ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"MANEJO DE UN DECODIFICADOR ÓPTICO (ENCODER) EN APLICACIÓN CON UN dsPIC"

PROYECTO DE LA MATERIA DE GRADUACIÓN

MICROCONTROLADORES AVANZADOS

PREVIA A LA OBTENCIÓN DEL TITULO DE:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRONICA Y AUTOMATIZACIÓN INDUSTRIAL

PRESENTADO POR:

JAVIER LENIN MEJÍA MENDIETA

WASHINGTON PAUL REINA LOAIZA

GUAYAQUIL – ECUADOR

AÑO 2009

AGRADECIMIENTO

Agradecemos a Dios y a todas aquellas personas que de una u otra manera colaboraron con su esfuerzo, dedicación, cariño y buena voluntad para la realización de este proyecto.

También a nuestros padres por su esfuerzo continuo para que podamos llegar a la culminación de una nueva meta en nuestras vidas.

Y por último también es necesario resaltar la ayuda brindada por aquellas personas que no conocemos pero que con sus conocimientos nos han brindado la información necesaria a través de la red de Internet.

DEDICATORIA

A todas aquellas personas a las cuales podemos brindar con nuestros conocimientos adquiridos, un poco de ayuda e información acerca del trabajo realizado; con la esperanza de que este sea un inicio para mejoras futuras y con la certeza de contribuir con el desarrollo del aprendizaje de los microcontroladores con esta aplicación de utilidad industrial.

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

JAVIER LENIN MEJÍA MENDIETA

WASHINGTON PAUL REINA LOAIZA

TRIBUNAL DE GRADUACIÓN

Ing. Carlos Valdivieso
Instructor de la Materia de Graduación

Ing. Hugo Villavicencio
Delegado del Decano

RESUMEN

Mediante el Control Automático en la actualidad, se ha revolucionado los procesos industriales en el mundo, permitiéndonos así mirar y avanzar al futuro con paso firme y seguro, con una mayor eficacia y rapidez en la producción.

En este proyecto se tratará esencialmente en una aplicación de Control Automático, que presentará el control PID de un motor DC. Utilizando como sistema de medición un juego de sensores que tomarán los datos de un encoder óptico y como controlador un dsPIC.

Se lo realizará en un montaje sencillo para ser utilizado como instrumento del laboratorio, y de esta manera dejar una herramienta didáctica y de fácil manejo para la enseñanza de los microcontroladores.

Y finalmente vamos a crear una interfaz tanto en Visual Basic como en la plataforma sistematizada LabView para poder visualizar los datos transmitidos por el microcontrolador y al mismo tiempo graficarlos en tiempo real.

INDICE GENERAL

ABREVIATURAS

dsPIC	Microcontrolador de Microchip con soporte para procesamiento de señales
DC	Corriente directa
A	Amperio
bit	Unidad de medida de información equivalente a la elección entre dos posibilidades igualmente probables
cm.	Centímetro
kOhm	Kilo-Ohmio
ohm	Ohmio
MHz	Mega hertz
PWM	Modulación de ancho de pulso
V	Voltio
PID	Proporcional Integral Derivativo
inc.	Incremento
dec.	Decremento
RPM	Revoluciones por minuto

INDICE DE IMÁGENES

INTRODUCCION

El trabajo presentado en este proyecto forma parte de la materia de graduación “Microcontroladores Avanzados” y consiste en el “Manejo de un Decodificador Óptico (Encoder) en aplicación con un dsPIC”.

En el capítulo uno de este informe trataremos acerca de los objetivos del proyecto a realizarse, como son: el control PID de un motor DC, por que realizarlo, cuales son sus aplicaciones en la industria, cuales son sus ventajas y desventajas.
También estableceremos el método a usarse para el desarrollo del este proyecto.

En el capítulo dos se abordará el tema de los equipos utilizados (hardware), sus características mas importantes, las estrategias utilizadas, los detalles de construcción y un análisis total de los costos de nuestro dispositivo.

En el capítulo tres nos adentramos a la programación de nuestro controlador, dando a conocer el código fuente utilizado y el diagrama de flujo correspondiente para su fácil comprensión.

En el cuarto capítulo se observará muestra las pruebas realizadas, curvas características y resultados que respaldan este documento.

Finalmente se presentan las conclusiones y recomendaciones, como valoración de los conocimientos aprendidos durante el desarrollo de este proyecto.

CAPITULO 1

ANTECEDENTES

1.1 OBJETIVOS DEL PROYECTO

La materia de graduación "Microprocesadores Avanzados", tuvo como objetivo profundizar en el manejo de microcontroladores y sus diferentes aplicaciones; realizando diferentes proyectos para cada uno de los cuales se realizaba no sólo una simulación de resultados sino también su respectivo montaje físico e implementación de software en el lenguaje aprendido como fue el Mikrobasic.

Se aprendió también el manejo de lenguaje visualizador para obtención de resultados y graficaciones, como es el Visual Basic; además se recibió la instrucción sobre la plataforma LabView, que también se utilizará como representador de resultados obtenidos.

Y para reforzar el conocimiento y crear en cada persona el aspecto de ingenio e investigación, este proyecto final ha sido desarrollado usando la línea de microcontroladores para procesamiento de señales conocidos como dsPICs orientando el análisis hacia el control de motores y sensores.

Con esto se espera finalmente poder unir todos los conocimientos aprendidos e investigados para realizar este proyecto final, que consta de la representación didáctica para la aplicación del control de velocidad de un motor DC por el método PID; que en aplicaciones industriales no sólo se controla la velocidad sino que se limita o se gobierna a un motor a trabajar al ritmo necesario en una función especifica, según el uso lo requiera.

1.1.1 CONTROL DE MOTOR CON UN PID

Un PID (Proporcional Integral Derivativo) es un mecanismo de control por realimentación que se utiliza en sistemas de control industriales. Un controlador PID corrige el error entre un valor medido y el valor que se quiere obtener calculándolo y luego sacando una acción correctora que puede ajustar al proceso acorde. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional determina la reacción del error actual. El Integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El Derivativo determina la reacción al cambio del error producido. La suma de estas tres acciones es usada para ajustar al proceso vía un elemento de control como la posición de una válvula de control o la energía suministrada a un calentador. Ajustando estas tres constantes en el algoritmo de control del PID, el controlador puede proveer un control para el proceso a realizar. La reacción del controlador puede ser descrita en términos de la respuesta del control ante un error, su esfuerzo por llegar al "set point", y el grado de oscilación del sistema. Nótese que el uso del PID para control no garantiza control óptimo del sistema o la estabilidad del mismo.

Tal control PID se rige por la siguiente formula:

$$\text{Control PID} = B + (\text{Kp.E}) + (Ki\int Et) + \left(Kd * \frac{\Delta E}{\Delta t}\right)$$

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

Un sensor, que determine el estado del sistema.

Un controlador, que genere la señal que gobierna al actuador.

Un actuador, que modifique al sistema de manera controlada.

El sensor proporciona una señal analógica o digital 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 de corriente eléctrica o frecuencia. En este último caso la señal es de corriente alterna, a diferencia de los dos anteriores, que son con corriente continua.

El controlador lee una señal externa que representa el valor que se desea alcanzar. Esta señal recibe el nombre de punto de consigna (o punto de referencia), la cual 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 (HMI-Human Machine Interface), son pantallas de gran valor visual y fácil manejo que se usan para hacer más intuitivo el control de un proceso.

El controlador resta la señal de punto actual a la señal de punto de consigna, obteniendo así la señal de error, que determina en cada instante la diferencia que hay entre el valor deseado (consigna) y el valor medido. La señal de error es utilizada por cada uno de los 3 componentes del controlador PID. Las 3 señales sumadas, componen la señal de salida que el controlador va a utilizar para gobernar al actuador. La señal resultante de la suma de estas tres se llama variable manipulada y no se aplica directamente sobre el actuador, si no que debe ser transformada para ser compatible con el actuador que usemos.

Las tres componentes de un controlador PID son: parte Proporcional, acción Integral y acción Derivativa. 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. Se pretenderá lograr que el bucle de control corrija eficazmente y en el mínimo tiempo posible los efectos de las perturbaciones.

CONTROL PROPORCIONAL.

La parte proporcional consiste en el producto entre la señal de error y la constante proporcional como para que hagan que el error en estado estacionario sea casi nulo, pero en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo, existe también un valor límite en la constante proporcional a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobre-oscilación y, por razones de seguridad, no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional ni siquiera produzca sobre-oscilación. Hay una relación lineal continua entre el valor de la variable controlada y la posición del elemento final de control (la válvula se mueve al mismo valor por unidad de desviación). La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo, es incluyendo y configurando las acciones integral y derivativa.

La fórmula del proporcional esta dada por:

$$P_{\text{sal}} = K_p\, e(t)$$

El error, la banda proporcional y la posición inicial del elemento final de control se expresan en tanto por uno. Nos indicará la posición que pasará a ocupar el elemento final de control.

CONTROL INTEGRAL.

El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el modo proporcional. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El error es integrado, lo cual tiene la función de promediarlo o sumarlo por un periodo de tiempo determinado; Luego es multiplicado por una constante I. I representa la constante de integración. Posteriormente, la respuesta integral es adicionada al modo Proporcional para formar el control P + I con el propósito de obtener una respuesta estable del sistema sin error estacionario.

El modo integral presenta un desfasamiento en la respuesta de 90º que sumados a los 180º de la retroalimentación (negativa) acercan al proceso a tener un retraso de 270º, luego entonces solo será necesario que el tiempo muerto contribuya con 90º de retardo para provocar la oscilación del proceso. <<< la ganancia total del lazo de control debe ser menor a 1, y así inducir una atenuación en la salida del controlador para conducir el proceso a estabilidad del mismo. >>> Se caracteriza por el tiempo de acción integral en minutos por repetición. Es el tiempo en que delante una señal en escalón, el elemento final de control repite el mismo movimiento correspondiente a la acción proporcional.

El control integral se utiliza para obviar el inconveniente del offset (desviación permanente de la variable con respeto al punto de consigna) de la banda proporcional.

La formula del integral esta dada por:

$$Isal = Ki\int_{0}^{t} e(\tau)d\tau$$

CONTROL DERIVATIVO.

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error; (si el error es constante, solamente actúan los modos proporcional e integral).

El error es la desviación existente entre el punto de medida y el valor consigna, o "SetPoint".

La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente.

Se deriva con respecto al tiempo y se multiplica por una constante D y luego se suma a las señales anteriores (P+I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor derivativa corresponde a un cambio más rápido y el controlador puede responder acordemente.

La fórmula del derivativo esta dada por:

$$D_{\mathrm{sal}} = K_d \frac{de}{dt}$$

El control derivativo se caracteriza por el tiempo de acción derivada en minutos de anticipo. La acción derivada es adecuada cuando hay retraso entre el movimiento de la válvula de control y su repercusión a la variable controlada.

Cuando el tiempo de acción derivada es grande, hay inestabilidad en el proceso. Cuando el tiempo de acción derivada es pequeño la variable oscila demasiado con relación al punto de consigna. Suele ser poco utilizada debido a la sensibilidad al ruido que manifiesta y a las complicaciones que ello conlleva.

El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones.

1.1.2 CAMPOS DE APLICACIÓN

Este tipo de control es muy utilizado a nivel industrial, especialmente en aquellos procesos en los que se requiere una exactitud mayor a los controladores proporcional, proporcional derivativo y proporcional integral se utiliza en aplicaciones más cruciales tales como control de presión, flujo, fuerza, velocidad, en muchas aplicaciones química, y otras variables. Además es utilizado en reguladores de velocidad de automóviles (control de crucero o cruise control), control de ozono residual en tanques de contacto.

Como por ejemplo; si deseáramos controlar el caudal de un flujo de entrada en un reactor químico. En primer lugar tendremos que poner una válvula de control del caudal de dicho flujo, con la finalidad de tener una medición constante del valor del caudal que circule. El controlador irá vigilando que el caudal que circule sea el establecido por nosotros, y en el momento que detecte un error, mandará una señal a la válvula de control de modo que esta se abrirá o cerrará corrigiendo el error medido. Y tendremos de ese modo el flujo deseado y necesario. El PID, es un cálculo matemático, lo que manda la información es el PLC.

Otro ejemplo lo podrías encontrar si deseamos mantener la temperatura interna de un reactor químico en su valor de referencia. Deberemos tener un dispositivo de control de la temperatura (ya puede ser un calentador, una resistencia eléctrica,...), y un sensor (termómetro). El P, PI o PID irá controlando la variable (en este caso la temperatura). En el instante que esta no sea la correcta avisará al dispositivo de control de manera que este actúe, corrigiendo el error.

1.2 DESCRIPCION.

Este proyecto tiene la finalidad de realizar un control PID de un motor DC cuya variable controlada será la velocidad. La programación del software se lo realizara en Mikrobasic utilizando la tecnología de los dsPIC's (en nuestro caso particular el dsPic30F4011), y tomando como base la nota de Aplicación AN937 de Microchip, siguiendo así una forma estandarizada de control en lazo cerrado y de cálculo de errores.

1.2.1 ESPECIFICACIONES TECNICAS DEL SISTEMA A IMPLEMENTARSE.

La lectura de la velocidad del motor se realiza a través de un encoder fijado al eje del motor, del cual al tomar las varias lecturas se enviarán directamente a una PC y se mostrarán las gráficas del proceso para diferentes situaciones de sintonización del control PID. En cada caso los valores de ganancia proporcional Kp, ganancia integral Ki, ganancia derivativa Kd y del SetPoint de velocidad (a la cual se desea controlar) son ingresados por el teclado y visualizados en la pantalla del LCD.

Este proceso de control se lo realizará por medio de tres hilos (líneas) de lecturas realizadas en los sensores de emisión-recepción infrarrojos los cuales detectan las lecturas (obstáculos de luz) en el encoder, tanto para el movimiento o giro y la ubicación del motor.

El encoder de lectura, será realizado en impresiones de láminas de acetato; de esta manera se da la facilidad para su fabricación y se contará con varios tipos de lectura comparables pues se presentan con diferentes divisiones.

También el desarrollo nos indicará el uso de dos valores distintos de fuentes, la de alimentación para el motor y la de polarización de los integrados utilizados. Para ello se ha previsto la utilización tan solo de una fuente de 9 V., para la alimentación del motor (según las especificaciones técnicas del mismo), la cual tiene una corriente de aproximada de 1.2 A., y para la polarización de los demás elementos del sistema se a utilizado un dispositivo de regulación de voltaje positivo de 3 terminales, obteniendo así 5 V. La corriente que ocupa el motor será aproximadamente de 1 A., siendo entonces la restante ocupada por el resto del circuito.

Los programas de visualización que se utilizarán para realizar las gráficas con los resultados obtenidos serán: Visual Basic y LabView. (Las mismas podrán ser observadas en el ANEXO F)

Los datos serán enviados directamente a través del RS232 (puerto serial).

Adicionalmente en el programa de Visual Basic se podrá almacenar los datos obtenidos en una hoja de cálculo en Excel, para poder llevar control de lo que se desea obtener o comparar datos en diferentes etapas o tiempos de lectura.

1.2.2 DIAGRAMA DE BLOQUES

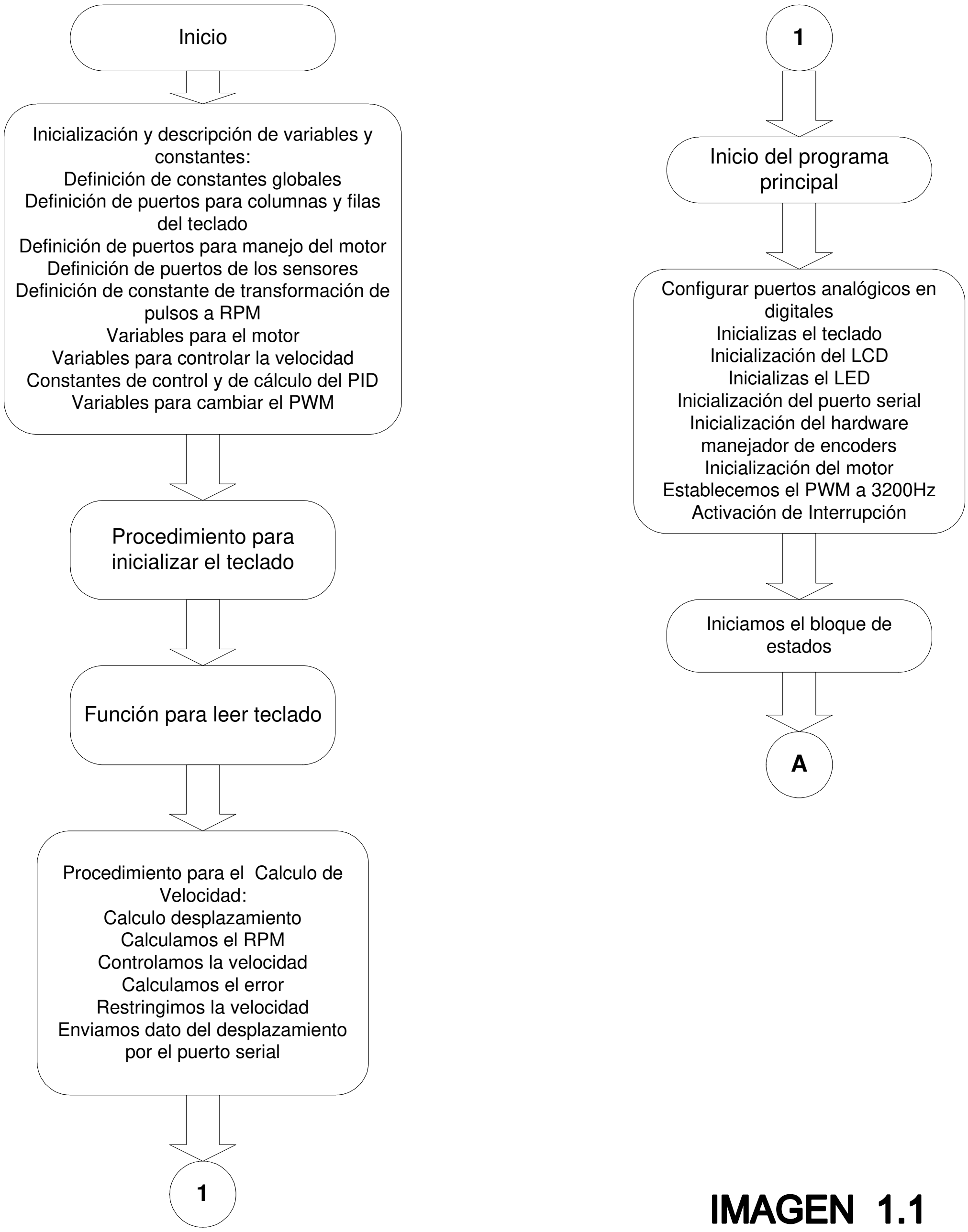


IMAGEN 1.1

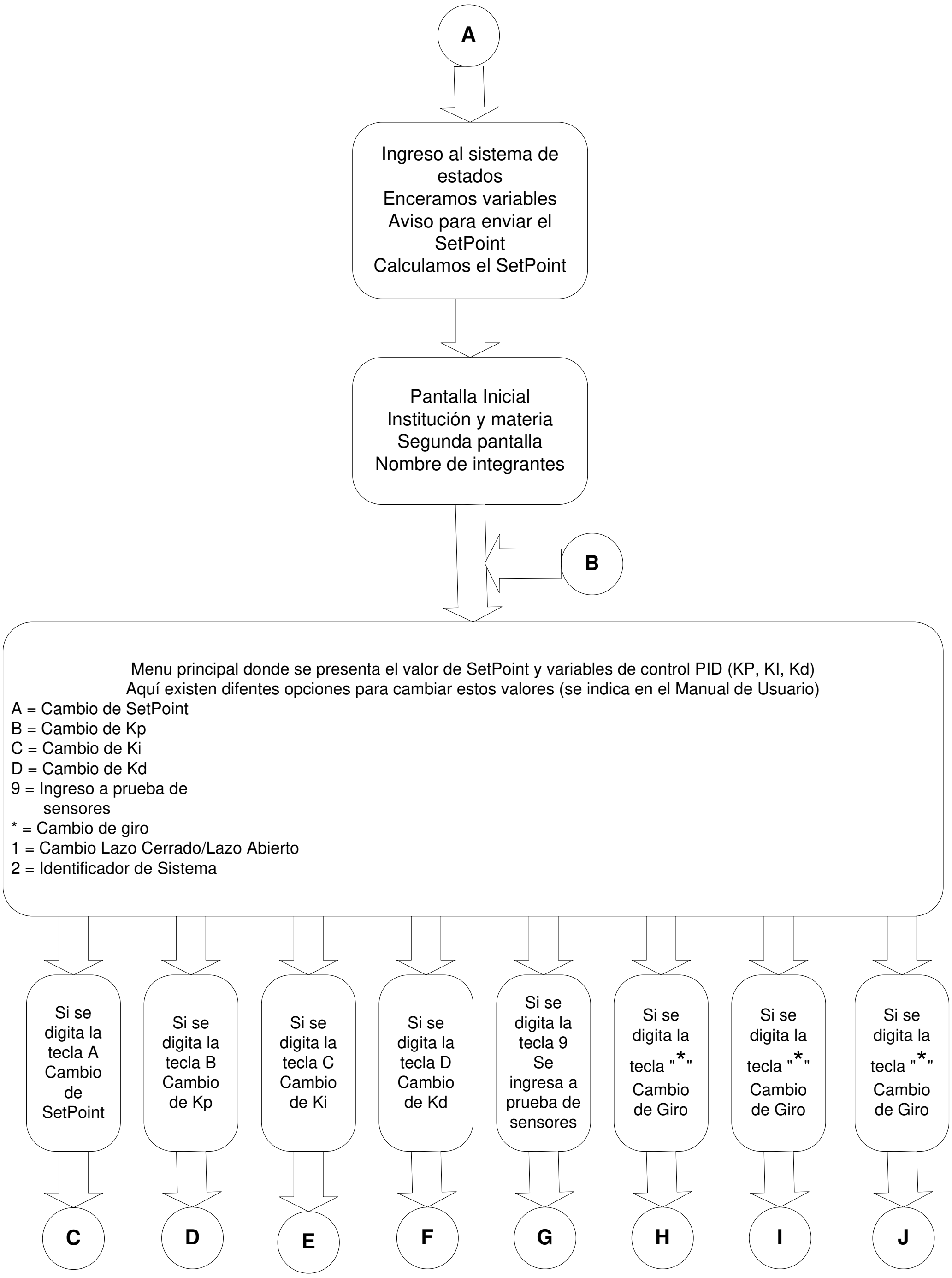

A
Ingreso al sistema de estados
Enceramos variables
Aviso para enviar el SetPoint
Calculamos el SetPoint
Pantalla Inicial
Institución y materia
Segunda pantalla
Nombre de integrantes
B
Menu principal donde se presenta el valor de SetPoint y variables de control PID (KP, KI, Kd)
Aquí existen difentes opciones para cambiar estos valores (se indica en el Manual de Usuario)
A = Cambio de SetPoint
B = Cambio de Kp
C = Cambio de Ki
D = Cambio de Kd
9 = Ingreso a prueba de sensores
* = Cambio de giro
1 = Cambio Lazo Cerrado/Lazo Abierto
2 = Identificador de Sistema
Si se digita la tecla A Cambio de SetPoint
Si se digita la tecla B Cambio de Kp
Si se digita la tecla C Cambio de Ki
Si se digita la tecla D Cambio de Kd
Si se digita la tecla 9 Se ingresa a prueba de sensores
Si se digita la tecla "*" Cambio de Giro
Si se digita la tecla "*" Cambio de Giro
Si se digita la tecla "*" Cambio de Giro
C
D
E
F
G
H
I
J

IMAGEN 1.2

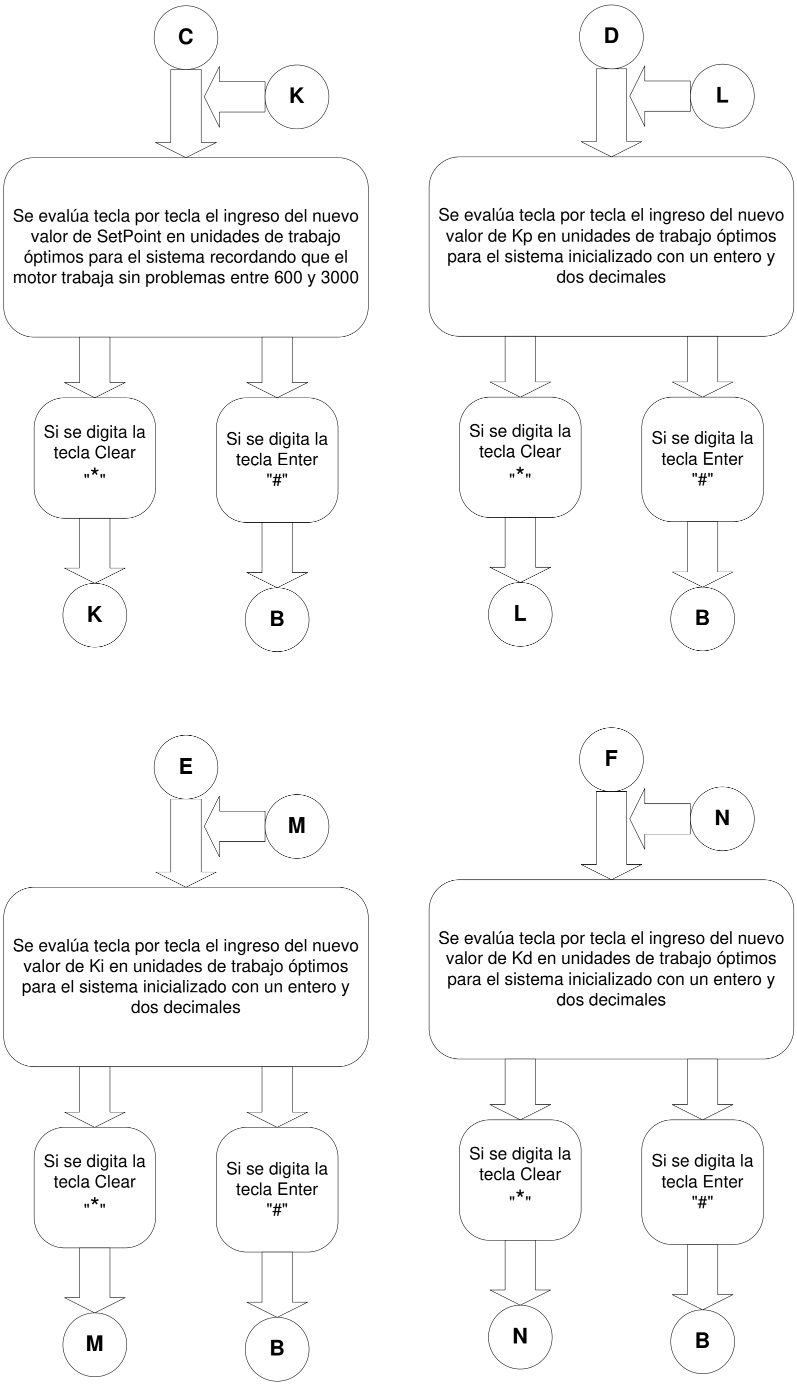

C
K
Se evalúa tecla por tecla el ingreso del nuevo valor de SetPoint en unidades de trabajo óptimos para el sistema recordando que el motor trabaja sin problemas entre 600 y 3000
Si se digita la tecla Clear "*"
Si se digita la tecla Enter "#"
K
B
D
L
Se evalúa tecla por tecla el ingreso del nuevo valor de Kp en unidades de trabajo óptimos para el sistema inicializado con un entero y dos decimales
Si se digita la tecla Clear "*"
Si se digita la tecla Enter "#"
L
B
E
M
Se evalúa tecla por tecla el ingreso del nuevo valor de Ki en unidades de trabajo óptimos para el sistema inicializado con un entero y dos decimales
Si se digita la tecla Clear "*"
Si se digita la tecla Enter "#"
M
B
F
N
Se evalúa tecla por tecla el ingreso del nuevo valor de Kd en unidades de trabajo óptimos para el sistema inicializado con un entero y dos decimales
Si se digita la tecla Clear "*"
Si se digita la tecla Enter "#"
N
B

IMAGEN 1.3

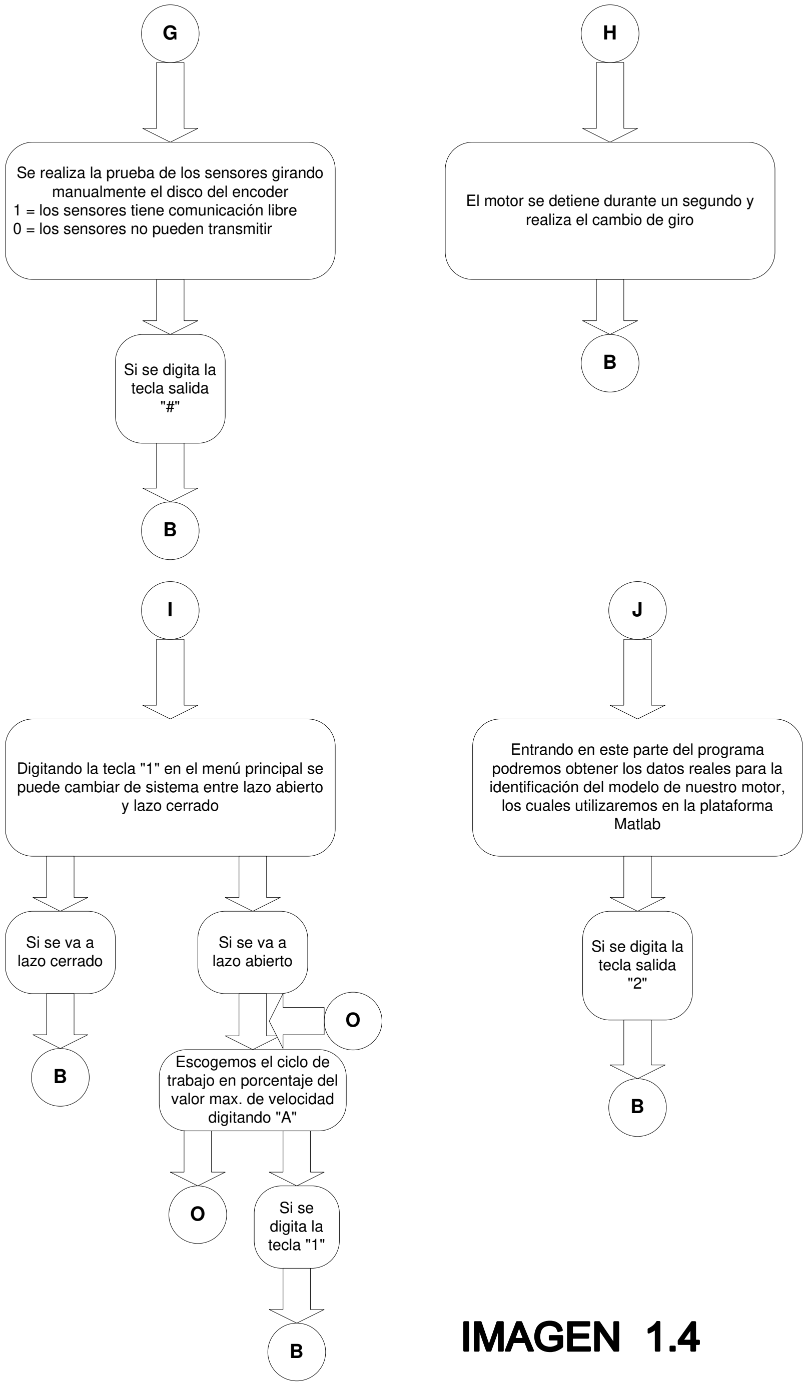

G
Se realiza la prueba de los sensores girando manualmente el disco del encoder
1 = los sensores tiene comunicación libre
0 = los sensores no pueden transmitir
Si se digita la tecla salida "#"
B
H
El motor se detiene durante un segundo y realiza el cambio de giro
B
I
Digitando la tecla "1" en el menú principal se puede cambiar de sistema entre lazo abierto y lazo cerrado
Si se va a lazo cerrado
B
Si se va a lazo abierto
O
Escogemos el ciclo de trabajo en porcentaje del valor max. de velocidad digitando "A"
O
Si se digita la tecla "1"
B
J
Entrando en este parte del programa podremos obtener los datos reales para la identificación del modelo de nuestro motor, los cuales utilizaremos en la plataforma Matlab
Si se digita la tecla salida "2"
B

IMAGEN 1.4

CAPITULO 2

DISEÑO E IMPLEMENTACIÓN

2.1 HARDWARE UTILIZADO

Nuestro proyecto consta de 7 secciones principales.

- Fuente.
- Controlador.
- Actuador.
- Planta.
- Encoder.
- Comunicación serial.
- Interfaz de usuario en el circuito.

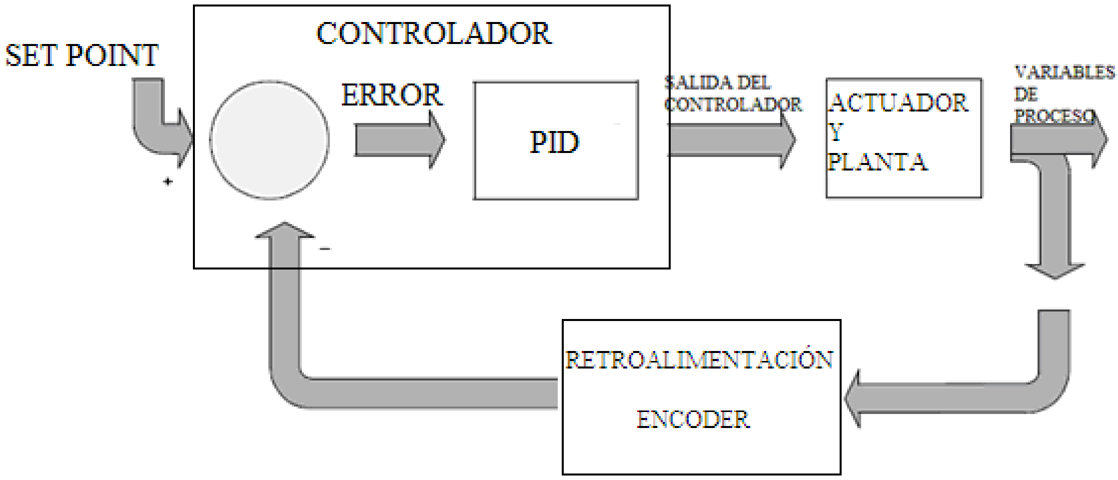


Diagrama del proceso de control

IMAGEN 2.1

Dentro de la fuente utilizamos un regulador de tres terminales positivos (7805) que acopla el voltaje de entrada del adaptador de 9V, reduciéndolo a 5V. Para mantener un suavizado en esta regularización, se adiciona unos filtros mejorando así el voltaje que alimenta del resto del circuito.

El controlador de nuestro sistema se lo realiza en el esquema de control del error y la función PID propiamente dicha; todo dentro de un dsPic que en nuestro caso particular utilizaremos el dsPic30F4011.

El actuador está conformado por un C.I.L293b encargado de darle al motor (PLANTA) la potencia necesaria tanto para el arranque como para el cambio de giro.

Luego contamos con un C.I. LM324 que recoge la señal proveniente del encoder fijado en el eje del motor por medio del juego de sensores infrarrojos, filtrando en un seguidor de voltaje y luego manteniendo estable la señal de salida pura a través del C.I 74ls14 para ser graficada.

La comunicación serial la realizamos por medio de un Max232, el cual con un circuito muy sencillo (grafica en el ANEXO D), comunica nuestro sistema con la PC.

Y finalmente la interfaz utilizada en nuestro circuito esta conformada por una pantalla LCD y un teclado.

2.1.1 ESTRUCTURA DE LOS DISPOSITIVOS

C.I. dsPIC30F4011

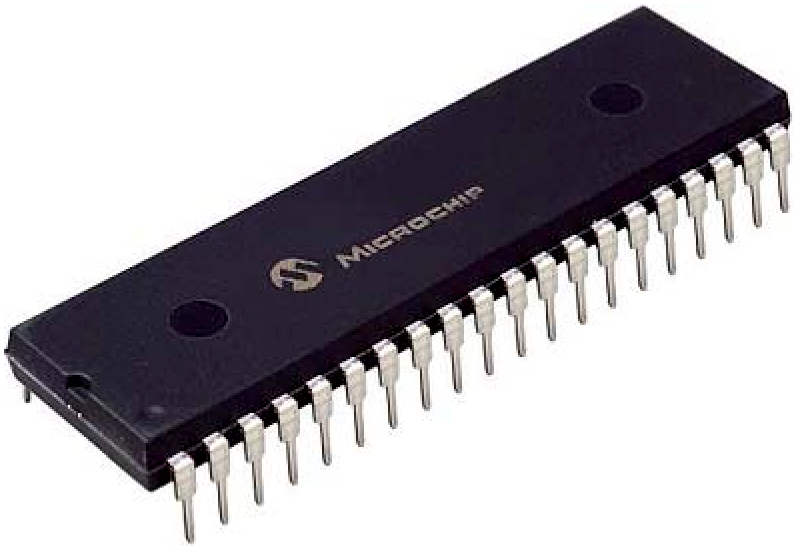


IMAGEN 2.2

Características de memória:

• 84 Bases de instrucciones.

• 48 Kbytes de espacio para la programación en memoria flash (16K de palabras de instrucción).

• 2 Kbytes de espacio en el chip para memoria RAM.

• 1 Kbytes de memoria EEPROM.

• Reloj externo de hasta 40 MHz.

• Oscilador de 4 MHz a 10 MHz con PLL activo (4x, 8x, 16x).

• 30 fuentes de interrupción interna.

• 3 fuentes de interrupción externa.

• 8 user selectable priority levels for each interrupt source

• 16 x 16-bit arreglos de registro de trabajo.

Características del módulo de control de motor por PWM

• 6 canales de salida de PWM.

• Salida de modo complementario o independiente.

• 3 generadores de duty cycle.

• Polaridad de salida programable.

• Control de tiempo muerto para modo complementario.

• Salida de control manual.

Características del módulo de interfaz de encoder:

• Entradas de fase A, fase B, y pulso índice.

• 16-bit para posición del contador inc./dec.

• Estado de la dirección del contador.

• Filtros de ruido digital programables en las entradas.

C.I. 74LS14

Este dispositivo contiene seis puertas, cada una de los cuales realiza la función lógica de inversión. Cada entrada tiene histéresis lo que aumenta la inmunidad al ruido y transforma una señal de cambio lento a una rápida.

ENTRADA	SALIDA
A	Y
L	H
H	L

H: ALTO VOLTAJE

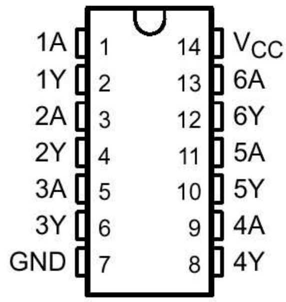
L: BAJO VOLTAJE

RANGOS MÁXIMOS

Voltaje alimentación: 7V

Voltaje de entrada: 7V

Máxima temperatura: 150o C



Ordenamiento de pines del 74LS14

IMAGEN 2.3

C.I. 7805

La serie de reguladores de tres terminales positivos, están disponibles en el paquete TO-220/D-PAK con varias tensiones de salida fija, lo que los hace útiles en una amplia gama de aplicaciones. Cada tipo emplea en su interior un limitador de corriente y protección térmica, por lo que es esencialmente indestructible. Este dispositivo puede entregar más de 1A de corriente de salida, estos dispositivos se puede utilizar con dispositivos externos que pueden ser ajustables para obtener voltajes y corrientes.

Características

• Corriente de salida de hasta 1A.

• Los voltajes de salida de 5, 6, 8, 9, 10, 12, 15, 18, 24V.

• Protección contra sobrecargas térmicas.

• Protección contra cortocircuitos.

• Caja de seguridad de funcionamiento del transistor de salida Zona de Protección.

• Máxima entrada de voltaje para voltaje de salida de 5V a 18V: 35V

• Máxima entrada de voltaje para voltaje de salida de 24V: 40V

IN COM OUT

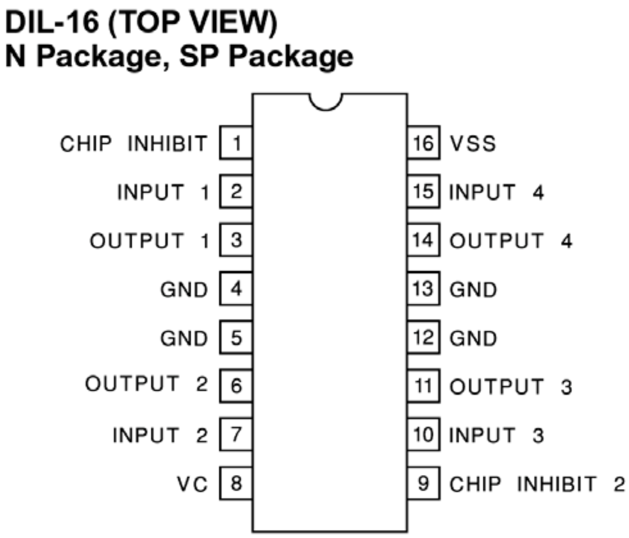
Ordenamiento de pines del 7805

IMAGEN 2.4

C.I. L293B

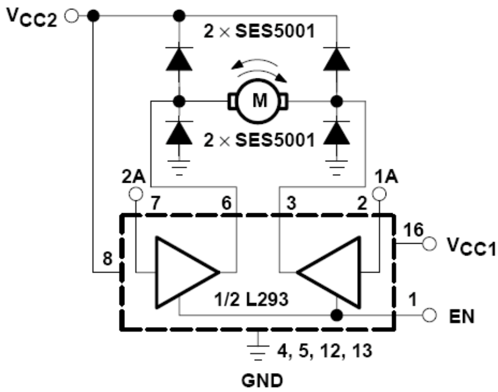
El dispositivo es un integrado monolítico de alta tensión, cuatro canales de alta corriente, driver diseñado para aceptar estándar DTL o aceptar niveles lógicos TTL y la unidad cargas inductivas (por ejemplo, relés solenoides, DC y de motores paso a paso) y transistores de potencia de conmutación. Para simplificar su uso cada par de canales está equipado con una entrada enable, lo que permite que el funcionamiento se de a un menor voltaje e incluye diodos de sujeción interior. Este dispositivo es adecuado para su uso en aplicaciones de conmutación a frecuencias de hasta 5 kHz.

El L293D se monta en un paquete de 16 pines, que dispone de 4 pines conectados y utilizados para heatsinking.



Ordenamiento de pines del L293

IMAGEN 2.6



EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Conexión del motor en el L293

IMAGEN 2.7

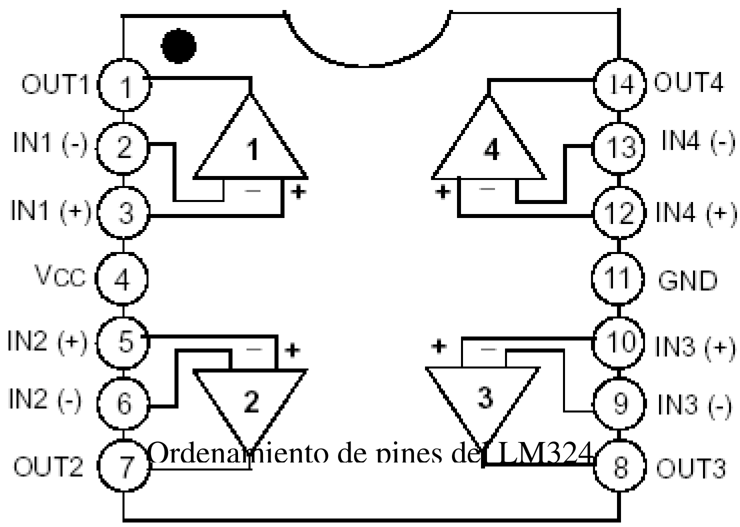
C.I. LM324

El LM324 consta de cuatro operacionales independientes y de alta ganancia, tienen su frecuencia internamente compensada, fueron diseñados específicamente para operar a partir de un único suministro de energía en una amplia gama de voltajes.

La operación de división de alimentación también es posible y el suministro de corriente de baja potencia es independiente de la magnitud de la tensión de alimentación.

Las áreas de aplicación incluyen amplificadores transductores, bloques de ganancia DC y de todos los circuitos convencionales con amplificadores operacionales, que ahora pueden ser más fáciles de implementar con una única fuente de alimentación en el sistema.

Por ejemplo, el LM324 puede ser directamente operado fuera de la norma de +5 V tensión de alimentación que se utiliza en sistemas digitales fácilmente y proporcionar la interfaz electrónica, sin necesidad adicional de ± 15V en su alimentación.



Ordenamiento de pines del LM324

VENTAJAS

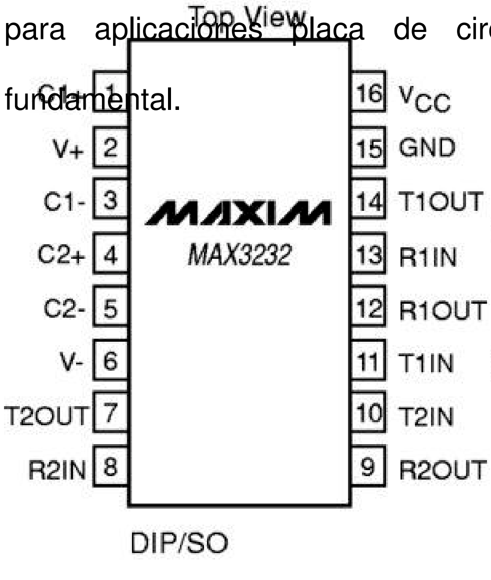
- Elimina la necesidad de doble suministro.
- Cuatro operacionales internos compensados en un solo paquete.
- Compatible con todas las formas de la lógica.
- Potencia de drenaje adecuado para la operación de baterías.

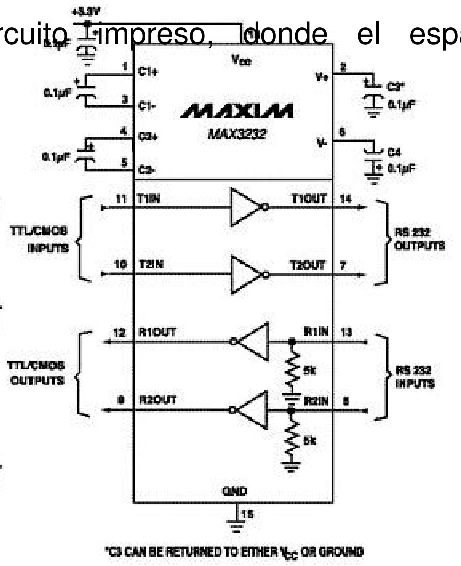
CARACTERÍSTICAS

- Frecuencia internamente compensada por unidad de ganancia
- Ganancia de voltaje de 100dB
- Gran ancho de banda (ganancia unitaria) 1 MHz (compensación de la temperatura)
- Amplia gama de alimentación: Única de 3V a 32V o de doble ± 1.5V a ± 16V
- Muy bajo suministro de corriente (700 µA)-esencialmente independiente de la tensión de alimentación
- Offset de entrada de baja tensión de 2 mV y compensación de corriente de entrada de 5 nA.
- Voltaje de modo común incluye tierra
- Rango de voltaje de entrada diferencial igual a la potencia tensión de alimentación
- Grande oscilación de voltaje de salida de 0 V a V + - 1,5 V.

C.I. MAX232

La familia MAX220-MAX249-MAX-232 son circuitos integrados capaces de establecer interfaces de comunicaciones V.28/V.24, en particular, las aplicaciones en las que la fuente es de ± 12V no está disponible. Estos integrados son especialmente útiles en los sistemas de baterías, ya que su baja potencia se reduce en modo de apagado disipación de energía a menos de 5µW. El MAX225, MAX233, MAX235, y MAX245/MAX246/MAX247 no usa componentes externos y se recomiendan para aplicaciones placa de circuito impreso, donde el espacio es fundamental.





APLICACIONES

- Ordenadores portátiles.
- Los módems de bajo consumo.
- Interfaz de traducción.
- Sistemas de Batería-Powered RS-232.
- Redes Multidrop RS-232.

FUNCIONAMIENTO

- Fuente única de +5v.
- Múltiples controladores y receptores.
- Línea Abierta de detección (MAX243).

PICkit

El PICkit 2 es un depurador de bajo costo para el desarrollo del programador.

IMAGEN 2.9 Configuración internadel MAX232

Legend:
1 – Status LEDs 3 – Lanyard Connection 5 – Pin 1 Marker
2 – Push Button 4 – USB Port Connection 6 – Programming Connector

IMAGEN 2.10

- **USB PORT CONNECTION**

  El puerto USB es una conexión USB conector mini-B. Conecte el PICkit 2 a la PC usando
  el cable USB suministrado.

- **STATUS LEDS**

  Los LEDs de estado indican el estado de la PICkit 2.
  1. Potencia (verde) La potencia que se aplica al PICkit 2 a través del puerto USB.
  2. Blanco (amarillo) PICkit 2 alimenta el dispositivo de destino.

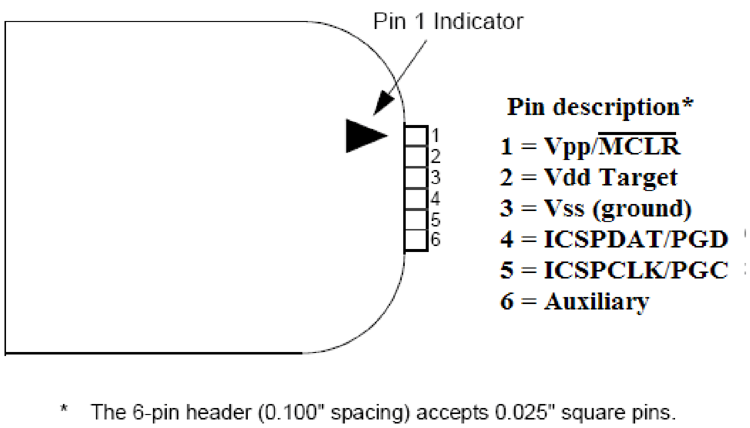
3. Ocupado (rojo) a-La PICkit 2 está ocupado con una función en curso, tales como programación.

- PUSH BUTTON

  El botón se puede utilizar para iniciar la función de programación de dispositivos cuando

- PROGRAMMING CONNECTOR

  El conector de programación de 6 pines (0,100 "espaciado) que se conecta a la PC, se muestra a continuación.



Pines de conexión del PICkit

IMAGEN 2.11

CABLE

La comunicación también esta dada por medio de un cable serial construido localmente en base al protocolo RS-232, dos pines para la transmisión y recepción respectivamente y un pin para la polarización con tierra, el resto de pines no necesariamente deben ser puenteados a menos que se encuentre en un área industrial, la conexión se la detalla en el anexo D

Cable serial

IMAGEN 2.12

2.2 ESTRATEGIAS UTILIZADAS

El banco de sensores montados en la estructura del encoder, y el disco con los obstáculos para la emisión-recepción entre ellos; es el sistema de medición de la velocidad del motor. Particularmente a este sistema se lo modificó con respecto a una versión anterior, en la parte de emisión al dejar tan solo una salida limitada y mínima para el paso de la señal, así se ha podido disminuir el tamaño de los obstáculos y tomar una medición con mayor precisión.

Este valor de dato tomado en el encoder, pasa a través de un seguidor de voltaje formado en el C.I. LM324, en el cual podemos trasladar la misma sin alteraciones, directamente a través de un fijador de valor o filtro como en este caso lo es el C.I. 74LS14, hacia el controlador para su respectiva graficación a en los métodos de visualización empleados.

2.3 ESQUEMATICO

2.3.1 ESQUEMATICO DE DESARROLLO

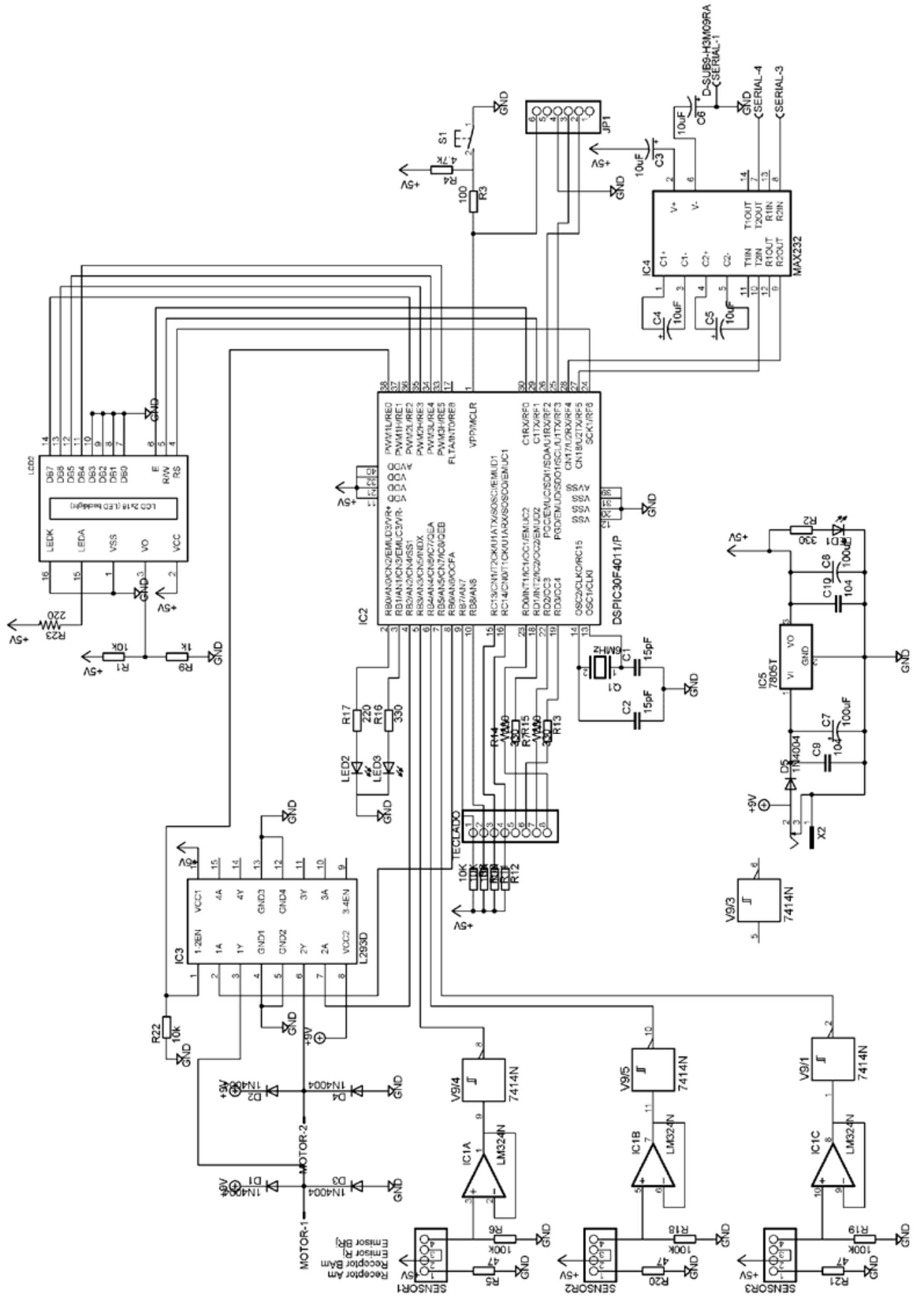


IMAGEN 2.13

2.3.2 ESQUEMATICO DE DISTRIBUCION DE COMPONENTES

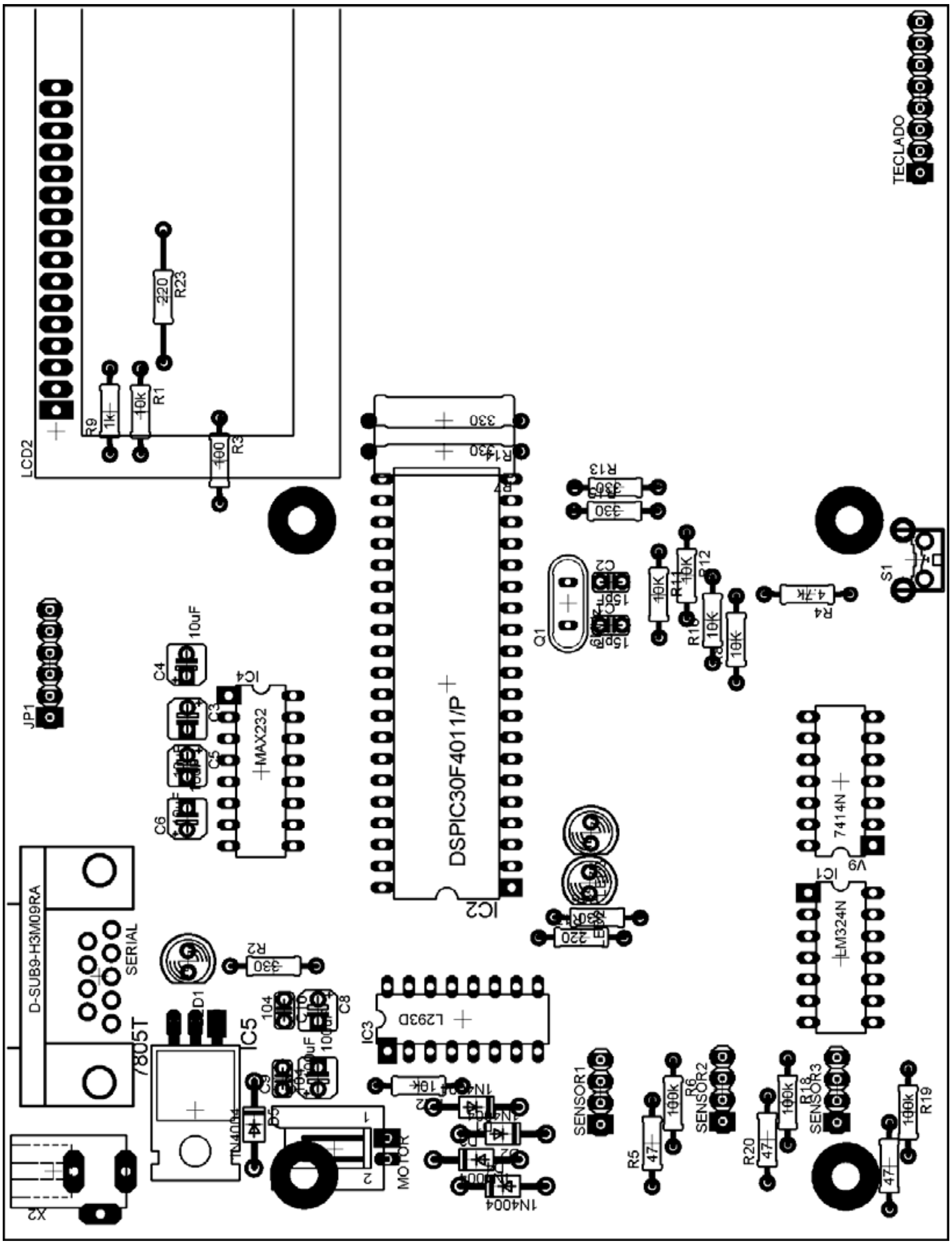


IMAGEN 2.14

2.4 DETALLES DE CONSTRUCCIÓN

Tanto la base como la cubierta fueron realizadas en acrílico para tener así la visualización necesario del circuito, una presentación didáctica como fue nuestro objetivo y el peso adecuado de fácil transportación.

La placa del controlador que contiene dicho circuito y la parte de acceso y visualización de datos, fue realizada en baquelita bajo el diseño realizado para esta función.

La estructura de montaje del motor a sido elaborada en madera, por ser un material de fácil manejo y resistencia necesaria. Además se necesitaba trabajar con precisión en el montaje del encoder y localización específica de los sensores.

2.5 ANALISIS DE COSTOS

2.5.1 RECURSOS POR HARDWARE Y SOFTWARE

Los costos de construcción del sistema de control PID para un motor DC son los siguientes:

Estructura de acrílico	$	20
Componentes de control	$	21.9
Estructura de soporte para el motor	$	10
Encoder óptico	$	1.2
Fuente	$	7
Tarjeta de control PID	$	30
Motor DC	$	2.5
Periféricos (teclado, LCD)	$	22
Elementos varios	$	17.71
Total	$	132.31

En el **anexo E** se adjunta una descripción detallada de cada rubro, en cuanto al costo software fue una herramienta proporcionada parte en el laboratorio de Microcontroladores y parte por personas que contribuyeron de esta manera a el desarrollo del proyecto.

2.5.2 RECURSO HUMANO

Diseño de estructura en acrílico	4 horas
Diseño de estructura del motor	24 horas
Diseño de tarjeta del controlador	60 horas
Diseño de encoder	3 horas
Instalación de elementos	4 horas
Programación en MikroBasic	72 horas
Programación en Visual Basic	12 horas
Programación en Labview	12 horas
Total	191 horas

CAPITULO 3

PROGRAMACIÓN Y APLICACIONES

3.1 UNIDAD DE PROGRAMACIÓN

Durante en el transcurso de las materias "**MICROPROCESADORES AVANZADOS**", dictada por el Ing. Carlos Valdivieso se aprendió el manejo de MikroBasick como lenguaje de programación para Pics y dsPICs.; siendo muy útil pues nos permite interactuar con los registros de control del microcontrolador, de modo que a pesar de ser un lenguaje de alto nivel, pueden notarse la variación de dichos registros además de contener opciones para configurar de manera sencilla tareas complejas como el diseño de un driver USB.

Adicionalmente se manejo lenguajes de visualización como Visual Basic y la plataforma LabVIEW.

Visual Basic es un lenguaje de fácil comprensión y uso, ya que su programación se realiza mediante la unión de objetos, los cuales traen su propio código de programación, inicialización y propiedades en conjunto agrupadas al utilizarlo.

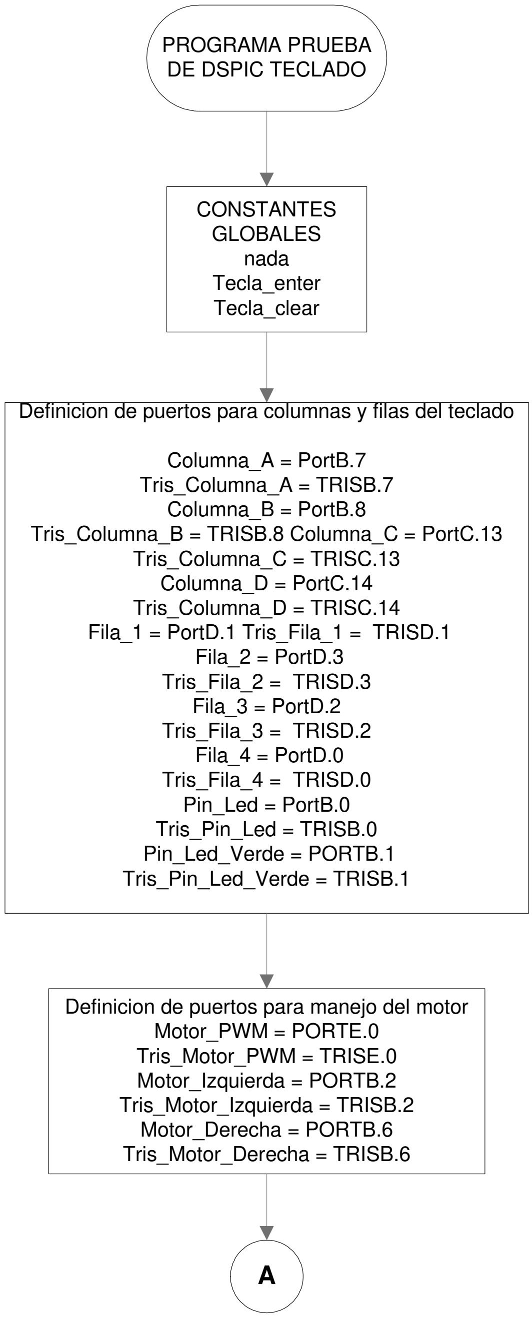
LabVIEW es un lenguaje de programación gráfico creado y desarrollado a lo largo de 20 años que permite a ingenieros y científicos realizar interfaces de medición y control de equipo rápidamente sin necesidad de tener amplios conocimientos de programación.

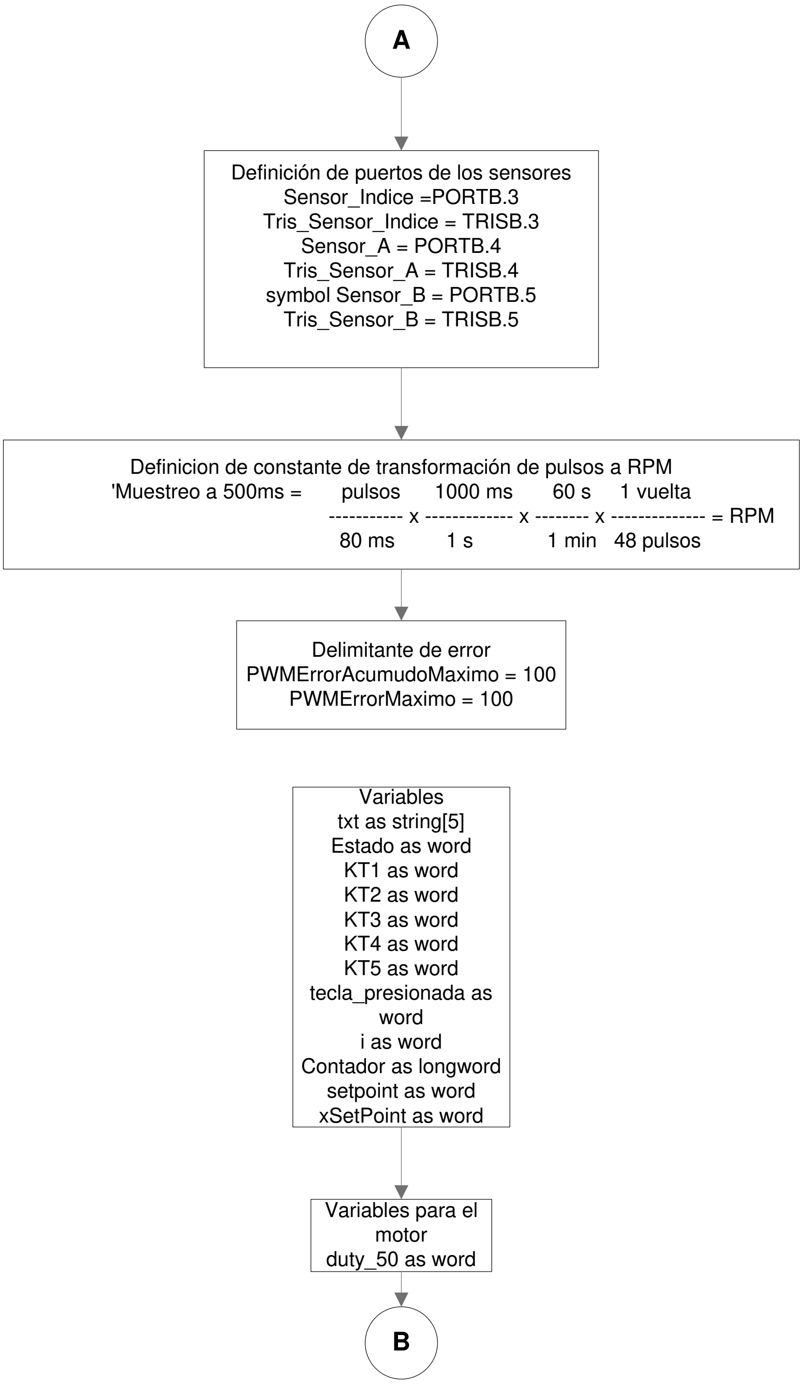
La versión 8.5 que se utilizó tiene algunas innovaciones que mejoran su interfaz y utiliza los nuevos procesadores con múltiples núcleos

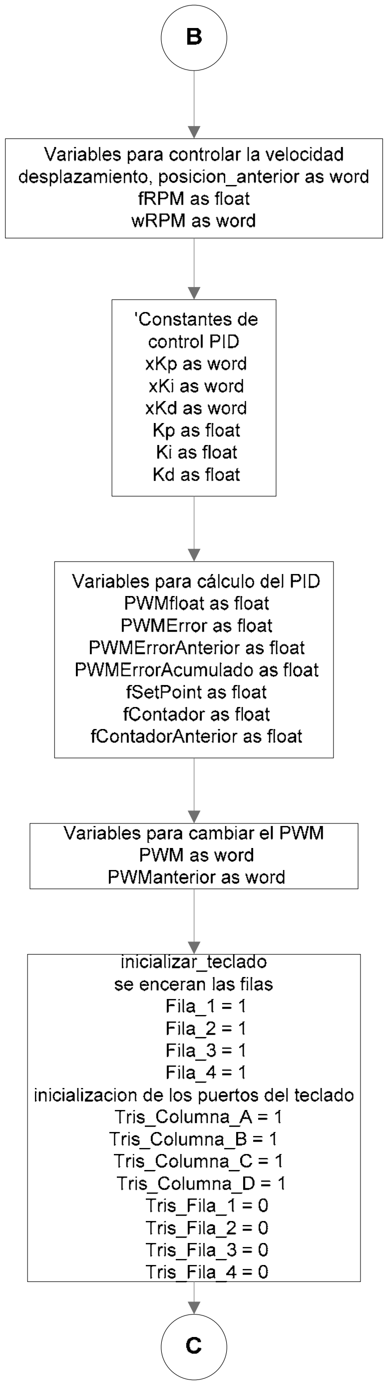
Entre sus características técnicas se puede apreciar las siguientes:

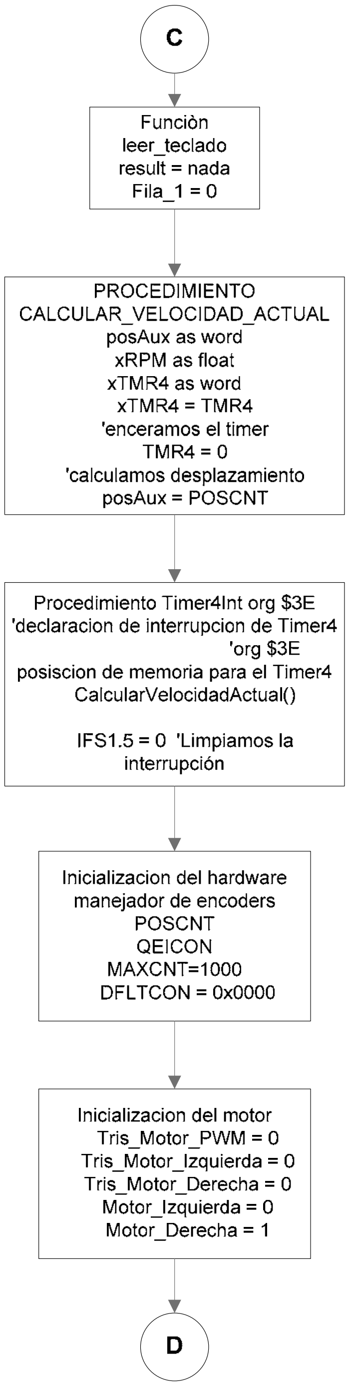
- Programación gráfica.
- Herramientas de desarrollo de alto nivel.
- Incluye funciones de medición y análisis.
- Existen versiones para los sistemas operativos Windows, MacOS y Linux.
- Soporta dispositivos embebidos.
- Conexión de Internet.
- Gran cantidad de accesorios y herramientas de terceros.

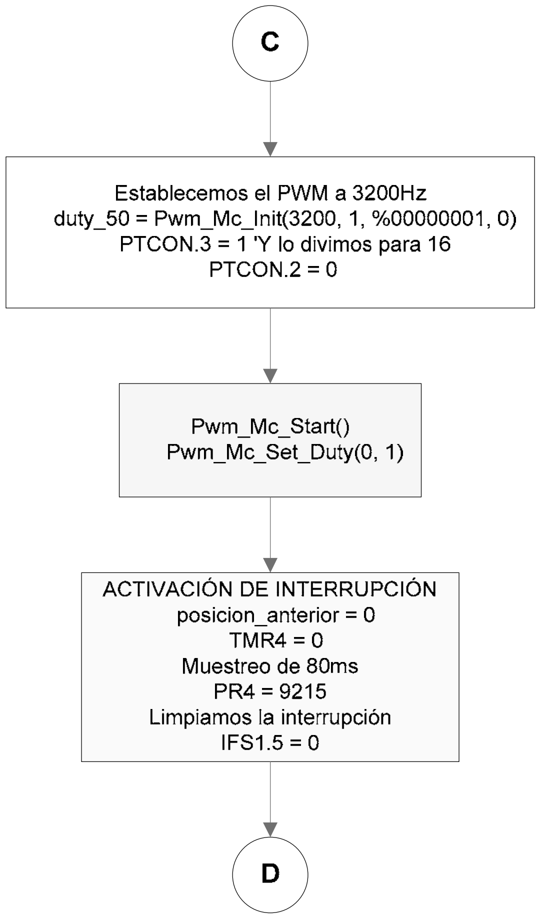
3.2 DIAGRAMAS DE FLUJO

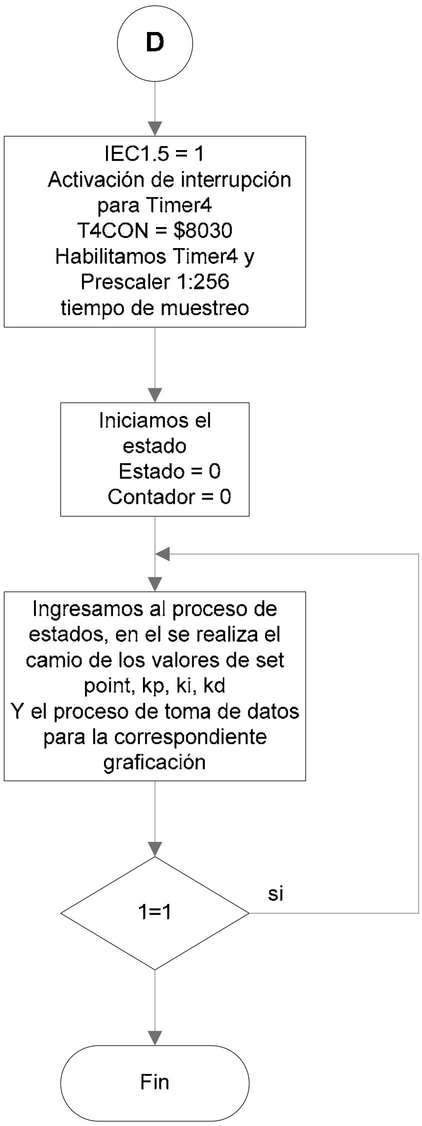



A
Definición de puertos de los sensores
Sensor_Indice =PORTB.3
Tris_Sensor_Indice = TRISB.3
Sensor_A = PORTB.4
Tris_Sensor_A = TRISB.4
symbol Sensor_B = PORTB.5
Tris_Sensor_B = TRISB.5
Definicion de constante de transformación de pulsos a RPM
'Muestreo a 500ms = pulsos 1000 ms 60 s 1 vuelta
----------- x ------------- x -------- x -------------- = RPM
80 ms 1 s 1 min 48 pulsos
Delimitante de error
PWMErrorAcumudoMaximo = 100
PWMErrorMaximo = 100
Variables
txt as string[5]
Estado as word
KT1 as word
KT2 as word
KT3 as word
KT4 as word
KT5 as word
tecla_presionada as word
i as word
Contador as longword
setpoint as word
xSetPoint as word
Variables para el motor
duty_50 as word
B


B
Variables para controlar la velocidad
desplazamiento, posicion_anterior as word
fRPM as float
wRPM as word
'Constantes de
control PID
xKp as word
xKi as word
xKd as word
Kp as float
Ki as float
Kd as float
Variables para cálculo del PID
PWMfloat as float
PWMError as float
PWMErrorAnterior as float
PWMErrorAcumulado as float
fSetPoint as float
fContador as float
fContadorAnterior as float
Variables para cambiar el PWM
PWM as word
PWManterior as word
inicializar_teclado
se enceran las filas
Fila_1 = 1
Fila_2 = 1
Fila_3 = 1
Fila_4 = 1
inicializacion de los puertos del teclado
Tris_Columna_A = 1
Tris_Columna_B = 1
Tris_Columna_C = 1
Tris_Columna_D = 1
Tris_Fila_1 = 0
Tris_Fila_2 = 0
Tris_Fila_3 = 0
Tris_Fila_4 = 0
C


C
Funciòn
leer_teclado
result = nada
Fila_1 = 0
PROCEDIMIENTO
CALCULAR_VELOCIDAD_ACTUAL
posAux as word
xRPM as float
xTMR4 as word
xTMR4 = TMR4
'enceramos el timer
TMR4 = 0
'calculamos desplazamiento
posAux = POSCNT
Procedimiento Timer4Int org $3E
'declaracion de interrupcion de Timer4
'org $3E
posiscion de memoria para el Timer4
CalcularVelocidadActual()
IFS1.5 = 0 'Limpiamos la
interrupción
Inicializacion del hardware
manejador de encoders
POSCNT
QEICON
MAXCNT=1000
DFLTCON = 0x0000
Inicializacion del motor
Tris_Motor_PWM = 0
Tris_Motor_Izquierda = 0
Tris_Motor_Derecha = 0
Motor_Izquierda = 0
Motor_Derecha = 1
D


C
Establecemos el PWM a 3200Hz
duty_50 = Pwm_Mc_Init(3200, 1, %00000001, 0)
PTCON.3 = 1 'Y lo divimos para 16
PTCON.2 = 0
Pwm_Mc_Start()
Pwm_Mc_Set_Duty(0, 1)
ACTIVACIÓN DE INTERRUPCIÓN
posicion_anterior = 0
TMR4 = 0
Muestreo de 80ms
PR4 = 9215
Limpiamos la interrupción
IFS1.5 = 0
D


D
IEC1.5 = 1
Activación de interrupción
para Timer4
T4CON = $8030
Habilitamos Timer4 y
Prescaler 1:256
tiempo de muestreo
Iniciamos el
estado
Estado = 0
Contador = 0
Ingresamos al proceso de
estados, en el se realiza el
camio de los valores de set
point, kp, ki, kd
Y el proceso de toma de datos
para la correspondiente
graficación
1=1
si
Fin

CAPITULO 4

DATOS EXPERIMENTALES

4.1 PRUEBAS REALIZADAS

A continuación se muestra segmentos de las pruebas realizadas, las mismas que pueden ser apreciadas gráficamente en el Anexo F.

Cabe mencionar que los datos son tomados cada 80 mili-segundos.

24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	0	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0
24/02/2009 17:42	3031	0

24/02/2009 17:42	3031	719
24/02/2009 17:42	3031	750
24/02/2009 17:42	3031	812
24/02/2009 17:42	3031	875
24/02/2009 17:42	3031	938
24/02/2009 17:42	3031	969
24/02/2009 17:42	3031	1031
24/02/2009 17:42	3031	1062
24/02/2009 17:42	3031	1125
24/02/2009 17:42	3031	1203
24/02/2009 17:42	3031	1234
24/02/2009 17:42	3031	1312
24/02/2009 17:42	3031	1328
24/02/2009 17:42	3031	1391
24/02/2009 17:42	3031	1422
24/02/2009 17:42	3031	1469
24/02/2009 17:42	3031	1516
24/02/2009 17:42	3031	1578
24/02/2009 17:42	3031	1609
24/02/2009 17:42	3031	1641
24/02/2009 17:42	3031	1688
24/02/2009 17:42	3031	1719
24/02/2009 17:42	3031	1750
24/02/2009 17:42	3031	1766
24/02/2009 17:42	3031	1812
24/02/2009 17:42	3031	1844
24/02/2009 17:42	3031	1859
24/02/2009 17:42	3031	1891
24/02/2009 17:42	3031	1922
24/02/2009 17:42	3031	1953
24/02/2009 17:42	3031	2000
24/02/2009 17:42	3031	1984
24/02/2009 17:42	3031	2016
24/02/2009 17:42	3031	2062
24/02/2009 17:42	3031	2062
24/02/2009 17:42	3031	2094
24/02/2009 17:42	3031	2125
24/02/2009 17:42	3031	2125
24/02/2009 17:42	3031	2156
24/02/2009 17:42	3031	2156

24/02/2009 17:42	3031	2531
24/02/2009 17:42	3031	2422
24/02/2009 17:42	3031	2484
24/02/2009 17:42	3031	2438
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2438
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2438
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2438
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2453
24/02/2009 17:42	3031	2469
24/02/2009 17:43	3031	2453
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2438
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2438
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2453
24/02/2009 17:43	3031	2453
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2469
24/02/2009 17:43	3031	2453
24/02/2009 17:43	3031	2500
24/02/2009 17:43	3031	2609
24/02/2009 17:43	3031	2688

24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3000
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3000
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3047
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3016
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3047
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3062
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3031
24/02/2009 17:43	3031	3062

4.2 CURVAS DE COMPORTAMIENTO

COMPORTAMIENTO PROPORCIONAL

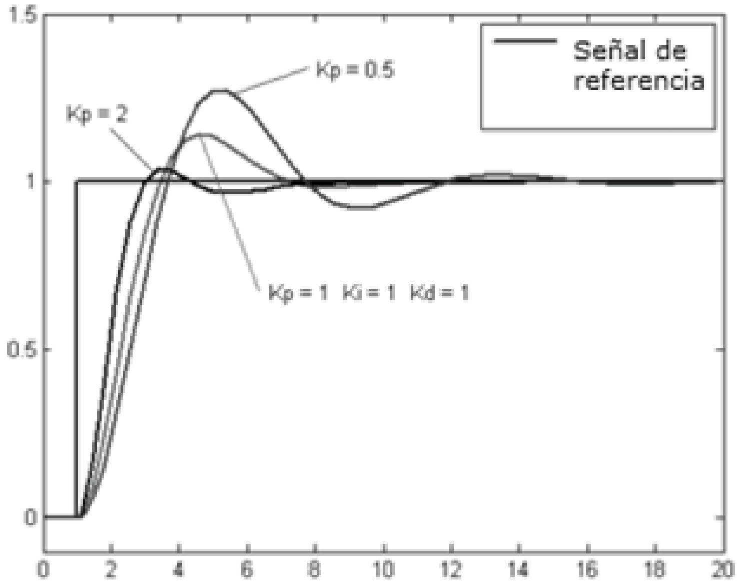


IMAGEN 4.1

COMPORTAMIENTO INTEGRAL

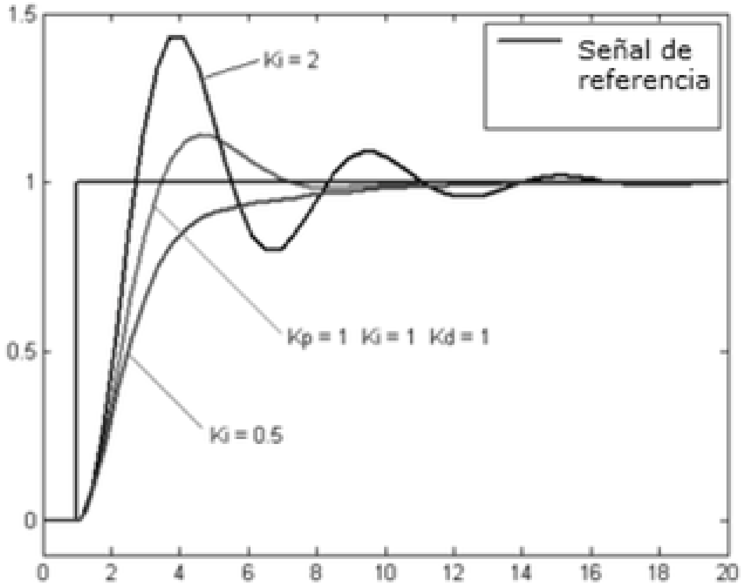


IMAGEN 4.2

COMPORTAMIENTO DERIVATIVO

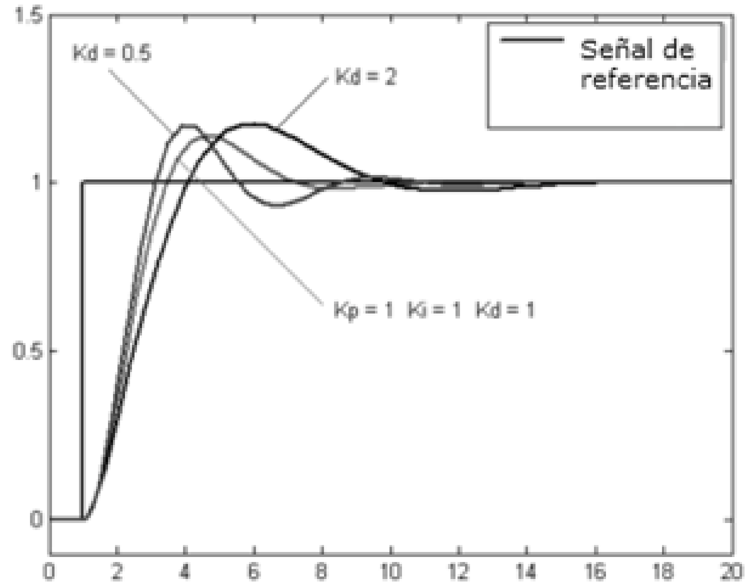


IMAGEN 4.3

4.3 OBTENCIÓN DE RESULTADOS Y ERRORES

Como se puede observar en la variación de valores al cambiar el SetPoint y en las gráficas obtenidas que se presentan en el ANEXO F, el sistemas se presenta sub-amortiguado con los valores seteados de la constantes de control (Kp, Ki y Kd); pero también se puede variar este comportamiento sin inconvenientes.

Se realiza la observación acerca del valor de velocidad cuando cruza los 2400 RPM, puesto que la curva tiende a saturarse en este valor, y esto se debe a las características internas del motor, ya que podríamos estar frente a un armónico.

Las curvas obtenidas no son líneas rectas puesto que se debe precisamente a las características del motor internamente y a los cálculos matemáticos que se realizan en las formulas, pues debido al cambio de variables que sufren los datos, se pierden o se alteran los valores decimales. Pero esto no afecta al fin deseado puesto que se puede mantener el control requerido.

CONCLUSIONES

El prototipo realizado cumple el objetivo principal de desarrollar un equipo de control PID de un motor DC con fines didácticos; puesto que es de fácil acceso a sus componentes y se pueden realizar diferentes mediciones intercambiando los discos del encoder que contiene diversas formas para los obstáculos de visibilidad entre los sensores

El control implementado brinda características especiales de funcionamiento como instrumento de laboratorio permitiendo que equipos de bajo costo integren una simulación de un proceso muy utilizado en la industria.

El microcontrolador dsPIC utilizado en la construcción de la tarjeta controladora junto con el lenguaje de programación MikroBasic, facilitaron la implementación de los lazos de control necesarios para este sistema, demostrando las prestaciones de esta familia de microcontroladores con comandos sencillos y prácticos.

La utilización de Visual Basic como lenguaje gráfico fue acertada puesto que a mas de ser práctico en su programación, es una herramienta didáctica y nos ayuda a observar el comportamiento de los circuitos a través de la conexión serial.

La selección de LabVIEW como entorno gráfico de programación permite a los futuros profesionales aprender una excelente herramienta que recorta el tiempo en la implementación de sistemas de control.

RECOMENDACIONES

Se recomienda el uso de LabVIEW como una potente herramienta de reconocimiento de imágenes que no requiere profundizar en el conocimiento de algoritmos de elevada complejidad. Se recuerda además que la universidad cuenta con estas herramientas y los módulos para desarrollar aplicaciones comerciales permitiendo a la ESPOL impulsar proyectos de control industrial en nuestro país.

Se aconseja el uso de los dsPICs por su buen desempeño como microcontrolador de altas prestaciones y bajo precio, tanto en el costo del dispositivo como en las herramientas de de programación.

Se sugiere utilizar el lenguaje MikroBasic como herramienta de desarrollo en la programación de microcontroladores Microchip de todos los tipos.

Finalmente, se exhorta a la asignación de recursos económicos para este tipo de proyectos de graduación con el fin de mejorar la calidad de prototipos a desarrollar.

ANEXO A

CODIGO FUENTE DEL PROGRAMA DEL

CONTROLADOR (dsPic) REALIZADO EN MIKROBASIC

```
program prueba_dsPic_teclado

symbol nada = $FFFF
symbol tecla_enter = "#"
symbol tecla_clear = "*"

'definicion de puertos para columnas y filas del teclado
symbol Columna_A = PortB.7
symbol Tris_Columna_A = TRISB.7
symbol Columna_B = PortB.8
symbol Tris_Columna_B = TRISB.8
symbol Columna_C = PortC.13
symbol Tris_Columna_C = TRISC.13
symbol Columna_D = PortC.14
symbol Tris_Columna_D = TRISC.14
symbol Fila_1 = PortD.1
symbol Tris_Fila_1 =  TRISD.1
symbol Fila_2 = PortD.3
symbol Tris_Fila_2 =  TRISD.3
symbol Fila_3 = PortD.2
symbol Tris_Fila_3 =  TRISD.2
symbol Fila_4 = PortD.0
symbol Tris_Fila_4 =  TRISD.0

symbol Pin_Led = PortB.0
symbol Tris_Pin_Led = TRISB.0
symbol Pin_Led_Verde = PORTB.1
symbol Tris_Pin_Led_Verde = TRISB.1
```

```
'Definicion de puertos para manejo del motor
symbol Motor_PWM = PORTE.0
symbol Tris_Motor_PWM = TRISE.0
symbol Motor_Izquierda = PORTB.2
symbol Tris_Motor_Izquierda = TRISB.2
symbol Motor_Derecha = PORTB.6
symbol Tris_Motor_Derecha = TRISB.6
'Definición de puertos de los sensores
symbol Sensor_Indice = PORTB.3
symbol Tris_Sensor_Indice = TRISB.3
symbol Sensor_A = PORTB.4
symbol Tris_Sensor_A = TRISB.4
symbol Sensor_B = PORTB.5
symbol Tris_Sensor_B = TRISB.5

'Definicion de constante de transformación de pulsos a RPM
'Muestreo a 500ms = pulsos   1000 ms     60 s     1 vuelta
'                   ------ x ------- x ------- x ----------- = RPM
'                   100 ms      1 s      1 min     48 pulsos
'symbol Desplazamiento_a_RPM = 12.5
'Muestreo a 500ms = pulsos   1000 ms     60 s     1 vuelta
'                   ------ x ------- x ------- x ----------- = RPM
'                    80 ms      1 s      1 min     48 pulsos
symbol Desplazamiento_a_RPM = 15.625
'Muestreo a = pulsos   2880 TMR4   1000 ms     60 s     1 vuelta
'             ------ x ---------- x ------- x ------- x ----------- = RPM
'              TMR4      25 ms       1 s      1 min     48 pulsos

symbol PWMErrorAcumuladoMaximo = 100      'valor con el que sube la curva paso a paso
symbol PWMErrorMaximo = 100

'Periodo de la curva de identificación = 5 x 80ms
symbol IdentPeriodo = 10
'Ciclo de trabajo mínimo para la identificación
symbol IdentMinimo = 3000 '20 * 14998 / 100
'Ciclo de trabajo máximo para la identificación
symbol IdentMaximo = 6000 '40 * 14998 / 100
```

```
dim txt as string[5]
dim Estado as word
dim KT1 as word
dim KT2 as word
dim KT3 as word
dim KT4 as word
dim KT5 as word

dim tecla_presionada as word
dim i as word
dim Contador as longword
dim setpoint as word
dim xSetPoint as word

'Variables para el motor
dim duty_50 as word

'Variables para controlar la velocidad
dim desplazamiento, posicion_anterior as word
dim fRPM as float
dim wRPM as word

'Variable para indicar lazo abierto o cerrado
dim Lazo as char

'Variable para el modo de identificación de sistema
dim Identificacion as char
dim Ident_Temporizador as word
dim Ident_Cual as word

 'Constantes de control PID
 dim xKp as word
 dim xKi as word
 dim xKd as word
 dim Kp as float
 dim Ki as float
 dim Kd as float
 'Variables para cálculo del PID
 dim PWMfloat as float
 dim PWMError as float
```

```
 dim PWMErrorAnterior as float
 dim PWMErrorAcumulado as float
 dim fSetPoint as float
 dim fContador as float
 dim fContadorAnterior as float
 'Variables para cambiar el PWM
 dim PWM as word
 dim PWManterior as word

sub procedure inicializar_teclado()
    'se enceran las filas
     Fila_1 = 1
     Fila_2 = 1
     Fila_3 = 1
     Fila_4 = 1
'inicializacion de los puertos del teclado
     Tris_Columna_A = 1
     Tris_Columna_B = 1
     Tris_Columna_C = 1
     Tris_Columna_D = 1
     Tris_Fila_1 = 0
     Tris_Fila_2 = 0
     Tris_Fila_3 = 0
     Tris_Fila_4 = 0
end sub

sub function leer_teclado() as word
       result = nada

       Fila_1 = 0
       nop                    'no hace nada durante un ciclo de instruccion
                             'para que la informacion del puerto cambie
       if Columna_A = 0 then
          while Columna_A = 0 wend
          result = 1
       end if
       if Columna_B = 0 then
          while Columna_B = 0 wend
          result = 2
       end if
```

```
if Columna_C = 0 then
  while Columna_C = 0 wend
  result = 3
end if
if Columna_D = 0 then
  while Columna_D = 0 wend
  result = "A"
end if
Fila_1 = 1

Fila_2 = 0
nop
if Columna_A = 0 then
  while Columna_A = 0 wend
  result = 4
end if
if Columna_B = 0 then
  while Columna_B = 0 wend
  result = 5
end if
if Columna_C = 0 then
  while Columna_C = 0 wend
  result = 6
end if
if Columna_D = 0 then
  while Columna_D = 0 wend
  result = "B"
end if
Fila_2 = 1

Fila_3 = 0
nop
if Columna_A = 0 then
  while Columna_A = 0 wend
  result = 7
end if
if Columna_B = 0 then
  while Columna_B = 0 wend
  result = 8
end if
```

```
      if Columna_C = 0 then
        while Columna_C = 0 wend
        result = 9
      end if
      if Columna_D = 0 then
        while Columna_D = 0 wend
        result = "C"
      end if
      Fila_3 = 1

      Fila_4 = 0
      nop
      if Columna_A = 0 then
        while Columna_A = 0 wend
'         result = "*"
         result = tecla_clear
      end if
      if Columna_B = 0 then
        while Columna_B = 0 wend
        result = 0
      end if
      if Columna_C = 0 then
        while Columna_C = 0 wend
'         result = "#"
         result = tecla_enter
      end if
      if Columna_D = 0 then
        while Columna_D = 0 wend
        result = "D"
      end if
      Fila_4 = 1
end sub

sub procedure CalcularVelocidadActual()
dim posAux as word
dim xRPM as float
dim xTMR4 as word
   xTMR4 = TMR4
   'enceramos el timer
   TMR4 = 0
```

```
'calculamos desplazamiento
posAux = POSCNT
if Motor_Izquierda = 1 then
  desplazamiento = posicion_anterior - posAux
end if
if Motor_Derecha = 1 then
  desplazamiento = posAux - posicion_anterior
end if
posicion_anterior = posAux
'Calculamos el RPM basandose en el TMR4
fRPM = Desplazamiento_a_RPM * desplazamiento
'Restringimos la velocidad a menos de 255
if desplazamiento < 255 then
  'Enviamos el desplazamiento por el puerto serial
  Uart2_Write_Char(desplazamiento)
end if
'Si estamos en Lazo cerrado
if Lazo = "C" then
  'Controlamos la velocidad
  if fSetPoint > 0 then
    'Calculamos el error
    fContador = desplazamiento
    PWMError = fSetPoint - fContador

    if PWMError <> 0 then
      if PWMError > PWMErrorMaximo then
        PWMError = PWMErrorMaximo
      else
        if PWMError < -PWMErrorMaximo then
          PWMError = -PWMErrorMaximo
        end if
      end if
      'Calculamos el error acumulado
      PWMErrorAcumulado = PWMErrorAcumulado + PWMError
      if PWMErrorAcumulado > PWMErrorAcumuladoMaximo then
        PWMErrorAcumulado = PWMErrorAcumuladoMaximo
      else
        if PWMErrorAcumulado < -PWMErrorAcumuladoMaximo then
          PWMErrorAcumulado = -PWMErrorAcumuladoMaximo
        end if
```

```
          end if
          'Calculamos el nuevo PWM con un controlador PID
          PWMfloat = PWMfloat + ( Kp * PWMError ) + ( Ki * PWMErrorAcumulado ) + (
Kd * ( PWMError - PWMErrorAnterior ) )
          'Almacenamos el fContadorAnterior
          PWMErrorAnterior = PWMError
          'Límitamos el valor del PWM
          if PWMfloat > 14998 then       'valor del PWM al 100%
             PWMfloat = 14998
          else
             if PWMfloat < 0 then
                PWMfloat = 0
             end if
          end if
          PWM = PWMfloat
          if PWManterior <> PWM then
             if PWM > 2000 then         'porcentaje de la velocidad maxima,
                                         'y relaciona al voltaje minimo con el que trabaja el motor sin
problemas
                'Cambiamos al nuevo valor
                Pwm_Mc_Set_Duty(PWM, 1)   'establecemos el ciclo de trabajo
                'Establecemos el PWManterior
                PWManterior = PWM
             end if
          end if
        end if
      else
        Pwm_Mc_Set_Duty(0, 1)    'para apagar el motor
      end if
    end if
    'Si estamos en modo de identificación
    if Identificacion = "i" then
       inc(Ident_Temporizador)
       if Ident_Temporizador = IdentPeriodo then
          'Enviamos el aviso que vamos a enviar el setpoint
          Uart2_Write_Char(255)
          if Ident_Cual = 0 then
             Pwm_Mc_Set_Duty(IdentMaximo, 1)   'establecemos el ciclo de trabajo
             Ident_Cual = 1
             'Enviamos el nuevo setpoint a la pc
```

```
            Uart2_Write_Char(203)
          else
            Pwm_Mc_Set_Duty(IdentMinimo, 1)   'establecemos el ciclo de trabajo
            Ident_Cual = 0
            'Enviamos el nuevo setpoint a la pc
            Uart2_Write_Char(106)
          end if
          Ident_Temporizador = 0
        end if
      end if
end sub

sub procedure Timer4Int org $3E   'declaracion de interrupcion de Timer4
                                  'org $3E posiscion de memoria para el Timer4
    CalcularVelocidadActual()

    IFS1.5 = 0  'Limpiamos la interrupción
end sub

main:
     ADPCFG = $FFFF    'configurar puertos analogicos en digitales Puerto
'Inicializas el teclado
     inicializar_teclado
'Inicializacion del LCD
     Lcd_Init(PORTE, 2,3,4,5, PORTF, 6,1,0)
'Inicializas el LED
     Tris_Pin_Led = 0
     Tris_Pin_Led_Verde = 0
'inicializacion del puerto serial
     Uart2_Init(19200)
'Inicializacion del hardware manejador de encoders
     POSCNT = 0
     'QEICONbits.QEISIDL = 0    'Continuar en modo idle (0)
     QEICON.13 = 0
     'QEICON.QEIM = 0b100       'Resolución 4x, modo de Reset por MAXCNT
     QEICON.10 = 1
     QEICON.9 = 1
     QEICON.8 = 0
     'QEICONbits.SWPAB = 0      'Phase-A y Phase-B pines originales
     QEICON.7 = 0
```

```
    'QEICONbits.PCDOUT = 1       'Activado el pin UPDN para informar el sentido de
giro del encoder
    QEICON.6  = 1
    'QEICON.TQGATE = 0        'Timer gate apagado
    QEICON.5 = 0
    'QEICONbits.TQCKPS = 0     'Prescaler 1:1
    QEICON.4= 0
    QEICON.3= 0
    'QEICONbits.POSRES = 0     'Un pulso en INDEX no hace un reset en POSCNT
    QEICON.2 = 0
    'QEICONbits.TQCS =0       'Usamos clock interno para el timer  Tcy
    QEICON.1 =0
     'QEICONbits.UDSRC=1       'Phase-B indica dirección
    QEICON.0=1

    MAXCNT=1000
    DFLTCON = 0x0000
'Inicializacion del motor
    Tris_Motor_PWM = 0
    Tris_Motor_Izquierda = 0
    Tris_Motor_Derecha = 0
    Motor_Izquierda = 0
    Motor_Derecha = 1

    'Establecemos el PWM a 3200Hz
    duty_50 = Pwm_Mc_Init(3200, 1, %00000001, 0)
    PTCON.3 = 1 'Y lo divimos para 16
                PTCON.2 = 0

'     WordToStr(duty_50, txt)
'     Uart2_Write_Text(txt)
'     Uart2_Write_Text(" ")

    Pwm_Mc_Start()
    Pwm_Mc_Set_Duty(0, 1)

'ACTIVACIÓN DE INTERRUPCIÓN
  posicion_anterior = 0
  TMR4 = 0
'  PR4 = 57599       'Muestreo de 500ms
```

```
'  PR4 = 28799       'Muestreo de 250ms
'  PR4 = 23039       'Muestreo de 200ms
'  PR4 = 11519       'Muestreo de 100ms
   PR4 = 9215       'Muestreo de 80ms
'  PR4 = 2880        'Muestreo de 25ms
  IFS1.5 = 0     'Limpiamos la interrupción
  IEC1.5 = 1     'Activación de interrupción para Timer4
  T4CON = $8030  'Habilitamos Timer4 y Prescaler 1:256    'tiempo de muestreo
'  T4CON = $8000  'Habilitamos Timer4 y Prescaler 1:1
'  T4CON = $8010  'Habilitamos Timer4 y Prescaler 1:8
'  T4CON = $8020  'Habilitamos Timer4 y Prescaler 1:64

'Iniciamos el estado
     Estado = 0
     Contador = 0

     WHILE (1=1)
       select case Estado
             Case 0
                  'Empezamos en Lazo cerrado
                  Lazo = "C"
                  'Enceramos las variables
                  Identificacion = " "
                  Kp = 0.5
                  Ki = 0.02
                  Kd = 0.01
                  setpoint = 0
                  fSetPoint = setpoint
                 'Enviamos el aviso que vamos a enviar el setpoint
                 Uart2_Write_Char(255)
                 'Calculamos el setpoint para enviar
                 Uart2_Write_Char(0)

                  'Probamos el Pic con el parpadeo de un led
                    for i=1 to 5
                        Pin_Led = 0
                        delay_ms(50)
                        Pin_Led = 1
                        delay_ms(50)
                    next i
```

```
    'Y nos vamos al siguiente estado
    Estado = 10
Case 10
    Lcd_Cmd(LCD_CURSOR_OFF)
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "  E S P O L  ")
    Lcd_Out(2, 1, "MAT. GRADUACION")
    delay_ms(2000)
    Estado = 20
Case 20
    Lcd_Cmd(LCD_CURSOR_OFF)
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "JAVIER MEJIA")
    Lcd_Out(2, 1, "WASHINGTON REINA")
    delay_ms(2000)
    Estado = 30
Case 30
    Lcd_Cmd(LCD_CURSOR_OFF)
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "RPM =      1340")
    if Lazo = "C" then
      Lcd_Out(2, 1, "SetPoint =     ")
    else
      Lcd_Out(2, 1, "C.Trabajo=     ")
    end if
    Lcd_Chr(1, 16, Lazo)
    Lcd_Chr(2, 16, Identificacion)
    wRPM = fRPM
    WordToStr(wRPM, txt)
    Lcd_Out(1,11,txt)
'          WordToStr(desplazamiento, txt)
'          Lcd_Out(1,11,txt)
    WordToStr(setpoint, txt)
    Lcd_Out(2, 11, txt)
    Estado = 40
Case 40
   'se obtiene el valor segun tecla presionada
   tecla_presionada = leer_teclado
   'si estamos en modo de identificación no podemos hacer
```

```
        'nada hasta salir del modo de identificación
        if Identificacion = "i" then
          select case tecla_presionada
          case 2      'Identificación del sistema
              Estado = 5000
          end select
        else
          select case tecla_presionada
          case "A"
              Estado = 500
          case "B"
              Estado = 700
          case "C"
              Estado = 900
          case "D"
              Estado = 1100
          case 9      'Depuración de sensores
              Estado = 2000
          case "*"    'Cambiar sentido de giro
              Estado = 3000
          case 1      'Cambiar Lazo cerrado a abierto y viceversa
              Estado = 4000
          case 2      'Identificación del sistema
              Estado = 5000
          end select
        end if
        Inc(Contador)
        if Contador > 60000 then
           Contador = 0
           if Lazo = "C" then
              Estado = 50
           else
              Estado = 30
           end if
        end if
    Case 50
         Lcd_Cmd(LCD_CURSOR_OFF)
         Lcd_Cmd(LCD_CLEAR)
         Lcd_Out(1, 1, "Kp   Ki   Kd   ")
'se lo coloca en este formato
```

```
            Lcd_Out(2, 1, "0.00 0.00 0.00")
            xKp = Kp * 100.00
            WordToStr(xKp, txt)
      'se muestran los valores en las posiciones, digito por digito
      'la matriz de datos stirng comienza en cero, por ello se desplazan los valores
desde 1
           if txt[2] <> " " then
              Lcd_Chr(2, 1, txt[2])
            end if
            if txt[3] <> " " then
              Lcd_Chr(2, 3, txt[3])
            end if
            Lcd_Chr(2, 4, txt[4])

            xKi = Ki * 100.00
            WordToStr(xKi, txt)
            if txt[2] <> " " then
              Lcd_Chr(2, 6, txt[2])
            end if
            if txt[3] <> " " then
              Lcd_Chr(2, 8, txt[3])
            end if
            Lcd_Chr(2, 9, txt[4])

            xKd = Kd * 100.00
            WordToStr(xKd, txt)
            if txt[2] <> " " then
              Lcd_Chr(2, 11, txt[2])
            end if
            if txt[3] <> " " then
              Lcd_Chr(2, 13, txt[3])
            end if
            Lcd_Chr(2, 14, txt[4])

            Estado = 60
         Case 60
           'se obtiene el valor segun tecla presionada
           tecla_presionada = leer_teclado
           select case tecla_presionada
           case "A"
```

```
        Estado = 500
    case "B"
        Estado = 700
    case "C"
        Estado = 900
    case "D"
        Estado = 1100
    case 9      'Depuración de sensores
        Estado = 2000
    case "*"    'Cambiar sentido de giro
        Estado = 3000
    case 1      'Cambiar Lazo cerrado a abierto y viceversa
        Estado = 4000
    case 2      'Identificación del sistema
        Estado = 5000
    end select
    Inc(Contador)
    if Contador > 60000 then
      Contador = 0
      Estado = 30
    end if
 Case 500
    Lcd_Cmd(LCD_CLEAR)
    if Lazo = "C" then
      Lcd_Out(1, 1, "Ingrese SetPoint")
    else
      Lcd_Out(1, 1, "Ingrese Ciclo de")
      Lcd_Out(2, 1, "Trabajo %")
    end if
    KT1=0
    KT2=0
    KT3=0
    KT4=0
    KT5=0
    Estado= 510
case 510
    KT1 = leer_teclado
    if KT1 <> nada then
      Estado= 520
    end if
```

```
 case 520
     select case KT1
     case tecla_enter, tecla_clear
         Estado= 510
         KT1 = 0
     case "A", "B", "C", "D"
         Estado = 510
     case else
         Lcd_Chr(2, 16, KT1+$30)
         Estado= 530
         xSetPoint=KT1
     end select
case 530
     KT2 = leer_teclado
     if KT2 <> nada then
       Estado= 540
     end if
case 540
     select case KT2
     case tecla_clear      'este es el clear
       Estado= 500
       KT2 = 0
     case tecla_enter      'este es el enter
       Estado= 610
       KT2 = 0
     case "A", "B", "C", "D"
         Estado = 530
     case else
       Lcd_Chr(2, 15, KT1+$30)
       Lcd_Chr(2, 16, KT2+$30)
       Estado= 550
       xSetPoint=(KT1*10)+ KT2
     end select
 case 550
     KT3 = leer_teclado
     if KT3 <> nada then
       Estado= 560
     end if
 case 560
     select case KT3
```

```
    case tecla_clear      'este es el clear
      Estado= 500
      KT3 = 0
    case tecla_enter      'este es el enter
      Estado= 610
      KT3 = 0
    case "A", "B", "C", "D"
        Estado = 550
    case else
      Lcd_Chr(2, 14, KT1+$30)
      Lcd_Chr(2, 15, KT2+$30)
      Lcd_Chr(2, 16, KT3+$30)
      Estado= 570
      xSetPoint=(KT1*100) + (KT2*10) +KT3
    end select
case 570
    KT4 = leer_teclado
    if KT4 <> nada then
      Estado= 580
    end if
case 580
    select case KT4
    case tecla_clear      'este es el clear
      Estado= 0
      KT4 = 0
    case tecla_enter      'este es el enter
      Estado= 610
      KT4 = 0
    case "A", "B", "C", "D"
        Estado = 570
    case else
      Lcd_Chr(2, 13, KT1+$30)
      Lcd_Chr(2, 14, KT2+$30)
      Lcd_Chr(2, 15, KT3+$30)
      Lcd_Chr(2, 16, KT4+$30)
      Estado= 590
      xSetPoint=(KT1*1000) + (KT2*100) + (KT3*10) + KT4
    end select
case 590
    KT5 = leer_teclado
```

```
    if KT5 <> nada then
      Estado= 600
    end if
case 600
    select case KT5
    case tecla_clear      'este es el clear
      Estado = 500
    case tecla_enter      'este es el enter
      Estado = 610
    case else
      Estado= 590
    end select
case 610
                   'Fin de cambio de setpoint
    setpoint = xSetPoint
    fSetPoint = xSetPoint / Desplazamiento_a_RPM
    xSetPoint = fSetPoint
    'Enviamos el aviso que vamos a enviar el setpoint
    Uart2_Write_Char(255)
    'Calculamos el setpoint para enviar
    Uart2_Write_Char(xSetPoint)
    'Si estamos en Lazo abierto establecemos la velocidad del motor
    if Lazo = "A" then
      if setpoint <> 0 then
        if setpoint > 100 then
          setpoint = 100
        end if
        if setpoint < 15 then
          setpoint = 15
        end if
      end if
      PWMfloat = ( 2.00 * duty_50 * setpoint ) / 100
      PWM = PWMfloat
      Pwm_Mc_Set_Duty(PWM, 1)   'establecemos el ciclo de trabajo
    end if
    'Regresamos al programa principal
    Estado = 30
case 700
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "Ingrese Kp")
```

```
    KT1=0
    KT2=0
    KT3=0
    KT4=0
    KT5=0
    Estado= 710
case 710
    KT1 = leer_teclado
    if KT1 <> nada then
      Estado= 720
    end if
case 720
    select case KT1
       case tecla_enter, tecla_clear
         Estado= 710
         KT1 = 0
       case "A", "B", "C", "D"
         Estado= 710
         KT1 = 0
       case else
         Lcd_Chr(2, 13, "0")
         Lcd_Chr(2, 14, ".")
         Lcd_Chr(2, 15, "0")
         Lcd_Chr(2, 16, KT1+$30)
         Estado= 730
         xKp = KT1
       end select
case 730
    KT2 = leer_teclado
    if KT2 <> nada then
      Estado= 740
    end if
case 740
    select case KT2
    case tecla_clear      'este es el clear
      Estado= 700
      KT2 = 0
    case tecla_enter      'este es el enter
      Estado= 800
      KT2 = 0
```

```
      case "A", "B", "C", "D"
        Estado= 730
        KT2 = 0
      case else
        Lcd_Chr(2, 13, "0")
        Lcd_Chr(2, 14, ".")
        Lcd_Chr(2, 15, KT1+$30)
        Lcd_Chr(2, 16, KT2+$30)
        Estado= 750
        xKp =(KT1*10)+ KT2
      end select
  case 750
      KT3 = leer_teclado
      if KT3 <> nada then
        Estado= 760
      end if
  case 760
      select case KT3
      case tecla_clear      'este es el clear
        Estado= 700
        KT3 = 0
      case tecla_enter      'este es el enter
        Estado= 800
        KT3 = 0
      case "A", "B", "C", "D"
        Estado= 750
        KT3 = 0
      case else
        Lcd_Chr(2, 13, KT1+$30)
        Lcd_Chr(2, 14, ".")
        Lcd_Chr(2, 15, KT2+$30)
        Lcd_Chr(2, 16, KT3+$30)
        Estado= 770
        xKp =(KT1*100) + (KT2*10) +KT3
      end select
  case 770
      KT5 = leer_teclado
      if KT5 <> nada then
        Estado= 780
      end if
```

```
case 780
    select case KT5
    case tecla_clear      'este es el clear
      Estado = 700
    case tecla_enter      'este es el enter
      Estado = 800
    case "A", "B", "C", "D"
      Estado = 770
    case else
      Estado= 770
    end select
case 800
                  'Fin de cambio de Kp
    Kp = xKp / 100
    Estado = 30

case 900
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "Ingrese Ki")
    KT1=0
    KT2=0
    KT3=0
    KT4=0
    KT5=0
    Estado= 910
case 910
    KT1 = leer_teclado
    if KT1 <> nada then
      Estado= 920
    end if
case 920
    select case KT1
       case tecla_enter, tecla_clear
         Estado= 910
         KT1 = 0
       case "A", "B", "C", "D"
         Estado = 910
         KT1 = 0
       case else
'               Lcd_Chr(2, 12, "0")
```

```
            Lcd_Chr(2, 13, "0")
            Lcd_Chr(2, 14, ".")
            Lcd_Chr(2, 15, "0")
            Lcd_Chr(2, 16, KT1+$30)
            Estado= 930
            xKi = KT1
        end select
    case 930
        KT2 = leer_teclado
        if KT2 <> nada then
          Estado= 940
        end if
    case 940
        select case KT2
        case tecla_clear      'este es el clear
          Estado= 900
          KT2 = 0
        case tecla_enter      'este es el enter
          Estado= 1000
          KT2 = 0
        case "A", "B", "C", "D"
          Estado = 930
          KT2 = 0
        case else
'          Lcd_Chr(2, 12, "0")
          Lcd_Chr(2, 13, "0")
          Lcd_Chr(2, 14, ".")
          Lcd_Chr(2, 15, KT1+$30)
          Lcd_Chr(2, 16, KT2+$30)
          Estado= 950
          xKi =(KT1*10)+ KT2
        end select
    case 950
        KT3 = leer_teclado
        if KT3 <> nada then
          Estado= 960
        end if
    case 960
        select case KT3
        case tecla_clear      'este es el clear
```

```
      Estado= 900
      KT3 = 0
    case tecla_enter      'este es el enter
      Estado= 1000
      KT3 = 0
    case "A", "B", "C", "D"
      Estado = 950
      KT3 = 0
    case else
      Lcd_Chr(2, 13, KT1+$30)
      Lcd_Chr(2, 14, ".")
      Lcd_Chr(2, 15, KT2+$30)
      Lcd_Chr(2, 16, KT3+$30)
      Estado= 970
      xKi =(KT1*100) + (KT2*10) +KT3
    end select
case 970
    KT5 = leer_teclado
    if KT5 <> nada then
      Estado= 980
    end if
case 980
    select case KT5
    case tecla_clear      'este es el clear
      Estado = 900
    case tecla_enter      'este es el enter
      Estado = 1000
    case "A", "B", "C", "D"
      Estado = 970
    case else
      Estado = 970
    end select
case 1000
    'Fin de cambio de Kp
    Ki = xKi / 100
    Estado = 30

         case 1100
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "Ingrese Kd")
```

```
        KT1=0
        KT2=0
        KT3=0
        KT4=0
        KT5=0
        Estado= 1110
    case 1110
        KT1 = leer_teclado
        if KT1 <> nada then
          Estado= 1120
        end if
    case 1120
        select case KT1
           case tecla_enter, tecla_clear
             Estado= 1110
             KT1 = 0
           case "A", "B", "C", "D"
             Estado = 1110
             KT1 = 0
           case else
'             Lcd_Chr(2, 12, "0")
             Lcd_Chr(2, 13, "0")
             Lcd_Chr(2, 14, ".")
             Lcd_Chr(2, 15, "0")
             Lcd_Chr(2, 16, KT1+$30)
             Estado= 1130
             xKd = KT1
        end select
    case 1130
        KT2 = leer_teclado
        if KT2 <> nada then
          Estado= 1140
        end if
    case 1140
        select case KT2
        case tecla_clear      'este es el clear
          Estado= 1100
          KT2 = 0
        case tecla_enter      'este es el enter
          Estado= 1200
```

```
            KT2 = 0
          case "A", "B", "C", "D"
            Estado = 1130
            KT2 = 0
          case else
'             Lcd_Chr(2, 12, "0")
            Lcd_Chr(2, 13, "0")
            Lcd_Chr(2, 14, ".")
            Lcd_Chr(2, 15, KT1+$30)
            Lcd_Chr(2, 16, KT2+$30)
            Estado= 1150
            xKd =(KT1*10)+ KT2
          end select
      case 1150
          KT3 = leer_teclado
          if KT3 <> nada then
            Estado= 1160
          end if
      case 1160
          select case KT3
          case tecla_clear      'este es el clear
            Estado= 1100
            KT3 = 0
          case tecla_enter      'este es el enter
            Estado= 1200
            KT3 = 0
          case "A", "B", "C", "D"
            Estado = 1150
            KT3 = 0
          case else
            Lcd_Chr(2, 13, KT1+$30)
            Lcd_Chr(2, 14, ".")
            Lcd_Chr(2, 15, KT2+$30)
            Lcd_Chr(2, 16, KT3+$30)
            Estado= 1170
            xKd =(KT1*100) + (KT2*10) +KT3
          end select
      case 1170
          KT5 = leer_teclado
          if KT5 <> nada then
```

```
      Estado= 1180
    end if
case 1180
    select case KT5
    case tecla_clear      'este es el clear
      Estado = 1100
    case tecla_enter      'este es el enter
      Estado = 1200
    case "A", "B", "C", "D"
      Estado = 1170
    case else
      Estado= 1170
    end select
case 1200
    'Fin de cambio de Kp
    Kd = xKd / 100
    Estado = 30
case 2000
    Lcd_Cmd(LCD_CLEAR)
    Lcd_Out(1, 1, "Sensores  #Salir")
    Pin_Led = 0
    Estado = 2100
case 2100
    if Sensor_Indice = 1 then
      Lcd_Chr(2,1,"1")
    else
      Lcd_Chr(2,1,"0")
    end if

    if Sensor_A = 1 then
      Lcd_Chr(2,3,"1")
    else
      Lcd_Chr(2,3,"0")
    end if

    if Sensor_B = 1 then
      Lcd_Chr(2,5,"1")
    else
      Lcd_Chr(2,5,"0")
    end if
```

```
   WordToStr(POSCNT, txt)
   Lcd_Out(2,11,txt)

   Pin_Led_Verde = 1
   delay_ms(50)
   Pin_Led_Verde = 0
   delay_ms(50)

   'se obtiene el valor segun tecla presionada
   tecla_presionada = leer_teclado
   select case tecla_presionada
   case "#"
       Estado = 30
   end select
Case 3000     'Cambio de giro
   Lcd_Cmd(LCD_CLEAR)
   Lcd_Out(1, 1, "Cambiando       ")
   Lcd_Out(2, 1, "      direccion ")
   'Apagamos el motor
   xSetPoint = setpoint 'Almacenamos el setpoint anterior
   setpoint = 0
   fSetPoint = setpoint
   'Esperamos 1 segundo
   delay_ms(1000)
   'Cambiamos la dirección de giro
   if Motor_Izquierda = 1 then
     Motor_Izquierda = 0
     Motor_Derecha = 1
   else
     Motor_Izquierda = 1
     Motor_Derecha = 0
   end if
   'Establecemos la nueva dirección
   setpoint = xSetPoint
   fSetPoint = xSetPoint / Desplazamiento_a_RPM
   'Volvemos al programa principal
   Estado = 30
Case 4000
   Lcd_Cmd(LCD_CLEAR)
```

```
            Lcd_Out(1, 1, "Cambiando        ")
            if Lazo = "C" then
              Lazo = "A"
              Lcd_Out(2, 1, " a Lazo Abierto ")
            else
              Lazo = "C"
              Lcd_Out(2, 1, " a Lazo Cerrado ")
            end if
            'Esperamos 1 segundo
            delay_ms(1000)
            'Volvemos al programa principal
            Estado = 30
          Case 5000  'Identificación del sistema
            Lcd_Cmd(LCD_CLEAR)
            Lcd_Out(1, 1, "Identif. de     ")
            if Identificacion = " " then
              Identificacion = "i"
              Lazo = "A"
              Ident_Temporizador = 0
              Ident_Cual = 0
              Lcd_Out(2, 1, "Sistema Activado")
            else
              Identificacion = " "
              Lazo = "C"
              Lcd_Out(2, 1, "Sistema Desactiv")
            end if
            'Esperamos 1 segundo
            delay_ms(1000)
            'Volvemos al programa principal
            Estado = 30
          Case Else
            Lcd_Out(1, 1, "Estado no existe")
            Lcd_Out(2, 1, "                ")
            WordToStr(Estado, txt)
            Lcd_Out(2, 1, txt)
            setpoint = 0
            fSetPoint = setpoint
      end select
    wend
end.
```

CODIGO FUENTE DEL PROGRAMA REALIZADO EN VISIAL BASIC PARA LA VISUALIZACION DE DATOS

```
'constantes
Const Factor = 15.625
Const SetPointMaximo = 100

'variables globales
Dim AnchoGrafico As Integer
Dim SetPoint As Integer
Dim ArchivoAbierto As Boolean
Dim Archivo As Integer
Dim TiempoAnterior As Single
'Indica que el próximo byte es el SetPoint
Dim VieneElSetPoint As Boolean

Private Sub btnCambiarDestino_Click() 'aqui se escoje donde se almacenara el
archivo en excel
   CommonDialog1.DefaultExt = "xls"
   CommonDialog1.ShowSave
   If CommonDialog1.FileName = "" Then Exit Sub
   txtRegistro.Text = CommonDialog1.FileName
End Sub

Private Sub btnEnviar_Click()
Dim xEnviar As Byte
Dim xSetPoint As Integer
   xSetPoint = txtSetPoint.Text
   xSetPoint = xSetPoint / Factor   'se divide para el factor para enviar el dato en
valores que el PIC reconozca dimension byte
   If xSetPoint <= SetPointMaximo Then  'se determina el valor del set point que se
puede enviar y variar desde el programa
      xEnviar = xSetPoint
   Else
      xEnviar = SetPointMaximo
   End If
```

```
   SetPoint = txtSetPoint.Text
   MSComm1.Output = Chr(xEnviar)
End Sub

Private Sub btnPuerto_Click()  'procedimiento para abrir el puerto y actibar botones
   MSComm1.CommPort = txtPuerto.Text
   MSComm1.PortOpen = True
   btnPuerto.Enabled = False
   btnEnviar.Enabled = True
End Sub

Private Sub chkAutoEscala_Click()
   MSChart1.Plot.Axis(VtChAxisIdY).ValueScale.Auto = (chkAutoEscala.Value = 1)
   txtEscala.Enabled = Not (chkAutoEscala.Value = 1)
   txtMajorDivision.Enabled = Not (chkAutoEscala.Value = 1)
   If chkAutoEscala.Value = 0 Then
      If txtEscala.Text > 0 Then
         MSChart1.Plot.Axis(VtChAxisIdY).ValueScale.Maximum = txtEscala.Text
      End If
      If txtMajorDivision.Text > 0 Then
         MSChart1.Plot.Axis(VtChAxisIdY).ValueScale.MajorDivision =
txtMajorDivision.Text
      End If
   End If
End Sub

Private Sub chkRegistrar_Click()  'Aqui habilitamos la grabacion de los datos en el
archivo
   btnCambiarDestino.Enabled = (chkRegistrar.Value = 0)
   If chkRegistrar.Value = 0 Then
      If ArchivoAbierto Then
         Close #Archivo
         ArchivoAbierto = False
      End If
   End If
End Sub

Private Sub Form_Load()
Dim xi As Integer
   AnchoGrafico = 512
```

```
   For xi = 1 To AnchoGrafico
      MSChart1.RowCount = xi      'control de serie de datos que se envian para
graficar
      MSChart1.Column = 1
      MSChart1.Row = xi
      MSChart1.Data = 0
      MSChart1.Column = 2        'control set point para graficar
      MSChart1.Row = xi
      MSChart1.Data = 0
   Next xi
   txtRegistro.Text = App.Path & "\" & "registro.xls"
   VieneElSetPoint = False
End Sub

Private Sub Form_Unload(Cancel As Integer) 'finalizacion del programa y cerrar
puerto de comunicacion
   If ArchivoAbierto Then
      Close #Archivo
   End If
   If MSComm1.PortOpen = True Then
      MSComm1.PortOpen = False
   End If
End Sub

Private Sub MSComm1_OnComm()
Dim xCadena As String
Dim xRecibido As Integer
Dim xRPM As Integer
Dim xi As Integer, yi As Integer
Dim xAux As Integer, xAux2 As Long
Dim xTiempo As Single
  Select Case MSComm1.CommEvent
  ' Controlar cada evento o error escribiendo
  ' código en cada instrucción Case

  ' Errores
    Case comBreak   ' Se ha recibido una interrupción.
    Case comEventFrame   ' Error de trama
    Case comEventOverrun   ' Datos perdidos.
    Case comEventRxOver   ' Desbordamiento del búfer
```

```
                                  ' de recepción.
   Case comEventRxParity   ' Error de paridad.
   Case comEventTxFull   ' Búfer de transmisión lleno.
   Case comEventDCB   ' Error inesperado al recuperar DCB.

' Eventos
   Case comEvCD   ' Cambio en la línea CD.
   Case comEvCTS   ' Cambio en la línea CTS.
   Case comEvDSR   ' Cambio en la línea DSR.
   Case comEvRing   ' Cambio en el indicador de
               ' llamadas.
   Case comEvReceive   ' Recibido nº SThreshold de
                ' caracteres.
       xCadena = MSComm1.Input
       For yi = 1 To Len(xCadena)
          xRecibido = Asc(Mid(xCadena, yi, 1))
          If Not VieneElSetPoint Then
             If xRecibido < 255 Then
                'Entonces el PIC nos está reportando la velocidad en RPM
                xTiempo = Timer
                txtTiempo.Text = xTiempo - TiempoAnterior
                TiempoAnterior = xTiempo
                xRPM = xRecibido * Factor 'Escalamos la velocidad del motor
                txtDato.Text = xRPM

                MSChart1.RowCount = MSChart1.RowCount + 1
                MSChart1.Column = 1
                MSChart1.Row = MSChart1.RowCount
                MSChart1.Data = xRPM
                MSChart1.Column = 2
                MSChart1.Row = MSChart1.RowCount
                MSChart1.Data = SetPoint

                If chkRegistrar.Value = 1 Then
                   If Not ArchivoAbierto Then
                      'Si el archivo no está abierto
                      Archivo = FreeFile
                      Open txtRegistro.Text For Output As #Archivo
                      ArchivoAbierto = True
                   End If
```

```
                Print #Archivo, Now & vbTab & SetPoint & vbTab & xRPM
             End If

             If MSChart1.RowCount > AnchoGrafico Then
                For xi = 1 To AnchoGrafico
                   'Colocamos el dato
                   MSChart1.Column = 1
                   MSChart1.Row = xi + 1
                   xAux = MSChart1.Data
                   MSChart1.Row = xi
                   MSChart1.Data = xAux

                   'Colocamos la referencia
                   MSChart1.Column = 2
                   MSChart1.Row = xi + 1
                   xAux = MSChart1.Data
                   MSChart1.Row = xi
                   MSChart1.Data = xAux
                Next xi
                MSChart1.RowCount = AnchoGrafico
             End If
          Else
             'Si llegó el 255 entonces viene el SetPoint
             VieneElSetPoint = True
          End If
       Else
          'Si VieneElSetPoint
          SetPoint = xRecibido * Factor  'Escalamos el nuevo SetPoint
          'Ya no viene el SetPoint
          VieneElSetPoint = False
       End If
    Next yi
  Case comEvSend   ' Hay un SThreshold
           ' caracteres en el búfer
           ' de transmisión.
  Case comEvEOF   ' Se ha encontrado un carácter
           ' EOF en la entrada.
 End Select
End Sub
```

```
Private Sub txtAncho_Change()
   If txtAncho.Text <> "" Then
      If txtAncho.Text > 100 Then
         AnchoGrafico = txtAncho.Text
         MSChart1.RowCount = 1      'control de serie de datos que se envian para
graficar
         MSChart1.Column = 1
         MSChart1.Row = 1
         MSChart1.Column = 2        'control set point para graficar
         MSChart1.Row = 1
         For xi = 1 To AnchoGrafico
            MSChart1.RowCount = xi      'control de serie de datos que se envian para
graficar
            MSChart1.Column = 1
            MSChart1.Row = xi
            MSChart1.Data = 0
            MSChart1.Column = 2        'control set point para graficar
            MSChart1.Row = xi
            MSChart1.Data = 0
         Next xi
      End If
   End If
End Sub

Private Sub txtEscala_Change()
   If txtEscala.Text = "" Then
      If txtEscala.Text > 0 Then
         MSChart1.Plot.Axis(VtChAxisIdY).ValueScale.Maximum = txtEscala.Text
      End If
   End If
End Sub

Private Sub txtMajorDivision_Change()
   If txtMajorDivision.Text = "" Then
      If txtMajorDivision.Text > 0 Then
         MSChart1.Plot.Axis(VtChAxisIdY).ValueScale.Maximum =
txtMajorDivision.Text
      End If
   End If
End Sub
```

PROGRAMACIÓN LA PLATAFORMA LABVIEW

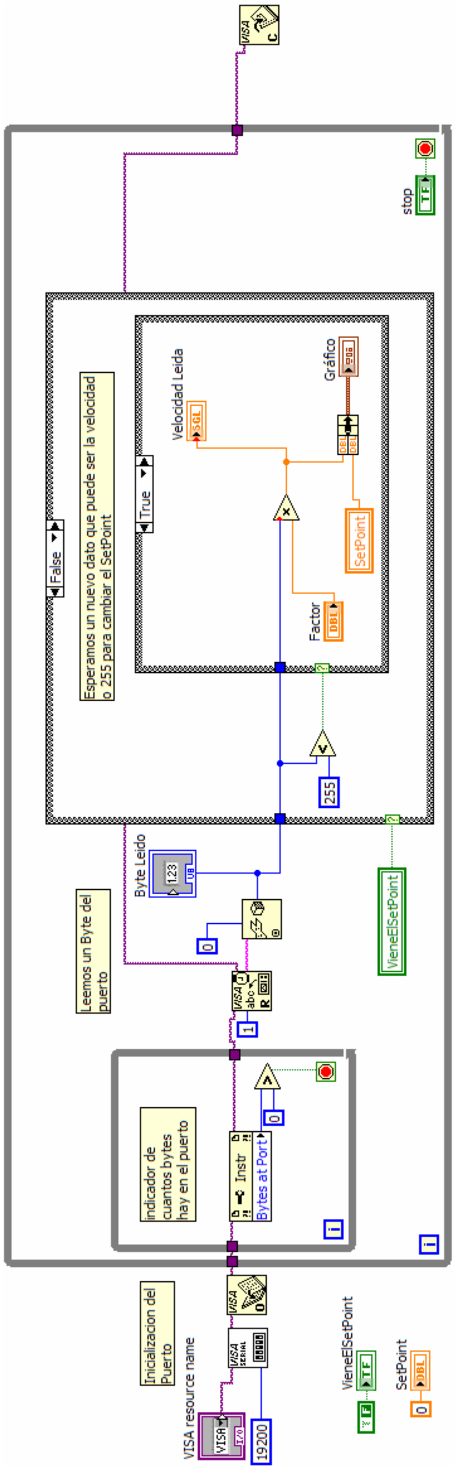


IMAGEN A.1

ANEXO B

DETALLE DE PLACA REALIZADA PARA EL PROYECTO

Se puede apreciar la placa realizada para el montaje del circuito del controlador, diseñada de tal forma que todos sus componentes puedan ser visibles y así se mantiene el objetivo de diseño didáctico de de fácil acceso y uso.

En esta también se aprecia la distribución en el teclado y visor LCD encontrándose distanciados del resto de los elementos de placa.

A continuación podemos apreciar el diseño.

1. CARA SUPERIOR

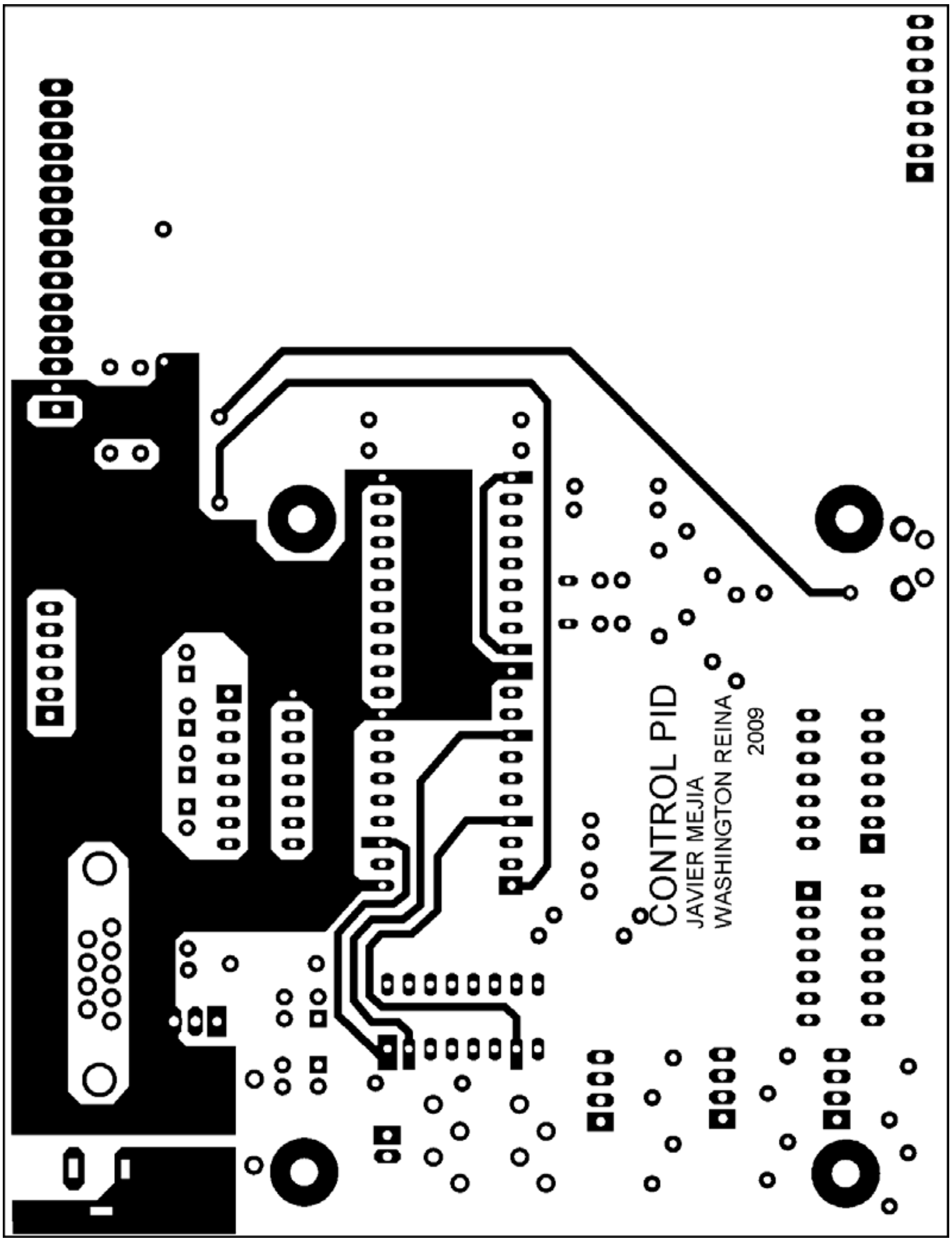


IMAGEN B.1

2. CARA INFERIOR

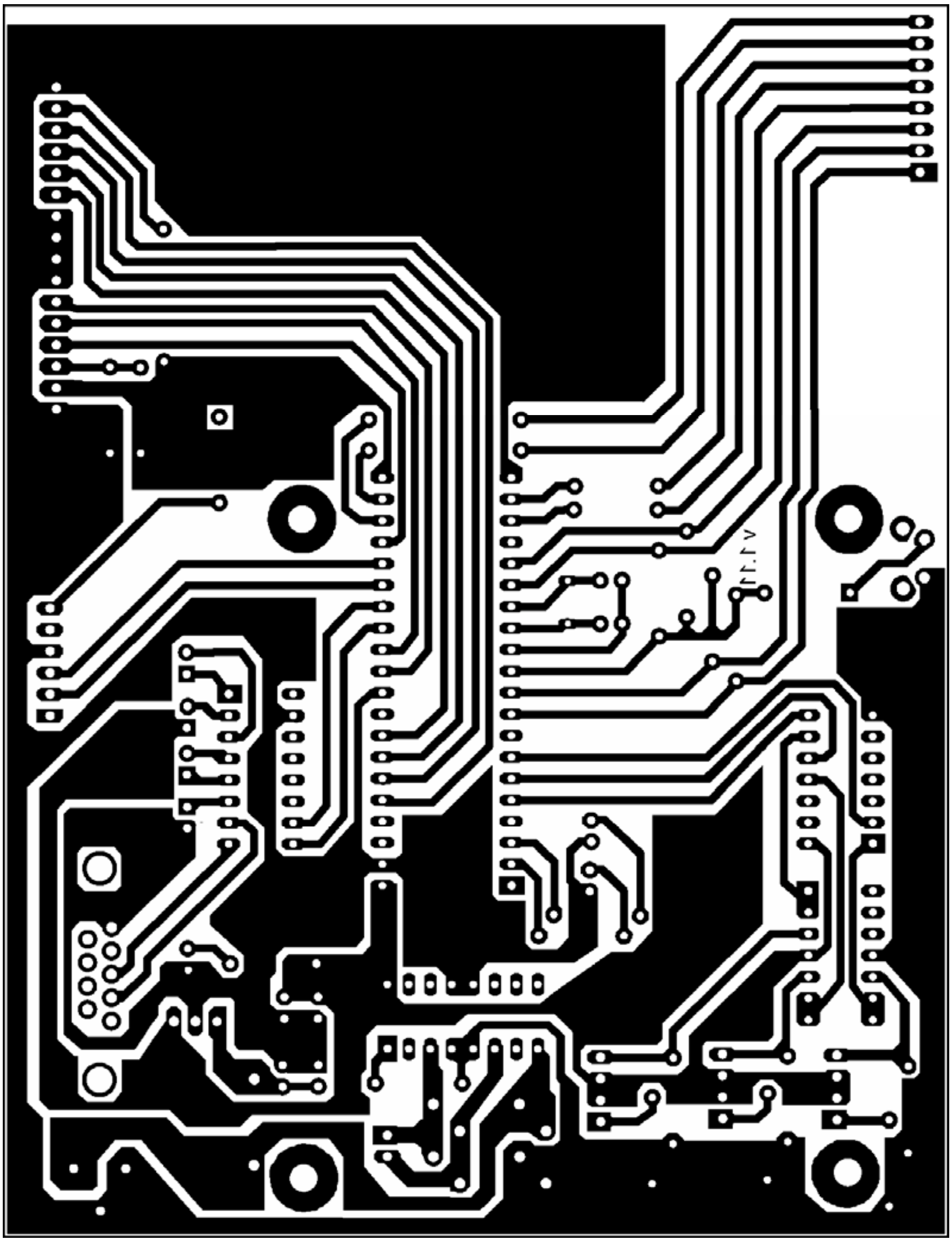


IMAGEN B.2

ANEXO C

DETALLE GRAFICO DE LA EXTRUCTURA DEL MONTAJE DEL MOTOR DC

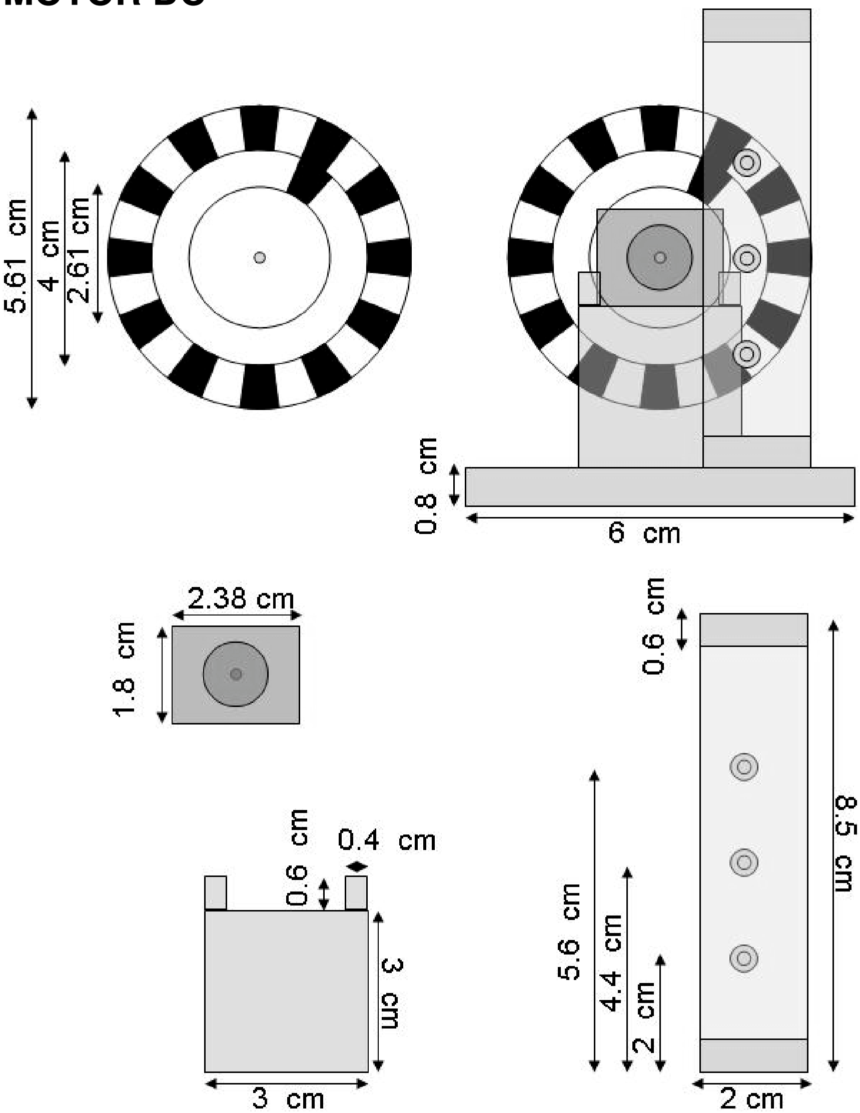


IMAGEN C.1

ANEXO D

ESQUEMATICO DE COMUNICACION SERIAL

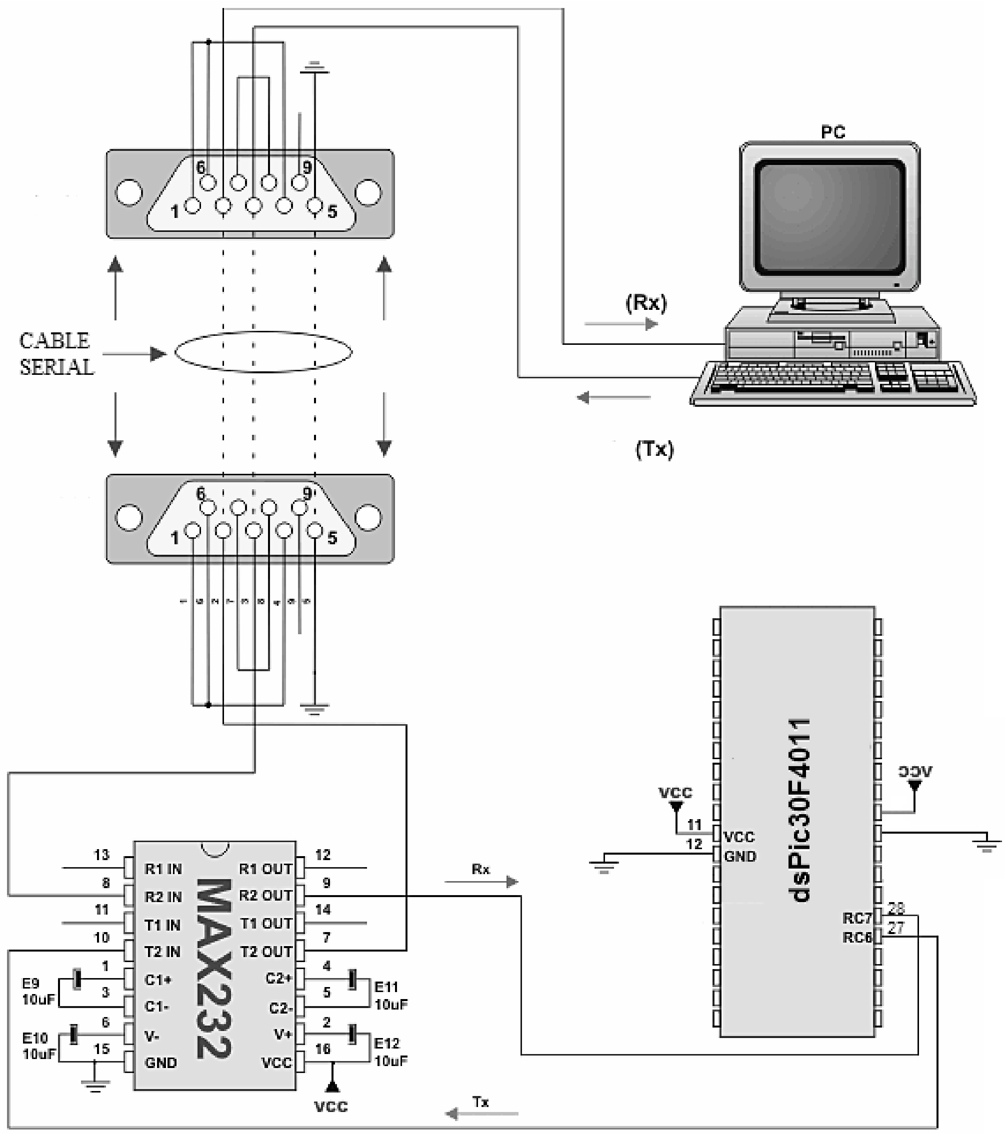


IMAGEN D.1 Fuente: ayuda de MikroBasic Max232

ANEXO E

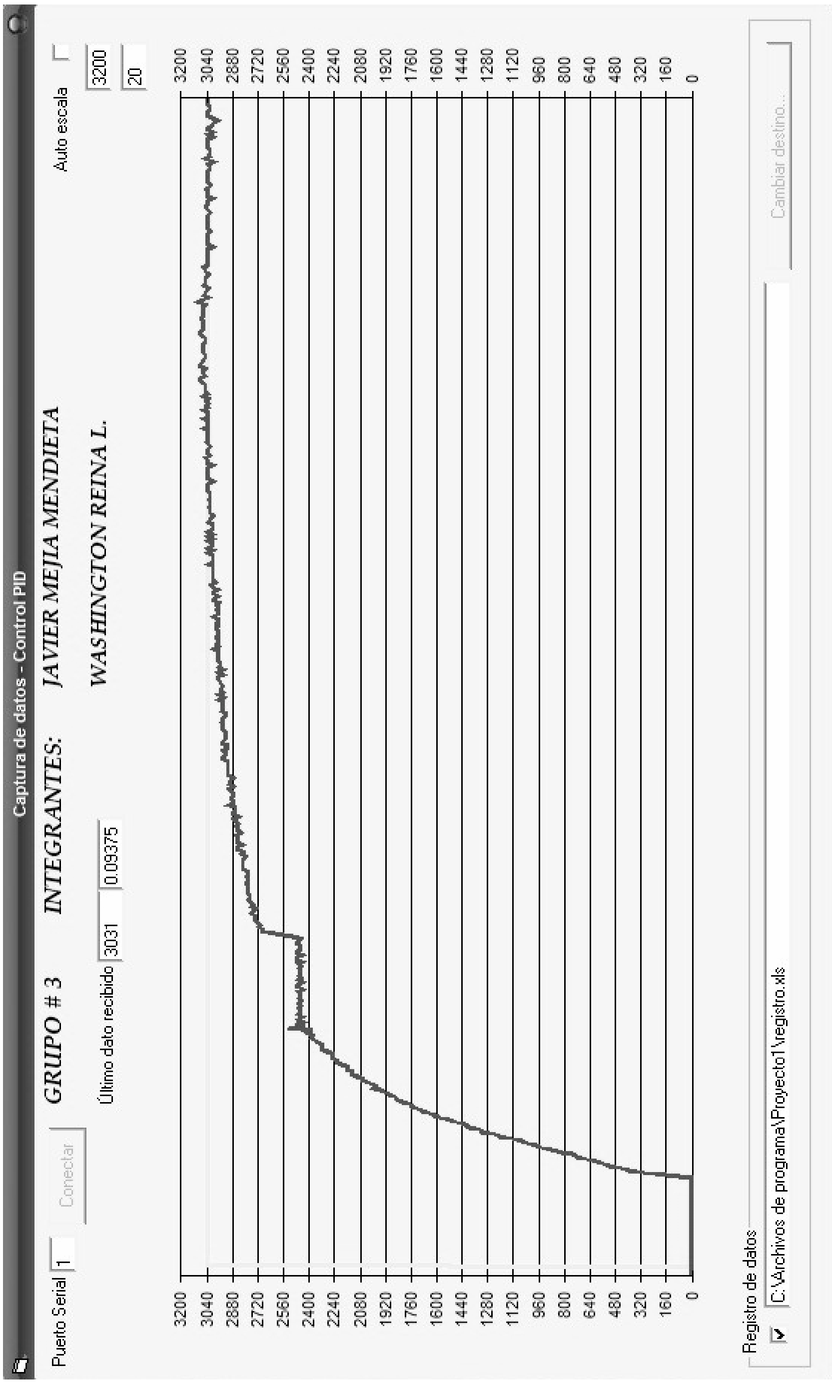
TABLA DE RUBROS PARCIALES Y TOTALES

CANTIDAD	ELEMENTO	P.UNIT.	TOTAL
2	Adaptador de fuente externa	0.30	0.60
2	Base socket 14 pines	0.80	1.60
2	Base socket 16 pines	0.80	1.60
1	Base socket 40 pines	1.50	1.50
1	Bicolor	0.25	0.25
1	C.I. 74ls14	0.70	0.70
1	C.I. 7805	0.40	0.40
1	C.I. dsPic30F4011	12.00	12.00
1	C.I. L293B	5.00	5.00
1	C.I. LM324	0.30	0.30
1	C.I. MAX232	3.50	3.50
1	Cable Utp	0.40	0.40
1	Cables varios	0.40	0.40
2	Capacitor 100uf/50v	0.20	0.40
4	Capacitor 10uf/50v	0.20	0.80
2	Capacitor ceramico 0.1uF	0.10	0.20
2	Capacitor ceramico 15pF/25V	0.10	0.20
1	Conector de 2 pines	0.15	0.15
3	Conector de 4 pines	0.25	0.75
1	Conector hembra DB9	0.60	0.60
1	Conector hembra db9 para placa	0.50	0.50
1	Conector macho DB9	0.60	0.60
1	Construcción tarjeta PCB doble cara	30.00	30.00
1	Cristal de 6 Mhz	0.65	0.65
5	Diodo 1n4004	0.05	0.25

CANTIDAD	ELEMENTO	P.UNIT.	TOTAL
1	Espadin hembra	1.20	1.20
1	Espadin macho	0.40	0.40
1	Espadin macho 90 grados	2.00	2.00
1	Fuente de 9V-1.5A	7.00	7.00
1	Led verde	0.05	0.05
1	Motor DC 9V	2.50	2.50
1	Pantalla LCD	10.00	10.00
3	Par emisor-receptor infrarojo	0.40	1.20
1	Plug para la fuente	0.25	0.25
1	Pulsador	0.20	0.20
22	Resistencia	0.03	0.66
1	Sellador, Pintura y base	10.00	10.00
1	Tiras de silicon y bases de caucho	1.50	1.50
1	Teclado	12.00	12.00
12	Tornillo y tuerca	5.00	5.00
1	Trabajo en Acrilico	15.00	15.00
	TOTAL		132.31

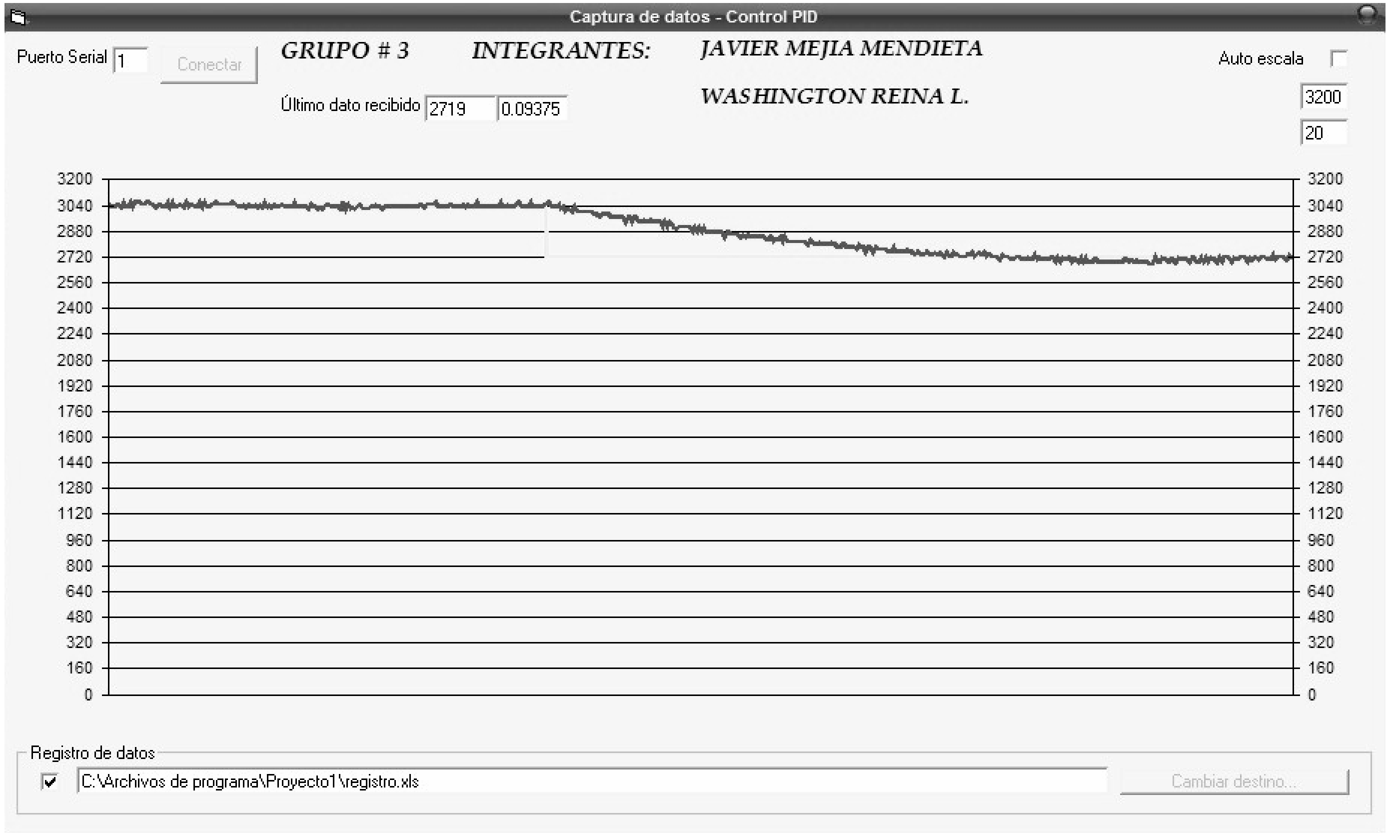
ANEXO F

GRAFICAS DE RESULTADOS OBTENIDOS EN VISUAL BASIC



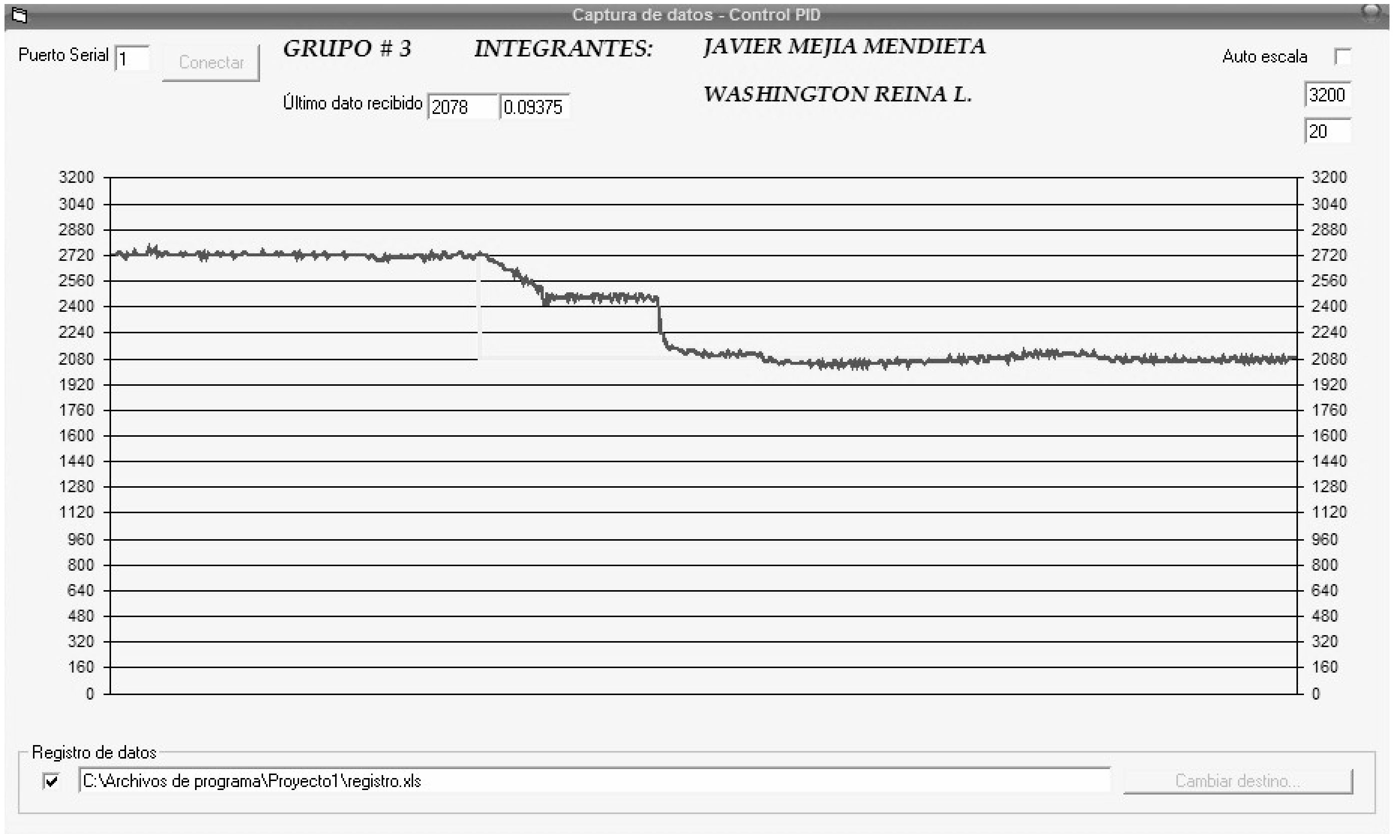
CAMBIO DE SetPoint de 0 a 3040

IMAGEN F.1



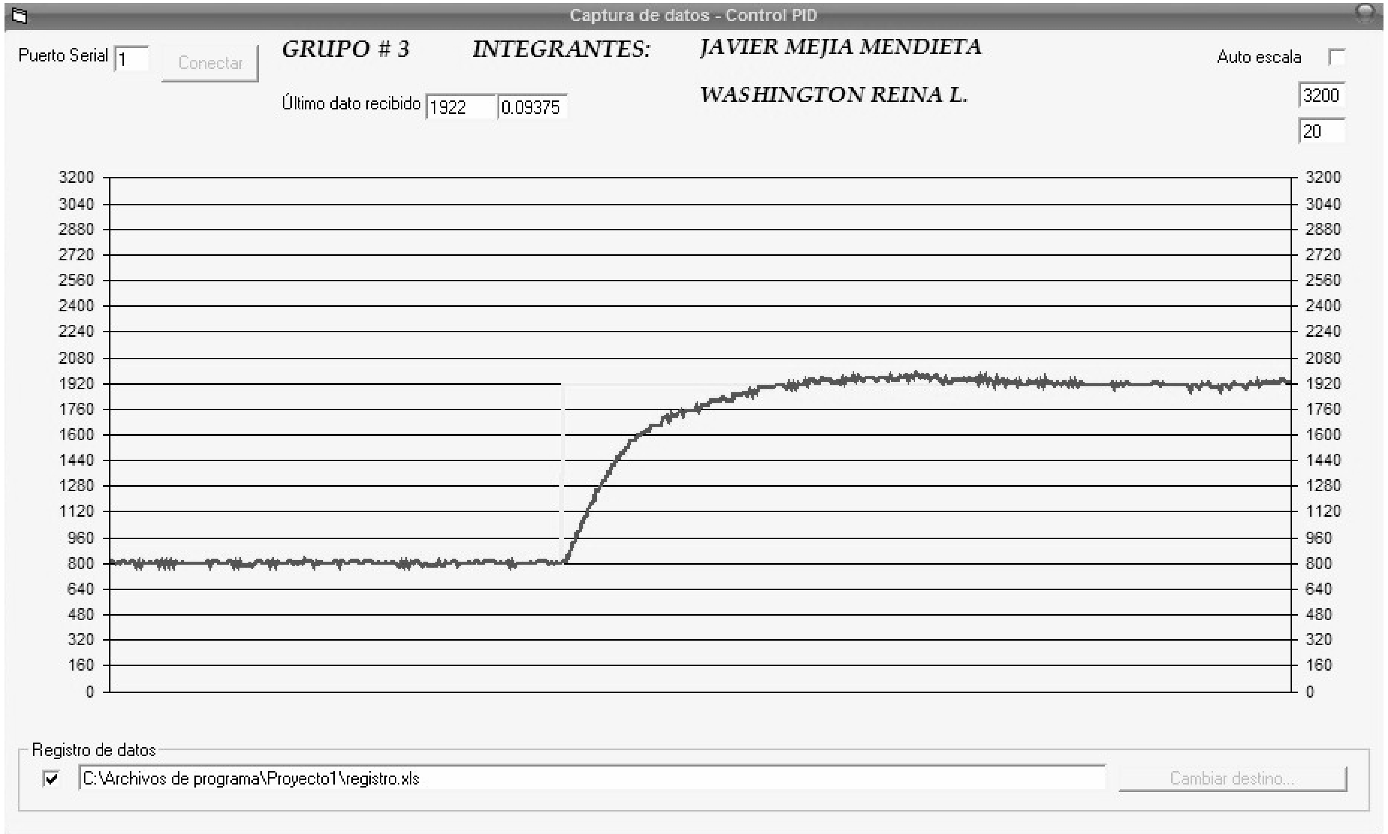
CAMBIO DE SetPoint de 3040 a 2720

IMAGEN F.2



CAMBIO DE SetPoint de 2720 a 2000

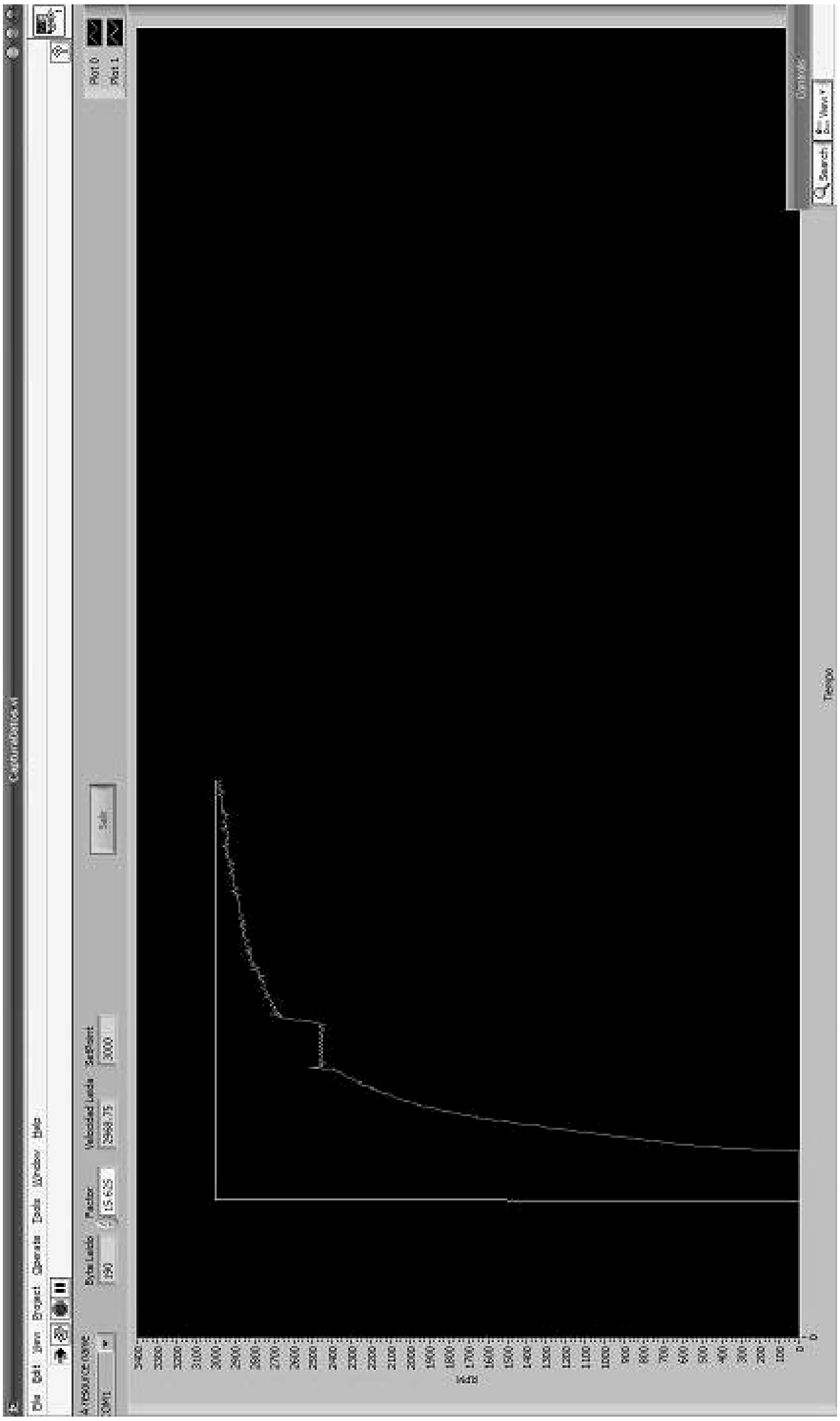
IMAGEN F.3



CAMBIO DE SetPoint de 800 a 1920

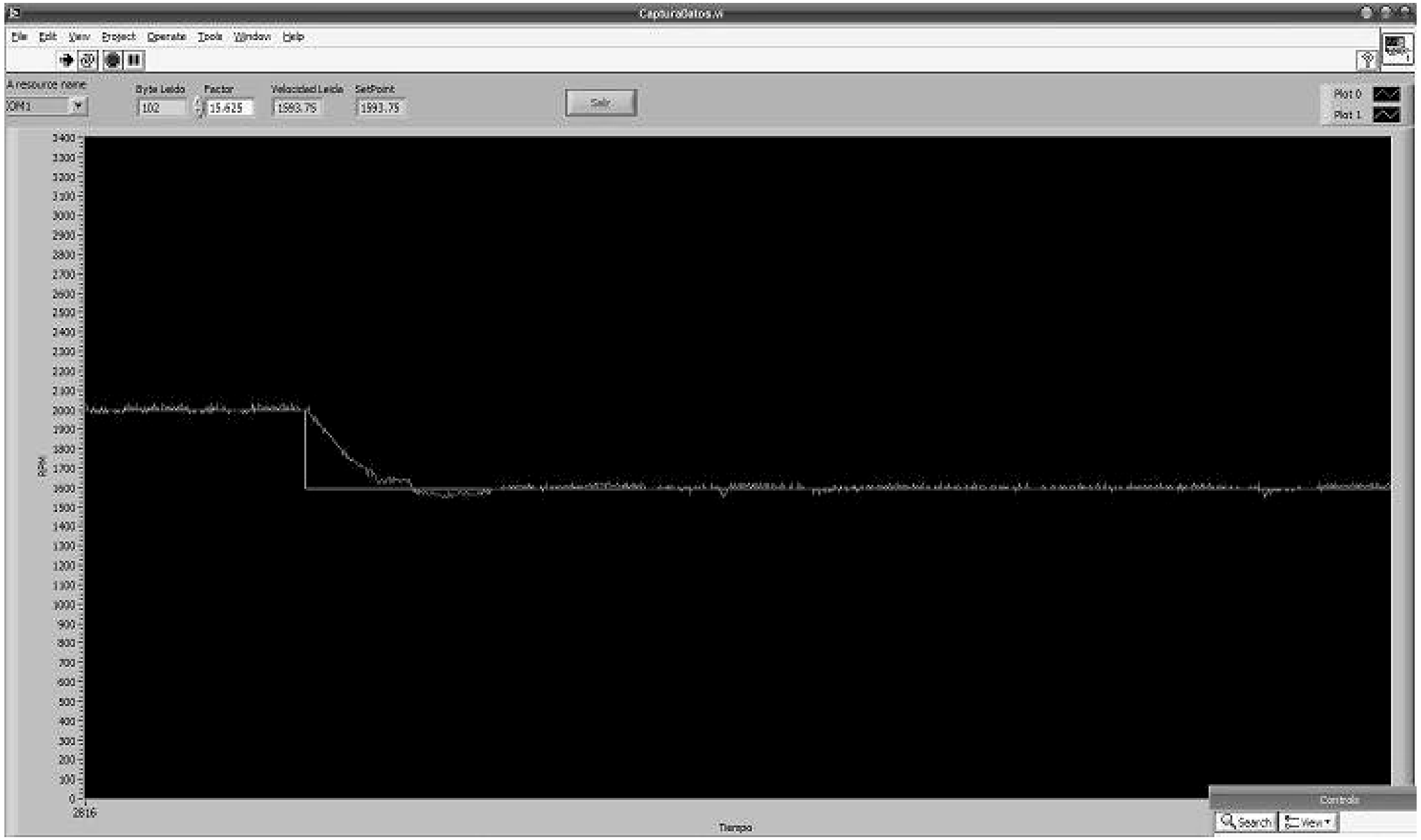
IMAGEN F.4

GRAFICAS DE RESULTADOS OBTENIDOS EN LA PLATAFORMA LAB VIEW



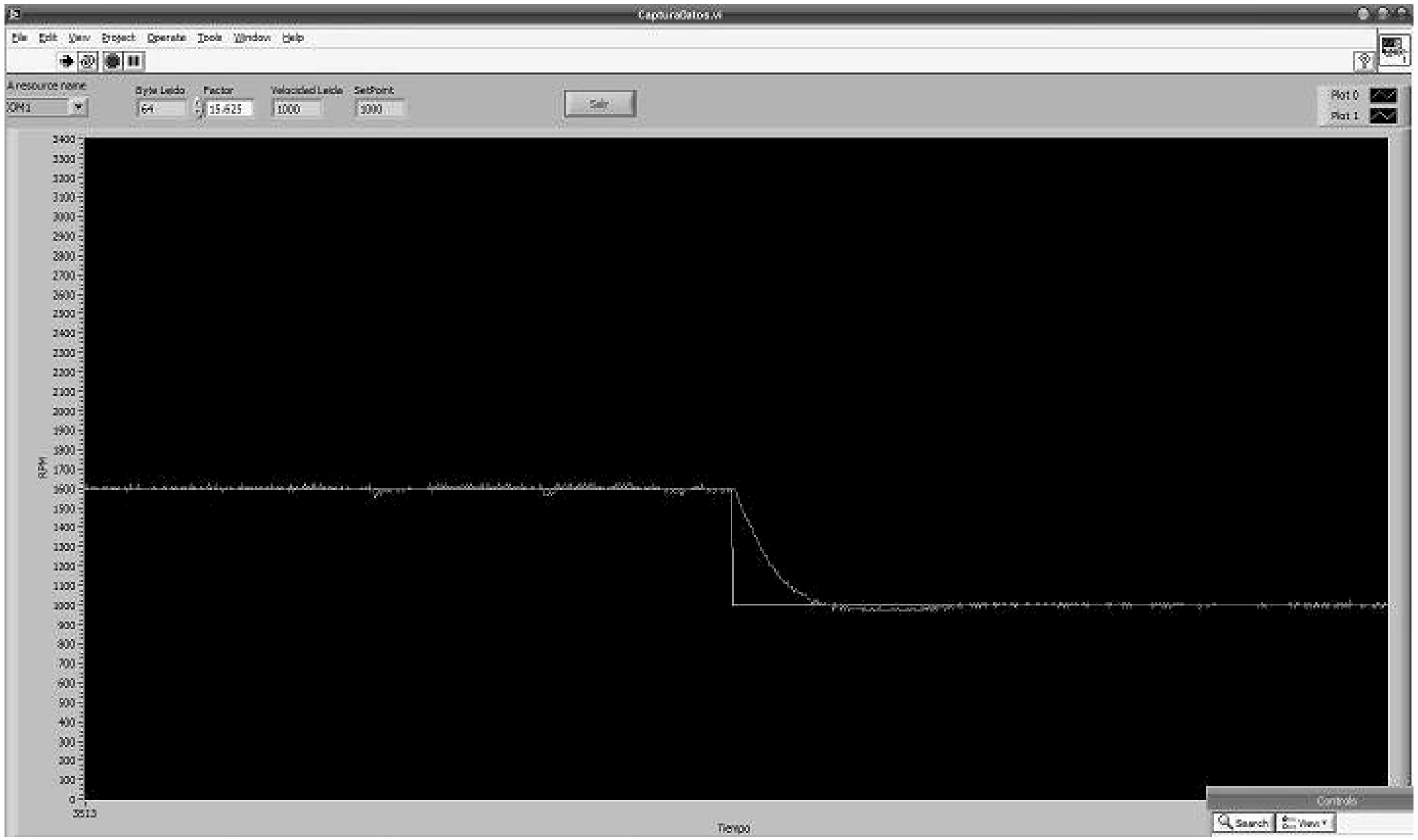
CAMBIO DE SetPoint de 0 a 3000

IMAGEN F.5



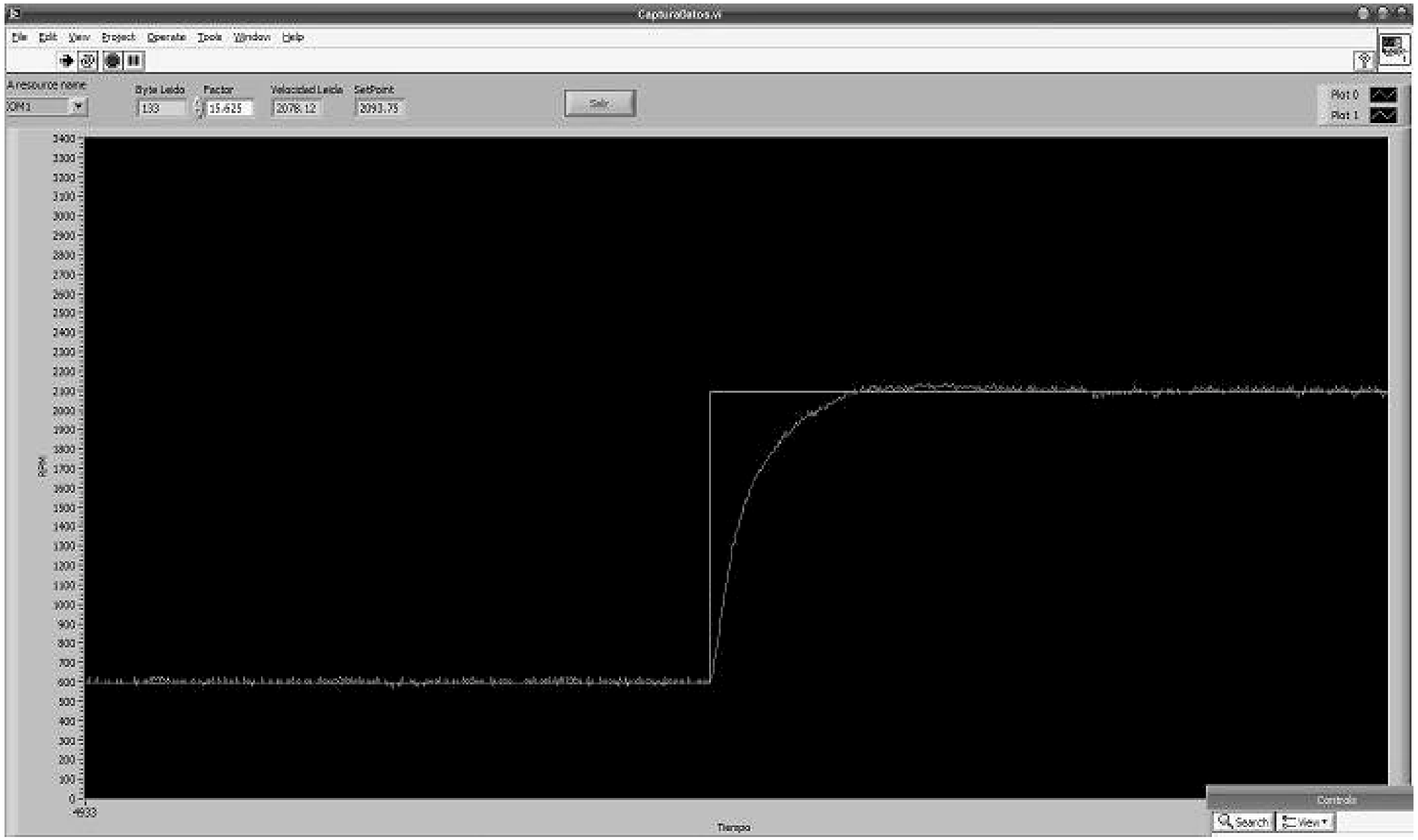
CAMBIO DE SetPoint de 2000 a 1600

IMAGEN F.6



CAMBIO DE SetPoint de 1600 a 1000

IMAGEN F.7

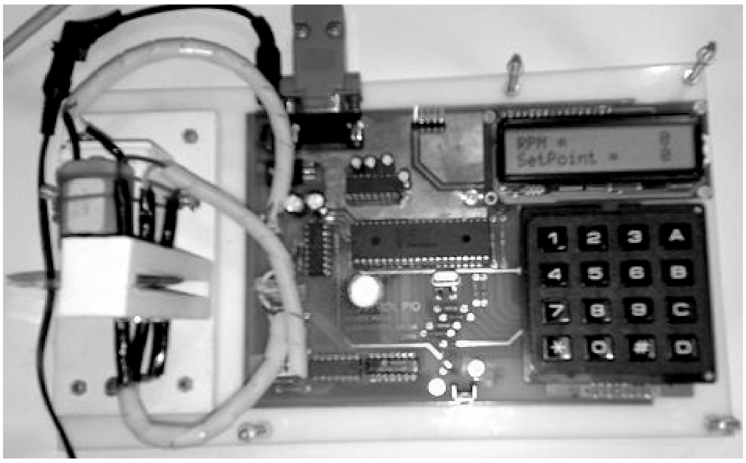


CAMBIO DE SetPoint de 600 a 2100

IMAGEN F.8

ANEXO G

FOTOGRAFIAS DEL PROYECTO TERMINADO



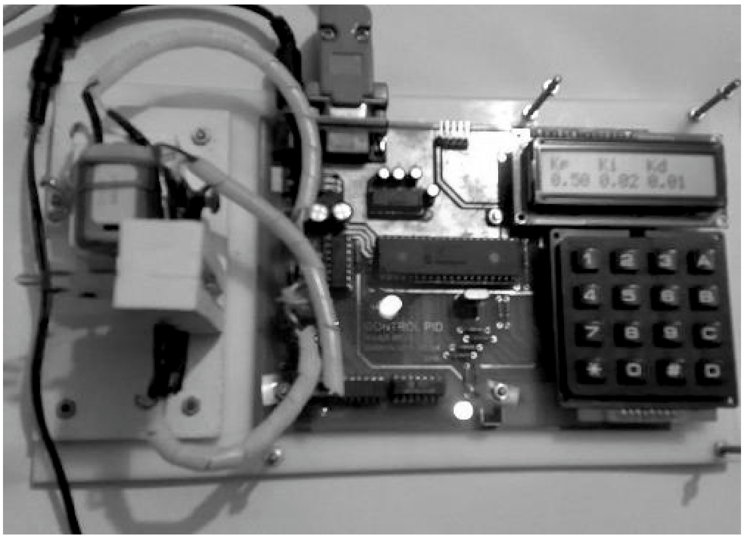


IMAGEN G.1 PROYECTO TERMINADO.
VISTA SUPERIOR

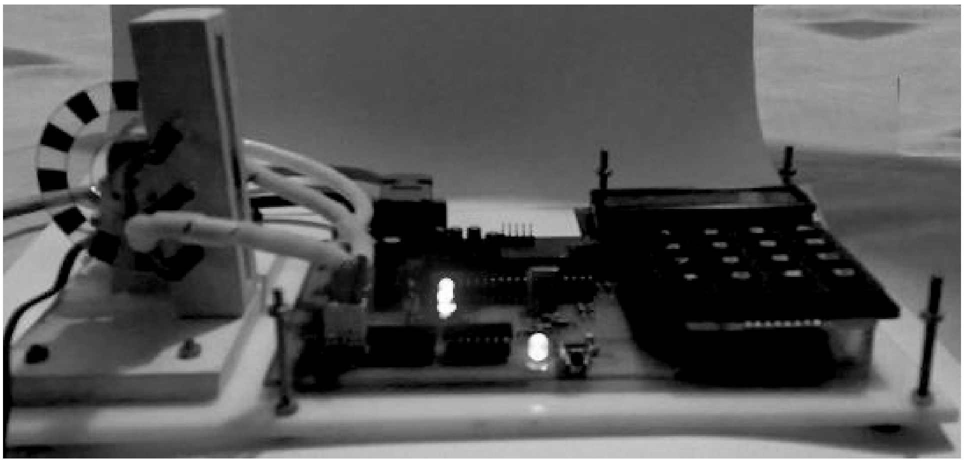


IMAGEN G.2 PROYECTO TERMINADO.
VISTA FRONTAL

MONTAJE DEL MOTOR
IMAGEN G.3

IMAGEN G.4 VISOR Y TECLADO

ANEXO H

MANUAL DE USUARIO

Al inicio, se debe energizar la unidad y a partir de ahí se pueden observar pantallas cambiantes con "MATERIA DE GRADUACIÓN" y los nombres de las personas que lo realizaron. Mientras esto sucede no se puede ejecutar ningún comando a excepción del RESET por medio de la botonera correspondiente. En el caso de que al hincar se tuviese complicaciones para proceder con el desarrollo, también se puede proceder al reseteo.

Ingreso de valor para la velocidad

Después de la etapa de inicialización, para cambiar el valor de velocidad se digita la tecla A, ingresando así a un ala pantalla en la cual nos da la opción de este ingreso. Procedemos al ingreso el valor recordando que la misma será expresada en RPM (se indica que los valores de trabajo sin inconvenientes mayores en este sistema podrán oscilar entre un máximo 3000 y mínimo 600). Una vez ingresado digitaremos la tecla “#” que es representativa del ENTER, y en caso de ingresar valores erróneos o no deseados se pueden eliminar con la tecla asterisco.

Cabe indicar que cuando se ingresa una velocidad mayor a 2400 RPM el sistema tiende a perder la linealidad en el control PID debido a las características propias del motor utilizado en esta aplicación.

Ingreso de las constantes para el control

Podemos cambiar el valor de las constantes Kp, Ki y Kd del control PID directamente por el modulo de ingreso digitando las teclas B, C o D respectivamente en el teclado. El valor que puede ser ingresado esta definido de tal manera que solo se puede ingresar un entero y dos decimales, igualmente como en el caso de la velocidad podemos corregir el valor ingresado con la tecla asterisco.

Prueba de encoder

Se puede comprobar el correcto funcionamiento de la lectura del sistema de sensores infrarrojos del encoder en este submenú, ingresando a través de la tecla “9”. En este menú lo único que debemos hacer es girar el disco manualmente y comprobar el cambio de valores entre 1 por 0 por cada cambio en la lectura de los sensores.

1 = los sensores tiene comunicación libre

0 = los sensores no pueden transmitir

Se puede salir de esta prueba digitando la tecla "#".

Cambio de giro

Puede realizarse el cambio de giro del motor digitando la tecla asterisco desde el menú principal; en el cual, el motor se detiene durante un segundo y realiza el cambio de giro.

Lazo Abierto

Para ingresar al proceso de lazo abierto, en el menú principal se puede digitar la tecla “1”, con la que se cambia al sistema o se regresa al original.

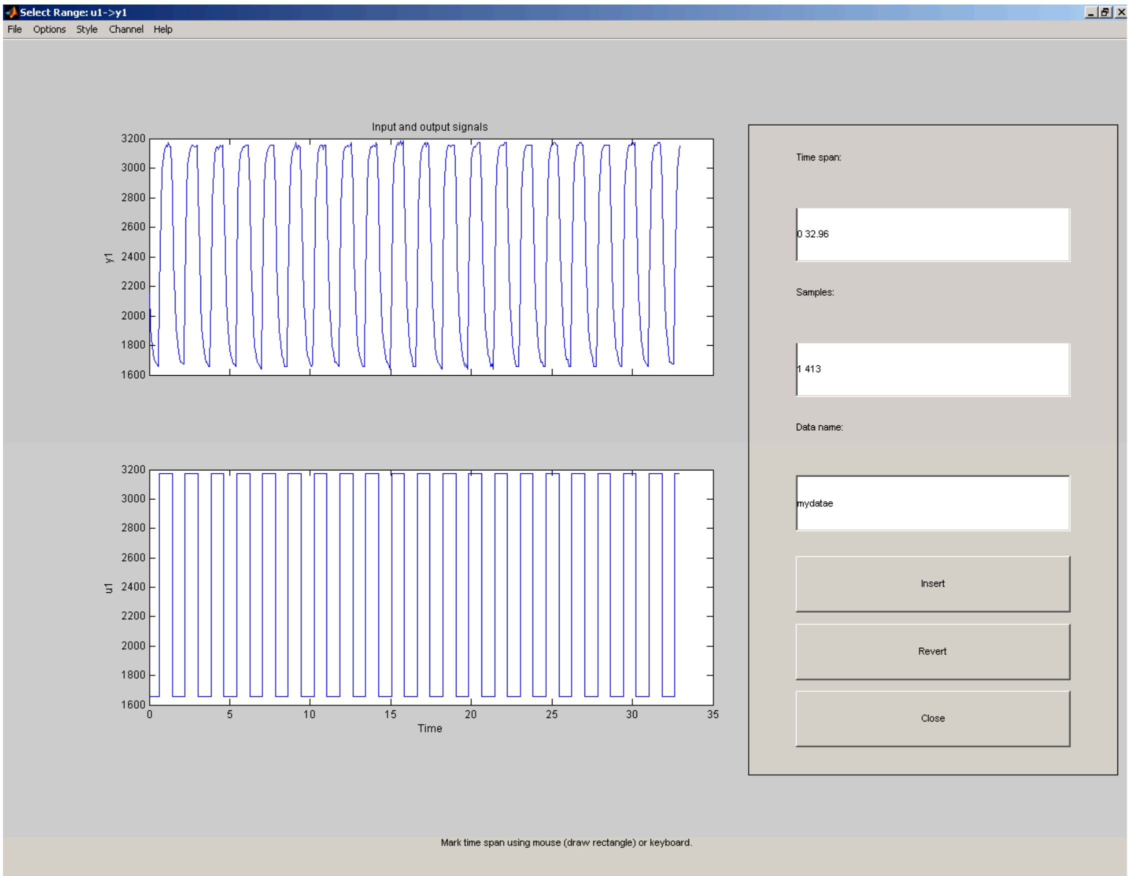
En el proceso de lazo abierto se trabaja con valores de porcentaje de región de trabajo, mas no con valores de RPM.

Identificación del sistema.

Mediante la tecla “2” en el menú principal, podemos ingresar al identificador del sistema, el cual nos ayudara a realizar la gráfica en lazo abierto entre el 20 y 40% de la región de trabajo, la misma que nos proporciona los datos para recrear el modelo del motor utilizando la plataforma de Matlab.

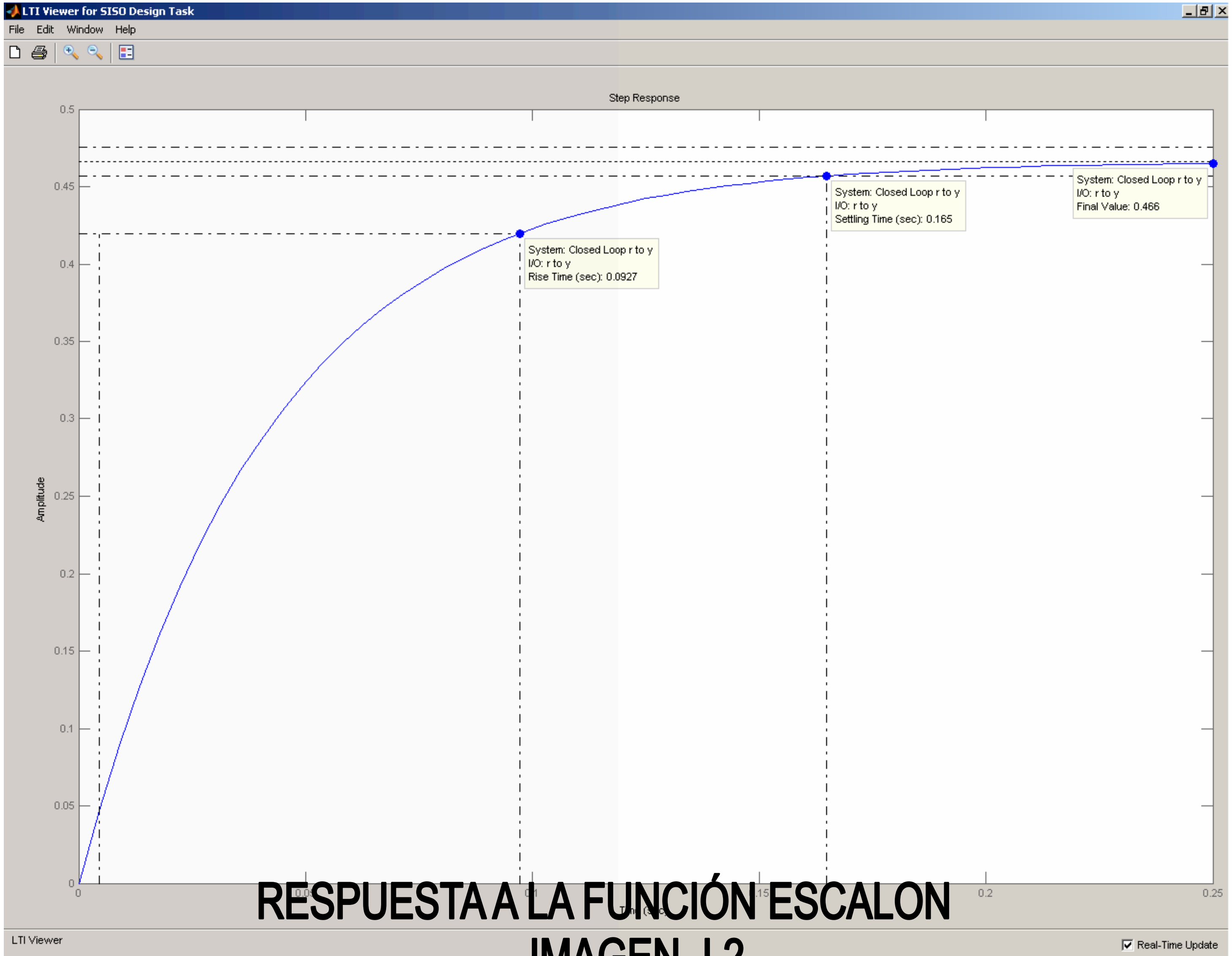
ANEXO I

SE REPRESENTA GRÁFICAMENTE EL PROCESO REALIZADO EN LA PLATAFORMA MATLAB PARA LA IDENTIFICACIÓN DEL SISTEMA, Y SU EXPLICACIÓN.



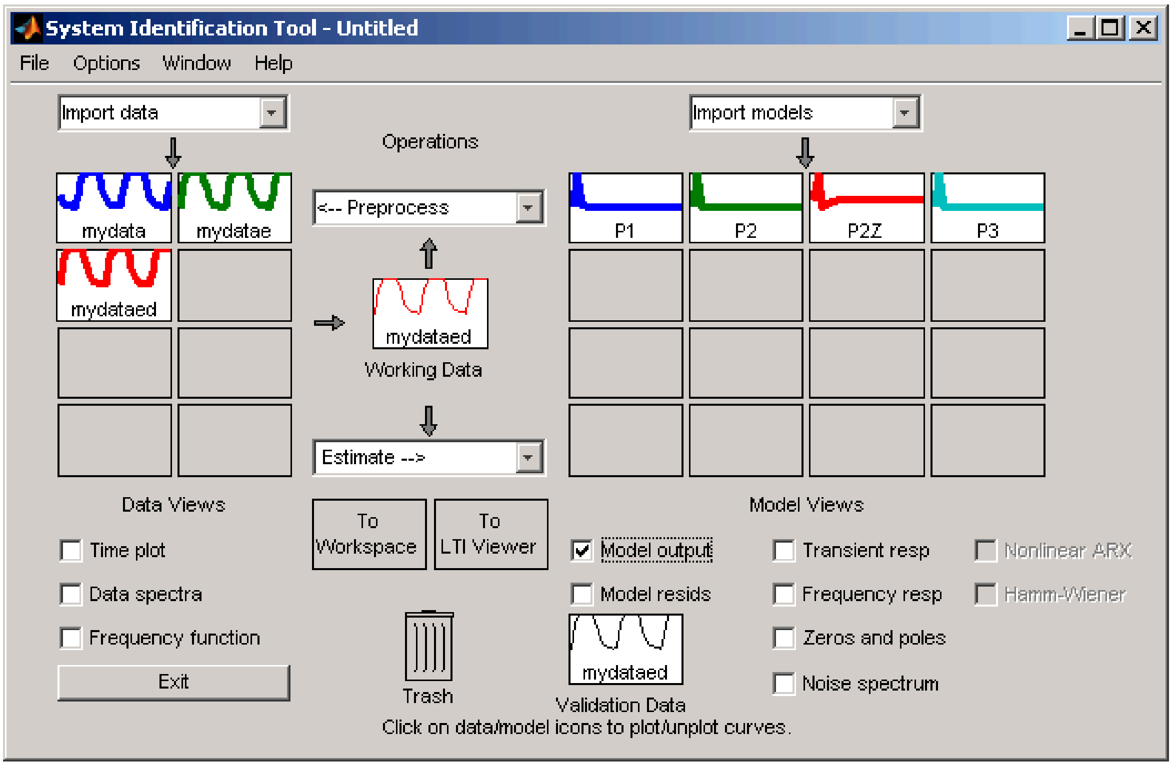
RESULTADOS QUE INGRESAN AL SISTEMA

IMAGEN I.1

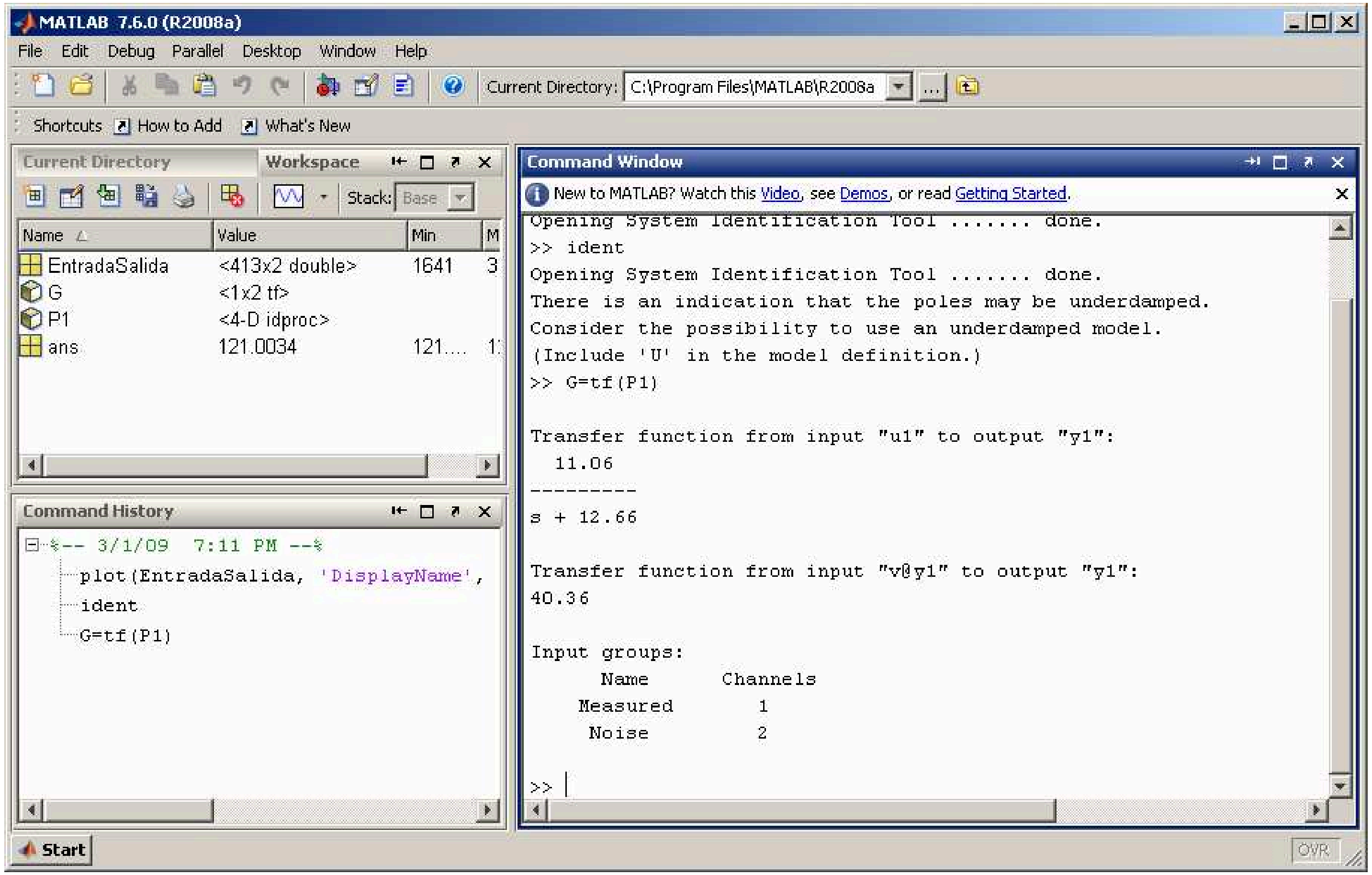


RESPUESTA A LA FUNCIÓN ESCALON

IMAGEN I.2



OBTENCION DE LAS DIFERENTES APROXIMACIONES
IMAGEN I.3

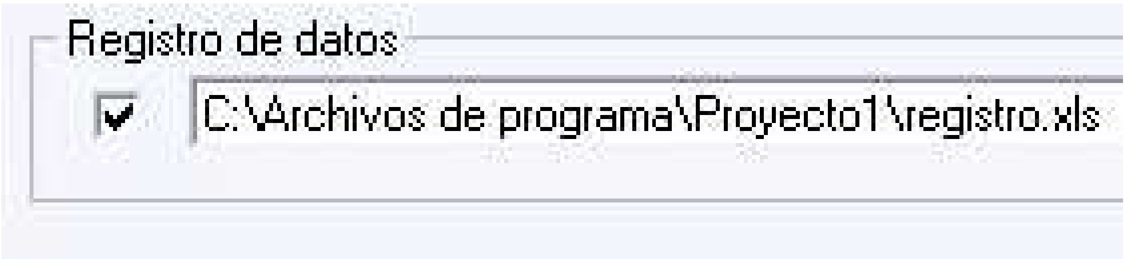


OBTENCION DE LA FUNCIÓN DE TRANSFERENCIA
IMAGEN I.4

IMASGEN I.1 RESULTADOS QUE INGRESAN AL SISTEMA

Para realizar esta prueba se trabaja en lazo abierto y con valores entre el 20 y 40% del valor máximo de velocidad del motor que estamos usando.

En el programa de Visual Basic tenemos la función de grabar los datos en un archivo de Excel, que se obtienen de las diferentes pruebas realizadas.



En este archivo contamos entonces con los valores reales que pueden ser ingresados en la Plataforma Matlab, obteniendo así la curva representativa de nuestro motor.

IMAGEN I.2 RESPUESTA A LA FUNCIÓN ESCALON

Se realiza esta prueba par ver la respuesta que tenemos cuando se escoge la función mas próxima a la de los datos reales.

IMAGEN I.3 OBTENCION DE LAS DIFERENTES APROXIMACIONES

Para poder escoger la función más próxima a la real, se realizan varias aproximaciones, y para este propósito usamos:

1. Con un polo
2. Con dos polos
3. Con dos polos y un cero
4. Con tres polos

Escogiendo finalmente la mas próxima, la de un polo.

IMAGEN I.4 OBTENCION DE LA FUNCIÓN DE TRANSFERENCIA

Obtenemos finalmente la función de transferencia:

$$tf = \frac{11.06}{s + 12.66}$$

BIBLIOGRAFIA

1. Smith, Carlos A. Corripio (1996). Control Automático de Procesos. Teoría y Práctica. Limusa Noriega Editores.

2. Ogata, Katsuhiko (1998). Ingeniería de Control Moderna. Tercera Edición. Prentice-Hall hispanoamericana, S.A.

3. Franklin, Gene. Powell, David. Emami-Naeine, Abbas (1991). Control de Sistemas Dinámicos con Retroalimentación. Addison-Wesley Iberoamericana.

4. Chen,Chi-Tsong (1993). Analog & Digital. Control System Design. Saunders College Publishing. Hartcourt Brace Jovanovich College Publishers.

5. Kuo, B.C.-“Sistemas de Control Automático”. Prentice Hall Hispanoamericana, México, 1996

6. Stephen J. Chapman.- “Maquinas Electricas”. Mc Graw Hill, Mexico, 1992

7. Ing. Mauricio Améstegui Moreno, Universidad Mayor de San Andres, La Paz – Bolivia, Enero de 2001
http://www.alumnos.usm.cl/~ignacio.morande/descargas/apuntes_de_control_pid.pdf

8. Control Automático 1, Automatización y Control Industrial, Universidad Nacional de Quilmas, Marzo 2002
http://iaci.unq.edu.ar/materias/control1/web/Apuntes/PID.pdf

9. Microchip, Hoja de datos del dsPIC30F4011, Feb 09, http://ww1.microchip.com/downloads/en/DeviceDoc/70135F.pdf.

10. Microchip, Manual de referencia de la familia dsPIC30F, Feb 09, http://ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf.

11. Microchip, Manual de usuario del Microchip PICkit 2, Feb 09, http://www.microchip.com/.

12. Definición de Wikipedia sobre control PID, Feb 09

http://es.wikipedia.org/wiki/Proporcional_integral_derivativo

13. Datasheetcatalog, Hoja de datos del LM324, Feb 09

http://www.datasheetcatalog.net/es/datasheets_pdf/L/M/3/2/LM324.shtml

14. Datasheetcatalog, Hoja de datos del 74ls14, Feb 09

http://www.datasheetcatalog.net/es/datasheets_pdf/7/4/1/4/7414.shtml

15. Datasheetcatalog, Hoja de datos del L293, Feb 09

http://www.datasheetcatalog.net/es/datasheets_pdf/L/2/9/3/L293.shtml

16. Monografias.com, Documento sobre componentes electrónicos , Feb 09

http://www.monografias.com/trabajos16/componentes-electronicos/componentes-electronicos.shtml

17. Ing. Juan del Pozo, Escuela Superior Politécnica del Litoral, Guayaquil – Ecuador, Marzo de 2009

http://200.9.176.189/web/herramientas/referencias.asp