AUTOMATIZACIÓN DEL SISTEMA DE ILUMINACIÓN DEL ESPECTRÓGRAFO BOLLER & Ch.

Versión 1.0.

F. Murillo, G. Sierra, B. Martínez, J.M. Murillo, G. Guisa, A. Córdoba, E. López.

Resumen.

El presente trabajo documenta el diseño e implementación de un sistema que permite la operación remota del conjunto de lámparas del espectrógrafo Boller & Ch. Se describe el diseño mecánico y electrónico, así como el programa de usuario.

Junio de 2008.

1 Introducción.

En años recientes ha surgido el interés por realizar observaciones remotas en el OAN. Para dirigir los esfuerzos de desarrollo en ese sentido, a la fecha se ha modificado buena parte de los instrumentos de observación para poder operarlos utilizando la red Ethernet. Para seguir con esta tarea se decidió construir un sistema que permitiera el manejo remoto del sistema de iluminación del espectrógrafo Boller & Ch. El cual hasta la fecha fue operado de manera manual.

La implementación de este sistema tiene como resultado el aumento de la eficiencia durante las observaciones con este instrumento, ya que anteriormente el asistente de cúpula tenia que subir al piso de telescopio para operarlo manualmente cada que se requería tomar una lámpara de comparación, actividad que requería de 3 minutos, actualmente se realiza en 5 segundos con el nuevo sistema. Este es un avance importante si se considera que el espectrógrafo Boller & Ch tiene una demanda importante entre los usuarios del OAN y tan solo en el primer semestre del 2008 ocupó el 33% del tiempo asignado en el telescopio de 2.1m.

La automatización del sistema de iluminación requirió de la construcción de un mecanismo para motorizar la selección de las lámparas. Además del diseño de un controlador de motores eficiente con capacidad de recibir instrucciones mediante la red Ethernet. Se desarrollo también un programa de interfaz de usuario amigable para su operación.

2. El sistema de iluminación del Boller & Ch.

El espectrógrafo Boller & Chivens cuenta con un sistema de iluminación para comparación que consta de dos lámparas, una de Cobre-Argón (CuAg) y otra de bulbo de Neón. Un arreglo periscópico permite introducir la fuente de comparación en el eje óptico del espectrógrafo cuando un espectro de comparación va a ser tomado.

La lámpara de Neón esta montada en un costado del espectrógrafo y la de Cobre-Argón en el costado opuesto. Las fuentes de alimentación de ambas lámparas se encuentran montadas sobre el cuerpo del instrumento y cuentan con interruptores de encendido y apagado que se operan de manera manual.

Para tomar el espectro de una lámpara, es necesario introducir el periscopio y proyectar la fuente de luz hacia la rendija. Para ello anteriormente se tenía un mecanismo, operado de manera manual, que consistía en una perilla (Ver figura 1) con dos ejes de movimiento, el primero denominado “Dentro-fuera” permitía introducir el periscopio; el segundo denominado “Giro”, con dos posiciones posibles permitía seleccionar una de las dos lámparas como fuente de luz.

El siguiente capitulo describe el trabajo realizado para la automatización de este sistema.

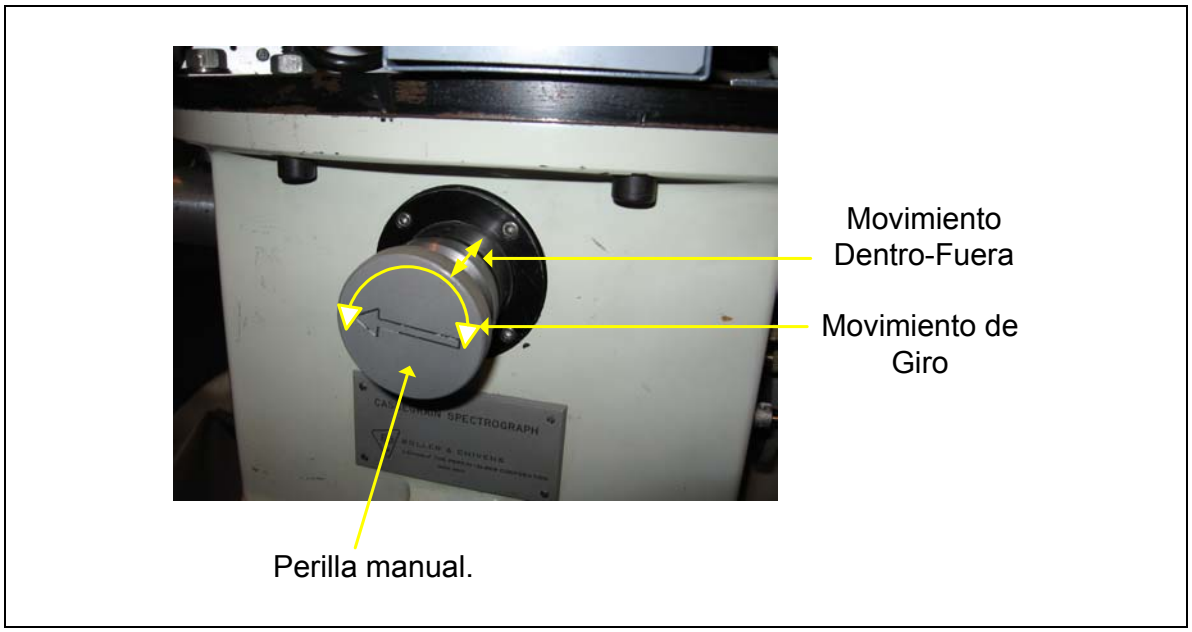


Figura 1. Perilla manual para la selección de lámparas.

3. Implementación del sistema automático.

Para automatizar el sistema de iluminación se diseño un mecanismo que permite mover la perilla en sus dos movimientos utilizando motores eléctricos. Se diseño también una electrónica de control de movimiento mediante la cual es posible manipular la perilla desde la computadora de usuario, que en este caso es Sonaja y esta ubicada en el cuarto de observación del telescopio de 2.1m (Ver Figura 2). En esta computadora se instaló un programa gráfico hecho en lenguaje TCL-TK con botones para manipular la montura de manera amigable. Se utiliza la red para comunicar Sonaja con el controlador de motores. En las siguientes secciones se describe con detalle el diseño e implementación cada una de las partes que forman el sistema.

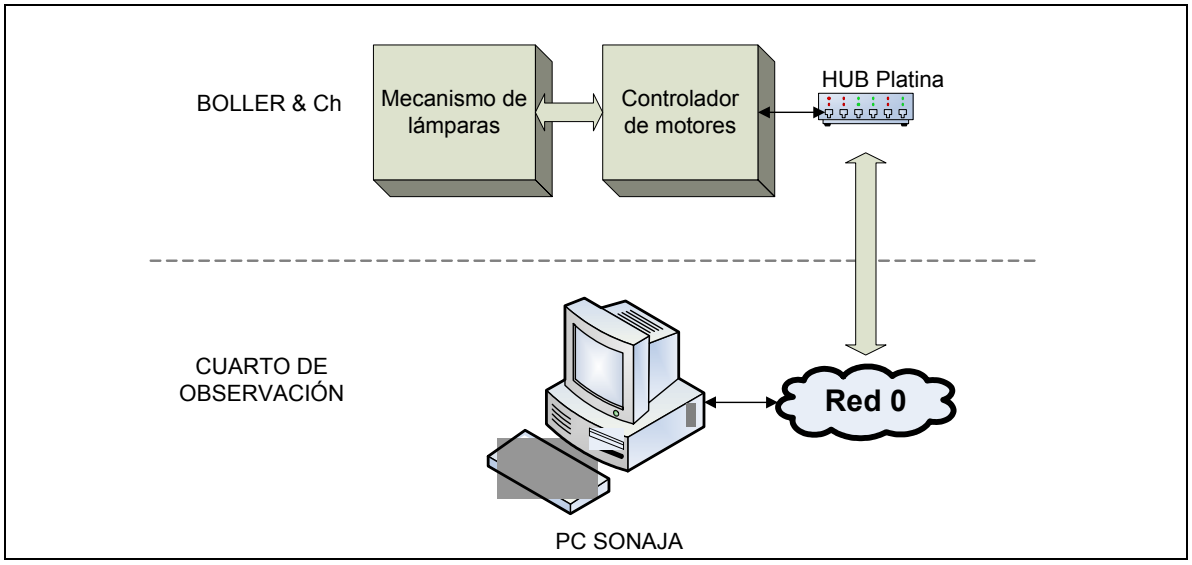


Figura 2. Distribución del sistema de lámparas del Boller & Ch en el telescopio de 2.1m.

3.1 Sistema mecánico para acoplamiento de motores de DC.

El sistema consta de dos mecanismo de movimientos el primero mete y saca el periscopio al eje óptico del instrumento, el segundo gira la perilla para escoger la lámpara adecuada.

El primer mecanismo consta del motor que mete y saca el periscopio móvil para dirigir el haz de luz al instrumento. En la figura 3 se muestra un bosquejo del mecanismo. Este mecanismo esta sujeto por un tornillos 7/16-20 que ensambla la montura del motor a la platina del Boller & Chivens y se ajusta por medio de un ojo de chal. El motor es un motor globe motor con reducción del tipo E-2030 y tiene un piñón sujeto a su eje. El piñón tiene 12 dientes con un paso de 24, un diámetro de paso de 0.5" y es de aluminio, del proveedor PIC design con numero de parte G57-12.

Nº DE ELEMENTO	NÚMERO DE PIEZA	DESCRIPCIÓN	CANT.
1	RSR 13 MOTOR gLOBE MOTORS	Motro globe motor 2030	1
2	001_B&Ch_baseMotor1		1
3	006_B&Ch_base motor		1
4	PIC Design-G57-12	No dinete: 12, P.D. 1/2" SS Spurg Gear-24 pitch	1
5	007_B&Ch_eje Motor2		1
6	HX-SHCS 0.138-32x0.375x0.375-S		2
7	16B_B&Ch CUBIERTAB	No mostrada	1
8	16_B&Ch CUBIERTA	No mostrada	1

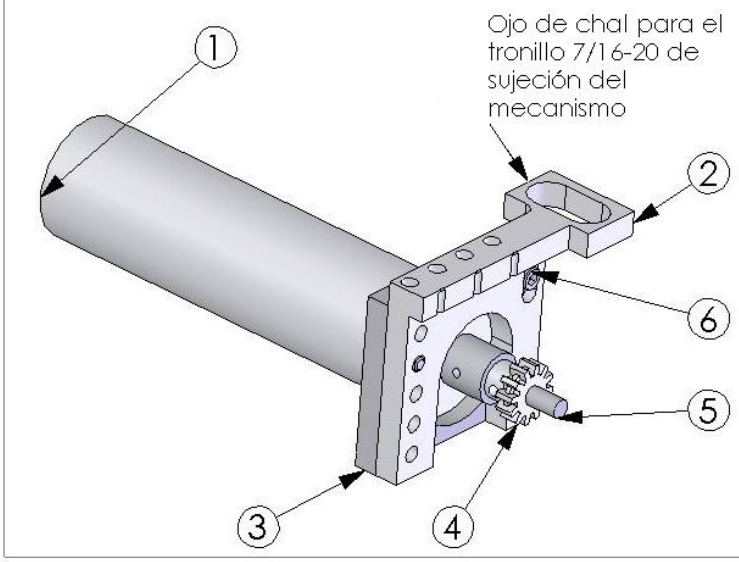


Figura 3. Bosquejo del primer mecanismo.

El segundo mecanismo es movido por el piñón del primer mecanismo, por medio de un sistema piñón-cremallera. Consta de una cremallera que tiene 2.5" de largo y 2 piezas en forma de abrazadera que se atornillan entre si para sujetar el tubo del periscopio (Ver figura 4). Sobre la abrazadera se sujeta un segundo motor utilizado en el eje de giro de la perilla. El motor cuenta con un piñón de bronce, que hace girar un engrane recto de acero inoxidable que se sujeta a la perilla.

Nº DE ELEMENTO	NÚMERO DE PIEZA	DESCRIPCIÓN	CANT.
1	005_B&Ch_eje Motor1		1
2	PIC Design -G78-20		1
3	PIC Design -G77-56		1
4	008_B&Ch_MonturaRACK		1
5	004_B&Ch_ABRAZADERA2	Abrazadera Izquierda	1
6	002_B&Ch_ABRAZADERA1	Abrazadera Derecha	1
7	RSR T3 MOTOR GLOBE MOTORS		1
8	003_B&Ch_baseMotor2		1
9	HX-SHCS 0.112-40x0.375x0.375-S		1
10	0025_B&Ch_ABRAZADERA1		1
11	Omron switch D2F		3
12	010_B&Ch_plaquita de Switch		1
13	Omron B&Ch_switch D2F		1
14	011_B&Ch Montura de Switch		1
15	MirrorOmron switch D2F		1
16	PIC Design -AG-31	Fine Pitch Rack - 20 Deg. Pressure Angle	1
17	Mirror010_B&Ch_plaquita de Switch		1
18	14_B&Ch Montura de Switch Giro		1
19	012_B&Ch Tope1 Switch		1
20	013_B&Ch Tope1 No eje		2
21	0125_B&Ch Tope1 Switch		1
22	15_B&Ch Regaton circular		1

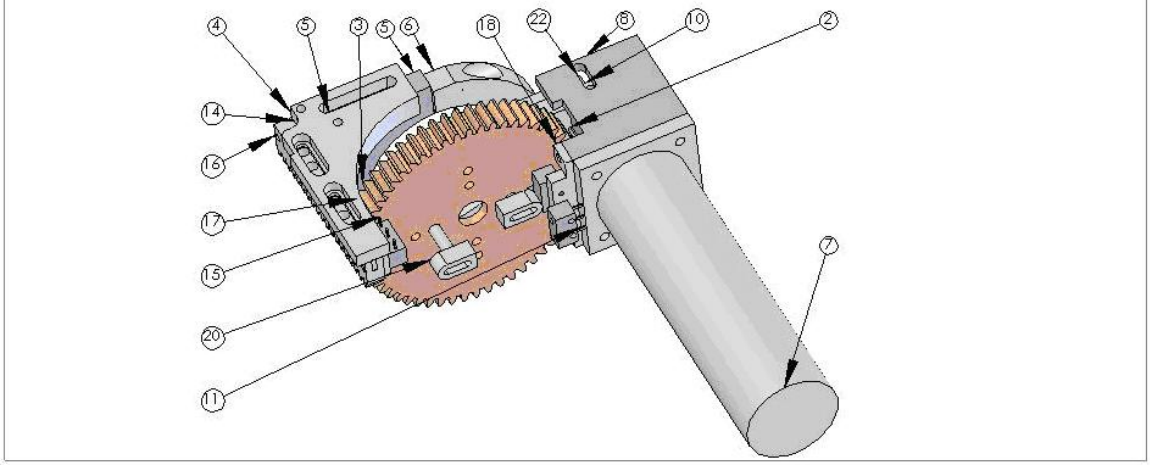


Figura 4. Bosquejo del segundo mecanismo.

En la figura 5 se muestra una fotografía del mecanismo de lámparas tomada en el laboratorio, en la figura 6 se muestra el mecanismo montado en el instrumento.

Finalmente tenemos una cubierta de lámina de aluminio que tapa y protege el sistema (Ver figura 7). Esta cubierta es de dos piezas remachadas y se monta con tres tornillos de sujeción sobre el mecanismo.

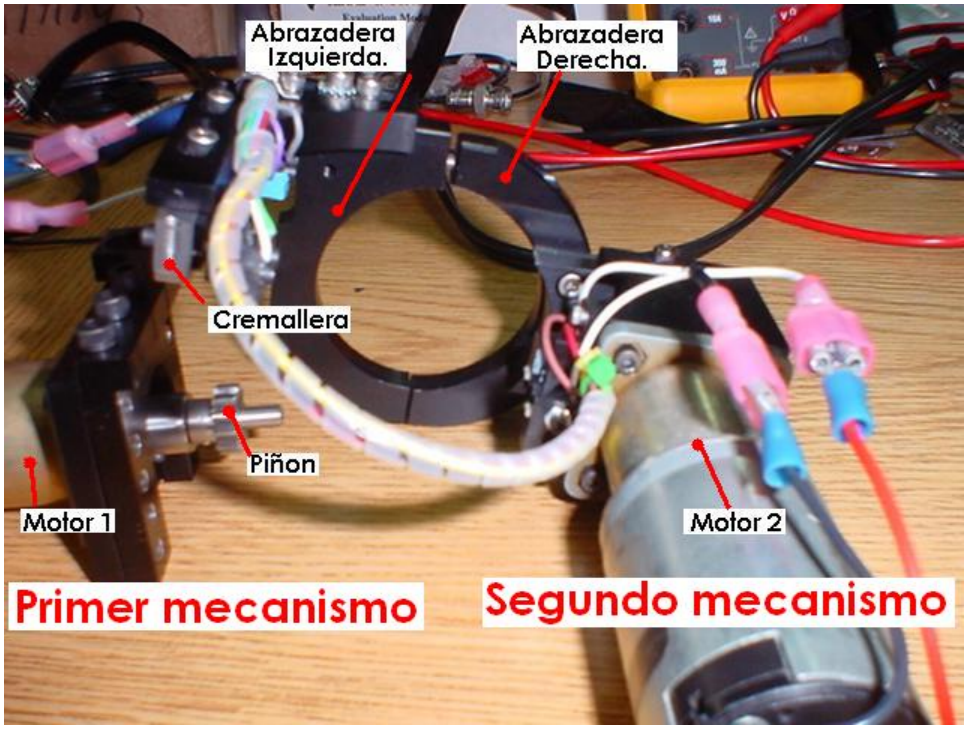


Figura 5. Vista del mecanismo de lamparas.

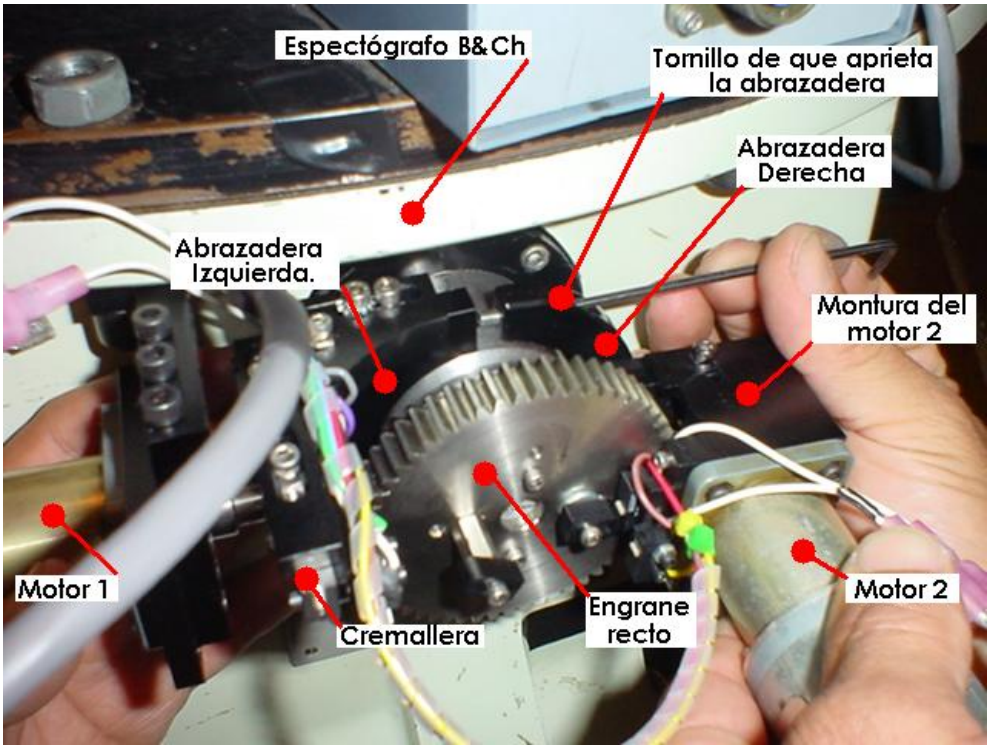


Figura 6. Montaje del mecanismo de lámparas sobre el instrumento.

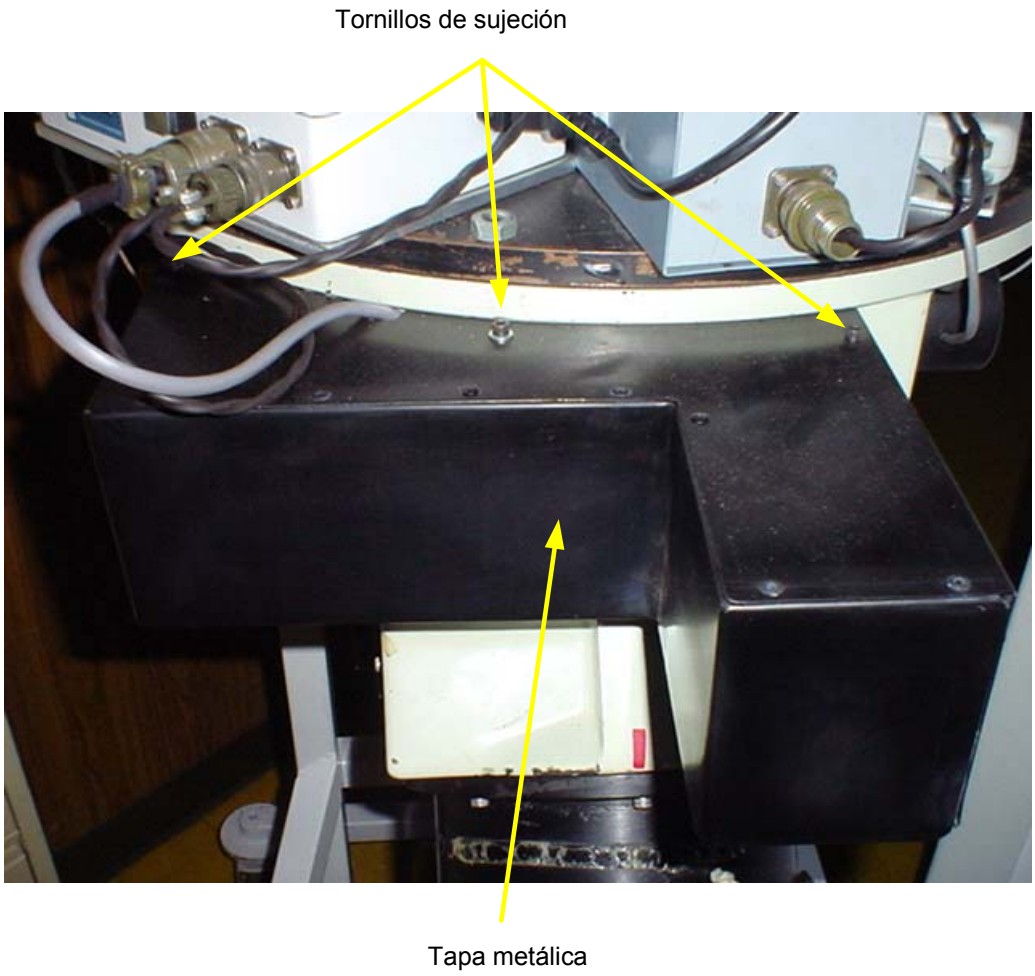


Figura 7. Vista de la cubierta metálica del mecanismo.

3.2 Controlador de motores.

Un controlador de motores fue diseñado para el manejo de la montura, dentro de las características de este control están las siguientes: tamaño pequeño ya que va montado al instrumento, eficiente en cuanto al consumo de potencia, la velocidad de movimiento de los motores es programable y puede ser operado vía red.

El controlador de motores consta de una caja metálica en cuyo interior alberga lo siguiente: Un par de fuentes de alimentación, una de 5V y otra de 24V; un circuito de control y un circuito de relevadores (Ver figura 8). En un costado se han colocado un par de conectores militares uno de 10 pines que lleva las señales de la montura, el segundo de 6 pines lleva las señales de encendido de las lámparas. También se han colocado dos botones para la operación manual de la montura (Ver figura 9).

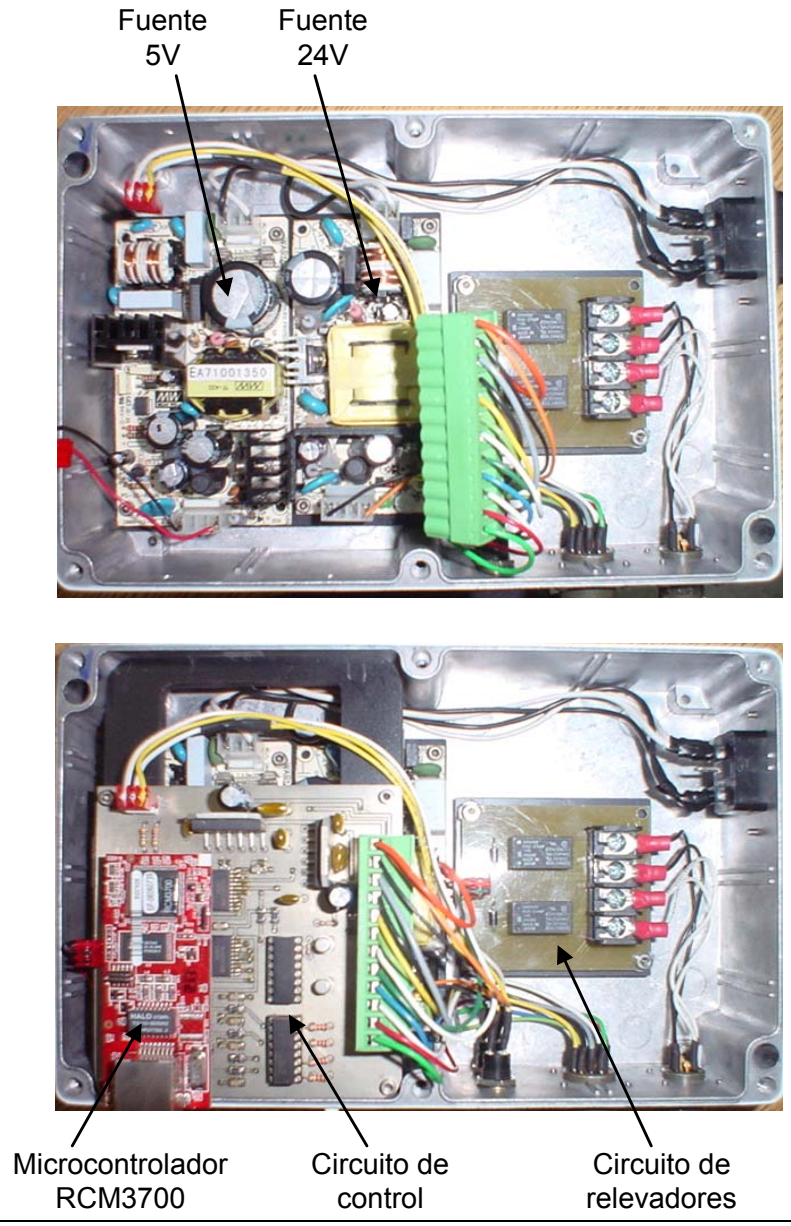


Figura 8. Vista interior del controlador de motores.

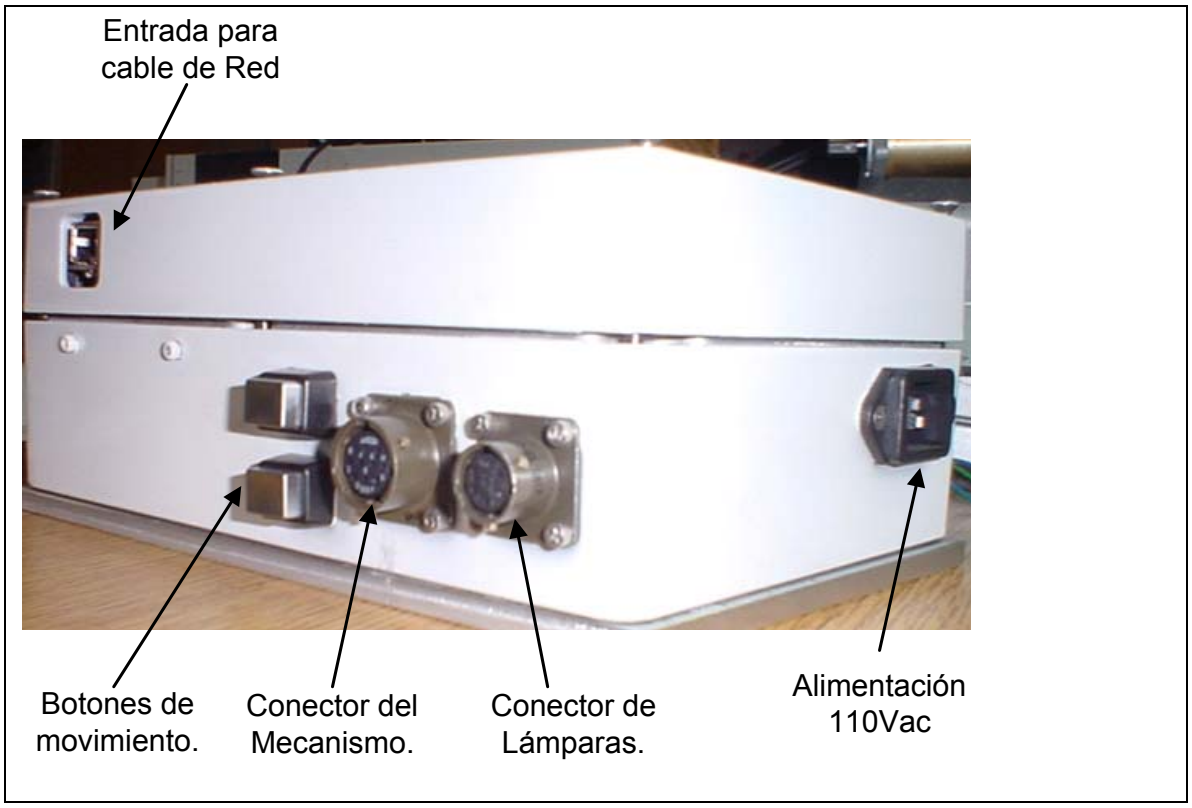


Figura 9. Vista exterior del controlador de motores.

3.2.1 Circuito de control.

El circuito de control diseñado para esta montura está organizado como se muestra en el diagrama a bloques de la figura 10, el esquemático y circuito impreso de esta tarjeta se muestran en los diagramas A1 y A2 del apéndice A. Como se muestra en la figura, el corazón del controlador esta formado por un microcontrolador RCM3700 con puerto ethernet, lo que permite realizar los movimientos de la montura de manera remota.

Un par de manejadores de motor de corriente directa LMD18200 es implementado, uno para cada motor. Este manejador se controla con una señal modulada en ancho de pulso PWM para el ajuste de la velocidad del motor, lo que hace que el circuito sea eficiente en cuanto a consumo de potencia. Una segunda señal es aplicada para definir el sentido del movimiento del motor.

Para el sensado de las posiciones de la perilla se utilizan cuatro interruptores, dos para el eje de movimiento dentro-fuera y dos para el eje de giro. Los interruptores funcionan a la vez como interruptores límite. La señal de los interruptores es optoacoplada y reforzada para ser introducida al microcontrolador.

El circuito acepta la entrada de un par de botones para operar la montura desde la caja que contiene el controlador de motores, que esta ubicada en el instrumento. Se ha dejado esta posibilidad ya que en ocasiones es deseable operar la montura estando al lado del instrumento, sin la necesidad de bajar hasta el cuarto de control, por ejemplo: cuando se quiere hacer una prueba de funcionalidad al instalar el instrumento. El botón superior es utilizado para realizar el movimiento dentro-fuera y el inferior para el giro.

El circuito suministra dos salidas a relevador para el encendido de las lámparas, estas salidas están formadas por un optoacoplador reforzado con un transistor 2N2222.

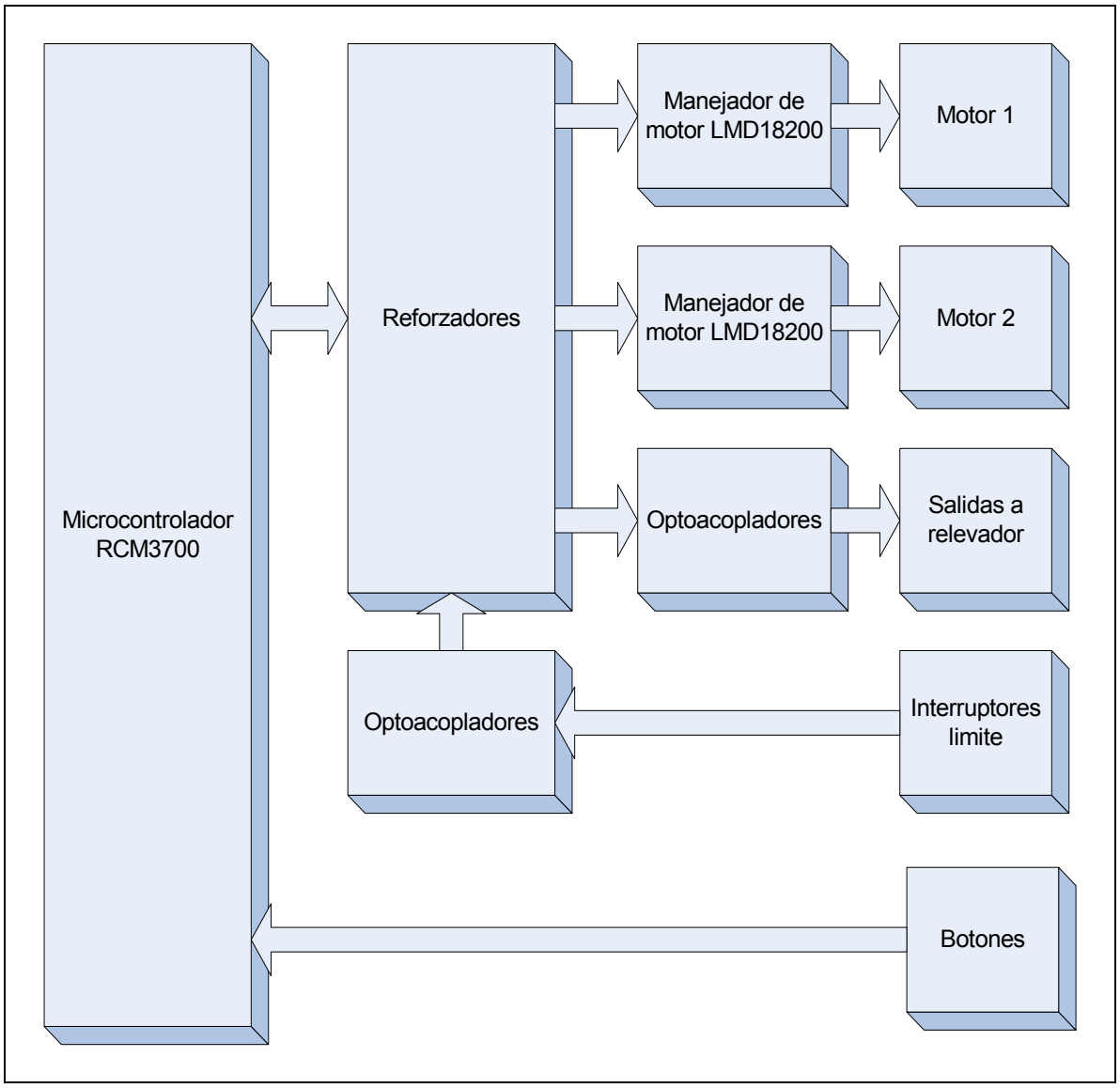


Figura 10. Organización del circuito de control de motores.

3.2.2 Circuito de relevadores.

Para el encendido y apagado de las lámparas se utiliza un par de relevadores que se activan con 24VDC, uno para cada lámpara. Estos relevadores son interruptores normalmente abiertos, y están conectados en paralelo a los interruptores manuales que existen en las fuentes de voltaje que alimentan las lámparas. Los diagramas A3 y A4 del apéndice A muestran el esquemático e impreso, respectivamente, del circuito de relevadores.

3.2.3 Cableado de los conectores militares.

Para el cableado de las señales que se utilizan en el manejo de la montura, como son la de los motores e interruptores de sensado, se ha utilizado un conector militar de 10

pines, colocado a un costado de la caja del controlador de motores (Ver figura 9), cuya asignación de pines se muestra en la tabla 1.

Tabla 1. Señales en el conector militar de 10 pines.

Pin	Señal
A	Puente1_1, Salida al motor 1 a través de los interruptores límite.
B	Puente1_2, Salida al motor 1 a través de los interruptores límite.
C	Puente2_1, Salida al motor 2.
D	Puente2_2, Salida al motor 2.
E	SW4, Entrada del interruptor de sensado 4.
F	SW3, Entrada del interruptor de sensado 3.
G	SW2, Entrada del interruptor de sensado 2.
H	SW1, Entrada del interruptor de sensado 1.

Para el cableado de las señales que controlan el encendido y apagado de las lámparas se utilizó un conector militar de 6 pines. La asignación de señales se muestra en la tabla 2.

Tabla 2. Señales en el conector militar de 6 pines.

Pin	Señal
A	N.C.
B	Al interruptor de la fuente de la lámpara de campo plano.
C	Al interruptor de la fuente de la lámpara de campo plano.
D	N.C.
E	Al interruptor de la fuente de la lámpara de CuAg.
F	Al interruptor de la fuente de la lámpara de CuAg.

3.3 Programa de control.

El programa de control es ejecutado por el microcontrolador RCM3700 que forma parte del circuito de control. Por completez de este reporte se ha incluido el listado completo de este programa en el apéndice B. El código denominado lámparas.c está escrito en lenguaje C y compilado en el paquete de desarrollo de la compañía Rabbit denominado “Dinamic C 8.30”.

3.3.1 Instrucciones del programa.

Las instrucciones del programa se dividen en dos grupos: mandos de operación y mandos para el diagnostico de fallas. Estos mandos se muestran en las tablas 3 y 4, todos los mandos tienen su sintaxis en letras mayúsculas y terminan con el paréntesis de cierre ‘)’.

Los mandos de operación son enviados por el programa de usuario para activar una de las dos lámparas. Cuando son recibidos, el programa de control ejecuta la serie de eventos para realizar la operación deseada, por ejemplo: para activar la lámpara 1 es necesario introducir la bayoneta, girar la perilla para seleccionar la lámpara y encender la fuente de voltaje.

Los mandos para el diagnostico de fallas se utilizan para ejecutar un evento en específico y supervisar su ejecución. De esta manera se puede determinar cual es la parte del sistema que no esta funcionando adecuadamente.

Tabla 3. Mandos de operación.

Mando	Descripción
LAMPARA1)	Mete bayoneta, selecciona lámpara 1 y enciende fuente.
LAMPARA2)	Mete bayoneta, selecciona lámpara 2 y enciende fuente.
FUERA_LAMPARAS)	Saca bayoneta y apaga fuentes.
ESTATUS)	Resporta el estado de la balloneta, responde "Dentro", "Fuera" o "Ningún SW activo".

Tabla 4. Mandos disponibles para el diagnostico de fallas.

Mando	Descripción
ENCIENDE1)	Enciende fuente de lámpara 1.
ENCIENDE2)	Enciende fuente de lámpara 2.
APAGA1)	Apaga fuente de lámpara 1.
APAGA2)	Apaga fuente de lámpara 2.
METE)	Mete bayoneta.
SACA)	Saca bayoneta.
SELECCIONA1)	Gira para seleccionar lámpara 1.
SELECCIONA2)	Gira para seleccionar lámpara 2.

3.3.2 Dirección IP y datos de la red.

Los datos de la red configurados en el microcontrolador RMC3700 del control de motores se muestran en la tabla 5, la dirección IP es estática y fue asignada por el departamento de computo de SPM, corresponde a la red 0 que se utiliza para el manejo de instrumentos. Los otros datos son los que se utilizan para todas las computadoras conectadas en la red 0.

Tabla 5. Configuración de red del RCM3700

IP ADRESS	192.168.0.24
NETMASK	255.255.255.0
NAMESERVER	192.168.1.1
GATEWAY	192.168.0.254
PORT	2020

3.4 Interfaz de Usuario.

El programa de interfaz de usuario fue echo en lenguaje TCL TK. Su ventana gráfica fue agregada al programa que se utiliza para manejar la cámara del ocular del instrumento (Ver figura 11). La figura 12 muestra la funcionalidad de la interfaz, si se desea que la lámpara de cobre-argón ilumine la rendija del espectrógrafo, solo hay que presionar el botón "CuAr" y esperar a que encienda el indicador visual en su costado (Ver figura 12 b). Si se desea apagar la lámpara y retirar la balloneta del camino óptico solo hay que presionar el botón etiquetado como "FUERA LAMPARAS".

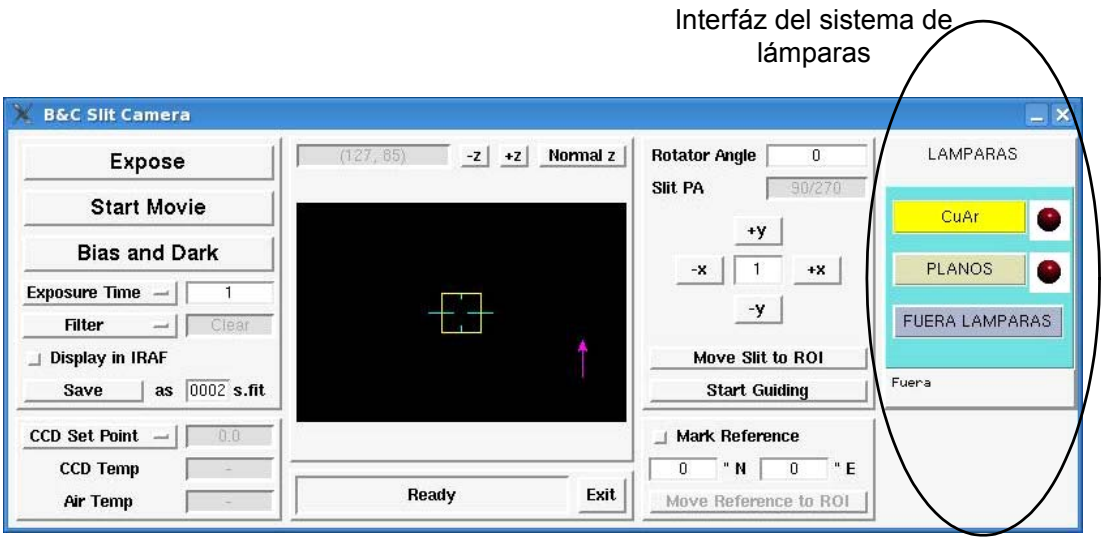


Figura 11. Incorporación de la interfáz de usuario del sistema de lámparas al programa de interfáz de la cámara del ocular.

a) b)

Figura 12. Interfaz gráfica del sistema de lámparas. a) Lámparas apagadas. b) Lámpara de CuAr encendida.

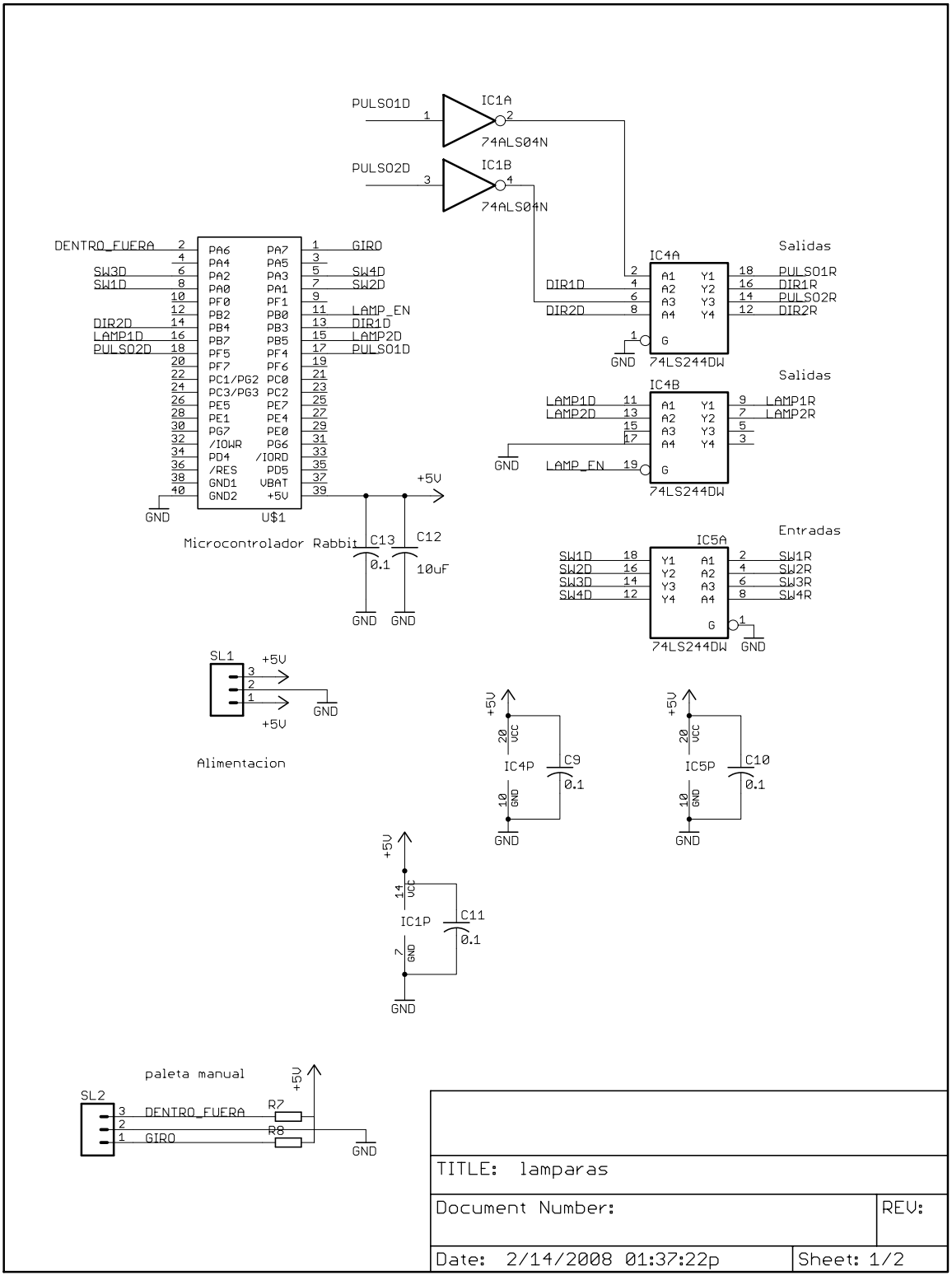
En la parte inferior de la ventana de interfáz de lámparas se ha dejado un espacio para mostrar mensajes del sistema de lámparas. Los mensajes que pueden aparecer en este espacio y su interpretación se muestran en la tabla 6.

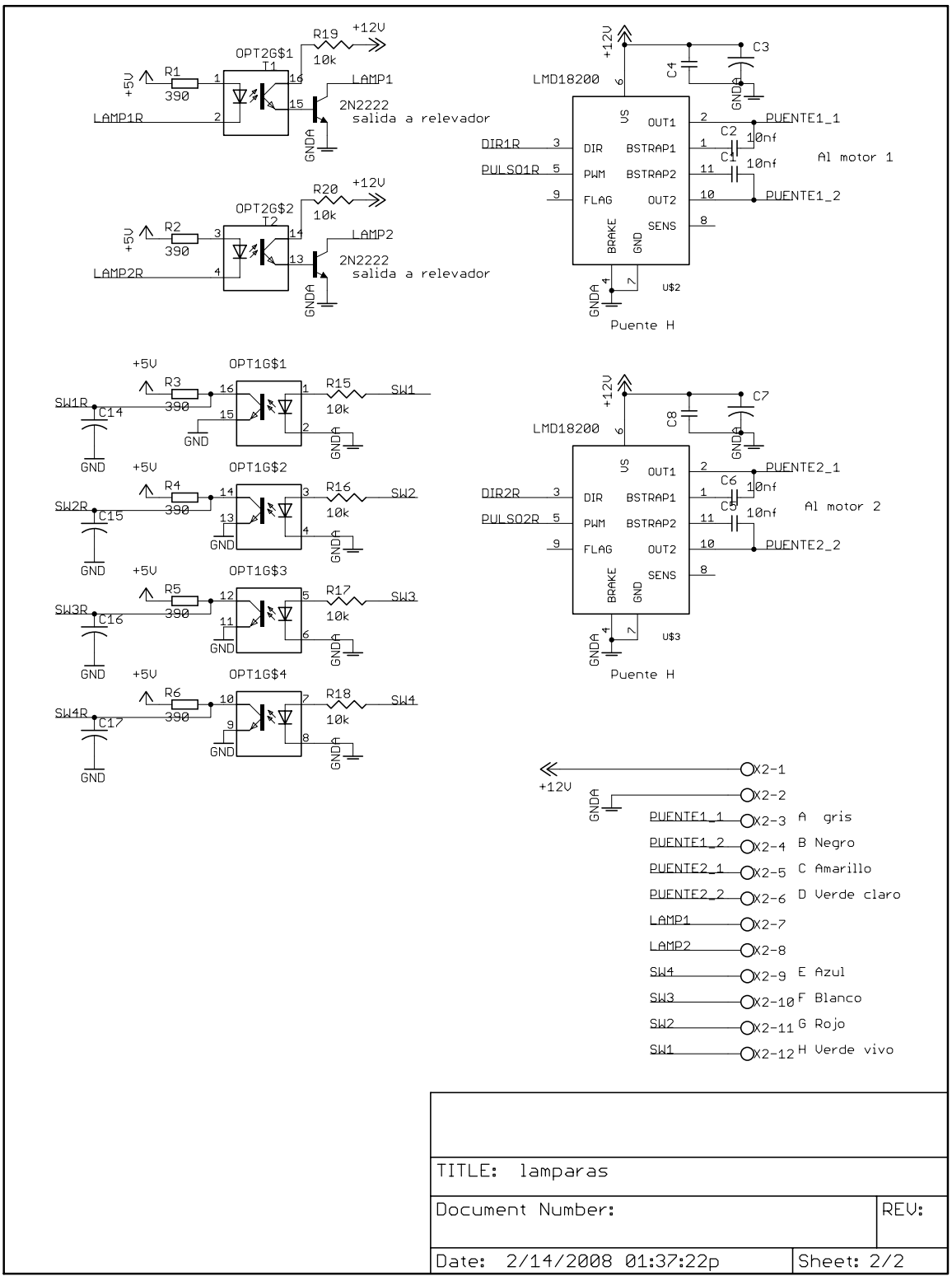
Tabla 6. Mensajes del sistema de lámparas.

Mando	Descripción
Dentro	La balloneta está dentro.
Fuera	La balloneta está fuera.
Ningún SW activo	La balloneta se encuentra en una región intermedia entre los dos límites.
Listo	La balloneta está dentro y la lámpara escogida se encuentra encendida.
No hay comunicación	No hay comunicación vía red entre la interfáz y el controlador.
No llegó SWx	Transcurrió tiempo suficiente y no se detecto el interruptor de sensado SWx. Donde x puede ser 1,2,3 o 4.

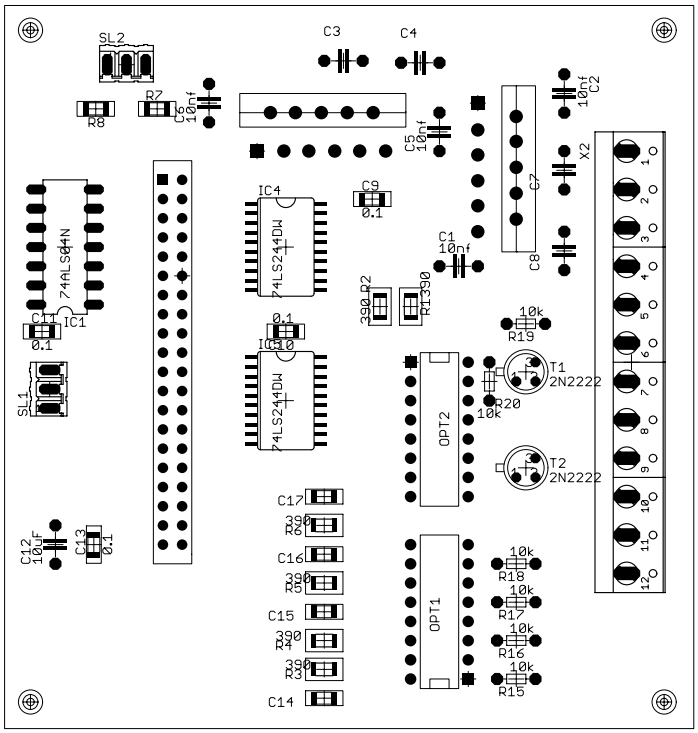
Apéndice A. Diagramas electrónicos.

A.1 Esquemático del control de motores.

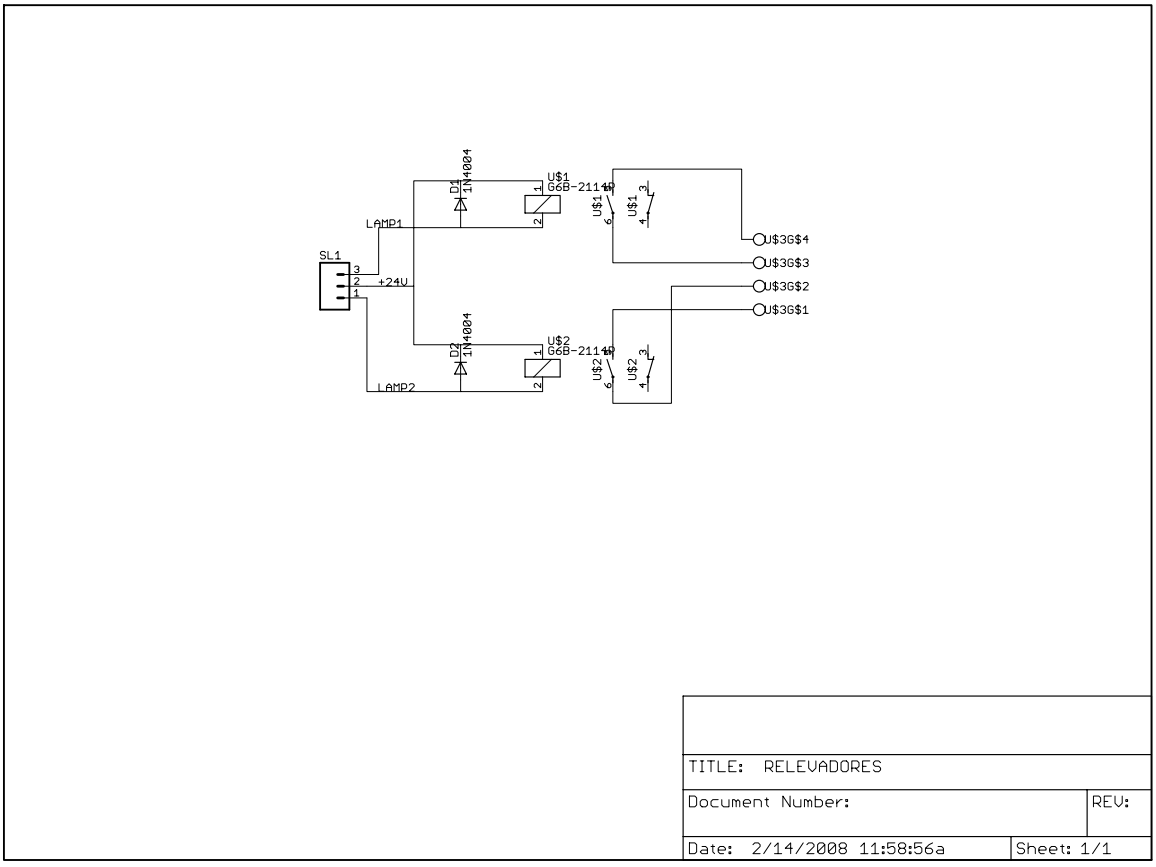



OPT2G$1
T1
R19
10k
+12V
R1
390
+5V
LAMP1
LAMP1R
2N2222
salida a relevador
GNDA
OPT2G$2
T2
R20
10k
+12V
R2
390
+5V
LAMP2
LAMP2R
2N2222
salida a relevador
GNDA
LMD18200
+12V
C3
C4
GNDA
VS
OUT1
PUENTE1_1
DIR1R
DIR
BSTRAP1
C2
10nf
PULSO1R
PWM
BSTRAP2
C1
10nf
Al motor 1
FLAG
OUT2
PUENTE1_2
BRAKE
GND
SENS
GNDA
U$2
Puente H
OPT1G$1
R3
390
+5V
R15
10k
SW1
SW1R
C14
GND
GNDA
OPT1G$2
R4
390
+5V
R16
10k
SW2
SW2R
C15
GND
GNDA
OPT1G$3
R5
390
+5V
R17
10k
SW3
SW3R
C16
GND
GNDA
OPT1G$4
R6
390
+5V
R18
10k
SW4
SW4R
C17
GND
GNDA
LMD18200
+12V
C7
C8
GNDA
VS
OUT1
PUENTE2_1
DIR2R
DIR
BSTRAP1
C6
10nf
PULSO2R
PWM
BSTRAP2
C5
10nf
Al motor 2
FLAG
OUT2
PUENTE2_2
BRAKE
GND
SENS
GNDA
U$3
Puente H
+12V
GNDA
X2-1
X2-2
PUENTE1_1
X2-3
A gris
PUENTE1_2
X2-4
B Negro
PUENTE2_1
X2-5
C Amarillo
PUENTE2_2
X2-6
D Verde claro
LAMP1
X2-7
LAMP2
X2-8
SW4
X2-9
E Azul
SW3
X2-10
F Blanco
SW2
X2-11
G Rojo
SW1
X2-12
H Verde vivo
TITLE: lamparas
Document Number:
REV:
Date: 2/14/2008 01:37:22p
Sheet: 2/2

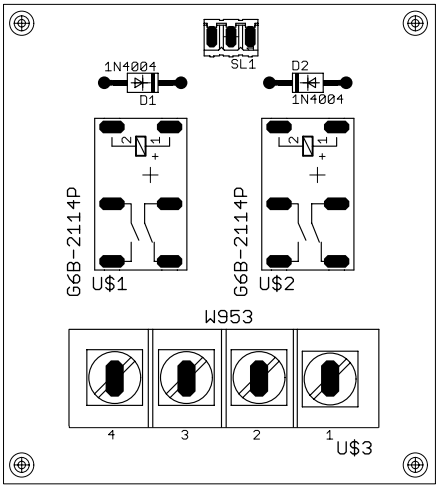
A.2 Mapa de componentes del control de motores.



A.3 Esquemático del circuito de relevadores de encendido de lámparas.



A.4 Mapa de componentes del circuito de relevadores.



Apéndice B. Programa del controlador de motores.

```
// Programa para el manejo del sistema de lamparas del
// espectrografo Boller & Chivens
// Echo por F. Murillo.
// Version 1.1, Junio de 2008.

#class auto
#define TCPCONFIG 1
#use "dcrtcp.lib"
#define PORT 2020
//Define senales de direccion.
#define DIR2_DER BitWrPortI(PBDR,&PBDRShadow,0,4)
#define DIR2_IZQ BitWrPortI(PBDR,&PBDRShadow,1,4)
#define DIR1_DER BitWrPortI(PBDR,&PBDRShadow,0,3)
#define DIR1_IZQ BitWrPortI(PBDR,&PBDRShadow,1,3)
//Define senales de salida a relevador
#define LAMP1_ON BitWrPortI(PBDR,&PBDRShadow,0,7)
#define LAMP2_ON BitWrPortI(PBDR,&PBDRShadow,0,5)
#define LAMP1_OF BitWrPortI(PBDR,&PBDRShadow,1,7)
#define LAMP2_OF BitWrPortI(PBDR,&PBDRShadow,1,5)
#define LAMP_EN  BitWrPortI(PBDR,&PBDRShadow,0,0)
#define MOTOR1_OF pwm_set(0, 1024, PWM_SPREAD)
#define MOTOR1_ON pwm_set(0, 768, PWM_SPREAD)
#define MOTOR2_OF pwm_set(1, 1024, PWM_SPREAD)
#define MOTOR2_ON pwm_set(1, 768, PWM_SPREAD)
//Define bits de interruptores de sensado
#define SW1 BitRdPortI(PADR,0)
#define SW2 BitRdPortI(PADR,1)
#define SW3 BitRdPortI(PADR,2)
#define SW4 BitRdPortI(PADR,3)
//Define bits de paleta manual
#define DENTRO_FUERA BitRdPortI(PADR,6)
#define GIRO BitRdPortI(PADR,7)

int mete,metiendo,saca,sacando,sel_lamp1,mov_lamp1,estatus;
int sel_lamp2,mov_lamp2,lampara1,lampara2,fuera_lamparas;
int dentro,lampara;  // Banderas
tcp_Socket socket;

void SecDelay(unsigned int delay)
{
  auto unsigned long done_time;
  done_time = read_rtc() + delay;
  while( (long) (read_rtc() - done_time) < 0 );
}
void inicializa()
{
  WrPortI(PBDDR,&PBDRShadow,189); // Configura bits 0,2,3,4,5 y 7 del PB como salida
  pwm_init(40000ul);
          MOTOR1_OF;
  MOTOR2_OF;
  LAMP1_OF;
  LAMP2_OF;
  LAMP_EN;
  dentro = 2;
}
```

```
void limpia_banderas()
{
  mete=0;
  metiendo=0;
  saca=0;
  sacando=0;
  sel_lamp1=0;
  mov_lamp1=0;
  sel_lamp2=0;
  mov_lamp2=0;
  lampara1=0;
  lampara2=0;
  estatus=0;
  fuera_lamparas=0;
}

void procesa_mando(char *buffer2)
{
  if(strstr(buffer2,"LAMPARA1")) {
        lampara1 = 1;
  }
  if(strstr(buffer2,"LAMPARA2")) {
        lampara2 = 1;
  }
  if(strstr(buffer2,"FUERA_LAMPARAS")) {
        fuera_lamparas = 1;
  }
  if(strstr(buffer2,"ENCIENDE1")) {
        LAMP1_ON;
  }
  if(strstr(buffer2,"ENCIENDE2")) {
        LAMP2_ON;
  }
  if(strstr(buffer2,"APAGA1")) {
        LAMP1_OF;
  }
  if(strstr(buffer2,"APAGA2")) {
        LAMP2_OF;
  }
  if(strstr(buffer2,"METE")) {
        mete=1;
  }
  if(strstr(buffer2,"SACA")) {
        saca=1;
  }
  if(strstr(buffer2,"SELECCIONA1")) {
        sel_lamp1=1;
  }
  if(strstr(buffer2,"SELECCIONA2")) {
        sel_lamp2=1;
  }
  if(strstr(buffer2,"ESTATUS")) {
        estatus=1;
  }
}
```

```
void main()
{
  int   bytes_read,bytes_suma,sw;
  char  buffer[50],buffer2[200],mensaje[100];
  auto unsigned long time_start1,time_start2;
  sock_init();
  inicializa();
  limpia_banderas();

  while(1) {
                costate { //Costate Socket
                                tcp_listen(&socket,PORT,0,0,NULL,0);
                                printf("Waiting for connection...\n");
                while(!sock_established(&socket) && sock_bytesready(&socket)==-1) {
                tcp_tick(NULL);
          yield;
          }
                printf("Connection received...\n");
                bytes_suma=0;
                buffer2[0]=0;
                do {
                bytes_read=sock_fastread(&socket,buffer,sizeof(buffer)-1);
                        bytes_suma = bytes_suma + bytes_read;
                buffer[bytes_read]=0;
                strcat(buffer2,buffer);
                if(strchr(buffer2,')')) {
                printf("%s\n",buffer2);
                procesa_mando(buffer2);
                //strcat(buffer2,">");
                //sock_write(&socket,buffer2,strlen(buffer2));
                strcpy(buffer2,"");
                        }
          yield;
          }while(tcp_tick(&socket));
         waitfor(DelayMs(10));
      } // Costate socket
```

```
costate { // Costate secuencias
        if(lampara1) {
          if(!SW1) { //si la balloneta esta dentro
                    if(!SW3) {  // Si la lampara esta en posicion
                    lampara1=0; // Desactiva secuencia
                    LAMP1_ON;
                sprintf(mensaje,"%s\n","Listo");
                if(sock_established(&socket)) {
                            sock_write(&socket,mensaje,strlen(mensaje));
                    sock_flush(&socket);
                            sock_close(&socket);
                    printf("Cerre conexion \n");
                    }
              } else {  // Si no esta en posicion
                    if (!mov_lamp1 && !mov_lamp2) { //Si no hay movimiento en proceso
                    sel_lamp1=1;
                 }
                }
           } else { // si la balloneta no esta dentro
                     if (!sacando && !metiendo) { // Si no hay movimiento en proceso
                    mete=1;
                 }
              }
          }
        if(lampara2) {
          if(!SW1) { //si la balloneta esta dentro
                    if(!SW4) {  // Si la lampara esta en posicion
                    lampara2=0; // Desactiva secuencia
                    LAMP2_ON;
                sprintf(mensaje,"%s\n","Listo");
                if(sock_established(&socket)) {
                            sock_write(&socket,mensaje,strlen(mensaje));
                    sock_flush(&socket);
                            sock_close(&socket);
                    printf("Cerre conexion \n");
                    }
              } else {  // Si no esta en posicion
                    if (!mov_lamp1 && !mov_lamp2) { //Si no hay movimiento en proceso
                    sel_lamp2=1;
                 }
                }
           } else { // si la balloneta no esta dentro
                     if (!sacando && !metiendo) { // Si no hay movimiento en proceso
                    mete=1;
                 }
              }
          }

          if(fuera_lamparas) {
                    LAMP1_OF;
                    LAMP2_OF;
                                                        saca=1;
                    fuera_lamparas = 0;
          }
                    waitfor(DelayMs(100));
        } // Costate Secuencias
```

```
costate {  // Costate eventos
  if(mete) {
       MOTOR1_OF;
              SecDelay(1);
       DIR1_DER;
       MOTOR1_ON;
       metiendo=1;    // Arranca proceso
       mete=0;
       time_start1 = read_rtc();
   }

   if(metiendo) {
       if(!SW1 || ( read_rtc() - time_start1 ) > 5) { // cinco segundos de timeout
            MOTOR1_OF;
            metiendo=0; // Para proceso
            dentro=1;
            if( ( read_rtc() - time_start1 ) > 5 ) {
                printf("Tiempo transcurrido \n");
              sprintf(mensaje,"%s\n","No llego SW1");
              lampara1=0; // Si ocurre un error detiene secuencias
              lampara2=0;
              if(sock_established(&socket)) {
                           sock_write(&socket,mensaje,strlen(mensaje));
               sock_flush(&socket);
               sock_close(&socket);
               printf("Cerre conexion \n"); // Si hay un socket abierto se cierra la conexion
               }
            } else {
                printf("Adentro \n");
            }
               }
   }

  if(saca) {
       MOTOR1_OF;
              SecDelay(1);
       DIR1_IZQ;
       MOTOR1_ON;
       sacando=1;    // Arranca proceso
       saca=0;
       time_start1 = read_rtc();
   }
```

```
if(sacando) {
     if(!SW2 || ( read_rtc() - time_start1 ) > 5) {
          MOTOR1_OF;
          sacando=0; // Para proceso
          dentro =0;
          if( ( read_rtc() - time_start1 ) > 5 ) {
            sprintf(mensaje,"%s\n","No llego SW2");
             printf("Tiempo transcurrido \n");
          } else {
             sprintf(mensaje,"%s\n","Fuera");
             printf("Afuera \n");
          }
          if(sock_established(&socket)) {
                                   sock_write(&socket,mensaje,strlen(mensaje));
                        sock_flush(&socket);
                                   sock_close(&socket);
                        printf("Cerre conexion \n");
                        }
             }
}

if(sel_lamp1) {
     MOTOR2_OF;
             SecDelay(1);
     DIR2_DER;
     MOTOR2_ON;
     mov_lamp1=1;    // Arranca proceso
     sel_lamp1=0;
     time_start2 = read_rtc();
}

 if(mov_lamp1) {
     if(!SW3 || ( read_rtc() - time_start2 ) > 5) {
          MOTOR2_OF;
          mov_lamp1=0; // Para proceso
          if(( read_rtc() - time_start2 ) > 5) {
             printf("Tiempo transcurrido \n");
            sprintf(mensaje,"%s\n","No llego SW3");
            lampara1=0; // Si ocurre un error detiene secuencias
            lampara2=0;
            if(sock_established(&socket)) {
                        sock_write(&socket,mensaje,strlen(mensaje));
             sock_flush(&socket);
             sock_close(&socket);
             printf("Cerre conexion \n"); // Si hay un socket abierto se cierra la conexion
             }
          } else {
             lampara = 1;
                    printf("lampara1 \n");
          }
      }
}
```

```
if(sel_lamp2) {
      MOTOR2_OF;
            SecDelay(1);
      DIR2_IZQ;
      MOTOR2_ON;
      mov_lamp2=1;    // Arranca proceso
      sel_lamp2=0;
      time_start2 = read_rtc();
 }

 if(mov_lamp2) {
      if(!SW4 || ( read_rtc() - time_start2 ) > 5) {
           MOTOR2_OF;
           mov_lamp2=0; // Para proceso
           if(( read_rtc() - time_start2 ) > 5) {
              printf("Tiempo transcurrido \n");
             sprintf(mensaje,"%s\n","No llego SW4");
             lampara1=0; // Si ocurre un error detiene secuencias
             lampara2=0;
             if(sock_established(&socket)) {
                        sock_write(&socket,mensaje,strlen(mensaje));
              sock_flush(&socket);
              sock_close(&socket);
              printf("Cerre conexion \n"); // Si hay un socket abierto se cierra la conexion
              }
           } else {
              lampara = 2;
              printf("lampara2 \n");
           }
              }
 }

 if(!DENTRO_FUERA && !sacando && !metiendo){  // Si se presiono el boton mete saca
   if(!SW2) { // Si esta fuera
              mete = 1;
   } else if (!SW1) { //Si esta dentro
              saca = 1;
   } else { // Lo saca por default
              saca = 1;
   }
 }

 if(!GIRO && !mov_lamp1 && !mov_lamp2){  // Si se presiono el boton de giro
   printf("Se presiono el boton de giro \n");
   if(!SW4) {
              sel_lamp1 = 1;
   } else if(!SW3) {
              sel_lamp2 = 1;
   } else {
              sel_lamp1 = 1;
   }
 }
```

```
        if(estatus) {
          if(!SW1) {
                    sprintf(mensaje,"%s\n","Dentro");
          } else if(!SW2) {
                    sprintf(mensaje,"%s\n","Fuera");
          } else {
            sprintf(mensaje,"%s\n","Nigun SW activo");
          }
          if(sock_established(&socket)) {
                    sock_write(&socket,mensaje,strlen(mensaje));
            sock_flush(&socket);
            sock_close(&socket);
          }
        }
        estatus = 0;
        waitfor(DelayMs(100));
      }  //Costate eventos

      costate {  // Costate eventos
          if(!GIRO){  // Si se presiono el boton de giro
             printf("Presionaste giro \n");
          }
          if(!DENTRO_FUERA) {
            printf("Presionaste dentro fuera \n");
          }
          if(!SW1) {
            printf("Presionaste SW1 \n");
          }
          if(!SW2) {
            printf("Presionaste SW2 \n");
          }
          if(!SW3) {
            printf("Presionaste SW3 \n");
          }
          if(!SW4) {
            printf("Presionaste SW4 \n");
          }
          waitfor(DelayMs(500));
      } // Costate pinta status

          } // While
}
```