Proyecto Final

Centro Multimedia.

Fundamentos de Sistemas Embebidos
Autor: Jiménez Flores José Emanuel

Objetivo

Desarrollar e implementar un centro multimedia utilizando una Raspberry Pi, diseñado como un sistema embebido, que permita la gestión de contenido de entretenimiento. Este sistema será capaz de reproducir películas, música en bucle, fotografías en modo presentación y videos tanto desde servicios en línea como desde dispositivos de almacenamiento extraíbles.

Introducción

El siguiente docuemnto presenta el desarrollo de un centro multimedia utilizando una Raspberry Pi, destacando su capacidad para gestionar contenido audiovisual como videos, música y fotografías desde diversas fuentes. El proyecto explora la funcionalidad y flexibilidad de los sistemas embebidos, mostrando cómo estos pueden adaptarse a necesidades específicas de entretenimiento digital (Estrada, 2014; Barr Massa, 2006).

Antecedentes

Los sistemas embebidos son soluciones tecnológicas diseñadas para realizar tareas específicas de manera eficiente, integrando hardware y software en dispositivos compactos y confiables (Barr Massa, 2006).La Raspberry Pi, una microcomputadora versátil y económica, destaca por su capacidad para ejecutar sistemas como Raspbian, una distribución de Linux optimizada para proyectos educativos y profesionales (Estrada, 2014).

Los centros multimedia, al incorporar tecnologías como la Raspberry Pi, permiten gestionar y reproducir música, videos y fotografías, integrando contenido local y en línea para una experiencia de entretenimiento flexible.

Materiales

a) **Placa Raspberry Pi:** Modelo 3 b +, de preferencia 4 b.

b) **MicroSD de al menos 8GB:** Configurada con el sistema operativo Raspbian.

c) **Acceso a Internet:** Mediante conexión por cable o WiFi, según disponibilidad.

d) **Pantalla compatible:** Monitor o televisor con entrada HDMI o VGA (requiere adaptador).

e) **Adaptador micro HDMI:** Necesario para conectar la Raspberry Pi al dispositivo de video.

f) **Periféricos de entrada:** Teclado y mouse con puertos USB estándar.

g) **Sistema de audio:** Bocinas o auriculares con conector de 3.5 mm para salida de sonido.

h) **Fuente de alimentación confiable:** Regulada a 5V con un mínimo de 2A de corriente.

Descripción del funcionamiento de la Raspberry Pi

La Raspberry Pi es un dispositivo de cómputo diseñado para ser compacto y accesible, ofreciendo capacidades avanzadas a través de un procesador ARM y soporte para conexiones modernas como USB, micro HDMI, Ethernet y WiFi. Es ampliamente utilizado en proyectos tecnológicos gracias a su bajo costo y versatilidad, permitiendo desarrollar desde sistemas básicos hasta aplicaciones complejas (Estrada, 2014).

Para su operación, es fundamental configurar una tarjeta microSD con Raspbian, un sistema operativo basado en Linux optimizado para el hardware de la Raspberry Pi. Una vez conectados los periféricos necesarios, como monitor, teclado, mouse y fuente de alimentación, la placa actúa como una computadora estándar. Este proyecto utiliza Python como base para el desarrollo, aprovechando la facilidad de instalación de módulos y la ejecución desde terminal, lo que agiliza el flujo de trabajo (Raspberry Pi Foundation, 2021).

Cuidado de componentes electrónicos y Cuidado de la salud y riesgos

Usa equipo antiestático al manipular la microSD o los terminales para evitar daños. Mantén el dispositivo dentro de los límites de temperatura recomendados para un funcionamiento seguro y prolongado.

Al utilizar la Raspberry Pi, es importante evitar el contacto prolongado con la placa mientras está en funcionamiento, ya que puede alcanzar temperaturas altas. Desconecta siempre el dispositivo antes de realizar cualquier ajuste o conexión adicional para minimizar el riesgo de accidentes eléctricos.

Configuración de la tarjeta controladora

Cambiar la imagen de arranque de Raspbian

Para eliminar la imagen predeterminada de inicio, se debe editar el archivo `/boot/config.txt` y añadir la siguiente línea al final:

```
disable_splash=1
```

Esto desactivará la imagen de inicio predeterminada. Para cambiar la imagen, debe renombrarse a `splash.png` y luego copiarla utilizando el siguiente comando:

```
$ sudo cp (ubicacion_imagen)/splash.png /usr/share/plymouth/themes/pix
```

Configurar la Raspberry Pi en modo consola

Ejecute `raspi-config` para acceder a la configuración de la tarjeta. Seleccione la opción `3 Boot options`, luego `B1 Desktop / CLI`, y finalmente `B2 Console Autologin`. Esta configuración hará que la Raspberry Pi arranque en modo consola automáticamente al reiniciar.

Configurar script de arranque

Para ejecutar un script de manera automática al iniciar, edite el archivo `rc.local` con el siguiente comando:

```
$ sudo nano /etc/rc.local
```

Inserte el siguiente script antes de la línea `exit 0`:

```
if ... fi /home/pi/Desktop/multimedia.py exit 0
```

Librerías a Instalar

A continuación se listan las librerías necesarias para el proyecto:

Librería	Descripción	Comando de Instalación
Tkinter	Librería para crear aplicaciones de escritorio.	`$ sudo apt-get install python3-tk`
Pyautogui	Librería de Python para controlar el mouse y el teclado.	`$ pip3 install pyautogui`
Pygame	Librería para la creación de aplicaciones multimedia (música).	`$ pip3 install pygame`
Subprocess	Librería para ejecutar comandos del sistema operativo.	`$ sudo apt-get install python-pygame`
OpenCV	Librería para la reproducción de fotos.	`$ pip3 install opencv-python-headless`
Pyudev	Librería para interactuar con el sistema de administración de dispositivos.	`$ sudo apt-get install python3-pyudev`
Keyboard	Librería para detectar y simular eventos del teclado.	`$ pip3 install keyboard`
Widevine	Librería para la decodificación de contenido protegido por DRM.	`$ sudo apt-get install widevine`
Pyttsx3	Librería de síntesis de voz para convertir texto a voz.	`$ pip3 install pyttsx3`

Desarrollo de componentes de software

En este proyecto, se desarrollaron diversos módulos para implementar las funcionalidades requeridas. Los módulos de **ventana principal**, **botón para Netflix**, **botón para Spotify** y **reproducción multimedia** fueron asignados a a los dos miembros del equipo, mientras que la implementación de los otros módulos fue responsabilidad del otro miembro del equipo.

1. Módulo de Ventana Principal

El módulo de la **ventana principal** fue desarrollado utilizando la librería `Tkinter`, la cual es una herramienta estándar de Python para crear interfaces gráficas. Este módulo se encarga de crear la ventana principal del sistema multimedia, establecer sus atributos y configuraciones iniciales. La ventana está configurada para ocupar toda la pantalla y tener un fondo personalizado.

```
import tkinter as tk

# Creación de la ventana principal
window = tk.Tk()
window.config(bg=colorFondo, cursor="circle")  # Establece el color de fondo y el cursor
window.geometry("%dx%d" % (width, height))  # Define la geometría de la ventana
window.attributes('-fullscreen', True)  # Configura la ventana en pantalla completa
window.title("Centro Multimedia")  # Título de la ventana
window.mainloop()  # Inicia la ventana
```

El código anterior configura la ventana principal, establece su tamaño, y la configura para que se muestre en pantalla completa. La función `mainloop` asegura que la ventana se mantenga activa durante toda la ejecución del programa.

Figura 1: Ventana principal

2. Módulo de Botón para Netflix

Para abrir el servicio de streaming **Netflix**, se utilizó la librería `webbrowser`, que permite abrir páginas web en el navegador predeterminado del sistema. Este módulo implementa una función que, al hacer clic en el botón correspondiente, abre el sitio web de Netflix.

```
import webbrowser as wb

# URL de Netflix
url_netflix = 'https://www.netflix.com/mx/'

# Función para abrir Netflix
def abrir_netflix():
    wb.open_new(url_netflix)  # Abre el navegador con el URL de Netflix
    window.after(500, pantalla_completa)  # Abre la aplicación en pantalla completa

# Crear el botón para Netflix
img_netflix = tk.PhotoImage(file="./iconos/netflix.png")
boton_netflix = tk.Button(window, image=img_netflix, borderwidth=0, bg=colorFondo,
cursor="X_cursor", command=abrir_netflix)
boton_netflix.pack()
boton_netflix.place(relx=0.5, rely=0.5)
```

Este código configura un botón con la imagen de Netflix que, al ser presionado, abre Netflix en el navegador.

3. Módulo de Botón para Spotify

El módulo para el **botón de Spotify** funciona de manera similar al de Netflix. Al presionar el botón, se abre el servicio de Spotify en el navegador.

```
# URL de Spotify
url_spotify = 'https://www.spotify.com/'

# Función para abrir Spotify
def abrir_spotify():
```

```
    wb.open_new(url_spotify)  # Abre el navegador con el URL de Spotify
    window.after(500, pantalla_completa)  # Abre la aplicación en pantalla completa

# Crear el botón para Spotify
img_spotify = tk.PhotoImage(file="./iconos/spotify.png")
boton_spotify = tk.Button(window, image=img_spotify, borderwidth=0, bg=colorFondo,
cursor="X_cursor", command=abrir_spotify)
boton_spotify.pack()
boton_spotify.place(relx=0.5, rely=0.6)
```

Similar al botón de Netflix, este código crea un botón con la imagen de Spotify y ejecuta la acción de abrir la página de Spotify en el navegador cuando se presiona.

4. Módulo de Reproducción Multimedia

El módulo de **reproducción multimedia** fue desarrollado en colaboración con otro miembro del equipo. Este módulo es responsable de la gestión y reproducción de contenido multimedia desde la unidad USB conectada, como fotos, videos y música.

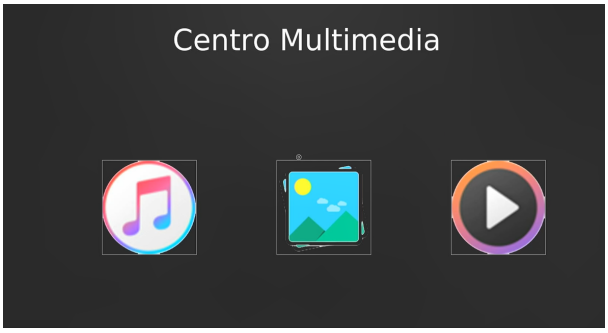

Figura 2: Módulo de reproducción multimedia

Reproducción de Fotos

La función `show_images` recibe la ruta de la unidad USB como parámetro, identifica los archivos de imagen válidos (con extensiones `.jpg` y `.png`) y los muestra en pantalla usando `OpenCV`. Las imágenes se visualizan en bucle hasta que se detenga la reproducción multimedia.

```
import cv2
import os

def show_images(self, usb):
    image_files = []  # Lista para almacenar los nombres de archivo de las imágenes
    # Iterar sobre los archivos en la unidad USB
    for file in os.listdir(usb):
        if file.endswith((".jpg", ".png")):
            image_files.append(file)
            # Si el archivo tiene una extensión válida, se agrega a la lista

    while self.media_playing:
        for file in image_files:
            if not self.media_playing:
                break
```

```
            image_path = os.path.join(usb, file)  # Ruta completa de la imagen
            self.show_image(image_path)
            # Mostrar la imagen utilizando la función show_image
```

Reproducción de Videos

La función `play_videos` permite identificar archivos de video en la unidad USB (con extensiones `.mp4` y `.avi`), y luego los reproduce utilizando una función adicional.

```
def play_videos(self, usb):
    video_files = []  # Lista para almacenar los nombres de archivo de los videos
    # Iterar sobre los archivos en la unidad USB
    for file in os.listdir(usb):
        if file.endswith((".mp4", ".avi")):
        # Si el archivo tiene una extensión válida, se agrega a la lista
            video_files.append(file)

    while self.media_playing:
        for file in video_files:
            if not self.media_playing:
                break
            video_path = os.path.join(usb, file)  # Ruta completa del video
            # Reproducir el video utilizando la función play_video
            self.play_video(video_path)
```

Reproducción de Música

La función `play_music` carga y reproduce archivos de música en formato `.mp3` desde la unidad USB utilizando `Pygame`. La música se reproduce en bucle hasta que se detenga la reproducción multimedia.

```
import pygame

def play_music(self, usb):
    music_files = []  # Lista para almacenar los nombres de archivo de música
    # Iterar sobre los archivos en la unidad USB
    for file in os.listdir(usb):
        if file.endswith(".mp3"):
         # Si el archivo tiene una extensión válida, se agrega a la lista
            music_files.append(file)

    while self.media_playing:
        for file in music_files:
            if not self.media_playing:
                break
            music_path = os.path.join(usb, file)  # Ruta completa del archivo de música
             # Inicializar el módulo mixer de Pygame para reproducir música
            pygame.mixer.init()
            # Cargar el archivo de música en el reproductor
            pygame.mixer.music.load(music_path)
            pygame.mixer.music.play()  # Reproducir la música
            while pygame.mixer.music.get_busy() and self.media_playing:
                continue
```

Video del funcionamiento

Sea anexa el funcionamiento del proyecto en el siguiente link de youtube: Video Funcionamiento - Centro Multimedia.

Repositorio del código

Sea anexa el código del proyecto, licencia y readme en el siguiente enlace: Repositorio del proyecto en GitHub.

Cuestionario

1. En el módulo de `detectar_usb`, ¿cómo modificarías el código para que, en lugar de solo detectar la conexión de un dispositivo USB, se muestre una lista de todos los archivos multimedia presentes en la unidad USB (fotos, videos, música) directamente en la interfaz gráfica?

2. En el módulo de reproducción de videos (`play_videos`), ¿cómo podrías agregar una funcionalidad para permitir al usuario pausar y reanudar la reproducción del video mediante un botón en la interfaz gráfica? Además, ¿cómo podrías incluir un control de volumen en esa interfaz?

3. En la función `play_music`, ¿cómo realizarías las modificaciones necesarias para que, al seleccionar una canción desde la unidad USB, se muestre un listado de todas las canciones disponibles en una lista desplegable o en un cuadro de texto en la interfaz gráfica, y que el usuario pueda elegir cuál reproducir?

4. En el módulo de visualización de imágenes (`show_image`), ¿cómo modificarías el código para permitir que el usuario rote o voltee las imágenes antes de mostrarlas, añadiendo un botón o un menú para elegir entre estas opciones de transformación?

Conclusiones

Se desarrolló con éxito un centro multimedia utilizando una Raspberry Pi, permitiendo acceder a servicios de streaming y reproducir contenido desde USB. Este proyecto ayudó a entender mejor cómo funcionan los sistemas embebidos y cómo integrarlos con hardware y software. Aunque fue un reto lograr la detección de dispositivos USB y configurar el inicio automático, fue una experiencia muy valiosa que permitió aprender y explorar las capacidades de la Raspberry Pi.

Referencias

Barr, M., & Massa, A. (2006). *Programming Embedded Systems: With C And GNU Development Tools* (2ª ed.). O'Reilly Media.

Raspberry Pi Foundation. (2021). *Raspberry Pi Documentation*. Recuperado de `https://www.raspberrypi.org/documentation/`

Halfacree, G. (2018). *Raspberry Pi User Guide* (4th ed.). Wiley.

Grayson, D. (2014). *Python and Tkinter Programming*. McGraw-Hill Education.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.

OpenCV Documentation. (2021). Recuperado de https://docs.opencv.org/

Pygame Documentation. (2021). Recuperado de https://www.pygame.org/docs/

Maxim Integrated. (2008). Overview of 1-Wire Technology and Its Use. Recuperado de https://www.analog.com/media/en/technical-documentation/tech-articles/guide-to-1wire-communication--maxim-integrated.pdf

Raspberry Pi Foundation. (2021). *Wi-Fi Configuration on Raspberry Pi*. Recuperado de https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

Pyudev Documentation. (2021). Recuperado de https://pyudev.readthedocs.io/

Python Software Foundation. (2021). *Python Documentation*. Recuperado de https://docs.python.org/3/