Anexo C.

Implementación Hardware y Software de control de posición, velocidad y par de un motor DC de imán permanente.

Cristian Julián Solarte Rosas
Jhon Edinson Muñoz Ordoñez

Director: Mg. Francisco Franco.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control.
Popayán, marzo de 2015.

Tabla de contenido.

1. Introducción.

El presente documento describe el proceso de implementación hardware y software necesario para controlar la posición, el par y la velocidad de un motor de DC con caja reductora cuyos valores nominales son: W_m =200rpm, T_m=1.8Kg-cm,V_a=6v con caja reductora de relación 100-1.

2. Implementación Hardware

La estructura hardware en esta guía es general para motores de corriente directa de imán permanente, pero se debe tener claro que este circuito es para un motor de baja potencia y los elementos se seleccionaron para dicho motor.

Si la aplicación es para un motor de mayor potencia se deben considerar las capacidades de los controladores para manejar los niveles de tención y corriente que exija el motor.

2.1. Diagrama general del controlador.

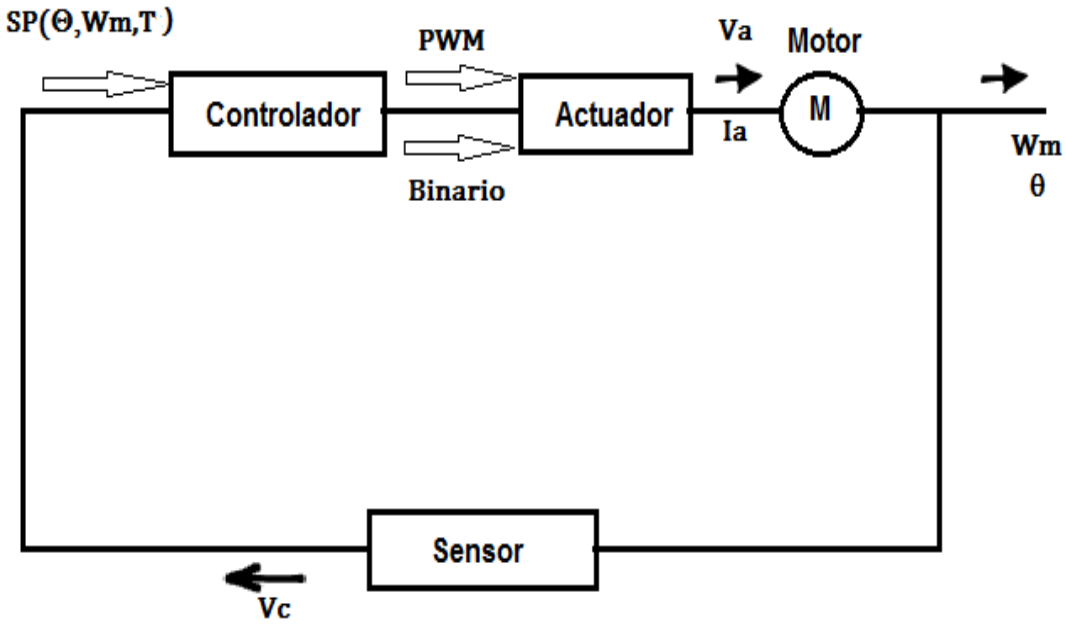

Figura 1. Diagrama general de controlador.

donde:

SP(θ,Wm,T) es el valor deseado de la variables de estado, θ es el valor asociado a la posición, Wm es el valor asociado a la velocidad mecánica del motor, T es el par del motor.

Los componentes del esquema general se describen a continuación así como los dispositivos usados para su implementación.

2.1.1. Módulo Controlador:

Es el encargado de realizar el control del actuador por medio de una señal PWM para controlar posición, velocidad y una variable de valor binario para realizar el control del par.

Controlador: el controlador escogido para desarrollar el controlador de velocidad, posición y par es una tarjeta Arduino MEGA 2560R3.

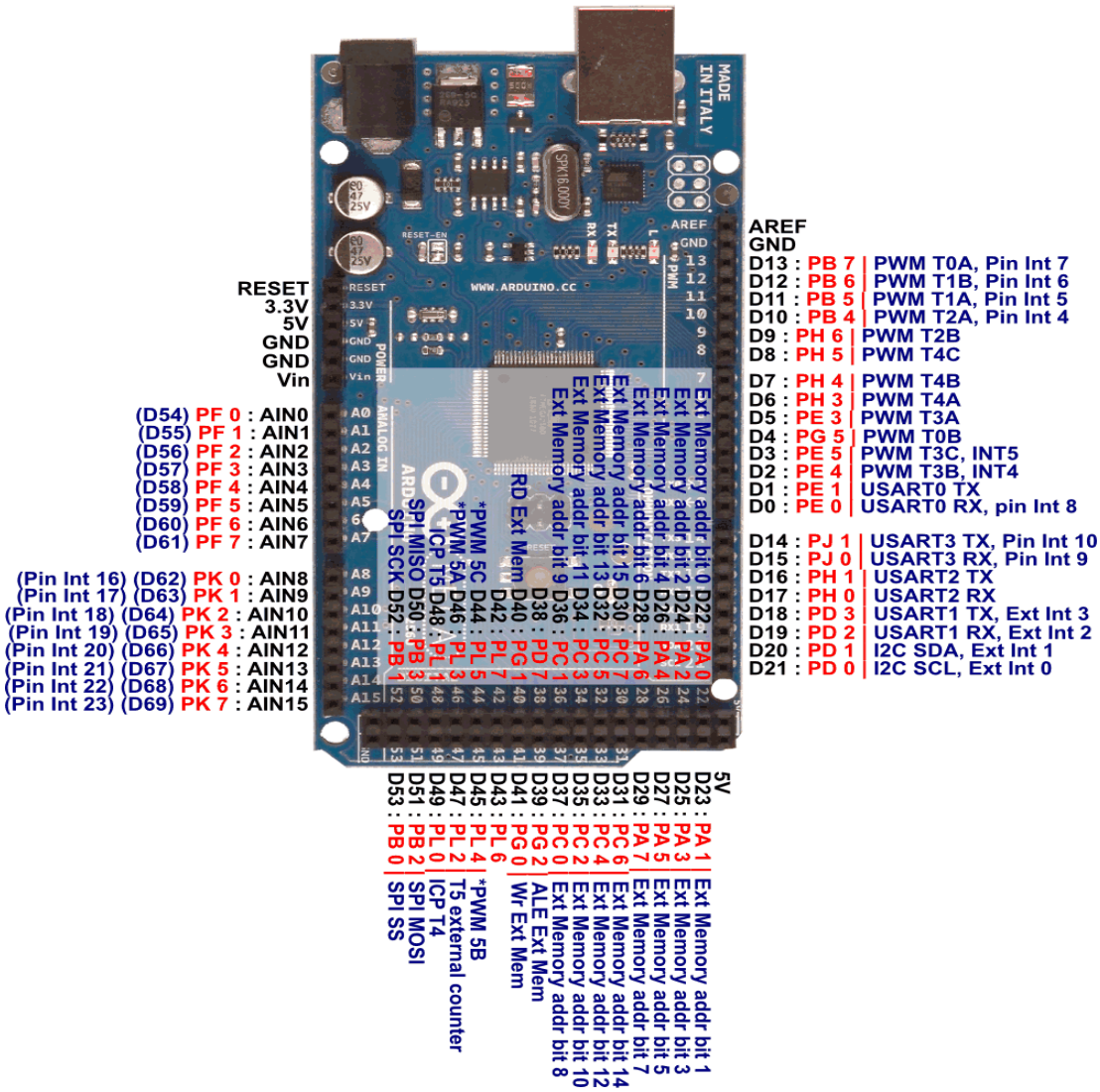

Figura 2. Esquema Físico y Diagrama de Pines Arduino Mega 2560 R3. Fuente: http://clasesdearduinoinem.blogspot.com/.

2.1.2. Módulo actuador:

Es el encargado de suministrar los valores requeridos de voltaje **Va** y corriente **Ia**, al motor de acuerdo a las señales emitidas por el controlador.
En nuestro caso el modulo actuador está compuesto por:

Driver Shield arduino L298N. Es un puente H dual para motores DC basado en el L298N, con 2 terminales de entrada de voltaje, uno diseñado para alimentar con 5v+ y otra que funciona en el rango de 5v a 35v, con un pico de corriente de 2A. Este driver se utiliza para controlar el sentido de giro del motor, así como la posición y la velocidad al igual que mantener protegida la placa Arduino de los picos de corriente.

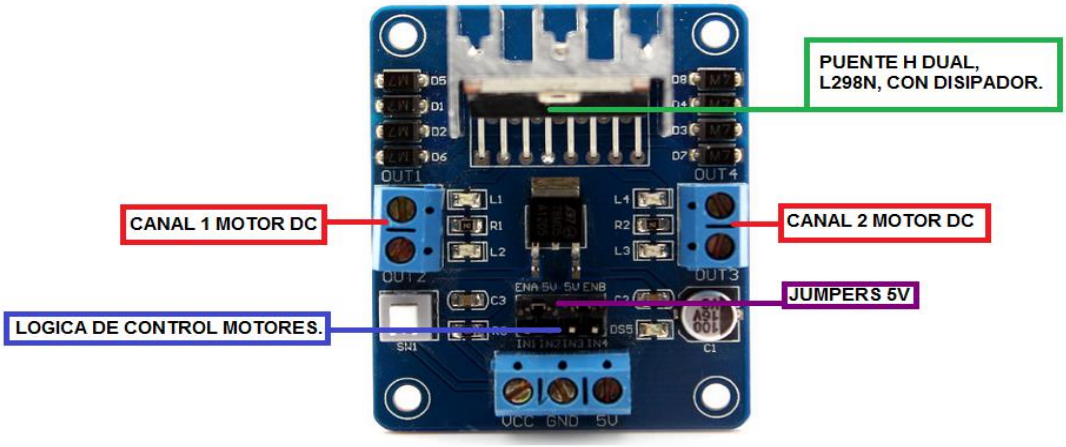

Figura 3. Esquema Físico del Driver L298N.
Fuente: http://clasesdearduinoinem.blogspot.com/.

2.1.2.1. Controlador de corriente diseño usando ICSA100A, potenciómetro digital MCP41050, comunicación SPI.

El control de corriente se implementó de manera electrónica, limitando la corriente antes de entrar en el puente H dual. Se utilizó el ICSA001A que es un módulo que permite controlar la corriente y voltajes de manera independiente por medio de 2 potenciómetros. La ventaja que otorga el módulo es que al controlar la corriente y el voltaje de manera independiente se logra mantener un voltaje constante mientras se varía la corriente de esta manera aprovechando esta característica se controla el par en el motor limitando la corriente y manteniendo el voltaje estable así se controla el par sin afectar la velocidad del motor.

El módulo ICSA001A tiene como entrada un rango de voltajes entre 4.5V y 30V, puede controlar corrientes entre 0.1A y 5A manteniendo en su salida ajustable siempre constantes entre 0.8V y 30V.

Figura 4. Esquema Físico del Módulo ICSA001A.
Fuente: http://clasesdearduinoinem.blogspot.com/.

Es necesario remover el potenciómetro que se encuentra el segmento de ajuste de corriente y cambiarlo por el esquema de la figura 5 que incluye resistencias y un potenciómetro digital, lo que se busca con las resistencias es que con una resistencia de 190K obtener una corriente constante sin el potenciómetro digital de 300mA, de esta manera al insertar y aumentar por medio del puerto SPI la resistencia del potenciómetro digital MCP41050 lograr variaciones de corriente entre 50mA y 300mA, que es el rango de funcionamiento del motor, el esquema circuital se muestra en la figura 5.

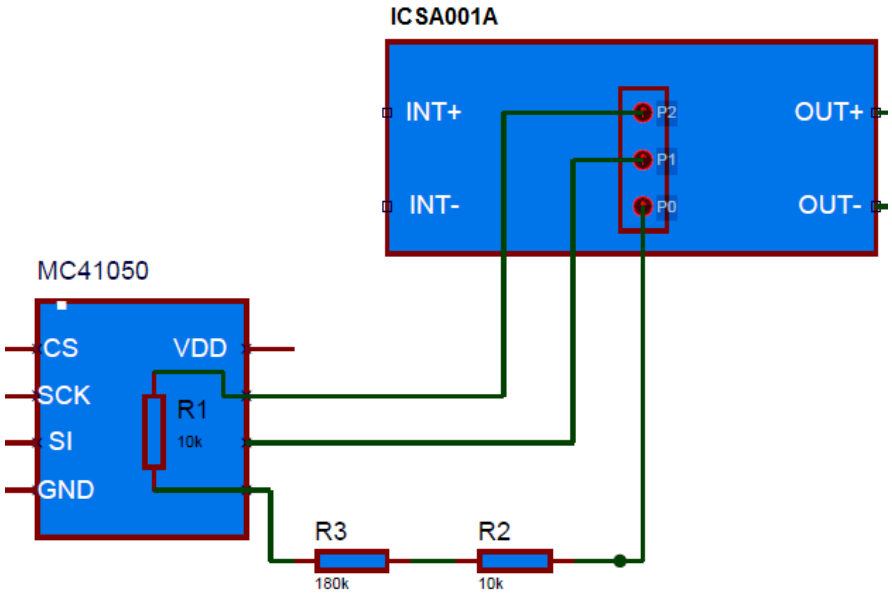

Figura 5. Esquema Controlador de Corriente Implementado.

Finalmente una los dos módulos anteriores y obtenga el esquema circuital mostrado en la figura 6.

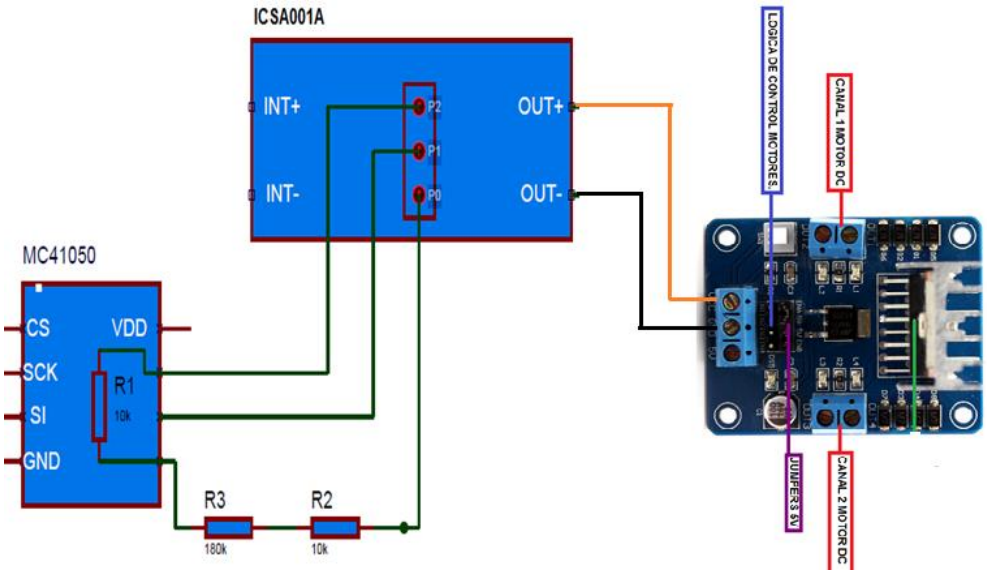

Figura 6. Modulo actuador del sistema de control.

2.1.3. Motor.

Es el elemento controlado y el encargado de entregar las respuestas de velocidad mecánica Wm y posición θ y par T del control del sistema. En nuestro caso el motor seleccionado es un motor de DC con caja reductora cuyos valores nominales son: W_m =200rpm, T_m=1.8Kg-cm,V_a=6v con caja reductora de relación 100-1.

2.1.4. Módulo sensor.

Se encarga de entregar los valores de la posición θ y velocidad mecánica Wm del motor por medio de una señal de voltaje de control Vc.
En nuestro caso el sensor seleccionado es:

2.1.4.1. Sensor de posición (DFR0058), sensor análogo de rotación.

Es un sensor compatible con arduino, está basado en un potenciómetro de precisión el cual puede efectuar hasta 10 vueltas, el control de posición y velocidad se va a caracterizar en Grados y Grados por segundo, sabiendo que la placa arduino asigna valores de 0 a 1024, a un voltaje mínimo y máximo respectivamente, se procede a caracterizar el sensor para que entregue a la interfaz analógica de arduino esta control en grados por segundo esto se logra de la siguiente forma.

El sensor al poseer 10 vueltas se obtienen 3600 grados en total de rotación lograda por el sensor de posición, a su vez existe un voltaje referencial que es el que interpretará arduino en su entrada analógica que asigna un valor entre 0

y 1024 a este rango de voltajes de esta forma se tiene que al voltaje que entrega el potenciómetro de acuerdo a su posición de le debe multiplicar el siguiente valor para que quede interpretado en grados, $\mathrm{Ctegrados} = 3.519$

$$C_{tgrados} = \frac{10*360}{1024} = 3.5156 = 3{,}519 \qquad (1).$$

Descripción física sensor lineal de posición DRF0058.

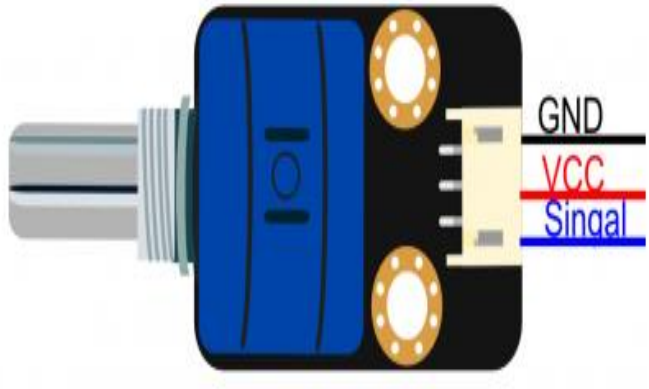

Figura 7. Esquema Físico y Diagrama de Pines DRF0058.
Fuente: http://clasesdearduinoinem.blogspot.com/.

Para obtener el esquema de control una los módulos hasta obtener el esquema ilustrado en la figura 9, es de destacar que el Motor y el sensor deben quedar acoplados como se muestra en la figura 8.

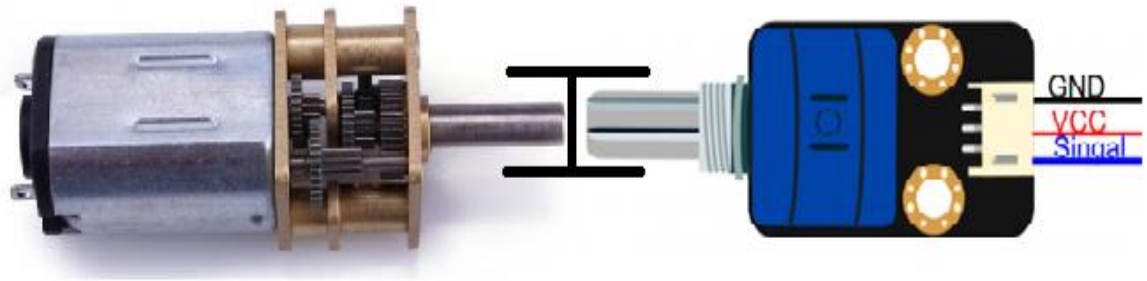

Figura 8. Acople de motor DC de imán permanente con sensor lineal de posición DRF0058.

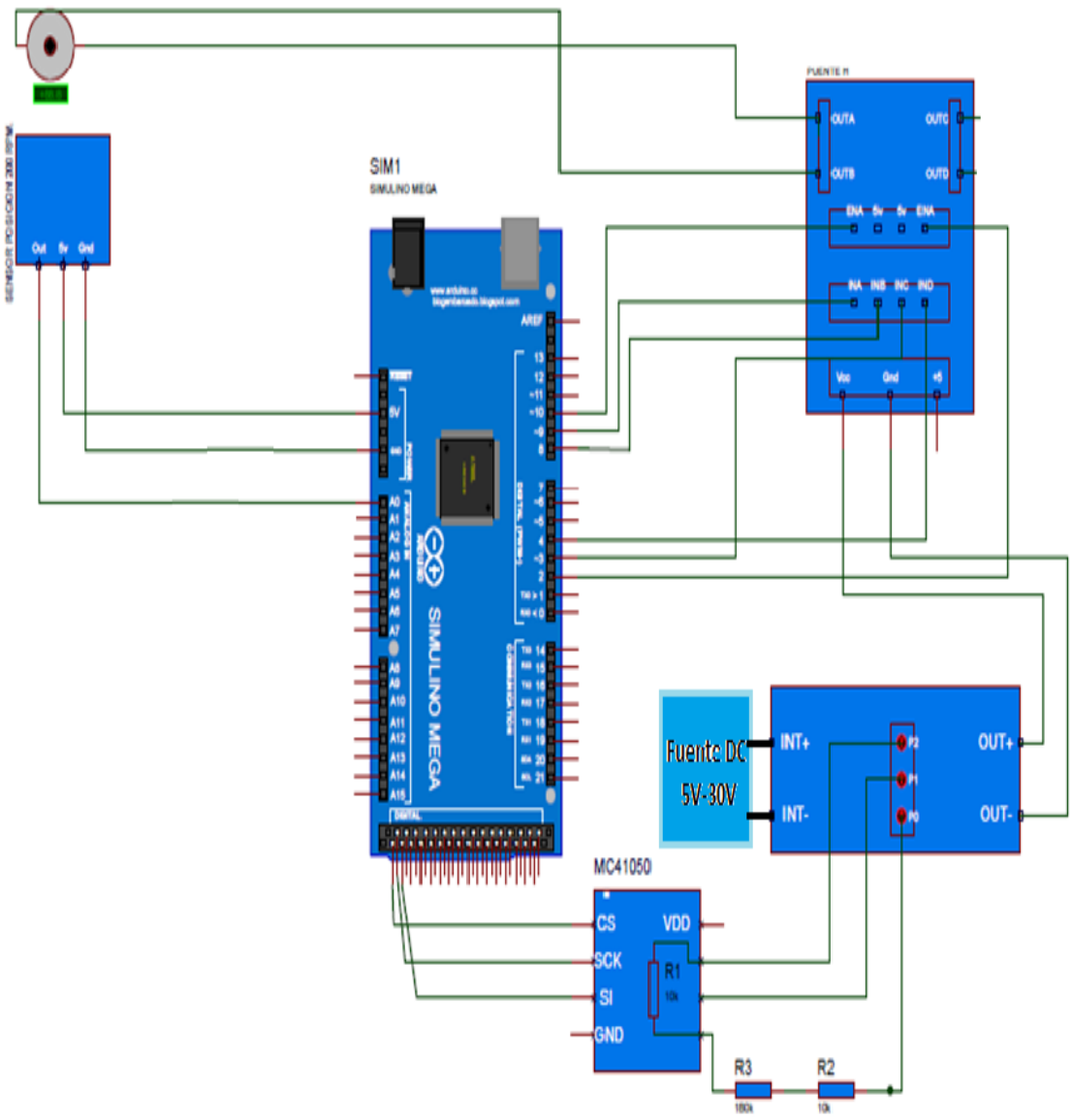

Figura 9. Esquema Controlador Completo.

3. Algoritmos del controlador de velocidad, posición y par de un motor DC de imán permanente.

3.1. Algoritmo realizado para el control de posición.

3.1.1. Diagrama de flujo del controlador PID de posición.

Para desarrollar el hardware del control de posición se tomó como base el diagrama de flujo de la figura 10.

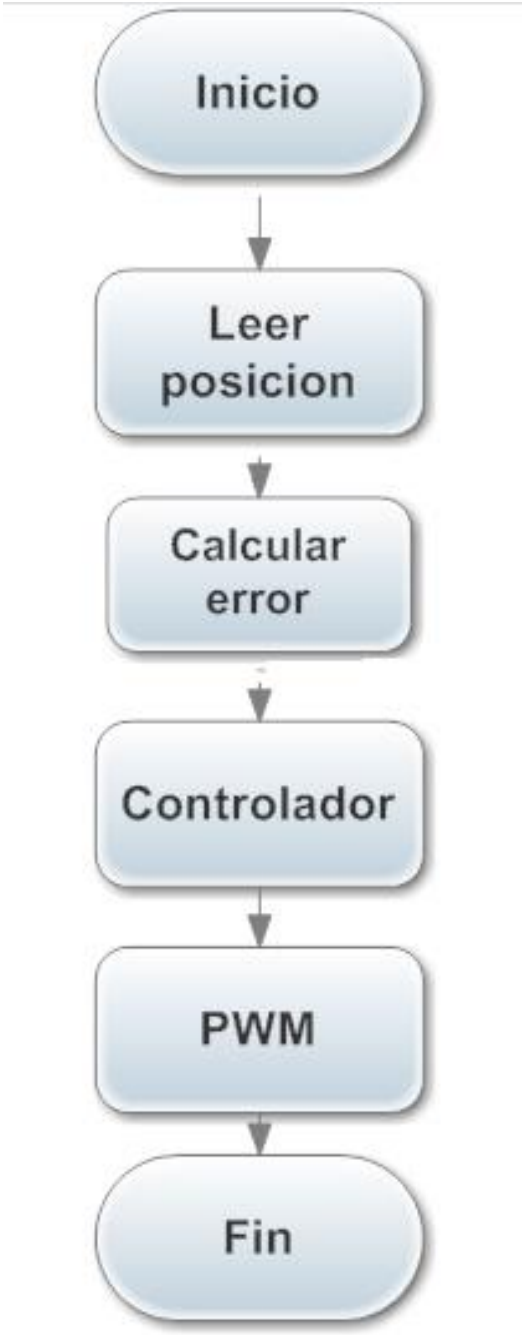

Figura 10. Diagrama de Flujo Control Posición.

3.2. Código implementado para el control de la posición PID de un motor DC de imán permanente.

3.2.1. Declare las variables de acuerdo al tipo y el uso requerido como se muestra continuación en la figura 11.

```
 // DECLARAR VARIABLES.
int   valor, control;
float ki,kp,kd;
float posicionfinal;
float rT,eT,iT,dT,yT,uT,iT0,eT0;
float  maxi, mini;
```

Figura 11. Código para declarar Variables de Posición.

3.2.2. Inicialice la comunicación serial y configure la velocidad de transmisión en 9600 baudios, luego configure como salida los pines digitales PWM de la placa Arduino Mega 2560 R3 como se muestra a continuación en la figura 12.

```
// CONFIGURACIÓN PARÁMETROS DE TRANSMISIÓN Y PINES.
void setup()
{
  Serial.begin(9600);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);

}
```

Figura 12. Código Segmento Setup Arduino.

3.2.3. Continúe con el ciclo infinito característico de arduino void loop(). Luego asigne los valores máximos y mínimos de control que utilizara el PID; también inicialice en 0, las variables de control de error 0 y tiempo integral 0; ahora ponga los valores a las variables de sintonización del PID encontradas en la Guía para la elaboración y sintonización de un controlador de par, posición y velocidad de un motor DC de imán permanente*, así como también* asigne el valor de posición final deseado; calibrando este valor con la lectura que tenga el sensor de posición en la posición 0 como se muestra a continuación en la figura 13.

```
void loop()
{
// CONFIGURACIÓN DE VARIABLES DE CONTROL.
 mini=-1000.0;
 maxi=1000.0;
 iT0=0.0;
 eT0=0.0;
 kp=30;
 ki=1.25;
 kd=0;
 posicionfinal =84.46+90;
```

Figura 13. Configuración de variables.

3.2.4. Lea el puerto analógico A0 y asigne la lectura de éste a la variable valor, luego haga la conversión de esta lectura analógica a grados y asígnela a la variable rT el valor de setpoint o posición final como se muestra a continuación en la figura 14.

```
// LECTURA SENSOR POSICIÓN Y ASIGNACIÓN A VARIABLES.
valor=analogRead(A0);
yT=valor*3.519;
rT=posicionfinal;
```

Figura 14. Configuración de Variables.

3.2.5. Defina el bloque de control PID como se muestra a continuación en la figura 15.

```
// CONTROL PID.
eT=rT-yT;
iT=ki*eT+iT0;
dT=kd*(eT-eT0);
uT=iT+kp*eT;
```

Figura 15. Bloque Control PID.

.
El bloque de control PID es el encargado de calcular en todo momento y en cada ciclo que efectúa arduino el error de consigna (eT) restando la posición deseada o setpoint de posición o lectura registrada en la variable yT, una vez hecho esto, calcula las 3 constantes de control del PID paralelo, la constante de tiempo integral, siguiendo la ecuación característica de esta, así como la derivativa, para que al final se obtenga la salida del PID y se guarde en la variable uT. Esto se repite en cada ciclo de ejecución de arduino.

3.2.6. Asigne los valores mínimos y máximos de control del PID, con un condicional de esta forma, cuando el resultado o salida del PID, supera al máximo se le asigna a la iT0 un valor de 1000 y un error cero nulo, de la misma forma si el valor uT es más pequeño que el valor mínimo se le asigna un valor lógico de -1000 a iT0, y un valor nulo de error cero como se muestra a continuación en la figura 16.

```
// ASIGNACIÓN VALORES MÁXIMOS Y MÍNIMOS DE CONTROL PARA EL PID.
if(uT > maxi)
{
uT=maxi;
iT0=1000.0;
eT0=0.0;
}
else
{
  if(uT<mini)
  {
  uT=mini;
  iT0=-1000.0;
  eT0=0.0;
}
}
```

Figura 16. Bloque Asignación Máximos y Mínimos PID.

3.2.7. Elija las salidas de los pines PWM 8 y 9, en alto o bajo según sea el caso, para de esta manera controlar el sentido del giro del motor. Si el error calculado en el bloque PID de ésta es negativo o positivo el motor gira en el sentido que logre disminuir el error hasta lograr entrar en un rango de 2° (grados) de error. Esto es si el error es negativo el motor girará en un sentido y si es positivo en el sentido contrario como se muestra a continuación en la figura 17.

```
// CONTROL SENTIDO DE GIRO DEL MOTOR.
if(eT<-2.0)
{
       digitalWrite(8, LOW);
       digitalWrite(9, HIGH);
}

if(eT>2.0)

{
      digitalWrite(8, HIGH);
      digitalWrite(9, LOW);
}
```

Figura 17. Bloque Control Giro Motor.

3.2.8. Mapee en un rango de valores numéricos de 0 a 1000 y lógico de 0 a 255, el valor absoluto de la salida del PID uT y utilice el pin 10 como salida analógica con los valores de la variable control, para controlar el voltaje sobre el jumper y así controlar la posición en el motor, para finalizar se asigne a las variables en estado cero iT0 y eT0 los valores encontrados en el bloque PID, imprima en el puerto serial la posición o estado de la variable yT como se muestra a continuación en la figura 18.

```
// MAPEO SALIDA PID Y ASIGNACIÓN AL PIN DE SALIDA DEL JUMPER.
control=map(abs(uT), 0,1000, 0, 255);
analogWrite(10, control);
iT0=iT;
eT0=eT;
Serial.println(yT-84.46);
```

Figura 18. Bloque Salida Final.

3.3. Algoritmo implementado para el control de la velocidad PID de Un motor DC de imán permanente.

3.3.1. Diagrama de flujo del controlador PID de velocidad.

Para desarrollar el hardware del control de posición se tomó como base el diagrama de flujo de la figura 19.

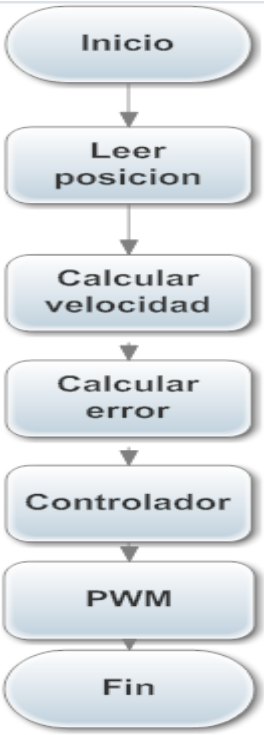

Figura 19. Diagrama de Flujo Control Velocidad.

3.4. Código implementado para el control de la velocidad PID de un motor DC de imán permanente.

El código para el control de velocidad es muy similar al código de control de posición PID, con 2 excepciones. Se requiere una variable más para guardar la posición en tiempos distintos y el cómo medir la velocidad para hacer las comparaciones entre el Setpoint de velocidad y la velocidad medida en cada momento.

3.4.1. Declare las variables necesarias, se requiere 2 variables para guardar la posición del sensor en tiempos distintos esto para calcular la velocidad como se muestra a continuación en la figura 20.

```
// Declarar Variables.
float  valor1,valor2;
int control;
float ki,kp,kd;
float velocidad, velocidadfinal;
float rT,eT,iT,dT,yT,uT,iT0,eT0;
float maxi,mini;
```

Figura 20. Código para Declarar Variables de Velocidad.

3.4.2. Tome la posición medida por el sensor en el Pin A0 en un tiempo 1 guárdelo en la variable valor1, haga la operación para pasar esta medida a grados/segundo y codifique un retardo de 10 ms, ahora tome de nuevo la medida del sensor en el pin A0 en un tiempo 2 y guárdela en la variable valor2 y opere estas 2 variables junto al retardo para obtener la velocidad como se muestra a continuación en la figura 21.

```
// Medir Velocidad.
valor1=analogRead(A0);
valor1=valor1*3.519;
delay (10);
valor2=analogRead(A0);
valor2=valor2*3.519;

velocidad= (valor2 - valor1)/0.01;
```

Figura 21. Código Monitorear Velocidad.

3.5. Código implementado para el control de Par de un motor DC de imán permanente.

3.5.1. Diagrama de flujo del controlador de par.

Para desarrollar el hardware del control de par se tomó como base el diagrama de flujo de la figura 22.

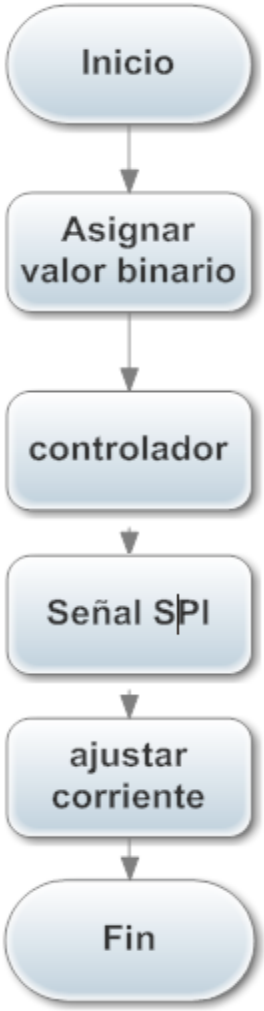

Figura 22. Diagrama de Flujo Control de Par.

3.6. Código implementado para el control de Par de Un motor DC de imán permanente.

El control de par se hace por medio de limitar la corriente disponible para el motor, usando el módulo ICSA001A que limita la corriente a valores constantes manteniendo voltajes constante independientes sin depender de la corriente.

3.6.1. Incluya la librería de comunicación SPI, configure la variable de salida en el puerto SPI SS número 53 e inicialice la comunicación SPI como se muestra a continuación en la figura 23.

```
// CONFIGURACIÓN.

#include <SPI.h>
const int slaveSelectPin = 53;
void setup()
{
  pinMode (slaveSelectPin, OUTPUT);
  SPI.begin();
}
```

Figura 23. Configuración de Variable de salida y Comunicación.

3.6.2. Asigne el valor equivalente de corriente deseado a la salida usando la función digitalPotWrite de la librería SPI como se muestra a continuación en la figura 24.

```
// Asignar valor de salida.
void loop()
{
  digitalPotWrite(50);
}
```

Figura 24. Asignar Valor de Salida.

3.6.3. Configure los diferentes parámetros para la comunicación por el puerto SPI como se muestra continuación en la figura 25.

```
 // Parámetros de comunicación
int digitalPotWrite(int value)
{
  digitalWrite(slaveSelectPin, LOW);
  SPI.transfer(0x11);
  SPI.transfer(value);
  digitalWrite(slaveSelectPin, HIGH);
}
```

Figura 25. Configuración Parámetros de Comunicación.

4. Bibliografía.

[1] Arduino “página oficial de arduino” Disponible en: http://www.arduino.cc/

[2] Draw.io Online Manual del usuario. Disponible en: https://www.draw.io/