# Proyecto Final: Centro Multimedia

Ocaña Casillas Jonathan Leonardo NL: 21

Noviembre 26, 2024

## 1. Introducción

Un sistema embebido es conocido como un circuito electrónico computarizado que está diseñado para cumplir una labor específica en un producto. Pues a diferencia de los sistemas computacionales de oficina y laptops, estos sistemas solucionan un problema específico y están dispersos en todos los ambientes posibles de la vida cotidiana. Es común encontrar sistemas embebidos en los vehículos, por ejemplo, por el sistema de frenado ABS, en el Airbag, alarmas o en los sistemas de inyección de combustible, etc [1].

La Raspberry Pi (RPi) es una plataforma versátil basada en el sistema operativo Linux, diseñada para proporcionar una base accesible y flexible para proyectos tecnológicos. Su compatibilidad con aplicaciones y bibliotecas de software de código abierto la convierte en una herramienta poderosa para desarrollar soluciones innovadoras. Con la capacidad de integrar fácilmente dispositivos como cámaras USB, teclados y adaptadores WiFi, elimina la necesidad de depender de soluciones propietarias, fomentando la creatividad y la personalización en una amplia gama de aplicaciones [2].

Tkinter es la interfaz estándar de Python para el kit de herramientas GUI Tk, permitiendo crear aplicaciones gráficas en múltiples plataformas, como Unix, Windows y Mac, con una apariencia nativa desde la versión 8.0. Su núcleo se basa en el módulo binario _tkinter, que actúa como puente de bajo nivel con Tk, aunque los programadores trabajan principalmente con el módulo principal Tkinter y su complemento Tkconstants, importados automáticamente para facilitar su uso [3]

## 2. Objetivo

El objetivo de este proyecto es desarrollar un sistema embebido de entretenimiento que permita a los usuarios reproducir películas, videos, música y fotografías tanto desde medios extraíbles como mediante servicios en línea seleccionados. Este sistema está diseñado para ofrecer una experiencia intuitiva y fluida, garantizando compatibilidad con contenido multimedia moderno y priorizando una interfaz accesible y fácil de usar.

## 3. Lista de materiales

- Raspberry Pi 4.
- Fuente de alimentación 5.0 V a 3.0 A para la Raspberry Pi.
- Cable micro HDMI a VGA o adaptador micro HDMI a HDMI.
- Tarjeta microSD 8 GB (mínimo).
- Unidad USB con contenido multimedia.
- Monitor.
- Teclado.

- Mouse.
- Bocinas o audífonos 3.5 mm.

# 4. Descripción del funcionamiento de los componentes

## 4.1. Cable micro HDMI a VGA

- Transmisión de Video y Audio: Este cable permite conectar la Raspberry Pi al monitor, transmitiendo video. Puede requerir un adaptador micro HDMI a HDMI dependiendo del modelo del monitor.

### 4.1.1. Unidad USB

- Almacenamiento de Datos: Utiliza tecnología de memoria no volátil para almacenar archivos multimedia y no requiere alimentación constante para mantener los datos.

## 4.2. Teclado y Mouse

- Entrada de Datos: Son los dispositivos primarios de interacción con el sistema, utilizados para navegar por la interfaz y controlar las funciones del centro multimedia. Se conectan a través de puertos USB o adaptadores compatibles.

## 4.3. Monitor

- Dispositivo de Salida Visual: Muestra las imágenes y videos generados por la Raspberry Pi.
- Conectividad: Puede conectarse mediante cables HDMI o adaptadores compatibles.

## 4.4. Fuente de Alimentación

- Suministro de Energía: Alimenta a la Raspberry Pi para su funcionamiento continuo. Es importante usar una fuente de 5V y al menos 3A para garantizar la estabilidad. Se recomienda el uso de la fuente de la misma marca de Raspberry.

## 4.5. Bocinas y Audifonos

- Tipos de Conexión: Se conectan mediante cables de audio (3.5 mm), proporcionando la salida de audio necesaria para el sistema multimedia.

## 4.6. Tarjeta microSD

- Almacenamiento del Sistema Operativo: Funciona como disco principal para el almacenamiento del sistema operativo (Raspbian Lite) y archivos esenciales del proyecto, permitiendo el inicio y funcionamiento del sistema.

# 5. Información sobre el cuidado de la salud

Evite tocar la placa de la Raspberry Pi mientras está en funcionamiento, ya que puede calentarse y causar quemaduras leves. Asimismo, asegúrese de que el sistema esté desconectado de la fuente de alimentación para prevenir descargas eléctricas. Si el proyecto implica exposición prolongada a pantallas o sonidos, tome descansos regulares para proteger su vista y evitar fatiga auditiva. Finalmente, mantenga una postura ergonómica al usar el teclado y el mouse para prevenir lesiones musculares o articulares.

# 6. Información sobre el cuidado de los componentes

Manipule los componentes electrónicos con precaución, evitando golpes, caídas y contacto con líquidos que puedan causar daños a cualquier componente. Manténgalos alejados de fuentes de calor y luz solar directa para prevenir sobrecalentamientos. Si está en su posibilidad, utilice cajas protectoras para la Raspberry Pi y mantenga los cables organizados para evitar desconexiones accidentales o desgaste prematuro.

# 7. Configuración de la tarjeta.

La Raspberry Pi debe de ser debidamente conectada con todos los periféricos necesarios y configurando el entorno. Una vez conectados los dispositivos y encendida la Raspberry Pi mediante una fuente de alimentación, se debe instalar un sistema operativo como Raspberry Pi OS Lite en una tarjeta microSD. Posteriormente, se configuran las herramientas y bibliotecas necesarias:

- **Ffmpeg**: Multimedia capaz de decodificar, codificar, transcodificar, multiplexar, demultiplexar, transmitir, filtrar y reproducir contenido multimedia [4].
- **Pulseaudio**: Servidor de sonido que gestiona el audio del sistema y permite su distribución a diferentes dispositivos [5].
- **Vlc y python3-vlc**: Utilizados para la reproducción de audio y video, soportan una gran variedad de formatos multimedia [6].
- **Python3-pyudev**: Permite interactuar con dispositivos USB y realizar detección automática a través de udev [7].
- **Tkinter**: Librería estándar de Python para la creación de interfaces gráficas de usuario [8].
- **Pillow (PIL)**: Extensión de procesamiento de imágenes en Python, utilizada para redimensionar y mostrar gráficos [9].
- **Screeninfo**: Biblioteca para obtener información sobre las dimensiones de la pantalla, esencial para interfaces dinámicas [10].
- **Chromium**: Navegador web que permite la visualización de contenido en plataformas de streaming y aplicaciones web [11].

# 8. Desarrollo de los módulos

- **Carpeta img_interfaz** Es la carpeta que almacena cada una de las imagenes que se ocupan dentro del centro multimedia.
- **Módulo Principal (main.py)**
  Es el archivo principal que ejecuta el sistema de Centro Multimedia. Utiliza la biblioteca tkinter para mostrar un menú principal con opciones para gestionar red, USB y streaming. Adapta dinámicamente la interfaz a la resolución de la pantalla y coordina la navegación entre las diferentes interfaces.
- **Modulo de Red (red.py)**
  Maneja la configuración de red, permitiendo la visualización de redes WiFi disponibles y la conexión a ellas mediante un cuadro de contraseña. Actualiza dinámicamente el estado de conexión y utiliza botones para navegar entre redes.
- **Módulo de Streaming (streaming.py)**

Proporciona una interfaz para acceder a plataformas de streaming como Netflix, Spotify y Disney+. Abre el navegador Chromium en modo pantalla completa y permite cerrar el navegador desde la interfaz.

- **Módulo USB (usb.py)**
  Administra la interfaz para dispositivos USB conectados. Detecta automáticamente archivos multimedia (imágenes, videos y música) y habilita opciones para visualizarlos o reproducirlos en la interfaz del Centro Multimedia.

- **Modulo de audio en USB (audio_usb.py)**
  Reproductor de música que utiliza VLC para la reproducción. Permite pausar, reanudar y cambiar entre canciones de una lista obtenida desde un dispositivo USB.

- **Módulo de video en USB (video_usb.py)**
  Módulo para gestionar la reproducción de videos desde dispositivos USB. Permite seleccionar videos para reproducirlos individualmente o iniciar una presentación automática de todos los videos disponibles.

- **Módulo de imagenes en USB (image_slideshow.py)**
  Implementa una presentación de imágenes que cambia automáticamente cada cinco segundos. Adapta las imágenes al tamaño de la pantalla y permite regresar al menú anterior.

- **Modulo de Gestor de medios (media_manager.py)**
  Este módulo gestiona dispositivos USB y medios de almacenamiento externos. Detecta automáticamente la conexión y desconexión de dispositivos, monta particiones, clasifica archivos multimedia (imágenes, audio, video) en carpetas específicas y envía actualizaciones sobre estos eventos a una cola para su posterior procesamiento.

- **Módulo de Gestor de redes (wifi_manager.py)**
  Este módulo administra las conexiones WiFi. Permite verificar si el dispositivo está conectado a Internet, escanear redes disponibles, y conectarse a redes especificando el SSID y contraseña. Utiliza herramientas de red como nmcli para gestionar la configuración inalámbrica.

# 9. Integración de los módulos

El Centro Multimedia se construye con una estructura modular que organiza y distribuye las diferentes funcionalidades del sistema en clases y módulos específicos. Esto facilita su integración y el flujo de información entre ellos para proporcionar una experiencia unificada. A continuación, se describe cómo se integran los módulos principales:

## 9.1. Interfaz gráfica (GUI)

El núcleo de la integración recae en la interfaz gráfica gestionada con Tkinter, que conecta y coordina todos los módulos del sistema.

### 9.1.1. main.py

- Funciona como el controlador principal que inicializa el sistema y coordina la navegación entre las interfaces de Red, USB y Streaming.

- Implementa métodos de transición entre pantallas (mostrar_red, mostrar_usb, mostrar_streaming) que instancian las clases correspondientes y limpian la pantalla anterior.
- Adapta dinámicamente el diseño a las dimensiones de la pantalla, detectadas mediante la biblioteca screeninfo.

## 9.2. Módulo USB

### 9.2.1. usb.py

- Gestiona la interacción con dispositivos USB detectados. Integra el módulo media_manager.py para monitorear dispositivos conectados, clasificando los archivos en imágenes, videos y música.
- A través de image_slideshow.py, audio_usb.py y video_usb.py, se permite la visualización o reproducción de los archivos multimedia detectados.
- Integra imágenes para botones y fondos mediante Pillow, garantizando la calidad visual y adaptando los elementos a la interfaz gráfica.
- Los botones de las categorías multimedia (Fotos, Videos, Música) se habilitan o deshabilitan dependiendo de los archivos disponibles en el dispositivo conectado.
- Finalmente media_manager.py es el puente entre los dispositivos USB y las interfaces gráficas, detectando automáticamente la conexión o desconexión de dispositivos, montándolos y clasificando sus archivos multimedia (imágenes, videos y música).

## 9.3. Módulo de Red

### 9.3.1. red.py

- Presenta una interfaz para gestionar la conexión a redes WiFi, utilizando el módulo wifi_manager.py para escanear redes disponibles, verificar el estado de conexión y conectar a redes específicas.
- Los datos recibidos del módulo de red se presentan en botones dinámicos, permitiendo la interacción directa del usuario con las opciones de conexión.
- La integración asegura la sincronización entre el estado de la red y su representación gráfica, actualizando el estado de conexión en tiempo real.

# 10. Conclusiones

El desarrollo de este Centro Multimedia permitió demostrar los conocimientos que obtuve a lo largo del curso, el llevarlo a cabo desde un concepto como lo son las especificaciones a la realidad, consolido mi pensamiento para buscar e implementar soluciones para el funcionamiento del proyecto, desde el diseño de interfaces gráficas con Tkinter hasta el manejo de bibliotecas como Pillow y VLC. Se logró un sistema funcional que conecta dispositivos USB, gestiona redes WiFi y accede a plataformas de streaming, cumpliendo con los objetivos planteados. Aunque se identificaron áreas de mejora, el proyecto cumplió su propósito y deja la oportunidad de tener actualizaciones futuras y desarrollar mejoras a la interfaz.

# 11. Cuestionario

1. **¿Cómo asegura el proyecto que la interfaz gráfica se adapta a diferentes resoluciones de pantalla?**
   El proyecto utiliza la biblioteca screeninfo para obtener las dimensiones del monitor. Todas las interfaces y elementos gráficos se dimensionan dinámicamente según el tamaño de pantalla, lo que

garantiza una visualización correcta en cualquier resolución compatible.

2. **¿Por qué las plataformas de streaming tienen videos protegidos con DRM y cómo se pueden visualizar en este proyecto?**
   Las plataformas usan DRM para proteger los derechos de autor y evitar la distribución no autorizada. En este proyecto, el contenido con DRM se visualiza mediante el navegador Chromium, que soporta Widevine para reproducir videos protegidos. La clase StreamingInterface abre las plataformas en modo inmersivo para facilitar la experiencia del usuario.

3. **¿Cómo se maneja la reproducción de contenido multimedia desde dispositivos USB?**

   El módulo usb.py utiliza media_manager.py para detectar y clasificar archivos multimedia en imágenes, música y videos.

   - **Imágenes**: Se gestionan con image_slideshow.py, permitiendo presentaciones automáticas.
   - **Música**: Se reproducen usando audio_usb.py con la ayuda de VLC.
   - **Videos**: Se manejan con video_usb.py, soportando reproducción individual y en secuencia.

# 12. Bibliografía de consulta.

# Referencias


[1] G. Galeano, *Programación de sistemas embebidos en C*, ser. Colección 3B. México: Alpha Editorial, 2009.

[2] D. Molloy, *Raspberry Pi® a fondo para desarrolladores.* Marcombo, 2019.

[3] F. Lundh. (1999) An introduction to tkinter. Accedido en 26 de noviembre de 2024. [Online]. Available: http://www.pythonware.com/library/tkinter/introduction/index.htm

[4] FFmpeg Team. (2024) About ffmpeg. Accedido en 26 de noviembre de 2024. [Online]. Available: https://www.ffmpeg.org/about.html

[5] P. Developers, "Pulseaudio documentation," https://www.freedesktop.org/wiki/Software/PulseAudio/, 2024, accessed: 2024-11-26.

[6] V. Organization, "Vlc media player," https://www.videolan.org/doc/, 2024, accessed: 2024-11-26.

[7] pyudev Team, "pyudev — libudev bindings for python," https://pyudev.readthedocs.io/en/latest/, 2024, accessed: 2024-11-26.

[8] P. S. Foundation, "Tkinter — python interface to tcl/tk," https://docs.python.org/3/library/tkinter.html, 2024, accessed: 2024-11-26.

[9] P. I. L. Team, "Pillow documentation," https://pillow.readthedocs.io/en/stable/, 2024, accessed: 2024-11-26.

[10] screeninfo Developers, "screeninfo: Get monitor information from the os," https://github.com/rr-/screeninfo, 2024, accessed: 2024-11-26.

[11] T. C. Projects, "Chromium browser," https://www.chromium.org/, 2024, accessed: 2024-11-26.

[12] G. LLC, "Widevine cdm," https://www.widevine.com/, 2024, accessed: 2024-11-26.

## 13. Adicional

Video de prueba en YouTube: `https://youtu.be/JCH2hj2dtEo?si=ZdE_1C7KUk0pxib1`

Repositorio: `https://github.com/m4rc0m0nt3l0ng0/Proyecto_Final_Centro-Multimedia.git`