***Obligatorio sistemas de control***

Prof.Andre Fonseca

Sistemas de control

Montevideo Uruguay 2022

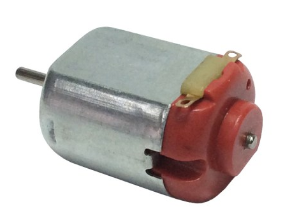
***Control de par motor-generador***

Figure 2

Marcos Curbelo (242549), Ignacio Olazabal (239937) y Nehuen Caroll (225459)

Universidad ORT Uruguay

***Resumen*—Este documento es un informe que investiga el control de un par generador motor. Se detalla el armado y aplicación de dos distintos sistemas de control y su respuesta ante una perturbación. A su vez se brinda información sobre el armado de la maqueta y el software utilizado.**

Palabras Clave—controlador; Par motor-generador; PID; perturbacion; Arduino; Scilab;

# Introduccion

Esta práctica busca aplicar nuestros conocimientos sobre los sistemas de control, que hasta ahora solo han sido teóricos o simulados. Buscaremos, a su vez, comparar las simulaciones con los datos prácticos obtenidos de una maqueta física. Para el modelo físico se decidió utilizar un par motor-generador, es decir un motor acoplado al generador. Buscaremos controlar la velocidad del motor y mantenerla constante ante perturbaciones. El generador nos permitirá conocer el estado de la velocidad angular del motor, ya que cuanto más rápido gire el motor más tensión generará. Este fenómeno se da gracias a la inducción magnética, proceso mediante el cual flujos magnéticos variables generan campos eléctricos, gracias a la ley de Faraday. Sabemos que el giro del imán del eje del generador produce una variación de flujo magnético dentro del bobinado, donde esta variación de flujo magnético genera una diferencia de potencial llamada fem, es esta diferencia de potencial que nosotros podemos medir en los bornes del generador.

# Experimento

## Construccion del par motor-generador

### Seleccion de motores

Los motores elegidos para esta práctica fueron los motores DC 12V RS-385 (*Fig. 1*). Estos fueron elegidos principalmente por su rápida velocidad de giro, necesaria para generar una fem significativa, y en especial por su alta durabilidad. Este motor alcanza los 9800RPM, exige una corriente nominal de 0.75A y tiene una eficiencia de alrededor 67%.

Figure 1

Se hizo una prueba acoplando un motor RS-385 con un micromotor de 5V(*Fig. 2*), pero como ha pasado en pruebas pasadas, se generaba un voltaje muy bajo de apenas 0.6V máximo, y se rompían con extrema facilidad.

### Acople entre motores

Se llevo a cabo una extensa investigación acerca de cómo se acoplan los motores en la industria. Los acoplamientos más utilizados son:

* Engranajes
* Correas
* Unión directa

Se tenían a disposición varios tipos de engranaje para lograr el acoplamiento. Un problema que surgió fue que los engranajes estaban diseñados para motores de 5V como el de *Fig. 2*, por lo tanto para lograr encastrarlos en el eje del motor se debieron agrandar los agujeros centrales de los engranajes con un taladro. Esto deformo leve pero significativamente el engrane central del engranaje, volviendo difícil una solución de engrane. Luego, a su vez se tenían poleas para acoplar los motores mediante correas como el alternador de un auto, pero esto no era fiable ya que las “correas” disponibles eran gomas elásticas poco duraderas y confiables.

Por estas razones se opto por un acoplamiento directo. Uno tendería a pensar que el motor y el generador se podrían acoplar directamente soldando sus ejes y listo. Si bien esta es una muy sencilla solución teórica esta es muy difícil de aplicar en la realidad, ya que siempre se presentan 2 tipos de errores de alineación, angular y paralela (*Fig. 3*).

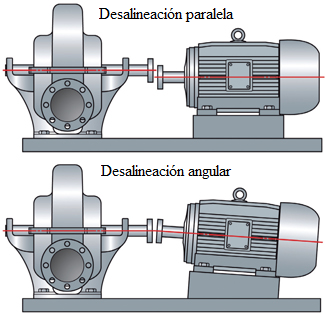
Estas desalineaciones causan una fricción excesiva dentro del motor que no solo no lo dejan girar correctamente, lo que lleva a que no generen una buena tensión, sino que lo desgasta rápidamente rompiéndolo.

Figure 3

En la industria se utilizan alineadores laser entre otras cosas, para evitar este desgastamiento.

Sin embargo la solución más sencilla a este problema es utilizar una unión lo suficientemente elástica para compensar cualquier desalineación. De nuevo en la industria, se utilizan pernos rodeados de goma y goma extra entre las uniones.

Tomando todo esto en cuenta se crearon dos pruebas de la cual se selecciono una. Se probo poner un engranaje en el extremo de ambos ejes, se le hizo tres agujeros a estos, y se los unió con tornillos con tuercas, pero no se los aprieto de manera que estos pudiesen moverse un poco para lograr compensar por desalineaciones. Sin embargo esto obtuvo resultados mixtos, ya que funcionaba pero sufría mucho por las vibraciones y era poco confiable.

Esto nos lleva a nuestro diseño final. Al igual que antes se coloca un engranaje en el extremo de los ejes, pero esta vez se los une pegando goma eva entre ellos, se usó uhu ya que al secarse no se endurece sino que es a su vez flexible (*Fig.4*). La goma eva cumple tres funcionalidades:

* Unir los motores
* Compensar errores de alineación
* Absorber vibraciones

también se probaron materiales como esponja pero la goma eva probo tener mejores resultados.

Por ultimo se creo una base 3D para los motores de la misma medida (*Fig. 5*). Esta fue necesaria para poder montar el sistema en una base, ya que la forma circular de los motores no es buena para adherirla a la misma. Los motores a las bases se adhirieron con “poxipol 10 minutos” y las bases a la base de madera también se pegaron con el mismo pegamento. Se probo la utilización de tornillos, pero el atornillado introducía errores de alineación. Shape

Description automatically generated

### Resultado final y prueba final de acople

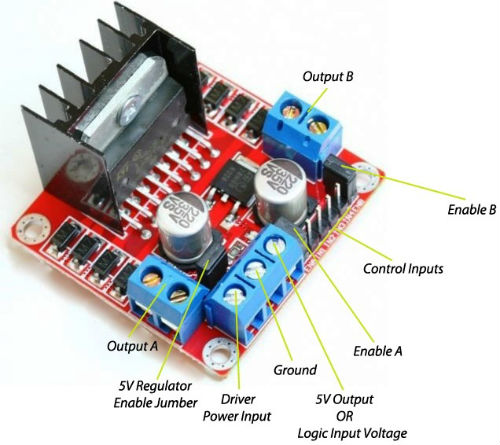
Finalmente se probó la maqueta suministrando uno de los motores con un generador de 12V. En los bornes del generador se logró medir una tensión de alrededor de 10V. Esta tensión es mas que suficiente para el experimento, por lo tanto concluimos el armado del sistema físico como exitoso.

## Conexión del sistema fisico con scilab y control con arduino

### Metodo de control del motor

Para controlar la velocidad del motor DC, se utilizará un control por frecuencia de pulso o señal PWM. Si conectamos un motor a una tensión constante este girara a la máxima velocidad que pueda a esa tensión. La idea de un control por frecuencia de pulso es enviar una señal periódica cuadrada con parte de su periodo encendido y la otra parte apagado. Variando el tiempo de apagado y encendido es que podemos variar la velocidad del motor. El problema es que la señal PWM del Arduino es demasiado pequeña como para controlar el motor por esto es que precisamos un Driver.

### Eleccion de driver de motor y conexión a arduino

El driver nos permite controlar el motor de manera accesible. La idea es conectar una fuente de tensión a este, y utilizar el PWM como control habilitando y deshabilitando la fuente de tensión. Esto se puede lograr con un MOSFET, pero para simplificar en esta práctica se utilizó un driver diseñado para Arduino que cumple con todas estas funcionalidades. El driver utilizado cuenta con un puente H contenido en el integrado L298n (*Fig. 6*). Este puente H se utiliza para controlar la dirección de giro de los motores y su potencia. La placa del driver cuenta con una entrada para voltajes mayores a 5V, a la cual le suministraremos potencia con el transformador. La placa también cuenta con un llamativo disipador de calor de color negro.

El módulo L298 cuenta con 2 puentes H y es el responsable de controlar los motores con las indicaciones del Arduino.

Un puente H en esencia se compone de un motor DC dispuesto en el centro, a cuyos bornes se conectan dos cables y luego 2 interruptores en paralelo. Dichos interruptores se conectan a una fuente de alimentación.

Esta estructura nos permite regular hacia donde girará el motor. Si tomamos el “cuadrado” formado por los interruptores (con la fuente de alimentación a la izquierda con el positivo hacia arriba) Fig.1 y activamos los de la diagonal principal Fig.2 la corriente entrará por la izquierda del motor y saldrá por el borne de la derecha, cerrando así el circuito con la fuente. Esto hará que el motor visto desde arriba gire en sentido horario. Activando la diagonal secundaria obtendremos el efecto opuesto. Fig.3 . El modelo real de un puente H es más complejo y utiliza diodos y transistores para su funcionamiento, pero investigar su funcionamiento más a fondo escapa el objetivo de este documento.

El puente H también nos da la posibilidad de cortocircuitar los bornes del motor, frenándolo así bruscamente. A su vez este módulo, nos permitirá regular la potencia de los motores utilizando una señal PWM.

El dispositivo necesita de 3 conexiones a la placa Arduino por motor, una entrada In1, una In2 y una Ena. Si se envía un ‘1’ lógico por In1, el motor girará en sentido horario y, al mandar uno por In2, se abrirán las compuertas para que el motor gire de manera antihoraria. Y por último Ena describirá la potencia en la que se desea operar al motor, este es un número del 1 al 255, siendo el ultimo el máximo.

Comenzamos conectando el controlador al transformador mediante una protoboard, conectándolo al Vin de 12V y conectando los GND. Luego conectamos el In1 e In2 a los pines 5 y 4 del Arduino, luego conectamos el ENA al pin 6 y conectamos el motor con los bornes del driver.

### Coneccion del generador a arduino

Al conectar los bornes del generador al pin analógico del Arduino nos enfrentamos a 3 posibles problemas:

* Polaridad de la salida
* Tensiones mayores a 5V
* Ruido

#### Polaridad de la salida

Tomando los bornes del generador debemos ser muy cautelosos de no conectar el negativo al pin analógico ya que esto podría dañar la placa, por lo tanto antes de la conexión se elige un sentido de giro y no se cambia por el resto de la práctica, ya que el sentido de giro dicta la polaridad de los bornes por la regla de la mano derecha.

Este sentido depende de las entradas de driver, y nosotros utilizamos el IN2 del driver prendido y IN1 a 0V.

#### Tensiones mayores a 5V

Cuando el motor es accionado a la máxima capacidad del sistema, con las perdidas del driver, llegamos a obtener alrededor de 10V. El pin analógico que comunicará la salida del generador con el conversor A/D solo podrá recibir máximo 5V, esto significa que no podemos conectarlo directamente. Para esto conectamos dos resistencias en paralelo de igual valor, ya que dos resistencias iguales generan una resistencia de 0.5ohm.

Por lo tanto tomando la ley de ohm manteniendo el voltaje de la fuente constante, la corriente de llegada al motor será ahora la mitad de lo que solía ser. Esto lleva a que si el motor a potencia máxima inducia 9V en el generador, ahora inducirá 4.5V máximo, voltaje seguro para el Arduino.

#### Ruido

Al poner a prueba nuestro sistema de motores la salida que se detecto es la de la figura XX, como se puede apreciar, gran parte de la señal es ruido.

A picture containing shoji, antenna

Description automatically generated

Para solucionar este problema se añadió un filtro RC. Este se calculo sacando la frecuencia del ruido al contar el numero de picos en un segundo de la figura XX.

Chart, bar chart

Description automatically generated

Esto nos dio un valor de frecuencia de ruido aproximado de 28Hz. Con esto buscamos un conjunto de resistencias y capacitores que cumplan con la relacion:

Nos conformamos con un capacitor de 33nF y un paralelo de resistencias de 220k ohm, lo cual no cumple con la relacion mostrada anteriormente, pero da una respuesta con menos ruido (figura XXY) que nos deja evaluar la salida de mejor manera.

A picture containing line chart

Description automatically generated

Chart, line chart

Description automatically generated

Buscaremos filtrar este ruido con un pasa bajos, formado por un circuito RC. Tomaremos la señal de entrada y la pasaremos a frecuencia con una FFT

### Utilizacion de Xcos para obtencion de datos y control

Para obtener los datos del sistema y controlarlo se usó un programa para Arduino llamado toolbox. Esta nos permite conectar el Arduino con Scilab en el cual usaremos su función de Xcos (Fig. XX). Aquí podremos enviarle instrucciones al Arduino y recibir información de este. Aquí también es donde se construirán los controladores, estos tomaran la información Diagram

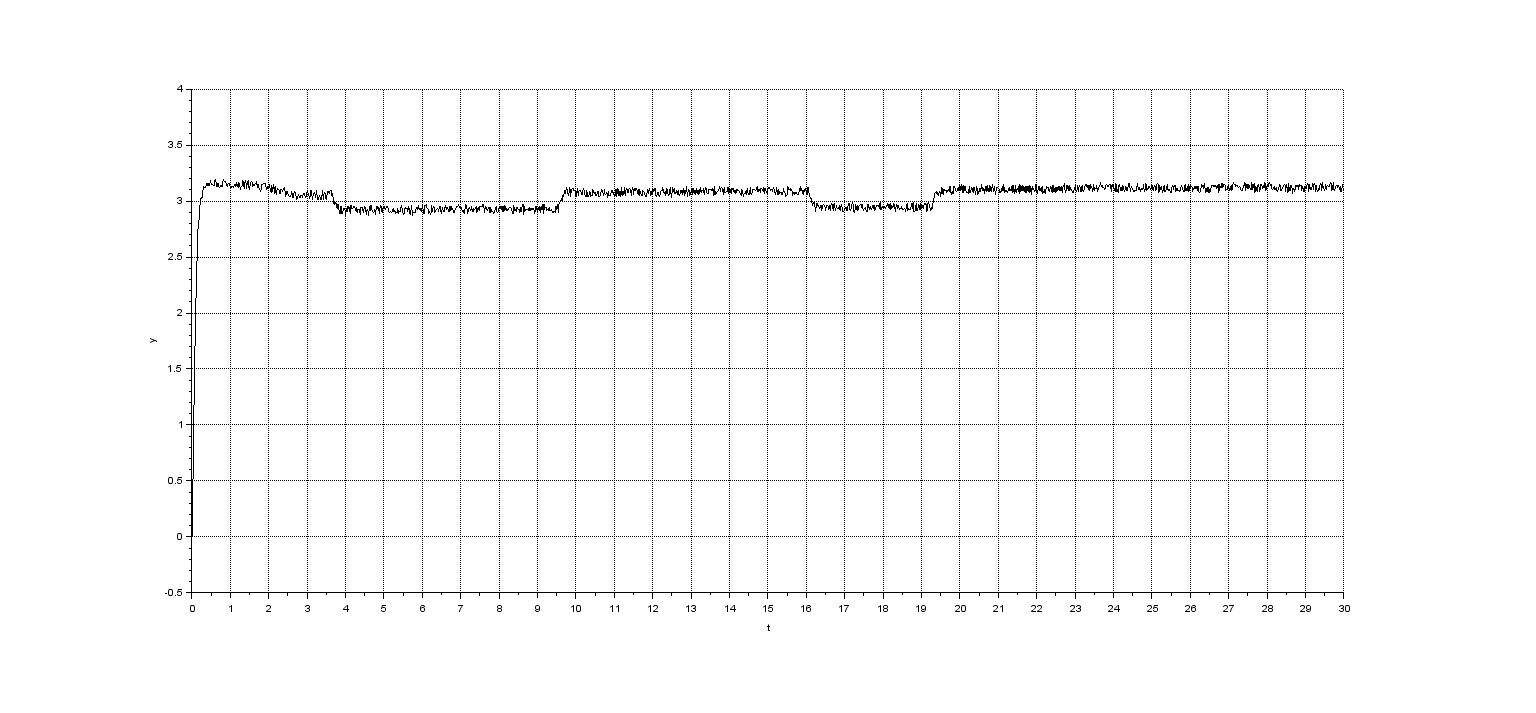
Description automatically generatedde salida del generador y regularan la velocidad del motor con la señal PWM.

## Obtencion de la transferencia del sistema fisico

La obtención de la función de transferencia se realizó primero de manera teórica y luego práctica. Para ello se realizó el modelo teórico del sistema a fin de obtener una idea de cómo sería la transferencia. Dicho modelo se encuentra detallado en el Anexo 1, del cual obtenemos que el sistema tiene una transferencia del siguiente tipo:

Donde GDC es la ganancia de continua y la constante de tiempo. La ganancia se obtuvo simplemente con la relación de entrada-salida del sistema cuando a la entrada inyectamos una señal continua de 5V

Luego procedimos a cambiar la señal de entrada por una señal compuesta por una señal cuadrada montada sobre una continua de forma de operar en la zona de trabajo del sistema, la cual se explica más adelante. De esta forma lo que se busca es observar la respuesta al escalón del sistema. La respuesta obtenida es la siguiente:



Mediante la herramienta “To Workspace” de Xcos se guardan los valores de dicha gráfica, en un vector que contiene una tupla el tiempo y otra tupla con los valores medidos, para poder operar con ellos.

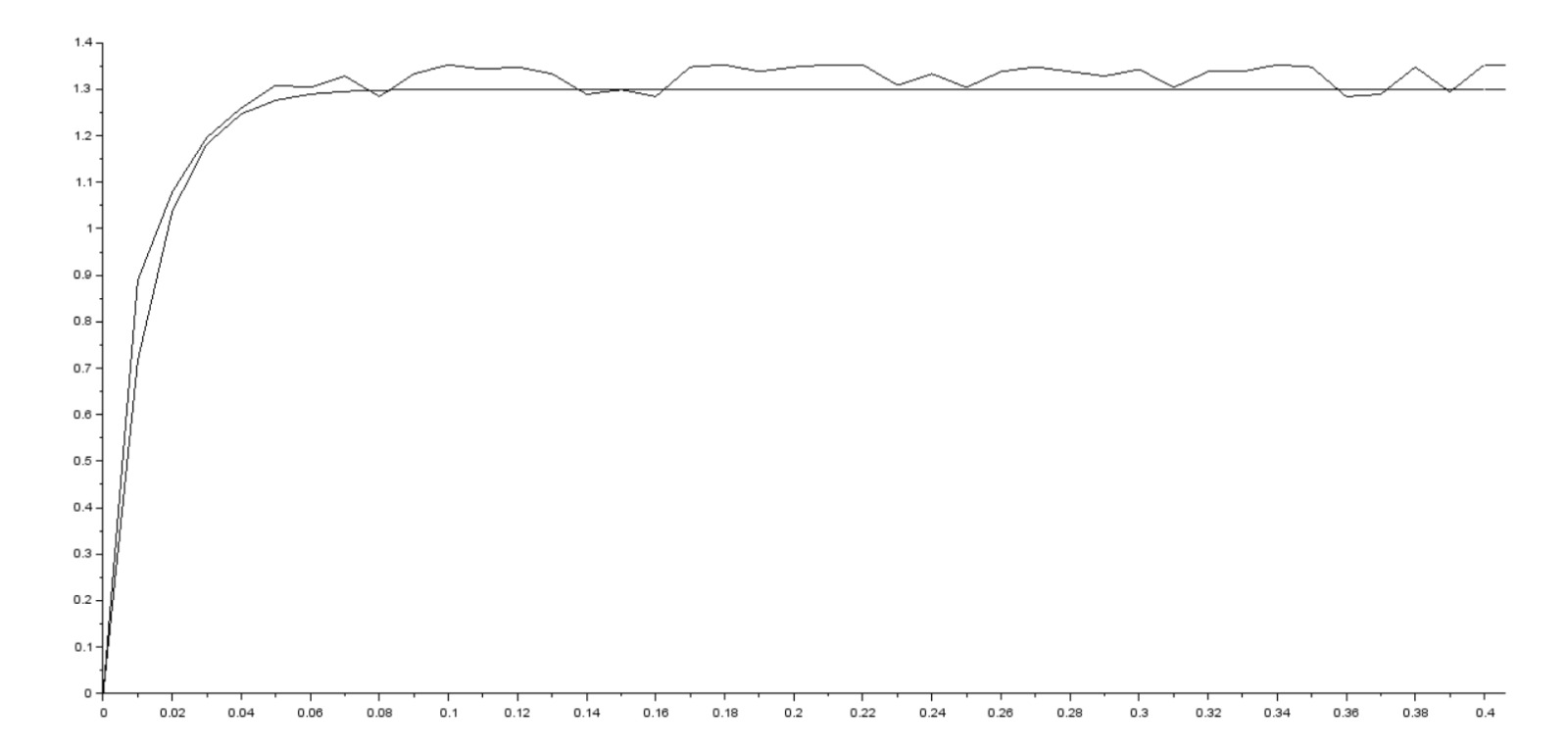
Se observa que a excepción del primer periodo los demás son muy similares, y esto se debe a que en el primer periodo los transitorios todavía son significativos. Por eso se decide utilizar el tercer periodo para medir la constante de tiempo.

Como los valores están almacenados en un vector, utilizando funciones de scilab se procede a recortar los datos separando la respuesta al escalón de subida del tercer periodo. En este punto se realiza una estimación de los valores de ganancia en continua y tau del sistema. Luego se traslada los valores obtenidos al origen restando los valores mínimos de tiempo y de valor. Con esto, se busca utilizar la función datafit que nos permitirá obtener la constante de tiempo tau de nuestro sistema.

En el anexo 2 se demuestra la respuesta al escalón de subida utilizado, el cual se le impone como función a datafit, además de una función de error y un vector con los valores estimados obtenidos anteriormente. El resultado obtenido es el siguiente

|  |  |  |
| --- | --- | --- |
|  | Ganancia | Tau |
| Estimado | 0.83 | 0.0125 |
| Datafit | 0.83 | 0.0125 |

Los datos obtenidos de manera visual y con la función datafit coinciden, y además, la función datafit nos aportó que el error promedio era del orden del 0,21%, lo que nos verifica el correcto procedimiento. A continuación, se muestra la comparación entre el modelo real y el calculado:



Por lo tanto, la función de transferencia de nuestro sistema es

## Diseño de la carga removible

Para evaluar el sistema de control debemos introducir una perturbación en el sistema, y este debe volver a su posición original (set point). Esta perturbación la haremos agregándole una carga al sistema, le colocaremos una resistencia en paralelo al motor de esta forma le llegara menos corriente al motor. La carga será encendida o apagada con un transistor desde el Arduino, se utilizará el pin 13 otorgarle corriente a la base, a la cual de protección le ponemos una resistencia de 1kohm por si acaso.

La resistencia utilizada estará compuesta por tres resistencias de 560ohm en paralelo, lo cual otorga una resistencia total de alrededor 187 ohm (186.667). Utilizamos esta composición ya que las resistencias son de ¼ whatt, y por lo tanto podrían sobrecalentarse gracias a la potencia del generador. De esta manera cada resistencia deberá soportar 1/3 de la potencia original. Podemos apreciar por lo tanto los efectos de la carga sobre el sistema (Figura XXX).

## Zona de trabajo y problema de Scilab & Arduino

Al hacer las pruebas iniciales del sistema se detecto un problema en la salida del PWM. Este problema se puede ver en la figura xXX, donde al probar el aumento de tensión en los motores de manera consistente, medimos la tensión generada por estos y vemos una clara zona muerta entre 2.5V y 3.2V.

Al analizarlo con mas detalle nos dimos cuenta de que muy probablemente el problema se da por la relación entre Xcos de Scilab y Arduino, al hacerse mas pruebas concluimos que la zona muerta se da en la salida del PWM, y se encuentra en el intervalo de 140-155 pwm.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Para solucionar el problema y evitar la zona muerta, se tomo la zona de trabajo entre 160-255 pwm, donde el punto medio de trabajo va a ser a 207 pwm.

Esta es la razón por la cual en el circuito se mide un divisor de tensión de dos resistencias iguales de 220k ohm, porque a 160pwm la tensión generada era 4.5V y a 255 pwm la tensión aumentaba a 10V aproximadamente. Como se explico anteriormente la entrada analógica de Arduino no aguanta más de 5V, así que para poder trabajar en nuestra nueva zona tuvimos que agregar un divisor de tensión.

## Circuito y Xcos de obtencion de datos

El circuito utilizado de Arduino es el que se puede apreciar en la figuraXXX, donde el motor en la imagen es el generador. Los pines 6, 5 y 4 van al driver como se comentó anteriormente, de estos el 6 tiene la señal PWM, el 5 es 0V y el 4 tiene señal constante. El pin 13 del Arduino es el que activa al transistor y cierra el circuito con la carga de 3 resistencias de 560 ohm en paralelo. El pin A0 es el que detecta la señal analógica de una de las resistencias de 220k ohm en paralelo.

A picture containing text, electronics

Description automatically generated

El circuito del Xcos es el de la figura XXU, en este podemos ver las configuraciones de los pines en scilab para que el sistema trabaje correctamente. Es importante que en este el pin 4 este a 1 y el pin 5 a 0, también el Digital write del pin 13 es el que acciona al transistor.

Diagram

Description automatically generated

## Construccion del PI

El primer control que se creará será un proporcional integrador (PI). Buscaremos que el controlador produzca una respuesta dos veces más lenta que la respuesta original. Para esto se cierra el lazo entre la transferencia y el controlador. Para el controlador solo basta con definir los parámetros Kp y Ti de la siguiente ecuación:

Buscaremos crear el controlador utilizando la herramienta de “lugar de las raíces”, esta nos permite predecir los polos del lazo cerrado según como variemos un “k”. Sabemos que el lazo cerrado tendrá un polo en -1/T que proviene de nuestra transferencia, el cual en nuestro caso T vale 0.0125, un polo en 0 y un cero en -1/Ti. Dependiendo de como nos tomemos Ti (mayor o menor a T) podemos tener dos posibles configuraciones de lugar de las raíces (Fig en anexo). Independientemente de esto nosotros buscaremos tener un polo en -2/T es decir dos veces más lento. Como primer medida nos tomaremos Ti menor a T, alrededor de 3 veces mas chico, ya que en la otra configuración, el polo en -1/T y el cero -1/Ti quedan muy cercanos y pueden llegar a cancelarse, acelerando nuestro controlador por polos de ordenes mas altos. Por esta razón decidimos que Ti tomara un valor arbitrario de 0.0009. Con esto rápidamente podemos calcular el lugar de las raíces que resulta al modificar K (FIG XXX). Diagram

Description automatically generated with medium confidence

Si nos centramos en el polo -1/T podemos observar que se encuentra en el -80 (-1/0.0125), por lo tanto debemos hallar el K el cual genera un polo en -40 (-2/0.0125). Para esto calculamos la transferencia en lazo cerrado del sistema la cual es:

Desarrollando y aislando solo el denominador obtenemos:

Ahora igualamos a cero y despejamos K:

Por último sustituyendo s por -40, obtenemos (el cálculo se hace en scilab para ser utilizado en el xcos sin perder cifras significativas):

Por lo tanto el controlador PI queda definido de la siguiente manera:

## Construccion de regulador de estados

Como segundo controlador utilizaremos un regulador de estados, para probar este método de control buscaremos transformar nuestro sistema a uno subamortiguado con .El regulador de estados se basa en el método de asignación de polos. Este nos asegura que si el sistema es completamente controlable y sus estados son medibles y disponibles a realimentación, “los polos del sistema en lazo cerrado se pueden colocar en cualquier posición deseada mediante una realimentación del estado a través de una adecuada matriz de ganancias de la realimentación de estado”[1]. Es importante aclarar que este método debe emplearse en transferencias que contengan un integrador que implica un error nulo al escalón, por lo tanto debemos incluir un integrador en nuestro controlador, este hará que la transferencia en lazo cerrado pase de ser de grado 1 a grado 2.

Primero comenzamos eligiendo los polos en lazo cerrado dominantes deseados el polinomio ps elegido fue:

Cuyas raíces son:

De (X) podemos verificar que y . Una vez que obtuvimos los polos deseados debemos igualar el polinomio característico de nuestro sistema a este. Para esto se hallaron las matrices del sistema en su forma canonica controlable, pero al ser de primer grado serán bastante sencillas:

Una vez halladas las matrices debemos hallar la matriz

Donde:

Por último solo basta con hallar el polinomio con el determinante de:

Y este determinante lo igualaremos al polinomio deseado. Estos últimos dos pasos se hicieron utilizando scilab, mas especifico con la función “ppol”. Esto nos dio los siguientes valores de k1 y ki respectivamente (notar que se debe tomar el negativo de ki):

# Resultados

## Resultado del control por PI

Primero se verifica el controlador PI de manera teórica utilizando la transferencia obtenida previamente. Se espera que el controlador reduzca la velocidad de la respuesta a la mitad y no tenga un sobretiro más grande que 5V, ya que quemaría el Arduino.

Diagram

Description automatically generated

El circuito de Xcos utilizado es el de la figura XX en este se muestra como se realimenta la señal del sistema, atraviesa el PI con los valores de Kp y Ti calculados anteriormente, seguido de eso atraviesa un bloque de saturación que tiene como limites el 0 y el 255, esto es porque el pwm no puede exceder 255 ni ser negativo. En el circuito podemos apreciar un conjunto de componentes en las entradas los cuales se usan para las pruebas del PI. A continuación, se mostrará como hacer esas pruebas en el Xcos y sus respectivos resultados.

### Funcionamiento basico con carga

Esta prueba es para verificar que el sistema esta legando al punto de estabilidad que le indicamos inicialmente, este como se puede ver en la figura Xcos es 175 pwm que al llevarlo a tensión es 3.43V, esto se comprueba al ver la figura XD vemos que la señal está estabilizada en 3.4V.

Aparte del funcionamiento básico del sistema, también probamos como reaccionaba el PI ante una disrupción, que se genera a partir de la conexión de carga. En la figura XD podemos ver en los momentos donde se cierra y abre el circuito con la carga y cuál es la reacción del sistema. En esta figura la señal de salida es la gráfica roja, la negra es el escalón del ideal y la verde es la entrada al sistema de par motor.

Chart

Description automatically generated

Al analizar la figura XD notamos como el PI modifica la entrada al sistema de par motor para volver a colocar la salida en el punto de trabajo seleccionado.

### Cambio de set point.

El cambio de set point es una prueba para verificar como se adapta nuestro sistema al cambiar el punto de trabajo establecido. Esto en el Xcos se genera con el escalón que este sumando a la entrada esta suma 30 rpm que es un equivalente de 0.6V. El resultado de esta prueba se ve en la figura GG. En esta figura se ve como en el segundo 1 se cambia el set point de 3.4V a 4V.

Chart, line chart

Description automatically generated

### Variacion de velocidad.

### Para confirmar que se cumplen las características dinámicas requeridas del sistema, utilizamos un escalón de 0.5V en régimen y comparamos las velocidades entre nuestra señal y la señal teórica del sistema no realimentado (ver figXcos). Para comparar estas velocidades comparamos el tiempo que demora cada una de las señales en pasar de 10% del nuevo valor de régimen a 90% del nuevo valor de régimen.

Chart, line chart

Description automatically generated

Al hacer estas comparaciones viendo la figuraAAa, el resultado final fue que la señal teórica sin realimentar demora 0.032s en subir y nuestra señal demora 0.056s aproximadamente. Con esto podemos decir que se logro enlentecer al sistema inicial y ahora demora prácticamente el doble en establecerse.

Esta conclusión es aproximada por los efectos del ruido de la señal, y la velocidad de muestreo de scilab, en donde la máxima velocidad de muestreo es de 0.005, y al tener una señal con polos tan rápidos como (s\*0.0125+1) los efectos del muestreo pueden afectar al analizar instancias de tiempo tan pequeñas. Donde si tuviéramos un muestreo más rápido se podría apreciar la diferencia de velocidad con mayor claridad.