Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías Industriales

Intensificación en Automática

DISEÑO Y CONTROL DE UN SISTEMA MOTOR-HÉLICE-BALANCÍN

Autor: Fernando Sayago Ruiz

Tutor: Manuel Gil Ortega Linares

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

DISEÑO Y CONTROL DE UN SISTEMA MOTOR-HÉLICE-BALANCÍN

Autor:
Fernando Sayago Ruiz

Tutor:
Manuel Gil Ortega Linares
Profesor titular

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: DISEÑO Y CONTROL DE UN SISTEMA MOTOR-HÉLICE-BALANCÍN

Autor: Fernando Sayago Ruiz

Tutor: Manuel Gil Ortega Linares

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia, mis padres, mis hermanos y mi tata, sin ellos no sería quien soy y nada de esto habría sido posible. Nunca les podré devolver todo lo que me han dado.

A mis amigos y compañeros, sin duda lo mejor de estos años, demasiadas horas juntos, amistades que durarán toda la vida.

Resumen

En el presente proyecto se realiza el diseño de un sistema motor-hélice-balancín, partiendo de un modelo inicial, al que se le realizan una serie de modificaciones en los componentes, para posteriormente realizar el control del sistema conjunto.

Un sistema motor-hélice-balancín está formado por una barra que tiene un grado de libertad, el cual es el giro alrededor de un eje vertical. El movimiento es producido por la fuerza de empuje ejercida por un motor con una hélice situado en el extremo libre de la barra. La variación de la velocidad de giro del motor permite controlar la fuerza de empuje, consiguiendo a través de un control adecuado llevar la barra a la posición deseada. La posición se determina a través de la medida del ángulo que forma la barra respecto de la vertical, para calcular este ángulo se plantean dos sensores, en primer lugar una unidad de medidas inerciales (IMU, Inertial Measurement Unit), y en segundo lugar un encoder.

Por otro lado, la integración de los datos y el control del sistema se realizan tanto en el entorno de desarrollo de Arduino, como en el entorno Simulink de Matlab, a partir de los cuales podemos, con un ordenador, controlar la velocidad de giro del motor y usar la medida del angulo como realimentación para controlar el sistema.

Finalmente se implementan una serie de controladores en el sistema y se incluyen las gráficas con los resultados obtenidos en los correspondientes experimentos.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

Sen	Función seno
Tg	Función tangente
Arctg	Función arco tangente
Cos	Función coseno
∂y ∂x	Derivada parcial de *y* respecto
$x_\circ$	Notación de grado, *x* grados.
<	Menor o igual
>	Mayor o igual

1 INTRODUCCIÓN

En el presente proyecto se pretende controlar la posición de un sistema mediante el uso de Arduino, y la herramienta Simulink incluida en Matlab. El sistema consta de una barra móvil anclada por uno de sus extremos a una barra verticar fija a modo de balancín como se puede observar en la figura 1, en uno de los extremos de dicha barra móvil se encuentra un motor con una hélice que será el encargado de producir la fuerza de empuje proporcional a la velocidad de giro de su hélice que será la responsable de los cambios de posición de nuestro sistema.

Figura 1. Sistema real y representación de la planta.

El sistema motor-hélice-balancín propone un problema de control de interés debido a su naturaleza, conteniendo zonas con puntos de equilibrio tanto estables como inestables. Como objetivos del presente trabajo, nos hemos propuesto, una vez completado el diseño inicial, en el cual realizaremos la elección del sensor de posición más adecuado tras someterlos a diferentes pruebas, realizar simulaciones sobre el modelo y comparar en cuanto a respuesta temporal el modelo matemático obtenido y el sistema físico, además de conseguir la regulación de la posición de la barra, logrando que en todo momento sea la deseada, implementando diferentes controladores.

En el presente proyecto, comenzaremos definiendo en detalle los componentes del sistema, tanto hardware como software, explicando su funcionamiento y características mas significativas.

Posteriormente, realizaremos un estudio del comportamiento del sistema, dónde desarrollaremos su modelo matemático, y obtendremos la función de transferencia del sistema para poder controlarlo.

Una vez definido el comportamiento del sistema, pasaremos a la elección del sensor de posición adecuado, dónde realizaremos diferentes pruebas tanto a la unidad de medidas inerciales (IMU), como al encoder, y analizaremos los resultados obtenidos.

Finalmente, una vez elegido el sensor de posición, se obtiene una función de tranferencia en un punto de funcionamiento, alternativa a la obtenida anteriormente, esta vez a partir de pruebas experimentales, y se diseñan diferentes controladores para el sistema.

2 DESCRIPCIÓN DE LOS COMPONENTES

El sistema está formado por dos barras, una barra fija, metálica de sección rectangular, colocada verticalmente y sujeta por la base inferior a un tablero de madera. En la parte superior de esta barra se encuentra atornillado un soporte cuya misión es sujetar el encoder, que servirá de punto de anclaje de la barra móvil. Dicha barra móvil contará con una longitud de 54 cm y que puede ser considerarda uniforme en su composición. En uno de los extremos de esta barra, que a partir de ahora llamaremos balancín, se encuentran el motor y la hélice, que proporcionarán la fuerza suficiente al sistema para que el balancín pueda elevarse. El balancín se encuentra unido al encoder en su punto medio que servirá como eje de giro, el cual es perpendicular al eje longitudinal de la barra, permitiendo el giro de la barra en el plano vertical. En el otro extremo del balancín se ha usado un pequeño lastre, que hemos ido variando hasta obtener el óptimo, para de este modo poder garantizar el movimiento completo del sistema, ya que el conjunto es relativamente pesado, y la fuerza máxima que genera la hélice no es la suficiente para garantizar el movimiento adecuado del balancín. Además tanto el encoder como el motor se encuentran conectados a una placa Arduino que es la encargada de realizar las operaciones y enviar la señal al driver controlador de motores que a su vez necesita de alimentación externa para ser capaz de proporcionar la potencia suficiente al motor. A continuación se explicarán con todo detalle el funcionamiento y el por qué de la elección de cada uno de los componentes empleados en la realización del proyecto.

2.1 Motor de corriente continua

El motor elegido ha sido un motor de corriente continua, de 12V con un consumo de 2,8W y una intensidad nominal de 0.4A según las especificaciones del vendedor.

Tiene una velocidad de hasta 12000 revoluciones por minuto cuando se le somete a la máxima potencia, y unas medidas y peso que lo hacen perfectamente adecuado para el funcionamiento que se desea, ya que mide unos 3 cm de largo, y su eje 1 cm aproximadamente. Además su peso apenas supera los 35 gramos, lo que lo convierten en el candidato perfecto.

Figura 2. Motor de corriente continua.

Este tipo de motores de corriente continua está compuesto de un estator y un rotor, generalmente en los motores mas pequeños, el estator está compuesto de imanes para crear un campo magnético.

El rotor es el dispositivo que gira en el centro del motor, y está compuesto de arrollados de cable conductores de corriente continua. Esta corriente continua es suministrada al rotor por medio de las escobillas.

Cuando un conductor por el que fluye una corriente continua es colocado bajo la influencia de un campo magnético, se induce sobre él una fuerza que es perpendicular tanto a las líneas de campo magnético como al sentido del flujo de la corriente. Esto es lo que sucede básicamente en el interior de un motor de corriente continua, pero en el rotor no hay solamente un conductor, sino muchos. Si se incluye otro conductor exactamente al otro lado del rotor y con la corriente fluyendo en el mismo sentido, el motor no girará pues las dos fuerzas ejercidas para el giro del motor se cancelan.

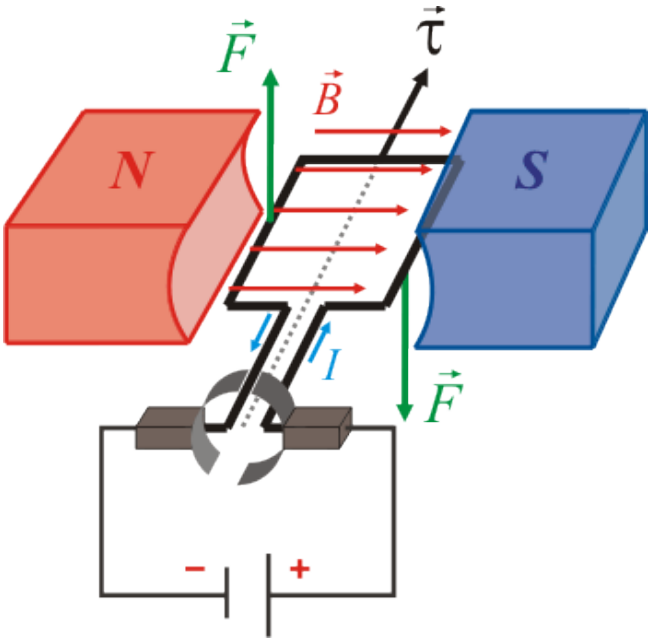

Figura 3. Funcionamiento del motor de corriente continua.

Es por esta razón que las corrientes que circulan por conductores opuestos deben tener sentidos de circulación opuestos. Si se hace esto, el motor girará por la suma de la fuerza ejercida en los dos conductores. Para controlar el sentido del flujo de la corriente en los conductores se usa un conmutador que realiza la inversión del sentido de la corriente cuando el conductor pasa por la línea muerta del campo magnético.

La fuerza con la que el motor gira (par motor) es proporcional a la corriente que hay por los conductores. A mayor tensión, mayor corriente y mayor par motor.

2.2 Arduino

Arduino es una placa electronica que contiene un microcontrolador y cuyo objetivo se basa en pequeños proyectos, para aficionados y amantes de la electronica 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 bateria sin necesidad de conectarse con el ordenador.

Arduino se programa a través de un programa gratuito 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.

2.2.1 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.

2.2.2 Comunicación Arduino – PC

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

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

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

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

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

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

2.2.3 Arduino Mega 2560

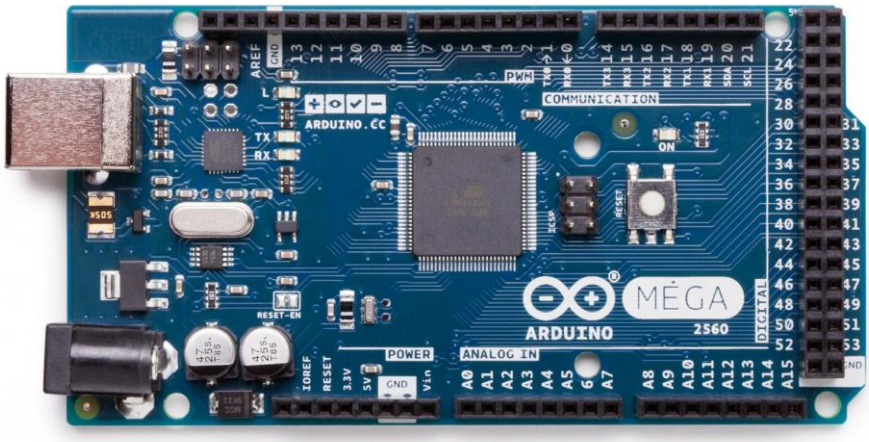

Figura 4. Placa Arduino Mega 2560.

En este proyecto se ha empleado la placa Arduino Mega 2560, la cual consta de 54 pines que pueden ser configurados como entrada o salida (de los cuales 15 se pueden utilizar como salida PWM), 16 entradas analógicas, pines analógicos que pueden configurarse para la conexión I2C, 4 UARTs (puertos serie de hardware), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP, y un botón de reset. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o a la corriente con un adaptador de CA a CC o una batería.

El Arduino Mega 2560 tiene 256 KB de memoria flash para almacenar el código (de la que se utilizan 8 KB para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leída y escrita).

Cada uno de los 54 pines digitales de la placa se puede utilizar como una entrada o como una salida, que operarán a 5 voltios. Cada pin puede proporcionar o recibir 20 mA como condición de funcionamiento recomendada y tiene una resistencia de pull-up (desconectada por defecto) de 20-50 k ohmios. Un máximo de 40 mA es el valor que no debe superarse para evitar daños permanentes en el microcontrolador. Además, algunos pines tienen funciones especializadas:

- Serie: Se utiliza para recibir (RX) y transmitir datos serie (TX) TTL.
- Interrupciones externas: Estos pines pueden configurarse para activar una interrupción en un nivel bajo, un flanco ascendente o descendente, o un cambio en el nivel.
- PWM: 2 a 13 y 44 a 46. proporcionan una salida PWM de 8 bits.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI utilizando la biblioteca SPI.
- LED: 13. Hay un led incorporado conectado al pin digital 13. Cuando el pin está a nivel alto, el led está encendido, cuando el pin está a nivel low, está apagado.
- TWI: 20 (SDA) y 21 (SCL). TWI soporte de comunicación utilizando la biblioteca Wire.

El Mega 2560 tiene 16 entradas analógicas, cada una de las cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se miden de masa a 5 voltios, aunque es posible cambiarlo.

2.3. Simulink

Simulink es una herramienta de Matlab que funciona mediante un entorno de programación visual, dónde 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). 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.

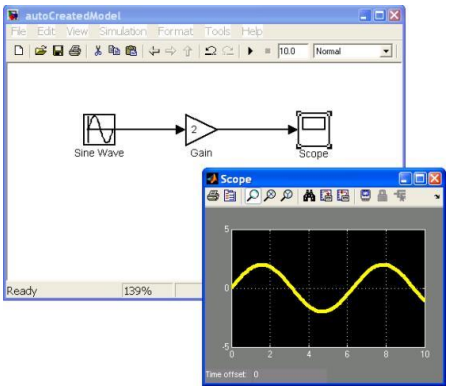

Figura 5. Entorno de programación Simulink.

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, 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.

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, sources.

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.

Hay varias bibliotecas que nos permiten trabajar con Arduino y Simulink, nosotros hemos elegido la que nos ofrece para instalar como suplemento Simulink ya que es gratuita, oficial y su uso es relativamente sencillo. Con esta biblioteca puedes establecer conexión con todos los tipos de arduino.

Al tratarse de un modelo que interactúa en todo momento con componentes físicos externos (Arduino, IMU, motor, encoder,…) es necesario que la simulación sea en tiempo real con el fin de verificar las interacciones del mismo, ya que están diseñados para un funcionamiento exclusivamente en tiempo real. De igual modo, si se quiere realizar un control sobre el sistema, al ser éste un sistema real con respuesta en tiempo real ante cambios es necesario una simulación en tiempo real que responda a las exigencias del sistema.

Para ello contamos con el modo External, que permite modificar los parámetros y visualizar los datos del

modelo de Simulink mientras se está ejecutando en el hardware Arduino en tiempo real. Esto permite un ajuste de parámetros más cómodo y rápido, ya que se puede visualizar en tiempo real cómo reacciona el sistema antes cambios de las variables durante la simulación.

El uso del modo externo conlleva algunas limitaciones del modelo:

No se pueden configurar bloques de envío y recepción por el puerto serie 0 debido a que el modo externo utiliza dicho puerto. En su lugar, permite utilizar bloques de comunicación serie con conexión Wi-Fi o el modo externo de TCP / IP.

No se puede utilizar los siguientes bloques servo para Arduino: Standard Servo Read, Standard Servo Write y Continuous Servo Write.

Estas limitaciones no afectan a nuestro modelo ya que el bloque que se va a utilizar es el envío de señal PWMy entrada de señal digital.

2.4. Driver para motores (L298N)

El L298N es un controlador (driver) de motores, que permite encender y controlar dos motores de corriente continua desde Arduino, variando tanto la dirección como la velocidad de giro, o un único motor paso a paso.

Arduino, y en general todos los autómatas, no disponen de potencia suficiente para mover actuadores. De hecho, la función de un procesador no debe ser ejecutar acciones si no mandar ejecutar acciones a drivers que realicen el "trabajo pesado".

La corriente máxima que el L298N puede suministrar a los motores es, en teoría**,** 2A por salida (hasta 3A de pico) y una tensión de alimentación de 3V a 35V**.** Sin embargo, el L298N tiene una eficiencia baja. La electrónica supone una caída de tensión de unos 3V, es decir, la tensión que recibe el motor es unos 3V inferior a la tensión de alimentación.

Estas pérdidas se disipan en forma de calor lo que se traduce en que, a efectos prácticos, es difícil que podamos obtener más de 0.8-1A por fase sin exceder el rango de temperatura de funcionamiento.

El L298N incorpora protecciones contra efectos que pueden producirse al manejar motores de corriente continua. Dispone de protecciones contra sobre intensidad, sobre temperatura, y diodos de protección contra corrientes inducidas (flyback).

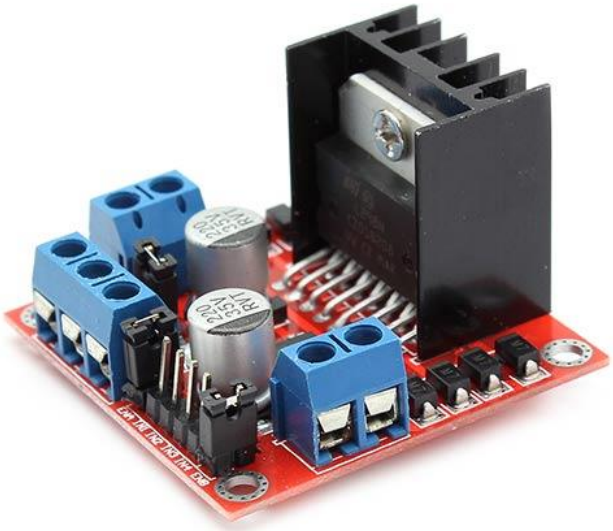

Figura 6. Driver controlador de motores L298N

Basicamente un L298N consiste en dos puentes H, uno para la salida A y otro para la salida B. Un puente H es un circuito electrónico, formado por interruptores o por transistores, que permite a un motor eléctrico de corriente continua girar en ambos sentidos, avance y retroceso, además de controlar su velocidad de giro.

Internamente el L298N emplea puentes H con 4 transistores, conectados entre Vcc y GND, con la carga a alimentar entre ellos. Dibujado en esquema el conjunto tiene forma de "H", de la que recibe su nombre su nombre.

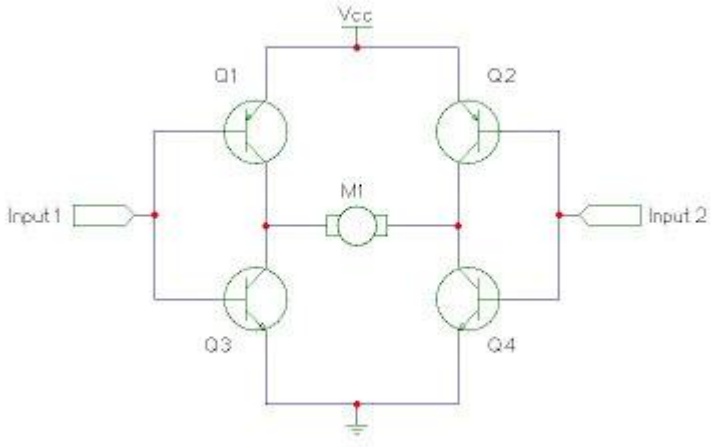

Figura 7. Estructura interna del L298N.

Actuando sobre los 4 transistores, activando los transistores opuestos en diagonal de cada rama, podemos variar el sentido en el que la corriente atraviesa la carga.

Conectando simultáneamente los transistores superiores o inferiores, podemos poner la carga Vcc o Gnd respectivamente, configuración que usaremos como freno. Nunca se debe encender ambos transistores de un mismo ramal (izquierda o derecha), ya que se provocaría un cortocircuito entre Vcc y GND.

No obstante, la placa L298N incorpora electrónica que simplifica la conexión al puente H, agrupando las conexiones en 3 pines accesibles (por cada salida) y eliminando la posibilidad de generar un cortorcuito.

2.5. Fuente de tension

Para suministrar al motor la potencia eléctrica necesaria se necesita una batería o fuente de alimentación que irá conectada al puente H que ha sido descrito en el apartado anterior.

Como el modelo va a ser usado en un laboratorio o en algún aula con acceso a la corriente eléctrica, emplearemos una fuente de corriente continua en lugar de una batería o algo similar. Como principal ventaja destaca que la fuente de corriente proporciona una corriente continua estable y constante durante todo el tiempo, pudiendo regular el voltajeque se le suministra al driver controlador del motor a través de un interruptor, algo que no ocurre en el caso de las baterías ya que pueden perder potencia conforme se van descargando. Es por ello que se ha escogido como alimentación del motor una fuente de corriente continua, que irá conectada al circuito a través de dos cables, positivo y negativo.

Contamos con un motor de corriente continua al cual hay que aplicar una tensión de 12 Voltios de corriente continua, por tanto un requisito indispensable para la fuente de corriente continua elegida, es que debe proporcionar una tensión de 12 Voltios con garantías y una intensidad de corriente mayor o igual a 3 Amperios.

La fuente de tensión elegida es la que se encuentra en el área de trabajo del laboratorio, la cual cumple con los requisitos depotencia que se han comentado anteriormente. Se trata del modelo EP-920 de la marca Silver Electronics, la cual se conecta a la red eléctrica de 230 Vac y 50 Hz, y puede darnos una salida de entre 3 y 15 Voltios, y hasta 18 Amperios de corriente continua.

Figura 8. Fuente de alimentación utilizada.

La fuente dispone de un botón giratorio donde podemos ajustar el voltaje de salida deseado, valor en Voltios que aparecerá indicado en un indicador de aguja. También dispone de dos conexiones tipo banana, entre las que se proporciona la tensión de salida indicada previamente con el botón giratorio.En estos dos conectores es donde se conectarán los cables dealimentación del driver, a una tensión de 12V. La fuente dispone de un interruptor ON/OFF, que permite apagar en cualquier momento el dispositivo. El tener un botón de apagado y encendido es otra ventaja respecto a usar una batería portátil, ya quete permite detener el motor en caso de fallo del sistema.

2.6. Unidad de medidas inerciales (IMU)

Una IMU o unidad de medidas inerciales es un dispositivo electrónico que aporta datos de velocidad, orientación y fuerza gravitacional a partir de los datos proporcionados por un acelerómetro y un giróscopo. Algunas IMUs incluyen también un magnetómetro, que es empleado para medir la fuerza y dirección de un campo magnético.

A partir de los datos medidos de aceleración y velocidad angular por el giróscopo y el acelerómetro respectivamente, se pueden realizar una serie de cálculos con los cuales podemos obtener los ángulos de inclinación de la unidad de medidas inerciales en cada uno de los tres ejes. Para ello existen diferentes métodos de integración que permiten obtener la posición del dispositivo respecto de un sistema de referencia.

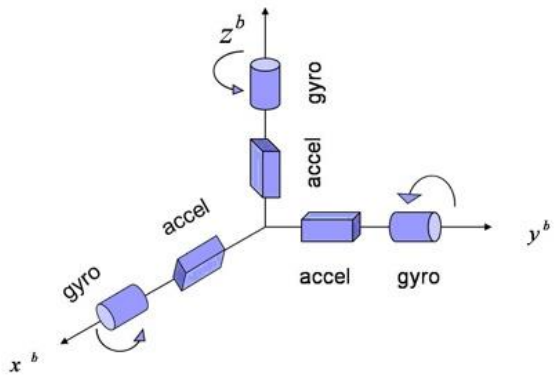

Figura 9. Funcionamiento de la Unidad de Medidas Inerciales

Una de las características más importantes de la IMU es que no necesita información externa para aproximar su posición, es decir, es un dispositivo autónomo, sin embargo presenta un inconveniente a tener en cuenta y es que, debido a que necesita integrar los valores de la aceleración y velocidad constantemente para obtener la posición, aparece un error acumulativo que va aumentando con el tiempo. Además, también aparece un error en la medida del ángulo ya que el acelerómetro, a parte de medir los valores de la gravedad, mide mas tipos de aceleración, que a veces no son las deseadas, como las vibraciones, lo que es conocido como ruido. Por ello suele ser necesario aplicar algún tipo defiltro a las medidas que se obtienende la IMU, será explicado más adelante, y de este modo poder obtener datos fiables y precisos.

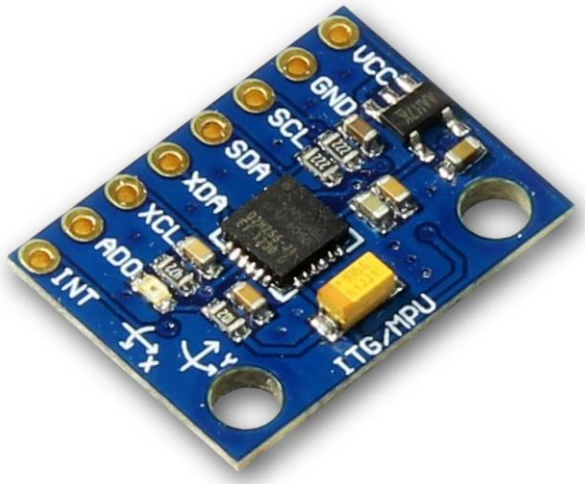

Figura 10. Unidad de Medidas Inerciales.

La IMU escogida ha sido el dispositivo MPU6050. Se trata de una unidad de medidas inerciales de pequeño tamaño, con buenas prestaciones, precisa y de bajo consumo y coste. Se dispone de una pequeña placa en la que se puede acceder fácilmente a todos los pines del MPU6050 y conectarlos a un microcontrolador para comunicarse con el dispositivo, en este caso a un Arduino.

El MPU 6050 es un dispositivo de 6 ejes, que contiene un giróscopo y un acelerómetrode 3 ejes cada uno.

Una de las consideraciones que hay que tener en cuenta a la hora de conectarlo con Arduino es que el protocolo de comunicación es I2C y que requiere una alimentación de 3.3V.Las conexiones se realizan a través de 4 pins: dos para la comunicación I2C y otros dos para la alimentación (3.3 V y GND).

2.7. Encoder

Un encoder es un transductor rotativo, que mediante una señal eléctrica (normalmente un pulso o una señal senoidal) nos indica el ángulo girado. Si este sensor rotatorio lo conectáramos mecánicamente con una rueda o un husillo, también nos permitiría medir distancias lineales.

Una clasificación de los encoder según el tipo de información sobre la posición que generan sería:

- Encoder incremental: La señal de salida se transmite por un hilo en el que se transmite un pulso por cada ángulo girado, de tal forma que si tenemos un encoder de 1000 ppr, tendremos un pulso por cada 360º/1000= 0,360º. El inconveniente es que no disponemos de una referencia absoluta de la posición en la que se encuentra el eje.

- Encoder absoluto: La posición se da en valor absoluto mediante un bus paralelo. Es decir, que si tenemos un encoder de 256 posiciones, tendremos un bus de 8 líneas que nos indicaran en binario cual es su posición (normalmente estos transductores codifican la posición en código gray para evitar errores). El inconveniente de estos encoders es la cantidad de líneas que necesitamos leer y conectar y que debido a la complejidad del disco óptico que codifica las posiciones la resolución no suele ser muy elevada.

En nuestro caso, para este proyecto vamos a usar un encoder incremental, que suelen ser los más utilizados y de los cuales encontramos gran variedad en el mercado, pudiendo elegir el que mejor se adapte a nuestras necesidades.

Las características básicas de un encoder incremental óptico son:

- Tensión de alimentación: Nos indica a que tensión puede trabajar el encoder. A veces es fija (5v, 12v, etc...), pero lo habitual es que sea un rango de tensiones.
- Resolución: Es el número de pulsos que da por revolución (ppr).
- Tipo de salida: Las salidas de los canales pueden ser de varios tipos; TTL, colector abierto, tótem-pole, etc..., por lo que habrá que utilizar el circuito adecuado para adaptar estas salidas.
- Número de canales: Suelen ser 1 o 2, más un canal adicional de index que de un pulso por vuelta. Con los encoders de un solo canal podemos saber el ángulo girado pero no la dirección de giro, por lo que la mayoría de los encoders llevan dos canales que generan señales cuadradas desplazadas 90º. Este desfase, como veremos más adelante, es el que nos permite determinar la dirección de giro.

En nuestro caso hemos elegido para nuestro proyecto un encoder incremental con 512 pulsos por revolución, tensión de alimentación de 5V, con un eje liso de unos 2cm aproximadamente, que produce una onda digital cuadrada, y cuya máxima velocidad de funcionamiento es de 10.000 revoluciones por minuto.

Está fabricado por la compañía americana Bourns, especializada en la fabricación de componentes electrónicos de este tipo.

Figura 11. Encoder empleado en la planta.

3 COMPORTAMIENTO DEL SISTEMA

Antes de comenzar con el control del sistema hay que conocer su comportamiento dinámico. En este capítulo se realiza un estudio del movimiento y a continuación se obtiene una representación matemática del comportamiento del sistema, es decir, una ecuación que define la posición del sistema en cada instante.

Un modelo matemático permite predecir la evolución del sistema, analizar el comportamiento, analizar el efecto de la variación de parámetros sobre la evolución y estudiar el efecto de las entradas sobre la evolución del sistema.

3.1 Movimiento de rotación de un sólido rígido

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

$$\Sigma r \times F_{ext} = I\alpha \qquad (3.1)$$

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

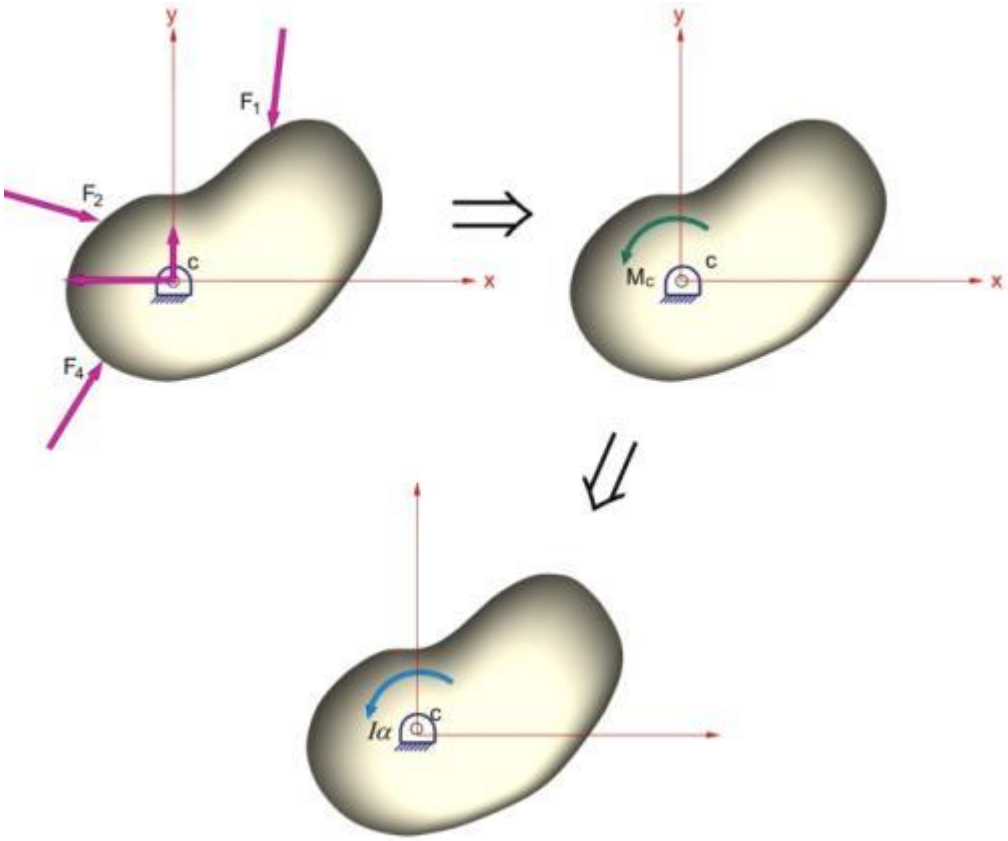

Figura 12. Movimiento de rotación de un sólido rígido.

3.2 Obtención de la función de transferencia

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

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

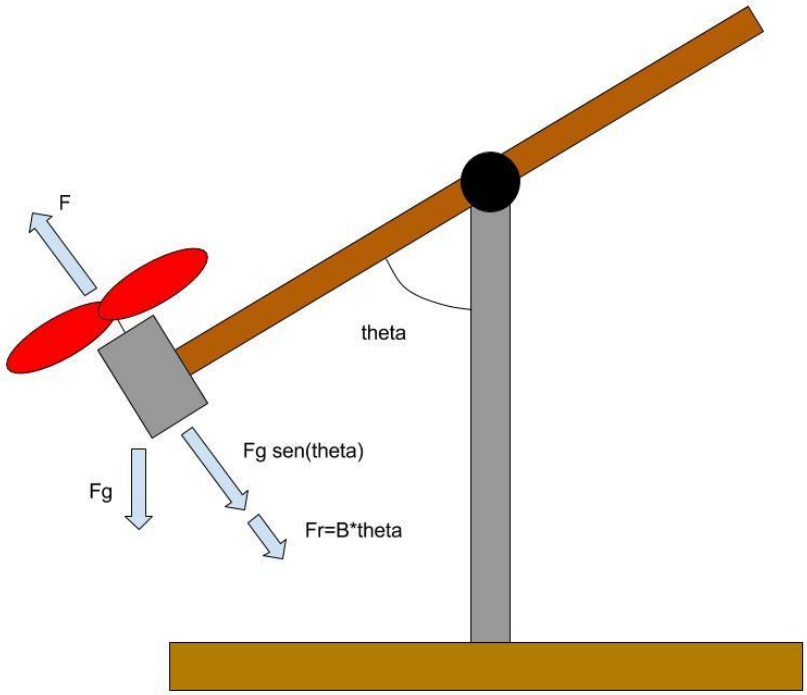

Figura 13.Representación de las fuerzas que actúan sobre el sistema.

Para determinar el modelo matemático aproximado de este sistema, se empleará la ecuación del movimiento de rotación de un sólido análoga a la segunda ley de Newton y representada por la ecuación 3.1. Una vez aplicada esta ecuación y luego de haber obtenido cada uno de sus parámetros se realizará la transformada de Laplace a la ecuación resultante, así de este modo se podrá obtener la función de transferencia del modelo aproximado.

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

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

$$lF(t) - B\dot{\theta}(t) - F_g\, sen\theta(t) = I\ddot{\theta}(t) \qquad (3.2)$$

donde:

Θ: posición de la barra respecto al eje de giro.

F. fuerza de empuje generada por el giro de la hélice.

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

B: coeficiente de rozamiento.

I: momento de inercia del Sistema móvil

F_g: Fuerza de gravedad que actúa sobre el motor

La ecuación (3.2) relaciona la fuerza de empuje que ejerce la hélice (entrada del sistema,F(t)), con el ángulo que forma la barra con la horizontal (salida del sistema, $\theta(t)$), permitiendo conocer su posición en todo instante de tiempo. Es una ecuación diferencial de segundo orden no lineal. Para poder obtener la función de transferencia que relacione la entrada con la salida habrá que linealizar la ecuación anterior en torno a un punto de equilibrio. Se pretende encontrar la funciónde transferencia G(s) que represente el comportamiento del sistema.

Con el fin de linealizar la ecuación (3.2), se considera la función $f(\ddot{\theta},\dot{\theta},\theta,F)$ de la forma:

$$f(\ddot{\theta},\dot{\theta},\theta,F) = I\ddot{\theta} + B\dot{\theta} + F_g sen\theta - lF = 0 \qquad (3.3)$$

Suponiendo un punto de equilibrio conocido ($\ddot{\theta}_0 = 0, \dot{\theta}_0 = 0, \theta = \theta_0, F = F_0$) tal que:

$$f(0,0,\theta_0,F_0) = F_g sen\theta_0 - lF_0 = 0 \qquad (3.4)$$

Se definen las variables incrementales α(t) y β(t), que dependen del punto de equilibrio.

$$\alpha(t) = \theta(t) - \theta_0 \quad \dot{\alpha}(t) = \dot{\theta}(t) \quad \ddot{\alpha}(t) = \ddot{\theta}(t)$$

$$\beta(t) = F(t) - F_0$$

Y se realiza el desarrollo en series de Taylor de la función f en torno al punto de equilibrio ($\ddot{\theta}_0 = 0, \dot{\theta}_0 = 0, \theta = \theta_0, F = F_0$):

$$f(\ddot{\theta},\dot{\theta},\theta,F) = f(0,0,\theta_0,F_0) + \frac{df}{d\ddot{\theta}}]_{\ddot{\theta}=0}\ddot{\alpha} + \frac{df}{d\dot{\theta}}]_{\dot{\theta}=0}\dot{\alpha} + \frac{df}{d\theta}]_{\theta=\theta_0}\alpha + \frac{df}{dF}]_{F=F_0}\beta + \sigma(\alpha^2,\beta^2) = 0 \qquad (3.5)$$

Donde $f(0,0,\theta_0,F_0)$ y $\sigma(\alpha^2,\beta^2)$ pueden considerarse nulos. Realizando las derivadas correspondientes a la ecuación anterior se obtiene:

$$\frac{df}{d\ddot{\theta}}]_{\ddot{\theta}=0}\ddot{\alpha} + \frac{df}{d\dot{\theta}}]_{\dot{\theta}=0}\dot{\alpha} + \frac{df}{d\theta}]_{\theta=\theta_0}\alpha + \frac{df}{dF}]_{F=F_0}\beta = 0 \qquad (3.6)$$

$$I\ddot{\alpha}(t) + B\dot{\alpha}(t) + F_g cos\theta_0\alpha(t) - lF_0\beta(t) = 0 \qquad (3.7)$$

A esta ecuación podemos aplicarle la transformada de Laplace, con el fin de obtener la función de transferencia $G(s) = \frac{\alpha(s)}{\beta(s)}$. Usando la propiedad de la transformada de Laplacede la derivada se llega a:

$$I[s^2\theta(s) - s\theta(0) - \theta(0)] + B[s\alpha(s) - \alpha(0)] + F_g\cos(\theta_0)\,\theta(s) - lF_0\beta(s) = 0 \qquad (3.8)$$

Considerando las condiciones iniciales nulas y despejando, se obtiene la siguiente función de transferencia aproximada del sistema, con entrada la fuerza que genera el giro de la hélice y salida el ángulo que forma la barra con la vertical:

$$G(s) = \frac{\theta(s)}{F_e(s)} \approx \frac{\alpha(s)}{\beta(s)} = \frac{l/I}{s^2 + \frac{B}{I}s + \frac{F_g}{I}} = \frac{K\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2} \qquad (3.9)$$

Como se puede observar la función de transferencia se corresponde a la de un sistema de segundo orden, en la cual algunos parámetros son conocidos, como la fuerza de la gravedad, la posición en el punto de equilibrio (ángulo), los valores de la fuerza, la distancia del eje de giro de la hélice al eje de giro de la barra, y hemos realizado un cálculo aproximado del momento de inercia, aunque más adelante realizaremos un cálculo más exacto a través de pruebas experimentales.

4 IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se tratará todo lo relacionado con la instalación de los complementos necesarios para el control del sistema, el montaje del circuito de la planta, las conexiones tanto del encoder como de la IMU, los pasos que hemos seguido para seleccionar el sensor de posición mas adecuado, así como el modelo que hemos empleado en Simulink para la recepción de los datos de ambos sensores.

4.1 Instalación de la biblioteca Arduino en Simulink

A continuación se detallarán todos los pasos a seguir para la instalación de la biblioteca Arduino para Simulink que se introdujo anteriormente en el capítulo 2 en el cual se presentaban los componentes que se emplearían en a realización del proyecto:

1- En primer lugar, una vez se ha abierto MATLAB, se debe pinchar en la pestaña "Add-Ons" y dentro de esta en el apartado "Get Hardware Support Packages".

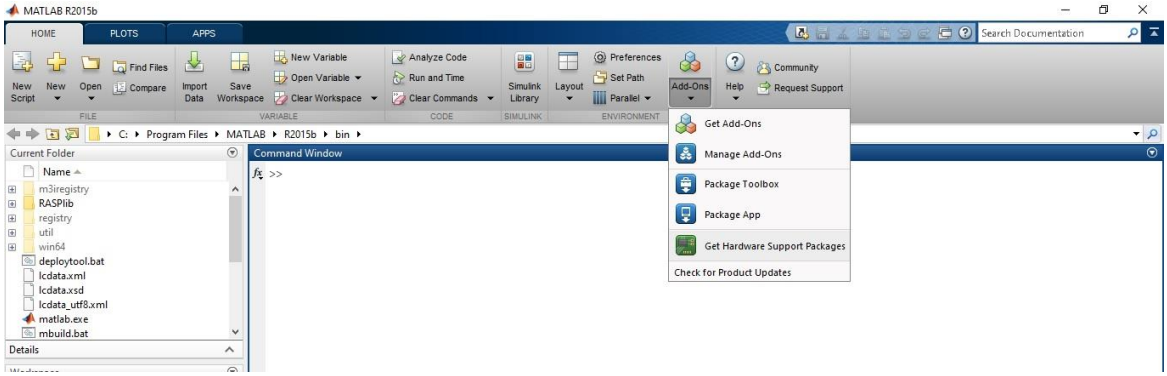

Figura 14. Instalador de paquetes de soporte de Matlab.

2- A continuación se nos abrirá una ventana en la cual se debe seleccionar el apartado "Install from Internet" y seguidamente pinchar en Next.

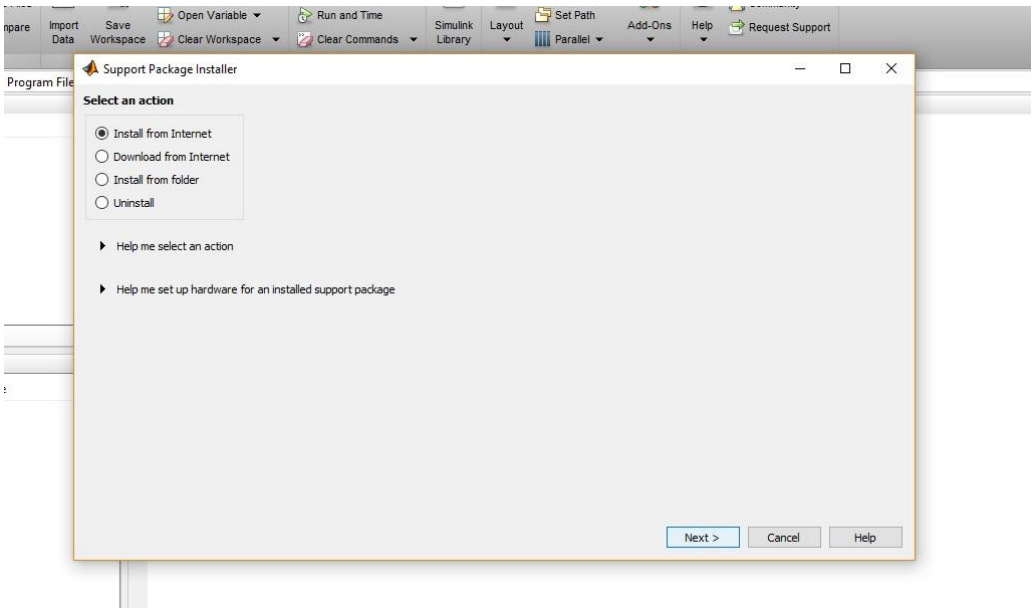

Figura 15. Diferentes opciones para instalar paquetes en Matlab.

3- Dentro del menú de la derecho que aparecerá en la nueva ventana, se debe seleccionar el apartado Arduino, y dentro de este aparecerán los dos programas que se pueden ver junto con la opción "Install" (en el caso de la imagen aparece "Reinstall" ya que se encontraban instalados), una vez seleccionados las dos pestañas pertinentes se debe pinchar en Next para continuar con la instalación.

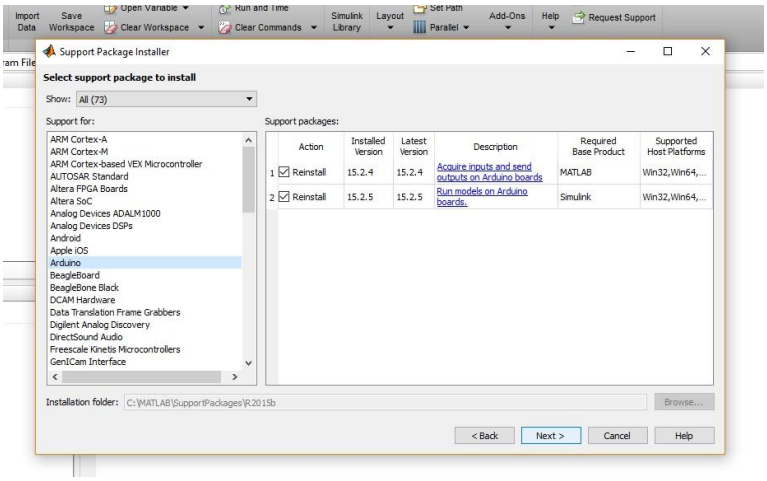

Figura 16. Paquetes Arduino disponibles para instalar

4- A continuación, en la siguiente ventana debemos de hacer click en "Log In"

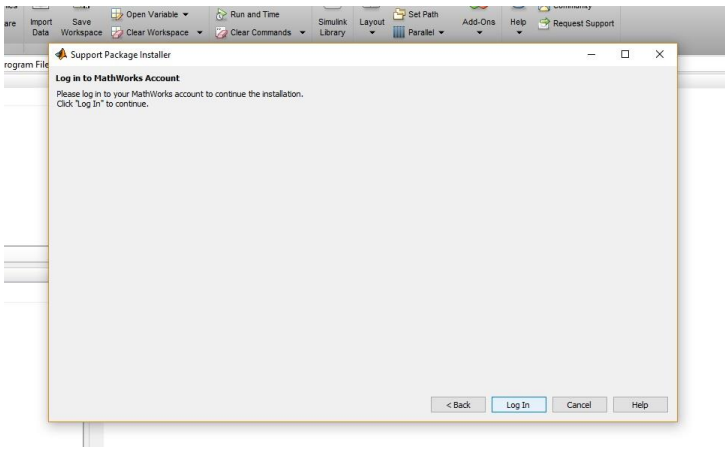

Figura 17. Ventana de acceso al registro.

5- En la ventana que aparecerá a continuación se puede elegir entre crear una cuenta nueva de MathWorks si aún no se cuenta con una, o de introducir nuestro usuario y contraseña si ya se cuenta con una.

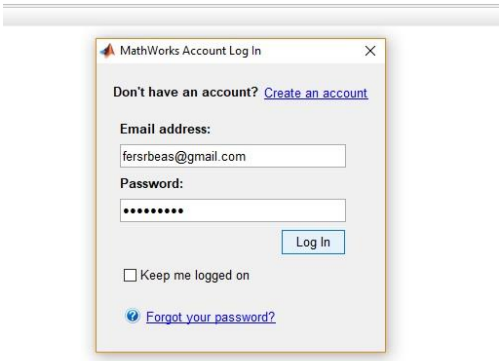

Figura 18. Ventana de *Log In*.

6- Una vez se introducen los datos de la cuenta, se debe seleccionar la pestaña “I accept” y pinchar en Next para continuar con el proceso.

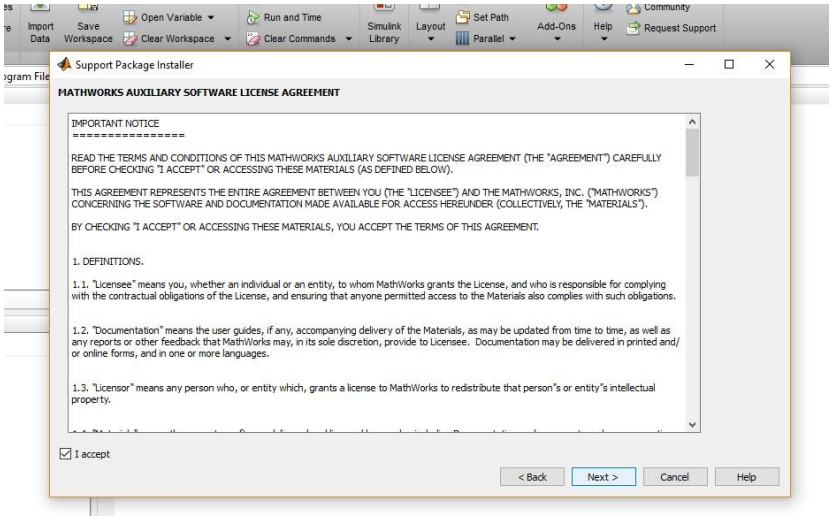

Figura 19. Términos y condiciones de acceso.

7- En la ventana que aparece a continuación se debe pinchar en Next.

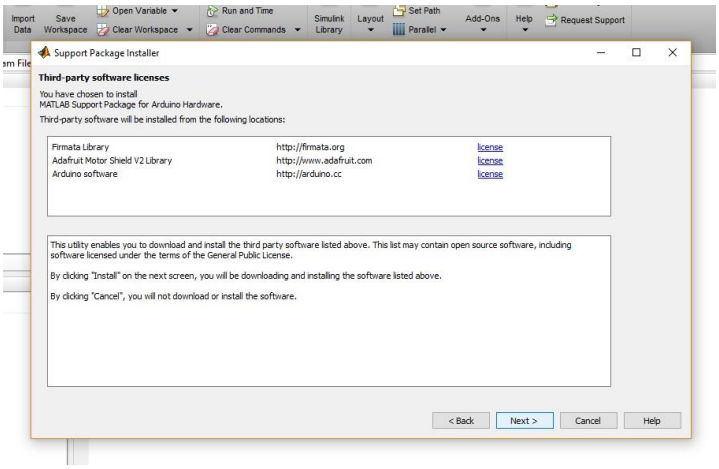

Figura 20. Aceptar licencias y descargar antes de instalar.

8- Finalmente, en la nueva ventana se pincha en “Install” para comenzar con el proceso de instalación.

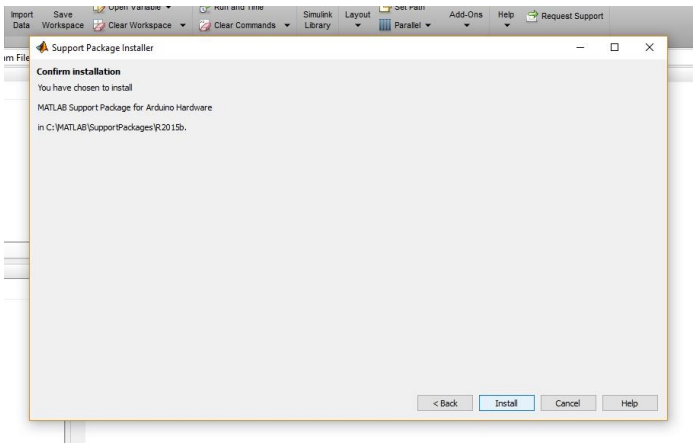

Figura 21. Instalar productos seleccionados.

9- Una vez completada la instalación es importante pinchar en Continue para realizar un ajuste necesario para el correcto uso de la placa Arduino.

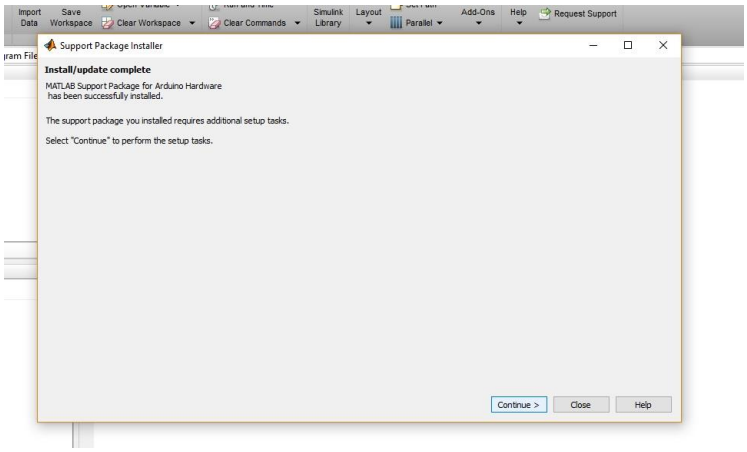

Figura 22. Ajustes necesarios para un correcto funcionamiento.

10- En la ventana que aparece a continuación se debe seleccionar la pestaña "Enable Installation of Arduino USB Driver" y pinchar en "Next".

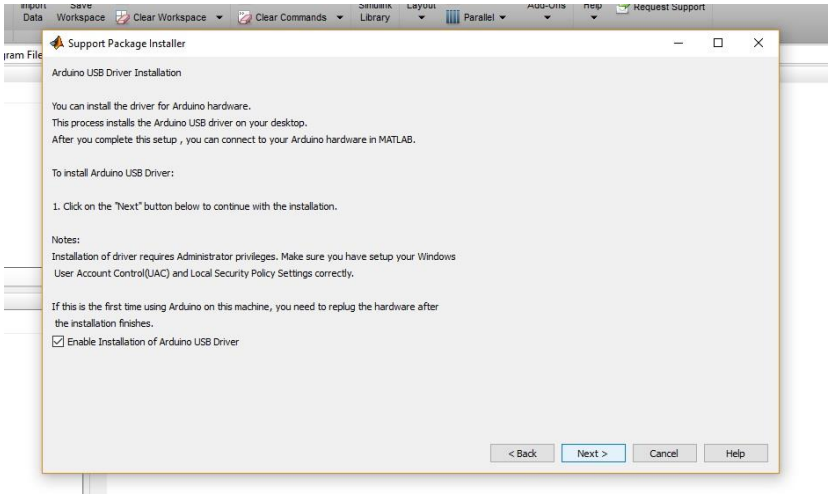

Figura 23. *Enable Installation of Arduino USB Driver.*

11- Finalmente se pincha en la pestaña "Finish" para concluir con la instalación de la biblioteca.

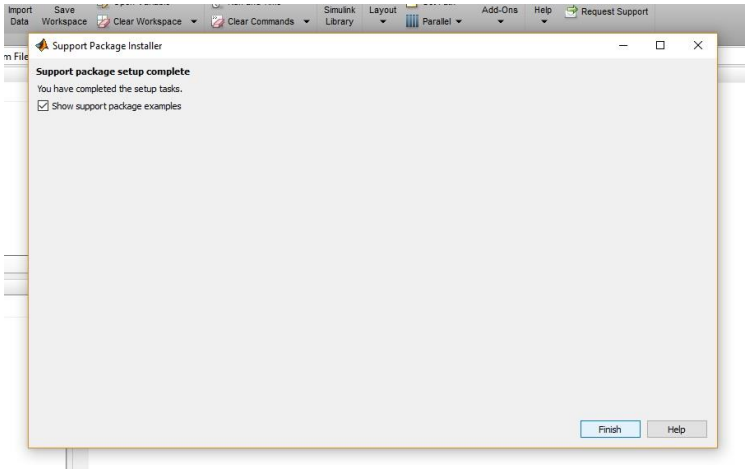

Figura 24. Fin de la instalación.

4.2 Circuito para el control de velocidad del motor

Para poder controlar la posición del balancín necesitamos como primer paso poder controlar la velocidad del motor, aumentarla o disminuirla a partir de la señal de control que será la modulación por ancho de pulsos (PWM).

La modulación por ancho de pulsos 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.

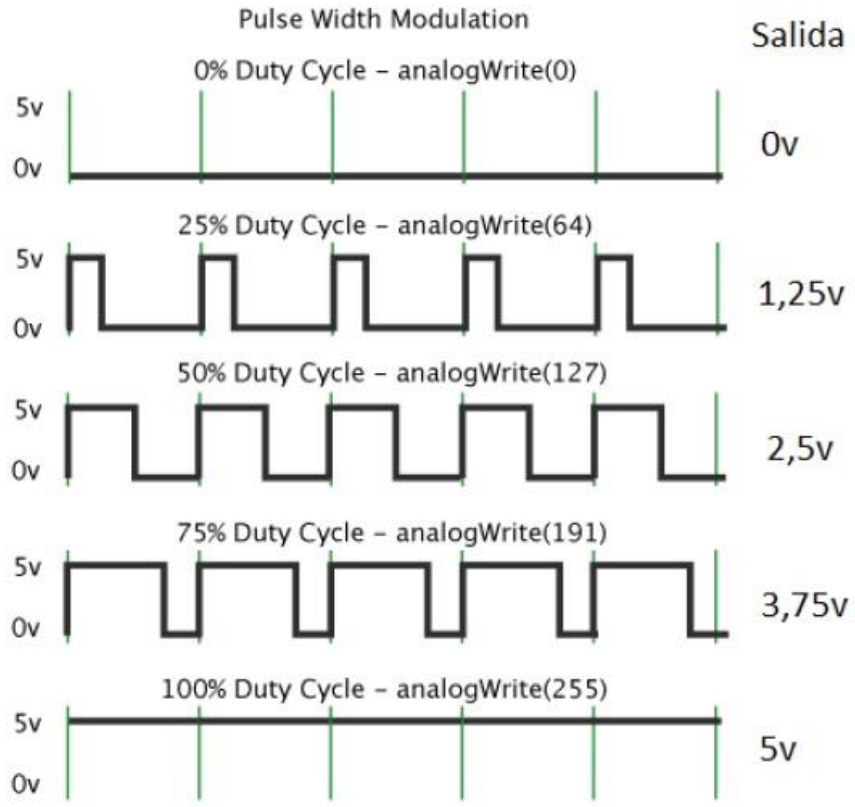

Figura 25. Funcionamiento PWM.

Esta señal PWM debe de ser recibida por el driver controlador de motores (L298N) a través de la placa Arduino, además esta placa está alimentado por la fuente externa a 12V que es la tensión de referencia del motor.

La placa Arduino y el L298N están conectados entre sí por tres cables, dos de ellos se encargan de controlar el sentido de giro del motor, son salidas digitales y dependiendo de si están a nivel alto o nivel bajo el motor girará en un sentido o en el contrario. El otro pin también conectado a una salida digital de la placa con PWM se encarga de enviar la señal de modulación por ancho del pulso al driver.

Es el driver el que está conectado directamente al motor y el que envía la señal de voltaje a este para que la fuerza de giro de la hélice sea la deseada.

Las tierras, tanto de la placa Arduino, como de la fuente de alimentación y del L298N están conectadas en común en la protoboard, de lo contrario podrían producirse fallos en el circuito y en el posterior control de la planta.

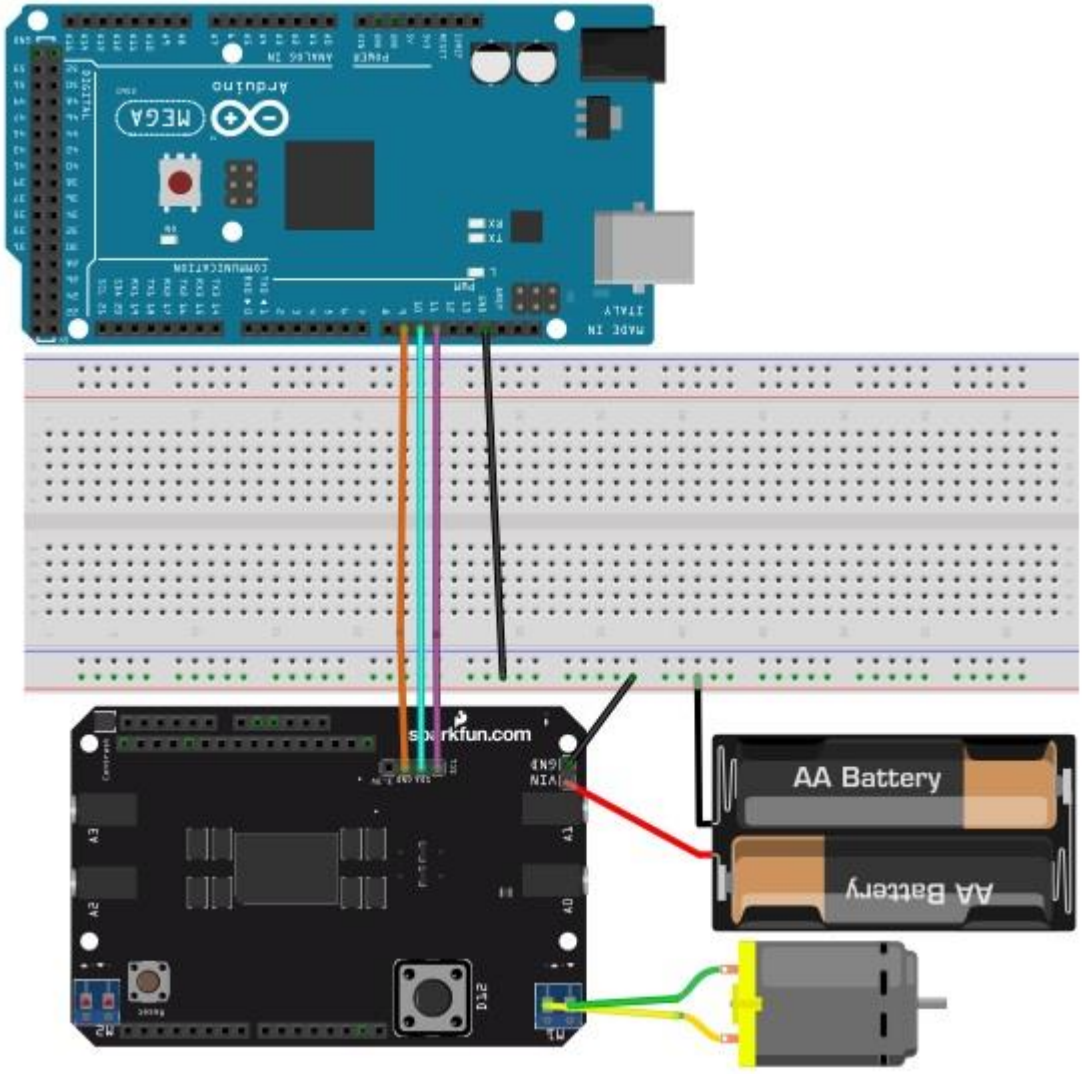

Figura 26. Esquema de montaje del L298N.

En Simulink para enviar la señal PWM al motor debemos de hacer el montaje de la figura, en el cual a través de los pines digitales 10 y 11 y según el nivel de la salida de estos como hemos dicho anteriormente controlamos el sentido de giro del motor, mientras que a través del pin número 12 de la placa Arduino enviamos la señal de control, que la podemos ir variando a través de un "Slider Gain" para aumentarla o disminuirla en función de nuestras necesidades.

En la imagen también se muestra el panel a partir del cual controlamos el valor al que enviamos el PWM que varía entre el mínimo, cero, y el máximo, 255 que sería el valor máximo de potencia que el driver podría aportar al motor.

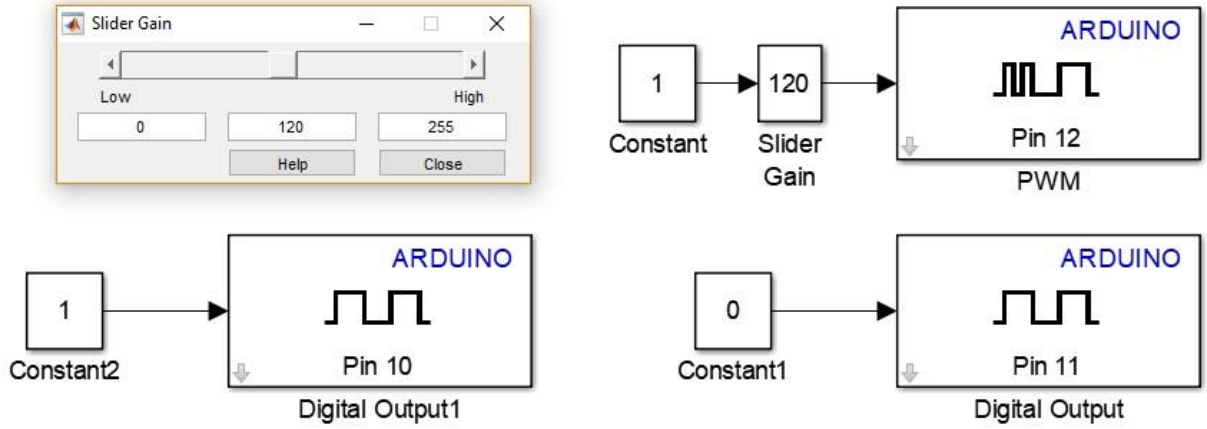

Figura 27. Paso del PWM a Arduino a través de Simulink.

4.3 Ejecución del programa en Arduino

Para cargarlo y ejecutarlo en la placa Arduino, se deben de seguir los siguientes pasos:

En primer lugar se debe pinchar en Tools -> Run on Target Hardware -> Options

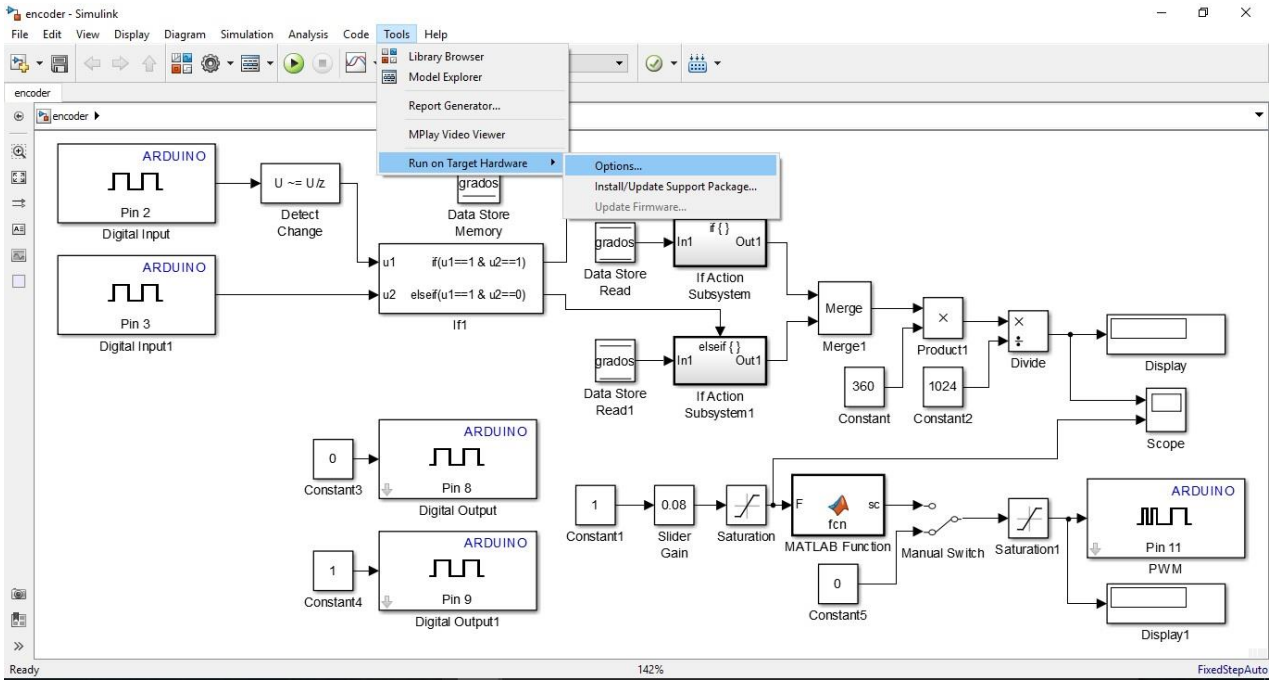

Figura 28. Preparación del modelo para funcionar en Arduino Mega 2560.

Una vez se ha abierto la ventana Configuration Parameters, se debe pinchar en Hardware Implementation que aparece en el menú de la izquierda, y en el menú desplegable de Hardware board se debe seleccionar la placa Arduino que se vaya a usar, en nuestro caso Arduino Mega 2560.

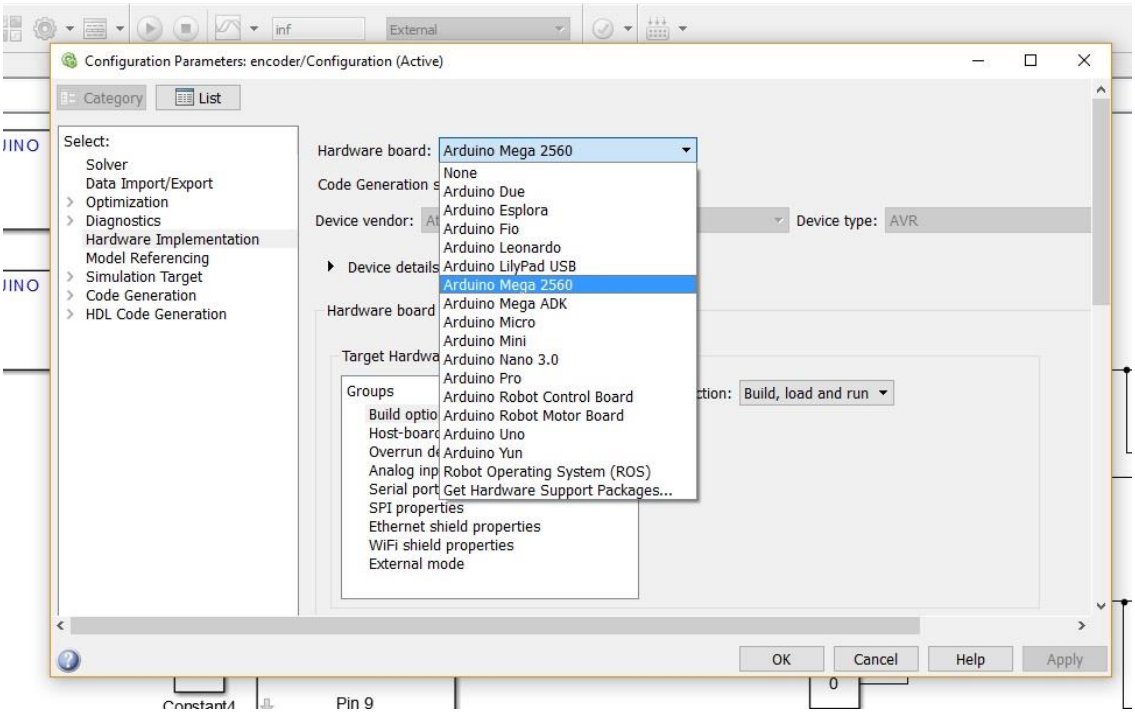

Figura 29. Selección del Hardware Arduino Mega 2560.

Una vez seleccionado, en el menú de Hardware board settings, se selecciona Host-board conections, y en el menú desplegable que aparece en Set host COM port se debe seleccionar Manually, y escribir el número de puerto al que está conectado la placa Arduino en COM por number.

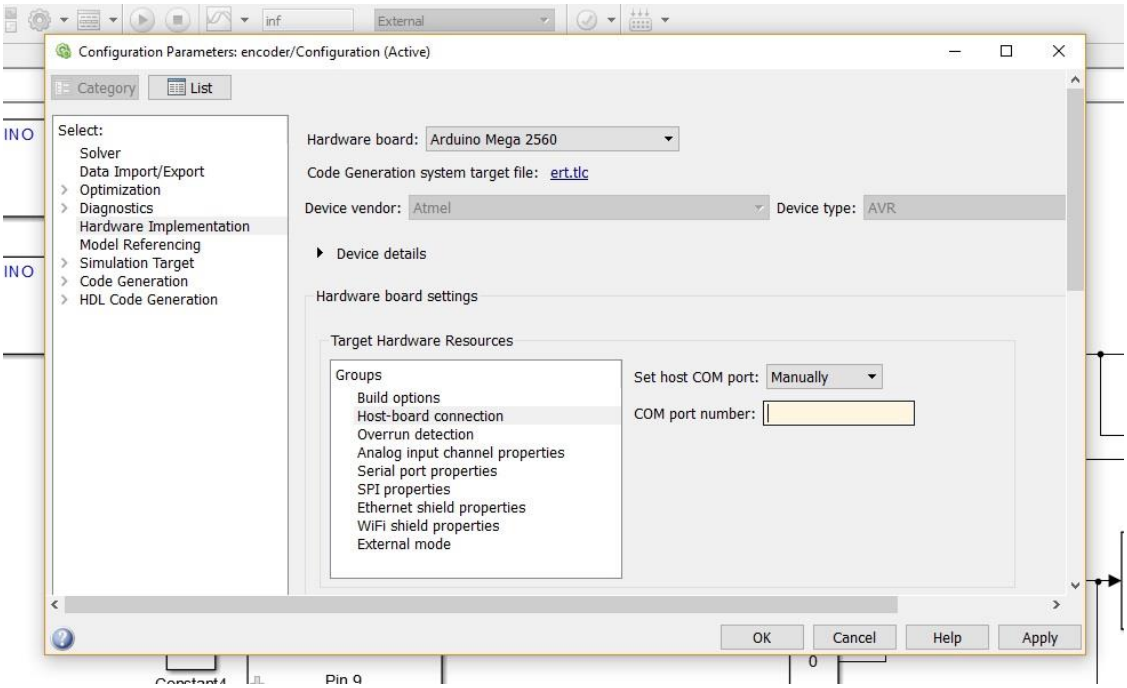

Figura 30. Selección del puerto USB al que se conecta Arduino.

A continuación se debe pinchar en el menú de la derecha en el apartado Solver, y una vez ahí, se debe escribir en Stop time: inf para que el tiempo de ejecución del programa sea el necesario hasta que decidamos pararlo, en Solver options -> Type se selecciona Fixed-step y en Solver options -> Solver se selecciona discrete (no continuous states), ya que de este modo se podrá ir variando el tiempo de muestreo a nuestro antojo como se explicará en el siguiente paso.

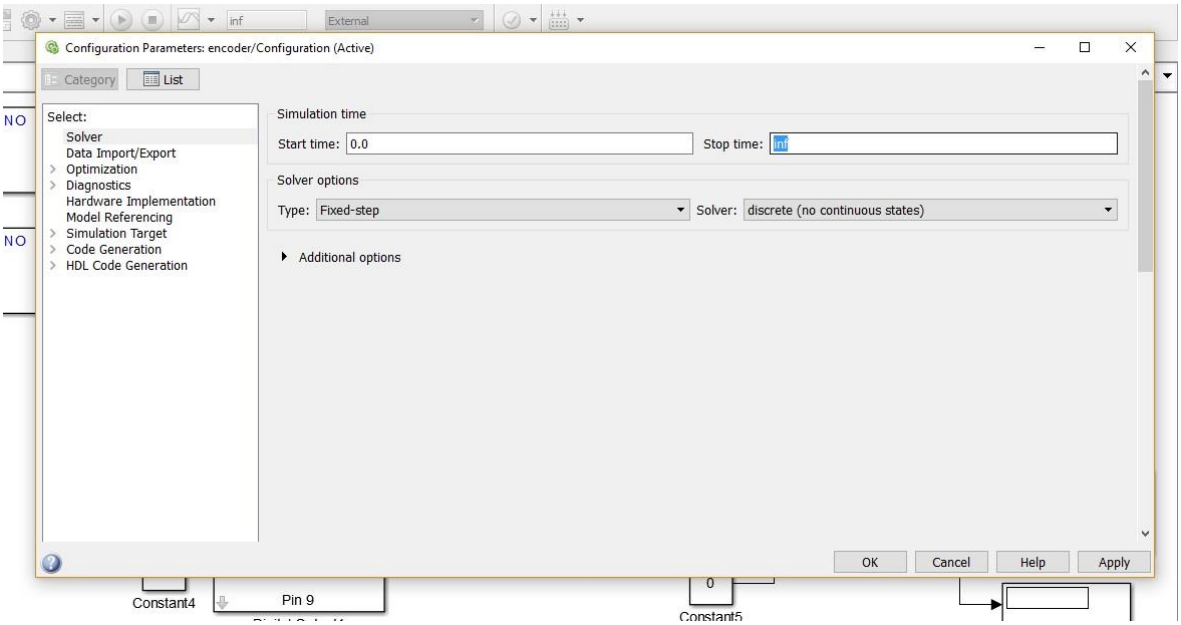

Figura 31. Selección del tipo de Solver.

Una vez realizado el paso anterior se debe pinchar sobre la pestaña Additional options, dónde en Fixed-step size se podrá seleccionar el tiempo de muestreo deseado, en este proyecto se emplea un tiempo de muestro de 0.001 segundos. Una vez realizados todos estos pasos se debe pinchar en OK para aplicar los cambios realizados.

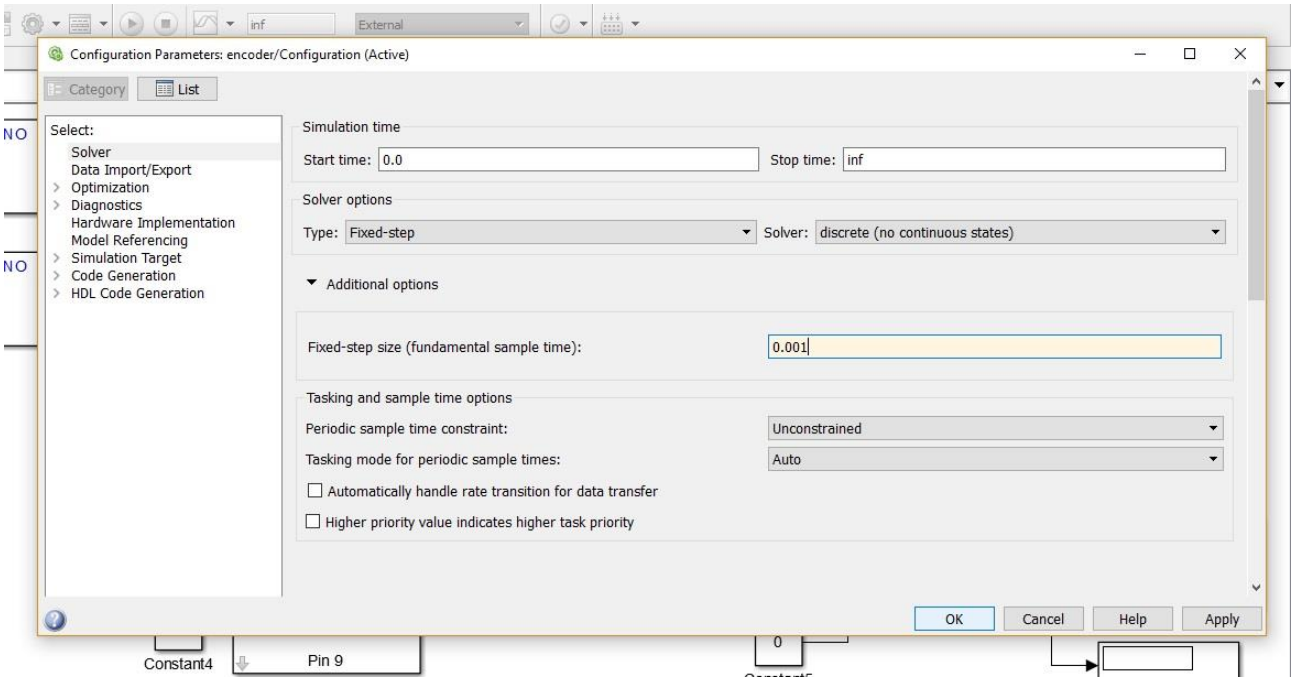

Figura 32. Selección del tiempo de muestreo.

Una vez que nos encontramos en el modelo, y antes de comenzar la ejecución se debe seleccionar External la pestaña desplegable dónde se encuentran todos los modos de ejecución.

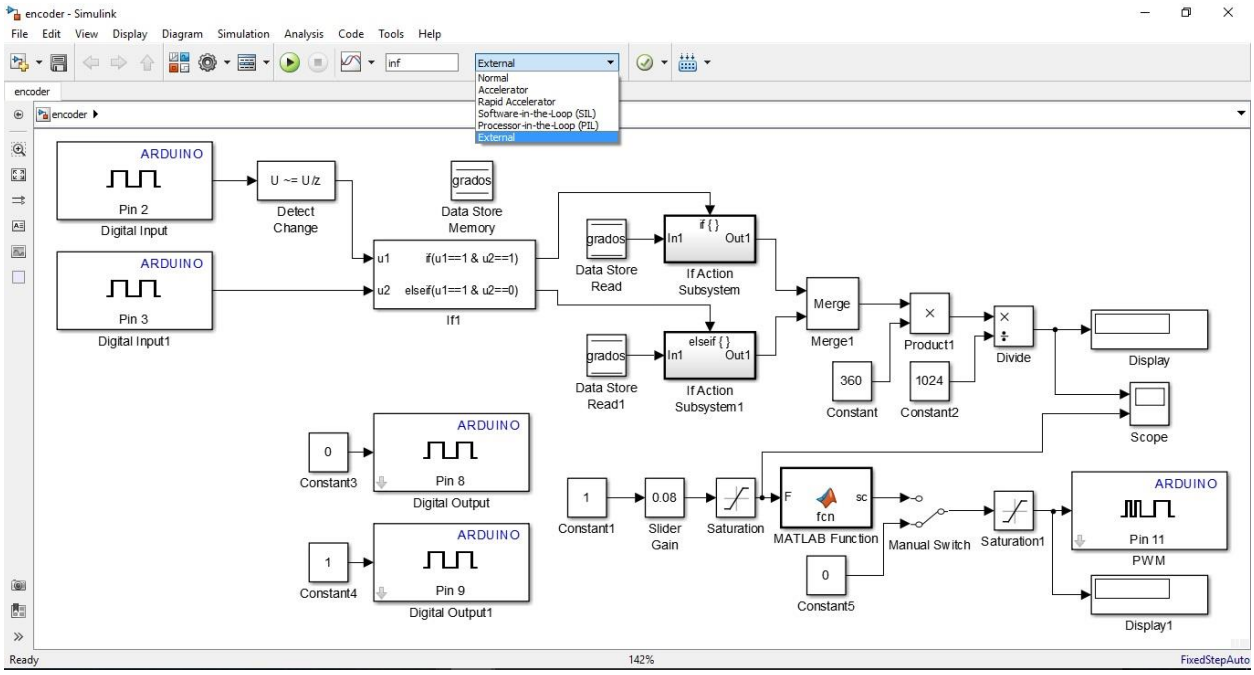

Figura 33. Selección del modo External.

Llegados a este punto, el modelo está preparado para ser ejecutado en nuestra placa Arduino Mega 2560, por lo que para comenzar la ejecución solo debemos pinchar en el botón de "run" que se encuentra indicado en la figura.

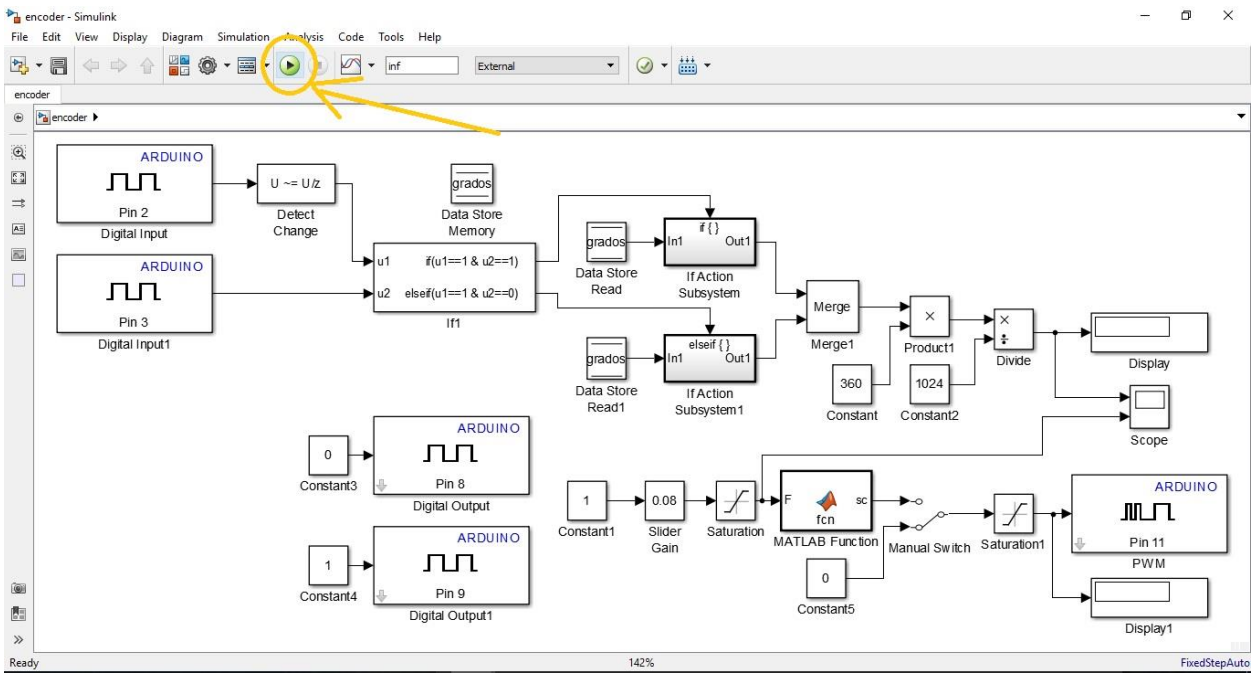

Figura 34. Comienzo de la simulación.

4.4 Modelo de recepción de datos de la IMU

En capítulos anteriores se han indicado las características de los sensores de medidas inerciales o IMU. Estos proporcionan medidas de aceleración y velocidad angular medidos por un conjunto de acelerómetros y giróscopos. Tanto el acelerómetro como el giróscopo miden los datos en tres ejes perpendiculares entre sí.

Para poder conectar la unidad de medidas inerciales a nuestra placa Arduino, se han realizado las conexiones que se detallan en la siguiente figura.

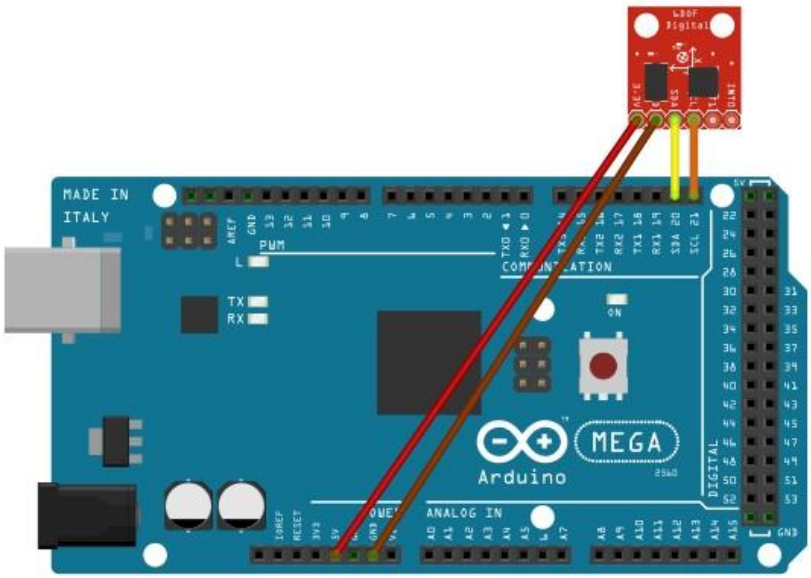

Figura 35. Esquema de montaje de la Unidad de Medidas Inerciales.

Hay que tener en cuenta a la hora de realizar las conexiones que el protocolo de comunicación es I^2C que emplea dos pines que son los que conectamos a los pines 20 (SDA) y 21 (SCL) de nuestra placa, además se emplean dos pines para comunicación (5V y GND).

4.4.1 Instalación de la librería RASPlib

A la hora de usar la IMU con la biblioteca de Arduino, existe el problema de que no existen bloques para usarla directamente, lo cual es una desventaja, ya que por ejemplo en el programa de Arduino existe la posibilidad de descargar una librería que puede usar unos comandos predeterminado lo cual facilita bastante la tarea, sin embargo, por internet se pueden encontrar diferentes librerías las cuales nos facilitan bastante realizar la tarea que deseamos implementar.

1- Desde [1] se puede descargar la biblioteca correspondiente a nuestra versión de MATLAB, de la que a continuación se explicarán en detalle los pasos para su instalación.

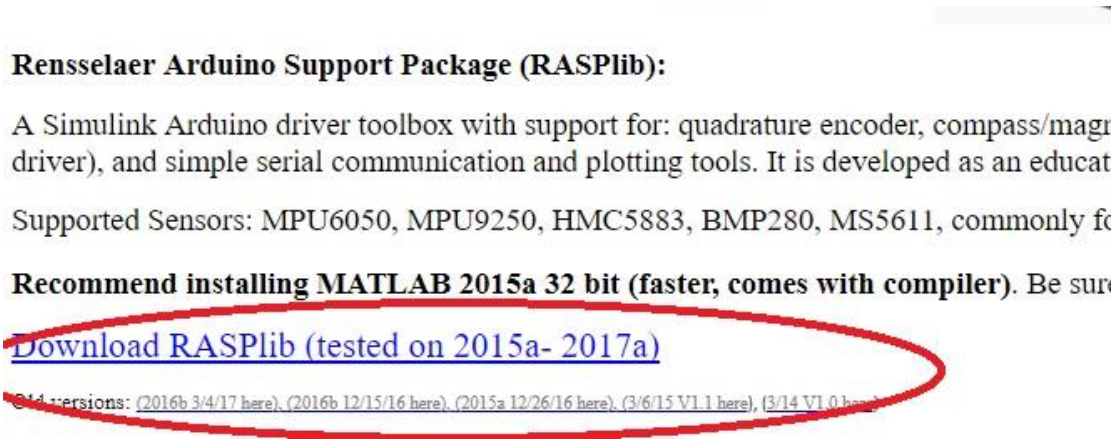

Figura 36. Descarga de la librería RASPlib.

2- Una vez que se ha descargado el archivo, se debe descomprimir, y dentro de la carpeta que aparecerá, habrá varias carpetas, archivos pdf y de otros tipos, sin embargo, solo de debe copiar la carpeta de nombre RASPlib.

Figura 37. Archivos incluidos en la librería.

3- A continuación se debe abrir Matlab y en la ventana de comandos, escribir "pwd" para ver cual es la dirección de la carpeta central, dónde se guardan los archivos por defecto

Figura 38. Carpeta central de Matlab.

4- Una vez que se sabe la dirección de la carpeta central se debe abrir en el explorador de archivos, y sobre ella se debe pegar la carpeta que se había copiado anteriormente "RASPlib". Además se debe crear un nuevo documento de texto cuyo nombre debe de ser "startup.m" y cuyo contenido debe de ser "addpath'[dirección de la carpeta central]\RASPlib'" como se puede observar en la figura.

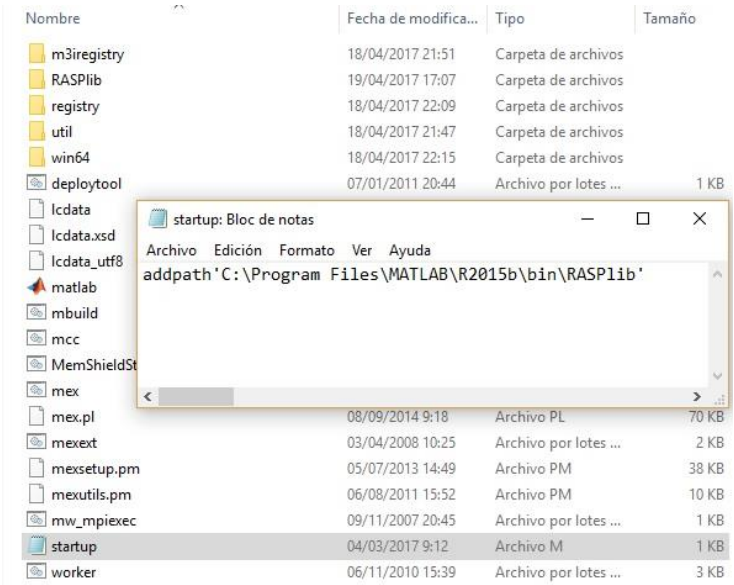

Figura 39. Instalación de la biblioteca.

4.4.2 Medición del ángulo con giroscopio

Una vez instalada la librería RASPlib, esta estará disponible cada vez que se abra Matlab. El siguiente paso que se va a seguir es emplear el giroscopio en un primer intento por obtener la medición angular.

Para ello a partir de la librería montamos el siguiente modelo en Simulink, con el bloque "Gyroscope", que será el encargado de tomar las medidas de velocidad en cada uno de los ejes del MPU6050.

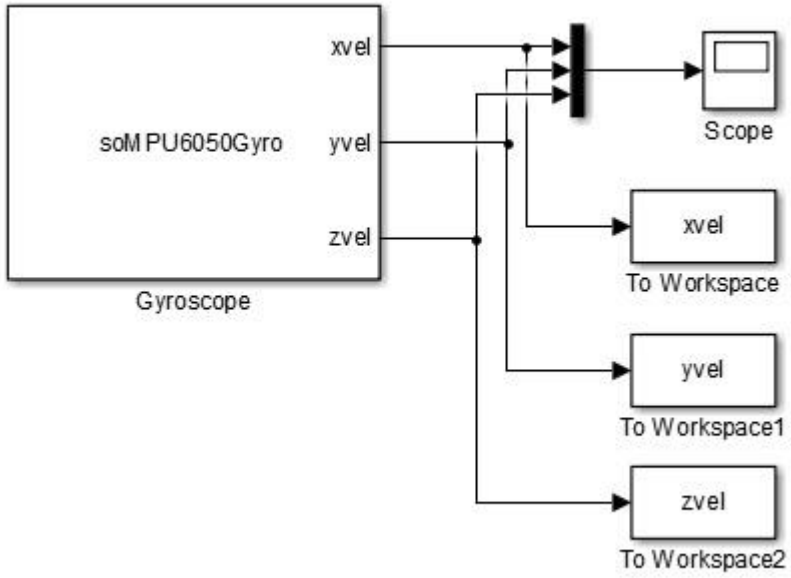

Figura 40. Modelo de recepción de datos con el giroscopio.

De este modo, se estarán representando en una misma gráfica del Scope las velocidades registradas por la IMU en cada eje, a la misma vez que se están enviando al "Workspace" de Matlab. Una vez que se realiza el experimento, obtenemos una gráfica con los siguientes valores.

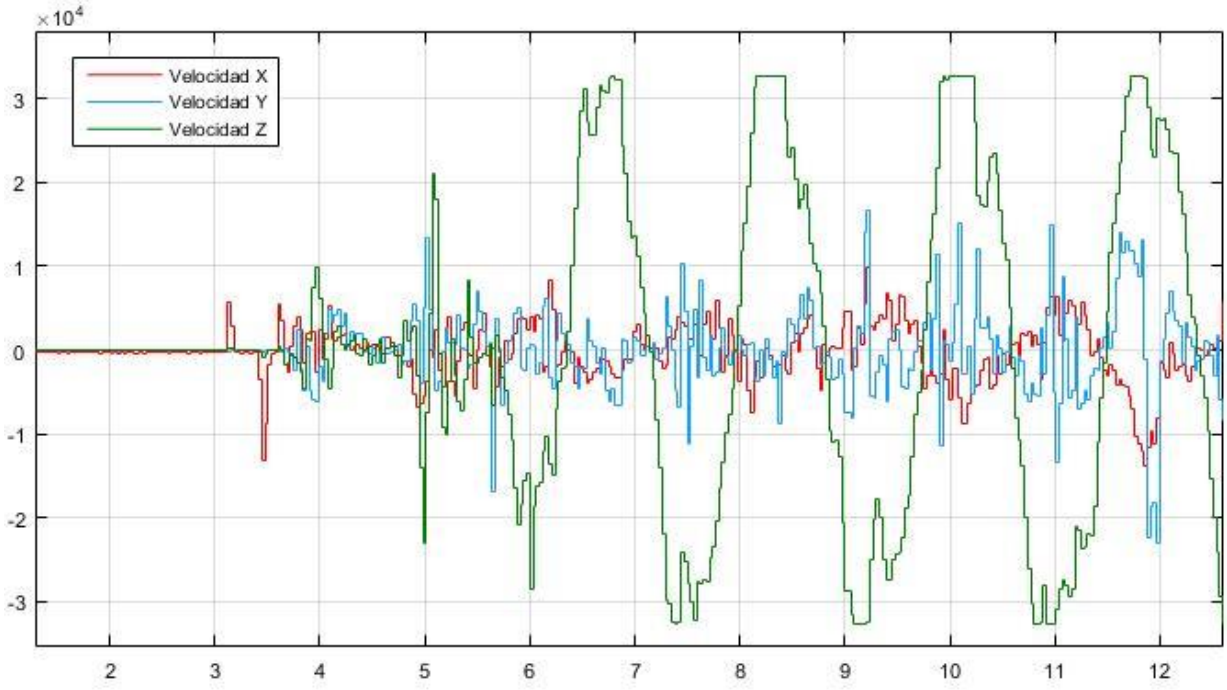

Figura 41. Velocidad obtenida en cada eje a partir del giroscopio.

En esta gráfica, se puede observar que la mayor variación de la velocidad al realizar una oscilación se produce en el eje Z de la unidad de medidas inerciales, es por eso que se debe tomar la velocidad en este eje como base para obtener la medida angular.

El siguiente paso que se va a realizar es la obtención de la posición angular a partir de la velocidad en uno de los ejes (el eje z en este caso). Para ello se sabe que la velocidad es la derivada de la posición, o dicho de otra manera, la posición es la integral de la velocidad, por tanto integrando la velocidad a partir del bloque Discrete-Time Integrator de Simulink, podemos obtener la posición angular del sistema en todo momento. En la siguiente figura se muestra el diagrama de bloques necesario para realizar estos cálculos.

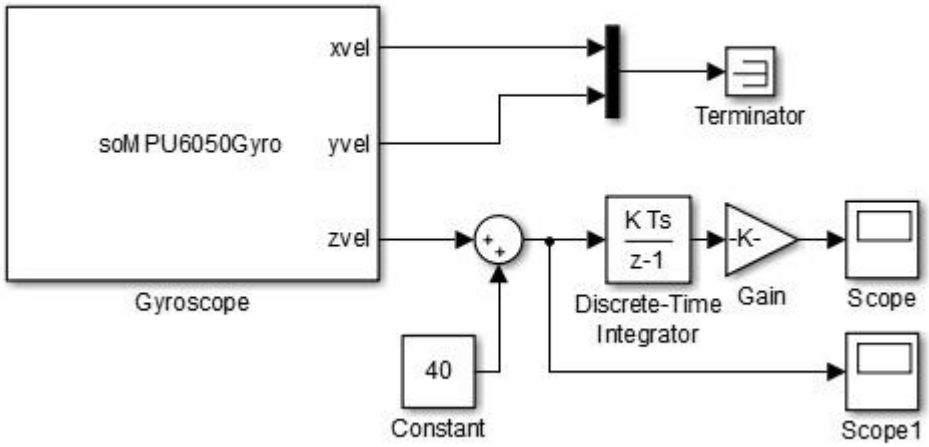

Figura 42. Modelo que transforma las velocidades del giroscopio a ángulos.

Se puede observar que solo se toman las medidas de velocidad del eje z, y además se pueden ver el integrador en tiempo discreto. Antes de la entrada del integrador se le suma una constante cuyo valor es de 40 a la velocidad de salida en el eje z, proporcionada por el giroscopio, para evitar errores de integración, ya que la medida se encuentra levemente desfasada. Finalmente, la salida del integrador se multiplica por una ganancia, para escalar el valor a grados. Una vez está todo listo, se deja correr el modelo, mientras se realizan movimientos del sistema, obteniéndose los siguientes valores.

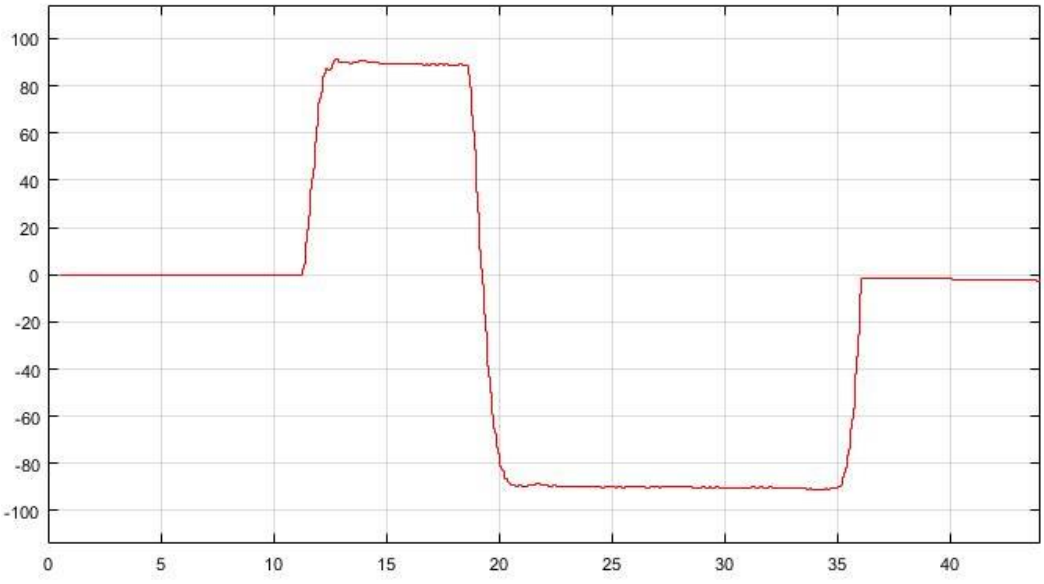

Figura 43. Medida del ángulo a partir del giroscopio.

En la figura anterior, se puede ver como se obtienen unos valores bastante fiables de la posición angular en un rango entre 90º y -90º, sin embargo, a la larga este modelo irá acumulando errores poco a poco debido al integrador, por lo que llegará un punto en el que las medidas que se obtengan no sean fiables. Es por esto que se debe de encontrar otro método que evite usar un integrador para evitar la acumulación de errores y poder garantizar el correcto funcionamiento del sistema.

4.4.3 Medición del ángulo con el acelerómetro

El MPU6050 cuenta con un acelerómetro que nos propociona valores de aceleración en cada uno de sus tres ejes. Para calcular la posición angular a partir de las medidas de aceleraciones, se necesita saber lo siguiente.

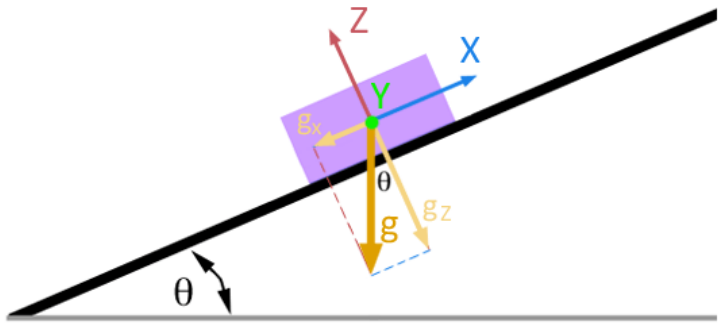

Figura 44. Descomposición de aceleraciones del acelerómetro.

En la figura podemos observar como a partir de las medidas de la aceleración de la gravedad que nos proporciona la unidad de medidas inerciales, se puede descomponer en dos componentes, de los ejes x e y en nuestro caso, para calcular la inclinación de la IMU respecto a su eje z.

Una vez que se conoce lo anterior, se puede calcular el ángulo respecto al eje z gracias a una simple fórmula trigonométrica:

$$\theta = \tan^{-1}\frac{a_x}{a_y}$$

Esto es todo lo necesario para poder calcularla, a continuación se monta el modelo en Simulink que realice y represente este cálculo. Para ello emplearemos el bloque "Trigonometric Function" que podemos encontrar en la biblioteca Math Operations seleccionando la opción atan2. El modelo quedaría de la siguiente manera.

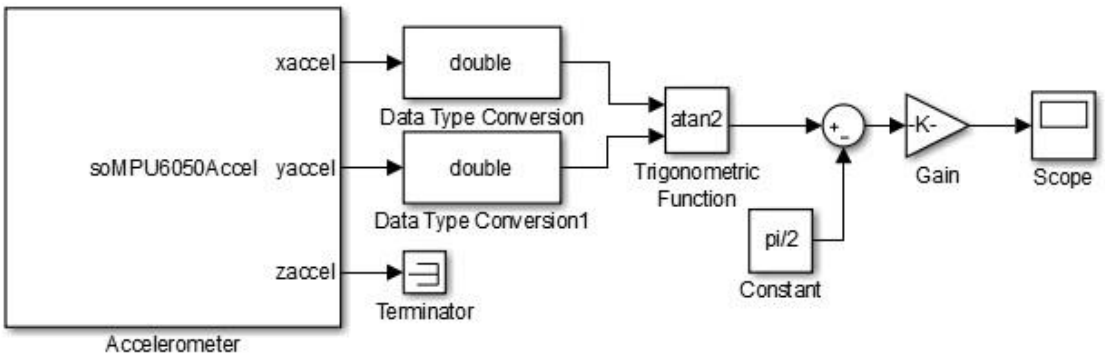

Figura 45. Modelo de recepción de datos del acelerómetro.

Como Matlab y Simulink por defecto trabajan en radianes, tenemos que multiplicar la salida por una ganancia para pasarla a grados, además antes se le resta pi/2 para corregir un desfase que el acelerómetro incluye por defecto. Una vez se realizan estos arreglos, se realizan los experimentos pertinentes para ver si la medición del ángulo a partir de las aceleraciones funciona de manera óptima.

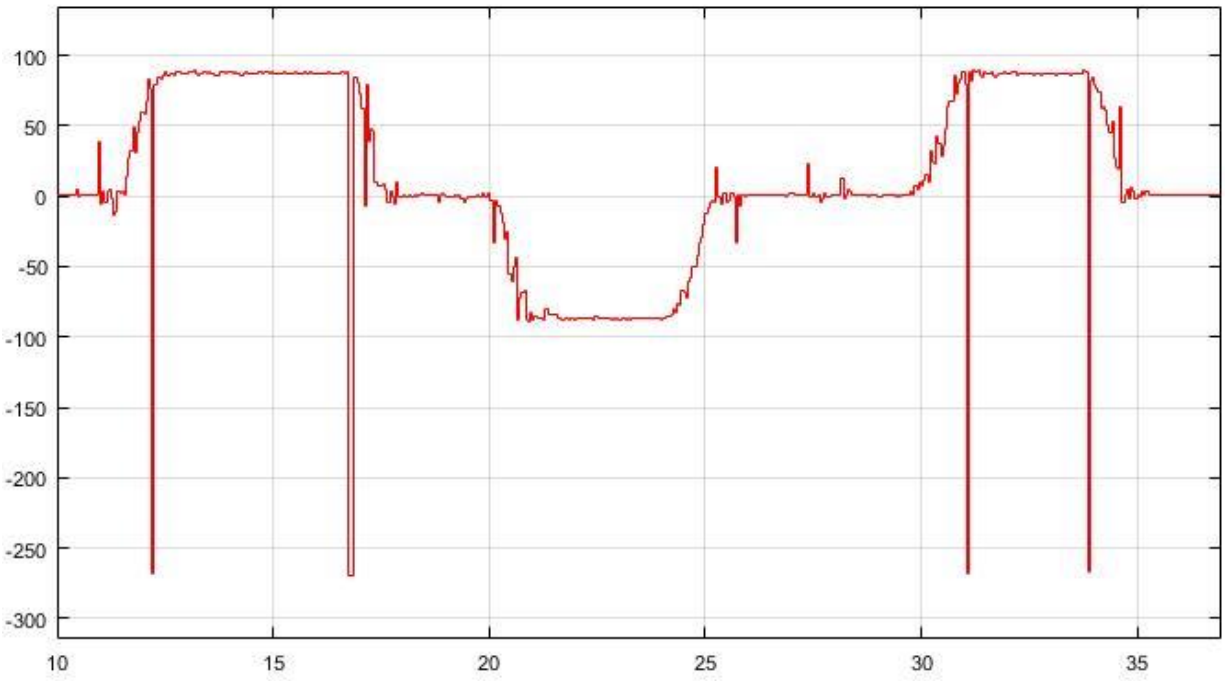

Figura 46. Medida del ángulo a partir del acelerómetro.

En esta gráfica se puede observar como la medida realizada por el modelo es adecuada, sin embargo se puede ver que existen unos picos, debidos a que si aparecen otro tipo de aceleraciones en la unidad de medidas inerciales, diferentes a la de la gravedad (que pueden ser debidas a algún golpe, vibración, etc.) generará este tipo de picos, lo cual es inapropiado para el control del sistema ya que la estimación deja de ser correcta durante un leve instante de tiempo pero de manera drástica.

Por ello una buena opción para corregirlo es aplicar el filtro complementario, el cual toma la posición angular medida a partir de la velocidad y la medida a partir de la aceleración, lo que ayuda a corregir este error.

4.4.4 Medición angular con el filto complementario

En este apartado se va a obtener la posición angular a partir del filtro complementario, el cual puede expresarse como:

$$\theta = A * \left(\theta_{prev} + \theta_{gyro}\right) + B * \theta_{acel} \qquad (4.1)$$

Donde A y B son dos constantes que, inicialmente, pueden tomarse como 0.98 y 0.02 respectivamente, pero que se irán calibrando hasta obtener una medida óptima del filtro, siempre cumpliendo la condición de que la suma de las dos sea uno.

El filtro complementario se comporta como un filtro de paso alto para la medición del giroscopio y un filtro de paso bajo para la señal del acelerómetro. Es decir, la señal del giroscopio manda a corto plazo, y la del acelerómetro y medio y largo, que es exactamente lo que se desea para compensar sus ventajas y defectos.

En la mayoría de aplicaciones, el filtro complementario es suficiente y proporciona valores muy similares al filtro Kalman, siempre que los valores de A y B esten debidamente calibrados.

Para implementarlo, se debe realizar el siguiente modelo en Simulink.

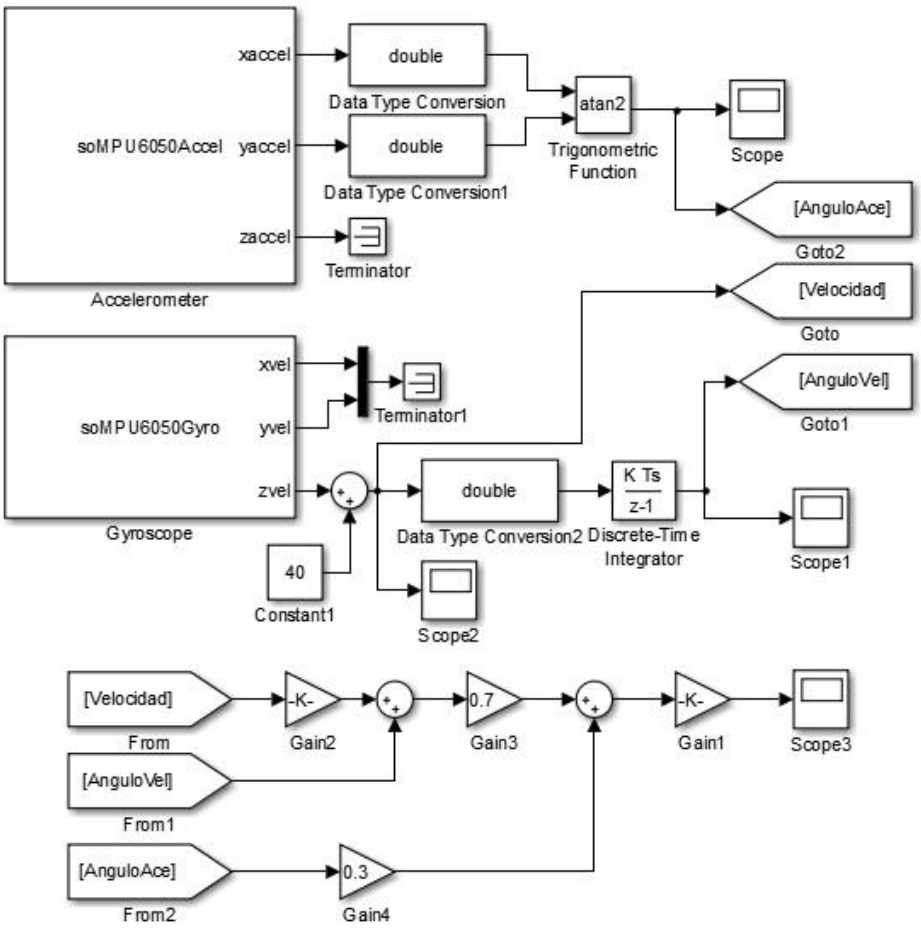

Figura 47. Modelo utilizado para implementar el filtro complementario.

Dónde como se ha explicado anteriormente se combinan las mediciones de posición a partir de velocidad angular y de aceleración angular, y dónde se le han dado unos valores a los parámetros A y B de 0.7 y 0.3 respectivamente.

De esta manera, al combinar las dos mediciones, se solucionan por un lado los errores de integración, y por otro lado las aceleraciones indeseadas, lo que lleva a obtener resultados como los que se muestran en la siguiente gráfica.

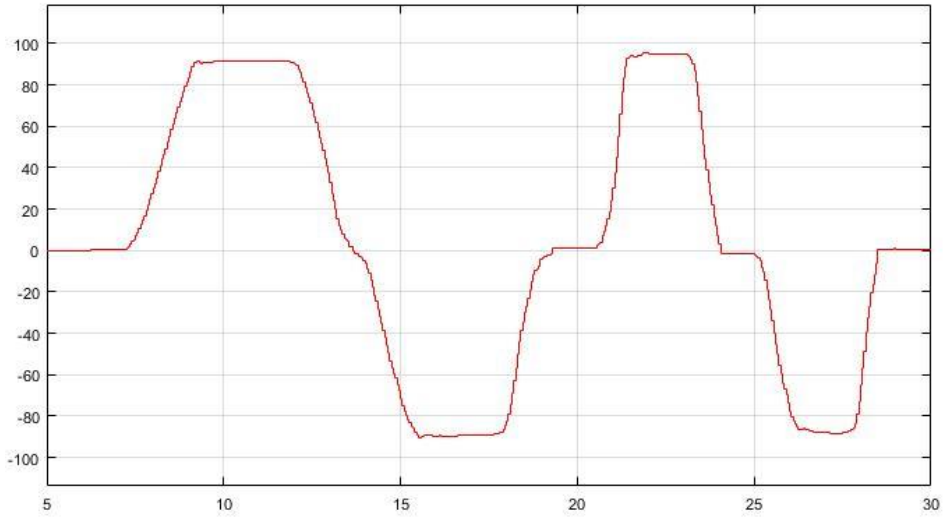

Figura 48. Medida del ángulo obtenida a partir del filtro complementario.

No obstante, a la hora de implementar en el Sistema real la IMU, al encender el motor, se producen una serie de vibraciones debida a este que hacen que las medidas aportardas tanto por el giroscopio como por el acelerómetro sean ilegíbles como se puede 33cceptab en la siguiente figura, dónde a partir de los 57 segundos aproximadamente de enciende el motor, obteniendo unas medidas a partir de las cuales es muy complicado realizar una estimación 33cceptable de la posición, por lo que será necesario implementar otro sensor de posición diferente como se verá en el siguiente apartado.

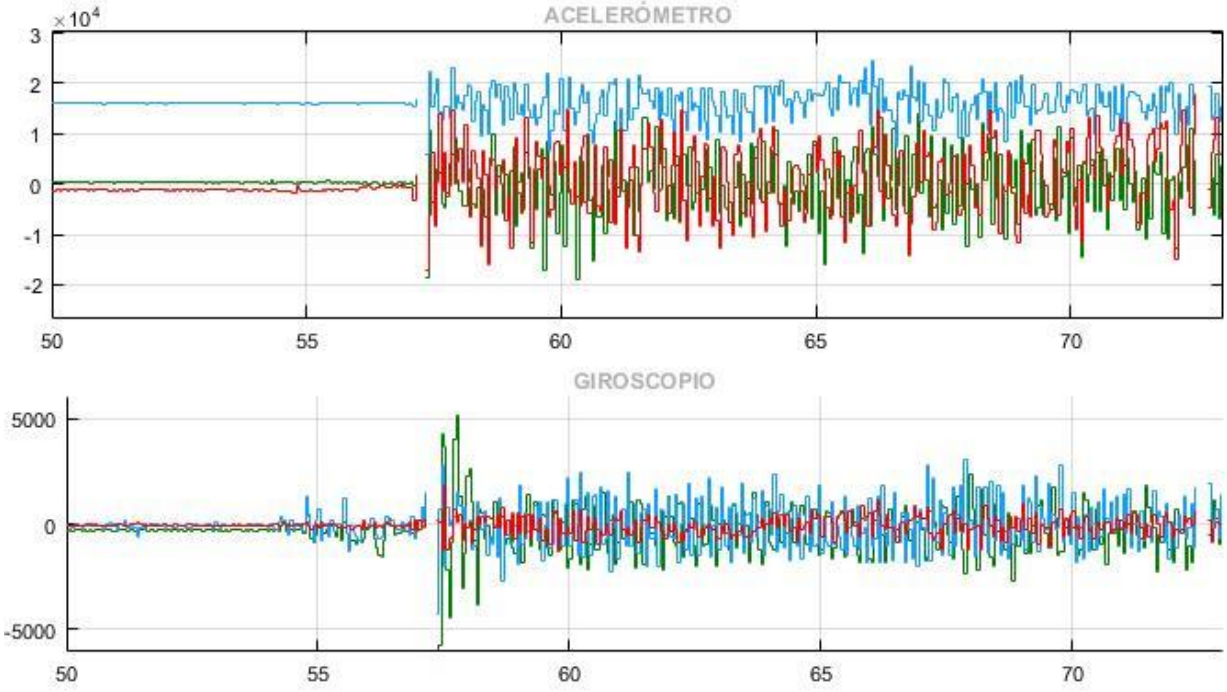

Figura 49. Señal del acelerómetro y el giroscopio con el motor apagado y encendido.

4.5 Modelo de recepción de datos del encoder

La medición de la posición usando un encoder incremental es relativamente sencilla y su precisión dependerá de la precisión y calidad del mismo. Como se ha explicado anteriormente, un encoder proporciona a través de cada uno de sus canales una señal digital, en nuestro caso contamos con un encoder de 512 pulsos por revolución y de tres canales, un primer canal que emite un pulso digital cada 512/360º de movimiento del eje del encoder, un segundo canal en cual marca el sentido de giro, y un tercer canal llamado index que emite un pulso por cada revolución completa.

La gran ventaja respecto a la IMU es que no se ve afectado en gran medida por las vibraciones que introduce el movimiento del motor y permite obtener valores de posición más que fiables.

En cada revolución completa del encoder podremos contar 512 pulsos, con un flanco de subida (transición de 0 a 1) y un flanco de bajada (transición de 1 a 0) cada uno. Para determinar el ángulo girado por el encoder se cuenta el número de flancos de subida (o de bajada) generados y la dirección vendrá determinada, en cada pulso, por el estado del otro canal, de tal forma que si está a 0 determinamos que va en una dirección y si es 1 irá en la dirección contraria.

Para poder conectar el encoder que cuenta con 6 pines, de los cuales conectamos 5 a nuestra placa Arduino, ya que en nuestro caso como no se va a necesitar contabilizar el número de revoluciones por lo que se deja sin conectar el canal index, se realizan las siguientes conexiones.

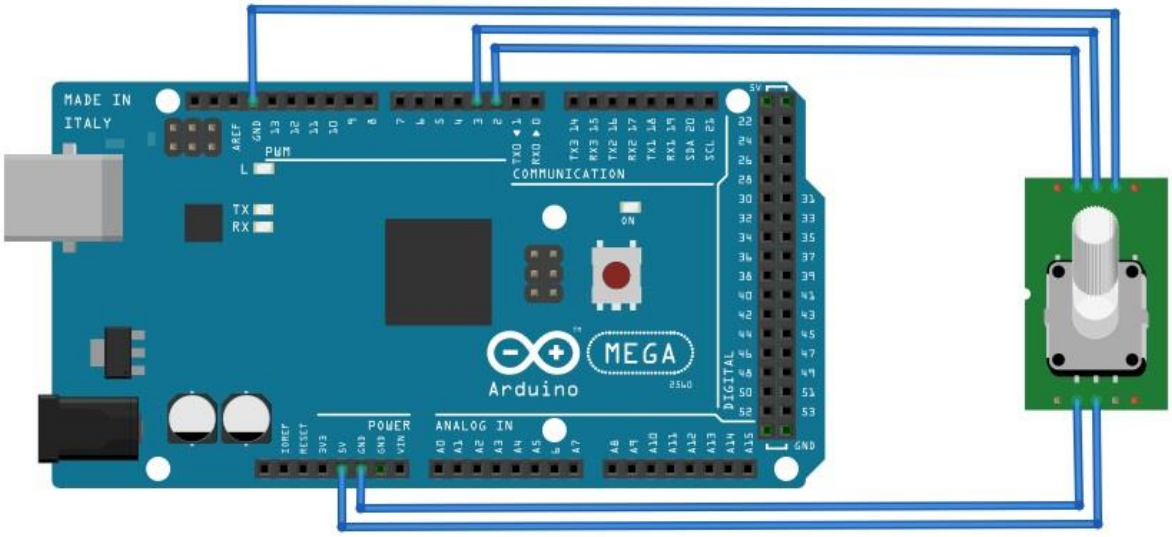

Figura 50. Esquema de montaje del encoder.

El encoder se alimenta con 5V, tiene un pin de tierra que va al pin GND de Arduino, además también conectamos el pin 6 (CS) del encoder a la tierra de la placa, ya que en el datasheet se indica que si no va a ser usado, sea conectado a tierra. Por otro lado como hemos indicado anteriormente el pin 4 (index) se deja sin conectar, y los pines 1 (pulso) y 2 (dirección) se conectan a las entradas digitales 2 y 3 de la placa.

Normalmente para gestionar el encoder a nivel de software se crea una variable que se va incrementando o decrementando según el sentido de giro en cada pulso, de tal forma que esta variable contiene la posición relativa del eje del encoder, si conocemos la posición inicial del motor (en nuestro caso siempre se considerará como 0), cuando se encendió el dispositivo, entonces sabremos la posición absoluta. El siguiente programa de Simulink implementa a través de bloque las rutinas necesarias para controlar el encoder, incrementando o decrementando la variable Grados.

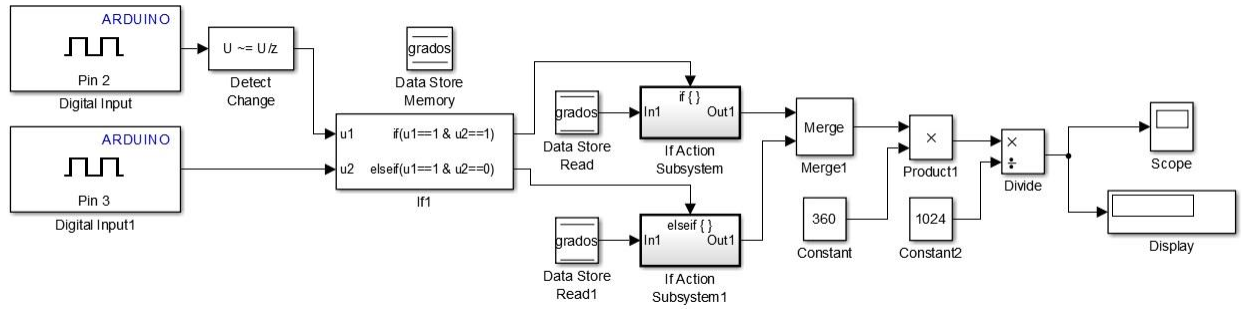

Figura 51. Modelo de recepción de datos del encoder.

Como vemos en el modelo, a través de los bloques Digital Input se recogen los pulsos de los pines 2 (canal de pulso) y 3 (canal de dirección), a continuación el bloque Detect Change se detectan los flancos de subida o de bajada de los pulsos, lo que lleva a un bloque if el cual si el pin 3 está a nivel alto aumenta en una unidad el valor de la variable Grados, o la decrementa en caso contrario. Ambas señales se unen a través del bloque Merge, esta cuenta incluida en la variable, se multiplica por 360 y se divide entre 1024 (ya que contamos los flancos de subida y de bajada, si se contara solo uno de los dos se debería dividir entre 512 que es la precisión del encoder), para pasar el valor a unidad de grados, finalmente el valor de la señal se lleva a un bloque Scope para graficarlo y a un bloque Display para mostrar el valor pon pantalla en tiempo real.

En la siguiente figura observamos la medida de la posición angular dada por el modelo a partir de los datos aportados por el encoder a través de sus dos canales, a medida que vamos variando la señal de control (PWM).

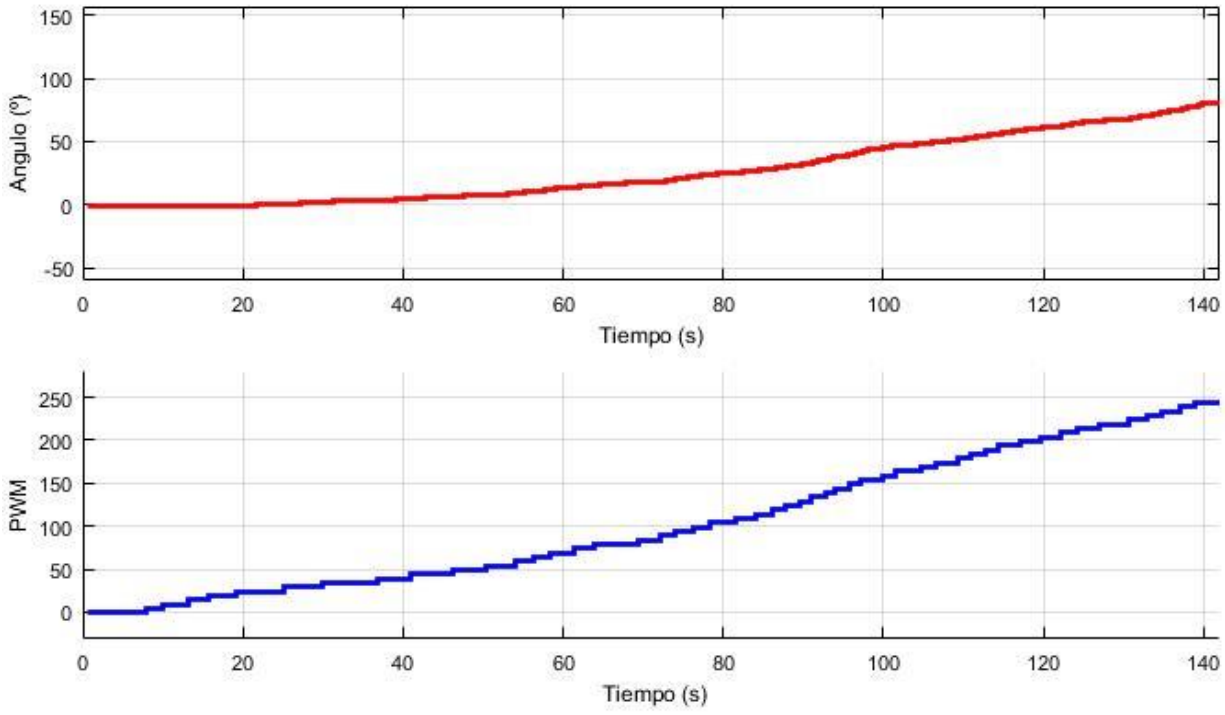

Figura 52. Medida aportada por el encoder a partir de la variación del PWM.

Con los resultados obtenidos de las pruebas con ambos sensores se ha determinado que el encoder es el mas preciso y fiable en los resultados, ya que entre otras cosas no es sensible a las vibraciones al encender el motor como la unidad de medidas inerciales.

5 CONTROL DEL SISTEMA

En este capítulo se trata todo lo relacionado con el control del sistema y los pasos seguidos para la implementación de los controladores seleccionados. En primer lugar se obtiene la gráfica de la característica estática, gracias a la cual se obtiene una ecuación de primer grado que relaciona la señal de control (PWM) con la fuerza proporcionada por el giro de la hélice. A continuación se detallan los pasos seguidos para la identificación de la función de transferencia y la implementación de diferentes controladores junto con los resultados obtenidos.

5.1 Característica estática

La característica estática mide la relación entre la entrada y la salida en regimen permanente, es decir cuando las derivadas de la función que define al sistema se consideran nulas. En esta planta, se tiene como salida el ángulo medido por el encoder y como entrada la fuerza aplicada por la hélice. Esta fuerza no se conoce inicialmente, ya que a priori, lo único que se conoce es el rango del valor de la señal de control que hay que introducer en el sistema para que se mueva el motor. Esta fuerza depende de la forma de la hélice, velocidad de giro o densidad del aire, que son datos complicados de medir y por tanto es un parámetro dificil de calcular.

La teoría de mecánica de fluidos dice que la fuerza que de sustentación que genera un perfil aerodinámico que se desplaza dentro de un fluido, queda definido por la ecuación:

$$L = \frac{1}{2}\rho V^2 S C_L \qquad (5.1)$$

dónde:

- L: sustentación del perfil.
- ρ: densidad del fluido, el aire en nuestro caso.
- V: velocidad relativa del fluido respecto a la hélice.
- S: superficie del perfil.
- C_L: coeficiente de sustentación del perfil.

Se podría aplicar dicha ecuación en este caso, ya que una hélice no es más que un perfil aerodinámico que se mueve de forma giratoria, es decir, que tiene una velocidad respecto del aire, por lo que produce sustentación o empuje, no obstante esta ecuación cuenta con parámetros que son dificil de medir ya que dependen de multiples factores, es por esto que emplearemos un método experimental para calcularla.

La necesidad de conocer el valor de la fuerza que genera la hélice (en Newtons) lleva a obtener la característica estática de esta frente a la señal de control, de forma que quede definido el valor de la fuerza en función del valor de la señal de control, lo que permite que la entrada del Sistema se de en valores de fuerza o Newtons.

Para ello comenzamos midiendo la fuerza que ejerce la hélice a partir de un método experimental en que se mide directamente la fuerza que ejerce la hélice en movimiento en un peso de precision de un gramo.

Se coloca la planta de manera que el extremo opuesto del motor esté en equilibrio situado sobre el peso, para ello se ha tenido que colocar algo de peso sobre la barra para que esta se mantuviera en la posición de equilibrio, a continuación se inicializa a cero el peso, y se va incrementando la señal de control (PWM) de 5 en 5 unidades, partiendo desde cero y llegando hasta 255.

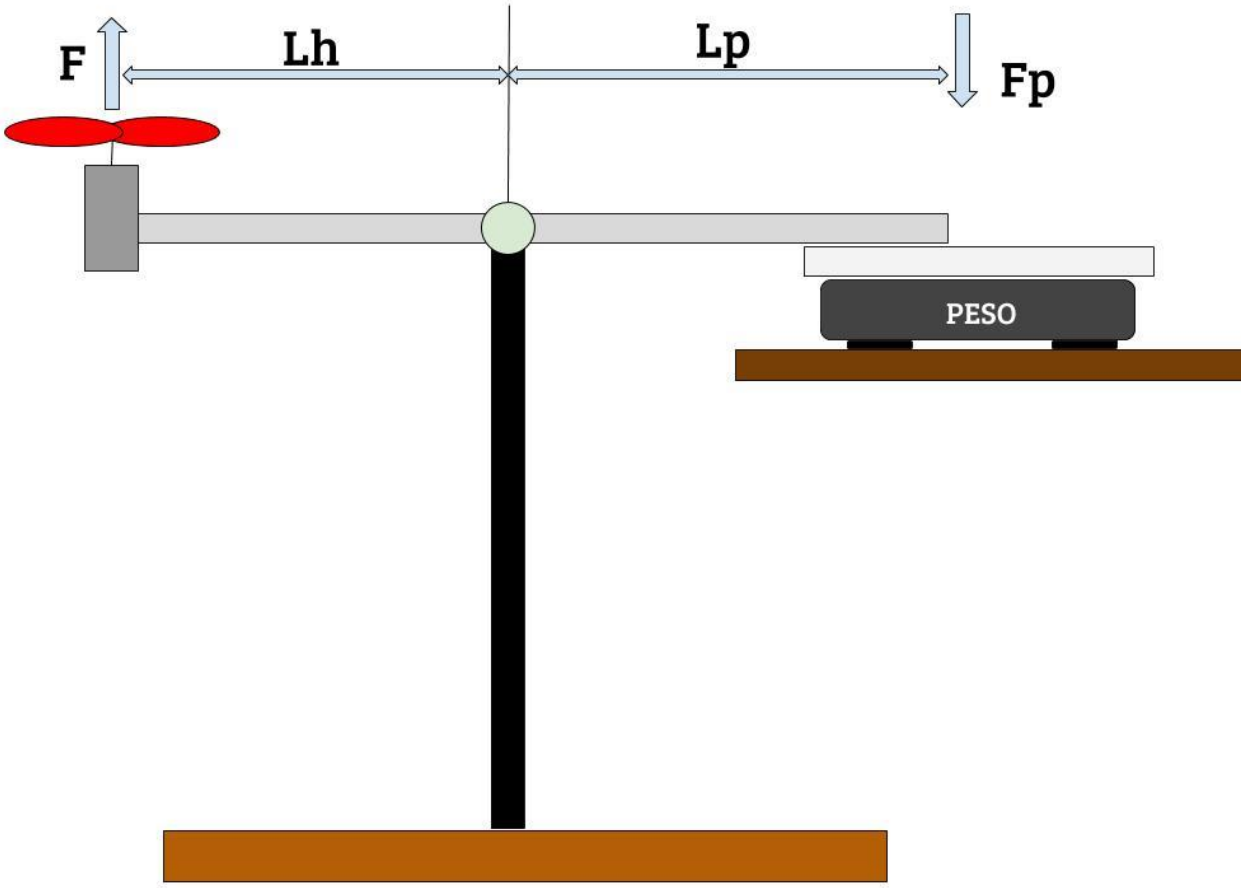

Figura 53. Método empleado para la medición experimental de la fuerza.

Al ser un sistema indeformable,toda fuerza que se genere hacia arriba en el extremo de la barra donde se encuentra la hélice es medida hacia abajo en el extremo opuesto con el peso. Dicha medida se obtiene con precisión de un gramo, por lo que habrá que pasarla a Newtons. Ya que el sistema está en equilibrio, el momento generado por la fuerza que genera la hélice se compensa por el momento generado por la fuerza de reacción que produce el peso, de tal forma que hay equilibrio de momentos e igualando la relación fuerza por distancia se obtiene que:

$$F.L_h = F_p.L_p \qquad (5.2)$$

En esta escuación, F es la fuerza de empuje ejercida por la hélice, L_h es la distancia desde el eje de giro del sistema hasta el eje de giro del motor, F_p es la fuerza medida por el peso, y L_p es la distancia desde el eje de giro hasta el peso. Despejando la incógnita F, la ecuación queda de la siguiente manera:

$$F = \frac{L_h}{L_p}.F_p \qquad (5.3)$$

Con lo que, conociendo las respectivas distancias al eje y el valor medido por el peso se puede obtener una medición aproximada del valor de la fuerza que genera la hélice. Teniendo en cuenta que se debe pasar el valor aportado por el peso de gramos a kilogramos, que las longitudes L_p y L_h son iguales (27 centímetros) y que el valor de la aceleración de la gravedad es de 9.81 $\frac{m}{s^2}$ la expresión a aplicar quedaría de la siguiente manera:

$$F\ [Newtons] = F[gramos] * Gravedad\left[\frac{m}{s^2}\right] * \frac{1kg}{1000g} = 9.81 * 10^{-3} * F[gramos] \qquad (5.4)$$

A continuación se muestran en la tabla los datos obtenidos, para ello se han usado tres columnas, en la primera se ha incluido el valor de PWM aplicado, en la segunda el peso en gramos obtenido por el peso y en la tercera el valor de fuerza correspondiente en Newtons.

Tabla 5-1. Medida experimental de la fuerza en función de la señal de control

PWM [0-255]	F_p [gramos]	F [Newtons]	PWM [0-255]	F_p [gramos]	F [Newtons]
0	0	0	130	15	0.147
5	0	0	135	16	0.157
10	0	0	140	17	0.167
15	0	0	145	19	0.186
20	0	0	150	19	0.186
25	0	0	155	20	0.196
30	0	0	160	20	0.196
35	3	0.029	165	21	0.206
40	3	0.029	170	22	0.216
45	4	0.039	175	22	0.216
50	5	0.049	180	24	0.235
55	5	0.049	185	26	0.255
60	6	0.059	190	26	0.255
65	7	0.069	195	27	0.265
70	8	0.079	200	28	0.275
75	9	0.088	205	28	0.275
80	10	0.098	210	28	0.275
85	10	0.098	215	29	0.285
90	11	0.108	220	29	0.285
95	11	0.108	225	30	0.294
100	12	0.118	230	31	0.304
105	12	0.118	235	32	0.314
110	13	0.128	240	32	0.314
115	13	0.128	245	32	0.314
120	14	0.137	250	35	0.343
125	14	0.137	255	37	0.363

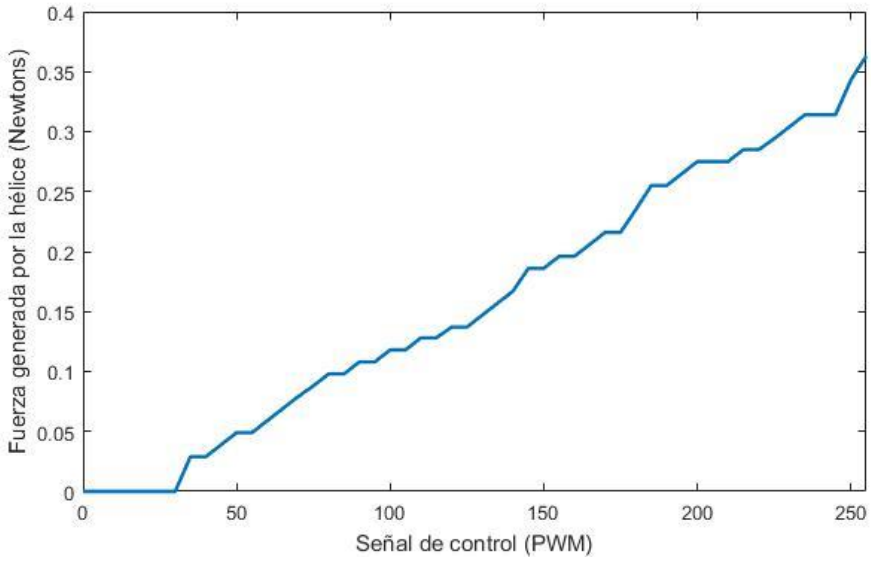

Figura 54. Fuerza generada por la hélice a partir de la variación del PWM.

En esta gráfica se puede observar como va aumentando la fuerza generada por la hélice de manera progresiva a medida que se aumenta la señal de control. En nuestra planta, como se ha dicho anteriormente, vamos a trabajar con una salida, que es el ángulo al que se encuentra el balancín con respecto a la vertical, y con una entrada, que será la fuerza aportada por la hélice. Es por esto que se necesita una ecuación que nos relacione la entrada real que hay que pasar al motor, que es el PWM y la fuerza generada por la hélice del motor, para ello será necesario realizar una gráfica en la cual en el eje de abcisas la fuerza y en el eje de ordenadas la señal de control.

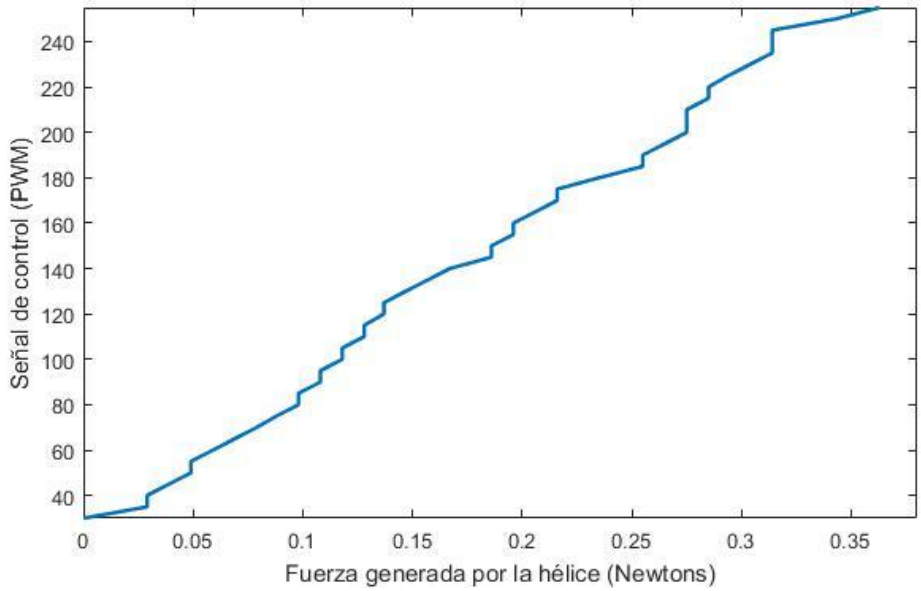

Figura 55. Variación del PWM frende a la fuerza generada por el perfil.

La opción Basic Fitting de Matlab permite obtener una ecuación que relacione dos variables entre sí a partir de una curva con datos de dichas variables. Esto permite obtener una función en la que la variable dependiente es la señal de control, y la variable independiente la fuerza que genera la hélice. Esta opción se encuentra en el

menú Tools de la barra de herramientasde la ventana que se abre al representar gráficamente una curva en Matlab, y da la posibilidad de elegir entre varios tipos de aproximación: por el interpolante tipo spline, por el interpolante que preserva la forma, o por aproximación polinómica de grado 1 hasta 10.

En este caso se tomara una aproximación de segundo grado, que será lo suficientemente exacta para poder transformar la fuerza deseada en cada instante a la señal PWM correspondiente.

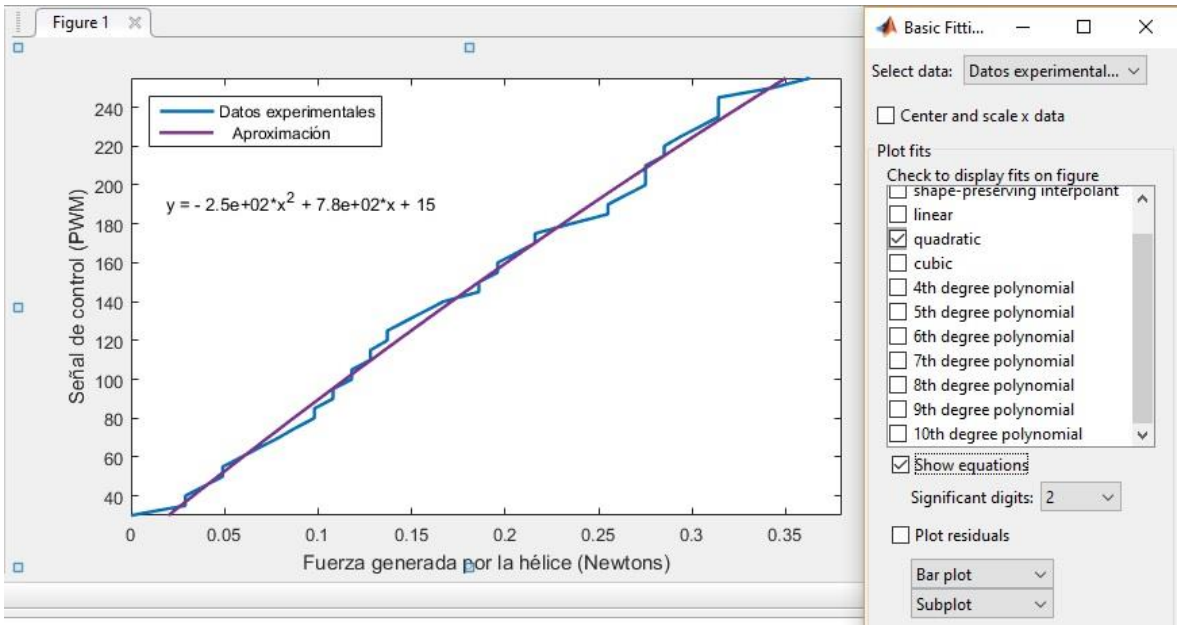

Figura 56. Opción Basic Fitting de Matlab.

A continuación se debe incluir esta ecuación en el modelo de Simulink dentro de un bloque "MATLAB function" que es el que se encargará de evaluar la ecuación cuando sea necesario y pasar el resultado correspondiente directamente al bloque de salida PWM que irá a la placa Arduino.

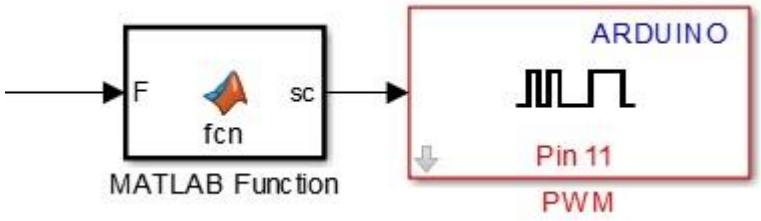

Figura 57. Bloque MATLAB Function conectado directamente a la salida PWM.

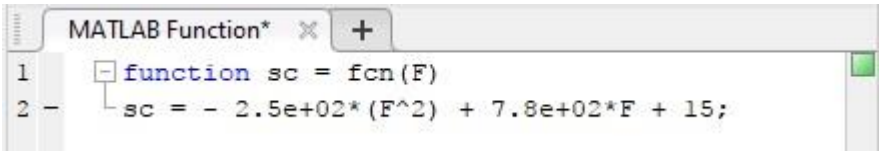

Figura 58. Ecuación implementada en el bloque MATLAB Function.

5.2 Identificación de la función de transferencia

En este apartado se podrá 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.

Para realizar este experimento empleamos el siguiente modelo de Simulink:

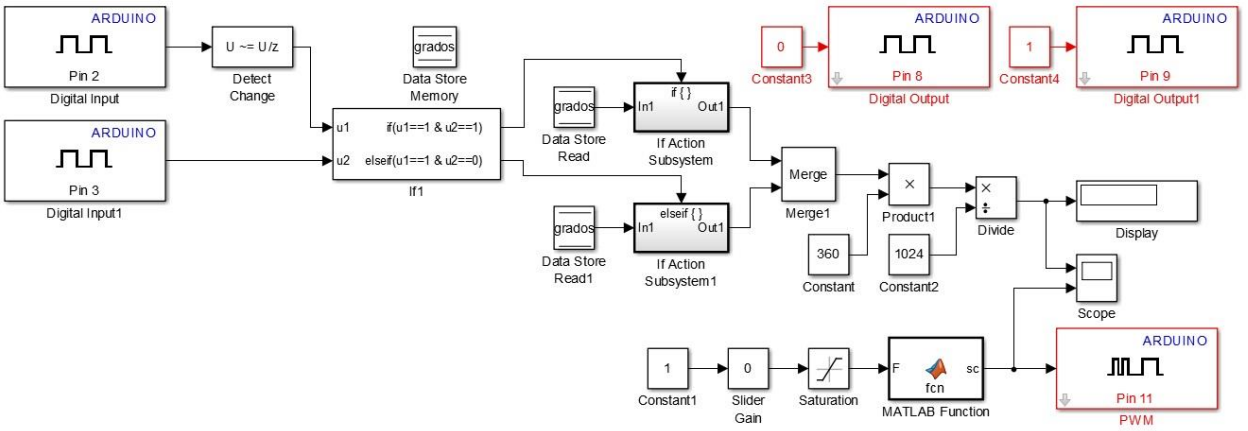

Figura 59. Modelo empleado para la obtención de la función de transferencia.

Para identificar la función de transferencia del Sistema, se van a aplicar escalones, tanto en subida como en bajada, los cuales serán graficados, y a partir de estas gráficas se obtendrán los parámetros correspondientes a nuestras funciones de transferencia.

La entrada será dada en unidades de fuerza (Newtons), mientras que el ángulo de salida será dado en grados.

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.

Se mostrarán dos gráficas donde se podrá observar la variación de la posición de la barra al aplicarle escalones de subida y de bajada, además junto a cada gráfica se realizará el cálculo de los parámetros correspondientes a una función de transferencia de segundo orden, y posteriormente, se simulará esta función de transferencia teórica junto al sistema real, y se podrán observar los resultados.

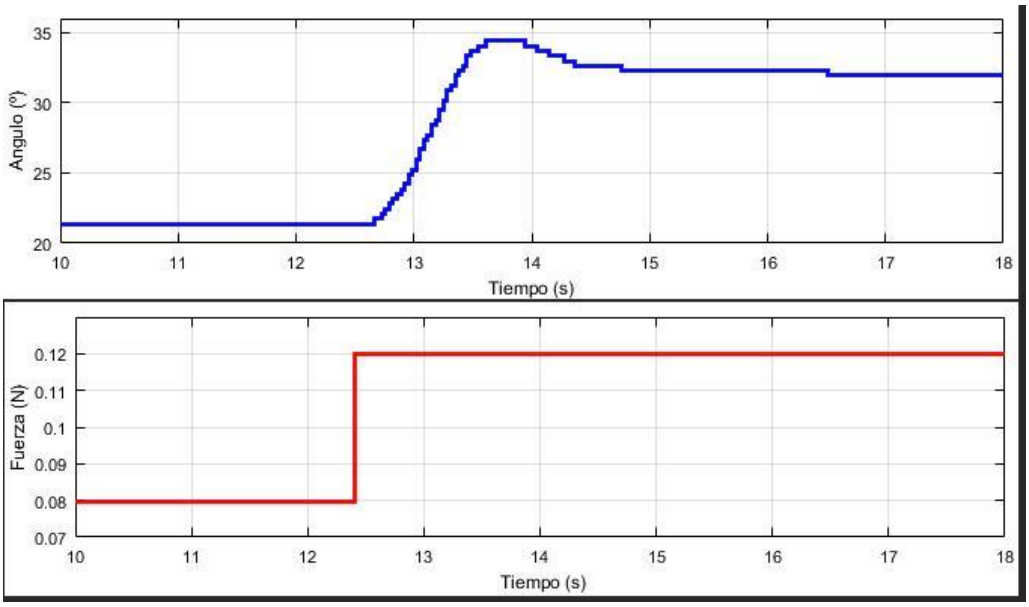

Figura 60. Respuesta del sistema ante un escalón de subida en fuerza.

En esta gráfica se puede observar como varía la posición de la barra al aplicar un escalón en subida desde 0.08N hasta 0.12N. Los parámetros correspondientes a la función de transferencia que representa esta dinámica de segundo orden se calculan de la siguiente manera:

$$Gsubida(s) = \frac{\theta(s)}{F_e(s)} = \frac{K\omega_n^2}{s^2+2\delta\omega_n s+\omega_n^2}$$

En este caso las incógnitas son los parámetros K, δ y ω_n los cuales están definidos y se pueden obtener experimentalmente a través de la relación de las especificaciones de la respuesta temporal con ellos. Para ello se analiza la respuesta temporal del sistema frente a una entrada de escalón, de donde se obtienen los valores:

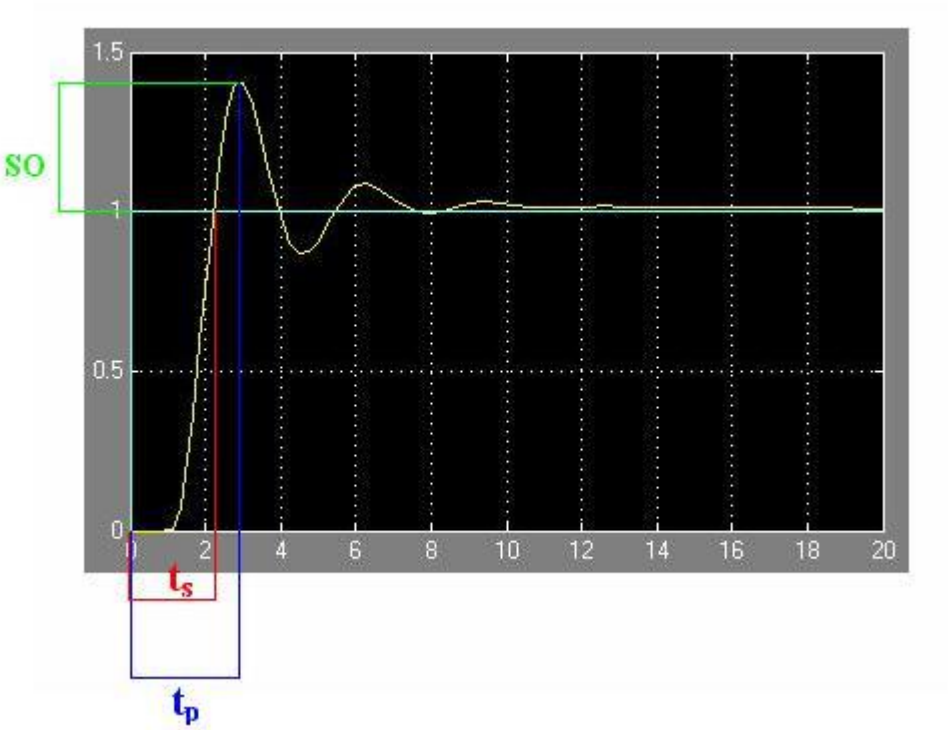

Figura 61. Respuesta típica de un sistema de segundo orden.

$$K = \frac{\Delta\theta_{rp}}{\Delta F_{rp}} = \frac{32 - 21.45}{0.12 - 0.08} = 263.75$$

$$S.O._{pu} = \frac{34.45 - 32}{32 - 21.45} = 0.2322$$

$$\delta = \sqrt{\frac{ln^2(S.O._{pu})}{\pi^2 + ln^2(S.O._{pu})}} = \sqrt{\frac{ln^2(0.2322)}{\pi^2 + ln^2(0.2322)}} = 0.4215$$

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\delta^2}} = \frac{\pi}{1.15\sqrt{1-0.4215^2}} = 3.0125\ rad/s$$

$$Gsubida(s) = \frac{\theta(s)}{F_e(s)} = \frac{K\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2} = \frac{2393.57}{s^2 + 2.5395s + 9.075}$$

A continuación se simulará la función de transferencia teórica obtenida, junto al sistema real para poder observar como de buena será nuestra aproximación y como responderá el sistema ante futuros controladores que se diseñen a partir de esta función de transferencia.

La simulación consiste repetir la situación descrita en la prueba experimental, introduciendo como entrada un escalón de subida de 0.04 Newtons. Una vez terminada la simulación se compara la respuesta obtenida con la obtenida en el sistema real para ver si se ha aproximado correctamente el comportamiento del sistema.

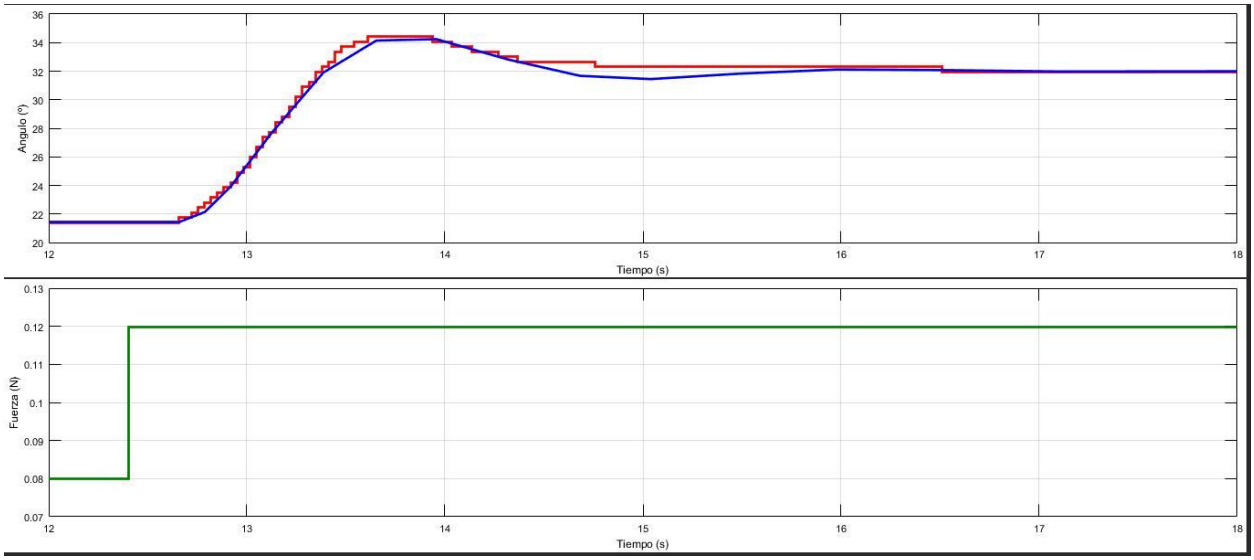

Figura 62. Simulación de la función de transferencia obtenida frente a la respuesta real.

Posteriormente se realizarán los mismos pasos pero para un escalón de bajada de 0.04 N, y se obtendrá la función de transferencia del sistema en bajada.

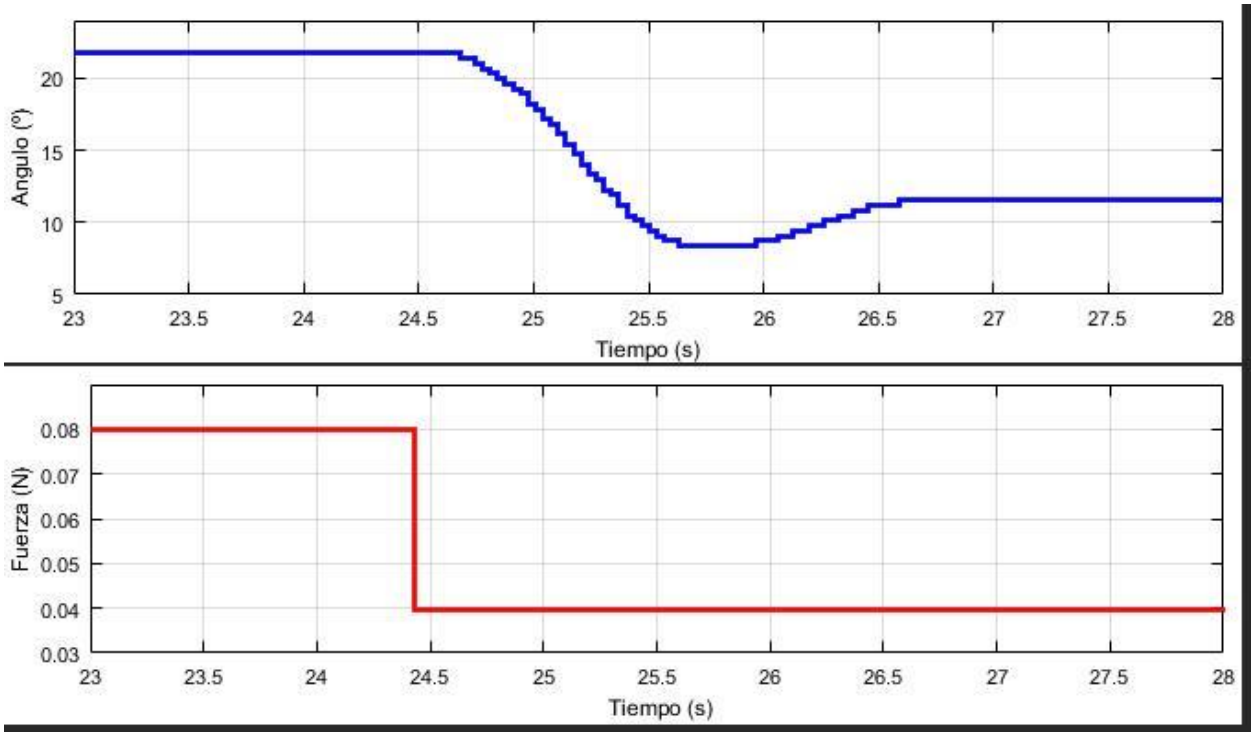

Figura 63. Respuesta del sistema ante un escalón de bajada en fuerza.

$$Gbajada(s) = \frac{\theta(s)}{F_e(s)} = \frac{K\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2}$$

$$K = \frac{\Delta\theta_{rp}}{\Delta F_{rp}} = \frac{21.8 - 11.6}{0.08 - 0.04} = 255$$

$$S.O._{pu} = \frac{11.6 - 8.44}{21.8 - 11.6} = 0.3098$$

$$\delta = \sqrt{\frac{ln^2(S.O._{pu})}{\pi^2 + ln^2(S.O._{pu})}} = \sqrt{\frac{ln^2(0.3098)}{\pi^2 + ln^2(0.3098)}} = 0.3495$$

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\delta^2}} = \frac{\pi}{1.12\sqrt{1-0.3495^2}} = 2.95\ rad/s$$

$$Gbajada(s) = \frac{\theta(s)}{F_e(s)} = \frac{K\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2} = \frac{2219.1375}{s^2 + 2.062s + 8.7025}$$

Realizando la simulación junto con el Sistema real se obtiene:

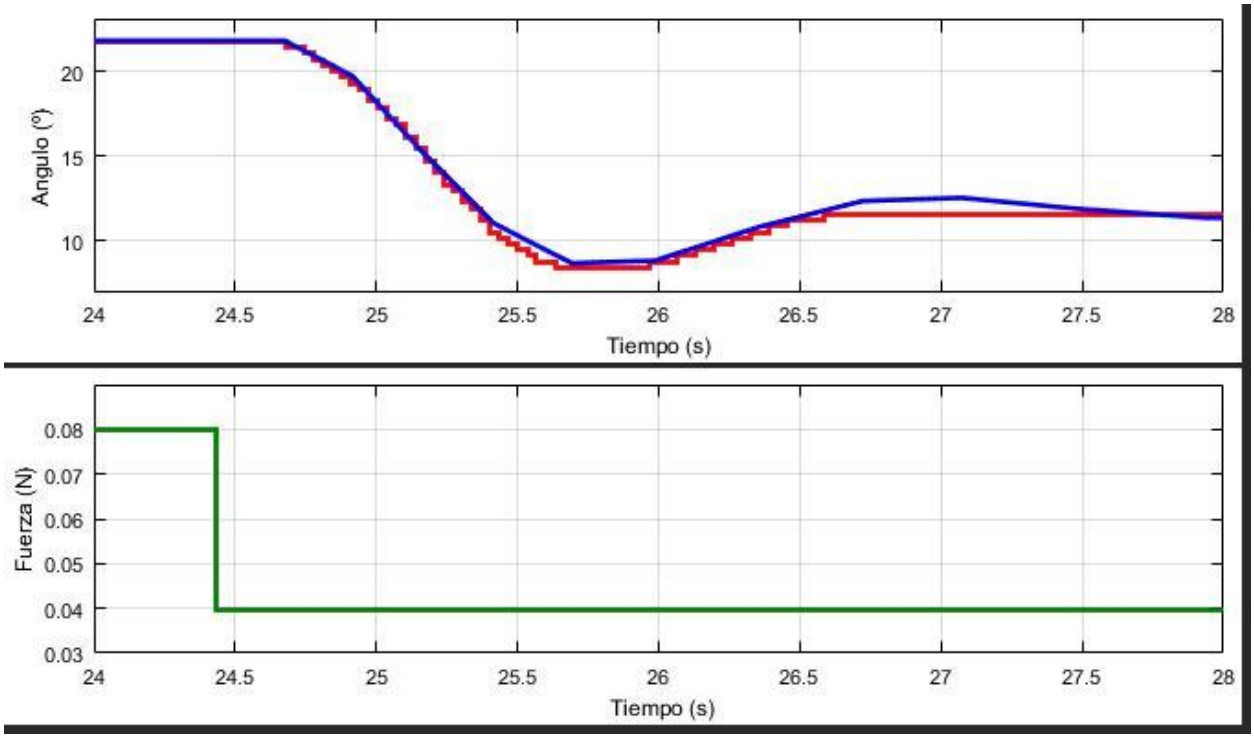

Figura 64. Simulación de la función de transferencia obtenida frente a la respuesta real.

Finalmente, para el diseño de los controladores que se dispondrán a continuación, se tomará una función de transferencia que estará formada por una media aritmética de los parámetros que se han obtenido anteriormente para Gsubida(s) y Gbajada(s). Por tanto la función de transferencia que se empleará estará formada por los siguientes parámetros y tendrá una forma tal que:

$$\bar{G} = \frac{\bar{K}\overline{\omega_n}^2}{s^2 + 2\bar{\delta}\overline{\omega_n}s + \overline{\omega_n}^2}$$

$$\bar{K} = \frac{263.75 + 255}{2} = 259.375$$

$$\bar{\delta} = \frac{0.4215 + 0.3495}{2} = 0.3855$$

$$\overline{\omega_n} = \frac{3.0125}{2.95} = 2.98125$$

$$\bar{G} = \frac{2305.2865}{s^2 + 2.2985s + 8.88785}$$

5.3 Control por cancelación de dinámicas

El control por cancelación de dinámicas es uno de los diseños más rápidos que podemos emplear, la idea básica es la de cancelar la dinámica de la planta con el cero que se le añade al controlador.

Se va a realizar además un control PID por realimentación negativa de la variable de salida (ángulo) en grados. Un controlador PID (Proporcional Integrativo Derivativo) es un mecanismo de control genérico sobre una realimentación de bucle cerrado, ampliamente usado en la industria para el control de sistemas. El PID es un sistema al que le entra un error calculado a partir de la salida deseada menos la salida obtenida y su salida es utilizada como entrada en el sistema que queremos controlar. El controlador intenta minimizar el error ajustando la entrada del sistema.

El controlador PID viene determinado por tres parametros: el proporcional, el integral y el derivativo. Dependiendo de la modalidad del controlador alguno de estos valores puede ser cero, por ejemplo un controlador Proporcional tendra el integral y el derivativo a cero mientras que un controlador PI solo el derivativo sera cero. Cada uno de estos parametros influye en mayor medida sobre alguna caracteristica de la salida (tiempo de establecimiento, sobreoscilacion,...) pero tambien influye sobre las demas, por lo que por mucho que ajustemos no encontrariamos un PID que redujera el tiempo de establecimiento a 0, la sobreoscilacion a 0, el error a 0,... sino que se trata mas de ajustarlo a un termino medio cumpliendo las especificaciones requeridas.

- Acción proporcional: La respuesta proporcional es la base de los tres modos de control, si los otros dos, control integral y control derivativo están presentes, éstos son sumados a la respuesta proporcional. Proporcional significa que el cambio presente en la salida del controlador es algún múltiplo del porcentaje del cambio en la medición. Este múltiplo es llamado ganancia del controlador.
- Acción integral: La acción integral da una respuesta proporcional a la integral del error. Esta acción

elimina el error en régimen estacionario, provocado por el modo proporcional. Por contra, se obtiene un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.

- Acción derivativa: La acción derivativa da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se disminuye el exceso de sobreoscilaciones.

Los controladores PID son suficientes para resolver el problema de control de muchas aplicaciones en la industria, particularmente cuando la dinámica del proceso lo permite (en general procesos que pueden ser descritos por dinámicas de primer y segundo orden), y los requerimientos de desempeño son modestos (generalmente limitados a especificaciones del comportamiento del error en estado estacionario y una rápida respuesta a cambios en la señal de referencia).

Su uso extensivo en la industria es tal que el 95% de los lazos de control que existen en las aplicaciones industriales son del tipo PID, de los cuales la mayoría son controladores PI.

El diagrama de bloques del sistema quedaría de la siguiente manera:

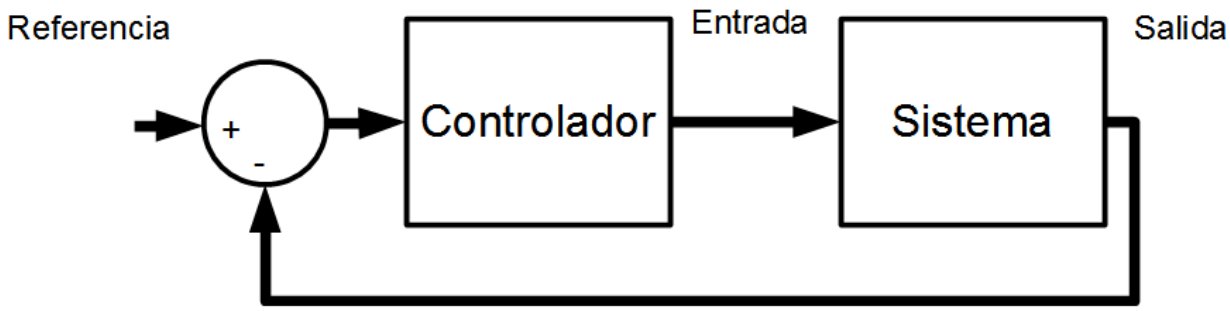

Figura 65. Diagrama de bloques del sistema.

Realizando una vista mas detallada del bloque del controlador quedaría:

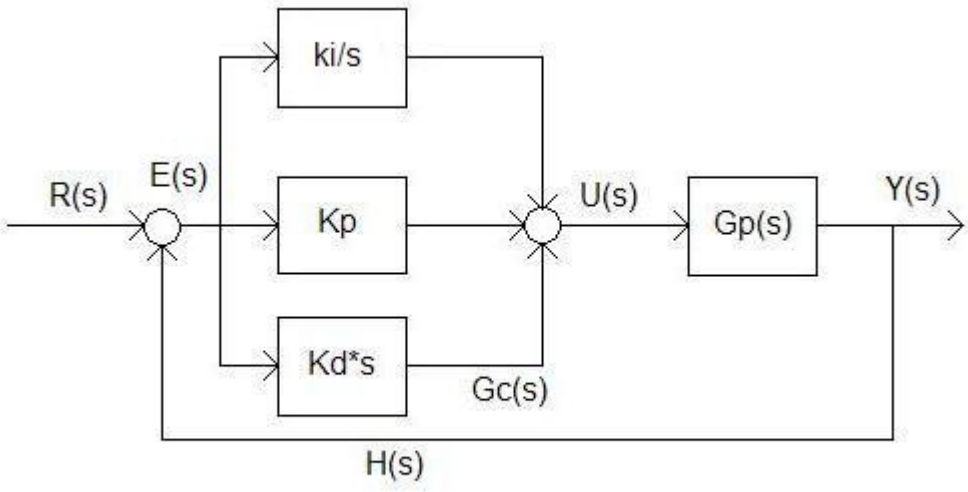

Figura 66. Diagrama de bloques detallado del controlador y el sistema.

Dónde Ki/s representa la parte integral del controlador, Kd*s la parte derivativa y Kp la parte proporcional, las cuales al unirse forman el controlador PID, por otro lado Gp(s) representa la función de transferencia del sistema a controlar.

El controlador elegido tendrá una función de transferencia tal que:

$$C(s) = \frac{Kc}{s} \cdot \frac{1}{\bar{G}(s)} \cdot \frac{1}{(\tau_{af} s + 1)}$$

Dónde los parámetros Kc y τ_{af} se diseñarán de la siguiente manera:

$$Kc = \frac{3}{t_s^{bc}}$$

$$\tau_{af} = \frac{t_s^{bc}}{50}$$

Se tiene un tiempo de muestreo de 0.001 segundos, además se sabe que la implementación del controlador se debe de realizar en tiempo discreto, por lo que antes de realizar experimentos, se debe realizar el siguiente procedimiento para implementar el controlador PID a partir de nuestra función de transferencia y obtener sus parámetros.

Se va a considerar que la función de transferencia del controlador es la siguiente, dejando a un lado el término $\frac{Kc}{s}$ el cual se implementará posteriormente:

$$C(s) = Kp. \frac{Ti.Td.s^2 + Ti.s + 1}{Ti.s(\tau_{af} s + 1)}$$

Para trabajar en tiempo discreto se realizará la siguiente conversion: $s = \frac{z-1}{zT}$

Realizando el cambio anterior se llega a que:

$$C(z) = \frac{Kp}{Ti} \cdot \frac{(Ti.Td + Ti.T + T^2)z^2 + (-2.Ti.Td - Ti.T)z + Ti.Td}{(T + \tau_{af})z^2 + (-2.\tau_{af} - T)z + \tau_{af}}$$

A continuación se igualan los términos para obtener los parámetros reales que serán implementados en el controlador.

$$C(s) = Kc. \frac{\frac{1}{\omega n^2} s^2 + \frac{2\delta}{\omega n} s + 1}{Ks} = Kp. \frac{Ti.Td.s^2 + Ti.s + 1}{Ti.s}$$

Se obtiene que:

$$Ti = \frac{2\delta}{\omega n}$$

$$Td = \frac{1}{\omega n^2.Ti} = \frac{\omega n}{\omega n^2.2\delta} = \frac{1}{2\delta\omega n}$$

$$Kp = \frac{Kc}{K} Ti$$

$$Kc = \frac{3}{t_s^{bc}}$$

Una vez que se encuentran diseñados los parámetros del controlador, se van a intentar implementar controladores con diferentes tiempos de subida, para ello se realizará el modelo de Simulink que se muestra en la figura, dónde para cada experimento se variarán los parámetros en el bloque "Discrete Transfer Fcn".

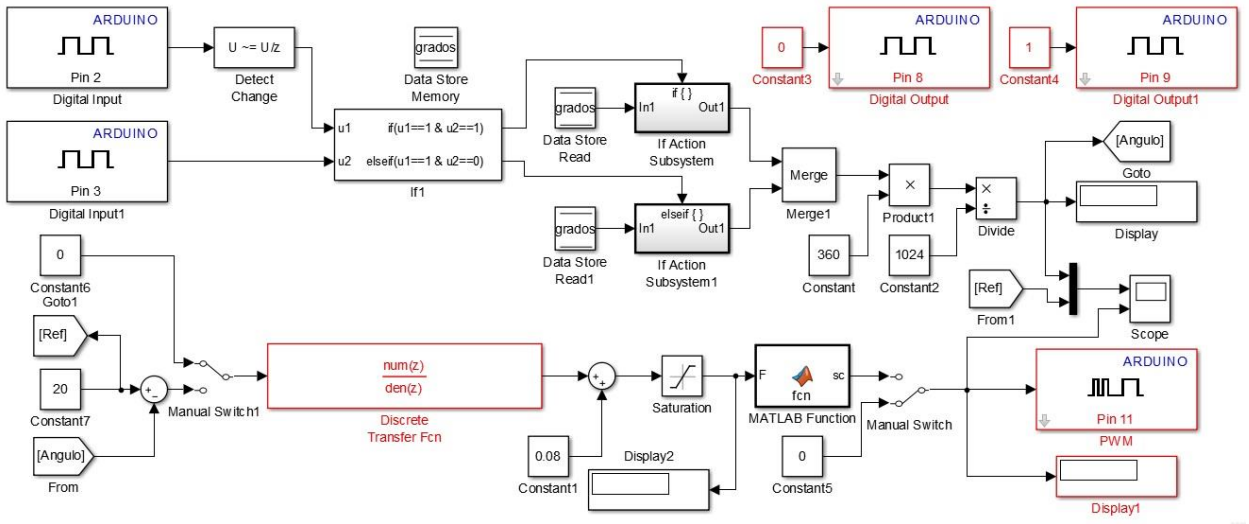

Figura 67. Modelo empleado para la implementación de los controladores.

Por un lado se puede ver en la parte superior el modelo de recepción de datos del encoder, mientras que en la parte inferior se puede observar el el bucle de control del sistema.

Se implementará un primer controlador con un tiempo de subida en bucle cerrado de 10 segundos. Por lo tanto tenemos que Kc=0.3 y $\tau_{af} = 0.2$

La simulación se ha basado principalmente en aplicar el controlador y dar valores a la referencia para ver la respuesta del sistema. En la siguiente figura se muestran los resultados de la simulación. La gráfica superior representa el valor del ángulo del sistema en cada instante medido por el encoder (señal roja) junto con el valor de referencia indicado en Simulink (señal azul). La gráfica inferior representa el valor de la señal de control que le llega al motor en valores de fuerza.

Lo que se ha hecho es, antes de activar el control, colocar el Sistema en una posición de aproximadamente 20º aplicando una fuerza constante de 0.08N. Es decir, una vez encendido el motor se le indica el valor de 0.08N en el bloque constant correspondiente. El interruptor que hay a la entrada del controlador se encuentra colocado hacia la constante de valor cero para anularlo, ya que si la entrada al controlador es cero, su salida también lo será. Con intención de que en el instante inicial cuando se aplique el controlador no se produzca un cambio brusco de la posición, se pone una referencia de 20º y posteriormente se activa el control cambiando la entrada en el interruptor que hay antes del controlador.

De este modo se consigue activar el controlador, pero al ser la referencia casi igual que la posición del Sistema, este no cambia de posición hasta que no se comienzan a cambiar los valores de la referencia.

En la gráfica obtenida podemos ver como el controlador tiene un buen comportamiento, llevando el sistema a su referencia, con un tiempo de unos 10 segundos aproximadamente como se pedía en las especificaciones.

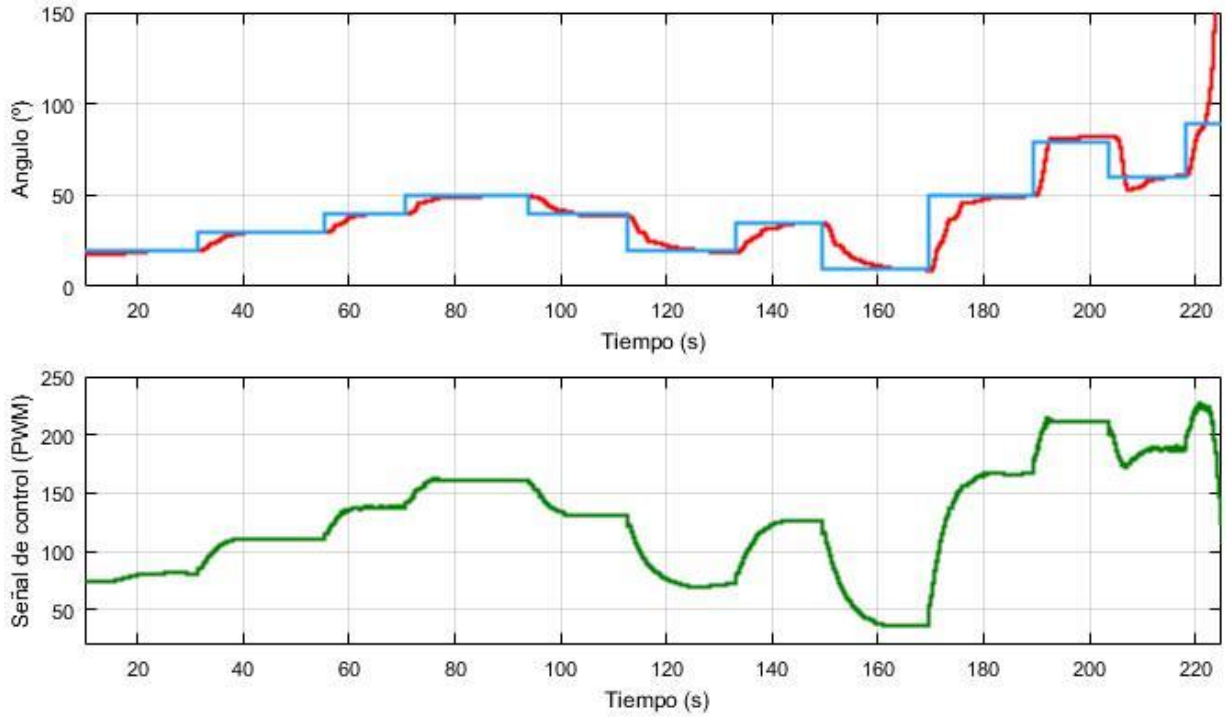

Figura 68. Resultado de simulación para controlador con ts=10 segundos.

A continuación se le introducirán unas perturbaciones al Sistema para ver de que modo actúa, se puede observar en la siguiente gráfica.

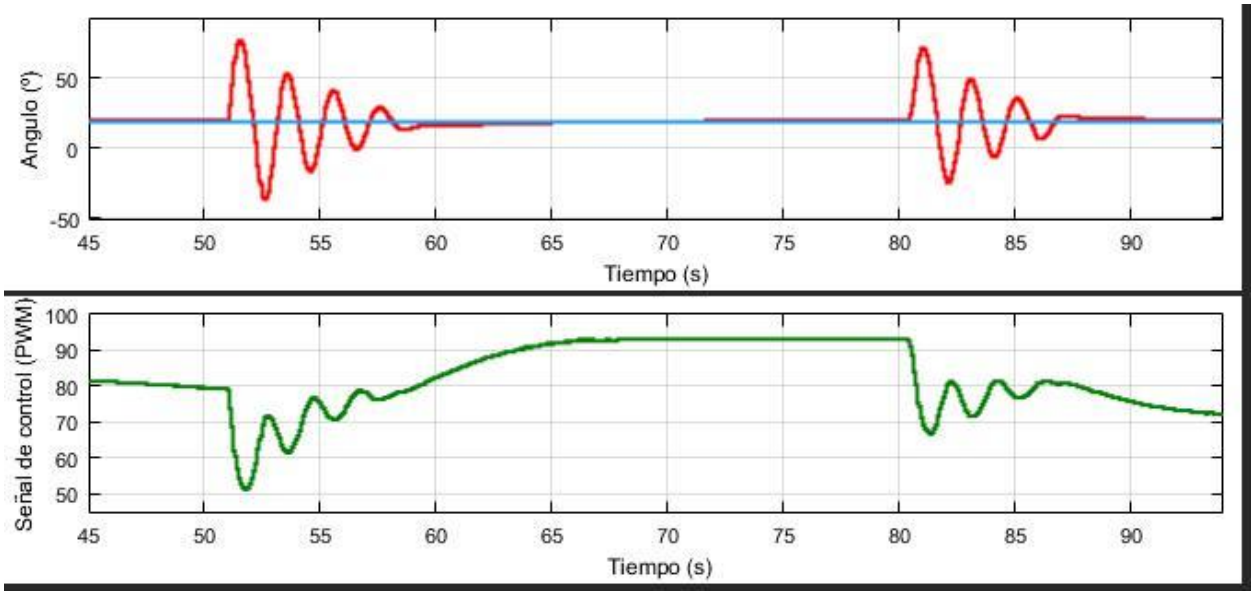

Figura 69. Resultado de simulación con perturbación para controlador con ts=10 segundos.

Se puede observar como el Sistema presenta un comportamiento acceptable ante perturbaciones, volviendo a su posición de equilibrio en el tiempo especificado de 10 segundos.

Para el segundo controlador se ha usado un tiempo de subida en bucle cerrado de 1 segundo, de este modo los parámetros Kc y τ_{af} tomarán unos valores de 3 y 0.02 respectivamente. La gráfica obtenida en la simulación es la siguiente, dónde los colores de las diferentes señales siguen la misma distribución que en el caso anterior.

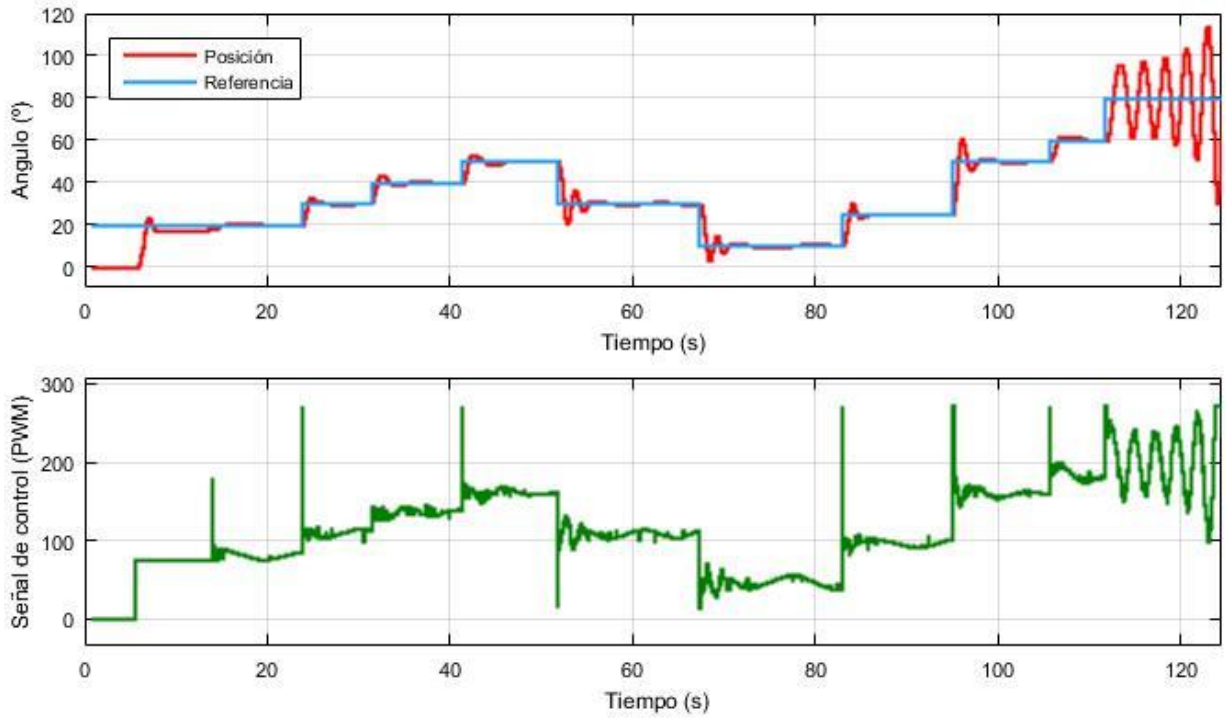

Figura 70. Resultado de simulación para controlador con ts=1 segundo.

El controlador presenta un buen comportamiento acorde a las especificaciones, sin embargo, cuando se le introducen perturbaciones, el sistema comienza a realizar oscilaciones cada vez mas grande y sin control alguno, como se puede observar en la siguiente gráfica.

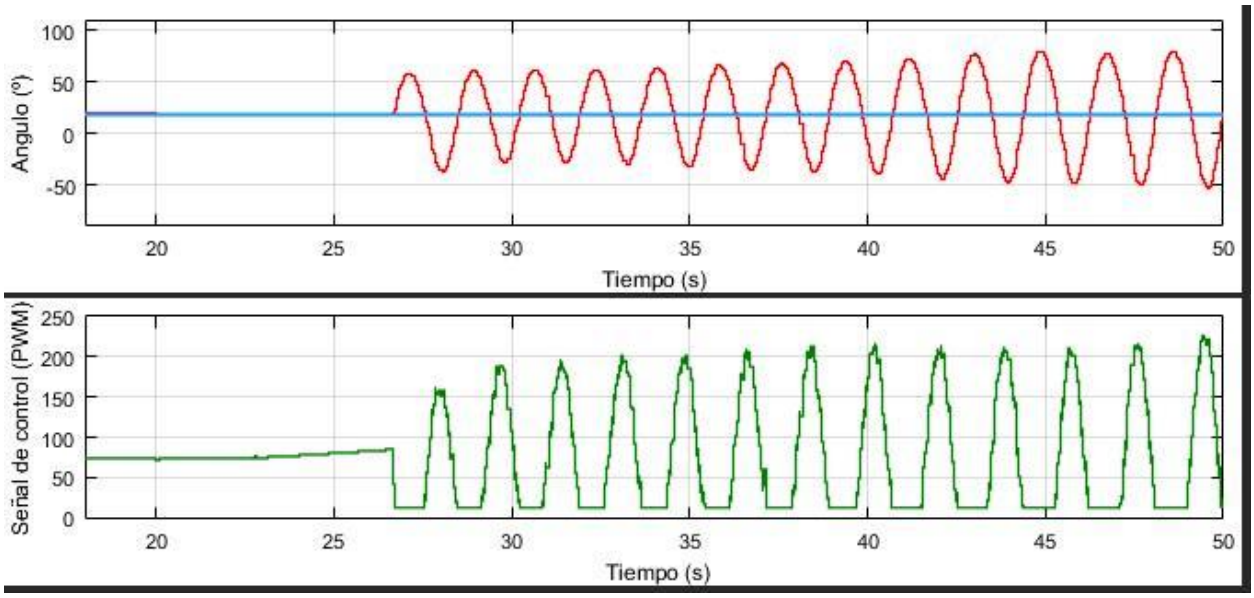

Figura 71. Resultado de simulación con perturbación para controlador con ts=1 segundo.

Finalmente, se intenta implementar un controlador con un tiempo de subida en bucle cerrado de 0.1 segundo, lo cual es una especificación demasiado ambiciosa, que como se puede ver en la gráfica, no aporta muy buenos resultados.

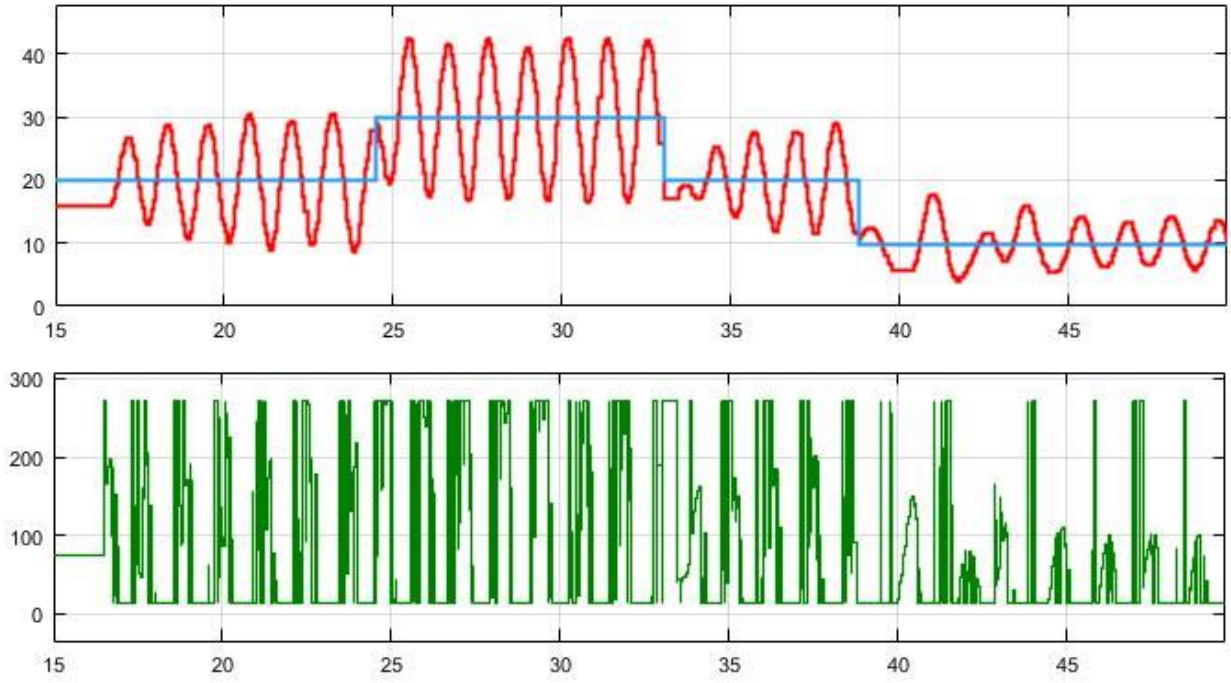

Figura 72. Resultado de simulación para controlador con ts=0.1 segundos.

Los parámetros Kc y τ_{af} toman para este controlador unos valores de 30 y 0.002 respectivamente.

5.4 Control sin cancelación de dinámicas

Para el diseño de este tipo de control, se empleará también un controlador PID, que será diseñado por el lugar geométrico de las raices.

Muchos estudios han desmostrado la estrecha relación que existe entre la respuesta transitoria de un sistema y la ubicación de las raíces de su ecuación característica en el plano s. Así mismo, se sabe que la variación de los parámetros físicos de un sistema que logran una modicación de su ecuación característica modican las raíces o polos de dicho sistema, de forma tal que se puede obtener una respuesta particular o deseada. Es por ello que, conocer la ubicación de las raíces en el plano s ante variaciones de un parámetro, puede representar una herramienta muy útil de análisis y diseño.

Cuando se trata de sistemas de control es sumamente importante conocer la ubicación de las raíces de la ecuación característica del lazo cerrado, lo cual puede conocerse utilizando un método sistemático y sencillo que muestra el movimiento de dichas raíces cuando se modica un parámetro de la ecuación. Dicho método permite elaborar lo que se conoce como el lugar geométrico de las raíces, que nos es otra cosa que las soluciones de la ecuación característica a lazo cerrado cuando se varía un parámetro.

El método de construcción para el lugar geométrico de las raíces de la ecuación característica a lazo cerrado cuando se varía un parámetro se fundamenta en un esquema de control con realimentación simple como el que se muestra en la figura, para el cual la ecuación característica a lazo cerrado es la siguiente, cuyas soluciones representan los polos del bucle cerrado.

$$1 + KG(s)H(s) = 0$$

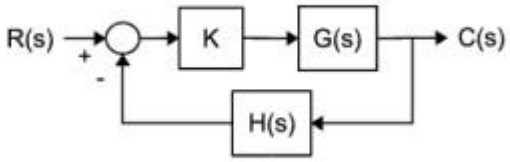

Figura 73. Esquema de realimentación negativa.

El lugar geométrico de las raíces se realiza para variaciones de K desde cero hasta innito, para las cuales dichas raíces deben satisfacer la ecuación anterior.

En este caso se ha empleado la siguiente función de transferencia para el controlador, la cual cuenta con un integrador, dos ceros y un polo:

$$C(s) = K_c \frac{(s+1)^2}{s(\tau_{af} s + 1)}$$

Para el diseño del controlador emplearemos la herramienta de Matlab "rltool" desde donde podremos ir modificando los parámetros hasta obtener el valor adecuado del PID.

Para ello, el primer paso será introducir la función de transferencia en el workspace de Matlab, y llamar a rltool con esta función de transferencia, como se indica en la figura.

Figura 74. Introducción de la función de transferencia en Matlab.

A continuación aparecerá una ventana, dónde en el apartado "Compensator Editor" se podrán añadir tanto los polos, como los ceros, y además se podrá indicar su posición, en este caso, se han añadido un integrador, dos ceros que se encontrarán juntos en el -1, y un polo de alta frecuencia, que estará situado en -50, como se puede observar en la figura.

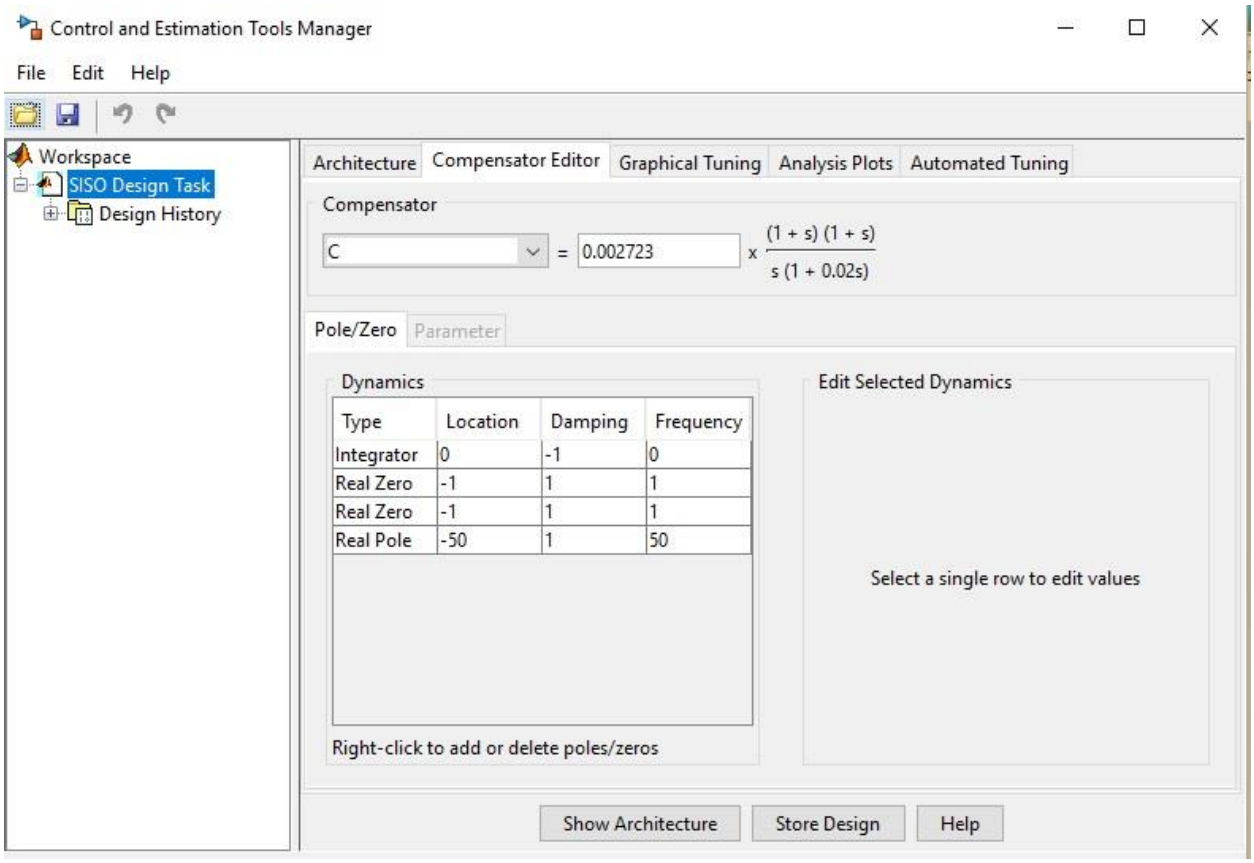

Figura 75. Ventana principal de la herramienta rltool de Matlab.

Finalmente, en la ventana dónde aparece dibujado el lugar de las raíces, se podrá variar la posición de la ganancia, hasta obtener la forma deseada.

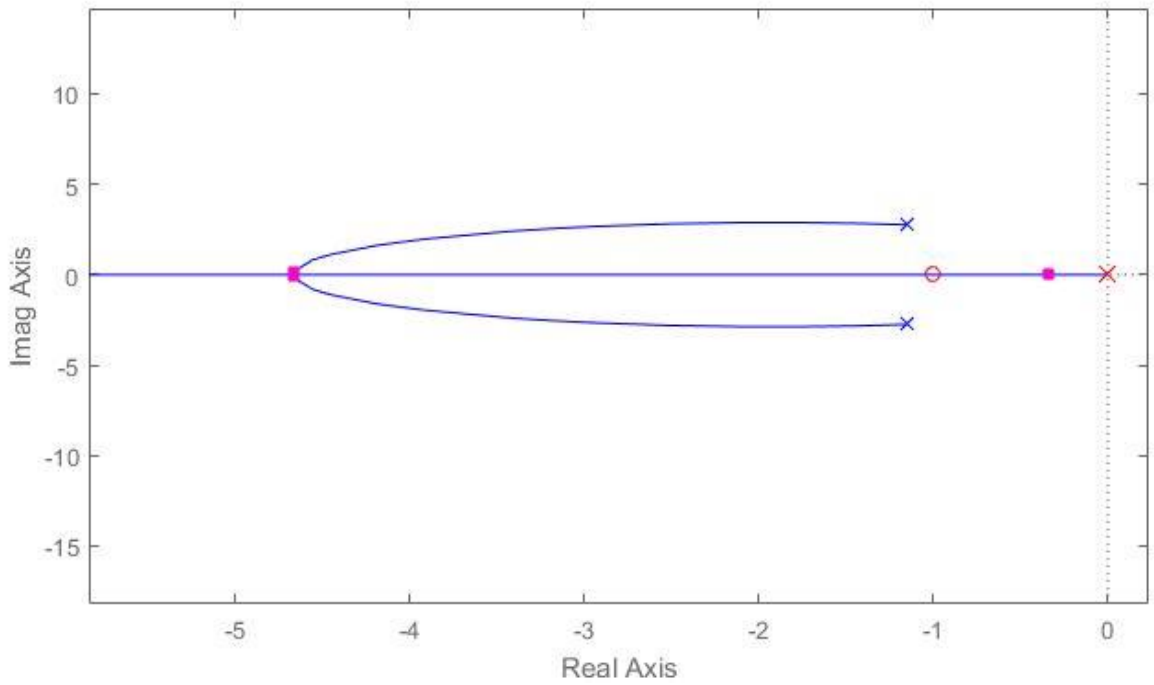

Figura 76. Dibujo del lugar geométrico de las raíces del sistema y el controlador.

Para implementar el controlador PID, se necesita obtener los parámetros Ti, Td y Kp, a partir de la función de transferencia obtenida gracias al lugar de las raíces, la cual quedó de la siguiente manera:

$$C(s) = Kc\frac{(s+1)^2}{s(\tau_{af}s+1)} = 0.002723\frac{(s+1)^2}{s(0.02s+1)} = 0.002723\frac{s^2+2s+1}{s(0.02s+1)}$$
$$= Kp.\frac{Ti.Td.s^2+Ti.s+1}{Ti.s}$$

Igualando los parámetros, se llega a que:

$\tau_{af} = 0.02$ Ti=2 $Ti.Td = 1 \rightarrow Td = \frac{1}{Ti} = \frac{1}{2} = 0.5$

$Kc = \frac{Kp}{Ti} \rightarrow Kp = Kc.Ti = 0.002723 * 2 = 0.005446$

El modelo Simulink que se usa en este apartado, es similar al usado en el apartado anterior, y lo único que se cambia es el contenido del bloque "Discrete Transfer Fcn", quedando de la siguiente manera:

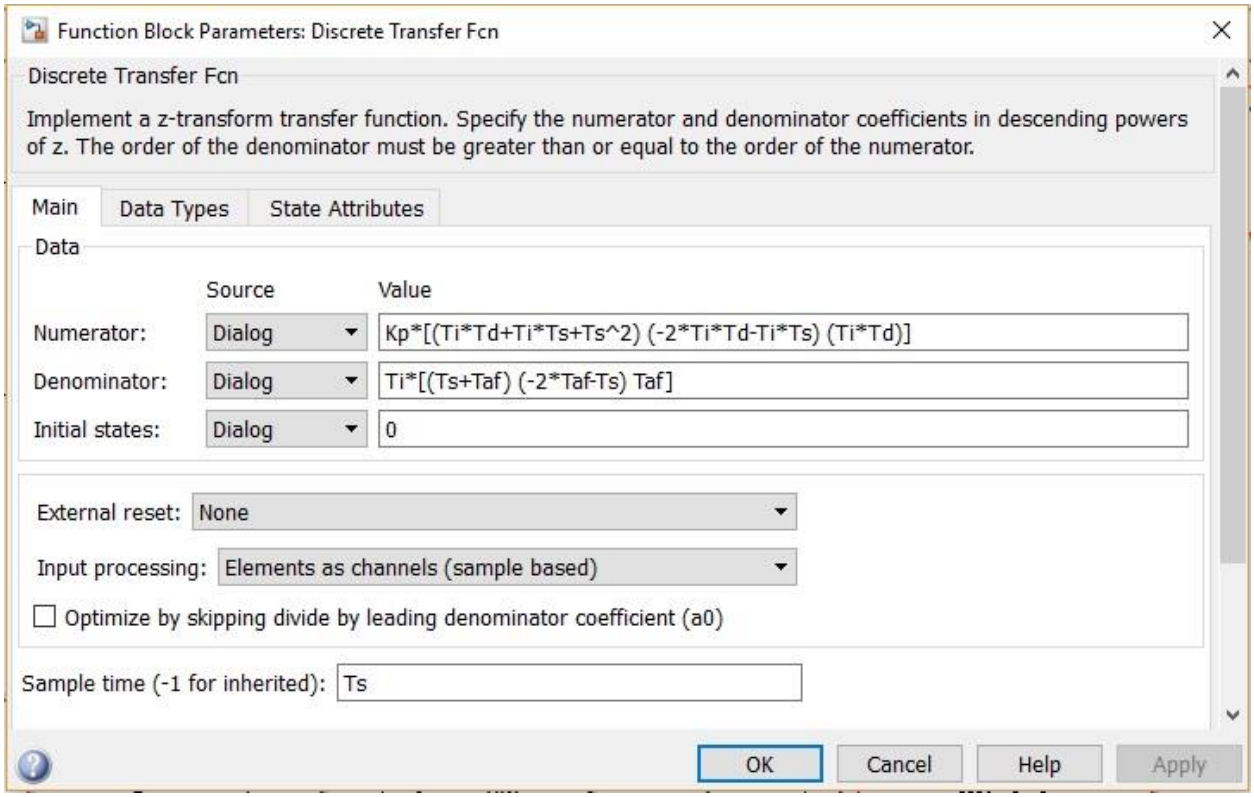

Figura 77. Ventana de parámetros de *Discrete Transfer Fcn*

Dónde Ts es el tiempo de muestreo del modelo, que está fijado en 0.001 segundos, y el resto de parámetros son los que se han calculado anteriormente.

Una vez que se ha concluido con el diseño del controlador y la configuración del modelo, se han realizado los experimentos pertinentes, dónde se han seguido los mismos pasos que en el apartado anterior, y dónde se han obtenido los siguientes resultados:

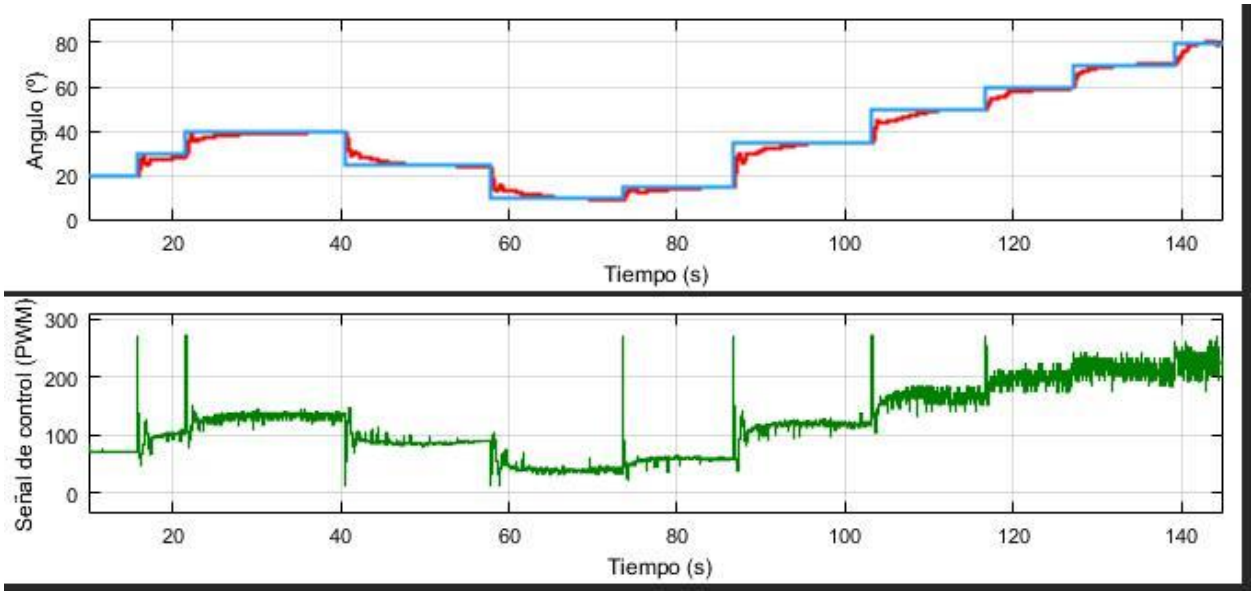

Figura 78. Resultado de la simulación para el control sin cancelación de dinámicas.

Se observa que la respuesta del sistema es la adecuada, incluso cuando se toman referencias mucho mayores que para las que está diseñado el controlador, ya que se ve que responde sin problemas incluso para los 80º.

Finalmente se ha sometido el sistema a una serie de perturbaciones, dónde el sistema ha respondido bien, volviendo a la posición de referencia de manera aceptable.

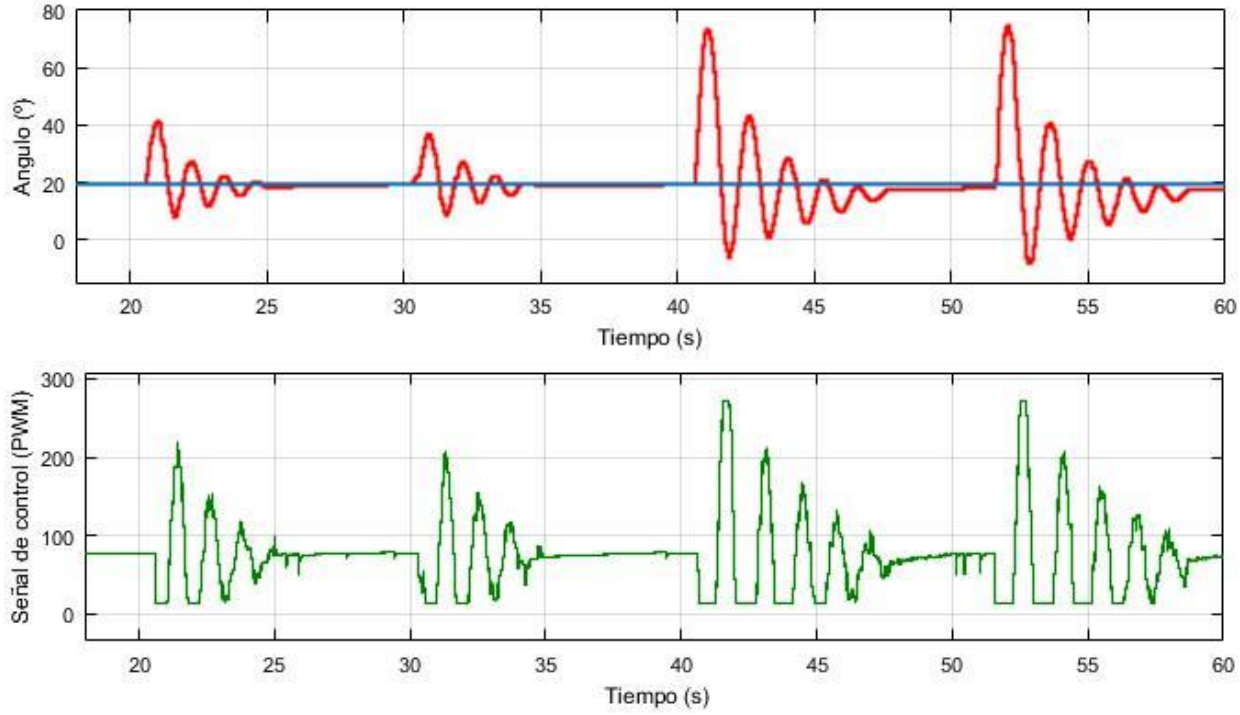

Figura 79. Resultado de la simulación con perturbación para el control sin cancelación de dinámicas.

BIBLIOGRAFÍA

[1] Rensselaer Arduino Support Package (RASPlib) http://homepages.rpi.edu/~hurstj2/

Información sobre encoder:

https://www.luisllamas.es/arduino-encoder-rotativo/

http://encoderarduinogrados.blogspot.com.es/

http://es.rs-online.com/web/p/codificadores-giratorios-mecanicos/7377694/

http://gerdslab.com/es/arduino_encoder_rotativo

http://www.abm-industrial.com/2013/02/07/que-es-un-encoder/

https://es.wikipedia.org/wiki/Codificador_rotatorio

Información sobre la Unidad de Medidas Inerciales:

http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

https://es.wikipedia.org/wiki/Unidad_de_medici%C3%B3n_inercial

http://smartdreams.cl/unidad-de-medicion-inercial-imu/

Información sobre el driver L298N:

https://aprendiendoarduino.wordpress.com/tag/l298n/

https://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/

https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/

http://www.prometec.net/l298n/

Información sobre motor de corriente continua:

https://aprendiendoarduino.wordpress.com/tag/motor-dc/

http://diymakers.es/control-velocidad-y-sentido-de-motor-dc/

http://www.prometec.net/motorcc/

Información sobre Arduino y control del sistema:

https://www.arduino.cc/

https://www.youtube.com/playlist?list=PLiJv_3SD9kXDsJuarmWEKnSWMSD39Hlzn

https://wechoosethemoon.es/2011/07/21/arduino-matlab-simulink-controlador-pid/