Instituto Tecnológico de Tuxtla Gutiérrez

Estancia con motivos de Residencia Profesional

Centro de Investigaciones en Óptica, A.C.

Periodo de estancia: Del mes de Enero al mes de Mayo

Proyecto:

"Automatización y caracterización del proceso de adelgazamiento de fibras ópticas convencionales"

Alumno:

Emmanuel Castillejos Villatoro

Asesor interno:

M. en C. Raúl Moreno Rincón

Asesor externo:

Dr. David Monzón Hernández

INDICE

INTRODUCCION

Las fibras ópticas son dispositivos fotónicos bien conocidos con cualidades extraordinarias para las telecomunicaciones. Pero hasta ahora había sido poco explotado su potencial como elemento transductor en sistemas de monitoreo de parámetros físicos, químicos o biológicos. Los sistemas de fibra óptica para sensar o medir tienen ventajas respecto a los sistemas tradicionales; son inmunes al ruido electromagnético ambiental, pueden ser utilizados sin ningún riesgo en ambientes corrosivos o en presencia de sustancias flamables y explosivas, se pueden construir sistemas con muchos sensores para monitorear distintos puntos interrogados al mismo tiempo de manera multiplexada y telemétrica.

En la actualidad se pueden encontrar sistemas comerciales de fibra óptica para medir parámetros como: el voltaje y corriente en líneas de alta tensión, sistemas de orientación y posición en aeronaves, monitoreo estructural en puentes o edificios, monitoreo de sustancias químicas, entre otros.

Existen ya algunos procedimientos exitosos que hacen sensibles a las fibras ópticas, es decir que nos permiten modificar alguno o algunos de los parámetros de la luz, simplemente perturbando la fibra óptica. Algunos procedimientos que se han explotado durante los últimos diez años para sensibilizar una fibra óptica consisten en cambiar su geometría para generar una zona en la que se puede interactuar con la luz y modificar sus parámetros. El procedimiento más sencillo y repetible, es el adelgazamiento o estrechamiento de la fibra óptica, que consiste en calentar una pequeña zona hasta suavizar el vidrio y al mismo tiempo tirar de los extremos de manera delicada y controlada.

JUSTIFICACION

Una fibra óptica convencional no es sensible a perturbaciones externas, es necesario hacer modificaciones en su estructura para sensibilizarla. Uno de estos procesos consiste en hacer un estrechamiento de la fibra óptica, bajo condiciones muy precisas, para inducirle cierta sensibilidad. El trabajo consiste en automatizar el proceso de estrechamiento de las fibras ópticas, caracterizar el procedimiento y encontrar una técnica para medir de manera automática el perfil del adelgazamiento o el perfil del diámetro de la fibra óptica y del mismo modo realizar pruebas del funcionamiento de todos los procesos.

OBJETIVOS

Objetivos Generales

1. Discutir los fundamentos teóricos y el procedimiento experimental para estrechar una fibra óptica de vidrio.
2. Desarrollar un procedimiento de estrechado de una fibra óptica monomodo estándar para modificar las propiedades de propagación de la luz, en particular la dependencia de la longitud de onda.
3. Caracterizar experimentalmente la respuesta de la fibra óptica monomodo que ha sido estrechada.

Objetivos Específicos

1. Poner en marcha el proceso automatizado de estrechamiento de fibras ópticas.
2. Probar el sistema de medición del diámetro. Utilizar otro método de medición para comprobar los resultados pero para también descubrir las ventajas y desventajas del método que se está proponiendo.
3. Realizar los experimentos debidos en la mezcladora de gases para verificar mejorías en el desarrollo de sensores.
4. Elaborar un manual de uso del equipo desarrollado.

CARACTERIZACION DEL AREA EN QUE PARTICIPO

En el Centro de Investigaciones en Óptica, fundado en Abril de 1980, el departamento de Fibras Ópticas está conformado por 10 investigadores y 8 estudiantes de postgrado que trabajan en tres líneas de investigación principalmente, fabricación de fibras ópticas micro-estructuradas (FOM), construcción de láseres y fabricación de dispositivos y sensores de fibra óptica.

El objetivo del Departamento es: diseñar y fabricar fibras de cristal fotonico , estudiar las propiedades ópticas espectrales y no lineales de las fibras de cristal fotonico y de las fibras dopadas con tierras raras, así como diseñar y estudiar varios tipos de láseres de fibra, sensores de fibra y dispositivos basados en fibras. Los trabajos de investigación realizados por los científicos del Departamento son apoyados por varias fundaciones del gobierno local y federal, por la Comunidad Europea y por empresas privadas. Los mejores resultados están cubiertos por patentes mexicanas, americanas y europeas.

La mayoría de los artículos basados en los resultados teóricos y experimentales obtenidos en el Departamento se publican en las revistas internacionales más importantes en los campos de fisica y optica. El Departamento tiene colaboración con diversas universidades e institutos de investigación mexicanos y extranjeros, tales como la Universidad Nacional Autónoma de México (Ciudad de México), el Institut de Ciències Fotòniques (Barcelona), la Universidad de Valencia (Valencia), el "A.M. Prokhorov General Physics Institute" (Moscú), el "Imperial College" (Londres), y otros similares.

PROBLEMAS A RESOLVER

1. Automatizar el proceso de adelgazamiento de fibras monomodo.

2. Encontrar un método de medición de diámetro de la fibra estrechada.
3. Probar las fibras estrechadas en forma de sensor de gases de hidrogeno.

ALCANCES Y LIMITACIONES

Se llego a entregar los diseños de la primera y segunda fase de la residencia, por razones de tiempo y de adquisición de los materiales.

Para las pruebas de diseño y caracterizacion de la fibra se realizaban los días viernes o sábados con previo aviso al Dr. David Monzón para la petición de llaves.

Se pospuso la tercera fase de la residencia por la falta de la mezcladora de gases que se encontraba fuera de servicio.

FUNDAMENTO TEORICO

El procedimiento de estrechamiento de una fibra óptica es conocido y se ha utilizado durante algunos años en la fabricación de dispositivos de fibra óptica, el más reconocido es el acoplador de fibra óptica que tiene la misma función que un divisor de haz convencional. Actualmente existen tres técnicas reconocidas y empleadas para estrechar fibras ópticas de vidrio, básicamente, consisten en calentar una pequeña sección de la fibra mientras se tira de los extremos de una manera suave y controlada. La diferencia fundamental en los tres métodos es la fuente que produce el calentamiento para suavizar el vidrio de la fibra óptica. Las tres fuentes utilizadas son el láser de CO_2, una flama producto de la mezcla de dos o más gases y la una descarga eléctrica entre dos electrodos. Cada fuente imprime características particulares al procedimiento y en consecuencia también en las características de propagación de la fibra óptica estrechada.

En nuestro caso utilizaremos un sistema semiautomático para estrechar las fibras ópticas, que utiliza la flama de un micro soplete generada por la combustión de butano y oxígeno. Las características de la máquina permiten hacer estrechamientos en los que las características de la luz transmitidas por la fibra óptica no se vean modificadas por el proceso. Una de nuestras primeras labores será la de establecer las condiciones de estrechamiento que nos permiten generar fibras adelgazadas manteniendo las pérdidas de potencia transmitida lo más baja posible. El procedimiento de fabricación debe incluir un monitoreo continuo de la potencia transmitida. Para esto implementaremos un arreglo sencillo para medir en todo momento la potencia que trasmite la fibra óptica durante el procedimiento de estrechado. Realmente es muy difícil establecer de manera teórica las condiciones óptimas de fabricación, por lo que este procedimiento es en su mayor parte un procedimiento de prueba y error.

PROCEDIMIENTO Y DESCRIPCION DE LAS ACTIVIDADES REALIZADAS

Para comprender el manejo de las fibras ópticas y sus propiedades fue sugerido tomar cursos impartidos en el centro de investigación con la participación de los siguientes doctores; el Dr. Olivier Pottiez y el Dr. Rubén Grajales Coutiño del departamento de Fibras Ópticas, quienes en los 4 meses de estancia aclararon dudas y enseñaron el buen manejo de la fibra óptica además de ayudar en los cálculos necesarios para usar las fibras ópticas como sensores.

Tomando en cuenta que también se tendría la posibilidad de calcular el diámetro de la fibra estrechada por decisión propia tome el curso de “Procesamiento de imágenes por MatLab” impartido por el Dr. Manuel Ibarra de la Torre del departamento de Metrologia Óptica.

En los momentos libres del Dr. David Monzón se realizaban pruebas con el actual montaje para estrechamiento de las fibras monomodo, el cual funciona del siguiente modo:

Características del sistema de estrechamiento de fibras ópticas

El montaje consta de tres partes diferenciadas: Sistema mecánico, electrónica control de potencia y sistema de control de gases.

a) Sistema mecánico (Figura 1)

m1, m2: Motores que realizan el estiramiento de la fibra.

r1, r2: Carros que sujetan la fibra y son desplazados por m1 y m2.

mq3: Motor que desplaza el quemador.

q: Quemador (Temperatura alrededor de los 800 ~ 1200 ‘C)

i1, i2: Interruptores que hacen cambiar el sentido de giro de mq3, y por tanto el sentido del barrido de la llama.

s1, s2, s3 y s4: Interruptores de seguridad. Cuando uno de ellos es accionado bien por m1 o por m2, la alimentación del motor correspondiente queda interrumpida, pasando a un estado de INACTIVIDAD.

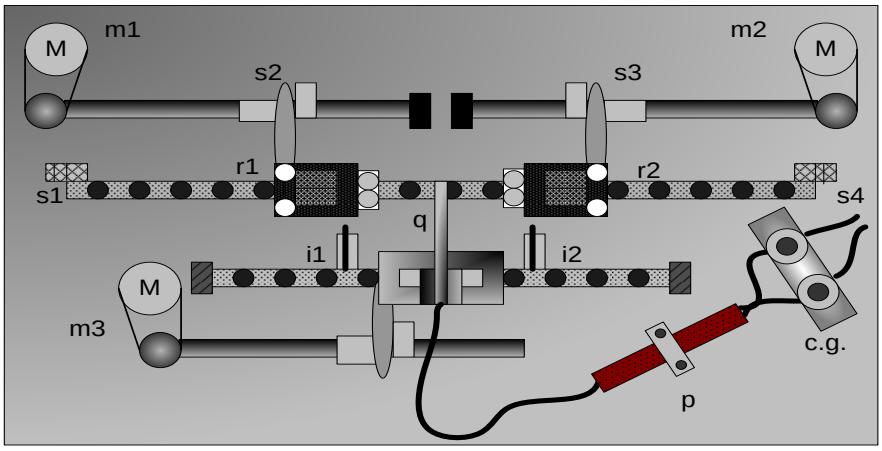


Figura 1. Sistema Mecánico

b) Electrónica control de potencia (Figura 2)

p1: Potenciómetro que permite variar la velocidad de estirado (Vc). La velocidad de retorno de carro no es controlable y esta ajustada a su valor máximo.

p2: Potenciómetro que permite variar la velocidad del quemador (Vq).

v1: Voltímetro que permite la monitorización de Vc.

v2: Voltímetro que permite la monitorización de Vq.

i1: Interruptor para actividad o desactivar los motores de estirado.

i2: Interruptor para activar o desactivar el quemador.

c1, c2: Conmutadores que permiten cambiar el sentido de giro de m1 y m2 (i= izquierda, p=parado, d=derecha). Cada uno de ellos corresponde a un motor, lo que nos permite controlar ambos motores de forma independiente.

s1i, s2d: Interruptores que nos permiten sacar a un motor que esta en estado de INACTIVIDAD. Estos interruptores han de estar SIEMPRE en off para que los interruptores de seguridad del montaje hagan su papel. SOLO se activaran (on) cuando necesitemos sacar alguno de los carros del estado de INACTIVIDAD.

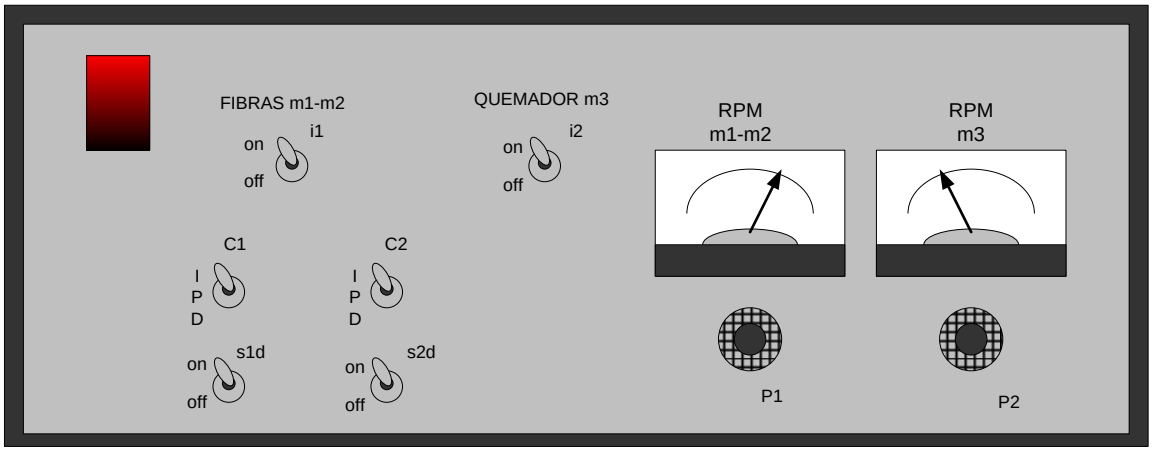


Figura 2. Panel control de potencia

c) Sistema de control de gases (Figura 1)

Permite regular el flujo de los gases (oxigeno y butano) y la proporción de cada uno de ellos en la combustión, para que la llama alcance las condiciones necesarias. Las botellas de gases deben incorporar reguladores de presión de salida. Las presiones de trabajo son de 0.3 ~ 0.5 bar para el oxigeno y 0.8 ~ 1 bar para el butano.

c.g.: Reguladores de flujo.

p: Llaves de paso.

Modo de operación

1. Posicionar los carros r1, r2 en la posición inicial. Resulta conveniente que los carros queden equidistantes respecto del centro (las reglas milimetradas están colocadas a una misma distancia del centro, pudiéndose utilizar como referencia) y separados una distancia suficiente para que el quemador no colisione con ellos. Por la asimetría de los motores m1 y m2 (uno gira en un sentido y el otro en el contrario), la velocidad de retorno de los carros r1 y r2 no es idéntica, siendo mas lento r1. Para colocar los carros en su posición inicial hay que hacer uso de los conmutadores c1 y c2 de la caja de electrónica potencia de control.

2. Fijar la amplitud de oscilación mediante el posicionamiento de i1 y i2. La amplitud de oscilación se corresponderá con la longitud del cuello del taper que deseamos conseguir en nuestro caso fibras estrechadas (consultar bibliografía). Resulta conveniente que el recorrido del quemador quede centrado respecto a los carros r1, r2.

3. Ajustar las velocidades de estiramiento (Ve ~ 2-3 mm/min) y del quemador (Vq ~ 2-3 mm/s). La figura 3 proporciona las curvas de calibración de velocidades (calculadas durante la estancia).

4. Encendido de la llama, que debe ser azul y de unos 2~3 mm de altura. Ajustar las presiones de salida de los gases a los valores especificados anteriormente (P02 ~ 0.5 bar, Pbut ~ 1 bar) y los reguladores de flujo en la posición ~25 para el O2 y ~95 para el butano. Abrir primero la llave de paso del butano y encender la llama. Seguidamente, abrir la llave de paso del O2 y esperar que la llama se estabilice y alcance las condiciones adecuadas. Una llama que no caliente suficiente hará que la fibra se nos rompa durante el

estiramiento y una llama excesivamente caliente nos introducirá deformaciones no deseadas que se traducen en altas perdidas.

5. Quitar el polímero protector de una sección de fibra y limpiar minuciosamente esa sección con alcohol o acetona. Colocar la fibra sobre los carritos r1, r2 y fijarla a estos utilizando las pinzas cromadas. La fibra debe quedar tensa y en el centro de los carritos r1, r2 (Los cilindros plateados situados sobre la base negra de los carritos resultan útiles a la hora de centrar la fibra; uno de ellos es tangente a la línea que pasa por el centro de los carritos).

6. Inyectar luz por un extremo de la fibra para medir perdidas del proceso.

7. Conectar el motor del quemador para iniciar el movimiento de vaivén del quemador.

8. Empujar el quemador hasta situar la llama debajo de la fibra y dejar que esta realice un par de recorridos precalentando de este modo la fibra. Si el precalentamiento introduce perdidas, esto nos indica que la llama es demasiado fuerte. Si no se aprecian perdidas, accionaremos el interruptor i1, para poner en marcha a los motores m1, m2 y empezar a estirar.

9. Una vez alcanzado el estiramiento necesario pararemos el proceso. Es importante que se retire la llama y se desactiven los motores m1, m2 al mismo tiempo.

10. Una vez retirado el taper o la fibra estrechada, utilizaremos los conmutadores c1, c2 de la caja de electrónica para cambiar el sentido

de giro de los motores de estirado y devolver los carros a la posición inicial.

Debido a que el sistema trabaja de modo semiautomático la repetición de la fabricación de las fibras estrechadas son únicas por no decir que imposibles de repetir el proceso idéntico al anterior. Tomando eso en cuenta se decidió realizar la automatización completa correspondiente a las actividades humanas como el calculo de tiempo/desplazamiento de los carros, como también el de controlar las velocidades de los motores.

Para saber que tanto se va a estirar la fibra se utiliza la siguiente ecuación que utilizaremos para la automatización del sistema (formula obtenida durante los cursos de fibras ópticas).

$$Pf = Po * \exp^{\left(-\frac{Z}{2Lo}\right)}$$

Donde:

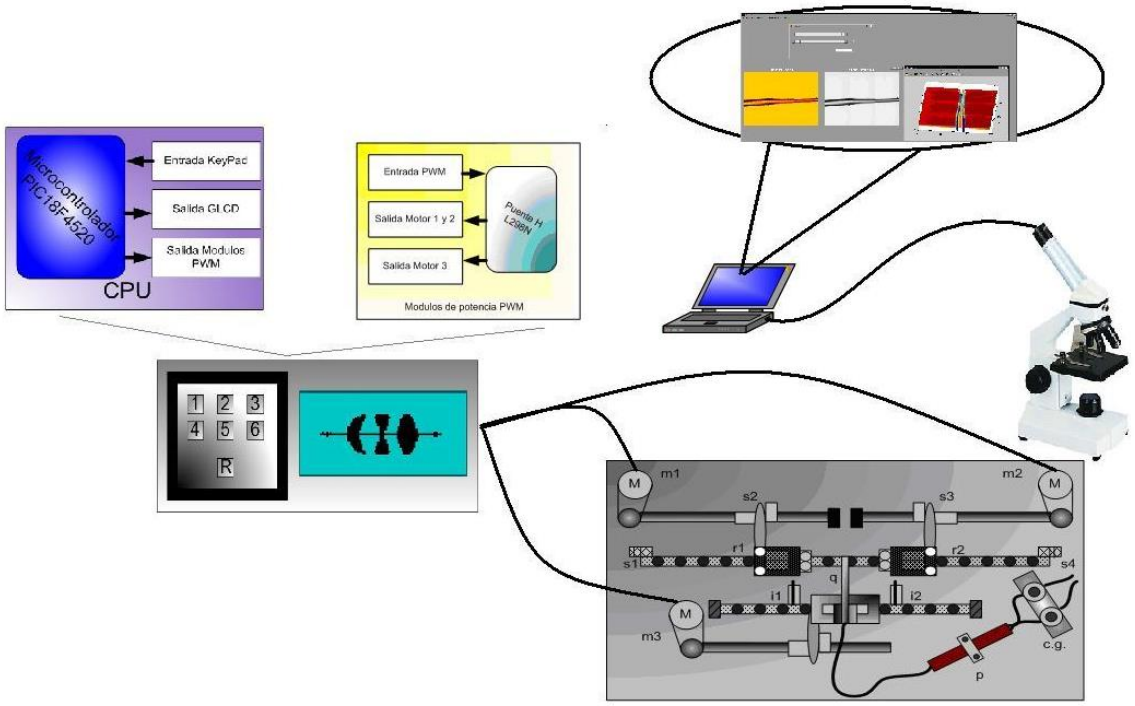
Pf= Es el diámetro de la fibra que la se quiere llegar

Po= Es el diámetro inicial que siempre será 125micrometros para iniciar

Z= Desplazamiento total

Lo= Zona de calentamiento de la flama

Diagrama a bloques de propuesta



Posibles soluciones a las problematicas

Lo que podemos aprovechar son los motores m1, m2 y mq3 para seguirlos usando ya que están montados en una superficie de buen manejo además con motores de CC fáciles de manejar debido que podemos variar su tiempo de trabajo de fácil modo (de marca Kelvin procedentes de España y se encontró sucursal en Puebla). Al igual que toda la estructura en si es buena sin ningún problema como para mandar a hacer una nueva.

Para el control del sistema completo podemos utilizar un microcontrolador el cual es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S.

Los motores podemos controlarlos con una técnica conocida como Modulación de ancho de pulso (PWM siglas en ingles pulse-width modulation), la modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (por ejemplo sinusoidal o cuadrada) ya sea para transmitir información a través de un canal de comunicaciones o control de la cantidad de energía que se envía a una carga o en este caso a los motores.

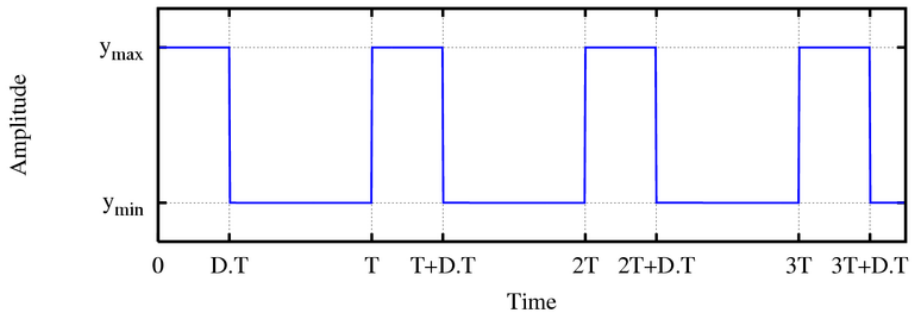
El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación al período. Matemáticamente:

$$D = \frac{\tau}{T}$$

D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función



Para la parte del mechero se utilizara un pistón o actuador como los que usan los seguros de los coches, el cual será activado según el sistema lo necesite.

Resolución de la problemática de Automatización del proceso de estrechamiento

Para iniciar el diseño y la creación del nuevo sistema empezaremos por ver las necesidades del operador, en este caso el Dr. David Monzón Hernández;

en platicas con el Dr. David Monzón y en la practica del adelgazado de las fibras pudimos notar varias cosas entre ellas que el mechero hace su barrido como debe pero es estático a la misma distancia y como nueva línea de investigación se puede hacer que esta distancia vaya aumentado relativamente con la estirada de la fibra óptica y ver que variaciones tenemos. El nuevo sistema debe ser fácil de operar, del mismo modo visual, que no ocupe tanto espacio y que sea de bajo coste.

Ahora que ya tenemos los datos recolectados y observados el funcionamiento del aparato podemos empezar a trabajar con el diseño, la programación y desarrollo del mismo.

Al inicio teníamos este diseño;

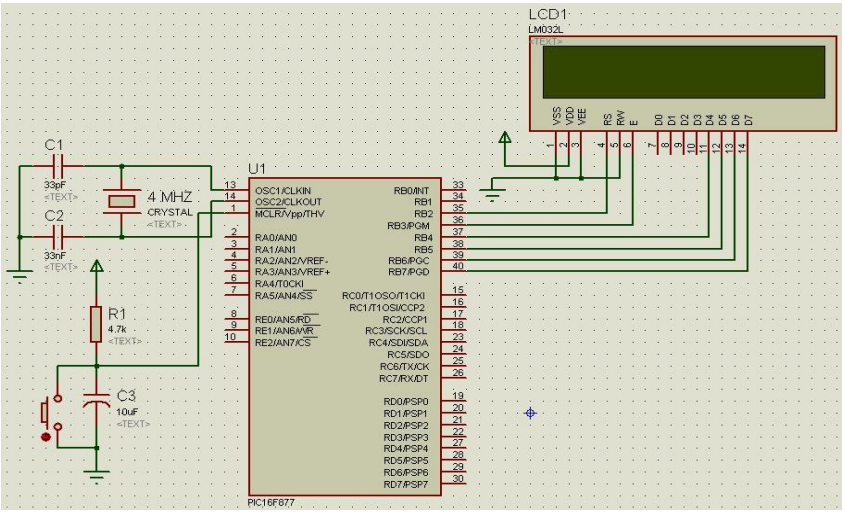


Figura 3. Primer diseño

Donde; el microcontrolador PIC16F887A tiene las siguientes características para nuestro proyecto.

- Velocidad de operación: hasta 20 MHz de reloj
- 8K x 14 bits por palabra de memoria de programa FLASH
- 368 x 8 bytes de memoria de datos (RAM)
- 256 x 8 bytes de memoria de datos EEPROM

- Dos módulos de Capture, Compare, PWM
    - Capture es de 16-bit, max. resolución es 12.5 ns
    - Compare es de 16-bit, max. resolución es 200 ns
    - PWM max. resolución de 10-bit

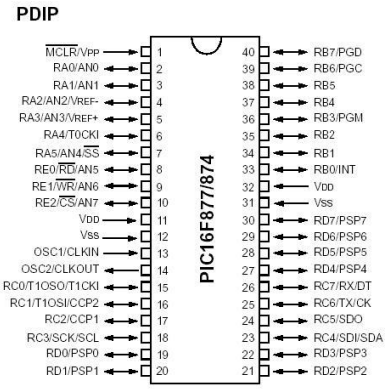


Figura 4. Imagen física del microcontrolador

Y contamos con un LCD el cual es nuestra pantalla de visualización de las operaciones que realiza nuestro sistema. Debido a la complejidad del manejo de varias variables la memoria del microcontrolador se vio afectada por lo mismo opte por utilizar uno nuevo en este caso el PIC18F4520 con mejor características. A lo que el nuevo diseño que del siguiente modo.

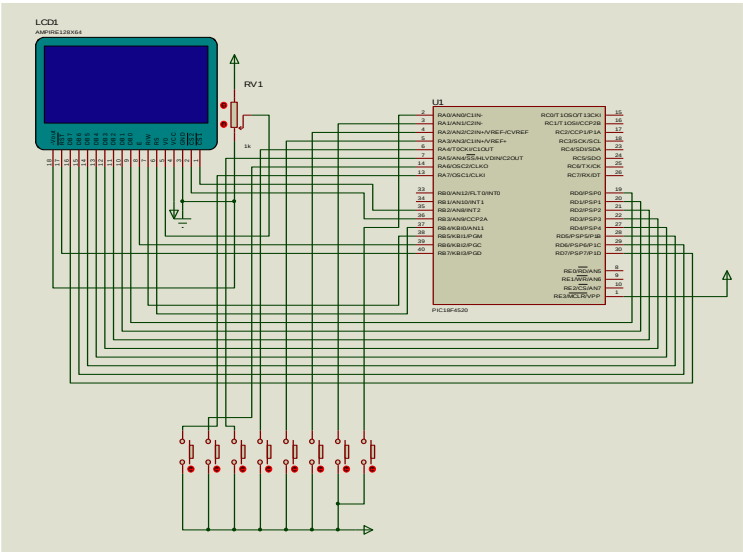


Figura 5. Nuevo diseño del sistema

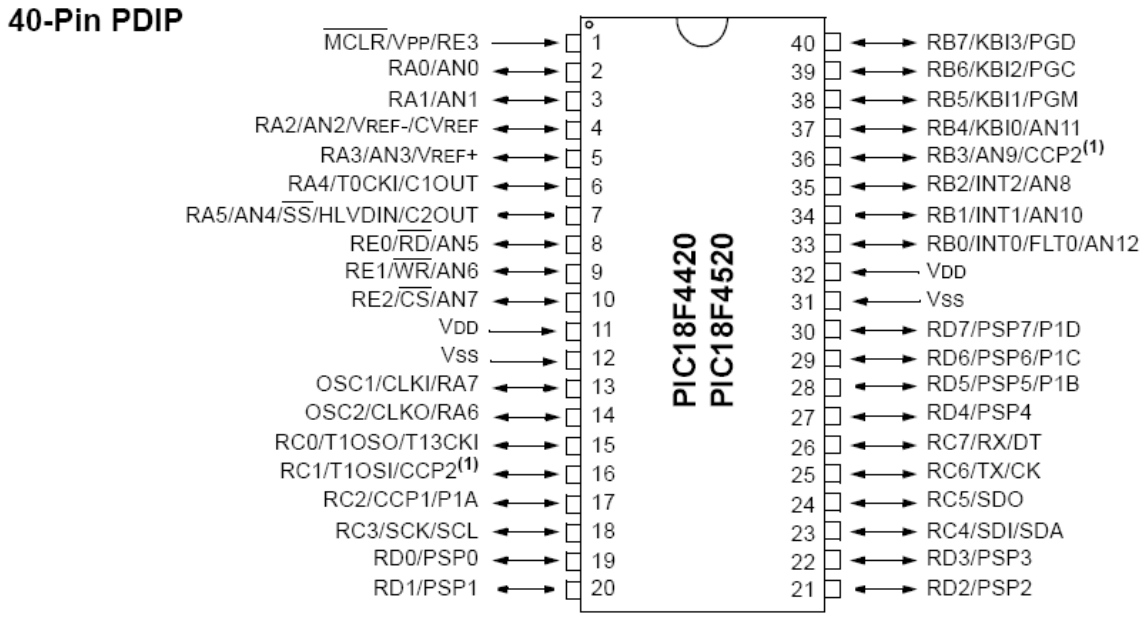


Figura 6. Imagen física del microcontrolador (2)

Del mismo modo que el PIC16F877A tiene 40 pines y cuenta con 2 PWM's

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C™			
PIC18F2420	16K	8192	768	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F2520	32K	16384	1536	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F4420	16K	8192	768	256	36	13	1/1	Y	Y	1	2	1/3
PIC18F4520	32K	16384	1536	256	36	13	1/1	Y	Y	1	2	1/3

Figura 7. Tabla de datos del microcontrolador PIC18F4520

Además cambiamos el LCD por uno Grafico para hacer mas visual el sistema y amigable para el usuario operador. Al ir avanzando en la programación del sistema mismo el Dr. David Monzón tubo la idea de realizar "Menús" a los cuales ir modificando los valores a trabajar y del mismo modo ser más vistoso.

La idea principal es la siguiente:

	Menu #1 Modo Automatico (1) Modo Manual (2)	Menu Automatico Posicionar Motores 1 y 2 a una distancia de (1) continuar (2) regresar
Menu Automatico Lo= Pf= Z= (1) continuar (2) regresar	Menu Automatico (3)'Lo' aumenta (4)'Lo' se mantiene igual (1) continuar (2) regresar	Menu Automatico Motor 1 y 2= Motor 3= (1) continuar (2) regresar
Menu Manual Motor # Direccion Velocidad (1) mover (2) detener (6)<	Menu Automatico Sistema Trabajando	Menu Automatico Fibra Optica Lista para retirar (1) menu inicial

Figura 8. Tabla de menús

A continuación presento las imágenes simuladas del GLCD.

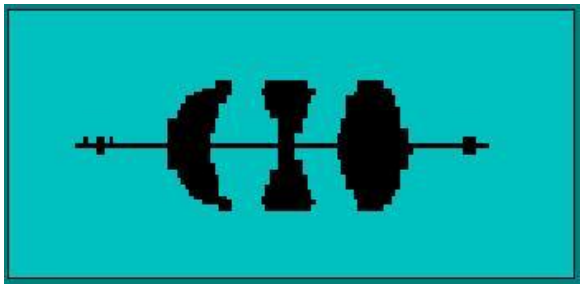


Figura 9. Icono del Cio como presentación

Figura 10. Menú de selección de modos

Aquí podemos seleccionar con la tecla 1 el modo automático y con la tecla 2 el modo manual.

Figura 11. Modo automático #1

En el modo automático nos presenta la opción de poner la distancia que deseamos colocar los 2 motores dados en centímetros.

Con la tecla 1 podemos pasar al siguiente paso y la 2 para regresar.

La tecla 3 incrementa 1 cm a la distancia y la tecla 4 incrementa 1 mm.

Figura 12. Modo automático #1 muestra de almacenamiento de la cantidad 4.90 cm

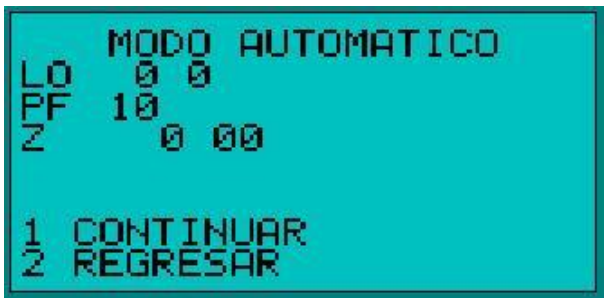


Figura 13. Modo automático #2

Del mismo modo que el anterior con la tecla 1 continuamos y 2 regresamos. Los datos adquiridos en esta parte son “Lo” que es el barrido de la

flama a realizar el motor 3, “Pf” que seria el diámetro a que se quiere llegar de la fibra; la “Z” es el resultado de los datos adquiridos anteriormente.

Figura 14. Modo automático #2 muestra

Lo = 1 cm

Pf = 40 micrometros

Z = 21.97 mm

Figura 15. Modo automático #3

En esta parte se piensa tomar la decisión si el barrido de la flama aumente con forme los motores se vayan separando o se mantenga oscilando en el mismo margen que se le pidió.

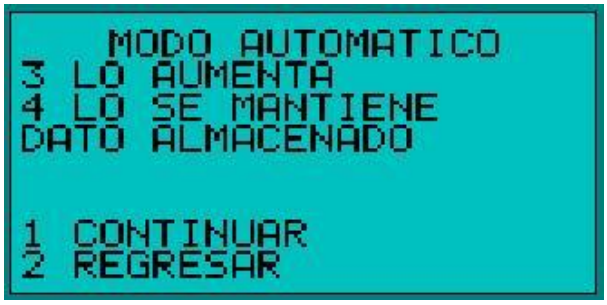


Figura 16. Modo automático #3 muestra

Después de seleccionar ya sea con la tecla 3 (Lo aumenta) o 4 (Lo se mantiene) aparece que fue almacenado el dato del usuario.

Figura 17. Modo automático #4

Aquí se seleccionara la velocidad de los motores dado en porcentaje, por el momento son incrementos de 10% por cada vez que se presionen ya sea la tecla 3 para aumentar la velocidad a los motores 1 y 2, y la tecla 4 para el motor 3.

Figura 18. Modo automático #4 muestra

Motor 1 y 2 a 10% de su velocidad, motor 3 a 50% de su velocidad.

Figura 19. Imagen final del proceso

Cuando se termino de pedir la velocidades de los motores se presenta el estado de “Sistema Trabajando” en esta parte los motores se moverán a sus posiciones y con la tecla 1 iniciara el proceso de adelgazamiento.

La siguiente serie de imágenes son los PWM’s que genera el microcontrolador dependiendo de la velocidades seleccionadas de los motores.

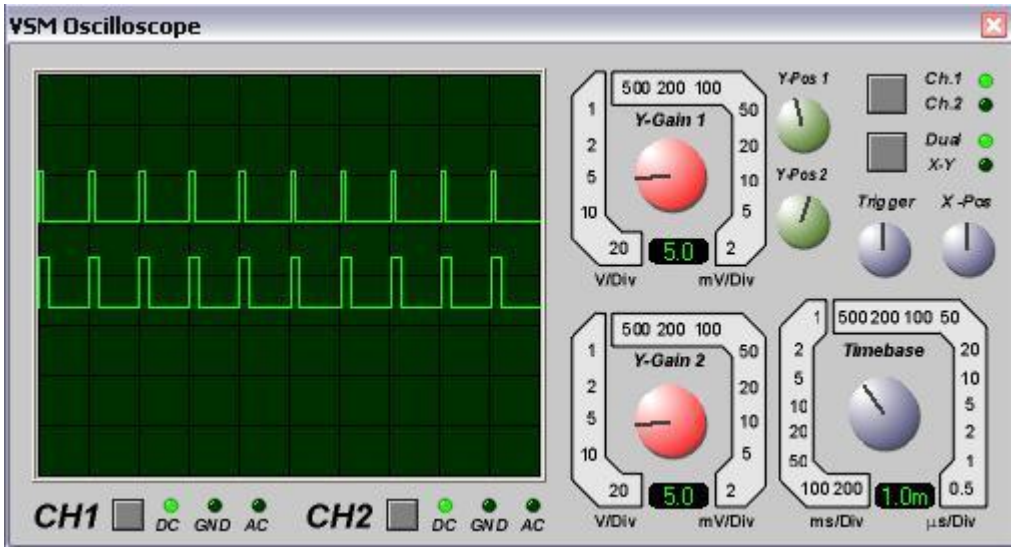


Figura 20. Aquí tenemos en la primera señal 10% de la capacidad vemos que son pulsos de 5 v, con un ancho de 1ms. La segunda señal es 20% de la capacidad.

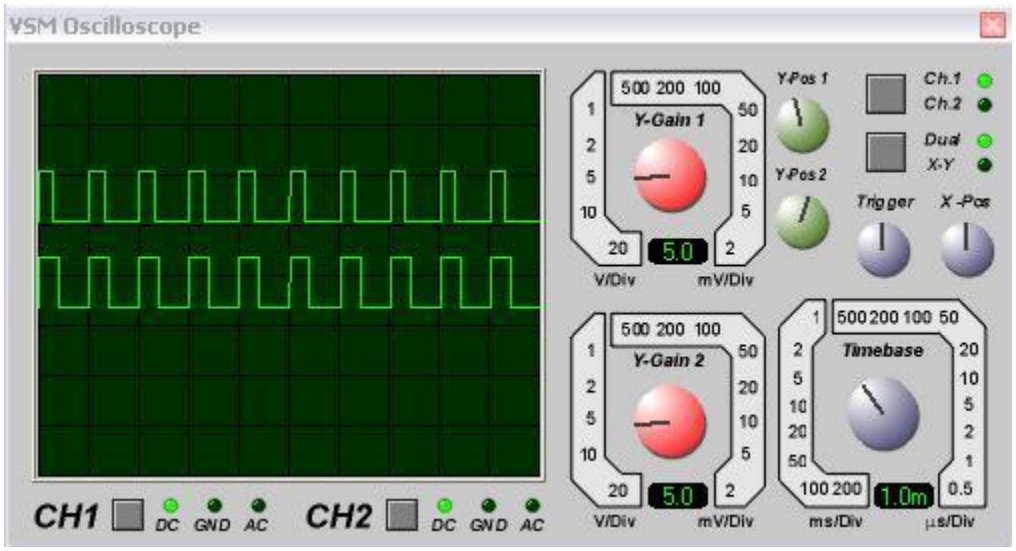


Figura 21. Aquí tenemos lo mismo que la anterior pero con 30% y 40%.

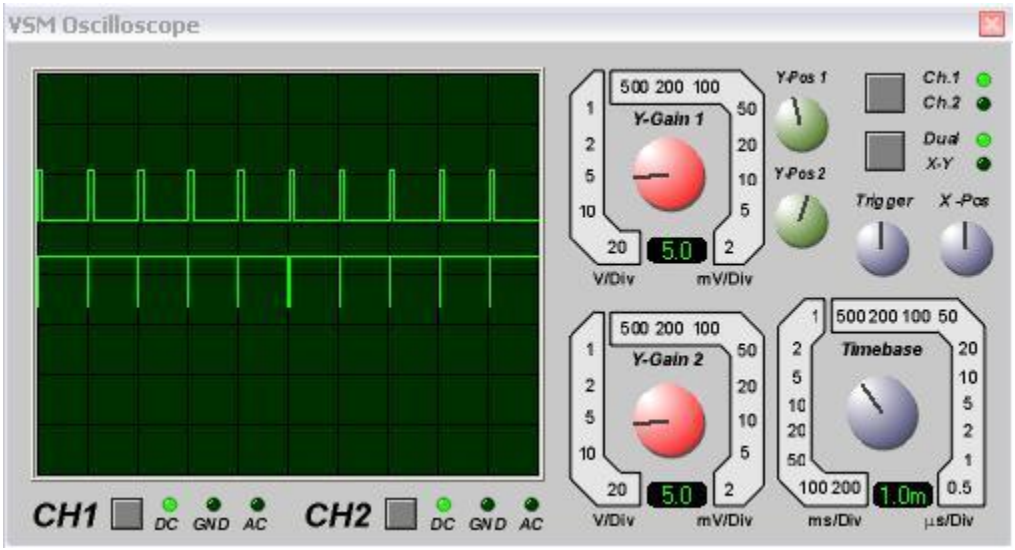


Figura 22. Esta es la muestra de la diferencia del 10% y el 100%. Podemos notar el tiempo de trabajo que tendrán los motores.

Ya que tenemos los procesos importantes bajo control podemos pasar a la parte del diseño de las placas donde va a ir nuestros circuitos. Para le manejo de los motores por el PWM utilizaremos un puente H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica y como convertidores de potencia. Los puentes H están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos.

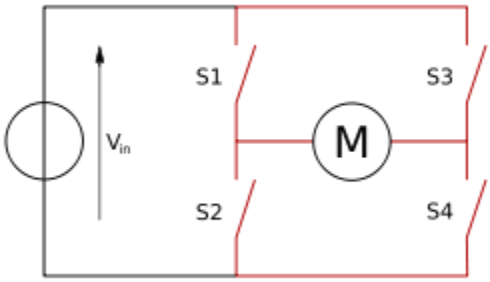


Figura 23. Estructura de un puente H (marcado en rojo)

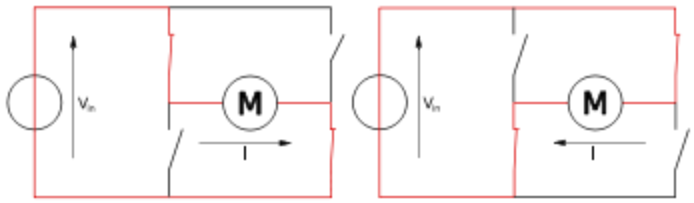


Figura 24. Los 2 estados básicos del circuito.

El término "puente H" proviene de la típica representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 (ver primera figura) están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.
Con la nomenclatura que estamos usando, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4.

S1	**S2**	**S3**	**S4**	**Resultado**
1	0	0	1	El motor gira en *avance*
0	1	1	0	El motor gira en *retroceso*
0	0	0	0	El motor se detiene bajo su inercia
0	1	0	1	El motor frena (*fast-stop*)

Elegí el L298N, que es un puente H encapsulado la razón es para dejar un diseño de módulos de potencia tratando de dejarlos del siguiente modo.

Figura 25. Imagen de muestra de posibles módulos puente H

Tendremos la entrada y salida para cada motor en específico y si en dado caso se llegara a descomponer algo por razones de calentamiento u otras cosas lo que se cambiarían son los módulos y no el sistema completo. El L298N es el integrado que podemos observar en la imagen anterior y se encuentra en el fondo, cuenta con las siguientes características.

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

Figura 26. Tabla de datos del L298N

Su diagrama de pines es el siguiente:

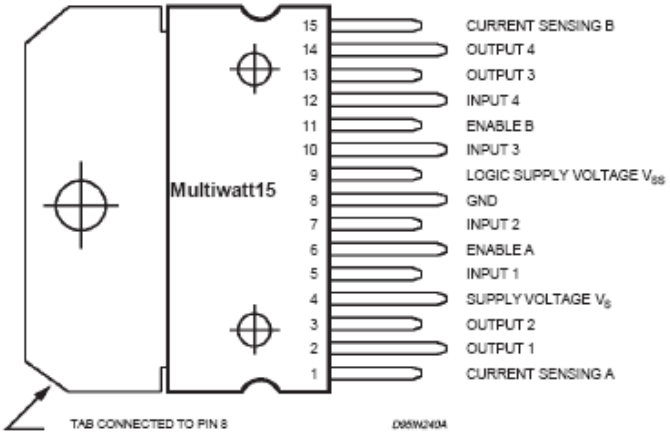
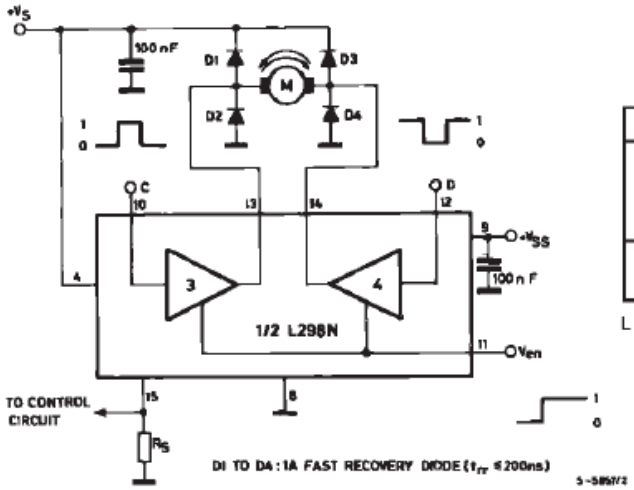


Figura 27. Imagen física del integrado L298N

Del mismo modo su diagrama de conexiones para cada motor seria el siguiente:



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

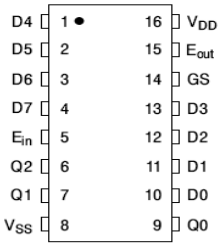
L = Low H = High X = Don't care

Figura 28. Circuito a utilizar el L298N

Donde C y D estarán conectados al microcontrolador para la selección de direcciones de los motores.

Ya tenemos los módulos de salida al cual también les sumaremos el integrado CMOS 4555 en las salidas de los PWM antes de enviarlo a los módulos de puente H. Otra mejora fue agregar el integrado CMOS 4532 a las entradas del microcontrolador para tener una entrada de tipo binaria al micro.

PIN ASSIGNMENT



TRUTH TABLE

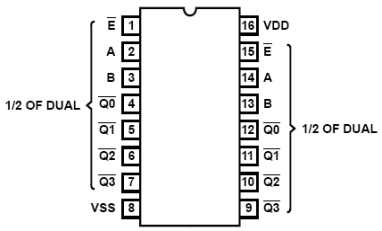
Input									Output				
E_{in}	D7	D6	D5	D4	D3	D2	D1	D0	GS	Q2	Q1	Q0	E_{out}
0	X	X	X	X	X	X	X	X	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	X	X	X	X	X	X	X	1	1	1	1	0
1	0	1	X	X	X	X	X	X	1	1	1	0	0
1	0	0	1	X	X	X	X	X	1	1	0	1	0
1	0	0	0	1	X	X	X	X	1	1	0	0	0
1	0	0	0	0	1	X	X	X	1	0	1	1	0
1	0	0	0	0	0	1	X	X	1	0	1	0	0
1	0	0	0	0	0	0	1	X	1	0	0	1	0
1	0	0	0	0	0	0	0	1	1	0	0	0	0

X = Don't Care

Figura 29. Tabla de estados e imagen física del CMOS 4532

Pinouts

CD4556BMS
TOP VIEW



TRUTH TABLE

INPUTS ENABLE SELECT			OUTPUTS CD4555BMS				OUTPUTS CD4556BMS			
E	B	A	Q3	Q2	Q1	Q0	Q3	Q2	Q1	Q0
0	0	0	0	0	0	1	1	1	1	0
0	0	1	0	0	1	0	1	1	0	1
0	1	0	0	1	0	0	1	0	1	1
0	1	1	1	0	0	0	0	1	1	1
1	X	X	0	0	0	0	1	1	1	1

X = Don't Care

Logic 1 ≡ High
Logic 0 ≡ Low

Figura 30. Tabla de estados e imagen física del CMOS 4555

Resolución de la problemática Método de medición de diámetro de la fibra

En el curso de "Procesamiento de imágenes" usando MatLab obtuve como resultado un programa capaz de digitalizar imágenes, aplicar filtros, introducir

ruidos, sacar el histograma, etc.; dentro del curso y del programa se llego a la planeación de la idea de poder medir el diámetro de la fibra mediante el procesamiento de la imagen de una fibra óptica estrechada tomada de un microscopio que se tenia en el laboratorio. A continuación se muestran algunas imágenes de muestra de dicho programa y el por que tomar esa decisión de método de medición.

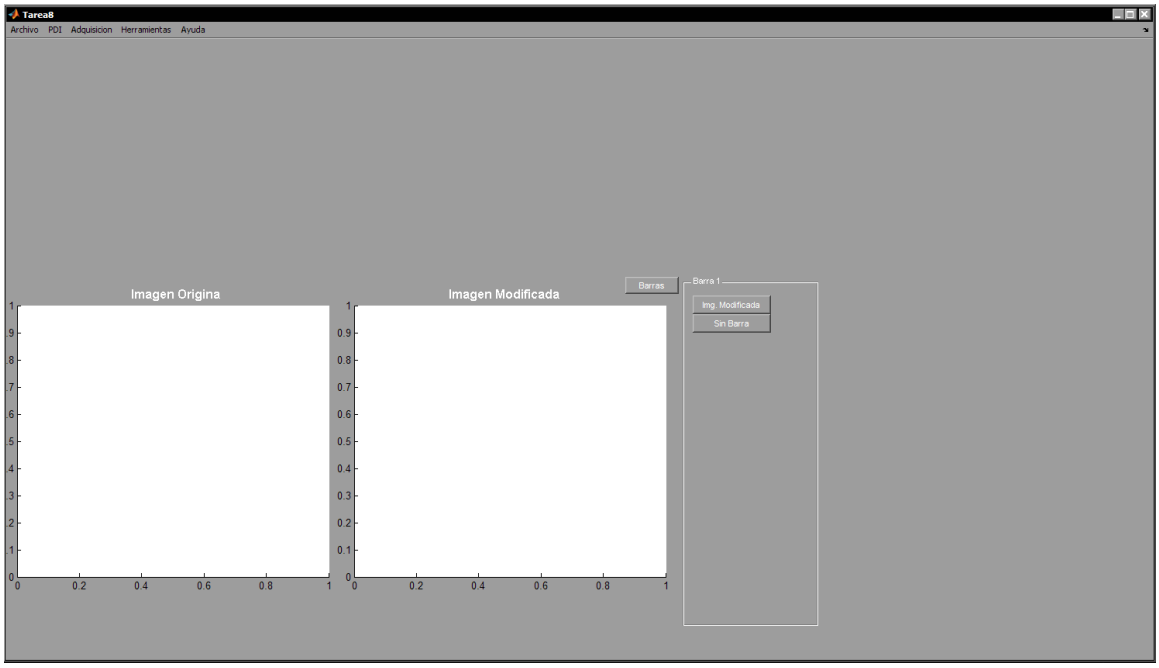
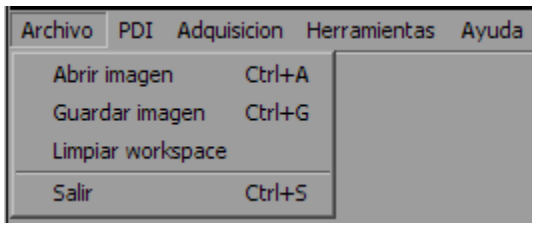
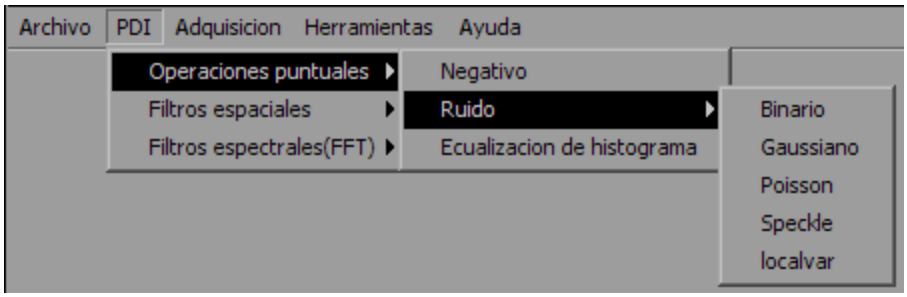
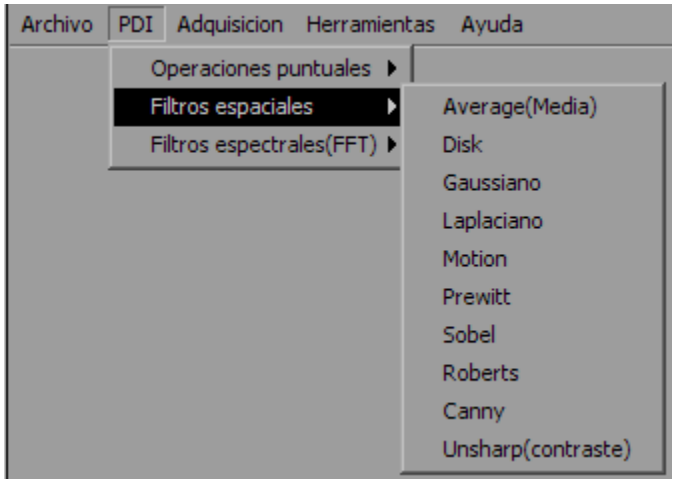


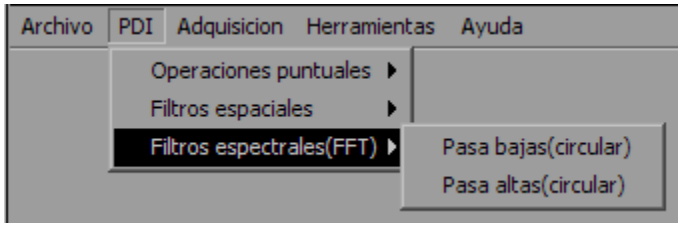
Figura 31. Imagen del programa al inicio

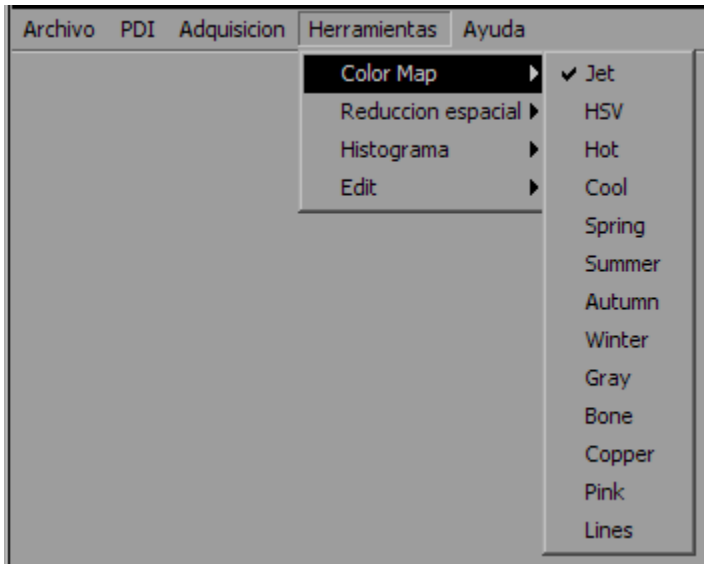
En la Figura 31. podemos ver menú inicial del programa donde se puede observar 2 recuadros donde ira la imagen original y la imagen modificada. En las pestañas de la parte de arriba tienen las siguientes opciones para modificar la imagen.

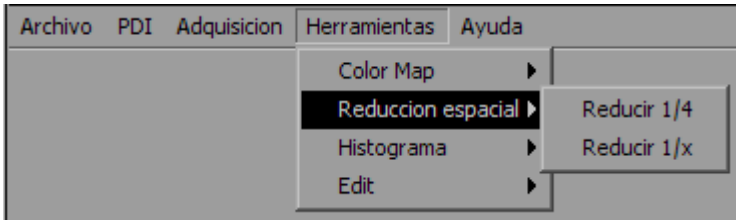


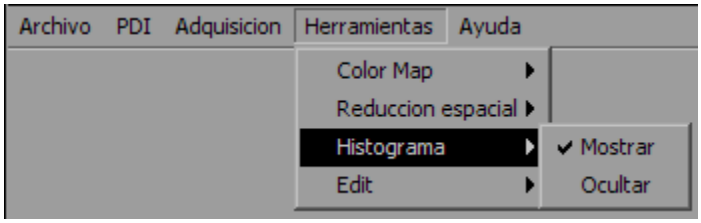

Archivo
PDI
Adquisicion
Herramientas
Ayuda
Operaciones puntuales
Filtros espaciales
Filtros espectrales(FFT)
Negativo
Ruido
Ecualizacion de histograma
Binario
Gaussiano
Poisson
Speckle
localvar

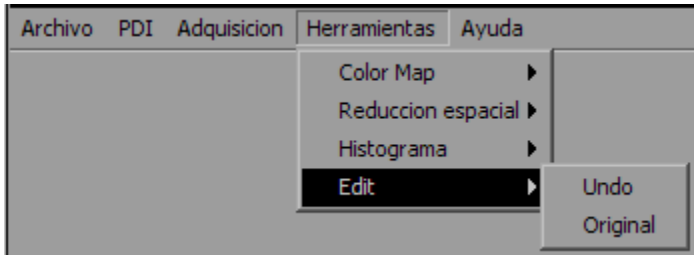

Archivo
PDI
Adquisicion
Herramientas
Ayuda
Operaciones puntuales
Filtros espaciales
Filtros espectrales(FFT)
Average(Media)
Disk
Gaussiano
Laplaciano
Motion
Prewitt
Sobel
Roberts
Canny
Unsharp(contraste)


Archivo
PDI
Adquisicion
Herramientas
Ayuda
Operaciones puntuales
Filtros espaciales
Filtros espectrales(FFT)
Pasa bajas(circular)
Pasa altas(circular)


Archivo
PDI
Adquisicion
Herramientas
Ayuda
Color Map
Reduccion espacial
Histograma
Edit
Jet
HSV
Hot
Cool
Spring
Summer
Autumn
Winter
Gray
Bone
Copper
Pink
Lines







Todas estas herramientas ayudaran para la digitalización de la imagen obtenida del microscopio, en la Opción “Adquisición” va dirigida a obtener la imagen directa desde una WebCam.

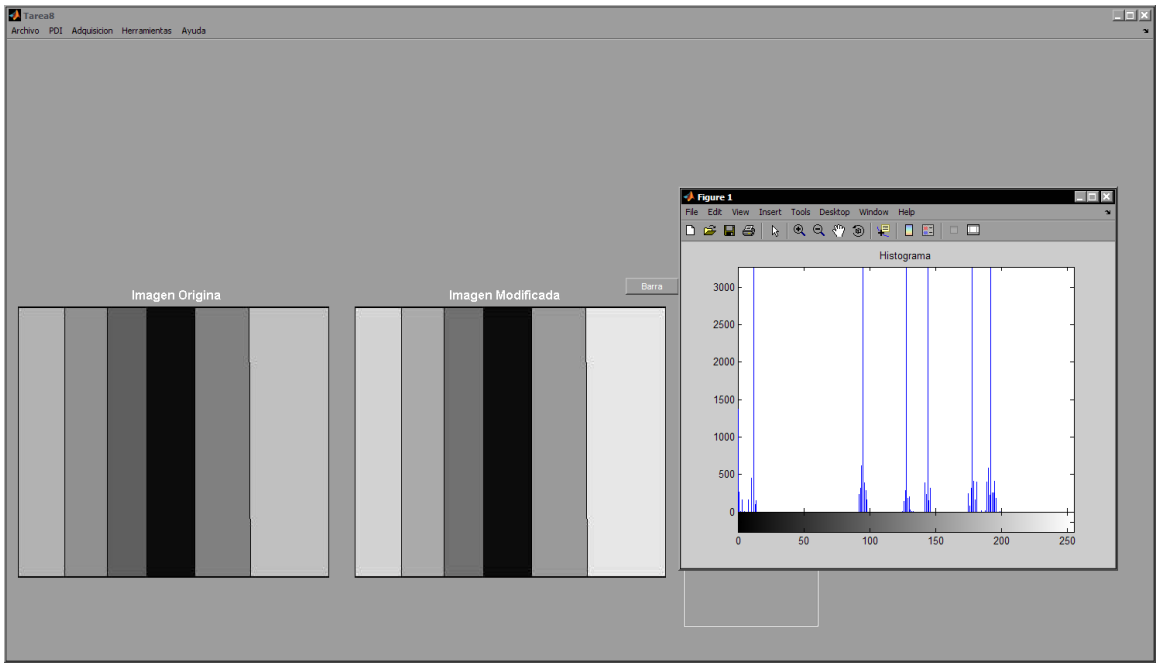


Figura 32. Prueba con escalas en grises

En la Figura 32. podemos ver que utilizando las escalas en grises y el histograma notamos 6 picos donde son los valores de cada línea de gris en la imagen, esto nos lleva a una idea, si pudiéramos medir la distancia entre una tonalidad y la otra utilizando este método seria bueno; platicando con el Dr. Manuel Ibarra de la Torre me recomendó utilizar otra forma de hacerlo, esto era utilizar una imagen y graficarla de forma 3D y a partir de la grafica observar la distancia; estas son algunas imágenes de muestra.

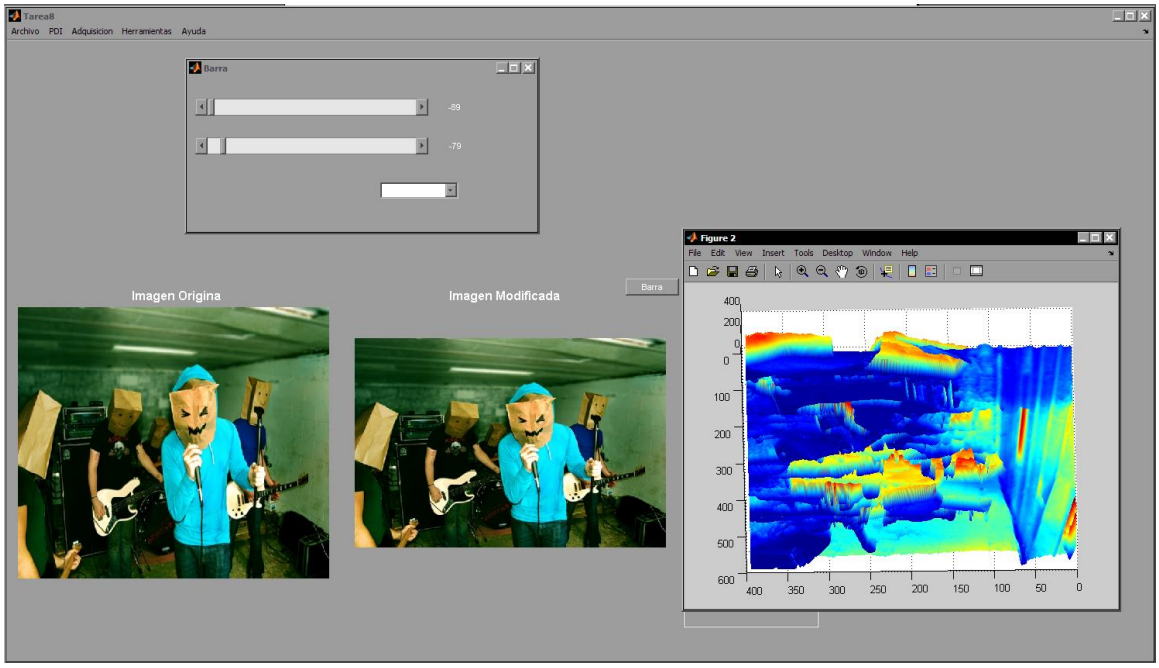


Figura 33. Graficación en 3D

En la anterior imagen podemos ver la foto que se grafico de forma 3D donde se observa el relieve de la misma, tomando esto en cuenta se puede llegar a un método de programación para medir la distancia entre un relieve a otro.

La aplicación va dirigida a que se pueda utilizar una cámara y tener imágenes en tiempo real y mediciones al mismo tiempo.

RESULTADOS, PLANOS, GRAFICAS, PROTOTIPOS Y PROGRAMAS

Resultados para “Automatización del proceso de estrechado de una fibra”

El nuevo diseño va quedando del siguiente modo.

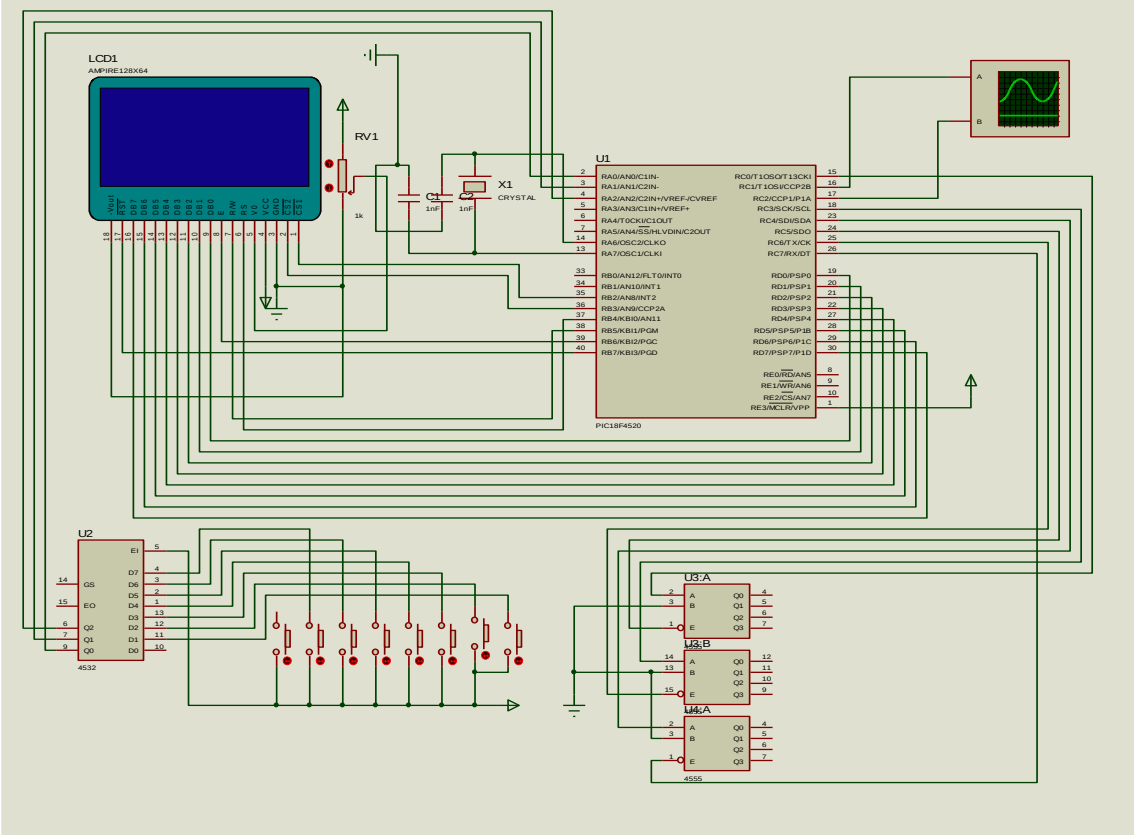


Figura 34. Diseño final

Por ultimo los diseños de los circuitos y de las placas a partir del circuito final y sus adaptaciones.

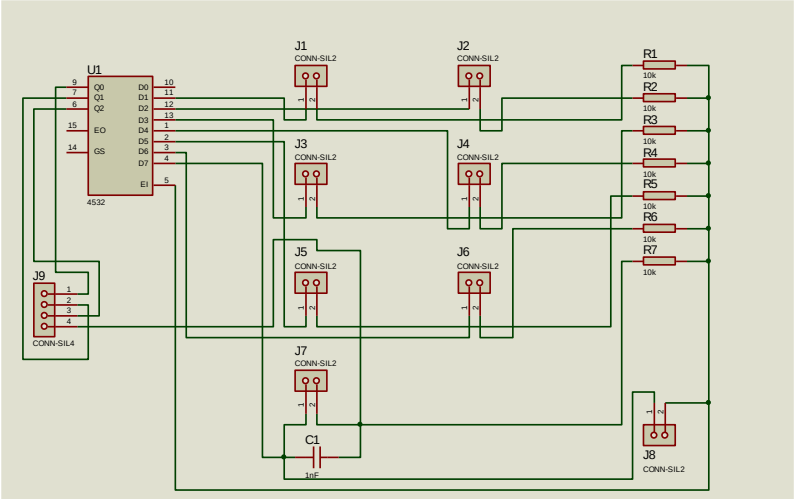


Figura 35. KeyPad a utilizar

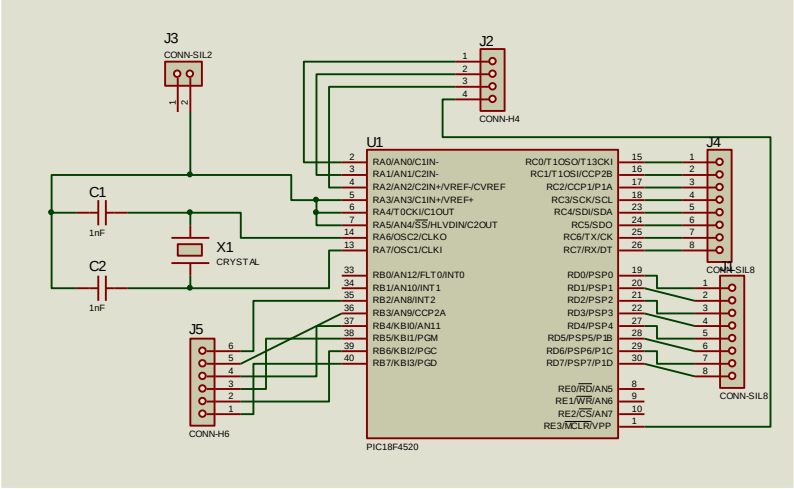


Figura 36. CPU con sus puertos de entrada y salida

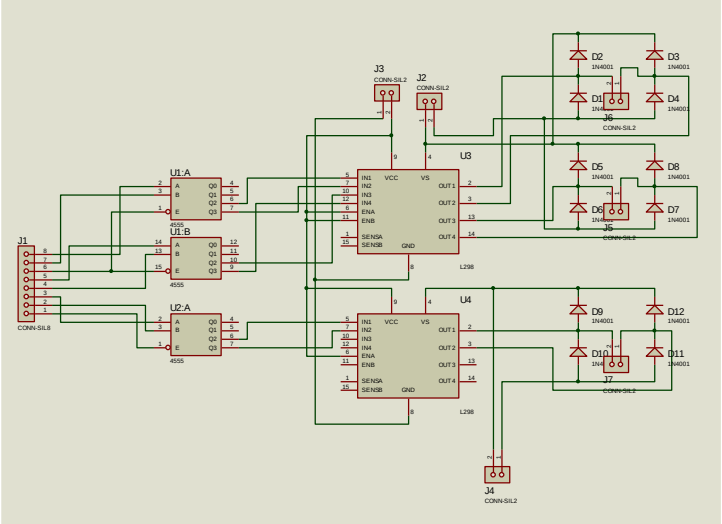
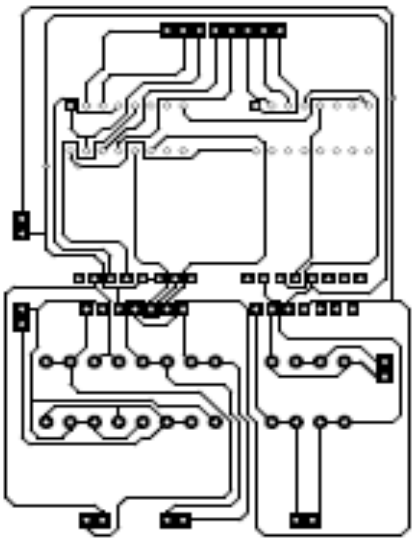
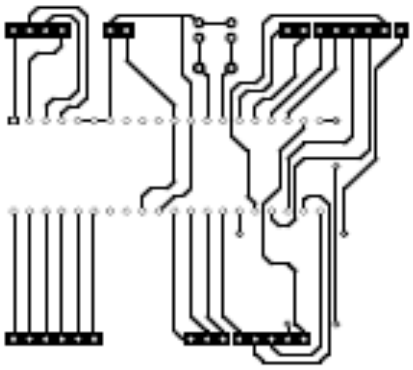
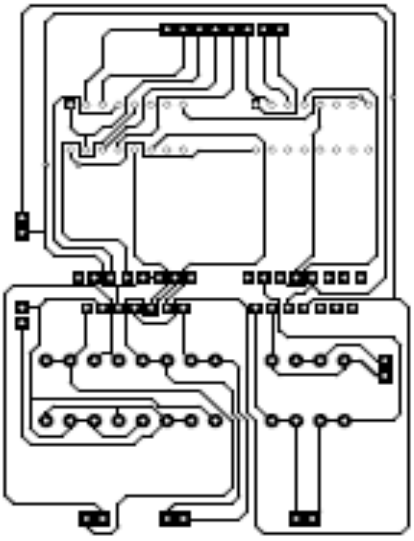


Figura 37. La sección de potencia de los motores







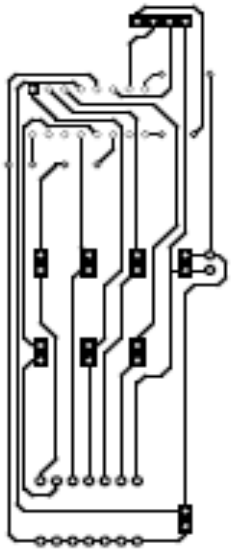
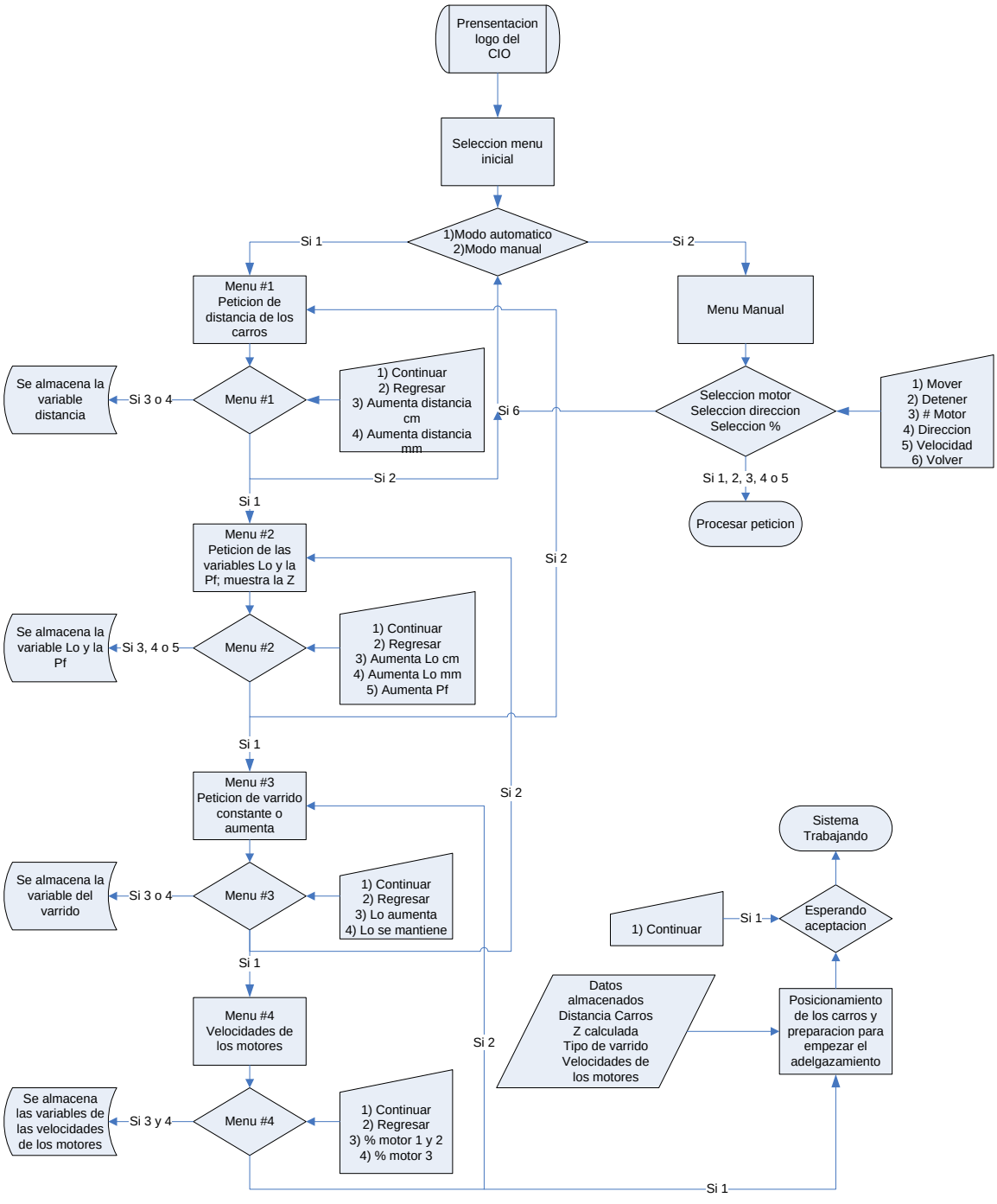


Diagrama de flujo del sistema automatizado



<u>Código del programa utilizado en el microcontrolador PIC18F4520</u>

```
unsigned char const Cio_bmp[1024] = {
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,128,128,128,128,  0,  0,  0,  0,  0,  0,  0,  0,128,128,128,128,128,128,
 128,128,128,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,128,128,128,128,128,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,128,224,248,248,252,252,254,254,254,255, 31, 31,  7,  0,  0,  0,  0,  0,  0,  0,  6,127,255,255,255,255,255,
 255,255,255,255, 30,  6,  0,  0,  0,  0,  0,  0,  0,248,248,252,255,255,255,255,255,255,254,254,248,192,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,192,  0,  0,192,192,  0,192,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,254,254,255,255,255,255,255,255,255,255, 31,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,255,255,255,
 255, 31, 31,  1,  0,  0,  0,  0,  0,  0,  0,224,254,255,255,255,255,255,255,255,255,255,255,255,255,255,240,240,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,192,192,192,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,  7,  7,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
1,127,127,255,255,255,255,255,255,255,255,225,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,129,129,255,255,255,
 255,241,241,  1,  1,  1,  1,  1,  1,  1,  1, 15,255,255,255,255,255,255,255,255,255,255,255,255,255,255,127,127,  7,
1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  7,  7,  7,  1,  1,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
1,  7, 15, 63,127,127,255,255,255,255,192,192,128,  0,  0,  0,  0,  0,  0,  0,192,248,255,255,255,255,255,
 255,255,255,255,240,128,  0,  0,  0,  0,  0,  0,  0, 15, 15,127,255,255,255,255,255,255,255,255, 63, 15,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  3,  0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  3,  3,  3,  3,
  3,  3,  3,  3,  3,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,  3,  3,  3,  3,  3,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
},blanc0_bmp[1024] = {
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
},M1[12]={0,25,51,76,102,127,153,178,204,229,255};

float dis=0,Lo=0,LoOp=0,Des=0;
int opt=0;
char *someText,cadena[10],*Direc[3]={"DERECHA","IZQUIERDA"},
*Porcen[12]={"0","10","20","30","40","50","60","70","80","90","100"};
unsigned short Po=125, Pf=10,M1duty=0, M3duty=0, Motor=1;
void sSec()
    {
    Delay_ms(1000);
    }
void mSec()
    {
    Delay_ms(500);
    }
void borrado()
    {
    Glcd_Fill(0x00);
    Glcd_Image(Blanc0_bmp);
    }
void Menu_Auto()
    {
     someText="MODO AUTOMATICO";
```

```
     Glcd_Write_Text(someText,20,0,1);
     someText="1 CONTINUAR";
     Glcd_Write_Text(someText,0,6,1);
     someText="2 REGRESAR";
     Glcd_Write_Text(someText,0,7,1);
    }
void Retros()
    {
    if(PORTA==0x01)
                    {
                    if(PORTA=0)
                            {
                            opt=opt+1;
                            borrado();
                            }
                    }
                if(PORTA==0x02)
                    {
                    if(PORTA=0)
                            {
                            opt=opt-1;
                            borrado();
                            }
                    }
    }
void FloatToString(double x, char *cadena, unsigned short nizq, unsigned short nder) {
  unsigned long int num, factor = 1;
  unsigned short k;
  for(k = 0; k < nder; k++)
    factor *= 10;
  num = factor*x;
  if(factor*x - num > 0.5) num++;
  cadena[nizq + nder + 1] = 0;
  for(k = nizq + nder; k > nizq; k--) {
    cadena[k] = num % 10 + '0';
    num /= 10;
  }
  cadena[nizq] = '.';
  for(k = nizq - 1; k > 0; k--) {
    cadena[k] = num % 10 + '0';
    num /= 10;
  }
  cadena[0] = num % 10 + '0';
  for(k = 0; cadena[k] == '0' && cadena[k+1] != '.'; k++)
    cadena[k] = ' ';
}
```

```
void Pwm_Power()
{
    Pwm1_Change_Duty(M1[M1duty]);
    Pwm2_Change_Duty(M1[M3duty]);
}

void Posicion()
{
    dis=(10-dis)*1000; ///Aqui debe ir el calculo de tiempo desplazamiento
    Vdelay_ms(dis);
}

void main(void)
    {
    ADCON1 = 0x0F;
    PORTA=0x00;
    TRISA=0x07;
    PORTC=0;
    TRISC=0;
    Pwm1_Init(5000);
    Pwm1_Start();
    Pwm2_Init(5000);
    Pwm2_Start();
    Glcd_Init(&PORTB,2,3,4,5,7,6,&PORTD);
    Glcd_Fill(0);
        for(;;)
        {       if(opt==0)
                {
                borrado();
                Glcd_Image(Cio_bmp);
                sSec();
                borrado();
                opt=1;
                }
        if(opt==1)
        {
        someText="MENU INICIAL";
        Glcd_Write_Text(someText,0,0,1);
        someText="1 MODO AUTOMATICO";
        Glcd_Write_Text(someText,0,1,1);
        someText="2 MODO MANUAL";
        Glcd_Write_Text(someText,0,2,1);

        if(PORTA==0x01)
                {
```

```
            if(PORTA=0)
                    {
                    opt=2;
                    borrado();
                    }
            }
    if(PORTA==0x02)
            {
            if(PORTA=0)
                    {
                    opt=7;
                    borrado();
                    }
            }
    }
    if(opt==2)
        {
        Menu_Auto();
        someText="POSICION DE MOTORES";
        Glcd_Write_Text(someText,0,1,1);
        someText="1 Y 2 A UNA DISTANCIA";
        Glcd_Write_Text(someText,0,2,1);
        someText="DE ";
        Glcd_Write_Text(someText,0,3,1);
        FloatToString(dis,cadena,2,2);
        Glcd_Write_Text(cadena,20,3,1);
        Retros();
        }
    if(opt==3)
        {
        Menu_Auto();
        someText="LO ";
        Glcd_Write_Text(someText,0,1,1);
        someText="PF ";
        Glcd_Write_Text(someText,0,2,1);
        someText="Z ";
        Glcd_Write_Text(someText,0,3,1);
        FloatToString(Lo,cadena,2,1);
        Glcd_Write_Text(cadena,20,1,1);
        FloatToString(Pf,cadena,2,0);
        Glcd_Write_Text(cadena,20,2,1);
        FloatToString(Des,cadena,3,2);
        Glcd_Write_Text(cadena,20,3,1);
        Retros();
        }
    if(opt==4)
```

```
        {
        Menu_Auto();
        someText="3 LO AUMENTA";
        Glcd_Write_Text(someText,0,1,1);
        someText="4 LO SE MANTIENE";
        Glcd_Write_Text(someText,0,2,1);
        if(LoOp>0)
        {
        someText="DATO ALMACENADO";
        Glcd_Write_Text(someText,0,3,1);
        }
        Retros();
        }
    if(opt==5)
        {
        Menu_Auto();
        someText="MOTOR 1 Y 2";
        Glcd_Write_Text(someText,0,1,1);
        someText="MOTOR 3";
        Glcd_Write_Text(someText,0,2,1);
        Glcd_Write_Text(Porcen[M1duty],80,1,1);
        Glcd_Write_Text(Porcen[M3duty],80,2,1);
        Retros();
        }
    if(opt==6||opt==8)
        {
        someText="SISTEMA TRABAJANDO";
        Glcd_Write_Text(someText,0,4,1);
        Pwm_Power();
        /*if(opt==6)
        {
        Pwm1_Change_Duty(255);
        Pwm2_Change_Duty(255);
        if(PORTA.F0==1&&PORTA.F1==1)
                        {
                        Posicion();
                        Pwm1_Change_Duty(0);
                        Pwm2_Change_Duty(0);
                        opt=8;
                        }
        }
        if(opt==8)
            {
            if(PORTA.F0==1)
            Pwm_Power();
            } */
```

```
        }
    if(opt==7)
        {
        someText="MENU MANUAL";
        Glcd_Write_Text(someText,40,0,1);
        someText="MOTOR NUMERO";
        Glcd_Write_Text(someText,0,1,1);
        someText="DIRECCION";
        Glcd_Write_Text(someText,0,2,1);
        someText="VELOCIDAD";
        Glcd_Write_Text(someText,0,3,1);
        someText="1 MOVER  2 DETENER";
        Glcd_Write_Text(someText,0,4,1);
        someText="3 REGRESAR";
        Glcd_Write_text(someText,0,5,1);
        }

    if(PORTA==0x03&&opt==2)
            {
            if(PORTA=0)
                    {
                    mSec();
                    dis=dis+1;
                    if(dis>5)
                    dis=0;
                    }
            }
    if(PORTA==0x04&&opt==2)
            {
            if(PORTA=0)
                    {
                    mSec();
                    dis=dis+0.1;
                    }
            }
    if(PORTA==0x03&&opt==4)
            {
            if(PORTA=0)
                    {
                    mSec();
                    LoOp=1;
                    }
            }
    if(PORTA==0x04&&opt==4)
            {
            if(PORTA=0)
```

```
                {
                mSec();
                LoOp=2;
                }
        }
if(PORTA==0x03&&opt==3)
        {
        if(PORTA=0)
                {
                mSec();
                Lo=Lo+1;
                if(Lo>5)
                Lo=0;
                }
        }
if(PORTA==0x04&&opt==3)
        {
        if(PORTA=0)
                {
                mSec();
                Lo=Lo+0.1;
                }
        }
if(PORTA==0x05&&opt==3)
        {
        if(PORTA=0)
                {
                mSec();
                Pf=Pf+10;
                if(Pf>60)
                Pf=10;
                }
        }
if(PORTA==0x06&&opt==3)
        {
        if(PORTA=0)
                {
                mSec();
                Pf=Pf+5;
                }
        }
if(PORTA==0x03&&opt==5)
        {
        if(PORTA=0)
                {
                mSec();
```

```
                        M1duty=M1duty+1;
                        if(M1duty>10)
                        M1duty=0;
                        }
                }
        if(PORTA==0x04&&opt==5)
                {
                if(PORTA=0)
                        {
                        mSec();
                        M3duty=M3duty+1;
                        if(M3duty>10)
                        M3duty=0;
                        }
                }

////FORMULA DEL SISTEMA//////////////////////////////////////////////////////////
Des=((log(Po/Pf))*2*Lo)*10;
   }
   }
```

El código se le debe trabajar un poco mas ya que para calcular la distancia recorrida por los motores usaremos la velocidad a 100% y se tomara nota del tiempo recorrido en 1 cm y a partir de ahí podemos ver la velocidad del motor a su máxima potencia $\overline{V}_x = \frac{desplazamiento}{\text{int}.de.tiempo} = \frac{\Delta x}{\Delta t}$.

Resultados para “Método de medición de diámetro”

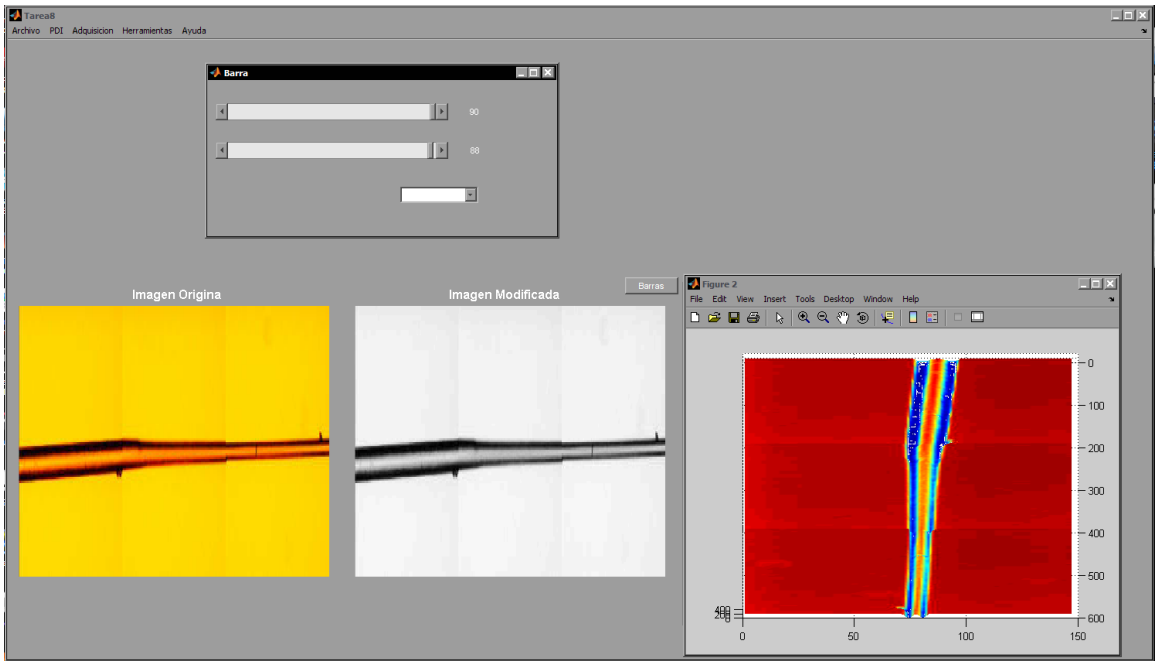


Figura 38. Muestra de una fibra adelgazada y graficación en 3D

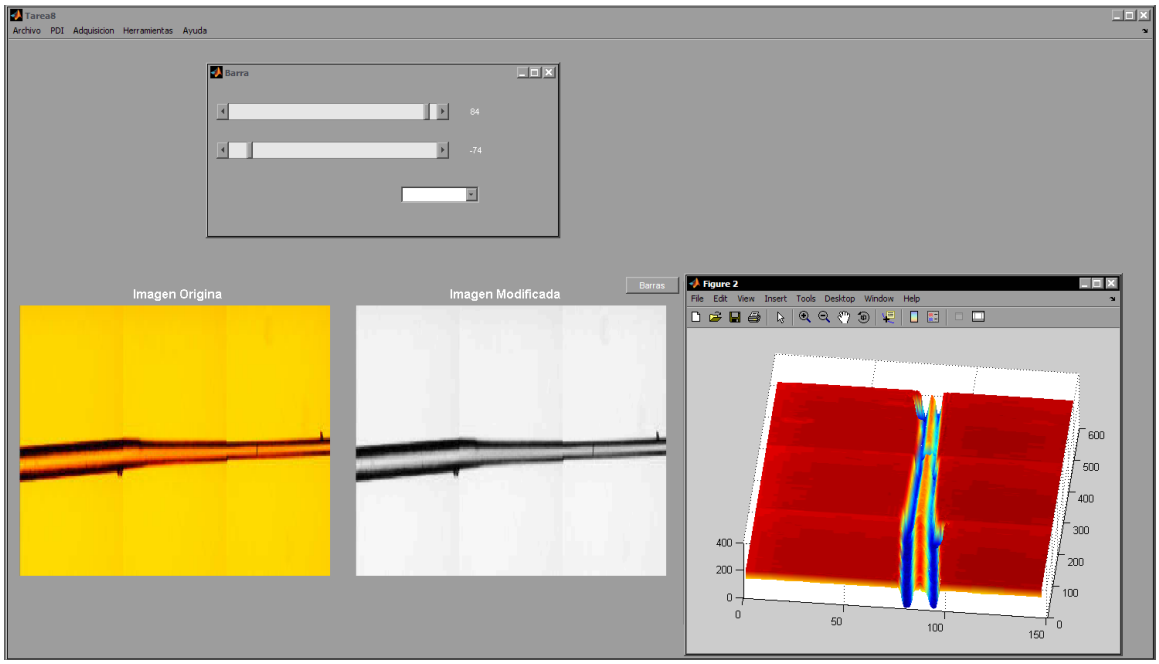


Figura 39. Manipulacion de la graficación para ver el relieve

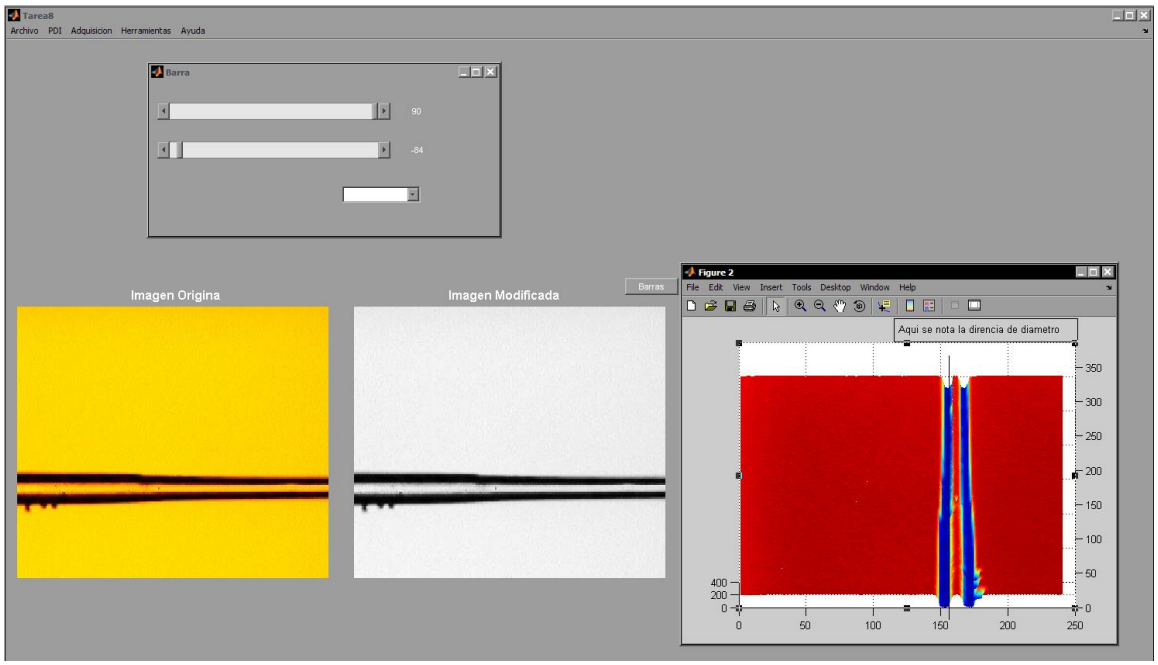


Figura 40. Imagen que muestra la diferencia de diametros

Código programación en MatLab

```
function varargout = Tarea8(varargin)
% TAREA8 M-file for Tarea8.fig
%      TAREA8, by itself, creates a new TAREA8 or raises the existing
%      singleton*.
%
%      H = TAREA8 returns the handle to a new TAREA8 or the handle to
%      the existing singleton*.
%
%      TAREA8('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TAREA8.M with the given input arguments.
%
%      TAREA8('Property','Value',...) creates a new TAREA8 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Tarea8_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Tarea8_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Tarea8

% Last Modified by GUIDE v2.5 26-Mar-2009 17:27:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Tarea8_OpeningFcn, ...
                   'gui_OutputFcn',  @Tarea8_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT


% --- Executes just before Tarea8 is made visible.
function Tarea8_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Tarea8 (see VARARGIN)

% Choose default command line output for Tarea8
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Tarea8 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
tam_forma = get(0,'ScreenSize');
fin = findobj (gcf, 'Tag','figure1');
set (fin,'position',[tam_forma(1) tam_forma(2) tam_forma(3)-200 tam_forma(4)-250]);
x=0;
 histon=0;

 assignin('base','histon',histon);
 assignin('base','flagimagen',x);


% --- Outputs from this function are returned to the command line.
function varargout = Tarea8_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --------------------------------------------------------------------
function mnuarchivo_Callback(hObject, eventdata, handles)
% hObject    handle to mnuarchivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function mnupdi_Callback(hObject, eventdata, handles)
% hObject    handle to mnupdi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function mnuabrir_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile( ...
{'*.jpg;*.gif;*.bmp;*.tif','Image Files (*.jpg,*.gif,*.bmp,*.tif)';
   '*.*',  'All Files (*.*)'}, ...
   'Open File');

 i = imread ([pathname, filename]);
 b=i;
 histon=evalin('base','histon');
```

```
 assignin('base','imoriginal',i);
 assignin('base','immodificada',b);
 axes(handles.axes1);
 image(i);
 axis off;
 axes(handles.axes2);
 imshow(b);
 axis off;
 if(histon==0)
     figure;
[x y z]=size(b);
assignin('base','z',z);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function mnuguardar_Callback(hObject, eventdata, handles)
guardar=evalin('base','immodificada');
cm=evalin('base','cm');
[filename, pathname] = uiputfile({'*.jpg;*.bmp;*.png','Image Files (*.jpg,*.bmp,*.png)';
 '*.jpg', 'JPG-files (*.jpg)';...
 '*.bmp','Imagen en Mapa de Bits (*.bmp)';...
 '*.tif','PNG-files (*.png)';...
 '*.*',  'All Files (*.*)'},...
 '*.jpg','*.jpg');
 [x y z]=size(guardar);

if z==3  % Si la imagen es RGB entonces se marca como RGB

    guardar = guardar;

else

    guardar = gray2ind(guardar);
    guardar = ind2rgb(guardar,cm);

end
imwrite(guardar,[pathname, filename]);


% --------------------------------------------------------------------
function mnusalir_Callback(hObject, eventdata, handles)
ask = questdlg('Seguro que desea salir?','Confirmacion','Si','No','Si');
switch ask
    case 'Si'
        delete(gcf)
        close all
        clear all

    case 'No'
       quit cancel;

end


% --------------------------------------------------------------------
function mnuayuda_Callback(hObject, eventdata, handles)


% --------------------------------------------------------------------
```

```
function mnunegativo_Callback(hObject, eventdata, handles)
x=evalin('base','flagimagen');
histon=evalin('base','histon');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b = 255 - (b);
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
imshow(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function mnuruido_Callback(hObject, eventdata, handles)



% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function mnujet_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
```

```
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (jet)
cm = colormap;
if(histon==0)
close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function Untitled_3_Callback(hObject, eventdata, handles)
msgbox ('Elaborado Por: Emmanuel Castillejos Villatoro' , 'About', 'warn')


% --------------------------------------------------------------------
function mnuhsv_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (hsv)
cm=colormap;
if(histon==0)
close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
```

```
function mnuhot_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (hot)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnucool_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
```

```
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (cool)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnuspring_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (spring)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnusummer_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
```

```
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (summer)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnuautumn_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (autumn)
cm=colormap;
```

```
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnuwinter_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (winter)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnugray_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
```

```
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (gray)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnubone_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (bone)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
```

```
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnucopper_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
       b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (copper)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnupink_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnulines'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
```

```
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (pink)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnulines_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');

fin = findobj (gcf, 'Tag', 'mnuhsv'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuhot'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucool'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuspring'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnusummer'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuautumn'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnuwinter'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnugray'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnubone'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnucopper'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnupink'); set (fin, 'Checked' , 'off');
fin = findobj (gcf, 'Tag', 'mnujet'); set (fin, 'Checked' , 'off');
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
[x y z]=size(b);
if z==3
        b=rgb2gray(b);
end
axes (handles.axes2)
imagesc (b)
axis off;
colormap (lines)
cm=colormap;
if(histon==0)
    close Figure 1;
figure;
imhist(b(:,:,1));
title('Histograma')
end
assignin ('base','immodificada',b);
assignin('base','undo',undo);
assignin('base','cm',cm);


% --------------------------------------------------------------------
function mnuregresar_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
```

```
b=evalin('base','immodificada');
original=evalin('base','imoriginal');
b=original;
assignin('base','immodificada',b);
axes(handles.axes2);
image(b);
axis off;
if(histon==0)
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function mnuundo_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
undo=evalin('base','undo');
assignin('base','immodificada',undo);
axes(handles.axes2);
image(undo);
axis off;
if(histon==0)
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_4_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_5_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_6_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
```

```
function Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_12_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_15_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_22_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_33_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_36_Callback(hObject, eventdata, handles)
evalin('base','clear all');
% --------------------------------------------------------------------
function Untitled_34_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;

        b = double(b);
        b = b(:,:,1);
        b = b - mean(b(:));
        transfor= fftshift(fft2(b));
        [M N] = size (b);
        MC = zeros(M,N);
        I = 1:M;J = 1:N;
        x = J -(N/2);y = (M/2)-I;
        [X,Y] = meshgrid(x,y);
        R = 25;
        A = (X.^2 + Y.^2 <= R^2);
        MC(A) = 1; Cmasc = MC;
        trans = transfor.*Cmasc;
        b = abs(ifft2(trans)/2000);
        axes(handles.axes2)
        imagesc(b);
        axis off;
        assignin('base','immodificada',b);
        assignin('base','undo',undo);
        if(histon==0)
    close Figure 1;
figure;
```

```
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma')
end


% --------------------------------------------------------------------
function Untitled_35_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;

        b = double(b);
        b = b(:,:,1);
        b = b - mean(b(:));
        transfor= fftshift(fft2(b));
        [M N] = size (b);
        MC = ones(M,N);
        I = 1:M;J = 1:N;
        x = J -(N/2);y = (M/2)-I;
        [X,Y] = meshgrid(x,y);
        R = 25;
        A = (X.^2 + Y.^2 <= R^2);
        MC(A) = 0; Cmasc = MC;
        trans = transfor.*Cmasc;
        b = abs(ifft2(trans)/2000);
        axes(handles.axes2)
        imagesc(b);
        axis off;
        assignin('base','immodificada',b);
        assignin('base','undo',undo);
        if(histon==0)
    close Figure 1;
figure;
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma')
end


% --------------------------------------------------------------------
function Untitled_23_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('average',[3 3]);
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
```

```
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_24_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('disk',3);
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_25_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('gaussian',[10 10],0.5);
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
```

```
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_26_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('laplacian',1);
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_27_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('motion',10,90);
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
```

```
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_28_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('prewitt');
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_29_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('sobel');
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
```

```
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_30_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
    b = edge(b(:,:,1),'roberts');
    axes(handles.axes2)
    imshow(b);
    axis off;
    assignin('base','immodificada',b);
    if(histon==0)
close Figure 1;
    figure;
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma')
end



% --------------------------------------------------------------------
function Untitled_31_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
    b = edge(b(:,:,1),'canny');
    axes(handles.axes2)
    imshow(b);
    axis off;
    assignin('base','immodificada',b);
    if(histon==0)
close Figure 1;
    figure;
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma')
end


% --------------------------------------------------------------------
function Untitled_32_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
H = fspecial('unsharp',0.5);
```

```
b = imfilter(b,H,'replicate');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes(handles.axes2)
image(b)
axis off
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_21_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b=histeq(b(:,:,1));
axes(handles.axes2)
imshow(b)
figure(2);
image(b)
figure(3);
imagesc(b)
axis off;
if(histon==0)
    close Figure 1;
    figure;
    imhist(b(:,:,1));
    title('Histograma')
end
assignin('base','immodificada',b);
assignin('base','undo',undo);









% --------------------------------------------------------------------
function Untitled_13_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');
fin = findobj (gcf, 'Tag', 'Untitled_14'); set (fin, 'Checked' , 'off');
histon=0;
assignin('base','histon',histon);
b=evalin('base','immodificada');
axes(handles.axes2)
imagesc(b)
axis off
```

```
close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end



% --------------------------------------------------------------------
function Untitled_14_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');
fin = findobj (gcf, 'Tag', 'Untitled_13'); set (fin, 'Checked' , 'off');
close Figure 1
histon=1;
assignin('base','histon',histon);


% --------------------------------------------------------------------
function Untitled_11_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
    b = imresize(b, .25);
    axes(handles.axes2)
    imagesc(b); colormap gray;
    axis off;
    assignin('base','immodificada',b);
    if(histon==0)
        close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
    end
 assignin('base','undo',undo);


% --------------------------------------------------------------------
function Untitled_8_Callback(hObject, eventdata, handles)
prompt = 'Valor X a dividir la imagen';
res = inputdlg(prompt);
res=str2double(res);
res=1/res;
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
```

```
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
    b = imresize(b, res);
    axes(handles.axes2)
    image(b); colormap gray;
    axis off;
    assignin('base','immodificada',b);
    assignin('base','undo',undo);
    if(histon==0)
        close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_9_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');
fin = findobj (gcf, 'Tag', 'Untitled_10'); set (fin, 'Checked' , 'off');
x=evalin('base','flagimagen');
x=1;
assignin('base','flagimagen',x);


% --------------------------------------------------------------------
function Untitled_10_Callback(hObject, eventdata, handles)
set (gcbo, 'Checked', 'on');
fin = findobj (gcf, 'Tag', 'Untitled_9'); set (fin, 'Checked' , 'off');
x=evalin('base','flagimagen');
x=0;
assignin('base','flagimagen',x);


% --------------------------------------------------------------------
function Untitled_16_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b = imnoise(b,'salt & pepper', 0.2);
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
image(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
```

```
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_17_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b = imnoise(b,'gaussian', 0,0.1);
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
image(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_18_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b = imnoise(b,'poisson');
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
image(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
```

```
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_19_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
b = imnoise(b,'speckle', 0.1);
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
image(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --------------------------------------------------------------------
function Untitled_20_Callback(hObject, eventdata, handles)
histon=evalin('base','histon');
x=evalin('base','flagimagen');
if(x==0)
    b=evalin('base','immodificada');
end
if(x==1)
    b=evalin('base','imoriginal');
end
undo=b;
V=rand(size(b));
b = imnoise(b,'localvar',V);
assignin('base','immodificada',b);
assignin('base','undo',undo);
axes (handles.axes2)
image(b)
axis off;
if(histon==0)
    close Figure 1;
     figure;
[x y z]=size(b);
if z<3
   subplot(3,1,1);imhist(b(:,:,1));
   title('Histograma')
else
```

```
subplot(3,1,1);imhist(b(:,:,1));
title('Histograma R')
subplot(3,1,2);imhist(b(:,:,2));
title('Histograma G')
subplot(3,1,3);imhist(b(:,:,3));
title('Histograma B')
end
 end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
f=get(hObject,'Value');
fin=findobj(gcf,'Tag','pushbutton1');
if f==1
    set(fin,'String','Img. Activa');
    x=evalin('base','flagimagen');
    x=1;
    assignin('base','flagimagen',x);
else
    set(fin,'String','Img. Modificada');
    x=evalin('base','flagimagen');
    x=0;
    assignin('base','flagimagen',x);
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
f=get(hObject,'Value');
fin=findobj(gcf,'Tag','pushbutton2');
if f==1
    set(fin,'String','Con Barra');
    Barra;
else
    set(fin,'String','Sin Barra');
    close 'Barra';
end


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2


% --- Executes on button press in togglebutton3.
function togglebutton3_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton3


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
f=get(hObject,'Value');
fin=findobj(gcf,'Tag','pushbutton4');
fon=findobj(gcf,'Tag','uipanel1');
fon2=findobj(gcf,'Tag','uipanel11');
```

```
if f==1
    set(fon2,'Visible','on');
    set(fon,'Visible','off');
    set(fin,'String','Barra 2');
else
    set(fon2,'Visible','off');
    set(fon,'Visible','on');
    set(fin,'String','Barra');
end


% --------------------------------------------------------------------
function AdCam_Callback(hObject, eventdata, handles)
    fin=findobj(gcf,'Tag','pushbutton4');
    fon=findobj(gcf,'Tag','uipanel1');
    fon2=findobj(gcf,'Tag','uipanel11');
    fon3=findobj(gcf,'Tag','listbox3');
    set(fon2,'Visible','on');
    set(fon,'Visible','off');
    set(fin,'String','Barra 2');
    set(fon3,'Visible','on');
    a=imaqhwinfo;
    set(fon3,'String',a.InstalledAdaptors);




% --- Executes on selection change in listbox3.
function listbox3_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox3
    f=get(hObject,'Value');
    a=imaqhwinfo;
    fin=findobj(gcf,'Tag','listbox4');
        assignin('base','cowin',a.InstalledAdaptors(1,f));
        a=imaqhwinfo(strvcat(a.InstalledAdaptors(1,f)));
        set(fin,'String',a.DeviceInfo.DeviceName);
        assignin('base','infowin',a);


% --- Executes during object creation, after setting all properties.
function listbox3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox4.
function listbox4_Callback(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox4 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox4
info=evalin('base','infowin');
f=get(hObject,'Value');
fin=findobj(gcf,'Tag','listbox5');
if f==1
```

```
    set(fin,'String',info.DeviceInfo.SupportedFormats);
end

% --- Executes during object creation, after setting all properties.
function listbox4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox5.
function listbox5_Callback(hObject, eventdata, handles)
% hObject    handle to listbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox5 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox5
f=get(hObject,'Value');
assignin('base','formats',f);

% --- Executes during object creation, after setting all properties.
function listbox5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PreviewB.
function PreviewB_Callback(hObject, eventdata, handles)
f=get(hObject,'Value');
cowin=evalin('base','cowin');
formats=evalin('base','formats');
h=evalin('base','infowin');
fin=findobj(gcf,'Tag','uipanel12');
obj=videoinput(strvcat(cowin),1,strvcat(h.DeviceInfo.SupportedFormats(1,formats)));
set(obj,'SelectedSourceName','input1');
src_obj=getselectedsource(obj);
a=get(src_obj);
if f==1
    set(fin,'Visible','on');
    Satura=a.Saturation;
    Contras=a.Contrast;
    Shar=a.Sharpness;
    Gama=a.Gamma;
    Brigh=a.Brightness;
    assignin('base','Satura',Satura);
    assignin('base','Contras',Contras);
    assignin('base','Shar',Shar);
    assignin('base','Gama',Gama);
    assignin('base','Brigh',Brigh);

fin=findobj(gcf,'Tag','text3');
set(fin,'string',num2str(Satura));
fin=findobj(gcf,'Tag','Saturacion');
set(fin,'value',Satura);
fin=findobj(gcf,'Tag','text4');
```

```
set(fin,'string',num2str(Contras));
fin=findobj(gcf,'Tag','Contraste');
set(fin,'value',Contras);
fin=findobj(gcf,'Tag','text5');
set(fin,'string',num2str(Shar));
fin=findobj(gcf,'Tag','Sharpness');
set(fin,'value',Shar);
fin=findobj(gcf,'Tag','text6');
set(fin,'string',num2str(Gama));
fin=findobj(gcf,'Tag','Gamma');
set(fin,'value',Gama);
fin=findobj(gcf,'Tag','text7');
set(fin,'string',num2str(Brigh));
fin=findobj(gcf,'Tag','Brightness');
set(fin,'value',Brigh);
end
start(obj); preview(obj);
assignin('base','Datas',src_obj);



% --- Executes on slider movement.
function Saturacion_Callback(hObject, eventdata, handles)
% hObject    handle to Saturacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

fin=findobj(gcf,'Tag','Saturacion');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text3');
set(fin,'string',num2str(val));
a=evalin('base','Datas');
set(a,'Saturation',val);


% --- Executes during object creation, after setting all properties.
function Saturacion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Saturacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

    fin=findobj(gcf,'Tag','Saturacion');
    fil=100;
    set(fin,'min',0); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text3');
    set(fin,'string',0);

% --- Executes on slider movement.
function Contraste_Callback(hObject, eventdata, handles)
% hObject    handle to Contraste (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
fin=findobj(gcf,'Tag','Contraste');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text4');
set(fin,'string',num2str(val));
a=evalin('base','Datas');
set(a,'Contrast',val);
```

```
% --- Executes during object creation, after setting all properties.
function Contraste_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Contraste (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
    fin=findobj(gcf,'Tag','Contraste');
    fil=100;
    set(fin,'min',0); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text4');
    set(fin,'string',0);

% --- Executes on slider movement.
function Sharpness_Callback(hObject, eventdata, handles)
% hObject    handle to Sharpness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
fin=findobj(gcf,'Tag','Sharpness');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text5');
set(fin,'string',num2str(val));
a=evalin('base','Datas');
set(a,'Sharpness',val);

% --- Executes during object creation, after setting all properties.
function Sharpness_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Sharpness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
    fin=findobj(gcf,'Tag','Sharpness');
    fil=100;
    set(fin,'min',0); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text5');
    set(fin,'string',0);

% --- Executes on slider movement.
function Gamma_Callback(hObject, eventdata, handles)
% hObject    handle to Gamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
fin=findobj(gcf,'Tag','Gamma');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text6');
set(fin,'string',num2str(val));
a=evalin('base','Datas');
set(a,'Gamma',val);

% --- Executes during object creation, after setting all properties.
function Gamma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Gamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
    fin=findobj(gcf,'Tag','Gamma');
    fil=100;
    set(fin,'min',0); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text6');
    set(fin,'string',0);

% --- Executes on slider movement.
function Brightness_Callback(hObject, eventdata, handles)
% hObject    handle to Brightness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
fin=findobj(gcf,'Tag','Brightness');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text7');
set(fin,'string',num2str(val));
a=evalin('base','Datas');
set(a,'Brightness',val);

% --- Executes during object creation, after setting all properties.
function Brightness_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Brightness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
fin=findobj(gcf,'Tag','Brightness');
    fil=100;
    set(fin,'min',0); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text7');
    set(fin,'string',0);

% --- Executes on slider movement.
function slider8_Callback(hObject, eventdata, handles)
% hObject    handle to slider8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Satura=evalin('base','Satura');
Brigh=evalin('base','Brigh');
Gama=evalin('base','Gama');
Shar=evalin('base','Shar');
a=evalin('base','Datas');
Contras=evalin('base','Contras');
fin=findobj(gcf,'Tag','text3');
set(fin,'string',num2str(Satura));
fin=findobj(gcf,'Tag','Saturacion');
set(fin,'value',Satura);
fin=findobj(gcf,'Tag','text4');
set(fin,'string',num2str(Contras));
fin=findobj(gcf,'Tag','Contraste');
set(fin,'value',Contras);
fin=findobj(gcf,'Tag','text5');
set(fin,'string',num2str(Shar));
fin=findobj(gcf,'Tag','Sharpness');
set(fin,'value',Shar);
fin=findobj(gcf,'Tag','text6');
set(fin,'string',num2str(Gama));
fin=findobj(gcf,'Tag','Gamma');
set(fin,'value',Gama);
fin=findobj(gcf,'Tag','text7');
set(fin,'string',num2str(Brigh));
fin=findobj(gcf,'Tag','Brightness');
set(fin,'value',Brigh);
set(a,'Brightness',Brigh);
set(a,'Gamma',Gama);
set(a,'Sharpness',Shar);
set(a,'Contrast',Contras);
set(a,'Saturation',Satura);
```

Código de Graficación en 3D

```
function varargout = Barra(varargin)
% BARRA M-file for Barra.fig
%      BARRA, by itself, creates a new BARRA or raises the existing
%      singleton*.
%
%      H = BARRA returns the handle to a new BARRA or the handle to
%      the existing singleton*.
%
%      BARRA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BARRA.M with the given input arguments.
%
%      BARRA('Property','Value',...) creates a new BARRA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Barra_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Barra_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Barra

% Last Modified by GUIDE v2.5 04-Mar-2009 12:01:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Barra_OpeningFcn, ...
```

```
                   'gui_OutputFcn',  @Barra_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Barra is made visible.
function Barra_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Barra (see VARARGIN)

% Choose default command line output for Barra
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Barra wait for user response (see UIRESUME)
% uiwait(handles.figure1);
el=0; az=0; contents=1;
assignin('base','el',el);
assignin('base','az',az);
assignin('base','pop',contents);
a=evalin('base','immodificada');
[x y z]=size(a);
if z==3
        a=rgb2gray(a);
end
a=double(a);
assignin('base','matriz',a);

% --- Outputs from this function are returned to the command line.
function varargout = Barra_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
a=evalin('base','matriz');
contents=evalin('base','pop');
el=evalin('base','el');
fin=findobj(gcf,'Tag','slider1');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text1');
set(fin,'string',num2str(val));
assignin('base','az',val);
if(contents==1)
    figure(2)
    surf(a); shading interp;
    view(val,el)
end
if(contents==2)
    figure(2)
    mesh(a); shading interp;
    view(val,el)
end
if(contents==3)
    figure(2)
    waterfall(a); shading interp;
```

```
    view(val,el)
end
if(contents==4)
    figure(2)
    ribbon(a);shading interp;
    view(val,el)
end
if(contents==5)
    figure(2)
    surface(a);shading interp;
    view(val,el)
end


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
    fin=findobj(gcf,'Tag','slider1');
    fil=90;
    set(fin,'min',-90); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text1');
    set(fin,'string',0);



% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
a=evalin('base','matriz');
contents=evalin('base','pop');
az=evalin('base','az');
fin=findobj(gcf,'Tag','slider3');
val=round(get(fin,'value'));
fin=findobj(gcf,'Tag','text2');
set(fin,'string',num2str(val));
assignin('base','el',val);
if(contents==1)
    figure(2)
    surf(a);shading interp;
    view(az,val)
end
if(contents==2)
    figure(2)
    mesh(a);shading interp;
    view(az,val)
end
if(contents==3)
    figure(2)
    waterfall(a);shading interp;
    view(az,val)
end
if(contents==4)
    figure(2)
    ribbon(a);shading interp;
    view(az,val)
end
if(contents==5)
    figure(2)
    surface(a);shading interp;
    view(az,val)
end


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
```

```
end
    fin=findobj(gcf,'Tag','slider3');
    fil=90;
    set(fin,'min',-90); set(fin,'max',fil);
    set(fin,'value',0);
    set(fin,'SliderStep',[(1/fil) (1/fil)]);
    fin=findobj(gcf,'Tag','text2');
    set(fin,'string',0);


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
contents=get(hObject,'Value');
assignin('base','pop',contents);
el=evalin('base','el');
az=evalin('base','az');
a=evalin('base','matriz');
if(contents==1)
    figure(2)
    surf(a);shading interp;
    view(az,el)
end
if(contents==2)
    figure(2)
    mesh(a);shading interp;
    view(az,el)
end
if(contents==3)
    figure(2)
    waterfall(a);shading interp;
    view(az,el)
end
if(contents==4)
    figure(2)
    ribbon(a);shading interp;
    view(az,el)
end
if(contents==5)
    figure(2)
    surface(a);shading interp;
    view(az,el)
end


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Conclusión

El sistema automatizado realizado en esta investigación se puede seguir avanzando en mejoras tanto de software y hadware de tal modo que llegaria a poder patentarce como una nueva opcion o propuesta para desarrollo de investigación en el area de las fibras opticas.

De igual modo el software se le puede trabajar un poco más, ademas de poder montar la camara en un microscopio con posibilidades de mover sus perillas según lo requiera el programa o el usuario para hacer barridos completos de la fibra y almacenarce en una base de datos.

Bibliografía

1.- Handbook of OPTICAL FIBRE sensing technology
Jose Miguel Lopez-Higuera
Chapter 1.- Introduction to fibre optics sensing technology
Chapter 4.- Optical Waveguides and their Manufacture
Chapter 13.- Gas Spectroscopy Techniques for Optical Fibre Sensors
Part Four: Applications

2.- Introduction to FIBER OPTICS
Ajoy Ghatak
K. Thyagarajan
Chapter 2.- Basic optics
Chapter 3.- Basic characteristics of the optical fiber
Chapter 7.- Modes in planar waveguides

3.- Fiber Optic Essentials
Casimer DeCusatis
Carolyn J. Sher DeCusatis
Chapter 8.- Fabrication and Measurement

4.- Fundamentals of Photonics
B.E.A Saleh
M.C. Teich
Chapter 2.- Wave optics
Chapter 5.- Electromagnetic optics
Chapter 7.- Guide-Wave optics

5.- Understanding fiber optics Jeff Hetch. Prentice Hall.

6.- Optical Fiber Sensor Technology Ed. K. T. V. Grattan and B. T. Meggitt, Chapman and Hall (1995)

7.- Análisis y diseño de experimentos, Humberto Gútierrez Pulido y Roman de la Vara Salazar, McGraw-Hill (2007)

8.- R.K. Kenny, T.A. Birks and K.P. Oakley, Electron. Lett, Vol. 77, pp. 1654 – 1656, 1991.

9.- T.A. Birks, Y. W. Li, J. Lightwave Technol., Vol. 50, pp. 432 – 438, 1992

CURSOS

1.- Introducción a Fibras Ópticas

Impartido por el Dr. Oliver Pottiez y el Dr. Rubén Grajales Coutiño

2.- MatLab

Impartido por el Dr. Manuel Ibarra de la Torre

PROGRAMAS UTILIZADOS

1.- MikroC (copilador lenguaje C para microcontroladores)

2.- Proteus 6 y 7 (simulador de circuitos)

3.- MatLab