UPM

CONTROL BALANCIN CON ARDUINO

Regulación automática

escuela técnica superior de ingeniería y diseño industrial

- Enrique Mateos Campo

Email: e.mateos@alumnos.upm.es

CONTENIDO

1. INTRODUCCION:

El objetivo de este proyecto es realizar una maqueta balancín-turbina, en la cual debemos poder controlar el ángulo de inclinación del brazo a través del empuje que nos aportara la turbina.

El control del mismo debe realizarse a través de la plataforma Matlab-Simulink y el hardware e IDE de Arduino.

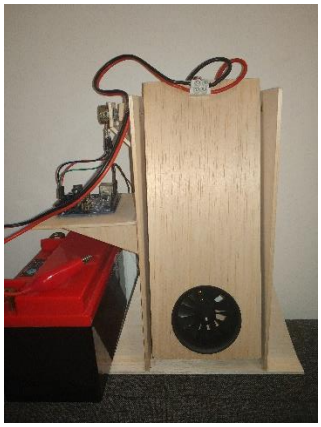
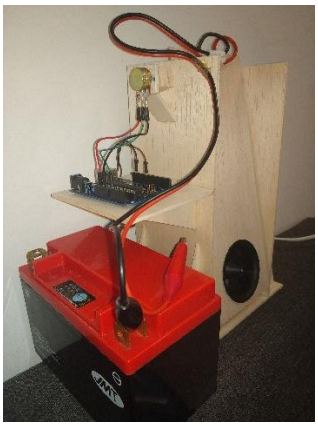
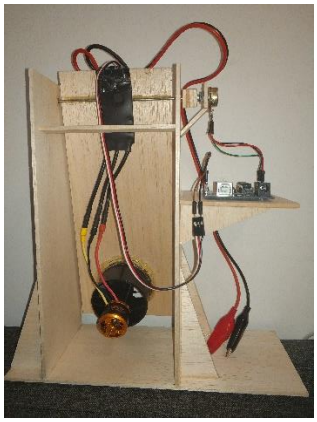
2. MATERIALES Y DESCRIPCION DE LA MAQUETA:

Los **materiales** empleados en la construcción de la misma son:

- Madera de balsa de 3 mm de espesor (estructura).
- Varilla de acero inoxidable de 2 mm de diámetro (eje).
- Potenciómetro de 10k (sensor).
- Motor sin escobillas de 12 aspas 5000k (CW) (turbina).
- Variador 50A.
- Rodamientos MR52ZZ 2x5x2.5mm
- Batería 12V de litio.
- Arduino Duemilanove.

Para la estructura de la maqueta, se ha empleado madera de balsa por su gran maleabilidad, y se le han añadido escuadras de este mismo material para mejorar la rigidez estructural. El eje del balancín es de acero y está apoyado sobre 2 rodamientos, uno en cada extremo. El potenciómetro será utilizado como sensor de giro, y está acoplado al eje a través de una pieza en madera que los mantiene concéntricos y solidarios.

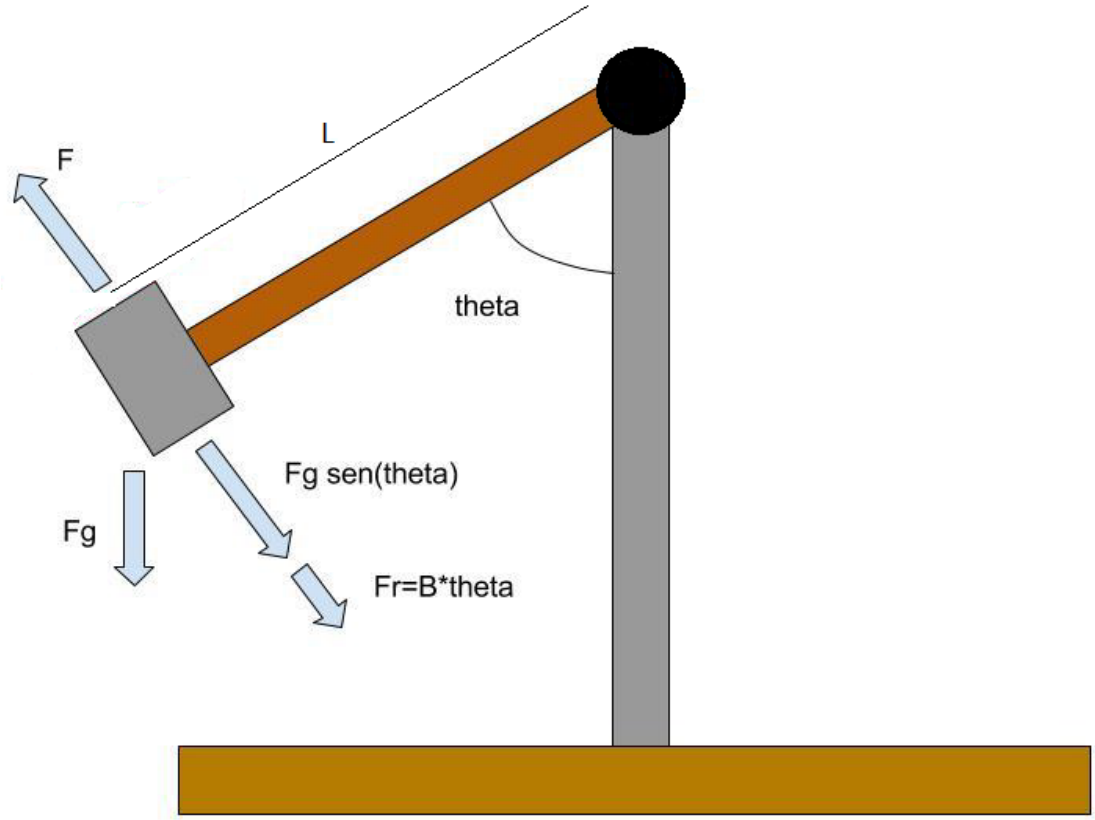
En la misma estructura se ha diseñado un espacio donde poder colocar la batería, la cual además le dará estabilidad a la maqueta, y superior a este un soporte para la placa de Arduino. En uno de los extremos del panel de madera que conforma el brazo, está posicionada la turbina, y en la cara inferior de esta misma plancha está fijado el variador, a través del cual manejaremos la potencia del motor.

3. METODOLOGIA:

3.1 MODELADO DEL SISTEMA EN ECUACIONES ALGEBRO-DIFERENCIALES

En la siguiente imagen, podemos ver una aproximación de la planta, en la cual se representan todas las fuerzas que actúan sobre el sistema, y que nos serán de interés para obtener el modelo algebro-diferencial del mismo.



El modelo matemático aproximado del sistema será el siguiente:

- La tensión aplicada del motor será absorbida por la resistencia eléctrica (R), la bobina (L) y la tensión contraelectromotriz (Ufem) del mismo:

$$U_e(t) = R * i(t) + L * \dot{I}(t) + U_{fem}(t)$$

- La tensión de la fuerza contraelectromotriz se define como una constante asociada al diseño del motor por su velocidad de giro:

$$U_{fem}(t) = K_v * W_m(t)$$

- El par que el motor genera en la hélice de la turbina será una constante también asociada a su diseño por la corriente que circula por el motor:

$$T_m(t) = K_i * i(t)$$

- Este par generado por el motor, será absorbido por el momento de inercia (Jm) y el rozamiento viscoso (Bm) de su propio eje, más el par resistente que produce el rozamiento del aire con las palas de la turbina (Tr) :

$$T_m(t) = J_m * \dot{W}_m(t) + B_m * W_m(t) + T_r(t)$$

- La fuerza de empuje generada, podemos aproximarla por la siguiente ecuación:

$$F_e(t) = K_1 * W_m^2 \quad (t)$$

- Por último, la ecuación que nos relaciona el empuje generado, con el peso y el resto de consumo de fuerzas será:

$$L * F_e(t) - M_b * g * L * sen\theta(t) = M_b * L^2 * \ddot{\theta}(t) + B_e * \dot{\theta}(t)$$

En esta última ecuación, ya que el motor es el elemento mas pesado con diferencia, se ha aproximado toda la masa del brazo en el eje del motor, justo donde también se produce la fuerza de empuje. Be es el rozamiento viscoso del eje del brazo (dado por los rodamientos y el giro del potenciómetro), θ es el giro del brazo y Mb es la masa del brazo.

3.2 MODELADO DEL SISTEMA EN MATLAB

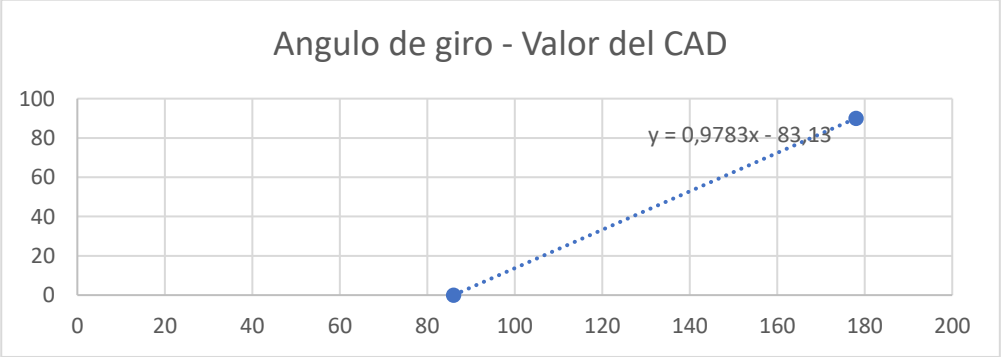
Obtención FDT del sistema.

Para Obtener la FDT del sistema, hemos utilizado una estrategia experimental que consiste en obtener la respuesta del mismo ante una entrada escalón, y posteriormente trasladar estos datos a Matlab para a través de sus herramientas obtener la FDT.

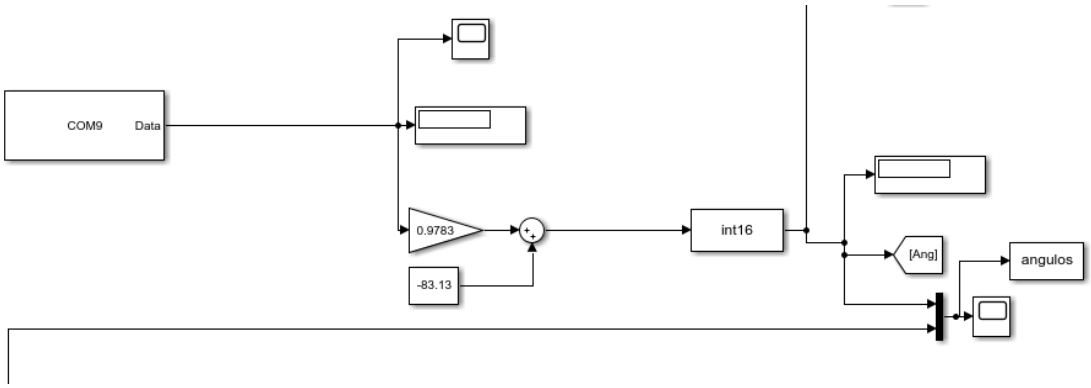
El procedimiento seguido para esto ha sido el siguiente:

- En primer lugar, la intención de este proyecto es que la consigna que le daremos al control este directamente en ángulos, por lo que debemos obtener su relación de conversión. El sensor que nos facilitara esta información es el potenciómetro, el cual según varie el ángulo cambiara la tensión que traslada al CAD del Arduino, este CAD es de 10 bits y transformara cada valor de tensión recibida en un numero comprendido entre 0 y 1023. Esto significa que debemos obtener la curva que representa la relación entre el Angulo de giro y el valor obtenido por el CAD.

  Para ello he obtenido experimentalmente dos valores, el valor de CAD correspondiente a 90º y el correspondiente a 0º y a través de la aplicación Excel, he obtenido la ecuación que define esta recta:

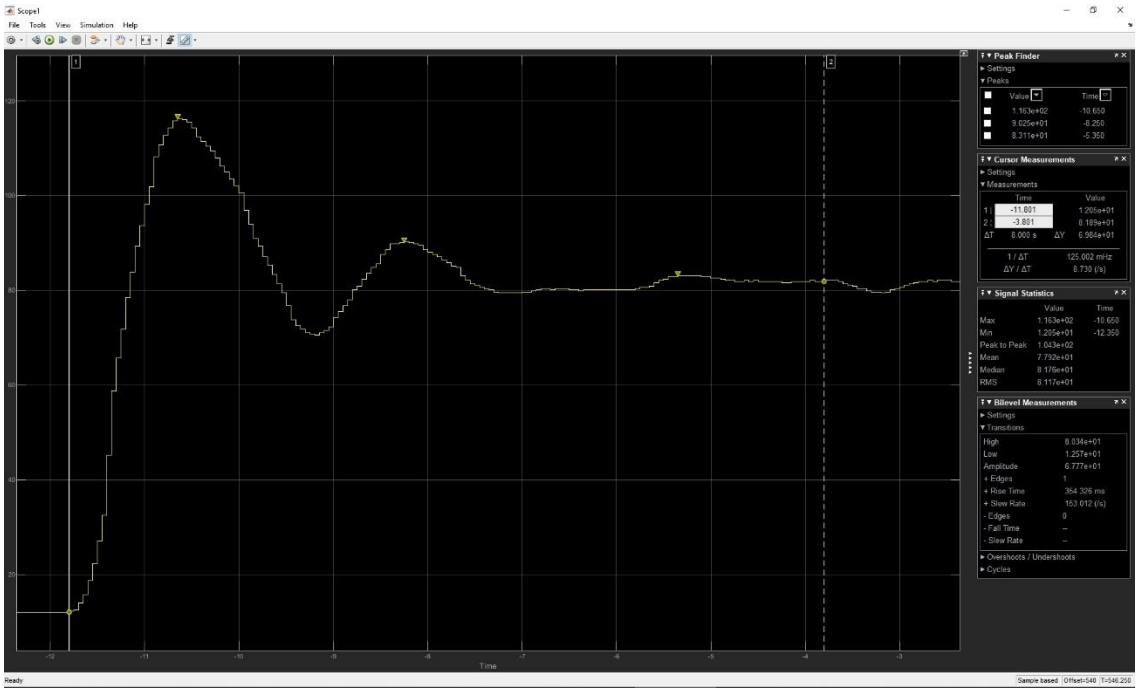


Posteriormente lo he implementado en simulink con el siguiente diagrama de bloques, y de esta forma ya podemos estimular la maqueta con la entrada escalón y estudiar su respuesta.

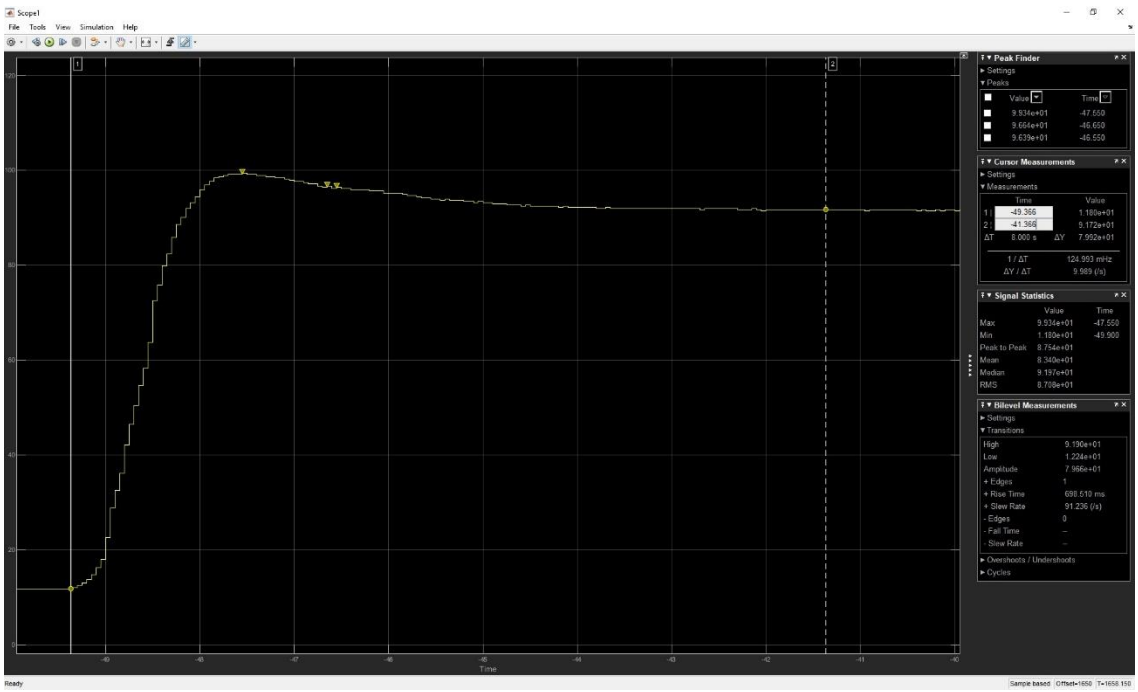


- Realizando distintas pruebas, observe que debía de calibrar el variador que controla el motor de la turbina. Sin pasar a detallar dicho procedimiento, ya que no forma parte del objeto de esta memoria, mostrare el cambio en la respuesta que obtuve.

  Antes de la calibración, con una gran sobre oscilación:

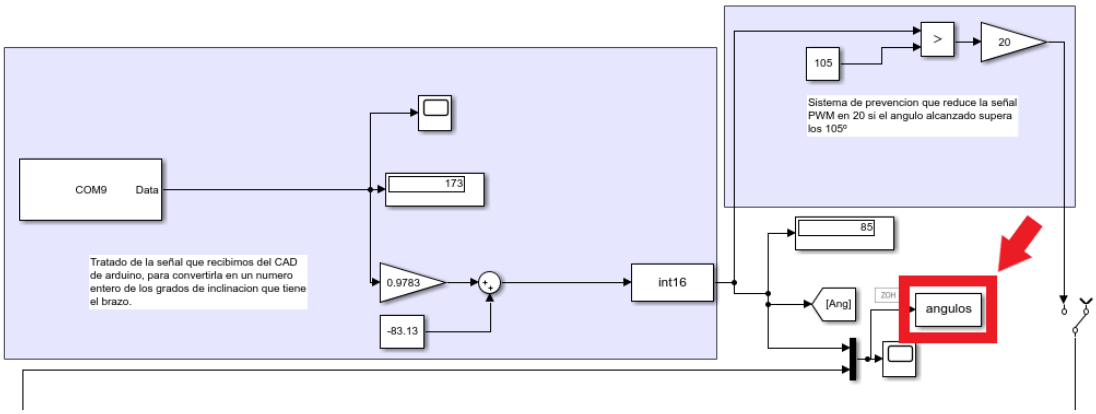


  Después de la calibración (Menos sobre oscilación y más suave):

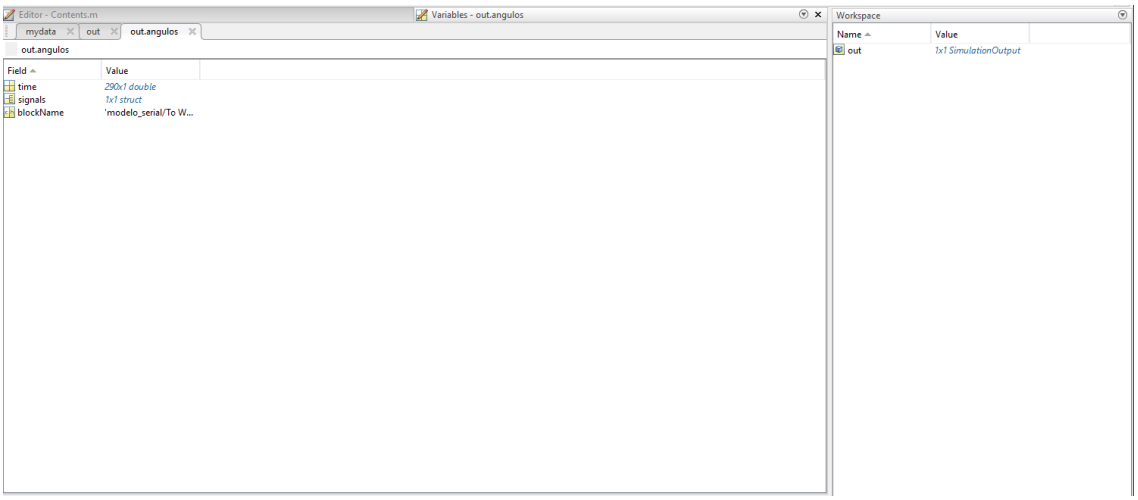


En la primera gráfica, como se pueda apreciar la consigna estaba en 80º, ya que sino la sobre oscilación llevaba al sistema a un punto de inestabilidad, provocando la rotura de la fijación del potenciómetro con el eje.

- En este punto ya podemos capturar la respuesta del sistema a través del valor que obtenemos en “ángulos” y el traslado de estos datos al workspace de Matlab.



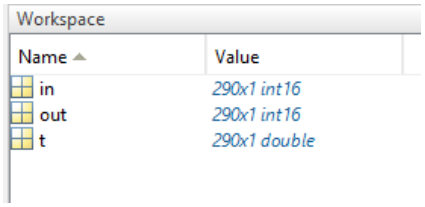
Después de excitar el sistema con al entrada escalón, vamos a la ventana de Matlab y encontraremos que los datos han sido enviados a nuestro workspace, como muestra la siguiente imagen:



Una vez aquí, debemos preparar estos datos para el posterior uso que haremos de ellos al identificar el sistema. Para ello lanzaremos los siguientes comandos en la ventana de comandos de Matlab:

```
t=out.angulos.time;

in=out.angulos.signals.values(:,2);

out=out.angulos.signals.values(:,1);
```

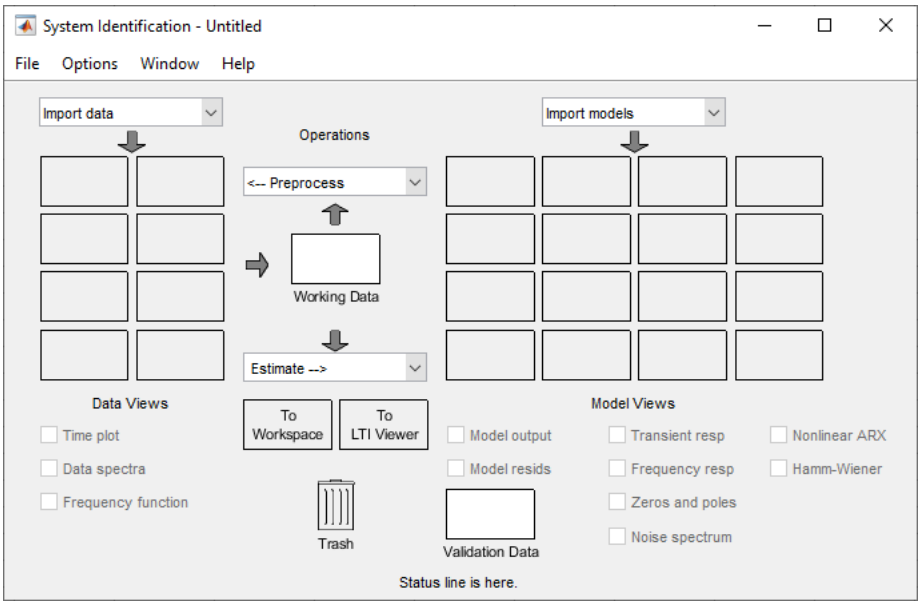
Con esto creamos 3 variables, las cuales han obtenido los valores de los datos capturados, siendo t el tiempo transcurrido, in el valor de la consigna, y out el valor que representa la lectura de los ángulos del brazo de la maqueta.



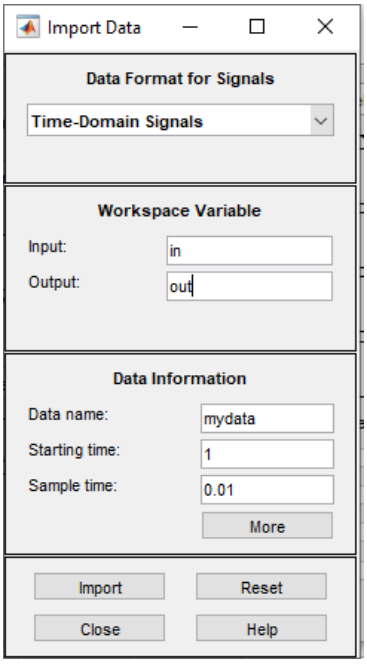
En este punto ya podemos lanzar la herramienta de identificación de sistema, que lo haremos con el siguiente comando en la ventana de comandos:

systemIdentification;

Se nos abrirá la siguiente ventana:

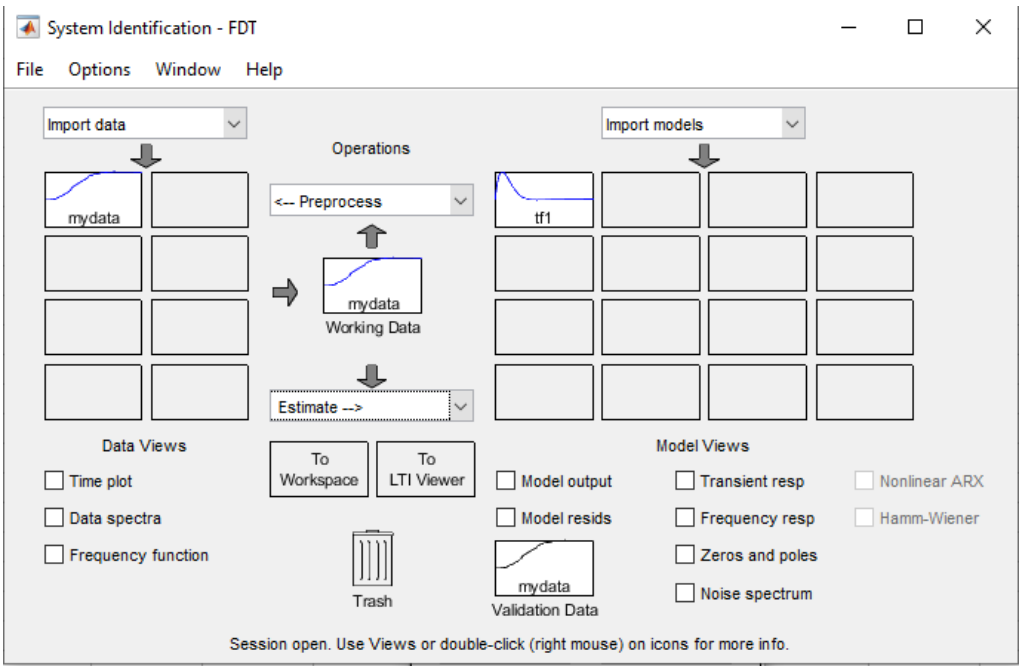


Haremos clic en “Impor data” y posteriormente en “Time domain data”, obteniendo la siguiente ventana:

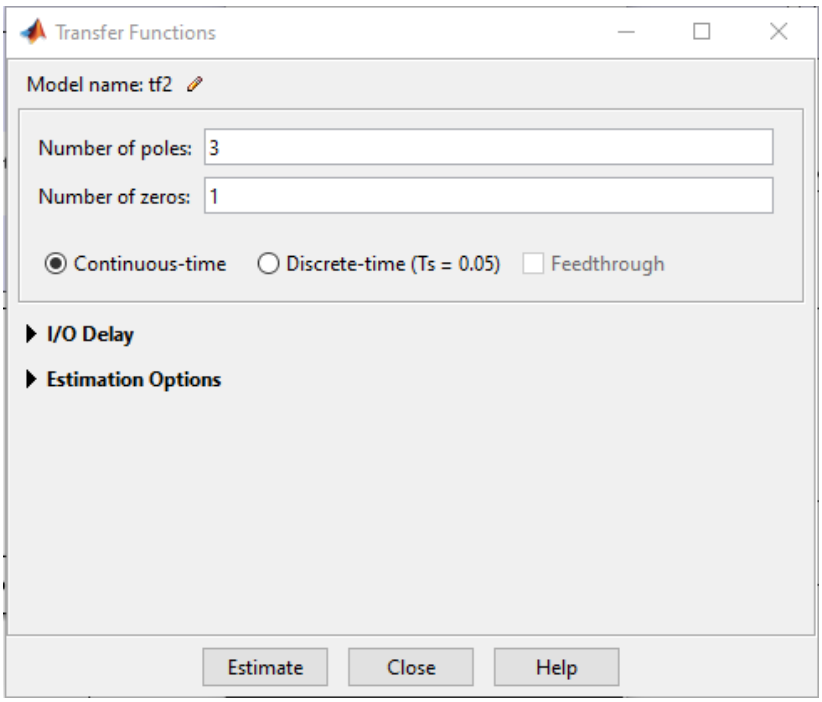


Definimos las variables tal como se muestra en la figura anterior, con un tiempo de muestreo de 0.01, el empleado en simulink .

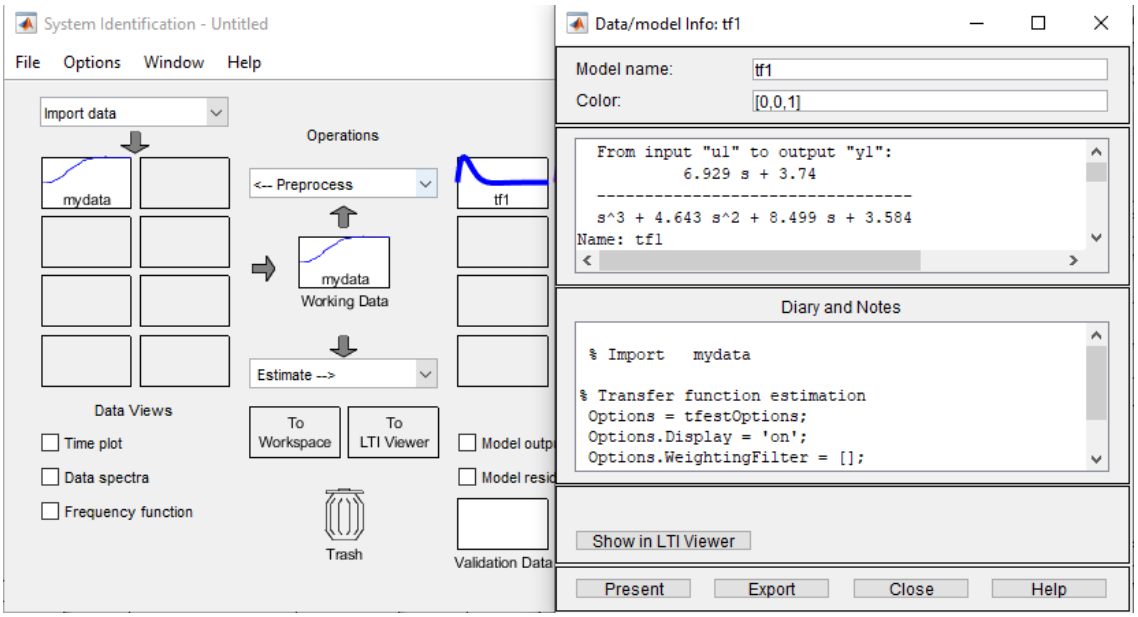
Hacemos click en import, y ya tendremos los datos en la herramienta, que nos permitirán estimar la FDT que buscamos, para ello debemos hacer clic en "estimate -→ Transfer Function Models", donde Matlab nos estimara la función de transferencia de la planta.



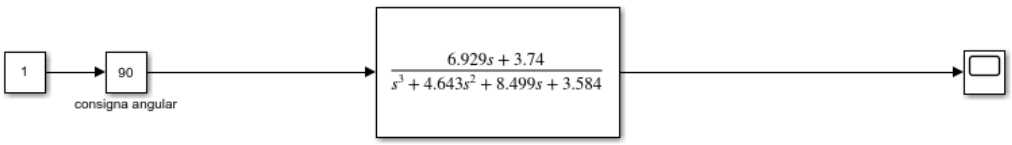
Se nos abrirá la siguiente ventana, y después de varias pruebas, en mi caso la FDT más precisa la obtuve con 3 polos y un cero.



Pulsamos estimar y obtenemos la FDT:

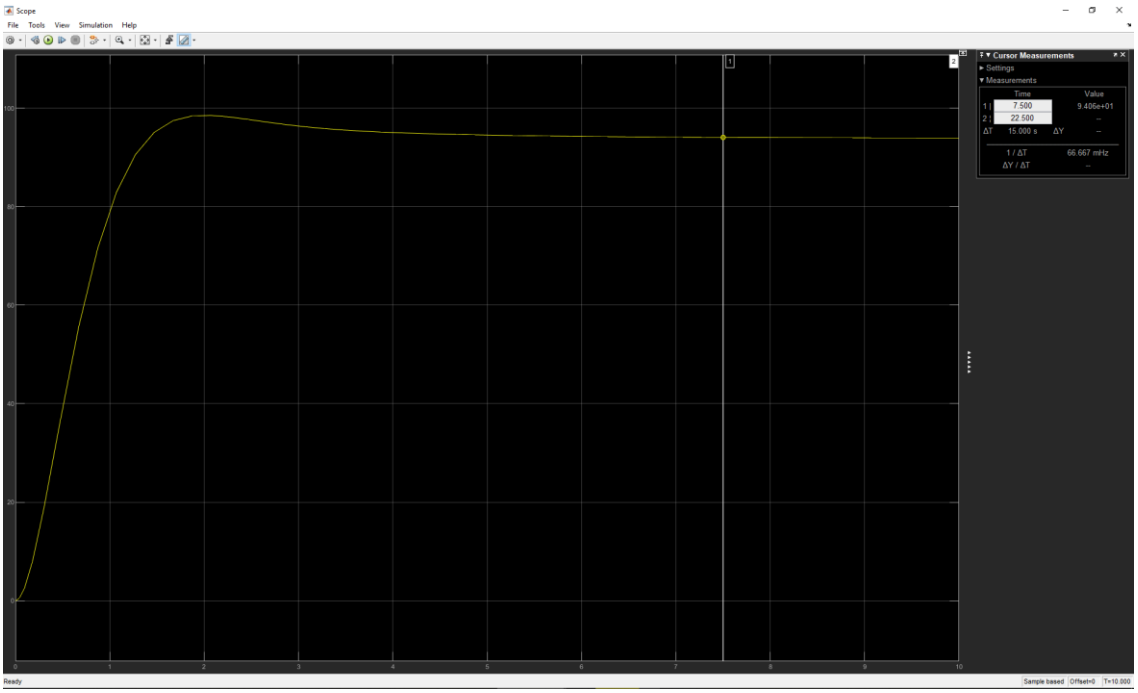


- En este punto, debemos comprobar si la FDT obtenida representa lo suficientemente bien al sistema real, y para ello podemos comparar su respuesta ante una entrada escalón, al igual que hicimos con la maqueta, y comparar dichas respuestas.
  Esto lo haremos con el siguiente diagrama de bloques:

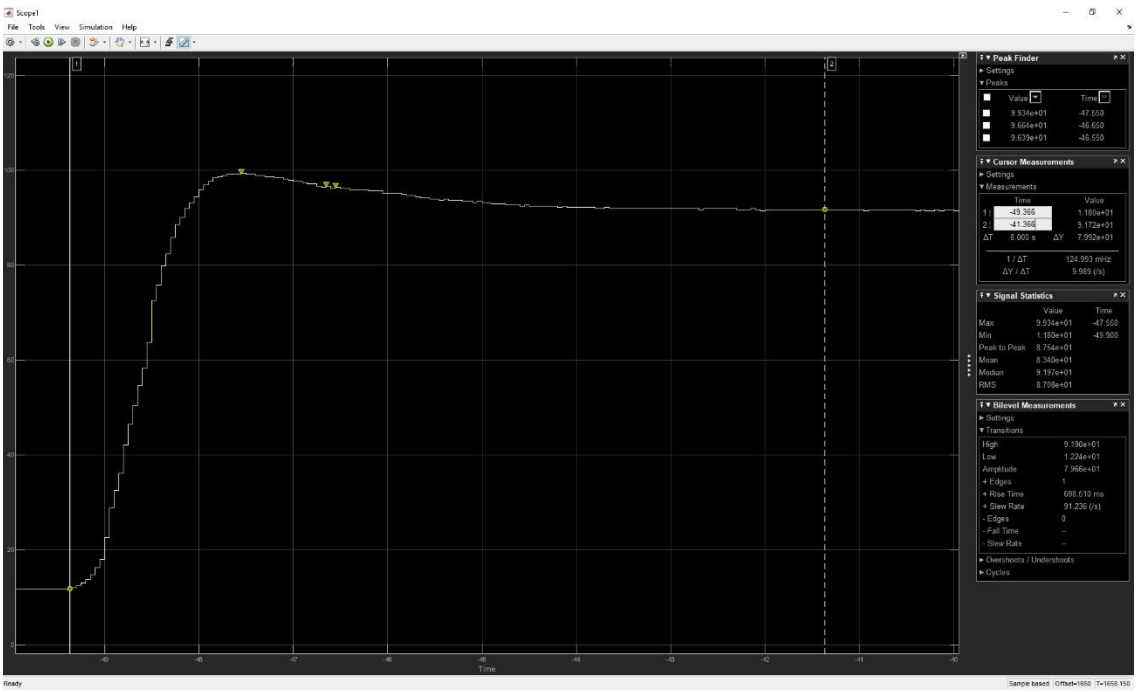


Obteniendo la siguiente grafica en el scope(1.), que si la compramos con la respuesta real del sistema(2.), podemos afirmar que efectivamente la FDT obtenida representa lo suficientemente bien nuestro sistema real:

1. Respuesta FDT obtenida



2.Respuesta real del sistema

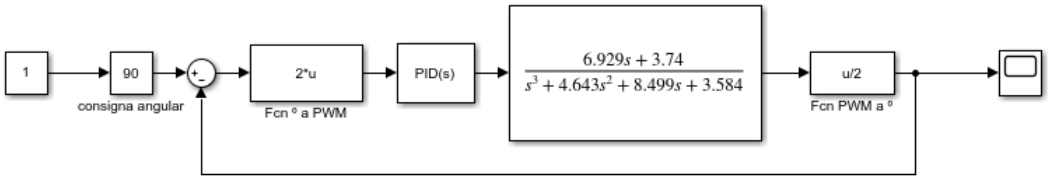


3.3 CONTROL DEL SISTEMA

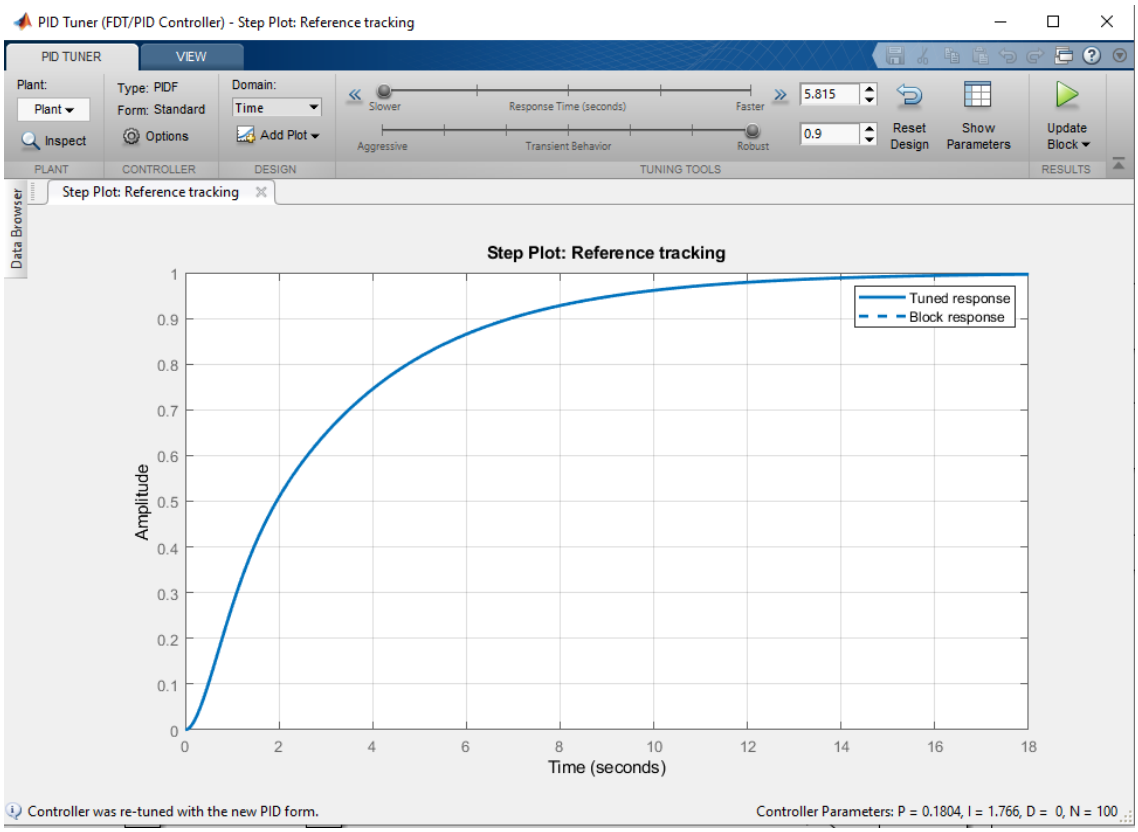
Modelado en Simulink.

Lo primero que debemos hacer, es obtener los valores óptimos para el regulador PID que utilizaremos en el control de la planta. Para ello, Simulink nos facilita una herramienta llamada tune, en donde podemos obtener los valores del regulador fácilmente indicando el tipo de respuesta que queremos para el sistema.

Para ello, realizamos un nuevo modelado más simple, como se muestra en el siguiente diagrama de bloques:



Haciendo doble click en el bloque PID, se nos abrirán las opciones del mismo, donde podemos acceder a Tune, herramienta que nos facilita mucho el diseño de este regulador:

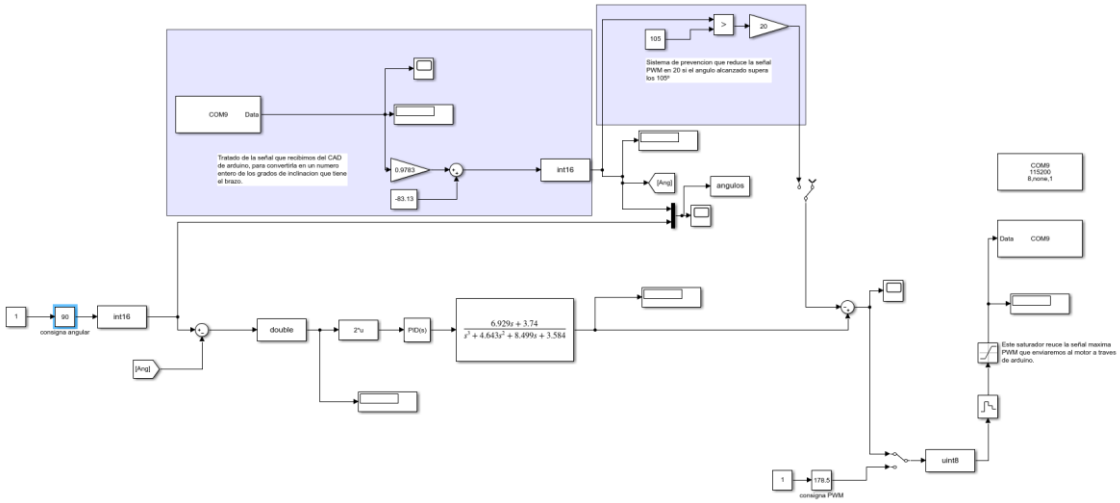


Debido a una serie de problemas con los que me he ido encontrando en la realización de este proyecto, los cuales describiré al final de esta memoria, debí buscar el regulador más lento y robusto con el fin de evitar mucha oscilación en el régimen permanente, y disminuir la posibilidad de que el sistema alcanzase puntos de inestabilidad.

Los valores obtenidos para nuestro regulador serán:

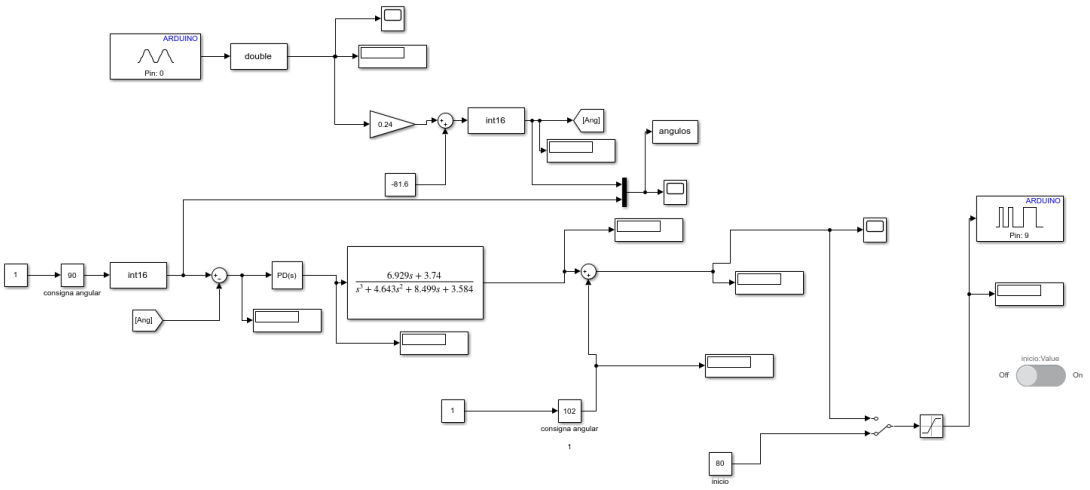
```
Controller Parameters: P = 0.1804, I = 1.766, D =  0, N = 100
```

Y este es el diagrama de bloques completo con el que el sistema será controlado:



Programa de control.

De forma experimental, descubrí que la utilización de la librería servo y el tratado de la señal a través de la función “writeMicroseconds” de Arduino, obtenía una mayor resolución en la señal que se envía al variador del motor, consiguiendo que el empuje obtenido por la turbina no fuera a grandes saltos, como si me pasaba en el primer modelo de simulink que diseñe, y que muestro a continuación:



En el cual, como se puede observar, utilizaba la propia librería de simulink para de forma transparente procesar la señal final que el variador recibiría. En este modelo, tal como comenté anteriormente, la respuesta del empuje generado en la turbina era muy escalonada, haciéndose imposible alcanzar limpiamente las consignas perseguidas, porque fácilmente se daba la casuística de que el valor previo PWM no la alcanzaba, y el inmediatamente posterior a este la sobrepasaba.

Para poder implementar el modelo finalmente utilizado, fue necesario programar Arduino de forma independiente, de modo que pudiera comunicarse por el puerto de comunicaciones con nuestro modelo de simulink, y de esta forma poder mejorar algo el desempeño del control de la maqueta procesando la señal enviada al variador a través de las funciones map() y writeMicroseconds().

El código que se ha cargado en Arduino es el siguiente, el cual para mayor claridad he comentado explicando la función de cada línea:

control_variador_skywalker_regulador_2 §

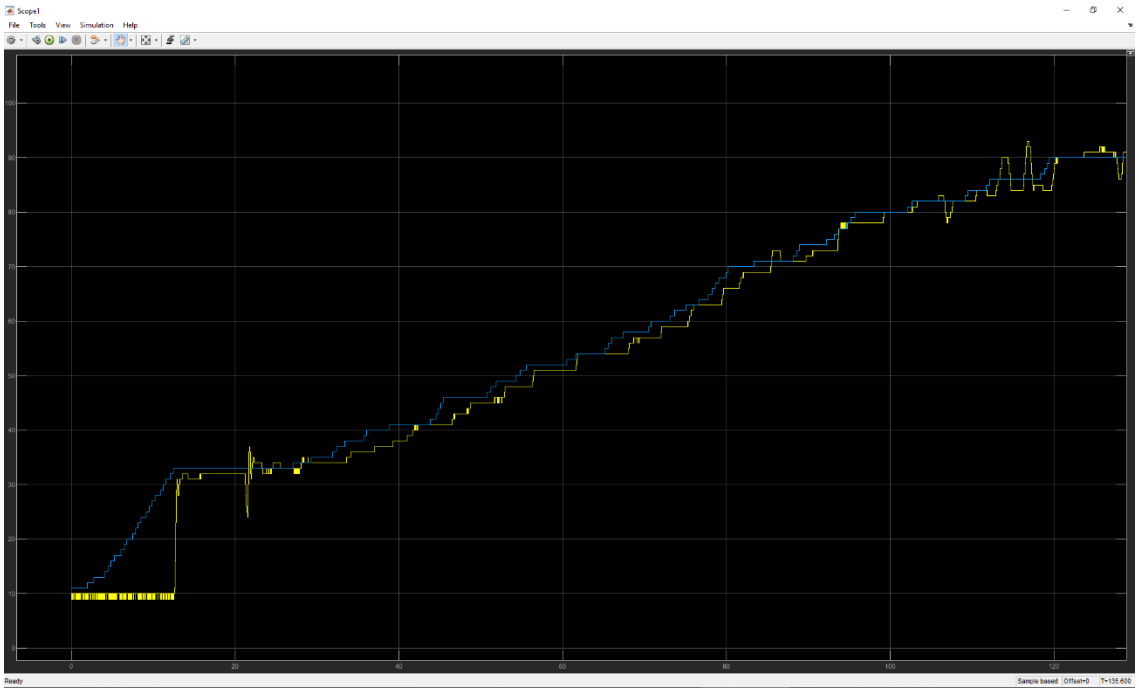
```
#include <Servo.h>

Servo var;

int Outvar=0;
int OutvarM=0;
int InputM=0;
int Input = 0;


void setup()
{
Serial.begin(115200);
var.attach(9); //pin de arduino que inyectara la señal PWM al variador
var.writeMicroseconds(630);//se inicializa el variador, en caso de conectar arduino despues de tener ya alimentado el variador
delay(3000);//le damos un tiempo prudente para la inicializacion
}

void loop()
{
Input = analogRead(A0);//realizamos la lectura analogica en el pin A0
InputM=map(Input, 0, 1023, 0, 255);//convertimos la señal obtenida del CAD de 10 bits a una señal de 8 bits
Serial.write(InputM);//Enviamos la lecuta del potencimetro a Simulink.

//Leemos el puerto para obtener los datos que nos envia Simulink
 if(Serial.available()){
 Outvar = Serial.read();
  }

//convertimos la señal recibida de simulink que oscila entre 0 y 255, en la señal que nos interesa para controlar el motor.
OutvarM = map(Outvar, 0, 255, 630, 885);//el valor maximo que enviamos al motor es 885 porque es mas que suficiente para generar empuje en cualquier situacion
//pero podria alcanzar el valor 2000 donde se obtiene toda la potencia que desempeña el motor

var.writeMicroseconds(OutvarM);

//este If es para hacer un apagado suave del motor
 if(Serial.available()==0){
  if (Outvar>0){
 Outvar--;
 var.writeMicroseconds(OutvarM);
 delay(50);
   }
 }
 //Con este delay obtenemos un mejor desempeño.
delay(20);
}
```

Tal como se ve en el código, también he implementado una función que detecta el cierre de la comunicación, y hace un apagado suave del motor.

4. RESULTADOS FINALES

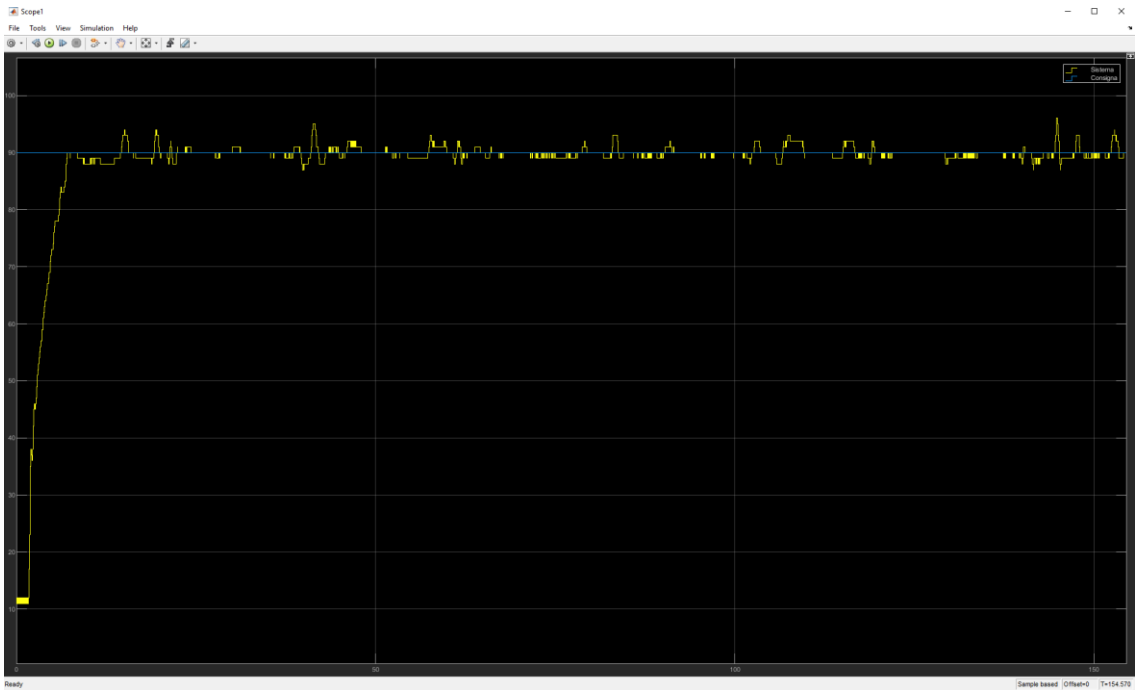
Por último, he realizado tres pruebas experimentales con las que ver con claridad la capacidad que tiene nuestro modelo de control final para perseguir cualquier tipo de consigna que se le ordene.

1. En la siguiente gráfica, la consigna se fue incrementando poco a poco hasta llegar a los 90º:
    - En amarillo vemos la respuesta del sistema.
    - En azul la señal de referencia.

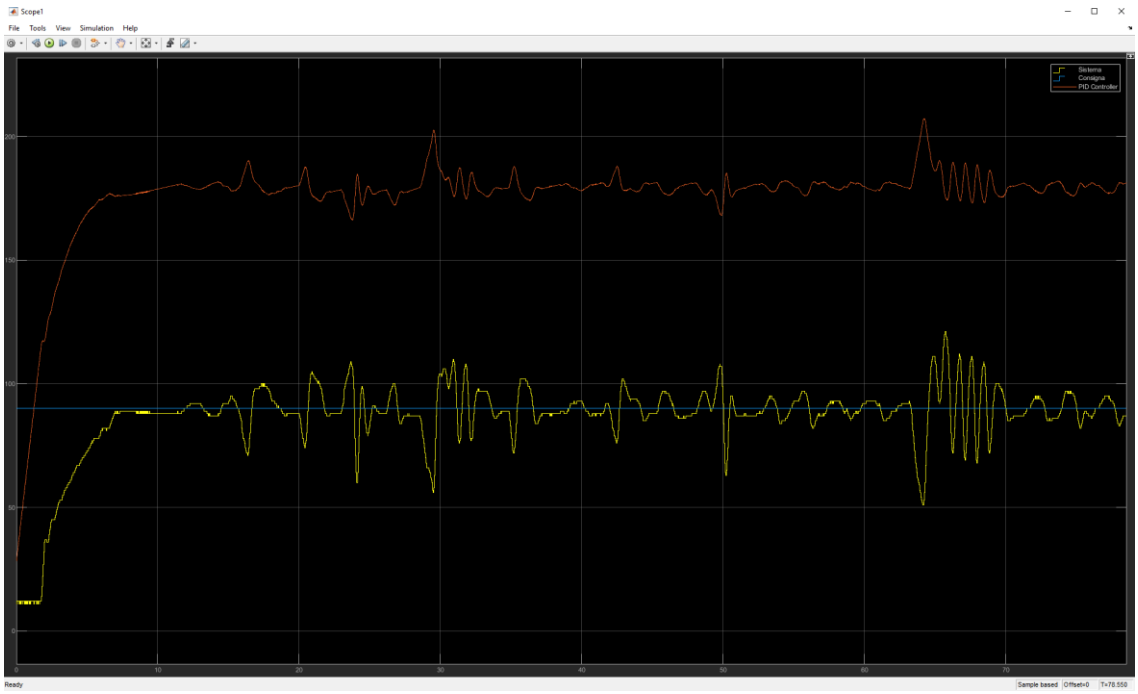


Como se puede observar, el modelo puede seguir con bastante precisión cualquier ángulo de referencia que se le indique, a partir del 33º, que es cuando la señal de control supera el umbral PWM que enciende el motor.

2. Otra prueba realizada, es alcanzar la consigna de 90º desde apagado, obteniendo el siguiente resultado:



3. Y por último, en esta prueba se arranco el sistema con 90º en la consigna, y una vez alcanzados simule perturbaciones moviendo manualmente el brazo de la maqueta. En este caso he incluido en la grafica la señal del controlador PID para poder ver su respuesta ante estas perturbaciones.
    - En amarillo vemos la respuesta del sistema.
    - En azul la señal de referencia.
    - En rojo la señal del regulador PID



5. CONCLUSIONES

A lo largo de la realización de esta memoria, el mayor problema con el que me he encontrado han sido las oscilaciones que el brazo de la maqueta realizaba al tratar de alcanzar la señal de referencia, debido en mayor medida a la resolución de la salida PWM con la que energizamos el variador, obteniendo siempre una respuesta de empuje muy escalonada, y pudiendo darse la situación de que el valor PWM inmediatamente inferior no alcance la consigna, y el posterior a este la sobrepase holgadamente.

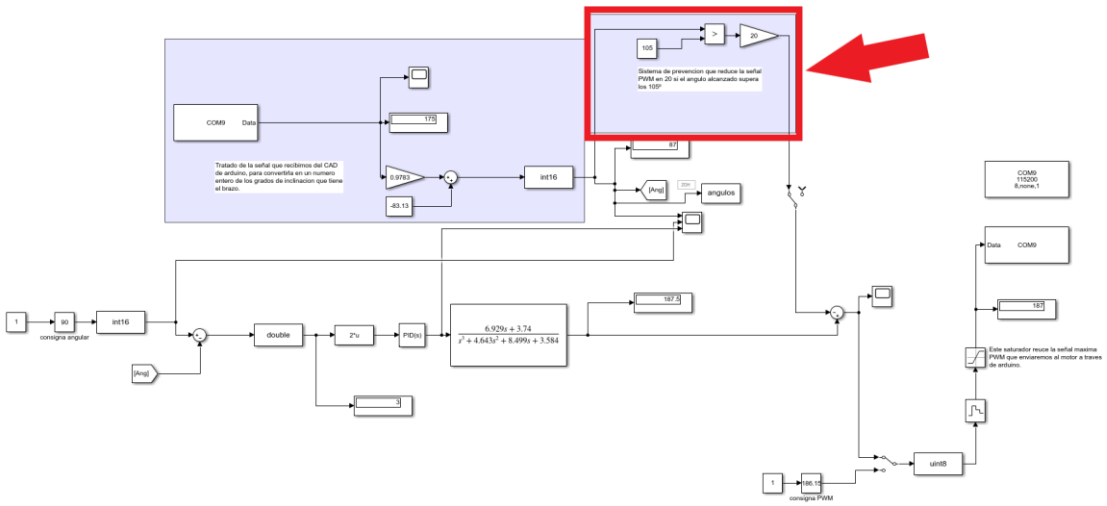
Esto es debido a dos factores:

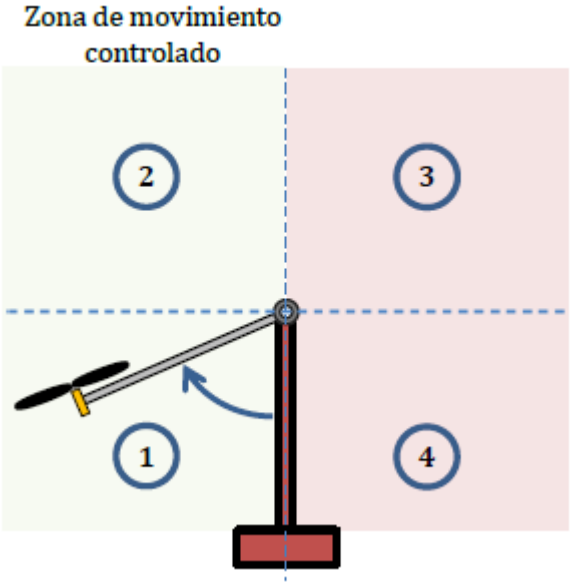
- Exceso muy amplio de potencia disponible en el empuje proporcionado por la turbina (agravado por el bajo peso del brazo de la maqueta)
- El ya comentado anteriormente, y relativo a la resolución de la señal PWM disponible para el control del motor.

Es debido a esto que el regulador que mejor ha funcionado ha sido el más lento y robusto, ya que si debido por ejemplo a la tensión de la batería en funcionamiento, se daba la circunstancia de no poder alcanzar la consigna con precisión, un regulador más rápido generaría oscilaciones al aumentar la señal PWM de control y disminuirla al sobrepasar la consigna de forma rápida y continua.

El mejor modo de resolver esto, seria aumentando la resolución de la señal que enviamos al variador, obteniendo así una mayor precisión en el control del empuje generado, siempre y cuando no sea el variador el que nos esté limitando esta resolución. No obstante, el resultado final obtenido, muestra que, aunque el empuje sigue siendo escalonado, las consignas se puedan alcanzar con bastante precisión.

Los resultados han sido en todo momento los esperados, y debido a que el regulador implementado es muy lento, se decidió implementar una medida de seguridad que evita que el sistema entre en zona inestable. Este sistema consiste en un comparador dentro del modelo de simulink, que cuando detecta que el ángulo del brazo sobrepasa los 105º, reduce en 20 puntos la señal PWM de salida, consiguiendo de esta manera mantener siempre el funcionamiento de la maqueta dentro del rango de estabilidad o movimiento controlado.





6. ENLACES DE INTERES

Repositorio del proyecto en Github:

https://github.com/Enrique-Mateos/Regulacion-automatica

Video demostrativo en Youtube:

https://youtu.be/_MGTpr9JmZs