CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN MECATRÓNICA

"Diseño de un sensor virtual para estimación del área de contacto en un vehículo"

Tesis que presenta:

Gamaliel Faustino González

Para obtener el grado de:

Maestro en Ciencias

En la especialidad de:

Ingeniería Eléctrica

Directores de Tesis:

Dr. Carlos Alberto Cruz Villar

Dr. Marco Antonio Moreno Armendáriz

México, D.F. Febrero 2015

.

A mis padres...
A mi esposa
A mi hermosa familia

.

AGRADECIMIENTOS

Primeramente quisiera agradecer al creador por darme vida y fuerzas para cumplir esta meta.

A mis padres por ser la combinación perfecta para darme el ejemplo de Fe y Constancia que me han traído hasta este punto, por el apoyo moral y económico que me han brindado de forma incondicional, porque siempre han estado para mi.

♡ A mi esposa por ser mi apoyo sentimental y cuidar de casa en mi ausencia, por librarme de preocupaciones para dedicarme tiempo completo a este proyecto.

A mis hermanos por acompañarme en esta aventura de la vida, por ser una razón más para luchar por este sueño.

En el ámbito profesional agradezco a mis asesores, el Dr. Carlos Alberto Cruz Villar y el Dr. Marco Antonio Moreno Armendáriz por dar forma a esta tesis, por apoyarme y enfocarme hacia una formación científica, por todos los consejos que me llevaron a disciplinarme, porque sin ellos hubiera sido imposible culminar con éxito.

A mis sinodales por las aportaciones dadas mediante sus revisiones, por el tiempo dedicado a la lectura del trabajo de tesis y por sus opiniones expresadas durante la evaluación del trabajo realizado.

A toda la planta académica y de apoyo de la sección de Mecatrónica, por la disposición a apoyarme, por las enseñanzas y la formación académica que me brindaron. A mis compañeros de generación, especialmente a Rubén, Chucho, Edgar y Miguel por darme su apoyo cuando surgían dudas, por convertirse en mis amigos. A Carlos A. Duchanoy, por brindarme el apoyo profesional como un compañero de equipo. Al asistente de investigación, el Ingeniero Andrés González Rodríguez, por ser el apoyo perfecto cuando mi trabajo requería asuntos técnicos y de construcción, por la eficiencia mostrada cuando me permitía encomendarle alguna tarea. Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico que me ha brindado mediante su programa nacional de becas, porque gracias a ellos es posible que jóvenes como yo logren sus estudios de maestría. Al CINVESTAV por darme la oportunidad de realizar mis estudios de posgrado.

A todas las personas que hicieron esto posible...

¡ G R A C I A S !

RESUMEN

Los sensores virtuales son programas de computadora que se implementan como una alternativa a los sensores físicos. Su principal aplicación es la estimación de variables que resultan ser difíciles o imposibles de medir directamente con un sensor físico. Éste es el caso del área de contacto de las llantas de un vehículo terrestre que se encuentra en marcha, cuya medición actualmente, no puede ser obtenida directamente mediante un sensor físico. En la actualidad, cuando se trata de medir el área de contacto real entre las llantas de un vehículo y el suelo, se recurre a la implementación de pistas de prueba y bancos de prueba con una cámara que pueda ver la huella de las llantas cuando éstas pasan sobre ella. En el caso de las pistas de prueba, la medición no puede hacerse en línea ya que solamente se tiene disponibilidad para medir con la cámara cuando el vehículo se encuentra sobre ella. Con los bancos de prueba se tiene una medición constante que no puede ser considerada como medición en línea debido a que el vehículo de prueba debe estar total o parcialmente estático, esto hace que el tipo de experimentos que se pueden realizar mientras se toma la medición sean restringidos.

Por lo anteriormente mencionado, y tomando en cuenta el desarrollo tecnológico con el que se cuenta. En el presente trabajo de tesis se propone el diseño de un Sensor Virtual que es capaz de estimar en línea el área de contacto de las llantas de un vehículo. Para ello se considera la construcción de un Sensor Virtual basado en datos implementando la técnica basada en Redes Neuronales Artificiales, la cual se construye por medio de entrenamiento en el que se utiliza un banco con un gran número de datos representativos del comportamiento del sistema. Los datos incluyen la medición de una cámara que se utiliza en conjunto con la tecnología óptica de Reflexión Interna Total Frustrada para obtener el área de contacto en las llantas del vehículo, que es la variable deseada, y la medición entregada por un conjunto de sensores físicos que determinan el estado del vehículo en el momento en que se toma la imagen con la cámara, dichas mediciones con sensores físicos son tales que puedan ser denominadas como variables relacionadas al proceso deseado. El vehículo de aplicación del Sensor Virtual es un auto a escala 1/5 tipo baja 5sc.

Los resultados obtenidos por la estimación del Sensor Virtual con respecto al área de contacto real son favorables al compararlos con el rango en que se mueven las salidas deseadas. Basándonos en el resultado de error con mayor magnitud se obtiene una variación causada por error menor al 5 % del valor de variación total del área de contacto. Por lo cual se concluye en un Sensor Virtual construido de forma correcta, siendo validado y habilitando la posibilidad de ser implementado para cualquier aplicación que requiera un factor de tolerancia mayor o igual al 5 %.

ABSTRACT

Virtual sensors are computer programs that are implemented as an alternative to physical sensors. Its main application is the estimation of variables that are difficult or impossible to be directly measured with a physical sensor. This is the case of the contact area of a land vehicle with the ground, whose current measurement can not be directly measured via a physical sensor. Today, when the real contact area between the tires of a vehicle and the ground must be measured, designers tipically use the implementation of track and testing bench with a camera that can see the trace of the tires when they pass over it. In the case of test tracks, the measurement cannot be done online because it is able to get a measurement only when the vehicle is over the camera. With testing bench the measure is constant but cannot be considered as an online measure due to the vehicle must be fully or partially static. This makes the experiments to be restricted.

By the above, and taking into account the technological advances, this thesis presents the design of a virtual sensor that is able to online estimate the contact area of a vehicle's tires. To do this, we consider the construction of a virtual sensor based on the implementation of Artificial Neural Networks techniques. Such an ANN is built through training with a representative large number of data of the behavior of the system. The data includes measurements of a camera that is used in conjunction with the Frustrated Total Internal Reflection technology to obtain the contact area of the vehicle, which is the desired variable, and the measurement provided by a set of physical sensors that determine the status of the vehicle at the time the image is taken with the camera, these measurements with physical sensors are such that they can be referred as variables related with the desired process. The used vehicle for the implementation of the virtual sensor is a 1/5 scale baja type car.

The results obtained by the estimation of the Virtual Sensor with respect to the contact area are favorable when compared to the range in which the desired outputs are moving. Based on the error result with greater magnitude, a variation due to the error is about 5 % of the total variation of the contact area is obtained. Therefore it is concluded that the virtual sensor was built in a correct way, being validated and enabling the possibility to be implemented for any application requiring a tolerance factor greater or equal to 5 %.


Cinvestav

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Antecedentes y Estado del Arte.

Los sensores virtuales son programas de computadora que se implementan como una alternativa a los sensores físicos. Su principal aplicación es la estimación de variables que resultan ser difíciles o imposibles de medir directamente con un sensor físico. Éste es el caso del área de contacto de las llantas de un vehículo terrestre que se encuentra en marcha, cuya medición actualmente, no puede ser obtenida directamente mediante un sensor físico.

En la instrumentación de sistemas y procesos, dentro de la industria, habitualmente se realizan mediciones indirectas de variables deseadas mediante la obtención de otras variables relacionadas. Esto se debe a la existencia de variables físicas que pueden ser medidas con mayor facilidad para que, a partir de éstas, se pueda obtener el valor de las variables deseadas. Un sensor virtual está dedicado a la estimación de variables en un sistema o proceso por medio de la medición de variables indirectas y/o modelos matemáticos, razón por la cual ha llegado a tener una gran importancia en el campo industrial, entre sus principales aplicaciones se encuentran [1]:

- Medición en línea cuando los sensores físicos no están disponibles.
- Estimación de variables en tiempo real para monitoreo y control.
- Prevención de fallas y validación, trabajando en paralelo con sensores en *hardware*.
- Obtención de una copia de seguridad en los instrumentos de medición.
- Reducción de costos cuando la implementación de un sensor en *hardware* resulta demasiado costosa.

En cuanto a su principio de operación, los sensores virtuales se clasifican en [2]:

- Basados en modelo: Este tipo de sensores se basan en ecuaciones matemáticas que describen el proceso físico, considerando las condiciones del mismo. Un modelo puede describir claramente el comportamiento de ciertos mecanismos; sin embargo el proceso de modelado requiere conocimiento de los factores que afectan en dicho comportamiento y las leyes físicas que lo rigen, considerando los estados ideales del sistema. No siempre se conoce el comportamiento de todos los mecanismos y existen casos en los que los factores que afectan el comportamiento son demasiados, lo cual incrementa la dificultad de implementar un sensor basado en modelo.

- Basados en datos: Estos modelos también llamados modelos de caja negra se construyen a partir de mediciones del proceso para determinar su comportamiento, se pueden aplicar a partir de la observación del proceso por medio de técnicas estadísticas de regresión sin necesidad alguna de la información interna del proceso. Debido a su fácil aplicación, los sensores basados en datos han ganado popularidad entre los procesos que presentan dificultades al ser modelados, operando en entornos no controlados y/o fuera de laboratorio. Para su implementación existen diversas técnicas de Inteligencia Artificial que pueden ser utilizadas en el diseño de este tipo de sensores, tales como las Redes Neuronales Artificiales [1], [2], [3], [4].

Debido a que en la industria química no es posible la medición de variables críticas en línea y a las características de no-linealidad en sus procesos, este tipo de industrias han sido las principales empleadoras de sensores virtuales basados en datos para monitoreo y control. En [2] se menciona la dificultad que se llega a tener para reaccionar a cualquier tipo de cambios en el proceso de medición y control de PH y se propone un Sensor Virtual con compensación o calibración del modelo de forma periódica que resuelve dicho problema de la siguiente manera: Primero se establece un modelo de sensor virtual basado en Redes Neuronales tipo perceptrón multicapa, el cual es entrenado con datos históricos de la variable objetivo y los procesos relacionados. Posteriormente, para mejorar el desempeño del sensor virtual, se desarrolla un algoritmo de fusión de datos basado en la salida del modelo y mediciones de campo con el cual se logra compensar las estimaciones del sensor virtual. Podemos encontrar un enorme número de trabajos aplicados en la industria química, para ello basta revisar el primer capítulo del libro de Fortuna [1]; sin embargo la dinámica de los procesos químicos industriales resulta ser lenta comparada con un proceso mecánico del mismo tipo y sufre cambios constantes en su comportamiento.

En [5] se emplea un método simple de sensor virtual para hacer mediciones de la velocidad del ángulo de guiñada en un vehículo terrestre, basados en un filtro de Kalman y el modelo dinámico de un vehículo de dos grados de libertad. Principalmente utiliza la medición de la aceleración lateral, la velocidad del vehículo y el ángulo del vehículo para predecir la velocidad

de guiñada.

El filtro de Kalman es un conjunto de ecuaciones matemáticas que proporciona una solución recursiva eficiente de la minimización del error cuadrático medio mediante el uso de una forma de control con retroalimentación, para la estimación del ángulo de guiñada, estas ecuaciones pueden ser divididas en ecuaciones de actualización de tiempo y ecuaciones de adaptación de medida. Las ecuaciones de actualización de tiempo pueden ser pensadas como ecuaciones predictivas que proyectan la estimación del estado actual por delante en el tiempo, mientras que las ecuaciones de adaptación de medida pueden ser consideradas como ecuaciones correctoras que ajustan la estimación proyectada con una medición real en ese momento.

No siempre es posible tener mediciones reales de la variable deseada de forma frecuente, en ocasiones es poco factible o hasta imposible hacer dicha medición. Es por ello que en [3] se aprovechan las características de un tipo de Sensor Virtual que puede ser construido fuera de línea para, posteriormente, hacer cálculos en línea. El proceso de trabajo de dicho Sensor Virtual es primeramente elegir un grupo de variables auxiliares que guardan relación con la variable deseada, luego se construye un modelo matemático con las variables auxiliares como entradas y la variable deseada como salida, se simula y revisa el modelo con *software* para comprobar que ha sido correctamente construido y finalmente se introducen datos del proceso real para que el modelo estime las variables deseadas. En general, el *software* de un sensor virtual de este tipo tiene las siguientes funciones:

- Recopilar datos desde el sistema a analizar.
- Analizar y confirmar la validez de los datos recopilados.
- Construir un modelo de Sensor Virtual basado (en este caso) en una Red Neuronal.
- Calcular el valor de la variable deseada usando el modelo de la Red Neuronal.
- Validar los datos estimados del modelo de Red Neuronal con datos de salida del sistema.

La función principal del *software* fuera de línea es la adquisición y procesamiento para construir un banco de datos del sistema que generalmente se archiva en algún formato de texto. Un gran número de datos es utilizado para construir el modelo de Red Neuronal por medio de un entrenamiento.
La función principal del Sensor Virtual en línea es capturar las variables relacionadas y calcular el valor de la variable deseada en tiempo real mediante el modelo de Red Neuronal que fue construido fuera de línea.

En [4], una Red Neuronal es entrenada para aprender el comportamiento dinámico de un sistema de grúa de pórtico. La mayoría de grúas de pórtico utilizan dos controladores para estabilizar la posición del carro y el ángulo de balanceo de la carga, por lo tanto se requiere

de dos sensores para medición. Lo anterior es reemplazado por un Sensor Virtual que, con la información de la corriente del motor que mueve el carro de la grúa y el modelo dinámico del sistema, puede estimar tanto la posición de carro como el ángulo de balanceo de la carga. Así los dos sensores, que en ocasiones se colocan en el lugar de la carga, son cambiados por un sólo sensor situado junto al motor que mueve al carro.

A pesar de los avances en el campo de los Sensores Virtuales, en la actualidad resulta complicado encontrar un trabajo en el que se utilice un Sensor Virtual para estimar datos de contacto, más aún, el área de contacto en los neumáticos. En cambio, se cuenta con un gran número de técnicas con las cuales se calcula dicha área de contacto. En [6] se describe el área de contacto por medio de una fórmula geométrica simple de un elipse. Distintos autores han intentado complementar las descripciones matemáticas del área de contacto de una llanta; sin embargo en su mayoría parten de formas geométricas como círculos, cuadrados, rectángulos y elipses.

En [7] se trata la importancia de la relación entre las dimensiones de la llanta y su comportamiento bajo carga, se hace una revisión completa del catálogo de cada llanta para definir la geometría de diez llantas convencionales diferentes y por medio de expresiones matemáticas se llega al cálculo del área de contacto. Esto representa una mejora a las descripciones geométricas tomando en cuenta que se contemplan factores de catálogo adicionales a las dimensiones, tales como la carga permitida por el neumático, la presión de inflado y la deflexión de la llanta. Pero cuando se utilizan valores de catálogo para cada parámetro solamente se puede llegar a valores del área nominal, los cuales son válidos únicamente para ciertas combinaciones de carga-presión de inflado. En diversos trabajos como el de [8] se proponen factores de corrección que dependen de distintos parámetros, en este caso de la carga real del neumático, para calcular el área de contacto en llantas con cargas y presiones arbitrarias utilizando las fórmulas existentes y valores de catálogo. La determinación del área de contacto es un problema complejo y se ha abordado por distintos autores; pero sin importar el número de coeficientes y factores de corrección, es probable que los cálculos matemáticos fallen debido a que no se tengan en cuenta todos los parámetros relacionados o a que la llanta simplemente no cumpla con los estándares del catálogo, por lo tanto es necesario comprobar la validez de lo que plantea cada autor con una medición real del área de contacto.

En [9] se realiza una comparación entre el cálculo realizado con fórmulas matemáticas planteadas por distintos autores y la medición real del área de contacto de diversas llantas. El patrón tomado como medición real fue la huella plasmada con tinta indeleble sobre una placa de cartón, que fue dispuesta previamente sobre el terreno limpio de interferencias para que un vehículo pasara sobre ella y grabara así su huella. Posteriormente se determinó el área de la huella con la ayuda de un planímetro polar. Los resultados de la comparación comprueban que ningún modelo matemático contemplado en el artículo ha sido exacto en cada circunstancia. Por lo tanto, cuando se desea asegurar que el área de contacto es calculada de forma exacta,

es preciso recurrir a los distintos métodos diseñados para tomar una medición real.

En [10] se utiliza un banco de pruebas que consta de un cristal resistente y una cámara de video en la parte posterior, la cual es utilizada para tomar la imagen del área de contacto de una llanta; por medio del procesamiento de dicha imagen se puede llegar a una relación entre la cantidad de pixeles que constituyen el patrón dejado por la huella de la llanta y el área de contacto real. Resulta factible implementar un banco de pruebas con una cámara de video para la obtención del área real debido a que una cámara con resolución básica (640 x 480 pixeles) tiene hasta aproximadamente 200 veces mayor resolución que los métodos basados en tecnologías resistivas o capacitivas, además de que es más fácil e inmediato utilizar una cámara que un panel de las tecnologías mencionadas. A pesar de las grandes ventajas que tienen las cámaras por encima de otras tecnologías, todavía queda un detalle por resolver al momento de realizar el cálculo del área de contacto: cuando se realiza el procesamiento de la imagen que deja la huella de la llanta, primero se aplica una transformación a escala de grises y posteriormente una umbralización, es así como los colores con tonos más oscuros como el negro son tomados como parte del área de contacto; sin embargo, los colores más oscuros pueden deberse a factores relacionados con fenómenos distintos al contacto de la llanta.

Una solución al problema de captar colores parecidos a los generados por la llanta en su área de contacto puede ser ocupar el fenómeno de Reflexión Interna Total Frustrada como se hace en [11] donde se construye un banco de pruebas con un cristal iluminado con luz fluorescente, con el fin de medir la distribución de la presión normal y la geometría del área de contacto. Cuando se utiliza este tipo de fenómenos ópticos es posible garantizar que en el área de contacto se distinga el ancho de banda del color de la luz que es determinado, más aún, entre mayor sea la presión en el área de contacto la intensidad de la luz del color elegido será mayor.

En la actualidad, cuando se trata de medir el área de contacto real entre las llantas de un vehículo y el suelo, se recurre a la implementación de pistas de prueba y bancos de prueba con una cámara que pueda ver la huella de las llantas cuando éstas pasan sobre ella. En el caso de las pistas de prueba, la medición no puede hacerse en línea ya que solamente se tiene disponibilidad para medir con la cámara cuando el vehículo se encuentra sobre ella. Con los bancos de prueba se tiene una medición constante que no puede ser considerada como medición en línea debido a que el vehículo de prueba debe estar total o parcialmente estático, esto hace que el tipo de experimentos que se pueden realizar mientras se toma la medición sean restringidos.

Por lo anteriormente mencionado, y tomando en cuenta el desarrollo tecnológico con el que se cuenta. En el presente trabajo se propone el diseño de un Sensor Virtual que sea capaz de estimar en línea el área de contacto de las llantas de un vehículo que se encuentra en movimiento. Para ello se considera la construcción de un Sensor Virtual basado en datos

implementando una técnica de Inteligencia Artificial denominada Redes Neuronales, la cual se construye por medio de un entrenamiento en el que se utiliza un banco con un gran número de datos del sistema. Los datos incluyen la medición de una cámara que se utiliza en conjunto con la tecnología óptica de Reflexión Interna Total Frustrada para obtener el área de contacto en las llantas del vehículo, que es la variable deseada, y la medición entregada por un conjunto de sensores físicos que determinan el estado del vehículo en el momento en que se toma la imagen con la cámara, dichas mediciones con sensores físicos son tales que puedan ser denominadas como variables relacionadas al proceso deseado.

1.2. Objetivo

Realizar el diseño de un sensor virtual basado en datos para la estimación en línea del área de contacto de las llantas de un vehículo en movimiento.

1.3. Metas

Para alcanzar el objetivo planteado se proponen las siguientes metas:

- Realizar el estudio sobre el estado del arte del tema para evaluar diversos métodos y tecnologías que pueden ser ocupadas para llevar a cabo el diseño del Sensor Virtual.
- Diseñar las etapas de construcción y validación para un Sensor Virtual basado en datos.
- Diseñar y construir un banco de pruebas para obtener una muestra de datos (banco de datos) para el entrenamiento del Sensor Virtual basado en datos, así como para llevar a cabo la validación del mismo.
- Proponer un algoritmo basado en técnicas de Inteligencia Artificial para implementar un Sensor Virtual basado en datos.
- Realizar la experimentación y validación del Sensor Virtual.

1.4. Planteamiento del Problema

Para llevar a cabo el diseño de un Sensor Virtual para estimar el área de contacto en las llantas de un vehículo, se propone en este trabajo de tesis utilizar un auto a escala 1/5 de tipo Baja 5sc SS marca hpi racing.

El Sensor Virtual basado en datos tiene como salida deseada el área de contacto de cada una de las llantas del vehículo. Las entradas deben ser elegidas tales que guarden una relación directa con el valor de la la salida deseada, para esto es necesario el estudio del modelo matemático de este tipo de vehículos. Por medio de un banco de datos con conjuntos de pares de entradas relacionadas y salidas deseadas se lleva a cabo la adecuación del Sensor Virtual para minimizar el error a la salida hasta obtener un modelo que estime de forma correcta la salida deseada para cada conjunto de entradas relacionadas.

El sensor físico que se toma como única medición del área de contacto real para la construcción del banco de datos de salida deseada es una cámara infrarroja montada en un banco de pruebas con la tecnología óptica de Reflexión Interna Total Frustrada, con lo cual se garantiza una medición precisa del área real.

1.5. Metodología

Para resolver el problema planteado y dadas las metas propuestas que se llevaron a cabo, la metodología que se siguió en el presente trabajo de tesis es la siguiente:

El estudio sobre diversos métodos y tecnologías que pudieron ser ocupadas en el diseño del Sensor Virtual ha sido de tal forma que orientó el trabajo de tesis hacia una técnica completa e innovadora para cumplir con el reto tecnológico que conlleva la construcción de un Sensor Virtual para dinámicas en las que anteriormente no se había incursionado, para llegar a esto fue necesaria la sinergia de distintas metodologías con bases científicas en campos de acción diferentes. El estado del Arte es el ya plasmado en la primera sección de este capítulo.

La metodología general para el diseño de las etapas del Sensor Virtual basado en datos fue la planteada en el libro de Fortuna [1], que consta de cuatro etapas. La primera etapa es la selección de datos medibles que se relacionan con la variable deseada; esta selección debe apoyarse en los modelos matemáticos existentes del sistema en cuestión, en caso de no existir un modelo matemático confiable en el cual basarse, la selección puede incluir datos medibles de experimentos prácticos e incluso de la observación del comportamiento del sistema. En este caso se tomaron en cuenta los modelos basados en un cuarto de auto [12] y de auto completo

[13]. La segunda etapa es determinar los sensores que serán empleados para medir las variables relacionadas; esto es, buscar y elegir entre las variables relacionadas a aquellas que resultan tener una relación con el fenómeno a medir y que son factibles de capturar por medio de instrumentación con sensores. La tercera etapa es el planteamiento de la técnica a utilizar para vincular las señales medidas con la variable deseada; en esta etapa se utilizan técnicas basadas en modelos matemáticos de entrada-salida en los cuales se minimiza el error a la salida hasta llegar a un modelo que describa correctamente el comportamiento entrada-salida deseado. La etapa final en el proceso de diseño del Sensor Virtual es la validación del modelo obtenido en la tercera etapa; esta etapa consta de realizar estimaciones de la variable deseada por medio del modelo que ha sido construido y comparar los resultados obtenidos con datos de mediciones reales de la misma variable, es así como se puede llegar a la validación del modelo.

El banco de pruebas que fue implementado para la recopilación del banco de datos de entrenamiento así como para la validación del Sensor Virtual es un diseño combinado entre los bancos de pruebas utilizados en [10] y [11]. Se ocupó una cámara infrarroja conjuntamente a la tecnología de Reflexión Interna Total Frustrada para tomar una imagen de la huella de las llantas del vehículo, posteriormente se aplicó procesamiento a la imagen y por medio de regla de tres simple se pudo obtener el valor del área de cada llanta y así tener el dato de salida deseada. Los datos relacionados a la salida deseada fueron tomados por medio de distintos sensores montados sobre el vehículo de forma estratégica. La recopilación de datos corrió a cargo de microcontroladores capaces de leer los valores obtenidos por cada sensor y comunicarlos a una computadora personal (*laptop*) en la cual fue creado un archivo para almacenamiento del banco de datos.

La propuesta de un algoritmo basada en técnicas de Inteligencia Artificial para la implementación del Sensor Virtual ha sido por medio de un modelo con Redes Neuronales, dicho modelo ocupa el archivo donde se encuentra el banco de datos de variables relacionadas y salida deseada para su proceso de aprendizaje. Los pasos comunes en el proceso de aprendizaje supervisado de una Red Neuronal Artificial son:

- Recolección de datos de entrada.
- Determinación de la estructura de la Red Neuronal.
- Entrenamiento utilizando el patrón de entrada y de salida deseada.
- Validación de la RNA entrenada.

Los experimentos considerados son secuencias combinadas de auto estático, aceleración, frenado y cambio en la dirección. Por medio del banco de pruebas se realizó un gran número de experimentos para obtener el conjunto de lecturas de los sensores con lo que se obtiene el

banco de datos y posteriormente se construye el Sensor Virtual. Una vez superada la etapa de construcción, es necesario validar el correcto funcionamiento del Sensor Virtual para utilizarlo en línea garantizando la efectividad de los valores estimados, esto se realiza estimando valores de la variable deseada con el Sensor Virtual de forma paralela a la medición real con lecturas del sensor físico, los resultados de ambos sensores son comparados y si se obtienen lecturas aproximadamente iguales se logra tener la validación del Sensor Virtual.

1.6. Contribución de la Tesis

Existen varios autores que han tratado de describir el comportamiento del área de contacto de las llantas en un vehículo, muchos de ellos por medio de fórmulas geométricas para cuadrados, rectángulos, círculos y elipses. Este tipo de aproximaciones resultan no ser exactas además de que no sufren cambios con la dinámica del vehículo, es decir, según estos modelos el área de contacto permanece constante cuando un vehículo está en marcha. Otros autores toman estas fórmulas geométricas y les adicionan valores del catálogo del proveedor con lo cual se obtiene un comportamiento más exacto, aunque aún constante. Otro enfoque interesante es tomar la variación de la altura de la llanta [13] para formular el comportamiento dinámico del área de contacto cuando un vehículo se encuentra en marcha a partir de la geometría de la llanta, usando trigonometría; sin embargo para los experimentos a realizar se pretende entrenar el sistema con mediciones reales.

Para hacer la toma de mediciones reales se han desarrollado distintas técnicas visuales, capacitivas, resistivas, inductivas, de contacto, etc. El problema con la mayoría de tecnologías radica en la resolución que puedan entregar, para efectos del área de contacto resulta muy importante la variación que se tiene debido a la resolución. Es por ello que en el presente trabajo de Tesis se plantea el diseño de un Sensor Virtual para estimar el área de contacto de las llantas de un vehículo que se encuentra en movimiento, se implementan las técnicas visuales con cámara debido a la resolución que pueden entregar.

Es conocido el reto tecnológico que se tiene al intentar llevar un Sensor Virtual a dinámicas en las que antes no se había incursionado; sin embargo, dicho reto tecnológico es también parte de la motivación que mueve a este trabajo. Tener la medición del área de contacto de las llantas cuando un vehículo se encuentra en marcha sin necesidad de un modelo matemático complejo ni sensores físicos que solamente puedan dar una medición exacta cada que el auto pase sobre ellos.

Para probar el desempeño del Sensor Virtual se construye un banco de pruebas en el cual se toma la medición del área de contacto real por medio de de un sensor físico, la cual se puede comparar con la estimación obtenida por el Sensor Virtual.

1.7. Organización del Documento.

La organización del trabajo de tesis está dada de la siguiente manera:

Capítulo 1 Presenta la introducción al trabajo de tesis, contiene los objetivos y metas a cumplir. De igual forma se incluye la metodología a seguir para llegar al cumplimiento de los objetivos y metas.

Capítulo 2 Contiene el diseño general de un sensor virtual, comenzando por la presentación del vehículo a escala elegido para la implementación del resultado a obtener. En este capítulo se desarrolla y resuelve de forma teórica cada etapa en el proceso de diseño del sensor virtual.

Capítulo 3 En este capítulo se desarrolla la instrumentación total de la plataforma experimental, que está conformada por el banco de pruebas y el vehículo a escala. Se muestra la puesta en marcha y solución de problemas para la obtención de mediciones de calidad con todos los sensores. Se diseñan las conexiones electrónicas, el montaje de sensores y la programación para integrar y sincronizar el sistema total.

Capítulo 4 Contiene la arquitectura de la Red Neuronal Artificial que será empleada como modelo matemático del sensor virtual, cada elemento dentro de la arquitectura es asignado o definido en éste capítulo. También se muestra el modelo matemático que representa la arquitectura elegida y la forma en que la RNA será entrenada para que entregue a la salida los resultados deseados. En la parte final de este capítulo se presenta la RNA entrenada, es decir, los elementos resultantes del entrenamiento de la red.

Capítulo 5 En este capítulo se presentan los detalles finales requeridos al momento de exportar la RNA para su implementación en un microcontrolador, se incluye el proceso de normalización de datos de entrada y salida, junto con la definición de las funciones de activación dentro de la plataforma del microcontrolador. Finalmente se presenta la validación de las estimaciones del sensor virtual implementado utilizando gráficas comparativas y el cálculo del error de estimación.

Capítulo 6 En este capítulo se presentan las conclusiones del trabajo de tesis y el trabajo futuro propuesto.

Capítulo 2

Diseño general del sensor virtual

El diseño es el proceso de visualización mental previo a la etapa de construcción en la búsqueda de soluciones a algún problema o conjunto de problemas planteados por el hombre en su continuo proceso de adaptación según sus necesidades. Diseñar es un proceso complejo que conlleva la integración y el cumplimiento de requisitos de distintas naturalezas, desde la necesidad más básica hasta un lujo innecesario. La etapa de diseño es la más importante dentro de la construcción de un nuevo proyecto, es en esta etapa en donde se plantean las necesidades y las metas que deben cumplirse, además de la metodología a seguir para llegar a la realización del proyecto. Diseñar en ingeniería es el proceso de aplicar ciencia y tecnología para llevar a cabo un plan que resulte en la satisfacción de una demanda o necesidad específica. A comparación con los problemas matemáticos o puramente científicos, los problemas de diseño no tienen una sola respuesta correcta ya que una respuesta adecuada en el presente puede ser una solución inadecuada en el futuro.

El procedimiento general en el diseño es un proceso iterativo en el que se pasa por varias etapas, se evalúan las propuestas y los resultados y se repite sucesivamente el proceso o parte de éste. Se puede pensar que el proceso de diseño de cualquier sistema involucra ciertas etapas específicas; comienza con la identificación de la necesidad a resolver, posteriormente se pasa a formular el problema y sus requerimientos, se desarrollan conceptos con los cuales se puede solucionar el problema, se elige cuál de los conceptos resulta ser la solución con mayor factibilidad y finalmente se implementa y evalúa la solución.

En mecatrónica, diseñar es tomar las necesidades en cada aspecto de un sistema y llegar a una solución conjunta sin dejar de lado las partes mecánica, electrónica, de programación y de control. Para ello se cuenta con distintas herramientas de *software* para plasmar los bocetos o diseños por separado, prevenir problemas alternos que no se tomaban en cuenta e incluso llevarlos a una simulación conjunta para visualizar el comportamiento que se puede llegar a tener.

En este capítulo se presenta el vehículo a escala con el que se pretende trabajar debido a que es el objeto de aplicación sobre el cual se desarrolla el Sensor Virtual, dicho vehículo forma parte de las restricciones y requerimientos de la misma etapa de diseño. De igual forma se presenta el diseño de un Sensor Virtual orientado a estimar el área de contacto de las llantas del vehículo presentado, para ello se lleva a cabo cada etapa en el proceso de diseño de Sensores Virtuales y adicionalmente se plasma la solución propuesta con el diseño de cada parte que conlleva dicha solución.

2.1. Vehículo a escala

Figura 2.1.1: Auto tipo Baja 5sc SS, modelo a escala (imagen tomada del manual del vehículo).

El vehículo que se ha elegido para la aplicación del Sensor Virtual es un auto a escala 1/5 de tipo baja impulsado por medio de un motor de combustión interna de 26 centímetros cúbicos y controlado por medio de radio frecuencia con un control remoto, en la figura 2.1.1 se puede observar la imagen del auto a utilizar en los experimentos de esta tesis, en la parte inferior de la imagen el auto tiene su cubierta estética y en la parte superior se puede ver sin ella. Los autos de este tipo se caracterizan por tener una suspensión levantada ya que comúnmente se ocupan para carreras de tipo todo terreno. Las características técnicas del vehículo a utilizar se encuentran en la tabla 2.1.1.

Marca:	hpi racing.
Modelo:	Baja 5sc SS # 105735.
Escala:	1/5.
Motor:	Fulie 26s, motor de dos tiempos, 26cc, 2.9 HP.
Clutch:	8,000 RPM.
Velocidad máxima:	40MPH, 64.4 Km/h, 17.88 m/s.
Neumáticos:	HB Rodeoo.
Ruedas:	TR-10 Llanta uso pesado, rin 15 grados, diámetros pequeños.
Tanque:	700 cc, dura hasta 45 minutos de uso.
Longitud:	900 mm.
Ancho F/R:	Delantero 440 mm / Trasero 460 mm.
Altura:	320 mm.
Distancia entre ejes:	570 mm.
Track F/R:	Delantero 370 mm / Trasero 380 mm.
Peso aproximado:	12.6 Kg, 27.9 lb, peso aproximado sin combustible.

Tabla 2.1.1: Características técnicas del auto a escala.

Para efectos de desarrollo del presente trabajo de tesis se cuenta con un Diseño Asistido por Computadora (CAD) del vehículo completo desarrollado en [12], el cual se puede observar en la figura 2.1.2.

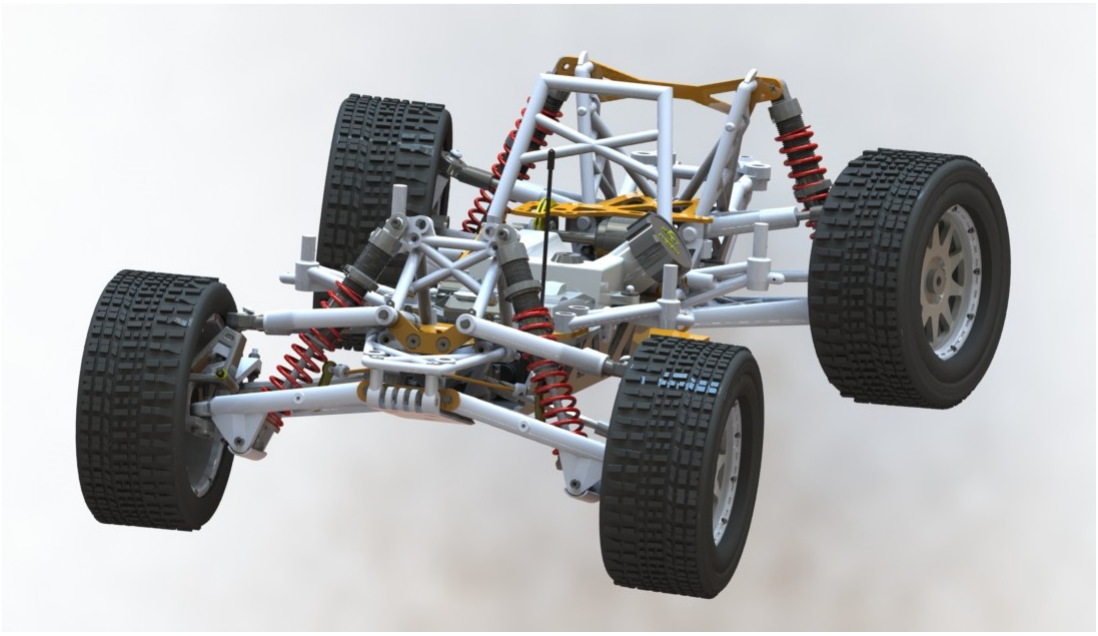

Figura 2.1.2: CAD del auto tipo Baja 5sc SS.

2.2. Diseño de un Sensor Virtual

El diseño del Sensor Virtual está basado en la metodología de diseño planteada en el libro de Fortuna [1] que consta de cuatro etapas:

1. Seleccionar los datos medibles relacionados con la variable deseada.
2. Determinar los sensores que serán empleados para medir las variables relacionadas.
3. Planteamiento de la técnica a utilizar para vincular las variables relacionadas con la variable deseada.
4. Validación del modelo obtenido.

2.2.1. Selección de datos medibles relacionados con la variable deseada

La selección de datos debe ser sustentada por los modelos matemáticos existentes del sistema a medir, en caso de no existir ningún modelo matemático, es posible incluir experimentos u observaciones del sistema para llegar a la correcta selección de las variables relacionadas. En el presente trabajo se sustenta la selección de datos en los modelos matemáticos de un cuarto de auto y auto completo desarrollados en [12] y [13], el modelo de auto completo es

desarrollado para un auto tipo baja de tamaño normal, mientras que el modelo de un cuarto de auto se basa en el vehículo a escala que se utiliza en esta tesis. Para efectos ilustrativos, a continuación se muestra parte del desarrollo general del modelo matemático de auto completo, centrando especial atención en la parte del modelo matemático del área de contacto de las llantas. Para mayor información sobre el desarrollo total de ambos modelos se recomienda acudir a la bibliografía citada, ya que el objetivo principal del presente trabajo de tesis es construir un Sensor Virtual sin la necesidad del conocimiento total del modelo del sistema, solamente se mencionará la parte de mayor interés y basta con comentar que existen distintos factores extras que influyen o se correlacionan con el fenómeno a medir.

Modelo matemático

Figura 2.2.1: Auto tipo SAE baja (imagen de CADENAS PART SOLUTIONS).

El modelo matemático sobre el cual se basa la selección de variables relacionadas con el área de contacto se refiere a un vehículo tipo SAE baja como el que se muestra en la figura 2.2.1, construido por estudiantes del Instituto Politécnico Nacional para fines de competencia. El modelo total se conforma del modelo de las llantas, modelo de la suspensión tomando en

consideración su geometría, modelo del sistema de frenado, modelo del sistema de propulsión, modelo del sistema de dirección y modelo del chasis, el cual interactúa con todos los demás sistemas. En la figura 2.2.2 se puede observar un diagrama a bloques de las partes que conforman el modelo matemático de auto completo y sus interacciones.

Figura 2.2.2: Diagrama a bloques del modelo de auto completo.

Como se ha mencionado anteriormente, cada parte del modelo matemático de auto completo influye directa o indirectamente sobre el comportamiento de las otras partes, esta comunicación puede darse por medio del chasis que es el elemento central al cual están ligados todos los elementos. A continuación se presenta una breve explicación de lo que incluye cada parte del modelo de auto completo haciendo especial énfasis en la parte del modelado de las llantas que corresponde al desarrollo del modelo del área de contacto. Para el resto de las partes del modelo basta con mencionar que influyen en el fenómeno que se estudia y que en la parte final del presente apartado se muestra una lista de algunas de las variables que influyen en el área de contacto de las llantas; sin embargo el estudio detallado del modelo matemático de auto completo no es parte de los intereses de esta tesis.

Modelado de las llantas

El modelado de las llantas es la parte de mayor interés en el presente trabajo, las llantas son el elemento por medio del cual interactúa el auto con el terreno sobre el que se encuentra. El comportamiento de las llantas es parecido al de un resorte que transmite la excitación a la entrada dada por la forma en el terreno hacia el cuerpo del auto llamado chasis, pasando antes por el sistema de suspensión. Para construir el modelo matemático del área de contacto de las llantas en función del comportamiento del vehículo se toma como punto de partida la variación de la altura Δz_t de la llanta con la cual se puede calcular la variación del largo y ancho del área de contacto usando trigonometría. Cabe mencionar que en este modelo se desprecian los efectos debidos a la deformación de la llanta ya que resulta de gran dificultad incluirlos, en vez de ello se considera que la llanta únicamente se comprime y no se deforma, representando las deformaciones como un corte en la llanta el cual genera un área de contacto de forma rectangular con largo x y ancho y. En la figura 2.2.3 puede observarse un diagrama con los cortes propuestos para formar el rectángulo que representa el área de contacto.

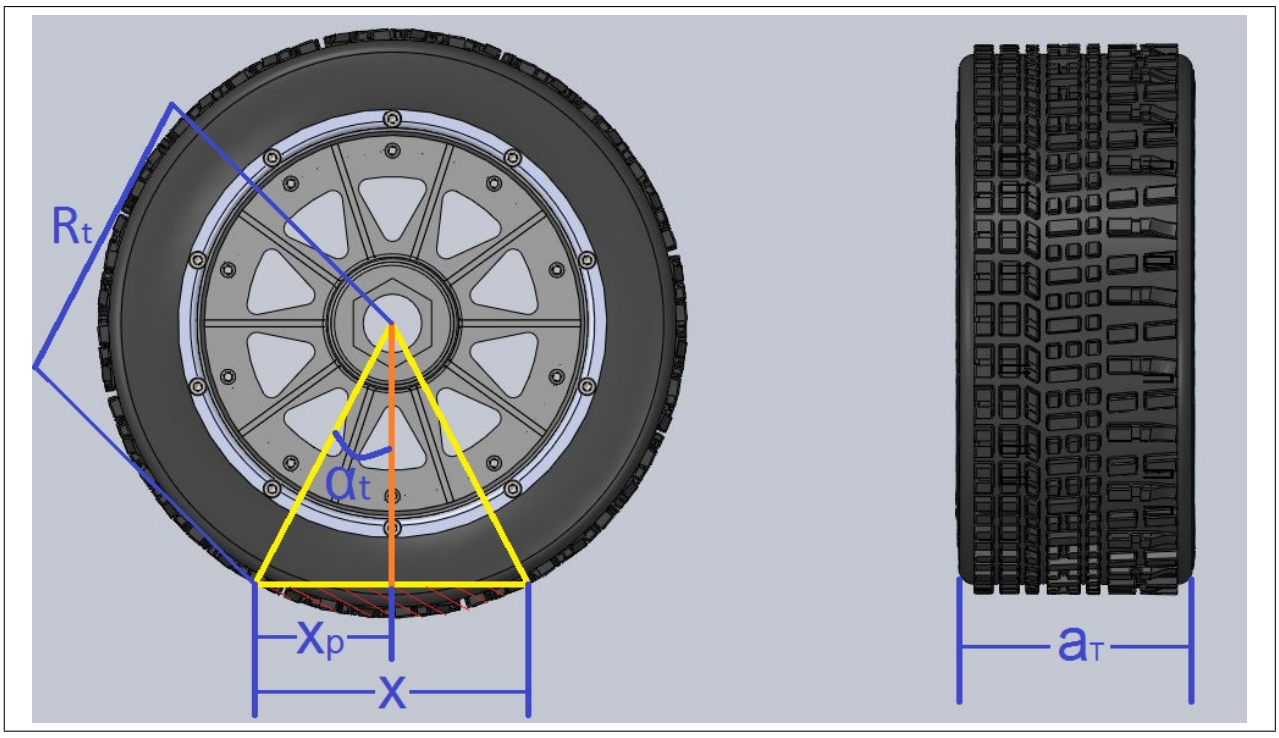

Figura 2.2.3: Diagrama del corte de la llanta.

Para obtener el factor x del área de contacto se dispone de las siguientes ecuaciones:

$$\alpha_t = arccos\left(\frac{R_t - \Delta J}{R_t}\right) \tag{2.2.1}$$

Donde:
α_t = Ángulo de corte de la llanta.
R_t = Radio de la llanta.
ΔJ = Distancia comprimida de la llanta.

$$\Delta J = Q - \Delta z \tag{2.2.2}$$

Donde:
Q = Variación del terreno.
Δz = Variación de la altura de la llanta.

$$x_p = R_t sin\left(\alpha_t\right) \tag{2.2.3}$$

Donde:
x_p = Mitad del largo del área de contacto.

$$x = 2x_p \tag{2.2.4}$$

Donde:
x = Largo del área de contacto.

Aplicando álgebra a las ecuaciones 2.2.1, 2.2.2, 2.2.3 y 2.2.4 se puede llegar a la ecuación 2.2.5.

$$x = 2R_t sin\left[arccos\left(\frac{R_t - \Delta J}{R_t}\right)\right] \tag{2.2.5}$$

Para obtener el factor y del área de contacto basta con hacer una simple sustitución.

$$y = a_t \tag{2.2.6}$$

Donde:
a_t = Ancho de la llanta.
y = Ancho del área de contacto.

Con lo cual llegamos a la expresión final del área de contacto de la llanta, la cual está dada por la ecuación 2.2.7

$$A_c = x \cdot y \tag{2.2.7}$$

Donde:
A_c = Área de contacto de la llanta.

El modelo de las llantas también incluye el cálculo del volumen de la llanta, cálculo del área transversal, cálculo del perímetro al centro de masa, cálculo del volumen perdido por la llanta, y cálculo de la fuerza ejercida por la llanta; de los cuales no se presentará el desarrollo en este trabajo.

Modelado de la suspensión

La suspensión tanto delantera como trasera son sistemas de doble horquilla, también llamado sistema de doble brazo o doble A debido a la forma de los brazos. El sistema de suspensión es un conjunto de elementos dedicados a absorber las perturbaciones a la entrada (irregularidades en el camino), comúnmente se compone de elementos mecánicos como barras, resortes y amortiguadores. Este sistema se encuentra entre la masa suspendida (chasis) y la masa no suspendida (llantas) con la finalidad de brindar confort a los tripulantes y control del vehículo.

El modelo de la suspensión se conforma por el cálculo de los ángulos de la suspensión delantera, análisis de fuerzas en la masa no suspendida por las partes delantera y trasera, análisis de fuerzas en el brazo inferior, cálculo de los ángulos de la suspensión trasera, cálculo para el brazo a torsión de la llanta. Es preciso indicar que el modelado de la suspensión en el que se basa este trabajo de tesis contempla el cambio en la geometría que sufre la suspensión con respecto a la entrada dada por el terreno, dicha observación es de suma utilidad al momento de elegir las variables y los sensores del Sensor Virtual.

Modelado del sistema de frenado

El sistema de frenado está dedicado a detener completamente o disminuir la velocidad del vehículo a voluntad del conductor por medio de la transformación de energía cinética del vehículo en calor o trabajo. La base de funcionamiento del sistema de frenos es la trasmisión de la presión, empleada por parte del conductor en el pedal, a través de un líquido o fluido que amplía la fuerza ejercida para conseguir detener el auto con el mínimo de esfuerzo posible.

El sistema de frenado que es modelado en el documento de referencia de este trabajo es un sistema hidráulico accionado por medio de un pedal, cuenta con 4 discos ventilados, uno en cada llanta. El modelo del sistema de frenado se compone por el cálculo de la fuerza del pedal, cálculo de la fuerza hidráulica y del cálculo de la fuerza del disco de frenos.

Modelado del sistema de propulsión

Al sistema encargado de mover al auto se le llama sistema de propulsión o tren de potencia, éste está típicamente conformado de varios componentes: motor, sistemas de transmisión, baterías y tanques de combustible. En el caso del vehículo modelado en [13] el sistema de propulsión consta del motor del vehículo, la transmisión fija y una transmisión variable CVT.

El modelado del sistema de propulsión se compone del modelado del motor, modelado de la transmisión variable continua y la transmisión fija, además de la aplicación de las relaciones de las transmisiones a las revoluciones y al torque.

Modelado del sistema de dirección

El conjunto de mecanismos que componen al sistema de dirección tienen la tarea de orientar las llantas delanteras del vehículo a fin de que éste tome la trayectoria deseada por el conductor. Para que el conductor no tenga que emplear un gran esfuerzo al intentar cambiar la orientación de las ruedas se dispone de un mecanismo desmultiplicador tal como una cremallera. El sistema de dirección del modelo en el cual se basa esta tesis es un sistema mecánico de piñón-cremallera con un tipo de geometría de Ackermann.

El modelado del sistema de dirección está conformado por el modelo de la cremallera, modelado de la geometría de la dirección, cálculo del radio de giro y el cálculo de la aceleración que se produce al tomar una curva.

Modelado del chasis

El chasis es una estructura generalmente de alguna aleación metálica que sostiene y aporta rigidez a un auto. El chasis de un vehículo consta de un armazón en el cual se sujeta e integra los componentes mecánicos del sistema de suspensión y los sistemas de propulsión, dirección y frenado incluyendo también la carrocería (la cual no debe confundirse con el chasis).

El modelado del chasis está compuesto por el cálculo de los momentos producidos por la suspensión, cálculo de los momentos producidos por el acelerador y el freno, cálculo de los momentos producidos por la dirección, el peso del vehículo, las componentes que genera cada uno de los efectos anteriores y cómo influyen en la altura del chasis y sus ángulos de cabeceo y alabeo.

Identificación de las variables relacionadas con la variable deseada

Diferencial de presión en las cámaras del amortiguador.
Desplazamiento en la cámara del amortiguador.
Ángulo de la horquilla superior.
Ángulo de la horquilla inferior.
Ángulo del amortiguador.
Cambios en la geometría de la suspensión.
Fuerza en el amortiguador.
Fuerza de las horquillas en el eje z.
Fuerza en las llantas en el eje z.
Peso del vehículo (incluyendo combustible).
Fuerzas producidas por la aceleración centrípeta.
Distancia al suelo de la masa no suspendida.
Distancia al suelo del punto fijo del amortiguador.
Distancia al suelo de la masa no suspendida.
Compresión / elongación del resorte.
Aceleración de la masa no suspendida en el eje z.
Aceleración de la masa suspendida en el eje z.
Ángulo de cabeceo del chasis.
Ángulo de alabeo del chasis.
Velocidad y aceleración del ángulo de cabeceo.
Velocidad y aceleración del ángulo de alabeo.
Aceleración en el eje x.
Aceleración en el eje y.
Ángulo de la dirección (volante).

Tabla 2.2.1: Identificación de variables relacionadas con el área de contacto.

Después de realizar el estudio basado en los modelos matemáticos de auto completo y de un cuarto de auto anteriormente citados, se ha llegado a la identificación de algunas variables que guardan relación con el área de contacto de las llantas. Dichas variables son

listadas en la tabla 2.2.1, cabe aclarar que es una lista resumida debido a que se generaliza el comportamiento de las llantas mencionando una sola ocasión lo que influye en las cuatro llantas, finalmente no pretende ser una lista única ya que es posible la existencia de variables que no son contempladas en este trabajo y que guardan relación con la variable deseada.

Selección de variables relacionadas

Una vez identificadas las variables que se relacionan con el área de contacto se procede a evaluar cuáles son aquellas que tienen un mejor cociente entre la relación que guarda con la variable deseada y la dificultad que presenta a ser medida con un sensor comercial. La selección final de variables junto con los rangos en los que se espera que se encuentren se muestra en la tabla 2.2.2, esta selección es encabezada con la variable deseada y posteriormente se listan las variables relacionadas elegidas.

Variable	**Rango**
Área de contacto.	$0 - 8cm^2$
Cambios en la geometría de la suspensión.	$0 - 5cm$
Aceleración de la masa no suspendida en el eje z.	$\pm 14m/s^2$
Ángulo de cabeceo del chasis.	$\pm 5.5°$
Ángulo de alabeo del chasis.	$\pm 8.5°$
Ángulo de la dirección (volante).	$\pm 60°$

Tabla 2.2.2: Selección de variables para el Sensor Virtual.

Área de contacto

Como se ha mencionado, el conjunto de variables deseadas o variable de salida del Sensor Virtual es el área de contacto de las cuatro llantas del vehículo. El rango esperado es calculado de la misma forma que se hace en [13].

Cambios en la geometría de la suspensión

En el caso particular del tipo de autos como el que se emplea en esta tesis, el cambio en la geometría de la suspensión tiene una relación directa con el cambio en el área de contacto de las llantas. Se elige esta variable ya que engloba el comportamiento de distintas variables

como los ángulos en las horquillas, la altura del chasis y el desplazamiento del sistema de suspensión. El rango esperado es medido directamente de los amortiguadores que son los elementos de la suspensión que sufren los cambios más significativos.

Aceleración de la masa no suspendida en el eje z

Esta variable se refiere a la aceleración que sufren las llantas y toda la masa que se encuentra antes del sistema de suspensión debido a un cambio repentino en la distancia entre el centro de masas de la masa no suspendida y el suelo. El rango en que se espera que se mueva esta variable se toma indirectamente del cálculo de la aceleración en z del chasis, tomando en cuenta que la aceleración en la masa no suspendida llega a ser menor que la de la masa suspendida.

Ángulo de cabeceo del chasis

Es uno de los ángulos de navegación con el cual se mide la inclinación con respecto al eje y del auto, esta variable es elegida debido a que sus cambios informan un cambio en la distribución del peso del vehículo entre las partes frontal y trasera con lo cual se obtiene también un cambio las áreas de contacto de las llantas traseras y delanteras. El rango esperado para el ángulo de cabeceo es estimado por la observación y medición en el vehículo.

Ángulo de alabeo del chasis

Es el ángulo de navegación que representa la inclinación del auto con respecto al eje x, se elige esta variable ya que un cambio en este ángulo representa un cambio en la distribución de pesos entre las partes izquierda y derecha del auto, lo cual resulta en un cambio en el área de contacto en las llantas de cada lado. El rango esperado para esta variable es estimado por la observación y medición directa en el vehículo.

Ángulo de la dirección(Volante)

A diferencia de los autos comunes, en los autos tipo baja se puede encontrar con que los cambios en la orientación de las llantas delanteras (sistema de dirección) influyen de forma radical en la forma en que las llantas entran en contacto con el suelo, por simple inspección visual se puede observar dicho comportamiento. Por ello se elige el ángulo de la dirección como variable de entrada del Sensor Virtual, con esto se puede obtener principalmente el cambio

en el área de contacto de las llantas delanteras debido al cambio en el sistema de dirección, sin descartar un posible cambio en las llantas traseras causado por el mismo movimiento. El rango esperado es medido directamente en las llantas delanteras del vehículo.

2.2.2. Elección de los sensores físicos que serán empleados para medir las variables relacionadas

Gracias a que en la etapa de elección de variables se ha considerado la capacidad de medir dichas variables con sensores comerciales, la elección de sensores resulta casi inmediata y se limita solamente a elegir entre tipos de sensores sin la necesidad de desarrollar un nuevo sensor. Para llevar a cabo la elección de sensores se toman en cuenta distintas características: Resolución, dimensiones físicas, rango de medición, precio, disponibilidad en México, tiempo de adquisición y tiempo de implementación.

Elección del sensor para medir el área de contacto real

Dentro de la gama de tecnologías contempladas para la elección del sensor con el cual se tomará la medición de lo que se considera el área de contacto real se encuentran las tecnologías resistivas, capacitivas y ópticas. En la tabla 2.2.3 se muestra una comparativa de las tres tecnologías y los parámetros que se toman en cuenta para elegir con cuál se trabajará para capturar la variable deseada, dicha tabla refleja los valores estimados para la medición de solamente una llanta por lo que en todos los casos se requiere de un aumento en los precios para la medición de las cuatro llantas.

Tecnología / **Característica**	**Resistiva**	**Capacitiva**	**Óptica**
Precio aproximado	$ 750.00 USD	$ 260.00 USD	$ 1900.00 MXP
Disponibilidad en México	SI	NO	SI
Tiempo de adquisición	6-8 Semanas	$>$ 8 Semanas	1 Semana
Tiempo de implementación	4-6 Semanas	3-6 Semanas	1-3 Semanas

Tabla 2.2.3: Comparación de tecnologías Resistiva, Capacitiva y Óptica.

El ejemplo contemplado para las tecnologías resistivas es un panel de galgas extensiométricas. Se parte de la hipótesis inicial de que un galga experimenta las mismas deformaciones que la superficie sobre la cual está pegada, dependiendo de las deformaciones que sufre la galga se

obtiene una variación en la resistencia de ésta. Por lo tanto el parámetro variable y sujeto a medida es la resistencia de la galga. Los elementos principales que componen esta tecnología son una superficie ligeramente deformable y una gran cantidad de galgas extensiométricas. Esta tecnología resulta poco viable ya que tiene una resolución aproximada de 3mm debido a las dimensiones de los elementos que la componen además la instrumentación requerida para su implementación hace que sea casi imposible el uso de esta tecnología ya que se forma de elementos muy costosos, sin mencionar el costo que implicaría extender esta versión de una llanta a la medición de las cuatro llantas.

En el caso de las tecnologías capacitivas se toma el uso de un panel capacitivo como el que se utiliza comúnmente en el desarrollo de pantallas táctiles. Esta técnica utiliza una matriz ortogonal de electrodos transmisores y receptores dispuestos en una organización de múltiples nodos de contacto pequeños creados por la geometría de la estructura. Uno de ellos es encargado de transmitir un tren de impulsos lógicos. El electrodo receptor se acopla al transmisor a través del panel dieléctrico que los separa. Cuando un dedo toca el panel, el acoplamiento del campo entre emisor y receptor se favorece y es así como el tacto es detectado. Los componentes principales que conforman esta tecnología son un panel capacitivo táctil y su controlador con protocolo de comunicación USB. A pesar de los pocos elementos utilizados en el desarrollo de la tecnología capacitiva, es necesaria la importación de dichos elementos, lo cual complica su implementación y la corrección en caso de fallos. Para extender la versión de una a cuatro llantas es necesario cuadruplicar el presupuesto.

Existen diversas tecnologías ópticas utilizadas para reconocer objetos y superficies, dichas tecnologías tienen en común el uso de cámaras o sensores de luz y dispositivos de iluminación como lámparas o diodos emisores de luz (LED's). Como ejemplo de las tecnologías ópticas se observa el experimento de Reflexión Interna Total Frustrada (FTIR). En esta técnica una placa de cristal acrílico es iluminada de modo que la luz es inyectada a su interior para ser completamente reflectada. Cuando un material con índice de refracción superior al del acrílico toca la frontera de la placa, la luz es desviada de tal forma que puede ser captada por una cámara orientada hacia la placa de acrílico, en las imágenes captadas por la cámara se tiene una relación entre los pixeles iluminados y el lugar donde se realizó el contacto. Los componentes principales de esta tecnología son LED's o lámparas, un panel de acrílico y una cámara preferiblemente con un filtro pasa banda para la luz elegida. Para poner en práctica la tecnología FTIR se cuenta con la disponibilidad de los componentes en México, además, si se quiere extender a la versión de cuatro llantas solamente es necesario utilizar una placa de acrílico más grande y una cámara con mayor resolución o en su defecto alejar la cámara para que pueda captar las cuatro llantas al mismo tiempo.

Después de revisar la comparativa entre las tecnologías resistiva, capacitiva y óptica se ha llegado a la decisión de utilizar la tecnología óptica para llevar a cabo la medición del área de contacto, por lo tanto el sensor elegido para este fin será una cámara infrarroja ya que se

trabajará con iluminación infrarroja. Para la elección de la cámara se tomaron en cuenta tres tipos de cámaras infrarrojas: Cámara termográfica industrial, cámara en kit de desarrollo y cámara web modificada.

Cámara / Característica	Termográfica	Kit de estudio	Web cam
Resolución	3 Mpixeles	3 Mpixeles	1.3 Mpixeles
Velocidad	60fps	1fps	30fps
Precio aproximado	5,355.37 €	$ 75.00 USD	$ 295.00 MXP
Disponibilidad en México	NO	NO	SI
Tiempo de adquisición	Indefinido	4-6 Semanas	2-3 Días

Tabla 2.2.4: Comparación de cámaras infrarrojas.

En la tabla 2.2.4 se puede ver la comparativa de las cámaras consideradas. Las cámaras termográficas en general tiene bastantes ventajas con respecto a los otros dos tipos de cámaras, llegan a tener una velocidad de hasta 120 fps y las hay en versiones monocromáticas que mejoran la sensibilidad a la luz infrarroja [14], todas estas cualidades resultan en una mejor calidad de imagen y una reducción en la necesidad de procesamiento de imágenes cuando se trabaja con la tecnología óptica de FTIR; sin embargo son bastante caras, no se encuentran disponibles en México y el tiempo en que se puede adquirir una de ellas es indefinido. Por otro lado, la cámara embebida en un kit de estudio/desarrollo es bastante lenta para los fines que persigue este trabajo de tesis con una velocidad de 2 a 3 fps, además de no ser un kit disponible en México. Finalmente una cámara web, después de remover de ella el filtro infrarrojo, resulta ser la mejor opción para la aplicación que se requiere debido a su disponibilidad inmediata y su bajo costo en comparación con las demás, es así como se llega a la elección de la cámara web modificada como sensor para medir el área de contacto real. En la figura 2.2.4 se puede observar el diseño CAD de la cámara seleccionada. Algunas características de interés de la cámara son:

- Resolución de 1.3 Mega pixeles.
- Velocidad 30 cuadros por segundo.
- Debido a la extracción del filtro infrarrojo tiene la capacidad de ver luz infrarroja.
- Modificación a lente gran angular de 120 grados de visión.
- Protocolo de comunicación USB.

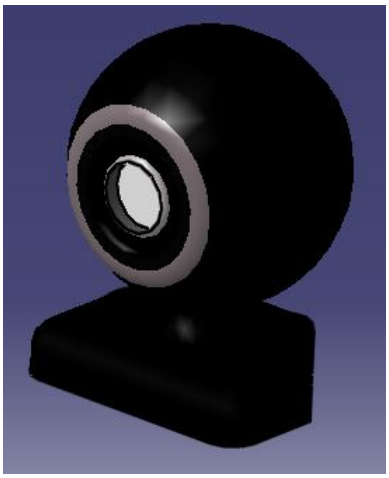

Figura 2.2.4: CAD de la cámara web.

Elección del sensor para medir cambios en la geometría de la suspensión

Pensando en las tecnologías que se pueden utilizar para detectar cambios cuando la geometría del vehículo llegara a cambiar se llega a la decisión de trabajar con un sensor de tipo resistivo ya que los movimientos en la suspensión no obedecen un orden específico con una carrera fácil de seguir, lo cual puede afectar en forma de ruido en otro tipo de sensores e incluso ocasionar daños y fallas debido al movimiento. Los sensores de tipo resistivo varían su resistencia de acuerdo a la variación del fenómeno físico a medir, en este caso se busca un sensor que pueda ser deformado al mismo tiempo que mide cuánto ha sido deformado, este es el caso de los potenciómetros de carrera lineal y los sensores de flexión. En la tabla 2.2.5 se encuentra la comparación entre estos dos sensores.

Sensor / Característica	Potenciometro	Flexión
Dimensiones	88 x 12.5 x 19.8 mm	112.24 x 6.35 x 0.7 mm
Rango de medición	0-10 K Ohm	10-25 K Ohm
Precio aproximado	$ 100.00 MXP	$ 290.00 MXP
Disponibilidad en México	SI	SI
Tiempo de adquisición	2-3 Días	2-3 Días

Tabla 2.2.5: Comparación de sensores para medición de cambios en la geometría.

A pesar de tener un par de sensores que resultan muy parecidos en las características de comparación, el sensor de flexión tiene ciertas ventajas sobre el potenciómetro de carrera

lineal para la tarea que se requiere. El potenciómetro es un elemento rígido que necesita ser montado de tal forma que el tipo de movimiento que soporta no se vea violado, además el elemento movible que causa la variación requiere especial cuidado debido a la dificultad de sujetarlo y a su tamaño reducido. Por su parte, el sensor de flexión es un dispositivo flexible totalmente deformable, para su montaje solamente requiere estar sujeto de sus extremos ya que cuenta con la protección suficiente para ser doblado, lo que hace de este sensor un dispositivo ideal para medir cambios o deformaciones en una geometría impredecible.

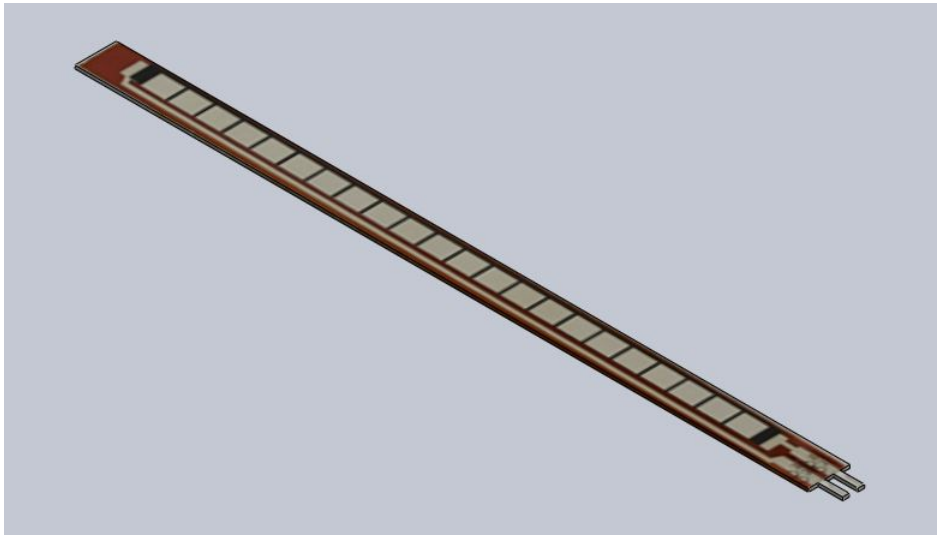

Figura 2.2.5: CAD del Sensor de Flexión.

La decisión final es utilizar el sensor de flexión del cual se puede ver el diseño CAD en la figura 2.2.5. Algunas de sus características se listan a continuación:

- 112.24 mm de largo.
- 6.35 mm de ancho.
- Longitud activa de 95.25 mm.
- Protección de silicón en la longitud activa.
- Rango de resistencia de 10K a 25K Ohms.

Elección del sensor para medir aceleración de la masa no suspendida en el eje z

La aceleración se define en física como el cambio de velocidad con respecto al tiempo, para llevar a cabo la medición de la aceleración de un objeto existe un tipo de sensor denominado acelerómetro. Actualmente es posible encontrar acelerómetros de tres ejes incluidos

en la misma placa donde se procesan las señales. El principio de operación de este tipo de dispositivos con tecnologías de sistemas microelectromecánicos (MEMS) está basado en el traspaso térmico, estos sensores miden los cambios de la transferencia de calor debidos a la aceleración de las moléculas de gas que se encuentran en su interior. En este trabajo de tesis se contempla trabajar con un acelerómetro para llevar a cabo la medición de la aceleración de la masa no suspendida sobre el eje z, para ello se realiza una comparación de distintos acelerómetros, dicha comparación puede verse en la tabla 2.2.6.

Acelerómetro / **Característica**	**ADXL335**	**ADXL345**	**LSM303DLMTR**
Resolución	8bits	13 bits	16 bits
Rango de medición	±3 g	± 2/4/8/16 g	± 2/4/8 g
Precio aproximado	$ 230.00 MXP	$ 290.00 MXP	$ 500.00 MXP
Disponibilidad en México	SI	SI	SI
Tiempo de adquisición	2-3 Días	2-3 Días	2-3 Días

Tabla 2.2.6: Comparación de acelerómetros.

En este caso se elige el sensor ADXL345 debido a que tiene un buen cociente entre la resolución, rango de medición y su precio. En la figura puede verse una imagen del diseño en CAD del sensor elegido, algunas características de este sensor son:

- Resolución seleccionable por el usuario ±2g / ±4g / ±8g / ±16g .
- Monitoreo de actividad / inactividad.
- Detección de caida libre
- Dimensiones 3 x 5 x 1 mm.
- Interfaces de comunicación I^2C y SPI de dos y tres cables.

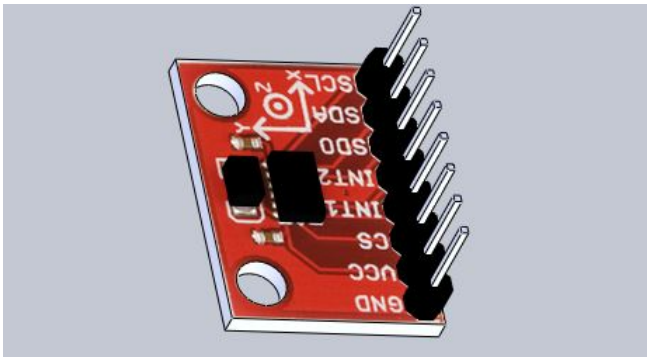

Figura 2.2.6: CAD del Acelerómetro elegido.

Elección del sensor para medir los ángulos de cabeceo y alabeo del Chasis

Los ángulos de navegación de guiñada (Yaw), cabeceo (Pitch) y alabeo (Roll) son comúnmente usados en aeronáutica para describir la orientación de un vehículo. Es también en aeronáutica donde la medición de dichos ángulos es parte de las primeras necesidades, supliendo dicha necesidad por medio de una Unidad de Medición Inercial o IMU por sus siglas en inglés. Una IMU es un dispositivo electrónico encargado de informar acerca de la orientación, velocidad y fuerzas en un vehículo, este dispositivo se compone de tres tipos de sensores: acelerómetros, giroscopios y magnetómetros.

En la tabla 2.2.7 se encuentra la comparación entre una IMU MTI-G y una IMU razor. Esta comparación con el fin de elegir cuál de ellas resulta conveniente para realizar la tarea de medición de los ángulos de cabeceo y alabeo del chasis del vehículo.

Tecnología / Característica	MTI-G	Razor
Dimensiones	58 x 58 x 22 mm	28 x 41 x 3 mm
Resolución	0.05 grados	0.01 grados
Precio aproximado	$ 25,000.00-30,000.00 MXP	$ 1200.00 MXP
Disponibilidad en México	NO	SI
Tiempo de adquisición	40 Días	2-3 Días

Tabla 2.2.7: Comparación de Unidades de Medición Inercial.

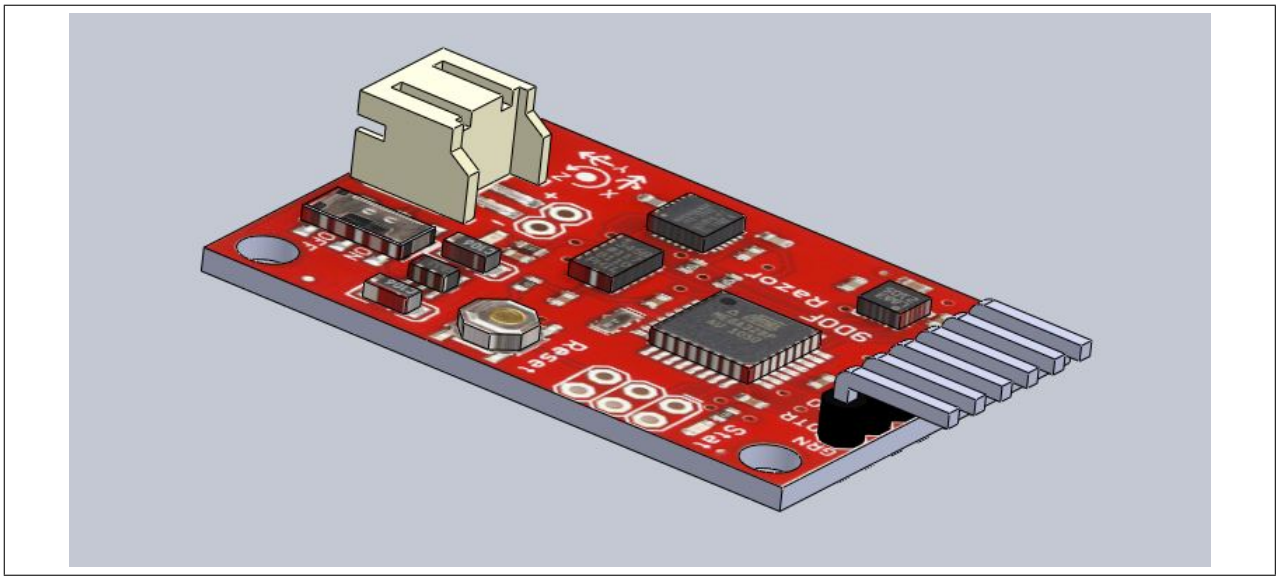

Figura 2.2.7: CAD de la IMU Razor.

A pesar de ser menos sofisticada, la elección de la IMU empleada para medir los ángulos de cabeceo y alabeo se inclina por la IMU Razor debido a que es suficiente para cumplir con la tarea que se requiere, tiene un precio accesible, cuenta con una resolución completamente aceptable y además se puede adquirir de forma casi inmediata ya que está disponible en México. En la figura 2.2.7 se observa el modelo en CAD de la IMU Razor, algunas de sus características importantes son:

- 9 grados de libertad.
- Giroscopio de tres ejes con salida digital ITG-3200.
- Acelerómetro triaxial con 13 bits de resolución ±16 g ADXL345.
- Magnetómetro digital de tres ejes HMC5883L.
- Procesador de datos ATmega328.
- Interfaz de comunicación FTDI.

Elección del sensor para medir ángulo de la dirección (Volante)

Para proponer qué tipo de sensores pueden ser empleados para llevar a cabo la medición del ángulo de la dirección, primeramente se revisa de forma breve a los componentes del sistema de dirección del auto en cuestión. Se trata de un sistema electromecánico que es actuado por un servomotor que se controla con una señal de modulación de ancho de pulsos (PWM), las señales de control que se entregan al servomotor viajan a través del aire gracias a un

sistema de comunicación por radiofrecuencia emisor-receptor. En la figura 2.2.8 se muestra un diagrama a bloques de los elementos del sistema de dirección y la forma en que se comunican.

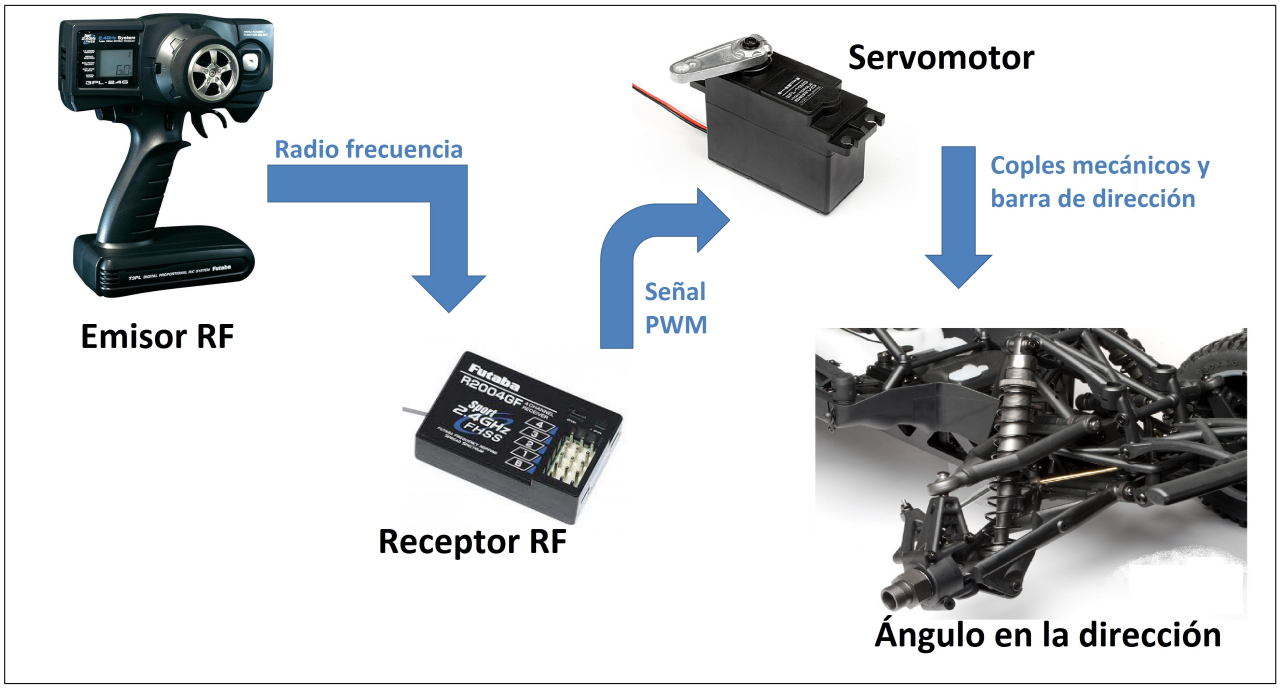

Figura 2.2.8: Diagrama a bloques del sistema de dirección.

Técnica / Característica	Receptor RF	Temporizador	Potenciómetro
Dimensiones	40 x 27 x 9 mm	1 x 1 x 15 mm	25 x 25 x 35 mm
Rango de medición	360 grados	360 grados	320 grados
Precio aproximado	$ 2,300.00 MXP	$ 0.00 MXP	$ 385.00 MXP
Disponibilidad en México	SI	SI	SI
Tiempo de adquisición	20 Días	Inmediata	2-3 Días
Tiempo de implementación	5-7 Días	2-3 Días	2-3 Días

Tabla 2.2.8: Comparación de técnicas de medición del ángulo de la dirección.

Con base en el diagrama del sistema de la dirección (Figura 2.2.8) del vehículo se puede observar que existen al menos tres maneras inmediatas de medir el ángulo de la dirección y/o sus variaciones: la primera forma es la lectura de la señal de radio frecuencia por medio de un módulo especial que pueda adquirir la frecuencia a la que trabaja el emisor, la segunda forma es la lectura de la señal de PWM por medio del temporizador en modo captura de

un microcontrolador y la tercera será medir directamente de las partes mecánicas por medio de un sensor que varíe cuando las llantas o el servomotor cambien su ángulo, esto último se puede hacer con un potenciómetro. En la tabla 2.2.8 se presenta la comparación de las tres técnicas mencionadas, todas las técnicas contemplan el uso de un microcontrolador para su implementación, así que no se toma en cuenta el precio del microcontrolador dentro de la tabla.

Tomando en cuenta los datos de la tabla 2.2.8, se puede notar que la técnica menos viable es implementar un receptor de radio frecuencia ya que tiene un precio bastante elevado en comparación con las otras dos técnicas y los tiempos de adquisición e implementación resultarían en un retraso en el proyecto, por lo cual no se tomará en cuenta. El potenciómetro puede ser causa de daños en el sistema mecánico de la dirección si no se monta de forma adecuada, en todo caso representa una demanda de esfuerzo extra en el servomotor. Ya que uno de los requerimientos en este proceso de diseño es no intervenir ni forzar el sistema mecánico del vehículo, se opta por abandonar esta técnica. Finalmente se elige el temporizador en modo captura con el cual se puede contar el tiempo en alto de la señal de PWM sin necesidad de rehacer la señal completa. Adicionalmente, no representa un gasto extra debido a que se implementa en el microcontrolador del sistema total.

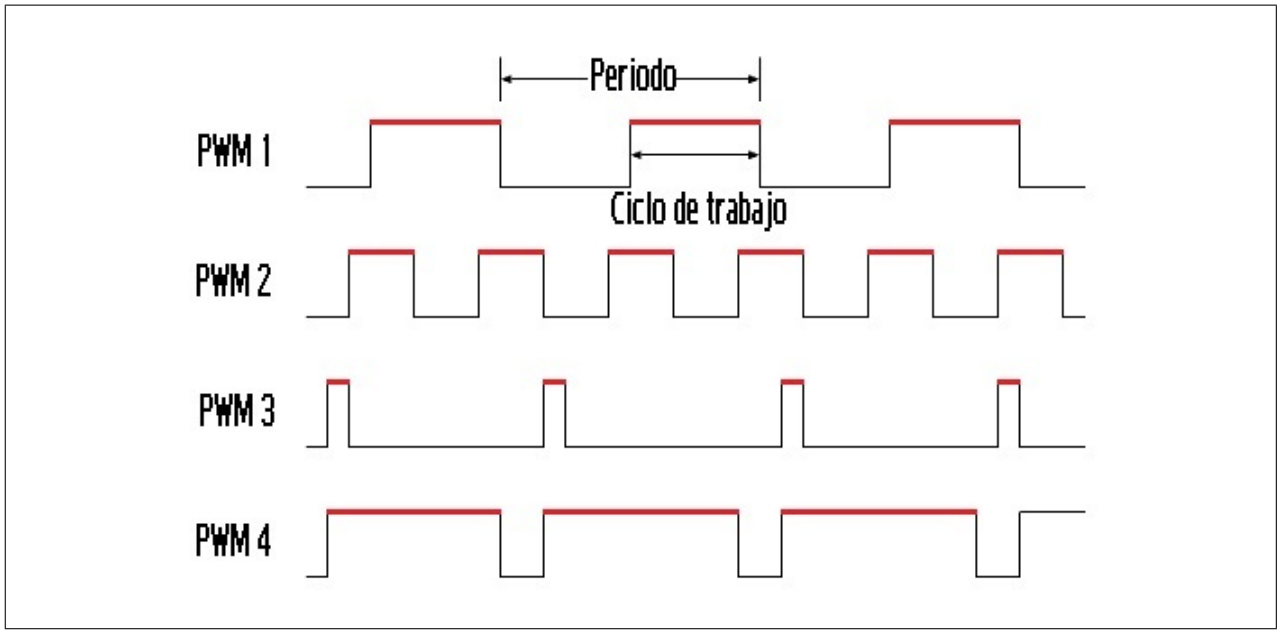

Figura 2.2.9: Señales de PWM con el ciclo de trabajo en rojo.

En la figura 2.2.9 se observan cuatro señales de PWM con el ciclo de trabajo en color rojo, la variable de interés a medir con el temporizador en modo captura es precisamente ese ciclo.

2.2.3. Planteamiento de la técnica a utilizar para vincular las variables relacionadas con la variable deseada

La tarea de vincular las variables relacionadas con la variable deseada consiste en hacer un modelo matemático que, dadas las entradas relacionadas, entregue como resultado la salida deseada. Esta tarea resulta ser de especial cuidado dentro del proceso de diseño de un Sensor Virtual ya que, incluso teniendo los datos correctos, una mala identificación del modelo puede resultar en la incorrecta construcción del Sensor Virtual llevando el proceso de diseño de vuelta a su etapa inicial. Es por ello que en esta etapa comúnmente se utilizan técnicas basadas en modelos de entrada-salida en los cuales se minimiza el error a la salida hasta llegar a un error máximo aceptable, este error máximo generalmente es pequeño, obteniendo así un modelo que describe correctamente el comportamiento del sistema estudiado.

Un enfoque ampliamente utilizado para esta etapa [2, 3, 4] es el basado en Redes Neuronales Artificiales, esta técnica de Inteligencia Artificial se caracteriza por aprender el comportamiento de un sistema a través de un entrenamiento sin necesidad del conocimiento de los modelos matemáticos que constituyen a dicho sistema. En este trabajo de tesis se pretende emplear una Red Neuronal para hacer la vinculación de las variables relacionadas con la deseada, a continuación se explica dicha técnica.

Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) son un conjunto de elementos que se interconectan para que de forma colaborativa produzcan un estímulo de salida, éstas se inspiran en las redes de neuronas que conforman el cerebro humano. Al igual que éste intentan aprender a partir de los datos que se le suministran. Algunas características que han dado popularidad a las Redes Neuronales son [15]:

- Capacidad de aprendizaje. A partir de datos de la experiencia pueden llegar a aprender el comportamiento de algún sistema mediante un entrenamiento, sin la necesidad de un estudio detallado del sistema ni de modelos matemáticos.
- Velocidad de respuesta. Una vez entrenada, una Red Neuronal tiene la facultad de dar respuestas en tiempo real. Al igual que el cerebro humano, una vez que aprende bien una tarea no necesita pensar mucho para volver a hacerla.
- Tolerancia a fallos. El aprendizaje de una Red Neuronal se extiende por todas las neuronas, de tal forma que si llega a fallar alguna neurona, la Red Neuronal continúa generando cierto número de respuestas correctas.

El elemento estructural más esencial en el sistema neuronal es la célula llamada neurona. Al igual que todas las células del organismo, una neurona consta de un cuerpo y un núcleo,

pero también tiene otros elementos importantes. El axón es una ramificación que la neurona ocupa como salida para enviar las señales de información por medio de señales electro-químicas. Las señales son enviadas a una distancia relativamente grande a partir de su origen y se ponen en contacto con otras células de distintos tipos (neuronas o no), pero sin llegar a fusionarse entre ellas, a esta zona de contacto se le denomina sinápsis. En una célula neuronal, la sinápsis recoge las señales electro-químicas que son enviadas por otras células con las que se conecta, estas señales son propagadas hacia el interior del núcleo de la neurona a través de un gran número de ramificaciones de entrada llamadas dendritas. El núcleo procesa la información recibida hasta tener una respuesta, la cual es nuevamente propagada por el axón. La información es enviada entre distintas células neuronales, formando así las redes de neuronas.

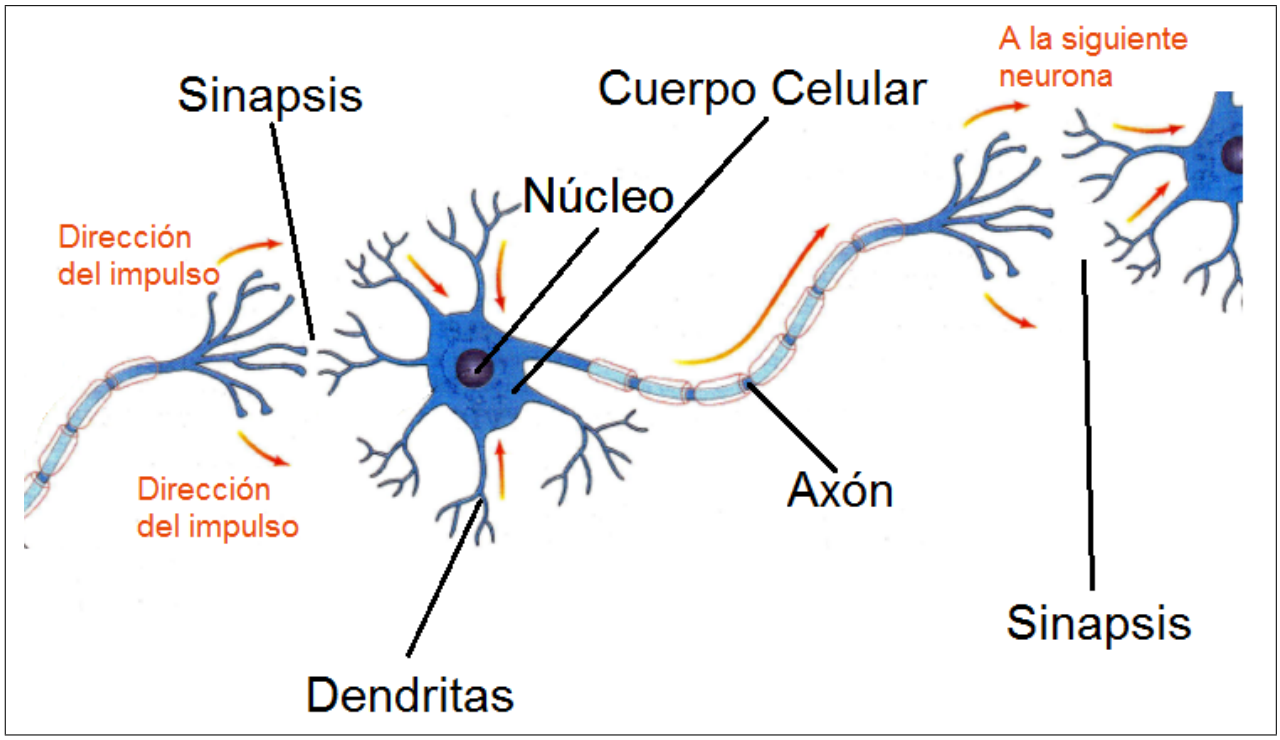

Figura 2.2.10: Descripción de una neurona humana típica.

Una Red Neuronal Artificial simula el comportamiento de los sistemas neuronales biológicos por medio de modelos matemáticos. El elemento más esencial en una RNA es la neurona artificial. Al igual que en el sistema neuronal humano, una neurona artificial tiene elementos importantes equivalentes a las dendritas (entrada), sinapsis (conexiones), núcleo (procesamiento) y axón (salida) de una neurona. Las entradas de una neurona artificial son un grupo de datos que se originan en el entorno con el que se interactúa o en otras neuronas, dicho grupo es representado por un vector de n entradas $X = x_1, x_2, ..., x_n$. Cada señal de entrada es multiplicada por un peso asociado $w_1, w_2, ..., w_n$. Cada peso corresponde a la fuerza de conexión entre la entrada y la neurona, es decir un factor de importancia, esto se representa

por un vector W. Una función de red calcula el valor total de entrada a la neurona como una suma ponderada $E = x_1w_1 + x_2w_2 + ... + x_nw_n$, que de forma vectorial puede verse como:

$$E = X^T W \tag{2.2.8}$$

Las señales E se procesan por medio de la función de activación F para calcular la salida S. Esta función es la clave que caracteriza el comportamiento de la neurona, habrá distintos tipos de neuronas dependiendo de la función F; por ejemplo funciones lineales, umbral, gausiana, sinusoidal, tangencial, sigmoidal, identidad, etc. En la figura 2.2.11 se muestra un ejemplo de una neurona artificial típica.

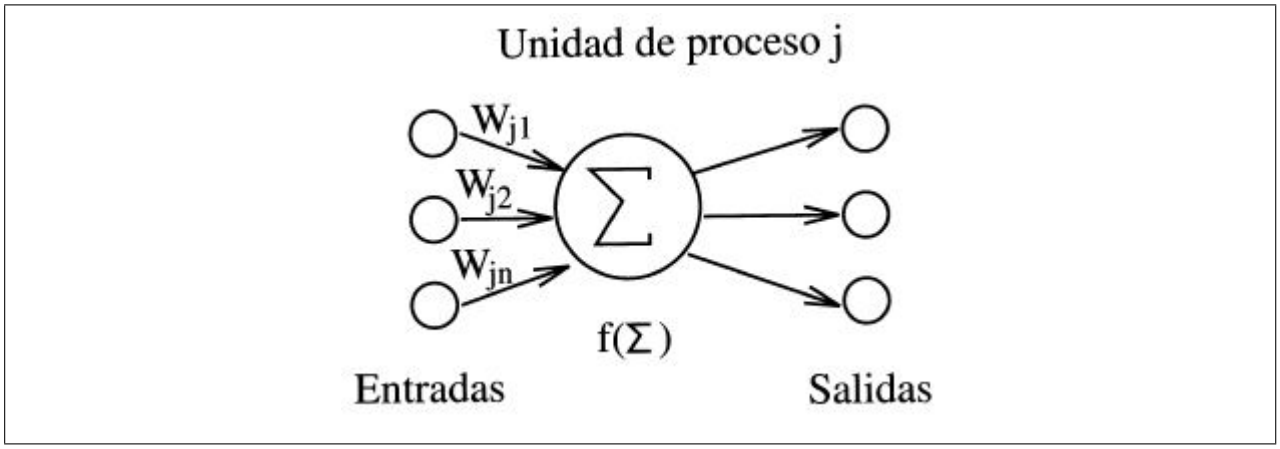

Figura 2.2.11: Estructura de una neurona artificial.

Si la información de entrada es tan compleja que no puede ser procesada por una sola neurona artificial, es posible conectar varias neuronas para que interactúen entre sí, dando origen a una Red Neuronal Artificial. A la manera en que se conectan unas células con otras se le denomina patrón de conectividad [15] o arquitectura de la red, adicionalmente, la distribución de neuronas se realiza formando capas de un cierto número de neuronas por capa. Una capa de neuronas se distingue porque sus entradas provienen de la misma fuente (valores de entrada u otra capa de neuronas) y sus salidas se dirigen hacia el mismo destino (salidas u otra capa). Dependiendo de su posición dentro del marco de la Red Neuronal Artificial, las capas pueden ser:

Capas de entrada: Es constituida por la células de entrada, las cuales reciben valores de patrones representados por vectores que sirven de entrada a la red. Sus entradas son recibidas directamente de las fuentes externas a la red.

Capas ocultas: Son constituidas por células internas a la red y no tienen ningún tipo de contacto al exterior, puede haber cualquier número de capas ocultas mientras sea posible

su procesamiento e incluso puede no haber capas ocultas. Las distintas formas de conectar las capas ocultas generan distintas tipologías de RNA.

Capas de salida: Es constituida por las neuronas de salida, que se conectan directamente con el exterior. Esta capa transfiere información de la red hacia el exterior ya que la salida de sus unidades sirve como salida de toda la red.

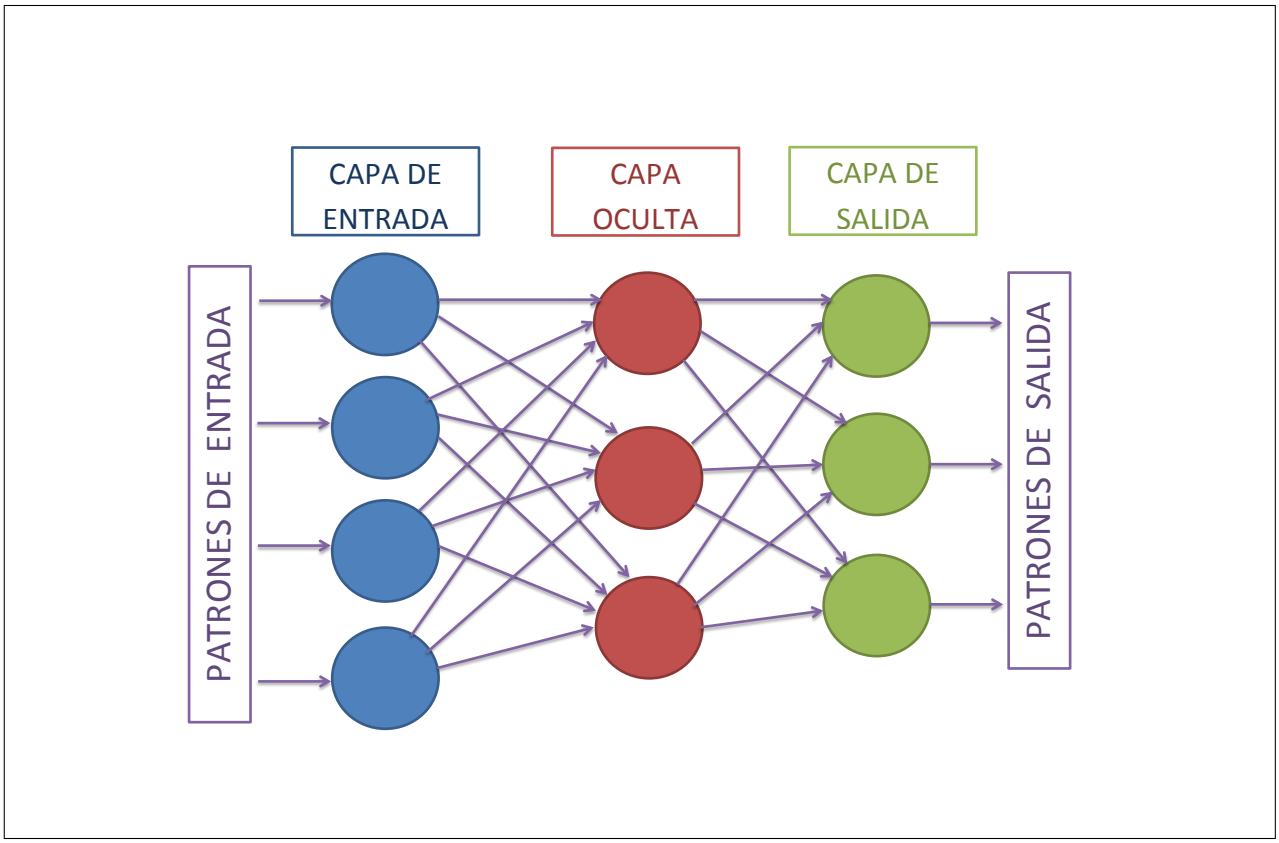

Figura 2.2.12: Esquema de una red neuronal artificial.

En la figura 2.2.12 se muestra un esquema de una RNA de tres capas (entrada, oculta, salida) totalmente interconectadas. Cada interconexión se comporta como una ruta a través de la cual viaja la información de una neurona a otra. La forma en que se comporta una RNA es simple. Cada valor de entrada es introducido a la red por medio de un vector de entrada. Cada célula en la red recibe la totalidad de sus entradas correspondientes, las procesa y genera una salida hacia la siguiente capa o hacia la salida de la red. Así se propagan todos los datos hacia adelante por la red hasta producir una salida. De esta manera, el esquema de la RNA de la figura 2.2.12 puede ser descrita matemáticamente por la ecuación 2.2.9

$$\vec{S} = F_3\left(F_2\left(F_1\left(\vec{X}\cdot W_1\right)\cdot W_2\right)\cdot W_3\right) \tag{2.2.9}$$

Donde:
W_1 = Matriz que contiene los pesos de las conexiones en la capa de entrada.
W_2 = Matriz que contiene los pesos de las conexiones en la capa oculta.
W_3 = Matriz que contiene los pesos de las conexiones en la capa de salida.
F_1 = Función de activación para las neuronas de la capa de entrada.
F_2 = Función de activación para las neuronas de la capa oculta.
F_3 = Función de activación para las neuronas de la capa de salida.
$\vec{X}$ = Vector de valores de entrada a la RNA.
$\vec{S}$ = Vector de valores de salida de la RNA.

Los valores de los pesos en las conexiones son modificados en la fase de aprendizaje para producir la Red Neuronal Artificial final. El esquema de aprendizaje de una RNA es muy importante ya que es determinante en el tipo de problemas que será capaz de resolver. La red neuronal aprende de los ejemplos disponibles del sistema o fenómeno que se quiere estudiar, dichos ejemplos deben tener las siguientes características:

- Ser significativos. Tiene que haber un cierto número de ejemplos, suficientes para que la red sea capaz de modificar sus pesos de forma eficiente.
- Ser representativos. Los ejemplos deben ser diversos y balanceados. No pueden asociarse mayormente a los efectos de un subconjunto de ejemplos, ya que de hacerlo así la red se especializará en el subconjunto de datos, perdiendo su capacidad de generalización.

Se dice que una RNA ha aprendido o ha sido entrenada de forma correcta cuando se determinan los valores de los pesos en todas sus conexiones de tal forma que la red sea capaz de resolver problemas de forma eficiente. Por lo que el proceso general de aprendizaje consiste en ir introduciendo ejemplos mientras se modifican los pesos con la finalidad de ir aproximando el comportamiento deseado. Una vez introducido el total de los ejemplos se comprueba si la red ha sido correctamente entrenada, de no ser así se vuelve a comenzar con el esquema de aprendizaje hasta cumplir con el correcto entrenamiento o algún otro criterio de finalización. La finalización en un esquema de aprendizaje se puede dar cuando:

- Se cumple un número fijo de ciclos.
- El error a la salida es menor al máximo aceptable.
- La modificación en los pesos en cada ciclo comienza a ser irrelevante.

Dependiendo del esquema de aprendizaje, éste se puede dividir en tres tipos:

Aprendizaje supervisado: En este esquema son dados los ejemplos de entradas junto con sus salidas deseadas, los pesos de las conexiones son modificados de acuerdo a la magnitud del error en la salida estimada. Si el error es muy grande, los pesos se modifican en gran magnitud pero si son parecidos se modifica de menor forma.

Aprendizaje por refuerzo: Es una variante del esquema supervisado en el que de igual forma se tienen los ejemplos de entrada junto con sus salidas deseadas, salvo que en este esquema solamente se indica si la salida es o no la correcta sin información acerca de la magnitud del error cometido.

Aprendizaje no supervisado: En este esquema solamente se cuenta con la información de los ejemplos sin nada que indique si la salida es correcta o la magnitud que llegue a tener el error. En este esquema los pesos se modifican de forma interna para determinar características de los datos de entrada. En otras palabras es un clasificador de ejemplos de entrada.

Para llevar a cabo el entrenamiento de una RNA, actualmente existen variadas herramientas de software que pueden ser utilizadas; por ejemplo: Emergent, FANN, OpenNN, MatLab, etc. En este trabajo de tesis se utilizan las cajas de herramientas de MatLab para dicho desarrollo. Una vez entrenada la Red Neuronal Artificial se procede a evaluar su desempeño por medio del proceso de validación, la cual se tratará en el siguiente apartado.

2.2.4. Técnica de validación del modelo obtenido

En el caso de una Red Neuronal Artificial la minimización del error a la salida con el conjunto de datos de entrenamiento no siempre proporciona buenos resultados. Cuando se minimiza el error con solamente un conjunto de datos, la capacidad de generalización se puede ver afectada debido a un efecto de sobre entrenamiento. Para evitar este sobre entrenamiento y lograr las salidas deseadas, se divide el banco de datos en dos subconjuntos, uno de entrenamiento y uno de validación. El conjunto de entrenamiento es utilizado para modificar los pesos de las conexiones de la misma manera que se ha explicado anteriormente, la diferencia es que en lugar de ocupar el mismo conjunto de entrenamiento para calcular el error, se utiliza el de validación. Dicho de otra manera, el proceso de validación de una RNA consiste en verificar su eficacia por medio de datos que no han sido utilizados para el entrenamiento. Los conjuntos de entrenamiento y validación deben cumplir con los siguientes rasgos:

- Ambos conjuntos deben ser independientes sin sesgos en la selección de datos de validación.
- Ambos conjuntos deben ser significativos y representativos para cumplir las propiedades ya descritas de un conjunto de ejemplos de entrenamiento.

Para la etapa de validación del Sensor Virtual se realiza inicialmente la validación de la RNA por medio del software MatLab, posteriormente se pasa a calcular datos que no hayan sido utilizados para la construcción del Sensor Virtual por medio del banco de pruebas que se utilizó para recopilar el banco de datos. Comparando los datos estimados con los datos reales, se llega a la evaluación final del funcionamiento del Sensor Virtual.

Capítulo 3

Instrumentación del sistema

3.1. Diseño del banco de pruebas

Con la finalidad de obtener un espacio adecuado para la construcción, desarrollo y evaluación del Sensor Virtual, se diseña un banco de pruebas que permita la medición del área de contacto en las llantas, soporte el peso, tenga un tamaño considerablemente mayor al del auto, y que pueda ser adherido a una pista de pruebas; de tal forma que un solo banco sea capaz de soportar cada etapa en el desarrollo del Sensor Virtual sin necesidad de construir distintos bancos de pruebas para cada tarea.

Para garantizar la medición correcta del área de contacto real de las llantas del vehículo, se utilizará la tecnología óptica llamada Reflexión Interna Total Frustrada, la cual se explica a continuación.

3.1.1. Reflexión Interna Total Frustrada

El experimento de Reflexión Interna Total Frustrada (FTIR por sus siglas en inglés) asienta sus bases en los fenómenos de la óptica geométrica. La óptica geométrica usa la noción de rayo luminoso; esto es una aproximación del comportamiento de las ondas electromagnéticas (luz) cuando se involucran objetos de tamaño mucho mayor que la longitud de onda usada. Esta aproximación permite derivar la óptica geométrica a partir de algunas de las ecuaciones de Maxwell [16].

En física, la óptica geométrica parte de los fenómenos de reflexión y refracción. A partir de ellos, basta hacer geometría con los rayos luminosos para la obtención de las leyes que gobiernan los instrumentos ópticos a que estamos acostumbrados como lo son espejos y lentes.

Reflexión

La reflexión es el cambio de dirección de una onda, que al estar en contacto con la superficie de separación entre dos medios cambiantes, regresa al medio donde se originó. Los ejemplos más comunes son la reflexión de la luz y el sonido.

La reflexión en óptica ocurre cuando los rayos de luz que intentan incidir en una superficie, chocan en ella, se desvían y regresan al medio que salieron formando un ángulo igual al de la luz incidente. La reflexión cumple con las leyes de Huygens [16] que rigen todo el movimiento ondulatorio:

1a Ley: El rayo incidente, el rayo reflejado, el refractado y la normal, se encuentran en un mismo plano.

2a Ley: Los ángulos de incidencia y reflexión son iguales, entendiendo por tales los que forman respectivamente el rayo incidente y el reflejado con la normal a la superficie de separación trazada en el punto de incidencia.

Refracción

La refracción es el cambio de dirección que experimenta una onda al pasar de un medio a otro. Solo se produce si la onda incide oblicuamente sobre la superficie de separación de ambos medios y si estos tienen índices de refracción distintos. Un ejemplo común es el de la interfaz aire-agua al sumergir un lápiz en un vaso de agua y visualmente pareciera que el lápiz se ha torcido. Al igual que la reflexión, la refracción cumple con las leyes de Huygens [16].

Reflexión Interna Total

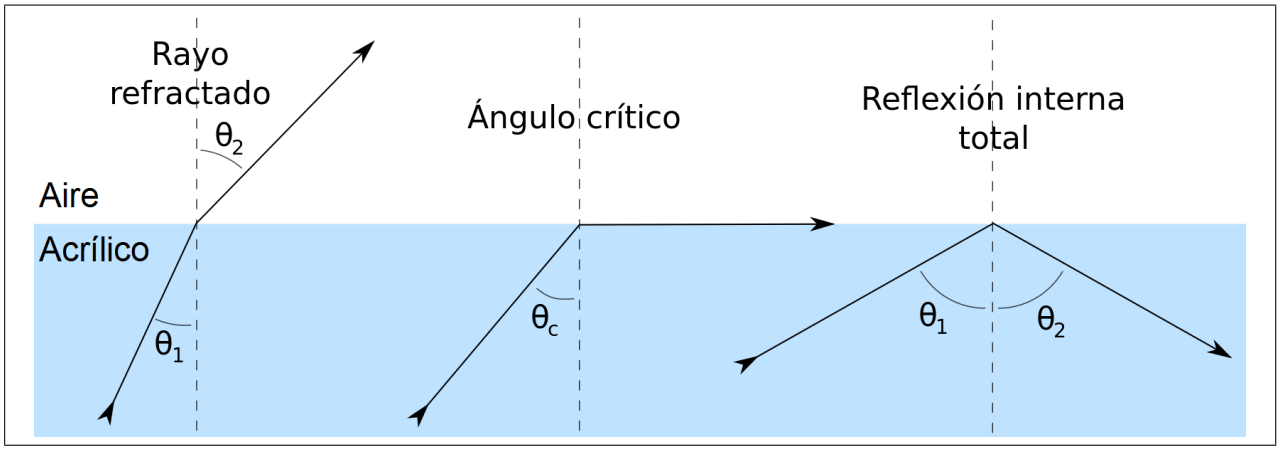

Figura 3.1.1: Refracción y Reflexión Interna Total.

La reflexión interna total es el fenómeno que se produce cuando un rayo de luz atraviesa un medio de índice de refracción menor que el índice de refracción en el que éste se encuentra, se refracta de tal modo que no es capaz de atravesar la superficie entre ambos medios reflejándose completamente. Este fenómeno solo se produce para ángulos de incidencia superiores a un cierto valor crítico en los cuales la luz deja de atravesar la superficie y es reflejada internamente de manera total.

El mejor ejemplo de éste fenómeno se puede encontrar en la fibra óptica que se utiliza para conducir luz sin pérdidas de energía. En la figura 3.1.1 se ilustra la interfaz aire-acrílico y el comportamiento ideal de un rayo incidente a distintos ángulos.

Experimento de Reflexión Interna Total Frustrada (FTIR)

La tecnología de FTIR está basada en el concepto óptico de reflexión interna total con una superficie extra para interacción, con la cual se modifica la interfaz que alberga dicho fenómeno para que éste se vea frustrado. Las ondas electromagnéticas se transmiten al interior de un material en donde serán reflejadas dentro de sus bordes de tal modo que se tenga una Reflexión Interna Total [14].

Comúnmente, en el experimento de FTIR se utiliza un panel transparente de acrílico con un conjunto de LED's infrarrojos que son colocados alrededor con la finalidad de inyectar luz infrarroja dentro del acrílico. Cuando un usuario toca alguna parte del acrílico, la luz escapa y es reflectada en el punto de contacto del dedo debido al mayor índice de refractividad causado por el cambio en la interfaz aire-acrílico por una interfaz dedo-acrílico. En la parte contraria a la zona de contacto se puede situar una cámara con filtro pasa-banda infrarrojo y así obtener la imagen de la zona de contacto. El resultado del experimento mencionado anteriormente se puede observar en la figura 3.1.2.

Figura 3.1.2: Experimento de Reflexión Interna Total Frustrada.

3.1.2. CAD del banco de pruebas

Tomando en cuenta las tareas que se pretenden realizar y la tecnología a emplear, se realiza el Diseño Asistido por Computadora del banco de pruebas, el cual se puede observar en la figura 3.1.4. El banco consta de una placa de acrílico, LED's infrarrojos y una cámara web modificada para captar luz infrarroja. Para el experimento, la placa de acrílico será iluminada con LED's infrarrojos por las orillas y con la cámara con filtro pasa-banda IR situada en la parte de abajo será posible captar la imagen del área de contacto real gracias a la tecnología FTIR.

Las dimensiones del banco son: 1.20 metros de ancho, pensando en que el auto pesa un poco más de 12 Kilogramos y la placa de acrílico debe ser capaz de soportar ese peso. 2.44 metros de largo y 0.8 metros de alto, se toma la altura promedio entre algunas mesas y espacios físicos que pueden fungir como partes de una pista pensando en que el banco podrá funcionar como pista de pruebas e incluso puede adjuntarse a una pista. Además se toma en cuenta que la cámara necesita una distancia de al menos 60 centímetros para poder obtener una imagen satisfactoria. Cabe mencionar que, para la construcción del banco de pruebas se requiere de elementos adicionales que no son tomados en cuenta en el CAD, tales como: cables estructuras metálicas, tripie para cámara, etc.

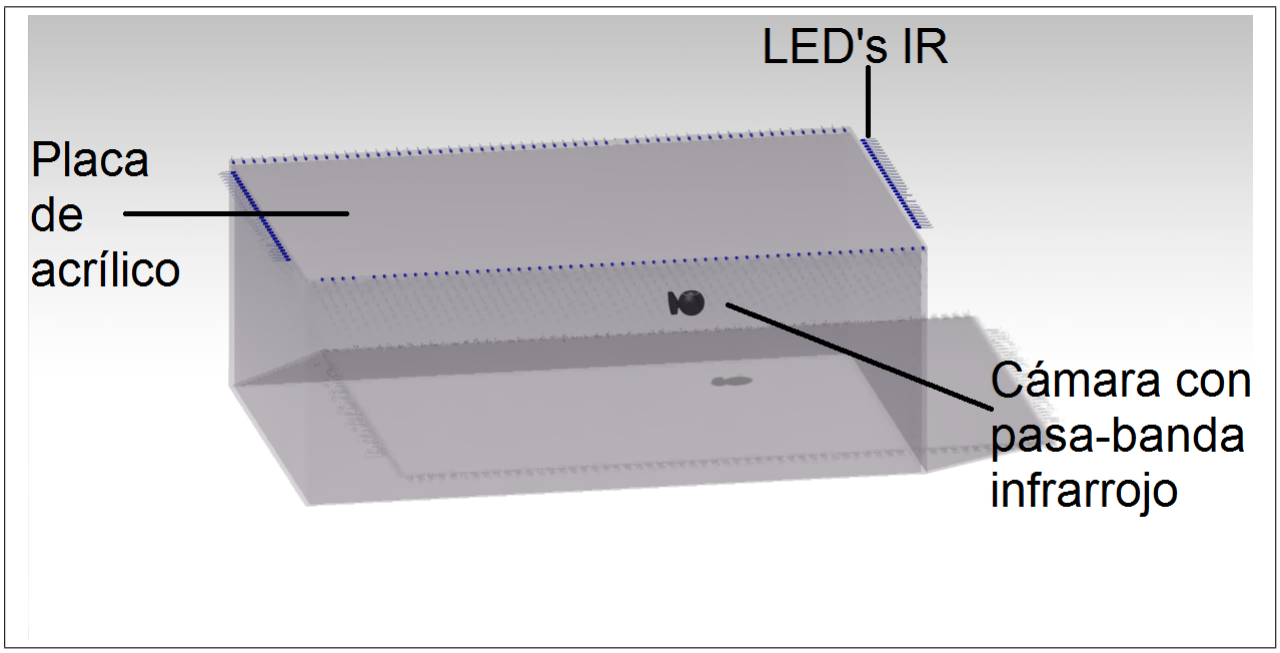

Figura 3.1.3: CAD del banco de pruebas.

3.1.3. Banco de pruebas final

El banco de pruebas final es un estilo de mesa cuya base es construida con perfil de aluminio estructural de 4.5 cm y en la parte superior tiene una placa de acrílico de 9 milímetros de grosor,

1.20 metros de ancho y 2.44 metros de largo. La base de aluminio tiene como dimensiones 0.75 metros de alto, 1.10 metros de ancho y 2.34 metros de largo. El acabado a las orillas del acrílico es de tipo espejo para facilitar la entrada de luz infrarroja que le será brindada por una tira de LED's IR. En la figura 3.1.4 se puede observar una imagen del banco de pruebas terminado.

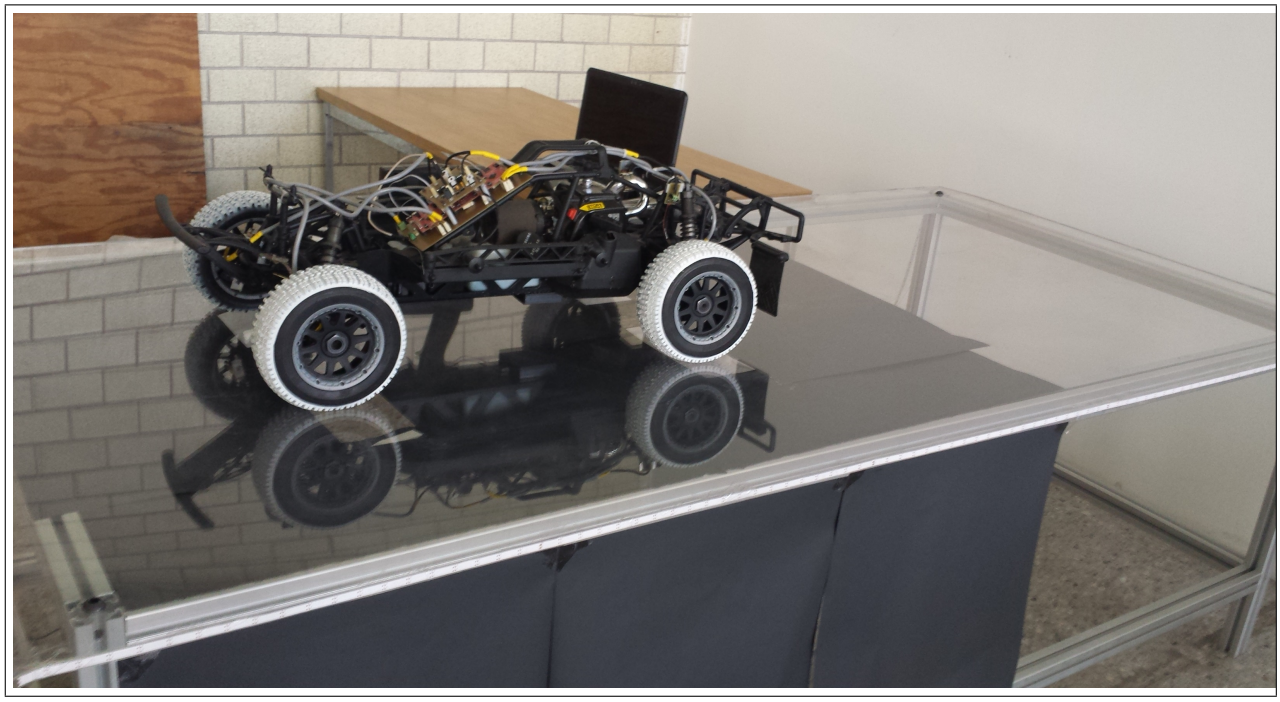

Figura 3.1.4: Banco de pruebas.

Con respecto a la cámara que se utiliza en el banco de pruebas, se trata de una cámara web modificada para captar el espectro de luz infrarroja, posee una resolución de 640 x 480 pixeles a una velocidad de 30 fps y tiene montado un lente gran angular de 120 grados de visión. La cámara es fijada en un tripie de 35 cm de altura con el fin de eliminar movimientos no deseados. Lo anterior brinda un área de visión teórica de 138.56 x 138.56 cm (19,198.9 cm^2).

La puesta en marcha de la cámara web, como con la mayoría de cámaras de este tipo, es *plug and play* en sistemas operativos Windows como el que se maneja en la PC que controla el proceso y por lo tanto no requiere de ningún procedimiento extra para ser ocupada.

3.1.4. Adquisición y tratamiento de imágenes

Como se mencionó anteriormente, en conjunto con el banco de pruebas se utiliza una cámara web modificada con lente gran angular fijada a un tripie para adquirir imágenes del área de contacto real en las llantas del vehículo. Todo esto es posible gracias al fenómeno

de Reflexión Interna Total Frustrada (FTIR) que brinda una imagen específica del área de contacto iluminada en luz infrarroja.

Para realizar la adquisición de imágenes es necesario conectar la cámara web a una computadora personal (PC) que en este caso trabaja con el sistema operativo Windows 7. A través de un programa dedicado a la adquisición de imágenes se genera un objeto en el *software* de Matlab para trabajar la adquisición de datos tipo imagen, el objeto trabaja con un disparo para adquirir algún número de datos tipo imagen y guardarlos en forma de matrices numéricas en variables temporales, al finalizar la adquisición de datos se tiene la opción de guardar dichos datos como matrices numéricas ó como imágenes. En la figura 3.1.5 se muestra un ejemplo de imagen adquirida mediante el sistema descrito.

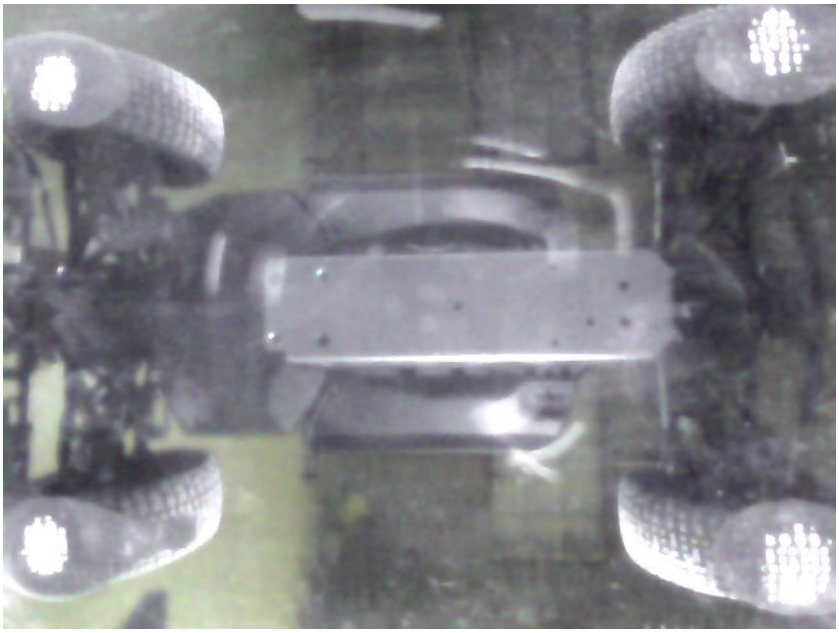

Figura 3.1.5: Imagen adquirida mediante la cámara web modificada.

Una vez adquirido el número de imágenes elegido por el usuario es necesario realizar un tratamiento de cada imagen para obtener el valor del área de contacto en cada llanta del vehículo. El tratamiento consta de tres etapas y se realiza por medio de las herramientas en los *toolboxes* de adquisición (*Image Acquisition Toolbox*) y procesamiento (*Image Processing Toolbox*) de imágenes de Matlab: en la primera etapa se transforma la imagen original a escala de grises, en la segunda etapa se transforma la imagen en escala de grises a binario (blanco y negro) y se separan las regiones de interés, finalmente, en la tercera etapa se aplica un umbral que separa el espectro de luz infrarroja. En la figura 3.1.6 se muestra el resultado de aplicar el tratamiento descrito a la imagen de la figura 3.1.5.

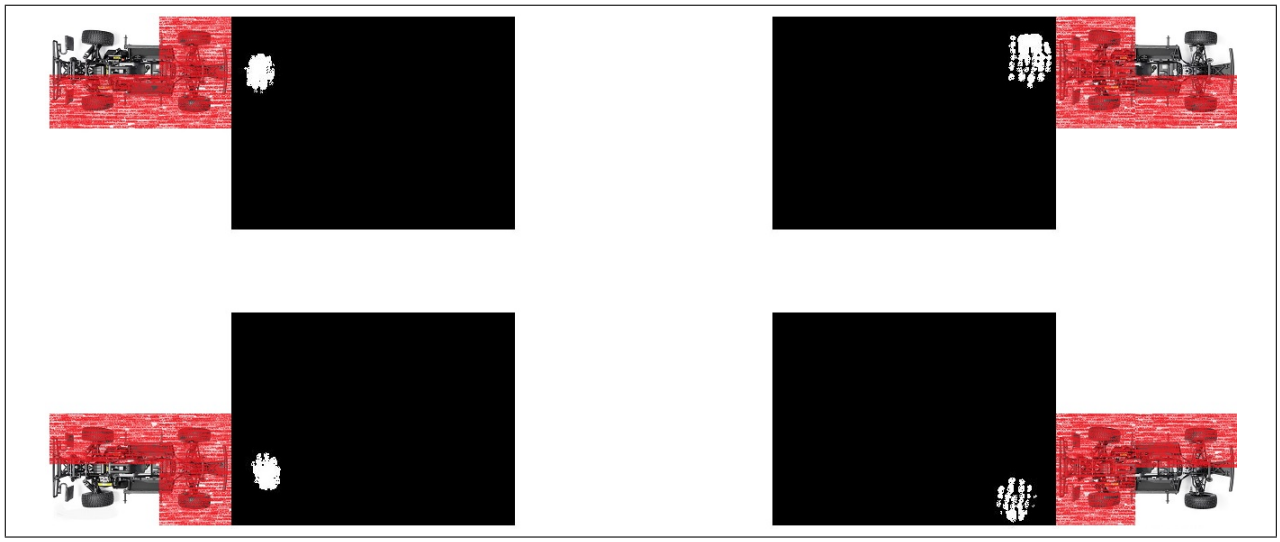

Figura 3.1.6: Tratamiento de la imagen adquirida.

Después de culminado el proceso de tratamiento, cada fotografía se somete a comparación con otra imagen de área conocida adquirida bajo las mismas condiciones con que se toman todas las imágenes para que, por medio de operaciones de proporcionalidad (regla de tres simple), se obtenga el área de contacto real de cada llanta del vehículo. En la figura 3.1.7 se observa un ejemplo de imagen de área conocida (2400 mm^2) que sirve como punto de comparación.

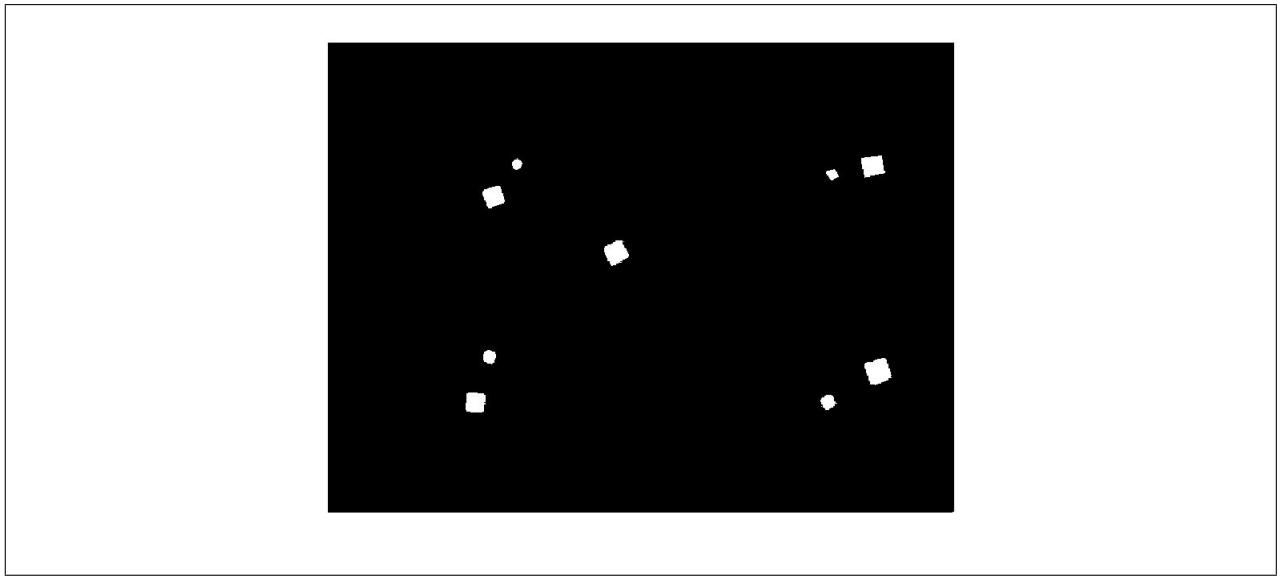

Figura 3.1.7: Tratamiento de la imagen de calibración.

Pixeles	Área en pixeles	Área en milímetros cuadrados
1 pixel	1	0.2881

Tabla 3.1.1: Área asignada en pixeles y milímetros cuadrados.

Uno de los requisitos que debe cumplir un pixel para ser considerado en el área de contacto es estar dentro de la región de interés asignada para cada llanta, además de estar próximo a una vecindad de pixeles de su mismo tipo. Por medio de las herramientas en el *toolbox* de procesamiento de imágenes de Matlab se obtiene el número total de pixeles que están dentro del área de contacto y se asigna un valor de 0 ó 1 para sumar su área total en pixeles. En la tabla 3.1.1 se muestra la asignación del área en pixeles y el área de contacto en milímetros que representa.

Para la figura 3.1.7, el total de área en pixeles es de 8330 con lo anterior se obtiene un área de 2,400 mm^2 (área conocida previamente) y se corrobora el correcto funcionamiento del sistema de visión y la comparación con la imagen de calibración (imagen de área conocida). Para finalizar solamente se necesita tomar el área en pixeles de cada imagen de contacto del vehículo para obtener su área en milímetros cuadrados; por ejemplo, para la figura 3.1.6 se tiene un área de contacto en la llanta trasera derecha de 982.47 mm^2, en la llanta trasera izquierda 1031.4 mm^2, en la llanta delantera derecha 952.8 mm^2 y en la llanta delantera izquierda 1745.0 mm^2.

De lo obtenido anteriormente se concluye que, para garantizar la calidad en las mediciones por medio del sensor visual es necesario realizar una calibración previa a los experimentos y corroborar el funcionamiento correcto con una imagen de área conocida.

3.2. Instrumentación del vehículo

La instrumentación y montaje de sensores en el vehículo a escala se realiza con la finalidad de colocar los sensores en el lugar adecuado para obtener las variables relacionadas elegidas en la etapa de diseño del sensor virtual sin intervenir ni mermar el funcionamiento mecánico del vehículo. El objetivo principal es obtener una variación en cada sensor cuando la variable relacionada cambie sin necesidad de recrear el valor exacto de la variable relacionada ya que la lectura de cada sensor físico será llevado como entrada al Sensor Virtual y la única medición real recreada será la del área de contacto.

3.2.1. Montaje de los sensores de flexión

Los sensores de flexión elegidos son una clase de galgas resistivas que cambian su resistencia al ser deformados, serán empleados para detectar los cambios en la geometría de la suspensión. Debido a que los movimientos de la suspensión son variados y poco predecibles, se opta por buscar puntos fijos en los cuales se pueden montar coples para adjuntar los sensores de flexión a la suspensión de forma indirecta, sin causar restricciones en el movimiento de los elementos de la suspensión.

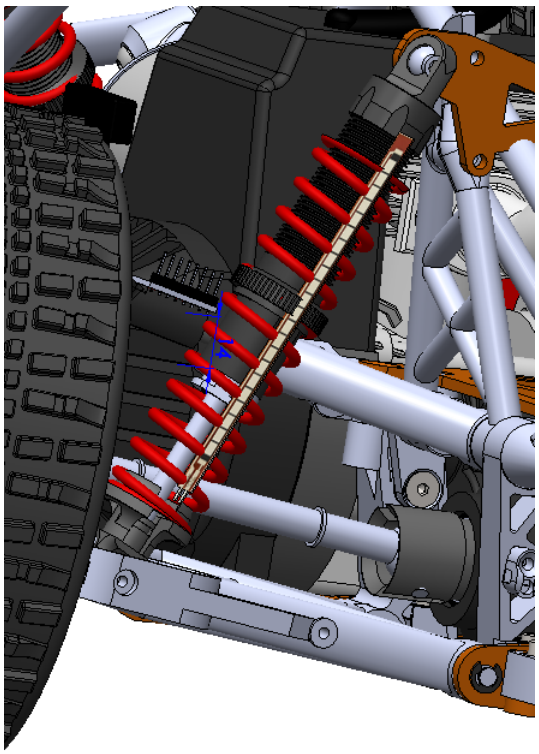

Figura 3.2.1: Montaje de sensores de flexión.

En la figura 3.2.1 se puede observar un CAD del sensor de flexión sobrepuesto en la zona de la suspensión trasera derecha. Puede notarse que la longitud del sensor es insuficiente para ser colocado a lo largo del amortiguador, cuyos extremos están anclados a la parte más alta y más baja del sistema de suspensión. Es por ello que se opta por tomar un punto fijo y un punto móvil con referencia en la masa no suspendida para montar coples que cambien su posición cuando la suspensión se deforme y así por medio del sensor de flexión obtener una variación directa al cambio de la geometría en cada parte de la suspensión (delantera, trasera, izquierda o derecha).

3.2.2. Montaje de los acelerómetros

El sensor que se utilizará para medir el cambio en la aceleración de la masa no suspendida será un acelerómetro de resolución elegible entre $\pm 2g$ / $\pm 4g$ / $\pm 8g$ / $\pm 16g$. Con la ventaja de que al cambiar el rango de medición se tiene la capacidad de eliminar cierta cantidad de ruido generada por las vibraciones del motor. Un acelerómetro no mide necesariamente la aceleración en el sistema de coordenadas que se encuentre el vehículo, sino en el sistema de referencias propio del sensor. Esto quiere decir que cuando la masa no suspendida del auto tiene una aceleración nula, el sensor medirá la aceleración debida a la gravedad y si el vehículo se encontrara en caida libre, el sensor indicaría una aceleración nula. Lo anterior también puede evitarse gracias a las funciones *off-set* del acelerómetro.

Figura 3.2.2: Montaje de acelerómetros.

En la figura 3.2.2 se muestra el CAD del montaje de los acelerómetros en la masa no suspendida del vehículo, parte trasera (imagen izquierda) y delantera (imagen derecha). Como puede observarse, para la parte trasera se pretende montar el sensor sobre la rótula que une la masa de la llanta con la horquilla superior de la suspensión. Para la parte delantera el sensor será montado directamente sobre la masa de la llanta. Al igual que en el caso del sensor de flexión, los acelerómetros serán unidos al vehículo por medio de coples mecánicos que los fijen al lugar de diseño.

3.2.3. Montaje de la IMU

Para llevar a cabo la medición en las variaciones de los ángulos de cabeceo y alabeo del chasis se utilizará una IMU de 9 grados de libertad que consta de tres sensores: un acelerómetro, un giroscopio y un magnetómetro. Por medio de un procesador ATmega328, la IMU adquiere, procesa y envía los datos de los ángulos de guiñada, cabeceo y alabeo.

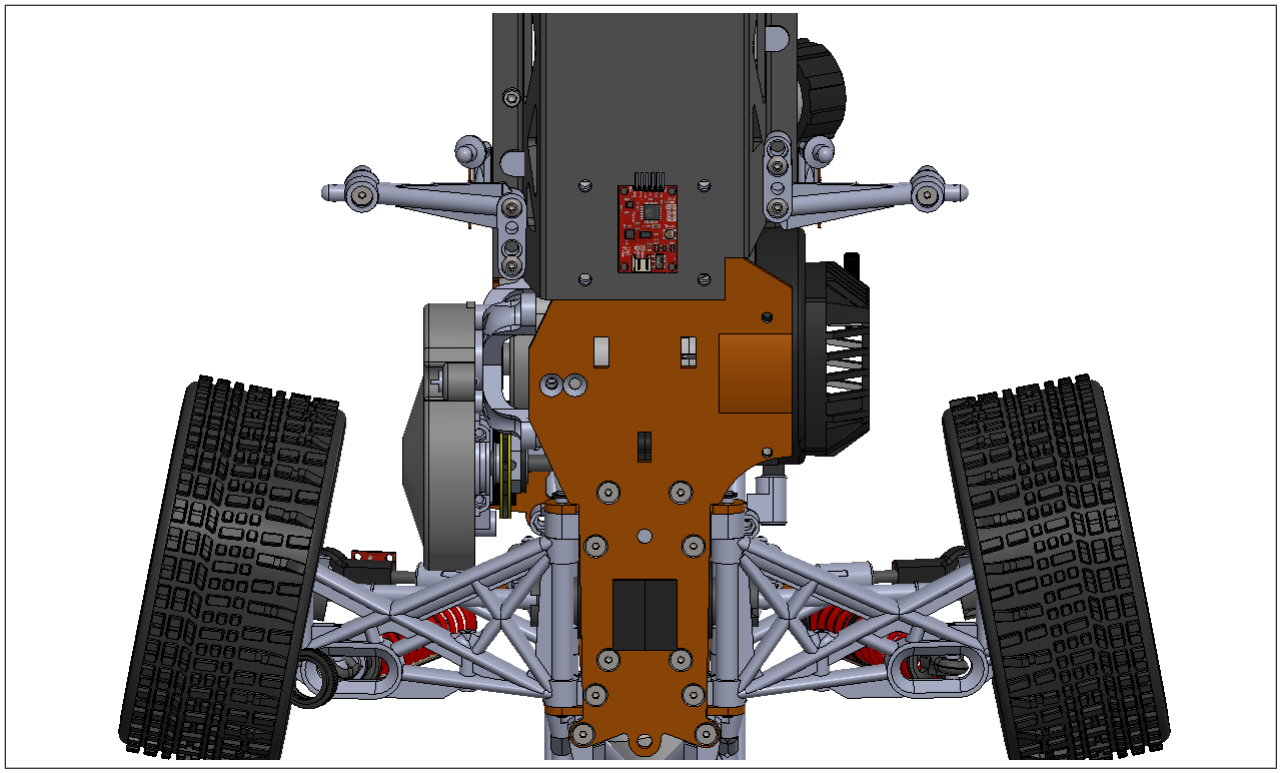

Figura 3.2.3: Diseño del montaje de la Unidad de Medición Inercial.

En la figura 3.2.3 se muestra el CAD del montaje de la IMU al chasis del vehículo. Se busca colocar el sensor lo más próximo al centro de masa del vehículo sin intervenir en el armado. Ya que en la parte superior del chasis se encuentran los elementos electromecánicos del motor y el sistema de dirección, se plantea la forma de colocar el sensor en la parte inferior del chasis justo bajo el sistema de control de la dirección. Para ello se construyó una caja protectora que evite que el sensor sea golpeado al mismo tiempo que lo adhiera al chasis para permitirle la correcta medición de los ángulos.

3.2.4. Montaje del temporizador

En el caso de la medición del ángulo de las llantas delanteras se utiliza el temporizador de un microcontrolador en modo captura parar obtener el tiempo en alto de la señal de PWM que es enviada al servomotor del sistema de dirección para controlarlo. De forma objetiva, no se está tratando con un sensor físico e incluso no se está pensando en medir directamente el ángulo en la dirección; sin embargo se toma una medición indirecta de una variable que tiene una relación directa al cambio del ángulo de la dirección, así se tendrá una entrada representativa al sensor virtual.

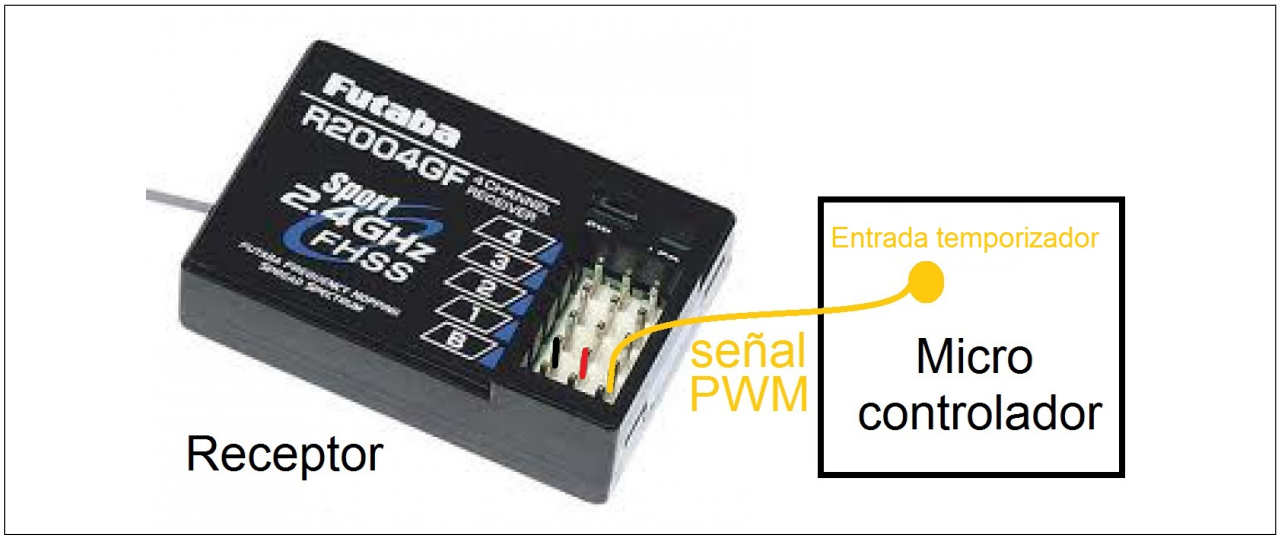

Figura 3.2.4: Diseño del montaje del Temporizador.

La figura 3.2.4 muestra una imagen del módulo receptor del sistema de dirección y el lugar físico donde pretende obtenerse la señal de PWM que será llevada al temporizador del microcontrolador para medir su tiempo en alto. El módulo consta de cinco canales de conexión con tres posiciones cada uno, el primer canal está dedicado para la conexión de la batería de alimentación mientras que los cuatro canales restantes son utilizados para retransmitir las señales de control recibidas. En lugar de emplear un sensor físico, se captura la señal del canal 1 del módulo receptor ya que este canal es asignado para controlar el servomotor del sistema de dirección. El canal 1 consta de tres posiciones que en la imagen han sido iluminados en color negro, rojo y amarillo, respectivamente. La posición en color negro es empleada como conexión de la tierra eléctrica, en la posición de color rojo se conecta la entrada de voltaje para el servomotor y en la posición de color amarillo se toma la señal de PWM que controla la posición del servomotor. Es en esta última posición donde se encuentra la señal que se desea observar, por lo cual se hace una conexión para llevar dicha señal hacia un pin de entrada del microcontrolador para utilizar el temporizador.

3.3. Puesta en marcha de sensores

La instrumentación del vehículo consta de un total de 10 sensores físicos. El primer sensor es una Unidad de Medición Inercial (IMU) y es utilizado para medir los ángulos de cabeceo y guiñada del vehículo. Otro sensor es el temporizador de un microcontrolador que será usado para captar el tiempo en alto de la señal de PWM que controla la dirección de las llantas delanteras. También se utilizan cuatro sensores de flexión para captar la deformación en la geometría del vehículo. Finalmente se emplean cuatro acelerómetros que medirán el cambio en la aceleración de las llantas en el eje z.

3.3.1. Unidad de Medición Inercial (IMU)

Para llevar a cabo la medición del cambio en los ángulos de cabeceo y alabeo del vehículo se implementa una IMU modelo Razor de la línea Sparkfun con dimensiones de 28 x 41 x 3 mm, 9 grados de libertad, una resolución en la medición de hasta 0.01 grados, giroscopio ITG-3200 de tres ejes con salida digital, acelerómetro triaxial ADXL345 con 13 bits de resolución ±16 g, magnetómetro digital de tres ejes HMC5883L y un procesador de datos ATmega328 que controla el proceso en los sensores de la IMU y brinda una comunicación con otros dispositivos por medio de una interfaz de comunicación FTDI.

El montaje de la IMU (figura 3.3.1) se realiza por la parte de abajo del auto mediante una placa maquinada en Nylamid de dimensiones 67 x 83 x 20 milímetros, que cumple con las funciones de cubrir, proteger de golpes y sujetar el sensor al auto. Las conexiones necesarias se implementan con cable plano de 4 vías: Tierra, Voltaje, Rx y Tx. Tanto el voltaje de alimentación como el de niveles lógicos es de +3.3 Volts.

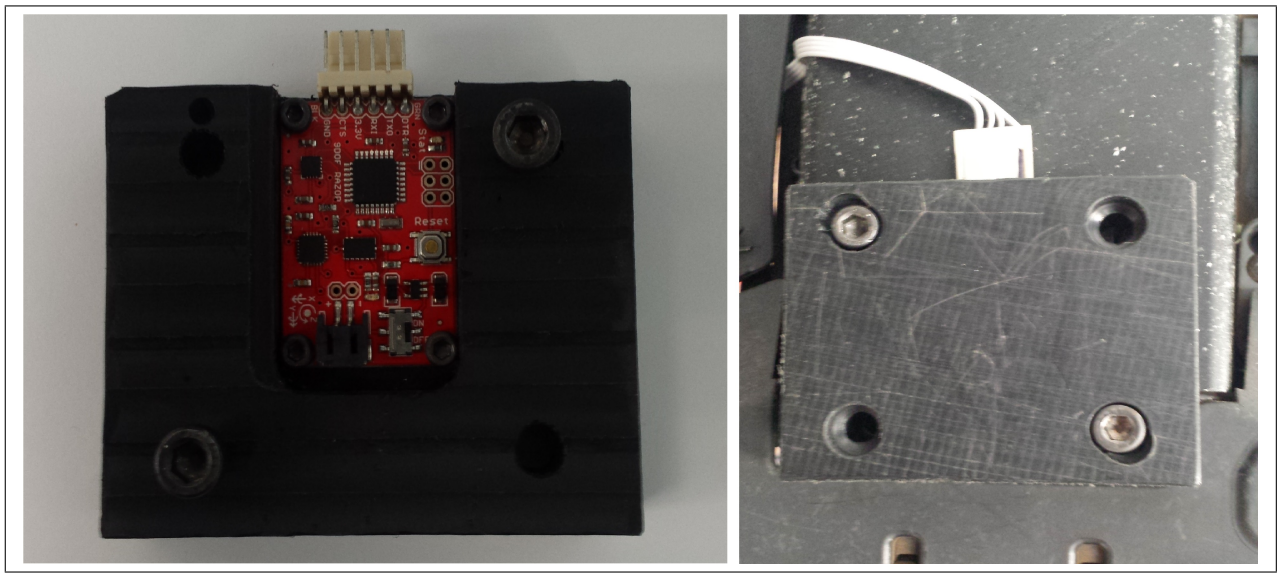

Figura 3.3.1: Montaje y cableado de la IMU.

Para poner en marcha la IMU es necesario conectarla a un convertidor RS-232 a USB y verificar que al conectar el convertidor a la PC sea reconocido, generando un puerto COM virtual. Una vez que el convertidor es conectado a la IMU y a la PC, es momento de abrir una terminal comunicada al puerto COM generado en la PC a una velocidad de 57600 baudios, el programa cargado por defecto en el procesador muestra en pantalla de manera continua renglones con nueve datos numéricos separados por comas, que corresponden a las mediciones de los sensores de la IMU. Con la ayuda de los nueve datos es posible llegar al valor de los ángulos de guiñada, cabeceo y alabeo; sin embargo, se cuenta con *firmware* disponible en la página web de Sparkfun con un proyecto desarrollado por Jordi Munoz el cual se encarga de recoger los datos

y gracias a la redundancia otorgada por tres tipos de sensores distintos se entrega como salida los ángulos de guiñada, cabeceo y alabeo. Para cargar el programa mencionado previamente se requiere reprogramar el procesador de datos ATmega328 que se encuentra embebido en la IMU.

Con la modificación en el procesador de datos se obtienen renglones continuos con los caracteres "#YPR=" seguidos por los valores de los ángulos de guiñada, cabeceo y alabeo, en ese orden, y finalizando con nueva línea y retorno de carro. La transmisión de datos se continúa haciendo a una velocidad de 57600 baudios por medio del puerto COM.

Por parte del dispositivo dedicado a adquirir y procesar los datos de la IMU es necesario tener una comunicación RS-232 y un programa dividido en dos partes: la primera parte del programa es adquirir el dato otorgado y la segunda parte se dedica a procesar el dato obtenido. Para reconocer cuando un dato completo ha sido obtenido se emplean los caracteres de nueva línea y retorno de carro sabiendo que se ha llegado al final de la medición. Para procesar los datos obtenidos se puede leer los caracteres "#YPR=" y desecharlos, posteriormente tomar los valores de los ángulos que son separados con comas hasta llegar a los caracteres de nueva línea y retorno de carro nuevamente. El proceso se puede repetir de forma iterativa hasta que sea necesario.

Con el fin de mostrar el desempeño del sensor, se realizan experimentos en los cuales se somete el vehículo a los movimientos que se pretende captar mientras se toman mediciones con la IMU. Los experimentos realizados son principalmente por inclinación del vehículo y adicionalmente se realiza un experimento de cambio en la dirección de las llantas delanteras. Las gráficas con los resultados de los experimentos se muestran en las figuras 3.3.2 a 3.3.7.

En todos los experimentos se toma un total de mil muestras. El experimento de auto estático consiste en realizar mediciones en el auto encendido pero sin avanzar ni mover la dirección de las llantas. Los experimentos de inclinación consisten en inclinar el auto de forma manual hacia enfrente, atrás, izquierda o derecha, según sea el caso. El experimento de cambio en la dirección consiste en mover la dirección de las llantas delanteras de forma remota por medio del control de radiofrecuencia. Para los experimentos de inclinación se inicia dando un pequeño golpe por la parte que se inclinará el vehículo para obtener un cambio brusco, posteriormente se inclina de forma suave y se retorna a la posición inicial, se vuelve a inclinar y regresar a la posición inicial cada vez con un ángulo mayor, hasta llegar al límite mecánico del vehículo. En el caso del experimento de cambio en el ángulo de la dirección, se comienza con la dirección orientada totalmente hacia la izquierda y se hace un cambio brusco a la derecha, se deja la dirección en la posición media por un momento y luego se comienza a cambiar la dirección de forma suave de izquierda a derecha hasta que se termina de tomar las mediciones.

En las gráficas 3.3.2 a 3.3.7, se presentan los resultados obtenidos en los experimentos.

Las mediciones obtenidas para el ángulo de cabeceo son presentadas mediante una línea en color azul y las mediciones obtenidas para el ángulo de alabeo son presentadas mediante una línea en color verde.

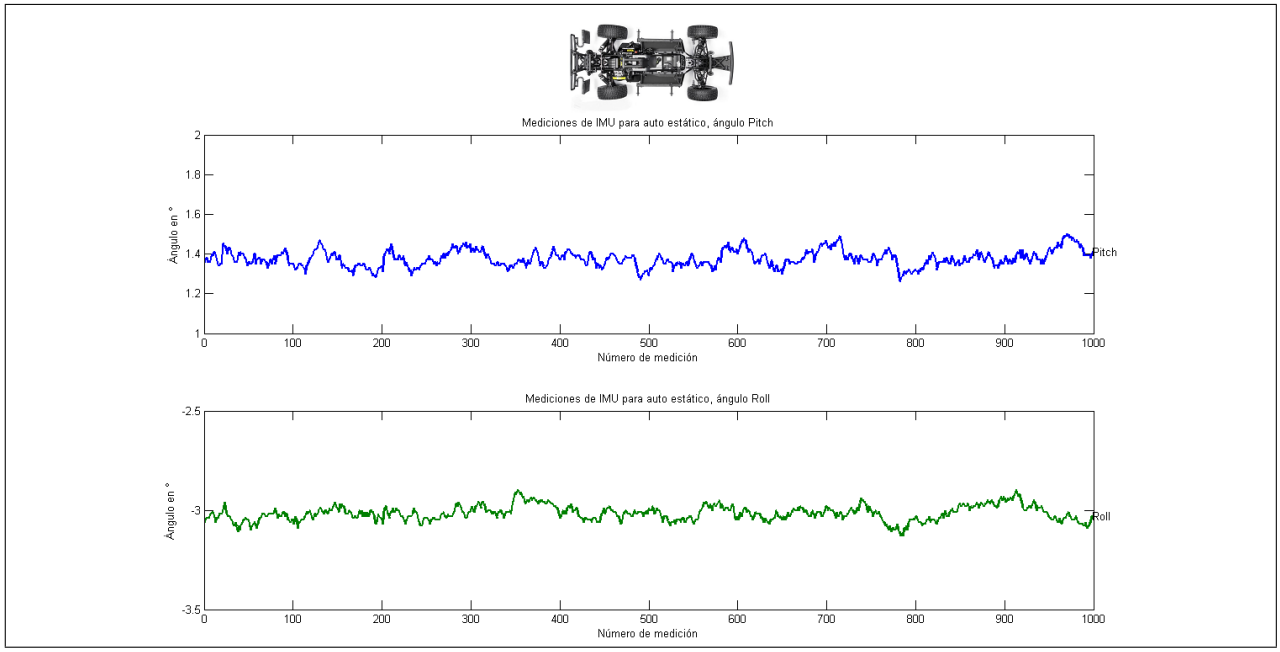

Figura 3.3.2: Gráfica de mediciones de la IMU para auto estático.

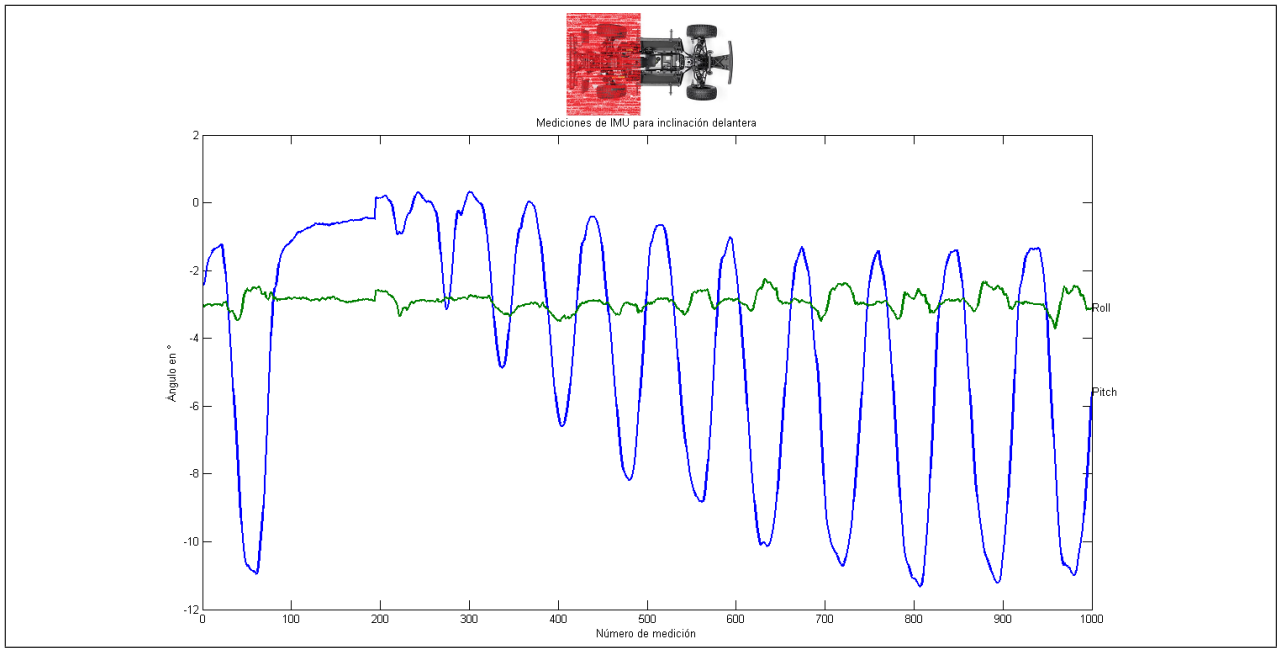

Figura 3.3.3: Gráfica de mediciones de la IMU para inclinación delantera.

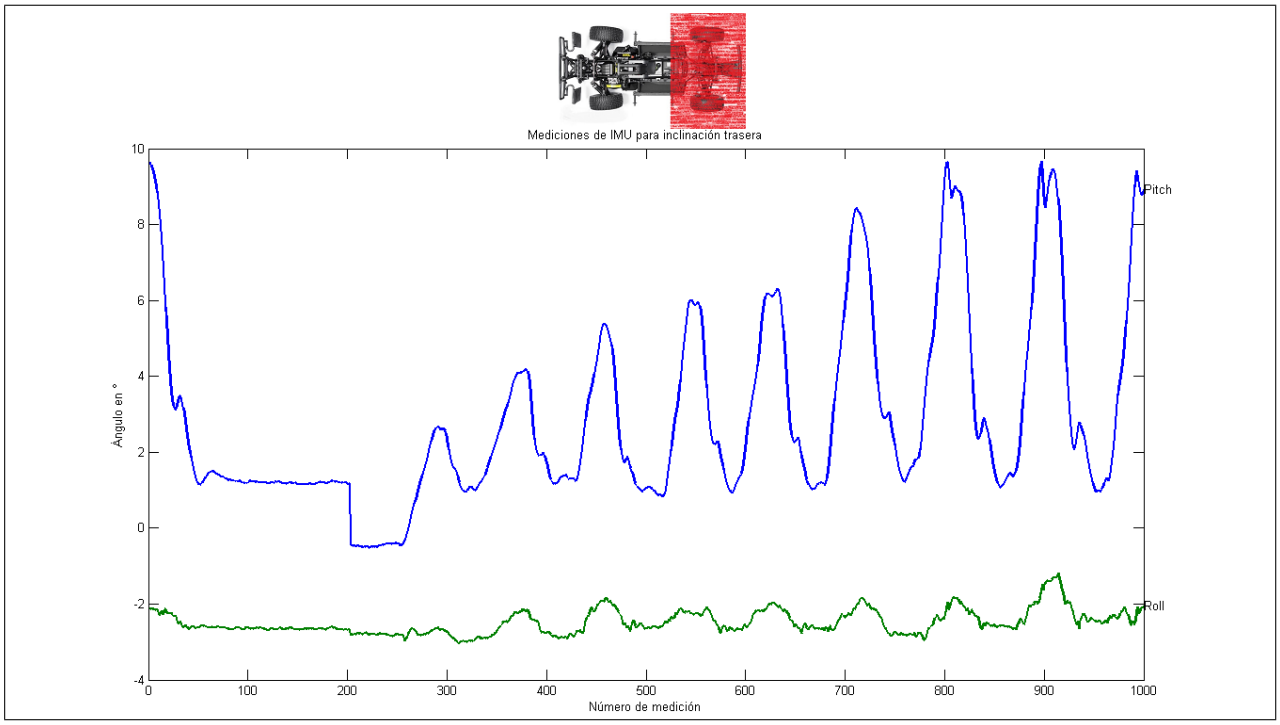

Figura 3.3.4: Gráfica de mediciones de la IMU para inclinación trasera.

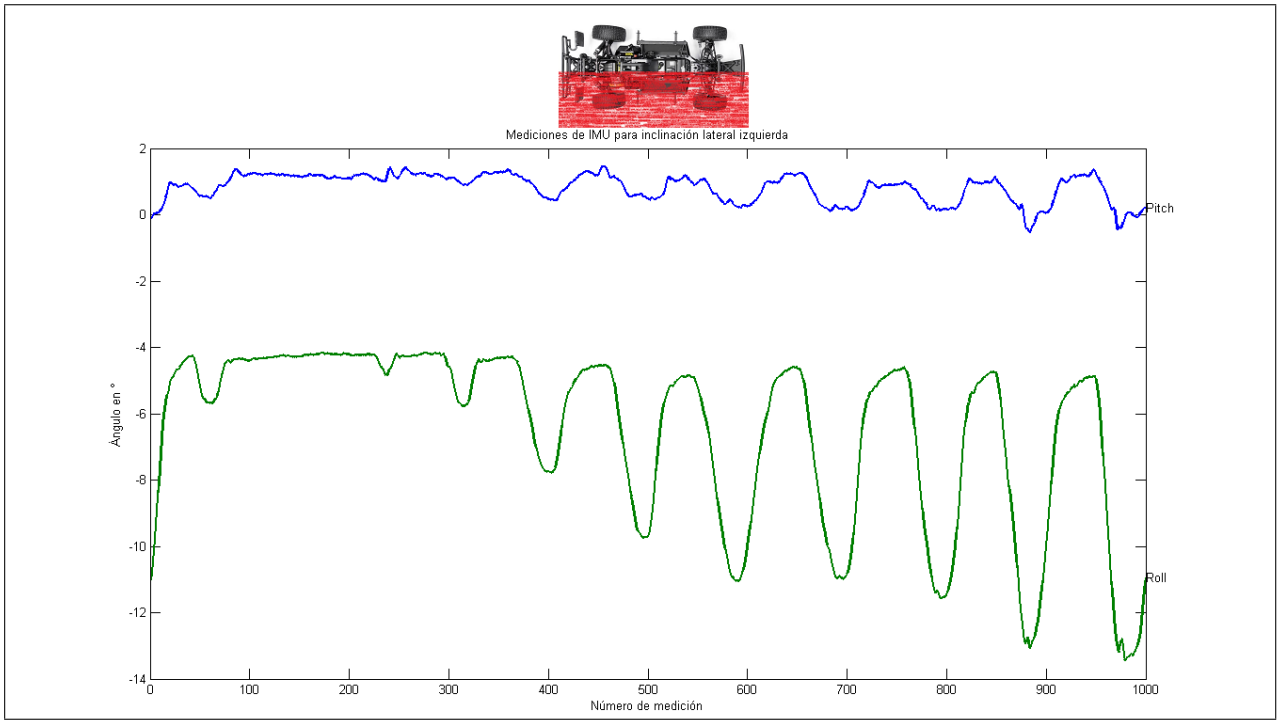

Figura 3.3.5: Gráfica de mediciones de la IMU para inclinación del lado izquierdo.

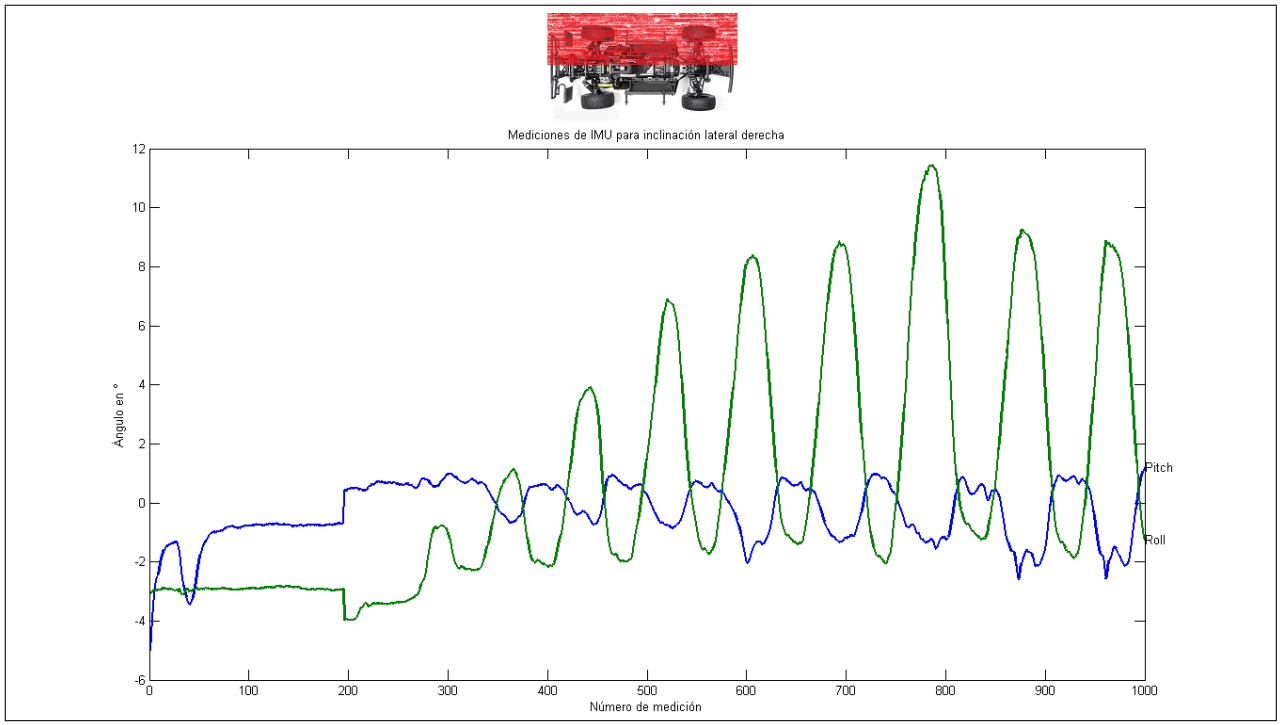

Figura 3.3.6: Gráfica de mediciones de la IMU para inclinación del lado derecho.

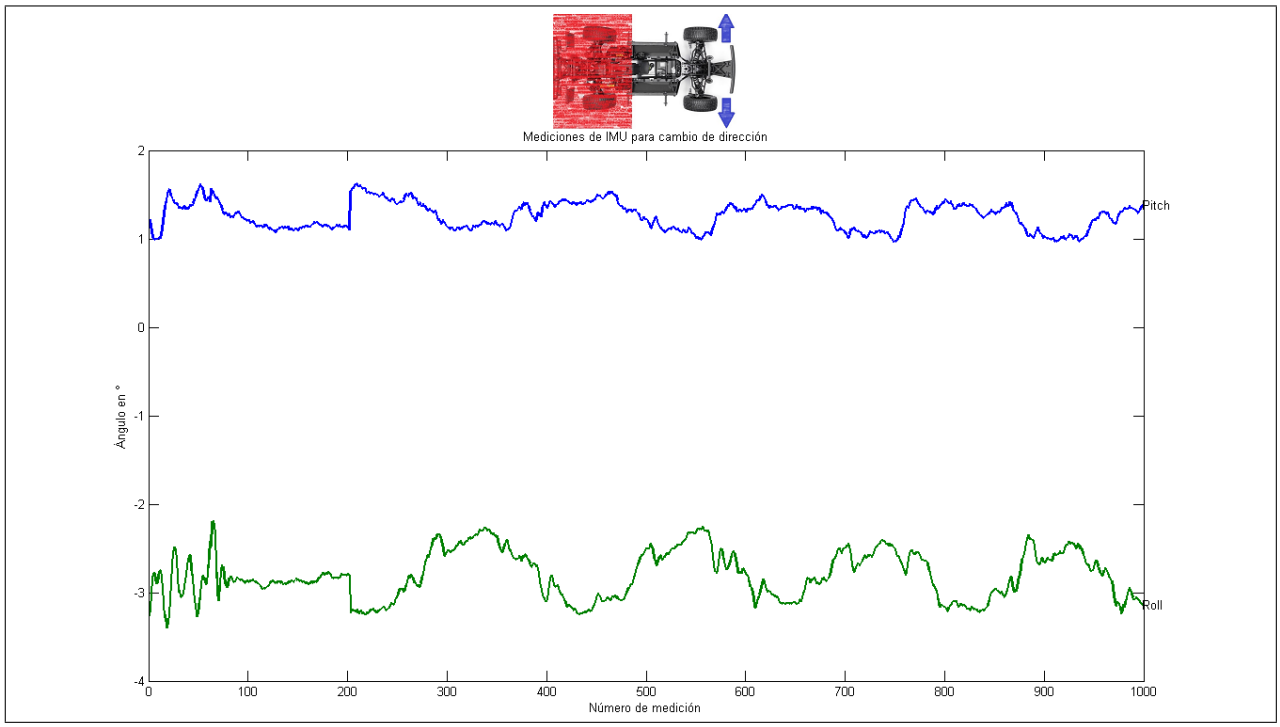

Figura 3.3.7: Gráfica de mediciones de la IMU para cambio en la dirección.

Experimento	**Rango cabeceo**	**Variación**	**Rango alabeo**	**Variación**
Auto estático	[1.26° , 1.50°]	0.24°	[-3.13° , -2.90°]	0.23°
Inclinación delantera	[-11.32° , 0.32°]	11.64°	[-3.72° , -2.25°]	1.47°
Inclinación trasera	[-0.52° , 9.66°]	10.18°	[-3.04° , -1.19°]	1.85°
Inclinación izquierda	[-0.53° , 1.46°]	1.99°	[-13.44° , -4.16°]	9.28°
Inclinación derecha	[-4.99° , 1.13°]	6.12°	[-3.97° , 11.43°]	15.40°
Cambio de dirección	[0.96° , 1.62°]	0.66°	[-3.40° , -2.19°]	1.21°

Tabla 3.3.1: Resultados de experimentos para la IMU.

Los valores de las variaciones obtenidas mediante los experimentos sobre los ángulos de cabeceo y alabeo son resumidos en la Tabla 3.3.1 y se detallan a continuación: En la figura 3.3.2 se muestra la gráfica de las mediciones obtenidas por la IMU para el auto estático, se puede observar que las variaciones en los resultados obtenidos son mínimas en comparación con el rango total de medición: 0.24 grados para el ángulo de cabeceo y 0.23 grados para el ángulo de alabeo. En la figura 3.3.3 se muestra la gráfica de las mediciones de la IMU cuando el auto se inclina hacia adelante, como se espera, el ángulo de cabeceo comienza a variar hasta una diferencia de 11.64 grados entre el valor máximo y el mínimo. En la figura 3.3.4 se muestra la gráfica de las mediciones de la IMU cuando el auto se inclina hacia atrás, se puede observar que en este caso el ángulo de cabeceo varía hasta 10.18 grados entre sus valores máximo y mínimo. Cuando el auto es inclinado hacia la izquierda, la gráfica de mediciones (Figura 3.3.5) muestra una variación en el ángulo de alabeo de hasta 9.28 grados entre sus valores máximo y mínimo. Cuando el auto se inclina hacia la derecha, la gráfica (Figura 3.3.6) muestra una variación en el ángulo de alabeo de hasta 15.4 grados entre sus valores máximo y mínimo. Finalmente, en la Figura 3.3.7 se muestra un experimento adicional en el que se cambia el ángulo en el sistema de dirección, lo cual influye principalmente en el ángulo de guiñada; sin embargo dicho ángulo no será tomado en cuenta para la construcción del Sensor Virtual ya que es calculado a partir de un sistema de referencia global por medio del magnetómetro. El cambio en la dirección causa variaciones de hasta 0.66 grados en el ángulo de cabeceo y 1.21 grados en el ángulo de alabeo.

Con base en los resultados obtenidos se puede concluir que la implementación de la Unidad de Medición Inercial es completamente viable ya que cumple con las variaciones esperadas; esto es, variaciones mínimas durante el experimento de auto estático para evitar ruido, variaciones marcadas en el ángulo de cabeceo cuando el auto se inclina hacia enfrente o hacia atrás y variaciones marcadas en el ángulo de alabeo cuando el auto se inclina de izquierda a derecha. Con respecto al experimento de cambio de dirección se puede concluir que la IMU no será suficiente para distinguir dicho movimiento debido a la pobre variación obtenida, razón por la cual más adelante se trata con un sensor dedicado a medir el cambio en el ángulo de la dirección.

3.3.2. Temporizador

El sistema de dirección del vehículo consta de un emisor RF (control remoto), un receptor RF que decodifica las señales del emisor y genera una señal de PWM que varía de acuerdo al ángulo deseado, un servomotor controlado por la señal de PWM y coples mecánicos que mueven la dirección de las llantas delanteras del vehículo. Así, es posible saber el ángulo de la dirección de las llantas delanteras sabiendo el tiempo en alto de la señal de PWM. Para medir las variaciones en el ángulo del sistema de dirección se utiliza el temporizador de un microcontrolador EKTM4C123GXL-Tiva C Series (Figura 3.3.8) conectado al receptor de radiofrecuencia del sistema de dirección. El microcontrolador contará cuántos ciclos de su temporizador tarda en alto la señal de PWM, cada ciclo es de aproximadamente 2.01 nano segundos.

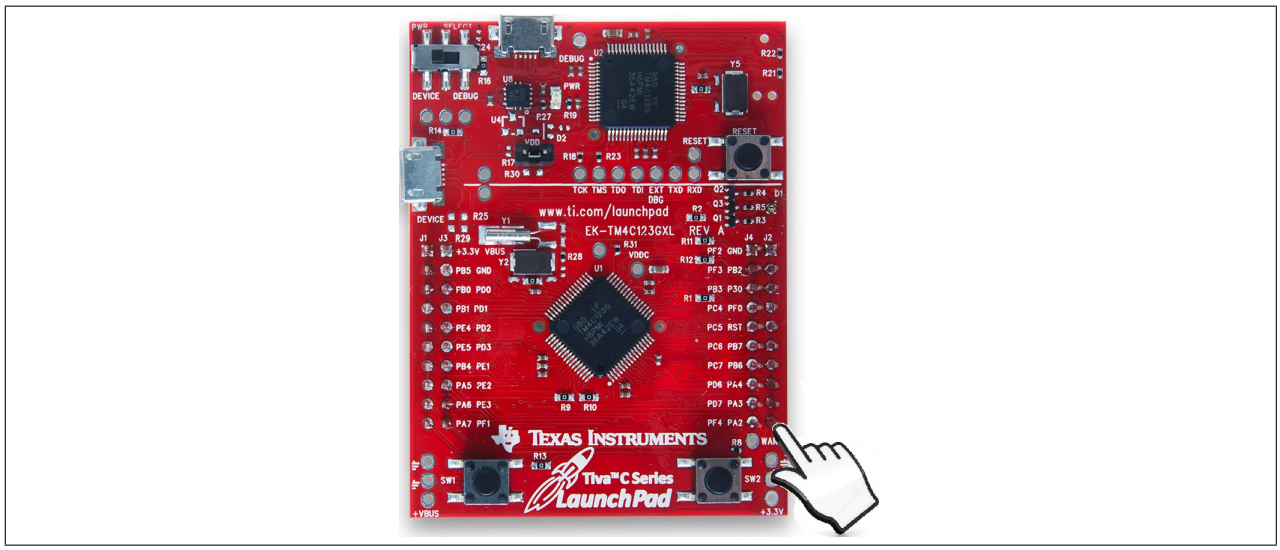

Figura 3.3.8: Pin de conexión de temporizador del microcontrolador.

Para llevar a cabo la conexión entre la señal de PWM y el pin del temporizador solamente es necesario compartir la tierra como señal de referencia y realizar la conexión por medio de un cable en el pin "PA2", el cual es señalado en la figura 3.3.8. Gracias a que la señal de PWM tiene valores lógicos de +3.3 Volts que pueden ser tratados directamente por el microcontrolador, no es necesario hacer ningún tipo de acoplamiento de niveles lógicos.

Para poner en marcha el temporizador es necesario realizar un programa que se encargue de habilitar el pin PA2 y lo declare como pin de temporizador en modo captura, posteriormente se hace una sub rutina que se encarga de generar una interrupción cuando hay un flanco de subida en la señal de PWM, el temporizador comienza a contar ciclos hasta que llega una segunda interrupción generada ahora por el flanco de bajada, el número de ciclos contados representa el tiempo en alto y es guardado en una variable para la presentación de datos. Lo anterior puede ser repetido cuantas veces sea necesario dentro de un bucle o por medio de un

disparador que indique cuando una medición es requerida.

La señal de PWM tiene un periodo de 13.6 mili segundos y el porcentaje de tiempo en alto varía de acuerdo con la señal enviada desde el emisor RF. Con fines ilustrativos, en la Figura 3.3.9 se muestra la señal de PWM medida para tres casos desde un osciloscopio. En la parte superior izquierda (sub figura A) se observan las mediciones cuando el ángulo deseado es totalmente a la izquierda, por la parte inferior central de la imagen (sub figura C) se observa la señal cuando el ángulo deseado es cero (posición de reposo del control remoto), del lado superior derecho (sub figura B) se observa la señal cuando el ángulo deseado está totalmente hacia la derecha. Es importante saber que el periodo (y por lo tanto la frecuencia) es constante, lo único que varía es el porcentaje de dicho periodo que se mantiene la señal en alto. De acuerdo con las mediciones obtenidas se observa que el porcentaje de tiempo en alto en la señal de PWM cuando el ángulo en la dirección deseado es totalmente a la izquierda es de 11.60 % (1.58 ms), cuando el ángulo deseado es cero el tiempo en alto es de 8.81 % (1.20 ms), y si el ángulo deseado es totalmente a la derecha se tiene un tiempo en alto del 6.43 % (0.87 ms). Con lo anterior, se esperan mediciones desde aproximadamente 43283 hasta 78436 ciclos del temporizador con un valor medio de 59750 cuando el emisor RF se encuentra en reposo. Cabe aclarar que cuando se habla del ángulo de la dirección totalmente hacia algún lado (derecha o izquierda) no necesariamente se indican ángulos de $\pm 90°$, sino que se habla del límite mecánico en las fronteras del sistema de medición.

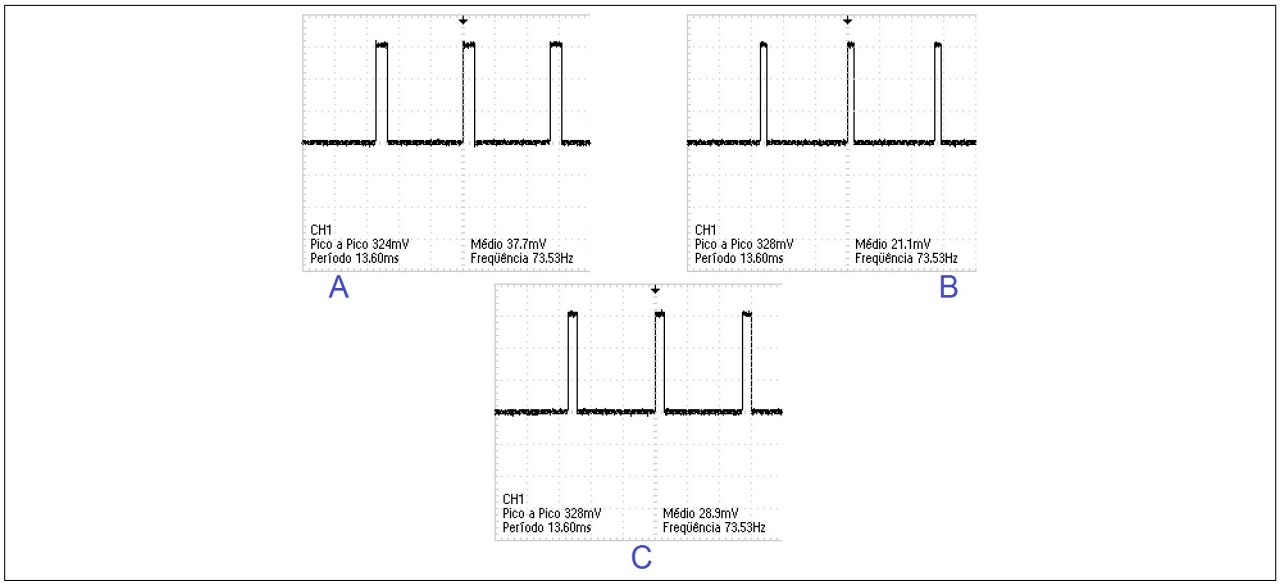

Figura 3.3.9: Señal de PWM medida desde un osciloscopio.

Para mostrar el desempeño del temporizador se realiza un par de experimentos con los cuales se pueda ilustrar el rango de mediciones tomadas por el sensor cuando el vehículo se somete al movimiento que se desea captar con el sensor en cuestión. Un primer experimento muestra el comportamiento de las mediciones cuando el ángulo deseado es cero, dicho de

otra manera, cuando se tiene el vehículo en reposo. En el segundo experimento se comienza con el ángulo deseado en cero y posteriormente se cambia la posición deseada a la izquierda y a la derecha de forma consecutiva y repetitiva hasta que se termina de adquirir datos. En ambos experimentos se toma un total de mil muestras. Las gráficas de las mediciones obtenidas se muestran en las Figuras 3.3.10 para el experimento de auto estático y 3.3.11 para el experimento de cambio en la dirección.

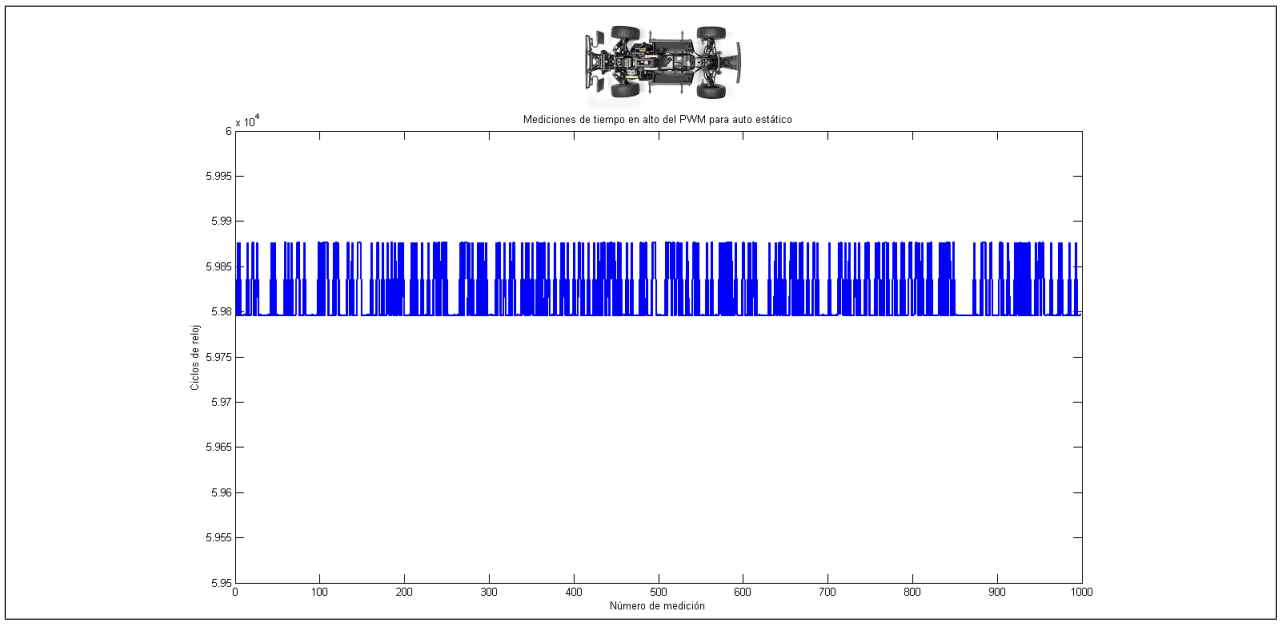

Figura 3.3.10: Gráfica de ciclos del temporizador para auto estático.

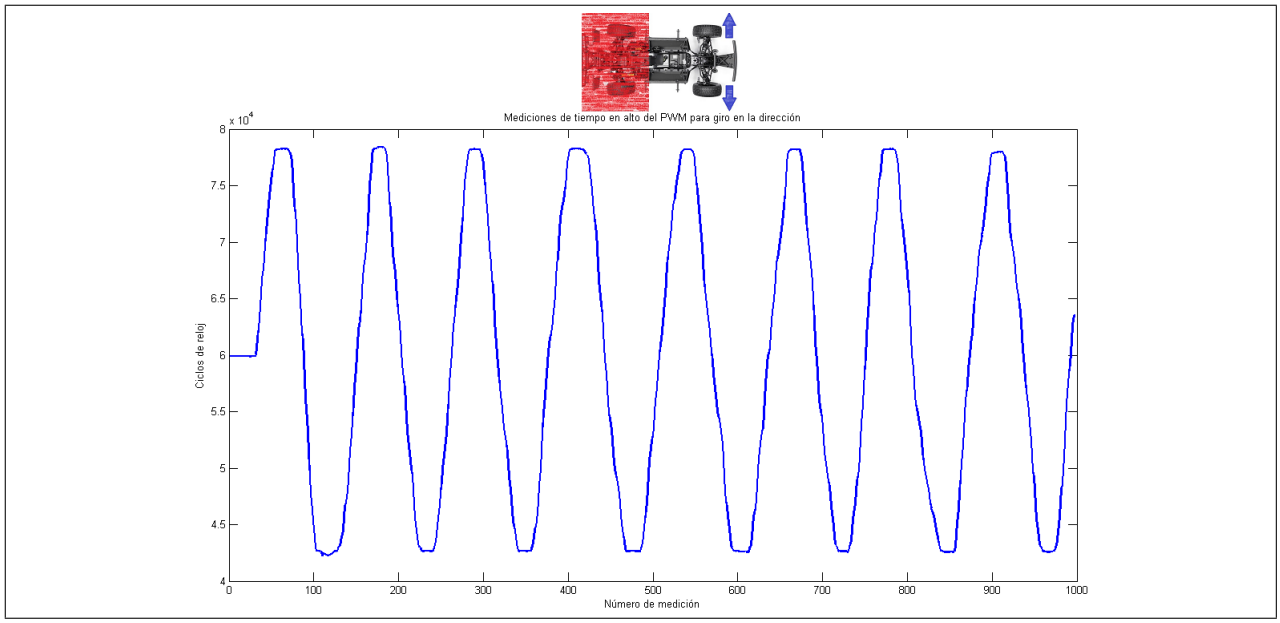

Figura 3.3.11: Gráfica de ciclos del temporizador para cambio en la dirección.

Experimento	**Rango de ciclos**	**Variación en ciclos**
Auto estático	[59797 , 59875]	78
Cambio de dirección	[42239 , 78436]	36197

Tabla 3.3.2: Resultados de experimentos para el temporizador.

Las variaciones obtenidas para la cuenta de ciclos del temporizador para capturar el tiempo en alto del PWM son presentadas en la Tabla 3.3.2 y se explican a continuación: En la Figura 3.3.10 se observa que las variaciones en el experimento de auto estático son pequeñas con respecto a los límites esperados. La señal muestra datos con un valor intermedio de 59836 ciclos con una variación de ± 39 ciclos. En la Figura 3.3.11 se muestra una gráfica con la variación en las mediciones cuando el ángulo en la dirección se somete a cambios de izquierda a derecha. Las mediciones tienen un valor mínimo de 42239 ciclos y un valor máximo de 78436, lo cual resulta en una variación de 36197 ciclos.

Con base en los resultados obtenidos en los experimentos de prueba para captar el tiempo en alto de la señal de PWM, se concluye que es viable implementar el temporizador para captar dicha señal ya que se cumple completamente con lo esperado; es decir, se tiene una mínima presencia de ruido y las variaciones durante el experimento de auto estático son despreciables con respecto a las variaciones marcadas del experimento de cambio en la dirección.

3.3.3. Sensores de Flexión

Un sensor de flexión tiene la capacidad de variar su resistencia cuando es deformado, así será posible captar las variaciones en la geometría del vehículo empleando cuatro sensores de flexión resistivos FS7548 de Sparkfun, uno para cada llanta. Dichos sensores tienen dimensiones 112.24 milímetros de largo, 6.35 milímetros de ancho y 0.7 milímetros de grosor. La longitud activa del sensor es de 95.25 milímetros y tiene un rango de resistencia teórico de $10K\Omega$ a $25K\Omega$ cuando se encuentra en reposo y de $45K\Omega$ a $125K\Omega$ cuando se dobla, dependiendo del radio de doblado. El montaje de los sensores se hace teniendo especial cuidado con la región inactiva del sensor ya que no cuenta con ningún tipo de protección y puede causar fracturas al sensor en el límite con la región activa, para ello se han ideado placas fenólicas y coples de aluminio que servirán para proteger al sensor, comunicar la señal medida y unirlo al auto mientras que se permite movimiento libre para que el sensor sufra deformación cuando cambia la geometría del vehículo. El cableado de cada sensor solamente requiere de dos vías que serán llevadas a una fase de amplificación.

En la Figura 3.3.12 se puede observar una imagen de la parte delantera del vehículo en donde se montan dos sensores de flexión. El montaje comienza con coples de aluminio unidos a las horquillas inferiores de ambos lados, el cople se une a otro cople que aprisiona el sensor de flexión para poder sujetarlo de un extremo, el otro extremo del sensor se encuentra sujeto a una placa fenólica con un trozo de acrílico que aprisiona la zona donde termina la región activa, la placa está diseñada para ser unida a un cople de aluminio y así tener sujetos ambos extremos del sensor al auto. El cableado es de color negro e inicia conectándose en una clema por debajo de la placa, al llegar a la parte central del frente del vehículo se observa una zona de color amarillo donde se unen los cables de ambos sensores y posteriormente son llevados por un cable color gris hacia un amplificador operacional. Los sensores en la parte trasera del vehículo son montados de forma similar.

Para poner en marcha cada sensor de flexión se requiere llevar una etapa de amplificación. En las especificaciones del sensor se recomiendan distintos esquemas de amplificación entre los cuales se encuentra un convertidor de resistencia a voltaje (Figura 3.3.13), esto es, un amplificador inversor con la resistencia variable del sensor de flexión conectada al voltaje de referencia. De igual forma se sugiere el uso de amplificadores operacionales como el LM358 o el LM324, de los cuales se elige el LM324.

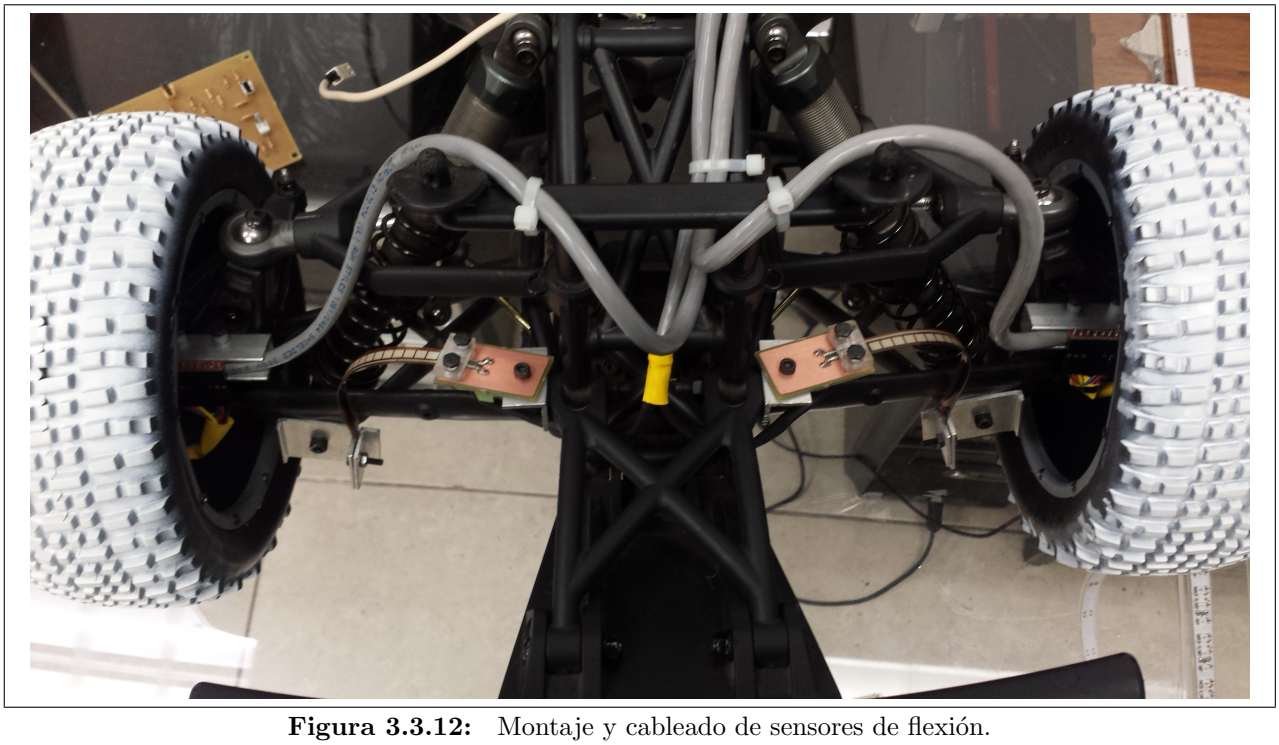

Figura 3.3.12: Montaje y cableado de sensores de flexión.

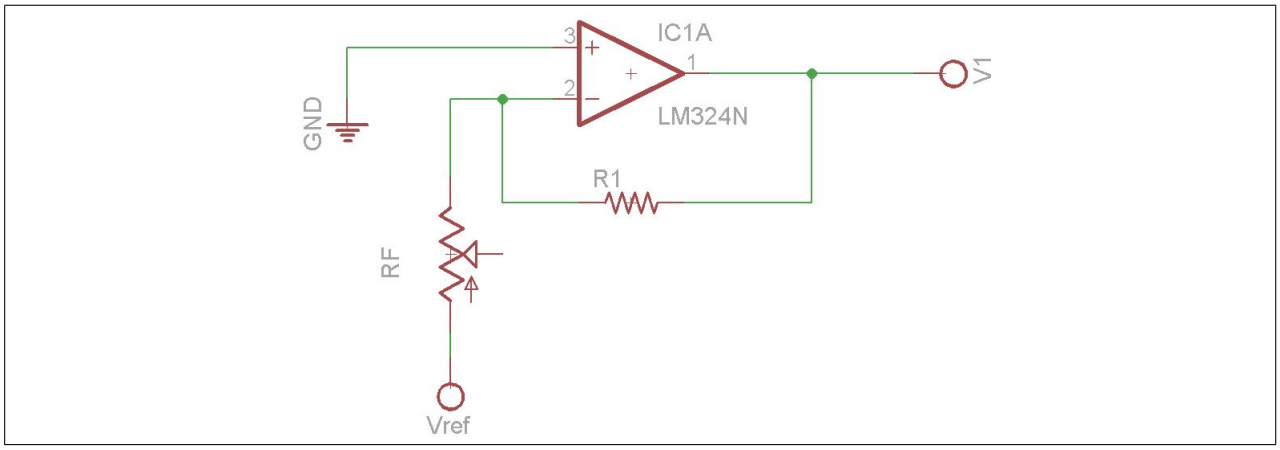

Figura 3.3.13: Propuesta de amplificación en *datasheet* del sensor de flexión.

Para el circuito de la Figura 3.3.13 se tiene una salida dada por:

$$V_1 = V_{ref}\left(\frac{R_1}{R_F}\right) \tag{3.3.1}$$

Donde:
V_{ref} = Voltaje de referencia en el sensor de flexión.
V_1 = Voltaje de salida del amplificador operacional.
R_1 = Resistencia de retroalimentación del amplificador operacional.
R_F = Resistencia variable del sensor de flexión.

Los valores variables de cada sensor de flexión son parámetros dados, mientras que los valores de la resistencia de retroalimentación y del voltaje de referencia son parámetros de diseño que sirven para obtener la salida deseada. En este caso se requiere de una salida que entregue variaciones de 3.3 Volts o menores.

El valor en la resistencia del sensor de flexión de la llanta delantera derecha se encuentra en un rango de $30K\Omega$ a $35K\Omega$ cuando se hace variar la geometría en el vehículo, el rango de la resistencia del sensor de flexión en la llanta delantera izquierda es de $32K\Omega$ a $35K\Omega$, el rango de la resistencia del sensor en la llanta trasera derecha es de $20K\Omega$ a $34K\Omega$ y finalmente el rango de la resistencia del sensor en la llanta trasera izquierda es de $23.5K\Omega$ a $38K\Omega$. Por lo cual se hace una caracterización de los cuatro sensores como resistencias variables de $20K\Omega$ a $38K\Omega$. Realizando operaciones por medio de una hoja de cálculo se llega a una propuesta del voltaje de referencia en 5 Volts y una resistencia de retroalimentación de $27K\Omega$. Con lo anterior se obtienen una variación en el voltaje de salida de 3.2 Volts con un valor máximo de -3.55 Volts y un valor mínimo de -6.75 Volts. Para llevar el voltaje de salida al rango de medición del ADC del microcontrolador (0 a +3.3 Volts) se conecta V_1 a un amplificador sumador inversor, teniendo como resultado el circuito de la figura 3.3.14.

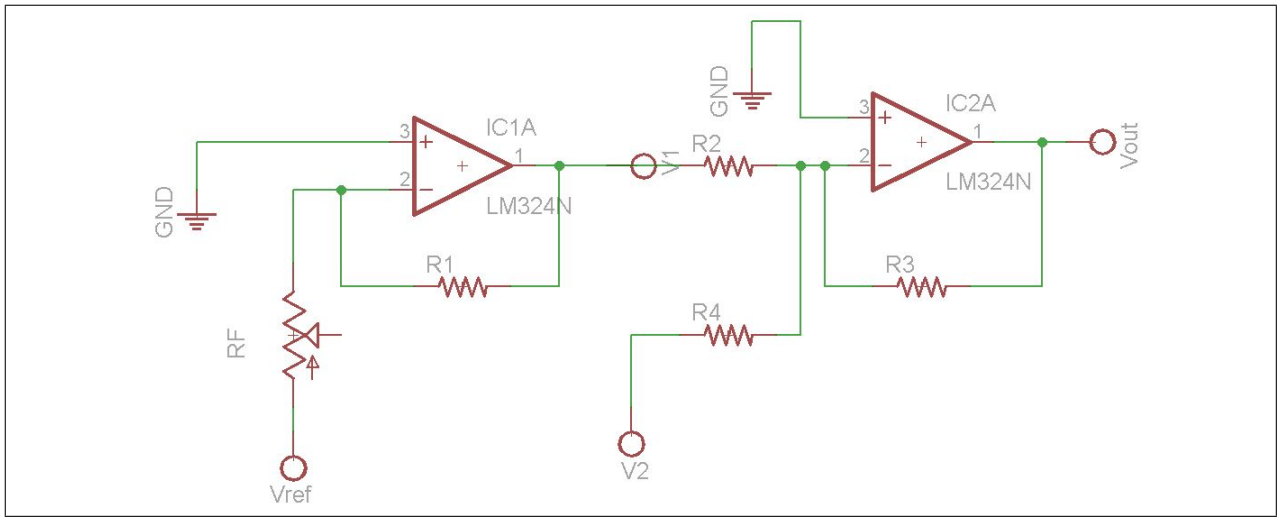

Figura 3.3.14: Amplificador operacional sumador inversor.

El circuito de la Figura 3.3.14 tiene una salida dada por:

$$V_{out} = -R_3 \left(\frac{V_1}{R_2} + \frac{V_2}{R_4} \right) \tag{3.3.2}$$

Los valores de V_2, R_2, R_3 y R_4 son variables de diseño; así que si elegimos $R_2 = R_3 = R_4 = 1K\Omega$, la ecuación 3.3.2 toma ahora la forma de la ecuación 3.3.3.

$$V_{out} = -(V_1 + V_2) \tag{3.3.3}$$

Eligiendo el valor de $V_2 = +3.55$ Volts se tiene como resultado un rango en la salida del amplificador operacional de 0 a +3.2 Volts, lo cual se encuentra dentro del rango del ADC del microcontrolador.

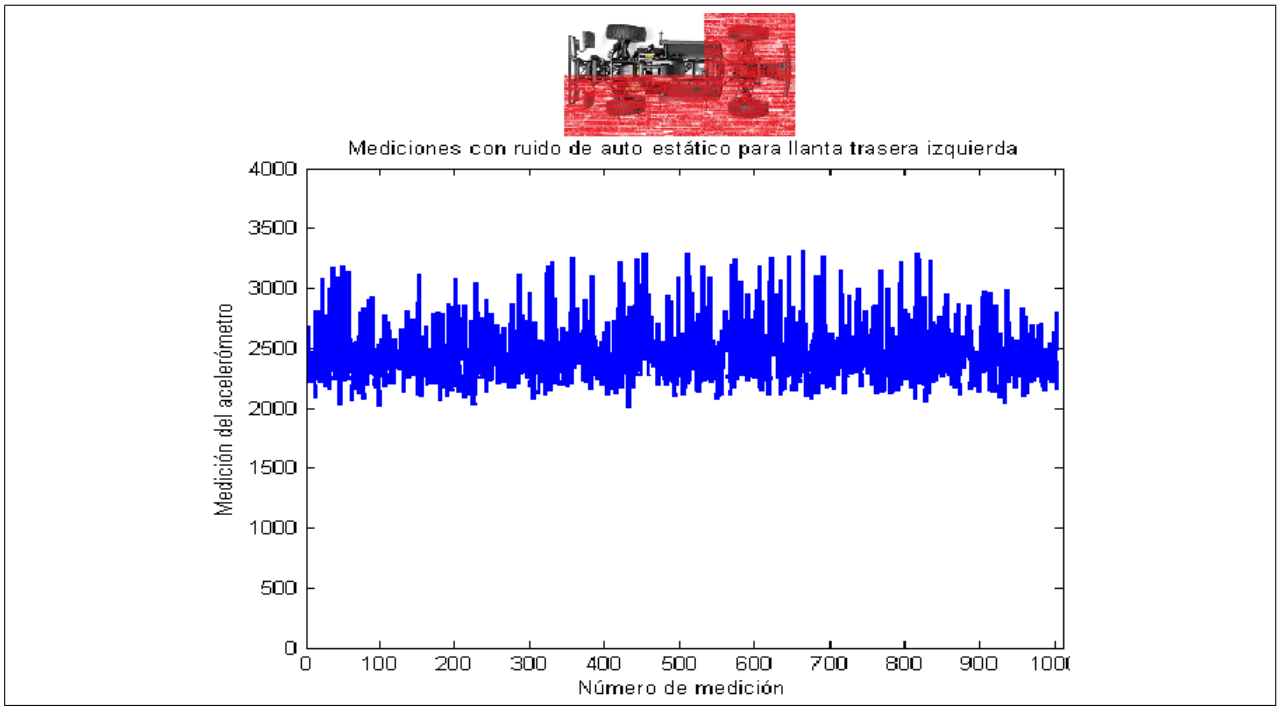

Figura 3.3.15: Gráfica de mediciones del ADC con ruido para auto estático.

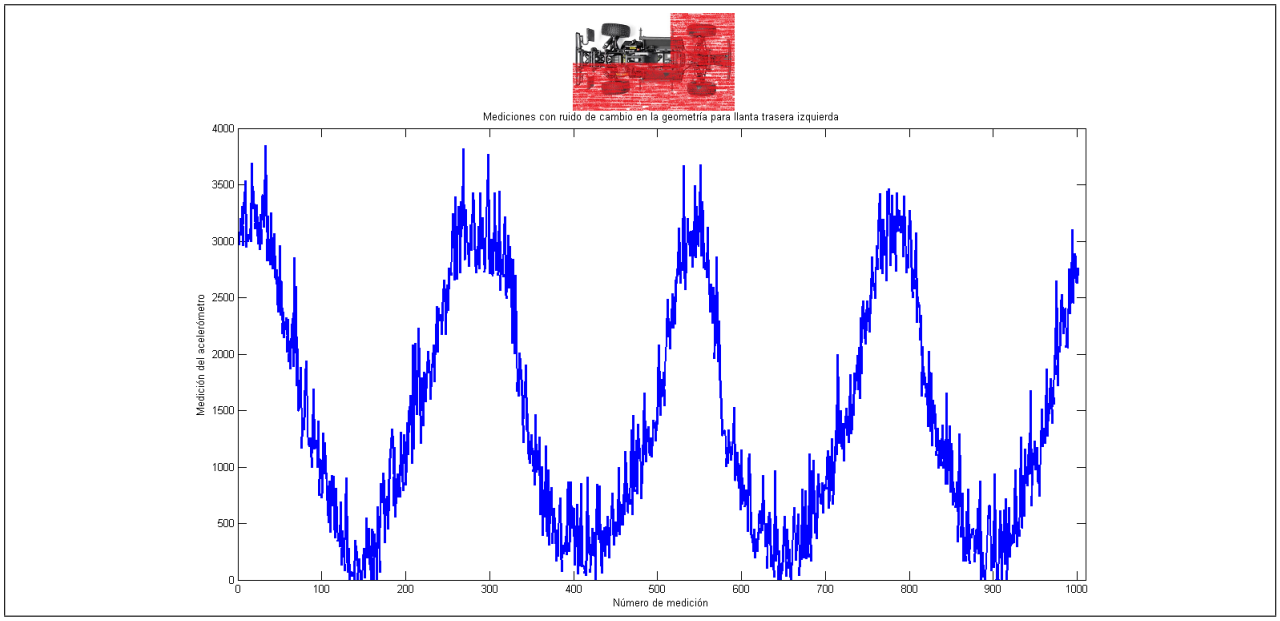

Figura 3.3.16: Gráfica de mediciones del ADC con ruido para auto sometido a deformación en la geometría.

Al tomar mediciones del auto estático y del auto sometido a deformación en la geometría con el circuito de la figura 3.3.14 se obtienen las gráficas de las figuras 3.3.15 para el auto

estático y 3.3.16 para el auto en deformación de la geometría. En ambas gráficas se puede observar la presencia de ruido que afecta de forma significativa las mediciones. Para el experimento de auto estático se tiene una variación de 1305 unidades de medición desde el valor mínimo (2007 unidades) hasta el valor máximo (3312 unidades), tomando en cuenta un ADC de 12 bits y su rango de variación total de 0 a 4096 unidades, se considera que el ruido presente puede causar fallas importantes ya que representa más de una cuarta parte del rango total. Para el experimento de cambio en la geometría se obtienen variaciones de 3849 unidades desde el valor mínimo (0 unidades) hasta el valor máximo (3849 unidades), con lo cual se confirma que la influencia del ruido es considerable, representando más de una tercera parte del rango de variación total del experimento de cambio en la geometría.

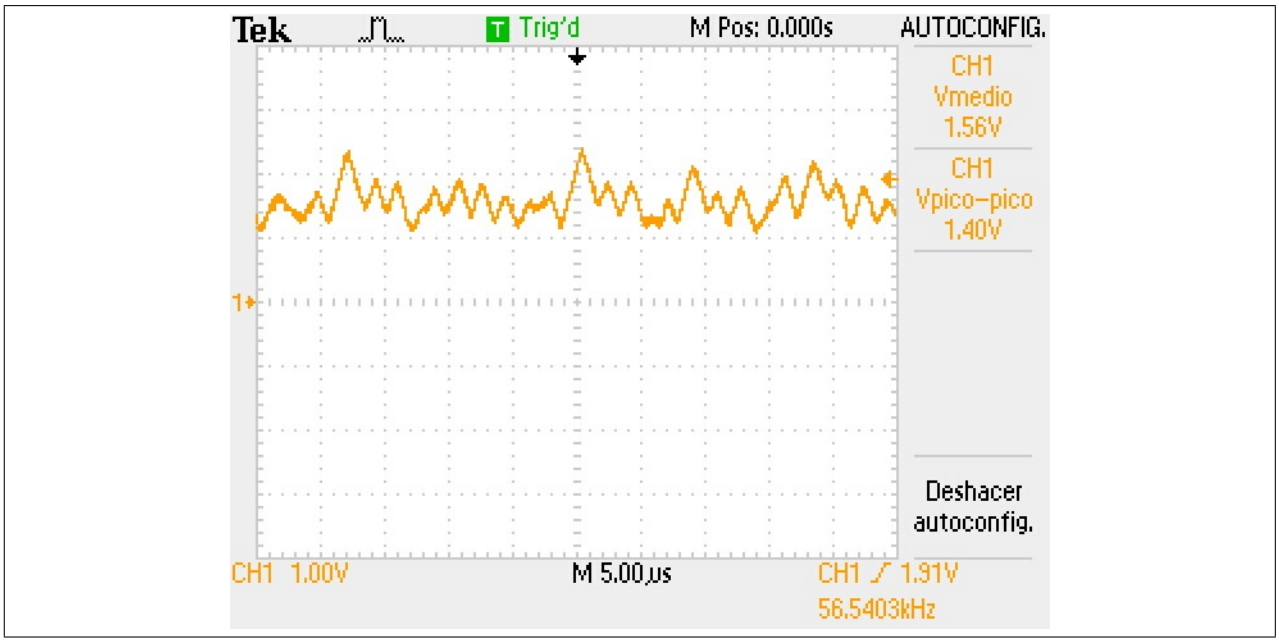

Figura 3.3.17: Medición de la señal de amplificadores operacionales con ruido.

Ya que las gráficas de las figuras 3.3.15 y 3.3.16 presentan ruido aparentemente causado por altas frecuencias, el circuito de la figura 3.3.14 se lleva a observación en osciloscopio para corroborar la frecuencia del ruido. En la figura 3.3.17 se presenta el resultado de la observación del ruido en el osciloscopio, dicha medición indica una frecuencia de 56.54 Kilo Hertz, por lo que se confirma la presencia de ruido de alta frecuencia.

La eliminación de ruido de alta frecuencia puede hacerse de distinas maneras. Una forma simple de hacerlo es por medio de un filtro pasa bajas compuesto por una resistencia conectada en serie con la señal a medir y un capacitor conectado en paralelo. La frecuencia de corte de dicho filtro pasa bajas está dada por la ecuación 3.3.4

$$F_c = \frac{1}{2\pi RC} \tag{3.3.4}$$

Donde:
F_c = Frecuencia de corte.
R = Valor de la resistencia en el filtro.
C = Valor del capacitor en el filtro.

Si se requiere de un filtro en el que la frecuencia de corte sea menor a 20 Hz y se propone un capacitor de 1μF, se obtiene como requisito una resistencia mayor a $7.9K\Omega$. Ya que las resistencias comerciales cercanas pero superiores al valor requerido son de $8.2K\Omega$ y $10K\Omega$, se elige la resistencia de $10K\Omega$. El resultado final de la frecuencia de corte será de aproximadamente 16 Hz. En la figura 3.3.18 se muestra el circuito de amplificadores operacionales con filtro pasa bajas a la salida.

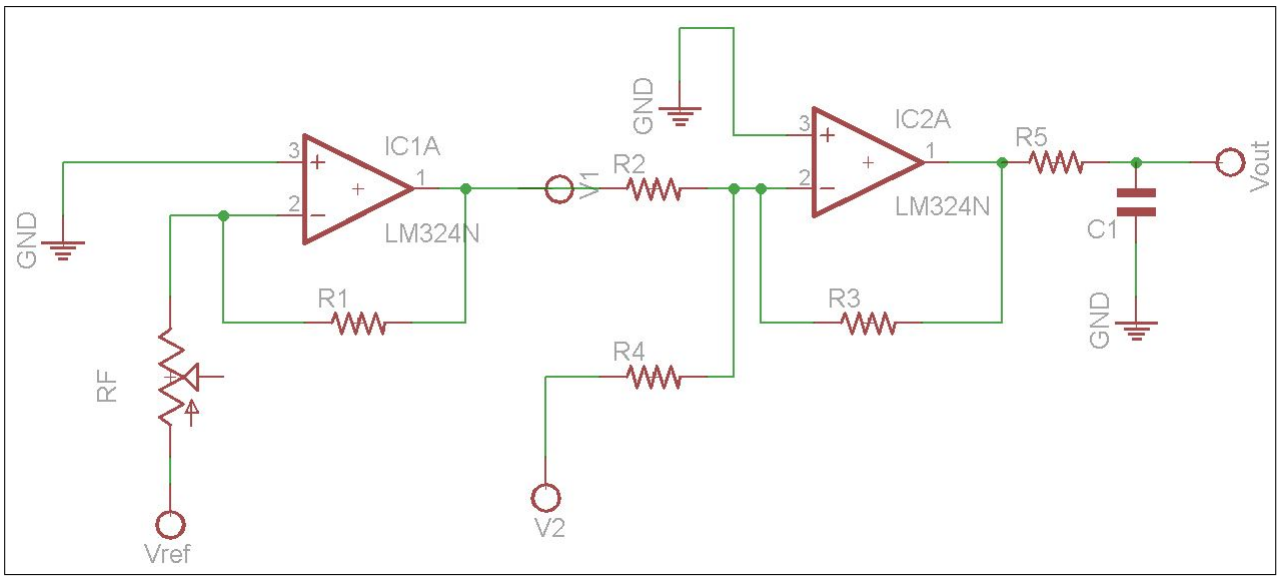

Figura 3.3.18: Amplificador operacional sumador inversor con filtro pasa bajas.

Después de aplicado el filtro se obtienen mediciones de mejor calidad, lo cual se puede observar comparando las figuras 3.3.15 y 3.3.16 (mediciones tomadas con ruido) contra las figuras 3.3.25 y 3.3.26 (mediciones tomadas con filtro). Con el fin de evaluar el desempeño del circuito en la figura 3.3.18 y del ADC del microcontrolador cuando el auto se somete a los cambios en la geometría que se pretende captar, se llevan a cabo experimentos de medición de los cuatro sensores para auto estático y para deformación en la geometría. Los experimentos son divididos por llanta, para cada llanta se realizan dos experimentos: el primer experimento de auto estático consiste en mantener el auto encendido pero sin avanzar ni girar la dirección, y el segundo experimento de deformación en la geometría consiste en aplicar peso de forma manual en la zona de la llanta en cuestión para lograr deformar la geometría del vehículo del lado en que se encuentra el sensor. Para todos los experimentos se toma un total de mil muestras. Las gráficas resultantes se muestran a continuación en las Figuras 3.3.19 a la 3.3.26.

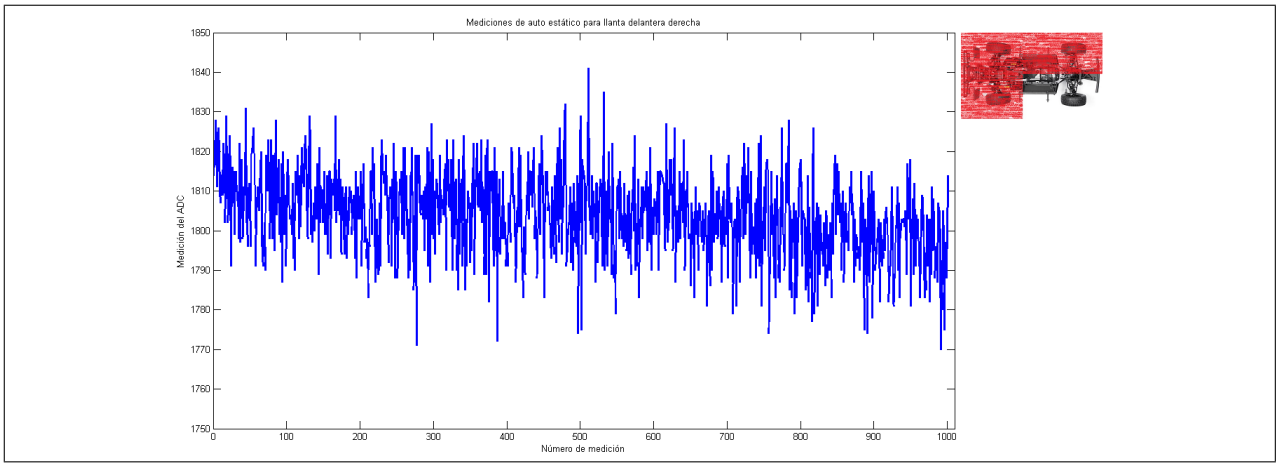

Figura 3.3.19: Gráfica de mediciones del ADC para auto estático, parte delantera derecha.

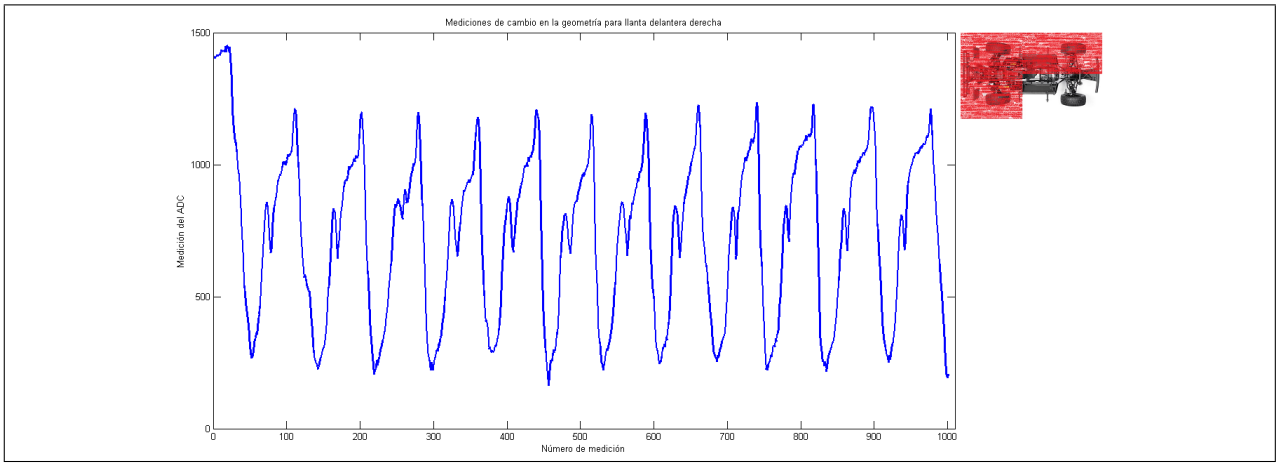

Figura 3.3.20: Gráfica de mediciones del ADC para deformación en la geometría, parte delantera derecha.

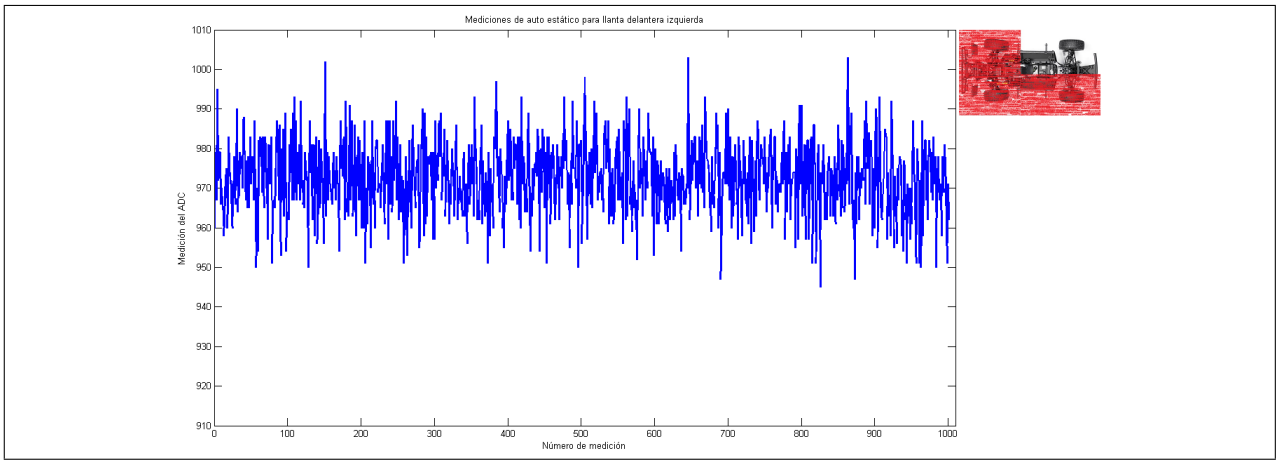

Figura 3.3.21: Gráfica de mediciones del ADC para auto estático, parte delantera izquierda.

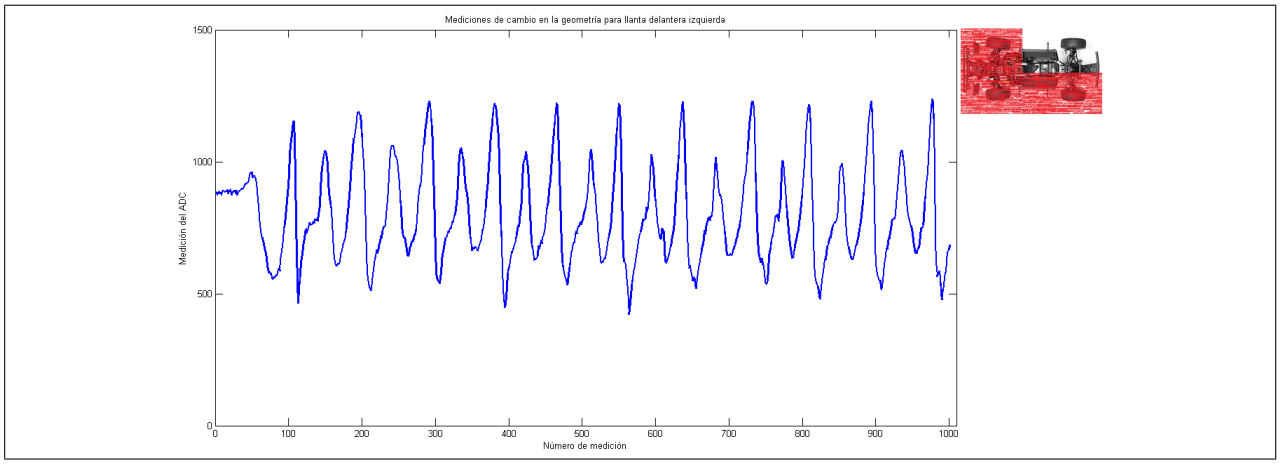

Figura 3.3.22: Gráfica de mediciones del ADC para deformación en la geometría, parte delantera izquierda.

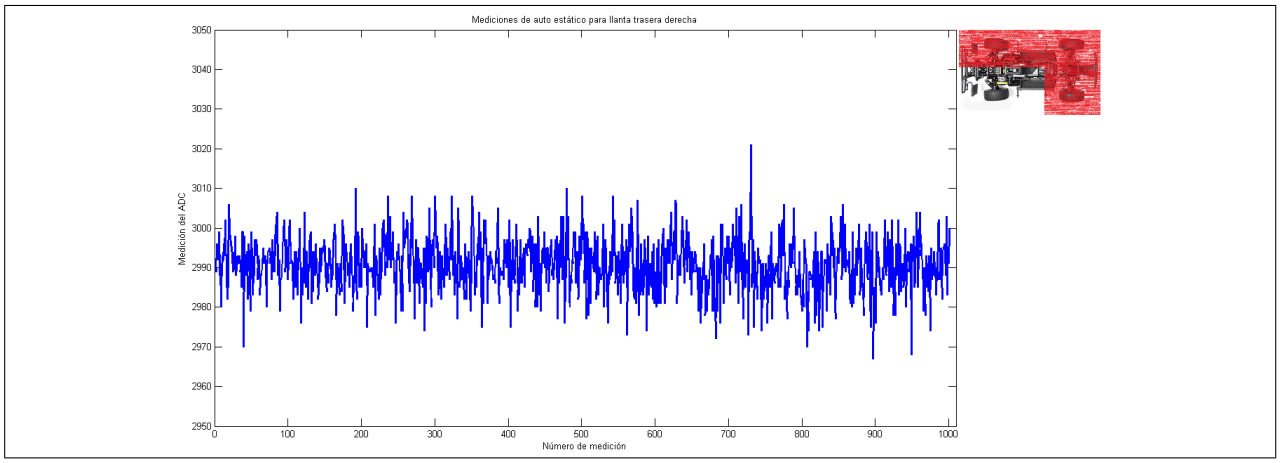

Figura 3.3.23: Gráfica de mediciones del ADC para auto estático, parte trasera derecha.

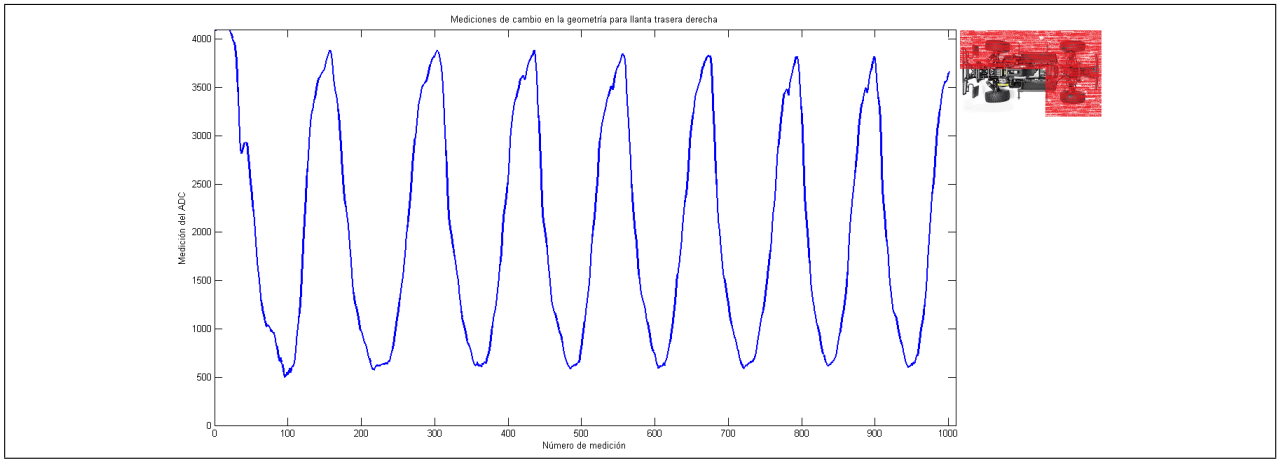

Figura 3.3.24: Gráfica de mediciones del ADC para deformación en la geometría, parte trasera derecha.

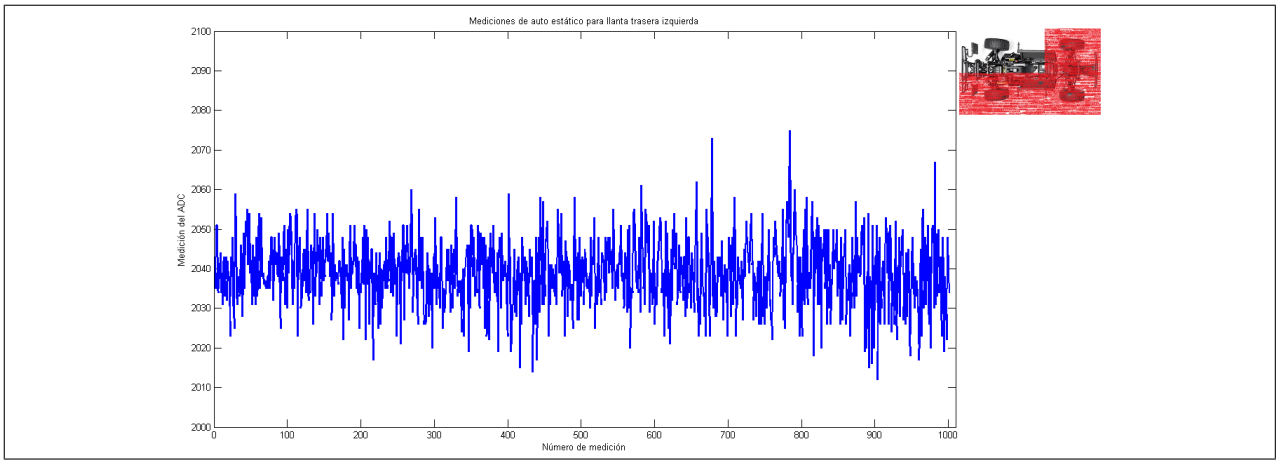

Figura 3.3.25: Gráfica de mediciones del ADC para auto estático, parte trasera izquierda.

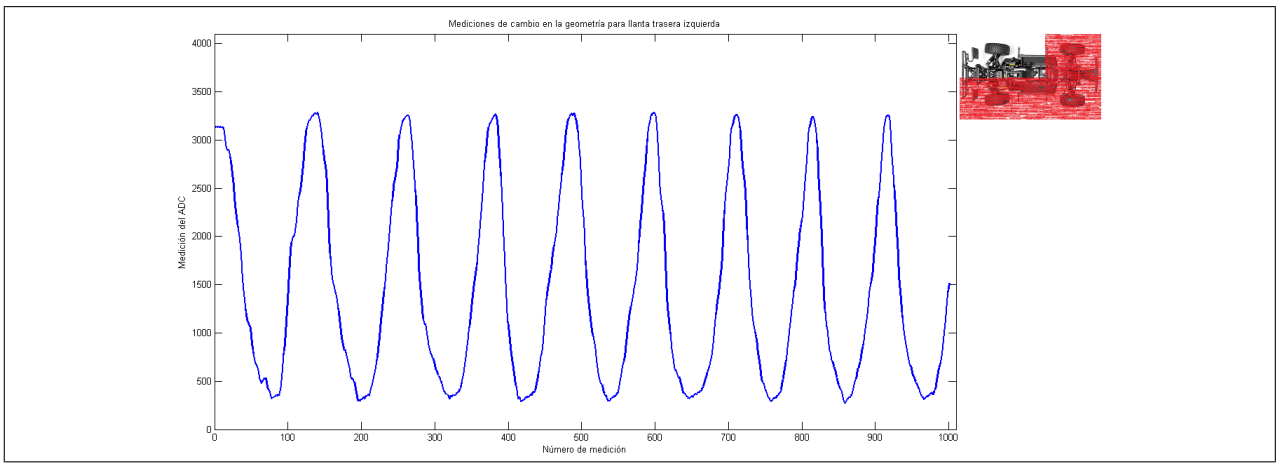

Figura 3.3.26: Gráfica de mediciones del ADC para deformación en la geometría, parte trasera izquierda.

De las gráficas de experimentos de auto estático y de cambio en la geometría (Figuras 3.3.19 a 3.3.26) se puede concluir que a pesar de estar utilizando el mismo tipo de sensor, la trayectoria del movimiento que sigue cada sensor influye a la salida. También se observa la atenuación de ruido de alta frecuencia hasta un promedio de 61.5 unidades, lo cual es más de 20 veces menor que el ruido que se tenía anteriormente (1305 unidades) sin el filtro y hasta 66 veces menor que el rango de medición completo (4096 unidades).

Los resultados obtenidos durante los experimentos de prueba para los sensores de flexión son resumidos en la tabla 3.3.3. En la Figura 3.3.19 se observa la gráfica de los resultados obtenidos para el experimento de auto estático en la parte delantera derecha, para este caso se obtuvieron variaciones desde 1770 unidades hasta 1841 unidades, brindando un rango total de 71 unidades. En la Figura 3.3.20 se tienen los resultados del experimento de cambio en la geometría para la parte delantera derecha, se obtienen variaciones desde 164 hasta 1452

unidades, con un rango total de hasta 1288 unidades. En la Figura 3.3.21 se tiene la gráfica de resultados para el experimento de auto estático por la parte delantera izquierda, se obtienen variaciones de 945 a 1003 unidades con un rango de 58. En la figura 3.3.22 se observa la gráfica de resultados para el experimento de cambios en la geometría por la parte delantera izquierda, se obtienen valores desde 422 hasta 1239 unidades con un rango de hasta 817 unidades. En la Figura 3.3.23 se observa la gráfica de resultados obtenidos para el experimento de auto estático para la parte trasera derecha, se obtienen valores desde 2967 hasta 3021 unidades con un rango de 54. En la Figura 3.3.24 se observa la gráfica de resultados para el experimento de cambios en la geometría por la parte trasera derecha, en este caso se obtienen valores de 499 a 4095 unidades, obteniendo un rango de 3596 unidades. En la Figura 3.3.25 se tiene la gráfica del experimento de auto estático para la parte trasera izquierda, con variaciones de hasta 63 unidades con un valor mínimo de 2012 y un máximo de 2075. Finalmente, en la Figura 3.3.26 se muestra la gráfica con los resultados del experimento de cambio en la geometría del auto por la parte trasera izquierda, en ella se observan variaciones de hasta 3014 unidades con un valor mínimo de 275 y un valor máximo de 3289 unidades.

Experimento	**Cuadrante**	**Val mínimo**	**Val máximo**	**Variación**
Auto estático	Delantero derecho	1770	1841	71
Auto estático	Delantero izquierdo	945	1003	58
Auto estático	Trasero derecho	2967	3021	54
Auto estático	Trasero izquierdo	2012	2075	63
Deformación	Delantero derecho	164	1452	1288
Deformación	Delantero izquierdo	422	1239	817
Deformación	Trasero derecho	499	4095	3596
Deformación	Trasero izquierdo	275	3289	3014

Tabla 3.3.3: Resultados de experimentos para sensores de flexión.

Con base en los resultados encontrados durante los experimentos posteriores a la aplicación del filtro pasa-bajas, se puede garantizar la calidad de las mediciones obtenidas por los sensores de flexión gracias a que se obtienen variaciones significativas y sin ruido cuando el auto se somete a deformación. Con respecto a los experimentos de auto estático, las variaciones son pequeñas y se deben principalmente a la deformación en la geometría causada por los movimientos del motor.

3.3.4. Acelerómetros

Para captar las variaciones en la aceleración con respecto al eje z de la masa no suspendida en cada lado del vehículo se contempla el uso del acelerómetro ADXL345 de Sparkfun. El ADXL345 es un acelerómetro digital de tres ejes con características importantes, entre las cuales se encuentran: 13 bits de resolución, rangos de medición seleccionables por el usuario de ±2 g / ±4 g / ±8 g / ±16 g, protocolos de comunicación I^2C y SPI de 3 y 4 cables, 2 pines de interrupción, resolución completa cuando se incrementa el rango hasta ±16 g manteniendo un factor de escala de 4mg/LSB en todos los rangos, ancho de banda elegible mediante comandos, dimensiones de 3 x 5 x 1 mm.

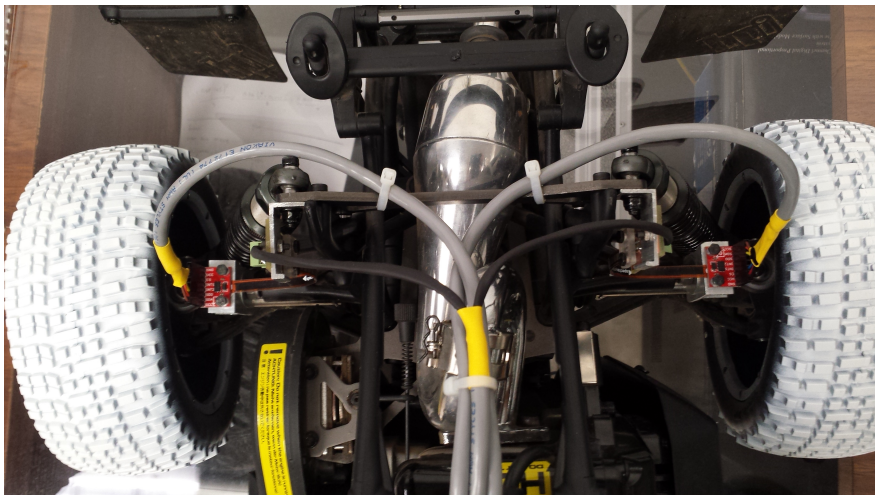

Figura 3.3.27: Montaje y cableado de acelerómetros.

Para el montaje de los acelerómetros (Figura 3.3.27) se utilizan coples maquinados en aluminio diseñados especialmente para unir cada sensor a algún elemento de la masa no suspendida del vehículo, específicamente de la rótula de cada llanta. El cableado de cada sensor será de 6 vías, despreciando 2 de las 8 vías del sensor que son dedicadas a interrupciones, el cable de 6 vías es de doble blindaje con el fin de aislar las señales de ruido causado por calor o por otros factores varios. Es importante mencionar que el cableado de los acelerómetros debe tener holgura en sus distancias, especialmente la sección que cruza de la masa no suspendida a la masa suspendida ya que en dicha zona se presentan los movimientos causados por la suspensión del vehículo. En la figura 3.3.27 se puede observar el cableado en color gris de los acelerómetros de la parte trasera del vehículo. Los 6 pines de conexión de cada acelerómetro son: Tierra (GND), Voltaje (VCC), Selección de Chip (CS), Pin de salida (SDO), Pin de entrada (SDA) y Señal de reloj (SCL) y la comunicación a utilizar será SPI de 4 cables. El montaje y cableado de los aclerómetros en la parte delantera se realiza de forma similar. Los niveles lógicos que utiliza este sensor pueden ser desde 1.7 Volts hasta el Voltaje de alimentación, así que el sensor se alimentará con un voltaje de +3.3 Volts para evitar problemas de compatibilidad en los niveles lógicos del sensor cuando se comunica con el microcontrolador.

Para poner en marcha cada acelerómetro, es necesario un dispositivo que controle la comunicación SPI y opere como bus maestro, en este caso será el microcontrolador EKTM4C123GXL-Tiva C Series.

El primer paso es la configuración del acelerómetro, esto se da por medio de 58 registros de los cuales se muestran los más importantes para los fines del presente trabajo en la tabla 3.3.4 y serán tratados más adelante, posterior a la explicación del esquema de comunicación del sensor. La velocidad máxima soportada del reloj SPI es de 5MHz con 100 pF de carga máxima, el esquema de tiempos sigue una polaridad de reloj (CPOL)=1 y una fase de reloj (CPHA)=1.

Como se indica en el *datasheet* del ADXL345, $\overline{CS}$ es la línea de habilitación del chip y debe ser controlada por el maestro SPI, esta línea debe estar en bajo lógico al inicio de cada transmisión y ser llevada a alto lógico al final, de la manera en que se muestra en las Figuras 3.3.28 y 3.3.29. SCK es el reloj del puerto serial y debe ser generado por el maestro SPI. Mientras $\overline{CS}$ se encuentra en alto, cuando se está en una etapa de no transmisión, SCK se mantiene en alto. SDI y SDO son los datos de entrada y salida serial, respectivamente. Cada dato se va transmitiendo en el flanco de subida del reloj (SCK). Para poder leer o transmitir múltiples datos en una sola transmisión, el bit múltiple (MB) localizado después del bit de lectura/escritura (R/$\overline{W}$) del primer byte transmitido debe ser puesto en alto (MB en figuras 3.3.28 y 3.3.29). De este modo, después de la dirección de registro y del primer byte de datos, cada conjunto de 8 pulsos de reloj causa que el sensor vaya al siguiente registro para su lectura o escritura. Estos cambios de registro continúan hasta que el reloj deje de dar pulsos y $\overline{CS}$ esté en alto. Para realizar lecturas o escrituras en diferentes registros que no son subsecuentes $\overline{CS}$ debe ser puesto en alto entre transmisiones y la dirección de cada registro será escrita cada vez. El diagrama de tiempos para comunicación SPI de cuatro cables para escritura y lectura proporcionados en el *datasheet* es mostrado en las Figuras 3.3.28 y 3.3.29, respectivamente.

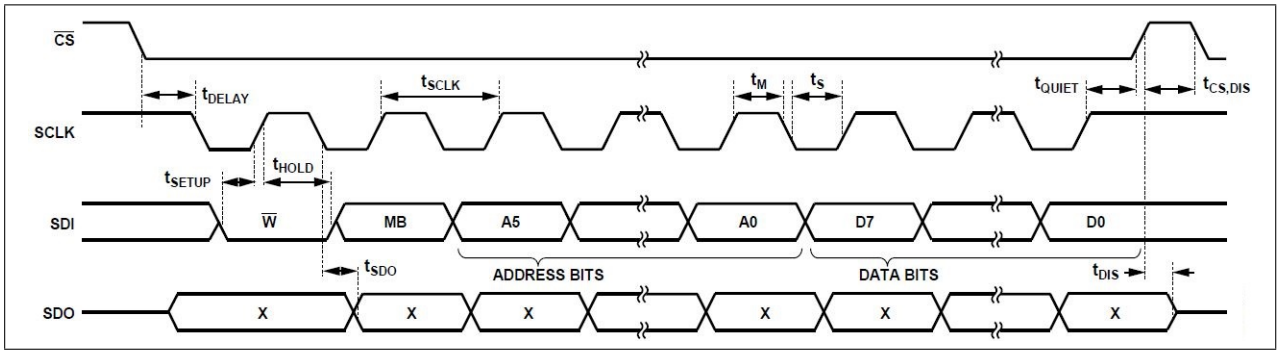

Figura 3.3.28: Diagrama de tiempos para escritura en SPI de 4 cables.

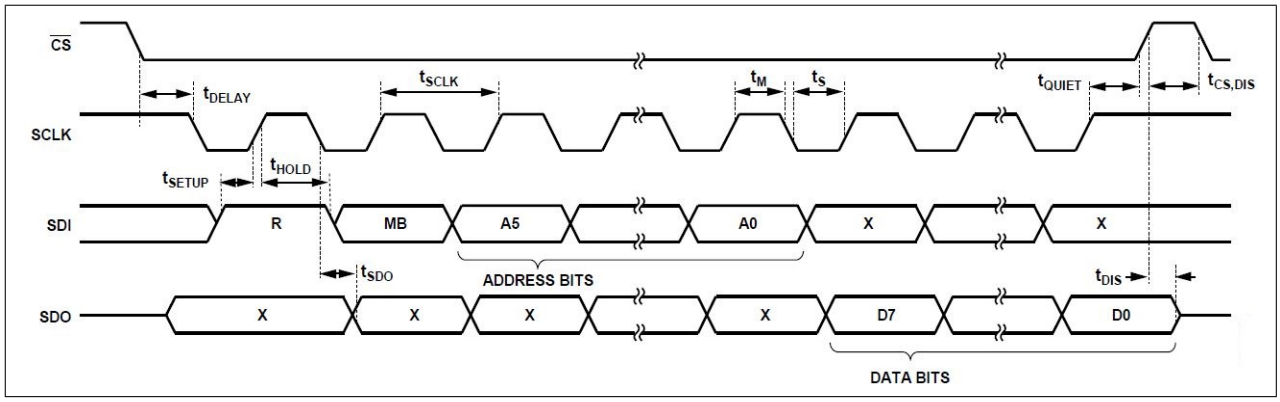

Figura 3.3.29: Diagrama de tiempos para lectura en SPI de 4 cables.

Hex	Dec	Nombre	Tipo	Valor por defecto
0x2C	44	BW_RATE	R/W	0000 1010
0x2D	45	POWER_CTL	R/W	0000 0000
0x31	49	DATA_FORMAT	R/W	0000 0000
0x32	50	DATAX0	R	0000 0000
0x33	51	DATAX1	R	0000 0000
0x34	52	DATAY0	R	0000 0000
0x35	53	DATAY1	R	0000 0000
0x36	54	DATAZ0	R	0000 0000
0x37	55	DATAZ1	R	0000 0000

Tabla 3.3.4: Mapa de registros del ADXL345.

Registro 0x2C - BW_RATE

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	low power	rate			

Tabla 3.3.5: Registro 0x2C.

En la tabla 3.3.5 se muestra la composición del registro 0x2C. Los bits **D7** al **D5** son siempre ceros. El bit **D4** es llamado de baja potencia, poniéndole en un valor de 0 se mantiene al acelerómetro en operación normal y poniéndole un valor de 1 se selecciona reducción de potencia, con lo cual se reduce el consumo de energía; sin embargo se puede obtener una

mayor cantidad de ruido por lo cual se elige un valor 0 para este bit. Los bits **D3** al **D0** son los bits de ancho de banda y se declaran con base al ancho de banda deseado y de acuerdo al *datasheet* de la forma que se muestra en la tabla 3.3.6.

Ancho de banda (Hz)	**Código**
3200	1111
1600	1110
800	1101
400	1100
200	1011
100	1010
50	1001
25	1000
12.5	0111
6.25	0110

Tabla 3.3.6: Ancho de banda captado por el acelerómetro.

Registro 0x2D - POWER_CTL

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	link	auto sleep	measure	sleep	wakeup	

Tabla 3.3.7: Registro 0x2D.

En la tabla 3.3.7 se muestra la composición del registro 0x2D. Los bits **D7** y **D6** son siempre 0. El bit **D5** es utilizado para ligar las funciones de actividad e inactividad del acelerómetro, cuando es puesto a 1 junto con la habilitación de las funciones de detección de actividad/inactividad, se activa un retraso al inicio de cada actividad mientras se detecta inactividad, poniendo el bit a 0 se obtienen las funciones de actividad/inactividad de forma concurrente. El bit **D4** es llamado *auto sleep*, poniendo este bit a 1 se lleva al acelerómetro a modo *sleep* de forma automática cuando se detecta un periodo de inactividad, poniendo este bit a 0 se inhabilita dicho cambio automático. El bit **D3** es el bit de disparo de medición, poniéndolo a 0 se lleva al acelerómetro a modo de espera y llevándolo a 1 se obtienen mediciones de forma continua. El bit **D2** es denominado bit de *sleep*, poniéndole en 0 se obtiene el funcionamiento normal del chip mientras que al ponerle un valor de 1 se lleva al

modo *sleep*. Finalmente los bits **D1** y **D0** controlan la frecuencia de las lecturas cuando el acelerómetro es llevado a modo *sleep* y son declarados de acuerdo a la frecuencia deseada basándose en la tabla 3.3.8.

Ancho de banda (Hz)	**Código**
8	00
4	01
2	10
1	11

Tabla 3.3.8: Ancho de banda captado en modo *sleep*.

Registro 0x31 - DATA_FORMAT

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
self test	SPI	int invert	0	full res	justify	range	

Tabla 3.3.9: Registro 0x31.

En la tabla 3.3.9 se muestra la composición del registro 0x31, este registro controla la presentación de los datos de los registros 0x32 al 0x37, todos los datos excepto los de rango ±16 g deben ser recortados para evitar desbordes. Poniendo a 1 el bit **D7** se aplica una fuerza de auto prueba para el sensor, provocando un cambio en los datos de salida, un valor de 0 inhabilita dicha fuerza de auto prueba. Un valor de 1 en el bit **D6** de SPI lleva al dispositivo al modo de comunicación SPI de 3 cables y un valor de 0 en dicho bit lleva al dispositivo a un modo de comunicación SPI de 4 cables. Un valor de 0 en el bit **D5** mantiene las interrupciones activas en alto, mientras que poniéndolo en 1 lleva a las interrupciones activas en bajo. El bit **D4** siempre se mantiene en 0. Cuando se lleva el bit **D3** a un valor de 1 el dispositivo entra en modo de resolución completa, manteniendo un factor de escala de 4 mg/LSB, cuando dicho bit se lleva a 0, el dispositivo entra en modo 10 bits y su factor de escala es determinado de acuerdo al rango en que se opere. Poniendo el bit **D2** a 1 se elige el modo justificado a la izquierda (MSB), mientras que poniéndolo a 0 se elige justificado a la derecha con extensión de signo.

Rango en g	Código
±2 g	00
±4 g	01
±8 g	10
±16 g	11

Tabla 3.3.10: Rangos de medición.

Finalmente, los bits **D1** y **D0** proporcionan el rango de medición y se eligen de acuerdo al rango deseado con respecto a la tabla 3.3.10.

Registro 0x32 al registro 0x37 - DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1

Estos seis bytes son dedicados a contener los datos de salida de cada eje. Los registros 0x32 y 0x33 contienen los datos del eje x, los registros 0x34 y 0x35 contienen los datos del eje y, y los registros 0x36 y 0x37 contienen los datos del eje z. Las salidas son dadas en complemento a dos con DATAx0 como el byte menos significativo y el byte DATAx1 como el más significativo (donde x representa X,Y o Z). Como se mencionó anteriormente, los datos en estos registros son controlados por el registro 0x31 (DATA_FORMAT).

Para todos los casos se recomienda llevar al dispositivo a modo *sleep* cuando se realizan cambios en los registros con la finalidad de evitar problemas de ruido o una falla en la lectura o escritura del registro. En la tabla 3.3.11 se muestra la configuración final de los registros 0x2C, 0x2D y 0x31; más adelante se presentan los resultados de las distintas configuraciones que llevan a la elección de la configuración final.

Bit / Registro	D7	D6	D5	D4	D3	D2	D1	D0
0x2C	0	0	0	0	0	1	1	0
0x2D	0	0	0	0	1	0	0	0
0x31	0	0	0	0	1	0	1	1

Tabla 3.3.11: Configuración final de registros.

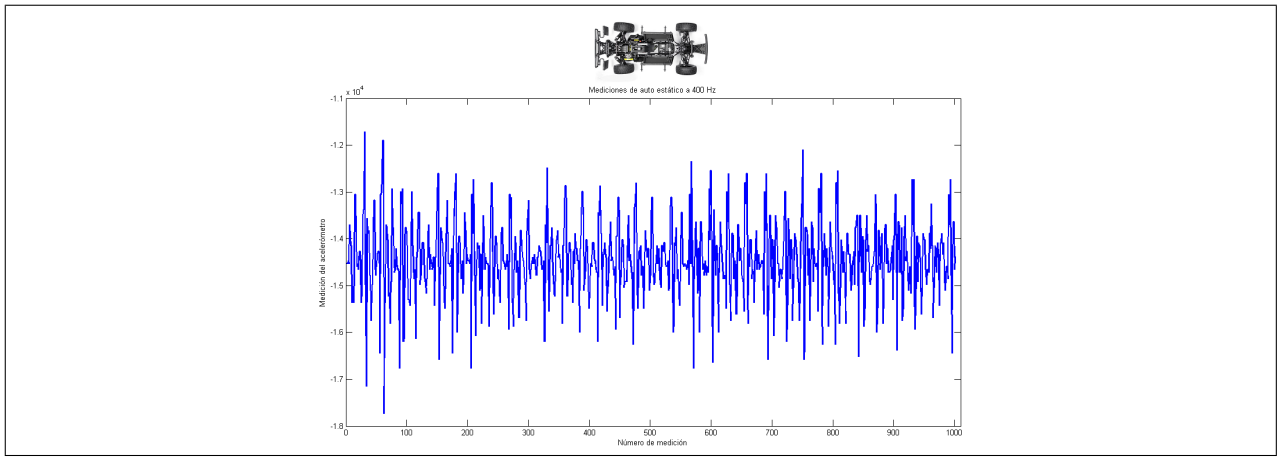

Figura 3.3.30: Experimento de auto estático con acelerómetro a 400 Hz.

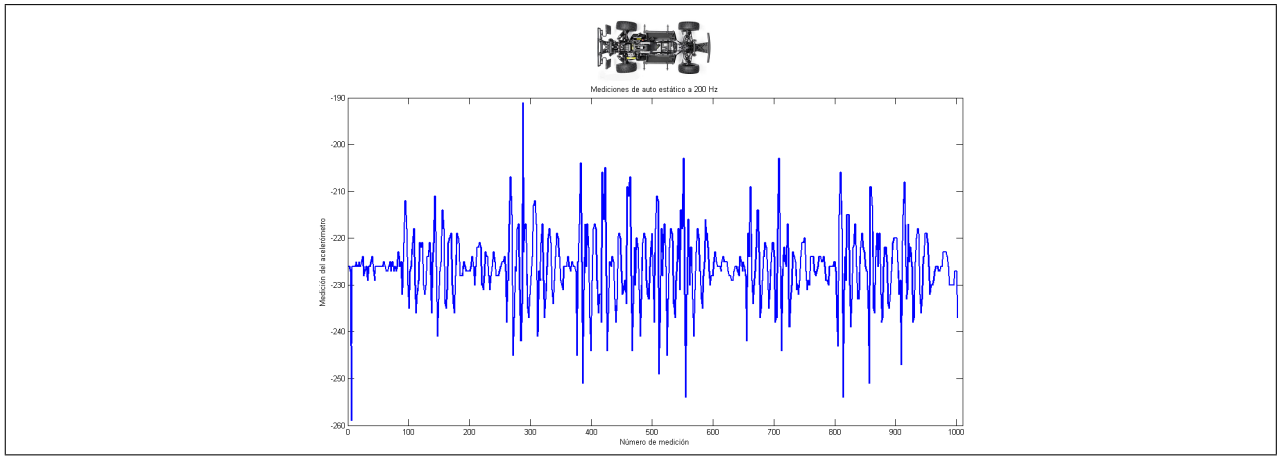

Figura 3.3.31: Experimento de auto estático con acelerómetro a 200 Hz.

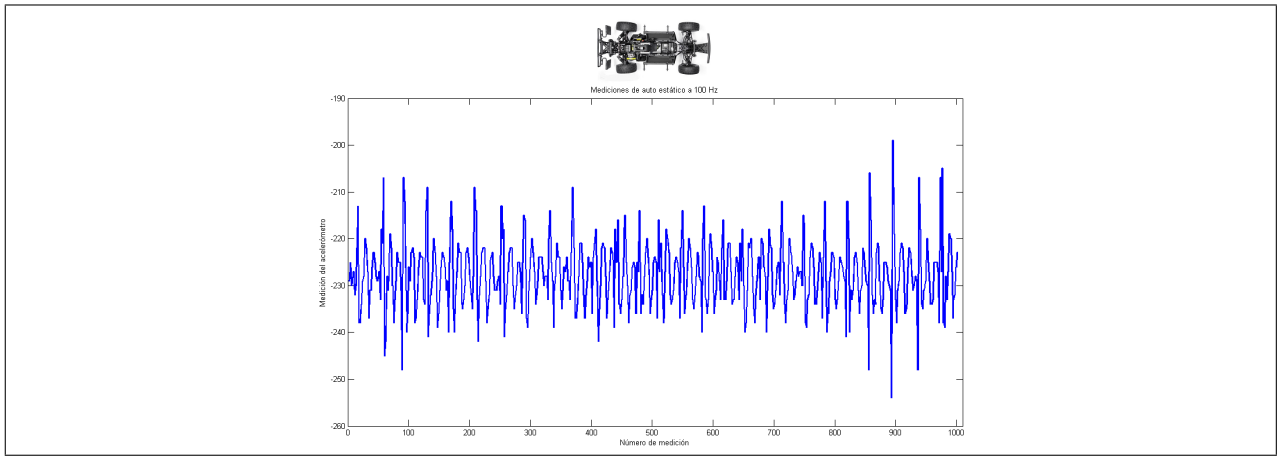

Figura 3.3.32: Experimento de auto estático con acelerómetro a 100 Hz.

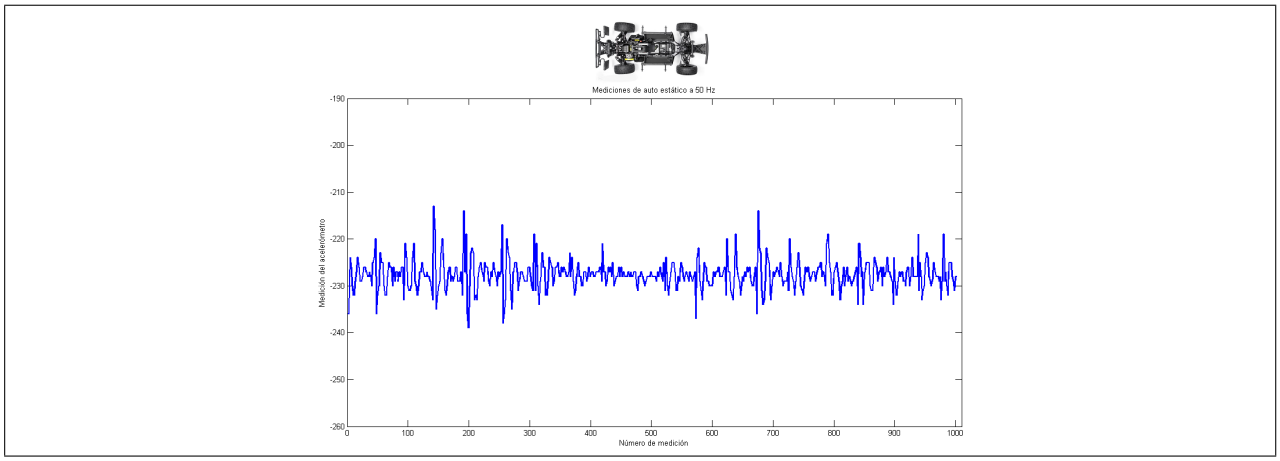

Figura 3.3.33: Experimento de auto estático con acelerómetro a 50 Hz.

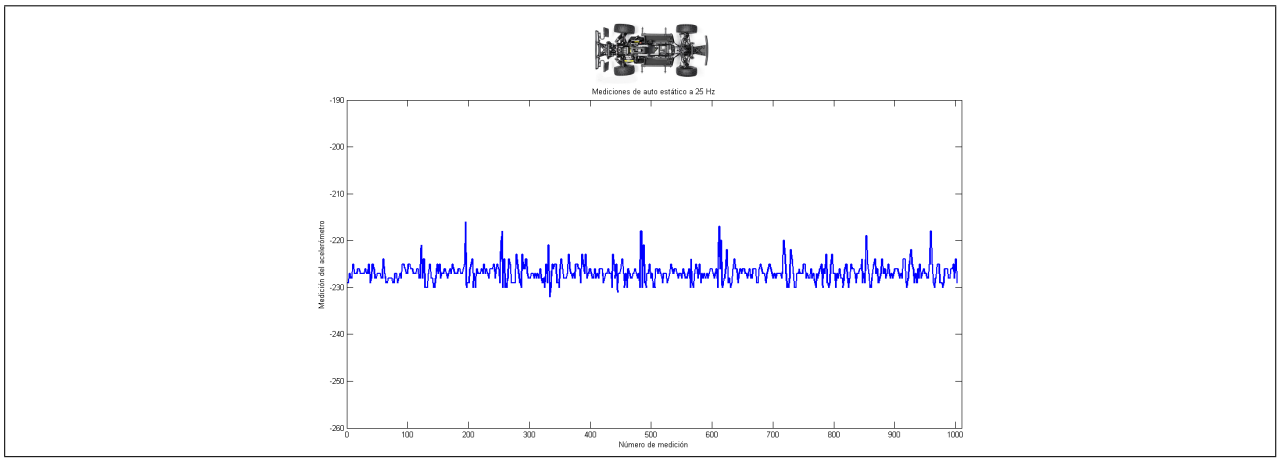

Figura 3.3.34: Experimento de auto estático con acelerómetro a 25 Hz.

Como se ha mencionado, para llegar a la configuración final de registros (particularmente del registro 0x2C) que presente lecturas de datos de mayor calidad, el sensor se somete a distintas pruebas. Las pruebas consisten en tomar lecturas del auto encendido pero sin hacerlo avanzar ni cambiar su dirección, todo esto con la finalidad de evitar en medida de lo posible la lectura de las vibraciones del motor y calibrar el sensor de tal forma que capte con un mayor efecto los movimientos de interés. Debido a la basta existencia de configuraciones en el sensor se presentan solamente los casos que arrojaron mejores resultados.

En las figuras 3.3.30 a 3.3.34 se muestran los resultados arrojados por las pruebas realizadas, una comparativa de dichos resultados puede verse en la tabla 3.3.12. El propósito de los experimentos de auto estático es obtener la menor variación posible con los acelerómetros, razón por la cual en la configuración final se elige el acelerómetro a 25 Hz (Tabla 3.3.11).

Frecuencia en Hz	Val mínimo	Val máximo	Variación
400 Hz	-17728	-11712	6016
200 Hz	-259	-191	68
100 Hz	-254	-199	55
50 Hz	-239	-213	26
25 Hz	-232	-216	16

Tabla 3.3.12: Resultados de experimento de auto estático para distintas frecuencias en acelerómetro.

Para evaluar el comportamiento de los acelerómetros cuando el auto es sometido a los movimientos que se desea captar, se realizan experimentos en los que se hace cambiar la aceleración en el eje z de la masa no suspendida. Se contempla un experimento para cada llanta con una toma de mil muestras, al inicio del experimento se da un pequeño golpe en la masa suspendida para hacerla bajar, se espera un breve instante y posteriormente se repite el movimiento un par de veces más para que culmine el experimento. Los resultados obtenidos durante los experimentos de prueba para los acelerómetros son mostrados en la tabla 3.3.13. En todos los casos se trata de experimentos de aceleración en la masa no suspendida. En la figura 3.3.35 se trata el cuadrante delantero derecho, para este caso se obtienen variaciones desde -347 hasta -74 unidades, brindando un rango total de 273 unidades. En la figura 3.3.36 se muestran los resultados para el cuadrante delantero izquierdo, se pueden observar variaciones desde -451 hasta 102 unidades con un rango total de 553. En la figura 3.3.37 se presentan los resultados para el cuadrante trasero derecho, en este experimento se obtienen valores desde -9 hasta 377 unidades con un rango de variación de 386 unidades. Finalmente, en la figura 3.3.38 se muestra la gráfica de resultados para el cuadrante trasero izquierdo, en este cuadrante se obtuvieron valores desde -22 hasta 434 unidades, con un rango de variación de 456 unidades.

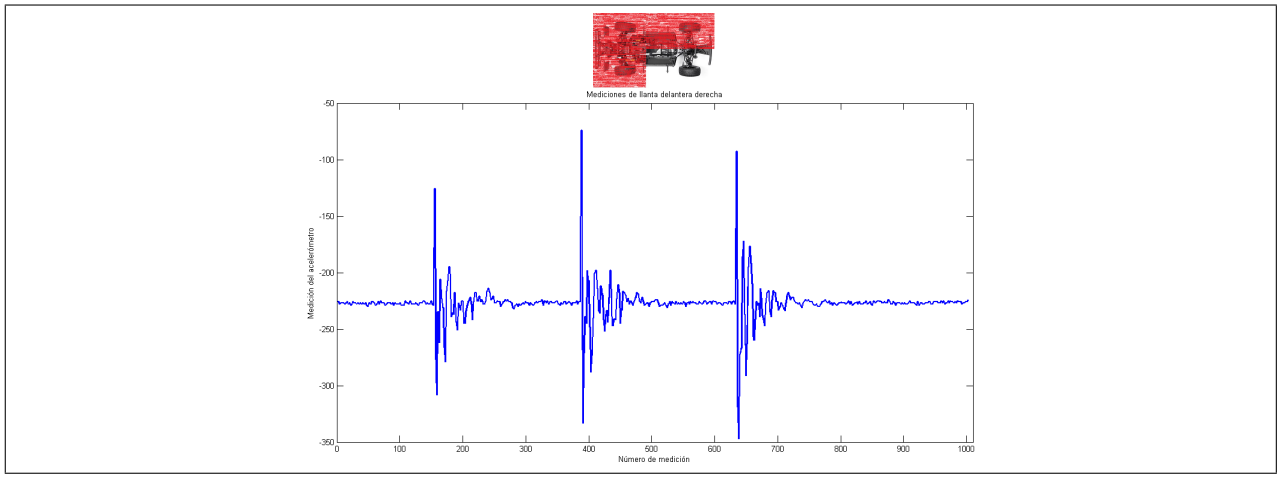

Figura 3.3.35: Experimento de aceleración en la masa no suspendida, parte delantera derecha.

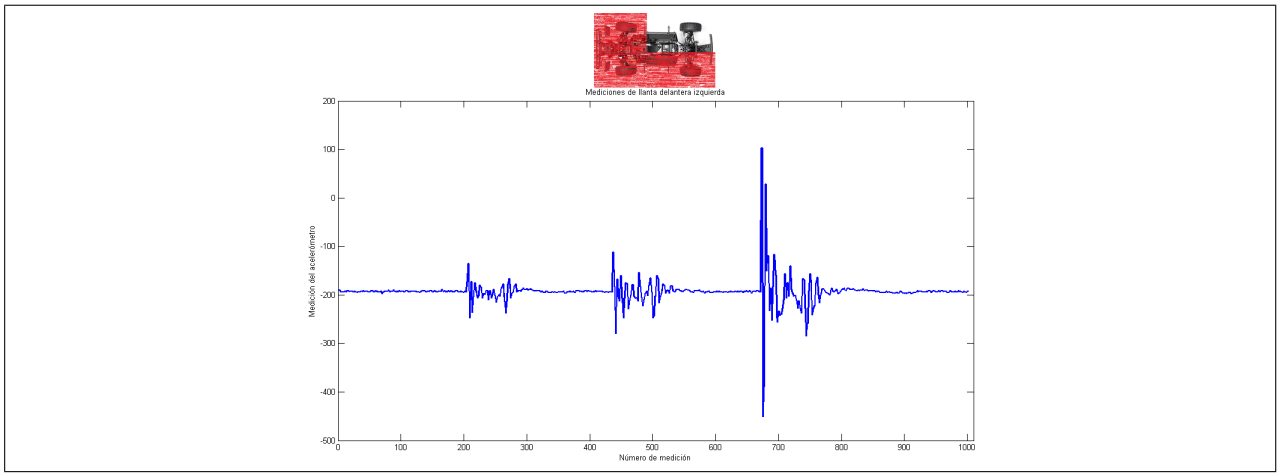

Figura 3.3.36: Experimento de aceleración en la masa no suspendida, parte delantera izquierda.

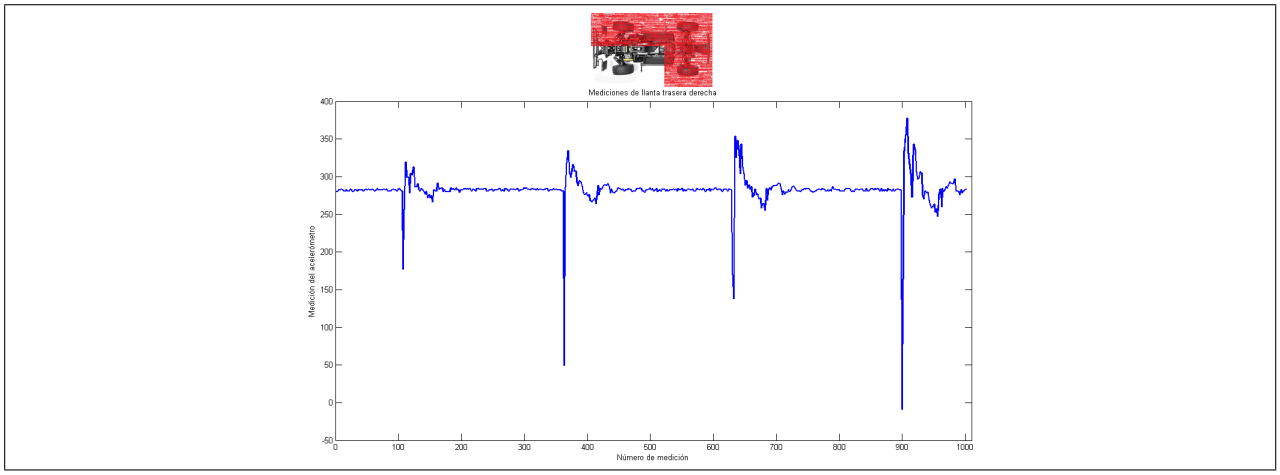

Figura 3.3.37: Experimento de aceleración en la masa no suspendida, parte trasera derecha.

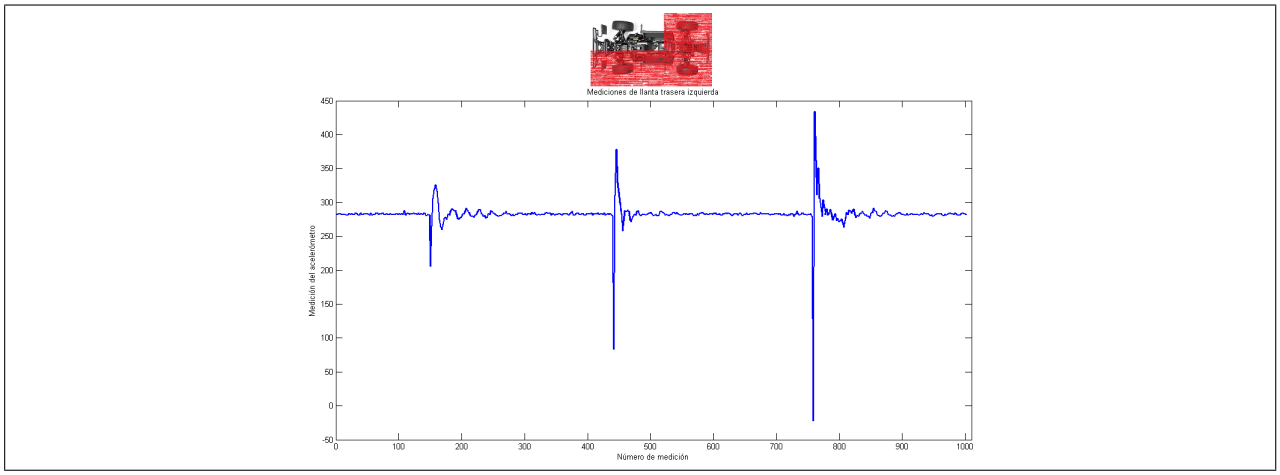

Figura 3.3.38: Experimento de aceleración en la masa no suspendida, parte trasera izquierda.

Cuadrante	Val mínimo	Val máximo	Variación
Delantero derecho	-347	-74	273
Delantero izquierdo	-451	102	553
Trasero derecho	-9	377	386
Trasero izquierdo	-22	434	456

Tabla 3.3.13: Resultado de experimentos de aceleración en la masa no suspendida para acelerómetros.

Basándonos en los resultados de los experimentos de prueba de acelerómetros, se puede concluir que los datos obtenidos por medio de los sensores en cuestión garantizan su calidad al ser sintonizados de forma correcta, con lo cual se ha llegado a captar los movimientos deseados mientras que se reduce la lectura de movimientos no deseados (vibraciones del motor).

3.4. Diseño electrónico y de conexiones

La parte electrónica del sistema del Sensor Virtual está conformada por todos los sensores físicos elegidos para medir las variables relacionadas, amplificadores operacionales que cumplen con el acondicionamiento de las señales medidas, tres microcontroladores EK-TM4C123GXL - Tiva C Series que se darán a la tarea de capturar y guardar los datos de forma instantánea, una cámara que se encargará de tomar imágenes del área de contacto real de las llantas del vehículo, una computadora personal (PC) que se empleará para controlar el proceso, procesar las imágenes tomadas por la cámara y guardar los datos de forma permanente en un banco o base de datos, finalmente un disparador que indicará el momento en que un proceso debe comenzar.

El desarrollo del Sensor Virtual puede ser dividido en tres etapas para fines del diseño electrónico. La primera parte será la etapa de construcción del Sensor Virtual, en esta etapa el sistema se encarga de adquirir los datos que se emplearán para entrenar una Red Neuronal Artificial, las mediciones de los sensores se darán como entradas mientras que las imágenes de la cámara después de ser procesadas se tomarán como el valor de salida deseada. La segunda etapa es la de validación, en esta etapa se evalúa el comportamiento de la RNA construida. Finalmente, en la tercera etapa se prescinde del sensor físico que toma las mediciones reales (en este caso la cámara) y el Sensor Virtual es empleado para estimar la variable deseada en tiempo real.

3.4.1. Diseño electrónico para la etapa de construcción

Como se mencionó anteriormente, en la etapa de construcción se toman los datos de los sensores físicos y de la cámara para construir la Red Neuronal del Sensor Virtual. En la figura

3.4.1 se muestra el esquema de conexiones eléctricas para la construcción del sensor virtual. Se utiliza un total de tres microcontroladores, el primero se encargará de la lectura de la señal de PWM y de los datos de la IMU, mientras que los dos restantes se encargarán de obtener los datos de dos sensores de flexión y dos acelerómetros cada uno. Adicionalmente, el primer microcontrolador se encargará de comunicarse con una PC que, por medio de un pulso digital de activación, le indicará cuándo se deben realizar mediciones; al recibir el pulso de activación, el primer microcontrolador se encargará de propagarlo hacia los dos microcontroladores restantes.

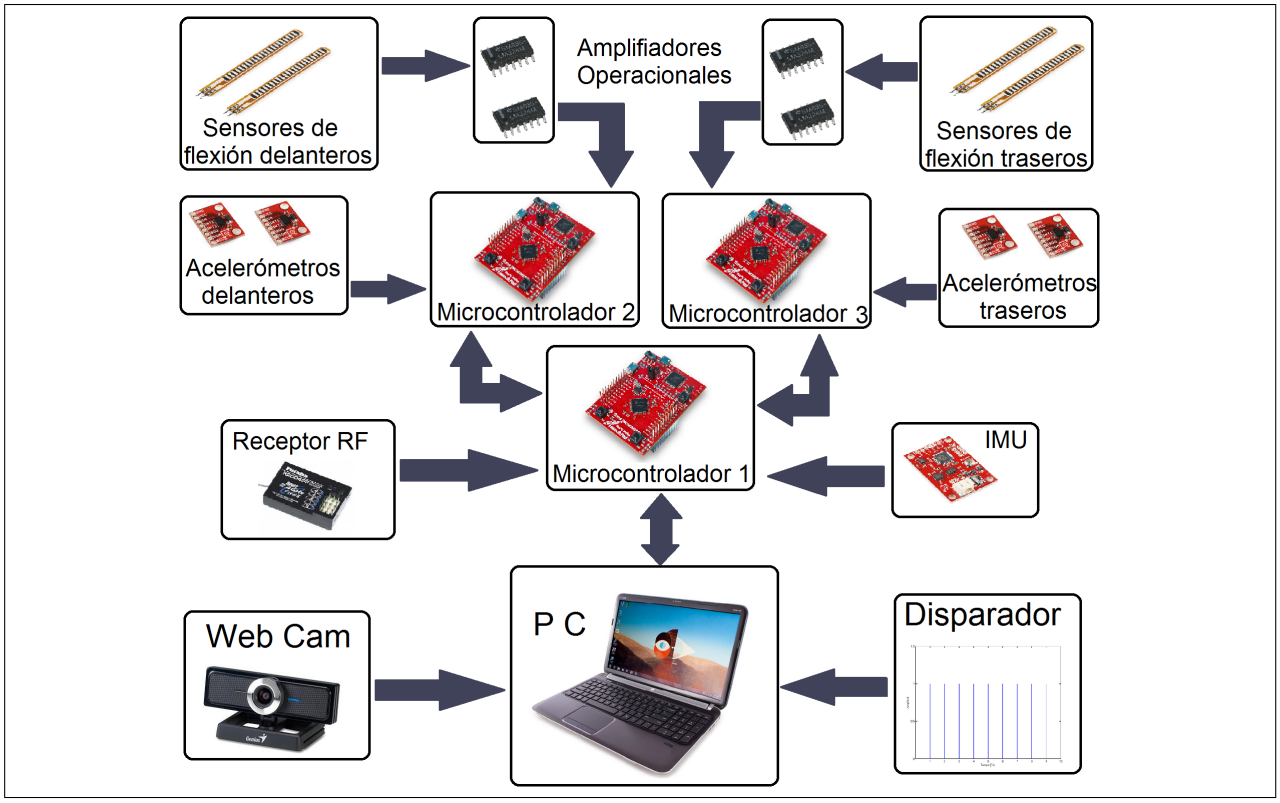

Figura 3.4.1: Esquema eléctrico de la etapa de construcción.

Se puede observar que los sensores de flexión requieren de una etapa de acondicionamiento de señal por medio de amplificadores operacionales previo a ser conectados a los microcontroladores, esto con el fin de obtener una señal analógica con rango de 0 a 3.3 Volts ya que éste es el voltaje que manejan los microcontroladores.

Los acelerómetros se conectarán directamente a los microcontroladores debido a que ambos manejan los mismos niveles lógicos, la interfaz por la cual se comunicarán es SPI de 4 cables. La IMU tiene un protocolo de comunicación RS-232 y la señal de PWM es una señal digital ambos con niveles lógicos de 0 a 3.3 volts, por lo tanto pueden comunicarse directamente a los microcontroladores sin necesidad de algún acondicionador de niveles lógicos.

El proceso de construcción del sensor virtual es controlado por una PC que tiene la facultad de comunicarse con el disparador del proceso que puede ser físico o digital, con la cámara web por medio de un puerto USB y con el primer microcontrolador ya sea por otro puerto USB o por un módulo Bluetooth de forma inalámbrica.

3.4.2. Diseño electrónico para la etapa de validación

En la etapa de validación, se emplea el sistema para realizar estimaciones al mismo tiempo que se toman mediciones del área real. Para los fines mencionados se eliminan dos de los tres microcontroladores utilizados en la etapa de construcción y la Red Neuronal entrenada se implementa en el microcontrolador restante. Todos los sensores físicos son conectados de la misma manera que en la etapa de construcción, salvo que ahora se conectan a un solo microcontrolador (figura 3.4.2). Debido a que la medición del área real y la estimación del Sensor Virtual tienen que darse relativamente en un mismo instante, la PC sigue siendo empleada para controlar el proceso e indicar cuándo el disparador manda a tomar medición. De la misma manera, se conserva la cámara Web comunicada por medio del puerto USB.

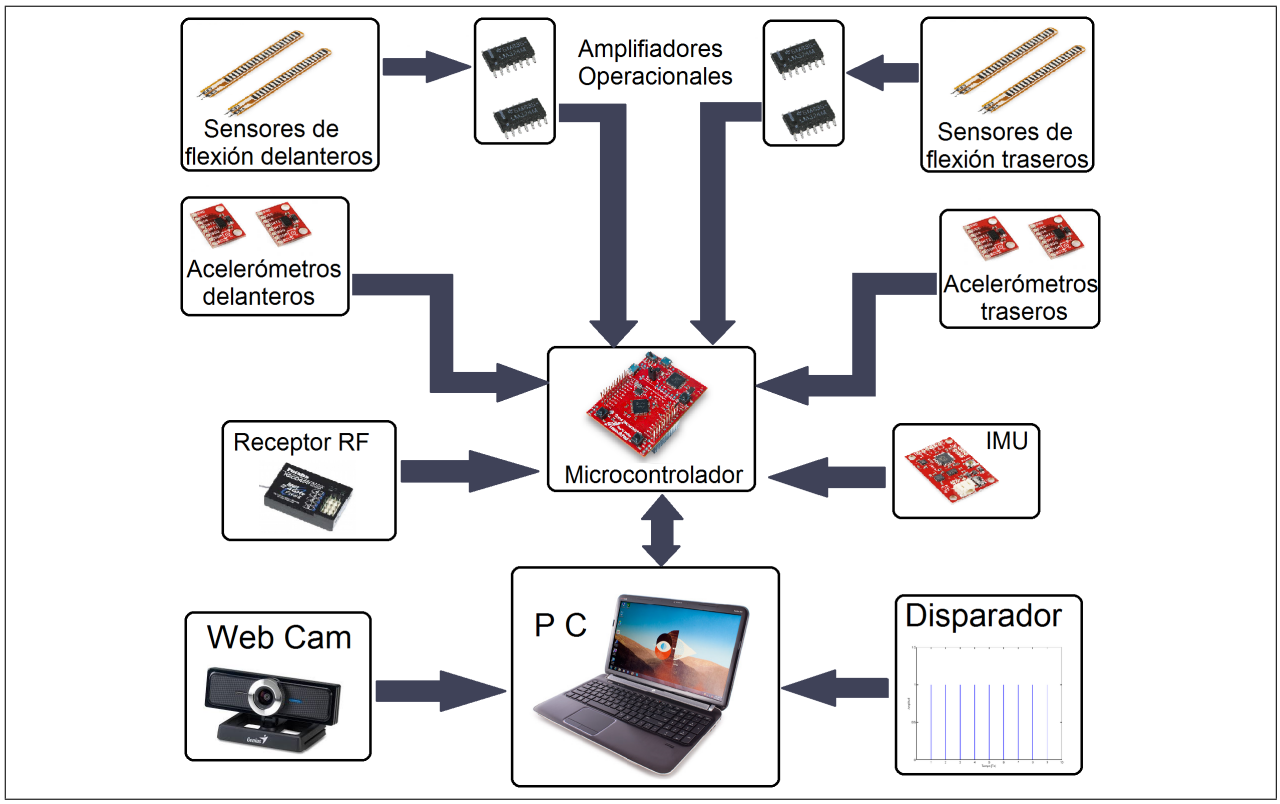

Figura 3.4.2: Esquema eléctrico de la etapa de validación.

3.4.3. Diseño electrónico para la etapa de implementación

La etapa final o de implementación del Sensor Virtual es cuando se ha demostrado que las estimaciones son lo suficientemente confiables y se prescinde de otros sensores físicos, para tomar la estimación en tiempo real como única medición del sistema. En esta etapa, el microcontrolador que contiene la Red Neuronal Artificial entrenada sigue teniendo la totalidad de sensores físicos conectados como entrada y a la salida ahora puede tener bastantes tipos de dispositivos a los cuales comunica la estimación realizada para que cada dispositivo pueda actuar de acuerdo a los fines perseguidos por el sistema en que se encuentra, ésto puede ser el inicio de una etapa de control, por ejemplo. Como el Sensor Virtual ha sido validado y ahora entrega mediciones de forma constante, al llegar a esta etapa se desconecta el sistema de medición real y es por ello que en la figura 3.4.3 no se observa la cámara Web ni el disparador.

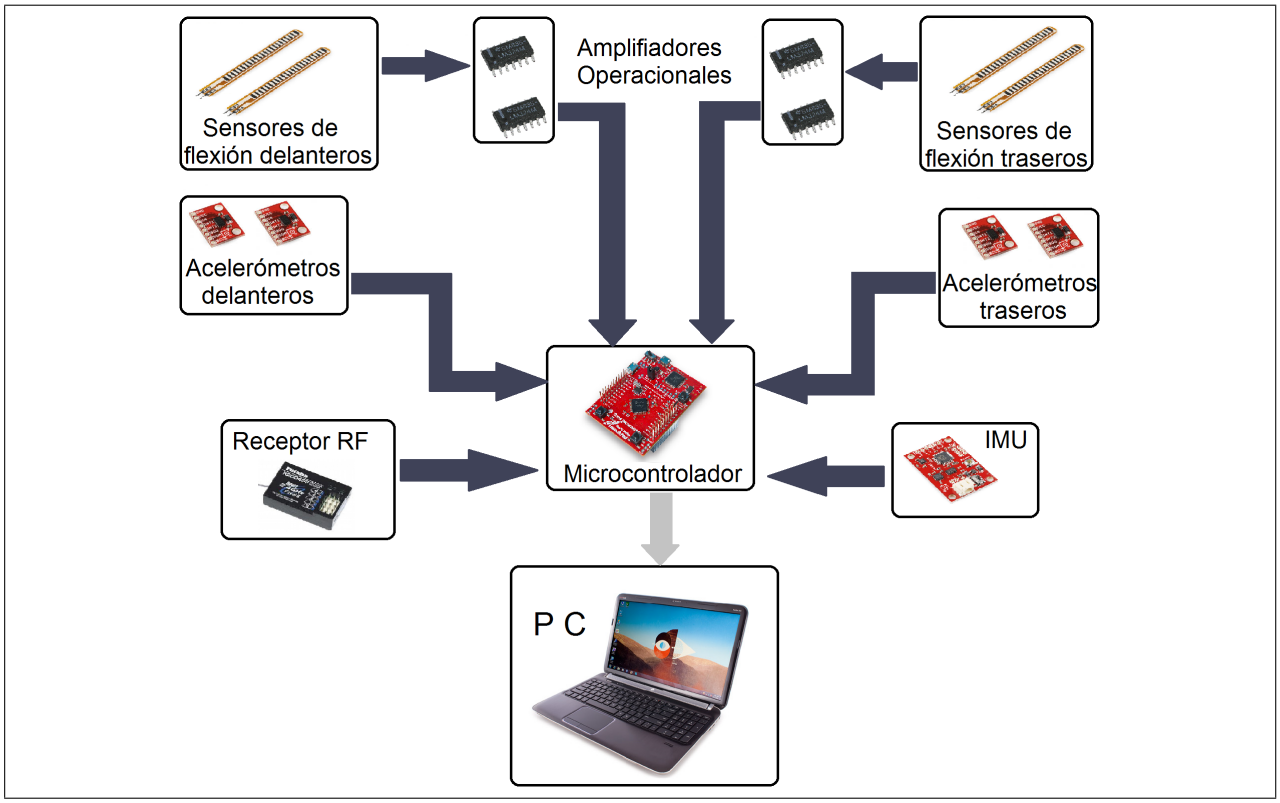

Figura 3.4.3: Esquema eléctrico de la etapa final.

3.5. Diseño de la programación

Para llevar a cabo la adquisición de los datos proporcionados por los sensores y su transferencia a una base de datos de una PC, donde se almacenarán de forma permanente, se utilizan tres microcontroladores EKTM4C123GXL-Tiva C Series y una Computadora Personal con el *software* MATLAB R2010a. El diseño de la programación contempla de forma

estratégica las etapas de construcción, validación e implementación con las que se realizó el diseño electrónico a fin de tener homogeneidad en el diseño integral.

3.5.1. Diseño de la programación para la etapa de construcción

En la etapa de construcción el sistema se da a la tarea de recopilar los datos obtenidos por los sensores en un instante específico y llevarlos a un banco de datos en la PC, posteriormente se ocupan los datos del banco para entrenar una Red Neuronal que será utilizada en las etapas de validación e implementación. A continuación se da una descripción de los programas y sus diagramas de flujo diseñados para cumplir con la etapa de construcción:

Sincronización del sistema En la figura 3.5.1 se muestra el diagrama de flujo de la sincronización del sistema. La etapa de construcción inicia con este programa dedicado a asegurar la calidad de los datos adquiridos y que dichos datos sean tomados en un lapso de tiempo tan pequeño como sea posible. Este programa es cargado en la PC y funciona de la siguiente manera: Se tiene como entrada un disparador de cualquier tipo (inicialmente por *software*) que indicará el momento en que comienza un proceso de medición. Cuando se activa el disparador, un pulso de sincronización es generado por la PC para marcar que el proceso de medición ha comenzado, dicho pulso es enviado al primer microcontrolador para ser propagado mientras en la PC se comienza a capturar una fotografía del contacto de las llantas por medio de la cámara web. Al finalizar la adquisición de la imagen, se piden los valores leídos por los sensores al microcontrolador y se muestran todos los datos para que, por medio de inspección visual se determine si fueron correctamente tomados, en caso de ser correctos se inicia un programa que se encargará del almacenamiento en un banco de datos.

Microcontrolador 1 En la figura 3.5.2 se muestra el diagrama del programa en el Microcontrolador 1. Este programa funciona de la siguiente manera: Inicia comunicación con la PC que le indica cuándo es momento de iniciar un proceso por medio de un pulso. Al momento de que las mediciones son requeridas por la PC, el microcontrolador se encarga de propagar el pulso hacia los otros dos microcontroladores para indicarles que es momento de iniciar mediciones. Posteriormente toma los valores del ángulo del sistema de dirección y de los ángulos de cabeceo y alabeo de la IMU . Los datos obtenidos se van almacenando en una matriz hasta llegar al número de datos requeridos por el usuario y se espera a que la PC los pida. Si la PC pide los datos, entonces se le envían en orden para que sean revisados y almacenados de forma correcta.

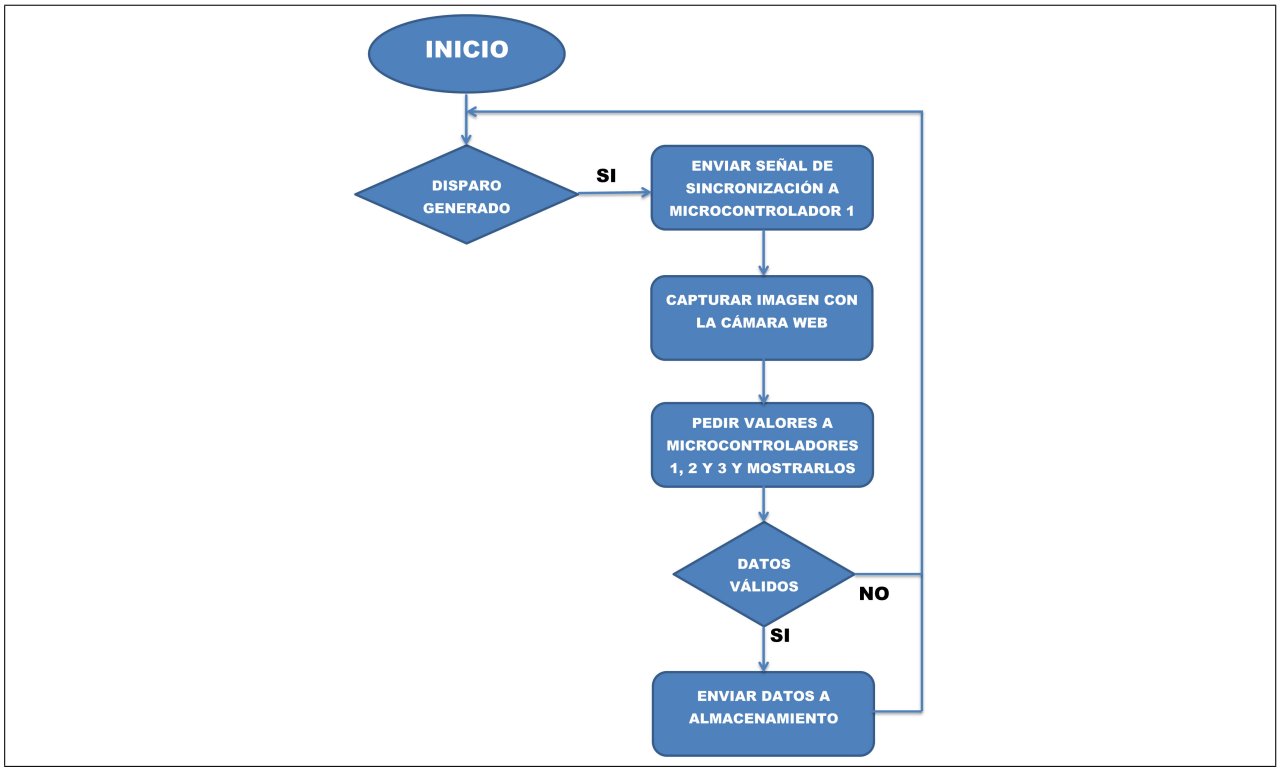

Figura 3.5.1: Diagrama de flujo del programa de Sincronización del sistema.

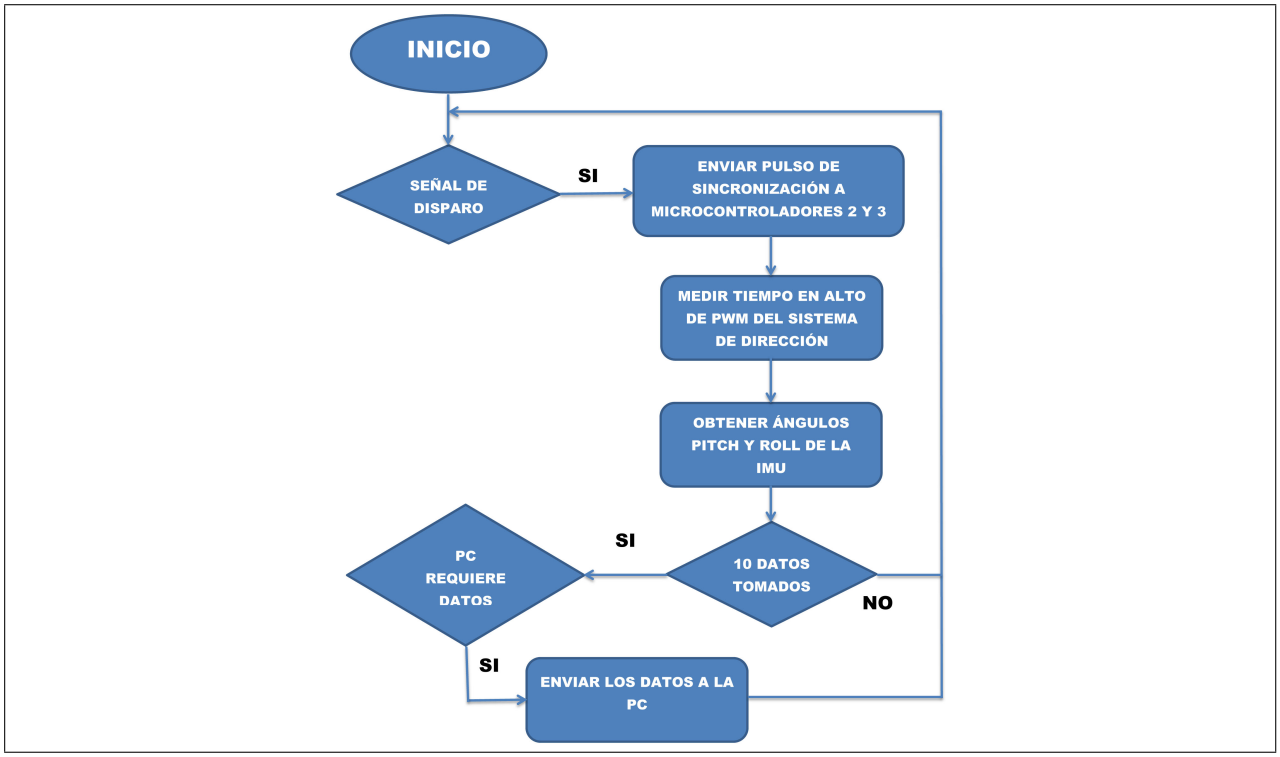

Figura 3.5.2: Diagrama de flujo del programa del Microcontrolador 1.

Microcontroladores 2 y 3 En la figura 3.5.3 se muestra el diagrama del programa en los Microcontroladores 2 y 3. Son dos programas idénticos que se cargan en cada uno de los microcontroladores restantes. Se inicia esperando que el microcontrolador 1 ordene iniciar mediciones por medio de una señal digital. Si se recibe la orden, se inicia la obtención de datos proporcionados por dos acelerómetros y dos sensores de flexión. Se realiza la medición el número de veces que el usuario desea y se almacenan los datos obtenidos hasta que son pedidos para llevarlos al banco de datos. El proceso termina cuando se requieren los datos y éstos son enviados.

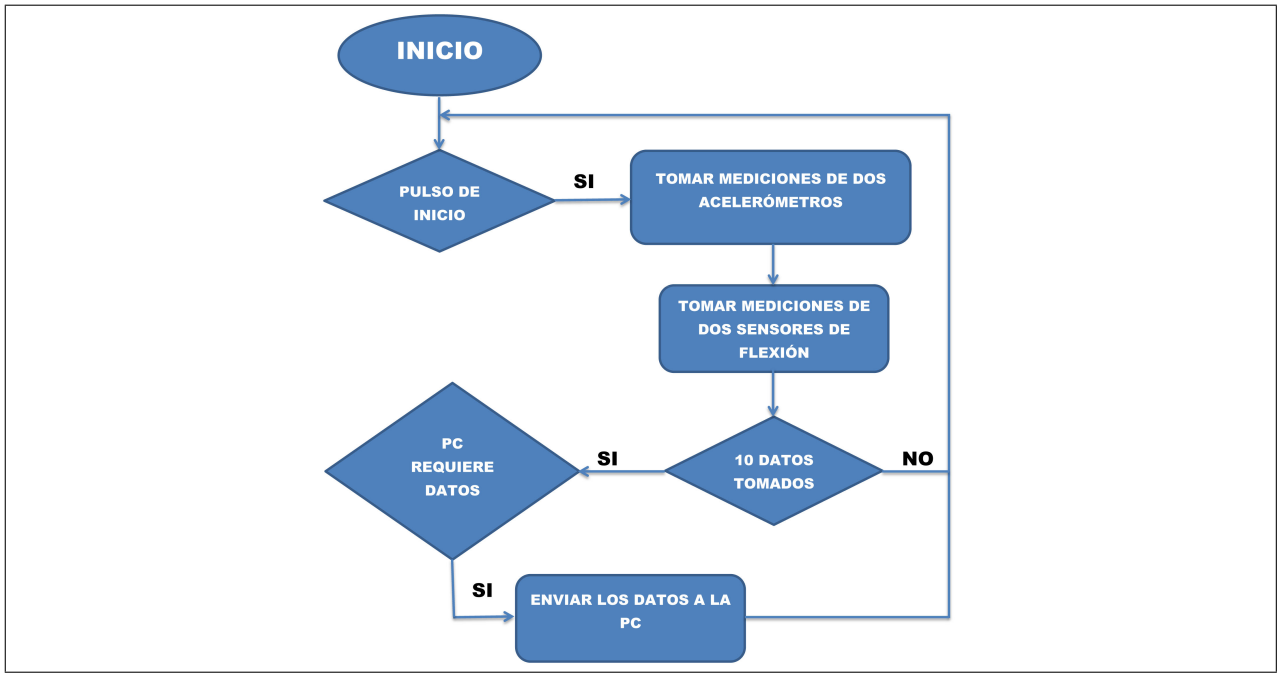

Figura 3.5.3: Diagrama de flujo del programa de los Microcontroladores 2 y 3.

Almacenamiento de datos En la figura 3.5.4 se muestra el diagrama de flujo del Almacenamiento de datos. Este programa es cargado en la PC y se encarga de recibir los datos válidos que se utilizarán para entrenamiento de la Red Neuronal Artificial. El programa únicamente recibe los datos que han sido validados previamente. Cuando se tienen la totalidad de esos datos, se inicia el procesamiento necesario de las imágenes de contacto de las llantas para determinar el valor numérico del área de contacto en cada una. Después de obtener todos los datos numéricos de los valores en los sensores y del área de contacto, se procede a almacenarlos en una base de datos generada en una hoja de cálculo en Excel.

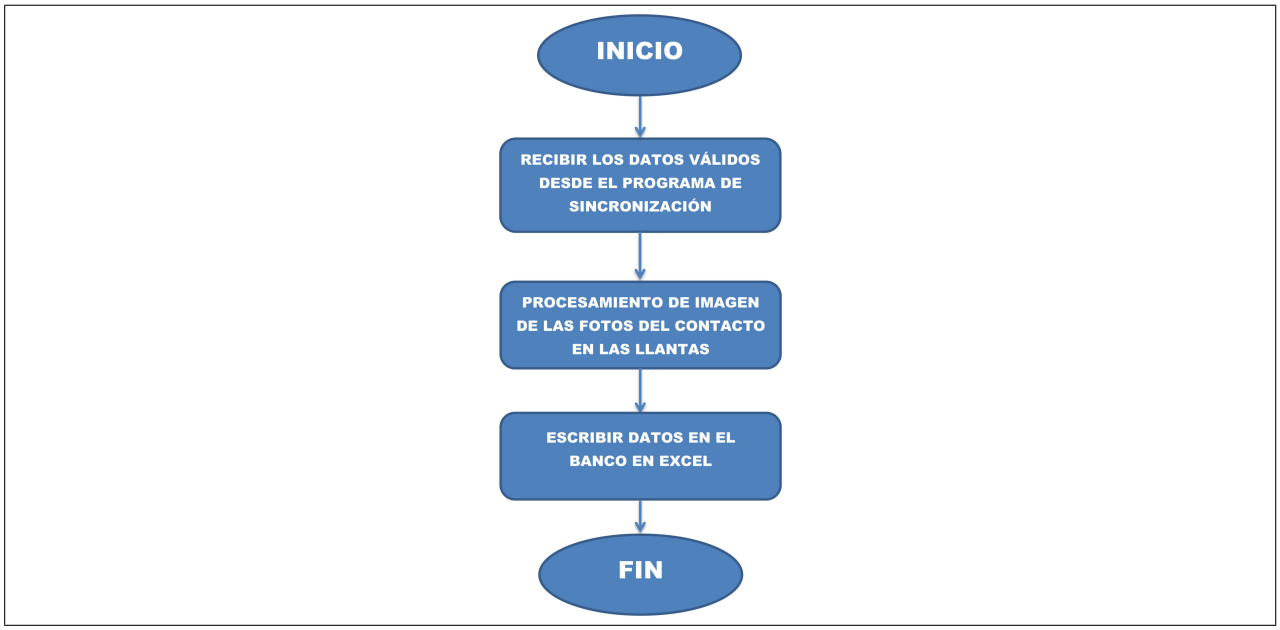

Figura 3.5.4: Diagrama de flujo del programa de Almacenamiento de datos.

Entrenamiento En la figura 3.5.5 se muestra el diagrama de flujo del programa de Entrenamiento. Por medio de las cajas de herramientas de Matlab se importan los datos del banco de datos en Excel y se entrena una Red Neuronal, la Red Neuronal resulta en un objeto que contiene vectores y matrices con los valores del número de capas, el número de neuronas por capa, el peso de las conexiones y el *bias*. Dichos valores se almacenan para ser cargados en un microcontrolador y pasar a la etapa de validación e implementación.

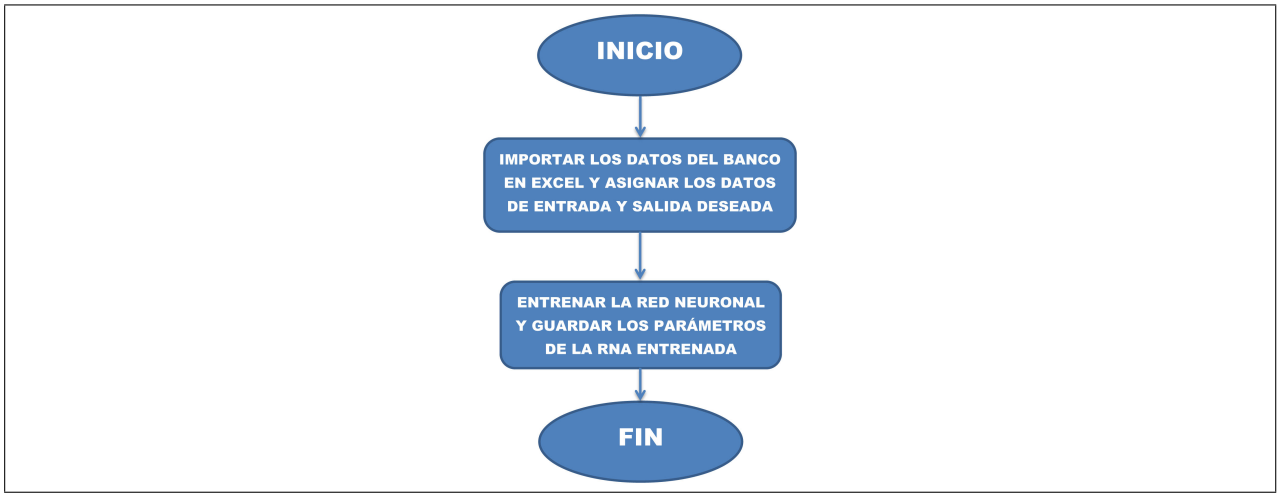

Figura 3.5.5: Diagrama de flujo del programa de Entrenamiento.

3.5.2. Diseño de la programación para las etapas de validación e implementación

Para las etapas de validación e implementación se diseñaron los siguientes programas:

Validación En la figura 3.5.6 se muestra el diagrama de flujo del programa de validación. Es un programa básico cargado en la PC que simplemente introduce datos del banco de datos, ya sea que hayan sido utilizados para el entrenamiento o no, y evalúa si la estimación de la RNA es correcta.

Figura 3.5.6: Diagrama de flujo del programa de Validación.

Sensor Virtual En la figura 3.5.7 se muestra el diagrama de flujo del programa en el Sensor Virtual. Este programa es cargado en un microcontrolador. Se encarga de esperar a que se le pida el área de contacto y entonces tomar la medición de todos los sensores ya que los dos microcontroladores restantes son eliminados para esta etapa. Después de tener los valores de los sensores, se hace el cálculo del área de contacto por medio de la Red Neuronal previamente cargada. Finalmente envía el valor del área de contacto hacia el dispositivo que lo haya requerido.

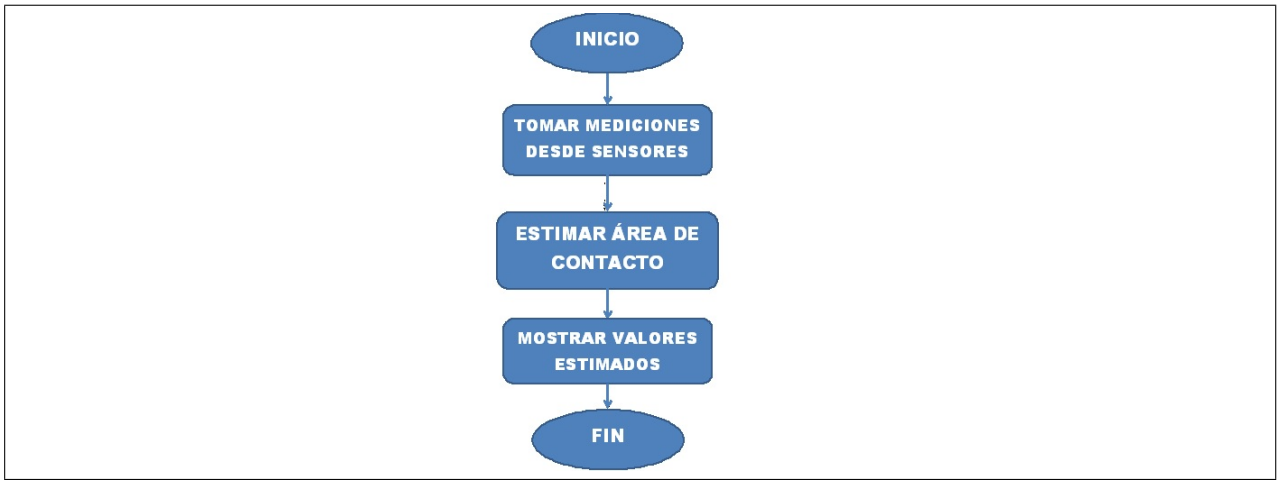

Figura 3.5.7: Diagrama de flujo del programa de Sensor Virtual.

Adquisición En la figura 3.5.8 se puede observar el diagrama de flujo del programa de adquisición. Es un programa cargado en la PC parecido al programa "Sincronización", la diferencia principal entre los dos programas es que en éste no se adquieren los datos de los sensores sino que ahora se obtiene la estimación del área de contacto proporcionada por el sensor virtual y no se intenta revisar si los datos son válidos en este punto, en lugar de ello se utiliza un programa de validación adicional.

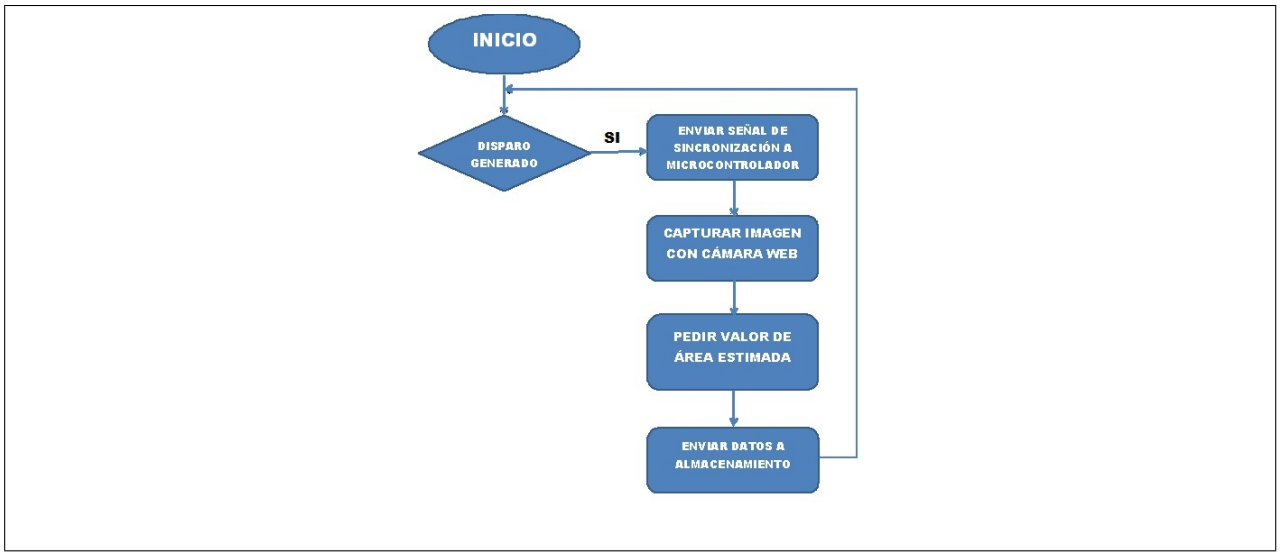

Figura 3.5.8: Diagrama de flujo del programa de Adquisición.

Procesamiento En la figura 3.5.9 se muestra el diagrama de flujo del programa de procesamiento. Dicho programa se encarga de procesar la imagen del área de contacto obtenida con la cámara para obtener el área de contacto real y, posteriormente compararlo con el área de contacto estimada, cuando se obtienen estimaciones muy cercanas a las mediciones reales, se tiene un Sensor Virtual confiable y listo para su implementación.

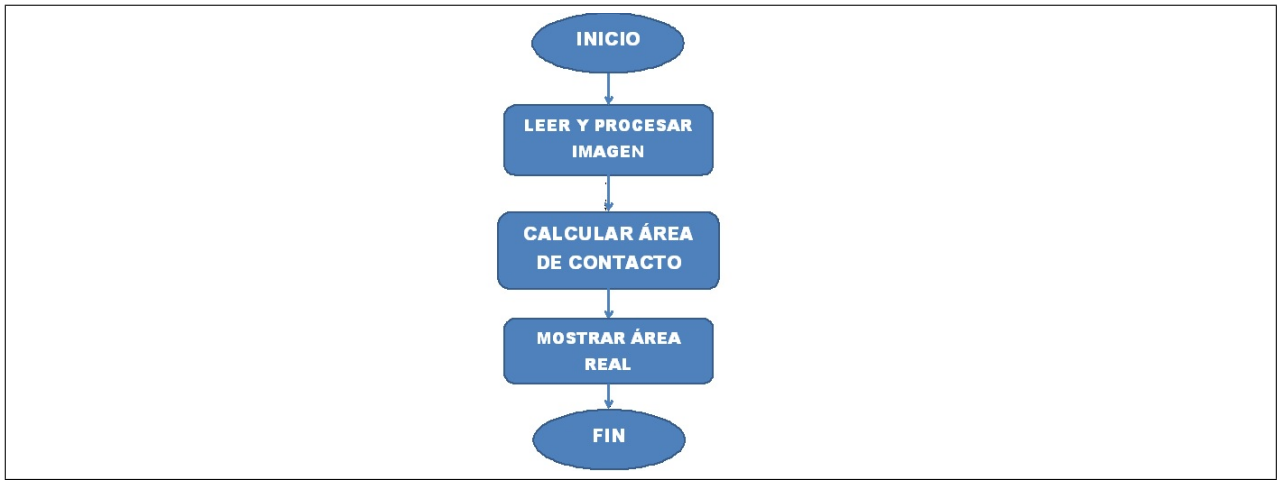

Figura 3.5.9: Diagrama de flujo del programa de Procesamiento.

3.6. Integración de la plataforma experimental.

Para llevar a cabo la adquisición de datos de los sensores y su transferencia a una base de datos en una PC, se utilizan tres microcontroladores EKTM4C123GXL-Tiva C Series. El montaje de los microcontroladores al vehículo se hace por medio de una placa fenólica en la que de igual forma se conecta el cableado de los sensores físicos (figura 3.6.1), de forma adicional se construyó una placa en la que se conectan directamente los sensores de flexión para pasar por una etapa de amplificación por medio de amplificadores operacionales (figura 3.6.2). En la figura 3.6.3 se puede observar el vehículo entero completamente instrumentado con sensores y microcontroladores, situado sobre el banco de pruebas.

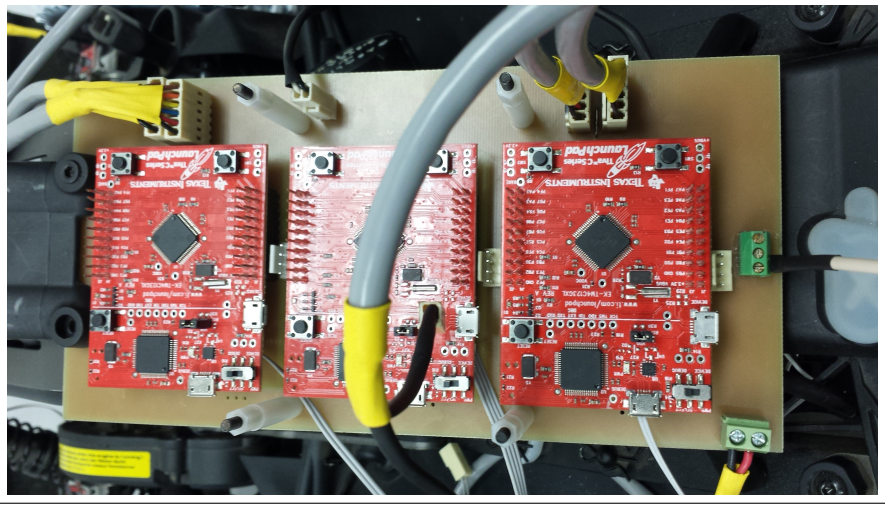

Figura 3.6.1: Placa con microcontroladores.

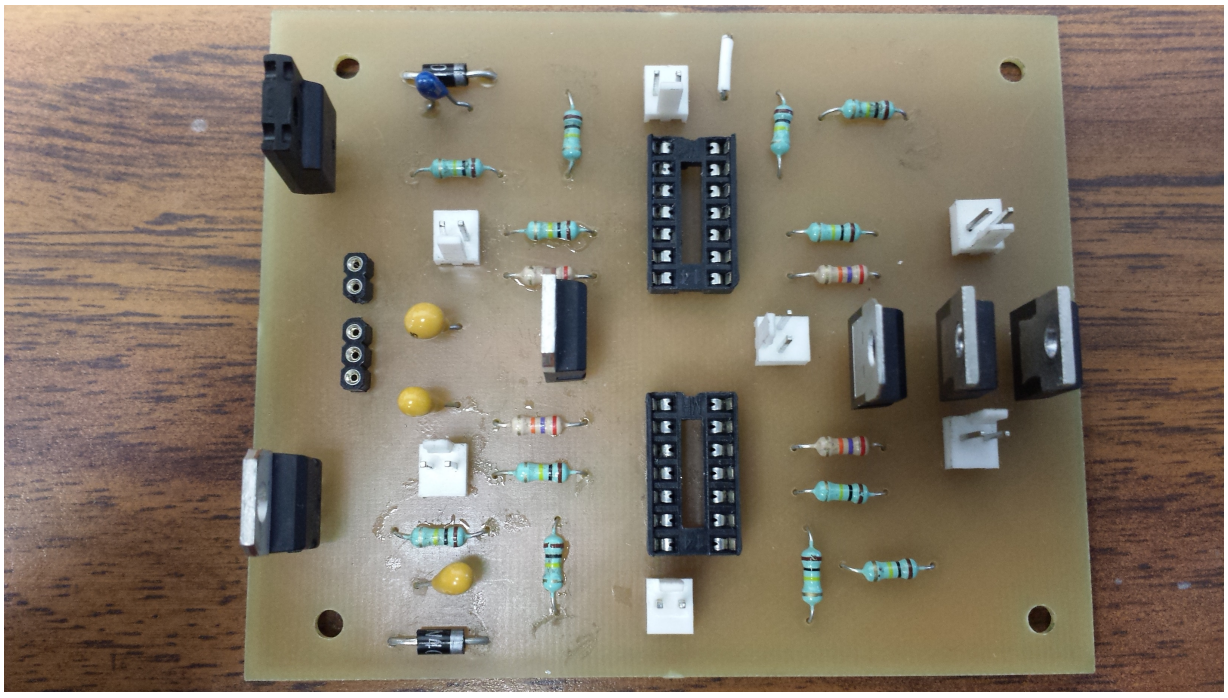

Figura 3.6.2: Placa del amplificador operacional.

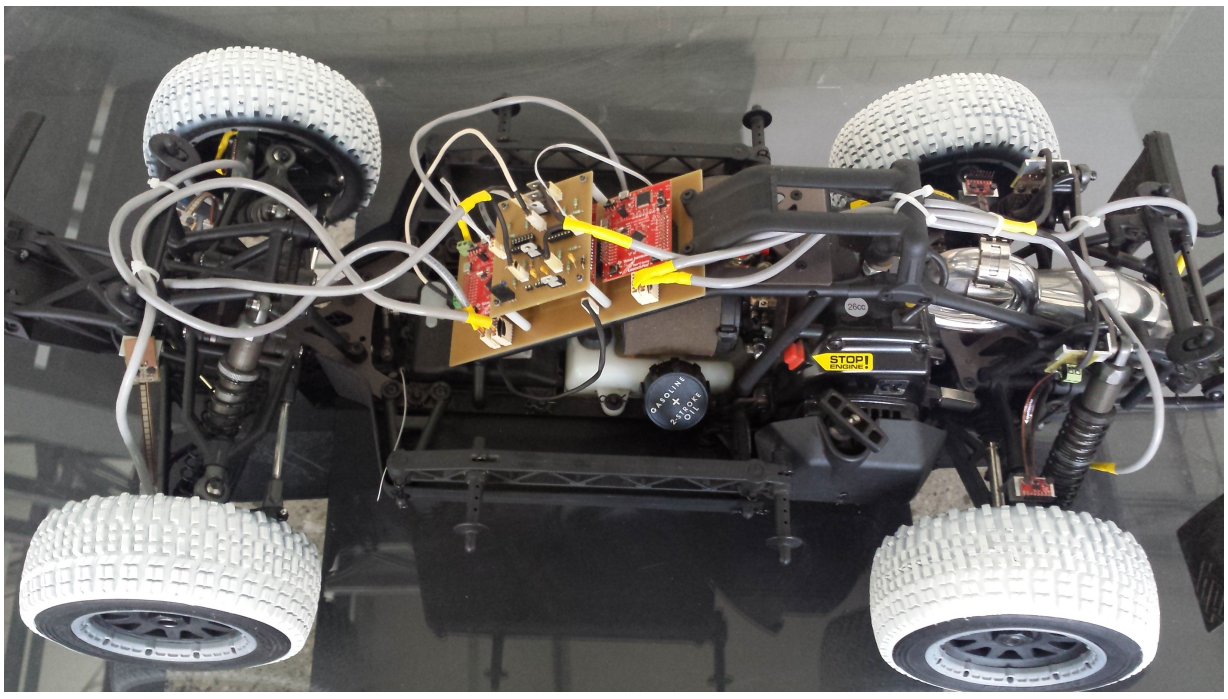

Figura 3.6.3: Vehículo instrumentado.

Para la etapa de construcción, cada microcontrolador cumple con una tarea específica.

Un primer microcontrolador se encarga de la comunicación entre los tres microcontroladores y la PC, además de tomar las mediciones otorgadas por la IMU y el temporizador. Los dos microcontroladores restantes solamente están a la espera de un pulso que indica que se deben iniciar las mediciones y cada uno se encarga de obtener los datos correspondientes a dos sensores de flexión y dos acelerómetros. Para saber el tiempo que tarda un proceso de muestreo completo, se toman mediciones por medio de *software* y de osciloscopio para cada sensor. Para obtener cada tiempo de muestreo medido por *software* se toma un total de 100 muestras y el tiempo en que tarda el programa en tomar dichas muestras, posteriormente se divide el tiempo tomado entre el número de muestras y se obtiene el tiempo de muestreo. En el caso de la obtención de tiempo de muestreo medido por osciloscopio, se crea un bucle con muestras continuas y al finalizar cada muestra se envía un pulso que será captado por el osciloscopio para indicar que ha concluido una medición, posteriormente se toma el tiempo de periodo como tiempo de muestreo. En la tabla 3.6.1 se muestran los resultados obtenidos para los tiempos de muestreo en cada sensor y el método que se utilizó para obtenerlo.

Método de medición	**Sensor**	**Tiempo de muestreo en milisegundos**
Software	Cámara web	33
Software	IMU	0.5
Osciloscopio	Timer	15
Osciloscopio	Flexión	0.2
Osciloscopio	Acelerómetro	0.7

Tabla 3.6.1: Tiempo de muestreo por cada sensor.

La sincronización de las lecturas en cada sensor se da de la siguiente forma: El proceso comienza con un disparo que indica a la PC que se deben iniciar las mediciones, posteriormente se escribe al primer microcontrolador para propagar la señal de sincronización; mientras el microcontrolador recibe la señal de sincronización, se comienza la captura de la primer imagen. Cuando el primer microcontrolador recibe la señal de sincronizción éste la propaga por medio de un pulso a los dos microcontroladores restantes y después pasa a tomar la medición en la IMU y en el temporizador. Teóricamente, los dos microcontroladores restantes reciben la señal de sincronización al mismo tiempo, mientras el primero comienza la lectura de la IMU. Una vez obtenida la señal de sincronización, ambos microcontroladores comienzan a tomar los datos de dos sensores de flexión y dos acelerómetros cada uno (de forma concurrente). Lo anterior brindará hasta un máximo de 30 muestras por segundo si el proceso es repetido de forma consecutiva. Cabe aclarar que durante el proceso presentado no se contempla el tratamiento de datos, lo cual se realiza fuera de línea.

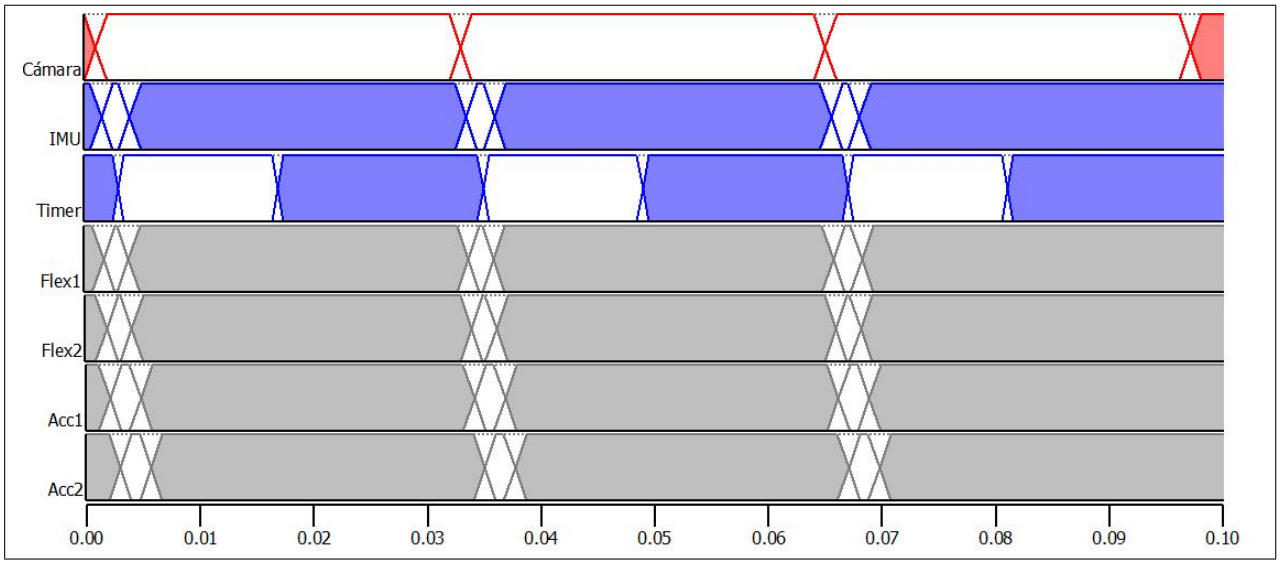

Figura 3.6.4: Diagrama de tiempos del proceso de muestreo.

Tomando los datos obtenidos y el proceso de sincronización, se construye el diagrama de tiempos mostrado en la figura 3.6.4. El proceso de muestreo dado por la PC se representa en colo rojo, para este caso solamente se toman los datos del sensor visual. El proceso de muestreo generado por el primer microcontrolador es presentado en color azul, éste contiene las mediciones de la IMU y el timer. Finalmente, en color gris se presenta el proceso de muestreo de los dos microcontroladores restantes, que funcionan de forma idéntica, tomando mediciones de dos sensores de flexión y dos acelerómetros cada uno. Nótese que el diagrama de tiempos está construido para un lapso de una décima de segundo.

Resumen

Después de haber concretado la etapa total de diseño, considerando desde el diseño básico del sensor virtual y cada una de sus etapas de desempeño, hasta el diseño del montaje físico de los sensores con que se realiza la instrumentación del vehículo. Tomando en cuenta también las etapas de programación y conexión de los elemento electrónicos. Es posible pasar a la construcción del sensor virtual, asegurando que cada etapa tendrá bases sólidas.

Con lo obtenido hasta el momento es posible la construcción completa de una Red Neuronal Artificial que sea implementada como la relación de entrada-salida para el sensor virtual. Se ha realizado con éxito la instrumentación completa del sistema para dar soporte a 11 entradas relacionadas y 4 salidas deseadas, con lo cual se puede garantizar la calidad de los datos de entrenamiento, así como los datos que serán estimados una vez que la Red Neuronal haya sido entrenada de forma correcta.

Capítulo 4

Arquitectura y entrenamiento de la Red Neuronal Artificial

En el capítulo 2 se habló del diseño general del sensor virtual, en él se puede observar que una parte importante durante la construcción del sensor virtual es la técnica que se utiliza para vincular las variables relacionadas con la variable deseada; es decir, un modelo matemático que, dados los valores de entrada, entregue como resultado la salida deseada. Durante este trabajo de tesis se utiliza la técnica basada en Redes Neuronales Artificiales (RNA), la cual fue presentada en el capítulo 2.

En el presente capítulo se presenta la arquitectura que toma la RNA que será implementada como modelo matemático del sensor virtual, también se muestra la forma en que se comporta dicha red en su propagación hacia enfrente, el modo en que se ha entrenado y finalmente los valores de la RNA en su forma totalmente entrenada.

4.1. Arquitectura de la Red Neuronal Artificial

La arquitectura de RNA que se utiliza para el sensor virtual es 11-20-4. Lo cual quiere decir que se cuenta con un total de 3 capas (de entrada, oculta y de salida), con 11 neuronas en la capa de entrada, 20 neuronas en la capa oculta y 4 neuronas en la capa de salida.

En la figura 4.1.1 se muestra un diagrama con la arquitectura de la RNA utilizada durante el trabajo de tesis. Se puede observar en color azul la capa de entrada con $X1$, $X2$, . . . , $X11$ como los valores leídos por los sensores y entregados a las neuronas de entrada. La capa de entrada sirve para comunicar la red con el exterior. En color morado se muestran las conexiones que se dan entre la capa de entrada y la capa oculta, cada valor wi, j representa el peso de conexión entre la i-ésima neurona en la capa de entrada con la j-ésima neurona

en la capa oculta, todas las neuronas de la capa de entrada tienen una conexión con cada una de las neuronas en la capa oculta. En la parte central, en color rojo se puede ver la capa oculta. Cada neurona de esta capa recibe 11 valores correspondientes a una conexión con alguna neurona de la capa de entrada multiplicada por el peso de conexión, adicionalmente, al interior de cada neurona se puede observar un valor *bias* que actúa como un dato extra de entrada con un peso de conexión equivalente a la unidad. En color anaranjado se muestran las conexiones entre la capa oculta y la capa de salida, cada valor vi, j representa el peso de la conexión entre la i-ésima neurona de la capa oculta con la j-ésima neurona en la capa de salida, todas las neuronas de la capa oculta tienen una conexión con cada una de las neuronas en la capa de salida. Finalmente, en color amarillo se observa la capa de salida con $S1$, $S2$, . . . , $S4$ como los valores de salida de la RNA. Cada neurona en esta capa tiene como entrada un total de 20 datos correspondientes a cada valor de conexión con una neurona de la capa oculta multiplicado por el peso de conexión, adicionalmente, cada neurona contiene su valor *bias*. La capa de salida comunica el resultado generado por la RNA hacia el exterior.

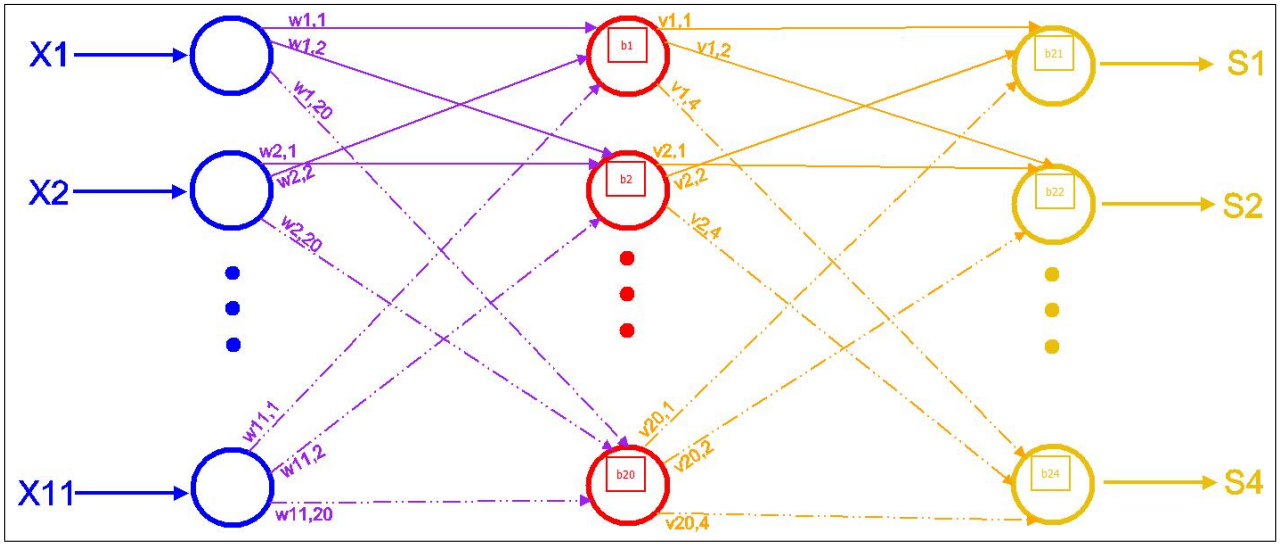

Figura 4.1.1: Arquitectura de la RNA empleada.

La representación matemática de la RNA que se muestra en la figura 4.1.1 está dada por la ecuación 4.1.1.

$$\vec{S} = F_2\left(F_1\left(\vec{X}\cdot W + B_1\right)\cdot V + B_2\right) \tag{4.1.1}$$

Donde:

$\vec{X}$ = Vector de 1 x 11 con los valores de entrada a la RNA.

$\vec{S}$ = Vector de 1 x 4 con los valores de salida de la RNA.

W = Matriz de 11 x 20 que contiene los pesos de las conexiones entre la capa de entrada y la capa oculta.

V = Matriz de 20 x 4 que contiene los pesos de las conexiones entre la capa oculta y la capa de salida.

F_1 = Función de activación para las neuronas de la capa oculta, en este caso la función tangente hiperbólica.

F_2 = Función de activación para las neuronas de la capa de salida, en este caso la función identidad.

$\vec{B_1}$ = Vector de 1 x 20 con los valores de los *bias* en las neuronas de la capa oculta.

$\vec{B_2}$ = Vector de 1 x 4 con los valores de los *bias* en las neuronas de la capa de salida.

4.2. Entrenamiento de la Red Neuronal Artificial

Entrenar una RNA es realizar ajustes en el valor de los pesos y *bias* hasta llegar al conjunto de valores tales que, para cada conjunto de entradas relacionadas, el modelo de RNA entregue el valor de salida deseada. Para llevar a cabo dicho proceso de entrenamiento es necesario recabar el conjunto de entradas deseadas y salidas relacionadas. A partir de este punto, entiéndase por **secuencia** a la lectura sincronizada de la totalidad de sensores expuestos en la etapa de instrumentación del sistema, mientras se realiza un conjunto de movimientos variados que emulan el comportamiento de un vehículo cuando éste es sometido a aceleración, frenado, curva (o cambio en la dirección) y algún tipo de carga estática en distinto orden.

Para llevar a cabo el proceso de entrenamiento, primeramente se realizan distintas secuencias para recabar bancos de datos de entrenamiento, cada banco de datos tiene un total de 15 mil datos de los cuales, 11 mil son datos de entradas relacionadas y 4 mil son datos de salida deseada, esto es, mil datos por cada entrada y mil datos por cada salida. Durante cada secuencia se utiliza el trabajo desarrollado hasta el momento, ya sea de puesta en marcha en los sensores, programación, banco de pruebas, etc. Por lo cual es posible realizar cada secuencia y posteriormente almacenar los datos resultantes en un archivo de excel.

Tomando en cuenta que los 11 valores leídos por los distintos sensores serán llevados a la capa de entrada, mientras que los 4 valores obtenidos mediante la capa de salida serán considerados como la salida del sensor virtual, durante el entrenamiento se implementa una codificación que sirve como apoyo, relacionando cada valor de entrada de la RNA con alguna variable de entrada proporcionada por los sensores y cada valor de salida con algún valor variable del área de contacto de alguna de las llantas del vehículo. En la tabla 4.2.1 se muestra la relación correspondiente a la codificación entre el área de contacto en cada llanta del

vehículo, asignando un nombre corto a cada llanta (o cuadrante) y relacionando cada una con alguna variable de salida de la RNA. Cabe aclarar que el nombre corto asignado a las llantas solamente representará el área de contacto cuando se haga evidente, en caso contrario representa la llanta o el cuadrante del vehículo en cuestión. En la tabla 4.2.2 se muestra la codificación entre las variables entregadas por los sensores, su nombre corto y la relación con las variables de entrada a la RNA.

Llanta del vehículo	Nombre corto	Variable en RNA
Delantera Derecha	DD	S1
Delantera Izquierda	DI	S2
Trasera Derecha	TD	S3
Trasera Izquierda	TI	S4

Tabla 4.2.1: Codificación implementada para salidas de la RNA durante el entrenamiento.

Variable del sensor	Nombre corto	Variable en RNA
Tiempo en alto	PWM	X1
Ángulo de cabeceo	PITCH	X2
Ángulo de alabeo	ROLL	X3
Aceleración en z de DD	ACC1	X4
Aceleración en z de DI	ACC2	X5
Aceleración en z de TD	ACC3	X6
Aceleración en z de TI	ACC4	X7
Deformación en la suspensión DD	ADC1	X8
Deformación en la suspensión DI	ADC2	X9
Deformación en la suspensión TD	ADC3	X10
Deformación en la suspensión TI	ADC4	X11

Tabla 4.2.2: Codificación implementada para entradas de la RNA durante el entrenamiento.

Una vez culminando la etapa de construcción de bancos de datos por medio de secuencias, se pasa al entrenamiento de la RNA. Empleando el *software* Matlab se importan las mil muestras del banco de datos desde el archivo de excel hacia el espacio de trabajo propio de Matlab, posteriormente, por medio de la herramienta **nftool** se asignan 700 muestras para el

entrenamiento de la RNA, 150 muestras para su validación y 150 muestras para la prueba de la RNA. Con la misma herramienta se indica la arquitectura de la RNA deseada y finalmente se manda a entrenamiento. Después de culminar el proceso, el resultado del entrenamiento de una RNA por medio de la herramienta **nftool** en Matlab es un objeto que contiene los valores de los pesos y *bias*. Adicionalmente, se dispone de las gráficas de la evolución de los errores de entrenamiento, validación y prueba, entre otras.

4.3. Red Neuronal Artificial Entrenada

Como se menciona en la sección anterior, el resultado del entrenamiento de la RNA en Matlab es un objeto con los valores de los pesos y *bias* conjuntamente con las gráficas de errores, regresión lineal y estado de los elementos de entrenamiento. En esta sección se muestra el resultado del entrenamiento de la RNA que será implementada en el sensor virtual.

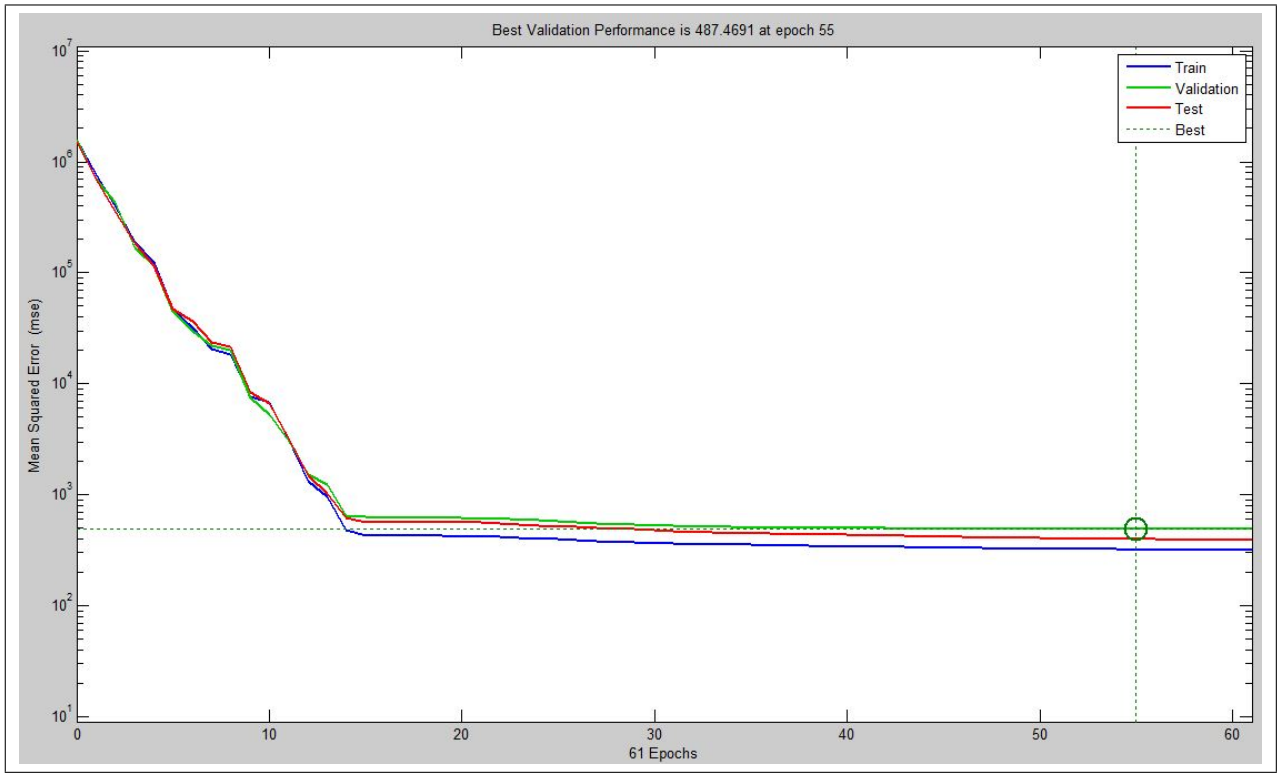

Figura 4.3.1: Gráfica de desempeño de la RNA entrenada.

En la figura 4.3.1 se muestra la gráfica de errores obtenidos en el proceso de entrenamiento, se trata de una gráfica a escala semi-logarítmica con las épocas de entrenamiento en la escala normal y el valor del error cuadrático medio en la escala logarítmica. El error de entrenamiento se representa mediante la línea en color azul, el error de validación se representa con la línea

en color verde y el error de prueba se representa mediante la línea en color rojo. También se pueden observar un par de líneas punteadas que presentan en su intersección el mejor valor de desempeño en la validación.

De forma general, en la gráfica de la figura 4.3.1 se puede observar que los errores tienden a permanecer estables después de un cierto número de ciclos. Es por ello que se puede concluir que: "*El aprendizaje ha terminado con éxito, dado que la red ha sido capaz de extraer las características del problema, alcanzando un buen nivel de generalización*" [15].

Ya que el aprendizaje fue concluido de forma correcta, los valores de los pesos y *bias* de la RNA obtenida pueden ahora ser exportados para su validación final. Los valores resultantes del entrenamiento de la RNA en Matlab son:

$$\mathbf{W} = \begin{bmatrix} -1.97 & -2.44 & -2.29 & 0.26 & -0.04 & -0.14 & 0.38 & 0.49 & -2.08 & 0.47 & 0.34 \\ -2.78 & 0.87 & 1.28 & -0.11 & -0.14 & -0.28 & -0.07 & 2.25 & -1.04 & 0.35 & 0.16 \\ 0.59 & 2.87 & -1.21 & 0.28 & -0.31 & -0.09 & -0.07 & 4.75 & -0.94 & -0.38 & 0.86 \\ 3.17 & -2.03 & -0.99 & 0.13 & 0.05 & 0.28 & 0.25 & -3.08 & 1.15 & 0.01 & -0.61 \\ -0.89 & 0.99 & 1.46 & -0.88 & -0.32 & 0.06 & -0.89 & 0.13 & -1.93 & 1.85 & -0.81 \\ -0.61 & -1.50 & -2.74 & 0.30 & -1.14 & 0.24 & 2.16 & 3.03 & -2.25 & -0.79 & 0.09 \\ 2.73 & -6.57 & 0.98 & 0.03 & -0.17 & 0.09 & 0.17 & 0.83 & -3.18 & -2.43 & 0.76 \\ -1.11 & 0.91 & 0.83 & -0.08 & 0.11 & -0.13 & -0.23 & -1.12 & 0.70 & 0.65 & -0.58 \\ 3.67 & -6.86 & 0.54 & 0.15 & 0.08 & 0.09 & 0.05 & 2.27 & -5.09 & -2.89 & 0.49 \\ -1.12 & 0.80 & 0.51 & 0.01 & 0.03 & 0.04 & 0.00 & 0.36 & -1.15 & 0.44 & -0.07 \\ 1.51 & 0.33 & 0.92 & 0.23 & 0.19 & -0.01 & -0.11 & -0.94 & -1.05 & 0.29 & -0.48 \\ 0.76 & -1.97 & -2.66 & -0.59 & -0.57 & -0.09 & 0.80 & 1.89 & -0.29 & 0.35 & 1.33 \\ -0.97 & -0.89 & 0.52 & -0.05 & 0.02 & -0.11 & -0.19 & -1.67 & 2.21 & 0.00 & -0.47 \\ -0.07 & 0.06 & -0.60 & 0.06 & 0.28 & 0.03 & -0.03 & -3.42 & 1.88 & -0.01 & -0.30 \\ -3.91 & 2.09 & -0.02 & -0.19 & -1.58 & -0.44 & 0.47 & 2.03 & 6.35 & 2.61 & 0.08 \\ 1.34 & 3.10 & 2.17 & 0.92 & -2.46 & 0.93 & 2.10 & 1.43 & -0.20 & -0.11 & -1.28 \\ 0.68 & 0.71 & -0.05 & 0.08 & -0.32 & -0.20 & 0.04 & 3.70 & -1.38 & 0.25 & 0.22 \\ -3.37 & 0.91 & 3.50 & -0.72 & -0.51 & -0.56 & 0.12 & 3.44 & -1.47 & 6.36 & -1.25 \\ 1.96 & 5.15 & 5.59 & -0.54 & 2.89 & 0.56 & 1.42 & 1.44 & 3.63 & 0.85 & -3.37 \\ 1.57 & 0.51 & -1.00 & 0.07 & 0.05 & -0.01 & 0.19 & -1.37 & 1.34 & -0.26 & -0.08 \end{bmatrix}^{\mathbf{T}}$$

$$\mathbf{V} = \begin{bmatrix} -0.23 & -0.09 & -1.02 & -0.27 \\ 0.43 & 0.01 & 2.30 & 0.33 \\ -0.26 & 0.17 & 0.20 & 0.41 \\ 0.25 & 0.04 & 1.69 & 0.21 \\ 0.17 & 0.03 & 0.07 & 0.20 \\ -0.04 & -0.08 & -0.33 & -0.07 \\ 1.39 & 0.23 & -0.86 & -0.37 \\ -0.74 & -0.12 & -1.71 & -0.32 \\ -1.35 & -0.68 & 0.69 & -0.17 \\ 0.67 & 1.86 & 1.79 & 1.31 \\ -0.77 & 0.26 & -0.04 & 0.46 \\ -0.30 & 0.23 & -0.02 & 0.11 \\ -0.24 & 1.30 & 1.01 & 1.10 \\ 2.31 & -1.11 & -1.97 & -2.32 \\ -0.14 & -0.26 & 0.16 & -0.06 \\ -0.03 & -0.01 & -0.22 & -0.05 \\ 1.74 & 0.41 & -1.44 & -1.09 \\ 0.22 & -0.37 & 0.13 & -0.33 \\ -0.06 & -0.01 & -0.10 & 0.01 \\ 0.53 & 1.16 & 1.98 & 1.39 \end{bmatrix} \qquad \vec{B_1} = \begin{bmatrix} -2.82 \\ 0.22 \\ -1.41 \\ -0.23 \\ 1.04 \\ -0.06 \\ 1.19 \\ 0.14 \\ 1.26 \\ -0.36 \\ -0.06 \\ -0.59 \\ 0.32 \\ 1.49 \\ -5.11 \\ -3.36 \\ -1.74 \\ -6.01 \\ 3.10 \\ 0.38 \end{bmatrix}^T$$

$$\vec{B_2} = \begin{bmatrix} -0.34 & 0.03 & -0.52 & -0.04 \end{bmatrix}$$

Teniendo los valores de los pesos de las conexiones y *bias*, es posible reconstruir o exportar la RNA hacia cualquier plataforma que pueda obtener los valores de entrada proporcionados por los sensores, hacer las operaciones de las funciones de propagación de cada capa y entregar a la salida el valor de la variable deseada.

Capítulo 5

Implementación y validación del sensor virtual

En el presente capítulo se presentan los resultados obtenidos mediante la validación del sensor virtual. En una primera sección se explica cómo se implementa la RNA exportada con cada acondicionamiento requerido y la manera en que funcionará. En la segunda sección se trata la forma en que se realiza el experimento de validación y los resultados que arrojan dichos experimentos. Con el apoyo de las gráficas de valores reales tomados con la cámara sobrepuestos a los valores estimados con el sensor virtual y una tabla de errores se llega a la validación final del sensor virtual.

5.1. Implementación del sensor virtual

La plataforma experimental sobre la cual se realiza la implementación del sensor virtual ha sido presentada en la parte final del capítulo 3. De forma general, se implementa un sensor virtual con 11 entradas relacionadas a 4 salidas deseadas. Las 11 entradas relacionadas son leídas desde los 11 sensores varios (presentados con anterioridad) con que se instrumentó el vehículo, mientras que las 4 salidas deseadas son proporcionadas por la cámara con que se instrumentó el banco de pruebas. La técnica utilizada para vincular de forma matemática las entradas relacionadas con las salidas deseadas es la basada en Redes Neuronales Artificiales, cuyo entrenamiento fue realizado en el capítulo 4. Como resultado del entrenamiento se tiene la arquitectura total de la RNA incluyendo número de capas, neuronas por capa, funciones de activación para cada capa, pesos de conexión y *bias* de cada neurona.

Para poder implementar la RNA en un microcontrolador es necesario apoyarse de los programas desarrollados en el capítulo 3 con el fin de tomar las mediciones de entrada proporcionadas por los sensores, posteriormente se aplica procesamiento de datos para

introducirlos en la RNA y finalmente se entrega como salida el área de contacto en cada llanta. Los esquemas eléctricos son los desarrollados en las figuras 3.4.2 y 3.4.3.

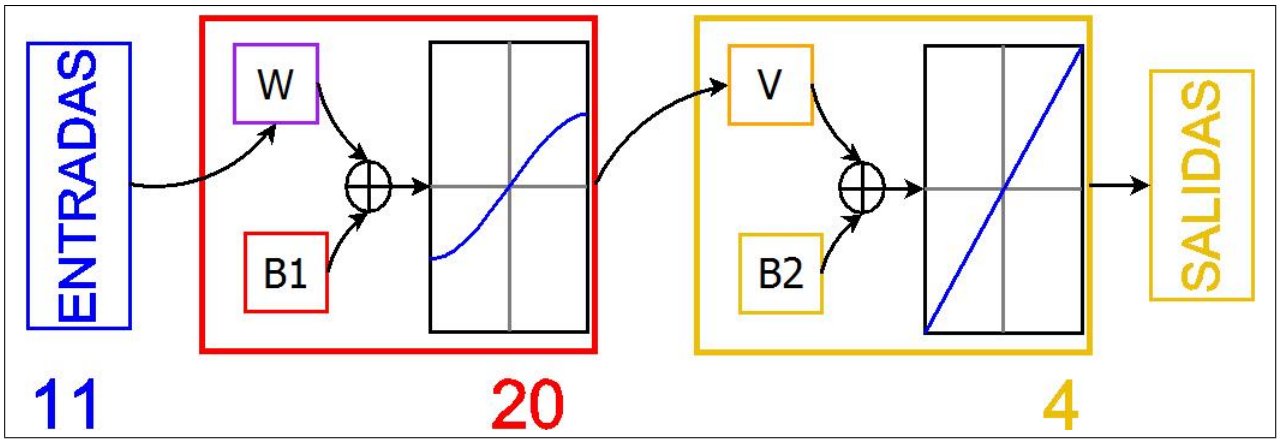

Figura 5.1.1: Esquema de implementación de la Red Neuronal Artificial.

En la figura 5.1.1 se observa el esquema de implementación para la Red Neuronal Artificial. De lado izquierdo puede notarse el bloque de entradas compuesto por 11 elementos, dichos elementos son procesados mediante un mapeo que cambiará el rango de mediciones entregadas por los distintos sensores a un rango de ±1, esto debido a que se utilizará una función de forma sigmoidal dentro de las funciones de activación neuronal. De forma general, si una variable o conjunto de variables x quiere ser mapeado dentro de un intervalo para la variable y, entonces el escalado o normalización viene dado por la ecuación 5.1.1.

$$\frac{y - y_{min}}{y_{max} - y_{min}} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.1.1}$$

Donde:

y = Variable normalizada.

y_{max} = Valor máximo requerido para la variable normalizada.

y_{min} = Valor mínimo requerido para la variable normalizada.

x = Variable a normalizar.

x_{max} = Valor máximo del rango en que se encuentra x.

x_{min} = Valor mínimo del rango en que se encuentra x.

Tomando el intervalo deseado para la variable y de [-1 , 1], la ecuación 5.1.1 de normalización para el caso particular de la implementación de nuestra RNA tomará la forma de la ecuación 5.1.2.

$$y = 2\left[\frac{x - x_{min}}{x_{max} - x_{min}}\right] - 1 \tag{5.1.2}$$

Donde los valores máximos y mínimos para el vector x son:

x_{min} =[42239 ; -3.58 ; -3.31 ; 259 ; 290 ; -247 ; -215 ; 825 ; 1481 ; 793 ; 690]

x_{max} =[78196 ; 10.06 ; 2.08 ; 276 ; 312 ; -219 ; -190 ; 3060 ; 4033 ; 2084 ; 1439]

con $x(i)$ como se propuso en la tabla 4.2.2.

Los valores de entrada normalizados son entregados a la RNA obtenida en el entrenamiento con los valores $\mathbf{W}$, $\mathbf{V}$, $\vec{B}_1$ y $\vec{B}_2$ presentados anteriormente. En el caso de las funciones de activación F_1 (función tangente hiperbólica) y F_2 (función identidad). Si n es el valor a evaluar o valor de entrada, entonces las funciones F_1 y F_2 vienen dadas por las ecuaciones 5.1.3 y 5.1.4, respectivamente.

$$F_1(n) = \frac{2}{1 + \exp\left(-2 * n\right)} - 1 \tag{5.1.3}$$

$$F_2(n) = n \tag{5.1.4}$$

Las salidas entregadas por la RNA deben seguir el proceso inverso a la normalización empleada a los datos de entrada, salvo que ahora se mapean al rango en que se encuentran las salidas deseadas, es preciso indicar en este punto que los rangos máximos y mínimos para las entradas y salidas de la RNA entrenada son tomados de los datos de entrenamiento. Para el mapeo inverso de salida, se toma la ecuación 5.1.1 con el rango de la variable y de [-1 , 1] y s como la salida deseada, obteniendo como resultado la ecuación 5.1.5.

$$s = (s_{max} - s_{min})\left[\frac{y + 1}{2}\right] + s_{min} \tag{5.1.5}$$

Donde los valores máximos y mínimos para el vector s son:

s_{min} =[171.829 ; 721.508 ; 486.977 ; 740.559]

s_{max} =[1790.336 ; 1845.720 ; 2269.617 ; 1826.922]

con $s(i)$ como se propuso en la tabla 4.2.1.

Al mapear los valores de salida entregados por la RNA hacia el rango de salida deseada obtenida en las secuencias de entrenamiento, se obtiene como resultado la salida estimada por el sensor virtual, culminando así la etapa de implentación.

5.2. Validación del sensor virtual

Con el fin de llevar a cabo la validación del sensor virtual construido se realizan distintas secuencias en la cuales se obtiene de forma sincronizada el valor del área de contacto en cada llanta estimado por el sensor virtual y la medición real tomada por medio de la cámara, posteriormente se realiza un proceso de comparación mediante gráficas y cálculos del error entre la estimación del sensor virtual y la medición real.

Dado que la tarea del sensor virtual es estimar el área de contacto cuando el vehículo es sometido a distintos tipos de movimiento (aceleración, frenado, curva o cambio de dirección y pesos en el auto estático). Durante cada experimento de validación, se realizan secuencias para tomar un muestreo total de dos mil datos, de los cuales, mil datos son los valores estimados por el sensor virtual y mil son los valores reales obtenidos con la cámara. Los resultados obtenidos durante las secuencias de validación son presentados mediante las gráficas 5.2.1 a la 5.2.4. De igual forma, en la tabla 5.2.1 se muestra una relación con los valores importantes del área de contacto en cada llanta del vehículo, los errores de estimación obtenidos y el porcentaje sobre la variación del área de contacto que representan dichos errores.

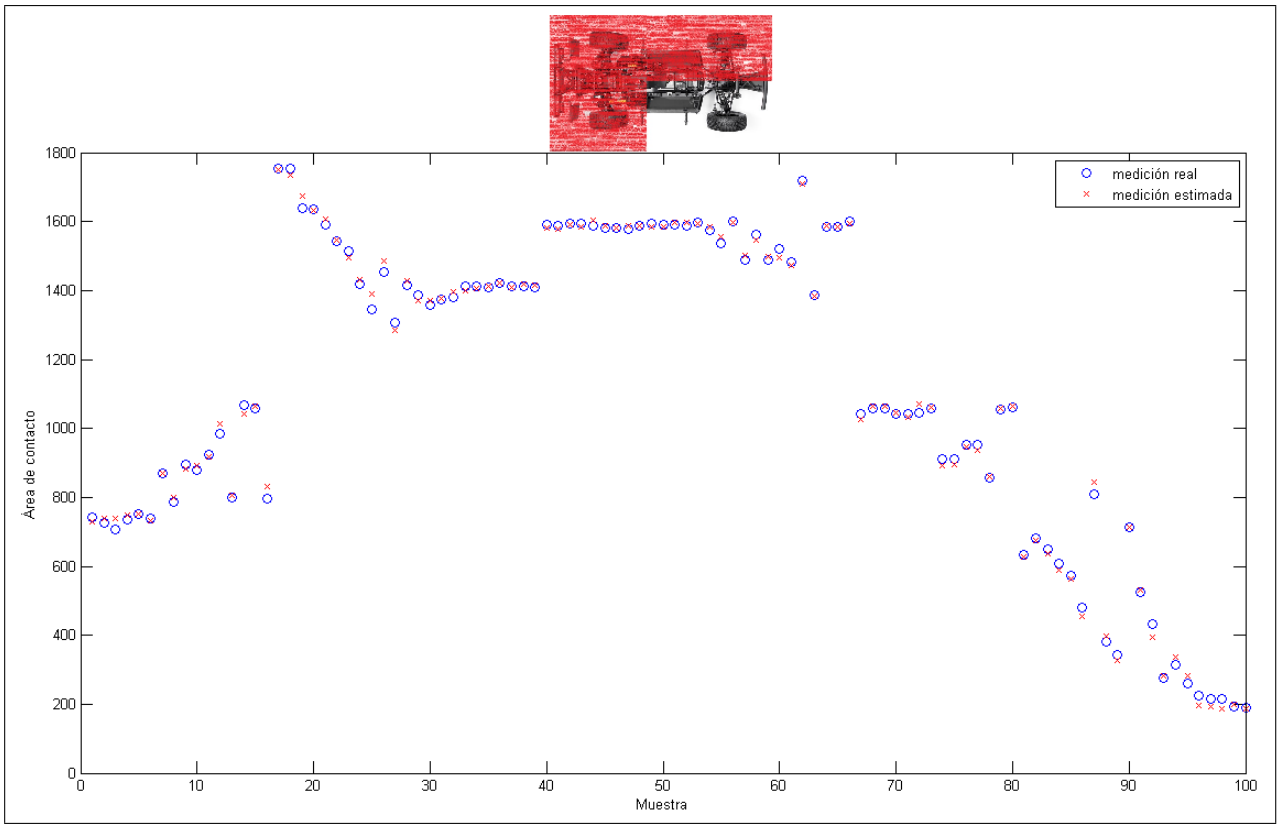

Figura 5.2.1: Área de contacto estimada para la llanta delantera derecha.

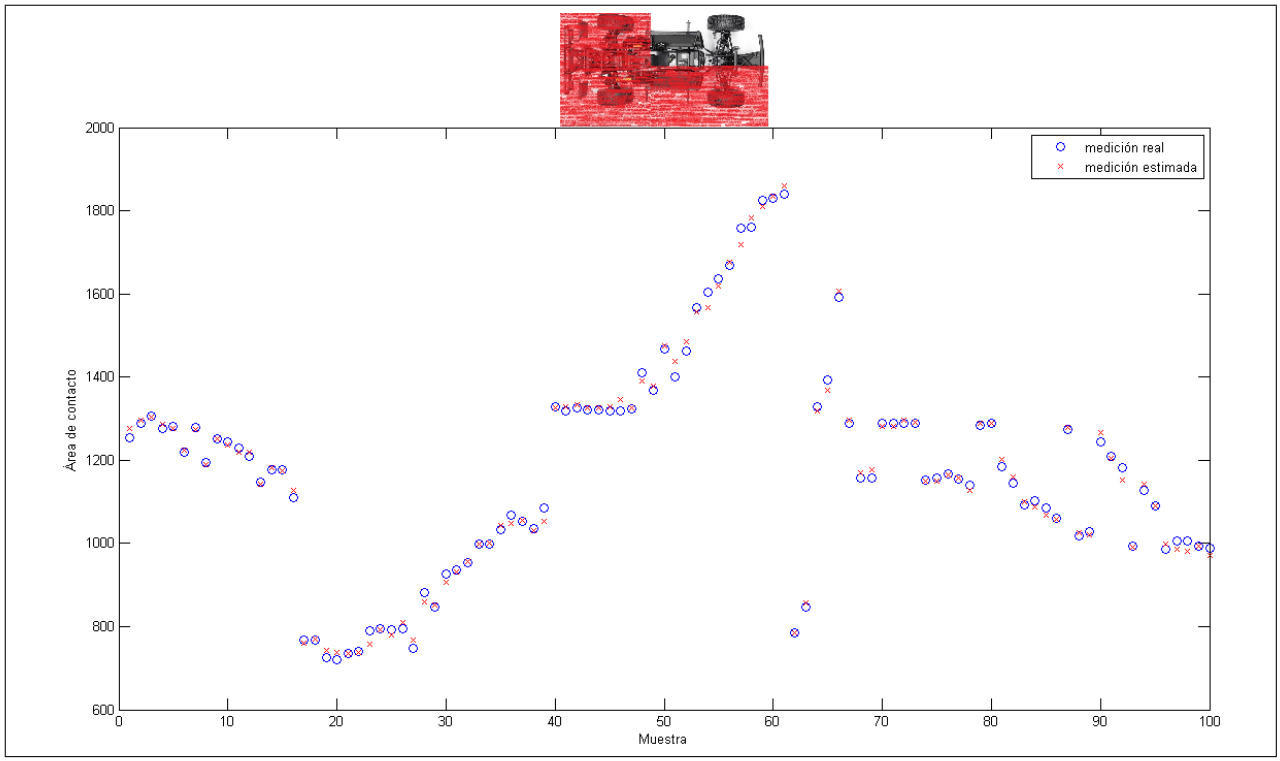

Figura 5.2.2: Área de contacto estimada para la llanta delantera izquierda.

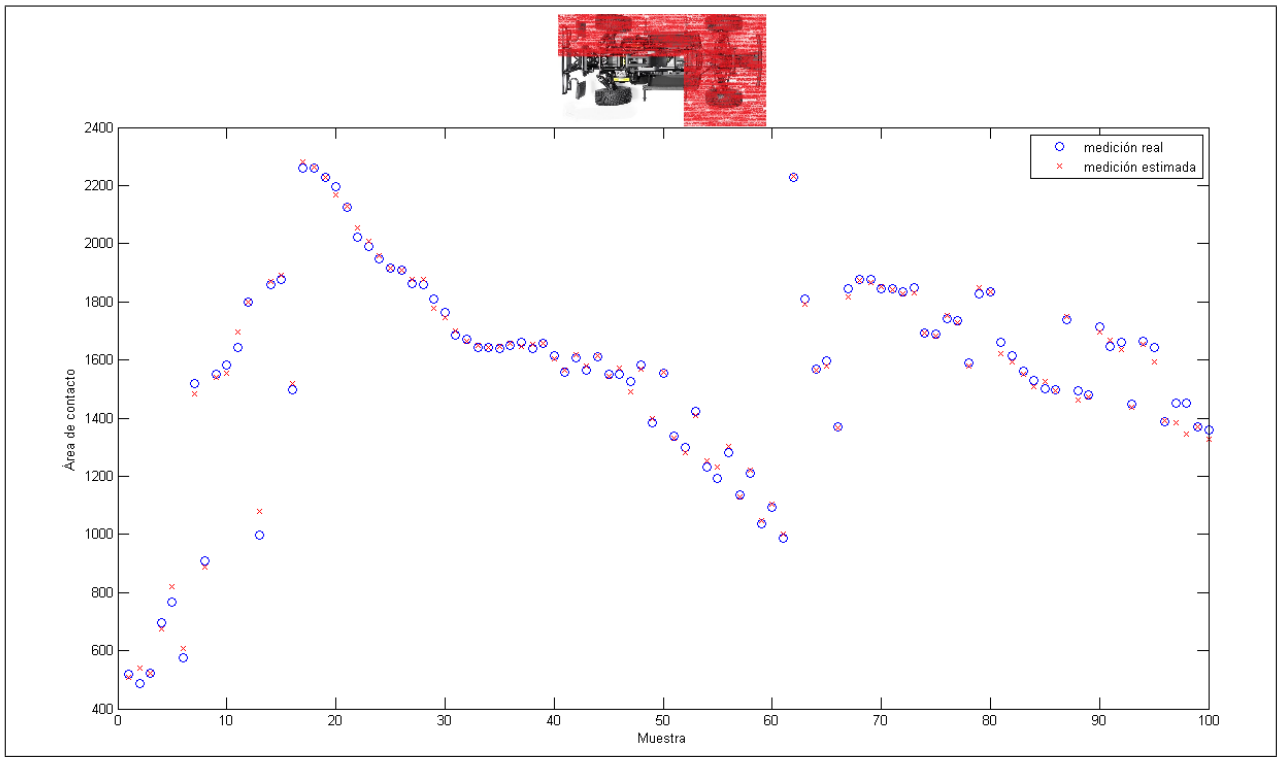

Figura 5.2.3: Área de contacto estimada para la llanta trasera derecha.

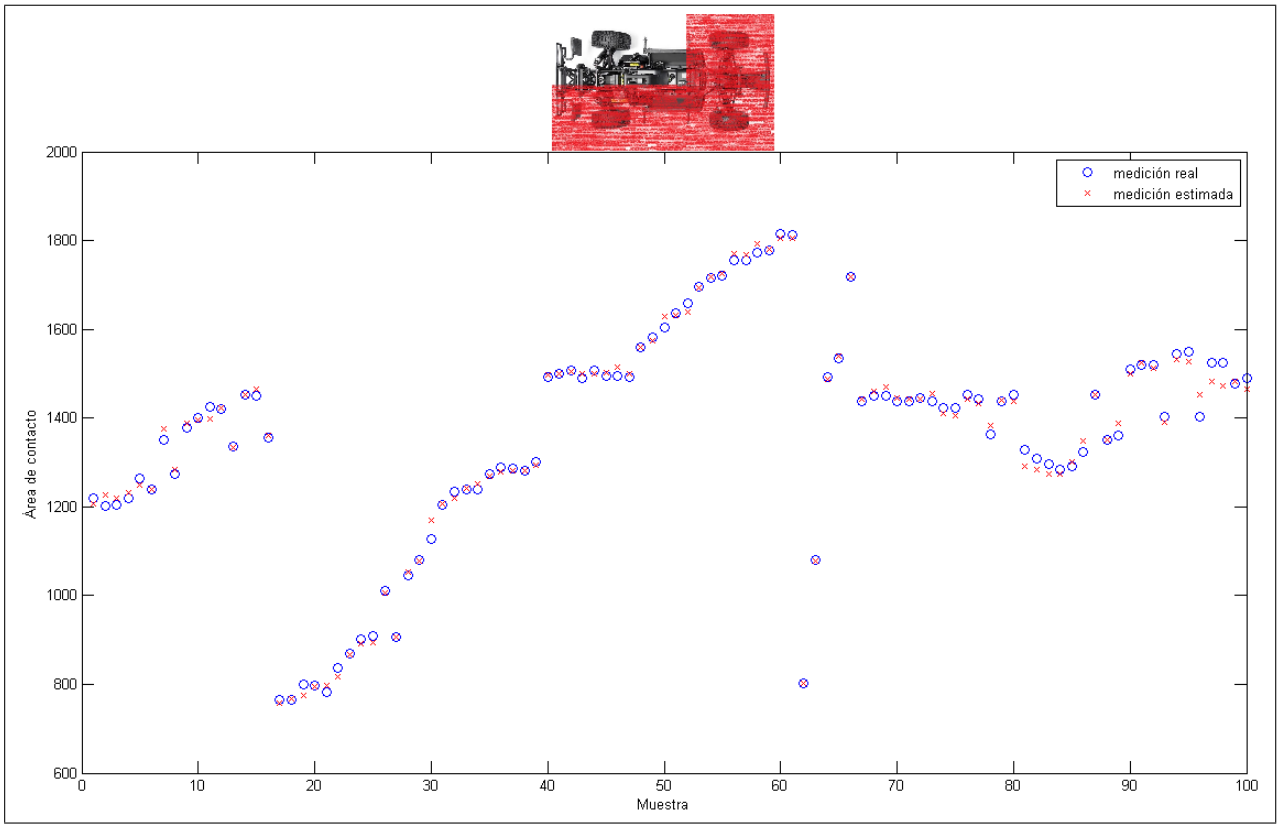

Figura 5.2.4: Área de contacto estimada para la llanta trasera izquierda.

En la figura 5.2.1 se muestra la gráfica de estimaciones del sensor virtual para el área de contacto de la llanta delantera derecha comparándola con la medición real tomada con la cámara, en la figura 5.2.2 se muestra la gráfica de estimaciones del sensor virtual para el área de contacto en la llanta delantera izquierda igualmente comparada con la medición real de la cámara, en la figura 5.2.3 se pueden observar los mismos elementos para el caso del área de contacto en la llanta trasera derecha, finalmente en la figura 5.2.4 se observan los resultados obtenidos para la llanta trasera izquierda.

En todos los casos se presenta un total de cien muestras, lo cual representa un sub-muestreo del 10 % de los mil datos obtenidos durante las secuencias de validación. Todas las gráficas muestran el número de sub-muestra contra el área de contacto en milímetros cuadrados. Los marcadores rojos en forma de cruz representan las estimaciones obtenidas por el sensor virtual para cada sub-muestra, mientras que los marcadores azules en forma de círculo representan el valor obtenido mediante la medición real para la sub-muestra correspondiente.

En la tabla 5.2.1 se muestra la relación de resultados obtenidos para el rango del área de contacto (valor máximo y mínimo), la variación que representa dicho rango del área de contacto, la media del error cometido por la estimación del sensor virtual, la desviación

estándar del error de estimación y el porcentaje de error representado con respecto al rango del área de contacto para cada llanta.

	Llanta DD	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1790.33	1845.72	2269.61	1826.92
Mínimo del área de contacto	171.83	721.51	482.97	740.55
Variación del área de contacto	1618.5	1124.21	1786.64	1086.37
Media del error de estimación	12.5	11.4	17.3	12.8
Desviación estándar del error	11.2	9.9	17.6	11.5
Porcentaje de error	1.46 %	1.89 %	1.95 %	2.23 %

Tabla 5.2.1: Errores obtenidos durante la validación del sensor virtual.

De la misma manera en que se realizó el experimento anterior, se llevan a cabo diez experimentos extras para obtener valores de validación variados por medio de distintas secuencias. Los resultados obtenidos para dichos experimentos son mostrados por medio de las tablas 5.2.2 a la 5.2.11. Finalmente, en la tabla 5.2.12 se presenta el promedio de los valores obtenidos durante los 10 experimentos.

1	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1325.78	1245.01	1605.13	1095.58
Mínimo del área de contacto	379.25	325.78	533.2	521.97
Variación del área de contacto	946.53	919.23	1071.93	573.61
Media del error de estimación	11.9	12.9	16.0	15.6
Desviación estándar del error	17.3	10.5	19.3	11.3
Porcentaje de error	3.08	2.55	3.29	4.69

Tabla 5.2.2: Errores obtenidos durante el experimento de validación 1.

2	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1233.08	1530.23	2069.05	1984.16
Mínimo del área de contacto	455.01	695.33	743.75	846.97
Variación del área de contacto	778.07	834.9	1325.3	1137.19
Media del error de estimación	17.3	11.4	17.3	12.8
Desviación estándar del error	12.8	9.9	17.6	11.5
Porcentaje de error	3.87	2.55	2.63	2.14

Tabla 5.2.3: Errores obtenidos durante el experimento de validación 2.

3	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	983.27	1230.85	1568.43	1324.22
Mínimo del área de contacto	325.84	459.68	225.43	302.76
Variación del área de contacto	657.43	771.17	1343	1021.46
Media del error de estimación	12.3	17.3	15.6	17.1
Desviación estándar del error	8.7	11.1	11.2	15.9
Porcentaje de error	3.19	3.68	2.00	3.23

Tabla 5.2.4: Errores obtenidos durante el experimento de validación 3.

4	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1832.78	1430.06	2017.72	1989.32
Mínimo del área de contacto	355.72	493.82	448.64	328.87
Variación del área de contacto	1477.06	936.24	1569.08	1660.45
Media del error de estimación	17.7	11	12.1	11.3
Desviación estándar del error	12.8	6.4	17.9	14.8
Porcentaje de error	2.06	1.86	1.91	1.57

Tabla 5.2.5: Errores obtenidos durante el experimento de validación 4.

5	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	835.54	993.95	1463.87	1097.03
Mínimo del área de contacto	178.96	421.32	401.53	367.89
Variación del área de contacto	656.58	572.63	1062.34	729.14
Media del error de estimación	11.1	9.7	15.3	11.5
Desviación estándar del error	10.7	11.3	16.1	9.7
Porcentaje de error	3.32	3.67	2.96	2.91

Tabla 5.2.6: Errores obtenidos durante el experimento de validación 5.

6	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1092.65	1710.21	1968.12	1856.07
Mínimo del área de contacto	183.79	465.88	422.31	512.25
Variación del área de contacto	908.86	1244.33	1545.81	1343.82
Media del error de estimación	17.3	12.3	15.6	13.7
Desviación estándar del error	11.4	10.8	17.6	16.9
Porcentaje de error	3.16	1.86	2.15	2.28

Tabla 5.2.7: Errores obtenidos durante el experimento de validación 6.

7	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1693.52	1178.19	2071.45	2215.33
Mínimo del área de contacto	321.15	461.04	426.45	592.12
Variación del área de contacto	1372.37	717.15	1645	1623.21
Media del error de estimación	17.7	10.6	11.7	16.9
Desviación estándar del error	17.1	9.7	15.2	18.2
Porcentaje de error	2.54	2.83	1.64	2.16

Tabla 5.2.8: Errores obtenidos durante el experimento de validación 7.

8	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	798.33	1482.97	1536.12	1665.78
Mínimo del área de contacto	175.46	653.02	425.6	602.66
Variación del área de contacto	622.87	829.95	1110.52	1063.12
Media del error de estimación	11.1	12.5	14.3	15.1
Desviación estándar del error	11.2	10.1	12.1	9.7
Porcentaje de error	3.58	2.72	2.38	2.33

Tabla 5.2.9: Errores obtenidos durante el experimento de validación 8.

9	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1892.37	1732.19	2109.56	1973.21
Mínimo del área de contacto	224.35	463.01	480.11	463.32
Variación del área de contacto	1668.02	1269.18	1629.45	1509.89
Media del error de estimación	17.1	13.5	16.2	15.1
Desviación estándar del error	12.8	11.2	17.1	17.2
Porcentaje de error	1.79	1.95	2.04	2.14

Tabla 5.2.10: Errores obtenidos durante el experimento de validación 9.

10	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1730.15	1965.64	2143.97	2043.12
Mínimo del área de contacto	232.74	430.51	412.18	501.42
Variación del área de contacto	1497.41	1535.13	1731.79	1541.7
Media del error de estimación	14.4	12.6	17.1	14.3
Desviación estándar del error	10.8	11.9	15.4	17.2
Porcentaje de error	1.68	1.60	1.88	2.04

Tabla 5.2.11: Errores obtenidos durante el experimento de validación 10.

Promedio	**Llanta DD**	**Llanta DI**	**Llanta TD**	**Llanta TI**
Máximo del área de contacto	1341.747	1449.93	1855.342	1724.382
Mínimo del área de contacto	283.227	486.939	451.92	504.023
Variación del área de contacto	1058.52	962.991	1403.422	1220.359
Media del error de estimación	14.79	12.38	15.12	14.34
Desviación estándar del error	12.56	10.29	15.95	14.24
Porcentaje de error	2.83	2.53	2.29	2.55

Tabla 5.2.12: Promedio de valores obtenidos en los experimentos de validación.

De forma general, el error obtenido por la estimación del sensor virtual con respecto al área de contacto real es aceptable en todos los casos al compararlos con el rango en que se mueven las salidas deseadas ya que el valor promedio en el porcentaje de error mostrado en la tabla 5.2.12 para cada llanta resulta ser menor al 3 %.

Basándonos en el resultado de error con mayor magnitud podemos darnos cuenta que, dicho error representa menos del 5 % del valor de variación total del área de contacto. Por lo cual se puede concluir que el sensor virtual ha sido construido de forma correcta, culminando así la etapa de validación y habilitando la posibilidad de ser implementado para cualquier aplicación que requiera un factor de tolerancia mayor o igual al 5 %. Adicionalmente, el sensor virtual puede ser calibrado cuando sea requerido debido a cambios en el vehículo tales como el desgaste de las llantas o algún cambio en la suspensión del vehículo.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

En el presente trabajo se desarrolló el diseño, implementación y validación de un sensor virtual capaz de estimar el área de contacto en las llantas de un vehículo tipo baja, escala 1/5.

Se realizó con éxito el estudio sobre el estado del arte con el cual se evaluaron los distintos métodos y tecnologías disponibles para iniciar el diseño del sensor virtual.

Se diseñó un sensor virtual basado en datos, tomando en cuenta cada etapa que conlleva el proceso general de diseño de un sensor virtual. Se realizó la elección de datos medibles que están relacionados con la salida deseada, posteriormente se eligieron los sensores mejor calificados para medir dichas variables relacionadas, se planteó la técnica matemática para vincular las variables relacionadas con la salida deseada y la técnica de validación del modelo resultante.

Se llevó a cabo el diseño y construcción de un banco de pruebas que sirvió como apoyo durante las etapas de puesta en marcha de los sensores, construcción de un banco de datos, entrenamiento del sensor virtual y validación del mismo. De igual forma se realizó el diseño de la instrumentación total del sistema, la puesta en marcha de dicha instrumentación incluyendo las conexiones electrónicas y la programación para llegar a la integración de la plataforma experimental sobre la cual se trabajó.

Se usó un algoritmo basado en Redes Neuronales Artificiales con lo cual se implementó el sensor virtual basado en datos. Se presentó la arquitectura de la Red Neuronal Artificial elegida y la forma en que ésta fue entrenada, arrojando como resultado los valores de cada elemento de la RNA entrenada.

Se realizaron experimentos de implementación y validación del sensor virtual. Implementando la RNA en un microcontrolador para estimar el área de contacto mientras que de forma sincronizada se tomaba la comparación con la medición real y el valor del error de estimación. Obteniendo como resultado un sensor virtual confiable con tolerancia menor al 5 %.

Se obtuvo como resultado final un sensor virtual basado en datos capaz de estimar en línea el área de contacto de las llantas de un vehículo.

6.2. Trabajo Futuro

Los siguientes puntos son planteados como tareas que pueden ser consideradas para trabajos futuros, con fines de realizar mejoras y ampliar el desarrollo logrado en el presente trabajo de tesis. Se incluyen también propuestas de aplicación del sensor virtual resultante como principio de nuevas líneas de investigación.

- Reproducir el sensor virtual obtenido con distintos sensores de mejor calidad, en una plataforma distinta; por ejemplo una FPGA.
- Mejorar la velocidad de estimación en línea del sensor virtual obtenido cambiando el sensor del ángulo en la dirección.
- Reproducir el trabajo presente prescindiendo de algunos sensores empleados y encontrar cuántos y cuáles son los sensores mínimos necesarios para construir con éxito el sensor virtual.
- Implementar el sensor virtual como inicio en una etapa de control de área de contacto en las llantas.
- Crear una pista de pruebas con la cual se pueda tener el área de contacto real en línea durante un recorrido considerable para validar el sensor virtual en un panorama distinto.
- Construir un sensor virtual basado en modelo y comparar los resultados obtenidos por ambos sensores.
- Estudiar los efectos en el control de un vehículo al mantener o prescindir de un área de contacto mínima.
- Reproducir el trabajo realizado para una versión en auto real.

Bibliografía

[1] Fortuna, L., Graziani, S., Rizzo, A., & Xibilia, M. G., *Soft sensors for monitoring and control of industrial processes.* Springer. (2007).

[2] Wu, Y., & Luo, X. (2009). *A design of soft sensor based on data fusion.* In 2009 International Conference on Information Engineering and Computer Science (pp. 1-4).

[3] Du, D., Wu, C., Luo, X., & Zuo, X. (2006, October). *Delay time identification and dynamic characteristics study on ANN soft sensor.* In Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on (Vol. 1, pp. 42-45). IEEE.

[4] Solihin, M. I., & Albagul, A. (2006, June). *Development of soft sensor for sensorless automatic gantry crane using RBF neural networks.*, In Cybernetics and Intelligent Systems, 2006 IEEE Conference on (pp. 1-6). IEEE.

[5] Zhenhai, G. (2003, October). *Soft sensor application in vehicle yaw rate measurement based on Kalman filter and vehicle dynamics.* In Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE (Vol. 2, pp. 1352-1354). IEEE.

[6] Hallonborg, U. (1996). *Super ellipse as tyre-ground contact area.* Journal of Terramechanics, 33(3), 125-132.

[7] Komandi, G. (1976). *The determination of the deflection, contact area, dimensions, and load carrying capacity for driven pneumatic tires operating on concrete pavement.* Journal of Terramechanics, 13(1), 15-20.

[8] Grecenko, A. (1995). *Tyre footprint area on hard ground computed from catalogue values.* Journal of terramechanics, 32(6), 325-333.

[9] Palancar, T. C., Terminiello, A. M., & Jorajuría, D. (2001). *Determinación Expeditiva del Área de Contacto Rueda-Suelo en Máquinas Agrícolas.* Universidad Nacional de La Plata.

[10] Ivanov, V. (2010). *Analysis of tire contact parameters using visual processing.* Advances in tribology, 2010.

[11] Castillo, J., Blanca, A. P. D. L., Cabrera, J. A., & Simón, A. (2006). *An optical tire contact pressure test bench.* Vehicle System Dynamics, 44(3), 207-221.

[12] Duchanoy, C. A., Moreno-Armendáriz, M. A., & Cruz-Villar, C. A. (2013, November). *Design of a Damper for a Suspension System via Evolutive Optimization.* In ASME 2013 International Mechanical Engineering Congress and Exposition (pp. V012T13A034-V012T13A034). American Society of Mechanical Engineers.

[13] Duchanoy, C. A. (2012). *Development of an integral dynamic model with 6 subsystems of an off-road vehicle, validation and study of handling and comfort (In Spanish).* Centro de Investigación en Computación, Instituto Politécnico Nacional.

[14] Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., ... & von Zadow, U. (2008). *Multi-touch surfaces: A technical guide.* IEEE Tabletops and Interactive Surfaces, 2.

[15] Viñuela, P. I., & León, I. M. G. (2004). *Redes de neuronas artificiales: un enfoque práctico.* Prentice Hall.

[16] Mouroulis, P., & Macdonald, J. (1997). *Geometrical optics and optical design* (Vol. 39). New York: Oxford University Press.

Apéndice A

Tablas de conexiones en microcontroladores

PIN	Función y/o Conexión
A2	Salida de señal de sincronización.
D6	Rx, conexión a Tx en IMU.
D7	Tx, conexión a Rx en IMU.
F4	Entrada de Timer modo captura.

Tabla A.0.1: Conexiones en microcontrolador maestro para la etapa de construcción.

PIN	Función y/o Conexión
A2	Entrada de señal de sincronización.
B2	Selección de chip, conexión a CS en acelerómetro 1.
B3	Selección de chip, conexión a CS en acelerómetro 2.
B4	Reloj SPI, conexión a SCL en acelerómetros.
B5	Selección de chip creado por defecto, No Conectar.
B6	Entrada SPI, conexión a SDO en acelerómetros.
B7	Salida SPI, conexión a SDA en acelerómetros.
E1	Entrada analógica, conexión a OPAM del sensor de flexión 1.
E2	Entrada analógica, conexión a OPAM del sensor de flexión 2.

Tabla A.0.2: Conexiones en microcontroladores esclavos para la etapa de construcción.

PIN	Función y/o Conexión
B2	Selección de chip, conexión a CS en acelerómetro 1.
B3	Selección de chip, conexión a CS en acelerómetro 2.
C4	Selección de chip, conexión a CS en acelerómetro 3.
C5	Selección de chip, conexión a CS en acelerómetro 4.
B4	Reloj SPI, conexión a SCL en acelerómetros.
B5	Selección de chip creado por defecto, No Conectar.
B6	Entrada SPI, conexión a SDO en acelerómetros.
B7	Salida SPI, conexión a SDA en acelerómetros.
D6	Rx, conexión a Tx en IMU.
D7	Tx, conexión a Rx en IMU.
E1	Entrada analógica, conexión a OPAM del sensor de flexión 1.
E2	Entrada analógica, conexión a OPAM del sensor de flexión 2.
E3	Entrada analógica, conexión a OPAM del sensor de flexión 3.
D3	Entrada analógica, conexión a OPAM del sensor de flexión 4.
F4	Entrada de Timer modo captura.

Tabla A.0.3: Conexiones en microcontrolador para la etapa de validación.

Apéndice B

Programas utilizados en la etapa de construcción

B.1. Microcontrolador maestro

Programa B.1: Código para microcontrolador maestro en la etapa de construcción

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include <float.h>
4  #include <math.h>
5  #include <stdio.h>      /* printf, NULL */
6  #include <stdlib.h>     /* strtof */
7  #include "inc/hw_ints.h"
8  #include "inc/hw_memmap.h"
9  #include "inc/hw_types.h"
10 #include "driverlib/debug.h"
11 #include "driverlib/fpu.h"
12 #include "driverlib/gpio.h"
13 #include "driverlib/interrupt.h"
14 #include "driverlib/pin_map.h"
15 #include "driverlib/rom.h"
16 #include "driverlib/rom_map.h"
17 #include "driverlib/sysctl.h"
18 #include "driverlib/timer.h"
19 #include "driverlib/uart.h"
20 #include "utils/uartstdio.h"
21 #include "driverlib/adc.h"
22 #include "inc/hw_timer.h"
23
24 uint8_t i;
25 uint8_t j;
26 uint8_t k;
27 uint8_t n;
```

```c

const int16_t muestras=1000;
char ini;
char sinc;
float P, R;

char buffer[1000][30];
char Pitch[1000][7], Roll[1000][7];

unsigned long edge_period[1000];
unsigned long value_edge1;
unsigned long value_edge2;
uint8_t edge_num;

void
ConfigureUART(void)
{
  // Enable the GPIO Peripheral used by the UART.
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);


  // Enable UART0
  SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

  // Configure GPIO Pins for UART mode.
  GPIOPinConfigure(GPIO_PA0_U0RX);
  GPIOPinConfigure(GPIO_PA1_U0TX);
  GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

  // Use the internal 16MHz oscillator as the UART clock source.
  UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);

  // Initialize the UART for console I/O.
  UARTStdioConfig(0, 115200, 16000000);
}

//,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
//interrupt
void Timer2IntHandler(void)
{

// Clear the timer interrupt
 TimerIntClear(TIMER2_BASE, TIMER_CAPA_EVENT);
 edge_num ++;
 if(edge_num==1)
{
 value_edge1 =  TimerValueGet(TIMER2_BASE,TIMER_A);
 TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_NEG_EDGE);
}
```

```
if(edge_num==2){
 value_edge2 =  TimerValueGet(TIMER2_BASE,TIMER_A);
 edge_period[n] = value_edge2-value_edge1;
 edge_num =0;
 TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_POS_EDGE);

 TimerDisable(TIMER2_BASE, TIMER_A);
 HWREG(TIMER2_BASE + TIMER_O_TAV) = 0;
 TimerIntDisable(TIMER2_BASE, TIMER_CAPA_EVENT);
}
}//end interrupt
//
  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,


void main(void) {
  SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL| SYSCTL_XTAL_16MHZ|
    SYSCTL_OSC_MAIN)
  //UART initialization
    ConfigureUART();
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
  GPIOPinConfigure(GPIO_PD6_U2RX);
  GPIOPinConfigure(GPIO_PD7_U2TX);
  GPIOPinTypeUART(GPIO_PORTD_BASE, GPIO_PIN_6 | GPIO_PIN_7);

  SysCtlPeripheralEnable(SYSCTL_PERIPH_UART2);
  UARTConfigSetExpClk(UART2_BASE, SysCtlClockGet(), 57600,
                      (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
                       UART_CONFIG_PAR_NONE));
  UARTEnable(UART2_BASE);

//,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
  GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);
  GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, 0x00);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
  GPIOPinConfigure(GPIO_PF4_T2CCP0);
  GPIOPinTypeTimer(GPIO_PORTF_BASE, GPIO_PIN_4);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);
  TimerConfigure(TIMER2_BASE, TIMER_CFG_SPLIT_PAIR|
    TIMER_CFG_A_CAP_TIME_UP);
  TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_POS_EDGE);

  IntEnable(INT_TIMER2A);
  TimerIntDisable(TIMER2_BASE, TIMER_CAPA_EVENT);
  IntMasterEnable();
  TimerIntRegister(TIMER2_BASE, TIMER_A, *Timer2IntHandler);
  TimerDisable(TIMER2_BASE, TIMER_A);

  edge_num =0;
```

```
124   n=0;
125
126   for(k=0;k<muestras;k++)
127   {
128     for(j=0;j<7;j++)
129     {
130       Pitch[k][j]=' ';
131       Roll[k][j]=' ';
132     }
133     for(j=0;j<30;j++)
134     {
135       buffer[k][j]=' ';
136     }
137   }
138
139 while(1)
140 {
141   sinc=UARTgetc();
142   if(sinc=='1')
143   {
144     GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, 0xFF);
145     UARTEnable(UART2_BASE);
146     TimerEnable(TIMER2_BASE, TIMER_A);
147     TimerIntEnable(TIMER2_BASE, TIMER_CAPA_EVENT);
148     while(1)
149     {
150       ini=UARTCharGet(UART2_BASE);
151       if(ini=='#')
152       {
153         break;
154       }
155     }
156     for(i=0;i<30;i++)
157     {
158       buffer[n][i]=UARTCharGet(UART2_BASE);
159       if(buffer[n][i]=='\r')
160       {
161         break;
162       }
163     }
164
165     GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, 0x00);
166     UARTDisable(UART2_BASE);
167     n++;
168     if (n>muestras-1)
169     {
170       n=0;
171     }
172   }
173
174   if (sinc=='T')
```

```
175   {
176     for(k=0;k<muestras;k++)
177     {
178       i=0;
179       j=0;
180       while(buffer[k][i+j]!='=')
181       {
182         j++;
183       }
184       i=i+j+1;
185       j=0;
186       while(buffer[k][i+j]!=',')
187       {
188         j++;
189       }
190
191       i=i+j+1;
192       j=0;
193       while(buffer[k][i+j]!=',')
194       {
195         Pitch[k][j]=buffer[k][i+j];
196         j++;
197       }
198
199       i=i+j+1;
200       j=0;
201       while(buffer[k][i+j]!='.')
202       {
203         Roll[k][j]=buffer[k][i+j];
204         j++;
205       }
206
207       Roll[k][j]=buffer[k][i+j];
208       j++;
209       Roll[k][j]=buffer[k][i+j];
210       j++;
211       Roll[k][j]=buffer[k][i+j];
212     }
213
214     for(n=0;n<muestras;n++)
215     {
216       UARTprintf(" %u\t",edge_period[n]);
217       if(n==muestras-1)
218       {
219         UARTprintf("\n");
220       }
221     }
222
223     UARTprintf("%s \n", Pitch);
224     UARTprintf("%s \n", Roll);
225     n=0;
```

```
226       for(k=0;k<muestras;k++)
227       {
228         for(j=0;j<7;j++)
229         {
230
231           Pitch[k][j]=' ';
232           Roll[k][j]=' ';
233         }
234         for(j=0;j<30;j++)
235         {
236           buffer[k][j]=' ';
237         }
238       }
239     }     // fin if (sinc=='T')
240 }         //fin while(1)
241
242 }
```

B.2. Microcontroladores esclavos

Programa B.2: Código para microcontroladores esclavos en la etapa de construcción

```
#include <stdbool.h>
#include <math.h>
#include <inttypes.h>//<stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "driverlib/adc.h"

const int16_t muestras=1000;
int16_t z1[1000], z2[1000];
uint32_t adcbuffer1[1000];
uint32_t adcbuffer2[1000];
uint32_t values[4];
uint32_t sinc;

void InitConsole(void)
{
  // Enable GPIO port A which is used for UART0 pins.
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
  GPIOPinConfigure(GPIO_PA0_U0RX);
  GPIOPinConfigure(GPIO_PA1_U0TX);
  // Enable UART0 so that we can configure the clock.
  SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
  // Use the internal 16MHz oscillator as the UART clock source.
  UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
  // Select the alternate (UART) function for these pins.
  GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
  // Initialize the UART for console I/O.
  UARTStdioConfig(0, 115200, 16000000);
}

void InitSPI(void)
{
  // The SSI2 peripheral must be enabled for use.
  SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);

  // Enable Port B
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

  // SETUP SPI CS Pin (to output)
  GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);

  // Set SPI CS to HIGH (active-low)
```

```
48    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0xFF);
49
50    // Configure the pin muxing for SSI2 functions on port B4, B5, B6, and
        B7.
51    GPIOPinConfigure(GPIO_PB4_SSI2CLK);
52    GPIOPinConfigure(GPIO_PB5_SSI2FSS);
53    GPIOPinConfigure(GPIO_PB6_SSI2RX);
54    GPIOPinConfigure(GPIO_PB7_SSI2TX);
55
56    // Configure the GPIO settings for the SSI pins.
57    GPIOPinTypeSSI(GPIO_PORTB_BASE, GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
58                   GPIO_PIN_7);
59    // Configure and enable the SSI port for SPI master mode.
60    SSIConfigSetExpClk(SSI2_BASE, SysCtlClockGet(), SSI_FRF_MOTO_MODE_3,
61                       SSI_MODE_MASTER, 4000000, 8);
62    // Enable the SSI0 module.
63    SSIEnable(SSI2_BASE);
64
65    // Read any residual data from the SSI port.
66    uint32_t scrap;
67    while(SSIDataGetNonBlocking(SSI2_BASE, &scrap));
68  }
69
70  void GetSPI (void)
71  {
72    // Z1_0
73    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
74    SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
75    while(SSIBusy(SSI2_BASE));
76    SSIDataPut(SSI2_BASE, 0x00);
77    SSIDataGetNonBlocking(SSI2_BASE, &values[0]);
78    while(SSIBusy(SSI2_BASE));
79    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
80
81    //  Z1_1
82    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
83    SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
84    while(SSIBusy(SSI2_BASE));
85    SSIDataPut(SSI2_BASE, 0x00);
86    SSIDataGetNonBlocking(SSI2_BASE, &values[1]);
87    while(SSIBusy(SSI2_BASE));
88    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
89
90    //  Z2_0
91    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
92    SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
93    while(SSIBusy(SSI2_BASE));
94    SSIDataPut(SSI2_BASE, 0x00);
95    SSIDataGetNonBlocking(SSI2_BASE, &values[2]);
96    while(SSIBusy(SSI2_BASE));
97    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
```

```
98
99    //  Z2_1
100   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
101   SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
102   while(SSIBusy(SSI2_BASE));
103   SSIDataPut(SSI2_BASE, 0x00);
104   SSIDataGetNonBlocking(SSI2_BASE, &values[3]);
105   while(SSIBusy(SSI2_BASE));
106   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
107
108   //The Z1 value is stored in values[5] and values[4].
109   z1[sinc-1] = ((int16_t)values[1]<<8)|(int16_t)values[0];
110
111   //The Z2 value is stored in values[11] and values[10].
112   z2[sinc-1] = ((int16_t)values[3]<<8)|(int16_t)values[2];
113
114   return;
115 }
116
117 //interrupt
118 void PortAIntHandler(void)
119 {
120   // Clear the timer interrupt
121   GPIOIntClear(GPIO_PORTA_BASE,GPIO_PIN_2);
122   sinc ++;
123   if(sinc==muestras)
124   {
125     GPIOIntDisable(GPIO_PORTA_BASE, GPIO_PIN_2);
126   }
127
128   GetSPI ();
129   ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH1 | ADC_CTL_IE |
        ADC_CTL_END);
130   ADCIntClear(ADC0_BASE, 3);
131   ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
132   while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
133   ADCIntClear(ADC0_BASE, 3);                  // Clear the ADC interrupt
         flag.
134   ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer1[sinc-1]);    // Read ADC
        Value.
135
136   ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH2 | ADC_CTL_IE |
        ADC_CTL_END);
137   ADCIntClear(ADC0_BASE, 3);
138   ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
139   while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
```

```
140   ADCIntClear(ADC0_BASE, 3);                    // Clear the ADC interrupt
        flag.
141   ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer2[sinc-1]);     // Read ADC
        Value.
142  }//end interrupt
143
144 int main(void)
145 {
146   // Set the clocking to run directly from the external crystal/
        oscillator.
147   SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
148                  SYSCTL_XTAL_16MHZ);
149   // Set up UART Serial Output
150   InitConsole();
151
152   // Set up SSI2 for SPI Communication
153   InitSPI();
154
155   //CS externo pin B2 y B3
156   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
157   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
158   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
159   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
160
161   //  Set up ADC
162   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);        // Puerto E para
        el canal analogico AN0
163   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);       // Conversor AD
164   //Configuramos el conversor AD
165   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1); //Configuramos el canal 0
166   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2); //Configuramos el canal 1
167   ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
168   ADCSequenceEnable(ADC0_BASE, 3);
169   ADCIntClear(ADC0_BASE, 3);
170
171   //  Inicializa Acc 1
172   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
173   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
174   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
175   while(SSIBusy(SSI2_BASE));
176   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
177   while(SSIBusy(SSI2_BASE));
178   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
179
180   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
181   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
182   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
183   while(SSIBusy(SSI2_BASE));
184   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
185   while(SSIBusy(SSI2_BASE));
186   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
```

```
187
188   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
189   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
190   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
191   while(SSIBusy(SSI2_BASE));
192   SSIDataPut(SSI2_BASE, 0x06); // Hz
193   while(SSIBusy(SSI2_BASE));
194   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
195
196   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
197   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
198   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
199   while(SSIBusy(SSI2_BASE));
200   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
201   while(SSIBusy(SSI2_BASE));
202   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
203
204   //  Inicializa Acc 2
205   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
206   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
207   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
208   while(SSIBusy(SSI2_BASE));
209   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
210   while(SSIBusy(SSI2_BASE));
211   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
212
213   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
214   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
215   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
216   while(SSIBusy(SSI2_BASE));
217   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
218   while(SSIBusy(SSI2_BASE));
219   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
220
221   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
222   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
223   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
224   while(SSIBusy(SSI2_BASE));
225   SSIDataPut(SSI2_BASE, 0x06); // Hz
226   while(SSIBusy(SSI2_BASE));
227   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
228
229   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
230   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
231   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
232   while(SSIBusy(SSI2_BASE));
233   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
234   while(SSIBusy(SSI2_BASE));
235   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
236
237   //  Pin de sincronizaci\'on
```

```
238   GPIOIntRegister(GPIO_PORTA_BASE, PortAIntHandler);
239   GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_2);
240   GPIOIntTypeSet(GPIO_PORTA_BASE, GPIO_PIN_2, GPIO_RISING_EDGE);
241   GPIOIntEnable(GPIO_PORTA_BASE, GPIO_PIN_2);
242
243   sinc=0;
244
245   while(1)
246   {
247     if(UARTgetc()=='T')
248     {
249       for(sinc=0;sinc<muestras;sinc++)
250       {
251         UARTprintf(" %i\t",z1[sinc]);
252         if(sinc==muestras-1)
253         {
254           UARTprintf("\n");
255         }
256       }
257       for(sinc=0;sinc<muestras;sinc++)
258       {
259         UARTprintf(" %i\t",z2[sinc]);
260         if(sinc==muestras-1)
261         {
262           UARTprintf("\n");
263         }
264       }
265       for(sinc=0;sinc<muestras;sinc++)
266       {
267         UARTprintf(" %u\t",adcbuffer1[sinc]);
268         if(sinc==muestras-1)
269         {
270           UARTprintf("\n");
271         }
272       }
273       for(sinc=0;sinc<muestras;sinc++)
274       {
275         UARTprintf(" %u\t",adcbuffer2[sinc]);
276         if(sinc==muestras-1)
277         {
278           UARTprintf("\n");
279         }
280       }
281
282       sinc=0;
283       GPIOIntEnable(GPIO_PORTA_BASE, GPIO_PIN_2);
284       }
285   }
286 }
```

Apéndice C

Programa utilizado en la etapa de validación

Programa C.1: Código para sensor virtual en la etapa de Validación

```
#include <stdbool.h>
#include <math.h>
#include <float.h>
#include <stdio.h>      /* printf, NULL */
#include <stdlib.h>     /* strtof */
#include <inttypes.h>   /*<stdint.h>*/
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/timer.h"
#include "driverlib/uart.h"
#include "driverlib/ssi.h"
#include "utils/uartstdio.h"
#include "driverlib/adc.h"
#include "inc/hw_timer.h"

int16_t z1, z2, z3,z4;
uint32_t adcbuffer1;
uint32_t adcbuffer2;
uint32_t adcbuffer3;
uint32_t adcbuffer4;
uint32_t muestra, values[8];
```

```
31 float P, R;
32 float X[11], S[4][1000], N1[20], N2[4], B1[20], B2[4];
33 float Xmax[11], Xmin[11], Smax[11], Smin[11];
34 float W[11][20];
35 float V[20][4];
36 char ini;
37 char buffer[30];
38 char Pitch[7], Roll[7];
39 unsigned long edge_period;
40 unsigned long value_edge1;
41 unsigned long value_edge2;
42 uint8_t edge_num;
43 uint32_t sinc;
44 uint8_t i;
45 uint8_t j;
46 uint8_t k;
47 uint8_t n;
48
49
50 void InitConsole(void)
51 {
52   // Enable GPIO port A which is used for UART0 pins.
53   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
54   GPIOPinConfigure(GPIO_PA0_U0RX);
55   GPIOPinConfigure(GPIO_PA1_U0TX);
56
57   // Enable UART0 so that we can configure the clock.
58   SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
59
60   // Use the internal 16MHz oscillator as the UART clock source.
61   UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
62
63   // Select the alternate (UART) function for these pins.
64   GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
65
66   // Initialize the UART for console I/O.
67   UARTStdioConfig(0, 115200, 16000000);
68 }
69
70 void InitSPI(void)
71 {
72   // The SSI2 peripheral must be enabled for use.
73   SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);
74
75   // Enable Port B
76   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
77
78   // SETUP SPI CS Pin (to output)
79   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
80
81   // Set SPI CS to HIGH (active-low)
```

```
82    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0xFF);
83
84    // Configure the pin muxing for SSI2 functions on port B4, B5, B6, and
        B7.
85    GPIOPinConfigure(GPIO_PB4_SSI2CLK);
86    GPIOPinConfigure(GPIO_PB5_SSI2FSS);
87    GPIOPinConfigure(GPIO_PB6_SSI2RX);
88    GPIOPinConfigure(GPIO_PB7_SSI2TX);
89
90    // Configure the GPIO settings for the SSI pins.
91    GPIOPinTypeSSI(GPIO_PORTB_BASE, GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
92                   GPIO_PIN_7);
93    // Configure and enable the SSI port for SPI master mode.
94    SSIConfigSetExpClk(SSI2_BASE, SysCtlClockGet(), SSI_FRF_MOTO_MODE_3,
95                       SSI_MODE_MASTER, 4000000, 8);
96    // Enable the SSI0 module.
97    SSIEnable(SSI2_BASE);
98
99    // Read any residual data from the SSI port.
100   uint32_t scrap;
101   while(SSIDataGetNonBlocking(SSI2_BASE, &scrap));
102
103   //CS externo pines B2, B3, C4 y C5
104   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
105   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
106   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
107   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
108   GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_4);
109   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
110   GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_5);
111   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
112 }
113 void InitADC (void)
114 {
115   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
116   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
117   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);       // Conversor AD
118   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
119
120   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1); //Configuramos el canal 0
121   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2); //Configuramos el canal 1
122   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3); //Configuramos el canal 2
123   GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_3); //Configuramos el canal 3
124   ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
125   ADCSequenceConfigure(ADC1_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
126   ADCSequenceEnable(ADC0_BASE, 3);
127   ADCSequenceEnable(ADC1_BASE, 3);
128   ADCIntClear(ADC0_BASE, 3);
129   ADCIntClear(ADC1_BASE, 3);
130 }
131
```

```
132 void InitIMU (void)
133 {
134   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
135   GPIOPinConfigure(GPIO_PD6_U2RX);
136   GPIOPinConfigure(GPIO_PD7_U2TX);
137   GPIOPinTypeUART(GPIO_PORTD_BASE, GPIO_PIN_6 | GPIO_PIN_7);
138
139   SysCtlPeripheralEnable(SYSCTL_PERIPH_UART2);
140   UARTConfigSetExpClk(UART2_BASE, SysCtlClockGet(), 57600,
141                       (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
142                        UART_CONFIG_PAR_NONE));
143   UARTEnable(UART2_BASE);
144
145   for(j=0;j<7;j++)
146   {
147     Pitch[j]=' ';
148     Roll[j]=' ';
149   }
150   for(j=0;j<30;j++)
151   {
152     buffer[j]=' ';
153   }
154 }
155
156 //interrupt
157 void Timer2IntHandler(void)
158 {
159   // Clear the timer interrupt
160   TimerIntClear(TIMER2_BASE, TIMER_CAPA_EVENT);
161   edge_num ++;
162   if(edge_num==1)
163   {
164     value_edge1 =   TimerValueGet(TIMER2_BASE,TIMER_A);
165     TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_NEG_EDGE);
166   }
167
168   if(edge_num==2)
169   {
170     value_edge2 =   TimerValueGet(TIMER2_BASE,TIMER_A);
171     edge_period = value_edge2-value_edge1;
172     edge_num =0;
173     TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_POS_EDGE);
174     TimerDisable(TIMER2_BASE, TIMER_A);
175     HWREG(TIMER2_BASE + TIMER_O_TAV) = 0;
176     TimerIntDisable(TIMER2_BASE, TIMER_CAPA_EVENT);
177   }
178 }//end interrupt
179
180 void InitTimerCap (void)
181 {
182   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
```

```
183   GPIOPinConfigure(GPIO_PF4_T2CCP0);
184   GPIOPinTypeTimer(GPIO_PORTF_BASE, GPIO_PIN_4);
185   SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);
186   TimerConfigure(TIMER2_BASE, TIMER_CFG_SPLIT_PAIR|
        TIMER_CFG_A_CAP_TIME_UP);
187   TimerControlEvent(TIMER2_BASE, TIMER_A, TIMER_EVENT_POS_EDGE);
188   IntEnable(INT_TIMER2A);
189   TimerIntDisable(TIMER2_BASE, TIMER_CAPA_EVENT);
190   IntMasterEnable();
191   TimerIntRegister(TIMER2_BASE, TIMER_A, *Timer2IntHandler);
192   TimerDisable(TIMER2_BASE, TIMER_A);
193   edge_num =0;
194   n=0;
195 }
196
197 void InitAcc (void)
198 {
199   //  Inicializa Acc 1
200   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
201   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
202   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
203   while(SSIBusy(SSI2_BASE));
204   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
205   while(SSIBusy(SSI2_BASE));
206   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
207
208   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
209   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
210   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
211   while(SSIBusy(SSI2_BASE));
212   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
213   while(SSIBusy(SSI2_BASE));
214   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
215
216   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
217   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
218   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
219   while(SSIBusy(SSI2_BASE));
220   SSIDataPut(SSI2_BASE, 0x06); // Hz
221   while(SSIBusy(SSI2_BASE));
222   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
223
224   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
225   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
226   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
227   while(SSIBusy(SSI2_BASE));
228   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
229   while(SSIBusy(SSI2_BASE));
230   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
231
232   //  Inicializa Acc 2
```

```
233   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
234   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
235   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
236   while(SSIBusy(SSI2_BASE));
237   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
238   while(SSIBusy(SSI2_BASE));
239   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
240
241   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
242   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
243   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
244   while(SSIBusy(SSI2_BASE));
245   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
246   while(SSIBusy(SSI2_BASE));
247   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
248
249   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
250   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
251   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
252   while(SSIBusy(SSI2_BASE));
253   SSIDataPut(SSI2_BASE, 0x06); // Hz
254   while(SSIBusy(SSI2_BASE));
255   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
256
257   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
258   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
259   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
260   while(SSIBusy(SSI2_BASE));
261   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
262   while(SSIBusy(SSI2_BASE));
263   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
264
265   //  Inicializa Acc 3
266   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
267   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
268   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
269   while(SSIBusy(SSI2_BASE));
270   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
271   while(SSIBusy(SSI2_BASE));
272   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
273
274   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
275   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
276   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
277   while(SSIBusy(SSI2_BASE));
278   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
279   while(SSIBusy(SSI2_BASE));
280   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
281
282   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
283   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
```

```
284   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
285   while(SSIBusy(SSI2_BASE));
286   SSIDataPut(SSI2_BASE, 0x06); // Hz
287   while(SSIBusy(SSI2_BASE));
288   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
289
290   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
291   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
292   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
293   while(SSIBusy(SSI2_BASE));
294   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
295   while(SSIBusy(SSI2_BASE));
296   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
297
298   //  Inicializa Acc 4
299   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
300   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
301   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
302   while(SSIBusy(SSI2_BASE));
303   SSIDataPut(SSI2_BASE, 0x00); //SLEEP MODE
304   while(SSIBusy(SSI2_BASE));
305   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
306
307   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
308   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
309   SSIDataPut(SSI2_BASE, 0x31); // DATA_FORMAT
310   while(SSIBusy(SSI2_BASE));
311   SSIDataPut(SSI2_BASE, 0x0B); //full res bit range +-16G
312   while(SSIBusy(SSI2_BASE));
313   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
314
315   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
316   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
317   SSIDataPut(SSI2_BASE, 0x2C); // BW_RATE
318   while(SSIBusy(SSI2_BASE));
319   SSIDataPut(SSI2_BASE, 0x06); // Hz
320   while(SSIBusy(SSI2_BASE));
321   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
322
323   SysCtlDelay(100 * (SysCtlClockGet() /1000 /3));
324   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
325   SSIDataPut(SSI2_BASE, 0x2D); // POWER_CTL
326   while(SSIBusy(SSI2_BASE));
327   SSIDataPut(SSI2_BASE, 0x08); // MEASURE MODE
328   while(SSIBusy(SSI2_BASE));
329   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
330 }
331
332 void GetSPI (void)
333 {
334   // Z1_0
```

```
335   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
336   SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
337   while(SSIBusy(SSI2_BASE));
338   SSIDataPut(SSI2_BASE, 0x00);
339   SSIDataGetNonBlocking(SSI2_BASE, &values[0]);
340   while(SSIBusy(SSI2_BASE));
341   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
342
343   //  Z1_1
344   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
345   SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
346   while(SSIBusy(SSI2_BASE));
347   SSIDataPut(SSI2_BASE, 0x00);
348   SSIDataGetNonBlocking(SSI2_BASE, &values[1]);
349   while(SSIBusy(SSI2_BASE));
350   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
351
352   //  Z2_0
353   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
354   SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
355   while(SSIBusy(SSI2_BASE));
356   SSIDataPut(SSI2_BASE, 0x00);
357   SSIDataGetNonBlocking(SSI2_BASE, &values[2]);
358   while(SSIBusy(SSI2_BASE));
359   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
360
361   //  Z2_1
362   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
363   SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
364   while(SSIBusy(SSI2_BASE));
365   SSIDataPut(SSI2_BASE, 0x00);
366   SSIDataGetNonBlocking(SSI2_BASE, &values[3]);
367   while(SSIBusy(SSI2_BASE));
368   GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
369
370   //  Z3_0
371   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
372   SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
373   while(SSIBusy(SSI2_BASE));
374   SSIDataPut(SSI2_BASE, 0x00);
375   SSIDataGetNonBlocking(SSI2_BASE, &values[4]);
376   while(SSIBusy(SSI2_BASE));
377   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
378
379   //  Z3_1
380   GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0x00);
381   SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
382   while(SSIBusy(SSI2_BASE));
383   SSIDataPut(SSI2_BASE, 0x00);
384   SSIDataGetNonBlocking(SSI2_BASE, &values[5]);
385   while(SSIBusy(SSI2_BASE));
```

```
386     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 0xFF);
387
388     //  Z4_0
389     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
390     SSIDataPut(SSI2_BASE, 0xB6); // 0x36 + bx1000 = 0xB6
391     while(SSIBusy(SSI2_BASE));
392     SSIDataPut(SSI2_BASE, 0x00);
393     SSIDataGetNonBlocking(SSI2_BASE, &values[6]);
394     while(SSIBusy(SSI2_BASE));
395     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
396
397     //  Z4_1
398     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0x00);
399     SSIDataPut(SSI2_BASE, 0xB7); // 0x37 + bx1000 = 0xB7
400     while(SSIBusy(SSI2_BASE));
401     SSIDataPut(SSI2_BASE, 0x00);
402     SSIDataGetNonBlocking(SSI2_BASE, &values[7]);
403     while(SSIBusy(SSI2_BASE));
404     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0xFF);
405
406     //The Z1 value.
407     z1 = ((int16_t)values[1]<<8)|(int16_t)values[0];
408
409     //The Z2 value.
410     z2 = ((int16_t)values[3]<<8)|(int16_t)values[2];
411
412     //The Z3 value.
413     z3 = ((int16_t)values[5]<<8)|(int16_t)values[4];
414
415     //The Z4 value.
416     z4 = ((int16_t)values[7]<<8)|(int16_t)values[6];
417
418     return;
419 }
420
421 void GetADC (void)
422 {
423     ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH1 | ADC_CTL_IE |
        ADC_CTL_END);
424     ADCIntClear(ADC0_BASE, 3);
425     ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
426     while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
427     ADCIntClear(ADC0_BASE, 3);                  // Clear the ADC interrupt
         flag.
428     ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer1);    // Read ADC Value.
429
430     ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH2 | ADC_CTL_IE |
        ADC_CTL_END);
431     ADCIntClear(ADC0_BASE, 3);
```

```
432   ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
433   while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
434   ADCIntClear(ADC0_BASE, 3);                  // Clear the ADC interrupt
         flag.
435   ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer2);    // Read ADC Value.
436
437   ADCSequenceStepConfigure(ADC1_BASE, 3, 0, ADC_CTL_CH1 | ADC_CTL_IE |
        ADC_CTL_END);
438   ADCIntClear(ADC0_BASE, 3);
439   ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
440   while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
441   ADCIntClear(ADC0_BASE, 3);                  // Clear the ADC interrupt
         flag.
442   ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer3);    // Read ADC Value.
443
444   ADCSequenceStepConfigure(ADC1_BASE, 3, 0, ADC_CTL_CH2 | ADC_CTL_IE |
        ADC_CTL_END);
445   ADCIntClear(ADC0_BASE, 3);
446   ADCProcessorTrigger(ADC0_BASE, 3);          // Trigger the ADC
        conversion.
447   while(!ADCIntStatus(ADC0_BASE, 3, false)){} // Wait for conversion to
        be completed.
448   ADCIntClear(ADC0_BASE, 3);                  // Clear the ADC interrupt
         flag.
449   ADCSequenceDataGet(ADC0_BASE, 3, &adcbuffer4);    // Read ADC Value.
450  }
451
452 void GetIMU (void)
453 {
454   GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, 0xFF);
455   UARTEnable(UART2_BASE);
456   TimerEnable(TIMER2_BASE, TIMER_A);
457   TimerIntEnable(TIMER2_BASE, TIMER_CAPA_EVENT);
458   while(1)
459   {
460     ini=UARTCharGet(UART2_BASE);
461     if(ini=='#')
462     {
463       break;
464     }
465   }
466   for(i=0;i<30;i++)
467   {
468     buffer[i]=UARTCharGet(UART2_BASE);
469     if(buffer[i]=='\r')
470     {
471       break;
```

```
472       }
473     }
474     GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, 0x00);
475     UARTDisable(UART2_BASE);
476
477     //Interprete del buffer
478     i=0;
479     j=0;
480     while(buffer[i+j]!=',')
481     {
482       j++;
483     }
484     i=i+j+1;
485     j=0;
486     while(buffer[i+j]!=',')
487     {
488       Pitch[j]=buffer[i+j];
489       j++;
490     }
491     i=i+j+1;
492     j=0;
493     while(buffer[i+j]!='.')
494     {
495       Roll[j]=buffer[i+j];
496       j++;
497     }
498     Roll[j]=buffer[i+j];
499     j++;
500     Roll[j]=buffer[i+j];
501     j++;
502     Roll[j]=buffer[i+j];
503
504     P=atof(Pitch);
505     R=atof(Roll);
506     for(j=0;j<7;j++)
507     {
508       Pitch[j]=' ';
509       Roll[j]=' ';
510     }
511     for(j=0;j<30;j++)
512     {
513       buffer[j]=' ';
514     }
515   }
516
517   void InitValues (void)
518   {
519     B1[0]=-2.82;   B1[1]=0.22;    B1[2]=-1.41;   B1[3]=-0.23;
520     B1[4]=1.04;    B1[5]=-0.06;   B1[6]=1.19;    B1[7]=0.14;
521     B1[8]=1.26;    B1[9]=-0.36;   B1[10]=-0.06;  B1[11]=-0.59;
522     B1[12]=0.32;   B1[13]=1.49;   B1[14]=-5.11;  B1[15]=-3.36;
```

```
523   B1[16]=-1.74; B1[17]=-6.01; B1[18]=3.10;  B1[19]=0.38;
524
525   B2[0]=-0.34;  B2[1]=0.03;   B2[2]=-0.52;  B2[3]=-0.04;
526
527   V[0][0]=-0.23;    V[0][1]=-0.09;    V[0][2]=-1.02;    V[0][3]=-0.27;
528   V[1][0]=0.43;     V[1][1]=0.01;     V[1][2]=2.30;     V[1][3]=0.33;
529   V[2][0]=-0.26;    V[2][1]=0.17;     V[2][2]=0.20;     V[2][3]=0.41;
530   V[3][0]=0.25;     V[3][1]=0.04;     V[3][2]=1.69;     V[3][3]=0.21;
531   V[4][0]=0.17;     V[4][1]=0.03;     V[4][2]=0.07;     V[4][3]=0.20;
532   V[5][0]=-0.04;    V[5][1]=-0.08;    V[5][2]=-0.33;    V[5][3]=-0.07;
533   V[6][0]=1.39;     V[6][1]=0.23;     V[6][2]=-0.86;    V[6][3]=-0.37;
534   V[7][0]=-0.74;    V[7][1]=-0.12;    V[7][2]=-1.71;    V[7][3]=-0.32;
535   V[8][0]=-1.35;    V[8][1]=-0.68;    V[8][2]=0.69;     V[8][3]=-0.17;
536   V[9][0]=0.67;     V[9][1]=1.86;     V[9][2]=1.79;     V[9][3]=1.31;
537   V[10][0]=-0.77;   V[10][1]=0.26;    V[10][2]=-0.04;   V[10][3]=0.46;
538   V[11][0]=-0.30;   V[11][1]=0.23;    V[11][2]=-0.02;   V[11][3]=0.11;
539   V[12][0]=-0.24;   V[12][1]=1.30;    V[12][2]=1.01;    V[12][3]=1.10;
540   V[13][0]=2.31;    V[13][1]=-1.11;   V[13][2]=-1.97;   V[13][3]=-2.32;
541   V[14][0]=-0.14;   V[14][1]=-0.26;   V[14][2]=0.16;    V[14][3]=-0.06;
542   V[15][0]=-0.03;   V[15][1]=-0.01;   V[15][2]=-0.22;   V[15][3]=0.05;
543   V[16][0]=1.74;    V[16][1]=0.41;    V[16][2]=-1.44;   V[16][3]=-1.09;
544   V[17][0]=0.22;    V[17][1]=-0.37;   V[17][2]=0.13;    V[17][3]=-0.33;
545   V[18][0]=-0.06;   V[18][1]=-0.01;   V[18][2]=-0.10;   V[18][3]=0.01;
546   V[19][0]=0.53;    V[19][1]=1.16;    V[19][2]=1.98;    V[19][3]=1.39;
547
548   W[0][0]=-1.97;  W[1][0]=-2.44;  W[2][0]=-2.29;  W[3][0]=0.26;
549   W[4][0]=-0.04;  W[5][0]=-0.14;  W[6][0]=0.38;   W[7][0]=0.49;
550   W[8][0]=-2.08;  W[9][0]=0.47;   W[10][0]=0.34;
551
552   W[0][1]=-2.78;  W[1][1]=0.87;   W[2][1]=1.28;   W[3][1]=-0.11;
553   W[4][1]=-0.14;  W[5][1]=-0.28;  W[6][1]=-0.07;  W[7][1]=2.25;
554   W[8][1]=-1.04;  W[9][1]=0.35;   W[10][1]=0.16;
555
556   W[0][2]=0.59;   W[1][2]=2.87;   W[2][2]=-1.21;  W[3][2]=0.28;
557   W[4][2]=-0.31;  W[5][2]=-0.09;  W[6][2]=-0.07;  W[7][2]=4.75;
558   W[8][2]=-0.94;  W[9][2]=-0.38;  W[10][2]=0.86;
559
560   W[0][3]=3.17;   W[1][3]=-2.03;  W[2][3]=-0.99;  W[3][3]=0.13;
561   W[4][3]=0.05;   W[5][3]=0.28;   W[6][3]=0.25;   W[7][3]=-3.08;
562   W[8][3]=1.15;   W[9][3]=0.01;   W[10][3]=-0.61;
563
564   W[0][4]=-0.89;  W[1][4]=0.99;   W[2][4]=1.46;   W[3][4]=-0.88;
565   W[4][4]=-0.32;  W[5][4]=0.06;   W[6][4]=-0.89;  W[7][4]=0.13;
566   W[8][4]=-1.93;  W[9][4]=1.85;   W[10][4]=-0.81;
567
568   W[0][5]=-0.61;  W[1][5]=-1.50;  W[2][5]=-2.74;  W[3][5]=0.30;
569   W[4][5]=-1.14;  W[5][5]=0.24;   W[6][5]=2.16;   W[7][5]=3.03;
570   W[8][5]=-2.25;  W[9][5]=-0.79;  W[10][5]=0.09;
571
572   W[0][6]=2.73;   W[1][6]=-6.57;  W[2][6]=0.98;   W[3][6]=0.03;
573   W[4][6]=-0.17;  W[5][6]=0.09;   W[6][6]=0.17;   W[7][6]=0.83;
```

```
574   W[8][6]=-3.18;  W[9][6]=-2.43;  W[10][6]=0.76;
575
576   W[0][7]=-1.11;  W[1][7]=0.91;   W[2][7]=0.83;   W[3][7]=-0.08;
577   W[4][7]=0.11;   W[5][7]=-0.13;  W[6][7]=-0.23;  W[7][7]=-1.12;
578   W[8][7]=0.70;   W[9][7]=0.65;   W[10][7]=-0.58;
579
580   W[0][8]=3.67;   W[1][8]=-6.86;  W[2][8]=0.54;   W[3][8]=0.15;
581   W[4][8]=0.08;   W[5][8]=0.09;   W[6][8]=0.05;   W[7][8]=2.27;
582   W[8][8]=-5.09;  W[9][8]=-2.89;  W[10][8]=0.49;
583
584   W[0][9]=-1.12;  W[1][9]=0.80;   W[2][9]=0.51;   W[3][9]=0.01;
585   W[4][9]=0.03;   W[5][9]=0.04;   W[6][9]=0.00;   W[7][9]=0.36;
586   W[8][9]=-1.15;  W[9][9]=0.44;   W[10][9]=-0.07;
587
588   W[0][10]=1.51;  W[1][10]=0.33;  W[2][10]=0.92;  W[3][10]=0.23;
589   W[4][10]=0.19;  W[5][10]=-0.01; W[6][10]=-0.11; W[7][10]=-0.94;
590   W[8][10]=-1.05; W[9][10]=0.29;  W[10][10]=-0.48;
591
592   W[0][11]=0.76;  W[1][11]=-1.97; W[2][11]=-2.66; W[3][11]=-0.59;
593   W[4][11]=-0.57; W[5][11]=-0.09; W[6][11]=0.80;  W[7][11]=1.89;
594   W[8][11]=-0.29; W[9][11]=0.35;  W[10][11]=1.33;
595
596   W[0][12]=-0.97; W[1][12]=-0.89; W[2][12]=0.52;  W[3][12]=-0.05;
597   W[4][12]=0.02;  W[5][12]=-0.11; W[6][12]=-0.19; W[7][12]=-1.67;
598   W[8][12]=2.21;  W[9][12]=0.00;  W[10][12]=-0.47;
599
600   W[0][13]=-0.07; W[1][13]=0.06;  W[2][13]=-0.60; W[3][13]=0.06;
601   W[4][13]=0.28;  W[5][13]=0.03;  W[6][13]=-0.03; W[7][13]=-3.42;
602   W[8][13]=1.88;  W[9][13]=-0.01; W[10][13]=-0.30;
603
604   W[0][14]=-3.91; W[1][14]=2.09;  W[2][14]=-0.02; W[3][14]=-0.19;
605   W[4][14]=-1.58; W[5][14]=-0.44; W[6][14]=0.47;  W[7][14]=2.03;
606   W[8][14]=6.35;  W[9][14]=2.61;  W[10][14]=0.08;
607
608   W[0][15]=1.34;  W[1][15]=3.10;  W[2][15]=2.17;  W[3][15]=0.92;
609   W[4][15]=-2.46; W[5][15]=0.93;  W[6][15]=2.10;  W[7][15]=1.43;
610   W[8][15]=-0.20; W[9][15]=-0.11; W[10][15]=-1.28;
611
612   W[0][16]=0.68;  W[1][16]=0.71;  W[2][16]=-0.05; W[3][16]=0.08;
613   W[4][16]=-0.32; W[5][16]=-0.20; W[6][16]=0.04;  W[7][16]=3.70;
614   W[8][16]=-1.38; W[9][16]=0.25;  W[10][16]=0.22;
615
616   W[0][17]=-3.37; W[1][17]=0.91;  W[2][17]=3.50;  W[3][17]=-0.72;
617   W[4][17]=-0.51; W[5][17]=-0.56; W[6][17]=0.12;  W[7][17]=3.44;
618   W[8][17]=-1.47; W[9][17]=6.36;  W[10][17]=-1.25;
619
620   W[0][18]=1.96;  W[1][18]=5.15;  W[2][18]=5.59;  W[3][18]=-0.54;
621   W[4][18]=2.89;  W[5][18]=0.56;  W[6][18]=1.42;  W[7][18]=1.44;
622   W[8][18]=3.63;  W[9][18]=0.85;  W[10][18]=-3.37;
623
624   W[0][19]=1.57;  W[1][19]=0.51;  W[2][19]=-1.00; W[3][19]=0.07;
```

```
625   W[4][19]=0.05;   W[5][19]=-0.01; W[6][19]=0.19;   W[7][19]=-1.37;
626   W[8][19]=1.34;   W[9][19]=-0.26; W[10][19]=-0.08;
627
628   Xmin[0]=42239; Xmin[1]=-3.58; Xmin[2]=-3.31; Xmin[3]=259;
629   Xmin[4]=290;    Xmin[5]=-247;   Xmin[6]=-215;   Xmin[7]=825;
630   Xmin[8]=1481;   Xmin[9]=793;    Xmin[10]=690;
631
632   Xmax[0]=78196; Xmax[1]=10.06; Xmax[2]=2.08; Xmax[3]=276;
633   Xmax[4]=312;    Xmax[5]=-219;   Xmax[6]=-190; Xmax[7]=3060;
634   Xmax[8]=4033;   Xmax[9]=2084;   Xmax[10]=1439;
635
636   Smin[0]=171.829;   Smin[1]=721.508;   Smin[2]=486.977; Smin[3]=740.559;
637
638   Smax[0]=1790.336; Smax[1]=1845.720; Smax[2]=2269.617; Smax
        [3]=1826.922;
639 }
640
641 void RunANN(void)
642 {
643   X[0]=2*(edge_period-Xmin[0])/(Xmax[0]-Xmin[0])-1;
644   X[1]=2*(P-Xmin[1])/(Xmax[1]-Xmin[1])-1;
645   X[2]=2*(R-Xmin[2])/(Xmax[2]-Xmin[2])-1;
646   X[3]=2*(z1-Xmin[3])/(Xmax[3]-Xmin[3])-1;
647   X[4]=2*(z2-Xmin[4])/(Xmax[4]-Xmin[4])-1;
648   X[5]=2*(z3-Xmin[5])/(Xmax[5]-Xmin[5])-1;
649   X[6]=2*(z4-Xmin[6])/(Xmax[6]-Xmin[6])-1;
650   X[7]=2*(adcbuffer1-Xmin[7])/(Xmax[7]-Xmin[7])-1;
651   X[8]=2*(adcbuffer2-Xmin[8])/(Xmax[8]-Xmin[8])-1;
652   X[9]=2*(adcbuffer3-Xmin[9])/(Xmax[9]-Xmin[9])-1;
653   X[10]=2*(adcbuffer4-Xmin[10])/(Xmax[10]-Xmin[10])-1;
654
655   for(i=0;i<20;i++)
656   {
657     N1[i]=0;
658     for(j=0;j<11;j++)
659     {
660       N1[i]=(N1[i]+(X[j]*W[j][i]));
661     }
662   }
663   for(i=0;i<20;i++)
664   {
665     N1[i]=2/(1+exp(-2*(N1[i]+B1[i])))-1;
666   }
667
668   for(i=0;i<4;i++)
669   {
670     N2[i]=0;
671     for(j=0;j<20;j++)
672     {
673       N2[i]=(N2[i]+(N1[j]*V[j][i]));
674     }
```

```
675   }
676   for(i=0;i<4;i++)
677   {
678     N2[i]=N2[i]+B2[i];
679     S[i][muestra]=(Smax[i]-Smin[i])*(N2[i]+1)/(2)+Smin[i];
680   }
681   muestra ++;
682 }
683
684 int main(void)
685 {
686   // Set the clocking to run directly from the external crystal/
        oscillator.
687   SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
688                  SYSCTL_XTAL_16MHZ);
689   // Set up UART Serial Output
690   InitConsole();
691
692   // Set up SSI2 for SPI Communication
693   InitSPI();
694   InitAcc();
695
696   //  Set up ADC
697   InitADC();
698
699   // Set up IMU
700    InitIMU();
701
702    // Set up Timer capture
703    InitTimerCap();
704
705    // Set up NN Values
706    InitValues();
707
708    muestra=0;
709
710   while(1)
711   {
712     if(UARTgetc()=='1')
713     {
714       GetSPI();
715       GetADC();
716       GetIMU();
717
718       RunANN();
719     }
720
721     if(UARTgetc()=='T')
722     {
723       for(muestra=0;muestra<1000;muestra++)
724       {
```

```
725           for(i=0;i<4;i++)
726           {
727             UARTprintf(" %f\t",S[i][muestra]);
728           }
729           UARTprintf("\n");
730         }
731         muestra=0;
732       }
733     }
734 }
```