Universidad Autónoma de Baja California

Facultad de Ciencias

PROGRAMACION PARA EL CONTROL DE UN SISTEMA DE SOPORTE ACTIVO PARA UN ESPEJO PRIMARIO DEFORMABLE

MEMORIAS DEL SERVICIO SOCIAL

Que para obtener el título de

Físico

presenta:

MAURICIO CARRILLO TRIPP

Ensenada, Baja California, México. Octubre de 1997.

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

"PROGRAMACION PARA EL CONTROL DE UN SISTEMA DE SOPORTE ACTIVO PARA UN ESPEJO PRIMARIO DEFORMABLE"

MEMORIAS DEL SERVICIO SOCIAL

Que para obtener el título de:

FISICO

Presenta:

Mauricio Carrillo Tripp.

Aprobado por:

Dr. LUIS SALAS CASALES
Presidente

Fís. LEONEL GUTIERREZ ALBORES
Secretario

Fís. FRANCISCO JUAREZ GARCIA
1er. Vocal

Quiero agradecer a Luis Salas Casales y a Leonel Gutiérrez Albores por haberme dado la oportunidad de trabajar en el proyecto.

Quiero dedicar este trabajo a mis padres y mejores amigos José Carrillo Cedillo e Irma C. Tripp Franyuti, que gracias a su apoyo y confianza he logrado llegar a donde estoy.

Resumen.

Resumen de las Memorias del Servicio Social de Mauricio Carrillo Tripp con título "Programación para el control de un sistema de soporte activo para un espejo primario deformable" presentada como requisito parcial para la obtención de la Licenciatura en Física. Ensenada, Baja California, México. Octubre de 1997.

Este trabajo tiene como objetivo el desarrollo de un programa de computadora que controle el sistema de soporte activo del espejo primario de 2.1 metros localizado en el telescopio del Observatorio Astronómico Nacional de la UNAM en la sierra de San Pedro Mártir. Este programa tiene que ser compatible con la mecánica y electrónica que se desarrollarán a la par.

Se presentan los diferentes factores que influyen en la calidad de la imagen, sus anchos de banda y posibles correcciones. Del mismo modo se discuten las diferencias entre lo que actualmente se conoce como óptica activa y óptica adaptiva, así como sus definiciones modernas.

El problema que se ataca es el de la conversión de un telescopio clásico a uno con un sistema de soporte activo con el objeto de mejorar la calidad de la imagen. Se analiza la estructura general del proyecto y la forma en que se planeó el control de las diferentes etapas o componentes del sistema. Así mismo, se discute el programa en sí, objetivo particular de este trabajo, y se explica cómo funciona, cómo controla el sistema y qué es lo que hace su ciclo principal. El código fuente se presenta en el apéndice C.

Finalmente, se presentan resultados con base en tres pruebas hechas usando diferentes métodos, indicando que se logró corregir la imagen con el sistema instalado.

Es importante recalcar que se discuten los diferentes elementos que componen al sistema, no sólo la parte computacional, con el propósito de darle la dimensión real al problema.

Abstract.

The intention of this work is to develop a computer software that controls OAN's 2.1 m primary mirror active system, which is located on the San Pedro Martir observatory. This software has to be in agreement with the mechanics and electronics that are developed at the same time.

We present the different factors that affect the image quality, their band width and their possible corrections. We also talk about the differences between what its know as active optics and adaptive optics, giving their modern definitions.

We attack the problem of converting a classic telescope to one with an active mirror support with the purpose of enhancing the image quality. We consider the project's general structure and the way the control of the different parts or components of the system were planed. Likewise, we also discuss the control program itself and explain how it works. The source code is on appendix C.

Finally, we present the results obtained by three tests using different methods. This results show that the image correction was achieved. It is important to remark that we discuss about all the components of the system, not just the computational part, to picture the problem with its real dimension.

Indice.

Contenido.

Figuras

Tablas.

Ecuaciones.

I. Introducción.

Por más de 300 años, desde su invención, el telescopio reflector ha sido pasivo en el sentido de que su sistema óptico está diseñado, fabricado y montado de acuerdo a un diseño predeterminado y subsecuentemente modificado solamente por alguna intervención de mantenimiento (Fig. 1). Su calidad óptica en cualquier momento está sujeta a errores tanto de fabricación (invariables) como de ajuste y soporte (variables). Los errores del último tipo siempre han sido serios pero se multiplican con la óptica moderna más ligera y flexible. En el año de 1989 el diseño de telescopios sufrió una revolución cuando el concepto de óptica activa se aplicó en el Telescopio de Nueva Tecnología (NTT) del Observatorio de Europa del Sur (ESO), y se corrigieron varios términos de aberración incluyendo uno muy grande debido a esfericidad. Se logró obtener imágenes que contenían el 80 % de la luz en un círculo de 0.3 arcsec de diámetro (d_{80}). A partir de entonces, todos los nuevos telescopios se construyen incorporando una serie de actuadores para deformar el espejo para cancelar las aberraciones que tenga. De este modo, un telescopio activo tiene un sistema de control que permite checar y optimizar automáticamente en cualquier momento la calidad óptica.

un orden de magnitud mayor. Aunque hay lugares con mejor seeing, la cantidad aceptada para la mayoría de los telescopios en el mundo es de d_b>0.5 arcseg. Debido a las "alas" de la función, lo que d_b significa, en la práctica, es el diámetro que contiene cerca de 75-80 % de la energía en la imagen de una fuente puntual.

Para aperturas mayores que un cierto parámetro r_0 [1] (Tamaño de las celdas de coherencia de la atmósfera, distancia en que, debido a variaciones en el índice de refracción, se obtiene un cambio de fase de 1 radián en una onda electromagnética), d_b es independiente de la apertura D. Incluso en algunos lugares como el observatorio de la ESO en La Silla, que es aceptado como muy buen lugar, este parámetro ráramente excede los 30 cm [2], así que, para telescopios grandes, d_b se puede tomar como independiente de D.

I.A.3. Calidad del espejo.

La especificación óptica del fabricante ha puesto el límite a la calidad de la imagen que se pueda obtener. Todos los telescopios grandes convencionales fabricados desde 1945 han tenido especificaciones d_c no muy de acuerdo con el mejor seeing esperado, de tal modo que la capacidad instrumental final de tales telescopios ha sido también d_c=0.5 arcseg, también independiente de D. De nuevo, debido a las "alas" de la función, d_c significa en la práctica, el diámetro que contiene el 75-80 % de la energía de la imagen de una fuente puntual.

Desafortunadamente, el hecho de que el fabricante óptico ha cumplido la especificación anterior d_c para los elementos ópticos básicos no significa que el funcionamiento de los telescopios tenga esta calidad. Un gran número de factores técnicos impiden a la mayoría de los telescopios tener sus especificaciones por más de una pequeña fracción del tiempo de observación. Este es el problema básico a que se refiere la óptica 'activa' (el segundo aspecto se refiere al mejoramiento de la calidad óptica más allá de la que el fabricante entrega).

Para aclarar esto, la tabla I lista 10 fuentes de error que pueden afectar la imagen óptica del telescopio:

Fuente de Error	Ancho de banda [Hz]
Diseño óptico	d.c
Fabricación óptica	d.c.
Errores teóricos de:	
Soportes del espejo	d.c. - 10^{-3}
Estructura (foco, centrado)	10^{-3}
Errores de mantenimiento de la estructura y soportes del espejo	10^{-5} (semanas)
Distorsiones térmicas:	
Espejos	10^{-4}
Estructura	10^{-3}
Distorsión mecánica de espejos	10^{-6} (años)
Efectos térmicos del ambiente	10^{-3} - 10^{2}
Deformaciones del espejo debido a	

corrientes de aire	10^1-10^2
Turbulencia atmosférica	2x 10^2 -10^3
Errores de guiado	5 - 10^2

Tabla I. Fuentes de error con sus respectivos anchos de banda. d.c. se refiere a corriente directa la cual tiene una frecuencia cero.

Los errores están considerados desde el punto de vista de su ancho de banda, el rango de frecuencia temporal sobre el cual pueden variar.

Es claro que el diseño óptico y la fabricación son efectos completamente de d.c.. De hecho, el diseño óptico es sólo de importancia para telescopios con un gran campo: la corrección en los ejes es trivial. De los errores teóricos de soporte y estructura, el último se debe principalmente a la flexión lateral o compresión longitudinal del tubo. El límite superior del ancho de banda en este caso es esencialmente el de movimientos del telescopio de guiado y búsqueda. Los errores de mantenimiento son usualmente más serios que los teóricos, ya que incluyen todo mal funcionamiento de soportes, desalineación de componentes debido a cambios y otros desajustes tales como mal posicionamiento axial o focal de elementos ópticos. El ancho de banda es muy pequeño ya que tales degradaciones ocurren en semanas o meses después de haber hecho el ajuste. Las distorsiones térmicas son esencialmente de bajo ancho de banda ya que la masa y, consecuentemente, la capacidad térmica involucrada en los espejos y estructuras de grandes telescopios es considerable, incluso si se toman medidas para reducir la masa en telescopios modernos comparados con los convencionales. La distorsión mecánica de espejos

ha tenido poca importancia en el pasado debido a la alta estabilidad conformacional de los vidrios, pero al considerar otros materiales [3,4] (metales o compuestos) llama la atención esta fuerte fuente de error. De nuevo, debido a que el proceso de liberación de esfuerzos o cambios estructurales son muy lentos, el ancho de banda es muy pequeño.

El primer error asociado con el aire se debe a efectos térmicos en el medio ambiente. Estos efectos son llamados comúnmente 'seeing del telescopio' y no siempre es fácil distinguirlo del seeing externo. El ancho de banda puede variar desde frecuencias bajas hasta frecuencias altas cuando se asocia con el seeing externo. Para muchos de los telescopios convencionales, ésta es una de las peores fuentes de error.

La deformación del espejo debido a corrientes de aire es un problema de gran importancia para telescopios modernos, con espejos muy grandes y flexibles, operando en un ambiente abierto para reducir errores debidos a efectos de aire anteriormente mencionados. En telescopios convencionales con domos el problema casi no existía, siendo el efecto del ambiente más peligroso. Debido a que los espejos grandes tienen una masa e inercia considerables, actúan como filtros de bajo ancho de banda para el espectro de corrientes de aire.

La turbulencia atmosférica (seeing) es el efecto que, si se dieran las condiciones de perfecto mantenimiento y condiciones del lugar, sería el factor limitante. De todas las fuentes de error, su ancho de banda es fácilmente el más grande. Finalmente, los

errores de guiado se deben principalmente a vibraciones, ya que las frecuencias bajas son absorbidas por el autoguiado.

I.B. Óptica activa y óptica adaptiva.

Desde sus orígenes en 1976 [5], se ha usado el término "óptica activa" para los sistemas de corrección de lazo cerrado para telescopios (que tiene autocorrección), usando un analizador de imágenes. En 1979 Pearson, Freeman y Reynolds [6] propusieron la definición general de que el término 'adaptivo' debería implicar un control en lazo cerrado mientras que 'activo' al control en lazo abierto (control manual). En 1982 Woolf [7] propuso el uso del término 'adaptivo' para corrección del seeing atmosférico y el término 'activo' para la corrección de errores del telescopio.

Con esta definición, "óptica activa" se refiere a todas las fuentes de error de baja frecuencia, "óptica adaptiva" a fuentes de frecuencia alta.

II. Metodología.

Ha quedado demostrado que la eficiencia de un telescopio es proporcional al cuadrado de la calidad de la imagen. Esto implica, por ejemplo, que un telescopio de

2 metros de diámetro con una imagen de 1.5 arcsec[†] es diez veces menos eficiente que uno del mismo diámetro pero con una imagen de 0.5 arcsec por lado. Este tipo de factores se han comprobado al comparar el funcionamiento del NTT con el antiguo telescopio de 3.6 metros de la ESO.

El objetivo general de este proyecto fué el de llevar el tamaño de imagen del espejo primario de 2.1 m del telescopio de San Pedro Mártir, en Baja California, de 1.5 a 0.5 arcsec (diámetro del círculo que encierra el 80% de la radiación). Durante 1993 y 1994 se realizaron varias pruebas en el telescopio, tendientes a determinar la causa primordial de la pobre calidad de imagen. Después de desarrollar y probar métodos diversos, se encontró que el problema principal es debido a aberraciones de orden bajo (2do. y 3o.), esto es, fundamentalmente astigmatismo (0.53 micras) y coma (0.32 micras). Se llegó a la conclusión de que estas aberraciones son corregibles mediante un soporte activo para el espejo primario y una colimación adecuada del secundario.

Con el método de Roddier[8], que prueba íntegramente el frente de onda en toda la superficie, fué posible determinar que la contribución de aberraciones de orden superior (esfericidad, 0.08 micras) y aberraciones de órdenes mayores debidos al mal pulido del espejo primario, tienen una contribución mínima a la aberración total. Por ésta razón, en marzo de 1994, se concluyó que no sería necesario repulir el espejo primario.

[†] Segundos de arco. Esta es una medida angular que se refiere al ángulo que subtiende la imagen de una estrella en el cielo, es decir, la imagen no es puntual, si no que mide un cierto ángulo en el cielo.

Diversas pruebas mostraron que es posible deformar el espejo primario en cantidades que concuerdan con lo que se espera teóricamente de la elasticidad del material (lo cual es suficiente para deformar el espejo lo requerido para corregir astigmatismo). De hecho, se llegó a la conclusión de que es posible deformar el espejo con un sistema de actuadores que únicamente empujen y que este sistema puede actuar adecuadamente hasta 60 grados cenitales (2 masas de aire).

Esto puede ser suficiente para la mayoría de las aplicaciones, ya que a masas de aire mayores otros factores comienzan a degradar la calidad de la imagen. A pesar de esto, se diseñó y construyó un prototipo de actuador que empuja y jala, pero no funcionó como se esperaba.

Además del método de Roddier para evaluar el frente de onda, se probaron otros (Luna 1993-94, Salas 1995), encontrándose resultados similares a los de la primer prueba, lo que generó más de una posibilidad para determinar los coeficientes de aberración necesarios para un sistema de óptica activa.

II.A. Estructura del proyecto.

Aunque el problema del soporte del espejo primario era el principal factor que deterioraba la imagen, fué necesario de cualquier forma atacar los otros problemas que podían deteriorar la imagen si se quería llegar a tener imágenes de 0.5 arcsec. Algunos de estos factores son: colimación del espejo secundario, seeing local y estabilidad en el guiado.

II.A.1. Seeing local.

Existe una estimación (Proyecto Magallanes) de que el deterioro del seeing como función del calor disipado cerca o debajo de un sitio es de 0.1 arcsec/KWatt. En el edificio del telescopio de 2.1 m, que además tiene una estructura en forma de chimenea, se disipan alrededor de 4 KWatts en línea regulada y 3 KWatts en línea no regulada, por lo que se podría esperar una imagen de 0.7 segundos de arco. Esto significa que si se corrigiera el astigmatismo y coma del telescopio, la imagen quedaría dominada por el seeing local. La apariencia de una imagen puntual en este caso, sería la de una imagen cambiante, cintilante, y no una imagen deformada pero estática. Si queremos mejorar eventualmente la calidad de la imagen, es también necesario controlar la turbulencia local. Durante 1994, el Ing. José C. Avelar, comenzó un proyecto para detener el efecto chimenea en el edificio. Para este efecto se ha construido una estructura de hierro que apoya un techo sólido de madera entre el piso de observación y el telescopio. También se ha instalado un sistema de ductos para ventilación, que eventualmente llevarán el aire caliente a unos 20 metros del edificio fuera de la dirección predominante del viento.

II.A.2. Suspensión activa del espejo primario.

Antes de implementar el nuevo sistema activo, la suspensión del espejo primario constaba de una bolsa de aire, a 1 psi de presión y un cinturón de mercurio. La presión de la bolsa de aire variaba cosenoidalmente con el ángulo cenital, mediante un regulador operado por gravedad, con una precisión del 1%. La propuesta para convertir en activo ese sistema, consistió en substituir la bolsa por un sistema de 18 actuadores distribuidos en dos anillos concéntricos, separados una distancia aproximada de 40 cm entre sí (fig. 2). Esto genera un sistema equivalente al sistema del NTT en cuanto a la razón separación/espesor, es decir, un sistema igualmente rígido localmente. Sin embargo, globalmente estaremos limitados a correcciones de orden menor (astigmatismo y coma), siendo muy limitada la capacidad de corregir esfericidad y órdenes superiores.

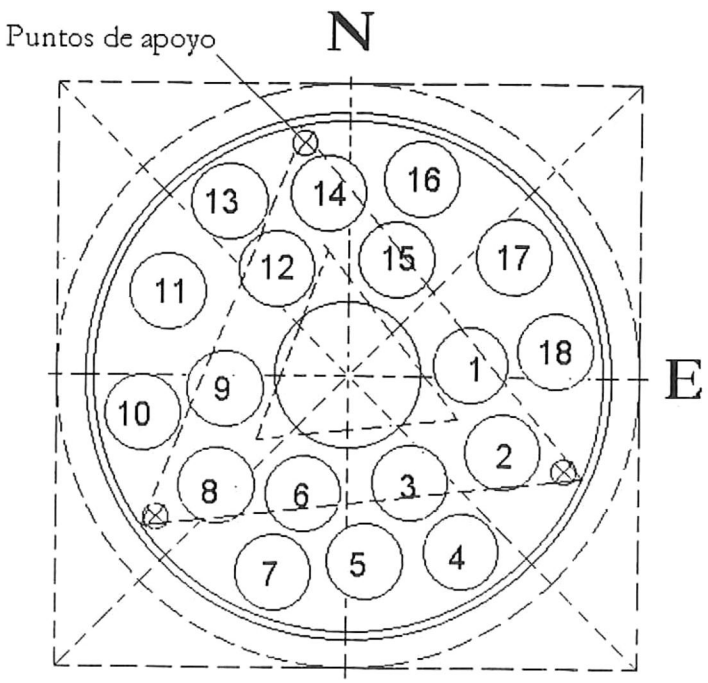


Figura 2. Posición de los 18 actuadores (bolsas de aire) y los tres puntos de apoyo en la celda de soporte del espejo primario.

El sistema funciona inicialmente en forma lo menos intrusiva posible. Para esto, la evaluación de la imagen y determinación de las presiones necesarias en las bolsas, se realiza independientemente con cualquiera de los métodos ya probados. Posteriormente, el sistema únicamente regula la presión necesaria para deformar el espejo. La variación cosenoidal requerida para contrarrestar el efecto de inclinación del espejo, sigue siendo regulada en forma independiente de la variación requerida para deformar activamente el espejo. Esto en el sentido de que, en caso de haber alguna falla con el sistema activo, aun quede operable el telescopio.

El sistema consiste de lo siguiente:

- 18 bolsas de 33 cm de diámetro. Presión máxima sin deformar el espejo: 1.81 psi al cenit. Factor de área de llenado 0.55, 111 Kg de carga por bolsa.
- 18 reguladores de presión, rango máximo 3.5 psi, repetibilidad en lazo abierto 5%.
- 3 celdas de carga para los puntos fijos. Capacidad de 500 Kg máximo con precisión de 0.15% (750 gr). Ruptura mecánica a 300% de carga máxima tal que el peso total del espejo puede descansar sobre estos tres puntos de apoyo.
- 18 transductores de presión, rango 0 a 15 psi con 0.25% de precisión (se requiere controlar presión en las bolsas a 1%).
- Computadora de control. Esta únicamente controla la presión de las 18 bolsas. No es la que evalúa la calidad de la imagen.
- Inclinómetro. Es necesario conocer la inclinación del espejo para 1) calcular la componente del peso soportada en los tres puntos fijos y poder cerrar el lazo externo de posición del espejo, y 2) en caso que la computadora tenga que efectuar la corrección por inclinación para la presión de las bolsas.
- Bombas de aire para mantener la presión.

II.A.3. Estabilidad del guiado.

La consola del telescopio es capaz de guiar con una estabilidad de unos dos pulsos RMS (0.33 arcsec). Esta estabilidad es suficiente para la calidad de imagen que se busca. Sin embargo, al cambiar de consola de control hay que tener cuidado de que esta cifra no aumente.

II.A.4. Automatización de los espejos secundarios.

Para corregir la coma de descentrado (que es la segunda aberración más importante del telescopio) es necesario poder desplazar y controlar la inclinación del espejo secundario con precisiones del orden de unas 5 micras. Actualmente esta operación se realiza manualmente y es sumamente tardada. Como es muy posible que para corregir adecuadamente la imagen del telescopio haya que seguir un proceso iterativo, la limitante impuesta por esta operación resultará en una maniobra muy complicada. Por esto es imperativo automatizar en el futuro los dos espejos secundarios más usados en el telescopio (el f/13.5 y f/7.5).

II.B. Lazos de control del sistema.

El propósito del control de presión en una primer fase, es controlar la presión M_i en cada una de las 18 bolsas (B_i), de tal manera que la carga soportada por cada una de las tres celdas de carga (P_j) sea igual a un valor predeterminado K_e (normalmente 2% del peso del espejo / 3), multiplicado por el coseno del ángulo cenital (z) en cada momento, i.e., $P_j=K_e \cdot \cos z$; j=1...3. Adicionalmente, la presión de cada bolsa se guardará en una tabla que puede ser comunicada y/o leída de/a otra computadora mediante puerto serie. Existen dos lazos de control, que se pueden cerrar o abrir a voluntad; el de presión y el de posición. Un tercer lazo se cerrará por medios ópticos.

II.B.1. Lazo interno o de presión.

El lazo más interno consta de 18 controles de presión para las 18 bolsas. Cada control consiste de un medidor de presión que mide la presión total en cada bolsa y un regulador de diafragma y resorte que proporciona una salida de presión que es la suma de una entrada neumática y una entrada de posición de una perilla, controlada digitalmente por un motor de pasos (fig 3). De esta manera, si la computadora no funciona, el regulador únicamente seguirá la entrada neumática, proporcionada por el inclinómetro, y mantendrá adecuadamente la presión en cada bolsa en lazo abierto. Esta presión es justamente la requerida para soportar la componente normal del peso del espejo, que varía como el coseno de z. Con la computadora funcionando,

se pretende sumar a la componente del peso una fuerza adicional que tiene el propósito de deformar al espejo.

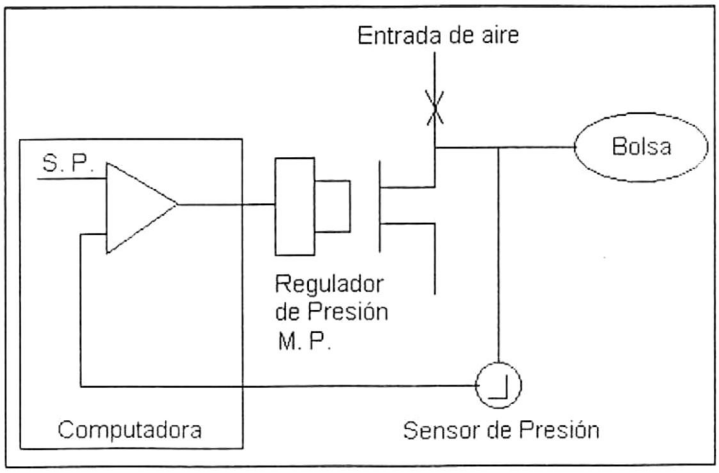


Figura 3. El lazo interno regula la presión en cada una de las bolsas de tal forma que tengan el valor deseado (S.P.) con una precisión mejor al 1%.

Esta última no dependerá del cos z, ya que la deformación que se quiere inducir no depende de la inclinación del espejo (en principio). Para realizar el control, se compara la señal deseada de presión S_i (o S.P. según la fig. 3) con la lectura de los medidores de presión restándole la componente normal del peso, para generar una señal de error:

$$E_i = S_i - (M_i - H_i \cos z). \qquad (1)$$

Aquí, H_i es una tabla de valores de peso que debe soportar cada bolsa, ya que obviamente son distintos. Si no se desea que la bolsa i-ésima se deforme, se tendrá que $S_i=0$, lo cual dejará a la señal de error únicamente controlando la diferencia entre lo medido y el peso del espejo. Si en un momento dado se determina que debe deformarse una porción del espejo, entonces S_i tomará algún valor positivo o negativo (en las mismas unidades que M_i). En ese momento se genera una señal de error. En este punto se tiene la opción de dejar abierto el lazo de control o cerrarlo, para propósitos de corregir algún pequeño error de programación. Si el lazo de control está cerrado, deberá procederse a mandar pulsos al motor de pasos para cancelar el error. Como el ancho de banda de los motores de pasos es bajo, es conveniente mandar pocos pulsos a la vez, de tal manera que en el siguiente paso por la rutina de control se pueda volver a evaluar si los pulsos anteriores fueron suficientes o se requiere mandar más, ya sea para subir o disminuir la presión. Se debe notar que al incrementar la presión en una de las bolsas se debe alterar un poco en todas la demás, esto para mantener la presión total constante. De no hacerse esto, el espejo flotaría.

Es necesario guardar en disco la posición real de cada uno de los motores de pasos de los reguladores, con el fin de no exceder la carrera de cada motor y así no forzarlos más allá de sus límites. Esto también ayuda a tener una idea de si un regulador deja de funcionar; si M_i no cambia a pesar de variar significativamente la posición del motor, o bien si ocurre lo contrario, que M_i comience a cambiar fuertemente sin que se mueva el motor de pasos.

II.A.2. Lazo externo o de posición.

En este lazo se pretende asegurar que la posición del espejo primario no cambie. Esto se logra asegurándose que el espejo no empiece a flotar o hundirse en sus 3 puntos definidores P_j (fig. 4). Para tal efecto es necesario medir la carga en cada punto de apoyo y compararla con el peso predeterminado K_e multiplicado por el cos z. Así tenemos 3 señales de error de peso:

$$E_{pj}=P_j-K_e \cos z \qquad (2)$$

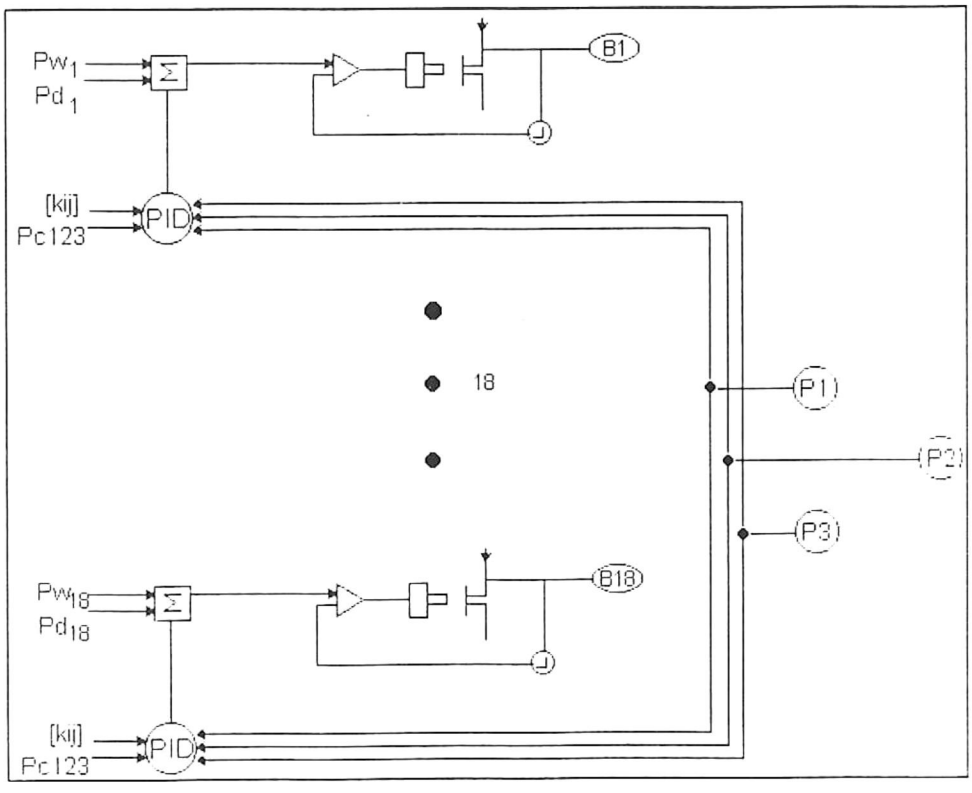


Figura 4. El lazo externo modifica el valor deseado de cada lazo interno como función de la fracción del peso del espejo correspondiente a cada bolsa (Pw_i), la fracción del

peso del espejo correspondiente a cada celda de carga ($Pc_{1,2,3}$), la matriz de transformación (k_{ij}) que define un plano de presiones, y la deformación deseada (Pd_i). Se utiliza un algoritmo PID para cerrar el lazo externo. La referencia al zenit está dada por un inclinómetro.

Se tuvo que tomar precaución de asegurar que la precisión de las celdas de carga (fig. 5) fuera suficiente para medir fracciones del 2% (13 Kg), aún a grandes valores de z. En caso de que el espejo comenzara a flotar, el valor de todos los P_j bajarán y consecuentemente hay que desinflar todas las bolsas. Esto se logra mejor modificando los valores de S_i a través de una matriz k_{ij} que opera sobre este error, esto es:

$$S_i=D_i-k_{ij}E_{pj} \qquad (3)$$

donde D_i es la deformación requerida para cada porción del espejo. En caso que el espejo no flote o se hunda, o bien si este lazo está abierto, se tendrá Si=Di.

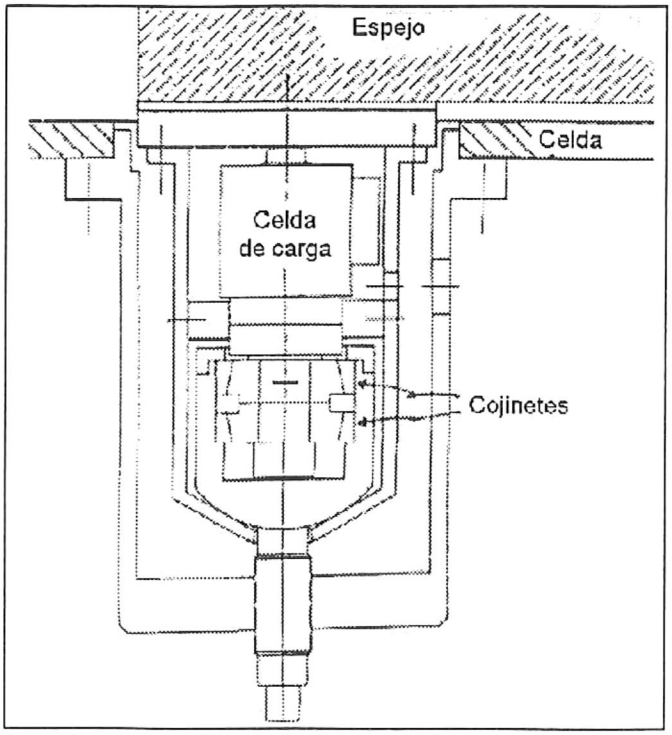


Figura 5. Diseño macánico de los puntos de apoyo. Una celda de carga mide el peso del espejo a través de un punto de contacto esférico. Un par de cojinetes permite el movimiento rotatorio.

En caso de que se esté realizando control, puede darse el caso de que sólo una celda de carga se mueva con respecto a las otras. Esto se puede deber a la acción del viento, o de otros factores. En tal caso es necesario nivelar el espejo, para lo cual es necesario modificar la presión principalmente en las bolsas que están cerca de la celda correspondiente, esto es, en el sector adecuado. Esto se realiza con ayuda de la matriz k_{ij} que propaga linealmene la perturbación de la celda j a cada una de las bolsas i, tomando en cuenta la distancia a la que se encuentran, y así su contribución relativa al error.

Para el funcionamiento correcto de este lazo de posición fué necesario implementar un algoritmo PID (Proporcional, Integral, Derivativo), que dosifica pulsos de correción a los reguladores de presión de las bolsas, no solo con base en el error (proporcional), sino también tomando en cuenta la historia pasada de su comportamiento (integral) y la rapidez con que se está modificando la presión actualmente (derivativo). Los pesos relativos de éstas tres contribuciones a los pulsos de corrección, se ajustaron empíricamente, logrando una convergencia excelente en tiempos razonablemente cortos.

II.B.3. Lazo óptico.

Este lazo estará cerrado en un futuro por un sensor de frente de onda y otra computadora más poderosa que finalmente determinará los valores deseados de deformación D_i. Inicialmente, los valores D_i quedan controlados manualmente, mediante un programa interactivo.

II.C. El programa de control.

II.C.1. Interfase.

Este programa procura una interfase fácil de usar mediante el mouse de la computadora. El despliegue consiste de un modelo a escala del espejo primario y de las 18 bolsas localizadas en la posición exacta en que se encuentran en el telescopio, todas representadas por círculos, en cuyo interior se despliega constantemente el valor deseado de la presión y el valor de la lectura medida por los sensores. De esta forma se puede saber rápidamente qué bolsa(s) no está(n) funcionando correctamente (fig. 6). Adicionalmente, cuando el programa detecta una diferencia mayor de cierto límite entre estos dos valores en alguna de las bolsas, inmediatamente dá una señal auditiva y cambia el color de la bolsa afectada a rojo, cuando normalmente su color es azul.

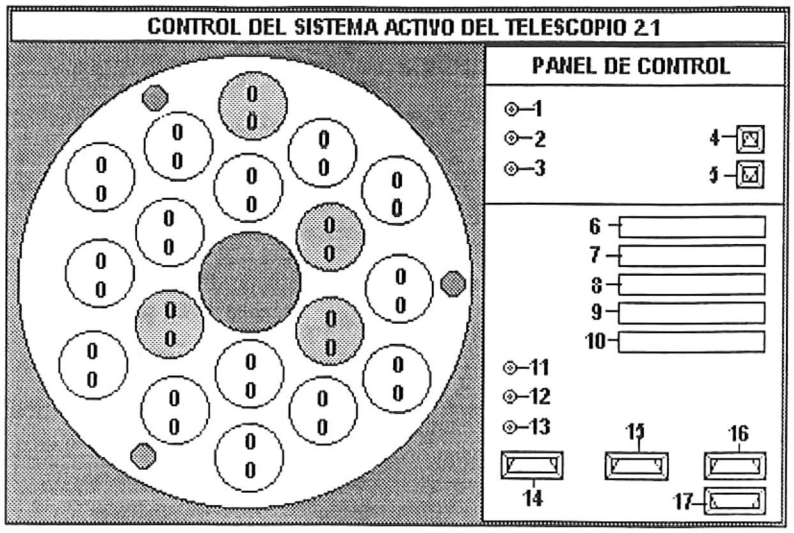


Figura 6. Esquema de la interfase. Ver la explicación de los componentes en el texto. Las bolsas con un color más obscuro están seleccionadas.

Junto al modelo del espejo y las bolsas, se incluye una tabla de controles manuales operables tanto con el mouse como con el teclado. Estos controles y despliegues incluyen:

1) Radio-botón para activar o desactivar el control de posición. Cuando está activado su color es verde, de lo contrario es rojo.
2) Radio-botón para activar el modo de presión global, cualquier incremento manual que se haga en la presión afectará a todas las bolsas. Al activarlo, cambiará su color a verde y se desactivará el modo de presión individual (3).
3) Radio-botón para activar el modo de presión individual, cualquier incremento manual que se haga en la presión afectará sólo a las bolsas que estén

seleccionadas. Al activarlo, cambiará su color a verde y se desactivará el modo de presión global (2).

4) Botón para aumentar la presión, la cantidad que esté establecida en el despliegue de Incremento en Presión (10). El incremento sólo afectará a las bolsas seleccionadas.
5) Botón para disminuir la presión, la cantidad que esté establecida en el despliegue de Incremento en Presión (10). El decremento sólo afectará a las bolsas seleccionadas.
6) Despliegue para mostrar el coseno del ángulo zenital medido.
7) Despliegue para mostrar la presión en la celda de carga 1.
8) Despliegue para mostrar la presión en la celda de carga 2.
9) Despliegue para mostrar la presión en la celda de carga 3.
10) Despliegue interactivo que muestra el valor actual de la variable Incremento en Presión. Al seleccionar con el mouse este despliegue, el programa pide que se introduzca el incremento en presión deseado a través del teclado.
11) Radio-botón para activar o desactivar el modo de normalización.
12) Radio-botón para activar o desactivar el modo de control de presión.
13) Radio-botón para activar o desactivar el modo de pintado de presión deseada. Cuando este botón está activado, el despliegue interno de cada una de las bolsas mostrará la presión que se desea que tenga esa bolsa, aparte de la presión real que se está midiendo por medio del sensor de presión. Esto con el propósito de poder monitorear posibles oscilaciones alrededor del valor

deseado, lo cual puede suceder cuando la forma en que el programa modifica las presiones mediante señales a los motores de pasos no es la adecuada. Si está desactivado, los despliegues internos de las bolsas mostrarán la contribución de deformación a la presión.

14) Este botón tiene el propósito de llevar a las bolsas a su presión normal, esto es, iguala las contribuciones de deformación y fuerza en la presión a cero.

15) Este botón tiene el propósito de desinflar las bolsas. Esto se lleva a cabo al igualar la contribución de deformación al negativo de la contribución por peso y asignar un valor de cero a la contribución de fuerza.

16) Este botón tiene el propósito de perdirle al programa que lea del disco un arreglo de valores que son la contribución a la presión correspondientes a la deformación deseada. Su primer acción es la de preguntar en qué archivo se encuentran estos valores. Está definido un nombre por default para este archivo (deforma.bag), de modo que si el programa no recibe el nombre, o no puede encontrar el archivo que el usuario le proporciona, se leerá el archivo default, indicando lo sucedido. Después de haber leído los valores deseados, o en su caso los de default, el programa actualiza las presiones.

17) Por último, el botón de Salida. Este botón tiene el propósito de terminar el programa y salir al sistema operativo. Antes de esto, regresa a todos los motores de pasos a su posición inicial. Esto se logra al guardar en disco periódicamente el número de pasos que se han dado en cada uno de los motores, así se puede saber la distancia (número de pasos) a la que se

encuentran los motores de la posición inicial que tenían cuando arrancó el sistema.

La forma de seleccionar las bolsas individualmente es oprimir el botón izquierdo del mouse sobre la bolsa que se desee seleccionar, la que cambiará de color indicando que el proceso fué exitoso. Para deseleccionarla se deberá presionar el botón derecho sobre ella, y regresará al color original. Se pueden seleccionar todas las bolsas que se deseen, una tras otra, aunque si se desea seleccionar todas es mejor usar el botón designado para esta tarea localizado en el tablero de control.

II.C.2. Control.

El algoritmo principal usado para el control del sistema fue el siguiente:

1) Medir el cos z.

2) Medir Pj; j=1...3 (la presión en las tres celdas de carga).

3) Para i=1...18.

    a) Medir Mi.

    b) Calcular Si (algoritmo PID).

    c) Calcular Ei (lazo interno).

    d) Actuar sobre la bolsa i.

4) Ir a 1)

II.C.3. Ciclo principal.

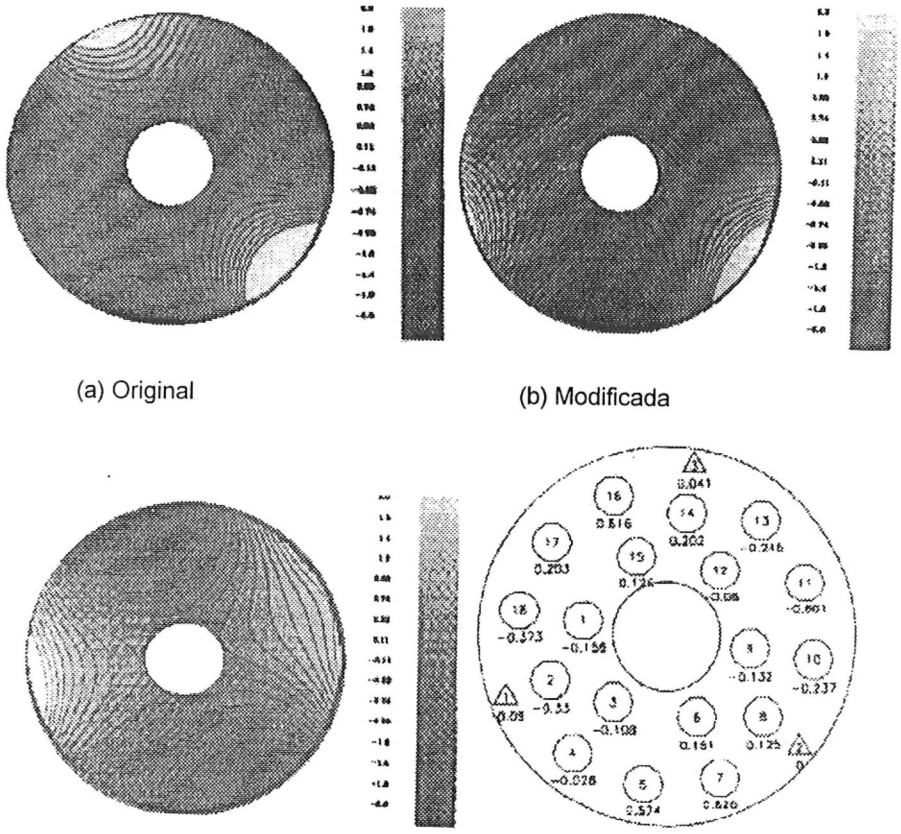
El diagrama de flujo se puede consultar en el apéndice A y el código fuente se puede consultar en el apéndice C. El ciclo principal está en la función *main()* la cual llama a otras cinco que se encargan de distintas funciones. La primer función que se llama es *inicia()*, la cual se encarga de la parte gráfica (despliegue de los elementos) así como de la inicialización de los objetos (La jerarquía de las clases creadas está en el apéndice B). Las siguientes tres funciones que se llaman son *lee_pasos()*, *lee_deforma()* y *lee_ks()* que se encargan de leer las posiciones de los 18 reguladores, leer los valores iniciales para la contribución a la presión debida a deformaciones y leer unas constantes que se usan en el control de presiones. La última función que se llama, *procesa()*, es la que se encarga del control. Esta función contiene un ciclo que se repite hasta que se sale del programa. Lo primero que hace es leer el ángulo cenital del telescopio a través del inclinómetro, después lee la presión que miden las celdas de carga, y sensa continuamente la actividad del mouse, esto es, su posición y si se presionó alguno de los botones sobre alguno de los componentes que integran el despliegue (radio botones, botones, campos de escritura, bolsas, celdas de carga, etc.), si es así, se encarga de realizar la función correspondiente. Se llama a la función *control()*, que se encarga de la salida digital de las instrucciones por el puerto serie. Se leen las presiones de las 18 bolsas y se actualizan las fuerzas correspondientes, así como las presiones deseadas.

A la hora que el usuario desea salir del programa se hacen una serie de operaciones, entre las cuales están la liberación de la memoria usada, salvar las posiciones finales de los reguladores o bien regresarlos a su posición inicial.

III. Resultados.

Se usaron tres métodos diferentes para probar y calificar las aberraciones ópticas. Dos de ellos siguieron la aproximación del sensado de curvatura y el otro fué la prueba de bi-Ronchi. Mientras que los métodos basados en el sensado de curvatura buscaban una solución para el frente de onda resolviendo una ecuación diferencial de segundo orden, la prueba bi-Ronchi deliberaba sobre la primera derivada de este, esto es algo similar a la prueba de Hartmann. Los tres métodos calculan un conjunto de coeficientes en la expansión radial de los polinomios de Zernike del frente de onda.

Se probaron diferentes configuraciones para presiones aplicadas sobre los 18 actuadores y se midieron las deformaciones obtenidas. Esta medición es la diferencia del frente de onda original (sin corregir) y el del frente de onda obtenido después de que el patrón de fuerzas fué modificado. En la figura 7 se muestra este frente de onda (a original, b modificado, c diferencia) juntas con el patrón de presiones para deformar el espejo (d). El frente de onda en esta figura se obtuvo por el método de separación de variables de sensado de curvatura.



(a) Original (b) Modificada

(c) Diferencia (d) Presiones aplicadas

Fig. 7. Efecto sobre el frente de onda al aplicar las presiones: (a) frente de onda original (micras); (b) frente de onda obtenido después de haber aplicado las presiones en los actuadores; (c) diferencia entre (b) y (a); (d) conjunto de las presiones aplicadas a cada actuador.

Como podemos ver, esta es una relación cercana entre la deformación obtenida (c) y el patrón de fuerzas aplicadas (d). Esto se expresa mejor en la figura 8, donde podemos ver una dependencia lineal entre la deformación obtenida y las presiones

adicionales aplicadas a cada actuador (los actuadores están numerados en esta gráfica).

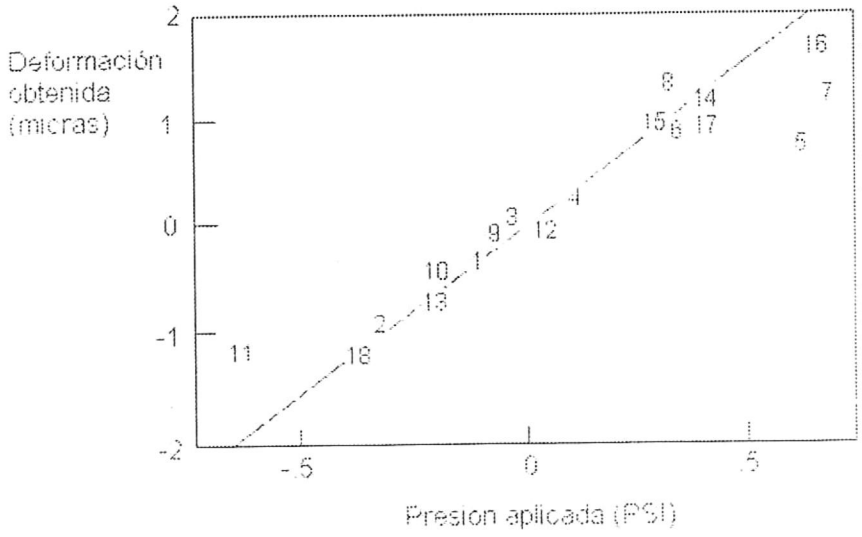


Fig. 8. Gráfica de las deformaciones obtenidas en la posición de los actuadores en función de las presiones aplicadas. Cada actuador está numerado. La linea sólida tiene una pendiente de 3 micras/psi.

La función de dispersión puntual que usualmente era de tres picos como resultado de coma y astigmatismo, ahora resulta ser de uno solo, como se puede ver en la reconstrucción de la función en la figura 9, obtenido con el método de análisis del frente de onda de bi-Ronchi. De los 1.2 arcsec se ha obtenido actualmente cerca de 0.5 arcsec.

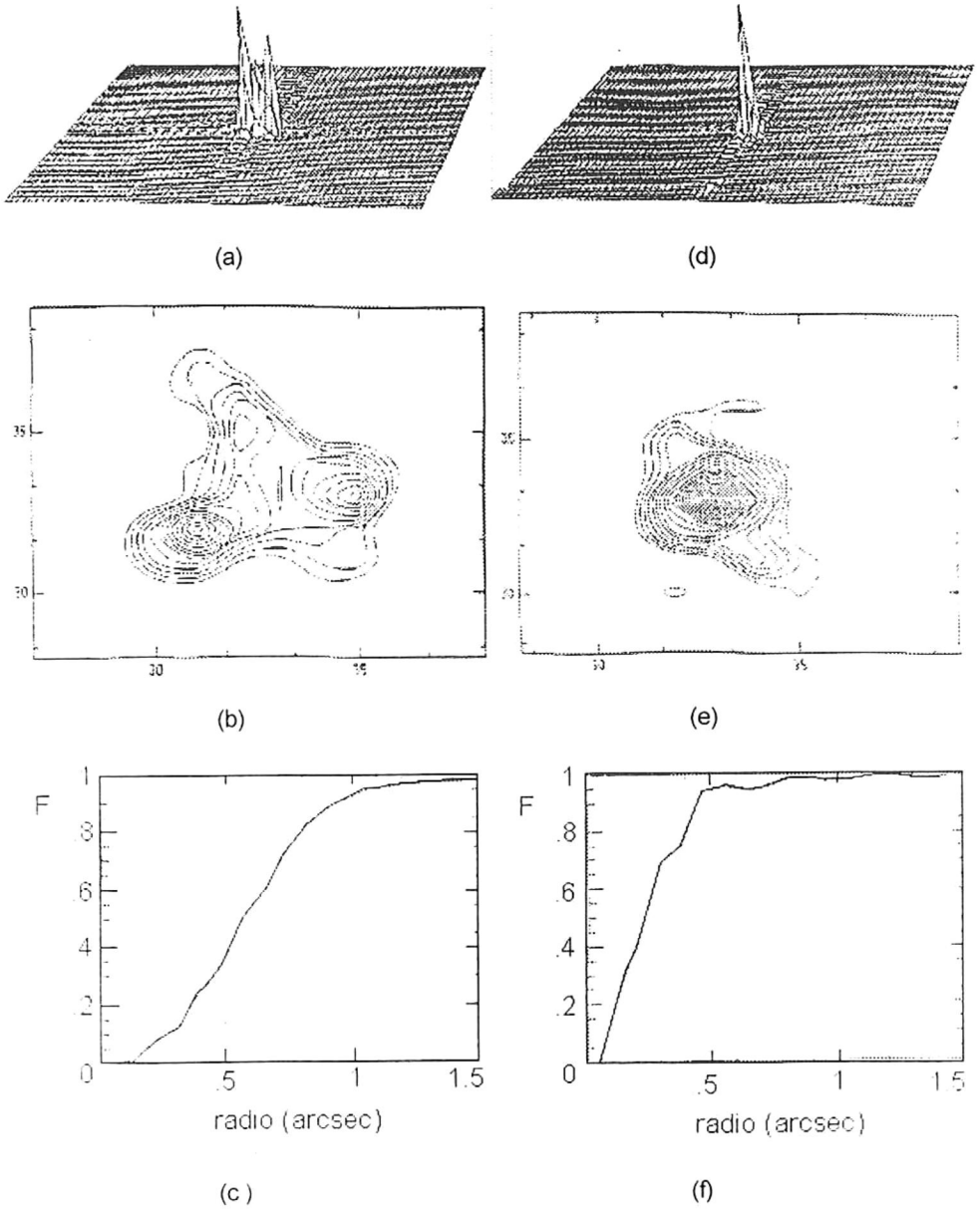


Fig. 9. Reconstrucción PSF de los análisis del frente de onda, antes (a)-(c) y después (d)-(f) de la introducción del sistema activo; (a) y (c) muestran la superficie del PSF, (b) y (e) muestran lineas de contorno con una escala de 0.15 arcsec por pixel, y (c) y (f) muestran el flujo integrado contenido en un circulo de radio r.

En Noviembre de 1995, mientras se aprendía más sobre el control de las aberraciones mixtas, y con condiciones de seeing no muy propicias, fueron tomadas imagenes a 0.7 segundos de arco (M. Tapia). La calidad de la imagen es ejemplificada por una imagen infrarroja de una región Ultra-Compacta HII G173.7 (fig. 10) obtenidas por la cámara infrarroja CAMILA en la configuración f/13.5.

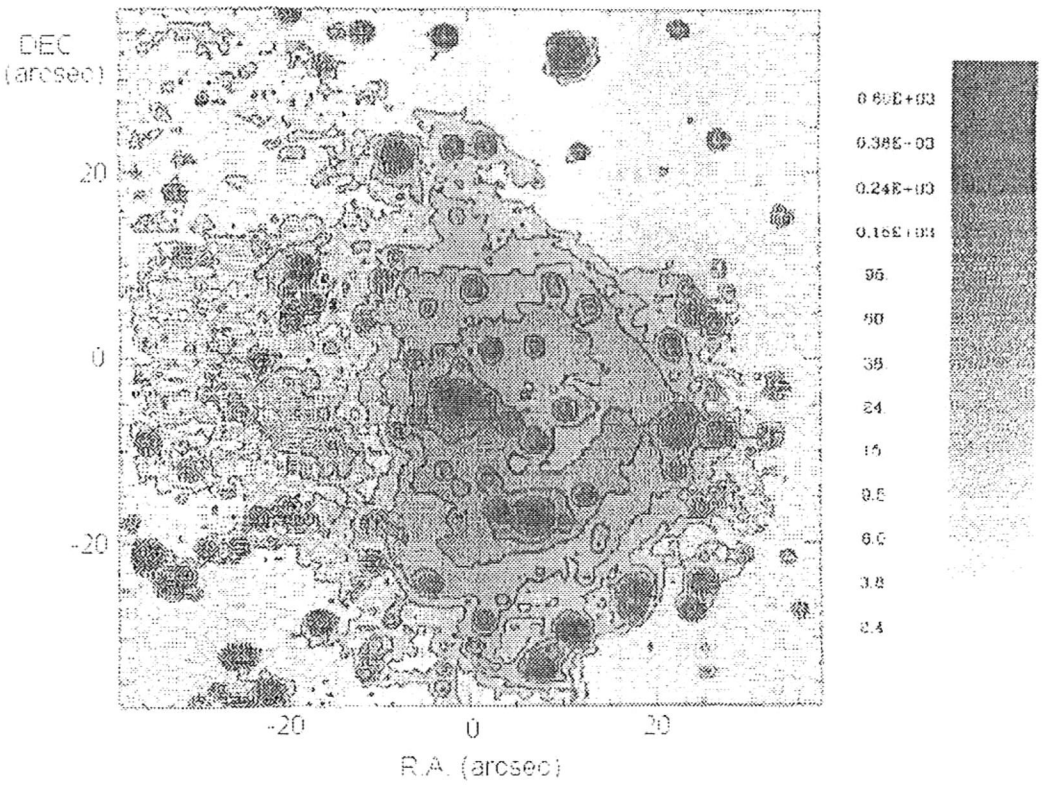


Fig. 10. Imagen de la región Ultra-Compacta HII G173.7, obtenida con la configuración corregida f/13.5 y la cámara infrarroja CAMILA. Los contornos están espaciados logaritmicamente con intervalos exponenciales de 0.2. El offset de la fuente central está dado en arcsecs para la declinación (DEC) y la ascención recta (R.A.).

IV. Conclusiones.

Se ha logrado el control activo del espejo primario de 2.1 m del telescopio localizado en el observatorio de San Pedro Mártir. El sistema ha mostrado sus capacidades al corregir el astigmatismo, y correcciones a aberraciones de orden más alto esperan para un estudio más detallado.

Este primer sistema fué colocado en la parte norte de la platina del telescopio obteniéndose resultados muy favorables en las pruebas hechas por Esteban Luna. Salvador Cuevas y Luis Salas, encontrándose una mejoría muy notable en la calidad de la imagen del telescopio, avanzando de una calidad de imagen de 1.2 arcsec a 0.5 arcsec para el espejo secundario de f/13.5.

Esto dejó algo muy bueno qué decir de la tecnología que se desarrolla actualmente en México, ya que en otros países se había intentado hacer algo similar sin conseguir ningún resultado favorable.

Aunque el sistema de control activo (electrónica) quedó instalado solamente como prototipo de enrollado, esto no fue limitante para que trabajara al 100% de eficiencia. El programa de control sufrió una modificación recientemente; se cambió el despliegue gráfico de la computadora localizada en el telescopio, por uno en modo texto, con el propósito de aumentar la velocidad de control. Ahora, el despliegue gráfico se instaló en una computadora localizada en el cuarto de observación. Esta computadora se comunica a través del puerto serie con la computadora en el telescopio mandando los comandos necesarios para leer o medir los valores

deseados, tales como la presión en las 18 bolsas, la presión en las 3 celdas de carga, etc..

V. Literatura.

Referencias.

[1] Fried, D. L., 1996, J. opt. Soc. Am., **56**, 1372.

[2] Roddier, F., 1984, Proc. ESO *Workshop on Site Testing for Future Large Telescopes*, La Silla, 1983 (Garching: ESO), p. 193.

[3] Wilson, R. N., and Mischung, K. N., 1987, Astron. Astrophys.

[4] Enars, D., Mischung, K. N., Schneermann, M., and Wilson, R. N., 1986, *Advanced Technology Optical Telescopes III*, Tucson, SPIE, Vol. 628, p. 528.

[5] Franza, F., Le Luyer, M., and Wilson, R. N., 1977, *3.6 m Telescope: The adjustment and test on the sky of the PF optics with the gascoigne plate correctors*, ESO Technical Report No. 8.

[6] Pearson, J. E., Freeman, R. H., and Reynolds, H. C., 1979, Applied Optics and Optical Engineering, Vol. 7, p.262.

[7] Woolf, N. J., 1984, Proc. IAU Colloquium No. 79. *Very Large Telescopes, their Instrumentation and Programs*, Garching, p. 221.

[8] Roddier, C., and Roddier, F., 1993, *Wave-front reconstruction from defocused images and testing of ground-based optical telescopes*, J. Opt. Soc. Am. A., Vol. 10, No. 11.

Bibliografía.

L. Salas y M.H. Pedrayes 1992, Calidad de la imagen de un telescopio con espejo primario deformable y soportes tipo NTT.

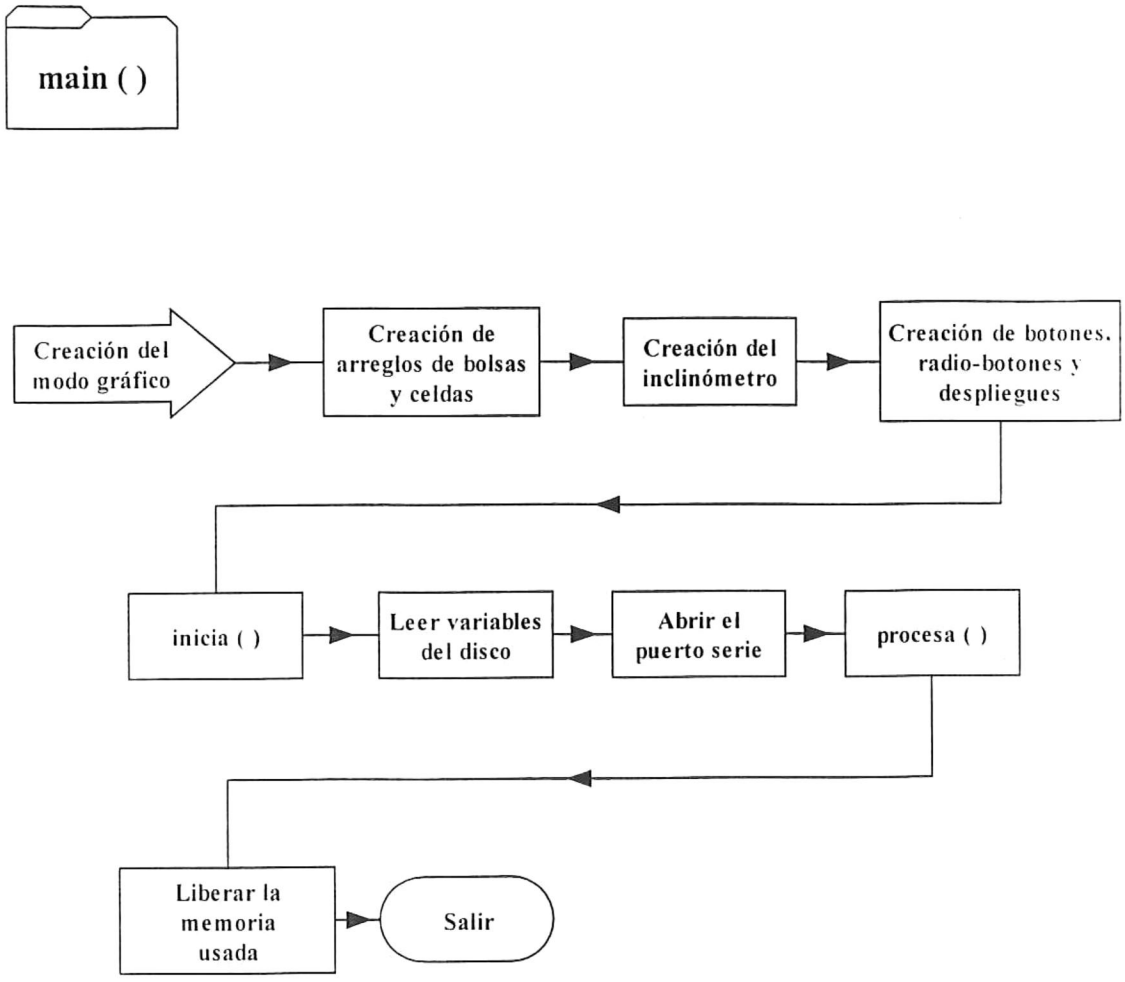
Cuevas, S., 1992, Primera reunión de Ingeniería del Telescopio Mexicano de Nueva Tecnología.

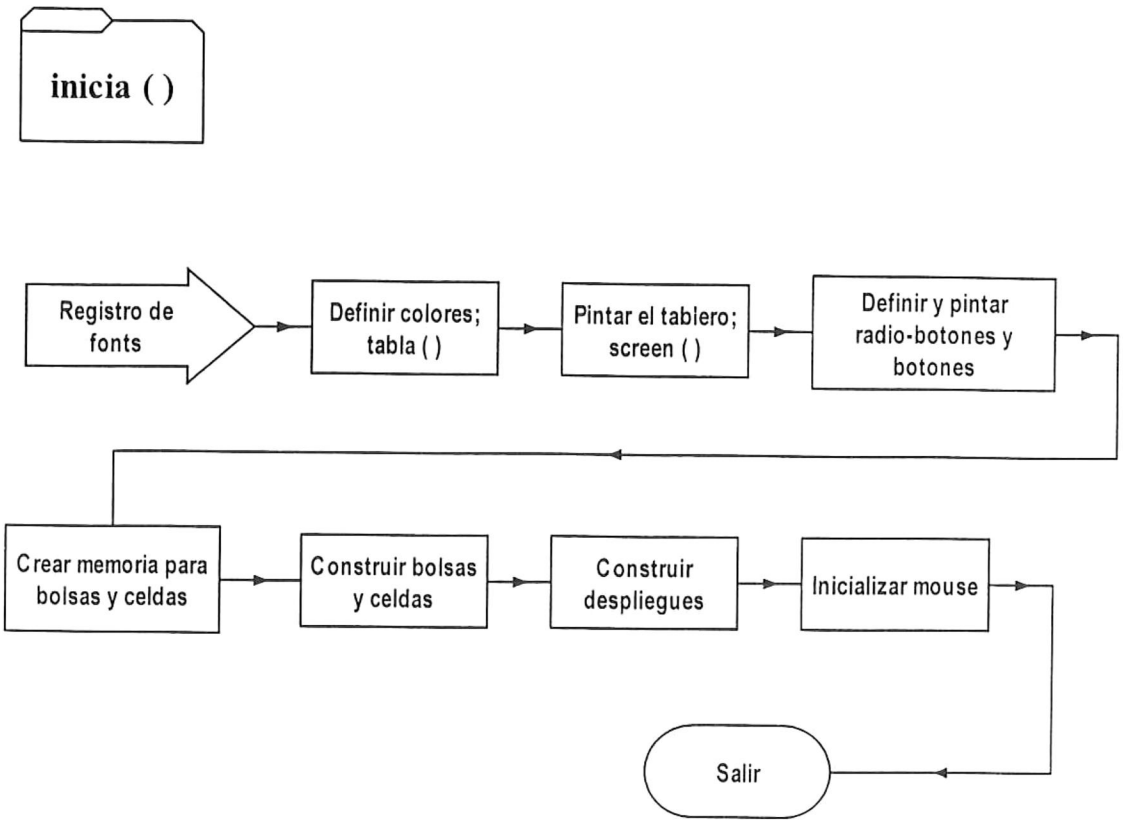
Echeverría, J., 1992, Primera reunión de Ingeniería del Telescopio Mexicano de Nueva Tecnología.

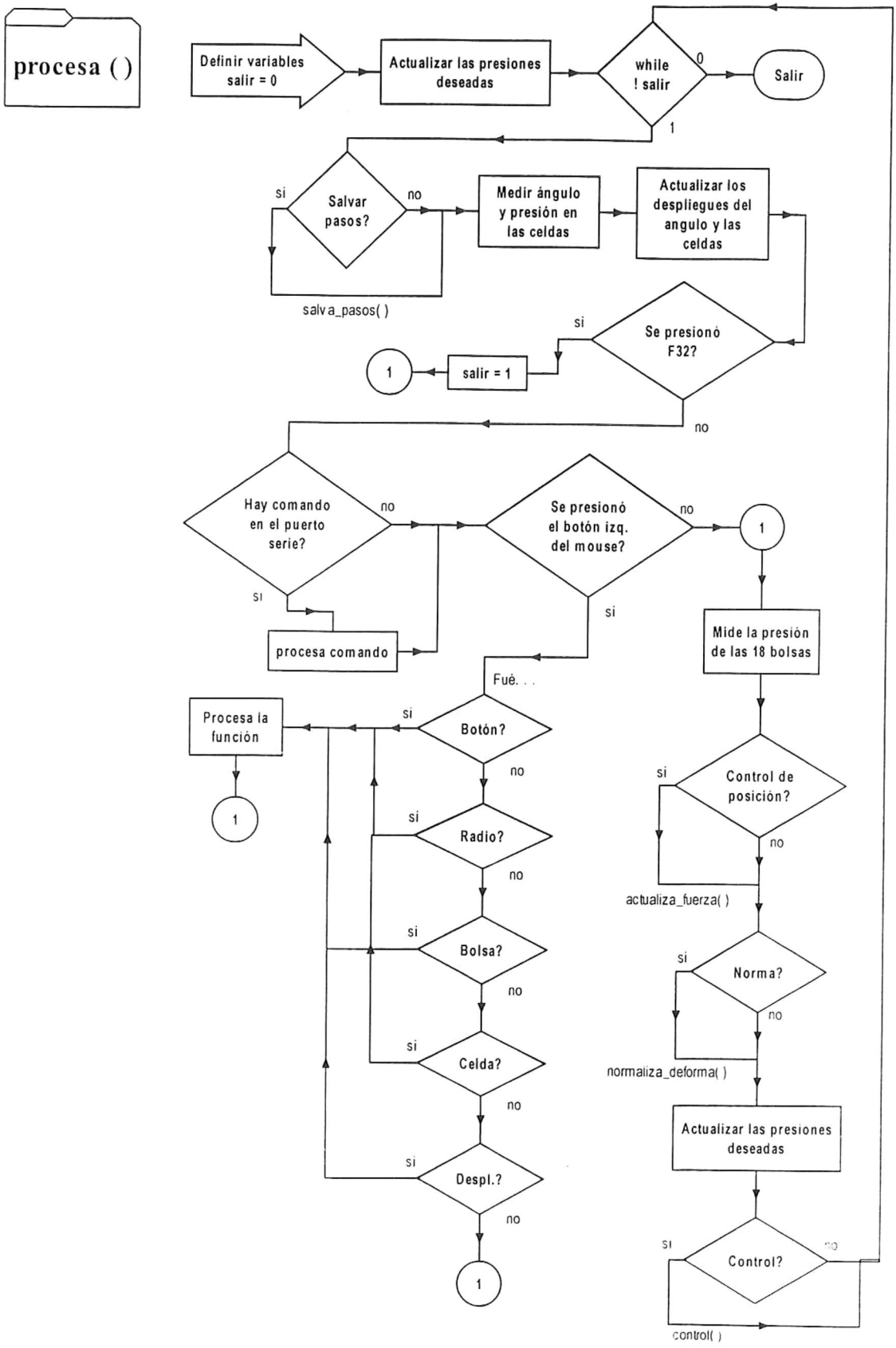
L. Salas, L. Gutiérrez, M. H. Pedrayez, J. Valdez, M. Carrillo, C. Carrasco, et al., 1995, Active Primary Mirror Support for the 2.1-m Telescope at San Pedro Mártir Observatory, UNAM. Applied Optics, **36**-16, 3708 (1997).

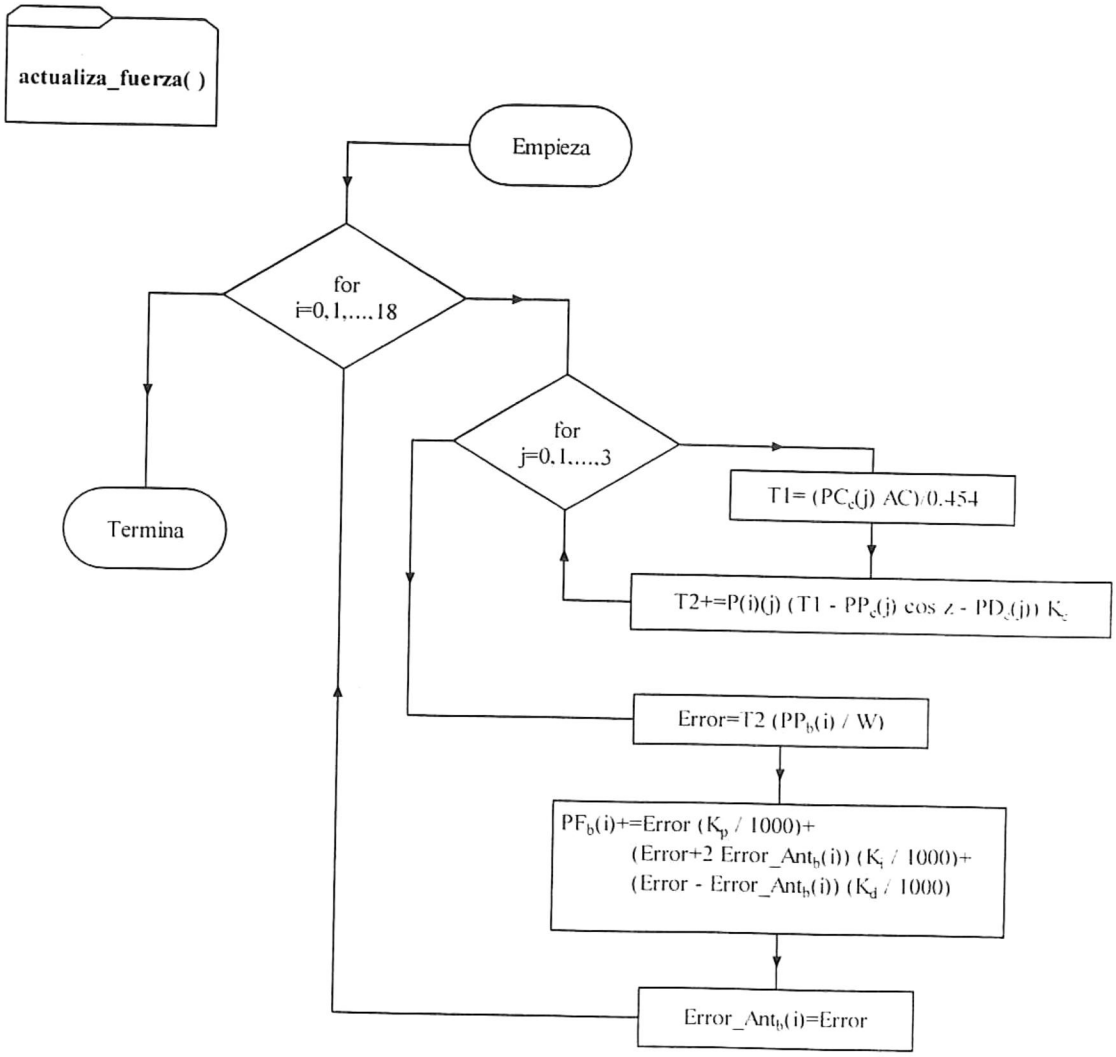
Apéndice A. Diagramas de flujo.

- main ( )
- inicia ( )
- procesa ( )
- actualiza_fuerza ( )
- control ( )

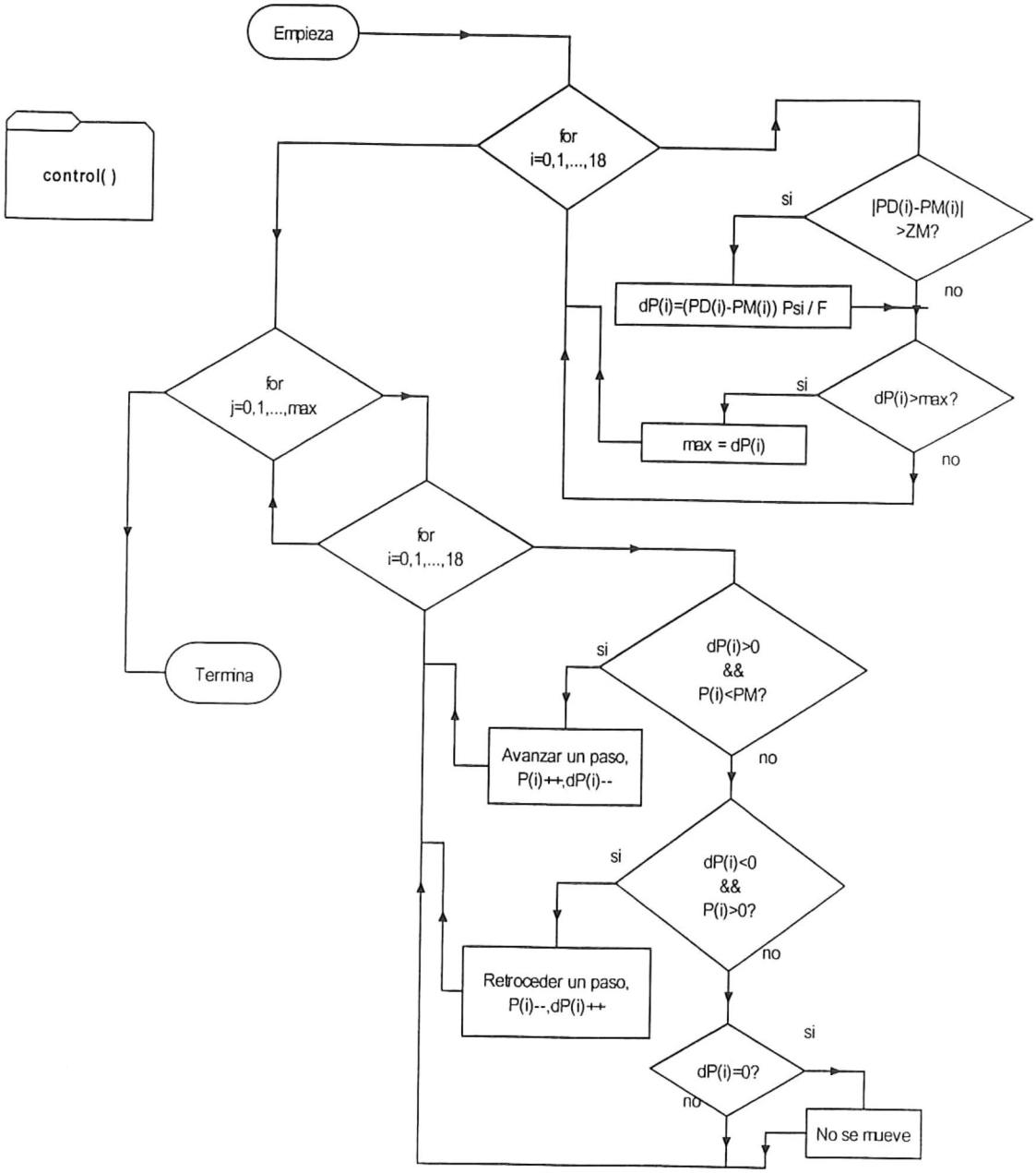

main ( )
Creación del modo gráfico
Creación de arreglos de bolsas y celdas
Creación del inclinómetro
Creación de botones, radio-botones y despliegues
inicia ( )
Leer variables del disco
Abrir el puerto serie
procesa ( )
Liberar la memoria usada
Salir


inicia ( )
Registro de fonts
Definir colores; tabla ( )
Pintar el tablero; screen ( )
Definir y pintar radio-botones y botones
Crear memoria para bolsas y celdas
Construir bolsas y celdas
Construir despliegues
Inicializar mouse
Salir


procesa ( )
Definir variables
salir = 0
Actualizar las presiones deseadas
while
! salir
0
Salir
1
si
Salvar pasos?
no
salva_pasos( )
Medir ángulo y presión en las celdas
Actualizar los despliegues del angulo y las celdas
si
Se presionó F32?
salir = 1
1
no
Hay comando en el puerto serie?
no
si
procesa comando
Se presionó el botón izq. del mouse?
no
si
1
Mide la presión de las 18 bolsas
Fuè. . .
Procesa la función
si
Botón?
no
si
Radio?
no
si
Bolsa?
no
si
Celda?
no
si
Despl.?
no
1
1
si
Control de posición?
no
actualiza_fuerza( )
si
Norma?
no
normaliza_deforma( )
Actualizar las presiones deseadas
si
Control?
no
control( )

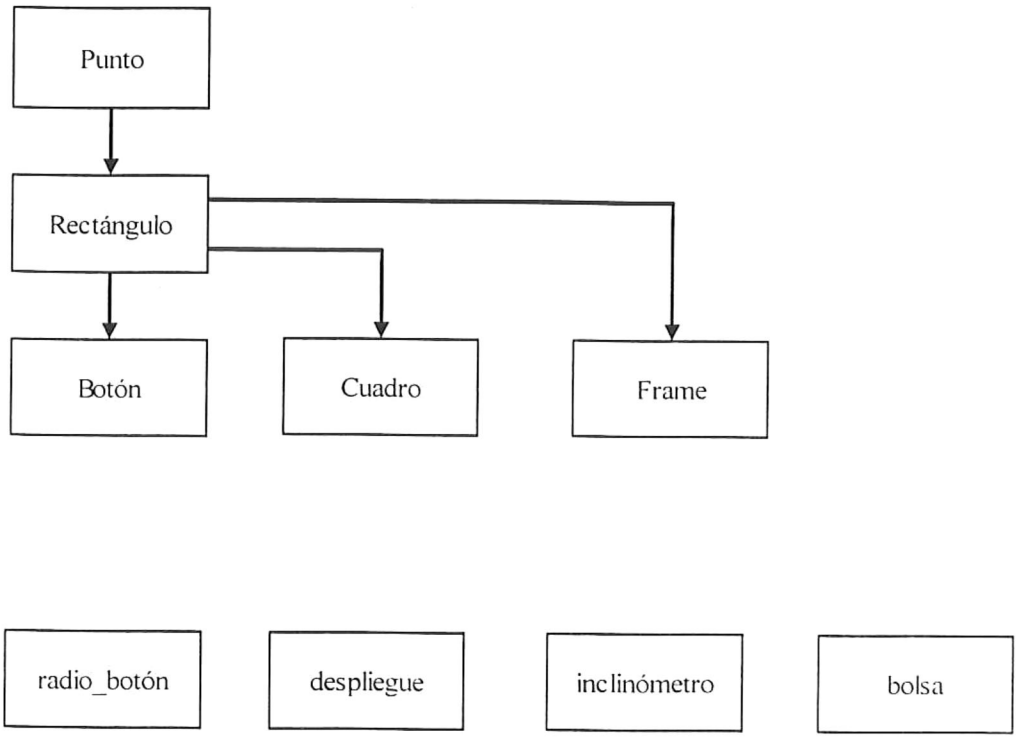


$PC_c(j)$=Presión medida en la celda de carga j
AC=Area de la celda
$PP_c(j)$=Peso Total * 0.02 / AC * 3
$PD_c(j)$=Deformación en la celda j
$PP_b(i)$=Peso que debe soportar la bolsa i-ésima
W=Peso promedio
$PF_b(i)$=Contribución de fuerza a la presión en la bolsa i-ésima
K_p=Constante Proporcional
K_i=Constante Integral
K_d=Constante Derivativa



PD(i)=Presión deseada en la bolsa i-ésima
PM(i)=Presión medida en la i-ésima bolsa
dP(i)=Pasos a dar correspondientes a la i-ésima bolsa
P(i)=Pasos dados correspondientes a la i-ésima bolsa

Apéndice B. Jerarquía de Clases.



Apéndice C. Código fuente.

- ctrlbags.cpp
- ctrlbags.h
- angulo.h
- angulo.cpp
- bag.h
- bag.cpp
- grafbags.h
- grafbags.cpp
- globales.h

ctrlbags.cpp

```
#include <string.h>
#include "ctrlbags.h"
#include "globales.h"
#include "..\serie.h"
//#define SIMULA

//*****************************************************************************
//                                                          FUNCION  PRINCIPAL
//*****************************************************************************

void simula();
void lee_ks();
void norma_selected();
void presion_selected();
void cual_selected();
void posicion_selected();
void all_selected();
void indiv_selected();
void bolsa_selected(int i);
int subprocesa(char *S);
void norma(char *S);
void presion(char *S);
void posicion(char *S);
void cual(char *S);
void selec_bolsa(char *S);
void pasos(char *S);
void sube_bolsa(char *S);
void baja_bolsa(char *S);
void presion_normal();
void presion_cero();
void ld();
void lm();
void lef();
void lc();
void la();
void def(char *S);

        InitGraf screen;  //Inicio de modo grafico
        bolsa *Bolsa;  //Arreglo de bolsas
        bolsa *Celda;  //Arreglo de celdas
        inclinometro Inclinometro;  //Crea el inclinometro
        double peso_estimado=0;
        radio_boton radio_all_selected(460,93,"Presion Global");
        radio_boton radio_pres_indiv(460,111,"Presion Individual");
        radio_boton radio_posicion(460,75,"Control de Posicion");
        radio_boton radio_norma(460,280,"Normalizacion");
        radio_boton radio_presion(460,298,"Control de Presion");
        radio_boton radio_cual(460,316,"Pinta presión deseada");
//-----------------Definicion de Botones-----------------------------------
        Boton salida(570, 450, 55, 15, none, "SALIR", salir);
        Boton selctd_up(600, 88, 15, 15, up, NULL, presion_up_all);
        Boton selctd_dn(600, 108, 15, 15, dn, NULL, presion_dn_all);
```

```
        Boton about(530, 450, 15, 15, none, "?", About);
        Boton ok(450, 450, 55, 15, none, "O K", OK);
        Boton pres_normal(450, 340, 80, 15, none, "Pres. Normal", Normal_pres);
        Boton pres_0(540, 340, 80, 15, none, "Pres. Cero", Cero_pres);
        Boton bot_lee_deforma(450,365, 80, 15, none, "Lee deforma", Load_deforma);
//------------------Definicion de Despliegues---------------------------------
        despliegue desp_angulo_z(540, 140, 80, 20, WHITE, Inclinometro.angulo_cenital);
        despliegue desp_celda_1(540, 165, 80, 20, WHITE, Celda[0].Pres_medida);
        despliegue desp_celda_2(540, 190, 80, 20, WHITE, Celda[1].Pres_medida);
        despliegue desp_celda_3(540, 215, 80, 20, WHITE, Celda[2].Pres_medida);
        despliegue desp_npasos(540, 240, 80, 20, colores[color_botones], d_p);
        //despliegue desp_celda1_deseada(540, 285, 80, 20, WHITE, d_p);
        //despliegue desp_celda2_deseada(540, 285, 80, 20, WHITE, d_p);
        //despliegue desp_celda3_deseada(540, 285, 80, 20, WHITE, d_p);

float prom_w;

void main()
{

        inicia();

        lee_pasos();

        lee_deforma();

        lee_ks();

        COM1 = new comm(R_COM1, B9600, 1, NO_PARIDAD,8,SIIRQ,NORCR); /* 9600 baud, no paridad, 1
        de stop,8 de dato*/

        COMM = COM1;
//------------------Ciclo ce Control-------------------------------------------
        procesa();

//------------------Liberacion de memoria usada-------------------------------
        delete COM1;

        delete[] Bolsa;
        delete[] Celda;

}; // main

//****************** Fin de la Funcion Principal ***************************


void inicia(void)
{

 registerbgifont(small_font); //Registro del font small_font

 tabla(); //definicion de colores

 Screen(); //Pinta el tablero

//------------------Definicion de Radio_botons--------------------------------
```

```
        radio_all_selected.pintate();
        radio_posicion.pintate();

        radio_pres_indiv.SELECTED = true;
        radio_pres_indiv.pintate();

        radio_norma.pintate();

        radio_presion.SELECTED = true;
        radio_presion.pintate();
        radio_cual.pintate();

        salida.dibuja();

        selctd_up.dibuja();

        selctd_dn.dibuja();

        about.dibuja();

        ok.dibuja();

        pres_normal.dibuja();

        pres_0.dibuja();

        bot_lee_deforma.dibuja();

        Bolsa = new bolsa[Bags_num];  //Crea memoria para bolsas
        //Crea memoria para celdas
        Celda = new bolsa[3];

        dibuja_espejo(origen_x, origen_y);  //Dibuja el espejo

//------------------Colocacion de las Bolsas---------------------------------

        for(int i=0; i<18; i++)
        {

                Bolsa[i].Pres_medida = Bolsa[i].Pres_deseada = 0;

                Bolsa[i].PINTA_PRES = 1;
        }

        Bolsa[0].constructor(BOLSA.1, origen_x+(18.294*4), origen_y-(3.226*4), 25,
                        colores[color_fondo], 3.45, 0.06, 0x0, 17, 53);
        Bolsa[0].motor_pulso = 128;
        Bolsa[0].motor_dir  = 64;

        Bolsa[1].constructor(BOLSA.2, origen_x+(25.671*4.5), origen_y+(9.344*4.5), 25,
                        colores[color_fondo], 3.35, 0.06, 0x4, 36, 56);
        Bolsa[1].motor_pulso = 32;
        Bolsa[1].motor_dir  = 16;

        Bolsa[2].constructor(BOLSA.3, origen_x+(11.940*4), origen_y+(14.230*4), 25,
                        colores[color_fondo], 3.45, 0.06, 0x8, 17, 53);
```

```
Bolsa[2].motor_pulso =  8;
Bolsa[2].motor_dir  =  4;

Bolsa[3].constructor(BOLSA,4, origen_x+(21.037*4.5), origen_y+(25.070*4.5), 25,
                     colores[color_fondo], 3.35, 0.06, 0xc, 29, 51);
Bolsa[3].motor_pulso =  2;
Bolsa[3].motor_dir  =  1;

Bolsa[4].constructor(BOLSA,5, origen_x+ (5.626*4.5), origen_y+(31.908*4.5), 25,
                     colores[color_fondo], 3.45, 0.06, 0x10, 21, 52);
Bolsa[4].motor_pulso = 128;
Bolsa[4].motor_dir  = 64;

Bolsa[5].constructor(BOLSA,6, origen_x- (6.353*4), origen_y+(17.456*4), 25,
                     colores[color_fondo], 3.45, 0.06, 0x14, 32, 52);
Bolsa[5].motor_pulso =  32;
Bolsa[5].motor_dir  =  16;

Bolsa[6].constructor(BOLSA,7, origen_x-(11.193*4.5), origen_y+(30.753*4.5), 25,
                     colores[color_fondo], 3.35, 0.06, 0x1, 10, 50);
Bolsa[6].motor_pulso =  8;
Bolsa[6].motor_dir  =  4;

Bolsa[7].constructor(BOLSA,8, origen_x-(20.927*4.5), origen_y+(17.560*4.5), 25,
                     colores[color_fondo], 3.35, 0.06, 0x5, 14, 52);
Bolsa[7].motor_pulso =  2;
Bolsa[7].motor_dir  =  1;

Bolsa[8].constructor(BOLSA,9, origen_x-(18.294*4), origen_y+ (3.226*4), 25,
                     colores[color_fondo], 3.35, 0.06, 0x9, 21, 54);
Bolsa[8].motor_pulso = 128;
Bolsa[8].motor_dir  = 64;

Bolsa[9].constructor(BOLSA,10, origen_x-(32.230*4.5), origen_y+(5.683*4.5), 25,
                     colores[color_fondo], 3.45, 0.06, 0xd, 23, 54);
Bolsa[9].motor_pulso =  32;
Bolsa[9].motor_dir  =  16;

Bolsa[10].constructor(BOLSA,11, origen_x-(30.540*4.5), origen_y-(11.116*4.5), 25,
                     colores[color_fondo], 3.45, 0.06, 0x11, 22, 51);
Bolsa[10].motor_pulso =  8;
Bolsa[10].motor_dir  =  4;

Bolsa[11].constructor(BOLSA,12, origen_x-(11.940*4), origen_y-(14.230*4), 25,
                     colores[color_fondo], 3.45, 0.06, 0x15, 15, 52);
Bolsa[11].motor_pulso =  2;
Bolsa[11].motor_dir  =  1;

Bolsa[12].constructor(BOLSA,13, origen_x-(21.037*4.5), origen_y-(25.070*4.5), 25,
                     colores[color_fondo], 3.45, 0.06, 0x2, 15, 53);
Bolsa[12].motor_pulso = 128;
Bolsa[12].motor_dir  = 64;

Bolsa[13].constructor(BOLSA,14, origen_x- (4.744*4.5), origen_y-(26.904*4.5), 25,
                     colores[color_fondo], 3.35, 0.06, 0x6, 18, 53);
Bolsa[13].motor_pulso =  32;
Bolsa[13].motor_dir  =  16;
```

```
        Bolsa[14].constructor(BOLSA,15, origen_x+ (6.353*4), origen_y-(17.456*4), 25.
                        colores[color_fondo], 3.45, 0.06, 0xa, 24, 52);
        Bolsa[14].motor_pulso =  8;
        Bolsa[14].motor_dir  =  4;

        Bolsa[15].constructor(BOLSA,16, origen_x+(11.193*4.5), origen_y-(30.753*4.5), 25.
                        colores[color_fondo], 3.45, 0.06, 0xe, 21, 52);
        Bolsa[15].motor_pulso =  2;
        Bolsa[15].motor_dir  =  1;

        Bolsa[16].constructor(BOLSA,17, origen_x+(24.820*4.5), origen_y-(20.826*4.5), 25.
                        colores[color_fondo], 3.45, 0.06, 0x12, 20, 50);
        Bolsa[16].motor_pulso = 128;
        Bolsa[16].motor_dir  = 64;

        Bolsa[17].constructor(BOLSA,18, origen_x+(32.230*4.5), origen_y- (5.683*4.5), 25.
                        colores[color_fondo], 3.45, 0.06, 0x16, 25, 50);
        Bolsa[17].motor_pulso =  32;
        Bolsa[17].motor_dir  =  16;

//----------Colocacion de las Celdas de Carga----------


                for(int j=0; j<3; j++)

                        Celda[j].PINTA_PRES = 0;

        Celda[0].constructor(CELDA,1, origen_x+(35.401*5), origen_y+(12.887*5), 8.
                        BLACK, 0, 0, 0x18, 72, 18);

        Celda[0].dead_w = 5;

        Celda[1].constructor(CELDA,2, origen_x-(28.864*5), origen_y+(24.220*5), 8.
                        BLACK, 0, 0, 0x19, 8, 17);

        Celda[1].dead_w = 5;

        Celda[2].constructor(CELDA,3, origen_x- (6.543*5), origen_y-(37.107*5), 8.
                        BLACK, 0, 0, 0x1a, 84, 15);

        Celda[2].dead_w = 4;

        desp_celda_1.PintaNum(Celda[0].Pres_medida, BLACK);

        desp_celda_2.PintaNum(Celda[1].Pres_medida, BLACK);

        desp_celda_3.PintaNum(Celda[2].Pres_medida, BLACK);

        desp_angulo_z.PintaNum(Inclinometro.cos_z, BLACK);

        desp_npasos.PintaNum(d_p, BLACK);

        if(m_presente())
        {
                zona.xi = 0;        zona.yi = 20;
                zona.xf = getmaxx(); zona.yf = getmaxy();
```

```
                ini_mouse(&zona);

                mouse_xy(587, 457);
        }

        kp_fuerza=KP_FUERZA; ki_fuerza=KI_FUERZA; kd_fuerza=KD_FUERZA;

} // inicia


void procesa()
{
        int sale = 0;

        Boton *bot;

        int X, Y;

        unsigned contador=0;

        char S[256];

 //************************** CICLO PRINCIPAL ************************
        actualiza_deseadas();

        while(!sale)
        {

                if(contador == SALVA)
                {
                        if(radio_presion.SELECTED)

                        salva_pasos();

                        contador = 0;
                }

                contador++;

#ifndef SIMULA

                Inclinometro.lee_angulo();

                for(int j=0; j<3; j++) Celda[j].mide_pres(Inclinometro.cos_z);

#endif

                if(Inclinometro.cos_ant!= Inclinometro.cos_z) {
                        cursor_off();

                        desp_angulo_z.PintaNum(Inclinometro.cos_z, BLACK);

                        cursor_on();

                        Inclinometro.cos_ant= Inclinometro.cos_z;
                }
```

```
                    if(Celda[0].Pres_old != Celda[0].Pres_celda) {

                            cursor_off();

                            desp_celda_1.PintaNum(Celda[0].Pres_celda, BLACK);

                            cursor_on();

                            Celda[0].Pres_old = Celda[0].Pres_celda;
                    }
                    if(Celda[1].Pres_old != Celda[1].Pres_celda) {

                            cursor_off();

                            desp_celda_2.PintaNum(Celda[1].Pres_celda, BLACK);

                            cursor_on();

                            Celda[1].Pres_old = Celda[1].Pres_celda;
                    }
                    if(Celda[2].Pres_old != Celda[2].Pres_celda) {

                            cursor_off();

                            desp_celda_3.PintaNum(Celda[2].Pres_celda, BLACK);

                            cursor_on();

                            Celda[2].Pres_old = Celda[2].Pres_celda;
                    }

//------Checa si se apreto el boton izq del mouse

                    if(kbhit()) {
                            int ch = lee_tecla();
                            if(ch == F32)
                                    sale =1;
                    }
                    else {
                    int pp;
                    pp=COMM->Trae_str(S,':',';');
//                  COMM->Envia_str("Hola.\n");
                    if(pp) {
                            subprocesa(S);
                    }

                    else if(boton_izq())
                    {

//----------Lee las coordenadas del cursor del mouse

                            lee_mouse(&X, &Y);

//----------Checa si presiono algun boton

                            if((bot=cual_boton(X, Y, &salida, &selctd_up, &selctd_dn, &about,
```

```
                                                                    &ok.
&pres_normal. &pres_0, &bot_lee_deforma))

                                        != NULL)
                                {
                                        cursor_off(); bot->oprime(); cursor_on();

                                                while(!izq_suelto()) {}

                                                sale = bot->funcion();

                                        cursor_off(); bot->suelta(); cursor_on();

                                }//endif

                                else

                                {

//----------Checa si se presiono algun radio_boton

                                if((radio_norma.selected(X,Y)) &&
                                        (!radio_all_selected.SELECTED))
                                {
                                        radio_norma.SELECTED = !radio_norma.SELECTED;
                                        norma_selected();

                                }//endif

                                if(radio_presion.selected(X,Y))
                                {
                                        radio_presion.SELECTED = !radio_presion.SELECTED;
                                        presion_selected();
                                }//endif

                                if(radio_cual.selected(X,Y))
                                {
                                        radio_cual.SELECTED = !radio_cual.SELECTED;
                                        cual_selected();
                                }//endif

                                if(radio_posicion.selected(X,Y))
                                {
                                        radio_posicion.SELECTED = !radio_posicion.SELECTED;
                                        posicion_selected();
                                }//endif

                                if((radio_all_selected.selected(X,Y)) &&
                                        (!radio_all_selected.SELECTED))
                                {
                                        all_selected();
                                }//endif


                                if((radio_pres_indiv.selected(X,Y)) &&
                                 (!radio_pres_indiv.SELECTED))
                                {
```

```
                    indiv_selected();
                }//endif

----------Checa si se presiono alguna bolsa

                for(int i=0; i<Bags_num; i++)

                    if(Bolsa[i].selected(X, Y))
                    {
                                if(!radio_all_selected.SELECTED)

                                    Bolsa[i].SELECTED = !Bolsa[i].SELECTED;

                                bolsa_selected(i);

                    }//endif  endfor

----------Checa si se presiono alguna celda

                for(i=0; i<3; i++)

                    if(Celda[i].selected(X, Y))
                    {
                                switch (i)
                                {
                                    case 0: for(int k=1; k<3; k--)
                                            {
                                                if(Celda[k].SELECTED)
                                                {

        Celda[k].SELECTED = !Celda[k].SELECTED;


        Celda[k].dibuja(0);
                                                }
                                            }  break;

                                    case 1: if(Celda[0].SELECTED)
                                                {

        Celda[0].SELECTED = !Celda[0].SELECTED;


        Celda[0].dibuja(0);
                                                }

                                            if(Celda[2].SELECTED)
                                                {

        Celda[2].SELECTED = !Celda[2].SELECTED;


        Celda[2].dibuja(0);
                                                }
                                            break;

                                    case 2: for(k=1; k>-1; k--)
```

```
                                                                {
                                                                        if(Celda[k].SELECTED)
                                                                        {

        Celda[k].SELECTED = !Celda[k].SELECTED:


        Celda[k].dibuja(0):
                                                                        }
                                                                } break:

                                                                default: :
                                                        }//endswitch


                                                        cursor_off():

                                                        if(!radio_all_selected.SELECTED)

                                                        Celda[i].SELECTED = !Celda[i].SELECTED:

                                                        Celda[i].dibuja(0):

                                                        while(boton_izq()) :
                                                        cursor_on():

                                        } endif  endfor

 ----------checa si se presiono algun despliegue

                                        if(desp_npasos.selected(X. Y))
                                        {
                                                float test = d_p:

                                                if(pide_num(&test))
                                                {
                                                        if(test > 1.0 || test < 0.001):

                                                        else
                                                        {
                                                                d_p = test:

                                                                cursor_off(): desp_npasos.PintaNum(d_p. BLACK):
cursor_on():
                                                        }
                                                } endif
                                        }//endif
                                        }//endelse
                                }//endif(mouse_izq)

                                }


 ----------C O N T R O L----------

=ifndef SIMULA
```

```
                    Presiones();
#endif
                    if(radio_posicion.SELECTED)
                        actualiza_fuerza();
                    if(radio_norma.SELECTED)
                        normaliza_deforma();
                    actualiza_deseadas();
                    pinta_presiones(radio_cual.SELECTED,0);
                    if(radio_presion.SELECTED)
#ifdef SIMULA
                        simula();
#else
                        control();
                        delay(1000);
#endif
        }//endwhile
} // procesa

//----------Pinta el Tablero----------

void Screen()
{
        int color_ini = getcolor();
        marco(5, 0, 630, 20, colores[color_fondo]); //Titulo
        prepara_texto(CENTER_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 5,
                        WHITE);
        outtextxy(getmaxx()/2, 7, "CONTROL DEL SISTEMA ACTIVO DEL TELESCOPIO v1.2up");
        marco(440, 25, 195, 450, colores[color_fondo]);         //Panel de control
        outtextxy(537, 25+textheight("M"), "PANEL DE CONTROL");
        setcolor(colores[color_letreros]);
        line(445, 35+textheight("M"), 630, 35-textheight("M"));
```

```
        line(445, 133, 630, 133);

        line(445, 435, 630, 435);

        prepara_texto(LEFT_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 4,
                                        WHITE);

        outtextxy(455, 150, "Coseno Z     ");

        outtextxy(455, 175, "Celda Numero 1");

        outtextxy(455, 200, "Celda Numero 2");

        outtextxy(455, 225, "Celda Numero 3");

        outtextxy(455, 250, "Incr. en Pres.");

        setcolor(color_ini);

};

void dibuja_espejo(int x, int y)
{
        int color_ini = getcolor();

        setfillstyle(SOLID_FILL, LIGHTBLUE);
        setcolor(BLUE);
        fillellipse(x, y, 200, 200);

        setfillstyle(SOLID_FILL, getbkcolor());
        fillellipse(x, y, 30, 30);

        setcolor(color_ini);
};

//------------------Funciones de los Botones---------------------------------

Boton *cual_boton(int x, int y, Boton *bot1, Boton *bot2, Boton *bot3,
                                        Boton *bot4, Boton *bot5, Boton *bot6, Boton *bot7,
                                        Boton *bot8)

{

        if(oprimido(bot1,x,y)) return bot1;

        else if(oprimido(bot2,x,y)) return bot2;

        else if(oprimido(bot3,x,y)) return bot3;

        else if(oprimido(bot4,x,y)) return bot4;
```

```
        else if(oprimido(bot5.x.y)) return bot5;
        else if(oprimido(bot6.x.y)) return bot6;
        else if(oprimido(bot7.x.y)) return bot7;
        else if(oprimido(bot8.x.y)) return bot8;
        else return NULL;
}: cual_boton

int salir(void)
{
        int color_ini=getcolor();

        if(mensaje_graf(""Est s seguro? (S/N)","SsNn")=='S')
        {
                setcolor(color_ini);
                salva_pasos();
                return 1;
        }
        else
        {
                setcolor(color_ini);
                return 0;
        }
}: salir

int presion_up(int num_bag)
{
 if(Bolsa[num_bag].SELECTED)
 {
        Bolsa[num_bag].BorraPresion();
        Bolsa[num_bag].Pres_deforma += d_p;
        if(Bolsa[num_bag].Pres_deforma > Bolsa[num_bag].Pres_max)
                Bolsa[num_bag].Pres_deforma = Bolsa[num_bag].Pres_max;
                Bolsa[num_bag].PintaPresion(radio_cual.SELECTED);
 }
        return 0;
}:
```

```
int presion_dn(int num_bag)
{
 if(Bolsa[num_bag].SELECTED)
 {
        Bolsa[num_bag].BorraPresion( );

        Bolsa[num_bag].Pres_deforma -= d_p;

        if(Bolsa[num_bag].Pres_deforma < -Bolsa[num_bag].Pres_max)

                Bolsa[num_bag].Pres_deforma = -Bolsa[num_bag].Pres_max;

                Bolsa[num_bag].PintaPresion(radio_cual.SELECTED);
 }

        return 0;
};

int presion_dn_all()
{
        for(int i=0; i<18; i++)

                presion_dn(i);

        return 0;
};

int presion_up_all()
{
        for(int i=0; i<18; i++)

                presion_up(i);

        return 0;
};

int OK()
{
        if(!radio_all_selected.SELECTED)
        {
                for(int i=0; i<18; i++)
                {
                        if(!Bolsa[i].SELECTED) ;

                        else

                        {
                                Bolsa[i].SELECTED = !Bolsa[i].SELECTED;

                                Bolsa[i].dibuja(radio_cual.SELECTED);
                        }
                }
        }

        return 0;
}
```

```
int About()
{
        mensaje_graf("Control Activo v1.2up copyright by NOPAL-WARE", NULL);

        return 0;
}

int Normal_pres()
{
        int color_ini=getcolor();


        if(mensaje_graf(""Presiones Normales? (S/N)","SsNn")=='N')
        {
                setcolor(color_ini);

                return 0;
        }

        else

        {

        if(radio_posicion.SELECTED)
        {
                radio_posicion.SELECTED = false;

                radio_posicion.pintate();
        }

        if(!radio_presion.SELECTED)
        {
                radio_presion.SELECTED = true;

                radio_presion.pintate();
        }

        for(int i=0; i<18; i++)
        {
                Bolsa[i].Pres_deforma = 0;

                Bolsa[i].Pres_fuerza = 0;
        }
        } endelse

        return 0;
}

int Cero_pres()
{
        int color_ini=getcolor();


        if(mensaje_graf(""Presiones a cero? (S/N)","SsNn")=='N')
        {
                setcolor(color_ini);
```

```
                return 0;
        }

        else

        {

        if(radio_posicion.SELECTED)
        {
                radio_posicion.SELECTED = false;

                radio_posicion.pintate();
        }

        if(!radio_presion.SELECTED)
        {
                radio_presion.SELECTED = true;

                radio_presion.pintate();
        }

        for(int i=0; i<18; i++)
        {
                Bolsa[i].Pres_deforma = -Bolsa[i].Pres_peso;

                Bolsa[i].Pres_fuerza = 0;
        }
        }//emdelse

        return 0;
}

int Load_deforma()
{

        char archivo[80]="deforma.bag";


        int lon =30, color_ini = getcolor();

        //char S[80];

        //itoa(*t,S,10);

        prepara_texto(LEFT_TEXT,CENTER_TEXT,SMALL_FONT,HORIZ_DIR,5,color_despl);

        int x=(MaxX-(lon+4)*textwidth("M"))/2;

        int y=MaxY/4-2*textheight("M");

        int Ancho = (lon+4)*textwidth("M");

        int Largo = 4*textheight("M");

        void *pt=NULL;
```

```
        guarda_seccion(&pt.x,y,Ancho,Largo);

        Frame Frame_dir(x,y,Ancho,Largo," ");

        Frame_dir.dibuja();

        cursor_off();

        edita_str_graf(x+4*textwidth("M"),y+Largo/2,archivo,
                "abcdefghijklmnopqrstuvwxyz:.1234567890\_",lon-4);

        cursor_on();

        regresa_seccion(&pt.x,y);



        FILE *fp;

        char S[80];

        if((fp = fopen (archivo, "rt")) == NULL)
        {
                mensaje_graf("No se puede abrir archivo...", NULL);

                return 0;
        }

        for(int i=0; i<18; i++) {
                float pp;
                if(fgets(S, 80, fp)) {
                        if(sscanf(S,"%f",&pp)) Bolsa[i].Pres_deforma= pp;
                }
        }
        for(i=0; i<3; i++) {
                float pp;
                if(fgets(S, 80, fp)) {
                        if(sscanf(S,"%f",&pp)) Celda[i].Pres_deforma= pp;
                }
        }

        fclose(fp);

        setcolor(color_ini);

        return 0;

}
 ----------Funciones de bolsa----------

 ------------------------------------------------------------------------

void control()
{

        static char word_30[5] = {0x00, 0x00, 0x00, 0x00, 0x00};
```

```
word_30[0] &=0x55;
word_30[1] &=0x55;
word_30[2] &=0x55;
word_30[3] &=0x55;
word_30[4] &=0x55;

int n_pasos = calcula_dpasos();

for(int j=0; j<n_pasos; j++) {

        for(int i=0; i<18; i++) {

                if( (Bolsa[i].d_pasos > 0) && (Bolsa[i].Pasos < PASOS_MAX) ) {

                        word_30[i/4] |= Bolsa[i].motor_pulso;

                        word_30[i/4] &= ~Bolsa[i].motor_dir;

                        Bolsa[i].Pasos += 1;

                        Bolsa[i].d_pasos --;

                }//endif

                else

                if( (Bolsa[i].d_pasos < 0 ) && (Bolsa[i].Pasos > 0))        {

                        word_30[i/4] |= Bolsa[i].motor_dir;

                        word_30[i/4] |= Bolsa[i].motor_pulso;

                        Bolsa[i].Pasos -= 1;

                        Bolsa[i].d_pasos++;

                }//endif

                else

                if( Bolsa[i].d_pasos == 0 )

                        word_30[i/4] &= ~Bolsa[i].motor_pulso;

        }//endfor

        outportb(wport_300, word_30[0]);
        outportb(wport_301, word_30[1]);
        outportb(wport_302, word_30[2]);
        outportb(wport_303, word_30[3]);
        outportb(wport_304, word_30[4]);

        delay(5);

        outportb(wport_300, word_30[0] & 0x55);
        outportb(wport_301, word_30[1] & 0x55);
        outportb(wport_302, word_30[2] & 0x55);
```

```
                outportb(wport_303, word_30[3] & 0x55);
                outportb(wport_304, word_30[4] & 0x55);

                delay(5);
        }

}

void simula()
{

        int n_pasos = calcula_dpasos();

        for(int j=0; j<n_pasos; j++)
        {

                for(int i=0; i<18; i++)
                {

                        if( Bolsa[i].d_pasos > 0 )
                        {

                                Bolsa[i].Pasos += 1;

                                Bolsa[i].d_pasos --;

                                if(Bolsa[i].Pres_medida < 3.5)

                                Bolsa[i].Pres_medida += 0.0005;

                        }//endif

                        else

                        if( Bolsa[i].d_pasos < 0 )
                        {

                                Bolsa[i].Pasos -= 1;

                                Bolsa[i].d_pasos++;

                                if(Bolsa[i].Pres_medida > 0.06)

                                Bolsa[i].Pres_medida -= 0.0005;

                        }//endif
                 }//endfori
                }//endforj

                delay(10);

                float sum = 0;

                for(int i=0; i<18; i++)

                        sum += Bolsa[i].Pres_medida * 114.7;
```

```
                sum *= 0.454;

                float peso_rest = (2013 - sum) / 3;

                for(j=0; j<3; j++)

                        Celda[j].Pres_celda = peso_rest;

                delay(500);

}

int calcula_dpasos()
{
        float temp;

        int max = 0;

        for(int i=0; i<18; i++)
        {
                if(fabs((temp = Bolsa[i].Pres_deseada - Bolsa[i].Pres_medida))

                        > zona_muerta)

                        Bolsa[i].d_pasos = temp * pasos_psi / factor;

                if(abs(Bolsa[i].d_pasos) > max) max = abs(Bolsa[i].d_pasos);

        };//endfor

        return max;

}

void Presiones()
{
                for(int i=0; i<Bags_num; i++)

                        Bolsa[i].Pres_old = Bolsa[i].Pres_medida;

        //Lee todos los canales de tres en tres

                for(i=0; i<6; i++)
                {
                        outportb(0x305,Bolsa[i].mask);
                        outportb(0x305,(Bolsa[i].mask)|0x20);
                        outportb(0x305,Bolsa[i].mask);
                        delay(5);
                        Bolsa[i].Pres_medida=((float)(inportb(0x300)-Bolsa[i].offset))

        Bolsa[i].ganancia;
                        outportb(0x305,(Bolsa[i].mask)+1);
                        Bolsa[i+6].Pres_medida=((float)(inportb(0x300)-Bolsa[i+6].offset))

        Bolsa[i+6].ganancia;
                        outportb(0x305,(Bolsa[i].mask)+2);
                        Bolsa[i+12].Pres_medida=((float)(inportb(0x300)-Bolsa[i+12].offset))
```

```
Bolsa[i+12].ganancia;
                    }//endfor
}
int pide_int(int *t)
{
        int sale = 0, lon =8;
        char S[80];
        itoa(*t,S,10);
        prepara_texto(LEFT_TEXT,CENTER_TEXT,SMALL_FONT,HORIZ_DIR,5,color_despl);
        int x=(MaxX-(lon+4)*textwidth("M"))/2;
        int y=MaxY/4-2*textheight("M");
        int Ancho = (lon+4)*textwidth("M");
        int Largo = 4*textheight("M");
        void *pt=NULL;
        guarda_seccion(&pt,x,y,Ancho,Largo);
        Frame Frame_dir(x,y,Ancho,Largo," ");
        Frame_dir.dibuja();
        cursor_off();
        if(edita_str_graf(x+4*textwidth("M"),y+Largo/2,S,"1234567890",lon-4)!=0) {
                *t = atoi(S);
                sale = 1;
        }
        cursor_on();
        regresa_seccion(&pt,x,y);
        return sale;
}  pide_int

int pide_num(float *t)
{
```

```
        int sale = 0, lon =12;
        char S[80];
        sprintf(S, "%6.2f", *t);
        prepara_texto(LEFT_TEXT,CENTER_TEXT,SMALL_FONT,HORIZ_DIR,5,color_despl);
        int x=(MaxX-(lon+4)*textwidth("M"))/2;
        int y=MaxY/4-2*textheight("M");
        int Ancho = (lon+4)*textwidth("M");
        int Largo = 4*textheight("M");
        void *pt=NULL;
        guarda_seccion(&pt,x,y,Ancho,Largo);
        Frame Frame_dir(x,y,Ancho,Largo," ");
        Frame_dir.dibuja();
        cursor_off();
        if(edita_str_graf(x+4*textwidth("M"),y+Largo/2,S,"1234567890.",lon-4)!=0) {
                *t = atof(S);
                sale = 1;
        }
        cursor_on();
        regresa_seccion(&pt,x,y);
        return sale;
}  pide_num
void salva_pasos()
{
        FILE *pasos;
        int color_ini=getcolor();
        setcolor(colores[color_letreros]);
        prepara_texto(LEFT_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 4,
                                        WHITE);
        outtextxy(455, 425, "Salvando...");
        if((pasos = fopen ("pasos.bag", "wb")) == NULL)
```

```
        {
                mensaje_graf("No se puede abrir el archivo...", NULL);
                return;
        }
        for(int i=0; i<18; i++)
                fwrite(&Bolsa[i].Pasos, sizeof(int), 1, pasos);
        fclose(pasos);
        setfillstyle(SOLID_FILL, colores[color_fondo]);
        bar(455, 420, 520, 430);
        setcolor(color_ini);
}

void lee_pasos()
{
        FILE *pasos;
        if((pasos = fopen ("pasos.bag", "rb")) == NULL)
        {
                mensaje_graf("No se puede abrir PASOS.BAG...", NULL);
                return;
        }
        for(int i=0; i<18; i++)
                fread(&Bolsa[i].Pasos, sizeof(int), 1, pasos);
        fclose(pasos);
}

void lee_deforma()
{ FILE *fp;
        char S[80];
        if((fp = fopen ("deforma.bag", "rt")) == NULL)
        {
                mensaje_graf("No se puede abrir DEFORMA.BAG...", NULL);
                return;
        }
        for(int i=0; i<18; i++) {
                float pp;
                if(fgets(S, 80, fp)) {
                        if(sscanf(S,"%f",&pp)) Bolsa[i].Pres_deforma= pp;
                }
        }
        for(i=0; i<3; i++) {
                float pp;
                if(fgets(S, 80, fp)) {
```

```
                        if(sscanf(S,"%f",&pp)) Celda[i].Pres_deforma= pp;
                }
        }
        fclose(fp);
}

void lee_ks()
{  FILE *fp;
        char S[80];

        if((fp = fopen ("CONSTS.BAG", "rt")) == NULL)
        {
                mensaje_graf("No se puede abrir CONSTS.BAG...", NULL);

                return;
        }

        float pp;
        if(fgets(S, 80, fp))
                if(sscanf(S,"%f",&pp)) kp_fuerza = pp;
        if(fgets(S, 80, fp))
                if(sscanf(S,"%f",&pp)) ki_fuerza = pp;
        if(fgets(S, 80, fp))
                if(sscanf(S,"%f",&pp)) kd_fuerza = pp;
        fclose(fp);
}

void pinta_presiones(int CUAL, int detodas)
{

        for(int k=0; k<18; k++) {

                if((Bolsa[k].Pres_old != Bolsa[k].Pres_medida) || detodas) {

                        cursor_off(); Bolsa[k].dibuja(CUAL); cursor_on();

                        Bolsa[k].Pres_old = Bolsa[k].Pres_medida;

                        if(fabs(Bolsa[k].Pres_deseada - Bolsa[k].Pres_medida) > 0.2)

                                sonido_corto();

                }
        }
} // pinta_presiones

void actualiza_deseadas()
{
        prom_w = 0;

        for (int i=0;i<18;i++)
        {
                prom_w += Bolsa[i].Pres_peso;

                Bolsa[i].Pres_deseada = Bolsa[i].Pres_peso * Inclinometro.cos_z
                                + Bolsa[i].Pres_deforma - Bolsa[i].Pres_fuerza;
```

```
                if(Bolsa[i].Pres_deseada > Bolsa[i].Pres_max)
                        Bolsa[i].Pres_deseada = Bolsa[i].Pres_max;

                if(Bolsa[i].Pres_deseada < Bolsa[i].Pres_min)
                        Bolsa[i].Pres_deseada = Bolsa[i].Pres_min;
        }

        prom_w /= 18;
}

void actualiza_fuerza()
{
        static int cont =0;

        float prov=0.0, prov1, error = 0.0;

        for (int i=0; i<18; i++) {

                prov=0.0;

                for (int j=0; j<3; j++)

                {

                prov1 = (Celda[j].Pres_celda/0.454)/A_celda;



                prov += Pij[i][j] * (prov1 - (Celda[j].Pres_peso*Inclinometro.cos_z)
                                        -Celda[j].Pres_deforma) * k_celda;
                }

                error = prov * (Bolsa[i].Pres_peso / prom_w);

                Bolsa[i].Pres_fuerza -= error*kp_fuerza/1000.0 +
                        (error - 2* Bolsa[i].error_ant)*ki_fuerza/1000.0 +
                        (error - Bolsa[i].error_ant) * kd_fuerza/1000.0;
                Bolsa[i].error_ant = error;
        }

        cont++;
}  actualiza_fuerza

void normaliza_deforma()
{
        float suma = 0.0;

        for(int i=0; i<18; i++)

                suma += Bolsa[i].Pres_deforma;

        suma /= 18;

        for(i=0; i<18; i++)

                Bolsa[i].Pres_deforma -= suma;
```

```
        suma = 0.0;

        for(i=0; i<3; i++)

                suma += Celda[i].Pres_deforma;

        suma /= 3;

        for(i=0; i<3; i++)

                Celda[i].Pres_deforma -= suma;
}

void norma_selected()
{
        cursor_off();
        radio_norma.pintate();
        while(boton_izq()) ;
        cursor_on();
} // norma_selected

void presion_selected()
{
        cursor_off();
        radio_presion.pintate();
        if(radio_posicion.SELECTED) {
                radio_posicion.SELECTED = false;
                radio_posicion.pintate();
        }//endif
        while(boton_izq()) ;
        cursor_on();
} // presion_selected

void cual_selected()
{
        cursor_off();
        radio_cual.pintate();
        pinta_presiones(radio_cual.SELECTED,1);
        while(boton_izq()) ;
        cursor_on();
} // cual_selected

void posicion_selected()
{
        cursor_off();
        radio_posicion.pintate();
        while(boton_izq()) ;
        cursor_on();
} // posicion_selected

void all_selected()
{
        cursor_off();
        if(radio_norma.SELECTED) {
                radio_norma.SELECTED = false;
                radio_norma.pintate();
        }//endif
```

```
        if(radio_pres_indiv.SELECTED)        {
                radio_pres_indiv.SELECTED = false;
                radio_pres_indiv.pintate();
        }//endif
        radio_all_selected.SELECTED = true;
        radio_all_selected.pintate();
        for(int i=0; i<18; i++) {
                Bolsa[i].SELECTED = true;
                Bolsa[i].dibuja(radio_cual.SELECTED);
        }//endfor
        while(boton_izq()) ;
        cursor_on();
} // all_selected

void indiv_selected()
{
        cursor_off();
        radio_pres_indiv.SELECTED = true;
        radio_pres_indiv.pintate();
        radio_all_selected.SELECTED = false;
        radio_all_selected.pintate();
        for(int i=0; i<Bags_num; i++) {
                Bolsa[i].SELECTED = false;
                Bolsa[i].dibuja(radio_cual.SELECTED);
        }//endfor
        while(boton_izq()) ;
        cursor_on();
} // indiv_selected

void bolsa_selected(int i)
{
        cursor_off();
        Bolsa[i].dibuja(radio_cual.SELECTED);
        while(boton_izq()) ;
        cursor_on();
}  bolsa_selected

int subprocesa(char *S)
{
        if(strstr(S,"nr") || strstr(S,"NR")) norma(S);
        else if(strstr(S,"pr") || strstr(S,"PR")) presion(S);
        else if(strstr(S,"po") || strstr(S,"PO")) posicion(S);
        else if(strstr(S,"cu") || strstr(S,"CU")) cual(S);
        else if(strstr(S,"in") || strstr(S,"IN")) indiv_selected();
        else if(strstr(S,"al") || strstr(S,"AL")) all_selected();
        else if(strstr(S,"bo") || strstr(S,"BO")) selec_bolsa(S);
        else if(strstr(S,"dp") || strstr(S,"DP")) pasos(S);
        else if(strstr(S,"bu") || strstr(S,"BU")) sube_bolsa(S);
        else if(strstr(S,"bd") || strstr(S,"BD")) baja_bolsa(S);
        else if(strstr(S,"pn") || strstr(S,"PN")) presion_normal();
        else if(strstr(S,"pc") || strstr(S,"PC")) presion_cero();
        else if(strstr(S,"lm") || strstr(S,"LM")) lm();
        else if(strstr(S,"ld") || strstr(S,"LD")) ld();
        else if(strstr(S,"lf") || strstr(S,"LF")) lef();
        else if(strstr(S,"lc") || strstr(S,"LC")) lc();
        else if(strstr(S,"la") || strstr(S,"LA")) la();
```

```
        else if(strstr(S,"df") || strstr(S,"DF")) def(S);
        return 1;
} // subprocesa

void norma(char *S)
{
        int pp;
        char ppp[40];

        char *pt=strstr(strupr(S),"NR");
        if(sscanf(pt,"%s %d",ppp,&pp)==2) {
                if(pp==0) radio_norma.SELECTED = 0;
                else radio_norma.SELECTED = 1;
                norma_selected();
        }
} // norma

void presion(char *S)
{
        int pp;
        char ppp[40];

        char *pt=strstr(strupr(S),"PR");
        if(sscanf(pt,"%s %d",ppp,&pp)==2) {
                if(pp==0) radio_presion.SELECTED = 0;
                else radio_presion.SELECTED = 1;
                presion_selected();
        }
} // presion

void posicion(char *S)
{
        char ppp[40];
        int pp;

        char *pt=strstr(strupr(S),"PO");
        if(sscanf(pt,"%s %d",ppp, &pp)==2) {
                if(pp==0) radio_posicion.SELECTED = 0;
                else radio_posicion.SELECTED = 1;
                posicion_selected();
        }
} // posicion

void cual(char *S)
{
        char ppp[40];
        int pp;

        char *pt=strstr(strupr(S),"CU");
        if(sscanf(pt,"%s %d",ppp, &pp)==2) {
                if(pp==0) radio_cual.SELECTED = 0;
                else radio_cual.SELECTED = 1;
                cual_selected();
        }
} // cual

void selec_bolsa(char *S)
```

```
{
        char ppp[40];
        int p1, p2;

        char *pt=strstr(strupr(S),"BO");
        if(sscanf(pt,"%s %d %d",ppp, &p1, &p2)==3) {
                if(p1<0) p1=0;
                if(p1>17) p1 = 17;
                if(p2==0) Bolsa[p1].SELECTED = 0;
                else Bolsa[p1].SELECTED = 1;
                bolsa_selected(p1);
        }
} /* selec_bolsa */

void pasos(char *S)
{
        char ppp[40];
        float pp;

        char *pt=strstr(strupr(S),"DP");
        if(sscanf(pt,"%s %f",ppp, &pp)==2) {
                if(pp > 1.0) pp = 1.0;
                if(pp < 0.001) pp = 0.001;
                d_p = pp;
                cursor_off(); desp_npasos.PintaNum(d_p, BLACK); cursor_on();
        }
}  pasos

void sube_bolsa(char *S)
{
        char ppp[40];
        int pp;

        char *pt=strstr(strupr(S),"BU");
        if(sscanf(pt,"%s %d",ppp, &pp)==2) {
                if(pp<0) pp=0;
                if(pp>17) pp = 17;
                presion_up(pp);
        }
}  sube_bolsa

void baja_bolsa(char *S)
{
        char ppp[40];
        int pp;

        char *pt=strstr(strupr(S),"BD");
        if(sscanf(pt,"%s %d",ppp, &pp)==2) {
                if(pp<0) pp=0;
                if(pp>17) pp = 17;
                presion_dn(pp);
        }
}  baja_bolsa

void presion_normal()
{
        if(radio_posicion.SELECTED) {
```

```
                radio_posicion.SELECTED = false;
                radio_posicion.pintate();
        }
        if(!radio_presion.SELECTED) {
                radio_presion.SELECTED = true;
                radio_presion.pintate();
        }
        for(int i=0; i<18; i++) {
                Bolsa[i].Pres_deforma = 0;
                Bolsa[i].Pres_fuerza = 0;
        }
} // presion_normal

void presion_cero()
{
        if(radio_posicion.SELECTED) {
                radio_posicion.SELECTED = false;
                radio_posicion.pintate();
        }
        if(!radio_presion.SELECTED) {
                radio_presion.SELECTED = true;
                radio_presion.pintate();
        }
        for(int i=0; i<18; i++) {
                Bolsa[i].Pres_deforma = -Bolsa[i].Pres_peso;
                Bolsa[i].Pres_fuerza = 0;
        }
} // presion_cero

void ld()
{
        char S[40];

        COMM->Envia_str(":LD ");
        for(int i=0; i<18; i++) {
                sprintf(S," %1.3f",Bolsa[i].Pres_deseada);
                COMM->Envia_str(S);
        }
        COMM->Envia_str(":");
}  ld

void lm()
{
        char S[40];

        COMM->Envia_str(":LM ");
        for(int i=0; i<18; i++) {
                sprintf(S," %1.3f",Bolsa[i].Pres_medida);
                COMM->Envia_str(S);
        }
        COMM->Envia_str(":");
}  lm


void lef()
{
        char S[40];
```

```
        COMM->Envia_str(":LF ");
        for(int i=0; i<18; i++) {
                sprintf(S," %1.3f",Bolsa[i].Pres_deforma);
                COMM->Envia_str(S);
        }
        COMM->Envia_str(";");
} // lf

void lc()
{
        char S[40];

        COMM->Envia_str(":LC ");
        for(int i=0; i<3; i++) {
                sprintf(S," %1.3f",Celda[i].Pres_celda);
                COMM->Envia_str(S);
        }
        COMM->Envia_str(";");
} // lc

void la()
{
        char S[40];

        sprintf(S,":LA %1.3f;",Inclinometro.cos_z);
        COMM->Envia_str(S);
} // la

void def(char *S)
{
        char S2[40];
        float pp[21];

        char *pt=strstr(strupr(S),"DF");
        if(sscanf(pt,"%s %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f",
                        S2 ,&pp[0],&pp[1] ,&pp[2] ,&pp[3] ,&pp[4] ,&pp[5] ,&pp[6] ,&pp[7] ,
                        &pp[8] ,&pp[9] ,&pp[10] ,&pp[11] ,&pp[12] ,&pp[13] ,&pp[14] ,&pp[15] ,
                        &pp[16] ,&pp[17] ,&pp[18] ,&pp[19] ,&pp[20] )==22) {
                for(int i=0; i<18; i++)
                        Bolsa[i].Pres_deforma = pp[i];
                for(i=0; i<3; i++)
                        Celda[i].Pres_deforma = pp[18+i];
        }
} // def
```

ctrlbags.h

```
#ifndef BAGH
#include "bag.h"
#endif

#ifndef INIGRAPHH
#include "c:\prog\h's\inigraph.h"
#endif

#ifndef MOUSE_H
#include "c:\prog\prj's\guiador\mouse.h"
#endif

#include "grafbags.h"

#ifndef GUIDERH
#include "c:\prog\prj's\guiador\guider.h"
#endif

//#ifndef CELDAH
//#include "celda.h"
//#endif

#include "angulo.h"

#include <math.h>

#define PI                      M_PI
#define origen_x                220
#define origen_y                260
#define Rings_num               2
#define Sectors_num_out         12
#define Sectors_num_in          6
#define Bags_num                18
#define wport_300               0x300
#define wport_301               0x301
#define wport_302               0x302
#define wport_303               0x303
#define wport_304               0x304
#define MaxX                    640
#define MaxY                    480
#define num_pasos               10
#define zona_muerta             0.02
#define pasos_psi               2500
#define factor                  5
#define k_celda                 0.107
#define NOW                     10
#define KP_FUERZA               20
#define KI_FUERZA               4
#define KD_FUERZA               5
#define PASOS_MAX               9400
#define SALVA                   100
```

```
ZONA zona;

int colores[8] = {GRIS_OBSCURO,GRIS_MEDIO,CYAN,BLACK,WHITE,
                  AZUL_OBSCURO,GRIS_MEDIO,GRIS_CLARO};

//------------------Prototipos de Ctrlbags.cpp-------------------------------

Boton *cual_boton(int x, int y,  Boton *bot1, Boton *bot2, Boton *bot3,
                  Boton *bot4, Boton *bot5, Boton *bot6, Boton *bot7, Boton *bot8);

int oprimido(Boton *bot, int x, int y);
void dibuja_espejo(int x, int y);
void Screen();
void control();
void Presiones();
int pide_int(int *t);
int pide_num(float *t);
void salva_pasos();
void lee_pasos();
void lee_deseadas();
void inicia();
void procesa();
int calcula_dpasos();
void inicia(void);
void procesa();
void pinta_presiones(int , int);
void actualiza_deseadas();
void actualiza_fuerza();
void normaliza_deforma();
void simula();
void lee_ks();
void lee_deforma();

int presion_up_all();
int presion_dn_all();

int presion_up(int num_bag);
int presion_dn(int num_bag);

int About();
int OK();

int Normal_pres();
int Cero_pres();
int Load_deforma();
```

angulo.h

```
#include <math.h>

#define offset_inc  8
#define max_read  243

class inclinometro
{
        public:

                float angulo_cenital, cos_z, cos_ant;

                inclinometro();

                ~inclinometro() {};

                void lee_angulo();
};
```

angulo.cpp

```
#include <dos.h>
#include "angulo.h"

inclinometro::inclinometro()
{
        cos_ant = cos_z = 1.0;
}

void inclinometro::lee_angulo()
{
        //Lee el inclinometro

                outportb(0x305,0x1c);
                outportb(0x305,0x1c | 0x20);
                outportb(0x305,0x1c);
                delay(5);
                cos_z =  (float)(inportb(0x300)-offset_inc) / (float)max_read;
                if(cos_z > 1.00) cos_z = 1.00;
                if(cos_z < 0.00) cos_z = 0.00;
                angulo_cenital = acos(cos_z) * (180.0 / M_PI);
}
```

bag.h

```
#include <graphics.h>
#include <stdio.h>
#include <dos.h>

//#include "grafbags.h"

#define BAGH
#define BOLSA 1
#define CELDA 0
#define A_celda 12.56637                    // Area de la celda en pulgadas cuadradas
#define Peso_total 2013.0/.454              // Peso del espejo en libras

class bolsa
{
        public:

                int Num_bag, centro_x_bag, centro_y_bag, radio_bag, color_bag;

                int SELECTED, PINTA_PRES, d_pasos;

                int motor_dir, motor_pulso, Pasos, ganancia, Pasos_max;

                float Pres_medida, Pres_deseada, Pres_max;

                float Pres_peso, Pres_deforma, Pres_fuerza, error_ant;

                float Pres_simula, Pres_old, Pres_min;

                bolsa() {};

                void constructor(int que, int N, int x, int y, int r, int c,
                        float pmax, float pmin, char mask, int offset,
                        int gan);

                void dibuja(int c);

                int selected(int x, int y);

                void PintaNum(int x,int y,double numero, int color);

                void PintaPresion(int);

                void BorraPresion();

                int aumenta_presion();

                int disminuye_presion();

//----------------------Variables para las celdas----------------------------
```

```
        float Pres_celda;
        char mask;
        int dead_w, offset;
        void mide_pres(float cosz);
        ~bolsa() {};
};
```

bag.cpp

```
#ifndef __STDLIB_H
#include <stdlib.h>
#endif

#ifndef BAGH
#include "bag.h"
#endif

#ifndef GUIDERH
#include "c:\prog\prj's\guiador\guider.h"
#endif

#include <math.h>

#define false 0
#define true 1

float P_peso[18]={1.7, 1.921,1.7,1.887,2.006,1.7,1.887,
                  1.921,1.7,1.887,2.006,1.7,1.887,1.921,
                  1.7,1.887,2.006,1.887};


void bolsa::constructor( int que, int N, int x, int y, int r, int c,
                  float pmax, float pmin, char Mask, int Offset,
                  int gan)
{
        SELECTED = false;

        PINTA_PRES = que;

        Num_bag = N;

        centro_x_bag = x;

        centro_y_bag = y;

        radio_bag = r;

        color_bag = c;

        Pres_deseada = Pres_min = pmin;

        Pres_deforma = Pres_fuerza = error_ant = 0;

        Pres_peso = P_peso[N-1];

        Pres_medida = 0; //mide_presion();
```

```
        Pres_simula = Pres_celda = 0;

        Pres_old = 0.06;

        motor_dir = motor_pulso = 0;

        Pres_max = pmax;

        Pasos = d_pasos = 0;

        Pasos_max = 9400;

        mask = Mask;  offset = Offset; ganancia = gan;

        dibuja(0);

        PintaNum(centro_x_bag, centro_y_bag+radio_bag+textheight("M"),
                        Num_bag, WHITE);

        if(que == CELDA)
        Pres_peso = (Peso_total*0.02)/(A_celda*3);

}

int bolsa::selected(int x, int y)
{
 if(  (x > (centro_x_bag - radio_bag))
        & (x < (centro_x_bag + radio_bag))
        & (y > (centro_y_bag - radio_bag))
        & (y < (centro_y_bag + radio_bag)))
        return 1;

 return 0;
}

void bolsa::PintaPresion(int CUAL)
{
        BorraPresion();

        PintaNum(centro_x_bag, centro_y_bag-textheight("M"),
                Pres_medida, WHITE);

        if(CUAL)PintaNum(centro_x_bag, centro_y_bag+5,
                Pres_deseada, YELLOW);
        else PintaNum(centro_x_bag, centro_y_bag+5,
                Pres_deforma, YELLOW);
}

void bolsa::BorraPresion()
{

        int color;

        if(SELECTED) color = CYAN;

        else color = color_bag;
```

```
        setfillstyle(SOLID_FILL,color);

        prepara_texto(CENTER_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 4, color);

        bar(centro_x_bag-(3*textwidth("W")), centro_y_bag-textheight("M")-5,
                centro_x_bag+(3*textwidth("W")), centro_y_bag+textheight("M")+5);

}

void bolsa::PintaNum(int x,int y,double numero, int COLOR)
{
        char *numero_str;

        numero_str = new char[25];

        if((numero < 1e-3) && (numero > -1e-3)) numero = 0;

        gcvt(numero, 3, numero_str);

        prepara_texto(CENTER_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 4, COLOR);

        outtextxy(x, y, numero_str);

        delete[] numero_str;

}


void bolsa::dibuja(int CUAL)
{

        int color_ini = getcolor(), fill_color;

        setcolor(WHITE);

        if((SELECTED)) fill_color = CYAN;

        else fill_color = color_bag;

        if(fabs(Pres_deseada - Pres_medida) > 0.2)
        {
                fill_color = RED;
        }

        setfillstyle(SOLID_FILL, fill_color);

        fillellipse(centro_x_bag, centro_y_bag, radio_bag, radio_bag);

        if(PINTA_PRES) PintaPresion(CUAL);

        setcolor(color_ini);
}

void bolsa::mide_pres(float cosz)
{

                        outportb(0x305,mask);
```

```
outportb(0x305,mask | 0x20);
outportb(0x305,mask);
delay(5);
Pres_celda =  ((float)(inportb(0x300)-

offset-(dead_w*cosz)) * 10) / ganancia;
}
```

grafbags.h

```
#ifndef __STDLIB_H
#include <stdlib.h>
#endif

#ifndef BAGH
#include "bag.h"
#endif

#define  AZUL_OBSCURO          DARKGRAY

#define  AZUL_MEDIO            LIGHTBLUE

#define  AZUL_CLARO            LIGHTGREEN

#define  GRIS_OBSCURO          LIGHTCYAN

#define  GRIS_MEDIO            LIGHTRED

#define  GRIS_CLARO            LIGHTMAGENTA

#define false 0
#define true  1


enum COLORES {color_fondo.color_borde.color_botones.color_sombra.

                color_sombra1.color_texto.color_letreros.color_despl};

enum SENT {none.up.dn.lf.rt};


class Punto
{

protected:

        int x;

        int y;


public:

        Punto(int XIni, int YIni);

        int X();

        int Y();
```

```
};


class Rectangulo : public Punto
{
protected:
        int ancho;
        int largo;
public:
        Rectangulo(int X, int Y, int Ancho, int Largo);
        int W();
        int L();
};


class Boton : public Rectangulo
{
        int suelto;
        void *buffer;
        SENT flecha;
        char *texto;
public:
        int (*funcion)(void);
        Boton(int X, int Y, int Ancho, int Largo, SENT Flecha, char *Texto,
                int (*Funcion)());
        Boton(int X, int Y, int Ancho, int Largo, SENT Flecha, char *Texto,
                int (*Funcion)(bolsa *bag));
        void oprime();
        void suelta();
        void hace_sombra(int opri);
        void dibuja();
```

```
    void borra(int borra);
};

class Cuadro : public Rectangulo {
        int tam1, tam2, tam3, tam4;
        void *lado1, *lado2, *lado3, *lado4;
public:
        Cuadro(int X, int Y, int Ancho, int Largo);
        ~Cuadro() {free(lado1); free(lado2); free(lado3);free(lado4); };
        void dibuja();
        void borra();
        void cambiaXY(int nx, int ny);
};

class Frame : public Rectangulo
{
        char *texto;
public:
        Frame(int X, int Y, int Ancho, int Largo, char *Texto);
        void dibuja();
};
class radio_boton
{
        public:
                int SELECTED,
                        pos_x, pos_y;
                char *texto;
                radio_boton(int x, int y, char *txt);
                void pintate();
                int selected(int x, int y);
```

```
                ~radio_boton() {};
};

class despliegue
{
        public:

                int X, Y, A, L, Color, SELECTED;

                float Numero;

                despliegue(int x, int y, int ancho, int largo, int color, float num);

                void PintaNum(float Num, int color);

                int selected(int x, int y);

                ~despliegue () {};
};

void marco(int x, int y, int ancho, int largo, int color);

void tabla(void);

void inicia_var(int xmax, int ymax);
```

grafbags.cpp

```
#include <graphics.h>
#include <stdlib.h>
#include "grafbags.h"
#include "c:\prog\prj's\guiador\guider.h"

Punto::Punto(int XIni, int YIni)
{
        x = XIni;
        y = YIni;
};
Rectangulo::Rectangulo(int X, int Y, int Ancho, int Largo) : Punto(X,Y)
{
        ancho = Ancho; largo = Largo;
};
Boton::Boton(int X, int Y, int Ancho, int Largo, SENT Flecha, char *Texto,
        int (*Funcion)()) : Rectangulo(X,Y, Ancho, Largo)
{
        flecha = Flecha;
        texto = Texto;
        buffer = NULL;
        suelto = true;
        funcion = Funcion;
};
Frame::Frame(int X, int Y, int Ancho, int Largo, char *Texto) : Rectangulo(X,Y, Ancho, Largo)
{
```

```
        texto = Texto;
};

int Punto::X(void) {
        return x;
};

int Punto::Y(void) {
        return y;
};

int Rectangulo::W(void) {
        return ancho;
};

int Rectangulo::L(void) {
        return largo;
};
void Boton::borra(int borra)
{
        int color_ini = getcolor();
        setfillstyle(SOLID_FILL, borra);
        setcolor(borra);
        bar(x-2, y-2, x+ancho+2, y+largo+2);
        setcolor(color_ini);
}
void Boton::dibuja()
{
        int color = getcolor();
        hace_sombra(0);
```

```
setfillstyle(SOLID_FILL,colores[color_botones]);
bar(x,y,x+ancho,y+largo);
setcolor(colores[color_texto]);
if(texto) {
        struct textsettingstype resp;
        gettextsettings(&resp);
prepara_texto(CENTER_TEXT,CENTER_TEXT,SMALL_FONT,HORIZ_DIR,4,colores[color_texto]);
        outtextxy(x+ancho/2,y+largo/2,texto);
        prepara_texto(resp.horiz,resp.vert,resp.font,resp.direction,resp.charsize,color);
}
switch(flecha) {
case up: {
                        moveto(x+ancho/2,y+3*largo/4);
                        linerel(0,-largo/2);linerel(3,3);
                        moverel(-3,-3);linerel(-3,3);break;
                        }
case dn: {
                        moveto(x+ancho/2,y+largo/4);
                        linerel(0,largo/2);linerel(3,-3);
                        moverel(-3,3);linerel(-3,-3);break;
                        }
case lf: {
                        moveto(x+3*ancho/4,y+largo/2);
                        linerel(-ancho/2,0);linerel(3,-3);
                        moverel(-3,3);linerel(3,3);break;
                        }
case rt: {
                        moveto(x+ancho/4,y+largo/2);
```

```
                                        linerel(ancho/2,0);linerel(-3,-3);
                                        moverel(3,3);linerel(-3,3);break;
                                       }
            }
            setcolor(color);
}
Cuadro::Cuadro(int X, int Y, int Ancho, int Largo):Rectangulo(X,Y,Ancho,Largo)
{
            tam1 = imagesize(x,y,x+ancho,y);
            tam2 = imagesize(x+ancho,y,x+ancho,y+largo);
            tam3 = imagesize(x,y+largo,x+ancho,y+largo);
            tam4 = imagesize(x,y,x,y+largo);
            lado1 = malloc (tam1);
            lado2 = malloc (tam2);
            lado3 = malloc (tam3);
            lado4 = malloc (tam4);
};

void Cuadro::cambiaXY(int dx, int dy)
{
            if(dx>=origenx && dx<=c_x_pix+origenx+15 && dy>=origeny-c_y_pix-55 && dy<=origeny-40) {
                        putimage(x, y, lado1,COPY_PUT);
                        putimage(x+ancho, y, lado2,COPY_PUT);
                        putimage(x, y+largo, lado3,COPY_PUT);
                        putimage(x, y, lado4,COPY_PUT);
                        x= dx;
                        y= dy;
                        dibuja();
            }
```

```
};

void Cuadro::borra()
{
        putimage(x, y, lado1,COPY_PUT);
        putimage(x+ancho, y, lado2,COPY_PUT);
        putimage(x, y+largo, lado3,COPY_PUT);
        putimage(x, y, lado4,COPY_PUT);
};

void Cuadro::dibuja()
{
        int color = getcolor();
        getimage(x,y,x+ancho,y,lado1);
        getimage(x+ancho,y,x+ancho,y+largo,lado2);
        getimage(x,y+largo,x+ancho,y+largo,lado3);
        getimage(x,y,x,y+largo,lado4);
        setcolor(colores[color_sombra1]);
        setlinestyle(DOTTED_LINE,SOLID_FILL,NORM_WIDTH);
        moveto(x,y);
        linerel(ancho,0);
        linerel(0,largo);
        linerel(-ancho,0);
        linerel(0,-largo);
        setcolor(color);
}

void Boton::oprime()
```

```
{
        if(suelto) {
                int tam = imagesize(x,y,x+ancho,y+largo);
                buffer = malloc(tam);
                getimage(x, y, x+ancho, y+largo, buffer);
                hace_sombra(1);
                suelto=false;
                putimage(x, y, buffer, NOT_PUT);
        }
}

void Boton::suelta()
{
        if(!suelto) {
                hace_sombra(0);
                suelto=true;
                putimage(x, y, buffer, COPY_PUT);
                free(buffer);
        }
}

void Boton::hace_sombra(int opri)
{
        int s=1;
        if(opri)
                setcolor(colores[color_sombra]);
        else
                setcolor(colores[color_sombra1]);
        setlinestyle(SOLID_LINE,SOLID_FILL,NORM_WIDTH);
```

```
        moveto(x-s,y+largo+s);
        linerel(0,-largo-(s<<1));
        linerel(ancho+(s<<1),0);
        if(opri)
                setcolor(colores[color_sombra1]);
        else
                setcolor(colores[color_sombra]);
        moveto(x-s,y+largo+s);
        linerel(ancho+(s<<1),0);
        linerel(0,-largo-(s<<1));
        setcolor(colores[color_letreros]);
}

void Frame::dibuja()
{
        int color = getcolor();
        marco(x,y,ancho,largo,colores[color_fondo]);
        setcolor(BLACK);
        line(x+3,y+18,x+ancho-4,y+18);
        setfillstyle(SOLID_FILL,colores[color_botones]);
        bar(x+3,y+3,x+ancho-4,y+17);
        if(texto!=NULL) {
                struct textsettingstype resp;
                gettextsettings(&resp);
                prepara_texto(CENTER_TEXT,CENTER_TEXT,SMALL_FONT,HORIZ_DIR,4,BLACK);
                outtextxy(x+(ancho>>1),y+10,texto);
                prepara_texto(resp.horiz,resp.vert,resp.font,resp.direction,resp.charsize,color);
        }
```

```
        setcolor(color);
} // Frame::dibuja


void marco(int x, int y, int ancho, int largo,int color)
{
        int color_ini = getcolor();
        setlinestyle(SOLID_LINE,SOLID_FILL,NORM_WIDTH);
        setfillstyle(SOLID_FILL,color);
        setcolor(colores[color_borde]);
        rectangle(x,y,x+ancho-1,y+largo-1);
        rectangle(x+1,y+1,x+ancho-2,y+largo-2);
        setcolor(colores[color_sombra]);
        moveto(x+2,y+largo-3);
        linerel(0,-largo+5);
        linerel(ancho-5,0);
        setcolor(GRIS_CLARO);
        moveto(x+2,y+largo-3);
        linerel(ancho-5,0);
        linerel(0,-largo+5);
        bar(x+3,y+3,x+ancho-4,y+largo-4);
        setcolor(color_ini);
} // marco
void despliegue::despliegue(int x, int y, int ancho, int largo, int color, float num)
{
        X = x; Y = y; A = ancho; L = largo; Color = color; Numero = num;
        marco(X, Y, A, L, Color);
        PintaNum(Numero, BLACK);
}
void despliegue::PintaNum(float numero, int COLOR)
{
        char *numero_str;
```

```
            numero_str = new char[25];
            marco(X, Y, A, L, Color);
            gcvt(numero, 4, numero_str);
            prepara_texto(RIGHT_TEXT, CENTER_TEXT, SMALL_FONT, HORIZ_DIR, 4, COLOR);
            outtextxy(X+A-5, Y+L/2, numero_str);
            delete[ ] numero_str;
}
int despliegue::selected(int x, int y)
{
 if(  (x > (X))
            & (x < (X + A))
            & (y > (Y))
            & (y < (Y + L)))
            return 1;
 return 0;
}
void radio_boton::radio_boton(int x, int y, char *txt)
{
            pos_x=x;
            pos_y=y;
            texto=txt;
            SELECTED=false;
            //pintate();
}
void radio_boton::pintate()
{
            int color_ini=getcolor(), fill_Color;
            setfillstyle(SOLID_FILL,WHITE);
            setcolor(BLACK);
            fillellipse(pos_x,pos_y,5,5);
            if(SELECTED) fill_Color=GREEN;
            else fill_Color=RED;
            setfillstyle(SOLID_FILL,fill_Color);
```

```
            fillellipse(pos_x,pos_y,3,3);
            prepara_texto(LEFT_TEXT.CENTER_TEXT,SMALL_FONT,HORIZ_DIR,4,
                                        WHITE);
            outtextxy(pos_x+15,pos_y,texto);
            setcolor(color_ini);
}
int radio_boton::selected(int x, int y)
{
 if( (x > (pos_x - 5))
            & (x < (pos_x + 5))
            & (y > (pos_y - 5))
            & (y < (pos_y + 5)))
           return 1;
 return 0;
}

/* Carga la tabla de color en los DACS */
void tabla(void)
{
            setrgbpalette(56, 0, 20, 43);// azul obscuro
            setrgbpalette(57, 0, 0, 63); // azul medio
            setrgbpalette(58, 0, 43, 63);// azul claro
            setrgbpalette(59, 25, 25, 25); // gris oscuro
            setrgbpalette(60, 30, 30, 30); // gris medio
            setrgbpalette(61, 40, 40, 40);// gris claro
}
```

globales.h

```
// Variales globales del programa de control de la celda del primario

float d_p = 0.02;

int kp_fuerza, ki_fuerza, kd_fuerza;

float Pij[18][3] = {{0.617968, 0.048698, 0.333334},
                    {0.816693, 0.091654, 0.091653},
                    {0.617969, 0.333333, 0.048697},
                    {0.834803, 0.333333, -0.168136},
                    {0.620850, 0.620849, -0.241699},
                    {0.333334, 0.617967, 0.0486988},
                    {0.333335, 0.834801, -0.168136},
                    {0.091654, 0.816692, 0.091654},
                    {0.048698, 0.617969, 0.333333},
                    {-0.168135,0.834801, 0.333333},
                    {-0.241698,0.620848, 0.620850},
                    {0.048697, 0.333333, 0.617969},
                    {-0.168136,0.333334, 0.834803},
                    {0.091654, 0.091656, 0.816691},
                    {0.333333, 0.048699, 0.617968},
                    {0.333332, -0.168135,0.834803},
                    {0.620848, -0.241697,0.620849},
                    {0.834801, -0.168135,0.333334}};
```