Yoel Ocmín **Control PID Discreto de un MOTOR DC**
yoelocmin@hotmail.com

Trabajo Practico Final
PID Discreto

Objetivo.-

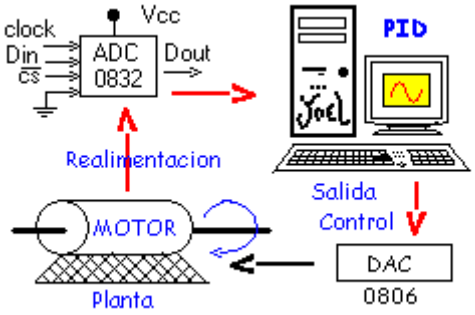
El Objetivo del Presente trabajo es poder terminar el curso de Ingeniería de Control 1, podiendo desarrollar Sistemas de control pero en el ámbito discreto.

Descripción del Control.-

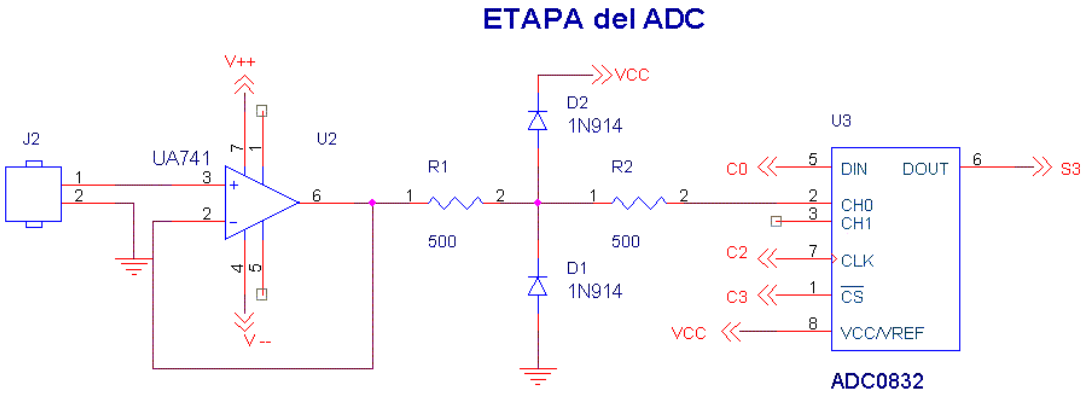
Este trabajo consiste en Discretizar el trabajo anteriormente realizado, donde se controlaba un Motor DC con los Siguientes Parámetros.

Kp= 1.7308 Kd=0.08654 Ki=133.61

Donde este motor Tiene una Entrada de Referencia de 0 – 5 V. Para eso usamos un programa desarrollado en Visual Basic. Que partiendo de un referencia nos permita controlar la velocidad revisando la salida por un ADC, para luego pasarlo a un DAC.
El Ckto tiene mas o menos el Siguiente Esquema:



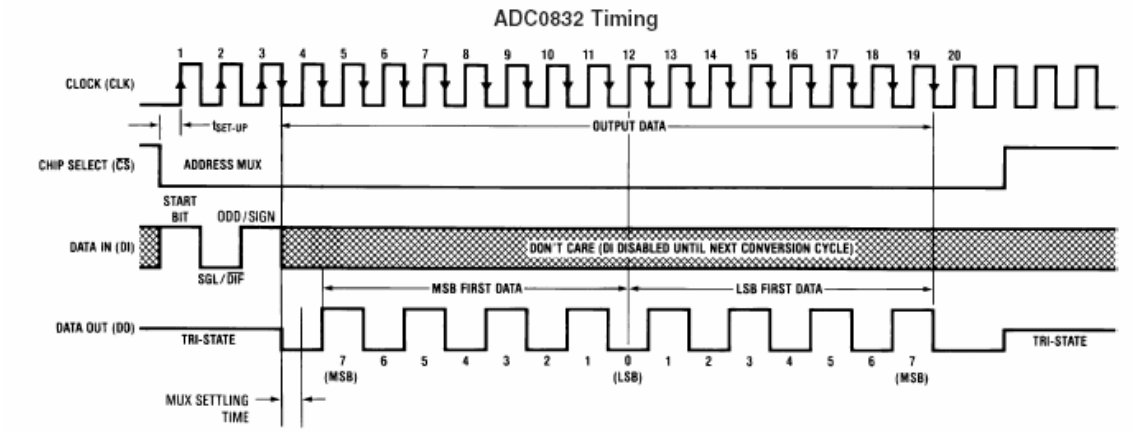
Donde la salida del Motor la pasamos a un ADC, para procesarlo en la Pc y luego dar la salida de control por el DAC.
En esta parte podemos observar que para ingresar los datos. Necesitamos de algún puerto de la Pc, yo he visto conveniente usar el puerto paralelo LPT1.
Para lo cual para no complicarme con el Hardware, he usado un ADC, serial; en este caso el ADC0832 de National instrument. Y la forma como, lo he conectado es la Siguiente:



Donde :
Din va ha ser controlado por el Pin de Control del LPT1 C0
CLK va ha ser controlado por el pin de control del LPT1 C2
CS va ha ser controlado por el pin de control del LPT1 C3

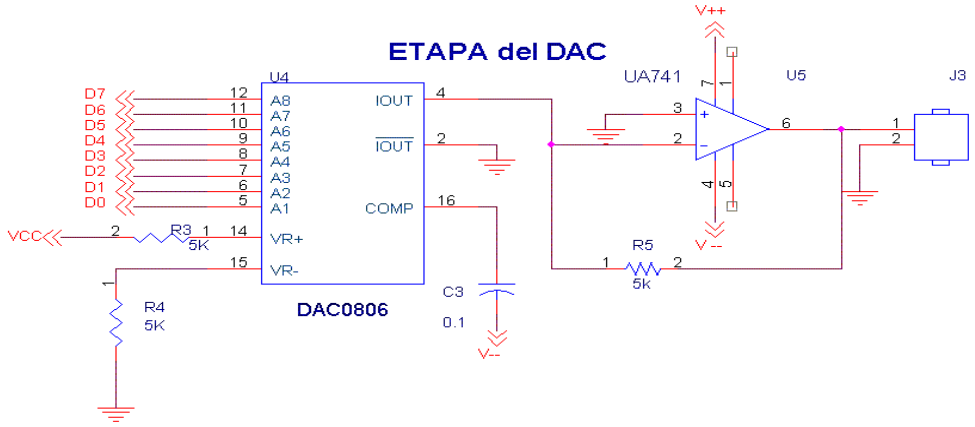
El Amplificador operacional que aparece en configuración seguidor, es para aislar la impedancia de entrada con la entrada del ADC.
Los Diodos en serie, sirven como seguridad a ala entrada del canal0 del ADC, en caso de que la referencia sea mayor que el voltaje máximo (5v)

Donde para obtener los datos del ADC debemos de seguir con el siguiente protocolo

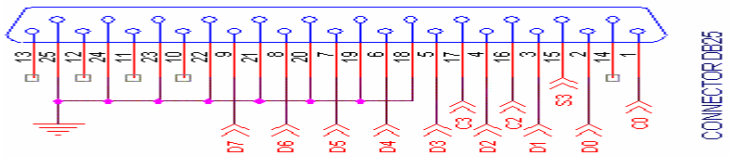


Para poder obtener estos datos, he implementado una DLL (librería de enlace dinámico), en ensamblador para así poder acceder de manera sencilla al Hardware.

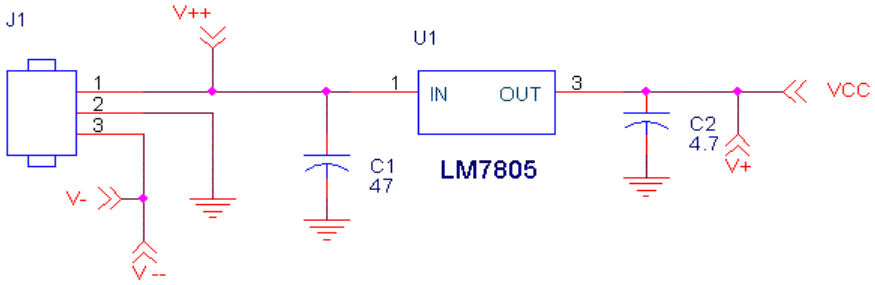
Etapa DAC.-
Esta etapa es más sencilla y consiste en que lo que entra Discretamentese transforma de Digital a Analogico.



Pero estos Ckto no van solos sino que tienen que ser acompañados de un Ckto de Conexión con la pc y otro de alimentación que nos proporcione los voltajes adecuados para nuestro DAC, ADC, y los Opam. Estos Ckto son:

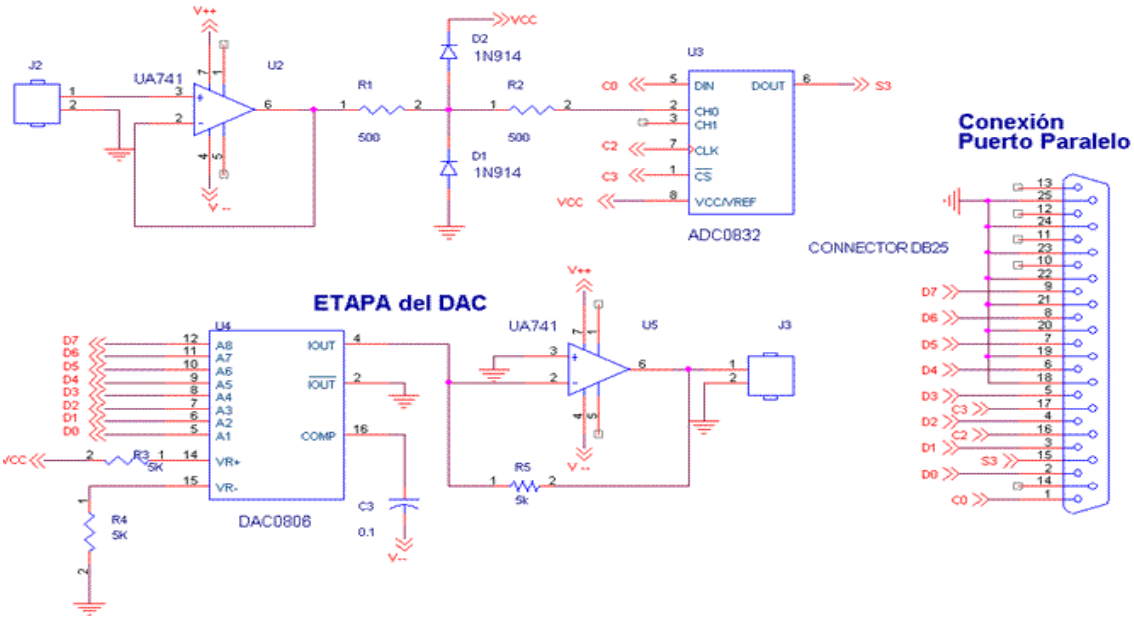


ETAPA de ALIMENTACION



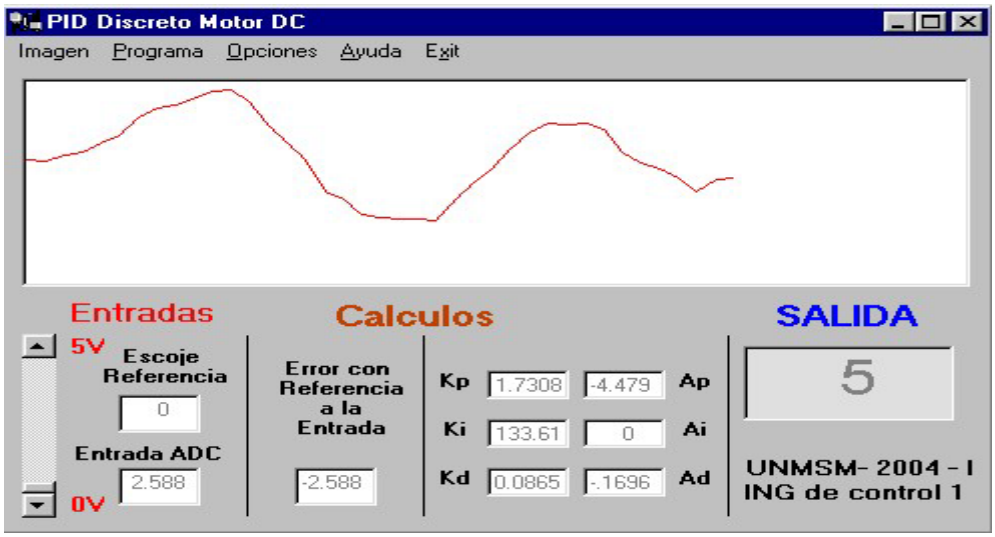
Donde el LM7805, es un Regulador de Voltaje que nos da a 5v a ala salida, y esto es preciso lo que queremos del Ckto.

ETAPA del ADC



ETAPA del DAC

PROGRAMACION DEL PID



yoelocmin@hotmail.com

Para hallar la referencia de la Barra se usa:


```
Private Sub vscRef_Change()
e(k - 2) = e(k - 1)
e(k - 1) = e(k)
e(k) = ValTomaBarra - dato(k)
txtError.Text = Str(e(k))
txtRef.Text = StrTomaBarra '
End Sub
Private Function StrTomaBarra() As String
StrTomaBarra = Str(ValTomaBarra)
End Function

Private Function ValTomaBarra() As Double
ValTomaBarra = 5 - vscRef.Value / 255 * 5
End Function
```

Ahora cada tiempo de 25ms tomamos una muestra debido a un Timer

```
Private Sub Timer1_Timer()
Dim Salida As Double
    'Inicializo Picture
    Picture1.Cls    'Borrar el gráfico
    Picture1.Line -(-10, dato(k)), &HFFFF

    dato(0) = 0 'Inicializo Dato
    e(0) = 0    'Error Inicial es 0
    e(1) = 0
                'Accion integral inicial es 0
    Ai(0) = 0
    Ai(1) = 0
                'Accion Total es 0
    Accion(0) = 0
    Accion(1) = 0

    txtError.Text = Str(e(k)) 'Error Anterior
    txtRef.Text = StrTomaBarra

For k = 2 To 52
    'Leo del Puerto Paralelo
    ADC = datillo()
    'ADC = in_ADC
    'del ADC0832, este vale de 0 - 255
    'Esto Demora Aprox 1ms
    Sleep 25 '25ms de retardo para obtener Tmuestreo
    'que sera nuestro tiempo de muestreo
    'Pero el dato vale de 0 a 5v
    'entonces lo transformamos
    dato(k) = ADC / 255 * 5
    'Coloco variable en String
    sADC = Str(dato(k))

    'Dibujo el dato(k) en el Picture
    'Dibuja una línea
```

```
    Picture1.Line -(k - 2, dato(k)), &HFF
    'Picture1.Refresh    'refresco Picture
    'Hallamos el Error con respecto a la Referencia
    e(k) = ValTomaBarra - dato(k)
    'Escojo Formula a Usar
    Tmuestreo = 20 / 1000 '50ms
    If formula = 1 Then
        Formula1
    Else
        Formula2
    End If

'Me Aseguro que la salida solo Este entre 5 y 0
If (Accion(k)) > 5 Then
Accion(k) = 5
ElseIf (Accion(k) < -5) Then
    Accion(k) = -5
End If
Salida = de5a02(Accion(k))
DAC = Salida * 51 ' * 51 ' *255/5
DAC = Inver(DAC)
Escribir &H378, DAC 'Mando Hacia el Dac 0806

txtInput.Text = sADC
txtAp.Text = Str(Ap)
txtAd.Text = Str(Ad)
txtAi.Text = Str(Ai(k))
txtOutput.Text = Str(Salida)
txtError.Text = Str(e(k))
If Detener = 1 Then
k = 52
End If
  DoEvents      'Reviso otros Eventos
Next k
End Sub
```

donde Formula1 y Formula 2 se basan en 2 Formas de hallar el PID

1.- Por Discretizacion por Partes

$E(k) = (referencia(Barra) - referencia(ADC))$:Error

$Ap(k) = Kp * E(k)$:Acción Proporcional

$Ad(k) = \frac{Kd}{T_{muestreo}}(E(k) - E(k-1))$:Acción derivativa

$Ai(k) = (E(k) * T_{muestreo}) * Ki + Ai(k-1)$:Acción Integral

$Ai(k-1) = Ai$:Anterior Ai

$Control(k) = Ap(k) + Ad(k) + Ai(k)$:Control

2.- Por Operación Derivada

$e(k) = referencia(Barra) - referencia(ADC)$: Error

$dAp = Kp * (e(k) - e(k-1))$:Diferencia Proporcional

$dAd = \frac{Kd}{T_{Muestreo}}[e(k) - 2e(k-1) + e(k-2)]$:Diferencia Derivativa

$dAi(k) = Ki * T_{Muestreo} * e(k)$:Diferencia Integral

$Accion(k) = dAp + dAd + dAi(k) + Accion(k-1)$:Acción Final

que en programación serian:

```
Private Sub Formula1()
Ap = Val(sKp) * e(k)
Ad = Val(sKd) * (e(k) - e(k - 1)) / Tmuestreo
Ai(k) = Tmuestreo * e(k) * Val(sKi) + Ai(k - 1)
Accion(k) = (Ap + Ad + Ai(k))
If Ai(k) > 5 Then 'Restringo que Ki vaya de 0 a 5v
Ai(k) = 5
ElseIf Ai(k) < 0 Then
Ai(k) = 0
End If
End Sub
```

```
Private Sub Formula2()
Ap = Val(sKp) * (e(k) - e(k - 1))
Ai(k) = Val(sKi) * Tmuestreo * e(k)
Ad = Val(sKd) * (e(k) - 2 * e(k - 1) + e(k - 2)) / Tmuestreo
Accion(k) = (Ap + Ai(k) + Ad + Accion(k - 1))
End Sub
```

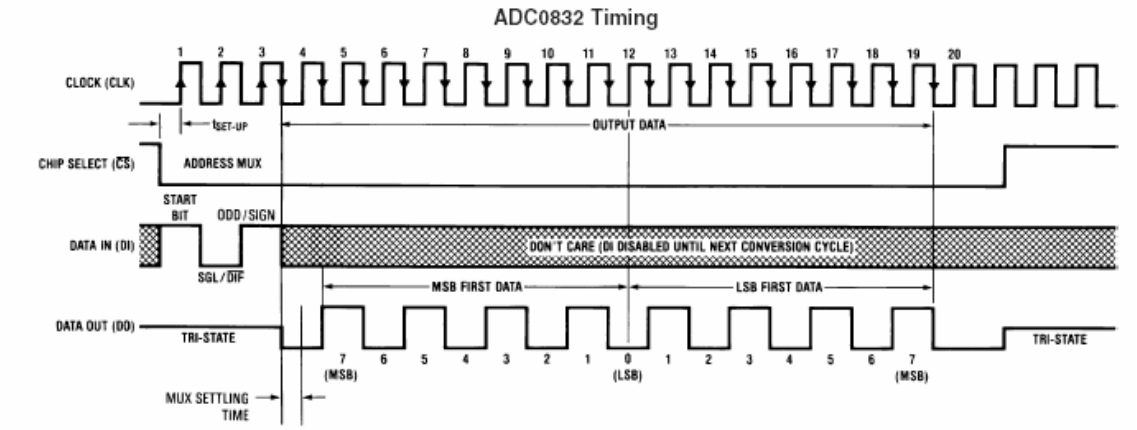
Para Acceder al DAC he usados una librería creada en Assembler, la función que devuelve el valor es:

```
datillo proc
       pusha
       _chip 0 ;Deshabilito Chip
       _clock1 ;hago un Clock: _sube, _baja

       _din 1  ;START BIT
       _chip 1 ;Habilito Chip
       _clock1       ;

       _din 1  ;SGL/~DIF
       _clock1

       _din 0 ;DOD/SIGN
       _clock1

       _clock2 ; Para la sincronización : _baja , _sube
       _clock1; en D0 (Mux Settling Time) : _sube, _baja
       mov esi,0
@datillo:
       _clock2
       _sube
       _bit
       mov byte ptr recogido[esi],al
       _baja
       inc esi
       cmp esi,8
       jnz @datillo
       call dato;'Recupero Datos
       mov esi,0
       mov byte ptr recogido, al
       _chip 0
       _clock1
       popa
       xor eax,eax
       mov al,recogido
       ret
datillo endp
```

Donde para introducir esta rutina que se encuentra dentro de la Dll inoutdac.dll
Usamos las siguientes sentencias dentro de un archivo objeto DAC.bas

```
Public Declare Function datillo Lib "inoutdac.dll" () As Byte
```

Estas rutinas se basaron en el diagrama que mostramos en un inicio:



Además aparte de esta librería hemos usado otra, llamada inpout32.dll, descargada de internet, que nos permite acceder a cualquier puerto de la Pc desde cualquier sistema operativo de windows.