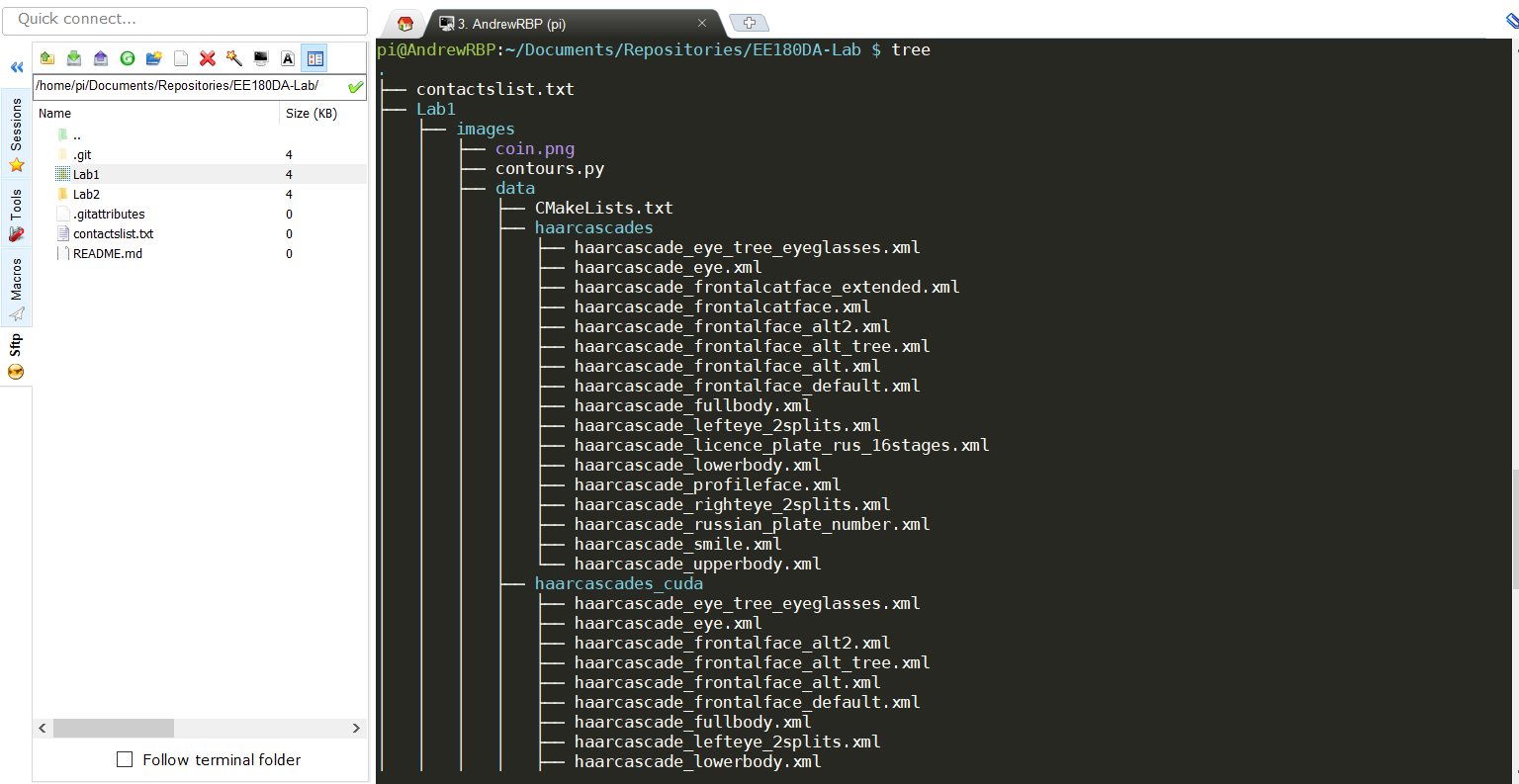
Issues experienced during the lab:

Originally, I was experiencing issues connecting to my RBP Zero W over Serial SSH. I wasn’t really able to do this until I was able to connect to it over wi-fi at home by supplying it my router’s SSID and password. That fixed my connection problem and I was able to install the other modules associated with this lab.

Additionally, open-cv had to be installed using sudo apt install instead of pip install, since the RBP Zero W has an architecture (ARMv6 I believe?) that pip does not have a wheel for. Doing sudo apt install installed a binary, which means no 9-hour long compile times which worked great.

Also, my OTG USB Hub for the RBP Zero W didn’t come in until Thursday, 10/10.

1. I git cloned my repo onto my raspberry pi. A ‘tree’ command resulted in the following output:



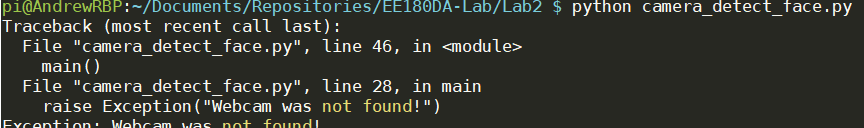
1. Here is a simple python script that prints to the terminal of the console. The implementation is very simple, it simply checks if python is running this file as \_\_main\_\_ and prints to the console this:



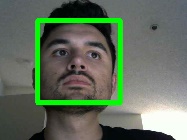
1. I modified the face-tracker implemented in my first lab to save the image to a file instead of using OpenCV to display to a GUI. Now it saves the file, and I can extract it using my SFTP client. For this, I’m using MobaXTerm on windows. On my computer, this face-tracker worked perfectly and saved to a file.

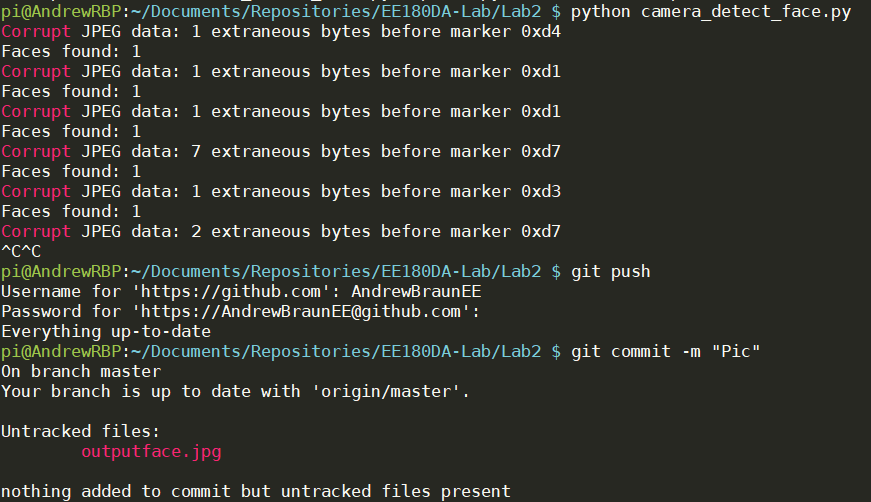


Without my OTG Hub, the python script outputs this since the webcam isn’t attached:



But after receiving my OTG Hub on Thursday Night, I was successfully able to reproduce my computer’s results.





Note that while on my computer, this Corrupt JPEG data error never showed up, for some reason, on my RBP setup it did. This may be due to the fact that either the RBP or OpenCV version I’m using is adding extra bytes to the JPEG, or extra bytes are being sent by my OTG USB Hub that I got off of Amazon. Needless to say, the RBP did work in taking the above picture.