UNIVERSITAT POLITÈCNICA DE CATALUNYA

MÀSTER:

AUTOMÀTICA I ROBÒTICA

TESI DE MÀSTER

DISEÑO E IMPLEMENTACIÓN DE UN SENSOR DE FUERZA DE 6 GRADOS DE LIBERTAD

JEAN CARLO SALAZAR CORTÉS

DIRECTORA: ALÍCIA CASALS GELPÍ

CURS ACADÈMIC 2010/2011

SETEMBRE 2011

PROJECTE DE FI DE MÀSTER

Autor: Jean Carlo Salazar Cortés

Directora: Alícia Casals Gelpí

Tribunal d'avaluació:

[NOM DEL PRESIDENT]

[NOM DEL VOCAL]

[NOM DEL SECRETARI]

Diseño e implementación de un sensor de fuerza de 6 grados de libertad

Jean Carlo Salazar Cortés
Departament d'Enginyeria de Sistemes
Automàtica i Informàtica Industrial
Universitat Politècnica de Catalunya
Barcelona

Septiembre de 2011

Abstract

El objetivo del presente proyecto final de máster es el diseño e implementación de un sensor de fuerza y par, capaz de medir los tres componentes de fuerza y los tres componentes de par, con alta velocidad de procesamiento de señales que posibiliten su implementación en tiempo real con conexión a la interfaz de usuario a través del protocolo USB, que posea la dimensión apropiada para poder ser montado entre la última articulación y el efector final de un robot manipulador.

Para lo cual se ha optado por una estructura en forma delta, con un anillo circular exterior y tres vigas radiales espaciadas un ángulo de 120º sujetas en el interior a una brida también circular. La estructura se ha mecanizado a partir de un bloque de aluminio, buscando que no existan uniones soldadas o sujetas con tornillos. En cada una de las vigas hay pegadas cuatro galgas extensiométricas que miden la flexión en dos direcciones ortogonales, son un total de 12 galgas extensiométricas fijadas a la estructura, una por cada cara, que se conectan a seis puentes de Wheatstone para transformar las variaciones resistivas en señales de voltaje.

El procesamiento de estas señales, se realiza utilizando un microcontrolador de 16-bits de la familia dsPIC30F de Microchip, luego de realizar el procedimiento de calibración, se pueden obtener las componentes de las fuerzas y pares que actúan sobre el sensor, esta información se transmite a la interfaz de usuario por medio del protocolo USB, que permite y facilita la integración a aplicaciones en tiempo real.

Para la validación final del diseño del sensor de fuerza se ha realizado una aplicación de control por fuerza de un robot manipulador SCARA. Con esta aplicación se evalúa el correcto funcionamiento del sensor en aspectos como su velocidad de respuesta y linealidad.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1. Introducción

Teniendo en cuenta que un robot manipulador realiza tareas donde interactúa con el medio y los objetos de su entorno, se hace necesario en la mayoría de los casos conocer o estimar las variables físicas del espacio de trabajo, así pues, en estos casos la adquisición y uso de la información del medio donde el robot opera es de alta importancia para la realización de la tarea [14][36]. Por ejemplo, para la realización de una tarea particular puede ser necesario saber la temperatura o humedad del ambiente o del objeto manipulado, o la posición, velocidad o aceleración de los objetos, o también la fuerza o par que se ejerce sobre un objeto determinado o que ejerce el entorno sobre el robot.

La información de las variables del entorno, robot u objetos, generalmente es obtenida usando sensores específicos para la variable que se desea medir. Este proyecto se centra en las variables de fuerza y par, que pueden ser ejercidas por un robot manipulador sobre un objeto o viceversa, para lo cual se utilizan sensores del tipo galgas extensiométricas. Estos sensores van pegados a un elemento deformable elásticamente, el cual bajo la acción de la fuerza que se desea medir sufre deformaciones junto con la galga, lo que ocasiona un cambio en la resistencia eléctrica del sensor.

Como ejemplo de sus múltiples aplicaciones en la Figura 1.1 se muestra la aplicación final del desarrollo de un sensor de fuerza para un torno paralelo para medir las fuerzas y los pares de la herramienta [29].

También se puede encontrar su utilización en la rehabilitación de la movilidad de extremidades, como por ejemplo la movilidad de una mano. En la Figura 1.2 se muestra su implementación para la rehabilitación de la mano de un paciente que ha sufrido un accidente cerebro vascular [18].

La memoria del proyecto está organizada de la siguiente manera: en el capítulo 1 se muestra la introducción, los objetivos, justificación y metodología desarrollados en el proyecto; en el capítulo 2 se presenta un estado del arte de los sensores de fuerza, como los tipos de sensores de fuerza, el principio de funcionamiento, las geometrías utilizadas en el diseño del sensor, y la formulación matemática, sustentado en una amplia revisión bibliográfica. En el capítulo 3 se explica el diseño del prototipo del sensor desarrollado, y en el capítulo 4 se presenta el análisis de los resultados

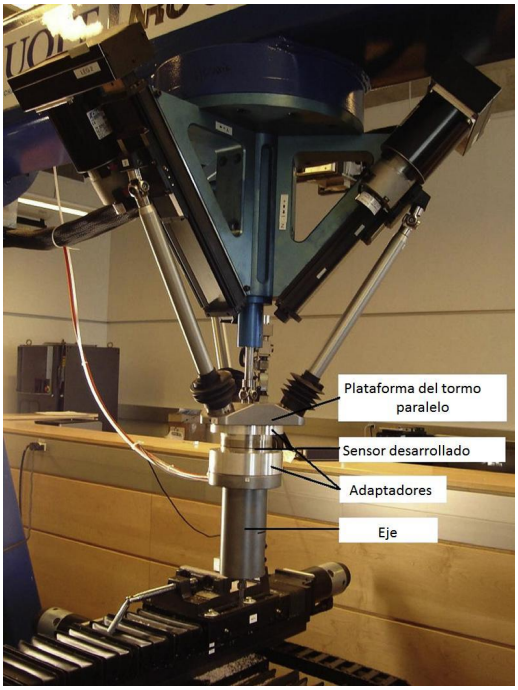

Figura 1.1: Desarrollo de un sensor de fuerza para un torno paralelo.

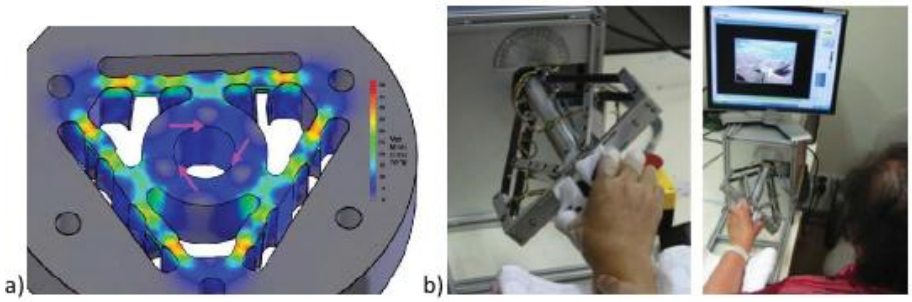

Figura 1.2: Sensor de fuerza utilizado para realizar rehabilitación de la movilidad de la mano, a) Análisis de elementos finitos de la estructura, b) aplicación final.

obtenidos y por último en el capítulo 5 se presentan las conclusiones y trabajos futuros del proyecto.

1.2. Motivación

El grupo de investigación GRINS de la UPC trabaja principalmente en la realización de prototipos robóticos innovadores, sobre todo en el campo de la biomedicina, con aplicaciones como por ejemplo robots para laparoscopia o cirugía asistida entre otras. En muchas de estas aplicaciones se necesita saber la magnitud de la fuerza que se ejerce con el robot por ejemplo sobre los tejidos del paciente, o limitar los movimientos que realiza un cirujano para proteger los órganos del paciente de un posible error, o también puede ser necesario conocer la fuerza que debe soportar una prótesis cuando el paciente la usa, etc.

Teniendo en cuenta que estas aplicaciones requieren de alta velocidad de respuesta por ser sistemas en tiempo real y que el coste de implementación se debe mantener bajo, se ve la necesidad de desarrollar un dispositivo sensor que cumpla con dichas características, y para tal fin se plantea la elaboración de este proyecto de fin de máster.

1.3. Objetivos

1.3.1. Objetivo General

El objetivo de esta tesis de máster es:

- Desarrollar un sensor de fuerza de seis grados de libertad para su integración al elemento terminal de un robot para aplicaciones que necesiten una estrategia de control basado en la fuerza aplicada.

1.3.2. Objetivos Específicos

- Diseñar el hardware para el acondicionamiento y adquisición de las señales del sensor de fuerzas.
- Efectuar un estudio de las diferentes estructuras elásticas utilizadas para sensores de fuerza.
- Efectuar el análisis matemático de la estructura elegida, para realizar la calibración del sensor.
- Diseñar el programa de adquisición de señales del sensor, procurando la mayor precisión y sensibilidad de las mediciones.
- Diseñar la interfaz de control para el robot, manteniendo un buen nivel de usabilidad para el usuario final.
- Verificar el funcionamiento del sensor en una aplicación de control en tiempo real, integrando un robot como elemento a controlar.
- Redactar un informe técnico sobre el proyecto.

1.4. Metodología

Para cumplir satisfactoriamente con los objetivos trazados, el presente proyecto se ha dividido en 6 fases: 1) análisis de requerimientos del sensor, 2) revisión bibliográfica, 3) diseño del sistema, 4) implementación del sistema, 5) conclusiones y trabajos futuros y 6) publicación de resultados.

1. Análisis de requisitos: analizar las características que debe cumplir el prototipo de sensor a diseñar. Este punto constituye el planteamiento del problema.

2. Revisión bibliográfica: como primer paso y para caracterizar el tema de investigación se lleva a cabo una amplia recolección y revisión documental, realizando una búsqueda de artículos referentes al problema planteado en bases de datos reconocidas por su impacto científico en el ámbito de la ingeniería como el IEEE Xplore, ScienceDirect, ISI Web of Knowledge, por supuesto se ha utilizado valiosa información de libros de autores y editoriales especializados en el tema.

3. Diseño del sistema: teniendo claras las características deseadas se procede al diseño del sistema del sensor. Este punto constituye la formulación de la hipótesis.

4. Implementación del sistema: es el siguiente paso luego de haber diseñado el sistema del sensor, se procede entonces a la elaboración de un prototipo de sensor.

5. Conclusiones y trabajos futuros: se contrarrestan los objetivos planteados con los resultados obtenidos y se proponen futuras líneas de trabajo.

6. Publicación de resultados: redactar un reporte técnico mostrando los resultados obtenidos.

En la Tabla 1.1 se puede observar el cronograma de actividades que se ha seguido para llevar a término el proyecto planteado.

TAREAS	Semanas											
	1	2	3	4	5	6	7	8	9	10	11	12
• Búsqueda y lectura de literatura pertinente.												
• Diseño de la disposición de las galgas en la estructura del sensor de fuerza.												
• Diseñar y montar el circuito electrónico para la interconexión de las galgas.												
• Elaborar el software que permita realizar la toma de medidas de las galgas.												
• Elaborar la tarjeta de adquisición de señales.												
• Determinar los coeficientes de temperatura y calibrar el prototipo del sensor.												
• Comprobar el correcto funcionamiento del prototipo desarrollado.												
• Redactar la memoria del proyecto.												

Tabla 1.1: Cronograma de actvidades.

Capítulo 2

Estado del arte

En esta sección se presenta el estado del arte de los temas relacionados con el desarrollo de esta tesis de máster. Este estado del arte estudia la teoría de los sensores de fuerza, enfocándose básicamente en los que usan transductores de galgas extensiométricas. En primer lugar se describen los tipos de sensores de fuerza/par existentes, y a continuación se presenta una revisión de la teoría de sensores de galgas extensiométricas, su estructura, y su formulación matemática.

2.1. Tipos de sensores de fuerza/par

Algunos tipos de transductores de fuerza se basan en el mismo principio, la medición de una deflexión causada por una fuerza en un material, pero se diferencian en el sensor que se utiliza para transformar esta deflexión en señal eléctrica.

De acuerdo al sensor que se utilice para transformar la deflexión provocada por la aplicación de una fuerza o par, se pueden encontrar diferentes tipos de transductores de fuerza, algunos de los más utilizados son:

Ópticos: utilizan un haz de luz proyectado sobre un sensor fotoeléctrico sensible a este haz, según ocurra una deflexión, el haz de luz se moverá respecto al punto donde se proyectaba anteriormente, la deflexión y el posterior desplazamiento del haz de luz dan una indicación de la magnitud de la fuerza que los ha causado [3][11][43]. En la Figura 2.1 se presenta un ejemplo de un sensor óptico [43], se aprecia el detalle del emisor de luz pegado a un extremo de la viga y el foto sensor pegado al otro extremo, tal que una flexión de la viga hace mover el haz de luz proyectado sobre el foto sensor.

Inductivos: el desplazamiento causado por la deflexión puede relacionarse con el cambio en el flujo electromagnético cuando se mide la inductividad mutua de dos bobinas que, o tienen un núcleo móvil o se desplazan relativamente una de otra (Figura 2.2).

Capacitivos: en este tipo de sensores lo que se busca es la variación de la distancia entre las placas de un condensador de acuerdo a la fuerza que actúa sobre él, teniendo en cuenta que la capacitancia viene dada por la siguiente expresión:

$$C = \epsilon_0 \epsilon_r \frac{A}{h} \tag{2.1}$$

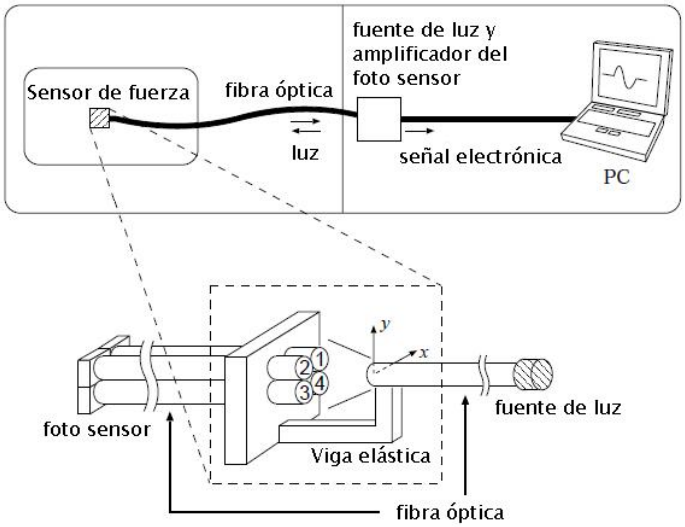

Figura 2.1: Ejemplo de sensor óptico.

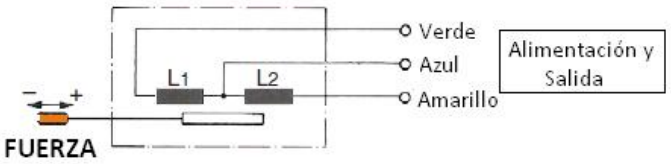

Figura 2.2: Esquema de un sensor inductivo.

Donde A representa el área de las placas, h es la distancia entre las placas y ϵ_0 es la permitividad al vacío y ϵ_r la permisividad relativa del dieléctrico. Por consiguiente, un cambio en cualquiera de los parámetros produce un cambio en la capacitancia, así la capacitancia del sensor cambia conforme cambia la distancia entre las placas a causa de la fuerza ejercida sobre él, una representación de este tipo de sensor se puede ver en la Figura 2.3 [2][42].

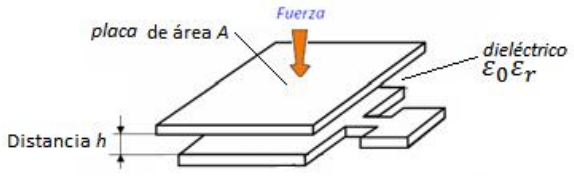

Figura 2.3: Esquema de un sensor capacitivo.

Galgas extensiométricas: con este tipo de sensores lo que se procura es poder establecer una relación entre la fuerza a medir y la resistencia eléctrica de la galga para deducir la magnitud de la fuerza aplicada [4].

Dadas las características de cada sensor anteriormente estudiadas, se puede ver que los sensores ópticos poseen alta sensibilidad en la medición, tienen baja dependencia de la temperatura [41][16], además que la electrónica de acondicionamiento de las señales se hace más compleja y aumenta su coste [4][10].

Las galgas extensiométricas presentan la ventaja de tener alta fiabilidad, al no requerir mayor mantenimiento, y para acondicionar sus señales sólo hace falta conectarlas a puentes de Wheatstone, lo que supone un menor coste. Pero, por el contrario, presentan susceptibilidad frente a cambios de temperatura, lo que incide negativamente en la calidad de la medición [10], para controlar esta desventaja hay que tener en cuenta el implementar un mecanismo que compense este efecto [24].

Teniendo en cuenta estas características, el sensor que mejor se adapta a las necesidades y especificaciones de este proyecto es el sensor de galgas extensiométricas. En las subsiguientes secciones y capítulos se ampliará más el estudio referente a este tipo de sensores.

2.2. Sensores de Galgas extensiométricas

Éste tipo de sensores de fuerza emplean galgas extensiométricas ligadas a la tensión de un elemento deformable elásticamente, las galgas extensiométricas son dispositivos que varían su resistencia eléctrica de acuerdo a la elongación o compresión que sufren, así pues, la flexión/tensión del elemento deformable elásticamente bajo la acción de una fuerza, se traduce en una variación de la resistencia eléctrica de la galga, se puede entonces establecer una relación entre la fuerza aplicada sobre el sensor y la resistencia de la galga.

Un sensor básicamente está compuesto por: el elemento sensor en este caso las galgas extensiométricas y un cuerpo elástico al que van pegadas las galgas y que se deforma elásticamente. Además también de la plataforma electrónica que hace el acondicionamiento de la señal y la procesa para calcular los componentes de fuerza y par.

Una galga extensiométrica es un arreglo en forma de bobina plana de un alambre conductor (Figura 2.4) cuya resistencia eléctrica viene dada por su longitud, su forma y su coeficiente de resistividad [4].

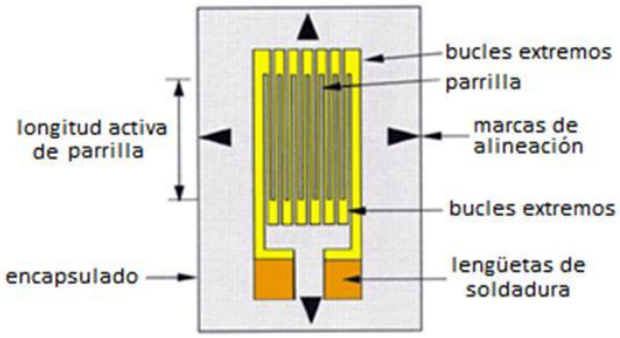

Figura 2.4: Galga extensiométrica

Tienen la propiedad de cambiar su resistencia eléctrica cuando están bajo el efecto de una acción de tensión, flexión o compresión, debido a que se está modificando la forma y la longitud del alambre conductor. Esto se explica de la siguiente manera, la resistencia de un alambre conductor de longitud l y área transversal A viene dada

por la siguiente expresión:

$$R = \rho \frac{l}{A} \tag{2.2}$$

Donde ρ es el coeficiente de resistividad del material. Al ocurrir un cambio en la resistividad del conducto, se tiene que:

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \frac{d\frac{l}{A}}{\frac{l}{A}} \tag{2.3}$$

Donde, el primer término del lado derecho depende del cambio de la resistividad del conductor, y el segundo término representa la deformación y se conoce como coeficiente de Poisson. Esto quiere decir que el cambio en la resistencia viene de un cambio de la forma y un cambio de la resistividad del material. Así pues, una galga extensiométrica varía su resistencia de acuerdo a la siguiente expresión:

$$\frac{\delta R}{R} = GF.\,\varepsilon \tag{2.4}$$

Donde la constante GF se conoce como factor de galga o sensibilidad de la galga, y ε es la tensión o esfuerzo sufrido por la galga.

Aprovechando la propiedad de la variación resistiva de la galga extensiométrica debido a la deformación sufrida, se puede establecer una relación entre estas dos para cuantificar la magnitud de la fuerza aplicada sobre la galga teniendo en cuenta que ésta deformación no puede llevar a la galga fuera de su margen elástico para evitar no linealidades y la rotura de la misma.

2.2.1. Estructura de los sensores de fuerza

En los sensores de fuerza se entiende como estructura la parte que sirve para transmitir la deformación sufrida por acción de las fuerzas o pares a los elementos sensores, también llamado en la literatura como cuerpo elástico. El cuerpo elástico debe estar hecho de un material con un coeficiente de elasticidad elevado y con poca histéresis, que le permita regresar a su estado original después de sufrir deformaciones, éste material suele ser generalmente aluminio o acero, aunque también hay prototipos realizados en acrílico [43] con las desventajas que esto puede suponer ya que el acrílico se comporta de manera elástica para un rango muy limitado, y puede llegar a sufrir deformaciones permanentes si se sobrepasa dicho margen. Para explicar la estructura de un sensor se puede partir de una viga elástica [1]. A partir de este elemento se pueden crear geometrías más elaboradas para por ejemplo tener más precisión o para incrementar los grados de libertad de la medición. Tenemos entonces por ejemplo, la estructura de viga de doble flexión (Figura 2.5.a), en la cual las galgas (1) van montadas en extremos opuestos de la viga (2). Este tipo de estructuras más simple es generalmente utilizado en las balanzas electrónicas, debido a que su deformación por fuerzas axiales es mucho más pequeño que a un momento flector.

La estructura de vigas paralelas (Figura 2.5.b) consiste en dos vigas elásticas paralelas (1) conectadas a dos elementos rígidos (2). Las galgas (3) están pegadas a la base de la viga para medir la flexión producida por la fuerza F [14].

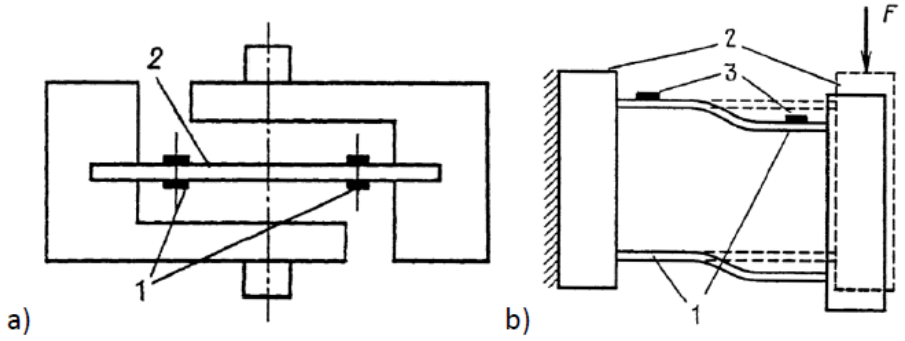

Figura 2.5: a) Doble viga de flexión, b) vigas paralelas.

Consideremos ahora los elementos usados para medir pares. Una manera simple de medir el par a través de la deformación de una viga en voladizo sometida a torsión (Figura 2.6.a). Para medir la torsión, las galgas deben estar pegadas a la viga formando un ángulo de 45º. Sin embardo esta configuración no asegura selectividad mecánica para el par medido M. En consecuencia, la influencia sobre la medición de acoplamientos de componentes no medidos puede ser significante.

Un sensor de par cruciforme hueco se muestra en la Figura 2.6.b. es una variación del de la Figura 2.5.b. El sensor tiene dos bridas circulares (1) conectadas por cuatro vigas elásticas (2). Las vigas están espaciadas 90º una respecto de la otra a la misma distancia del eje del sensor. Las galgas (3) están pegadas en el medio del lado lateral más ancho de las vigas, cerca de las bridas superior e inferior. Cada viga se dobla tangencialmente y gira debido al par axial M [14].

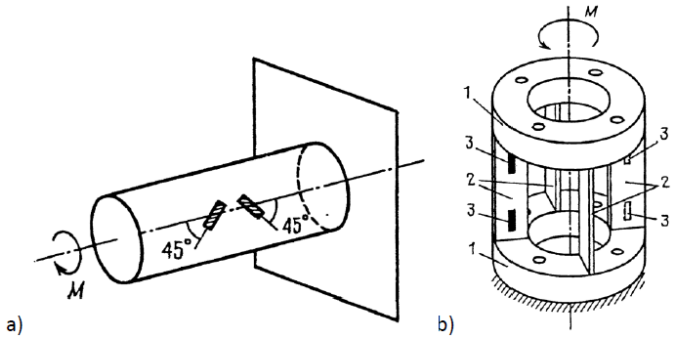

Figura 2.6: a) Estructura de viga en voladizo, b) estructura cruciforme.

Formando combinaciones con las estructuras como las vistas en la anteriormente se pueden conseguir estructuras de sensor que sirven para medir tanto fuerzas como pares.

Es de notar que las mediciones del sensor de fuerza no pueden ser usadas directamente en los algoritmos de control del robot, ya que éste describe las fuerzas equivalentes que actúan sobre el sensor las cuales difieren de las fuerzas aplicadas al efector final del manipulador. Es necesario entonces transformar estas fuerzas del sistema de referencia s del sensor al sistema de referencia c de las restricciones del efector final [36] como se observa en la Figura 2.7.

Hasta ahora se habían visto elementos elásticos básicos que solo responden a

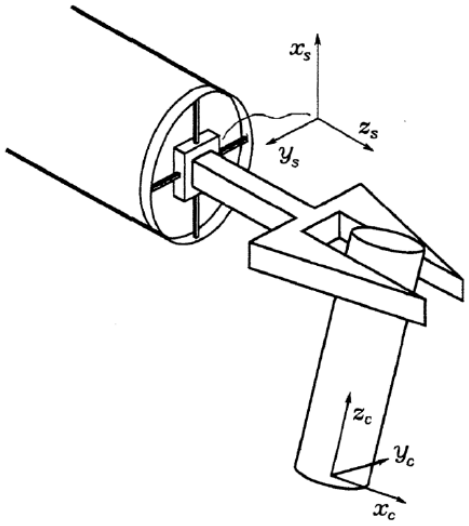

Figura 2.7: Sensor de tipo muñeca.

un único componente de fuerza o par. Un enfoque diferente para lograr sensores de múltiples componentes (grados de libertad) es el uso de elementos elásticos que respondan a varios componentes de fuerza y par simultáneamente. Por ejemplo un sensor que mida dos componentes, podría ser tan sencillo como dos vigas elásticas conectadas en el ángulo adecuado.

Una de las estructuras elásticos más comunes y ampliamente usadas para un sensor de tres componentes es la cruz de Malta visto en la Figura 2.8. Está compuesto por dos bridas, una interior (1) y otra exterior (2), conectados por cuatro vigas elásticas (3). Cuatro pares de galgas (4) pegadas a las superficies opuestas de la viga se conectan formando cuatro medios puentes. El sensor mide la fuerza axial F_z y los dos momentos laterales M_x y M_y que actúan en el plano del sensor. Los componentes de fuerza laterales F_x y F_y aplicados al centro del sensor causan deformaciones de compresión-tensión en las vigas pero el sensor no responde a estos componentes.

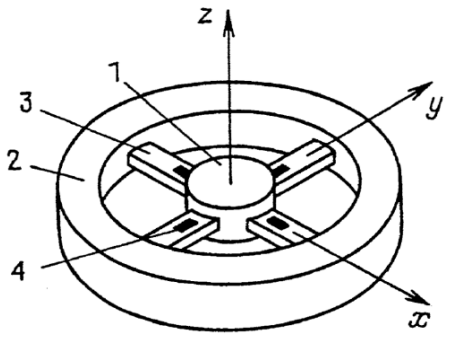

Figura 2.8: Elemento en forma de cruz de Malta.

Si las vigas son bastante delgadas, entonces la salida del sensor puede no ser lineal para cargas grandes, ya que puede sobrepasar el límite elástico del material. En los sensores de este tipo si las vigas están pegadas a las bridas por ejemplo con tornillos en lugar de maquinarse como un único elemento, tienden a presentar una

histéresis alta, ocasionada por pequeños desajustes en la estructura [14].

Un diseño más sofisticado de la cruz de Malta, para medir seis componentes de fuerza/par se muestra en la Figura 2.9. Está mecanizado a partir de una pieza monolítica de metal. Los extremos de las vigas (1) están soportados por placas delgadas (2) separadas de la brida exterior por franjas (3). Las galgas extensiométricas (4) están pegadas en medio de la superficie de la viga y cerca de la brida interior van conectadas formando ocho puentes. El brazo del manipulador va montado en la brida exterior y el efector final va montado en la brida interior [15][26][27][37].

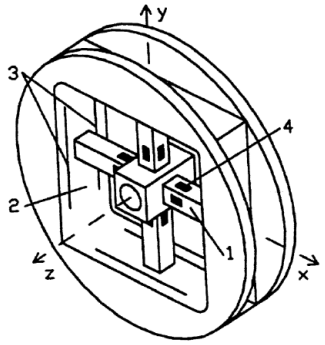

Figura 2.9: Elemento elástico en forma de cruz de Malta con soporte flexible.

En la Figura 2.10 se muestra el análisis de elementos finitos realizado a esta estructura, en la cual se puede apreciar de forma exagerada el efecto de deformación que presenta bajo la aplicación de un par en el eje z [29].

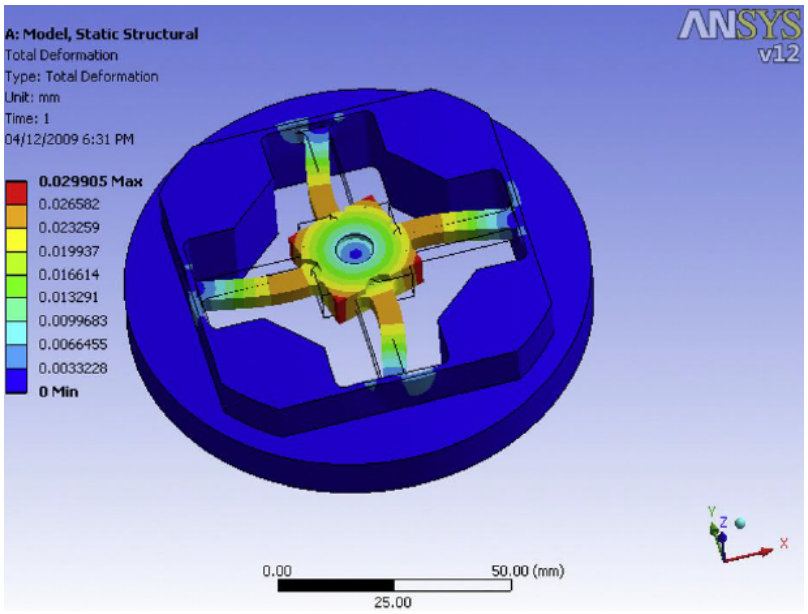

Figura 2.10: Elementos finitos de la estructura cruz de Malta.

Esta estructura de sensor es uno de los diseños más conocidos en términos de precisión, linealidad, y rango de medida. Debido a estas características, una variación de este tipo de estructura es el utilizado en este proyecto para el diseño del sensor, el diseño en lugar de tener cuatro vigas tiene tres, esto además disminuye los

costes de maquinado y de galgas para instrumentar las vigas, pero puede acarrear acoplamiento en algunos de los componentes de fuerza o par.

En la Figura 2.11 se muestran dos estructuras hechas con base a la cruz de Malta. En la Figura 2.11.a los elementos del sensor están mecanizados como una sola unidad, el anillo (1) está conectado por cuatro vigas radiales (2) al centro (3). El centro a su vez está conectado con un anillo (5) a través de cuatro vigas de soporte (4), y las galgas (6) están cableadas formando ocho puentes.

La Figura 2.11.b muestra una estructura con dos cruces de Malta, donde las vigas están unidad a dos bridas (1) y (2), y el centro (3). Las galgas (4) están pegadas a la superficie superior e inferior de cada viga y conectadas formando siete puentes [14].

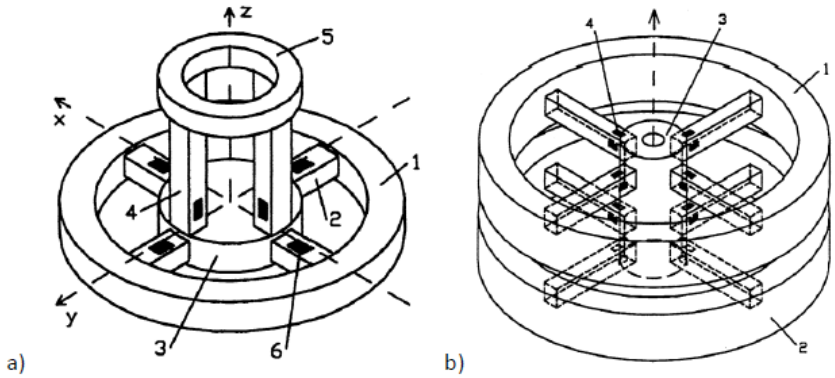

Figura 2.11: a) Elemento cruz de Malta conectado con un paralelogramo; b) elemento elástico de doble cruz de Malta.

También se pueden convertir las fuerzas y pares aplicados utilizando células de carga. Tal diseño de sensor se muestra en la Figura 2.12; posee dos bridas (1, 2) que están conectadas a través de seis células de carga (3, 4) colocados en ángulo con respecto al eje del sensor.

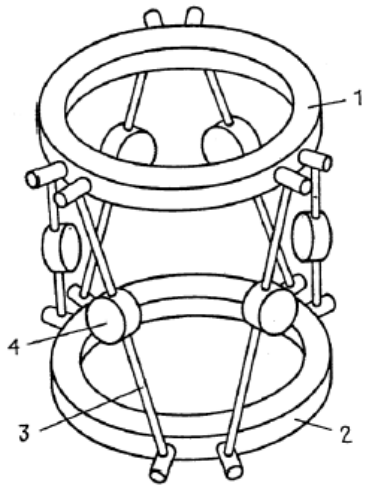

Figura 2.12: Estructura de sensor de seis componentes con células de carga.

Ejemplos de aplicaciones que usan esta estructura se pueden encontrar en [13]

[21] [30]. En la Figura 2.13 se presenta el análisis de elementos finitos nuevamente realizados en ANSYS a una estructura del tipo de plataforma de Stewart modificada, lo que permite simular el comportamiento que va a tener dicha estructura bajo los efectos de la aplicación de fuerzas o pares [30].

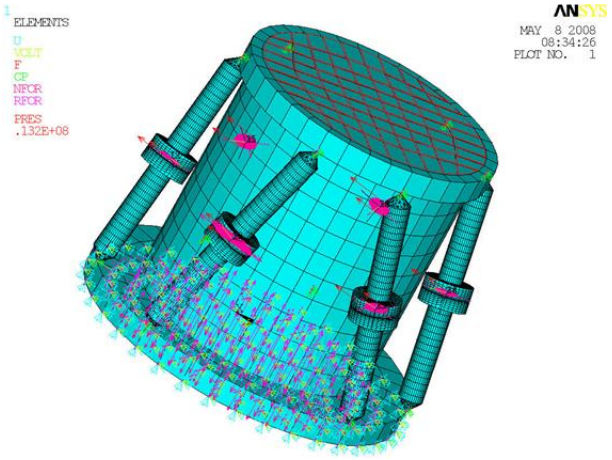

Figura 2.13: Elementos finitos de estructira Stewart.

En la Figura 2.14 se aprecia una estructura que parte de la utilización de soportes en forma de T que están bajo el efecto de una fuerza en el eje z. Está instrumentada con 20 galgas extensiométricas que sirven para calcular los seis grados de libertad. Los autores realizaron un análisis de elementos finitos para optimizar el diseño y el tamaño óptimo de los soportes que componen la estructura. Sin embargo, se encontraron con que esta estructura presenta un alto acoplamiento entre los seis componentes de fuerza del sensor [31].

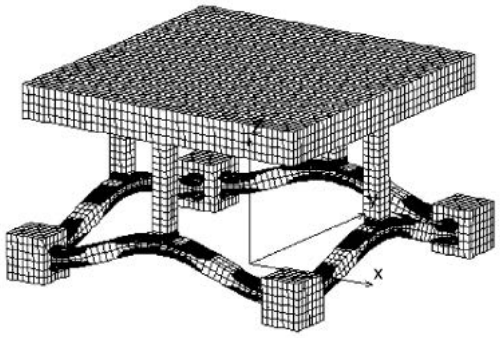

Figura 2.14: Estructura con soportes en forma de T.

En la Figura 2.15 se muestra una estructura que es una variación de una estructura en forma delta. Va instrumentada con 2 galgas extensiométricas por cada viga, en total cuenta con 6 galgas para medir los seis componentes de fuerza y par [32]. Se aprecia el detalle de la instrumentación de la viga, que va unida al anillo mediante un pin, lo que se ha identificado según los autores como un punto débil y que causa un fallo en la medición al no transmitir correctamente a las galgas las fuerzas aplicadas a la estructura del sensor.

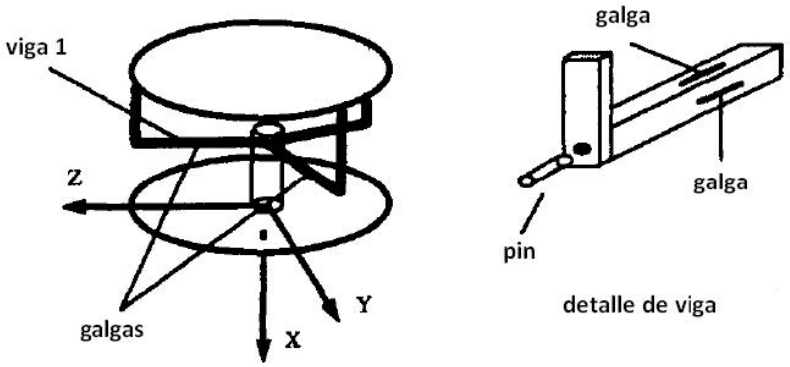

Figura 2.15: Estructura de sensor tipo delta.

En la Tabla 2.1 se presenta un resumen de las características más relevantes de las estructuras de los sensores de fuerza estudiados con anterioridad. Se puede notar una relación proporcional entre la cantidad de grados de libertad, la cantidad de galgas necesarias para conseguirlos y la complejidad de construcción del diseño. Por ejemplo, para obtener un sensor con seis grados de libertad, la estructura más ampliamente utilizada es la de Cruz de Malta [14] y mínimo 8 galgas, generalmente se utilizan 16 galgas [5] para obtener mayor sensibilidad, lo que incrementa el coste de diseño y construcción, respecto a si se utilizara una estructura de tipo delta con 12 galgas.

Tabla 2.1: Resumen de características de las estructuras más comunes.

Tipo	Cantidad de grados de libertad	Cantidad de Galgas	Complejidad de construcción
Doble viga de flexión (Figura 2.5a)	1 f_x	2	Baja
Vigas paralelas (Figura 2.5.b)	1 f_x	2	Baja
Viga en voladizo (Figura 2.6a)	1 M_z	2	Baja
Cruciforme (Figura 2.6b)	1 M_z	4	Media
Cruz de Malta (Figura 2.8)	3 f_x, f_y, f_z	4	Media
Cruz de Malta (Figura 2.9)	6 f_x, f_y, f_z M_x, M_y, M_z	8	Media
Cruz de Malta con Paralelogramo (Figura 2.11a)	6 f_x, f_y, f_z M_x, M_y, M_z	8	Alta
Doble cruz de Malta (Figura 2.11b)	6 f_x, f_y, f_z M_x, M_y, M_z	16	Alta
Plataforma Stewart (Figura 2.12)	6 f_x, f_y, f_z M_x, M_y, M_z	6	Alta
Viga en forma de T (Figura 2.14)	6 f_x, f_y, f_z M_x, M_y, M_z	20	Alta
Delta (Figura 2.15)	6 f_x, f_y, f_z M_x, M_y, M_z	6	Media

El criterio para decidir qué tipo de estructura se ha de utilizar en el diseño y elaboración de un sensor de fuerzas viene dado por la cantidad de grados de libertad que se necesite medir y también, el lograr un balance entre complejidad (coste) y sensibilidad del sensor.

2.3. Formulación matemática del sensor de fuerzas y par

En general, un sensor de fuerzas de galgas extensiométricas, se vale del comportamiento lineal de estructuras hechas generalmente de aleaciones de metal, para transformar los desplazamientos causados por la aplicación de fuerza/pares, en señales eléctricas medibles.

Considerando una estructura a la que se le aplica en un punto particular un vector de fuerza desconocido $\vec{F}^T = F_x, F_y, F_z, M_x, M_y, M_z$, que se mantiene en su rango lineal de elasticidad; además, si dicha estructura se instrumenta con un conjunto de galgas extensiométricas ubicadas en su superficie, se puede formar un vector de las mediciones de las galgas llamado $\vec{S}_s^T = S_1, S_2, S_3, ..., S_n$ provocadas por la fuerza $\vec{F}$.

De acuerdo al principio de superposición y a que la estructura se comporta en el rango elástico del material, se produce la siguiente relación lineal entre los vectores $\vec{F}$ y $\vec{S}$ [38]:

$$\vec{S}_s = [A] \cdot \vec{F} \tag{2.5}$$

Donde $[A]$ es la matriz de acomodación con dimensión $n \times 6$. Esta matriz relaciona las mediciones de las galgas con los componentes de la fuerza $\vec{F}$ que provocan dichas mediciones.

Luego, para encontrar el valor de la fuerza desconocida es necesario hacer un procedimiento de inversión:

$$\vec{F} = [A]^{-1} \cdot \vec{S}_s = [C] \cdot \vec{S}_s \tag{2.6}$$

Donde $[C] = [A]^{-1}$ es conocida como la matriz de calibración, que representa una matriz que directamente se multiplica al vector de tensiones para obtener el vector de fuerza $\vec{F}$ que debe ser medido. Cuando $\vec{S}_s$ contiene más de seis elementos resultando $[C]$ en una matriz no cuadrada, no se puede entonces hacer la inversión directa, por lo cual se debe emplear la técnica de Moore-Penrose [7] para hacer la inversión de la matriz y así determinar $\vec{F}$:

$$\vec{F} = [C] \cdot \vec{S}_s = ([A]^T \cdot [A])^{-1} \cdot [A]^T \cdot \vec{S}_s \tag{2.7}$$

La matriz $[C]$ puede ser obtenida analíticamente usando un método de análisis de la estructura (geometría) o experimentalmente usando un método de calibración, mediante la aplicación de una fuerza pura individualmente por cada componente y luego la organización de las tensiones obtenidas [38]. Dado que los componentes del vector de fuerzas $\vec{F}$ son fuerzas y pares, los valores de las tres primeras columnas de la matriz $[C]$ y las tres últimas son las tensiones inducidas por unidad de fuerza y por

unidad de par respectivamente. Éste procedimiento se repite para cada componente que el transductor debe medir.

2.4. Calibración

Como se describió anteriormente, el proceso de calibración es un proceso experimental, en el que se aplican fuerzas puras a cada uno de los componentes que tiene el sensor. La matriz de calibración se puede construir mediante la aplicación de un número suficiente de fuerzas conocidas $\{Y\}$ y midiendo las correspondientes tensiones $\{X\}$, como se muestra en la Tabla 2.3 [6]. A continuación se describirá el proceso de calibración para un sensor de 6 grados de libertad y n galgas extensiométricas.

Tabla 2.2: Procedimiento de toma de datos para la calibración del sensor.

Etapa	Aplicado	Medido
1	$y_{11}y_{21}y_{31}\cdots y_{61}$	$x_{11}x_{21}x_{31}\cdots x_{n1}$
2	$y_{12}y_{22}y_{32}\cdots y_{62}$	$x_{12}x_{22}x_{32}\cdots x_{n2}$
3	$y_{13}y_{23}y_{33}\cdots y_{63}$	$x_{13}x_{23}x_{33}\cdots x_{n3}$
$\vdots$	$\vdots$	$\vdots$
p	$y_{1p}y_{2p}y_{3p}\cdots y_{6p}$	$x_{1p}x_{2p}x_{3p}\cdots x_{np}$
$\cdots$	$Y_{(p\times 6)}$	$X_{(n\times p)}$

Se obtiene una cantidad suficiente de muestras p, que para un sensor de seis grados de libertad p tendría que ser igual o mayor a 6, en forma general p debe ser igual o mayor a la cantidad de grados de libertad que tenga el sensor.

En analogía con la ecuación 2.7 $\vec{F} = [C] \cdot \vec{S}_s$, se tiene que:

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{61} & y_{62} & \cdots & y_{6p} \end{bmatrix}_{(6\times p)} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{61} & c_{62} & \cdots & c_{6n} \end{bmatrix}_{(6\times n)} \times \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{61} & x_{62} & \cdots & x_{6p} \end{bmatrix}_{(n\times p)}$$

O lo que es lo mismo:

$$[Y]^T = [C] \cdot [X]^T \tag{2.8}$$

Transponiendo queda:

$$[Y] = [X] \cdot [C]^T$$

Multiplicando a ambos lados de la igualdad por $[X]^T$ obtenemos:

$$[X]^T \cdot [X] \cdot [C]^T = [X]^T \cdot [Y]$$

Y despejando $[C]$:

$$[C]^T = [([X]^T \cdot [X])^{-1} \cdot [X]^T \cdot [Y]] \tag{2.9}$$

Donde $[C]$ se denomina matriz de calibración. Éste método, debido a que es simple de implementar y a que ofrece buenos resultados es el más utilizado para hacer la calibración de los sensores de fuerza y par, según lo encontrado en la siguiente bibliografía [5][6][7][20][22][31][32][33][34][35][38][42]. Para lograr una medición fiable se han de tomar precauciones en la instrumentación del sensor, como lo es que las galgas deben estar colocadas de tal manera que por lo menos un elemento se deforme apreciablemente ante cualquier posible orientación de fuerzas y momentos.

Además, un componente de fuerza debe inducir el menor número posible de deformaciones en la estructura para lograr tener desacoplamiento entre componentes [14]. Una matriz de calibración simple ahorra considerablemente tiempo de cálculo y hace posible obtener un control en tiempo real del robot [44].

2.5. Conversión de la medida

Para medir la deformación generalmente se utiliza un puente de Wheatstone [4], habitualmente formado por tres resistencias iguales y una resistencia que se desconoce. En el caso del sensor de fuerza se forma utilizando cuatro galgas extensiométricas como se explica en la Figura 2.16, a esta configuración se le llama puente completo, al estar compuesto por 4 galgas extensiométricas.

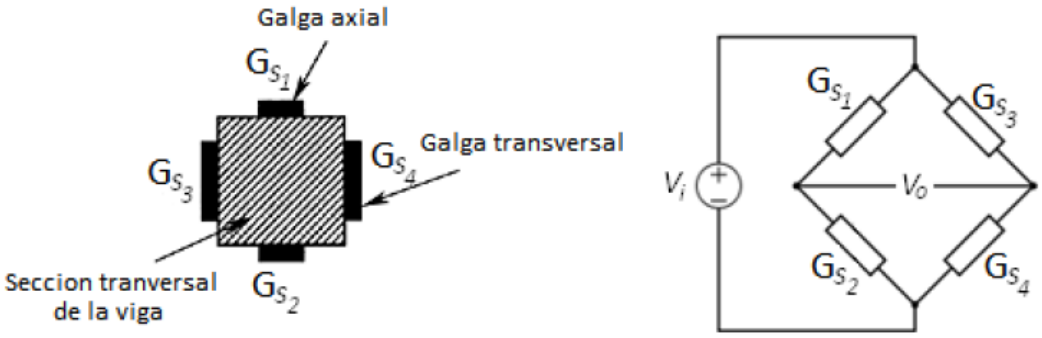

Figura 2.16: Ubicación de las galgas en el soporte y puente de Wheatstone.

Para medir la magnitud de la fuerza que genera la deformación, se dividen las cuatro galgas en dos medios puentes y en un brazo del puente se ponen las galgas que están expuestas a la aplicación de una fuerza que las deforma, éstas galgas son las que están en lados opuestos de la viga elástica, y se completa el puente con dos galgas libres de deformaciones por fuerzas (Figura 2.17).

Ésta configuración también sirve para compensar las variaciones de las galgas debido a la temperatura ya que el puente completo se hace sensible a la temperatura y no sólo una parte de él lo que hace que la salida diferencial del puente se mantenga estable frente a variaciones de temperatura, además que proporciona mayor sensibilidad de las medidas porque por un lado de la viga se hace una tensión

y por el otro una compresión de las galgas, aumentando la variación del voltaje de salida en el puente [31].

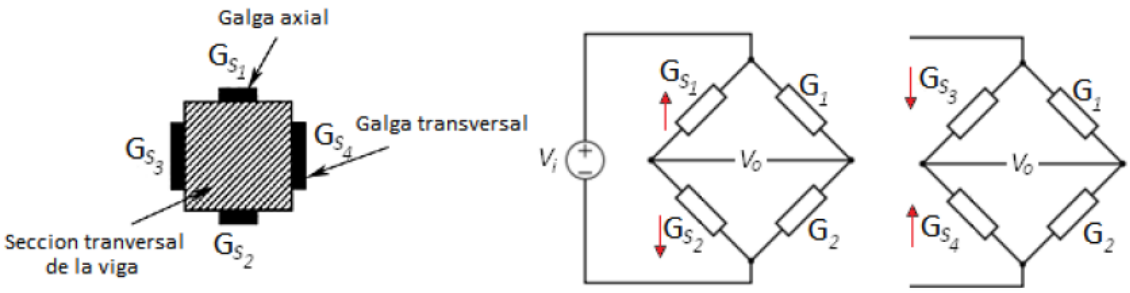

Figura 2.17: Configuración de puente completo con compensación de temperatura.

El voltaje V_0 resultante del puente viene dado por la siguiente ecuación:

$$V_0 = \left(\frac{R_{S2}}{(R_{S1} + R_{S2})} - \frac{R_{S4}}{(R_{S3} + R_{S4})} \right) \cdot V_i \tag{2.10}$$

Capítulo 3

Desarrollo del sensor

El sensor desarrollado ha sido diseñado para ir montado entre el brazo del robot y el efector final, es decir, se trata de un sensor de muñeca que es capaz de hacer mediciones en seis grados de libertad. En la Figura 3.1 se puede apreciar la ubicación del sensor en el brazo robot y las etapas de acondicionamiento y adquisición de las señales, procesamiento de las señales y presentación de los datos en la interfaz de usuario.

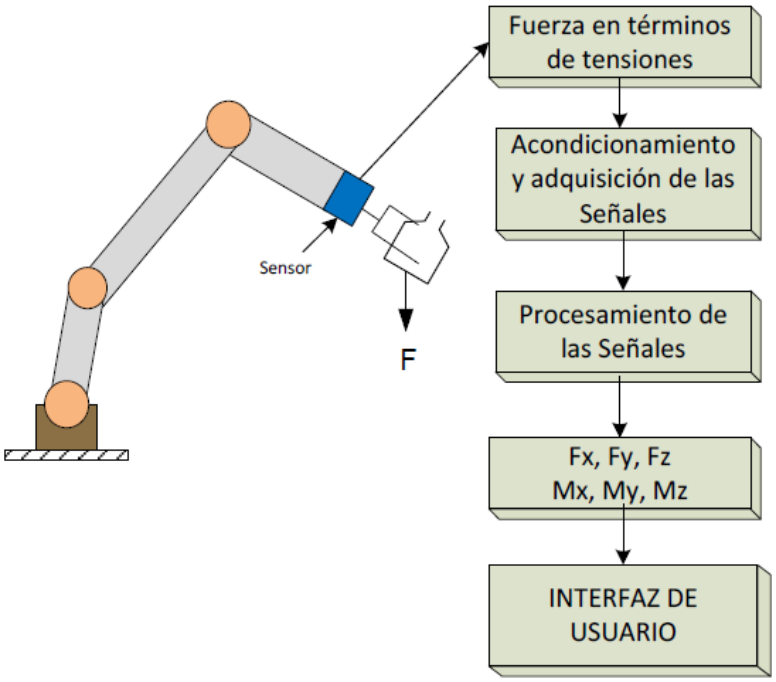

Figura 3.1: Diagrama de funcionamiento del sensor.

3.1. Estructura del sensor

Teniendo en cuenta los factores de coste y las características se ha optado por fabricar una estructura para el sensor como la mostrada en la Figura 3.2, se ha instrumentado con 4 galgas extensiométricas por cada viga, es decir, que en total

cuenta con 12 galgas extensiométricas para la medición de las flexiones de la estructura, además, se han colocado 2 galgas extensiométricas más en la estructura que sirven para completar los puentes de Wheatstone y así compensar los efectos de las variaciones de temperatura sobre la medición.

La estructura utilizada para el sensor se fabricó utilizando control numérico a partir de un trozo monolítico de aluminio, está compuesta por un anillo exterior unido a una brida interior o centro mediante tres vigas axiales en un ángulo de 120º una respecto de la otra, como se ve en la Figura 3.2. Su diseño se realizó en aluminio debido a que este metal posee un coeficiente de elasticidad aceptable, además de tener buena maleabilidad que permite realizar un mecanizado rápido disminuyendo el coste en un 60 % con respecto al mecanizado del acero que aunque tiene un coeficiente de elasticidad más elevado se debe debilitar de alguna manera, por ejemplo haciendo las vigas más delgadas o haciéndolas huecas para disminuir un poco su alta rigidez.

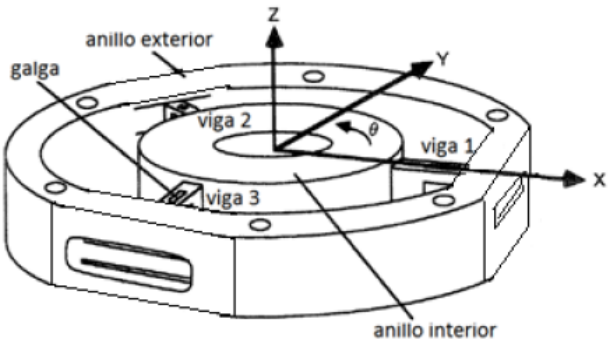

Figura 3.2: Estructura del sensor de fuerza utilizada.

Como se aprecia es una variación de la estructura mostrada en la Figura 2.9, cruz de Malta con vigas soportadas por placas delgadas pero cuenta con tres vigas en lugar de cuatro. El cuerpo del sensor ha sido construido con las siguientes dimensiones (Figura 3.3):

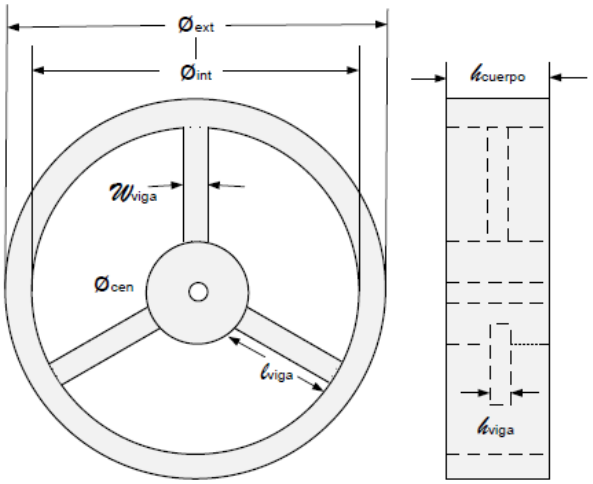

Figura 3.3: Dimensiones del transductor de fuerza/par desarrollado.

- Diámetro de anillo exterior ϕ_{ext}: 74 mm.
- Diámetro del anillo interior ϕ_{int}: 55mm.
- Diámetro del anillo central ϕ_{cen}: 20 mm.
- Altura del cuerpo h_{cuerpo}: 20 mm.
- Longitud de la viga l_{viga}: 20mm.
- Ancho de la viga W_{viga}: 2.60 mm.
- Altura de la viga h_{viga}: 5 mm.

Teniendo en cuenta la mecánica de vigas y haciendo un análisis de elasticidad de la estructura se puede obtener de manera teórica la respuesta del cuerpo elástico cuando éste se somete a fuerzas o pares. Como se puede observar en la Figura 3.4 una viga se puede someter a torsiones o flexiones y las ecuaciones que describen su comportamiento bajo estas situaciones son:

Para torsión:

$$d = \frac{4FL^3}{Eah^3} \tag{3.1}$$

Despejando la fuerza de (3.1) queda:

$$F = \frac{Eadh^3}{4L^3} \tag{3.2}$$

Para flexión:

$$\alpha = \frac{2\Pi L}{\pi G r^4} \tag{3.3}$$

Despejando el par de (3.3), resulta:

$$\Pi = \frac{\alpha \pi G r^4}{2L} \tag{3.4}$$

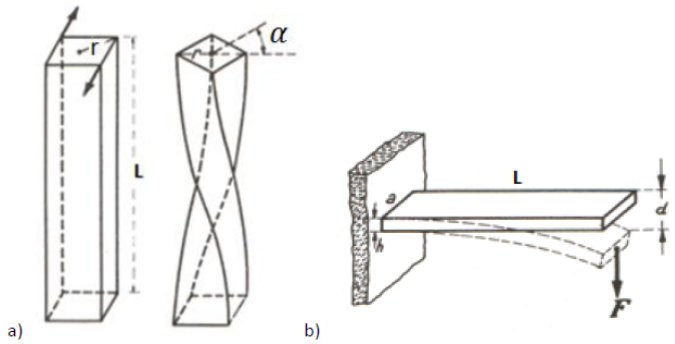

Figura 3.4: a) Torsión y b) flexión de una viga.

Realizando un análisis de la estática de la estructura (Figura 3.2) se obtienen los componentes de fuerzas y pares:

$$\begin{cases} F_x = f_{1x} + \Pi_{2x} + \Pi_{3x} + f_{2x} + f_{3x} \\ F_y = f_{1y} + \Pi_{2y} + \Pi_{3y} + f_{2y} + f_{3y} \\ F_z = f_{1y} + f_{2y} + f_{3y} \\ \Pi_x = \Pi_{1x} + f_{2x} + f_{3y} \\ \Pi_y = f_{1y} + \Pi_{2y} + \Pi_{3y} \\ \Pi_z = f_{1z} + f_{2z} + f_{3z} \end{cases} \tag{3.5}$$

Donde:

$$f_{ij} = \frac{Eadh^3}{4L^3} \tag{3.6}$$

$$\Pi_{ij} = \frac{\alpha\pi G r^4}{2L} \tag{3.7}$$

Sea $i = 1, 2, 3$ cada una de las vigas de la estructura del sensor y $j = x, y, z$ cada uno de los ejes ordenados con origen en el centro de la estructura. Y cada símbolo representa:

F	Fuerza a la que está sometido (N)
Π	Momento de fuerza (N·m)
α	Ángulo de torsión (radianes)
G	Módulo de rigidez (N/m2)
L	Longitud (m)
π	Constante Pi
r	Radio (m)
E	Módulo de Young (N/m2)
h	Grosor (m)
a	Ancho (m)
d	Deformación (m)

3.2. Matriz de calibración

La matriz de calibración se obtiene mediante el método descrito por [38] y ampliamente utilizado como se comprueba con la abundante referencia bibliográfica encontrada [2][3][5][6][7][16][22][28][32][34][35][42][45]. Para calcular la matriz de calibración se ha seguido éste método que ya fue descrito en la sección 2.2, que consiste en aplicar una fuerza por cada uno de los componentes y medir el voltaje generado a la salida del puente de galgas, y con estos datos calcular la matriz de calibración del sensor de fuerza.

Para calibrar el sensor de fuerza se construyó un banco de pruebas con varias masas conocidas para que cada una ejerza una fuerza de 2, 5 y 10 newtons y en él se

tomaron 36 muestras por cada eje coordenado tanto por el lado positivo y negativo del eje. En total se tomaron 684 muestras de las respuestas que da el sensor a cada una de las fuerzas.

Para calcular la matriz de calibración se siguió el procedimiento de inversión mediante Moore-Penrose descrito por la ecuación (9) y utilizando las 684 muestras tomadas, dando como resultado la siguiente matriz:

$$C = \begin{bmatrix} 0{,}0319 & -0{,}0093 & -0{,}0104 & -0{,}0016 & -0{,}0172 & 0{,}0066 \\ -0{,}0029 & -0{,}0149 & -0{,}0170 & 0{,}0021 & 0{,}0226 & 0{,}0100 \\ 0{,}0318 & -0{,}0503 & 0{,}0572 & -0{,}0692 & 0{,}0627 & -0{,}0322 \\ 0{,}0319 & -0{,}0093 & -0{,}0104 & -0{,}0016 & -0{,}0172 & 0{,}0066 \\ -0{,}0029 & -0{,}0149 & -0{,}0170 & 0{,}0021 & 0{,}0226 & 0{,}0100 \\ 0{,}0318 & -0{,}0503 & 0{,}0572 & -0{,}0692 & 0{,}0627 & -0{,}0322 \end{bmatrix}$$

En su forma completa a partir de (6) queda:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ \Pi_x \\ \Pi_y \\ \Pi_z \end{bmatrix} = \begin{bmatrix} 0{,}0319 & -0{,}0093 & -0{,}0104 & -0{,}0016 & -0{,}0172 & 0{,}0066 \\ -0{,}0029 & -0{,}0149 & -0{,}0170 & 0{,}0021 & 0{,}0226 & 0{,}0100 \\ 0{,}0318 & -0{,}0503 & 0{,}0572 & -0{,}0692 & 0{,}0627 & -0{,}0322 \\ 0{,}0319 & -0{,}0093 & -0{,}0104 & -0{,}0016 & -0{,}0172 & 0{,}0066 \\ -0{,}0029 & -0{,}0149 & -0{,}0170 & 0{,}0021 & 0{,}0226 & 0{,}0100 \\ 0{,}0318 & -0{,}0503 & 0{,}0572 & -0{,}0692 & 0{,}0627 & -0{,}0322 \end{bmatrix} \cdot \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{bmatrix}$$

Se puede comprobar que para todo valor del vector de muestreo s corresponde un valor de fuerzas y pares.

3.3. Tarjeta de adquisición de datos

Teniendo en cuenta que las galgas extensiométricas son sensibles a la temperatura, y como medida de compensación, un brazo de los puentes de Wheatstone está formado por dos galgas libres de tensiones y el otro brazo del puente está formado por dos galgas situadas a los lados opuestos de cada viga, exponiéndose todas las galgas a las mismas variaciones de temperatura. Ahora bien, para acondicionar las señales que vienen de los puentes de galgas extensiométricas se ha utilizado un amplificador de instrumentación (INA128) que amplifica las señales para después capturarlas con un convertidor analógico a digital. La captura de datos se hace multiplexando la entrada del convertidor analógico a digital a cada una de las salidas de los seis puentes que forman las 14 galgas distribuidas en el cuerpo elástico. La adquisición y procesamiento de señales, y comunicación de datos mediante puerto USB con la interfaz de usuario se lleva a cabo usando un microcontrolador con características de DSP (dsPIC30f2011), que trabaja a su máximo de 30Mips.

El diagrama del sistema se muestra en la Figura 3.5. Consiste en un conjunto de galgas dispuestas sobre la estructura elástica que son multiplexadas en pares para con otras dos galgas formar un puente completo cuya salida se mide mediante un amplificador de instrumentación y se captura con un conversor analógico a digital. Teniendo en cuenta que el ADC sólo puede capturar voltajes de 0 a Vref, antes de cada captura del ADC, el amplificador de instrumentación se debe referenciar a un

voltaje tal que su salida esté en este rango, para lo cual primero se debe ejecutar una rutina de ajuste de cero (descrita más adelante). Para poner la referencia del amplificador de instrumentación se utiliza un convertidor digital a análogo y un amplificador operacional. Luego los datos son transmitidos a la interfaz de usuario utilizando el protocolo USB, después, la interfaz se encarga del control del robot utilizando los datos recibidos.

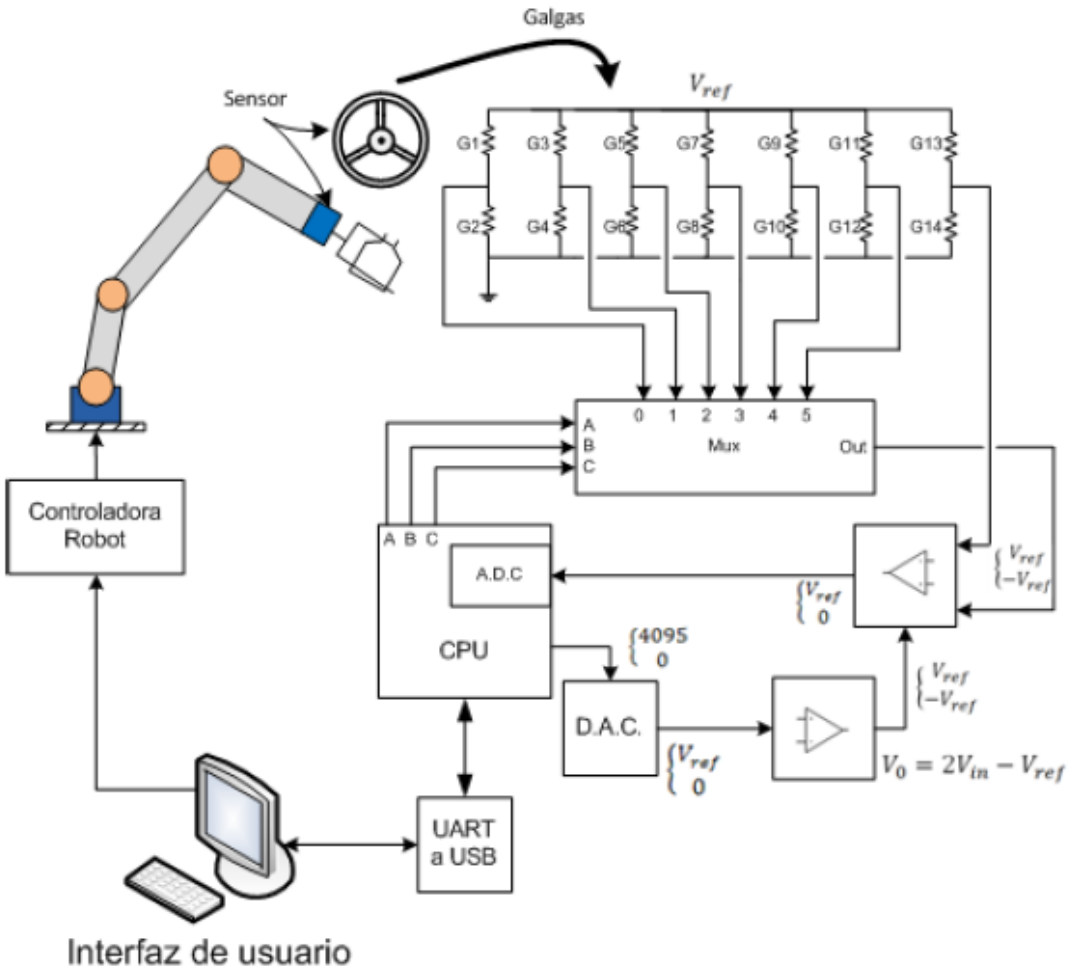

Figura 3.5: Diagrama conceptual del sistema.

El diagrama esquemático de la Figura 3.6, muestra los componentes utilizados para la construcción de la placa de acondicionamiento, adquisición y procesamiento de las señales de los puentes de galgas.

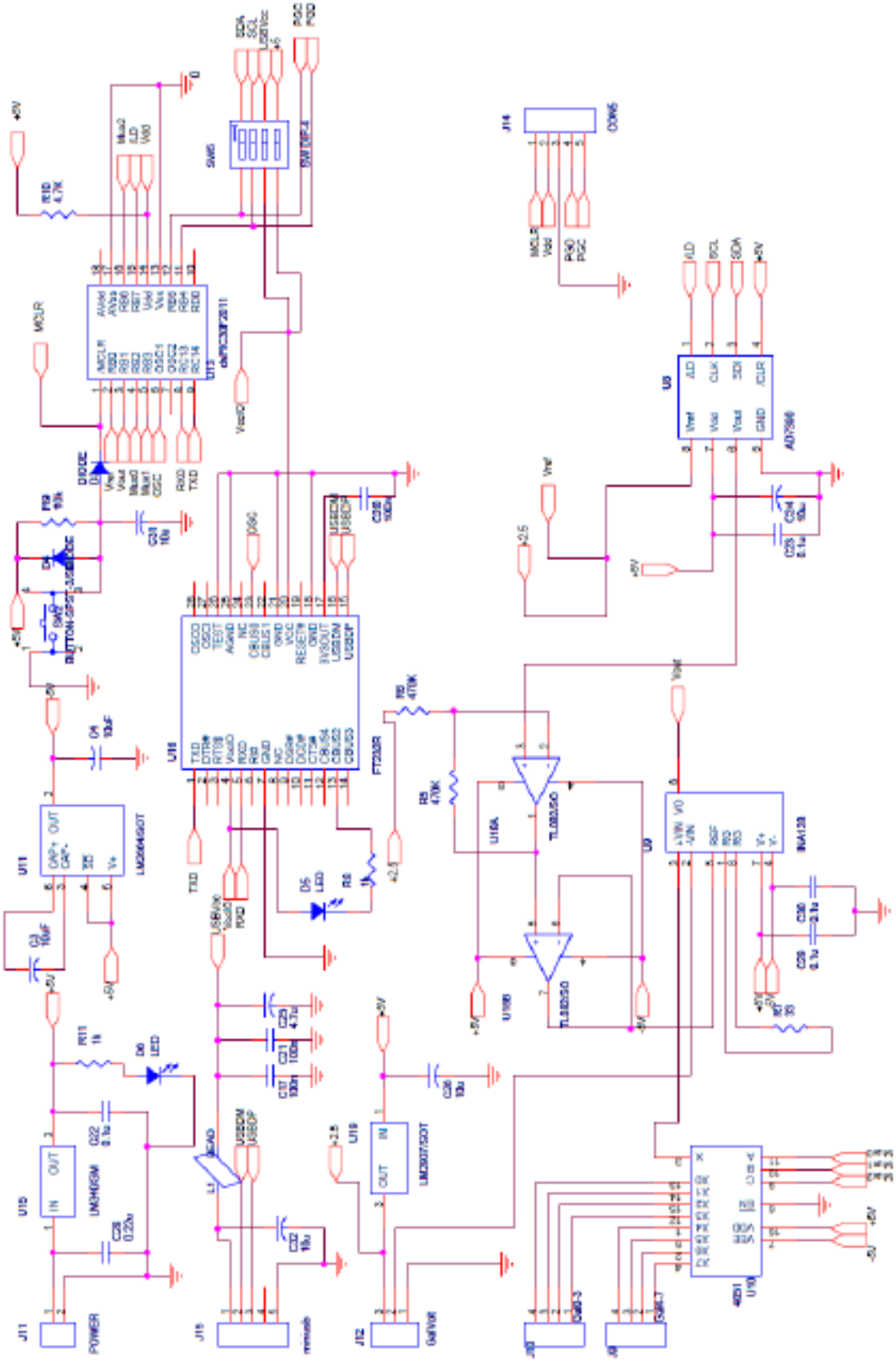

Figura 3.6: Esquemático de la tarjeta de adquisición de datos.

La placa del circuito impreso de la tarjeta de acondicionamiento y adquisición de señales se puede observar en la Figura 3.7. La dimensión es la misma que el cuerpo del sensor para que de esta manera queden unidos y formen un solo conjunto.

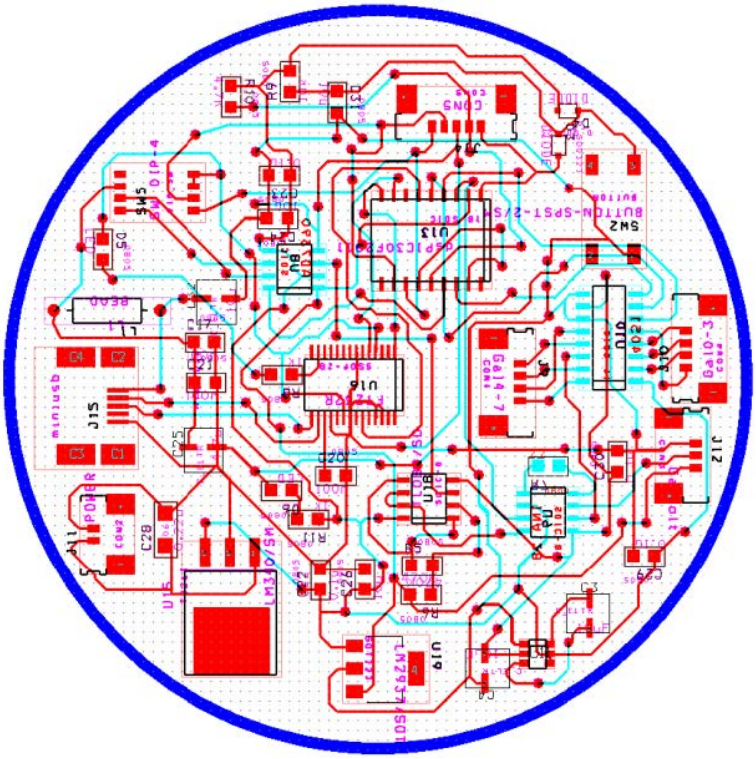

Figura 3.7: Circuito impreso de la tarjeta de adquisición de datos.

El hardware del prototipo del sensor de fuerzas de seis grados de libertad se muestra en la Figura 3.8.

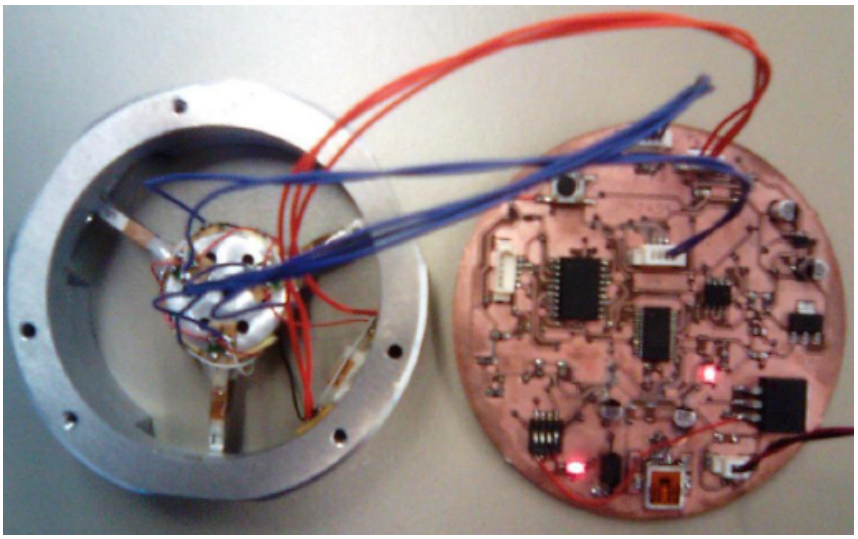

Figura 3.8: Resultado final del sensor de fuerza/par.

3.4. Programación

El procedimiento de ajuste del cero del sensor se realiza para compensar los efectos de la temperatura sobre las galgas que generan una variación de voltaje en los puentes. Este proceso consiste en medir los voltajes de salida de los puentes con el sensor libre de la acción de fuerzas o pares.

Este procedimiento se detalla en el diagrama de bloques de la Figura 3.9. Primero se inicializa un contador para cada uno de los puentes, en este caso se tienen 6 puentes de galgas. Se carga en el DAC el máximo valor posible, en este caso se cuenta con un DAC de 12 bits, entonces el máximo valor es 4095 en decimal, cuya salida tendrá que ser Vref. Luego se multiplexa a la entrada del amplificador de instrumentación el primer brazo del puente para completarlo con el brazo que está libre de deformaciones causadas por fuerza o pares. A continuación se mide el voltaje de salida del puente, si este voltaje es 2047, es decir Vref/2 se guarda su valor, si no, se decrementa el valor del DAC y se continua midiendo el voltaje hasta que éste llegue a Vref/2 y se guarda el valor del DAC, luego, se incrementa el contador y se continúa con los siguientes puentes.

Por medio de este procedimiento se construye un vector de 6 valores, cada uno es el valor que se debe poner en el DAC antes de realizar la medición de cada puente de galgas.

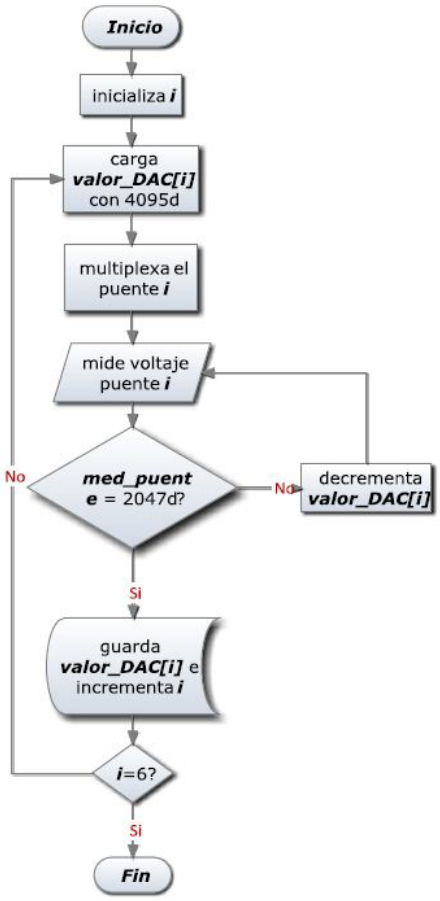

Figura 3.9: Diagrama de bloques rutina de ajuste del cero del sensor.

Después de tener ajustado el cero del sensor, se puede empezar a utilizar para realizar las mediciones de los puentes de galgas.

En este caso se ha diseñado un bucle infinito que transmite las lecturas hechas a los puentes de galgas del sensor (Figura 3.10). El bucle empieza inicializando un contador para cada uno de los puentes, luego se carga el valor del DAC correspondiente al puente que fue guardado previamente en el proceso anterior de ajuste a cero, se multiplexa el puente en cuestión y se mide su voltaje, después y se incrementa el contador para continuar con los siguientes puentes, después se transmite al usuario. Cuando se ha terminado de hacer la medición de todos los puentes, se empieza el procedimiento de nuevo.

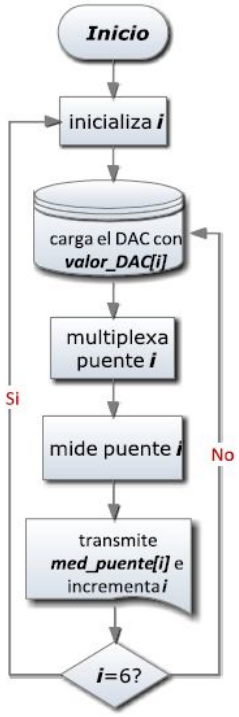

Figura 3.10: Diagrama de bloques bucle infinito de medición de los puentes.

Capítulo 4

Análisis de los resultados obtenidos

Para la calibración del sensor de fuerza se realizó una toma de muestras como se explicó anteriormente, en la Figura 4.1 y en la Figura 4.2 se muestra un gráfico del promedio de dichas muestras para una fuerza positiva y negativa respectivamente. En ellas se puede notar la linealidad que tienen las variaciones en cada eje coordenado tanto para valores positivos como para valores negativos de dichas fuerzas.

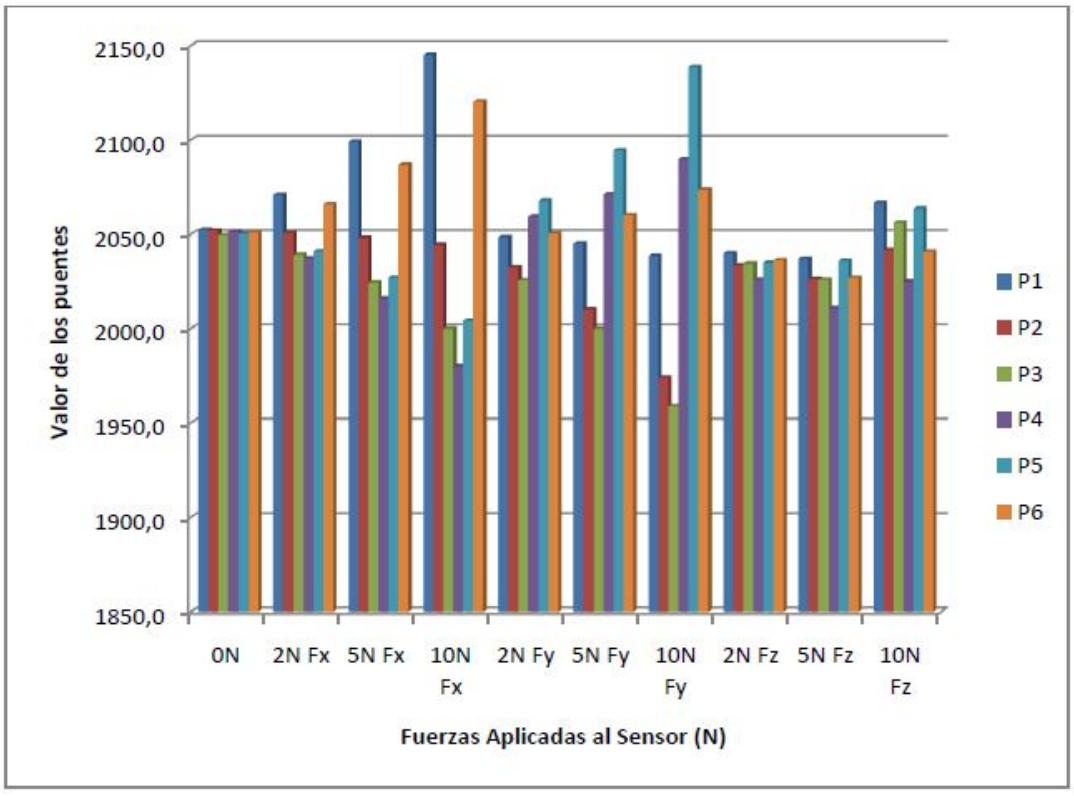

Figura 4.1: Valor promedio de las muestras tomadas para la calibración, aplicando fuerzas puras positivas en cada eje coordenado.

Esta linealidad es la que ha permitido obtener un sensor de fuerza y par estable

en la medida, ya que se ha comprobado en el banco de pruebas que no varía la medición de las fuerzas, es decir que la respuesta es estable.

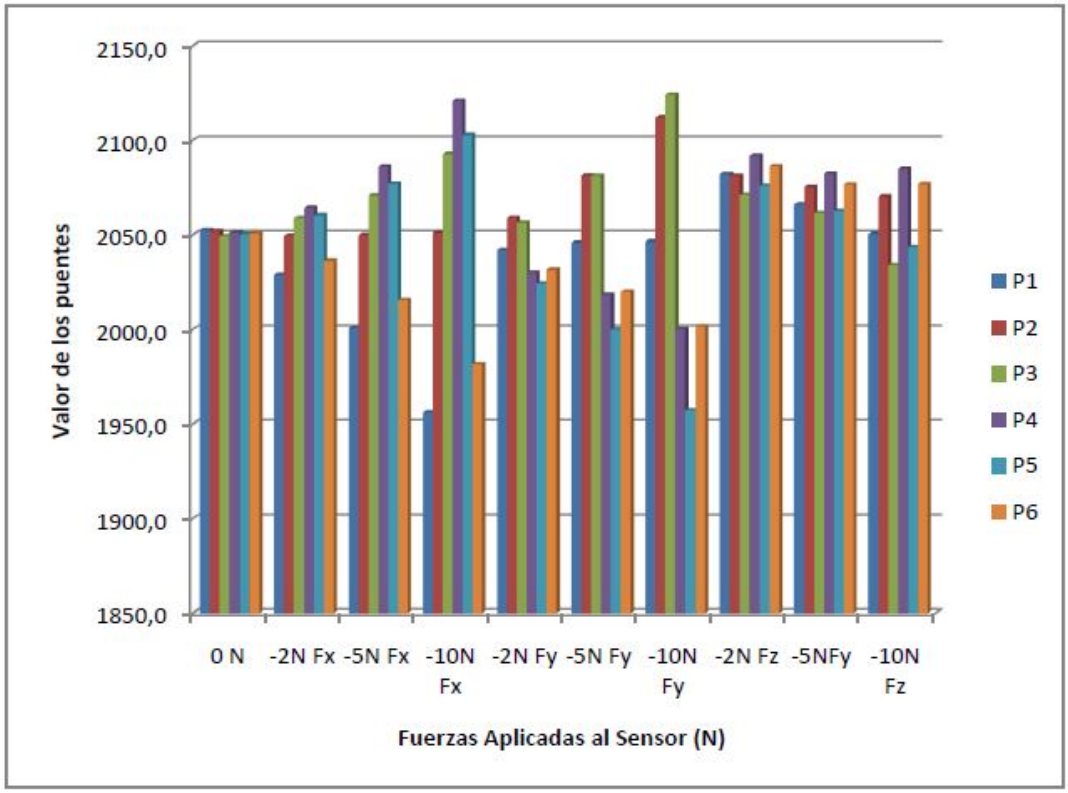

Figura 4.2: Valor promedio de las muestras tomadas para la calibración, aplicando fuerzas puras negativas en cada eje coordenado.

Para la validación del diseño del sensor de fuerzas y pares, se ha desarrollado una aplicación de control de un robot SCARA como el visto en la Figura 4.3, el control se hace a partir de los datos de las fuerzas que actúan sobre el sensor y que son interpretadas por la interfaz de control para generar los comandos de control para el movimiento del robot.

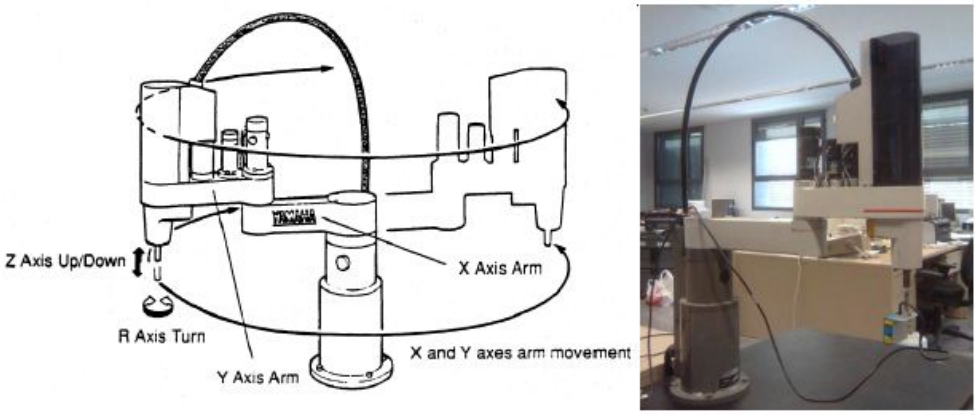

Figura 4.3: Robot SCARA de Yamaha.

El sensor de fuerzas y pares se comunica con la interfaz de control a través del puerto USB del ordenador para obtener los datos de las mediciones de los puentes de galgas. Dichos datos se pueden apreciar en la Figura 4.4, que también cuenta con una representación visual en forma de barras para los componentes de fuerza.

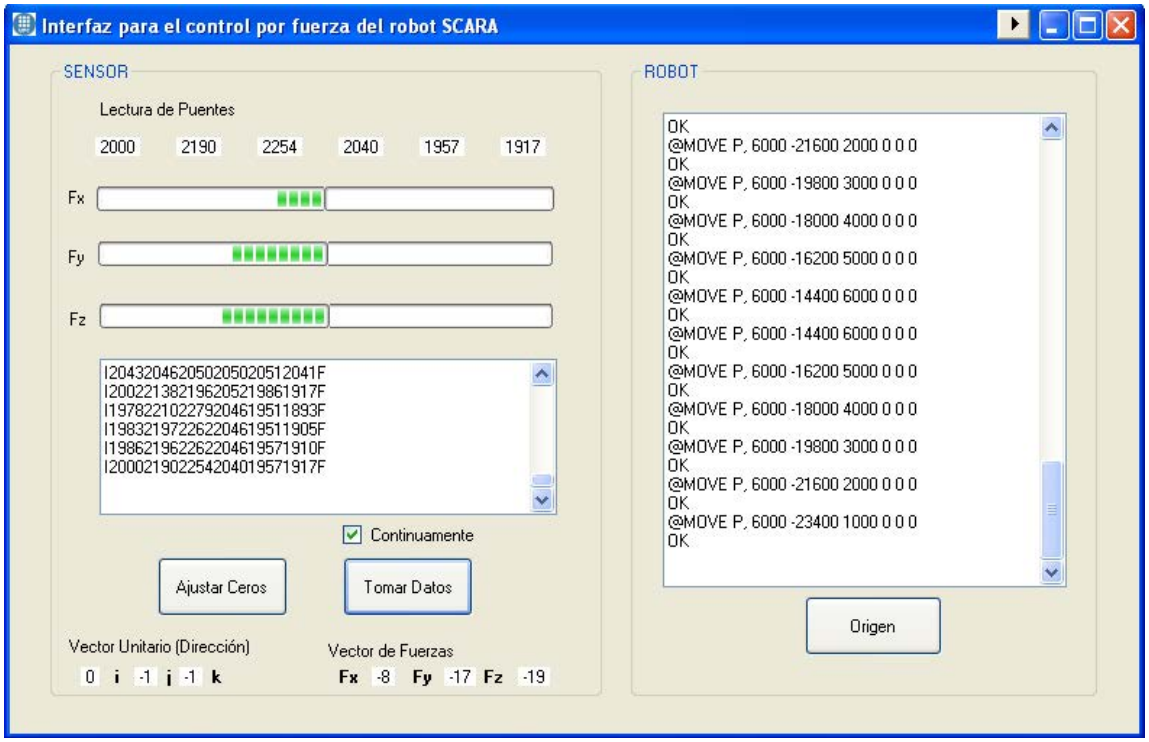

Figura 4.4: Interfaz del programa de control por fuerza del robot SCARA.

El control del robot se ejecuta enviándole comandos de movimiento en las tres dimensiones espaciales para el efector final a la controladora del robot. Para ello, se parte del conocimiento del punto inicial en el que comienza el robot a operar, se calcula un vector incremento $\vec{I}_p(i,j,k)$ que luego es sumado al vector posición inicial $\vec{P}_o(i,j,k)$ dando como resultado el vector posición final $\vec{P}_f(i,j,k)$ tal y como se indica en la Figura 4.5.

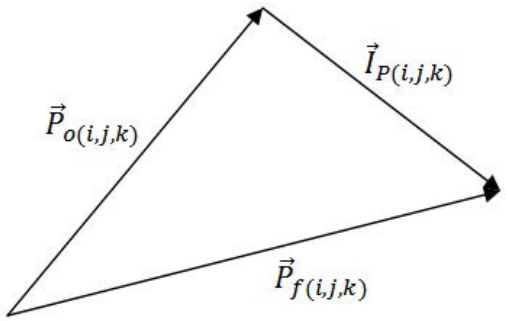

Figura 4.5: Cálculo de comando de posición para el robot.

El programa de la interfaz de control se explica en la Figura 4.6, es básicamente

un bucle infinito donde se lleva el proceso de lectura de datos y cálculos para el control del robot industrial, explicado anteriormente.

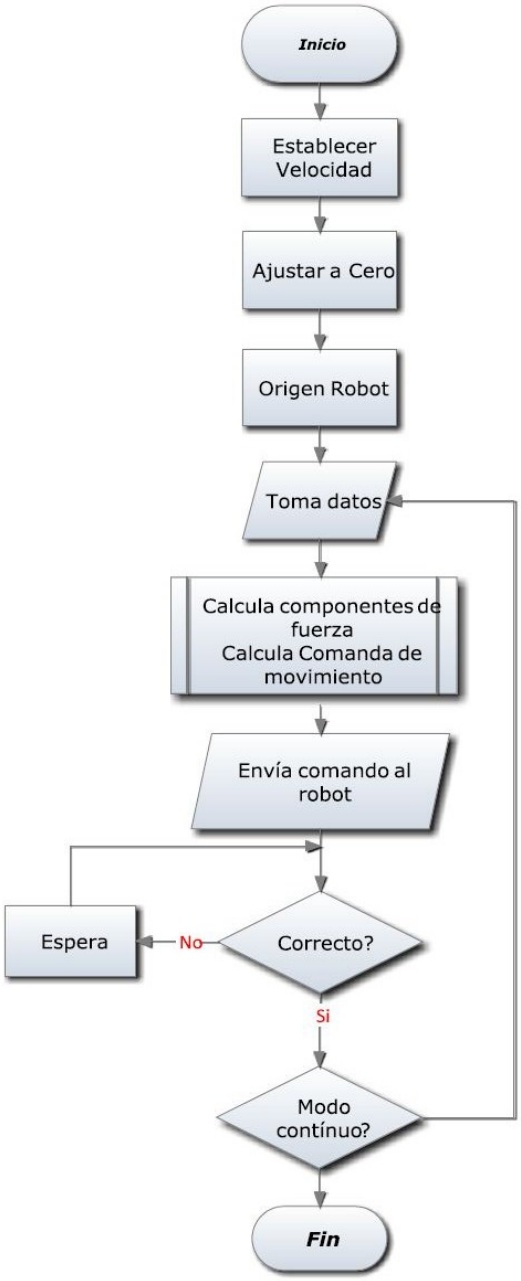

Figura 4.6: Diagrama de bloques de funcionamiento del programa de la Interfaz.

La aplicación de verificación del sensor de fuerzas y pares ha resultado satisfactoria en términos de velocidad de respuesta del sensor y respuesta al control del robot. Usando las herramientas de simulación de MPLAB y el contador de tiempo Stopwatch se ha medido el tiempo que tarda la tarjeta de acondicionamiento y adquisición de las señales en realizar las mediciones y enviar los datos de todos los puentes de galgas extensiométricas, este tiempo es de aproximadamente 1,290 ms como se muestra en la Figura 4.1.

Tabla 4.1: Tiempo de ejecución de tareas.

Función	Tiempo
Escribir SPI	$1,7\mu s$
Cambiar puente	$7,0\mu s$
Pausa estabilización	$60,0\mu s$
Medir puente	$146,3\mu s$
Total por puente	$215,0\mu s$
Total	$1290,0\mu s$

En otras palabras, el hardware desarrollado es capaz de muestrear todos los puentes a una frecuencia de 775,2 Hz. En aplicaciones de biomedicina, de interacción con pacientes, por ejemplo para ayudar a la rehabilitación o fisioterapia la frecuencia de los movimientos puede oscilar entre los 10 Hz y los 100 Hz, así que el sensor con ésta frecuencia de muestreo es más que admisible.

Capítulo 5

Conclusiones y trabajos futuros

Se realizó el diseño e implementación de un sistema sensor de fuerzas y pares de seis grados de libertad. El sistema diseñado cumple con las características deseadas desde un principio, como son:

- Tener las dimensiones apropiadas para ser montado en la muñeca del robot manipulador.
- Mantener bajos los costes de implementación
- Poseer conexión mediante USB para transmitir los datos a la interfaz del usuario.
- Tener elevada sensibilidad en las mediciones de fuerza y par.
- Tener elevada velocidad para posibilitar su implementación en sistemas de tiempo real, lo que se ha conseguido utilizando un procesador de señales.

Se ha validado su funcionamiento implementando una aplicación de control por fuerza aplicada a un robot manipulador industrial.

Se pudo comprobar que el acoplamiento entre las componentes de fuerzas es bajo para este tipo de estructura elástica, utilizando tan sólo 14 galgas extensiométricas, una cantidad inferior a la generalmente utilizada de 18 o más.

Para la realización del proyecto se presentaron las siguientes dificultades:

- Pequeños errores en las mediciones causados por desviaciones en el posicionamiento de las galgas extensiométricas sobre las vigas del sensor.
- En el montaje de la placa de acondicionamiento y adquisición de señales, no se tuvo en cuenta el tamaño que ocuparían los componentes electrónicos, así, que para encajar la placa en la estructura elástica, se han de colocar tornillos largos, otra solución es rediseñar el circuito impreso para darle el tamaño adecuado.
- En el diseño de la placa de acondicionamiento de señales, en un principio no se tuvo en cuenta el ruido, debido a esto se obtenían variaciones muy significativas en las señales de las galgas después de capturarlas con el ADC, para solucionarlo se diseñaron filtros RC paso bajo, a la frecuencia de 8KHz, frecuencia de la armónica principal vista en el analizador de señales.

Como trabajos futuros se proponen los siguientes:

- Realizar una mejora en cuanto al posicionamiento de las galgas en las vigas, ya que algunas presentan pequeñas desviaciones que influyen en las mediciones realizadas.
- Mejorar el posicionamiento de los componentes dentro del circuito impreso de la tarjeta de acondicionamiento y adquisición de señales para poder encajarla dentro de la estructura del sensor de fuerzas y pares.
- Continuar verificando su funcionamiento en otras aplicaciones con robot industriales.
- Comprobar el correcto funcionamiento utilizando redes neuronales para el procedimiento de calibración, una primera aproximación puede se usar redes del tipo SOM y a partir de las muestras tomadas hacer una clasificación para deducir los componentes de las fuerzas y pares.

Capítulo 6

Bibliografía

[1] Beer, F., Johnston, J., DeWolf, J. & Mazurek, D. 2008, *Mechanics of Materials, Third Edition edn, McGraw-Hill*.

[2] Beyeler, F., Muntwyler, S., Nagy, Z., Moser, M. & Nelson, B.J. 2007, "A multi-axis MEMS force-torque sensor for measuring the load on a microrobot actuated by magnetic fields", *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* San Diego, CA, USA, pp. 3803-3808.

[3] Beyeler, F., Muntwyler, S. & Nelson, B.J. 2009, "Design and calibration of a microfabricated 6-axis force-torque sensor for microrobotic applications", *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on,* pp. 520.

[4] Boyes, W. 2003, *Instrumentation Reference Book*, 3rd Edition edn, Elsevier.

[5] Canadija, M. & Avramovic, N. 2008, "Optimization of 6-Axis Force Transducer", *Advanced Engineering*, vol. 2, pp. 177-186.

[6] Canbay, E., Ersoy, U. & Tankut, T. 2004, "A three component force transducer for reinforced concrete structural testing", *Engineering Structures*, vol. 26, no. 2, pp. 257.

[7] Chao, L. & Chen, K. 1997, "Shape optimal design and force sensitivity evaluation of six-axis force sensors", *Sensors and Actuators A: Physical*, vol. 63, no. 2, pp. 105-112.

[8] De Silva, C.W. 2004, *Mechatronics: An Integrated Approach*, CRC Press, New York.

[9] Dorogoy, A. & Rittel, D. 2008, "Optimum location of a three strain gauge rosette for measuring mixed mode stress intensity factors", *Engineering Fracture Mechanics*, vol. 75, no. 14, pp. 4127.

[10] Elbestawi, M. 1999, "Force Measurementïn *The Measurement, Instrumentation and Sensors Handbook on CD-ROM* CRC Press.

[11] Fernandez, A.F., Berghmans, F., Brichard, B., Gusarov, A., Deparis, O., Decréton, M., Mégret, P., Blondel, M. & Delchambre, A. 2000, "Multi-component force sensors based on multiplexed fiber Bragg grating strain sensors", *Proc. SPIE 14th Int. Conf. Optical Fiber Sensors (OFS-14)*, pp. 106.

[12] Gaillet, A. & Reboulet, C. 1983, "An isostatic six component force and torque sensor", *Proceedings of the 13th international symposium on industrial robotics*, pp. 102-111.

[13] Garshelis, I. 1999, "Torque and Power Measurementïn *The Measurement,*

Instrumentation and Sensors Handbook on CD-ROM CRC Press.

[14] Gorinevsky, D., Formalsky, A. & Schneider, A. 1997, *Force Control of Robotics Systems*, CRC-Press.

[15] Goto, T., Inoyama, T. & Takeyasu, K. 1974, "Precise insert operation by tactile controlled robot HI-T_HAND EXPERT 2", *4th International Symposium on Industrial Robots*, pp. 209-217.

[16] Hirose, S. & Yoneda, K. 1990, "Development of optical six-axial force sensor and its signal calibration considering nonlinear interference", Robotics and Automation, 1990. *Proceedings., 1990 IEEE International Conference on*, pp. 46.

[17] Hu, X., Wang, R., Wu, F., Jin, D., Jia, X., Zhang, J., Cai, F. & Zheng, S. 2007, "Finite element analysis of a six-component force sensor for the transfemoral prosthesis", *Proceedings of the 1st international conference on Digital human modeling Springer-Verlag*, Berlin, Heidelberg, pp. 633.

[18] Jacq, C., Lüthi, B., Maeder, T., Lambercy, O. & Gassert, R. 2010, "Thick-film multi-DOF force / torque sensor for wrist rehabilitation", *Sensors and Actuators*, vol. 162, no. 2, pp. 361-366.

[19] Jin, W.L., Mote, C.D. & Jr. 1998, "Development and calibration of a submillimeter three-component force sensor", *Sensors and Actuators A: Physical*, vol. 65, no. 1, pp. 89.

[20] Kang, C. 2005, "Performance Improvement of a 6-Axis Force-torque Sensor via Novel Electronics and Cross-shaped Double-hole Structure", *International Journal of Control Automation and System*, vol. 3, no. 3, pp. 469-476.

[21] Kang, C. 2001, "Closed-form force sensing of a 6-axis force transducer based on the Stewart platform", *Sensors and Actuators A: Physical,* vol. 90, no. 1-2, pp. 31-37.

[22] Kim, G. 2007, "Design of a six-axis wrist force/moment sensor using FEM and its fabrication for an intelligent robot", *Sensors and Actuators A: Physical*, vol. 133, no. 1, pp. 27.

[23] Kim, G., Kang, D. & Rhee, S. 1999, "Design and fabrication of a six-component force/moment sensor", *Sensors and Actuators A: Physical*, vol. 77, no. 3, pp. 209-220.

[24] Kim, G., Shin, H. & Yoon, J. 2008, "Development of 6-axis force/moment sensor for a humanoid robot's intelligent foot", *Sensors and Actuators A: Physical*, vol. 141, no. 2, pp. 276.

[25] Krouglicof, N., Alonso, L.M. & Keat, W.D. 2004, "Development of a mechanically coupled, six degree-of-freedom load platform for biomechanics and sports medicine", *2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 4426.

[26] Li, Y.F. & Chen, X.B. 1998, "On the dynamic behavior of a force/torque sensor for robots", *Instrumentation and Measurement, IEEE Transactions on*, vol. 47, no. 1, pp. 304-308.

[27] Li, Y., Sun, B., Zhang, J., Jia, Z. & Qian, M. 2008, "Research on Piezoelectric Six-Dimensional Heavy Force Sensor with Four-Point Supporting Structure", *Proceedings of the First International Conference on Intelligent Robotics and Applications: Part I* Springer-Verlag, Berlin, Heidelberg, pp. 314.

[28] Liang, Q., Ge, Y., Song, Q. & Ge, Y. 2009, "A novel thin six-dimensional

wrist force/torque sensor with isotropy", *ICIA '09. International Conference on Information and Automation*, 2009, pp. 1135.

[29] Liang, Q., Zhang, D., Song, Q., Ge, Y., Cao, H. & Ge, Y. 2010, "Design and fabrication of a six-dimensional wrist force/torque sensor based on E-type membranes compared to cross beams", *Measurement*, vol. 43, no. 10, pp. 1702-1719.

[30] Lin, S., Liu, W., Wang, Y., Jia, Z. & Wang, F. 2008, "A Novel Six-Axis Heavy Force Sensor", *International Conference on Image Analysis and Recognition 2008*, Springer Berlin / Heidelberg.

[31] Liu, S.A. & Tzo, H.L. 2002, "A novel six-component force sensor of good measurement isotropy and sensitivities", *Sensors and Actuators A: Physical*, vol. 100, no. 2-3, pp. 223-230.

[32] Lu, T., Lin, G.C.I. & He, J.R. 1997, "Neural-network-based 3D force/torque sensor calibration for robot applications", *Engineering Applications of Artificial Intelligence*, vol. 10, no. 1, pp. 87.

[33] Luo, R.C. 1988, "A microcomputer-based intelligent sensor for multiaxis force/torque measurement", *IEEE Transactions on Industrial Electronics*, vol. 35, no. 1, pp. 26-30.

[34] Park, F.C. & Martin, B.J. 1994, "Robot sensor calibration: solving AX=XB on the Euclidean group", Robotics and Automation, *IEEE Transactions on*, vol. 10, no. 5, pp. 717-721.

[35] Sanders, J.E., Miller, R.A., Berglund, D.N. & Zachariah, S.G. 1997, "A modular six-directional force sensor for prosthetic assessment: a technical note", *Journal of Rehabilitation Research and Development*, vol. 34, no. 2, pp. 195-202.

[36] Sciavicco, L. & Siciliano, B. 1995, *Modelling and control of robot manupulators*, McGraw-Hill, New York.

[37] Shi, W. & D Hall, S. 2005, "A novel six-axis force sensor for measuring the loading of the racing tyre on track", *1st International Conference on Sensing Technology*, pp. 408-413.

[38] Shimano, B. & Roth, B. 1977, "On force sensing information and its use in controlling manipulators", *Proceedings of the IFAC International Symposium on Information-Control Problems in Manufacturing Technology.*

[39] Song, A. 2008, "Multi-Dimensional Force Sensor Design for Haptic Human-Computer Interactionïn *Human Computer Interaction*, ed. I. Pavlidis, InTech, pp. 239-256.

[40] Song, A., Wu, J., Li, J., Zeng, Q. & Huang, W. 2006, "A Novel Four Degree of Freedom Wrist Force/Torque Sensor with Low Coupled Interference", *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4423.

[41] Stefanescu, D.M. 2011, *Handbook of Force Transducers*, 1st Edition edn, Springer-Verlag, Berlin.

[42] Sun, Y.; Kim, K.; Voyles, R.; Nelson, B. 2007, "Calibration of Multi-Axis MEMS Force Sensors Using the Shape-From-Motion Method", *Sensors Journal*, IEEE, vol. 7, no. 3, pp. 344-351.

[43] Takahashi, N., Tada, M., Ueda, J., Matsumoto, Y. & Ogasawara, T. 2003, "An optical 6-axis force sensor for brain function analysis using fMRI", *In Proceedings of IEEE Sensors*, pp. 253.

[44] Tsuji, T. & Hanyu, R. 2010, "Fault tolerance measurement using a six-

axis force/torque sensing system with redundancy", *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1890.

[45] Voyles, R., Morrow, J. & Khosla, P. 1997, "The Shape from Motion Approach to Rapid and Precise Force/Torque Sensor Calibration", *Journal of Dynamic Systems, Measurement, and Control*, vol. 119, pp. 229–235.

[46] Weiyi, H., Hongming, J. & Hanqing, Z. 1993, "Mechanical analysis of a novel six-degree-of-freedom wrist force sensor", *Sensors and Actuators A: Physical*, vol. 35, no. 3, pp. 203-208.

[47] Xu, K., Mei, T., Li, Q. & Wu, T. 2004, "Measurement of wrist force/torque using data fusion technique", *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, pp. 3027.

[48] Zhang, X. 2009, "Development of a novel three-axis force sensor", *Technology and Innovation Conference 2009 (ITIC 2009), International*, pp. 1.

Capítulo 7

Anexos

7.1. Anexo I. Código fuente programa principal

```
/*****************************************************************/
/*  Proyecto:    Sensor de fuerzaa de 6 grados de libertat       */
/*  Module:      Lectura de 6 galgas, control del DAC por SPI    */
/*                                                               */
/*  Autor(es):   Jean Carlo, Oriol Escoté, Aina Barceló          */
/*                                                               */
/*  Revision:    17/08/2011                                      */
/*****************************************************************/
#include "p33FJ12GP202.h"
#include "p33fxxxx.h"
#include "delay.h"
#include <uart.h>
#include <timer.h>
#include <adc.h>
#include <stdio.h>
#include <spi.h>
#include <libpic30.h>
/*****************************************************************/
/* Configuration words                                           */
/*****************************************************************/
_FOSC(POSCMD_EC & FNOSC_PRIPLL); //CLK switching and fail safe monitor
    disable and External clock with PLL16
_FWDT(FWDTEN_OFF); //Watch dog disable
_FPOR(FPWRT_PWR1); //Voltage protection disable
_FGS( GCP_OFF );//code protection disable
/*****************************************************************/
/* Hardware                                                      */
/*****************************************************************/
#define BAUD_RATE     115200  //usart frequency
#define FXT         6000000LL //oscilation frequency used
#define PLL         16 //pll used
#define FCY         (FXT * PLL) / 4  //24 Mhz
#define BRG         (FCY / (16L * BAUD_RATE)) - 1L //baud rate generator
    register
#define ANALOG_SUPPLY       LATBbits.LATB1 //entrada análoga
#define mux0            PORTDbits.RD0
#define mux1            PORTCbits.RC15
```

```
36 #define mux2          PORTBbits.RB7
37 /*********************************************************************/
38 /* Global Variable declaration                                       */
39 /*********************************************************************/
40 char uiCharacter;
41 unsigned int valor_DAC[8],valor[8],valor_puente[8];
42 void medir_puentes (void);
43 void ajustar_ceros (void);
44 void rutina_calibracion(void);
45 void cambiar_puentes();
46 void Delay_Us();
47 /*********************************************************************/
48 /* Interrupts                                                        */
49 /*********************************************************************/
50 /*********************************************************************/
51 /* Configuración de Periféricos                                      */
52 /*********************************************************************/
53 void ConfigIO()
54 {    // Config input & output ports
55   LATA  = 0x0000;
56   TRISA = 0x3FFF;     // PortC as input
57   LATB  = 0x0000;
58   TRISB = 0x002B;     // PortD as input
59   ADPCFG = 0xFFFF;
60 }
61 /*********************************************************************/
62 void AnalogConfig()
63 {
64   AD1CON1bits.ADON = 0;    // Turn off ADC
65   OpenADC1(ADC_MODULE_ON &// Turn on ADC
66   ADC_IDLE_CONTINUE &      // Continue in idle mode
67   ADC_AD12B_12BIT &        // 12 bit, 1-channel ADC operation
68   ADC_FORMAT_INTG &        // Integer output format
69   ADC_CLK_AUTO &           // Internal counter ends sampling and starts
70   ADC_AUTO_SAMPLING_OFF & // Sampling begins when SAMP bit set
71   ADC_SAMP_OFF,            // Writing 1 will start sampling
72   ADC_VREF_EXT_AVSS  &     // Volt. ref. Vref+ external, Vref- is AVss
73   ADC_SAMPLES_PER_INT_1 & // Convert 1 samples per interrupt
74   ADC_SCAN_OFF &           // Do not scan inputs
75   ADC_ALT_BUF_OFF &        // Buffer mode 16 bits
76   ADC_ALT_INPUT_OFF,       // Do not use alternate input mode
77   ADC_SAMPLE_TIME_31 &     // Auto-sample time bits ->Tsample = 31 Tad
78   ADC_CONV_CLK_SYSTEM &    // A/D Source Clock derived from system clock
79   ADC_CONV_CLK_32Tcy,      // A/D Conversion Clock -> Tad = 32 Tcy
80   ADC_DMA_BUF_LOC_128,     // Allocates words of buffer to each analog
       input
81   ADC_CH123_POS_SAMPLEA_0_1_2, // A/D CH1 pos i/p is AN0, CH2 pos i/p
       is AN1, CH3 pos i/p is AN2
82   ADC_CH0_NEG_SAMPLEB_AN1,    // CH0 negative input is AN1
83   ENABLE_AN1_ANA,          // AN1 as analog
84   ADC_SCAN_OFF);           // Do not perform input scan
85 }
86 /**************** * LECTURA DEL ADC * ****************************/
87 unsigned int AcquireAnalog()
88 {
89   unsigned int uiTemp=0, j=0;
```

```
90    AD1PCFGLbits.PCFG1 = 0;
91    SetChanADC1(ADC_CH0_NEG_SAMPLEA_AN1 &//A/D sel for SAMPLE A is AN1
92     ADC_CH0_NEG_SAMPLEA_VREFN); // Set chan. 0 neg. input to Vref-
93    for (j=0;j<10;j++){
94      IFS0bits.ADIF = 0;            // Reset ADC interrupt flag
95      ADCON1bits.SAMP=1;            // Start sampling
96      while(!IFS0bits.ADIF);        // Wait until conversion is finished
97      uiTemp = (uiTemp + ((ReadADC12(0))/10));// Promedio lecturas ADC
98    }
99    uiTemp = ReadADC1(0);
100   return uiTemp;
101 }
102 /****************** * CONFIGURACIÓN DE LA UART * ********************/
103 void UARTConfig()
104 {
105 ConfigIntUART1(UART_RX_INT_DIS &
106        UART_RX_INT_PR6 &
107        UART_TX_INT_DIS &
108        UART_TX_INT_PR4);
109        OpenUART1(UART_EN &     // Enable UART
110        UART_IDLE_CON &         // Continue on idle mode
111        UART_DIS_WAKE &         // Disable wake-on start
112        UART_DIS_LOOPBACK &     // Disable loopback mode
113        UART_DIS_ABAUD &        // Disable autobaud mode
114        UART_NO_PAR_8BIT &      // 8bits / No parity
115        UART_1STOPBIT,          // 1 Stop bit
116        UART_TX_PIN_NORMAL &    // Tx break bit normal
117        UART_TX_ENABLE     &    // Enable Transmission
118        UART_INT_RX_CHAR,       // Interrupt on every char received
119        BRG);                   // Baudrate
120 }
121 /********************** * ENVIAR DATO AL DAC * **********************/
122 void escribir_SPI(short command)
123 {
124   short temp;
125   PORTBbits.RB2 = 1;  // lower the slave select line
126   temp = SPI1BUF;     // read SPI1BUF register to clear SPIRBF flag
127   SPI1BUF = command;  // write the data out to the SPI peripheral
128 //while (!SPI1STATbits.SPIRBF);  // wait for the data to be sent out
129   PORTBbits.RB2 = 0;  // raise the slave select line
130 }
131 /****************** * RUTINA CAMBIO DE PUENTES **********************/
132 void cambiar_puentes(int j){
133   switch (j){
134     case 0:
135       mux0=0x0;
136       mux1=0x0;
137       mux2=0x0;
138       Delay_Us(250); //propagación de las señales del multiplexor
139       break;
140     case 1:
141       mux0=0x1;
142       mux1=0x0;
143       mux2=0x0;
144       Delay_Us(250);
145       break;
```

```
146     case 2:
147       mux0=0x0;
148       mux1=0x1;
149       mux2=0x0;
150       Delay_Us(250);
151       break;
152     case 3:
153       mux0=0x1;
154       mux1=0x1;
155       mux2=0x0;
156       Delay_Us(250);
157       break;
158     case 4:
159       mux0=0x0;
160       mux1=0x0;
161       mux2=0x1;
162       Delay_Us(250);
163       break;
164     case 5:
165       mux0=0x1;
166       mux1=0x0;
167       mux2=0x1;
168       Delay_Us(250);
169       break;
170     default:
171       break;
172   }
173 return;
174 }
175 /******************* * RUTINA PRINCIPAL * ************************/
176 int main(void)
177 {
178   char ucText[80];
179   ConfigIO();    // Configurar puertos de entrada y salida
180   UARTConfig();                 // Configurar UART para la comunicación
181   Init_SPI();                   // Configurar interfaz SPI para
        comunicación con el DAC
182   AnalogConfig();                  // Initialize the A/D converter
183   cambiar_puentes(0);
184   putsUART1((unsigned int*)"\r\nSensor de Fuerza de 6 GDL\r\n");
185   while(BusyUART1());           // Wait until the character is
        transmitted
186   putsUART1((unsigned int*)"-----------------------------------\r\n");
187   while(BusyUART1());           // Wait until the character is
        transmitted
188   escribir_SPI(4095);       // Enviar valor más alto a DAC para evitar
        daño del puerto del ADC
189   Delay_Us(100);          // Esperar para propagación de la señal
190   while(1){                          // Infinite loop
191     while(!DataRdyUART1());      // Wait until a character is received
192     uiCharacter = ReadUART1();  // Read the received character
193     switch (uiCharacter){
194       case 'a':   //Ajustar ceros
195         putsUART1((unsigned int*)"\r\nAjustando ceros, por favor espera
              ...\r\n");
```

```
196         while(BusyUART1());          // Wait until the character is
                transmitted
197         ajustar_ceros();
198         uiCharacter='n';
199         break;
200       case 'c':   //Medición contínua de los puentes
201         //while(1){
202         medir_puentes();
203         sprintf (ucText,"I %04d %04d %04d %04d %04d %04dF\r\n",valor_puente
                [0],valor_puente[1],valor_puente[2],valor_puente[3],
                valor_puente[4],valor_puente[5]);
204         putsUART1((unsigned int*)ucText);
205         while(BusyUART1());
206           uiCharacter='n';
207           break;
208       case 's':   //Empieza Rutina de Calibración (Recolección de Datos
              )
209         rutina_calibracion();
210         uiCharacter='n';
211         break;
212     case '0':   //Medir puente 0
213     escribir_SPI(valor_DAC[0]);
214     cambiar_puentes(0);
215     Delay_Us(100); //60ns de propagación de las señales del multiplexor
216     valor[0]= AcquireAnalog();
217     sprintf(ucText,"\r\nValor Puente 0: %04d\r\n",valor[0]);
218     putsUART1((unsigned int*)ucText);
219     while(BusyUART1());
220     uiCharacter='n';
221     break;
222     case '1':   //Medir puente 1
223     escribir_SPI(valor_DAC[1]);
224        cambiar_puentes(1);
225        Delay_Us(100); //60ns de propagación de las señales del
              multiplexor
226        valor[1]=AcquireAnalog();
227        sprintf(ucText,"\r\nValor Puente 1: %04d\r\n",valor[1]);
228        putsUART1((unsigned int*)ucText);
229        while(BusyUART1());
230        uiCharacter='n';
231        break;
232     case '2':   //Medir puente 2
233        escribir_SPI(valor_DAC[2]);
234        cambiar_puentes(2);
235        Delay_Us(100); //60ns de propagación de las señales del
              multiplexor
236        valor[2]=AcquireAnalog();
237        sprintf(ucText,"\r\nValor Puente 2: %04d\r\n",valor[2]);
238        putsUART1((unsigned int*)ucText);
239        while(BusyUART1());
240        uiCharacter='n';
241        break;
242     case '3':   //Medir puente 3
243        escribir_SPI(valor_DAC[3]);
244        cambiar_puentes(3);
```

```
245         Delay_Us(100); //60ns de propagación de las señales del
                multiplexor
246         valor[3]=AcquireAnalog();
247         sprintf(ucText,"\r\nValor Puente 3: %04d\r\n",valor[3]);
248         putsUART1((unsigned int*)ucText);
249         while(BusyUART1());
250         uiCharacter='n';
251         break;
252      case '4':   //Medir puente 4
253         escribir_SPI(valor_DAC[4]);
254         cambiar_puentes(4);
255         Delay_Us(100); //60ns de propagación de las señales del
                multiplexor
256         valor[4]=AcquireAnalog();
257         sprintf(ucText,"\r\nValor Puente 4: %04d\r\n",valor[4]);
258         putsUART1((unsigned int*)ucText);
259         while(BusyUART1());
260         uiCharacter='n';
261         break;
262      case '5':   //Medir puente 5
263         escribir_SPI(valor_DAC[5]);
264         cambiar_puentes(5);
265         Delay_Us(100); //60ns de propagación de las señales del
                multiplexor
266         valor[5]=AcquireAnalog();
267         sprintf(ucText,"\r\nValor Puente 5: %04d\r\n",valor[5]);
268         putsUART1((unsigned int*)ucText);
269         while(BusyUART1());
270         uiCharacter='n';
271         break;
272      default:
273           uiCharacter='n';
274           break;
275         }
276       }
277 return 0;
278 }//fin main
279
280 /************************ * SUBRUTINES * ***************************/
281 /************** * LECTURA CONTINUADA DEL SENSOR * ******************/
282 void medir_puentes (void)
283 {
284   short int i;
285   for (i=0;i<6;i++){
286     escribir_SPI(valor_DAC[i]);
287     cambiar_puentes(i);
288     valor_puente[i]=AcquireAnalog();
289   }
290 }
291 /************************ * AJUSTAR CEROS * ************************/
292 void ajustar_ceros (void){
293 short int k;
294 char ucLectura[80];
295         sprintf(ucLectura,"\r\n                    Puente     Valor_DAC
                Valor_puente\r\n");
296         putsUART1((unsigned int*)ucLectura);
```

```
297         while(BusyUART1());
298     for (k=0;k<6;k++){
299         cambiar_puentes(k);
300         sprintf(ucLectura,"Ajuste del puente: %01d  -->     ", k);
301         putsUART1((unsigned int*)ucLectura);
302         while(BusyUART1());
303 
304         valor_DAC[k]=4095;
305 again:    escribir_SPI(valor_DAC[k]);
306         Delay_Us(130);
307         valor_puente[k]=AcquireAnalog();
308 
309         if (valor_puente[k]<=2048){
310             valor_DAC[k]=(valor_DAC[k])+1;
311             goto again;
312         }else if (valor_puente[k]>=2050){
313             valor_DAC[k]=(valor_DAC[k])-1;
314             goto again;
315         }
316 
317         sprintf(ucLectura,"%04d       --> %04d\r\n", valor_DAC[k],
                valor_puente[k]);
318         putsUART1((unsigned int*)ucLectura);
319         while(BusyUART1());
320 
321     }
322 
323 return;
324 }
325 /*************** * RUTINA PARA REALIZAR LA CALIBRACIÓN * ***********/
326 void rutina_calibracion(){
327 char ucLectura[80];
328 int samples, i;
329 samples=36;
330 
331 //Empieza a tomar datos con el eje X
332 sprintf(ucLectura,"Estos datos son respecto al eje X, oprima una tecla
        para continuar\r\n");
333 putsUART1((unsigned int*)ucLectura);
334 while(BusyUART1());
335 while(!DataRdyUART1());      // Wait until a character is received
336 uiCharacter = ReadUART1();  // Read the received character
337 
338 for (i=0;i<samples;i++){
339     medir_puentes();
340     sprintf(ucLectura,"0 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
341     putsUART1((unsigned int*)ucLectura);
342     while(BusyUART1());
343 }
344 
345 sprintf(ucLectura,"Estos datos son respecto al eje +X, oprima una tecla
        para continuar\r\n");
346 putsUART1((unsigned int*)ucLectura);
347 while(BusyUART1());
```

```
348 while(!DataRdyUART1());      // Wait until a character is received
349 uiCharacter = ReadUART1();  // Read the received character
350
351 for (i=0;i<samples;i++){
352     medir_puentes();
353     sprintf(ucLectura,"2 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
354     putsUART1((unsigned int*)ucLectura);
355     while(BusyUART1());
356 }
357
358 sprintf(ucLectura,"Estos datos son respecto al eje +X, oprima una tecla
        para continuar\r\n");
359 putsUART1((unsigned int*)ucLectura);
360 while(BusyUART1());
361 while(!DataRdyUART1());      // Wait until a character is received
362 uiCharacter = ReadUART1();  // Read the received character
363
364 for (i=0;i<samples;i++){
365     medir_puentes();
366     sprintf(ucLectura,"5 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
367     putsUART1((unsigned int*)ucLectura);
368     while(BusyUART1());
369 }
370
371 sprintf(ucLectura,"Estos datos son respecto al eje +X, oprima una tecla
        para continuar\r\n");
372 putsUART1((unsigned int*)ucLectura);
373 while(BusyUART1());
374 while(!DataRdyUART1());      // Wait until a character is received
375 uiCharacter = ReadUART1();  // Read the received character
376
377 for (i=0;i<samples;i++){
378     medir_puentes();
379     sprintf(ucLectura,"10 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
380     putsUART1((unsigned int*)ucLectura);
381     while(BusyUART1());
382 }
383
384 sprintf(ucLectura,"Estos datos son respecto al eje +Y, oprima una tecla
        para continuar\r\n");
385 putsUART1((unsigned int*)ucLectura);
386 while(BusyUART1());
387 while(!DataRdyUART1());      // Wait until a character is received
388 uiCharacter = ReadUART1();  // Read the received character
389
390 for (i=0;i<samples;i++){
391     medir_puentes();
392     sprintf(ucLectura,"0 2 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
```

```
393     putsUART1((unsigned int*)ucLectura);
394     while(BusyUART1());
395 }
396
397 sprintf(ucLectura,"Estos datos son respecto al eje +Y, oprima una tecla
        para continuar\r\n");
398 putsUART1((unsigned int*)ucLectura);
399 while(BusyUART1());
400 while(!DataRdyUART1());      // Wait until a character is received
401 uiCharacter = ReadUART1();   // Read the received character
402
403 for (i=0;i<samples;i++){
404     medir_puentes();
405     sprintf(ucLectura,"0 5 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
          valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
          [3], valor_puente[4], valor_puente[5]);
406     putsUART1((unsigned int*)ucLectura);
407     while(BusyUART1());
408 }
409
410 sprintf(ucLectura,"Estos datos son respecto al eje +Y, oprima una tecla
        para continuar\r\n");
411 putsUART1((unsigned int*)ucLectura);
412 while(BusyUART1());
413 while(!DataRdyUART1());      // Wait until a character is received
414 uiCharacter = ReadUART1();   // Read the received character
415
416 for (i=0;i<samples;i++){
417     medir_puentes();
418     sprintf(ucLectura,"0 10 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
          valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
          [3], valor_puente[4], valor_puente[5]);
419     putsUART1((unsigned int*)ucLectura);
420     while(BusyUART1());
421 }
422
423 sprintf(ucLectura,"Estos datos son respecto al eje +Z, oprima una tecla
        para continuar\r\n");
424 putsUART1((unsigned int*)ucLectura);
425 while(BusyUART1());
426 while(!DataRdyUART1());      // Wait until a character is received
427 uiCharacter = ReadUART1();   // Read the received character
428
429 for (i=0;i<samples;i++){
430     medir_puentes();
431     sprintf(ucLectura,"0 0 2 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
          valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
          [3], valor_puente[4], valor_puente[5]);
432     putsUART1((unsigned int*)ucLectura);
433     while(BusyUART1());
434 }
435
436 sprintf(ucLectura,"Estos datos son respecto al eje +Z, oprima una tecla
        para continuar\r\n");
437 putsUART1((unsigned int*)ucLectura);
438 while(BusyUART1());
```

```
439 while(!DataRdyUART1());      // Wait until a character is received
440 uiCharacter = ReadUART1();   // Read the received character
441
442 for (i=0;i<samples;i++){
443     medir_puentes();
444     sprintf(ucLectura,"0 0 5 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
445     putsUART1((unsigned int*)ucLectura);
446     while(BusyUART1());
447 }
448
449 sprintf(ucLectura,"Estos datos son respecto al eje +Z, oprima una tecla
        para continuar\r\n");
450 putsUART1((unsigned int*)ucLectura);
451 while(BusyUART1());
452 while(!DataRdyUART1());      // Wait until a character is received
453 uiCharacter = ReadUART1();   // Read the received character
454
455 for (i=0;i<samples;i++){
456     medir_puentes();
457     sprintf(ucLectura,"0 0 10 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
458     putsUART1((unsigned int*)ucLectura);
459     while(BusyUART1());
460 }
461
462 //**********************    VALORES NEGATIVOS **********************/
463
464 sprintf(ucLectura,"Estos datos son respecto al eje -X, oprima una tecla
        para continuar\r\n");
465 putsUART1((unsigned int*)ucLectura);
466 while(BusyUART1());
467 /while(!DataRdyUART1());      // Wait until a character is received
468 uiCharacter = ReadUART1();   // Read the received character
469
470 for (i=0;i<samples;i++){
471     medir_puentes();
472     sprintf(ucLectura,"-2 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
            [3], valor_puente[4], valor_puente[5]);
473     putsUART1((unsigned int*)ucLectura);
474     while(BusyUART1());
475 }
476
477 sprintf(ucLectura,"Estos datos son respecto al eje -X, oprima una tecla
        para continuar\r\n");
478 putsUART1((unsigned int*)ucLectura);
479 while(BusyUART1());
480 while(!DataRdyUART1());      // Wait until a character is received
481 uiCharacter = ReadUART1();   // Read the received character
482
483 for (i=0;i<samples;i++){
484     medir_puentes();
```

```
485     sprintf(ucLectura,"-5 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
        [3], valor_puente[4], valor_puente[5]);
486     putsUART1((unsigned int*)ucLectura);
487     while(BusyUART1());
488 }
489
490 sprintf(ucLectura,"Estos datos son respecto al eje -X, oprima una tecla
      para continuar\r\n");
491 putsUART1((unsigned int*)ucLectura);
492 while(BusyUART1());
493 while(!DataRdyUART1());      // Wait until a character is received
494 uiCharacter = ReadUART1();  // Read the received character
495
496 for (i=0;i<samples;i++){
497     medir_puentes();
498     sprintf(ucLectura,"-10 0 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
        valor_puente[3], valor_puente[4], valor_puente[5]);
499     putsUART1((unsigned int*)ucLectura);
500     while(BusyUART1());
501 }
502
503 sprintf(ucLectura,"Estos datos son respecto al eje -Y, oprima una tecla
      para continuar\r\n");
504 putsUART1((unsigned int*)ucLectura);
505 while(BusyUART1());
506 while(!DataRdyUART1());      // Wait until a character is received
507 uiCharacter = ReadUART1();  // Read the received character
508
509 for (i=0;i<samples;i++){
510     medir_puentes();
511     sprintf(ucLectura,"0 -2 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
        [3], valor_puente[4], valor_puente[5]);
512     putsUART1((unsigned int*)ucLectura);
513     while(BusyUART1());
514 }
515
516 sprintf(ucLectura,"Estos datos son respecto al eje -Y, oprima una tecla
      para continuar\r\n");
517 putsUART1((unsigned int*)ucLectura);
518 while(BusyUART1());
519 while(!DataRdyUART1());      // Wait until a character is received
520 uiCharacter = ReadUART1();  // Read the received character
521
522 for (i=0;i<samples;i++){
523     medir_puentes();
524     sprintf(ucLectura,"0 -5 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
        [3], valor_puente[4], valor_puente[5]);
525     putsUART1((unsigned int*)ucLectura);
526     while(BusyUART1());
527 }
528
```

```
529 sprintf(ucLectura,"Estos datos son respecto al eje -Y, oprima una tecla
        para continuar\r\n");
530 putsUART1((unsigned int*)ucLectura);
531 while(BusyUART1());
532 while(!DataRdyUART1());      // Wait until a character is received
533 uiCharacter = ReadUART1();   // Read the received character
534
535 for (i=0;i<samples;i++){
536     medir_puentes();
537     sprintf(ucLectura,"0 -10 0 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
           valor_puente[3], valor_puente[4], valor_puente[5]);
538     putsUART1((unsigned int*)ucLectura);
539     while(BusyUART1());
540 }
541
542 sprintf(ucLectura,"Estos datos son respecto al eje -Z, oprima una tecla
        para continuar\r\n");
543 putsUART1((unsigned int*)ucLectura);
544 while(BusyUART1());
545 while(!DataRdyUART1());      // Wait until a character is received
546 uiCharacter = ReadUART1();   // Read the received character
547
548 for (i=0;i<samples;i++){
549     medir_puentes();
550     sprintf(ucLectura,"0 0 -2 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
           [3], valor_puente[4], valor_puente[5]);
551     putsUART1((unsigned int*)ucLectura);
552     while(BusyUART1());
553 }
554
555 sprintf(ucLectura,"Estos datos son respecto al eje -Z, oprima una tecla
        para continuar\r\n");
556 putsUART1((unsigned int*)ucLectura);
557 while(BusyUART1());
558 while(!DataRdyUART1());      // Wait until a character is received
559 uiCharacter = ReadUART1();   // Read the received character
560
561 for (i=0;i<samples;i++){
562     medir_puentes();
563     sprintf(ucLectura,"0 0 -5 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n",
            valor_puente[0], valor_puente[1], valor_puente[2], valor_puente
           [3], valor_puente[4], valor_puente[5]);
564     putsUART1((unsigned int*)ucLectura);
565     while(BusyUART1());
566 }
567
568 sprintf(ucLectura,"Estos datos son respecto al eje -Z, oprima una tecla
        para continuar\r\n");
569 putsUART1((unsigned int*)ucLectura);
570 while(BusyUART1());
571 while(!DataRdyUART1());      // Wait until a character is received
572 uiCharacter = ReadUART1();   // Read the received character
573
574 for (i=0;i<samples;i++){
```

```
575     medir_puentes();
576     sprintf(ucLectura,"0 0 -10 0 0 0 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
577     putsUART1((unsigned int*)ucLectura);
578     while(BusyUART1());
579 }
580
581 /********************** *  PARES * ******************************/
582
583 sprintf(ucLectura,"Estos datos son respecto al eje X, oprima una tecla
        para continuar\r\n");
584 putsUART1((unsigned int*)ucLectura);
585 while(BusyUART1());
586 while(!DataRdyUART1());      // Wait until a character is received
587 uiCharacter = ReadUART1();   // Read the received character
588
589 for (i=0;i<samples;i++){
590     medir_puentes();
591     sprintf(ucLectura,"0 0 0 0.26 0 0 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
592     putsUART1((unsigned int*)ucLectura);
593     while(BusyUART1());
594 }
595
596 sprintf(ucLectura,"Estos datos son respecto al eje X, oprima una tecla
        para continuar\r\n");
597 putsUART1((unsigned int*)ucLectura);
598 while(BusyUART1());
599 while(!DataRdyUART1());      // Wait until a character is received
600 uiCharacter = ReadUART1();   // Read the received character
601
602 for (i=0;i<samples;i++){
603     medir_puentes();
604     sprintf(ucLectura,"0 0 0 0.65 0 0 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
605     putsUART1((unsigned int*)ucLectura);
606     while(BusyUART1());
607 }
608
609 sprintf(ucLectura,"Estos datos son respecto al eje X, oprima una tecla
        para continuar\r\n");
610 putsUART1((unsigned int*)ucLectura);
611 while(BusyUART1());
612 while(!DataRdyUART1());      // Wait until a character is received
613 uiCharacter = ReadUART1();   // Read the received character
614
615 for (i=0;i<samples;i++){
616     medir_puentes();
617     sprintf(ucLectura,"0 0 0 1.3 0 0 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
618     putsUART1((unsigned int*)ucLectura);
619     while(BusyUART1());
```

```
620 }
621
622 sprintf(ucLectura,"Estos datos son respecto al eje Y, oprima una tecla
        para continuar\r\n");
623 putsUART1((unsigned int*)ucLectura);
624 while(BusyUART1());
625 while(!DataRdyUART1());      // Wait until a character is received
626 uiCharacter = ReadUART1();   // Read the received character
627
628 for (i=0;i<samples;i++){
629     medir_puentes();
630     sprintf(ucLectura,"0 0 0 0 0.26 0 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
631     putsUART1((unsigned int*)ucLectura);
632     while(BusyUART1());
633 }
634
635 sprintf(ucLectura,"Estos datos son respecto al eje Y, oprima una tecla
        para continuar\r\n");
636 putsUART1((unsigned int*)ucLectura);
637 while(BusyUART1());
638 while(!DataRdyUART1());      // Wait until a character is received
639 uiCharacter = ReadUART1();   // Read the received character
640
641 for (i=0;i<samples;i++){
642     medir_puentes();
643     sprintf(ucLectura,"0 0 0 0 0.65 0 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
644     putsUART1((unsigned int*)ucLectura);
645     while(BusyUART1());
646 }
647
648 sprintf(ucLectura,"Estos datos son respecto al eje Y, oprima una tecla
        para continuar\r\n");
649 putsUART1((unsigned int*)ucLectura);
650 while(BusyUART1());
651 while(!DataRdyUART1());      // Wait until a character is received
652 uiCharacter = ReadUART1();   // Read the received character
653
654 for (i=0;i<samples;i++){
655     medir_puentes();
656     sprintf(ucLectura,"0 0 0 0 1.3 0 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
657     putsUART1((unsigned int*)ucLectura);
658     while(BusyUART1());
659 }
660
661 sprintf(ucLectura,"Estos datos son respecto al eje Z, oprima una tecla
        para continuar\r\n");
662 putsUART1((unsigned int*)ucLectura);
663 while(BusyUART1());
664 while(!DataRdyUART1());      // Wait until a character is received
665 uiCharacter = ReadUART1();   // Read the received character
```

```
666
667 for (i=0;i<samples;i++){
668     medir_puentes();
669     sprintf(ucLectura,"0 0 0 0 0 0.26 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
670     putsUART1((unsigned int*)ucLectura);
671     while(BusyUART1());
672 }
673
674 sprintf(ucLectura,"Estos datos son respecto al eje Z, oprima una tecla
        para continuar\r\n");
675 putsUART1((unsigned int*)ucLectura);
676 while(BusyUART1());
677 while(!DataRdyUART1());      // Wait until a character is received
678 uiCharacter = ReadUART1();   // Read the received character
679
680 for (i=0;i<samples;i++){
681     medir_puentes();
682     sprintf(ucLectura,"0 0 0 0 0 0.65 %04d %04d %04d %04d %04d %04d;\r\n
            ", valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
683     putsUART1((unsigned int*)ucLectura);
684     while(BusyUART1());
685 }
686
687 sprintf(ucLectura,"Estos datos son respecto al eje Z, oprima una tecla
        para continuar\r\n");
688 putsUART1((unsigned int*)ucLectura);
689 while(BusyUART1());
690 while(!DataRdyUART1());      // Wait until a character is received
691 uiCharacter = ReadUART1();   // Read the received character
692
693 for (i=0;i<samples;i++){
694     medir_puentes();
695     sprintf(ucLectura,"0 0 0 0 0 1.3 %04d %04d %04d %04d %04d %04d;\r\n"
            , valor_puente[0], valor_puente[1], valor_puente[2],
            valor_puente[3], valor_puente[4], valor_puente[5]);
696     putsUART1((unsigned int*)ucLectura);
697     while(BusyUART1());
698 }
699 }
```

7.2. Anexo II. Código fuente Interfaz de Control

```
1 'Imports System
2 'Imports System.ComponentModel
3 'Imports System.Threading
4 Imports System.IO.Ports
5 Public Class Form1
6     Dim recibido1 As String
7     Dim cadena1 As String = "" 'Buffer de recepcion para el sensor
8     Dim recibido2, cadena2 As String 'Buffer de recepcion para el robot
```

```
9     Dim P0, P1, P2, P3, P4, P5 As Integer
10    Dim Fx, Fy, Fz, Mx, May, Mz As Double
11    Dim c_i1 , c_f1 , c_i2 , c_f2 , PR1, PR2 As String
12    Dim count1 As Integer = 0
13    Dim count2 As Integer = 0
14    Dim vector_unitario(2) As Integer
15    Dim punto As String
16    Dim v_continuar As Boolean = False
17    Dim punto_inicial() As Integer = {0, 0, 0}
18    Dim punto_intermedio() As Integer = {0, 0, 0}
19    Dim punto_final() As Integer = {0, 0, 0}
20    Dim incremento() As Integer = {0, 0, 0}
21    Dim raiz As Integer
22
23
24
25    Public Sub New()
26        InitializeComponent()
27        ' Abrir puerto mientras se ejecute la aplicación
28        If Not SerialPort1.IsOpen Then
29            Try
30                SerialPort1.Open()
31            Catch ex As System.Exception
32                MessageBox.Show(ex.ToString())
33            End Try
34        End If
35
36        If Not SerialPort2.IsOpen Then
37            Try
38                SerialPort2.Open()
39            Catch ex As System.Exception
40                MessageBox.Show(ex.ToString())
41            End Try
42        End If
43        'Ejecutar la funcion Recepcion por disparo del Evento '
              DataReceived'
44        AddHandler SerialPort1.DataReceived, AddressOf Recepcion_Sensor
45        AddHandler SerialPort2.DataReceived, AddressOf Recepcion_Robot
46    End Sub
47
48
49    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.
          EventArgs) Handles MyBase.Load
50        'Dim comando As String
51        ' comando = "@SPEED 75" & vbCrLf
52        ' TextRobot.AppendText(comando)
53        'SerialPort2.WriteLine(comando)
54    End Sub
55
56    'Al recibir los datos
57    Private Sub Recepcion_Sensor(ByVal sender As Object, ByVal e As
          System.IO.Ports.SerialDataReceivedEventArgs)
58        'acumular los carÃ¡cteres recibidos a nuestro 'buffer' (string)
59        recibido1 = SerialPort1.ReadExisting()
60        'Invocar o llamar al proceso de tramas
61        Me.Invoke(New EventHandler(AddressOf Actualizar_sensor))
```

```
62
63      End Sub
64      'Al recibir los datos
65      Private Sub Recepcion_Robot(ByVal sender As Object, ByVal e As
            System.IO.Ports.SerialDataReceivedEventArgs)
66          'acumular los carácteres recibidos a nuestro 'buffer' (string)
67          recibido2 = SerialPort2.ReadExisting()
68          'Invocar o llamar al proceso de tramas
69          Me.Invoke(New EventHandler(AddressOf Actualizar_robot))
70      End Sub
71      'Multiplicación con la matriz de calibración
72      Private Sub calculo_componentes()
73          Dim v() As Integer = {P0, P1, P2, P3, P4, P5}
74          Dim C(5, 5) As Double
75          Dim comando As String
76
77          C(0, 0) = 0.017523
78          C(0, 1) = -0.0085859
79          C(0, 2) = 0.013899
80          C(0, 3) = -0.092781
81          C(0, 4) = 0.014349
82          C(0, 5) = 0.055502
83          C(1, 0) = -0.0095379
84          C(1, 1) = -0.087047
85          C(1, 2) = -0.0031638
86          C(1, 3) = 0.044903
87          C(1, 4) = 0.0030096
88          C(1, 5) = 0.051863
89          C(2, 0) = 0.0051326
90          C(2, 1) = -0.0028747
91          C(2, 2) = 0.0035497
92          C(2, 3) = -0.0096412
93          C(2, 4) = 0.0027653
94          C(2, 5) = 0.0010382
95          C(3, 0) = -0.00029734
96          C(3, 1) = -0.00012312
97          C(3, 2) = 0.0034434
98          C(3, 3) = -0.000059579
99          C(3, 4) = -0.0030458
100         C(3, 5) = 0.000083483
101         C(4, 0) = 0.0049575
102         C(4, 1) = -0.00099358
103         C(4, 2) = -0.0012309
104         C(4, 3) = -0.00034941
105         C(4, 4) = -0.0017448
106         C(4, 5) = -0.00064051
107         C(5, 0) = 0.0019018
108         C(5, 1) = -0.0021176
109         C(5, 2) = 0.0018629
110         C(5, 3) = -0.0012504
111         C(5, 4) = 0.0017143
112         C(5, 5) = -0.0021319
113
114         Fx = (C(0, 0) * v(0) + C(0, 1) * v(1) + C(0, 2) * v(2) + C(0,
                3) * v(3) + C(0, 4) * v(4) + C(0, 5) * v(5))
```

```
115         Fy = (C(1, 0) * v(0) + C(1, 1) * v(1) + C(1, 2) * v(2) + C(1,
                3) * v(3) + C(1, 4) * v(4) + C(1, 5) * v(5))
116         Fz = (C(2, 0) * v(0) + C(2, 1) * v(1) + C(2, 2) * v(3) + C(2,
                3) * v(3) + C(2, 4) * v(4) + C(2, 5) * v(5))
117
118         Mx = (C(3, 0) * v(0) + C(3, 1) * v(1) + C(3, 2) * v(3) + C(3,
                3) * v(3) + C(3, 4) * v(4) + C(3, 5) * v(5))
119         May = (C(4, 0) * v(0) + C(4, 1) * v(1) + C(4, 2) * v(4) + C(4,
                3) * v(3) + C(4, 4) * v(4) + C(4, 5) * v(5))
120         Mz = (C(5, 0) * v(0) + C(5, 1) * v(1) + C(5, 2) * v(5) + C(5,
                3) * v(3) + C(5, 4) * v(4) + C(5, 5) * v(5))
121
122         Label10.Text = Fx
123         Label11.Text = Fy
124         Label12.Text = Fz
125
126         Label28.Text = Mx
127         Label29.Text = May
128         Label30.Text = Mz
129
130         If ((Fx > 0) And (Fx <= 50)) Then
131             Fx_pos.Value = Fix(Fx) * 2
132             Fx_neg.Value = 0
133         End If
134         If ((Fx < 0) And (Fx >= -50)) Then
135             Fx_neg.Value = Fx * -2
136             Fx_pos.Value = 0
137         End If
138         If Fx = 0 Then
139             Fx_pos.Value = 0
140             Fx_neg.Value = 0
141         End If
142         If Fx > 50 Then
143             Fx_pos.Value = 100
144         End If
145         If Fx < -50 Then
146             Fx_pos.Value = 100
147         End If
148         '**************************************
149         If ((Fy > 0) And (Fy <= 50)) Then
150             Fy_pos.Value = Fy * 2
151             Fy_neg.Value = 0
152         End If
153         If ((Fy < 0) And (Fy >= -50)) Then
154             Fy_neg.Value = Fy * -2
155             Fy_pos.Value = 0
156         End If
157         If Fy = 0 Then
158             Fy_pos.Value = 0
159             Fy_neg.Value = 0
160         End If
161         If Fy > 50 Then
162             Fy_pos.Value = 100
163         End If
164         If Fy < -100 Then
165             Fy_pos.Value = 100
```

```
166             End If
167             '****************************************
168             If ((Fz > 0) And (Fz <= 50)) Then
169                 Fz_pos.Value = Fz * 2
170                 Fz_neg.Value = 0
171             End If
172             If ((Fz < 0) And (Fz >= -50)) Then
173                 Fz_neg.Value = Fz * -2
174                 Fz_pos.Value = 0
175             End If
176             If Fz = 0 Then
177                 Fz_pos.Value = 0
178                 Fz_neg.Value = 0
179             End If
180             If Fz > 50 Then
181                 Fz_pos.Value = 100
182             End If
183             If Fz < -50 Then
184                 Fz_pos.Value = 100
185             End If
186             '****************************************
187             'Mx
188             If ((Mx > 0) And (Mx <= 50)) Then
189                 Mx_pos.Value = Mx * 2
190                 Mx_neg.Value = 0
191             End If
192             If ((Mx < 0) And (Mx >= -50)) Then
193                 Mx_neg.Value = Mx * -2
194                 Mx_pos.Value = 0
195             End If
196             If Mx = 0 Then
197                 Mx_pos.Value = 0
198                 Mx_neg.Value = 0
199             End If
200             If Mx > 50 Then
201                 Mx_pos.Value = 100
202             End If
203             If Mx < -50 Then
204                 Mx_pos.Value = 100
205             End If
206             '****************************************
207             'My
208             If ((May > 0) And (May <= 50)) Then
209                 My_pos.Value = May * 2
210                 My_neg.Value = 0
211             End If
212             If ((May < 0) And (May >= -50)) Then
213                 My_neg.Value = May * -2
214                 My_pos.Value = 0
215             End If
216             If May = 0 Then
217                 My_pos.Value = 0
218                 My_neg.Value = 0
219             End If
220             If May > 50 Then
221                 My_pos.Value = 100
```

```
222         End If
223         If May < -50 Then
224             My_pos.Value = 100
225         End If
226         '****************************************
227         'Mz
228         If ((Mz > 0) And (Mz <= 50)) Then
229             Mz_pos.Value = Mz * 2
230             Mz_neg.Value = 0
231         End If
232         If ((Mz < 0) And (Mz >= -50)) Then
233             Mz_neg.Value = Mz * -2
234             Mz_pos.Value = 0
235         End If
236         If Mz = 0 Then
237             Mz_pos.Value = 0
238             Mz_neg.Value = 0
239         End If
240         If Mz > 50 Then
241             Mz_pos.Value = 100
242         End If
243         If Mz < -50 Then
244             Mz_pos.Value = 100
245         End If
246         '****************************************
247         raiz = Math.Sqrt(Fx ^ 2 + Fy ^ 2 + Fz ^ 2)
248
249         If raiz = 0 Then
250             vector_unitario(0) = 0
251             vector_unitario(1) = 0
252             vector_unitario(2) = 0
253         Else
254             vector_unitario(0) = (Fx / raiz)
255             vector_unitario(1) = (Fy / raiz)
256             vector_unitario(2) = (Fz / raiz)
257         End If
258
259         Label14.Text = vector_unitario(0)
260         Label15.Text = vector_unitario(1)
261         Label17.Text = vector_unitario(2)
262         'calcular el comando y enviarlo al robot
263         'rutina_movimiento()
264         incremento(0) = (vector_unitario(0) * 130 * System.Math.Abs(Fx)
                )
265         incremento(1) = (vector_unitario(1) * 130 * System.Math.Abs(Fy)
                )
266         incremento(2) = (vector_unitario(2) * 130 * System.Math.Abs(Fz)
                )
267
268         punto_final(0) = incremento(0) + punto_inicial(0)
269         punto_final(1) = incremento(1) + punto_inicial(1)
270         punto_final(2) = incremento(2) + punto_inicial(2)
271
272         If punto_final(0) >= 102000 Then
273             punto_final(0) = 102000
274         End If
```

```
275         If punto_final(0) <= -102000 Then
276             punto_final(0) = -102000
277         End If
278 
279         If punto_final(1) >= 120000 Then
280             punto_final(1) = 120000
281         End If
282         If punto_final(1) <= -120000 Then
283             punto_final(1) = -120000
284         End If
285 
286         If punto_final(2) >= 42000 Then
287             punto_final(2) = 42000
288         End If
289         If punto_final(2) <= 0 Then
290             punto_final(2) = 0
291         End If
292 
293         ' PR1 = punto_intermedio(0) & " " & punto_intermedio(1) & " " &
                punto_intermedio(2) & " 0 0 0"
294         '  comando = "@P0= " & PR1 & vbCrLf
295         '  SerialPort2.WriteLine(comando)
296         '  TextRobot.AppendText(comando)
297         ' WaitSeconds(0.3)
298 
299         '  PR2 = punto_final(0) & " " & punto_final(1) & " " &
                punto_final(2) & " 0 0 0"
300         '   comando = "@P1= " & PR2 & vbCrLf
301         ''    SerialPort2.WriteLine(comando)
302         '   TextRobot.AppendText(comando)
303         ' WaitSeconds(0.3)
304 
305         comando = "@MOVE P, " & punto_final(0) & " " & punto_final(1) &
                " " & punto_final(2) & " 0 0 0" & vbCrLf
306         'comando = "@MOVE L,P0,P1" & vbCrLf
307         'SerialPort2.WriteLine(comando)
308         TextRobot.AppendText(comando)
309         'punto_inicial = punto_intermedio
310         ' punto_intermedio = punto_final
311         punto_inicial = punto_final
312 
313 
314 
315 
316     End Sub
317     'Procesar los datos reciidos en el buffer  y extraer tramas
            completas
318     Private Sub Actualizar_sensor(ByVal s As Object, ByVal e As
            EventArgs)
319         'asignar el valor de la trama al textbox
320         TextSensor.AppendText(recibido1)
321         cadena1 += recibido1
322         count1 = Len(cadena1)
323         If count1 = 28 Then
324             c_i1 = Microsoft.VisualBasic.Left(cadena1, 1)
325             If c_i1 = "I" Then
```

```
326                 P0 = Mid(cadena1, 2, 4)
327                 Label4.Text = P0
328                 P1 = Mid(cadena1, 6, 4)
329                 Label5.Text = P1
330                 P2 = Mid(cadena1, 10, 4)
331                 Label6.Text = P2
332                 P3 = Mid(cadena1, 14, 4)
333                 Label7.Text = P3
334                 P4 = Mid(cadena1, 18, 4)
335                 Label8.Text = P4
336                 P5 = Mid(cadena1, 22, 4)
337                 Label9.Text = P5
338                 cadena1 = ""
339                 count1 = 0
340             End If
341         End If
342         If count1 > 28 Then
343             cadena1 = ""
344         End If
345     End Sub
346     'Procesar los datos reciidos en el buffer  y extraer tramas
            completas
347     Private Sub Actualizar_robot(ByVal s As Object, ByVal e As
            EventArgs)
348         'asignar el valor de la trama al textbox
349         TextRobot.AppendText(recibido2)
350     End Sub
351
352     Private Sub Ajustar_Click(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles Ajustar.Click
353         Dim mBuffer As Byte() = New Byte(0) {}
354         mBuffer(0) = Asc("a")
355         SerialPort1.Write(mBuffer, 0, mBuffer.Length)
356     End Sub
357
358     Private Sub Empezar_Click(ByVal sender As System.Object, ByVal e As
            System.EventArgs) Handles Empezar.Click
359         Dim mBuffer As Byte() = New Byte(0) {}
360         mBuffer(0) = Asc("c")
361         SerialPort1.Write(mBuffer, 0, mBuffer.Length)
362         WaitSeconds(0.3) 'Espera que se separen los datos en variables
                independientes
363         calculo_componentes() 'calculo de componentes y envÃo al Robot
                del comando
364         WaitSeconds(0.3)     'espera a que el robot complete la
                instrucción
365
366         While continuo.Checked = True    'si está marcado se queda en el
                loop haciendo lo mismo
367             SerialPort1.Write(mBuffer, 0, mBuffer.Length)
368             WaitSeconds(0.3) 'Espera que se separen los datos en
                    variables independientes
369             calculo_componentes() 'calculo de componentes y envÃo al
                    Robot del comando
370             WaitSeconds(0.3)     'espera a que el robot complete la
                    instrucción
```

```
371             End While
372
373         End Sub
374         Private Sub WaitSeconds(ByVal nSecs As Double)
375             ' Esperar los segundos indicados
376             ' Crear la cadena para convertir en TimeSpan
377             Dim s As String = "0.00:00:" & nSecs.ToString.Replace(",", ".")
378             Dim ts As TimeSpan = TimeSpan.Parse(s)
379             ' Añadirle la diferencia a la hora actual
380             Dim t1 As DateTime = DateTime.Now.Add(ts)
381             ' Esta asignación solo es necesaria
382             ' si la comprobación se hace al principio del bucle
383             Dim t2 As DateTime = DateTime.Now
384             ' Mientras no haya pasado el tiempo indicado
385             Do While t2 < t1
386                 ' Un respiro para el sitema
387                 System.Windows.Forms.Application.DoEvents()
388                 ' Asignar la hora actual
389                 t2 = DateTime.Now
390             Loop
391         End Sub
392
393         Private Sub Origen_Click(ByVal sender As System.Object, ByVal e As
                System.EventArgs) Handles Origen.Click
394             Dim mBuffer As String
395             mBuffer = "@ORIGIN" & vbCrLf
396             TextRobot.AppendText(mBuffer)
397             SerialPort2.WriteLine(mBuffer)
398             punto_inicial(0) = 0
399             punto_inicial(1) = 0
400             punto_inicial(2) = 0
401
402         End Sub
403
404 End Class
```