Anexo 1

Conceptos generales

Electromagnetismo

Es la generación del magnetismo no sobre el giro de los electrones dentro de un núcleo, como es de forma natural, sino aplicando una corriente sobre un conductor, con lo que se logra el movimiento libre de los electrones [1].

La generación del campo magnético se da cuando existe un diferencial de corriente directa en un espacio libre o por medio de un conductor, como en el caso de un electroimán.

Por lo tanto para definir la magnitud del campo generado se utiliza la ley de Biot-Savart, en la cual se establece que en cualquier punto la magnitud del campo magnético es proporcional al producto de la corriente, el diferencial de longitud del conductor y el seno del ángulo formado, denotándolo de forma vectorial mediante la ecuación (1):

$$dH = \frac{IdL \times a_R}{4\pi R^2} = \frac{IdL \times R}{4\pi R^3} \quad \ldots(1)$$

Donde **H** es la intensidad del campo magnético (A/m) amperes por metro, **L** es la longitud que se desplaza.

Aplicando finalmente la ley Biot-Savart de forma integral:

$$H = \oint \frac{IdL \times a_R}{4\pi R^2} \quad \ldots(2)$$

La cual nos define la intensidad del campo magnético según la corriente presente en el área y la longitud que esta se desplaza [2].

Electroimán

Es el Material ferromagnético, el cual genera un campo magnético proporcional a la corriente que se le es aplicada [3].

El magnetismo dado a estos componentes los podemos obtener mediante el frotamiento de un imán natural con un artificial, además de la percusión dentro de un campo magnético; por otro lado, también se le puede aplicar una corriente eléctrica a través de un solenoide para convertirlo de este modo en un imán [3].

Lo necesario para generar un campo magnético con un electroimán, es la cantidad de corriente que fluye por un alambre enrollado a un núcleo de hierro. La magnitud de este campo generado depende en su mayoría de la calidad con la que se fabricó dicho núcleo [4].

La aplicación de los electroimanes lo podemos observar desde la levitación en los trenes eléctricos [5], hasta investigación en pequeña escala como la toma de datos en la aceleración de particular [6].

Microcontrolador

Pequeños dispositivos electrónicos que tiene un circuito integrado al cual se le permite ser programados mediante diferentes lenguajes de máquina, a fin de realizar una variedad de tareas específicas según las indicaciones del usuario.

Hay una variedad de marcas que podemos encontrar en el mercado actual, los más conocidos son el ATMEGA de la empresa Atmel (figura 2.7), luego existe el PIC16Fxxx de Microchip ® Architecture..

Figura 2.7. Atmega88 – Atmel [7]

Figura 2.8. PIC16F84A – Microchip ® Architecture [8]

Motor de corriente continúa

Es una maquina electromecánica cuya función es la de transformar la energía eléctrica continua en energía mecánica, obteniendo de este modo un movimiento giratorio causada por la interacción de un campo magnético giratorio y un rotor, o como también puede ser de forma inversa.

Se puede sacar la relación eléctrica entre el voltaje que se es asignado al motor, la corriente con la que trabaja y la velocidad del motor, según la siguiente fórmula 1.2.1:

$$V_t\ t\ =\ L_a \frac{di(t)}{dt} +\ R_a i\ t\ +\ K_e * \dot{\theta}(t) \text{.....} (3)$$

En donde V_t es el voltaje con que se alimenta el circuito, L_a y R_a son las inductancias y la resistencia respectivamente.

Por otro lado también se puede obtener la relación que existe entre la corriente y el par de torsión del motor, como se presenta en la fórmula 1.2.2.

$$T_m\ t\ =\ K_T * i(t) \ldots (4)$$

En donde T_m es el par de torsión generado por el motor, mientras que K_T es la ganancia del par de torsión [9].

Temporizador

Es el circuito dedicado a la contabilización del tiempo para cierta cantidad de funciones o tareas específicas. Mayormente es diseñado con el temporizador 555, el cual permite realizar una gama de diseños y circuitos adecuados a cualquier tipo de trabajos, evitando el alto costo, además de tener una gran simplicidad de manejo y su versatilidad [10], lo cual permite tener una mayor precisión en el control del tiempo.

También se puede programar un reloj gracias a la tecnología actual de los microcontroladores, donde gracias a un reloj interno que cuentan, podemos adecuar un contador según el tiempo que requiera el operador.

Luego tenemos una variedad de temporizadores contadores tanto programables como el ICM 8240, ICM 8250, etc. ó como temporizador fijo como el XR 2242 [10].

Plancha

Lamina de metal que se tiene ubicado en la parte superior del equipo, el cual sirve para colocar la solución con la que se va a trabajar [11].

Magneto

Material metálico con propiedades magnéticas, lo cual permite interactuar con imanes por medio de atracciones o repulsiones [11].

Agitación

Efecto de mover con cierta frecuencia cualquier elemento [11].

Fuente

[1] BUBAN, Peter

1987 Electricidad y electrónica, aplicaciones prácticas. México: Tipografía Barsa S.A. Tomo I.

[2] HAYT, William. BUCK, John.

2007 Teoría electromagnética. 7ma Edición. México: McGraw-Hill Interamericana.

[3]COLEGIO DE CAPITANES Y PILOTOS DE LA MARINA MERCANTE NACIONAL A.G.

2010 Magnetismo. Chile. [consultado en línea 2011/10/27] <http://www.colcap.cl/doctos/magnetismo.pdf>

[4] ULABY, Fawwaz

2010 Fundamentals of applied electromagnetics / Fawwaz T. Ulaby, Eric Michielssen, Umberto Ravaioli. 6a. ed. Boston: Prentice Hall.

[5] MOCHIZUKI, Asahi.

2010 Electric Trains and Japanese Technology. *Japan Railway & Transport Review.* [en línea]. No. 55. Marzo 2010. [consultado 2011/10/27] <http://www.jrtr.net/jrtr55/pdf/30-38web.pdf>

[6] RODRÍGUEZ, W. CARDONA, J.

2011 Automatización de la Medición de Perfiles de Campo Magnético en Imanes y Electroimanes utilizados en Aceleradores de Partículas. *Revista Colombiana de Física*. [en línea]. Vol. 4 (2). 2011. [consultado 2011/10/27]. < http://revcolfis.org/ojs/index.php/rcf/article/view/430230/pdf>

[7] SUPER SCG

2012 Atmega88 mm21 [en linea] [Consultado 03/10/2012] <http://www.superscg.com/atmega88-pi-196.html>

[8] SERGIO HERNANDEZ BLOG

2012 PIC16F84A [En línea] [Consultado 03/10/2012] <http://sergio-hdz.blogspot.com/2012/05/pic16f84a.html>

[9] CETINKUNT, Sabri

2009 Mecatrónica. México: Grupo Editorial Patria.

[10] WILLIAMS, Arthur.

1990 Dispositivos PLL de fuentes reguladas y telecomunicaciones. México: McGraw-Hill Interamericana.

[11] REAL ACADEMIA ESPAÑOLA

Vigésima segunda edición [consultado en línea]

Anexo 2

Código MatLab

```
%valores con el motor M2232U de DC-Micromotors
clear all;
close all;
clc;

J=4.5;      %momento de inercia gm^2/s^2
b=1.18;     %coeficiente de amortiguamiento mNms
K=6.83;     %constante de fuerza electromotriz mNm/amp
R=1.60;     %Resistencia eléctrica mohm
L=0.750;      %inductancia eléctrica mH
num=K;      %constante PID
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];

s = tf('s');                              %variable S de transformada
P_motor = K/((J*s+b)*(L*s+R)+K^2);    %funcion de transferencia del
motor
zpk(P_motor);                             %o mejor dicho de la Planta


Ts = 0.05;
dP_motor = c2d(P_motor,Ts,'zoh');     %Funciond e transferencia de
la
zpk(dP_motor);                            %planta de forma discreta

sys_cl = feedback(dP_motor,1);            %sistema de lazo cerrado
[y,t] = step(sys_cl,12);                  %con respuesta ante un escalon
stairs(t,y);
xlabel('Tiempo (s)')
ylabel('Velocidad (rad/s)')
title('Respuesta al escalon: Original')


%Adicion de un control proporcional
Kp=100;                                 %valor de la constante
proporcional
Ki=200;
Kd=18.5;
C=Kp+Ki/s+Kd*s;                              %Ecuacion de control, en
este caso P
dC=c2d(C,Ts,'tustin');                  %forma discreta
sys_cl= feedback(dC*dP_motor,1);     %evaluacion del sistema
multiplicando

%ahora revisaremos como responde ante un escalon
[x2,t] = step(sys_cl,12);
stairs(t,x2)
grid
xlabel('Tiempo (s)')
ylabel('Velocidad (rad/s)')
title('Respuesta escalon - control proporcional')



%Debido a la inestabilidad presentada
%se observaran las raices donde nuestro sistema es estable
```

```
%y encontrar la constante deseada
rlocus(dC*dP_motor)
axis([-1.5 1.5 -1 1])
title('Sistema de compensacion - Raices')

%de este modo modificamos los polos
%para tener unas ganancias que al menos cumplan en cierto rango
%movemos la grafica del eje -0.82
z = tf('z',Ts);
dC = dC/(z+0.94);
rlocus(dC*dP_motor);
axis([-1.5 1.5 -1 1])
title('Grafica de raices');

%Buscamos el valor de K dentro de la grafica
% y dentro del rango de ganancias permitidas
%[K,poles] = rlocfind(dC*dP_motor);

%una vez seleccionado el valor de K anterior
%lo introducimos dentro de la función de transferencia
% y observar comor responde el mismo k=0.5861 prox 0.6
sys_cl = feedback(0.6*dC*dP_motor,1);
[x3,t] = step(sys_cl,8);
stairs(t,x3)
xlabel('Tiempo (seconds)')
ylabel('Velocidad (rad/s)')
title('Respuesta escalon - Control PID modificado')
```

Gráficas Matlab

```
b=1.18;      %coeficiente de amortiguamiento mNms
K=6.83;      %constante de fuerza electromotriz mNm/amp
R=1.60;      %Resistencia eléctrica mohm
L=0.750;       %inductancia eléctrica mH
num=K;       %constante PID
den=[(J*L)  ((J*R)+(L*b))  ((b*R)+K^2)];

s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2);
>> P_motor

Transfer function:
          6.83
---------------------------
3.375 s^2 + 8.085 s + 48.54
```

Figura A2.1. Función de transferencia de la planta.

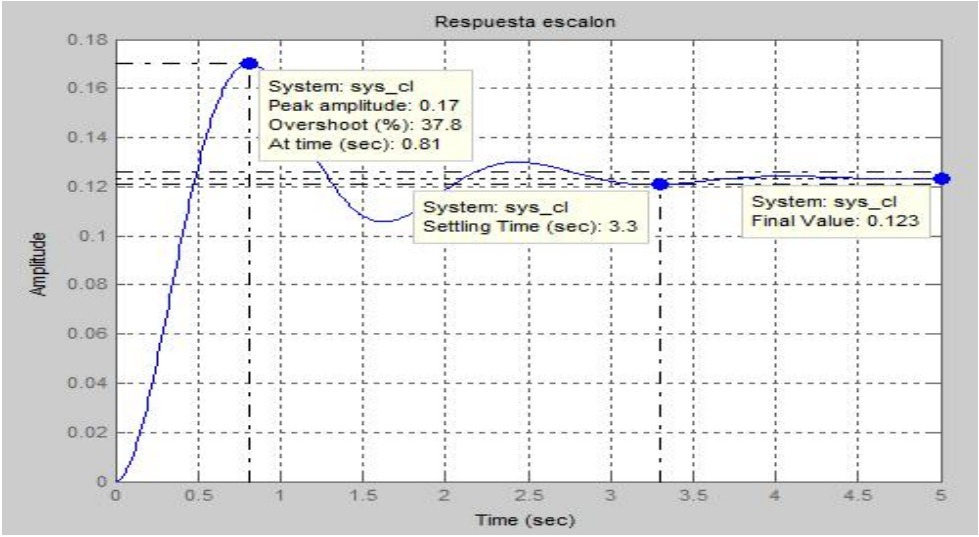

Figura A2.2. Respuesta escalón de la planta

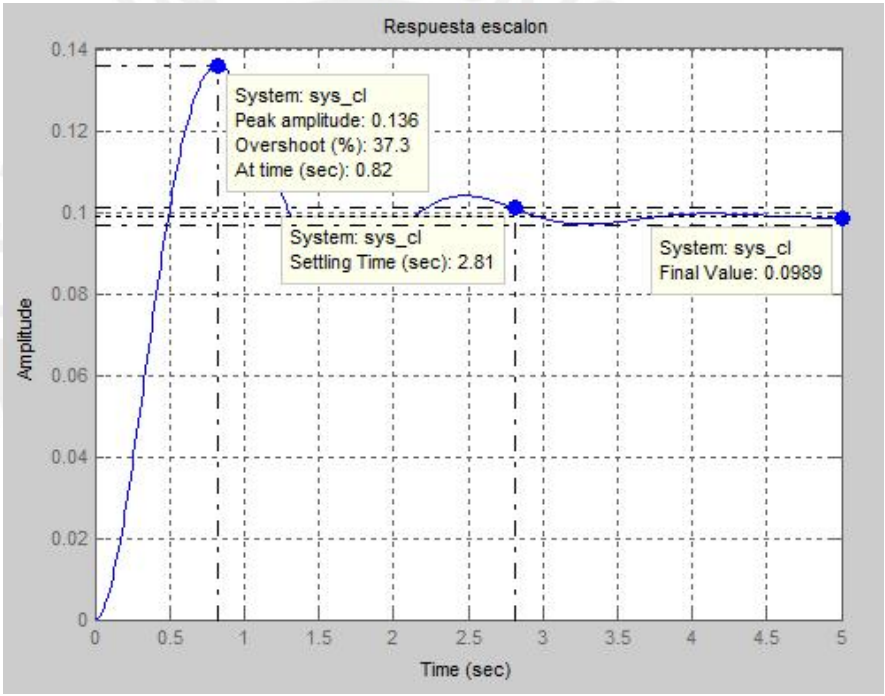

Figura A2.3. Respuesta escalón con control proporcional

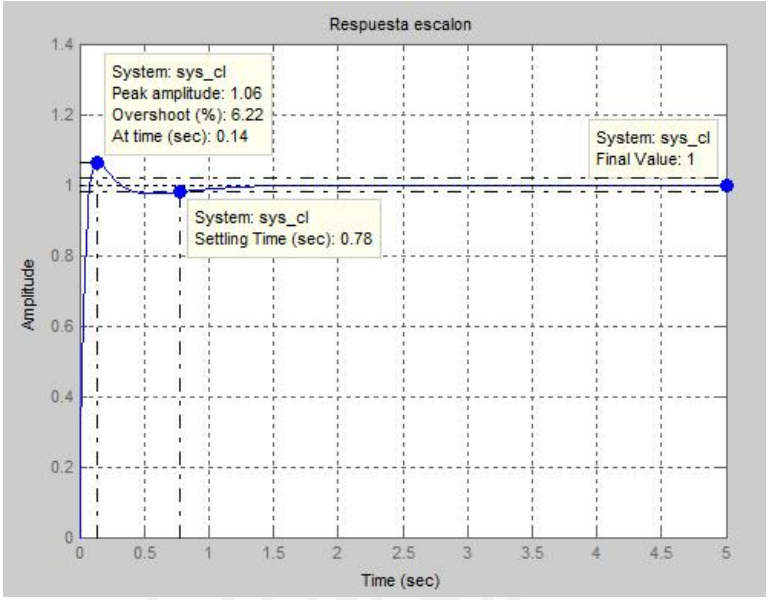

Figura A2.4. Respuesta escalón con valores de control PID

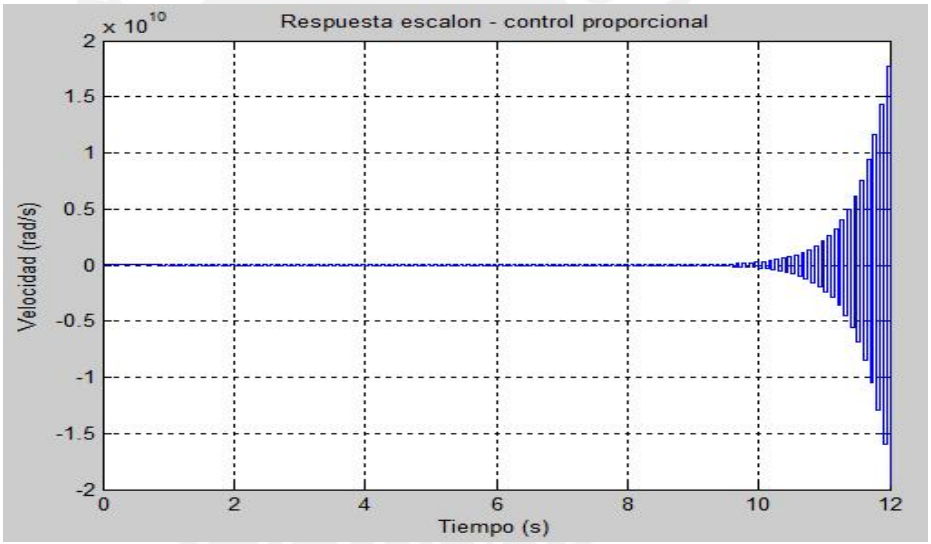

Figura A2.5. Respuesta escalón de forma discreta.

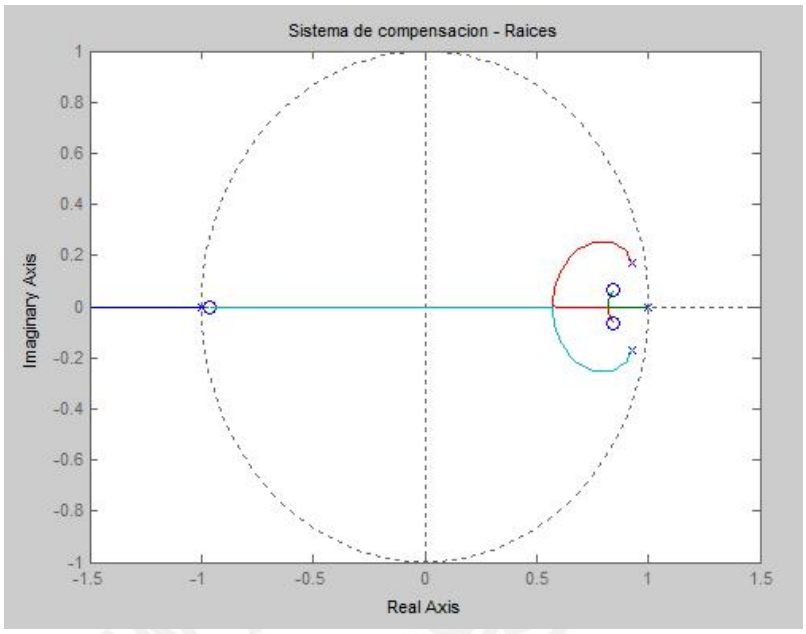

Figura A2.6. Gráfica de raíces del sistema.

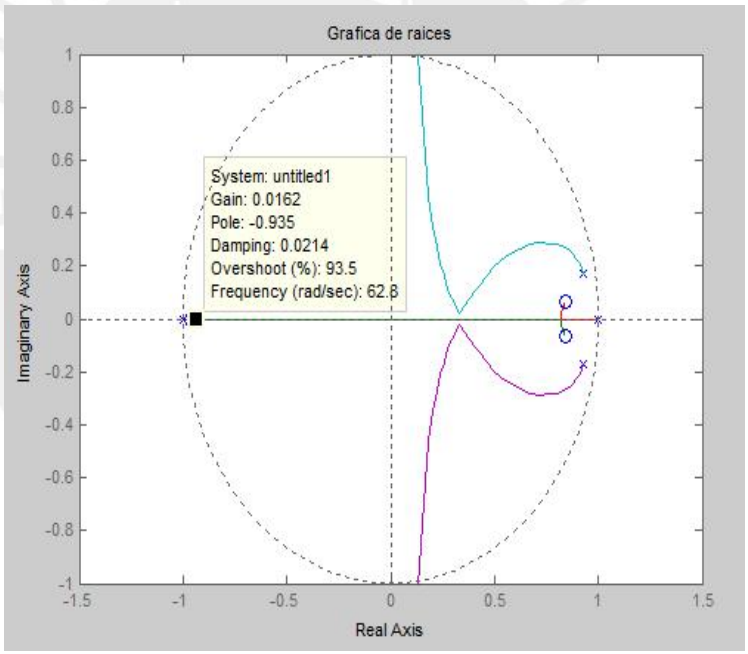

Figura A2.7. Función discreta más el polo adicionado.

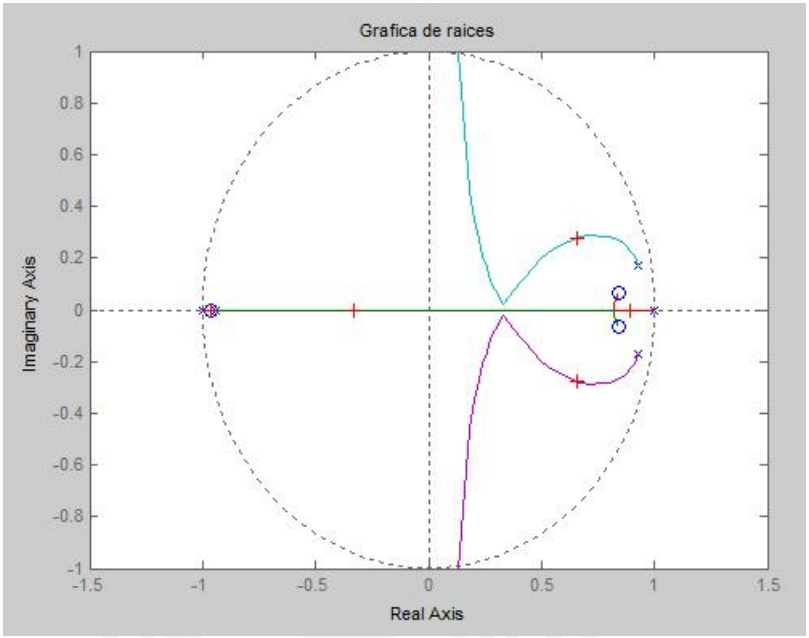

Figura A2.8. Selección de punto de ganancia dentro de la gráfica.

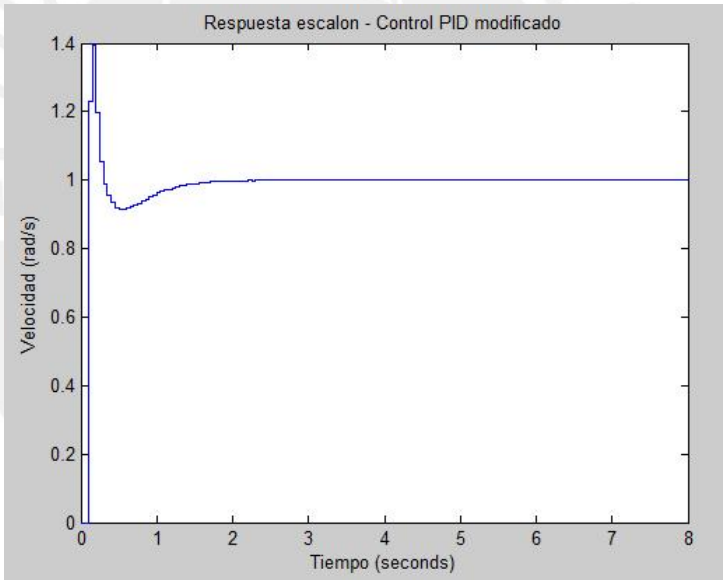

Figura A2.9. Gráfica respuesta del sistema PID con control modificado.

Código AVR Studio 6.0

```
/*************************************************************************
 *  Proyecto_tesis.c
 *  Proyecto:        Agitador Magnético cronometrado con control de
 *                              temperatura y con alarma sonora y visual
 *
 *  Author:                 Diego Calderón Lamas
 *  Carrera:         Ingeniería Electrónica
 *  Código:                 20062312
 *  Universidad:     Pontificia Universidad Católica del Perú
 *
 *  Tarea:                  Programa para el agitador magnético
 *                              Se hace el detalle del control de la
velocidad del motor
 *                              según las entradas que ofrece el usuario y
los sensores
 *                              de velocidad
 *                              al igual que ocurre con la temperatura,
nivelando calor
 *                              que se va a ofrecer a la solución
 *  Procesador:             Atmega88
 *
 *  F_CPU:                  8 MHz
 *************************************************************************/

// definimos las ganancias del motor
#define P_GAIN 100
#define I_GAIN 200
#define D_GAIN 15

// determina las ganancias de la temperatura
#define P_GAIN_t 0.2045
#define I_GAIN_t 0.1023
#define D_GAIN_t 1.0225

#include "avr_compiler.h"
#include "lcd.h"


#include <avr/io.h>

// definimos las variables con las que se va a trabajar a lo
// largo del programa

int set_rpm = 0;
int error, feedback_rpm, output;
int e1=0, e2=0;
int q0=0, q1=0, q2=0;
int    u=0;


int set_temp = 0;
int error_temp, feedback_temp, salida_temp;
int et1=0, et2=0;
int qt0=0, qt1=0, qt2=0;
int    p=0;

int umax=1500, umin=0;                  // valores maximos y minimos de rpm
int    tmax=100, tmin=0;                // valores maximos y minimos de
temperatura
```

```
short  reloj=0;                                    //indicador de si se presiono
o no temporizador
char buff[16];

short  c = 0;
unsigned int pulses, tiempo_seg=0, tiempo_lcd;

/*********************************************************************/
/*             funcion para retraso de t milisegundos                */
/* Realiza un retraso de la cantidad de segundos quqe se ingrese           */
/*********************************************************************/
void delay_ms(unsigned int t)
{
       while(t--)
       delay_us(1000);
}

/*********************************************************************/
/*             funcion para retraso de t segundos                    */
/* Realiza un retraso de la cantidad de segundos quqe se ingrese           */
/*********************************************************************/
void delay_sec(unsigned short t)
{
       short i=0;
       for(i = 0;i < t; i++)
       _delay_ms(250);
}

/*********************************************************************/
/* Configuracion de los puertos del microcontrolador                 */
/* Configura los puertos de entrada y salida segun los pulsadores y los    */
/* potenciometros
                                  */
/*********************************************************************/
void ini_ports()
{
       DDRD=(0<<2);        //PD2 - INT0 entrada del encoder para verificar la
cantidad de pulsos por vuelta (100 pulsos)

       DDRC|=(0<<3)|(0<<4)|(0<<5);             // entrada de potenciometros y
pulsdores
       PORTC |= (1<<3)|(1<<4)|(1<<5);     // PC3-PC5 configurar como entradas
de los pulsadores pull-ups

       DDRB = (1<<6)|(1<<7);                   // portb como salida de pin de
indicador LED y de la alarma sonora.
       PORTB |= (0<<6)|(0<<7);                 // limpiamos las salidas
}



/*********************************************************************/
/* Configuracion de interruptores                                    */
/* inicio interruptores para el calculo de la velocidad
       */
/* iniciar interrupcion externa (INT0)
                     */
/* iniciar interrupcin de cambio de pin
                     */
/*********************************************************************/

void init_interrupts()
```

```
{
	PCICR = (1<<PCIE1);                         //interruptor por cambio de
PIN del pulsador
	PCMSK1 = (1<<3)|(1<<4)|(1<<5);                          //pulsador pin 3
PC3 - inicio de periodo

	EICRA |= (1 << ISC00)|(1 << ISC01);
	// INT0 activado
	EIMSK |= (1<<INT0);
	sei();
}


/*********************************************************************/
/* Configuracion del ADC                                             */
/* Establecer los valores para la toma de datos y conversion ADC del */
/* microcontrolador                                                  */
/* Aref=AVcc, pres=128                                               */
/*********************************************************************/
void ini_ADC()
{
	ADMUX=(1<<REFS0);                          //  Aref=AVcc;
	ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); //prescalar div
factor =128
}


/*********************************************************************/
/* Configuracion del timer 2                                         */
/* Timer2 modo CTC Frecuencia = F_CPU/(N*(1+OCR2A)) = 4 Hz           */
/* interrupcion despues de 250ms; N = 1024                           */
/*********************************************************************/
void ini_timer2()
{
	// Prescalar 1024
	TCCR2B |= (1 << CS20) | (1 << CS21) | (1 << CS22) ;
	// modo CTC
	TCCR2A |= (1 << WGM21);
	// Enable Interrupt from timer2 compare
	TIMSK2 |= (1 << OCIE2A);
	TCNT2 = 0x00;
	OCR2A = 1952;
}


/*********************************************************************/
/* Rutina de interrupcion para comparacion timer2                    */
/* Toma la cantidad de pulsos que genera el encoder luego de 500 ms  */
/* entradas:  pulsos del encoder                                     */
/* salidas:   cantidad de pulsos por 500 ms, lectura feedback para el*/
/*            PID del motor                                          */
/*********************************************************************/

ISR (TIMER2_COMPA_vect)
{
	TCNT2=0;
	c++;
	if(2==c)
	{
		feedback_rpm = pulses*5;
		pulses = 0;
		c=0;
```

```
	}
}


/*********************************************************************/
/* Interruptor para el pulso externo                                 */
/* Función de detectar el pulso cada ves que entra por medio del encoder*/
/* Definiendo de esta forma la velocidad del motor
		*/
/*********************************************************************/

ISR (INT0_vect)
{
	pulses++;  // aumenta el pulso +1
}


/*********************************************************************/
/* Temporizador                                                      */
/* Trabajo luego de que se presiono el cambio de pin para seleccionar el*/
/* tiempo que va a realizar la tarea                                 */
/* entradas: --
                                        */
/* salidas:  tiempo del cronometro
*/
/*********************************************************************/
ISR (PCINT1_vect)
{

	char dato = ~(PINC|0xC7);		// enmascaro el Pin 3 PC3, PC4 y PC5

	switch(dato)
	{
		case 8:
			if (reloj==0)
			{
				reloj=1;

			}
			break;
		case 16:
			tiempo_seg++;
			break;
		case 32:
			tiempo_seg--;
			break;
	}


		// indicación del tiempo de trabajo por parte del usuario
		if (tiempo_seg>=12)
		{
			tiempo_seg=12;

		}
		// indicación del tiempo de trabajo por parte del usuario
		if (tiempo_seg<=0)
		{
			tiempo_seg=0;

		}
```

```
}


/*********************************************************************/
/* Lectura del ADC respectivo                                        */
/* Recibir y convertir el dato analogico para pasarlo a digital, a fin  */
/* trabajar correctamente con el dato dentro del microcontrolador       */
/* entradas: canal de lectura (0..5)                                 */
/* salidas:  ADC convertido                                          */
/*********************************************************************/
uint16_t ReadADC(uint8_t ch)
{
       ADMUX &= 0xF8;     // separo los adc, dependiendo del canal a leer
       ADMUX |= ch;

       ADCSRA|=(1<<ADSC); //iniciamos conversion

       while(ADCSRA & (1<<ADSC)); // esperamos a que se realice la conversion

       ADCSRA|=(1<<ADIF); //Reiniciamos el valor de ADIF para la siguiente
conversion

       return (ADC);
}

/*********************************************************************/
/* Inicializacion del PWM del microcontrolador                       */
/* Se inicia la configuración de la onda de PWM para el microcontrolador*/
/* Entradas: --                                                      */
/* Salidas:  OCR1A = 0, ICR1= 1700 (tope), OCR1B=0                   */
/*********************************************************************/
void ini_PWM()
{

       /* Configuración del Timer1 para el motor y de la temperatura
        * motor =1700:300rpm - - temperatura = 1700:100°C
     * - Bits WGM: PWM de Fase y Frecuencia Correctas con tope de conteo =
ICR1
     * - Bits COM: PWM no-invertida en los dos canales A y B
     * - Bits CS:  Fuente de reloj = F_CPU/8
     */
    TCCR1A = (1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
    TCCR1B = (1<<WGM13)|(1<<CS11)|(1<<CS10);    // pre=8
    DDRB = (1<<1)|(1<<2);          // Pines OC1A salidas PWM PB1 y PB2

       ICR1 = 1700;      // frecuencia PWM = 1MHZ/(2*tope ) = 300 Hz - tope=
1700 aprox
       TCNT1 = 0;
       OCR1A = 0;
       OCR1B = 0;

}

/*********************************************************************/
/* Funcion control de la velocidad del motor                         */
/* entradas: seteo de la velocidad, velocidad actual del motor       */
/* salidas:  correcion del error de la velocidad                     */
/*********************************************************************/
void PID_motor(int a, int b)
{
       //calculamos el error proporcinal
       error = a - b;
```

```
	u=u+q0*error+q1*e1+q2*e2;                       //calculamos el valor
de salida de las rpm
	if (u*umax/1700>umax)
	// según el dato de entrada y la velocidad del motor
	{  u=umax;                                                      //si
sobrepasa el valor maximo, quedarse ahi
	}
	if (u<umin)                                                     //si es
menor al valor minimo, quedarse ahi
	{  u=umin;
	}

	e2=e1;
	e1=error;
	return u;

}

/*********************************************************************/
/* Funcion control de la temperatura                                 */
/* entradas: seteo de la temperatura, temperatura actual del sistema */
/* salidas:  correcion del error de la temperatura                   */
/*********************************************************************/
void PID_temp(int x, int y)
{
	//calculamos el error proporcinal
	error_temp = x - y;

	p=p+qt2*error_temp+qt1*et1+qt2*et2;
	//calculamos el valor de salida de las rpm

	if (p*tmax/1700>tmax)
	// según el dato de entrada y la velocidad del motor
	{  p=tmax;                                                      //si
sobrepasa el valor maximo, quedarse ahi
	}
	if (p<tmin)                                                     //si es
menor al valor minimo, quedarse ahi
	{  p=tmin;
	}

	et2=et1;
	et1=error_temp;
	return p;

}




/*********************************************************************/
/* Programa principal                                                */
/* Realiazcion del trabajo de la velocidad del motor, como del tiempo de*/
/* trabajo y el control de la temperatura                            */
/*********************************************************************/
int main(void)
{

	uint16_t temp_res_adc;            // variable resistencia temperatura
	uint16_t temp_adc;                // variable detector de temperatura
	int vel_mo;          //velocidad del motor para mostrar en LCD
```

```
        int temp_sis;        // temperatura del sistema


        ini_ports();         //iniciamos los puertos

        lcd_init();          //iniciamos pantalla LCD

        ini_timer2();        //iniciamos el contador CTC

        ini_PWM();           //iniciamos el PWM

        ini_ADC();           //iniciamos el ADC

        init_interrupts();          //iniciamos interrupciones

        q0=P_GAIN+D_GAIN/0.002;
        q1=(-P_GAIN)+I_GAIN*0.002-2*(D_GAIN/0.002);
        q2=D_GAIN/0.002;

        qt0=P_GAIN_t+D_GAIN_t/0.002;
        qt1=(-P_GAIN_t)+I_GAIN_t*0.002-2*(D_GAIN_t/0.002);
        qt2=D_GAIN_t/0.002;

        while(1)
        {
        // programación para regulacion del motor y temperatura sin necesidad
de un cronometro

        /*while(tiempo_seg==0 ^ reloj==0)
        {
                delay_sec(1);
                set_rpm=ReadADC(1);                 // leo el valor que setea el
usuario

                temp_adc=ReadADC(0);
        //sensor lm35 de temperatura de la resistencia
                temp_res_adc=ReadADC(2);
        //Variación de la temperatura según el usuario
                feedback_temp=temp_adc*1.66;                                    //
obtenemos el valor en que se encuentra la temperatura de la resistencia
                set_temp=temp_res_adc*1.66;         // se multiplica por la
constante (1700/1024)
                set_rpm=set_rpm*1.66;       // para que el valor sea igual al que
se emite por el OCR1A o OCR1B

                PID_motor(set_rpm,feedback_rpm);
                PID_temp(set_temp,feedback_temp);

                OCR1A=u/10;                                 //definimos los valores
del PWM para el motor
                OCR1B=p/10;                                 // y para la
temperatura

                //comenzamos a mandar caracteres por el LCD
                lcd_gotorc(1,1);                            //nos ubicamos en la
fila 1 y columna 2
                lcd_puts("Velocidad motor:");       // especificamos la velocidad
del motor
                vel_mo=floor(OCR1A*0.88);           // convertimos el valor del
OCR1A en RPM

                //enviamos valor por el LCD
```

```
                lcd_gotorc(2,1);
                sprintf(buff, " %5drpm", vel_mo);  // Formatear
                lcd_puts(buff);
                delay_sec(1);
                lcd_clear();

                //mostramos valor de temperatura luego de 2 segundos
                lcd_gotorc(1,1);                        //nos ubicamos en la
fila 1 y columna 2
                lcd_puts("Temperatura:");   // especificamos la velocidad del
motor
                temp_sis=floor(OCR1B*0.05);        // convertimos el valor del
OCR1A en RPM
                lcd_gotorc(2,1);
                sprintf(buff,"%5d C", temp_sis);
                lcd_puts(buff);

        }

        while(reloj==1)
        {*/
                // iniciamos indicando la cantidad de tiempo que se requiere
hacer la prueba

                do
                {
                        PORTB |= (0<<6)|(0<<7);                 // limpiamos las
salidas del led y alarma
                        lcd_clear();
                        lcd_gotorc(1,1);
                        lcd_puts("Tiempo de prueba:");
                        tiempo_lcd=tiempo_seg*5;
                        lcd_gotorc(2,1);
                        sprintf(buff,"%4d min",tiempo_lcd);
        // preguntamos al usuario el tiempo de prueba
                        lcd_puts(buff);
                        delay_ms(300);


                } while (reloj==0);
                // hasta que presione el boton de CRONOMETRO

                tiempo_lcd=tiempo_seg*5;
                reloj=0;

                do
                {
                set_rpm=ReadADC(1);                // leo el valor que setea el
usuario

                temp_adc=ReadADC(0);
        //sensor lm35 de temperatura de la resistencia
                temp_res_adc=ReadADC(2);
        //Variación de la temperatura según el usuario
                feedback_temp=temp_adc*1.66;                                    //
obtenemos el valor en que se encuentra la temperatura de la resistencia
                set_temp=temp_res_adc*1.66;        // se multiplica por la
constante (1700/1024)
                set_rpm=set_rpm*1.66;       // para que el valor sea igual al que
se emite por el OCR1A o OCR1B

                //comenzamos a mandar caracteres por el LCD
```

```
            lcd_gotorc(1,1);                        //nos ubicamos en la
fila 1 y columna 2
            lcd_puts("Velocidad y temp:");     // especificamos la velocidad
del motor y la temperatura

            //enviamos valor por el LCD
            lcd_gotorc(2,1);
            sprintf(buff, "%3d rpm %3d C", set_rpm, set_temp);  // escribir
los datos
            lcd_puts(buff);
            delay_ms(300);

            } while (reloj==0);

            reloj=0;

            do
            {

            //empieza a funcionar el equipo mientras cuenta el cronometro
            PID_motor(set_rpm,feedback_rpm);
            PID_temp(set_temp,feedback_temp);

            OCR1A=u/10;                             //definimos los valores
del PWM para el motor
            OCR1B=p/10;                             // y para la
temperatura

            lcd_clear();
            lcd_gotorc(1,1);
            lcd_puts("tiempo restante:");
            sprintf(buff, "%3d min", tiempo_lcd);                  // se
muestra la cantidad de tiempo que falta
            lcd_gotorc(2,1);
      // para que acabe el trabajo.
            lcd_puts(buff);
            delay_sec(60);
            tiempo_lcd--;

            } while (tiempo_lcd>0);

            do
            {
                  lcd_clear();
                  lcd_gotorc(1,1);
                  lcd_puts("Prueba terminada");                        //
se indica al usuario que la tarea se acabo
                  delay_ms(300);
                  OCR1A=0;
                  OCR1B=0;
                  tiempo_seg=0;
                  PORTB |= (1<<6)|(1<<7);                  // activamos el
led y la alarma sonora.
            } while (reloj==0);                           // no realiza
otra tarea hasta que el usuario no indique o presione el pulsador CRONOMETRO
            reloj=0;

            }

}
```

Imágenes simulaciones PROTEUS

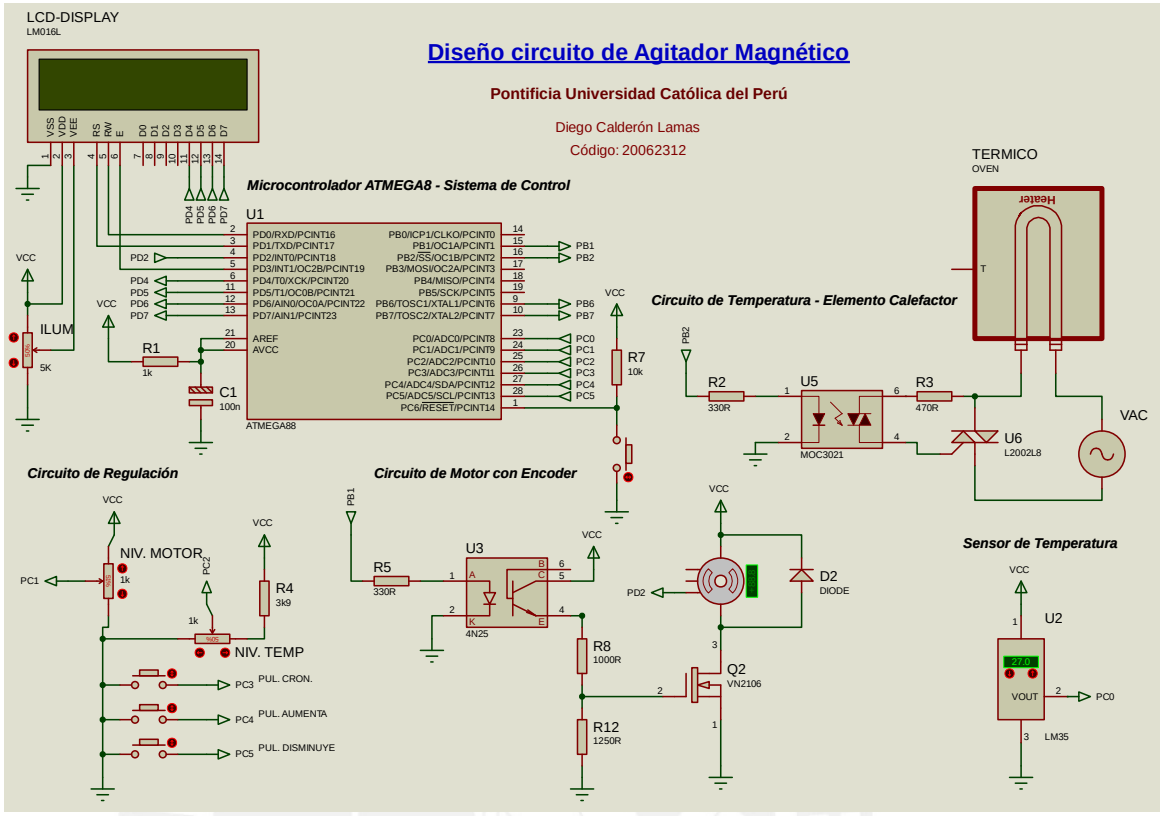

Figura A2.10. Circuito total del agitador magnético.

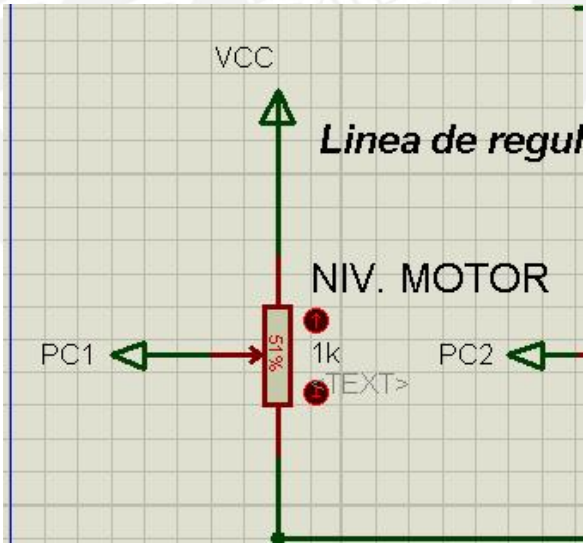

Figura A2.11. Nivel medio de velocidad.

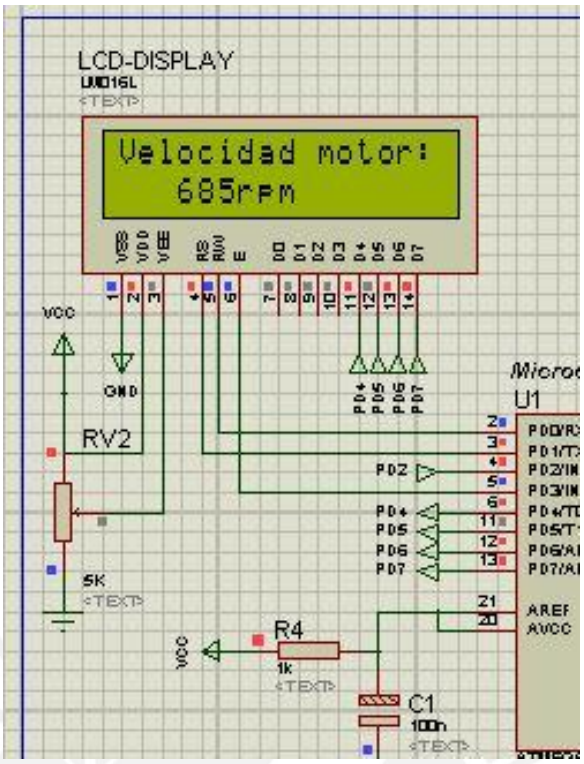

Figura A2.12. Velocidad del motor en display LCD

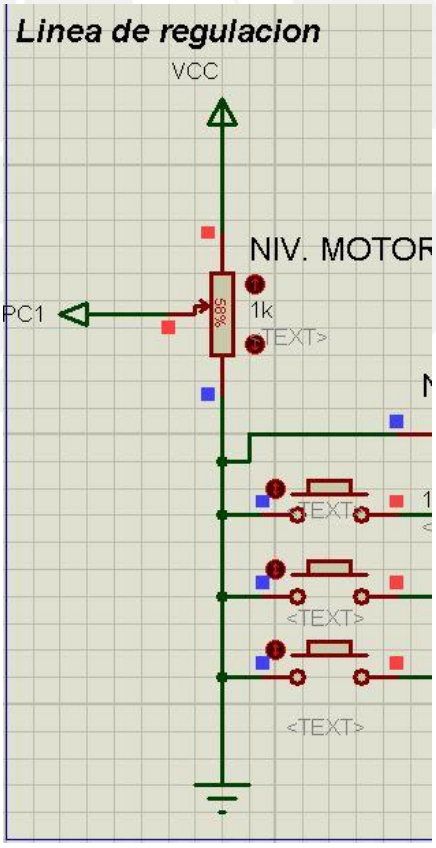

Figura A2.13. Valor del potenciómetro a 900 revoluciones aproximadamente

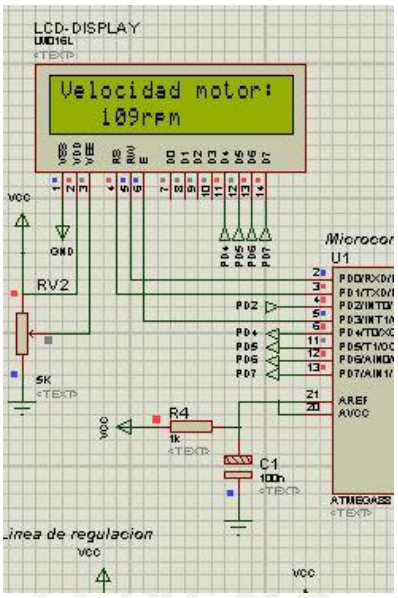

Figura A2.14. Aumento de la velocidad hasta la deseada.

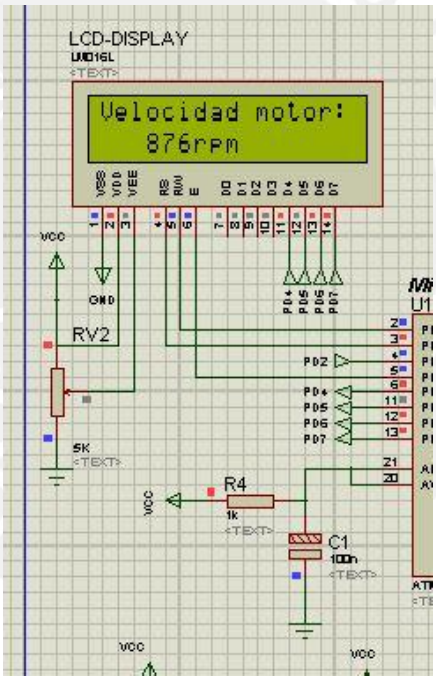

Figura A2.15. Velocidad casi similar a la deseada.

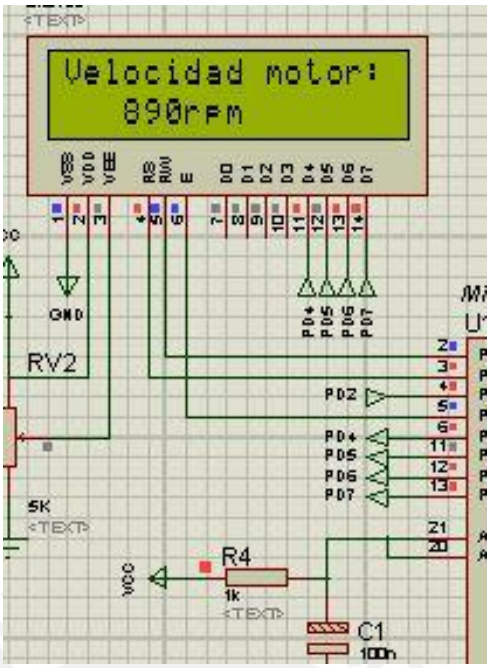

Figura A2.16. Velocidad deseada con variaciones muy pequeñas.

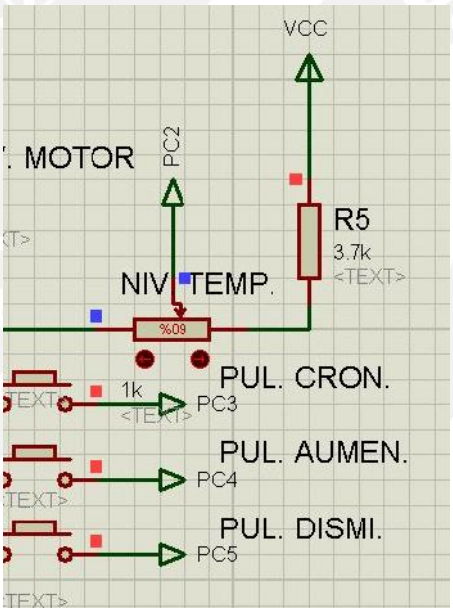

Figura A2.17. Regulación 60°C del sistema.

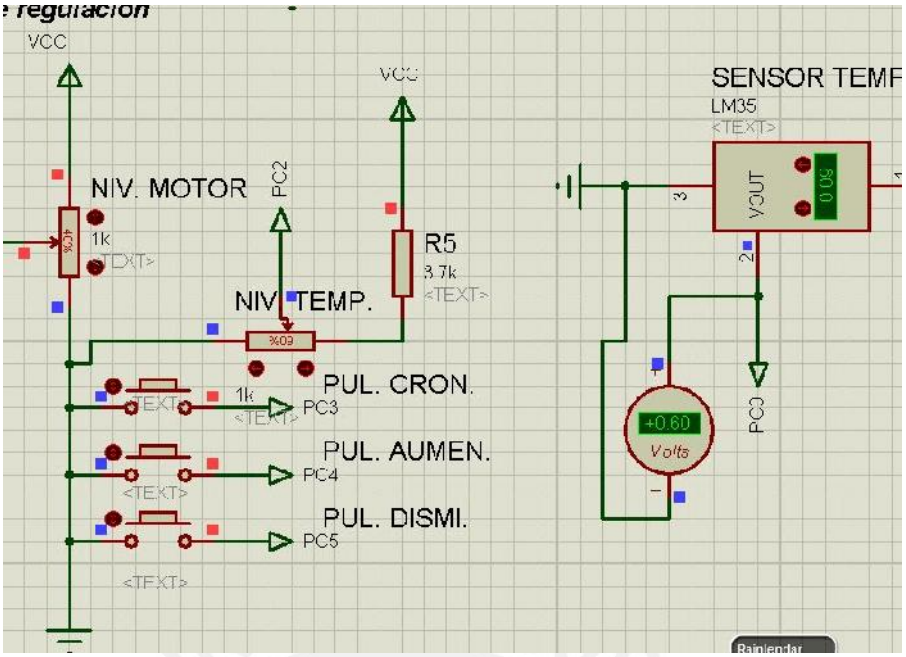

Figura A2.18. Regulación del sensor para ver el cambio del sistema.

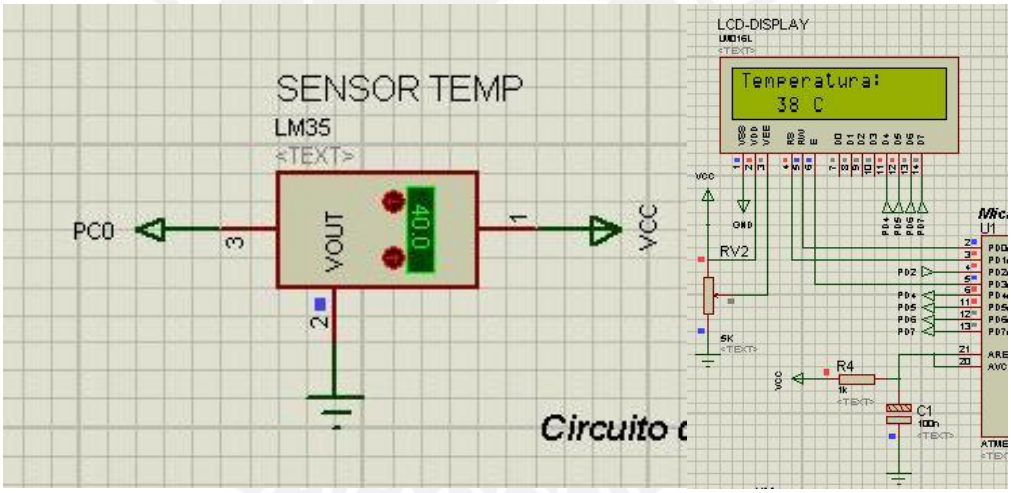

Figura A2.19. Variación de temperatura mostrada en el display.

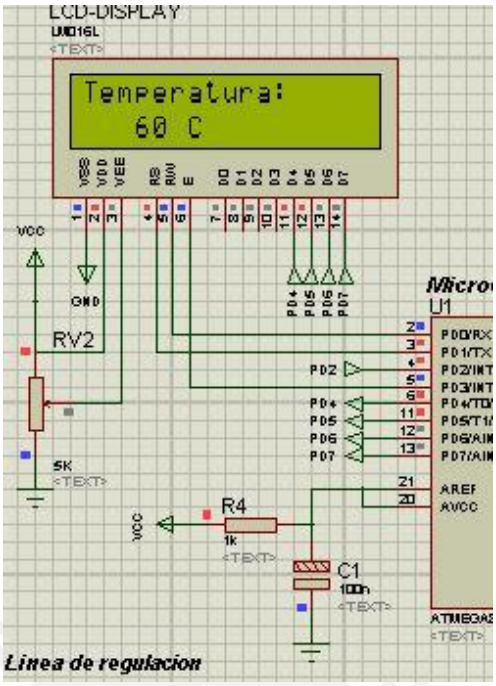

Figura A2.20. Temperatura alcanzada.

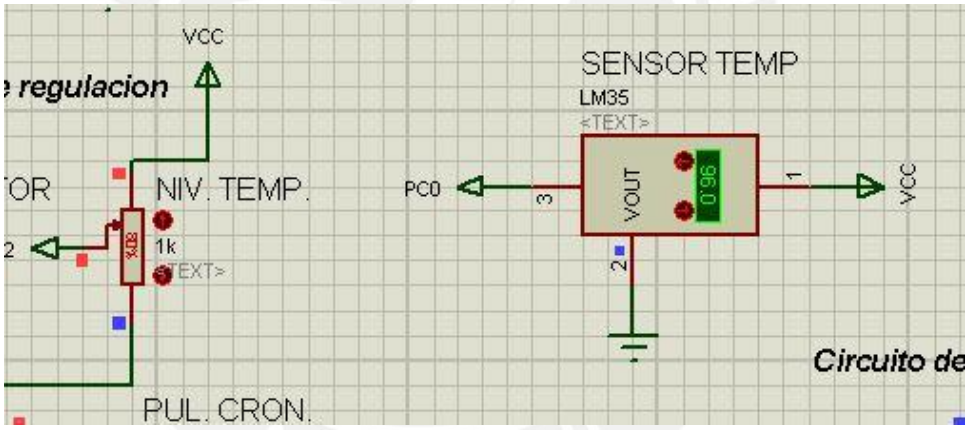

Figura A2.21. Regulación de temperatura 90°C en el potenciómetro.

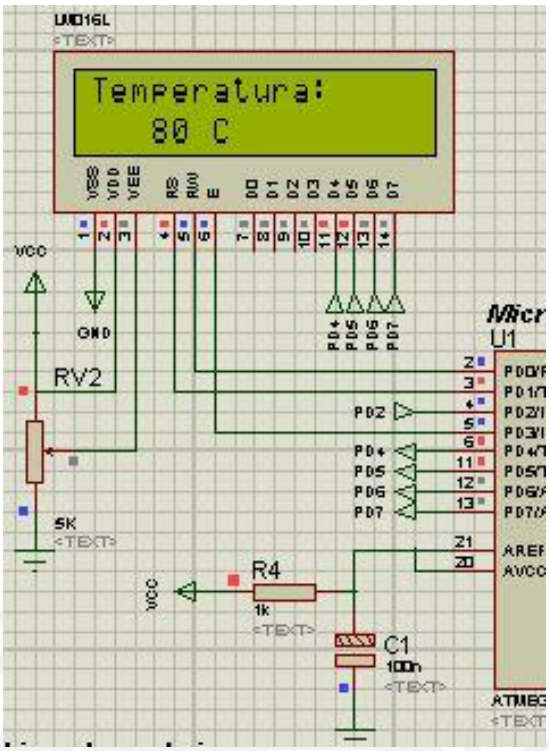

Figura A2.22. Aumento de la temperatura a regular.

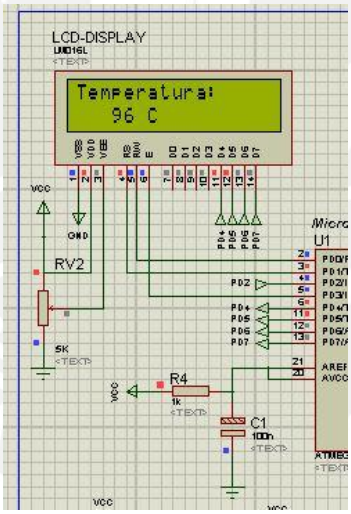

Figura A2.23. Temperatura deseada.

Imágenes prueba de tiempo (cronometro) PROTEUS

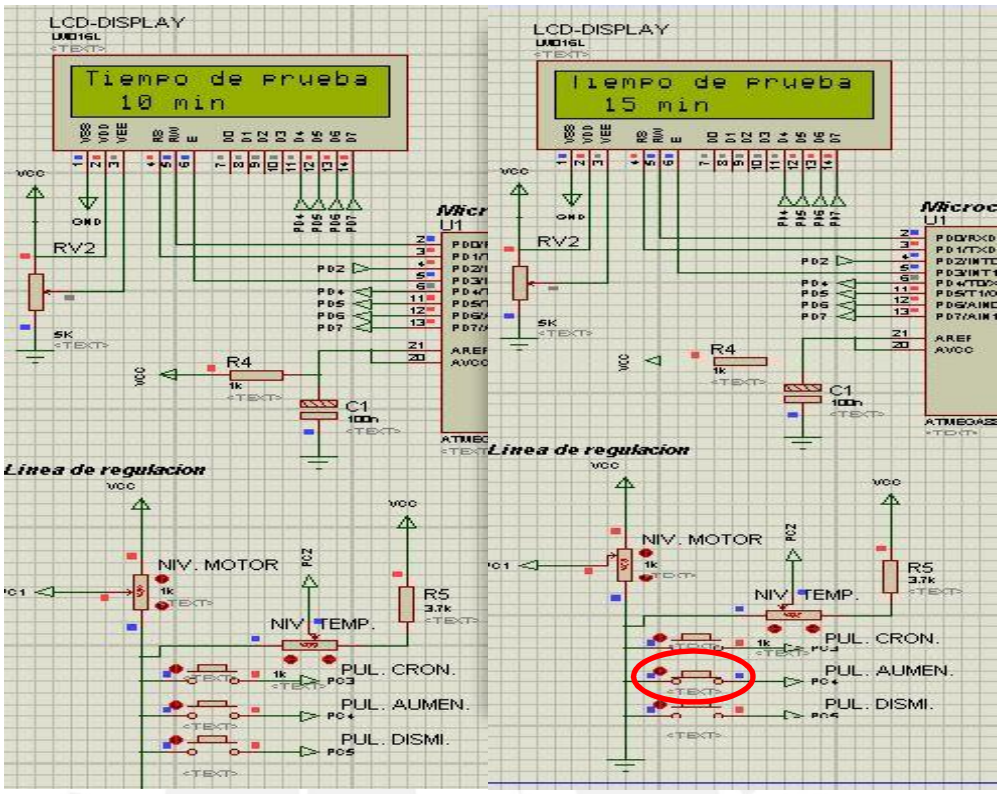

Figura A2.24. Aumento de tiempo.

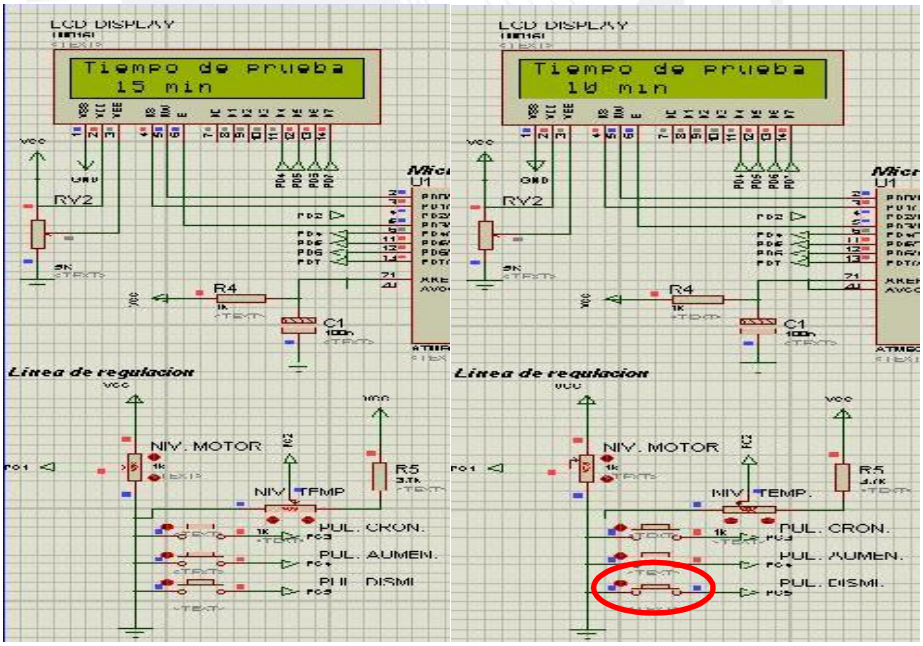

Figura A2.25. Disminución de tiempo.

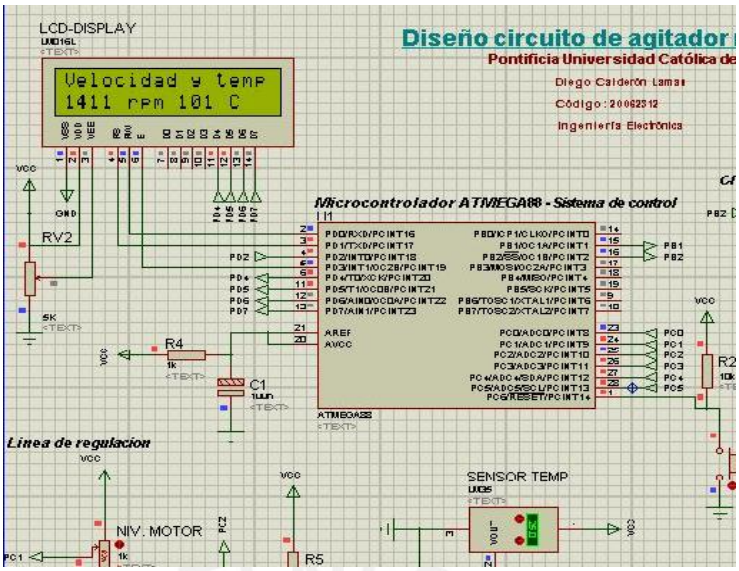

Figura A2.26. Indicador de velocidad y temperatura.

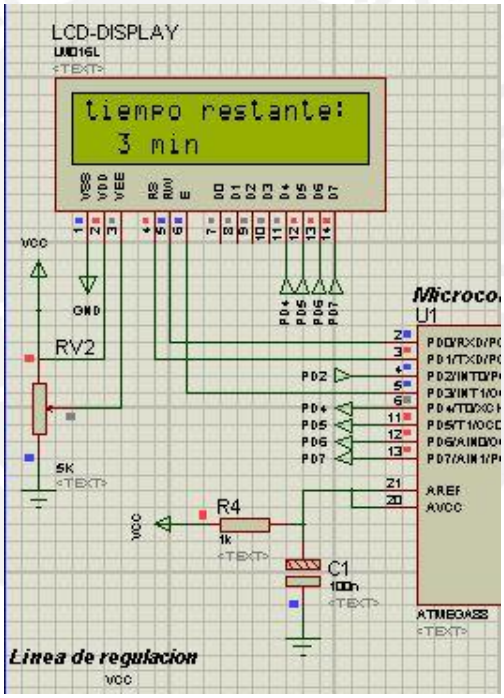

Figura A2.27. Tiempo disminuye hasta los tres minutos.

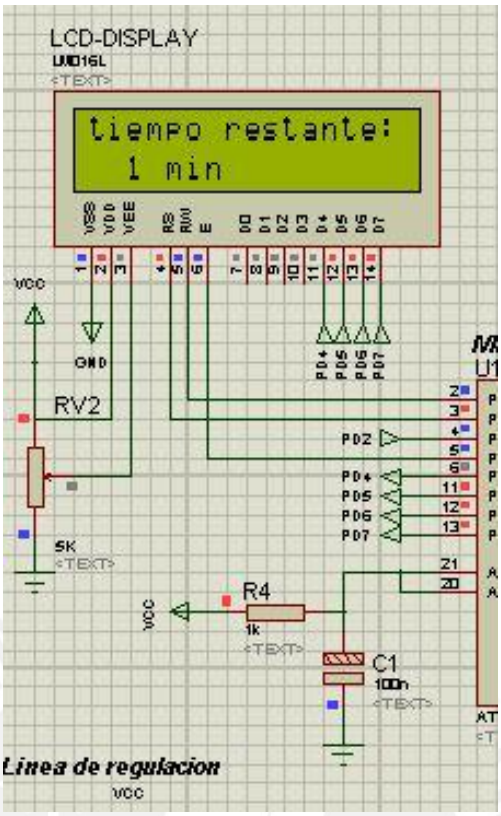

Figura A2.28. Tiempo disminuye hasta el minuto.

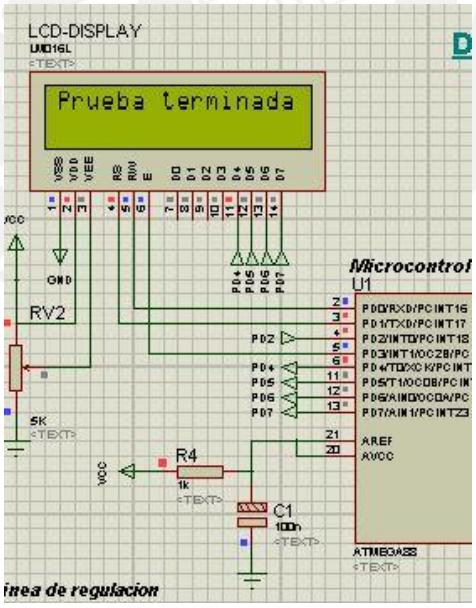

Figura A2.29. Tiempo de prueba finalizado.