UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Ingeniería Técnica industrial

Especialidad en Electrónica Industrial

Departamento de Ingeniería de Sistemas y Automática

Control de posición de un balancín con motor y hélice.

Junio 2012

Autor: Vladimir Viltres La Rosa

Tutor: Francisco Javier García Ruiz

Anexos

Referencias

CAPITULO 1

INTRODUCCIÓN

1.1. Resumen.

Mediante el siguiente trabajo, se pretende explicar la implementación de un control de posición en una barra que presenta un grado de libertad, el cual consiste en el giro respecto a un eje que pasa por su centro de gravedad, el movimiento de giro será provocado por una fuerza de empuje producida por una hélice y un motor de corriente directa, de manera que controlando la velocidad de giro del motor se podrá regular la fuerza de empuje que actúa sobre la barra y con ello la posición de la misma, para muchos todo lo anterior se puede resumir como "helicóptero con un grado de libertad", quizás la anterior frase sea más ilustrativa y permita a todos crear un esquema mental del sistema.

1.2. Objetivos.

-Obtener el modelo matemático del sistema en cuestión, utilizando la analogía de la segunda ley de Newton para el movimiento de rotación de un sólido rígido y las transformadas de Laplace.

-Realizar simulaciones sobre el modelo y comparar en cuanto a respuesta temporal el modelo matemático obtenido y el sistema físico.

- Conseguir la regulación de la posición de la barra, lograr que en todo momento sea la deseada.

-Lograr que el sistema alcance la referencia con el menor sobrepico posible, que en todo momento su evolución temporal sea estable.

-Buena respuesta ante perturbaciones, el sistema deberá ser capaz de corregir su posición ante la acción de fuerzas externas que puedan provocar una desviación de la posición de la barra respecto a la referencia.

-Construcción del módulo de potencia necesario para el control de la planta.

-Implementación del algoritmo de control utilizando Arduino y Simulink.

1.3. Antecedentes.

El proyecto actual tiene como antecedente el funcionamiento de un helicóptero el cual consigue su sustentación gracias al giro de su rotor principal, impulsando el aire desde la parte superior a la inferior de su rotor, y generando un potente chorro de aire debido al "Principio o Teorema de Bernouilli" (aplicable también a los fluidos), esto causa que la masa superior de aire, al aumentar su velocidad, disminuya su presión, creando así una succión que sustenta la aeronave. El perfil de las palas está diseñado de tal forma que el aire circula a mayor velocidad por su parte superior que por la inferior, y a mayor velocidad hay menor presión (=sustentación), y a menor velocidad habrá mayor presión. Una vez en el aire, tiende a girar sobre si mismo pero en sentido contrario al giro de su rotor principal, por ello, este giro ha de ser sincronizado con el giro de su rotor secundario generando lo que se denomina "efecto antipar". En la mayoría de los helicópteros consiste en una doble pala situada en la cola (rotor de cola), en un plano vertical y que empuja en el mismo sentido que el giro del rotor principal, en esta práctica este efecto no será notable ya que el sistema esta fijo a una bancada que restringe la tendencia del sistema a girar respecto al eje vertical.

CAPITULO 2

DESCRIPCION DE LOS ELEMENTOS NECESARIOS

2.1. Características de la planta.

La planta como se observa en la figura 2.1 está formada por dos barras de cobre, el balancín o barra móvil cuya longitud es de 56 cm y la barra bancada cuya altura es de 32 cm, el balancín va unido a la bancada por medio de dos cojinetes o cajas de bolas que permiten el movimiento de giro, en uno de los extremos del balancín hay un soporte donde va colocado el motor y la hélice, el motor es del fabricante Mabuchi con una potencia nominal de 3 Watts y un eje de 2 mm de diámetro, la hélice tiene un diámetro de 13.5 cm. Solidario al eje de giro mediante unos pequeños tornillos está fijado el cursor de un potenciómetro que actúa como sensor de posición.
Es aquí donde comienza todo el flujo de datos, el sensor en este caso potenciométrico, es alimentado con la fuente de 5 Volts que proporciona el microcontrolador, de manera que a cada valor de tensión entre cursor y común le corresponderá una posición determinada del balancín. Este valor de tensión será registrado por el micro mediante una de sus entradas analógicas.

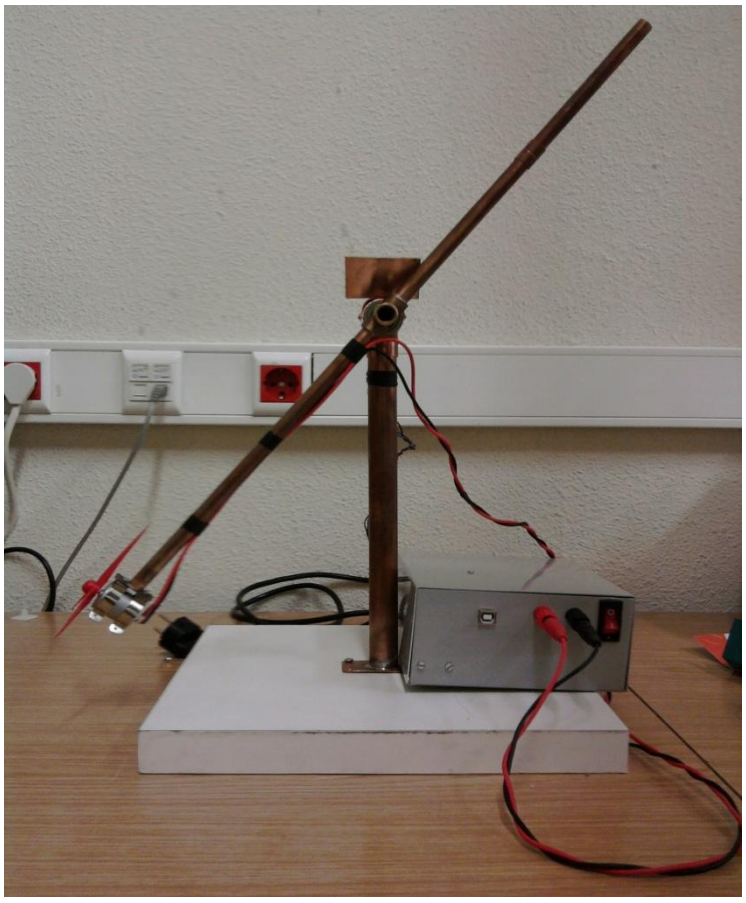


Fig.2.1. Planta real

2.2. Sensor potenciométrico.

Para registrar la posición de la barra, como ya se había comentado anteriormente, se eligió un potenciómetro de cursor giratorio ya que el movimiento del cursor se asemeja al movimiento que se quiere paramétrizar. Se ha seleccionado un resistor variable de manera que la variación de resistencia en función de la posición del cursor sea lineal, de esta manera tendremos una ley de variación matemáticamente sencilla

Para cada posición del cursor habrá un valor diferente de resistencia R(α) y con ello de tensión V(R), este último parámetro será el registrado mediante la lectura de las entradas analógicas del microcontrolador.

El potenciómetro seleccionado tiene un valor nominal de 47 kΩ, si dividimos la tensión de alimentación empleada para dar energía a este sensor entre valor nominal de resistencia, obtendremos en cuanto variará la tensión leída por cada ohm que se incremente debido al desplazamiento del cursor, en otras palabras, la resolución, que

indicará cuanto debe variar la posición de la barra para notar algún cambio en la lectura del valor indicado por el sensor, si hacemos el cálculo tenemos:

$$Res = \frac{V_{cc}}{R} = \frac{5V}{47k\Omega} = 0.1V/k\Omega$$

Para el rango de tensiones en el que se trabaja esta resolución es buena ya que ante mínimas variaciones de posición tendremos un cambio apreciable en la lectura del sensor.

2.3. Arduino.

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Los programas hechos con Arduino se dividen en tres partes principales: *estructura*, *valores* (variables y constantes), y *funciones*. El Lenguaje de programación Arduino se basa en C/C++. En general, un programa en arduino tiene la estructura que se observa en la figura 2.2.

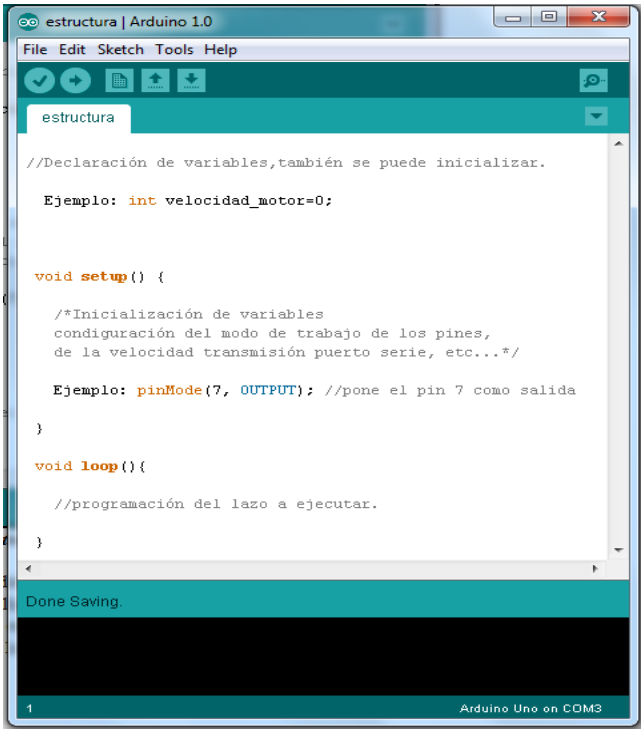


Fig.2.2. Entorno programación arduino.

En donde *setup*() es la parte encargada de recoger la configuración , se invoca una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los pines, o el puerto serie. Debe ser incluida en un programa aunque no haya declaración que ejecutar.
La función *loop*() contiene el código que se ejecutara cíclicamente lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzcan en la tarjeta (lectura de entradas, activación de salidas, etc.). Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.
Ambas funciones son necesarias para que el programa trabaje.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede incidir sobre aquello que le rodea controlando luces, motores y otros actuadores. Para el desarrollo de esta aplicación en especial se emplearan nueve de las tantas funciones que componen la biblioteca de Arduino, la cuales serán explicadas a continuación:

- pinMode(pin, mode)

Esta instrucción es utilizada en la parte de configuración setup () y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT (entrada) u OUTPUT (salida).

pinMode(pin, OUTPUT); // configura ‘pin’ como salida

Los terminales de Arduino, por defecto, están configurados como entradas, por lo tanto no es necesario definirlos en el caso de que vayan a trabajar como entradas.

- Serial.begin(rate)

Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el ordenador es 9600, aunque otras velocidades pueden ser soportadas.

- analogRead(pin)

Lee el valor de un determinado pin definido como entrada analógica con una resolución de 10 bits. Esta instrucción sólo funciona en los pines (0-5). El rango de valor que podemos leer oscila de 0 a 1023, ejemplo:

```
valor = analogRead(pin);   // asigna a valor lo que lee en la entrada ´pin'
```

- analogWrite(pin, value)

Esta instrucción sirve para escribir un pseudo-valor analógico utilizando el

procedimiento de modulación por ancho de pulso (PWM) a uno de los pines de Arduino marcados como “pin PWM”. El Arduino Uno, que implementa el chip ATmega328, permite habilitar como salidas analógicas tipo PWM los pines 3, 5, 6, 9, 10 y 11. El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen de 0-255.

```
analogWrite(pin, valor);        // escribe 'valor' en el 'pin' definido como analógico
```

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios - el valor HIGH de salida equivale a 5v (5 voltios). Teniendo en cuenta el concepto de señal PWM, por ejemplo, un valor de 64 equivaldrá a mantener 0 voltios de tres cuartas partes del tiempo y 5 voltios a una cuarta parte del tiempo; un valor de 128 equivaldrá a mantener la salida en 0 la mitad del tiempo y 5 voltios la otra mitad del tiempo.
Debido a que esta es una función de hardware, en el pin de salida analógica (PWM) se generará una onda constante después de ejecutada la instrucción analogWrite hasta que se llegue a ejecutar otra instrucción analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

- Serial.write(value)

Escribe datos binarios en el puerto serie. Estos datos se envían como un byte o una serie de bytes.

- Serial.read()

Lee o captura un byte (un carácter) desde el puerto serie, devuelve -1 si hay ninguno.

- Delay(val)

Detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción. De tal manera que 1000 equivale a 1seg.

- map(value, fromLow, fromHigh, toLow, toHigh)

Re-mapea un número desde un rango hacia otro. Ésto significa que, un valor (*value*) con respecto al rango *fromLow-fromHight* será mapeado al rango *toLow-toHigh*.

No se limitan los valores dentro del rango, ya que los valores *fuera de rango* son a veces objetivos y útiles. Tener en cuenta que los límites "inferiores" de algún rango

pueden ser mayores o menores que el límite "superior" por lo que *map()* puede utilizarse para revertir una serie de números, por ejemplo:

y = map(x, 1, 50, 50, 1);

La función maneja correctamente también los números negativos, por ejemplo:

y = map(x, 1, 50, 50, -100);

también es válido y funciona correctamente.

La función *map()* usa matemática de enteros por lo que no generará fracciones, aunque fuere el resultado correcto. Los resultados en fracciones se truncan, y no son redondeados o promediados.

Parámetros

value: el número (valor) a mapear.
fromLow: el límite inferior del rango actual del valor.
fromHigh: el límite superior del rango actual del valor.
toLow: límite inferior del rango deseado.

toHigh: límite superior del rango deseado.

Devuelve

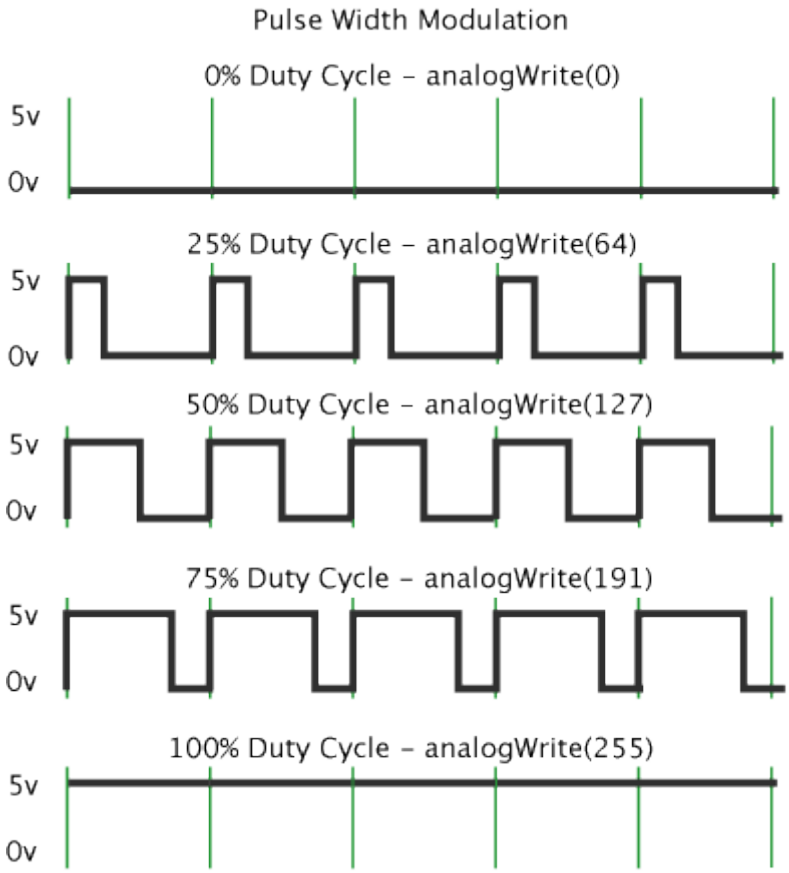
El valor mapeado.

A continuación veamos con algo más de detalle en que consiste una de las técnicas que proporciona arduino y la cual es empleada para el control de nuestra aplicación, el PWM o modulación por ancho de pulso:

La Modulación por Ancho de Pulso (PWM = Pulse Width Modulation) es una técnica para simular una salida analógica con una salida digital. El control digital se usa para crear una onda cuadrada, una señal que conmuta constantemente entre encendido y apagado. Este patrón de encendido-apagado puede simular voltajes entre 0 (siempre apagado) y 5 voltios (siempre encendido) simplemente variando la proporción de tiempo entre encendido y apagado.

A la duración del tiempo de encendido (ON) se le llama Ancho de Pulso (pulse Width). Para variar el valor analógico cambiamos, o modulamos, ese ancho de pulso. Si repetimos este patrón de encendido-apagado lo suficientemente rápido por ejemplo con un LED el resultado es como si la señal variara entre 0 y 5 voltios controlando el brillo del LED.

En el grafico de abajo las líneas verdes representan un periodo regular. Esta duración o periodo es la inversa de la frecuencia del PWM. En otras palabras, con la Arduino la frecuencia PWM es bastante próxima a 500Hz lo que equivale a periodos de 2 milisegundos cada uno. La llamada a la función analogWrite() debe ser en la escala desde 0 a 255, siendo 255 el 100% de ciclo (siempre encendido), el valor 127 será el 50% del ciclo (la mitad del tiempo encendido), etc.



Una vez cargado y ejecutado el ejemplo mueve la arduino de un lado a otro, lo que ves es esencialmente un mapeado del tiempo a lo largo del espacio. A nuestros ojos el movimiento difumina cada parpadeo del LED en una línea. A medida que la luminosidad del LED se incrementa o atenúa esas pequeñas líneas crecen o se reducen. Ahora estás viendo el ancho de pulso (pulse width).

Las funciones anteriores son las necesarias para llevar a cabo la aplicación, ya que proporcionan las herramientas para realizar cada una de las acciones necesarias en el proyecto, como lo son: la lectura de sensores, comunicación entre ordenador y tarjeta de adquisición de datos y control del proceso en general.

2.4. Comunicación Arduino-PC.

En la comunicación con el computador Arduino emplea la comunicación asincrónica. Esto es, requiere de sólo dos líneas de conexión que corresponden con los pines 2 y 3: Pin 2 (Rx) pin de recepción y pin 3 (Tx) pin de transmisión, y del establecimiento de un nivel de tierra común con el computador, esto es, ambas tierras deben estar conectadas, estableciendo el mismo nivel de voltaje de referencia.

Además de realizar las conexiones físicas entre el microcontrolador y el computador, para que pueda establecerse la comunicación serial debe existir un acuerdo previo en la manera cómo van a ser enviados los datos. Este acuerdo debe incluir los niveles de voltaje que serán usados, el tamaño y formato de cada uno de los mensajes (número de bits que constituirán el tamaño de la palabra, existirá o no un bit de inicio y/o de parada, se empleará o no un bit de paridad), el tipo de lógica empleada (qué voltaje representará un cero o un uno), el orden en que serán enviados los datos (será enviado primero el bit de mayor peso o el de menor peso) y la velocidad de envío de datos.

Arduino facilita este proceso para que sólo sea necesario especificar la velocidad de envío de los datos. Esta velocidad es conocida como "baud rate" o rata de pulsos por segundo. Velocidades frecuentes de uso son 9600, 19200, 57600 y 115200.

2.4.1. Puerto Serie.

Un **puerto serie** o **puerto serial** es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por ordenadores y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente. La comparación entre la transmisión en serie y en paralelo se puede explicar usando una analogía con las carreteras. Una carretera tradicional de un sólo carril por sentido sería como la transmisión en serie y una autovía con varios carriles por sentido sería la transmisión en paralelo, siendo los vehículos los bits que circulan por el cable.

En tecnologías básicas, un puerto serie es una interfaz física de comunicación en serie a través de la cual se transfiere información mandando o recibiendo un bit. A lo largo de la mayor parte de la historia de los ordenadores, la transferencia de datos a través de los puertos de serie ha sido generalizada. Se ha usado y sigue usándose para conectar las computadoras a dispositivos como terminales o módems. Los ratones, teclados, y otros periféricos también se conectaban de esta forma.

Mientras que otras interfaces como Ethernet, FireWire, y USB mandaban datos como un flujo en serie, el término "puerto serie" normalmente identifica el hardware más o menos conforme al estándar RS-232, diseñado para interactuar con un módem o con un dispositivo de comunicación similar.

Actualmente en la mayoría de los periféricos serie, la interfaz USB ha reemplazado al puerto serie por ser más rápida. La mayor parte de los ordenadores están conectados a dispositivos externos a través de USB y, a menudo, ni siquiera llegan a tener un puerto serie.

El puerto serie se elimina para reducir los costes y se considera que es un puerto heredado y obsoleto. Sin embargo, los puertos serie todavía se encuentran en sistemas

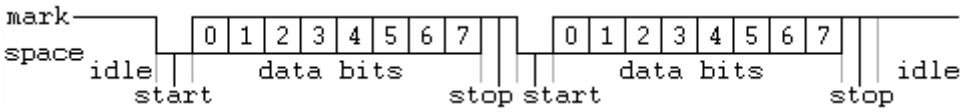
de automatización industrial y algunos productos industriales y de consumo.Los dispositivos de redes, como los enrutadores y switches, a menudo tienen puertos serie para modificar su configuración. Los puertos serie se usan frecuentemente en estas áreas porque son sencillos, baratos y permiten la interoperabilidad entre dispositivos. La desventaja es que la configuración de las conexiones serie requiere, en la mayoría de los casos, un conocimiento avanzado por parte del usuario y el uso de comandos complejos si la implementación no es adecuada.

2.4.2. Puerto serie asíncrono.

A través de este tipo de puerto la comunicación se establece usando un protocolo de transmisión asíncrono. En este caso, se envía en primer lugar una señal inicial anterior al primer bit de cada byte, carácter o palabra codificada. Una vez enviado el código correspondiente, se envía inmediatamente una señal de stop después de cada palabra codificada.

La señal de inicio (*start*) sirve para preparar al mecanismo de recepción o receptor, la llegada y registro de un símbolo, mientras que la señal de stop sirve para predisponer al mecanismo de recepción para que tome un descanso y se prepare para la recepción del nuevo símbolo.

La típica transmisión *start-stop* es la que se usa en la transmisión de códigos ASCII a través del puerto RS-232, como la que se establece en las operaciones con teletipos.



El puerto serie RS-232 (también conocido como COM) es del tipo asincrónico, utiliza cableado simple desde 3 hilos hasta 25 y conecta computadoras o microcontroladores a todo tipo de periféricos, desde terminales a impresoras y módems pasando por mouses.

2.4.3. USB.

El bus universal en serie **USB** es un estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos y dispositivos electrónicos.

El USB, en este caso servirá para enviar y recibir datos desde y hacia el microcontrolador, además de proveer al mismo de alimentación.

Es un sistema de bus serie que puede conectarse con varios dispositivos. Hoy en día es muy utilizado para aplicaciones, tal como la conexión de dispositivos de forma sencilla al ordenador. El USB ha sustituido al RS-232 porque es más rápido, utiliza baja tensión para el transporte de datos (3,3V) y es fácil de conectar. Este puerto es Half duplex, lo que significa que solo puede enviar o recibir datos en un mismo instante de tiempo.

El USB 2.0 tiene 4 hilos: VCC, GND, Datos entrada y Datos de salida, con una velocidad de transferencia de hasta 480 Mbps (60 MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los PC actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y un cuarto que es el negativo o retorno.

2.4.4. Control de comunicación UART.

UART es el Receptor-Transmisor Universal Asíncrono, las funciones principales del chip UART son las de manejar las interrupciones de los dispositivos conectados al puerto serie y de convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora. El UART toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART reensambla los bits en bytes completos. La transmisión serie de la información digital (bits) a través de un cable único u otros medios es mucho más efectiva en cuanto a costo que la transmisión en paralelo a través de múltiples cables. Se utiliza un UART para convertir la información transmitida entre su forma secuencial y paralela en cada terminal de enlace. Cada UART contiene un registro de desplazamiento que es el método fundamental de conversión entre las forma serie y paralelo.

El UART normalmente no genera directamente o recibe las señales externas entre los diferentes módulos del equipo. Usualmente se usan dispositivos de interfaz separados para convertir las señales de nivel lógico del UART hacia y desde los niveles de señalización externos.

El chip UART de Arduino recibe o envía señales a nivel TTL por lo que otro chip integrado en la tarjeta Arduino se encarga de convertir las señales a los niveles de tensión necesarios para hacer posible la comunicación por el USB y que el ordenador reconozca al USB como un puerto COM.

2.5. Simulink.

Simulink es una herramienta de Matlab que funciona mediante un entorno de programación visual, las funciones están representadas por bloques, lo que hace muy sencillo su utilización sin necesidad de emplear lenguajes complejos de programación. Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab (archivos con extensión .m). Simulink genera archivos con extensión .mdl (de "model"). Al ejecutar un modelo implementado en simulink se genera un código en C que el ordenador reconoce y ejecuta.

Para implementar el control de la planta será necesario utilizar varios bloques de Simulink que permitirán la comunicación serie y la ejecución de algoritmos de control, veamos cuales son eso bloques a continuación:

El proceso de comunicación serie es posible gracias al uso de tres bloques, *Serial Configuration*, *Serial Receive* y *Serial Send*, estos bloques emplean el *Universal Serial Port*(USB), para el envio y recepción de datos, este puerto es del tipo *Half-Duplex*, lo cual significa que solo se puede recibir o enviar datos, en un mismo instante de tiempo, o sea, que para enviar o recibir datos el puerto debe estar libre de trafico.

Serial Configuration: Este bloque configura los parámetros de un puerto serie que se puede utilizar para enviar y recibir datos. Se debe dar valores a todos sus parámetros antes de colocar un Serial Send o un Serial Receive, los parámetros a configurar son:

Communication port

Especifica el puerto serie a configurar. Se debe seleccionar un puerto disponible de la lista. Por defecto no hay puerto serie seleccionado, el mismo puerto debe ser utilizado para el Serial Send y el Serial Receive. Cada Serial Send y Receive debe tener un Serial Configuration, lo que implica que si se utilizan varios puertos en una misma simulación, se debe agregar tantos bloques de Serial Configuration como puertos series diferentes haya en la aplicación.

Baud rate

Especifica la velocidad de transmisión en baudios, por defecto es 9600.

Data bits

Especifica el número de bits que se van a enviar por la interfaz serie.

Parity

Especifica como chequear los bits de paridad en los datos transmitidos.

Stop bits

Especifica la cantidad de bits que determinaran el final de un byte.

Flow control

Especifica el proceso de gestión de la tasa de transmisión de datos en el puerto serie.

Timeout

Especifica el tiempo que el modelo va a esperar a los datos durante cada paso de tiempo de simulación. El valor predeterminado es 10 (segundos).

Serial Receive: Este bloque configura y abre una interfaz a una dirección remota especificada usando el Protocolo Serie. La configuración e inicialización ocurre una vez al comienzo de la simulación. El bloque adquiere datos durante el tiempo de ejecución del modelo. Los parámetros principales usados en la aplicación son:

Communication port

Especifica el puerto a través del cual se van a recibir los datos.

Data size

En esta pestaña es importante indicar el número de bits a recibir.

Data type

Como se puede deducir, se debe indicar en esta pestaña el tipo de dato a recibir.

Block sample time

Por último se debe indicar el tiempo de muestreo del bloque, o sea la frecuencia con la que se leerá el dato.

Serial send: Mediante este bloque se enviarán los datos generados por el controlador hacia el Arduino. Sus parámetros son muy similares a los del *Serial Recieve* por lo cual no serán explícitamente comentados.

Convert_to: Este es un bloque del tipo Data Type Conversion, su única función es convertir un tipo de dato en otro, en el caso de la aplicación se emplea la conversión a formato *double* y *uint8*, el formato double se emplea en los elementos gráficos de Simulink y el uint8, es el necesario para trasmitir y recibir por el puerto serie, por lo que este tipo de conversiones se debe garantizar tanto en la comunicación Arduino →PC como viceversa.

Gain: Mediante este bloque se hace posible la visualización de los datos recibidos y enviados, asi como el valor de la referencia, todo ello en unidades de *Volts*, por ello el valor de ganancia aplicado es 5/255 que representa la resolución de un conversor D/A de 8 bits.

Es importante destacar que la lectura de tensión que realiza el Arduino tiene una resolución de 10 bits y por el puerto serie solo se pueden enviar datos de 8 bits, por lo que es necesario llevar el valor inicial leído por el micro controlador, a un rango de 0-255, o lo que es lo mismo, a una resolución de 8 bits. Este hecho explica el por qué del valor del bloque *Gain*.

PID : Aquí es donde se implementa el algoritmo de control PID, gracias a este bloque tendremos un sistema capaz de responder automáticamente a las variaciones en el punto de referencia. El algoritmo que implementa a este controlador en Simulink es el siguiente, para un controlador en paralelo realizable:

$$C(s) = [P + I\frac{1}{s} + D\frac{Ns}{s+N}]$$

Donde P es la ganancia proporcional, I es el tiempo de acción integral, D el tiempo derivativo y N es el coeficiente de filtrado que determina la ubicación del polo del filtro derivativo. La implementación de esta ecuación empleando bloques de simulink, es la que se observa en el diagrama:

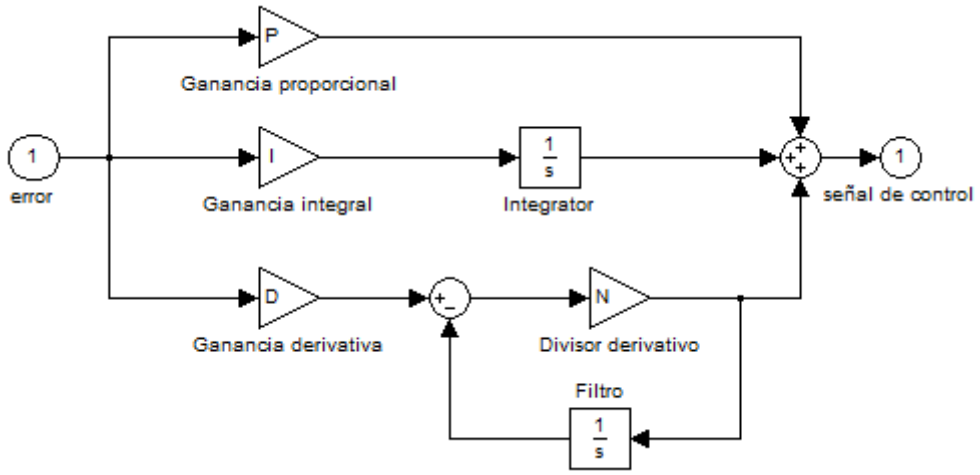


Fig.2.3. Implementación PID paralelo en Simulink

Como vemos tendremos una señal a la salida que siendo los valores de las ganancias P, I y D, no nulos, será una combinación de las tres acciones básicas de control.

CAPITULO 3

MODELADO MATEMÁTICO

3.1. Movimiento de rotación de un sólido rígido.

El movimiento a controlar en esta planta es el de rotación de un sólido alrededor de su eje central de inercia. La variación del *estado de rotación* de un sólido viene determinada por la variación de su velocidad angular por lo que, si queremos describir el movimiento de rotación debemos encontrar una ecuación que nos permita calcular la **aceleración angular** del mismo, para ello la siguiente ecuación:

$$\sum \vec{r} \times \vec{F}_{ext} = I\vec{\alpha} \qquad (3.1)$$

Ésta es la **ecuación del movimiento de rotación de un sólido rígido** que, como puede observarse, es análoga a la segunda ley de Newton y será esta la empleada para plantear la ecuación que rige el movimiento de nuestro sistema.

3.2. Momento de inercia.

3.2.1. Momento de inercia para una masa puntual.

En este caso el sistema contiene dos elementos que aportan momento de inercia, uno es la masa de la barra y el otro la masa del motor. Cada uno de estos aporta un momento de inercia que se calcula de manera independiente y distinta y que finalmente añadiremos para caracterizar el momento de inercia del sistema en general. Veamos a continuación como calcular el momento de inercia para una masa puntual, en este caso la masa del motor.

Para el cálculo del momento de inercia de masas puntuales se empleará la siguiente ecuación:

$$I = \sum m_i l_i^2 \qquad (3.2)$$

donde l_i es la distancia de la partícula de masa m_i al eje de rotación. En el caso de nuestro sistema solo tenemos una masa puntual por lo que la ecuación se puede escribir de la siguiente manera:

$$I_m = m_m l^2 \qquad (3.3)$$

Teniendo en cuenta lso siguientes datos:

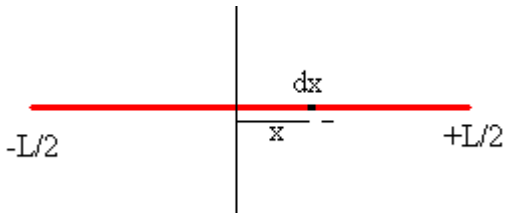
-m_m (masa del motor)= 71g = 0.071kg -*l*= 28 cm = 0.28 m

Al sustituir los valores se obtiene, I_m= 0.0056 kgm^2

Una vez obtenido este resultado se puede proceder al cálculo del momento de inercia para la barra, cuyo eje de giro se encuentra en su centro de gravedad. Veamos como calcularlo.

3.2.2. Momento de inercia de una varilla cuyo eje de giro pasa por su centro de gravedad

Apoyándonos en el siguiente esquema vamos a calcular el momento de inercia de una varilla de masa *M* y longitud *L* respecto de un eje perpendicular a la varilla que pasa por el centro de masas.



La masa *dm* del elemento de longitud de la varilla comprendido entre *x* y *x*+*dx* es

$$dm = \frac{M}{L} dx$$

El momento de inercia de la varilla es:

$$I_c = \int_{-L/2}^{L/2} \frac{M}{L} x^2 dx = \frac{1}{12} ML^2 \qquad (3.4)$$

Pues ahora necesitamos conocer la masa de la barra y la calcularemos de la siguiente manera:
Conociendo la densidad del cobre, material del cual está compuesta la barra, podemos utilizar la ecuación

$\rho = {}^{M}/_{V}$, donde m es la masa total, v el volumen y ρ la densidad.

La densidad del cobre es un dato conocido (ρ=8.93*10^3 kg/m^3) y el volumen de la barra, si la abrimos, se puede aproximar al de un hexágono V=a*b*c, donde a es la profundidad, b la altura y c el ancho, teniendo en cuenta esto, la ecuación para el volumen quedaría:

$V = a * \pi d * c$, siendo d= 8 mm el diámetro de la circunferencia base de la barra, a=2mm y c la longitud de la barra c= 56cm.

Al sustituir y calcular, se obtiene un volumen V= 0,00002813 m^3, que multiplicado por la densidad nos da la masa total de la barra, M= 0,25124 kg.

En estos momentos disponemos de los datos necesarios para el cálculo del momento de inercia que aporta la barra, empleando la ec. (5.4) obtendremos el momento de la barra respecto a su centro de gravedad, sustituyamos los datos en la ecuación:

$$I_c = \frac{1}{12} ML^2 = \frac{1}{12} 0{,}25\ kg * (0{,}56\ m)^2 = 0{,}00653\ kgm^2$$

3.3. Momento de inercia total.

Llegado a este punto, solo falta sumar el momento de inercia de la barra y el aportado por la masa del motor, para obtener de esta manera el momento de inercia total del sistema, que luego se empleará en la obtención del modelo matemático:

$$I_T = I_c + I_m = 0{,}00653\ kgm^2 + \ 0{,}0056\ kgm^2 = 0{,}01213\ kgm^2$$

Ahora estamos en condiciones de continuar con el trabajo matemático que nos ayudará a obtener el modelo en el domino de la frecuencia, con la ayuda de las transformadas de Laplace.

3.4. Obtención de la función de transferencia.

Como ya es de suponer la relación que se quiere obtener es como varia la posición θ en función de la fuerza de empuje producida por la hélice F_e. Veamos entonces las fuerzas que actúan sobre el sistema con el objetivo de aplicar la ecuación 3.1

En la figura 3.1 se puede observar un dibujo de la planta en el cual se representan las fuerzas actuantes sobre el sistema y que son de interés para la obtención del modelo matemático del mismo.

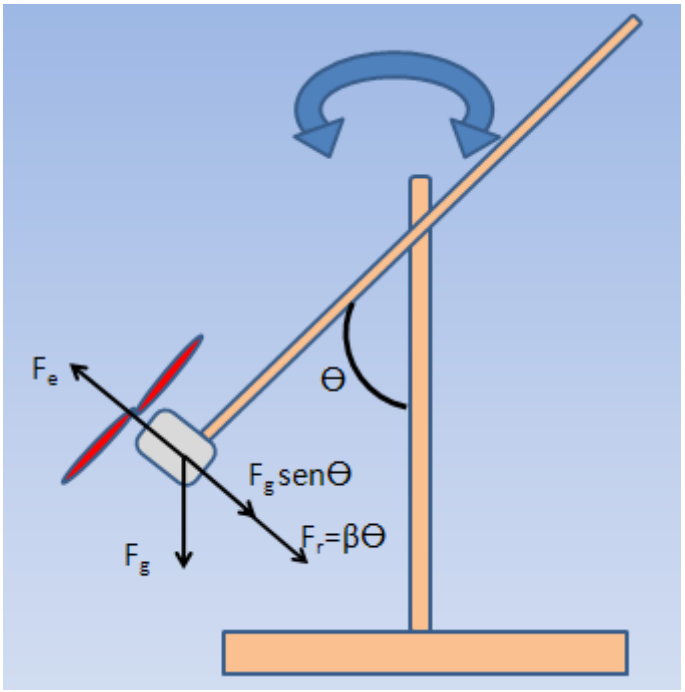


Fig.3.1.Fuerzas sobre el sistema

Para determinar el modelo matemático aproximado de este sistema, se empleara la ecuación del movimiento de rotación de un sólido análoga a la segunda ley de Newton y representada por la ecuación 3.1. Una vez aplicada esta ecuación y luego de haber obtenido cada uno de sus parámetros se realizará la transformada de Laplace a la ecuación resultante, así de este modo se podrá obtener la función de transferencia del modelo aproximado, sobre la cual se desarrollarán algunas pruebas para comprobar la respuesta del sistema modelado y compararla con la del sistema real.

Para obtener la ecuación de movimiento que caracteriza a la planta se tendrá en cuenta que el **eje de inercia de la barra móvil está ubicado justo en su centro de gravedad**, razón por la cual la fuerza resultante sobre la barra será nula, al estar todas las fuerzas de un lado y del otro, de su eje de inercia, compensadas entre sí. Aclarado este punto, cabe señalar que solo serán de interés las fuerzas debidas a la masa del motor y las de fricción con el eje de giro, ya que el giro del cursor del potenciómetro influye notablemente; oponiendo resistencia al movimiento de la barra.

Una vez planteadas las consideraciones principales, estamos en condiciones de aplicar la ec. (3.1):

$$lF_e - lF_g \sin\theta - l\beta\theta = I\alpha \qquad (3.5)$$

donde;

θ: posición de la barra respecto al eje de giro, se consideraran desplazamientos muy pequeños por lo que **sin θ≈ θ** en [**rad**].

l: longitud desde el centro de la barra móvil a uno de sus extremos.

F_g: fuerza de gravedad que actúa sobre el motor.

β: coeficiente de rozamiento.

I: momento de inercia del sistema móvil.

α: aceleración angular.

F_e: fuerza de empuje generada por el giro de la hélice, $F_e = \frac{1}{2}\rho V^2 S_{ref} C_L$, donde;

-ρ: densidad del aire [Kg/m^3] -V: velocidad de giro de la hélice [m/s]

-S_{ref}: área que forma la hélice al girar [m^2] -C_L: coeficiente de elevación [adimens]

La ec. (5.1) puede escribirse de la siguiente manera:

$$\frac{lF_e}{I} - \frac{lF_g}{I}\theta - \frac{l\beta}{I}\frac{d\theta}{dt} = \frac{d^2\theta}{dt^2} \qquad (3.6)$$

Apliquemos ahora las transformadas de Laplace para obtener la función de transferencia:

$$L(\frac{lF_e}{I}) - L(\frac{lF_g}{I}\theta) - L(\frac{l\beta}{I}\frac{d\theta}{dt}) = L(\frac{d^2\theta}{dt^2})$$

$$\frac{l}{I}F_e(s) - \frac{lF_g}{I}\theta(s) - \frac{l\beta}{I}[s\theta(s) - \theta(0)] = s^2\theta(s) - s\theta(0) - \theta(0)$$

Partiendo de condiciones iniciales nulas, la ecuación queda:

$$\frac{l}{I}F_e(s) = s^2\theta(s) + \frac{l\beta}{I}s\theta(s) + \frac{lF_g}{I}\theta(s) \qquad (3.7)$$

Sacando factor común en miembro derecho y despejando, obtenemos la siguiente función de transferencia, que relacionará directamente la entrada con la salida:

$$\frac{\theta(s)}{F_e(s)} = \frac{l/I}{s^2 + \frac{l\beta}{I}s + \frac{lFg}{I}} \qquad (3.8)$$

3.5. Identificación de un sistema de segundo orden.

Como se puede observar, la ecuación obtenida ec. (3.8) corresponde a un sistema de segundo orden que se puede representar de la siguiente forma:

$$\frac{Y(s)}{X(s)} = \frac{K}{s^2 + 2\zeta\,\omega_n + \omega_n^2} \qquad (3.9)$$

Donde ω_n es la frecuencia natural no amortiguada y ξ es el factor de amortiguamiento.

Haciendo la analogía entre los denominadores de la ec. (3.8) y la ec. (3.9) podemos realizar el siguiente planteamiento:

- $2\xi\omega_n = \frac{l\beta}{I}$ $\qquad\qquad$ $\omega_n^2 = \frac{lF_g}{I}$

A continuación la tarea será calcular todos los parámetros del sistema de segundo orden, para ello utilizaremos la gráfica de la figura 3.2 que representa la evolución de la posición del sistema real, obtenida a través de la lectura del sensor potenciométrico cuando se ha sometido la barra a oscilaciones libres.

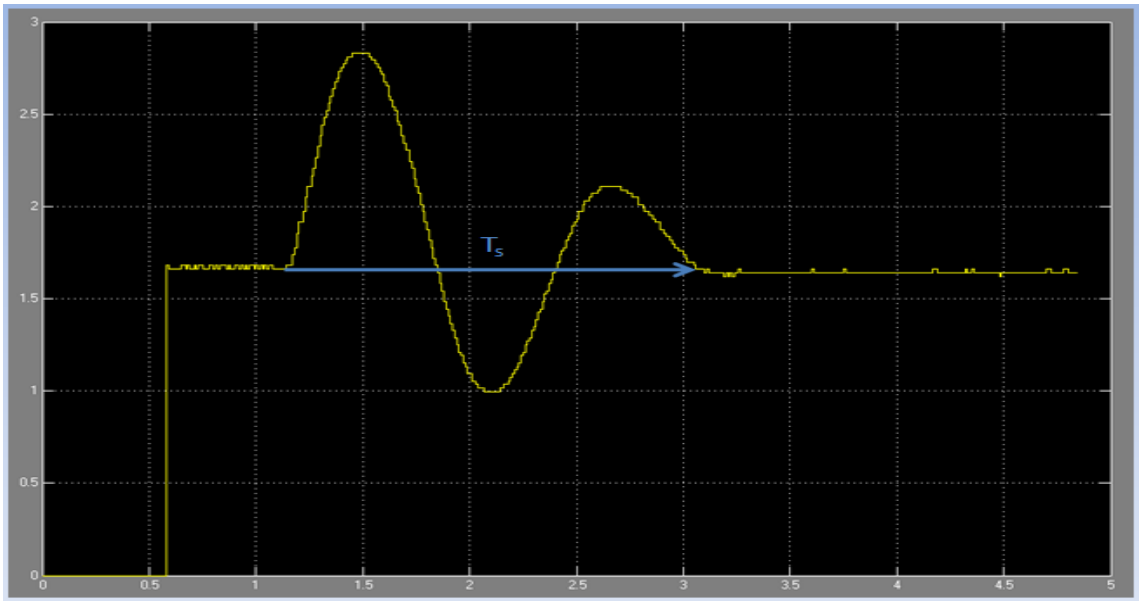


Fig.3.2. Evolución de la posición con oscilaciones libres.

Como se puede observar en la grafica al aplicarle un impulso a la barra, partiendo desde una posición de equilibrio, el tiempo que tarda en volver a su posición estacionaria es T_s= 3,2-1,2= 2 seg, con este parámetro determinado se puede determinar el factor de amortiguamiento ζ, por otra parte tenemos que:

$$\omega_n = \sqrt{\frac{lF_g}{I}}$$, sustituyendo los valores de estos parámetros

$$\omega_n = \sqrt{\frac{0{,}28\ m * 0{,}071\ kg * 10\ m/s^2}{0{,}01213\ kgm^2}} = 4{,}05\ rad/s$$

Con el valor de la frecuencia natural, estamos en condiciones de hallar el coeficiente de amortiguamiento empleando la siguiente ecuación:

$$t_s = \frac{4}{\zeta \omega_n}$$

sustituyendo tenemos que,

$$\zeta = \frac{4}{t_s \omega_n} = \frac{4}{2\ s * 4{,}05\ rad/s} = 0{,}49$$

Con este valor obtenido se puede decir que estamos en presencia de un sistema subamortiguado, ya que $0 \leq |\xi| \leq 1$.

Escribamos la ecuación del sistema con todos los parámetros ya calculados y comprobemos la respuesta del sistema frente a un escalón, de esta manera se podrá saber la fiabilidad de los resultados obtenidos empíricamente para el factor de amortiguamiento:

$$\frac{Y(s)}{X(s)} = \frac{23{,}08}{s^2 + 3{,}69s + 16{,}4} \qquad (3.10)$$

Utilizando la función step() de Matlab, comprobaremos la respuesta del sistema, para ello ejecutemos los siguientes comandos:

- fdt= tf([23.08] , [1 3.69 16.4])

Transfer function:
23.08
---
s^2 + 3.69 s + 16.4

- step(fdt, 3.5)

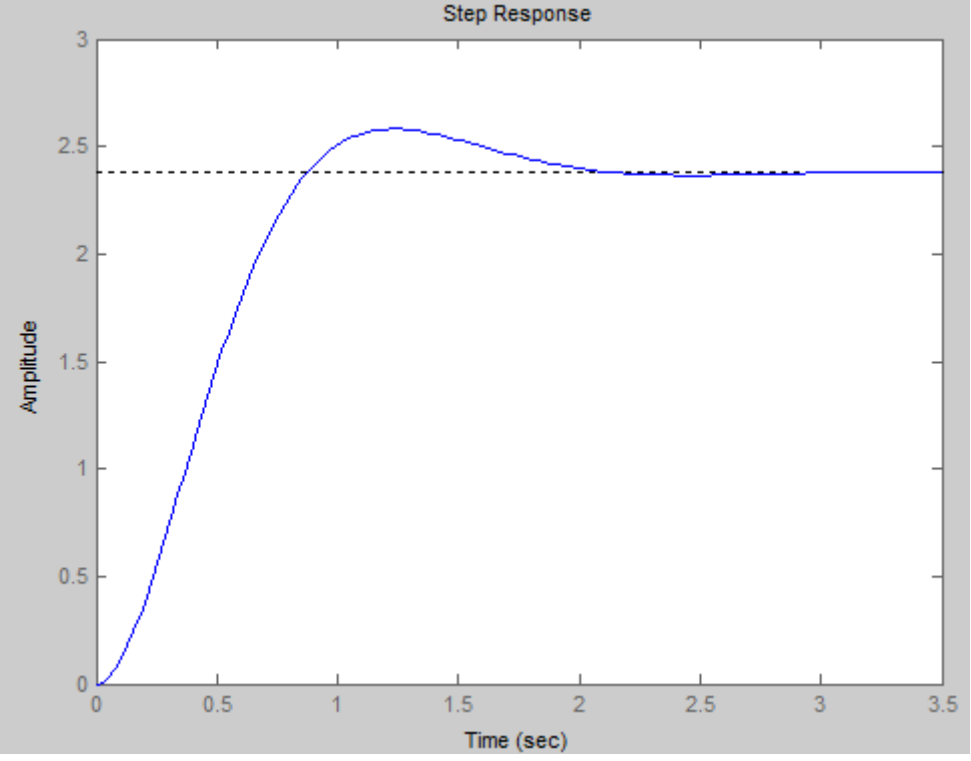


Fig.3.3. Respuesta sistema segundo orden frente a un paso.

Como se observa en esta figura, la respuesta se corresponde a la de un sistema de segundo orden con factor de amortiguamiento aproximadamente de 0,5.

Comprobemos ahora la estabilidad del sistema, mediante el lugar geométrico de las raíces. Las características básicas de la respuesta transitoria de un sistema de lazo cerrado son determinadas por los polos de lazo cerrado, por eso es importante ubicar los polos de lazo cerrado en el plano s y de esto se encarga el lugar de las raíces. La idea básica del lugar de las raíces, es que los valores de *s* que hacen a la función transferencia alrededor del lazo igual a -1, deben satisfacer la ecuación característica del sistema.

El lugar de las raíces de la ecuación característica del sistema de lazo cerrado al variar la ganancia desde cero hasta infinito, da su nombre al método. Este diagrama indica claramente las contribuciones de cada polo o cero a las posiciones de los polos de lazo cerrado.

Este método, permite encontrar los polos de lazo cerrado, partiendo de los polos y ceros de lazo abierto tomando a la ganancia como parámetro. El lugar de las raíces resulta ser muy útil pues indica la forma en que hay que modificar la posición de los polos y ceros de lazo abierto para que la respuesta cumpla con las especificaciones de comportamiento del sistema.

Mediante el comando rlocus() de Matlab podremos ver el lugar de las raíces para distintos valores de ganancia, aunque el valor que nos interesa es el de 23.08

- rlocus(fdt, linspace(0,35,1000))

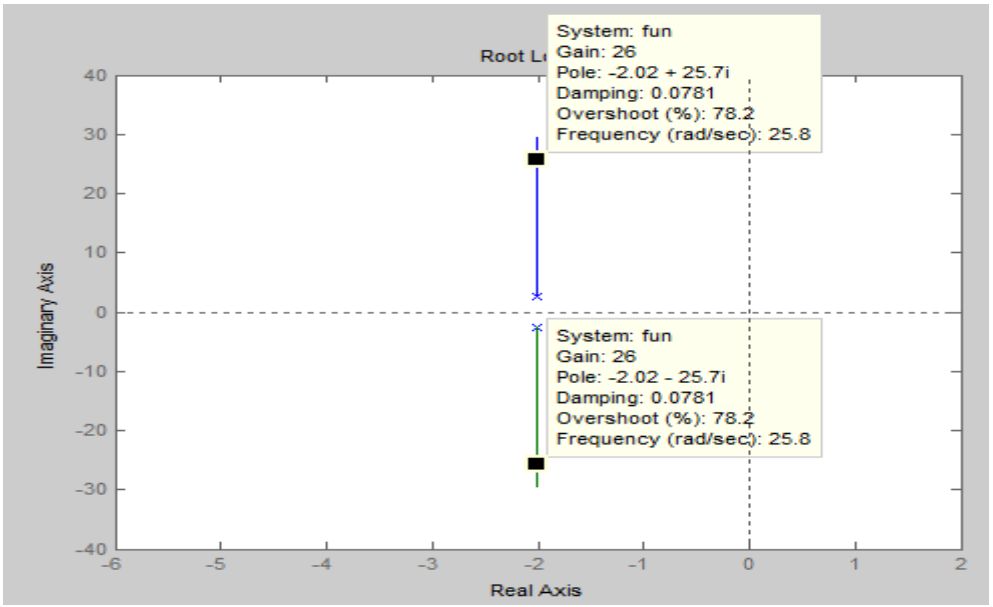


Fig.3.4. Lugar de las raíces

Se puede observar que para el valor de ganancia determinado, el sistema es estable ya que los polos quedan ubicados en el semiplano izquierdo.

3.6. Simulación en lazo cerrado del modelo obtenido.

Para la simulación del modelo en lazo cerrado utilizaremos la herramienta Simulink, añadiremos un regulador PI que será sintonizado mediante el método de prueba y error, el lazo queda de la siguiente manera:

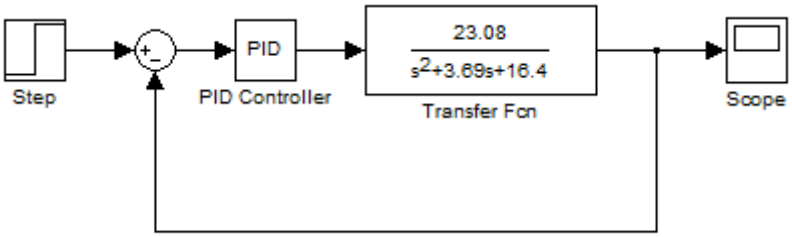


Fig.3.5. Lazo de control

Tras la realización de algunas pruebas y con el objetivo de aproximar la respuesta del sistema modelado a la del sistema real, se obtuvieron los valores para la sintonización del regulador PI, siendo estos P=0.09, I=0.4s.

En la figura 3.6 se observa una grafica de la respuesta del sistema ante un paso escalón

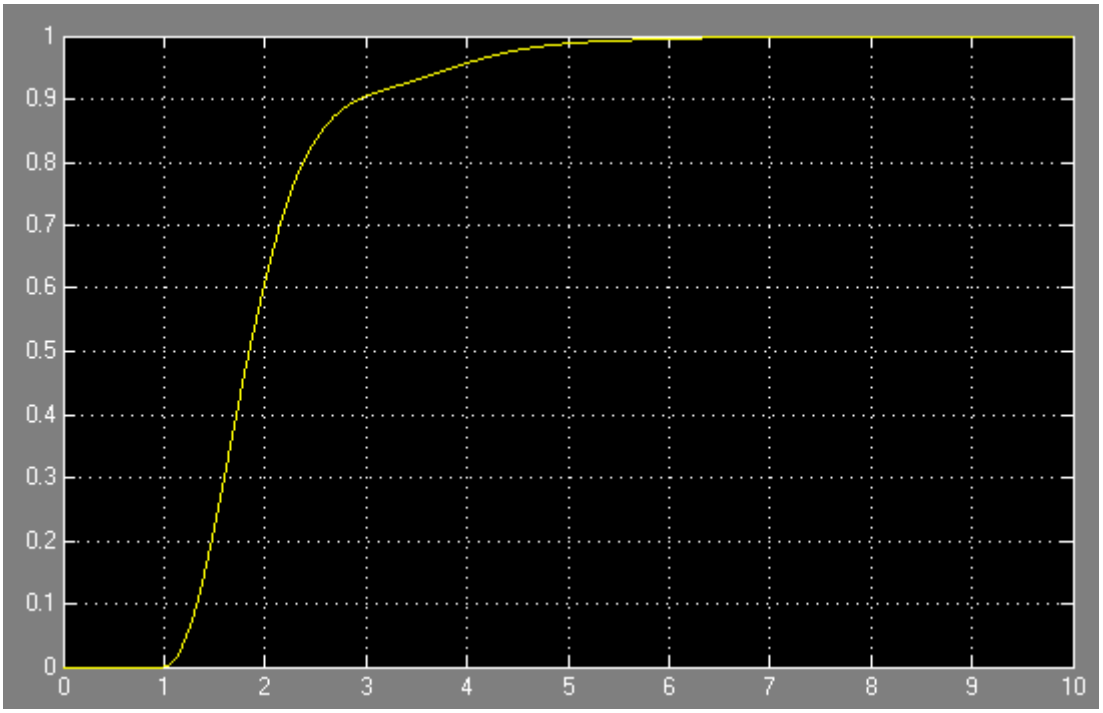


Fig.3.6. Salida frente a un paso escalón.

Como se observa la respuesta del sistema de segundo orden, con los parámetros anteriores, se puede aproximar a la de un sistema de primer orden con retardo.

El comportamiento de un sistema de primer orden con retardo es ideal para lograr una respuesta sin sobrepicos y lo más lineal posible, por eso se intentará que el sistema real tenga una respuesta lo más aproximadamente posible a un primer orden con retardo.

CAPITULO 4

IMPLEMENTACIÓN DEL SISTEMA DE CONTROL

4.1. Regulador PID.

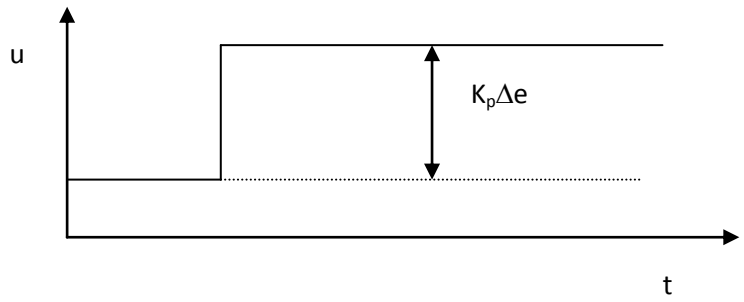
Un PID (Proporcional Integral Derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional determina la reacción del error actual. El Integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El Derivativo determina la reacción del tiempo en el que el error se produce. La suma de estas tres acciones es usada para ajustar al proceso vía un elemento de control como la posición de una válvula de control o la energía suministrada a un calentador, por ejemplo. Ajustando estas tres variables en el algoritmo de control del PID, el controlador puede proveer un control diseñado para lo que requiera el proceso a realizar. La respuesta del controlador puede ser descrita en términos de respuesta del control ante un error, el grado el cual el controlador llega al "set point", y el grado de oscilación del sistema.

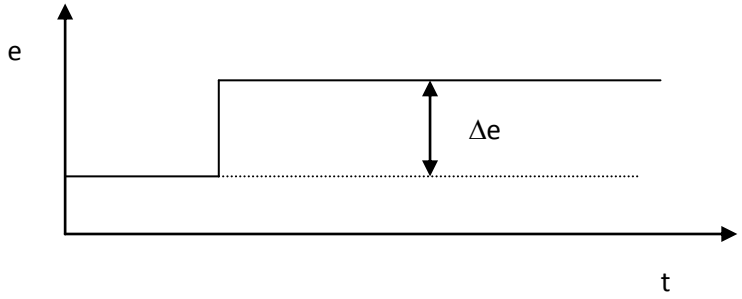
- **Acción proporcional:**

En la rama superior de la figura 2.3, tendremos como resultado la multiplicación de la señal de error por la ganancia proporcional, esto implica que el controlador proporcional produce una variable de salida u proporcional al error del sistema *e*.

$$u(t) = K_p\, e(t)$$

La ganancia proporcional K_p ó P es la cantidad por la cual la variable de control u cambia cuando el error cambia Δe.





Un controlador proporcional responde rápidamente ante el error del sistema pero no es capaz de eliminar completamente las perturbaciones, o eliminar completamente el error. Puede producir inestabilidad.

En la práctica cuando queramos conocer datos sobre la ganancia proporcional de un PID comercial, probablemente no la encontremos como K_p ó P, sino que se utiliza el término Banda Proporcional, PB. La relación entre una y otra se expresa mediante:

$$PB = \frac{100}{K_p}$$

Se escribe de esta forma porque normalmente PB se expresa en porcentaje.

Una banda proporcional ancha es lo mismo que una ganancia baja, y una banda proporcional estrecha equivale a una ganancia alta.

La banda proporcional también se define como el error que se requiere para llevar la salida del controlador del valor más bajo hasta el más alto.

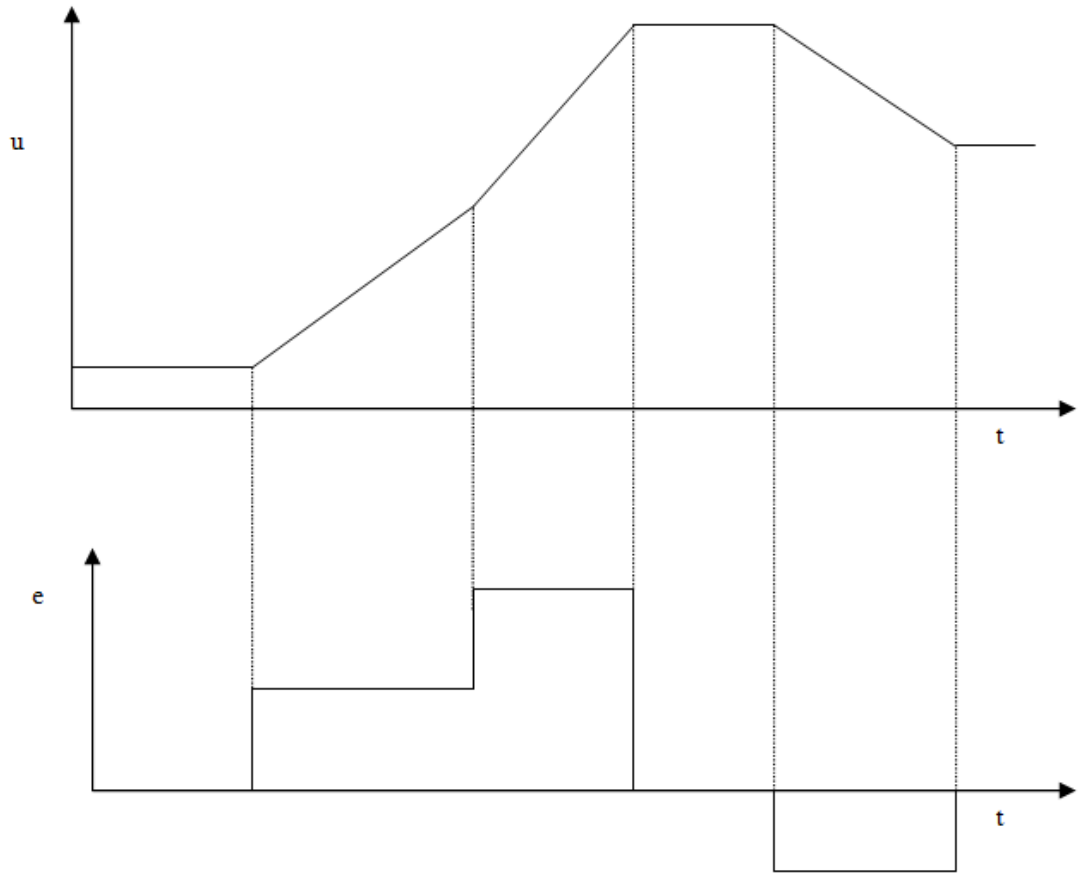
- **Acción integral :**

Esta acción básica de control, se observa en la rama intermedia de la figura 2.3, donde obtendremos una señal que será la multiplicación de la ganancia integral I ó tiempo de acción integral por la variación del error en el tiempo, de forma matemática:

$$u(t) = T_i \int_0^t e(t)dt$$

Ti es el tiempo que tarda la acción integral en igualar a la acción proporcional.

Si lo vemos de forma grafica será así:

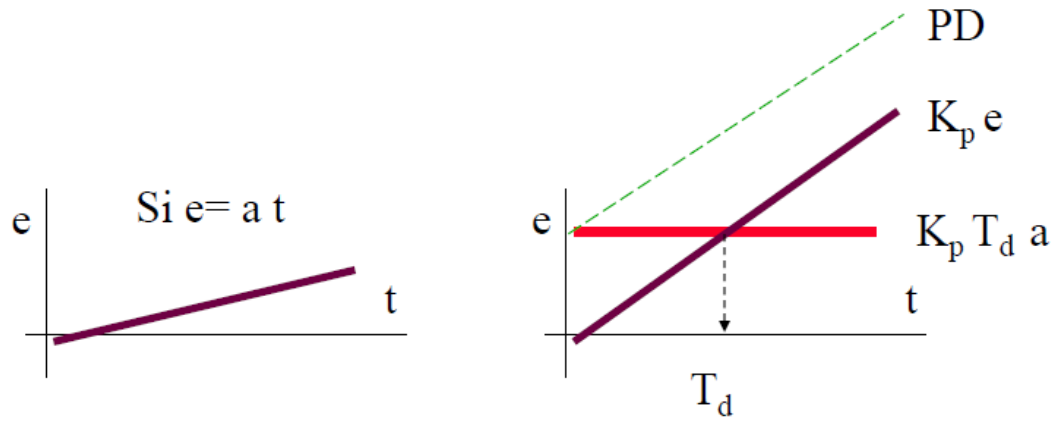


El controlador integral se utiliza sobre todo cuando tenemos problemas de error en régimen estacionario, ya que su finalidad es corregir las desviaciones sobre la referencia y lograr finalmente que error de estado estacionario se haga cero.

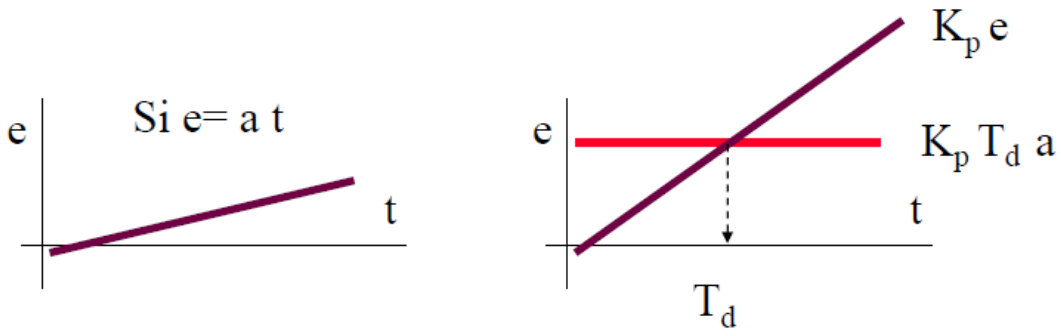
- **Acción derivativa :**

Un regulador P con ganancia alta para dar respuesta rápida puede provocar oscilaciones por señal de control u excesiva. La acción derivativa acelera la u si e crece y la modera si e decrece, evitando oscilaciones. En la rama inferior de figura 2.3 se obtiene una señal que viene a ser la implementación real de la acción derivativa, teóricamente la acción derivativa se representa:

$$u(t) = T_d \frac{de(t)}{dt}$$



Con e variando linealmente, la acción derivativa da la misma u que la acción proporcional daría Td sg. más tarde. Acción anticipativa, no influye en el estado estacionario.



Td tiempo que tarda la acción derivativa en igualara la acción proporcional si e= a*t.

Esta acción de control es buena para procesos lentos y donde el ruido no sea un factor determinante, ya que la acción derivativa tiende a introducir ruidos cuando se producen saltos en la referencia.

- **Control Proporcional-Integral:**

En este tipo de control la salida del controlador es una suma aritmética de una acción proporcional más una acción integral.

El controlador proporcional tiene mejor característica dinámica, ya que actúa ante cualquier desviación en el error de forma casi inmediata, sin embargo es incapaz de eliminar los errores en régimen estacionario

El controlador integral disminuye este error en régimen estacionario pero responde más lentamente que el controlador proporcional.

$$m(t) = K_p e(t) + K_i \int_0^t e(t)dt$$

o

$$m(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt$$

El valor de T_i equivale al tiempo requerido en el modo integral para alcanzar el cambio en la salida producido por el modo proporcional.

Este tipo de control es recomendado en la mayoría de los casos ya que se puede lograr que el sistema a controlar responda satisfactoriamente, señal de salida con poco ruido, error estacionario nulo.

4.2. Interfaz física.

Como es de suponer, el micro controlador arduino es incapaz de suministrar los niveles de tensión y corriente que se necesitan para hacer funcionar el motor empleado en la aplicación con la potencia requerida, por lo tanto el uso de una fuente de tensión externa se hace indispensable, además necesitaremos un elemento que sea capaz de conmutar a una frecuencia determinada, con el objetivo de aumentar o disminuir los niveles de tensión que llegan al motor y que finalmente es lo que permitirá controlar la fuerza de empuje que produce la hélice. La interfaz física de forma esquemática se puede observar en la siguiente figura:

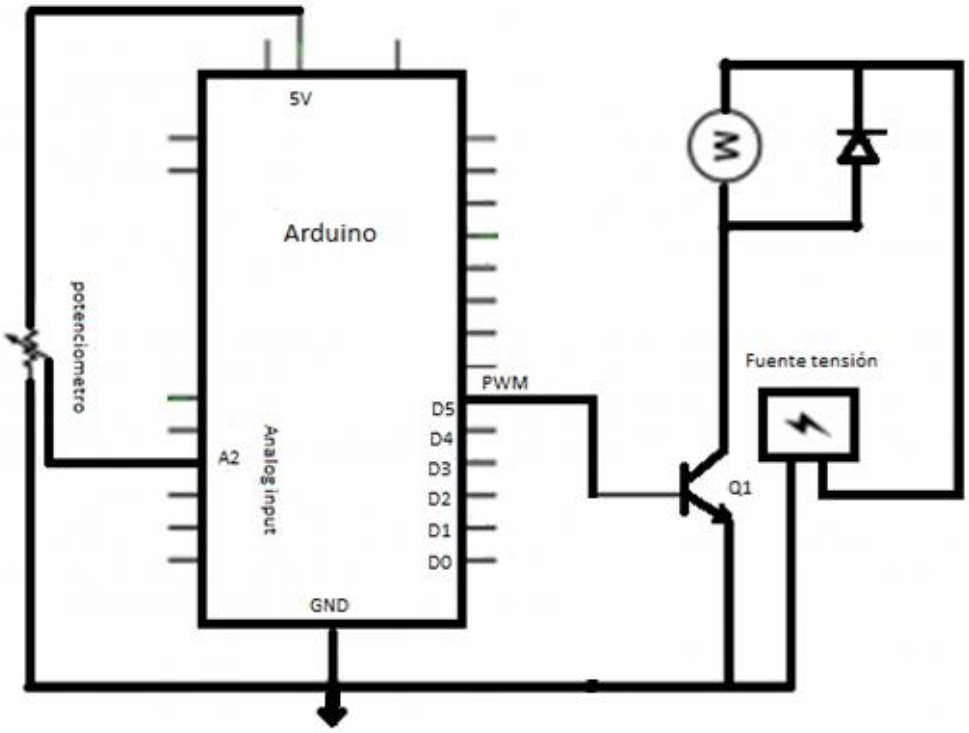


Fig.4.1. Interfaz física.

El microcontrolador enviará a la puerta del transistor una señal PWM, y por tanto esta misma señal PWM pero de potencia será la que reciba el motor, de manera que se podrá controlar la tensión del motor aumentando o disminuyendo el ancho del pulso y con ello el valor medio de la tensión que llega al motor. La representación de este montaje será la siguiente:

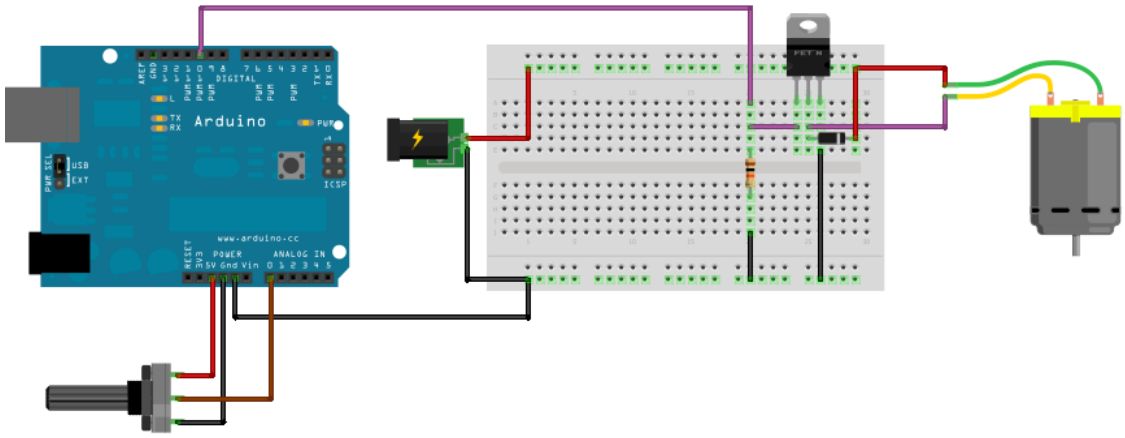


Fig.4.2. Esquema eléctrico.

Este esquema representa toda la parte física que se necesitará para el control y la puesta en marcha de la aplicación, todo lo anterior se instaló en el interior de un chasis y el resultado se puede apreciar en las siguientes imágenes:

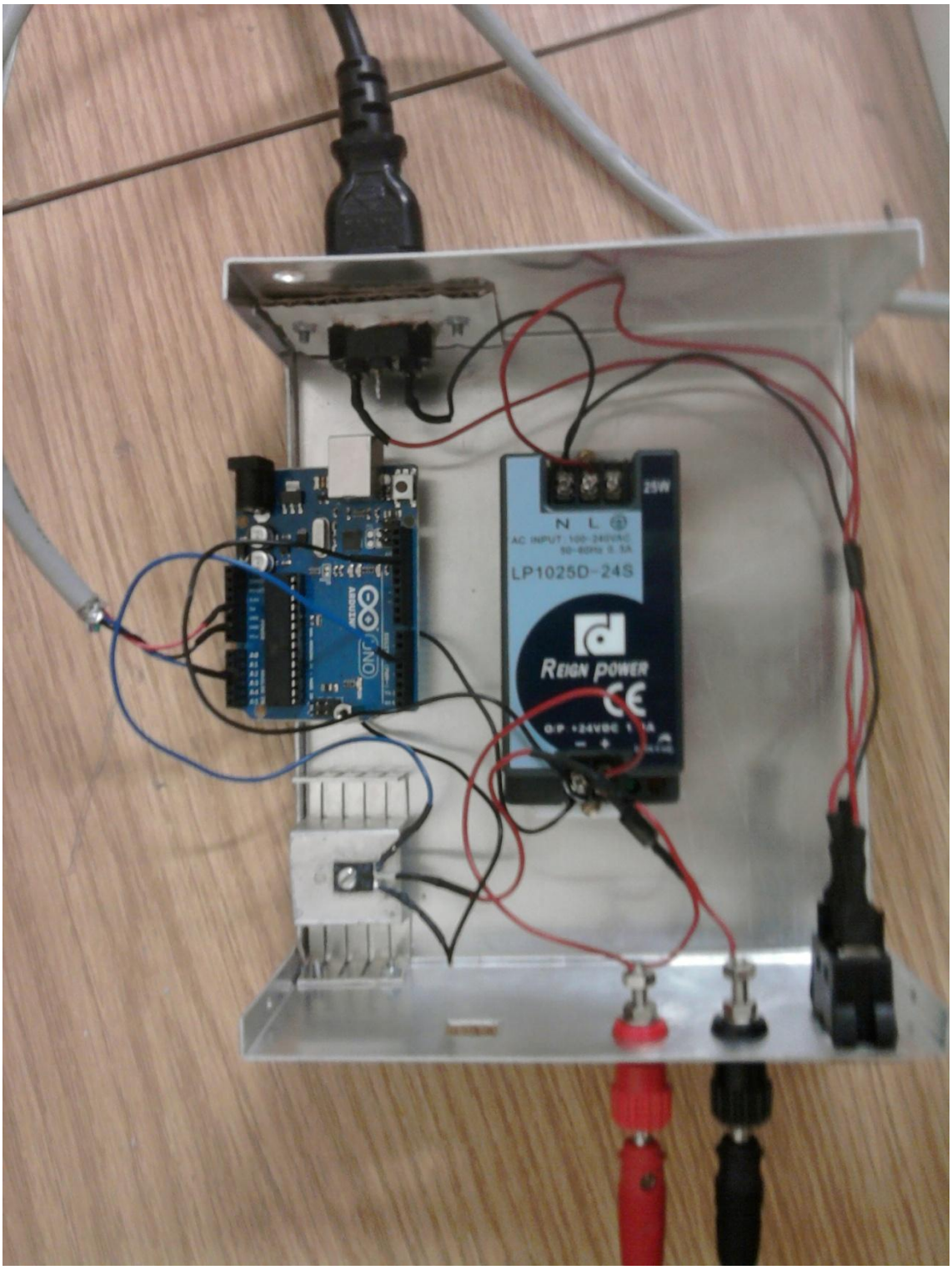
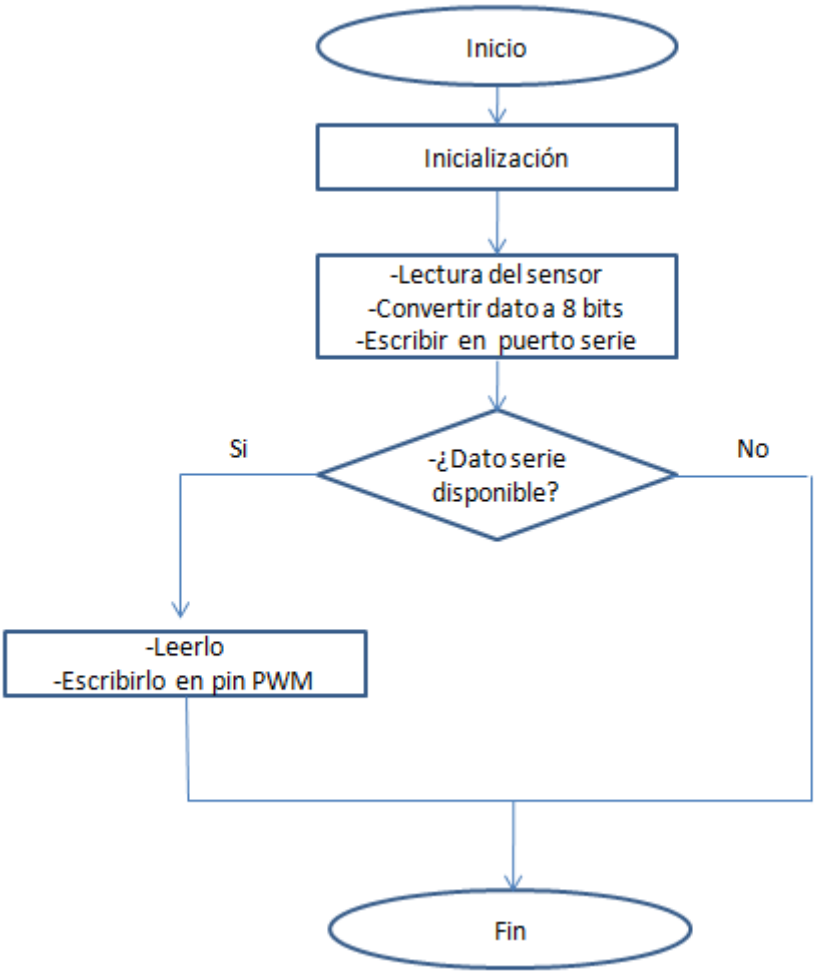


Fig.4.3. Montaje en el chasssis

4.3. Programación de Arduino.

El programa en Arduino es muy sencillo y después de haber explicado en el apartado 2.2 las funciones que se utilizaron para implementar esta aplicación, es momento de ver el código programado, comencemos por ver un diagrama de flujo que brindará una idea concisa de lo que se quiere programar:



En general la idea es comenzar declarando las variables necesarias para la aplicación, inicializar la función setup() de arduino y la velocidad de transmisión del puerto serie, luego de esto comenzaría a ejecutarse el lazo o loop() que permitirá de manera cíclica la parte principal del proceso, que sería realizar la lectura del sensor potenciométrico, convertir el dato leído con una resolución de 10 bits a un dato de 8 bits de manera que podrá ser enviado por el puerto serie. Una vez realizada esta primera parte el programa en arduino deberá esperar a que haya un dato listo en el puerto serie para ser leído, por lo que se hace necesario realizar esta encuesta al puerto serie, ¿Hay dato disponible?, en

caso positivo se procede a leerlo y luego a escribirlo en la salida digital marcada como PWM, en caso contrario el programa se termina y vuelve a comenzar el lazo.
Veamos ahora como se implementa lo anterior en el código de programación de arduino:

```
byte pinout=5;                          // pin digital de salida PWM.
byte ctrlmot=0;                         // variable que se escribirá en la salida PWM.
int pot_val=0;                          // variable que almacena lectura del potenciómetro

void setup(){
Serial.begin(9600);
pinMode(pinout, OUTPUT);
}

void loop(){

  pot_val=analogRead(2);                //leyendo potenciómetro.
  pot_val=map(pot_val, 0, 1023, 0, 255);  //convertir a número de 8 bits.
  Serial.write(pot_val);                //escribiendo dato en puerto serie.

  if(Serial.available()>0){             // verifica si el puerto serie está habilitado.
    ctrlmot=Serial.read();              // leer dato en el puerto serie.
    analogWrite(pinout, ctrlmot);       // escritura de la señal de control en pinout.
  }
  delay(20);                            // espera para estabilizar el conversor.
}
```

Con este pequeño código será posible leer el sensor potenciométrico, enviar el dato hacia el PC, recibir el dato de la señal de control generada por el PID y escribir el mismo en pin digital de salida PWM.

Esta es toda la programación que se necesitará.

4.4. Lazo de control. Simulink.

En la siguiente figura se puede observar el lazo de control creado mediante la herramienta de Matlab, Simulink.

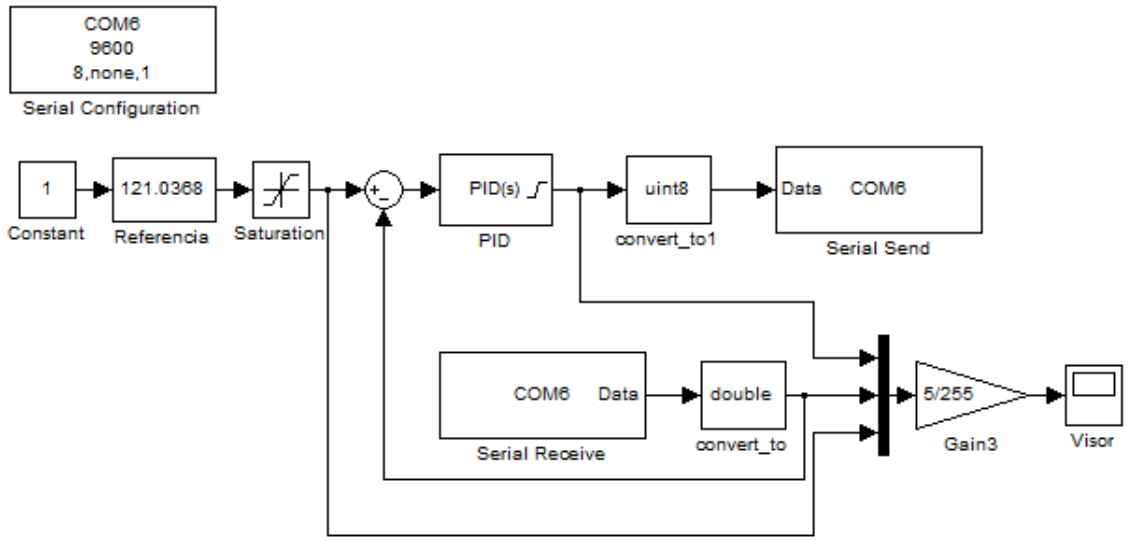


Fig.4.4. Lazo de control.

Todo comienza con la lectura de los datos enviados desde Arduino, de esto se encarga el bloque Serial Receive, que muestrea el buffer del puerto serie, seguidamente es necesario convertir estos datos recibidos en forma de bits en un número de valor doble entendible para el usuario, en este paso ya tendríamos el valor de la variable que queremos controlar, ahora la muestreamos para compararla con la referencia y corregir el error, además de graficar su valor en unidades
de tensión, para ello es necesario multiplicar el valor en doble por la resolución de un conversor de 8 bits (5/255), ya que los datos transmitidos ocupan como máximo un byte.

El valor de referencia es ajustable mediante un control deslizable, se ha colocado un bloque de saturación seguido de la referencia que permite limitar el movimiento en un rango deseado por el usuario y que permita al sistema moverse en una zona de estabilidad, este bloque de saturación deja pasar el dato tal cual, siempre que se encuentre en el rango definido por los limites inferior y superior del bloque, si está por debajo la salida del bloque será el límite inferior, será el superior cuando ocurra lo contrario.

Una vez realizada la comprobación del valor de referencia, se compara esta con el valor recibido del sensor y el error producido por la diferencia entre ambas es enviado al bloque PID, que creará la señal de control correspondiente para la corrección de ese

error. La salida del regulador se ha limitado en un rango de 50 como valor mínimo y 255 como máximo, son necesarios estos valores para poder manejar de manera correcta y eficaz la señal PWM que escribirá arduino en su salida y que será la responsable de hacer conmutar al transistor cuya función es trabajar como interruptor entre la fuente de

tensión y el motor, incrementando o disminuyendo el valor medio de tensión que le llega al motor.

Por último, a la salida del PID otro conversor de tipo de datos para realizar el proceso inverso a la primera conversión, o sea, convertir a formato unit8, número de 8 bits, permitiendo de esta manera que el bloque Serial Send pueda transmitir los datos a través del puerto serie.

4.5. Sintonización del regulador PI.

Se ha elegido un regulador PI ya que este nos permite reducir el error de estado estacionario a cero, tiene buen comportamiento ante saltos en la referencia, no introduce ruidos, hecho que estaría presente si el controlador tuviese acción derivativa.

Para sintonizar el PI se utilizará el método de prueba y error, teniendo en cuenta que se quiere logra una respuesta sin sobrepico y lo más lineal posible. A continuación veamos la respuesta del sistema para diferentes valores de la ganancia proporcional y el tiempo integral.

En la grafica que muestra la figura 4.5 tenemos la evolución de la posición de la barra siendo 1 y 2 los valores de las ganancias proporcional e integral, como se puede apreciar estos valores tan elevados generan una señal de control (color amarillo) con mucho ruido, hecho que no es deseable para la aplicación. Por tal motivo, salta a la luz que hay que disminuir estas ganancias sobre todo la proporcional, así que en la siguiente prueba eso será lo que se hará, disminuir estos valores.

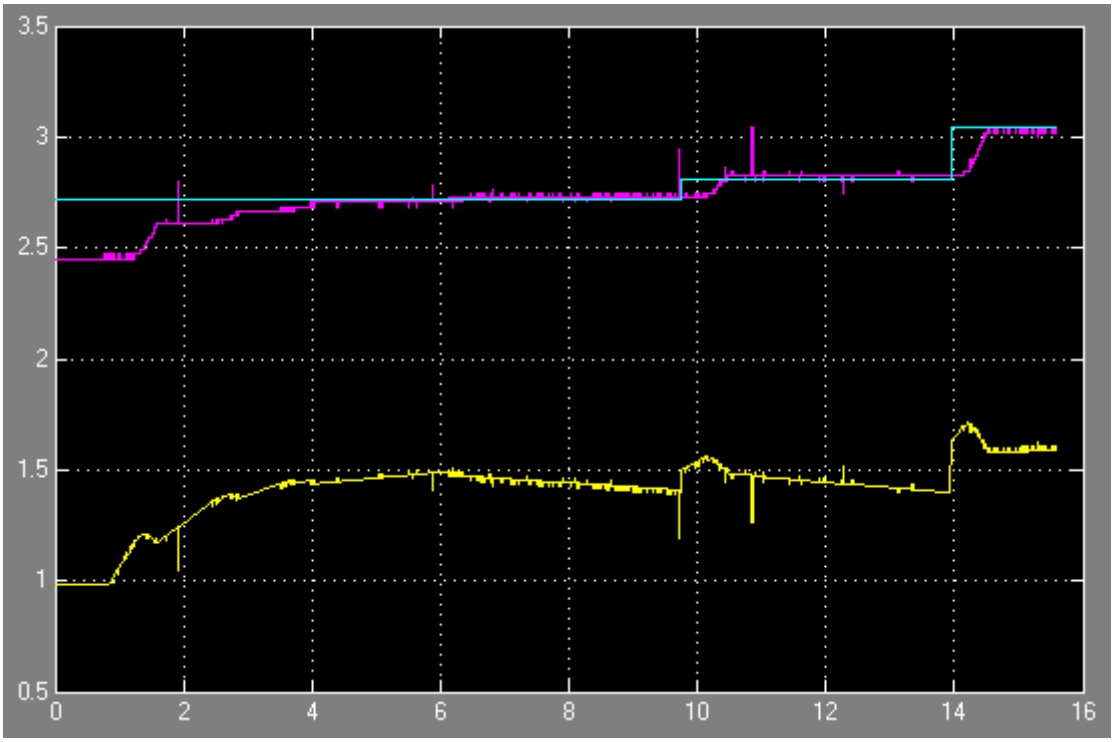


Fig.4.5. Respuesta del sistema con P=1 I=2.

Luego de repetidas pruebas similares a la anterior se llegó a la conclusión de que para tener una respuesta que sea la deseada, sin ruido en la señal de control, sin sobrepicos en la evolución de la posición y con buenos tiempos de respuesta, los valores de las ganancias proporcional e integral debían ser de 0.1 y 1.4 respectivamente. En la siguiente imagen, figura 4.6, se puede observar la respuesta del sistema para estos valores.

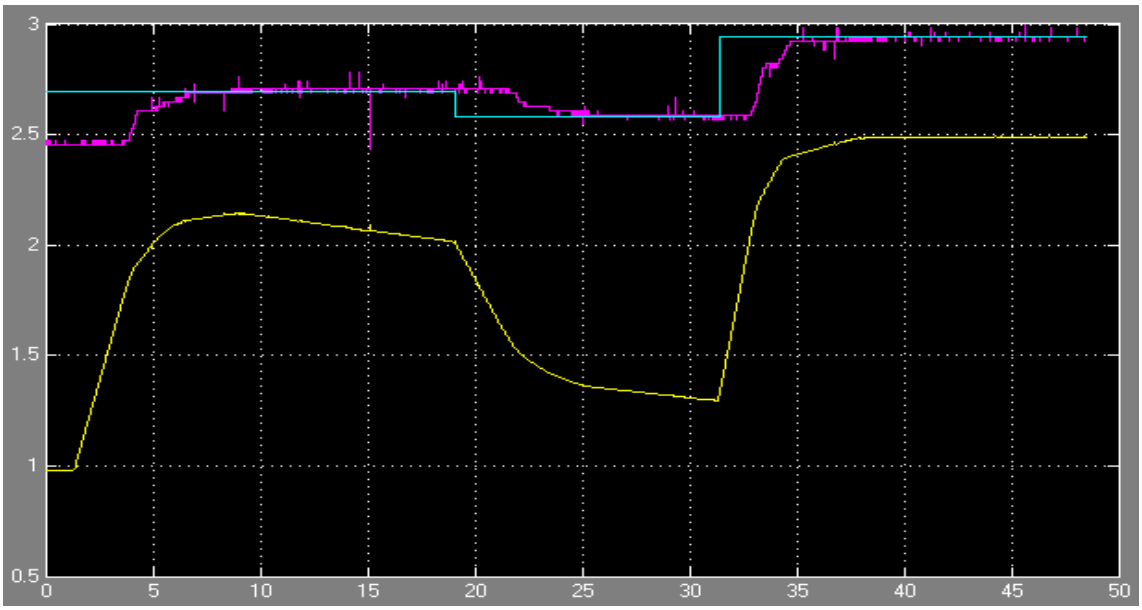


Fig.4.6. Respuesta del sistema con P=0.1 I=2.

CAPITULO 5

RESULTADOS Y CONCLUSIONES

5.1. Resultados.

Al realizar pruebas sobre el sistema se obtuvieron los resultados que se muestran en la figura 5.1. En verde tenemos la referencia fijada para la barra, el lila es la posición de la barra leída por el potenciómetro y en amarillo se observa la señal de control producida por el regulador PI.

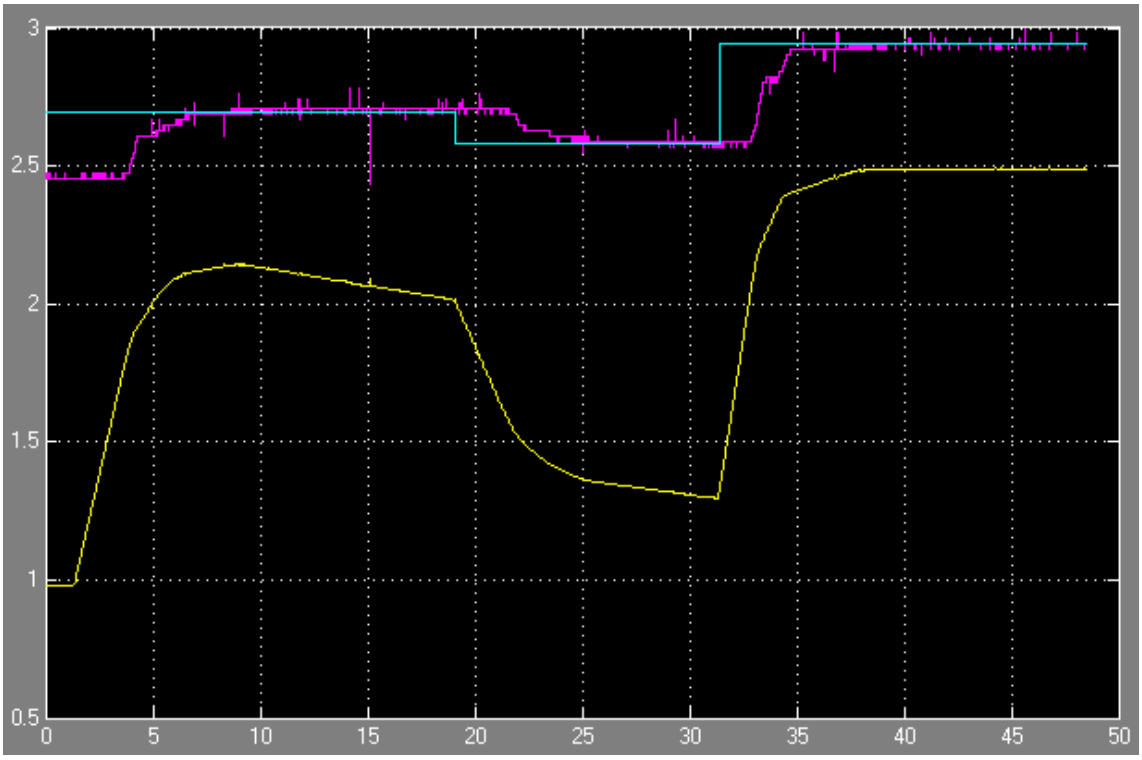


Fig.5.1. Respuesta del sistema.

Como se observa el sistema tiene una buena respuesta, la referencia se alcanza sin que haya sobrepicos en la evolución de la posición, la cual se incrementa prácticamente de forma lineal con cierto retraso, hecho que nos permite aproximar el comportamiento de este sistema de segundo orden al de un sistema de primer orden con retardo. Cabe destacar que se observa algo de ruido hecho que es provocado por las vibraciones que introduce el movimiento de giro del motor.

5.2. Conclusiones.

Las pruebas realizadas al concluir la aplicación han sido satisfactorias, se ha logrado implementar un sistema que tiene un comportamiento aceptable y que cumple con los objetivos planteados, sin sobrepicos y cuya posición evoluciona linealmente ante cambios en la referencia. Desde el punto de vista teórico, al comparar el modelo

matemático en lazo cerrado con el sistema de control real en cuanto a respuesta temporal, vemos que aunque con valores diferentes para las ganancias proporcional e

integral del regulador PI, ambas respuestas son muy similares y semejantes a las de un sistema de primer orden con retardo.

Con la realización de este proyecto se han puesto en práctica una buena parte de los conocimientos adquiridos durante los años de formación en la titulación de Ingeniería técnica en Electrónica Industrial. Me ha aportado un punto de vista analítico a la hora de reconocer como utilizar los conocimientos adquiridos para resolver cierto problema de la vida real, hecho que me servirá de experiencia para mi desenvolvimiento como profesional de la rama electrónica.

5.3. Líneas de desarrollo futuras.

El sistema de control ha sido implementado para un mecanismo con un solo grado de libertad, que haciendo un símil con un helicóptero sería el de ascenso y descenso (giro respecto al eje vertical), para futuros proyectos no estaría mal añadir otro grado de libertad que bien puede ser el de desplazamiento (giro respecto al eje horizontal), con esto ya estaríamos creando un sistema con un mayor grado de complejidad y algo más cercano al helicóptero tradicional con un rotor principal y el rotor secundario o de antipar.

ANEXOS

Anexo I. Diseño de una fuente de alimentación.

El siguiente circuito nos permitirá construir una robusta fuente de alimentación de las tensiones más comúnmente usadas y un alto valor de corriente, el valor de la tensión de salida dependerá, entre otras cosas, de los valores de las resistencias R1, R2 y R3. A pesar de ser muy simple, hay que prestar atención en algunos detalles, ya que estamos construyendo un circuito que puede manejar una potencia importante, en nuestro caso construiremos una fuente de 9 V 3 A lo que equivale a una potencia de 27 W.

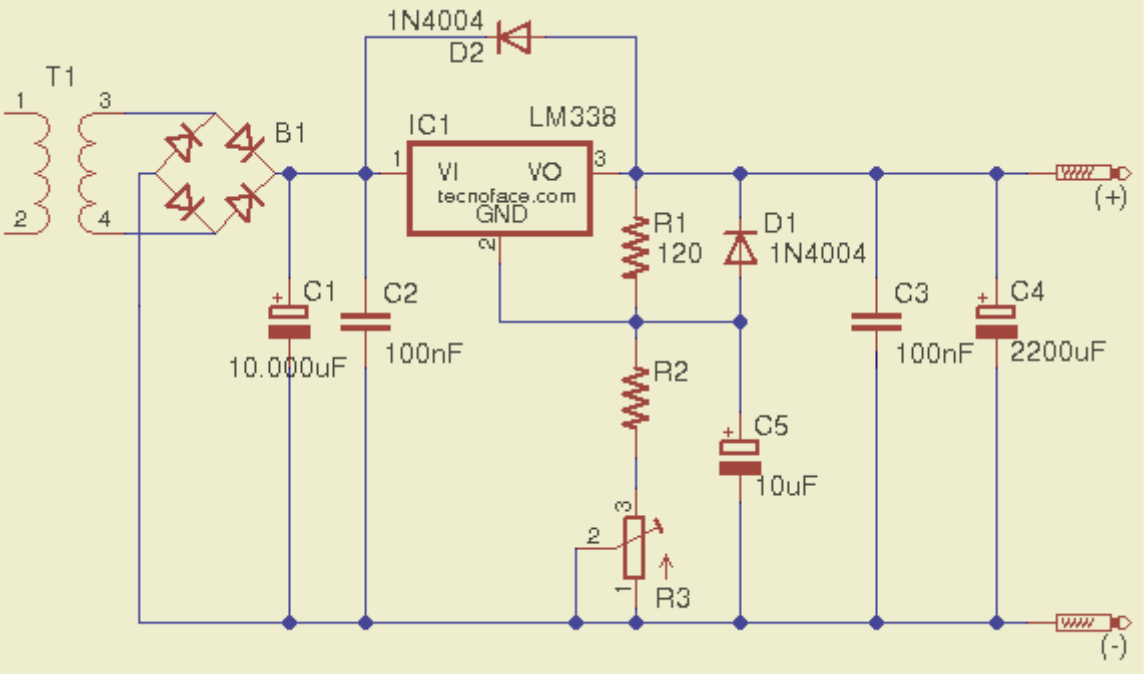


Fig.8.1. Esquema eléctrico fuente de tensión.

El diseño se basa en el bien conocido circuito integrado LM338 - regulador lineal de National. Este integrado nos va proveer una tensión fija a la salida, dada por la relación de las resistencias R1 y R2. En la entrada tenemos el transformador T1, puente rectificador B1 y un capacitor electrolítico C1 de 10.000 micro faradios. El capacitor C1 se encarga de mantener la tensión lo suficientemente continua en la entrada del regulador. En la salida del regulador vemos en serie las resistencias R1 y R2 que definen el valor de la tensión de salida, y el preset de ajuste R3. Los capacitores C2, C3 y C5 son recomendados por el fabricante para eliminar el posible ruido y mantener el circuito estable (bypass). El capacitor C4 ayuda a mantener la tensión de salida constante ante bruscas variaciones de carga.

Tensión de salida

Como ya mencionamos antes, la tensión de salida depende del valor de las resistencias R1, R2 y R3. En este diseño, el valor de R1 es fijo, 120 ohm. R2 y R3 nos van a determinar de qué voltaje será la fuente. En la siguiente tabla se presentan los valores ambos resistores, como también el valor del transformador de entrada.

Tension de salida	*T1*	*R2*	*R3*
5 v	9 v	300	100
9 v	12 v	680	150
12 v	15 v	910	200
15 v	17 v	1200	270
24 v	24 v	2000	470

Para el caso de la fuente que se quiere diseñar de 9 V, según la tabla anterior, necesitamos un transformador con una tensión en el secundario de 12 V, R2= 680 Ω y R3= 150Ω. R3 es un preset o potenciómetro variable que debemos ajustar una vez ensamblada la fuente para que a la salida tenga el valor de tensión deseado. Para poder tener el rango de ajuste adecuado, se debe implementar un preset del valor que se indica en la tabla para cada valor de tensión de salida.

Corriente de salida

Veamos, primero, el porqué del alto valor en el capacitor de entrada C1. La tensión de entrada proviene de la línea de corriente alterna, que varía en el tiempo. A la salida del puente rectificador la tensión varía entre el valor pico máximo y 0v. Esta variación es periódica: la tensión crece desde los cero voltios hasta alcanzar el valor máximo, luego baja hasta cero, luego vuelve a crecer y así sucesivamente. Este ciclo se repite 100 veces por segundo (considerando que la frecuencia de la tensión de línea es de 50Hz). La tensión de entrada estará entregando corriente cuando esta próxima a su valor pico. Dicha corriente fluye hasta la salida y además carga el capacitor C1. A medida que la tensión disminuye, cada vez se entrega menos corriente. A partir de cierto valor de tensión, la corriente entregada por la tensión de entrega es nula y toda la corriente de salida se mantiene gracias a la carga almacenada en el capacitor C1. El mismo deberá ser capaz de almacenar la suficiente cantidad de electrones como para poder proveer la corriente necesaria a la salida, de ahí su alto valor.
Debido a este efecto, en los instantes cuando la tensión de entrada pasa por los valores máximos de tensión, el circuito estará tomando un importante valor de corriente desde la línea de CA: deberá reponer toda la carga que perdió el capacitor, como también continuar aportando corriente de salida. En dichos instantes, se producen picos de

corriente muy altos en la entrada del circuito, en la tabla siguiente presentamos los picos de corriente en la entrada, para diferentes valores de la corriente de salida:

Corriente de salida	Corriente de entrada máx
5 A	16 A
4 A	13 A
3 A	10 A
2 A	7 A
1 A	3,5 A

Podemos observar que estos picos de corriente en la entrada superan en más de tres veces la corriente de salida. Debemos tener mucho cuidado al respecto. En primer lugar, el puente de diodos sería muy recomendable utilizar puentes de diodos integrados, es decir, un componente de 4 terminales que ya tiene los 4 diodos en su interior. Estos dispositivos normalmente se piden por la máxima corriente que soporta en forma continua. Además, el fabricante específica cual es el pico de corriente máxima que soporta el dispositivo y por cuánto tiempo (por ej., 20A durante 8,3ms).

Lamentablemente, estos datos varían de fabricante en fabricante, por lo que, para una fuente de 5A a la salida, los puentes de 5A de diferentes fabricantes pueden tener distinta especificación de corriente máxima.

Si estaríamos diseñando la fuente para una producción de miles de equipos, deberíamos estudiar en detalles estos datos y encontrar un componente más económico que satisface los requisitos del diseño. Pero, nosotros fabricaremos una sola fuente para uso personal, donde el ahorro de unos pocos centavos no tiene sentido. A cambio, tendremos un equipo altamente seguro, con muy bajas portabilidades de falla. Así que ahí va mi consejo: utilicen el puente de diodos de un valor igual o mayor a la corriente máxima indicada. Por ejemplo, para una corriente de salida de 3 A, que es la deseada, no sería descabellado utilizar un puente de 15A.

Algo similar sucede con el transformador, solo que ahora no vamos a ser tan exigentes. Normalmente, dejando 1-2A de seguridad es suficiente, así que para nuestro caso con un transformador de 4 ó 5 A, estaría bien. Algunas fuentes traen transformadores de igual valor a la corriente de salida. Sin embargo, no es muy recomendable, y es poco probable que puedan funcionar en forma segura durante largo rato.

El disipador

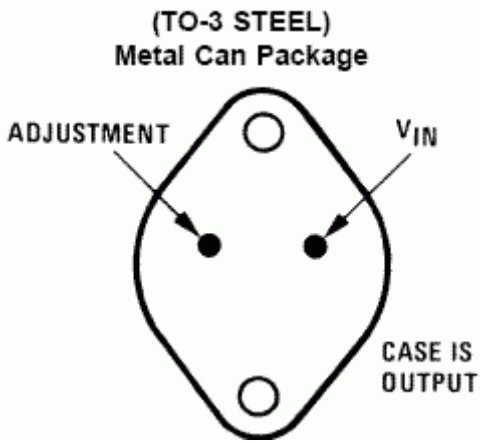
Otra cosa muy importante, poder disipar el calor del regulador LM338. La regla de siempre también vale aquí: cuanto más grande - mejor. Para el caso de 5A es recomendable un disipador de 10 x 15 cm, con 3 o 4 aletas de 3 cm a lo largo sería suficiente. También se puede usar un disipador para microcontrolador de alguna PC antigua que está en desuso, y de paso, aprovechamos el ventilador. Es recomendable, una vez ensamblada la fuente, colocarle una carga importante, en lo posible, que consuma una corriente próxima al valor máximo de salida. Deberíamos dejarla funcionar por un tiempo y verificar que el regulador no calienta demasiado. Si puedes apoyar el dedo sin quemarte, el disipador está bien dimensionado.

Vale la pena aclarar cuáles son los parámetros que hacen que el regular se caliente. Obviamente, uno de ellos es la corriente, a mayor corriente de trabajo, mayor es la temperatura. Pero, el otro parámetro no menos importante es la diferencia de potencial. Presten atención al siguiente detalle: en la entrada del regulador se aplica una cierta tensión, y a la salida tenemos otro. Cuanto mayor es la diferencia de tensión entre la entrada y salida del regulador, más potencia debe disipar y por lo tanto, genera más calor. Normalmente, la tensión en la entrada del regulador debe ser 3 V mayor que la de salida.

Además, el fabricante asegura el correcto funcionamiento del regulador cuando la diferencia de potencial no supera los 35 V entre la en entrada y la salida. Esto nos da un importante margen de maniobra. Por ejemplo, podríamos usar un transformador de 15v para construir una fuente 5v. Pero, qué pasa con la diferencia de tensión restante? Pues se aplica entre la entrada y la salida del regulador, aumentando la cantidad de calor generada y el tamaño del disipador. Por lo tanto, es muy recomendable utilizar los transformadores según se indica en la tabla anterior, justamente para prevenir el sobrecalentamiento.

El regulador

No hay mucho que decir sobre el regulador. Solo debemos tener cuidado con el siguiente aspecto: el regulador se provee en dos tipos de encapsulados: TO-3 y TO-220. El primero tiene una gran ventaja sobre el segundo: disipa mucho mejor el calor. Para que tengan una idea, si un TO-3 se calienta 10ºC, el TO-220 se calienta 40º para la misma potencia. La diferencia es más que notable, por eso es muy recomendable siempre que se pueda utilizar el encapsulado TO-3. En la siguiente figura se puede ver el pinout del regulador (visto desde abajo). Tenemos:



- Patilla Vin - es la tensión de entrada. Se conecta a la salida del puente. En el circuito se indica como (1)
- Patilla Adjustment - es la pata de ajuste o tierra (GND), se conecta al resistor R1 y R2. En el circuito se indica como (2)
- Carcasa del encapsulado - es la tensión de salida. Se conecta al terminal (+). En el circuito se indica como (3).

Para resumir veamos en la siguiente tabla los valores de todos los componentes necesarios para la construcción de la fuente de 9 V 3 A, según el esquema eléctrico de la figura 8.1.

Componente	Uds.	Valor
T1 transformador	1	220/12V- 4A
BR1 puente de diodos	1	I_{pico}=15 A
C1	1	10 000uF-16 V
C2 C3	2	100nF-50V
C4	1	2200uF
LM338 encapsulado TO-3	1	-
R1	1	120Ω 1/8W
R2	1	680Ω 1/4W
R3 potenciómetro	1	150Ω
D1 D2 1N4004	2	-

Anexo II. Hoja de características BD677A.

1 Absolute maximum ratings

Table 2. Absolute maximum ratings

Symbol	Parameter		Value			Unit
		NPN	BD677 BD677A	BD679 BD679A	BD681	
		PNP	BD678 BD678A	BD680 BD680A	BD682	
V_{CBO}	Collector-base voltage (I_E = 0)		60	80	100	V
V_{CEO}	Collector-emitter voltage (I_B = 0)					
V_{EBO}	Emitte-base voltage (I_C = 0)		5			V
I_C	Collector current		4			A
I_{CM}	Collector peak current		6			A
I_B	Base current		0.1			A
P_{TOT}	Total dissipation at T_{case} = 25°C		40			W
T_{stg}	Storage temperature		-65 to 150			°C
T_J	Max. operating junction temperature		150			°C

Note: For PNP types voltage and current values are negative

Electrical characteristics

(T_{case} = 25°C; unless otherwise specified)

Table 3. Electrical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
I_{CEO}	Collector cut-off current (I_B = 0)	V_{CE} = half rated V_{CEO}			0.5	mA
I_{CBO}	Collector cut-off current (I_E = 0)	V_{CE} = rated V_{CBO} V_{CE} = rated V_{CBO} T_c = 100 °C			0.2 2	mA
I_{EBO}	Emitter cut-off current (I_C = 0)	V_{EB} = 5 V			2	mA
$V_{CEO(sus)}$(1)	Collector-emitter sustaining voltage (I_B = 0)	for BD677, BD677A, BD678, BD678A I_C = 50 mA	60			V
		for BD679, BD679A, BD680, BD680A I_C = 50 mA	80			
		for BD681, BD682 I_C = 50 mA	100			
$V_{CE(sat)}$(1)	Collector-emitter saturation voltage	for BD677, BD678, BD679, BD680, BD681, BD682 I_C = 1.5 A I_B = 30 mA			2.5	V
		for BD677A, BD678A, BD679A, BD680A I_C = 2 A I_B = 40 mA			2.8	
V_{BE}(1)	Base-emitter voltage	for BD677, BD678, BD679, BD680, BD681, BD682 I_C = 1.5 A V_{CE} = 3 V			2.5	V
		for BD677A, BD678A, BD679A, BD680A I_C = 2 A V_{CE} = 3 V				
h_{FE}(1)	DC current gain	for BD677, BD678, BD679, BD680, BD681, BD682 I_C = 1.5 A V_{CE} = 3 V	750			
		for BD677A, BD678A, BD679A, BD680A I_C = 2 A V_{CE} = 3 V				

1. Pulsed duration = 300 ms, duty cycle ≥1.5%.

➢ Referencias.

-http://wechoosethemoon.es/2011/arduino-matlab-simulink-controlador-pid/
-http://es.wikipedia.org/wiki/Puerto_serie
-http://es.wikipedia.org/wiki/UART
-http://es.wikipedia.org/wiki/USB
-http://arduino.cc/en/Reference/HomePage
-http://arduino.cc/it/Reference/Board
-http://arduino.cc/en/Main/arduinoBoardUno
-Ayuda Matlab/Simulink