

Lab 1

Thursday section

Due Oct 1, 2020

Jiaqi Sun (js3599)
Pranav Gupta (ppg22)

Introduction

The first lab was intended for setting up the Raspberry Pi, the piTFT, and video and audio playback on the piTFT. We also developed a few Python scripts to control the video playback on piTFT using buttons. In the first week, We inserted the new SD card loaded with the Raspbian Buster OS and installed the Raspbian Kernel using Raspi-config. After the kernel was installed, we installed the piTFT screen and configured the touchscreen operation and mplayer application. In the second week, we wrote a fifo and Python scripts to control mplayer with a FIFO, to detect input from piTFT buttons through GPIO pins, and to control mplayer by pressing piTFT buttons. We also created a bash script to launch mplayer and python scripts.

Design and Testing

Include design steps involved in the lab and describe your progress in each of the lab sections (for example, in Lab1, describe how the Linux kernel was installed and modified to include the piTFT). Please include any issues you experienced as well as smooth progress in the various sections of the lab. Describe testing you performed to confirm that steps of the lab worked as planned.

Part 1:

In the first part of the lab, we assembled the lab 1 kit and installed the Raspbian Linux kernel. The R-Pi was first installed in the case and the breakout cable was installed on the bottom of the piTFT. The white stripe on the breakout cable was right under the piTFT buttons. We then connected the piTFT to R-Pi GPIO pins and the assembly part was finished. The R-Pi and piTFT was powered with a 3.0 amp power adapter and connected to the laptop with an ethernet cable.

We used a terminal emulator to access the R-Pi on our laptop. Command “ssh pi@host_name.local” was used to log into the R-Pi, “sudo shutdown -h now” was used to power off the R-Pi and “sudo reboot” was used to restart the OS. We installed the Raspbian kernel using the command “Raspi-config”. We changed the locale to “en-us. UTF8 UTF8” and the timezone to “New York, America”. We also expanded the file system and forced sound over the 3.5 mm jack on the R-Pi. After all the changes were made, we updated the raspi--config tool to the latest version and rebooted the system. Command “uname -a” could be used to check the updated kernel version.

```
pi@js3599:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]
Get:3 http://archive.raspberrypi.org/debian buster/main armhf Packages [331 kB]
Get:4 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
Get:5 http://raspbian.raspberrypi.org/raspbian buster/contrib armhf Packages [5.7 kB]
Fetched 13.4 MB in 18s (746 kB/s)
Reading package lists... Done
pi@js3599:~ $ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:2 http://archive.raspberrypi.org/debian buster InRelease
Reading package lists... Done
pi@js3599:~ $ uname -a
Linux js3599 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l GNU/Linux
pi@js3599:~ $
```

Figure 1: Check the updated kernel version

Because both of us used the ethernet cables and laptops to access the R-Pi, we used the VNC server and viewer to access the R-Pi desktop on our laptops. The VNC server was enabled in raspi-config and the VNC viewer was installed on our laptops and connected to the server.

Part 2

In the second part of the lab, we installed the piTFT and configured the mplayer application. We installed the python packages on the Pi and tested the piTFT with evtest and libts-bin application. We also installed evdev to read and write inputs on Linux. Whereafter, we added the piTFT info to the config.txt to configure the speed setting and rotate setting, etc. The piTFT started out white and then turned to black after the changes were made. The next step was to test the touchscreen operation on the piTFT. We ran “sudo dmesg” to check if the modules were installed correctly and added a udev rule to support the use of the touchscreen. After the rule files were created, we unload the touchscreen driver and reload it back. The touchscreen was associated with event 0 after we checked the operation with command “ls -l /dev/input/touchscreen”. We used evtest to check the touchscreen operation. Every time we taped the screen, there was a list of information showing up in the terminal. Figure 2 shows that the touchscreen operation was running correctly.


```

pi@js3599:~ $ sudo evtest /dev/input/touchscreen
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "stmpe-ts"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value      0
      Min        0
      Max       4095
    Event code 1 (ABS_Y)
      Value      0
      Min        0
      Max       4095
    Event code 24 (ABS_PRESSURE)
      Value      0
      Min        0
      Max       255
Properties:
Testing ... (interrupt to exit)
Event: time 1601486563.946961, type 3 (EV_ABS), code 0 (ABS_X), value 2262
Event: time 1601486563.946961, type 3 (EV_ABS), code 1 (ABS_Y), value 1815
Event: time 1601486563.946961, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 1
0
Event: time 1601486563.946961, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1601486563.946961, ----- SYN_REPORT -----
Event: time 1601486563.955110, type 3 (EV_ABS), code 0 (ABS_X), value 2270
Event: time 1601486563.955110, type 3 (EV_ABS), code 1 (ABS_Y), value 1807
Event: time 1601486563.955110, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 8
Event: time 1601486563.955110, ----- SYN_REPORT -----
Event: time 1601486563.963233, type 3 (EV_ABS), code 0 (ABS_X), value 2266

```

Figure 2: Data associated with touchscreen event

We then set the calibration of the piTFT by adding the line "35 -5782 21364572 4221 35 -1006432 65536" in the file /etc/pointercal. To start the console window on piTFT, we first included the "fbcon=map:10 fbcon=font:VGA8x8" in the file /boot/cmdline.txt. The font and font size were also changed to display the console. Command "sdo sh -c 'TERM=linux setterm -blank 0 >/dev/tty0'" were used to turn off piTFT blanking. We rebooted the system after the changes were made and the linux console showed up on the piTFT screen. The last step was to install the mplayer application and play a sample video on the piTFT. We used the command "sudo apt-get install mplayer" to get the mplayer application. The command "sudo SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop video.mp4" ran the video on the piTFT. We could use "amixer -q -M sset Heaphone 100%" to adjust the audio volume. Figure 3 shows that the volume was changed to 50% and the video was played.



```

.: MPlayer : F = Fullscreen/Win...
=====
lavcodec audio decoders
/5.21% (ratio: 19998->384000)
(FFmpeg AAC (MPEG-2/MPEG-4 Audio))
=====
d
r sample)
irect movie aspect.
0/ 0 3% 80% 0.9% 79 0

Exiting... (Quit)
pi@js3599:~ $ amixer -q -M sset Headphone 50%
pi@js3599:~ $ amixer -M sget Headphone
Simple mixer control 'Headphone',0
  Capabilities: pvolume pvolume-joined pswitch pswitch-joined
  Playback channels: Mono
  Limits: Playback -10239 - 400
  Mono: Playback -1406 [50%] [-14.06dB] [on]
pi@js3599:~ $ SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framed
p /home/pi/lab1_files_f20/bigbuckbunny320p.mp4
MPlayer 1.3.0 (Debian), built with gcc-8 (C) 2000-2016 MPlayer Team
do_connect: could not connect to socket
connect: No such file or directory
Failed to open LIRC support. You will not be able to use your remote control.

Playing /home/pi/lab1_files_f20/bigbuckbunny320p.mp4.
libavformat version 58.20.100 (external)
libavformat file format detected.
[mov,mp4,m4a,3gp,3g2,mj2 @ 0xb6813308]Protocol name not provided, cannot determine if input is local or a network protocol, buffers and access patterns cannot be configured optimally without knowing the protocol
[lavf] stream 0: video (h264), -vid 0
```

Figure 3: Adjust the volume and play the video

Part 3

In the last part, we wrote a few python scripts to control the video playback on the piTFT by pressing buttons. To control the mplayer with fifo, we first created a fifo file and ran the mplayer to play the video with command "mplayer -input file=/home/pi/test_fifo video.mp4". Then we opened a new process and echoed "pause" to the fifo file. The "pause" command would be readed by the mplayer and the video on piTFT would be paused. In step 3, we wrote a python script to send commands to the fifo file. Every time the command the keyboard is valid, the python routine sent the command to the fifo file and thus controled the mplayer. It could also recognize a command to quit the program. Figure 4 shows that the video paused after we entered the command and the program could also recognize an invalid command.


```

I > Your computer drives are accessible through the /drives path
28.149:0.0
LAY is automatically forwarded
by a special symbol (✓ or ✕)

. The Professional edition
for your company: you can add
ur welcome message and generate
or a portable executable.
velop the plugins you need.
term.mobatek.net/download.html

[Administrator.yyxi] > ssh pi@js3599.local
Linux js3599 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Sep 30 13:08:35 2020 from fe80::c087:c647:340c:8095%eth0
pi@js3599:~ $ cd pi
-bash: cd: pi: No such file or directory
pi@js3599:~ $ ls
Desktop  Downloads  Music      Public     Videos    test_fifo
Documents  MagPi      Pictures   Templates  lab1_files_f20
pi@js3599:~ $ cd lab1_files_f20/
pi@js3599:~/lab1_files_f20 $ python3 fifo_test.py
Enter the command: pause
Enter the command: k
Command is not valid!
Enter the command: █

```

Figure 4: Fifo_test.py testing

In step 4 and 5, we tried to find the GPIO pins that were connected to the buttons and outputted a notification when the buttons were pressed. We used a while loop to implement the function. Figure 5 shows the testing of one_button.py and figure 6 shows the testing of four_buttons.py.

```

pi@js3599:~/lab1_files_f20 $ python3 one_button.py

Button 17 pressed...

Button 17 pressed...

Button 17 pressed...

Button 17 pressed...
█

```

Figure 5: One_button testing

```
python3: can't open file 'four_buttons.py': [Errno 2] No such file or directory
pi@js3599:~/lab1_files_f20 $ python3 four_buttons.py

Button 17 pressed...

Button 17 pressed...

Button 22 pressed...

Button 23 pressed...

Button 23 pressed...

Button 22 pressed...

Button 17 pressed...

Button 22 pressed...

Button 27 pressed...
pi@js3599:~/lab1_files_f20 $
```

Figure 6: Four_buttons.py testing

First, all the values of the buttons were initialized as high. Then we suspended the execution for 0.2 sec in case that one button press was monitored repetitively. If the value of the button switched to low, we knew that the button was pressed and printed the message. One of the buttons was used to quit the program. In step 6, we created a python program to control the mplayer to pause, fast forward 10 sec, rewind 10 sec, quit. Based on the previous python program, we associated each button with different commands. Every time a button was pressed and the GPIO pins detected that, we sent different commands to fifo and further control the mplayer. Figure 7 shows that we could fast forward the video for 10 sec when pressed button 22

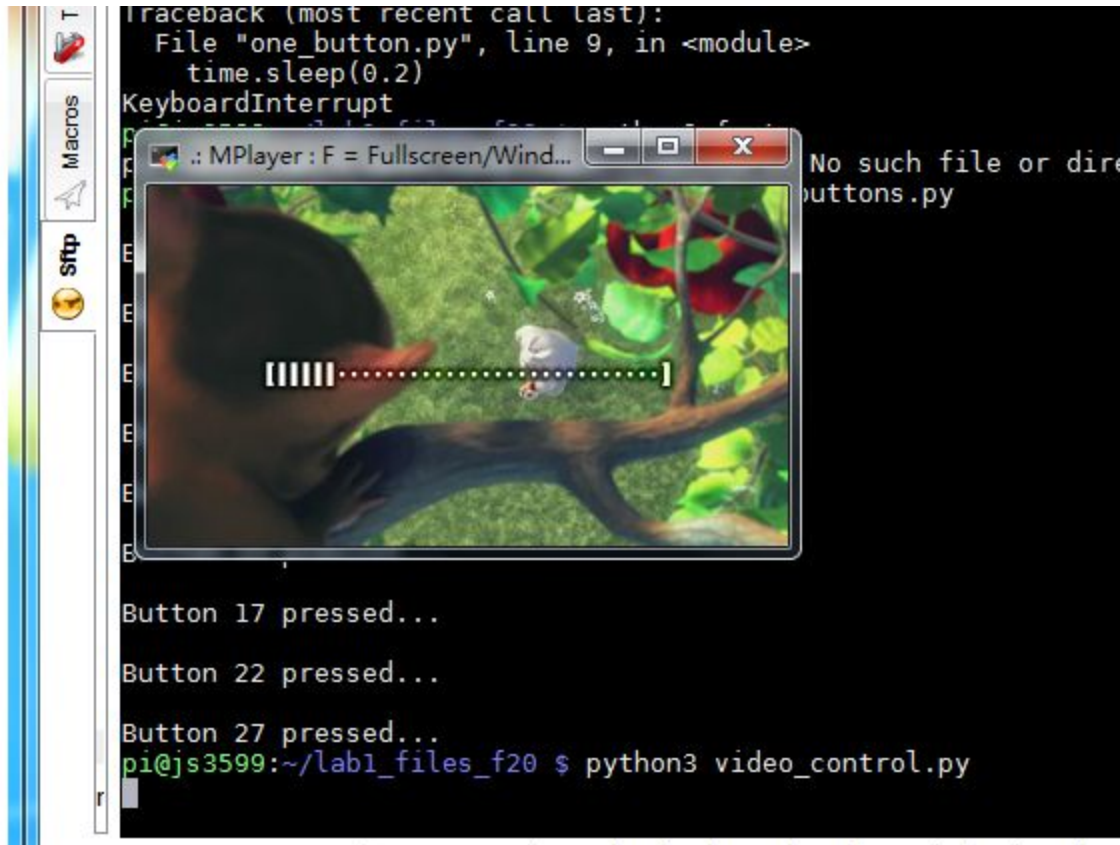


Figure 7: Fast forward 10 sec

The last step was to create a bash script to launch mplayer and the python program in step 6. We started the python program first in the background and then launched the mplayer. It seemed that “sudo” could impact the video and audio playback. When I typed in the command without “sudo”, the video showed up on the mplayer window and there was audio on the headphone jack, but if I typed in the command with “sudo”, the video was played on the piTFT but the audio disappeared. All the python codes worked in both situations, I played the video on the mplayer window so that it was easy to present the result.

Results

Results should also be discussed in this section.

Did everything perform as planned?

Did your team meet the goals outlined in the lab description?

Did you complete each week’s checkout on time?

Most of the time the lab performed as planned and we outlined the goals in the lab description. One problem we encountered was setting up the VNC viewer. At first, we enabled the VNC server and changed the VNC server field to wifi address, but the viewer still didn’t display the R-Pi desktop. The problem was solved by using raspi-config to make more adjustments. In boot options, we changed to “boot to desktop, no login” and in the advanced options we changed the

resolution to the highest resolution. The solution was from one piazza answer. The second problem we encountered was to display the video on piTFT through VNC viewer, the command “sudo SDL_VIDEODRIVER=fbcon SDL_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop video.mp4” only created a small window to display the video on the VNC instead of on the piTFT. However, if we typed the command in the terminal, it would start the video on the piTFT. The problem hasn't been solved. There were many typos or funny characters if we used copy_and_paste to enter the commands, therefore, typing in commands was preferred. We completed the checkout on time as well.

Conclusions

What worked and what didn't work during the lab? Were there any issues you experienced during the lab? Are there any improvements you would suggest? Were there any items that could be clearer in the lab description?

In conclusion, the lab went pretty smoothly. There were some issues we discussed above, some of them were solved but some of them were not. We had a brief understanding of installing the Linux kernel, piTFT, video and audio playback. We also started to create python scripts and bash script to control the video the piTFT physically. The lab manual was long but it was straightforward to us.

Code:

Each code module should include a header with the names and netids of each team member, Lab assignment number, and date. Clear comments in the code are essential. There are two methods for including code in the report: