UNIVERSIDAD de VALLADOLID

ESCUELA de INGENIERÍAS INDUSTRIALES

INGENIERO TÉCNICO INDUSTRIAL, ESPECIALIDAD EN ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

CONTROL DE POSICIÓN DE UN BALANCÍN CON ARDUINO

Autores:

Martín Ballesteros, Ángel

Del Río Carbajo, Mario

Tutor:

García Ruiz, Francisco Javier — Ingeniería de Sistemas y automática

MAYO – 2013

1. INTRODUCCIÓN

1.1. Resumen

En el presente proyecto se pretende controlar la posición de un sistema mediante el uso de la herramienta simulink de Matlab. El sistema consta de dos barras ancladas a modo de balancín, en uno de los extremos de la barra se encuentra un motor con una hélice que será el encargado de producir la fuerza de empuje proporcional al giro de su hélice que será la responsable de los cambios de posición de nuestro sistema. Para controlar el motor y visualizar un circuito utilizaremos un montaje que incluirá una tarjeta de Arduino Uno.

1.2. Objetivos

- Conocer el funcionamiento de las tarjetas arduino y su entorno.
- Conocer las posibilidades de manejo de nuestro sistema mediante la herramienta simulink.
- Conocer los pasos a seguir para la instalación del software necesario para el funcionamiento del sistema en cualquier equipo.
- Obtención del modelo matemático del sistema a través de las leyes físicas y con datos extraídos de ensayos.
- Elaboración de un sistema en lazo abierto que nos permita darle diferentes entradas al sistema.
- Elaboración de un sistema en lazo cerrado con un controlador PID, que controle de una forma correcta la posición de la barra ante posibles perturbaciones ajenas al sistema.
- Identificar los parámetros de un controlador PID a través del método Ziegler-Nicholls.
- Introducción de mejoras que haga nuestro sistema más estable y fiable.

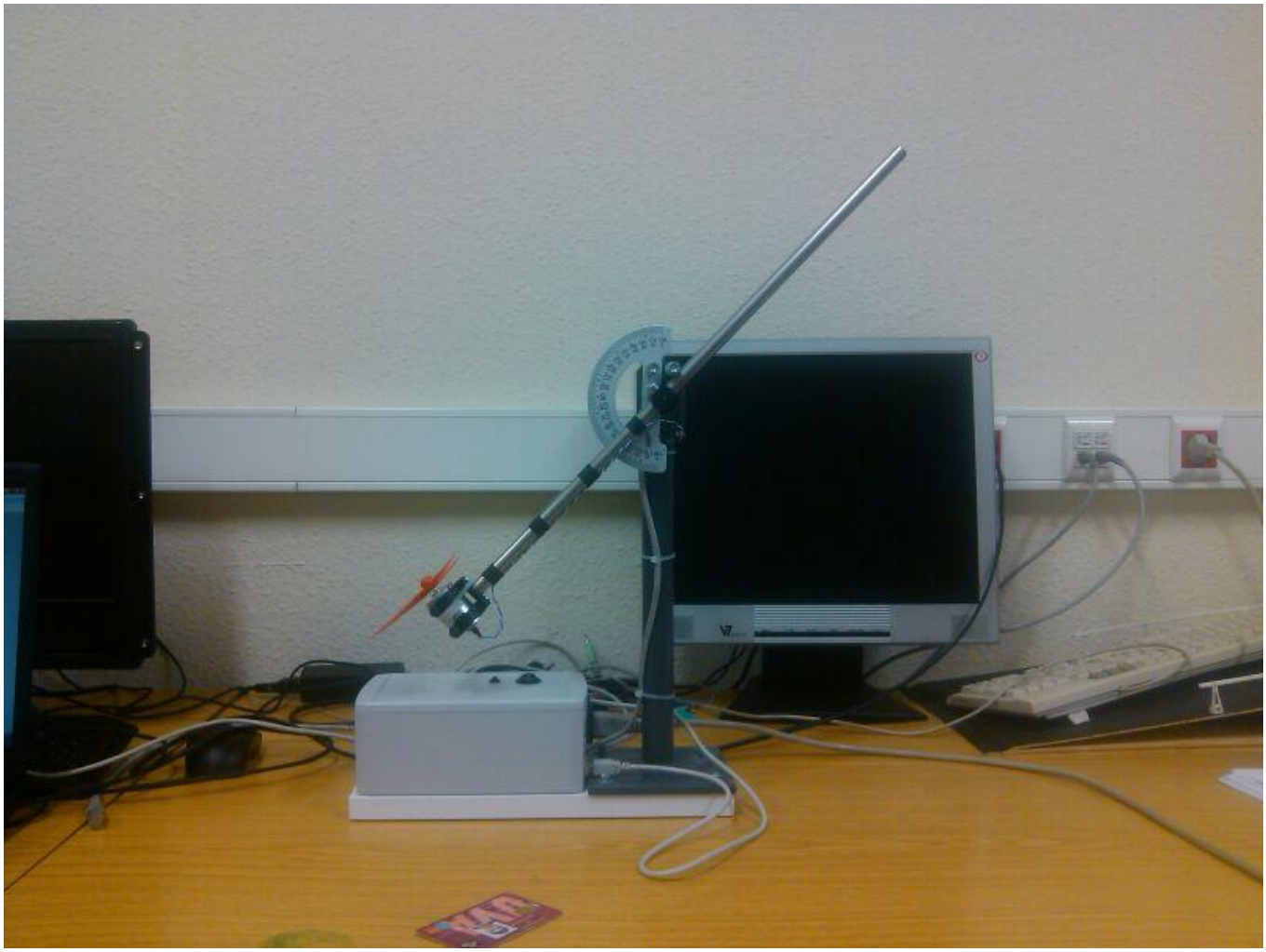
1.3. Descripción de la planta

La planta consta de un sistema compuesto por dos barras, un motor una hélice un soporte y una caja donde están el resto de componentes físicos.

El sistema está colocado tal que encima del soporte se coloco la caja, y una de las barras en posición vertical, la otra está unida a esta permitiendo el giro, solidario a este giro se encuentra un potenciómetro que nos servirá para saber la posición de la barra, en el extremos de la segunda barra se encuentra el motor y pegado al rotor de este está la hélice que es la encargada del movimiento ya que cuanto más voltaje tenga el motor entre sus terminales a más velocidad girara la hélice y la barra cambiara de posición.

Dentro de la caja se encuentra el circuito electrónico formado por una tarjeta de arduino uno, un transistor NPN (BD 139) con un radiador solapado para que disipe el calor y un transformador de

corriente alterna-continua que nos permite conectarlo a la red eléctrica y obtener 24 voltios de cc a través de sus terminales. Este circuito es el encargado de transmitir las órdenes que enviemos a través de simulink y de recibir la posición de la barra para poder visualizar su estado con el ordenador.



Fotografía del dispositivo

1.4. Funcionamiento básico

El funcionamiento se basa en el giro del motor y como consecuencia de la hélice, al girar la hélice esta transmite una fuerza que hace girar la barra móvil (este proceso será estudiado rigurosamente en el apartado de modelo matemático, para saber con exactitud las fuerzas que influyen en nuestra práctica). El motor dependiendo del voltaje que reciba girara más deprisa o más despacio haciendo que nuestra barra al estar unida con un grado de libertad con respecto a la otra barra vertical fija se mueva con un movimiento circular, por ello hemos colocado un transportador de ángulos fijado a la barra, para en todo momento poder visualizar la posición en ángulos y poder comprobar cuál ha sido el desplazamiento del movimiento en cuestión.

Entre las dos barras se ha fijado un potenciómetro que cambiara el valor de su resistencia dependiendo del ángulo de la barra móvil con respecto a la fija. Las dos patillas extremas están conectadas una a tierra y la otra a la señal de 5voltios, la patilla central la conectamos a una de las entradas analógicas de nuestra placa de arduino. Esta conexión es la llamada "regulador de tensión" y lo que hace es que dependiendo de la posición del potenciómetro tendremos diferentes tensiones, siempre entre los límites marcados de 5 voltios límite superior y 0 voltios el inferior.

Fotografía del potenciómetro

En conclusión a través de una salida digital de arduino movemos la barra y a través de una entrada analógica sabemos la posición que ocupa esa barra, pero esto no es tan sencillo ya que el motor necesita un potencial superior al que dispone la placa de arduino que es de 5 voltios y por lo tanto necesitamos un circuito de potencia, que se explicara en profundidad en el apartado" circuito eléctrico"

2. FUNDAMENTOS TEÓRICOS

2.1. PID

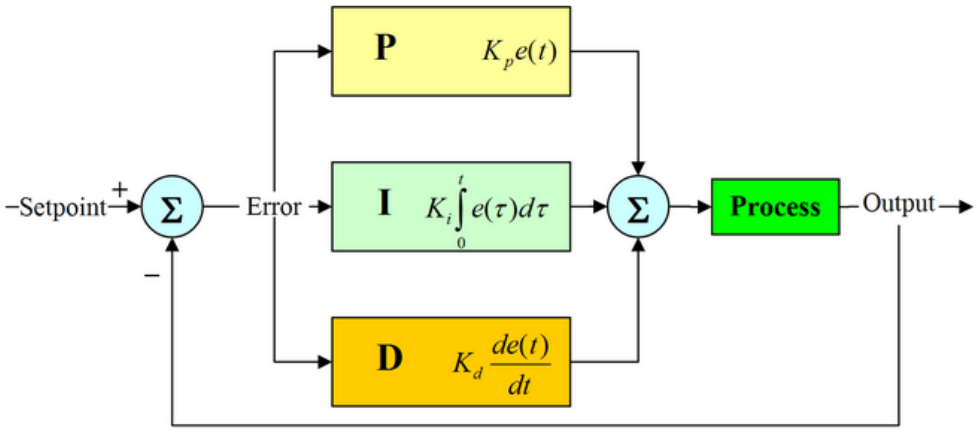
2.1.1. Introducción

El controlador PID es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario, entre la señal de referencia y la señal de salida de la planta, sea cero.

El control PID es con diferencia el algoritmo de control más común, siendo utilizado en el 95% de los lazos de control que existen en la industria.

2.1.2. Estructura del PID

Las tres componentes de un controlador PID son: la acción proporcional, acción Integral y la acción derivativa. A continuación mostramos el diagrama de bloques con el que se representa este controlador.



Estructura PID

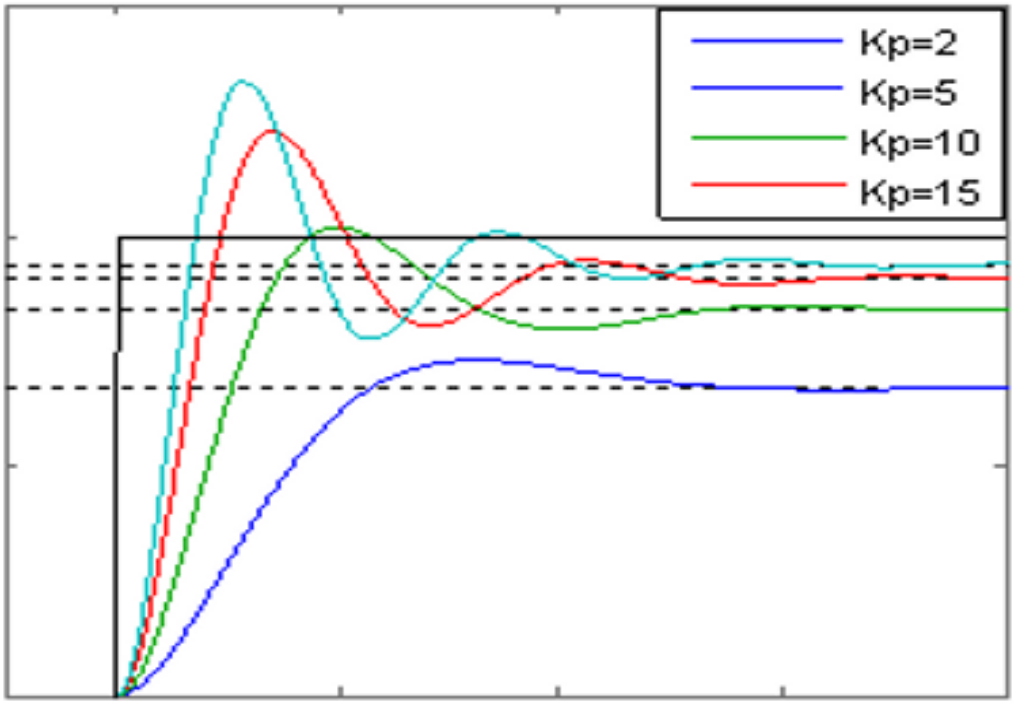
2.1.2.1. Acción de control proporcional

El objetivo de esta acción es que una vez ajustado el error en estado estacionario sea cero respecto a una referencia fija.

La salida que obtenemos de ella es proporcional al error siendo esta $u_{(t)} = K_p \cdot e_{(t)}$, por lo tanto la función de transferencia de la acción proporcional será nada más que una ganancia ajustable.

$$C_p(s) = K_p$$

Esta acción no corregirá el error en estado permanente. En la siguiente figura podemos ver el funcionamiento de un controlador P.



Respuesta de un sistema con diferentes K_p

Como se puede ver en la figura el error estacionario disminuye a medida que la constante aumenta, la velocidad de este tambien en mayor pero las sobreoscilaciones y oscilaciones aumentan tardando más en oscilar.

2.1.2.2. Acción de control integral

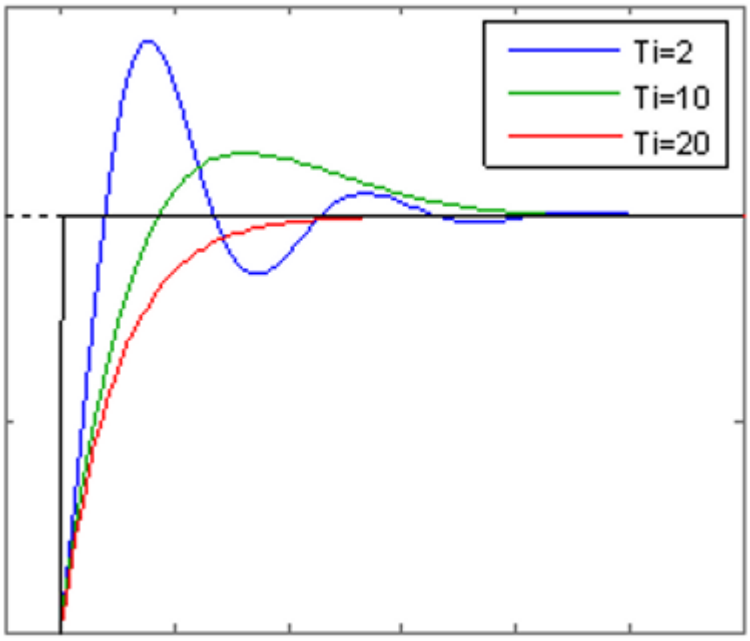
La salida de este controlador es proporcional al error acumulado, por la tanto sera de respuesta lenta. Las fdt de las salida del controlador y del error son:

$$u_{(t)} = K_i \int_0^t e_{(t)}\, dt \qquad C_i(s) = \frac{K_i}{s}$$

La finalidad de esta acción es que la salida concuerde con la referencia en estado estacionario, pudiendo esta cambiar sin tener que cambiar la K_i a diferencia del control proporcional. Se produce un

mejor ajuste que con la acción proporcional por que con esta un pequeño error con el tiempo se hace grande por lo que se tiende a corregir.

Las características de la acción integral se pueden ver en la siguiente figura en la que representamos un control PI en el que variamos la parte proporcional.



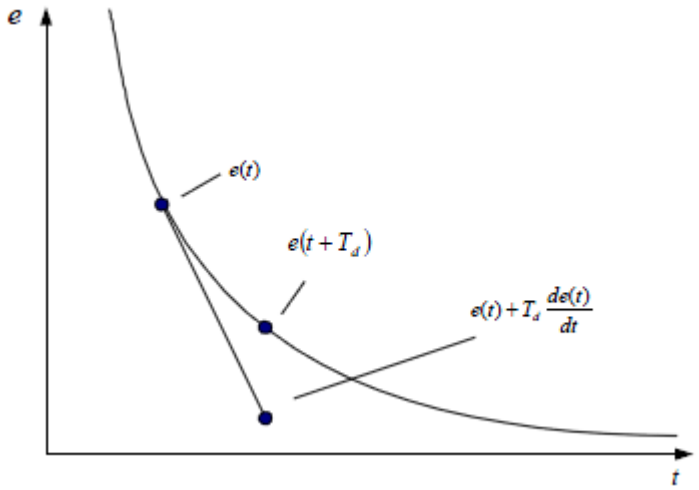
Respuesta de un sistema para diferentes valores de Ti

Siendo $T_1=\frac{1}{k_i}$, por lo tanto cuanto mayor sea la constante mayor será la rapidez del sistemas pero también mayor serán sus amortiguaciones pudiendo llegar a desestabilizarse si esta es desasido grande.

2.1.2.3. Acción derivativa

Esta acción actúa cuando hay un cambio en valor absoluto del error. Por lo tanto no se empleara nunca ella sola ya que solo corrige errores en la etapa transitoria. Es una acción predictible por lo tanto de acción rápida.

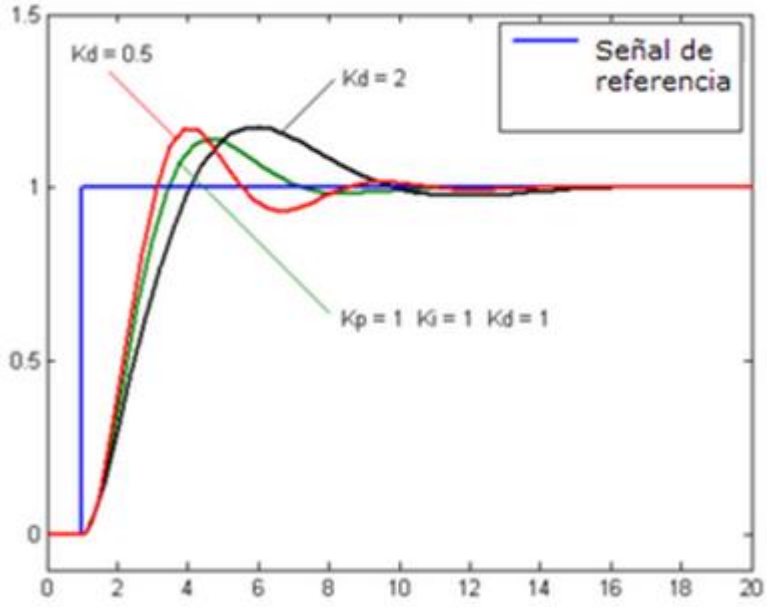
Su objetivo es corregir la señal de error antes de que se haga está demasiado grande. La predicción se hace por la extrapolación del error de control en la dirección de la tangente a su curva respectiva, como se muestra en la figura.



Extrapolación del error de control

Tiene la desventaja de amplia las señales de ruido pudiendo provocar saturación en el controlador. Puede emplearse en sistemas con tiempo de retardo considerables, porque permite una repercusión rápida de la variable después de presentarse una perturbación en el proceso.

Las características de la acción derivativa se pueden ver en la siguiente figura en la que representamos un control PID en el que variamos la parte derivativa.



Respuesta de un sistema para diferentes valores de Kd

2.1.3. Sintonización del controlador PID

No siempre se usan las tres acciones del controlador PID, normalmente suele aparecer la acción proporcional acompañada solamente de la integral o la derivativa. De esta forma conforman el controlador PI y el PD. Estos junto al controlador PID son los que más comúnmente nos encontramos.

El control PI se puede demostrar que funcionara de forma correcta en respuestas donde su dinámica es esencialmente de primer orden.

Para su correcto funcionamiento es necesaria una correcta sintonización. Esto se puede hacer de varias formas, en la actualidad existen programas que te lo ajustan automáticamente, pero también se puede hacer manualmente. Para ello tenemos reglas que nos facilitan esta labor. A continuación veremos las reglas propuestas por Ziegler-Nichols.

2.1.4. Reglas para sintonizar controladores PID de Ziegler-Nichols

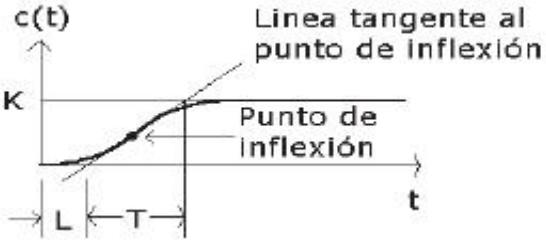
Ziegler y Nichols propusieron unas reglas para determinar los valores de ganancia proporcional K_p, del tiempo integral T_i y del tiempo derivativo T_d.

Existen dos métodos denominados reglas de sintonización de Ziegler-Nichols. En ambas se pretende obtener un 25% de sobrepaso máximo en la respuesta escalón.

Primer método

La respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de S (si la respuesta no exhibe una curva con forma de S, este método no es pertinente). Tales curvas de respuesta escalón se generan experimentalmente o a partir de una simulación dinámica de la planta.

La curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T. El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo y la línea c (t)=K, como se aprecia a continuación.



En este caso, la función de transferencia C(s)/U(s) se aproxima mediante un sistema de primer orden con un retardo de transporte del modo siguiente:

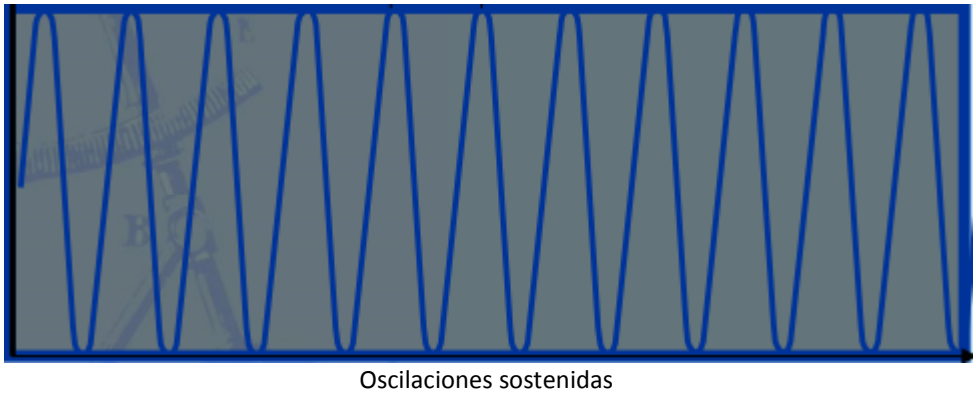
$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1}$$

Ziegler y Nichols establecieron los valores de K_p, T_i y T_d de acuerdo con la siguiente tabla.

Tipo de Controlador	K_p	T_i	T_d
P	$\frac{T}{KL}$	∞	0
PI	$0.9\frac{T}{KL}$	$\frac{T}{0.3}$	0
PID	$1.2\frac{T}{KL}$	2L	0.5L

Segundo método

Este método se realiza con sistema en lazo cerrado.
Primero establecemos $T_i=\infty$ y $T_d=0$. Usando sólo la acción de control proporcional, se incrementa K_p de 0 a un valor crítico K_c en donde la salida exhiba oscilaciones sostenidas (si la salida no presenta oscilaciones sostenidas para cualquier valor que pueda tomar K_p, no se aplica este método).



Oscilaciones sostenidas

Por tanto, la ganancia crítica K_c y el periodo P_c , que es el periodo de estas oscilaciones, se determinan experimentalmente. Ziegler-Nichols sugirieron que se establecieran los valores de los parámetros K_p, T_i y T_d de acuerdo con la fórmula que aparece en la tabla.

Tipo de Controlador	K_p	T_i	T_d
P	0.5 K_c	∞	0
PI	0.45 K_c	$\frac{1}{1.2}P_c$	0
PID	0.6 K_c	$0.5P_c$	$0.125P_c$

2.2. LAZOS DE CONTROL

La regulación automática es la parte de la ingeniera que se encarga del control interno del proceso de una determinada característica, por ejemplo cuando queremos mantener una velocidad de un motor en un valor determinado, recurriremos a la regulación automática para poder logarlo.

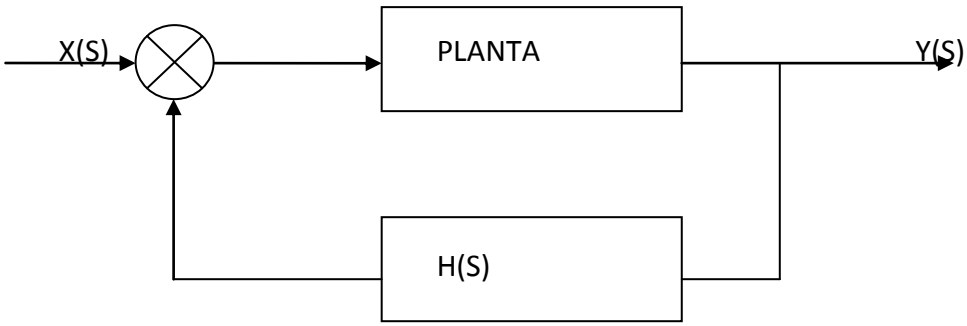
Dentro de la regulación automática se estudian los sistemas de una manera teórica ya que se caracterizan las plantas con ecuaciones matemáticas, para poder usarlas de una manera más sencilla con la posibilidad de usar ordenadores para resolver los problemas.

Una vez que tenemos la planta caracterizada con una función matemática ya tenemos nuestro primer sistema:

Representación sistema lazo abierto

Este sistema se denomina de lazo abierto por su principal característica no es otra que la salida depende únicamente de la entrada a este circuito. No se utiliza demasiado ya que no regula el valor de la salida sino que la planta actúa de una manera atemporal, es decir para una entrada determinada la salida sea siempre la misma.

Existe otro tipo de circuito que son los llamados lazo cerrado:



Representación sistema lazo cerrado

Estos circuitos se caracterizan porque la salida no depende solo de la entrada sino que depende tanto de la entrada como de la salida. Estos circuitos son ideales para la regulación por el hecho de estar comparando la salida con la entrada. hay que añadir que la línea de arriba no solo estará la planta sino que puede haber muchas otras funciones como controladores etc., al igual que la línea de abajo que puede tener muchas funciones, pero a partir de aquí vamos a considerar la línea de arriba como G(S) y la de abajo como H(s).

Llegado a este punto vamos a definir la función de transferencia que no es más que la función matemática que relaciona el cociente entre la salida y la entrada:

Función de transferencia: F(s)=Y(s)/X(s)

La función de transferencia constara de un numerador donde se encuentran los ceros del sistema (son los valores que hacen cero al numerador), y un denominador donde se encuentran los polos del sistema (son los valores que hacen cero al denominador).

La manera de clasificar a los diferentes sistemas es a través del grado del polinomio que tenga la ecuación característica (denominador de la función de transferencia), con este criterio obtenemos tres grupos: los sistemas de primer orden (polinomio de grado 1), sistemas de segundo orden (polinomio de grado 2) y sistemas de orden superior (polinomio de grado 3 o superior).

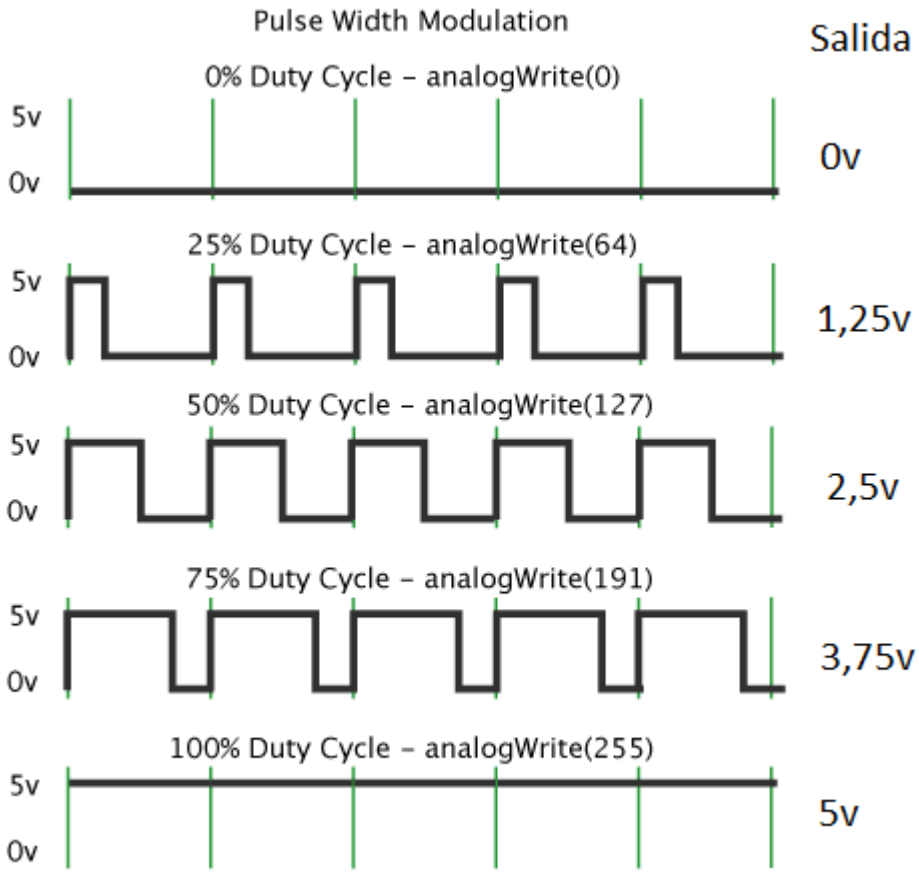
2.3. PWM

La modulación por ancho de pulsos (PWM) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el tiempo de encendido (llamado ancho de pulso) en relación con el periodo.

El PWM tiene varias aplicaciones pero a nosotros la que nos interesa es la de convertidor ADC, lo que nos permite 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 su ciclo de trabajo.

A continuación se muestra una imagen del funcionamiento del pwm en arduino.



Ejemplo de uso PWM

En la imagen se puede ver como el aumento del ciclo de trabajo es proporcional al aumento de la salida

3. DESCRIPCIÓN DE HARDWARE Y SOFTWARE

3.1. Arduino

3.1.1. Introducción

Arduino es una placa electrónica que contiene un microcontrolador y su objetivo se basa en pequeños proyectos, para aficionados y amantes de la electrónica en general. Su principal característica es la facilidad con la que se programa, a diferencia de las demás placas con microcontroladores del mercado que su programación es mas laboriosa, además arduino es una empresa basada en software y hardware libre con la ventaja que se puede utilizar en cualquier ambiente y proyecto.

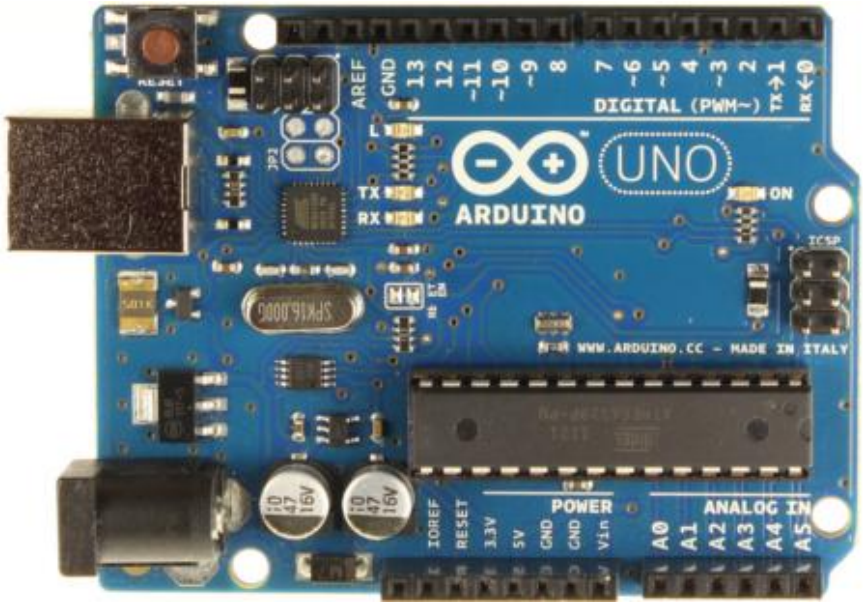
Las placas de arduino se pueden utilizar de diferentes maneras, ya sean alimentadas a través de USB por medio del ordenador o con una pequeña batería sin necesidad de conectarse con el ordenador. Arduino se programa a través de un programa gratis que se descarga a través de la página web de arduino, y a través de este se transfiere el programa que se escriba desde el ordenador a la placa, estos programas utilizan un lenguaje de programación propio de Arduino basado en Wiring. Pero también se puede utilizar arduino con otros programas, como por ejemplo, simulink de Matlab(tiene librerías para utilizar arduino), pero siempre cargando un programa a la placa que interacciona correctamente con simulink, suelen ser programas básicos que vienen con la librería de Matlab, o con la librería del programa que quieres utilizar.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede interaccionar con aquello que le rodea controlando luces, motores y otros actuadores.

Arduino dispone de diferentes placas dependiendo de la necesidad que tenga el proyecto.

3.1.2. Arduino uno

Arduino uno es una placa que contiene 14 pines de entradas/salidas digitales (de las cuales 6 se pueden utilizar como salidas PWM), 6 pines de entradas analógicas, un botón de reset, una conexión para USB, una conexión para alimentar el arduino de forma externa, todo ello basado en el microcontrolador ATmega328.



Placa Arduino UNO

3.1.2.1. Principales características de arduino uno.

Microcontrolador	ATmega328
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite inferior/superior)	6-20V
Pines digitales Entrada/Salida	14 (de los cuales 6 proporcionan salida PWM)
Pines de entrada analógica	6
Corriente para pines de entrada/salida	40 mA
Corriente para Pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) de los cuales 0,5 KB utilizado por el gestor de arranque
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad del reloj	16 MHz

Alimentación

El Arduino uno se puede alimentar de dos formas a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.
La fuente externa (no USB) puede venir a través de un convertidor CA/CC o con una batería. El convertidor se conecta a través de la clavija Jack de alimentación de la placa. Si dispones de la batería esta se alimenta a través de los pines Gnd y Vin.

Los pines de alimentación son los siguientes:

- **VIN:** Es la tensión de entrada a la placa Arduino cuando se utiliza una fuente de alimentación externa.
- **5V:** Este pin da una salida de 5V regulada por el regulador de la placa.
- **3.3V:** Este pin da una salida de 3.3 voltios regulados por la placa, con una corriente máxima de 50 mA.
- **GND:** pin de tierra.

IOREF: Este pin de la placa Arduino proporciona la tensión de referencia con el que el microcontrolador opera. Un escudo correctamente configurado puede leer el voltaje del pin IOREF y seleccionar la fuente de alimentación adecuada.

Entrada y salida

Cada uno de los 14 pines digitales de la placa de Arduino uno se puede usar como una entrada o salida, utilizando las funciones pinMode_() , digitalWrite_() , y digitalRead_() . Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50 k. Además, algunos pines tienen funciones especializadas:

- **Serie 0 (RX) y 1 (TX):** Se utiliza para recibir (RX) y transmitir (TX) datos serie TTL. Estos pines están conectados a los pines correspondientes del ATmega8U2 USB-to-Serial TTL chips.
- **Las interrupciones externas (2 y 3):** Estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.
- **PWM(3, 5, 6, 9, 10, y 11):** Proporcionar 8-bit de salida PWM con la función analogWrite()

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** Estos pines son de apoyo para la comunicación SPI.
- **TWI: A4 o A5 y SDA pin o pines SCL.** Apoyo para la comunicaciónTWI.
- **AREF.** Tensión de referencia para las entradas analógicas. Se utiliza con la función analogReference ().
- **Reset.** Pon este pin a cero para reiniciar el microcontrolador. Normalmente se utiliza para agregar un botón de reinicio a los escudos que bloquean la placa.

El Arduino uno tiene 6 entradas analógicas, con la etiqueta A0 a A5, cada una de los cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se mide desde cero a 5 voltios.

3.2. 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 que interpreta 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.

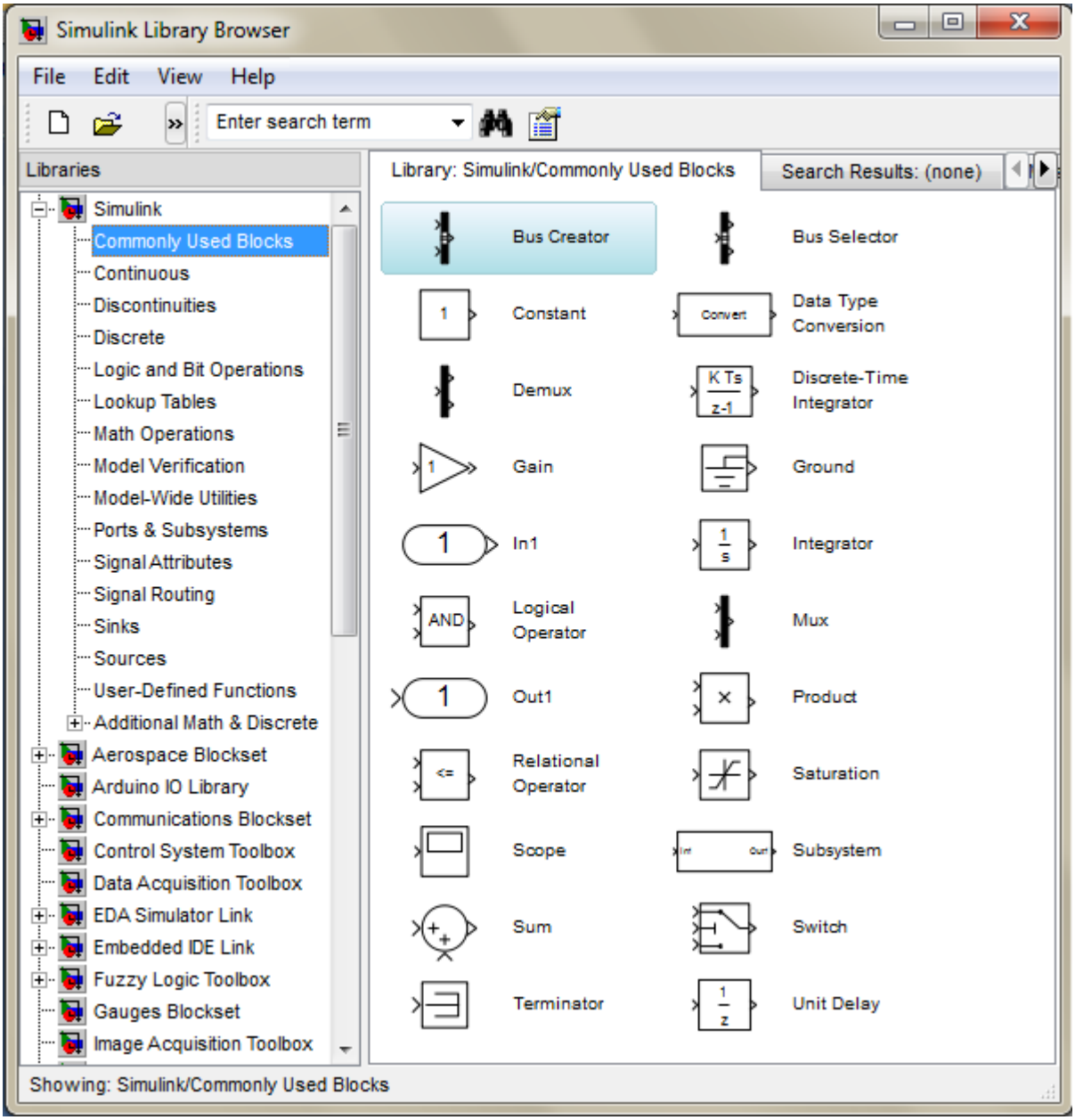
Admite el diseño y la simulación a nivel de sistema, la generación automática de código y la prueba y verificación continua de los sistemas embebidos.

Mediante simulink se puede construir y simular modelos de sistemas físicos y sistemas de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo.

Tiene una conexión directa con Matlab, pudiendo exportar los resultados obtenidos en simulink para hacer análisis más exhaustivos y poder obtener nuevos resultados. También algunos bloques nos dan la posibilidad de incorporar algoritmos propios de Matlab.

Simulink ofrece la posibilidad de conectar el modelo con hardware para comprobar en tiempo real y de una manera física el funcionamiento de este.

En *Simulink* es posible crear y simular modelos mecánicos, eléctricos, electrónicos, aeronáuticos, etc. gracias a la gran variedad de bloques (*blocksets*) de los que dispone. Estos conjuntos de bloques se encuentran agrupados en la *Simulink library browser*, que se despliega al ejecutar *Simulink*, y presenta el aspecto de la figura.



Simulink Library browser

La librería principal de bloques se encuentra bajo la carpeta llamada *Simulink* y en ella aparecen los bloques agrupados en las siguientes categorías: continuos, no lineales (*Discontinuities*), discretos, tablas, operaciones matemáticas, verificación de modelos, puertos y subsistemas, señales, dispositivos de salida (*Sinks*), generadores (*Sources*).

A continuación daremos una pequeña explicación de los bloques que utilizamos en nuestro proyecto.

Constant: Genera una señal cuyo valor no cambia en el tiempo. Para especificar el valor de este bloque, sólo basta configurar constant value, en la ventana de configuración de

parámetros.

Scope. Si la señal de entrada es u (t), entonces muestra un gráfico de u en función de t.

Slider Gain. Sirve para poder modificar durante la simulación la ganancia. Dentro de el configuraremos sus valores low y high.

Sum: Este bloque realiza suma o restas de sus entradas. Estas pueden ser un escalar, un vector o una matriz. Las entradas tienen que ser todas el mismo tipo de dato.

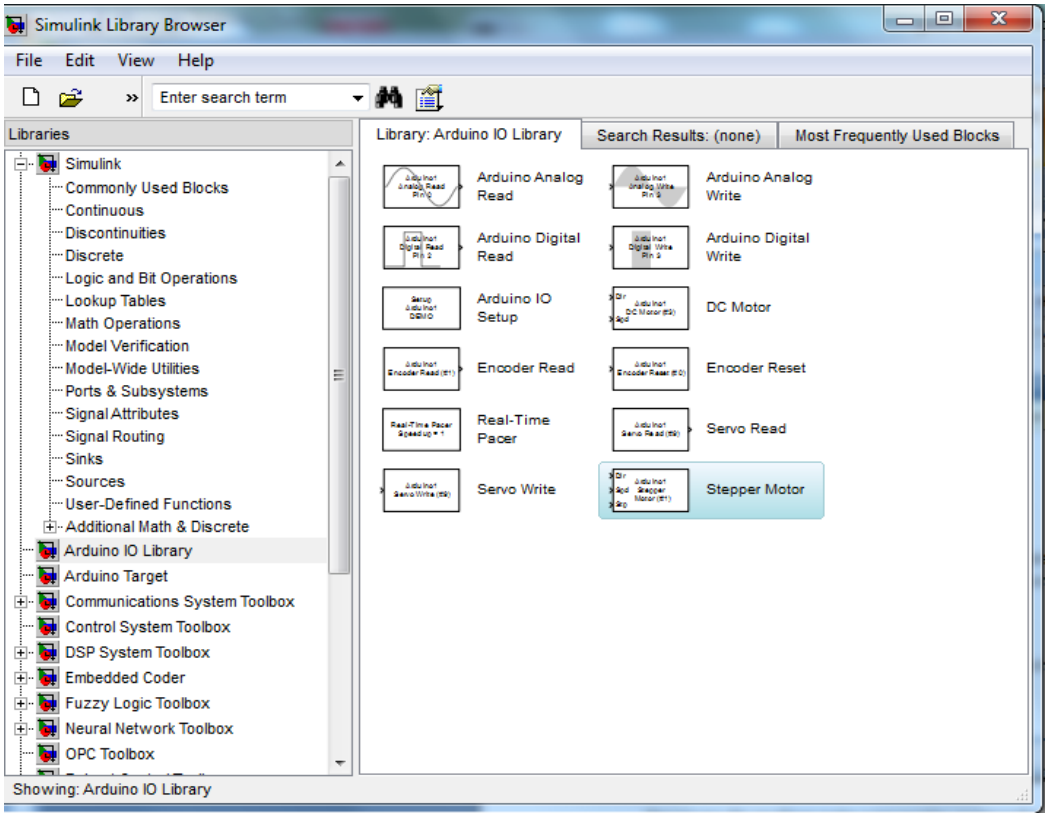
Mux: Combina sus entradas en unico vector de salida. Las entradas pueden ser una señal escalar o vector. Todas ellas tendran que ser el mismo tipo de dato. En nuestro caso lo utilizamos para que en el scope nos muestre dos gráficas a la vez.

Product: Nos da como resultado la multiplicación de sus dos entradas: dos escalares, un escalar y un no escalar o dos escalares de las mismas dimensiones.

La biblioteca de diagramas de bloques es manipulable y se podrá ampliar según sean nuestras necesidades. Estas bibliotecas se podrán bajar directamente desde Mathworks para instalarlas en simulink. En nuestro caso la biblioteca que usaremos será ArduinoIO.

3.2.1. ArduinoIO

Hay varias bibliotecas que nos permiten trabajar con arduino y simulink, nosotros hemos elegido esta ya que es gratuita y su uso es relativamente sencillo. Con esta biblioteca puedes establecer conexión con todos los tipos de arduino. Está formada por los bloques que vemos en la siguiente figura.



Arduino IO Library

A continuación explicamos brevemente los bloques utilizados en nuestro proyecto:

Setup: Este bloque es necesario para establecer la conexión con la placa a través del puerto USB. Dentro de él tendremos que configurarlo poniendo la placa de arduino que estemos usando y el puerto a la que está conectada. Tiene también la opción de trabajar en simulación en el caso de que no dispongamos de placa.

Analog Read: Este bloque lo utilizaremos para leer la entrada desde un pin de la tarjeta arduino. Los pines desde los que puede leer son los analog in (A1, A2, A3, A4, A5). Estos pines leen las entradas analógicas y lo convierten en una cifra digital de 10 bits, es decir funcionan como un conversor A/D. Los valores de entrada irán de 0 a 5V que se corresponderán a las cifras 0 y 1023 que nos ofrecerá nuestro bloque. Dentro de este bloque se configurara el puerto que se quiere leer.

Analog Write: Este bloque lo utilizaremos para escribir desde Matlab en los pines de Arduino PWM (3, 5, 6, 9, 10, 11). Estas salidas adquieren diferentes valores gracias a la técnica de modulación de ancho de pulso que consiste en la conmutación entre encendidos y apagados para simular una salida analógica a través de una digital. El tiempo de encendido respecto al total (encendido + apagado) será proporcional a la tensión de salida siendo este máximo cuando todo el tiempo este encendido. El máximo valor que puede entregar será de 5V, por lo que la tensión variara desde estos 5V hasta 0V. El

valor que se le dará desde simulink a este bloque será una cifra de 0 a 256 ya que requiere una cifra digital de 8 bits.

Digital Read: Su uso será el mismo que el bloque de analog read pero esta vez la lectura será de forma digital (valores 0 y 1).

Digital Write: Al igual que el analog write nos sirve para escribir, esta vez de forma digital. Los pines que se usan para esto son 2, 4, 7, 8, 12, 13. Este pin se seleccionara dentro del bloque.

3.3. Circuito eléctrico

En este apartado del proyecto vamos a explicar de una forma detallada las partes de este que tengan que ver con un circuito electrónico. Dentro de este "circuito electrónico "podemos distinguir tres apartados diferenciados entre sí: el primero tienen que ver con la alimentación del propio circuito, el segundo se centra en el apartado de control que se basa en un microcontrolador y el tercer se basa en un actuador (motor mas hélice) que es el resultado final del proyecto.

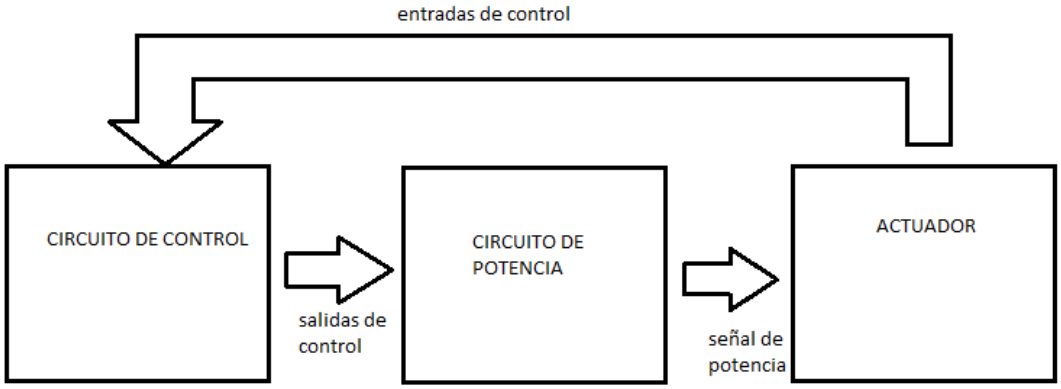
Estos tres apartados no están separados físicamente entre sí, sino que forman parte del mismo todo, pues para que el motor funcione se le tiene que dar la alimentación adecuada en la etapa de potencia y esta está controlada por el circuito de control.

El circuito de control se hace a través del microcontrolador de arduino, pero de una manera interna a través de programas, en este caso a través de simulink con lo que para referirnos al circuito eléctrico el arduino solo haremos referencia a unas señales que entran al arduino (entradas) y unas señales que salen del arduino (salidas).

El circuito de alimentación su objetivo es la alimentación del motor de corriente continua porque la placa de arduino está alimentada a través de USB con el ordenador. Se necesita una fuente externa ya que a la placa de arduino se alimenta a través del USB con una tensión de 5v y nuestro motor necesita un potencial más elevado con lo que es necesario una fuente externa.

La tercera parte del circuito es el actuador, en este caso el motor más la hélice, esta parte no tiene que ver con un circuito eléctrico en sí, sino que depende del circuito de potencia y del de control.

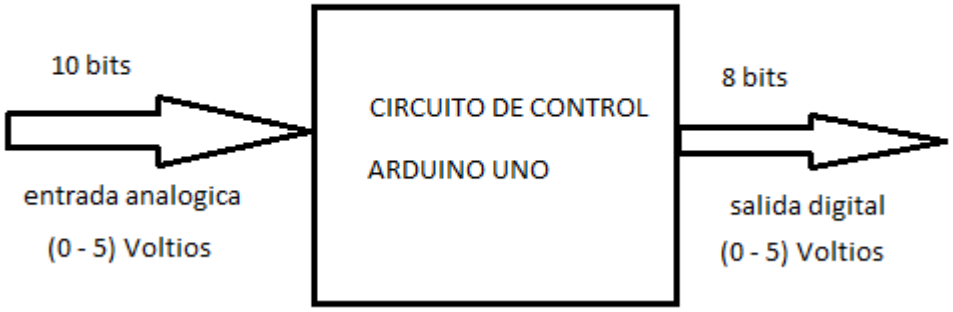
En conclusión tenemos un circuito de control que controla gracias a su programación interna y a las señales de entrada y salida del circuito. Este circuito de control gobierna sobre el circuito de potencia que permite que el actuador actué de una forma u otra.



Representación en bloques del circuito eléctrico

En este esquema se ve de una forma clara lo que se quiere implementar, un circuito de control que envía una señal al circuito de potencia, este recibe esta señal y envía otra señal al actuador que como su nombre indica actúa devolviendo en este caso una señal que será la que llegue al circuito de control.

En nuestro proyecto el circuito de control es una placa de arduino, en concreto arduino UNO, esta placa de arduino posee entradas y salidas de tipo analógico y de tipo digital, en concreto nuestra salida será de tipo digital que para nuestra placa es una señal de 8 bits, como arduino se alimenta de una señal de 5 voltios(a través de USB) esta señal variara de 0 a 5 voltios. La entrada de control que recibe el circuito en este caso la placa de arduino, va a estar conectada a un pin de una entrada analógica y están poseen 10 bits de precisión para el convertidor interno, es decir la placa de arduino de nuestro proyecto recibe una señal de 10 bits de precisión y se capaz de enviar una señal de 8 bits de precisión siempre siendo una señal de 0 a 5 voltios, hay que tener en cuenta que el almacenamiento de las entradas y el control de las salidas se hará vía software programando la placa de arduino o como en este caso programando la placa de arduino para que sea capaz de responder a un programa hecho a través de simulink (MATLAB).

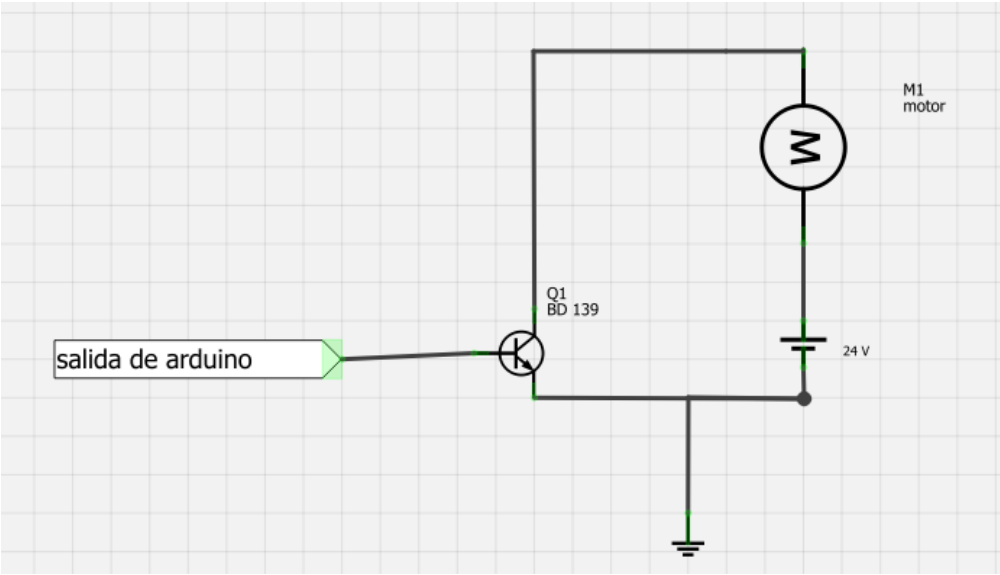


Circuito de control

CIRCUITO DE POTENCIA

Debido a que como actuador usamos un motor de corriente continua cuyo potencial de funcionamiento es mayor que los 5 voltios (máximo) que proporciona la placa de arduino necesitamos una fuente externa que permita al motor funcionar en el rango de potencial necesario. Hemos elegido una fuente de alimentación de 24 voltios ya que es el máximo potencial que admite el motor, esta fuente la conectamos a la red (220 voltios) y la salida serán siempre 24 voltios.

Como la salida de nuestra fuente de alimentación es constante necesitamos un circuito de potencia que nos permite variar la cantidad de voltaje que enviamos al motor, para ello nos ayudamos de un transistor bipolar, y entre todo el catalogo hemos elegido el más económico que satisfaga nuestras necesidades, están son dos: la tensión colector-emisor debe ser mayor de 24 voltios (es la tensión máxima que va a pasar debido a la fuente de alimentación) y que la corriente de base deba soportar más de 40 mA que es la corriente que sale del arduino. Con esos dos datos hemos elegido el transistor BD 139 que posee una tensión colector emisor máxima de 100 voltios (suficiente nos valdría con que fuera superior a 24 voltios) y una corriente de base de 0,5 Amperios (también nos vale ya que es superior a los 40 mA requeridos). Entonces el circuito de potencia nos quedaría algo parecido a esto:



Circuito de potencia

La función que tiene el transistor es la de regular la tensión que pasa por el motor

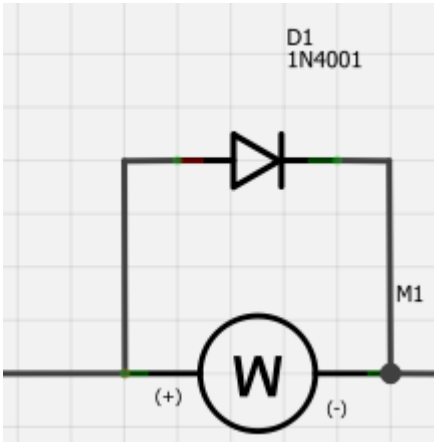
ACTUADOR

El resultado final del circuito es suministrar potencia a un motor que será nuestro actuador. Para ello hemos diseñado un circuito de control y un circuito de potencia. Al motor le va a llegar un potencial determinado, cuando más alto sea este con más celeridad girara el motor y por lo tanto la hélice

colocada en el extremo del motor ejercerá más fuerza para levantar la barra. Hay que recordar que el potencial máximo que dejamos pasar por los extremos del motor es de 24 voltios que coinciden con la tensión de nuestra fuente de alimentación.

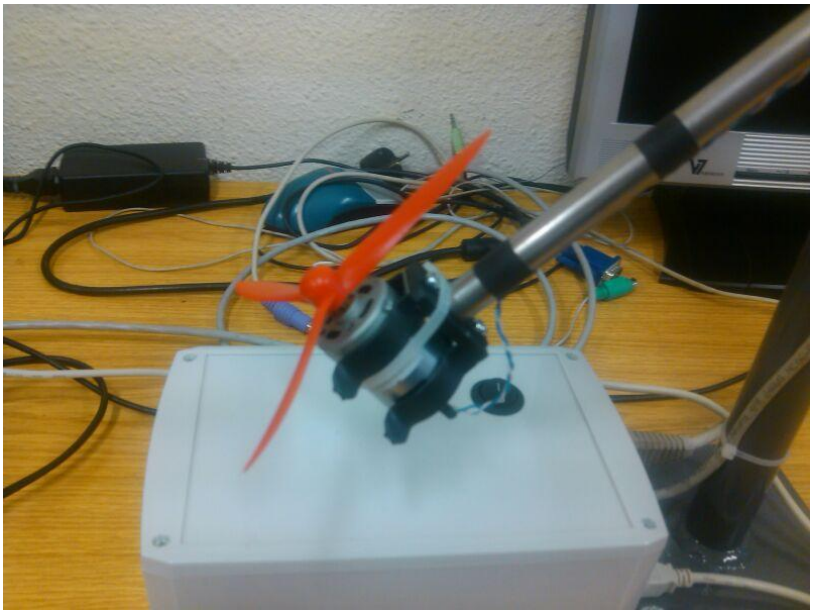
La tensión que llega al motor está controlada de la forma PWM que esta explicada con detalle en otro apartado de este proyecto.

A la hora de montar el motor físicamente debemos colocar un diodo en antiparalelo, este es necesario ya que si no lo hacemos correremos el riesgo de estropear el transistor que tenemos en el circuito de potencia. Esto se debe a que a la corriente de fugas que produce el motor a la hora de desconectarse, es decir, el motor funciona de manera correcta y el diodo no tiene ninguna influencia pero cuando se deja de suministrar voltaje al motor crea una corriente de fugas con la energía que había almacenado, esta corriente tiene que salir por algún lado entonces al poner el diodo en antiparalelo esta corriente circula por él asimilando la potencia de esta corriente, sino hubiéramos puesto un diodo en antiparalelo la corriente de fugas pasaría por el transistor estropeándole y por lo tanto el circuito dejaría de funcionar correctamente.



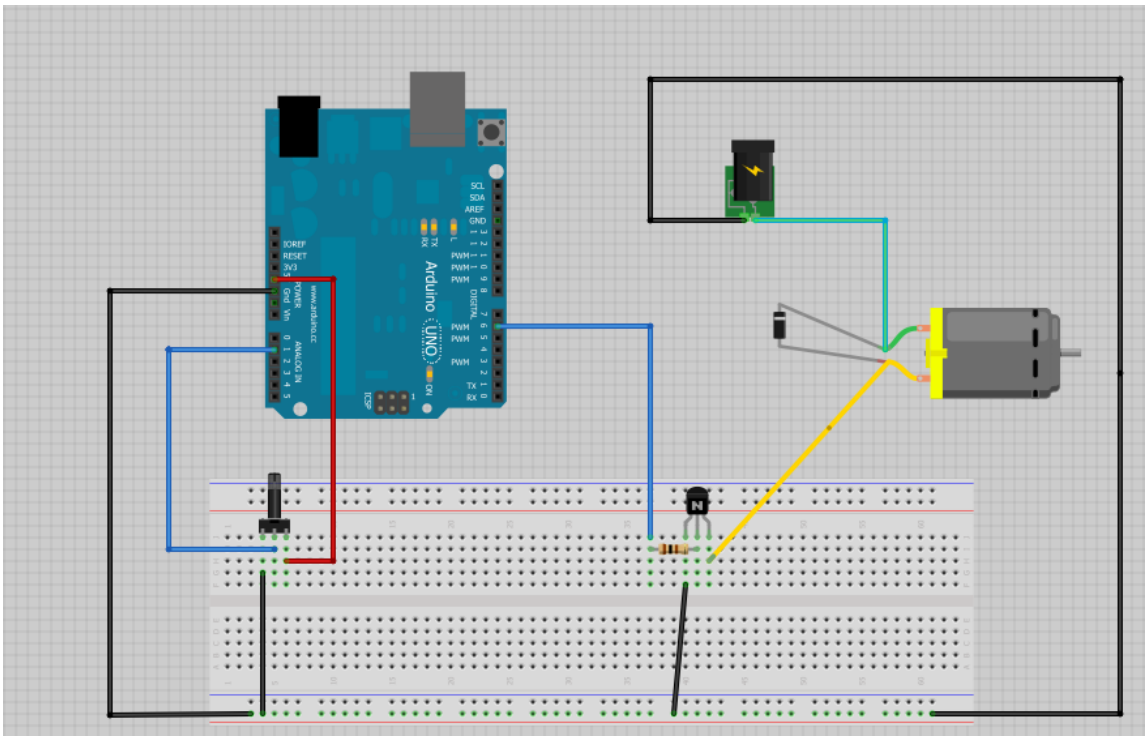
Motor con diodo en antiparalelo

Como se observa en la imagen el ánodo del diodo coincide con el positivo del motor (el motor no es que tenga un lado positivo y uno negativo pero al alimentarlo de esta manera el motor gira en el sentido que nosotros queremos) y el cátodo está conectado al lado negativo.



Fotografía del motor

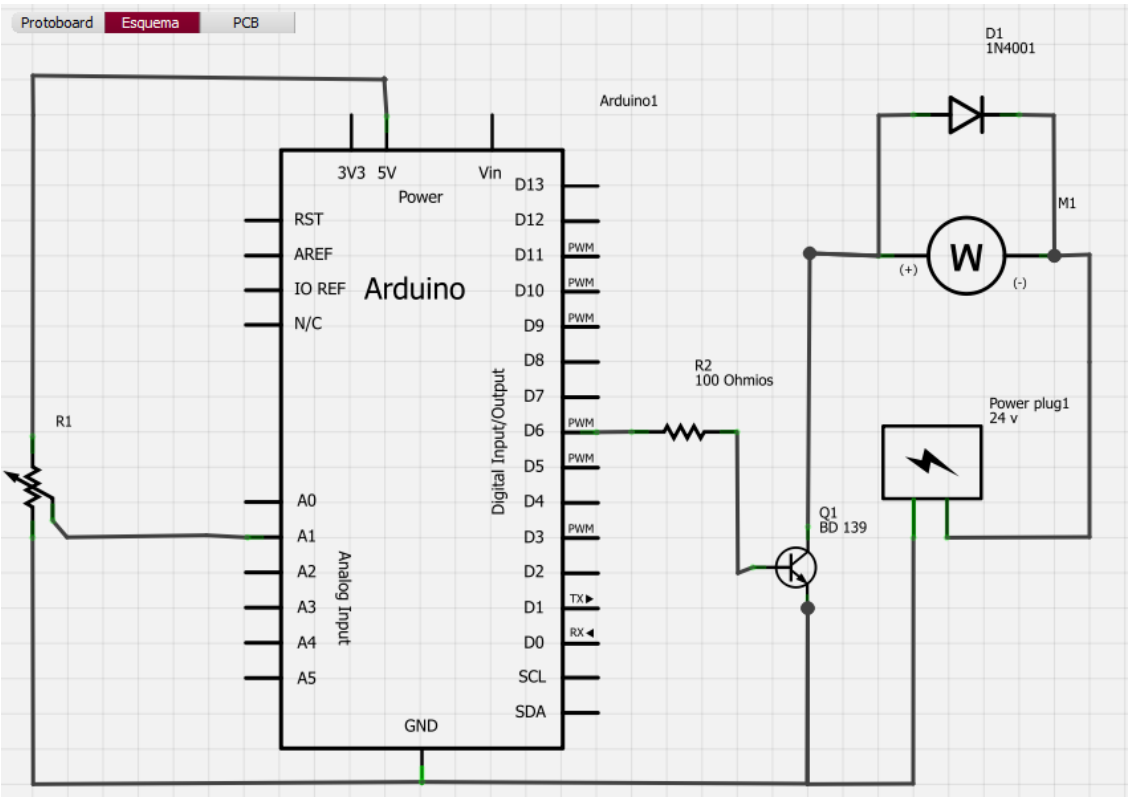
En la intersección de la barra vertical con la barra móvil, hemos colocado un potenciómetro solidario al giro de la barra móvil para poder registrar el movimiento. Este potenciómetro está alimentado en sus patillas extremas con los pines de la paca de arduino de +5V y GND y la patilla central con una entrada analógica (en concreto la A1). Con ello creamos un divisor de tensión que nos permite registrar la posición de la barra móvil ya que al mover esta la resistencia del potenciómetro varia y por la tanto la tensión que registra el arduino también.



Interfaz física

Este es el montaje físico de nuestro esquema eléctrico, como se puede observar en él están todos los elementos que hemos explicado con anterioridad y las conexiones que tenemos que hacer en la placa de arduino. Este esquema es solo ilustrativo ya que como se ha señalado con anterioridad el potenciómetro va colocado en la intersección de las dos barras y no en una protoboard, además se ha añadido un radiador al transistor para que disipe mejor el calor.

Este circuito hay que alimentarlo de dos formas, hay que enchufar la fuente de alimentación a una toma de red de 220 voltios y la placa de arduino (a través de un cable de USB) al ordenador para poder controlar el circuito vía Matlab y a la vez alimentar la placa de arduino.



Esquema eléctrico

4. MODELO MATEMÁTICO

Para establecer el modelo matemático de nuestro sistema tendremos que tener un completo conocimiento sobre él y las leyes físicas que le afectan. A grandes rasgos nuestro sistemas se puede definir como 2 barras ancladas, una de ellas totalmente fija y otra con un grado de libertad que realiza un movimiento rotacional. El centro de rotación es el punto donde se anclan ambas barras.

A continuación realizaremos el estudio de nuestro sistema para conseguir el modelo matemático.

4.1. Estudio del movimiento

El movimiento que realiza la barra móvil es el de rotación de un cuerpo rígido alrededor de un eje fijo. En estos movimiento lo que acelera al cuerpo angularmente es el torque y no solo la fuerza que se le aplica. Ya que en estos movimientos una fuerza aplicada sobre el sistema puede que no produzca cambio en el movimiento.

El torque relaciona la fuerza ejercida, con su distancia y dirección al eje de giro. Su fórmula es:

$$\tau = r \times F$$

"r" es el vector de posición de la fuerza aplicada que va desde el eje de giro hasta el punto de aplicación de esta. Se puede ver que fuerzas aplicadas en el eje de giro no tendrán torque por lo tanto no producirán cambio en el sistema.

F es la fuerza aplica. Como se ve es un producto vectorial por lo tanto dependerá del ángulo de aplicación de la fuerza.

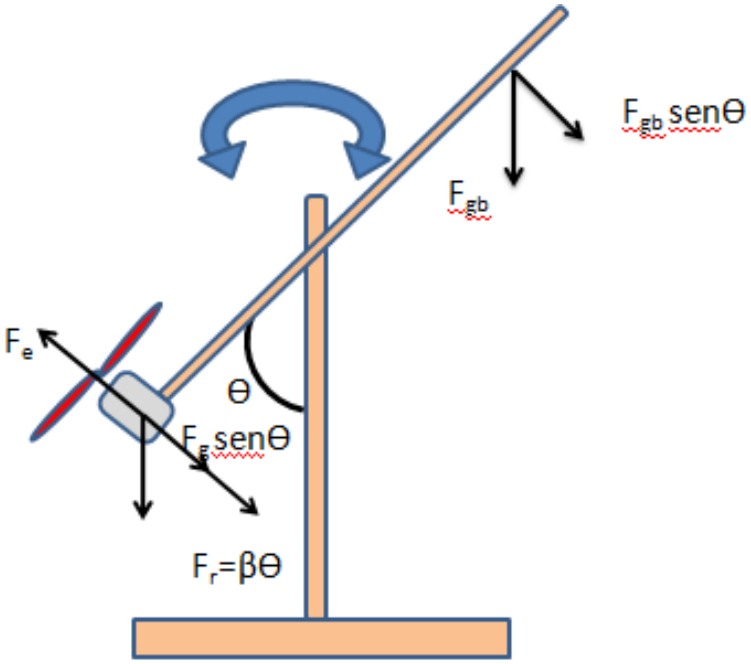
El torque está relacionado con el momento de inercia, otro término importante en los movimientos rotacionales ya que todo elemento que gira en torno a un eje posee momento de inercia. La fórmula que la relación es:

$$\sum \tau = I \propto$$

Donde $\propto$ es la aceleración angular del cuerpo.

A partir de estas dos fórmulas podemos obtener las ecuaciones de nuestro sistema. Primero calcularemos el momento de inercia de la barra y del motor y los torques de las fuerzas.

En la siguiente figura vemos un resumen de las fuerzas a las que se ve expuesto nuestro sistema.



Fuerzas del sistema

Fe es la fuerza que proporciona el giro del motor y a la encargada de hacer cambiar de posición a la barra.

Las otras fuerzas son debidas a la gravedad, F_r es la fuerza total gravitatoria de la masa del motor, F_{gb} es la debida a la barra, que se representa en la parte derecha por ser en este lado más larga.

4.2. Torque

4.2.1. Torque fuerza de la gravedad

La fuerza de la gravedad actúa tanto sobre la barra como sobre el motor y hélice por lo que lo calcularemos por separado.

Si el punto de gravedad de la barra estuviera en su eje de gravedad el torque total que produce la gravedad sobre la barra sería nulo ya que se compensarían los torques de un lado del eje de giro con el otro. En nuestro caso el centro de gravedad estar desplazado 3 cm por lo tanto la acción de la gravedad si producirá torque que será:

$$\sum \tau_b = \int_{0.22}^{0.28} g\frac{M}{L}xdx = \int_{0.22}^{0.28} 9.8\frac{0.454}{0.5}xsen\theta(t)dx = 0.1335sen\theta(t)$$

Para calcular el torque del motor y de la hélice, suponemos todo el conjunto como una masa puntual que se encuentra en el punto medio del motor. Por lo tanto el torque de este conjunto será:

$$\tau_m = r \times F_g = -0.245 \cdot 9.8 \cdot 0.076 \cdot sen\theta(t) = -0.1825\, sen\theta(t)$$

4.2.2. Torque fuerza de empuje

El motor y la hélice estarán orientados de tal forma que la fuerza que ejercen formara un ángulo de 90º con el vector de posición. Su torque será:

$$\tau_e = r \times F_g = 0.245 \cdot F_g \cdot sen90º = 0.245 F_g$$

4.3. Momento de inercia

Se define como la resistencia de un cuerpo a obtener una aceleración angular. Más concretamente el momento de inercia es una magnitud escalar que refleja la distribución de masas de un cuerpo o un sistema de partículas en rotación, respecto al eje de giro. El momento de inercia sólo depende de la geometría del cuerpo y de la posición del eje de giro; pero no depende de las fuerzas que intervienen en el movimiento.

Físicamente dado un sistema de partículas y un eje arbitrario, el momento de inercia del mismo se define como la suma de los productos de las masas de las partículas por el cuadrado de la distancia *r* de cada partícula a dicho eje. Matemáticamente se expresa como:

$$I = \sum m_i \cdot r_i^2$$

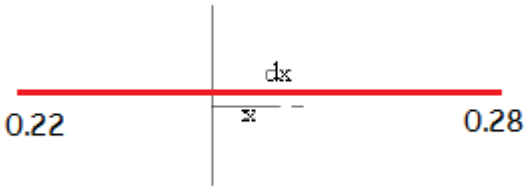
Para un cuerpo de masa continua se generaliza como:

$$I = \int r^2 dm$$

A continuación calcularemos el momento de inercia de la barra y del motor con la hélice.

4.3.1. Momento de inercia de la barra.

Aplicando la fórmula calculamos el momento de inercia de nuestra barra que tiene una masa M=454g y una longitud L=50cm. En la siguiente figura mostramos un esquema de nuestra barra.



Distribución de la barra

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

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

Por lo tanto el momento de inercia nos queda:

$$I_b = \int_{-0.22}^{0.28} \frac{0.454}{0.5} x^2 dx = 9.867 \cdot 10^{-3}\ kgm^2$$

4.3.2. Momento de inercia del motor y hélice

Al calcular el momento de inercia del motor hélice lo tomaremos como si toda su masa estuviera concentrada en un punto. Para calcularlo utilizaremos la fórmula de la masa puntual:

$$I = \sum m_i \cdot r_i^2$$

El conjunto tiene una masa m=0.076Kg y se encuentra a una distancia r=0.245m, por lo tanto su momento de inercia será:

$$I = 0.076 * 0.245^2 = 4.562 \cdot 10^{-3}\ kgm^2$$

4.3.3. Momento de inercia total

Sera la suma de los dos elementos anteriores:

$$I_T = 1.443 \cdot 10^{-2}\ kgm^2$$

4.4. Obtención de la función de transferencia

En nuestra función de transferencia como es obvio la entrada será la fuerza de empuje y la salida será el ángulo que forma la barra móvil con la vertical. Por lo tanto la función de transferencia quedara de la forma $\frac{\theta}{F_e}$.

La ecuación de nuestro sistema la obtendremos con la ayuda de la fórmula. En el supuesto de que funcionara de forma ideal sería correcta, pero como todo sistema posee rozamiento por lo tanto le tendremos que unir este término. Por lo tanto la fórmula quedara:

$$\sum \tau\ - B\dot{\theta} = I \propto$$

Como ya tenemos los torques del sistema calculados y los momentos de inercia solo tendremos que sustituir. $\propto$ Es la aceleración angular por lo tanto se puede poner como $\frac{d^2\theta}{dt}$.

$$\tau_{b+}\tau_{m+}\tau_e - B\frac{d\theta}{dt} = I\frac{d^2\theta}{dt}$$

$$0.1335sen\theta(t) - 0.1825sen\theta(t) + 0.245F_g - B\frac{d\theta}{dt} = 0.01443\frac{d^2\theta}{dt}$$

$$0.245F_g = 0.049sen\theta(t) + B\dot{\theta} + 0.01443\ddot{\theta}$$

Al encontrar derivadas en nuestra ecuación utilizaremos la transformada de Laplace para poder obtener la función de transferencia $\frac{\theta}{F_e}$. Como la ecuación no es lineal, la linealizamos utilizando las series de Taylor en cuanto a un punto de equilibrio

($\dot{\theta}_0 = \ddot{\theta}_0 = 0, \theta = 45º, F_g$=0.049sen(45)/0.245=0.141N)

$$f = 0.245F_g - 0.049sen\theta(t) - B\dot{\theta} - 0.01443\ddot{\theta} = 0$$

$$\Delta f = \frac{\partial f}{\partial \ddot{\theta}}\Delta\ddot{\theta} + \frac{\partial f}{\partial \dot{\theta}}\Delta\dot{\theta} + \frac{\partial f}{\partial \theta}\Delta\theta + \frac{\partial f}{\partial F_g}\Delta F_g = 0$$

$$\Delta f = 0.245\Delta F_g - 0.049cos45º\Delta\theta - B\Delta\dot{\theta} - 0.01443\Delta\ddot{\theta}$$

La ecuación linealizada quedara:

$$0.245\Delta F_g = 0.01443\Delta\ddot{\theta} + B\Delta\dot{\theta} + 0.0346\Delta\theta$$

Ahora ya podemos aplicar la transformada de La place:

$$0.245\Delta F_g(s) = 0.01443s^2\Delta\theta(s) + Bs\Delta\theta(s) + 0.0346\Delta\theta(s)$$

Por lo tanto la función de transferencia de nuestro sistema suponiendo condiciones iniciales nulas y tomando ΔF=0 será:

$$\frac{\Delta\theta(s)}{\Delta F_g(s)} = \frac{0.245}{0.01443s^2 + Bs + 0.0346}$$

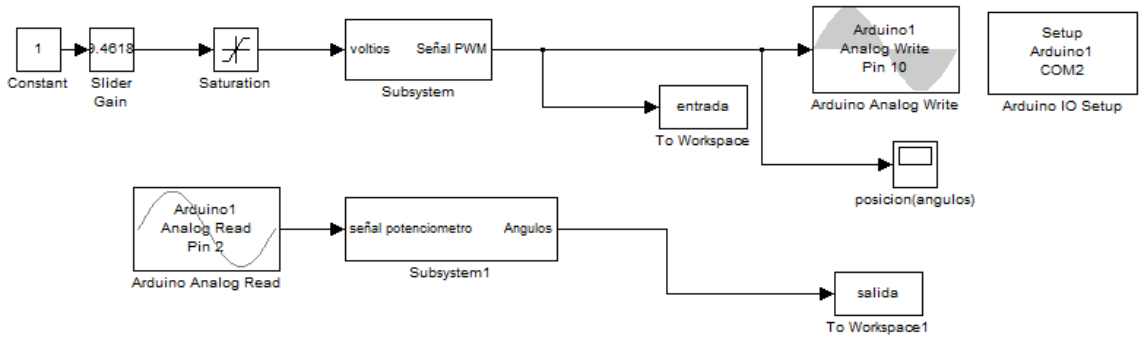
El coeficiente de rozamiento B cambiara de un dispositivo a otro, ya que el rozamiento de cada uno es distinto.

5. IMPLEMENTACIÓN

5.1. Lazo abierto

La aplicación en lazo abierto nos sirve para saber cómo responde el sistema a diferentes entradas. Al no poseer realimentación a misma entradas no dará siempre las mismas salidas ya que dependerá en otros aspectos como la posición y entrada anterior.

La aplicación en lazo abierto de simulink es la siguiente:



Esquema en lazo abierto

La entrada la daremos en voltio y la salida en ángulos. Para ello tenemos dos bloques que nos transforman las señales de 8 bits y 10 bits en voltios y ángulos respectivamente. Se encontrara una explicación más extensa de estos bloques a lo largo de la memoria.

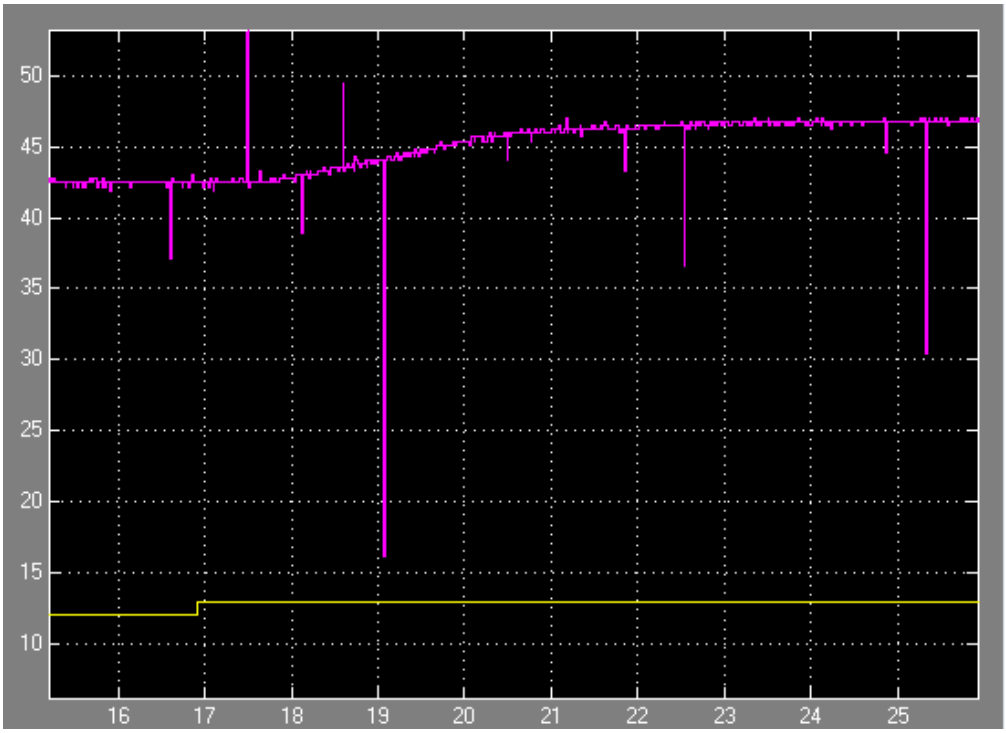
Como se verá en las imágenes la respuesta de nuestro sistema es una respuesta sobreamortiguada de un sistema de 2º orden. Esto indica que su coeficiente de amortiguación será mayor que la unidad y sus polos serán reales negativos.

Mostraremos tres imágenes donde se ve la variación de posición de la barra al variarle la entrada que esta expresada en voltios. En la primera figura la variación será de 12 a 13 voltios, la segunda de 13 a 15.5 voltios y la tercera de 15.5 a 12 voltios.

Con estas gráficas nos podemos hacer una idea del funcionamiento del dispositivo. Por ejemplo se puede ver que la posición final en la última gráfica es mayor que en el de la primera aunque la variable de entrada sea mayor en esta última. Esto es debido a que en la última, parte de una posición más elevada por lo que la fuerza del motor solo la utilizara para mantener cierta posición mientras que en la primera esta tendrá que ser utilizada para cambiar su posición teniendo que vencer además el rozamiento del sistema que se opondrá al movimiento.

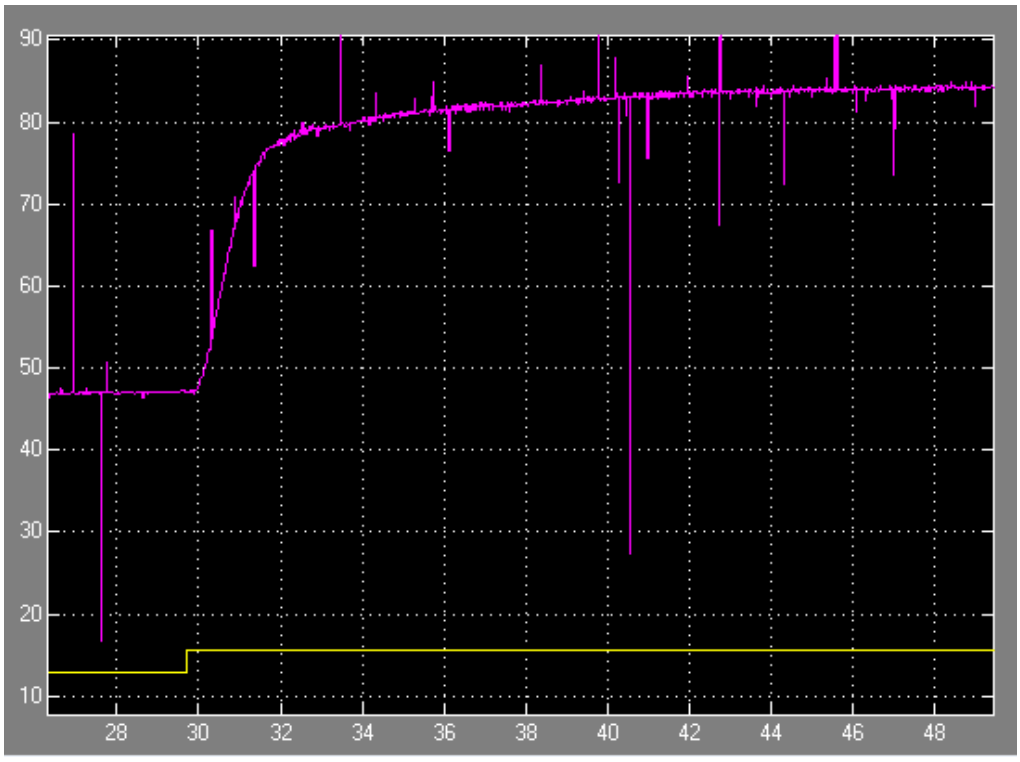
Durante las diferentes pruebas de funcionamiento hemos visto un cambio de comportamiento del sistema cuando la barra superaba los 90º. Cuando superaba esa altura el sistema mostraba menos oposición a cambiar su movimiento. Esto es debido a que la parte de la barra móvil donde se encuentra el motor supera la horizontal.

En la primera gráfica tenemos una variacion en la entrada de 12 a 13 voltios que provoca un cambio en la salida de 42,5 a 46,74 grados.



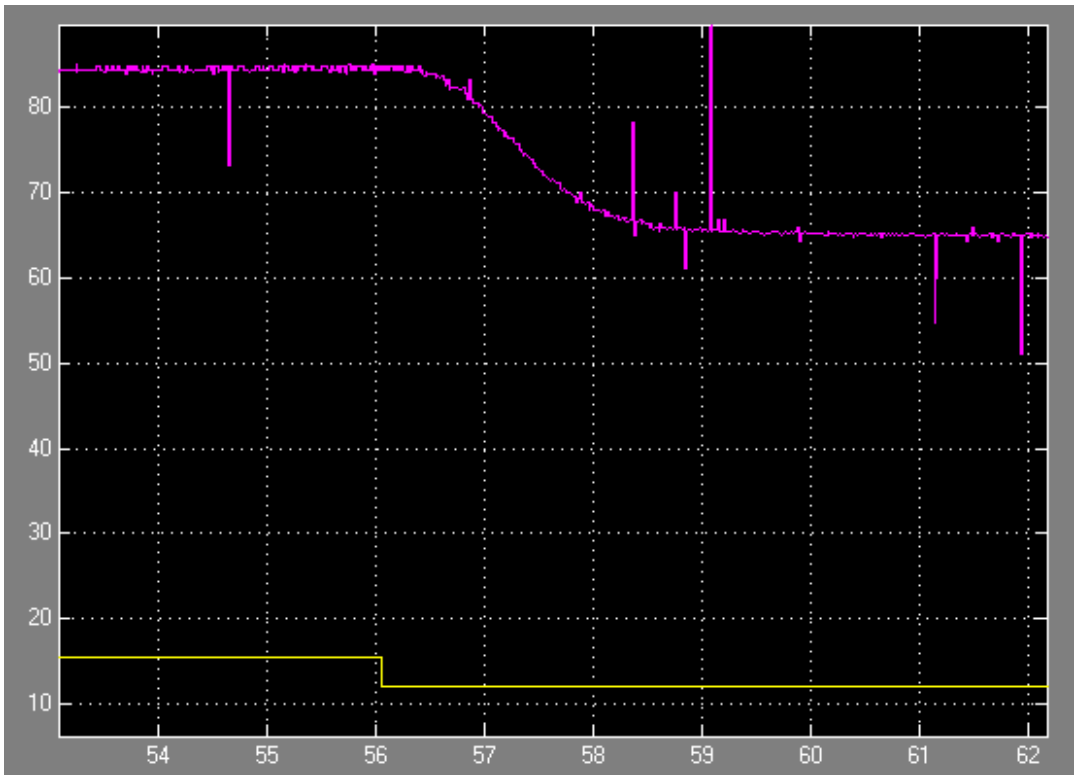
Respuesta en lazo abierto 1

En la segunda gráfica la variación de entrada de de 13 a 15,5 voltios obteniendo un cambio en la salida de 46,74 a 83,5.



Respuesta en lazo abierto 2

Por último en la tercera figura vemos una variación negativa de la entrada de 15,5 a 12 que nos proporciona un cambio negativo en la salida de 84,5 a 64,6 grados.



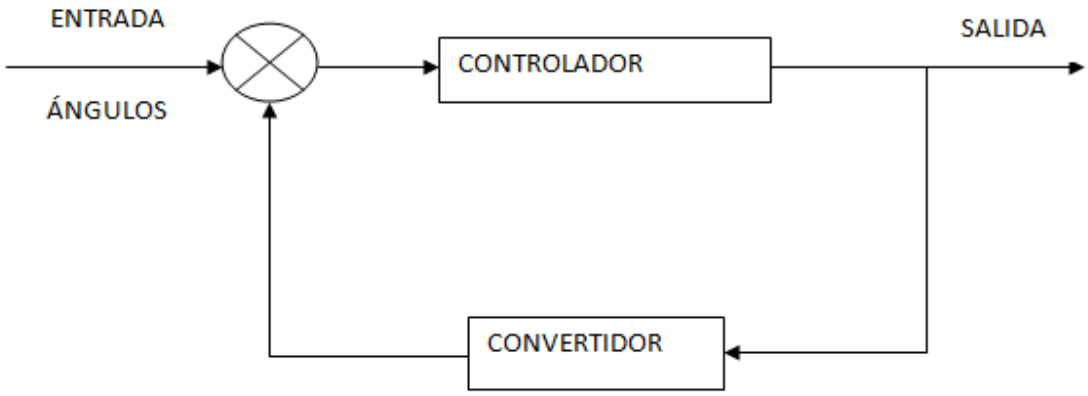
Respuesta en lazo abierto 3

5.2. Lazo cerrado

Con el control en lazo cerrado lo que se pretende es tener el control de la barra en todo momento, es decir lo que pretendemos es mandar una señal de entrada en ángulos y que la barra se ponga en esa posición tardando el menor tiempo posible y teniendo una precisión adecuada.

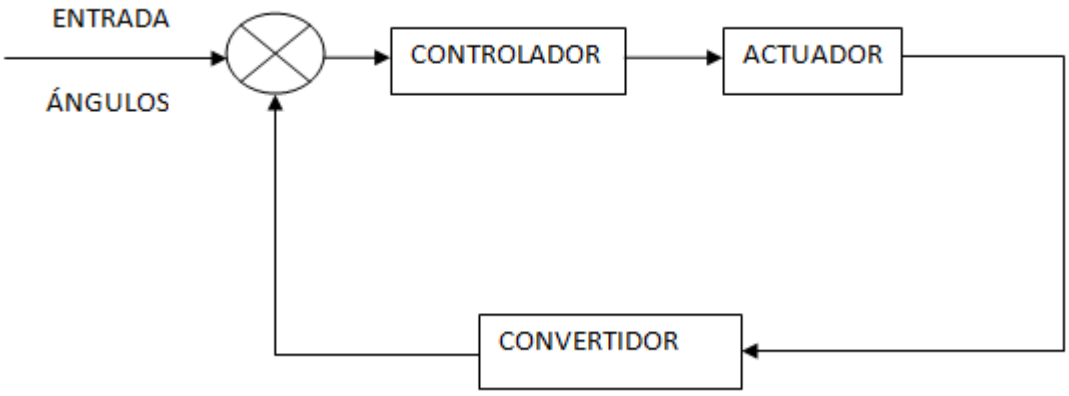
Para ello nos valdremos de la señal que envía el potenciómetro que nos dará el ángulo de la barra, hay que tener en cuenta que la señal del potenciómetro es analógica y que esta entra al arduino a través de su convertidor la transforma en una señal digital de 10bits. La señal de entrada la enviaremos en ángulos, ya que es la manera más fácil y visual de observar el resultado (recordar que en el montaje físico existe un transportador de ángulos que de una manera visual indica al usuario la posición de la barra).

Entonces lo que queremos es un montaje del siguiente tipo:



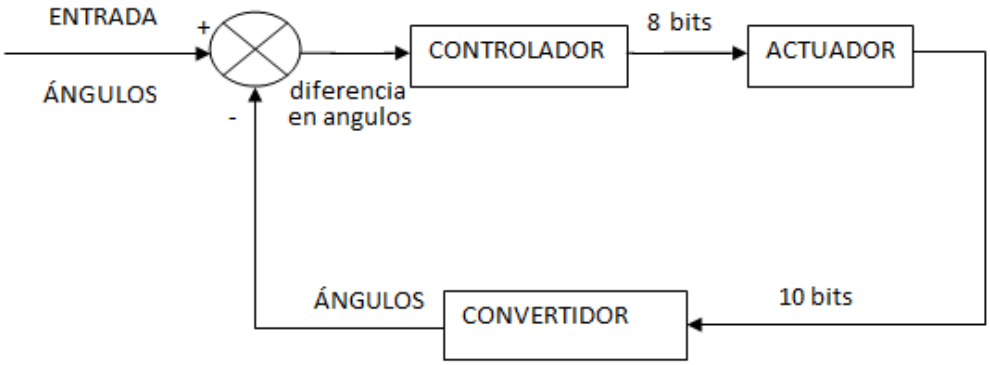
Primer esquema lazo cerrado

En este montaje metemos una entrada en ángulos la comparamos con la salida y la enviamos a un controlador y este da una salida dependiendo la entrada que le llega. Este esquema no es del todo real en nuestro circuito ya que la salida que le enviamos al arduino es una salida digital de 8 dígitos, es decir si habláramos en decimal la salida varía desde 0 hasta 255 que es su máximo, pero en cambio lo que nosotros recibimos es una señal analógica que arduino la transforma a una digital de 10 dígitos que en caso decimal varía desde 0 hasta una máximo de 1023. Esto implica que este circuito no nos sirve ya que la salida antes de ser realimentada tiene que entrar en un actuador, y medir la posición y realimentarla, un circuito más realista sería el siguiente:



Esquema cerrado real

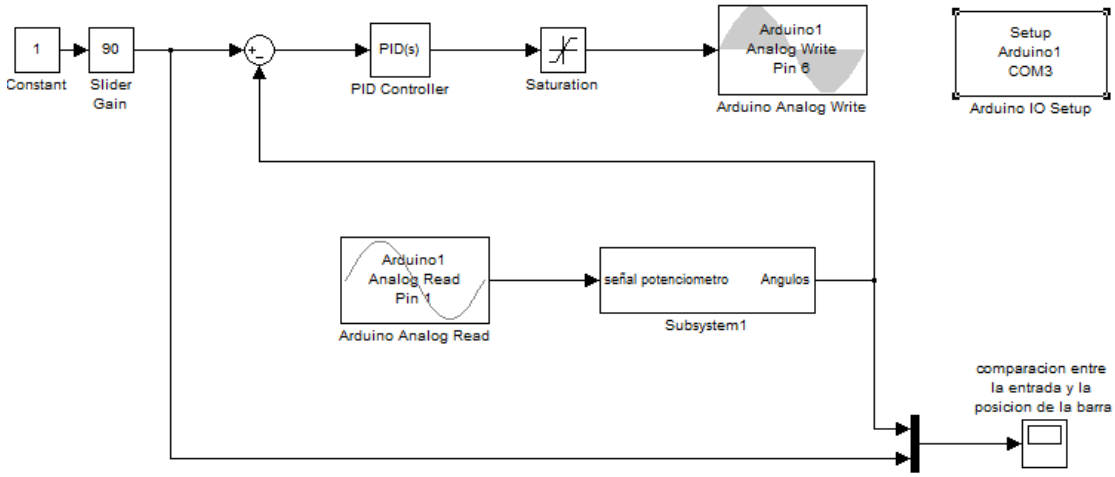
Como se ve en este esquema no existe una salida propiamente dicha ya que esa salida es la que entra al actuador y se comprueba el nivel de este actuador para transformarlo en ángulos y poder compararlo con la entrada y volver a enviar otra señal más oportuna hasta que la señal de entrada y la que detectamos en el actuador sean la misma.



Esquema lazo cerrado con buses de línea

En el esquema anterior se observa ya de una forma más clara las características del funcionamiento. Este es el siguiente: introducimos una entrada en ángulos, la comparamos con la "salida" y la diferencia de estas la introducimos en el controlador, el controlador será de tipo PID y dará una salida de 8 bits (0 - 255), esa salida será la que introducimos al actuador que en nuestro montaje físico se transforma en una tensión que hace girar a un motor de corriente continua ,como este motor esta unido a una hélice hace levantar una barra pero esto se explica con mayor detalle en otro apartado, lo que nos interesa es que la barra se levanta unos determinados grados, estos son medidos gracias un potenciómetro y transformados en una señal de 10 bits (0 - 1023) gracias a arduino, esta señal se transforma en el convertidor para que este a escala de voltios y poder compararla con la señal de entrada y volver a empezar otro ciclo.

Para hacer el esquema de simulink nos hemos ayudado de la librería que mathworks proporciona gratuitamente para trabajar con arduino, teniendo en cuenta el esquema anterior hemos hecho el esquema en simulink lo más parecido posible, este es el siguiente:



Lazo cerrado con Simulink

Este es nuestro esquema de simulink, en el la comunicación con la placa de arduino la establece el bloque denominado Arduino IO Setup el comando COM3 de este bloque indica el numero de puerto que tenemos conectado la placa de arduino, este número cambia con cada placa es decir que si se cambia la placa este circuito valdría siempre que se ponga el numero de puerto correcto.

El bloque Slider Gain es un bloque con el cual nosotros metemos la entrada que queramos. Hay que destacar que ese bloque esta acotado entre los valores de 45º y 135º que son los valores máximos y mínimos que nos permite físicamente nuestro sistema. Además cabe reseñar que con esta función mientras está funcionando el sistema se puede variar la barra varíe tiempo real.

El bloque PID Controller nos permite introducir un controlador PID pudiendo establecer los valores de la parte proporcional integral y derivativo que deseemos, dichos valores serán explicados de una manera más ilustrativa más adelante.

Después del controlador PID va un bloque de saturación que limita la señal a un rango entre 0 de mínimo y de 255 de máximo que es la señal que entra en el bloque Arduino analog Write, este bloque permite entradas desde 0 hasta 255 por ello hemos tenido que meter la saturación con anterioridad. El bloque Analog Write es el que nos permite escribir en el arduino un señal analógica en el Pin 6 que es donde físicamente tenemos la señal que queremos entregar.

Con el bloque Analog Read leemos la señal que arduino recibe desde el pin 1, que este está conectado físicamente a la patilla central del potenciómetro y nos da la posición de la barra.

Por último tenemos un convertidor hecho a partir de un bloque de simulink en el cual transformas la señal del potenciómetro (10 bits) en ángulos, esta señal en ángulos se lleva al comparador para restarla a la señal de entrada y sacar la diferencia, este bloque será explicado detalladamente en el apartado: lectura en ángulos.

Se ha añadido un scope (bloque que genera graficas) que unido a las señales de entrada deseada y a la salida en ángulos del potenciómetro, para que se pueda visualizar desde la pantalla del ordenador las dos señales y de una forma fácil poder comparar el funcionamiento del sistema.

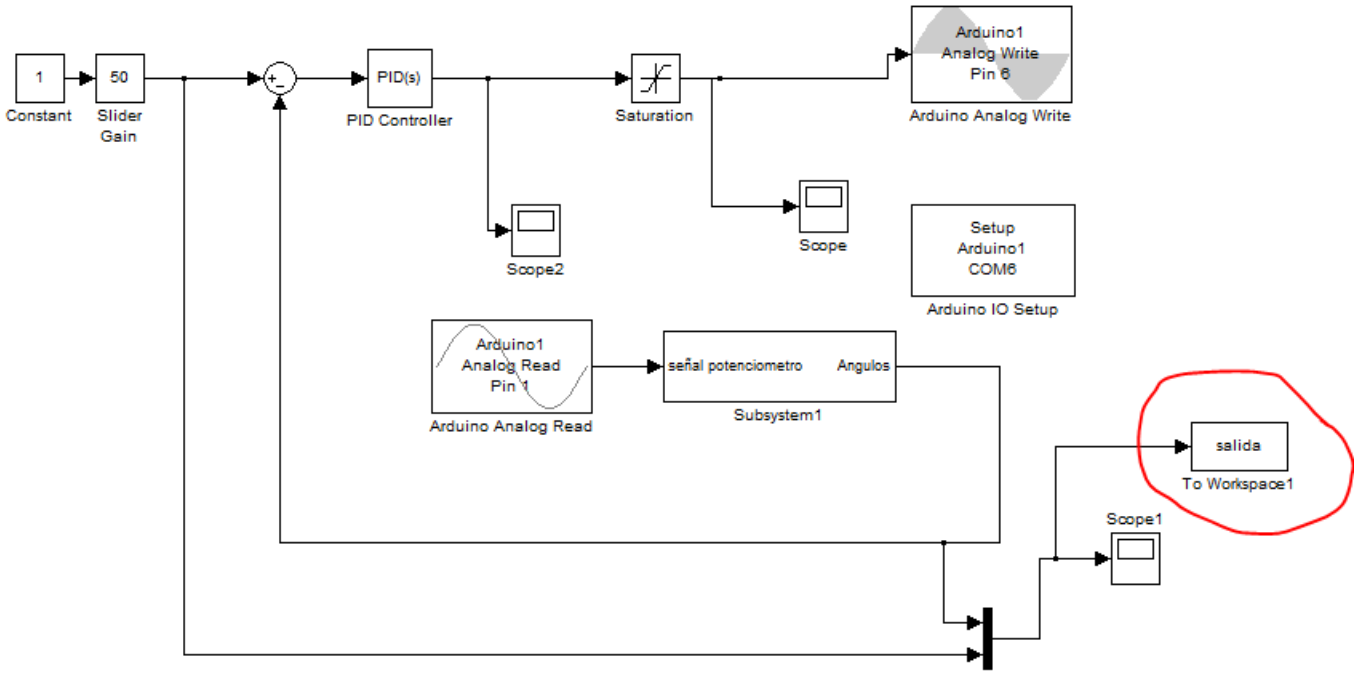
5.2.1. Obtención de los valores del PID.

Para buscar los valores del nuestro controlador PID hemos decido utilizar los criterios de Ziegler-Nicholls, Ziegler-Nicholls tiene dos métodos, en nuestro caso hemos elegido el 2º método ya que se hace a partir de un lazo cerrado.

Para nuestro controlador debemos elegir los valores de la parte proporcional que llamaremos a partir de ahora “P”, de la parte integral “I” y de la parte derivativa “D”, estos parámetros deberán satisfacer varias pruebas:

- Partiendo de una ángulo pequeño (pero siempre mayor a los 45º-46º que es el límite físico de la maqueta) alcanzar un ángulo medio (que será un ángulo entorno a los 90º).
- Partiendo de un ángulo pequeño llegar a un ángulo grande (consideramos grande a partir de los 110º).
- Partiendo desde un ángulo medio o grande estable (consideramos un ángulo estable cuando la barra se mantiene fija en la posición correcta) descender ese ángulo hacia otro más pequeño.
- Teniendo un ángulo estable introducirle una perturbación. (nuestras perturbaciones consistirán en un pequeño golpe o empuje físico, es decir estando la barra estable darla un pequeño toque hacia arriba o hacia abajo para ver lo que tarda el sistema en volver a la posición anterior al toque).

Para observar y trabajar mejor con las graficas de salida del sistema, hemos añadido en simulink un bloque que nos permite trabajar desde matlab una vez que hemos finalizado la prueba.



Esquema lazo cerrado simulink

Al ponerle un nombre a este bloque “salida”, podemos hacer referencia a la grafica desde Matlab y con el comando “plot” poder dibujarla, guardarla y trabajar con ella a posteriori.

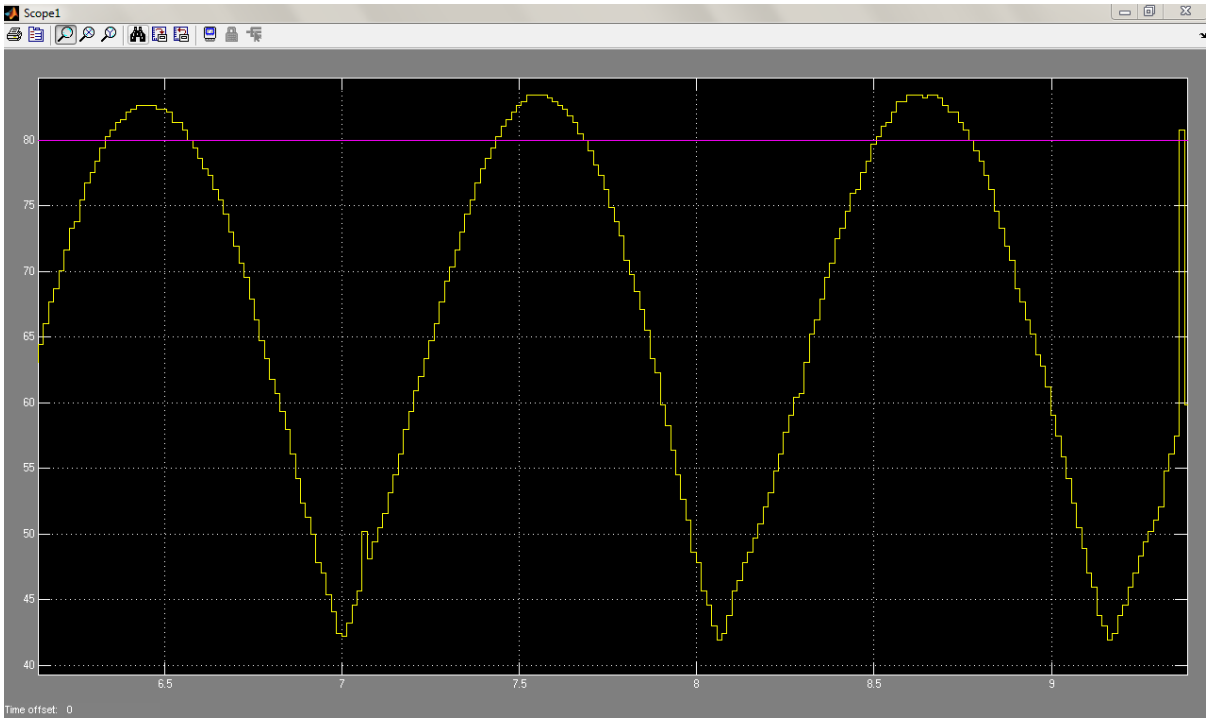
En este método que estamos utilizando lo primero es hacer funcionar al sistema con un control únicamente proporcional, y partiendo desde una ganancia proporcional cero hasta llegar a un valor

critico en el cual el sistema presente una oscilación mantenida, una oscilación mantenida es una oscilación que se mantiene periódicamente es decir que con el paso del tiempo la oscilación no tiende ni a aumentar ni a reducir, en términos matemáticos se considera como una senoide pura.

Una vez que hemos alcanzado la grafica del sistema con las oscilaciones mantenidas, tenemos que sacar dos parámetros:

- K_c= que es la ganancia critica a la cual hemos alcanzado las oscilaciones mantenidas.
- T_c= el periodo de las oscilaciones mantenidas

Para obtener nuestra grafica le hemos dado un ángulo medio de 80º para que el sistema pueda tener recorrido suficiente y a partir de aquí ir dando valores a la ganancia proporcional hasta encontrar la ganancia critica.



Grafica de salida con oscilaciones mantenidas.

En la figura se ven las oscilaciones mantenidas, la ganancia crítica es de 6.2, y el periodo de estas oscilaciones es de 1.08 segundos.

Ahora con estos datos nos vamos a la tabla de Ziegler-Nicholls:

Tipo de Controlador	K_p	T_i	T_d
P	0.5 K_c	∞	0
PI	0.45 K_c	$\frac{1}{1.2}P_c$	0
PID	0.6 K_c	$0.5P_c$	$0.125P_c$

Nuestro controlador va a ser de tipo PI ya que no es aconsejable hacer un controlador PID cuando las señales tienen ruido y como nuestro sistema tiene un ruido elevado hemos escogido un controlador PI.

Ahora solo nos queda sustituir nuestros valores de K_c y T_c en la tabla:

$K_p = 0.45 * K_c = 0.45 * 6.2 = 2.79$

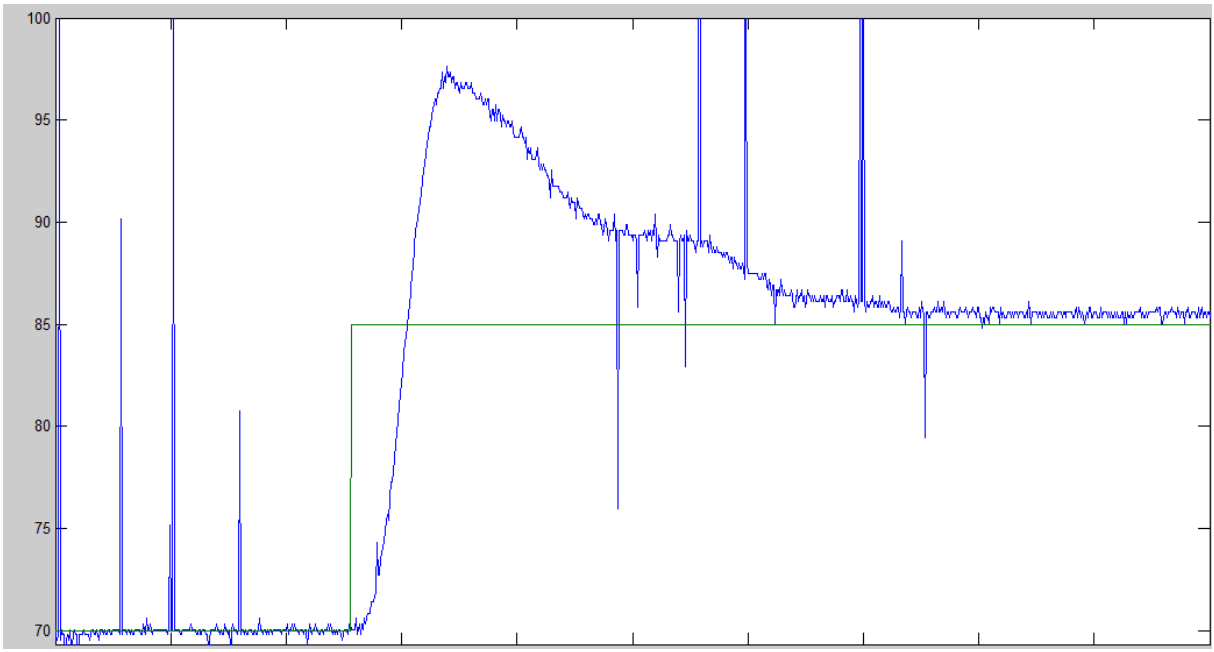
$Ti = \frac{1}{1.2}T_c = \frac{1}{1.2} * 1.08 = 0{,}9$

Como en los parámetros del controlador PID lo que hay que introducir son las ganancias el Ti no nos vale y necesitamos el Ki.

$Ki = \frac{1}{Ti} = \frac{1}{0.9} = 1.11$

Ahora solo nos queda probar nuestros valores (P=2.79 I=1.11):

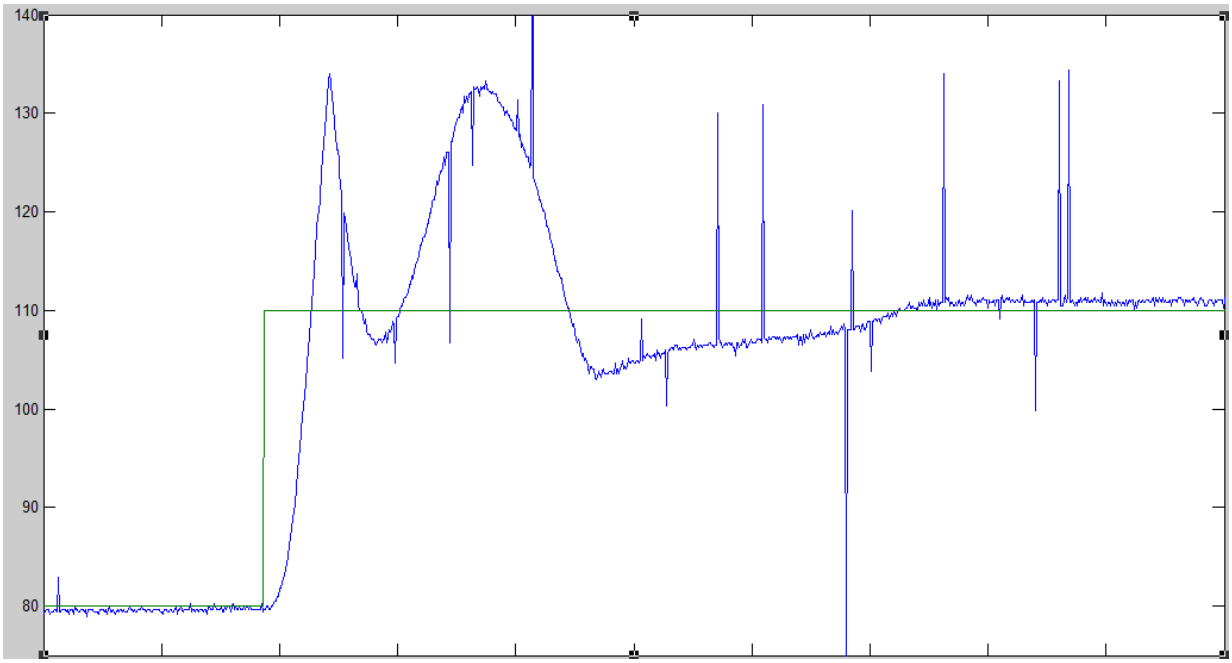
La primera prueba es partiendo desde un ángulo pequeño avanzar hacia un ángulo medio:



Grafica de 70º a 90º

Como se observa en la grafica el sistema pasa de una situación estable de 70º y le damos un salto de 15º para llegar hasta los 85º, se observa que hay un sobreimpulso que llega más allá de los 95º para ir reduciendo hasta los 85º.

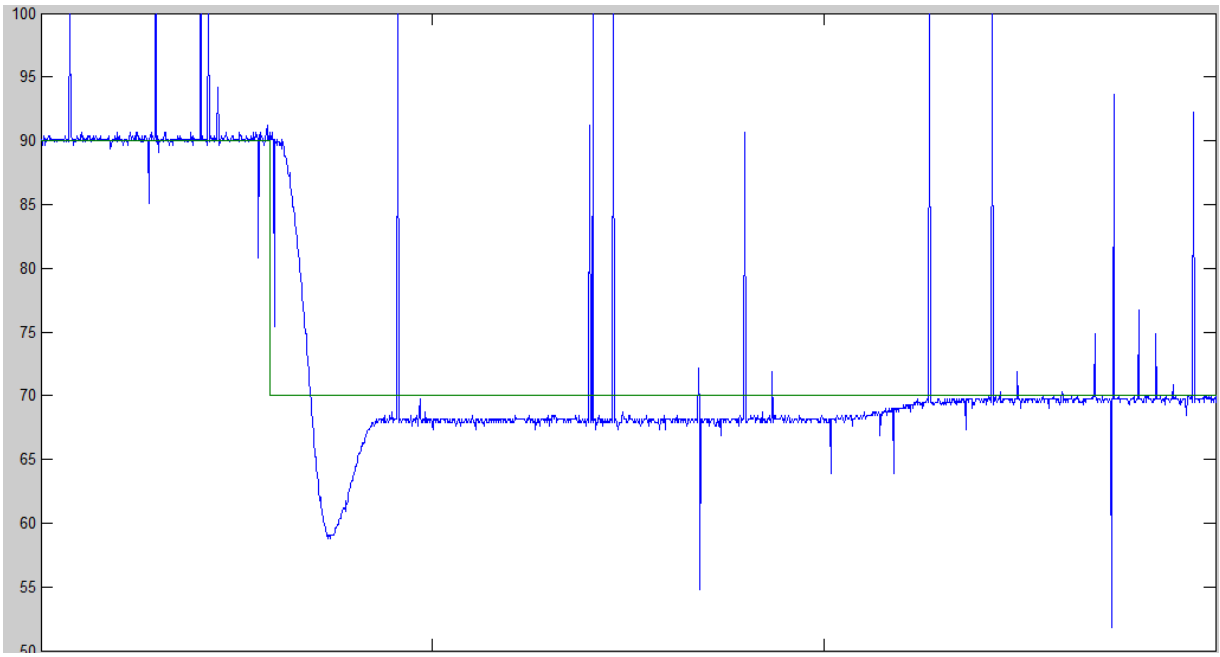
La siguiente prueba la vamos a efectuar a un valor elevado:



Grafica de 80º a 110º

En esta grafica pasamos de un situación estable de 80º a un nivel de lo llamado grandes de 110º, se observa que la grafica crece de una forma rápida hasta que se choca con la limitación física (136º) la cual la hace rebotar hasta los 110º y efectuar una oscilación que hace que el sistema sea un poco más lento.

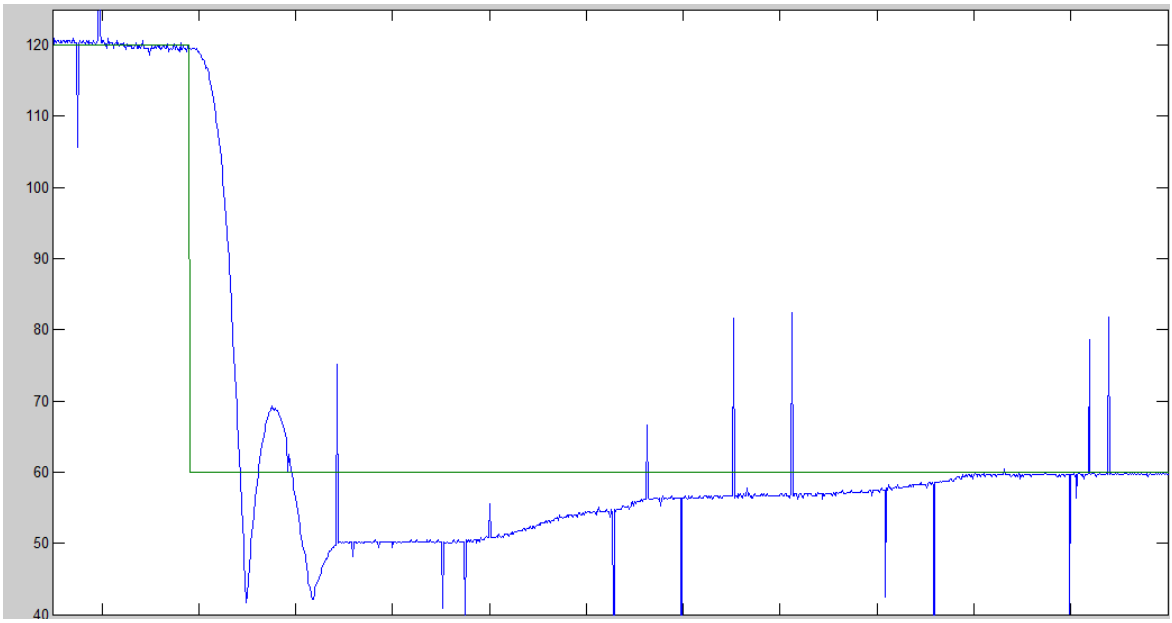
Para la siguiente prueba vamos a pasar desde un ángulo grande hacia otro más pequeño:



Grafica de 90º a 70º

En esta grafica se ve una circunstancia que es que cuando el sistema se acerca a su valor final (en este caso 70º) se ve como le cuesta reaccionar siendo su respuesta más lenta, pero la precisión es notable.

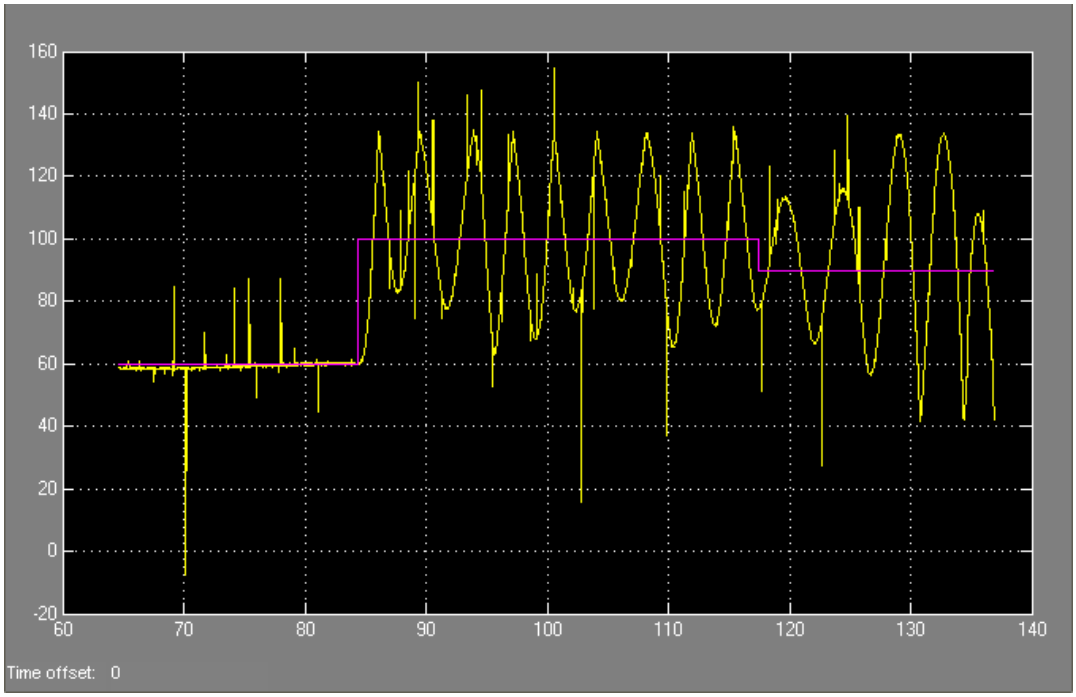
Para la siguiente prueba vamos a reducir el ángulo de una manera más drástica:



Grafica de 120 a 60º

En esta grafica se efectúa un descenso desde un ángulo muy grande 120º a uno muy pequeño 60º, como se puede observar el sistema choca con su limitación física (43º) rebota y vuelve a chocar, a partir de este momento empieza a crecer hasta que llega a su objetivo de los 60º.

Por último vamos a efectuar una prueba ante una oscilación:



Grafica de oscilaciones

Para efectuar esta oscilación hemos dado un pequeño toque al sistema para que el sistema ascienda, como la entrada no ha variado (100º) el sistema tiene que ser capaz de volver a ese punto, y como se observa en la figura el sistema no es capaz de estabilizarse creando oscilaciones cada vez con mayor amplitud.

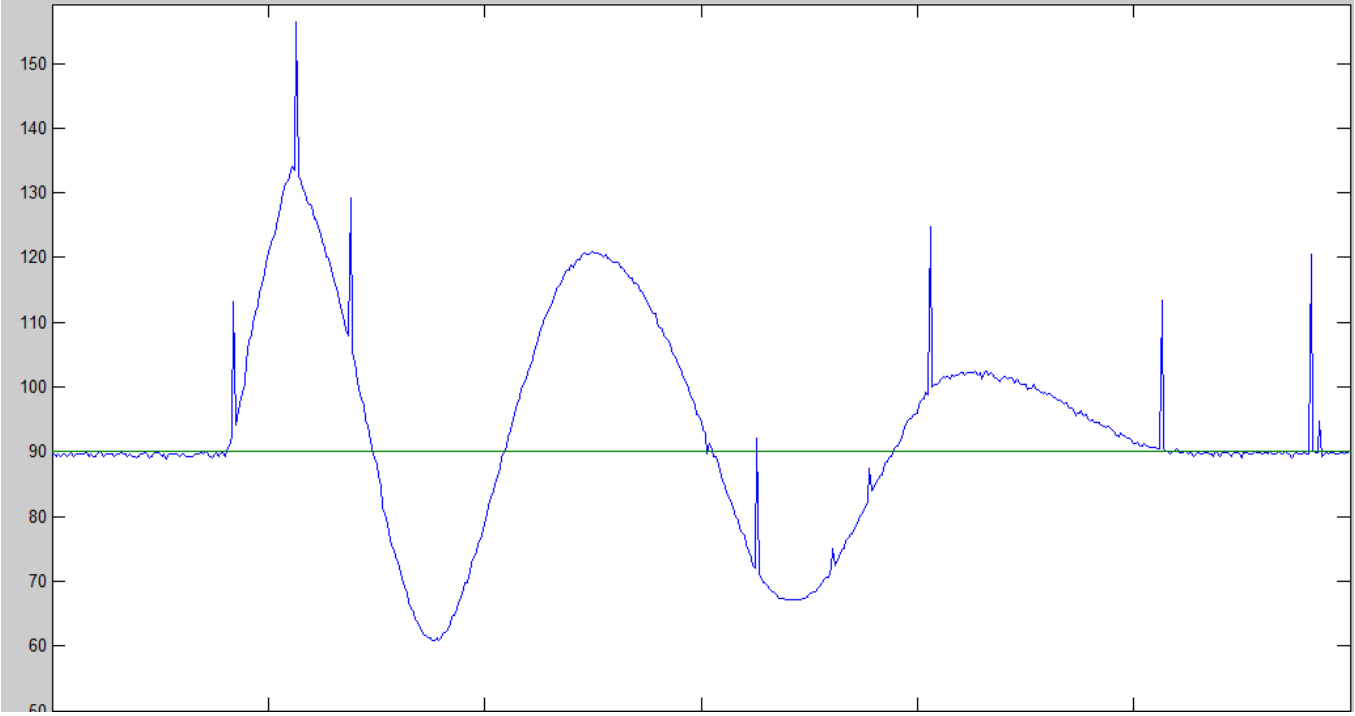
Partiendo de estos valores (P=2.79 I=1.11) y utilizando el método de prueba y error vamos a ajustarles para que sean capaces de responder a oscilaciones de la manera correcta y además que estos valores que encontremos también sean capacees de reaccionar ante entradas de tipo salto.

Tras varias pruebas de ensayo en las que hemos ido modificando tanto el parámetro P como el parámetro I hemos llegado a la conclusión de que los valores más acertados para nuestro sistema son:

P = 2,3

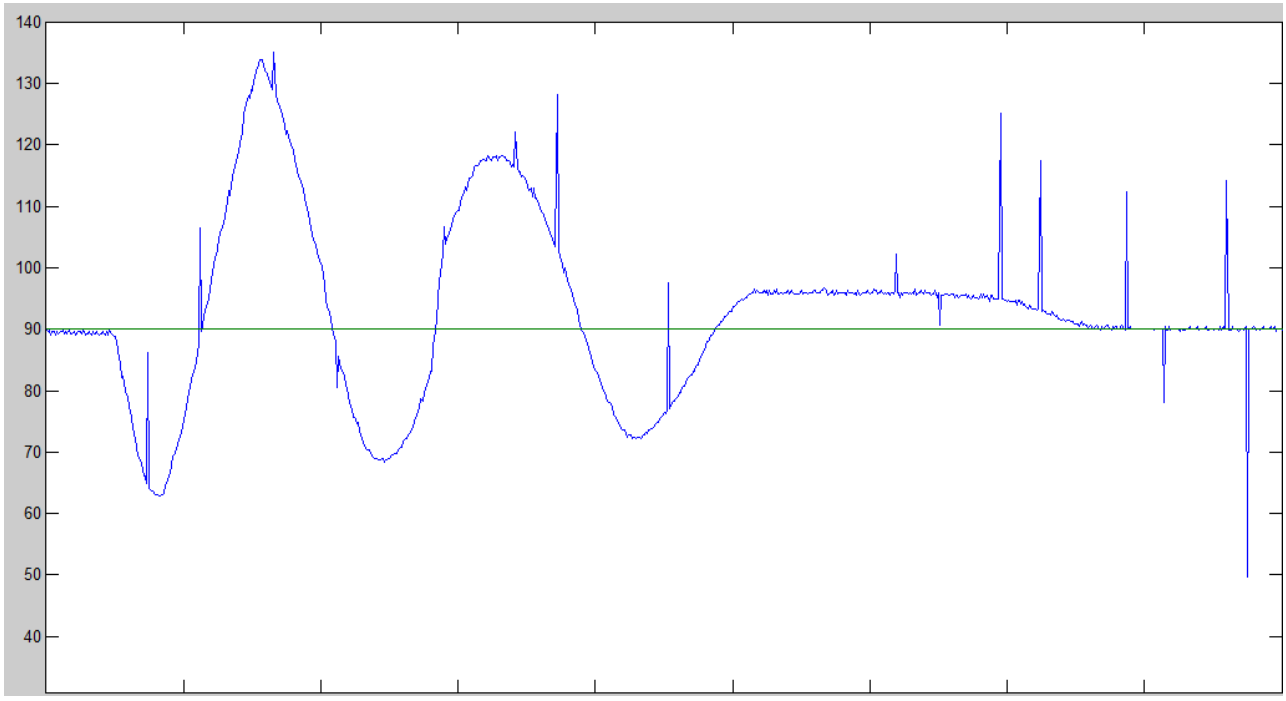
I = 0,91

A continuación mostramos las graficas que demuestran que estos valores funcionan correctamente:



Oscilación a 90º

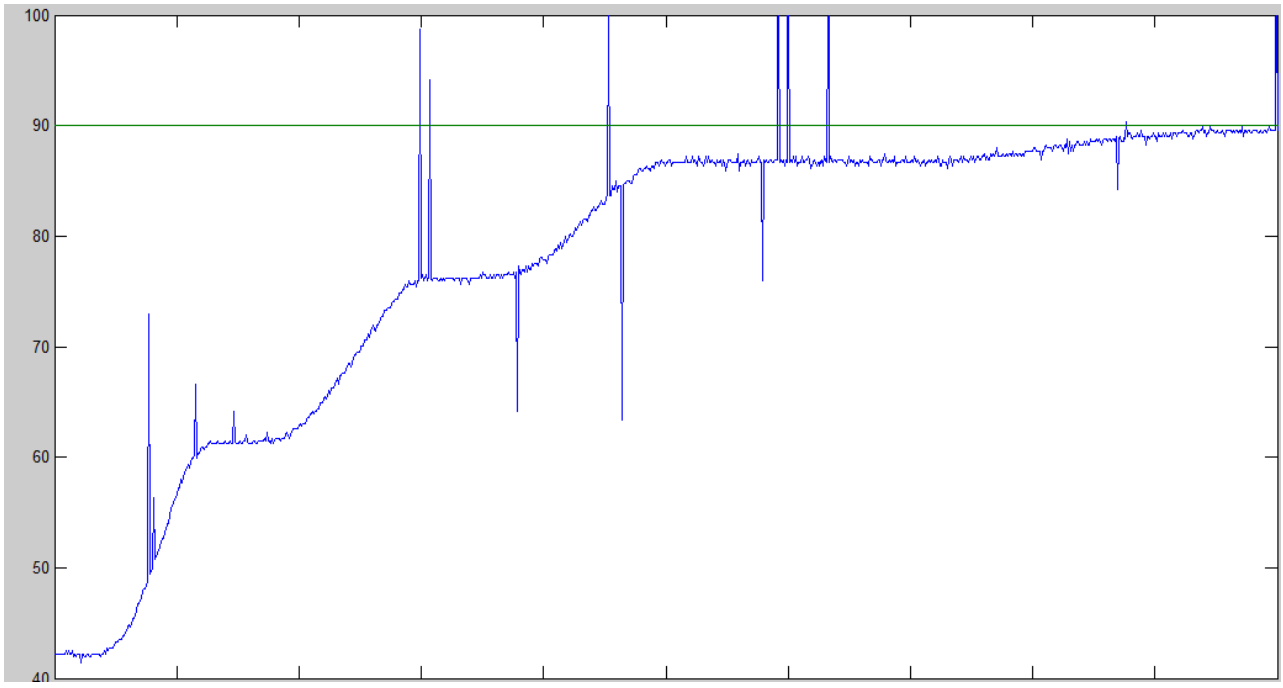
Como se observa en la figura con los nuevos valores del sistema las oscilaciones funcionan de una manera correcta.



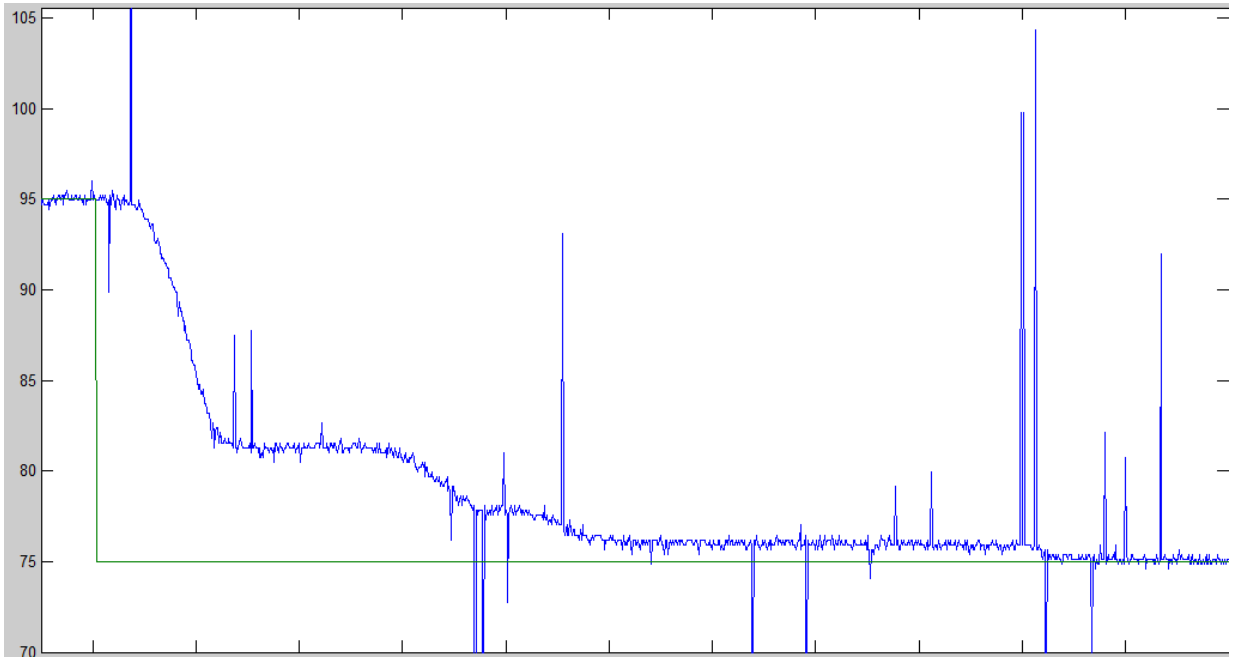
Oscilación a 90º.

En este caso la perturbación tendrá sentido opuesto, decrementando la posición de la barra, pero el resultado también es satisfactorio ya que el sistema es capaz de volver a la posición inicial.

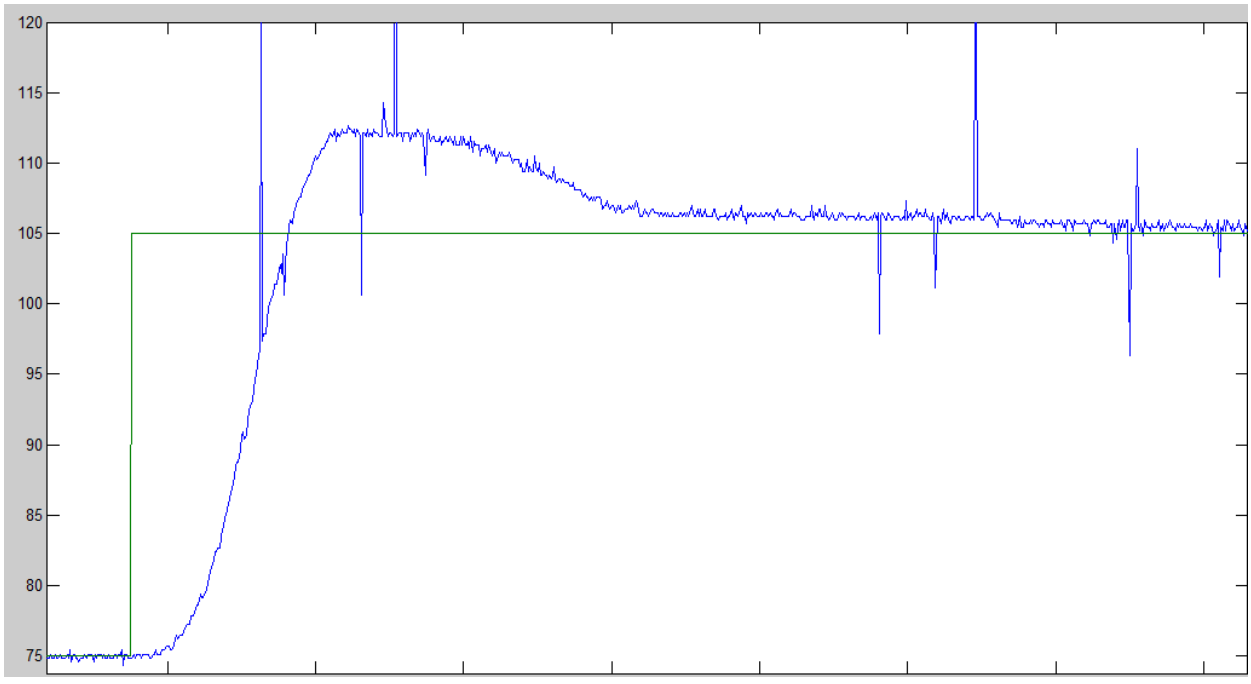
Con estos nuevos parámetros las entradas ante escalones funcionan de una manera similar a como funcionaban con los parámetros antiguos, este se ve reflejado en las siguientes graficas:



Grafica de 90º



Grafica de 95º a 75º.



Grafica de 75 a 105º

En conclusión los nuevos parámetros PI (P=2.3 I=0.91) han superado las diferentes pruebas a los que las hemos sometido, hemos observado que nuestro sistema es más preciso que rápido pero aun así tiene una velocidad correcta, además es estable para cualquier valor y reacciona bien ante las posibles oscilaciones ya sean ascendentes o descendentes.

5.3. Escritura en voltios

Es necesario un conversor que convierta los voltios que queremos darle a nuestro sistema (0-24v) en su correspondiente cifra de 8 bits.

Nos parece más conveniente que la entrada este dada en voltios, ya que los voltios son una magnitud física que se ve de forma más intuitiva a la hora de hacerse una idea de las variaciones que queramos hacer.

El transformador tiene entre sus bornes 24 voltios de corriente continua que gracias al sistema de potencia formado por el transistor mas el arduino podemos introducir el potencial deseado al motor siempre como máximo los 24 voltios, que son el máximo que podemos sacar y que coincide cuando escribamos en el arduino los 8 bits a uno o hablando en decimal le metamos 255, y le meteremos un potencial de 0 voltios cuando introduzcamos un 0 en el AnalogWrite.

Como nosotros lo que escribimos en el AnalogWrite es un digito de 8 bit (0-255 en decimal), hay que hacer una aproximación que aproxime los 0-255 de arduino con los 0-24 voltios que están entrado de potencial al motor, para ello cogemos valores de diferentes datos para poder hacer una tabla y ver qué tipo de aproximación sería la más conveniente.

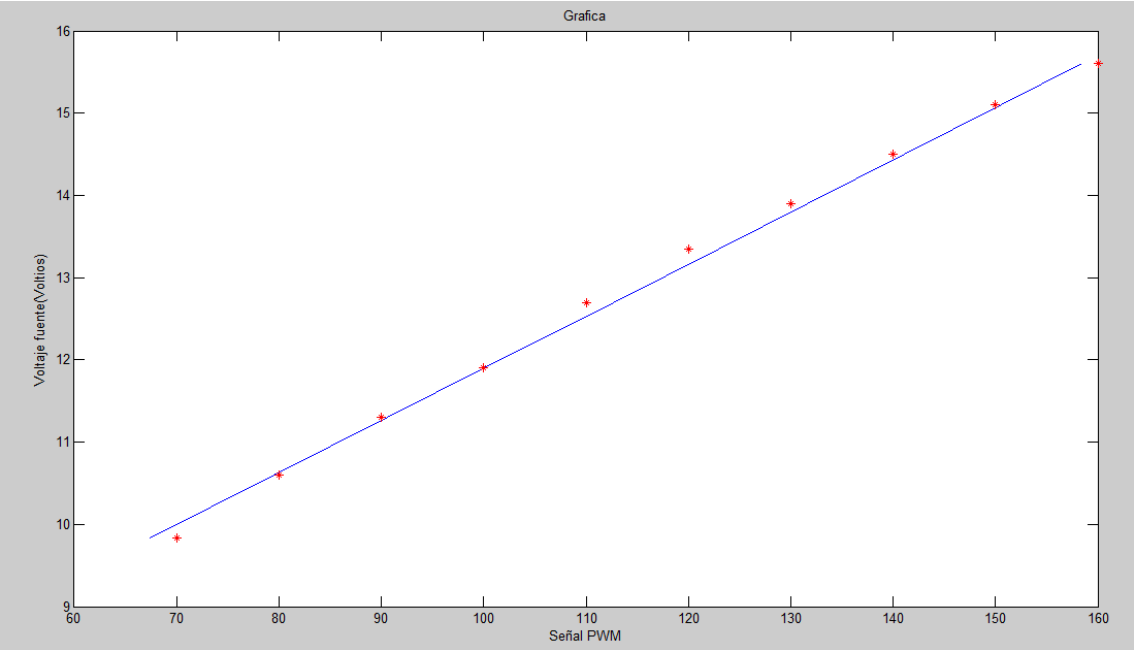
Los datos escogidos son:

Digito arduino	Potencial motor
70	9.83
80	10.60
90	11.30
100	11.90
110	12.70
120	13.35
130	13.90
140	14.50
150	15.10
160	15.6

Los datos escogidos varían desde 70 hasta 160 ya que es con el valor de 70 cuando el motor imprime la fuerza necesaria como para la barra despegue de su apoyo, y con 160 el motor tiene la fuerza necesaria como para llegar sin problemas hasta el otro resorte. Los demás datos que no entran en esta selección los hemos descartado ya que al seleccionar estos datos nos queda una relación lineal, que con una simple line recta hacemos la aproximación.

Dicha recta es la siguiente:

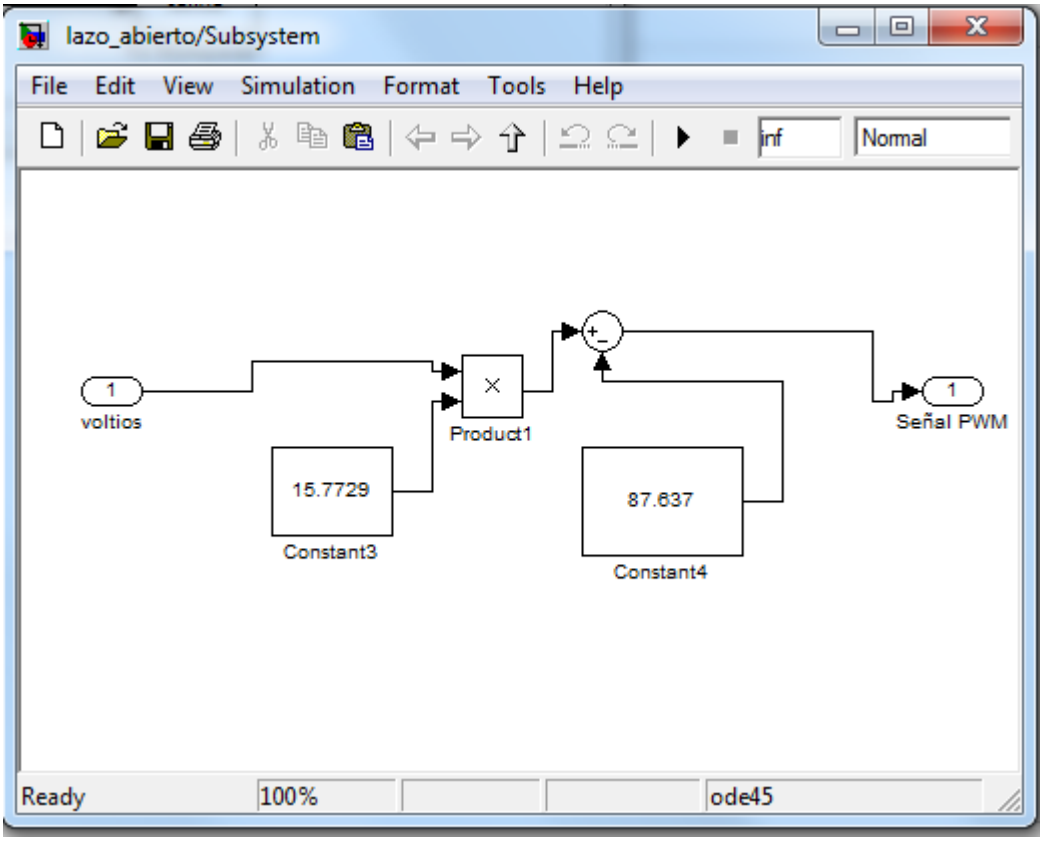
Y=15.7729*x – 87.637



Grafica Voltaje-Señal PWM

Como se puede observar en la grafica anterior los asteriscos representan los valores medidos en el laboratorio y la línea representa la aproximación lineal que hemos calculado. Se ve que la aproximación es bastante buena ya que en general la línea se acerca a los puntos, entonces a la hora de hacer nuestro conversor utilizaremos esta aproximación.

En MATLAB a la hora de meter esta aproximación utilizaremos la función Subsystem, donde los datos de entrada serán los voltios de la fuente y el dato de salida será el dato en PWM, para ello metemos la función de esta manera:



Bloque convertidor voltios señal PWM

5.4. Lectura en ángulos

Como se ha dicho con anterioridad la posición de la barra será medida a través del potenciómetro, este envía una señal analógica a la placa de arduino (en nuestro caso un voltaje comprendido entre 0 y 5 voltios) y la placa de arduino transforma esa señal analógica en un digito de diez bits. Es decir tenemos un valor en decimal de cero 1023 que dependiendo de la posición de la barra tendremos un voltaje en voltios entrando a la entrada analógica, la placa de arduino a través de un convertidor analógico-digital convierte nuestra señal analógica en una digital de 10 bits que nosotros lo que visualizamos en el ordenador es un numero decimal del 0 al 1023 que nos indica la posición de la barra.

En nuestro proyecto hemos colocado un transportador de ángulos para poder ver la posición de la barra, entonces tenemos que aproximar el ángulo de la barra que estamos observando con el valor que nos está ofreciendo la placa de arduino.

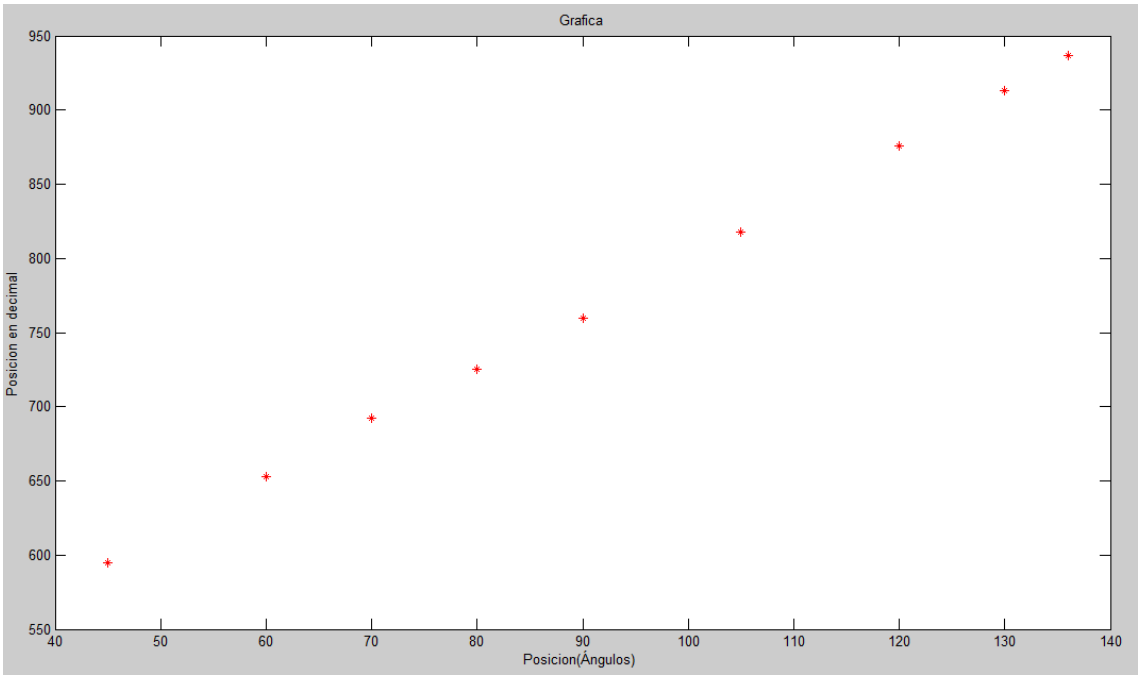
Hacemos una tabla que relacione el valor en grados con el valor en decimal de la posición, para ello es necesario un circuito de prueba a través de simulink. El más sencillo seria el formado por un bloque de AnalogRead (con el pin adecuado, en nuestro caso el pin numero 1) conectado a un visualizador como por ejemplo un scope. El movimiento de la barra lo hacemos de manera manual ya que es la forma más

sencilla, pero se podía haber hecho incluyendo en el programa de prueba un AnalogWrite junto con una slidergain una constant para mover la barra a través del motor y hacer girar la barra hasta el ángulo deseado.

Realizamos una grafica que nos indique las relaciones entre los ángulos y la salida en decimal, esta relación puede ser una relación lineal, parabólica u otra relación más compleja. En nuestro caso hemos cogido 9 datos:

Posición en ángulos	**Posición en decimal**
136º	937
130º	913
120º	876
105º	818
90º	760
80º	725
70º	692
60º	653
45º	595

Hay que tener en cuenta que debido a la limitación física el mínimo ángulo es de 45º y el máximo equivale a 136º.



Grafica de ajuste en ángulos

Al ver la grafica que formarían estos puntos nos damos cuenta que su relación es lineal, por lo tanto podemos relacionarlos con una línea recta a través de su fórmula:

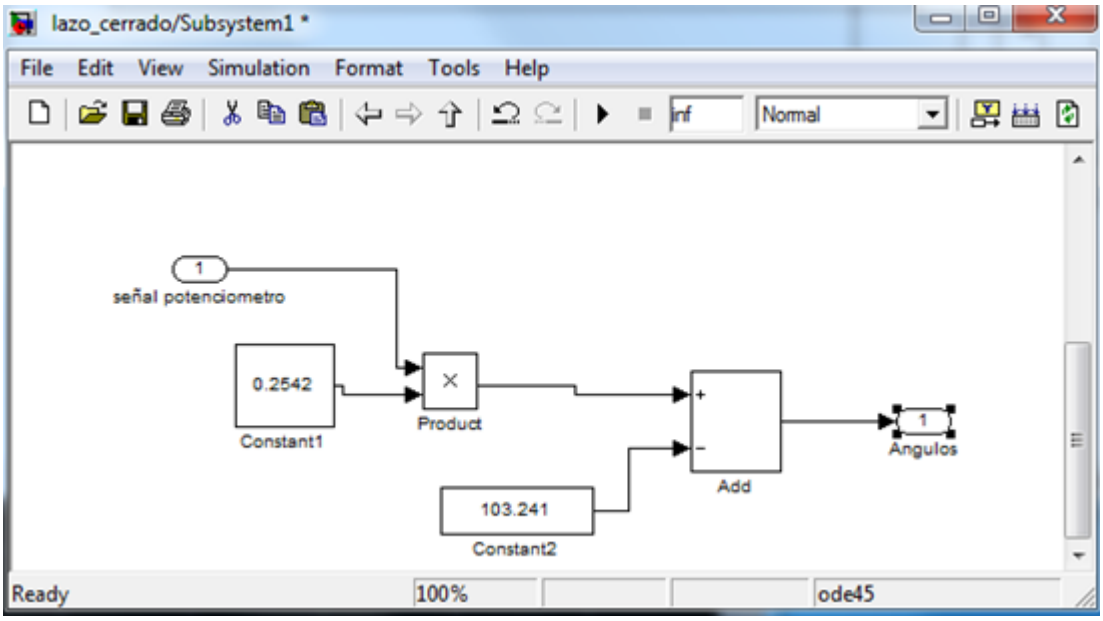
Y= a*x + b

Donde la "y" representa la posición en ángulos, la "x" la posición en decimal, la "a" es la pendiente de la recta y la b es la ordenada en el origen.

Al meter los datos nos queda una recta:

Y=0.2542*x – 103.241

Ahora hacemos un esquema en simulink que será nuestro convertidor de bits a ángulos, la señal de entrada a este bloque serán los bits en decimal y la de salida será en ángulos.



Bloque convertidor potenciómetro ángulos

6. Conclusiones

Los resultados conseguidos con estos proyectos han sido satisfactorios. Se han cumplido los objetivos que nos marcados al principio del inicio de este, obteniendo un correcto funcionamiento y control así como un completo conocimiento de todo lo que nos ha permitido llegar a ello.

Con la realización de este proyecto no ha permitido poner en práctica y desarrollar los conocimientos anteriormente adquiridos en nuestra formación, en especial los relacionados con la automática, control y electrónica en general. Esto nos ha permitido llevar a nuestro proyecto hacia una solución que podemos considerar óptima.

Arduino es uno de los grandes aportes que hemos descubierto con la realización de este proyecto, aunque no hayamos hecho un amplio uso de posibilidades nos ha permitido ser conscientes de todo su potencial a la hora de hacer cualquier tipo de proyecto relacionado con la electrónica. Una de

sus grandes ventajas es su versatilidad ya que puedes trabajar con él desde diferentes software en nuestro caso hemos utilizado arduino via Matlab-Simulink, este software nos ha permitido una poder trabajar con datos de una manera sencilla e intuitiva ya que hemos tenido un control directo del sistema desde la CPU.

Con la realización de este proyecto hemos aprendido a adaptar nuestro ordenador para trabajar con las diferentes placas de arduino desde la herramienta simulink de Matlab, lo cual conlleva la instalación de librerías en Matlab.

Nuestro sistema describe un movimiento circular, el trabajo con el nos ha permitido conocer sus características en las que destacamos: conceptos como momento de inercia y torque. Estos conocimientos junto a la 2º ley de Newton nos permitido identificar el sistema así como sacar la función de transferencia.

En referencia al circuito eléctrico lo más destacable ha sido conocer el uso del potenciómetro como sensor de posición angular y el uso necesario del diodo en antiparalelo con el motor, como elemento protector del circuito.

Hemos realizado un control en lazo cerrado de nuestro sistema con un controlador PID, con la ayuda del método de Ziegler-Nicholls y su posterior ajuste hemos identificado los parámetros del controlador PID, que hacen que nuestro circuito funciona de una manera correcta. Esto no ha permitido llegar a la conclusión que el controlador PID es idóneo para nuestro sistema y que a través del método de Ziegler-Nicholls nos da una idea aproximada para la identificación de sus parámetros.

Todo ello nos ha permitido cumplir nuestro objetivo principal que era el de control de posición de un balancín

Este proyecto tiene amplio uso didáctico, ya que su uso sencillo permitirá a los futuros alumnos trabajar con conceptos como lazo abierto, lazo cerrado, función de transferencia, controlador PID… viendo reflejado sus resultados en un medio físico.

7. Anexos

7.1. Guía de instalación

En esta guía vamos a separar dos aspectos fundamentales, en el primero vamos a explicar cómo instalar el software propio de arduino el que facilita la propia empresa de una manera gratuita, que no podía ser de otra forma ya que arduino es una empresa de software y sobretodo de hardware libre. En la segunda parte de la guía vamos a explicar cómo poder ejecutar arduino a través de un programa externo, en este caso simulink de MATLAB.

7.1.1. Instalación del software propio de arduino:

Lo primero que hay que reseñar es que arduino es una empresa de software y de hardware libre, como todo el software libre se puede conseguir de una manera gratuita. En este caso vamos a la página web oficial de arduino: http://arduino.cc/es/ en este caso la web en español, en esta web existe todo tipo de información de arduino, cuenta con un foro donde los usuarios pueden intercambiar experiencias proyectos etc., también cuenta con una tienda virtual donde poder comprar los productos de arduino de forma segura, además de talleres ejemplos proyectos donde los usuario pueden aprender a manejar arduino de una forma relativamente sencilla.

En la mencionada web hay un apartado donde poder descargar el software gratuito, es un apartado de nombre “descarga” si hemos accedido a la página web en español o bien “Download” si hemos accedido a la página por defecto en ingles.

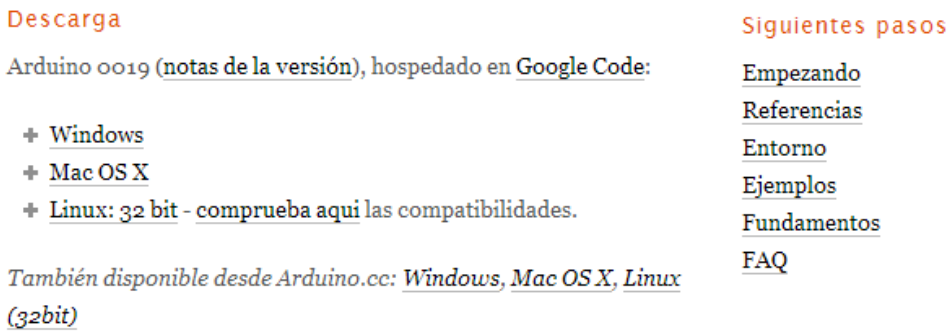
Arduino ha desarrollado un entorno interactivo de desarrollo denominado IDE (Integrated Development Environment), el IDE permite programar la tarjeta de una forma fácil, se basa en procesing y en wiring que es el lenguaje que utiliza arduino.

Arduino ofrece su IDE en varios formatos, dependiendo del sistema operativo que posea nuestro ordenador, actualmente arduino se llega por la versión número 19, pero te puedes descargar cualquiera de las IDE anteriores a esta por si tuvieras algún tipo de problema de compatibilidad.

En nuestro caso como tenemos que instalar la tarjetas en un equipo provisto de Windows elegiremos la última versión que hay para Windows, no importa la versión que se posea de Windows ya que este software no hace distinciones entre las diferentes versiones de un mismo sistema operativo, es decir da igual si por ejemplo le instalamos en un Windows vista o en un Windows 7.

En cada versión de arduino se explica cual han sido las funciones que han mejorado de ese programa o las que se han incluido de nuevas por lo que no está de más meterse de vez en cuando en la página web y revisar si hay nuevas versiones que nos puedan interesar.

Al darle a "Windows" directamente descargamos una carpeta.



Una vez descargada la carpeta, la descomprimimos y la colocamos en un lugar de fácil acceso ya que vamos a necesitarla en repetidas ocasiones. En esta carpeta nos encontramos las librerías, ejemplos, drivers y el propio programa en si llamado arduino.

En estos momentos ya poseemos el programa con el cual poder a empezar a escribir un programa en un entorno de arduino pero no hemos configurado nuestro ordenador para poder conectarlo con una placa de arduino.

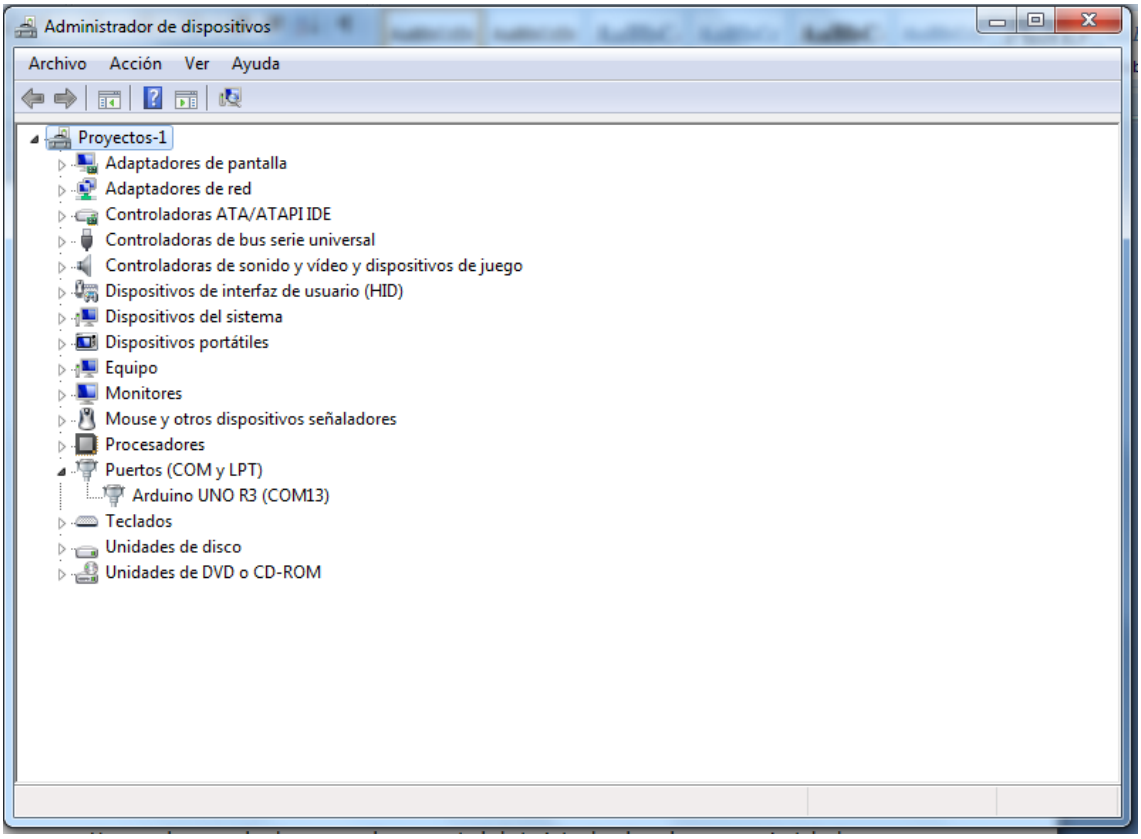
7.1.2. Instalación de la tarjeta de arduino

Cada tarjeta de arduino que conectamos la tenemos que configurar para poder utilizar correctamente, si la tarjeta no es la primera vez que la conectamos a un ordenador este la reconocerá y la pondrá un puerto, con lo que lo único que habría que hacer darse cuenta de ese número de puerto y colocarlos en el IDE (se explica de una forma más detallada más adelante), si es la primera vez que se enchufa la placa el proceso es más complicado, se explica detalladamente a continuación:

Lo primero que hay que hacer es conectar la tarjeta al ordenador para eso nos ayudaremos de un cable USB esté conectado el led verde "on" de la placa de arduino debe estar iluminado.

Como es la primera vez que conectamos la tarjeta a un ordenador, este no la reconocerá, y te dará una aviso para instalar software, si le damos a instalar software se intentara instalar pero dará un error diciendo que la instalación ha fallado, esto es normal se puede evitar no dando a instalar software. Para instalar correctamente los drivers nos tenemos que dirigir a la carpeta administrador de dispositivos (Windows -> panel de control-> sistema y seguridad-> sistema ->administración de dispositivos) en ella aparece todos los dispositivos conectados al equipo tales como el teclado el ratón la pantalla etc.

El que nos interesa es el que pone puertos (COM y LPT), al pinchar aparecen los puertos COM conectados al equipo debería salir (ya que nuestra tarjeta ya la habíamos conectado a otro equipo previamente) uno que pone Arduino UNO (COM xx), (sale como arduino UNO porque la placa que hemos conectado es la "UNO") para nuestro caso en particular el puerto que nos ha asignado el ordenador ha sido el 13.



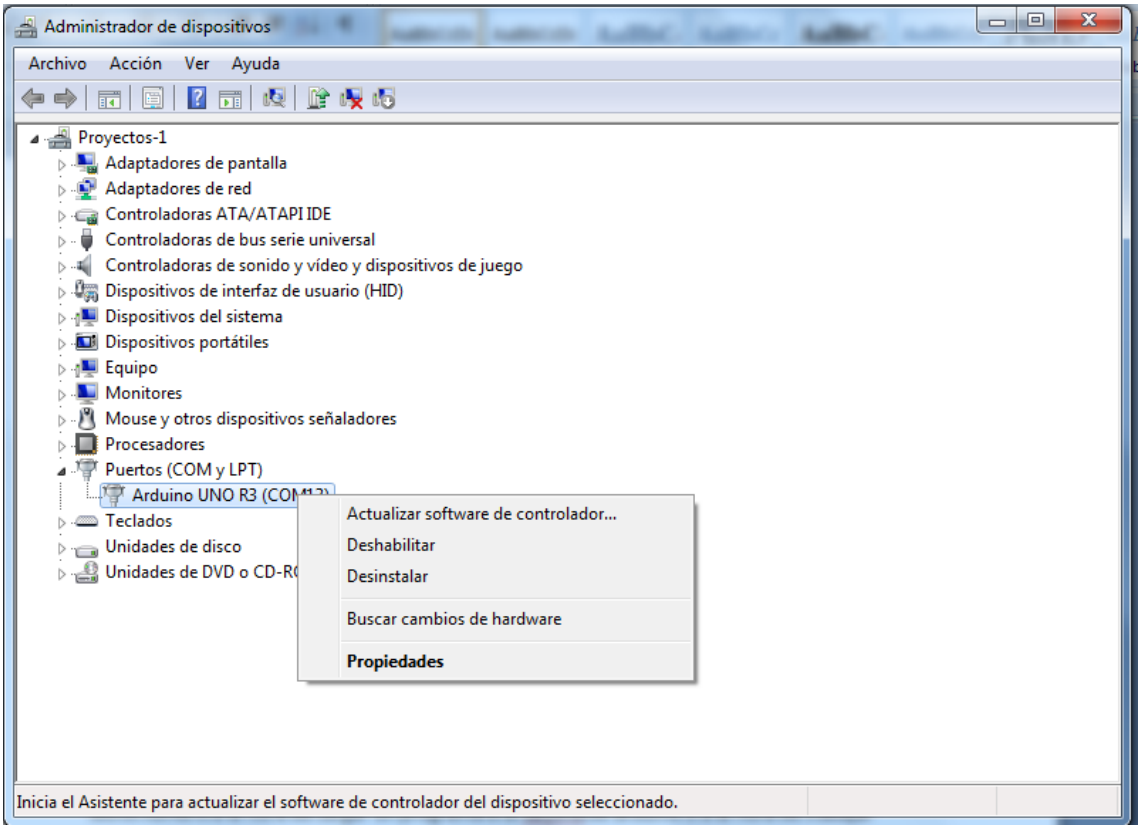
Administrador de dispositivos del equipo.

Si es la primera vez que introducimos la tarjeta a un ordenador entonces no aparecerá en los puertos (COM y LPT), sino que aparecerá como dispositivo desconocido, y no se le asignara ningún puerto de manera automática.

El numero de puerto que se os asigne es indiferente siempre y cuando se tenga en cuenta dicho numero a la hora de cargar un programa a la tarjeta de arduino o a la hora de trabajar con la tarjeta vía simulink pero ya os lo explicaremos en cada caso como hay que utilizarlo.

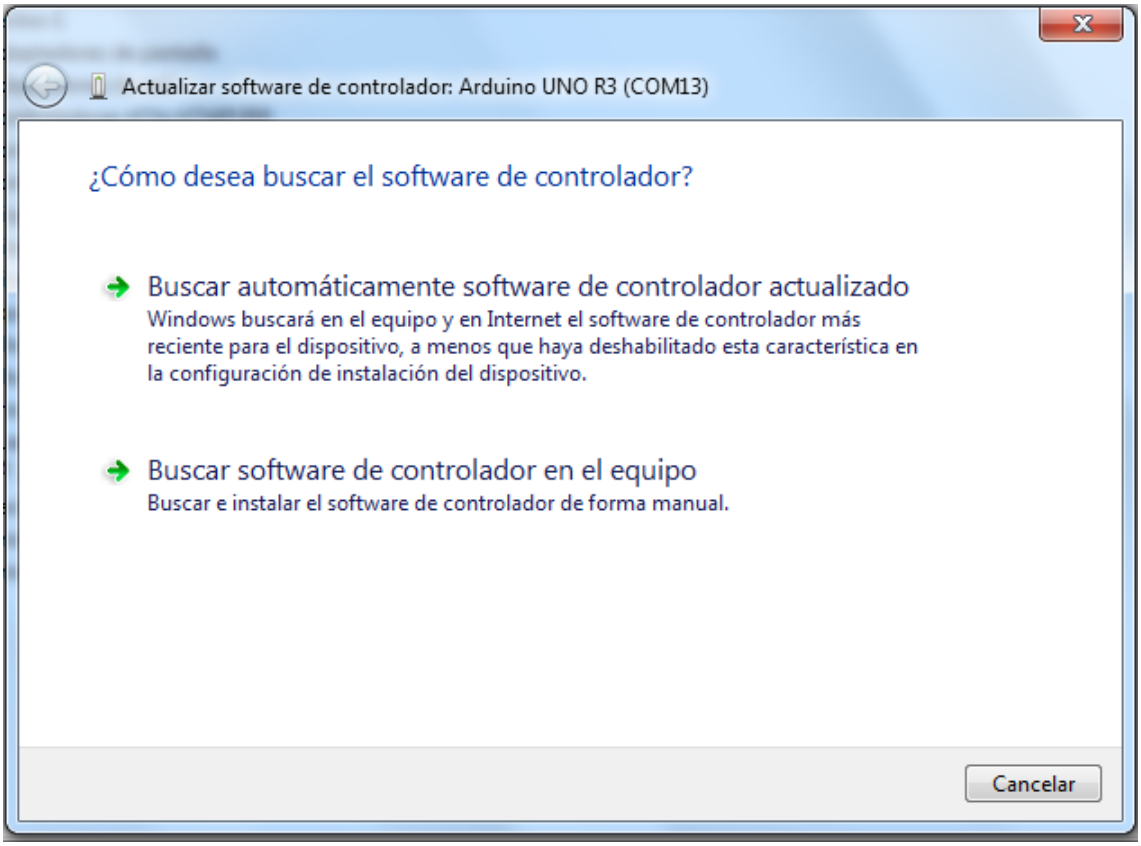
Independientemente de que la tarjeta sea la primera vez o no que la conectamos los pasos no difieren, porque le vamos a cargar los drivers del programa IDE de arduino que acabamos de descargar ósea que en un caso queremos que estos drivers se cambien por los que tenía antes o sino que los meta simplemente.

Para ello pinchamos con el botón derecho en el puerto que queremos, desconocido si es la primera vez, o en el numero de puerto que queremos configurar, en nuestro caso en el 13.



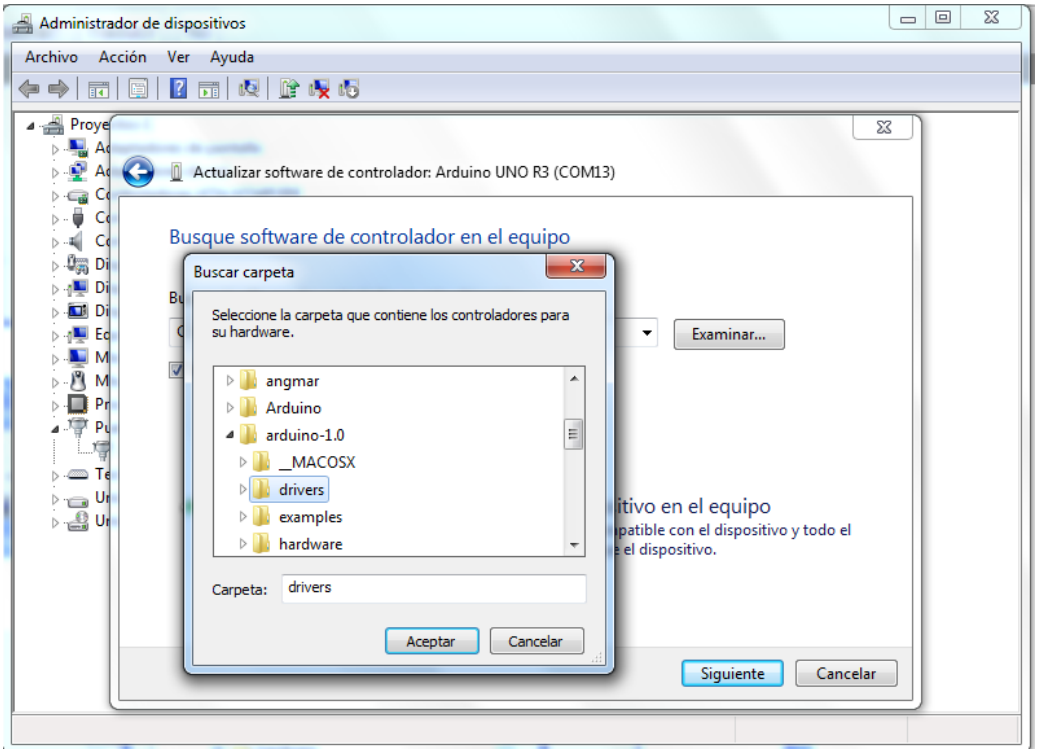
Puerto arduino

Pinchamos en la que dice “Actualizar software de controlador…” y nos saldrá esta pantalla:



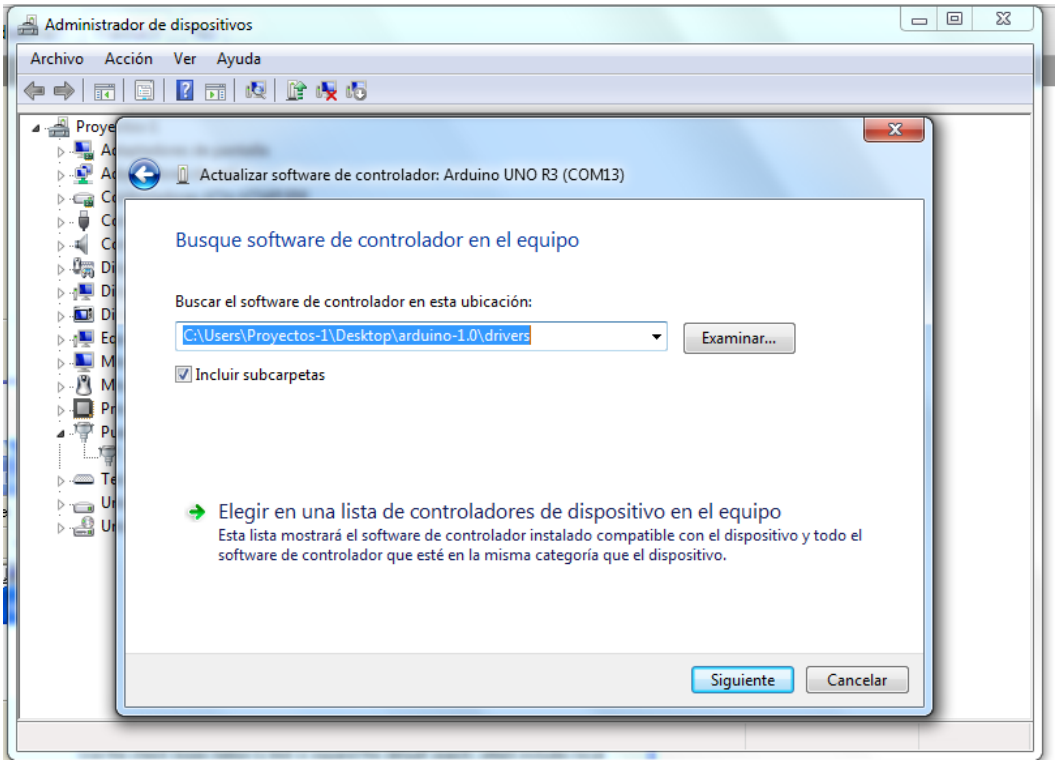
Configuración del puerto

Elegimos la opción de "Buscar software de controlador en el equipo", ya que en las carpetas que hemos descargado con anterioridad hemos descargado los drivers necesarios. Nos aparecerá otra pantalla donde pincharemos en el botón de examinar y buscaremos la carpeta de arduino que hemos descargado (en nuestro caso se llama arduino 1.0, pero si te has descargado una versión anterior o posterior el nombre será diferente) la abrimos y nos encontramos con otra carpeta llamada drivers la seleccionamos y le damos a aceptar.



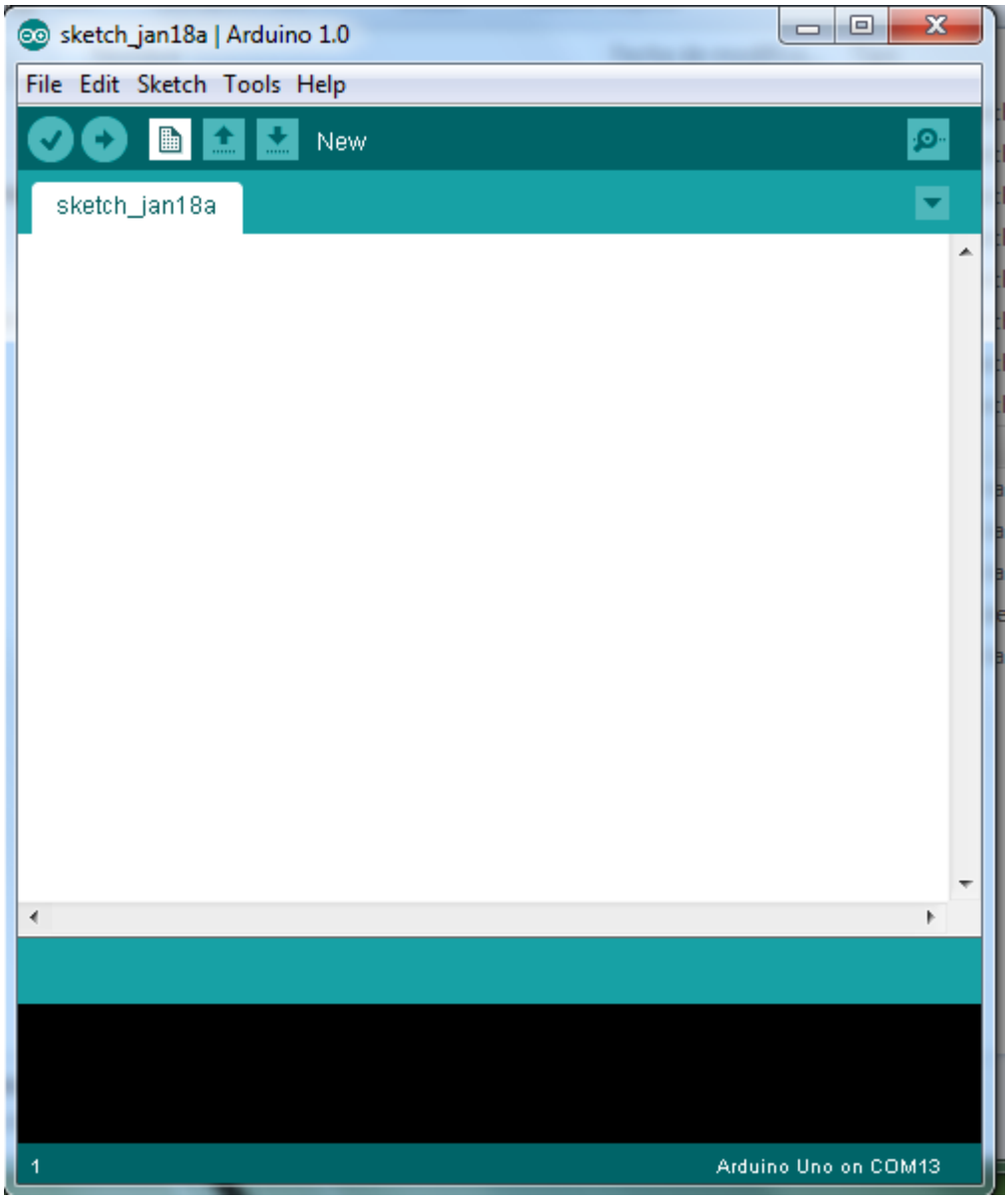
Selección del driver

Le damos a siguiente y comenzara la instalación del driver correcto, una vez finalizado le damos a finalizar y ya tenemos la placa de arduino lista para poder introducir los programas que deseemos.



Carga de drivers

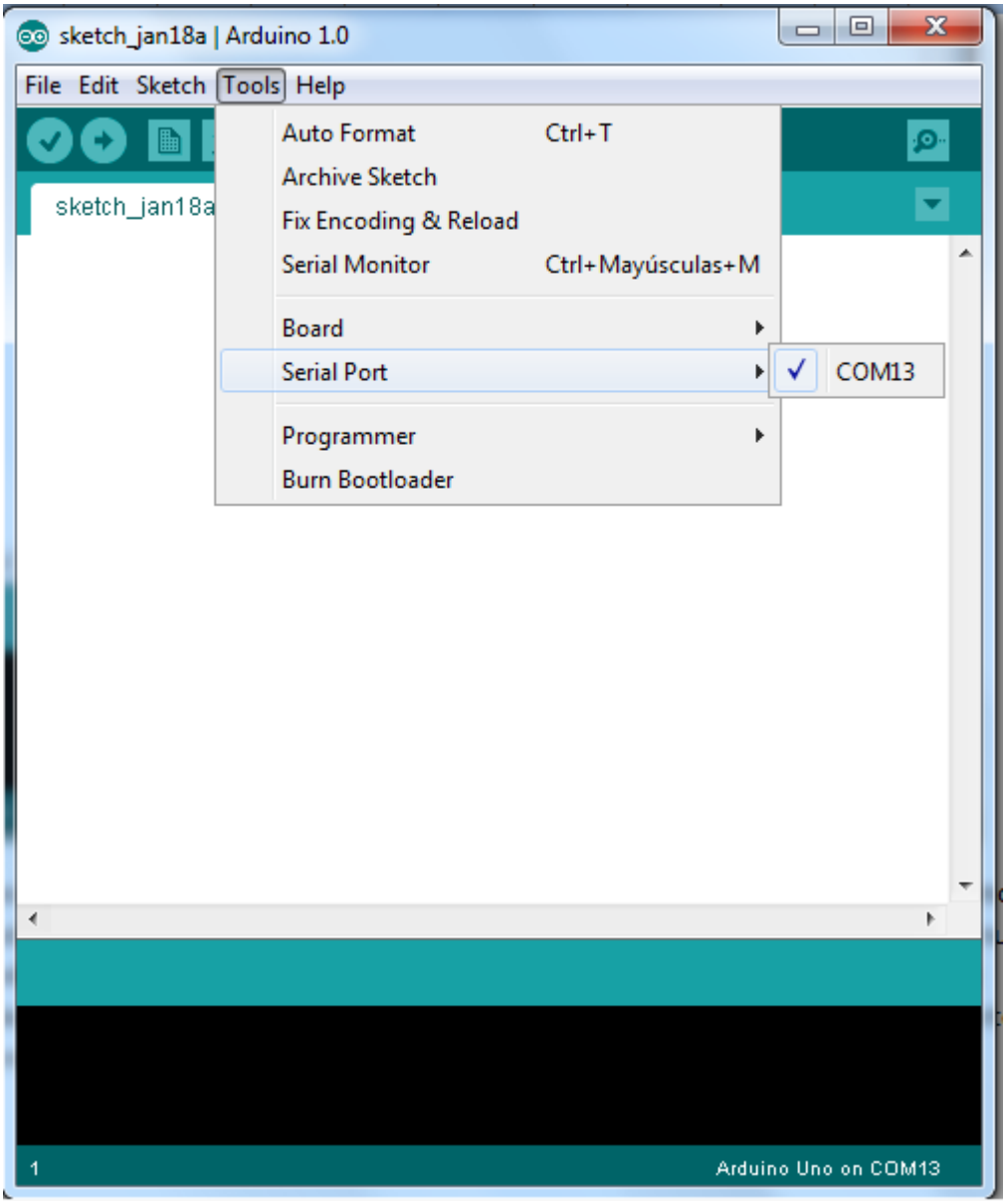
Una vez que hemos instalado todo lo necesario para el correcto funcionamiento de nuestra placa de arduino es la hora de empezar a cargarle los programas que deseemos, para ello vamos a la carpeta arduino-1.0, en ella aparece una aplicación de nombre arduino, la abrimos. La aplicación será algo similar a esto:



IDE de Arduino

Nos aseguramos que estamos utilizando la placa de arduino correcta, para ello pinchamos en Tools -> Board y comprobamos que esta seleccionado la opción Arduino UNO que es nuestro modelo de placa, en caso de que no tengáis seleccionado esta opción la tendréis que seleccionar. También comprobamos que estamos funcionando con el numero de puerto correcto para comprobarlo volvemos a seleccionar

Tools -> Serial Port y en ella tendrá que aparecer nuestro numero de puerto en nuestro caso en particular el COM13, hay que asegurarse de que esta seleccionado el puerto.



Selección de puerto en la IDE

Una vez que ya sabemos que estamos trabajando con la placa y el puerto correcto podemos empezar a escribir nuestros propios programas, utilizar los ejemplos que vienen en la IDE o bajarnos algún programa para comprobar que nuestra placa funciona correctamente.

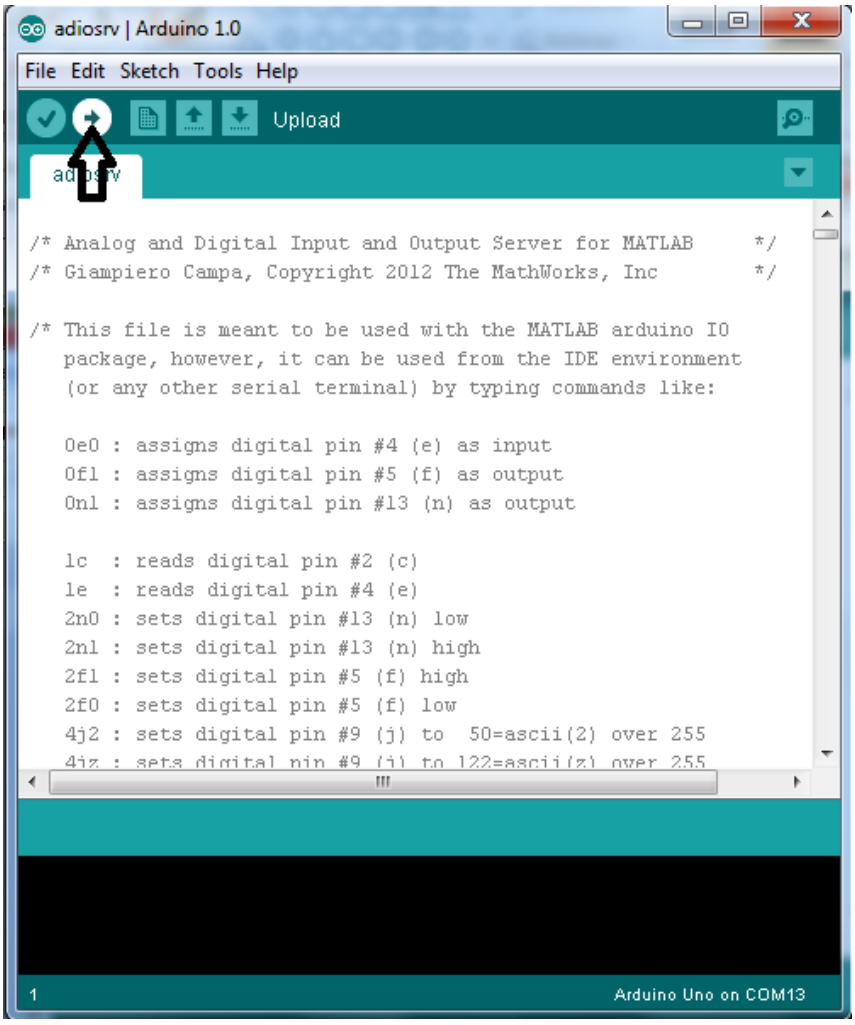
7.1.3. Instalar el software de Matlab/simulink

En nuestro caso vamos a cargar un programa predeterminado, este programa se encuentra en un paquete gratuito de mathworks, para ello nos metemos en la página de mathworks y nos descargamos este paquete:

http://www.mathworks.com/matlabcentral/fileexchange/32374-matlab-support-package-for-arduino-aka-arduinoio-package

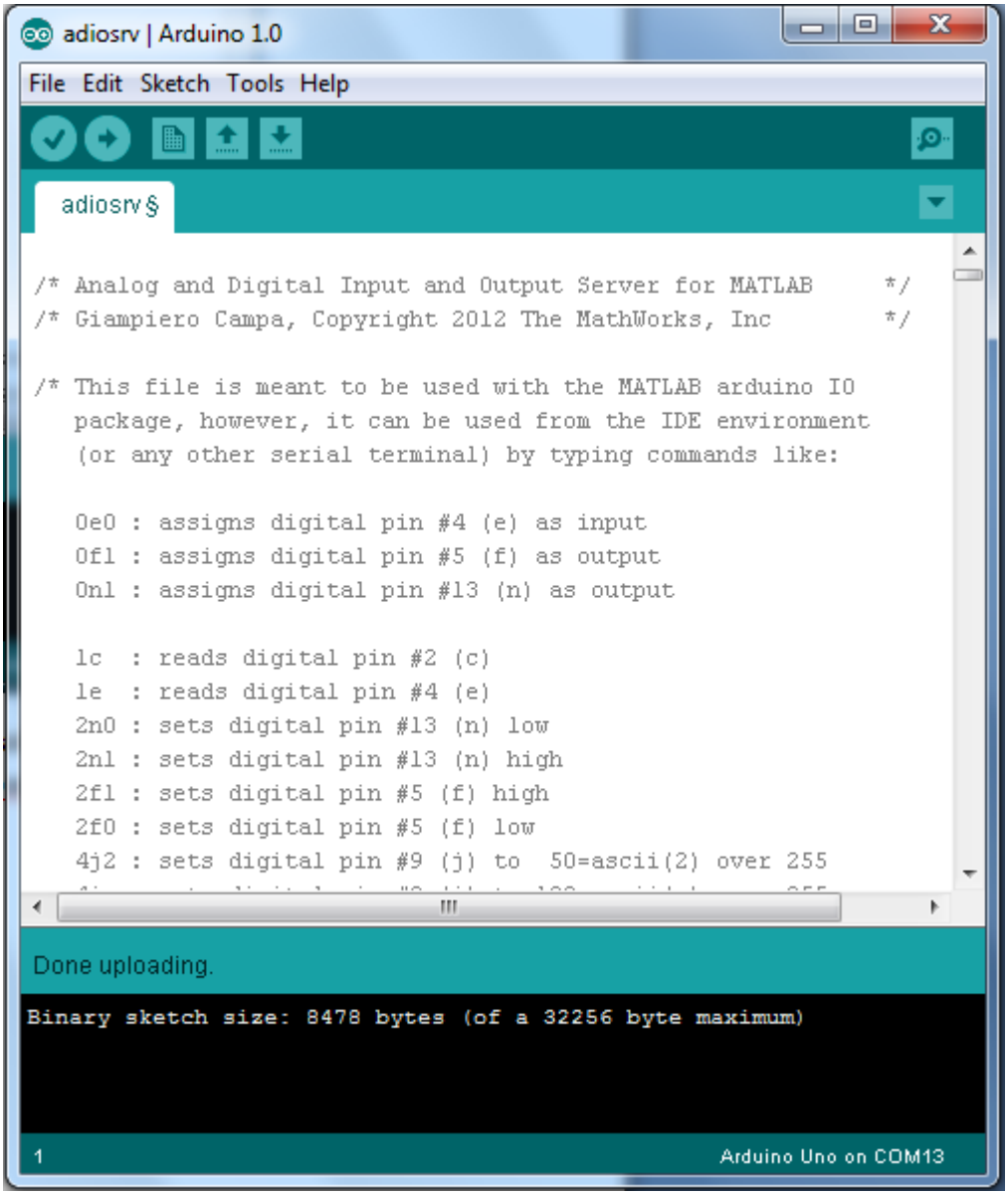
Con la carpeta ya descargada la guardamos en la carpeta de Matlab que se encuentra en nuestro disco duro para después poder encontrarla de una forma sencilla y de manera que nunca se pierda.

Ahora vamos a proceder a cargar el programa desde la aplicación de arduino para ello le damos a FILE->OPEN y abrimos el programa que se encuentra en la carpeta de MATLAB del disco duro, en la carpeta de ArduinoIO ->pde->adiosrv. Ahora lo que nos falta es cargar nuestro programa de arduino a la placa para ello le damos al botón de cargar (botón de la flecha mirando hacia la derecha)



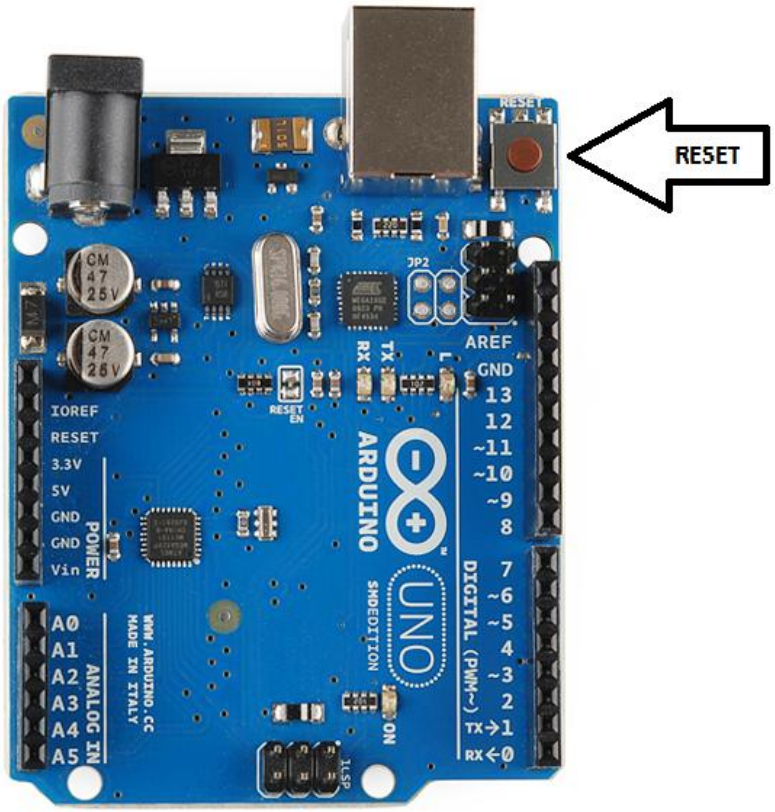
Carga del programa

Con ello lo que hacemos es compilar el programa y si no tiene fallos cargarlo a la placa, cuando la carga se realice correctamente deberá aparecer “done uploading” en la parte inferior y esa será nuestra señal de que el programa se ha cargado correctamente y ya podemos trabajar con nuestra placa.



Carga completada

En el caso de que queramos meter otro programa a la placa es recomendable resetear manualmente, para ello la placa arduino uno dispone de un botón, que al pulsarle reinicia la placa y ya podemos cargar otro programa, también es recomendable desconectar la placa del ordenador y si a la hora de cargar el programa te da un error de tipo que no encuentra el puerto, lo que hay que hacer es desconectar la placa totalmente, es decir todos los posibles cables que hemos colocado a los pines de nuestra placa y ahora debería funcionar.

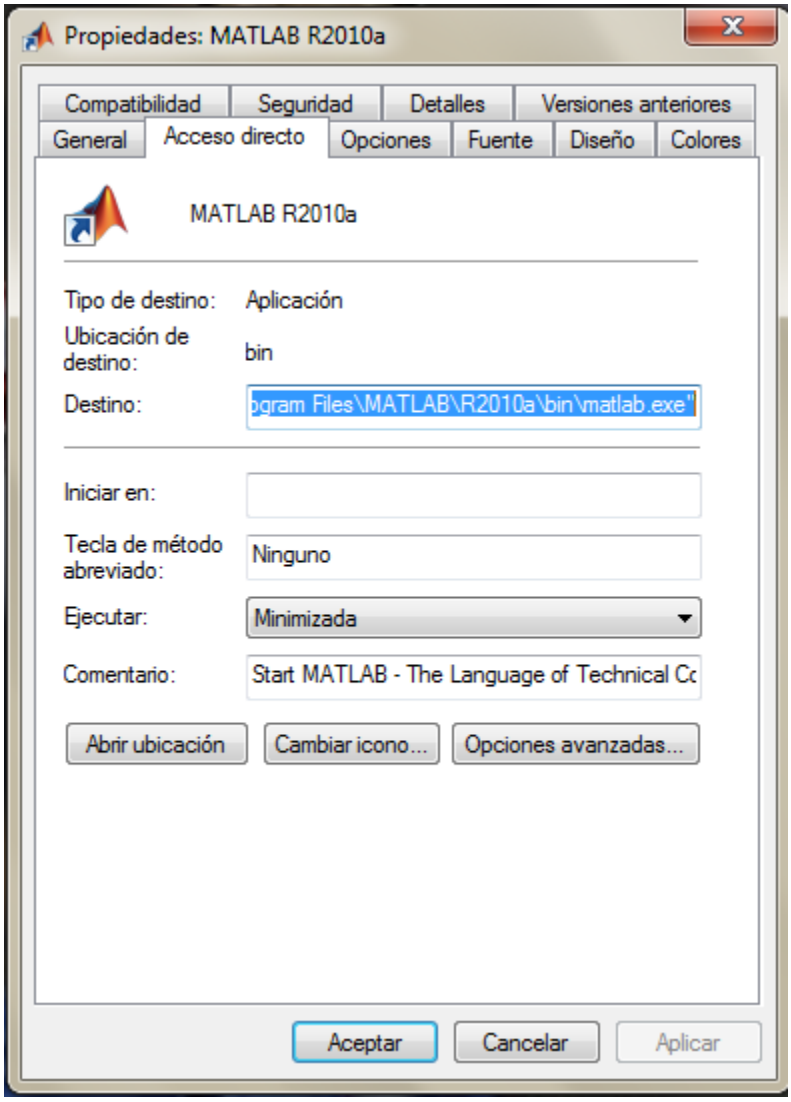


Tarjeta Arduino Uno

Para poder trabajar con arduino vía simulink tenemos que cargar una toolbox, en el paquete descargado previamente contiene una que es la que vamos a utilizar.

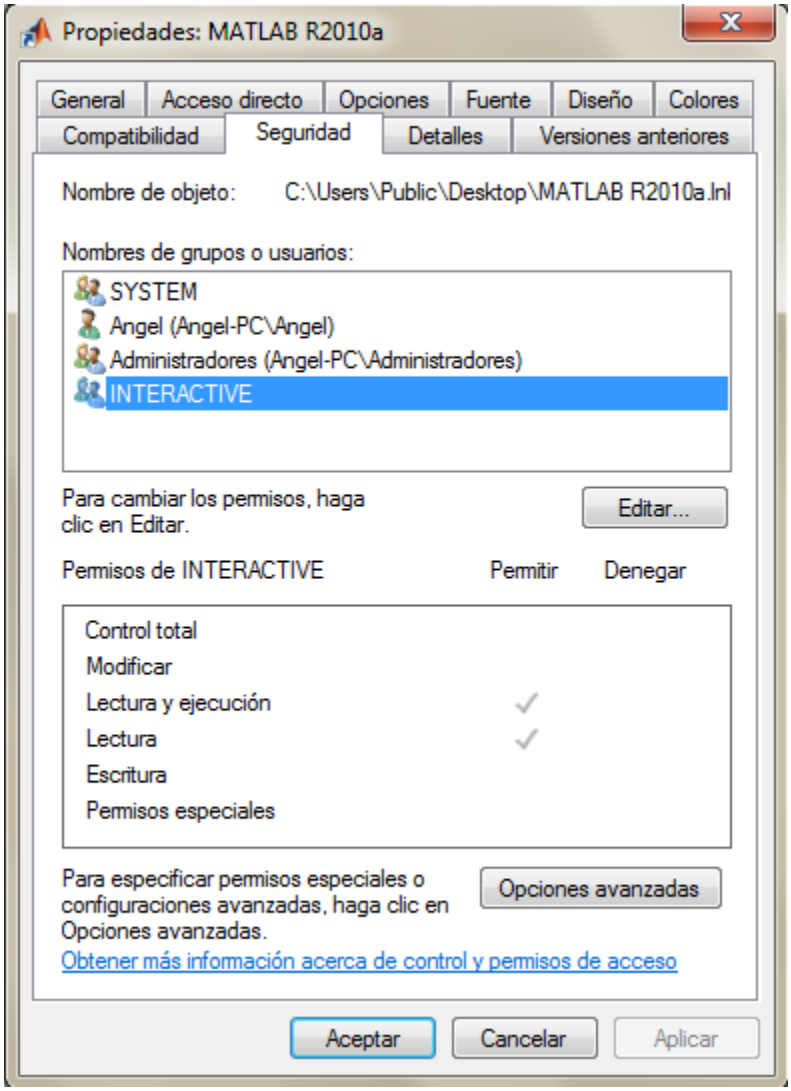
Hay que añadir que para instalar una toolbox a Matlab se necesita ser superusuario de Matlab, es decir se necesitan tener todos los permisos (escritura, lectura y ejecución), en el caso de que el usuario no posea todos los derechos no se podrá instalar ninguna toolbox.

Para poder obtener todos los permisos hay que dárselo previamente pulsando el botón derecho sobre el icono de Matlab, aparecerá una lista en la que tenemos que pulsar la opción d propiedades. Una vez que estamos dentro de las propiedades del programa de Matlab aparecerá una pantalla como esta:



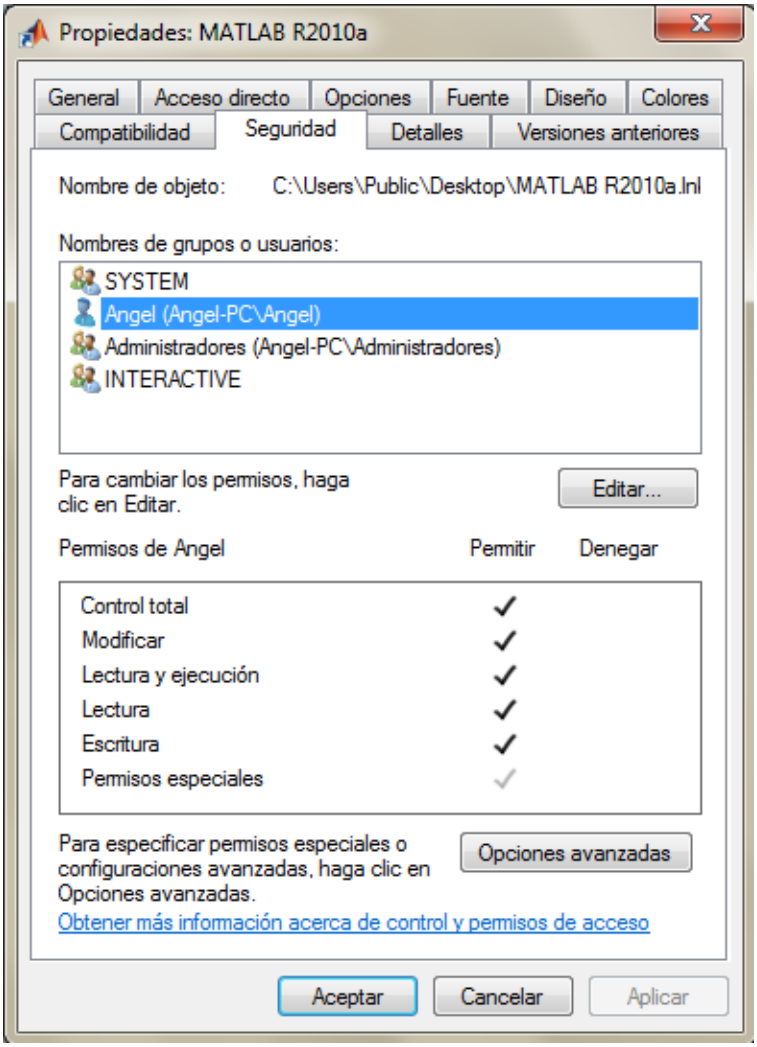
Ventana de propiedades

En esta ventana tenemos que ir a la pestaña de seguridad para poder cambiar los permisos, en ella aparecen todos los usuario que utilizan el equipo, tenemos que asegurar que el usuario que va a utilizar Matlab aparezca en esta lista, en el caso de que no apareciera hay que pinchar en opciones avanzadas, en ella aparecerá una lista con todos los usuarios, hay que distinguir cual es el usuario que queremos que utilice Matlab le seleccionamos y le damos al botón de agregar y lo aceptamos, ahora aparecerá este usuario en la lista:



Ventana de seguridad

Ahora tan solo queda darle permisos para ello seleccionamos el usuario que previamente hemos escogido y le damos a control total para darle todos los permisos, luego le damos al botón de aplicar para aceptar los cambios. Al hacerlo nos tiene que salir un “tick” en todas las opciones por ejemplo:



Ventana de seguridad.

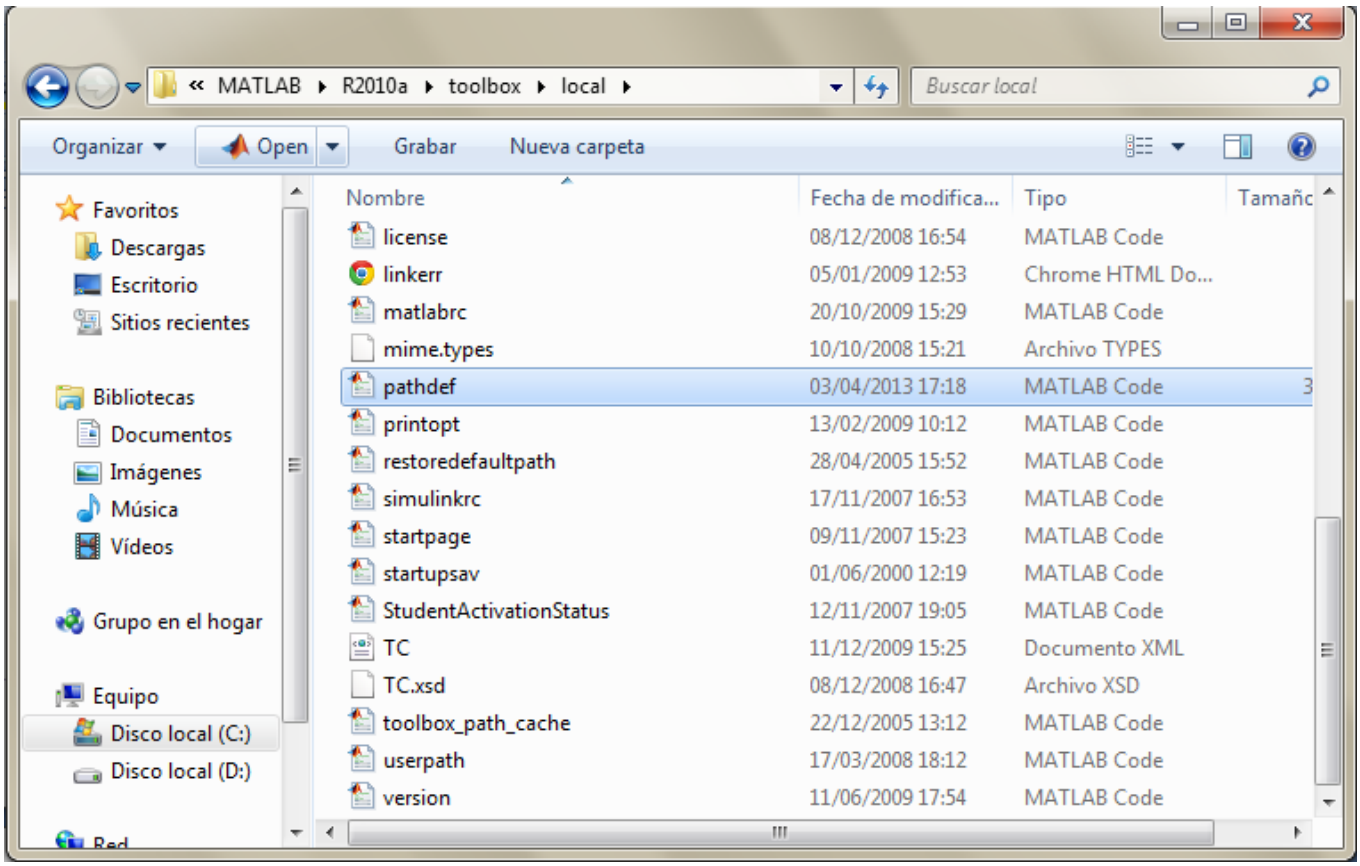
Ahora tenemos que asegurarnos que de un archivo especial de Matlab tenemos todos los derechos, este archivo se encuentra en la carpeta de MATLAB del disco duro, en este caso:

C:\archivosdeprograma\MATLAB

En este carpeta aparecerán las distintas versiones de Matlab que tenemos instaladas, abrimos la carpeta de la versión que vallamos a utilizar la toolbox (en el caso de que tengamos más de una versión es preferible instalarla en la versión más reciente). De entre todas las carpetas que aparecen escogemos “toolbox” y dentro de esta otra carpeta de nombre “local” y dentro de esta carpeta esta el archivo que buscamos, en mi caso el camino es el siguiente:

C:\Program Files\MATLAB\R2010a\toolbox\local

El archivo en cuestión es un archivo de Matlab de nombre pathdef.

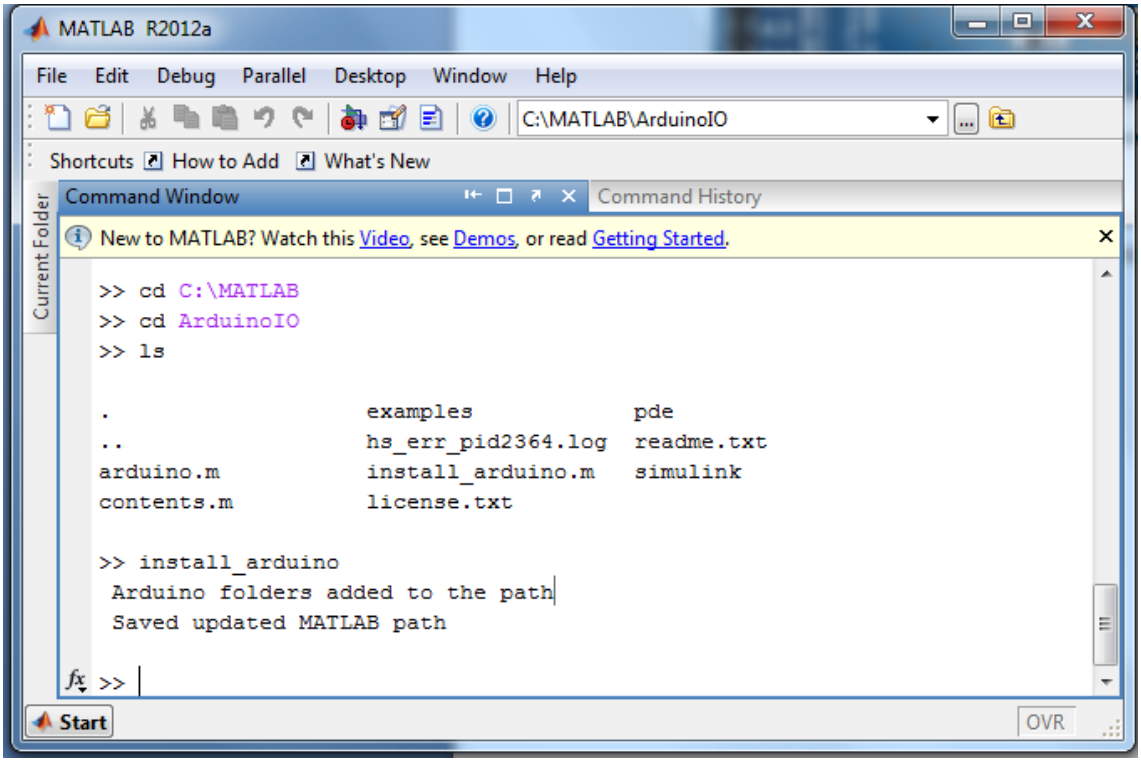


Carpeta local de Matlab.

Pinchamos con el botón derecho sobre este archivo, le damos a la opción de seguridad y ahí buscamos la ventana de seguridad. En ella buscamos nuestro usuario y nos cercioramos de que tenemos el control total, en caso contrario hay que darle los permisos de la misma forma que hemos hecho con Matlab. Este archivo solo tiene que tener todos los permisos no hay que ejecutarle ni hacer otra tarea con él.

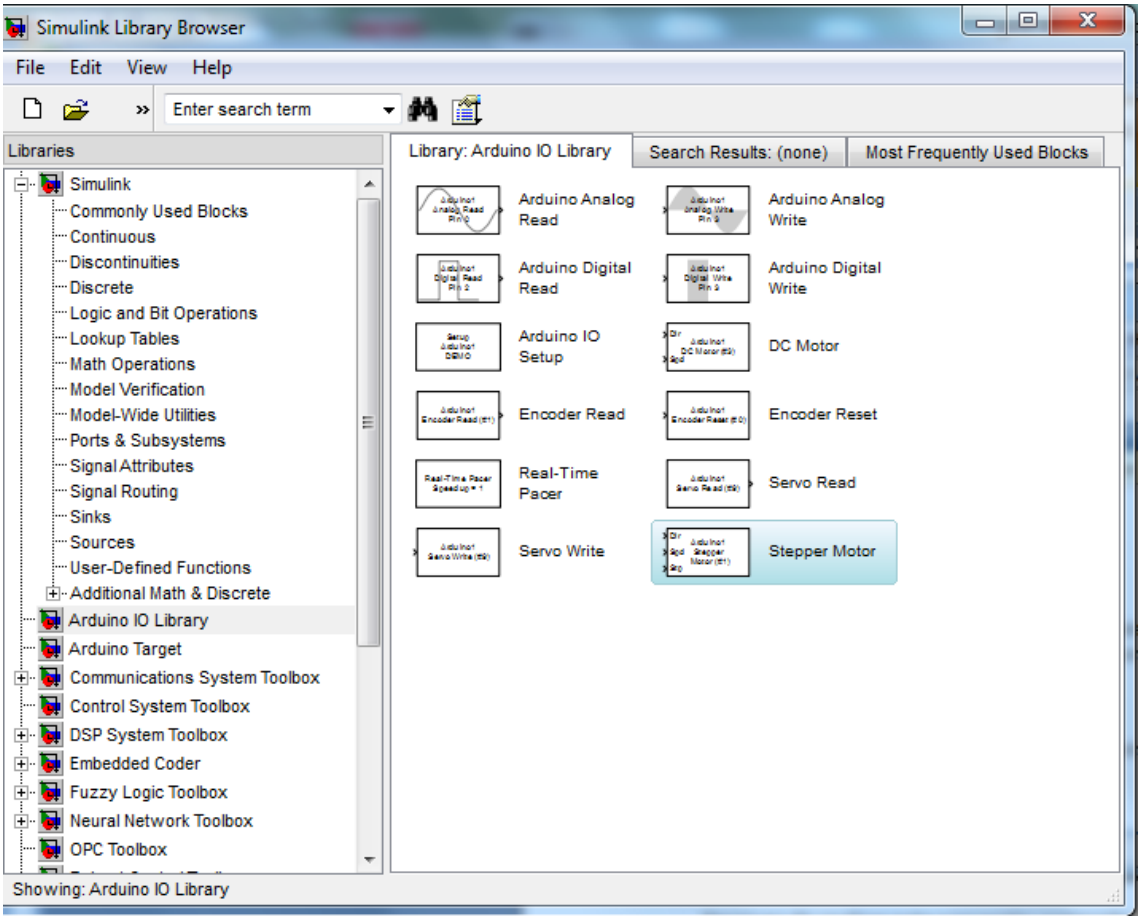
Una vez que tenemos todos los permisos vamos con la instalación de la toolbox.

Para cargarla necesitamos abrir Matlab y ejecutar el programa de install_arduino.m que se encuentra en la carpeta de ArduinoIO que previamente hemos descargado.



Carga de la librería ArduinoIO

Después de realizar esta operación comprobamos que se ha creado correctamente la nueva librería de simulink con nombre ArduinoIO Library donde se encuentran los bloques que nos harán falta para trabajar con la placa de arduino. Una vez que hemos instalado la toolbox hay que reiniciar Matlab para que aparezca en nuestro programa.



Bloques de ArduinoIO

7.2. Tabla de fuerzas

A continuación les montamos la tabla de fuerzas que ejerce nuestro motor, para ello lo que hicimos fue ayudarnos de un dinamómetro.

Cabe reseñar que este experimento lo efectuamos antes de poner a nuestro aparato el potenciómetro con lo cual el rozamiento es menor que el que tenemos luego en la práctica, además ese rozamiento será particular para cada dispositivo con lo que estas tablas solo serán validas para una aproximación ya que los datos no van a ser siempre exactos.

Hemos cogido dos tipos de valores según incrementamos o disminuíamos el voltaje del motor y los resultados son diferentes, también hay que indicar que los saltos de voltaje son pequeños.

INCREMENTAR EL VOLTAJE

POTENCIAL (Voltios)	FUERZA(Newton)
6	0.01
6.5	0.02
7	0.04
7.5	0.06
8	0.08
8.5	0.12
9	0.13
9.5	0.16
10	0.20
10.5	0.22
11	0.25
11.5	0.28
12	0.32
12.5	0.35
13	0.40
13.5	0.43
14	0.48
14.5	0.54
15	0.56
15.5	0.61
16	0.67

DECREMENTAR EL VOLTAJE

POTENCIAL (Voltios)	FUERZA(Newton)
15.5	0.66
15	0.63
14.5	0.58
14	0.54
13.5	0.50
13	0.47
12.5	0.42
12	0.39
11.5	0.35
11	0.32
10.5	0.28
10	0.26
9.5	0.23
9	0.20
8.5	0.17
8	0.15
7.5	0.12
7	0.10
6.5	0.09
6	0.07
5.5	0.05
5	0.04
4.5	0.03
4	0.00

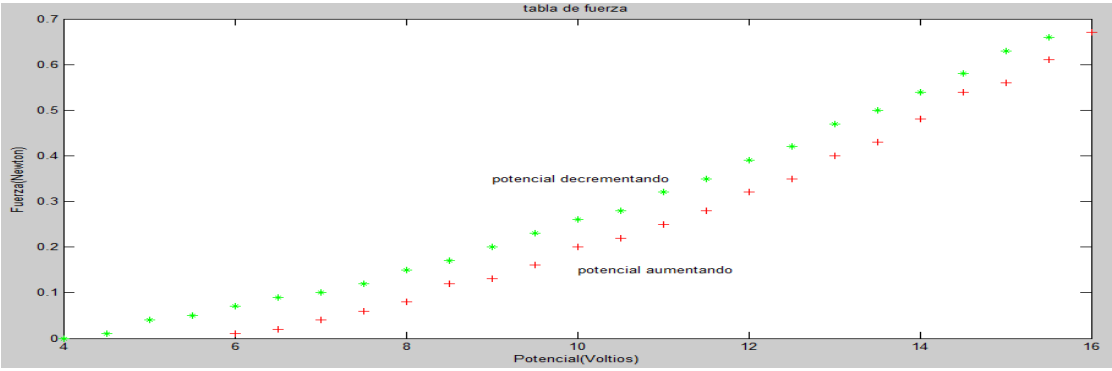


Tabla de fuerzas

7.3. Hoja de especificaciones del transistor

NPN power transistors BD135; BD137; BD139

FEATURES

High current (max. 1.5 A)

Low voltage (max. 80 V).

APPLICATIONS

Driver stages in hi-fi amplifiers and television circuits.

DESCRIPTION

NPN power transistor in a TO-126; SOT32 plastic package. PNP complements: BD136, BD138 and BD140.

PINNING

PIN	DESCRIPTION
1	emitter
2	collector, connected to metal part of mounting surface
3	base

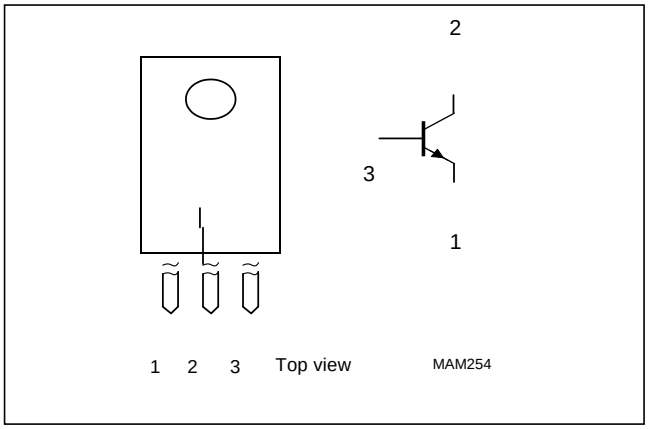


Fig.1 Simplified outline (TO-126; SOT32) and symbol

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	BD135			45	V
	BD137			60	V
	BD139			100	V
V_{CEO}	collector-emitter voltage	open base			
	BD135			45	V
	BD137			60	V
	BD139			80	V
V_{EBO}	emitter-base voltage	open collector		5	V
I_C	collector current (DC)			1.5	A
I_{CM}	peak collector current			2	A
I_{BM}	peak base current			1	A
P_{tot}	total power dissipation	T_{mb} 70 C		8	W
T_{stg}	storage temperature		65	+150	C
T_j	junction temperature			150	C
T_{amb}	operating ambient temperature		65	+150	C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient	note 1	100	K/W
$R_{th\ j\text{-}mb}$	thermal resistance from junction to mounting base		10	K/W

Note

1. Refer to TO-126; SOT32 standard mounting conditions.

CHARACTERISTICS

T_j = 25 C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{CBO}	collector cut-off current	I_E = 0; V_{CB} = 30 V			100	nA
		I_E = 0; V_{CB} = 30 V; T_j = 125 C			10	A
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = 5 V			100	nA
h_{FE}	DC current gain	V_{CE} = 2 V; (see Fig.2) I_C = 5 mA I_C = 150 mA I_C = 500 mA	 40 63 25		 250	
	DC current gain BD135-10; BD137-10; BD139-10 BD135-16; BD137-16; BD139-16	I_C = 150 mA; V_{CE} = 2 V; (see Fig.2)	 63 100		 160 250	
V_{CEsat}	collector-emitter saturation voltage	I_C = 500 mA; I_B = 50 mA			0.5	V
V_{BE}	base-emitter voltage	I_C = 500 mA; V_{CE} = 2 V			1	V
f_T	transition frequency	I_C = 50 mA; V_{CE} = 5 V; f = 100 MHz		190		MHz
$\frac{h_{FE1}}{h_{FE2}}$	DC current gain ratio of the complementary pairs	I_C = 150 mA; V_{CE} = 2 V		1.3	1.6	

NPN power transistors BD135; BD137; BD139

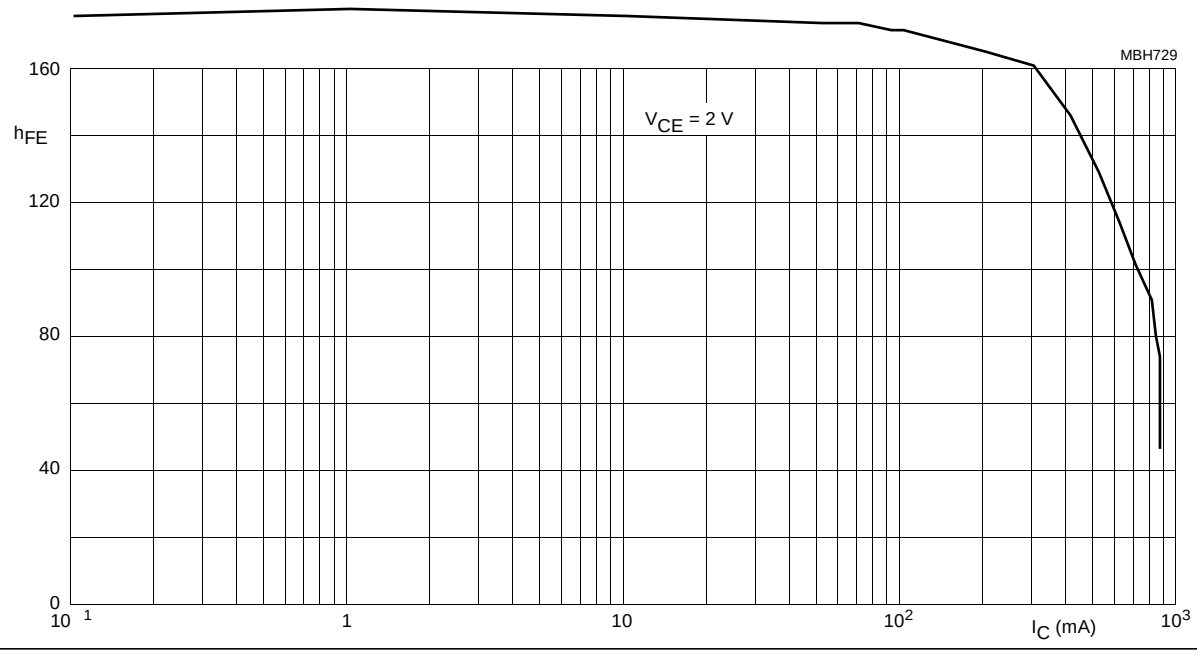
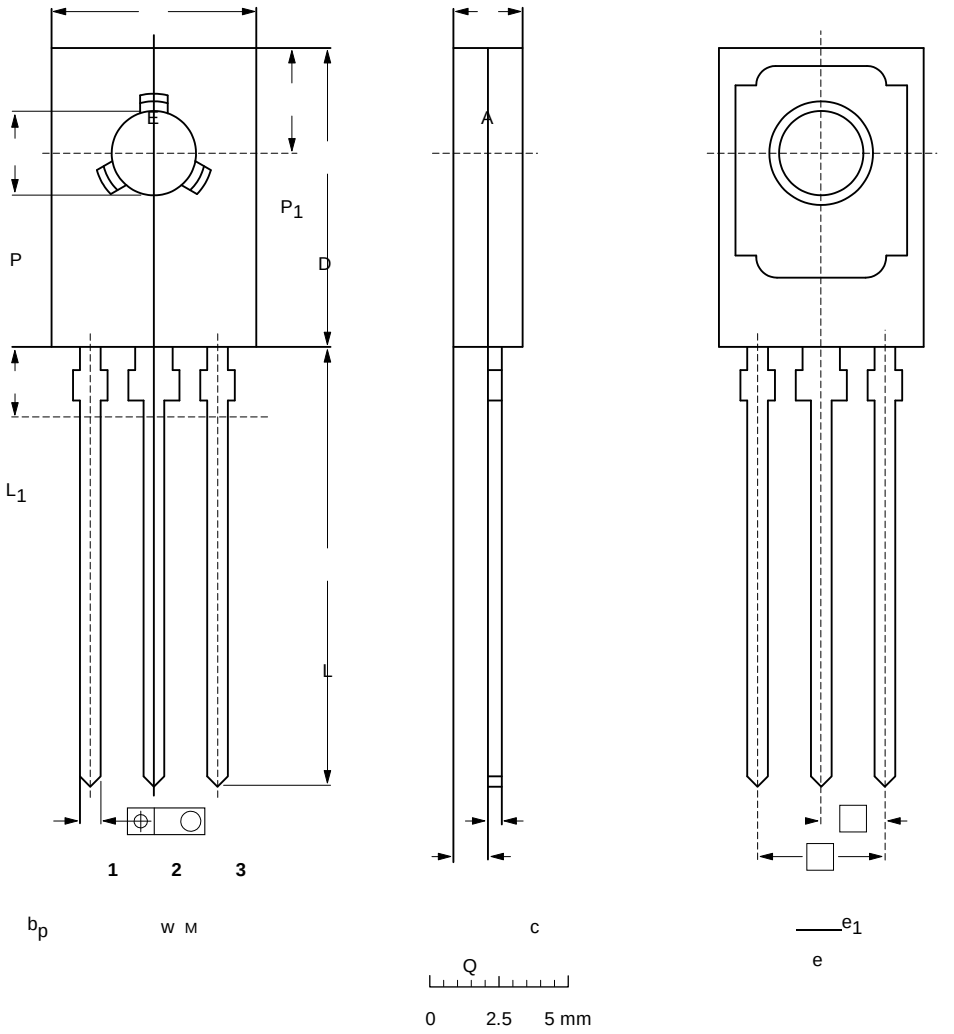


Fig.2 DC current gain; typical values.

PACKAGE OUTLINE

Plastic single-ended leaded (through hole) package; mountable to heatsink, 1 mounting hole; 3 leads SOT32



DIMENSIONS (mm are the original dimensions)

UNIT	A	b_p	c	D	E	e	e_1	L	L_1(1) max	Q	P	P_1	w
mm	2.7 2.3	0.88 0.65	0.60 0.45	11.1 10.5	7.8 7.2	4.58	2.29	16.5 15.3	2.54	1.5 0.9	3.2 3.0	3.9 3.6	0.254

Note

1. Terminal dimensions within this zone are uncontrolled to allow for flow of plastic and terminal irregularities.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT32		TO-126				97-03-04

Philips Semiconductors

Pr

oduct specification

NPN power transistors BD135; BD137; BD139

DEFINITIONS

Data Sheet Status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

8. Bibliografía

http://www.arduino.cc/es/
http://web.usal.es/~sebas/PRACTICAS/PRACTICA%207.pdf
http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/regresion1/regresion1.htm
http://wechoosethemoon.es/2011/07/21/arduino-matlab-simulink-controlador-pid/
http://www.mathworks.es/products/matlab/
https://mail-attachment.googleusercontent.com/attachment/?ui=2&ik=4cfaa18dc7&view=att&th=13c3dab98f7a96c5&attid=0.1&disp=inline&realattid=f_hbyvlfp60&safe=1&zw&saduie=AG9B_P_2nMMpbD2F9HZL5xwt9grt&sadet=1369152115334&sads=faNkrF_KM3bPnpOVotTks4oqUWo&sadssc=1
http://fritzing.org/
Ingenieria de control moderna "Katsuhiko Ogata"

1. Resumen

En el presente proyecto se pretende controlar la posición de un sistema mediante el uso de la herramienta simulink de Matlab. El sistema consta de dos barras ancladas a modo de balancín, en uno de los extremos de la barra se encuentra un motor con una hélice que será el encargado de producir la fuerza de empuje proporcional al giro de su hélice que será la responsable de los cambios de posición de nuestro sistema. Para controlar el motor y visualizar un circuito utilizaremos un montaje que incluirá una tarjeta de Arduino Uno.

2. Objetivos

- Conocer el funcionamiento de las tarjetas arduino y su entorno.
- Conocer las posibilidades de manejo de nuestro sistema mediante la herramienta simulink.
- Conocer los pasos a seguir para la instalación del software necesario para el funcionamiento del sistema en cualquier equipo.
- Obtención del modelo matemático del sistema a través de las leyes físicas y con datos extraídos de ensayos.
- Elaboración de un sistema en lazo abierto que nos permita darle diferentes entradas al sistema.
- Elaboración de un sistema en lazo cerrado con un controlador PID, que controle de una forma correcta la posición de la barra ante posibles perturbaciones ajenas al sistema.
- Identificar los parámetros de un controlador PID a través del método Ziegler-Nicholls.
- Introducción de mejoras que haga nuestro sistema más estable y fiable.

3. Funcionamiento básico.

La barra móvil, en la que uno de sus extremos hemos colocado un motor que empezara a girar cuando por el circule intensidad, ejercerá una fuerza, que según sea mayor o menor que la anterior podrá desplazar esta en un sentido u otro. La intensidad que circula por el motor estará controlada por un circuito eléctrico en el que tendremos control directo desde nuestra CPU gracias a su conexión USB con la placa Arduino. Desde el ordenador podremos controlar la diferencia de potencial que se le aplica al motor, como es el caso del circuito en lazo abierto, o la posición que queremos que ocupe nuestra barra móvil cuando estemos trabajando en lazo cerrado.

La posición se conocerá gracias a un potenciómetro alimentado desde la placa arduino y a os datos que entregue a una de las entradas analógicas. Interpretando estos datos correctamente podremos conocer la posición de la barra móvil a través la pantalla del ordenador.

4. Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Estos prototipos son un conjunto de placas, que nos permitirá tomar información de nuestro entorno a través de sus pines de entrada y

ejecutar órdenes a través de sus pines de salidas. Todas ellas poseen un microcontrolador programable desde el lenguaje de programación de Arduino (wiring). Tiene la posibilidad de ejecutar sus proyectos sin la necesidad de estar de estar conectados a un ordenador, en estos casos tendrá que contar con una alimentación externa. Otra ventaja es su comunicación con diferentes tipos de software. Todo ello convierte a Arduino en una herramienta muy útil para principiantes aficionados y personal en formación.

Posee una gran variedad de placas para satisfacer nuestras diversas necesidades. En nuestro caso trabajaremos con la placa Arduino UNO

Matlab-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.
Tiene una conexión directa con Matlab, pudiendo exportar los resultados obtenidos en simulink para hacer análisis más exhaustivos y poder obtener nuevos resultados. También nos dan la posibilidad de incorporar algoritmos propios de Matlab. Además ofrece la posibilidad de conectar el modelo con hardware para comprobar en tiempo real y de una manera física el funcionamiento de este.

Simulink es uno de los diferentes tipos de software que puede establecer comunicación con Arduino.

Comunicación Arduino-Simulink

Simulink tiene una biblioteca para trabajar con arduino (ArduinoIO). Para que Arduino pueda trabajar con simulink primero le tendremos que cargar un programa predefinido y que nos facilita Matlab, que configurara sus pines para que pueda establecer conexión con los diferentes bloques de la biblioteca ArduinoIO. La conexión de Arduino con el ordenador será USB.

5. Sistema físico.

El movimiento que realiza la barra móvil es el de rotación de un cuerpo rígido alrededor de un eje fijo. En estos movimiento lo que acelera al cuerpo angularmente es el torque y no solo la fuerza que se le aplica. El torque relaciona la magnitud de la fuerza con el punto y dirección de aplicación de esta, y se define como:

$$\tau = \vec{r} \times F$$

Donde F es la fuerza aplicada y *"r"* es el vector de posición de la fuerza aplicada que va desde el eje de giro hasta el punto de aplicación de esta. El torque es el producto vectorial de estos dos términos.

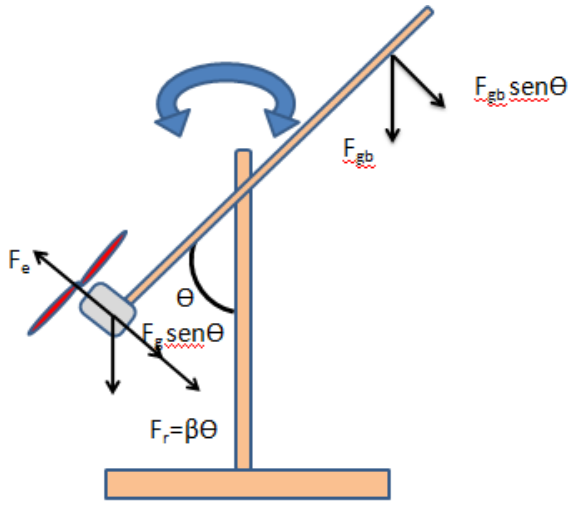
El torque está relacionado con el momento de inercia, otro término importante en los movimientos rotacionales ya que todo elemento que gira en torno a un eje posee momento de inercia. Se define como la resistencia de un cuerpo a obtener una aceleración angular.

De la relación de estos dos términos obtenemos la ecuación ideal de nuestro sistema.

$$\sum \tau = I \propto$$

De la que obtendremos la ecuación final incorporando el rozamiento.

$$\sum \tau \ - B\dot{\theta} = I \propto$$



Calculamos los torques de las diferentes fuerzas del sistema (fuerza del motor y la fuerza de la gravedad que actúa sobre el motor y la barra), representadas de una manera esquemática en la figura de la izquierda y los momentos de inercia de los elementos que están en rotación (la barra móvil y el motor más la hélice). Una vez obtenidos los sustituimos en la ecuación anterior y ya tenemos la ecuación de nuestro sistema:

$$0.1335sen\theta(t) - 0.1825sen\theta(t) + 0.245F_g - B\frac{d\theta}{dt} = 0.01443\frac{d^2\theta}{dt}$$

$$0.245F_g = 0.049sen\theta(t) + B\dot{\theta} + 0.01443\ddot{\theta}$$

Función de transferencia

Al encontrar derivadas en nuestra ecuación utilizaremos la transformada de Laplace para poder obtener la función de transferencia $\frac{\theta}{F_e}$. Como la ecuación no es lineal, la linealizamos utilizando las series de Taylor en cuanto a un punto de equilibrio

$(\dot{\theta}_0 = \ddot{\theta}_0 = 0, \theta = 45º, F_g$=0.049sen(45)/0.245=0.141N)

La ecuación linealizada quedara:

$$0.245\Delta F_g = 0.01443\Delta\ddot{\theta} + B\Delta\dot{\theta} + 0.0346\Delta\theta$$

Ahora ya podemos aplicar la transformada de La place:

$$0.245\Delta F_g(s) = 0.01443s^2\Delta\theta(s) + Bs\Delta\theta(s) + 0.0346\Delta\theta(s)$$

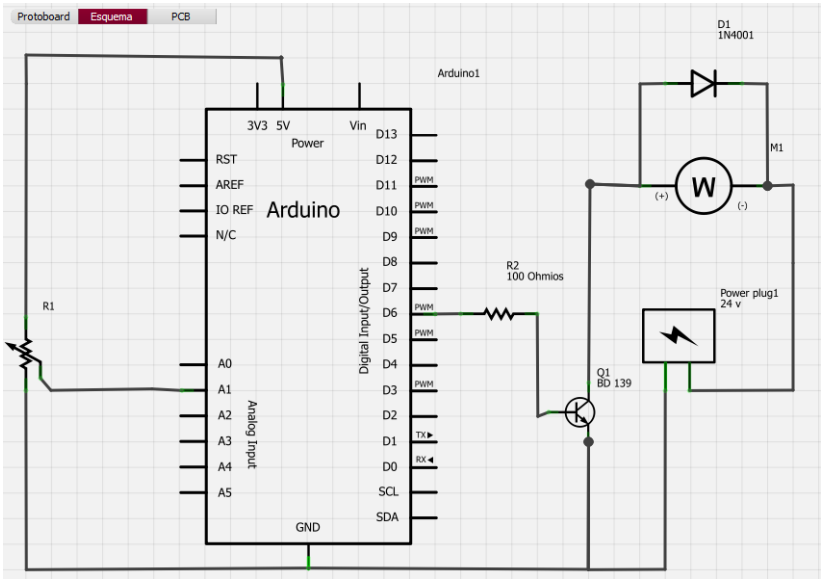
Por lo tanto la función de transferencia de nuestro sistema suponiendo condiciones iniciales nulas y tomando ΔF=0 será:

$$\frac{\Delta\theta(s)}{\Delta F_g(s)} = \frac{0.245}{0.01443s^2 + Bs + 0.0346}$$

El coeficiente de rozamiento B cambiara de un dispositivo a otro, ya que el rozamiento de cada uno es distinto.

6. Circuito eléctrico.

El circuito eléctrico de nuestro sistema es el siguiente:



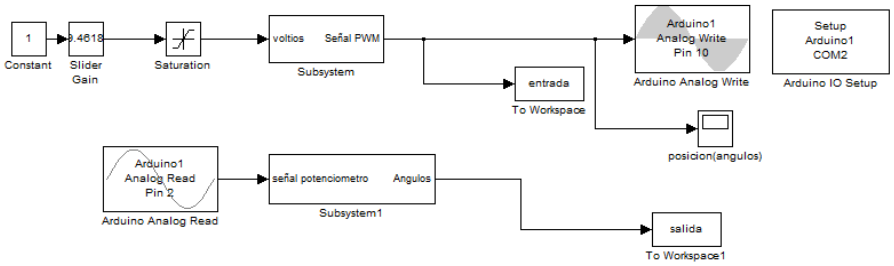
En la parte central se encuentra la placa de arduino que ya se ha comentado anteriormente. El potenciómetro alimentado desde Arduino nos servirá para poder leer la posición de la barra móvil gracias a su funcionamiento como divisor de tensión.

En la parte derecha se encuentra la parte responsable de suministrar corriente al motor. Como es obvio arduino no será capaz de entregarle los valores de tensión e intensidad necesarios al motor por lo tanto disponemos de una fuente de tensión de 24v. El control de intensidad que llega al motor se hace desde el transistor y la intensidad de base que se le suministra y que dependerá del nivel de tensión que nos encontremos en la salida PWM del Arduino. El motor tendrá en antiparalelo un diodo, para que cuando detenga su funcionamiento las corrientes de fuga que genera se disipen a través de él y no dañen al transistor.

7. Implementación

7.1. Lazo abierto

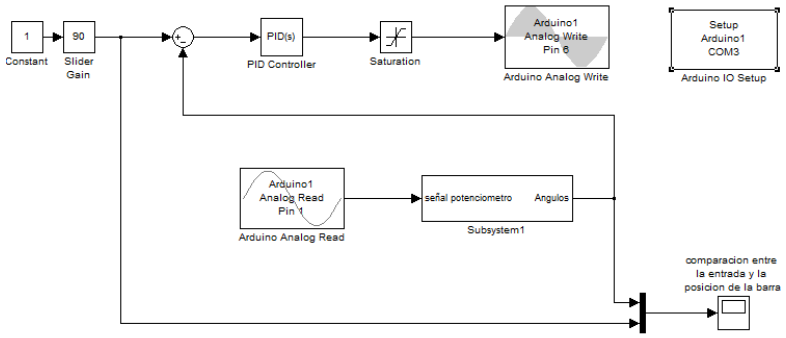
El funcionamiento en lazo abierto será muy básico. Mediante nuestro ordenador seleccionaremos la diferencia de potencial que queremos tener entre los bornes de nuestro motor y conoceremos la posición de la barra mediante el potenciómetro como hemos explicado anteriormente. El circuito en simulink quedara de la siguiente forma:



La salida no dependerá exclusivamente de la entrada pudiendo adoptar esta diferentes valores para la misma entrada.

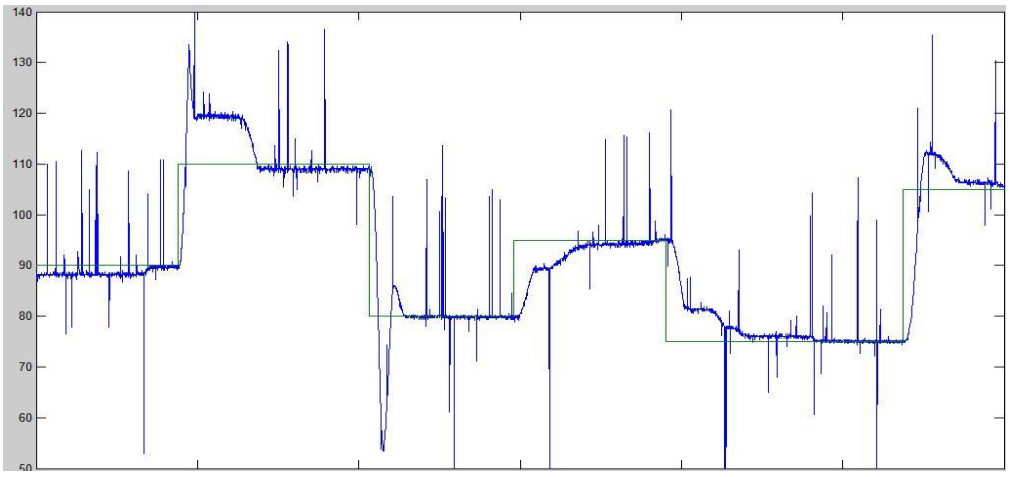
7.2. Lazo cerrado

En este caso nosotros introduciremos el ángulo que queremos que adopte nuestra barra, el cual adoptara en cierto periodo de tiempo, siendo capaz de recuperarlo en el caso de que se produzcan perturbaciones. El circuito es el siguiente:



Para el control utilizamos un controlador PI, con valores P = 2,3 I = 0,91. Estos han sido obtenidos mediante el método de Ziegler-Nicholls que se aplica en circuitos de lazo cerrado y un posterior ajuste mediante prueba-error.

En la siguiente gráfica se representa la respuesta de nuestro sistema:



8. Conclusiones

Los resultados conseguidos con estos proyectos han sido satisfactorios. Se han cumplido los objetivos que nos marcados al principio del inicio de este, obteniendo un correcto funcionamiento y control así como un completo conocimiento de todo lo que nos ha permitido llegar a ello.

Con la realización de este proyecto no ha permitido poner en práctica y desarrollar los conocimientos anteriormente adquiridos en nuestra formación, en especial los relacionados con la automática, control y electrónica en general. Esto nos ha permitido llevar a nuestro proyecto hacia una solución que podemos considerar óptima.

Arduino es uno de los grandes aportes que hemos descubierto con la realización de este proyecto, aunque no hayamos hecho un amplio uso de posibilidades nos ha permitido ser conscientes de todo su potencial a la hora de hacer cualquier tipo de proyecto relacionado con la electrónica. Una de sus grandes ventajas es su versatilidad ya que puedes trabajar con él desde diferentes software en nuestro caso hemos utilizado arduino vía Matlab-Simulink, este software nos ha permitido una poder trabajar con datos de una manera sencilla e intuitiva ya que hemos tenido un control directo del sistema desde la CPU.

Con la realización de este proyecto hemos aprendido a adaptar nuestro ordenador para trabajar con las diferentes placas de arduino desde la herramienta simulink de Matlab, lo cual conlleva la instalación de librerías en Matlab.

Nuestro sistema describe un movimiento circular, el trabajo con el nos ha permitido conocer sus características en las que destacamos: conceptos como momento de inercia y torque. Estos conocimientos junto a la 2º ley de Newton nos permitido identificar el sistema así como sacar la función de transferencia.

En referencia al circuito eléctrico lo más destacable ha sido conocer el uso del potenciómetro como sensor de posición angular y el uso necesario del diodo en antiparalelo con el motor, como elemento protector del circuito.

Hemos realizado un control en lazo cerrado de nuestro sistema con un controlador PID, con la ayuda del método de Ziegler-Nicholls y su posterior ajuste hemos identificado los parámetros del controlador PID, que hacen que nuestro circuito funciona de una manera correcta. Esto no ha permitido llegar a la conclusión que el controlador PID es idóneo para nuestro sistema y que a través del método de Ziegler-Nicholls nos da una idea aproximada para la identificación de sus parámetros.

Todo ello nos ha permitido cumplir nuestro objetivo principal que era el de control de posición de un balancín

Este proyecto tiene amplio uso didáctico, ya que su uso sencillo permitirá a los futuros alumnos trabajar con conceptos como lazo abierto, lazo cerrado, función de transferencia, controlador PID… viendo reflejado sus resultados en un medio físico.