Final project - Media Center

Paul Jaime Felix Flores.

May 26 Th, 2022

1 Objetive

The media center project specified by the teacher will be created.

2 Material

None.

3 Introduction

First we will define what an embedded system is, we can define it microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations. Complexities range from a single microcontroller to a suite of processors with connected peripherals and networks; from no user interface to complex graphical user interfaces. The complexity of an embedded system varies significantly depending on the task for which it is designed.Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems.
A Media Center is a computer adapted to play music, view movies and images stored on a local hard drive or over a LAN computer network (in some cases), view DVD movies, and often view and record broadcasts. of TV. Some software is capable of doing other tasks, such as fetching news feeds (RSS) from the internet. Media Centers are often operated with remote controls, connected to a television set for video output, and can often function as a personal computer. A media center can be purpose built, or created by individuals adding media center software to a PC or some other computer, for example an Xbox. With this knowledge we will try to achieve a multimedia center implemented as an embedded system and that works correctly.
[1] [2] [3]

4 You can download the project from here:

- `https://github.com/Darkahnott/ears_pjff_FinalProject`

5 You can see the video in the following link:

- `https://youtu.be/aexmLuIcrpM`

6 Safety Measures

- The only recommended measures, since they are not something so exaggerated, because there is no physical installation, however, we recommend you not to spend too much time on the screen, since your vision could be damaged, not to spend so much time in the chair, so as not to affect problems with the spine.

7 Project specifications

7.0.1 Entertainment center for playing movies, videos, music, and photos. The embedded system described should perform the following functions:

- Optional internet connection wizard.
- Power-up menu that will allow you to choose between online services for video (example, Netflix, HBO-Go, Blim), music (example, Spotify, Cloudfare, Deezer, etc.) or play removable media.
- When a removable medium (USB memory) with photos is inserted, they will be played in a presentation.
- When a removable media (USB memory) with music is inserted, the tracks will be played one after the other in another in infinite loop.
- When a removable media (USB stick) with videos is inserted, it should be possible to choose which movie play or whether the videos will be played one after another in slideshow mode.
- When a removable medium (USB memory) with mixed content is inserted, the device should ask what action to take
- Remote control support (pointer and on-screen keyboard is acceptable)

8 Software modules used

8.0.1 We implemented five modules to make our multimedia center work, they were created according to the project specifications.

- We import certain libraries
  - from tkinter import *:Imports every exposed object in Tkinter into your current namespace. import Tkinter imports the "namespace" Tkinter in your namespace and import Tkinter as tk does the same, but "renames" it locally to 'tk' to save you typing.
  - from pygame import mixer:In order to play music/audio files in pygame , pygame.mixer is used (pygame module for loading and playing sounds). This module contains classes for loading Sound objects and controlling playback.
  - import webbrowser:The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will open url using the default browser . You have to import the module and use open() function.
  - import cytpes: Ctypes is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python.
  - import os: This module provides a versatile way to use operating system dependent functionality. If you want to read or write a file see open() , if you want to manipulate paths see the os.
  - import vlc: The VLC media player is a free and open-source portable cross-platform media player software and streaming media server developed by the VideoLAN project. VLC is available for desktop operating systems and mobile platforms, such as Android, iOS, iPadOS, Tizen, Windows 10 Mobile, and Windows Phone. We can use the VLC media player with the help of python as well, in order to install vlc module in python we will use the command given below.
  - import Time: This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

- In our modules, we have a module called USB has 3 specific functions, when it reads the USB, when it reads the files from it and when it is disconnected. In our opinion it was the heaviest, so together with my colleague we programmed the functions so that our embedded system could have this module and work in the most correct way.

```
#Function to show the available USB directories
def USBNav(root):

        #Deletes the recieved window
        root.destroy()

        #Starting the window with the desired styling
        wind2 = Tk()
        wind2.title("USB Selection Screen")
        wind2.geometry('500x400')
        wind2.config(bg="#8dc3d5")

        bg=(PhotoImage(file = "Resources/img/back/logo3.png"))
        #Setting up the label
        Start=Label(wind2,text="Media Center",image=bg)
        Start.pack()

        #Managing the USB filesystems

        #For use on a raspberryPI uncomment the following line
        #path = "/media/pi/"

        #For use on a linux filesystem uncomment the following line
        path = "/media/"+os.getlogin()+"/"

        #Listing all the files on the USB
        USBS = os.listdir(path)

        #Counting the number of USB devices
        numUsb = len(USBS)
```

Figure 1: *Module when it recognizes the USB*

```
#Function to show the different media players
def EnterUSB(root,usb):

        #Deletes the recieved window
        root.destroy()

        #Starting the window with the desired styling
        wind3 = Tk()
        wind3.title("Media Player")
        wind3.geometry('500x400')

        bg1=(PhotoImage(file = "Resources/img/back/logo2.png"))
        #Setting up the label
        Test=Label(wind3,image=bg1)
        Test.pack()


        #Show the button to choose to play "Music"
        MusicIMG=PhotoImage(file='Resources/img/buttons/Music.png')
        MusicButton=Button(wind3,image=MusicIMG,command=lambda:playMusic(wind3,usb),width=90,height=50,bg='black')
        MusicButton.place(x=90,y=150)

        #Show the button to choose to play "Videos"
        VideoIMG=PhotoImage(file='Resources/img/buttons/Video.png')
        VideoButton=Button(wind3,image=VideoIMG,command=lambda:playVideo(wind3,usb),width=90,height=50,bg='black')
        VideoButton.place(x=220,y=150)

        #Show the button to choose to watch "Images"
        PhotoIMG=PhotoImage(file='Resources/img/buttons/Photo.png')
        PhotoButton=Button(wind3,image=PhotoIMG,command=lambda:playImage(wind3,usb),width=90,height=50,bg='black')
        PhotoButton.place(x=350,y=150)
```

Figure 2: *Module when USB is selected*

```python
#Function that manages the use cases for wether the device was disconnected or its empty
def Unplugged(root,usb):

    #Deletes the recieved window
    root.destroy()

    #Starting the window with the desired styling
    windTemp = Tk()
    windTemp.title("Device offline")
    windTemp.geometry('500x400')
    windTemp.config(bg="#5FCC8F")

    #Error message
    errorMSG=Label(windTemp,text="Error while reading the device: "+usb+"\nCheck if it contains valid files or if it is still connected.")
    errorMSG.place(x=60,y=100)

    #Control buttons, allows you to go back to the previous menu
    GoBack=Button(windTemp,text="Return to previous menu",bg="#F64A5C",command=lambda:USBNav(windTemp))
    GoBack.place(x=160,y=340)
```

Figure 3: *Module when USB is disconnected*

- In the module of the links of the web page, there are several modules that will take us to the corresponding websites

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

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

#Function to open the desired service using the default web browser
def Youtube():
        webbrowser.open("https://www.youtube.com/",new=2, autoraise=True)

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

#Function to open the desired service using the default web browser
def PrimeVideo():
        webbrowser.open("https://www.primevideo.com/",new=2, autoraise=True)

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

Figure 4: *Web page module*

- The following module corresponds to when the USB reads the songs, this module has the specific task of reading the USB and reading all the music it has (only in mp3 format), in this module I was the one who took care of me the most, I took care of what read in a list all the songs that were read respectively from the USB.In turn, I programmed the buttons for the next song, for the song, stop the song and the previous song, they were complicated at first to understand how I would do it, but over time I became a little more skilled in how to implement it. I had to be investigating how to implement certain things, since I had never delved into this topic, but I found things on the internet that I was able to implement.

```
#Creating and styling the button to "Play previous song"
PrevSongIMG=PhotoImage(file='Resources/img/buttons/Prev.png')
PrevSongButton=Button(wind4,image=PrevSongIMG,command=lambda:PrevSong(NumSong),width=30,height=30,bg='black')
PrevSongButton.place(x=150,y=200)

#Creating and styling the button to "Pause song"
PauseSongIMG=PhotoImage(file='Resources/img/buttons/Pause.png')
PauseButton=Button(wind4,image=PauseSongIMG,command=lambda:mixer.music.pause(),width=30,height=30,bg='black')
PauseButton.place(x=200,y=200)

#Creating and styling the button to "Play"
PlaySongIMG=PhotoImage(file='Resources/img/buttons/Play.png')
ResumeButton=Button(wind4,image=PlaySongIMG,command=lambda:mixer.music.unpause(),width=30,height=30,bg='black')
ResumeButton.place(x=250,y=200)

#Creating and styling the button to "Play next song"
NextSongIMG=PhotoImage(file='Resources/img/buttons/Next.png')
NextSongButton=Button(wind4,image=NextSongIMG,command=lambda:NextSong(NumSong),width=30,height=30,bg='black')
NextSongButton.place(x=300,y=200)

#Listening to
NameSongButton=Button(wind4,text="You are listening to: "+NameSong,bg="black",state=DISABLED)
NameSongButton.place(x=7,y=10)

#Creating and styling the button to "Exit the media player"
ExitButton=Button(wind4,text="Return to media selection",bg="#66D8C6",command=lambda:ExitMedia(wind4,usb))
```

Figure 5: *Music Module*

```
#Function Music player
def playMusic(root,usb):

        #Deletes the recieved window
        root.destroy()

        #Starting the window with the desired styling
        wind4 = Tk()
        wind4.title("Music library")
        wind4.geometry('500x400')

        bg=(PhotoImage(file = "Resources/img/back/logo5.png"))

        #Setting up the label
        Start=Label(wind4,text="Media Center",image=bg)
        Start.pack()

        try:
                #Getting all the files from the device

                #For use on a raspberryPI uncomment the following line
                #path = "/media/pi/" + usb

                #For use on a linux filesystem uncomment the following line
                path = "/media/"+os.getlogin()+"/"+ usb


                #Creating the list of contents
                files = os.listdir(path)
```

Figure 6: *When the music plays*

```
def NextSong(NumSong):
        #If we are playing the first song, go to the last one
        if NumSong[0] == len(MusicList)-1:
                NumSong[0] = 0
        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 raspberryPI uncomment the following line
        mixer.music.load("/media/"+os.getlogin()+"/"+usb+"/"+MusicList[NumSong[0]])


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

Figure 7: *When you advance to the next song*

- The next module is the images module, in this module it reads all the images obtained from the USB, read only (.jpg and .png), it will show us a slider that will last the time required by the project requirements.

```
#Image player
def playImage(root,usb):
        root.destroy()
        #Styling and creating window
        wind6 = Tk()
        wind6.title("Photo Player")
        wind6.geometry('500x400')
        wind6.config(bg="#69A6E7")

        bg=(PhotoImage(file = "Resources/img/back/logo6.png"))
        #Setting up the label
        Start=Label(wind6,text="Photos",image=bg)
        Start.pack()



        try:
                #Getting all the files from the device

                #For use on a raspberryPI uncomment the following line
                #path = "/media/pi/" + usb

                #For use on a linux filesystem uncomment the following line
                path = "/media/"+os.getlogin()+"/" + usb


                #Listing all the files
                files = os.listdir(path)
```

Figure 8: *Image Module*

- The last module is the video player, in this module I worked with my colleague and it is the one that cost us the most work of the power implemented, since they were not something as trivial as the other 3, however we got the objective, because we obtained a complication, we believe that it can be improved in case we want to continue with this project,thereby most of our code is commenting so someone with basic knowledge of this area could greatly improve this project.

```
#Video player
def playVideo(root,usb):
        root.destroy()
        #Styling and creating window
        wind5 = Tk()
        wind5.title("Video Player")
        wind5.geometry('500x400')
        wind5.config(bg="#A272E4")



        try:
                #Getting all the files from the device

                #For use on a raspberryPI uncomment the following line
                #path = "/media/pi/" + usb

                #For use on a linux filesystem uncomment the following line
                path = "/media/"+os.getlogin()+"/" + usb

                #Creamos una lista con todos los files dentro de la usb
                files = os.listdir(path)

                #Creating an aray to save the files
                VidList = []
```

Figure 9: *Video Module*

9 Questionnaire

9.0.1 ¿Can it be run on a Raspberry Pi card?

Yes of course ! It can be run on a card since only certain lines of code need to be modified.

9.0.2 ¿How important is a multimedia center today?

A multimedia center is important since in these times of modernization, what is mostly wanted is that everything is at hand and that it works, so this multimedia center can be bought by companies so that in this way they obtain more users to their platforms.

10 Conclusions

In this Project I have learned a lot about embedded systems, I must say that it is the most complicated project I have done throughout my career, either because of the specifications it has. However, I think it is a great learning, to be able to carry out a high-level project.

We decided to create the multimedia center from the console, since we did not have the economic resources to buy the card for the material, however, our project can be adapted to the card since the changes are not so many, it is only to uncomment certain lines of code.

The most complicated thing was to create the USB model, since it is not something so trivial, since we must see that the system recognizes the USB, as well as when disconnecting it and reading the information that it has in its inside.The most complicated thing was to go to other pages, since here there is not so much science.

I believe that we met most of the objectives according to the project specifications, however, it was not as perfect as we wanted since we had two errors, it is worth mentioning that we believe that it can be improved and implemented for any company that wants to use it , since our errors are not so serious.

What can be implemented in the future could be to apply it in a web application, as well as improve errors, protect it from possible hacks, as well as improve the code and do it in fewer lines of code. Another detail that escaped us is to create the control, since we did not know how to implement it, but I think that this knowledge can be implemented so that the project works perfectly

References

[1] "Centro multimedia." (2017), [Online]. Available: `https://es-academic.com/dic.nsf/eswiki/245848` (visited on 03/22/2022).

[2] "Sistema embebido y sus características — conceptos fundamentales." (2021), [Online]. Available: `https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas` (visited on 05/23/2022).

[3] V. "¿qué son los sistemas embebidos?" (2016), [Online]. Available: `https://www.azulweb.net/que-son-los-sistemas-embebidos/` (visited on 05/23/2022).