de comunicación del puerto USB.

4.1.2 SITUACIÓN NACIONAL:

En Colombia el desarrollo de dispositivos electrónicos que incorporan la tecnología de los puertos USB ha sido impulsado fundamentalmente por la academia, que desde las universidades alientan el espíritu emprendedor de los estudiantes de Ingeniería y ciencias de la computación en este caso.

Colciencias por medio del Instituto para el Desarrollo de la Ciencia y la Tecnología “Francisco José de Caldas” impulsa el desarrollo de nuevas ideas de proyectos, que pretenden fortalecer las capacidades tecnológicas y de gestión de incubadoras de Empresas de Base Tecnológica (IEBT).

CHAVEZ TREJO, Ana María. Ambiente de Telepresencia en la WEB para la realización de prácticas de laboratorio con el prototipo didáctico mecatrónico RefriLAB [en línea]. Instituto Tecnológico de Orizaba [citado Noviembre 26 y 27 de 2009]. Disponible en Internet: http://www.mecamex.net/anterior/cong08/articulos/45.pdf

Este tipo de instituciones buscan la generación de empresas que se conviertan en articuladoras fundamentales en el marco de la estrategia de innovación y desarrollo tecnológico, mediante la generación y transferencia de tecnologías que contribuyan a crear empresas competitivas para la búsqueda de nuevos mercados tanto internos como externos. Es allí que el desarrollo de dispositivo que incorporan la tecnología del puerto USB se encuentra en etapa de incubación como idea de negocio en nuestro país, por lo cual no encontramos que en Colombia se desarrolle este tipo de tecnologías como país productor, sino mas bien como país consumidor de estos desarrollos tecnológicos.

En 2006 en la Pontificia Universidad Javeriana de la ciudad de Cali se presenta un proyecto que utiliza un guante que va conectado al puerto USB de la computadora el cual sirve para controlar una interfaz grafica 3D desarrollada bajo el lenguaje de programación JAVA.

En ese mismo año en Medellín en el grupo de Investigación en Control Automático y Robótica ICARO del Politécnico Colombiano Jaime Isaza Cadavid, Se presenta el desarrollo de dos sistemas para la tele-operación de robots desde Internet. El primer sistema fue implementado en Java y comprende un panel de monitoreo y control desde donde se comandan los movimientos del robot y se pueden monitorear con una cámara web por medio del puerto USB. El segundo sistema fue implementado en LabVIEW y permite enviar comandos de control de velocidad, dirección y monitorear variables como orientación del vehículo, velocidad de los motores, consumo de corriente, estado de las baterías etc.

4.2 ESTRUCTURA DEL SISTEMA

El proyecto está estructurado en 3 etapas que son:

- Aplicación Web.
- Tarjeta electrónica y Firmware.
- Actuadores.

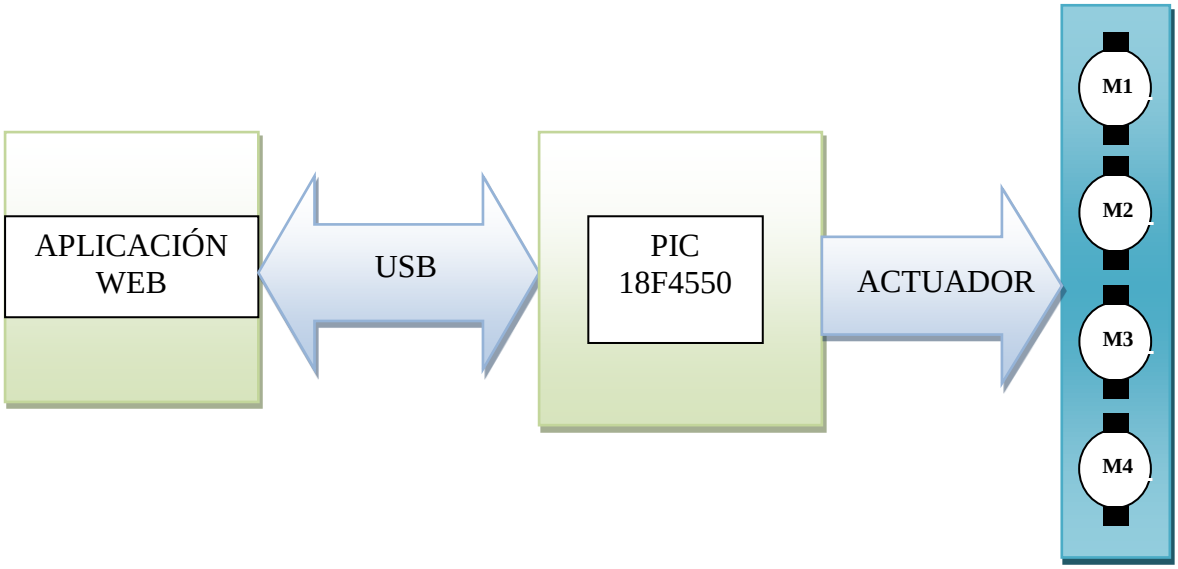
4.2.1 APLICACIÓN WEB: Es el software desarrollado en lenguaje JAVA, el cual se ejecuta desde un entorno Web y se encarga de enviar las señales enviadas por el usuario por medio de una interfaz gráfica; desde la computadora a través del puerto USB, hasta la tarjeta electrónica, que finalmente se encarga de controlar lo motores de corriente continua convencionales.

4.2.2 TARJETA ELECTRONICA Y FIRMWARE: Es el componente en donde se encuentran todos los dispositivos electrónicos necesarios para el control de los motores de corriente continua, aquí se encuentra el microcontrolador PIC 18F4550 de Microchip el cual se encarga de conectarse a la computadora y recibir las señales provenientes de esta, para luego enviar las ordenes de control a los motores CC.

El Firmware es el software que se encuentra embebido en el microcontrolador, este firmware se escribe con el lenguaje de programación C; el cual contiene las instrucciones necesarias para que le microcontrolador pueda recibir las señales he interpretarlas, para luego transmitirlas a los diferentes puertos que se encargan de enviar las señales a los motores CC.

4.2.3 ACTUADORES: Los actuadores son motores de corriente continua convencionales; estos dispositivos se encargan de recibir las señales eléctricas provenientes del microcontrolador mediante un circuito denominado puente H, el cual permite que un motor eléctrico de corriente continua gire en ambos sentidos denominados avance y retroceso, y convertirlas en energía mecánica la cual se manifiesta en un movimiento rotatorio a diferentes velocidades.

Figura 1. Estructura del sistema.



4.3 SISTEMA DE TRANSMISIÓN DE DATOS PUERTO USB

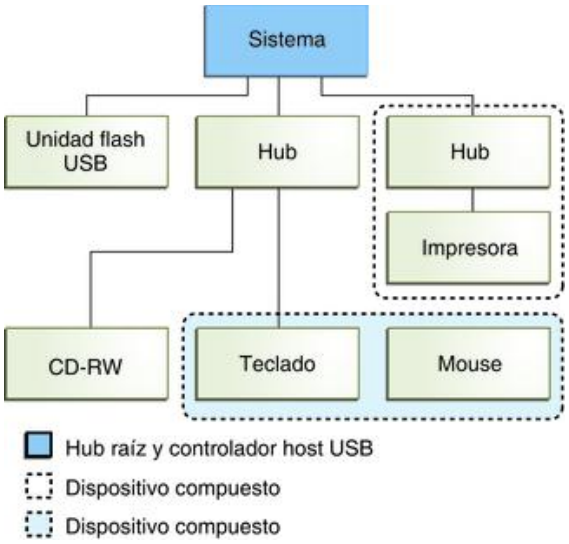
El puerto USB se define como bus universal en serie el cual tiene la característica de ser conectado a la computadora sin necesidad de reiniciarla, estos dispositivos se configuran de manera automática con solo conectarse físicamente a la computadora. Este tipo de puertos hacen parte de un sistema de comunicación entre dispositivos electrónicos informáticos que solamente transmiten una unidad de información (bit) a la vez.

Este bus puede trabajar de dos modos distintos, uno es a baja velocidad y el otro es a alta velocidad. En el modo de baja velocidad estos dispositivos funcionan con una velocidad de transferencia de información que esta en el rango de 1,5 Mbps (Mega Bits por Segundo); algunos de los dispositivos que trabajan a estas velocidades son: mouse, teclado, los cuales no manejan gran cantidad de información. A alta velocidad los dispositivos que se conectan por el puerto USB trabajan con una velocidad de transferencia de la información que está en 12 Mbps, entre los cuales se encuentran dispositivos que tienen alto manejo de la información como CDROM, altavoces, módem, tarjetas electrónicas para la transmisión y la adquisición de datos etc. Este tipo de

puerto solo dispone de 4 hilos, dos se utilizan para la transmisión de datos y dos para la alimentación. De acuerdo a estas características una de las principales ventajas es en cuanto al diseño simplificado que presentan estos puertos lo que supone un ahorro en espacio y en material.

El USB permite la conexión de múltiples dispositivos los cuales van organizados bajo una estructura de árbol descendente que van conectados a un mismo bus de la computadora. Por lo tanto en un solo controlador es posible conectar hasta 127 dispositivos, además todos estos componentes son configurados por software desde el controlador del puerto, lo que nos lleva a la tecnología conocida como plug and play.

Figura 2. Estructura de capas del puerto USB. [Citado en 2015-04-04] Disponible en <http://docs.oracle.com/cd/E26921_01/html/E25879/figures/usb_dev_hie.png>



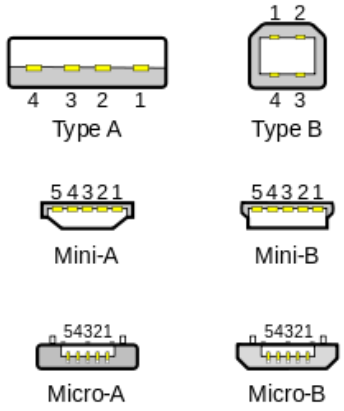
4.3.1 INTERFASE FISICA:

La interfaz de los puertos USB (Bus Universal en Serie) por lo general se identifican con el siguiente símbolo:

Figura 3. Símbolo puerto USB. [Citado en 2015-04-04] Disponible en <http://img.xataka.com/2008/03/img_3160_usb-logo.jpg>

En la siguiente figura se pueden identificar los diferentes tipos de conectores USB y sus respectivos pines.

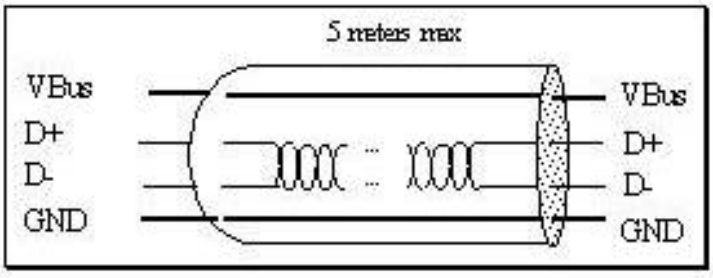
Figura 4. Tipos de conectores USB. [Citado en 2015-04-04] Disponible en <http://upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Types-usb_new.svg/271px-Types-usb_new.svg.png>



Pin 1	V_{CC} (+5 V)
Pin 2	Data-
Pin 3	Data+
Pin 4	Tierra

Por medio de los pines son transmitidas las señales que transportan la información así como también la energía eléctrica para lo cual se dispone de 4 cables. Las señales de transmisión de datos se mueven por medio de los segmentos entre los dos dispositivos USB con un rango de velocidad que oscila entre 1.5 Mbps y 12Mbps, para las transmisiones tanto de baja como de alta velocidad. Los otros dos cables son utilizados para proporcionar energía a los dispositivos electrónicos con un potencial eléctrico de 5V positivos que son suministrados por el pin VBus, esto le permite a los cables USB manejar distancias que van desde algunos centímetros hasta los 5 metros.

Figura 5. Disposición de pines en el puerto USB. [Citado en 2015-04-04] Disponible en <http://regmedia.co.uk/2009/05/22/usb_3_9.png>



4.3.2 PROTOCOLO DEL BUS:

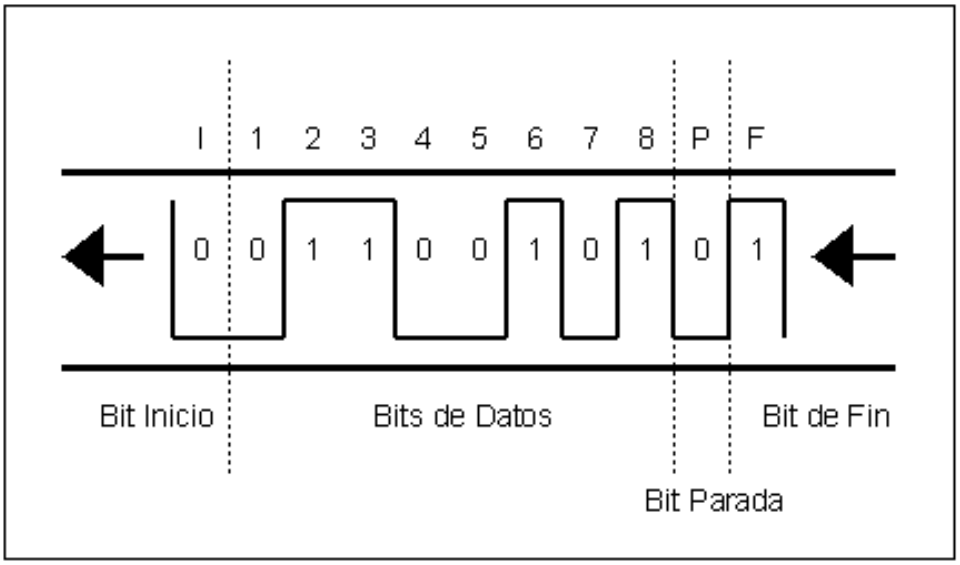
En toda transferencia de datos o transacción en donde se utilice el bus como medio de transmisión, están involucrados tres paquetes de datos. Una transacción es posible cuando el controlador del puerto decide cual dispositivo esta listo para usar el bus, para lo cual se envía un paquete al dispositivo correspondiente. Estos paquetes poseen un número único de identificación, el cual es proporcionado a través del controlador en el momento en el cual es encendida la computadora o cuando se conecta un nuevo dispositivo al puerto USB. Todo esto le permite a los distintos dispositivos conectados al puerto, saber cual paquete de datos le corresponde específicamente. El nombre que se le da a este paquete es el Token Packet, tan pronto el periférico correspondiente recibe el permiso de transmitir, empieza la comunicación entre los dispositivos hasta que finaliza la transmisión de datos, momento en el cual el controlador envía el paquete que indica el fin de la transmisión y continua con el siguiente dispositivo. Este protocolo cuenta con un sistema de recuperación de errores que se llama CRC (Código de Redundancia Cíclica).

4.3.3 TRANSMISION ASINCRONICA:

Cuando se trata de la transmisión en serie de los datos, la información que es generada por el transmisor debe ser recuperada de la misma forma por el receptor, esto implica la necesidad de generar un sincronismo entre ambos extremos de la comunicación (Emisor – Receptor). Esto es posible mediante el uso de relojes los cuales debe funcionar a la misma frecuencia tanto en el dispositivo emisor como en el receptor lo cual hace posible una comunicación exitosa. Derivado de este concepto surge la transmisión de datos asincrónica utilizada en la comunicación entre equipos servidores o host y sus terminales.

En este tipo de comunicación, cuando un dispositivo quiere transmitir información produce un grupo de bits en donde el bit número 1 es conocido como el bit de arranque y un conjunto de 7 u 8 bits de datos, uno o dos bits de paridad los cuales se encargan del manejo de errores y por ultimo uno o dos bits de parada.

Figura 6. Esquema de transmisión asincrónica. [Citado en 2015-04-04]
Disponible en <http://html.rincondelvago.com/000732533.jpg>



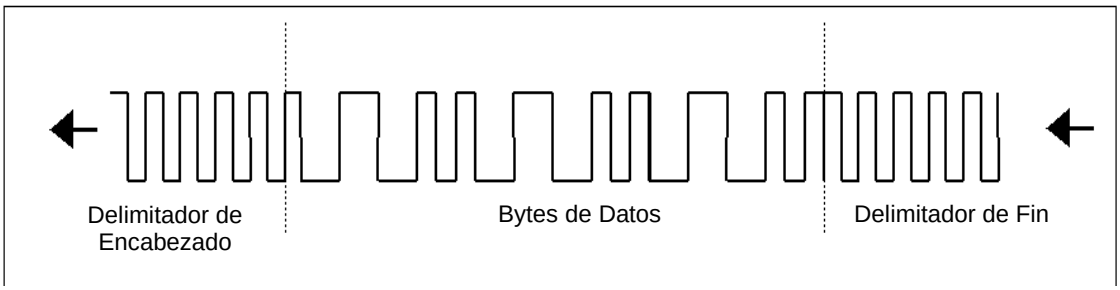
Se le denomina a este tipo de transmisión asincrónica no por el hecho de que no exista ningún tipo de sincronismo, sino por que el sincronismo no lo podemos hallar en la señal, son los relojes que trabajando a la misma frecuencia generan el sincronismo necesario para efectuar la comunicación y la transmisión de datos; hay que entender que el concepto de sincronía depende de la señal y no de los equipos de transmisión y recepción de datos.

4.3.4 TRANSMISION SINCRONICA:

En este tipo de transmisión de datos el sincronismo viaja dentro de la misma señal que es transmitida desde el transmisor hacia el receptor lo cual permite el mejor aprovechamiento del canal de comunicación, así como también el poder alcanzar una mayor distancia. Haciendo un paralelo entre la comunicación sincrónica y la asincrónica podemos decir que, al tratarse de transmitir información de manera asincrónica los grupos de datos se componen generalmente por 10 bit de los cuales 4 son utilizados para el control. Mientras que en la transmisión de datos sincrónica los grupos de datos están compuestos por 128 Bytes o más, dependiendo del canal por donde se transmita la información.

TRADE PRESS RELATIONS & TRANSLATION SERVICES. Nueva tecnología de mando por Ethernet para accionamientos y sistemas [en línea]. TPR International [citado marzo de 2011].
Disponible en Internet: http://www.tradepressrelations.com/en/file/presse/es/presse1346.php

Figura 7. Esquema de transmisión sincrónica. [Citado en 2015-04-04] Disponible en <http://html.rincondelvago.com/000732534.jpg>

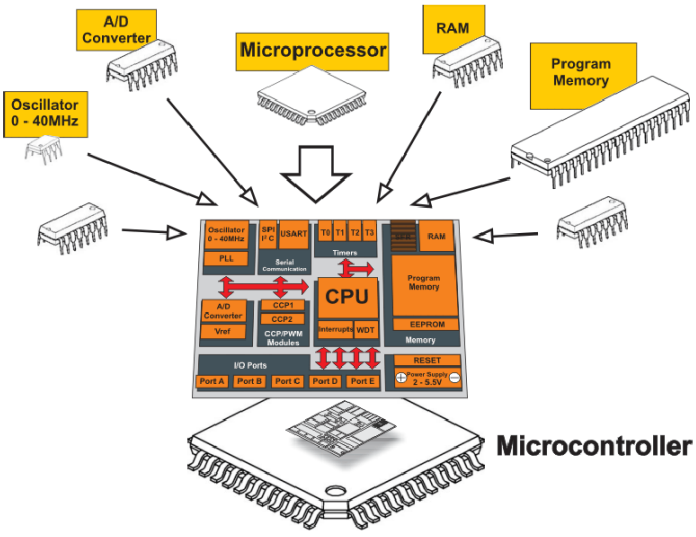


4.4 MICROCONTROLADOR

Los microcontroladores son dispositivos que se han popularizado en el mundo entero ya que los podemos encontrar en una gran cantidad de aparatos electrónicos; los encontramos controlando mouse, teclados de computadoras, teléfonos, hornos microondas, televisores etc. Estos componentes son utilizados ampliamente para el control de uno o varios procesos, esta función y el hecho de que son circuitos integrados le dan el nombre que los identifica como microcontroladores.

Un microcontrolador es un dispositivo electrónico el cual puede ser programable con el objetivo de efectuar una tarea específica. Estos dispositivos internamente están compuestos por una serie de elementos que hacen posible que estos actúen como pequeñas computadoras programables, la cantidad de estos componentes varían dependiendo de los fabricantes; de todas formas los componentes básicos de un microcontrolador son: Microprocesador, memoria RAM, memoria de programa, convertidor A/D, oscilador, puerto de comunicación entre otros. Estos dispositivos poseen una gran versatilidad lo que ha generado su popularidad y ha incrementado su utilización en prácticamente cualquier dispositivo electrónico.

Figura 8. Componentes de los microcontroladores. [Citado en 2015-04-04] Disponible en <http://www.mikroe.com/img/publication/spa/pic-books/programming-in-c/chapter/01/fig0-1.gif>



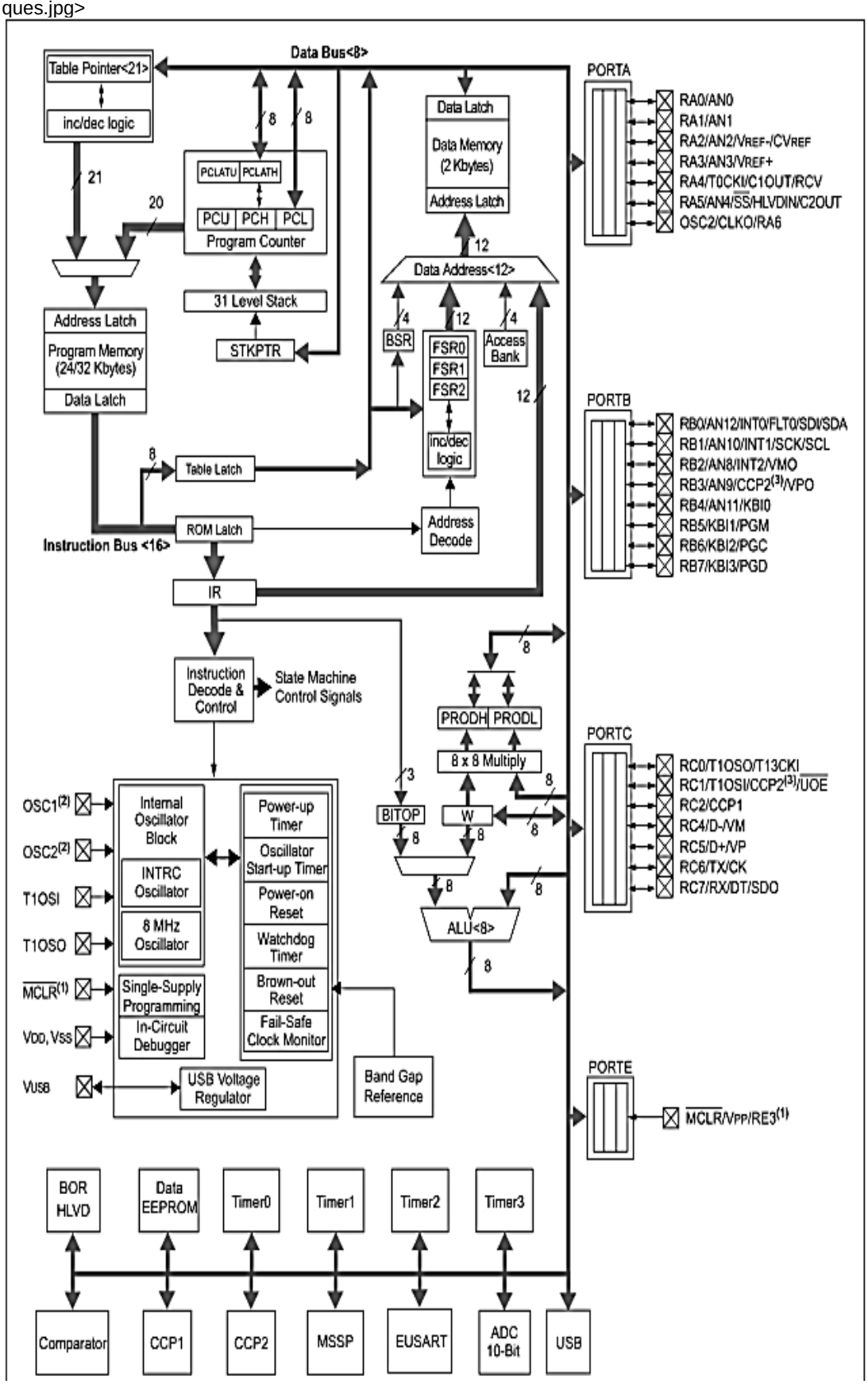
GARCIA, Eduardo. Compilador C CCS y Simulador Proteus para Microcontroladores PIC. Primer Edición, México: AlfaOmega grupo editor, S.A, junio de 2008.

Al momento de trabajar con microcontroladores se hace necesario tener cierto tipo de consideraciones como: La cantidad de entradas y salidas que pueda tener el dispositivo, si se necesita o no de un convertidor A/D, si se requiere algún modulo especializado de comunicación etc. Teniendo en cuenta los requerimientos necesarios para el desarrollo de una aplicación especifica, se puede elegir dentro de una gran variedad de microcontroladores, el adecuado para realizar el trabajo propuesto; existen muchas marcas y fabricantes de microcontroladores entre los cuales podemos destacar los siguientes: AVR de Atmel, Freescale (antes Motorola), Basic Stamp de Parallax y PICmicro de Microchip. En este caso nos ocuparemos del microcontrolador PICmicro 18F4550 por ser unos de los microcontroladores mas populares en el mercado el cual ha sido desarrollado por la empresa Microchip.
La popularidad de estos microcontroladores radica en el hecho de que los fabricantes tienen una política de ofrecer documentación y todo el software necesario sin ningún costo para el usuario, lo que hace de estos dispositivos ideales para su uso por parte de principiantes, aficionados y profesionales.

4.4.1 ARQUITECTURA INTERNA:

Los microcontroladores PIC están construidos bajo la arquitectura Harvard, lo que implica el uso de memorias independientes, una de las cuales se utiliza para el programa y la otra para el almacenamiento de los datos, cada una de estas memorias poseen sus respectivos buses. Esto le permite a estos microcontroladores el acceso y uso de simultáneo a ambas memorias, lo que implica mejorar considerablemente el rendimiento de estos dispositivos. Adicionalmente estos cuentan con la tecnología RISC, lo que hace que el manejo de instrucciones sea reducido, en donde solamente las instrucciones de carga y almacenamiento tienen acceso a la memoria de datos, todo esto se hace con el fin de posibilitar la segmentación y el paralelismo en la ejecución de las instrucciones, favoreciendo con ello la reducción en los accesos a memoria.

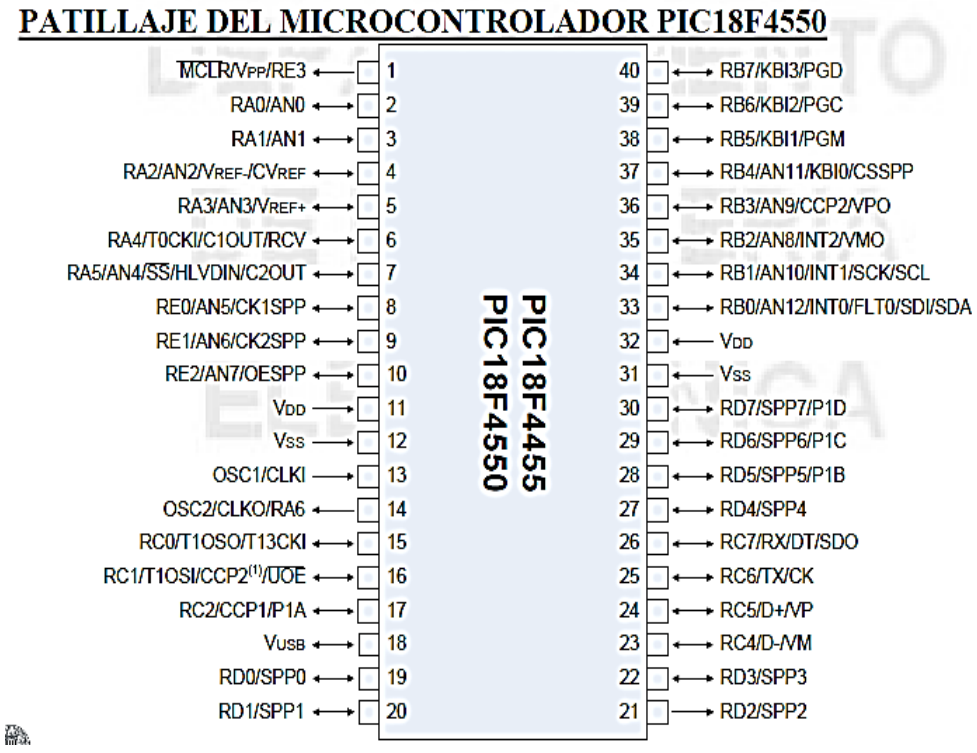
Figura 9. Diagrama de bloques del PIC 18F4550. [Citado en 2015-04-04] Disponible en <http://www.muchotrasto.com/images/Tutoriales/RobotMovil/SistemaElectronico/DiagramaDeBloques.jpg>



4.4.2 HOJA DE DATOS DEL PIC18F4550:

Esta familia de microcontroladores están diseñados en diferentes gamas que van desde 8-bit, 16-bit y 32-bit. Dentro de la gama más simple de 8-bit se puede encontrar el microcontrolador PIC18F4550 el cual pertenece a la familia PIC18 MCU. Este tipo de microcontroladores es muy usado en diferentes tipos de aplicaciones ya que posee ciertas características de memoria de programa, memoria RAM, número de entradas y salidas, número de canales analógicos y tipos de puertos de comunicación.

Figura 10. Esquema de pines del PIC 18F4550. [Citado en 2015-04-04] Disponible en <http://mlv-s2-p.mlstatic.com/pic-18f4550-microcontrolador-pic18f4550-ip-1814-MLV2705355772_052012-O.jpg>



CLAVIJO MENDOZA, Juan Ricardo. Diseño y Simulación de sistemas Microcontrolados en Lenguaje C. Primera edición, Colombia: ISBN 978-958-44-8619-6, Mayo de 2011.

Tabla 1. Características de la familia PIC18. [Citado en 2015-04-04] Disponible en <http://4.bp.blogspot.com/-lgFxQY_uPHo/UNz55c1l7WI/AAAAAAAABx4/MzSyFn8QPMM/s320/tabla1.jpg>

MICROCONTROLADORES PIC18F2455, PIC18F2550, PIC18F4455 y PIC18F4550

CARACTERISTICAS	PIC18F2455	PIC18F2450	PIC18F4455	PIC18F4450
Frecuencia de Operación	Hasta 48MHz	Hasta 48MHz	Hasta 48MHz	Hasta 48MHz
Memoria de Programa (bytes)	24.576	32.768	24.576	32.768
Memoria RAM de Datos (bytes)	2.048	2.048	2.048	2.048
Memoria EEPROM Datos (bytes)	256	256	256	256
Interrupciones	19	19	20	20
Líneas de E/S	24	24	35	35
Temporizadores	4	4	4	4
Módulos de Comparación/Captura/PWM (CCP)	2	2	1	1
Módulos de Comparación/Captura/PWM mejorado (ECCP)	0	0	1	1
Canales de Comunicación Serie	MSSP,EUSART	MSSP,EUSART	MSSP,EUSART	MSSP,EUSART
Canal USB	1	1	1	1
Puerto Paralelo de Transmisión de Datos (SPP)	0	0	1	1
Canales de Conversión A/D de 10 bits	10 Canales	10 Canales	13 Canales	13 Canales
Comparadores analógicos	2	2	2	2
Juego de instrucciones	75 (83 ext.)	75 (83 ext.)	75 (83 ext.)	75 (83 ext.)
Encapsulados	PDIP 28 pines SOIC 28 pines	PDIP 28 pines SOIC 28 pines	PDIP 40 pines QFN 40 pines TQFP 40 pines	PDIP 40 pines QFN 40 pines TQFP 40 pines

4.5 ACTUADORES:

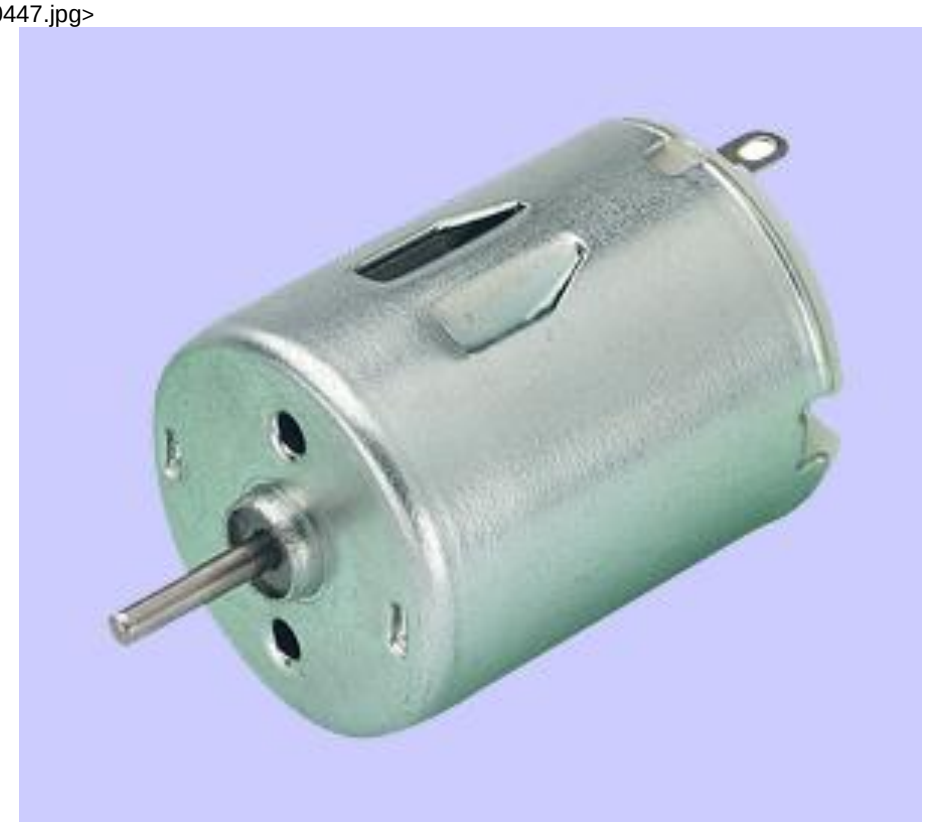
En este proyecto se utiliza como dispositivo actuador, motores de corriente continua convencionales ya que son máquinas que tienen la capacidad de convertir energía eléctrica continua en energía mecánica, estos dispositivos son muy versátiles por que son de fácil control, tanto en velocidad, par y posición, lo que los hace ideales para las aplicaciones que se usan en el control y la automatización de procesos.

Estos motores son muy utilizados en adaptaciones que impliquen potencia y precisión. Accionar un motor DC es muy simple, solo es necesario aplicar tensión de corriente continua entre sus bornes; para invertir el sentido de giro de estos motores es suficiente con invertir la polaridad de la fuente de alimentación en sus borneras e inmediatamente girara en sentido opuesto.

A diferencia de los motores paso a paso y los servomecanismos, los motores DC no pueden ser posicionados o enclavados en una posición específica. Estos simplemente giran a la máxima velocidad y en el sentido en el que la corriente de alimentación se lo permite.

GARCIA HARO, Juan Miguel. Desarrollo de un controlador para motores D.C brushless basado en CompactRIO y LabVIEW de National Instruments para el estudio de nuevos algoritmos de control [en línea]. Universidad Carlos III de Madrid [citado en Leganés, Noviembre de 2011]. Disponible en Internet: http://e-archivo.uc3m.es/bitstream/10016/13615/1/PFC_JuanMiguel_Garcia_Haro.pdf

Figura 11. Imagen de un motor DC convencional. [Citado en 2015-04-04] Disponible en <http://img.directindustry.es/images_di/photo-g/motor-corriente-continua-665-4780447.jpg>

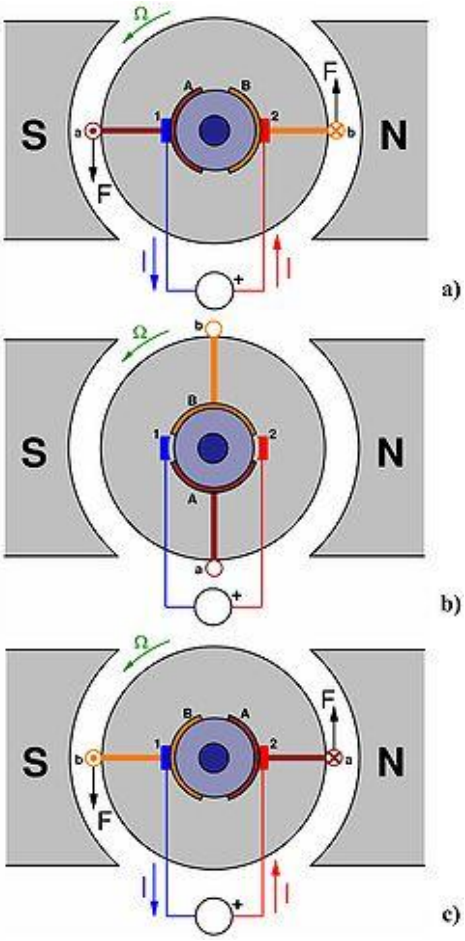


4.5.1 FUNCIONAMIENTO DE UN MOTOR DE CORRIENTE CONTINUA:

El principio de funcionamiento de un motor de corriente continua se basa en el principio de repulsión que ejercen los polos magnéticos de un imán permanente, de acuerdo con la Ley de Lorentz, interactúan con los polos magnéticos de un electroimán que se encuentra montado en un eje. Este electroimán se denomina “rotor” y su eje le permite girar libremente entre los polos magnéticos norte y sur del imán permanente situado dentro de la carcasa o cuerpo del motor “estator”.

Cuando una corriente circula a través de la bobina que hace parte del electroimán, se genera un campo electromagnético, el cual interactúa con el campo magnético del imán permanente. Si los polos del electroimán giratorio coinciden con los polos del imán permanente se produce inmediatamente un rechazo y un torque, lo que se conoce como par de fuerza, lo que provoca que el rotor rompa la inercia y empiece a girar sobre su propio eje en el sentido de las manecillas del reloj, en algunos casos o en sentido contrario, todo esto depende de la polaridad que se le aplique al momento de ser energizado.

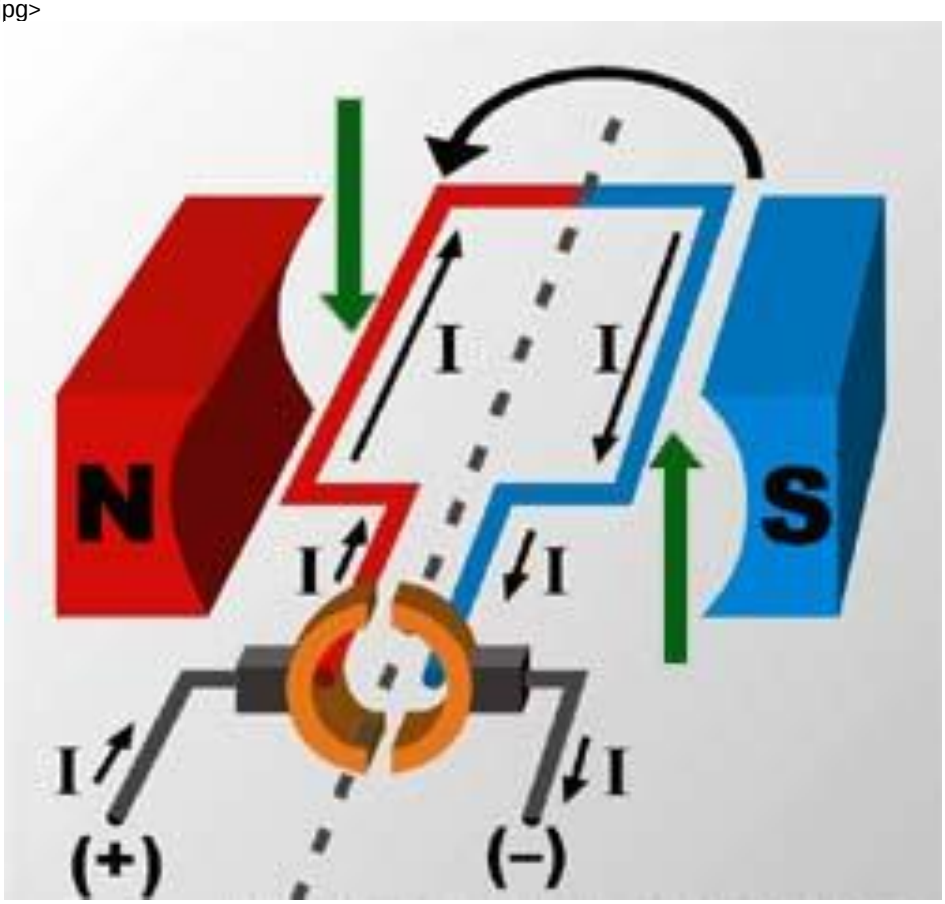
Figura 12. Imagen del principio de funcionamiento de un motor D.C.
[Citado en 2015-04-04] Disponible en <http://upload.wikimedia.org/wikipedia/commons/thumb/0/02/Principio_motor_c.c.jpg/245px-Principio_motor_c.c.jpg>



1,2 – Escobillas.
A, B – Delgas (Láminas de Cobre).
a, b – Lados de las bobinas conectados a las delgas.

En la siguiente figura se observa de forma esquemática muy simple, un motor de corriente continua, en donde se representa un rotor formado por una bobina simple de una sola espiral el cual se aprecia de color rojo y azul, que sirve para diferenciar cada mitad. Si sigue el recorrido de la corriente eléctrica (**I**) y asumiendo que esta fluye de modo convencional, del polo positivo al negativo de la fuente de corriente continua, cuando en la mitad izquierda de color roja se forma el polo norte (**N**) coincidiendo con la misma polaridad del imán permanente, inmediatamente se produce una fuerza de repulsión entre los polos iguales. Lo mismo ocurre cuando en el rotor se forma el polo sur, lo que funcionando al unísono y en combinación de estas fuerzas se genera el movimiento del rotor sobre su propio eje.

Figura 13. Esquema de un motor de corriente continua. [Citado en 2015-04-04] Disponible en <http://www.asifunciona.com/electrotecnia/af_motor_cd/img_motor_cd/img19_mot_cd_250px.jpg>

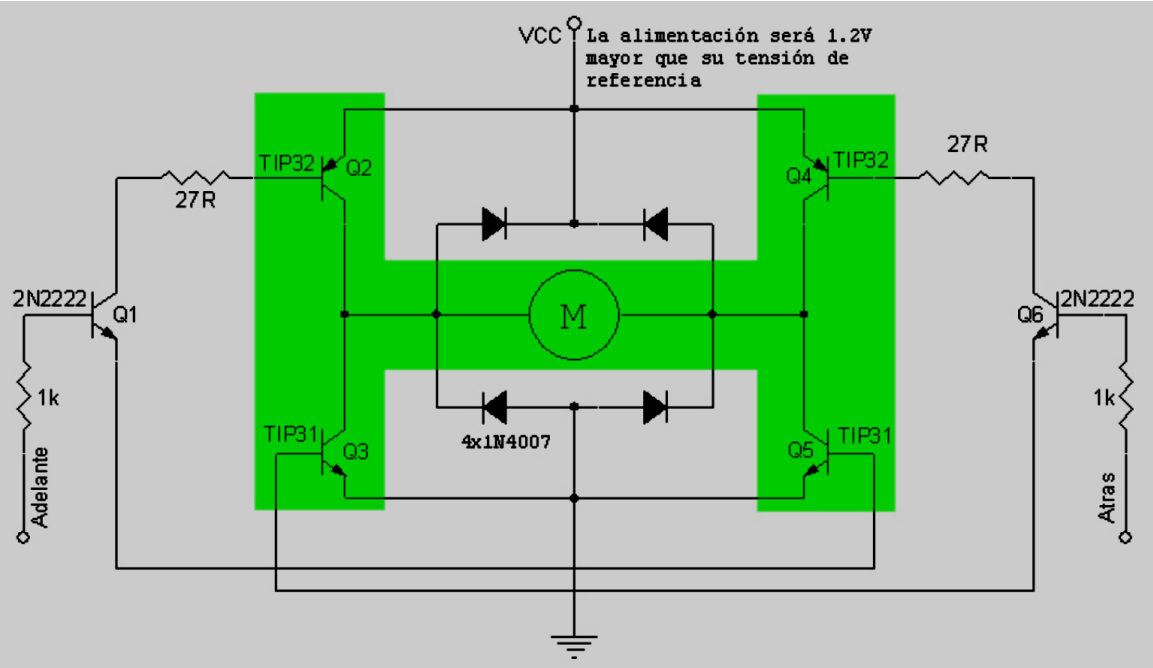


4.5.2 PUENTE H:

Es un circuito integrado constituido principalmente por transistores, los cuales están configurados de tal forma que de allí proviene el nombre de puente H. Este tipo de configuración es de las más usadas para el control de motores de corriente continua.

OVERSTREET, William. (1999). "An InternetBased Real-Time Control Engineering Laboratory", IEEE Control Systems Magazine, Vol. 19, pp. 19-34.

Figura 14. Configuración del puente H. [Citado en 2015-04-04] Disponible en <http://www.globu.net/ES/circuito_puente.JPG>



4.5.2.1 FUNCIONAMIENTO:

Si se aplica una señal positiva (1) en la entrada "Adelante" el transistor Q1 se pone en conducción saturándose. La corriente de colector de Q1 circula por la base de Q2 y la de emisor por la de Q5, lo que provoca que al terminal positivo del motor llegue VCC, debido a la saturación de Q2, y que el negativo quede conectado a tierra por la saturación de Q5.

Si, en cambio, se aplica una señal positiva en la entrada "Atrás" conducirá el transistor Q6, que cierra su corriente por las bases de Q4 y Q3. En este caso se aplica VCC al terminal negativo del motor y es el terminal positivo el queda conectado a tierra, haciendo que el motor gire en sentido contrario al anterior. Pero el puente en H no solo puede controlar el sentido de giro, también permite estrategias de frenado diferentes de la pasiva. Así, es posible frenar el motor de forma dinámica, que provoca un frenado mas rápido del motor. Esta forma de frenado, en su forma básica, consiste en forzar un frenado electromagnético mediante la creación de un cortocircuito de los terminales del motor. La forma de cortocircuitar los terminales es sencilla: Adelante=Atrás=0. Otra forma de provocar el frenado rápido del motor (muy rápido en este caso) es mediante la inversión de la tensión en sus extremos durante el tiempo necesario para producir la parada del mismo. Mientras mas potente sea el motor menos aconsejable es este sistema de frenado.

SCHMID, Chris. (1992): “Real – Time Control with CADACS-PC”. Recent Advances in computer – Aided Control Systems Engineering. M. Jamshidi and C.J. Herget (Editors), 337 – 355. North – Holland, Amsterdam, 1992.

4.5.2.2 PUENTE H INTEGRADO L298N:

El integrado L298N es un circuito integrado (C.I) monolítico el cual viene en encapsulados SO20 y Multiwatt de 15 pines. Este es un C.I que tiene la capacidad de manejar niveles altos de voltaje y corriente mediante un controlador completo compuesto por 2 puentes H, diseñados para aceptar los niveles estándar de la tecnología TTL (Transistor-Transistor logic), y controlar cargas inductivas tales como: Relés, solenoides, motores paso a paso y motores de corriente continua convencionales.

Figura 15. Tipos de encapsulado del puente H. [Citado en 2015-04-04] Disponible en <http://esdocs.org/pars_docs/refs/72/71309/71309_html_1349fa17.gif>

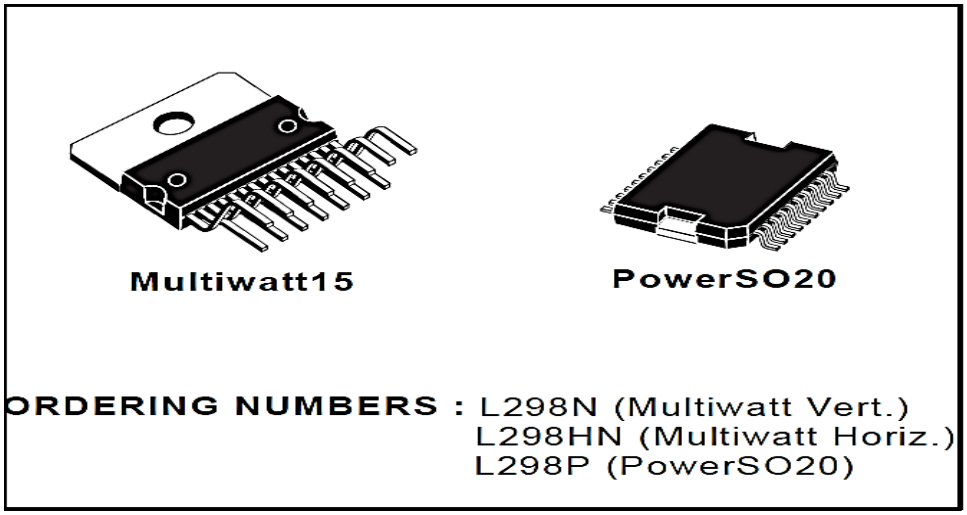
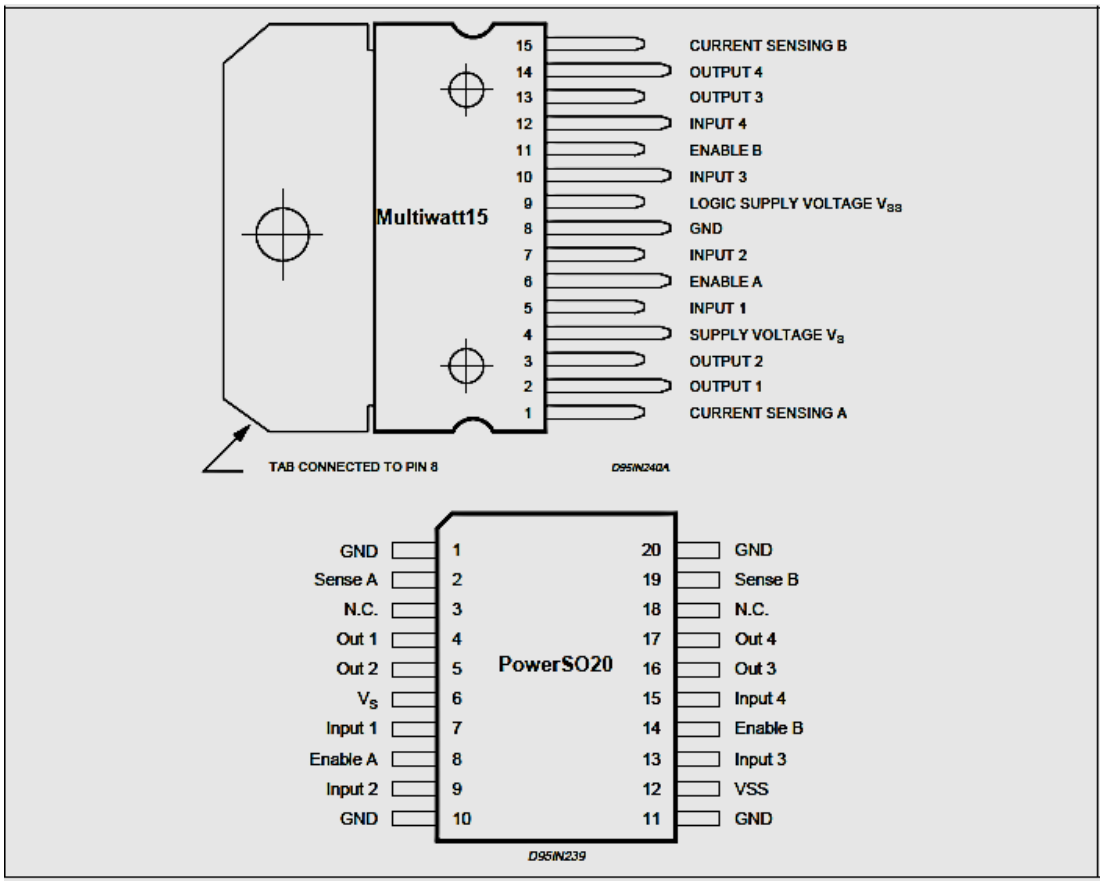


Figura 16. Configuración de los pines. [Citado en 2015-04-04] Disponible en <http://www.wvshare.com/img/pinout/L298_l.jpg>



VALERA, Andrew. (2001) “LabConRob: Virtual Laboratory of Real-Time Robot Control”, Robótica 43. ISBN 152701/00. SIN 0874-9019. Pp 40-45.

Tabla 2. Características eléctricas del puente H. [Citado en 2015-04-04] Disponible en <http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf>

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

5. METODOLOGÍA

5.1 TIPO DE TRABAJO:

El presente trabajo se enmarca en un desarrollo tecnológico, al interior del área de investigación en Inteligencia Computacional, del grupo de investigación y desarrollo en Informática y Telecomunicaciones de la Facultad de Ciencias e Ingeniería.

5.2 PROCEDIMIENTO:

El proyecto se realizará en cuatro fases, así:

5.2.1 FASE 1. ANÁLISIS: Precisar la funcionalidad y los alcances que tiene el proyecto.

- Actividad 1. Alcance. Aquí se fijan los límites y los actores que intervienen en el proyecto.

- Actividad 2. Funcionalidad. Definición de los lenguajes y herramientas que serán utilizados en la elaboración del proyecto.

5.2.3 FASE 2. DESARROLLO: Realización de proceso de definición de las diferentes herramientas que serán utilizadas en el desarrollo del proyecto.

- Actividad 1. Realización del software necesario, utilizando los lenguajes de programación seleccionados para la realización del proyecto.

- Actividad 2. Se hacen las pruebas necesarias de verificación del código fuente.

5.2.4 FASE 3. IMPLEMENTACIÓN: Se hace el montaje real del proyecto.

- Actividad 1. Pruebas. Se hace prueba de la aplicación en un ambiente simulado con el fin de determinar el correcto funcionamiento.

- Actividad 2. Demostración real. Se ejecuta la aplicación en un entorno real y se comprueba su funcionalidad.

6. RESULTADOS

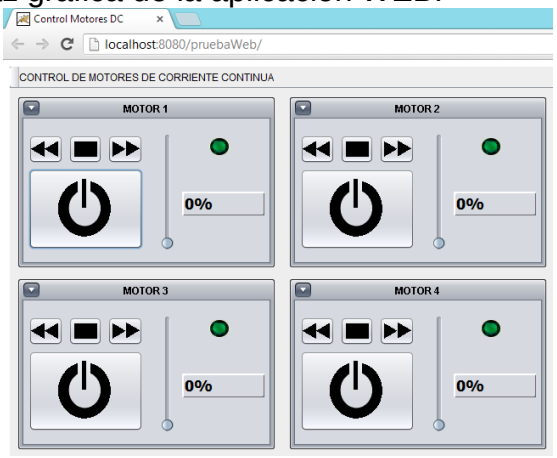
Desde la interfaz gráfica del usuario del sistema, la cual consiste en una aplicación web; el usuario puede disponer de los controles necesarios que le permitan hacer uso del mismo, el cual permite encender, apagar, controlar el sentido de giro bien sea a la derecha o a la izquierda, de igual forma se permite el control de velocidad de los motores de corriente continua convencionales, todas estas acciones son ejecutadas mediante los botones correspondientes.

Desde esta interfaz es posible controlar hasta 4 motores de corriente continua convencionales, así como también se tiene la posibilidad de visualizar el estado de encendido o apagado de cada uno de ellos mediante unos dispositivos simuladores de diodos led que indican el encendido o apagado de cada uno de los dispositivos actuadores (Motores DC), de igual forma se dispone en pantalla de indicadores porcentuales de velocidad los cuales indican el nivel de porcentaje de velocidad al cual están girando.

6.1 APLICACION WEB:

Esta es una aplicación que se ejecuta en un entorno Web, desde la cual es posible enviar los comandos necesarios utilizados en el control de los motores de corriente continua los cuales funcionan como actuadores, esta información es recibida por el sistema mediante un click de mouse sobre los respectivos botones, la aplicación interpreta estas órdenes y luego envía las señales correspondientes, por medio del puerto USB de la computadora a la tarjeta electrónica, la cual es la encargada de interpretarlas y después ejecutar las acciones sobre los puertos correspondientes del microcontrolador, para que posteriormente sean enviadas las señales hacia los motores, los cuales ejecutan la acción que se desea.

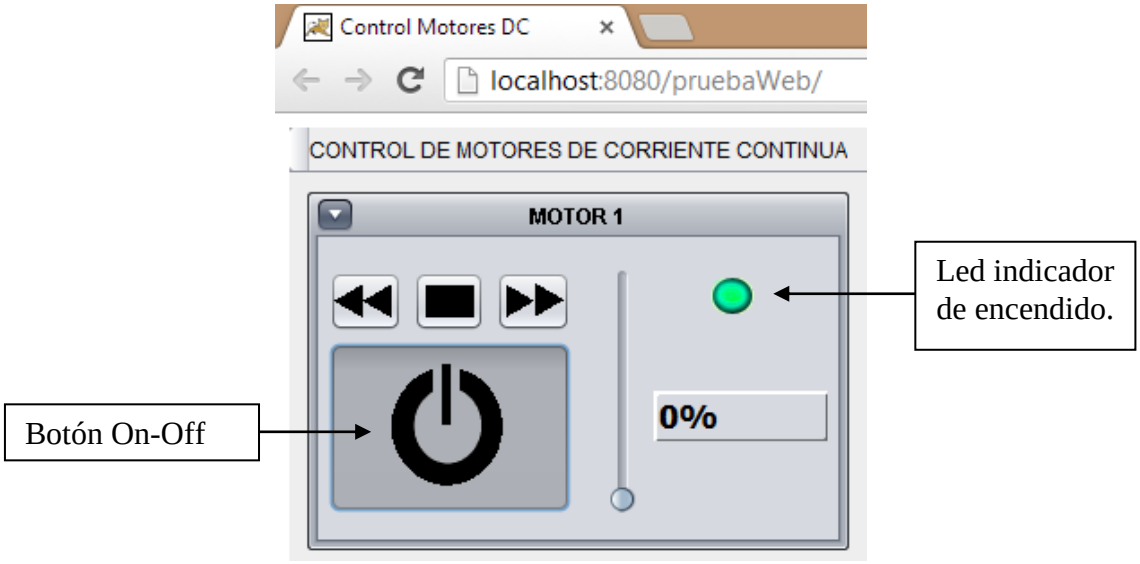
Figura 17. Interfaz gráfica de la aplicación WEB.



6.1.1 ENCENDIDO DE LOS MOTORES:

El encendido de cada uno de los motores se efectúa haciendo click sobre el botón que indica encendido o apagado como se puede ver en la siguiente figura.

Figura 18. Botón de encendido y apagado de la aplicación Web.



6.1.2 CONTROL DEL SENTIDO DE GIRO DE LOS MOTORES:

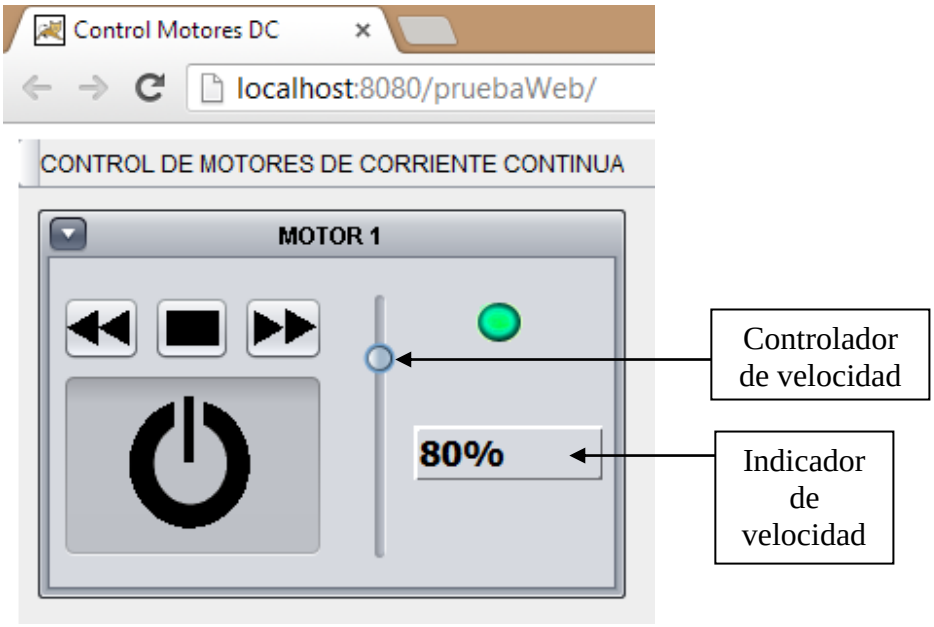
El control del sentido de giro de los motores es efectuado por medio de un click que se hace con el botón izquierdo del mouse sobre los botones respectivos así como se indica en la siguiente figura.

Figura 19. Botones de control de giro de los motores.

6.1.3 CONTROL DE VELOCIDAD DE LOS MOTORES:

La velocidad de los motores puede ser controlada desplazando el cursor de la barra de desplazamiento desde donde se indica el porcentaje de velocidad que alcanzara el motor, este porcentaje es visualizado desde el indicador de velocidad. Es de aclarar que la velocidad de los motores se indica en porcentajes y no en revoluciones.

Figura 20. Control de velocidad de los motores.



6.2 IMPLEMENTACIÓN DEL HARDWARE DEL SISTEMA:

El hardware del sistema es el medio físico por medio del cual son recibidas he interpretadas las señales que viajan por medio del puerto USB de la computadora, las cuales son enviadas por medio de la aplicación Web, este componente está construido con base en un microcontrolador PIC 18F4550.

Este microcontrolador se encarga de interpretar y ejecutar las ordenes que provienen de la computadora además de servir como enlace entre la aplicación Web y los dispositivos actuadores que en este caso son motores de corriente continua convencionales, este control se hace mediante los puertos del microcontrolador por donde viajan las señales que son recibidas por los motores, por medio de los unos dispositivos integrados que se denominan puente H los cuales se pueden encontrar en los circuitos integrados con referencia L298, tal y como se observa en la siguiente figura.

Figura 21. Esquema eléctrico de hardware del sistema.

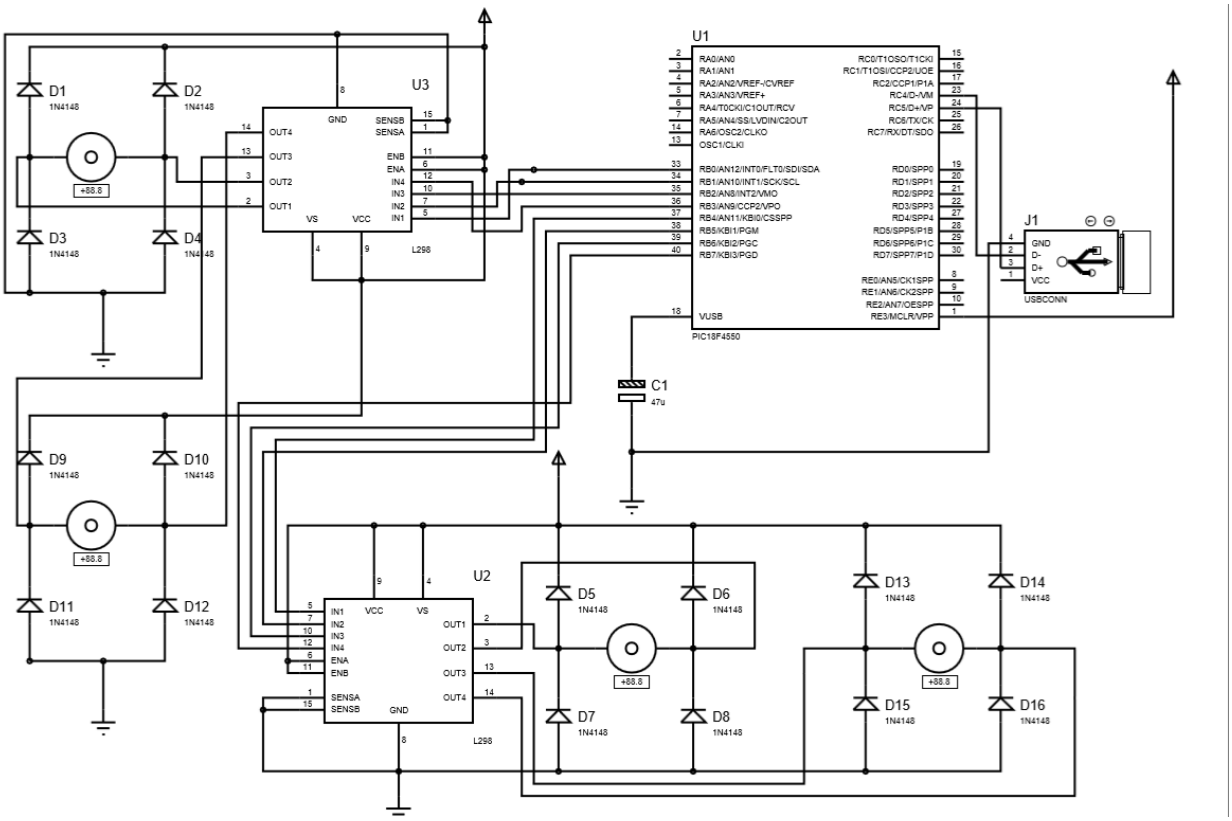


Figura 22. Modelado en 3D del hardware.

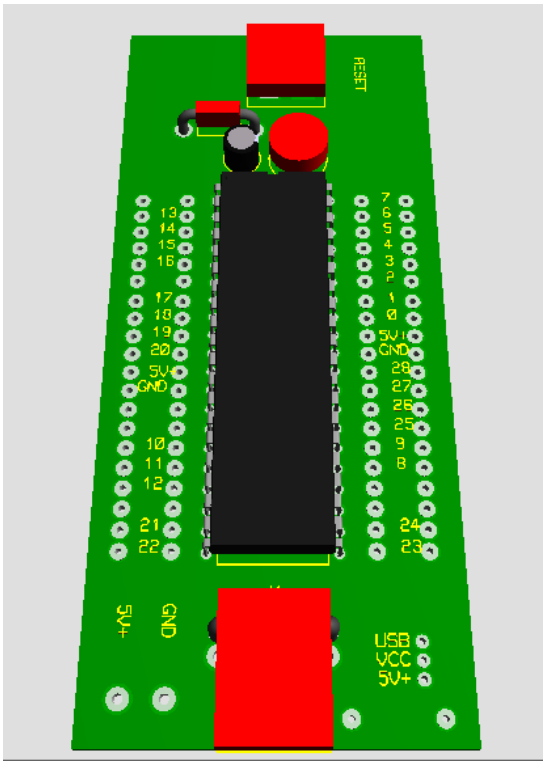


Figura 23. Modelado 3D del puente H.

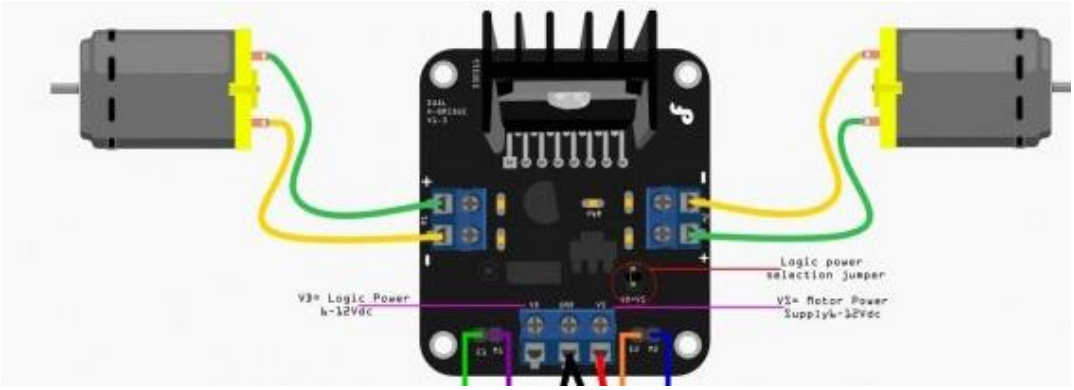
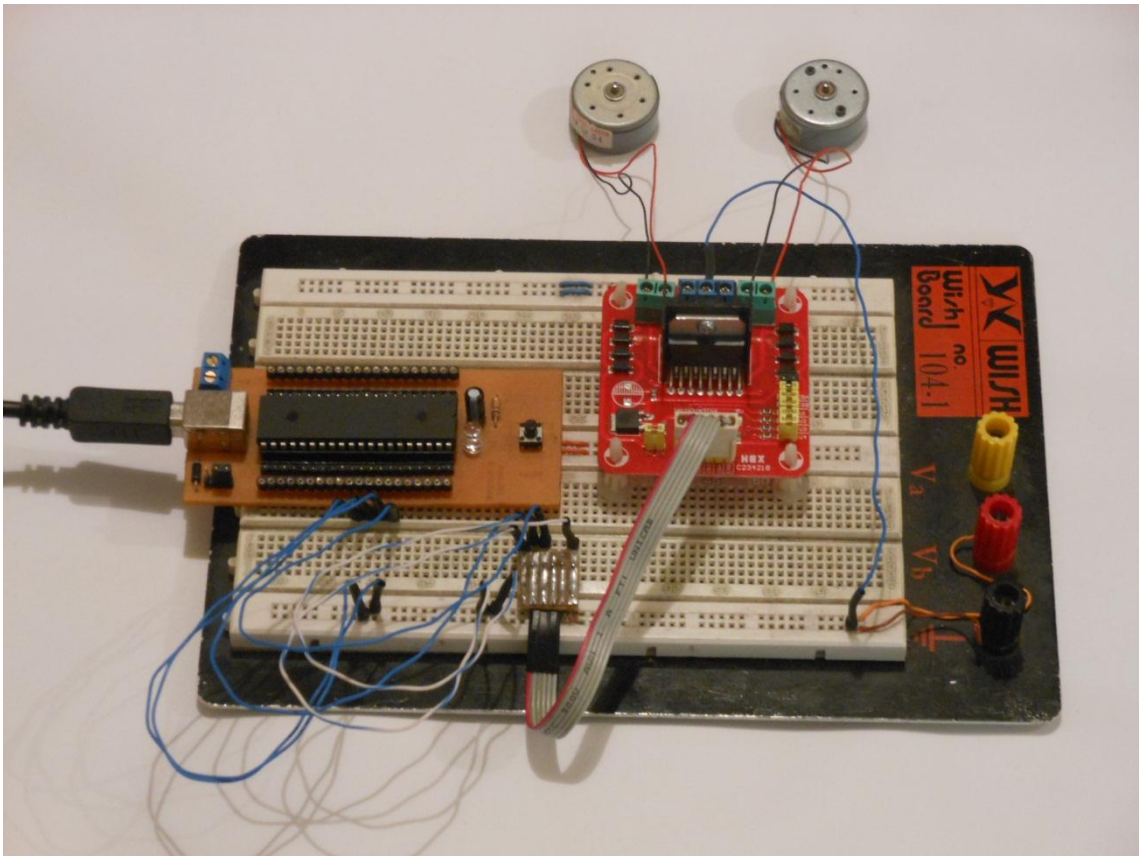


Figura 24. Hardware del sistema.

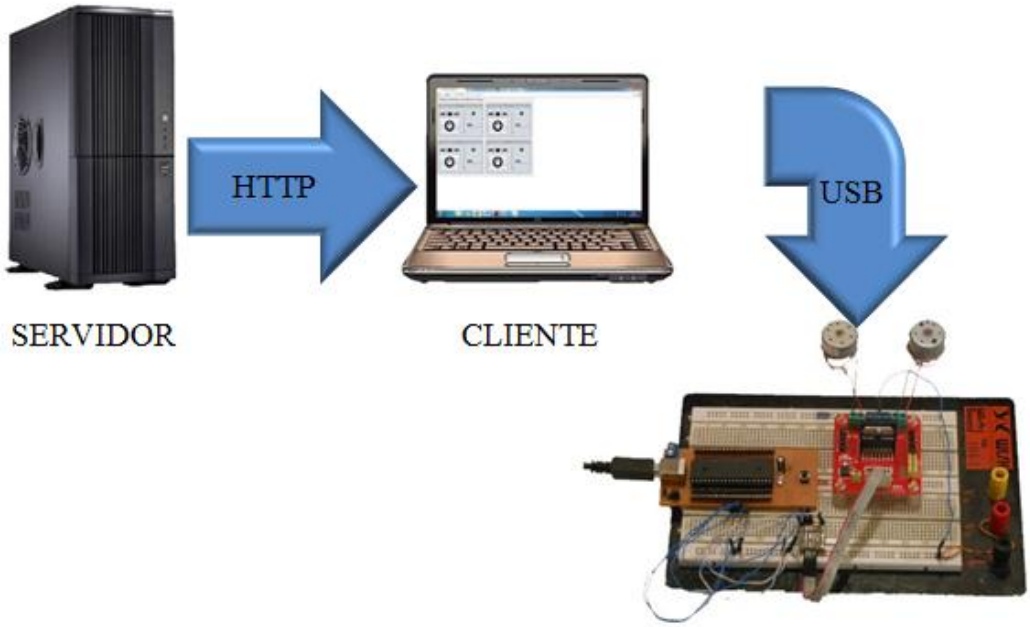


6.3 IMPLEMENTACION Y PRUEBAS:

El sistema se ejecuta dentro de un entorno web es decir, la aplicación se encuentra alojada en un servidor desde el cual se ejecutan todas las acciones y órdenes que posteriormente son enviadas por el puerto USB

para que el hardware interprete dichas instrucciones y los actuadores (Motores D.C convencionales) funcionen en consecuencia.

Figura 25. Arquitectura de la aplicación.



6.3.1 IMPLEMENTACION Y PRUEBAS DE LA APLICACIÓN WEB:

Para alojar la aplicación WEB basada en clases del lenguaje de programación Java, se ha utilizado el proyecto Tomcat de Apache, como solución al acceso. Para esto la tecnología JSP, fue necesaria para la programación de las interfaces de solicitud HTTP.

6.3.1.1 ANALISIS DE LOS RESULTADOS:

El sistema al momento de la pruebas funciono de manera correcta siendo implementada en la plataforma de Windows 7, y se puede decir que no se presentaron problemas al momento del funcionamiento de la interfaz la cual fue ejecutada desde diferentes navegadores WEB.

6.3.2 IMPLEMENTACION Y PRUEBAS DE COMUNICACIÓN CON EL SERVIDOR:

Para el desarrollo de la aplicación se ha utilizado herramientas de software libre, con el fin de contar con una solución económicamente viable. La herramienta utilizada para este proyecto fue EasyPHP la cual se instala en la máquina configurando un servidor Apache, que se pudo

utilizar para establecer la comunicación con los dispositivos, por medio del puerto USB, desde un entorno WEB.

6.3.2.1 RECURSOS UTILIZADOS:

- Puerto USB.
- Servidor Apache.
- Tarjeta con microcontrolador 18f4550.
- Tarjeta con puente H.

6.3.3 PRUEBAS DE LA APLICACIÓN WEB:

La aplicación WEB inicialmente fue probada bajo un ambiente de simulación virtual por medio de la utilización de la aplicación Proteus versión 7, el cual es un Software que es utilizado para el diseño, construcción, fabricación y simulación de circuitos electrónicos, constatando el correcto funcionamiento de los diferentes controles que se encuentran en la interfaz de la aplicación.

También se pudo comprobar que el sistema operativo, en este caso Windows 7 reconoce el hardware del proyecto, para lo cual es necesario instalar el respectivo controlador para el reconocimiento del MICRO PIC 18F4550 a través del puerto USB (MCHPUSB Driver), este controlador se puede descargar desde la página oficial de Microchip fabricante del microcontrolador utilizado en el proyecto. Se verifica la comunicación entre la aplicación y el hardware del sistema por medio de puerto USB resultando óptima.

Posteriormente se realizaron las respectivas pruebas de la aplicación WEB ejecutándola y verificando que el hardware del sistema responde a los controles de la interfaz, se pudo verificar que efectivamente tanto los controles de encendido, apagado, control de giro y el control de velocidad funcionan correctamente para los 4 motores. Inicialmente se encontró una dificultad al momento de controlar la velocidad de los motores ya que estos daban saltos a velocidades bajas, dicho problema fue posible solucionarlo incrementando la frecuencia con la que la aplicación verifica el estado del control de velocidad en la clase Timer la cual se utiliza para actualizar el ciclo de trabajo para el envío de señales desde la aplicación WEB, hacia el puerto USB; para luego enviar los respectivos datos al Hardware del sistema.

BIBLIOGRAFÍA

- CANDELAS HERIAS, Francisco A. Recursos didácticos basados en Internet para el apoyo a la enseñanza de materias del área de ingeniería de Sistemas y Automática. ISSN: 1697-7912. Vol. 2, Número. 2, Abril 2005.

- CLAVIJO MENDOZA, Juan Ricardo. Diseño y Simulación de sistemas Microcontrolados en Lenguaje C. Primera edición, Colombia: ISBN 978-958-44-8619-6, Mayo de 2011.

- CHAVEZ TREJO, Ana María. Ambiente de Telepresencia en la WEB para la realización de prácticas de laboratorio con el prototipo didáctico mecatrónico RefriLAB [en línea]. Instituto Tecnológico de Orizaba [citado Noviembre 26 y 27 de 2009]. Disponible en Internet: http://www.mecamex.net/anterior/cong08/articulos/45.pdf

- DORMIDO BENCOMO, Sebastián. Red de laboratorios de control automático a través de Internet [en línea]. E.T.S de Ingeniería Informática – UNED [citado en 2010]. Disponible en Internet: http://www.dia.uned.es/~fmorilla/Ultimas_publicaciones/2010_e-Automatica.pdf

- GAN CUBA, Wilson Antonio. Etapas de diseño de sistema de control electrónico vía WEB. Universidad de Pamplona Instituto de Investigación y Desarrollo de Tecnologías Aplicadas. ISSN: 1692-7257. Vol. 2, Número 10, 2007.

- GARCIA, Eduardo. Compilador C CCS y Simulador Proteus para Microcontroladores PIC. Primer Edición, México: AlfaOmega grupo editor, S.A, junio de 2008.

- GARCIA HARO, Juan Miguel. Desarrollo de un controlador para motores D.C brushless basado en CompactRIO y LabVIEW de National Instruments para el estudio de nuevos algoritmos de control [en línea]. Universidad Carlos III de Madrid [citado en Leganés, Noviembre de 2011]. Disponible en Internet: http://e-archivo.uc3m.es/bitstream/10016/13615/1/PFC_JuanMiguel_Garcia_Haro.pdf

- JOHANSSON, Moses. GÄFVERT, Michael. ASTRÖM, Kenneth. (1998) "Interactive Tools for Education in Automatic Control". IEEE Control Systems Magazine, Vol. 18, pp. 33-40.

- LORENZO MARTINEZ, Noelia. Aplicación docente de una plataforma de accionamientos mecatrónicos controlada a través de Internet [en línea]. Treballs acadèmics UPC, Universitat Politècnica de Catalunya

[citado en 2006]. Disponible en Internet: http://upcommons.upc.edu/pfc/handle/2099.1/3620

- NAUGHTON, Patrick. Manual de Java. Primera Edición, Aravaca (Madrid): Editorial McGRAW – HILL, 1996.

- OVERSTREET, William. (1999). "An InternetBased Real-Time Control Engineering Laboratory", IEEE Control Systems Magazine, Vol. 19, pp. 19-34.

- SCHMID, Chris. (1992): "Real – Time Control with CADACS-PC". Recent Advances in computer – Aided Control Systems Engineering. M. Jamshidi and C.J. Herget (Editors), 337 – 355. North – Holland, Amsterdam, 1992.

- TRADE PRESS RELATIONS & TRANSLATION SERVICES. Nueva tecnología de mando por Ethernet para accionamientos y sistemas [en línea]. TPR International [citado marzo de 2011]. Disponible en Internet: http://www.tradepressrelations.com/en/file/presse/es/presse1346.php

- VALERA, Aurelio. Control remoto de procesos industriales con Matlab Web Server. Universidad Politécnica de Valencia Camino de Vera 14, 46022 Valencia (España).

- VALERA, Andrew. (2001) "LabConRob: Virtual Laboratory of Real-Time Robot Control", Robótica 43. ISBN 152701/00. SIN 0874-9019. Pp 40-45.

ANEXOS

ANEXO A. Especificación del problema

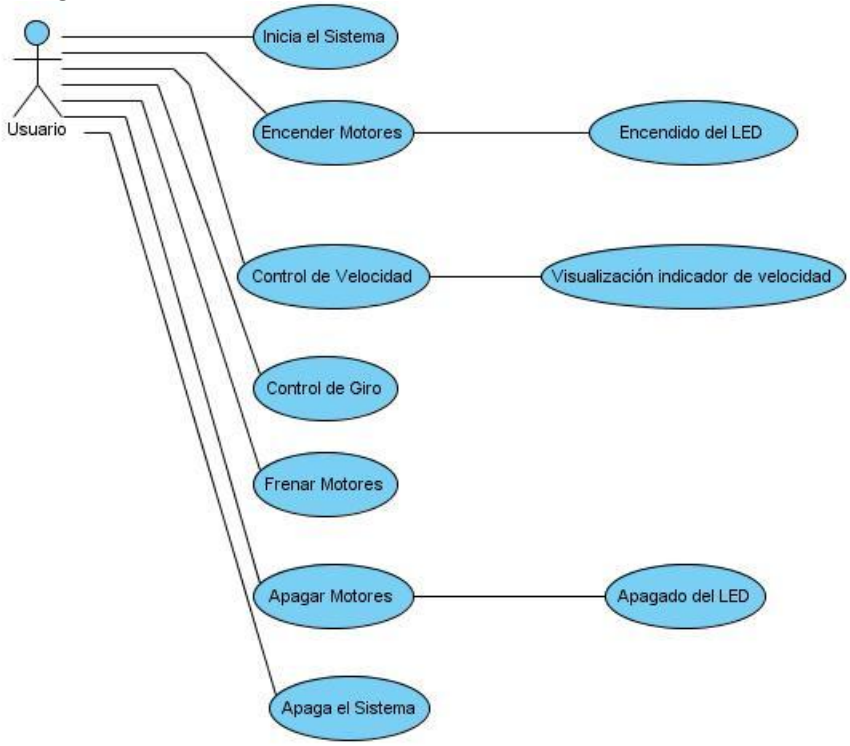
a) Descripción del caso de estudio:

Se desea crear una aplicación WEB que permita establecer la comunicación por medio del puerto USB con una tarjeta electrónica construida con base en el microcontrolador 18F4550 de microchip, con el fin de controlar desde la interfaz el movimiento de 4 motores de corriente continua convencionales, este control debe permitir establecer el encendido, apagado, control de giro y velocidad de los motores.

Por medio de la aplicación WEB el usuario del sistema tiene la posibilidad de encender cada uno de los motores para lo cual se enciende un led en la interfaz indicando el encendido del motor correspondiente, es de anotar que cada motor cuenta con sus respectivos controles, desde los cuales se puede indicar la acción deseada. Si el usuario desea que el motor gire a la izquierda es necesario oprimir el botón que indica el sentido de giro a la izquierda, haciendo clic con el botón principal del mouse, el mismo procedimiento aplica para los demás controles a saber: giro derecha, frenado, encendido y apagado. Para el control de velocidad es necesario desplazar con el clic sostenido del mouse, la barra de desplazamiento que controla la velocidad del motor, luego de realizar esta acción la velocidad del motor se puede visualizar en un indicador que muestra la velocidad a la que gira el motor en términos de porcentaje.

b) Modelo de casos de uso:

Figura 27. Diagrama de casos de uso.



Descripción de los casos de uso:

- **Caso de uso: Inicia el Sistema.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida y el estado del registro de datos en el servidor se ha actualizado.

PASO	DESCRIPCION
1	El usuario inicializa un programa navegador (Googlechrome, Mozilla, Internet explorer, Opera, etc…..)
2	El usuario inicia el sistema digitando la siguiente dirección localhost:8080/pruebaWeb/, desde cualquier programa navegador.
3	En la pantalla se dibuja la interfaz Web, que contiene todos los controles que se necesitan para el control de 4 motores de corriente continua, que le permite al usuario del sistema interactuar con la aplicación.
4	Fin del caso de uso.

- **Caso de uso: Encender motores.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida.

PASO	DESCRIPCION
1	El usuario hace click con el botón izquierdo del mouse, sobre el botón de encendido o apagado del motor que haya seleccionado.
2	Inmediatamente se puede visualizar el encendido del led (Diodo emisor de luz) que se encuentra en la interfaz, indicando que el respectivo motor se encuentra encendido.
3	La aplicación Web interpreta la señal enviada por el usuario y envía el código correspondiente al encendido del motor, por el puerto USB.
4	Fin del caso de uso.

- **Caso de uso: Control de Velocidad.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida.

PASO	DESCRIPCION
1	El usuario hace click sostenido con el control izquierdo del mouse, sobre el control de la barra de desplazamiento del control de velocidad.
2	Aun con el click sostenido el control se desplaza de arriba hacia

	abajo, hasta seleccionar el porcentaje de velocidad deseado.
3	La interfaz detecta la señal de desplazamiento del control de velocidad y envía la señal respectiva por el puerto USB de la computadora.
4	La señal del desplazamiento del control de velocidad se visualiza sobre el indicador de velocidad de la aplicación.
5	Fin del caso de uso.

- **Caso de uso: Control de giro.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida.

PASO	DESCRIPCION
1	El usuario hace click con el botón izquierdo del mouse, sobre el control de giro que haya elegido para controlar el motor.
2	La aplicación Web envía la señal que indica el sentido de giro del motor, por el puerto USB.
3	Fin del caso de uso.

- **Caso de uso: Frenar Motores.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida.

PASO	DESCRIPCION
1	El usuario hace click con el botón izquierdo del mouse sobre el control que se utiliza para frenar el motor correspondiente.
2	La aplicación Web envía la señal que corresponde al frenado del motor correspondiente, por medio del puerto USB.
3	Fin del caso de uso.

- **Caso de uso: Apagar Motores.**

  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: La conexión con el servidor continúa establecida.

PASO	DESCRIPCION
1	El usuario hace click con el botón izquierdo del mouse, sobre el botón de encendido o apagado del motor que haya seleccionado.
2	Inmediatamente se puede visualizar el apagado del led (Diodo emisor de luz) que se encuentra en la interfaz, indicando que el respectivo motor se encuentra apagado.
3	La aplicación Web interpreta la señal enviada por el usuario y envía el código correspondiente al apagado del motor, por el puerto USB.
4	Fin del caso de uso.

- **Caso de uso: Apaga el Sistema.**

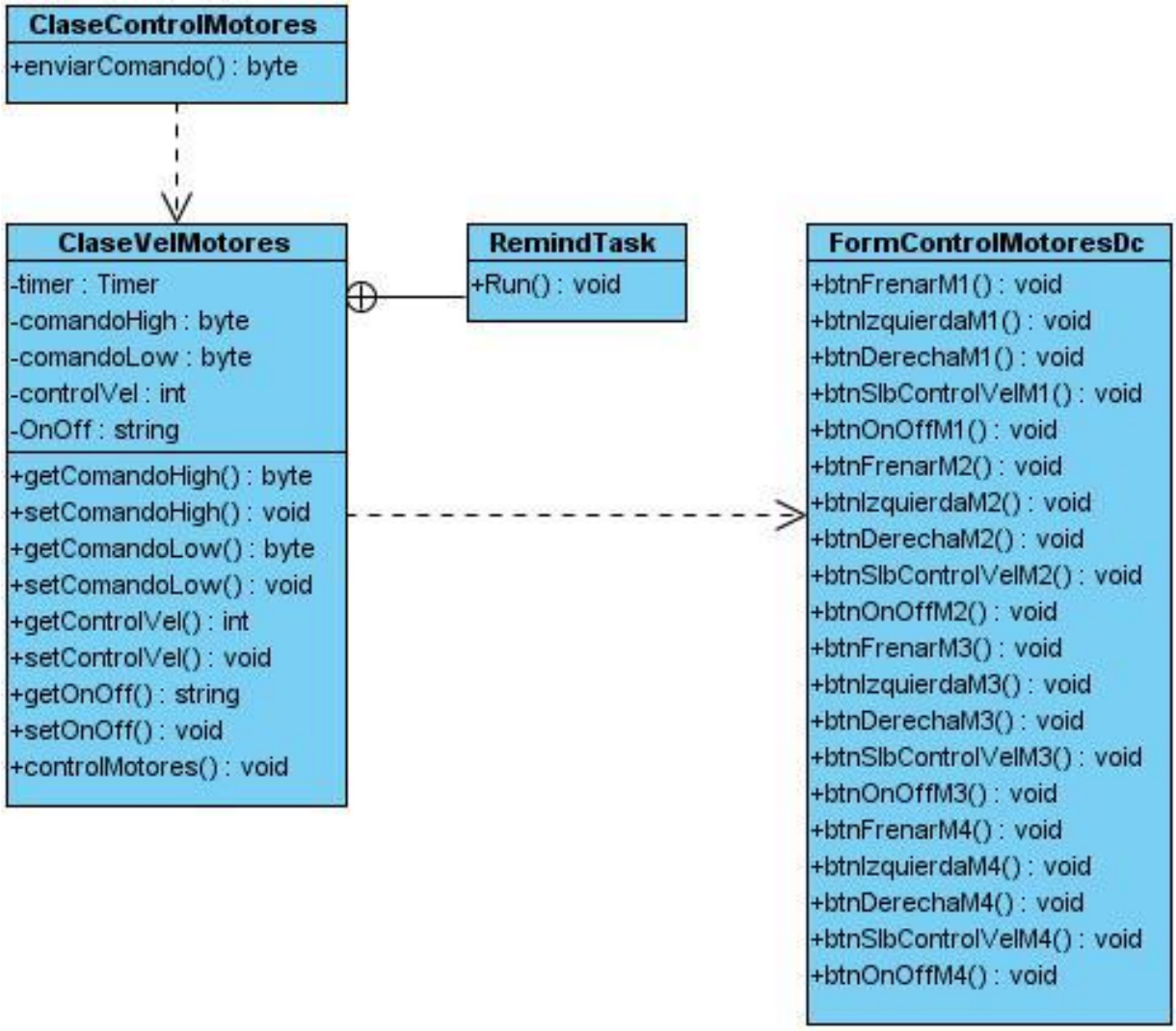
  Precondiciones: La conexión con el servidor se encuentra establecida.
  Postcondiciones: Se cierra la conexión con el servidor.

PASO	DESCRIPCION
1	El usuario hace click sobre el botón de cerrar la aplicación del navegador.
2	Se cierra el navegador.
3	Finaliza la comunicación por el puerto USB entre la aplicación y el hardware del sistema.
4	Fin del caso de uso.

ANEXO B. Análisis

1. Modelo estático

a. Diagrama de clases.



b. Diccionario de Clases

Clase: ClaseControl Motores
Descripción: Esta clase se encarga de establecer la comunicación entre la aplicación WEB y el microcontrolador 18F4550 por medio del puerto USB.

Descripción de métodos.

Método	Descripción
enviarComando()	Este método se encarga de enviar el flujo de datos a través de comandos tipo byte, los cuales le indican al microcontrolador las acciones que debe efectuar.

Clase: ClaseVelMotores
Descripción: Esta clase se encarga de gestionar los datos enviados por la interfaz del sistema.

Descripción de atributos.

Atributo	Descripción	Tipo de dato
timer	Este atributo que sirve para instanciar un objeto de la clase Timer().	TIMER
comandoHigh	Este atributo se utiliza con el fin de recibir un dato de tipo byte, con el fin de poder controlar el el pulso alto de la señal de PWM.	{10-11-12-13-14-15-16-17}
comandoLow	Este atributo se utiliza con el fin de recibir un dato de tipo byte, con el fin de poder controlar el el pulso bajo de la señal de PWM.	{00-01-02-03-04-05-06-07}
controlVel	El atributo controlVel es usado para recibir el dato de la barra de desplazamiento de la interfaz, la cual entrega un valor del tipo entero.	{0…..100}
OnOff	Este atributo recibe un dato del tipo String el cual es utilizado con el	{On-Off}

fin de controlar el estado de encendido y apagado de los motores.

Descripción de métodos.

Método	Descripción
getComandHigh()	Método que des-encapsula los datos que corresponden al pulso alto del PWM.
setComandoHigh()	Método que encapsula los datos que corresponden al pulso alto del PWM.
getComandoLow()	Método que des-encapsula los datos que corresponden al pulso bajo del PWM.
setComandoLow()	Método que encapsula los datos que corresponden al pulso bajo del PWM.
getControlVel()	Método que des-encapsula los datos que corresponden al control de velocidad de los motores y sirve de referencia de comparación para establecer el periodo de duración de los semiciclos del PWM.
setControlVel()	Método que encapsula los datos que corresponden al control de velocidad de los motores y sirve de referencia de comparación para establecer el periodo de duración de los semiciclos del PWM.
getOnOff()	Método que des-encapsula los datos que son enviados desde la interfaz del sistema y sirve para controlar la función de encendido y apagado de los motores.
setOnOff()	Método que encapsula los datos que son enviados desde la interfaz del sistema y sirve para controlar la función de encendido y apagado de los motores.
controlMotores()	Este método es utilizado para lanzar una tarea por medio de la clase Timer, la cual consiste en actualizar permanentemente el envío de datos desde la interfaz, con el fin de enviar en tiempo real las ordenes que debe ejecutar el microcontrolador, por medio de la interfaz gráfica de la aplicación.

Clase: RemindTask
Descripción: Clase que ejecuta tareas específicas en un tiempo determinado.

Descripción de métodos.

Método	Descripción
Run()	Este método lleva a cabo la ejecución del temporizador de la clase Timer()

Clase: FormControlMotores
Descripción: Esta clase corresponde a la interfaz gráfica del sistema.

Descripción de métodos.

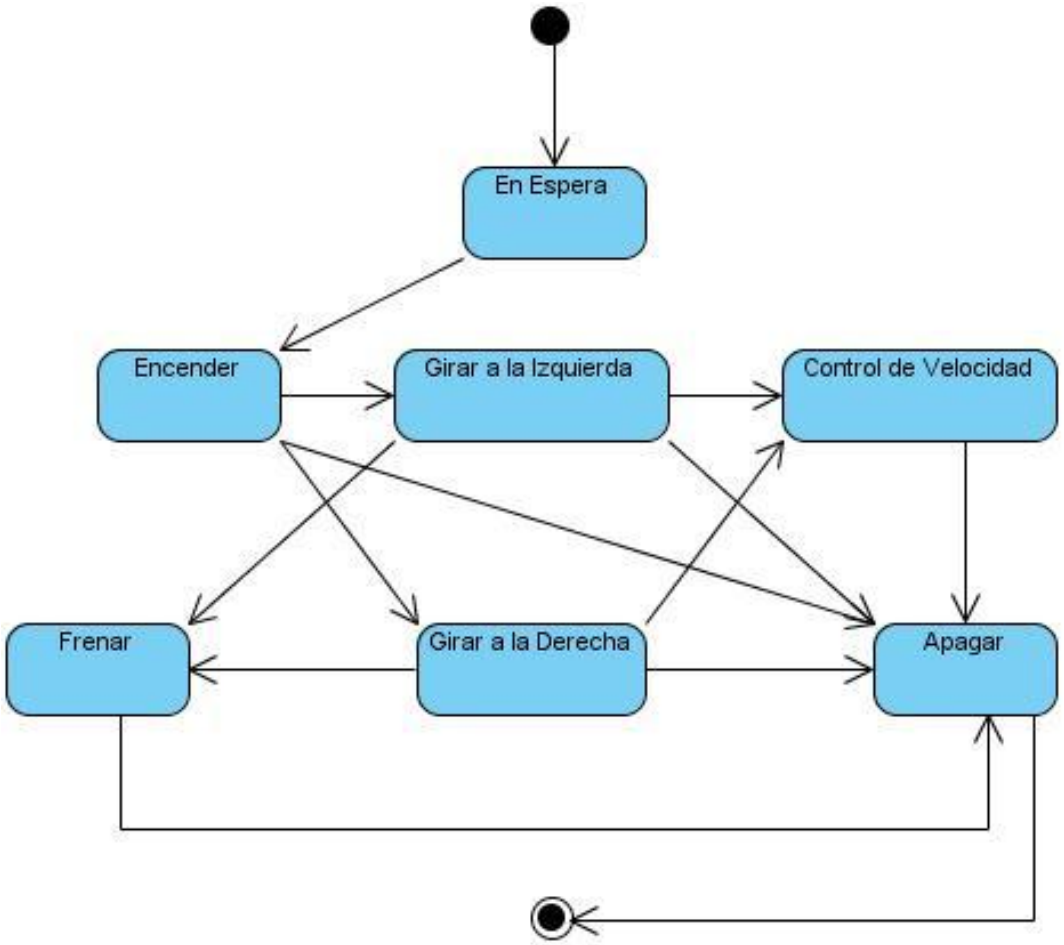
Método	Descripción
btnFrenarM1()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de frenado del motor 1.
btnIzquierdaM1()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la izquierda del motor 1.
btnDerechaM1()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la derecha del motor 1.
btnSlbControlVelM1()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de control de velocidad del motor 1.
btnOnOffM1()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de encendido y apagado del motor 1.
btnFrenarM2()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de frenado del motor 2.
btnIzquierdaM2()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la izquierda del motor 2.
btnDerechaM2()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la derecha del motor 2.
btnSlbControlVelM2()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de control de velocidad del motor 2.
btnOnOffM2()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de encendido y apagado del motor 2.
btnFrenarM3()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de frenado del motor 3.
btnIzquierdaM3()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la izquierda del motor 3.
btnDerechaM3()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la derecha del motor 3.
btnSlbControlVelM3()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de control de velocidad del motor 3.
btnOnOffM3()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de encendido y apagado del motor 3.

btnFrenarM4()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de frenado del motor 4.
btnIzquierdaM4()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la izquierda del motor 4.
btnDerechaM4()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de giro a la derecha del motor 4.
btnSlbControlVelM4()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de control de velocidad del motor 4.
btnOnOffM4()	Método que define las acciones que se deben ejecutar al momento de enviar la señal de encendido y apagado del motor 4.

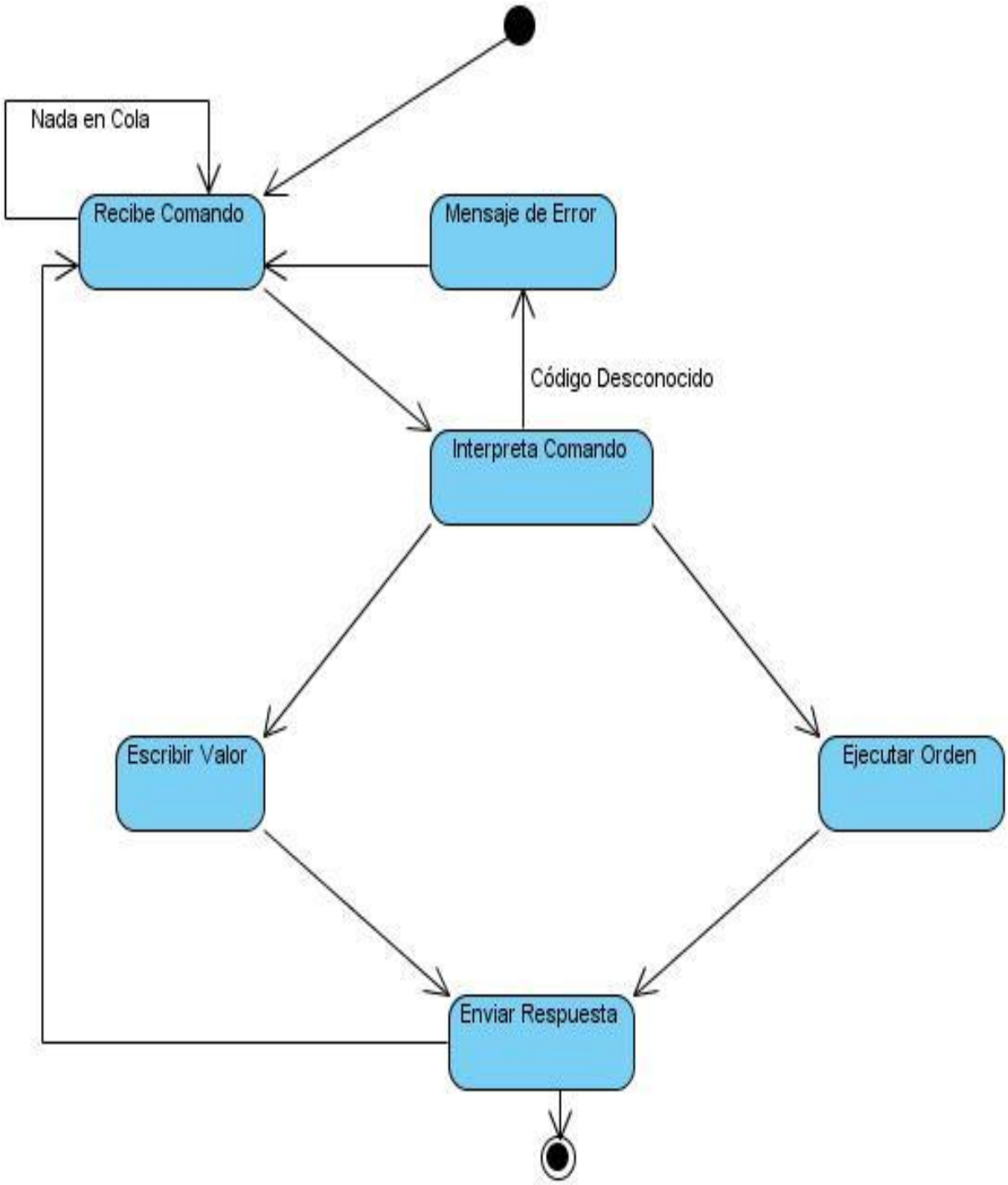
2. Modelo dinámico

a. Diagrama de Estados

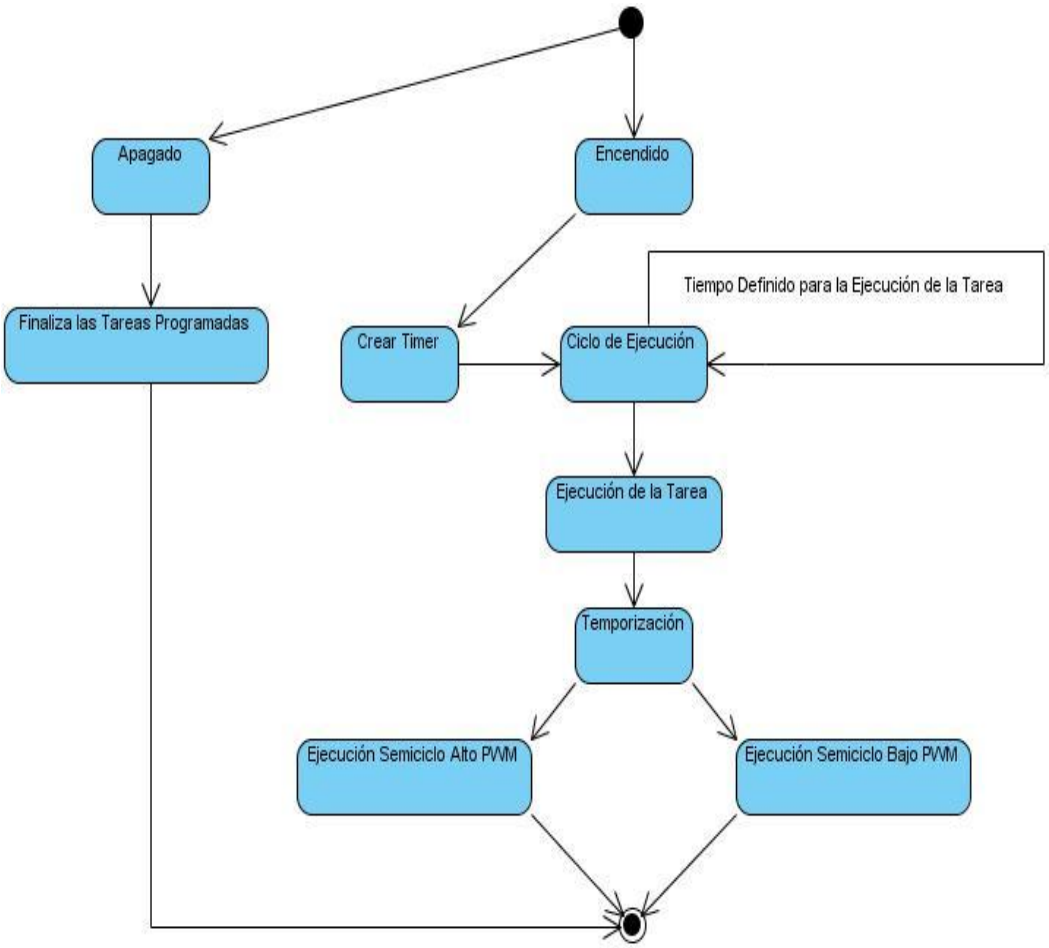
Clase Formulario:



Clase Control Motores:



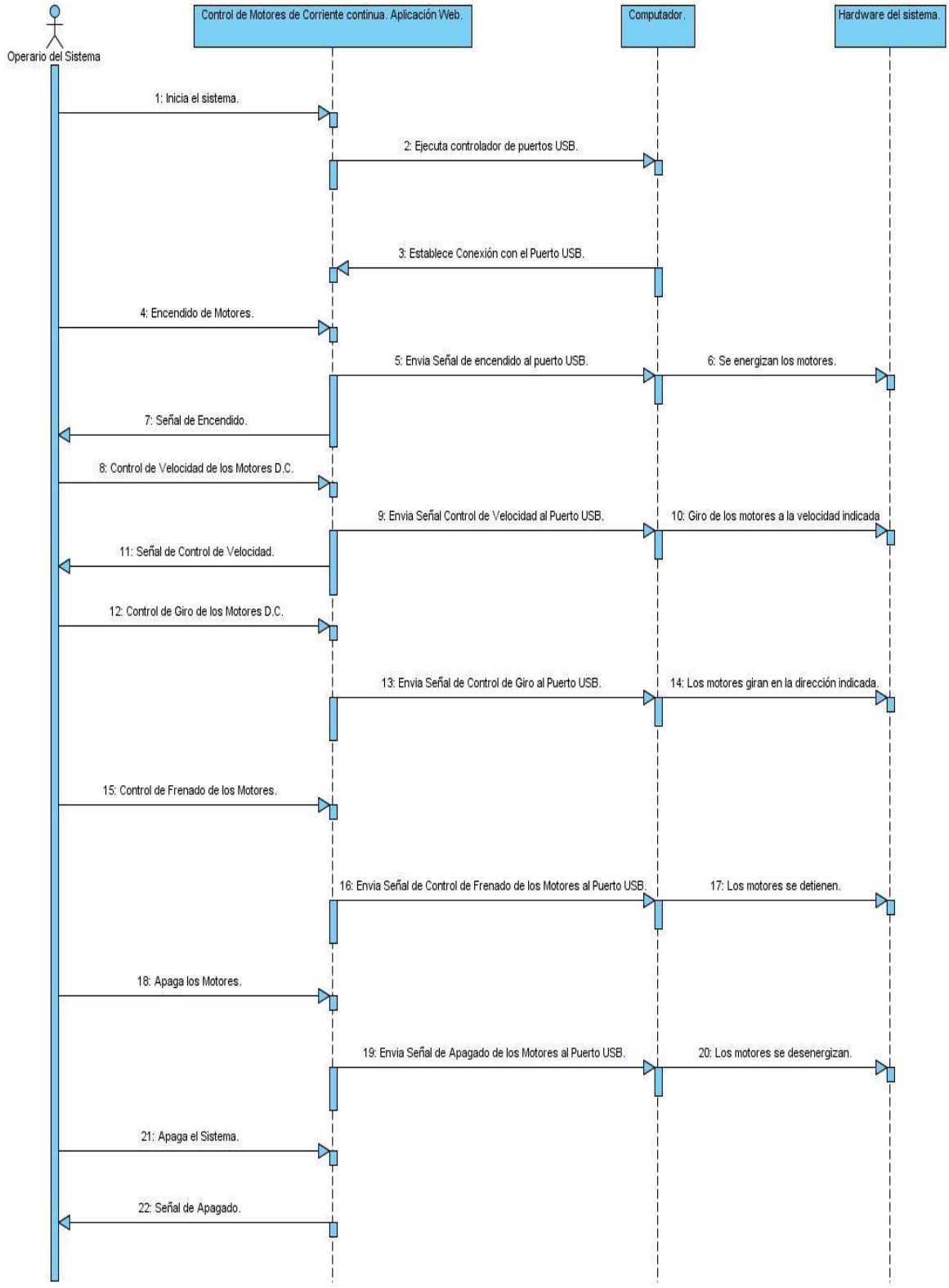
Clase Velocidad Motores:



3. Modelo de Actividades

En el diagrama de secuencia se puede observar una interacción ordenada y secuencial de una cadena de eventos temporales. En este tipo de diagramas el eje vertical representa la línea de tiempo, en el eje horizontal se puede apreciar cada uno de los actores y objetos que intervienen en la interacción. Las flechas representan mensajes que se dan entre los diferentes actores u objetos. El tiempo en este tipo de diagramas siempre fluye de arriba hacia abajo; este tipo de diagramas se utilizan para representar de manera simplificada las interacciones que se tienen entre los diferentes objetos, en la siguiente figura se puede apreciar el diagrama de secuencia de la aplicación Web.

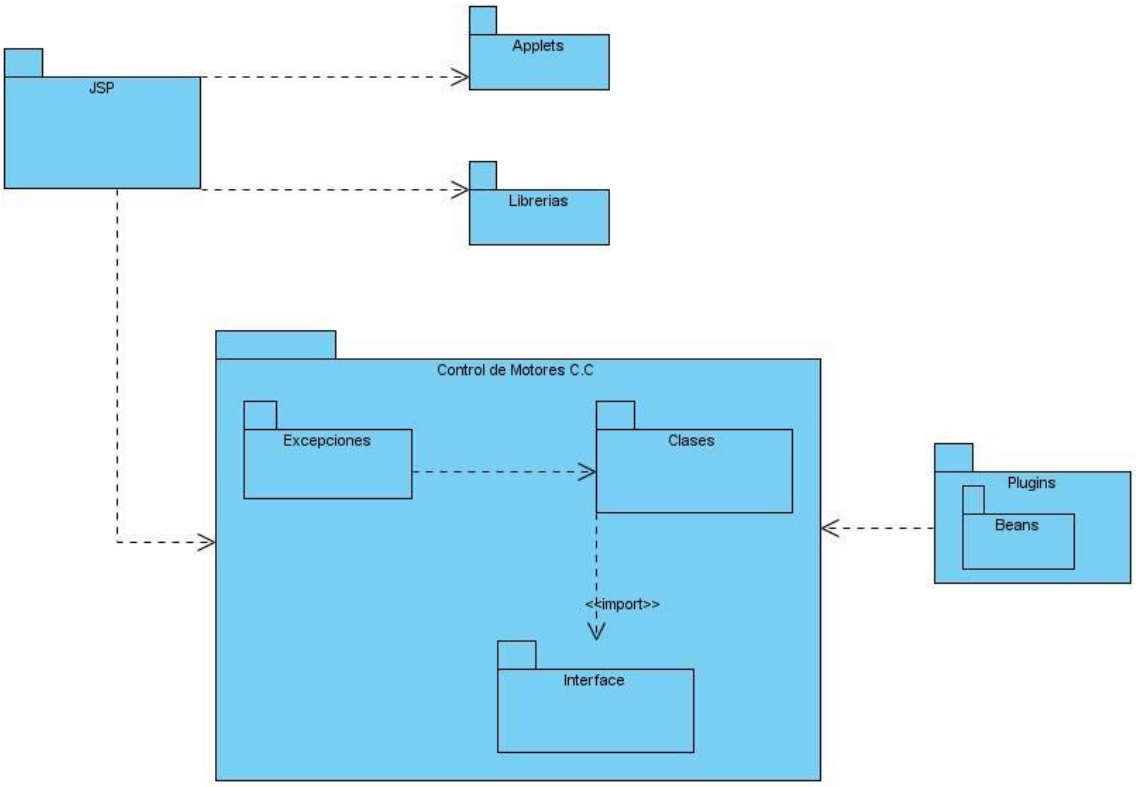
a. Diagrama de Secuencia


Operario del Sistema
Control de Motores de Corriente continua. Aplicación Web.
Computador.
Hardware del sistema.
1: Inicia el sistema.
2: Ejecuta controlador de puertos USB.
3: Establece Conexión con el Puerto USB.
4: Encendido de Motores.
5: Envia Señal de encendido al puerto USB.
6: Se energizan los motores.
7: Señal de Encendido.
8: Control de Velocidad de los Motores D.C.
9: Envia Señal Control de Velocidad al Puerto USB.
10: Giro de los motores a la velocidad indicada
11: Señal de Control de Velocidad.
12: Control de Giro de los Motores D.C.
13: Envia Señal de Control de Giro al Puerto USB.
14: Los motores giran en la dirección indicada.
15: Control de Frenado de los Motores.
16: Envia Señal de Control de Frenado de los Motores al Puerto USB.
17: Los motores se detienen.
18: Apaga los Motores.
19: Envia Señal de Apagado de los Motores al Puerto USB.
20: Los motores se desenergizan.
21: Apaga el Sistema.
22: Señal de Apagado.

ANEXO C. Diseño

1. Diseño del sistema

a. Diagrama de paquetes



b. Herramientas

Software: Se debe desarrollar un sistema que permita realizar el control de motores de corriente continua convencionales, que funcione como aplicación WEB utilizando el puerto USB de la computadora como el medio para la transmisión de los datos.

Con base en el anterior planteamiento se decidió utilizar el lenguaje de programación JAVA por sus grandes características que contribuyen a satisfacer las necesidades de la aplicación, otorgándole una gran usabilidad al sistema.

La interfaz fue diseñada utilizando un entorno WEB el cual utiliza plantillas hechas con lenguaje XHTML (Lenguaje de Marcas de Hipertexto Extendido), se utiliza un Applet que se ejecuta en un navegador, el cual sirve para realizar el control del sistema. El servidor WEB que se utiliza para la ejecución de la aplicación es Tomcat Apache.

La interfaz del sistema se comunica con unas clases que permiten la interpretación de las ordenes que son enviadas al módulo de hardware, que es controlado por el microcontrolador PIC 18F4550; la comunicación se establece

mediante un paquete de clases llamado jpicusb 1.1.1 que se encarga de hacer las transacciones y enviar los datos por medio del puerto USB.

Hardware: El hardware del sistema consiste en un módulo construido a partir de un microcontrolador PIC 18F4550 y una tarjeta electrónica que contiene un puente H integrado L298, que se encarga de interpretar los comandos enviados desde la aplicación WEB por medio del puerto USB.

La siguiente tabla muestra los estados que corresponden a los comando enviados a través del puerto USB para el control de los motores.

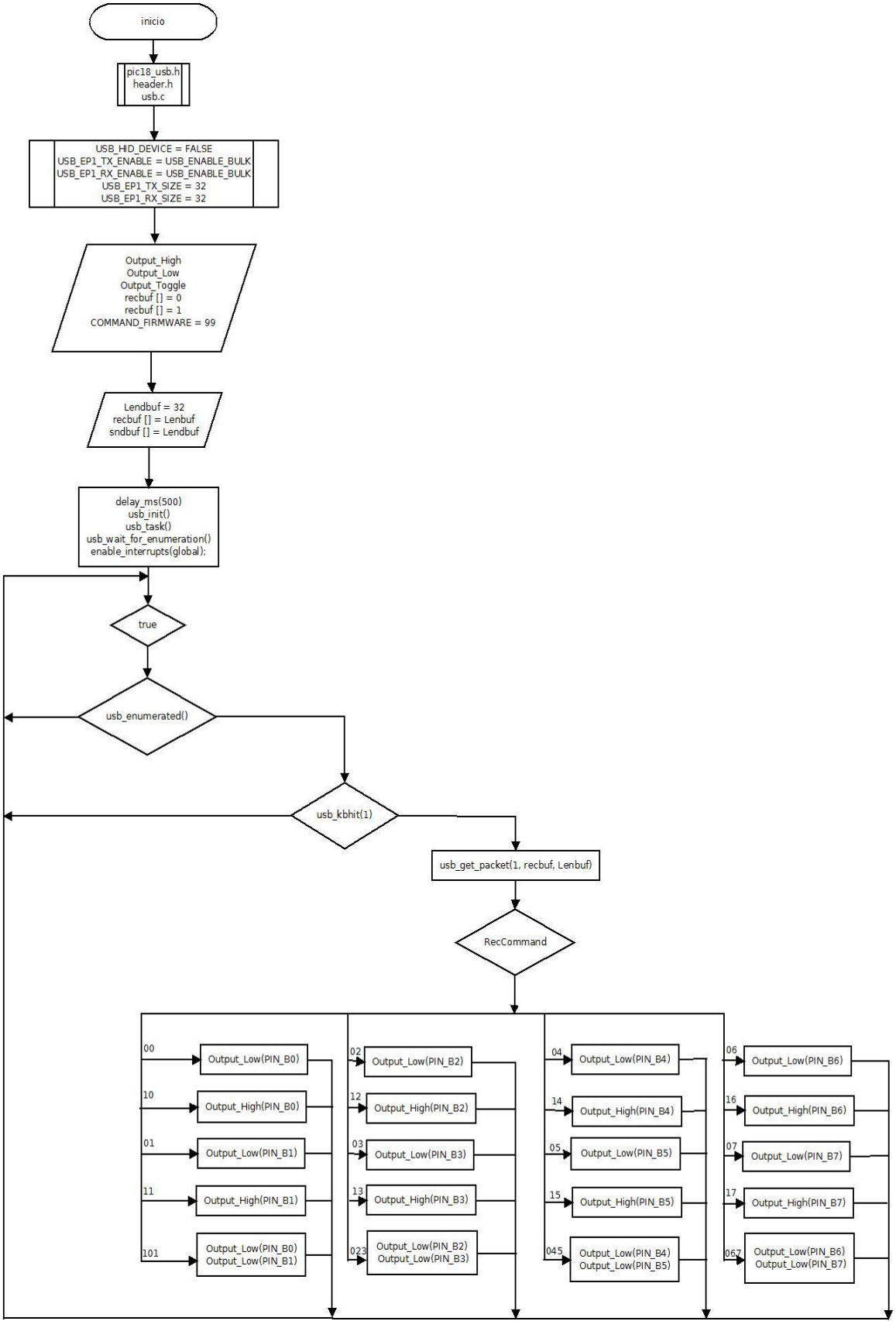
CUADRO DE ESTADOS DEL PUERTO B DEL PIC															
MOTOR 1				MOTOR 2				MOTOR 3				MOTOR 4			
GIRO DERECHA PIN B0		GIRO IZQUIERDA PIN B1		GIRO DERECHA PIN B2		GIRO IZQUIERDA PIN B3		GIRO DERECHA PIN B4		GIRO IZQUIERDA PIN B5		GIRO DERECHA PIN B6		GIRO IZQUIERDA PIN B7	
PWML	OO	PWML	O1	PWML	O2	PWML	O3	PWML	O4	PWML	O5	PWML	O6	PWML	O7
PWMH	1O	PWMH	11	PWMH	12	PWMH	13	PWMH	14	PWMH	15	PWMH	16	PWMH	17
PARAR PIN B0 - PIN B1				PARAR PIN B2 - PIN B3				PARAR PIN B4 - PIN B5				PARAR PIN B6 - PIN B7			
PWML	101			PWML	O23			PWML	O45			PWML	O67		

Firmware: El firmware es el programa que se encuentra grabado dentro del microcontrolador el cual contiene las instrucciones que le permiten actuar de la manera deseada, ejecutando así las instrucciones que son enviadas desde la computadora, dichas instrucciones viajan por el puerto USB de la computadora y son recibidas por los puertos de este componente para luego ser interpretadas y ejecutadas.

Los comandos necesarios para efectuar las acciones requeridas sobre los motores como son: Encendido, apagado, sentido de giro y velocidad; están detallados en una tabla de valores, esta nos indica el tipo de instrucciones que deben ser enviadas y luego ejecutadas por el microcontrolador, es de anotar que estos datos del tipo Byte o sea son números del tipo binario, compuestos por 8 bits.

También se debe resaltar que es sobre el puerto B del microcontrolador desde donde se conectan los circuitos integrados que contienen los puentes H que son los encargados de controlar el sentido de giro y velocidad de los motores, en la siguiente tabla se pueden apreciar los estados del puerto B.

c. Diagrama de flujo del microcontrolador 18F4550



ANEXO D. Código fuente de la aplicación

1. Código Java de la Aplicación WEB

a. ClaseControlMotores

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Alexander
 */

import jPicUsb.iface;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ClaseControlMotores {


    public ClaseControlMotores(){
        try {
            iface.load();
        } catch (Exception ex) {

Logger.getLogger(ClaseControlMotores.class.getName()).log(Level.SEVE
RE, null, ex);
        }
    }

    public void enviarComando(byte comando){
        byte [] salida = {comando};
        iface.QWrite(salida, 2, 1000);
    }
}
```

b. ClaseVelMotores

```
import java.util.Timer;
import java.util.TimerTask;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
 *
 * @author alexander
 */



public class ClaseVelMotores extends ClaseControlMotores{
    private Timer timer;
    private byte comandoHigh;
    private byte comandoLow;
    private int controlVel;
    private String OnOff;

    /**
     * @return the comandoHigh
     */
    public byte getComandoHigh() {
        return comandoHigh;
    }

    /**
     * @param comandoHigh the comandoHigh to set
     */
    public void setComandoHigh(byte comandoHigh) {
        this.comandoHigh = comandoHigh;
    }

    /**
     * @return the comandoLow
     */
    public byte getComandoLow() {
        return comandoLow;
    }

    /**
     * @param comandoLow the comandoLow to set
     */
    public void setComandoLow(byte comandoLow) {
        this.comandoLow = comandoLow;
    }

    /**
     * @return the controlVel
     */
    public int getControlVel() {
        return controlVel;
    }
```

```
    /**
     * @param controlVel the controlVel to set
     */
    public void setControlVel(int controlVel) {
        this.controlVel = controlVel;
    }

    /**
     * @return the OnOff
     */
    public String getOnOff() {
        return OnOff;
    }

    /**
     * @param OnOff the OnOff to set
     */
    public void setOnOff(String OnOff) {
        this.OnOff = OnOff;
    }

    public void controlMotores(){
        if(getOnOff()=="On"){
            timer = new Timer();
            timer.schedule(new RemindTask(), 0, 1*1000);    //Se le asigna una
tarea al timer
        }
        if(getOnOff()=="Off"){
            timer.cancel();
        }
    }

    class RemindTask extends TimerTask{

        public void run() {
            int pwmh = getControlVel(); //Semiperiodo alto del PWM
            int pwml = 100 - pwmh;      //Semiperiodo bajo del PWM

            for(pwmh=pwmh;pwmh>0;pwmh--){
                enviarComando(getComandoHigh());    //Salida comando alto del
PWM
            }
            for(pwml=pwml;pwml>0;pwml--){
                enviarComando(getComandoLow());     //Salida comando bajo
del PWM
            }
        }
    }
}
```

c. FormControlMotoresDC

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author censa
 */
public class FormAppletControlMotDc extends javax.swing.JApplet{
    ClaseVelMotores objMotor1 = new ClaseVelMotores();
    ClaseVelMotores objMotor2 = new ClaseVelMotores();
    ClaseVelMotores objMotor3 = new ClaseVelMotores();
    ClaseVelMotores objMotor4 = new ClaseVelMotores();

    /**
     * Initializes the applet FormAppletControlMotDc
     */
    @Override
    public void init() {
        /*
         * Set the Nimbus look and feel
         */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
        /*
         * If Nimbus (introduced in Java SE 6) is not available, stay with the
         * default look and feel. For details see
         *
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(FormAppletControlMotDc.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(FormAppletControlMotDc.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(FormAppletControlMotDc.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(FormAppletControlMotDc.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /*
         * Create and display the applet
         */
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {

                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**
     * This method is called from within the init() method to initialize the
     * form. WARNING: Do NOT modify this code. The content of this
method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jToolBar1 = new javax.swing.JToolBar();
        jLabel1 = new javax.swing.JLabel();
        jInternalFrame1 = new javax.swing.JInternalFrame();
        btnIzquierda = new javax.swing.JButton();
        btnDerecha = new javax.swing.JButton();
        btnParar = new javax.swing.JButton();
        btnOnOffM1 = new javax.swing.JToggleButton();
        slbControlVelM1 = new javax.swing.JSlider();
        lblLedM1 = new javax.swing.JLabel();
        txtPorcentajeM1 = new javax.swing.JLabel();
        jInternalFrame2 = new javax.swing.JInternalFrame();
        btnIzquierda1 = new javax.swing.JButton();
        btnDerecha1 = new javax.swing.JButton();
        btnParar1 = new javax.swing.JButton();
```

```
        btnOnOffM2 = new javax.swing.JToggleButton();
        slbControlVelM2 = new javax.swing.JSlider();
        lblLedM2 = new javax.swing.JLabel();
        txtPorcentajeM2 = new javax.swing.JLabel();
        jInternalFrame3 = new javax.swing.JInternalFrame();
        btnIzquierda2 = new javax.swing.JButton();
        btnDerecha2 = new javax.swing.JButton();
        btnParar2 = new javax.swing.JButton();
        btnOnOffM3 = new javax.swing.JToggleButton();
        slbControlVelM3 = new javax.swing.JSlider();
        lblLedM3 = new javax.swing.JLabel();
        txtPorcentajeM3 = new javax.swing.JLabel();
        jInternalFrame4 = new javax.swing.JInternalFrame();
        btnIzquierda3 = new javax.swing.JButton();
        btnDerecha3 = new javax.swing.JButton();
        btnParar3 = new javax.swing.JButton();
        btnOnOffM4 = new javax.swing.JToggleButton();
        slbControlVelM4 = new javax.swing.JSlider();
        lblLedM4 = new javax.swing.JLabel();
        txtPorcentajeM4 = new javax.swing.JLabel();

        getContentPane().setLayout(null);

        jToolBar1.setRollover(true);

        jLabel1.setText("CONTROL DE MOTORES DE CORRIENTE
CONTINUA");
        jToolBar1.add(jLabel1);

        getContentPane().add(jToolBar1);
        jToolBar1.setBounds(0, 0, 642, 25);

        jInternalFrame1.setTitle("MOTOR 1");
        jInternalFrame1.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jInternalFrame1.setVisible(true);

        btnIzquierda.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnIzquierda.gi
f"))); // NOI18N
        btnIzquierda.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIzquierdaActionPerformed(evt);
            }
        });

        btnDerecha.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnDerecha.gif
"))); // NOI18N
        btnDerecha.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDerechaActionPerformed(evt);
            }
        });

        btnParar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/parar.gif"))); //
NOI18N
        btnParar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnPararActionPerformed(evt);
            }
        });

        btnOnOffM1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/onOff.gif"))); //
NOI18N
        btnOnOffM1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOnOffM1ActionPerformed(evt);
            }
        });

        slbControlVelM1.setOrientation(javax.swing.JSlider.VERTICAL);
        slbControlVelM1.setValue(0);
        slbControlVelM1.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                slbControlVelM1StateChanged(evt);
            }
        });

        lblLedM1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif"))); //
NOI18N

        txtPorcentajeM1.setFont(new java.awt.Font("Tahoma", 1, 18)); //
NOI18N
        txtPorcentajeM1.setText("0%");

txtPorcentajeM1.setVerticalAlignment(javax.swing.SwingConstants.BOTT
OM);
        txtPorcentajeM1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAI
SED));

txtPorcentajeM1.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);
```

```
        javax.swing.GroupLayout jInternalFrame1Layout = new
javax.swing.GroupLayout(jInternalFrame1.getContentPane());
        jInternalFrame1.getContentPane().setLayout(jInternalFrame1Layout);
        jInternalFrame1Layout.setHorizontalGroup(

jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(jInternalFrame1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jInternalFrame1Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                    .addComponent(btnOnOffM1,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addGroup(jInternalFrame1Layout.createSequentialGroup()
                        .addComponent(btnIzquierda,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                        .addComponent(btnParar,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                        .addComponent(btnDerecha,
javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addComponent(slbControlVelM1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jInternalFrame1Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                    .addGroup(jInternalFrame1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                        .addComponent(txtPorcentajeM1,
javax.swing.GroupLayout.DEFAULT_SIZE, 90, Short.MAX_VALUE))
                    .addGroup(jInternalFrame1Layout.createSequentialGroup()
                        .addGap(35, 35, 35)
                        .addComponent(lblLedM1)
                        .addGap(0, 29, Short.MAX_VALUE)))
                .addContainerGap())
        );
```

```
        jInternalFrame1Layout.setVerticalGroup(

jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(jInternalFrame1Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jInternalFrame1Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                    .addGroup(jInternalFrame1Layout.createSequentialGroup()
                        .addComponent(lblLedM1,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(39, 39, 39)
                        .addComponent(txtPorcentajeM1))

.addGroup(jInternalFrame1Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                        .addGroup(jInternalFrame1Layout.createSequentialGroup()

.addGroup(jInternalFrame1Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                                .addComponent(btnParar,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(btnDerecha,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(btnIzquierda))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                            .addComponent(btnOnOffM1,
javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(slbControlVelM1,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        getContentPane().add(jInternalFrame1);
        jInternalFrame1.setBounds(10, 36, 302, 200);

        jInternalFrame2.setTitle("MOTOR 2");
        jInternalFrame2.setVisible(true);

        btnIzquierda1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnIzquierda.gi
f"))); // NOI18N
```

```
        btnIzquierda1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIzquierda1ActionPerformed(evt);
            }
        });

        btnDerecha1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnDerecha.gif
"))); // NOI18N
        btnDerecha1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDerecha1ActionPerformed(evt);
            }
        });

        btnParar1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/parar.gif"))); //
NOI18N
        btnParar1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnParar1ActionPerformed(evt);
            }
        });

        btnOnOffM2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/onOff.gif"))); //
NOI18N
        btnOnOffM2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOnOffM2ActionPerformed(evt);
            }
        });

        slbControlVelM2.setOrientation(javax.swing.JSlider.VERTICAL);
        slbControlVelM2.setValue(0);
        slbControlVelM2.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                slbControlVelM2StateChanged(evt);
            }
        });

        lblLedM2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif"))); //
NOI18N

        txtPorcentajeM2.setFont(new java.awt.Font("Tahoma", 1, 18)); //
NOI18N
        txtPorcentajeM2.setText("0%");
```

```
txtPorcentajeM2.setVerticalAlignment(javax.swing.SwingConstants.BOTTOM);
        txtPorcentajeM2.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

txtPorcentajeM2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        javax.swing.GroupLayout jInternalFrame2Layout = new javax.swing.GroupLayout(jInternalFrame2.getContentPane());
        jInternalFrame2.getContentPane().setLayout(jInternalFrame2Layout);
        jInternalFrame2Layout.setHorizontalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jInternalFrame2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(btnOnOffM2, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addGroup(jInternalFrame2Layout.createSequentialGroup()
                        .addComponent(btnIzquierda1, javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(btnParar1, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(btnDerecha1, javax.swing.GroupLayout.PREFERRED_SIZE, 42, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addComponent(slbControlVelM2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jInternalFrame2Layout.createSequentialGroup()
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addComponent(txtPorcentajeM2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGroup(jInternalFrame2Layout.createSequentialGroup()
                    .addGap(35, 35, 35)
                    .addComponent(lblLedM2)
                    .addGap(0, 29, Short.MAX_VALUE)))
            .addContainerGap())
        );
        jInternalFrame2Layout.setVerticalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(jInternalFrame2Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                    .addGroup(jInternalFrame2Layout.createSequentialGroup()
                        .addComponent(lblLedM2,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(39, 39, 39)
                        .addComponent(txtPorcentajeM2))

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                        .addGroup(jInternalFrame2Layout.createSequentialGroup()

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                                .addComponent(btnParar1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(btnDerecha1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(btnIzquierda1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                            .addComponent(btnOnOffM2,
javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(slbControlVelM2,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)))
                .addContainerGap(14, Short.MAX_VALUE))
```

```
        );

        getContentPane().add(jInternalFrame2);
        jInternalFrame2.setBounds(330, 36, 302, 200);

        jInternalFrame3.setTitle("MOTOR 3");
        jInternalFrame3.setVisible(true);

        btnIzquierda2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnIzquierda.gi
f"))); // NOI18N
        btnIzquierda2.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIzquierda2ActionPerformed(evt);
            }
        });

        btnDerecha2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnDerecha.gif
"))); // NOI18N
        btnDerecha2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDerecha2ActionPerformed(evt);
            }
        });

        btnParar2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/parar.gif"))); //
NOI18N
        btnParar2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnParar2ActionPerformed(evt);
            }
        });

        btnOnOffM3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/onOff.gif"))); //
NOI18N
        btnOnOffM3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOnOffM3ActionPerformed(evt);
            }
        });

        slbControlVelM3.setOrientation(javax.swing.JSlider.VERTICAL);
        slbControlVelM3.setValue(0);
        slbControlVelM3.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
```

```
            slbControlVelM3StateChanged(evt);
        }
    });

    lblLedM3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif"))); //
NOI18N

    txtPorcentajeM3.setFont(new java.awt.Font("Tahoma", 1, 18)); //
NOI18N
    txtPorcentajeM3.setText("0%");

txtPorcentajeM3.setVerticalAlignment(javax.swing.SwingConstants.BOTT
OM);
    txtPorcentajeM3.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAI
SED));

txtPorcentajeM3.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);

    javax.swing.GroupLayout jInternalFrame3Layout = new
javax.swing.GroupLayout(jInternalFrame3.getContentPane());
    jInternalFrame3.getContentPane().setLayout(jInternalFrame3Layout);
    jInternalFrame3Layout.setHorizontalGroup(

jInternalFrame3Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
        .addGroup(jInternalFrame3Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jInternalFrame3Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                .addComponent(btnOnOffM3,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                .addGroup(jInternalFrame3Layout.createSequentialGroup()
                    .addComponent(btnIzquierda2,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addComponent(btnParar2,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
```

```
                    .addComponent(btnDerecha2,
javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addComponent(slbControlVelM3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jInternalFrame3Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                    .addGroup(jInternalFrame3Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addComponent(txtPorcentajeM3,
javax.swing.GroupLayout.DEFAULT_SIZE, 90, Short.MAX_VALUE))
                    .addGroup(jInternalFrame3Layout.createSequentialGroup()
                        .addGap(35, 35, 35)
                        .addComponent(lblLedM3)
                        .addGap(0, 29, Short.MAX_VALUE)))
                .addContainerGap())
        );
        jInternalFrame3Layout.setVerticalGroup(

jInternalFrame3Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(jInternalFrame3Layout.createSequentialGroup()
                .addGap(19, 19, 19)

.addGroup(jInternalFrame3Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                    .addGroup(jInternalFrame3Layout.createSequentialGroup()
                        .addComponent(lblLedM3,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(39, 39, 39)
                        .addComponent(txtPorcentajeM3))

.addGroup(jInternalFrame3Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                    .addGroup(jInternalFrame3Layout.createSequentialGroup()

.addGroup(jInternalFrame3Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                            .addComponent(btnParar2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                        .addComponent(btnDerecha2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(btnIzquierda2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addComponent(btnOnOffM3,
javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(slbControlVelM3,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)))
            .addContainerGap(14, Short.MAX_VALUE))
        );

        getContentPane().add(jInternalFrame3);
        jInternalFrame3.setBounds(10, 250, 302, 200);

        jInternalFrame4.setTitle("MOTOR 4");
        jInternalFrame4.setVisible(true);

        btnIzquierda3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnIzquierda.gi
f"))); // NOI18N
        btnIzquierda3.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnIzquierda3ActionPerformed(evt);
            }
        });

        btnDerecha3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/btnDerecha.gif
"))); // NOI18N
        btnDerecha3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDerecha3ActionPerformed(evt);
            }
        });

        btnParar3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/parar.gif"))); //
NOI18N
        btnParar3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnParar3ActionPerformed(evt);
            }
        });
```

```
        btnOnOffM4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/onOff.gif"))); //
NOI18N
        btnOnOffM4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOnOffM4ActionPerformed(evt);
            }
        });

        slbControlVelM4.setOrientation(javax.swing.JSlider.VERTICAL);
        slbControlVelM4.setValue(0);
        slbControlVelM4.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                slbControlVelM4StateChanged(evt);
            }
        });

        lblLedM4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif"))); //
NOI18N

        txtPorcentajeM4.setFont(new java.awt.Font("Tahoma", 1, 18)); //
NOI18N
        txtPorcentajeM4.setText("0%");

txtPorcentajeM4.setVerticalAlignment(javax.swing.SwingConstants.BOTT
OM);
        txtPorcentajeM4.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAI
SED));

txtPorcentajeM4.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);

        javax.swing.GroupLayout jInternalFrame4Layout = new
javax.swing.GroupLayout(jInternalFrame4.getContentPane());
        jInternalFrame4.getContentPane().setLayout(jInternalFrame4Layout);
        jInternalFrame4Layout.setHorizontalGroup(

jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(jInternalFrame4Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                    .addComponent(btnOnOffM4,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                    .addGroup(jInternalFrame4Layout.createSequentialGroup()
```

```
                .addComponent(btnIzquierda3,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                .addComponent(btnParar3,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                .addComponent(btnDerecha3,
javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGap(18, 18, 18)
            .addComponent(slbControlVelM4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                .addGroup(jInternalFrame4Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                .addComponent(txtPorcentajeM4,
javax.swing.GroupLayout.DEFAULT_SIZE, 90, Short.MAX_VALUE))
                .addGroup(jInternalFrame4Layout.createSequentialGroup()
                .addGap(35, 35, 35)
                .addComponent(lblLedM4)
                .addGap(0, 29, Short.MAX_VALUE)))
            .addContainerGap())
      );
      jInternalFrame4Layout.setVerticalGroup(

jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
          .addGroup(jInternalFrame4Layout.createSequentialGroup()
            .addGap(19, 19, 19)

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                .addGroup(jInternalFrame4Layout.createSequentialGroup()
                .addComponent(lblLedM4,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(39, 39, 39)
                .addComponent(txtPorcentajeM4))
```

```
.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                    .addGroup(jInternalFrame4Layout.createSequentialGroup()

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING, false)
                        .addComponent(btnParar3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(btnDerecha3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(btnIzquierda3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                      .addComponent(btnOnOffM4,
javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(slbControlVelM4,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        getContentPane().add(jInternalFrame4);
        jInternalFrame4.setBounds(330, 250, 302, 200);
    }// </editor-fold>

    private void btnIzquierdaActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 11;
        byte comandoLow = 01;
        objMotor1.setComandoHigh(comandoHigh);
        objMotor1.setComandoLow(comandoLow);
    }

    private void btnPararActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        byte comandoHigh = 101;
        byte comandoLow = 101;
        objMotor1.setComandoHigh(comandoHigh);
        objMotor1.setComandoLow(comandoLow);
    }

    private void btnDerechaActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 10;
```

```
        byte comandoLow = 00;
        objMotor1.setComandoHigh(comandoHigh);
        objMotor1.setComandoLow(comandoLow);
    }

    private void btnOnOffM1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        if(btnOnOffM1.isSelected()){
            byte comandoHigh = 101;
            byte comandoLow = 101;
            objMotor1.setComandoHigh(comandoHigh);
            objMotor1.setComandoLow(comandoLow);
            objMotor1.setOnOff("On");
            objMotor1.controlMotores();
            lblLedM1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led.gif")));
        }else{
            byte comandoHigh = 101;
            byte comandoLow = 101;
            objMotor1.setComandoHigh(comandoHigh);
            objMotor1.setComandoLow(comandoLow);
            objMotor1.setOnOff("Off");
            objMotor1.controlMotores();
            lblLedM1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif")));
        }
    }

    private void
slbControlVelM1StateChanged(javax.swing.event.ChangeEvent evt) {
        // TODO add your handling code here:
        int controlVel = slbControlVelM1.getValue();
        objMotor1.setControlVel(controlVel);
        txtPorcentajeM1.setText(""+slbControlVelM1.getValue()+"%");
    }

    private void btnIzquierda1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 13;
        byte comandoLow = 03;
        objMotor2.setComandoHigh(comandoHigh);
        objMotor2.setComandoLow(comandoLow);
    }

    private void btnDerecha1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 12;
```

```
        byte comandoLow = 02;
        objMotor2.setComandoHigh(comandoHigh);
        objMotor2.setComandoLow(comandoLow);
    }

    private void btnParar1ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        byte comandoHigh = 023;
        byte comandoLow = 023;
        objMotor2.setComandoHigh(comandoHigh);
        objMotor2.setComandoLow(comandoLow);
    }

    private void btnOnOffM2ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        if(btnOnOffM2.isSelected()){
            byte comandoHigh = 023;
            byte comandoLow = 023;
            objMotor2.setComandoHigh(comandoHigh);
            objMotor2.setComandoLow(comandoLow);
            objMotor2.setOnOff("On");
            objMotor2.controlMotores();
            lblLedM2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led.gif")));
        }else{
            byte comandoHigh = 023;
            byte comandoLow = 023;
            objMotor2.setComandoHigh(comandoHigh);
            objMotor2.setComandoLow(comandoLow);
            objMotor2.setOnOff("Off");
            objMotor2.controlMotores();
            lblLedM2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif")));
        }
    }

    private void
slbControlVelM2StateChanged(javax.swing.event.ChangeEvent evt) {
        // TODO add your handling code here:
        int controlVel = slbControlVelM2.getValue();
        objMotor2.setControlVel(controlVel);
        txtPorcentajeM2.setText(""+slbControlVelM2.getValue()+"%");
    }

    private void btnIzquierda2ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 15;
```

```
        byte comandoLow = 05;
        objMotor3.setComandoHigh(comandoHigh);
        objMotor3.setComandoLow(comandoLow);
    }

    private void btnDerecha2ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 14;
        byte comandoLow = 04;
        objMotor3.setComandoHigh(comandoHigh);
        objMotor3.setComandoLow(comandoLow);
    }

    private void btnParar2ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        byte comandoHigh = 045;
        byte comandoLow = 045;
        objMotor3.setComandoHigh(comandoHigh);
        objMotor3.setComandoLow(comandoLow);
    }

    private void btnOnOffM3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        if(btnOnOffM3.isSelected()){
            byte comandoHigh = 045;
            byte comandoLow = 045;
            objMotor3.setComandoHigh(comandoHigh);
            objMotor3.setComandoLow(comandoLow);
            objMotor3.setOnOff("On");
            objMotor3.controlMotores();
            lblLedM3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led.gif")));
        }else{
            byte comandoHigh = 045;
            byte comandoLow = 045;
            objMotor3.setComandoHigh(comandoHigh);
            objMotor3.setComandoLow(comandoLow);
            objMotor3.setOnOff("Off");
            objMotor3.controlMotores();
            lblLedM3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif")));
        }
    }

    private void
slbControlVelM3StateChanged(javax.swing.event.ChangeEvent evt) {
        // TODO add your handling code here:
```

```
        int controlVel = slbControlVelM3.getValue();
        objMotor3.setControlVel(controlVel);
        txtPorcentajeM3.setText(""+slbControlVelM3.getValue()+"%");
    }

    private void btnIzquierda3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 17;
        byte comandoLow = 07;
        objMotor4.setComandoHigh(comandoHigh);
        objMotor4.setComandoLow(comandoLow);
    }

    private void btnDerecha3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        byte comandoHigh = 16;
        byte comandoLow = 06;
        objMotor4.setComandoHigh(comandoHigh);
        objMotor4.setComandoLow(comandoLow);
    }

    private void btnParar3ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        byte comandoHigh = 067;
        byte comandoLow = 067;
        objMotor4.setComandoHigh(comandoHigh);
        objMotor4.setComandoLow(comandoLow);
    }

    private void btnOnOffM4ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        if(btnOnOffM4.isSelected()){
            byte comandoHigh = 067;
            byte comandoLow = 067;
            objMotor4.setComandoHigh(comandoHigh);
            objMotor4.setComandoLow(comandoLow);
            objMotor4.setOnOff("On");
            objMotor4.controlMotores();
            lblLedM4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led.gif")));
        }else{
            byte comandoHigh = 067;
            byte comandoLow = 067;
            objMotor4.setComandoHigh(comandoHigh);
            objMotor4.setComandoLow(comandoLow);
            objMotor4.setOnOff("Off");
```

```
            objMotor4.controlMotores();
            lblLedM4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/led1.gif")));
        }
    }

    private void
slbControlVelM4StateChanged(javax.swing.event.ChangeEvent evt) {
        // TODO add your handling code here:
        int controlVel = slbControlVelM4.getValue();
        objMotor4.setControlVel(controlVel);
        txtPorcentajeM4.setText(""+slbControlVelM4.getValue()+"%");
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnDerecha;
    private javax.swing.JButton btnDerecha1;
    private javax.swing.JButton btnDerecha2;
    private javax.swing.JButton btnDerecha3;
    private javax.swing.JButton btnIzquierda;
    private javax.swing.JButton btnIzquierda1;
    private javax.swing.JButton btnIzquierda2;
    private javax.swing.JButton btnIzquierda3;
    private javax.swing.JToggleButton btnOnOffM1;
    private javax.swing.JToggleButton btnOnOffM2;
    private javax.swing.JToggleButton btnOnOffM3;
    private javax.swing.JToggleButton btnOnOffM4;
    private javax.swing.JButton btnParar;
    private javax.swing.JButton btnParar1;
    private javax.swing.JButton btnParar2;
    private javax.swing.JButton btnParar3;
    private javax.swing.JInternalFrame jInternalFrame1;
    private javax.swing.JInternalFrame jInternalFrame2;
    private javax.swing.JInternalFrame jInternalFrame3;
    private javax.swing.JInternalFrame jInternalFrame4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JToolBar jToolBar1;
    private javax.swing.JLabel lblLedM1;
    private javax.swing.JLabel lblLedM2;
    private javax.swing.JLabel lblLedM3;
    private javax.swing.JLabel lblLedM4;
    private javax.swing.JSlider slbControlVelM1;
    private javax.swing.JSlider slbControlVelM2;
    private javax.swing.JSlider slbControlVelM3;
    private javax.swing.JSlider slbControlVelM4;
    private javax.swing.JLabel txtPorcentajeM1;
    private javax.swing.JLabel txtPorcentajeM2;
    private javax.swing.JLabel txtPorcentajeM3;
    private javax.swing.JLabel txtPorcentajeM4;
    // End of variables declaration
```

}

d. Código fuente del microcontrolador

```
#include<18F4550.h>
#fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,V
REGEN,XT
#use delay(clock=48000000)
#USE STANDARD_IO(b)


#define USB_HID_DEVICE FALSE // deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE USB_ENABLE_BULK // turn on EP1(EndPoint1) for IN bulk/interrupt transfers
#define USB_EP1_RX_ENABLE USB_ENABLE_BULK // turn on EP1(EndPoint1) for OUT bulk/interrupt transfers
#define USB_EP1_TX_SIZE 32 // size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE 32 // size to allocate for the rx endpoint 1 buffer


#include <pic18_usb.h> // Microchip PIC18Fxx5x Hardware layer for CCS's PIC USB driver
#include "header.h" // Configuración del USB y los descriptores para este dispositivo
#include <usb.c> // handles usb setup tokens and get descriptor reports

#define Enciende Output_High
#define Apaga Output_Low
#define Conmuta Output_Toggle
#define RecCommand recbuf[0]
#define LedParam recbuf[1]
#define COMMAND_FIRMWARE 99
#define COMMAND_LEDS 88

const int8 Lenbuf = 32;

////////////////////////////////////////////////////////////////////////////////
//
// RAM, RAM, RAM
//
////////////////////////////////////////////////////////////////////////////////
```

```
char Version[] = "v.1.0";
int8 recbuf[Lenbuf];
int8 sndbuf[Lenbuf];

////////////////////////////////////////////////////////////////////////////////////////////////
//
// M A I N
//
////////////////////////////////////////////////////////////////////////////////////////////////
void main(void) {
  delay_ms(500);
  usb_init();
  usb_task();
  usb_wait_for_enumeration();
  enable_interrupts(global);
  while (TRUE){
    if(usb_enumerated()){
      if (usb_kbhit(1)){
        usb_get_packet(1, recbuf, Lenbuf);
        switch(RecCommand){
          case 00:
            Output_Low(PIN_B0);
          break;
          case 10:
            Output_High(PIN_B0);
          break;
          case 01:
            Output_Low(PIN_B1);
          break;
          case 11:
            Output_High(PIN_B1);
          break;
          case 101:
            Output_Low(PIN_B0);
            Output_Low(PIN_B1);
          break;
          case 02:
            Output_Low(PIN_B2);
          break;
          case 12:
            Output_High(PIN_B2);
          break;
          case 03:
            Output_Low(PIN_B3);
```

```
        break;
        case 13:
          Output_High(PIN_B3);
        break;
        case 023:
         Output_Low(PIN_B2);
         Output_Low(PIN_B3);
        break;
        case 04:
          Output_Low(PIN_B4);
        break;
        case 14:
          Output_High(PIN_B4);
        break;
        case 05:
          Output_Low(PIN_B5);
        break;
        case 15:
          Output_High(PIN_B5);
        break;
        case 045:
         Output_Low(PIN_B4);
         Output_Low(PIN_B5);
        break;
        case 06:
          Output_Low(PIN_B6);
        break;
        case 16:
          Output_High(PIN_B6);
        break;
        case 07:
          Output_Low(PIN_B7);
        break;
        case 17:
          Output_High(PIN_B7);
        break;
        case 067:
         Output_Low(PIN_B6);
         Output_Low(PIN_B7);
        break;
      }
    }
  }
}
```

```
}
```