

23 jun 02 19:42	gterm.c	Página 1/6
<pre> /***** /* gterm.c (c) Juan Gonzalez Gomez. Junio 2002 */ /*-----*/ /* Licencia GPL. */ /***** #include <gnome.h> #include <glade/glade.h> #include "interface.h" #include "serie.h" /*-----*/ /* CONSTANTS */ /*-----*/ #define VERSION "0.1" /*-----*/ /* PROTOTYPES */ /*-----*/ void univ_update_particulas(gboolean futuro); /*-----*/ /* VARIABLES */ /*-----*/ interface_t interface; dialog_t windialog; dialog2_t windialog2; int serial_fd; /* Descriptor del fichero del puerto serie */ gint pos; /* Posicion en el terminal */ /* -- main timer -- */ static int main_timer; /*-- Variables for GNOME */ static gchar app_id[] = {"GTerm"}; /***** /* Parametros de usuario */ /***** static gboolean flag; /* User parameters description */ struct poptOption options[]={ {"flag", 'h', POPT_ARG_NONE, &flag, 0, "Activar la opcion Flag", NULL}, {NULL, '\0', 0, NULL, 0, NULL, NULL} }; /*-----*/ /* CALLBACK FUNCTIONS */ /*-----*/ void main_quit(GtkWidget *window, gpointer data) { gtk_timeout_remove(main_timer); cerrar_puerto_serie(); gtk_main_quit(); } void help_about (GtkMenuItem *menuitem, gpointer user_data) </pre>		

23 jun 02 19:42	gterm.c	Página 2/6
<pre> { GtkWidget *about = NULL; about = create_about(); gtk_widget_show (about); } static void remove_cr(gchar *cad, gint length) /***** /* Eliminar los caracteres <CR> devueltos por el model */ /***** { gint i; for (i=0; i<length; i++) { if (cad[i]=='\r') cad[i]=' '; } } static gboolean main_timer_cb(gpointer data) /***** /* Temporizador principal para la lectura de datos por */ /* el puerto serie */ /***** { gchar resp[80]; gint n; /* Si hay datos pendientes... */ if (car_waiting()) { n=read(serial_fd, resp, 80); remove_cr(resp, n); gtk_editable_insert_text(GTK_EDITABLE(interface.terminal), resp, n, &pos); } return TRUE; } static void enviar_cmd(const gchar *car_cmd) /***** /* Enviar un comando al modem */ /***** { GString *cmd; cmd=g_string_new(car_cmd); /* Añadir caracter CR antes de enviar al modem */ g_string_append_c(cmd, '\r'); /* Enviar comando al modem */ enviar_bloque(cmd->str, cmd->len); g_print("Comando:%s\n", cmd->str); g_string_free(cmd, TRUE); } void entry_cmd(GtkWidget *widget, gpointer data) /***** /* Funcion de retollamada de un comando introducido */ /***** </pre>		

23 jun 02 19:42	gterm.c	Página 3/6
<pre> { GString *cmd; /* Obtener el comando */ cmd=g_string_new(gtk_entry_get_text(GTK_ENTRY(widget))); /* Borrar el comando tecleado de la entrada de datos */ gtk_entry_set_text(GTK_ENTRY(widget),""); g_print ("Comando:%s\n",cmd->str); /* Añadir caracter CR antes de enviar al modem */ g_string_append_c(cmd,'\r'); /* Enviar comando al modem */ enviar_bloque(cmd->str,cmd->len); g_string_free(cmd,TRUE); } void dialog2(GtkWidget *GtkWidget, gpointer data) /***** /* Funcion de retrollamada de los botones de la */ /* caja de dialog de introduccion del numero de */ /* mensaje */ *****/ { int n; GString *cmd; /* Obtener el numero de boton */ n=atoi((char *)data); switch(n) { case 1: /* Boton OK, borrar mensaje */ g_print("OK\n"); cmd=g_string_new(""); g_string_sprintf(cmd,"AT+CMGD=%s",gtk_editable_get_chars(GTK_EDITABLE(windialog2.entry_nm),0,-1)); enviar_cmd(cmd->str); g_print("Comando: %s\n",cmd->str); gtk_widget_destroy(windialog2.dialog); g_string_free(cmd,TRUE); break; case 2: /* Boton cancel, borrar mensaje */ g_print("Cancel\n"); gtk_widget_destroy(windialog2.dialog); break; case 3: /* Boton OK, leer mensaje */ g_print("OK\n"); cmd=g_string_new(""); g_string_sprintf(cmd,"AT+CMGR=%s",gtk_editable_get_chars(GTK_EDITABLE(windialog2.entry_nm),0,-1)); enviar_cmd(cmd->str); g_print("Comando: %s\n",cmd->str); gtk_widget_destroy(windialog2.dialog); g_string_free(cmd,TRUE); break; case 4: /* Boton cancel, leer mensaje */ gtk_widget_destroy(windialog2.dialog); break; } } </pre>		

23 jun 02 19:42	gterm.c	Página 4/6
<pre> } void dialog_sms_button(GtkWidget *widget, gpointer data) /***** /* Funcion de retrollamada de los botones de la */ /* caja de dialog de envio de mensajes SMS */ *****/ { int n; GString *trama; gchar cz[2]; trama=g_string_new(""); /* Obtener el numero de boton */ n=atoi((char *)data); switch(n) { case 1: /* Boton OK */ g_print("OK\n"); /* Obtener el número de telefono */ g_string_sprintf(trama,"AT+CMGS=\"%s\n\"%s", gtk_entry_get_text(GTK_ENTRY(windialog.entry_tlf)), gtk_editable_get_chars(GTK_EDITABLE(windialog.editable_sms), 0,-1)); g_print("COMANDO: %s\n",trama->str); /* Enviar comando al modem */ enviar_bloque(trama->str,trama->len); cz[0]=26; enviar_bloque(cz,1); gtk_widget_destroy(windialog.dialog); break; case 2: /* Boton cancel */ g_print("Cancel\n"); gtk_widget_destroy(windialog.dialog); break; } } void main_button(GtkWidget *widget,gpointer data) /***** /* Funcion de retrollamada de los botones */ *****/ { int n; gchar cz[2]; /* Obtener el numero del boton */ n=atoi((char *)data); switch(n) { case 1: /* Limpiar el terminal */ pos=0; gtk_editable_delete_text(GTK_EDITABLE(interface.terminal),0,-1); gnome_appbar_set_status(GNOME_APPBAR(interface.app_bar), "Terminal limpiado"); break; case 2: /* Listar telefonos */ enviar_cmd("AT+CPBR=1,100"); </pre>		

23 jun 02 19:42

gterm.c

Página 5/6

```

    break;
case 3: /* Listar los mensajes cortos */
    enviar_cmd("AT+CMGL=\"ALL\"");
    break;
case 4: /* Leer un sms */
    create_dialog2(&windialog2);
    gtk_signal_connect(GTK_OBJECT(windialog2.boton_ok), "clicked",
        GTK_SIGNAL_FUNC(dialog2,"3");
    gtk_signal_connect(GTK_OBJECT(windialog2.boton_cancel), "clicked",
        GTK_SIGNAL_FUNC(dialog2,"4");
    gtk_widget_show(windialog2.dialog);
    break;
case 5: /* Borrar un mensaje */
    g_print("Borrar mensaje\n");
    create_dialog2(&windialog2);
    /* Conectar las señales de OK y Cancel */

    /*-- OK --*/
    gtk_signal_connect(GTK_OBJECT(windialog2.boton_ok), "clicked",
        GTK_SIGNAL_FUNC(dialog2,"1");

    /*-- Cancel --*/
    gtk_signal_connect(GTK_OBJECT(windialog2.boton_cancel), "clicked",
        GTK_SIGNAL_FUNC(dialog2,"2");

    gtk_widget_show(windialog2.dialog);
    break;
case 6: /* Enviar un sms */
    g_print("Enviar SMS\n");
    create_dialog_send_sms(&windialog);
    gtk_widget_show(windialog.dialog);
    break;
case 7: /* Enviar control-z */
    cz[0]=26;
    enviar_bloque(cz,1);
    break;
default:
    g_print("Boton no implementado\n");
}
}

/*-----*/
/*      P R O G R A M A      P R I N C I P A L      */
/*-----*/
gint main (gint argc, gchar *argv[])
{
    poptContext contexto;

    /* Initialize GNOME, using user parameters */
    gnome_init_with_popt_table (app_id, VERSION, argc,argv,
        options, 0, &contexto);
    poptFreeContext (contexto);

    /*-- Comprobar parametros de usuario --*/
    if (flag) g_print ("Opcion Flag...\n");

    /*-- Init the Glade library -- */
    glade_gnome_init ();

    /*-----*/
    /*  Crear el interfaz      */

```

23 jun 02 19:42

gterm.c

Página 6/6

```

/*-----*/
create_app(&interface);
pos=0;

/* Temporizador principal, para lectura de los */
/* datos que vienen por el puerto serie */
main_timer=gtk_timeout_add(400, main_timer_cb, NULL);

/*-----*/
/* Inicializar el puerto serie */
/*-----*/
if (abrir_puerto_serie(COM1)==0) {
    printf ("Error al abrir puerto serie: %s\n",getserial_error());
    exit(1);
}
baudios(9600);
serial_fd=getserial_fd();

/*-- Que comience la fiesta!!!! -- */
gtk_main();

return 0;
}

```