**ECE 5725 Lab Report**

**Lab 1**

Chenghui Li(cl2228)

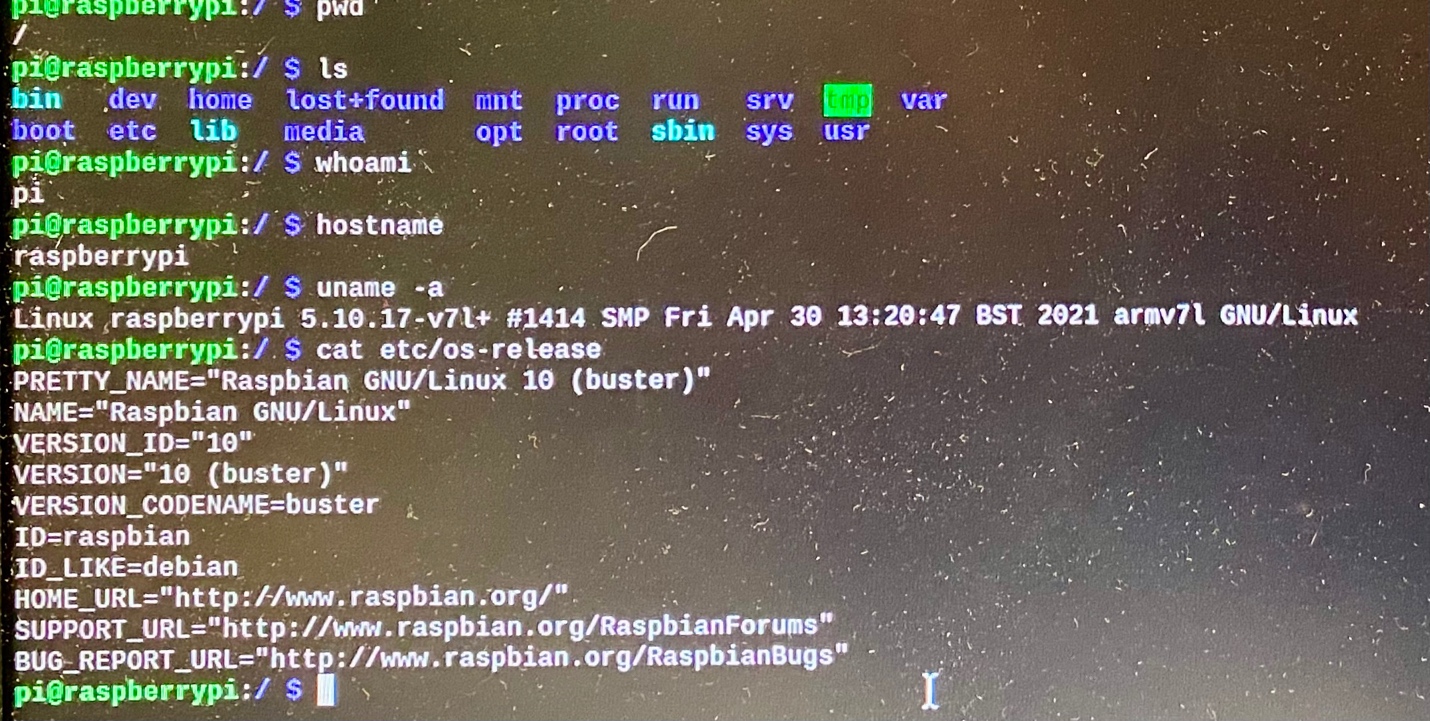
Hehong Li(hl778)

Lab Section: Thursday

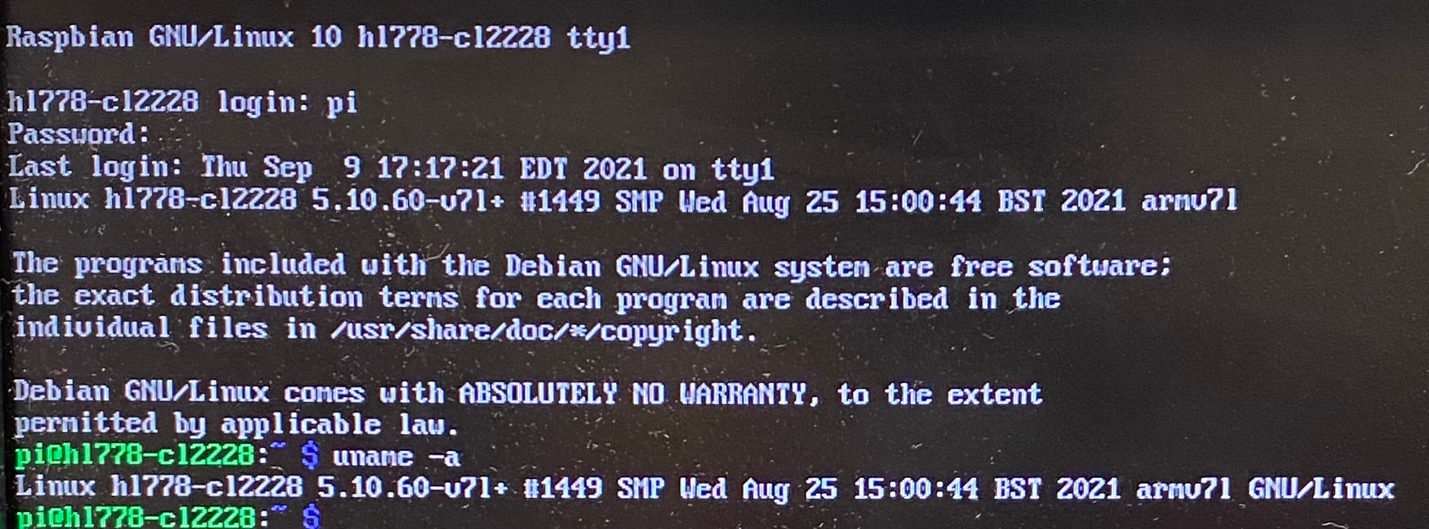
Date: 09/09 and 09/16

**Design and Testing**

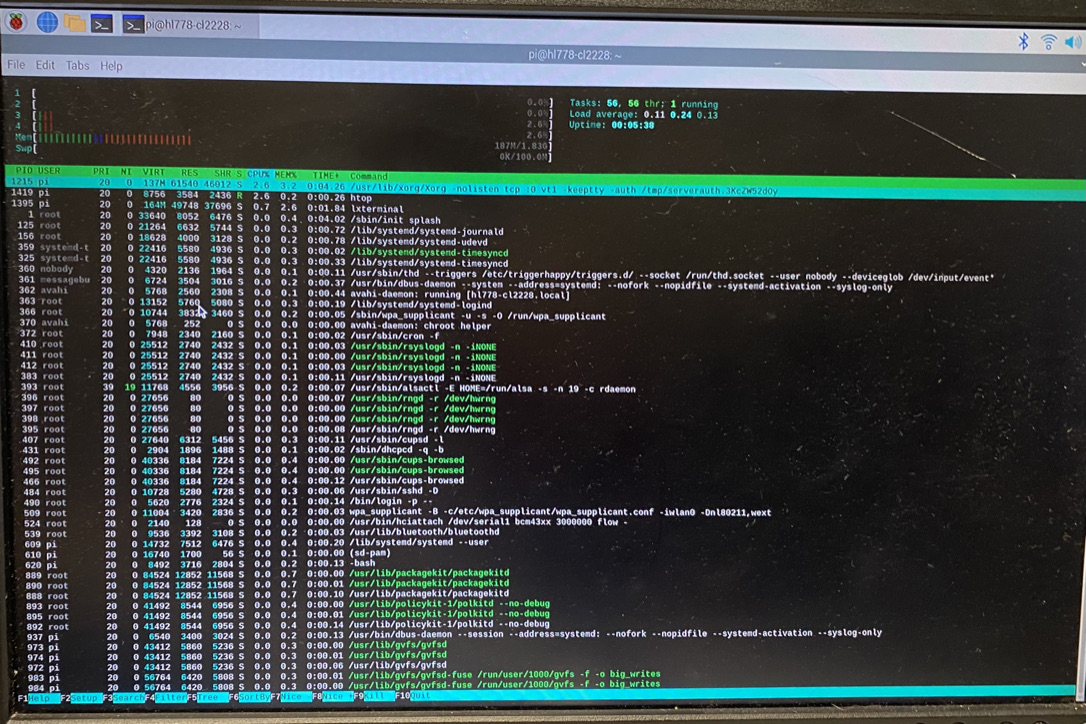
We first installed and configured the Raspberry Pi, after power-up, we used basic commands like “whoami”, “hostname”, “uname -a” to get the information of our raspberry Pi:



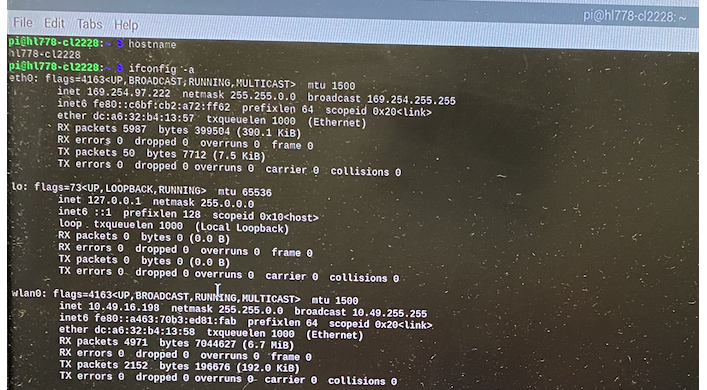
We can see that the Linux version is not up-to-date, and we have not changed the host name yet. After we updated it and change the user information, we can get:



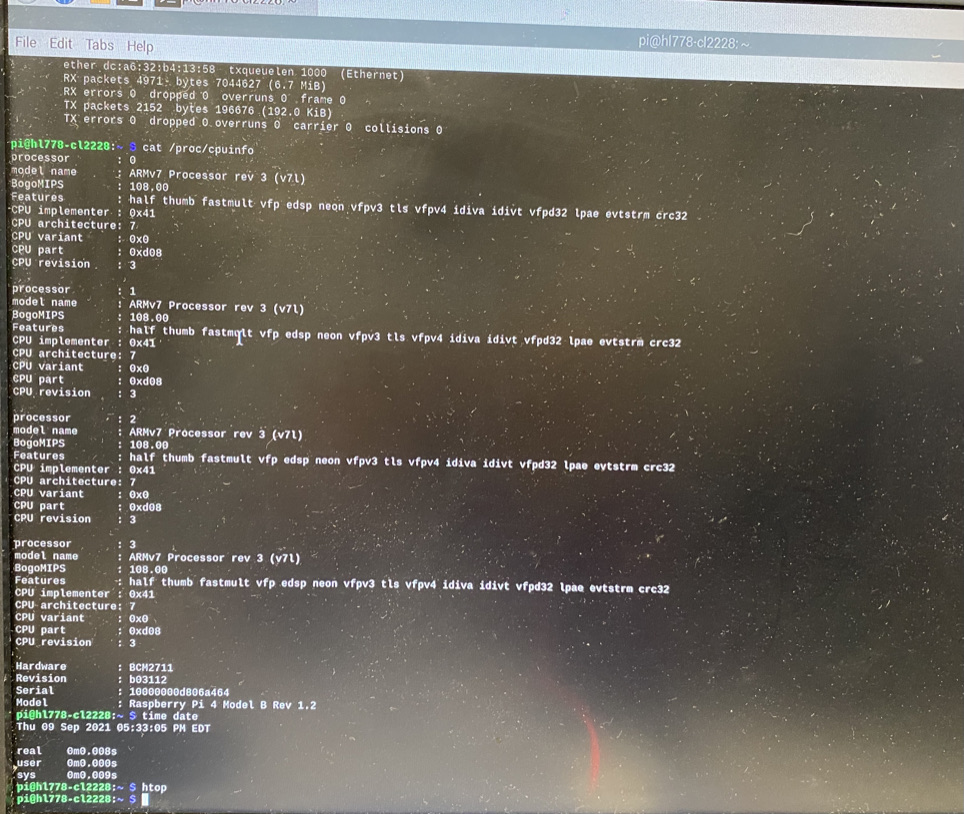
Which shows that the host name is “hl778-cl2228” the netids of our team, and the Linux version is up-to-date now, then we used the “htop” command to show the running processes:



The next step was that we needed to connect the pi using ssh from our laptops, so we used the command “ifconfig -a” to find out the IP address of the raspberry pi:



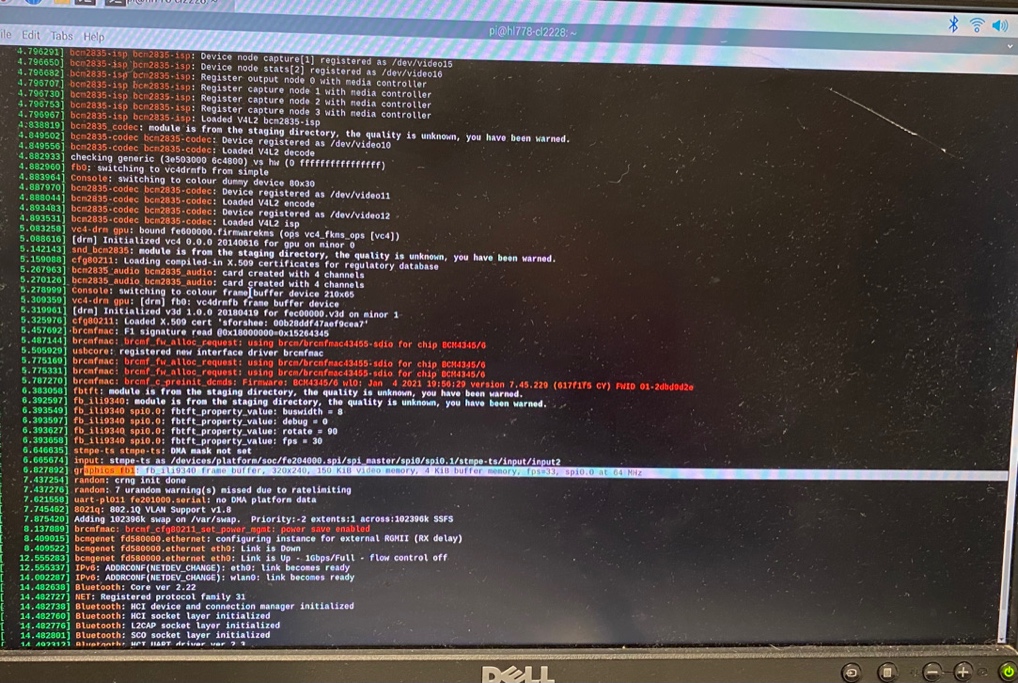
The WLAN IP address of our raspberry pi is 10.49.16.198, and we successfully got access to the pi using ssh on my laptop with “ssh @pi10.49.16.198”. Next, we checked the CPU information of the pi by using “cat /proc/cpuinfo” :

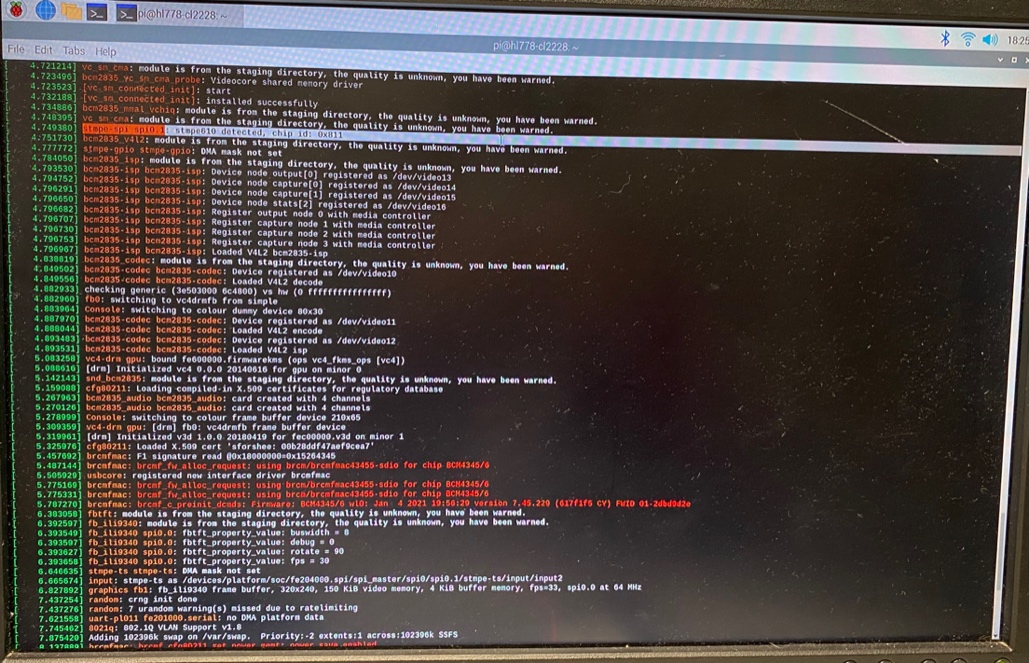


From the photo we can see that the number of cores is four. After this, we used about half an hour to back up the pi. First, we used “sudo shutdown -h now” to shut down the pi, but we saw that the piTFT was still on, we asked TAs and found out as long as the green light by the pi is off, we can pull off the power.

The system set up was done, we proceeded to set up the piTFT. First, we used “sudo apt-get install -y bc fbi git python-pip python-smbus python0spidev evtest libts-bin” and “sudo pip install evdev” to install the necessary softwares.

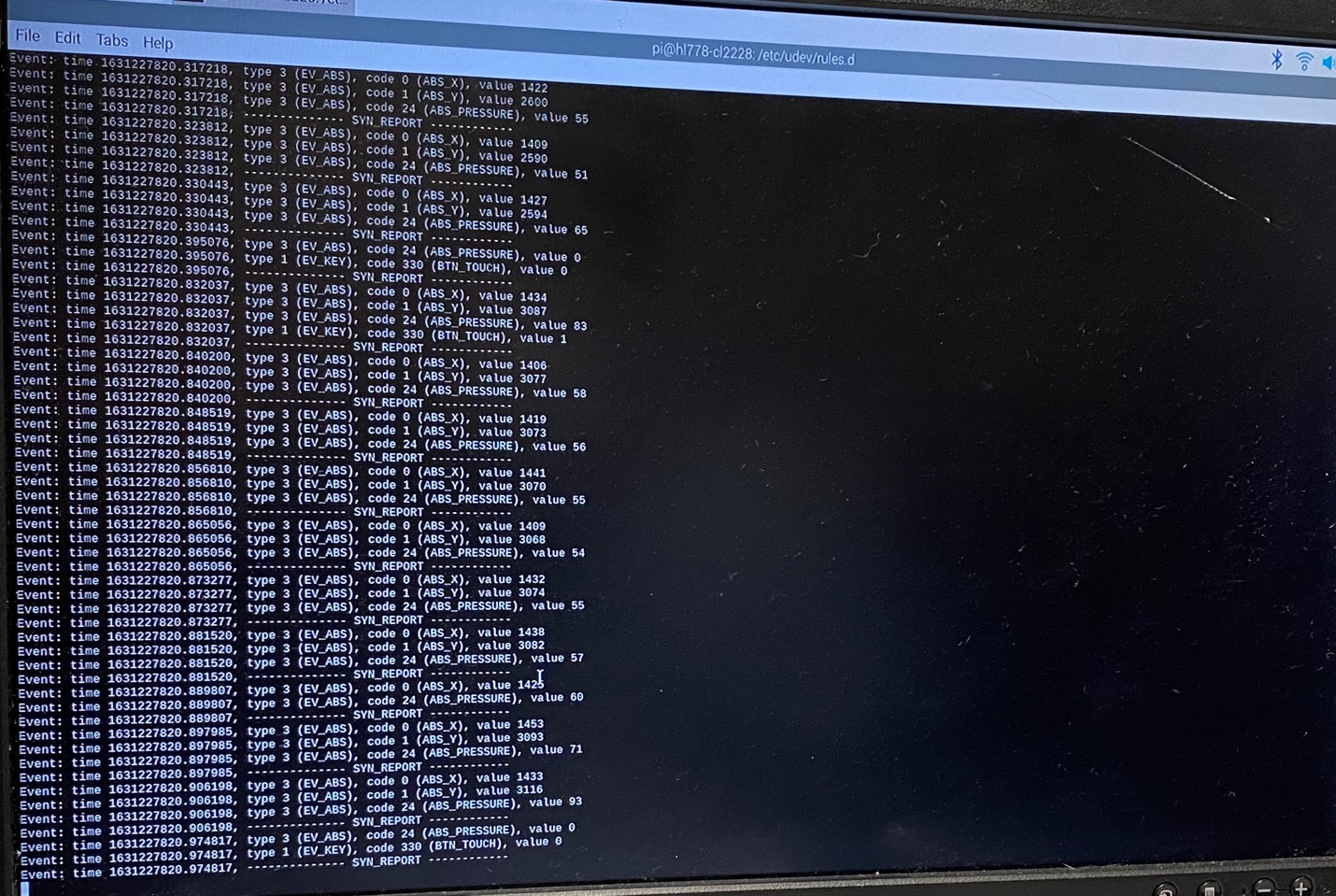
After adding piTFT info to config.txt, we used “dmesg” to check whether the configuration is right:





We checked the link started with “stmpe-spi” and “graphics fb1” and made sure they are all set. The screen size is 320x240, 4KB buffer, fps=33, 64MHz.

After set up the piTFT, we added udev rules to “/etc/udev/rules.d/95-stmpe.rules”, “/etc/udev/rules.d/95-touchmouse.rules”, and “/etc/udev/rules.d/95-ftcaptouch.rules” to catch events of the piTFT. We made a typo mistake here by typing “touchmose” rather than “touchmouse”, thanks to TAs helped us solve it. After that, we rebooted the Pi and tested whether the piTFT runs correctly. We detected events caused by touching the screen:



We unloaded the driver and reloaded again according to the instruction, after a serious of configurations were set right, we finally reached the step to play the video by the command “sudo SDL\_VIDEOFRIVER=fbcon SDL\_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop bigbuckbunny32p.mp4”. Again, we had a typo here by typing “sd1” rather than “sdl”, a bug that hard to detect. With help from TAs, we finally found this bug and ran the video successfully.

**Week 2:**

First, we created a fifo file at “/home/pi/0916/video\_fifo”, and started the video with “mplayer -input file=/home/pi/0916/video\_fifo bigbuckbunny320p.mp4”. We tested the fifo by using another ssh connect on my laptop to “echo ”pause” > /home/pi/0916/video\_fifo”, the video paused successfully, meant that the fifo worked!

Next, we created a python script called “fifo\_test.py” which reads the input and if the input is a single “q”, the program sends a command to the fifo to quit the mplayer. And if the input to python is “p”, the program sends a command to the fifo to pause the mplayer.

After testing the python, we started to work with GPIO, the first stop was to write a python program to detect whether a button is pressed. We created a file called “on\_button.py” to detect whether the GPIO 17 works, which shows no error in testing.

Next, we moved forward to test the four buttons using a python program called “four\_buttons.py”, which initiated the GPIO and using a forever loop to detect if a button is pressed. We made a small mistake here that let the while loop contain the GPIO setup statements, which works but it is not a right choice.

After debugging our “four\_buttons.py”, we started to create a “video\_control.py” to control the video by four buttons on the piTFT. Similarly, we used a while loop that runs forever until a break. It keeps detecting our input, if we press the 17 button, the program excutes “echo ”pause” > /home/pi/0916/video\_fifo” to pause the video; if we press the 22 button, the program excutes “echo ”seek 10” > /home/pi/0916/video\_fifo” to fast forward 10 seconds; ; if we press the 23 button, the program excutes “echo ”seek -10” > /home/pi/0916/video\_fifo” to rewind 10 seconds; and if we press the 23 button, the program excutes “echo ”quit” > /home/pi/0916/video\_fifo” to quit the mplayer as well as break the while loop of python, leading the end of the program.

The python program went well and we did not meet any bug, so me moved to the last step—using a bash shell to run the video in the front and run the python program in the back so that we can use four buttons on the piTFT to control the video with the python program running in the backend. We put the program in the backed by “python /home/pi/0916/video\_control.py &”. We wrapped this statement and the mplayer statement to the bash shell.

Thank god we did not make any bug and these steps went great magically, so we finish the whole lab!

**Conclusions**

**Week 1:**

We actually went quiet well at the first-half part of the task, including installing and configuring the Raspberry Pi. We did not meet any problems in connecting the Raspberry Pi to Cornell WiFi, this is lucky because I heard from my friend Jinyang who worked in Lab Wednesday that they had some daunting troubles about the WiFi which took them near an hour.

But after having set up the Linux system and installed the necessary applications, we have trouble in configuring the PiTFT, mainly because typo, especially when it came the the command “*sudo SDL\_VIDEOFRIVER=fbcon SDL\_FBDEV=/dev/fb1 mplayer -vo sdl -framedrop bigbuckbunny32p.mp4*”. We spent at least half an hour with troubles regarding this command, and professor also talked about my silly mistakes at the lecture next day, this is an important lesson that I really need to get rid of my bad habit of typo. Thanks much to my partner Hehong for not killing me for my typos. I also learnt from the week one that it would be much more convenient and faster to use SSH on my laptop than use the keyboard and the monitor at the lab. I improved by doing this at the week two and it turned out to be super efficient!

Although backup is important, it really took a long time to back up the Raspberry Pi during the lab time (we used about half an hour on it). This is the main reason that we nearly missed the checkout time that day, but the good side is that I saw a wonderful night view at Cornell that day and it was the first time I was still at Cornell after sum goes down.

**Week 2:**

Week 2 is remarkable success as we finished the lab ahead of the schedule by more than an hour! In week 2, the main knowledge I learned is how to use script, fifo, and python to control the mplayer. Since I have many experience in writing scripts in Linux and python programs, this was not a hard job for me so we went quite fast! Fifo is magical and the reference on Canvas is useful! Thanks to great help from my partner, professor and TAs!