Instituto Nacional de Astrofísica, Optica y Electrónica.

REPORTE TECNICO

COORDINACION DE ASTROFISICA

MANUAL DESCRIPTIVO DEL SISTEMA DE CONTROL DEL RT5

Tlatelpa-Osorio Y.E., Orozco-Serna B., Palacios-Fonseca J.S., Mendoza-Torres J.E.

Tabla de Contenido

Introducción

Este manual descriptivo del Radio Telescopio de 5 metros (RT5) ubicado en el Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), tiene la intención brindarle al usuario, tal como su nombre lo indica, una descripción del hardware y software que componen el sistema de control de movimiento de este instrumento.

El objetivo de este manual no es la enseñanza de radioastronomía ni de los instrumentos que se utilizan para su estudio, sin embargo en la introducción se ha incluido un marco teórico, con los elementos básicos necesarios para saber que es un radio telescopio, ya que este manual puede ser leído también por usuarios de áreas diferentes a la radioastronomía.

Debido a que las tablas de conexiones del sistema de control son extensas, se incluyen pero en forma de apéndices al final del documento. Un apéndice con el índice de figuras, también es agregado.

Radio telescopios

Un Radiotelescopio es un instrumento que permite a los radio astrónomos estudiar fuentes de radio extraterrestre, es decir, objetos celestes que producen ondas de radio en varios procesos. Las ondas de radio se pueden clasificar según el tipo de proceso que las genera de la siguiente manera:

- Radiación térmica de objetos sólidos como planetas
- Radiación térmica de objetos sólidos como planetas.
- Bremmstrahlung (del alemán, radiación de frenado) radiación de gas caliente en el medio interestelar.
- Radiación Synchrotron, de electrones moviéndose a velocidades cercanas a la velocidad de la luz en campos magnéticos débiles.
- Radiación de línea espectral, de transiciones en átomos o moléculas en el medio interestelar o el gas de las estrellas.
- Radiación pulsada, por el efecto de la rápida rotación de estrellas de neutrones rodeadas por un campo magnético y electrones energéticos.

Los radiotelescopios son usados para medir la temperatura de los planetas e incluso medir la velocidad de rotación de estos. También se han utilizado para identificar el tipo de moléculas presentes en cuerpos celestes o el medio interestelar (se han identificado 150). También han sido utilizados para ver objetos muy lejanos ya que las ondas de radio se ven poco frenadas por la atmosfera terrestre. Inclusive han sido la clave para identificar la “Radiación de Fondo” que se considera resultado de la Gran Explosión (en inglés Big Bang) que emite radio ondas en todas direcciones.

Las primeras detecciones de ondas de radio extraterrestres provenientes de la Vía Láctea fueron realizadas por casualidad por el ingeniero Karl G. Jansky cuando estaba investigando en los años 30

acerca del ruido en las transmisiones de voz transatlánticas que hacía para los laboratorios de Bell Telephone, pero no concluyó su investigación. Fué Reber, ingeniero de radio en el Instituto Tecnológico de Illinois quien decidió construir su propio radiotelescopio en el patio de su casa. Su diseño fue considerablemente más avanzado que el de Jansky. Consistía de un espejo de metal parabólico de 9 m de diámetro, enfocado en un radio receptor a 8 m sobre el espejo. El dispositivo, completado en 1937, estaba montado en un soporte inclinable que permitía apuntarlo en varias direcciones, aunque no girarlo.

Reber no obtuvo señales extraterrestre con su primer receptor, que operaba a 3300 MHz, ni con el segundo, operado a 900 MHz. Su tercer intento, a 160 MHz (1938), fue exitoso, confirmando el hallazgo de Jansky.

En 1944 publica el primer mapa de radio de la Vía Láctea. Su actividad cartográfica durante la postguerra fue el disparador de la explosión en el interés por la radioastronomía, época en la que los instrumentos de radio utilizados en la guerra se utilizaron para mirar al cielo.

Partes de un Radio telescopio

Figura 1 Partes de un radio telescopio.

Un radio telescopio funciona captando las ondas de radio en el espejo primario, las cuales son reflejadas al espejo secundario (foco primario) que tiene una forma convexa y que a su vez las refleja a un orificio en el espejo primario (foco secundario) donde se encuentra un radio receptor, luego la señal es amplificada, después almacenada y adecuada para su análisis. En La Figura 3 podemos observar el proceso descrito.

La calidad de las observaciones de un radio telescopio es principalmente determinada por la resolución angular del mismo. La resolución angular depende de la longitud de onda de la observación dividida por el tamaño de la antena. Incluso cuando operan a una longitud de onda más corta tienen una resolución de unos cuantos segundos de arco, mucho más pobre que cualquier telescopio óptico. Como los radio telescopios operan a longitudes de onda más grandes que los ópticos necesita ser más grandes para alcanzar la misa resolución angular. Una ventaja es que las distorsiones introducidas por la atmosfera son menos apreciables que para las longitudes de onda ópticas. Para alcanzar grandes resoluciones se emplean los principios de interferometría, para sintetizar una apertura con varias antenas.

Montura Ecuatorial

Otra parte importante para una buena calidad en las observaciones de un radio telescopio es la precisión en el apuntado y seguimiento de los objetos celestes, la mayoría de los radio telescopios están sobre una montura ecuatorial.

Esta montura es en la que se encuentra el RT5 que es el radiotelescopio de nuestro interés.

La montura ecuatorial utiliza como plano fundamental el ecuador celeste (proyección del ecuador terrestre). Este diseño usa las coordenadas ecuatoriales, ascensión recta (AR) y declinación (DEC), que son proyecciones de las coordenadas terrestres longitud y latitud, respectivamente, sobre la esfera celeste.

Su mayor ventaja es la posibilidad de seguir a los objetos celestes con solo mover un eje. También puede ser motorizado, para que el seguimiento sea automático y los objetos se mantengan centrados en el campo visual.

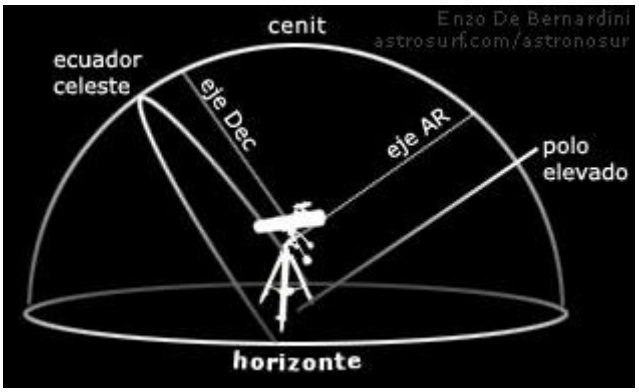

Figura 2 Montura ecuatorial.

El Radio Telescopio de 5 metros (RT5)

El RT5, como mencionado anteriormente es un radio telescopio cuyo plato (antena parabólica) tiene un diámetro de 5 metros y que esta sobre una montura de tipo ecuatorial, por lo tanto tiene dos ejes, el de ascensión Recta (AR) y Declinación (DEC) con los cuales se realiza el apuntado a objetos celestes, así como el seguimiento de los mismo, el cual resulta sencillo por el tipo de montura .

En la Figura 3 se muestra una imagen del RT5 instalado en el INAOE.

Figura 3 Radio Telescopio de 5 metros.

La localización de las partes de un radio telescopio en el RT5 se muestra en la Figura 4

Figura 4 Ubicación de las partes sobre el RT5

En la Figura 5 y la Figura 6 se identifican los ejes de Ascensión Recta y Declinación en le RT5.

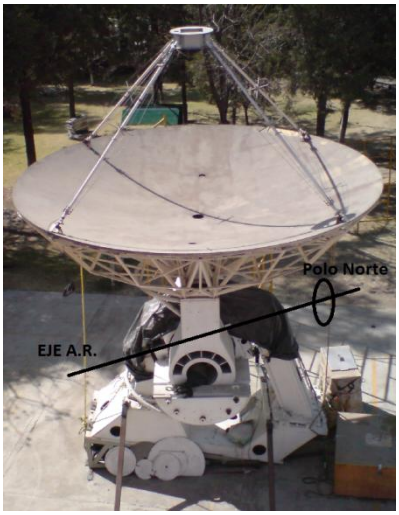

Figura 5 Eje de Ascensión Recta.

Figura 6 Eje de Declinación.

Estos dos ejes son los que permiten el apuntado de la antena a objetos celestes que se quieran estudiar, es por eso que se ha desarrollado todo un sistema para controlar sus movimientos, dados por dos motores de corriente directa montados en cada uno de los ejes al igual que un encoder en cada eje que permite conocer la posición de cada uno y controlarla.

Descripción del sistema de control de movimiento del RT5 (hardware)

El diagrama a bloques de la Figura 7 establece la relación de alto nivel entre la interfaz de usuario y el sistema de control del telescopio.

Figura 7 Diagrama a bloques del sistema.

Interfaz de usuario

La interfaz de Usuario es una computadora personal con sistema operativo Linux en la cual se ejecuta el software diseñado con el fin específico de manipular los movimientos de la antena. La interfaz de usuario y el sistema de control están conectados mediante una red Ethernet.

Gabinete de control

El diagrama de la Figura 8 muestra el diagrama a bloques del sistema de control del RT5.

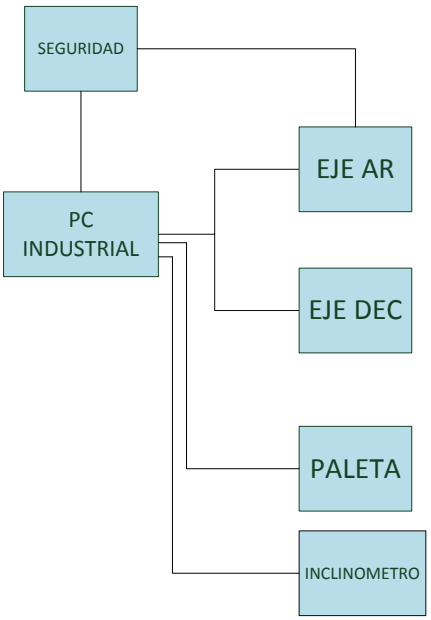

Figura 8 Diagrama a bloques del Sistema de Control.

Como mencionado anteriormente, el RT5 tiene movimiento en cada uno de sus ejes. La Figura 9 muestra el diagrama a bloques del sistema de control para uno solo de los ejes.

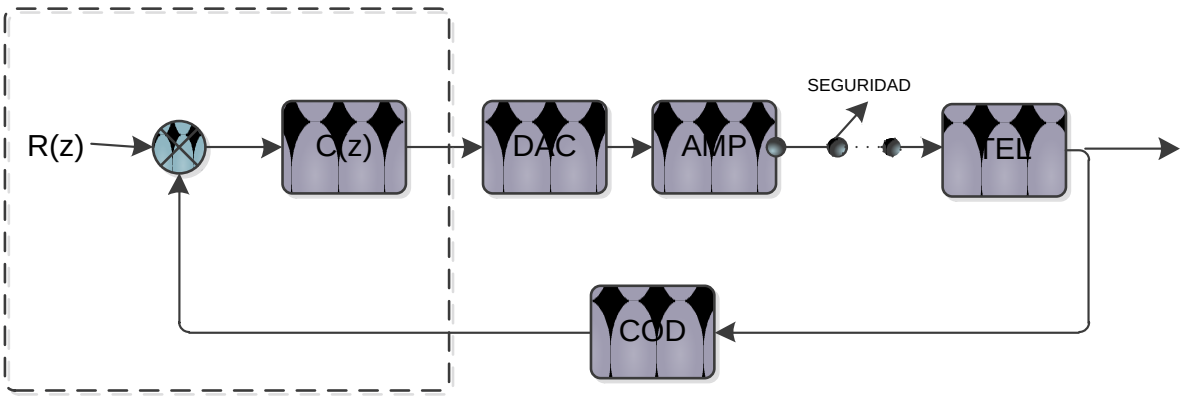

Figura 9 Diagrama a bloques para el sistema de control de uno de los ejes.

Donde:

- TEL: Eje del telescopio (AR o DEC)
- COD: Codificador o encoder.
- C(z): Ley de control del sistema.
- DAC: Convertidor analógico digital
- SEG: Sistema de seguridad.

La parte encerrada en una línea punteada se ejecuta en la computadora industrial de control.

El sistema de control está compuesto por una computadora industrial de una sola tarjeta PCA-6772 (micro controlador) y su tarjeta de adquisición MFIO-3A (controlador).Esta computadora industrial de control se encarga del control de bajo nivel del telescopio, ejecutando los algoritmos de control digital (PID discretos).

El micro controlador de la tarjeta PCA-6772, actúa como servidor y atiende a la computadora personal y al controlador, la tarjeta de adquisición de datos MFIO-3A, que a su vez sirve de interfaz entre el micro controlador y los dispositivos de entrada y salida.

Entre el micro controlador y las entradas y salidas se encuentran los opto acopladores que aíslan el circuito electrónico del de potencia, que es el circuito encargado de alimentar a los motores que brindarán el movimiento en los ejes.

En cada uno de los ejes están montados unos codificadores o encoders ópticos de tipo incremental con salida de pulsos en cuadratura, que retroalimentan al control en posición, para lograr la precisión requerida.

Los sensores de seguridad o de límite sirven para activar los frenos, que además también pueden ser activados por el control.

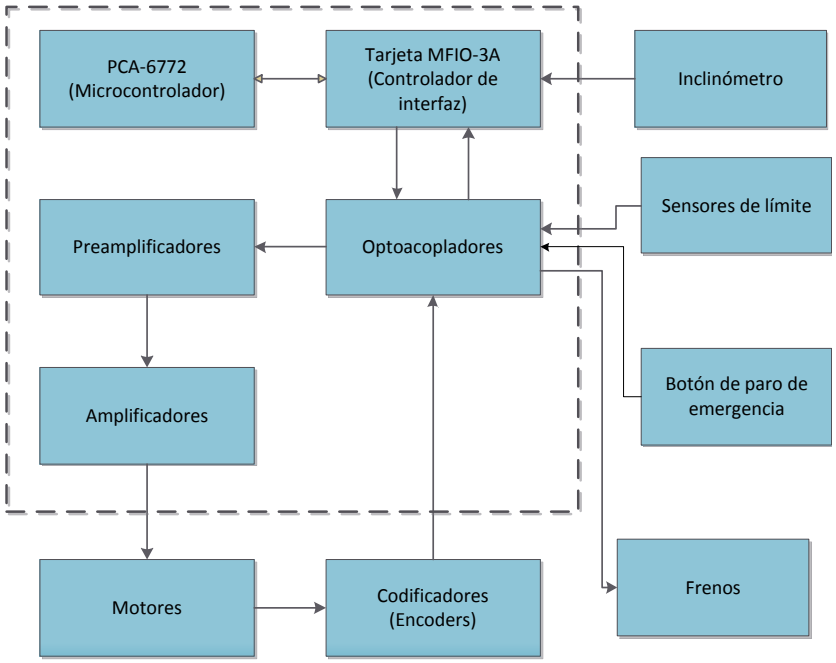

Figura 10 Diagrama a bloques de los componentes del sistema de control

Todos los componentes del sistema de control que muestra el diagrama a bloques de la Figura 10 están interconectados conforme al diagrama de la Figura 11. Y los que están dentro de las líneas punteadas están contenidos dentro de lo que llamamos “gabinete de control” que es la caja negra de la Figura 7 que ilustra al sistema de control, lo demás está montado en la montura del radio telescopio.

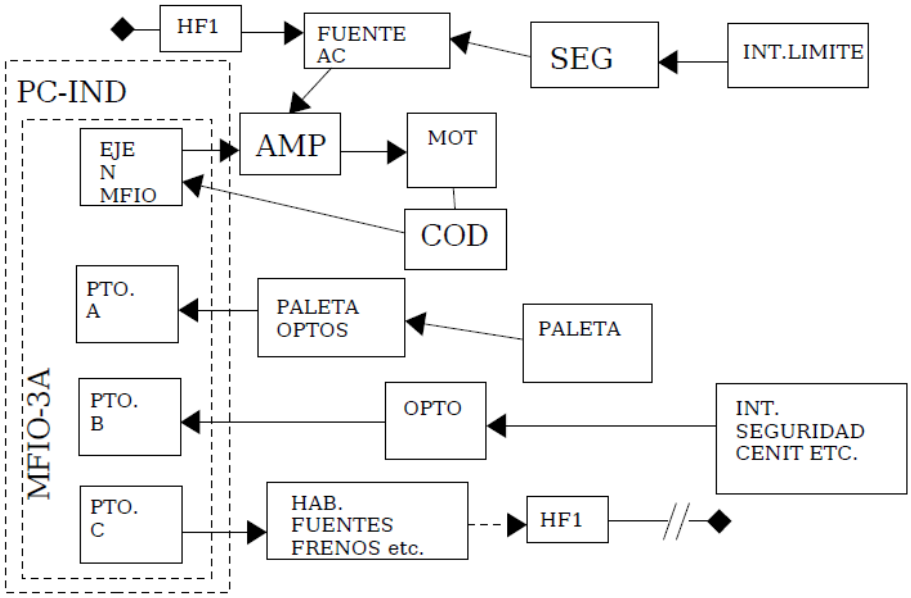

Figura 11 Diagrama a bloques de conexiones entre componentes.

El diagrama unifilar eléctrico de la Figura 12 muestra las conexiones eléctricas del gabinete de control.

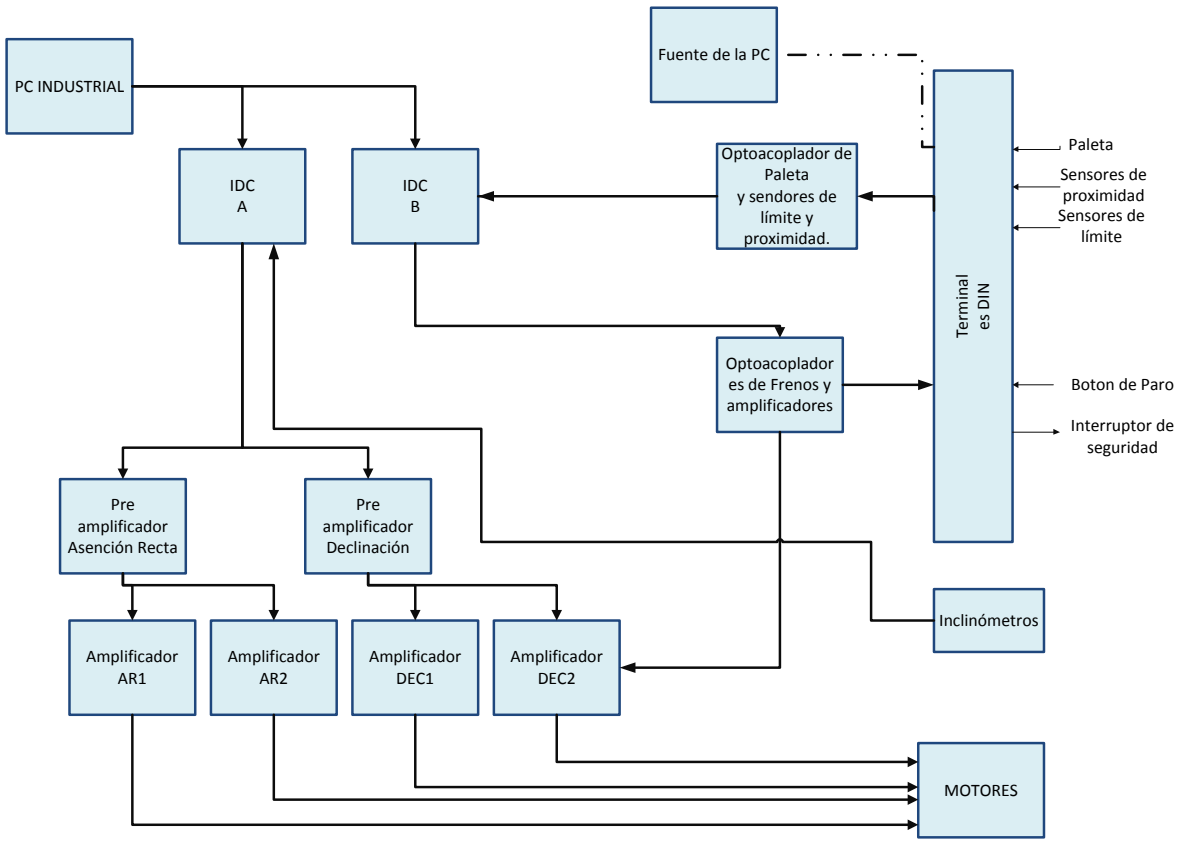

Figura 12 Diagrama unifilar eléctrico.

Los componentes del sistema de control se encuentran distribuidos dentro del gabinete de control como se muestra en la Figura 13.

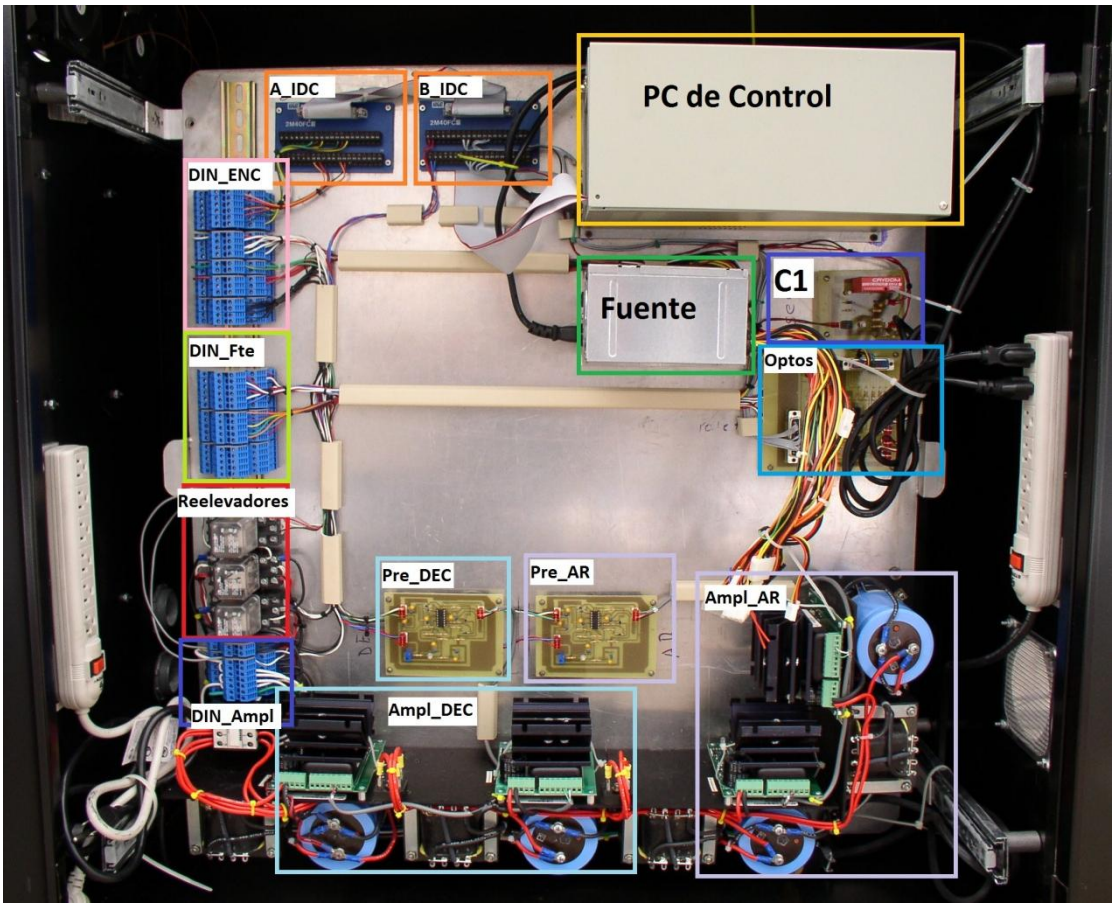

Figura 13 Distribución del gabinete de control.

En donde:

- PC de control: PCA-6772
- B_IDC, A_IDC, terminales IDC de la tarjeta MFIO-3ª.
- Fuente: Fuente de alimentación de PC para opto acopladores, Pre Amplificadores etc. (+/-12v, +5v).
- C1: Tarjeta con opto acoplador para apagar y/o prender frenos o amplificadores.
- Optos : Tarjeta de opto acopladores de paleta y sensores de proximidad.
- Relevadores
- Pre_DEC, Pre_AR: Preamplificadores de Declinación y Ascensión Recta respectivamente.
- Ampl_DEC, Ampl_AR: Amplificadores de Declinación y Ascensión Recta respectivamente.
- DIN_xxxx. Terminales DIN para hacer las interconexiones.

A continuación se da una descripción de los componentes que se encuentran dentro del gabinete de control.

PC de control: PCA-6772

Como ya se mencionó con anterioridad, esta computadora industrial se encarga del control de bajo nivel del telescopio. Ejecuta los algoritmos de control digital (PID discretos).

Se trata de una computadora industrial de una sola tarjeta.
Modelo: PCA 6772
Fabricante: Advantech
Insertable en ducto ISA.
Memoria: 64 Megabytes
Disco de memoria Flash: 64 megabytes (Compactflash)
Tarjeta de adquisición: MFIO-3A de Precision Micro Dynamics PMDi.

Figura 14 PCA 6772

Sus características principales son :

- ISA Slot SBC,
- VGA/LCD/CFC/1
- Ethernet A
- VIA Eden 400/667 CPU de bajo consumo de potencia
- Conexión Ethernet 10/100
- Soporta 100/133MHz FSB
- 4 x AGP
- Operación sin abanicos en el CPU

Esta computadora industrial utiliza los siguientes recursos de programación:

- Linux con extensiones de tiempo real.
- Kernel: 2.4.20
- Módulos : control3.o
- Programas de aplicación: cons84serv-s

Imagen de Linux que se carga en RAM a la hora de iniciar la máquina.

Tarjeta MFIO-3A.

Esta tarjeta insertable en un puerto Isa (plug and play) cuenta con:

- 3 entradas de codificador en cuadratura de alta velocidad
- Salidas digital analógicas de 16 bits (+/- 10 volts)
- Contadores de 24 bits.

Sockets abiertos:

Número de socket	Función
9023	Como un telnet
4955	Servidor de conexiones del programa de control de telescopio

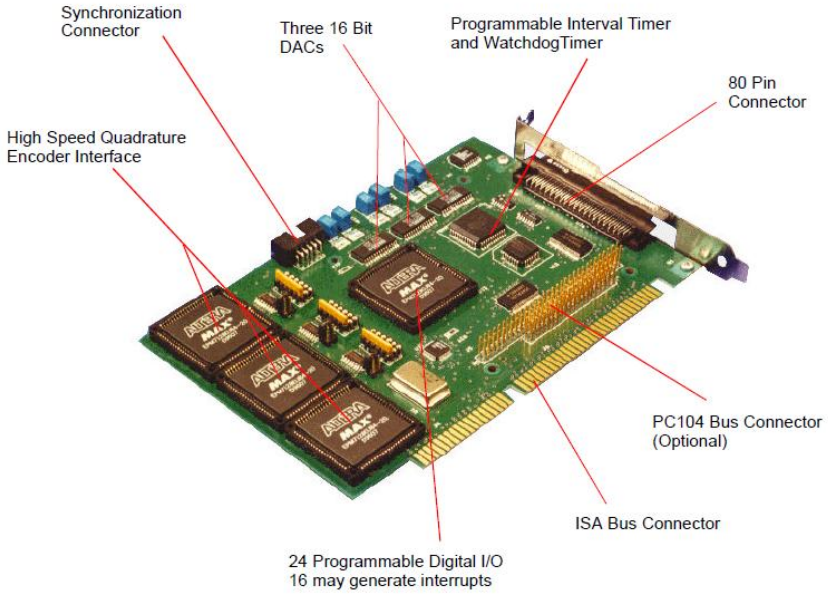

Figura 15 Tarjeta MFIO -3A

Terminales B_IDC y A_IDC

Las terminales o tiras B_IDC y A_IDC están unidas a un conector en la tarjeta MFIO-3A y se usan para interconectar las señales de codificadores y amplificadores a la tarjeta y realizar el control digital.

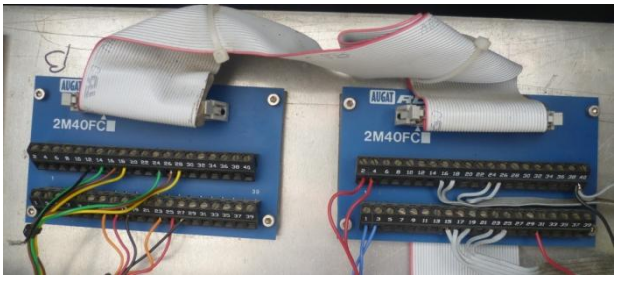

Figura 16 Terminales B_IDC y A_IDC.

EN el anexo A se encuentran las tablas de conexiones de la tira A_IDC y B_IDC

Tarjeta de opto acoplador para activar frenos y/o amplificadores (C1)

Esta tarjeta de opto acoplador, se usa para activar el encendido de la parte de potencia del sistema (amplificadores ó frenos). La tarjeta se controla por medio de un bit de la tarjeta MFIO-3A. Cuando la señal es cero, o está en alta impedancia, son los frenos los que están activados, cuando la señal es positiva, entonces la alimentación que se activa es la de los amplificadores y los frenos se desactivan.

La salida va a relevadores que son los que finalmente permiten activar la alimentación de los frenos o de los amplificadores.

En la Figura 17 se muestra la distribución de los conectores de entradas y salidas de la tarjeta y en la Figura 18 se ve una imagen real de la tarjeta.

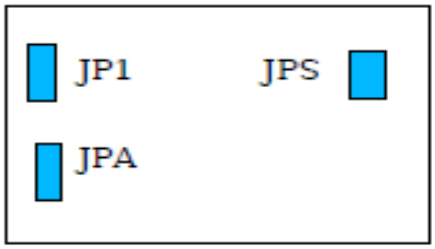

Figura 17 Distribución de la tarjeta C1.

Figura 18 Imagen real de la tarjeta C1.

Las señales que entran y/o salen de los conectores son:

- JP1- Señal de control, es decir , activa amplificadores (en nivel alto) y/o frenos (en nivel bajo)
- JPA- Es la alimentación de la tarjeta.
- JPS- Salida a relevador.

Para saber en donde se conectan estas entradas y salidas ver el anexo B que contiene las tablas de conexiones de esta tarjeta.

Relevadores

Los Relevadores son activados por la señal proveniente de La tarjeta de opto acoplador de activación de frenos y/o amplificadores. Son estos relevadores quienes finalmente permiten la activación de frenos o amplificadores, permitiendo el paso de corriente para su alimentación.

En la Figura 19 se muestra el diagrama de conexiones de los relevadores

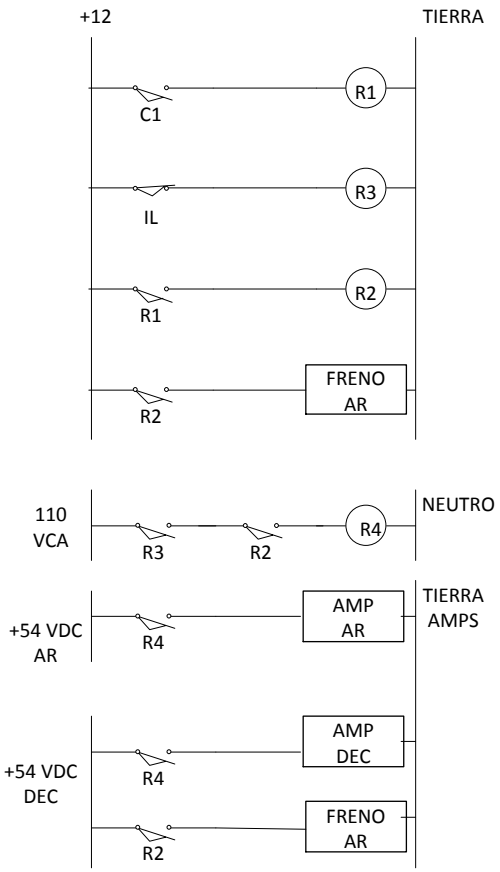

Figura 19 Diagrama de conexiones de relevadores.

En donde:

- C1 – Contacto normalmente abierto del relevador de estado sólido que controla la tarjeta MFIO-3A para habilitar los frenos y/o amplificadores (entre los dos está la tarjeta de opto acopladores). Cuando una señal positiva es enviada en el contactor C1, este se cierra, activando R1 que a su vez activa a R2 y esto manda alimentación a los frenos que se desactivan con una señal positiva.

- IL – Interruptores para límites de seguridad, deben ser interruptores normalmente cerrados, alambrados en serie. Mientras este contactor no se abra y el contactor C1 esté cerrado, se activará R4, que a su vez permitirá la alimentación de los amplificadores y por lo tanto permitirá su activación.

- R1,R2,R3 – Relevadores KUP de 12 volts, R4 es un contactor de 4 contactos con bobina de 110 VCA.

Quedan disponibles dos contactos normalmente abiertos (NA) del relevador R1 y otros dos contactos NA del relevador R4. Ambos relevadores se pueden activar o desactivar por programación por medio de un puerto de la MFIO-3A.
Se conectaron las fuentes de los amplificadores de AR y DEC a través de un diodo para que compartieran la carga como se muestra en la Figura 20.

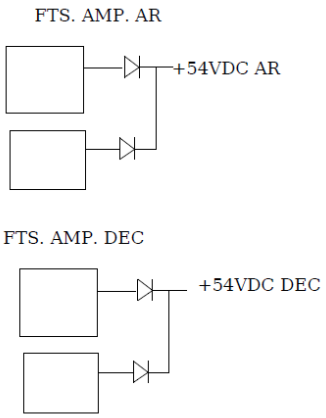

Figura 20 conexión de las fuentes de amplificadores.

Freno de emergencia

El freno de emergencia funciona como los interruptores de límite de la figura 19, cortando la alimentación a los amplificadores al abrir el contacto normalmente cerrado "LI" y activa los frenos abriendo también el contacto normalmente abierto "c1" y como los frenos son activos en nivel bajo o alta impedancia, estos se activan. Tiene esta función de los interruptores de límite, ya que en su actual instalación en el INAOE no están conectados los interruptores de límite.

La figura 21 muestra una imagen real del botón de paro que activa el freno de emergencia.

Figura 21 botón de paro de emergencia.

Tarjeta de opto acopladores de paleta y sensores de proximidad (Optos)

Se usa una tarjeta con opto-acopladores para las señales de paleta manual del telescopio y de los sensores de proximidad que detectan la posible llegada a un límite físico. Las salidas de esta tarjeta se conectan a puertos de entrada de la tarjeta MFIO-3A para que el programa de control ejecute la acción correspondiente.
Esta tarjeta tiene 7 conectores dispuestos como se muestra la Figura23 y en la Figura 22 se ve una imagen de la tarjeta real.

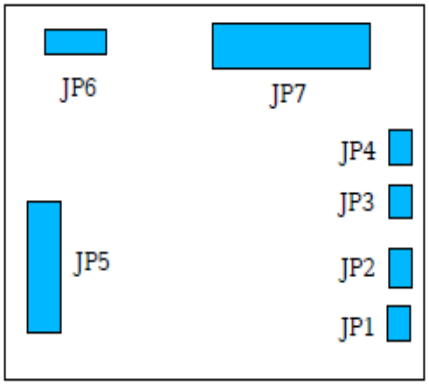

Figura 23 Distribución de la tarjeta de Optoacopladores de paleta manual.

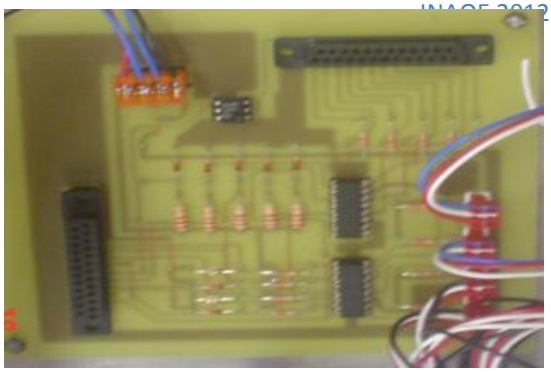

Figura 22 Imagen de la tarjeta real de Optos.

Las señales de salida y/o entrada de los diferentes conectores con las siguientes:

- Sensores de Proximidad
  - JP2
  - JP3
  - JP4
- Salidas optoacopladas
  - JP5
- Alimentación
  - JP6
- Entradas de Paleta
  - JP7

Para saber en donde se conectan estas entradas y salidas ver el anexo C que contiene las tablas de conexiones de esta tarjeta.

Pre amplificadores de AR y DEC

Se usan dos tarjetas para acoplar la señal de control de la tarjeta MFIO- 3A a los amplificadores de potencia, una para Asención Recta y otra para Declinación.
La finalidad de las tarjetas es mandar la señal de entrada y de precarga al amplificador correspondiente. Cuando a un amplificador llega la señal de control al opuesto le llega la señal de precarga y viceversa.

Los conectores de esta tarjeta están dispuestos como se muestra en el siguiente diagrama.

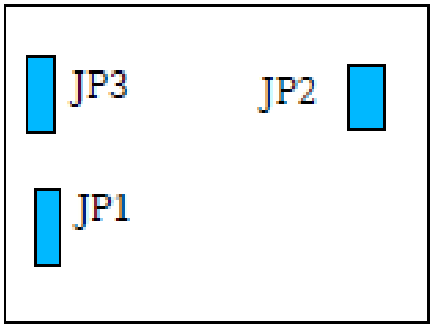

Figura 24 distribución de la tarjeta de preamplificación.

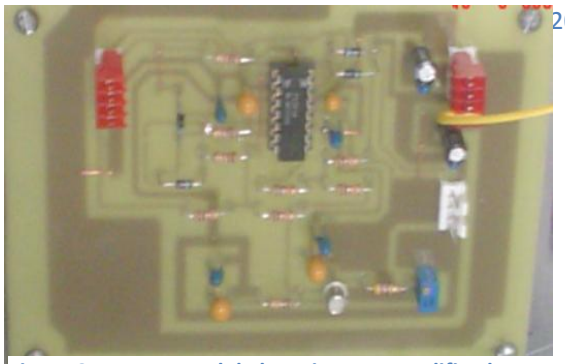

Figura 25 Imagen real de la tarjeta preamplificadora.

Las señales de salida y/o entrada de los diferentes conectores con las siguientes:

- JP1- Es la entrada de control.
- JP2- Salida de control a amplificadores.
- JP3- Alimentación de la tarjeta.

Para saber en donde se conectan estas entradas y salidas ver el anexo D que contiene las tablas de conexiones de esta tarjeta.

Amplificadores de AR y DEC

Los amplificadores son del tipo D modulado en ancho de pulso “PWM”.
Modelo: 10A8J-DS1
Serie 700 de Dynetic Systems Company.

Estos amplificadores son los que alimentan a los actuadores finales que son los motores que mueven a cada eje (AR y DEC) por lo tanto se utilizan 4, dos para Ascensión Recta y dos para Declinación.

Algo que es muy importante y debe siempre tomarse en cuenta es el ajuste de ganancia de estos amplificadores, ya que esto permitirá un movimiento suave del radio telescopio y alcanzar la estabilidad óptima del sistema. Es por esto que los pasos a seguir para realizar el ajuste de ganancia, se han añadido en el manual de errores y en el anexo E que contiene la documentación para esta tarjeta.

Figura 26 Tarjeta de amplificadores.

Las terminales del amplificador se describen en la tabla del anexo E. así como más información sobre las tarjetas de amplificadores.

También se puede encontrar la información de los amplificadores en la dirección de la red:
http://www.servosystems.com/amc_dcbrush.htm
Número de parte: 10A8.

Terminales DIN

Se usan terminales insertables en un riel DIN para la interconexión de los componentes del sistema de control. Ya que estas terminales tienen más de 80 pines, la correspondencia entre las señales y su función se presenta en el anexo F.

Radio Telescopio (Montura)

El RT5 tiene un plato de 5 metros de diámetro con una montura ecuatorial cuya distribución de partes y ejes podemos ver en la Figura 4, Figura 5 y Figura 6 respectivamente.

En esta estructura están montados:

- Contrapesos.
- Dos motores por cada uno de los ejes.
- Engranes de reducción de velocidad.
- Un Codificador por cada uno de los ejes.
- Antena parabólica.
- Receptores.
- Octágono.
- Inclinómetro.

Contrapesos

Tienen la función de mantener en balance la montura. El balance de la montura es importante para que los movimientos del RT5 al seguir o apuntar un objeto celeste sean suaves y precisos.

Actualmente están montados 5 contrapesos de plomo de 10 Kg cada uno en el octágono, por el otro lado se tienen contrapesos fijos en el RT5.

Motores

Cada eje de la montura es accionado por dos motores de torque de corriente continua.

Cada motor tiene las siguientes especificaciones:

Marca: INLAND MOTOR CORPORATION

Modelo No: T-10004-E

No Serie: 62172-2

	Voltaje [V]	Corriente [A]	Torque [Lb Ft]
continuo	38	3.7	37
pico	102	10	100

Engranes de reducción de velocidad

- **EJE AR**

  Corona: 291 dientes

  Piñón: 18 dientes

  De lo que resultan 291/18 * 36 rev. por la reducción del codificador. Es decir se tienen 582 rev. del codificador por una rev. del eje de AR.

  En pulsos por seg. de arco [582 * 14400 / (360*3600)] = 6.46666667

- **EJE DEC**

  Se tiene 3 reducciones:

  Motor -> 6.5 -> 8 -> (**x/18)**

  Donde x número de dientes de la corona no lo tengo.

  Si suponemos que las coronas de AR y DEC son iguales se tiene:

  (6.5)(8)(291/18) (14400)/(360*3600) = 9.340740740740 pulsos por seg. de arco.

Codificadores o Encoders

En cada eje hay un codificador en cuadratura acoplado al eje.

Fabricante: Gurley Precision Instruments

Modelo:9220S03600Q5L01C18SQ04EN

Resolución: 3600 pulsos por revolución (14400 en cuadratura).

Figura 27 Decodificadores.

Salida	Color de cable	PIN DB15
A	amarillo	8
/A	cafe	7
B	verde	5
/B	naranja	4
IND	azul	2
/IND	blanco	1
FLT	violeta	12
/FLT	gris	11
+V	rojo	10
COMUN	negro	13
CARCAZA	cable sin aislante	9
N.C.		3,6,14,15

Octágono

Sirve para montar sobre él los receptores y se encuentra en el foco de la Antena. Otra de las funciones y es montar sobre él, el espejo secundario.

En la Figura 28 se muestra una imagen de un receptor montado sobre el octágono.

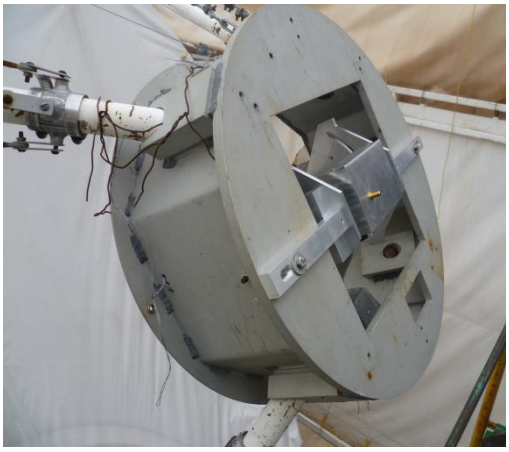

Figura 28 Receptor montado sobre el octágono del RT5.

Inclinómetro

Está sobre la montura y nos brinda en grados la información de la inclinación del RT5.

Ya que nos otorga la medida del ángulo a partir de la horizontal para X y Y, la función principal del inclinómetro en el RT5 es precisamente marcarnos el plano horizontal (X, Y) para asegurar que la antena está apuntada al cenit y entonces darle la referencia a los encoders, ya que estos son de tipo incremental y necesitan la referencia para saber dónde está su cero.

El inclinómetro montado en el RT5 fué construido especialmente para este instrumento, para mayor información al respecto, revisar las referencias.

El inclinómetro utilizado es el ADIS 16209 el cual provee la medición de los dos ejes ortogonales, X y Y con respecto a la horizontal, con un intervalo de medición de ±90° con una precisión de 0.1° y una resolución de 0.025°. Tiene una sensibilidad en aceleración 0.244 E-3 g en los dos ejes.

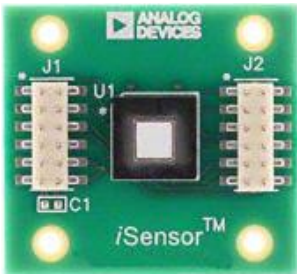

Figura 29 inclinómetro ADIS 16209

El diagrama de la Figura 30, muestra lo que fue necesario para la instrumentación del inclinómetro.

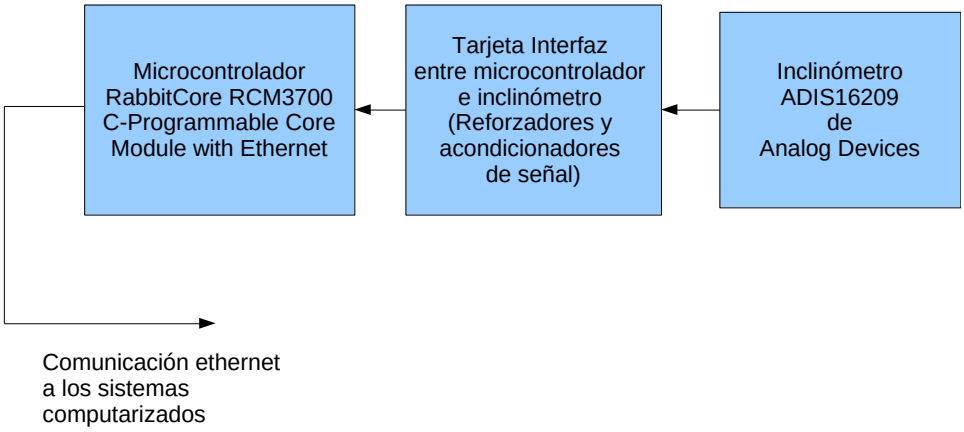

Figura 30 Diagrama a bloques del inclinómetro.

El inclinómetro es leído directamente por la PC de control mediante una conexión Ethernet.

Descripción del sistema de control de movimiento del RT5 (software)

El software que involucra el sistema de control de movimiento del Radio Telescopio de 5 metros, va desde la interfaz de usuario en la computadora personal, hasta los algoritmos de control y de adquisición de datos de la PC industrial en el gabinete de control.

Interfaz de Usuario

La interfaz de usuario del sistema de control del RT5, está realizada en el programa GLADE, que es un desarrollador de interfaces de usuario para el kit de herramientas GTK+ y el entorno de escritorio GNOME, las interfaces de usuario diseñadas con Glade se guardan como archivos XML, pero mediante el objeto GTK Builder GTK estos pueden ser cargados por diferentes aplicaciones según sea necesario. Mediante el GTKBuilder, los archivos XML de Glade se pueden utilizar en numerosos lenguajes de programación como lo son C, C++, C#, Vala, Java, Perl, Python, y otros. Glade es software libre distribuido bajo la licencia GNU GPL .

En el caso de la interfaz de usuario para control del RT5, los archivos XML de Glade son utilizados en el lenguaje C, en un sistema operativo Linux. La Figura 31 muestra la interfaz de usuario desarrollada en Glade, abierta desde el mismo programa y no por la aplicación final.

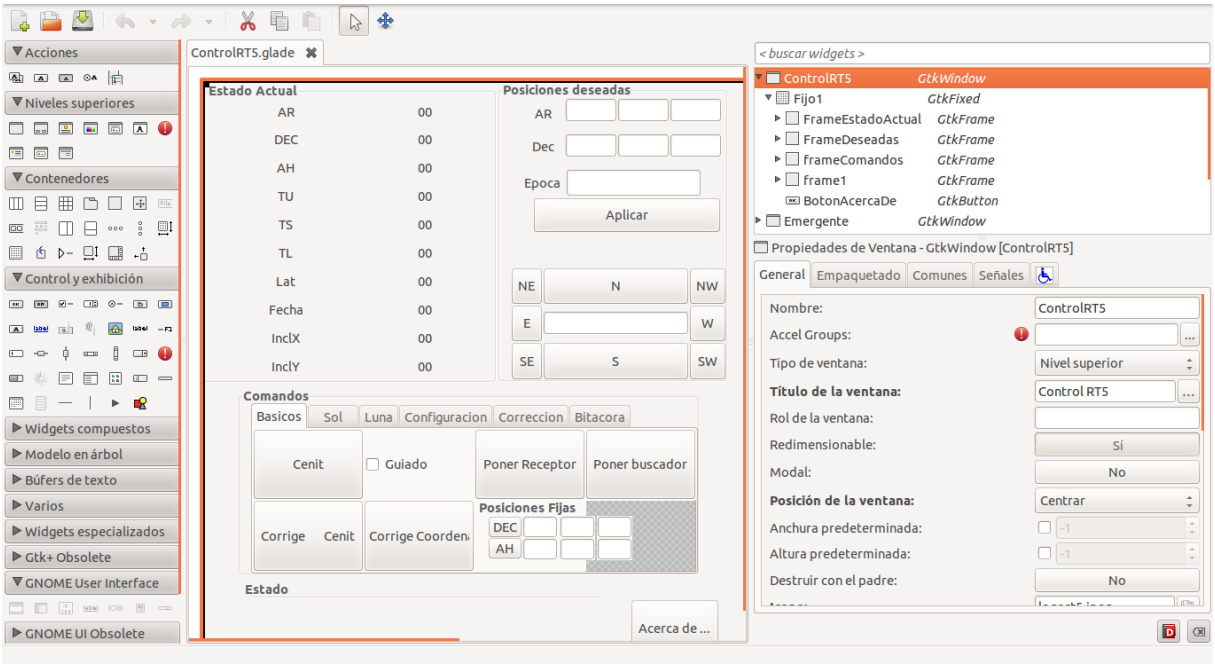

Figura 31Interfaz de usuario del RT5 abierta desde glade.

Como mencionado anteriormente, la interfaz de usuario desarrollada en Glade, es utilizada con el lenguaje C, en el que se programan las funciones de cada botón de dicha interfaz, así como todas las acciones que esta puede realizar. Básicamente, cada botón programado en C, ejecuta una instrucción que es enviada mediante comunicación Ethernet a la PC de control, la cual a partir de dicha instrucción, ejecuta los algoritmos de control que en ella están programados, ya sea para mover el

RT5, detenerlo o simplemente estar en comunicación constante con la interfaz de usuario. Por ejemplo, cuando desde la interfaz de usuario se manda un movimiento del RT5, la posición deseada es enviada vía Ethernet a la PC de control, a partir de la posición en que se encuentra el RT5 y la posición deseada, la PC de control ejecuta los algoritmos que tiene programados para que el movimiento del RT5 sea controlado y preciso.

El programa en C para ejecutar la Interfaz de usuario, antes de poder ser usada en la PC personal, primero que nada debe ser compilado en la misma mediante la ejecución en una terminal de Linux del Shell llamado *"compila.sh"* que se incluye en el Anexo G. En el anexo H se incluye el código fuente "ControlRT5.c" que al ser compilado nos permite visualizar la interfaz de usuario así como el código de la librería del socket "benisocket.h" que permite la comunicación Ethernet constante entre PC personal y PC de control, en Anexo I.

La Figura 32 muestra la Interfaz de usuario, cuando ya es ejecutada y esta lista para usarse.

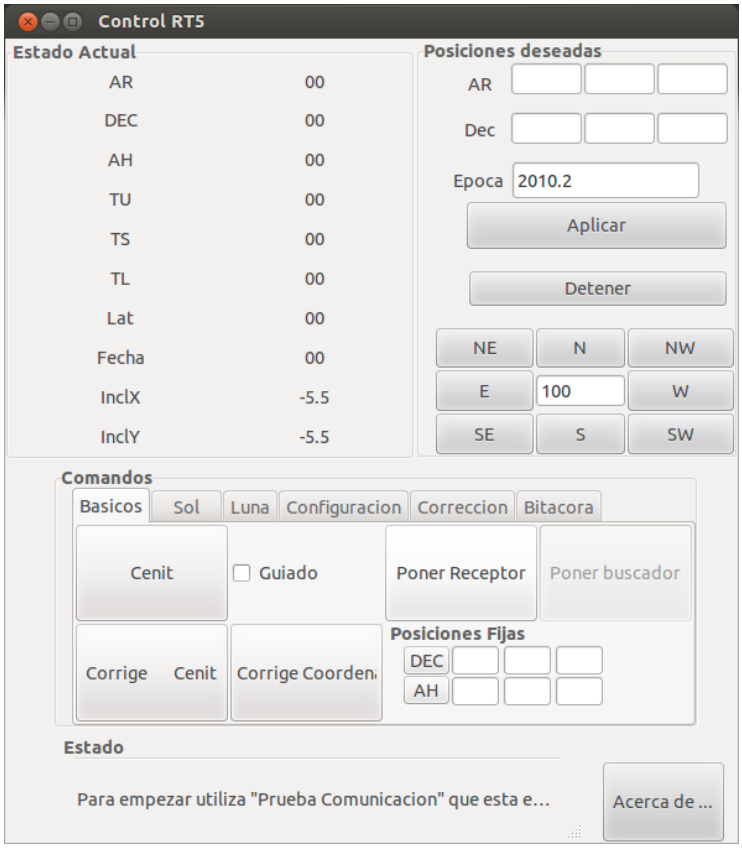

Figura 32 Interfaz de Usuario del sistema de control del RT5 lista para utilizarse.

En la Figura 32 se pueden distinguir cuatro secciones distinguidas en letras negritas, estas son, Estado Actual; en esta sección se muestran los datos de la posición actual del RT5, si como algunos datos esenciales como lo son fecha y posición de la ciudad desde donde se observa, Posiciones Deseadas; aquí se ingresan las coordenadas en Ascensión Recta y Declinación de donde se desea apuntar el RT5, Comandos; esta sección, a su vez tiene varias pestañas, que según el título de cada una de ellas y de los botones que contienen, pueden realizar acciones relacionadas, Estado; es donde se muestra el estado de la interfaz de usuario y de la comunicación, entre ella y la PC de control.

Descripción de las secciones de la interfaz de usuario

El actual sistema de control del RT5 no es del todo automático, ya que antes de iniciar la interfaz de usuario y mover automática y controladamente la antena, se debe establecer la conexión Ethernet entre PC de control y PC personal, de forma manual, mediante un par de comandos en la terminal de Linux donde se ejecutará la interfaz.

Estos comandos son:

$sudo ifconfig eth0 192.168.0.10

> *Que al introducir la contraseña de usuario,nos permite configurar la dirección IP de la PC personal con la de la PC de control.*

$ping 192.168.0.205

> *Que nos permite configurar la red.*

Una vez la ejecución exitosa de los comandos anteriores y teniendo compilado el programa *"controlrt5"* que genera el ejecutable del mismo nombre, este se puede iniciar desde terminal mediante la instrucción *$./controlrt5* o bien haciendo doble click sobre el ejecutable en la carpeta que lo contiene, ejecutándose la interfaz de la Figura 32.

Una vez abierta la Interfaz de usuario, para comenzar a hacer observaciones y el uso de las funciones de la Interfaz de usuario, hay una serie de pasos a realizar, estos son descritos con detalle en el Manual de uso del RT5, por lo cual no serán descritos en este documento.

Estado Actual

En esta sección se muestra, como su nombre lo dice, la posición actual de la antena, los datos *AR, DEC, AH, TU, TS, TL, Lat y Fecha* con datos obtenidos de la PC de control, ya que a ella están conectado, mediante la tarjeta MFIO-3A, los inclinómetros, por lo tanto la PC de control interpreta la información de los mismos, la convierte a *AR, DEC, etc* y la envía a la interfaz para ser mostrada, la Lat y Fecha, también son las de la PC de control, aún cuando esta esté sincronizada con la PC personal.

Los datos *InclX* e *InclY*, es la posición en grados, obtenida del inclinómetro montado en la antena, estos datos al igual que los de los encoders, son recibidos, pero mediante Ethernet, por la PC de control donde también son interpretados y enviados a la interfaz para ser mostrados.

Posiciones Deseadas

En esta sección se introducen la posición en *AR y DEC* a los que se quiere mover la antena en los recuadros en blanco, así como la época en la que se están realizando las observaciones, una vez introducidos estos valores, se presiona el botón *Aplicar*, lo que hará que estos datos se envíen mediante Ethernet a la PC de control, donde serán interpretados para mediante la tarjeta MFIO-3A, enviar la información a la etapa de amplificación que a su vez permite el movimiento controlado hasta la posición deseada, de los motores que mueven la antena.

En esta misma sección se localiza el *botón Detener*, el cual permite detener el movimiento de la antena, en caso de que así se desee, ya sea debido a la detección de alguna falla, o simplemente

porque se desea corregir la posición final a que se moverá la antena. El movimiento de la antena es detenido sin la necesidad de abrir el lazo de control, como lo es en el caso del uso del botón rojo de paro de emergencia, para reiniciar o continuar con las observaciones.

Al presionar el botón Detener, cuando la antena está en movimiento, esta de detiene en un rango de +-2° a partir del momento en que el botón es presionado, esto con la finalidad de que el paro no sea brusco y pueda dañar el sistema. Una vez detenido el movimiento de la antena, y si se desea cambiar las posiciones a la que se moverá la antena, la nueva posición se introduce en los recuadros blancos y se presiona Aplicar, ya que el sistema sigue con su funcionamiento normal.

Comandos

La sección comandos incluye varias pestañas que son: Básico, Sol, Luna, Configuración, Corrección y Bitácora.

La siguiente tabla, muestra las funciones o comandos que se pueden ejecutar en cada una de las pestañas de esta sección, así como una pequeña descripción de aquellas que se les conoce y se usan constantemente.

Básicos	Cenit	Lleva la antena a la posición del cenit.
	Guiado	
	Poner Receptor	
	Corrige Cenit	Corrige la posición inicial del cenit, tomada de la posición inicial de la antena, con la posición real del Cenit después de haber realizado los ajustes necesarios.
	Corrige Coordenadas	
	Posiciones Fijas	
Sol	Sol	
	Guiado Sol	Al estar seleccionado, y después d ehaber apuntado al Sol, permite hacer el seguimiento constante del mismo.
	Barrido Sol	
	Tamaño Barrido (min)	
Luna		
Configuración	Inicializar Cerrar Lazo	Inicia la lectura de inclinómetro y encoder para cerrar el lazo de control.

	Abrir Lazo	Abre el lazo de control.
	Prueba de comunicación	Permite saber si se ha establecido correctamente la comunicación entre PC de control e Interfaz de usuario.
	Lectura Inclinómetro	Al estar seleccionado permite la lectura de inclinómetros.
	Corrige Hora	Sincroniza la hora de la PC de control con la del usuario.
	Busca Cenit	Permite, antes de iniciar la observación, apuntar la antena al cenit basándose en la información del inclinómetro.
Corrección		
Bitácora	Usar Bitácora	Al estar seleccionado, escribe constantemente en el archivo d ela bitácora de observación.
	Ver Bitácora	Abre una ventana nueva con el texto que se ha grabado en la bitácora, en esta ventana es posible modificar la bitácora.
	Grabar Encabezado	Esta formado por un botón que graba en la bitácora el contenido de tres pestañas en donde se ingresa información sobre el instrumento utilizado, los participantes en la observación e incluso algunos comentarios relevantes.

Bibliografía

Precision micro Dynamics Inc. *MFIO-3A User's Manual, PC Motion Control Interface Card Version 1.5*. Victoria, British Columbia, Canada.

ADVANCED MOTION CONTROLS. SERIES *MC/MF MOUNTING CARDS Models: MC1X510, MC2X510, MC3X510, MF1X510, MF2X510 and MF3X510*, Montville, NJ 07045.

Lozano F., Orozco B., Hiriart D., Zazueta S. *DISEÑO DEL CONTROL DE MOVIMIENTO DEL TELESCOPIO 5m, SECCION DE DIAGRAMAS A BLOQUES (2006).*

Lozano F., Orozco B., Hiriart D., Zazueta S. & Murillo F. (2011). *Sistema de Inclinómetro Digital para Telescopios Astronómicos*. Instituto de Astronomía U.N.A.M.

Fuentes electrónicas

http://almaak.tripod.com/temas/radioastronomia.htm

http://radioastronomia.galeon.com/

HINOJOSA RODRIGO C. (2007). Radioastronomía. Revista de astronomía Argo Navis (Chile).

ANEXO A

Terminal A_IDC

PIN	TIRA A_IDC	DESCRIPCIÓN Y/O SEÑAL
1	DAC1	Salida al amplificador de DEC
2	AGND	Tierra Señal
3	DAC2	Salida al amplificador de AR
4	AGND	Tierra Señal
5	DAC3	
6	AGND	
7	N.C	
8	N.C	
9	N.C	
10	N.C	
11	N.C	
12	N.C	
13	GND	
14	5V	
15	PA0	AR Este
16	PA1	AR Oeste
17	PA2	DEC Sur
18	PA3	DEC Norte
19	PA4	Rápidos/Lentos
20	PA5	
21	PA6	
22	PA7	
23	PB0	Interruptor Límite Norte
24	PB1	Interruptor Límite Sur
25	PB2	Interruptor Límite Este
26	PB3	Interruptor Límite Oeste
27	PB4	
28	PB5	
29	PB6	
30	PB7	
31	PC0	Activa amplificadores y/o frenos del telescopio (Activo en bajo)
32	PC1	
33	PC2	
34	PC3	
35	PC4	
36	PC5	
37	PC6	
38	PC7	
39	5V	
40	GND	

ANEXO A

Terminal B_IDC

PIN	TIRA B_IDC	DESCRIPCIÓN Y/O SEÑAL
1	.+5	
2	GND	
3	.+5	
4	GND	
5	.+5	
6	GND	
7	WDOG	
8	GND	
9	.+12	
10	GND	
11	.+5	
12	GND	
13	A1	FASE A DEC
14	/A1	FASE /A DEC
15	B1	FASE B DEC
16	/B1	FASE /B DEC
17	Z1	INDICE DEC
18	/Z1	INDICE /DEC
19	.+12	
20	GND	
21	.+5	
22	GND	
23	A2	FASE A AR
24	/A2	FASE /A AR
25	B2	FASE B AR
26	/B2	FASE /B AR
27	Z2	INDICE AR
28	/Z2	INDICE /AR
29	.+12	
30	GND	
31	.+5	
32	GND	
33	A3	
34	/A3	
35	B3	
36	/B3	
37	Z3	
38	/Z3	
39	SYNC	
40	GND	

ANEXO B

Tabla de conexiones Tarjeta de Optoacoplador para frenos y amplificadores.

CONECTOR JP1		
PIN	DESCRIPCIÓN	CONECTA A
1	ACTIVA AMPL. Y/O FRENOS (ACTIVA EN NIVEL BAJO)	31 TIRA A_IDC
2	TIERRA	40 TIRA A_IDC

CONECTOR JPA (ALIMENTACIÓN)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 13 TERMINAL DIN
2	TIERRA	TIERRA 34 TERMINAL DIN

CONECTOR JPS (SALIDA)		
PIN	DESCRIPCIÓN	CONECTA A
1	CONTACTO 1 RELEVADOR ESTÁTICO.	A BOBINA (+12V) RELEVADOR 1
2	CONTACTO 2 RELEVADOR ESTÁTICO.	A BOBINA (-) RELEVADOR 1

ANEXO C

Tabla de conexiones para tarjeta de Optoacopladores de la paleta manual y sensores de proximidad

CONECTOR JP1 (SENSOR DE PROXIMIDAD)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 46 (TERMINAL DIN) (ALIMENTACIÓN DEL SENSOR)
2	SEÑAL DEL SENSOR (NORTE)	PIN 48 (TERMINAL DIN)
3	TIERRA	PIN 50 (TERMINAL DIN)(COMUN ALIMENTACIÓN Y SEÑAL DEL SENSOR)

CONECTOR JP2(SENSOR DE PROXIMIDAD)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 40 (TERMINAL DIN) (ALIMENTACIÓN DEL SENSOR)
2	SEÑAL DEL SENSOR (SUR)	PIN 42 (TERMINAL DIN)
3	TIERRA	PIN 44 (TERMINAL DIN)(COMUN ALIMENTACIÓN Y SEÑAL DEL SENSOR)

CONECTOR JP3(SENSOR DE PROXIMIDAD)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 45(TERMINAL DIN) (ALIMENTACIÓN DEL SENSOR)
2	SEÑAL DEL SENSOR (ESTE)	PIN 47 (TERMINAL DIN)
3	TIERRA	PIN 49 (TERMINAL DIN)(COMUN ALIMENTACIÓN Y SEÑAL DEL SENSOR)

CONECTOR JP4(SENSOR DE PROXIMIDAD)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 39(TERMINAL DIN) (ALIMENTACIÓN DEL SENSOR)
2	SEÑAL DEL SENSOR (OESTE)	PIN 41 (TERMINAL DIN)
3	TIERRA	PIN 43 (TERMINAL DIN)(COMUN ALIMENTACIÓN Y SEÑAL DEL SENSOR)

ANEXOC

CONECTOR JP5 (SALIDAS OPTO ACOPLADAS)		
PIN	DESCRIPCIÓN	CONECTA A
1	NORTE	PIN 18 TIRA A_IDC
2	SUR	PIN 17 TIRA A_IDC
3	ESTE	PIN 15 TIRA A_IDC
4	OESTE	PIN 16 TIRA A_IDC
5	R/L (RÁPIDO O LENTO)	PIN 19 TIRA A_IDC
6	LÍMITE NORTE	PIN 23 TIRA A_IDC
7	LÍMITE SUR	PIN 24 TIRA A_IDC
8	LÍMITE ESTE	PIN 25 TIRA A_IDC
9	LÍMITE OESTE	PIN 26 TIRA A_IDC
10 a 22	NO CONECTADAS	
23,24,25	TIERRA	PIN 40 TIRA A_IDC

CONECTOR JP6 (ALIMENTACIÓN)		
PIN	DESCRIPCIÓN	CONECTA A
1	.+12V	PIN 16 TERMINAL DIN (ALIMENTACIÓN DE LA TARJETA)
2	NO CONECTADO	PIN 34 TERMINAL DIN (COMÚN)
3	TIERRA	
4	.+5V	PIN 28 TERMINAL DIN (ALIMENTACIÓN DE LA TARJETA)

ANEXOC

CONECTOR JP7 (ENTRADAS DE PALETA)		
PIN	DESCRIPCIÓN	CONECTA A
1	NORTE (+)	PIN 51 TERMINAL DIN (INTERRUPTOR DE PALETA)
2	NORTE (-)	PIN 52 TERMINAL DIN (INTERRUPTOR DE PALETA)
3	SUR (+)	PIN 53 TERMINAL DIN (INTERRUPTOR DE PALETA)
4	SUR (-)	PIN 54 TERMINAL DIN (INTERRUPTOR DE PALETA)
5	ESTE (+)	PIN 55 TERMINAL DIN (INTERRUPTOR DE PALETA)
6	ESTE (-)	PIN 56 TERMINAL DIN (INTERRUPTOR DE PALETA)
7	OESTE (+)	PIN 57 TERMINAL DIN (INTERRUPTOR DE PALETA)
8	OESTE (-)	PIN 58 TERMINAL DIN (INTERRUPTOR DE PALETA)
9	R/L (+)	PIN 59 TERMINAL DIN (INTERRUPTOR DE PALETA)
10	R/L (-)	PIN 60 TERMINAL DIN (INTERRUPTOR DE PALETA)

ANEXO D

Tabla de conexiones de la tarjeta de Preamplificadores

CONECTOR JP1 (ENTRADA DE CONTROL)		
PIN	DESCRIPCIÓN	CONECTA A
1	Vi (VOLTAJE DE ENTRADA DE CONTROL)	PIN 1 TIRA A_IDC
2 Y 3	NO CONECTADOS	
4	TIERRA	PIN 2 TIRA A_IDC

CONECTOR JP2 (SALIDAS DE CONTROL A APMPLIFICADORES)		
PIN	DESCRIPCIÓN	CONECTA A
1	VA1 (VOLTAJE DE ENTRADA DE CONTROL AMPLIFICADOR 1)	PIN 5 (TIRA TERMINAL DEL AMPLIFICADOR)
2	VA2 (VOLTAJE DE ENTRADA DE CONTROL AMPLIFICADOR 2)	PIN 5 (TIRA TERMINAL DEL AMPLIFICADOR)
3	NO CONECTADO	
4	TIERRA	PIN 6 (TIRA TERMINAL DEL AMPLIFICADOR)

CONECTOR JP3 (ALIMENTACIÓN)		
PIN	DECRIPCIÓN	CONECTA A
1	.+12V	PIN 18 (TERMINAL DIN)
2	TIERRA	PIN 36 (TERMINAL DIN)
3	NO CONECTADO	
4	.-12V	PIN 24 (TERMINAL DIN)

ANEXO E

Calibración de la ganancia en la tarjeta de Amplificadores

Para calibrar la ganancia de los amplificadores deben seguirse los siguientes pasos :

- Conectar "+ C" a "+ 10 V" en la terminal de tornillo.
- Ajustar el Pot 2,7, 12 (´para los canales#1, 2, 3) para obtener -5 V en el punto de prueba localizado entre Pot 2, 7, 12 y el borde de la tarjeta.
- Ajustar el potenciometro localizado a un lado del modulo de amplificación para obtener -7.25 V en los contactos izquierdos de C3, 43, y 63 (Los cuales estan conectados al pin 10 del modulo).

Notas:

- “+C” es el PIN 5 del conector P3, P4, P5
- Para cualquier duda en la ubicación de las terminales mencionadas, referirse al anexo ¿ que contiene la tabla y diagramas de conexiones de las tarjetas de Amplificador.

Tablas de funciones y conexiones de la Tarjeta de Amplificadores

FUNCIONES DE POTENCIÓMETROS (EJES 1,2 Y 3 RESPECTIVAMENTE		
POTENCIOMETRO	DESCRIPCIÓN	GIRO EN SENTIDO DEL RELOJ
POT 5, 10, 15	AJUSTE DE GANANCIA DE TACÓMETRO (NORMALMENTE NO NECESARIO Y NO INSTALADO DE FABRICA)	INCREMENTA
POT 4, 9, 14	AJUSTE DE LA CANTIDAD DE COMPENSACIÓN DE RETROALIMENTACIÓN DE IR	INCREMENTA
POT 3, 8, 13	AJUSTE D ELÍMITE DE CORRIENTE	INCREMENTA
POT 2, 7, 12	AJUSTE DE GANANCIA DE ENTRADA	INCREMENTA
POT 1, 6, 11	AJUSTE DE CUALQUIER OFFSET O INBALANCE EN EL ENSAMBLE DE LA TARJETA DE AMPLIFICADOR O EN LA SEÑAL DE ENTRADA	N/A

ANEXO E

Tablas de funciones y conexiones de la Tarjeta de Amplificadores

CONECTOR	PIN	NOMBRE	DESCRIPCIÓN/NOTAS	I/O
P2	1	.+HV	ALIMENTACION CD	I
	2	GND	TIERRA DE ALIMENTACIÓN	GND
	3	gnd	SEÑAL DE TIERRA (LA MISMA QUE P2-2)	GND
	4	.+10V @ 3mA	PARA USO DEL CLIENTE	O
	5	.-10V @ 3mA	PARA USO DEL CLIENTE	O

CONECTOR	PIN	NOMBRE	DESCRIPCIÓN/NOTAS	I/O
P3,P4,P5	1	.-M1, -M2, -M3	.-MOTOR	O
	2	.+M1, +M2, +M3	.+MOTOR	O
	3	.-T1, -T2, -T3	.-TACOMETRO	I
	4	.+T1, +T2, +T3	.+TACOMETRO	I
	5	.-C1, -C2, -C3	.+SEÑAL DE COMANDO	I
	6	.+C1, +C2, +C3	.-SEÑAL DE COMANDO	I
	7	INH 1,2,3	APLICAR DE +3V A +15V @ 3mA PARA INHIBIR	I
	8	gnd	TIERRA DE REFERENCIA	GND

SELECCIÓN DE MODO				
MODOS DE OPERACIÓN	SW1	SW2	SW3	SW4
MODO DE CORRIENTE	OFF	ON	OFF	OFF
MODO DE VOLTAJE	OFF	OFF	ON	OFF
MODO DE COMPENSACIÓN IR	OFF	OFF	ON	ON
MODO CON TACÓMETRO	OFF	OFF	OFF	OFF
MODO DE PRUEBA	ON	OFF	ON	OFF
MODO DE POSICIÓN	OFF	OFF	ON	OFF

ANEXO F

Tabla de señales de las terminales DIN

PIN	DESCRIPCIÓN DE LA SEÑAL
1	FASE A DEC (A1)
2	FASE /A DEC (/A1)
3	FASE B DEC (B1)
4	FASE /B DES (/B1)
5	IND. DEC (Z1)
6	/IND. DEC (/Z1)
7	FASE A AR (A2)
8	FASE /A AR (/A2)
9	FASE B AR (B2)
10	FASE /B AR (/B2)
11	IND. AR (Z2)
12	/IND. AR (/Z2)
13 … 20	.+12VDC
21 … 24	.-12VDC
25 … 30	.+5VDC
31 … 38	TIERRA
39	.+12VDC
40	.+12VDC
41	SENSOR PROXIMIDAD OESTE
42	SENSOR PROXIMIDAD SUR
43	TIERRA
44	TIERRA
45	.+12VDC
46	.+12VDC
47	SENSOR PROXIMIDAD ESTE
48	SENSOR PROXIMIDAD NORTE
49	TIERRA
50	TIERRA

ANEXO F

PIN	DESCRIPCIÓN DE LA SEÑAL
51	PALETA NORTE (+)
52	PALETA NORTE (-)
53	PALETA SUR (+)
54	PALETA SUR (-)
55	PALETA ESTE (+)
56	PALETA ESTE (-)
57	PALETA OESTE (+)
58	PALETA OESTE (-)
59	PALETA RÁPIDO/LENTO (+)
60	PALETA RÁPIDO/LENTO (-)
61 … 70	DISPONIBLES
71	TIERRA (FRENOS AR)
72	.+12VDC (FRENOS AR)
73	TIERRA (FRENOS DEC)
74	.+54VDC (FRENOS DEC)
75	.+12VDC (INTERRUPTOR DE SEGURIDAD)
76	RETORNO (INTERRUPTOR DE SEGURIDAD)
77,79,81,83,85	NEUTRO ALIMENTACIÓN 110VCA
78,80,82,84,86	FASE ALIMENTACIÓN 110VCS
87	TIERRA DE SEGURIDAD 110VCA (TIERRA FÍSICA)

Anexo G compila.sh

```
# basic GTK+ app makefile

gcc -o controlrt5-prueba ControlRT5.c $(pkg-config --cflags --libs gtk+-2.0 gmodule-2.0)

# end of file
```

Anexo H ControlRT5.c

```
/*
 * Compilar con:
 *   gcc -o controlrt5 ControlRT5.c $(pkg-config --cflags --libs gtk+-
2.0 gmodule-2.0)
 */

#include <gtk/gtk.h>
#include <gdk/gdkkeysyms.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include "benisocket.h"

#define deg2rad       1.745329251994330E-2   /* Degrees to radians */
#define pi       3.14159265358979323846 /* Pi */
#define twopi         6.28318530717958623    /* 2*Pi  */

struct     {  char callsign[17];
         double stnlat;
         double stnlong;
         int stnalt;
      }  qth;

    GtkWidget  *window;

GtkWidget *BitacoraTexto;
GtkWidget *TextoBitacoraInstrumento;
GtkWidget *TextoBitacoraParticipantes;
GtkWidget *TextoBitacoraComentarios;
char nombre_archivo_bitacora[200];
GtkWidget  *EntryARHora;
GtkWidget  *EntryARMinuto;
GtkWidget  *EntryARSeg;
GtkWidget  *EntryDecGrado;
GtkWidget  *EntryDecMinuto;
GtkWidget  *EntryDecSeg;
GtkWidget *EntryOffset;
GtkWidget *EntryEpoca;
GtkWidget  *EntryDecFijoG;
GtkWidget  *EntryDecFijoM;
GtkWidget  *EntryDecFijoS;
GtkWidget  *EntryAHFijoH;
GtkWidget  *EntryAHFijoM;
GtkWidget  *EntryAHFijoS;
GtkWidget  *EntradaCorreccionAcimut;
```

```
GtkWidget  *EntradaCorreccionAltura;

GtkWidget * LabelAR;
GtkWidget * LabelDec;
GtkWidget * LabelAH;
GtkWidget * LabelTS;
GtkWidget * LabelTL;
GtkWidget * LabelTU;
GtkWidget * LabelFecha;
GtkWidget * LabelLat;
GtkWidget * LabelInclinometroX;
GtkWidget * LabelInclinometroY;
GtkWidget * BarraEstado;

GtkWidget * ToggleBuscaCenit;
GtkWidget * ToggleGenerarBitacora;
GtkWidget * ToggleActivarCorreccion;
GtkWidget * BotonPonerBuscador;

gint TagInclinometro;
gint TagSol;
gint TagLuna;
gint TagActualiza;
gint TagBuscaCenit;
gint TemporizadorSol=15000;
gint TemporizadorLuna=1000;
gint TemporizadorInclinometro=200;
gint TemporizadorActualiza=1000;
gint TemporizadorBuscaCenit=5000;
gint id_contexto_barraestado;
double PI=3.141592;
gint manejador_snooper;

double moon_az, moon_el, moon_dx, moon_ra, moon_dec, moon_gha, moon_dv;
char buffbitacora[200];

void agrega_a_bitacora(const char * texto)
{
  FILE *archivobitacora;
  time_t rawtime;
  struct tm * timeinfo;
  //char nombre[200];
  char texto_bitacora[30000];

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );
  sprintf
(nombre_archivo_bitacora,"Bitacora_rt5_%04d%02d%02d.txt",timeinfo-
>tm_year+1900,timeinfo->tm_mon+1,timeinfo->tm_mday);

  archivobitacora=fopen(nombre_archivo_bitacora,"a");

  sprintf(texto_bitacora,"%02d:%02d:%02d %s\n",timeinfo-
>tm_hour,timeinfo->tm_min,timeinfo->tm_sec,texto);
  fwrite(texto_bitacora,1,strlen(texto_bitacora),archivobitacora);
  fclose(archivobitacora);
}
```

```
int CorreccionSencillaDec(int *DCG,int *DCM,double *DCS,int HSH,int
HSM,double HSS,double CorrAZ,double CorrAL)
{
    float hs,Dec;
    hs = HSH + (double)(HSM) / 60.0+HSS/3600;
    Dec= (*DCG) + (double)(*DCM)/60.0+(*DCS)/3600.0;
    Dec-=hs*CorrAZ;
    Dec-=CorrAL;

    *DCG=floor(Dec);
    *DCM=floor((Dec - floor(Dec)) * 60);
    *DCS=((Dec -floor(Dec)) * 60 - (*DCM)) * 60;
    if (Dec<0)
    {
          *DCM=-*DCM;
          *DCS=-*DCS;
    }
}
/*
int ecu2hor(int AH,int AM,double AS,int DG,int DM,double DS,int HSH,int
HSM,double HSS,double *AZ,double *AL)
{
    double hs,ar,dec,latitud,h,altura,acimut;

    hs = HSH + (double)(HSM) / 60.0+HSS/3600;
    ar=AH+(double)(AM)/60.0+AS/3600.0;
      //<!--Declinación-->
    dec=(DG+(double)(DM)/60.0+DS/3600)*PI/180.0;
    printf("Entro hs:%d:%d:%f ar:%f y
dec:%f\n",HSH,HSM,HSS,ar,dec*180.0/PI);
    latitud=(19.0+1.0/60.0+59.69/3600.0)*PI/180.0;//qth.stnlat*deg2rad;
// North latitude of tracking station
    h=(hs-ar)*PI/180.0;

    altura=asin(cos(h)*cos(dec)*cos(latitud)+sin(dec)*sin(latitud));
    acimut=atan2(sin(h)*cos(dec),cos(h)*cos(dec)*sin(latitud)-
sin(dec)*cos(latitud));

    *AZ=acimut*180.0/PI;
    *AL=altura*180.0/PI;
    printf("dentrode ecu2hor altura:%f acimut:%f\n",*AL,*AZ);
}
*/
/*
int hor2ecu(double AZ,double AL,double CorrAZ,double CorrAL,int HSH,int
HSM,double HSS,int *RAH,int *RAM,int *RAS,int *DCG,int *DCM,int *DCS)
{

    double altitud,acimut,hs,latitud,dec,ar,DC,h;
    printf("Entro AZ:%f y AL:%f\n",AZ,AL);
    altitud=(AL+CorrAL)*PI/180.0;
    acimut=(AZ+CorrAZ)*PI/180.0; //rotando 90 grados
    hs = HSH + HSM / 60+HSS/3600;
    latitud=(19.0+1.0/60.0+59.69/3600.0)*PI/180.0;//qth.stnlat*deg2rad;
// North latitude of tracking station

    dec=asin(sin(altitud)*sin(latitud)-
cos(acimut)*cos(altitud)*cos(latitud));
```

```
h=atan2(sin(acimut)*cos(altitud),cos(acimut)*cos(altitud)*sin(latitud)+
sin(altitud)*cos(latitud));

    h=h*180/PI;

    ar=hs-h;

    *RAH=floor(ar);
    *RAM=floor((ar - floor(ar)) * 60);
    *RAS=((ar -floor(ar)) * 60 - *RAM) * 60;

    DC=dec*180/PI;
    printf("dentrode hor2ecu dec:%f ar:%f\n",DC,ar);

    *DCG=floor(DC);
    *DCM=floor((DC - floor(DC)) * 60);
    *DCS=((DC -floor(DC)) * 60 - *DCM) * 60;
    if (DC<0)
    {
          *DCM=-*DCM;
          *DCS=-*DCS;
    }
}
*/
// done hiding from old browsers -->
void on_BotonActivarCorreccion_toggled(GtkToggleButton *button,gpointer
user_data)
{
      if(gtk_toggle_button_get_active(button))
      {

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Se activo la correccion de coordenadas
con %s grados en Acimut y %s en Altura",

      gtk_entry_get_text(GTK_ENTRY(EntradaCorreccionAcimut)),gtk_entry_g
et_text(GTK_ENTRY(EntradaCorreccionAltura)));
            agrega_a_bitacora(buffbitacora);
          }
      }
      else
      {

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Se desactivo la correccion chafa de
coordenadas");
            agrega_a_bitacora(buffbitacora);
          }
      }
}

gint MandaYLeeSocket(char mensaje[],int nrecibir,char *recibido)
{
```

```
      BeniSocket Socket;
      unsigned short port=4955;
      char hostname[]="192.168.0.205";
      char caracteres[200];
      int ini;

      ini=initBeniSocketClient(&Socket,hostname,port);

      if(ini==0)
      {
        Socket.rc=write(Socket.socket,mensaje,strlen(mensaje)+1);
        if(Socket.rc<0)
        {
              printf("Error escritura del socket\n");

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Error escritura del socket");

        }
        else
        {
          read(Socket.socket,caracteres,nrecibir);

          //printf("%s\n",caracteres);
          strcpy(recibido,caracteres);
        }
        close(Socket.socket);
      }
      else

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Error en socket");

      return(Socket.rc);
}

gint ActualizaCoordenadas(gpointer datos)
{
      gtk_timeout_remove( TagActualiza );  //Limpia tiempo de espera ,
para ciclo infinito

      BeniSocket Socket;
      char mensaje[20];
      strcpy(mensaje,"TEL");
      unsigned short port=4955;
      char hostname[]="192.168.0.205";
      char caracteres[200];
      gint
ArH,ArM,DecG,DecM,DecS,AHH,AHM,TSH,TSM,basura1,basura2,TLH,TLM,TUH,TUM;
      float ArS,AHS,TSS,basura3,TLS,TUS;
      char AR[50],DEC[50],AH[50],TS[50],TL[50],TU[50],Fecha[50];

      int init=initBeniSocketClient(&Socket,hostname,port);

      if(init==0)
      {
        Socket.rc=write(Socket.socket,mensaje,10);
        if(Socket.rc<0)
```

```
        {
             g_print("Error escritura del socket\n");
        }
        else
        {
          read(Socket.socket,caracteres,200);

          //printf("%s\n",caracteres);

          sscanf(caracteres,"AR %d:%d:%f",&ArH,&ArM,&ArS);
          strcpy(caracteres,strstr(caracteres,"DEC "));
          sscanf(caracteres,"DEC %d %d'%d",&DecG,&DecM,&DecS);
          strcpy(caracteres,strstr(caracteres,"AH "));
          sscanf(caracteres,"AH %d:%d:%f",&AHH,&AHM,&AHS);
          strcpy(caracteres,strstr(caracteres,"TS: "));
          sscanf(caracteres,"TS: %d:%d:%f",&TSH,&TSM,&TSS);
          strcpy(caracteres,strstr(caracteres,"TL: "));
          sscanf(caracteres,"TL: %d:%d:%f",&TLH,&TLM,&TLS);
          strcpy(caracteres,strstr(caracteres,"TU: "));
          sscanf(caracteres,"TU: %d:%d:%f%s",&TUH,&TUM,&TUS,Fecha);

          sprintf(AR,"%02d:%02d:%0.1f\n",ArH,ArM,ArS);
          gtk_label_set_text(GTK_LABEL (LabelAR),AR);
          sprintf(DEC,"%02d %02d %02d\n",DecG,DecM,DecS);
          gtk_label_set_text(GTK_LABEL (LabelDec),DEC);
          sprintf(AH,"%02d:%02d:%0.1f\n",AHH,AHM,AHS);
          gtk_label_set_text(GTK_LABEL (LabelAH),AH);

          //AR = g_markup_printf_escaped ("<span color=\"blue\"
style=\"italic\" font=\"Century Schoolbook L
30\">%02d:%02d:%02.1f</span>",ArH,ArM,ArS);
          //gtk_label_set_markup (GTK_LABEL (LabelAR), AR);
          //DEC = g_markup_printf_escaped ("<span color=\"red\"
style=\"italic\" font=\"Century Schoolbook L 30\">%02d %02d
%02d</span>",DecG,DecM,DecS);
          //gtk_label_set_markup (GTK_LABEL (LabelDec), DEC);
          //AH = g_markup_printf_escaped ("<span color=\"blue\"
style=\"italic\" font=\"Century Schoolbook L
16\">%02d:%02d:%0.1f</span>",AHH,AHM,AHS);
          //gtk_label_set_markup (GTK_LABEL (LabelAH), AH);
          //g_free (AR);
          //g_free (DEC);
          //g_free (AH);




if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            printf("Activado generando bitacora...\n");
          }
          //sprintf(AR,"%02d:%02d:%0.1f\n",ArH,ArM,ArS);
          //gtk_label_set_text(GTK_LABEL(LabelAR),AR);
          //sprintf(DEC,"%02d %02d %02d\n",DecG,DecM,DecS);
          //gtk_label_set_text(GTK_LABEL(LabelDec),DEC);
          //sprintf(AH,"%02d:%02d:%0.1f\n",AHH,AHM,AHS);
```

```
          //gtk_label_set_text(GTK_LABEL(LabelAH),AH);
          sprintf(TS,"%02d:%02d:%0.1f",TSH,TSM,TSS);
          gtk_label_set_text(GTK_LABEL(LabelTS),TS);
          sprintf(TL,"%02d:%02d:%0.1f",TLH,TLM,TLS);
          gtk_label_set_text(GTK_LABEL(LabelTL),TL);
          sprintf(TU,"%02d:%02d:%0.1f",TUH,TUM,TUS);
          gtk_label_set_text(GTK_LABEL(LabelTU),TU);
          gtk_label_set_text(GTK_LABEL(LabelFecha),Fecha);
          //fTS=(float)TSH+TSM/60.0+TSS/3600.0;
          close(Socket.socket);
        }
      }
      else


gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Error en Socket");

      TagActualiza=gtk_timeout_add(
TemporizadorActualiza,ActualizaCoordenadas,datos );
      return TRUE;
}

gint RutinaInclinometro(gpointer datos)

{
      gtk_timeout_remove( TagInclinometro );

      BeniSocket Socket;
      char mensaje[20];
      char X[100],Y[100];
      strcpy(mensaje,":P;");
      unsigned short port=4545;
      char hostname[]="192.168.0.111";
      char caracteres[200];
      float AnguloX,AnguloY,basurafloat;
      int entera,fraccint,AXi,AXf,AYi,AYf;
      float fraccionaria;

      //int ini=0;
      int ini=initBeniSocketClient(&Socket,hostname,port);

      if(ini==0)

      {
        {
             g_print("Error escritura del socket\n");
        }
        else
        {
          //sprintf(caracteres,":12.34 56.79 12;");

          sscanf(caracteres,":%d.%d %d.%d
%d;",&AXi,&AXf,&AYi,&AYf,&basurafloat);
          sprintf(X,"%02d.%02d",AXi,AXf);
          gtk_label_set_text(GTK_LABEL(LabelInclinometroX),X);
          sprintf(Y,"%02d.%02d",AYi,AYf);
          gtk_label_set_text(GTK_LABEL(LabelInclinometroY),Y);
```

```
          close(Socket.socket);
        }
      }
      else

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Error de socket leyendo inclinometro");

      TagInclinometro=gtk_timeout_add(
TemporizadorInclinometro,RutinaInclinometro,datos );
      return TRUE;
}

void on_BotonInclinometro_toggled(GtkToggleButton *button,gpointer
user_data)

{
      gpointer datos;
      if(gtk_toggle_button_get_active(button))

      {
          printf("Leyendo inclinometro...\n");
          TagInclinometro=gtk_timeout_add(
TemporizadorInclinometro,RutinaInclinometro,datos );

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Leyendo inclinometro");

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Leyendo inclinometro");
            agrega_a_bitacora(buffbitacora);
          }
      }
      else

      {
          printf("Deje de leer inclinometro...\n");
          gtk_timeout_remove( TagInclinometro );

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Deje de leer inclinometro");
                {
            sprintf(buffbitacora,"Deje de leer inclinometro");
            agrega_a_bitacora(buffbitacora);
          }
      }
}

void on_BotonGenerarBitacora_toggled(GtkToggleButton *button,gpointer
user_data)
{
            {
          printf("Activado generando bitacora...\n");
      }
      else
```

```
      {
          printf("Deje de generar bitacora...\n");
      }
}

void on_BotonGrabarEncabezado_clicked(GtkButton *button, gpointer data)
{
    FILE *bitacora;
    printf("Grabando encabezado en bitacora....\n");
    GtkWidget *Texto;
    GtkTextIter   InicioIter;
    GtkTextIter FinIter;
    GtkTextBuffer *bInstrumento;
    GtkTextBuffer *bParticipantes;
    GtkTextBuffer *bComentarios;
    char texto[30000];
    int i=0;
    char texto2[30000];
    char texto3[30000];
    char *apuntador;
    time_t rawtime;
    struct tm * timeinfo;

    printf("1");
    time ( &rawtime );
    timeinfo = localtime ( &rawtime );
    sprintf
(nombre_archivo_bitacora,"Bitacora_rt5_%04d%02d%02d.txt",timeinfo-
>tm_year+1900,timeinfo->tm_mon+1,timeinfo->tm_mday);

    sprintf(texto,"Bitacora de trabajos del RT5\n\nFecha:
%04d/%02d/%02d\n",timeinfo->tm_year+1900,timeinfo->tm_mon+1,timeinfo-
>tm_mday);
    strcat(texto,"PARTICIPANTES:\n");

bParticipantes=gtk_text_view_get_buffer(GTK_TEXT_VIEW(TextoBitacoraPart
icipantes));
    gtk_text_buffer_get_bounds (bParticipantes,&InicioIter,&FinIter);

strcat(texto,gtk_text_buffer_get_text(bParticipantes,&InicioIter,&FinIt
er,FALSE));

bInstrumento=gtk_text_view_get_buffer(GTK_TEXT_VIEW(TextoBitacoraInstru
mento));
    gtk_text_buffer_get_bounds (bInstrumento,&InicioIter,&FinIter);
    strcat(texto,"\n\nINSTRUMENTO:\n");

strcat(texto,gtk_text_buffer_get_text(bInstrumento,&InicioIter,&FinIter
,FALSE));

bComentarios=gtk_text_view_get_buffer(GTK_TEXT_VIEW(TextoBitacoraComent
arios));
    gtk_text_buffer_get_bounds (bComentarios,&InicioIter,&FinIter);
    strcat(texto,"\n\nCOMENTARIOS:\n");

strcat(texto,gtk_text_buffer_get_text(bComentarios,&InicioIter,&FinIter
,FALSE));
```

```
    strcat(texto,"\n\nACTIVIDADES>\n");

    bitacora=fopen(nombre_archivo_bitacora,"r");
    if(bitacora!=NULL) //si el archivo existe lee lo que tiene
    {
      do
      {
      fread(&texto2[i],1,1,bitacora);
      i++;
      }while(!feof(bitacora));
      fclose(bitacora);
      //busca Actividades> para ver si ya existia un encabezado. y lo
borra
      apuntador=strstr(texto2,"ACTIVIDADES>");
      if(apuntador==NULL)
      strncat(texto,texto2,i-1);
      else
      {
        strcpy(texto3,texto2);
        strncat(texto,&texto2[(apuntador-texto2)+13],i-1-(apuntador-
texto2)-13);
      }
    }

    bitacora=fopen(nombre_archivo_bitacora,"w");
    fwrite(texto,1,strlen(texto),bitacora);
    fclose(bitacora);
}

gint SigueSol(gpointer datos)
{
      int decg,decm;
      double decs;
      gint TSH,TSM;
      float TSS;
      double CorrAZ,CorrAL;

      gtk_timeout_remove( TagSol );
      //printf("Calculando Sol...\n");
      char mensaje[200];
      char temporal[200];
      //const float pi=3.141592;
      float anio,mes,dia;
      float
temp,AH,ARr,C,LS,K,L0,M0,M0r,Tt,d,d0,t0,s0,gmst,tu,ts,longitud;
      float ARSol,DECSol;
      time_t rawtime;
      struct tm * ptm;
      int iARH,iARM,iDecG,iDecM;
      float iARS,iDecS;

      time ( &rawtime );
      ptm = gmtime ( &rawtime ); //tambien localtime
      //printf ("hora :   %04d/%02d/%02d %02d:%02d:%02d\n",
      //             ptm->tm_year+1900,ptm->tm_mon+1,ptm->tm_mday,
      //             ptm->tm_hour, ptm->tm_min,ptm->tm_sec);
```

```
      tu = (float)(ptm->tm_hour)+(float)(ptm->tm_min)/60.0 +
(float)(ptm->tm_sec)/3600.0;
      //tu=(float)19+(float)(10)/60.0+(float)(55)/3600.0;
      anio=ptm->tm_year+1900.0;
      mes=ptm->tm_mon+1.0;
      dia=ptm->tm_mday;

      d0 = 367.0*anio - (int)(7.0/4.0
*(anio+(int)((mes+9.0)/12.0)))+(int)(275.0*mes/9.0) +dia-730531.5;
      t0 = d0 / 36525.0;
      // tiempo sideral en Greenwich
      s0 = 6.6974 + 2400.0513 * t0;
      gmst = s0 + (366.2422 / 365.2422) * tu;
      // longitud de puebla 98 12 23
      longitud = -98.20638888888888889/15.0 ;
      ts = gmst + longitud;
      ts = (int)(ts) % 24 + ts-(int)ts;

      d = d0 + tu/24.0;
      Tt = d/36525.0;
      temp=280.466+36000.770*Tt;
      L0 = (int)(temp)%360 + temp-(int)temp;
      temp=357.529+35999.050*Tt;
      M0 = (int)(temp)%360 + temp-(int)temp;
      M0r = M0 * pi/180.0;
      C = (1.915 - 0.005* Tt) *  sin( M0r ) + 0.020 * sin( 2.0*M0r );
      LS = L0 + C;
      K= 23.439 - 0.013 *Tt;
      ARr = atan( tan(LS * pi/180.0)*cos(K* pi/180.0));
      ARSol = ARr *180.0/pi;
      if(ARSol < 0 )
            ARSol = ARSol + 360.0;
      //ARSol=14 % 5;
      DECSol  = 180.0/pi * asin ( sin( ARr  ) * sin( K* pi/180.0 ));

      ARSol = ARSol/15.0; //?, DEC
      AH = ts - ARSol;

      //printf("TS=%f AH=%f AR=%f DEC=%f\n",ts,AH,ARSol,DECSol);

      iARH=ARSol;
      iARM=(ARSol-iARH)*60.0;
      iARS=((ARSol-iARH)*60.0 - iARM)*60.0;
      sprintf(temporal,"%02d",iARH);
      gtk_entry_set_text(GTK_ENTRY(EntryARHora),temporal);
      sprintf(temporal,"%02d",abs(iARM));
      gtk_entry_set_text(GTK_ENTRY(EntryARMinuto),temporal);
      sprintf(temporal,"%0.2f",fabs(iARS));
      gtk_entry_set_text(GTK_ENTRY(EntryARSeg),temporal);

      iDecG=DECSol;
      iDecM=(DECSol-iDecG)*60.0;
      iDecS=((DECSol-iDecG)*60.0 - iDecM)*60.0;

      sprintf(temporal,"%02d",iDecG);
      gtk_entry_set_text(GTK_ENTRY(EntryDecGrado),temporal);
      sprintf(temporal,"%02d",abs(iDecM));
      gtk_entry_set_text(GTK_ENTRY(EntryDecMinuto),temporal);
```

```
      sprintf(temporal,"%0.2f",fabs(iDecS));
      gtk_entry_set_text(GTK_ENTRY(EntryDecSeg),temporal);



      sprintf(mensaje,"DEC %d %d %d AR %d %d %0.1f
ACT",iDecG,abs(iDecM),abs((int)iDecS),iARH,abs(iARM),iARS);
      printf("Siguiendo el sol, posicion actual:\nDec %d %d %d\nAR %d %d
%0.1f\n",iDecG,abs(iDecM),abs((int)iDecS),iARH,abs(iARM),iARS);

      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleActivarCor
reccion)))
      {
          //Obtiene el tiempo sideral.

sscanf(gtk_label_get_text(GTK_LABEL(LabelAH)),"%d:%d:%f",&TSH,&TSM,&TSS
);

CorrAZ=atof(gtk_entry_get_text(GTK_ENTRY(EntradaCorreccionAcimut)));

CorrAL=atof(gtk_entry_get_text(GTK_ENTRY(EntradaCorreccionAltura)));

          //int CorreccionSencillaDec(int *DCG,int *DCM,double *DCS,int
HSH,int HSM,double HSS,double CorrAZ)
          decg=iDecG;
          decm=iDecM;
          decs=iDecS;

CorreccionSencillaDec(&decg,&decm,&decs,TSH,TSM,TSS,CorrAZ,CorrAL);

//ecu2hor(atoi(ARH),atoi(ARM),atof(ARS),atoi(DecG),atoi(DecM),atof(DecS
),TSH,TSM,TSS,&AZ,&AL);

//hor2ecu(AZ,AL,CorrAZ,CorrAL,TSH,TSM,TSS,&ARHi,&ARMi,&ARSi,&DCGi,&DCMi
,&DCSi);
          printf("Cordenadas con correccion:\n");
          sprintf(mensaje,"AR %d %d %0.1f DEC %d %d %0.1f
ACT\n",iARH,abs(iARM),iARS,decg,decm,decs);
          printf("AR %d %d %0.1f DEC %d %d
%0.1f\n",iARH,abs(iARM),iARS,decg,decm,decs);
      }

      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Nueva posicion de SOL: AR %d %d %0.1f
DEC %d %d %0.1f\n",iARH,abs(iARM),iARS,decg,decm,decs);
          agrega_a_bitacora(buffbitacora);
      }

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      TagSol=gtk_timeout_add( TemporizadorSol,SigueSol,datos );
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Siguiendo al sol...");
```

```
}

void on_BotonGuiadoSol_toggled(GtkToggleButton *button,gpointer
user_data)
{
      gpointer datos;
      if(gtk_toggle_button_get_active(button))
      {
          printf("Guiando el sol...\n");
          TagSol=gtk_timeout_add( TemporizadorSol,SigueSol,datos );

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Guiando al sol");

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Guiando al sol");
            agrega_a_bitacora(buffbitacora);
          }
      }
      else
      {
          printf("Deje de guiar al sol...\n");
            gtk_timeout_remove( TagSol );

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Guiando al sol CANCELADO");

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Deje de guial al sol");
            agrega_a_bitacora(buffbitacora);
          }
      }
}

void on_ToggleBarridoSol_toggled(GtkToggleButton *button,gpointer
user_data)
{
      gpointer datos;
      if(gtk_toggle_button_get_active(button))
      {
          printf("Haciendo barrido...\n");

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Haciendo barrido");
      }
      else
      {
          printf("Cancele barrido...\n");

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Cancele barrido");
      }
}
```

```
/******************Función de Paro del RT5 por software elaborado por: Berenice Rodríguez Pedroza y colaboración de Yarai Elizabeth Tlatelpa Osorio. Por el momento esta función de paro por software sólo detiene el telescopio en movimiento, si el apuntado del telescopio se está realizando introduciendo coordenadas deseadas. Actualmente se está trabajando para que también funcione al estar apuntando usando la paleta de movimiento fino*****************************************/

gboolean on_BotonParo_clicked(GtkWidget *widget, GdkEvent *event,
gpointer data)
{
        BeniSocket Socket;
      char mensaje[200],dir_NS[10],dir_EO[10];
      strcpy(mensaje,"TEL");
      unsigned short port=4955;
      char hostname[]="192.168.0.205";
      char caracteres[200];
        char ARH[10],ARM[10],ARS[10],DecG[10],DecM[10],DecS[10];
      gint
ArH1,ArM1,DecG1,DecM1,DecS1,ArM_d,ArH_d,ArS_d,DecG_d,DecM_d,DecS_d;
      float ArS1,res_G,res_M,res_aG,res_aM;
      char Epoca[10], DeCG[10];
        int decg,decm, Mov_EO, Mov_NS;
      double decs;




      int init=initBeniSocketClient(&Socket,hostname,port);


        printf("Se está deteniendo el telescopio...");
        if(init==0)
      {
        Socket.rc=write(Socket.socket,mensaje,10);
        if(Socket.rc<0)
        {
              g_print("Error escritura del socket\n");
                  strcpy(mensaje,"EPOCA 2012.2 AR 0 0 0 DEC 0 0 0
ACT\n"); // Condicion inicial proteccion
        }
        else
        {
          read(Socket.socket,caracteres,200);

          //printf("%s\n",caracteres);

        strcpy(ARH,gtk_entry_get_text(GTK_ENTRY(EntryARHora)));
      if(strcmp(ARH,"")==0) strcpy(ARH,"0");
      strcpy(ARM,gtk_entry_get_text(GTK_ENTRY(EntryARMinuto)));
      if(strcmp(ARM,"")==0) strcpy(ARM,"0");
      strcpy(ARS,gtk_entry_get_text(GTK_ENTRY(EntryARSeg)));
      if(strcmp(ARS,"")==0) strcpy(ARS,"0");
      strcpy(DecG,gtk_entry_get_text(GTK_ENTRY(EntryDecGrado)));
      if(strcmp(DecG,"")==0) strcpy(DecG,"0");
```

```
      strcpy(DecM,gtk_entry_get_text(GTK_ENTRY(EntryDecMinuto)));
      if(strcmp(DecM,"")==0) strcpy(DecM,"0");
      strcpy(DecS,gtk_entry_get_text(GTK_ENTRY(EntryDecSeg)));
      if(strcmp(DecS,"")==0) strcpy(DecS,"0");


              //Servidor
            strcpy(caracteres,"AR 10:08:00 DEC 20:01:10234240 ACT \n");

            sscanf(caracteres,"AR %d:%d:%f",&ArH1,&ArM1,&ArS1);
          strcpy(caracteres,strstr(caracteres,"DEC "));
          sscanf(caracteres,"DEC %d:%d:%02d",&DecG1,&DecM1,&DecS1);




             //Conversión de string a int
            ArM_d=atoi(ARM);
            DecG_d=atoi(DecG);
            ArH_d=atoi(ARH);
            ArS_d=atoi(ARS);
            DecS_d=atoi(DecS);
            DecM_d=atoi(DecM);

            printf("\n servidor: AR %d %d %f DEC %d %d %d
\n",ArH1,ArM1,ArS1,DecG1,DecM1,DecS1);
            printf("\n deseada: AR %d %d %d DEC %d %d %d
\n",ArH_d,ArM_d,ArS_d,DecG_d,DecM_d,DecS_d);

          //Determinar dirección
          res_G=DecG_d-DecG1;
          res_M=DecM_d-DecM1;
          res_aG=ArH_d-ArH1;
          res_aM=ArM_d-ArM1;

           //calculos en asención
           ArH1= abs(ArH_d-ArH1);
           ArM1= abs(ArM_d-ArM1);
           ArS1= abs(ArS_d-ArS1);
           Mov_EO=ArH1*(15*3600)+ArM1*(0.25*3600)+ArS1*(3600/240);
//Delta de compensación 3600''=1°
           if (Mov_EO>=3600)
           Mov_EO=Mov_EO-3600;


           if(res_aG<=0)
            {
                  if(res_aG==0)
              {
                     if(res_aM<=0)
                     strcpy(dir_EO,"+A");  //este

                     else
                         strcpy(dir_EO,"-A");   //oeste
                  }
            else
                       strcpy(dir_EO,"+A");
              }
```

```
            else
               strcpy(dir_EO,"-A");

          DecG1= abs(DecG_d-DecG1);
          DecM1= abs(DecM_d-DecM1);
          DecS1= abs(DecS_d-DecS1);
          Mov_NS=DecG1*(3600)+DecM1*(60)+DecS1; //Delta de compensación
3600'' = 1°
          if (Mov_NS>=3600)
          Mov_NS=Mov_NS-3600;


            printf("diferencia AR %d %d %f DEC %d %d %d
\n",ArH1,ArM1,ArS1,DecG1,DecM1,DecS1);
            printf("compensación Mov_Ns %d Mov_Eo %d
\n",Mov_NS,Mov_EO);



          if (res_G <= 0)
           {
                 if (res_G==0)
              {
                    if (res_M<=0)
                  strcpy(dir_NS,"+D");
                    else
                         strcpy(dir_NS,"-D");
                  }
           else
                     strcpy(dir_NS,"+D");
             }
             else
               strcpy(dir_NS,"-D");




             }
           }
          else
            printf("Error en la comunicación...no se puede utilizar el
botón paro");




            sprintf(mensaje,"AROF %d DECOF %d %s
%s",Mov_EO,Mov_NS,dir_EO,dir_NS);
            printf(" %s \n", mensaje);
            printf("%s  %s \n",dir_EO,dir_NS);
         MandaYLeeSocket(mensaje,200,caracteres);
```

```
}

        //**********************Boton Apply**********************//
gboolean on_BotonEnvia_clicked(GtkWidget *widget, GdkEvent *event,
gpointer data)
{
      BeniSocket Socket;
      char caracteres[200];
      char mensaje[100];
      char mensaje2[100];
      char ARH[10],ARM[10],ARS[10],DecG[10],DecM[10],DecS[10],Epoca[10];
      double AZ,AL;
      int ARHi,ARMi,ARSi,DCGi,DCMi,DCSi;
      int decg,decm;
      double decs;
      gint TSH,TSM;
      float TSS;
      double CorrAZ,CorrAL;

      strcpy(ARH,gtk_entry_get_text(GTK_ENTRY(EntryARHora)));
      if(strcmp(ARH,"")==0) strcpy(ARH,"0");
      strcpy(ARM,gtk_entry_get_text(GTK_ENTRY(EntryARMinuto)));
      if(strcmp(ARM,"")==0) strcpy(ARM,"0");
      strcpy(ARS,gtk_entry_get_text(GTK_ENTRY(EntryARSeg)));
      if(strcmp(ARS,"")==0) strcpy(ARS,"0");
      strcpy(DecG,gtk_entry_get_text(GTK_ENTRY(EntryDecGrado)));
      if(strcmp(DecG,"")==0) strcpy(DecG,"0");
      strcpy(DecM,gtk_entry_get_text(GTK_ENTRY(EntryDecMinuto)));
      if(strcmp(DecM,"")==0) strcpy(DecM,"0");
      strcpy(DecS,gtk_entry_get_text(GTK_ENTRY(EntryDecSeg)));
      if(strcmp(DecS,"")==0) strcpy(DecS,"0");
      strcpy(Epoca,gtk_entry_get_text(GTK_ENTRY(EntryEpoca)));
      if(strcmp(Epoca,"")==0) strcpy(DecG,"2010.2");
      sprintf(mensaje,"EPOCA %s AR %s %s %s DEC %s %s %s
ACT\n",Epoca,ARH,ARM,ARS,DecG,DecM,DecS);
      printf("Enviando a:EPOCA %s AR %s %s %s DEC %s %s
%s\n",Epoca,ARH,ARM,ARS,DecG,DecM,DecS);
        printf(" %s",mensaje);

      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Nuevas coordenadas AR %s %s %s DEC %s %s
%s",Epoca,ARH,ARM,ARS,DecG,DecM,DecS);
          agrega_a_bitacora(buffbitacora);
      }

      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleActivarCor
reccion)))
      {
          //Obtiene el tiempo sideral.
```

```
sscanf(gtk_label_get_text(GTK_LABEL(LabelAH)),"%d:%d:%f",&TSH,&TSM,&TSS
);

CorrAZ=atof(gtk_entry_get_text(GTK_ENTRY(EntradaCorreccionAcimut)));

CorrAL=atof(gtk_entry_get_text(GTK_ENTRY(EntradaCorreccionAltura)));

          //int CorreccionSencillaDec(int *DCG,int *DCM,double *DCS,int
HSH,int HSM,double HSS,double CorrAZ)
          decg=atoi(DecG);
          decm=atoi(DecM);
          decs=atof(DecS);

CorreccionSencillaDec(&decg,&decm,&decs,TSH,TSM,TSS,CorrAZ,CorrAL);

//ecu2hor(atoi(ARH),atoi(ARM),atof(ARS),atoi(DecG),atoi(DecM),atof(DecS
),TSH,TSM,TSS,&AZ,&AL);

//hor2ecu(AZ,AL,CorrAZ,CorrAL,TSH,TSM,TSS,&ARHi,&ARMi,&ARSi,&DCGi,&DCMi
,&DCSi);
          printf("Cordenadas con correccion:\n");
          sprintf(mensaje,"AR %s %s %s DEC %d %d %0.1f
ACT\n",ARH,ARM,ARS,decg,decm,decs);
          printf("AR %s %s %s DEC %d %d
%0.1f\n",ARH,ARM,ARS,decg,decm,decs);

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Coordenadas corregidas AR %s %s %s DEC
%d %d %0.1f",ARH,ARM,ARS,decg,decm,decs);
            agrega_a_bitacora(buffbitacora);
          }
      }

      if(MandaYLeeSocket(mensaje,200,caracteres)<0)
        printf("Error en socket...\n");
      //printf("%s\n",caracteres);
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Moviendo Telescopio y activando guiado");
}

void
destroy(GtkWidget *widget, gpointer data)
{

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
    {
      sprintf(buffbitacora,"Sali del programa");
      agrega_a_bitacora(buffbitacora);
    }
    printf("Adios vuelve pronto....\n");
    gtk_main_quit();
}

//Boton cambiar fecha y hora
```

```
void
on_BotonCorrigeHora_clicked(GtkButton *button,gpointer user_data)
{
      time_t rawtime;
      struct tm * ptm;


    time ( &rawtime );

    ptm = localtime ( &rawtime ); //tambien localtime



    printf ("Mandando Hora %04d/%02d/%02d %02d:%02d:%02d a computadora
de control\n", ptm->tm_year+1900,ptm->tm_mon+1,ptm->tm_mday,ptm-
>tm_hour, ptm->tm_min,ptm->tm_sec);


      BeniSocket Socket;
      char mensaje[100];
      sprintf(mensaje,"TL %02d %02d %02d FECHA %02d %02d %04d",ptm-
>tm_hour, ptm->tm_min,ptm->tm_sec,ptm->tm_mday,ptm->tm_mon+1,ptm-
>tm_year+1900);
      unsigned short port=4955;
      char hostname[]="192.168.0.205";
      char caracteres[200];

      initBeniSocketClient(&Socket,hostname,port);

      Socket.rc=write(Socket.socket,mensaje,100);
      if(Socket.rc<0)
      {
            g_print("Error escritura del socket\n");
      }
      else
      {

        read(Socket.socket,caracteres,200);

        //printf("%s\n",caracteres);

        close(Socket.socket);
      }
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Correccion de hora en computadora de
control %s",mensaje);
          agrega_a_bitacora(buffbitacora);
      }

      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Hora corregida en la computadora de control");
}

void on_BotonInicializar_clicked(GtkButton *button,gpointer user_data)
{
      gpointer datos;
```

```
	char mensaje[500];
	char caracteres[500];
	gtk_timeout_remove(TagActualiza);
	sprintf(mensaje,"CONTRASENA 641024");
	printf("Enviando comandos de inicializacion\n");
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);
	usleep(50000);
	sprintf(mensaje,"PPSAR 6.4 PPSDEC 9.2444444");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(100000);
	sprintf(mensaje,"KP_AR 8 KI_AR .0001 KD_AR 30");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(50000);
	//sprintf(mensaje,"KP_DEC 5 KI_DEC .0001 KD_DEC 10"); estas son
las que le dejo chava
	sprintf(mensaje,"KP_DEC 3 KI_DEC .0001 KD_DEC 10");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(50000);
	sprintf(mensaje,"VELGUIA 0.096263013698");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(50000);
	sprintf(mensaje,"PID_AR? PID_DEC? HRD?");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	printf("%s\n",caracteres);

	usleep(50000);
	sprintf(mensaje,"DECMAX 80.0 DECMIN -60.0");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(50000);
	sprintf(mensaje,"CONTRASENA 641024");
	//printf("Mensaje: %s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	//printf("%s\n",caracteres);

	usleep(50000);
	TagActualiza=gtk_timeout_add(
TemporizadorActualiza,ActualizaCoordenadas,datos );
	sprintf(mensaje,"LISTO");
	MandaYLeeSocket(mensaje,200,caracteres);
	printf("Termine la inicializacion con exito...\n");
```

```
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Telescopio inicializado");
          agrega_a_bitacora(buffbitacora);
      }

      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Telescopio inicializado");
}

void on_BotonCenit_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[100];
      char caracteres[500];
      sprintf(mensaje,"CENIT");
      printf("Enviando al cenit\n");

      MandaYLeeSocket(mensaje,200,caracteres);
      gtk_widget_set_sensitive(BotonPonerBuscador,FALSE);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Telescopio al cenit");
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Telescopio a Cenit");
}

void on_BotonAcercaDe_clicked(GtkButton *button,gpointer user_data)
{
  GdkPixbuf *Logotipo;

  Logotipo=gdk_pixbuf_new_from_file("LogoControRT5_2.png",NULL);

  gchar *authors[] = { "","Benito Orozco Serna <orozco@inaoep.mx>","",
                  "Agradezco la valiosa colaboracion de:","",
                  "Eduardo Mendoza <mend@inaoep.mx>",
                  "Victor H. De La Luz",
                  "Guillermo Herrera","",
                  "Agradezco de antemano cualquier comentario o
critica","",
                  "Gracias por utilizar ControlRT5",NULL };
       gtk_show_about_dialog (NULL,
                         "program-name", "ControlRT5",
                   "version","V0.1",
                        "copyright","Copyright Benito Orozco @2010",
                        "website","http://www.inaoep.mx/~rt5",
                        "logo",Logotipo,
                   "authors",authors,
                         "title", "Acerca de ControlRT5",
                         NULL);
    gdk_pixbuf_unref(Logotipo);
}
```

```
void on_BotonPonerReceptor_clicked(GtkButton *button,gpointer
user_data)
{
      char mensaje[100];
      sprintf(mensaje,"FIJODEC -56 0 0 AH 0 0 0");
      printf("Mandando a posicion de poner receptor...\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      gtk_widget_set_sensitive(BotonPonerBuscador,TRUE);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Telescopio a posicion de poner
receptor");
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Telescopio a poner Receptor");
}

void on_BotonPonerBuscadorChico_clicked(GtkButton *button,gpointer
user_data)
{
      char mensaje[100];
      char caracteres[500];

      sprintf(mensaje,"FIJODEC -45 0 0 AH -3 30 0");
      printf("Mandando a posicion de poner buscador  chico...\n");

      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Telescopio a posicion de poner Buscador
chico");
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Telescopio a poner Buscador");

}

void on_BotonFijoDec_clicked(GtkButton *button,gpointer user_data)
{
      BeniSocket Socket;
      char caracteres[200];
      char mensaje[100];
      char DecG[10],DecM[10],DecS[10];

      strcpy(DecG,gtk_entry_get_text(GTK_ENTRY(EntryDecFijoG)));
      if(strcmp(DecG,"")==0) strcpy(DecG,"0");
```

```
      strcpy(DecM,gtk_entry_get_text(GTK_ENTRY(EntryDecFijoM)));
      if(strcmp(DecM,"")==0) strcpy(DecM,"0");
      strcpy(DecS,gtk_entry_get_text(GTK_ENTRY(EntryDecFijoS)));
      if(strcmp(DecS,"")==0) strcpy(DecS,"0");
      sprintf(mensaje,"FIJODEC %s %s %s",DecG,DecM,DecS);
      //printf("%s\n",mensaje);

      if(MandaYLeeSocket(mensaje,200,caracteres)<0)
        printf("Error en socket...\n");
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Movimiento Fijo en Declinacion
%s",mensaje);
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento Fijo en DEC");
      //printf("%s\n",caracteres);
}

void on_PresionaTecla(GtkWidget   *widget,GdkEventKey *event,gpointer
user_data)
{
    printf("Presiono una tecla...\n");
}

void on_BotonFijoAH_clicked(GtkButton *button,gpointer user_data)
{
      BeniSocket Socket;
      char caracteres[200];
      char mensaje[100];
      char AHH[10],AHM[10],AHS[10];

      //printf("1\n");
      strcpy(AHH,gtk_entry_get_text(GTK_ENTRY(EntryAHFijoH)));
      if(strcmp(AHH,"")==0) strcpy(AHH,"0");
      //printf("2\n");
      strcpy(AHM,gtk_entry_get_text(GTK_ENTRY(EntryAHFijoM)));
      if(strcmp(AHM,"")==0) strcpy(AHM,"0");
      //printf("3\n");
      strcpy(AHS,gtk_entry_get_text(GTK_ENTRY(EntryAHFijoS)));
      if(strcmp(AHS,"")==0) strcpy(AHS,"0");
      sprintf(mensaje,"AH %s %s %s",AHH,AHM,AHS);
      printf("Enviando a posicion fija AH %s %s %s\n",AHH,AHM,AHS);

      if(MandaYLeeSocket(mensaje,200,caracteres)<0)
        printf("Error en socket...\n");
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Movimiento fijo en Angulo horario
%s",mensaje);
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento Fijo en AH");
      //printf("%s\n",caracteres);
```

```
}

void on_BotonCorrigeCenit_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[100];
      sprintf(mensaje,"LISTO\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Cenit corregido");
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Corrigio cenit");
      //printf("%s\n",caracteres);
      printf("Cenit Corregido\n");
}

void on_BotonCorrigeCoordenadas_clicked(GtkButton *button,gpointer
user_data)
{
      char mensaje[100];
      sprintf(mensaje,"CORR");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Coordenadas corregidas");
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Corrigio coordenadas");
      printf("Coordenadas corregidas\n");

}

void on_BotonPruebaComm_clicked(GtkButton *button,gpointer user_data)
{
      BeniSocket Socket;
      gpointer datos;
      char mensaje[20];
      strcpy(mensaje,"TEL");
      unsigned short port=4955;
      char hostname[]="192.168.0.205";
      char caracteres[200];
      gint
ArH,ArM,DecG,DecM,DecS,AHH,AHM,TSH,TSM,basura1,basura2,TLH,TLM,TUH,TUM;
      float ArS,AHS,TSS,basura3,TLS,TUS;
      char AR[50],DEC[50],AH[50],TS[50],TL[50],TU[50],Fecha[50];
```

```
      gtk_timeout_remove(TagActualiza);
      printf("Probando comunicacion....\n");
      int init=initBeniSocketClient(&Socket,hostname,port);

      if(init==0)
      {
        Socket.rc=write(Socket.socket,mensaje,10);
        if(Socket.rc<0)
        {
              g_print("Error escritura del socket\n");
        }
        else
        {
          read(Socket.socket,caracteres,200);

          //printf("%s\n",caracteres);

          sscanf(caracteres,"AR %d:%d:%f",&ArH,&ArM,&ArS);
          strcpy(caracteres,strstr(caracteres,"DEC "));
          sscanf(caracteres,"DEC %d %d'%d",&DecG,&DecM,&DecS);
          strcpy(caracteres,strstr(caracteres,"AH "));
          sscanf(caracteres,"AH %d:%d:%f",&AHH,&AHM,&AHS);
          strcpy(caracteres,strstr(caracteres,"TS: "));
          sscanf(caracteres,"TS: %d:%d:%f",&TSH,&TSM,&TSS);
          strcpy(caracteres,strstr(caracteres,"TL: "));
          sscanf(caracteres,"TL: %d:%d:%f",&TLH,&TLM,&TLS);
          strcpy(caracteres,strstr(caracteres,"TU: "));
          sscanf(caracteres,"TU: %d:%d:%f %s",&TUH,&TUM,&TUS,Fecha);

          sprintf(AR,"%02d:%02d:%0.1f\n",ArH,ArM,ArS);
          gtk_label_set_text(GTK_LABEL (LabelAR),AR);
          sprintf(DEC,"%02d %02d %02d\n",DecG,DecM,DecS);
          gtk_label_set_text(GTK_LABEL (LabelDec),DEC);
          sprintf(AH,"%02d:%02d:%0.1f\n",AHH,AHM,AHS);
          gtk_label_set_text(GTK_LABEL (LabelAH),AH);

          //AR = g_markup_printf_escaped ("<span color=\"blue\"
style=\"italic\" font=\"Century Schoolbook L
30\">%02d:%02d:%0.1f</span>",ArH,ArM,ArS);
          //gtk_label_set_markup (GTK_LABEL (LabelAR), AR);
          //DEC = g_markup_printf_escaped ("<span color=\"red\"
style=\"italic\" font=\"Century Schoolbook L 30\">%02d %02d
%02d</span>",DecG,DecM,DecS);
          //gtk_label_set_markup (GTK_LABEL (LabelDec), DEC);
          //AH = g_markup_printf_escaped ("<span color=\"blue\"
style=\"italic\" font=\"Century Schoolbook L
16\">%02d:%02d:%0.1f</span>",AHH,AHM,AHS);
          //gtk_label_set_markup (GTK_LABEL (LabelAH), AH);
          //g_free (AR);
          //g_free (DEC);
          //g_free (AH);

          //sprintf(AR,"%02d:%02d:%0.1f\n",ArH,ArM,ArS);
          //gtk_label_set_text(GTK_LABEL(LabelAR),AR);
          //sprintf(DEC,"%02d %02d %02d\n",DecG,DecM,DecS);
          //gtk_label_set_text(GTK_LABEL(LabelDec),DEC);
          //sprintf(AH,"%02d:%02d:%0.1f\n",AHH,AHM,AHS);
          //gtk_label_set_text(GTK_LABEL(LabelAH),AH);
```

```
            sprintf(TS,"%02d:%02d:%0.1f",TSH,TSM,TSS);
            gtk_label_set_text(GTK_LABEL(LabelTS),TS);
            sprintf(TL,"%02d:%02d:%0.1f",TLH,TLM,TLS);
            gtk_label_set_text(GTK_LABEL(LabelTL),TL);
            sprintf(TU,"%02d:%02d:%0.1f",TUH,TUM,TUS);
            gtk_label_set_text(GTK_LABEL(LabelTU),TU);
            gtk_label_set_text(GTK_LABEL(LabelFecha),Fecha);
            close(Socket.socket);

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Comunicacion OK");
          }
      }
      else

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Error comunicacion con control del telescopio");
      TagActualiza=gtk_timeout_add(
TemporizadorActualiza,ActualizaCoordenadas,datos ); //Repite la funcion
        //periodicamente despues de un intervalo =
TemporizadorActualiza

}

void on_BotonGuiado_toggled(GtkToggleButton *button,gpointer user_data)
{
      char mensaje[100];
      if(gtk_toggle_button_get_active(button))
      {
          sprintf(mensaje,"PON_GUIADO");

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Telescopio Guiando");
          printf("Telescopio Guiando...\n");

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Guiado del telescopio activado");
            agrega_a_bitacora(buffbitacora);
          }
      }
      else
      {
          sprintf(mensaje,"QUITA_GUIADO");

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Guiado del telescopio desactivado");
            agrega_a_bitacora(buffbitacora);
          }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Telescopio Sin Guiado");
          printf("Telescopio NO Guiando\n");
      }
```

```
	char caracteres[200];
	//MandaYLeeSocket(mensaje,200,caracteres);

	//printf("%s\n",caracteres);
}

void on_BotonAbrirLazo_clicked(GtkButton *button,gpointer user_data)
{
	gpointer datos;
	char mensaje[100];
	sprintf(mensaje,"CONTRASENA 641024");
	//printf("%s\n",mensaje);
	usleep(50);

	char caracteres[200];
	MandaYLeeSocket(mensaje,200,caracteres);

	sprintf(mensaje,"DA_AR 0");
	//printf("%s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	usleep(50);

	sprintf(mensaje,"DA_DEC 0");
	//printf("%s\n",mensaje);
	MandaYLeeSocket(mensaje,200,caracteres);
	printf("Telescopio en lazo abierto\n");
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
	{
	    sprintf(buffbitacora,"Telescopio en lazo abierto");
	    agrega_a_bitacora(buffbitacora);
	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Telescopio en Lazo Abierto, lo puedes mover con la mano");
}

void on_BotonNorte_clicked(GtkButton *button,gpointer user_data)
{
	char mensaje[100];
	sprintf(mensaje,"DECOF %s
+D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
	printf("+D\n");

	char caracteres[200];
	MandaYLeeSocket(mensaje,200,caracteres);

	//printf("%s\n",caracteres);
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
	{
	    sprintf(buffbitacora,"Paleta Norte %s
segundos",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
	    agrega_a_bitacora(buffbitacora);
	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta NORTE");
}
```

```
void on_BotonNorte_key_press_event(GtkButton *button,GdkEventKey
*event, gpointer func_data)
{
      char mensaje[100];

      switch(event->keyval)
      {
          case GDK_Up:
          case 'w':
          case 'W':
          {
            sprintf(mensaje,"DECOF %s
+D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Norte\n");
            break;
          }
          case GDK_Down:
          case 'x':
          case 'X':
          {
            sprintf(mensaje,"DECOF %s -
D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Sur\n");
            break;
          }
          case GDK_Left:
          case 'a':
          case 'A':
          {
            sprintf(mensaje,"AROF %s
+A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Este\n");
            break;
          }
          case GDK_Right:
          case 'd':
          case 'D':
          {
            sprintf(mensaje,"AROF %s -
A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Oeste\n");
            break;
          }
          default:
          {
            printf("Tecla sin funcion, lo siento.\n");
            break;
          }
      }
      //gtk_window_set_focus(button, NULL);
      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);
      //printf("%s\n",caracteres);
}

static gboolean
snooper (GtkWidget *widget, GdkEventKey *event, gpointer data)
{
```

```
      char mensaje[100];

      switch(event->keyval)
      {
          case GDK_Up:
          case 'w':
          case 'W':
          {
            sprintf(mensaje,"DECOF %s
+D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Norte\n");
            break;
          }
          case GDK_Down:
          case 'x':
          case 'X':
          {
            sprintf(mensaje,"DECOF %s -
D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Sur\n");
            break;
          }
          case GDK_Left:
          case 'a':
          case 'A':
          {
            sprintf(mensaje,"AROF %s
+A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Este\n");
            break;
          }
          case GDK_Right:
          case 'd':
          case 'D':
          {
            sprintf(mensaje,"AROF %s -
A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
            printf("Oeste\n");
            break;
          }
          default:
          {
            printf("Tecla sin funcion, lo siento.\n");
            break;
          }
      }
      //gtk_window_set_focus(button, NULL);
      //char caracteres[200];
      //MandaYLeeSocket(mensaje,200,caracteres);
      //printf("%s\n",caracteres);
  return FALSE; /* pass event further */
}

void on_EntryOffset_focus_in_event(GtkWidget *widget,GdkEventKey
*event, gpointer func_data)
{
      //gtk_key_snooper_remove (manejador_snooper);
```

```
      //manejador_snooper=gtk_key_snooper_install
((GtkKeySnoopFunc)snooper, 0);
      //printf("entry offset tiene foco\n");
}

static gboolean on_EntryOffset_focus_out_event(GtkWidget
*widget,GdkEventKey *event, gpointer func_data)
{
      //gtk_key_snooper_remove (manejador_snooper);
      //printf("entry offset perdio el foco\n");
      return FALSE;
}

void on_BotonSur_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[100];
      sprintf(mensaje,"DECOF %s -
D",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
      printf("-D\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Paleta Sur %s
segundos",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta SUR");
}

void on_BotonEste_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[100];
      sprintf(mensaje,"AROF %s
+A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
      printf("+A\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Paleta Este %s
segundos",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
          agrega_a_bitacora(buffbitacora);
      }
      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta ESTE");
}

void on_BotonOeste_clicked(GtkButton *button,gpointer user_data)
```

```
{
      char mensaje[100];
      sprintf(mensaje,"AROF %s -
A",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
      printf("-A\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))

      {
          sprintf(buffbitacora,"Paleta Oeste %s
segundos",gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
          agrega_a_bitacora(buffbitacora);
      }

      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta OESTE");

}

void on_BotonNE_clicked(GtkButton *button,gpointer user_data)

{
      char mensaje[100];
      char temp[20];


      //printf("%s\n",caracteres);
      if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
      {
          sprintf(buffbitacora,"Paleta Norte %s segundos\nPaleta Este
%s",gtk_entry_get_text(GTK_ENTRY(EntryOffset)),gtk_entry_get_text(GTK_E
NTRY(EntryOffset)));
          agrega_a_bitacora(buffbitacora);
      }

      gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta NORESTE");
}

void on_BotonNW_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[100];
      char temp[20];
      strcpy(temp,gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
      sprintf(mensaje,"AROF %s DECOF %s -A +D",temp,temp);
      printf("-A +D\n");

      char caracteres[200];
      MandaYLeeSocket(mensaje,200,caracteres);

      //printf("%s\n",caracteres);
```

```
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
	{
	    sprintf(buffbitacora,"Paleta Norte %s segundos\nPaleta Oeste
%s",gtk_entry_get_text(GTK_ENTRY(EntryOffset)),gtk_entry_get_text(GTK_E
NTRY(EntryOffset)));
	    agrega_a_bitacora(buffbitacora);

	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta NOROESTE");

}

void on_BotonSE_clicked(GtkButton *button,gpointer user_data)

{
	char mensaje[100];
	char temp[20];


	//printf("%s\n",caracteres);
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
	{
	    sprintf(buffbitacora,"Paleta Sur %s segundos\nPaleta Este
%s",gtk_entry_get_text(GTK_ENTRY(EntryOffset)),gtk_entry_get_text(GTK_E
NTRY(EntryOffset)));
	    agrega_a_bitacora(buffbitacora);

	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta SURESTE");
}

void on_BotonSW_clicked(GtkButton *button,gpointer user_data)
{
	char mensaje[100];
	char temp[20];
	strcpy(temp,gtk_entry_get_text(GTK_ENTRY(EntryOffset)));
	sprintf(mensaje,"AROF %s DECOF %s -A -D",temp,temp);
	printf("-A -D\n");

	char caracteres[200];
	MandaYLeeSocket(mensaje,200,caracteres);

	//printf("%s\n",caracteres);
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))

	{
	    sprintf(buffbitacora,"Paleta Sur %s segundos\nPaleta Oeste
%s",gtk_entry_get_text(GTK_ENTRY(EntryOffset)),gtk_entry_get_text(GTK_E
NTRY(EntryOffset)));
	    agrega_a_bitacora(buffbitacora);
	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Movimiento paleta SUROESTE");
```

```
}

double CurrentDaynum()
{
      /* Read the system clock and return the number
         of days since 31Dec79 00:00:00 UTC (daynum 0) */

      int x;
      struct timeval tptr;
      double usecs, seconds;

      x=gettimeofday(&tptr,NULL);

      usecs=0.000001*(double)tptr.tv_usec;
      seconds=usecs+(double)tptr.tv_sec;

      return ((seconds/86400.0)-3651.0);
}

double FixAngle(double x)
{
      /* This function reduces angles greater than
         two pi by subtracting two pi from the angle */

      while (x>twopi)
            x-=twopi;

      return x;
}

double PrimeAngle(double x)
{
      /* This function is used in the FindMoon() function. */

      x=x-360.0*floor(x/360.0);
      return x;
}


void FindMoon(double daynum)
{
      /* This function determines the position of the moon, including
         the azimuth and elevation headings, relative to the latitude
         and longitude of the tracking station.  This code was derived
         from a Javascript implementation of the Meeus method for
         determining the exact position of the Moon found at:
         http://www.geocities.com/s_perona/ingles/poslun.htm. */

      double     jd, ss, t, t1, t2, t3, d, ff, l1, m, m1, ex, om, l,
            b, w1, w2, bt, p, lm, h, ra, dec, z, ob, n, e, el,
            az, teg, th, mm, dv;

      jd=daynum+2444238.5;

      t=(jd-2415020.0)/36525.0;
      t2=t*t;
      t3=t2*t;
      l1=270.434164+481267.8831*t-0.001133*t2+0.0000019*t3;
```

```
      ff=11.250889+483202.0251*t-0.003211*t2-0.0000003*t3;
      om=259.183275-1934.142*t+0.002078*t2+0.0000022*t3;
      om=om*deg2rad;

      /* Additive terms */

      l1=l1+0.000233*sin((51.2+20.2*t)*deg2rad);
      ss=0.003964*sin((346.56+132.87*t-0.0091731*t2)*deg2rad);
      l1=l1+ss+0.001964*sin(om);
      m=m-0.001778*sin((51.2+20.2*t)*deg2rad);
      m1=m1+0.000817*sin((51.2+20.2*t)*deg2rad);
      m1=m1+ss+0.002541*sin(om);
      d=d+0.002011*sin((51.2+20.2*t)*deg2rad);
      d=d+ss+0.001964*sin(om);
      ff=ff+ss-0.024691*sin(om);
      ff=ff-0.004328*sin(om+(275.05-2.3*t)*deg2rad);
      ex=1.0-0.002495*t-0.00000752*t2;
      om=om*deg2rad;

      l1=PrimeAngle(l1);
      m=PrimeAngle(m);
      m1=PrimeAngle(m1);
      d=PrimeAngle(d);
      ff=PrimeAngle(ff);
      om=PrimeAngle(om);

      m=m*deg2rad;
      m1=m1*deg2rad;
      d=d*deg2rad;
      ff=ff*deg2rad;

      /* Ecliptic Longitude */

      l=l1+6.28875*sin(m1)+1.274018*sin(2.0*d-m1)+0.658309*sin(2.0*d);
      l=l+0.213616*sin(2.0*m1)-ex*0.185596*sin(m)-0.114336*sin(2.0*ff);
      l=l+0.058793*sin(2.0*d-2.0*m1)+ex*0.057212*sin(2.0*d-m-
m1)+0.05332*sin(2.0*d+m1);
      l=l+ex*0.045874*sin(2.0*d-m)+ex*0.041024*sin(m1-m)-
0.034718*sin(d);
      l=l-ex*0.030465*sin(m+m1)+0.015326*sin(2.0*d-2.0*ff)-
0.012528*sin(2.0*ff+m1);

      l=l-0.01098*sin(2.0*ff-m1)+0.010674*sin(4.0*d-
m1)+0.010034*sin(3.0*m1);
      l=l+0.008548*sin(4.0*d-2.0*m1)-ex*0.00791*sin(m-m1+2.0*d)-
ex*0.006783*sin(2.0*d+m);

      l=l+0.005162*sin(m1-d)+ex*0.005*sin(m+d)+ex*0.004049*sin(m1-
m+2.0*d);
      l=l+0.003996*sin(2.0*m1+2.0*d)+0.003862*sin(4.0*d)+0.003665*sin(2.
0*d-3.0*m1);

      l=l+ex*0.002695*sin(2.0*m1-m)+0.002602*sin(m1-2.0*ff-
2.0*d)+ex*0.002396*sin(2.0*d-m-2.0*m1);

      l=l-0.002349*sin(m1+d)+ex*ex*0.002249*sin(2.0*d-2.0*m)-
ex*0.002125*sin(2.0*m1+m);
```

```
      l=l-ex*ex*0.002079*sin(2.0*m)+ex*ex*0.002059*sin(2.0*d-m1-2.0*m)-
0.001773*sin(m1+2.0*d-2.0*ff);

      l=l+ex*0.00122*sin(4.0*d-m-m1)-
0.00111*sin(2.0*m1+2.0*ff)+0.000892*sin(m1-3.0*d);

      l=l-ex*0.000811*sin(m+m1+2.0*d)+ex*0.000761*sin(4.0*d-m-
2.0*m1)+ex*ex*.000717*sin(m1-2.0*m);

      l=l+ex*ex*0.000704*sin(m1-2.0*m-2.0*d)+ex*0.000693*sin(m-
2.0*m1+2.0*d)+ex*0.000598*sin(2.0*d-m-2.0*ff)+0.00055*sin(m1+4.0*d);

      l=l+0.000538*sin(4.0*m1)+ex*0.000521*sin(4.0*d-
m)+0.000486*sin(2.0*m1-d);

      l=l-0.001595*sin(2.0*ff+2.0*d);

      /* Ecliptic latitude */

      b=5.128189*sin(ff)+0.280606*sin(m1+ff)+0.277693*sin(m1-
ff)+0.173238*sin(2.0*d-ff);
      b=b+0.055413*sin(2.0*d+ff-m1)+0.046272*sin(2.0*d-ff-
m1)+0.032573*sin(2.0*d+ff);

      b=b+0.017198*sin(2.0*m1+ff)+9.266999e-03*sin(2.0*d+m1-
ff)+0.008823*sin(2.0*m1-ff);
      b=b+ex*0.008247*sin(2.0*d-m-ff)+0.004323*sin(2.0*d-ff-
2.0*m1)+0.0042*sin(2.0*d+ff+m1);

      b=b+ex*0.003372*sin(ff-m-2.0*d)+ex*0.002472*sin(2.0*d+ff-m-
m1)+ex*0.002222*sin(2.0*d+ff-m);

      b=b+0.002072*sin(2.0*d-ff-m-m1)+ex*0.001877*sin(ff-
m+m1)+0.001828*sin(4.0*d-ff-m1);

      b=b-ex*0.001803*sin(ff+m)-0.00175*sin(3.0*ff)+ex*0.00157*sin(m1-m-
ff)-0.001487*sin(ff+d)-ex*0.001481*sin(ff+m+m1)+ex*0.001417*sin(ff-m-
m1)+ex*0.00135*sin(ff-m)+0.00133*sin(ff-d);

      b=b+0.001106*sin(ff+3.0*m1)+0.00102*sin(4.0*d-
ff)+0.000833*sin(ff+4.0*d-m1);

      b=b+0.000781*sin(m1-3.0*ff)+0.00067*sin(ff+4.0*d-
2.0*m1)+0.000606*sin(2.0*d-3.0*ff);

      b=b+0.000597*sin(2.0*d+2.0*m1-ff)+ex*0.000492*sin(2.0*d+m1-m-
ff)+0.00045*sin(2.0*m1-ff-2.0*d);

      b=b+0.000439*sin(3.0*m1-
ff)+0.000423*sin(ff+2.0*d+2.0*m1)+0.000422*sin(2.0*d-ff-3.0*m1);

      b=b-ex*0.000367*sin(m+ff+2.0*d-m1)-
ex*0.000353*sin(m+ff+2.0*d)+0.000331*sin(ff+4.0*d);

      b=b+ex*0.000317*sin(2.0*d+ff-m+m1)+ex*ex*0.000306*sin(2.0*d-2.0*m-
ff)-0.000283*sin(m1+3.0*ff);
```

```
      w1=0.0004664*cos(om*deg2rad);
      w2=0.0000754*cos((om+275.05-2.3*t)*deg2rad);
      bt=b*(1.0-w1-w2);

      /* Parallax calculations */

      p=0.950724+0.051818*cos(m1)+0.009531*cos(2.0*d-
m1)+0.007843*cos(2.0*d)+0.002824*cos(2.0*m1)+0.000857*cos(2.0*d+m1)+ex*
0.000533*cos(2.0*d-m)+ex*0.000401*cos(2.0*d-m-m1);

      p=p+0.000173*cos(3.0*m1)+0.000167*cos(4.0*d-m1)-
ex*0.000111*cos(m)+0.000103*cos(4.0*d-2.0*m1)-0.000084*cos(2.0*m1-
2.0*d)-ex*0.000083*cos(2.0*d+m)+0.000079*cos(2.0*d+2.0*m1);

      p=p+0.000072*cos(4.0*d)+ex*0.000064*cos(2.0*d-m+m1)-
ex*0.000063*cos(2.0*d+m-m1);

      p=p+ex*0.000041*cos(m+d)+ex*0.000035*cos(2.0*m1-m)-
0.000033*cos(3.0*m1-2.0*d);

      p=p-0.00003*cos(m1+d)-0.000029*cos(2.0*ff-2.0*d)-
ex*0.000029*cos(2.0*m1+m);

      p=p+ex*ex*0.000026*cos(2.0*d-2.0*m)-0.000023*cos(2.0*ff-
2.0*d+m1)+ex*0.000019*cos(4.0*d-m-m1);

      b=bt*deg2rad;
      lm=l*deg2rad;
      moon_dx=3.0/(pi*p);

      /* Semi-diameter calculation */
      /* sem=10800.0*asin(0.272488*p*deg2rad)/pi; */

      /* Convert ecliptic coordinates to equatorial coordinates */

      z=(jd-2415020.5)/365.2422;
      ob=23.452294-(0.46845*z+5.9e-07*z*z)/3600.0;
      ob=ob*deg2rad;
      dec=asin(sin(b)*cos(ob)+cos(b)*sin(ob)*sin(lm));
      ra=acos(cos(b)*cos(lm)/cos(dec));

      //printf("Geocentricas Dec %f, AR %f\n",dec/deg2rad,ra/deg2rad);
      if (lm>pi)
            ra=twopi-ra;

      /* ra = right ascension */
      /* dec = declination */
      //Coordenadas del Rt5 en Tona>
      //Lat = 19 01 59.69 N
      //Lon = 98 18 51.61 O
      //Elev= 2175m sobre el nivel del mar
      n=(19.0+1.0/60.0+59.69/3600.0)*deg2rad;//qth.stnlat*deg2rad;    /*
North latitude of tracking station */
      e=-(98.0+18/60.0+51.61/3600.0)*deg2rad;//-98.20638888888888889;//-
qth.stnlong*deg2rad;  /* East longitude of tracking station */
      //longitud = -98.20638888888888889

      /* Find siderial time in radians */
```

```
      t=(jd-2451545.0)/36525.0;
      teg=280.46061837+360.98564736629*(jd-2451545.0)+(0.000387933*t-
t*t/38710000.0)*t;

      while (teg>360.0)
            teg-=360.0;

      th=FixAngle((teg-qth.stnlong)*deg2rad);
      h=th-ra;

      az=atan2(sin(h),cos(h)*sin(n)-tan(dec)*cos(n))+pi;
      el=asin(sin(n)*sin(dec)+cos(n)*cos(dec)*cos(h));

      moon_az=az/deg2rad;
      moon_el=el/deg2rad;

      /* Radial velocity approximation.  This code was derived
         from "Amateur Radio Software", by John Morris, GM4ANB,
         published by the RSGB in 1985. */

      mm=FixAngle(1.319238+daynum*0.228027135);  /* mean moon position
*/
      t2=0.10976;
      t1=mm+t2*sin(mm);
      dv=0.01255*moon_dx*moon_dx*sin(t1)*(1.0+t2*cos(mm));
      dv=dv*4449.0;
      t1=6378.0;
      t2=384401.0;
      t3=t1*t2*(cos(dec)*cos(n)*sin(h));
      t3=t3/sqrt(t2*t2-t2*t1*sin(el));
      moon_dv=dv+t3*0.0753125;

      moon_dec=dec/deg2rad;
      moon_ra=ra/deg2rad;
      moon_gha=teg-moon_ra;

      if (moon_gha<0.0)
            moon_gha+=360.0;
}

gint SigueLuna(gpointer user_data)
{
    char mensaje[200];
    char temporal[200];
    int iARH,iARM,iDecG,iDecM;
    float iARS,iDecS;
    gpointer datos;

    gtk_timeout_remove( TagLuna );
    //printf("la luna esta chida...\n");

    FindMoon(CurrentDaynum());

    iARH=moon_ra;
    iARM=(moon_ra-iARH)*60.0;
    iARS=((moon_ra-iARH)*60.0 - iARM)*60.0;
    sprintf(temporal,"%02d",iARH);
```

```
    gtk_entry_set_text(GTK_ENTRY(EntryARHora),temporal);
    sprintf(temporal,"%02d",abs(iARM));
    gtk_entry_set_text(GTK_ENTRY(EntryARMinuto),temporal);
    sprintf(temporal,"%0.1f",fabs(iARS));
    gtk_entry_set_text(GTK_ENTRY(EntryARSeg),temporal);

    iDecG=moon_dec;
    iDecM=(moon_dec-iDecG)*60.0;
    iDecS=((moon_dec-iDecG)*60.0 - iDecM)*60.0;

    sprintf(temporal,"%02d",iDecG);
    gtk_entry_set_text(GTK_ENTRY(EntryDecGrado),temporal);
    sprintf(temporal,"%02d",abs(iDecM));
    gtk_entry_set_text(GTK_ENTRY(EntryDecMinuto),temporal);
    sprintf(temporal,"%0.1f",fabs(iDecS));
    gtk_entry_set_text(GTK_ENTRY(EntryDecSeg),temporal);

    sprintf(mensaje,"DEC %d %d %d AR %d %d %0.1f
ACT",iDecG,abs(iDecM),abs((int)iDecS),iARH,abs(iARM),iARS);
    //printf("Mensaje: %s\n",mensaje);

    //char caracteres[200];
    //MandaYLeeSocket(mensaje,200,caracteres);

    //printf("%s\n",caracteres);
    //gtk_timeout_remove( TagSegundo );

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
    {
      sprintf(buffbitacora,"Nuevas coordenadas de luna %s",mensaje);
      agrega_a_bitacora(buffbitacora);
    }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Siguiendo a la Luna, Presiona de nuevo el boton \"Guiado Luna\" para
cancelar");
    TagLuna=gtk_timeout_add( TemporizadorLuna,SigueLuna,datos );

}

void on_BotonLuna_clicked(GtkButton *button,gpointer user_data)
{
    char mensaje[200];
    char temporal[200];
    int iARH,iARM,iDecG,iDecM;
    float iARS,iDecS;

    //printf("la luna esta chida...\n");

    FindMoon(CurrentDaynum());

    iARH=moon_ra;
    iARM=(moon_ra-iARH)*60.0;
    iARS=((moon_ra-iARH)*60.0 - iARM)*60.0;
    sprintf(temporal,"%02d",iARH);
    gtk_entry_set_text(GTK_ENTRY(EntryARHora),temporal);
    sprintf(temporal,"%02d",abs(iARM));
```

```
    gtk_entry_set_text(GTK_ENTRY(EntryARMinuto),temporal);
    sprintf(temporal,"%0.1f",fabs(iARS));
    gtk_entry_set_text(GTK_ENTRY(EntryARSeg),temporal);

    iDecG=moon_dec;
    iDecM=(moon_dec-iDecG)*60.0;
    iDecS=((moon_dec-iDecG)*60.0 - iDecM)*60.0;

    sprintf(temporal,"%02d",iDecG);
    gtk_entry_set_text(GTK_ENTRY(EntryDecGrado),temporal);
    sprintf(temporal,"%02d",abs(iDecM));
    gtk_entry_set_text(GTK_ENTRY(EntryDecMinuto),temporal);
    sprintf(temporal,"%0.1f",fabs(iDecS));
    gtk_entry_set_text(GTK_ENTRY(EntryDecSeg),temporal);

    sprintf(mensaje,"DEC %d %d %d AR %d %d %0.1f
",iDecG,abs(iDecM),abs((int)iDecS),iARH,abs(iARM),iARS);
    //printf("Mensaje: %s\n",mensaje);

    //char caracteres[200];
    //MandaYLeeSocket(mensaje,200,caracteres);

    //printf("%s\n",caracteres);
    //gtk_timeout_remove( TagSegundo );

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
    {
      sprintf(buffbitacora,"Nuevas coordenadas de luna %s",mensaje);
      agrega_a_bitacora(buffbitacora);
    }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Posicion de la luna calculada, para mover telescopio oprime el boton
\"Aplicar\"");

}

void on_BotonGuiadoLuna_toggled(GtkToggleButton *button,gpointer
user_data)
{
      gpointer datos;
      if(gtk_toggle_button_get_active(button))
      {
          printf("Guiando la luna...\n");
          TagLuna=gtk_timeout_add( TemporizadorLuna,SigueLuna,datos );

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Inicio guiado luna");
            agrega_a_bitacora(buffbitacora);
          }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Guiando la luna");
      }
      else
```

```
      {
          printf("Deje de guiar la luna...\n");
            gtk_timeout_remove( TagLuna );

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Finalizo guiado luna");
            agrega_a_bitacora(buffbitacora);
          }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Guiando la luna CANCELADO");
      }
}

void on_BotonSol_clicked(GtkButton *button,gpointer user_data)
{
      char mensaje[200];
      char temporal[200];
      //const float pi=3.141592;
      float anio,mes,dia;
      float
temp,AH,ARr,C,LS,K,L0,M0,M0r,Tt,d,d0,t0,s0,gmst,tu,ts,longitud;
      float ARSol,DECSol;
      time_t rawtime;
      struct tm * ptm;
      int iARH,iARM,iDecG,iDecM;
      float iARS,iDecS;

      time ( &rawtime );
      ptm = gmtime ( &rawtime ); //tambien localtime
      //printf ("hora :   %04d/%02d/%02d %02d:%02d:%02d\n",
      //              ptm->tm_year+1900,ptm->tm_mon+1,ptm->tm_mday,
      //              ptm->tm_hour, ptm->tm_min,ptm->tm_sec);

      tu = (float)(ptm->tm_hour)+(float)(ptm->tm_min)/60.0 +
(float)(ptm->tm_sec)/3600.0;
      //tu=(float)19+(float)(10)/60.0+(float)(55)/3600.0;
      anio=ptm->tm_year+1900.0;
      mes=ptm->tm_mon+1.0;
      dia=ptm->tm_mday;

      d0 = 367.0*anio - (int)(7.0/4.0
*(anio+(int)((mes+9.0)/12.0)))+(int)(275.0*mes/9.0) +dia-730531.5;
      t0 = d0 / 36525.0;
      // tiempo sideral en Greenwich
      s0 = 6.6974 + 2400.0513 * t0;
      gmst = s0 + (366.2422 / 365.2422) * tu;
      // longitud de puebla 98 12 23
      longitud = -98.20638888888888889/15.0 ;
      ts = gmst + longitud;
      ts = (int)(ts) % 24 + ts-(int)ts;

      d = d0 + tu/24.0;
      Tt = d/36525.0;
      temp=280.466+36000.770*Tt;
      L0 = (int)(temp)%360 + temp-(int)temp;
```

```
	temp=357.529+35999.050*Tt;
	M0 = (int)(temp)%360 + temp-(int)temp;
	M0r = M0 * pi/180.0;
	C = (1.915 - 0.005* Tt) *  sin( M0r ) + 0.020 * sin( 2.0*M0r );
	LS = L0 + C;
	K= 23.439 - 0.013 *Tt;
	ARr = atan( tan(LS * pi/180.0)*cos(K* pi/180.0));
	ARSol = ARr *180.0/pi;
	if(ARSol < 0 )
		ARSol = ARSol + 360.0;
	//ARSol=14 % 5;
	DECSol  = 180.0/pi * asin ( sin( ARr  ) * sin( K* pi/180.0 ));

	ARSol = ARSol/15.0; //?, DEC
	AH = ts - ARSol;

	//printf("TS=%f AH=%f AR=%f DEC=%f\n",ts,AH,ARSol,DECSol);

	iARH=ARSol;
	iARM=(ARSol-iARH)*60.0;
	iARS=((ARSol-iARH)*60.0 - iARM)*60.0;
	sprintf(temporal,"%02d",iARH);
	gtk_entry_set_text(GTK_ENTRY(EntryARHora),temporal);
	sprintf(temporal,"%02d",abs(iARM));
	gtk_entry_set_text(GTK_ENTRY(EntryARMinuto),temporal);
	sprintf(temporal,"%0.1f",fabs(iARS));
	gtk_entry_set_text(GTK_ENTRY(EntryARSeg),temporal);

	iDecG=DECSol;
	iDecM=(DECSol-iDecG)*60.0;
	iDecS=((DECSol-iDecG)*60.0 - iDecM)*60.0;

	sprintf(temporal,"%02d",iDecG);
	gtk_entry_set_text(GTK_ENTRY(EntryDecGrado),temporal);
	sprintf(temporal,"%02d",abs(iDecM));
	gtk_entry_set_text(GTK_ENTRY(EntryDecMinuto),temporal);
	sprintf(temporal,"%0.1f",fabs(iDecS));
	gtk_entry_set_text(GTK_ENTRY(EntryDecSeg),temporal);

	sprintf(mensaje,"DEC %d %d %d AR %d %d %0.1f
",iDecG,abs(iDecM),abs((int)iDecS),iARH,abs(iARM),iARS);
	//printf("Mensaje: %s\n",mensaje);

	//char caracteres[200];
	//MandaYLeeSocket(mensaje,200,caracteres);

	//printf("%s\n",caracteres);
	//gtk_timeout_remove( TagSegundo );
	if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBit
acora)))
	{
	    sprintf(buffbitacora,"Nuevas coordenadas de sol %s",mensaje);
	    agrega_a_bitacora(buffbitacora);
	}
	gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraest
ado,"Posicion del sol calculada, presiona \"Aplicar\" para mover
telescopio a esa posicion");
```

```
}

gint BuscaCenit(gpointer datos)
{
      float X,Y; //angulo en Grados
      float x;   //angulo en segundos de arco
      float y;   //angulo en grados
      char mensaje[200];
      gboolean listoX=FALSE;
      gboolean listoY=FALSE;
      X=atof(gtk_label_get_text(GTK_LABEL(LabelInclinometroX)));//en
grados
      Y=atof(gtk_label_get_text(GTK_LABEL(LabelInclinometroY)));
      printf("X=%f Y=%f\n",X,Y);
      //24hrs - 360
      //  x   - X
      x=X *60.0*30.0;
      if(x<-0.4 || x>0.4)
      {
          if(x<0)
          {
            //printf("DECOF %d +D\n",(int)fabs(x));
            sprintf(mensaje,"DECOF %d +D ",(int)fabs(x));
            printf("Mensaje: %s\n",mensaje);

            char caracteres[200];
            MandaYLeeSocket(mensaje,200,caracteres);
          }
          else
            //printf("DECOF %d -D\n",(int)fabs(x));
          {
            //printf("DECOF %d +D\n",(int)fabs(x));
            sprintf(mensaje,"DECOF %d -D ",(int)fabs(x));
            printf("Mensaje: %s\n",mensaje);

            char caracteres[200];
            MandaYLeeSocket(mensaje,200,caracteres);
          }
      }
      else
          listoX=TRUE;
      y=24.0*Y/360.0*60.0*60.0;
      if(y<-0.4 || y>0.4)
      {
          if(y<0)
            //printf("AROF %d +A\n",(int)fabs(y));
          {
            //printf("DECOF %d +D\n",(int)fabs(x));
            sprintf(mensaje,"AROF %d -A ",(int)fabs(y));
            printf("Mensaje: %s\n",mensaje);

            char caracteres[200];
            MandaYLeeSocket(mensaje,200,caracteres);
          }
          else
            //printf("AROF %d -A\n",(int)fabs(y));
          {
            //printf("DECOF %d +D\n",(int)fabs(x));
```

```
            sprintf(mensaje,"AROF %d +A ",(int)fabs(y));
            printf("Mensaje: %s\n",mensaje);

            char caracteres[200];
            MandaYLeeSocket(mensaje,200,caracteres);
          }

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
          {
            sprintf(buffbitacora,"Inicio buscar cenit");
            agrega_a_bitacora(buffbitacora);
          }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Buscando Cenit");

      }
      else
          listoY=TRUE;
      if(listoY && listoX)
      {
        printf("Felicidades Encontre el Cenit....\n");
        printf("CORRIGE_CENIT\n");
        gtk_timeout_remove(TagBuscaCenit);

gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(ToggleBuscaCenit),FALSE)
;

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
        {
            sprintf(buffbitacora,"Cenit encontrado");
            agrega_a_bitacora(buffbitacora);
        }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Encontre cenit, ejecuta el comando de corregir cenit");

      }
      return(1);
}

void on_BotonBuscaCenit_toggled(GtkToggleButton *button,gpointer
user_data)
{
      gpointer datos;

        if(gtk_toggle_button_get_active(button))
      {
        //BuscaCenit(datos);
        printf("buscando cenit\n");
        TagBuscaCenit=gtk_timeout_add(
TemporizadorBuscaCenit,BuscaCenit,(gpointer)button );

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Buscando Cenit");
      }
```

```
      else
      {
        //printf("cancele busqueda cenit\n");
        gtk_timeout_remove( TagBuscaCenit );

if(gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(ToggleGenerarBitacora
)))
        {
            sprintf(buffbitacora,"Buscar cenit cancelado");
            agrega_a_bitacora(buffbitacora);
        }

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Buscando Cenit CANCELADO");
      }
}

gboolean on_BotonGrabarBitacora_clicked(GtkWidget *widget, GdkEvent
*event, gpointer data)
{
    FILE *bitacora;
    printf("Grabando bitacora....\n");
    GtkWidget *Texto;
    GtkTextIter   InicioIter;
    GtkTextIter FinIter;
    GtkTextBuffer *buffer;
    gchar *texto;
    buffer=gtk_text_view_get_buffer(GTK_TEXT_VIEW(BitacoraTexto));
    gtk_text_buffer_get_bounds (buffer,&InicioIter,&FinIter);
  //gtk_text_buffer_get_start_iter( buffer,&InicioIter );
    //gtk_text_buffer_get_end_iter( buffer,&FinIter );

    texto=gtk_text_buffer_get_text(buffer,&InicioIter,&FinIter,FALSE);
    //printf("nombre archivo: %s\n%s\n",nombre_archivo_bitacora,texto);

    bitacora=fopen(nombre_archivo_bitacora,"w");
    fwrite(texto,1,strlen(texto),bitacora);
    fclose(bitacora);
    return(FALSE);
}

gboolean on_BotonHora_clicked(GtkWidget *widget, GdkEvent *event,
gpointer data)
{
    FILE *bitacora;
    GtkWidget *Texto;
    GtkTextIter   InicioIter;
    GtkTextIter FinIter;
    GtkTextBuffer *buffer;
    gchar *texto;
    char hora_actual[30];

  time_t rawtime;
  struct tm * timeinfo;

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );
```

```
  sprintf(hora_actual,"%02d:%02d:%02d ",timeinfo->tm_hour,timeinfo-
>tm_min,timeinfo->tm_sec);

    buffer=gtk_text_view_get_buffer(GTK_TEXT_VIEW(BitacoraTexto));
    gtk_text_buffer_get_bounds (buffer,&InicioIter,&FinIter);
  //gtk_text_buffer_get_start_iter( buffer,&InicioIter );
    //gtk_text_buffer_get_end_iter( buffer,&FinIter );

    texto=gtk_text_buffer_get_text(buffer,&InicioIter,&FinIter,FALSE);
    //printf("nombre archivo: %s\n%s\n",nombre_archivo_bitacora,texto);
    strcat(texto,hora_actual);
    gtk_text_buffer_set_text(buffer,texto,strlen(texto));
    gtk_text_view_set_buffer (GTK_TEXT_VIEW(BitacoraTexto),buffer);

    bitacora=fopen(nombre_archivo_bitacora,"w");
    fwrite(texto,1,strlen(texto),bitacora);
    fclose(bitacora);
    return(FALSE);
}

void
on_VentanaBitacora_destroy(GtkWidget *widget, gpointer data)
{
    printf("Saliendo de bitacora....\n");
}

gboolean on_BotonVerBitacora_clicked(GtkWidget *widget, GdkEvent
*event, gpointer data)
{
    GtkBuilder *builder;
    GtkWidget *VentanaBitacora;
    //GtkWidget *BitacoraTexto;
    GtkTextBuffer *buffer;
    char texto[30000];
    FILE *bitacora;
    int i=0;

    time_t rawtime;
    struct tm * timeinfo;
    //char nombre_archivo_bitacora[200];

    time ( &rawtime );
    timeinfo = localtime ( &rawtime );
    sprintf
(nombre_archivo_bitacora,"Bitacora_rt5_%04d%02d%02d.txt",timeinfo-
>tm_year+1900,timeinfo->tm_mon+1,timeinfo->tm_mday);

    GError     *error = NULL;
    builder = gtk_builder_new();
    if( ! gtk_builder_add_from_file( builder, "ControlRT5.glade",
&error ) )
    {
        g_warning( "%s", error->message );
        g_free( error );
        return;
    }
    VentanaBitacora =
GTK_WIDGET(gtk_builder_get_object(builder,"VentanaBitacora"));
```

```
    BitacoraTexto =
GTK_WIDGET(gtk_builder_get_object(builder,"VistaTexto"));
    gtk_builder_connect_signals( builder, NULL );
    g_signal_connect((gpointer) VentanaBitacora, "destroy",
G_CALLBACK(on_VentanaBitacora_destroy), NULL);
    g_signal_connect((gpointer) VentanaBitacora,
"on_BotonGrabarBitacora_clicked",
G_CALLBACK(on_BotonGrabarBitacora_clicked), NULL);
    g_signal_connect((gpointer) VentanaBitacora,
"on_BotonHora_clicked", G_CALLBACK(on_BotonHora_clicked), NULL);
    buffer=gtk_text_buffer_new(NULL);
    bitacora=fopen(nombre_archivo_bitacora,"r");
    if(bitacora!=NULL)
    {
      do
      {
      fread(&texto[i],1,1,bitacora);
      i++;
      }while(!feof(bitacora));
      //printf("%s\n",texto);
      gtk_text_buffer_set_text(buffer,texto,i-1);
      gtk_text_view_set_buffer (GTK_TEXT_VIEW(BitacoraTexto),buffer);
      gtk_widget_show(VentanaBitacora);
      g_object_unref( G_OBJECT( builder ) );
      fclose(bitacora);
    }
    else
      printf("No hay archivo de bitacora de hoy\n");
    return(FALSE);
}

gboolean Esconde_Emergente(gpointer data){
    GtkWidget * Emergente = (GtkWidget *) data;
    gtk_widget_hide(Emergente);
    gtk_widget_show(window);
    return(FALSE);
}

int
main( int    argc,
      char **argv )
{
    GtkBuilder *builder;
    //GtkWidget  *window;
    GtkWidget  *Emergente;

    GError     *error = NULL;

    /* Init GTK+ */
    gtk_init( &argc, &argv );

    /* Create new GtkBuilder object */
    builder = gtk_builder_new();
    /* Load UI from file. If error occurs, report it and quit
application.
     * Replace "tut.glade" with your saved project. */
    if( ! gtk_builder_add_from_file( builder, "ControlRT5.glade",
&error ) )
```

```
    {
        g_warning( "%s", error->message );
        g_free( error );
        return( 1 );
    }

    Emergente =
GTK_WIDGET(gtk_builder_get_object(builder,"Emergente"));
    gtk_widget_show(Emergente);
    g_timeout_add (3000, Esconde_Emergente, Emergente);


    /* Get main window pointer from UI */
    window = GTK_WIDGET( gtk_builder_get_object( builder, "ControlRT5"
) );
    EntryARHora = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryARHora" ) );
    EntryARMinuto = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryARMinuto" ) );
    EntryARSeg = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryARSeg" ) );
    EntryDecGrado = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecGrado" ) );
    EntryDecMinuto = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecMinuto" ) );
    EntryDecSeg = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecSeg" ) );
    EntryOffset = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryOffset" ) );
    EntryEpoca = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryEpoca" ) );
    EntryDecFijoG = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecFijoG" ) );
    EntryDecFijoM = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecFijoM" ) );
    EntryDecFijoS = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryDecFijoS" ) );
    EntryAHFijoH = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryAHFijoH" ) );
    EntryAHFijoM = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryAHFijoM" ) );
    EntryAHFijoS = GTK_WIDGET( gtk_builder_get_object( builder,
"EntryAHFijoS" ) );
    EntradaCorreccionAcimut = GTK_WIDGET( gtk_builder_get_object(
builder, "EntradaCorreccionAcimut" ) );
    EntradaCorreccionAltura = GTK_WIDGET( gtk_builder_get_object(
builder, "EntradaCorreccionAltura" ) );
    LabelAR = GTK_WIDGET( gtk_builder_get_object( builder, "LabelAR" )
);
    LabelDec = GTK_WIDGET( gtk_builder_get_object( builder, "LabelDec"
) );
    LabelAH = GTK_WIDGET( gtk_builder_get_object( builder, "LabelAH" )
);
    LabelTS = GTK_WIDGET( gtk_builder_get_object( builder, "LabelTS" )
);
    LabelTL = GTK_WIDGET( gtk_builder_get_object( builder, "LabelTL" )
);
```

```
    LabelTU = GTK_WIDGET( gtk_builder_get_object( builder, "LabelTU" )
);
    LabelFecha = GTK_WIDGET( gtk_builder_get_object( builder,
"LabelFecha" ) );
    LabelLat = GTK_WIDGET( gtk_builder_get_object( builder, "LabelLat"
) );
    LabelInclinometroX = GTK_WIDGET( gtk_builder_get_object( builder,
"LabelInclinometroX" ) );
    LabelInclinometroY = GTK_WIDGET( gtk_builder_get_object( builder,
"LabelInclinometroY" ) );
    ToggleBuscaCenit = GTK_WIDGET( gtk_builder_get_object( builder,
"BotonBuscaCenit" ) );
    ToggleGenerarBitacora = GTK_WIDGET( gtk_builder_get_object(
builder, "BotonGenerarBitacora" ) );
    ToggleActivarCorreccion = GTK_WIDGET( gtk_builder_get_object(
builder, "BotonActivarCorreccion" ) );
    BarraEstado = GTK_WIDGET( gtk_builder_get_object( builder,
"BarraEstado" )
);
    BotonPonerBuscador = GTK_WIDGET( gtk_builder_get_object( builder,
"BotonPonerBuscadorChico" ) );
    TextoBitacoraInstrumento =
GTK_WIDGET(gtk_builder_get_object(builder,"TextoBitacoraInstrumento"));
    TextoBitacoraParticipantes =
GTK_WIDGET(gtk_builder_get_object(builder,"TextoBitacoraParticipantes")
);
    TextoBitacoraComentarios =
GTK_WIDGET(gtk_builder_get_object(builder,"TextoBitacoraComentarios"));

    gtk_widget_set_sensitive(BotonPonerBuscador,FALSE);
    gtk_entry_set_text(GTK_ENTRY(EntryOffset),"100");
    gtk_entry_set_text(GTK_ENTRY(EntryEpoca),"2010.2");
    gtk_label_set_text(GTK_LABEL(LabelInclinometroX),"-5.5");
    gtk_label_set_text(GTK_LABEL(LabelInclinometroY),"-5.5");
    id_contexto_barraestado=gtk_statusbar_get_context_id
(GTK_STATUSBAR(BarraEstado),NULL);

gtk_statusbar_push(GTK_STATUSBAR(BarraEstado),id_contexto_barraestado,"
Para empezar utiliza \"Prueba Comunicacion\" que esta en
\"Configuracion\"");

    //char *markup;
    //markup = g_markup_printf_escaped ("<span underline_color=\"red\"
underline=\"error\" color=\"blue\" style=\"italic\" font=\"Century
Schoolbook L Bold 30\"> 0:00:00.0 </span>");
    //gtk_label_set_markup (GTK_LABEL (LabelAR), markup);
    //markup = g_markup_printf_escaped ("<span underline_color=\"red\"
underline=\"error\" color=\"red\" style=\"italic\" font=\"Century
Schoolbook L Bold 30\"> 0:00:00.0 </span>");
    //gtk_label_set_markup (GTK_LABEL (LabelDec), markup);
    //markup = g_markup_printf_escaped ("<span underline_color=\"red\"
underline=\"error\" color=\"blue\" style=\"italic\" font=\"Century
Schoolbook L 16\"> 0:00:00 </span>");
    //gtk_label_set_markup (GTK_LABEL (LabelAH), markup);
    //g_free (markup);
    /* Connect signals */
    gtk_builder_connect_signals( builder, NULL );
```

```
    g_signal_connect((gpointer) window, "destroy", G_CALLBACK(destroy),
NULL);
    g_signal_connect((gpointer) window, "on_BotonParo_clicked",
G_CALLBACK(on_BotonEnvia_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonEnvia_clicked",
G_CALLBACK(on_BotonEnvia_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonInicializar_clicked",
G_CALLBACK(on_BotonInicializar_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonCorrigeHora_clicked",
G_CALLBACK(on_BotonCorrigeHora_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonCenit_clicked",
G_CALLBACK(on_BotonCenit_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonGuiado_toggled",
G_CALLBACK(on_BotonGuiado_toggled), NULL);
    g_signal_connect((gpointer) window, "on_BotonNorte_clicked",
G_CALLBACK(on_BotonNorte_clicked), NULL);
    g_signal_connect((gpointer) window,
"on_BotonNorte_key_press_event",
G_CALLBACK(on_BotonNorte_key_press_event), NULL);
    g_signal_connect((gpointer) window, "on_BotonNE_clicked",
G_CALLBACK(on_BotonNE_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonEste_clicked",
G_CALLBACK(on_BotonEste_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonSE_clicked",
G_CALLBACK(on_BotonSE_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonSur_clicked",
G_CALLBACK(on_BotonSur_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonSW_clicked",
G_CALLBACK(on_BotonSW_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonOeste_clicked",
G_CALLBACK(on_BotonOeste_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonNW_clicked",
G_CALLBACK(on_BotonNW_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonCorrigeCenit_clicked",
G_CALLBACK(on_BotonCorrigeCenit_clicked), NULL);
    g_signal_connect((gpointer) window,
"on_BotonCorrigeCoordenadas_clicked",
G_CALLBACK(on_BotonCorrigeCoordenadas_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonPruebaComm_clicked",
G_CALLBACK(on_BotonPruebaComm_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonAbrirLazo_clicked",
G_CALLBACK(on_BotonAbrirLazo_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonInclinometro_toggled",
G_CALLBACK(on_BotonInclinometro_toggled), NULL);
    g_signal_connect((gpointer) window, "on_BotonGuiadoSol_toggled",
G_CALLBACK(on_BotonGuiadoSol_toggled), NULL);
    g_signal_connect((gpointer) window, "on_ToggleBarridoSol_toggled",
G_CALLBACK(on_ToggleBarridoSol_toggled), NULL);
    g_signal_connect((gpointer) window, "on_BotonSol_clicked",
G_CALLBACK(on_BotonSol_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonBuscaCenit_toggled",
G_CALLBACK(on_BotonBuscaCenit_toggled), NULL);
    g_signal_connect((gpointer) window,
"on_BotonPonerReceptor_clicked",
G_CALLBACK(on_BotonPonerReceptor_clicked), NULL);
    g_signal_connect((gpointer) window,
"on_BotonPonerBuscadorChico_clicked",
G_CALLBACK(on_BotonPonerBuscadorChico_clicked), NULL);
```

```
    g_signal_connect((gpointer) window, "on_BotonFijoDec_clicked",
G_CALLBACK(on_BotonFijoDec_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonFijoAH_clicked",
G_CALLBACK(on_BotonFijoAH_clicked), NULL);
    g_signal_connect((gpointer) window,
"on_BotonGenerarBitacora_toggled",
G_CALLBACK(on_BotonGenerarBitacora_toggled), NULL);
    g_signal_connect((gpointer) window, "on_BotonGrabarEncabezado",
G_CALLBACK(on_BotonGrabarEncabezado_clicked), NULL);
    g_signal_connect((gpointer) window, "on_Fijo1_key_press_event",
G_CALLBACK(on_PresionaTecla), NULL);
    g_signal_connect((gpointer) window, "on_BotonLuna_clicked",
G_CALLBACK(on_BotonLuna_clicked), NULL);
    g_signal_connect((gpointer) window, "on_BotonGuiadoLuna_toggled",
G_CALLBACK(on_BotonGuiadoLuna_toggled), NULL);
    g_signal_connect((gpointer) window,
"on_BotonActivarCorreccion_toggled",G_CALLBACK(on_BotonActivarCorreccio
n_toggled
) , NULL);
    g_signal_connect((gpointer) window, "on_BotonAcercaDe_clicked",
G_CALLBACK(on_BotonAcercaDe_clicked), NULL);
    g_signal_connect((gpointer) window,
"on_EntryOffset_focus_in_event",
G_CALLBACK(on_EntryOffset_focus_in_event), NULL);
    g_signal_connect((gpointer) window,
"on_EntryOffset_focus_out_event",
G_CALLBACK(on_EntryOffset_focus_out_event), NULL);
    g_signal_connect((gpointer) window, "on_BotonVerBitacora_clicked",
G_CALLBACK(on_BotonVerBitacora_clicked), NULL);

    //gtk_key_snooper_install ((GtkKeySnoopFunc)snooper, 0);

    /* Destroy builder, since we don't need it anymore */
    g_object_unref( G_OBJECT( builder ) );


    /* Show window. All other widgets are automatically shown by
GtkBuilder */
    //gtk_widget_show( window );

    /* Start main loop */
    gtk_main();

    return( 0 );
}
```

Anexo I benisocket.h

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
```

```
typedef struct
{
  int socket;
  int msgsock;
  char *buffer_transmision;
  char *buffer_recepcion;
  char Direccion[50];
  char Puerto[10];
  unsigned char Tipo; //Cliente=1 o servidor=0
  struct sockaddr_in MiDir; //Datos del local
  struct sockaddr_in SuDir; //Datos del remoto
  int rc;
}BeniSocket;

int initBeniSocketServer(BeniSocket *Lsock,int TcpPuerto,int BuffSize)
{
  struct sockaddr_in server;
  Lsock->buffer_transmision=(char*)malloc(sizeof(char) * BuffSize);
  Lsock->buffer_recepcion=(char*)malloc(sizeof(char)*BuffSize);
  Lsock->socket=socket(AF_INET,SOCK_STREAM,0);
  if(Lsock->socket<0)
  {
    perror("socket");
    Lsock->socket=-1;
    Lsock->msgsock=-1;
    return(1);
    //exit(1);
  }
  server.sin_family=AF_INET;
  server.sin_addr.s_addr=INADDR_ANY;
  server.sin_port=htons(TcpPuerto);
  if(bind(Lsock->socket,&server,sizeof(server)))
  {
    perror("bind");
    Lsock->socket=-1;
    Lsock->msgsock=-1;
    return(1);//exit(1);
  }
  return(0);
}

int initBeniSocketClient(BeniSocket *Lsock,char *DirSvr,int TcpPuerto)
{
  struct hostent * hp;
```

```
  if((hp=gethostbyname(DirSvr))==0)
  {
    perror("gethostbyname");
    Lsock->socket=-1;
    Lsock->msgsock=-1;
    return(1);
  }
  if((Lsock->socket=socket(AF_INET,SOCK_STREAM,0))<0)
  {
    perror("socket");
    Lsock->socket=-1;
    Lsock->msgsock=-1;
    return(1);
    //exit(1);
  }
  bzero((char*)&Lsock->SuDir,sizeof(Lsock->SuDir));
  Lsock->SuDir.sin_family=AF_INET;
  Lsock->SuDir.sin_port=htons(TcpPuerto);
  memcpy(&Lsock->SuDir.sin_addr,hp->h_addr,hp->h_length);
  if(Lsock->SuDir.sin_addr.s_addr<=0)
  {
    perror("bad address after gethostbyname");
    exit(1);
  }

  //fcntl (Lsock->socket, F_SETFL, O_NONBLOCK);//establece al socket como no bloqueante

  if(connect(Lsock->socket,(struct sockaddr*)&Lsock->SuDir,sizeof(Lsock->SuDir))<0)
  {
    perror("connect");
    Lsock->socket=-1;
    Lsock->msgsock=-1;
    return(1);
  }
  return(0);
  //perror("Se conecto de poca madre");
}

doRead(int sock,char *buf,int amountNeeded)
{
  register int i;
  int rc;
  char *bpt;
  int count=amountNeeded;
  int amtread;
```

```
  bpt=buf;
  amtread=0;
again:
  if((rc=read(sock,bpt,count))<0)
  {
    perror("doRead: reading socket stream");
    return(1);
  }
  amtread+=rc;
  if(amtread<amountNeeded)
  {
    count=count-rc;
    bpt=bpt+rc;
    goto again;
  }
}
```