UNIVERSIDAD
DE MÁLAGA

ESCUELA DE
INGENIERÍAS
INDUSTRIALES

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

CONTROL AVANZADO DEL ROBOT MOVIL KROBOT CON SENSORES INTERNOS

Grado en

Ingeniería Electrónica, Robótica y Mecatrónica

Autor: José María Díaz de Haro

Tutor: Antonio José Muñoz Ramírez

MÁLAGA, Septiembre de 2.018

RESUMEN – CONTROL AVANZADO DEL ROBOT KROBOT CON SENSORES INTERNOS

Palabras Clave: Robot, Skid Steer, Controlador, Arduino, Matlab, Simulink y Wifi

Este trabajo de Fin de Grado se basa en dotar a un robot de tipo Skid Steer de un mejor control de bajo nivel, partiendo de un remodelado del diseño hardware que busca optimizar el estado de los elementos ya existentes, así como instalar nuevos periféricos y acabando con el diseño y la programación de los controladores pertinentes para el robot.

Para lograrlo, se utilizará el software de Matlab y Simulink donde se realizarán todos los programas, incluyendo código y esquemáticos gráficos y como unidad de procesamiento utilizaremos una placa Arduino Due.

Será necesario desarrollar aplicaciones de comunicación WiFi para poder intercambiar información con el robot a distancia, y con estas se obtendrán funciones matemáticas que permitan modelar el comportamiento del robot en diferentes terrenos y pudiendo así diseñar controladores para cada situación como puede ser el movimiento del robot en el aire, donde sus ruedas no están en contacto con nada, o subiendo una cuesta en un terreno de tierra.

También se ha realizado el estudio de unos sensores de corrientes conectados a los motores buscando tener más información interna del sistema, así como la utilización de una IMU para conocer la orientación del robot en el espacio.

ABSTRAC – ADVANCED CONTROL OF THE MOVILE ROBOT KROBOT USING INTERNAL SENSORS

Keywords: Robot, Skid Steer, Controller, Arduino, Matlab, Simulink and Wifi

This Final Degree Project is based on providing to a Skid Steer robot a better low level control, begining with a change in the hardware design that seeks to optimize the status of the already existing elements, as well as installing new peripherals and finishing with the design and the programming of the controllers for the robot.

In order to achieve this goals, it Will be used the software of Matlab and Simulink, where all the programs Will be made, including code and schematic graphics and for the processing unit we will use an Arduino Due board.

It Will be necesarry to design WiFi communication applications wich provide us to exchange information with the robot at a distance, giving us the posibility of obtaining the mathematics functions that models the behavior of the robot in different terrains and situations and also, we will be able to create controllers for each of this situations such as the movement of the robot in the air where the wheels aren't in touch with anythinh or going up a hill on earth ground.

It has also been made the study of current sensors connected to engines seeking to get more internal system information, as well as the use of an IMU sensor to know the orientation of the robot in the space.

Agradecimientos

A mis padres, por aguantarme tantos años y siempre darlo todo lo que tienen y lo que no para que yo pueda estar donde estoy ahora mismo. A Jessica, por siempre estar ahí, por enseñarme el valor de esforzarse en lo que te importa y por recordarme todos los días lo bonito que es aprender algo nuevo y por último a Antonio, por sacar el verdadero ingeniero de mí y nunca perder la paciencia.

INDICE

Índice de Figuras

Capítulo 1 - Introducción

1.1 Antecedentes

Este Trabajo de Fin de Grado toma como objeto de trabajo el robot móvil Krobot, cuyo movimiento se describe por giro por deslizamiento, a partir de ahora Skid-Steer, diseñado por el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga, que se encuentra en constante desarrollo. Este robot ya fue desarrollado en el TFG de Raúl Pimentel Castillo [1], donde se hizo el diseño y el montaje del robot Krobot para el departamento de Ingeniería de Sistemas y Automática y el TFG de Carlos Miguel Jiménez [2] , en el cual se realizó un primer control del robot, también realizado en el mismo departamento.

El Krobot (figura 1.1) se trata de un robot móvil de 4 ruedas con movimiento de tipo Skid Steer[3], dotado de un motor para cada rueda, así como un encoder para cada motor, un sensor IMU-9250[4] (Acelerómetro, Giroscopio y Magnetómetro), un módulo wifi ESP8266 y una placa Arduino Due[5] que actúa de controlador de todos estos elementos. Además, también posee un ordenador de a bordo, una Raspberry Pi 3 que tiene el control de una cámara Kinect y un sensor laser Hokuyo.

Figura 1.1 - Robot Móvil Krobot

Los conocimientos que se utilizaran como base de trabajo son los adquiridos, principalmente, en la asignatura de Laboratorio de Robótica donde se trabajó con el robot Piero, un robot móvil con dos ruedas de tracción diferencial. En esta asignatura se aprendió como modelar y controlar el sistema con la combinación de la placa Arduino y el software Simulink[6] de Matlab[7].

1.2 Objetivos

El principal objetivo del presente trabajo es la mejora del control de bajo nivel que ya posee el Krobot, optimizando los controladores de los motores, añadiendo sensores de consumo[8] en cada motor, incluir el sensor IMU en el bucle de control del sistema y buscar mejorar la cinemática que se posee para una mayor precisión en el posicionamiento.

1.3 Fases de Trabajo

Para lograr el objetivo, se seguirán las siguientes etapas de actuación:

1. Remontaje del sistema: para corregir errores del montaje anterior, añadir claridad al mismo e incorporar los nuevos sensores de consumo.

2. Diseño de un mejor sistema de control de velocidad del robot con múltiples controladores en función del estado en el que se encuentre (Tipo de terreno, tipo de movimiento, etc.)

3. Estudiar la cinemática del sistema considerando su diferenciación como Robot de tipo Skid Steer frente al modelo cinemático de robot diferencial ya conocido, así como su incorporación en el control del sistema.

4. Estudiar el comportamiento del sensor IMU e incorporar un filtro con este que permita mejorar el posicionamiento del robot.

1.4 Estructura de la Memoria

La presenta memoria esta divida en 6 capitulos, a continuación se resumen el resto de capítulos que siguen a este:

En el capítulo 2, se enumeran todos los elementos que componen el sistema de bajo nivel del Krobot y se hace una breve introducción de los mismos, además se explica el remodelado del sistema.

En el capítulo 3 se explica todos los pasos necesarios para obtener el modelo del robot Krobot, desde el estudio de la obtención de la velocidad por los encoders hasta los diferentes modelos de los motores.

En el capítulo 4 se describe la metodología utilizada para la obtención de los diferentes controladores de velocidad para los modelos obtenidos en el capitulo 3 así como las características de los mismos.

En el capítulo 5, se estudian los nuevos sensores de consumo y se explica le implementación que se ha hecho de un filtro con el sensor incercial y su aplicación en el diseño.

Finalmente, el capítulo 6 sirve de conclusión para los trabajos realizados así como un análisis de las siguientes posibles líneas de actuación para continuar desarrollando el robot Krobot.

Capítulo 2 - Descripción de los elementos Hardware del Krobot (Bajo Nivel)

2.1 Introducción

Como se comentó en los objetivos, al comienzo del desarrollo del TFG se realizó un remontaje del robot con dos objetivos, en primer lugar, la familiarización con los componentes que lo forman y en segundo, instalar nuevos sensores de consumo a cada rueda, buscar una configuración de los elementos más accesible y corregir errores de la instalación anterior.

2.2 Elementos Hardware

2.2.1 Placa Controladora Arduino Due

La Arduino Due es una placa electrónica basada en la CPU Atmel SAM3X8E ARM Cortex-M3 denominado SAM3X8E de Atmel que trabaja hasta a 84Mhz, lo que supone un gran aumento de potencia y velocidad comparado con otras placas de la familia, por lo que es ideal para el robot Krobot permitiendo procesar todas las interrupciones de los cuatro encoders, gestionar todo el bucle de control del sistema y al mismo tiempo realizar las comunicaciones serie pertinentes.

Además de lo mencionado anteriormente, la placa Arduino Due tiene 54 pines digitales de entrada y salida (y de estos, 12 pueden ser usados como salidas PWM), 12 entradas analógicas, 2 salidas analógicas, 4 UART (puertos seriales hardware), el ya mencionado cristal oscilador de 84MHz, una conexión compatible con USB-OTG que permite la utilización de ratones y teclados, 2 TWI, el Jack de alimentación, conexión JTAG, botón reset y un botón borrar.

Como todas las placas de la familia, la placa Arduino Due se programa en el software IDE de Arduino, pero también podemos programarla utilizando el software SIMULINK de Matlab, permitiéndonos trabajar a alto nivel con una programación visual para una mayor comodidad de usuario y al mismo tiempo, pudiendo utilizar también bloques de bajo nivel cuando sea necesario.

Sin embargo, una de las características importantes que debemos tener en cuenta de la placa Due que es que, al contrario que el resto de las placas Arduino, los pines I/O trabajan a 3.3V y no a los 5V habituales lo cual puede llegar a dar problemas en la lectura de las señales o degenerar en problemas para la placa en sí.

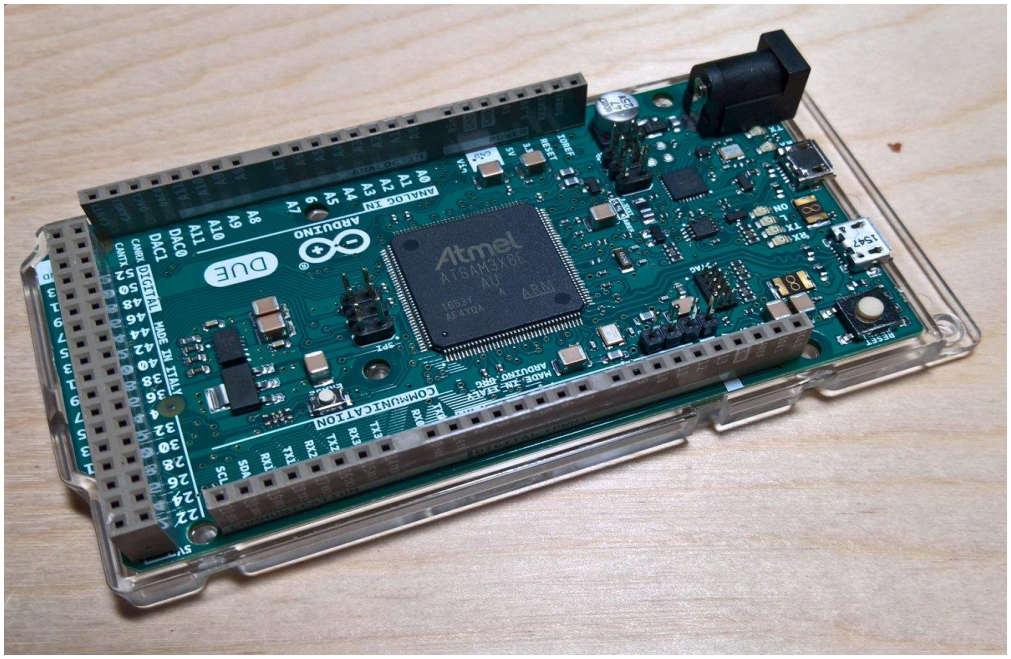

Figura 2. 1 - Placa Arduino DUE

2.2.2 Motor DC HN-GH7.2-2414T-50:1

El Krobot posee cuatro motores de corriente continua con tensión nominal de 7.2V, que ofrecen 175 revoluciones por minuto (En vacío). Además, los motores traen incorporados sensores de codificación de posición ópticos de efecto hall (Encoders) formado por un rotor con tres imanes y dos pines de activación, además de una reductora 50:1(por tanto, si tenemos 3 imanes, dos pines de activación, dos flancos y la relación de 50 giros de motor por giro de rueda tenemos que:

Señales de encoder por vuelta = 3*2*2*50= 600 señales/vuelta de rueda

Figura 2. 2 - Motor DC 7.2V 175 RPM

Figura 2. 3 - Encoder Óptico

2.2.3- Drivers de Potencia LN298

Los drivers de potencia tienen como función convertir la señal PWM de los controladores en la señal PWM de actuación que recibirá cada motor. En este caso, se utilizan los LN298 que se tratan de puentes H duales, es decir, cada uno se encarga de controlar dos motores mediante dos canales diferenciados, recibiendo las señales de PWM desde Arduino. También ofrecen la posibilidad de utilizar señales de habilitación para cada motor.

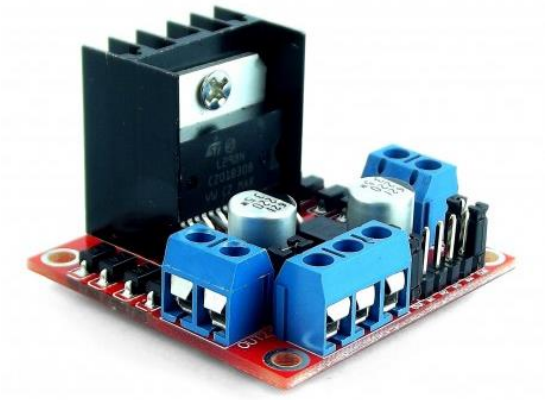

Figura 2. 4 - Driver de Potencia LN298

2.2.4- Modulo Wifi ESP8266

Este chip nos permite establecer conexiones wifi, mediante comandos de texto AT, utilizando una puerta serie que se puede configurar a distintas velocidades. Además, el módulo Wifi del Krobot está configurado para emitir señal por sí mismo por lo que basta con conectarnos a su red para establecer comunicaciones con el ordenador. Otra característica del módulo ESP8266 es que incluye un pequeño procesador interno que podríamos programar para funcionar de modo autónomo y que incluso dispone de un par de puerto GPIO.

Sin embargo, también posee un pequeño inconveniente y es el consumo del chip, sobre todo durante el arranque, por lo que no se puede alimentar directamente con la placa Arduino si no que se conecta a la fuente de alimentación externa.

Figura 2. 5 - Modulo Wifi ESP8266

2.2.5-Unidad inercial MPU 9250

El dispositivo de control de movimiento MEMS de 9 ejes MPU-9250 cuenta con un giroscopio, un acelerómetro y una brújula MEMS combinados en un procesador de movimiento digital integrado. La unidad de procesamiento de movimiento (MPU) de baja potencia ofrece un sistema encapsulado que incorpora la MPU-6500 y la brújula digital AK8963 de 3 ejes. Todas estas características quedan reunidas en un solo chip de tamaño reducido que presenta además un muy bajo consumo frente al modelo anterior MPU-9150.

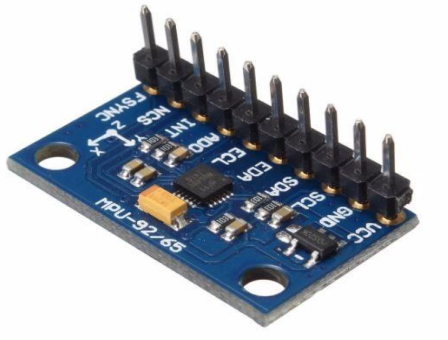

Figura 2. 6 - MPU 9250

2.2.6 -Sensores de Consumo de Corriente ACS712

El funcionamiento de este sensor se basa en el efecto Hall lineal utilizando una pista de cobre conductor ubicado cerca de la superficie de la matriz. La corriente aplicada que fluye a través de esta pista de conducción de cobre genera un campo magnético que se convierte en una tensión proporcional. En nuestro caso, será interesante estudiar las lecturas del sensor debido a que la intensidad que circulará por el motor (una vez esté controlado) variará en función de la oposición de la rueda con el medio, lo cual nos permitirá saber, por ejemplo, si la rueda está apoyada o no.

Figura 2. 7 - Sensor de Consumo

2.2.7-Baterias

Las baterías del Krobot son de metal de hidruro de 7,2 V y 4.700 mAh, ya que poseen una tensión igual a la de los motores no se necesita ningún tipo de transformador de tensión y ofrecen un alto amperaje, ideal para soportar el consumo de la Arduino, los motores y el módulo ESP8266.

Figura 2. 8 - Baterías

2.3 – Remontaje del Krobot

Una vez se habían identificado todos los elementos con los que se iba a trabajar, se realizó un desmontaje del sistema para poder volverlo más accesible, corregir errores y posteriormente añadir al sistema los sensores de consumo. El primer aspecto del robot (Figura 2.9) era muy caótico por lo que se optó por retirar todos los elementos e intentar colocarlos optimizando el espacio y buscando en todo momento mantener el centro de gravedad lo más cerca posible del centro geométrico. Para obtener toda la robustez posible en el montaje, se utilizaron tiras de velcro para fijar los elementos de mayor tamaño (Puentes H, sensores de consumo) y se sujetaron los cables más largos con bridas adhesivas evitando así aglomeraciones.

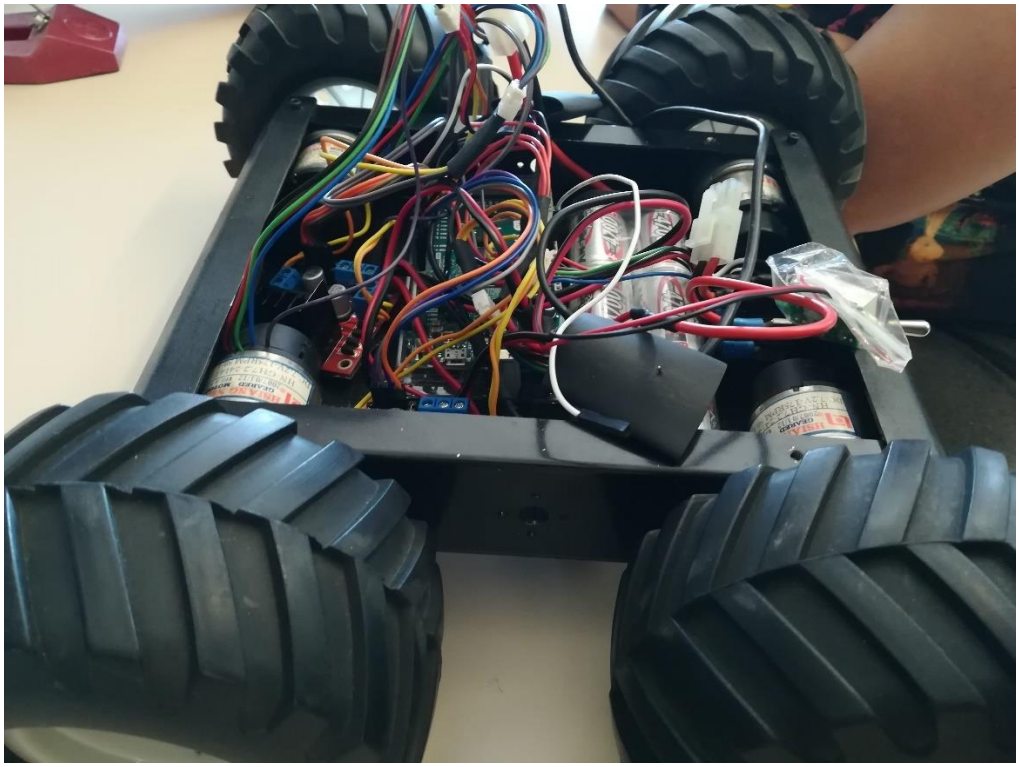

Figura 2. 9- Configuración Inicial

Además, hubo que corregir algunas conexiones, por ejemplo, los encoders recibían la alimentación directamente desde las baterías y esto podía llegar a dañar la placa Arduino por lo que se cambió y ahora recibe la tensión directamente desde la propia placa. También se instaló en el frontal de la carrocería un voltímetro para permitir al usuario conocer el estado actual de las baterías y así saber si es necesario cargar el robot. Cuando el montaje estuvo listo y se cerró el sistema con su tapa, se fijó con velcro el sensor IMU al centro de la tapa, buscando eliminar perturbaciones a la hora de su funcionamiento

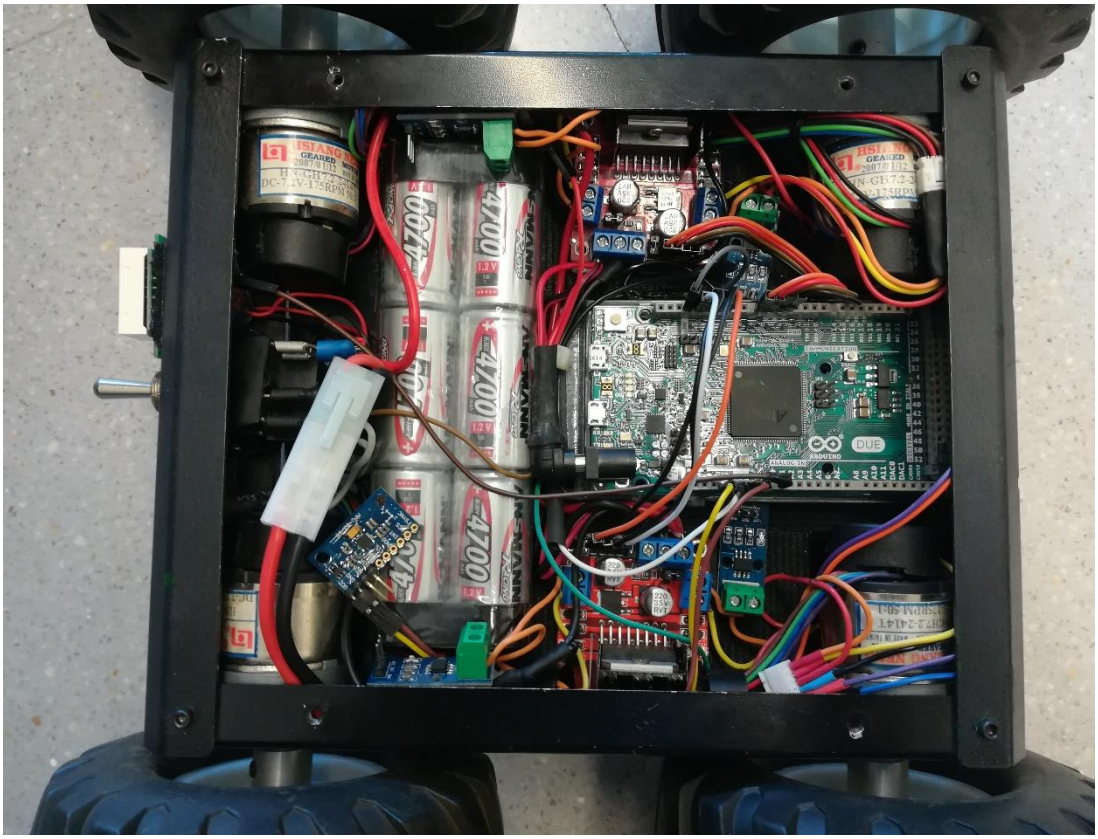

Figura 2. 10 – Montaje Final Destapado

Figura 2. 11 – Montaje Final Tapado

Capítulo 3 – Modelado del Krobot

3.1 Introducción

En este capítulo vamos a explicar cómo se desarrolló el modelo en Simulink del modelo del Krobot, partiendo desde un sistema básico con envio de señales PWM directamente a los motores hasta desarrollar un control de velocidad en bucle cerrado para diferentes situaciones.

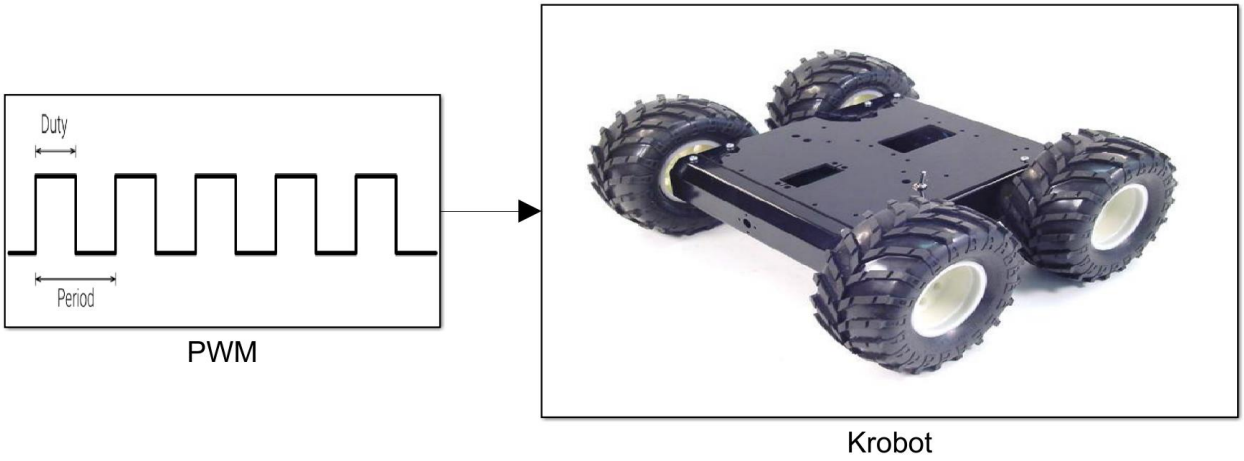

Figura 3. 1 – Nivel TOP del diseño Simulink

3.2 Comprobación de funcionamiento del sistema

En primer lugar, se realizó un esquemático básico donde solo se tiene una señal PWM fija (de un valor de 100) que se conecta directamente a los pines donde están los controladores del Puente H. Cada motor se puede activar por dos pines, uno para cada sentido de giro, y para evitar posibles malfuncionamientos se han limitado las señales que se pueden llegar a enviar en un rango de 0 a 255.

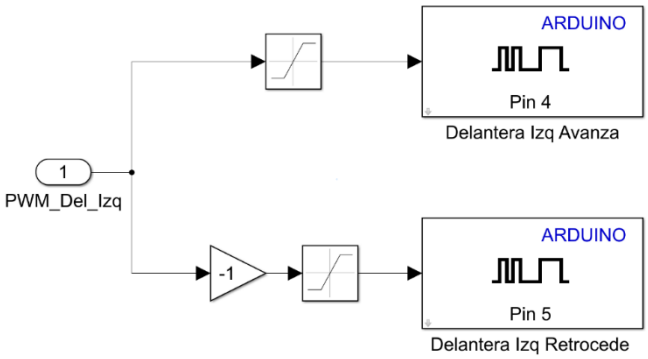

Figura 3. 2 Bloque controlador de una rueda

Cuando se comprobó el correcto funcionamiento del montaje, se pasó al estudio de los encoders. El primer punto de partida fue la S-Function ya realizada en el primer control del Krobot. Este bloque tiene asociadas funciones de interrupciones a los cambios de valor de los pines A y B de cada encoder. Cada vez que se lanza una de estas interrupciones, se evalúa tanto el estado actual del pin que ha provocado la interrupción como el de su pareja para conocer si el movimiento ha sido en un sentido o en otro.

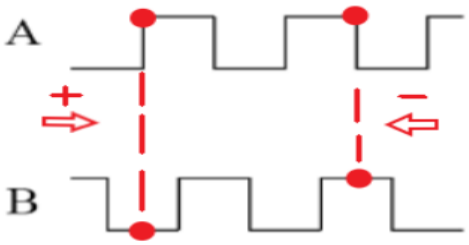

Figura 3. 3 - Evaluación del sentido de giro de los Encoders

Sin embargo, además de la cuenta de pulsos, en el bloque de encoder se calculaba la velocidad de giro en función del tiempo que había tenido entre las dos últimas interrupciones, así se decidió comparar la resolución y el ruido entre esta señal de velocidad y la derivada discreta de la posición para determinar cuál utilizar posteriormente.

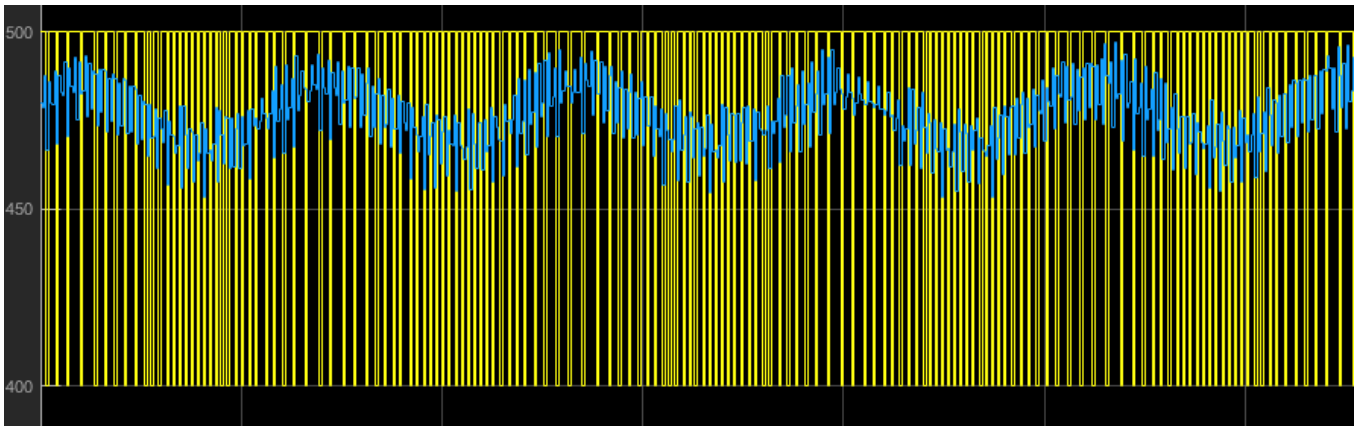

Figura 3. 4 - Graficas de comparativa de señales de velocidad

La señal amarilla se corresponde con la derivada de la cuenta de pulsos y la señal azul a la velocidad calculada por el tiempo entre interrupciones. Es fácil ver a simple vista que la señal procedente de la diferencia de tiempo ofrece mayor fiabilidad ya que está oscilando aproximadamente entre valores de 480 y de 460, mientras que la derivada solo alcanza valores de 400 o 500.

3.3 Identificación de los motores y controladores de velocidad

Con todo el sistema correctamente funcionando y con una señal de velocidad de los motores fiable, el siguiente paso era obtener las funciones de transferencia del comportamiento de los motores. Como se deseaba realizar un control de mayor calidad se obtuvieron diferentes funciones para diferentes situaciones y así tener controladores a medida.

3.3.1 Motores en el aire

Para la identificación de motores se utilizó una señal cuadrada periódica que se movía entre los 200 y los -200 de señal PWM para ver así la respuesta de los motores frente a velocidades casi máximas. Para poder obtener la función de transferencia usaremos una aplicación de Matlab llamada "System Identification", por lo que será necesarió guardar los valores de las señales de los motores en variables y luego enviarlas a Matlab, esta función la realizará el bloque "To WorkSpace" que guardará los datos en un array bidimensional, en un eje el tiempo y en el otro el valor de la señal.

La aplicación System Identification es una herramienta muy potente, una vez introducidas las señales se pueden aplicar filtros para eliminar ruidos no deseados, eliminar rangos de la señal, entre otras opciones. A la hora de estimar obtener funciones de transferencia, modelos en espacio de estados, modelos no líneales, etc.

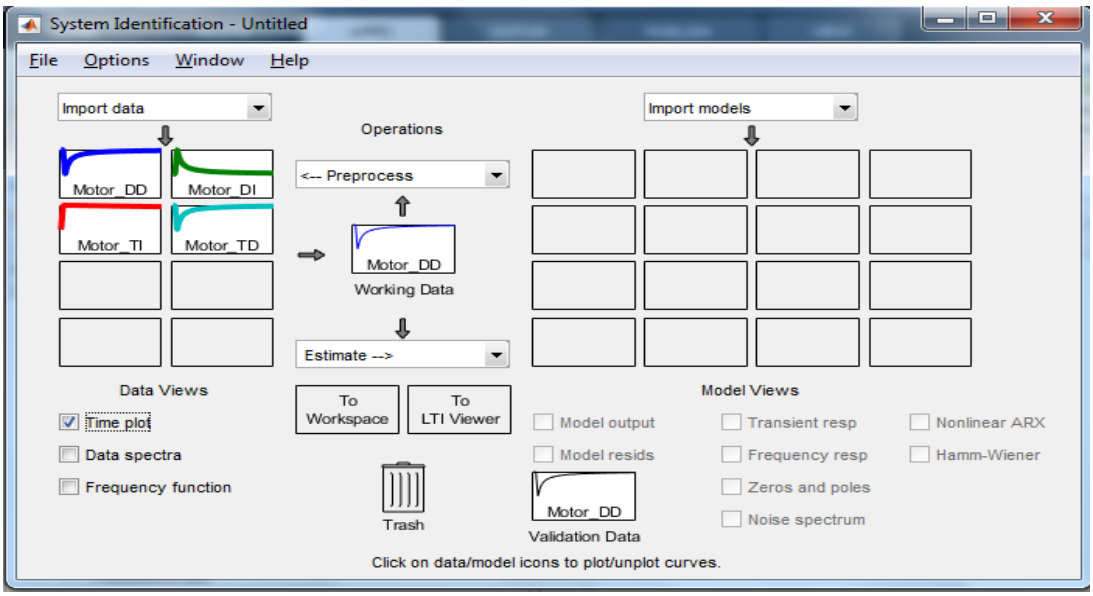

Figura 3. 5 - Interfaz de System Identification

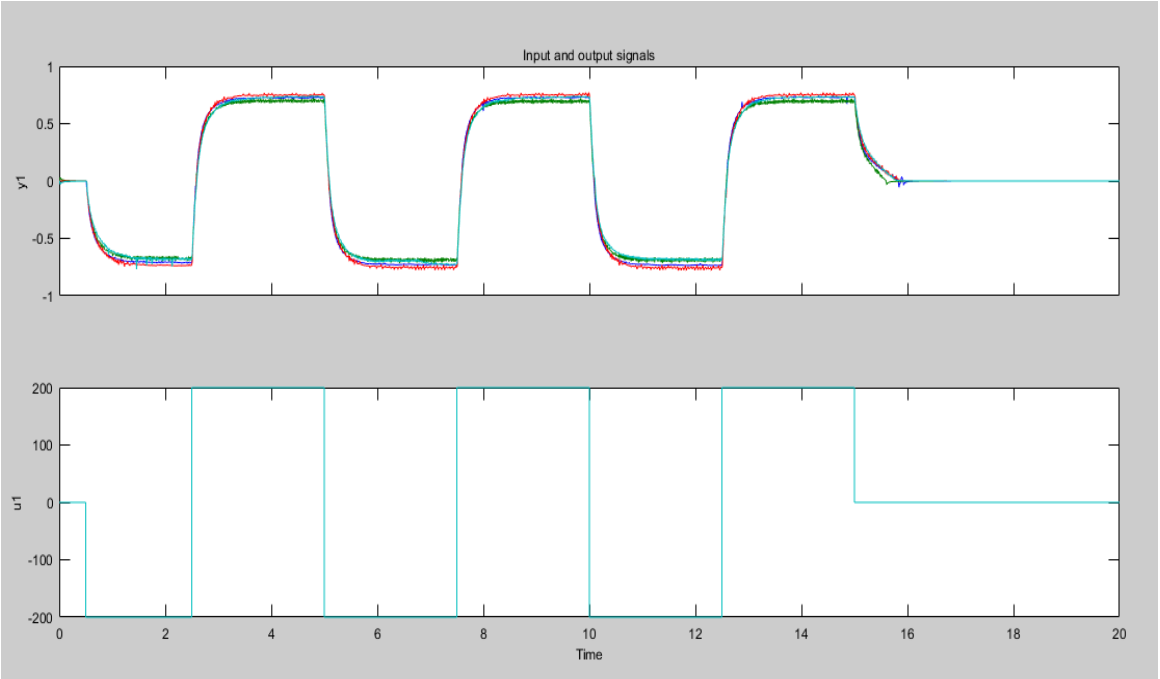

Figura 3. 6 - Señales de los motores durante la identificación

Ahora con todas las señales guardadas, vamos a buscar una función de transferencia lo más exacta posible para cada motor. La aplicación ofrece la posibilidad de escoger la cantidad de polos y ceros que queremos en la función de transferencia, por lo que se probaron para todos los motores tres tipos de funciones:

1. Con dos polos y un cero (tf1)
2. Con dos polos (tf2)
3. Con un polo (tf3)

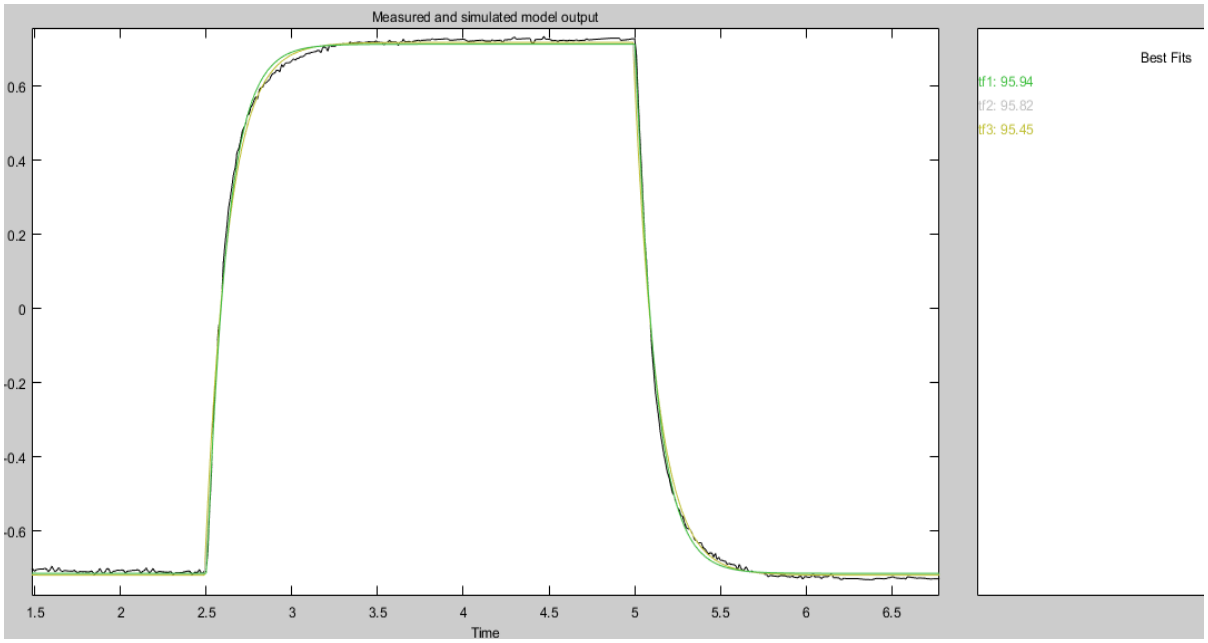

Figura 3. 7 - Comparación de funciones de transferencia (aire)

En la imagen anterior podemos apreciar las respuestas de las tres funciones superpuestas a la respuesta real del motor, siendo tf1 (dos polos y un cero) la función con el mayor porcentaje de similitud, aunque, si nos fijamos en la gráfica, adelanta a la señal del motor y, sin embargo, tf3 (un solo polo) es la que mejor sigue su comportamiento y presenta un 95.45% de exactitud. Para el resto de los motores también se eligieron funciones con solo 1 polo.

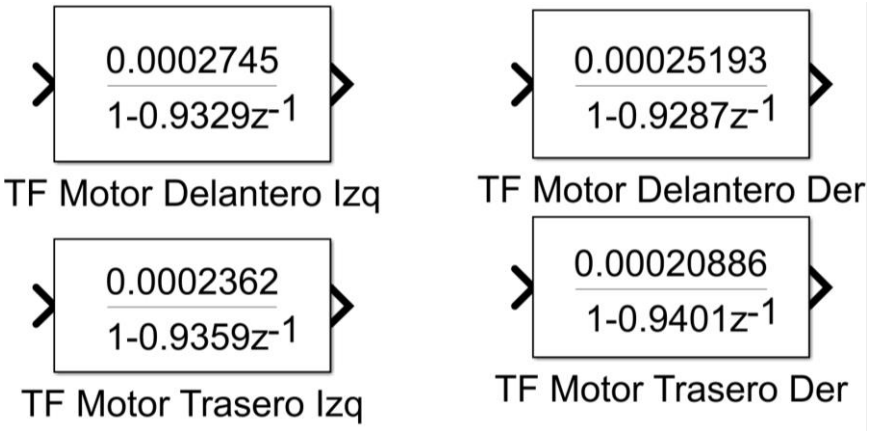

Figura 3. 8 Funciones de Transferencia(aire)

3.3.2 Desarrollo de la aplicación de comunicación Wifi

Para la identificación de motores en el suelo el esquemático a utilizar es prácticamente el mismo, pero nace un nuevo problema: el robot deja de poder estar conectado al ordenador por cable. Necesitamos esta conexión para poder almacenar los datos por lo que establecemos comunicaciones serie por Wifi a través del módulo ESP8266 instalado. Para realizar esta comunicación se desarrollaron dos diseños que necesitan ejecutarse a la vez, uno en el PC y otro en el propio robot.

En primer lugar, tenemos el modelo en Simulink que ejecutamos en el PC que hará de monitor para el usuario y guardará los datos recibidos en variables de Matlab. Para que el esquemático funcione necesitamos que el ordenador esté conectado a la red Wifi que genera el Krobot. Después, durante el primer segundo de funcionamiento enviaremos a la dirección IP del dispositivo (por defecto viene configurada en 192.168.4.1, puerto 23) una señal arranque, en nuestro caso la señal que se envía es letra ‘s’ como una variable tipo uint8. Esto hará que el Krobot sepa que estamos conectados y listos para la recepción y empezará a actuar y enviarnos datos, los cuales recibiremos por la misma dirección wifi y decodificaremos según el protocolo de comunicación que usemos en el programa del Krobot.

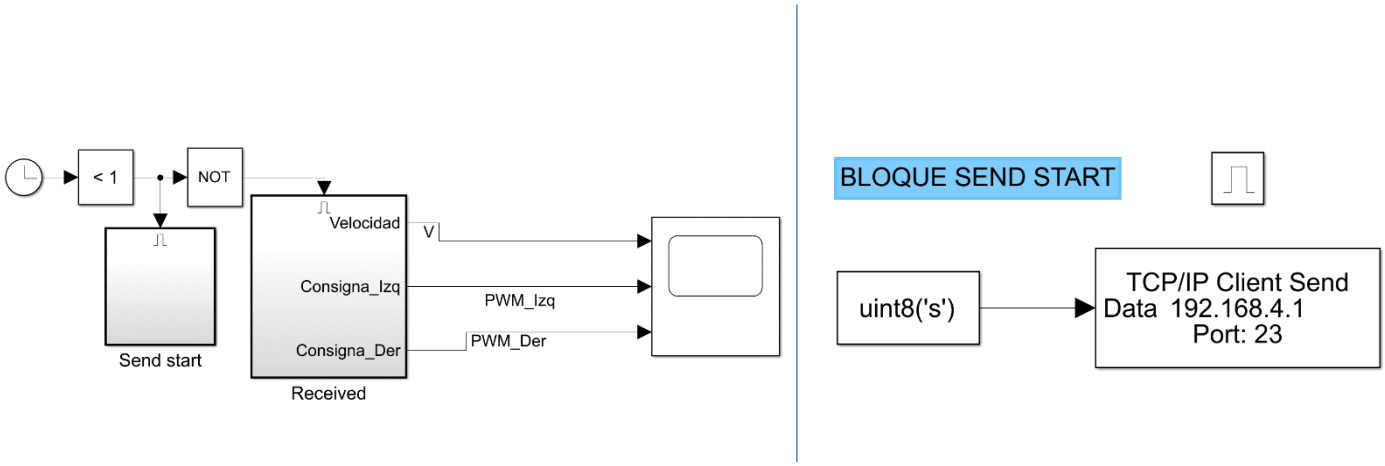

Figura 3. 9 - Aplicación de Comunicación Wifi (PC)

En segundo lugar, tenemos el programa para el Krobot, es decir, el que se ejecutará en la placa Arduino. Este diseño se mantiene a la espera de recibir la señal de arranque para habilitar la consigna de PWM, para asegurarnos que la señal recibida es la correcta se compara con el valor 115 (es el valor de la letra 's' convertida a uint8) y se integra para que no dure un único paso discreto si no hasta que la reseteemos a voluntad posteriormente. Esta misma señal habilita también el envio de señales por serie.

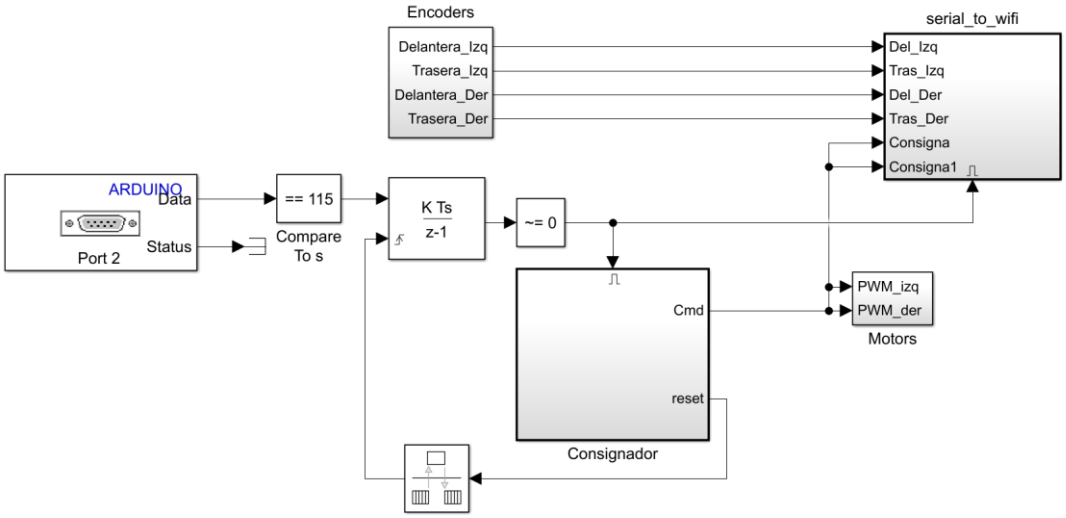

Figura 3. 10 - Aplicación de Comunicación Wifi (Krobot)

En el bloque "Serial to Wifi" convertimos la señal para poder enviarla por el puerto serie de la ESP8266, el primer protocolo que se utilizó era aplicar una ganancia de 100 en el envio y posteriormente convertirlo a a int16, guardando así dos decimales de la señal que considerando la magnitud de los datos utilizados (Los PWMs valen o +200 o -200 y los encoders nos dan señales del orden del decimal o los dos decimales m/s) es una precisión apropiada.

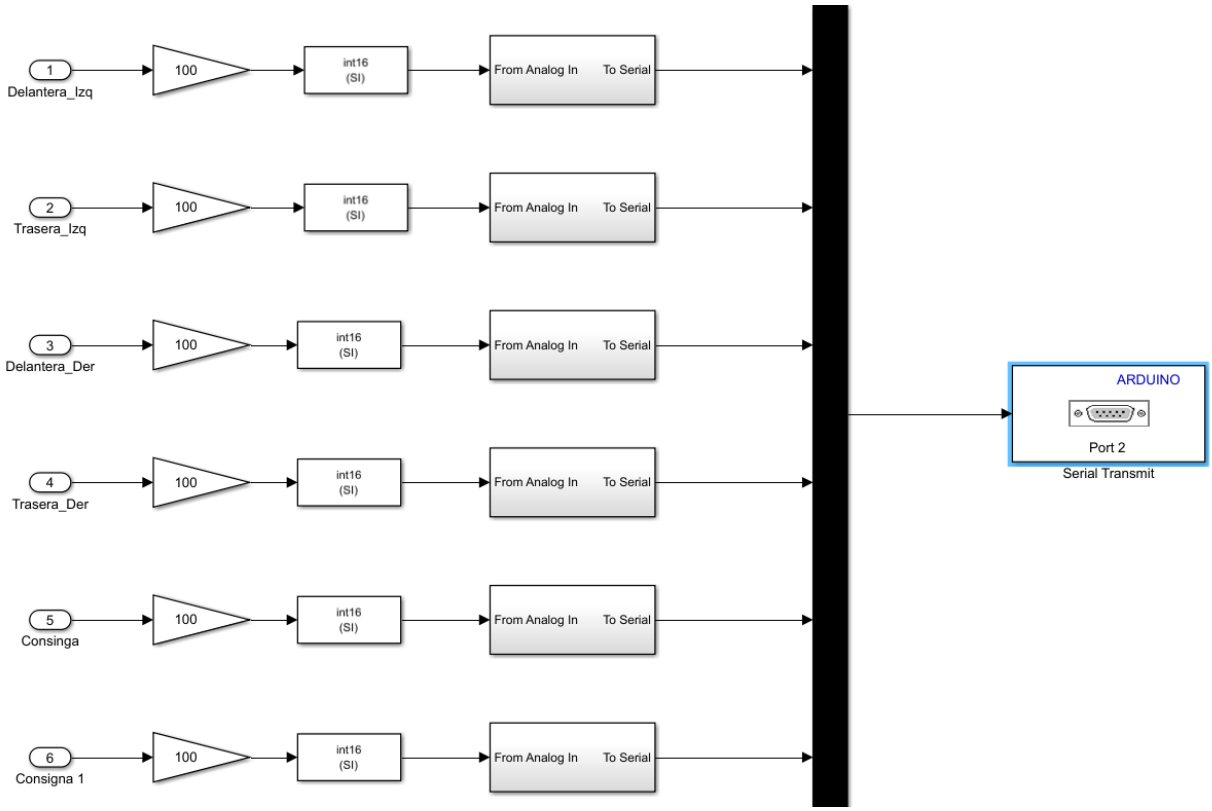

Figura 3. 11 - Primer protocolo de comunicación(Krobot)

Sin embargo, el protocolo no funcionaba a la hora de ejecutar las dos aplicaciones, en el ordenador se recibía errores de TimeOut, en la mayoría de los experimentos, al poco de comenzar la conexión. Tras varios experimentos y muestreo de las diferentes señales se comprobó que la señal de arranque se recibía y que los valores enviados por el Krobot se encontraban dentro de los rangos esperados por lo que debería suponer un problema por desbordamiento de bits, así que se hicieron pruebas enviando las señales por separado, enviando señales con diferentes rangos y se llegó a la conclusión que el problema radicaba en ciertos valores negativos de las señales de los encoders que parecían ser reconocidos como Instrucciones para el módulo ESP8266 en lugar de datos a enviar y esto rompía la comunicación. Para solucionarlo, se optó por aplicar un Offset de valor unitario a las señales para evitar cualquier posible número negativo.

Con estos cambios las comunicaciones comenzaron a funcionar sin ningún error de cuelgue así que se utilizó este mismo protocolo para todas las identificaciones restantes, aunque posteriormente sí que se necesitó cambiar para el envio de las señales de los sensores de consumo.

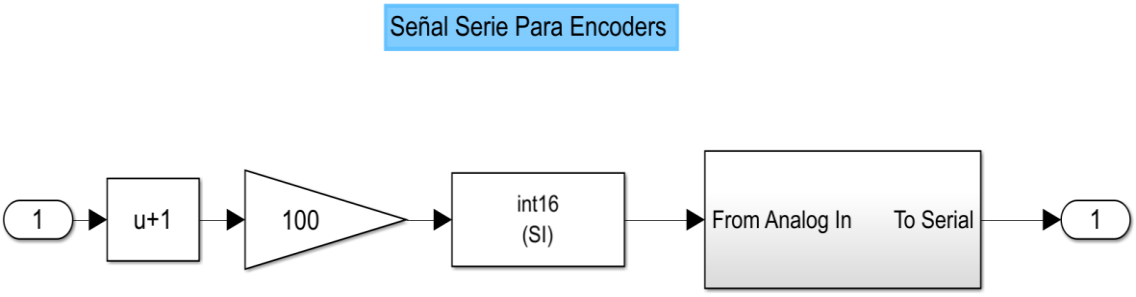

Figura 3. 12 - Modificación del Protocolo de comunicación (Krobot)

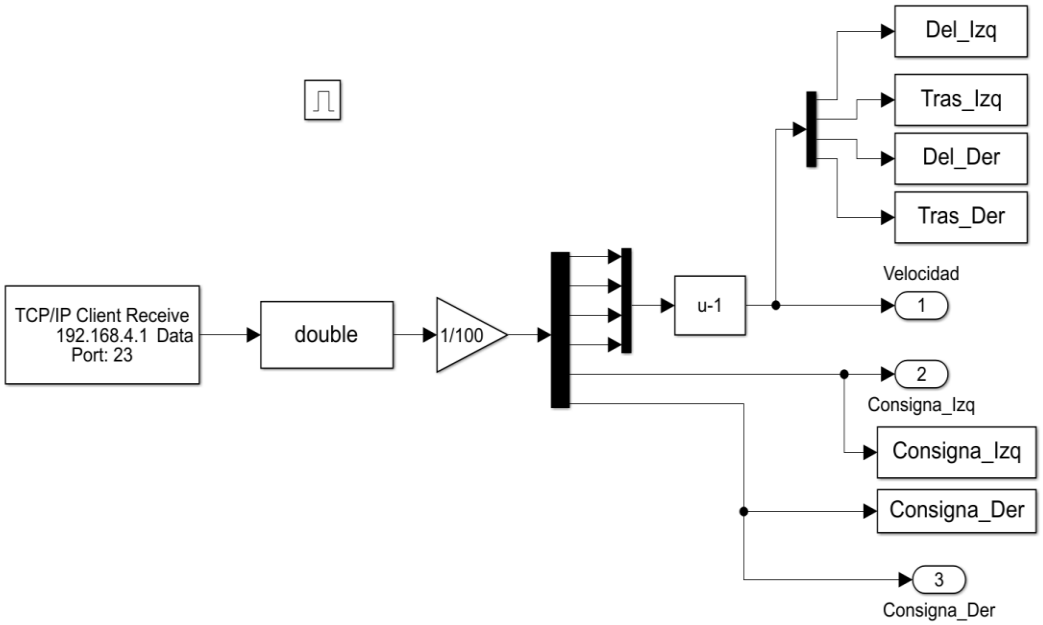

Figura 3. 13 - Protocolo de Comunicación(Pc)

En esta última imagen (Figura 3.13) podemos ver que las señales se reciben por la dirección IP del dispositivo y se aplica la conversión a datos tipo double para que puedan tener decimales (Se envían como enteros), se elimina la ganancia dividendo los valores entre 100 y se resta el offset a las señales de los encoders. Finalmente, todos los valores se guardan en Matlab como Arrays bidimensionales.

3.3.3 Identificación de Motores en Suelo (Lineal)

Utilizando la aplicación Wifi explicada con anterioridad repetimos el experimento de la identificación enviando el mismo tren de pulsos con valores de +200 y -200 de PWM y registrando en Matlab las variables y usando el System Identification. A continuación (Figura 3.14) se muestra la relación entre los valores recibidos y la señal utilizada visto desde el Scope de la aplicación de Comunicación del PC.

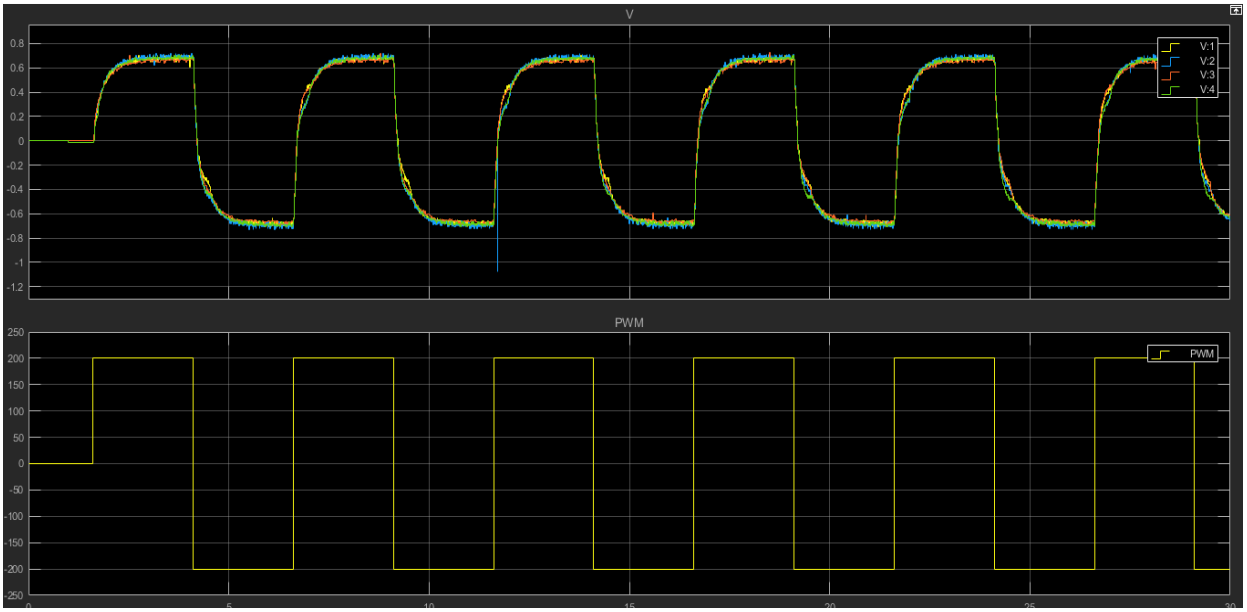

Figura 3. 14 - Señales de Identificación (Suelo)

Podemos ver que al fin la aplicación funciona perfectamente y que recibimos las señales de los cuatro motores, así como la consigna. Ahora vamos a pasar estos datos por la aplicación System Identification.

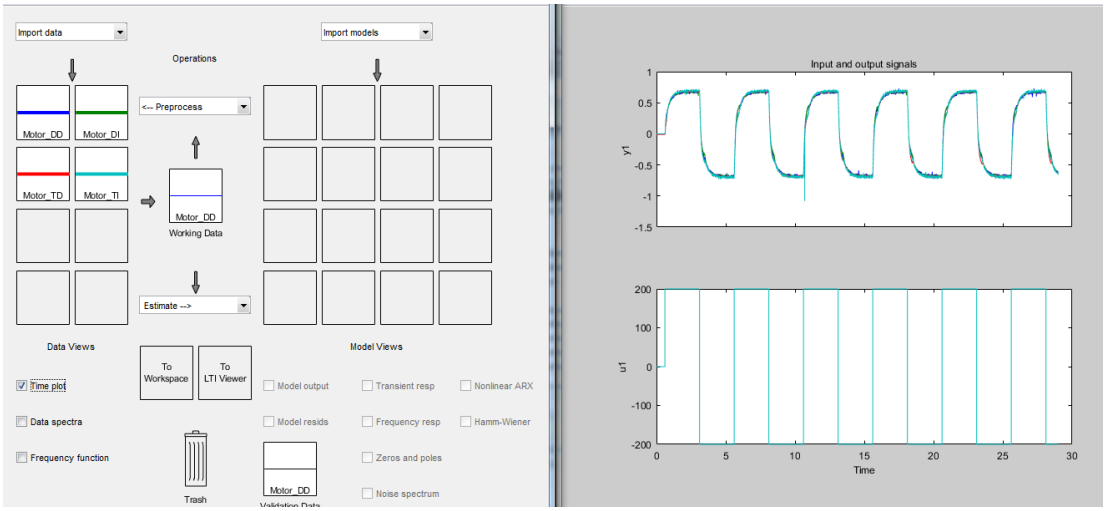

Figura 3. 15 - Datos en el System Identification (Suelo-lineal)

Tal y como se dijo en el apartado 3.3.1, se utilizaron funciones de transferencia de un solo polo y ningun cero debido a su proximidad a la respuesta real del motor y sobretodo porque seguía muy bien la señal sin llegar a adelantarla en ningun momento, así que se ha utilizado el mismo criterio con estas señales. En la figura 3.16 podemos ver la respuesta de las señales obtenidas:

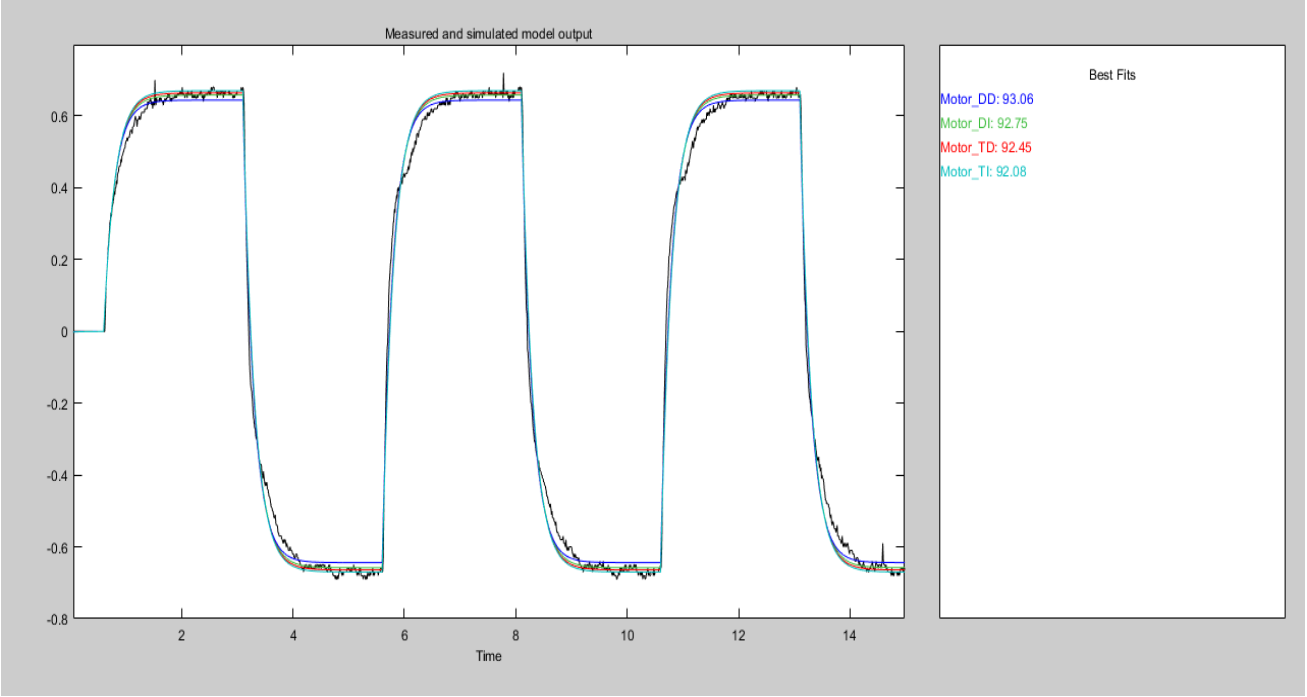

Figura 3. 16 Respuesta de las Funciones de Transferencia (Suelo Lineal)

Si comparamos con la Figura 3.7 podemos apreciar como para los mismos valores de PWM se estan alcanzando velocidades mas pequeñas debido al rozamiento de las ruedas con el suelo. De esta experiencia podemos apreciar la importancia de la realización de multiples funciones de transferencia para poder ofrecer el control más exacto posible.

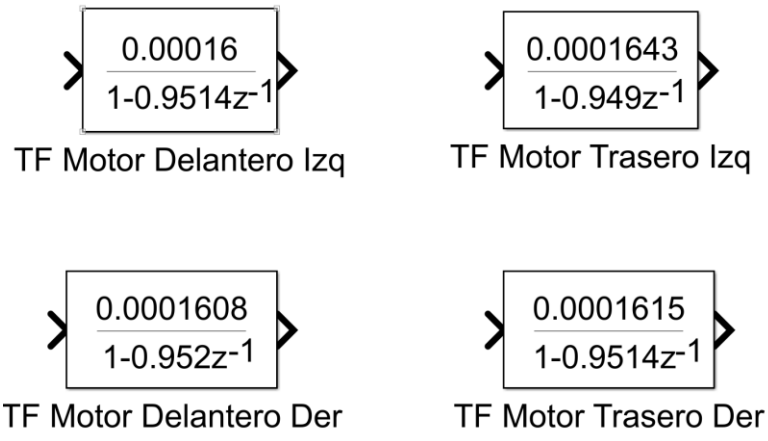

Figura 3. 17 Funciones de Transferencia (suelo-lineal)

3.3.4 Identificación de Motores en Suelo (Girando)

A continuación, modelaremos el comportamiento de los motores cuando el robot gira sobre sí mismo. Resulta especialmente interesante tomar estas medidas ya que, en los giros, mientras mayor sea el sincronismo de los motores eliminaremos posibles fuerzas de deslizamiento entre las ruedas, haciéndolo así más sencillo de controlar. La aplicación a utilizar será la misma que en la identificación en el suelo lineal, salvo que ahora vamos a invertir la señal de uno de los dos laterales, de modo que cuando el lado izquierdo reciba la señal de +200 de PWM el lado derecho tendrá -200 de PWM. A continuación, vemos los resultados en Matlab:

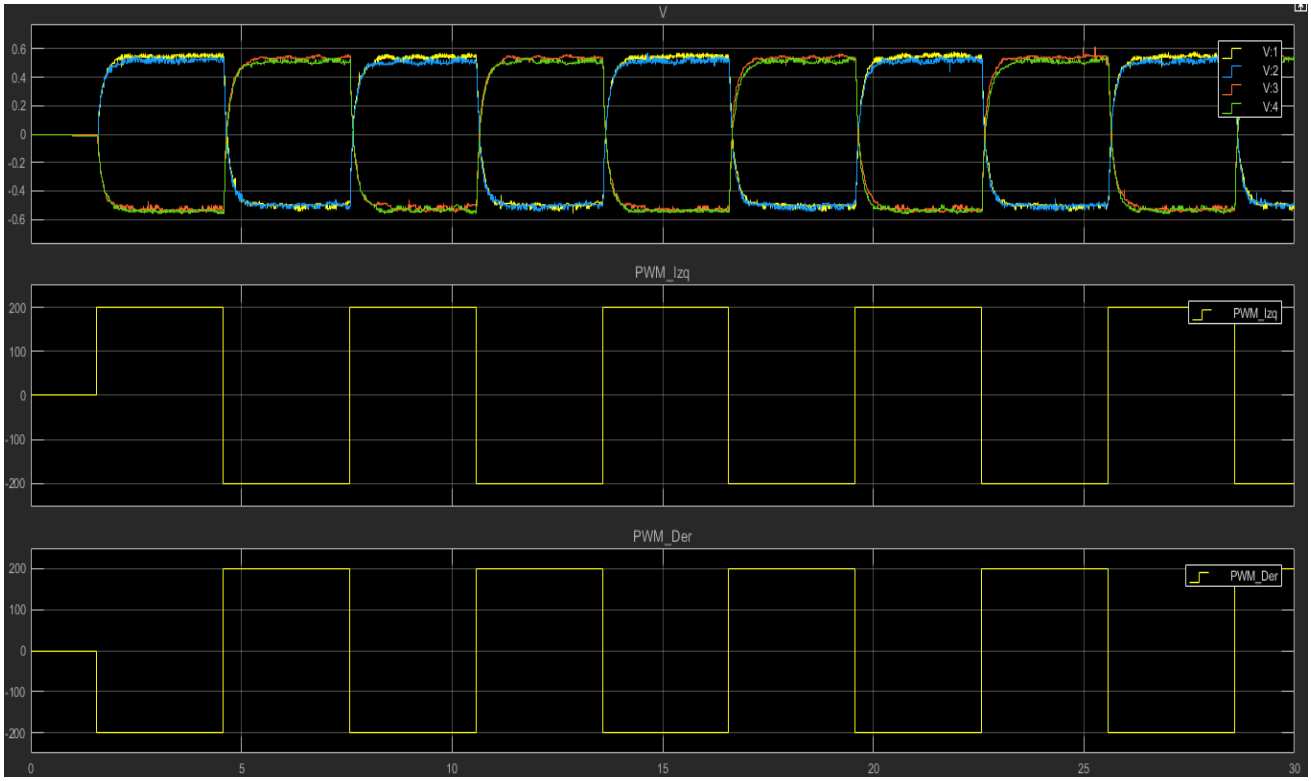

Figura 3. 18 - Señales de Identificación (Giro)

Es fácil de apreciar que las señales 1 y 2 (señales amarilla y azul) de las velocidades correspondientes con los motores izquierdos siguiendo la gráfica del PWM_Izq y que las otras dos (señales verde roja y verde) están siguiendo la del PWM_Der. También podemos fijarnos en que esta señal está ofreciendo incluso menos velocidad que cuando se movía de forma lineal. Vamos a enviar los datos al System Identificator y obtener los datos de las funciones de transferencia.

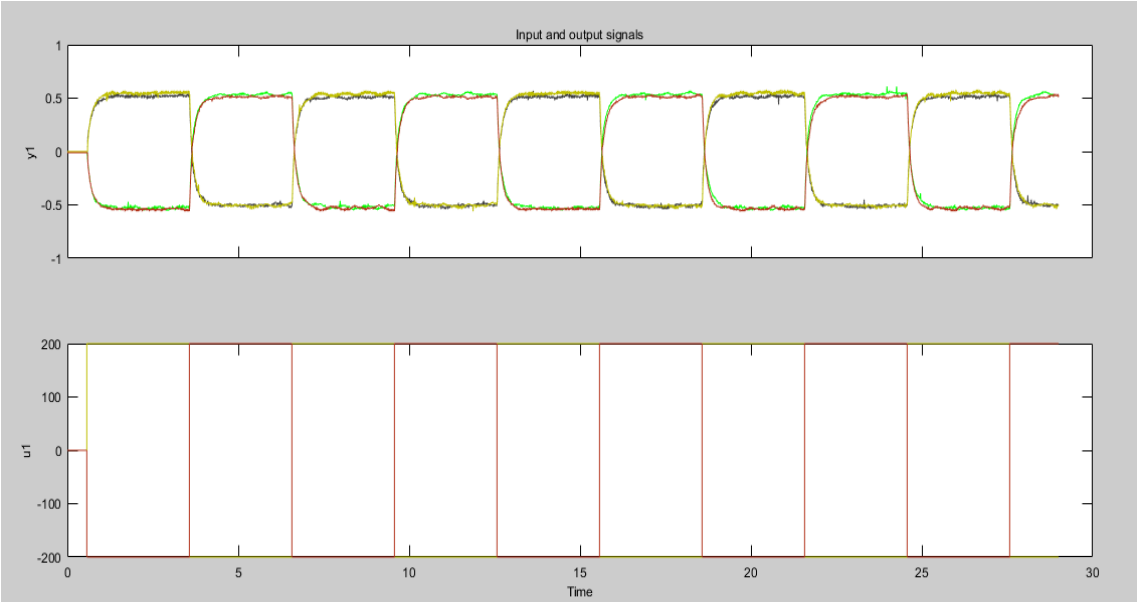

Figura 3. 19 - Datos en el System Identification (Suelo-giro)

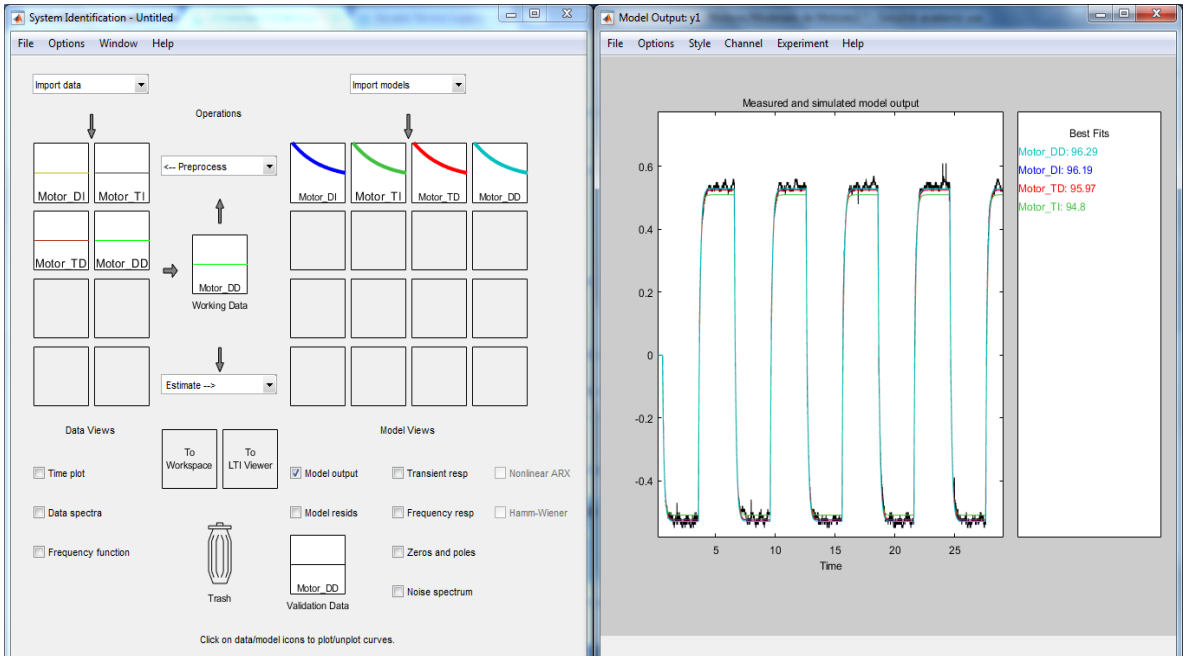

Figura 3. 20 - Respuesta de las Funciones de Transferencia (Suelo Giro)

Las funciones de transferencia obtenidas son de nuevo de primer orden, sin embargo a la hora del giro se hace más evidente la diferencia en la velocidad máxima que puede alcanzar cada motor (figura 3.20)

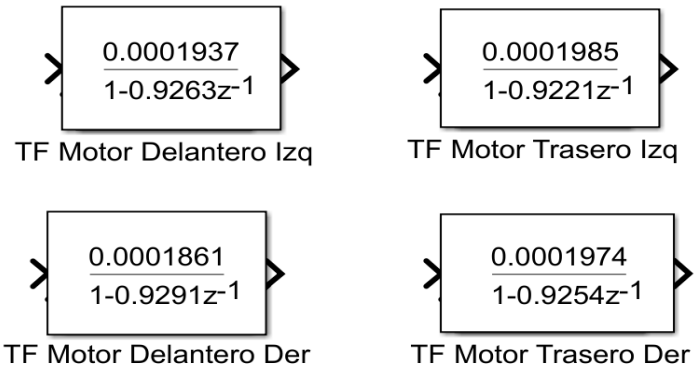

Figura 3. 21 - Funciones de Transferencia (Suelo Giro)

3.3.5 Identificación de Motores en Tierra (Lineal y Giro)

Como el objetivo del Krobot es ser un robot móvil que se pueda utilizar en exteriores también se han repetido las experiencias de identificación en una explanada (figura 3.22). La aplicación que se utilizó es idéntica a la utilizada a la de suelo de interiores, con el mismo protocolo de comunicación Wifi y las mismas señales. A continuación, las señales obtenidas:

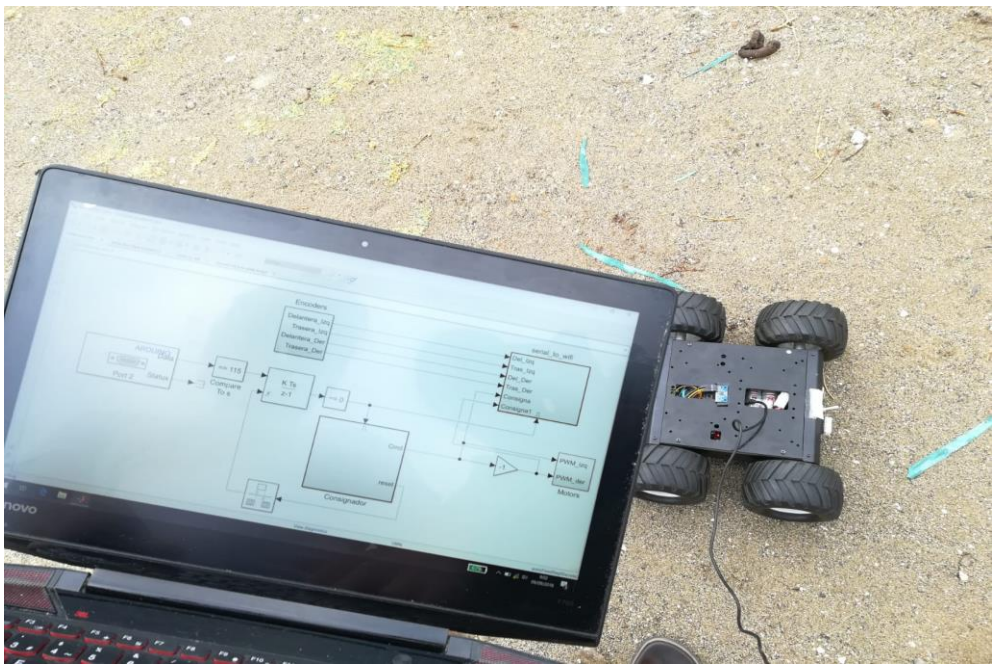

Figura 3. 22 - Krobot en Exterior

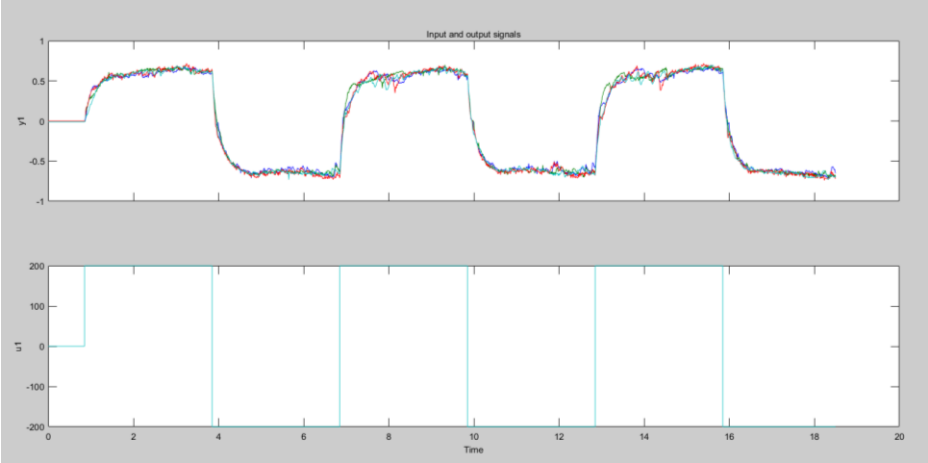

Figura 3. 23 - Señales Identificación Tierra Lineal

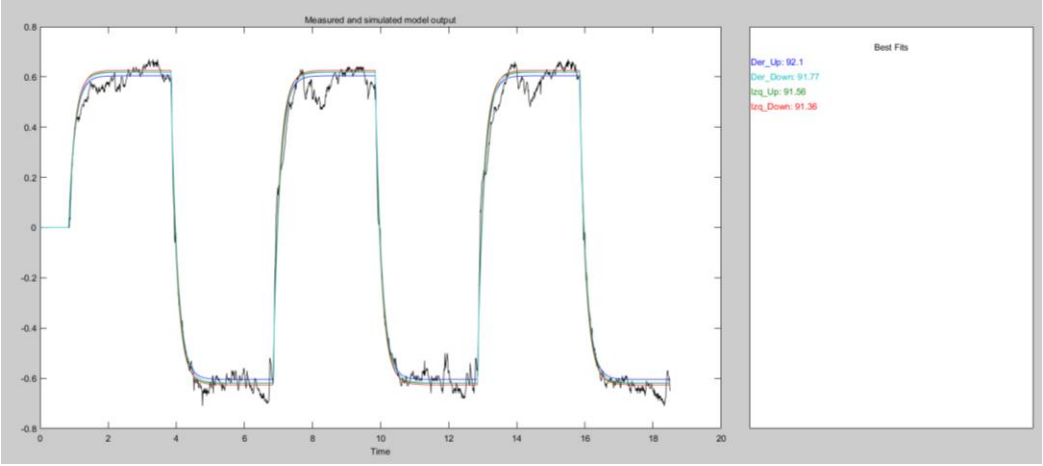

Figura 3. 24 - Respuesta de las Funciones de Transferencia (Tierra Lineal)

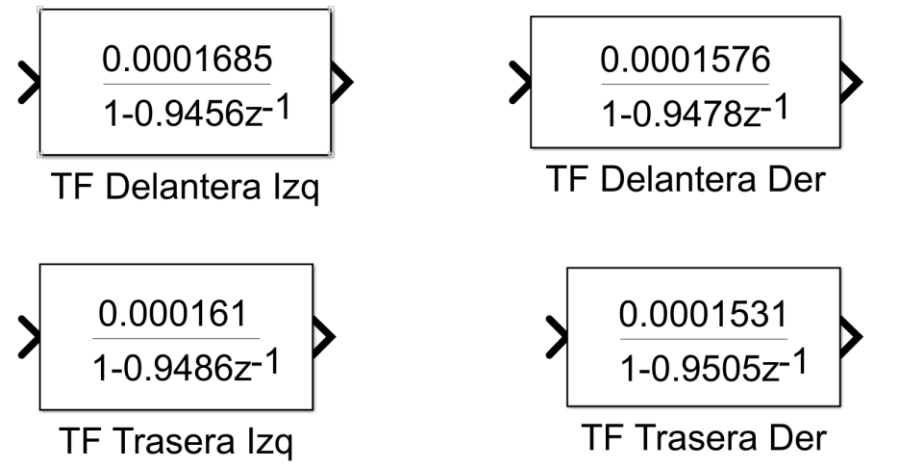

Figura 3. 25 Funciones de Transferencia Tierra Lineal

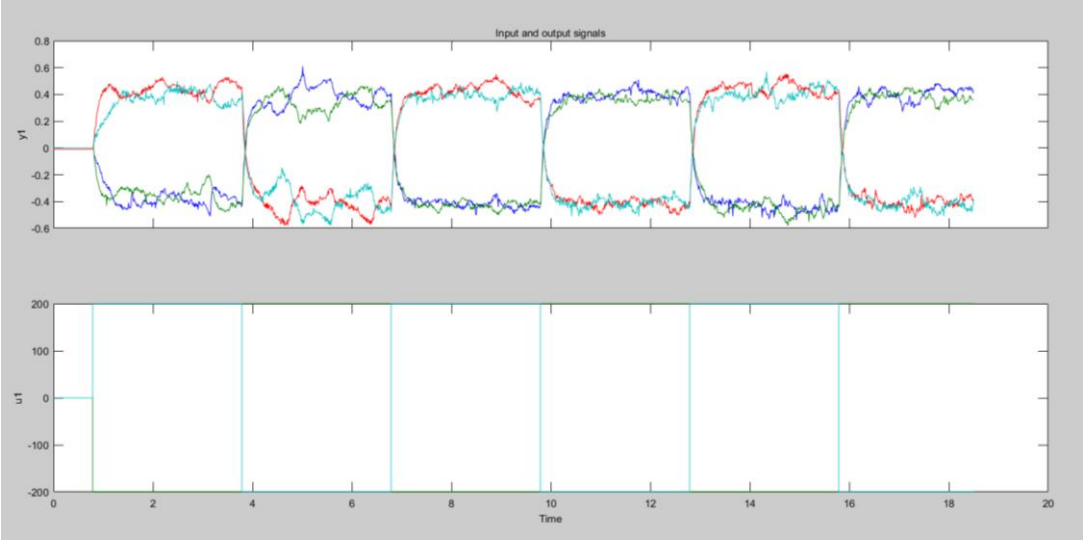

Figura 3. 27 - Respuesta de las Funciones de Transferencia (Tierra Giro)

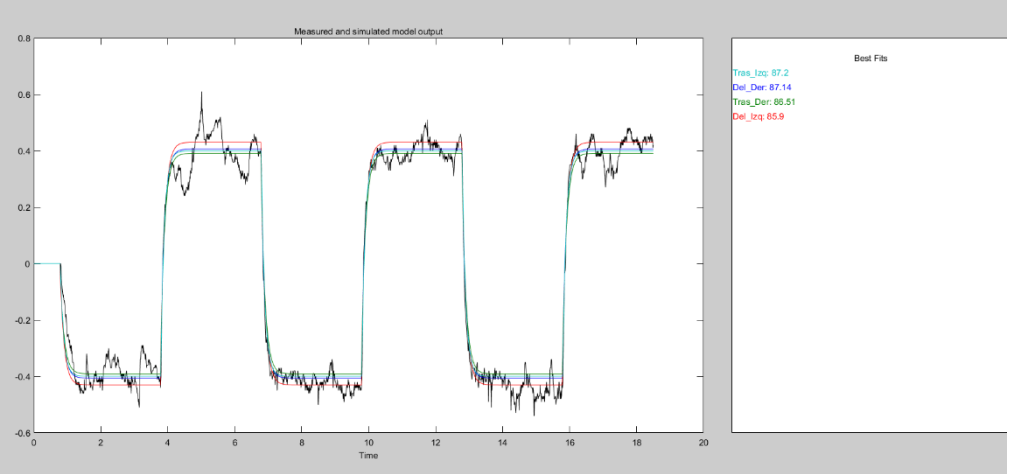

Figura 3. 26 - Respuesta de las Funciones de Transferencia (Tierra Giro)

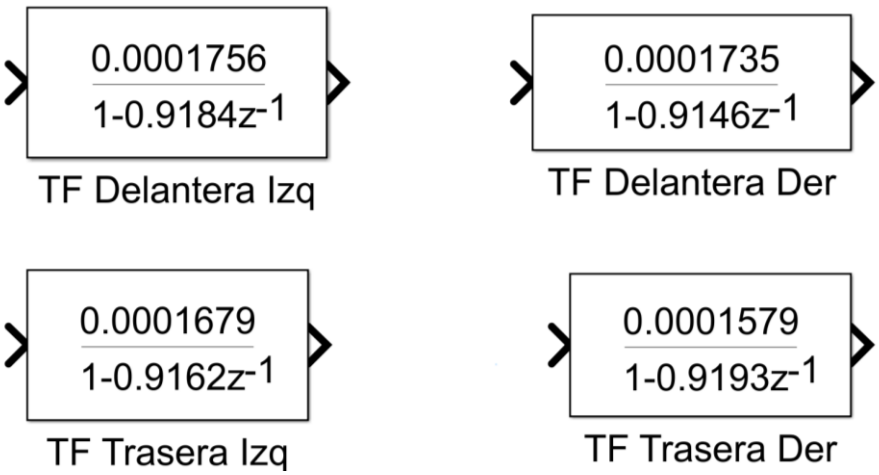

Figura 3. 28 - Funciones de transferencia (Tierra Giro)

Podemos comparar estas señales con las obtenidas en la identificación en suelo de interiores, vemos que estas nuevas señales son mucho más ruidosas y muy poco constantes en comparación con sus análogas, realzando así la importancia de que tengan controladores propios si se desea poder controlar correctamente al Krobot como robot móvil de exteriores.

3.3.6 Identificación de Motores en Tierra (En Cuestas)

Por último, aprovechando que se estaba haciendo la identificación en tierra se decidió realizar dos identificaciones más, con el robot en exteriores en cuesta arriba y en cuesta abajo. La idea de esta identificación era combinarla con el sensor inercial para que el control funcionase de forma automática. La identificación se repitió en varias cuestas de diferentes inclinaciones aunque solo se acabó guardando una de las funciones porque el resto resultaron ser demasiado inclinadas y el robot no respondía bien, o muy poco y apenas había diferencia con el modo Tierra Lineal. Solo se muestran las señales de subida, por la inmensa similitud en los resultados con los de bajada.

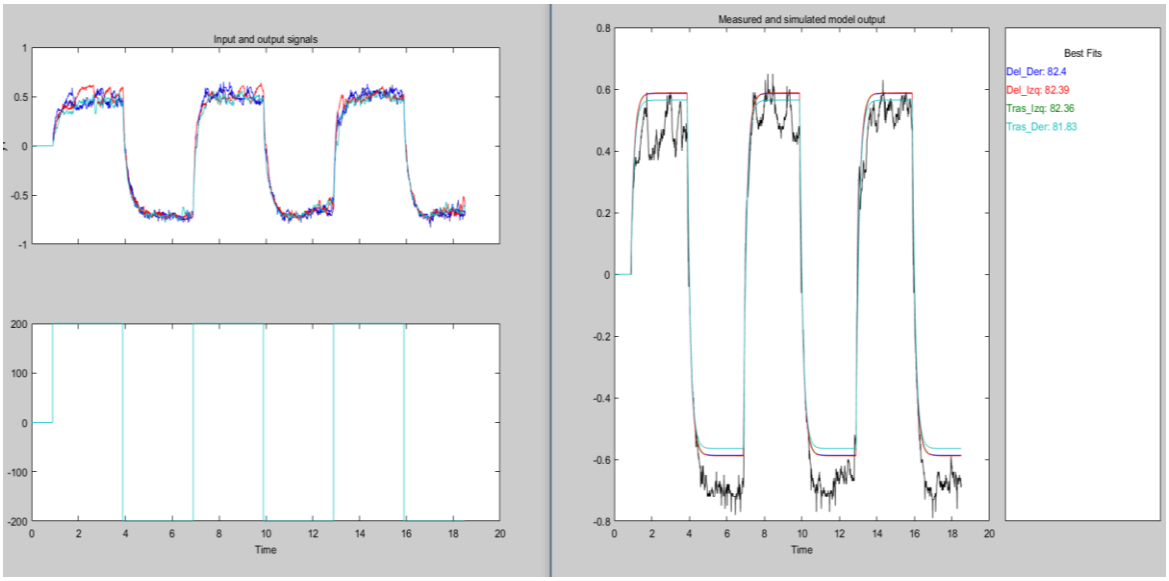

Figura 3. 29 - Identificación Tierra Subida

Se aprecia en las respuestas de los motores la dificultad para subir con velocidades más bajas y ruidosas y la facilidad para bajar con velocidades bastantes más altas y sin apenas perturbaciones, sin embargo, las funciones de transferencias no pueden expresar esta diferencia.

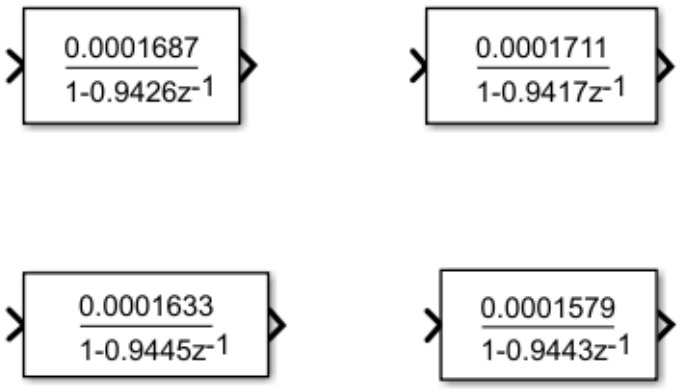

Figura 3. 30 Funciones de Transferencia Subida

Capítulo 4 – Controladores de velocidad

4.1 Introducción

En este capítulo vamos a diseñar controladores para las funciones de transferencia que simulan el comportamiento del robot para poder conseguir una respuesta sincronizada de todos los motores.

4.2 Bucle de Control

En primer lugar, tenemos que definir el bucle de control del sistema. La consigna que vamos a utilizar ya no es el PWM si no directamente la velocidad que se desea que nos den los motores. Por otro lado, las funciones de transferencia serán todas las obtenidas durante los experimentos del capítulo anterior. Para los controladores usaremos los bloques PID de Simulink que incorporan una función de sintonización automática (Tune) que nos permite obtener los parámetros del controlador mediante una interfaz gráfica donde ajustamos la robustez y velocidad de la respuesta deseada. Finalmente bastará con cerrar el bucle de control con realimentación de la consigna con la salida y empezar a utilizar la función Tune.

4.3 Sintonización

Es importante destacar tres criterios fundamentales a la hora de la sintonización de los motores: En primer lugar, las 4 señales (de cada situación) deben estar en completa sincronía ya que basta con que uno de los motores este ligeramente desfasado con el resto para que esto provoque fallos en la localización del robot, por ejemplo, haciéndole girar levemente durante y perdiendo la orientación inicial. En segundo lugar, debemos marcar que la salida del controlador debe saturarse entre +-255 que son los valores de PWM que aceptará el motor. Por último, debemos recordar que los componentes utilizados no son ideales, es decir que las respuestas deben tener tiempos de estabilización y de subida con valores cercanos al medio segundo, pues valores inferiores no serán posibles o podrían provocar deterioro de los motores.

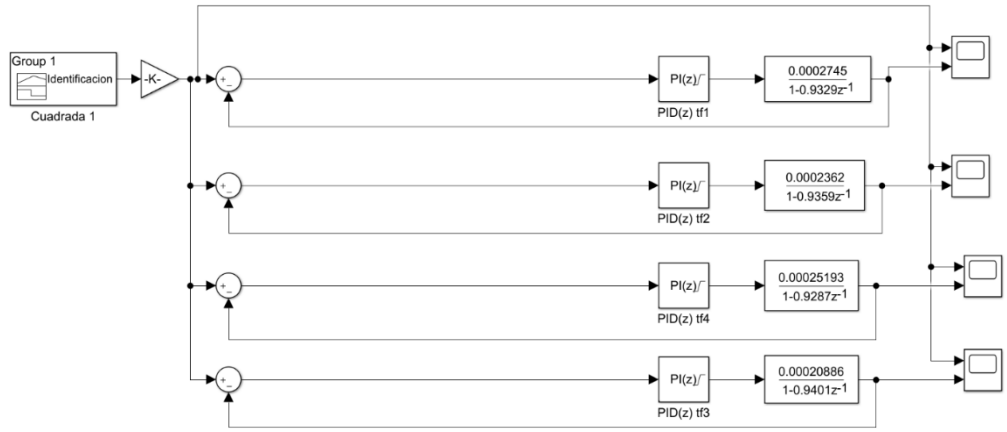

Figura 4. 1 - Bucle de Control (TF aire)

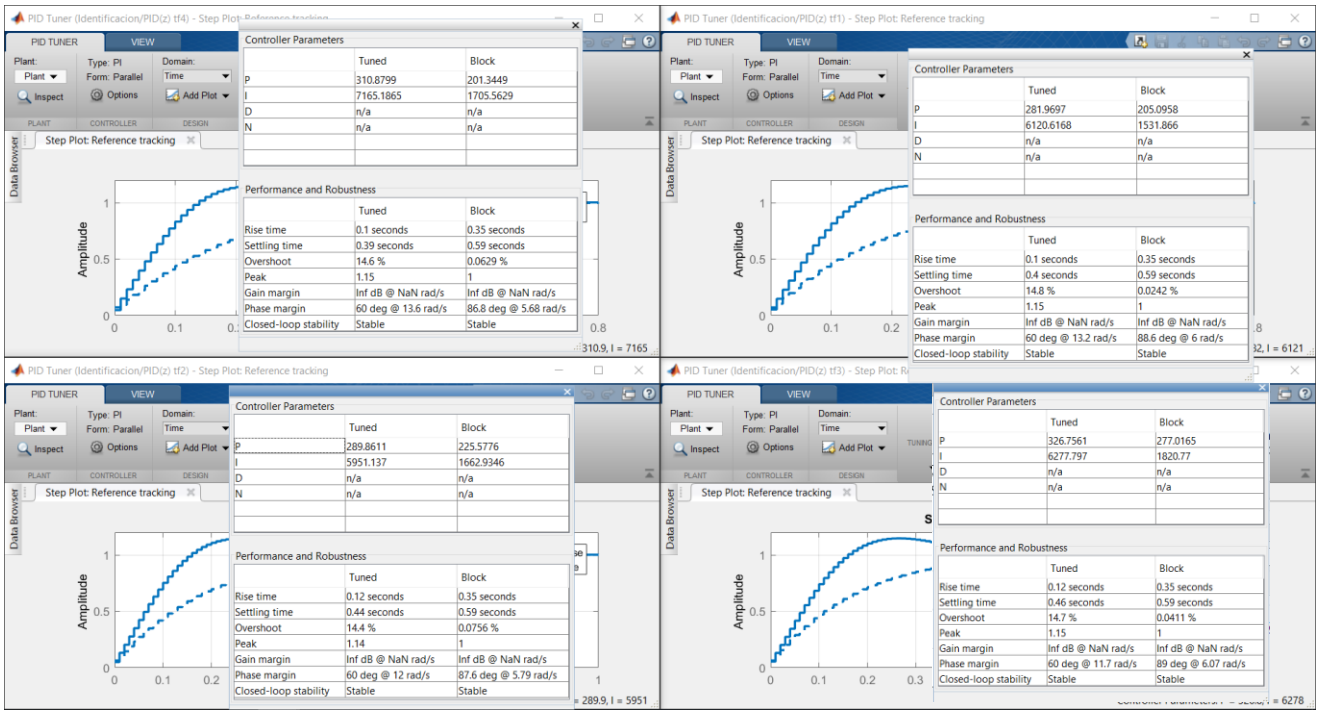

Figura 4. 2 - Sintonización de la Respuesta (aire)

En la figura 4.2 podemos apreciar que se ha obtenido para las cuatro funciones señales que ofrecen un comportamiento (Block response) de primer orden donde se han fijado los mismos parámetros de subida (0.35s) y de estabilización (0.59s). Aunque si que es cierto que existe un ligero sobreoscilación variante entre todas, estamos hablando de valores muy pequeños 0.00576%, por ejemplo, que si lo aplicamos a una señal deseada de 0.4 m/s estaríamos una señal de 0.400023, esta diferencia de valor es completamente despreciable. A continuación, vemos la diferencia entre las respuestas frente a las consignas con y sin controlador

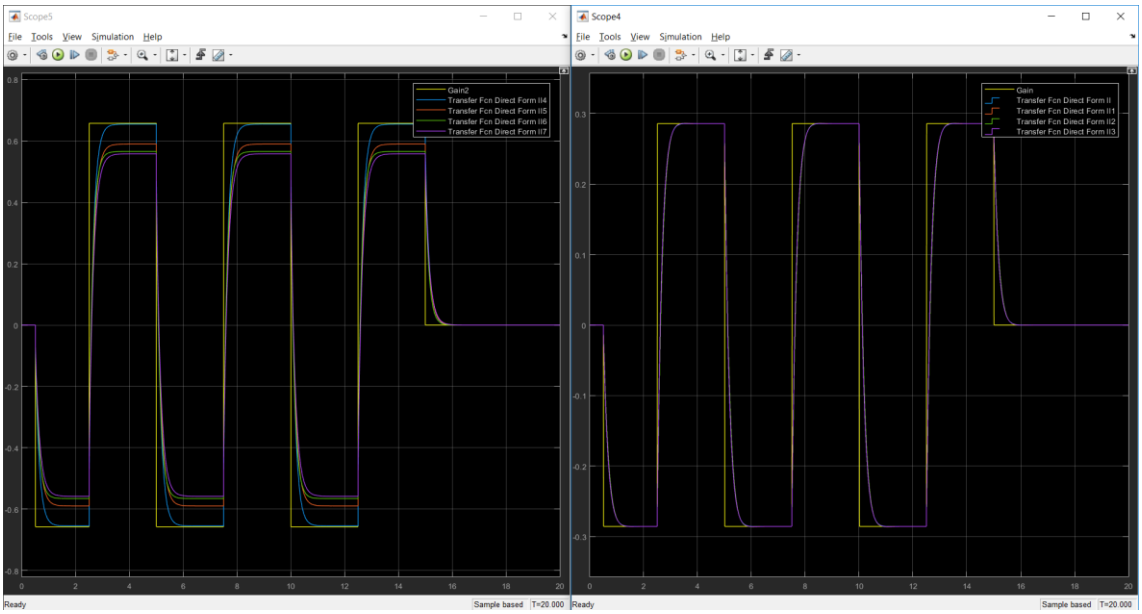

Figura 4. 4 - Comparativa de Con y Sin Controladores (aire)

En la figura 4.3 es evidente el cambio, se ha pasado de unas señales que no iban a la misma velocidad y que no alcanzaban los mismos valores de estabilización a unas señales que siguen la consigna de un modo ideal. Para el resto de los controladores se siguieron los mismos criterios y a continuación se van a mostrar solo las imágenes de las comparativas, ampliando el escalón inicial que es la zona de mayor interés para ver el seguimiento de la señal y la sincronía:

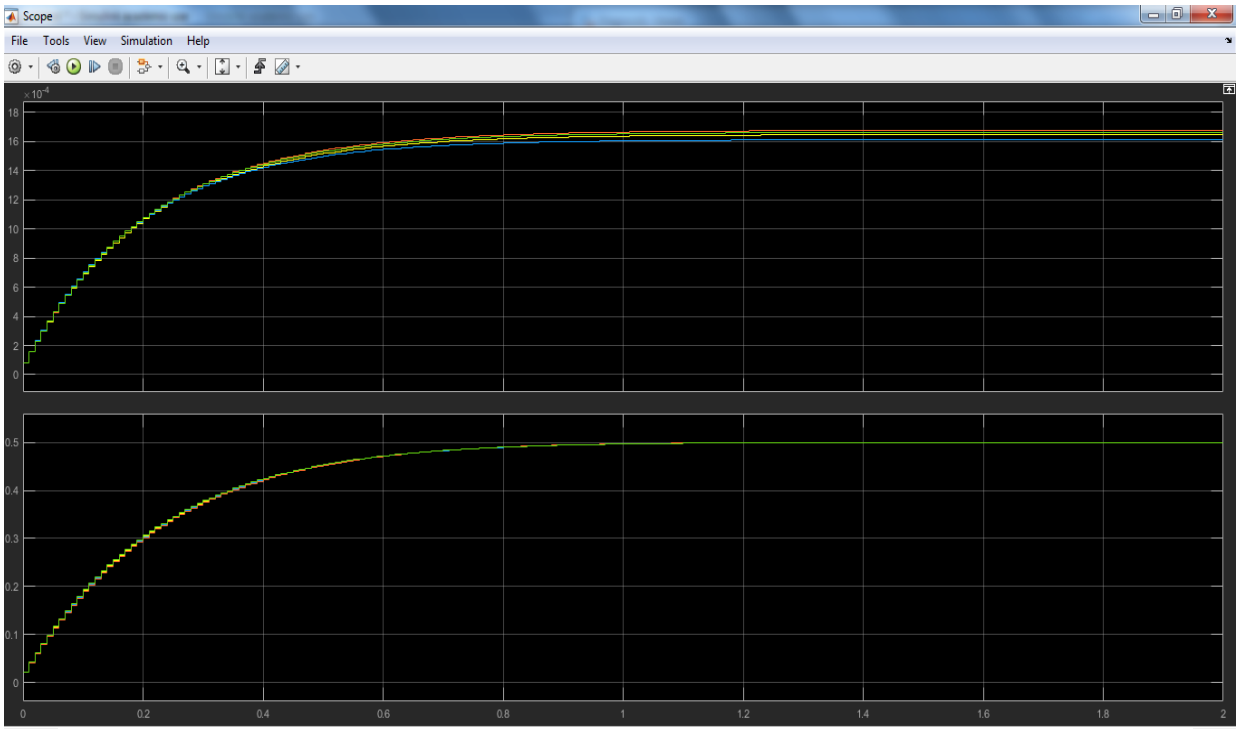

Figura 4. 3 Comparativa de Con y Sin Controladores (Suelo, Lineal)

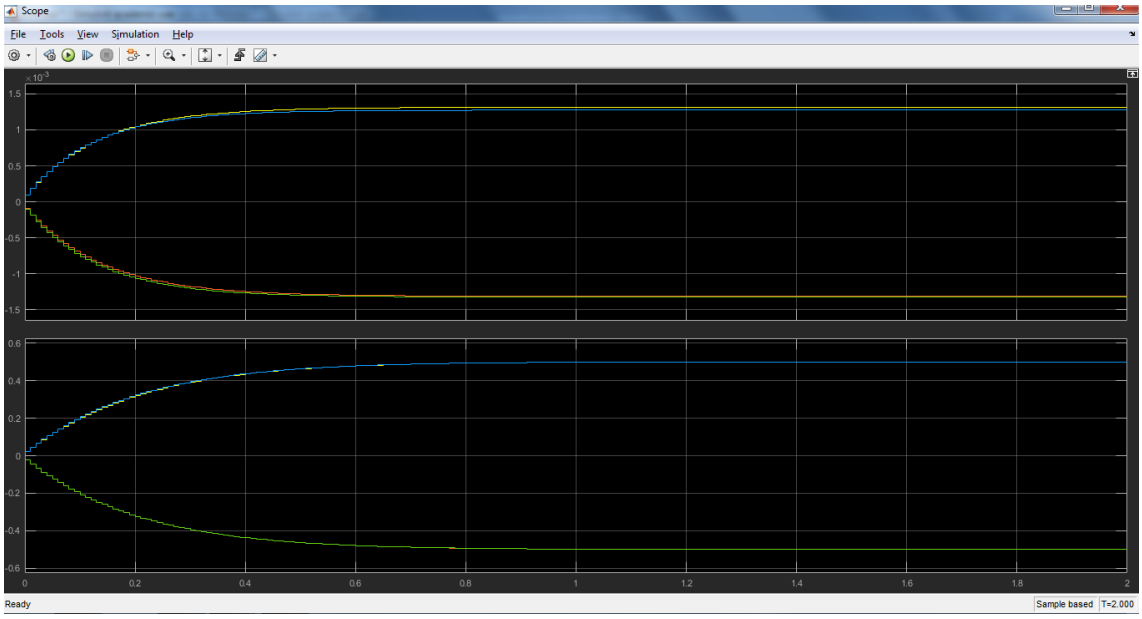

Figura 4. 5 Comparativa de Con y Sin Controladores (Suelo, Giro)

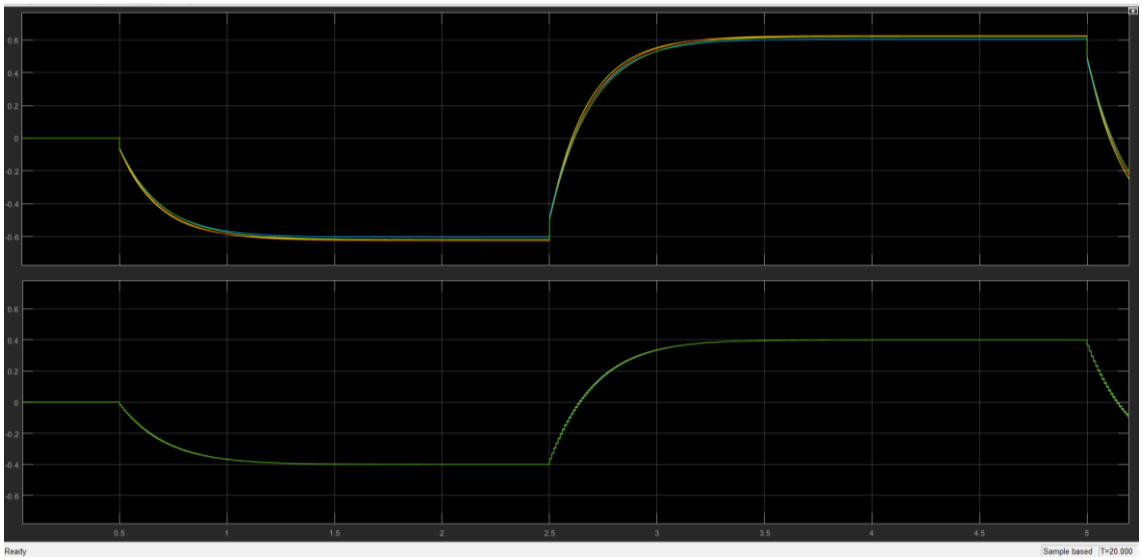

Figura 4. 6 - Comparativa con y sin Controladores (Tierra, Lineal)

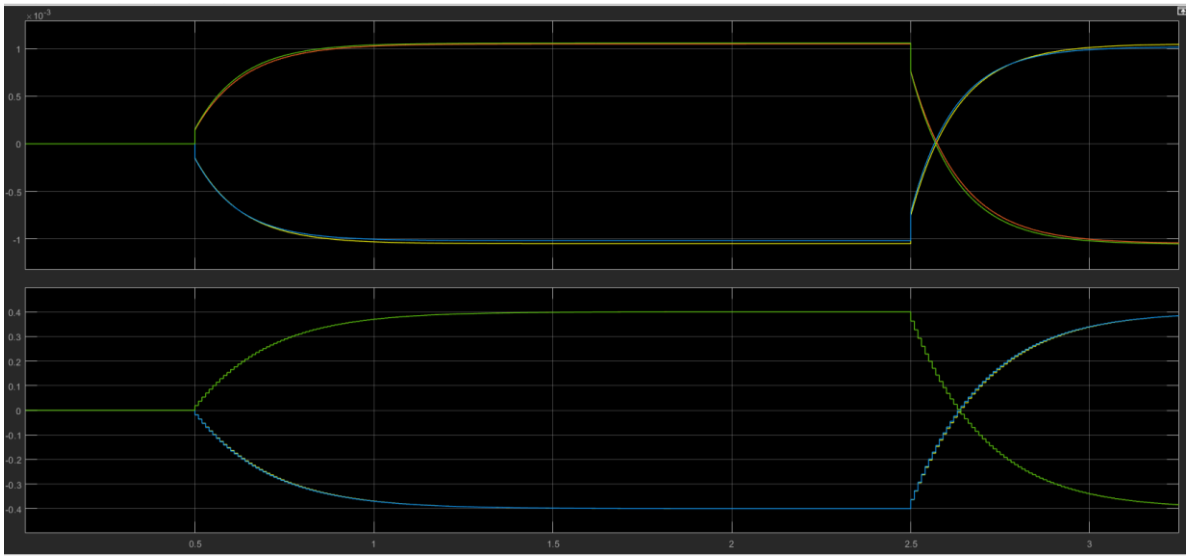

Figura 4. 7 - Comparativa con y sin Controladores (Tierra, Giro)

4.4 Implementación de los controladores en el sistema

Con todos los controladores listos es el momento de introducirlos en el diseño principal para que regulen el funcionamiento de nuestros motores, además también se han implementado las funciones de transferencia obtenidas para poder simular el funcionamiento del robot desde Simulink.

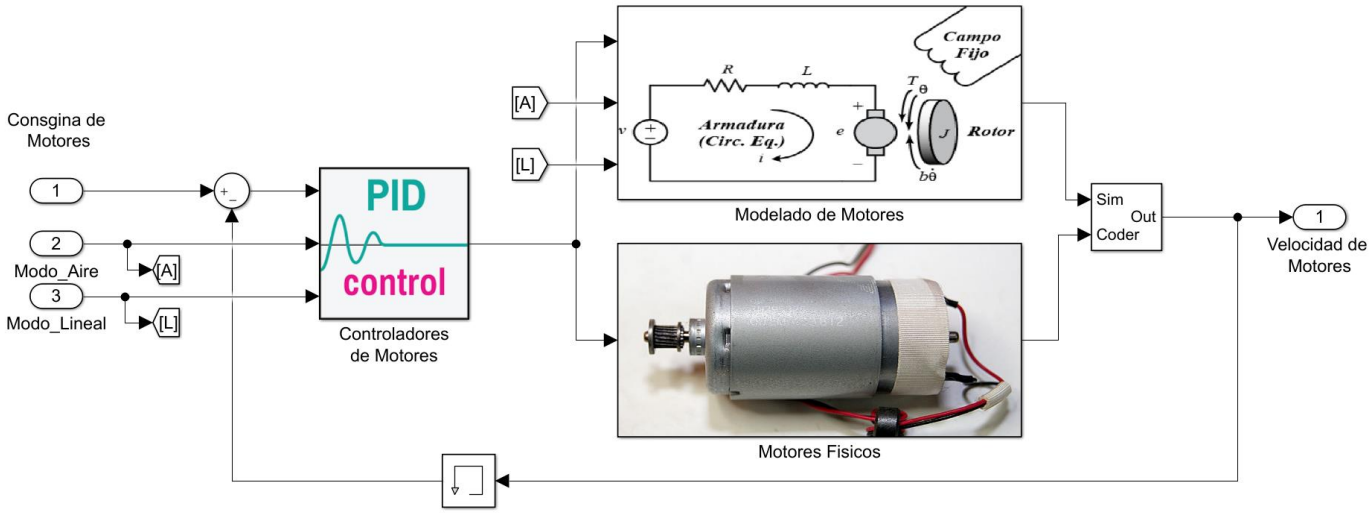

Figura 4. 8 Esquemático Principal con los Controladores

Caben destacar dos bloques esenciales para la implementación de estos elementos al diseño, en primer lugar, el bloque de memoria que vemos en la realimentación de la consigna, es necesario añadirlo, ya que sin él Simulink es incapaz de procesar la señal en tiempo real porque intenta modificar la salida al mismo tiempo que la entrada, este bloque retrasa la operación evitando el bucle algebraico. En segundo lugar, el bloque de Control de Ambiente que recoge las salidas de los modelos teóricos y de los motores y las multiplexa en función del modo de ejecución, cuando el programa se esté simulando, dará como salida la de las funciones de transferencia mientras que cuando estemos trabajando directamente en el robot el valor de la velocidad vendrá dado por las lecturas de los encoders.

Para que los controladores funcionen correctamente, se ha implementado un sistema de Switches que cambia entre controladores (y entre funciones de transferencia para las simulaciones) para adaptarse en cada momento. Actualmente, existen 4 variables que determinan que controlador se va a aplicar en cada instante:

1. Modo Aire: Se trata de una constate que, si su valor es mayor que cero, estamos determinando que el robot se encuentra en suspensión, es decir, sus ruedas no están tocando el suelo. En este caso usaremos los controladores de la primera identificación, en caso contrario, dependerá del resto de variables.
2. Modo Externo: Con esta constante, indicamos al robot si se encuentra en una superficie de tierra. Si es así, usará controladores de Tierra Lineal o Tierra Giro. Si en su defecto, enviamos un 0, el sistema usará los controladores de Suelo.
3. Modo Lineal: Ya que sabemos que las consignas de avance dan la misma velocidad a todas las ruedas y que las de giro dan distintas velocidades al lado derecho que al lado izquierdo podemos usar esta relación para saber si estamos girando o no. O lo que es lo mismo, si la diferencia de las señales derecha e izquierda es 0, es que nuestro robot avanza linealmente y de lo contrario está girando.
4. Inclinación: Esta señal procede del sensor inercial IMU, utilizando el ángulo de Pitch (ángulo del sistema en el eje Y) podemos saber si estamos subiendo o bajando una cuesta y en ese caso podemos usar los controladores diseñados para esta situación

A continuación, vamos a explicar el mapa de switches paso a paso:

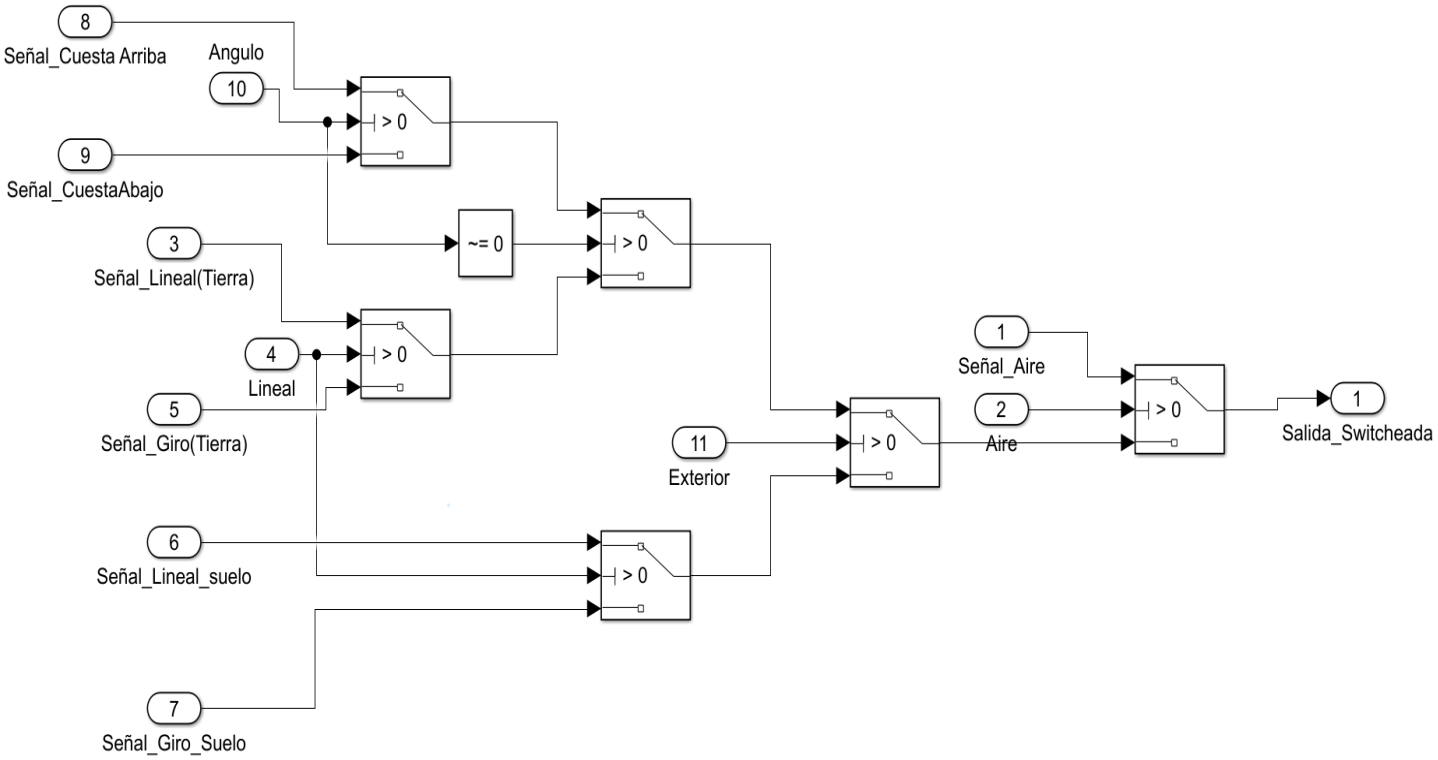

Figura 4. 9 - Mapa de Switches de controladores

Para una mayor comprensión del sistema de toma de decisiones de la figura anterior se va a explicar el funcionamiento del bloque derecha a izquierda. La primera decisión que encontramos es si el sistema está en suspensión o no, si es el caso, usaremos los controladores de aire si no, debemos seguir hacia atrás. Ahora comprobamos si el sistema está en el exterior o en suelo liso, si estamos en suelo liso solo tenemos que saber si estamos girando o no y ya tendremos el controlador decidido. Sin embargo, para el caso de exteriores tenemos que fijarnos también en la señal angular, para saber si estamos en cuesta, actualmente se compara la variable de ángulo con 0 y en caso de sea distinta usan controladores de cuesta, además será necesario determinar unos límites para darle a la señal un valor de 1 en cuestas de pendientes positivas y -1 en pendientes negativas. En caso de que no estemos en cuesta, usaremos controladores de tierra de giro o avance recto usando la variable “lineal”.

Capítulo 5 – Sensores de Consumo e incorporación de la IMU

5.1 Introducción

Los sensores de Consumo de Corriente ACS712 se instalaron en el Krobot con el fin de tener más información del estado del sistema, y estudiar sus posibles integraciones como por ejemplo, estudiar el consumo bajo diferentes condiciones para intentar detectar el entorno en el que el Krobot se encuentre de forma automática (es decir, si el consumo variase drásticamente podríamos deducir si se encuentra en suspensión, en suelo o en tierra de forma automática), otro objetivo que se quería estudiar era la comparación entre la salida de los sensores y la relación de la salida de velocidad de los encoders y el PWM.

Los datos correspondientes a la señal de los sensores de consumo se obtienen a través de los pines analógicos de Arduino, eso quiere decir, que recibiremos un valor de tensión que en Matlab será traducido como un numero entre 0 y 1023, posteriormente podemos reconvertir este valor usando la sensibilidad del sensor (185mv/A) y sabiendo que el valor central de la recta para A=0 es la media de la alimentación del sensor, es decir, 2.5V.

5.2 Desarrollo de la nueva aplicación Wifi

Ya que queremos registrar los datos de consumo de todas las ruedas lejos del ordenador tenemos que utilizar una nueva aplicación Wifi similar a la de la identificación, solo que introduciendo más datos en el envio. Se hizo la primera adaptación enviando un total de 10 datos: 4 señales de velocidad, 4 señales de consumo y los PWM correspondientes a cada lado del robot y las comunicaciones no funcionaban, los valores recibidos no tenían sentido (Figura 5.1), así que se volvió a la fase de debug para localizar el error en el envio.

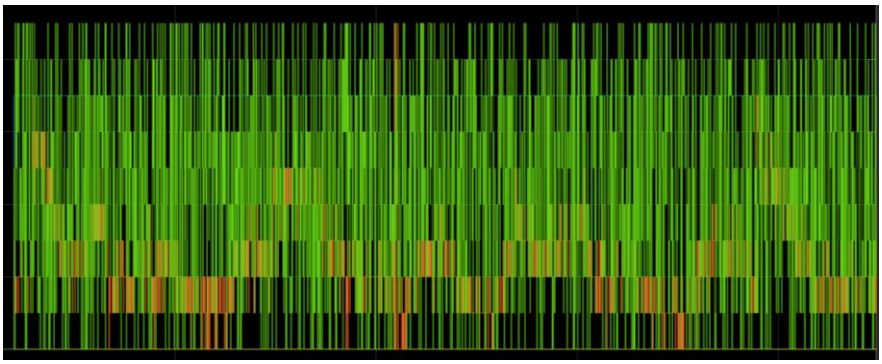

Figura 5. 1 - Primera recepción de datos de los sensores de consumo

Se hizo una aplicación exclusiva para el envio de la señal de un único sensor de consumo y tampoco funcionaba por lo que se localizó el problema en el envio de estos valores, sin embargo, al contrario que con los números negativos de los encoders no se ha llegado a un motivo para que ocurra esto y tampoco se pudo solucionar utilizando ningún tipo de offset o ganancia. Finalmente, la solución fue descomponer el valor del consumo en 4 digitos y enviar cada uno como un dato y recomponerlo en la aplicación de PC.

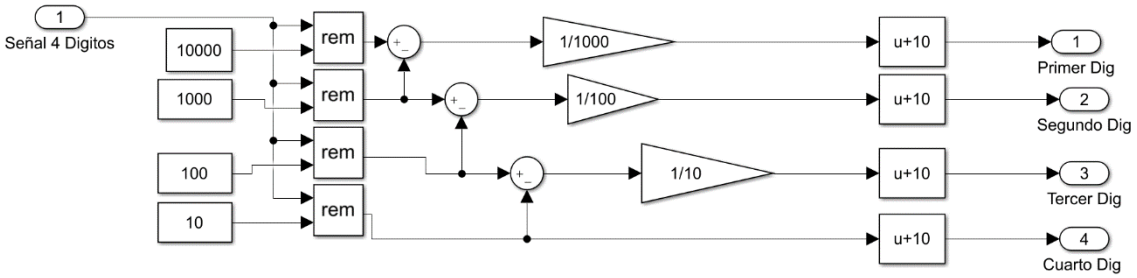

Figura 5. 2 Descomposición de señal de Consumo a 4 dígitos

5.2 Pruebas con los valores de las señales de consumo

Con la nueva aplicación lista, se estudió el comportamiento de los sensores de consumo para saber que funciones podrían desempeñar para mejorar el control del sistema. Primero se hizo un ensayo en el que se enviaba una consigna de PWM constante y se dejaba al robot avanzar libremente. Este fue el valor de consumo obtenido:

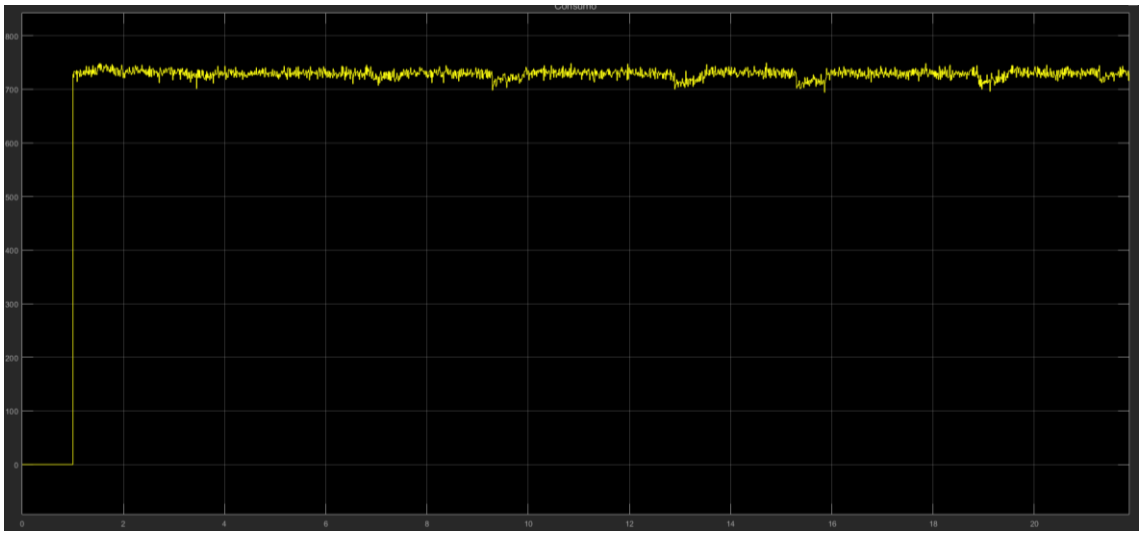

Figura 5. 3 - Valor de consumo de una rueda, movimiento libre en suelo

Ahora con una referencia, se repitió el experimento realizando paradas en seco del robot.

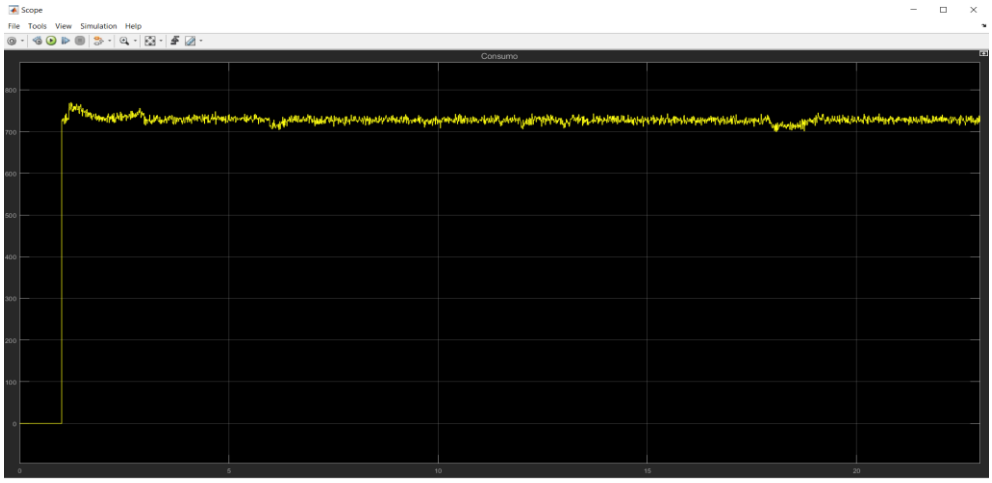

Figura 5. 4 Valor de consumo, con movimiento y parada

Si comparamos las dos gráficas se ve que en esta última señal hay pequeños desniveles que no aparecen en la otra, esto es debido a las paradas ya que al no circular corriente por el motor disminuye levemente el valor de la señal, y sobre todo es interesante destacar los picos que aparecen después de las paradas. Esto se debe a que los motores tienen un consumo bastante mayor durante su arranque que durante su funcionamiento en régimen permanente. Como primera conclusión podemos sacar que la sensibilidad del sensor es muy baja ya que apenas hay diferencia entre el movimiento y la parada.

La siguiente experiencia que se estudió fue observar el consumo en diferentes terrenos, primero se hizo una dejando al robot en el aire y luego poniéndolo en suelo durante el envio y luego se repitió al revés, es decir, arrancando en tierra y levantándolo durante el proceso. Lamentablemente no hubo diferencias destacables entre ambas gráficas, en ambas aparecen perturbaciones en la señal, pero no son síncronas a los momentos de cambio de terreno, así que

la conclusión que se llegó fue que no podemos conocer el tipo de terreno en el que estamos trabajando solo con los sensores de corriente.

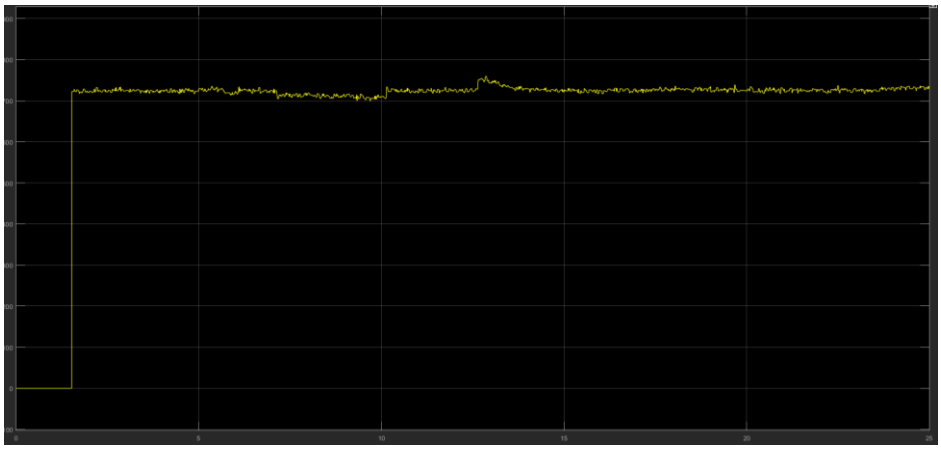

Figura 5. 5 - Consumo de arranque en aire y posterior puesta a suelo

La tercera experiencia se realizó para asegurar el correcto funcionamiento de los sensores debido a la experiencia anterior. En teoría los valores del sensor deberían ser altamente distintos si los sentidos de giro son distintos ya que esto provoca intensidades de sentidos distintos. Los resultados fueron de acuerdo con lo esperado, confirmando que funcionan correctamente y que quizás sea necesario buscar sensores de mayor sensibilidad para poder apreciar los cambios de terreno.

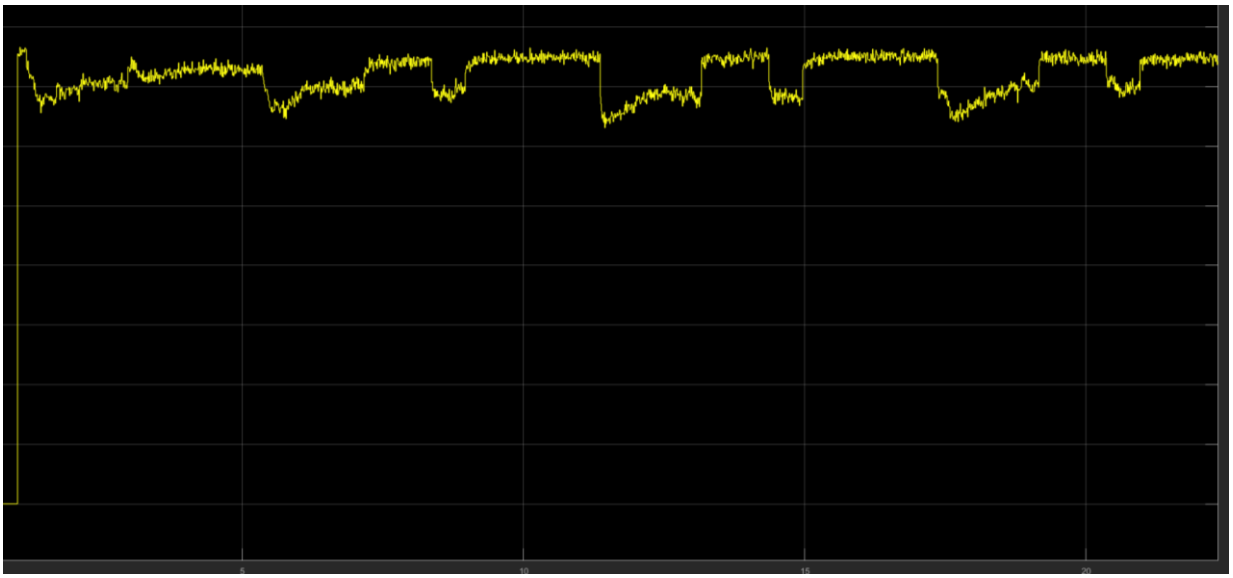

Figura 5. 6 - Consumo con cambios de sentido en la velocidad

La última experiencia, comparar el valor del consumo con la relación de velocidad/PWM no puedo completarse con éxito tampoco. La aplicación de comunicación volvió a dejar de funcionar y esta vez no se consiguió de ningún modo repararla, en el mejor caso en el que se bajó el tiempo de muestreo y se envió el consumo de una única rueda así como solo la velocidad de la misma se recibieron datos completamente ilógicos.

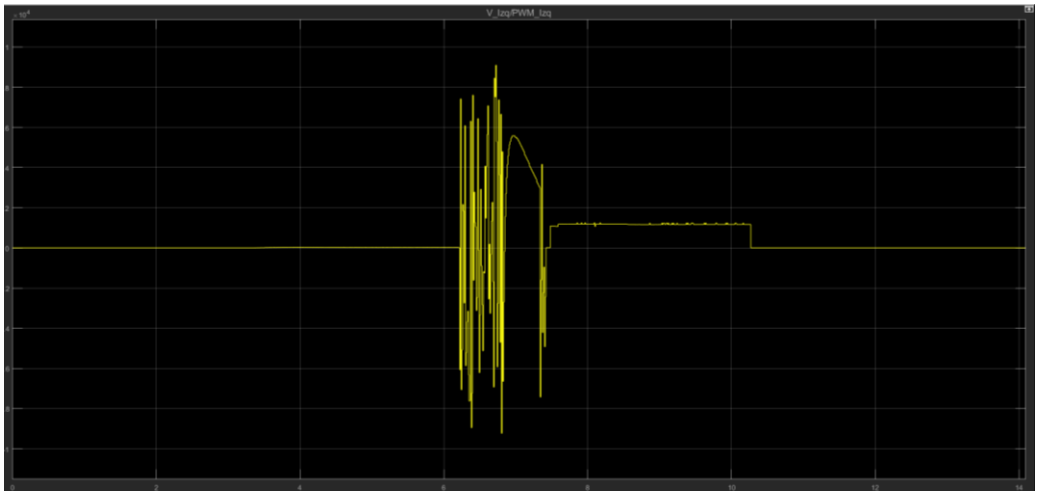

Figura 5. 7 - Error en el envio de la velocidad y el PWM

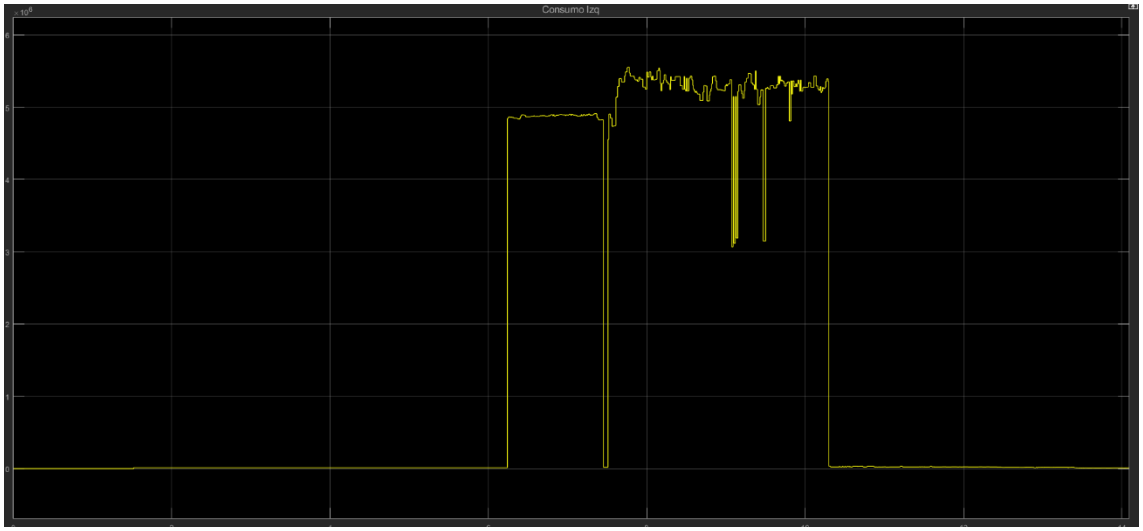

Figura 5. 8 - Error en el envio del consumo

5.3 Implementación del filtro de la IMU al control

En principio, en esta etapa se iba a utilizar el filtro Madgwick desarrollado anteriormente en el TFG Control Cinemático del Rambler de Daniel Ruiz Gómez [9] sin embargo a la hora de incorporarlo al diseño daba valores angulares que no correspondían y no se consiguió corregir, se cree que se debe al algoritmo de calibración.

Sin embargo, más tarde se utilizó el filtro desarrollado Daniel Martínez Morales, en su TFG Control Cinemático Avanzado del Robot Piero [10] que funciona perfectamente en el sistema. Este filtro utiliza todas las componentes del IMU (Las 3 señales del acelerómetro, las 3 señales del giroscopio y las 3 señales del magnetómetro) para obtener la orientación del robot en el espacio tridimensional filtrándolas complementariamente entre sí mismas.

Esencialmente aprovechamos del filtro la señal del Pitch, que nos indica la inclinación del robot respecto del eje Y, para habilitar los controladores de Cuesta arriba o Cuesta abajo. Sin embargo, el filtro tiene mucho más potencial, por ejemplo, combinando las señales de este con las de la cinemática del robot podríamos obtener una mayor fidelidad del posicionamiento del mismo o corregir su orientación en caso de perturbaciones.

Capítulo 6 – Conclusiones y Trabajos Futuros

6.1Conclusiones

De los objetivos propuestos al comienzo del trabajo se han cumplido la mayoría satisfactoriamente. Se consiguió mejorar el montaje del robot Krobot, deshaciéndolo por completo, corrigiendo errores del montaje anterior, añadiendo los nuevos sensores de consumo, rediseñando el cableado anterior para volverlo más accesible y siempre manteniendo el diseño lo más equilibrado posible.

Se han conseguido instaurar múltiples controladores de velocidad para el robot Krobot adecuados a sus posibles zonas de acción ya que se diseñó como un robot todoterreno y, además, se ha añadido un sistema de selección de estos que le permite pasar fácilmente de un terreno con facilidad.

Se han realizado aplicaciones de comunicación por Wifi que permiten la recibir datos de todos los sensores del Krobot, así como protocolos de comunicación que corrigen casi todos los errores que se han dado durante estos meses.

Se han conseguido realizar experimentos con los sensores de consumo, familiarizándonos con su funcionamiento, así como incorporar una aplicación con el sensor IMU que complementa los controladores de velocidad.

Además, a nivel personal, creo que esta experiencia me ha enseñado como abordar verdaderos proyectos de ingeniería con objetivos reales donde surgen muchos problemas y tienes que aprender a solucionarlos por ti mismo, buscando más documentación por uno mismo y dependiendo menos de los profesores.

6.2 Trabajos Futuros

Durante el transcurso de este TFG han surgido ideas para líneas de desarrollo futuras para el robot Krobot que por limitación en el tiempo no han podido ser abordadas. A continuación, se exponen estas ideas:

- En primer lugar y más importante, el desarrollo del control dinámico del robot que se planteó como un objetivo inicial. Es el único objetivo no cumplido realmente y se debe tanto a la falta de tiempo como a su complejidad, existen

muchos estudios al respecto del control dinámico de los robots de Skid Steer y son bastante complejos, sería muy interesante y útil para el desarrollo del robot llegar a obtener un modelo dinámico del mismo para su.

- Implementación de más funciones con el Filtro complementario de la IMU como corrección de la orientación de los encoders o introducir un Filtro de Kalman Extendido que permita mejorar el posicionamiento.
- Renovar el módulo Wifi por uno con un menor consumo que presente menos errores con las comunicaciones.
- Desarrollo de aplicación de comunicación con el alto nivel del Krobot para sincronización de información y mejora de la fiabilidad de la posición y la orientación.

Para apoyar estas líneas de desarrollo, todo el trabajo de fin de grado ha sido desarrollado en un proyecto de Simulink, fácilmente reutilizable, el cuál alberga todos los esquemáticos de control, las aplicaciones wifi, todos los archivos necesarios para el funcionamiento del sistema, y los valores de todos los experimentos e identificaciones realizadas durante el desarrollo del trabajo.

BIBLIOGRAFIA

[1] R. Pimentel-Castillo, “Diseño y construcción de un robot móvil de dimensiones reducidas dotado de sensores y capacidad de teleoperación con interfaz táctil,” Universidad de Málaga, 2012.

[2] C. M. Jiménez, “CONTROL DE ROBOT MÓVIL KROBOT,” Universidad de Málaga, 2017.

[3] F. Ren, X. -h. Liu, J. -s. Chen, P. Zeng, and X. -f. Jia, “Analysis of Skid Steer Loader Steering Characteristic,” *Adv. Mech. Eng.*, vol. 7, no. 1, pp. 245713–245713, Jan. 2015.

[4] “MPU-9250 and Arduino (9-Axis IMU) | Robotics, Teaching & Learning.” [Online]. Available: http://www.lucidarme.me/?p=5057. [Accessed: 08-May-2017].

[5] “Arduino Due.” [Online]. Available: https://store.arduino.cc/arduino-due. [Accessed: 20-Nov-2017].

[6] “Simulink.” [Online]. Available: https://es.mathworks.com/products/simulink.html. [Accessed: 17-Aug-2017].

[7] “MatLab.” [Online]. Available: https://es.wikipedia.org/wiki/MATLAB.

[8] Allegro MycroSystems LLC, “ACS712 DataSheet,” 2017.

[9] D. R. Gómez, “Control cinemático del Rambler,” p. 107, 2017.

[10] Daniel Martínez Morales, “Control Cinemático Avanzado del Robot Piero”,2018.