PROYECTO FINAL
SISTEMAS DE CONTROL 2 – CONTROLADOR MOTOR DE POTENCIA

José Bernardo Roldán Torres – Daniel Alejandro Sánchez
Universidad Sergio Arboleda
jose.roldan@correo.usa.edu.co – danielalej.sanchez@correo.usa.edu.co

1. RESUMEN

En este proyecto se realizó la caracterización de un motor DC de potencia media (5w-20w) y el diseño de su controlador PI digital mediante la implementación de este en un sistema embebido de la familia atmega.

2. INTRODUCCIÓN

Para realizar el controlador de un sistema de primer orden como lo es un motor DC, debido a que posee un solo elemento que almacena energía que es el embobinado del mismo, se debe partir de la siguiente ecuación:

$$G(s) = \frac{K * e^{-sTo}}{\tau * s + 1}$$

Para diseñar el polinomio del sistema deseado se deben partir de unos parámetros de desempeño que son los que se muestran a continuación:

% Overshoot < 15%

Setling time < 0.6s

El tiempo de muestreo a utilizar lo describe la siguiente ecuación:

$$Ts \leq \frac{tr}{10}$$

Para ello se necesita saber el tiempo de respuesta mediante la identificación del sistema en malla cerrada.

$$T(s) = \frac{G(s) * C(s)}{1 + G(s) * C(s)}$$

Con el overshoot y el settling time se llegó a los valores de ζ =0.52 y Wn= 15.38 con lo cual se pudo hallar el tiempo de respuesta:

$$tr = \frac{1.8}{Wn}$$

Tr = 0.25 s

Dado que el tiempo de respuesta es de 0.25s, el tiempo de muestreo debe ser menor o igual a 25 ms.

Ts=10 ms

La ecuación que describe el controlador PID a diseñar es la siguiente:

$$PID(s) = Kp + \frac{Ki}{s} + Kd * s$$

3. MATERIALES UTILIZADOS

- Microcontrolador atmega.

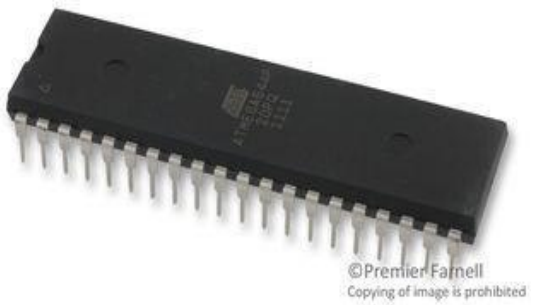

Imagen 1. Microcontrolador atmega644A.

Resolución del ADC = 10 BITS
2 timers de 8 bits y uno de 16 bits.
Frecuencia de operación de hasta 8 MHz.
Frecuencia de PWM de hasta 40 KHz.

- Motor DC de potencia media

Imagen 2. Motor DC 5w.

- Sensor de pulsos y encoder.

Imagen 3. Kit de encoder y sensor.

- MOSFET IRFZ44

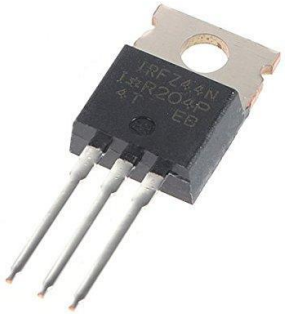

Imagen 4. Transistor Mosfet.

4. PROCEDIMIENTO

- Primero se procedió a caracterizar la planta (el motor en sí) con la obtención de sus rpm mediante un sensor de pulsos conectado al microcontrolador y además su voltaje mediante el ADC del mismo.

- A partir de la ecuación de la planta se hacen los cálculos del controlador PID para los parámetros de desempeño escogidos.

- Se hace la simulación en el software Matlab para mirar el comportamiento del sistema con el controlador y revisar Que cumpla los parámetros de desempeño.

- Se realiza la discretización de la planta mediante los métodos de backward, forward y tustin, y se evalúan en Matlab para corroborar que tienen el desempeño deseado en el sistema con malla cerrada.

- Fue necesaria una etapa de acondicionamiento de señales, debido a que, para la medición del voltaje, el ADC del microcontrolador no soporta más de 5v para lo cual en principio se utilizó un divisor de voltaje.

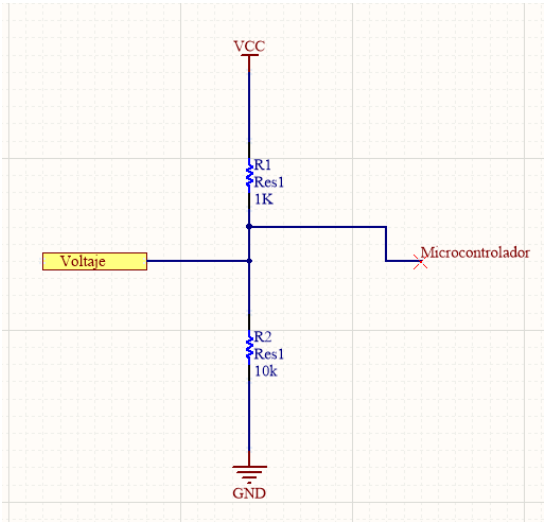

Imagen 5. Divisor de voltaje.

- Para la caracterización de la etapa de potencia con la que se maneja la señal de pwm del motor, se implementa un Mosfet que soporta hasta 40 A con el objetivo de poder manejar las altas corrientes del motor.

- Se procede a embeber el controlador en el sistema embebido escogido mediante las ecuaciones que describen a las constantes PI halladas.

5. ANÁLISIS DE RESULTADOS

- Caracterización de la planta

Los resultados obtenidos para la ecuación del sistema son los siguientes:

$\tau = 0.2$ s

K= 4.4*10^6

Ts = 10 ms

To = 400 ms (tiempo muerto)

To=50ms (tiempo muerto con ajuste fino)

Ecuación hallada:

$$G(s) = \frac{4.4 * 10^6 * e^{-s*0.4}}{0.2 * s + 1}$$

Modelo de la planta en simulink:

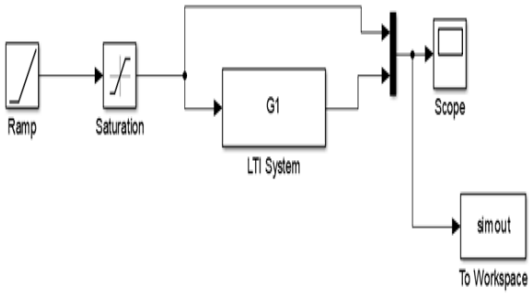

Figura 1. Planta en simulink.

Aplicando un escalón unitario, la respuesta del sistema es la siguiente:

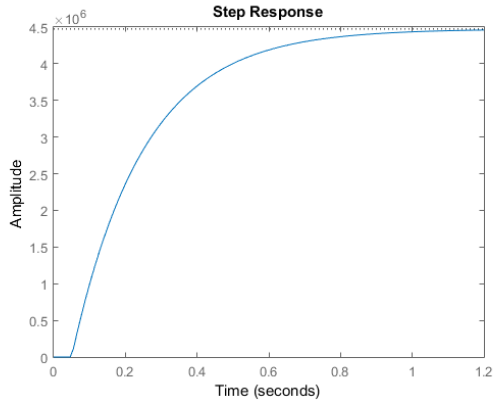

Figura 2. Respuesta a un escalón de la planta.

Se procedió a realizar varias pruebas con tiempos de muestreo diferentes y mirar la entrada y la salida de la planta.:

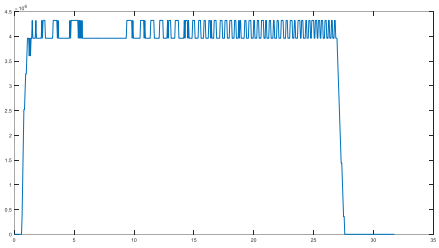

Figura 3. Sistema en malla abierta con tiempo de muestreo de 50 ms.

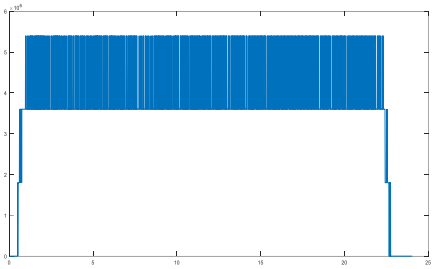

Figura 4. Sistema en malla abierta con tiempo de muestreo de 10 ms.

A partir de la ecuación obtenida de manera práctica, se compara con los resultados generados desde simulink. Comparando ambas salidas y entradas se concluye que se debe realizar un ajuste fino a la ecuación para que reducir el error entre los resultados.

Con el ajuste realizado:

$$G(s) = \frac{4.474 * 10^6 * \ e^{-s*0.05}}{0.2 * s + 1}$$

La señal de entrada obtenida después de realizar el ajuste fino es la siguiente.

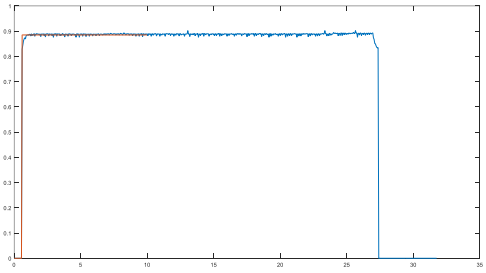

Figura 5. Señal de entrada practica vs teórica.

Haciendo zoom, para visualizar mejor el seguimiento de la entrada práctica respecto a la entrada teórica es la siguiente:

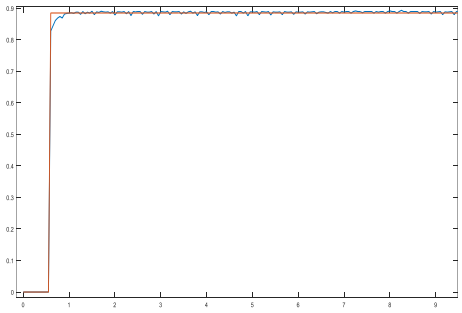

Figura 6. Señal de entrada practica vs teórica.

La respuesta obtenida en la salida después del ajuste fino es.

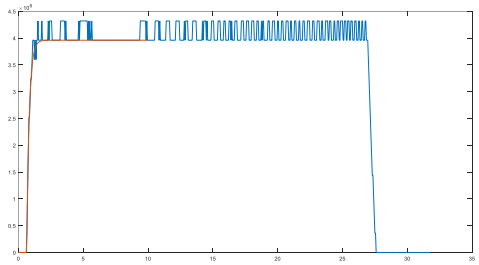

Figura 7. Señal de salida practica vs teórica.

Realizando un zoom:

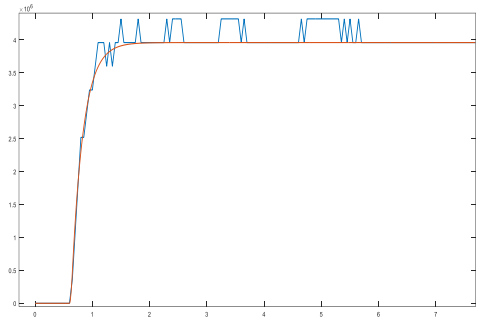

Figura 8. Señal de salida practica vs teórica.

- Realimentación unitaria en malla cerrada con controlador PI simulado.

Utilizando la herramienta pid tool de Matlab se obtuvieron las constantes del controlador PI deseado:

Kp = 3.7721*10^-7

Ki = 2.2902*10^-6

K=4.474*10^6

Tao=0.2

To=0.05

$$G(s) = \frac{K(Kp * s + Ki) * e^{-sTo}}{\tau\, s^2 + s + K(Kp * s + Ki)}$$

Así se obtiene en la simulación con el controlador:

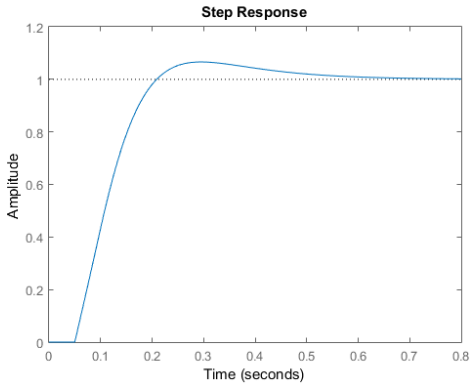

Figura 9. Realimentación unitaria del sistema.

- Ecuaciones del sistema en malla cerrada discretizadas:

Backward:

$$G(z) = \frac{Z^{-5}(\,2.182 * 10^5)}{(z - \,0.9512)}$$

Forward:

$$G(z) = \frac{Z^{-5}(\,1.1 * 10^5 * z \,+\, 1.082 * 10^5)}{(z - \,0.9512)}$$

Tustin:

$$G(z) = \frac{Z^{-5}(\,1.091 * 10^5 * z \,+\, 1.091 * 10^5)}{(z - \,0.9512)}$$

Modelo de la planta con PI continúo en simulink:

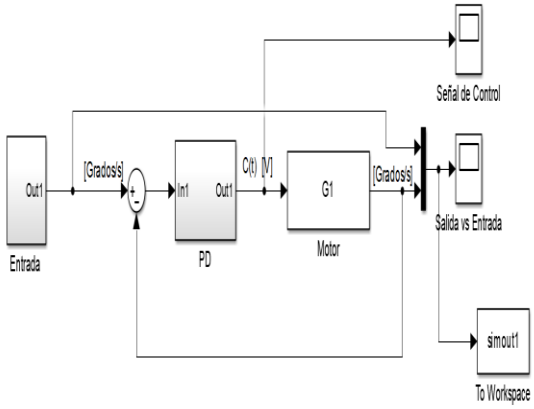

Figura 10. Controlador PI en simulink

Modelo de la planta continúa con PI discreto por los tres métodos en simulink:

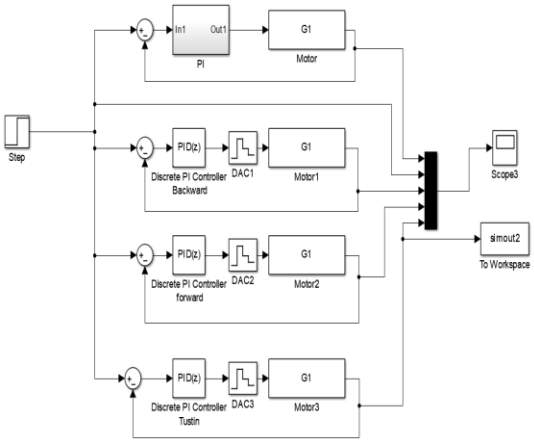

Figura 11. Controlador PI y sus discretizaciones en simulink.

- Gráficas en discreto del sistema en malla cerrada.

BACKWARD:

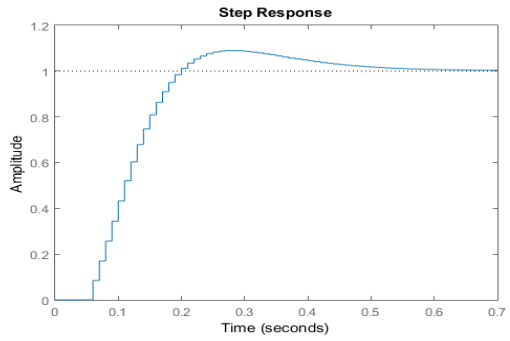

Figura 12. Discretización por método backward del sistema.

FORWARD:

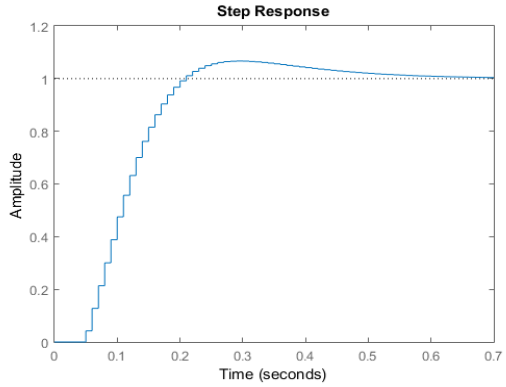

Figura 13. Discretización por método forward del sistema.

TUSTIN:

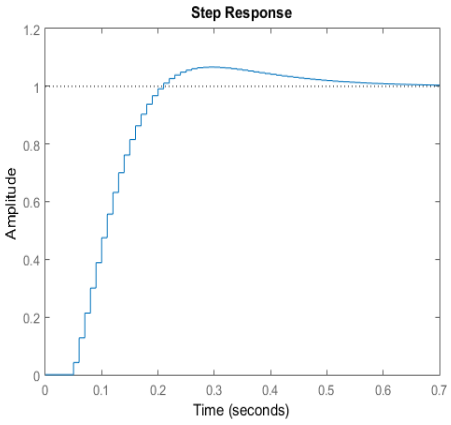

Figura 14. Discretización por método tustin del sistema.

- Diseño de la etapa de potencia

Para el diseño de la etapa de potencia se tuvo en cuenta el hecho de que no será requerido que el motor gire en las dos direcciones, es decir, el diseño solo debe garantizar como mínimo la entrega de 1 Amperio y un máximo de 5 Amperios. Para el cumplimiento de las especificaciones se eligió el MOSFET IRF44Z, el cual soporta un máximo de 50 Amperios a temperatura de 25 °C, según fabricante.

Inicialmente se tenía diseñado un circuito que constaba del MOSFET y dos transistores BJT, uno pnp y el otro npn, con el fin de garantizar que el encendido y apagado del MOSFET fuese rápido, para que este no se calentara. Ya que, si el voltaje gate no supera el voltaje de ruptura, el MOSFET se encontrará en la zona óhmica y actuará como resistencia disipando calor. El esquemático del circuito es el siguiente:

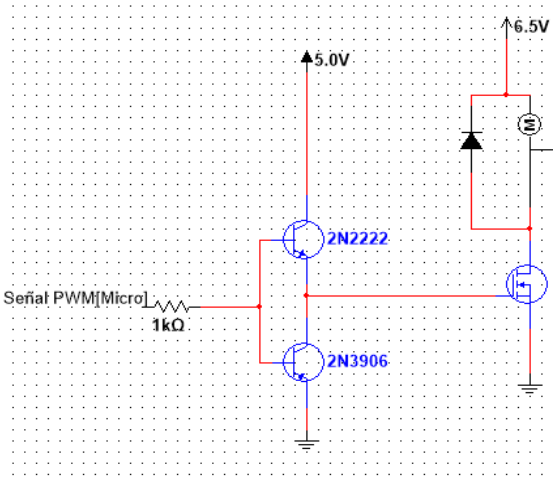

Figura 15. Etapa de potencia.

Sin embargo, realizando pruebas sin el par de transistores BJT's se observó que no eran necesarios en este caso. Por lo que la etapa de potencia quedó reducida a lo siguiente:

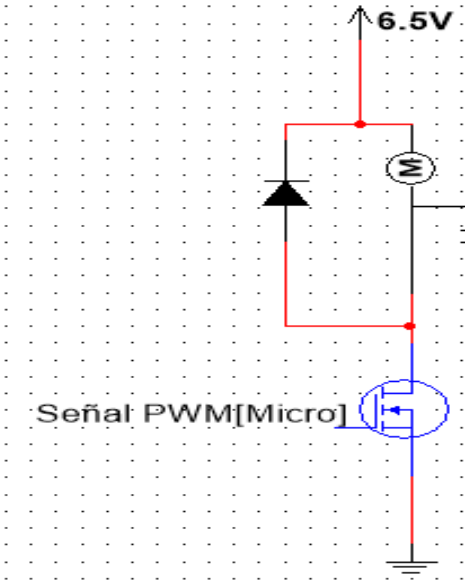

Figura 15. Implementación del mosfet.

Es necesario poner un diodo en paralelo al motor de esa manera como se ve en el esquemático para evitar que la FEM producida por el motor genere una tensión que queme el MOSFET.

Con el diseño establecido, se iniciaron pruebas para conocer el comportamiento de la etapa de potencia respecto a distintas entradas, que serán dadas por el ciclo útil de la señal PWM. Este procedimiento se tuvo que hacer 2 veces, ya que primero se utilizó una protoboard y después el montaje se realizó en una baquela.

Los resultados obtenidos en protoboard son los siguientes:

Registro 8 bi	V_out
40	0,353
70	1
90	1,54
110	1,97
125	2,35
150	2,9
170	3,38
190	4,09
200	4,165
230	4,66
240	4,9
250	4,98
255	5,2

Tabla 1. Resultados en protoboard.

Con estos datos se puede sacar un equivalente a una recta, obteniendo su ecuación:

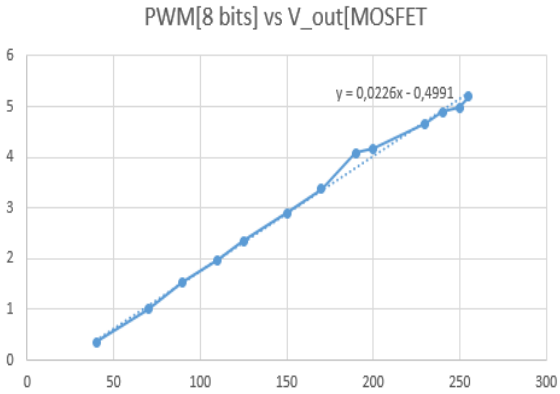

Figura 16. Señal pwm vs voltaje en protoboard.

Al pasar el circuito en la baquela se comprobó que al mejorar las conexiones del circuito se generan menos pérdidas.

Registro 8 bits	V_out
40	0,414
70	1,08
90	1,53
110	2,01
125	2,32
150	2,89
170	3,33
190	3,77
200	4,02
230	4,74
240	4,85
250	5,03
255	5,41

Tabla 2. Resultados en baquela.

La ecuación de la recta para el circuito en la baquela es la siguiente:

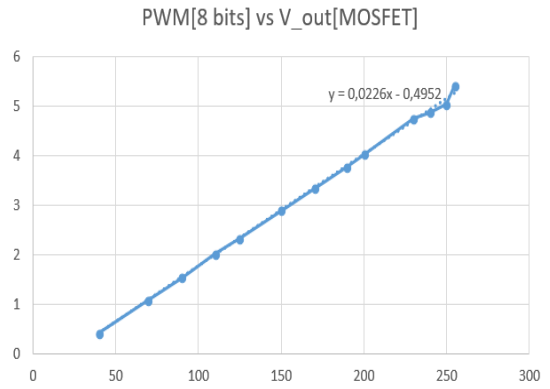

Figura 17. Señal pwm vs voltaje en baquela.

- Implementación del PI en el microcontrolador

Para embeber el siguiente sistema de control en malla cerrada en el microcontrolador ATMEGA644A.

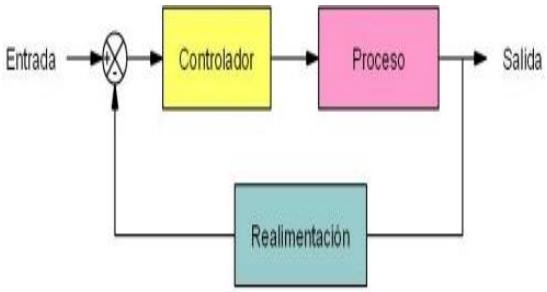

Figura 18. Sistema de control en malla cerrada.

Como primer paso se fijó el tiempo de muestreo que debe ser menor a la décima parte del tiempo de subida del motor, es decir, debe ser menor a 20 ms.

El tiempo de muestreo se estableció en 10 ms. a causa de las limitaciones que posee el sensor encoder en cuanto a su tiempo de respuesta.

Los módulos a utilizar de este microcontrolador son 3, el módulo de ADC, TIMER, y USART.

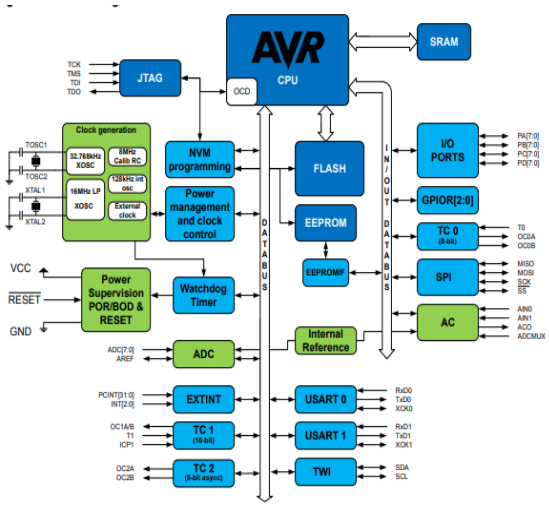

Figura 19. Arquitectura avr del microcontrolador.

En cuanto al ADC, se utiliza un canal para conocer el voltaje que le ingresa al motor.

```
void adc_init(){
    //ADC  CON INTERRUPCIONES Fmax Vref
    ADMUX|=(1<<REFS0);
    ADMUX&=~(1<<REFS1);
    ADCSRA|=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)|(1<<ADIE);

}
```

De acuerdo a las necesidades del proyecto, se utilizan los 3 timers que posee el microcontrolador, uno para tener una base de tiempo fijada a 1 ms, otro para recibir los pulsos enviados del sensor encoder y por último el timer que se encarga de generar la señal PWM.

```
void pwm_init(){
    //timer 2
    TIMSK2|=(1<<TOIE2);
    TCCR2B&=~(1<<CS21);
    TCCR2B|=(1<<CS20);
    TCCR2B&=~(1<<CS22);
    OCR2A=62;
    DDRD|=(1<<PD7); // SALIDA PWM A
    TCCR2A|=(1<<COM2B1);
    TCCR2A&=~(0<<COM2B0);
    TCCR2A|=(1<<COM2A1);
    TCCR2A&=~(0<<COM2A0);
    TCCR2A|=(1<<WGM20)|(1<<WGM21);
    //TCCR2B|=(1<<WGM22);
}
```

```
  void counter1_init(){
      // TIMSK1|=(1<<TOIE1); // INTERRUPCION
      TCCR1B&=~(1<<CS10); //falling edge
      TCCR1B|=(1<<CS12); // falling edge
      TCCR1B|=(1<<CS11); // falling edge
      TCNT1H=0;
      TCNT1L=0;
      DDRB&=~(1<<PB1);
      PORTB|=(1<<PB1);
      }
```

```
void TIMER_init(){
    TIMSK0|=(1<<TOIE0);
    TCCR0B|=(1<<CS00);
    TCCR0B&=~(1<<CS02);
    TCCR0B|=(1<<CS01);
    TCNT0=130;
}
```

Por último, el módulo para generar una comunicación serial(RS232) y lograr enviar datos al computador.

```
void UART0_init(unsigned int brr){
    UBRR0H=(unsigned char)(brr>>8);
    UBRR0L=(unsigned char)brr;
    UCSR0B|=(1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0); //rx y tx enable
    UCSR0C=(0<<UMSEL00)|(0<<UMSEL01)|(0<<UPM00)|(0<<UPM01)|(3<<UCSZ00)|(0<<USBS0);
    //modo asincrono-bit de paridad-8bits- bit de parada

}
```

Teniendo los módulos perfectamente funcionando, se programa la lógica para implementar el controlador PI.

```
void Controller(){
    ErrorA=(float)setpoint-(float)pulsos*Conversion;
    //IntError=IntError+((ErrorA-LastError)/2)*Ts;
    IntError=IntError+(ErrorA)*Ts;
    out=Kp*ErrorA+Ki*IntError;
    out=(out+0.4952)/0.0226 ;
    if(out>=255){
        out=254;
    }
    LastError=ErrorA;
    OCR2A=out;
}
```

Por último se hace el algoritmo encargado de recibir los pulsos del sensor encoder, paso siguiente de generar la señal de control y finalmente de enviar los datos al computador, esta operación cada 10 ms.

```
    while (1){
        adc_read(0);
        if(read_safe()==0){
            pulsos=TCNT1;
            Controller();
            sprintf(temp0,"%u,%u\r\n",pulsoref,pulsos);
            put_str(temp0);
            contadores[T_motor]=10;
            TCNT1=0;
        }
    }
}
```

Ahora bien, se procede a realizar las pruebas en físico dándole varios valores de referencia.

Inicialmente se desea que el motor siga 10 pulsos, es decir, 18.000.000 grados/segundo.

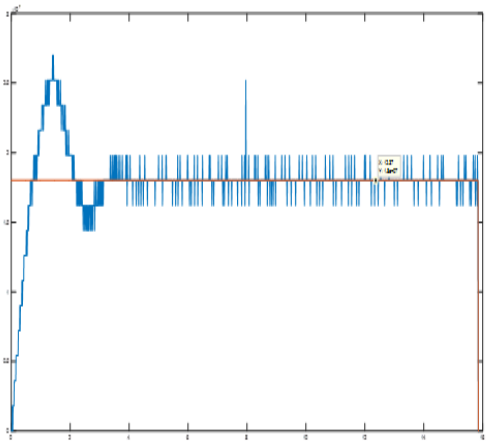

Figura 20. Primera prueba.

Se realizó la misma prueba con 8 pulsos que equivalen a 14.400.000 grados/segundo.

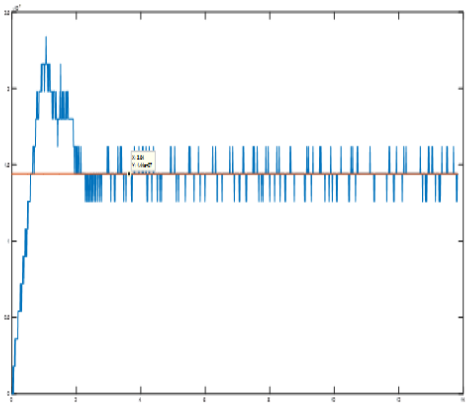

Figura 21. Segunda prueba.

Se realiza otra prueba con una referencia en 5 pulsos, es decir, 9.000.000 grados/segundo.

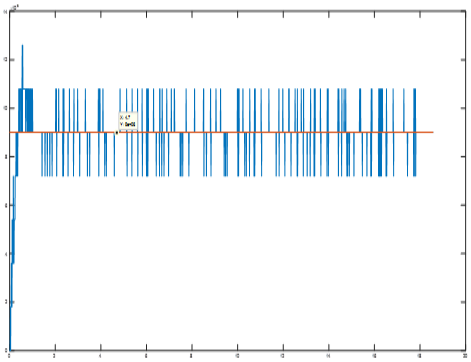

Figura 22. Tercera prueba.

Como última prueba se establece como referencia 3 pulsos, ósea 5.400.000 grados/s.

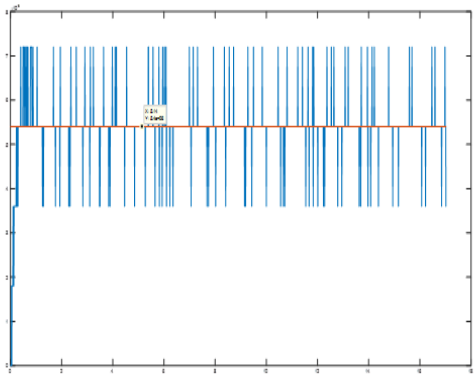

Figura 23. Prueba final.

Analizando de manera gráfica los resultados obtenidos con referencia fijada en 14.400.000 grados/segundo, obtenemos que el tiempo de establecimiento aproximado esta en 2.05 segundos y un porcentaje de sobre impulso de 33.3%, donde el pico máximo que se obtiene es 2.16x10^7.

Después se procedió a hacer varias perturbaciones sobre el motor obteniendo las siguientes gráficas, inicialmente con una referencia en 19.800.000:

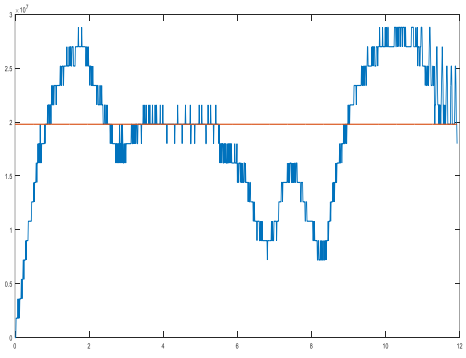

Figura 24. Prueba con varias perturbaciones.

Después se fijó como referencia 10.800.000 grados/segundos

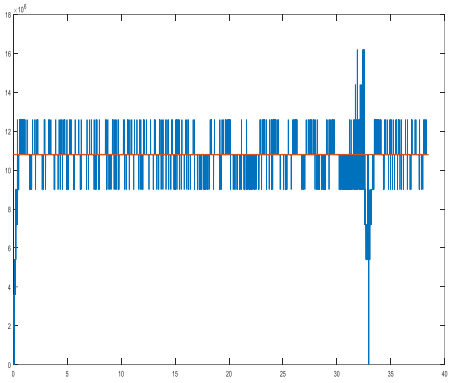

Figura 25. Primera prueba con perturbación.

Realizando un zoom en el área donde se presenta la perturbación con el fin de analizar el tiempo que tarda en retomar la referencia:

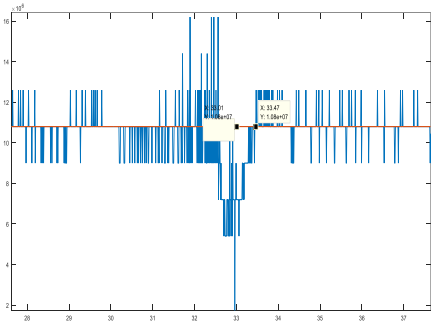

Figura 26. Segunda prueba con perturbación.

De acuerdo con la gráfica, el momento donde se deja de ingresar la perturbación es en 33.01 segundos y el sistema regresa a su valor de referencia en 33.47 segundos, es decir, el sistema corrige el error en 0.47 segundos.

6. CONCLUSIONES

- Al tener una constante integral mayor que la constante proporcional, el motor tiene una respuesta rápida, pero con un frenado más óptimo.
- La velocidad de respuesta del sensor encoder utilizado no permite utilizar al máximo el motor en cuanto a su velocidad ya que después de 7 voltios aplicados en el motor, este gira tan rápido que el sensor pierde la capacidad de enviar pulsos al microcontrolador.
- A pesar de que el controlador embebido funciona correctamente, no se cumple el tiempo de establecimiento requerido del sistema, porque el enconder cuenta con una baja resolución, lo cual causara perdidas de información respecto al ángulo del motor, en otras palabras, se crean tiempos muertos en los que no se conocer la posición del motor, esto afectara el desempeño del controlador.
- Fue de gran importancia el procedimiento donde se caracterizó el comportamiento del MOSFET, ya que se obtuvo la relación registro respecto a voltaje, de aquí que se haya podido de manera correcta convertirla señal de control generada por el PI a un equivalente de ciclo útil en la señal PWM.
- Es importante conocer y tener claro la manera correcta de establecer el tiempo de muestreo para la implementación del controlador digital ya que el tiempo de respuesta del sistema en malla abierta es diferente al tiempo de respuesta que posee el sistema en malla cerrada.