Proyecto Final:
Centro Multimedia

Fundamentos de Sistemas Embebidos

Alumno: Páez López Didier Marcelo

05 de diciembre de 2023

Introducción y objetivo

En la actualidad, los sistemas embebidos se encuentran presentes en una amplia gama de dispositivos, desde electrodomésticos hasta vehículos. Uno de los campos de aplicación más comunes de los sistemas embebidos es el entretenimiento.

Los sistemas embebidos de entretenimiento permiten a los usuarios disfrutar de una variedad de contenidos multimedia, como películas, vídeos, música y fotografías. Estos sistemas suelen estar integrados en televisores, reproductores de Blu-ray, consolas de videojuegos y otros dispositivos.

El objetivo de este proyecto es implementar un centro de entretenimiento embebido que permita a los usuarios reproducir películas, vídeos, música y fotografías, tanto de medios extraíbles como de servicios en línea.

Este proyecto es un desafío técnico que requiere que los estudiantes desarrollen sus habilidades en el desarrollo de sistemas embebidos y la reproducción de multimedia. El proyecto también es relevante y significativo, ya que responde a una necesidad real de los usuarios.

Lista de materiales o componentes electrónicos

- Raspberry Pi 3 o superior
- Tarjeta microSD con al menos 8 GB de almacenamiento
- Fuente de alimentación para Raspberry Pi
- Cable HDMI
- Control remoto
- Memoria USB con contenido multimedia (opcional)
- Pantalla
- Altavoces

Descripción del funcionamiento de los componentes electrónicos

Memoria USB: La memoria USB se utiliza para almacenar datos, como películas, videos, música y fotografías. La memoria puede ser de tipo estático (SRAM) o de tipo dinámico (DRAM).

Pantalla: La pantalla se utiliza para mostrar imágenes y videos. Las pantallas pueden ser de tipo LCD, LED o OLED.

Altavoces: Los altavoces se utilizan para reproducir sonido. Los altavoces pueden ser de tipo pasivo o activo.

Descripción del funcionamiento de la tarjeta controladora

Raspberry Pi 3

La Raspberry Pi 3 es una computadora de placa única (SBC) basada en el procesador ARM Cortex-A53 de cuatro núcleos a 1,2 GHz. Cuenta con 1 GB de memoria RAM, un puerto HDMI para salida de video, un puerto Ethernet para conexión a Internet y un puerto USB para conectar dispositivos externos.

El funcionamiento de la Raspberry Pi 3 se basa en el principio de la programación. El usuario puede programar la Raspberry Pi para realizar una variedad de tareas, incluyendo la reproducción de multimedia.

Información sobre el cuidado de la salud y advertencias de riesgos

Al trabajar con Raspberry Pi 3, es importante tomar medidas para proteger la salud. Para la función que tendrá en este proyecto es muy recomendable pero no indispensable, basta tener cuidado y tomar precauciones para evitar conflictos entre los componentes, por lo que se propone seguir las siguientes medidas.

Use gafas de seguridad para proteger los ojos de las partículas que se puedan desprender de la Raspberry Pi 3.

Use guantes de protección para proteger las manos de las descargas eléctricas y los cortes.

Use ropa suelta que no se pueda enganchar en la Raspberry Pi 3.

No trabaje con la Raspberry Pi 3 si está cansado o se siente mal.

Advertencias de riesgos específicos al trabajar con Raspberry Pi 3

Quemaduras: La Raspberry Pi 3 puede calentarse mucho, especialmente si está utilizando un disipador de calor. No toque la Raspberry Pi 3 si está caliente.

Electrocución: La Raspberry Pi 3 está conectada a la corriente eléctrica. No toque los pines de alimentación de la Raspberry Pi 3 si está conectada a la corriente eléctrica.

Inhalación de humos tóxicos: La Raspberry Pi 3 puede liberar humos tóxicos si se calienta demasiado o se daña. No trabaje con la Raspberry Pi 3 en un área cerrada sin ventilación.

Irritación de la piel: La Raspberry Pi 3 puede contener sustancias químicas que pueden irritar la piel. Lávese las manos con agua y jabón después de trabajar con la Raspberry Pi 3.

Cuidado de los componentes electrónicos

Los componentes electrónicos delicados, como las memorias, son sensibles a una variedad de factores que pueden dañarlos. Es importante tomar medidas para proteger estos componentes para garantizar un funcionamiento correcto y una larga vida útil.

Recomendaciones específicas para el proyecto de centro de entretenimiento embebido.

1. Utilice una pulsera anti-estática siempre que trabaje con componentes electrónicos delicados.
2. Limpie los terminales de los componentes electrónicos delicados con regularidad.
3. Utilice un disipador de calor o un ventilador para mantener los componentes electrónicos delicados a una temperatura adecuada.
4. Almacene los componentes electrónicos delicados en un lugar limpio y seco.
5. Evite tocar los componentes electrónicos delicados con las manos sucias.
6. No deje caer los componentes electrónicos delicados.

Configuración de la tarjeta controladora

En esta sección se implementó el proyecto con ayuda del sistema operativo Raspberry Pi OS se eligió la versión 6.1 del sistema operativo con escritorio con fecha de lanzamiento del 3 de mayo del 2023 en su versión de 32 bits [Raspberry, 2023], este debía contener la condición de prevenir que no se pudiera salir de la interfaz programada por el equipo, consideramos que con las restricciones a nivel de código quedó correctamente prevenido este campo, una vez elegido el sistema operativo proseguimos con la selección de los paquetes y librerías necesarias para ejecutarlo correctamente.

Para lograr cumplir esta condición el código incluye las siguientes medidas:

Modo de pantalla completa: self.root.attributes('-fullscreen', True) establece el modo de pantalla completa al iniciar la aplicación.

Desactivar el cambio de tamaño: self.root.resizable(False, False) evita que el usuario cambie el tamaño de la ventana.

Manejo del cierre de la ventana: Se ha vinculado la función exit_app al evento de cierre de la ventana.

Las librerías que utilizaremos son:

- VLC media player: Es una biblioteca multimedia utilizada para reproducir archivos de audio y video. [VLC, 2023]
- tkinter (tk): Proporciona herramientas para crear interfaces gráficas de usuario (GUI) en aplicaciones Python. [Python, 2023a]
- os: Proporciona funciones para interactuar con el sistema operativo, como manipulación de archivos y directorios. [Python, 2023b]

Desarrollo de los módulos de software

Para el modulo de Netflix

En la función open_netflix_kiosk, se utiliza el módulo os para ejecutar un comando del sistema. Se abre el navegador Chromium en modo quiosco (–kiosk), que se utiliza para mostrar una página web en pantalla completa. La URL que se carga es la de Netflix (`https://www.netflix.com`). Chromium se ejecuta sin el modo de protección del sistema de archivos (–no-sandbox) para evitar problemas de acceso al sistema de archivos desde la interfaz de kiosco. En resumen, cuando el usuario hace clic en el

```
def open_netflix_kiosk(self):
    url_servicios_video = "https://www.netflix.com"
    os.system(f"chromium-browser --no-sandbox --kiosk {url_servicios_video}")
```

Figura 1: Función para reproducir Netflix

botón "Netflix.[en] la interfaz principal, se abre un navegador en modo quiosco que carga la página de Netflix a pantalla completa.

Para el modulo de Youtube En esta función open_youtube_kiosk, se utiliza el mismo enfoque que en el caso de Netflix. Se abre el navegador Chromium en modo quiosco y se carga la URL de YouTube (`https://www.youtube.com`).

```
def open_youtube_kiosk(self):
    url_servicios_video = "https://www.youtube.com"
    os.system(f"chromium-browser --no-sandbox --kiosk {url_servicios_video}")
```

Figura 2: Función para reproducir Youtube

Cuando el usuario hace clic en el botón "YouTube.[en] la interfaz principal, esta función se ejecuta, abriendo el navegador en modo quiosco para mostrar la página de YouTube a pantalla completa.

Para la búsqueda de Google En esta función open_google_kiosk, nuevamente se utiliza el módulo os para ejecutar un comando del sistema. Se abre el navegador Chromium en modo quiosco y se carga la URL de Google (`https://www.google.com`).

```
def open_google_kiosk(self):
    url_servicios_video = "https://www.google.com"
    os.system(f"chromium-browser --no-sandbox --kiosk {url_servicios_video}")
```

Figura 3: Función para abrir Google

Para el uso de una memoria USB

La reproducción de contenido desde una unidad USB implica seleccionar archivos multimedia como imágenes, música o videos y reproducirlos. Hay una función llamada play_usb_content que permite al usuario seleccionar un directorio en una unidad USB y luego muestra una interfaz para reproducir el contenido multimedia presente en ese directorio.

```
def play_usb_content(self):
    usb_path = filedialog.askdirectory(title="Seleccionar USB")
    if usb_path:
        self.stop_media_player()  # Detener la reproduccion actual si la hay
        self.show_usb_content_interface(usb_path)
```

Figura 4: Función para reproducir contenido de una USB

Cuando se llama a esta función, se utiliza filedialog.askdirectory para permitir al usuario seleccionar un directorio (unidad USB). Si se selecciona un directorio, se detiene la reproducción actual (si la hay) y se llama a la función show_usb_content_interface con la ruta del directorio seleccionado.

La función show_usb_content_interface crea una nueva ventana (Toplevel) que muestra una lista de archivos multimedia presentes en el directorio USB seleccionado. Luego, el usuario puede seleccionar un archivo multimedia y reproducirlo.

Es importante tener en cuenta que la ejecución de los programas depende de que Chromium esté instalado en el sistema y disponible en el PATH del sistema. Además, el uso de os.system para ejecutar comandos del sistema puede variar según el sistema operativo en el que se esté ejecutando la aplicación, al trabajar con Linux se utiliza la estructura presentada.

Integración de los componentes de software o módulos en una solución de software

Para hacer la integración de cada módulo de software se implementa un menú principal con varios botones de acceso que permiten realizar diferentes acciones.

```
# Crear botones de acceso
buttons_info = [
    {"name": "Netflix", "icon": "netflix.png", "command": self.open_netflix_kiosk},
    {"name": "YouTube", "icon": "youtube.png", "command": self.open_youtube_kiosk},
    {"name": "Google", "icon": "google.png", "command": self.open_google_kiosk},
    {"name": "Reproducir", "icon": "usb.png", "command": self.play_usb_content},
    {"name": "Reproducir con VLC", "icon": "vlc.png", "command": self.play_vlc_content}
]
```

Figura 5: Función buttons_info

En esta sección del código, se define la información para cada botón del menú en la lista buttons_info. Cada entrada en la lista contiene el nombre del botón, el nombre del icono asociado y la función que se ejecutará cuando se haga clic en el botón.

Luego, se utiliza un bucle for para crear y configurar cada botón con la información proporcionada. Cada botón es un objeto ttk.Button de Tkinter. Se le asigna un texto (text) con el nombre del botón, un ícono (image) cargado desde un archivo de imagen, y se especifica la función (command) que se ejecutará cuando se haga clic en el botón.

Figura 6: Estilos para botones

En esta sección, se crea un objeto ttk.Style para configurar los estilos de los botones. Se configura un estilo llamado "WhiteButton.TButtonçon un fondo blanco, texto negro y un borde blanco.

Figura 7: Añadir título y fecha

En estas líneas, se añaden etiquetas (Labels) en la parte superior izquierda y derecha de la ventana para mostrar el título "Smart TVz la hora actual.

El menú principal consta de botones con diferentes opciones, como Netflix, YouTube, Google, reproducción de contenido desde una unidad USB y reproducción de contenido con VLC. Cada botón está asociado a una función específica que se ejecuta cuando el usuario hace clic en el botón correspondiente. Además, se configuran estilos para los botones y se añaden elementos adicionales, como el título y la hora en la parte superior de la ventana principal.

Conclusiones

El desarrollo de este proyecto sin duda presentó un reto para el equipo, para poder desarrollarlo fue necesario entender en primer lugar los requerimientos correctamente así como las limitantes que teníamos para poder concluirlo correctamente, una vez desarrollado este análisis, se hizo la selección de los módulos correspondientes a cada librería y así poder tener las herramientas de software necesarias para poder seguir con el desarrollo correctamente, estas fueron mencionadas anteriormente.

El reto más grande que se presentó durante el proyecto fue lidiar con los errores que teníamos, el primero se presentó a la hora de utilizar la plataforma que tienen DRM (Digital Rights Management) ya que sin este permiso nos era imposible reproducir contenido de las plataformas propuestas en el proyecto, una vez resuelto esto elegir la librería de python para la interfaz fue más sencillo ya que el propio lenguaje nos proporciona una en la que podemos hacerlo de manera más sencilla (tkinter), ahora había que implementar que se iniciara automáticamente, hay varias opciones, por lo que se optó con la primera que se investigó en la que había que modificar el documento de autostart, al hacer este cambio basta con reinciar la tarjeta y acceder a la aplicación.

Sin duda se fue un reto complicado en el que se tuvieron que poner en marcha muchos conceptos que hemos revisado a lo largo de las clases de teoría como de laboratorio para poder lograr concluir, a pesar de no haber sido en el tiempo esperado, se logró presentar un producto funcional.

Referencias

[Python, 2023a] Python (2023a). `https://docs.python.org/3/library/tkinter.html`.

[Python, 2023b] Python (2023b). `https://docs.python.org/es/3.10/library/os.html`.

[Raspberry, 2023] Raspberry (2023). `https://www.raspberrypi.com/documentation/computers/os.html`.

[VLC, 2023] VLC (2023). `https://www.videolan.org/vlc/libvlc.html`.