Práctica 5:
Kiosco multimedia

Fundamentos de Sistemas Embebidos

Autor: José Mauricio Matamoros de Maria y Campos

1. Objetivo

El alumno aprenderá a configurar la Raspberry Pi como un kiosco multimedia para la reproducción de archivos de audio y video en modo desatendido.

2. Introducción

Cajeros automáticos, mapas interactivos, máquinas de turnos y puntos de venta de autoservicio son algunos ejemplos de kioscos informáticos que ya forman parte de la vida diaria de las personas. Así, los kioscos son una de las aplicaciones más comunes que tienen los sistemas embebidos de gama alta.

Un kiosco integra un conjunto mínimo de periféricos con un procesador de bajo costo para llevar a cabo tareas dedicadas con una interfaz sencilla y amigable para el usuario. Al mismo tiempo, el desempeño del kiosco debe ser óptimo sin subutilizar sus recursos, pero ofreciendo un mínimo de componentes interactivos para evitar el mal manejo del mismo. Es por este motivo que se suelen ocupar tarjetas controladoras de gama media con sistemas operativos que carecen de los elementos clásicos de escritorio que permitirían que cualquier usuario malintencionado pueda hacerse del control del mismo. Para este fin puede utilizarse una Raspberry Pi con un sistema operativo tipo Linux configurado con un gestor gráfico simple como `nodm` (véase la Sección 3), y un conjunto de paquetes de reproducción de multimedios como `ffmpeg` (véase la Sección 4) o `vlc` (véase la Sección 5).

3. NoDM

`nodm` es un manejador gráfico mínimo que inicia sesión gráfica de forma automática con un usuario predeterminado. Al ser un manejador gráfico mínimo `nodm` no integra código para realizar una composición de ventanas sobre un escritorio u operarlas. Es más, no incluye siquiera controles para iniciar sesión o pedir una contraseña.

De acuerdo con el sentido común y con la documentación misma de `nodm`, esto representaría un problema de seguridad en cualquier computadora, pues se da acceso a cualquiera al sistema. No obstante, el autor mismo reconoce que es justamente para esto que se creó `nodm`: para inicializar entornos gráficos en sistemas embebidos tales como dispositivos móviles, paneles de control, y kioscos.

Aunque `nodm` se considera un proyecto abandonado desde 2017 y hoy día manejadores gráficos más pesados como *lightdm* o *gdm* permiten iniciar sesión de forma automática sin necesidad de ingresar contraseña en entornos gráficos de escritorio como *Gnome Desktop* de Ubuntu o Xfce, `nodm` sigue siendo una excelente opción en sistemas embebidos comerciales e industriales por ser extremadamente ligero y rápido, al punto que sigue disponible para su instalación en sistemas basados en Debian como *Raspberry Pi OS Lite* o *Ubuntu Core*.

4. FFmpeg

De acuerdo con la documentación oficial, FFmpeg es un proyecto de software gratuito y de código abierto que consiste en un conjunto de bibliotecas y programas para manejar archivos de audio, video y transmisiones multimedia.

El núcleo es la mismísima herramienta de línea de comandos `ffmpeg` diseñada para el procesar y recodificar archivos multimedia. Sin embargo es `ffplay` la herramienta que es de mayor interés en los kioscos multimedia, ya que permite la reproducción de virtualmente cualquier archivo multimedia soportado por FFmpeg, incluyendo virtualmente todas las opciones de transcodificación y transformación aplicables en tiempo real.

La reproducción de archivos es trivial, basta con ejecutar el comando `ffplay` seguido del nombre del archivo a reproducir. Por ejemplo

```
$ ffplay video.mp4
```

5. VLC media player

De acuerdo con la documentación oficial, *VLC media player* es un software de reproductor de archivos multimedia multiplataforma, gratuito y de código abierto, además de ser portátil e incorporar un servidor de transmisión o *streaming* desarrollado por el proyecto VideoLAN. *VLC* es quizá el reproductor multimedia más utilizado por el simple hecho de ser capaz de reproducir virtualmente cualquier archivo multimedia, ya sea audio o video, además de medios físicos como DVSs y BluRays sin necesidad de instalar codecs opcionales.

La reproducción de archivos es trivial, basta con ejecutar el comando `ffplay` seguido del nombre del archivo a reproducir. Por ejemplo

```
$ vlc video.mp4
```

Una ventaja de *VLC media player* es que puede ser controlado facilmente desde python con el paquete `python3-vlc`, lo que lo convierte en una herramienta increíblemente poderosa para todo tipo de tareas multimedia.

6. Material

Se asume que el alumno cuenta con un una Raspberry Pi 3B o posterior con sistema operativo **Raspberry Pi OS Lite** y un monitor conectado a la misma. Se aconseja encarecidamente el uso de *git* como programa de control de versiones.

Importante

Esta práctica precisa de un sistema mínimo **SIN ESCRITORIO** como *Raspberry Pi OS Lite* o *Ubuntu Core*. Si el sistema instalado es una versión de escritorio o cuenta con un manejador gráfico, no podrá realizar la práctica.

7. Instrucciones

1. Descargue e instale los paquetes necesarios (`nodm`, `ffmpeg` y `vlc`, entre otros).
2. Haga una prueba con `ffplay`
3. Haga una prueba con `vlc`
4. Configure la raspberry para que reproduzca videos desde el arranque en modo desatendido.

7.1. Paso 1: Instalación de paquetes

Antes de instalar paquetes, verifique que su Raspberry esté conectada a internet.
A continuación, actualice el gestor de paquetes aptitude con la línea

```
# apt update
```

Posteriormente, proceda a instalar los paquetes necesarios con la línea

```
# apt -y install nodm ffmpeg vlc python3-vlc wget
```

Durante la instalación de `nodm` se le mostrará una advertencia de seguridad, preguntándole si desea habilitar `nodm` como gestor gráfico. Confirme su decisión.

Finalmente, descargue el video de prueba con la línea:

```
$ wget -qO- http://media.tar.gz | tar xvz -C /home/pi
```

7.2. Paso 2: Prueba de reproducción de video con FFmpeg

El siguiente paso consiste en probar la reproducción de video con *ffplay*.

De manera predeterminada la raspberry Pi estará configurada para enviar el canal de audio al monitor HDMI conectado. Como la mayoría de los monitores no cuentan con bocinas integradas, es necesario reconfigurar la raspberry para utilizar la salida de audio estándar con la utilidad de configuración `raspi-config`.

Una vez configurada la salida de audio, reproduzca el video de prueba con la línea

```
$ ffplay ~/videos/video.mp4
```

En cuestión de segundos verá el video desplegarse en su pantalla. Para terminar la ejecución utilice la combinación de teclas `CTRL+C`.

7.3. Paso 3: Prueba de reproducción de fotografías con VLC

El siguiente paso consiste en probar la reproducción de fotografías *VLC*. Lo anterior se logra mediante la creación de una lista de reproducción que *VLC* pueda usar para reproducir las fotografías en modo presentación una tras otra.

Para generar dicha lista ejecute los siguientes comandos:

```
$ cd ~/pictures
$ echo "#EXTM3U" > playlist.m3u
$ for f in *.jpg; do realpath "$f" >> playlist.m3u; done
```

Finalmente, inicie la reproducción con *VLC* con la línea:

```
$ vlc playlist.m3u --repeat
```

En cuestión de segundos verá las fotografías desplegarse en pantalla. Para terminar la ejecución utilice la combinación de teclas `CTRL+C`.

Cabe mencionar que *VLC* permite la reproducción de todos los medios en un directorio simplemente especificando la ruta del directorio en lugar de la ruta de un archivo. ¡Incluso *VLC* es capaz de acceder a contenido multimedia en línea y reproducirlo! Sin embargo, se ha optado por la opción de crear una lista de reproducción a fin de tener más control en el contenido que *VLC* reproduce.

7.4. Paso 4: Configuración de kiosco en modo desatendido

Para configurar la Raspberry Pi en modo kiosco, bastaría con editar el archivo `/etc/rc.local`. Sin embargo, la reproducción de multimedios sin control alguno es prácticamente inútil, por lo que ésta suele delegarse a un script que controle el flujo y permita reaccionar a eventos tales como el toque de la pantalla o de un botón. Aquí es donde entra en juego el *wrapper* de Python para la API de *VLC*.

El uso de la API es increíblemente simple. Una vez instalada, basta con seguir 5 simples pasos: i) importar el paquete vlc en python, ii) crear un objeto reproductor o *player*, iii) crear un objeto media por cada multimedio a reproducir, iv) cargar el objeto media en el reproductor y, finalmente, v) iniciar la reproducción.

src/minikiosk.py

```
1 import vlc
2 player = vlc.MediaPlayer()
3 video = vlc.Media('/home/pi/videos/video.mp4')
4 player.set_media(video)
5 player.play()
```

La función `play()` comienza la reproducción del medio de forma asíncrona y regresa de inmediato, por lo que la ejecución se interrumpirá si el programa termina a continuación. Dependerá del programador controlar durante cuanto tiempo se ejecuta el video, por ejemplo al presionarse un botón. Se puede dejar al video ejecutarse hasta el final consultando la propiedad `is_playing` del objeto `MediaPlayer`.

Finalmente para configurar la ejeución del script cuando la Raspberrry Pi reinicie agregue la siguiente línea al archivo `/etc/rc.local` justo antes de la línea de retorno `return 0`.

```
python3 /home/pi/minikiosk.py &
```

8. Experimentos

1. [2 pts] Modifique el archivo de ejemplo `minikiosk.py` para que reproduzca el video de prueba durante 10 segundos e inmediatamente después reproduzca en un bucle infinito cada una de las imágenes de muestra por espacio de 3 segundos.

2. [4 pts] Integre el archivo `minikiosk.py` modificado con el detector de medios extraibles `usbdetect.py` para que la presentación infinita de imágenes se detenga cuando se inserta una USB y, en lugar de las imágenes de muestra, se reproduzcan las imágenes del medio extraíble.

3. [+2 pts] Modifique el programa anterior para que el video se reproduzca por espacio de 20 segundos. El video debe comenzar con volumen cero que se incrementará gradualmente hasta 100 % durante los primeros 5 segundos, reproducirse a máximo volumen durante 10 segundos y, finalmente, bajar el volumen gadualmente de 100 % a cero durante los últimos 5 segundos.

4. [+5 pts] Modifique el programa anterior para que la lista de reproducción sea controlable utilizando los pines GPIO2, GPIO3, GPIO4, GPIO17, GPIO27 y GPIO22 como anterior, siguiente, parar, pausar/resumir volumen+ y volumen- respectivamente.

9. Cuestionario

1. [1.0pt] ¿Qué funciones o métodos se utilizan para subir y bajar el volumen durante una reproducción? Ilustre con código de ejemplo.

2. [1.0pt] ¿Qué funciones o métodos se utilizan para ejecutar la reproducción en modo de pantalla completa? Ilustre con código de ejemplo.

3. [2.0pt] ¿Cuáles son las funciones que permiten desplazarse en una lista de reproducción y controlarla (siguiente, anterior, pausa, detener y reanudar)? Ilustre con código de ejemplo

A. Programa Ejemplo: `minikiosk.py`

minikiosk.py

```
import vlc
import time

player = vlc.MediaPlayer()
video = vlc.Media('/home/pi/videos/video.mp4')
player.set_media(video)
player.play()
while player.is_playing:
  time.sleep(0)
```

B. Programa Ejemplo: `usbdetect.py`

usbdetect.py

```
import os
import pyudev
from time import sleep
import subprocess as sp

def print_dev_stats(path):
  photos = []
  for file in os.listdir(path):
    if file.endswith(".jpg") \
    or file.endswith(".png"):
      photos.append(file)
  print("{} has {} photos.".format(path, len(photos)))
#end def

def print_dev_info(device):
  print("Device sys_path: {}".format(device.sys_path))
  print("Device sys_name: {}".format(device.sys_name))
  print("Device sys_number: {}".format(device.sys_number))
  print("Device subsystem: {}".format(device.subsystem))
  print("Device device_type: {}".format(device.device_type))
  print("Device is_initialized: {}".format(device.is_initialized))
#end def

def auto_mount(path):
  args = ["udisksctl", "mount", "-b", path]
  sp.run(args)
#end def

def get_mount_point(path):
  args = ["findmnt", "-unl", "-S", path]
  cp = sp.run(args, capture_output=True, text=True)
  out = cp.stdout.split(" ")[0]
  return out
#end def

context = pyudev.Context()
monitor = pyudev.Monitor.from_netlink(context)
monitor.filter_by(subsystem="block", device_type="partition")
while True:
  action, device = monitor.receive_device()
  if action != "add":
    continue
  print_dev_info(device)
  auto_mount("/dev/" + device.sys_name)
  mp = get_mount_point("/dev/" + device.sys_name)
  print("Mount point: {}".format(mp))
  print_dev_stats(mp)
  break
```