Contents

Media Center: Overview and tutorial

May 26Th, 2022

1 Introduction

A multimedia center is an adapted computer intended to play music, watch movies, and photo albums that are stored on a local device or that come from an external one plugged into the system. With this project we intended to create as software capable of doing all these tasks using what we learned during the course and applying a few more concepts.

The idea was to show a UI that can handle different approaches, for example, connecting the users to a streaming service site or allowing them to insert an external device and be able to watch their personal media files, such as music, videos, or photos.

With the following tutorial you will be able to generate and understand the core mechanics of a media center application.

2 Objective

As stated before, the main objective of this tutorial will be to guide you through the process of implementing a media center, complete with a user interface and some basic functions that will give you a better scope on how to handle certain events and use cases that are present in the most basic of media play software.

We will be using python as our main development language and a variety of libraries that will make our life easier when implementing everything that our media center software will be capable of.

Also keep in mind that there will be some indications for you to apply or not given the case that you want the code to be used on a `Ubuntu Linux OS` or if you want the program to be able to work while running it on a `RaspberryPi` environment.

Without further ado, let's talk about **requirements**.

3 Requirements

Before we talk more about the project and it's many details, you will need to check if you have all the basic requirements for the following implementation, keep in mind that all these are the basic requirements for running the program on a Ubuntu environment, if you're looking to make a physical implementation I suggest you check the following tutorial: Run python script on RaspberryPi[1]

Now, let us move on. Here are the basic requirements list for the tutorial:

- **A computer with Ubuntu, we recommend version 20.04**
- **A stable Internet connection**
- **The** `MultimediaCenter.py| file provided to you by on this repository: Repository`
- **Git**
- **A text editor, we recommend Sublime3[2]**
- **[Optional] A user with admin/root privileges**
- **[Optional][Physical] A RaspberryPi card**

If you have met all these requirements then you are all setup to continue with the rest of the tutorial!

4 Safety measures

Please, for your safety, take your time to read and to complete the tutorial, it is recommended to work for **intervals of 1 hour** and taking some rest from the screen of the computer to protect your eyes.

To prevent body aches, especially back problems, it is recommended to get up and walk around the house or work space for a few minutes before returning to work, this way you can prevent the most common health problems when working in front of the computer and during long periods of time in a sit down position. For more information about this and more concerns, please check this site[4] to get a better understanding of this matter.

5 Tutorial

5.1 Setting up the workspace

5.1.1 Step 1

If you have already cloned the repository or downloaded it's contents then I suggest you place them in an easy to find folder and go to the next step, if have not, then please run the following commands:

- `cd /[The folder you want the contents to be saved]/`
- `git clone https://github.com/Darkahnott/ears_pjff_FinalProject`
- Once the files are copied: `cd src/`

5.1.2 Step 2

Once you have the repository's contents on your computer, open a terminal on the `src` folder

Then enter the following commands, these will get you the recommended software to work on the project, you will be downloading Firefox, VLC player, and a few python libraries that are necessary for the project.

- `sudo apt-get update //If you're using ubuntu this line gets you the latest Firefox vers`
- `sudo apt-get install firefox //Just in case you don't have it`
- `sudo apt-get install vlc`
- `pip3 install python-vlc`
- `sudo apt-get install libx11-dev`
- `sudo apt-get install xauth dbus-x11`
- `sudo apt-get install python`
- `sudo apt-get install python3-tk`
- `sudo apt-get install python3-pygame`

5.1.3 Step 3

Before you run the code, you have to keep in mind that by default the code its suited to work on a Ubuntu or Linux environment, to be able to use it on the RaspberryPi you will have to change a few lines.

Using Sublime or any text editor of your liking find all the lines that contain the next message:

```
#For use on a raspberryPI uncomment the following line
```

And uncomment the immediate next line, the one that starts with **path** by removing the # symbol.

After that you will have to comment by adding the # symbol on the line that appears after the following message:

```
#For use on a linux filesystem uncomment the following line
```

5.1.4 Step 4

Once you ran all the previous commands and installed everything needed, you will run the following command:

```
python3 MultimediaCenter.py
```

You will see a screen like this one:

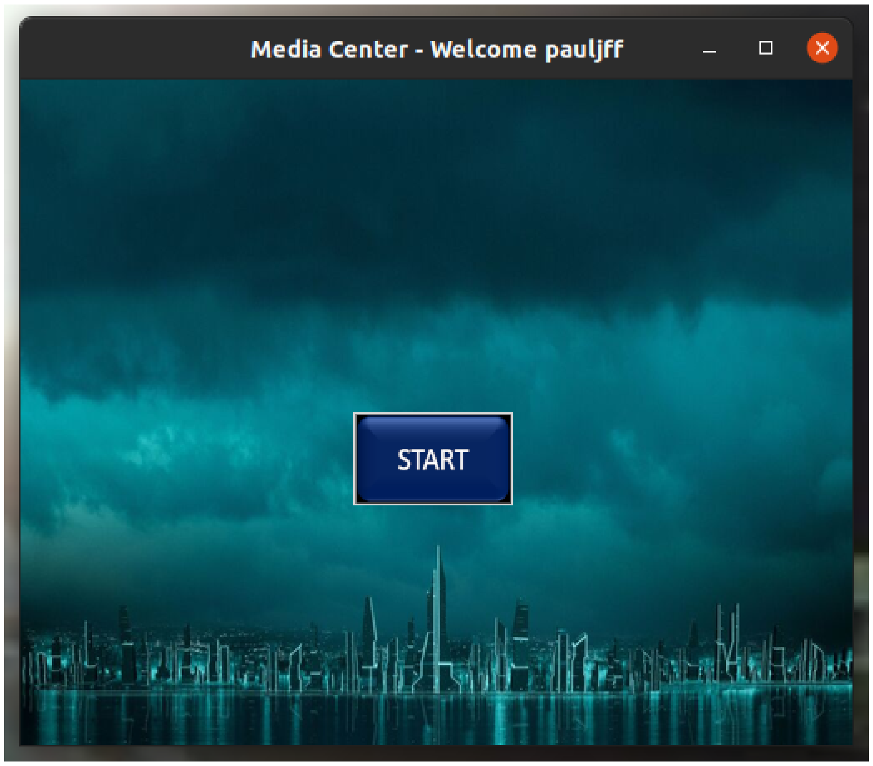

Figure 1: *Start screen*

5.2 Use and coding

As you could see on the previous figure, the program starts a windows of certain dimensions with a button on the middle of it, if you click on the button you are going to access the main menu screen.

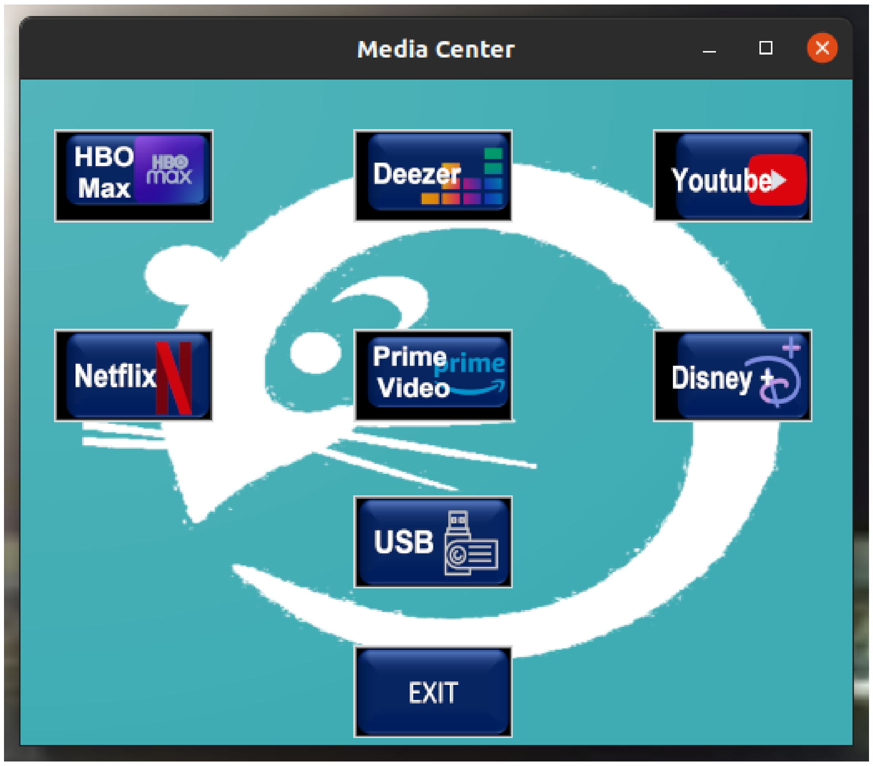

Figure 2: *Main menu screen*

Let us take a look into this section's code. The next extract of the code (which you can check for yourself using sublime to open the `MultimediaCenter.py` file) contains the basic structure for this window.

```
        ...

    def MainMenu(root):
            root.destroy()
            #Creating and styling window
            wind1 = Tk()
            wind1.title("Media␣Center")
            wind1.geometry('500x400')

            bg=(PhotoImage(file = "Resources/img/back/logo.png"))
            #Setting up the label
            Start=Label(wind1,text="Media␣Center",image=bg)
            Start.pack()

        #Creating the button and styling it,
        #the calling the function of the service: HBO
            HBOIMG=PhotoImage(file='Resources/img/buttons/HBO.png')
            HBOButton=Button(wind1,image=HBOIMG,command=HBO,width=90,
            height=50,bg='black')
            HBOButton.place(x=20,y=30)

            ...
```

As you can see the program deletes the previous window from the user perspective and then creates a new one with the information that we want the user to see, then we see how we create the style on the "HBO" button and give it style and position on the current window. We also define how the code will behave when clicked, as you can see it calls a function call HBO, that you can check here:

```
...

#Function to open the desired service using
#the default web browser
def HBO():
    webbrowser.open("https://play.hbomax.com/login",
    new=2, autoraise=True)

    ...
```

As we can see, the HBO function opens a new tab/window of the default browser and directs you to the login page of the streaming site.

Let's take a look into the USB selection screen. This one has a lot of work behind it, it lets you insert a USB and then check its contents divided by 3 main categories, Music, Videos & Photos.

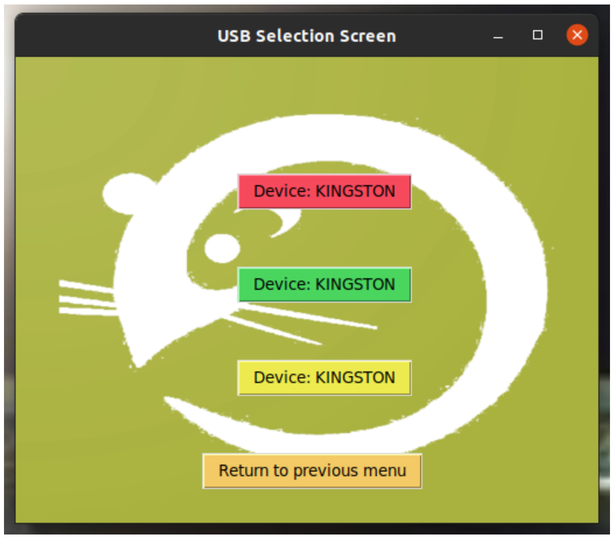

Figure 3: *USB selection screen*

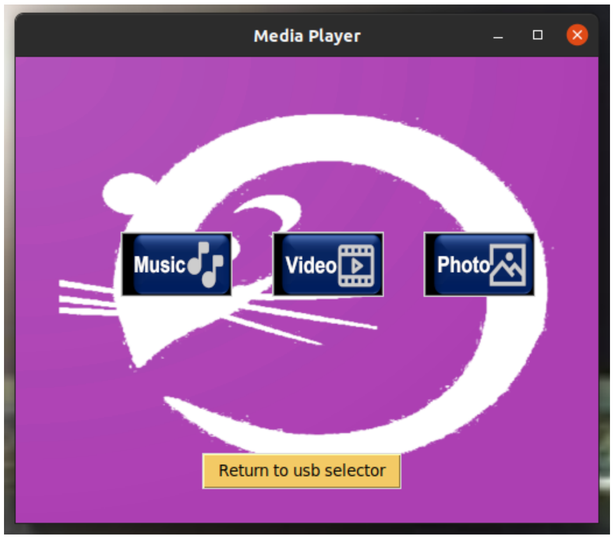

Figure 4: *USB selection screen*

As you can imagine, each button redirects you into another window, each one with its unique design and functions. First, we have the Music screen; it works by reading all files and taking all the ones that match the extension of the file, in this case, `.mp3`.

```
...

#Listing the labeled array
                    for song in MusicList:
                              TextSong=Label(wind4,text=song)
                              NameSong = song
                              TextSong.pack()

                    #Initializing the player with
                    #the first song of the list
                    mixer.init()

                    #For use on a raspberryPI uncomment
```

```
                                        #the following line
                                        #mixer.music.load("/media/pi/"+usb+"/"
                                        +MusicList[NumSong[0]])

                                        #For use on a linux filesystem uncomment
                                        #the following line
                                        mixer.music.load("/media/"+os.getlogin()+
                                        "/"+usb+"/"+MusicList[NumSong[0]])


                                        mixer.music.set_volume(0.5)
                                        mixer.music.play()

                                        ...
```

This section of the code is the one that saves the marching files (if any) and then sends it in an array to the music player.

For the controls we implemented different functions to go through the array in an specific order or to make it pause the current song, play the next one and the previous one. All by moving the array index depending on the function or by simply giving the signal to the player to make the desired action.

```
    ...

    #Stopping the current song and
    #starting the previous one
                    def PrevSong(NumSong):
                            #If we are at the
```

```
#last song, go back to the first one
if NumSong[0] == 0:
        NumSong[0] = len(MusicList)-1
else:
        NumSong[0] -= 1

#Stop
mixer.music.stop()

#Routing the player

#For use on a raspberryPI uncomment
#the following line
#mixer.music.load("/media/pi/"+usb+"/"
+MusicList[NumSong[0]])

#For use on a linux filesystem
#uncomment the following line
mixer.music.load("/media/"+
os.getlogin()+"/"
+usb+"/"+MusicList[NumSong[0]])

#Play the file
mixer.music.play()
```

...

The functionality for the other two media players follows the same rules, they get the analyzed array of files that matches their criteria and then make it possible for the user to watch said contents, in the case of the photo player it gets all the matching files and start showing them one by one following an interval between photo and photo.

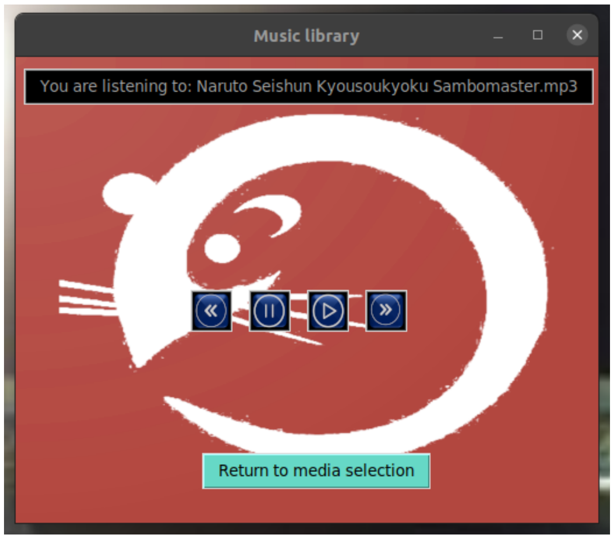
Music library
You are listening to: Naruto Seishun Kyousoukyoku Sambomaster.mp3
Return to media selection

Figure 5: *Music Player screen*

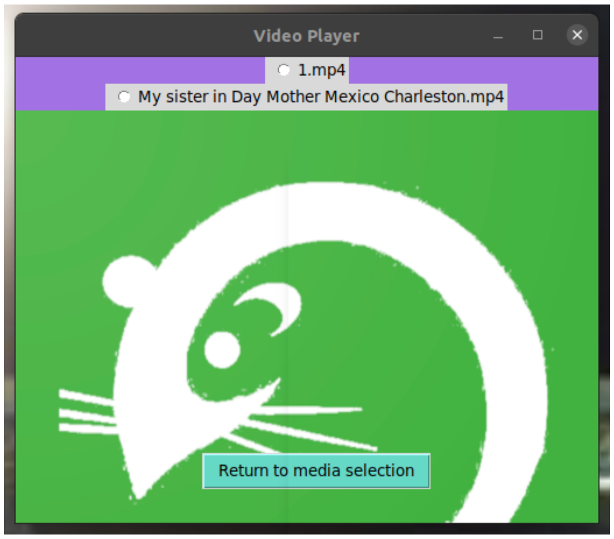
Video Player
1.mp4
My sister in Day Mother Mexico Charleston.mp4
Return to media selection

Figure 6: *Video Player screen*

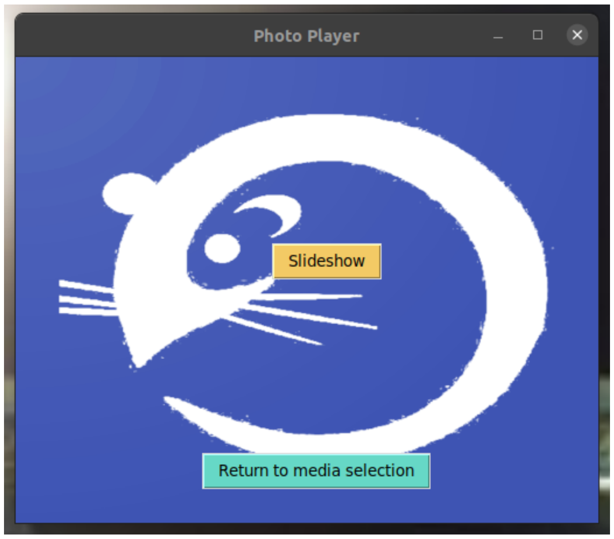

Figure 7: *Photo Player screen*

6 Conclusions

As you can see there are a lot of things to consider before even thinking to implement a media center or a software like this to begin with. We can make a lot of more functionalities and add them into the program to make it more compelling for the user, like opening different streaming sites or letting them enter an external device with their personal files inside them to play or watch together as they want. The idea is to keep in mind the commodities that the software has to accomplish and looking for an effective and simple way to deliver it.

During the development of this tool we found quite the challenge, there were a lot of things to consider and like said before, we had to figure out the best way to get it done without using too much resources or making it hardware/software specific. In the end, we made what we could with what we had at our disposal. We believe it paid off after all the time invested.

7 Questionnaire

Read and answer the following questions, please, justify your answers

- What do you think are the limitations of a program like this one?
- What other approaches do you think can be implemented using a program like this one?
- Do you think that it would be necessary to have knowledge of embedded systems to implement a system like this one? Why? Why not?

8 References

References

[1] Back-End, T. R. (2022). Raspberry pi – run python script in the terminal. `https://roboticsbackend.com/raspberry-pi-run-python-script-in-the-terminal/`. Retrieved: 26/05/2022.

[2] Ltd, S. H. P. (2022). Sublime text - text editing, done right. `https://www.sublimetext.com/`. Retrieved: 26/05/2022.

[3] Overleaf (2022). Overleaf - documentation. `https://es.overleaf.com/learn`. Retrieved: 26/05/2022.

[4] Petty, L. (2017). Health and safety when working with computers: An office guide. `https://www.highspeedtraining.co.uk/hub/computer-health-and-safety/`. Retrieved: 26/05/2022.