Extra Practice:

Emulation of retro video games with the Raspberry Pi

Fundamentals of Embedded Systems

Author:
Rodrigo Iván Olvera Martínez

1. Aim

The student will learn to make the necessary configurations to emulate retro video games on the Raspberry PI.

2. Introduction

This practice summarizes the steps to follow to display video games on the screen, using the Mednafen emulator and by making some configurations we can use a control from either Playstation or Xbox to be able to play without the need for a keyboard.

2.1. Mednafen

Mednafen is an emulator of multiple retro video game systems. Its name is an abbreviation of "My Emulator Doesn't Need A Frickin' Excellent Name." It is a free and open source software that allows the user to play and enjoy retro console games.

Mednafen supports a wide range of systems, including NES, SNES, Game Boy, Game Boy Advance, Sega Genesis, Sega Saturn, Atari Lynx and many other popular platforms. The emulator focuses on precision and hardware emulation, meaning that it attempts to faithfully replicate the operation of the original consoles and their games.

Using Mednafen, the user can load ROM or ISO images of games into the emulator and play them on their computer. The software offers various configuration options such as graphics, sound and controls, allowing players to customize their gaming experience according to their preferences.

3. Material

It is assumed that the student has aa Raspberry Pi with Raspbian operating system and Python interpreter installed. Using git as a version control program is highly recommended.

- PS4 or Xbox controller
- USB v8 cable or the one necessary if the control is wireless.
- HDMI to HDMI or HDMI to VGA cable if your monitor needs it
- Monitor and power cable
- Keyboard and Mouse with USB input

4. Instructions

1. Install Mednafen.
2. Setting.
3. Program.

4.1. Step 1: Install Mednafen.

At this point the student has already properly connected the Raspberry to the Internet network, then in the Raspberry Pi OS console insert the command “sudo apt update”

```
axel@raspberrypi:/home$ sudo apt update
Obj:1 http://security.debian.org/debian-security bullseye-security InRelease
Obj:2 http://deb.debian.org/debian bullseye InRelease
Obj:3 http://deb.debian.org/debian bullseye-updates InRelease
Obj:4 http://archive.raspberrypi.org/debian bullseye InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
axel@raspberrypi:/home$
```

Figure 1: update command

Then insert the command “sudo apt install mednafen”

```
axel@raspberrypi:/home$ sudo apt install mednafen
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
mednafen ya está en su versión más reciente (1.26.1+dfsg-1).
El paquete indicado a continuación se instaló de forma automática y ya no es nec
esario.
  libfuse2
Utilice «sudo apt autoremove» para eliminarlo.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
axel@raspberrypi:/home$
```

Figure 2: command to install mednafen

Give yes to the installation and finally restart the raspberry with “reboot”.

4.2. Step 2: File Configuration.

We are going to create a service for the Raspberry to start with a custom image or video. We create a file in the root directory with the command “sudo nano play_boot.sh”.
And the following is written:

```
#!/bin/bash
fbi -T 1 -noverbose -a /home/Home/splash.png
```

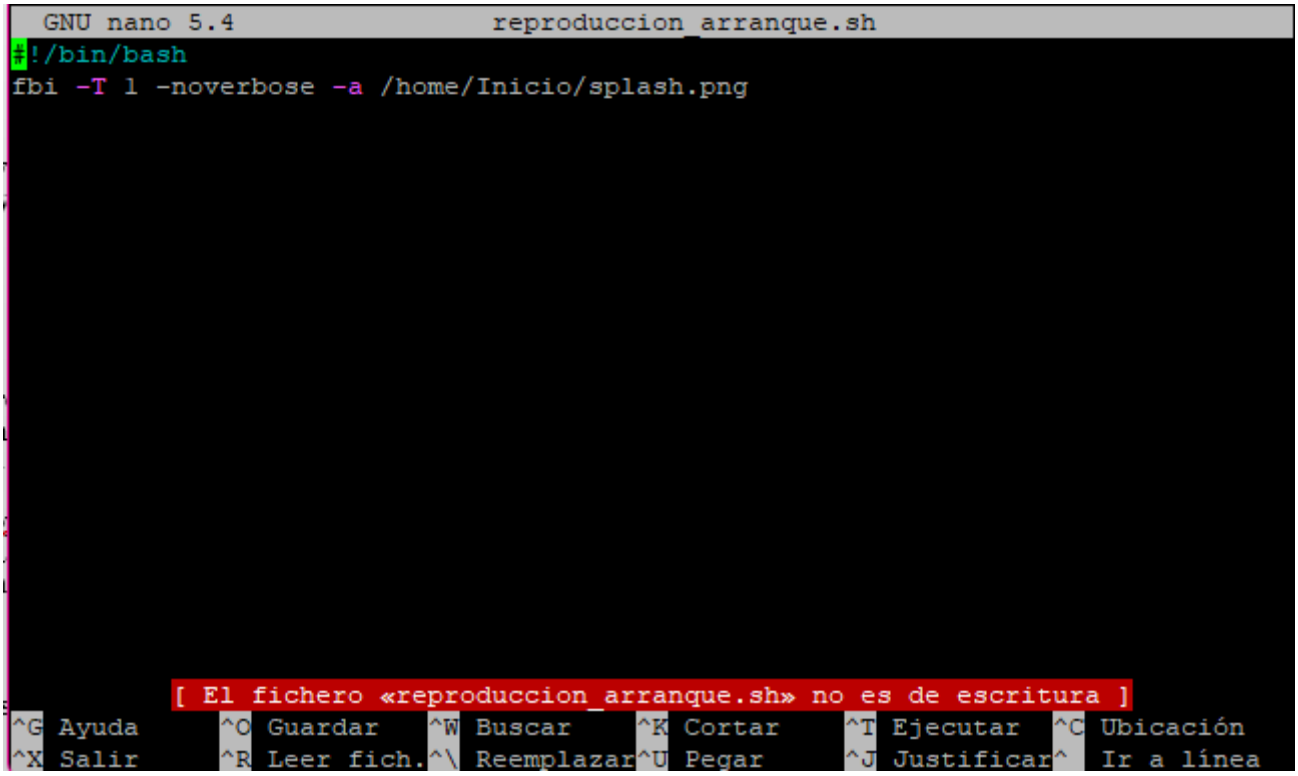


Figure 3: Creating the file for custom image execution

We save the file. Pay attention to the address or path of the file, to later save it.

Then enable and activate the startup service

“sudo systemctl enable start_play” “sudo systemctl start start_play”

Restart the Raspberry, so that the changes can be seen, when restarting the customized boot image should be displayed on the screen.

Figure 4: Presentation image of our embedded system

4.3. Step 3: Copy the ROMs to a path.

Search for ROMs on the web and copy them to the preferred path, in this case "/home/rms/", it may be the one that be of convenience.

```
axel@raspberrypi:/home/rms$ ls
 CAPAME.nes
'Chip '\''n Dale - Rescue Rangers 2 (USA).nes'
'Donkey Kong Classics (USA, Europe).nes'
'DuckTales (USA).nes'
'Flintstones 2 - The Surprise at Dinosaur Peak!, The (U).nes'
'Legend of Zelda, The (U) (PRG1) [!].nes'
'Metroid (U).nes'
'Ninja Gaiden (USA).nes'
 SF3.nes
'Shadow Warriors 2 (E) [!].nes'
'Simpsons, The - Bartman Meets Radioactive Man (USA).nes'
 SMB3.nes
 SMW.nes
 splash.png
'Super Mario Advance 2 - Super Mario World (USA, Australia).gba'
'Super Mario Kart (USA).sfc'
'Teenage Mutant Ninja Turtles (USA).nes'
'Terminator, The (USA, Europe).nes'
'Tiny Toon Adventures (USA).nes'
'Zelda 2 - The Adventure of Link (U).nes'
axel@raspberrypi:/home/rms$
```

Figure 5: ROMs copied to the specific path

4.4. Step 4: Program

The proposed program is saved in the same route or in the convenient one. We execute the command "sudo nano run.py"

```
import pygame
import sys
import os
import subprocess
import inputs
import keyboard


# Inicializar Pygame
pygame.init()

# Configuración de la pantalla
screen = pygame.display.set_mode((1800, 1000))
pygame.display.set_caption('Menú con mando de PlayStation')

#Cargar sonido
button_sound = pygame.mixer.Sound("/home/sonido2.wav")
button_sound2 = pygame.mixer.Sound('/home/sonido1.wav')


# Colores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)

# Fuente y texto
font = pygame.font.Font(None, 36)
options = ["1. Super Mario Bros 3", "2. Super Mario World", "3. Captan America", "4. Chip 'n Dale - Rescue Rangers 2",
           "5. Donkey Kong Classics", "6. DuckTales", "7. Flintstones 2", "8. Ninja Gaiden",
           "9. Shadow Warriors 2", "10. Simpsons, The - Bartman Meets Radioactive Man", "11. Teenage Mutant Ninja Turtles", "12. Terminator",
           "13. Tiny Toon Adventures", "14. Legend of Zelda", "15. Metroid", "APAGAR","REINICIAR"]
selected_option = 0
change_speed = 1
delay = 160
BUTTON_TO_EXIT = 6

def close_mednafen():
    subprocess.run(['pkill', 'mednafen'])

def play_rom(rom_path):
```

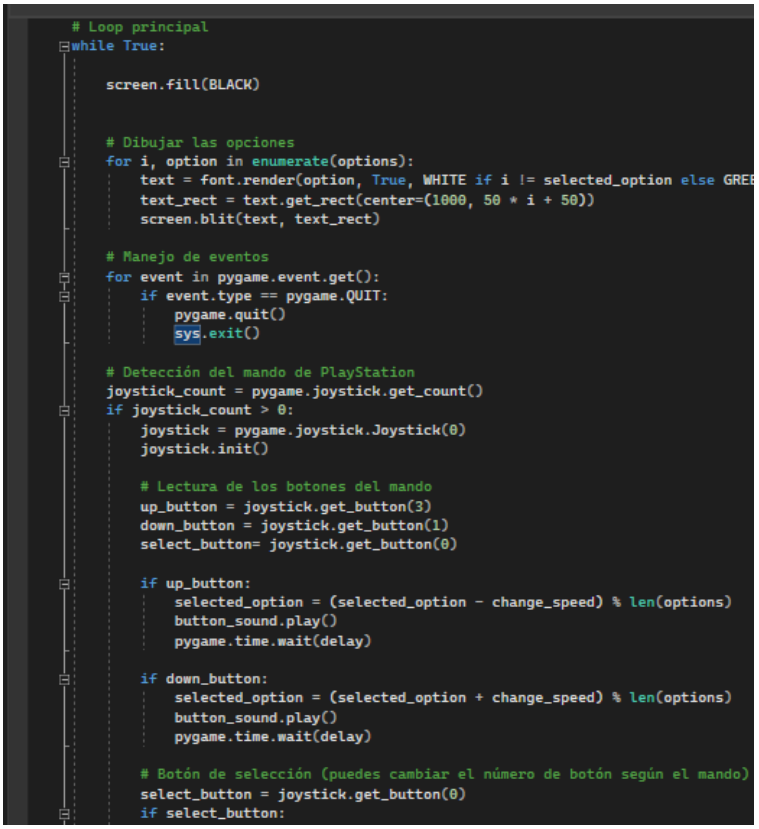
Figure 6.0: Menu

Initialize the libraries that would help us create our menu, "pygame" was used to control our instructions from the X-box one console controller and identify the controller from Python, "subprocess" was mainly useful to initialize and open our files .NES with the help of the emulator

```
BUTTON_TO_EXIT = 6

def close_mednafen():
    subprocess.run(['pkill', 'mednafen'])

def play_rom(rom_path):
    print("Iniciando Mednafen...")
    subprocess.call(["mednafen", rom_path])  # Ejecutar Mednafen con la ruta de la ROM

# Código correspondiente al botón que deseas asignar como F12

BOTON_A_ASIGNAR = 'BTN_SELECT'

def verificar_botones():
    events = inputs.get_gamepad()
    for event in events:
        if event.ev_type == 'Key' and event.code == BOTON_A_ASIGNAR:
            if event.state == 1:  # Verifica si el botón está siendo presionado
                #keyboard.press_and_release('F12')
                close_mednafen()

# Loop principal
while True:

    screen.fill(BLACK)


    # Dibujar las opciones
    for i, option in enumerate(options):
        text = font.render(option, True, WHITE if i != selected_option else GREEN)
        text_rect = text.get_rect(center=(1000, 50 * i + 50))
        screen.blit(text, text_rect)

    # Manejo de eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    # Detección del mando de PlayStation
    joystick_count = pygame.joystick.get_count()
    if joystick_count > 0:
```

Figure 6.1: Menu

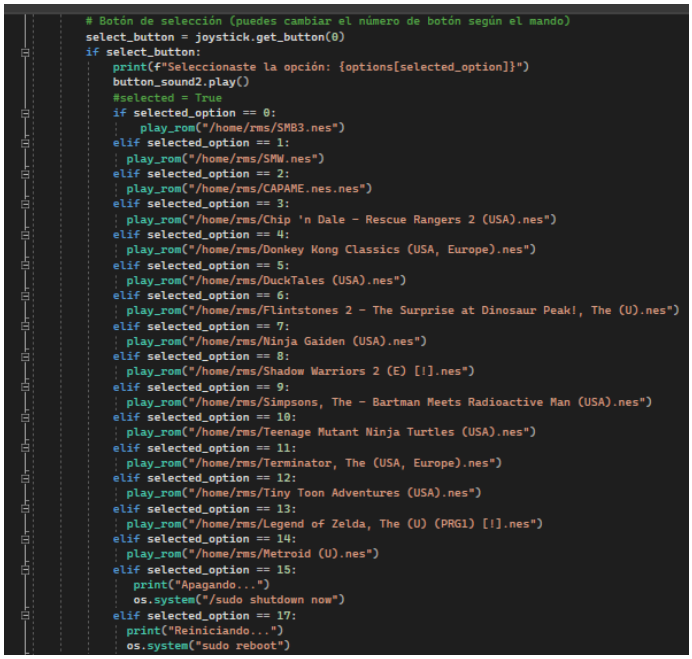
The close_mednafen and play_rom functions were created to open and close the “mednafen” emulator process.



```
# Loop principal
while True:

    screen.fill(BLACK)


    # Dibujar las opciones
    for i, option in enumerate(options):
        text = font.render(option, True, WHITE if i != selected_option else GREE
        text_rect = text.get_rect(center=(1000, 50 * i + 50))
        screen.blit(text, text_rect)

    # Manejo de eventos
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    # Detección del mando de PlayStation
    joystick_count = pygame.joystick.get_count()
    if joystick_count > 0:
        joystick = pygame.joystick.Joystick(0)
        joystick.init()

        # Lectura de los botones del mando
        up_button = joystick.get_button(3)
        down_button = joystick.get_button(1)
        select_button= joystick.get_button(0)

        if up_button:
            selected_option = (selected_option - change_speed) % len(options)
            button_sound.play()
            pygame.time.wait(delay)

        if down_button:
            selected_option = (selected_option + change_speed) % len(options)
            button_sound.play()
            pygame.time.wait(delay)

        # Botón de selección (puedes cambiar el número de botón según el mando)
        select_button = joystick.get_button(0)
        if select_button:
```

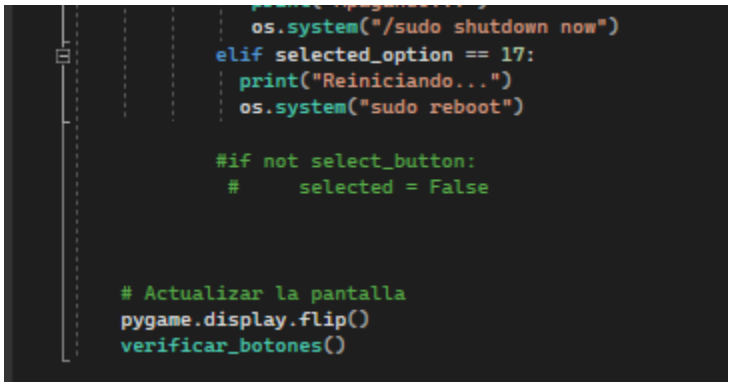
Figure 6.2: Menu

A main loop was created where the buttons that we would select to manage the menu and the scrolling of the games list were initialized, a delay had to be added since it moved too fast



```
        # Botón de selección (puedes cambiar el número de botón según el mando)
        select_button = joystick.get_button(0)
        if select_button:
            print(f"Seleccionaste la opción: {options[selected_option]}")
            button_sound2.play()
            #selected = True
            if selected_option == 0:
                play_rom("/home/rms/SMB3.nes")
            elif selected_option == 1:
                play_rom("/home/rms/SMW.nes")
            elif selected_option == 2:
                play_rom("/home/rms/CAPAME.nes.nes")
            elif selected_option == 3:
                play_rom("/home/rms/Chip 'n Dale - Rescue Rangers 2 (USA).nes")
            elif selected_option == 4:
                play_rom("/home/rms/Donkey Kong Classics (USA, Europe).nes")
            elif selected_option == 5:
                play_rom("/home/rms/DuckTales (USA).nes")
            elif selected_option == 6:
                play_rom("/home/rms/Flintstones 2 - The Surprise at Dinosaur Peak!, The (U).nes")
            elif selected_option == 7:
                play_rom("/home/rms/Ninja Gaiden (USA).nes")
            elif selected_option == 8:
                play_rom("/home/rms/Shadow Warriors 2 (E) [!].nes")
            elif selected_option == 9:
                play_rom("/home/rms/Simpsons, The - Bartman Meets Radioactive Man (USA).nes")
            elif selected_option == 10:
                play_rom("/home/rms/Teenage Mutant Ninja Turtles (USA).nes")
            elif selected_option == 11:
                play_rom("/home/rms/Terminator, The (USA, Europe).nes")
            elif selected_option == 12:
                play_rom("/home/rms/Tiny Toon Adventures (USA).nes")
            elif selected_option == 13:
                play_rom("/home/rms/Legend of Zelda, The (U) (PRG1) [!].nes")
            elif selected_option == 14:
                play_rom("/home/rms/Metroid (U).nes")
            elif selected_option == 15:
                print("Apagando...")
                os.system("/sudo shutdown now")
            elif selected_option == 17:
                print("Reiniciando...")
                os.system("sudo reboot")
```

Figure 6.3: Menu



```
                os.system("/sudo shutdown now")
            elif selected_option == 17:
                print("Reiniciando...")
                os.system("sudo reboot")

            #if not select_button:
             #    selected = False



    # Actualizar la pantalla
    pygame.display.flip()
    verificar_botones()
```

Figure 6.4: Menu

In this part, when selecting an option we call the “play_rom” function along with the address of our .NES files and finally the projection of our graphic print.

This menu is responsible for loading the roms and with system calls we start mednafen automatically, verify very well if the paths and names of the roms are correct, otherwise the game will not run. And all this controlled from the X-box controller

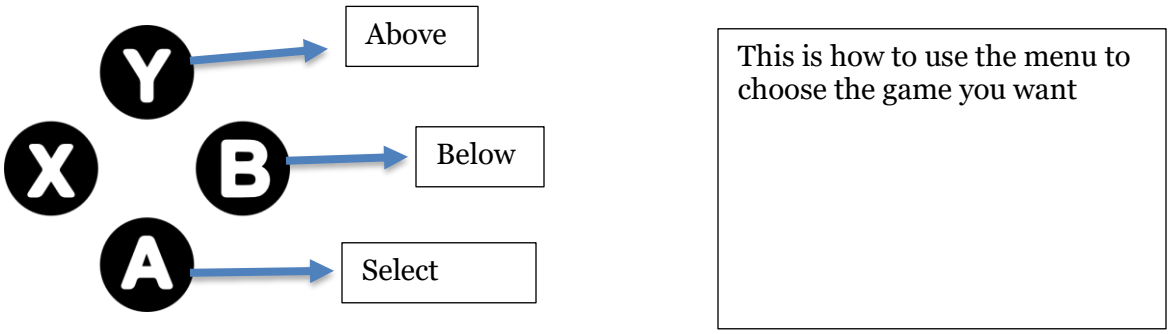


Figure 7: Menu control diagram

4.5. Step 4: Boot File Configuration

As we can see, before the program (run.py) is executed, which is our menu, we give it a sleep of 30 seconds since previously the program could not be executed at that time since it needed resources to be loaded that the program would later use. program was going to need this, it was created in the path /etc/systemd/system/programa.service

```
[Unit]
 Description=programa

[Service]
 ExecStartPre=/bin/sleep 30
 ExecStart=/usr/bin/python3 /etc/init.d/run.py
 Restart=always
 User=root

[Install]
 WantedBy=multi-user.target
```

Figure 11: Boot file.5

After this we reload systemd with sudo systemctl daemon-reload, we enable the service sudo systemctlenablemy_program.service and finally we can start the service manually to test it.

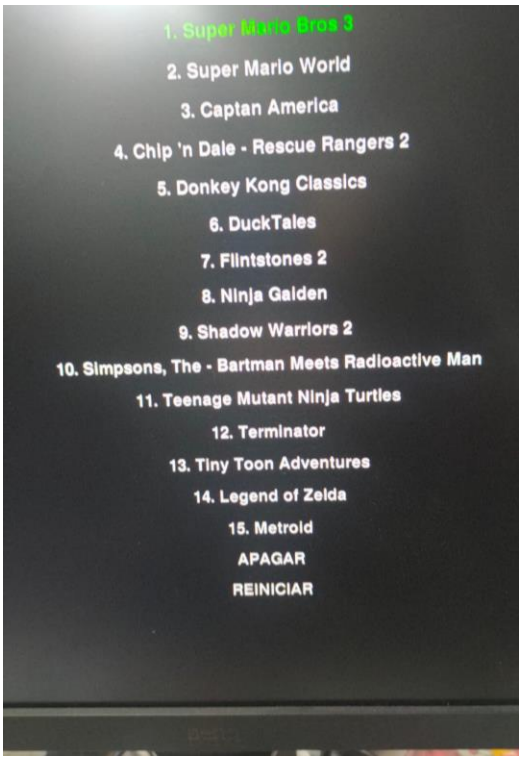


Figure 8: Result of the menu graphical interface.

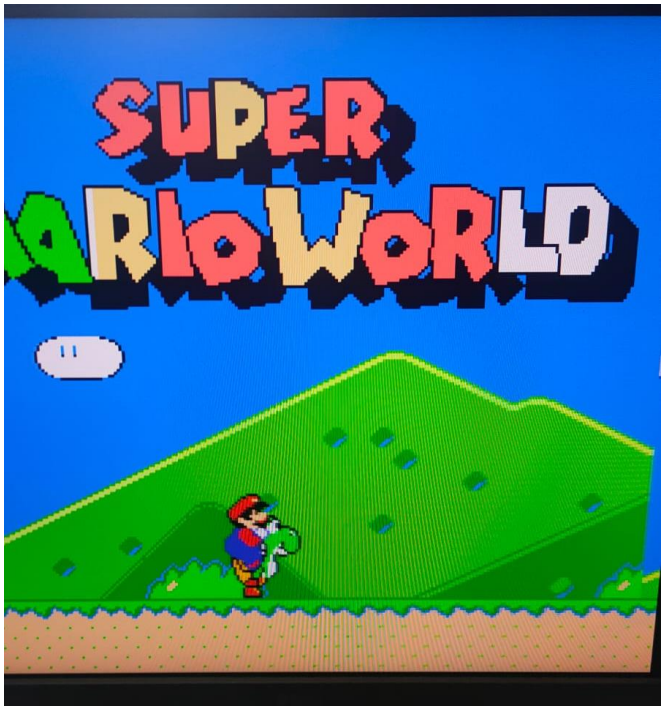


Figure 9: Visualization of the video games.

Figure 10: Link to the video uploaded to YouTube

https://youtu.be/6l1N-wfPnZo

4.6. Step 5: Command or control configuration.

When starting any game we press Ctrl+Shift+1 and follow the instructions on the screen, you must press the buttons shown to configure them and save them. This step is only done the first time, later it will no longer be necessary.

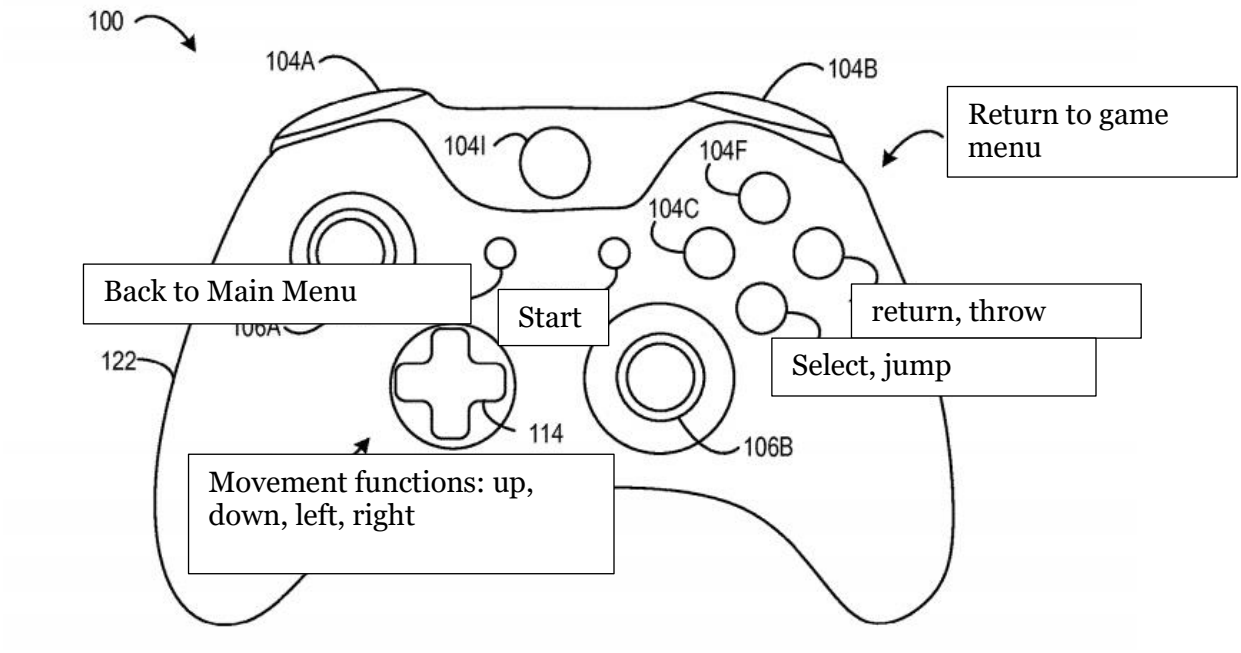


Figure 11: Controller configuration within the emulator

1. Experiments

[5pt] Run a game and configure the buttons for the controller you have chosen

[2pt] Modify the program so that when closing a game all available games are presented again.[3pt] Add at least 5 more

ROMs, show them in the interface and run the games.

[+3pt] Make the configurations so that when you turn on the raspberry the program runs without having to start itmanually.

2. Questionnaire

What other emulation tools exist, for the purpose of this practice?

Can we load playstation or gamecube games with mednafen?

Is it considered illegal to reproduce copies of Nintendo games in the manner in which they are made in this practice?

3. References

[1] insertmorecoins forum - How do you configure the command in Mednafen? (sf). https://insertmorecoins.es/foro/consolas/como-se-configura-el-mando-en-mednafen/Ludmila Maceková. 1- wire-the technology for sensor networks. Acta Electrotechnica et Informatica, 12(4):52,2012. Last accessed 06/14/2023

[2] *Mednafen General Documentation*. (sf). https://mednafen.github.io/documentation/. Last Accessed 06/11/2023

[3] Emulators - Raspberry Pi Gaming - page 62. (sf). http://what-when-how.com/Tutorial/topic-213lnr/Raspberry-Pi-Gaming-90.html Last Accessed 06/11/2023

[4] Ellingwood, J. (2020). How to use Systemctl to manage Systemd services and units. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services- and-units-es.Last Accessed 14/11/2023

4.Links.

5.

https://youtu.be/6l1N-wfPnZo

https://github.com/Rodrigoivan09/FSEm-2024-1.git