ECE 5725 Lab 1

Mingda Yang my432
Yi Chen yc2329

1. Introduction

The main purpose of Lab 1 is to get familiar with Raspberry Pi and its basic operations.

In the first week, we assembled the lab kit for ECE5725 and used the NOOBS install on the SD card, installed and configured the Raspbian Linux kernel according to lab handout. We also installed and configure drivers and software for the piTFT display as well as video and audio playback. At the end of the section, we displayed our work to TA.

In the second week, we played with FIFO(first in first out) command control, and designed Python scripts to control video played on the screen with physical buttons installed on the piTFT board. And finally we integrate the play video command and video control scripts in one bash script and demoed to TA.

2. Design and Testing

The lab include three main topics, first Raspberry-Pi setup and configuration of kernel, 2nd piTFT screen and driver setup, 3rd python scripts to control video.

2.1 Section 1: Assemble lab kit, install and configure Linux kernel

1. Install Raspberry-Pi in the R-Pi case and the breakout cable on the underside of the TFT. Plug the TFT into the RPi 40 pin GPIO connector. Insert miniSD Card to R-Pi and plug mouse and keyboard into USB ports, the display into HDMI port and Ethernet cable. Power up the kit.
2. Download the NOOBS and set up SD card with the NOOBS, here we finished download before class. Boot R-Pi to the Raspbian kernel. After successfully installed, reboot.

3. Configure the system such as password change, time zone change, followed by the lab instruction. Update and upgrade the Raspbian kernel and reboot.
4. Load and install some applications.
5. Backup SD card.

2.2 Section 2: Configure piTFT

1. Download and install the piTFT kernel.
2. Add the device tree overlay manually by changing the config file;
3. Modify modules.
4. Setup touchscreen, use AutoMatic calibration to calibrate the touchscreen.
5. Config console for piTFT installation.
6. Load sample video and setup piTFT for video playback.
7. Install Alsa audio driver, load sample audio and setup piTFT for audio playback.
8. Backup the SD card.

2.3 Section 3: Control playback by piTFT board using Python scripts

1. Explore mplayer and learn the types of video controls.
2. Control mplayer with a FIFO.
3. Using the python script ‘fifo_test.py’ to control mplayer with a FIFO.
4. Get input from a button connected to GPIO using the python script ‘one_buttons.py’. We picked button #22 as the ‘One button’.
5. Get input from four buttons connected to GPIO using the python script ‘four_buttons.py’. We add button #17, #23, #27 this time.

6. Control mplayer through a FIFO using a python script ‘video_control.py’. Button #17 is pausing, #22 is forwarding 10 seconds, #22 is rewinding 10 seconds, and #22 is quitting.
7. Create a bash script to launch mplayer and ‘video_control.py’ at same time using ‘start_video’.
8. Backup SD Card.

2.4 Issues

1. When we install R-Pi in the bottom of the case, it’s hard to assemble the edge to the plastic tabs, and once failed, it’s hard to get the board out of the case.
2. We failed to set the keyboard due to we choose a wrong time zone. And the consequent result is we cannot quit the ‘startx’ desktop window. TA helped us fixing this problem.
3. There are several slight difference in the website link compare to the lab instruction. At first we were following the web link, and met several weird problem. After carefully review the lab instruction, we solved the problems.
4. In the make FIFO file part, the file address given in the web link is a little bit confusing, we figured the problem out with TA’s help.
5. The first python script is a little bit tricky, cause both of us haven’t us Python for a while, after reviewing some basic example and typo tutorial online, we finished the rest part easily.

3. Result

3.1 Ras-Pi and piTFT setup

We successfully setup the device and choose the right configuration, and demoed our device, image and sound to TAs. We've accomplished the following tasks.

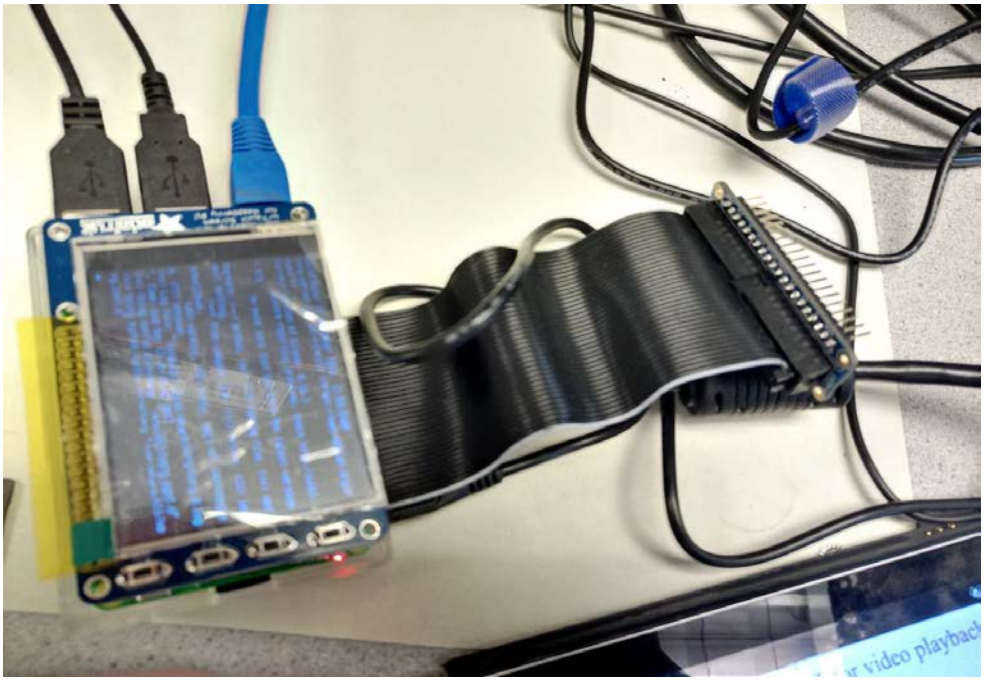

Figure 1: Ras-Pi assembled with power on testing

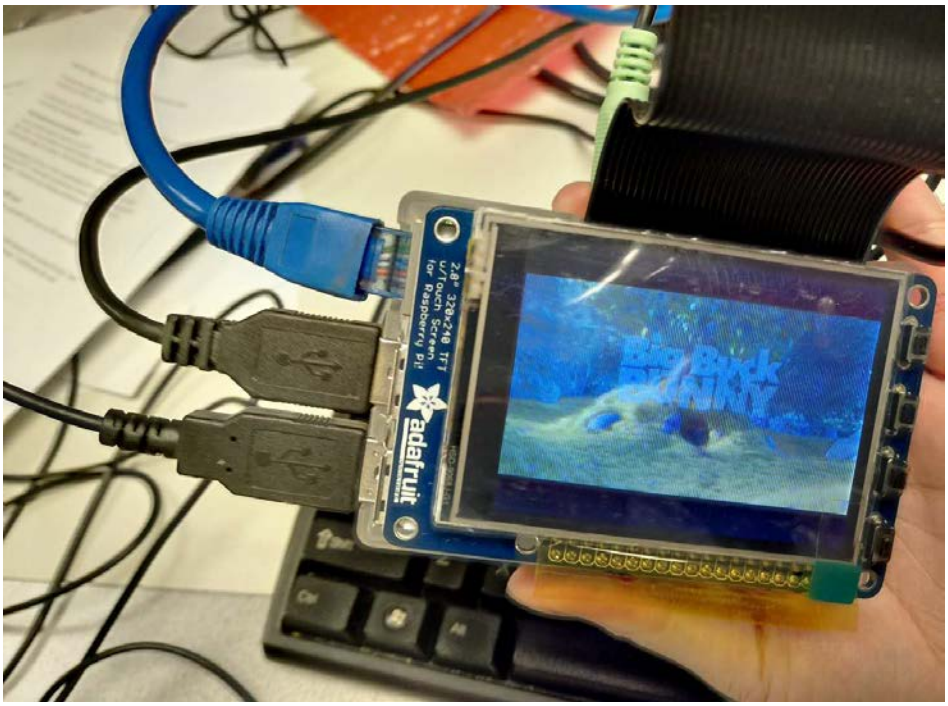

Figure 2:video on piTFT testing

Figure 3:audio testing with demo to TA

3.2 Video control

We first successfully tested the FIFO command test in Linux system. Controlled video by sending FIFO command in another user terminal. Then we developed several python scripts followed by lab instruction. Finally integrate the play video command and the video controll script together in a bash file. We've accomplished the following tasks:

1. Control video with FIFO command in Linux.
2. One button press test, report the button number on screen when that button is pressed.
3. Four button press test, same as one button test, plus the quit program function when the edge button is pressed. In our code, 27 is the 'quit' button.
4. Video control by button test, assign each button a unique function corresponding to the lab instruction (pause, forward, backward, and quit). When each button is pressed, video behaved as our expectation.
5. Integrate video play command and video control script in bash file, finally we only need to run one file to finish all the functions we tried above.

Button press and video control tests are demoed to TA

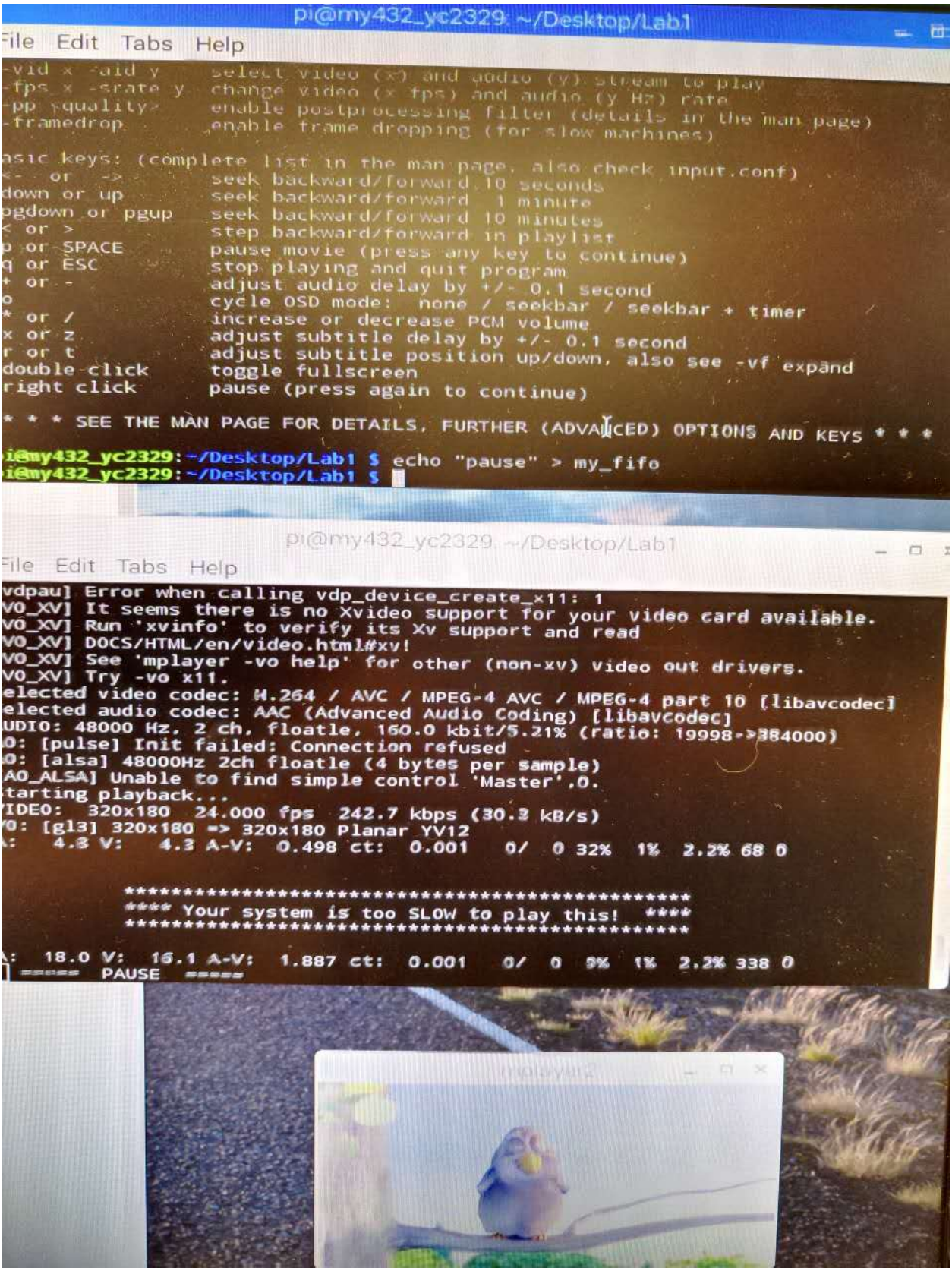

Figure 4: FIFO pause test

4. Conclusion

In lab 1, we learned the safety procedures of using Ras-Pi, either on how to contain the device safe and our own safety. And we successfully assembled and setup the device with touch screen. Finally we finished our first script to control the Ras-Pi device by Python. During the lab, we learned how to set up the I/O port and how to get response from Ras-Pi. Those will be really useful in our future study and design with Ras-Pi

5. Code Appendix

5.1 fifo_test.py

```
# Lab 1 09/13/2017
# Yi Chen yc2329
# Mingda Yang my432

import subprocess

while True:
        input_cmd = raw_input("Please enter your command: ")
        # let user enter their commend and stored in input_cmd
        cmd = 'echo "' + input_cmd + '" > my_fifo'
        # Form the correct typo for fifo commend, pass user input as an arbitrary component
        print subprocess.check_output(cmd,shell=True)
        # Run the comment with subprocess, first check whether comment is exist  and correct, then if yes return, else raise error flag
        if input_cmd == 'quit':
        # quit the program when met 'quit'
                break
```

5.2 One_button.py

```
# Lab 1 09/13/2017
# Yi Chen yc2329
```

```
# Mingda Yang my432

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)   # Set for broadcom numbering not board numbers...
# setup piTFT buttons
#                   V need this so that button doesn't 'float'!
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 22 as input GPIO pin, as pull up network)

while True:
   time.sleep(0.2)  # Without sleep, no screen output!
   if ( not GPIO.input(22) ):                # Check GPIO 22, if it is pressed, report
      print (" ")
      print "Button 22 pressed...."
```

5.3 four_buttons.py

```
# Lab 1 09/13/2017
# Yi Chen yc2329
# Mingda Yang my432

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)   # Set for broadcom numbering not board numbers...
# setup piTFT buttons
#                   V need this so that button doesn't 'float'!
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 17 as input GPIO pin, as pull up network)
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 22 as input GPIO pin, as pull up network)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)
 # set up broadcom number 23 as input GPIO pin, as pull up network)
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 27 as input GPIO pin, as pull up network)

while True:
   time.sleep(0.2)                        # Without sleep, no screen output!
   if ( not GPIO.input(17) ):
```

```
            # Check GPIO 17, if it is pressed, report
            print (" ")
            print "Button 17 pressed...."

    elif( not GPIO.input(22) ):
            # Check GPIO 22, if it is pressed, report
            print (" ")
            print "Button 22 pressed...."
    elif( not GPIO.input(23) ):
            # Check GPIO 23, if it is pressed, report
            print (" ")
            print "Button 23 pressed...."
    elif( not GPIO.input(27) ):
            # Check GPIO 27, if it is pressed, report, then quit the program
            print (" ")
            print "Button 27 pressed...."
            break
```

5.4 video_control.py

```
# Lab 1 09/13/2017
# Yi Chen yc2329
# Mingda Yang my432

import RPi.GPIO as GPIO
import time
import subprocess

GPIO.setmode(GPIO.BCM)   # Set for broadcom numbering not board numbers...
# setup piTFT buttons
#                       V need this so that button doesn't 'float'!
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 17 as input GPIO pin, as pull up network)
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 22 as input GPIO pin, as pull up network)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 23 as input GPIO pin, as pull up network)
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# set up broadcom number 27 as input GPIO pin, as pull up network)

while True:
    time.sleep(0.2)  # Without sleep, no screen output!
```

```
    if ( not GPIO.input(17) ):
        # Check whether 17 is pressed, if yes, send cmd echo pause to my_fifo
        cmd = 'echo "pause" > my_fifo'
                    print subprocess.check_output(cmd,shell=True)
                    # Run the commend with subprocess, first check whether commend is exist  and correct,
            then if yes return, else raise error flag
    elif( not GPIO.input(22) ):
    # Check whether 17 is pressed, if yes, send cmd echo seek +10s to my_fifo
                    cmd = 'echo "seek 10" > my_fifo'
                    print subprocess.check_output(cmd,shell=True)
                    # Run the commend with subprocess, first check whether commend is exist  and correct,
            then if yes return, else raise error flag
    elif( not GPIO.input(23) ):
    # Check whether 17 is pressed, if yes, send cmd echo seek -10s to my_fifo
                    cmd = 'echo "seek -10" > my_fifo'
                    print subprocess.check_output(cmd,shell=True)
                    # Run the commend with subprocess, first check whether commend is exist  and correct,
            then if yes return, else raise error flag
    elif( not GPIO.input(27) ):
    # Check whether 17 is pressed, if yes, send cmd echo quit to my_fifo
                    cmd = 'echo "quit" > my_fifo'
                    print subprocess.check_output(cmd,shell=True)
                    # Run the commend with subprocess, first check whether commend is exist  and correct,
            then if yes return, else raise error flag
                    break                                   # Then quit the program.
```

5.4 start_video

```
# Lab 1 09/13/2017
# Yi Chen yc2329
# Mingda Yang my432

#!/bin/bash
#
#
#

echo "play video &"
sudo SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop -input
file=my_fifo bigbuckbunny320p.mp4 &  #run the video on ras pi screen

echo "Running video contrl" #run the video control program
```

python video_control.py