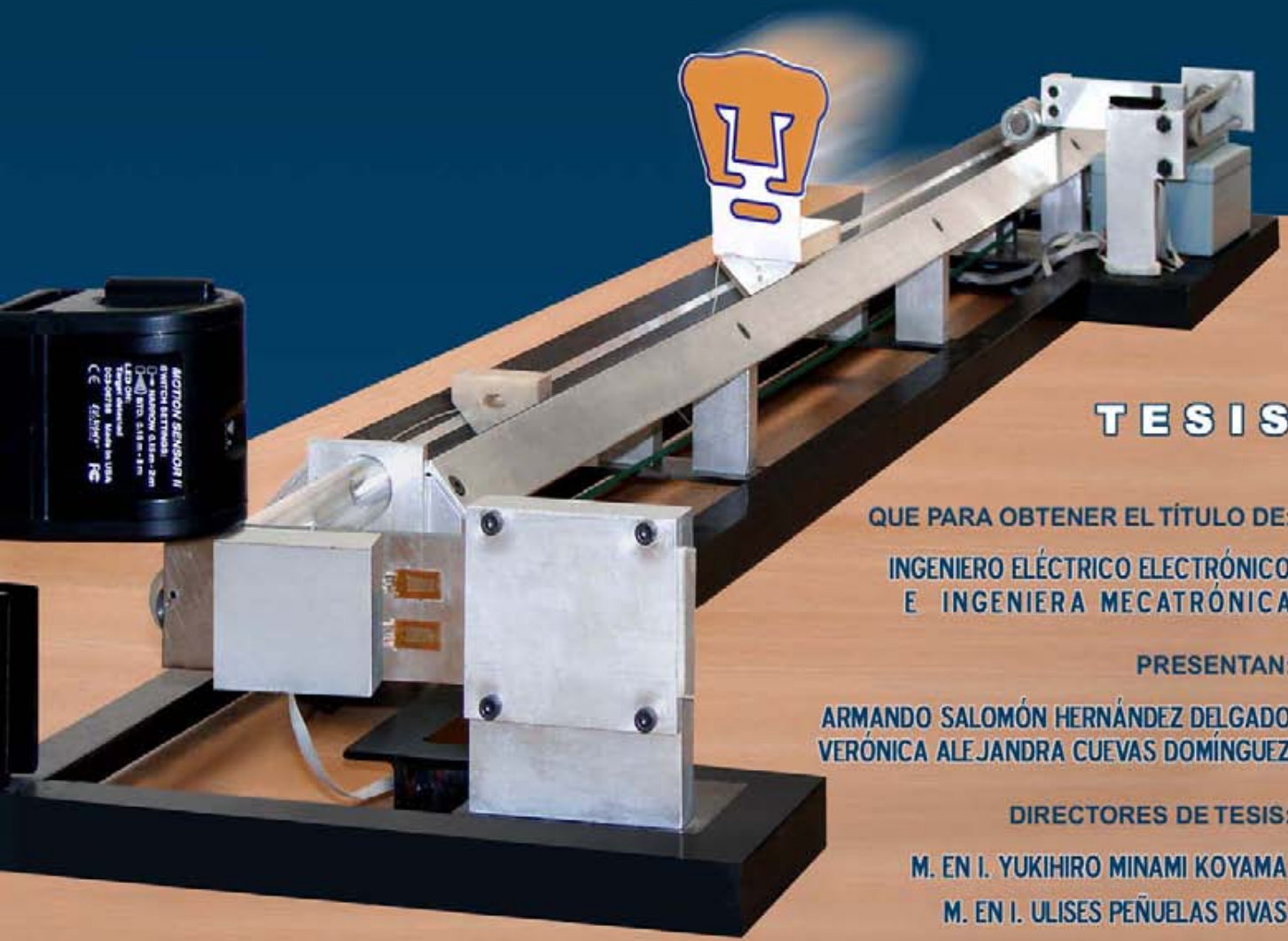
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INSTRUMENTACIÓN Y CONTROL PARA LA OPERACIÓN REMOTA DE LA PRÁCTICA DE TRABAJO Y ENERGÍA



TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO
E INGENIERA MECATRÓNICA

PRESENTAN:

ARMANDO SALOMÓN HERNÁNDEZ DELGADO
VERÓNICA ALEJANDRA CUEVAS DOMÍNGUEZ

DIRECTORES DE TESIS:

M. EN I. YUKIHIRO MINAMI KOYAMA
M. EN I. ULISES PEÑUELAS RIVAS

CD. UNIVERSITARIA, MÉXICO, D. F. ENERO 2009

RESUMEN

En estos tiempos donde el empleo de sistemas tecnológicos y automatizados se ha hecho común, es importante que los métodos de enseñanza estén también a la vanguardia.

Debido principalmente al uso de Internet y el desarrollo de las tecnologías web en los últimos años, la educación ha presentado un cambio radical. Los cursos a distancia, los libros electrónicos y las plataformas interactivas de educación a distancia juegan un papel cada vez más importante en el proceso de aprendizaje.

En este proyecto se diseñaron la interfaz electrónica y el control de un dispositivo masa-resorte que fue construido para su uso en la práctica Trabajo y Energía, como parte de la Creación de un laboratorio remoto para la asignatura Cinemática y Dinámica de la Facultad de Ingeniería, en la Universidad Nacional Autónoma de México.

En el presente trabajo de tesis se describen las características principales del sistema a controlar, el diseño e implementación de la interfaz electrónica de dicho control y la interfaz con el usuario. Además se presenta la comunicación por vía puerto serial de la computadora con el microcontrolador maestro y del microcontrolador maestro al microcontrolador esclavo utilizando herramientas de programación como Mplab para microcontroladores y Visual Studio en lenguaje C# para aplicaciones web.

En las conclusiones se presentan las pruebas realizadas y los resultados obtenidos.

ABSTRACT

Today, the use of technological systems and automated has become common, because of this it is important that the teaching methods are at the top too.

Due mainly to the development and use of Internet and Web technologies in recent years, education has submitted a radical change. The Web-based learning, distance courses, books and electronic interactive platforms distance learning plays an increasingly important role in the learning process.

In this project was designed the control and the user interface of a mass-spring device that was built for use in "Work-Energy" practice for the creation of a remote laboratory for the Kinematics and Dynamics subject of the Engineering Faculty at the Universidad Nacional Autónoma de México.

This thesis work reports the design and implementation of the electronic interface for the control and user interface. To do this, we used principles of programming in C language for microcontrollers and used software for the communication to the computer.

In the paper describes the main features of the system to control. In addition presents the via serial port communication of the computer with the master microcontroller, and master microcontroller to the slave microcontroller using tools such as Visual Studio and MPLAB in C # language. The final chapter presents the tests performed and results obtained.

CONTENIDO

CAPÍTULO 1
INTRODUCCIÓN

1.1 PROBLEMÁTICA

En la Facultad de Ingeniería de la Universidad Nacional Autónoma de México se imparte la asignatura Cinemática y Dinámica, la cual consta de un laboratorio tradicional como espacio de experimentación tanto de estudiantes como de profesores. Es evidente, en el ámbito académico, la enorme importancia que tiene para el aprendizaje la experimentación directa del alumno en el laboratorio tradicional, ya que permite reforzar lo aprendido en las aulas de clase, facilitando el planteamiento de problemas que permiten al estudiante la aplicación de los conocimientos adquiridos, entrenándose en la aplicación del método científico. La principal ventaja del laboratorio tradicional es su alta interactividad, al poner en contacto al alumno con el experimento real, la motivación que supone observar el experimento, el desarrollo de habilidades cognitivas que se ponen en práctica en el mismo, etc.

Aunque el laboratorio tradicional es un lugar idóneo de experimentación, también presenta inconvenientes, entre los que destacan:

a) El material de instrumentación es excepcionalmente caro, lo que hace difícil que cada alumno pueda realizar todos los experimentos que necesite.

b) Los recursos en personas y espacios son restringidos, debido a la masificación de la enseñanza por problemas presupuestarios.

c) Las prácticas necesitan de una supervisión más directa por parte del profesor y que cada alumno experimente por sí mismo, por lo que éstas no se pueden impartir para un gran número de personas.

d) El laboratorio tradicional requiere de la presencia física del estudiante.

1.2 JUSTIFICACIÓN

Por tales motivos, se creó el proyecto EN106204 del PAPIME (Programa de Apoyo a Proyectos para la Innovación y Mejoramiento de la Enseñanza) que lleva por nombre "Creación de un laboratorio remoto accedido por medio de la Internet para la asignatura Cinemática y Dinámica", en el que se desea construir un laboratorio remoto el cual pueda ser utilizado por cualquier alumno con acceso a Internet para la realización de las prácticas de esta asignatura y disminuir la saturación que actualmente existe en los laboratorios; pretende también, brindar su servicio a más alumnos para que puedan realizar las prácticas de dicha asignatura.

Este proyecto tiene contemplado desarrollar seis prácticas diferentes que podrán ser realizadas de manera remota. Los objetivos de tal proyecto se concentran en diversos aspectos como son: el diseño de las prácticas; el diseño, fabricación o adaptación e implementación de los elementos mecánicos, electrónicos y de control, requeridos para la realización de cada práctica y el desarrollo del software que se necesita para el funcionamiento del laboratorio remoto el cual debe ejecutarse en un servidor configurado particularmente para proporcionar el servicio.

El presente trabajo hace referencia a una práctica, llamada "Trabajo y Energía", donde se obtiene la relación existente entre esfuerzos (fuerzas) y deformaciones en un resorte, conocida como la ley de Hooke y además, se confirma experimentalmente el Principio del Trabajo y Energía, donde se realiza trabajo por medio de la energía almacenada en el resorte, determinando la energía cinética del móvil (bloque) y adicionalmente calculando el coeficiente de fricción entre las superficies involucradas.

1.3 OBJETIVOS

En un trabajo de tesis previo se diseñó solamente el mecanismo con el cual se realizará la práctica. **Los objetivos principales del presente trabajo son la implementación de la instrumentación y la página web que permitirá controlar desde Internet el sistema masa-resorte.**

Como parte de este trabajo se desarrollaron los siguientes puntos:

1 Selección e implementación de los sensores, actuadores y controladores
2 Diseño de la interfaz electrónica y del usuario
3 Puesta en marcha del sistema desde internet.

Los alcances específicos fueron:

1 Desarrollo de la interfaz (hardware y software) para el control de los actuadores del sistema masa-resorte
2 Construcción de los sensores y del programa de adquisición de datos para la PC
3 Creación de la interfaz web con el usuario para controlar el sistema desde internet, de tal forma que sea sencilla de abordar y donde se presenten los datos obtenidos en cada experimento.

No se abordaron temas respecto a la cámara que se instalará en el laboratorio remoto, ni sobre la seguridad y el control de acceso desde la internet al servidor.

CAPÍTULO 2
ANTECEDENTES

2.1 LABORATORIOS REMOTOS

Los laboratorios remotos son laboratorios donde los usuarios pueden interactuar con dispositivos reales a través de la Internet. Normalmente los usuarios a través de una interfaz web pueden cambiar algunos parámetros de control, realizar experimentos, ver los resultados y descargar los datos del experimento [22].

Los campos de aplicación son muchos, hay laboratorios remotos de química, mecánica, biología o biomédica. Pero las dos áreas que están más avanzadas y en las que se investiga más con “WebLabs” son la electrónica y la automática.

Originalmente, los primeros laboratorios remotos se desarrollaron a finales de los noventa en Estados Unidos, los Collaboratories, y en Europa el proyecto DYNACORE, en 1996, que permite el control de un telescopio en Canarias. A partir de esa semilla se van desplegando laboratorios remotos en distintos países y con distintos usos [22].

Actualmente existen muchas universidades que cuentan con laboratorios remotos, algunos ejemplos son:

En España, la Universidad Politécnica de Valencia, la UNED, la Universidad de León, la de Deusto en el País Vasco, etc.

En América Latina NETLAB INACAP, es el laboratorio remoto más grande, que consiste en una combinación de hardware y software, que permite tener acceso a los estudiantes e instructores a los equipo de red Cisco en forma real, solamente utilizando una conexión a la Internet y un navegador web.

En México, existe la Corporación Universitaria para el Desarrollo de Internet (CUDI) que tiene como objetivo fomentar la colaboración, el intercambio de información y el desarrollo de proyectos conjuntos entre las instituciones miembros de CUDI, para utilizar la red Internet 2.

En el Instituto de Física de la UNAM se encuentra el Laboratorio de Microscopía que es el primer Laboratorio en México que ha sido compartido vía Internet mediante una colaboración virtual.

El Tecnológico de Monterrey comenzó a operar este año el proyecto de una Red Intercampus de Laboratorios Remotos, que busca impulsar el desarrollo de nuevas metodologías educacionales dentro de las áreas de automatización, eléctrica, electrónica y control.

2.2 DESCRIPCIÓN DE LA PRÁCTICA TRABAJO Y ENERGÍA

Los objetivos específicos de la práctica son obtener la constante del resorte de la ecuación de la Ley de Hooke, la rapidez con que se desplaza el bloque, y el coeficiente de fricción entre el bloque y la superficie sobre la que se desplaza. A continuación se muestra una lista del equipo empleado actualmente para realizar los experimentos:

a) un plano de laminado plástico
b) un resorte
c) una placa de sujeción para resorte
d) un dinamómetro
e) un bloque de madera
f) un flexómetro
g) dos metros de hilo delgado

La práctica se realiza en dos partes:

a) la caracterización del resorte
b) la realización de trabajo por medio de la energía almacenada en el resorte.

En la primera parte se mide la fuerza y la distancia, y en la segunda sólo la distancia recorrida.

En la actualidad la práctica se lleva a cabo manualmente, y para obtener la pareja de valores $k = f\left(d_i, F_i\right)$ de un resorte, se procede de la siguiente forma:

1 Se coloca el extremo A del resorte en la placa de sujeción y se unen los extremos B del resorte y C del dinamómetro, tal como se indica en la Figura 2.1.

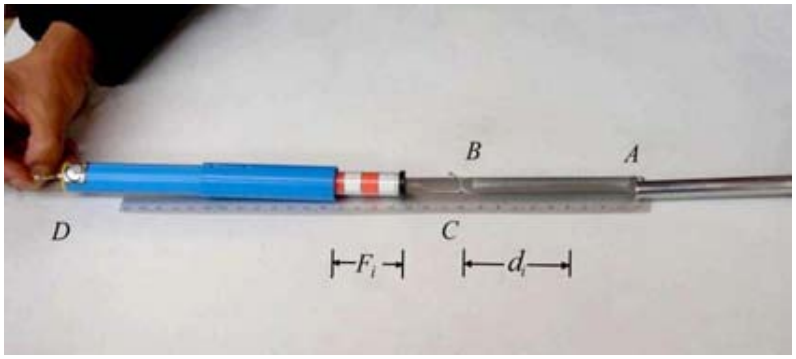


Figura 2.1 Caracterización del resorte. (A) extremo del resorte colocado en la placa, (B) otro extremo del resorte colocado en (C) un extremo del dinamómetro, (D) otro extremo del dinamómetro, (F_i) fuerza aplicada en newtons, (d_i) deformación del resorte en metros.

2 Se sujeta el extremo D del dinamómetro y se desplaza horizontalmente; este efecto produce una deformación x en el resorte, debido a la fuerza aplicada en el dinamómetro y que es la misma fuerza que se transmite al resorte; el valor de su módulo queda registrado en el vástago de lectura de dicho instrumento, tal como se muestra en la Figura 2.1, en la que:

d_i = deformación del resorte, en metros.
F_i = fuerza del resorte, en newtons.

Con base en el procedimiento anterior, se registran experimentalmente 10 parejas diferentes de datos (d_i, F_i).

3 Después de obtener las parejas (d_i, F_i), se arma el arreglo que se muestra en la Figura 2.2.

Figura 2.2 (d) Deformación inicial cero, (1) Posición inicial del bloque.

4 A continuación, se desplaza el bloque una distancia d cualquiera, no necesariamente igual a las registradas anteriormente, con el objeto de deformar el resorte esa misma distancia, tal como se indica en la Figura 2.3.

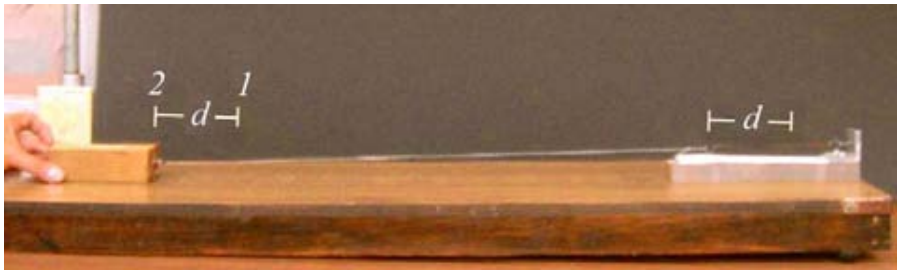


Figura 2.3 (d) Deformación, (1) Posición inicial del bloque,
(2) Posición intermedia del bloque.

5 Por último, se suelta el bloque, dejándolo mover hasta que se detenga y se registra el alcance máximo L, medido a partir de la posición desde la cual se soltó, tal como se indica en la Figura 2.4.

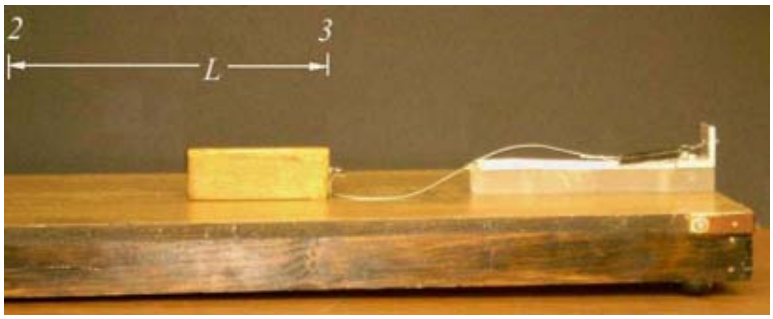


Figura 2.4 (2) Posición intermedia de la masa, (3) Posición final del bloque después de ser liberada y haber recorrido una distancia (L).

Con base en el procedimiento presentado en el punto anterior, se reproduce el experimento 10 veces para una misma distancia d, hasta obtener 10 desplazamientos L.

Realizar la práctica de esta manera permite que se presenten errores experimentales como: desvío del bloque durante su desplazamiento, en ocasiones el bloque llega a chocar contra el resorte por mal empleo de los instrumentos, mala medición de las lecturas, entre otras. Por ello, se construyó un dispositivo capaz de llevar a cabo estas acciones disminuyendo los errores que pudieran existir. Se busca que el sistema tenga una respuesta controlada, a fin de lograr un buen desempeño y hacer la práctica accesible vía Internet, permitiendo que el alumno dedique más tiempo al análisis de los datos experimentales que a la realización de los experimentos.

2.3 DESCRIPCIÓN DEL SISTEMA MASA-RESORTE

Las partes principales que integran el sistema masa-resorte son:
- bastidor
- sistema de deslizamiento
- soporte para el sensor de fuerza y guía para el resorte
- transmisión de recarga
- transmisión de precisión y sujeción de la masa
- soportes para sensores

A continuación se describe brevemente el funcionamiento de cada una de las partes.

2.3.1 BASTIDOR

Se divide en dos partes: bastidor principal y extensión del bastidor. Es la base para fijar todos los elementos del prototipo

El bastidor principal (Figura 2.5), es la estructura donde se monta el sistema de deslizamiento, la transmisión de recarga, el soporte para el sensor de fuerza y resorte, así como los soportes adicionales para la transmisión de recarga y la medición de distancia recorrida por la masa.

Figura 2.5 Bastidor principal.

La extensión del bastidor (Figura 2.6), contiene únicamente al mecanismo de precisión y sujeción del bloque.

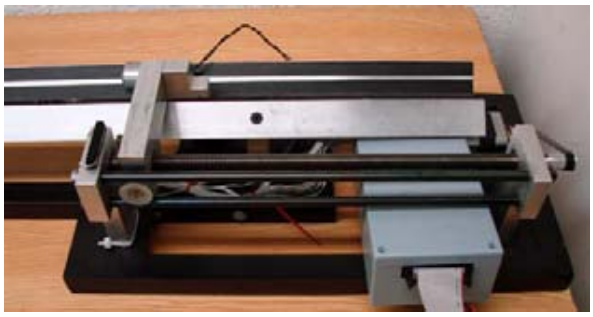


Figura 2.6 Extensión del bastidor.

2.3.2 SISTEMA DE DESLIZAMIENTO

La pista o sistema de deslizamiento construida (Figura 2.7), está compuesta de dos ángulos de aluminio dispuestos en forma de M con una separación intermedia entre crestas y manteniendo el ángulo de 90° entre las caras de la pista donde desliza el móvil, con una longitud de 1.6 *m*. La ranura entre las dos caras de la pista permite que sobresalga un cuerpo que es movido por la transmisión de recarga, situada por debajo de la pista, con el objeto de llevar al bloque a la posición inicial.

Figura 2.7 Pista de deslizamiento.

El bloque (Figura 2.8) tiene una masa de 50 *g* y se conforma de una caja que se ajusta a la geometría de la pista y la cara opuesta al frente, contiene una lámina delgada de fierro que puede sujetar un electroimán para deformar al resorte y, en la cara posterior la imagen de un puma que permitirá al sensor de distancia detectar la posición del bloque. La unión entre el resorte y el móvil se hace mediante hilo de nylon de caña de pescar.

Figura 2.8 Bloque o masa del sistema.

2.3.3 SOPORTE PARA EL SENSOR DE FUERZA Y GUÍA PARA EL RESORTE

En este soporte se fija el sensor de fuerza, el cual es una lámina de acero inoxidable que sirve como base para los transductores (galgas extensiométricas) y tiene un orificio en uno de sus extremos donde se sujetará el resorte (Figura 2.9).

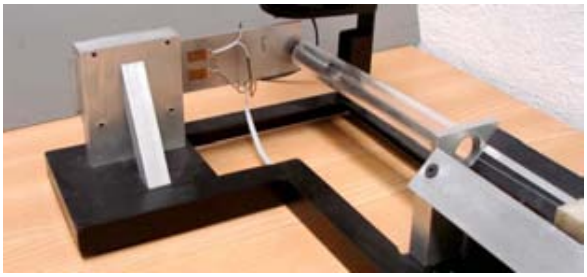


Figura 2.9 Soporte del sensor, guía del resorte y resorte.

El sensor de fuerza sirve como soporte para el resorte, el cual al estar en reposo tiende a caerse, para evitar esto, vibraciones que se puedan generar al soltarlo, e impedir que llegue a enredarse con el hilo, se utilizó un tubo de acrílico como guía del resorte (Figura 2.9).

2.3.4 TRANSMISIÓN DE PRECISIÓN Y SUJECIÓN DEL MÓVIL

Consta de un husillo, el cual es movido mediante un motor de pulsos con resolución de 1.8° por paso. Para transmitir el movimiento del motor al husillo se usa una banda dentada (Figura 2.10).

El husillo puede proporcionar una carrera de 279 *mm,* para el desplazamiento lineal del carro que sujeta el bloque, tiene un diámetro de 19.5 *mm* y una relación de 2.54 *mm* por vuelta del husillo.

La sujeción del bloque se hace por medio de un electroimán, está acoplado a un carro que contiene una tuerca de bronce, y se desplaza con el movimiento del husillo de la transmisión de precisión, auxiliado de una carretilla y dos barras de acero, las cuales sirven de guía.

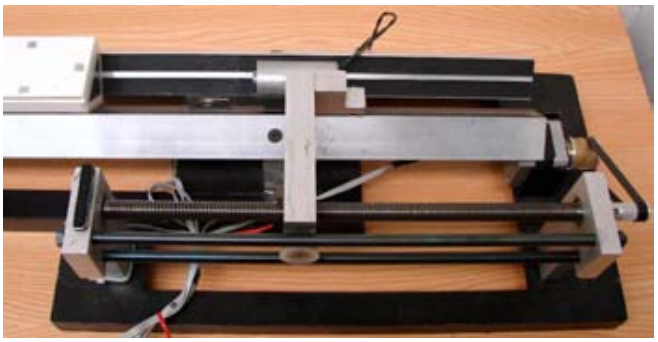


Figura 2.10 Sistema de transmisión de precisión.

2.3.5 TRANSMISIÓN DE RECARGA

Esta transmisión trabaja por debajo de la pista (Figura 2.11), dejando sobresalir únicamente la parte superior del cuerpo que posiciona al bloque. Un motor de corriente continua se sitúa al principio de la pista, junto al primer soporte, de modo que la polea auxiliar se encuentra junto al último soporte de la pista.

2.3.6 SOPORTES PARA SENSORES

Para detectar cuándo el cuerpo que empuja a la masa ha llegado al fin de carrera establecido, se utilizan sensores fotoeléctricos (Figura 2.11), sus soportes se encuentran ubicados por debajo de la pista en sus extremos.

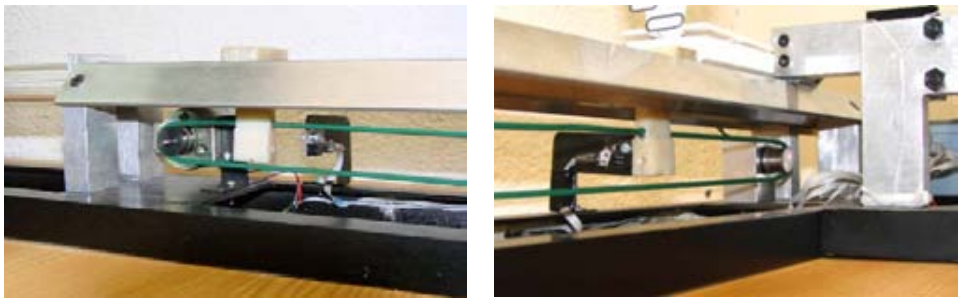


Figura 2.11 Sistema de transmisión de recarga y sensores de fin de carrera.

Las características que tiene el prototipo ya terminado son las siguientes:

• longitud total: 1920 *mm*
• ancho total: 315 *mm*
• longitud total de la pista: 1600 *mm*
• carrera del bloque empujador de recarga: 1060 *mm*
• avance del carro: 2.54 *mm* por vuelta del husillo
• motor de pulsos con 1.8° por paso con volante de inercia
• bloque de 50 *g*.

2.4 REQUERIMIENTOS PARA LA INSTRUMENTACIÓN Y EL CONTROL

En esta parte se revisarán los actuadores del sistema, identificando sus funciones y especificando sus límites de operación que determinarán los sensores necesarios para la implementación del control electrónico y su ubicación en el prototipo.

Se necesita definir la posición inicial (HOME), la máxima deformación del resorte, y el inicio y final de carrera de los actuadores.

La posición inicial del sistema (HOME) es la posición en la que el resorte se encuentra en su longitud natural, el hilo que lo conecta con el bloque (móvil) está totalmente extendido sobre el riel, y el electroimán y el bloque posicionador se encuentran en su inicio de carrera; el electroimán y el bloque (móvil) están en contacto y el riel está despejado para permitir el libre deslizamiento de la masa.

El inicio de carrera del bloque posicionador se encuentra al principio del riel y su final de carrera a 1.060 *m* adelante, donde se encuentra el inicio de carrera del electroimán. La deformación máxima que se le puede aplicar al resorte son 279 *mm* que es la longitud total del husillo.

Una vez definidos estos parámetros se procederá a identificar las funciones de los actuadores para establecer los requerimientos de control.

Para la recarga, el actuador consiste en una transmisión banda polea accionada por un motor de corriente continua, la banda es deslizante de sección transversal circular, conectada al bloque posicionador cuya función es llevar a la posición inicial al bloque deslizándolo sobre el la pista. Para esto se necesita detectar cuándo el bloque posicionador llega al inicio de carrera y al final de carrera.

En el caso de la sujeción del bloque (móvil), hay que identificar cuándo el electroimán ha llegado a su inicio de carrera, sin que el carro que lo transporta choque con los soportes del husillo, ya que si lo hace, es posible que el motor de pulsos se sobrecargue pudiendo dañarse éste y el circuito que lo alimenta. Además, si el carro se pone en contacto con los soportes, pueden llegar a apretarse el husillo y la tuerca y no seria posible desbloquearlo remotamente ya que el motor no tiene el par necesario para hacerlo.

También se requiere asegurar que la sujeción del bloque se lleve a cabo, que no se sobrepase la deformación máxima del resorte y que al liberar el bloque, éste no choque en su desplazamiento por la pista con el bloque posicionador, ubicado al inicio de ésta.

El manejo de este sistema se deberá realizar accediendo a una página de Internet donde se presenten las actividades de la práctica y los resultados de las mediciones obtenidas en los experimentos.

REQUERIMIENTOS DE MEDICIÓN

Fuerza: resolución de 0.1 *N* y rango de 2 *N* que es la que se obtiene con un dinamómetro con el que se realizaba la práctica en el laboratorio.

Distancia: resolución 1 *mm* y rango de 250 *mm* al deformar el resorte y, 1 *mm* de resolución y rango de 1.31 *m* (1.060 *m* + 0.250 *m*) al deslizar la masa. La resolución mínima de 1 *mm* es la obtenida con un flexómetro al realizar la práctica de forma tradicional, o bien, usando el sensor de distancia ultrasónico empleado actualmente en el laboratorio.

Se desea que los sistemas de medición de fuerza y distancia sean de fácil adquisición, operación e implementación en el sistema.

CAPÍTULO 3
IMPLEMENTACIÓN DE LOS SENSORES

En este capítulo se aborda el diseño y construcción del sensor de fuerza empezando por identificar la variable física a medir, los tipos de transductores que se pueden utilizar, la selección y construcción del instrumento y la integración de éste al dispositivo experimental.

También se aborda la integración del sensor de posición ultrasónico con que se contaba.

3.1 SENSOR DE FUERZA

3.1.1 FUNDAMENTOS

Cuando se aplican fuerzas externas a un objeto estacionario, se obtienen como resultado un esfuerzo que se opone y una deformación del material. El análisis experimental del esfuerzo se realiza al medir la deformación de un elemento bajo carga e inferir el estado del esfuerzo que existe a partir de las deflexiones medidas.

En el lenguaje técnico, el esfuerzo se puede definir como la resultante de las fuerzas internas del objeto que se opone a las fuerzas externas; la deformación, como la extensión o compresión que sufre el objeto.

Para ilustrar estos conceptos, se considera una barra delgada que se coloca en reposo, luego en tensión y después en compresión uniaxial como se muestra en la Figura 3.1.

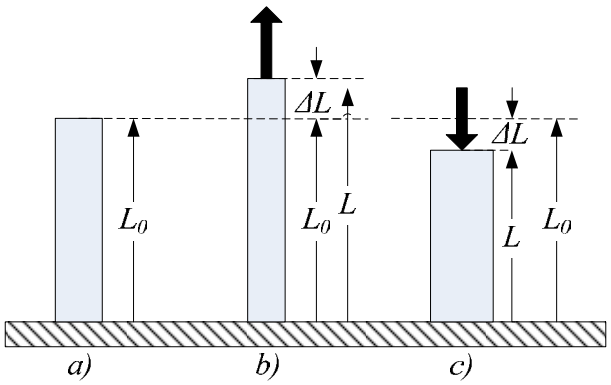


Figura 3.1 Barra delgada a) en reposo, b) sometida a tensión, c) sometida a compresión.

Para una distribución de fuerzas internas homogénea o constante, el esfuerzo σ, puede ser calculado como la relación de la fuerza aplicada, F, por unidad de área de sección transversal, A, de la barra.

$$\sigma = \frac{F}{A} \tag{3.1}$$

La variación relativa de longitud, ε, es el cociente del cambio de longitud, ΔL, respecto a la longitud de referencia, L_0. La letra griega ε es usada para representar este cociente.

$$\varepsilon = \frac{\Delta L}{L_0} \tag{3.2}$$

La variación relativa de longitud es un término técnico estandarizado llamado deformación (‘strain’ en inglés), este término se utiliza para procesos de alargamiento de elementos (deformación por tensión) así como para acortamiento (deformación por compresión). En el primer caso se habla de una deformación positiva y en el segundo de una deformación negativa.

De forma análoga, la diferencia de longitud, ΔL, es positiva para tensión y negativa para compresión, y en la mayoría de los casos es muy pequeño, en comparación con la longitud de referencia o reposo, L_0; por lo común pueden ser algunos milímetros o incluso micrómetros; este cambio de longitud suele especificarse en *μm/m*.

En la práctica, los esfuerzos mecánicos en materiales no se pueden medir puntualmente; por otra parte, sí se puede medir la deformación, ya que ésta se presenta en cualquier parte del material bajo esfuerzo, pudiendo relacionar directamente la deformación con el esfuerzo. Para efectos prácticos, esta deformación se mide sobre la superficie del material.

El principio de medición de esfuerzos mecánicos en materiales a través de la medición de la deformación que experimentan, fue presentado por primera vez por el científico inglés Roberto Hooke en el año de 1678, de forma que es válida para la condición de esfuerzo uniaxial.

En el estudio experimental de las propiedades mecánicas de los materiales, se acostumbra construir gráficas de la relación entre esfuerzo y deformación en una prueba en particular. En estos diagramas semejantes al de la Figura 3.2**,** se acostumbra utilizar el eje de ordenadas para el esfuerzo y el de abscisas para la deformación. Los diagramas esfuerzo-deformación que se determinan en forma experimental difieren mucho según los distintos materiales; aun para el mismo material difieren según la temperatura a la que se efectúa el ensayo, la velocidad de prueba y otras variables.

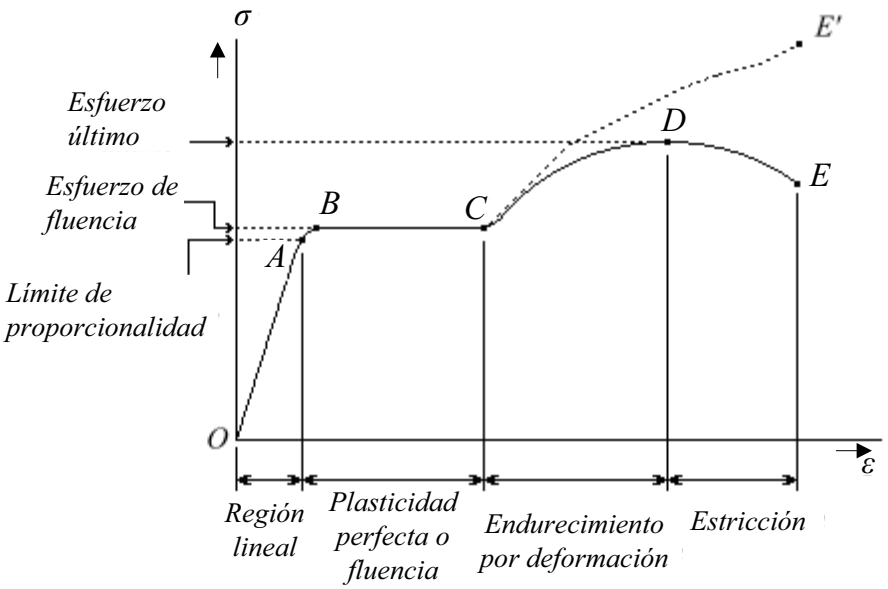


Figura 3.2 Diagrama esfuerzo – deformación.

Un aspecto importante que se debe observar en estos diagramas, es el punto A, que está sobre una recta que parte del origen y sigue estrechamente la trayectoria de la gráfica. Este punto se llama límite de proporcionalidad del material. La pendiente de la recta desde O hasta A representa la característica del material llamada *módulo de elasticidad*, E, o *módulo de Young,* la cual representa la rigidez del material a una carga impuesta.

Ahora, se puede establecer la relación entre la deformación de un material y el esfuerzo al que se, somete si se conoce la gráfica esfuerzo-deformación de dicho material, o simplemente su módulo de elasticidad E, que se hallan en tablas de características de materiales.

$$\sigma = E \cdot \varepsilon \tag{3.3}$$

Se puede decir que hasta cierto punto, por ejemplo, el punto A, la relación entre esfuerzo-deformación es lineal para todos los materiales. Esta idealización y generalización es la base de la ley de Hooke. Por tanto, dicha ley sólo aplica hasta el límite de proporcionalidad del material.

Las relaciones establecidas forman la base del análisis experimental de esfuerzos, y además son usadas para determinar el esfuerzo de un material a partir de su deformación. La mayoría de los transductores para medición de fuerza se basan en este principio.

3.1.2 DISEÑO E IMPLEMENTACIÓN

3.1.2.1 Transductores

En un sistema de instrumentación electrónico se pueden identificar las siguientes etapas principales: un dispositivo de entrada, un dispositivo acondicionador de señal o de procesamiento, y un dispositivo de salida.

La variable física del dispositivo de entrada no es eléctrica, en la mayoría de los casos. Con el fin de utilizar métodos y técnicas de medición eléctricos, las cantidades no eléctricas se convierten en una señal eléctrica por medio de un transductor.

“El transductor es un dispositivo que al ser afectado por la energía de un sistema de transmisión, proporciona energía en la misma forma o en otra a un segundo sistema de transmisión” [3]. Ésta transmisión de energía puede ser eléctrica, mecánica, química, óptica (radiante) o térmica.

Esta definición incluye, por ejemplo, dispositivos que convierten fuerza o desplazamiento mecánico en una señal eléctrica.

La *selección del transductor* apropiado, es tal vez el primer y más importante paso en la obtención de resultados exactos. Se deben responder algunas preguntas antes de seleccionar un transductor, por ejemplo:

a) ¿Cuál es la cantidad física a medir?
b) ¿Cuál principio de transducción es el mejor para medir esta cantidad?
c) ¿Qué exactitud se requiere en esta medición?

Se puede empezar respondiendo que la cantidad física que se medirá será un desplazamiento, pues como se analizó anteriormente, el esfuerzo en un material está relacionado con la deformación que éste sufre, y que es más notoria en su superficie.

La exactitud que se requiere se determina con base en las mediciones que se han realizado hasta ahora en el laboratorio de Cinemática y Dinámica con un dinamómetro con rango de 0 a 10 *N*, con resolución de 0.1 *N*; el rango puede variar en función de la rigidez del resorte que se utilizará para el equipo de experimentación, ya que si es muy rígido aumentará o si es blando se reducirá.

Para determinar cuál será el mejor principio de transducción, se revisará solamente aquéllos que se utilizan para medir fuerza y desplazamiento.

La Tabla 3.1 muestra una clasificación de éstos, según el principio eléctrico en que se basan. Se listan los transductores que requieren potencia externa, llamados transductores pasivos, los cuales producen una variación en algún parámetro eléctrico como resistencia, capacitancia, etc, que se puede medir como una variación de tensión o corriente.

Tabla 3. 1 Transductores pasivos de desplazamiento y fuerza.

Parámetro eléctrico y clase de transductor	Principio de operación y naturaleza del dispositivo	Aplicación típica
Resistencia		
Potenciómetro	El posicionamiento de un cursor por medio de una fuerza externa varía la resistencia de un potenciómetro o circuito puente	Presión, desplazamiento
Galga extensiométrica	La resistencia de un alambre o semiconductor cambia según la elongación o compresión debida a esfuerzos aplicados externamente	Fuerza, par, desplazamiento
Capacitancia		
Medidor de presión de capacitancia variable	Una fuerza aplicada externamente varía la distancia entre dos placas paralelas	Desplazamiento, presión
Inductancia		
Detector de reluctancia	La reluctancia de un circuito magnético varía al cambiar la posición del núcleo de hierro de una bobina	Presión, desplazamiento, vibración, posición
Transformador diferencial	La tensión diferencial de dos devanados secundarios de un transformador, varía al mover el núcleo magnético por medio de una fuerza aplicada desde el exterior	Presión fuerza, vibración, desplazamiento, posición
Medidor de corriente parásita	La inductancia de una bobina se altera por la proximidad de una placa con corrientes parásitas inducidas	Desplazamiento, espesor.
Medidor de magnetostricción	Las propiedades magnéticas cambian por esfuerzos y presión	Fuerza, presión

Los requerimientos del sistema de medición de fuerza servirán para determinar cuál es el transductor adecuado para los propósitos de este proyecto y se mencionan a continuación:

1 El sistema debe tener al menos el mismo rango y resolución de los dinamómetros empleados actualmente en la práctica, es decir, un rango de 0 a 10 *N* con resolución de 0.1 *N*.
2 Debe ser compacto y de fácil adquisición o construcción.
3 Su manejo y mantenimiento deben ser sencillos.
4 Deberá ser robusto, para evitar una deformación plástica del material con que se construye, reduciendo errores en lecturas de deformación y por el efecto de histéresis mecánica.

La mayoría de los transductores presentados en la tabla requieren un espacio considerable para su implementación, a menos que sean instrumentos de alta integración y ocupen muy poco espacio, pero con el inherente aumento del costo de adquisición e implementación. Los sensores ópticos ofrecen alta sensibilidad y resolución, pero son delicados y no son muy prácticos para fines industriales.

Por otra parte, las galgas extensiométricas pueden adherirse al material de estudio, reduciendo en gran medida el espacio de instalación, siempre y cuando el material tenga superficie necesaria para la adhesión de las mismas.

Una ventaja que se presentó en la construcción del transductor, fue que al solicitar algunas cotizaciones de estos productos, se encontró una empresa mexicana [15], distribuidora de galgas extensiométricas del fabricante HBM [16], la cual obsequió a este proyecto tres tipos diferentes de galgas, con las cuales se implementaron varios prototipos de transductor.

A continuación se describe el proceso de diseño y construcción de los transductores con galgas.

3.1.2.2 Galga extensiométrica

Es una rejilla o malla metálica cubierta por ambas caras por laminillas aislantes muy delgadas, la malla metálica está hecha de un hilo conductor muy fino y resistente, por lo común, aleaciones como el constantán o nicromo. En la Figura 3.3 se muestra un tipo de galga y su construcción básica.

Figura 3.3 a) Galgas extensiométricas típicas (uniaxial), b) construcción esquemática de una galga uniaxial.

Esta laminilla es adherida a la superficie del material que se someterá a esfuerzos. Cuando el material es deformado por estos esfuerzos, se asume que se transmitirá la misma deformación a la galga por medio del adhesivo. La variación de resistencia eléctrica de la malla metálica es un indicativo de la deformación.

Para entender como trabajan los medidores de deformación, considérese un conductor de área transversal uniforme, A, de longitud L, hecho de un material con una resistividad eléctrica, ρ. Para este conductor, la resistencia, R, está dada por

$$R = \rho \frac{L}{A} \ [ohms] \tag{3.4}$$

Cuando se tensa o comprime la galga, los tres parámetros cambiarán y a través de la medición de la resistencia resultante se puede calcular la deformación del conductor y de manera indirecta, la deformación en el espécimen bajo prueba.

El cambio en resistencia de un medidor de deformación, por lo general, se representa en términos de un parámetro llamado factor de galga, GF. Cada fabricante especifica este factor que está dado por la siguiente relación:

$$GF = \frac{\Delta R / R}{\Delta L / L} \tag{3.5}$$

Donde ΔR y ΔL son el cambio en resistencia y longitud, respectivamente, cuando la galga se deforma. R y L son la resistencia y longitud de la galga en estado natural o de cero deformación.

El término $\frac{\Delta L}{L}$ en el denominador de la ecuación es la deformación, ε establecida en la ecuación 3.2, de tal forma que el factor de galga se puede expresar como:

$$GF = \frac{\Delta R / R}{\varepsilon} \tag{3.6}$$

Las galgas utilizadas para los transductores tienen los siguientes valores nominales:

resistencia nominal: 120.0 Ω ± 0.35%
factor de galga: 2.04 ± 0.5 %
sensibilidad transversal: -1.1%
material: constantán.

3.1.2.3 Puente de Wheatstone

La variación de la resistencia en una galga es muy pequeña respecto a su valor nominal al medirse sobre un material de prueba bajo esfuerzo [5].

Para lograr medir estas deformaciones, la galga se debe conectar a un circuito que sea sensible a las pequeñas variaciones de resistencia. Un circuito muy conocido para medir cambios en resistencia bajos es el puente de Wheatstone resistivo como el que se muestra en la Figura 3.4.

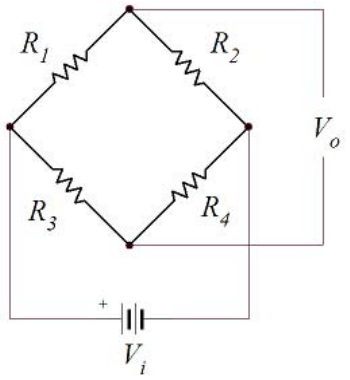


Figura 3.4 Puente de Wheatstone resistivo.

El análisis eléctrico del circuito se presenta en el Apéndice B, del cual se obtiene la ecuación 3.7 (ecuación B.7) que relaciona la tensión de salida, V_o, con las resistencias, R_i, de cada rama y la tensión, V_i, que alimenta al circuito.

$$V_o = V_i\left(\frac{R_1}{R_1 + R_2} - \frac{R_3}{R_3 + R_4}\right) \tag{3.7}$$

De esta ecuación, se puede apreciar que la tensión de salida será cero si las cuatro resistencias son iguales; en estas condiciones el puente se encuentra balanceado y si los resistores son las galgas adheridas a un material bajo prueba, entonces será el reflejo de un esfuerzo nulo.

Por lo general, puede existir un ligero desbalance al medir esfuerzos con galgas debido a diversos factores como: el autocalentamiento de los resistores, la temperatura ambiente, la disipación térmica del material de prueba, la calidad de la fuente de alimentación del puente y la instalación de las galgas, por mencionar algunos.

Algunas técnicas para eliminar estos desbalances, involucran la utilización de más de una galga, o configuraciones especiales en el alambrado del puente.

Se pueden construir transductores sencillos con una, dos o cuatro galgas activas; por esta característica, los circuitos empleados se llaman ¼ de puente, ½ puente y puente completo respectivamente. La siguiente ecuación que se obtiene en el Apéndice B (ecuación B.14), es la representación general de la tensión de salida para los tipos de puente mencionados considerando las galgas activas.

$$\Delta V_o = V_i \frac{GF}{4}\left(\varepsilon_1 - \varepsilon_2 + \varepsilon_4 - \varepsilon_3\right) \tag{3.8}$$

Para esta ecuación se considera que las galgas tienen resistencia y factor de galga GF nominales iguales, de tal forma que la tensión de salida del puente, es función de la deformación que experimenta una o hasta cuatro galgas.

Se puede observar en la ecuación 3.8 que para una o más galgas, las deformaciones en brazos opuestos se suman y en brazos adyacentes se restan. Esta característica se puede aprovechar para incrementar la salida del puente y para compensar *deformaciones aparentes* o desbalance del puente, manifestadas como ciertos cambios en resistencia que no son debidas a esfuerzos, por ejemplo, aumento de temperatura debido al efecto Joule en las galgas o deformaciones transversales del espécimen [5].

3.1.2.4 Construcción del transductor

Conociendo el comportamiento del puente de Wheatstone y su relación con las deformaciones en las galgas, el siguiente paso es construir el transductor. Los tipos más comunes de transductor son los de deformación axial, la barra en flexión o empotrada, los de tipo anillo y los de barra de torsión [5,7]. Existen otros tipos de

construcción geométrica que no se abordaron debido a que se optó por una configuración sencilla de barra en flexión o empotrada [9] y además, el objetivo es simplemente medir una fuerza en una sola dirección con exactitud.

Se construyeron tres transductores de flexión, uno en configuración ¼ de puente, otro en ½ puente y el tercero en puente completo. En la Figura 3.5 se muestra la configuración de cada uno y en el Apéndice B se obtienen las tensiones de salida correspondientes.

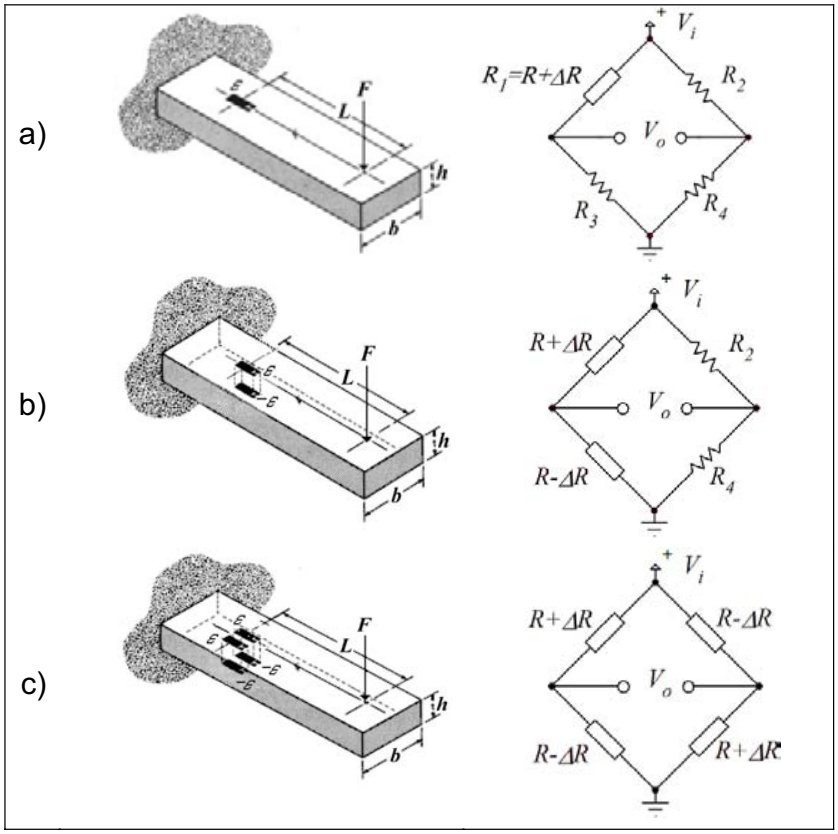


Figura 3.5 Transductor en a) ¼ de puente, b) ½ puente y c) puente completo.

El material empleado inicialmente para los dos primeros, fue lámina metálica de 70 x 50 *mm* de 0.4 *mm* de espesor. Este material no fue el idóneo pues la flexión se apreciaba a simple vista; con el paso del tiempo y uso constante, llegaría a fracturarse tanto el material, como la galga. Estos primeros transductores pueden servir para medir cargas pequeñas, especialmente si se utilizan como básculas de bajo gramaje.

El tercer transductor se construyó con una placa de acero inoxidable de 1.5 *mm* de espesor y 209.6 *mm* x 50.8 *mm*; se utilizó como dispositivo final por su robustez.

La construcción de los transductores se realizó siguiendo la mayoría de las recomendaciones del fabricante de las galgas [12]. Desafortunadamente no se contó con todos los aditamentos especiales como los químicos para limpieza del elemento de prueba, el adhesivo y las capas protectoras adhesivas para el elemento de prueba terminado.

Sin embargo, el proveedor recomienda utilizar para limpieza, alcohol isopropílico, y como adhesivo uno a base de cianoacrilato del tipo “kola loka”. Para proteger el espécimen, se optó por acondicionarlo a un gabinete plástico cubierto con papel aluminio, de forma que sea una caja de Faraday para evitar que el circuito capte ruido electromagnético.

Se muestran en la Figura 3.6 los elementos de prueba preliminares listos para ser puestos a prueba.

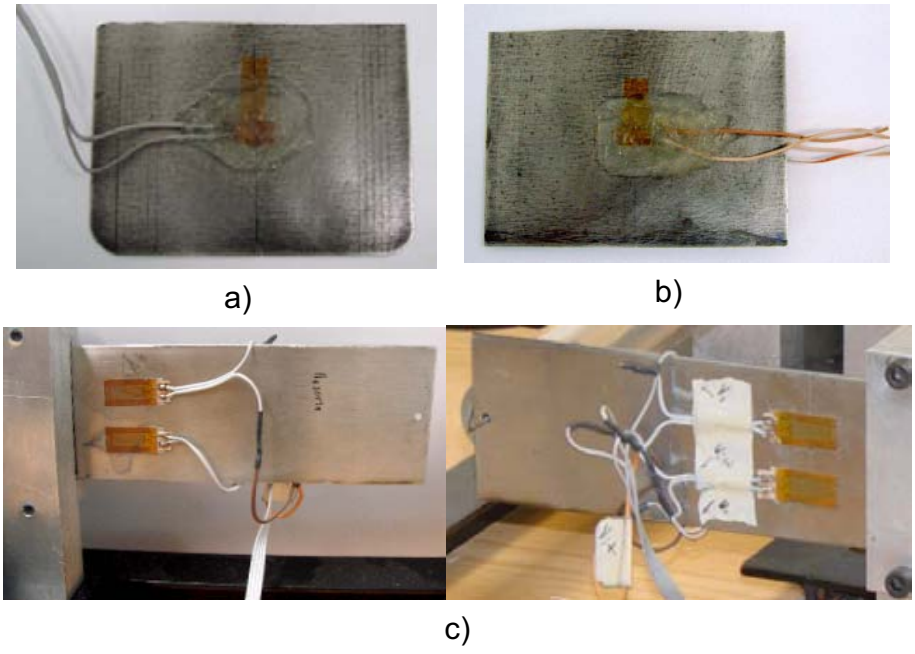


Figura 3.6 transductores construidos, a) ¼ de puente, b) ½ puente y c) puente completo, parte frontal y posterior.

3.1.2.5 Acondicionamiento de señal

En la hoja de datos de las galgas [18], se especifica como tensión máxima de alimentación del puente 13 *V*, para evitar daños a las mismas y sobrecalentamiento que afecte las mediciones.

Debido a que la variación de resistencia es pequeña, la salida del puente de Wheatstone es también pequeña. Por este motivo, es necesario utilizar un amplificador, y para poder registrar estas lecturas, es necesario uno que entregue una señal de 0 a 5 *V* a escala completa, con el propósito de procesar posteriormente los datos de forma digital.

Una opción eficiente y económica, en contraste con los costosos amplificadores comerciales de propósito especifico para este tipo de transductores, es usar un amplificador de instrumentación. Las opciones de amplificadores propuestos se muestran en la Figura 3.7 y son los siguientes: construir un amplificador de instrumentación empleando tres amplificadores operacionales, resistores y potenciómetros externos, o un amplificador de instrumentación integrado.

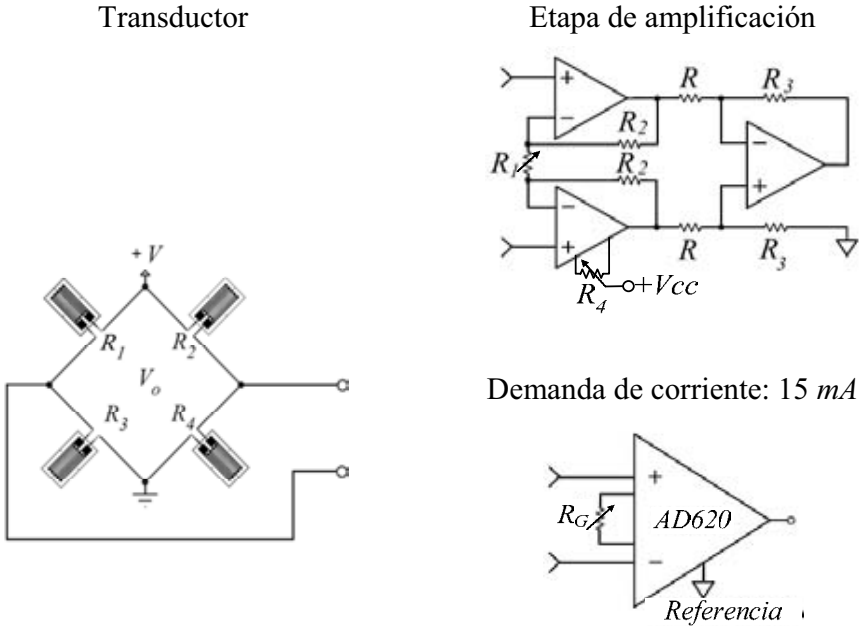


Figura 3.7 Propuestas para el amplificador

Las características principales de estos dispositivos se describen en seguida.

1 Amplificador construido basado en LF355

Los amplificadores operacionales son del tipo BIFET [14], es decir, tienen en sus pines de entrada transistores JFET, lo cual ofrece una alta impedancia de entrada, demandando de la fuente de señal una corriente mínima casi despreciable, para efectos de caída de tensión en el cableado hacia el transductor, reduciendo errores de lectura. Para su implementación, el fabricante recomienda el uso de resistores con una tolerancia muy pequeña. El resistor R_1 fija la ganancia, y el resistor R_4 ajusta el offset de la señal. Es necesaria una fuente de alimentación bipolar (no funciona como tipo "rail to rail"), lo que aumentaría el costo si se usa una fuente rectificada, o si se emplea baterías. También se considera en el diseño de este circuito, el espacio que puede utilizar en un circuito impreso.

2 Amplificador integrado AD620 [10]

Tiene las características de un amplificador de instrumentación construido, con la ventaja de que al ser integrado, cada amplificador operacional funciona en igualdad de condiciones térmicas, además de que la reducción de tamaño y la proximidad de los elementos reducen la capacidad distribuida y el retardo de tiempo.

Otra ventaja de utilizar el amplificador integrado, es un consumo de corriente menor en comparación con el construido si se llega a implementar como un instrumento que utilice baterías, ahorrando energía; además, el control de ganancia (que puede ajustarse desde 1 hasta 1000) es solamente a través de un potenciómetro R_G, y se puede referenciar la señal de salida a una tensión determinada a través de la terminal llamada *Referencia,* ya sea para ajuste de cero u offset para propósitos de funcionamiento con una sola fuente de energía, pues en este modo de operación, el amplificador no trabaja en todo el rango de tensión de polarización debido a que no es un amplificador de tipo "rail to rail".

Para poder trabajar en el rango de 0 a 5 *V* y con una sola fuente, se buscó la forma de generar una tensión negativa para polarizar ambos amplificadores. Un circuito muy útil es el empleado en la Figura 3.8 el cual es un convertidor de corriente directa positiva a negativa.

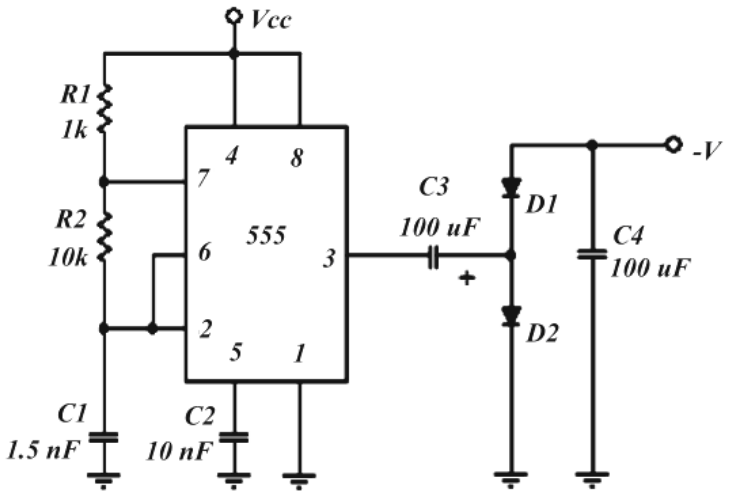


Figura 3.8 Convertidor de corriente directa positiva a negativa.

Este circuito es un oscilador astable que genera una señal cuadrada de alta frecuencia aproximadamente de 50 *kHz* y con un ciclo de trabajo de 52%; al pasar la señal por el condensador *C3*, esta se convierte en una señal de alterna, que después es rectificada y filtrada, obteniendo una tensión negativa aproximada de -5 *V* si la polarización de este circuito es de 8 *V* [19].

Al tener un ciclo de trabajo alto, el rizo de la señal se reduce al mínimo en el condensador de salida C4. Se estima que entrega una corriente máxima de 20 *mA*, con lo cual se asegura un buen funcionamiento tanto del AD620 que opera con 1.3 *mA* como del amplificador construido.

La ganancia del amplificador construido [14] se establece como:

$$G = \frac{R_3}{R}\left[\frac{2R_2}{R_1} + 1\right] \qquad 3.9$$

y para el amplificador integrado [10]:

$$G = \frac{49.4\,k\,\Omega}{R_G} + 1 \qquad 3.10$$

Una vez implementados los circuitos en una tableta de prototipos, se procedió a probar los amplificadores con los transductores de una y dos galgas, completando el puente con resistores variables de 20 vueltas, con el propósito de ajustar las resistencias a un valor próximo al de las galgas. En las primeras pruebas, el valor de carga máxima de 10 *N* se estableció para 5 *V* ajustando la ganancia. La Tabla 3.2 resume aspectos a destacar de las pruebas hechas con los amplificadores.

Tabla 3.2 Resumen de pruebas de los amplificadores.

Prueba del amplificador construido	Prueba del amplificador AD620
Al intentar obtener la señal para 10 *N* en forma repetida, se observó un ligero desbalance al cambiar a carga nula, por lo que se corregía ajustando el potenciómetro R_4, que permite ajustar el offset de la señal, pero al hacer este ajuste la ganancia también cambiaba, por lo que había que reajustarla con el potenciómetro R_1. Si se desea que el instrumento se calibre a distancia, el ajuste de estos dos potenciómetros es un problema más a resolver.	En su implementación, no se observó cambio significativo en la ganancia del amplificador (la cual se ajusta con una resistor variable de 20 vueltas de 5 *kΩ*), por lo que ésta no requiere de ajuste. Para establecer la condición de balance del puente, se elimina el posible *offset* de la señal, utilizando el pin $Referencia$, donde se puede conectar una tensión positiva o negativa, con el propósito de llevar al nivel deseado la salida del puente; en este caso, interesa que la tensión de salida sea de 0 *V* en el estado de carga nula. La forma en que se puede aplicar una tensión en este pin de forma controlada es con un resistor digital DS1869 [17]

En la hoja de especificaciones del fabricante del amplificador de instrumentación, se muestra una tabla comparativa de las fuentes de error y el error acumulado de cada circuito, presentando un error menor total el amplificador integrado de casi la mitad del construido [10].

El circuito que mostró un mejor comportamiento a largo plazo fue el del AD620. Durante las pruebas de los transductores, no se tuvo que volver a ajustar la ganancia,

además de que a partir de una fuente unipolar se generó una tensión negativa para su funcionamiento; este aspecto repercutirá en el diseño final, pues se ahorrará en la adquisición o construcción de una fuente bipolar.

Finalmente, por la facilidad de manejo e implementación, por la posibilidad de establecer el cero del instrumento de manera sencilla y por presentar un menor ruido en la señal amplificada, se determinó que el amplificador usado para el instrumento será el AD620.

El sistema de medición hasta este momento queda conformado por los siguientes elementos: transductor, amplificador, ajuste de offset, y fuente de energía; se muestra un diagrama de bloques de este sistema en la Figura 3.9. El ajuste de offset se realiza con pulsadores por medio del resistor variable digital DS1869, la implementación del ajuste en forma automática, se aborda en el siguiente capítulo.

Para medir la señal entregada por el amplificador, se utilizó un osciloscopio, y para la calibración se utilizó un dinamómetro de 0 a 12 *N* con resolución de 0.1 *N* y otro de 0 a 2 *N* con resolución de 0.01*N*.

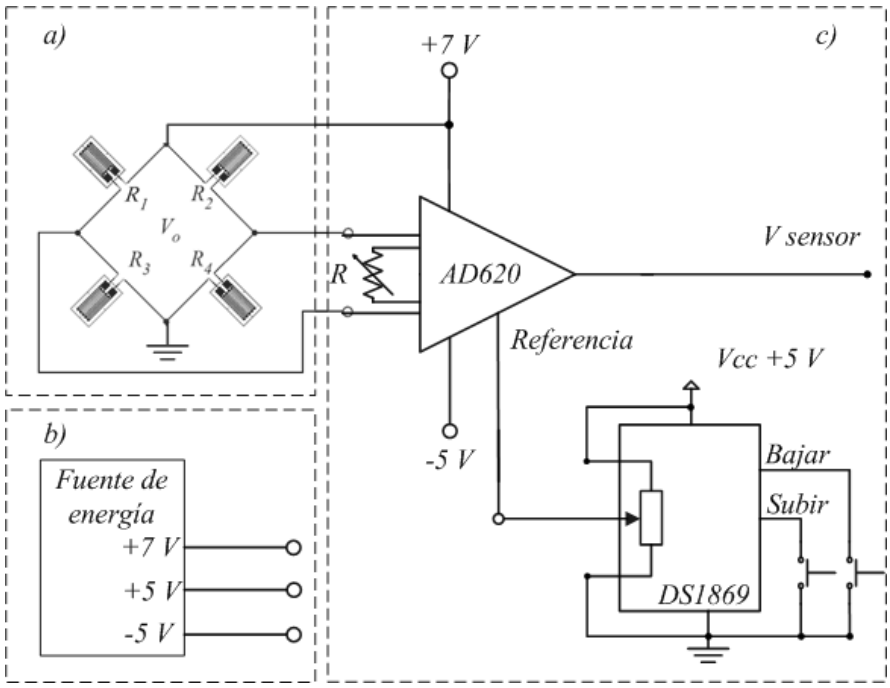


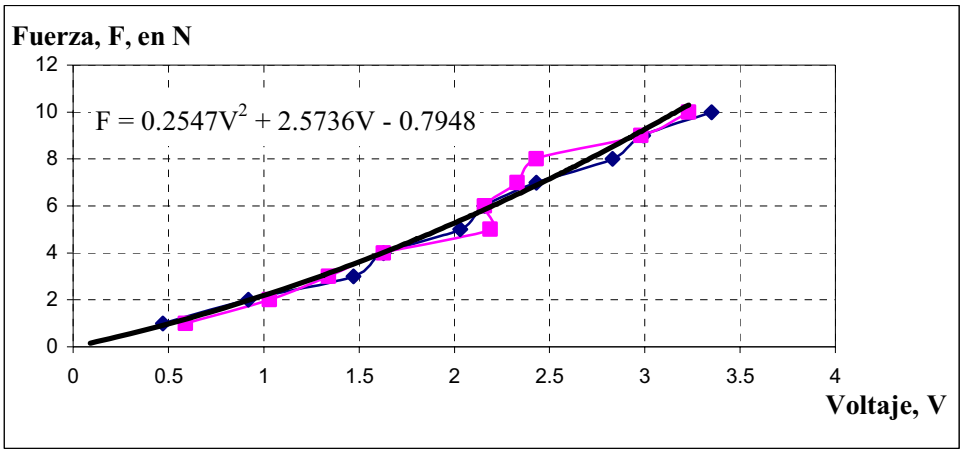
Figura 3.9 Circuito eléctrico del sensor: a) transductor, b) alimentación y c) amplificación y ajuste de Offset.

3.1.3 PRUEBA Y CARACTERIZACIÓN

En esta etapa se realizaron mediciones discretas de la señal de salida del instrumento de medición con los tres transductores, con el propósito de observar su comportamiento y linealidad con el paso del tiempo, aspecto importante recordando que la relación entre esfuerzo y deformación es lineal.

Existe una amplia teoría en torno al análisis de cada tipo de transductor y su posible no-linealidad y técnicas de linealización [11, 13]. Los dos primeros tipos de transductor son no lineales; se presentan las curvas características y la ecuación que relaciona las variables mostradas en la Figura 3.10. Por otra parte, la tercera configuración del transductor es lineal.

a)



b)

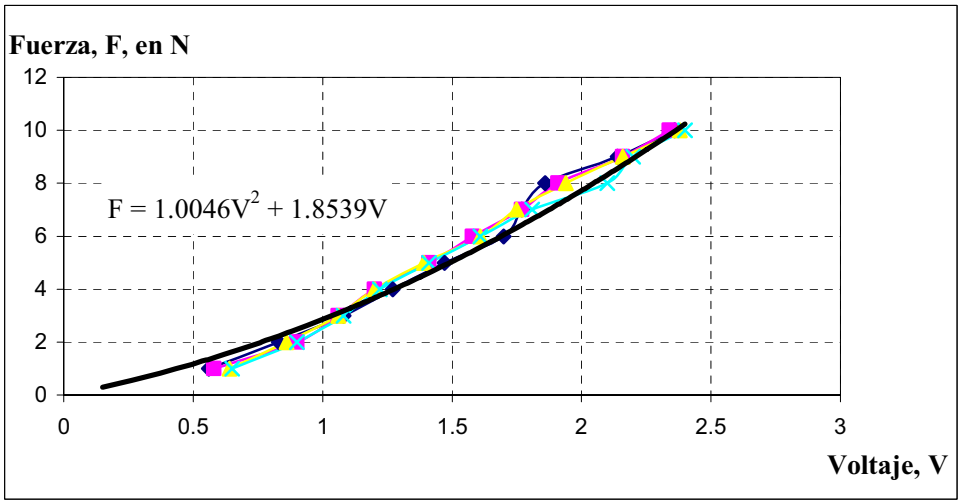


Figura 3.10 Curvas características a) ¼ de puente, b) ½ puente.

Una forma para especificar la tensión de salida del transductor de viga empotrada en puente completo, es utilizando la expresión B.25 del Apéndice B:

$$\frac{dV_o}{V_i} = GF\varepsilon$$

Antes se debe calcular la deformación máxima $\varepsilon_{máx}$, a la que se sometería el elemento de prueba, además de conocer su modulo de elasticidad E. Para hallar ε se utiliza la ecuación B.21 del Apéndice B.

$$\varepsilon_{máx} = \frac{6F_{máx} \cdot L}{Ebh^2}$$

Los parámetros a utilizar son los siguientes:

E del acero inoxidable = 180-200 x10^9 N/m^2
L = 107.95 mm = 0.10795 m
b = 50.8 mm = 0.0508 m
h = 1.5 mm = 0.0015 m
$F_{máx}$ = 10 N
GF= 2.04

Por lo tanto:

$$\varepsilon_{máx} = 0.000283 \ m/m \quad o \quad \varepsilon_{máx} = 283 \ \mu m/m$$

y la tensión de salida del sensor se especifica como 0.578 *mV/V*; este resultado se interpreta como 0.578 *mV* de salida del sensor por cada volt de alimentación del puente, por lo que, mientras más grande sea esta última tensión, más grande será la tensión de salida, hecho importante para incrementar la relación señal a ruido del instrumento.

Si se alimenta el puente con 10 *V*, la tensión máxima del sensor será de 5.78 *mV*, por lo que para tener un valor aproximado a 5 *V* para su tratamiento digital, es necesario que el amplificador tenga una ganancia un poco menor a 1000.

Para diseñar este tipo de instrumentos se puede seguir este método, sin embargo, también se puede hacer de una forma práctica estableciendo para la carga mínima un valor de 0 *V* ajustando el offset del amplificador, y para la carga máxima 5 *V*, ajustando la ganancia del mismo.

Esta última forma fue la que se llevó a cabo, debido a que se estuvo variando el rango de fuerza que se aplicaba al sensor al cambiar varias veces el resorte por la siguiente razón: al realizar pruebas del sensor con el resorte conectado, se observó que el primer resorte de prueba era muy rígido, pues al deformarlo la máxima distancia (250 *mm*), el móvil chocaba con el límite del la pista donde se desliza; por tal motivo se probaron varios resortes, hasta encontrar uno que pudiera desplazar al móvil en casi la totalidad de la pista sin chocar, al aplicar la deformación máxima.

Finalmente, al adecuar un resorte que cumpliera dicho propósito, se midió la fuerza máxima, obteniendo 2 *N*. Para la calibración y caracterización del sensor se utilizó un dinamómetro de 0 a 2 *N* con resolución de 0.01 *N*.

La Figura 3.11 muestra la curva característica y la ecuación que relaciona las dos variables para el transductor en puente completo.

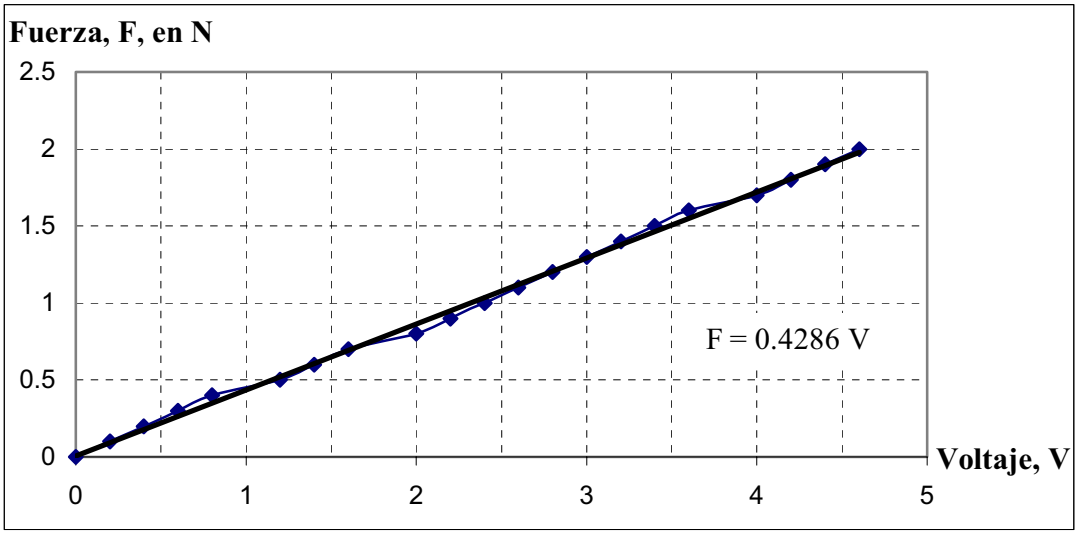


Figura 3.11 Curva característica del sensor de puente completo.

En los dos primeros casos, la fuerza máxima aplicada fue de 10 *N* y la ganancia se ajustó para este rango. Es decir, la salida del sensor es de 5 *V* a escala completa. Existe la posibilidad de variación en las mediciones debido a calentamiento de los resistores que completan el puente.

Por otra parte, el tercer transductor al tener en los cuatro brazos galgas activas, la temperatura de cada una se compensa con su opuesta en el puente, ya que están sobre el mismo material que además por ser de acero, tiene la capacidad de disipar sin problema el calor.

Además, al utilizar cuatro galgas, se tiene una tensión de salida del puente mayor, reduciendo efectos de ruido eléctrico que afecten las mediciones. En la medida de las posibilidades, se recomienda un transductor de cuatro galgas.

La Tabla 3.3 contiene las especificaciones del sensor de fuerza analógico.

Tabla 3.3 Características eléctricas del sensor de fuerza.

Parámetro	Valor
Carga máxima:	2 *N*
Tensión de salida del puente:	1.1733 *mV/V*
Alimentación del puente:	7.92 *V*
Resistencia compuesta del puente:	120 *Ω*
Corriente de consumo del puente sin carga:	66 *mA*
Potencia disipada en cada galga:	130 *mW*
Control de ajuste de cero	

En los capítulos correspondientes al controlador e interfaz del sistema, se abordará la digitalización de la señal y el manejo de este sensor con la interfaz web.

3.2 SENSOR DE POSICIÓN LINEAL

Una de las especificaciones establecidas al principio del proyecto, fue el aprovechar el material disponible en el laboratorio, y aún más al tratarse de sensores e interfaz electrónica y con el usuario, ya que este material es de gran utilidad al tratarse de un sistema que ha mostrado gran funcionalidad.

Para medir la posición del móvil, se tomaron en cuenta varias opciones, entre ellas, sensores infrarrojos, sonar basado en un microcontrolador PIC, incluso formas para medir indirectamente la distancia, por ejemplo, a través de la distancia que recorrería el bloque de posicionamiento grueso del móvil desde el lugar que alcanzó a deslizarse éste, hasta la posición inicial.

Algunos de estos dispositivos se listan en la Tabla 3.4.

Tabla 3.4 Dispositivos para medir distancia.

Dispositivo	Rango	Resolución
Sensor de distancia Ultrasónico Maxbotix LV-EZ1	0 - 6.45 *m*	25.4 *mm*
Sensor de distancia ultrasónico SRF05	0 - 4 *m*	10 *mm*
Sensor de distancia analógico GP2Y0A21YK0F	0.1 - 0.8 *m*	no especificado
Sensor ultrasónico Pasco CI6742	0.15 - 8 *m*	1 *mm*

La opción fue naturalmente implementar en el proyecto el sensor ultrasónico con que se contaba, y además, de las opciones disponibles, tiene las mejores prestaciones; pero no sólo fue necesario el hardware, sino además el software de adquisición de datos, el cual es una aplicación para PC con derechos de autor, por lo que no sería posible presentarla en la página de Internet de la práctica, ni manejarla de manera oculta al usuario tal y como se encuentra esta aplicación.

Sin embargo, dentro del mismo proyecto, Creación de un laboratorio remoto accedido por internet, del cual esta práctica forma parte, se desarrolló un trabajo de tesis [8] del cual se derivó una aplicación de consola para manejar el sonar y guardar los datos en un archivo de texto, además de mostrar un ejemplo de cómo integrar este tipo de aplicaciones a una página web para manejar el dispositivo desde Internet.

La forma de usar la aplicación que controla el sonar se ejecuta de la siguiente manera:

1 Se instala el controlador de la interfaz electrónica siguiendo los pasos de la guía incluida.
2 Desde la consola de comandos, se abre la carpeta *Sonar* donde se encuentra el ejecutable “pasco.exe” instalado en el primer paso, por ejemplo *C:\Sonar>.*
3 Se ejecuta la aplicación “pasco.exe” incluyendo como argumentos el periodo de muestreo seguido del tiempo del evento, ambos en milisegundos, por ejemplo, *C:\Sonar\pasco.exe 50 1000.*
   Con esta ejecución del programa, el sonar tomará lecturas cada 50 *ms* durante 1 segundo.
4 En la misma carpeta donde se encuentra la aplicación, se genera un archivo llamado lecturas, la cual contiene la hora en que se tomaron las mediciones y la distancia medida.

Después de probar esta aplicación con el sonar, se encontró que con un periodo de muestreo de 25 *ms*, se tiene alta repetibilidad en las lecturas, en comparación con un tiempo más corto para medir la misma distancia, pues se obtienen errores de lectura. Este error es más acentuado cuando se realizan mediciones dinámicas y con periodo de muestreo menor a 25 *ms*.

Por otra parte, al revisar los datos, se estimó que la aplicación los recibe de la interfaz electrónica aproximadamente cada 15 *ms*.

Para obtener datos confiables se establecerá un periodo de muestreo de 25 *ms* y el tiempo del evento se determinará al realizar las pruebas finales del sistema al realizar el experimento de Trabajo y Energía.

La forma en que la aplicación de consola se integrará a la página web, y la forma en que se recuperan los datos desde el archivo de texto generado, se abordarán en el capítulo 5.

CAPÍTULO 4
DISEÑO DEL CONTROLADOR Y LA INTERFAZ ELECTRÓNICA

En este capítulo se aborda el diseño de la interfaz electrónica y la programación del controlador que gestiona los actuadores y sensores dispuestos en el sistema, de los cuales, ya se describieron su funciones en el Capitulo 2.

4.1 REQUERIMIENTOS DE LA INTERFAZ

Para realizar la operación del sistema, se cuenta con los siguientes dispositivos:

- un motor de pulsos, del cual se controlará el sentido y posición angular
- un motor de corriente continua, controlando solamente el sentido de giro
- un electroimán de corriente continua
- un sensor de fuerza, el cual entrega una señal de 0 a 5 *V*, por lo que se medirá esta señal analógica y deberá almacenarse en algún dispositivo para su posterior análisis.

Para establecer la condición de inicio del sistema (Home) es necesario:

- detectar la presencia del carro en los extremos de la superficie deslizante
- detectar el inicio de carrera del carro que transporta el electroimán.

Además, el control de esta interfaz se realiza por medio de una PC, por lo que se debe implementar un protocolo de comunicación entre estos dispositivos.

4.2 DISPOSITIVOS DE CONTROL

En el trabajo de tesis *"Práctica de trabajo y energía; diseño mecánico"*, a partir del cual se crea el sistema que se pretende controlar, se proponen como opciones dos dispositivos de control muy utilizados y de amplio uso en la industria: el microcontrolador PIC y los PLC's. En la Tabla 4.1 se presentan sus características, ventajas y desventajas.

Tabla 4.1 Dispositivos de control

Dispositivo	Características	Ventajas	Desventajas
 **Microcontrolador PIC**	Circuito integrado programable que contiene todos los componentes necesarios para controlar un proceso determinado. Excelente para control de sistemas mecánicos. Programa de control fácilmente modificable. Posibilidad de realizar control distribuido utilizando varios PIC o realizar conexión serial con una PC.	Bajo costo de adquisición e implementación. Existe gran cantidad de información en Internet y además de la proporcionada por el fabricante. Soporta programación en diversos lenguajes. Se puede dedicar parte de la memoria a guardar datos y programas auxiliares.	Sensible a efectos electromagné-ticos. No soporta sobrevoltaje ni sobrecarga.
 **PLC**	Equipo electrónico programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente de tipo industrial, procesos secuenciales. Interfaz de programación específico, dependiendo del fabricante.	Pequeño, buen control en procesos secuenciales. Economía de mantenimiento al no tener con contactos móviles. Existen programadores con interfaz a PC haciendo más fácil y rápida la reprogramación.	Alto costo. Limitación en los procesos internos que puede realizar. Necesidad de diverso cableado.

El dispositivo elegido fue el microcontrolador PIC, pues las características que ofrece para la implementación de la interfaz, además de las mencionadas en la tabla, incluyen la modificación del diseño de acuerdo a los requerimientos de los dispositivos como: circuitos integrados, sensores y actuadores, ocupando un reducido espacio; como ventaja adicional, se tiene conocimiento del ambiente de programación para PIC tanto en lenguaje ensamblador como en lenguaje C.

La programación en un lenguaje de alto nivel como C presenta varias ventajas, por ejemplo:

1 el desarrollo de un programa es más fácil
2 el mantenimiento, modificación y actualización del programa es más fácil
3 es más sencillo probar un programa
4 es más amigable al usuario y menos propenso a errores
5 es más fácil documentar un programa.

Sin embargo, los lenguajes de alto nivel también presentan desventajas. Por ejemplo, el tamaño del código ensamblador se incrementa, lo cual se traduce en un programa mayor, ocupando más espacio de memoria, en comparación a escribir desde el inicio todo el código en ensamblador de manera más eficiente; además, el programa tarda un poco más en ejecutarse.

4.2.1 MICROCONTROLADOR PIC18F452

Pertenece a la gama alta de microcontroladores de Microchip, conserva la arquitectura Harvard y un procesador RISC, con muchas prestaciones adicionales a los PIC de gama media, como memoria FLASH de programa de 32 *kbytes*, 1536 bytes de memoria RAM y 256 bytes de memoria EEPROM, una pila de 31 localidades de 21 bits, puede operar a una frecuencia de trabajo hasta de 40 *MHz*, permite experimentar con la velocidad de operación de la mayoría de los módulos integrados para obtener el mejor desempeño, cuenta con 5 puertos de propósito general y configurables para usar con algún módulo integrado, entre ellos, un módulo de conversión analógica digital de 10 bits, el cual cuenta con 8 entradas analógicas multiplexadas, un módulo de comunicación serial USART y comunicación paralela PSP. Se puede encontrar mayor información en la hoja de datos del dispositivo [32].

Algunos factores para la elección de este dispositivo se mencionan a continuación: la programación en C es prácticamente la misma tanto para PICs de gama media como de gama alta, sólo hay que tener en cuenta que las direcciones de los registros son diferentes (se pueden consultar en la hoja de datos); presenta gran funcionalidad en el desarrollo de nuevos y complejos proyectos, gracias a su capacidad de memoria y la velocidad de ejecución del programa, comparado con otros dispositivos de su tipo.

4.2.2 HERRAMIENTAS DE PROGRAMACIÓN

La programación del microcontrolador se realiza usando el lenguaje C con el compilador CCS 4.057. La interfaz de este compilador es muy sencilla (Figura 4.1) y cuenta con diversas barras de herramientas, algunas de las cuales son muy útiles si se cuenta con hardware adicional, que se puede adquirir con el desarrollador del compilador [31].

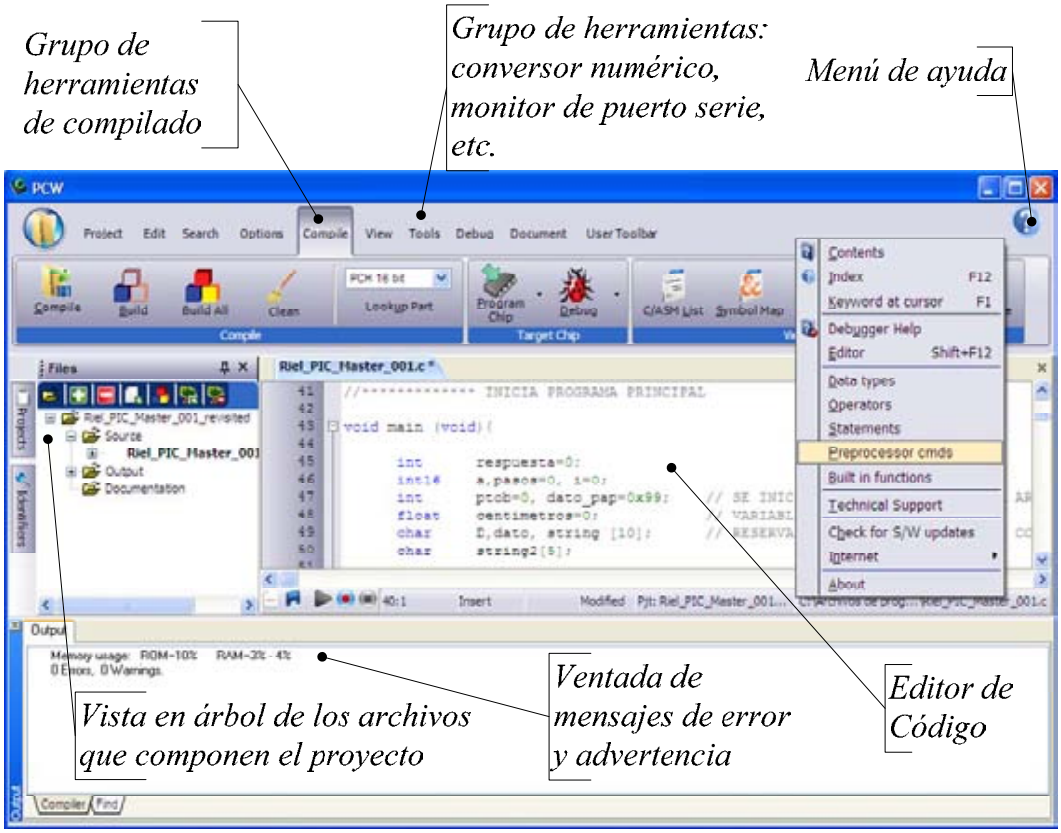


Figura 4.1 Ambiente del Compilador CCS, C para PIC.

Adicionalmente al compilador CCS, se utiliza MPLAB IDE del fabricante de los PIC, Microchip [30], ya que se utiliza una herramienta de simulación y además ventanas adicionales que contienen información de todos los registros del PIC, con el propósito de visualizar el espacio de memoria libre, y asignar, sin errores de sobreescritura en memoria de programa, localidades disponibles para guardar los datos del sensor de fuerza (Figura 4.2) en un PIC secundario o esclavo. El compilador MPLAB trabaja con la misma herramienta de compilado que el CCS C, haciéndolo una potente herramienta para generación de código en C y permitiendo una depuración más interactiva que la realizada sólo con el compilador CCS.

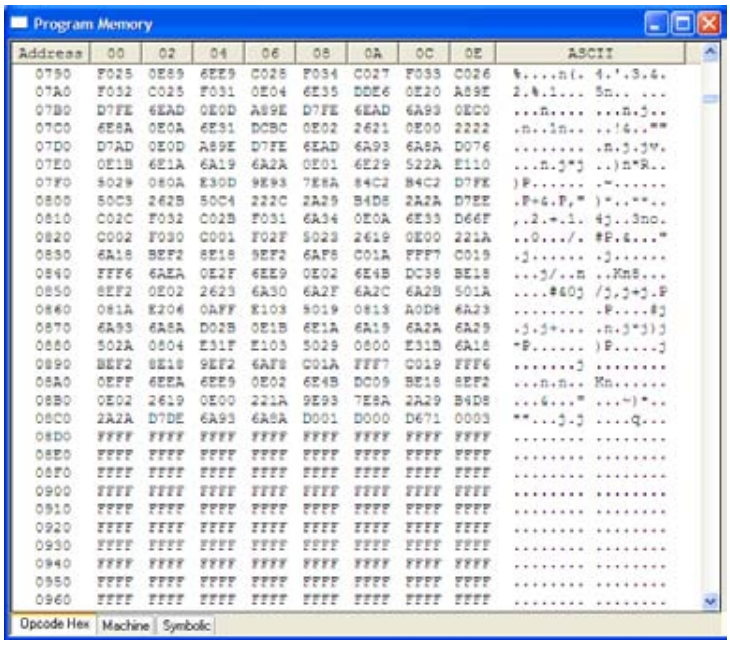


Figura 4.2 Ventana en MPLAB de la memoria de programa.

Finalmente, otra herramienta que se utilizó para el desarrollo del proyecto fue un *bootloader* de código abierto [29]; esta herramienta es un programa pequeño que se graba en el PIC, facilitando su reprogramación sin necesidad de hacerlo con el programador específico cada vez; basta con usar uno para grabar el bootloader al PIC y los nuevos programas que se quieran guardar en el dispositivo, será por medio de una interfaz gráfica para PC (Figura 4.3) y a través de un circuito de comunicación serial RS232, que deberá tener el sistema; cuenta además con un monitor de puerto serial con el que se puede establecer comunicación con el PIC si se configura el

protocolo de comunicación, que puede ser de gran ayuda para la depuración y detección de errores del programa.

Figura 4.3 Interfaz gráfica para el uso del bootloader.

4.2.3 CIRCUITOS DE INTERFAZ

Las señales de entrada y salida que maneja el microcontrolador PIC, deben manejarse en nivel TTL (lógica transistor-transistor) que en términos generales, una señal alta (de valor 1 lógico) equivale a +5 volts y una señal baja (0 lógico) cero volts.[1]

Los microcontroladores PIC son incapaces de drenar una corriente mayor a 20 *mA* por cada pin y por tal motivo son necesarias etapas de potencia que se controlen con señales TTL, y que puedan drenar la corriente necesaria a los elementos que lo necesitan. De igual forma, las señales entrantes, por ejemplo de sensores, deben acondicionarse a esos niveles de tensión para evitar dañar al PIC.

1

Para una señal de entrada alta, el rango de voltaje es de 2 a 5 volts.
Para una señal de entrada baja, de 0 a 0.8 volts.
Para una señal de salida alta, de 2.4 a 5 volts.
Para una señal de salida baja, de 0 a 0.4 volts.

4.2.3.1 Control del motor de pulsos

Cuando el electroimán es energizado, se está en condiciones de iniciar la deformación del resorte una determinada distancia; para tener control directo de ésta sin necesidad de un control en lazo cerrado, se utiliza un motor de pulsos, que en conjunto con el husillo y tuerca, se caracterizan para establecer una relación directa entre pulsos (pasos) y desplazamiento.

El motor utilizado para el desplazamiento fino es un motor de pulsos bipolar, con los siguientes datos nominales:

- Voltaje nominal: 2.2 *V*
- Corriente nominal: 2.2 *A*
- Grados por paso: 1.8°.

El motor seleccionado es del tipo hibrido, y está conformado por dos devanados en el estator con un cierto numero de polos (ocho en para este motor), mientras que el rotor consta de un imán cilíndrico magnetizado axialmente, dispuesto entre dos piezas de hierro dulce laminado que tienen varios dientes, ligeramente mayor al número de dientes totales en los polos del estator como se puede apreciar en la Figura 4.4, con lo que se obtienen ángulos de paso muy pequeños.

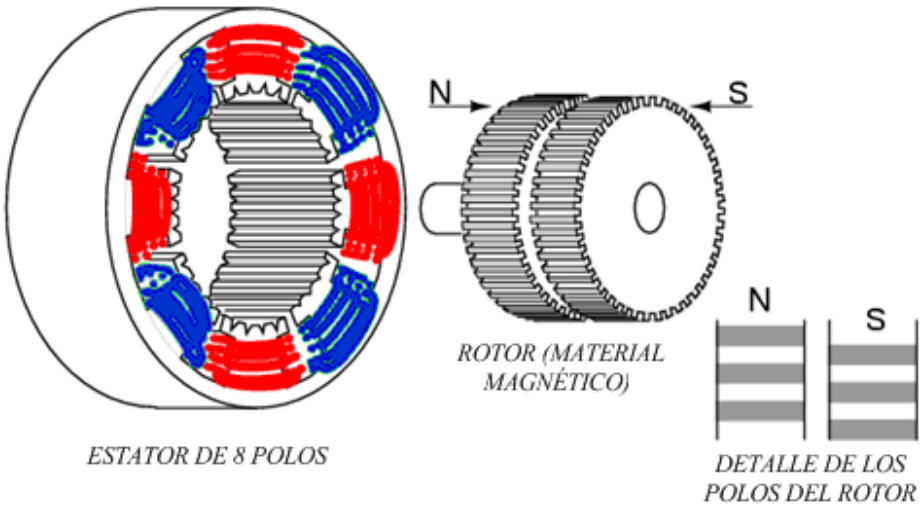


Figura 4.4 Motor de pulsos bipolar (detalle de construcción).

Cuando se polarizan los devanados, los dientes de los polos del estator atraen a los dientes del rotor con la polaridad contraria más cercanos, de manera que polos y dientes se alinean. Así, mediante la conmutación de la corriente al siguiente devanado y a un ligero defase de los dientes, se logra que gire el rotor.

Para generar el movimiento del rotor se debe aplicar una secuencia específica de pulsos en los devanados del motor. Para ilustrar la secuencia de control, en la Figura 4.5 se muestra la construcción básica del motor de pulsos bipolar y su principio de funcionamiento.

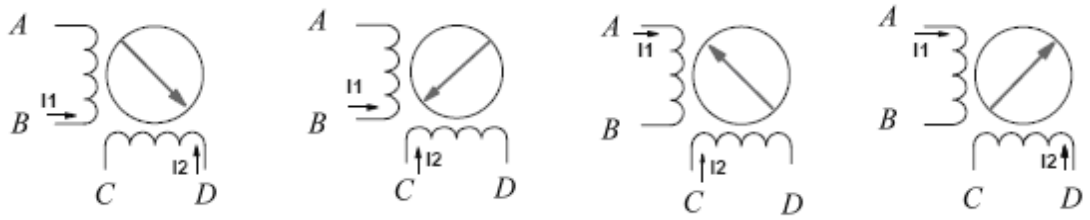


Figura 4.5 Principio de funcionamiento.

La punta de la flecha del rotor indica un polo norte magnético, la flecha en los devanados muestra el sentido de la corriente, en otras palabras, muestra la polaridad del devanado. De acuerdo a la polarización de los devanados será la posición del rotor y deberá de seguir la secuencia descrita en la figura y que se resume en la Tabla 4.2.

Tabla 4.2 Secuencia para movimiento.

Paso	A	C	B	D
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0
4	1	0	0	1

Un circuito que permite el cambio de sentido de la corriente es el puente H, y se necesita uno por cada devanado. El circuito integrado L298 contiene dos puentes H, y puede drenar hasta 2 *A* [33], los necesarios para que el motor gire.

Para controlar el motor de pulsos bastan cuatro bits que deberán seguir la secuencia de control de la Tabla 4.2 si el movimiento es, por ejemplo, en sentido horario, o bien, si es en sentido antihorario con la secuencia de pasos inversa 4,3,2,1; se emplea además un bit adicional que habilita o deshabilita el puente H.

La etapa de potencia para el control del motor de pulsos tiene la configuración mostrada en la Figura 4.6.

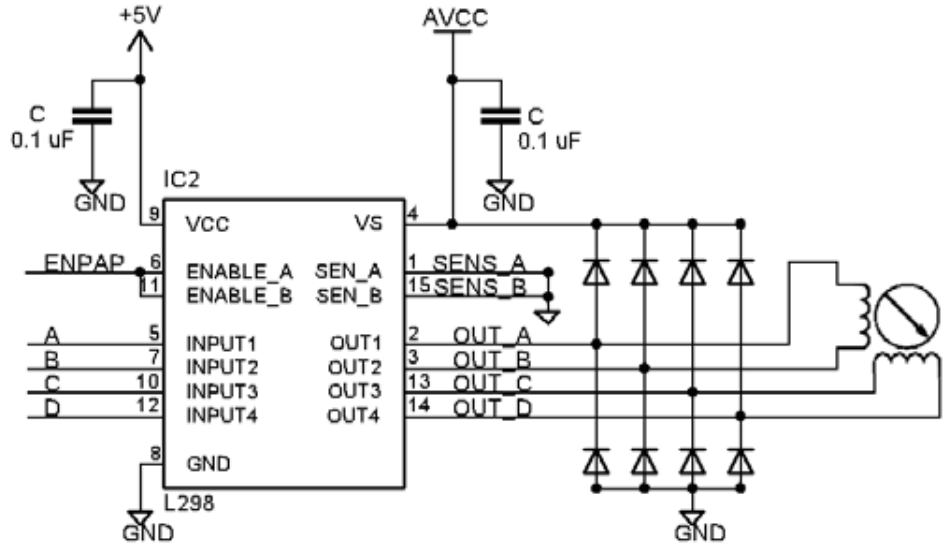


Figura 4.6 Etapa de potencia para motor de pulsos.

4.2.3.2 Control del motor de corriente directa y del electroimán

El carro que recupera el móvil es activado por el sistema compuesto por un motor de corriente directa y un mecanismo banda-polea; la detección de fin de carrera de este carro se hace por medio de sensores ópticos reflectivos, y están ubicados de tal forma que se puede asegurar que el electroimán quedará en contacto con el móvil.

Para lograr que el motor gire en ambas direcciones se empleó un puente H. El voltaje de alimentación del motor es de 10 *V*, para establecer este valor de operación se hicieron pruebas con distintos valores, observando el movimiento del carro sobre el riel; con valores menores el movimiento es muy lento y con valores mayores a 10 *V*, es muy brusco.

Al alimentar el electroimán con el mismo voltaje con el que se alimenta este motor, se observó que se sobrecalentaba y que consumía 1 *A* de corriente, por lo que se propuso bajar este valor por medio de un regulador.

Para establecer el valor del voltaje regulado se energizó el electroimán, estableciendo un nivel de voltaje para el cual la fuerza magnética de este es ligeramente mayor a la fuerza máxima que ejerce el resorte. El valor se fijó en 3 *V*, con un consumo de corriente de 100 *mA*.

El circuito final se muestra en la Figura 4.7**,** el circuito queda con una línea libre etiquetada con *buzz* para implementar en un futuro una señal auditiva.

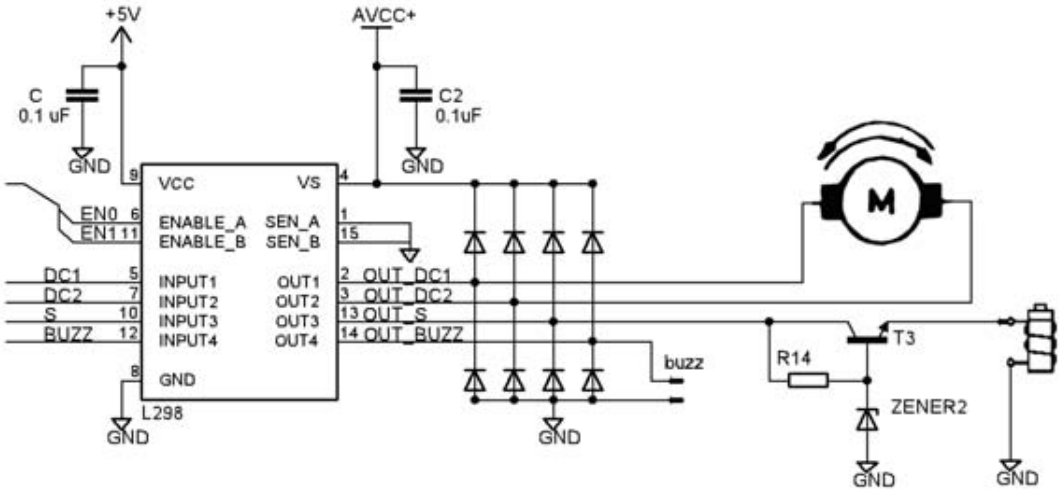


Figura 4.7 Etapa de potencia del motor y del electroimán de corriente directa.

4.2.3.3 Sensor de inicio de carrera magnético

Para establecer la posición de inicio del sistema, el carro de posición fina debe situarse al inicio del husillo; para detectar esta posición, un sensor de inicio de carrera *reed switch* o interruptor magnético se activa; con este sensor se evita contacto físico entre elementos y se reduce la posibilidad de falla por falso contacto o por daño mecánico, los reed switch alcanzan fácilmente 10^8 ciclos de funcionamiento. El aspecto y conexión del interruptor se muestra en la Figura 4.8

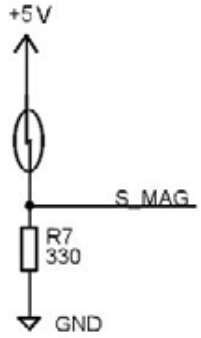


Figura 4.8 Conexión del sensor de inicio de carrera.

4.2.3.4 Sensores de carrera óptico reflectivos

Los sensores de final de carrera detectan la presencia del carro en los extremos de la pista; se implementan dos, uno en cada extremo. El dispositivo se basa en un sensor óptico reflectivo CNY70 el cual integra un diodo emisor infrarrojo y un fototransistor; al estar colocados en la misma dirección, el fototransistor se satura al recibir el haz de luz infrarroja reflejada sobre un objeto. La distancia entre el objeto y el sensor debe ser de 5 a 10 *mm* y está en función de la intensidad de corriente del diodo y del colector.

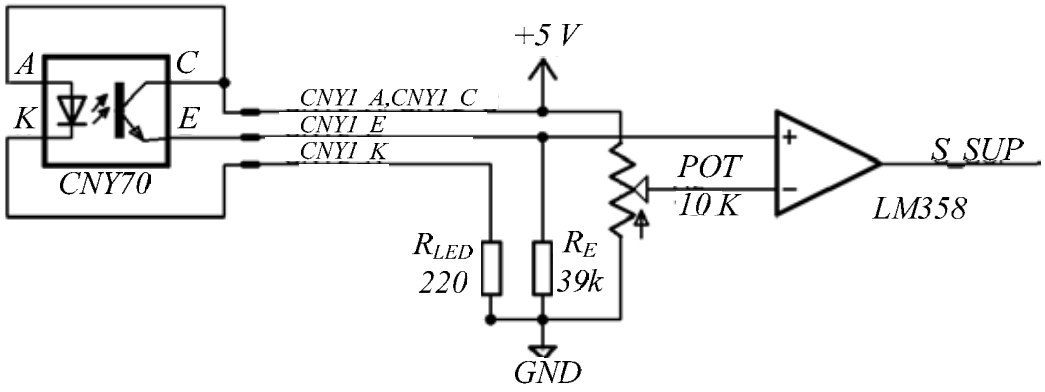


Figura 4.9 Circuito detector con comparador.

En la Figura 4.9 se muestra la conexión para detectar un objeto de color blanco, y con ajuste del umbral de detección por medio de un comparador de voltaje implementado con un amplificador operacional dual LM358.

Los valores de las resistencias del circuito se calcularon como sigue.

Resistencia de polarizado del led Infrarrojo
Tensión de operación del led: 1.25 *V*, corriente de operación = 20 *mA*
Vcc = 5 *V*

$$R_{LED} = \frac{Vcc - 1.25}{20 \times 10^{-3}}$$
$$R_{LED} = 187.5\,\Omega$$

Pueden usarse valores comerciales de 180 o 220 *Ω*, y se utiliza este último como medida de seguridad.

Resistencia de polarización del fototransistor
Para estimar la cantidad de corriente que drena el colector, se utilizan las tablas de la hoja de datos del CNY70, si se detecta el objeto a una distancia de 6 *mm* la corriente del colector se aproxima a 0.15 *mA*, por lo que la resistencia de emisor es

$$R_E = \frac{5}{0.15 \times 10^{-3}}$$
$$R_E = 33.333\,k\Omega$$

Se fija el valor de la resistencia en 39 *kΩ.*

4.2.3.5 Convertidores de nivel RS-232

Para poder establecer comunicación con una PC (el servidor de Internet), las vías disponibles son: el puerto paralelo, el puerto serial y puerto USB [8]; por facilidad de implementación y por tener experiencia en su implementación y uso, se empleó el puerto serial (transmisión serial) con el protocolo de comunicación RS-232 que usa solamente dos líneas, una para transmitir y otra para recibir, por lo que es posible una comunicación full duplex. El puerto paralelo usa al menos 16 líneas para su implementación, además de que se puede estar usando como puerto de impresora, y con respecto al puerto USB, por el momento no se ha abordado la programación que es un poco más extensa para su aplicación con una interfaz gráfica de PC.

El estándar RS-232 incluye aspectos como:

- el protocolo usado para la transmisión de datos
- los niveles de tensión que se usan para las líneas de transmisión/recepción
- los conectores para este tipo de comunicación.

Una descripción detallada del protocolo de comunicación la podemos encontrar en la referencia [34], el compilador CCS C tiene integrada la función de transmitir datos de forma serial con la instrucción *printf("Mensaje")*, con opciones configurables de velocidad de comunicación, número de bits a transmitir y otras opciones.

Los umbrales de tensión con los cuales trabajan las líneas son de +3 *V* a +15 *V* para un "0 lógico" (RS-232) y de -3 *V* a -15 *V* para un "1 lógico " (RS-232). Estos valores son incompatibles con los niveles TTL del PIC, por lo que es necesaria una interfaz para convertirlos. Un circuito muy empleado para convertir los niveles de tensión de TTL a RS-232 y viceversa es un chip transceptor (transmisor/receptor) MAX232 del fabricante Maxim, sin embargo en nuevos diseños se emplea el MAX233 [33], que necesita menos componentes externos. La interfaz mostrada en la Figura 4.10 se basa en el circuito Integrado MAX233, además se muestra el conector estándar (9 pines tipo D o DB-9)

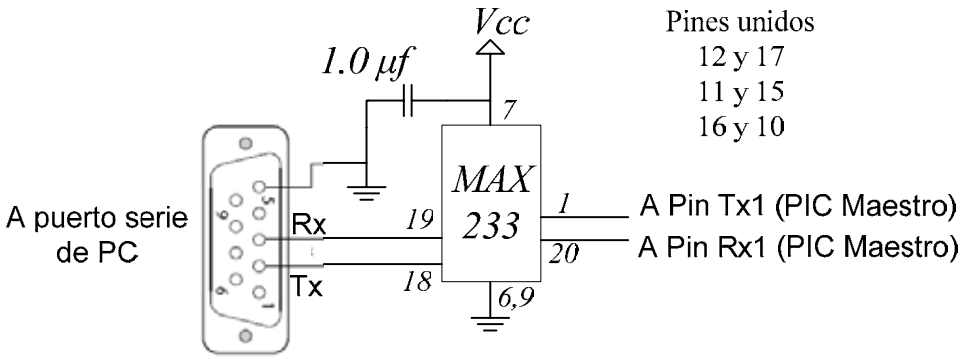


Figura 4.10 Convertidor de nivel usando el C.I. MAX233.

Para la comunicación PIC maestro-esclavo, también se usó la comunicación serial. Al implementar la comunicación directamente de PIC a PIC (nivel TTL), se presentaron problemas, probablemente causados por la longitud de los cables que se conectaban entre los dispositivos; al implementar el convertidor de nivel MAX233 se solucionaron estos errores; esta fue la razón de emplear protocolo RS-232 entre los dos dispositivos. La configuración final de los dos circuitos se muestra en la Figura 4.11.

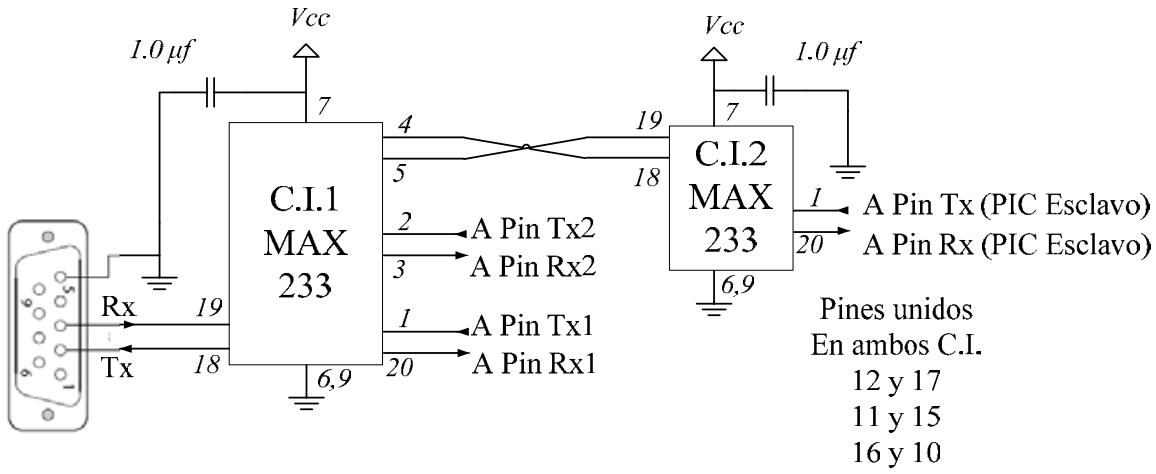


Figura 4.11 Interfaz de comunicación PC/PIC Maestro/Esclavo.

4.2.3.6 Descripción de conexiones del PIC maestro

Los actuadores y sensores descritos anteriormente son gestionados por el microcontrolador maestro, las conexiones entrada/salida (I/O) se describen en la Tabla 4.3 y en la Figura 4.12 se muestra el diagrama en bloques de las conexiones.

Tabla 4.3 Conexión I/O con los circuitos de interfaz.

Ubicación	Alias	Tipo	Descripción
PORTC<3>	*TX2*	O	Puerto serial 2: PIC maestro-PIC esclavo
PORTC<4>	*RX2*	I	
PORTC<6>	*TX1*	O	Puerto serial 1: PC (servidor)-PIC maestro
PORTC<7>	*RX1*	I	
PORTD<0:3>	*A, B, C, D*	O	Devanados del motor de pulsos
PORTD<4>	*DC1*	O	Terminales de conexión del motor de CD
PORTD<5>	*DC2*	O	
PORTD<6>	*S*	O	Terminal activa del electroimán
PORTD<7>	*BUZZ*	O	Alarma auditiva (no implementada)
PORTB<0>	*S_MAG*	I	Sensor magnético de inicio de carrera
PORTB<1>	*S_INF*	I	Sensor óptico de inicio de carrera
PORTB<2>	*S_SUP*	I	Sensor óptico de fin de carrera
PORTB<4>	*TEST-SYS*	I	Botón de prueba manual del sistema
PORTB<5>	*EN1*	O	Habilitación del driver del electroimán
PORTB<6>	*EN0*	O	Habilitación del driver del motor de CD
PORTB<7>	*ENPAP*	O	Habilitación del driver del motor de pulsos

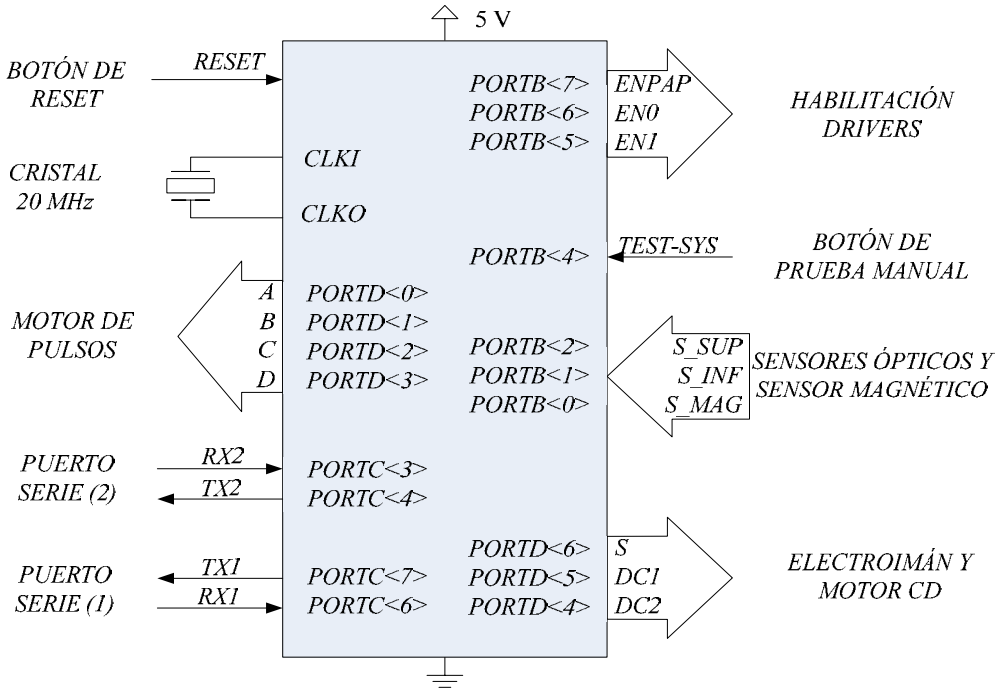


Figura 4.12 Asignación de los pines del PIC con los circuitos interfaz.

4.2.3.7 Programa de control de la interfaz PIC maestro-PC

La programación del microcontrolador para identificar un carácter válido como instrucción, se realiza esperando el dato entrante por el puerto serie del PIC y preguntando si es una instrucción. Las sentencias de control que lleva a cabo este trabajo es el proceso *switch*, el cual evalúa la *expresión* que es el carácter recibido por puerto serie, éste se compara con las *constantes* definidas, y si coincide la expresión y la constante, se realizarán las sentencias asociadas. El comando *break* provoca la salida del switch. Si ninguna *constante* corresponde a la *expresión*, se ejecuta el caso por omisión o *default* que incluso puede ser opcional. La Figura 4.13 muestra un ejemplo de evaluación de un dato entrante *C* y la ejecución de sentencias.

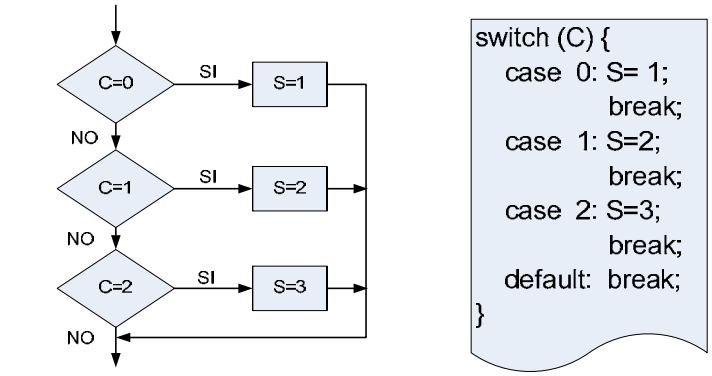


Figura 4.13 Evaluación de un dato entrante (instrucción) con sentencia Switch

Otro proceso de control empleado son las declaraciones *if-else* que pueden ser sencillos o anidados, siendo opcional la declaración e*lse*, la Figura 4.14 muestra una forma de emplear este proceso.

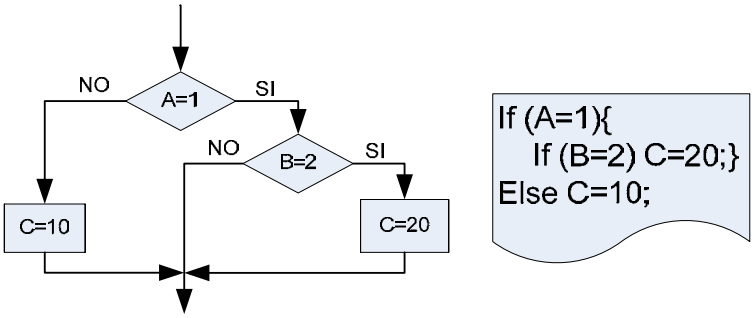


Figura 4.14 Sentencia de control if-else

Otra sección dentro del programa del PIC maestro a destacar es una e*structura* llamada *PORTD_BITS*; previamente establecida la dirección física del puerto D, se definen cuatro bits para el control del motor de pulsos, los restantes bits se dedican para controlar el motor de C.D. y el electroimán.

```
struct PORTD_BITS
{
BYTE motor_pap: 4; // bits 0-3  PINES 0 AL 3 DEL PUERTO D        -MOTOR DE PULSOS
BYTE carro:       2;  // bit 4,5   PIN 4 Y 5 PARA CONTROL DEL     -MOTOR DE CD
BYTE Solenoide: 1;  // bit 6      PIN QUE ACTIVA EL                -SOLENOIDE
BYTE un_bit:      1;  // bit 7   ---PIN SIN USAR---
};
```

El elemento *motor_pap* de la estructura, funciona como un puerto de 4 bits que se puede manipular individualmente para obtener la secuencia de control del motor del pulsos, que ya se revisó en la sección 4.3.4.1.

Los elementos *carro* y *solenoide* también pueden cambiar de estado, uno a la vez. El elemento carro corresponde al control de giro del motor de CD de rotación continua y el elemento solenoide, al estado (encendido/apagado) del electroimán. Las líneas para habilitar los circuitos de potencia son EN0 para el motor y EN1 para el electroimán y el motor de CD. El estado de estos elementos se describe a continuación en la Tabla 4.4

Tabla 4.4 Valores de control del motor de CD y electroimán

EN0	Carro	Valor
1	Avanza izquierda	0x01
1	Avanza derecha	0x02
1	Parado	0x00
0	Desconectado, giro libre	X
EN1	Solenoide	Valor
1	Encendido	1
1	Apagado	0
0	Desconectado	X

Otro aspecto a destacar dentro de la programación en C para PIC son las *directivas de pre-procesamiento*, aunque no son parte de la sintaxis de C se aceptan, debido a su funcionalidad. Estas directivas se ejecutan separadas de la compilación del programa principal y de hecho, se ejecutan antes de la misma. Los preprocesadores más comunes son *#define* y *#include.*

La directiva *#include <18F452.h>* permite que el compilador utilice el archivo *18F452.h* dentro de la compilación del programa (no se guarda dentro del programa que se carga al PIC), el archivo contiene las direcciones de cada registro del PIC asignadas a algún identificador, por ejemplo, este archivo contiene una línea donde se define la localidad de memoria 31744 con el identificador PIN_A0.

#define PIN_A0 31744

Así, cada terminal del PIC, al igual que otros parámetros de control de los módulos integrados, tiene un identificador para facilitar al programador en el desarrollo del programa.

La directiva #fuses indica la configuración general del PIC al momento de ser grabado el programa. Las opciones disponibles dependen de cada PIC y está relacionado con los módulos que se utilizan y la forma en que están configurados. Se encuentra más información sobre estas opciones en la hoja de datos del dispositivo.

La configuración para el PIC maestro es la siguiente.

#fuses HS, NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP

Las librerías de retardos, entrada/salida y comunicaciones del compilador CCS C requieren de una directiva de preprocesamiento para la definición de parámetros e inicialización de estos sistemas. De estas directivas, *#use delay* y *#use rs232* se utilizan dentro de la compilación del programa. La primera sirve para establecer la frecuencia de operación del PIC y la segunda para definir los parámetros de un puerto serie que utilizará protocolo RS-232.

En el código del PIC maestro, se establece la frecuencia de operación en 20 *MHz* y se configuran dos puertos serie (aunque el dispositivo tiene un solo módulo en hardware), generando uno adicional por software, asignando los pines de I/O de la siguiente manera

```
#use delay(clock = 20M)
#use rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=COM_A,)
#use rs232(baud=9600, parity=N, xmit=PIN_C3, rcv=PIN_C4, stream=COM_B, FORCE_SW)
```

A continuación se describe por medio de diagramas de flujo en las Figuras 4.15 y 4.16 el programa de control de la interfaz por medio de la PC (servidor). El PIC responde de acuerdo a la instrucción recibida y envía de regreso un carácter de reconocimiento como control de flujo de los datos enviados y recibidos. La lista de comandos del servidor con la interfaz electrónica, se trata más a fondo en el siguiente capítulo, al abordar la interacción de ésta con la interfaz web de usuario.

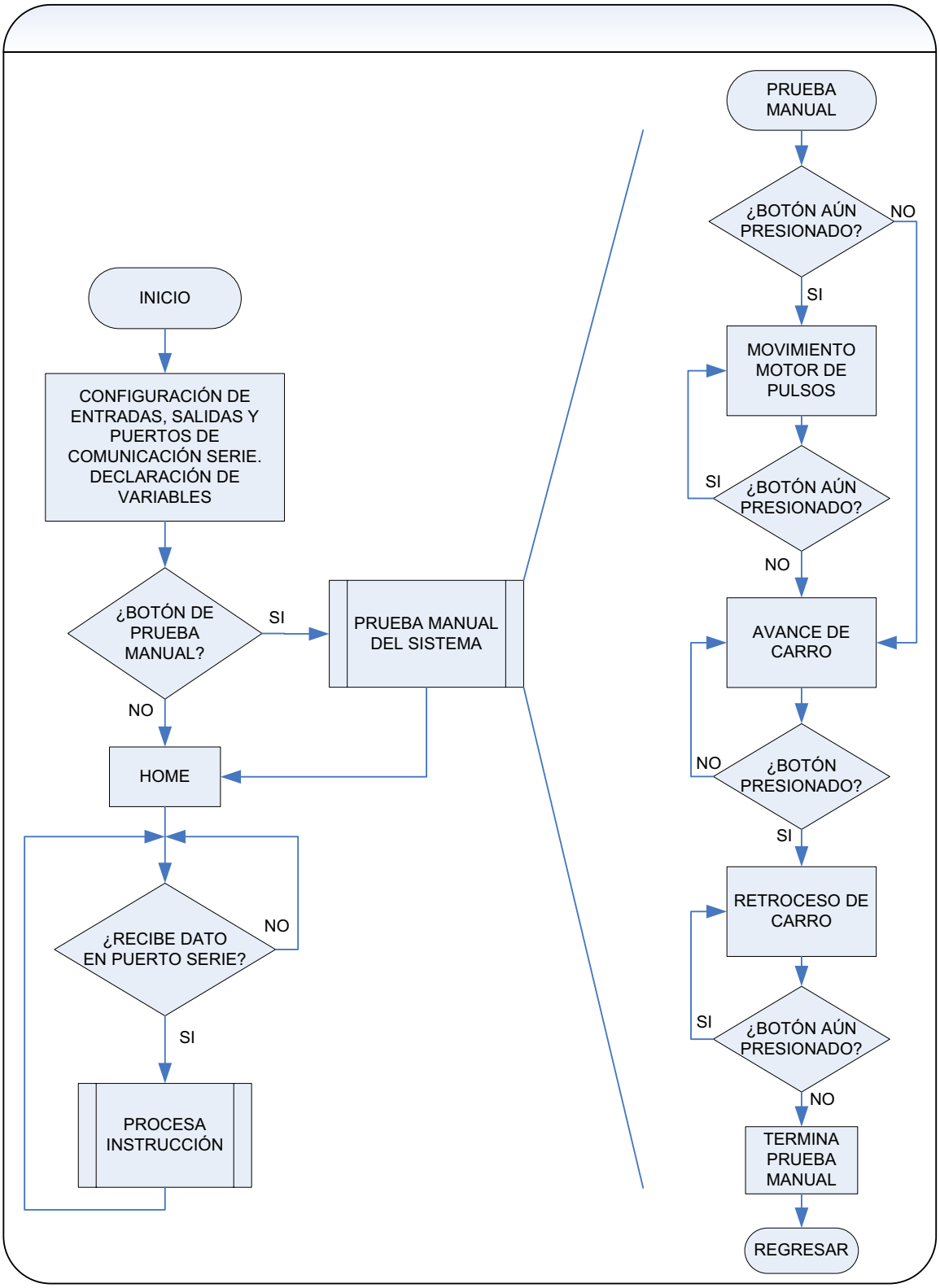


Figura 4.15 Diagrama de flujo del programa del PIC maestro

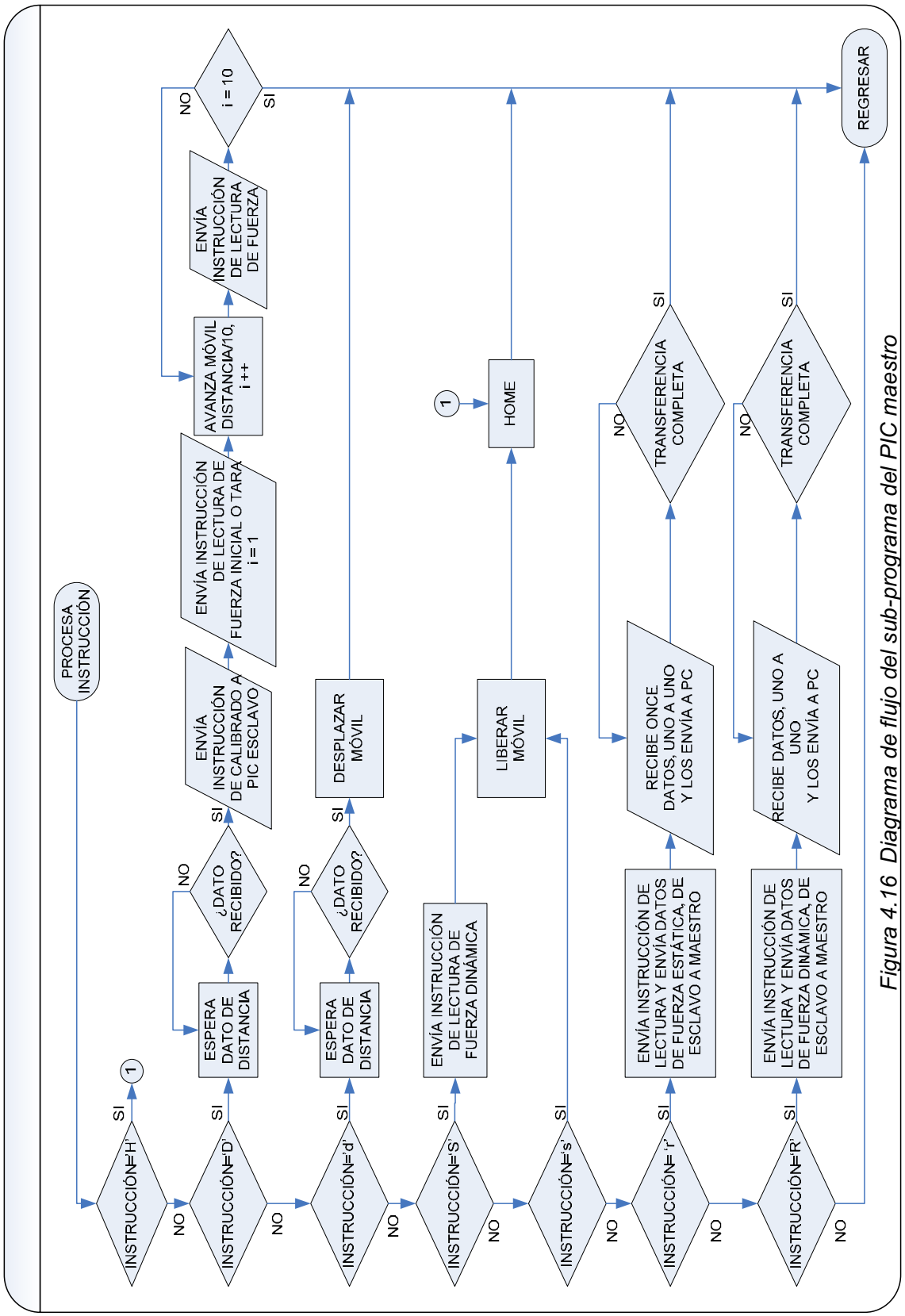


Figura 4.16 Diagrama de flujo del sub-programa del PIC maestro

4.2.4 INTERFAZ DEL SENSOR DE FUERZA

Para la medición de la señal analógica que entrega el sensor de fuerza, se implementó otro microcontrolador específicamente para funcionar como medidor de fuerza digital, ya que si se utiliza un solo PIC para control y para medición, es posible que la ejecución del programa tenga mayor latencia en los procesos internos (recuérdese que la programación se hace en C).

El proceso de lectura de la señal analógica se realizó por medio del convertidor analógico digital que tiene integrado el PIC. Este módulo obtiene un número que se relaciona con la proporción de la señal de entrada a la de referencia del módulo. El número de bits N que se emplea para la representación de la señal es de 10, es decir, se pueden obtener desde 0 hasta 2^{10}-1 o 1023 representaciones, el rango de tensión V_{REF} en el que opera el módulo es de 5 *V*, por lo tanto la resolución de cada bit es

$$resolución = \frac{V_{REF}}{2^N - 1}$$
$$resolución = 4.8\ mV.$$

De acuerdo con las especificaciones del sensor de fuerza analógico, la escala total del dispositivo es de 2 N. Si el modulo del convertidor A/D opera con un voltaje de 5 *V*, en primera instancia se puede afirmar que la resolución del sensor analógico es de (2 *N*)/(5 *V*), es decir 0.4 *N/V*. Haciendo la conversión necesaria para unidades de *mV*, se obtiene la resolución de 0.4 *mN/mV*. Tomando en cuenta la resolución del convertidor A/D de 4.8 *mV* por cada bit, el medidor digital tiene una resolución de

$$resolución = \left(0.4\ \frac{mN}{mV}\right)\left(4.8\ mV\right)$$
$$resolución \approx 2\ mN.$$

4.2.4.1 Operación del sensor de fuerza

La señal analógica proveniente del transductor, ingresa al PIC por una terminal del puerto A configurado como entrada analógica para su uso con el convertidor A/D, se estima una frecuencia de muestreo de 800 *Hz* ya que este proceso se realiza por medio de interrupciones. La Figura 4.17 muestra el proceso de adquisición y

almacenamiento, la cual se realiza usando interrupción de terminación de conversión analógica-digital en el microcontrolador PIC.

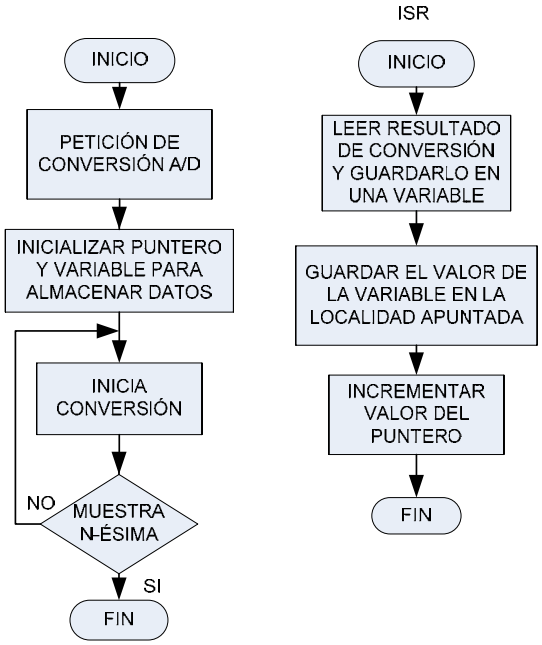


Figura 4.17 Proceso de lectura y almacenamiento de datos del sensor fuerza.

Al medir una señal estática para la caracterización del resorte, y una señal decreciente en el tiempo, al medir la disminución de la fuerza en éste, al soltar el bloque, no es necesario el uso de filtros ni de una frecuencia de muestreo específica. Para obtener el mayor número de muestras en el segundo evento, el proceso de conversión A/D y almacenado de datos en memoria, se realiza usando interrupciones.

La operación del sensor de fuerza en el sistema se describe a continuación.

Los eventos de medición de fuerza se llevan a cabo durante la caracterización del resorte al deformarlo una distancia total *D,* que se divide en 10 partes iguales, en cada tramo se lleva a cabo la medición de la fuerza ejercida por el resorte, incluida la posición de cero deformación.

Para tal efecto, el PIC maestro realiza la medición indirecta de la distancia contando el número de pasos que gira el motor de pulsos, y al avanzar cada tramo, se envía

una señal al PIC esclavo (medidor de fuerza) para realizar una serie de lecturas y obtener su promedio, para después almacenarlas en memoria.

Una vez obtenidas las 11 lecturas, al recibir una instrucción de nueva lectura, los datos se sobrescribirán; para controlar el flujo de datos, se espera un carácter de reconocimiento o terminación de los procesos.

Un segundo proceso se lleva a cabo al realizar la medición de la dinámica de la fuerza, es decir, se realiza su registro desde el instante de máxima deformación hasta deformación nula cuando se libera el móvil o masa para que deslice por la pista, esto con la finalidad de ofrecer información adicional, que no se puede obtener de la forma tradicional en que se realiza la práctica.

Estos procesos (y un tercero adicional que consiste en el control de ajuste de cero) se representan en dos leds para indicar visualmente cuál de estos se encuentra realizando durante el experimento, con la finalidad de hacer ajustes y depuración del programa del PIC esclavo y/o maestro y/o de la interfaz de usuario dentro de las pruebas que se realizan en conjunto con el dispositivo de experimentación

4.2.4.2 Control de ajuste de cero

Un tercer proceso que se realiza en el medidor de fuerza y que es transparente al usuario (si se desea), es la puesta a cero del sensor, la cual se realiza aprovechando la terminal de *referencia* del amplificador AD620. Es necesario mencionar que para realizar este ajuste de nivel, se detectó la forma en que el transductor puede entregar una tensión ligeramente negativa y sin carga, con la finalidad de sumar a esta señal, la referencia positiva de la misma cantidad y así poder establecer el cero del instrumento, ya que debido a la manipulación de la placa (transductor), ésta puede no encontrarse totalmente plana.

La aplicación de la tensión a la terminal referencia es por medio de un potenciómetro digital DS1869 [17] configurado con control dual de resistencia (aumentar y disminuir); estos controles son activados al llevar a tierra la terminal correspondiente al menos por un tiempo de 10 *ms* para cambiar un nivel, para después dejar la terminal en condición de flotante. El dispositivo ofrece 64 niveles o valores de resistencia (5 *kΩ* máximo) operando de 3 a 5 *V*.

Para permitir la condición flotante de las terminales de control, se emplean transistores BC548, cada uno es excitado en su base por una línea conectada a una terminal del PIC configurada como salida, por medio de un resistor limitador de corriente. El circuito de ajuste de cero se muestra en la Figura 4.18

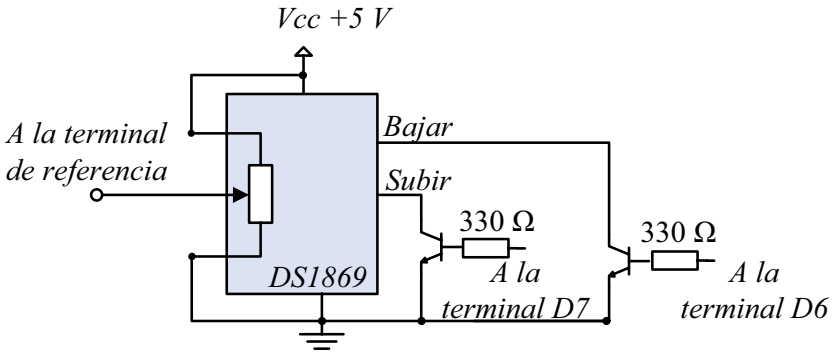


Figura 4.18 Circuito de ajuste de cero digital.

4.2.4.3 Descripción de conexiones del PIC esclavo

El sensor digital de fuerza se controla de forma semiautomática por el PIC esclavo, para realizar las acciones de tomar y enviar lecturas y ajustar el cero del instrumento, necesita la orden proveniente del PIC maestro. Las conexiones entrada/salida (I/O) se describen en la Tabla 4.5 y en la Figura 4.19 se muestra el diagrama de bloques de las conexiones.

Tabla 4.5 Conexiones I/O del PIC esclavo.

Ubicación	Alias	Tipo	Descripción
PORTA<0>	*SENS*	I	Señal analógica del sensor de fuerza
PORTB<7>	*LED1*	O	Monitores de estado del PIC
PORTB<6>	*LED2*	O	
PORTD<7>	*U_REF*	O	Controles subir/bajar nivel de cero
PORTD<6>	*D_REF*	O	
PORTC<7>	*RX*	I	Puerto serie 2: PIC maestro-PIC esclavo
PORTC<6>	*TX*	O	

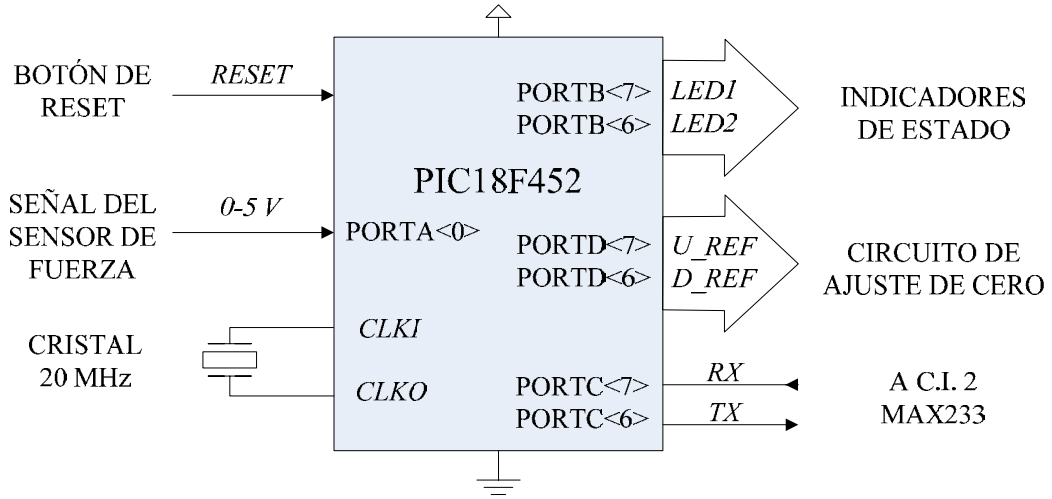


Figura 4.19 Asignación de las terminales del PIC esclavo.

4.2.4.4 Programa de control de la interfaz PIC maestro-PIC esclavo

De igual forma que como se hizo con el PIC maestro, el programa del PIC esclavo se presenta a continuación en un diagrama de flujo en la Figura 4.20. El PIC esclavo recibe del maestro un carácter correspondiente a una instrucción específica.

Las instrucciones disponibles se describen en la Tabla 4.6 que contiene la lista de comandos entre los PIC maestro-esclavo, cada instrucción se ejecuta por el esclavo si el dato o comando enviado por puerto serie por el maestro es el indicado.

Tabla 4.6 Comandos e Instrucciones usados para microcontrolador esclavo

COMANDO	INSTRUCCIÓN
D	RECOLECCIÓN DE DATOS DEL SENSOR, DINÁMICA DE LA FUERZA
C	CALIBRADO DEL SENSOR (PUESTA A CERO)
r	ENVÍO DE DATOS DE CARACTERIZACIÓN
R	ENVÍO DE DATOS DE DINÁMICA DE LA FUERZA
I	RECOLECCIÓN DE DATOS DEL SENSOR, CARACTERIZACIÓN DEL RESORTE
B	BORRADO DE DATOS (SÓLO EN DEPURACIÓN DEL PROGRAMA)

De igual forma que en el PIC maestro, la evaluación de los datos o comandos entrantes por el puerto serie, se realiza con la sentencia de control *switch.*

La lectura de la señal analógica del sensor de fuerza se realiza empleando el módulo de conversión analógica-digital del PIC, la forma de emplear este módulo comienza utilizando en la cabecera del programa el preprocesador *#device* el cual controla el número de bits que se leen por la función *read_adc(),* resultado de la conversión A/D; en este caso, se leen los 10 bits que ofrece el dispositivo

Los preprocesadores empleados en el programa del PIC esclavo se muestran a continuación.

```
#include    <18f452.h>
#device    adc=10
#fuses      HS,  NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP
#use        delay(clock = 20000000)
#use        rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7)
```

Los procesos del PIC esclavo se monitorean visualmente a través de dos leds con la finalidad de que sean una ayuda para depurar el programa, los estados se definen en la Tabla 4.7

Tabla 4.7 Monitor de proceso en ejecución.

LED1	*LED2*	ESTADO
0	0	INACTIVO
0	1	CALIBRADO DE SENSOR
1	0	TOMA DE LECTURA
1	1	ENVÍO DE DATOS

En las Figuras 4.21 a 4.25 se presentan los diagramas de conexiones de los dispositivos de interfaz. Se empleó el software de diseño gráfico de esquemas electrónicos ISIS versión 7 del sistema PROTEUS.

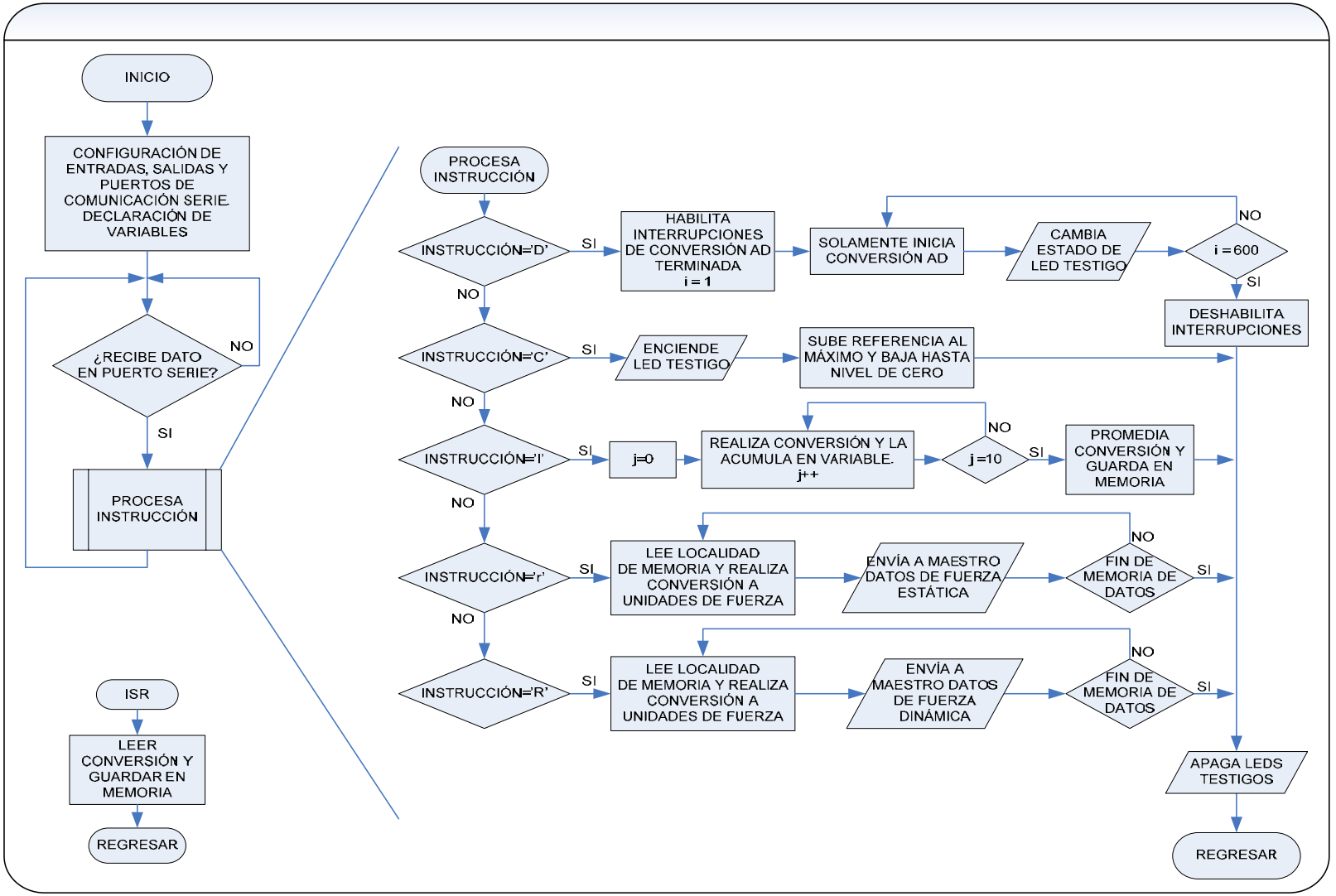


Figura 4.20 Diagrama de flujo del programa del PIC esclavo (para sensor de fuerza)

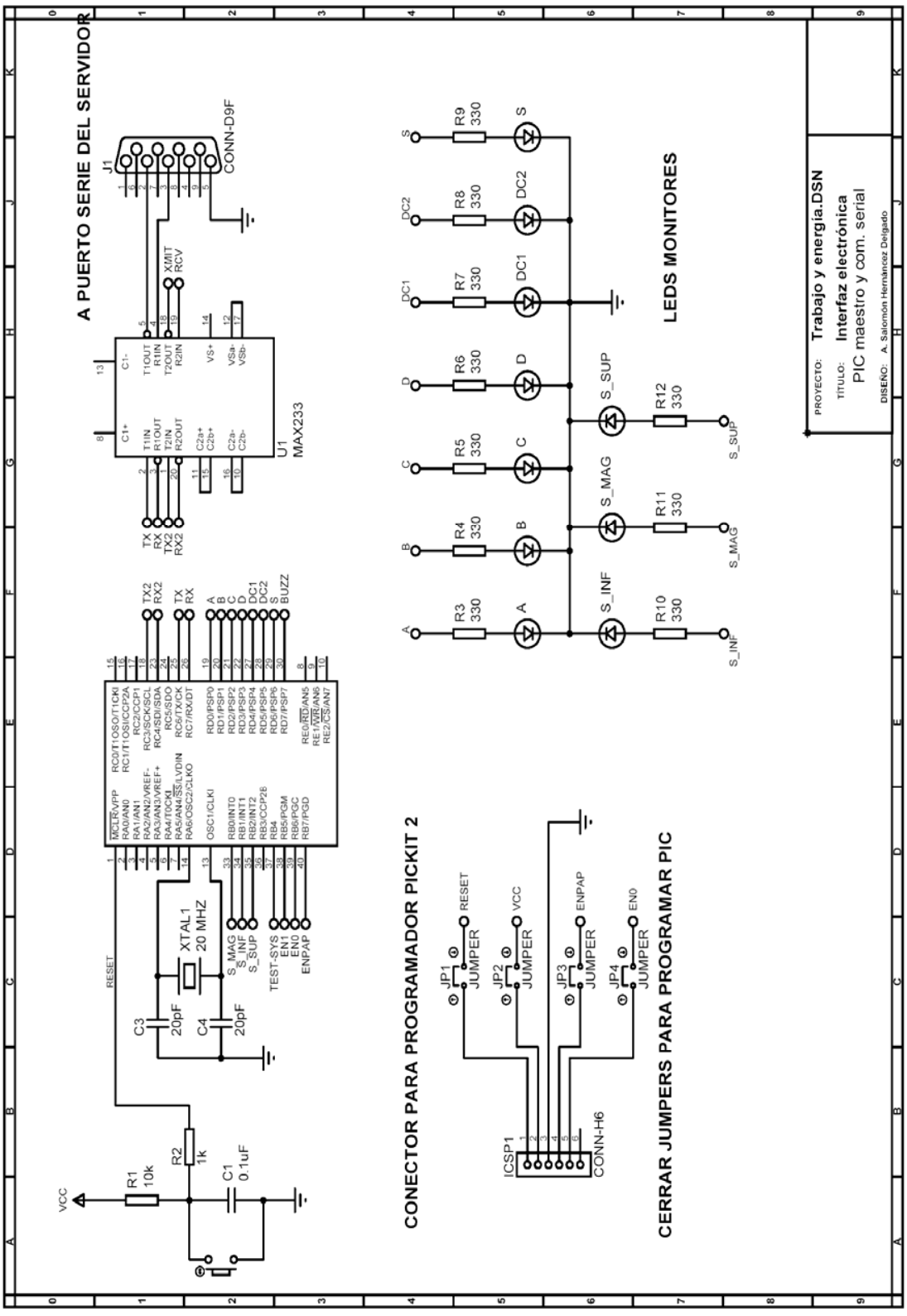


Figura 4.21 Diagrama de la interfaz electrónica

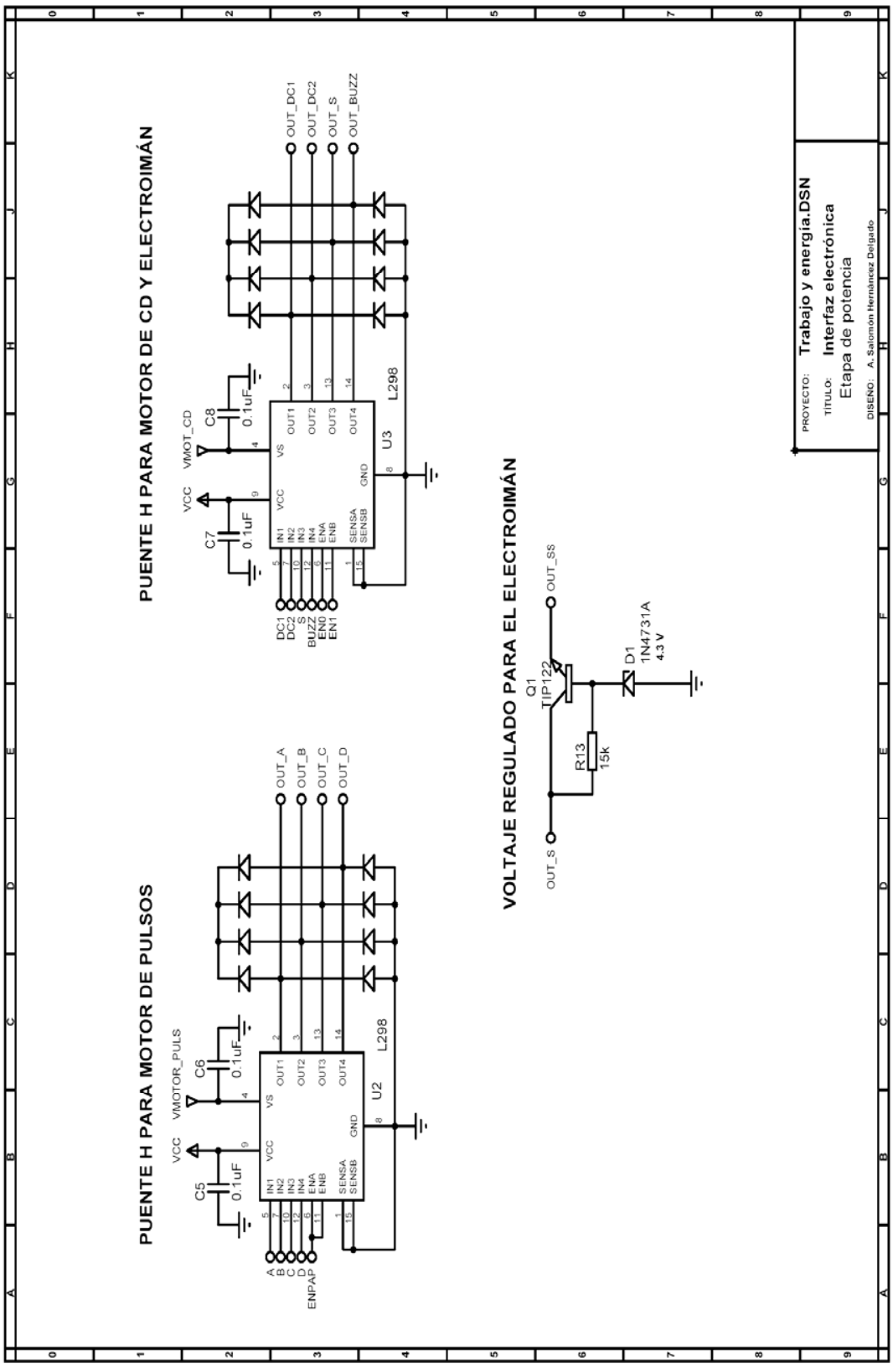


Figura 4.22 Diagrama de la interfaz electrónica (continuación)

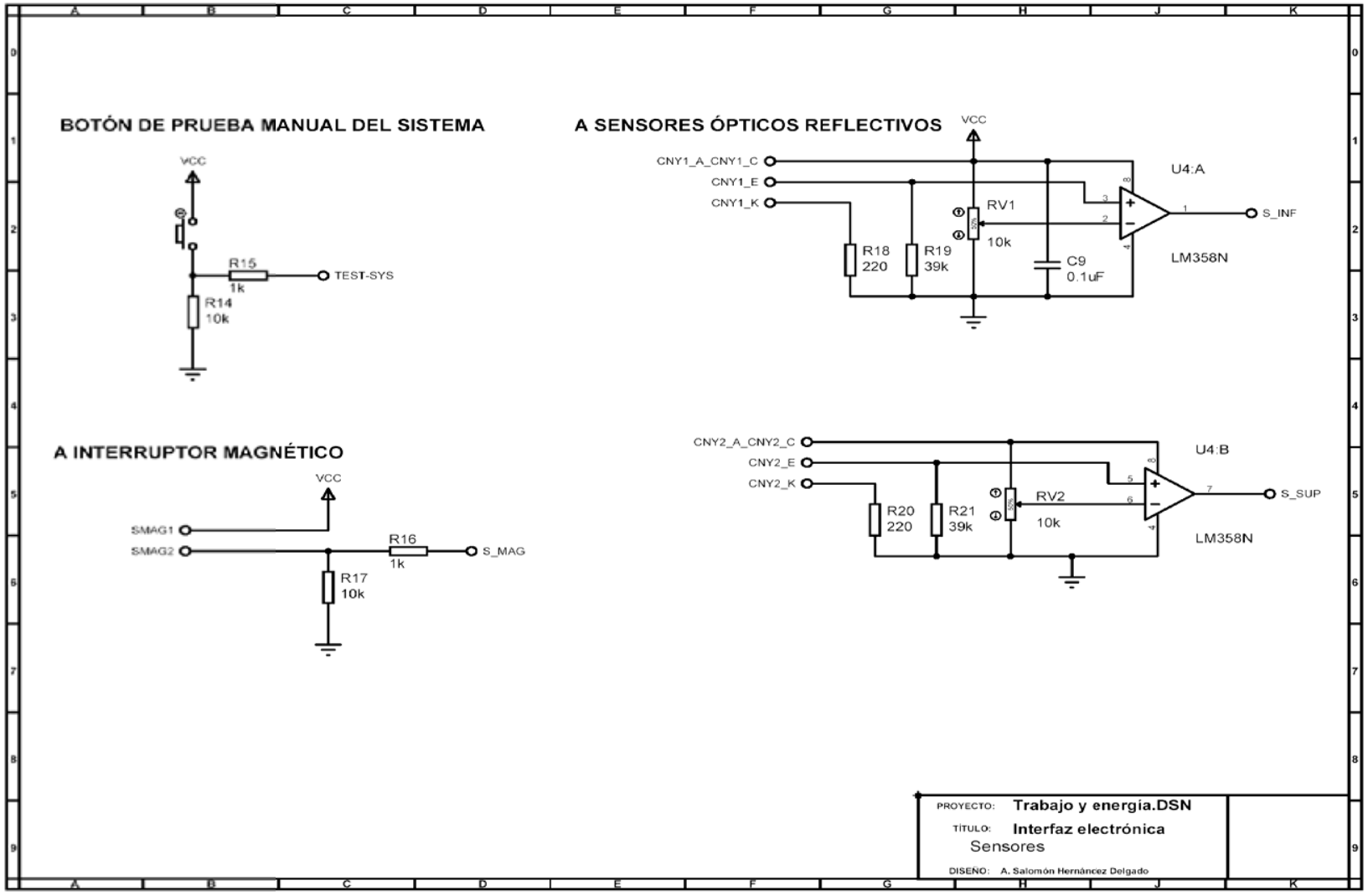


Figura 4.23 Diagrama de la interfaz electrónica (continuación)

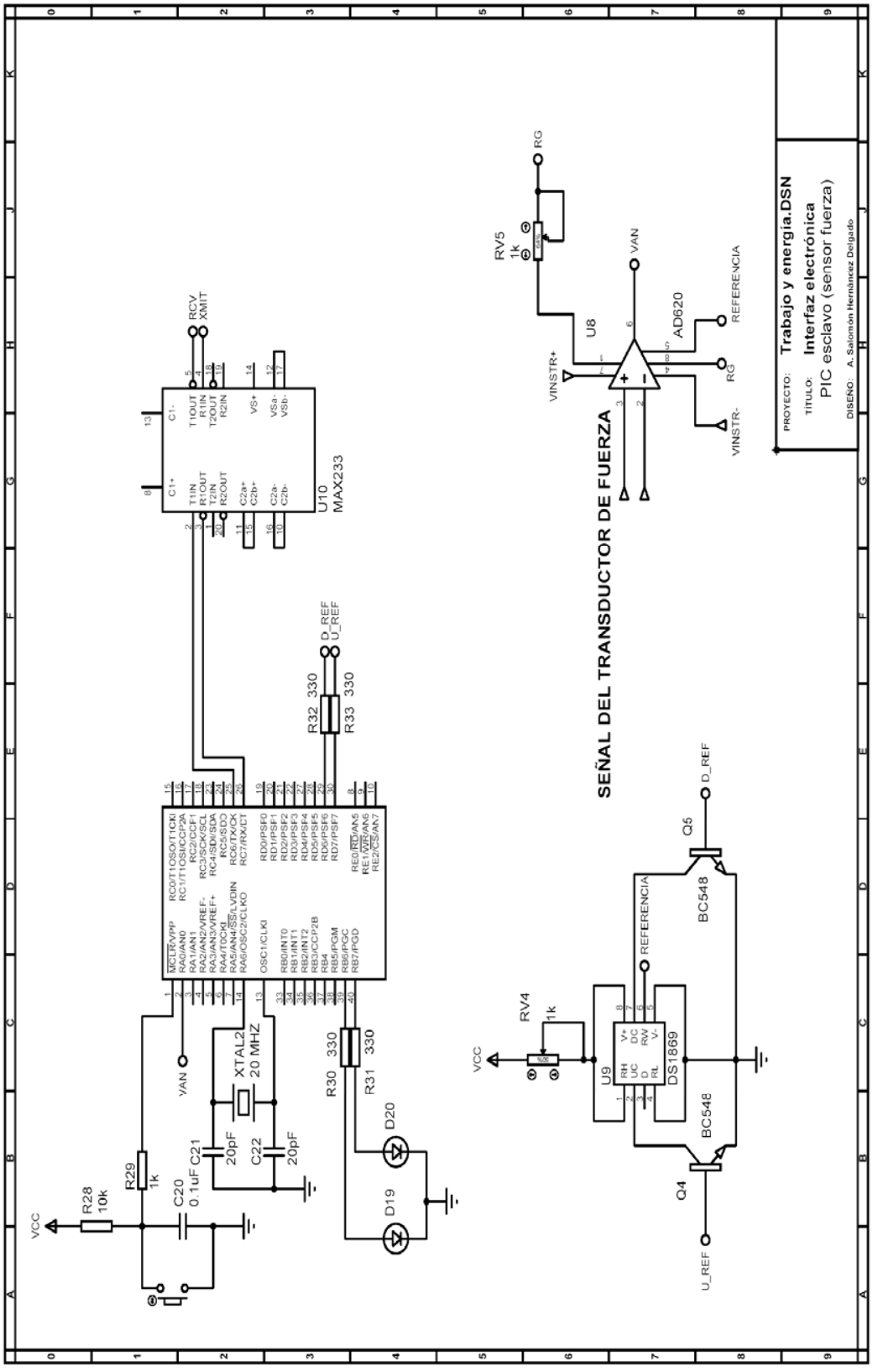


Figura 4.24 Diagrama de la interfaz electrónica (continuación)

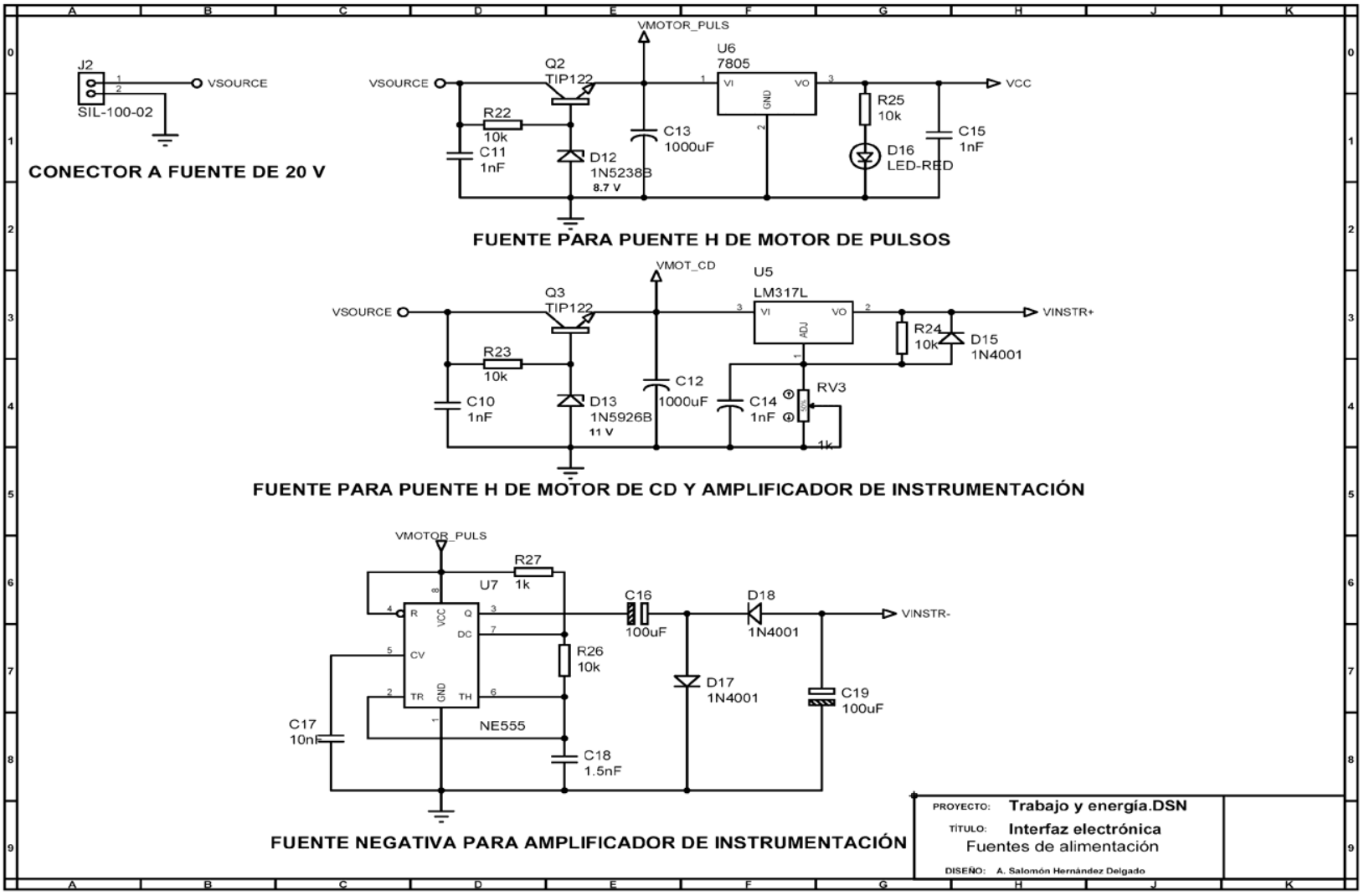


Figura 4.25 Diagrama de la interfaz electrónica (continuación)

CAPÍTULO 5
INTERFAZ CON EL USUARIO

La interfaz con el usuario es la aplicación web a la que el usuario accederá a través de Internet y le permitirá controlar el sistema. El programa de la interfaz con el usuario se ejecutará en el servidor ubicado en el laboratorio remoto.

En el diseño se consideraron los objetivos específicos de este trabajo, el tipo de usuario y el alcance del proyecto para encontrar una solución. La interfaz con el usuario se realizó con la herramienta Visual Web Developer de Microsoft Visual Studio 2008, en lenguaje C#. Se basa en un sistema de navegación que funcionará adecuadamente, enfocándose en el contenido de la práctica, la flexibilidad para ser modificado posteriormente y la facilidad para que el usuario maneje el sistema masa-resorte.

Cabe señalar que aún no se ha desarrollado la página web que contendrá todas las prácticas, que no se abarcaron el manejo de la cámara, ni cuestiones de seguridad y acceso al servidor. La interfaz desarrollada es únicamente para esta práctica y las pruebas realizadas y los resultados obtenidos se reportan en el capítulo 6.

En este capítulo se describirá brevemente el software empleado, la comunicación con el microcontrolador maestro, cómo se controla el sistema masa-resorte desde la interfaz con el usuario y el programa de la interfaz.

5.1 SOFTWARE EMPLEADO

El lenguaje C# y Visual Studio 2008 son dos herramientas poderosas para crear aplicaciones en la computadora. C# es un lenguaje de programación de propósito general y Visual Studio 2008 es un ambiente de desarrollo que hace más fácil la elaboración de aplicaciones Windows, aplicaciones web, servicios web y aplicaciones para dispositivos móviles. El estilo de estas aplicaciones frecuentemente es de programación basada en eventos, presentando una interfaz gráfica con la cual el usuario interactúa [6].

Microsoft Visual Studio 2008 incluye la herramienta de desarrollo Visual Web Developer para aplicaciones web.

Una aplicación web se puede utilizar accediendo a un servidor a través de Internet o de una intranet mediante un navegador (Internet Explorer, Fire Fox, Netscape, etc.). En otras palabras, es un documento que se codifica en un lenguaje soportado por los navegadores web.

5.1.1 PROGRAMACIÓN BASADA EN EVENTOS

El usuario genera acciones de control en eventos basados en el programa. Un ejemplo de esto aparece en la forma de la Figura 5.1.

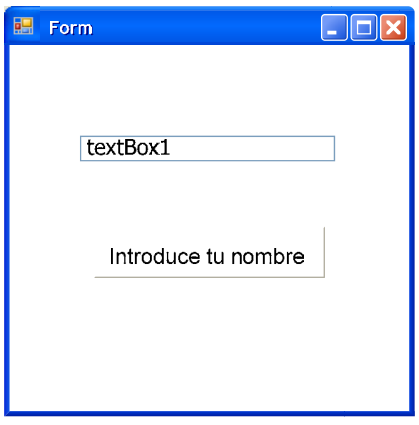


Figura 5.1 Forma simple.

La aplicación que se muestra en la Figura 5.1 espera a que el usuario introduzca un nombre en la caja de texto (TextBox). Cuando el usuario introduce un nombre, el nombre reemplaza el mensaje inicial en la etiqueta (Label). Si el usuario no introduce un nombre o inclusive se retira, entonces no se ejecuta ningún código. El sistema ejecuta código cuando el usuario introduce texto.

Usar Visual Studio significa que se tendrá que escribir poca cantidad de código. Para una aplicación basada en eventos, el código que se necesita escribir consiste en manipuladores de evento. Los manipuladores de evento responden a un evento generado por el usuario [6].

En la aplicación mostrada en la Figura 5.1, se necesita manipular el evento que ocurre cuando el usuario introduce texto en el TextBox. El código consistiría sólo de una línea usada para copiar el contenido del TextBox al Label. Visual Studio escribe el código para crear el TextBox y el Label, posicionarlos en la forma, y cambiar sus propiedades físicas como su dimensión y apariencia. Provee la estructura de código que se necesita para correr la aplicación.

5.1.2 DESARROLLO DE LA APLICACIÓN WEB

Visual Studio provee las formas y los controles que permiten construir aplicaciones web Figura 5.2. Estas aplicaciones se alojan en el sitio web, los usuarios entran al sitio web, abren la aplicación desde sus computadoras y presentan la información que introducen en las formas de la aplicación, al servidor ubicado en el laboratorio.

Figura 5.2 Cuadro de herramientas de algunos tipos de controles.

Figura 5.3 Algunos controles tipo estándar.

La interfaz con el usuario es un sitio web ASP.NET en la que se utilizan controles estándar como controles Button, TextBox, Label, ListBox e ImageButton (Figura 5.3).

Los controles web se ejecutarán en el servidor y analizarán la información para después mandarla al microcontrolador maestro; si se detectara algún error avisarán al usuario, de lo contrario, enviarán la respuesta del programa de vuelta al usuario. La Figura 5.4 muestra una forma web que verifica la distancia de deformación ingresada por el usuario.

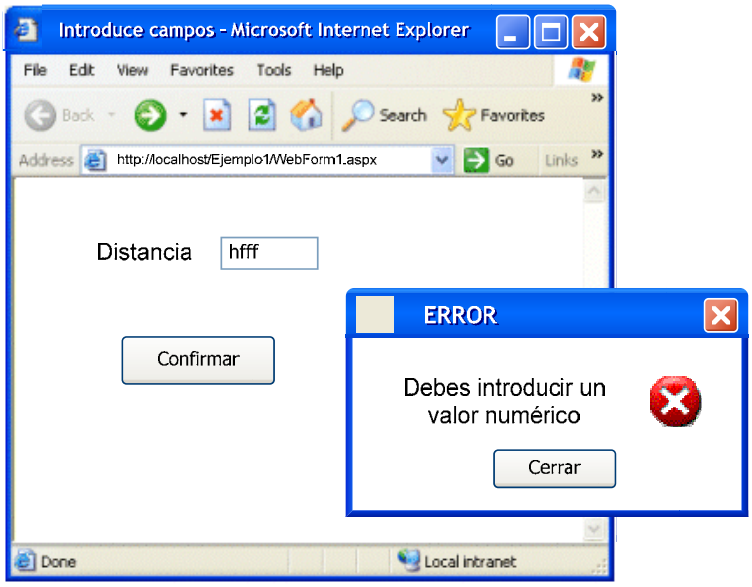


Figura 5.4 Validando las entradas en una forma web.

En la aplicación de la interfaz se agregaron los controles y se programaron sus eventos, Visual Studio generó el código en HTML (Lenguaje de Marcas de HiperTexto) de cada control. Esto generó dos archivos con el mismo nombre pero diferente extensión, uno con extensión “aspx.cs” que contiene el programa de los eventos de los controles en lenguaje C#, y el otro con extensión “aspx” que almacena el diseño y el código HTML, este último archivo es el que manda a llamar el usuario desde Internet.

La aplicación web se dividió en cuatro páginas Figura 5.5, cada una con sus dos archivos de extensión diferente. La primera página contiene la introducción a la página web de la práctica; la segunda contiene la primera parte de la práctica, que es

la de caracterización del resorte; la tercera contiene la segunda parte de la práctica, realización de trabajo por medio de la energía almacenada en el resorte, y la cuarta la descarga de los archivos generados en cada repetición. De esta manera el usuario no tendrá conflictos sobre qué parte de la práctica se está realizando y qué acción del sistema masa-resorte está controlando, sabrá a dónde dirigirse de acuerdo a las instrucciones.

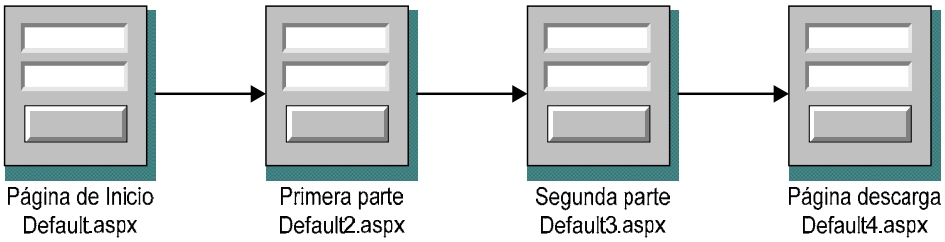
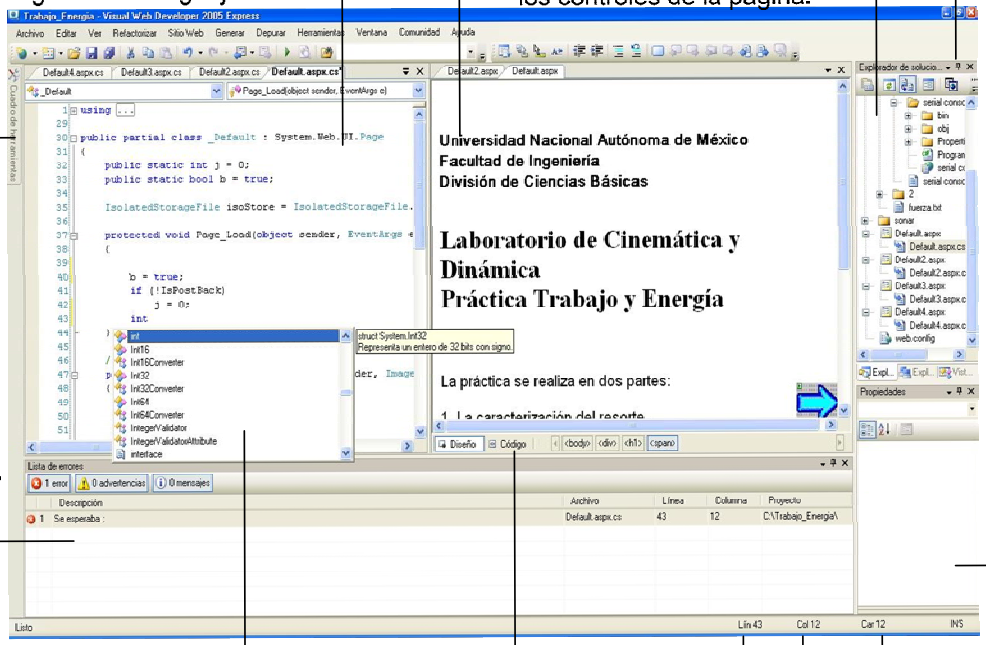


Figura 5.5 Grupo de páginas de la aplicación web.

En la Figura 5.6 se muestra un screen-shot del proyecto ASP.NET de la práctica, donde se explican algunas de las partes principales del área de trabajo. Por ejemplo, se puede ver la lista de todas las propiedades que tiene una clase o control determinado, o se puede ver el código del programa de los controles web en lenguaje C# o HTML. También se puede ver el diseño de la página web y el contenido del proyecto.



Archivo con extensión 'aspx.cs'. Contiene el código del programa en lenguaje C#.

Archivo con extensión 'asp'. Contiene el diseño y código html de la página web, en la vista diseño se observan los controles de la página.

Explorador de soluciones. Muestra el contenido de la carpeta de la solución web.

Vista de clases. Muestra las clases del sitio web.

Cuadro de herramientas. Contiene los controles web para agregar a la página.

Lista de errores. Ventana donde se muestran los errores generados durante la depuración.

Intellisense. Proporciona una gama de opciones que facilitan el acceso a las referencias de lenguaje.

Vista código. Botón para cambiar de vista a el código html de la página.

Barra de estado. Muestra la línea, columna y carácter actual.

Ventana de propiedades. Muestra la propiedades del control web seleccionado; en caso de estar en la pestaña Vista de clases se muestran las propiedades, métodos y eventos de la clase seleccionada.

Figura 5.6 Ambiente de trabajo de Visual Studio ASP.NET.

5.2 DESCRIPCIÓN DE LA INTERFAZ CON EL USUARIO

Se trató de mostrar una presentación que no fuera cansada, difícil de entender y que no incluyera muchas imágenes para evitar una larga espera para que el documento sea cargado. Las únicas imágenes que aparecerán durante todo el desarrollo de la práctica son la del botón de continuar y la de la cámara que se instalará en el laboratorio. Como ya se mencionó, en este trabajo no se abarca el manejo de la cámara.

Para ingresar al sitio web de la práctica Trabajo y Energía, el usuario deberá ingresar en la barra del navegador la dirección:

http://Dirección IP del servidor/Trabajo_Energia/Default.aspx

Cuando un usuario entra a la página web se presenta la práctica y se da una breve introducción sobre lo que tendrá que realizar (Figura 5.7).

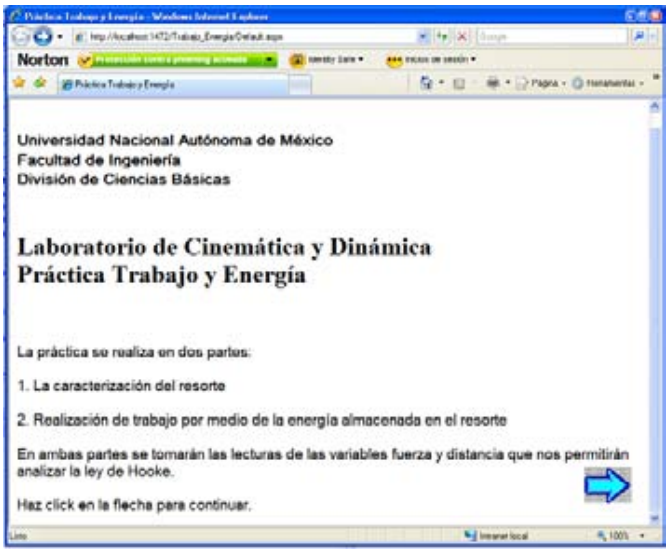


Figura 5.7 Introducción a la página web.

Conforme el usuario avanza dentro de la aplicación web, van apareciendo las instrucciones a seguir y deshabilitándose las instrucciones anteriores, para evitar que el usuarios pudiera presionarlas nuevamente.

Como primera parte se realizará la “Caracterización del resorte”. La primera instrucción será llevar a la posición de HOME el sistema, para iniciar con la caracterización del resorte (Figura 5.8).

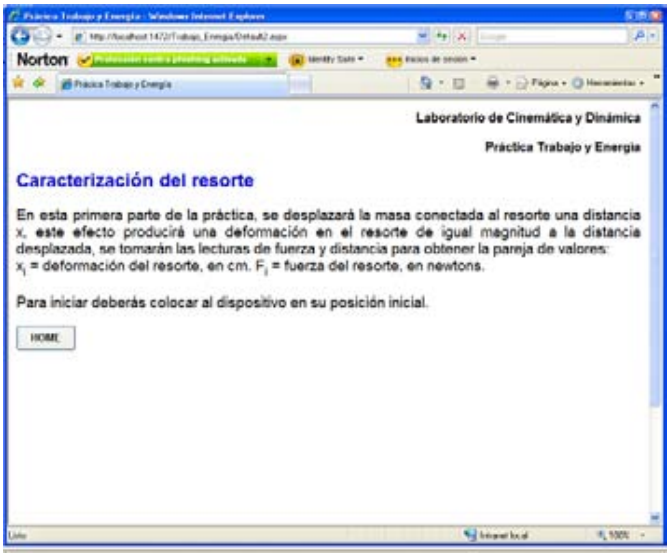


Figura 5.8 HOME. Primera instrucción de la primera parte de la práctica.

Al llegar a HOME, se ocultará el botón y aparecerá en la pantalla el TextBox de la segunda instrucción para ingresar la distancia a deformar (Figura 5.9).

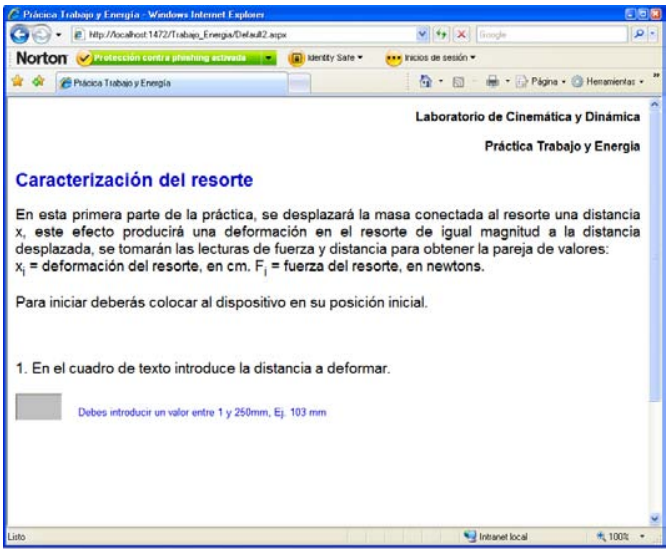


Figura 5.9 Introduce distancia a deformar. Segunda instrucción de la primera parte de la práctica.

Las distancias mínima y máxima de deformación se establecieron para la primera parte en 5 *mm* y 250 *mm*, respectivamente. Una vez ingresada la distancia correcta, se permitirá al usuario presionar el botón INICIO (Figura 5.10). Si no, aparecerá un cuadro de dialogo indicando un error (Figura 5.11).

Figura 5.10 INICIO. Tercera instrucción de la primera parte de la práctica.

Figura 5.11 Mensaje de error al introducir distancia incorrecta.

En el instante que se presione el botón INICIO comenzará la deformación del resorte hasta alcanzar la distancia introducida por el usuario. Al finalizar, se deshabilitarán las instrucciones y se procederá a liberar el cuerpo permitiendo al usuario oprimir el botón SOLTAR (Figura 5.12).

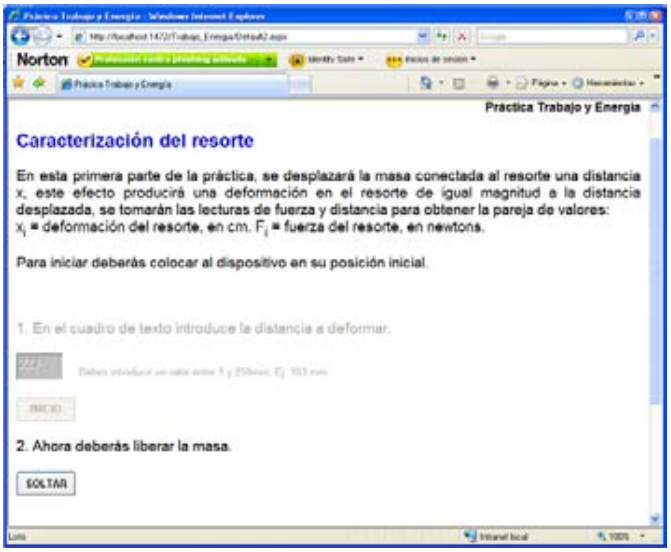


Figura 5.12 SOLTAR. Cuarta instrucción de la primera parte de la práctica.

Las lecturas tomadas por los sensores de fuerza y distancia se mostrarán al usuario en un ListBox (Figura 5.13).

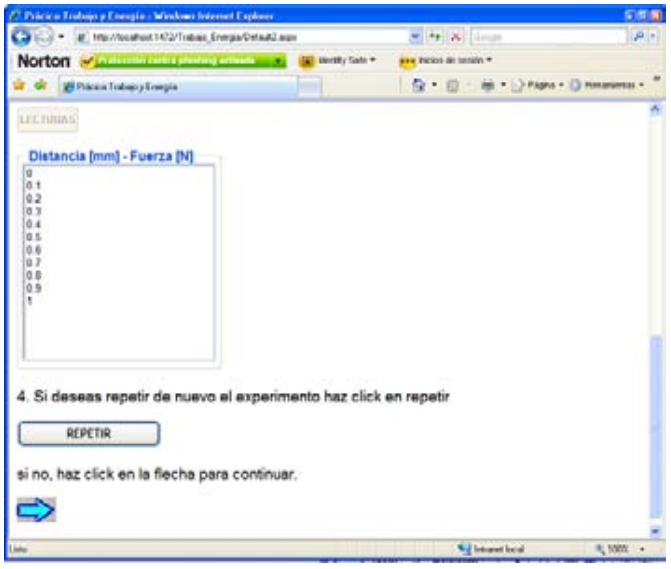


Figura 5.13 La interfaz muestra las lecturas al usuario.

Si el usuario deseará realizar otro evento, podrá repetir el experimento hasta 10 veces.

Para la segunda parte de la práctica “Realización de trabajo por medio de la energía almacenada en el resorte”, se realizarán las instrucciones de la misma manera que en la primera parte (Figura 5.14), y también podrá repetirse hasta 10 veces, con las diferencias de que el rango de distancias es de 100 a 250 *mm* y, como se verá en el desarrollo del programa, la toma de lecturas se realiza de diferente manera y en momentos distintos a los de la primera parte.

Figura 5.14 Primeras cuatro instrucciones de la segunda parte de la práctica.

Todos los archivos que se generen podrán descargarse al final de las dos partes de la práctica (Figura 5.15); las lecturas de cada parte se descargarán en una carpeta comprimida y en diferentes archivos de texto, junto con las lecturas se descargará un documento que explicará cómo se realiza el acoplamiento y manejo de las lecturas tomadas por los sensores. Dentro de los documentos se incluyen, la fuerza, la distancia y los tiempos de muestreo de distancia.

Figura 5.15 Final de la práctica.

En todo momento el usuario podrá ver lo que está pasando en el laboratorio gracias a la cámara que se instalará en él. De esta forma, el usuario podrá controlar remotamente la acción que el sistema masa-resorte debe realizar.

5.3 COMUNICACIÓN SERIAL

La comunicación entre el PIC maestro y el servidor se realiza mediante el puerto serie. En la comunicación serial el protocolo es similar al utilizado por varios dispositivos para instrumentación. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

Se llama serial, porque los bits se reciben uno detrás de otro o “en serie”. La comunicación serial implementada utiliza el protocolo RS-232 que es el más común de los métodos de comunicación serial.

Las características más importantes que se requirieron establecer en la comunicación serial son la velocidad de transmisión, el número de bits de datos, de bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que estas características sean iguales.

CONVERTIDOR USB-SERIAL

Actualmente, algunas PC’s no cuentan con el puerto serial, ya que la tecnología y los dispositivos para el envío y recepción de datos han evolucionado, por lo que este puerto ha sido sustituido por puertos con velocidades de transmisión más altas como el USB; por lo tanto, a falta de un puerto serial en una computadora, se utilizó un convertidor USB a serial (Figura 5.16), el cual hace un buen manejo de los protocolos para crear la compatibilidad con dispositivos con protocolo RS-232.

Figura 5.16 Dispositivo USB-Serial.

El convertidor transforma el dispositivo de conexión USB al conector serial macho DB9, proporcionando la posibilidad de una conexión serial a la computadora; este dispositivo es muy usado en las laptops que no tienen un puerto serial.

5.4 DESARROLLO DEL PROGRAMA

Como ya se vio, el sistema masa-resorte está formado por un sistema mecánico, una interfaz electrónica que cuenta con dos microcontroladores (PIC maestro y PIC esclavo), un sensor ultrasónico (SONAR) y suministro eléctrico. La manera en que se encuentran conectados con el servidor se observa en la Figura 5.17.

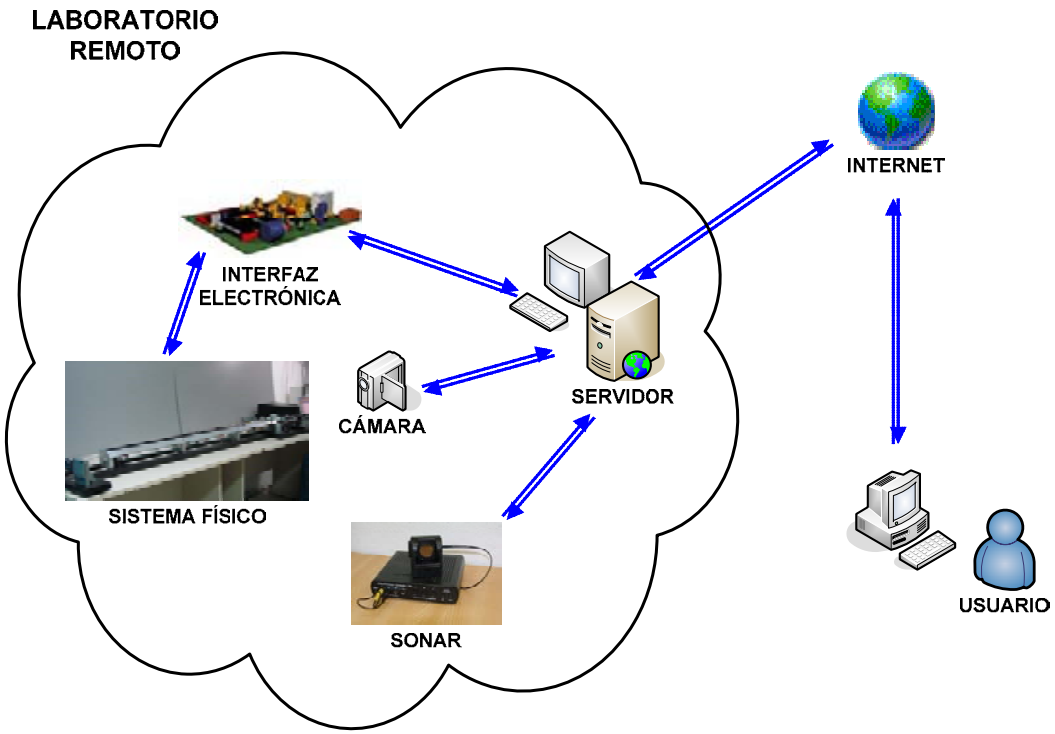


Figura 5.17 Diagrama general de la red

5.4.1 COMANDOS DE LA PC CON LA INTERFAZ ELECTRÓNICA

La interfaz con el usuario se comunica, de forma trasparente al usuario, con el PIC maestro a través del puerto serie del servidor enviando las instrucciones predefinidas, estas instrucciones son comandos que la PC envía a la interfaz electrónica, la cual los interpreta para ejecutar la acción indicada (Tabla 5.1).

Por lo tanto, las instrucciones y los comandos no se pueden cambiar sin tener que modificar los programas utilizados en la interfaz electrónica, ya que podría causar daños, problemas y posiblemente la necesidad de modificar la parte física de acuerdo al programa.

Tabla 5.1 Comandos y funciones usados para el microcontrolador maestro.

COMANDOS PC-INTERFAZ ELECTRÓNICA	
COMANDO	**INSTRUCCIÓN**
H	HOME
d	DISTANCIA PARTE 1
D	DISTANCIA PARTE 2
s	SOLTAR BLOQUE PARTE 1
S	SOLTAR BLOQUE PARTE 2
r	LECTURAS PARTE 1
R	LECTURAS PARTE 2
K	CARÁCTER DE RECONOCIMIENTO

La computadora envía al PIC maestro la instrucción a seguir, después, éste identifica el dato para proceder a ejecutar la acción que indica el comando recibido.

El primer carácter mostrado en la Tabla 5.1 corresponde a la primera función a realizar, la letra “H” representa la posición de HOME del dispositivo masa-resorte.

Los comandos “d” y “D” van seguidos de la distancia a deformar introducida por el usuario, preparan al PIC para recibir la distancia y convertirla a número de pasos del motor. Cuando el bloque (cuerpo) llega a la distancia indicada, el usuario puede liberarlo ejecutando en su navegador la instrucción *soltar bloque*, provocando que el servidor envíe los comandos “s” o “S” al PIC maestro. Los comandos “r” y “R” le piden al PIC maestro, el envío de las lecturas del sensor de fuerza.

Los comandos en minúscula corresponden a la primera parte de la práctica y los comandos en mayúscula a la segunda.

Las funciones explicadas son producto del envío de comandos de la PC al PIC maestro; en la Figura 5.18 se muestran gráficamente los comandos utilizados así como cada una de las respuestas que el PIC regresa.

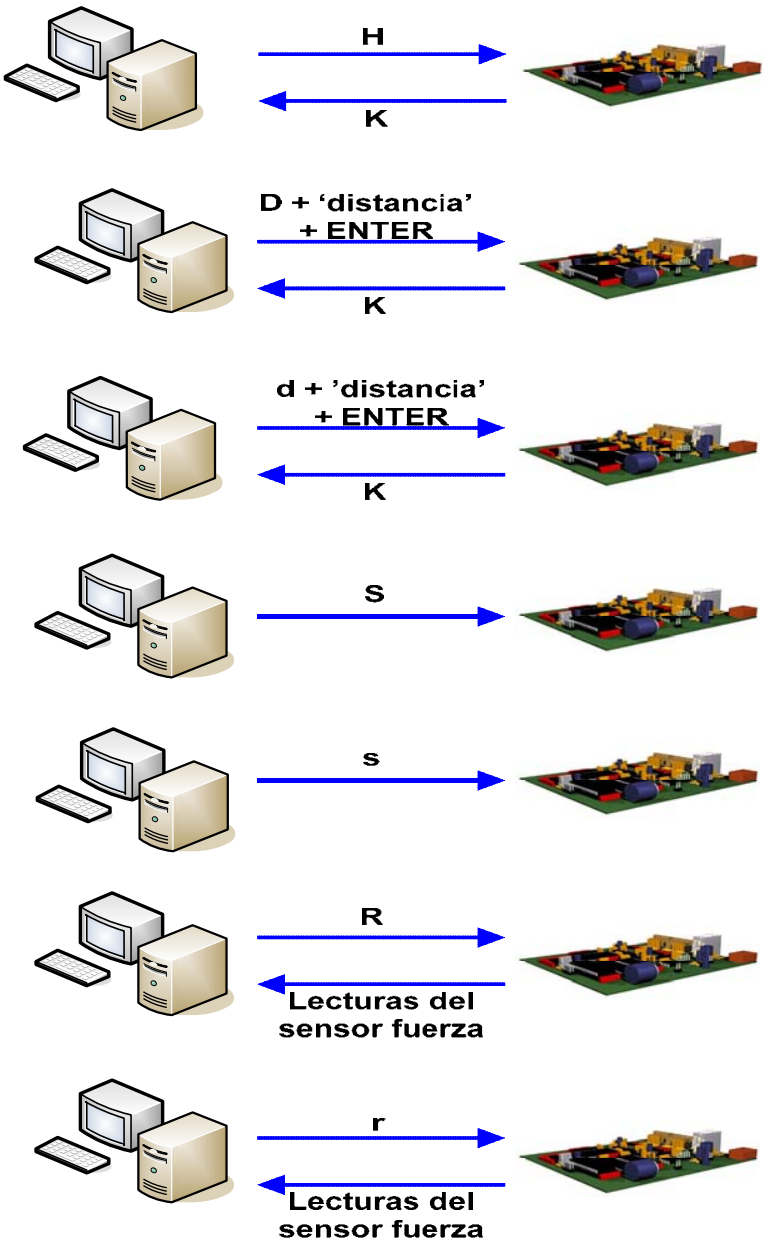


Figura 5.18 Comunicación entre la interfaz con el usuario y el PIC maestro.

Para saber si el sistema masa-resorte ha terminado de realizar alguna acción, el PIC maestro regresa un comando de reconocimiento a la computadora (K), para esto se deja el puerto abierto y se lee continuamente la entrada de éste hasta recibir dicho carácter de reconocimiento; entonces, la computadora continúa con la siguiente instrucción de la interfaz con el usuario.

Si de alguna manera se enviara un carácter que no se encontrara dentro de las instrucciones predefinidas, no se realizaría ninguna acción en el microcontrolador.

Se tomaron en cuenta los tiempos de recepción y envío de los datos por el puerto serial en ambos extremos, computadora y dispositivo físico, para evitar colisiones de paquetes, sobre escritura o empalmes y pérdida de información.

5.4.2 DESCRIPCIÓN DE LOS PRINCIPALES EVENTOS DEL PROGRAMA DE LOS CONTROLADORES DE LA PÁGINA EN LENGUAJE C#

Como ya se explicó, son cuatro archivos “aspx.cs” que contienen el programa de las páginas del sitio web; es necesario mencionar que, para que cada página funcione correctamente, hay que configurar el puerto que se va a utilizar de la siguiente manera:

```
System.IO.Ports.SerialPort serialPort = new
System.IO.Ports.SerialPort("COM1", 9600);
```

En este caso está configurado el puerto “COM1”, con una velocidad de 9600 baudios.

La Figura 5.19 muestra los diagramas de cómo se desarrolla el programa de cada página en la aplicación web. A continuación se explicarán las partes de código más relevantes.

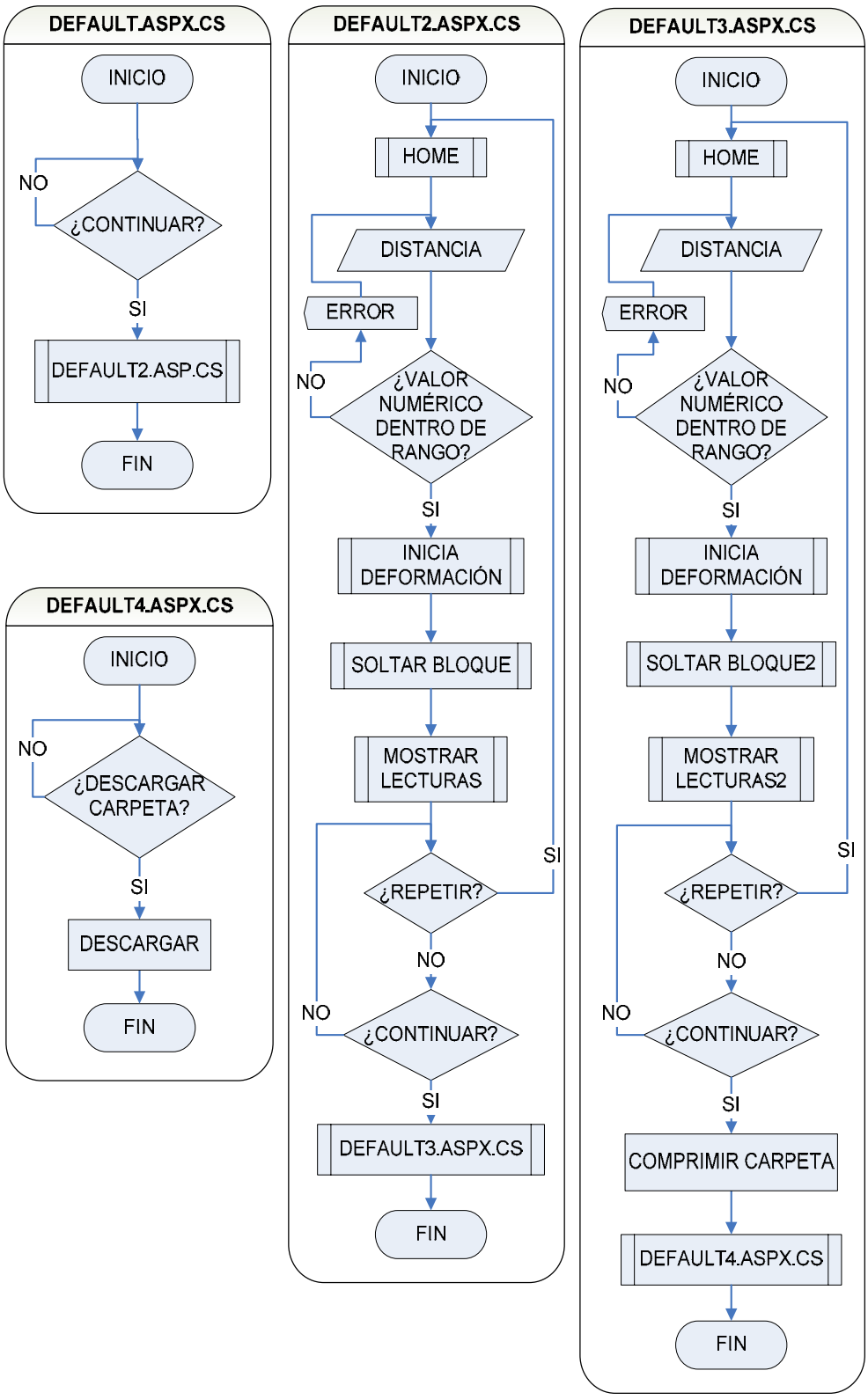


Figura 5.19 Diagramas de flujo de las páginas del sitio web.

Para empezar a describir los controles de las páginas, lo haremos con el primer botón (HOME), que es el encargado de enviar el carácter 'H' al PIC maestro en ambas partes de la práctica.

Para enviar un dato por el puerto serie se abre el puerto, envía el dato y se cierra el puerto, el código siguiente muestra de que manera se realiza esto.

```
serialPort.Open();
serialPort.Write("H");
serialPort.Close();
```

La Figura 5.20 muestra el diagrama de flujo del evento que se produce al presionar el botón HOME en ambas partes de la práctica.

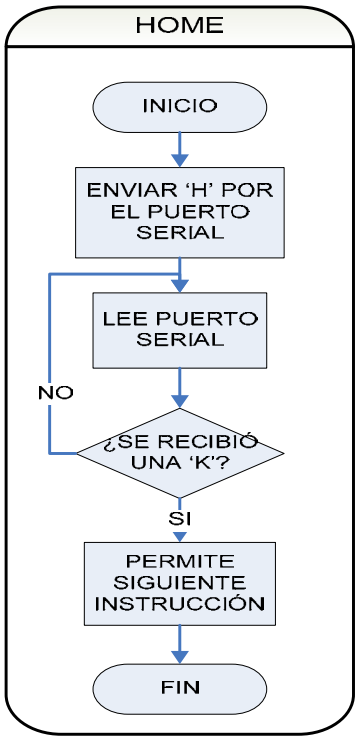


Figura 5.20 Diagrama de flujo de botón HOME.

Para la distancia ingresada en los TextBox de la práctica, se requiere verificar que el dato ingresado sea un número y se encuentre dentro del rango especificado. Esto se realiza mandando llamar la función "IsNumber" y preguntando si el dato es válido como se muestra a continuación:

```
//Estableciendo valores mínimo y máximo
public decimal myMaxvalue = 250;
public decimal myMinvalue = 5;
//Función IsNumber
public bool IsNumber(string inputvalue)
{
     Regex isnumber = new Regex(@"^-?[0-9]+(\.?[0-9]+)?$");
      return isnumber.IsMatch(inputvalue);
}
//Preguntando si el valor está dentro del rango
if (myMinvalue <= numero && numero <= myMaxvalue)
{ …
}
```

La deformación del resorte se hace presionando el botón INICIO. El diagrama de flujo completo del el evento Iniciar deformación para amabas partes de la práctica es el mismo y se observa en la Figura 5.21.

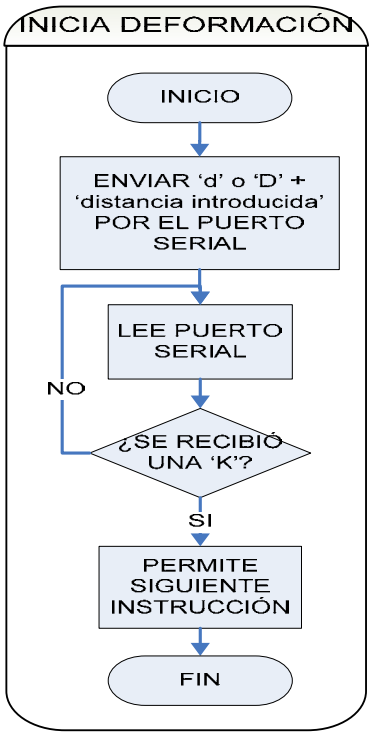


Figura 5.21 Diagrama de flujo del botón INICIO.

La instrucción que sigue de iniciar la deformación del resorte en ambas partes de la práctica es Soltar el Bloque, sin embargo son ligeramente distintos los eventos que se ocasionan al presionar el botón SOLTAR de cada parte de la práctica.

En la primera parte, no se hace uso del sonar, por lo tanto sólo se envía el carácter 's' al PIC maestro, como se observa en la Figura 5.22a.

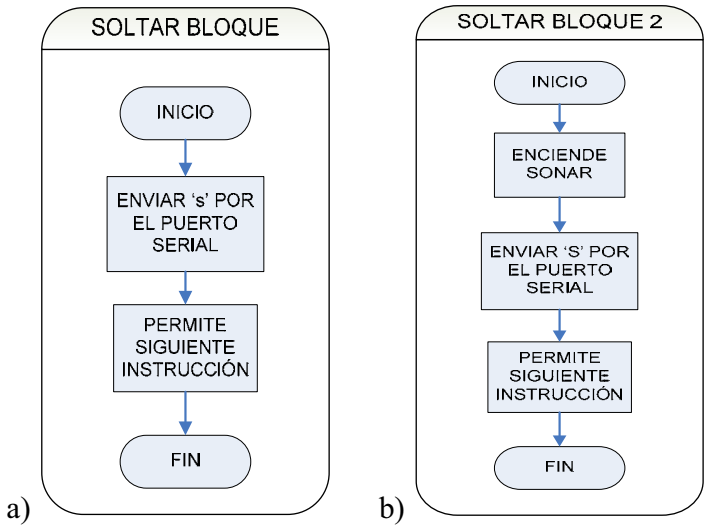


Figura 5.22 a) Diagrama de flujo del botón SOLTAR (a) de la primera parte y (b) de la segunda parte de la práctica.

En la segunda parte de la práctica sí se utiliza el sonar para poder mostrar al usuario la posición en la que se encuentra y la posición que alcanza al ser liberado, por lo que antes de enviar 'S' por el puerto serie, se ejecuta el programa del sonar (Figura 5.22b).

Para encender el sonar se debe ejecutar en el servidor la aplicación de consola *pasco.exe*, que además debe recibir los argumentos de periodo de muestreo y tiempo total tomando muestras. Para realizar esto desde la interfaz con el usuario se realiza lo siguiente.

1 se crea el proceso para ejecución del programa
2 se establece la ruta de acceso al archivo pasco.exe y los argumentos a enviar
3 se define el directorio en el que se creará el archivo de las lecturas recibidas
4 se ejecuta el proceso

5 se reciben las lecturas en la ubicación definida anteriormente
6 al finalizar el tiempo de toma de muestras, se cierra el proceso.

```
//ENCIENDE SONAR
      String datos;
      System.Diagnostics.ProcessStartInfo psi = new
                  System.Diagnostics.ProcessStartInfo();
      psi.FileName = @"C:\Trabajo_Energia\sonar\pasco.exe";
      psi.Arguments = "25";
      psi.Arguments += " " + "3000";
      psi.WorkingDirectory = @"C:\Trabajo_Energia\sonar\";
      psi.UseShellExecute = F;
      psi.RedirectStandardOutput = V;
      System.Diagnostics.Process p =
                  System.Diagnostics.Process.Start(psi);
      System.IO.StreamReader sOut = p.StandardOutput;

//LECTURAS DEL SONAR
       datos = sOut.ReadToEnd();
       datos = datos.Substring(121);
```

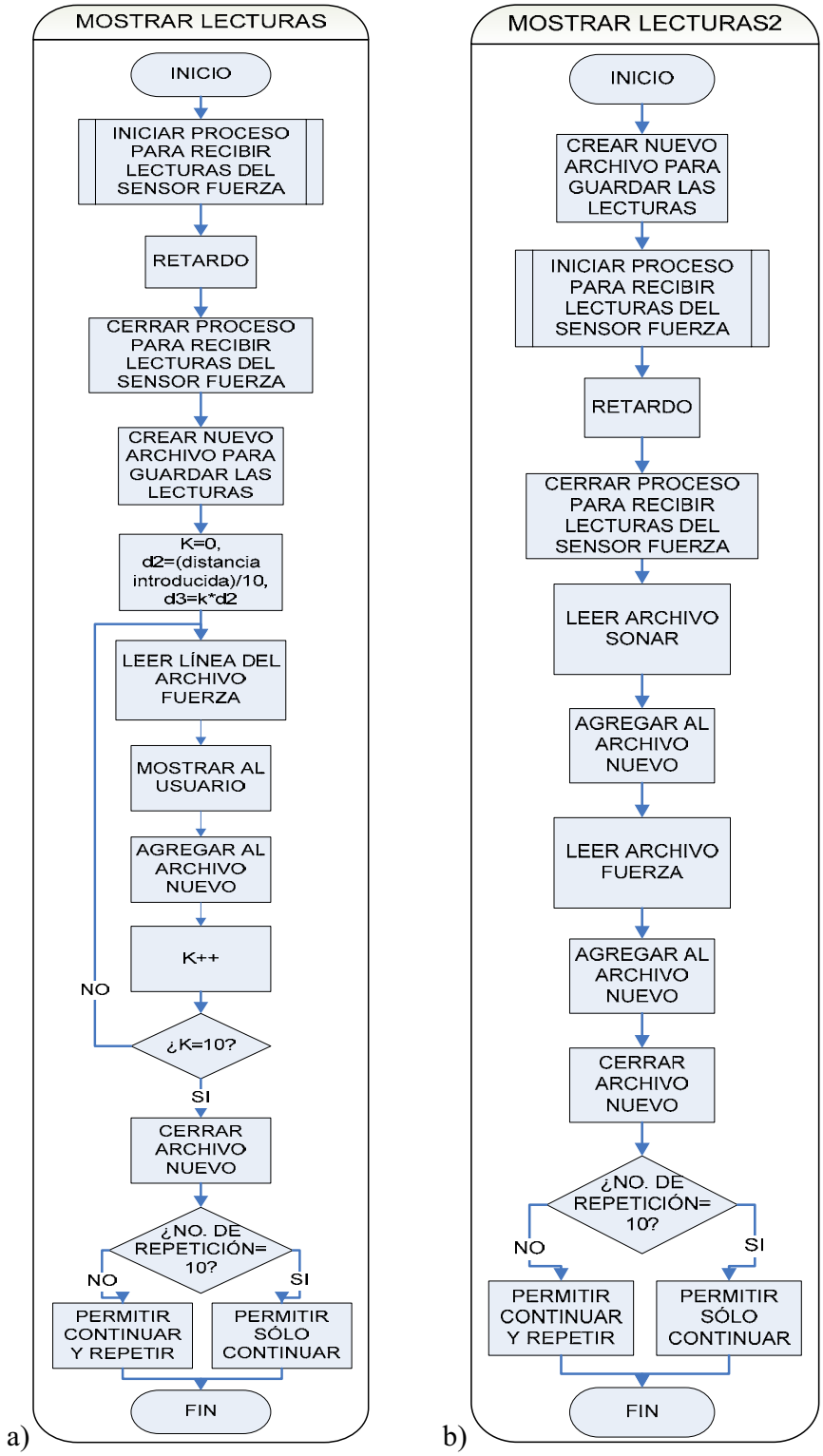


Figura 5.23 Diagrama de flujo del botón LECTURAS (a) de la primera parte y (b) de la segunda parte de la práctica.

Para recibir las lecturas del sensor de fuerza, se inicia el proceso que describe el diagrama de la Figura 5.24, se espera un tiempo y se cierra el proceso de la siguiente forma:

```
//Proceso para recibir lecturas del sensor_fueerza
System.Diagnostics.Process proc = new
     System.Diagnostics.Process();
proc.StartInfo.FileName =
     @"C:\Trabajo_Energia\sensor_fuerza\1\serialconsole\bin\
     Debug\serialc.exe";
proc.StartInfo.CreateNoWindow = true;
proc.Start();
//Pausa para esperar a recibir datos
Int32 miliseconds_to_sleep1 = 4000;
System.Threading.Thread.Sleep(miliseconds_to_sleep1);
//Cierre del proceso
proc.Kill();
```

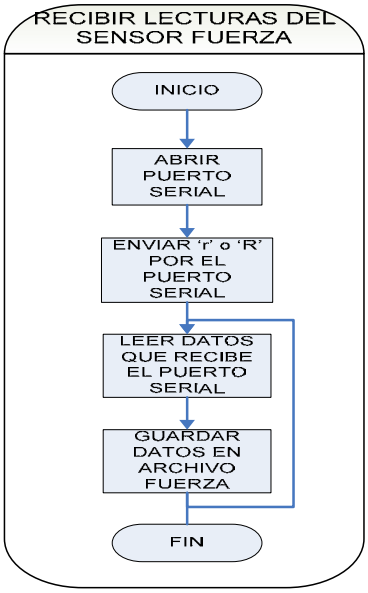


Figura 5.24 Diagrama de flujo del proceso para recibir las lecturas del sensor de fuerza.

A pesar de que el proceso para recibir las lecturas del sensor fuerza es igual para las dos partes de la práctica, se requirieron de dos ejecutables distintos, ya que se reciben distintas cantidades de lecturas, para la primera parte son las 10 lecturas de los 10 tramos y para la segunda son todas las lecturas desde el inicio hasta el cierre

del proceso. En la primera parte el ejecutable envía ‘r’ al PIC maestro y en la segunda parte envía ‘R’.

Los programas del sonar y del sensor fuerza, envían los datos obtenidos a archivos predeterminados, en la interfaz con el usuario se mandan a llamar estas lecturas de cada uno de los archivos y se guardan en archivos nuevos, para evitar que cuando se ejecuten de nuevo los programas, se pierdan las lecturas anteriores

Las lecturas se muestran al usuario en un ListBox al presionar el botón LECTURAS. En el caso de la primera parte sólo se muestran las del sensor de fuerza y los tramos donde fueron tomadas estas lecturas, que se establecen dividiendo la distancia ingresada por el usuario en diez tramos (Figura 5.23a).

Para el evento de mostrar lecturas de la segunda parte, sí se muestran los datos de los archivos generados por ambos sensores, y se guardan nuevamente pero ahora en un solo archivo (Figura 5.23b).

El usuario tiene permitido realizar hasta diez experimentos de cada parte de la práctica, si ya ha repetido diez veces, se le indicará al usuario y sólo aparecerá el botón para continuar.

Al finalizar ambas partes de la práctica, el usuario podrá descargar los archivos que se crearon en cada repetición. Para lograr que se descargaran todos en la misma carpeta, se tuvo que comprimir la carpeta contenedora. El código del programa para la compresión se muestra a continuación:

```
//COMPRIMIR DIRECTORIO
using (ZipFile zip = new ZipFile("c:\\Lecturas.zip\\"))
{
      zip.AddDirectory("c:\\Trabajo_Energia\\Lecturas\\");
      zip.Save();
}
```

5.5 CREACIÓN DEL DIRECTORIO VIRTUAL EN EL SERVIDOR

Para publicar una página web es necesario crear un directorio virtual en el servidor con la herramienta de publicación de páginas de Internet IIS (Internet Information Server).

Un directorio virtual es un directorio en el servidor al que se puede acceder a través de un navegador web, como si estuviera dentro del directorio de publicación habitual de IIS C:\Inetpub\wwwroot.

Se accede al directorio virtual desde el explorador web con la dirección http://servidor/directorio_virtual, pero no tiene porque existir una correspondencia en disco de este directorio dentro de la carpeta de publicación, es decir, no tiene por qué existir el directorio C:\Inetpub\wwwroot\directorio_virtual, sino que dicho directorio podría estar en cualquier otro sitio de nuestro disco duro, por ejemplo C:\mis_paginas. Los directorios virtuales se pueden mapear hacia otro directorio del disco duro, o incluso a otro directorio situado en otro ordenador de la red.

Para definir un directorio virtual en la consola administrativa de IIS, se pulsa con el botón derecho del ratón sobre el sitio web predeterminado y se selecciona la opción "Nuevo > Directorio Virtual..." (Figura 5.25). Entonces aparece un asistente que indica paso a paso el proceso.

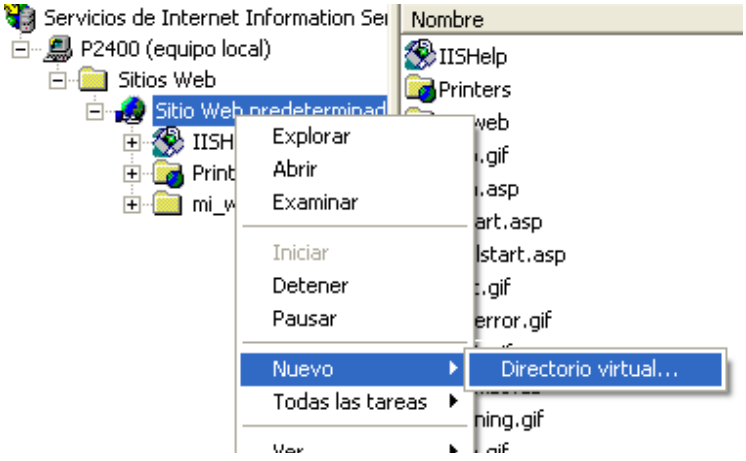


Figura 5.25 Ventana del explorador de IIS (Servicios de "Internet Information Server").

En el primer paso del asistente se solicita el "alias" o nombre que se dará al directorio, en el caso de la práctica Trabajo y Energía fue “TE”. El segundo paso pide la localización física de ese directorio en el disco duro o en la red local. Finalmente solicita los permisos que se desean asignar a ese directorio (Figura 5.26). Los permisos *Leer* y *Ejecutar secuencias de comandos (Por ejemplo, ASP)* suelen ser suficientes para la mayoría de los casos; sin embargo en el caso de la práctica Trabajo y Energía, fue necesario modificar las propiedades de seguridad de la carpeta contenedora de todo el proyecto. La manera en cómo se realizó esto se indica más adelante.

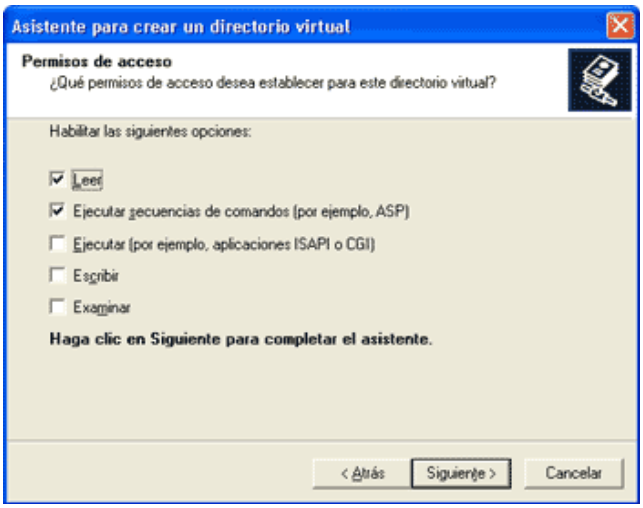


Figura 5.26 Asistente para crear un directorio virtual (solicitud de permisos).

Una vez finalizado el asistente, se ha creado el directorio virtual y se podrá acceder a través del alias que se haya seleccionado. Para acceder a la página de la práctica la dirección que se debe escribir en el explorador web es http://IP_servidor/TE.

MODIFICACIÓN DE LAS PROPIEDADES DE SEGURIDAD DEL DIRECTORIO FÍSICO

Además de establecer las propiedades de lectura, escritura y ejecución del directorio virtual, es necesario modificar las propiedades de seguridad del directorio físico como se muestra a continuación:

1 Hacer click con el botón secundario del mouse en el directorio físico, y seleccionar *Propiedades* (Figura 5.27).

Figura 5.27 Menú propiedades.

2 En la pestaña *Seguridad* de la ventana propiedades que se abre, seleccionar *Cuenta de invitado para internet* y marcar las casillas de *Lectura* y *ejecución*, *Mostrar contenido de la carpeta*, *Leer* y *Escribir* como permisos permitidos (Figura 5.28).

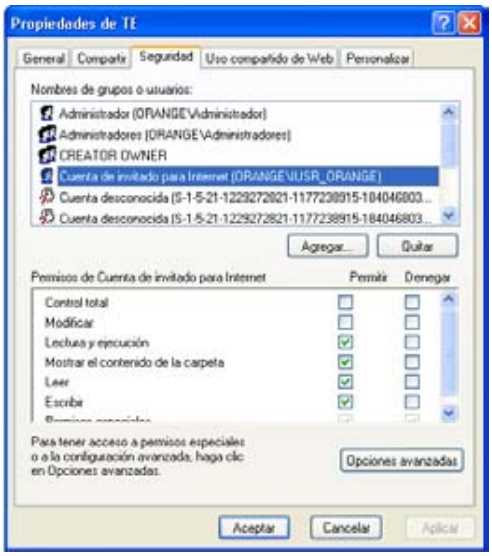


Figura 5.28 Propiedades de la carpeta del proyecto.

3 Después, hacer click en *Opciones Avanzadas*, se abre la ventana *Configuración de seguridad avanzada para TE*. En la pestaña *Permisos*, en *Entradas de permisos* seleccionar *Cuenta de invitado para internet (Tipo permitir)* y hacer click en el botón *Modificar* (Figura 5.29).

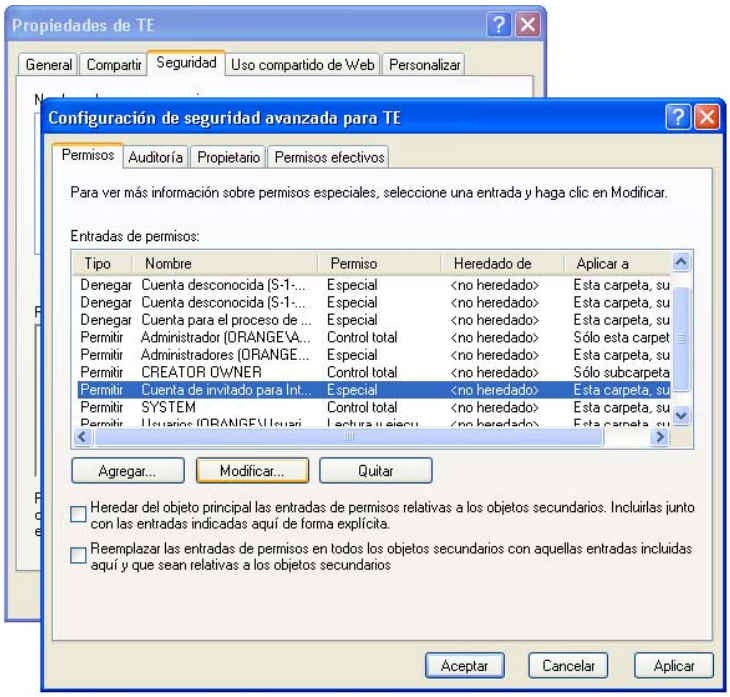


Figura 5.29 Configuración de seguridad avanzada.

4 Se abre la ventana *Entrada de permiso para TE* (Figura 5.30), ahí se seleccionan las casillas de permisos que se quieran permitir. Para la carpeta de la práctica de esta tesis se activaron los permisos:

- Recorrer carpeta/Ejecutar archivo
- Listar carpeta/Leer datos
- Atributos de lectura
- Crear archivos/Escribir datos
- Crear carpetas/Anexar datos
- Atributos de escritura
- Atributos extendidos de escritura
- Eliminar subcarpetas y archivos, y
- Permisos de lectura.

Todos estos permisos fueron necesarios ya que se están leyendo, creando y modificando continuamente, nuevos archivos y carpetas donde se almacenan las lecturas de los sensores de fuerza y distancia, de otra forma no podrían crearse las carpetas y archivos que guardan los datos de los sensores.

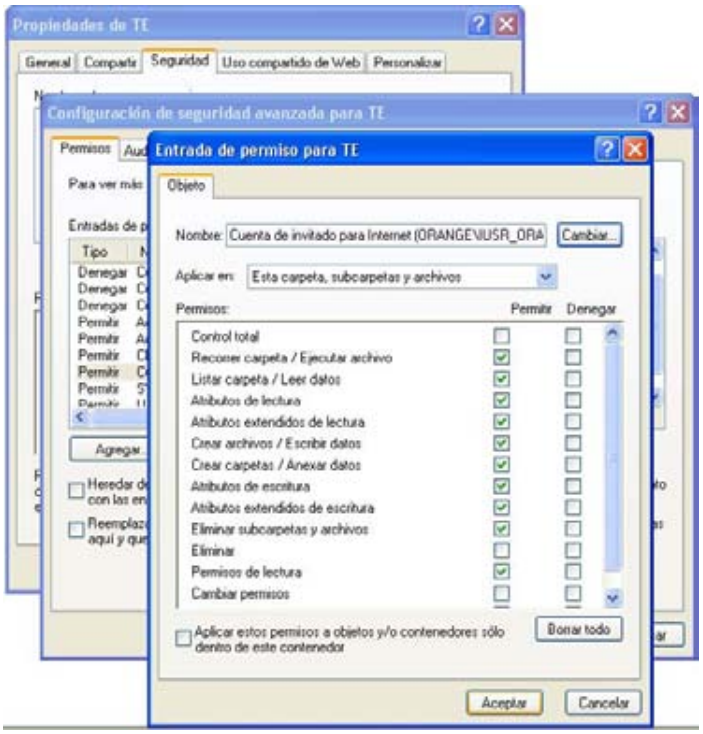


Figura 5.30 Ventana para modificar permisos.

5 Se da click en el botón *Aceptar* para guardar los cambios, en todas las ventanas que se abrieron.

En el caso de los archivos ejecutables del sonar y del sensor fuerza, se colocaron en una carpeta distinta a la que contiene todo el proyecto, ya que por cuestión de seguridad, IIS no permite ejecutar archivos contenidos en el directorio virtual, ya que los interpreta como ejecutables que se reciben del exterior y que pueden ser virus o causar daños al equipo.

El programa desarrollado en ASP realiza funciones específicas de manejo de archivos y ejecución de procesos en el servidor, sin que ello signifique vulnerabilidades en la página y que el servidor pudiera ser blanco de ataque por

hackers. Los procesos llevados a cabo por el código ASP, tienen permiso de ejecución en el servidor, estos mismos procesos son los que llaman a los archivos ejecutables del sonar y del sensor fuerza para recibir los datos de estos sensores, sin que exista conflicto de seguridad.

RESULTADOS Y CONCLUSIONES

La metodología que se siguió desde el inicio, fue dividir en etapas el proyecto e ir logrando metas particulares, y que al ir avanzando en el mismo, se fueran alcanzando los objetivos planteados en este trabajo.

La primera meta a conseguir fue la construcción de los módulos de potencia que forman parte de la interfaz electrónica; una vez probado su funcionamiento, se procedió al diseño e implementación del control basado en un microcontrolador PIC que funcionara de forma independiente, es decir, sin necesidad de una interfaz con una PC. Hasta ese momento no se tenía contemplada la implementación de un segundo PIC para la medición de fuerza; sin embargo, el transductor de fuerza ya se había construido y probado su funcionamiento.

Una vez lograda esta etapa, la siguiente fue lograr la comunicación con protocolo RS-232 entre el PIC maestro y la PC; el PIC que se usó en estas pruebas fue el PIC18F452, sin embargo, se evidenció la necesidad de un segundo PIC para implementar el sensor fuerza.

La comunicación entre los PICs presentó algunos problemas debido a la falta de experiencia en el empleo de los lenguajes de programación, lo cual se solucionó con la investigación del funcionamiento del envío y recepción de datos, y además con la implementación de programas sencillos que permitieron identificar los posibles problemas o errores de programación.

Una vez construida la interfaz electrónica, se realizaron las primeras pruebas de comunicación con una PC en una interfaz gráfica preeliminar realizada en Visual Basic. Sin embargo, para facilitar futuras modificaciones de la interfaz, se estableció que la plataforma de desarrollo para las páginas del proyecto PAPIME, se realizara en C#.

Conforme se avanzó en la programación de la interfaz, se realizaban constantes pruebas de comunicación con el PIC maestro hasta lograr el control del sistema masa resorte desde la PC.

Después, se realizaron pruebas simulando una conexión con un servidor desde la misma PC (Localhost); al resultar satisfactorias se procedió a la etapa final que consistió en probar el control desde internet con conexión a un servidor real.

Las pruebas en el servidor presentaron fallas cuando el programa de la interfaz web manejaba archivos y carpetas ubicados en el servidor, los cuales contienen las lecturas de los sensores de fuerza y distancia.

Después de investigar sobre la administración de sitios web, se logró el manejo de los archivos modificando las propiedades de seguridad que impedían acceder a ellos, ya que por omisión, la configuración del servidor permite sólo la lectura de paginas html.

Se realizaron pruebas para comparar los resultados entre la práctica realizada manualmente con los instrumentos de medición aún utilizados en el laboratorio (dinamómetro y flexómetro), y la práctica realizada con el sistema masa-resorte a través de la interfaz web. En los resultados se apreció una disminución del error en la obtención de la rapidez del bloque y una mejora en los datos de caracterización del resorte, debido a una mejora en la precisión, exactitud y repetibilidad en los eventos.

Se logró controlar el mecanismo del experimento y obtener las lecturas del sonar y sensor fuerza a través de internet, empleando un servidor web para hospedar la interfaz de usuario y una PC con conexión a internet que actuó como usuario.

Las pruebas finales no presentaron inconvenientes en el funcionamiento de los componentes del sistema, tanto de hardware (mecanismos e interfaz electrónica) como de software (interfaz web de usuario y programas de los microcontroladores).

Se debe considerar que estas pruebas se realizaron a una distancia muy corta entre el servidor y el usuario, y no se observaron tiempos de ejecución largos entre eventos en el programa de la interfaz y el funcionamiento del dispositivo experimental.

El desarrollo de la interfaz electrónica y la interfaz de usuario para este proyecto es tan sólo una aproximación a lo que puede ser un proyecto a mayor escala, y sin duda se espera que sea una aportación al desarrollo tecnológico en la Facultad de Ingeniería.

Se puede considerar que se alcanzaron los objetivos planteados para este proyecto de tesis; sin embargo, existen algunos aspectos que se pueden mejorar, por ejemplo:

- incrementar la frecuencia de muestreo del sonar o sensor de distancia ya que tiene una frecuencia de muestreo baja, aproximadamente de 40 muestras por segundo
- sincronizar la toma de lecturas de los sensores de fuerza y distancia al realizar la medición de la segunda parte de la práctica, pues no están sincronizados
- la página de internet aun tiene riegos de quedarse realizando algún proceso si se hace un mal uso de los controles; se aconseja mejorar la detección de errores y repetición de comandos por parte del usuario. Además, darle un diseño más atractivo
- debido al avance de la tecnología, los puertos seriales en las PCs han venido en desuso, siendo reemplazados por puertos más rápidos como el USB. Por tal motivo, se aconseja actualizar el protocolo de comunicación RS-232 por el USB
- instalar la cámara en la página para poder visualizar el experimento, haciendo más didáctica la realización de la práctica para el alumno.

APÉNDICE A
PRINCIPIO DE TRABAJO Y ENERGÍA

A.1 TRABAJO

Si la fuerza resultante F que actúa sobre una partícula es constante (en magnitud y dirección) el movimiento se realiza en línea recta en la dirección de la fuerza. Si la partícula se desplaza una distancia *x* por efecto de la fuerza F (Figura A.1), entonces se dice que la fuerza ha realizado trabajo U sobre la partícula de masa m, que en este caso particular se define como:

$$U=Fx \tag{A.1}$$

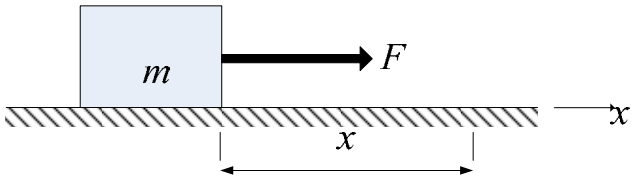


Figura A.1 Trabajo sobre una partícula.

Si la fuerza constante no actúa en la dirección del movimiento, el trabajo que se realiza es debido a la componente de la fuerza en la dirección paralela al movimiento, como se ve en la Figura A.2. La componente *y* de la fuerza, perpendicular al desplazamiento, no realiza trabajo sobre el cuerpo.

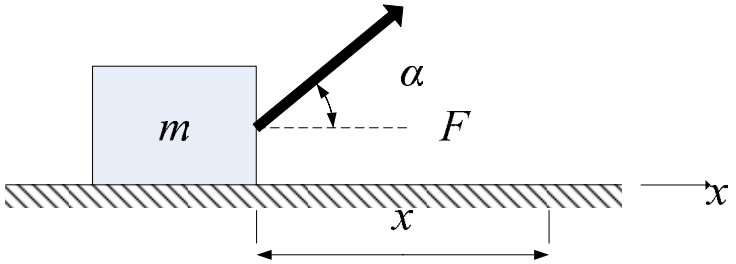


Figura A.2 Trabajo debido sólo a la componente horizontal de la fuerza.

Si α es el ángulo medido entre la dirección del desplazamiento *x* y la fuerza F, el valor del trabajo U es ahora:

$$U = (F\cos\alpha)\cdot x \tag{A.2}$$

El trabajo es una magnitud física escalar, obtenido del producto escalar de los vectores fuerza y posición. Su unidad de medida en el SI es $N\cdot m$ que se llama *julio* o *joule* (J).

Si una fuerza variable F está moviendo a un objeto a lo largo del eje *x* desde una posición inicial a otra final, ahora la expresión para calcular el trabajo es:

$$U = \int_{x_i}^{x_f} F_x dx \tag{A.3}$$

A.2 TRABAJO REALIZADO POR UN RESORTE

Un sistema físico común en el que la fuerza varía con la posición, es el de un cuerpo conectado a un resorte. Si el resorte, orientado en dirección del eje *x,* se deforma desde su configuración inicial, es decir se estira o se comprime (Figura A.3), por efecto de alguna fuerza externa sobre el resorte, instantáneamente actúa una fuerza producida por el resorte contra la fuerza externa, cuya magnitud es:

$$F = -kx \tag{A.4}$$

Donde *x* es la magnitud del desplazamiento del resorte desde su posición no deformada en *x = 0* y k una constante positiva, llamada *constante de fuerza del resorte*, que es una medida de la rigidez (dureza) del resorte. Esta ecuación se llama *ley de Hooke*, y es válida para pequeños desplazamientos, ya que si el resorte se estira demasiado, puede deformarse y no recuperar su forma original. El signo negativo indica que la dirección de esta fuerza es siempre opuesta al desplazamiento. Si el cuerpo se desplaza desde una posición inicial a la final, el trabajo realizado por el resorte es:

$$U_{1\to 2} = -\int_{X_1}^{X_2} kx\, dx = \frac{1}{2}kx_1^2 - \frac{1}{2}kx_2^2 \tag{A.5}$$

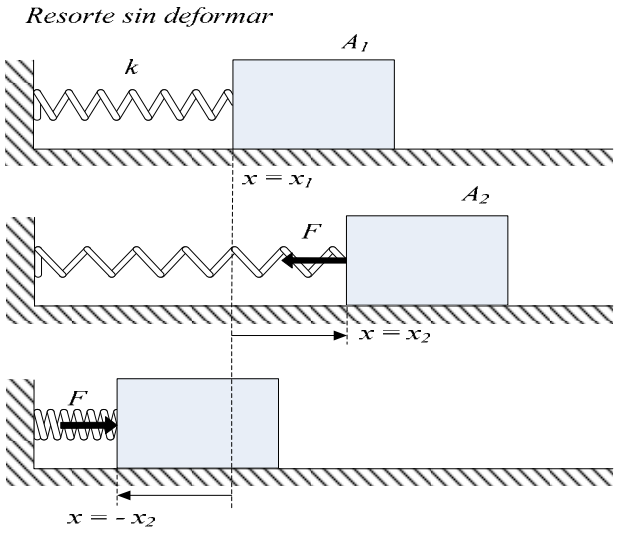


Figura A.3 Trabajo que realiza un resorte.

A.3 ENERGÍA POTENCIAL

Cuando el trabajo de la fuerza F es independiente de la trayectoria real que sigue la partícula desde la posición A_1 hasta A_2, entonces se dice que la fuerza es conservativa y, por tanto, su trabajo realizado se expresa de la siguiente forma

$$U_{1\to 2} = V_1 - V_2 \tag{A.6}$$

Donde V es la energía potencial asociada con F, V_1 y V_2 representan, respectivamente, los valores de V en las posiciones A_1 y A_2. Las energías potenciales asociadas con la fuerza F ejercida por un resorte se expresan como

$$V_e = \frac{1}{2}kx^2 \tag{A.7}$$

A.4 ENERGÍA CINÉTICA

Cuando una fuerza variable actúa sobre un cuerpo, le produce una aceleración durante su desplazamiento. El trabajo realizado por la fuerza para mover al cuerpo suponiendo la fuerza paralela al desplazamiento es:

$$U = \int_{s_1}^{s_2} \sum F \cdot ds \tag{A.8}$$

Por la segunda ley de Newton se tiene:

$$\sum F = m\frac{dv}{dt} \tag{A.9}$$

considerando

$$m\frac{dv}{dt} = m\frac{dv}{ds}\frac{ds}{dt} \tag{A.10}$$

sustituyendo en la ecuación A.5

$$\sum F = mv\frac{dv}{ds} \tag{A.11}$$

reemplazando en el trabajo total se tiene:

$$U = \int_{s_1}^{s_2} mv\frac{dv}{ds} \cdot ds \tag{A.12}$$

$$U = m\int_{v_2}^{v_2} v\,dv \tag{A.13}$$

$$U = \frac{1}{2}mv_2^2 - \frac{1}{2}mv_1^2 \tag{A.14}$$

El término $\frac{1}{2}mv^2$ se define como la energía cinética E_c del cuerpo.

Por lo tanto, el trabajo realizado por la fuerza resultante sobre una partícula es igual al cambio de energía cinética, enunciado que se conoce como el ***Teorema del Trabajo y la Energía***. Cuando la rapidez es constante, no hay variación de energía cinética y el trabajo de la fuerza neta es cero.

Dado que las componentes de fuerza perpendiculares a la trayectoria no realizan trabajo, el trabajo realizado por la componente de la fuerza tangencial a la trayectoria es

$$U = \int_{s_1}^{s_2} \sum F_t\,ds \tag{A.15}$$

Matemáticamente, el principio del trabajo y la energía se puede expresar como

$$\int_{s_1}^{s_2} \sum F_t\,ds = \frac{1}{2}mv_2^2 - \frac{1}{2}mv_1^2 \tag{A.16}$$

A.5 CONSERVACIÓN DE LA ENERGÍA MECÁNICA

Cuando una partícula se mueve por la acción de una fuerza conservativa, por el teorema del trabajo y la energía, se tiene que el trabajo realizado por la fuerza es igual a la variación de energía cinética de la partícula:

$$U = \Delta E_C \tag{A.17}$$

Pero como la fuerza es conservativa, entonces

$$U = -\Delta E_P \tag{A.18}$$

donde E_P puede ser la energía potencial gravitacional, elástica o cualquier otra forma de energía potencial mecánica. Igualando ambas expresiones del trabajo se obtiene:

$$\Delta E_C = -\Delta E_P \tag{A.19}$$

$$\Rightarrow \Delta E_C + \Delta E_P = 0$$

$$\Delta\left(E_C + E_P\right) = 0 \tag{A.20}$$

Esta ecuación se puede escribir también de la siguiente forma:

$$E_{Ci} + E_{Pi} = E_{Cf} + E_{Pf} \tag{A.21}$$

La ley de conservación de la energía mecánica establece que la energía mecánica total de un sistema permanece constante si las únicas fuerzas que realizan trabajo sobre el sistema son conservativas.

Si las fuerzas presentes en un sistema mecánico no son conservativas, como ocurre en los sistemas reales, la energía aparentemente no se conserva, porque se transforma en otro tipo de energía. Por ejemplo, la fuerza de fricción disipa energía, que se transforma en calor en la superficie de contacto entre los cuerpos. Se puede aplicar el teorema del trabajo y la energía tomando en cuenta la existencia de las fuerzas no conservativas.

Si U_{NC} es el trabajo sobre una partícula de todas las fuerzas no conservativas y U_C el trabajo de todas las fuerzas conservativas, entonces

$$U_{NC} + U_C = \Delta E_C \tag{A.22}$$

Como $U_C = -\Delta E_P$ entonces

$$U_{NC} = \Delta E_C + \Delta E_P$$

$$U_{NC} = \left(E_{Cf} - E_{Ci}\right) + \left(E_{Pf} - E_{Pi}\right) \tag{A.23}$$

$$U_{NC} = \left(E_{Cf} + E_{Pf}\right) - \left(E_{Ci} + E_{Pi}\right)$$

$$U_{NC} = E_f - E_i \tag{A.24}$$

Es decir, el trabajo realizado por todas las fuerzas no conservativas es igual al cambio de energía mecánica total del sistema.

La fuerza no conservativa de interés en el experimento que se realizará es la fuerza de fricción cinética, que se presenta cuando el cuerpo está en movimiento por efecto de la fuerza tangencial aplicada F (Figura 2.4). La fuerza de fricción se define como

$$f_k = \mu_k N \tag{A.25}$$

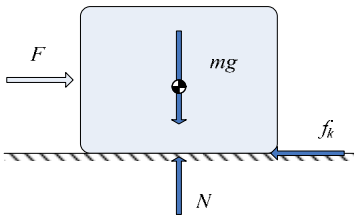


Figura A.4 Diagrama de cuerpo libre de un objeto en desplazamiento.

En la figura:

- F fuerza que provoca movimiento
- m masa del cuerpo
- g aceleración del campo gravitatorio del lugar
- N fuerza normal a la superficie de deslizamiento
- f_k fuerza de fricción cinética.

APÉNDICE B
PRINCIPIOS DE TRANSDUCTOR CON GALGAS

B.1 PUENTE DE WHEATSTONE

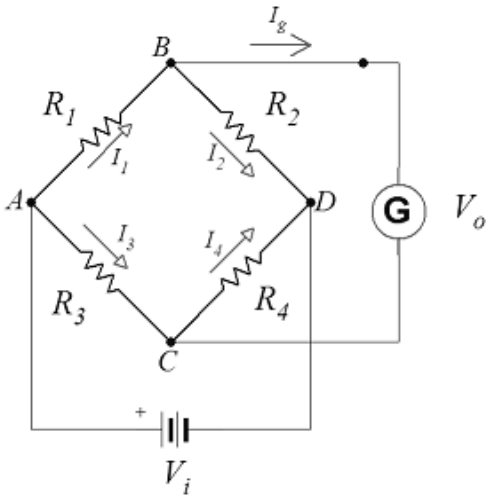


Figura B.1 Puente de Wheatstone resistivo.

El circuito resistivo de la Figura B.1 es conocido como puente de Wheatstone, *la condición de equilibrio* del puente se establece cuando el flujo de corriente por el galvanómetro medidor es cero, además se considera que éste tiene impedancia muy grande

$$I_g = 0 \qquad \text{(B.1)}$$

Dada esta condición, las corrientes en cada brazo del puente son iguales

$$I_1 = I_2 \quad \text{e} \quad I_3 = I_4 \tag{B.2}$$

o en función de las resistencias de cada brazo y tensión de alimentación

$$\begin{aligned} I_1 &= \frac{V_i}{R_1 + R_2} \\ I_3 &= \frac{V_i}{R_3 + R_4} \end{aligned} \tag{B.3}$$

Por lo tanto, la diferencia de potencial nula entre los puntos B y C se establece con las siguientes ecuaciones

$$\begin{aligned} 0 &= I_1 R_1 - I_3 R_3 \\ 0 &= I_2 R_2 - I_4 R_4 \end{aligned} \tag{B.4}$$

Considerando la ecuación B.2 y resolviendo simultáneamente las ecuaciones B.4 se establece la relación entre los elementos resistivos para un puente en equilibrio

$$\frac{R_2}{R_1} = \frac{R_4}{R_3} \tag{B.5}$$

conociendo la condición de equilibrio en la ecuación B.4, la tensión V_o es

$$V_o = I_1 R_1 - I_3 R_3 \tag{B.6}$$

sustituyendo las ecuaciones B.3 en B.6 y factorizando se obtiene la ecuación que representa la tensión de salida V_o en función de las resistencias y la tensión de alimentación

$$V_o = V_i \left(\frac{R_1}{R_1 + R_2} - \frac{R_3}{R_3 + R_4} \right) \tag{B.7}$$

En el caso de que los resistores sean galgas extensiométricas en estado inicial de cero deformación, V_o es igual a cero.

En condiciones de desequilibrio, la diferencia de potencial no nula entre los puntos B y C, es indicación directa del cambio de resistencia en uno o más brazos del puente. Si cada una de las galgas son sometidas a deformación de manera que las resistencias cambian dR_i donde $i=1,\ 2,\ 3\ y\ 4,$ entonces el cambio en la tensión de salida del puente se expresa como

$$dV_o = \sum_{i=1}^{4} \frac{\partial V_o}{\partial R_i} dR_i \tag{B.8}$$

evaluando las derivadas parciales

$$\begin{aligned}
\frac{\partial V_o}{\partial R_1} dR_1 &= V_i \frac{R_2 dR_1}{\left(R_1 + R_2\right)^2} \\
\frac{\partial V_o}{\partial R_2} dR_2 &= V_i \frac{-R_1 dR_2}{\left(R_1 + R_2\right)^2} \\
\frac{\partial V_o}{\partial R_3} dR_3 &= V_i \frac{-R_4 dR_3}{\left(R_3 + R_4\right)^2} \\
\frac{\partial V_o}{\partial R_4} dR_4 &= V_i \frac{R_3 dR_4}{\left(R_3 + R_4\right)^2}
\end{aligned} \tag{B.9}$$

obteniendo dV_o

$$dV_o = V_i \left[\frac{R_2 dR_1 - R_1 dR_2}{\left(R_1 + R_2\right)^2} + \frac{R_3 dR_4 - R_4 dR_3}{\left(R_3 + R_4\right)^2} \right] \tag{B.10}$$

Si el factor de galga es $GF = \dfrac{\Delta R / R}{\varepsilon}$ la variación de resistencia de cada elemento se pueden representar como

$$dR_i = R_i GF_i \varepsilon_i,\ \ i=1,\ 2,\ 3,\ 4 \tag{B.11}$$

donde ε_i es la deformación del i-ésimo elemento resistivo.

Sustituyendo la ecuación B.11 en B.10 se obtiene ΔV_o

$$\Delta V_o = V_i \left[\frac{R_2 R_1 GF_1 \varepsilon_1 - R_1 R_2 GF_2 \varepsilon_2}{\left(R_1 + R_2\right)^2} + \frac{R_3 R_4 GF_4 \varepsilon_4 - R_4 R_3 GF_3 \varepsilon_3}{\left(R_3 + R_4\right)^2} \right]$$

$$\Delta V_o = V_i \left[\frac{R_1 R_2}{\left(R_1 + R_2\right)^2} \left(\varepsilon_1 GF_1 - \varepsilon_2 GF_2\right) + \frac{R_3 R_4}{\left(R_3 + R_4\right)^2} \left(\varepsilon_4 GF_4 - \varepsilon_3 GF_3\right) \right] \qquad \text{(B.12)}$$

Para fines prácticos, suele utilizarse un mismo valor de resistencia R, por lo tanto

$$\frac{\Delta V_o}{V_i} = \frac{1}{4} \left(\varepsilon_1 GF_1 - \varepsilon_2 GF_2 + \varepsilon_4 GF_4 - \varepsilon_3 GF_3\right) \qquad \text{(B.13)}$$

Es común que los fabricantes de galgas ofrezcan conjuntos de galgas con iguales características como resistencia y factor de galga GF; bajo estas condiciones

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4} \left(\varepsilon_1 - \varepsilon_2 + \varepsilon_4 - \varepsilon_3\right) \qquad \text{(B.14)}$$

Se puede reescribir la ecuación B.14 en función de los elementos resistivos y su posible variación debida a esfuerzos

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4} \left(\frac{\Delta R_1}{R_1} - \frac{\Delta R_2}{R_2} + \frac{\Delta R_4}{R_4} - \frac{\Delta R_3}{R_3} \right) \qquad \text{(B.15)}$$

de tal forma que si no existe variación en el elemento R_i o ésta es poco significativa respecto a las demás, el factor $\frac{\Delta R_i}{R_i}$ se considera nulo, quedando únicamente los términos de la galga o galgas activas sometidas a esfuerzo.

B.2 CONCEPTOS DE LA VIGA EMPOTRADA

Se considera una viga empotrada de espesor h, ancho b y distancia L desde el punto de carga hasta el eje x, como se muestra en la Figura B.2**.**

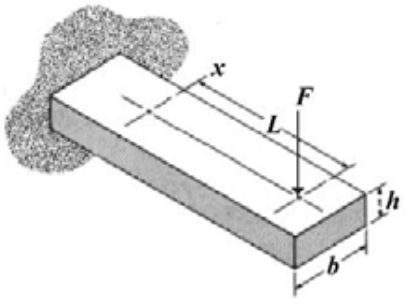


Figura B.2 Viga empotrada.

La deformación longitudinal sobre el eje x se establece como:

$$\varepsilon = \frac{\sigma_x}{E} \tag{B.16}$$

Donde E es el módulo de Young del material, y σ_x es el esfuerzo máximo por flexión sobre el eje x definido como:

$$\sigma = \frac{M}{S} \tag{B.17}$$

Donde M es el momento flexionante debido a la fuerza F, es decir

$$M = F \cdot L \tag{B.18}$$

Y además, S es el módulo elástico de sección definido como

$$S = \frac{I}{c} \tag{B.19}$$

Donde I es el momento de inercia de la sección transversal. Si la sección es rectangular, sólo en este caso $c = \frac{h}{2}$ y el momento de inercia es

$$I = \frac{bh^3}{12} \tag{B.20}$$

Finalmente la deformación máxima $\varepsilon_{máx}$ se puede establecer como:

$$\varepsilon_{máx} = \frac{6F_{máx} \cdot L}{Ebh^2} \tag{B.21}$$

Para fines de instrumentación o análisis de un material del cual se conoce su módulo de elasticidad, se mide o calcula la deformación para estimar la fuerza aplicada

$$F = \frac{Ebh^2}{6L}\varepsilon \quad . \tag{B.22}$$

B.3 TRANSDUCTORES DE VIGA EMPOTRADA

Para transductores de tipo empotrado, en configuración ¼ de puente de Wheatstone como el de la Figura B.3

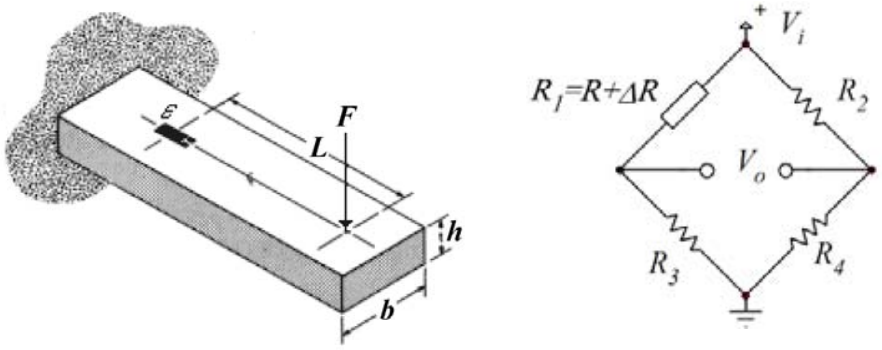


Figura B.3 Transductor ¼ de puente.

El cambio del voltaje de salida respecto al voltaje de polarización del puente es

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4}\varepsilon \tag{B.23}$$

Esta ecuación será válida siempre y cuando $\Delta R / R$ <<*1*, además, no se recomienda mucho su uso para mediciones que necesiten de exactitud, ya que es poco estable debido a la variación de resistencia por autocalentamiento y deformación transversal en la galga sin compensar.

Para transductores en configuración ½ de puente de Wheatstone como el de la Figura B.4

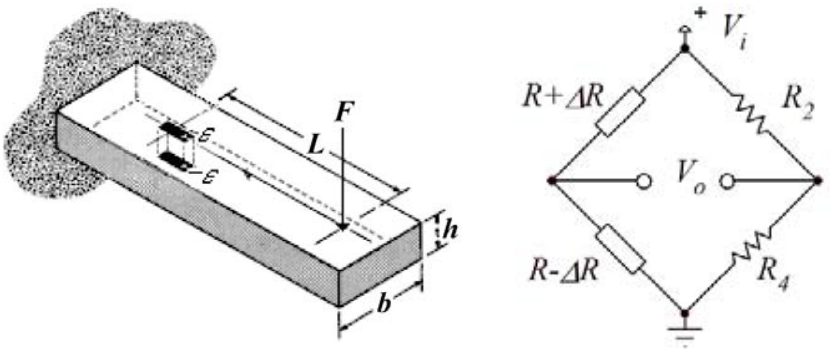


Figura B.4 Transductor ½ puente.

Si se considera que las galgas están alineadas, que la deformación que experimentan es la misma, una en tensión (positiva) y otra en compresión (negativa), y que se colocan en brazos adyacentes para incrementar la salida del puente, se tiene

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4}\left(\varepsilon_1 - \varepsilon_3\right)$$

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4}\left(\varepsilon_1 - \left(-\varepsilon_3\right)\right) \qquad \text{(B.24)}$$

$$\frac{\Delta V_o}{V_i} = \frac{GF}{2}\varepsilon$$

Para transductores en configuración de puente de Wheatstone completo como el de la Figura B.5

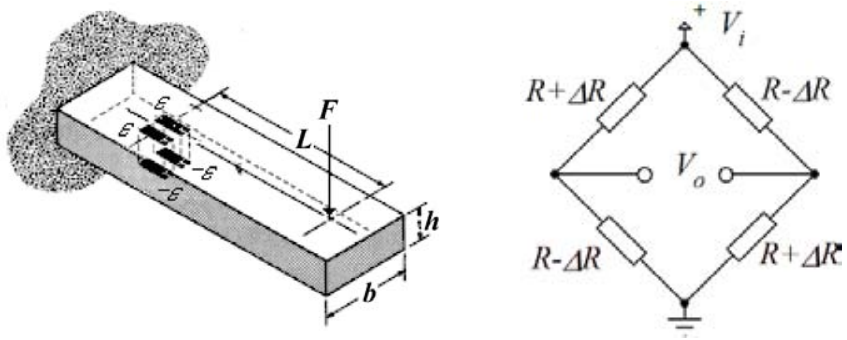


Figura B.5 Transductor de puente completo.

si se considera que las deformaciones de los brazos opuestos se suman y que las de brazos adyacentes son las mismas pero de signo contrario, según la ecuación B.14, para obtener la mayor señal de salida, se tiene

$$\frac{\Delta V_o}{V_i} = \frac{GF}{4}\left(\varepsilon_1 - (-\varepsilon_2) + \varepsilon_4 - (-\varepsilon_3)\right)$$

$$\frac{\Delta V_o}{V_i} = GF\varepsilon\,. \qquad \text{(B.25)}$$

APÉNDICE C
PROGRAMAS

PROGRAMA DEL PIC MAESTRO

```
/*
CONFIGURACIÓN DE PINES PUERTO B
      PIN_B0  SEÑAL DEL SWITCH MAGNÉTICO                              1
      PIN_B1  SEÑAL DEL SENSOR ÓPTICO REFLECTIVO INFERIOR             1
      PIN_B2  SEÑAL DEL SENSOR ÓPTICO REFLECTIVO SUPERIOR             1
      PIN_B4  BOTÓN 'TEST' SÓLO DESPUÉS DE UN RESET                   1
      PIN_B5  HABILITA DRIVER DE SOLENOIDE                            0
      PIN_B6  HABILITA DRIVER DE MOTOR C.D.                           0
      PIN_B7  PIN DE HABILITACIÓN DE DRIVER L298 MOTOR_PAP            0
*/

#include    <18F452.h>
      // palabra de configuración
#fuses      HS,NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP
      // reloj de 20 MHZ
#use        delay(clock = 20000000)
      // configuración de un puerto serie PC-PIC maestro
#use        rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=COM_A,
FORCE_SW)
      // configuración de un puerto serie PIC maestro-esclavo
#use        rs232(baud=9600, parity=N, xmit=PIN_C3, rcv=PIN_C4, stream=COM_B,
FORCE_SW)
      // Llamado a librería para conversión de tipos de Datos
#include    <stdlib.h>
      //PUERTO D EN DIRECCIÓN 0x3971
#BYTE       PORTD = 3971
      // Preprocesador para Reconocer entre mayúsculas y mínúsculas
```

```
      // ESTRUCTURA DEFINIDA PARA SELECCIONAR BITS PARA INTERFAZ CON MOTORES
      // Y SOLENOIDE

struct PORTD_BITS
   {
   BYTE motor_pap: 4;         // bits 0 - 3     PINES 0 AL 3 DEL PUERTO
                              //                D PARA -MOTOR A PASOS
   BYTE carro:     2;         // bit 4,5        PIN 4 Y 5 PARA CONTROL DEL
                              //                MOTOR DE DC
   BYTE Solenoide: 1;         // bit 6          PIN QUE ACTIVA EL SOLENOIDE
   BYTE un_bit:    1;         // bit 7          ---PIN SIN USAR---
   };

struct PORTD_BITS portd_bits;

//************* DEFINICIÓN DE FUNCIONES
void HOME (void);
void HOME_test(void);

//************* INICIA PROGRAMA PRINCIPAL

void main (void){

int16    a, i=0, mini_pasos=0, mini_dist=0;
int      j=0, dato_pap=0x99;  // SE INICIALIZA DATO_PAP=0x3 PARA ARRANQUE
                              // DEL MOTOR A PASOS

int   milimetros=0;
char     D,dato, string [10];       // RESERVA ESPACIO PARA CADENA QUE
                                    // CONTIENE LA DISTACIA A DESPLAZAR
char     string2[8];
float    pasos=0;
set_tris_d(0x00);                   // CONFIGURACIÓN PUERTO D COMO -SALIDAS-
set_tris_b(0xFF);                   // CONFIGURACIÓN PUERTO B

portd_bits=0x00;                    // INICIALIZA REGISTRO INTERNO
PORTD_BITS = 0
PORTD = portd_bits;                 // ASIGNACIÓN DEL VALOR DEL REG INTERNO
                                    // A PUERTO D
delay_ms(500);

      fprintf(COM_A, "K");
```

```
      for(;;){
//*****************************************************************************
         dato=getc();
                        switch (dato) {
                        case 'H':   HOME();     // HOME, llevar al sistema
                                                // a la posición inicial
                                    break;

//********** INGRESO DE DATO DE DISTANCIA A DESPLAZAR EN CARACTERIZACIÓN
                        case 'd' :  (gets(string)!=D );
                                    milimetros=atoi(string);
                                    pasos= 39.4 * milimetros;
//dividiendo la distancia total en 10 tramos
                                    mini_pasos=(pasos/10);
                                    mini_dist=mini_pasos*(200/7880);
// habilita driver de electroimán
                                   output_high(PIN_B5);
                                   portd_bits.Solenoide=0x01;
                                   PORTD=portd_bits;
// habilita driver de carro
                                    output_high(PIN_B6);
// recupera el móvil hacia el electroimán
                                    do{
                                       portd_bits.carro=2;
                                       PORTD=portd_bits;
                                       }while(!input(PIN_B2));
                                    do{
                                       portd_bits.carro=1;
                                       PORTD=portd_bits;
                                       }while(!input(PIN_B1));

                                    portd_bits.carro=0;
                                    PORTD=portd_bits;

                                    fprintf(COM_B, "C");
                                    delay_ms(2000);
                                    fprintf(COM_B, "I");
                                    delay_ms(500);
                                    for(j=1;j<=10; j++){
// habilitar driver de MOTOR DE PULSOS
                                    output_high(PIN_B7);

                                       for(i = 1; i <=mini_pasos; i++) {
```

```
// se manda la primer secuencia '0x3' al motor pap
                                        portd_bits.motor_pap = dato_pap;
// se lee la última salida o el dato que hay en el motor pap
                                        PORTD=portd_bits;
                                         delay_us(2700);
// se rota la secuencia
                                        rotate_right(&dato_pap, 1);
// se manda al motor el dato
                                        portd_bits.motor_pap = dato_pap;
                                       PORTD=portd_bits;
                                         delay_us (2700);  // retardo
                                         }
// termina secuencia de tramo recorrido
// deshabilita driver DE MOTOR A PASOS
                                    output_low(PIN_B7);
                                    delay_ms(200);
                                    fprintf(COM_B, "I");
                                    delay_ms(1000);
                                    }
                                    output_low (PIN_B7);
                                    fprintf(COM_A, "K");
                                    break;

//****************** INGRESO DE DATO DE DISTANCIA EN TRABAJO Y ENERGÍA
                        case 'D' :  (gets(string)!=D );
                                    milimetros=atoi(string);
                                    pasos= 39.4 * milimetros;
// habilita driver de electroimán
                                    output_high(PIN_B5);
                                    portd_bits.Solenoide=0x01;
                                    PORTD=portd_bits;
                                    output_high(PIN_B6);
// recupera el móvil hacia el electroimán
                                   do{
                                       portd_bits.carro=2;
                                       PORTD=portd_bits;
                                       }while(!input(PIN_B2));
                                    do{
                                       portd_bits.carro=1;
                                       PORTD=portd_bits;
                                       }while(!input(PIN_B1));

                                    portd_bits.carro=0;
```

```
                                    PORTD=portd_bits;


      for  (i = 1; i <= pasos; i++) {
      // habilitar driver
      output_high(PIN_B7);
      // se manda la primer secuencia '0x3' al motor pap
      portd_bits.motor_pap = dato_pap;
      // se lee la última salida o el dato que hay en el motor pap
      PORTD=portd_bits;
      delay_us(2700);
      rotate_right(&dato_pap, 1);         // se rota la secuencia
      portd_bits.motor_pap = dato_pap;    // se manda al motor el dato
      PORTD=portd_bits;
      delay_us (2700);                    // retardo
      }
      output_low (PIN_B7);                // deshabilita driver DE MOTOR A
                                          //  PASOS
      fprintf(COM_A, "K");                // carácter de reconocimiento
      break;

case 'O':  putc('K');
           break;


                                    // soltar móvil desenergizando
                                    electroimán
case 's':  output_high(PIN_B5);
           portd_bits.Solenoide=0x00;
           PORTD=portd_bits;
           delay_ms(1000);
           break;

case 'S':  fprintf(COM_B, "D");        // INICIAR LECTURAS DEL SENSOR FUERZA
           portd_bits.Solenoide=0x00; // Solamente desenergizar electroimán
           PORTD=portd_bits;
           output_low(PIN_B5);
           delay_ms(3000);
           break;

case 'W':  putc('w');
           fprintf(COM_B, "D");     // INICIAR LECTURAS DEL SENSOR FUERZA
           delay_ms(3000);
```

```
            break;

case 'C':   fprintf(COM_B, "C");     // INICIAR CALIBRADO DEL SENSOR
            break;
case 'R':   fprintf(COM_B, "R");           // RECEPCIÓN DE DATOS
            for(a=1; a<=600; a++){
            fgets(string2,COM_B);
            fprintf(COM_A,"%s\r",string2);
            }
            break;

case 'r':   fprintf(COM_B, "r");           // RECEPCIÓN DE DATOS
            for(a=1; a<=11; a++){
            fgets(string2,COM_B);
            fprintf(COM_A,"%s\r",string2);
            }
            break;

case 'B':   fprintf(COM_B, "B");           // BORRA MEMORIA DEL SENSOR
            break;

case 'x':   output_high(PIN_B5);
            portd_bits.Solenoide=0x01;
            PORTD=portd_bits;
            delay_ms(1000);
            break;

default:  break;
  }
 }
}
//*******FUNCION HOME
void HOME (void){
output_high(PIN_B7);                       // habilitar driver de motor PAP
do {
    portd_bits.motor_pap=0x3;
    PORTD=portd_bits;
    delay_ms(4);
    portd_bits.motor_pap=0x6;
    PORTD=portd_bits;
    delay_ms(4);
    portd_bits.motor_pap=0xC;
    PORTD=portd_bits;
    delay_ms(4);
```

```
    portd_bits.motor_pap=0X9;
    PORTD=portd_bits;
    delay_ms(4);
    }while (!input(PIN_B0));
output_low(PIN_B7);                 // deshabilitar driver del motor pap
// Posicionamiento de ajuste grueso
output_high(PIN_B5);                // habilita driver de Solenoide
portd_bits.Solenoide=0x01;          // energizar electroimán
PORTD=portd_bits;
output_high(PIN_B6);                // habilita driver de motor C.D
do{                                 // recupera el móvil hacia el
electroimán
   portd_bits.carro=2;
   PORTD=portd_bits;
   delay_ms(2000);
   portd_bits.carro=1;
   PORTD=portd_bits;
   delay_ms(1000);
   portd_bits.carro=2;
   PORTD=portd_bits;
   delay_ms(2000);
  }while(!input(PIN_B2));

do{                                 //  Regresando carro
   portd_bits.carro=1;
   PORTD=portd_bits;
   }while(!input(PIN_B1));
portd_bits.carro=0;
PORTD=portd_bits;
portd_bits.Solenoide=0x00;
output_low(PIN_B5);                 // deshabilita driver de Solenoide
output_low(PIN_B6);                 // deshabilita driver de motor C.D
putc('K');
}
//*****FUNCIÓN PARA PRUEBA MANUAL
void HOME_test (void){
output_high(PIN_B7);                // habilitar driver
do {
portd_bits.motor_pap=0x3;
PORTD=portd_bits;
delay_ms(4);
portd_bits.motor_pap=0x6;
PORTD=portd_bits;
delay_ms(4);
```

```
portd_bits.motor_pap=0xC;
PORTD=portd_bits;
delay_ms(4);
portd_bits.motor_pap=0X9;
PORTD=portd_bits;
delay_ms(4);
}while (input(PIN_B4));// presionar push botton para detener motor de pulsos
output_low(PIN_B7);          // deshabilitar driver del motor pap
delay_ms(1000);
if (!input(PIN_B4));
output_high(PIN_B5);         // habilitacion medio puente H (Solenoide)
portd_bits.Solenoide=0x01;   // energizar electroimán
PORTD=portd_bits;
output_high(PIN_B6);         // habilitación medio puente (Motor dc)
do{                          // recupera el movil hacia el electroimán
portd_bits.carro=2;
PORTD=portd_bits;
}while(!input(PIN_B4));
//  printf("\r\n Regresando carro ");
do{
portd_bits.carro=1;
PORTD=portd_bits;
}while(input(PIN_B4));

portd_bits.carro=0;
PORTD=portd_bits;
}
```

PROGRAMA DEI PIC ESCLAVO

```
/*
LOS PINES PARA SUBIR Y BAJAR LA REFERENCIA PARA LA RESISTENCIA DIGITAL SON:
PIN_D7 SUBIR REFERENCIA
PIN_D6 BAJAR REFERENCIA

LEDS MONITORES DE PROCESOS:
PIN_B7 CAPTURA DE DATOS
PIN_B6 CALIBRADO
AMBOS ENCENDIDOS, ENVÍO DE DATOS
*/

#include    <18f452.h>
#device     adc=10           // convertidor analógico digital de 10 bits
// palabra de configuración
#fuses      HS,  NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP //,H4
```

```
#use        delay(clock = 20000000)       // reloj de 20 MHZ
/* USAR ESTE CODIGO PARA DEPURACIÓN EN UNA TARJETA DIFERENTE
#use rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=COM_A)
#use rs232(baud = 9600, parity = N, xmit = PIN_C4, rcv = PIN_C3,
stream=COM_B, FORCE_SW)
*/
#use rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7)
// índice para direccionar las localidades de memoria, se inicia en esta
// localidad
int16 i=0x1B00;
// variables que manejara el ADC, la primera para recabado de datos y la
//otra para calibrado
int16 value=0x0000,cal_cero=0x0000, buffer=0, a =0;

void sube_referencia(void);
void baja_referencia(void);


////////////////////////////////////////////////////////////////////////////

#int_ad     // Interrupción cuando termina la conversión A/D
void adc_logger(void){
// únicamente se lee el resultado de la conversión
value = read_adc(ADC_READ_ONLY);
// argumentos: i=localidad inicial, &value= apuntador del dato, 2=No. de
// bytes del dato a escribir
write_program_memory(i,&value,2);
i=i+2;      // direcciona siguientes dos bytes de memoria
            // inicia escritura de 16 bits en flash, incremento i en 2 para
            // no sobreescribir
}
////////////////////////////////////////////////////////////////////////////


//***********PROGRAMA PRINCIPAL

void main (void ){
int   k=0;
char  car=0;
// variable junto con las funciones para subir o bajar referencia
flota dato=0;
int16 j=0, dato_caract=0, dato_cal=0,prom=0;
setup_adc_ports( ALL_ANALOG );      // inicializa el convertidor
setup_adc( ADC_CLOCK_DIV_64 );
```

```
set_adc_channel(0);

            for(;;){

//                  CODIGO PARA DEPURACIÓN
//                  car=fgetc(COM_B);

                    car=getc();
                    switch (car) {

/////// PROCESO QUE REALIZA EL DATALOGGING
case 'D':   // habilita interrupción AD para datalogging
            enable_interrupts(INT_AD);
            enable_interrupts(GLOBAL); // y habilitación global de ints.
            i=0x1C00;               // se reinicia el índice de localidades
                                    // para una nueva recolección de datos
            while(i!=0x2200)        // 0x2200 es la última localidad para
                                    // almacenar datos
                                        {
            read_adc(ADC_START_ONLY );  // solamente iniciar conversión AD
            output_toggle(PIN_B7);      // enciende monitor, indica que se
                                        // realiza una conversión AD
            }                           // termina recabado de datos

            output_b(0);                // se limpia puerto B
            disable_interrupts(INT_AD); //  deshabilita interrupción AD y
            disable_interrupts(GLOBAL); // y habilitación global de ints.
            break;

// inicia ahora el envío de todos los datos hacia el maestro para que éste
// lo envíe al PC



//////// PROCESO DE CALIBRADO DEL INSTRUMENTO ///////////

case 'C':   output_high(PIN_B6);
            for(k=0;k<70;k++){      // establece el nivel más alto de
                                    // voltaje alcanzado para calibrar
            sube_referencia();      // se toman en cuenta 70 niveles para la
                                    // resistencia digital
            }
            do {
            read_adc(ADC_START_ONLY );
```

```
            delay_us(30);
            cal_cero=read_adc(ADC_READ_ONLY);
            delay_us(20);
            baja_referencia();      // disminuye de uno en uno el nivel
                                    // hasta el deseado
            delay_ms(10);
            }while(cal_cero>=0x0014); // se alcanza el nivel mínimo de 100
                                    // mV y se toma como cero
            output_low(PIN_B6);
            k=0;
            break;                  // termina calibrado

case 'R':   delay_ms(2);
            output_b(0X03);
            for(a=0x1C00; a<=0x2200; a=a+2){
            buffer = read_program_eeprom(a);
            dato= (buffer*0.0020928);
            printf("%2.4f \r",dato);
            output_b(0xC0);
            delay_ms(10);
            }
            printf("\r");
            output_b(0);
            break;

case 'r':   delay_ms(2);            //  ENVÍA DATOS DE CARACTERIZACIÓN
            output_b(0X03);
            for(a=0x1B00; a<=0x1B14; a=a+2){
            buffer = read_program_eeprom(a);
            dato= (buffer*0.0020928);
            printf("%2.4f \r",dato);
            output_b(0xC0);
            delay_ms(10);
            }
            printf("\r");
            output_b(0);
            break;

case 'I':   i=0x1B00;
            for(j=1; j<=10; j++){   // toma 10 lecturas
            output_toggle(PIN_B7);  // enciende monitor, indica que se
                                    // realiza una conversión AD
```

```
            dato_caract+= read_adc();
            }
            prom=dato_caract/10;          //  promedio
            i=i+k;
            write_program_memory(i,&prom,2);    // guarda en memoria
                                                // iniciando en 0xA00
            k=k+2;
            prom=0;
            dato_caract=0;
            if (i>=0x1B14)k=0;
            output_b(0);                        // se limpia puerto B
            break;

case 'B':   i=0x1B00;                           // Borrado de memoria
            for(j=0; j<=1024; j++){
            write_program_memory(i,0xFFFF,2);
            i=i+2;
            output_toggle(PIN_B7);
            }
            output_b(0);
            break;
default: break;
}     // termina switch
}     // termina ciclo for
}     // termina main


/////////// FUNCIONES AUXILIARES //////////
void sube_referencia(void){
output_high(PIN_D7);
delay_ms(1);            // retardo necesarios entre pulsos alto y bajo para
                        // cambiar valor de resistencia
output_low(PIN_D7);// en el PIN_D1 se aplica el pulso para aumentar la
                   // resistencia digital (aumentar la referencia de voltaje
delay_ms(1);
}

void baja_referencia(void){
output_high(PIN_D6);
delay_ms(1);      // retardo necesarios entre pulsos alto y bajo para
                  // cambiar valor de resistencia
output_low(PIN_D6);
delay_ms(1);
```

CÓDIGO DE LAS PÁGINAS DEL SITIO WEB

```
//::::::::::::::::::::::::::::::::::://
//::  PÁGINA DE INTRODUCCIÓN  ::://
//::::::::::::::::::::::::::::::::::://

//DIRECTIVAS
using System;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.IO;

public partial class _Default : System.Web.UI.Page
{

      protected void Page_Load(object sender, EventArgs e)
      {
      }

      //FLECHA CONTINUAR
      protected void bthFLECHA_Click
      (object sender, ImageClickEventArgs e)
      {

            //CREA DIRECTORIO DÓNDE SE GUARDARAN TODAS LAS LECTURAS
          Directory.CreateDirectory(@"C:\Trabajo_Energia\Lecturas\");

            //SI EXISTE BORRA EL ARCHIVO COMPRIMIDO
          if (File.Exists("c:\\Lecturas.zip"))
          {
          File.Delete("c:\\Lecturas.zip");
          }

      //REDIRECCIONA A LA SIGUIENTE PÁGINA DEFAULT2.ASPX
      Response.Redirect
      ("http://localhost:1472/Trabajo_Energia/Default2.aspx");



//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::://
//::    PRIMERA PARTE "CARACTERIZACIÓN DEL RESORTE"  ::://
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::://

//DIRECTIVAS
using System;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.Data.Common;
using System.Windows;
using System.Windows.Forms;
using System.Text.RegularExpressions;

public partial class Default2 : System.Web.UI.Page
{
    //ÍNDICE DEL NÚMERO DE REPETICIONES
    public static int j = 1;

    //VARIABLES PARA VERDADERO Y FALSO
    bool V = true, F = false;

    //VALORES MÍNIMO Y MÁXIMO DE DISTANCIA PERMITIDOS
    public decimal myMaxvalue = 250;
    public decimal myMinvalue = 5;

    //VALOR INICIAL DE DISTANCIA
    public decimal numero = 0;

      //ARCHIVO DÓNDE SE GUARDARÁN LAS LECTURAS DE LA PRIMERA PARTE
    string path = @"C:\Trabajo_Energia\Lecturas\parte1_" +
      j.ToString() + ".txt";

    protected void Page_Load(object sender, EventArgs e)
    {

        //SI EXISTE EL DIRECTORIO LO BORRA CON TODO SU CONTENIDO
        if (Directory.Exists(@"C:\Trabajo_Energia\Lecturas\"))
        {
            Directory.Delete(@"C:\Trabajo_Energia\Lecturas\", true);
        }
    }

    //CONFIGURACIÓN DEL PUERTO SERIAL
    System.IO.Ports.SerialPort serialPort = new
      System.IO.Ports.SerialPort("COM5", 9600);

    //HOME (Posición Inicial)
    protected void btnHOME_Click(object sender, EventArgs e)
    {

            //ABRIR PUERTO, ENVIAR 'H' Y CERRAR PUERTO
            serialPort.Open();
            serialPort.Write("H");
            serialPort.Close();

            //ESPERA A RECIBIR EL CARÁCTER 'K'...
            serialPort.Open();
            serialPort.ReadTo("K");
            serialPort.Close();

        //PERMITE VISUALIZAR LA SIGUIENTE INSTRUCCIÓN Y LOS CONTROLES
            PARA INGRESAR LA DISTANCIA
```

```
    lblPasol.Visible = V;
    txtDISTANCIA.Text = "";
    txtDISTANCIA.Visible = V;
    lblEjemplo.Visible = V;
    btnHOME.Visible = F;
}


  //FUNCIÓN NÚMERO
public bool IsNumber(string inputvalue)
{
    Regex isnumber = new Regex(@"^-?[0-9]+(\.?[0-9]+)?$");
    return isnumber.IsMatch(inputvalue);
}

//INTRODUCE DISTANCIA
protected void txtDISTANCIA_TextChanged
  (object sender, EventArgs e)
{
    //LLAMA A LA FUNCIÓN NÚMERO
    IsNumber("0" + txtDISTANCIA.Text);

    //PREGUNTA SI ES NÚMERO LA DISTANCIA INGRESADA
    if (IsNumber("0" + txtDISTANCIA.Text))
    {
        //CONVIERTE A DECIMAL
        numero = Convert.ToDecimal("0" + txtDISTANCIA.Text);

        //PREGUNTA SI ESTÁ DENTRO DEL RANGO
        if (myMinvalue <= numero && numero <= myMaxvalue)
        {
            //SI LO ESTÁ MUESTRA EL BOTÓN INICIO
            btnINICIO.Visible = V;
            btnINICIO.Enabled = V;
        }

        //SI NO ESTÁ DENTRO DEL RANGO MUESTRA MENSAJE DE ERROR Y
              OCULTA BOTÓN INICIO
        else
        {
            MessageBox.Show
              ("Debes ingresear un valor entre 5 y 250[mm]", "Error",
              MessageBoxButtons.OK,MessageBoxIcon.Error);
            txtDISTANCIA.Text = "";
            btnINICIO.Visible = F;
            btnINICIO.Enabled = F;
        }
    }
    //SI NO ES NÚMERO MUESTRA MENSAJE DE ERROR LIMPIA EL TEXTBOX
        Y OCULTA BOTÓN INICIO
    else
    {
        txtDISTANCIA.Text = "";
        MessageBox.Show
              ("Debes ingresear un valor numérico", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        btnINICIO.Enabled = F;
```

```
        btnINICIO.Visible = F;
    }
}

//INICIA DEFORMACIÓN DEL RESORTE
protected void btnINICIO_Click(object sender, EventArgs e)
{
    //LLAMA A LA FUNCIÓN NÚMERO
    IsNumber("0" + txtDISTANCIA.Text);

    //PREGUNTA SI ES NÚMERO
    if (IsNumber(txtDISTANCIA.Text))
    {
        //SI ES NÚMERO LO CONVIERTE A DECIMAL
        numero = Convert.ToDecimal("0" + txtDISTANCIA.Text);

        //PREGUNTA SI ESTÁ DENTRO DEL RANGO
        if (myMinvalue <= numero && numero <= myMaxvalue)
        {
           //SI LO ESTÁ ENVÍA 'd' Y LA DISTANCIA INGRESADA
                    EN EL CUADRO DE TEXTO

                serialPort.Open();
                serialPort.Write("d");
                serialPort.Close();
                TimeSpan delayerTimer = new TimeSpan(0, 0, 8);
                serialPort.Open();
                serialPort.Write(txtDISTANCIA.Text + '\u000D');
                serialPort.Close();
                TimeSpan delayTimer = new TimeSpan(0, 0, 8);

                //ESPERA A RECIBIR EL CARÁCTER 'K'...
                serialPort.Open();
                serialPort.ReadTo("K");
                serialPort.Close();

                //OCULTA LA INSTRUCCIÓN ANTERIOR Y APARECE EL
                    BOTÓN DE SOLTAR BLOQUE
                lblPaso1.Enabled = F;
                txtDISTANCIA.Enabled = F;
                lblEjemplo.Enabled = F;
                btnINICIO.Enabled = F;
                lblPaso2.Visible = V;
                btnSOLE.Visible = V;
        }

        //SI NO ESTÁ DENTRO DEL RANGO MUESTRA MENSAJE DE ERROR,
              LIMPIA EL TEXTBOX Y OCULTA BOTÓN INICIO
        else
        {
            txtDISTANCIA.Text = "";

              MessageBox.Show("Debes ingresear un valor entre 5 y
              250[mm]", "Error", MessageBoxButtons.OK,
              MessageBoxIcon.Error);
            btnINICIO.Visible = F;
        }
```

```
    //SI NO ES NÚMERO MUESTRA MENSAJE DE ERROR, LIMPIA EL TEXTBOX
        Y OCULTA BOTÓN INICIO
    else
    {
        txtDISTANCIA.Text = "";
        MessageBox.Show
        ("Debes ingresear un valor numérico", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        btnINICIO.Visible = F;


//LIBERAR BLOQUE
protected void btnSOLE_Click(object sender, EventArgs e)
{
        //ENVÍA 's'
        serialPort.Open();
        serialPort.Write("s");
        serialPort.Close();

        //ABRE EL ARCHIVO FUERZA PARA ESCRITURA, SÓLO PARA BORRAR
              CONTENIDO Y LO CIERRA
        StreamWriter hola = new
              StreamWriter
              (@"C:\Trabajo_Energia\sensor_fuerza\fuerza.txt");
        hola.Close();

        //PAUSA DEL PROGRAMA DE 5 SEGUNDOS PARA ESPERAR A QUE
              TERMINE EL PIC MAESTRO
        Int32 miliseconds_to_sleep0 = 5000;
        System.Threading.Thread.Sleep(miliseconds_to_sleep0);

    //OCULTA INSTRUCCIÓN ANTERIOR Y MUESTRA BOTÓN LECTURAS
    lblPaso2.Enabled = F;
    btnSOLE.Enabled = F;
    lblPaso3.Visible = V;
    btnLecturas.Visible = V;
}


  //MOSTRAR LECTURAS
protected void btnLecturas_Click(object sender, EventArgs e)
{

        // INICIA PROCESO PARA RECIBIR LECTURAS DEL SENSOR_FUERZA
        System.Diagnostics.Process proc = new
              System.Diagnostics.Process();
         proc.StartInfo.FileName =
         @"C:\Trabajo_Energia\sensor_fuerza\1\serial
         console\bin\Debug\serialc.exe";
        proc.StartInfo.CreateNoWindow = true;
        proc.Start();

        //PAUSA DE 4 SEGUNDOS PARA RECIBIR DATOS
```

```
    Int32 miliseconds_to_sleep1 = 4000;
    System.Threading.Thread.Sleep(miliseconds_to_sleep1);

    //CIERRA EL PROCESO DEL SENSOR_FUERZA
    proc.Kill();

    //PAUSA PARA ESPERAR A QUE CIERRE EL ARCHIVO FUERZA
    Int32 miliseconds = 500;
    System.Threading.Thread.Sleep(miliseconds);

    //CREAR EL ARCHIVO PARTE1_J QUE CONTENDRÁ LAS LECTURAS DE
          LA REPETICIÓN NO. J
    FileStream car = File.Create(path);
    car.Close();

//ABRE EL ARCHIVO PARA ESCRIBIR LOS TÍTULOS DE LA COLUMNAS Y
    LO CIERRA
StreamWriter lec = new StreamWriter(path);
lec.WriteLine("Distancia[mm]" + "\t" + "Fuerza[N]");
lec.Close();

//LIMPIA EL LISTBOX QUE CONTENDRÁ LAS LECTURAS PARA
    MOSTRARLAS AL USUARIO Y ESCRIBE TÍTULOS DE LAS COLUMNAS
listbox1.Items.Clear();
listbox1.Items.Add("Distancia[mm]" + "\t" + "Fuerza[N]");

//CONVIERTE EL TEXTO DEL TEXTBOX A TIPO DOUBLE
double d1 = Convert.ToDouble(txtDISTANCIA.Text);

//DIVIDE LA DISTANCIA ENTRE 10 (d2) PARA OBTENER LOS 10
    TRAMOS (d3) EN LOS QUE SE TOMARÁN LECTURAS
double d3 = 0, d2 = d1 / 10;


//CONSTANTE QUE SE MULTIPLICARÁ POR d2 PARA INCREMENTAR EL
    VALOR DEL TRAMO (d3)
int k = 0;

//ABRE EL ARCHIVO FUERZA PARA LECTURA
System.Net.WebClient Client = new System.Net.WebClient();
Stream strm = Client.OpenRead(@"
    C:\Trabajo_Energia\sensor_fuerza\fuerza.txt");
StreamReader sr = new StreamReader(strm);
string line;

//LEE LAS LÍNEAS DEL ARCHIVO FUERZA MIENTRAS QUE K<11, LAS
    AGREGA AL NUEVO ARCHIVO Y CIERRA EL ARCHIVO FUERZA
do
{
    line = sr.ReadLine();
    d3 = k * d2;
    //AGREGA EL VALOR DEL TRAMO (d3) DONDE SE TOMÓ LA LECTURA
          Y LA LÍNEA LEÍDA AL ARCHIVO PARTE1_J Y AL LISTBOX
    using (StreamWriter cara = File.AppendText(path))
    {
        cara.WriteLine("{0}\t\t{1}", d3.ToString(), line);
        listbox1.Items.Add(d3.ToString() + '\t' + line);
```

```
            d2 = d1 / 10;
            cara.Flush();
            cara.Close();
        }
        k = k + 1;
    }
    while (k < 11);
    strm.Close();

    //OCULTA INSTRUCCIÓN ANTERIOR
    lblPaso3.Enabled = F;
    btnLecturas.Enabled = F;

    //MUESTRA EL LISTBOX, EL PANEL QUE LO CONTIENE Y LA FLECHA DE
        CONTINUAR
    pnlDistancia.Visible = V;
    listbox1.Visible = V;
    btnFLECHA2.Visible = V;

    //PREGUNTA EL NÚMERO DE REPETICIÓN DE LA PARTE 1
    //SI NO HA PASADO DE 10 REPETICIONES PERMITE VER LA
        INSTRUCCIÓN DE REPETIR O CONTINUAR
    if (j <= 9)
    {
        btnREPETIR.Visible = V;
        lblPaso4.Visible = V;
        lblPaso5.Visible = V;
    }
    //SI YA LLEGO A DIEZ REPETICIONES, SE LE INDICA AL USUARIO Y
        SÓLO PERMITE CONTINUAR
    else
    {
        btnREPETIR.Visible = V;
        btnREPETIR.Enabled = F;
        lblPaso4.Text = "Haz repetido 10 veces";
        lblPaso4.Visible = V;
        lblPaso5.Text = "Click en la flecha para continuar";
        lblPaso5.Visible = V;
    }
}

//FLECHA CONTINUAR
protected void btnFLECHA2_Click(object sender,
  ImageClickEventArgs e)
{
    //REDIRECCIONA A LA PÁGINA DEFAULT3.ASPX
    Response.Redirect
        ("http://localhost:1472/Trabajo_Energia/Default3.aspx");
}

//BOTÓN REPETIR
protected void btnREPETIR_Click(object sender, EventArgs e)
{
    //SI SE PRESIONA INCREMENTA EL ÍNDICE DE REPETICIONES Y PONE
        LAS BANDERAS DE LOS BOTONES EN VERDADERO
    if (IsPostBack)
        j++;
```

```
        //REDIRECCIONA LA PÁGINA A DEFAULT2.ASPX, VOLVIENDO A LAS
            PROPIEDADES INICIALES DE LOS CONTROLES DE LA PÁGINA
        Response.Redirect
            ("http://localhost:1472/Trabajo_Energia/Default2.aspx");
    }
}

//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::://
//:SEGUNDA PARTE "REALIZACIÓN DE TRABAJO POR MEDIO DE LA ENERGÍA:::://
//::::::::::::::::::::ALMACENADA EN EL RESORTE ::::::::::::::::::::::://
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::://

//  DIRECTIVAS
using System;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.ComponentModel;
using System.Diagnostics;                  //Para abrir procesos en pc
using System.Web.UI.WebControls.Adapters;
using System.Windows;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using System.IO.Compression;
using System.Collections.Generic;
using System.Text;
using Ionic.Utils.Zip;                     //Para comprimir directorio

public partial class Default3 : System.Web.UI.Page
{
    //ÍNDICE DEL NÚMERO DE REPETICIONES
    public static int j = 1;

    //VARABLES PARA VERDADERO Y FALSO
    bool V = true, F = false;

    //VALORES MÍNIMO Y MÁXIMO DE DISTANCIA PERMITIDOS
    public decimal myMaxvalue = 250;
    public decimal myMinvalue = 100;

    //VALOR INICIAL DE DISTANCIA
    public decimal numero = 0;
    public decimal entero = 0;
    public decimal entero1 = 0;
    public decimal pos = 0;


    //ARCHIVO DÓNDE SE GUARDARÁN LAS LECTURAS DE LA SEGUNDA PARTE
    string pathd = @"C:\Trabajo_Energia\Lecturas\parte2_" + j.ToString() +
".txt";
```

```
protected void Page_Load(object sender, EventArgs e)
{
}

//CONFIGURACIÓN DEL PUESRTO SERIAL
System.IO.Ports.SerialPort serialPort = new
  System.IO.Ports.SerialPort("COM5", 9600);

//HOME
protected void btnHOME2_Click(object sender, EventArgs e)
{
        //ABRIR PUERTO, ENVIAR 'H' Y CERRAR PUERTO
        serialPort.Open();
        serialPort.Write("H");
        serialPort.Close();

        //ESPERA A RECIBIR EL CARÁCTER 'K'...
        serialPort.Open();
        serialPort.ReadTo("K");
        serialPort.Close();

    //PERMITE VISUALIZAR LA SIGUIENTE INSTRUCCIÓN Y LOS CONTROLES
        PARA INGRESAR LA DISTANCIA
    btnHOME2.Visible = F;
    lblPaso5.Visible = V;
    txtDISTANCIA2.Text = "";
    txtDISTANCIA2.Visible = V;
    lblEjemplo2.Visible = V;
 }

//FUNCIÓN NÚMERO
public bool IsNumber(string inputvalue)
{
    Regex isnumber = new Regex(@"^-?[0-9]+(\.?[0-9]+)?$");
    return isnumber.IsMatch(inputvalue);
}

//INTRODUCE DISTANCIA
protected void txtDISTANCIA2_TextChanged(object sender, EventArgs e)
{
    //LLAMA A LA FUNCIÓN NÚMERO
    if (IsNumber("0" + txtDISTANCIA2.Text))
    {
        //CONVIERTE A DECIMAL
        numero = Convert.ToDecimal(txtDISTANCIA2.Text);

        //PREGUNTA SI ESTÁ DENTRO DEL RANGO
        if (myMinvalue <= numero && numero <= myMaxvalue)
        {
            //SI LO ESTÁ MUESTRA EL BOTÓN INICIO
            btnINICIO2.Visible = V;
            btnINICIO2.Enabled = V;
        }
        //SI NO ESTÁ DENTRO DEL RANGO MUESTRA MENSAJE DE ERROR Y
              OCULTA EL BOTÓN INICIO
        else
        {
```

```
                MessageBox.Show
                  ("Debes ingresear un valor entre 100 y 250[mm]", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                    txtDISTANCIA2.Text = "";
                    btnINICIO2.Visible = F;
                    btnINICIO2.Enabled = F;
                }
            }

        //SI NO ES NÚMERO MUESTRA MENSAJE DE ERROR LIMPIA EL TEXTBOX Y
OCULTA BOTÓN INICIO
        else
        {
            btnINICIO2.Visible = F;
            btnINICIO2.Enabled = F;
            MessageBox.Show
            ("Debes ingresear un valor numérico", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
            txtDISTANCIA2.Text = "";
            lblEjemplo2.Visible = V;
        }
    }

    //INICIA DEFORMACIÓN DEL RESORTE
    protected void btnINICIO2_Click(object sender, EventArgs e)
    {
        //LLAMA A LA FUNCIÓN NÚMERO
        IsNumber("0" + txtDISTANCIA2.Text);

        //PREGUNTA SI ES NÚMERO
        if (IsNumber(txtDISTANCIA2.Text))
        {
            //SI ES NÚMERO LO CONVIERTE A DECIMAL
            numero = Convert.ToDecimal("0" + txtDISTANCIA2.Text);

            //PREGUNTA SI ESTÁ DENTRO DEL RANGO
            if (myMinvalue <= numero && numero <= myMaxvalue)
            {
                //SI LO ESTÁ ENVÍA 'D' Y LA DISTANCIA INGRESADA EN EL
                  CUADRO DE TEXTO

                    serialPort.Open();
                    serialPort.Write("D");
                    serialPort.Close();
                    TimeSpan delayerTimer = new TimeSpan(0, 0, 8);
                    serialPort.Open();
                    serialPort.Write(txtDISTANCIA2.Text + '\u000D');
                    serialPort.Close();
                    TimeSpan delayTimer = new TimeSpan(0, 0, 8);

                    //ESPERA A RECIBIR EL CARÁCTER 'K'...
                    serialPort.Open();
                    serialPort.ReadTo("K");
                    serialPort.Close();

                    //OCULTA LA INSTRUCCIÓN ANTERIOR Y APARECE EL
                        BOTÓN DE SOLTAR BLOQUE
```

```
                lblPaso5.Enabled = F;
                txtDISTANCIA2.Enabled = F;
                lblEjemplo2.Enabled = F;
                btnINICIO2.Enabled = F;
                lblPaso6.Visible = V;
                btnSOLE2.Visible = V;
        }
        //SI NO ESTÁ DENTRO DEL RANGO MUESTRA MENSAJE DE ERROR,
              LIMPIA EL TEXTBOX Y OCULTA BOTÓN INICIO
        else
        {
            btnINICIO2.Visible = F;
            txtDISTANCIA2.Text = "";
            MessageBox.Show
               ("Debes ingresear un valor entre 100 y 250[mm]", "Error",
               MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    //SI NO ES NÚMERO MUESTRA MENSAJE DE ERROR, LIMPIA EL TEXTBOX
         Y OCULTA BOTÓN INICIO
    else
    {
        btnINICIO2.Visible = F;
        txtDISTANCIA2.Text = "";
        MessageBox.Show
         ("Debes ingresar un valor numérico", "Error",
         MessageBoxButtons.OK, MessageBoxIcon.Error);
     }
}

//LIBERAR BLOQUE
protected void btnSOLE2_Click(object sender, EventArgs e)
{
        //ENCIENDE SONAR
        String datos;
        System.Diagnostics.ProcessStartInfo psi = new
              System.Diagnostics.ProcessStartInfo();
        psi.FileName = @"C:\Trabajo_Energia\sonar\pasco.exe";
        psi.Arguments = "25";
        psi.Arguments += " " + "3000";
        psi.WorkingDirectory = @"C:\Trabajo_Energia\sonar\";
        psi.UseShellExecute = F;
        psi.RedirectStandardOutput = V;
        System.Diagnostics.Process p =
              System.Diagnostics.Process.Start(psi);
        System.IO.StreamReader sOut = p.StandardOutput;

        //PAUSA DE 1 SEGUNDO PARA TOMAR LECTURAS Y ENVIAR 'S'
        Int32 miliseconds_sleep = 1000;
        System.Threading.Thread.Sleep(miliseconds_sleep);
        serialPort.Open();
        serialPort.Write("S");
        serialPort.Close();

        //LECTURAS DEL SONAR
        datos = sOut.ReadToEnd();
        datos = datos.Substring(121);
```

```
    //OCULTA INSTRUCCIÓN ANTERIOR Y MUESTRA BOTÓN LECTURAS
    lblPaso6.Enabled = F;
    btnSOLE2.Enabled = F;
    lblPaso7.Visible = V;
    btnLecturas2.Visible = V;
}

//MOSTRAR LECTURAS
protected void btnLecturas2_Click(object sender, EventArgs e)
{

        //CREA ARCHIVO PARTE2_J PARA GUARDAR LAS LECTURAS DE
              AMBOS SENSORES
        FileStream car2 = File.Create(pathd);
        car2.Close();

        //ABRE EL ARCHIVO FUERZA PARA ESCRIBIR EL TÍTULO DE
              COLUMNA Y LO CIERRA
        StreamWriter hola3 = new
              StreamWriter(@"C:\Trabajo_Energia\sensor_fuerza\
              fuerza.txt");
        hola3.WriteLine("\rFuerza[N]");
        hola3.Close();

        // INICIA PROCESO PARA RECIBIR LECTURAS DEL SENSOR_FUERZA
        System.Diagnostics.Process proc = new
              System.Diagnostics.Process();
        proc.StartInfo.FileName =
              @"C:\Trabajo_Energia\sensor_fuerza\2\serial
              console\bin\Debug\serialc.exe";
        proc.StartInfo.CreateNoWindow = true;
        proc.Start();

        //PAUSA DE 5 SEGUNDOS PARA RECIBIR DATOS
        Int32 miliseconds_to_sleep1 = 5000;
        System.Threading.Thread.Sleep(miliseconds_to_sleep1);

        //CIERRA EL PROCESO DEL SENSOR_FUERZA
        proc.Kill();

        //PAUSA PARA ESPERAR A QUE CIERRE EL ARCHIVO FUERZA
        Int32 miliseconds_to_sleep2 = 500;
        System.Threading.Thread.Sleep(miliseconds_to_sleep2);

    //ABRE EL ARCHIVO PARA BORRAR CONTENIDO Y LO CIERRA
    StreamWriter lec = new StreamWriter(pathd);
    lec.WriteLine("");
    lec.Close();

    //LIMPIA EL LISTBOX QUE CONTENDRÁ LAS LECTURAS PARA
        MOSTRARLAS AL USUARIO
    listbox2.Items.Clear();

    //LEER LECTURAS DEL ARCHIVO QUE GENERÓ EL PROCESO DEL SONAR
    System.Net.WebClient Client2 = new System.Net.WebClient();
    Stream strm2 =
```

```
    Client2.OpenRead
    (@"C:\Trabajo_Energia\sonar\lecturas.txt");
StreamReader sr2 = new StreamReader(strm2);
string line2;

//ÍNDICE PARA SABER QUE LÍNEA ESTÁ LEYENDO
int a = 0;

//LEE LAS LÍNEAS DEL ARCHIVO DEL SONAR MIENTRAS QUE LA LÍNEA
    NO ESTÉ VACÍA, LAS AGREGA AL NUEVO ARCHIVO Y CIERRA EL ARCHIVO
    DEL SONAR
do
{
    using (StreamWriter cara2 = File.AppendText(pathd))
    {
        line2 = sr2.ReadLine();
        if (line2 != null)
        {
            //PREGUNTA SI NO ES LA PRIMER LÍNEA
            //SI NO ES LA PRIMER LÍNEA LIMPIA LAS LECTURAS
                PARA OBTENER EL TIEMPO RELATIVO
            if (a != 0)
            {
                string[] split1 = Regex.Split(line2, ",",
                      RegexOptions.IgnoreCase);
                string[] split2 = Regex.Split(split1[0], ":",
                      RegexOptions.IgnoreCase);
                string[] split3 = split2[2].Split(new char[]
                      { '.' });
                //SI ES LA SEGUNDA LÍNEA OBTIENE EL VALOR DE
                      SEGUNDOS PARA RESTARLO EN LAS SIGUEINETES
                      LÍNEAS
                if (a == 1)
                {
                    entero = Convert.ToDecimal(split3[0]);
                    pos = Convert.ToDecimal(split2[2]);
                    decimal resta = pos - entero;
                    cara2.WriteLine("{0}\t\t{1}",
                       resta.ToString(), split1[1]);
                    listbox2.Items.Add(resta.ToString() +
                      '\t' + '\t' + split1[1]);
                    a = a + 1;
                }
                //SE RESTA EL VALOR DE SEGUNDOS DE LA SEGUNDA
                      LÍNEA A LAS SIGUIENTES LÍNEAS PARA OBTENER UN
                      TIEMPO RELATIVO
                else
                {
                    decimal resta1 =
                      Convert.ToDecimal(split2[2]) - entero;
                       cara2.WriteLine("{0}\t\t{1}",
                      resta1.ToString(), split1[1]);
                    listbox2.Items.Add(resta1.ToString() +
                      '\t' + '\t' + split1[1]);
                }
            }
            //SI ES LA PRIMER LÍNEA, SE TRATA DE LOS TÍTULOS,
```

```
            // LOS AGREGA AL NUEVO ARCHIVO Y AL LISTBOX
            else
            {
                string[] split = Regex.Split(line2, ",",
                      RegexOptions.IgnoreCase);
                cara2.WriteLine("{0}\t\t{1}", split[0],
                      split[1]);
                listbox2.Items.Add(split[0] + '\t' + '\t' +
                      split[1]);
                a = a + 1;
            }
        }
        cara2.Flush();
        cara2.Close();
    }
}
while (line2 != null);
strm2.Close();

//LEER ARCHIVO DE LECTURAS SENSOR FUERZA
System.Net.WebClient Client3 = new System.Net.WebClient();
Stream strm3 =
    Client3.OpenRead
    (@"C:\Trabajo_Energia\sensor_fuerza\fuerza.txt");
StreamReader sr3 = new StreamReader(strm3);
string line3;

//LEE LAS LÍNEAS DEL ARCHIVO FUERZA MIENTRAS NO ESTÁ VACÍO,
    LAS AGREGA AL NUEVO ARCHIVO Y CIERRA EL ARCHIVO FUERZA
do
{
    using (StreamWriter cara3 = File.AppendText(pathd))
    {
        line3 = sr3.ReadLine();
        cara3.WriteLine("{0}", line3);
        listbox2.Items.Add(line3);
        cara3.Flush();
        cara3.Close();

}
while (line3 != null);
strm3.Close();

pnlDistancia2.Visible = V;
listbox2.Visible = V;
btnFLECHA3.Visible = V;

//PREGUNTA EL NÚMERO DE REPETICIÓN DE LA PARTE 1
//SI NO HA PASADO DE 10 REPETICIONES PERMITE VER LA
    INSTRUCCIÓN DE REPETIR O CONTINUAR
if (j <= 9)
{
    btnRepetir2.Visible = V;
    lblContinuar.Visible = V;
    lblRecibir2.Visible = V;
}
//SI YA LLEGO A DIEZ REPETICIONES, SE LE INDICA AL USUARIO Y
```

```
            SÓLO PERMITE CONTINUAR
        else
        {
            btnRepetir2.Visible = V;
            btnRepetir2.Enabled = F;
            lblRecibir2.Text = "Haz repetido 10 veces";
            lblRecibir2.Visible = V;
            lblContinuar.Text = "Click para continuar";
            lblContinuar.Visible = V;
        }

        //OCULTA INSTRUCCIÓN ANTERIOR
        lblPaso7.Enabled = F;
        btnLecturas2.Enabled = F;
    }

    //FLECHA PARA CONTINUAR
    protected void btnFLECHA3_Click(object sender,
      ImageClickEventArgs e)
    {
        //COMPRIME CARPETA QUE CONTIENE LOS ARCHIVOS DE AMBAS PARTES
      DE LA PRÁCTICA
        using (ZipFile zip = new ZipFile("c:\\Lecturas.zip\\"))
        {
            zip.AddDirectory("c:\\Trabajo_Energia\\Lecturas\\");
            zip.Save();
        }

        //BORRA DIRECTORIO Y SU CONTENIDO, PARA UNA NUEVA SESIÓN
        Directory.Delete("c:\\Trabajo_Energia\\Lecturas\\",true);

        //REDIRECCIONA A LA PÁGINA DEFAULT4.ASPX

      Response.Redirect("http://localhost:1472/Trabajo_Energia/Default4.aspx
      ");
    }

    //BOTÓN REPETIR
    protected void btnRepetir2_Click(object sender, EventArgs e)
    {
        //SI SE PRESIONA INCREMENTA EL ÍNDICE DE REPETICIONES Y PONE
            LAS BANDERAS DE LOS BOTONES EN VERDADERO
        if (IsPostBack)
            j++;

        //REDIRECCIONA LA PÁGINA A DEFAULT3.ASPX, VOLVIENDO A LAS
            PROPIEDADES INICIALES DE LOS CONTROLES DE LA PÁGINA

      Response.Redirect("http://localhost:1472/Trabajo_Energia/Default3.aspx
      ");


//::::::::::::::::::::::::::::::::::::::::::::://
//::PÁGINA PARA DESCARGAR ARCHIVOS GENERADOS:://
//::::::::::::::::::::::::::::::::::::::::::::://
```

```
using System;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.IO;
using System.ComponentModel;
using System.Threading;
using System.Web.UI.WebControls.Adapters;
using System.Text.RegularExpressions;

public partial class Default4 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    //BOTÓN DESCARGAR
    protected void btnSalir_Click(object sender, EventArgs e)
    {
        //ARCHIVO A DESCARGAR
        string filepath = "c:\\Lecturas.zip";
        //OBTIENE PROPIEDADES DEL ARCHIVO A DESCARGAR
        FileInfo file = new FileInfo(filepath);
        //VERIFICA SI EL ARCHIVO EXISTE
        try
        {
            if (file.Exists)
            {
                // BORRAR EL CONTENIDO
                Response.ClearContent();
                 //AGREGAR EL NOMBRE AL ARCHIVO ADJUNTO Y
                  MUESTRA EL CUADRO DE DIALOGO
                Response.AddHeader("Content-Disposition",
                  "attachment; filename=" + file.Name);
                // AÑADIR EL TAMAÑO DEL ARCHIVO EN LA CABECERA DE LA
                  RESPUESTA
                  Response.AddHeader("Content-Length",
                  file.Length.ToString());
                //ESTBLACER EL TIPO DE CONTENIDO
                Response.ContentType = "text/plain";
                //ESCRIBIR EL ARCHIVO EMN LA RESPUESTA
                Response.WriteFile(file.FullName);
                //FIN DE LA RESPUESTA
                  Response.End();
            }
        }
        catch (Exception ex)
        {
            //CAPTURAR ERROR, SI LO HAY
            Response.Write("Error : " + ex.Message);
        }
    }
}

//::::::::::::::::::::::::::::::::::::::::::::::::::::://
//::PROCESO PARA RECIBIR LECTURAS DEL SENSOR FUERZA:://
```

```
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::://

#region Namespace Inclusions
using System;
using System.Timers;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.IO.Ports;
#endregion

namespace serial_console

{
    class SerialPortProgram
    {
         //CONFIGURACIÓN DEL PUERTO SERIAL
        private SerialPort port = new SerialPort("COM5", 9600,
            Parity.None, 8, StopBits.One);

        [STAThread]

        static void Main(string[] args)
        {
            new SerialPortProgram();
        }

        private SerialPortProgram()
        {
           //CONFIGURACIÒN DE TIMER
            System.Timers.Timer myTimer = new System.Timers.Timer();
            myTimer.Interval = 1;
            myTimer.Enabled = true;

            Console.WriteLine("Datos Recibidos:");
            //TAMAÑO DEL BUFFER
            int tambuf = 100000;
            port.ReadBufferSize = tambuf;

            //CADA QUE RECIBE UN DATO EL PUERTO, LLAMA A LA FUNCIÓN
              PARA ESCRIBIR LOS DATOS EN EL ARCHIVO
              port.DataReceived += new
              SerialDataReceivedEventHandler(port_DataReceived);

            //ABRE PUERTO
            port.Open();

              // Envía carácter para empezar el envío de datos desde el PIC
              EN LA SEGUNDA PARTE DE PRÁCTICA SE ENVÍA "R"
            port.WriteLine("r");


            // ESCRIBE EN LA CONSOLA LAS LECTURAS
            port.WriteLine(Console.ReadLine());
            Console.ReadLine();
        }
```

```
     private void port_DataReceived(object sender,
     SerialDataReceivedEventArgs e)
    {
       //ESCRIBE EN EL ARCHIVO FUERZA.TXT LAS LECTURAS RECIBIDAS
        TimeSpan elapsed = DateTime.Now - DateTime.MinValue;
                    string path =
        @"C:\Trabajo_Energia\sensor_fuerza\fuerza.txt";
        using (StreamWriter sw = File.AppendText(path))
        {
            string dato = port.ReadExisting();
            sw.Write("{0}", dato);
            sw.Flush();
        }
    }
}
}
```

APÉNDICE C
PROGRAMAS

PROGRAMA DEL PIC MAESTRO

```
/*
CONFIGURACIÓN DE PINES PUERTO B
      PIN_B0  SEÑAL DEL SWITCH MAGNÉTICO                          1
      PIN_B1  SEÑAL DEL SENSOR ÓPTICO REFLECTIVO INFERIOR         1
      PIN_B2  SEÑAL DEL SENSOR ÓPTICO REFLECTIVO SUPERIOR         1
      PIN_B4  BOTÓN 'TEST' SÓLO DESPUÉS DE UN RESET               1
      PIN_B5  HABILITA DRIVER DE SOLENOIDE                        0
      PIN_B6  HABILITA DRIVER DE MOTOR C.D.                       0
      PIN_B7  PIN DE HABILITACIÓN DE DRIVER L298 MOTOR_PAP        0
*/

#include    <18F452.h>
      // palabra de configuración
#fuses      HS,NOWDT, NOPROTECT, NOPUT, NOBROWNOUT, NOLVP
      // reloj de 20 MHZ
#use        delay(clock = 20000000)
      // configuración de un puerto serie PC-PIC maestro
#use        rs232(baud=9600, parity=N, xmit=PIN_C6, rcv=PIN_C7, stream=COM_A,
FORCE_SW)
      // configuración de un puerto serie PIC maestro-esclavo
#use        rs232(baud=9600, parity=N, xmit=PIN_C3, rcv=PIN_C4, stream=COM_B,
FORCE_SW)
      // Llamado a librería para conversión de tipos de Datos
#include    <stdlib.h>
      //PUERTO D EN DIRECCIÓN 0x3971
#BYTE       PORTD = 3971
      // Preprocesador para Reconocer entre mayúsculas y minúsculas
```

APÉNDICE C
MANEJO DE LAS LECTURAS

C.1 ECUACIONES EMPLEADAS

Modelo matemático de la ley de Hooke para resortes:

$$F_k = F_{0+}\, kx \tag{1}$$

Trabajo total desarrollado por el resorte:

$$U_k = \frac{1}{2}kx^2 + F_0 x \tag{2}$$

Rapidez del móvil empleando el principio de Trabajo y Energía:

$$v = \sqrt{\frac{\left(kx^2 + 2F_0 x\right)}{m} - 2\mu_d g x} \tag{3}$$

Rapidez del móvil:

$$v' = \sqrt{2\mu_d g\left(L - x\right)} \tag{4}$$

Coeficiente de fricción entre las superficies

$$\mu_d = \frac{\left(kx^2 + 2F_0 x\right)}{2Lgm} \tag{5}$$

C.2 INFORME DE PRÁCTICA REALIZADA MANUALMENTE

Tabla 1 Parejas de datos distancia, fuerza (x, F) para la caracterización manual del resorte.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
x_i	10	20	30	40	50	60	70	80	90	100	110	120	130
F_i	0.3	0.4	0.5	0.6	0.65	0.7	0.75	0.8	0.9	0.95	1	1.1	1.1

i	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	
x_i	140	150	160	170	180	190	200	210	220	230	240	250	
F_i	1.15	1.2	1.25	1.3	1.4	1.45	1.5	1.6	1.65	1.7	1.75	1.8	

Tabla 2 Distancia recorrida por el móvil al deformar el resorte 0.17 m.

L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
0.758	0.733	0.718	0.718	0.717	0.703	0.698	0.743	0.733	0.730

Distancia de deformación del resorte *d= 0.17 m.*

Con las parejas de valores (x_i, F_i) registrados en la Tabla 1, con el comando polyfit de Matlab se determinó el modelo matemático lineal (ec. 1) de la ley de Hooke para el resorte, y se dibujó en una misma gráfica, los datos experimentales obtenidos (con asterisco) y la ecuación de la recta de la ley de Hooke.

```
>> p= polyfit(data(1:25,1),data(1:25,2),1)
    0.3235
```

Donde *data* es el nombre del archivo que contiene los datos de la Tabla 1.

El resultado anterior implica que el modelo matemático del experimento queda:

$$\mathbf{F_k = 5.9731x + 0.3235}$$

Donde:
k = 5.9731 *N/m* es la constante de rigidez
F_0= 0.3235 *N* es la fuerza remanente del resorte calculada con el comando polyfit.

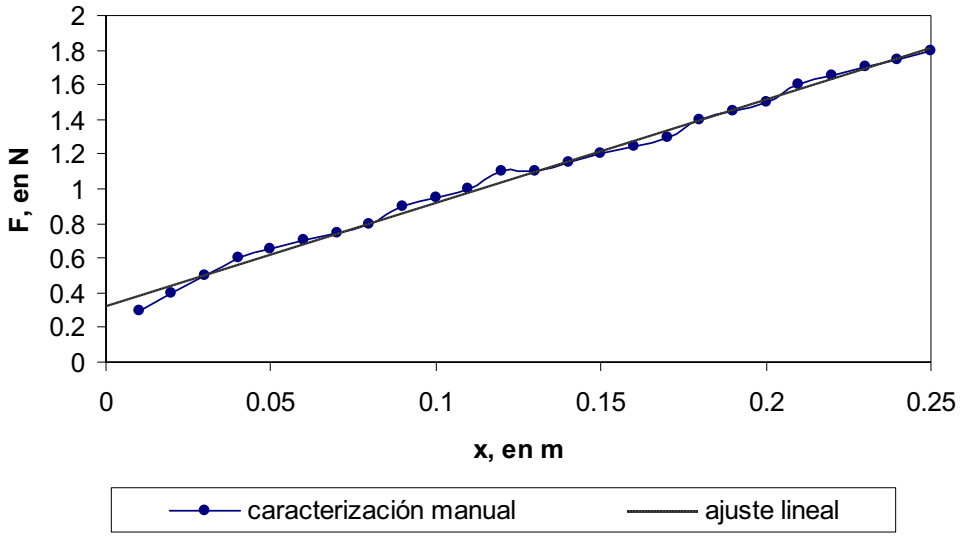


Figura C 1 Recta representativa de la caracterización del resorte.

Con el valor promedio de L (L_P), el cual deberá obtenerse de la Tabla 2, y los valores de k, F_0, d, g y m; obtener el valor numérico del coeficiente de fricción dinámica, dada por la ecuación 5:

L_P = 0.7251 *m*, m = 50 *g,* g = 9.78 m/s^2

$$\mu_d = \frac{\left((5.9731)(0.17)^2 + 2(0.3235)(0.17)\right)}{2(0.7251)(9.78)(50)}$$

μ_d = <u>0.398x10^{-3}</u>

Con el empleo de las ecuaciones 3 y 4 obtener la rapidez del bloque, con la ecuación 3:

$$v = \sqrt{\frac{\left(5.9731)(0.17)^2 + 2(0.3235)(0.17\right)}{50} - 2\mu_d(9.78)(0.17)}$$

v = <u>0.0742</u> m/s

y con la ecuación 4, considerando el mayor valor de L medido:

$$v' = \sqrt{2\mu_d (9.78)(0.758 - 0.17)}$$

v' = <u>0.0676</u> *m/s*

Obtener el porcentaje de diferencia entre los dos valores obtenidos en el punto anterior, a partir de la expresión:

$$\%D = \frac{|v - v'|}{v} \times 100$$

%D = <u>8.9</u> %

Calcular las pérdidas $U_\mu = U_k$ en el sistema mecánico debido al efecto de la fuerza de fricción:

$$U_k = \frac{1}{2} k d^2 + F_0 d$$

U_μ = <u>0.1413</u> *J*

Con el uso de Matlab, obtener la gráfica de la rapidez v del cuerpo en función de su posición x, teniendo como rango del dominio $0 \leq x \leq L$.

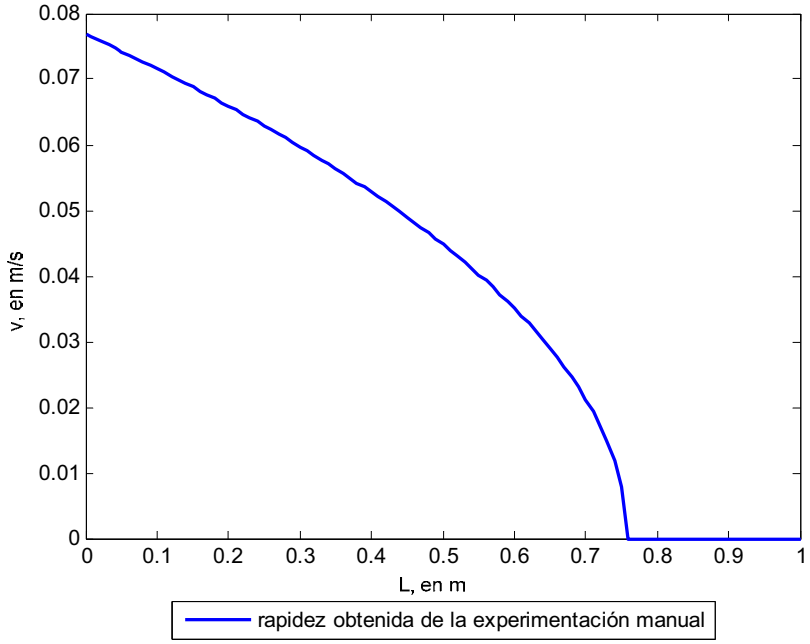


Figura C 2 Rapidez en función de la posición.

C.3 INFORME DE PRÁCTICA REALIZADA CON DISPOSITIVO MASA-RESORTE

Tabla 3 Parejas de datos distancia, fuerza (x, F) obtenidas en la caracterización del resorte, obtenidas con el sensor fuerza.

I	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
x_i	0	25	50	75	100	125	150	175	200	225	250
F_i	0	0.2888	0.4436	0.5755	0.7387	0.8726	1.0233	1.1615	1.3184	1.4419	1.5988

Tabla 4 Distancia recorrida por el móvil, obtenidas con el sonar.

L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
0.9729	0.9043	0.872	*0.892*	*0.8819*	*0.879*	*0.8916*	*0.9051*	*0.861*	*0.8856*

Distancia de deformación del resorte *d*= *0.17* *m.*

En Matlab:

```
>> polyfit(datos2(1:11,1),datos2(1:11,2),1)
   5.8318    0.1431
```

Por lo que el modelo matemático del experimento queda:

$$\mathbf{F_k = 5.8318x + 0.1431}$$

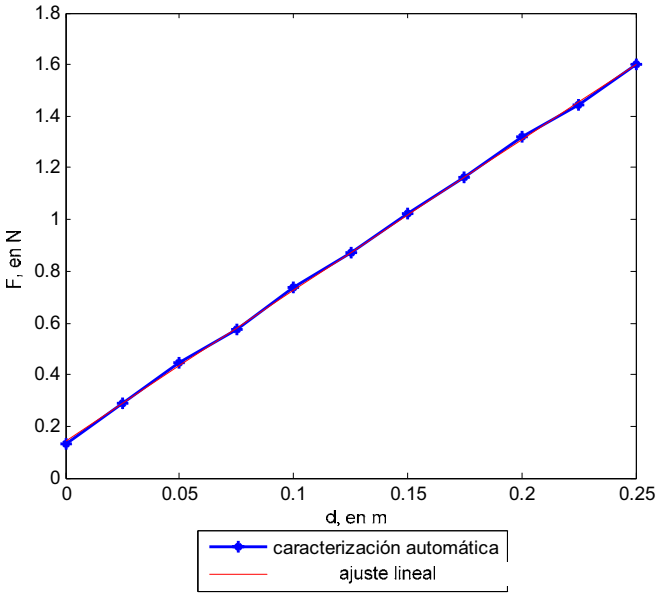


Figura C 3 Caracterización automática del resorte

Empleando las dos tablas anteriores y realizando el procedimiento de la práctica realizada manualmente se calculan los siguientes valores:

El promedio de la Tabla 4:

$$L_P = 0.8945$$

Coeficiente de fricción:

$$\mathbf{\mu_d = 0.248 \times 10^{-3}}$$

Rapidez obtenida con la ecuación 3:

$$\mathbf{v = 0.0598\ m/s}$$

Rapidez obtenida con la ecuación 4:

v' = 0.0621 m/s

Porcentaje de error

%D = <u>3.84</u> %

Pérdidas U_μ en el sistema mecánico debido al efecto de la fuerza de fricción:

U_μ = <u>0.1086</u> J

Gráfica de la rapidez v del cuerpo en función de su posición x, teniendo como rango del dominio $0 \leq x \leq L$

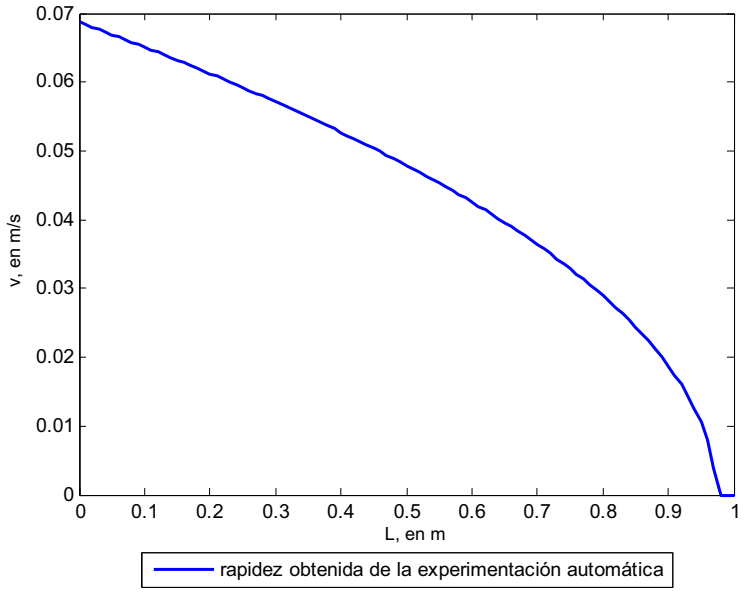


Figura C 4 Rapidez en función de la posición

REFERENCIAS

BIBLIOGRAFÍA

[1] **Barnett, Richard H., Cox Sarah, O'Cull Larry**
"Embedded C Programming and the Microchip PIC"
Ed. Thompson Delmar Learning, USA 2003.

[2] **Besteiro, Marco Antonio**
"Microsoft Visual C# .NET Referencia de Lenguaje"
Ed. McGraw Hill, 1ª edición, México 2002.

[3] **Cooper, William D. Albert D. Helfrick**
"*Instrumentación Electrónica Moderna y Técnicas de Medición*"
Ed. Prentice Hall, México 1951.

[4] **Deitel, Harvey M. y Paul J. Deitel**
"Cómo programar en C#"
Ed. Pearson Educación, 2ª edición, México 2007.

[5] **Figliola, Richard S., Beasley Donald E.**
"Mediciones Mecánicas. Teoría y diseño"
Ed. Alfaomega, 3ª edición, México 2003.

[6] **Gittleman, Art**
"C#.NET Iluminated"
Ed. Jones and Barlett Publishers, USA 2005.

[7] **Holman, Jack P.**
"Métodos experimentales para ingenieros"
Ed. McGraw Hill, 2ª edición, México 1986.

[8] **Monrroy Cano, Abraham Israel**
"Desarrollo de una interfaz USB para el control remoto de sistemas electromecánicos"
Tesis, UNAM Facultad de Ingeniería, México 2008.

[9] **Popov, Edgar P.**
"Introducción a la Mecánica de Sólidos"
Ed. Limusa, México 1976.

MESOGRAFÍA

[10] http://www.analog.com/static/imported-files/data_sheets/AD620.pdf
Hoja de datos del amplificador de instrumentación AD620.

[11] http://www.hbm.com/fileadmin/mediapool/hbmdoc/technical/s1569.pdf
Applying the Wheatstone Bridge Circuit.

[12] http://www.hbm.com/fileadmin/mediapool/hbmdoc/technical/s1421.pdf
Practical hints for the installation of strain gages.

[13] http://www.vishay.com/docs/11057/tn5071.pdf
Errors Due to Wheatstone Bridge Nonlinearity.

[14] http://cache.national.com/ds/LF/LF155.pdf
Hoja de datos del amplificador operacional LF355.

[15] http://www.byasa.com.mx
Empresa distribuidora de galgas extensiométricas .

[16] http://www.hbm.com
Empresa especializada en medición de fuerza. Información técnica.

[17] http://datasheets.maxim-ic.com/en/ds/DS1869.pdf
Hoja de datos de la resistencia digital DS1869.

[18] http://www.disensors.com/HTML/pdf/G%20Series.pdf
Hoja de datos de las galgas extensiométricas.

[19] http://www.ipsiamoretto.it/utenti/azzani/public_html/dati-tecnici-componenti/application/n_555.pdf
Aplicación del CI 555 como conversor de C.C. positiva a negativa.

[20] http://www.microsoft.com/express/download/
Descargas de Visual Studio, Consulta abril de 2007.

[21] http://www.elguille.info/NET/cursoCSharpErik/index.htm
Curso C#, Consulta septiembre de 2007.

[22] http://www.videnet.gatech.edu/cookbook.es/list_page.php?topic=2&url=remotelab.html&level=2&sequence=2.5&name=Laboratorios%20remotos
Laboratorios remotos, Consulta octubre de 2007.

[23] http://www.uag.mx/66/proceso2.htm
Tutorial de diseño de interfaz gráfica, Consulta noviembre de 2007.

[24] http://support.microsoft.com/
Ayuda y soporte de Microsoft, Consulta noviembre de 2007.

[25] http://www.lawebdelprogramador.com/
Comunidad de programadores, Consulta enero de 2008.

[26] http://es.csharp-online.net/
Programación en C#, Consulta marzo de 2008.

[27] http://www.java2s.com/Tutorial/CSharp/0300__File-Directory-Stream/Catalog0300__File-Directory-Stream.htm
Tutorial C#, Consulta septiembre de 2008.

[28] http://weblogs.asp.net/carloslone/archive/2008/02.aspx
Descarga de DLL para comprimir archivos, Consulta octubre 2008.

[29] http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm
Tiny PIC Bootloader.

[30] http://www.microchip.com
Fabricante de Microcontroladores PIC, información, notas técnicas.

[31] http://www.ccsinfo.com
Desarrollador del compilador C para PIC.

[32] http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf
Hoja de datos del microcontrolador PIC18F452.

[33] http://www.datasheetcatalog.org/datasheet/maxim/MAX220-MAX249.pdf
Hoja de datos del C.I. MAX233.

[34] http://en.wikipedia.org/wiki/RS-232
Descripción del protocolo RS-232.