Proyecto Final

Centro Multimedia.

Fundamentos de Sistemas Embebidos

Autor: Baez Cadena Duilio Giovani

1. Objetivo

Este proyecto busca desarrollar un centro multimedia autónomo utilizando una Raspberry Pi. El sistema, diseñado como un sistema embebido, permite gestionar contenido de entretenimiento, incluyendo reproducción de películas, música, fotografías y videos desde servicios en línea o dispositivos de almacenamiento extraíbles.

2. Introducción

En este informe presenta un análisis detallado del diseño, configuración y desarrollo de un sistema multimedia integrado, construido sobre una Raspberry Pi. Se describen los elementos fundamentales del proyecto, incluyendo la implementación de módulos específicos que permiten la reproducción de películas, videos, música y fotografías, tanto desde servicios de streaming como desde dispositivos de almacenamiento USB, ofreciendo una solución eficiente y versátil para el entretenimiento digital (Raspberry Pi Foundation, 2021).

3. Antecedentes

Los sistemas embebidos son tecnologías diseñadas para realizar tareas específicas de manera eficiente, integrándose directamente en el hardware. La Raspberry Pi, una microcomputadora compacta y económica, ha revolucionado el desarrollo de proyectos tecnológicos, combinando versatilidad y accesibilidad. Su capacidad para ejecutar sistemas operativos optimizados la convierte en una herramienta ideal para soluciones como centros multimedia (Raspberry Pi Foundation, 2021).

Los centros multimedia han evolucionado hacia sistemas inteligentes que permiten gestionar contenido audiovisual desde dispositivos externos o servicios en línea. Este proyecto aprovecha la flexibilidad de la Raspberry Pi para implementar un sistema eficiente y adaptable, ofreciendo una experiencia de entretenimiento accesible y funcional (Estrada, 2014).

4. Materiales

- Raspberry Pi 4 Modelo B (Modelo 3 b+ puede funcionar)
- Tarjeta de memoria microSD de 8GB (con sistema operativo Raspbian booteable)
- Conexión a internet (alámbrica o inalámbrica)
- Monitor o dispositivo de salida de video (con soporte HDMI o VGA con adaptador)
- Adaptador micro HDMI (o HDMI a micro HDMI)
- Mouse y teclado (con entrada USB)
- Dispositivo de audio (con conexión 3.5mm o bocinas externas)
- Fuente de alimentación (regulada de 5V, con al menos 2A de salida)

Descripción del funcionamiento de la Raspberry Pi

La Raspberry Pi es una placa base que integra los componentes esenciales de un ordenador, incluyendo un procesador ARM de hasta 1500 MHz, un chip gráfico, memoria RAM y múltiples entradas para USB, micro HDMI, audio, Ethernet, WiFi y Bluetooth. Utiliza el sistema operativo Raspbian, una distribución de Linux basada en Debian, optimizada para su funcionamiento.

Para operar, se inserta una microSD con el sistema operativo, se conecta a un monitor mediante un adaptador micro HDMI, y se añade un mouse, teclado y fuente de alimentación. Una vez encendida, permite trabajar como un ordenador convencional. Este proyecto utiliza Python como lenguaje principal, con las librerías necesarias instaladas desde la terminal. La ejecución y prueba de los programas también se realizan directamente desde la terminal, simplificando el flujo de trabajo.

Cuidado de la salud y riesgos

Manipular la tarjeta con cuidado para evitar quemaduras, especialmente al operar durante largos períodos, ya que puede calentarse. Asegúrate de desconectar el dispositivo al realizar ajustes para evitar riesgo de electrocución.

Cuidado de componentes electrónicos

Evita tocar directamente los terminales de la tarjeta y la memoria microSD para prevenir daños por estática o acumulación de grasa. Usa una pulsera antiestática al manipularlos y mantén el dispositivo dentro de sus límites de temperatura operativa.

5. Configuración de la tarjeta controladora

Configuración del sistema operativo

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:

1. **Tkinter:** Librería para crear aplicaciones de escritorio. Instálala con:

   ```
   $ sudo apt-get install python3-tk
   ```

2. **Pyautogui:** Librería de Python para controlar el mouse y el teclado de manera automatizada. Instálala con:

   ```
   $ pip3 install pyautogui
   ```

3. **Pygame:** Librería para la creación de aplicaciones multimedia, como un reproductor de música. Instálala con:

   ```
   $ pip3 install pygame
   ```

4. **Subprocess:** Librería para ejecutar comandos del sistema operativo. Instálala con:

   ```
   $ sudo apt-get install python-pygame
   ```

5. **OpenCV:** Librería utilizada para la reproducción de fotos. Instálala con:

   ```
   $ pip3 install opencv-python-headless
   ```

6. **Pyudev:** Librería que permite interactuar con el sistema de administración de dispositivos. Instálala con:

   ```
   $ sudo apt-get install python3-pyudev
   ```

7. **Keyboard:** Librería que permite detectar y simular eventos del teclado. Instálala con:

   ```
   $ pip3 install keyboard
   ```

8. **Widevine:** Librería necesaria para la decodificación de contenido protegido por DRM. Instálala con:

   ```
   $ sudo apt-get install widevine
   ```

9. **Pyttsx3:** Librería de síntesis de voz para convertir texto a voz. Instálala con:

   ```
   $ pip3 install pyttsx3
   ```

6. Desarrollo de los componentes de software

Desarrollo de Módulos

En este proyecto, se desarrollaron varios módulos clave para la **conexión WiFi**, la **detección de USB** y el **apagado del sistema** utilizando Python y diversas librerías. Además, el equipo colaboró en la parte de **reproducción multimedia**, que incluye la visualización de fotos, la reproducción de videos y música desde una unidad USB.

1. Módulo de Conexión WiFi

El módulo de conexión WiFi se encargó de detectar redes disponibles y permitir la conexión a una red específica. Se utilizó el comando `iwlist wlan0 scan` para obtener las redes WiFi cercanas, y luego se modificó el archivo de configuración `wpa_supplicant.conf` para conectar la Raspberry Pi a la red deseada. La función para obtener las redes WiFi disponibles es la siguiente:

```
import subprocess
import re

def obtener_redes_wifi():
    # Comando para obtener la lista de redes Wi-Fi disponibles
    comando = 'sudo iwlist wlan0 scan'
    # Ejecutar el comando en la terminal y capturar la salida
    resultado = subprocess.check_output(comando, shell=True, text=True)
    # Buscar los SSID en la salida utilizando expresiones regulares
    ssids = re.findall(r'ESSID:"(.*?)"', resultado)
    return ssids
```

Para conectar a una red WiFi, se modificó el archivo `wpa_supplicant.conf` con el SSID y la clave de la red seleccionada, otorgando permisos de edición y reiniciando el sistema para aplicar los cambios.

```
def conectarRed(ssid, key):
    arch = '/etc/wpa_supplicant/wpa_supplicant.conf'
    subprocess.call(['sudo', "chmod", "777", arch])  # Se da permiso de edición
    with open(arch, 'w') as fp:
        fp.write('ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev')
        fp.write('\nupdate_config=1')
        fp.write('\ncountry=MX')
        fp.write('\nnetwork={\n')
        fp.write(f'\tssid="{ssid}"\n')
        fp.write(f'\tpsk="{key}"\n')
        fp.write('\tkey_mgmt=WPA-PSK\n')
        fp.write('}')
    # Reiniciar el sistema para que los cambios tengan efecto
    subprocess.run(["reboot"])
```

2. Módulo de Detección de Dispositivos USB

El módulo de detección de dispositivos USB fue implementado utilizando la librería `pyudev`, que permite monitorear los eventos de cambio en los dispositivos conectados a la Raspberry Pi. Este módulo detecta si se conecta un dispositivo USB y actualiza la interfaz de usuario en consecuencia, mostrando u ocultando botones relacionados con la unidad USB.

```
import pyudev
from pyudev import Monitor, MonitorObserver

def detectar_usb(observer, device):
    usb_conectada = False
    # Iterar sobre los dispositivos de bloque en busca de una unidad USB
    for dev in context.list_devices(subsystem='block', DEVTYPE='disk'):
        if dev.get('ID_BUS') == 'usb':
```

```
            usb_conectada = True
            break
    if usb_conectada:
        botonNoUsb.place_forget()  # Ocultar el botón de "sin USB"
        botonUsb.pack()  # Mostrar el botón de "con USB"
        botonUsb.place(relx=0.2, rely=0.7)
    else:
        botonUsb.place_forget()  # Ocultar el botón de "con USB"
        botonNoUsb.pack()  # Mostrar el botón de "sin USB"
        botonNoUsb.place(relx=0.2, rely=0.7)

context = pyudev.Context()
monitor = Monitor.from_netlink(context)
monitor.filter_by(subsystem='block')
observer = MonitorObserver(monitor, detectar_usb)
observer.start()
```

3. Módulo de Apagado del Sistema

La función de apagado del sistema fue desarrollada utilizando la librería `subprocess`, que permite ejecutar comandos del sistema operativo directamente desde Python. Esta funcionalidad es útil para apagar la Raspberry Pi de manera controlada desde la interfaz gráfica, mediante un botón.

```
import subprocess

def apagar():
    subprocess.call(['shutdown', "-h", "now"])

def funcion_Salir():
    labelSalir = Label(window, text="Hasta pronto.",
    fg="#fff", bg=colorFondo, font=("Verdana Bold", 60))
    labelSalir.place(relx=0, rely=0, relheight=1, relwidth=1)
    window.after(5000, apagar)  # Retraso de tiempo antes de apagar
```

4. Reproducción Multimedia

La parte de **reproducción multimedia** fue desarrollada en equipo. Cada miembro se encargó de una parte específica: uno de los integrantes se centró en la reproducción de música, otro en la visualización de imágenes y un tercero en la reproducción de videos. Se utilizaron la librería `Pygame` para la reproducción de música y la librería `OpenCV` para mostrar imágenes y videos. El sistema puede detectar automáticamente los archivos multimedia en una unidad USB y reproducirlos según el tipo de contenido.

1. Función para Verificar y Reproducir el Contenido Multimedia

La función `check_usb` recibe el parámetro `media_type`, que determina el tipo de contenido multimedia a reproducir: música, fotos o videos.

```
def check_usb(self, media_type):
    usb_path = "/media/pi"  # Ruta donde se montan las USB en Raspberry Pi
    # Obtener la lista de archivos en la unidad USB
```

```
    usb_files = os.listdir(usb_path)
    if len(usb_files) > 0:
        usb = os.path.join(usb_path, usb_files[0])
         # Obtener la ruta completa del primer archivo en la unidad
        if media_type == "Música":
            # Iniciar un hilo para reproducir música
            self.media_thread = threading.Thread(target=self.play_music, args=(usb,))
            self.media_thread.start()
        elif media_type == "Fotos":
            # Iniciar un hilo para mostrar imágenes
            self.media_thread = threading.Thread(target=self.show_images, args=(usb,))
            self.media_thread.start()
        elif media_type == "Videos":
            # Iniciar un hilo para reproducir videos
            self.media_thread = threading.Thread(target=self.play_videos, args=(usb,))
            self.media_thread.start()
        self.media_playing = True  # Establecer la bandera de reproducción en True
        self.media_thread.start()  # Iniciar el hilo de reproducción de multimedia
```

2. Reproducción de Fotos

La función `show_images` recibe la ruta de la unidad USB como parámetro y se encarga de cargar las imágenes con extensiones válidas (`.jpg`, `.png`). Estas imágenes se muestran en pantalla de manera continua, creando una presentación de diapositivas.

```
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")):
        # Si el archivo tiene una extensión válida, se agrega a la lista
            image_files.append(file)

    # Reproducir las imágenes en bucle mientras se esté reproduciendo multimedia
    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
```

La función `show_image` utiliza OpenCV para mostrar cada imagen en pantalla completa.

```
def show_image(self, image_path):
    image = cv2.imread(image_path)  # Leer la imagen utilizando OpenCV
    cv2.namedWindow("Image", cv2.WND_PROP_FULLSCREEN)
     # Crear una ventana con nombre "Image"
    # Establecer la propiedad de pantalla completa en la ventana
    cv2.setWindowProperty("Image", cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
    cv2.imshow("Image", image)  # Mostrar la imagen en la ventana
    key = cv2.waitKey(3000)  # Esperar durante 3 segundos
```

```
    if key == ord("q"):  # Si se presiona la tecla 'q', detener la reproducción
        self.media_playing = False
    cv2.destroyAllWindows()  # Cerrar todas las ventanas abiertas por OpenCV
```

3. Reproducción de Videos

La función `play_videos` se encarga de identificar los archivos de video en la unidad USB (con extensiones `.mp4` o `.avi`) y reproducirlos uno por uno.

```
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")):
            video_files.append(file)  # Si el archivo tiene una extensión válida
            #se agrega a la lista

    # Reproducir los videos en bucle mientras se esté reproduciendo multimedia
    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
            self.play_video(video_path)  # Reproducir el video
```

4. Reproducción de Música

La función `play_music` se encarga de cargar y reproducir los archivos de música en formato `.mp3` desde la unidad USB. Utiliza la librería Pygame para manejar la reproducción de audio.

```
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"):
            music_files.append(file)  # Si el archivo tiene una extensión válida
            #se agrega a la lista

    # Reproducir la música en bucle mientras se esté reproduciendo multimedia
    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
            pygame.mixer.init()  # módulo mixer de Pygame para reproducir música
            pygame.mixer.music.load(music_path)  # Cargar el archivo en el reproductor
            pygame.mixer.music.play()  # Reproducir la música
            while pygame.mixer.music.get_busy() and self.media_playing:
                continue
```

7. Video del funcionamiento

El funcionamiento del proyecto está disponible en el siguiente enlace: Funcionamiento Centro Multimedia. https://youtu.be/Sw32-vD0ufk

8. Repositorio del código

El código fuente del proyecto está disponible en el siguiente enlace: Repositorio en GitHub.

9. Cuestionario

1. ¿Cómo modificarías la función `detectar_usb` para que, además de detectar si una unidad USB está conectada, también muestre el nombre del dispositivo USB (modelo o marca) en la interfaz gráfica?

2. ¿Qué cambios realizarías en la función `play_videos` para que, al reproducir un video desde la unidad USB, se muestre en la pantalla un contador que indique el tiempo transcurrido del video en segundos?

3. En la función `play_music`, ¿cómo podrías modificar el código para que, al reproducir una canción desde la unidad USB, se imprima en la consola el título de la canción y el nombre del artista correspondiente?

4. En la función `show_image`, ¿cómo realizarías las modificaciones necesarias para que las imágenes mostradas desde la unidad USB se muestren en escala de grises en lugar de color?

10. Conclusiones

Se implementó con éxito un centro multimedia basado en una Raspberry Pi, capaz de gestionar contenido de streaming como Netflix y Spotify, además de reproducir archivos almacenados en dispositivos USB. Este proyecto permitió profundizar en los principios de los sistemas embebidos y en los aspectos clave para su desarrollo, incluyendo configuraciones de software y hardware para un rendimiento eficiente. Uno de los mayores desafíos fue integrar la funcionalidad de detección automática de dispositivos extraíbles y configurar el inicio de la aplicación tras la carga del sistema operativo, lo que representó una experiencia enriquecedora y una valiosa oportunidad de aprendizaje sobre el funcionamiento y las capacidades de la Raspberry Pi.

11. Referencias

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

Estrada, F. (2014). Sistema multimedia Raspberry. Madrid, España: Ediciones Díaz de Santos.

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). What is a Raspberry Pi?. Recuperado de https://www.raspberrypi.org