DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMA EMBEBIDO BASADO EN PIC18

GARY BALDRICH PINEDO
GERMAÍN DÍAZ LÓPEZ

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
PROGRAMA DE INGENIERIA ELECTRÓNICA
CARTAGENA DE INDIAS, D.T. Y C.
JULIO DE 2011

DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMA EMBEBIDO BASADO EN PIC18

GARY BALDRICH PINEDO
GERMAÍN DÍAZ LÓPEZ

TRABAJO PARA OPTAR AL TÍTULO DE
INGENIERO ELECTRÓNICO

DIRECTOR
ING. LUIS ACOSTA GALVÁN

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
PROGRAMA DE INGENIERIA ELECTRÓNICA
CARTAGENA DE INDIAS, D.T. Y C.
JULIO DE 2011

ARTICULO 107

La Universidad Tecnológica de Bolívar se reserva el derecho de propiedad de los trabajos de grado aprobados y no pueden ser explotados comercialmente sin autorización.

Cartagena D. T. Y C., Julio 5 de 2011

Señores,

Programa de Ingeniería Electrónica
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
La ciudad

Cordial saludo:

A través de la presente me permito entregar a la Universidad Tecnológica de Bolívar el trabajo titulado **DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMA EMBEBIDO BASADO EN PIC18** realizada por los estudiantes GARY BALDRICH PINEDO y GERMAÍN DÍAZ LÓPEZ.

Atentamente,

Ing. Luis Acosta Galván

Cartagena D. T. Y C., Julio 5 de 2011

Señores,
Programa de Ingeniería Electrónica
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
La ciudad

Yo Gary Baldrich Pinedo, manifiesto en este documento mi voluntad de ceder a la Universidad Tecnológica de Bolívar los derechos patrimoniales, consagrados en el artículo 72 de la Ley 23 de 1982 sobre Derechos de Autor, del trabajo final denominado **DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMAS EMBEBIDO BASADO EN PIC18** producto de mi actividad académica para optar al título de Ingeniero Electrónico de la Universidad Tecnológica de Bolívar.

La Universidad Tecnológica de Bolívar, entidad académica sin ánimo de lucro, queda por lo tanto facultada para ejercer plenamente los derechos anteriormente cedidos en su actividad ordinaria de investigación, docencia y extensión. La cesión otorgada se ajusta a lo que establece la Ley 23 de 1982. Con todo, en mi condición de autor me reservo los derechos morales la obra antes citada con arreglo al artículo 30 de la Ley 23 de 1982. En concordancia suscribo este documento que hace parte integral del trabajo antes mencionado y entrego al sistema de bibliotecas de la Universidad Tecnológica de Bolívar.

73'203.644 Julio-05-2011

Gary Baldrich Pinedo
c.c. 73203644 de Cartagena

REPUBLICA DE COLOMBIA
MARIELA URIBE CARDONA
Notaria Septima Encargada
Cartagena

NOTARIA SÉPTIMA
DEL CIRCULO DE CARTAGENA
ANTE LA SUSCRITA NOTARIA, Compareció
Gary Baldrich Pinedo
C.C. 73.203.644 DE Cgena
y dijo que reconoce como suya la firma estampada en el anterior documento.
Así como el contenido del mismo.
73'203644 Cgena
Fecha: 05 JUL. 2011

Cartagena D. T. Y C., Julio 5 de 2011

Señores,
Programa de Ingeniería Electrónica
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
La ciudad

Yo Germaín Díaz López , manifiesto en este documento mi voluntad de ceder a la Universidad Tecnológica de Bolívar los derechos patrimoniales, consagrados en el artículo 72 de la Ley 23 de 1982 sobre Derechos de Autor, del trabajo final denominado **DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMAS EMBEBIDO BASADO EN PIC18** producto de mi actividad académica para optar al título de Ingeniero Electrónico de la Universidad Tecnológica de Bolívar.

La Universidad Tecnológica de Bolívar, entidad académica sin ánimo de lucro, queda por lo tanto facultada para ejercer plenamente los derechos anteriormente cedidos en su actividad ordinaria de investigación, docencia y extensión. La cesión otorgada se ajusta a lo que establece la Ley 23 de 1982. Con todo, en mi condición de autor me reservo los derechos morales la obra antes citada con arreglo al artículo 30 de la Ley 23 de 1982. En concordancia suscribo este documento que hace parte integral del trabajo antes mencionado y entrego al sistema de bibliotecas de la Universidad Tecnológica de Bolívar.

73007102 Julio 5 de 2011

Germaín Díaz López
c.c. 73007102 de Cartagena

REPUBLICA DE COLOMBIA
MARIELA URIBE CARDONA
Notaria Septima Encargada
Cartagena

NOTARIA SÉPTIMA
DEL CIRCULO DE CARTAGENA
ANTE LA SUSCRITA NOTARIA, Compareció
Germaín José Díaz López.
C.C. 73007102.
DE Ctgna
y dijo que reconoce como suya la firma estampada en el anterior documento Así como el contenido del mismo.
HUELLA
Fecha: 05 JUL 2011

AGRADECIMIENTOS

Este trabajo de grado que hoy encontró término
en medio de un camino de tantas emociones que son fáciles de inferir, he encontrado enseñanzas que van mucho más allá del saber hacer de la disciplina que escogí.

Aprendí con Dios, como mi Padre, que siempre estuvo, está y estará siempre con migo a pesar de mis faltas.

Aprendí con mi abuela Ana, que la razón humana más digna para entregar la vida es la familia.

Aprendí con mi mamá Casilda, que los proyectos que se emprenden no afectan solo a su ejecutor, sino a todas las personas que están en derredor.

Aprendí con mi papá Freddys, que nunca es tarde para cambiar el rumbo de la vida y corregir o por lo menos intentar reparar las consecuencias de las malas decisiones del pasado.

Con mi hermano Darwin, que es posible alcanzar los sueños no importa cuán lejos estos se vean.

Con mi novia Luci, que la llave de la victoria está en persistir, persistir y persistir delante de Dios y nunca rendirse… dicho con otras palabras.

Con mis pastores Henry y Yaneth, que todo lo que un cristiano es, tiene y vive proviene de dos cosas Palabra de Dios y Oración.

Aprendí con mi amigo Luis Falquez, que si hay gente capaz de dar arriesgar mucho por un amigo.

Aprendí con mi amigo y colega Edgar villa, que ser hermanos en Cristo no es solo un título bonito, es una vivencia.

Aprendí con mi amigo Jory, que si hay personas que ayudan desinteresadamente al punto de tomarse a pecho las necesidades de otro como si fueran de sí mismos.

Emprendo hoy mi camino como profesional con este orden: primero Dios, luego mi familia, mi servicio a Dios, mis amigos y entonces mi trabajo.

Gary Baldrich Pinedo

AGRADECIMIENTOS

En primer lugar, quiero agradecerle a Dios por haberme orientado sabia y espiritualmente en la consecución de este proyecto.

A mis Padres por su dedicación, apoyo y sacrificios que me ayudaron a convertirme en un profesional de la ingeniería y de la vida.

A todas las personas que su dedicación y sabiduría ayudaron a lograr que este proyecto se llevara a cabo. En especial mi compañero el Ingeniero Edgar Villa.

Germaín Díaz López

RESUMEN

Este documento llamado **DISEÑO E IMPLEMENTACION DE UN SERVOMOTOR MEDIANTE UN SISTEMA EMBEBIDO BASADO EN PIC18**, se encuentra organizado en tres secciones: fundamentación teórica, diseño y validación. En la primera, damos una descripción detallada de todos los conceptos y elementos manejados en el proyecto, la segunda registra ampliamente todos los cálculos, análisis que hicieron parte del diseño del prototipo; la tercera y última sección contiene todas las pruebas, análisis de resultados y ajustes necesarios en la implementación del mismo.

El objetivo de este proyecto es diseñar e implementar un sistema servomotor controlado en velocidad mediante un microcontrolador de la serie PIC18 con el fin de poner a prueba su desempeño en sistemas robustos de control similares a los encontrados en la industria, realizando tareas simultaneas como control digital, y comunicación bidireccional mediante protocolo USB con un PC para efectos de monitoreo y configuración.

Para el control de sentido giro se diseñó un Half Bridge con Mosfet's capaz de soportar la corriente demandada por el motor. Las funciones de la etapa de control básicamente son realizar el control PID y establecer la comunicación bidireccional con la PC, para lo cual se decidió usar un el microcontrolador PIC18F4550 por ser el dispositivo de más fácil y de económica adquisición que maneja comunicación USB y tiene los periféricos necesarios para la interacción con este proyecto; permitiendo a su vez un futuro ensanchamiento del mismo. La programación para este microcontrolador se desarrolló en el lenguaje CCS C debido a su carácter intuitivo, además de ser un lenguaje de alto nivel que abstrae muchas porciones de código en ocasiones causantes de retrasos por falta de conocimiento del programador de los detalles más pequeños de la arquitectura del

microcontrolador. Para efecto de monitoreo y modificación de parámetros del sistema, se realizó una aplicación en Java capaz de gestionar una comunicación USB entre el PIC y la PC, monitorear la variable a controlar y modificar variables como el setpoint, constantes PID, sentido de giro y tipo control (*lazo abierto/cerrado*), permitiendo compactar en una sencilla aplicación las funciones necesarias a la medida de este proyecto.

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Este proyecto nace con la necesidad de introducir el uso de sistemas embebidos basados en microcontroladores en el control de sistemas robustos como los usados en la industria. Para este proyecto se ha seleccionado el control de un motor DC, por ser uno de los elementos de uso más frecuente y que está presente en la mayoría de los lazos de control usados en la industria siempre que la potencia requerida sea inferior a 50HP.

Este documento de tesis se encuentra organizado en tres secciones: fundamentación teórica, diseño y validación. En cuanto al primer ítem se estará haciendo un acercamiento a los conceptos principales que se manejarán a lo largo del documento. Entrando en la sección de diseño se describirán detalladamente los procedimientos matemáticos usados para la creación del Driver o Half Bridge, la programación del dispositivo de control y la caracterización de la planta. Finalmente será necesario hacer la validación respectiva a través de pruebas y experimentos que nos lleven a evaluar de manera objetiva si se cumplieron los propósitos del diseño mediante la presentación de resultados en tablas y gráficas.

El objetivo de este proyecto es implementar un sistema servomotor controlado en velocidad mediante un microcontrolador de la serie PIC18 con el fin de poner a prueba su desempeño en sistemas robustos de control similares a los encontrados en la industria, realizando tareas simultaneas como control digital, y comunicación bidireccional mediante protocolo USB con un PC para efectos de monitoreo y configuración.

En este documento se describe detalladamente la fundamentación teórica que respalda el trabajo realizado como el proceso de diseño paso a paso para la elaboración de cada fase y al final se presentan los resultados que validan el

cumplimiento de los objetivos planteados.

Una visión general del proyecto se puede apreciar en la Figura 1.

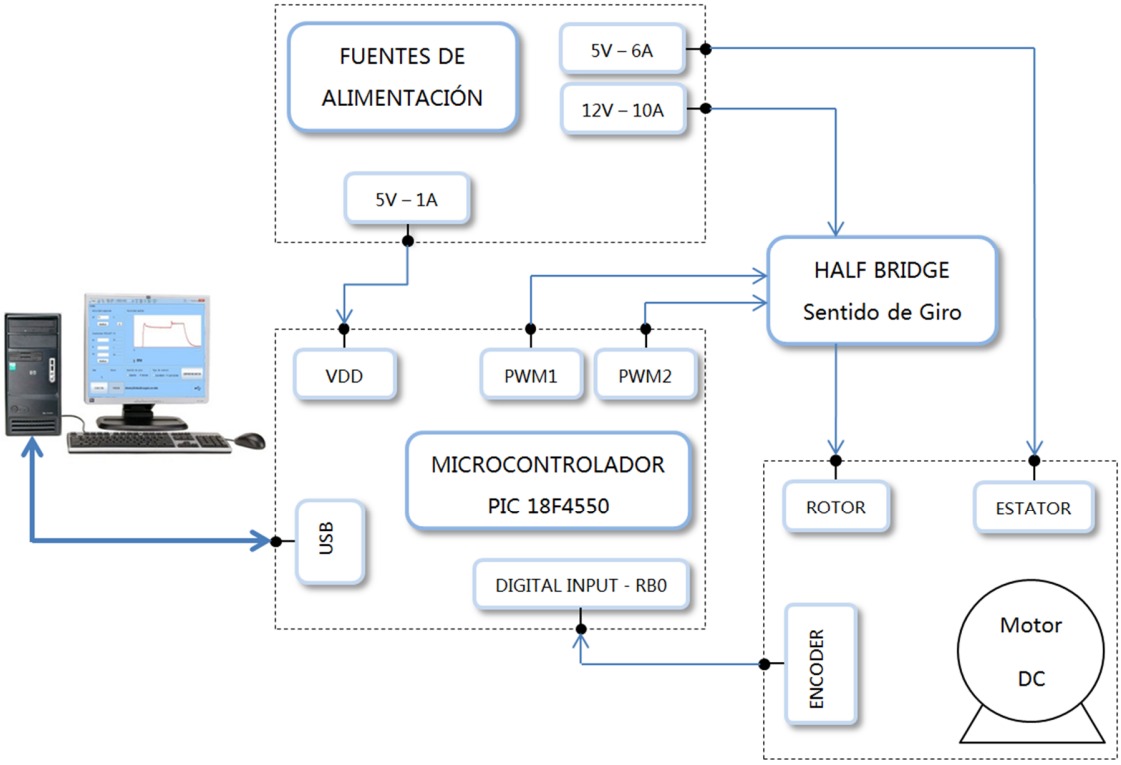


Figura 1. Diagrama de bloques general

1. FUNDAMENTACIÓN TEÓRICA

1.1 Servomotor

Un servomecanismo, o servo es un dispositivo automático que usa error de realimentación para controlar el funcionamiento del mecanismo. El término se aplica correctamente solo a sistemas donde la realimentación o las señales de corrección de error ayudan a controlar posición mecánica u otros parámetros.

Existen diferentes tipos de servomecanismos: están los hidráulicos, neumáticos, magnéticos y para el caso específico de este trabajo los eléctricos y parcialmente electrónicos los cuales usan un motor eléctrico como fuente primaria para crear fuerza mecánica.

Un servomotor es entonces un motor DC el cual por medio de un sistema de realimentación puede realizar un control de posición y/o velocidad sobre el eje de rotación con una muy buena precisión. En la actualidad es tan alta la variedad de servomotores como las aplicaciones en muchos ámbitos tomando como los principales la industria y la robótica.

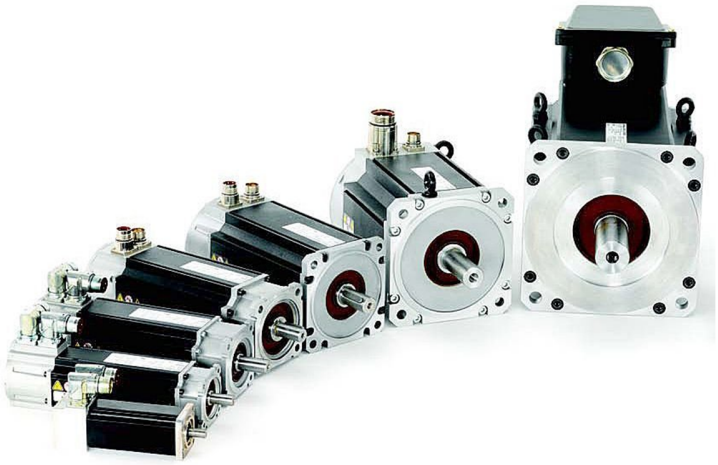


Figura 2. Servomotores de aplicaciones industriales y de equipos especializados

1.2 Sistema embebido

Un **sistema embebido o empotrado** es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas. Aunque básicamente cumple la misma función de una computadora personal, un sistema embebido es una mejor opción en cuanto a que su capacidad de procesamiento está a la medida de la aplicación que maneja reduciendo costos y mejorando el mantenimiento del sistema.

Algunas de las características de un sistema embebido son:

Poseen una CPU o unidad que aporta capacidad de cómputo al sistema esta puede ser un microprocesador, microcontrolador, DSP, etc.

Un sistema de comunicación alámbrica o inalámbrica que según las necesidades y las características de la aplicación puede ser mediante cualquier puerto de comunicaciones como: RS-232, RS-485, SPI, I^2C, CAN, USB, ETHERNET, Wi-Fi, GSM, GPRS, DSRC, etc.

Algunos cuentan con un subsistema de monitoreo mediante una pantalla gráfica, táctil, LCD, alfanumérico, etc.

Posee también actuadores o elementos electrónicos que el sistema se encarga de controlar. Puede ser un motor eléctrico, un conmutador tipo relé etc. El más habitual puede ser una salida de señal PWM para control de la velocidad en motores de corriente continua.

Un módulo de E/S analógicas y digitales mediante las cuales pueda interactuar con el resto del sistema leyendo señales analógicas procedentes de sensores,

activando diodos LED, reconociendo el estado abierto cerrado de un conmutador o pulsador, etc.

Un módulo de reloj encargado de generar las diferentes señales que permitirán que el sistema este sincronizado, para lograr una frecuencia de operación adecuada y para que el sistema tenga la estabilidad necesaria.

El módulo de energía se encarga de generar las diferentes tensiones y corrientes necesarias para alimentar los diferentes circuitos del SE. Usualmente se trabaja con un rango de posibles tensiones de entrada que mediante conversores ac/dc o dc/dc se obtienen las diferentes tensiones necesarias para alimentar los diversos componentes activos del circuito.

Conversores ac/dc y dc/dc, para interactuar con variables analógicas del sistema.

1.3 Microcontrolador

Un microcontrolador es un circuito integrado de bajo costo capaz de llevar a cabo procesos lógicos que pueden ser programados por un usuario. Es manufacturado especialmente para aplicaciones de control se sistemas computacionales embebidos y aplicaciones de control digital.

Los microcontroladores están normalmente constituidos por módulos de circuitos internos necesarios para aplicaciones de control computacional, tales como: Procesador o CPU (Unidad Central de Procesamiento). Memoria RAM para contener los datos. Memoria para el programa tipo ROM/PROM/EPROM. Líneas de E/S para comunicarse con el exterior. Diversos módulos para el control de periféricos (temporizadores, Puertos Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.). Generador de pulsos de reloj que sincronizan el funcionamiento de todo el sistema

Estos dispositivos ofrecen las siguientes ventajas: Aumento de prestaciones, aumento de la fiabilidad al reemplazar el microcontrolador por un elevado número de elementos disminuyendo el riesgo de averías, reducción del tamaño en el producto acabado, debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna.

Figura 3. Microcontrolador PIC

1.4 Compilador

Un compilador es un programa informático que traduce un código escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación.

1.5 Bootloader

Un bootloader es un conjunto de instrucciones que se le cargan a un microcontrolador en sus primeras posiciones de memoria y su función es administrar la sección de memoria donde se carga el programa que se quiere ejecutar. Un bootloader maneja dos modos de operación: ejecución y boot los cuales se seleccionan a través de un par de pulsadores uno de reset y otro de boot. Si al energizar el microcontrolador el pulsador de boot está presionado

entonces este entrara en modo boot abriendo la posibilidad de borrar o cargar el programa a ejecutar en él, pero si no lo encuentra presionado o se oprime reset el microcontrolador entrará a ejecutar el programa que tenga en la posición de memoria que el bootloader haya establecido. Esto brinda la posibilidad de hacer modificaciones al programa sin necesidad de retirar el microcontrolador de la aplicación ahorrando tiempo al programador y evitando posibles daños al dispositivo por excesiva manipulación.

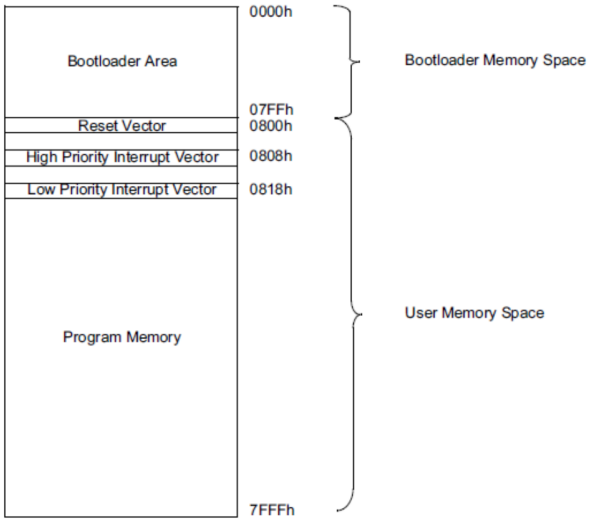


Figura 4. Distribución de memoria de programa de un microcontrolador con Bootloader

1.6 Motor DC de excitación independiente

Como su nombre lo indica, estos motores dc tienen el rotor y el estator separados. Estos obtienen la alimentación del rotor y del estator de dos fuentes de tensión independientes. Con esto, el campo del estator es constante al no depender de la carga del motor, y el par de fuerza es entonces prácticamente constante.

Para la aplicación en cuestión se pretende fijar un valor de tensión en el estator que produzca un par de fuerza constante y mediante otra fuente de tensión variable aplicada al rotor se ajusta la velocidad deseada.

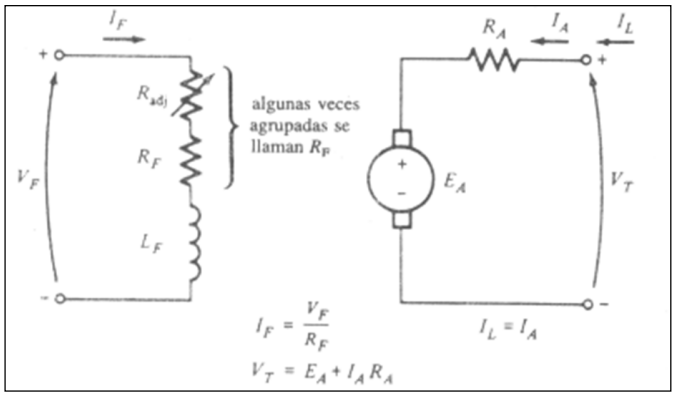


Figura 5. Diagrama de un motor de excitación independiente

1.7 Encoder

Es un dispositivo capaz de codificar un desplazamiento o velocidad en una salida discreta. En su configuración más básica consta de un disco transparente con una serie de marcas opacas colocadas radialmente y equidistantes entre sí las cuales codifican el disco. Existen de diferentes materiales y funcionan con principios distintos como detección por paso de luz, magnetismo entre otros.

Para nuestro caso, existe un LED emisor de luz y otro receptor de forma que el LED emisor está colocado tal que el haz de luz pueda atravesar el disco codificado y el receptor recibir el haz de luz.

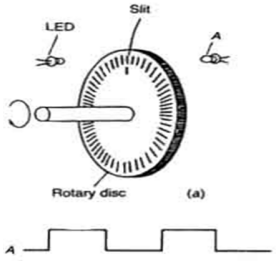


Figura 6. Encoder incremental

El disco se coloca en el eje que se quiera medir, a medida que el eje gire se

generarán una serie de pulsos. Cada vez que la luz atraviese cada marca del disco, y llevando una cuenta de estos pulsos es posible conocer la posición del eje y su velocidad de rotación. Algunos discos tienen una marca especial llamada marca de cero la cual su misión principal es la de tener constancia que se ha dado una vuelta completa y que, por tanto hay que empezar una cuenta de nuevo.

1.8 Modulación por ancho de pulso PWM

La **modulación por anchura de pulsos** es una técnica de modulación en la que se modifica el **ciclo de trabajo** de una señal periódica (por ejemplo sinusoidal) para portar información.

El **ciclo de trabajo** de una señal periódica es el ancho relativo de su parte positiva en relación al período. Matemáticamente:

$$D = \frac{T_{ON}}{T}$$

Esta técnica es utilizada para controlar dispositivos, o para proveer un voltaje variable de corriente continua. La señal generada tendrá frecuencia fija y tiempos de encendido y apagado variables. En otras palabras, el período de la señal se mantendrá constante, pero la cantidad de tiempo que se mantiene en alto y bajo dentro de un período puede variar.

El ciclo de trabajo del total del período ($T = t_{On} + t_{Off}$) es t_{On}, es decir el tiempo que se mantiene en alto.

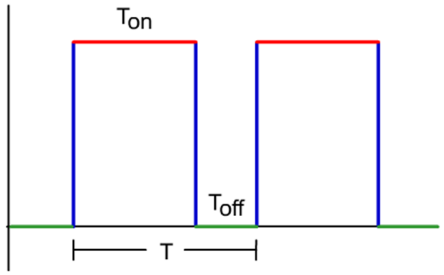


Figura 7. Modulación PWM

Algunas aplicaciones en las que se utiliza PWM son controles de motores, de iluminación y de temperatura, siendo las más comunes el control de servomotores y como sistema de comunicación.

En el caso de control de servomotores el funcionamiento es como sigue: la velocidad de rotación del motor será función de la anchura del pulso. Dado que la velocidad de rotación depende del valor medio de la tensión suministrada, cuanto más ancho sea el pulso mayor será el valor medio de la tensión aplicada al motor y por ende mayor velocidad de rotación. En caso de querer disminuir la velocidad de este, solo debemos de aplicar pulsos más estrechos los cuales nos darán como resultado un valor medio neto de tensión aplicada al motor menor que en el caso anterior.

1.9 Puente Half-Bridge (Driver)

Un Puente H o Half Bridge en H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos. Son ampliamente usados en robótica y como convertidores de potencia. Los puentes H están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos.

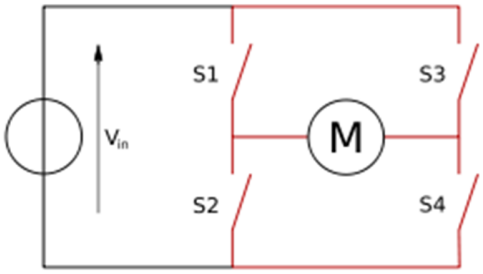


Figura 8. Representación de un Puente H

Un puente H consta de cuatro interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido.

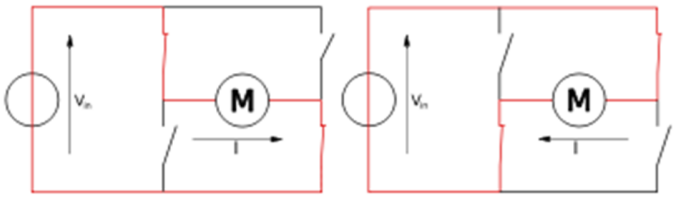


Figura 9. Funcionamiento de un Puente H

Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Además los dispositivos que sean usados para la conmutación de este sistema deben ser capaces de manejar las condiciones más extremas de demanda de potencia.

1.10 Implementación de algoritmos discretos de control

Una función de transferencia de un controlador discreto se puede desarrollar de diferentes maneras las cuales son matemáticamente equivalentes. Estas solo se diferencian en la forma en que son implementadas, ya que dependiendo de cuál se aplique se obtendrán distintas eficiencias computacionales, sensibilidades a errores y diferentes niveles de complejidad a la hora de la programación.

Una función de transferencia que tiene la forma de polinomios de transformada Z se debe implementar en un algoritmo de control que contenga retardos, multiplicadores y sumadores.

Una de estas formas de implementación es la estructura en paralelo la cual es menos sensible a errores y consume menos recurso computacional. En esta estructura la función de transferencia se describe matemáticamente como la suma de funciones de transferencia de primer y segundo orden como se muestra a continuación:

$$D(z) = a_0 + D_1 + D_2 + \ldots + D_m, 1 < m < n$$

Las funciones de transferencia de segundo orden son de la forma:

$$D_2(z) = \frac{a_1 + a_2 z^{-1}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

Los cuales se representan en la forma de estructura en paralelo como se muestra a continuación

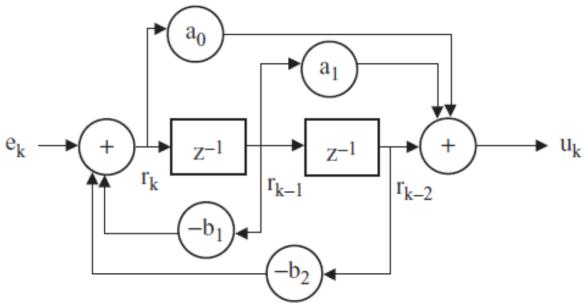
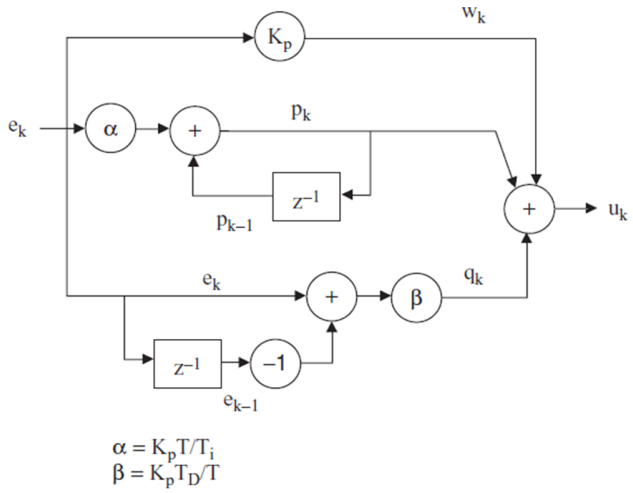


Figura 10. Estructura en paralelo para representación de sistemas discretos

Dado que la función de transferencia expresada en transformada Z de un controlador PID es

$$D(z) = K_p + \frac{K_p T}{T_i(1 - z^{-1})} + \frac{K_p T_d(1 - z^{-1})}{T}$$

Esta función pude ser implementada como una estructura en paralelo sumando sus constantes proporcional, integral y derivativa como se ve



$$p_k = \alpha e_k + p_{k-1};$$

$$q_k = \beta(e_k - e_{k-1}).$$

$$u_k = w_k + p_k + q_k.$$

$$w_k = K_p e_k.$$

Figura 11. Estructura en paralelo de un controlador PID

1.11 Control PID

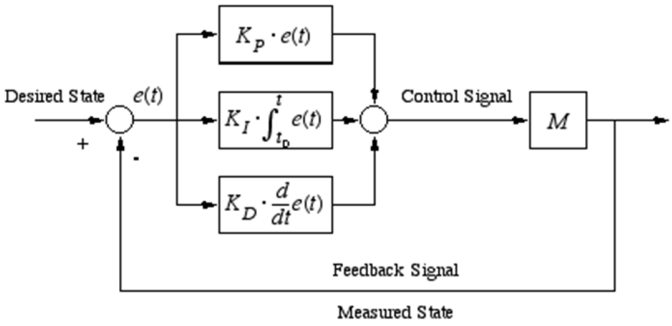


Figura 12. Diagrama de bloques de un controlador PID

Un controlador **PID** (**Proporcional Integral Derivativo**) es un sistema de control que, mediante un actuador, es capaz de mantener una variable o proceso en un punto deseado dentro del rango de medición del sensor que la mide. Es uno de los métodos de control más frecuentes y precisos dentro de la regulación automática.

Para el correcto funcionamiento de un controlador PID que regule un proceso o sistema se necesita, al menos:

1. Un sensor, que determine el estado del sistema.
2. Un controlador, que genere la señal que gobierna al actuador.
3. Un actuador, que modifique al sistema de manera controlada (resistencia eléctrica, motor, válvula, bomba, etc.)

El sensor proporciona una señal digital o analógica al controlador, la cual representa el *punto actual* en el que se encuentra el proceso o sistema. La señal puede representar ese valor en tensión eléctrica, intensidad eléctrica o frecuencia.

El controlador lee una señal externa que representa el valor que se desea alcanzar (Set Point). Esta es de la misma naturaleza y tiene el mismo rango de valores que la señal que proporciona el sensor. Para hacer posible esta compatibilidad y que, a su vez, la señal pueda ser entendida por un humano, habrá que establecer algún tipo de interfaz.
El controlador resta la señal de *punto actual* a la señal de *punto de referencia*, obteniendo así la señal de *error*, que determina en cada instante la diferencia que hay entre el valor deseado y el valor medido. La señal de error es utilizada por cada una de las tres componentes de un controlador PID propiamente dicho para generar las tres señales que, sumadas, componen la señal que el controlador va a utilizar para gobernar al actuador. La señal resultante de la suma de estas tres señales se llama *variable manipulada* y no se aplica directamente sobre el actuador, sino que debe ser transformada para ser compatible con el actuador que usemos.
Las tres componentes de un controlador PID son: parte **P**roporcional, acción **I**ntegral y acción **D**erivativa. El peso de la influencia que cada una de estas partes tiene en la suma final, viene dado por la *constante proporcional*, el *tiempo integral* y el *tiempo derivativo*, respectivamente.

2. DISEÑO

2.1 Estudio de las características eléctricas del motor

Para desarrollar este proyecto nos es necesario partir de los elementos finales de control que usaremos.

Una de las premisas de diseño sugeridas por el tutor de este trabajo fue el uso de un motor dc de potencia moderada. Debido a que no fue fácil encontrar un motor dc con esta especificación el tutor sugirió usar un motor universal el cual puede trabajar como un motor dc de excitación independiente y es de fácil consecución en el mercado.

Con el fin de seleccionar los valores de tensión que aplicaremos en lo sucesivo a ambos devanados del motor, se realiza este experimento en el cual evaluaremos el comportamiento del mismo ante estas tensiones de excitación. Se escoge 4V por varias razones: la primera es que su corriente asociada guarda un margen prudente para no forzar el devanado del estator a trabajar a su máxima corriente que es 6A, en segundo lugar

estator		rotor			
Vent real{V}	I real{V}	Vent real{V}	I real{V}	rpm	T
4,045	4,730	2,400	0,280	126,400	42,2
4,045	4,730	4,700	0,285	276,000	19,3
4,047	4,730	7,400	0,300	464,000	11,5
4,047	4,730	9,900	0,325	652,000	8,19
4,047	4,730	11,800	0,330	784,000	6,81
4,046	4,740	14,900	0,345	1005,000	5,31
4,046	4,740	16,900	0,355	1140,000	4,68
4,046	4,750	19,700	0,365	1346,000	3,97
4,045	4,760	22,200	0,378	1500,000	3,56
4,045	4,760	23,800	0,386	1630,000	3,27
4,042	4,770	26,100	0,392	1800,000	2,97
4,042	4,770	28,900	0,405	2000,000	2,67

Tabla 1. Selección de la tensión de alimentación para el rotor

En los datos de esta tabla se puede notar que la corriente consumida por el mismo es prácticamente constante independientemente se esté variando el valor de tensión en el rotor.

Otro aspecto interesante es la linealidad que presenta el motor en los primeros valores como se puede observar en la siguiente gráfica.

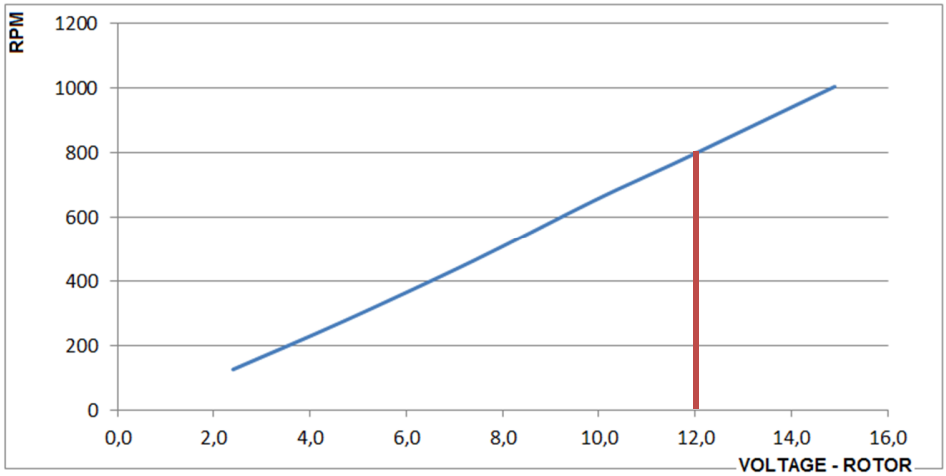


Figura 13. Selección de tensión de trabajo del rotor

Teniendo en cuenta esta información y considerando también el rango de velocidades en el que se desea se decide tomar como valor máximo de alimentación de rotor **12V**.

Entonces se tiene que para el funcionamiento del motor deseado en este proyecto se requiere que sea alimentado de la siguiente manera:

ESTATOR: 4V@4,047V

ROTOR: 12V@330mA - *corriente en marcha.* (ver tabla No.1, Fila 5)

12V@8A - *rotor frenado*

2.2 Fase de potencia (Half-Bridge)

La etapa de potencia y de control de sentido de giro se maneja mediante un Half Bridge. Para el diseño de esta etapa es necesario hacer algunos experimentos que arrojan las premisas para proceder.

En primer lugar es necesario hacer una prueba de escalón. La finalidad de esta prueba es determinar el tiempo que tarda el sistema en pasar de un nivel bajo a uno alto. Para esto implementamos el circuito de la Figura 14. en donde la inductancia corresponde al rotor.

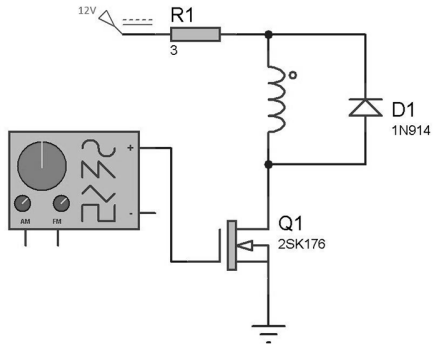


Figura 14. Montaje para pruebas del motor

Cuando se aplica un pulso en la compuerta del Mosfet, el voltaje de este abre el campo de drenador a surtidor polarizando el devanado del rotor con una corriente I1. Cuando el pulso está en cero aparece una corriente de inercia causada por la tendencia del devanado del rotor a mantener la misma corriente. Para evitar que esta tensión destruya el Mosfet y evitar la deformación de la forma de onda en el drenaje del transistor, se coloca un diodo volante que mantenga la corriente de inercia del devanado del motor y consuma la energía almacenada en forma de

campo magnético en el menor tiempo posible

Se le suministran 12V con una fuente del laboratorio y se le aplica una señal de activación en la compuerta del Mosfet mediante un generador de señales. Una vez hecho esto se puede observar en el osciloscopio el siguiente resultado:

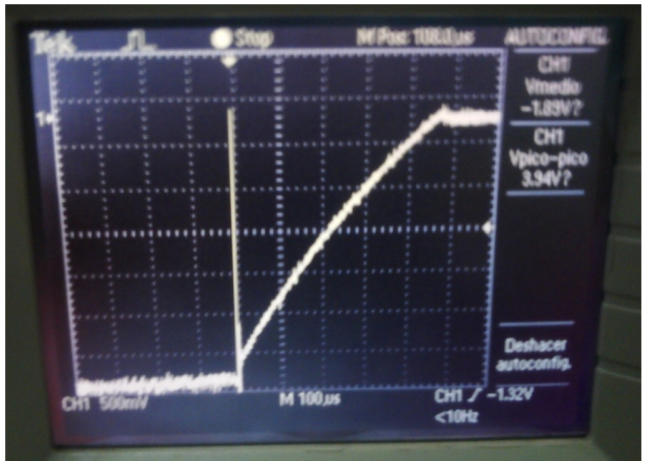


Figura 15. Prueba de escalón al motor DC

En el gráfico es posible observar que el tiempo que se tarda en llegar desde la base hasta la cima del pulso es de ***470uS***. Este tiempo indica el límite mínimo y máximo de ancho de pulso de la señal pwm, ya que es el mínimo de tiempo muerto que debemos tener para hacer un cambio de sentido de giro con el fin de que no se traslapen las señales de ambos lazos y halla conflicto en el sistema.

Luego de obtener esta información se hace un barrido en frecuencia el cual nos dará información acerca de cómo responde el motor ante las distintas frecuencias. Esto se hace con el fin de elegir la frecuencia a la que se manejara la señal de PWM aplicada.

Para esto se usa el mismo montaje de la Figura 14. y con el generador de señales variamos las frecuencias y vamos observando la respuesta del motor. Después de realizar esta experiencia se determina lo siguiente:

- Al hacer un barrido de 0 a 1700Hz, se puede ver que a mayor frecuencia se pierde cada vez más torque o fuerza.
- Para frecuencias menores a 30Hz, se puede ver un movimiento rotatorio pulsante que no es continuo.
- Se observa que el punto de equilibrio en el que se alcanza una mejor combinación de torque con movimiento continuo es en 60Hz

La explicación más acertada que se concluye después de este hallazgo es que debido a que es un motor universal diseñado para trabajar con corriente alterna a 60Hz, por lo tanto esta es la frecuencia a la que mejor responde este dispositivo.
De aquí entonces que la frecuencia a la que se maneja el PWM de este proyecto es 60Hz.

Una vez se tiene la frecuencia se hace una última practica consistente en conocer la respuesta del motor con un PWM de 60Hz a distintos ciclos de trabajo. Para este caso se hará un barrido de 30 a 70% que es lo que permite el generador de señales del laboratorio. De esta manera se obtiene una respuesta con buen torque en todo el rango de operación, siendo un poco más fuerte para los ciclos de trabajo superiores. Además el rango de velocidad correspondiente para este barrido es de 450RPM a 772RPM.

El diseño se implementa entonces con cuatro Mosfet's IRF540N los cuales tienen un tiempo de encendido y apagado de 35nS, lo que significa que al apagar un lazo para cambiar de giro nunca se traslaparan las señales. Otra de las razones por las que seleccionamos estos dispositivos y no BJT's por ejemplo, es su aislamiento con la compuerta lo que a su vez nos permite proteger la etapa de control aun más

y por ultimo debido a su resistencia de corte baja RDSon=0.055Ω la caída de tensión en el dispositivo es muy pequeña a diferencia de los BJT's.

Por otro lado la potencia del Mosfet que se calcula así: $P_D \cong I_o V_{DS}$ es para el caso más exigente que es DC puro:

$$P_{Dmax} \cong (4.5A)(1V) = 4.5W$$

La referencia de Mosfet mencionada puede disipar hasta 130W lo que asegura que no habrá recalentamientos en estos dispositivos.

Estos dispositivos, también cuentan con diodos en paralelo de conducción inversa para protegerlo de tensiones inversas producidas por cargas inductivas como en nuestro caso.

La compuerta de los mosfet's será alimentada con 12V para aquellos que están referenciados a tierra, aquellos que tienen referencia flotante es necesario adaptarles un elevador de tensión de modo que puedan tener el voltaje suficiente en la compuerta para trabajar correctamente.

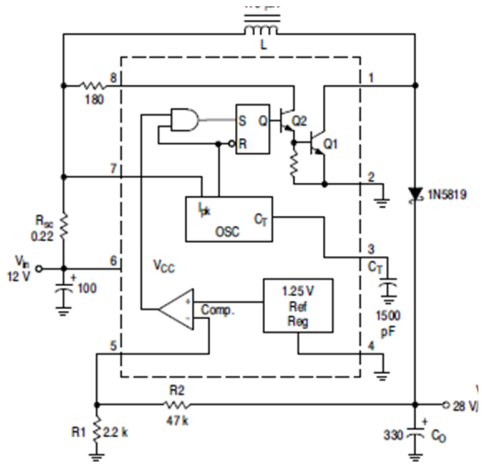


Figura 16. Esquema del desplazador de tensión para MOSFET

Se usaran cuatro optoacopladores para separar las señales de control provenientes del microcontrolador y la etapa de potencia. El circuito se implementara como se muestra en la Figura 17.

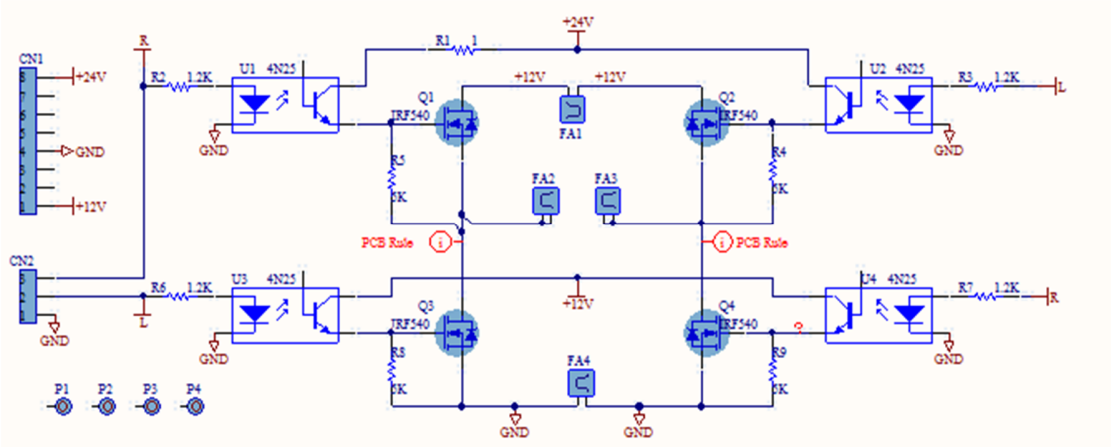


Figura 17. Esquemático en Half-Bridge H

Las gráficas a continuación muestran la respuesta del Driver construido al aplicarle una señal pwm con el generador de señales a 60Hz de 30 a 70% del ciclo útil.

Pwm(30%) – 60Hz Pwm(50%) – 60Hz Pwm(70%) – 60Hz

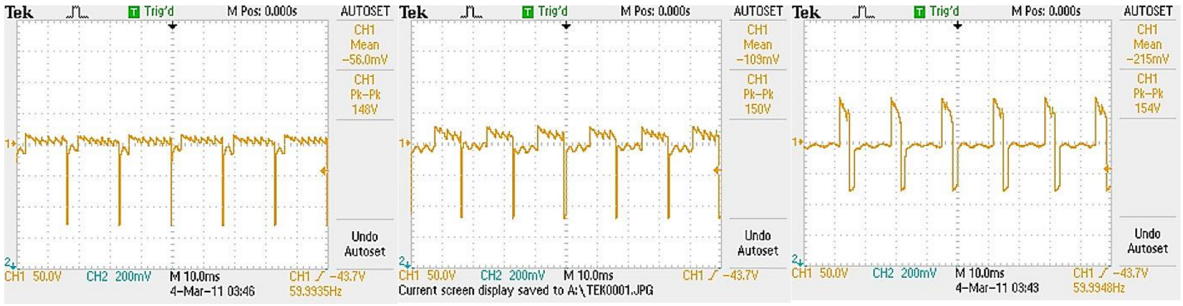


Figura 18. Pruebas del ciclo útil de PWM a 60Hz

2.3 Fase de selección de Hardware y Software de control

2.3.1 Hardware

Para este proyecto una de las premisas en cuanto a hardware de control, es que este cuente con la capacidad de establecer una comunicación bidireccional USB entre el sistema embebido y la computadora; la cual sea útil para conocer en tiempo real el estado de la variable controlada y manipulada a la vez de enviar a la planta la información de configuración deseada.

Otro elemento importante que se requiere del dispositivo de control es que cuente con Hardware de multiplicación, esto con el fin de agilizar las operaciones del controlador PID que va a ser implementado en el, ahorrando valioso tiempo de procesamiento.

Además es necesario que cuente con múltiples fuentes de interrupción para priorizar el procesamiento del dispositivo y una muy buena velocidad de operación de tal forma que se eviten al máximo los retardos.

Teniendo en cuenta estas premisas, hay en el mercado cercano dos dispositivos de la familia Microchip que cumplen mínimamente con todas las condiciones mencionadas. Estos son: PIC18F2550 y PIC18F4550. Aunque ambos cumplen con los requerimientos, el segundo cuenta con 40 pines que aunque no se usarán en su totalidad, sirven para que el proyecto sea escalable y no se quede corto ante requerimientos adicionales que surjan en el proceso de diseño.

2.3.2 Software

Para la programación del microcontrolador se usará el conocido entorno de programación MPLAB IDE de Microchip, a su vez fabricante de los microcontroladores PIC. Nuestro motivo es que es un entorno de programación fácil de usar, diseñado especialmente para microcontroladores del mismo fabricante y por último mediante el programador PicKit2 se acopla fácilmente de tal manera que podemos programar el PIC sin intermediarios y los archivos de configuración de hardware se cargan automáticamente y no hay necesidad de configurarlos manualmente a expensas de errores de usuario.

El compilador que se usará es CCS C, el cual es muy usado en el mundo para la programación de PICs porque tiene un entorno amigable e intuitivo haciendo mucho más sencillo la programación de los PIC's y no requiere que el programador sea un experto en la arquitectura especifica ni que necesite estar pegado al datasheet del mismo para poder avanzar en su código.

Por último en comparación con los compiladores Hi-Tech y C18, el CCS tiene comandos que facilitan el trabajo con bootloader, facilitando la organización del mapa de memoria evitando ampliamente los traslapos de codigo que con los otros si sucedia.

2.4 Fase de control de velocidad

El objetivo de esta fase es implementar un algoritmo capaz de realizar el control PID de la velocidad en un motor DC. Inicialmente se creará todo el entorno que hará esto posible, incluyendo la comunicación con la PC a partir de simulación en Proteus. Luego de eso se desarrollará un programa en Java que monitoreará y controlará la planta desde la PC. Una vez hecho esto y de construida una tarjeta de control que sea capaz de mandar las señales de control del PIC al Half Bridge, se procederá a la caracterización de la planta real para así después de haber

calculado las constantes PID, incluirlas en la programación y finalizar el proceso de implementación del sistema de control de velocidad.

2.4.1 Algoritmo de control PID

Lo primero que es necesario analizar antes de entrar al algoritmo PID como tal es la estrategia de captura de la señal de realimentación.

Para medir la velocidad de rotación del motor se decidió implementar de modo digital a través de un encoder incremental mediante interrupciones externas. El modo de funcionamiento se puede apreciar en la Figura 6. del marco teórico inicial.

Dentro de las opciones que se tienen para hacer la traducción de los pulsos del encoder a valores de RPM están: programar una ventana de tiempo de un segundo y con otro timer contar cuantos pulsos de ranura hay dentro de esta ventana. La segunda es contar con un timer el tiempo que se tarda en pasar de ranura a ranura y hacer el cálculo de RPM proporcional a la porción de revolución predefinida. La opción más viable para el caso de este proyecto es la segunda ya que el valor de RPM se actualiza cada vez que se lee un nuevo pulso del encoder y no cada segundo ya que ese tiempo es demasiado para una planta que responde tan rápido como es un motor DC.

Una vez capturado el valor de RPM se programa su visualización en un display 12x12 para efectos de comparación. En Proteus se implementa una simulación de este sistema remplazando el encoder por un generador de señales que ajustamos en 100Hz. Haciendo el cálculo para un encoder de 20 ranuras esta frecuencia debería arrojar un valor de velocidad igual a 300RPM. Multiplicamos entonces el inverso del tiempo contado (**`ticks`**) por un factor tal que nos arroje este resultado.

```
#int_EXT
void  EXT_isr(void)
{
   flag=true;
   if (PIN_B0) {
      ticks=get_timer0();
      set_timer0(0);
   }
}
```

```
void leerEncoder(void){
  lcd_gotoxy(1,1);
  rpmt = floor((140640.0)/ticks);
  if(flag){
    printf(lcd_putc,"%ld",65536-(65536-(rrpm)))
  }else{
    printf(lcd_putc,"%.1f",0.0);
  }
}
```

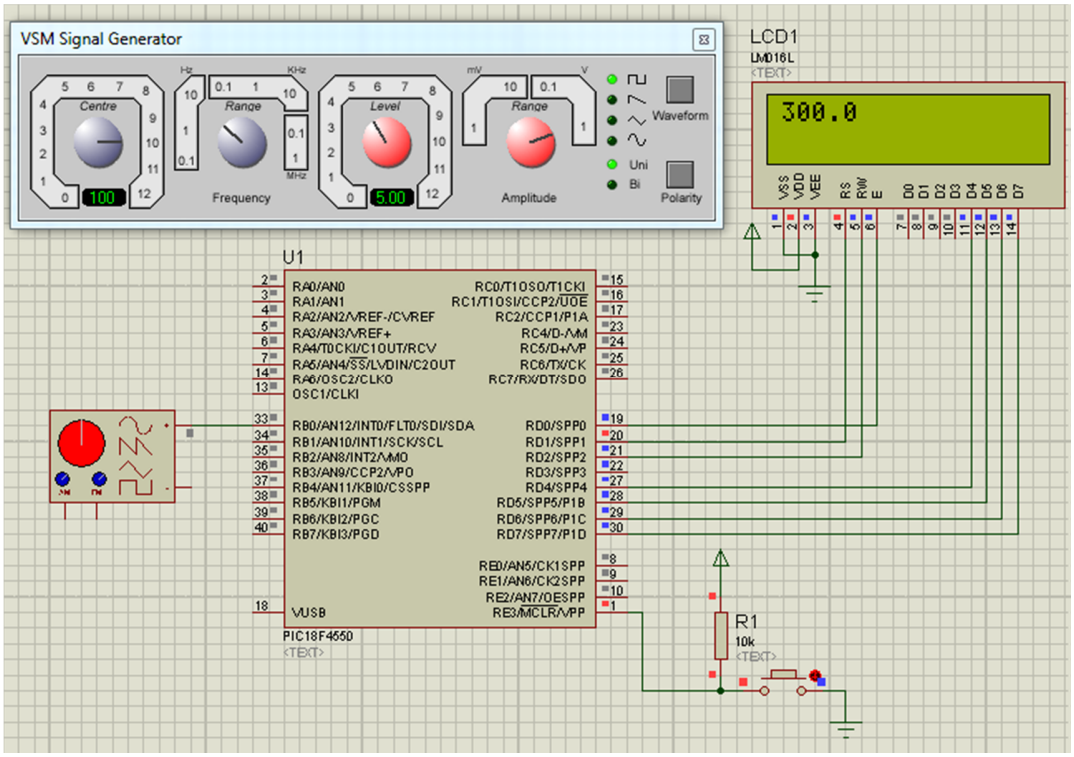


Figura 19. Calibración del medidor de RPM del PIC

Ya nos hemos asegurado “en simulación” que el lector de RPM esta calibrado. Cuando esto se implemente físicamente, seguramente tendrá que recalibrarse.

El paso a seguir entonces es la implementación del algoritmo PID, usando la metodología definida en el marco teórico, de la siguiente manera:

```
eT=ref-rpmt;          //Cálculo error
pT=a*eT;

  iT=b*eT+iT0;          //Cálculo del término integral
  dT=c*(eT-eT0);        //Cálculo del término derivativo
  uT=pT+iT+dT;          //Cálculo de la salida PID

  if (uT>max){          //Condición de saturación
    uT=max;
}

  iT0=iT;               //Guardar los valores antiguos en memoria
  eT0=eT;

PWM2=floor(uT*99/max);//Normalización de salida
pwm22 = (int8)PWM2;
pwm_duty(pwm22);
```

Como se puede observar, existe una condición de saturación que impide que la respuesta del controlador exceda cierto límite definido por el programador. También cuenta con una condición de normalización de tal manera que hace que la salida del controlador oscile únicamente entre 0 y 99 esto debido a que esta respuesta es el ciclo útil de la señal PWM el cual está definido en ese mismo rango.

Este código por su puesto se ejecuta en un ciclo infinito

```
while (TRUE){

leerEncoder();

calculoControlador();
```

Como ya se mencionó la salida del controlador es el porcentaje de ciclo útil de la señal PWM y se llama a una función llamada "**`pwm_duty()`**", esta función es la encargada construir la señal PWM.

¿Porque no se usa el módulo PWM del PIC? La respuesta es porque debido a que el oscilador que estamos trabajando es de 20MHz, al usar el módulo PWM se observó que lo mínimo a lo que se puede reducir la frecuencia por todos los medios posibles es a 1200Hz y por razones ya explicadas anteriormente debemos usar una frecuencia de PWM de 60Hz.

Es por esto que usando el timer1 que es de 16bits construimos la señal PWM como se muestra:

```
void pwm_duty(int8 porct){
  porc = porct;

// "freq" es directamente proporcional
// a la frecuencia de la señal pwm
//*******************************
freq = 0.53; //0.53 = 60Hz
//*******************************

  d_on = (int16)((65536*freq)+(1-(porc*0.01))*(65536*(1-freq)));
  d_off = (int16)((65536*freq)+(porc*0.01)*(65536*(1-freq)));
}

#int_TIMER1
void  TIMER1_isr(void)
{
  if (on == 0){
    output_high(PIN_C1);
    set_timer1(d_on);
    on = 1;
  }else{
    output_low(PIN_C1);
    set_timer1(d_off);
    on = 0;
  }
}
```

El PWM en nuestro sistema representa la variable manipulada que le indica al actuador (motor dc) como debe comportarse para para corregir la variable controlada que es la velocidad.

2.4.2 Software de monitoreo

Fue necesario diseñar un programa de PC que fuera capaz de establecer una comunicación USB bidireccional con el sistema embebido, Monitorear numérica y gráficamente la variable controlada; monitorear el porcentaje de ciclo útil de PWM, error de realimentación. Además sea capaz de monitorear y configurar el SetPoint y las constantes PID. Adicionalmente el programa debe ser capaz de exportar los datos correspondientes a las gráficas de la variable manipulada que se hallan realizado con la finalidad de que pueda el usuario final hacer distintos análisis según requiera con la planta. Este programa debe contar con la opción de configurar el sentido al que se desee gire el motor y si se quiere manipular la planta en lazo abierto o cerrado. Un PrintScreen puede verse a continuación.

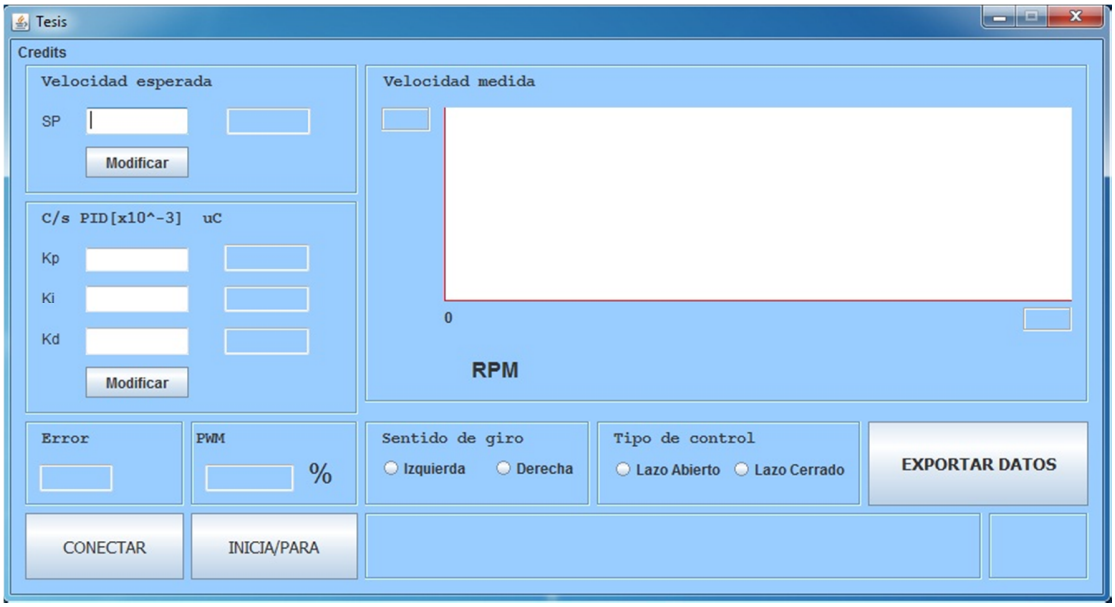


Figura 20. Interfaz de monitoreo y control del sistema

2.4.3 Comunicación USB

Para establecer la comunicación USB hay básicamente dos formas HID y BULK TRANSFER la primera es más organizada y ampliamente usada para dispositivos

como mouse's, teclados entre otros la velocidad no es lo más importante sino la organización de los datos. Por otro lado el método BULK TRANSFER es usado para transmitir grandes ráfagas de datos bidireccionalmente donde la velocidad es lo más importante. En este proyecto se implementa el método BULK TRANSFER via USB 2.0 a la máxima velocidad.

Para la programación de una comunicación USB desde la perspectiva del PIC es necesario tener en cuenta algunos elementos:

#fuses HSPLL y PLL5: La frecuencia de oscilación necesaria para el USB 2.0 es de 48Mhz. Como en este proyecto se usa un cristal de cuarzo de 20Mhz se necesita hacer uso del módulo PLL interno del PIC. Para ello utilizamos el fuse HSPLL. Como el módulo PLL requiere una oscilación de entrada de 4Mhz debemos utilizar el divisor 1:5 indicado con el fuse PLL5 para obtener los 20/5 = 4Mhz requeridos.

USB_ENABLE_BULK y SIZE 32: Para activar el método de transferencia masiva mediante el USB debemos configurar los EndPoint de transmisión y recepción, **USB_EP1_TX_ENABLE** y **USB_EP1_RX_ENABLE**, indicándolo con la constante **USB_ENABLE_BULK**. Es imprescindible deshabilitar el método HID (Human Interface Device). El tamaño del buffer de transferencia lo podemos ajustar desde 1 a 32 bytes como máximo. Para este proyecto se establecerá el máximo de 32 bytes por envío o recepción de paquetes USB.

Una vez iniciada la ejecución del método principal **`main()`**, es necesario inicializar y operar el modulo USB mediante las funciones **`usb_init()`**, **`usb_task()`**, **`usb_wait_for_enumeration()`**, **`usb_enumerated()`**, **`usb_kbhit()`**, **`usb_get_packet()`** y **`usb_put_packet()`**. Estas están disponibles en la librería que proporciona el compilador CCS C para el manejo del USB 2.0 y vienen definidas e implementadas en los **`includes pic18_usb.h, usb.c y usb.h`**

que se encuentran en el directorio de instalación de CCS "*Archivos de Programa\PICC\Drivers\...*".

`usb_init(), usb_task() y usb_wait_for_enumeration()` se utilizan solo para establecer la comunicación y se ejecutan únicamente tras un reset del micro.

Si ya ha podido el programa entrar al bucle infinito **`while(true)`**, solo se puede actuar si la función **`usb_enumerated()`** nos devuelve true, o sea que estamos correctamente conectados y reconocidos por el Windows del PC.

A partir de este punto solo queda esperar a recibir un comando proveniente del PC. Esto se detecta mediante la función **`usb_kbhit()`** que al devolvernos true nos indicará que se tiene algo pendiente de recibir. Y los vamos a recoger mediante **`usb_get_packet()`** quedando disponible en **`recbuf`**.

Para contestar o enviar datos al PC se usa la función **`usb_put_packet().`**

Desde la perspectiva de Java la comunicación USB gira alrededor de una clase de Java llamada jPicUSB. Esta clase permite a una aplicación Java hacer llamados a la librería dinámica "jpicusb.dll" la cual implementa todas las funciones de la API USB de Microchip "mpusbapi.dll", con la diferencia de que esta especialmente recompilada para permitir a la clase jPicUSB hacer llamados a sus funciones.

Para el uso de la clase jPicUSB son necesario tener en cuenta cuatro elementos básicos:

1. Es necesario antes de usar cargar la librería “jPicUSB.jar”

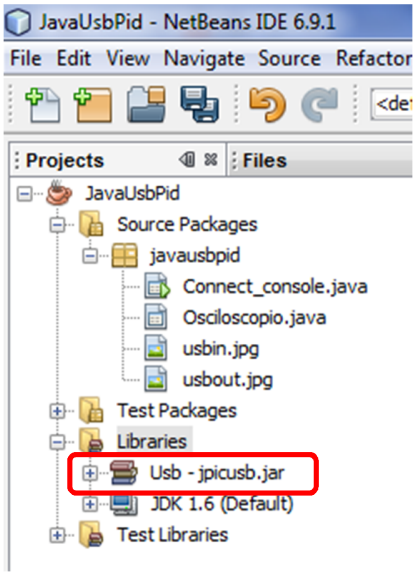


Figura 21. Ilustración para cargar la librería

2. Importar a la aplicación de java todos los paquetes contenidos en la clase jPicUSB.

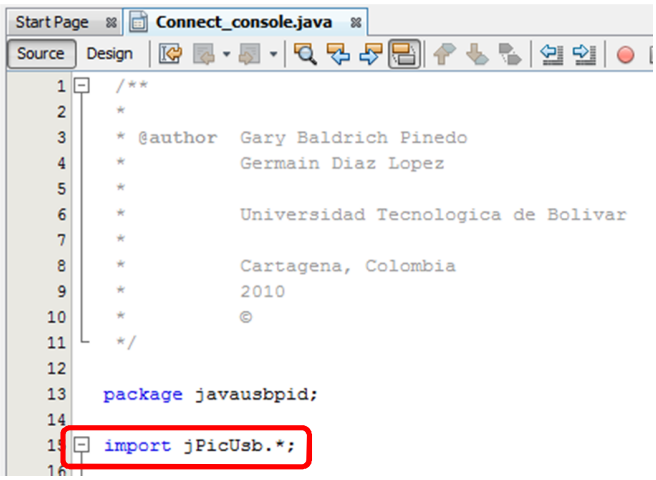


Figura 22. Ilustración de la importación de paquetes de jPicUSB

3. Inicializar la librería en el código antes de usarla.

```
try{
  iface.load();        //Se carga la libreria jPicUsb.dll del directorio raiz
  info.setText("");    //Se borra lo que esta en la linea
  info.setText("Libreria jPicUsb.dll cargada con éxito"); //Se informa
  usb.setIcon(new javax.swing.ImageIcon(getClass().getResource("/javausbpid/usbin.jpg")));
}
catch (Exception err){
  info.setText("");               //Se borra lo que esta en la linea
  info.setText(err.getMessage()); //y se escribe el mensaje de error si lo hay.
  return;
}
iface.set_instance(0);
iface.set_vidpid("vid_04d8&pid_000b"); //Se actiban las funciones de acceso rapido
```

Figura 23. Ilustración de los llamados para iniciar la librería

4. Copiar la librería jpicsb.dll a la carpeta donde se guarda el ejecutable del programa, antes de ejecutarlo.

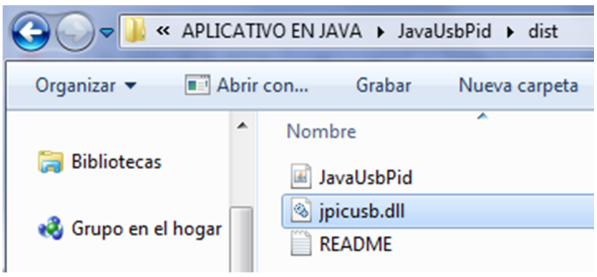


Figura 24. Ilustración para indicar donde se guarda la librería

En el caso de este proyecto java está constantemente enviando una trama a el PIC mediante el método `iface.QWrite()` de jPicUSB.

```
byte[] comando = {identificador,(byte)VsetPoint,kp1,kp0,ki1,ki0,kd1,kd0,sg,tc};

iface.QWrite(comando, 32, 1000);
```

Para recibir información del PIC se hace de esta manera:

```
respuesta = new String(iface.QRead(32, 500), "utf-8");
```

Java envía siempre delante de la trama un identificador que le indica al PIC que valores debe leer en ese envío.

Por otro lado el PIC le envía a Java: constantes PID, SetPoint, RPM y %PWM. Los dos primeros se envían solo cuando es solicitado mas los dos segundos se envían constantemente.

```
usb_put_packet(1,hdr_rpm,4,USB_DTS_TOGGLE);
```

Y para recibir y guardar en **recbuf** con **Lenbuf** número de bytes

```
usb_get_packet(1, recbuf, Lenbuf);
```

2.4.4 Modelamiento matemático y sintonización de la planta.

Se realiza para este fin una prueba a lazo abierto se fija el PWM en 26% y después de estabilizarse se modifica a 27%. Para extraer el modelo de la planta se analizara el segundo escalón.

Después de hacer esta práctica, se exportan los datos y se grafican.

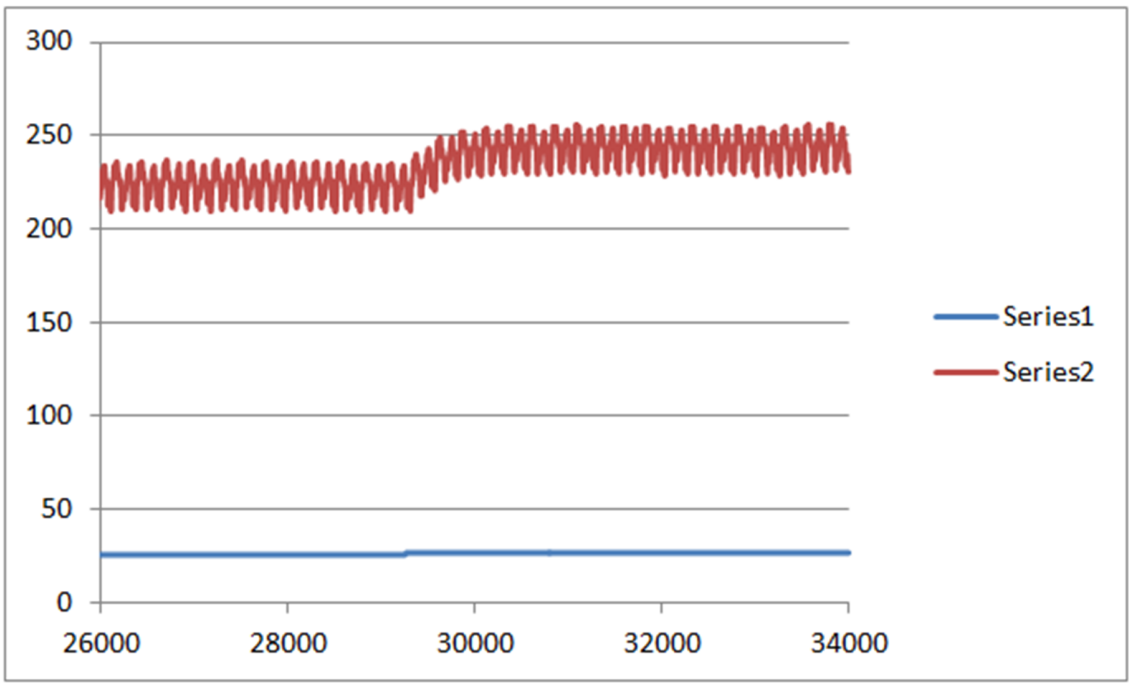


Figura 25. Prueba de escalón a 26% y 27% para caracterización

Debido a que la respuesta oscila mucho, aplicamos un promedio móvil de 35 puntos y obtenemos el resultado que se muestra en la Figura 26.

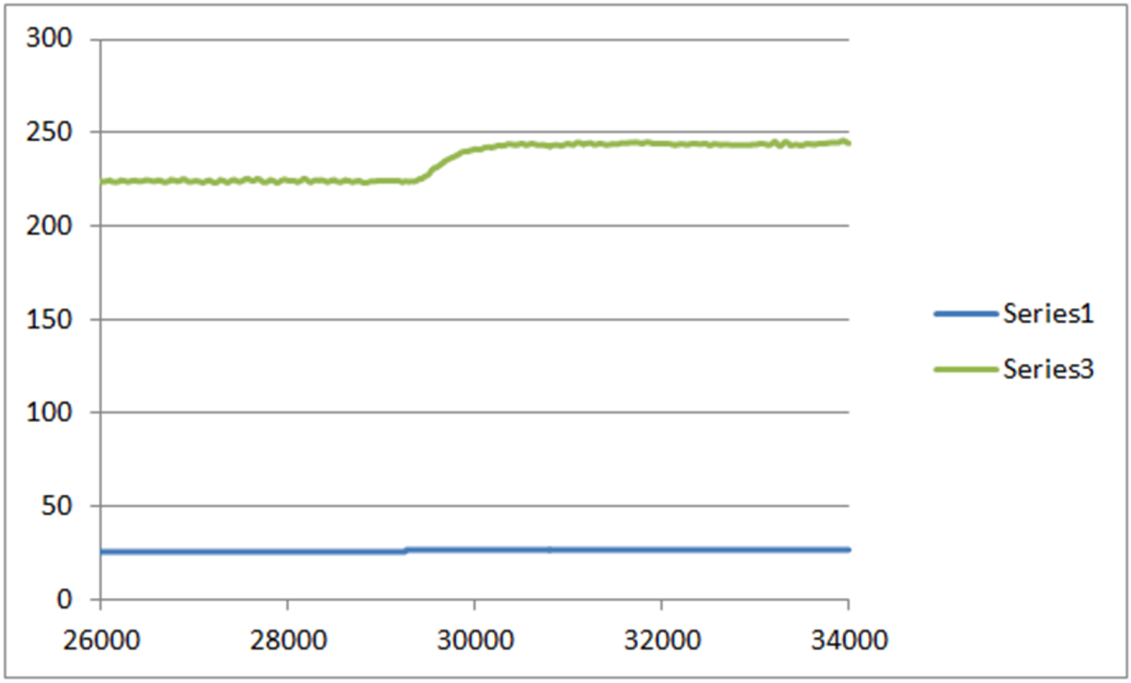


Figura 26. Prueba de escalón filtrada con promedio móvil de 35 puntos

Ahora se reduce la gráfica a la sección que se desea analizar.

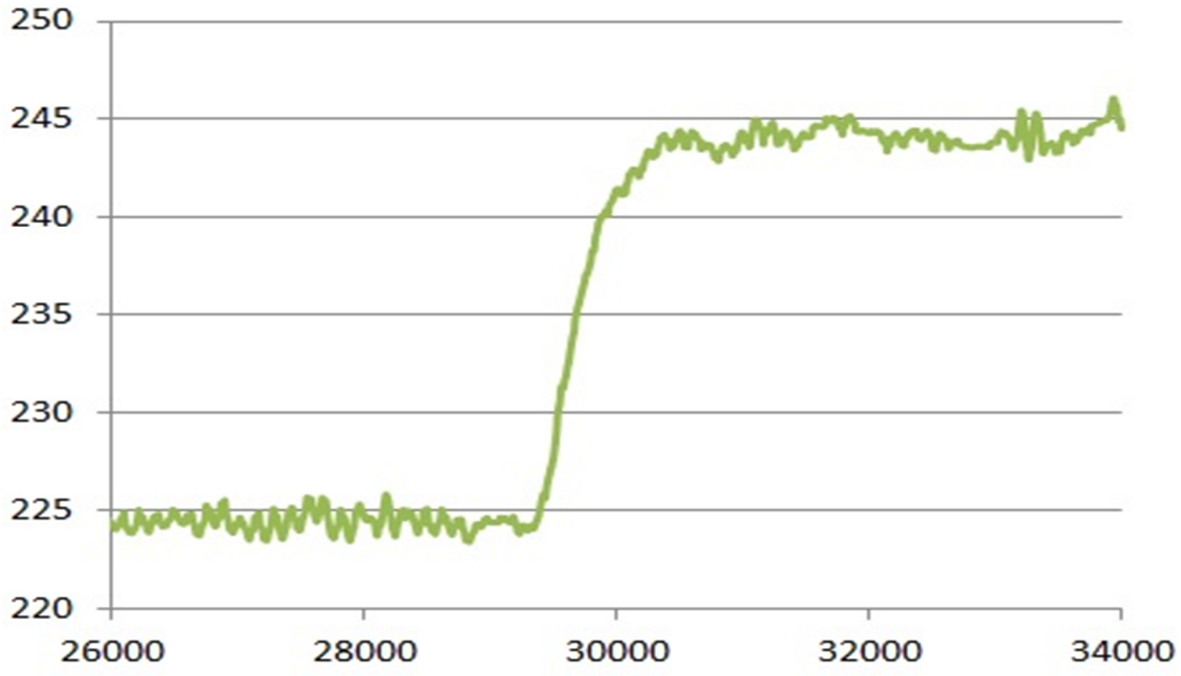


Figura 27. Sección de respuesta del escalón para análisis

Realizando el análisis de recta tangente de Ziegler Nichols se tiene.

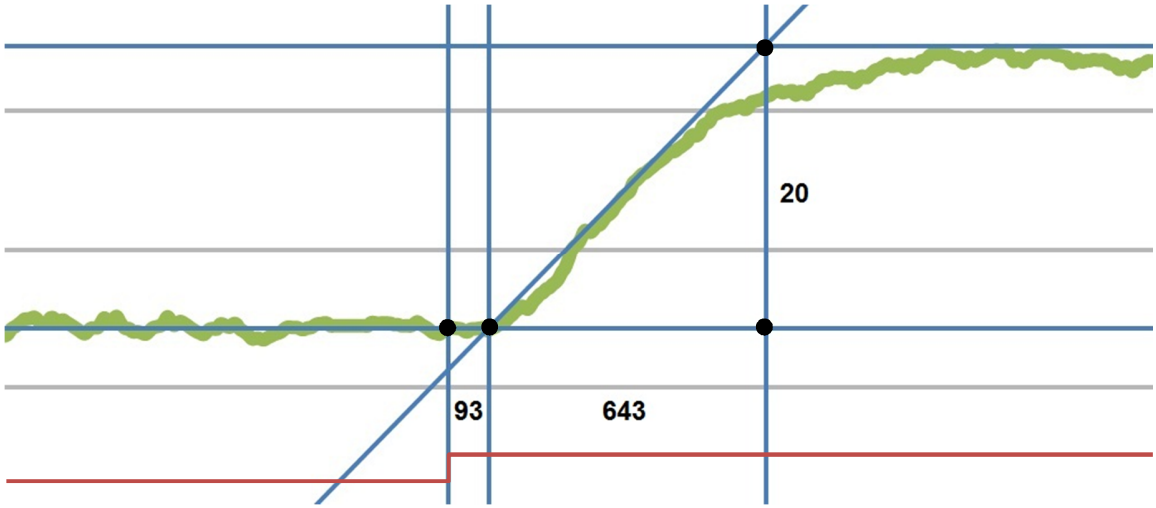


Figura 28. Análisis de recta tangente con el método de Zieger Nichols

El modelo resultante es: $G(s)=\frac{K\cdot e^{\theta s}}{1+sT_1};K=20;\theta=93;T_1=643\Rightarrow \boxed{G(s)=\frac{20\cdot e^{93s}}{1+643s}}$

Las constantes PID entonces son:

$$kp=\frac{T_1}{K\cdot T_D}=345\times10^{-3}$$

$$ki=\frac{0.9T_1}{K\cdot T_D}=311\times10^{-3}$$

$$kp=\frac{1,2T_1}{K\cdot T_D}=414\times10^{-3}$$

3. RESULTADOS

Figura 29. Prueba de Operación

Se puede observar en la Figura 29. La regulación inmediata del sistema, a pesar de que la señal adquirida es muy oscilante se puede apreciar que mantiene un promedio estable, además al comprobar con un tacómetro digital en el

laboratorio, la velocidad coincide en un 99,5% sobre el setPoint como lo muestra la tabla 2.

SetPoint	RPM medida con tacómetro digital	Índice de coincidencia
100	100.2	99.8%
150	150.1	99.3%
180	180.2	99.4%
	Promedio	**99.5%**

Tabla 2. Prueba de regulación del controlador

Sabiendo como ya se dijo que la señal conserva un promedio estable, se procede a someter a la señal a un filtrado mediante un Moving Average de 35 puntos en Excel obteniendo la curva de color negro de la Figura 30.

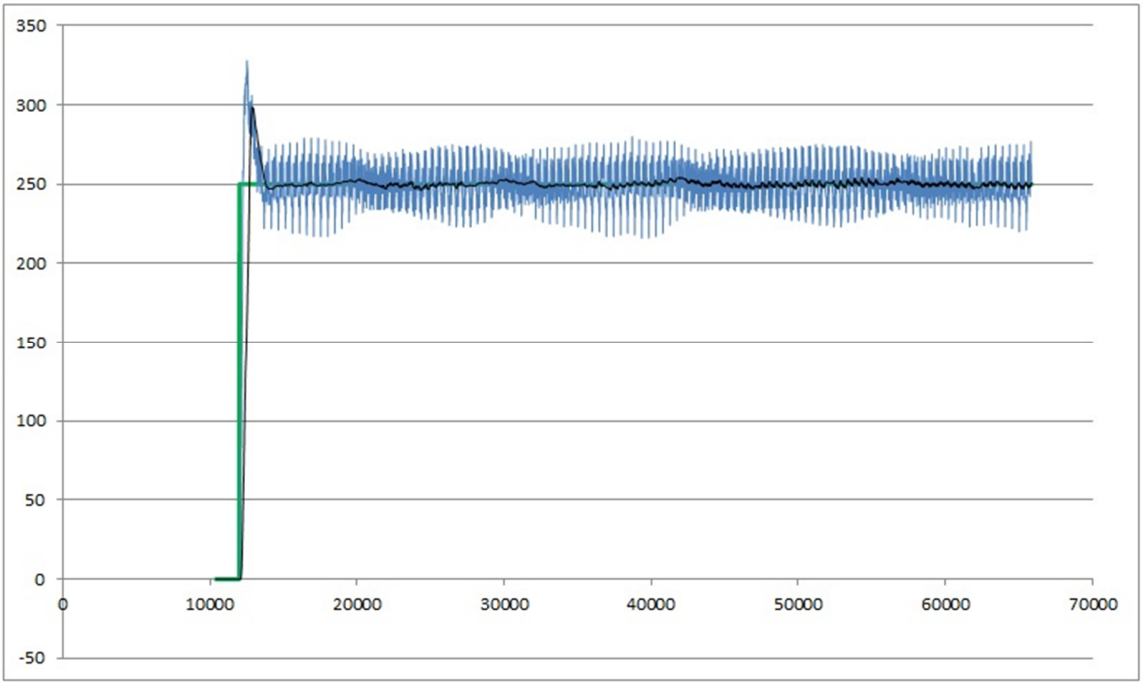


Figura 30. Datos exportados desde Java a Excel: SetPoint, Señal adquirida, Señal filtrada

En la Figura 31. Se puede ver más claramente la relación que existe entre la señal filtrada y el SetPoint.

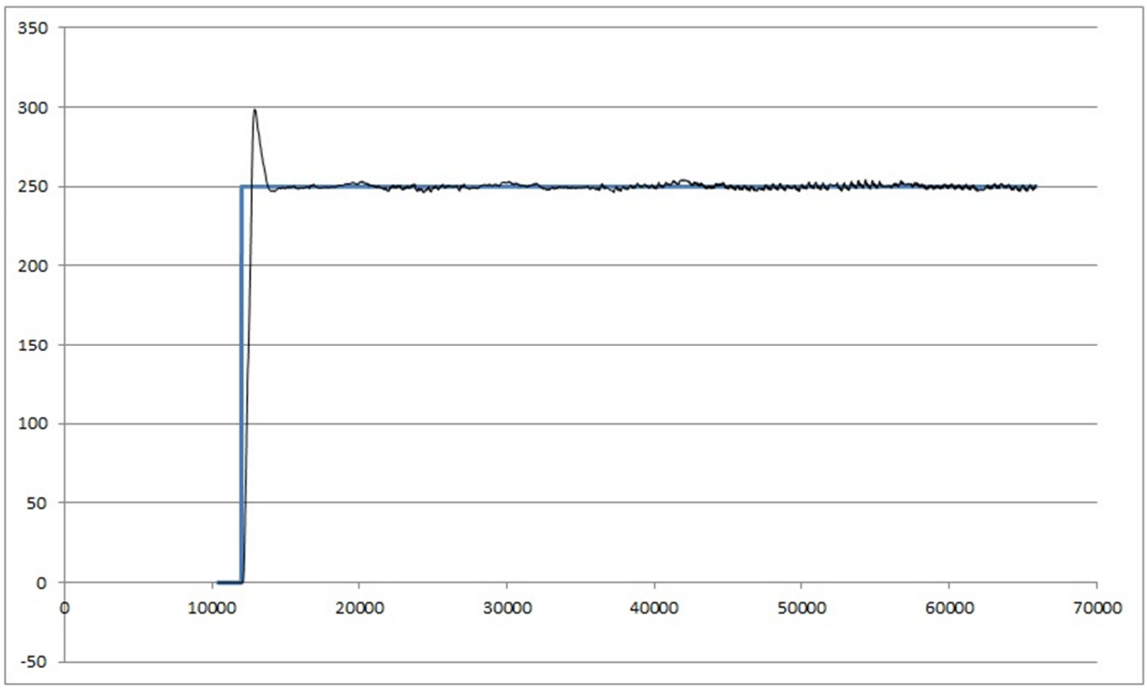


Figura 31. SetPoint, Señal Adquirida Filtrada

CONCLUSIONES

Dados los avances en la tecnología de los microcontroladores PIC18, su hardware interno y su comunicación rápida USB 2.0; se ha demostrado que son completamente fiables para aplicaciones de control robustos. Al mismo tiempo se puede decir que a pesar de la complejidad de los cálculos de un control PID esta gama de microcontroladores puede mantener operaciones paralelas complejas como comunicación USB 2.0, monitoreo en display lectura de variables mediante sus periféricos sin utilizar hardware adicional.

La regulación es precisa, las oscilaciones se deben a defectos en el eje del rotor. Ya que el encoder usado esta acoplado a este eje la señal que llega al PIC es oscilante también, lo que dificulta un poco más la tarea del controlador. Aun así, el microcontrolador PIC18F4550 demostró ser una herramienta robusta para la automatización de este proceso independientemente de las perturbaciones.

La comunicación USB por el método BULK TRANSFER, es moderadamente confiable y la presencia de errores es mínima. Aun así no es conveniente utilizar este método de comunicación para la transmisión de variables de control en procesos críticos. Se recomienda conservar los métodos de comunicación estandarizados en la industria.

RECOMENDACIONES

Para mejorar el desempeño del proyecto se sugiere:

1. Programar en Java un algoritmo de sintonización automática
2. Al monito gráfico se le puede aplicar directamente un filtro móvil de 35 puntos para que la señal no se vea tan oscilante.

BIBLIOGRAFÍA

[1] Jan Axelson. **USB Complete: The Developer's Guide, Fourth Edition.** Lakeview Research LLC, 2009.

[2] Ali Mazidi. **Pic Microcontroller and embedded systems: using assembly and C for PIC C18**, Pearson 2008.

[3] Angúlo Usategui. **Microcontroladores PIC: diseño práctico de aplicaciones, 2ª Edición**. Mc Graw-Hill, 1999

[4] Bates, Martin. **Programing 8 bit PIC microcontrollers in C: with interactive hardware simulation**. Elsevier, 2008.

[5] D. Ibrahim. **Microcontroller based applied digital control**. John Wiley & Sons. 2006.

[6] D. Ibrahim. **Advanced PIC microcontroller Projects in C: From USB to RTOS with the PIC18F series**. Elsevier, 2008.

[7] Eduardo García Breijo, **Compilador C CCS y simulador PROTEUS para microcontroladores PIC,** Alfaomega, 2008.

[8] Timothy J. Maloney. **Electrónica Industrial Moderna (Spanish Edition)**. Pearson, 2006.

[9] Paul J. Deitel. **Java How to Program, 7th Edition**. Prentice Hall, 2007

[10] Charles L. Phillips. **Digital Control System Analysis and Design (3rd Edition)**. Prentice Hall, 1995.

[11] Dorf, Richard C; Bishop Robert H . **Modern Control System. (12th edicion).** Pearson, 2011.

[12] CUSTOM COMPUTER SERVICES INC, **C Compiler reference manual June 2010**. <http:/www.ccsinfo.com/dowloads/ccs_c_manual.pdf> .

[13] MICROCHIP**, Low-Cost Bidirectional Brushed DC Motor Control Using the PIC16F684 [en linea]**. <http:/ww1.microchip.com/downloads/en/AppNotes/00893a.pdf>.

[14] DIVIDE AND CONQUER, **jPicUSB: Comunicación PIC + USB usando JAVA**. <http:/www.divideandconquer.com.ar/2009/01/jpicusb-comunicación-pic-usb-usando-java/>. <http:/www.scribd.com/doc/40201924/jPicUSB> .

[15] FOROS TODOPIC. Tema: **Bootloader USB para PIC18F4550 a full**. < http://www.todopic.com.ar/foros/index.php?topic=23135.0>.

ANEXOS

ANEXO A: Programa del PIC

```
/*
        Universidad Tecnologica de Bolivar
        Programa: JavaUsbPid v1.0
        Autores: Gary Baldrich Pinedo gbaldrich@msn.com
                 Germain Diaz Lopez germa_@hotmail.com
*/

#include <18F4550.h>
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)

#build(reset=0x800, interrupt=0x808)
#org 0x000, 0x7ff { }

#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
#include <stdlib.h>
#include <internal_eeprom.c>
#include <lcd_portd.c>
//          PIC                         Function    LCD
// D0       19    enable                  Pin  6  -->GND
// D1  20    rs                          Pin  4  -->GND
// D2  21     unused
// D3  22    rw                          Pin  5
// D4  27    D4                          Pin 11
// D5  28    D5                          Pin 12
// D6  29    D6                          Pin 13
// D7  30    D7                          Pin 14

#include <math.h>
```

```
#define head recbuf[0]
#define SetPoint recbuf[1]
#define rx_byte2 recbuf[2]
#define rx_byte3 recbuf[3]
#define rx_byte4 recbuf[4]
#define rx_byte5 recbuf[5]
#define rx_byte6 recbuf[6]
#define rx_byte7 recbuf[7]
#define Sentido_Giro recbuf[8]
#define Tipo_Control recbuf[9]

#define SET_POINT 100      //d
#define CONS_PID 104       //h
#define CPID 105           //i
#define RIGHT 106          //j
#define LEFT 107           //k
#define L_ABIERTO 109      //m
#define L_CERRADO 110      //n
#define RPM 99                    //c
#define INICIALIZAR 88     //X

const int8 Lenbuf = 32;
char buf[Lenbuf];



//char hdr_ini[Lenbuf] ={"X"};
char hdr_rgt[Lenbuf] ={"j"};
char hdr_lft[Lenbuf] ={"k"};
char hdr_lab[Lenbuf] ={"m"};
char hdr_lcr[Lenbuf] ={"n"};

int16 num = 0, rrpm;
int8 recbuf[Lenbuf];

int8 sp = 0;
int16 kp=0,ki=0,kd=0,kpr=0,kir=0,kdr=0,z=0,dir=0;
float rpmt;

boolean flag;
int16 ticks;


float eT=0.0,eT0=0.0,a=20.0,b=1,c=250.0;
float iT=0.0,iT0=0.0,dT=0.0,pT=0.0,uT=0.0, freq=0.0;
```

```
int8 yT,ref,int_uT, pwm22;
int16 min=0;
int32 max=150000,PWM2=0;

int s=0, on=0;
int16 d_on, d_off = 0, porc=0;

void convertirachar (int16 numero){
        //char buf[3];
        itoa(numero, 10, buf);
}

void enviarConsPid(void){
  char hdr_pid[Lenbuf] ={"h"};
        char skp[6];
        char ski[6];
        char skd[6];
        char sep[2] ={"-"};
        //char end[3] ={"-/"};

        kpr = read_int16_eeprom(CONS_PID);
        itoa(kpr, 10, skp);
        strcat(skp,sep);

        kir = read_int16_eeprom(CONS_PID+2);
        itoa(kir, 10, ski);
        strcat(ski,sep);

        kdr = read_int16_eeprom(CONS_PID+4);
        itoa(kdr, 10, skd);

        strcat(hdr_pid,skp);
        strcat(hdr_pid,ski);

        strcat(hdr_pid,skd);
        usb_put_packet(1,hdr_pid,18,USB_DTS_TOGGLE);
}
//--------------------
void enviarRpm(void){
        char sep[2] ={"-"};
        char hdr_rpm[Lenbuf] ={"c"};

        rrpm =(int16)rpmt;
        convertirachar(rrpm);
        strcat(hdr_rpm,buf);
```

```
            convertirachar(pwm22);
            strcat(hdr_rpm,sep);
            strcat(hdr_rpm,buf);


            if (rpmt<10){
                        if (pwm22<10)
                        usb_put_packet(1,hdr_rpm,4,USB_DTS_TOGGLE);
                        if (pwm22<100)
                        usb_put_packet(1,hdr_rpm,5,USB_DTS_TOGGLE);
                        if (pwm22<1000)
                        usb_put_packet(1,hdr_rpm,6,USB_DTS_TOGGLE);
            }
            if (rpmt<100){
                        if (pwm22<10)
                        usb_put_packet(1,hdr_rpm,5,USB_DTS_TOGGLE);
                        if (pwm22<100)
                        usb_put_packet(1,hdr_rpm,6,USB_DTS_TOGGLE);
                        if (pwm22<1000)
                        usb_put_packet(1,hdr_rpm,7,USB_DTS_TOGGLE);
            }
            if (rpmt<1000){
                        if (pwm22<10)
                        usb_put_packet(1,hdr_rpm,6,USB_DTS_TOGGLE);
                        if (pwm22<100)
                        usb_put_packet(1,hdr_rpm,7,USB_DTS_TOGGLE);
                        if (pwm22<1000)
                        usb_put_packet(1,hdr_rpm,8,USB_DTS_TOGGLE);
            }
}

void enviarSetPoint(void){
            char hdr_spt[Lenbuf] ={"d"};
            sp = read_eeprom(SET_POINT);
            convertirachar(sp);
            strcat(hdr_spt,buf);

            usb_put_packet(1,hdr_spt,7,USB_DTS_TOGGLE);
}

void pwm_duty(int8 porct){
            porc = porct;

// "freq" es directamente proporcional
// a la frecuencia de la señal pwm
```

```
//*********************************
freq = 0.53; //0.53 = 60Hz
//*********************************

        d_on = (int16)((65536*freq)+(1-(porc*0.01))*(65536*(1-freq)));
        d_off = (int16)((65536*freq)+(porc*0.01)*(65536*(1-freq)));
}

//*********************//
//*********************//
//*********************//
void leerEncoder(void){
        lcd_gotoxy(1,1);
  rpmt = floor((140640.0)/ticks);
  if(flag){
    printf(lcd_putc,"%ld",65536-(65536-(rrpm)));
  }else{
        printf(lcd_putc,"%.1f",0.0);
  }
}


void calculoControlador(void){
        ref = read_eeprom(SET_POINT);
        if (ref != 0){
                if (read_eeprom(70) == L_CERRADO){
                        a= read_int16_eeprom(CONS_PID);   //se lee el término proporcional de la eeprom
                        b= read_int16_eeprom(CONS_PID+2); //integral
                        c= read_int16_eeprom(CONS_PID+4); //derivativo
                        eT=ref-rpmt;                      //Cálculo error
                        pT=a*eT;

                        iT=b*eT+iT0;        //Cálculo del término integral
                        dT=c*(eT-eT0);      //Cálculo del término derivativo
                        uT=pT+iT+dT;        //Cálculo de la salida PID

                        if (uT>max){       //Condición de saturación
                        uT=max;
                        }

                        iT0=iT;           //Guardar los valores antiguos en memoria
                        eT0=eT;

                        PWM2=floor(uT*99/max);//Normalización de salida
                        pwm22 = (int8)PWM2;
```

```
                    pwm_duty(pwm22);
                }else{
                    if (read_eeprom(70) == L_ABIERTO){
                        if (ref>100){
                            ref = 0;
                            output_low(PIN_C1);
                        }else{
                            pwm22=ref;
                            pwm_duty(pwm22);
                        }
                    }
                }
            }else{
                output_low(PIN_C1);
                pwm22=0;
            }
}

#int_TIMER0
void  TIMER0_isr(void)
{
  flag = false;
  lcd_gotoxy(1,1);
  lcd_putc ("\f");
  ticks = 0;
  set_timer0(0);
}


#int_TIMER1
void  TIMER1_isr(void)
{
flag = true;
//Control de sentido de giro
            if      (ref != 0){
                if      (dir == 0){
                    if (on == 0){
                        output_high(PIN_C1);
                        set_timer1(d_on);
                        on = 1;
                    }else{
                        output_low(PIN_C1);
                        set_timer1(d_off);
                        on = 0;
                    }
```

```
                    }
                    if          (dir == 1){
                                if (on == 0){
                                            output_high(PIN_C2);
                                            set_timer1(d_on);
                                            on = 1;
                                }else{
                                            output_low(PIN_C2);
                                            set_timer1(d_off);
                                            on = 0;
                                }
                    }
          }
}

#int_EXT
void  EXT_isr(void)
{
  flag=true;
  if (PIN_B0) {
    ticks=get_timer0();
    set_timer0(0);
  }
}

void main() {
 write_eeprom(70,L_CERRADO);

          lcd_init();
          set_tris_b(0b11111111);

//************************************************************************
 setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
          setup_timer_1(T1_INTERNAL|T1_DIV_BY_2);
 set_timer0(0);

 enable_interrupts(int_ext);
 enable_interrupts(int_timer0);
 enable_interrupts(int_timer1);
 enable_interrupts(global);

          if (s == 0) {
                    output_low(PIN_C2);
                    set_timer1(65534);
                    s = 1;
```

```
                    on=0;
            }
/*
  setup_ccp2(CCP_PWM);              // Configura CCP2 como PWM
  setup_timer_2(T2_DIV_BY_16,255,1); // f=1225.5Hz (T=816 us)
  set_pwm2_duty(0);
*/
  delay_ms(500);
  usb_init();
  usb_task();
  usb_wait_for_enumeration();

  while (TRUE){


            leerEncoder();

            calculoControlador();

                    if(usb_enumerated()){
            if (usb_kbhit(1)){
            usb_get_packet(1, recbuf, Lenbuf);

                                    //----------------------------------------

                                    if(head==SET_POINT){
                                            write_eeprom(SET_POINT,SetPoint);
                                            enviarSetPoint();
                                            //output_b(sp);
                                    }
                                    //----------------------------------------

                                    if(head==CONS_PID){
                                            enviarConsPid();
                                    }

                                    //----------------------------------------

                                    if(head==RIGHT){
                                            dir=0;
                                            usb_put_packet(1,hdr_rgt,1,USB_DTS_TOGGLE);
                                    }

                                    //----------------------------------------
```

```
                                        if(head==LEFT){
                                                dir=1;
                                                usb_put_packet(1,hdr_lft,1,USB_DTS_TOGGLE);
                                        }

                                        //----------------------------------------

                                        if(head==L_ABIERTO){
                                                write_eeprom(70,L_ABIERTO);
                                                usb_put_packet(1,hdr_lab,1,USB_DTS_TOGGLE);
                                        }

                                        //----------------------------------------

                                        if(head==L_CERRADO){
                                                write_eeprom(70,L_CERRADO);
                                                usb_put_packet(1,hdr_lcr,1,USB_DTS_TOGGLE);
                                        }
                                        //=======================================

                                        if (head==CPID){
                                                kp = make16(rx_byte2,rx_byte3);
                                                kd = make16(rx_byte4,rx_byte5);
                                                ki = make16(rx_byte6,rx_byte7);

                                                write_int16_eeprom(CONS_PID,kp);
                                                write_int16_eeprom(CONS_PID+2,kd);
                                                write_int16_eeprom(CONS_PID+4,ki);

                                                enviarConsPid();
                                        }
                                        //=======================================µC
        if(head==RPM){
                                                enviarRpm();
        }

                                        //----------------------------------------
        if(head==INICIALIZAR){
                                                enviarSetPoint();
        }
                                        //----------------------------------------
      }
    }
  }
}
```

ANEXO B: Programa de Java

```
/**
 *
 * @author  Gary Baldrich Pinedo
 *          Germain Diaz Lopez
 *
 *          Universidad Tecnologica de Bolivar
 *
 *          Cartagena, Colombia
 *          2010
 *          ©
 */

package javausbpid;
import java.awt.*;
import java.io.UnsupportedEncodingException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
import jPicUsb.*;
import java.io.BufferedWriter;
import java.lang.String.*;
import java.util.*;
import java.util.Calendar;
import java.util.Date;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.io.FileWriter;
import java.io.IOException;



//import javax.swing.Timer;



public class Connect_console extends javax.swing.JFrame {

  Vector <int[]> myv= new Vector<int[]>();
  //public String img = "usbout.jpg";

  public Connect_console() {
    initComponents();       //Se inicializan los controles gráficos
```

```
 rigth.setSelected(true);
 closeLoop.setSelected(true);
}

@SuppressWarnings("unchecked")
 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
 private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    buttonGroup2 = new javax.swing.ButtonGroup();
    jPanel2 = new javax.swing.JPanel();
    jPanel8 = new javax.swing.JPanel();
    usb = new javax.swing.JLabel();
    jPanel9 = new javax.swing.JPanel();
    label16 = new java.awt.Label();
    left = new javax.swing.JRadioButton();
    rigth = new javax.swing.JRadioButton();
    jPanel7 = new javax.swing.JPanel();
    label15 = new java.awt.Label();
    openLoop = new javax.swing.JRadioButton();
    closeLoop = new javax.swing.JRadioButton();
    jButton1 = new javax.swing.JButton();
    info = new javax.swing.JTextField();
    jPanel4 = new javax.swing.JPanel();
    label12 = new java.awt.Label();
    label2 = new java.awt.Label();
    osciloscopio1 = new javausbpid.Osciloscopio();
    jLabel3 = new javax.swing.JLabel();
    maxTime = new javax.swing.JTextField();
    maxSpeed = new javax.swing.JTextField();
    rpm = new javax.swing.JTextField();
    marcha = new javax.swing.JToggleButton();
    jButton3 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel();
    label9 = new java.awt.Label();
    label8 = new java.awt.Label();
    label3 = new java.awt.Label();
    label4 = new java.awt.Label();
    setPidConst = new javax.swing.JButton();
    kpMicro = new javax.swing.JTextField();
    kiMicro = new javax.swing.JTextField();
    kdMicro = new javax.swing.JTextField();
    kpUser = new javax.swing.JTextField();
    kiUser = new javax.swing.JTextField();
    kdUser = new javax.swing.JTextField();
```

```
jPanel3 = new javax.swing.JPanel();
label11 = new java.awt.Label();
setSetPoint = new javax.swing.JButton();
label10 = new java.awt.Label();
setPointMicro = new javax.swing.JTextField();
setPoint = new javax.swing.JTextField();
setSetPoint1 = new javax.swing.JButton();
jPanel5 = new javax.swing.JPanel();
label13 = new java.awt.Label();
error = new javax.swing.JTextField();
jPanel6 = new javax.swing.JPanel();
label14 = new java.awt.Label();
pwm = new javax.swing.JTextField();
label5 = new java.awt.Label();
jMenuBar1 = new javax.swing.JMenuBar();
jMenu2 = new javax.swing.JMenu();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Tesis");
setBackground(new java.awt.Color(255, 255, 255));
setFocusTraversalPolicyProvider(true);
setForeground(java.awt.Color.white);
setResizable(false);

jPanel2.setBackground(new java.awt.Color(153, 204, 255));

jPanel8.setBackground(new java.awt.Color(153, 204, 255));
jPanel8.setBorder(javax.swing.BorderFactory.createEtchedBorder());

usb.setIcon(new javax.swing.ImageIcon(getClass().getResource("/javausbpid/usbout.jpg"))); // NOI18N

javax.swing.GroupLayout jPanel8Layout = new javax.swing.GroupLayout(jPanel8);
jPanel8.setLayout(jPanel8Layout);
jPanel8Layout.setHorizontalGroup(
   jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(jPanel8Layout.createSequentialGroup()
      .addContainerGap()
      .addComponent(usb)
      .addContainerGap(15, Short.MAX_VALUE))
);
jPanel8Layout.setVerticalGroup(
   jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel8Layout.createSequentialGroup()
      .addContainerGap(15, Short.MAX_VALUE)
      .addComponent(usb)
```

```
            .addContainerGap())
        );

        jPanel9.setBackground(new java.awt.Color(153, 204, 255));
        jPanel9.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        label16.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        label16.setFont(new java.awt.Font("DialogInput", 1, 14));
        label16.setText("Sentido de giro");

        left.setBackground(new java.awt.Color(153, 204, 255));
        buttonGroup2.add(left);
        left.setText("Izquierda");
        left.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                leftActionPerformed(evt);
            }
        });

        rigth.setBackground(new java.awt.Color(153, 204, 255));
        buttonGroup2.add(rigth);
        rigth.setText("Derecha");
        rigth.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                rigthActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel9Layout = new javax.swing.GroupLayout(jPanel9);
        jPanel9.setLayout(jPanel9Layout);
        jPanel9Layout.setHorizontalGroup(
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(label16,                                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel9Layout.createSequentialGroup()
                        .addComponent(left)
                        .addGap(18, 18, 18)
                        .addComponent(rigth)))
                .addContainerGap(9, Short.MAX_VALUE))
        );
        jPanel9Layout.setVerticalGroup(
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addComponent(label16,                                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(1, 1, 1)
                .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(left)
                    .addComponent(rigth))
                .addContainerGap(10, Short.MAX_VALUE))
        );

        jPanel7.setBackground(new java.awt.Color(153, 204, 255));
        jPanel7.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        label15.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        label15.setFont(new java.awt.Font("DialogInput", 1, 14));
        label15.setText("Tipo de control");

        openLoop.setBackground(new java.awt.Color(153, 204, 255));
        buttonGroup1.add(openLoop);
        openLoop.setText("Lazo Abierto");
        openLoop.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                openLoopActionPerformed(evt);
            }
        });

        closeLoop.setBackground(new java.awt.Color(153, 204, 255));
        buttonGroup1.add(closeLoop);
        closeLoop.setText("Lazo Cerrado");
        closeLoop.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeLoopActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
        jPanel7.setLayout(jPanel7Layout);
        jPanel7Layout.setHorizontalGroup(
            jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel7Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(label15,                                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel7Layout.createSequentialGroup()
```

```
                .addComponent(openLoop)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(closeLoop)))
            .addContainerGap(10, Short.MAX_VALUE))
        );
        jPanel7Layout.setVerticalGroup(
            jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel7Layout.createSequentialGroup()
                .addComponent(label15,                                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(2, 2, 2)
                .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(openLoop)
                    .addComponent(closeLoop))
                .addContainerGap(10, Short.MAX_VALUE))
        );

        jButton1.setFont(new java.awt.Font("Tahoma", 1, 14));
        jButton1.setText("EXPORTAR DATOS");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        info.setBackground(new java.awt.Color(153, 204, 255));
        info.setFont(new java.awt.Font("Tahoma", 1, 14));
        info.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jPanel4.setBackground(new java.awt.Color(153, 204, 255));
        jPanel4.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        jPanel4.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N

        label12.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        label12.setFont(new java.awt.Font("DialogInput", 1, 14));
        label12.setText("Velocidad medida");

        label2.setFont(new java.awt.Font("Dialog", 1, 18));
        label2.setText("RPM");

        osciloscopio1.setName("Tesis"); // NOI18N

        javax.swing.GroupLayout osciloscopio1Layout = new javax.swing.GroupLayout(osciloscopio1);
        osciloscopio1.setLayout(osciloscopio1Layout);
        osciloscopio1Layout.setHorizontalGroup(
```

```
        osciloscopio1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 481, Short.MAX_VALUE)
    );
    osciloscopio1Layout.setVerticalGroup(
        osciloscopio1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 163, Short.MAX_VALUE)
    );

    jLabel3.setText("0");

    maxTime.setBackground(new java.awt.Color(153, 204, 255));
    maxTime.setEditable(false);
    maxTime.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    maxSpeed.setBackground(new java.awt.Color(153, 204, 255));
    maxSpeed.setEditable(false);
    maxSpeed.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    rpm.setBackground(new java.awt.Color(153, 204, 255));
    rpm.setFont(new java.awt.Font("Tahoma", 1, 18));
    rpm.setHorizontalAlignment(javax.swing.JTextField.TRAILING);
    rpm.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
    jPanel4.setLayout(jPanel4Layout);
    jPanel4Layout.setHorizontalGroup(
        jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(label12, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(jPanel4Layout.createSequentialGroup()
                    .addComponent(maxSpeed, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel4Layout.createSequentialGroup()
                            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 15,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 428,
Short.MAX_VALUE)
                            .addComponent(maxTime, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(osciloscopio1, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addGroup(jPanel4Layout.createSequentialGroup()
                    .addGap(1, 1, 1)
                    .addComponent(rpm, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(label2, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.createSequentialGroup()
                .addComponent(label12, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(osciloscopio1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(maxSpeed, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addComponent(jLabel3)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
Short.MAX_VALUE)
                        .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(label2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(rpm, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(maxTime, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        marcha.setFont(new java.awt.Font("Tahoma", 0, 14));
        marcha.setText("MARCHA");
        marcha.setActionCommand("INICIAR/PARAR");
        marcha.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                marchaActionPerformed(evt);
            }
        });
```

```
jButton3.setFont(new java.awt.Font("Tahoma", 0, 14));
jButton3.setText("CONECTAR");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jPanel1.setBackground(new java.awt.Color(153, 204, 255));
jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

label9.setText("Kp");

label8.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
label8.setFont(new java.awt.Font("DialogInput", 1, 14));
label8.setText("Constantes PID[x10^-3]");

label3.setText("Ki");

label4.setText("Kd");

setPidConst.setText("Modificar");
setPidConst.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        setPidConstActionPerformed(evt);
    }
});

kpMicro.setBackground(new java.awt.Color(153, 204, 255));
kpMicro.setEditable(false);
kpMicro.setBorder(javax.swing.BorderFactory.createEtchedBorder());

kiMicro.setBackground(new java.awt.Color(153, 204, 255));
kiMicro.setEditable(false);
kiMicro.setBorder(javax.swing.BorderFactory.createEtchedBorder());

kdMicro.setBackground(new java.awt.Color(153, 204, 255));
kdMicro.setEditable(false);
kdMicro.setBorder(javax.swing.BorderFactory.createEtchedBorder());


kpUser.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));
```

```
kiUser.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));


kdUser.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
      jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addComponent(label8, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
          .addGroup(jPanel1Layout.createSequentialGroup()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
              .addComponent(label4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
              .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(label3, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(label9, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
              .addComponent(setPidConst, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
              .addComponent(kpUser)
              .addComponent(kiUser, javax.swing.GroupLayout.DEFAULT_SIZE, 73, Short.MAX_VALUE)
              .addComponent(kdUser))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 38,
Short.MAX_VALUE)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
              .addComponent(kpMicro, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(kiMicro, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(kdMicro, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addContainerGap())
    );
    jPanel1Layout.setVerticalGroup(
```

```
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
            .addComponent(label8, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(label9, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(label3, javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(kpMicro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(kpUser, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 20,
Short.MAX_VALUE)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(kiMicro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(kiUser, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(label4, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 25, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(kdMicro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(kdUser, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(setPidConst)
            .addContainerGap())
    );

    label8.getAccessibleContext().setAccessibleName("Constantes PID[x10^-3] uC");

    jPanel3.setBackground(new java.awt.Color(153, 204, 255));
    jPanel3.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    label11.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
    label11.setFont(new java.awt.Font("DialogInput", 1, 14));
```

```
        label11.setText("Velocidad esperada");

        setSetPoint.setText("Modificar");
        setSetPoint.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                setSetPointActionPerformed(evt);
            }
        });

        label10.setText("SP");

        setPointMicro.setBackground(new java.awt.Color(153, 204, 255));
        setPointMicro.setEditable(false);
        setPointMicro.setBorder(javax.swing.BorderFactory.createEtchedBorder());


setPoint.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));

        setSetPoint1.setText("Apagar");
        setSetPoint1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                setSetPoint1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(label11,                                  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(label10,                              javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addGroup(jPanel3Layout.createSequentialGroup()
                                .addGap(9, 9, 9)
                                .addComponent(setSetPoint))
                            .addGroup(jPanel3Layout.createSequentialGroup()
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(setPoint)))
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 38,
Short.MAX_VALUE)
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                    .addComponent(setSetPoint1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(setPointMicro, javax.swing.GroupLayout.DEFAULT_SIZE, 69,
Short.MAX_VALUE))))
            .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(label11, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(label10, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(setSetPoint)
                            .addComponent(setSetPoint1)))
                    .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(setPointMicro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(setPoint, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );

        jPanel5.setBackground(new java.awt.Color(153, 204, 255));
        jPanel5.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        label13.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        label13.setFont(new java.awt.Font("DialogInput", 1, 14));
        label13.setText("Error");

        error.setBackground(new java.awt.Color(153, 204, 255));
        error.setEditable(false);
        error.setFont(new java.awt.Font("Tahoma", 1, 18));
        error.setHorizontalAlignment(javax.swing.JTextField.TRAILING);
        error.setBorder(null);
```

```
        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(label13, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(53, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel5Layout.createSequentialGroup()
                .addContainerGap(43, Short.MAX_VALUE)
                .addComponent(error, javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        jPanel5Layout.setVerticalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addComponent(label13, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(1, 1, 1)
                .addComponent(error, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(12, Short.MAX_VALUE))
        );

        jPanel6.setBackground(new java.awt.Color(153, 204, 255));
        jPanel6.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        label14.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        label14.setFont(new java.awt.Font("DialogInput", 1, 14));
        label14.setText("Pwm");

        pwm.setBackground(new java.awt.Color(153, 204, 255));
        pwm.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        pwm.setHorizontalAlignment(javax.swing.JTextField.TRAILING);
        pwm.setBorder(null);
        pwm.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                pwmActionPerformed(evt);
            }
        });

        label5.setFont(new java.awt.Font("Dialog", 0, 18));
        label5.setText("%");
```

```
        javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
        jPanel6.setLayout(jPanel6Layout);
        jPanel6Layout.setHorizontalGroup(
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel6Layout.createSequentialGroup()
                        .addComponent(pwm, javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(label5, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(label14, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(12, 12, 12))
        );
        jPanel6Layout.setVerticalGroup(
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel6Layout.createSequentialGroup()
                .addComponent(label14, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(2, 2, 2)
                .addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(label5, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(pwm, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(14, 14, 14))
        );

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 118, Short.MAX_VALUE)
                            .addComponent(jPanel6, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 118, Short.MAX_VALUE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(marcha,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel5,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addComponent(jPanel1,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel3,                javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jPanel4,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                    .addComponent(info, javax.swing.GroupLayout.DEFAULT_SIZE, 472, Short.MAX_VALUE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jPanel8,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addComponent(jPanel9,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jPanel7,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jButton1)))
            .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jPanel3,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanel1,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jPanel4,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(6, 6, 6)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel5,                    javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(jPanel9,                    javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jPanel7,                    javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1,                    javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addComponent(jPanel6,          javax.swing.GroupLayout.PREFERRED_SIZE,          62,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(marcha, javax.swing.GroupLayout.DEFAULT_SIZE, 64, Short.MAX_VALUE)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jButton3,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(info)
                    .addComponent(jPanel8,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
            .addContainerGap())
    );

    jMenuBar1.setBackground(new java.awt.Color(153, 204, 255));
    jMenuBar1.setBorder(null);

    jMenu2.setText("Credits");
    jMenuBar1.add(jMenu2);

    setJMenuBar(jMenuBar1);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel2,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel2,                    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
    );

    pack();
```

```
    }// </editor-fold>//GEN-END:initComponents

 private void generateCsvFile(int time, String rpm)
  {
            try
            {



                bw.write(String.valueOf(time));
          bw.write(";");
          if (closeLoop.isSelected()){
             bw.write(String.valueOf(VsetPoint));
          }else{
             bw.write(String.valueOf(pwm.getText()));
          }
          bw.write(";");
          bw.write(rpm);
          bw.write("\n");









            }
            catch(IOException e)
            {
                e.printStackTrace();
            }
    }

    private void setPidConstActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_setPidConstActionPerformed


      short kp=0, ki=0, kd=0;
      boolean flagkp, flagki, flagkd;

      try{
        flagkp = true;
        kp = (short) Integer.parseInt(kpUser.getText());    //Se guarda en entero la cadena que se escriba en el campo
de texto
```

```
    }
    catch(Exception e){
     flagkp = false;
     JOptionPane.showMessageDialog(this,"Dijite un numero válido para Kp");  //En caso de que se escriban caracteres distintos a numeros                                    //saldra este mensaje en una ventana de dialogo.
     kpUser.setText("");                                  // y se borrara el contenido del campo de texto
    }
//---------------------------------------------------
    try{
     flagki = true;
     ki = (short) Integer.parseInt(kiUser.getText());    //Se guarda en entero la cadena que se escriba en el campo de texto
    }
    catch(Exception e){
     flagki = false;
     JOptionPane.showMessageDialog(this,"Dijite un numero válido para Kp");  //En caso de que se escriban caracteres distintos a numeros                                    //saldra este mensaje en una ventana de dialogo.
     kiUser.setText("");                                  // y se borrara el contenido del campo de texto
    }

//---------------------------------------------------
    try{
     flagkd = true;
     kd = (short) Integer.parseInt(kdUser.getText());    //Se guarda en entero la cadena que se escriba en el campo de texto
    }
    catch(Exception e){
     flagkd = false;
     JOptionPane.showMessageDialog(this,"Dijite un numero válido para Kp");  //En caso de que se escriban caracteres distintos a numeros                                    //saldra este mensaje en una ventana de dialogo.
     kdUser.setText("");                                  // y se borrara el contenido del campo de texto
    }

    if (flagkp && flagki && flagkd){
     kp = (short) Integer.parseInt(kpUser.getText());
     ki = (short) Integer.parseInt(kiUser.getText());
     kd = (short) Integer.parseInt(kdUser.getText());
    }
    else{;}
    //Division de kp en dos bytes
    kp0 = (byte)kp;
    kp1 = (byte)(kp>>8);
```

```
    //Division de ki en dos bytes
    ki0 = (byte)ki;
    ki1 = (byte)(ki>>8);
    //Division de kd en dos bytes
    kd0 = (byte)kd;
    kd1 = (byte)(kd>>8);

    identificador='i'; //99 en decimal

   }//GEN-LAST:event_setPidConstActionPerformed


   private void leerConsPid(){


    identificador='h'; //99 en decimal


   }

   private void setSetPointActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_setSetPointActionPerformed

    leerSetPoint();

}//GEN-LAST:event_setSetPointActionPerformed


   private void soloLeerSetPoint(){

    identificador='X'; //99 en decimal
   }

   private void leerSetPoint(){
    VsetPoint = Integer.parseInt(setPoint.getText());
    if (closeLoop.isSelected()){

     if (VsetPoint>250)
        JOptionPane.showMessageDialog(this,"Dijite un valor de 0-250rpm");
     else
       identificador='d'; //99 en decimal
    }
    else{
```

```
        if (VsetPoint>100)
         JOptionPane.showMessageDialog(this,"Dijite un valor de 0-100%");
        else
        {
          identificador='d'; //99 en decimal
           VsetPoint = (int)(Integer.parseInt(setPoint.getText()));
        }
       }
      }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton3ActionPerformed
      // TODO add your handling code here:

      try{
       iface.load();      //Se carga la libreria jPicUsb.dll del directorio raiz
       info.setText("");   //Se borra lo que esta en la linea
       info.setText("Libreria jPicUsb.dll cargada con éxito"); //Se informa
       usb.setIcon(new javax.swing.ImageIcon(getClass().getResource("/javausbpid/usbin.jpg")));
      }
      catch (Exception err){
       info.setText("");             //Se borra lo que esta en la linea
       info.setText(err.getMessage()); //y se escribe el mensaje de error si lo hay.
       return;
      }
      iface.set_instance(0);
      iface.set_vidpid("vid_04d8&pid_000b"); //Se actiban las funciones de acceso rapido

      //String kp = kpUser.getText();

      //*********

      identificador='0'; //99 en decimal
      identificador_antes='0';
      readHilo = new ReadHilo();
      readHilo.start();


      //**********
     soloLeerSetPoint();

     try {
      Thread.sleep(250);
     } catch (InterruptedException ex) {
      Logger.getLogger(Connect_console.class.getName()).log(Level.SEVERE, null, ex);
```

```
    }

    leerConsPid();
//************************************************


    }//GEN-LAST:event_jButton3ActionPerformed

    private void marchaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_marchaActionPerformed
      // TODO add your handling code here:
       if (marcha.isSelected()){
         marcha.setText("PARADA");
        identificador='c';
 try
             {

         writer = new FileWriter("D:\\test.csv");
         bw = new BufferedWriter (writer);

         bw.write("Tiempo[ms]");
         bw.write(";");
         bw.write("Set Point[0-100%]");
         bw.write(";");
         bw.write("Velocidad[rpm]");
         bw.write("\n");
             }
             catch(IOException e)
             {
                 e.printStackTrace();
             }


       }
       else {
         marcha.setText("MARCHA");
         identificador = '0';
         //identificador_antes='0';

       }
    }//GEN-LAST:event_marchaActionPerformed

    private void leftActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_leftActionPerformed
      sg='0';
      identificador='k'; //99 en decimal
```

```
    }//GEN-LAST:event_leftActionPerformed

    private void rigthActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_rigthActionPerformed
      sg='1';

      identificador='j'; //99 en decimal
    }//GEN-LAST:event_rigthActionPerformed

    private void openLoopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_openLoopActionPerformed
      tc='1';
      identificador='m'; //99 en decimal

      label10.setText("PWM");
      kpUser.setEditable(false);
      kiUser.setEditable(false);
      kdUser.setEditable(false);
      setPidConst.setEnabled(false);

      if (Integer.parseInt(setPointMicro.getText().trim()) > 100) {
        JOptionPane.showMessageDialog(this,"Dijite un valor de 0-100%");
      }
    }//GEN-LAST:event_openLoopActionPerformed

    private void closeLoopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_closeLoopActionPerformed
      tc='0';
      identificador='n';

      label10.setText("SP");
      kpUser.setEditable(true);
      kiUser.setEditable(true);
      kdUser.setEditable(true);
      setPidConst.setEnabled(true);
    }//GEN-LAST:event_closeLoopActionPerformed

    private void setSetPoint1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_setSetPoint1ActionPerformed
        VsetPoint = 0;

        identificador='d'; //99 en decimal
    }//GEN-LAST:event_setSetPoint1ActionPerformed

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
```

```
FIRST:event_jButton1ActionPerformed
    // TODO add your handling code here:

  }//GEN-LAST:event_jButton1ActionPerformed

  private void pwmActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_pwmActionPerformed
    // TODO add your handling code here:
  }//GEN-LAST:event_pwmActionPerformed


  public static void main(String args[]) {              // metodo Main, es el codigo a ejecutar
                                        // Aqi se llama a la clase jFrame llamada "Connect_console"
                                        // que contiene el codigo a ejectutar
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new Connect_console().setVisible(true);     // Se configura para que sea visible.
      }
    });
  }

  // Variables declaration - do not modify//GEN-BEGIN:variables
  private javax.swing.ButtonGroup buttonGroup1;
  private javax.swing.ButtonGroup buttonGroup2;
  private javax.swing.JRadioButton closeLoop;
  private javax.swing.JTextField error;
  private javax.swing.JTextField info;
  private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton3;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JMenu jMenu2;
  private javax.swing.JMenuBar jMenuBar1;
  private javax.swing.JPanel jPanel1;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JPanel jPanel3;
  private javax.swing.JPanel jPanel4;
  private javax.swing.JPanel jPanel5;
  private javax.swing.JPanel jPanel6;
  private javax.swing.JPanel jPanel7;
  private javax.swing.JPanel jPanel8;
  private javax.swing.JPanel jPanel9;
  private javax.swing.JTextField kdMicro;
  private javax.swing.JTextField kdUser;
  private javax.swing.JTextField kiMicro;
  private javax.swing.JTextField kiUser;
  private javax.swing.JTextField kpMicro;
```

```
    private javax.swing.JTextField kpUser;
    private java.awt.Label label10;
    private java.awt.Label label11;
    private java.awt.Label label12;
    private java.awt.Label label13;
    private java.awt.Label label14;
    private java.awt.Label label15;
    private java.awt.Label label16;
    private java.awt.Label label2;
    private java.awt.Label label3;
    private java.awt.Label label4;
    private java.awt.Label label5;
    private java.awt.Label label8;
    private java.awt.Label label9;
    private javax.swing.JRadioButton left;
    private javax.swing.JToggleButton marcha;
    private javax.swing.JTextField maxSpeed;
    private javax.swing.JTextField maxTime;
    private javax.swing.JRadioButton openLoop;
    private javausbpid.Osciloscopio osciloscopio1;
    private javax.swing.JTextField pwm;
    private javax.swing.JRadioButton rigth;
    private javax.swing.JTextField rpm;
    private javax.swing.JButton setPidConst;
    private javax.swing.JTextField setPoint;
    private javax.swing.JTextField setPointMicro;
    private javax.swing.JButton setSetPoint;
    private javax.swing.JButton setSetPoint1;
    private javax.swing.JLabel usb;
    // End of variables declaration//GEN-END:variables
  FileWriter writer;
  BufferedWriter bw;
   public static byte identificador;
   public static byte identificador_antes;
   public static int VsetPoint=0;
   byte kp0,kp1,ki0,ki1,kd0,kd1,sg,tc;



   public ReadHilo readHilo;
   public class ReadHilo extends Thread {          // Este metodo se ejecutara siembre que la
                                        // variable booleana "onOFF" tenga el valor TRUE
    boolean onOFF=true;                    // se define la variable booleana que
  //-----------------------------------------------
    int xx=0;
```

```
int yy=0;
int x1[];
String respuesta;//condiciona el ciclo infinito
String respuesta2;
String respuesta3;
String varRpm;
String varPwm;
int TmrTest = 0,myTimer=0,cnt=0;


@Override
public void run(){

 long tiempoInicial = System.currentTimeMillis();


   while(onOFF) {
    //System.out.println("ooooooooooo");
    if(identificador!='0') {
       //***********lectura  de pic


     byte[] comando = {identificador,(byte)VsetPoint,kp1,kp0,ki1,ki0,kd1,kd0,sg,tc};

//System.out.println(identificador+".."+VsetPoint+".."+kp1+".."+kp0+".."+ki1+".."+ki0+".."+kd1+".."+kd0+".."+sg);

     iface.QWrite(comando, 32, 1000);
            try {
               respuesta = new String(iface.QRead(32, 500), "utf-8");
            } catch (UnsupportedEncodingException ex) {
               Logger.getLogger(Connect_console.class.getName()).log(Level.SEVERE, null, ex);
            }
      //System.out.println(respuesta);
      // System.out.println(respuesta.length());


     if(respuesta.length()!=0)
     {
     //**********GRAFICAR
     if(respuesta.charAt(0)=='c') {


       respuesta3=respuesta.substring(1, respuesta.length());

       //System.out.println("verificar espacios? --> "+respuesta);
```

```
            Scanner rpmPwm = new Scanner(respuesta3);
            rpmPwm.useDelimiter("-");

            while(rpmPwm.hasNext()){
               varRpm = rpmPwm.next().trim();
               varPwm = rpmPwm.next().trim();
            }
 //System.out.println("RRPPMM: "+varRpm);
            error.setText(String.valueOf(Integer.parseInt(setPointMicro.getText().trim())-
Integer.parseInt(varRpm.trim())));
             try{
            xx=(int)(System.currentTimeMillis()-tiempoInicial);
              }catch (Exception e){
//System.out.println("Error");
              }

            yy = Integer.parseInt(varRpm);

             //System.out.println("Error xxxxxxxxxxxxxxx");
            rpm.setText(varRpm);
            pwm.setText(String.valueOf(Integer.parseInt(varPwm)));
            int x[]={xx,yy};
            myv.addElement(x);
            osciloscopio1.dibujando(myv);

            try
            {


           if(marcha.getText()=="PARADA")
           {


             generateCsvFile(xx,varRpm);

              }
           else
           {
            bw.close(); }


            }
            catch(Exception e){ System.out.println(e);}
```

```
  respuesta="0";
 }

 //**********giro Derecha -
 if(respuesta.charAt(0)=='j') {

  respuesta="0";
  identificador=identificador_antes;
 }

 //**********giro Izquierda -
 if(respuesta.charAt(0)=='k') {

  respuesta="0";
  identificador=identificador_antes;
 }

 //**********Lazo abierto
 if(respuesta.charAt(0)=='m') {

  respuesta="0";
  identificador=identificador_antes;
 }

 //**********Lazo cerrado -
 if(respuesta.charAt(0)=='n') {

  respuesta="0";
  identificador=identificador_antes;
 }

//************setsetponit
 if(respuesta.charAt(0)=='d') {

   setPointMicro.setText(respuesta.substring(1, respuesta.length()));
   respuesta="0";
   identificador=identificador_antes;
  }
  //********************constantes pid
  if(respuesta.charAt(0)=='h') {
   respuesta2=respuesta.substring(1, respuesta.length());
   Scanner s = new Scanner(respuesta2);

   s.useDelimiter("-");
```

```
        while(s.hasNext()){

          kpMicro.setText(s.next());
          kiMicro.setText(s.next());
          kdMicro.setText(s.next());
        }
        respuesta="0";
        identificador=identificador_antes;
        }
        //******************
        //*********************
        if (marcha.isSelected()){  identificador_antes='c'; }

        //System.out.println("Error: yyyyyyyyyyyyyyyyy");
          //****************************
        }
        }
       }

    }
  }
}
```

ANEXO C: Manual de operación

1. CONFIGURACIÓN DEL SISTEMA EMBEBIDO.
2. CONFIGURACIÓN DEL APLICATIVO JAVA.
3. DESCRIPCIÓN DE HERRAMIENTAS DEL APLICATIVO.
4. INSTRUCCIONES PARA EL USO DEL APLICATIVO.
5. RELACIONES IMPORTANTES ENTRE LOS CÓDIGOS FUENTE.

1. CONFIGURACIÓN DEL SISTEMA EMBEBIDO.

Para usar el sistema embebido por primera vez en una computadora, es necesario realizar los siguientes pasos:

a) Conectar el sistema embebido al PC y esperar su detección (sonido característico de detección de dispositivo USB).

Figura 32. Instante en que Windows detecta que el PIC fue conectado al PC.

Al hacer esto aparece el mensaje que se muestra en la Figura 1.

b) Luego se instala el driver para este dispositivo, de la siguiente manera:

Figura 33. Paso 1 en la instalación del driver

Figura 34. Paso 2 en la instalación del driver

Se escribe la ruta de ubicación del driver en el DVD del proyecto con E:\05-PROGRAMACION Y SIMULACION\PIC18F4550\DRIVERS

Figura 35. Paso 3 en la instalación del driver.

Una vez finalizado el proceso de instalación, se verifica en el panel de control/sistema/administrador de dispositivos si el driver fue instalado correctamente o si tiene algún conflicto. Debe aparecer como se muestra en la Figura 5.

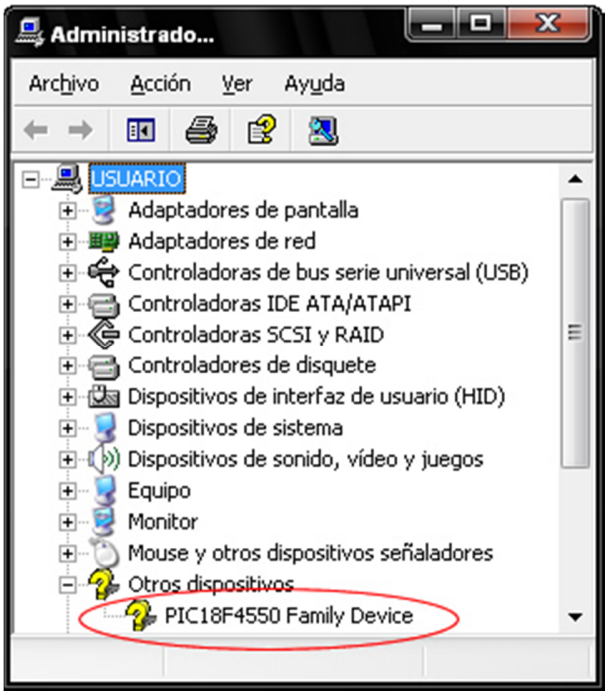


Figura 36. Paso 4 - Verificación de la instalación

Después de esto el sistema embebido está listo para usarse.

2. CONFIGURACIÓN DEL APLICATIVO JAVA.

Para que el aplicativo de Java pueda ejecutarse correctamente debe estar instalado el Runtime de Java que se descarga gratis de la web y el Runtime de Visual C++, de lo contrario Java nunca cargará la librería jpicusb.dll. Esta se encuentra en esta ubicación dentro del DVD

E:\00-SOFTWARE NECESARIO\JAVA\plug in Visual C++\vcredist_x86.exe

En segundo lugar la librería jpicusb.dll debe estar guardada en la misma carpeta raíz donde este el ejecutable JavaUsbPid.jar. Esta librería la podemos encontrar en la siguiente ruta en el DVD.

E:\05-PROGRAMACION Y SIMULACION\APLICATIVO EN JAVA\librería jpicusb 1.1.1\

Una vez hecho esto el sistema está completamente configurado y listo para la manipulación del usurario.

3. DESCRIPCIÓN DE HERRAMIENTAS DEL APLICATIVO

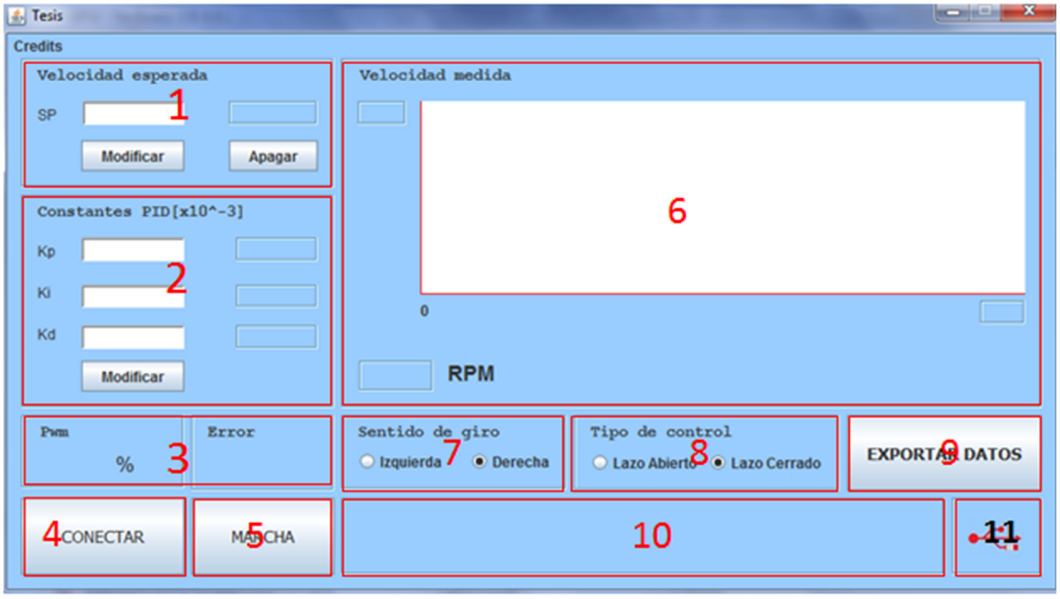


Figura 37. Descripción de herramientas del aplicativo Java.

1	Configuración (campo blanco) y Monitoreo (campo azul) del SetPoint.
2	Configuración (campo blanco) y Monitoreo (campo azul) de constantes PID.
3	Monitoreo de error y de porcentaje de ciclo útil del PWM.
4	Botón para iniciar la comunicación bidireccional.
5	Botón para iniciar y parar la captura de datos.
6	Área de monitoreo gráfico y acumulativo de la variable controlada.
7	Selección del sentido de giro del motor.
8	Selección de tipo de control Lazo abierto o cerrado.
9	Botón para exportar datos al disco duro en formato Excel.
10	Zona de notificaciones
11	Indicación de conexión USB

4. INSTRUCCIONES PARA EL USO DEL APLICATIVO.

a) Conectar el sistema embebido al PC mediante el cable USB.
b) Esperar detección por parte de Windows.
c) Ejecutar la aplicación de Java.
d) Pulsar en el botón “CONECTAR” para establecer la comunicación USB entre ambas partes.
e) Hacer click en el botón “Apagar” para llevar el motor a velocidad inicial cero, para la nueva práctica.(opcional).
f) Dar click en “MARCHA” para inicializar la captura de datos en el gráfico.
g) Establecer un valor de SetPoint entre 0-250 RPM y se le da click en Modificar.

Para modificar las constantes PID basta con escribir valores en los tres campos de kp, ki y kd y luego dar click en modificar.
Para exportar datos a Excel se debe dar click en “PARADA”, y luego en “EXPORTAR DATOS” este archivo debe buscarse en el disco “C:”

5. RELACIONES IMPORTANTES ENTRE LOS CÓDIGOS FUENTE.

Las relaciones entre los códigos del PIC y Java están definidas por la comunicación USB.

Para la programación de una comunicación USB desde la perspectiva del PIC es necesario tener en cuenta algunos elementos:

#fuses HSPLL y PLL5: La frecuencia de oscilación necesaria para el USB 2.0 es de 48Mhz. Como en este proyecto se usa un cristal de cuarzo de 20Mhz se necesita hacer uso del módulo PLL interno del PIC. Para ello utilizamos el fuse HSPLL. Como el módulo PLL requiere una oscilación de entrada de 4Mhz debemos utilizar el divisor 1:5 indicado con el fuse PLL5 para obtener los 20/5 = 4Mhz requeridos.

USB_ENABLE_BULK y SIZE 32: Para activar el método de transferencia masiva mediante el USB debemos configurar los EndPoint de transmisión y recepción, **USB_EP1_TX_ENABLE** y **USB_EP1_RX_ENABLE**, indicándolo con la constante **USB_ENABLE_BULK**. Es imprescindible deshabilitar el método HID (Human Interface Device). El tamaño del buffer de transferencia lo podemos ajustar desde 1 a 32 bytes como máximo. Para este proyecto se establecerá el máximo de 32 bytes por envío o recepción de paquetes USB.

Una vez iniciada la ejecución del método principal **`main()`**, es necesario inicializar y operar el modulo USB mediante las funciones **`usb_init()`**, **`usb_task()`**, **`usb_wait_for_enumeration()`**, **`usb_enumerated()`**, **`usb_kbhit()`**, **`usb_get_packet()`** y **`usb_put_packet()`**. Estas están disponibles en la librería que proporciona el compilador CCS C para el manejo del USB 2.0 y vienen

definidas e implementadas en los **includes pic18_usb.h, usb.c y usb.h** que se encuentran en el directorio de instalación de CCS "*Archivos de Programa\PICC\Drivers\...*".

usb_init(), usb_task() y usb_wait_for_enumeration() se utilizan solo para establecer la comunicación y se ejecutan únicamente tras un reset del micro.

Si ya ha podido el programa entrar al bucle infinito **while(true)**, solo se puede actuar si la función **usb_enumerated()** nos devuelve true, o sea que estamos correctamente conectados y reconocidos por el Windows del PC.

A partir de este punto solo queda esperar a recibir un comando proveniente del PC. Esto se detecta mediante la función **usb_kbhit()** que al devolvernos true nos indicará que se tiene algo pendiente de recibir. Y los vamos a recoger mediante **usb_get_packet()** quedando disponible en **recbuf**.

Para contestar o enviar datos al PC se usa la función **usb_put_packet().**

Desde la perspectiva de Java la comunicación USB gira alrededor de una clase de Java llamada jPicUSB. Esta clase permite a una aplicación Java hacer llamados a la librería dinámica "jpicusb.dll" la cual implementa todas las funciones de la API USB de Microchip "mpusbapi.dll", con la diferencia de que esta especialmente recompilada para permitir a la clase jPicUSB hacer llamados a sus funciones.

Para el uso de la clase jPicUSB son necesario tener en cuenta cuatro elementos básicos:

5. Es necesario antes de usar cargar la librería “jPicUSB.jar”

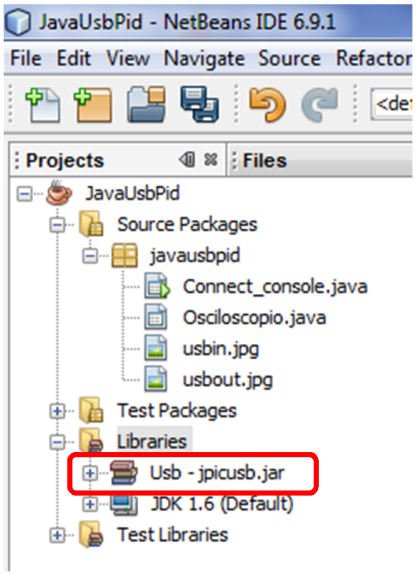


Figura 38. Carga de librería

6. Importar a la aplicación de java todos los paquetes contenidos en la clase jPicUSB.

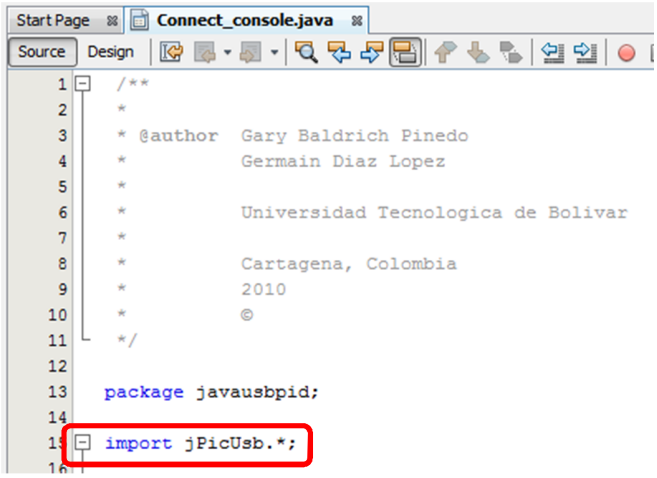


Figura 39. Importación de los métodos jPicUSB

7. Inicializar la librería en el código antes de usarla.

```
try{
  iface.load();       //Se carga la libreria jPicUsb.dll del directorio raiz
  info.setText("");   //Se borra lo que esta en la linea
  info.setText("Libreria jPicUsb.dll cargada con éxito"); //Se informa
  usb.setIcon(new javax.swing.ImageIcon(getClass().getResource("/javausbpid/usbin.jpg")));
}
catch (Exception err){
  info.setText("");               //Se borra lo que esta en la linea
  info.setText(err.getMessage()); //y se escribe el mensaje de error si lo hay.
  return;
}
iface.set_instance(0);
iface.set_vidpid("vid_04d8&pid_000b"); //Se actiban las funciones de acceso rapido
```

Figura 40.

Ilustración de los llamados para iniciar la librería

8. Copiar la librería jpicsb.dll a la carpeta donde se guarda el ejecutable del programa, antes de ejecutarlo.

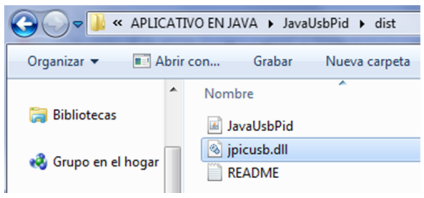


Figura 41. Ilustración para indicar donde se guarda la librería

En el caso de este proyecto java está constantemente enviando una trama a el PIC mediante el método `iface.QWrite()`de jPicUSB.

```
byte[] comando = {identificador,(byte)VsetPoint,kp1,kp0,ki1,ki0,kd1,kd0,sg,tc};

iface.QWrite(comando, 32, 1000);
```

Para recibir información del PIC se hace de esta manera:

```
respuesta = new String(iface.QRead(32, 500), "utf-8");
```

Java envía siempre delante de la trama un identificador que le indica al PIC que valores debe leer en ese envío.

Por otro lado el PIC le envía a Java: constantes PID, SetPoint, RPM y %PWM. Los dos primeros se envían solo cuando es solicitado mas los dos segundos se envían

constantemente.

```
usb_put_packet(1,hdr_rpm,4,USB_DTS_TOGGLE);
```

Y para recibir y guardar en **recbuf** con **Lenbuf** número de bytes

```
usb_get_packet(1, recbuf, Lenbuf);
```