

Galán Galván Arturo David

Número de lista: 10

Brigada: 6

Proyecto Final: Consola de videojuegos *Retro*

Objetivo

El objetivo de este proyecto y tutorial es guiar a un usuario paso a paso para el desarrollo de un centro de entretenimiento emulación o consola retro de videojuegos, usando los conocimientos acerca de los sistemas embebidos adquiridos durante el curso.

Introducción

Para lograr el objetivo planteado el emulador base que se decidió utilizar fue Mednafen, además, el sistema embebido debe cumplir con las siguientes características: Reproducción de animación o imagen estática personalizada con sonido durante el arranque, arranque directo a emulador (sin escritorio), control completo de la consola usando gamepad o joystick, set de al menos 15 ROMS precargadas, que cuando se inserte un medio extraíble con ROMS, el sistema pausará la simulación, copiará las ROMS a la tarjeta de almacenamiento local y mostrará al usuario la lista de ROMS disponibles, por último, que tenga soporte para control remoto.

Para cumplir con las características se emplearon scripts de Python, uso básico de BASH y varios paquetes importantes los cuales serán explicados a más detalle en el desarrollo del tutorial.

Lista de materiales

- Raspberry Pi 4 de 8GB
- Kit de ventilación con caja, disipadores y ventilador para Raspberry Pi 4
- Mouse y teclado (USB o bluetooth).
- Memoria SD de mínimo 8GB.
- Memoria USB de 8GB.
- Control alámbrico de Xbox ONE.
- Monitor.

Descripción del funcionamiento de la tarjeta controladora

Es una tarjeta microcontroladora que cuenta con un procesador ARM, 8GB de SRAM (en nuestro caso), bluetooth integrado, puerto ethernet, dos puertos USB 3.0 y dos puertos USB 2.0, entre otras más especificaciones que tiene, en esencia, un pequeño ordenador en la palma de la mano, además de que cuenta con un sistema operativo dedicado llamado Raspbian basado en Linux. También cuenta con la posibilidad de conectar un Monitor, esta tarjeta puede servir como un escritorio de propósito general o un sistema embebido dedicado.

Información sobre el cuidado de los componentes electrónicos delicados

Se recomienda el uso del kit de ventilación de la Raspberry pi, no solo para tener un mejor desempeño, sino también para evitar que la misma tarjeta llegue a calentarse demasiado y se llegase a quemar o que los componentes de esta misma se puedan dañar.

Antecedentes

Un sistema embebido es un sistema de cómputo de propósito específico para cumplir una o pocas funciones dedicadas. Prácticamente todos están formados por software y hardware diseñados específicamente para la tarea que tienen que cumplir, e interactuando muy cercanamente entre ellos. (Pedre,

S. 2017). Para el caso de este proyecto el resultado final será, en esencia, una consola de videojuegos dedicada que no necesita cumplir otra función que no sea jugar roms y copiarlos de una memoria.

“Mednafen es un emulador multisistema portátil, que utiliza OpenGL y SDL, basado en argumentos (línea de comandos).”¹ En el caso de este proyecto los juegos que se van a poder emular serán de PSX, NES y Game Boy, es por esto por lo que se decidió escoger Mednafen para desarrollarlo ya que admite esos módulos de emulación además de muchos otros. Mednafen también tiene la capacidad de reasignar funciones y teclas de manera rápida directamente desde el mismo emulador para que esto resulte más conveniente.

El ‘directorio base’ de Mednafen es el directorio en el que almacena sus datos y busca varios datos auxiliares de forma predeterminada. El directorio será ~/.mednafen, en donde estarán almacenados los archivos más importantes, incluyendo el BIOS para los *roms* americanos. (Mednafen, 2022). También se encontrará la ruta para el comando de ejecución para los juegos y del *front-end* en /usr/games/, misma ruta que tendrá dos ejecutables, ./mednafen para la ejecución de juegos y ./mednafte para la ejecución de la GUI.

Mplayer es un reproductor de archivos de video principalmente, con un amplio rango de controladores de salida soportado. Esto ayudara a la reproducción de sonido durante el arranque de la tarjeta.

pyudev es la biblioteca de información y gestión de dispositivos y hardware de Linux. Con esto entonces se puede detectar dispositivos externos y poder gestionarlos desde el microcontrolador.

wmctrl, “una herramienta de línea de comandos para interactuar con un X Window Manager compatible con EWMH/NetWM.”² Nos servirá para controlar la configuración de pantalla completa de nuestro emulador.

El paquete de herramientas tk de Python sirve para la construcción de interfaces gráficas simples, que nos ayudaran a mostrar la lista de *roms* cuando se inserte un medio extraíble.

La herramienta *xdotool* permite la entrada del teclado y la actividad del mouse. También puede mover, cambiar o administrar ventanas, todo de manera simulada sin la necesidad de tener nada conectado. Con esto se pueden simular entradas una vez se inserte el medio extraíble.

Para utilizar el control como mouse y teclado se utiliza un programa llamado QJoyPad “que convierte el movimiento y las pulsaciones de botones en un gamepad o joystick en pulsaciones de teclas, clics del mouse y movimientos del mouse en XWindows.”³ Una gran ventaja de esta aplicación es que, una vez configuradas las teclas deseadas en el control, no abre una ventana cuando se abre, esto es útil pues no queremos ventanas innecesarias cuando se inicialice el emulador.

El uso de control remoto se ha vuelto una herramienta muy utilizada para la gestión de distintas tecnologías de información. Existe mucho software comercial de terceros que implementan este tipo de acceso. Uno de estos es VNC, que ya está directamente integrado en el sistema operativo de Raspbian, simplemente es necesario activarlo en las configuraciones.

¹ Mednafen. (2005). *Introducción a Mednafen*. <https://mednafen.github.io/>

² Ramachandran, S. (2021). *wmctrl*. <https://github.com/saravanabalagi/wmctrl>

³ Panzenböck, M. (2022). *QJoyPad 4*. <https://github.com/panzi/qjoypad>

Desarrollo

Una vez teniendo todos los materiales y tener claros los antecedentes ya establecidos, es momento de iniciar el proceso para la creación de la ‘consola’. Es importante recalcar que el resultado final no tendrá un escritorio, sin embargo, preparar todo (descargas, scripts, visualización de archivos, etc.) será mucho más sencillo y rápido en un ambiente de escritorio, posteriormente este mismo se desactivará y se tendrá un sistema mínimo.

Lo primero que se lleva a cabo, además de aspectos básicos de la configuración de la tarjeta, es descargar el repositorio de Github (adjunto al final del proyecto), el cual contiene todo lo desarrollado y necesario para la realización correcta de todo. Una vez descargado hay que configurar la ruta del proyecto:

```
# cd /home/pi

# wget https://github.com/EduardoFM11/consola-de-videojuegos-
retro/archive/refs/heads/main.zip -P /home/pi/

# unzip main.zip ; rm main.zip mv consola-de-videojuegos-retro-main/
proyecto
```

Posteriormente se debe de hacer es la instalación de todos los paquetes y aplicaciones necesarias. Mediante la ejecución de un archivo *bash* se da permisos de ejecución.

```
# cd /scripts

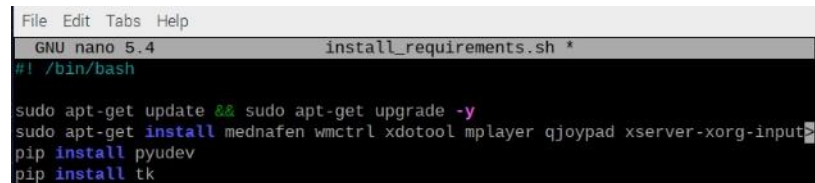
# chmod +x ./*

# ./install_requirements.sh
```

Como se observa en la figura 1, el archivo contiene las acciones de instalación que se llevaran a cabo, para instalar mednafen, el emulador, *wmctrl*, para manejar las ventanas, *xdotool*, para la entrada de teclas, *mplayer*, para el sonido de arranque, *qjoypad*, para utilizar el *gamepad* como controlador, *xserver-xorg-input*, para que la tarjeta reconozca específicamente el control de Xbox ONE, *pyudev*, para gestionar el USB y *tk* para la generación de ventanas.

Lo siguiente es hacer las configuraciones para tener una *splash screen* y ruido de arranque personalizado. Para el proyecto se creó una imagen completamente personalizada y se utilizó un sonido específico, los cuales también pueden ser descargados del repositorio, sin embargo, se puede descargar cualquier otra imagen y sonido que se desee utilizar como remplazo.

Se debe de ejecutar el comando `# sudo nano /boot/config.txt` para agregar la línea `disable_splash=1` al final del archivo. Ahora es necesario configurar otro archivo con `# sudo nano /boot/cmdline.txt` agregando las líneas que se muestran en la figura 3 (hasta *skip*), todo en una misma línea.



```
File Edit Tabs Help
GNU nano 5.4 install_requirements.sh *
#!/bin/bash

sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install mednafen wmctrl xdotool mplayer qjoypad xserver-xorg-input
pip install pyudev
pip install tk
```

Figura 1. Instalación.



```
File Edit Tabs Help
GNU nano 5.4 /boot/config.txt *
arm_64bit=1

# Disable compensation for displays with overscan
disable_overscan=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

[pi4]
# Run as fast as firmware / board allows
arm_boost=1

[all]
dtoverlay=w1-gpio
disable_splash=1
```

Figura 2. config.txt.

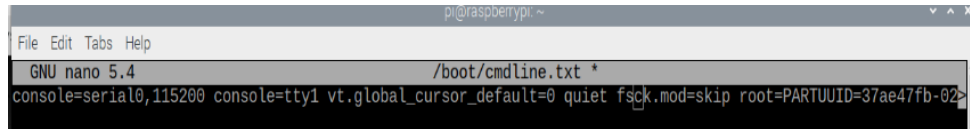


Figura 3. cmdline.txt.

Estas configuraciones son importantes para que no se desplieguen mensajes de consola a la hora de iniciar la tarjeta. Para remplazar la nueva imagen de *splash*, esta se debe de copiar de la carpeta *img*, a la carpeta que contenía la *splash* original. Es importante que el nombre de la imagen sea *splash.png* para que el sistema la logre reconocer. El comando que se debe de teclear es el siguiente:

```
# sudo cp /home/pi/proyecto/img/splash.png /usr/share/plymouth/themes/pix/splash.png
```

Después se debe de copiar el archivo de sonido a la carpeta correspondiente:

```
# sudo mkdir /etc/sound
```

```
# sudo cp /home/pi/proyecto/sonido/arranque.mp3 /etc/sound/arranque.mp3
```

Lo siguiente que se debe de hacer en cuestión de configuraciones es el montaje para el dispositivo extraíble, en este caso una USB. Esto se logra con un archivo de configuración del SO y con la creación de un servicio que llama a dos scripts en *bash*, los comandos son los que se muestran en la figura 4:

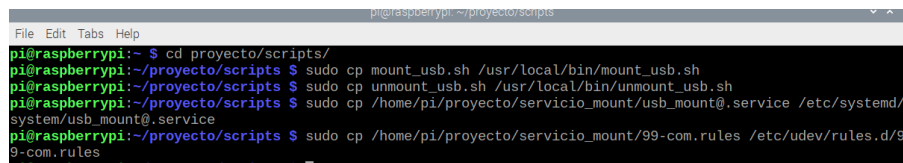


Figura 4. Comandos para montaje de USB.

Para tener los *roms* precargados estos se pueden descargar del mismo repositorio o descargar propios, los cuales se van a encontrar dentro de la carpeta proyecto/Roms.

Para seguir aprovechando la interfaz gráfica (escritorio), será necesario configurar el control de Xbox ONE con la aplicación de QJoyPad. Primero se debe confirmar que la tarjeta reconozca el control conectado (realizado en la parte de instalación), después con la aplicación abierta, se le puede dar un nombre al mando especificado para que, una vez hecha su configuración, esta sea la predeterminada en el futuro. Como se observa en la figura 5 se puede escoger que teclas se le asignan a los botones del control, en este caso se puede utilizar el arreglo que se desee, pero se recomienda que en la opción de la joystick izquierda del control se utilicen las opciones que se muestran en la ventana derecha de la figura 5. Una vez configurado todo se deben guardar los cambios y se puede cerrar la ventana.

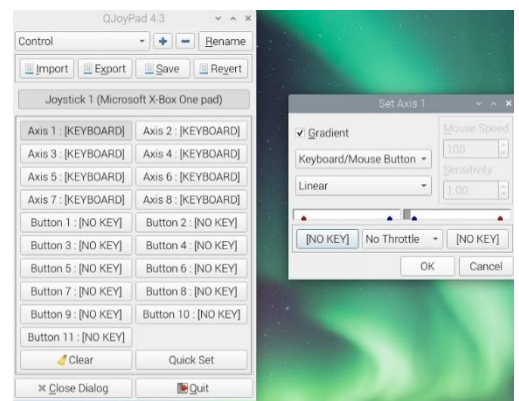


Figura 5. QJoyPad.

Galán Galván Arturo David

Número de lista: 10

Brigada: 6

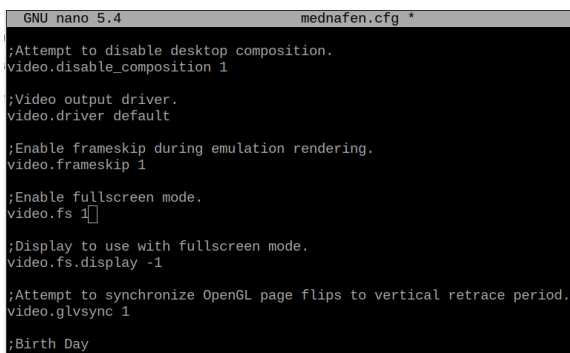
Sera necesario hacerle cambios a la aplicación de Mednafen también. La documentación indica que es necesario agregar un archivo de firmware al directorio de base del emulador, esto para que se puedan jugar los juegos de PSX (como agregar la BIOS de la consola).

```
# cd proyecto/firmware
# cp scp5501.bin ~/.mednafen/firmware/
```

La ubicación de 5501 indica que el firmware va a servir para juegos de Norte América (USA).

Es necesario hacer más configuraciones entonces hay que acceder al archivo de configuración de mednafen con el siguiente comando: `# cd ; cd .mednafen/ ; sudo nano mednafen.cfg`

Los cambios a realizar serán, habilitar el modo en pantalla completa para los juegos y cambiar el driver de sonido a SDL (de lo contrario no se oye el emulador). Estos cambios se muestran en las figuras 6, y 7.



```
GNU nano 5.4 mednafen.cfg *
;Attempt to disable desktop composition.
video.disable_composition 1

;Video output driver.
video.driver default

;Enable frameskip during emulation rendering.
video.frameskip 1

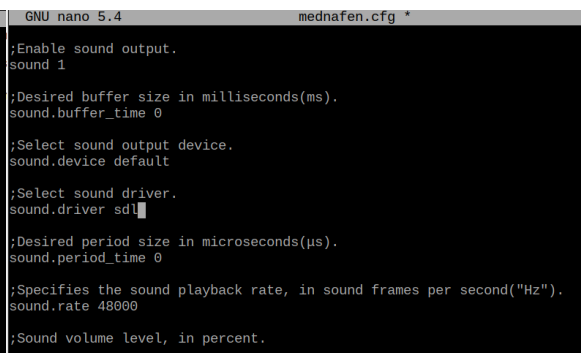
;Enable fullscreen mode.
video.fs 1

;Display to use with fullscreen mode.
video.fs.display -1

;Attempt to synchronize OpenGL page flips to vertical retrace period.
video.glvsync 1

;Birth Day
```

Figura 6. Habilitar pantalla completa en juegos.



```
GNU nano 5.4 mednafen.cfg *
;Enable sound output.
sound 1

;Desired buffer size in milliseconds(ms).
sound.buffer_time 0

;Select sound output device.
sound.device default

;Select sound driver.
sound.driver sdl

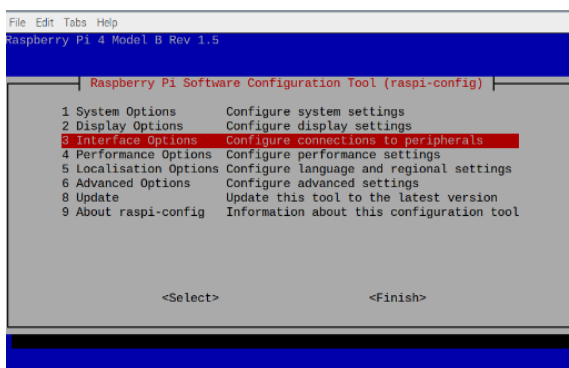
;Desired period size in microseconds(us).
sound.period_time 0

;Specifies the sound playback rate, in sound frames per second("Hz").
sound.rate 48000

;Sound volume level, in percent.
```

Figura 7. Cambio de driver de sonido.

Por último, antes de pasar a la parte de la integración se va a configurar el VNC. Esto es algo simple pues ya viene integrado en la Raspberry pi, solamente es cuestión de entrar a la configuración de la tarjeta con `# sudo raspi-config` y seleccionar las opciones que se muestran desde la figura 8 hasta la figura 15.

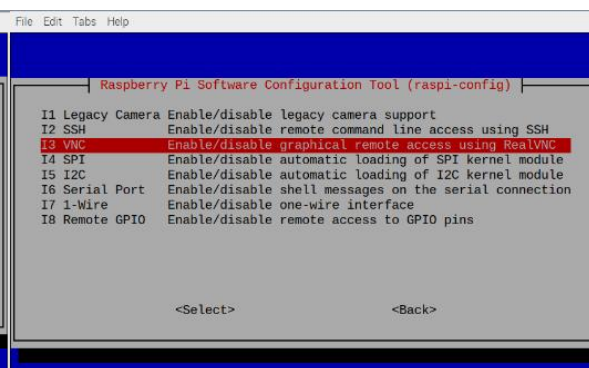


```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options      Configure system settings
2 Display Options     Configure display settings
3 Interface Options   Configure connections to peripherals
4 Performance Options Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options   Configure advanced settings
8 Update              Update this tool to the latest version
9 About raspi-config  Information about this configuration tool

<Select>              <Finish>
```

Figura 8. Menú de configuración.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH            Enable/disable remote command line access using SSH
I3 VNC            Enable/disable graphical remote access using RealVNC
I4 SPI            Enable/disable automatic loading of SPI kernel module
I5 I2C            Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins

<Select>              <Back>
```

Figura 9. Selección de VNC.

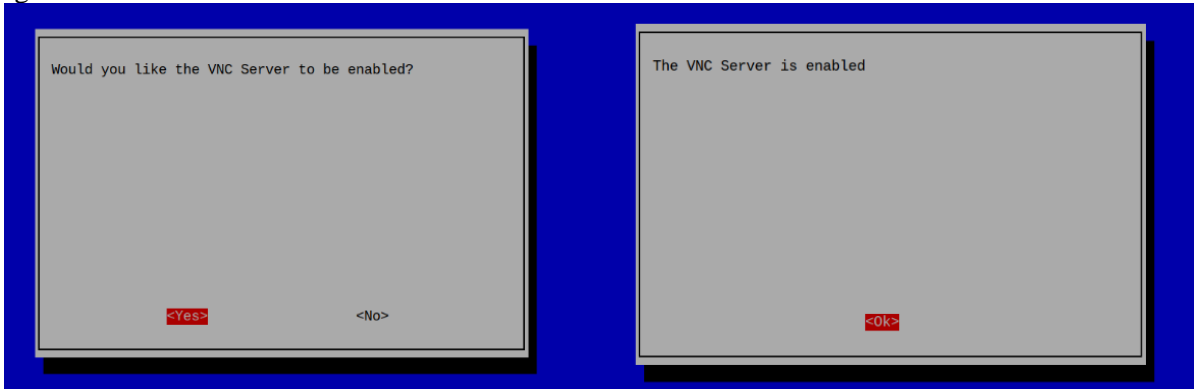


Figura 10. Habilitación de VNC.

Figura 11. VNC habilitado.

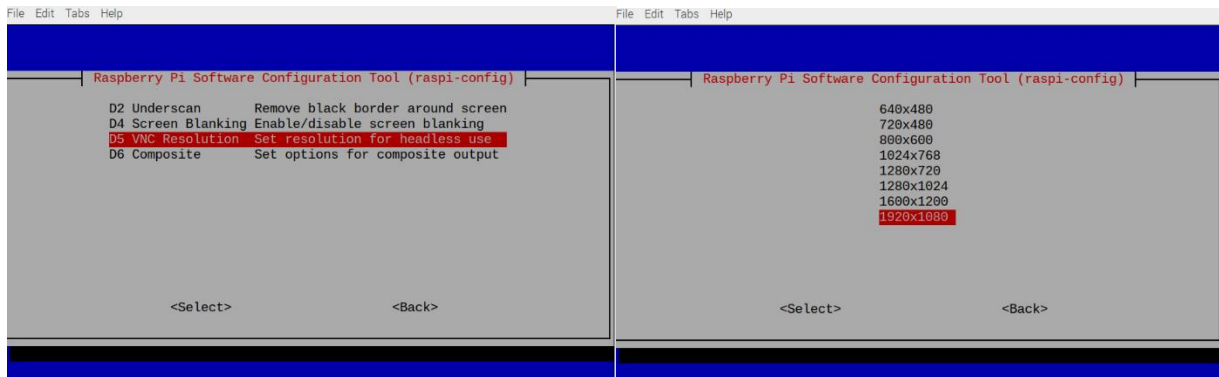


Figura 12. Resolución.

Figura 13. Resolución seleccionada.

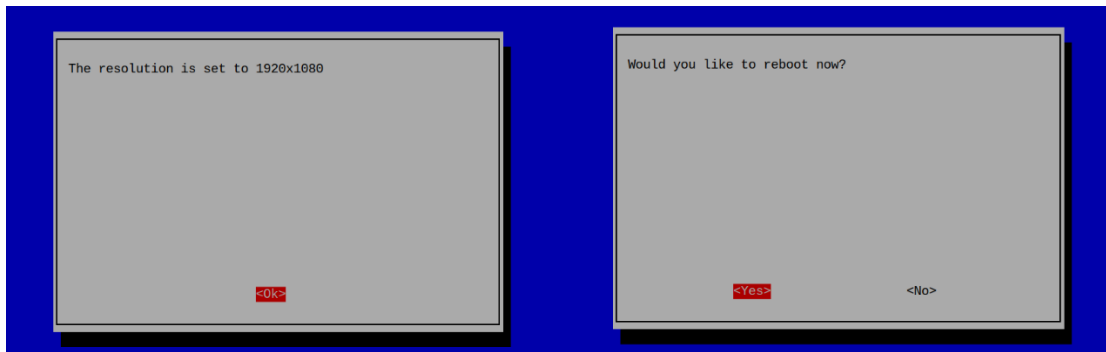


Figura 14. Resolución definida.

Figura 15. Reinicio para que se apliquen los cambios.

Ya que esta todo configurado, descargado, con permisos y preparado es momento de deshacernos del escritorio y juntar los módulos de software. Primero se debe teclear la siguiente línea:

```
# sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

El archivo que abre la línea anterior es de suma importancia ya que ahí se controla todo lo que se va a iniciar cuando se arranque la tarjeta. Las primeras líneas comentadas que se muestran en la figura 16 sirven para deshabilitar todo el entorno gráfico de Raspbian que no necesitamos, la cuarta línea es para que arranque el sonido personalizado que ya se había agregado, la quinta y séptima línea son scripts de Python (que se explicaran a detalle más adelante) para el montaje del USB y para abrir mednafen en pantalla completa, respectivamente, por último, la sexta línea abrirá la aplicación de QJoyPad desde arranque para que todo se pueda controlar con el gamepad.

Galán Galván Arturo David

Número de lista: 10

Brigada: 6

Con todo lo anterior realizado, una vez que se vuelva a reiniciar o apagar y prender la Raspberry Pi, el resultado debería ser el mismo que en el siguiente video demostrando el correcto funcionamiento del proyecto:

Video demostrativo

<https://youtu.be/sxd1POM44xg>

Integración de los scripts

Para finalizar, se va a explicar más a detalle el funcionamiento de los códigos utilizados y mencionados en el tutorial. Para el servicio de USB se tendrá el archivo 99-com.rules al cual se le agregan las siguientes líneas:

```
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/bin/systemctl start usb_mount@%k.service"
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="remove", RUN+="/bin/systemctl stop usb_mount@%k.service"
```

Estas líneas son las que nos van a ayudar a detectar cuando un USB haya sido insertado o removido de la tarjeta. Con *systemctl* se llama al servicio y lo importante es ver la ruta del dispositivo; para lograr esto se utilizan expresiones regulares que se pasan al servicio usando *usb_mount@%k*.

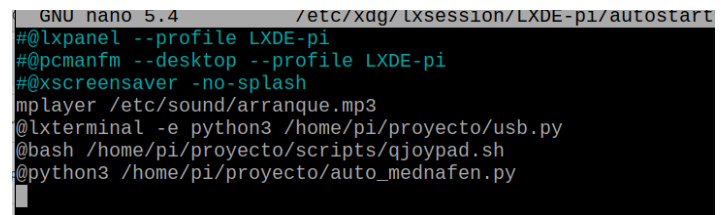
```
[Unit]
Description=USB MOUNT on %i

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/usr/local/bin/mount_usb.sh add %i
ExecStop=/usr/local/bin/unmount_usb.sh remove %i
Restart=on-failure
```

Figura 17. Servicio de montaje para USB.

El bash de la figura 18 ayuda a construir la ruta del dispositivo, la ruta de *pi6061-6154* nos sirve para el montaje de cualquier dispositivo USB.

Para la acción de detener el juego una vez se insertará la memoria y que se mostrara la lista de *roms* disponibles de la memoria se usan los siguientes códigos de las figuras 19 y 20 respectivamente.



```
GNU nano 5.4 /etc/xdg/LXsession/LXDE-pi/autostart
#@lxpanel --profile LXDE-pi
#@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splash
mplayer /etc/sound/arranque.mp3
@lxterminal -e python3 /home/pi/proyecto/usb.py
@bash /home/pi/proyecto/scripts/qjoypad.sh
@python3 /home/pi/proyecto/auto_mednafen.py
```

Figura 16. Autostart.

La figura 17 muestra dos scripts en bash que nos sirven para montar el USB de manera correcta, el porcentaje nos ayuda a completar la ruta de donde inicialmente esta la USB para posteriormente montarla.

```
#!/bin/bash
ACTION=$1
DEVBASE=$2
DEVICE="/dev/${DEVBASE}"
sudo mkdir /media/pi6061-6154
sudo mount ${DEVICE} /media/pi6061-6154
```

Figura 18. Bash para montaje USB.

```

----USB.py----

Subprocess.call(['bash', '/home/pi/proy
ecto/scripts/presionar_enter.sh',
contenido[0]])

----presionar_enter.sh----

#!/bin/bash

ID_VENTANA=$1

xdotool key -window ${ID_VENTANA}
Return

```

```

Shutil.copytree(source, "/home/pi/TMP_ROMS/")

Os.popen("cp -r /home/pi/TMP_ROMS/*
/home/pi/Proyecto/Roms")

```

Figura 20. Copiar Roms a la tarjeta.

Figura 19. Pausar juego.

En la figura 19 se llama a un subprocesso que viene siendo el ID del juego o de la ventana del juego que se este ejecutando, una vez obtenido esto con *xdotool* se simula la tecla de 'Enter', que en este caso sirve para pausar la ventana de los juegos en ejecución. En la figura 20 se utiliza la librería *shutil* que nos ayuda a crear una copia de los *roms* a un directorio temporal, luego de que esta lista se recupera, se usa un comando de línea para copiar todo al directorio interno del emulador.

```

rootwindow = tkinter.Tk()

rootwindow.title("Lista de ROMS USB")

rootwindow.geometry("800x500")

roms = Text(rootwindow, font=("Helvetica", 20))

for x in lst_usb_roms:

    roms.insert(END, x + '\n')

roms.pack()

rootwindow.eval('tk::PlaceWindow . center')

rootwindow.mainloop()

```

Figura 21. Despliegue de ventana con roms.

Para desplegar la ventana con *roms* se utiliza la librería *tkinter*. Se le da título a la ventana, fuente utilizada, tamaño de la ventana y se muestran los *roms* dentro de esta como se observa en la figura 21.

Para finalizar, y si nos referimos al script llamado *auto_mednafen.py*, lo que hace es ejecutar la interfaz del emulador en pantalla completa.

Con `os.popen("cd/usr/games;/ /mednaffe")` se ejecuta la aplicación mientras que con la línea `os.popen("wmctrl -r 'Mednaffe' -b toggle,fullscreen")` se abre en pantalla completa gracias a la librería *wmctrl*.

Repositorio de código fuente

<https://github.com/EduardoFM11/consola-de-videojuegos-retro>

Conclusiones

Para concluir este proyecto/tutorial cabe destacar que crear un sistema embebido definitivamente no es nada sencillo. Se lograron todos los objetivos, sin embargo, cada paso fue muy tardado de desarrollar a causa de que muchas librerías, módulos y documentación no servían para nuestras necesidades específicas o simplemente no eran compatibles entre ellos.

En el caso del control de Xbox, era necesario usar el comando específico de *xorg* para que la pi lo reconociera, pero encontrar ese comando fue más difícil de lo esperado; también mucha documentación recomendaba utilizar *crontab*, *.bashrc* o *init.d* para modificar el arranque, pero ninguno de estos métodos nos resultó útil hasta que encontramos *autostart*.

En cuestión de código hubo varios problemas a la hora de implementar *os.popen* ya que el script necesitaba de *sleeps()* para que no se siguieran los pasos de manera paralela. También ocurrieron errores inesperados en el momento de unir todas las partes; en el entorno gráfico el USB se montaba sin problema, pero en el momento de desactivarlo, este dejaba de hacerlo correctamente; una vez que todo estaba listo si iniciábamos la tarjeta con una sesión de VNC abierta desde un principio, por alguna razón los scripts dejaban de funcionar, entonces simplemente se iniciaba sesión después de que todo cargara bien.

A pesar de todo lo anterior se logro encontrar un balance para que la consola funcionara correctamente, por partes todo parece y resulta ser muy simple en algunos casos, sin embargo, cuando se requiere implementar todo junto para obtener una solución tan específica con materiales y necesidades tan detalladas me di cuenta que es ahí en donde entra la dificultad y la necesidad de verdaderamente comprender lo que se esta haciendo y que herramientas utilizar para poder lograrlo.

La solución al proyecto fue muy específica respecto a los componentes utilizados, software y librerías. Para mejorarlo se podría tratar de realizar de manera más amplia y encontrar la manera en que cada paso a seguir sirva para distintos modelos de Raspberry pi o inclusive distintos emuladores (como lo podría ser Snes9x).

Referencias

Pedre, S. (2017). Sistemas embebidos. Laboratorio de Robótica y Sistemas Embebidos, Departamento de computación FCEN UBA.

Raspberry Pi. (2020). *Raspberry Pi 4 Tech Specs*. <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/?variant=raspberry-pi-4-model-b-8gb>

Mednafen. (2005). *Introducción a Mednafen*. <https://mednafen.github.io/>

Mednafen. (2022). *Documentación General*. https://mednafen.github.io/documentation/#Section_base_directory

Wiesner, S. (2011). *pyudev – pure Python libudev binding*. <https://pyudev.readthedocs.io/en/latest/>

Ramachandran, S. (2021). *wmctrl*. <https://github.com/saravanabalagi/wmctrl>

Python. (2023). *tkinter — Interface de Python para Tcl/Tk*. <https://docs.python.org/es/3/library/tkinter.html>

Sissel, J. (2022). *xdotool - x11 automation tool*. <https://github.com/jordansissel/xdotool>

Galán Galván Arturo David

Número de lista: 10

Brigada: 6

MPlayer team. (2022). *MPlayer - El reproductor de Películas para LINUX*.
<http://www.mplayerhq.hu/DOCS/HTML/es/MPlayer.html#intro>

Panzenböck, M. (2022). *QJoyPad 4*. <https://github.com/panzi/qjoypad>

Gurtubay, U. G. (2010). *Control remoto multiplataforma basado en software libre: automatización de sesión VNC integrando gestión de incidencias e inventario*. RedIRIS: boletín de la Red Nacional de I+D RedIRIS, (88), 193-197.