Proyecto Final.
Centro Multimedia

Báez Cadena Diestefano Michel

14 Junio 2023

1 Objetivo

Elaborar un sistema embebido utilizando los conocimientos adquiridos a lo largo del semestre.

2 Introducción

El presente proyecto se centra en la creación de un centro multimedia utilizando una Raspberry Pi. La Raspberry Pi es una pequeña computadora de bajo costo y alto rendimiento que ha ganado popularidad en los últimos años debido a su versatilidad y facilidad de uso. Este centro multimedia aprovecha al máximo las capacidades de la Raspberry Pi para ofrecer una experiencia multimedia completa y personalizada.

En la actualidad, el acceso a contenido multimedia ha experimentado un crecimiento exponencial, con una amplia variedad de formatos y opciones disponibles. Los usuarios desean tener un control total sobre su experiencia de entretenimiento, accediendo a películas, series, música y otros contenidos en un solo lugar y de manera sencilla. Es en este contexto que surge la necesidad de un centro multimedia eficiente y accesible.

El objetivo principal de este proyecto es utilizar la Raspberry Pi para crear un centro multimedia que permita la reproducción de diversos formatos de contenido, como películas, música, imágenes y videos, de manera fácil y conveniente. Además, se busca proporcionar funcionalidades adicionales, como la posibilidad de acceder a servicios de streaming, administrar una biblioteca personalizada de medios y controlar el centro multimedia a través de una interfaz intuitiva.

3 Antecedentes

En el ámbito de la tecnología y la informática, la Raspberry Pi ha ganado reconocimiento como una plataforma versátil y de bajo costo que permite una amplia gama de aplicaciones. Su capacidad de procesamiento y su capacidad de interfaz con diversos componentes la convierten en una opción popular para proyectos de bricolaje y desarrollo de software.

En este contexto, surge la necesidad de crear soluciones personalizadas que aprovechen al máximo las capacidades de la Raspberry Pi. Uno de los campos en los que se ha explorado su potencial es el desarrollo de centros multimedia que permitan a los usuarios disfrutar de una experiencia de entretenimiento completa y personalizada.

En lugar de utilizar soluciones preexistentes como Kodi, este proyecto se basa en el desarrollo de una aplicación personalizada utilizando el lenguaje de programación Python y la biblioteca Tkinter para la creación de interfaces gráficas de usuario (GUI, por sus siglas en inglés). Tkinter es una herramienta ampliamente utilizada y compatible con la Raspberry Pi, que permite la creación de interfaces gráficas intuitivas y atractivas.

El enfoque principal del proyecto es utilizar Python y Tkinter para desarrollar una interfaz de usuario interactiva que permita a los usuarios realizar diversas acciones, como abrir aplicaciones, configurar la conexión

a Internet y leer archivos almacenados en una unidad USB. Esto proporcionará a los usuarios un control completo sobre su centro multimedia y les permitirá adaptarlo según sus necesidades y preferencias individuales.

El desarrollo de una aplicación personalizada ofrece la ventaja de adaptar el centro multimedia a las necesidades específicas del usuario, brindando una experiencia más personalizada y flexible. Además, el uso de Python como lenguaje de programación ofrece una gran cantidad de bibliotecas y funcionalidades que facilitan la implementación de diversas características y la integración de componentes adicionales.

En resumen, este proyecto se centra en el desarrollo de un centro multimedia basado en Raspberry Pi utilizando Python y Tkinter. El objetivo es proporcionar una interfaz de usuario interactiva y personalizable que permita a los usuarios disfrutar de contenido multimedia, configurar su conexión a Internet y acceder a archivos almacenados en una unidad USB, todo ello en un dispositivo compacto y de bajo costo.

4 Materiales

- Raspberry Pi 3B+ con su respectivo eliminador 5V 2A
- Teclado inalambrico tipo "Gamepad" para poder manipular el cursor y el teclado cuando sea requerido.
- Monitor con entrada HDMI.
- Cable HDMI-HDMI para conectar la tarjeta al monitor.

5 Desarrollo

5.1 Información sobre el cuidado de la salud y advertencias de riesgos

- Manipulación segura: Al manipular la Raspberry Pi o sus componentes, es importante hacerlo con cuidado y evitar aplicar fuerza excesiva. Asegúrese de no dejar caer la Raspberry Pi o exponerla a impactos fuertes que puedan dañar los componentes internos.
- Fuente de alimentación adecuada: Utilice siempre una fuente de alimentación recomendada y adecuada para la Raspberry Pi. Evite utilizar cargadores genéricos o de baja calidad, ya que podrían no proporcionar la potencia necesaria o causar fluctuaciones eléctricas que podrían dañar el dispositivo.
- Temperatura: Mantenga la Raspberry Pi en un entorno adecuado en cuanto a temperatura. Evite exponerla a altas temperaturas, ya que esto podría afectar su rendimiento o dañar los componentes. Asimismo, evite ubicarla en áreas muy frías, ya que la condensación de humedad podría dañar los circuitos electrónicos.
- Ventilación: Asegúrese de proporcionar una adecuada ventilación para evitar el sobrecalentamiento de la Raspberry Pi.

5.2 Configuración de la tarjeta controladora.

5.2.1 Configuración del sistema operativo.

- Cambiar la imagen de arranque de raspbian Quitar la imagen predefinida: Editar el archivo */boot/config.txt* y agregar al final *disable_splash = 1* para eliminar la imagen de inicio predeterminado.

  Cambiar la imagen: La imagen debe llamarse "*splash.png*", después copiar con la siguiente línea.

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

- Configurar la Raspberry en modo consola.

```
$ sudo raspi-config
```

Al ejecutar *raspi-config* abrirá la configuracion de la tarjeta, se tiene que seleccionar la opción 3 *Boot options*, despues la opción B1 *Desktop / CLI* y por último seleccionar la opción B2 *Console Autologin*; esta configuracion hará que al reiniciar arranque en modo consola.

- Configurar script de arranque Abrir el archivo *rc.local*

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

Insertar el script justo antes de *exit 0*

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

- Librerias a instalar

1. Tkinter - Librería que funciona para la creación y el desarrollo de aplicaciones de escritorio.
   ```
   $ sudo apt-get install python3-tk
   ```
2. Pyautogui - Librería de Python que permite controlar o automatizar tareas como controlar el mouse y teclado en el sistema operativo.
   ```
   $ pip3 install pyautogui
   ```
3. Pygame - Librería para el reproductor de música.
   ```
   $ pip3 install pygame
   ```
4. Subprocess - Librería para proporcionar funcionalidades para ejecutar comandos del sistema operativo.
   ```
   $ sudo apt-get install python-pygame
   ```
5. Opencv - Librería que se utilizará para la reproducción de fotos.
   ```
   $ pip3 install opencv-python-headless
   ```
6. Pyudev - Biblioteca de Python que permite interactuar con el sistema de administración de dispositivos .
   ```
   $ sudo apt-get install python3-pyudev
   ```

5.3 Desarrollo de los componentes de software.

5.3.1 Creación de la ventana

Para crear la ventana del sistema, se utiliza la librería Tkinter, que tambien nos ayudará a crear botones,label, etc.

```
# Creación de ventana principal
window = tk.Tk()
window.config(bg=colorFondo, cursor="circle")
# Geometria de la ventana
window.geometry("%dx%d" % (width, height))
# Atributos de la ventana, en este caso tiene que ser en pantalla completa
window.attributes('-fullscreen', True)
# Nombre de la ventana
window.title("Multimedia Center")
#Iniciar la ventana
window.mainloop()
```

5.3.2 Creación de botones para aplicaciones.

```
# URL's de las paginas web de servicios de streaming
urlN  = 'https://www.netflix.com/mx/'
# Creación de funciones para cada servicio de streaming
def funcion_Netflix():
  wb.open_new(urlN) #Abre el buscador con el URL asignado
  window.after(500,pantCompleta) #Abre la aplicacion en pantalla completa

img_netflix = tk.PhotoImage(file="./iconos/netflix.png")
# Creación de los botones
botonNetflix = tk.Button(window, image=img_netflix, #Se asigna la imagen al boton
                         borderwidth=0, bg=colorFondo,
                         cursor="X_cursor",
                         command = funcion_Netflix) #Se llama a la funcion
```

5.3.3 Configuración de red WiFi

Para poder obtener la lista de redes disponibles se utilizó el comando *sudo iwlist wlan0 scan* el cual nos da la lista de redes detectadas, y al realizar una extraccion de los SSID podemos mostrarlos en una lista.

```
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 la configuración de red es necesario modificar el archivo *wpa_supplicant.conf*, para esto hay que otorgar permisos al usuario para poder editarlo, y después de editarlo es necesario reiniciar el sistema para que se ejecuten los cambios.

```
def conectarRed(ssid, key):
  arch = '/etc/wpa_supplicant/wpa_supplicant.conf'
  subprocess.call(['sudo', "chmod", "777", arch]) # Se da permiso de edición
  #Entra al archivo y edita la configuración de la nueva conexión
  with open(arch, 'w') as fp:
                  fp.write('ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev')
  with open(arch, 'a') as fp:
                  fp.write('\nupdate_config=1')
  with open(arch, 'a') as fp:
                  fp.write('\ncountry=MX')
  with open(arch, 'a') as fp:
                  fp.write('\nnetwork={ \n')
  with open(arch, 'a') as fp:
                  fp.write('\tssid="{}" \n'.format(str(ssid)))
  with open(arch, 'a') as fp:
                  fp.write('\tpsk="{}" \n'.format(str(key)))
```

```
        with open(arch, 'a') as fp:
                        fp.write('\tkey_mgmt=WPA-PSK')
        with open(arch, 'a') as fp:
                        fp.write('\n}')

      #Reinicio para guardar y ejecutar los cambios en la red.
      def reinicio():
        subprocess.run(["reboot"])
```

5.4 Detectar un dispositivo USB

Para detectar una unidad USB en tiempo real se utiliza la biblioteca pyudev para monitorear los eventos de cambio en los dispositivos de bloque y realiza acciones correspondientes según la detección de una unidad USB.

```
def detectar_usb(observer, device):
  # Función para detectar la conexión de una unidad USB

  # Variable para almacenar el estado de conexión de una unidad USB
  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':
      # Si se encuentra una unidad USB, se establece usb_conectada en True y se finaliza el bucle
      usb_conectada = True
      break

  if usb_conectada:
    # Si se detecta una unidad USB, se oculta el botón "botonNoUsb" y se muestra el botón "botonUsb"
    botonNoUsb.place_forget()
    botonUsb.pack()
    botonUsb.place(relx=0.2, rely=0.7)
  else:
    # Si no se detecta una unidad USB, se oculta el botón "botonUsb" y se muestra el botón "botonNoUsb"
    botonUsb.place_forget()
    botonNoUsb.pack()
    botonNoUsb.place(relx=0.2, rely=0.7)

  # Crear un objeto Context de pyudev
  context = Context()

  # Crear un monitor para detectar eventos de cambio en los dispositivos de bloque
  monitor = Monitor.from_netlink(context)
  monitor.filter_by(subsystem='block')

  # Crear un observador para monitorear los eventos de cambio
  observer = MonitorObserver(monitor, detectar_usb)
  observer.start()
```

5.5 Reproducción multimedia

La función *checkUsb* recibe como parámetro *mediaType*, que indica el tipo de contenido multimedia a reproducir (Música, Fotos o Videos).

```
def check_usb(self, media_type):
  # Función para verificar si una unidad USB está conectada
  usb_path = "/media/pi"  # Ruta donde se montan las USB en Raspberry Pi
  usb_files = os.listdir(usb_path)  # Obtener la lista de archivos en la ruta de la unidad USB

    if len(usb_files) > 0:
      # Si se encuentran archivos en la unidad USB
      # Obtener la ruta completa del primer archivo en la unidad
      usb = os.path.join(usb_path, usb_files[0])

      if media_type == "Música":
        # Si el tipo de contenido multimedia es 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":
        # Si el tipo de contenido multimedia es 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":
        # Si el tipo de contenido multimedia es 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 de multimedia en True
    self.media_thread.start()  # Iniciar el hilo de reproducción de multimedia
```

5.5.1 Leer y reproducir Fotos

La función show_images recibe la ruta de la unidad USB como parámetro (usb). Primero, crea una lista image_files para almacenar los nombres de archivo de las imágenes. Luego, itera sobre los archivos en la unidad USB y verifica si tienen una extensión válida de imagen (".jpg" o ".png"). Si es así, agrega el nombre de archivo a la lista image_files. Después, inicia un bucle que se ejecutará mientras self.media_playing sea verdadero. Dentro de este bucle, itera sobre la lista image_files y muestra cada imagen utilizando la función self.show_image.

```
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 .jpg o .png, se agrega a la lista de archivos de imagen
      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 utilizando la función show_image

# Función para mostrar una 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', se establece la bandera media_playing en False para detener
    self.media_playing = False

  cv2.destroyAllWindows()  # Cerrar todas las ventanas abiertas por OpenCV
```

5.5.2 Leer y reproducir videos

La función play_videos recibe la ruta de la unidad USB como parámetro (usb). Primero, crea una lista video_files para almacenar los nombres de archivo de los videos. Luego, itera sobre los archivos en la unidad USB y verifica si tienen una extensión válida de video (".mp4" o ".avi"). Si es así, agrega el nombre de archivo a la lista video_files. Después, inicia un bucle que se ejecutará mientras self.media_playing sea verdadero. Dentro de este bucle, itera sobre la lista video_files y reproduce cada video utilizando la función self.play_video.

```
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 .mp4 o .avi, se agrega a la lista de archivos de video
      video_files.append(file)

  # 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 utilizando la función play_video
```

5.5.3 Leer y reproducir música

La función play_music toma la ruta de la unidad USB (usb) como parámetro. Dentro de la función, se crea una lista llamada music_files para almacenar los nombres de archivo de música.

Después, hay un bucle while que se ejecuta mientras self.media_playing es verdadero. Este bucle itera sobre los archivos de música en music_files y reproduce cada archivo de música utilizando el módulo pygame.mixer.

Dentro del bucle, se carga el archivo de música en el reproductor de música de Pygame utilizando pygame.mixer.music.load(music_path). Luego, se reproduce la música utilizando pygame.mixer.music.play().

```
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 .mp3, se agrega a la lista de archivos de música
      music_files.append(file)

    # 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()  # Inicializar el módulo mixer de Pygame para reproducir música
      pygame.mixer.music.load(music_path)  # Cargar el archivo de música en el reproductor de música
      pygame.mixer.music.play()  # Reproducir la música

      while pygame.mixer.music.get_busy() and self.media_playing:
         # Continuar el bucle mientras la música esté reproduciéndose
         continue
```

5.5.4 Función para apagar el sistema

Para apagar el sistema necesitamos ejecutar comandos desde python, para eso se llama *subprocess* que ejecuta comandos del sistema operativo.

```
        def apagar():
          #Se ejecuta el comando shutdown
          subprocess.call(['shutdown', "-h", "now"])
        def funcion_Salir():
          labelSalir = Label(window, text="Hasta pronto.",
                   fg="#fff",    # Foreground
                   bg=colorFondo,   # Background
                   font=("Verdana Bold",60))
          #Muestra la despedida
          labelSalir.place(relx=0, rely=0, relheight=1, relwidth=1)
          #Retraso de tiempo antes de apagar
          window.after(5000, apagar)
```

6 Análisis de Costos

El análisis de costos para el sistema multimedia utilizando solo una Raspberry Pi 3 implica considerar los diferentes elementos necesarios para el proyecto. A continuación, se presentan los posibles componentes y sus costos estimados:

- Raspberry Pi 3: El costo de una Raspberry Pi 3 varía según el proveedor y la región, pero en promedio puede rondar los 45−**50 USD.**
- Tarjeta microSD: Se necesita una tarjeta microSD para almacenar el sistema operativo y los archivos relacionados. El costo de una tarjeta microSD de buena calidad puede ser de alrededor de 7−**15 USD.**
- Adaptador de corriente: Se requiere un adaptador de corriente para alimentar la Raspberry Pi 3. Puede usarse un adaptador USB estándar de 5V con al menos 2.5A de salida. El costo puede ser de aproximadamente 8−**15 USD.**
- Caja o carcasa: Para proteger la Raspberry Pi 3, se recomienda utilizar una caja o carcasa. El costo puede variar, pero generalmente se encuentra en el rango de 5−**8 USD.**
- Cable HDMI: Para conectar la Raspberry Pi 3 a un monitor o televisión, se necesita un cable HDMI. El costo puede ser de alrededor de 5−**10 USD**

El costo estimado para un sistema multimedia utilizando solo una Raspberry Pi 3 y los componentes básicos mencionados puede rondar los 65−**100 USD**

7 Conclusiones

A lo largo del semestre se realizaron diferentes prácticas para poder llegar a esta instancia con las bases necesarias para poder llevar a cabo el diseño, análisis, planeación, implementación y pruebas de un sistema embebido. Sin los conocimientos que se tomaron durante el curso no hubiera sido posible llevar a cabo este proyecto, aunque ya teníamos conocimientos previos, este semestre se fortalecieron y se dirigieron en la dirección de crear un proyecto como el que se presentó.

8 Cuestionario

1. En la función detectar_usb, modifica el código para que, además de detectar una USB conectada, se muestre el nombre del dispositivo USB detectado en la pantalla.
2. Modifica la función play_video para que, al reproducir un video, se muestre un contador en la pantalla que indique la duración del video en segundos.
3. En la función play_music, realiza los cambios necesarios para que, al reproducir una canción, se muestre en la consola el título de la canción y el artista correspondiente.
4. Realiza las modificaciones necesarias en la función show_image para que la imagen se muestre en escala de grises en lugar de color.

9 Url a video demostración

`https://youtu.be/mpO6JsOjpPc`

10 Bibliografía de consulta

- ¿Cómo configurar una red Wi-Fi en Raspberry Pi? (2019, septiembre 16). Descubrearduino.com. `https://descubrearduino.com/como-configurar-una-red-wi-fi-en-tu-raspberry-pi/`
- Carles, J. (2018, abril 1). Iniciar Raspberry Pi en modo consola o en modo gráfico. geekland; Joan Carles. `https://geekland.eu/iniciar-raspberry-pi-en-modo-consola/`

- Cómo ejecutar un comando del sistema. (s/f). Recursos Python. https://micro.recursospython.com/recursos/como-ejecutar-un-comando-del-sistema.html
- Fernandez, R. (2018, noviembre 16). Tkinter - Label, Entry, método place y grid. https://unipython.com/tkinter-label-entry-metodos-place-y-grid/
- Fromaget, P. (s/f). ¿Raspberry Pi: Cómo Iniciar Automáticamente Un Programa? Raspberrytips.es. https://raspberrytips.es/iniciar-un-programa-raspberry-pi/
- Greyrat, R. (s/f). ¿Cómo crear una ventana de pantalla completa en Tkinter? Barcelonageeks.com. https://barcelonageeks.com/como-crear-una-ventana-de-pantalla-completa-en-tkinter/