«Proyecto Final»

Fundamentos de Sistemas Embebidos

Autor: Charlie Brian Monterrubio López

Entrega: lunes 18 de agosto, 2021

Objetivo

Crear un sistema embebido que simule un centro de entretenimiento multimedia para la reproducción de películas, vídeos, música y fotografías haciendo uso del sistema operativo Raspberry Pi OS.

Materiales

- Sistema operativo Raspberry Pi OS.
- Simulador de sistemas operativos VirtualBox.
- Lenguaje de programación de alto nivel, Python.

Descripción del funcionamiento de los componentes relevantes.

- Sistema operativo Raspberry Pi OS.
  Es el sistema operativo oficial y compatible para el dispositivo Raspberry, proporciona un entorno de escritorio y la mayor parte del software para poder trabajar (RaspberryPi, 2021).
- Simulador de sistemas operativos VirtualBox.
  Es un producto de software que permite la virtualización para uso empresarial y doméstico, es gratuito y de código abierto bajo la licencia GNU/GPL (VirtualBox, 2021).
- Lenguaje de programación de alto nivel, Python.
  Python es un lenguaje de programación muy popular en la actualidad, es de tipo interpretado y se caracteriza por ser muy simple en su legibilidad, soporta programación funcional y programación orientada a objetos (Python, 2021).
- Modulo python-vlc.
  Este módulo se basa en una instalación ya existente de VLC en el sistema operativo y posee una API nativa linvlc que permite la reproducción de multimedia a través de Python (Pypi, 2021).

Desarrollo de los componentes de software

1. Desarrollo de la interfaz gráfica.
   Para el desarrollo de una interfaz gráfica que pueda ser amigable con el usuario se utilizó el módulo Tkinter de Python3, el cual, proporciona varias herramientas para crear ventanas, botones, barras de estado, entre muchas otras cosas.

```
mainWindow = tk.Tk()
mainWindow.geometry("960x600")
mainWindow.title('Media Player')
mainWindow.resizable(width=False,height=False)
imgbackground = Image.open('src_img/background.jpg')
imgbackground = imgbackground.resize((960, 600), Image.ANTIALIAS) # Redimension (Alto, Ancho)
imgbackground = ImageTk.PhotoImage(imgbackground)
panel = tk.Label(mainWindow, image = imgbackground)
panel.image = imgbackground
panel.pack()
```

Segmento de código 1: Creación de la ventana principal

2. Creación de botones.
   Con el propósito de tratar de imitar algunas aplicaciones de reproducción de medios, se añadieron botones con imagen, los cuales, ejecutan una acción.

Imagen 1: Vista del botón de Netflix

```
imgNetflix = Image.open('src_img/Netflix-logo.jpg')
imgNetflix = imgNetflix.resize((250, 150), Image.ANTIALIAS) # Redimension (Alto, Ancho)
imgNetflix = ImageTk.PhotoImage(imgNetflix)
tk.Button(mainWindow, command=ml.playNetflix, image=imgNetflix, text="Netflix").place(x=50, y=20)
```

Segmento de código 2: Creación del botón Netflix

3. Implementación de las funciones de los botones.
   Para el evento de botón Netflix y HBO.

```
def playNetflix():
    var = webbrowser.open("https://www.netflix.com/mx/Login",new=2, autoraise=True)
    time.sleep(5)
    pyautogui.press('f11')

def playHBO():
    webbrowser.open("https://play.hbomax.com/page/urn:hbo:page:home",new=2, autoraise=True)
    time.sleep(5)
    pyautogui.press('f11')
```

Segmento de código 3: Evento que se dispara al presionar el botón Netflix o HBO

   Para el evento de reproductor de medios.

```
def openFile():
    """This function open files of type .bmp .png and .jpg"""
    file = fd.askopenfilename(initialdir = os.getcwd(), title = 'Seleccione archivo',

def openDireactory():
    """This function open a directory media"""
    directory = fd.askdirectory()
    print('directory: ', directory)
    ml.playMedia(directory)
```

Segmento de código 4: Evento que se dispara al presionar el botón de Medios visuales

4. Lectura de fotos.
   Para la lectura de multimedia se hizo uso del módulo vlc, que recibe como parámetros una lista de nombres de archivos y un tiempo de transición entre cada imagen.

```
def reproducirFotos(mymedia,tiempo):
    #instancia del reproductor
    vlc_instance = vlc.Instance()
    player = vlc_instance.media_list_player_new()#funcion para hacer slideshow
    Media = vlc_instance.media_list_new(mymedia)
    player.set_media_list(Media)

    #cada elemento de la lista se reproduce en pantalla por 4 segundos
    for index, name in enumerate(mymedia):
        player.play_item_at_index(index)
        time.sleep(tiempo)#el tiempo de reproduccion de las fotos, videos o musica
    player.stop()#IMPORTANTE, debe cerrarse el reproductor
```

Segmento de código 5: Lectura de imágenes

5. Lectura de música y videos.
   Para la lectura de música y videos también se hace uso del modulo vlc, que de igual forma itera sobre una lista de directorios y los va reproduciendo uno a uno.

```
def reproducirMusicaVideo(file):
    for f in file:
        vlc_instance = vlc.Instance()
        player = vlc_instance.media_player_new()
        media = vlc_instance.media_new(f)
        player.set_media(media)
        player.set_fullscreen(True)
        player.play()
        time.sleep(1.5)
        duration = player.get_length() / 1000
        time.sleep(duration)
        player.stop()
    player.stop()
```

Segmento de código 6: Lectura de audio y video

6. Conexión de la memoria UBS
   Para esta parte del proyecto se utilizo un hilo de ejecución, el cual, de manera constante esta checando si se ha conectado una memoria USB. Para el caso de Raspbian el segmento de código ésta verificando el archivo “mounts” que contiene una lista con los dispositivos montados sobre la máquina.

```
process = multiprocessing.Process(target=checkUSBconnection, args=(None,))
process.start()

mainWindow.mainloop()
process.terminate()
```

Segmento de código 7: Creación del hilo usando el módulo multiprocesing

```
def checkUSBconnection(var):
    play = True
    while True:
        d={}
        for l in open('/proc/mounts'):
            if(l[0] == '/'):
                l = l.split()
                d[l[0]] = l[1]

        if('/dev/sdb1' in d):
            eventUSB(d['/dev/sdb1'], play)
            play = False
```

Segmento de código 8: Verifica la conexión de una UBS

7. Simulación de la inserción de la memoria USB
   Al estar utilizando un entorno virtual, desde VirtualBox se agregó un disco duro de 4GB que simulara el dispositivo de almacenamiento externo.

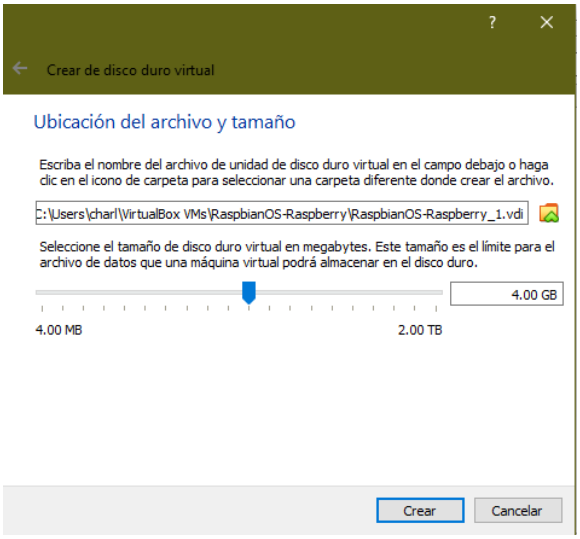

Imagen 2: Creación del disco duro en VirtualBox

Se puede visualizar desde la terminal que el disco ha sido agregado correctamente. Aparece como /dev/sdb.

```
Disco /dev/sda: 16 GiB, 17179869184 bytes, 33554432 sectores
Modelo de disco: VBOX HARDDISK
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xb8a63e0b

Disposit.  Inicio Comienzo    Final Sectores Tamaño Id Tipo
/dev/sda1  *          2048 11819007 11816960   5.7G 83 Linux
/dev/sda2         11821054 33552383 21731330  10.4G  5 Extendida
/dev/sda5         11821056 13819903  1998848   976M 82 Linux swap / Solaris
/dev/sda6         13821952 33552383 19730432   9.4G 83 Linux


Disco /dev/sdb: 4 GiB, 4294967296 bytes, 8388608 sectores
Modelo de disco: VBOX HARDDISK
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xa471eaac
```

Imagen 3: Ejecución del comando fdisk -l en termina de Raspbian

Se procede a particionarlo y darle formato, en este caso NTFS, tal como tengo mi memoria USB. Se puede visualizar como se modificó. El primer paso fue seleccionar el disco /dev/sdb.

```
pi@raspberry:~ $ sudo fdisk /dev/sdb

Bienvenido a fdisk (util-linux 2.33.1).
Los cambios solo permanecerán en la memoria, hasta que decida escribirlos.
Tenga cuidado antes de utilizar la orden de escritura.

El dispositivo no contiene una tabla de particiones reconocida.
Se ha creado una nueva etiqueta de disco DOS con el identificador de disco 0x14
842e2a.

Orden (m para obtener ayuda): m
```

Imagen 4: Selección del disco /dev/sdb

Posteriormente se añadió una nueva partición de tipo primaria y se seleccionó los sectores que abarcaría dicha partición.

```
Orden (m para obtener ayuda): n
Tipo de partición
   p   primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
   e   extendida (contenedor para particiones lógicas)
Seleccionar (valor predeterminado p): p
Número de partición (1-4, valor predeterminado 1): 1
Primer sector (2048-8388607, valor predeterminado 2048):
Último sector, +/-sectores o +/-tamaño{K,M,G,T,P} (2048-8388607, valor predeter
minado 8388607):
```

Imagen 5: Partición del disco /dev/sdb

Finalmente se cambió el formato a NTFS y se escribió la tabla de particiones correctamente.

```
Orden (m para obtener ayuda): t
Se ha seleccionado la partición 1
Código hexadecimal (escriba L para ver todos los códigos): 7
Se ha cambiado el tipo de la partición 'Linux' a 'HPFS/NTFS/exFAT'.

Orden (m para obtener ayuda): w
Se ha modificado la tabla de particiones.
Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.
```

Imagen 6: Escritura de la tabla de particiones.

Posteriormente se comprobó que el procedimiento había resultado.

```
Disco /dev/sdb: 4 GiB, 4294967296 bytes, 8388608 sectores
Modelo de disco: VBOX HARDDISK
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0x14842e2a

Disposit.  Inicio Comienzo   Final Sectores Tamaño Id Tipo
/dev/sdb1             2048 8388607  8386560     4G  7 HPFS/NTFS/exFAT
```

Imagen 7: Despliegue del comando fdisk -l

Con esto ya se puede simular que se tiene una USB insertada en el dispositivo y que es reconocida por el sistema operativo, solo faltaría montarla en /media/usb donde normalmente estaría.

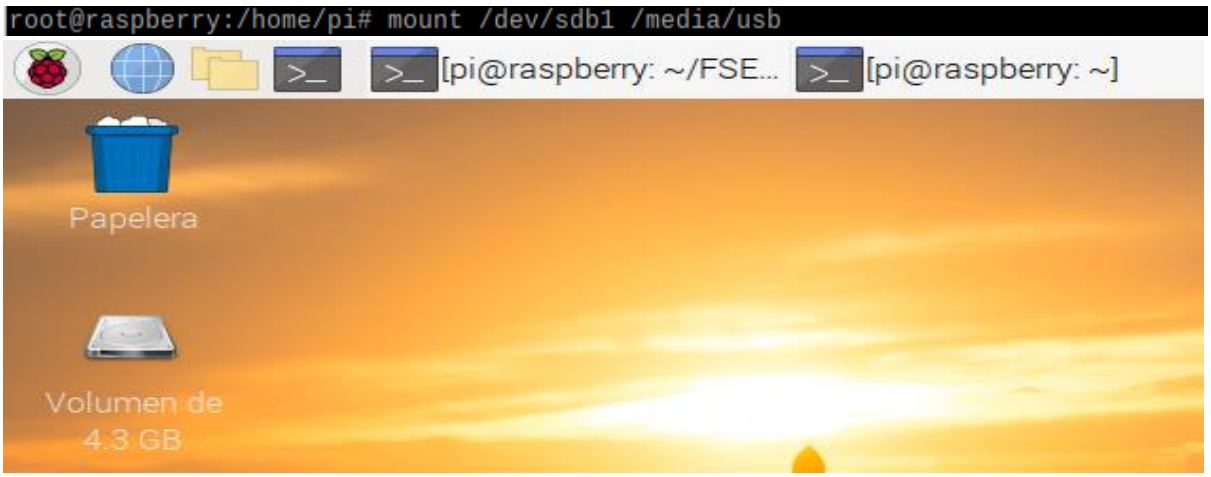

Imagen 8: Montaje del dispositivo de 4GB

Ahora para agregar contenido multimedia que debería tener la memoria USB insertada, se copia y pegan archivos de vídeo, audio e imágenes.

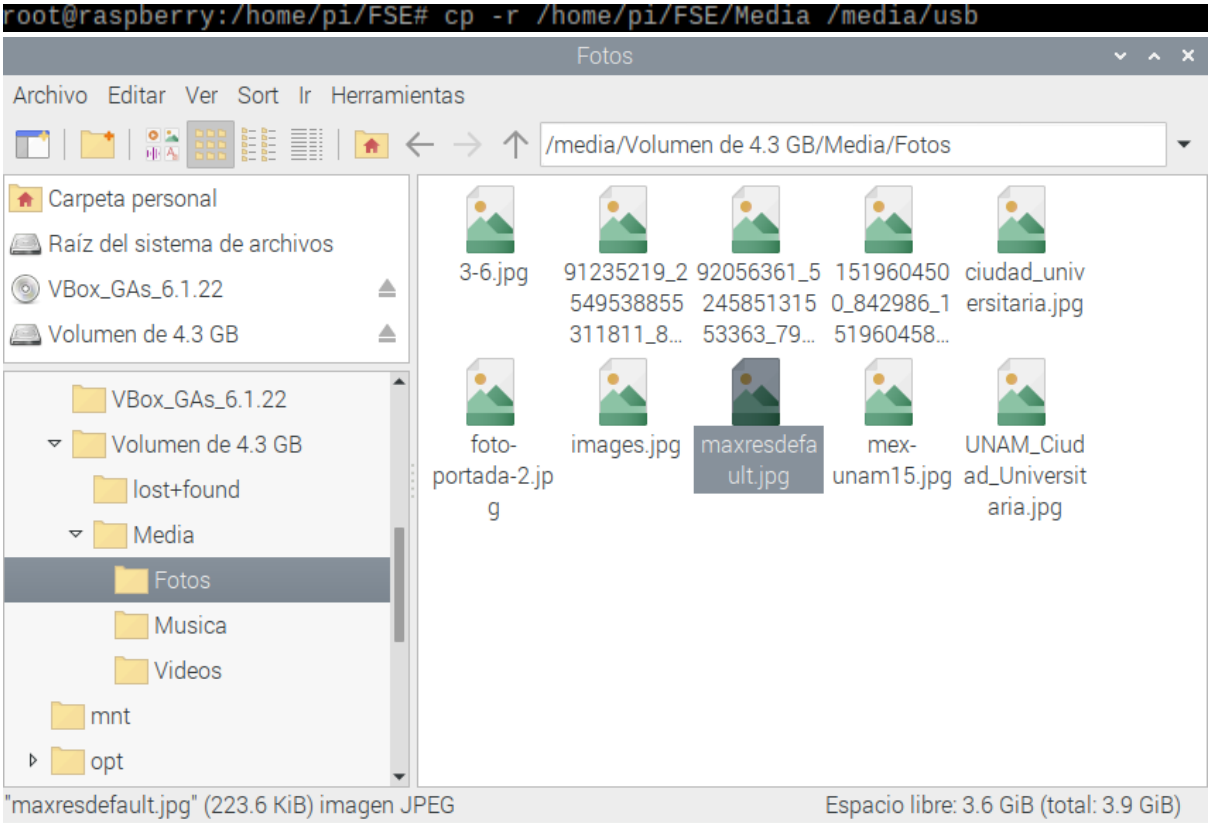

Imagen 9: Imágenes copiadas al dispositivo simulado USB

8. Dificultades

Al estar utilizando una máquina virtual, por el momento no se encontró una solución viable para poder conectar una memoria USB a la computadora y que esta fuera reconocida por la máquina virtual. Se probo la instalación de las Guest Additions (Michael, 2017), sin embargo, no se logró con éxito debido a una incompatibilidad del sistema para instalar estos componentes, por el cual, se opto por generar un disco duro virtual y desde el mismo sistema operativo montar el dispositivo simulado.

Por otro lado, se optó por utilizar el navegador nativo de Raspbian para poder acceder al contenido de streaming como Netflix y Spotify, y para disimular la apertura del navegador se coloca en pantalla completa cinco segundos después de haber iniciado.

De igual forma, también falto incluir la interacción del sistema con control remoto. Este mismo no se consideró implementar debido a la poca interacción física con la Raspberry Pi. Sin embargo, soporta perfectamente el apuntador y el teclado de la pantalla para que el usuario puede interactuar con él.

9. Máquina virtual.
   La máquina virtual que se utilizó para la simulación de este proyecto fue el sistema operativo Raspbian Buster en su versión 9.3 virtualizado desde el software VirtualBox versión 6.1 en una máquina anfitrión con sistema operativo Windows 10.

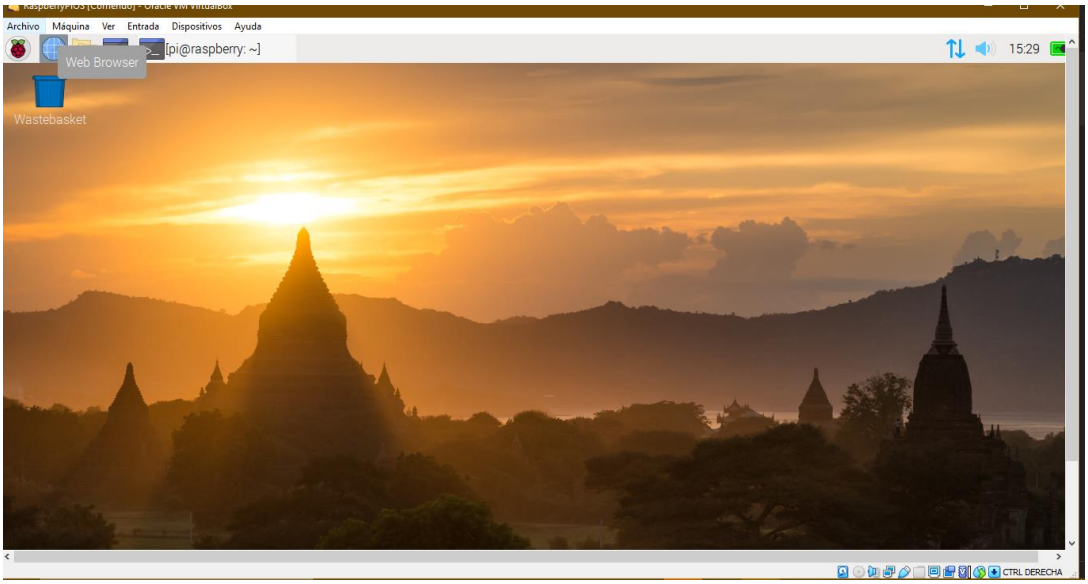

Imagen 10: Vista del entorno de escritorio de Raspbian

10. Módulos de Python
    Para el desarrollo de este sistema se utilizaron algunos módulos adicionales tales como *python-vlc*, *pillow*, *tkinter y webbrowser*, *pyautogui*, los cuales, se instalaron desde el gestor de paquetes **pip**.

```
pi@raspberry:~ $ pip install python-vlc
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting python-vlc
  Using cached https://files.pythonhosted.org/packages/e3/8e/1d0f30c4f8741f0901
4f961d49c55b1590d546e2199a54f396d288e978dd/python_vlc-3.0.11115-py3-none-any.wh
l
Installing collected packages: python-vlc
Successfully installed python-vlc-3.0.11115
```

Imagen 11: Instalación de python.vlc

Integración de los componentes de software

1. Bajar código fuente del repositorio de GitHub
   El software y el código fuente se pueden encontrar en el siguiente link: https://github.com/CharlieBrianML/ProyectoFinalFSE

```
pi@raspberry:~/FSE $ git clone https://www.github.com/CharlieBrianML/ProyectoFi
nalFSE
Clonando en 'ProyectoFinalFSE'...
warning: redirigiendo a https://github.com/CharlieBrianML/ProyectoFinalFSE.git/
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 36 (delta 8), reused 31 (delta 3), pack-reused 0
Desempaquetando objetos: 100% (36/36), listo.
```

Imagen 12: Clonando el repositorio de GitHub del proyecto final

2. Archivo ejecutable.
   El archivo principal llamado **mainMediaCenter.py** es el que contiene el código para crear la interfaz y llamar a las demás funciones.

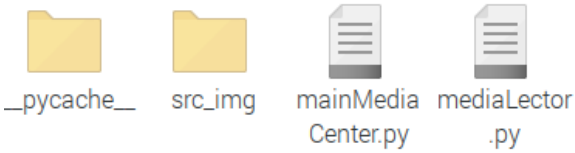

Imagen 13: Visualización de la carpeta src

3. Ejecución del software.
   La interfaz gráfica consta de una ventana simple. Esta personalizado con un fondo oscuro y con botones los cuales se ven de la siguiente forma:

Imagen 14: Vista completa de la interfaz gráfica

4. Menú del software.
   Contiene tres botones que ejecutan una acción distinta. Los iconos proporcionan una idea del medio que pueden reproducir.

Imagen 15: Vista de los botones disponibles en la interfaz

5. Ver Netflix.
   Al presionar sobre el botón de Netflix se abrirá una ventana con el inicio de sesión para esta plataforma.

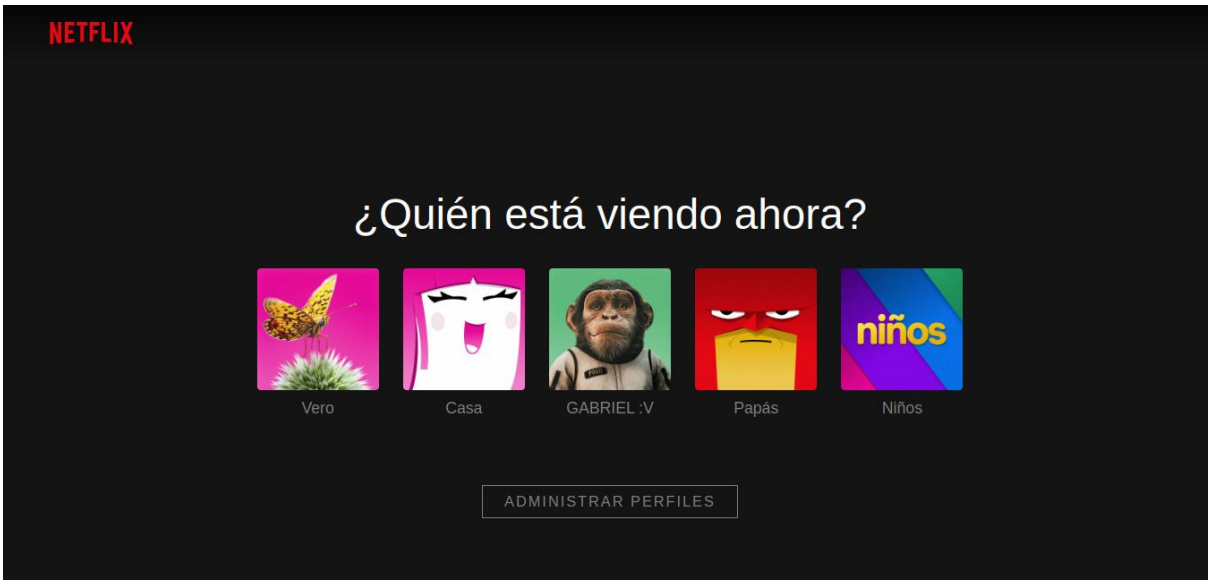

Imagen 16: Reproducción de Netflix

6. Ver HBO.
   Al presionar sobre el botón de HBO se abrirá una ventana en donde se podrá visualizar el inicio de la plataforma.

Imagen 17: Reproducción de Spotify

7. Reproducir contenido USB.
   Al presionar sobre el botón de medios se abrirá un explorador de archivos, el cual, permite seleccionar alguna carpeta que contenga archivos multimedia y estos se reproducirán de forma automática.

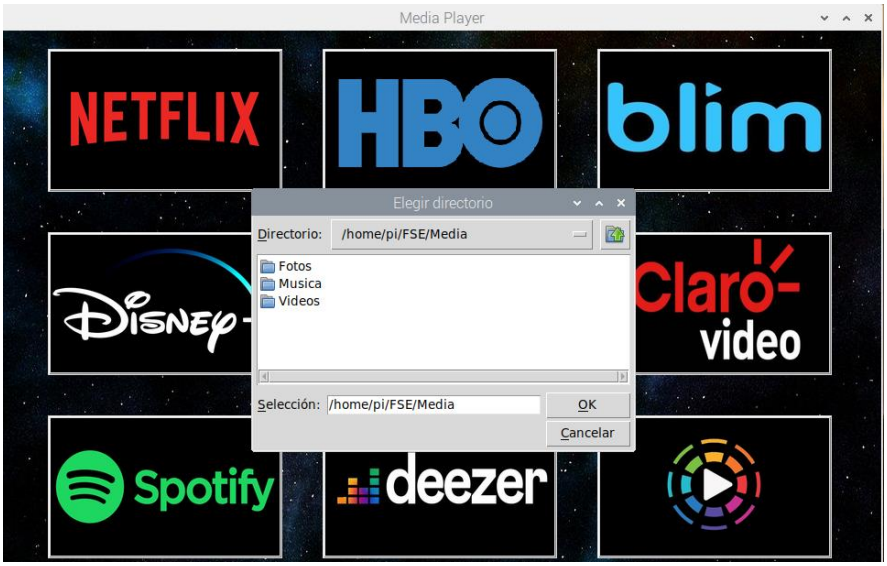

Imagen 18: Explorador de carpetas del sistema

Al seleccionar un directorio, por ejemplo, de imágenes o de video, se reproducirá automáticamente.

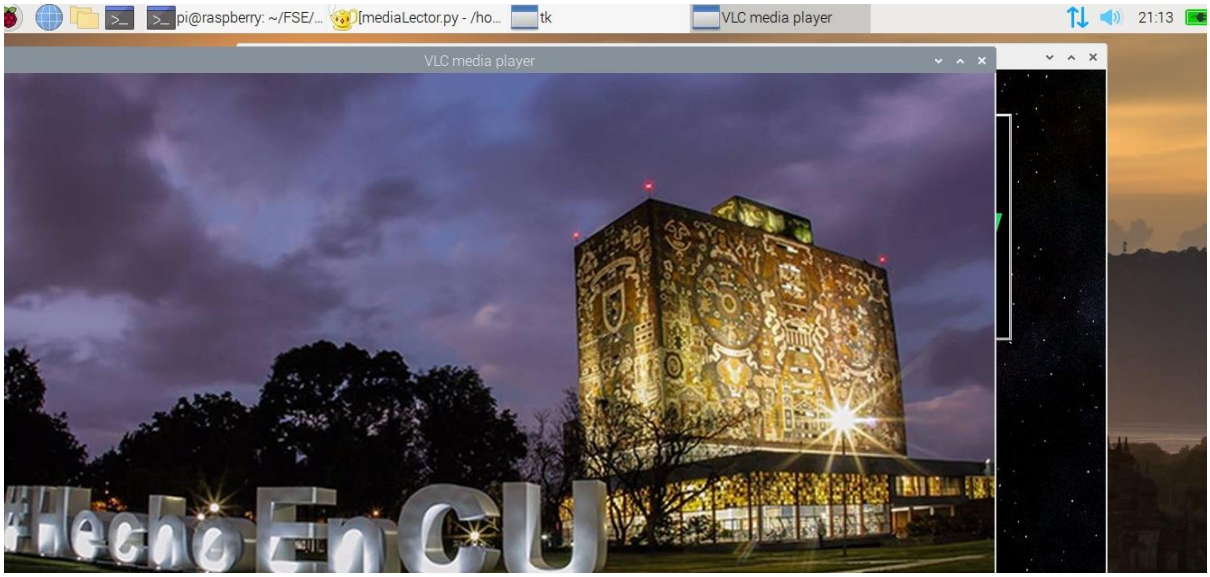

Imagen 19: Visualización de imágenes contenida en la carpeta Fotos

Imagen 20: Reproducción de contenido multimedia desde un directorio.

Con un directorio con archivos .mp3 también se reproducirá la secuencia de música, pero no es posible demostrar con una imagen así que se recomienda ver el vídeo demo subido a Drive: https://drive.google.com/file/d/1hfWF6cJY0kolzsONm9XbECiZPuwdLnx-/view?usp=sharing

Conclusiones

Con este proyecto se pudo aprender cómo se amaría un sistema embebido haciendo uso de herramientas de virtualización, aunque no se tenía físicamente la Raspberry se podría virtualizar el sistema operativo, lo cual, me hace pensar que de igual forma habría funcionado en el sistema real ya que solo había que instalar unos cuantos módulos de Python, programar un poco para utilizar las herramientas necesarias a través del script y listo, se podría ejecutar de manera exitosa lo que sería un centro multimedia con una interfaz gráfica interactiva. Quizá no fue una interfaz gráfica como suelen verse en una SmartTV, sin embargo, por efectos de tiempo ya no fue posible programarlo, aunque el conocimiento adquirido me deja las bases suficientes para consultar la información debida frente al reto de programar algún otro sistema embebido.

Bibliografía

- goto-linux. (10 de enero de 2020). *Cómo reproducir audio con VLC en Python*. Obtenido de https://goto-linux.com/es/2020/1/10/como-reproducir-audio-con-vlc-en-python/
- Michael. (25 de agosto de 2017). *Core Electronics*. Obtenido de Run Raspberry Pi x86 on your Windows Desktop Computer (Virtual Machine): https://core-electronics.com.au/tutorials/raspberry-pi/run-raspberry-pi-x86-on-your-windows-desktop-computer-virtual-machine.html
- Pypi. (2021). *python-vlc 3.0.11115*. Obtenido de https://pypi.org/project/python-vlc/
- Python. (2021). *Welcome to Python*. Obtenido de https://www.python.org/
- RaspberryPi. (2021). *RaspberryPi*. Obtenido de https://www.raspberrypi.org/software/raspberry-pi-desktop/
- Sergio. (27 de marzo de 2020). *Raspberry para novatos*. Obtenido de https://raspberryparanovatos.com/tutoriales/drm-navegador-raspberry-pi/
- VirtualBox. (2021). *VirtualBox*. Obtenido de https://www.virtualbox.org/