

# **LAB1**

**Wednesday, Group6**

**September 24, 2021**

**Yanan Liu(yl2248), Suhui Yu(sy466)**

# Introduction

The first experiment is to give us a preliminary understanding of the Raspberry Pi. By manually assembling the Raspberry Pi system, connect external devices such as PiTFT, Keyboard, mouse and monitor to the Raspberry Pi. By setting up the Raspberry Pi and using Python and Bash to control PiTFT and play video and audio with the Raspberry Pi. Lab1 also familiarizes us with the GPIO pins on the Raspberry Pi, and controls the video playback through the pins.

## Week1

### 1. Set up Raspberry Pi:

We should plug the PiTFT into the RPi 40 GPIO connector, it should pay attention to the direction of 40-pin cable header when plug on the PiTFT and RPi. After the cable is oriented correctly, we need to connect the keyboard and mouse through the USB interface, connect the monitor through HDMI, and connect to the network cable. At the same time, we also need to insert the SD card into the Raspberry Pi. When the above operations are completed, we can power-up the Raspberry Pi system.

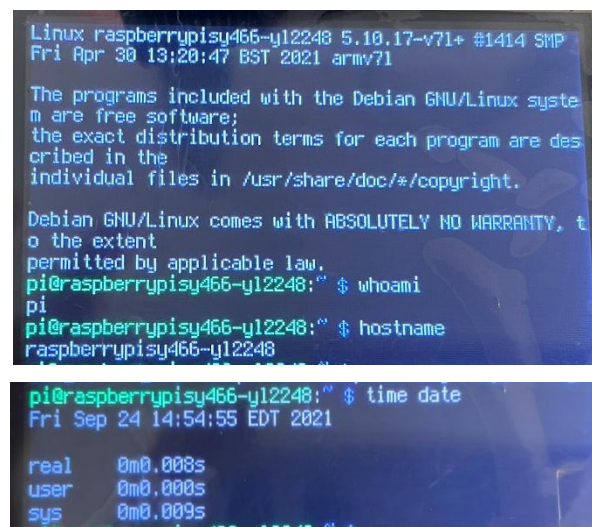
### 2. Set up System:

We have already load the Raspbian linux kernel to SD card before Lab1. As long as we insert the SD card into the Pi, we can access the Raspbian Linux OS.

Then, use the command 'sudo raspi-config' to setup the system from the console window directly. We need to set the hostname and password, configure our Wifi connection, setup localization (change the locale to 'en-us. UTF8 UTF8' and Timezone to US, Eastern), configure the keyboard layout.

After setting all of these down, we need to reboot the Pi to make these changes work on the Pi.

When we check these setup, we found that the time is wrong. After guidance from the professor, we solve this problem by registering our address on Cornell IT website.



```
Linux raspberrypisy466-y12248 5.10.17-v71+ #1414 SMP
Fri Apr 30 13:20:47 BST 2021 armv7l

The programs included with the Debian GNU/Linux system
are free software;
the exact distribution terms for each program are des-
cribed in the
individual files in /usr/share/doc/*/copyright.

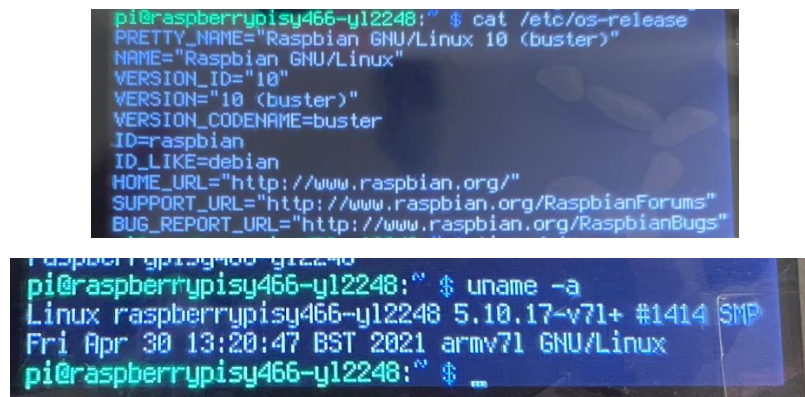
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to
the extent
permitted by applicable law.
pi@raspberrypisy466-y12248:~$ whoami
pi
pi@raspberrypisy466-y12248:~$ hostname
raspberrypisy466-y12248
pi@raspberrypisy466-y12248:~$ time date
Fri Sep 24 14:54:55 EDT 2021

real    0m0.008s
user    0m0.000s
sys     0m0.009s
pi@raspberrypisy466-y12248:~$
```

Figure1.1 Setup the System

### 3. Update Kernel:

- 1) Login as a pi user, we need to update the Raspbian kernel, here are commands:
  - `sudo apt-get update`
  - `sudo apt-get full-upgrade`
- 2) After update completed, we need to reboot the Pi, and then to load an update for the vim editor:
  - `sudo apt-get install vim`
- 3) We can use the command 'startx' to see the command line on the monitor. And then, we can check whether the setup is successful. When all of items finished, we must back up our SD cards.



```
pi@raspberrypisy466-y12248:~$ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"

pi@raspberrypisy466-y12248:~$ uname -a
Linux raspberrypisy466-y12248 5.10.17-v7l+ #1414 SMP
Fri Apr 30 13:20:47 BST 2021 armv7l GNU/Linux
pi@raspberrypisy466-y12248:~$
```

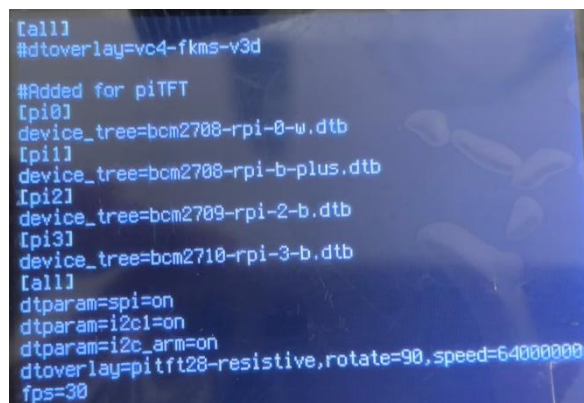
Figure1.2 The OS information

### 4. Set up PiTFT

- 1) install the PiTFT:

First of all, we need to install python-pip to install software to support the PiTFT, including git, python-pip, python-smbus, python-spideve, evtest and lists-bin. We also need to install the package: evdev.

Secondly, we need to edit the file /boot/config.txt to add PiTFT info into it and reboot the Pi, we saw the PiTFT screen started out white and turned into the black, which means the configurations are working successfully.



```
[all]
#dtoverlay=vc4-fkms-v3d

#Added for piTFT
[pi0]
device_tree=bcm2708-rpi-0-w.dtb
[pi1]
device_tree=bcm2708-rpi-b-plus.dtb
[pi2]
device_tree=bcm2709-rpi-2-b.dtb
[pi3]
device_tree=bcm2710-rpi-3-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=piTFT28-resistive,rotate=90,speed=64000000,
fps=30
```

Figure1.3 The content of configure.txt

Thirdly, we need run dmesg to check that all of modules for the touch screen has been installed to the Pi. The "stmpe-spi" and "graphics fb1" need to be checked particularly.

```

pi@raspberrypisy466-yl2248:~$ dmesg | grep stmpe-spi
[ 4.716961] stmpe-spi spi0.1: stmpe610 detected, chip id: 0x811
pi@raspberrypisy466-yl2248:~$ dmesg | grep graphics
[ 10.103939] graphics fb1: fb_ili9340 frame buffer, 320x240, 150 KiB video mem
ory, 4 KiB buffer memory, fps=33, spi0.0 at 64 MHz
pi@raspberrypisy466-yl2248:~$

```

Figure1.4 Content of dmesg verifying a successful installation

## 2) Setup the PiTFT

Add udev rule: we need to add a udev rule to make the PiTFT to become a touchscreen. The evtest is a very useful way to verify the touchscreen operation. When evtest is running, every time we touch the screen of PiTFT, an event will be showed on the command line.

```

pi@raspberrypisy466-yl2248:~$ cat /etc/udev/rules.d/95-stmpe.rules
SUBSYSTEM=="input", ATTRS{name}=="*stmpe*", ENV{DEVNAME}=="*event*",
SYMLINK+="input/touchscreen"
pi@raspberrypisy466-yl2248:~$ cat /etc/udev/rules.d/95-touchmouse.rules
SUBSYSTEM=="input", ATTRS{name}=="touchmouse", ENV{DEVNAME}=="*event*",
SYMLINK+="input/touchscreen"
pi@raspberrypisy466-yl2248:~$ cat /etc/udev/rules.d/95-ftcaptouch.rules
SUBSYSTEM=="input", ATTRS{name}=="EP0110M09", ENV{DEVNAME}=="*event*",
SYMLINK+="input/touchscreen"
pi@raspberrypisy466-yl2248:~$ sudo rmmod stmpe-ts
pi@raspberrypisy466-yl2248:~$ sudo modprobe stmpe-ts
pi@raspberrypisy466-yl2248:~$ ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Sep 24 16:32 /dev/input/touchscreen -> event2
pi@raspberrypisy466-yl2248:~$

```

Figure1.5.1 Associate with an event number

```

Properties:
Testing ... (interrupt to exit)
Event: time 1632515668.020176, type 3 (EV_ABS), code 0 (ABS_X), value 737
Event: time 1632515668.020176, type 3 (EV_ABS), code 1 (ABS_Y), value 3267
Event: time 1632515668.020176, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 201
Event: time 1632515668.020176, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1632515668.020176, ----- SYN_REPORT -----
Event: time 1632515668.028336, type 3 (EV_ABS), code 0 (ABS_X), value 709
Event: time 1632515668.028336, type 3 (EV_ABS), code 1 (ABS_Y), value 3264
Event: time 1632515668.028336, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 141
Event: time 1632515668.028336, ----- SYN_REPORT -----
Event: time 1632515668.037231, type 3 (EV_ABS), code 0 (ABS_X), value 711
Event: time 1632515668.037231, type 3 (EV_ABS), code 1 (ABS_Y), value 3280
Event: time 1632515668.037231, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 138
Event: time 1632515668.037231, ----- SYN_REPORT -----
Event: time 1632515668.045643, type 3 (EV_ABS), code 0 (ABS_X), value 689
Event: time 1632515668.045643, type 3 (EV_ABS), code 1 (ABS_Y), value 3249
Event: time 1632515668.045643, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 140
Event: time 1632515668.045643, ----- SYN_REPORT -----

```

Figure1.5.2 Data associated with each tap

Set initial calibration: setup information for calibration of the PiTFT.

Start console window: we need to modify some files and then reboot the Pi, and then the console window will be start on the PiTFT.

## 3) Play video:

Firstly, we need to install the mplayer if the result of checking is not found the mplayer. And then, we can play the video on the PiTFT by using the command:

- `sudo SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop bigbuckbunny320p.mp4`

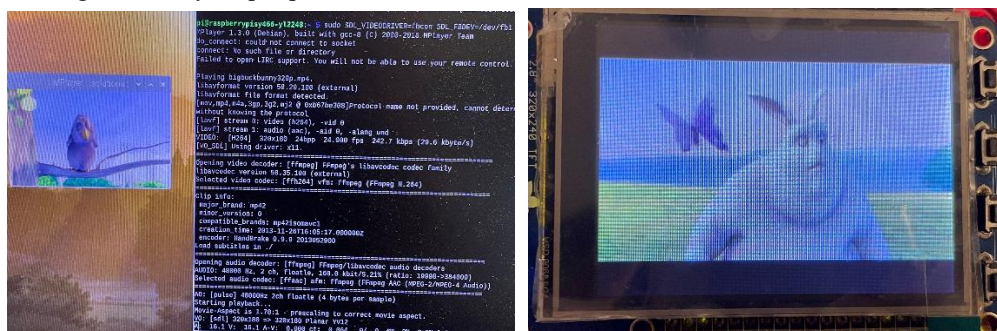


Figure1.6 The video play on the monitor and Raspberry Pi



We also should test playing video from different consoles, here are results:

	ssh	PiTFT	monitor
sudo	video / no audio	video/ no audio	video/ audio
without sudo	audio/ no video	video/ audio	video/ audio

## Week2

### 1. Control mplayer with a FIFO:

FIFO means ‘First In First Out’. FIFO file works like a queue or a pipe, so we don’t need a related process to exchange data. FIFO enables us to transfer command to PiTFT as a file.

We use the “startx” command, to make the command line display on the monitor so that we can operate the Pi more convenient.

Firstly, create a fifo called video\_fifo by mkfifo command. Open another console window to run mplayer and use the previous window to sent the command to mplayer via fifo. Here are commands:

**Terminal 1:**

- \$ mplayer -input file=video\_fifo bigbuckbunny320p.mp4

**Terminal 2:**

- \$ echo “pause” > video\_fifo
- \$ echo “quit” > video\_fifo

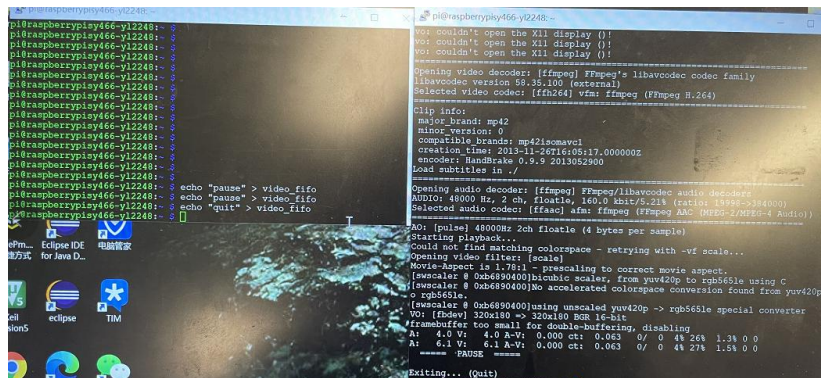


Figure2.1 Control mplayer with FIFO

The video paused and quit as we expected.

### 2. use Python to control mplayer with a FIFO:

We need to create a python program named the fifo\_test.py. We use a while loop to get the input instructions constantly:

- cmd = raw\_input(“Enter a command:”)

There should be an “if” statement to determine the special case that cmd is “exit”. In this case, the while loop should be continuing the loop and wait for input anymore, it should quit immediately.

And then create a string variable named cmd1 to create the command to pass to terminal, and use the subprocess.check\_output() function pass the command to Bash.

- cmd1 = “echo ”+ cmd + “ > video\_fifo”
- subprocess.check\_output(cmd1, shell=True)

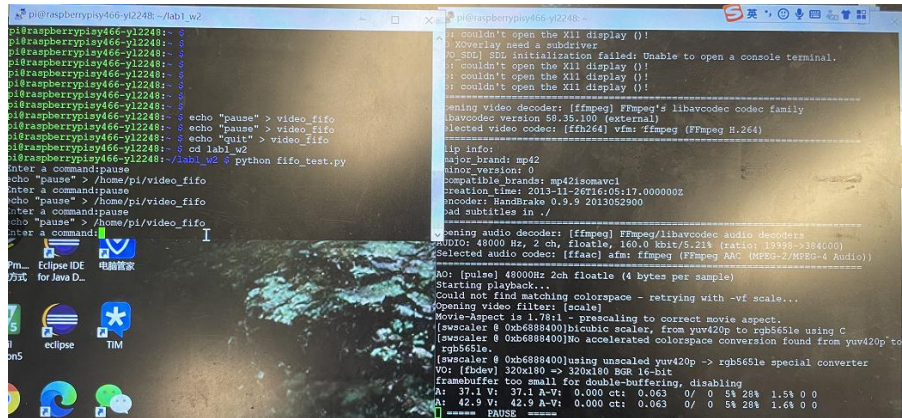


Figure2.2 Use python to control mplayer with FIFO

We can see the result in the picture. We used one of the console window to run the mplayer and use the another window to run the python file. When we type the “pause”, the video will stop as we expected. The other commands also worked successfully.

However, this part couldn’t work at the beginning.

The first reason is that we forgot to type the space after the “echo”, so we create a wrong command that couldn’t be recognized.

Secondly, we found that this was because we create a python program in the directory of Lab1\_w2. However, the video\_fifo had in the /home/pi directory, so we have to add a root to make sure the Pi to find it. The code should be like this:

- cmd1 = “echo ”+ cmd + “ > /home/pi/video\_fifo”
3. Get input from button connected to GPIO:

In this parts, we need to initialize the buttons as the GPIO inputs. In the PiTFT schematic, we can find which GPIO pins the buttons connect to and we can also know that when the button was pressed, the GPIO will have a fall edge. In program, GPIO will read high normally and will read low when the button is pressed.

If we want to use python, the first thing is to import the package and active the Broadcom specific pin numbers:

- import RPi.GPIO as GPIO
- import time
- GPIO.setmode(GPIO.BCM)

**one button:**

For the one button parts, we just need to initialize one button connected to the GPIO as input, and when we detect the pin read the low, the program should print a sentence, “the button 22 pressed”.

- GPIO.setup(22, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

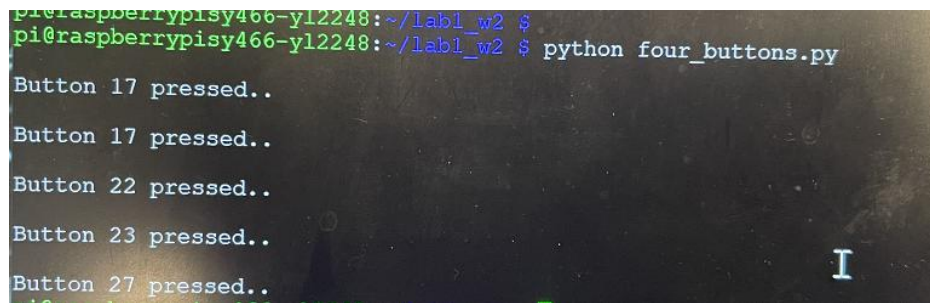


Figure2.3 One button was pressed

**four button:**

As for the parts of four buttons, the code will be based on the one\_button.py. We need to initialize the other buttons as same as the first button and detect the state constantly. When the

button is pressed, GPIO input will become low, and the program should print the sentence to remind us and help us to test the result.



```

pi@raspberrypi466-y12248:~/lab1_w2 $
pi@raspberrypi466-y12248:~/lab1_w2 $ python four_buttons.py

Button 17 pressed..

Button 17 pressed..

Button 22 pressed..

Button 23 pressed..

Button 27 pressed..

```

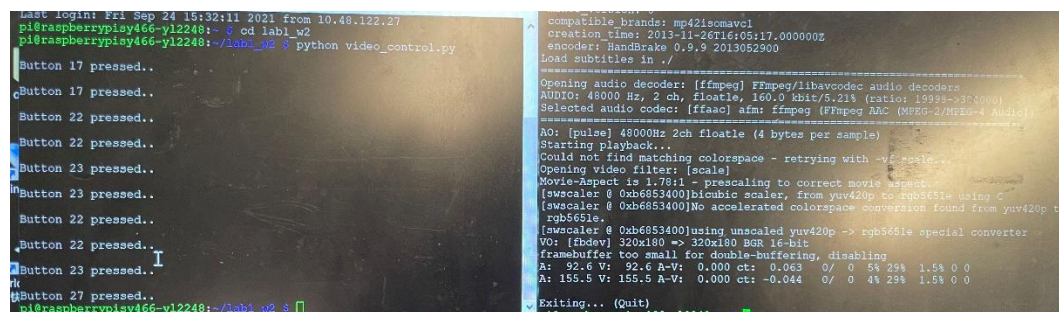
Figure2.4 Four buttons were pressed separately

#### 4. Control mplayer through FIFO using Python:

As we have already initialized 4 buttons in the previous step, now we have to create a file called “video\_control.py” to control the mplayer using the four buttons.

We create this new “video\_control.py” program by reediting the previous “four\_buttons.py” file. We just added “subprocess.check\_output()” function to pass the button command to video\_fifo. In our case, we have pin 17 for pause, pin 27 for quit, pin 22 for fast forward, pin 23 for backward.

After setting up four buttons with different function: pause, fast forward, rewind and quit, we open two console windows to check whether it works correctly. In the first window we run the mplayer, in the second console we ran “video\_control.py”. Once we hit the button, the video pause, fast forward, rewind and quit the code in our python file. What’s more, we also added a break to terminate both mplayer and the python program if we hit “quit” button (pin 27).



```

Last login: Fri Sep 24 15:32:11 2021 from 10.48.122.27
pi@raspberrypi466-y12248:~$ cd lab1_w2
pi@raspberrypi466-y12248:~/lab1_w2 $ python video_control.py

Button 17 pressed..

Button 17 pressed..

Button 22 pressed..

Button 22 pressed..

Button 23 pressed..

Button 23 pressed..

Button 22 pressed..

Button 22 pressed..

Button 23 pressed..

Button 27 pressed..

pi@raspberrypi466-y12248:~/lab1_w2 $

compatible_brands: mp42isomavc1
creation_time: 2013-11-26T16:05:17.000000Z
encoder: HandBrake 0.9.9 2013052900
Load subtitles in ./
=====
Opening audio decoder: [ffmpeg] FFmpeg/libavcodec audio decoder
AUDIO: 48000 Hz, 2 ch, floatle, 160.0 kbit/5.21% (ratio: 19999--331000)
Selected audio codec: [ffaac] afm: ffmpeg (FFmpeg AAC (MPEG-2/MPEG-4 Audio))
=====
AO: [pulse] 48000Hz 2ch floatle (4 bytes per sample)
Starting playback...
Could not find matching colorspace - retrying with -vf scale...
Opening video filter: [scale]
Movie-Aspect is 1.78:1 - prescaling to correct movie aspect.
[swscale @ 0xb6853400]bicubic scaler, from yuv420p to rgb565le using C
[swscale @ 0xb6853400]No accelerated colorspace conversion found from yuv420p to
rgb565le.
[swscale @ 0xb6853400]using unscaled yuv420p -> rgb565le special converter
VO: [fbdev] 320x180 => 320x180 BGR 16-bit
framebuffer too small for double-buffering, disabling
A: 92.6 V: 92.6 A-V: 0.000 ct: 0.063 0/ 0 5% 2% 1.5% 0 0
A: 155.5 V: 155.5 A-V: 0.000 ct: -0.044 0/ 0 4% 2% 1.5% 0 0
Exiting... (Quit)
pi@raspberrypi466-y12248:~$

```

Figure2.5 control the mplayer using python

(When we took this picture, we didn’t have a monitor, so we change the fb1 to fb0 in the video\_fifo file to display the video on the Pi. And print the button message when the button was pressed)

#### 5. Bash script:

In order to make full control of the entire program, we created a bash file called “start\_video” to launch the mplayer and run the “video\_control.py” in the background as well.

First of all, we should include “#!/bin/bash” in the first line of our program, which tells the OS this is a bash file.

Then, the second line we added “python video\_control.py &”, this is to run the python code on the background the “&” sign means the python file works in the background.

Finally, we run the mplayer and to tell the PiTFT to get command from video\_fifo using this command:



- `sudo SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop -input file=/home/pi/video_fifo /home/pi/bigbuckbunny320p.mp4`

```
Opening video decoder: [ffmpeg] FFmpeg's libavcodec codec family
libavcodec version 58.35.100 (external)
Selected video codec: [ffh264] vfm: ffmpeg (FFmpeg H.264)
=====
Clip info:
major_brand: mp42
minor_version: 0
compatible_brands: mp42isomavc1
creation_time: 2013-11-26T16:05:17.000000Z
encoder: HandBrake 0.9.9 2013052900
Load subtitles in /home/pi/
=====
Opening audio decoder: [ffmpeg] FFmpeg/libavcodec audio decoders
AUDIO: 48000 Hz, 2 ch, floatle, 160.0 kbit/5.21% (ratio: 19998->384000)
Selected audio codec: [ffaac] afm: ffmpeg (FFmpeg AAC (MPEG-2/MPEG-4 Aud
io))
=====
AO: [pulse] Init failed: Connection refused
Failed to initialize audio driver 'pulse'
AO: [alsa] 48000Hz 2ch floatle (4 bytes per sample)
Starting playback...
Movie-Aspect is 1.78:1 - prescaling to correct movie aspect.
VO: [sdl] 320x180 => 320x180 Planar YV12
: 113.7 V: 113.7 A-V: -0.017 ct: 0.080 0/ 0 5% 37% 1.3% 0 0
Button 22 pressed..
: 133.5 V: 133.5 A-V: -0.005 ct: -0.017 0/ 0 4% 37% 1.4% 0 0
Button 22 pressed..
: 147.0 V: 147.0 A-V: -0.007 ct: -0.015 0/ 0 4% 30% 1.9% 0 0
Button 22 pressed..
: 162.1 V: 162.1 A-V: -0.018 ct: -0.006 0/ 0 5% 38% 1.8% 0 0
Button 22 pressed..
: 491.2 V: 491.2 A-V: -0.008 ct: -0.026 0/ 0 5% 37% 1.3% 0 0
Button 27 pressed..
A: 491.3 V: 491.3 A-V: 0.017 ct: -0.023 0/ 0 5% 37% 1.3% 0 0
Exiting... (Quit)
pi@raspberrypis466-y12248:~/lab1_w2 $
```

Figure2.6 Using bash file to display and control the video

(As we mentioned above, we didn't have a monitor. We can see using the bash file to execute the python file is different from executing in terminal. When they print the button message, they will insert in the mplayer message)

However, when we executed our bash file, we failed. After stepping into the file carefully, we found 2 problems:

- 1) As the video file is not in the same directory of the bash file, we need to add the location of the video file so that it can find where it is.
- 2) we cannot execute the file, since we have to change the permission of the bash file to 744.

## Results:

1. We have achieved all the goals outlined at the beginning of Lab 1 file smoothly, and we have demonstrated all our results to TAs on time.
2. We have uploaded our code to the class server located at "/home/Lab1/W\_y12248\_sy466\_Lab1"
3. The most part of lab worked very well, however, we did encounter some issues during our lab.

### ***Problem1: The date of our system is wrong.***

we tried multiple times to set the time zone to US but it did not show up the right time. After asking for help from professor, we knew that something wrong with the internet connection. We should not only click the WIFI sign on the right upper of our desktop, but also are we supposed to register ourselves on Cornell IT website. Our date was correct by registering on the website.

### ***Problem2: we cannot save and exit vim editor using ":wq"***



When we added PiTFT configure to “config.txt” file, we forget to use “sudo” to edit this file. Therefore, we cannot save the changes to that file.

***Problem3: The video played without audio even though we have plugged in the speaker***

When we played bigbuckbunny320p.mp4 on our pi, we don’t have audio. This is because we did not set the voice channel to headphone. After changing the channel, the video played smoothly with audio on as well.

***Problem4: Forget to add space to the command which needs to send to terminals using python***

As Linux is case sensitive, we should pay attention to adding space. When we use python to create our command for example “echo “pause” > video\_fifo”, we forgot to leave the space between echo and pause. As a result, the video did not pause after running the python file. After adding a space between them, the video reacted as we coded accurately.

Apart from these problems mention above, we also be aware of *some important practices* during this lab.

- Whenever we need to access a file in python code, we need to specify the location if this file is not in the same directory with python file.
- We should always place the electron device on the white mat in case of the electrical shock.
- After “shutdown” command, only when the green light did not flash, can we unplug the butterfly of our Pi.
- Don’t use number keypad on the right side.
- backup: Remember BACKUP!

## **Conclusions:**

In conclusion, every parts of our lab works smoothly. Despite of some problems we have mentioned above, we have figured all of those issues out with the help of professor and TA. Thanks so much for their kind instructions and patience.

we have learnt to load the Raspbian Kernel to our pi and initialize the external settings and external devices. We also have a knowledge of how to implement PiTFT and using monitor to help us to better control the PiTFT. Apart from that, we created a FIFO file which established the communication between our pi and PiTFT. Moreover, we are able to code python programs to control video play through FIFO with buttons on the PiTFT. The lab instruction file is very clear for us; we don’t see anything need to be changed. It was really an informative and resourceful Lab. Thanks again for the time and instructions from our professor and all the TAs.

## Code Appendix:

```
#FileName:fifo_test.py
#Lab1,Wednesday,group6
#Name(NetID):Suhui Yu(sy466);Yanan Liu(yl2248)

import subprocess

while True:
    cmd = raw_input("Enter a command:") #asking for user's input
    #create a command
    cmd1 = "echo " + "'" + cmd + "'" + " > /home/pi/video_fifo"
    #check the correctness of the command for debug if needed
    print(cmd1)
    #pass the command to terminal
    subprocess.check_output(cmd1, shell = True)

    #if enter exit, break the while loop and terminate the program
    if cmd == "exit":
        break
```

```

#FileName:one_button.py
#Lab1,Wednesday,group6
#Name(NetID):Suhui Yu(sy466);Yanan Liu(yl2248)

import RPi.GPIO as GPIO # give a RPi.GPIO a "nick name" GPIO
import time

#set GPIO to refers to the pin by Broadcom numberring
GPIO.setmode(GPIO.BCM)

#initilize pin 22 as GPIO input,set high as normal and
#low as button is pressed
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
while True:# continuous run the following code
    time.sleep(0.2)
    if (not GPIO.input(22)):# if low is detected on pin 22
        print(" ")
        print "Button Sw1/22 pressed..."

```

```

#FileName:four_button.py
#Lab1,Wednesday,group6
#Name(NetID):Suhui Yu(sy466);Yanan Liu(yl2248)

import RPi.GPIO as GPIO # give a RPi.GPIO a "nick name"GPIO
import time

GPIO.setmode(GPIO.BCM)
#set GPIO to refers to the pin by Broadcom numberring

#initilize four pins,set high as normal and low as button is pressed
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True: # continuous run the following code
    if (not GPIO.input(22)): #if low is detected on pin 22
        print(" ")
        print "Button 22 pressed.."
    elif (not GPIO.input(17)): #if low is detected on pin 17
        print(" ")
        print "Button 17 pressed.."
    elif (not GPIO.input(27)): #if low is detected on pin 27
        print(" ")
        print "Button 27 pressed.."
        break #break the while loop and terminate the program
    elif (not GPIO.input(23)): #if low is detected on pin 23
        print(" ")
        print "Button 23 pressed.."

```



```

#FileName:video_control.py
#Lab1,Wednesday,group6
#Name(NetID):Suhui Yu(sy466);Yanan Liu(yl2248)

import RPi.GPIO as GPIO # give a RPi.GPIO a "nick name"GPIO
import subprocess
import time

#set GPIO to refers to the pin by Broadcom numberring
GPIO.setmode(GPIO.BCM)

#initilize four pins,set high as normal and low as button is pressed
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True: #continuous run the following code
    time.sleep(0.2)
    if (not GPIO.input(22)): #if low is detected on pin 22
        print(" ")
        print "Button 22 pressed.."
        #create a command to fast forward the video by 10
        cmd = "echo seek 10 0 > /home/pi/video_fifo"
        #send the command to terminal
        subprocess.check_output(cmd, shell=True)
    elif (not GPIO.input(17)): #if low is detected on pin 22
        print(" ")
        print "Button 17 pressed.."
        cmd = "echo pause > /home/pi/video_fifo"
        #create a command to pause the video
        subprocess.check_output(cmd, shell=True)
    elif (not GPIO.input(27)):
        print(" ")
        print "Button 27 pressed.."
        cmd = "echo quit > /home/pi/video_fifo"
        #create a command to quit the video
        subprocess.check_output(cmd, shell=True)
        break #terminate the while loop
    elif (not GPIO.input(23)):
        print(" ")
        print "Button 23 pressed.."
        cmd = "echo seek -10 0 > /home/pi/video_fifo"

```

```
#create a command to rewind the video by 10
subprocess.check_output(cmd, shell=True)

#FileName:start_video
#Lab1,Wednesday,group6
#Name(NetID):Suhui Yu(sy466);Yanan Liu(yl2248)

#!/bin/bash

python video_control.py &

sudo  SDL_VIDEODRIVER=fbcon  SDL_FBDEV=/dev/fb1  mplayer  -input
file=/home/pi/video_fifo      -vo      sdl      -framedrop
/home/pi/bigbuckbunny320p.mp4
```