

# Desarrollo de aplicaciones embedded usando GTK+ y GPE

Ariel Rios

GPE Project  
ariel@gnu.org

**Resumen** El objetivo de este trabajo es dar una introducción al desarrollo de aplicaciones para plataformas embedded usando las tecnologías proporcionadas por GPE y GTK como guía de referencia para el tutorial impartido del mismo nombre.

## 1. Introducción

Primero se da un repaso a las diversas opciones de distribuciones de Linux y plataformas de hardware para después pasar a una instalación de la distribución Familiar Linux para las Ipaq así como del ambiente de desarrollo necesario para realizar aplicaciones. Después se pasa a explicar la forma correcta de compilar programas para la arquitectura arm así como problemas comunes que se encuentran al hacer la cross compilación y como solucionarlos. Finalmente, se da una introducción al desarrollo usando las librerías de GPE , GTK+ para realizar aplicaciones multimedia así como recomendaciones que se deben seguir al crear aplicaciones.

## 2. Linux en plataformas embedded

Linux se ha convertido en la opción más viable en sistemas operativos para servidores y lentamente también se ha convertido para entornos de escritorio gracias a proyectos como GNOME. Ahora, Linux está siendo usado también como opción para sistemas operativos en plataformas embedded para productos comerciales de diversas compañías que incluyen la línea completa Zaurus de Sharp, algunos teléfonos móviles de Motorola, Datang , Panasonic et al.

Al igual que en el escritorio donde existen diversos proyectos de interfaz de usuario que incluyen a GNOME y KDE, simil modo existen proyectos basados en GTK+ y QT para estas arquitecturas.

Troll Tech, la compañía creadora de QT, comercializa QTopia , un ambiente de escritorio para handhelds y móviles que se encuentra con la mejora posición en el mercado mientras que GPE la alternativa basada en GTK+ apenas comienza a ser usada comercialmente por lo que es necesario la aparición de programadores deseos de contribuir en GPE.

### 3. Instalación de Linux en la ipaq

La instalación de Linux en una ipaq es un proceso sumamente sencillo pero que debe ser seguido al pie de la letra y con cuidado para evitar convertir nuestra ipaq en un bonito ladrillo sin mayor utilidad. De inicio debemos contar al menos con alguna de estas ipaqs:

- Una Ipaq de la serie H3600, H3700, H3800, H3900, H5100, H5400 y H5500,
- Una Ipaq de las series H51xx, H54xx, H55xx. (estas solamente soportan la instalación vía serial)

Además, será necesario contar también con:

- Tarjeta de Memoria Flash y/o
- Una computadora corriendo una terminal conectada a la ipaq via puerto serial.

Se selecciona los archivos deseados para el hardware (elegir GPE) en la siguiente dirección:

<http://familiar.handhelds.org/releases/v0.8.1/install/download.html>

Una vez que se tienen estos archivos el siguiente paso es instalar el bootloader para lo cual existen dos opciones, usando una conexión serial o instalar usando una tarjeta Compact Flash. Los pasos a seguir para la última son:

- Copiar reflash.ctl , md5sums y los archivos de tipo jffs2 al directorio raíz de la tarjeta cuidando que ésta esté formateada con vfat.
- Insertar la tarjeta en la ipaq.
- Apretar para abajo en el joypad y aprietar el boton de reset ubicado en la parte trasera de la unidad.
- Al reiniciar la unidad verificar que la versión del bootloader es al menos la 2.21.12 .
- Presionar el botón de grabación (ubicado a un costado de la unidad) para que se muestre el menú de opciones de imágenes a ser seleccionadas.
- Una vez elegida apretar la parte central del joypad seguida de boton de grabación .
- Esperar pacientemente pues este proceso puede tardar varios minutos.
- Reiniciar la unidad apretando la parte central del joypad.

La ipaq cuenta con una cuenta de root con password vacío.

### 4. Instalación de Open Embedded

Open Embedded (OE) es un ambiente de desarrollo que permite apuntar hacia una gran variedad de devices soportando múltiples compilaciones, versiones y configuraciones. Soporta varias arquitecturas y múltiples versiones para estas e incluye todas las herramientas para poder realizar la cross compilation.

Para poder hacer uso de estas facilidades se deben seguir los siguientes pasos:

-Obtención de bitbake , la herramienta de compilación:  
 svn co svn://svn.berlios.de/bitbake/trunk/bitbake  
 Configurar su PATH para que las herramientas sean accesibles para correrlo sin instalarlo o bien instalarlo de esta manera:  
 ./setup.py install --prefix=/usr/local --install-data=/usr/local/share  
 -Obtener open embedded:  
 bk clone bk://openembedded.bkbits.net/openembedded  
 -Descomprimir el archivo.  
 tar xjf ../openembedded-exported-2005-03-01-0700.tar.bz2  
 -Crear la configuración local tomando como ejemplo el archive incluido en openembedded/conf/local.conf.sample  
 -Agregar una variable BBPATH al ambiente indicando el lugar donde se encuentra bitbaker.  
 -Compilar paquetes de la siguiente forma:  
 -bitbake nano (crea un solo paquete)  
 -bitbake gpe-image (crea una imagen basada en gpe)  
 -bitbake world (crea paquetes para todo lo q se encuentre en el repositorio)

## 5. Compilando programas

En ocasiones debemos compilar paquetes que no se encuentran dentro de nuestro ambiente OE por múltiples razones que incluyen el no poder usarlo.

Normalmente, parecería muy sencillo compilar un programa. Cuando compilamos para nuestros cajas de Linux basta con el conocido configure y make y quizás sin más opciones que darle el prefijo para la configuración. Se corren los scripts y regularmente todo funciona. Desgraciadamente, al compilar para otra arquitectura esto no siempre sucede con frecuencia y muchas veces el configure falla miserablemente. Cuando el pánico nos inunda los siguientes consejos son útiles:

- Asegurarse que pkg-config apunte a los archivos pc del device y no a los del sistema x86.
- Correr configure con las banderas de host, prefix apuntando hacia donde se encuentren las librerías de cross compiling:
- Asegurarse que las librerías de X que tome sean igualmente las del dispositivo y no las del x86:

```
PKG_CONFIG_PATH=/opt/arm/lib/pkgconfig \  

./configure --prefix=/opt/arm/ --host=arm-linux \  

--x-includes=/opt/arm/include/X \  

--x-libs=/opt/arm/lib
```

- Un error común de los scripts de configuración es que intentan hacer pruebas de ciertas opciones en las librerías compilando y ejecutando programas, cosas que no podemos realizar con la compilación cruzada. En esos casos eliminar dichas pruebas del configure y correrlo. En la mayoría de los casos no hay problema.

- Instalar usando `make install-strip` para reducir el tamaño de la aplicación.

## 6. Agregando paquetes a Open Embedded

Si ya pudimos compilar algún programa y deseamos agregarlo a nuestro árbol de OE se deben seguir los siguientes pasos:

-Creación del archivo que provee la información necesaria de un paquete como manera de compilarse, donde se debe instalar, etc.

-Usar la convención:

`nombre_del_paquete-version`

-Se pueden usar rutinas para realizar los paquetes. Note que debe indicar su uso previamente usando `inherit`.

-Una forma rutinaria de hacer un paquete sería en este orden:

- `do_fetch`
- `do_unpack`
- `do_patch`
- `do_configure`
- `do_compile`
- `do_stage`
- `do_install`
- `do_package`

Existen además varias secciones donde se pueden incluir los paquetes. Normalmente, el desarrollo para gpe se llevará a cabo bajo gpe.

## 7. Usando las librerías de GPE para desarrollar

Gpe nos da varios widgets que sirven para realizar aplicaciones. En algunos casos, estos widgets son usados en substitución de los de GTK+ y en otros, proveen widgets con funcionalidad de los encontrados en `gnome-libs`. Se presenta a continuación una lista con algunos de los widgets más usuales.

### 7.1. Widgets específicos de GPE

**GtkDateCombo** Selección de fecha mediante un combo. La fecha tiene un formato dependiente del locale usado en la computadora. Provee además un calendario que se utiliza para poder cambiar los valores una una manera sencilla además de que puede ser editado directamente en el widget.

```
GtkWidget* gtk_date_combo_new          (void);
void        gtk_date_combo_set_date    (GtkDateCombo*,
                                        guint year,
```

```

                                guint month,
                                guint day);
void      gtk_date_combo_clear      (GtkDateCombo *dp);
void      gtk_date_combo_week_starts_monday
                                (GtkDateCombo*,
                                gboolean);
void      gtk_date_combo_ignore_year (GtkDateCombo*,
                                gboolean);

```

**GtkSimpleMenu** Provee una implementación sencilla de selección de menús con una interface igual de sencilla de usar.

```

GtkWidget*  gtk_simple_menu_new      (void);
void        gtk_simple_menu_append_item (GtkSimpleMenu *sel,
                                const gchar *item);
void        gtk_simple_menu_flush    (GtkSimpleMenu *sel);

```

**GtkTimeSel** Widget usado para escribir valores de tiempo. Incluye un campo de edición así como un botón que abre una ventana para selección del tiempo.

```

GtkWidget*  gpe_time_sel_new      ();
void        gpe_time_sel_get_time (GpeTimeSel *sel,
                                guint *hour,
                                guint *minute);
void        gpe_time_sel_set_time (GpeTimeSel *sel,
                                guint hour,
                                guint minute);

```

**gpe icon** Provee un conjunto de métodos para mantener íconos compartidos y que se evite la carga de conjuntos de íconos varias veces, guardándolos dentro de una lista local e identificados por una cadena.

```

GdkPixbuf*  gpe_find_icon          (const char *name);
GdkPixbuf*  gpe_find_icon_scaled   (const char *name,
                                GtkIconSize size);
GdkPixbuf*  gpe_try_find_icon      (const char *name,
                                gchar **error);
gboolean     gpe_find_icon_pixmap   (const char *name,
                                GdkPixmap **pixmap,
                                GdkBitmap **bitmap);
void        gpe_set_window_icon     (GtkWidget *window,
                                gchar *icon);

```

**DirBrowser** Provee una manera sencilla de navegar a través de directorios.

```
GtkWidget* gpe_create_dir_browser (gchar *title,  
                                   gchar *current_path,  
                                   GtkSelectionMode mode,  
                                   void (*handler) (gchar *));
```

**TrayTools** No un widget en sí sino una serie de funciones que permiten el manejo de aplicaciones que están contenidos dentro de un panel o área de notificación ya sea un panel de Gnome o de Matchbox.

```
void gpe_system_tray_dock (GdkWindow *window);  
guint gpe_system_tray_send_message (GdkWindow *window,  
                                     const gchar *text,  
                                     unsigned int timeout);  
void gpe_system_tray_cancel_message (GdkWindow *window,  
                                     guint id);
```

**GpeButton** Widget para la creación de botones cuyo tamaño se ajuste a la de la pequeña pantalla. .

```
GtkWidget * gpe_picture_button_aligned (GtkStyle *style,  
                                       gchar *text,  
                                       gchar *icon,  
                                       GpePositionType pos);  
GtkWidget *gpe_picture_button(GtkStyle *style, gchar *text,  
gchar *icon);  
GtkWidget *gpe_button_new_from_stock  
    (const gchar *stock_id, int type);
```

## 7.2. Métodos de aplicaciones GPE

```
gboolean gpe_application_init (int *argc,  
                               char **argv[]);  
void gpe_saved_args (gint *argc,  
                    gchar **argv[]);  
gboolean gpe_stylus_mode (void);
```

## 8. Ejemplo de Aplicación

Siobhan es un reproductor de archivos multimedia creado con la sola intención de ser usado en plataformas embebidas. Se presenta una parte del código

donde se ilustra el cambio de algunas funciones de GTK+ por sus equivalentes en GPE.

```
/* Siobhan Audio Player
 *
 * (c) 2005 Ariel Rios <ariel@gnu.org>
 *
 * main.c: main program
 *
 * This file is part of Siobhan Audio Player.
 *
 * Siobhan is free software; you can redistribute it
 * and/or modify it under the terms of the GNU General
 * Public License as published by the Free Software
 * Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * Siobhan is distributed in the hope that it will
 * be useful, but WITHOUT ANY WARRANTY; without even
 * the implied warranty of MERCHANTABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU General
 * Public License for more details.
 *
 * You should have received a copy of the GNU General
 * Public License along with Foobar; if not, write to
 * the Free Software Foundation, Inc., 59 Temple Place,
 * Suite 330, Boston, MA 02111-1307 USA
 *
 */

#ifdef HAVE_GPE
#include <gpe/init.h>
#endif

#include <gst/gst.h>
#include <gtk/gtk.h>

#include "timer.h"
#include "callback.h"

#define GST_PLAYER_ERROR (gst_player_error_quark ())

static GQuark
gst_player_error_quark ()
```

```

{
    static GQuark qk = 0;
    if (!qk)
        qk = g_quark_from_static_string ("gst-player-error-quark");

    return qk;
}

int
main (int argc, char *argv [])
{
    GtkWidget *app;
    GtkWidget *button_play, *stop, *pause, *properties;
    GtkWidget *hbox, *vbox;
    GtkWidget *filesel;
    GtkWidget *slider;
    GtkWidget *video;

    GstElement *filesrc, *decoder, *audiosink;
    GstElement *play;

    CallbackData *data = g_new (CallbackData, 1);
    GError *err = NULL;

    gst_init (&argc, &argv);

#ifdef HAVE_GPE
    gtk_init (&argc, &argv);
#else
    gpe_init (&argc, &argv);
#endif

    play = gst_element_factory_make ("playbin", "play");

    data->thread = play;
    data->filesrc = filesrc;
    data->decoder = decoder;

    audiosink = gst_element_factory_make
        ("ximagesink", "video-sink");

```

```

        g_object_set (play, "video-sink", audiosink, NULL);
        g_object_set (G_OBJECT (play), "uri", argv[1], NULL);

        g_signal_connect
(G_OBJECT (play), "eos", G_CALLBACK (eos), play);

        app = gtk_window_new (GTK_WINDOW_TOPLEVEL);

#ifdef HAVE_GPE
        button_play = gpe_button_new_from_stock
(GTK_STOCK_MEDIA_PLAY, GPE_BUTTON_TYPE_ICON);
        pause = gpe_button_new_from_stock
(GTK_STOCK_MEDIA_PAUSE, GPE_BUTTON_TYPE_ICON);
        stop = gpe_button_new_from_stock
(GTK_STOCK_MEDIA_STOP, GPE_BUTTON_TYPE_ICON);
#else
#ifdef HAVE_GTK_2_6
        button_play = gtk_button_new_from_stock
                (GTK_STOCK_MEDIA_PLAY);
        pause = gtk_button_new_from_stock
                (GTK_STOCK_MEDIA_PAUSE);
        stop = gtk_button_new_from_stock
                (GTK_STOCK_MEDIA_STOP);
#else
        button_play = gtk_button_new_with_label (">");
        pause = gtk_button_new_with_label ("||");
        stop = gtk_button_new_with_label ("[]");
#endif
#endif

#ifdef HAVE_GPE

        filesel = gpe_button_new_from_stock (GPE_STOCK_OPEN,
                GPE_BUTTON_TYPE_ICON);
#else
        filesel = gtk_button_new_from_stock (GTK_STOCK_OPEN);
#endif

        slider = gst_player_timer_new (play);
        video = gst_player_video_new (audiosink, play);
        g_print ("chin chun chan");

        hbox = gtk_hbox_new (TRUE, 0);

```

```

vbox = gtk_vbox_new (FALSE, 0);

data->widget = slider;

gtk_window_set_title (GTK_WINDOW (app),
                      "Siobhan Audio Player");

gtk_box_pack_start (GTK_BOX (hbox),
                    button_play, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (hbox),
                    pause, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (hbox),
                    stop, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (hbox),
                    filesel, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (vbox),
                    video, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (vbox),
                    slider, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (vbox),
                    hbox, FALSE, TRUE, 0);

gtk_container_add (GTK_CONTAINER (app), vbox);

g_signal_connect (G_OBJECT (app), "delete_event",
                  G_CALLBACK (gtk_main_quit), NULL);
g_signal_connect (G_OBJECT (button_play), "clicked",
                  G_CALLBACK (cb_play), data);
g_signal_connect (G_OBJECT (pause), "clicked",
                  G_CALLBACK (cb_pause), play);
g_signal_connect (G_OBJECT (stop), "clicked",
                  G_CALLBACK (cb_stop), play);
g_signal_connect (G_OBJECT (filesel), "clicked",
                  G_CALLBACK (cb_choose), data);

g_idle_add (cb_iterate, (gpointer) data);

gtk_widget_show_all (app);

gtk_main ();
}

```