PROYECTO FINAL ELECTRONICA DIGITAL II
CONTROLADOR PID ANALOGO, CON AJUSTE DIGITAL DE LOS PARAMETROS DEL CONTROLADOR

Alejandro Bañol Escobar
Universidad Tecnológica de Pereira, Risaralda, Pereira, Colombia
alejandro.@utp.edu.co

I.INTRODUCCION

EL presente proyecto tiene como objetivo aplicar un control PID análogo a un sistema de orden 3, el cual modela el comportamiento de una turbina. Para lograr esto las ganancias del controlador; ganancia proporcional (Kp), integral (Ki) y derivativa (Kd). Serán ajustadas de manera digital mediante el uso de potenciómetros digitales que a su vez serán controlados desde una interfaz gráfica de usuario(GUI). Para lograr esto, vamos hacer uso del microcontrolador JM60 el cual nos servirá como interfaz de comunicación entre la GUI y nuestro sistema, al igual como sistema de adquisición de datos.

EL controlador PID servirá para que nuestro sistema esté en la capacidad de seguir una referencia, que para nuestro ejemplo sería el equivalente a una velocidad constante a la cual se desea que gire la turbina. Dependiendo del valor que adopten los potenciómetros, cada una de las ganancias del controlador va a variar y por ende la respuesta de nuestro sistema ante determinada referencia también va a cambiar. Ver figura 1.

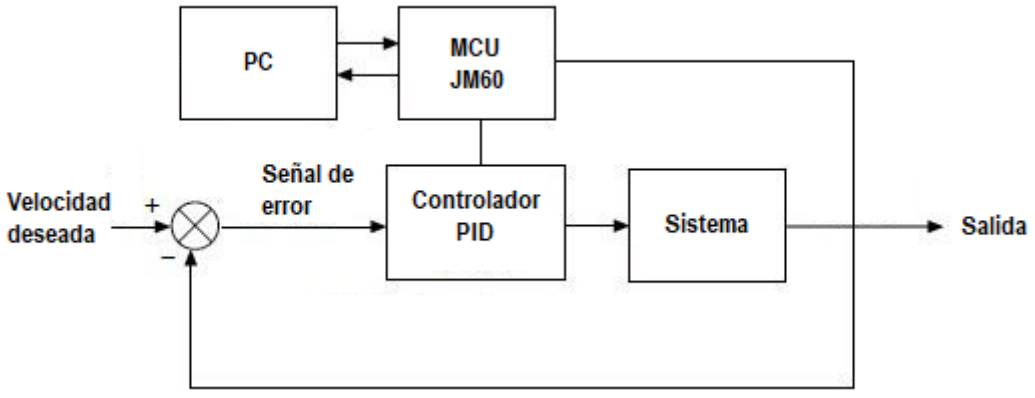


Figura 1. Diagrama de bloques del proyecto.

II.CONTENIDO

Para dar inicio al proyecto se diseñó e implemento, el circuito que simula el comportamiento de nuestro sistema, mediante amplificadores operacionales. El cual está definido por la siguiente función de transferencia de orden 3.

$$H_{(s)} = \frac{\frac{k}{r}w^2}{s^3 + \frac{(2zwr+1)s^2}{r} + \frac{(rw^2+2zw)s}{r} + \frac{w^2}{r}}$$

A continuación, se muestra el esquemático del circuito.

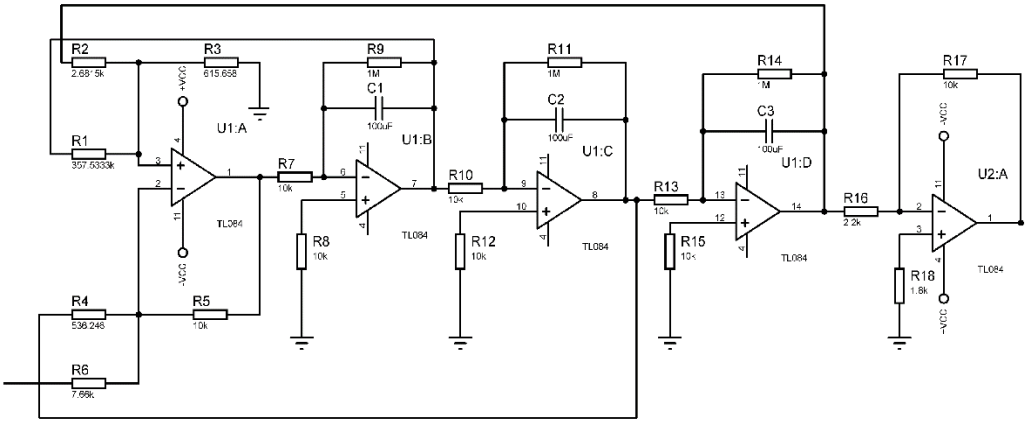


Figura 2. Circuito que modela el sistema.

Después se procedió a obtener la gráfica de la respuesta natural del sistema. En la figura 3 se puede observar la respuesta obtenida mediante el software de simulación Proteus. En la figura 4 se puede observar el resultado obtenido mediante las pruebas experimentales realizadas, la cual se obtuvo gracias a un osciloscopio.

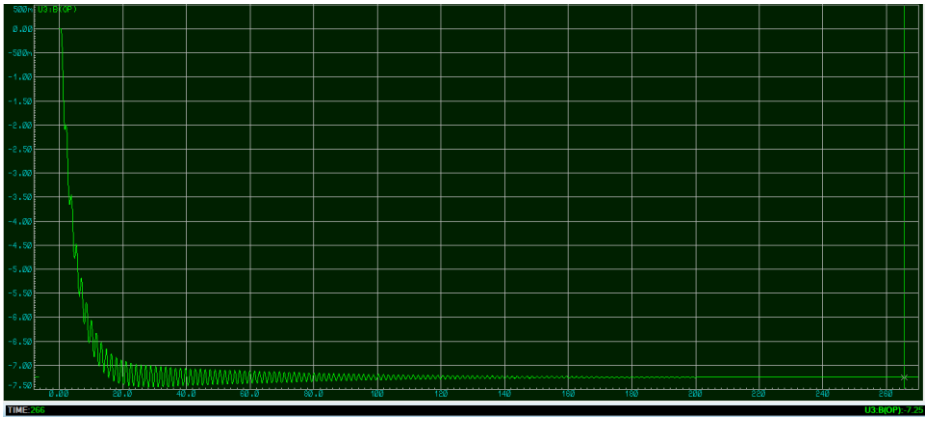


Figura 3. Respuesta al escalón del sistema.

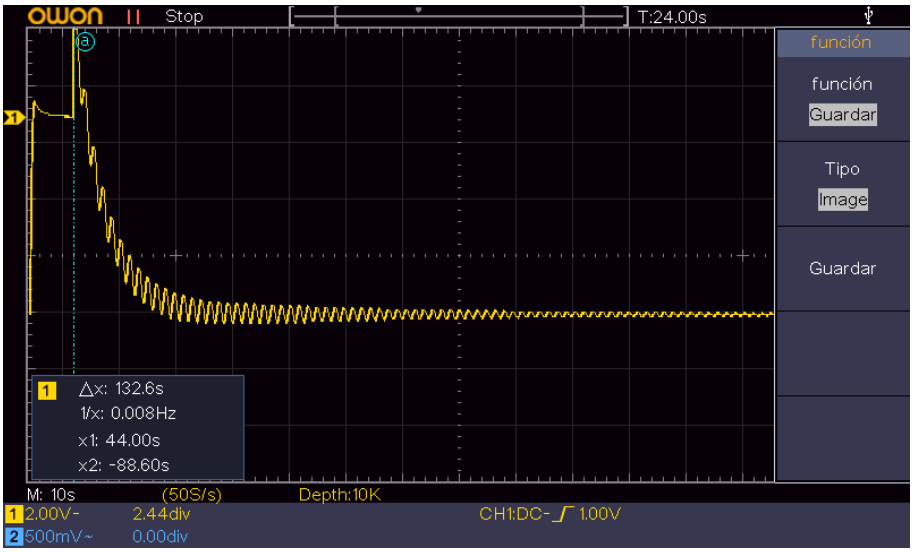


Figura 4. Respuesta al escalón del sistema.

Después se procedió a implementar el controlador PID análogo. A la entrada del circuito de la figura 2. Como se puede observar en la figura 5.

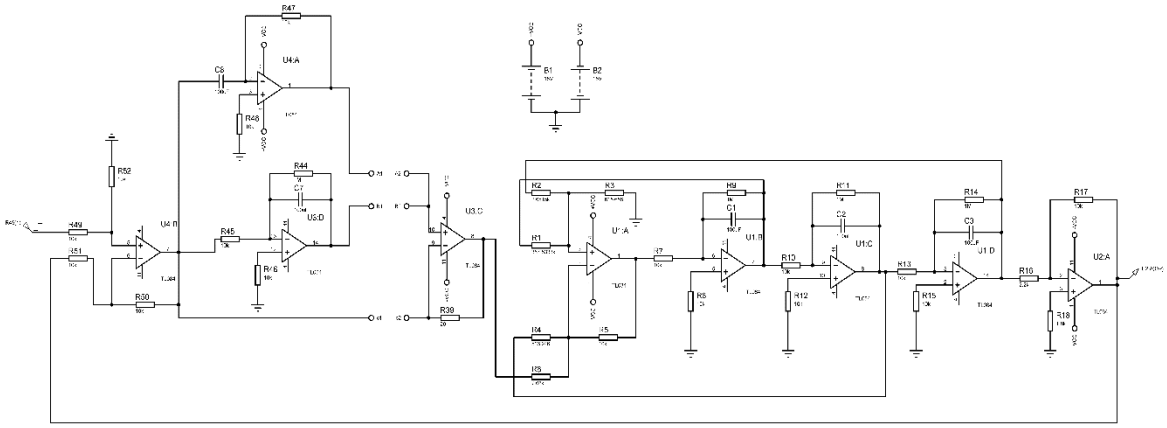


Figura 5. Controlador PID conectado al sistema.

Como se puede observar en la figura 5, tenemos 6 terminales donde a cada par le corresponde un potenciómetro; es decir. En las terminales A1 y A2 estará conectado el potenciómetro que me ajusta la ganancia derivativa (Kd), después tenemos las terminales B1 y B2 donde estará conectado el potenciómetro que me ajusta la ganancia integral (Ki) y por ultimo tenemos las terminales c1 y c2 donde estará conectado el potenciómetro que me ajusta la ganancia proporcional (Kp). A continuación, se puede observar con mejor detalle la conexión de cada potenciómetro digital a sus correspondientes terminales, ver figura 6. Donde también se especifican a que pines del microcontrolador están conectados los demás pines restantes del potenciómetro digital. Para el proyecto se hizo uso de 3 potenciómetros digitales de 10K de referencia X9C103.

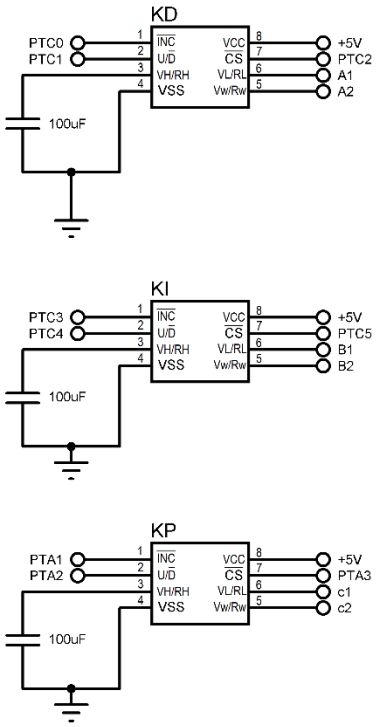
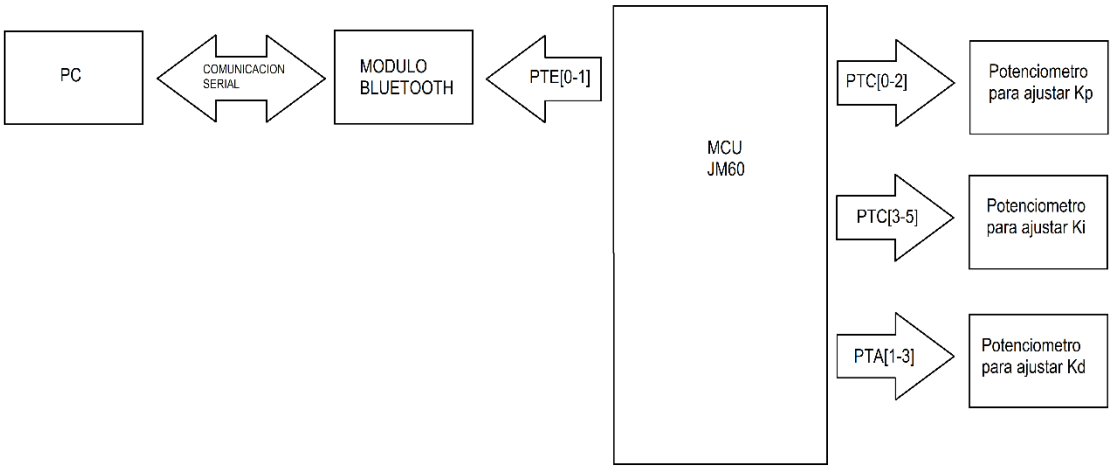


Figura 6. Conexiones de cada potenciómetro al circuito.

DIAGRAMA DE BLOQUES



PRUEBAS REALIZADAS PARA EVALUAR EL RENDIMIENTO DEL CONTROLADOR PID

Previamente antes de añadir los potenciómetros digitales y el microcontrolador, a nuestro controlador PID se realizó una serie de pruebas para poder evaluar el rendimiento de este y que efectivamente si estuviese en la capacidad de llevar la respuesta de nuestro sistema hacia la referencia deseada en este caso la velocidad a la cual se desea que gire la turbina. Para las pruebas se estableció un escalón unitario a 5V, a continuación, se presenta los resultados obtenidos con la simulación y las pruebas experimentales.

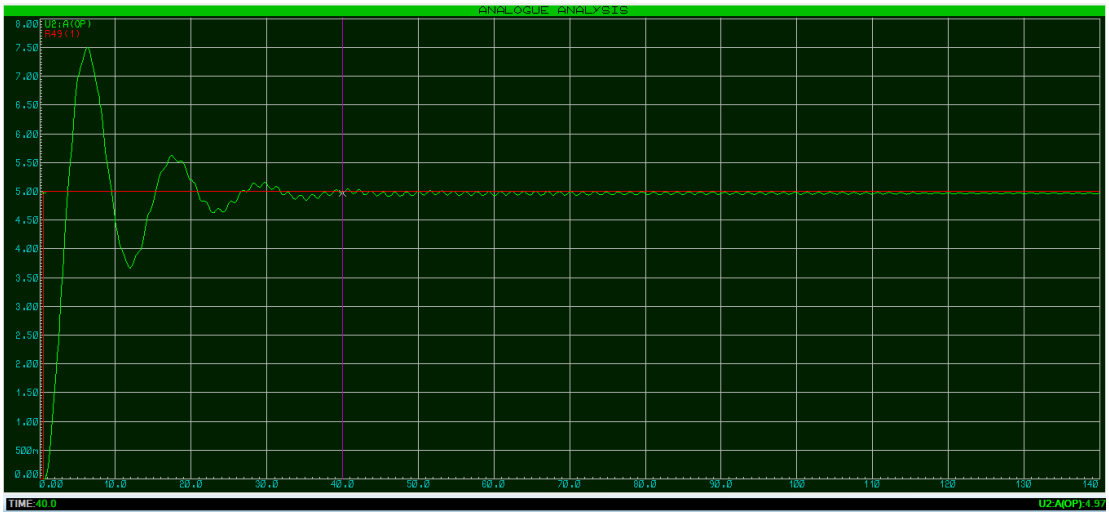


Figura 7. Escalón unitario (rojo), respuesta del sistema (verde).

Como se puede observar en la figura 7, al figar un cursor (línea de color morado) ese punto el sistema ya ha alcanzado el estado estable, lo cual demora aproximadamente 40 segundos, tal como se puede observar en la parte inferior izquierda. Como también para ese punto al haber alcanzado el estado estable ya el sistema se aproxima a la referencia, en este caso 5V, lo anterior se puede comprobar en la parte inferior derecha donde tenemos un valor correspondiente de 4.97V.

A continuación, se presentan los resultados obtenidos con el osciloscopio, ver figura 8, donde se puede observar que el sistema si alcanza la referencia de 5V, al ubicar unos cursores que me ayudan a obtener el voltaje, lo cual se puede observar en la parte inferior izquierda, para esta captura se manejó una escala de 2V por división con una escala en tiempo de 50 seg por división.

Figura 8. Respuesta al escalón unitario del sistema.

CONFIGURACION Y OPERACIÓN DE LOS POTENCIOMETROS DIGITALES

Como se había mencionado previamente para el proyecto se hizo uso de tres potenciómetros digitales de 10K de referencia X9C103. Para determinar el modo de operación de estos dispositivos y como se lograba el incremento o decremento del valor de la resistencia de cada potenciómetro; se consultó la hoja de datos proporcionada por el fabricante, donde se encontró una tabla de la verdad que nos ayudaría con esta tarea. Ver figura 9.

MODE SELECTION

$\overline{CS}$	$\overline{INC}$	U/$\overline{D}$	Mode
L	↘	H	Wiper Up
L	↘	L	Wiper Down
↗	H	X	Store Wiper Position
H	X	X	Standby Current
↗	L	X	No Store, Return to Standby

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from Low to High	Will change from Low to High
	May change from High to Low	Will change from High to Low
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

Figura 9. Tabla de la verdad para modificar el valor de los potenciómetros digitales, obtenida de la hoja de datos del fabricante.

Con lo anterior se sabía cuál debían ser los estados de los pines CS, INC y U/D de los potenciómetros digitales si se quería incrementar o decrementar el valor de cada potenciómetro. Después se procedió a implementar una función en Code warrior que me ayudara a cumplir con esta tarea, dado que el cambio en cada potenciómetro es en pasos de 100 Ω; dicha función me recibe cual potenciómetro deseo modificar (variable potenciómetro), el tamaño de cada paso el cual es constante que equivale a 100 (variable valor) y recibe un valor entre 1 y 0 (variable estado) que le indica que incremente si estado=1 o en el caso contrario decremento cuando estado=0.

```
void ajuste (int potenciometro,int valor,int estado) {
    int conversion;
    int i=0;
    conversion =  ((valor)*(100))/10000;
    switch (potenciometro){
    case 1: // SI SE SELCCIONA EL PRIMER POTENCIOMETRO


        if (estado==1)  //INCREMENTO
        {
            for(i=0;i<conversion;i++){
                ud=1;
                in =1;
                retardo();
                in=0;
                retardo();
                in=1;
            }
        }
        else{           //DECREMENTO


            for(i=0;i<conversion;i++){
                ud=0;
                in =1;
                retardo();
                in=0;
                retardo();
                in=1;
            }
        }
```

Figura 10. Función implementada para modificar el valor de cada potenciómetro.

Después se procedió a verificar si efectivamente si se estaba modificando el valor de los potenciómetros mediante el osciloscopio, dado que en la figura 9, en la sección WAVEFORM que en español traduce “forma de onda” nos muestra como debe ser la forma de onda cuando se está modificando el valor del potenciómetro.

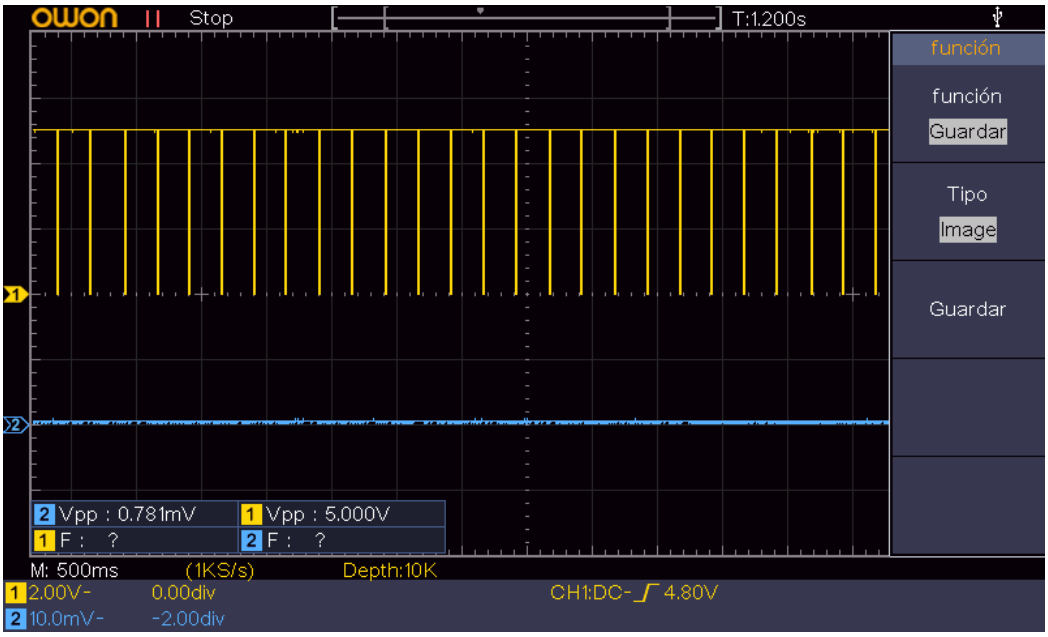


Figura 11. Forma de la señal respecto a una de las entradas (CS, INC, U/D) cuando está siendo modificada.

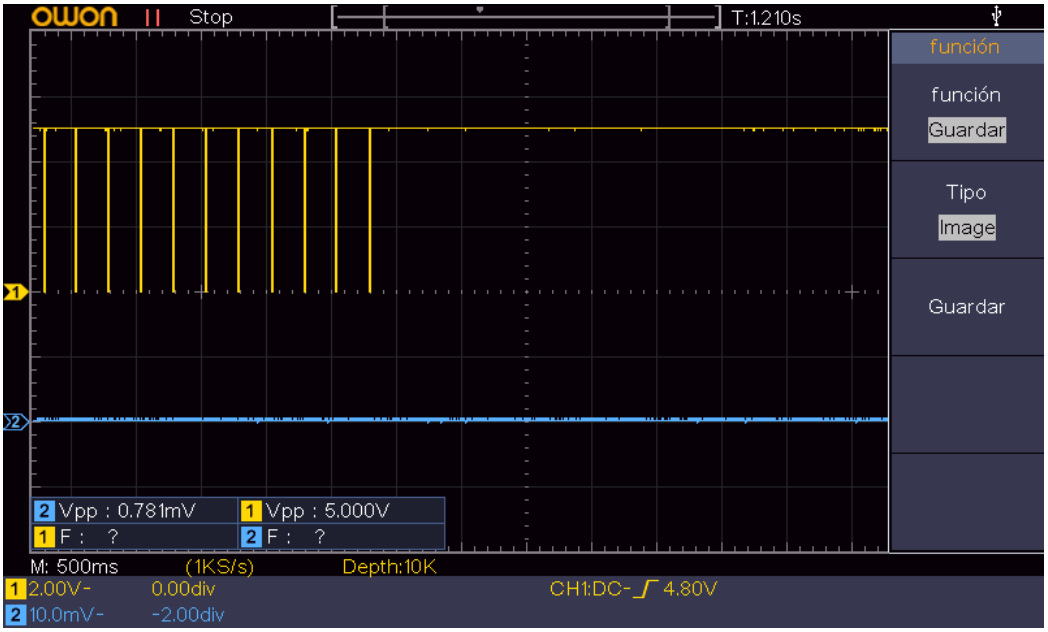


Figura 12. Forma de la señal respecto a una de las entradas (CS, INC, U/D) cuando finaliza el proceso de modificación.

Para modificar el valor de los potenciómetros se implementó una GUI en Matlab, como se puede observar en la figura 13, la cual me enviaba una serie de caracteres al microcontrolador, después de ingresar un determinado valor y pulsar el botón de enviar. La comunicación entre el microcontrolador y el pc se estableció mediante comunicación serial haciendo uso de un módulo bluetooht HC-05.

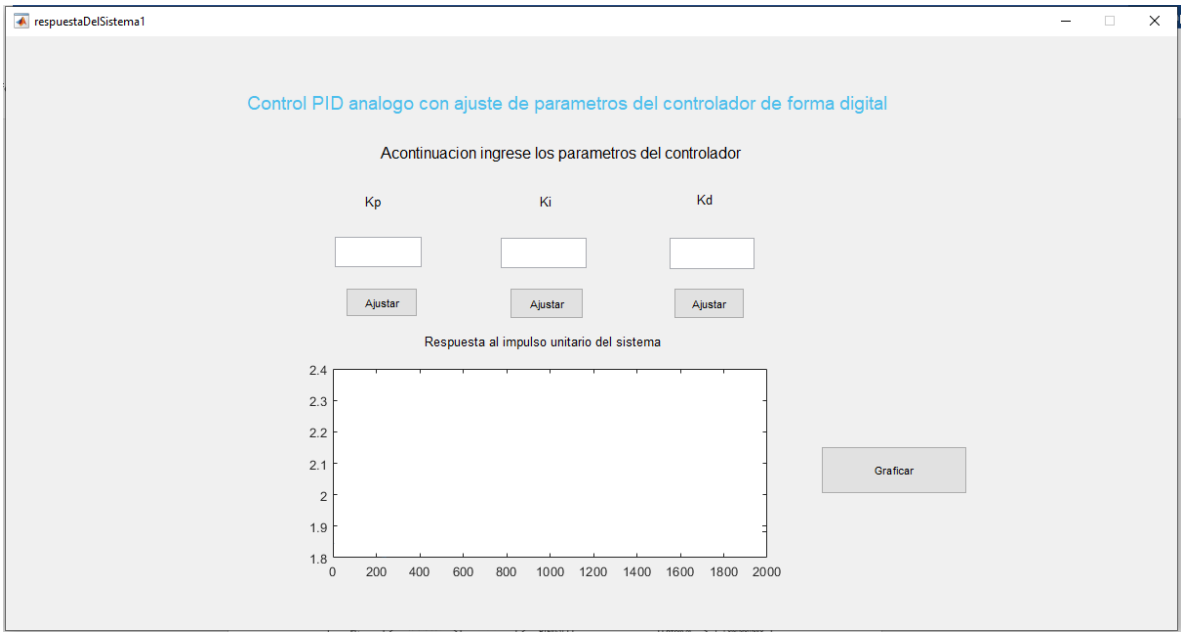


Figura 13. GUI en Matlab.

Después se procedió a realizar diferentes pruebas, variando para diferentes valores de resistencia y se obtuvo la respuesta del sistema.

Kp	Ki	Kd
400	5000	200

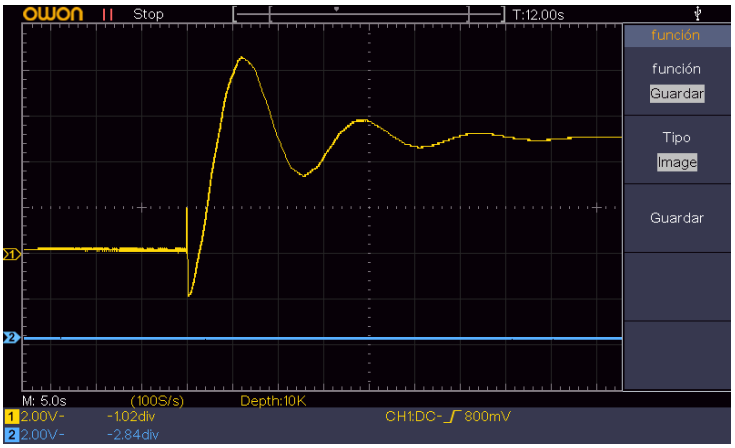


Figura 14. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
8000	8000	8000

Figura 15. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
2000	5000	400

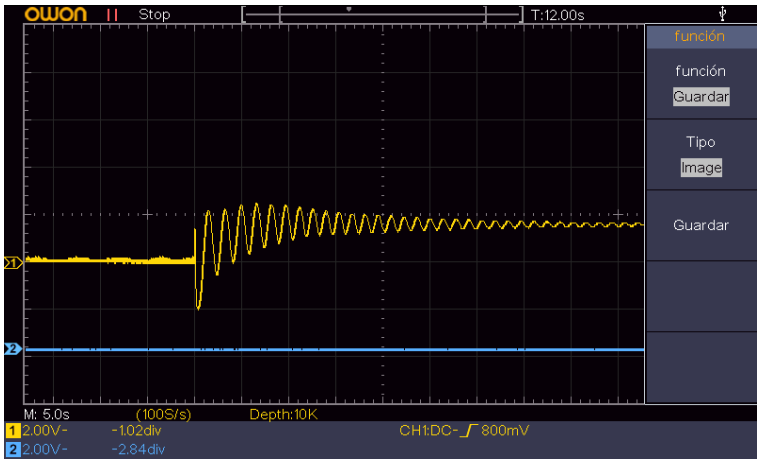


Figura 16. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
400	400	400

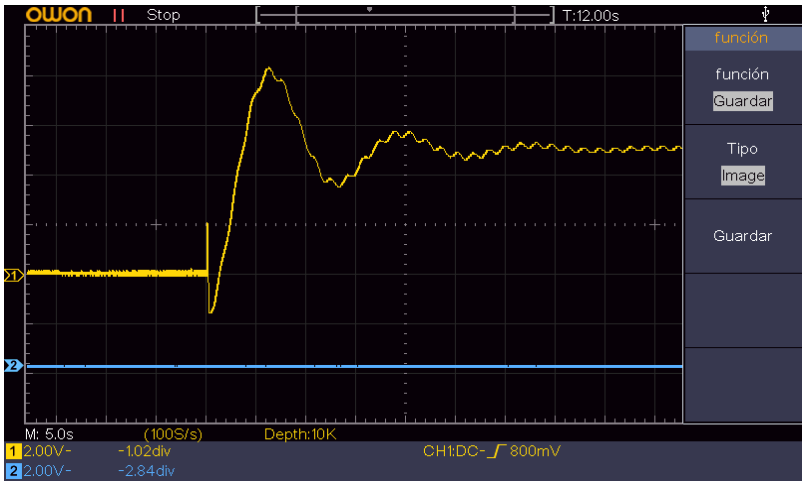


Figura 17. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
100	9000	800

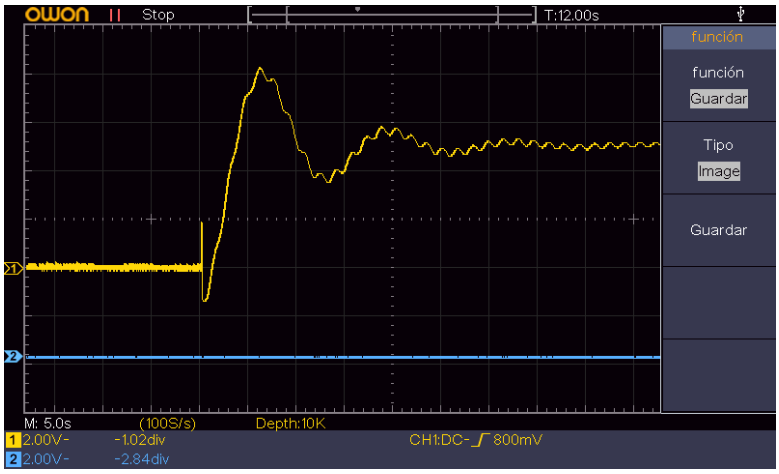


Figura 18. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
100	100	8000

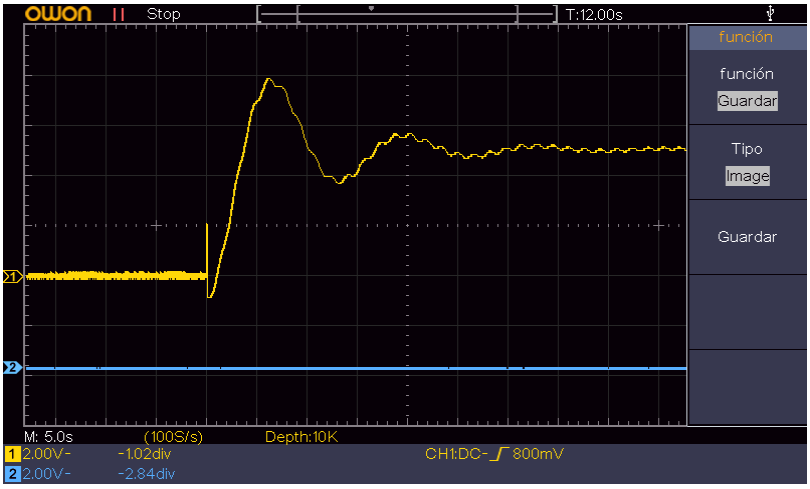


Figura 19. Respuesta al escalón unitario del sistema.

Kp	Ki	Kd
200	6000	1000

Figura 20. Respuesta al escalón unitario del sistema.

CONVERTIDOR ANALOGO DIGITAL (ADC) Y ENVIO DE INFORMACION PARA SU POSTERIOR GRAFICACION EN LA GUI

Para efectos de evitar daños en la ADC con tensiones superiores a 5V dado los sobre impulsos que tiene el sistema ante un escalón unitario, se configuro una señal de referencia de 2.5V al poner un divisor de tensión que como señal de entrada me recibía 5V proveniente de un pin del microcontrolador; pero a la salida me suministraba 2.5V aproximadamente la cual sería nuestro escalón unitario. Después se procedió a establecer ciertos valores para los parámetros del controlador en la GUI. Seguidamente se dio clic en el botón graficar y se obtuvo la gráfica correspondiente a una porción de la respuesta de nuestro sistema.

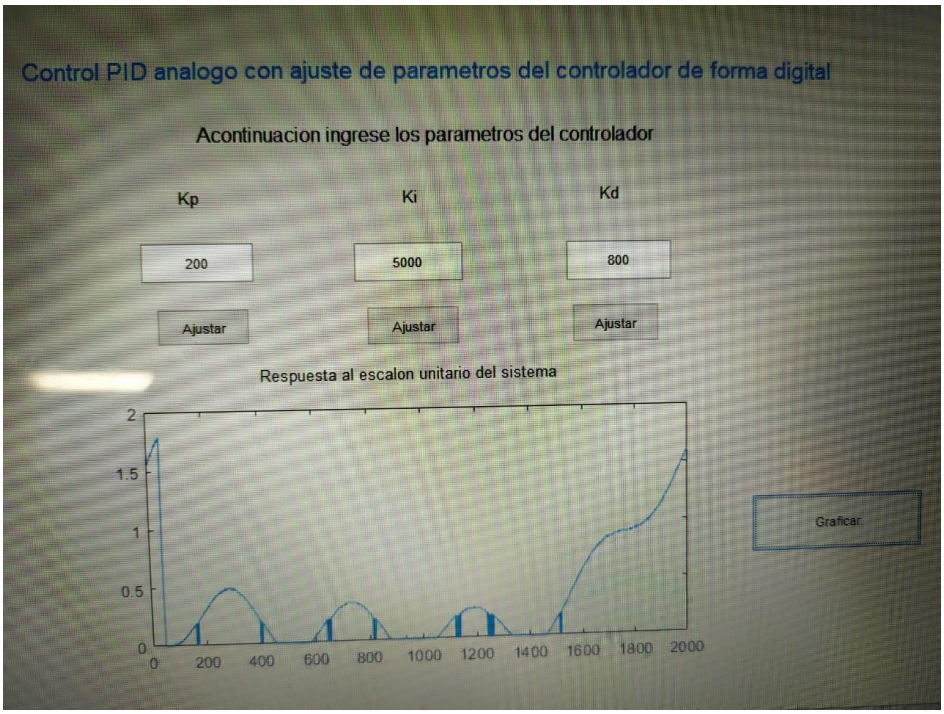


Figura 21. Grafica de la respuesta del sistema obtenida gracias a la ADC.

A continuación, se presenta la respuesta del sistema para los parámetros que se establecieron en la figura 21, mediante la ayuda de un osciloscopio con el objetivo de realizar una comparación entre la gráfica de la GUI que se construyó gracias a la información proporcionada por la ADC y un instrumento de medida.

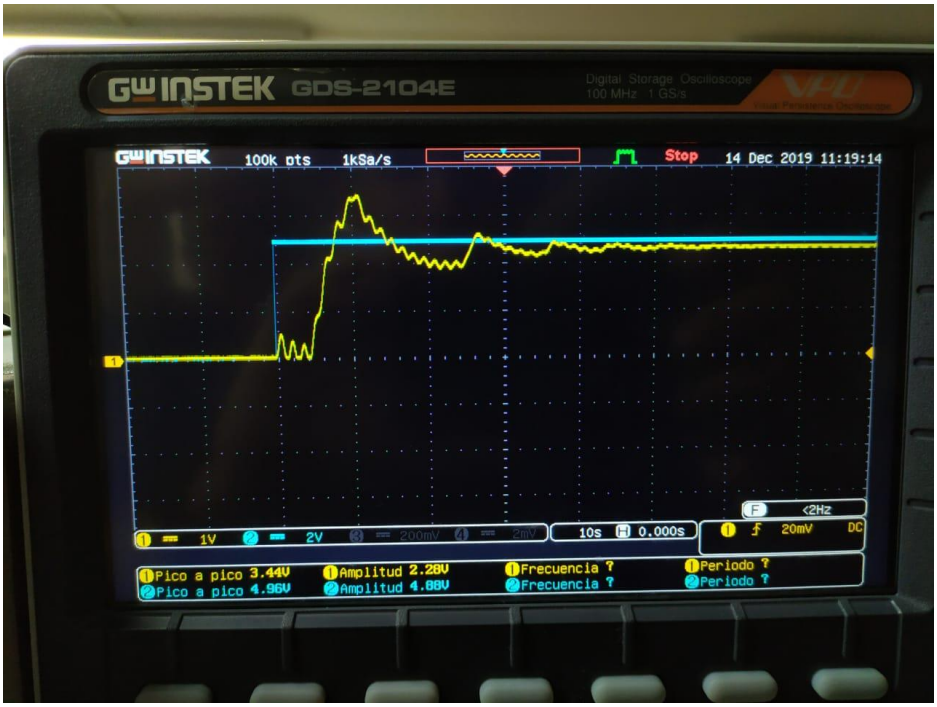


Figura 22. Grafica de la respuesta del sistema obtenida gracias al osciloscopio.

Como se puede observar en la figura 22 para el canal 1 que corresponde a la gráfica de color amarillo se manejó una escala de 1V por división y para la gráfica azul que corresponde al canal 2 se manejó una escala de 2V por división.

La grafica azul corresponde a los 5V que llegan al divisor de tensión que me establece mi valor de referencia en este caso 2.5 ya que se construyó con resistencias de 1K. La grafica amarilla corresponde a la respuesta del sistema que evidentemente dada la escala tiende a un valor aproximado de 2.4V.

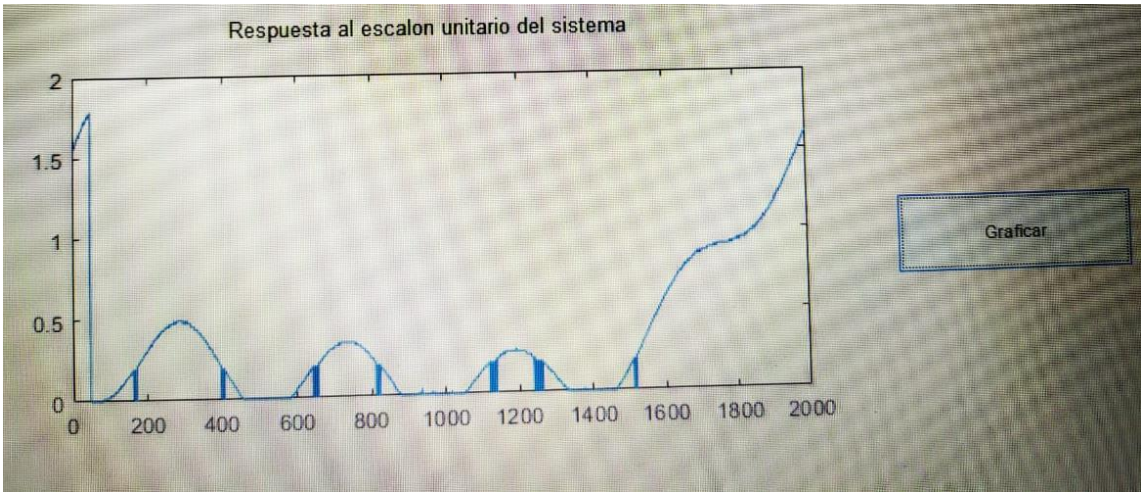


Figura 23. Vista más detallada de la respuesta obtenida mediante la ADC y la GUI.

CONSIDERACIONES DE DISEÑO

Un aspecto importante para tener en cuenta en este proyecto, fue las medidas de protección que se implementaron para no dañar el convertidor análogo digital del microcontrolador, dado que este no admite una tensión superior a 5V. Porque al momento de poner un escalón unitario a nuestro sistema este por lo general tenía un sobre impulso de 2 voltios por encima del valor de referencia, lo cual podría causar un daño a nuestro microcontrolador y también donde la respuesta de nuestro sistema es muy lenta. Para brindar solución a este problema se puso un amplificador en modo seguidor el cual se conectó a la entrada de la ADC el cual esta polarizado con 5V, que me garantiza que para salidas superiores a 5V este se saturara y de esta manera puedo proteger el convertidor análogo digital del microcontrolador.

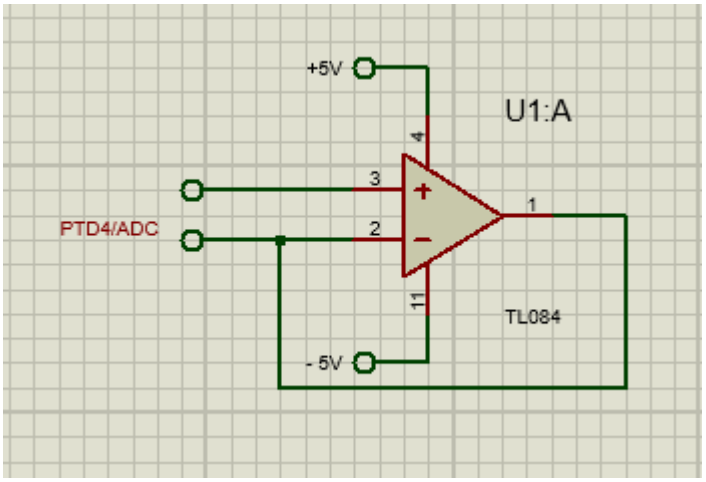


Figura 24. Circuito de protección para la ADC del microcontrolador.

CODIGO DEL PROYECTO EN CODE WARRIOR

```
//* PLANTILLA BOOTLOADER PARA EL MCU JM60                              *
//*********************************************************************/
//Alejandro bañol escobar 1088353182.
#include <hidef.h>      /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */


const byte NVOPT_INIT @0x0000FFBF = 0x02;    // vector redirect, flash
unsecure
const byte NVPROT_INIT @0x0000FFBD = 0xFA;    // 0xFC00-0xFFFF are
protected
```

```
#define in PTCD_PTCD0 //SALIDA POTENCIOMETRO 1 // POTENCIOMETRO DEL KP //
PUERTO C
#define ud PTCD_PTCD1 //SALIDA POTENCIOMETRO 1
#define cs PTCD_PTCD2 //SALIDA POTENCIOMETRO 1

#define in1 PTCD_PTCD3 //SALIDA POTENCIOMETRO 2 // POTENCIOMETRO DEL KI
// PUERTO C
#define ud1 PTCD_PTCD4 //SALIDA POTENCIOMETRO 2
#define cs1 PTCD_PTCD5 //SALIDA POTENCIOMETRO 2

#define in2 PTAD_PTAD1 //SALIDA POTENCIOMETRO 3 // POTENCIOMETRO DEL KD
// PUERTO A
#define ud2 PTAD_PTAD2 //SALIDA POTENCIOMETRO 3
#define cs2 PTAD_PTAD3 //SALIDA POTENCIOMETRO 3



extern void _Startup(void);

unsigned char K1@0xB0, K2@0xB1, CONT1@0xB2;

//FUNCION DE RETARDO1.
void retardo(void);
//FUNCION DE RETARDO2.
void retardo2(void);
void ajuste (int potenciometro,int valor,int estado);

char FLAG=0;

//Función para la interrupción por recepción serial (Espera caracter de
solicitud)
interrupt 20 void int_rec(void) {
      //Variable auxiliar
      char AUX;
      long int i=0;
      //Reconoce recepción
      AUX=SCI1S1;
      //Valida si es el caracter de solicitud y si hay solicitudes
pendientes
      if (SCI1D=='a'){ //[a]=potenciometro 1,pasos de 100, incremento.
            PTAD_PTAD0=0;
            ajuste(1,100,1); // se deja como constante el valor de 100,
dado que para el incremento de 100 en cieen
            // en el potenciometro 1, se va hacer enviando varias veces
el caracter a desde matlab

            //PTAD_PTAD0=0;
            //PTAD_PTAD1=1;//aqui encendia el led rojo para validar si,
si estaba recibiendo el caracter a.
      }

      if (SCI1D=='b'){  //[b]=potenciometro 1,pasos de 100, decremento.
            PTAD_PTAD0=0;
            ajuste(1,100,0);
```

```
            //PTAD_PTAD0=1;
            //PTAD_PTAD1=0; //aqui apagaba el led para validar si, no
estaba recibiendo el caracter a.
      }

      // ajuste con el potenciometro 2
      if (SCI1D=='c') { //incremento potenciometro 2
            PTAD_PTAD0=0;
            ajuste(2,100,1);
      }

      if (SCI1D=='d') { //decremento potenciometro 2
            PTAD_PTAD0=0;
            ajuste(2,100,0);
      }


      //// ajuste con el potenciometro 3
      if (SCI1D=='e') { //incremento potenciometro 3
            PTAD_PTAD0=0;
            ajuste(3,100,1);
      }

      if (SCI1D=='f') { //decremento potenciometro 3
            PTAD_PTAD0=0;
            ajuste(3,100,0);
      }

      if (SCI1D=='A'){ //RECIVE LA SOLICITUD PARA ENVIAR LA INFORMACION Y
GRAFICAR
            PTAD_PTAD0=1; //ENVIA ESCALON UNITARIO 5V
            for(i=0;i<2000;i++){
                  ADCSC1_ADCO=0;
                  while (ADCSC1_COCO==0);
                  SCI1D=ADCRL;
                  while(SCI1S1_TC==0);
                  SCI1D=0x0A;
                  while(SCI1S1_TC==0);
                  SCI1D=0x0D;
                  while(SCI1S1_TC==0);
                  //retardo2();
            }
      }
}
```

```
void main(void) {


      //CONFIGURACION HARDWARE DEL MCU.
      SOPT1 = 0x20; //DESACTIVA COP
      PTCDD=0xFF; //PUERTO C COMO SALIDA TODOS LOS BITS. (1 = salida)
      PTCDS=0x01;
      PTADD=0xFF; //PUERTO A COMO SALIDA PARA CONECTAR EL DIODO LED Y EN
ESTE CASO SE CONECTA AL BIT PTA0. (EN CONFIGURACIONSALIDA=1)
      PTADS=0xFF; //MODO ALTA CORRIENTE.

      PTDDD=0x00; //PUERTO D COMO ENTRADA TODOS LOS BITS.

      //Configura el módulo de comunicación serial 1 (SCI1)
      SCI1C1=0x00; //Tamaño del dato=8 bits y sin paridad
      SCI1C2=0x2C; //Habilita interrupción por recepción serial
      SCI1C3=0x00; //No habilita demás fuentes de interrupción
      SCI1BD=0x9C; //Velocidad de comunicación=9600bps
      EnableInterrupts;
      //con lo anterior configure el puerto E0(Tx) y E1(Rx) para la
      comuncacion serial
      //LAZO PRINCIPAL

      //FLAG=0;

      //PTAD_PTAD0=0;// CON ESTO ME ASEGURO QUE EL LED ESTE APAGADO AL
ENERGIZAR EL MCU
      //CONVERDITOR ANALOGO DIGITAL (ADC)
      ADCCFG=0xF1;
      ADCSC2=0x00;
      ADCSC1=0x0B;
      APCTL2_ADPC11=1; //Habilita canal 11 del ADC

      cs=0;
      cs1=0;
      cs2=0;

      for (;;) {



      }
}
```

```
void retardo(void) { //funcion de retardo de 50ms
      asm {
            clr   CONT1
            LAZO:   mov   #200,K1   ; Inicio subrutina de retardo
            LAZO1:  mov   #170,K2
            LAZO2:  dbnz  K2,LAZO2
            dbnz  K1,LAZO1  ; Fin subrutina de retardo
            inc   CONT1      ; Código para controlar las
            lda   CONT1      ; 10 repeticiones.
            cmp   #5 //(5)(10)
            bne   LAZO
            rts
      }
}


//VARIABLES PARA EL RETARDO2.
unsigned char K3@0xB3, K4@0xB4, CONT2@0xB5;

//FUNCION DE RETARDO2.
void retardo2(void) { //funcion de retardo de 1000ms
      asm {
                              clr   CONT2
                  LAZO:   mov   #200,K3   ; Inicio subrutina de retardo
                  LAZO1:  mov   #170,K4
                  LAZO2:  dbnz  K4,LAZO2
                              dbnz  K3,LAZO1  ; Fin subrutina de retardo
                              inc   CONT2      ; Código para controlar las
                              lda   CONT2      ; 100 repeticiones.
                              cmp   #10
                              bne   LAZO
                              rts
            }
}


void ajuste (int potenciometro,int valor,int estado) {
      int conversion;
      int i=0;
      conversion =  ((valor)*(100))/10000;
      switch (potenciometro){
      case 1: // SI SE SELCCIONA EL PRIMER POTENCIOMETRO

            //PTAD_PTAD1=1;

            if (estado==1)
            {
                  for(i=0;i<conversion;i++){
                        ud=1;
                        in =1;
                        retardo();
                        in=0;
                        retardo();
                        in=1;
                  }
            }
```

```
	else{
		//PTAD_PTAD1=0;

		for(i=0;i<conversion;i++){
			ud=0;
			in =1;
			retardo();
			in=0;
			retardo();
			in=1;
		}
	}


	break;
case 2: // SI SE SELCCIONA EL SEGUNDO POTENCIOMETRO
	if (estado==1)
	{
		for(i=0;i<conversion;i++){
			ud1=1;
			in1=1;
			retardo();
			in1=0;
			retardo();
			in1=1;
		}
	}
	else{
		for(i=0;i<conversion;i++){
			ud1=0;
			in1=1;
			retardo();
			in1=0;
			retardo();
			in1=1;
		}
	}

	break;

case 3: // SI SE SELCCIONA EL  POTENCIOMETRO
	if (estado==1)
	{
		for(i=0;i<conversion;i++){
			ud2=1;
			in2=1;
			retardo();
			in2=0;
			retardo();
			in2=1;
		}
	}
	else{
		for(i=0;i<conversion;i++){
			ud2=0;
			in2=1;
			retardo();
```

```
                        in2=0;
                        retardo();
                        in2=1;
                    }
                }
                break;
            } //del switch
} // del void




/*********************************************************************
*********************

*********************************************************************
******************/



//Redireccionamiento de vectores de interrupción
/*Dummy ISR */
interrupt void Dummy_ISR(void) {
}
void (* volatile const _UserEntry[])()@0xFABC= {
            0x9DCC,             // asm NOP(9D), asm JMP(CC)
            _Startup
};

// redirect vector 0xFFC0-0xFFFD to 0xFBC0-0xFBFD
void (* volatile const _Usr_Vector[])()@0xFBC4= {
            Dummy_ISR,        // Int.no.29 RTC               (at FBC4)
(at FFC4)
            Dummy_ISR,// Int.no.28 IIC               (at FBC6) (at FFC6)
            Dummy_ISR,// Int.no.27 ACMP              (at FBC8) (at FFC8)
            Dummy_ISR,// Int.no.26 ADC               (at FBCA) (at FFCA)
            Dummy_ISR,// Int.no.25 KBI               (at FBCC) (at FFCC)
            Dummy_ISR,// Int.no.24 SCI2 Transmit     (at FBCE) (at FFCE)
            Dummy_ISR,// Int.no.23 SCI2 Receive      (at FBD0) (at FFD0)
            Dummy_ISR,// Int.no.22 SCI2 Error        (at FBD2) (at FFD2)
            Dummy_ISR,// Int.no.21 SCI1 Transmit     (at FBD4) (at FFD4)
            int_rec,// Int.no.20 SCI1 Receive      (at FBD6) (at FFD6)
            Dummy_ISR,// Int.no.19 SCI1 error        (at FBD8) (at FFD8)
            Dummy_ISR,// Int.no.18 TPM2 Overflow     (at FBDA) (at FFDA)
            Dummy_ISR,// Int.no.17 TPM2 CH1          (at FBDC) (at FFDC)
            Dummy_ISR,// Int.no.16 TPM2 CH0          (at FBDE) (at FFDE)
            Dummy_ISR,// Int.no.15 TPM1 Overflow     (at FBE0) (at FFE0)
            Dummy_ISR,// Int.no.14 TPM1 CH5          (at FBE2) (at FFE2)
            Dummy_ISR,// Int.no.13 TPM1 CH4          (at FBE4) (at FFE4)
            Dummy_ISR,// Int.no.12 TPM1 CH3          (at FBE6) (at FFE6)
            Dummy_ISR,// Int.no.11 TPM1 CH2          (at FBE8) (at FFE8)
            Dummy_ISR,// Int.no.10 TPM1 CH1          (at FBEA) (at FFEA)
            Dummy_ISR,// Int.no.9  TPM1 CH0          (at FBEC) (at FFEC)
            Dummy_ISR,// Int.no.8  Reserved          (at FBEE) (at FFEE)
```

```
            Dummy_ISR,// Int.no.7  USB Statue        (at FBF0) (at FFF0)
            Dummy_ISR,// Int.no.6  SPI2              (at FBF2) (at FFF2)
            Dummy_ISR,// Int.no.5  SPI1              (at FBF4) (at FFF4)
            Dummy_ISR,// Int.no.4  Loss of lock      (at FBF6) (at FFF6)
            Dummy_ISR,// Int.no.3  LVI               (at FBF8) (at FFF8)
            Dummy_ISR,// Int.no.2  IRQ               (at FBFA) (at FFFA)
            Dummy_ISR,// Int.no.1  SWI               (at FBFC) (at FFFC)
};

/*************************************************************
 * Código inicialización del Bootloader (IMPORTANTE..no modificar*
 *************************************************************/
#pragma CODE_SEG Bootloader_ROM

void Bootloader_Main(void);

void _Entry(void) {
      PTGD = 0x00;
      PTGDD = 0xF0;                    //PTG0-3 used for KBI input
      PTGPE = 0x0F;                           //Pull-up enable

      // MCG clock initialization, fBus=24MHz
      MCGC2 = 0x36;
      while (!(MCGSC & 0x02))
            ;                 //wait for the OSC stable
      MCGC1 = 0x1B;
      MCGC3 = 0x48;
      while ((MCGSC & 0x48) != 0x48)
            ;     //wait for the PLL is locked

      // Flash clock
      FCDIV = 0x4E; // PRDIV8=1; DIV[5:0]=14, flash clock should be 150-
200kHz
      // bus clock=24M, flash clock=fbus/8/(DIV[5:0]+1)
      // bus clock=24M, flash clock=fbus/8/(DIV[5:0]+1)
      if (!PTGD_PTGD0) {
            SOPT1 = 0x20; // disable COP only if bootloader mod is
requested
            // PTG0 is pressed
            USBCTL0 = 0x44;
            Bootloader_Main();            // Bootloader mode
      } else
            asm JMP _UserEntry;
      // Enter User mode
}

#pragma CODE_SEG default
```