ECE 5725: Lab 1

Outline

In this first lab, we will go through the following steps:

- Part 1:
  - Assemble the lab kit for ECE5725
  - Install and configure the Raspbian Linux kernel
- Part 2:
  - Install and configure drivers and software for the piTFT display
  - Install and configure drivers and software for video and audio playback
- Part 3:
  - Develop Python scripts to control playback on the screen using physical buttons installed on the piTFT board

Lab safety

We have all been in lab before and handled electronic components. Please apply all the cautions that you have learned including the items below:
Integrated electronics are REALLY susceptible to static discharge.

- Avoid walking around while holding bare components
- Confine assembly to a single location and discharge static electricity frequently

Personal safety

- Safety goggles are REQUIRED for soldering (even if you solder at home!)

Experimental Safety

- If something on your board is
  - Really hot
  - Smoking
  - Just doesn't look right
- Immediately unplug power
- Train yourself so that this is your first reaction; know where the power switch/cutoff is for your experiment.

Parts of the ECE5725 kit

Raspberry Pi 4 Model B
5V, 2.5A power adapter (USB-c)
16 GByte SD card (mini card in SD carrier)
2.8 inch TFT screen
USB Keyboard and Mouse (if available)
DVI connected display (if available)
DVI to HDMI adapter
40-pin breakout cable
Ethernet cable

Notes before you start:

- The R-Pi is powered up at the END of this assembly AFTER you show your assembly to the TA.
- The 40-pin GPIO connectors are NOT keyed. This means they will fit in a number of different orientations. Check that your assembly matches the photos AND check your assembly with the TA before applying power.

Initial assembly and start up:
In the following order, assemble all the components (please don't forget to discharge static electricity) Here is a photo of the initial system elements:

Elements of the ECE5725 Lab system: from top, left to right, RPi 4 case and cover, PiTFT display, RPi 4, Pi breakout cable and 40-pin cable header, 5V, 3.0 amp power supply, Ethernet cable, HDMI cable.

Assemble your system using the following steps:

- Unpack the R-Pi
- Install the R-Pi in the bottom of the R-Pi case. To install the RPi in the case, Align the RPi so they the connectors are oriented correctly in the case.

- Set the RPi over the 4, round standoffs in the case
- Apply gentle pressure until the RPi ‘seats’ into the case, snapping into the standoffs.
- Note that use of the bottom half of the case provides some additional protection against ESD damage for the RPi electronics
- The top of the case will not be used. Please save the top of the case along with your other kit parts.

These photos show the RPi placed in the case (left) and the side view of the RPi in the case showing the correct alignment of the RPi connectors (right).

Next assembly steps:

- Unpack the TFT
- Unpack the breakout cable.
- Install the breakout cable on the underside of the TFT (the reverse of the display side). Shown below is the underside of the piTFT showing the connectors. The breakout cable is shown on the left side of the piTFT:

This photo shows the breakout cable (left) and the underside of the piTFT including the 40-pin connectors

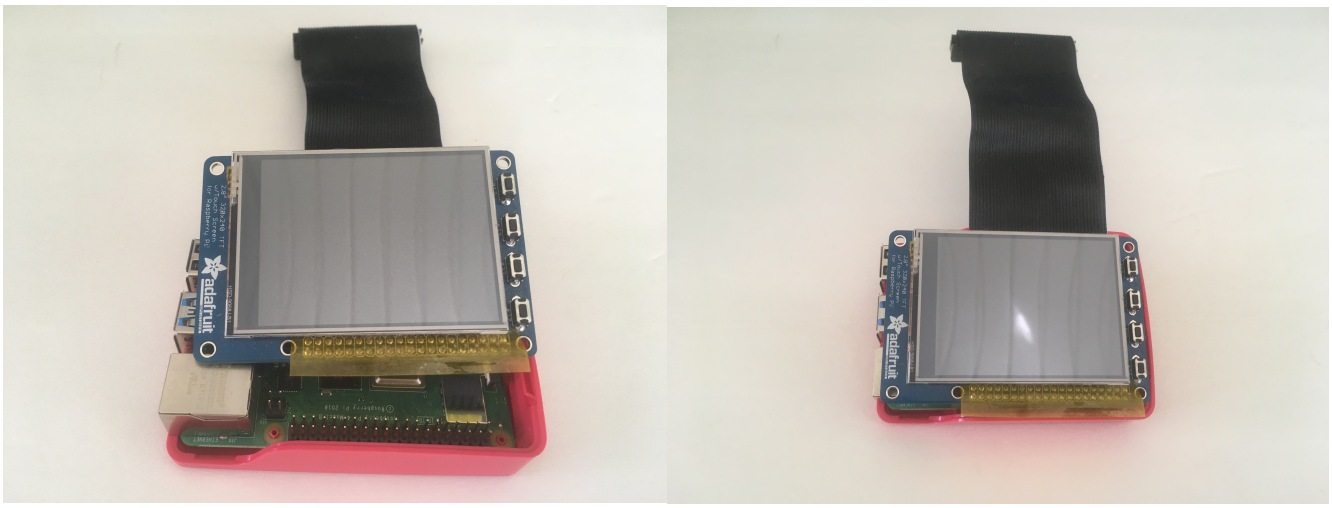
- Plug the TFT into the RPi 40 pin GPIO connector.
- Below is a photo of the piTFT, with the cable correctly plugged, shown next to the RPi.
- Note that the cable is oriented correctly in the photo below. The white stripe on the cable is shown on the left side of the PiTFT. In this orientation, the, white stripe of the cable will be under the 4 PiTFT buttons.
- When plugging the cable, pay special attention:
    - It is easy to mis-plug the 40-pin cables either off by one row or by one column. Check that your cable is correctly aligned with all 40 pins.
    - The 40-pin connectors take some special care to plug and unplug. Work the connectors slowly from side to side rather than trying to plug, or unplug, the cable in one motion.

This Photo shows the correct installation of the expansion cable on the back of the piTFT display. Note the position of the white stripe on the cable relative to the piTFT

Flip the screen onto the RPi and carefully plug the screen into the 40-pin connector on the RPi. Your completed R-Pi, PiTFT screen, and breakout cable should look like the photos, below.

Note that the white stripe on the cable is positioned next to the small '1' on the pin side of the piTFT. The white stripe on the cable should be under the buttons on the side of the TFT screen.

Note that the breakout cable header is not installed in the photos (save this part in your parts box; it will be installed in later labs):



The above photos show the piTFT positioned on the RPi just before installation (left) and the piTFT correctly installed on the RPi 40-pin connector (right)

In the above photos, note the installation of the PiTFT on the RPi. In the previous step, you installed the breakout cable on the back of the PiTFT. Now, position the PiTFT (with cable installed) over the RPi and plug the 40 pin connector socket on the underside of the PiTFT to the 40 pins on the PiTFT. Make sure the pins and socket are lined up before you plug everything together. See the photos above for guidance

Once this assembly step is completed, extract the mini SD Card from the carrier and insert in the bottom slot of the R-Pi. Note the correct position of the mini-SD.

Accessing the Raspberry Pi:

At your lab station, there should be

- an HDMI cable attached to the monitor
- USB mouse
- USB keyboard
- wired ethernet cable

Plug each of these into the appropriate adapters on the RPi. Note that you should plug in the HDMI cable to HDMI connection 1 which is right next to the USB C power adapter. DO NOT plug in power yet…see the next step.

Before you power-up your system:

- **Show the TA your setup**
- Double check: HDMI must be plugged in before booting the RPi
- Plug in the mini-USB connector from the power adapter
- Plug adapter into 110V outlet

Once you plug in your RPi, you should notice a tiny red LED and a tiny blinking green LED.

Important Note: From Top 10 Linux Commands, shutdown and power-off

When you need to power off the R-Pi, please use the following commands:

- Issue the command
  `sudo shutdown –h now`
  which instructs Linux to enter a 'shutdown' state.
- Once on-screen messages are completed and the green LED on the R-Pi stops flashing, unplug power. Note that the flashing green led on the RPi indicates that the RPi is accessing the SD card
- It is best to unplug the 110V power adapter rather than the mini-USB adapter (on the R-Pi board).
- To restart without powering down, Issue the command
  `sudo reboot`
  which instructs Linux to restart the operating system.

Initial Raspbian Kernel install

The system will initially boot to the monitor. There will be a sequence of startup screens.

'Welcome to Raspberry Pi Desktop'
Hit next

'Set Country'
For Country, Select 'United States'
For Language, Select 'American English'
For Timezone, Select 'Eastern'
Click 'Use English Language' and 'Use US Keyboard'
Hit Next

Create User

Enter UserName: pi
Enter Password: something_you_can_remember!
Confirm Password: something_you_can_remember!

Important Note: If your keyboard has a keypad, DO NOT use this to enter any Password information

You may get a message "You have used a known default for the username or password". Please click the 'OK' button at this screen.

Set Up Screen

Leave the default for 'Reduce the size of the Desktop on this Monitor'. Hit **next**

"Select WIFI Network" hit **skip**

"Update Software" hit **skip**

"Setup Complete" Hit **restart**

Once the Raspberry Pi reboots, please continue with the following configuration

The system should boot to the Raspbian desktop on the HDMI connected monitor. We will use the raspi-config tool which will start a configuration utility to assist in setting up the Raspbian kernel.

There are some raspi-config details (for reference only) in the following links:
https://www.raspberrypi.org/documentation/configuration/raspi-config.md
https://www.raspberrypi.org/documentation/configuration/
Raspi-config helps with some standard Linux install tasks that you can also run via the Linux command line, or by modifying Linux config files.

The RPi will boot to the RPi desktop. In the upper left is a black icon used to start a command line window. For setup, open a command line window. Once you open this window, type the following commands:

- Run the following commands to get some system information including the OS version

```
pi@raspberrypi:~ $ whoami
pi
pi@raspberrypi:~ $ hostname
raspberrypi

pi@raspberrypi:~ $ uname -a
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST
2023 aarch64 GNU/Linux

pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL=https://bugs.debian.org/
```

NOTE: The last 2 commands indicate the operating system version. We will be using the 64 bit, 6.1 version(see the output of the `uname` command), Bullseye release (see the output of the `cat /etc/os-release` command). If you see something different, please check with the TAs!

After having a brief look at the system, continue with the following install steps:

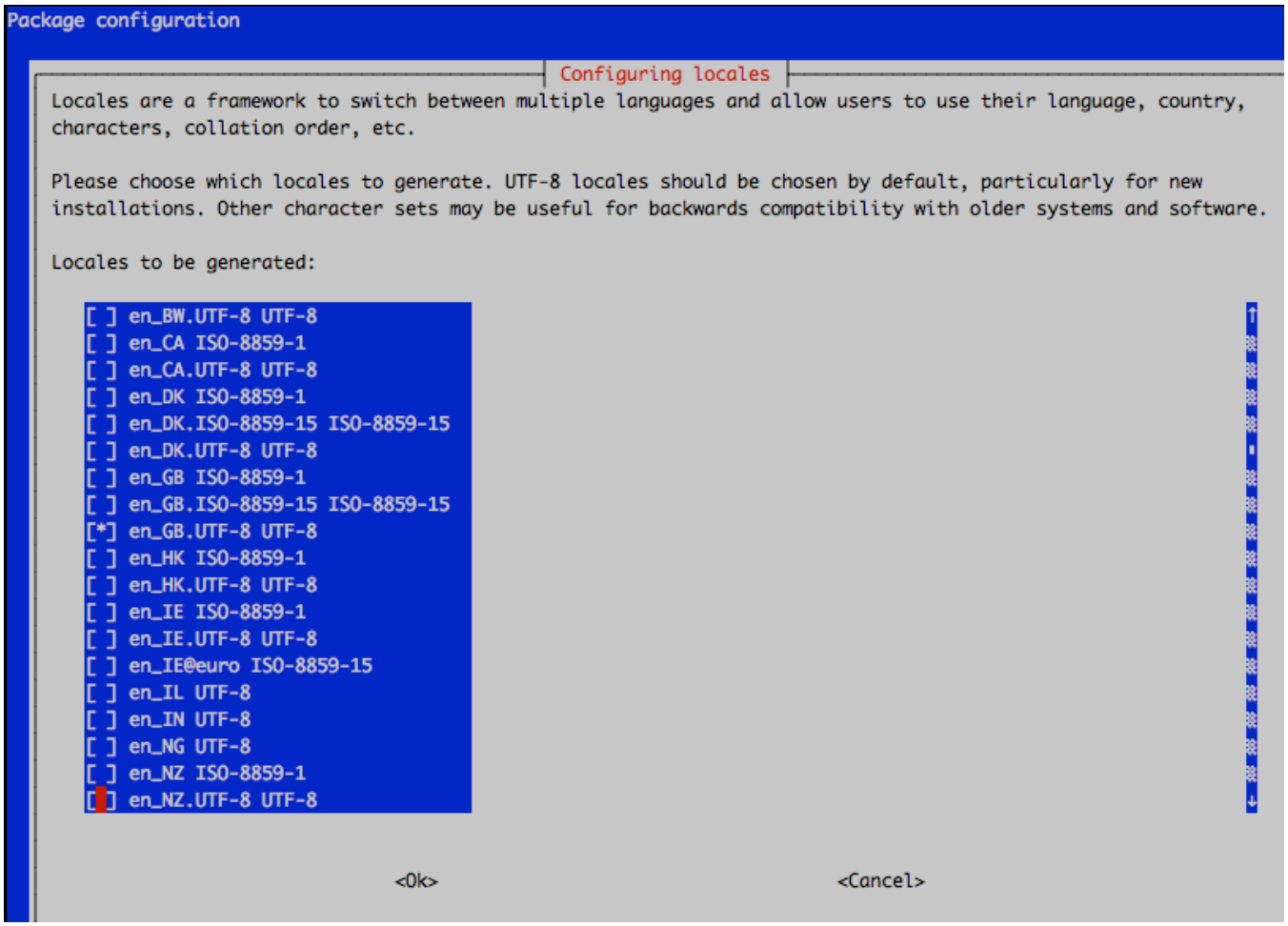
- On the upper RHS of the window, hit the WIFI icon (2 arrows)
- If prompted, click 'Click here to set WI-FI country' to the United States
- Once you see a list of networks, select **RedRover**
- Open a command line window on the desktop

Working your way through the config screens, please make sure to apply the following settings:

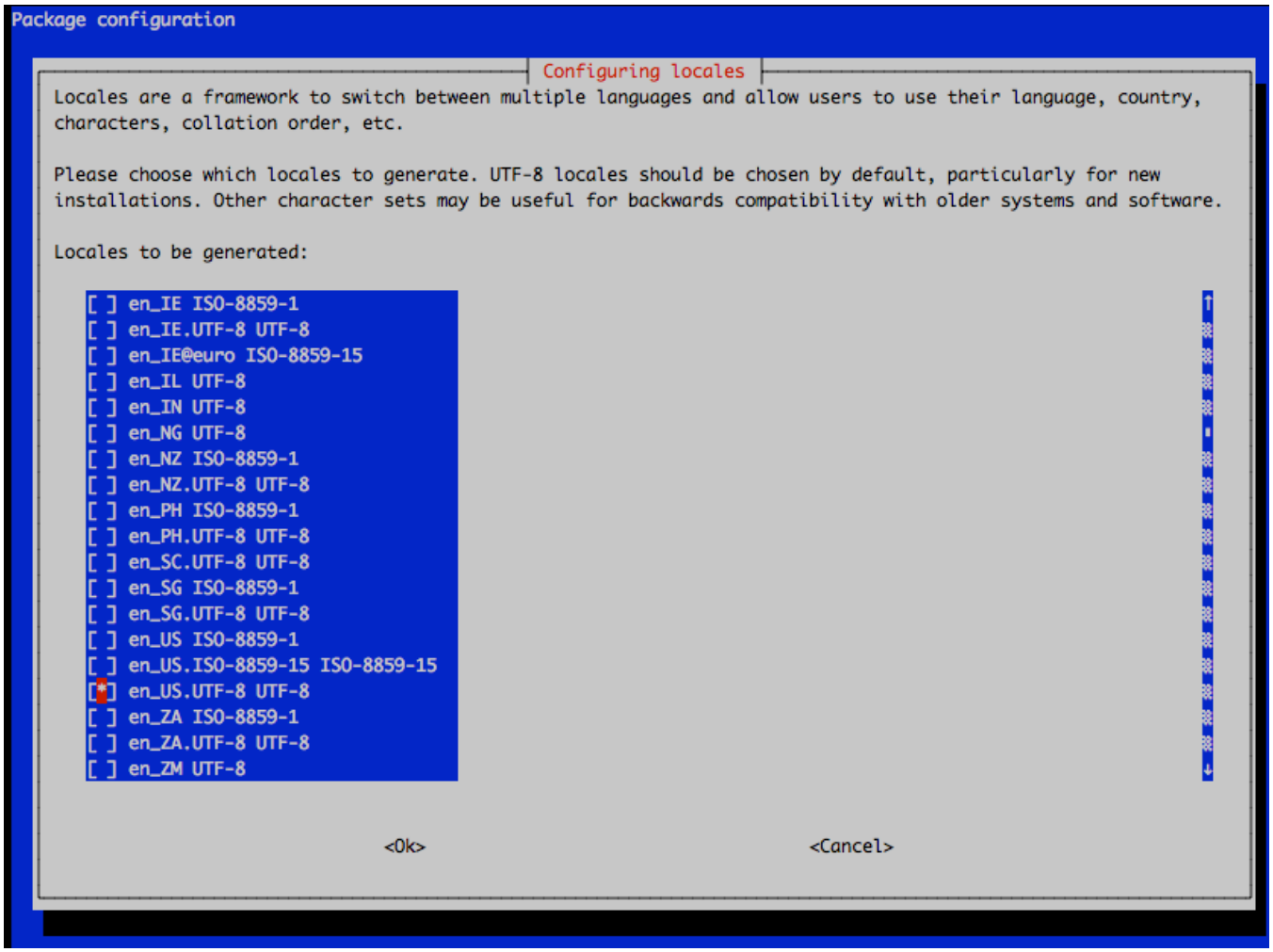
- First, issue the command 'sudo raspi-config'
- In 'System Options', select 'S4 Hostname': set a unique hostname to 'netid-netid' where 'netid-netid' are the Cornell netids of the two team members in your group.
- In 'System Options', select 'S5 Boot/Auto Login'. Select 'B1 Console'
    - This setting will boot the raspberry Pi to the console window.

- You will be able to run the desktop by issuing the command “startx’ in the console window.
- To exit from the desktop back to the console, you will use the keystrokes ‘CTRL-ALT-backspace’ (setup later!)
- These instructions will come in handy later…..!

- In ‘Localization options’:
  - Select ‘locale’:
    - **Use the arrow keys, and tab to navigate**
    - Use the Spacebar to select / de-select
    - If set, De-select ‘en_GB.UTF-8 UTF-8’.
    - If NOT set, Set the Locale by selecting ‘en-us.UTF8 UTF8’.
    - Hit ‘Ok’
    - When prompted, select ‘en_US.UTF-8’ for the Default Locale
    - Hit ‘Ok’

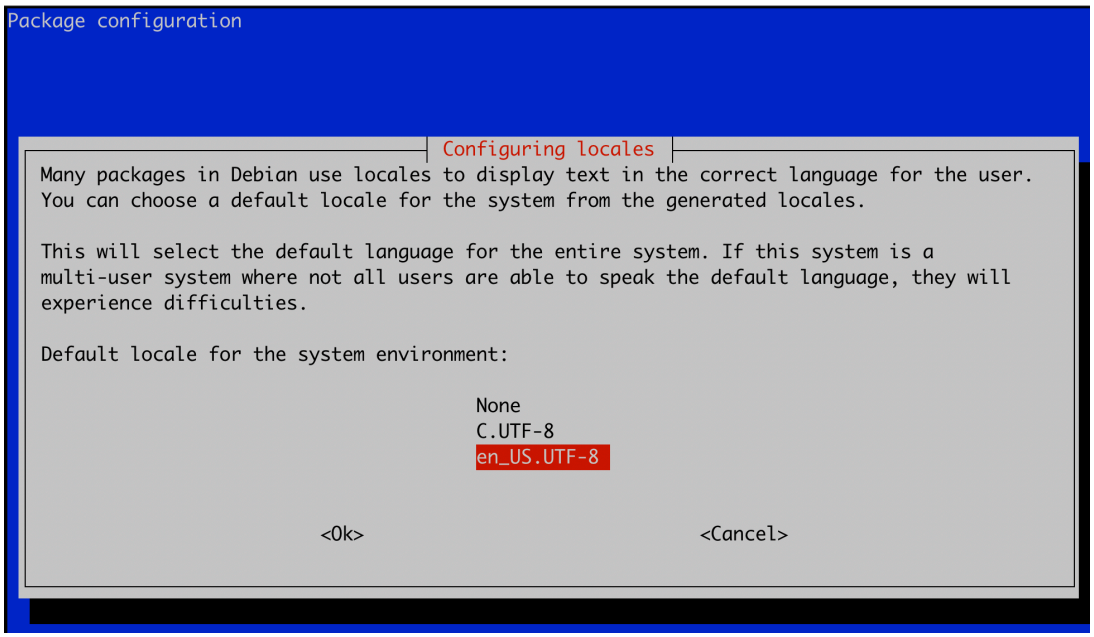


This clip shows the selection of the en_GB locale. You will unclick this selection



This clip shows the correct en_US setting

One you have the correct settings, tab to OK and hits enter.



Select en_US.UTF-8 on the above screen

- Again, in Localization Options, Select 'Keyboard' and make the following changes:
  - 'Generic 101 key PC' keyboard
  - Select 'English US' – note, this may be set on two screens
    - Next, select 'No AltGR key'
    - next, select 'no compose key'
    - finally, select 'YES' to allow "Ctrl-Alt-backspace to exit the X server"

  Why these settings? These are the best settings for the time, keyboard and mouse you will be using.

- Again, in Localization Options, Change Timezone
  Change the timezone to match your location. In Ithaca, for example, set the time zone as follows:
  - Set the geographic area to US
  - Set the Time Zone to 'Eastern'

Select Finish. If a pop-up asks 'reboot now?' Click, yes. If not, enter '`sudo reboot`' to reboot the system at this point.

- Login as user 'pi'
- Enter your new password

- Once logged in, run 'sudo raspi-config' to continue with configuration.

Additional configuration:
- In 'Interface Options': enable ssh
- In the 'Advanced Options' settings
  - A1, expand the file system
- In the 'System Options' settings
  - S2, Audio, Select "Headphones"
- In 'Update': update the raspi-config tool to the latest version

Once you hit 'Finish', raspi-config will save your changes and ask if you want to reboot now. Answer 'yes'.

If it does not ask for a reboot, issue the following command:

```
pi@netID:~ $ sudo reboot
```

To reboot the RPi.

Note that you can always run raspi-config to adjust configuration settings at a later time.

Updates

Once the system boots to the command line, login as the pi user

The following command (run earlier) shows the initial version of the Operating System:

```
pi@RPI-jfs9:~ $ uname -a
Linux netID-netID 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023
aarch64 GNU/Linux
```

Check for Raspbian kernel upgrades by running the following two commands:

- `sudo apt-get update`
- `sudo apt-get full-upgrade`

If you watch the screen during these operations, you will notice the packages that are loaded and installed to update the kernel.

The upgrade could take a few minutes.

Important Notes:

- Watch the update and upgrade as they proceed. The commands print a lot of cryptic messages on progress (most Linux commands print console messages). In this case, look for messages that state:
  `Unable to resolve 192.168.10.20  # where 192.168.10.20 is some IP address`
  This means that the update command is having problems getting to the server and the update will not be complete.
- You may have to answer some questions for the upgrade to proceed. Read the comments and answer the prompts to allow the upgrade to continue.

Finally, Load an update for the vim editor:
`'sudo apt-get install vim'`

Reboot to start the upgraded kernel

Once updates are done, run:

```
'sudo reboot'   # to reboot the system
```

You can also shut down the system cleanly using:

```
'sudo shutdown –h now'
```

To restart the system after a shutdown, cycle power on the RPi.

Note: `'sudo shutdown –h now'` should always be run before you unplug the R-Pi setup. The shutdown command allows Linux to gracefully shutdown all processes before the system is powered down.

have a look at the 'shutdown' command to explore the –r and –h flags

Run '`uname –a'` to show the updated kernel version.

```
pi@RPI-jfs9:~ $ uname -a
Linux netID-netID 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023
aarch64 GNU/Linux
```

Note; There are many online discussions about the Raspberry Pi. If you stumble across the following command, NEVER run:

DO NOT RUN !!! > `'sudo rpi-update'` (This loads the latest, un-released, un-tested, development kernel)

… just a warning, because you might see this as you wander the online R-Pi pages

If you are using an HDMI connected monitor, From the command line issue the command

```
pi@RPI-jfs9:~ $ startx
```

Which will start the Raspberry Pi Desktop

Once in the desktop, exit back to the command line window using

ctrl-ALT-BACKSPACE

which you setup earlier using the raspi-config tool

Once you have a command line window running on the desktop, try some simple checks:

- 'hostname'
- 'ifconfig –a' to check network connections
- 'cat /proc/cpuinfo' to check the number of cores
- 'time date'
- 'htop' # Note, exit htop with 'CTRL-C'

If you are using an HDMI monitor, using MobaXTerm or putty (on windows) or terminal (on mac), login to the R-Pi to make sure you can! From your laptop, you will use secure shell. An example (on a Mac) is:

```
ssh pi@123.456.78.90
```

Where pi is the username and 123.456.78.90 is the IP of your 'wlan0' WIFI connection.

If you are having trouble accessing RPi with ssh, revisit the configuration for ssh in 'Interface Options' using the command 'sudo raspi-config'. Use command 'sudo systemctl status ssh' on Rpi to verify that the ssh.service is active.

If Rpi says the service is *inactive*, *failed*, or *not found*, then:

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

A Word of caution: You can load a lot of programs on RPi. Sometimes, the installs can drop into many different directories and create large files. In later labs, we might need additional free space to proceed, and it may be difficult to recall what was installed and how to get rid of it. You are indeed encouraged to explore. Here are some tips for keeping your SD card trim:

- You may want to consider a second SD card for loading experimental applications
- Keep a list of all installed apps (and how to completely remove them)
- Use list of best practices for good SD card behavior
- Keep backups as you go…restore one of the cleaner backups during a later lab if you are running out of space.

At this point, the RPi contains a clean install of the latest Raspbian kernel.

Backup the SD card!

You will thank yourself later……

TFT Screen Install

There are detailed instructions posted on the Adafruit site on installation of the piTFT with the most updated versions. But we must downgrade to the stable version of Jessie to be able to use the piTFT.

Use The following instructions:

Install software to support the piTFT:

Run:

```
sudo apt-get install -y bc fbi git python3-pip python3-smbus evtest libts-bin
```

The evtest and libts-bin applications are used for testing the piTFT.

Bad example:

The following shows an example of a failed installation; you might see something like this and it is important to read through the messages to determine what is going on:

```
pi@netID-netID:~ $ sudo apt-get install -y bc fbi git python3-pip python3-smbus
evtest libts-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libts-0.0-0' instead of 'tslib'
git is already the newest version (1:2.11.0-3+deb9u4).
python-smbus is already the newest version (3.1.2-3).
python-pip is already the newest version (9.0.1-2+rpt2).
python-spidev is already the newest version (20170223~145721-1).
The following additional packages will be installed:
  evemu-tools ghostscript gsfonts libevemu3 tsconf
Suggested packages:
  imagemagick ghostscript-x
The following NEW packages will be installed:
  bc evemu-tools evtest fbi ghostscript gsfonts libevemu3 libts-0.0-0 libts-bin
tsconf
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 3,480 kB of archives.
After this operation, 5,494 kB of additional disk space will be used.
Get:1 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf tsconf all
1.0-12 [13.4 kB]
Get:2 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf libts-0.0-
0 armhf 1.0-12 [25.8 kB]
Get:3 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf bc armhf
1.06.95-9 [96.3 kB]
Get:4 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf libevemu3
armhf 2.6.0-0.1 [11.0 kB]
Get:5 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf evemu-
tools armhf 2.6.0-0.1 [14.1 kB]
```

```
Err:6 http://raspbian.raspberrypi.org/raspbian stretch/main armhf ghostscript armhf
9.26~dfsg-0+deb9u2
  404  Not Found [IP: 93.93.128.193 80]
Get:7 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf fbi armhf
2.10-2+b1 [53.5 kB]
Get:8 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf gsfonts
all 1:8.11+urwcyr1.0.7~pre44-4.3 [3,126 kB]
Get:9 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf libts-bin
armhf 1.0-12 [28.6 kB]
Get:10 http://mirror.us.leaseweb.net/raspbian/raspbian stretch/main armhf evtest
armhf 1:1.33-1 [13.2 kB]
Fetched 3,381 kB in 3s (952 kB/s)
E: Failed to fetch
http://raspbian.raspberrypi.org/raspbian/pool/main/g/ghostscript/ghostscript_9.26~d
fsg-0+deb9u2_armhf.deb  404  Not Found [IP: 93.93.128.193 80]
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-
missing?
```

Note the messages ‘Not Found’ and ‘Failed to fetch’. These indicate there were some issues with loading the applications. There is a hint in the last message to run some commands to try and fix the problem. Running apt-get update, as suggested in the message, led to the following example.

Good Example:

```
pi@netID-netID:~ $ sudo apt-get install -y bc fbi git python3-pip python3-smbus
evtest libts-bin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
bc is already the newest version (1.07.1-2+b2).
bc set to manually installed.
git is already the newest version (1:2.30.2-1+deb11u2).
python3-smbus is already the newest version (4.2-1+b1).
python3-pip is already the newest version (20.3.4-4+rpt1+deb11u1).
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  evemu-tools libevemu3 libts0
Suggested packages:
  imagemagick
The following NEW packages will be installed:
  evemu-tools evtest fbi libevemu3 libts-bin libts0
0 upgraded, 6 newly installed, 0 to remove and 11 not upgraded.
Need to get 206 kB of archives.
After this operation, 828 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main arm64 libevemu3 arm64 2.7.0-3
[12.4 kB]
Get:2 http://deb.debian.org/debian bullseye/main arm64 evemu-tools arm64 2.7.0-3
[15.3 kB]
Get:3 http://deb.debian.org/debian bullseye/main arm64 evtest arm64 1:1.34-1 [15.7
kB]
Get:4 http://deb.debian.org/debian bullseye/main arm64 fbi arm64 2.10-4 [59.0 kB]
Get:5 http://deb.debian.org/debian bullseye/main arm64 libts0 arm64 1.22-1+b1 [63.4
kB]
Get:6 http://deb.debian.org/debian bullseye/main arm64 libts-bin arm64 1.22-1+b1
[40.5 kB]
Fetched 206 kB in 1s (343 kB/s)
Selecting previously unselected package libevemu3:arm64.
(Reading database ... 98806 files and directories currently installed.)
Preparing to unpack .../0-libevemu3_2.7.0-3_arm64.deb ...
Unpacking libevemu3:arm64 (2.7.0-3) ...
Selecting previously unselected package evemu-tools.
Preparing to unpack .../1-evemu-tools_2.7.0-3_arm64.deb ...
Unpacking evemu-tools (2.7.0-3) ...
Selecting previously unselected package evtest.
Preparing to unpack .../2-evtest_1%3a1.34-1_arm64.deb ...
Unpacking evtest (1:1.34-1) ...
Selecting previously unselected package fbi.
Preparing to unpack .../3-fbi_2.10-4_arm64.deb ...
Unpacking fbi (2.10-4) ...

… many more lines of installation
```

Add piTFT info to config.txt

Edit the file /boot/config.txt using vim and add the following lines to the end of the file, before the final [all]. Note that the /boot/config.txt file is accessible by administrators so you must use 'sudo' when editing:

```
# 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-capacitive,rotate=90,speed=64000000,fps=30
```

Notes

- Comments in this file begin with leading #
- Pay attention of upper and lower case
- Watch for differences in underscore "_" and dash "-"
- The various device_tree settings include code for the appropriate Pi model.
- The 'rotate' setting is used to rotate the screen 0, 90, 180, or 270 degrees. We will be using a landscape rotation of 90 degrees.

The speed setting is used by the driver to determine how fast to drive the display. 64Mhz (64000000) is the suggested setting. We might decide to reduce this to 32MHz at some point.

Once these changes are in place, **reboot the RPi** and observe the startup sequence. If all is well, the PiTFT screen should start out white and switch to black as the boot sequence finishes. If this happens, the initial changes are working correctly.

Run dmesg to check for piTFT

At this point, run ‘dmesg’ to see that the modules for the touch screen are installed correctly. In particular, look for the “stmpe-spi” and ‘graphics fb1” lines (highlighted in the example below.

Notes:

- The chipID is for the resistive touch circuits on the PiTFT
- Note the settings in the “graphics fb1” entry; some of these were set in previous steps.

Example dmesg output:

```
[    7.904086] SPI driver fb_ili9340 has no spi_device_id for ilitek,ili9340
[    7.908236] fb_ili9340 spi0.0: fbtft_property_value: buswidth = 8
[    7.908286] fb_ili9340 spi0.0: fbtft_property_value: debug = 0
[    7.908304] fb_ili9340 spi0.0: fbtft_property_value: rotate = 90
[    7.908326] fb_ili9340 spi0.0: fbtft_property_value: fps = 30
[    8.163171] graphics fb1: fb_ili9340 frame buffer, 320x240, 150 KiB video
memory, 4 KiB buffer memory, fps=31, spi0.0 at 64 MHz
[    9.053166] input: BRLTTY 6.3 Linux Screen Driver Keyboard as
/devices/virtual/input/input8
[    9.403557] 8021q: 802.1Q VLAN Support v1.8
```

Add a udev rule

Next, some files will be created to support the use of touch on the pitft. The following changes will create a 'udev' rule which will assign the touchscreen an eventX number which will make accessing the touchscreen more straightforward

The files are all in /etc/udev/rules.d which is accessible by administrators. As a result, use 'sudo' when you edit the files.

Edit /etc/udev/rules.d/95-stmpe.rules (this will create a new file), and add the line:

```
SUBSYSTEM=="input", ATTRS{name}=="*stmpe*", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"
```

Edit /etc/udev/rules.d/95-touchmouse.rules (this will create a new file), and add the line:

```
SUBSYSTEM=="input", ATTRS{name}=="touchmouse", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"
```

Edit /etc/udev/rules.d/95-ftcaptouch.rules (this will create a new file), and add the line:

```
SUBSYSTEM=="input", ATTRS{name}=="EP0110M09", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"
                          |    |                |       |
```

Note that these are "curly braces" = "{ }", NOT parentheses " ( ) ".

Once you have these files in place, unload the driver for the touch screen and reload it (so the system can pick up the new information):

Run :

```
sudo rmmod stmpe-ts
```

to stop the touchscreen module

Note: you may see this message:

```
pi@netID-netID:~ $ sudo rmmod stmpe-ts
rmmod: ERROR: Module stmpe_ts is not currently loaded
```

This just means that the module is not currently loaded and running, this is an OK response!

Run:

```
sudo modprobe stmpe-ts
```

To restart the touchscreen module, and pick up the new changes

Set initial piTFT calibration

Create the file with `sudo vim /etc/pointercal` and add the line, which are fixed calibrations with a space between each of the numbers for the piTFT:

```
320 65536 0 -65536 0 15728640 65536
```

Start console window on piTFT

The following changes will allow the Linux console window to start on the piTFT. All files in this section are controlled by the administrator.

First, modify /boot/cmdline.txt to include the following right after 'rootwait':

```
fbcon=map:10 fbcon=font:VGA8x8
```

Do not include any 'returns' in this one-line file and note these variables are separated by spaces. Here is an example of the file on a test system:

```
pi@netID-netID:~ $ cat /boot/cmdline.txt
console=serial0,115200 console=tty1 root=PARTUUID=dc208b7b-02 rootfstype=ext4
fsck.repair=yes rootwait fbcon=map:10 fbcon=font:VGA8x8 quiet splash
plymouth.ignore-serial-consoles
```

Modify the font and font size for displaying the console on the piTFT. Modify the 2 entries in /etc/default/console-setup as follows:

```
FONTFACE="Terminus"
FONTSIZE="6*12"
```

This will give a more compact, readable font on the smaller piTFT screen. Example of the altered file:

```
pi@netID-netID:~ $ cat /etc/default/console-setup
# CONFIGURATION FILE FOR SETUPCON
# Consult the console-setup(5) manual page.
ACTIVE_CONSOLES="/dev/tty[1-6]"
CHARMAP="UTF-8"
CODESET="guess"
FONTFACE="Terminus"
FONTSIZE="6*12"
VIDEOMODE=
# The following is an example how to use a braille font
# FONT='lat9w-08.psf.gz brl-8x8.psf'
```

Add an entry in /etc/rc.local to turn off piTFT blanking when the console is displayed. At the end of the file but BEFORE the trailing exit, add the following:

```
# disable console blanking on PiTFT
sudo sh -c "TERM=linux setterm -blank 0 >/dev/tty0"
```

Here is an example file including the required change:

```
pi@netID-netID:~ $ cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

# disable console blanking on PiTFT
sudo sh -c "TERM=linux setterm -blank 0 >/dev/tty0"

exit 0
```

Once all the changes are completed:

reboot the system

If all is well, the Linux console should start on the PiTFT. Experiment with the ‘startx’ command to explore different environments now configured on the system.

The instructions in lab up to this point should replicate all the install steps that previously existed at the Adafruit site.

One set of original Adafruit instruction pages that is correct describes how to run a video on the piTFT. These instructions are here, for reference only:

Playing Videos :
https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/playing-videos

From this page, here are the step-by-step instructions for playing a video from the RPi on the PiTFT:

Load the sample video from the adafruit repository onto the RPi; run the following 'wget' command:

```
pi@netID-netID:~ $ wget http://adafruit-download.s3.amazonaws.com/bigbuckbunny320p.mp4
--2024-08-12 16:42:28--  http://adafruit-download.s3.amazonaws.com/bigbuckbunny320p.mp4
Resolving adafruit-download.s3.amazonaws.com (adafruit-download.s3.amazonaws.com)... 3.5.30.156,
3.5.3.152, 16.182.68.145, ...
Connecting to adafruit-download.s3.amazonaws.com (adafruit-download.s3.amazonaws.com)|3.5.30.156|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 30325854 (29M) [video/mp4]
Saving to: 'bigbuckbunny320p.mp4'

bigbuckbunny320p.mp4
100%[=====================================================================>]  28.92M  29.1MB/s    in
1.0s

2024-08-12 16:42:29 (29.1 MB/s) - 'bigbuckbunny320p.mp4' saved [30325854/30325854]
```

After the wget command completed, run 'ls -l' to check that the video was correctly downloaded:

```
pi@netID-netID:~ $ ls -l
total 29652
-rw-r--r-- 1 pi pi 30325854 Nov 26  2013 bigbuckbunny320p.mp4
drwxr-xr-x 2 pi pi     4096 Jul  3 20:10 Bookshelf
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Desktop
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Documents
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Downloads
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Music
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Pictures
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Public
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Templates
drwxr-xr-x 2 pi pi     4096 Aug  8 17:54 Videos
```

Get mplayer application to run the video:

Check to see if mplayer is installed; Run:

```
pi@netID-netID:~ $ mplayer -h
-bash: mplayer: command not found
```

If it is installed, go directly to the step for playing the video. In the above case, mplayer is not installed so run:

```
sudo apt-get update
sudo apt-get install mplayer
```

Play the video on the piTFT:

If you have an HDMI monitor installed, it will be assigned /dev/fb0 and the piTFT will be assigned /dev/fb1. If you do not have an HDMI monitor installed, the piTFT will be assigned /dev/fb0 since this is the only display. Run the following to show the framebuffers on the RPi:

```
pi@netID-netID:~ $ ls -l /dev/fb*
crw-rw---- 1 root video 29, 0 Aug  6 13:30 /dev/fb0
crw-rw---- 1 root video 29, 1 Aug  6 13:35 /dev/fb1
```

The following command works in the case where the piTFT is assigned to /dev/fb1:

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

This command sets up the sdl video driver to run the video on the piTFT (configured as /dev/fb1)

NOTE: use 'q' to quit mplayer. Unlike most applications, mplayer doesn't exit correctly when using 'ctrl-c'

Audio was setup in the earlier raspi-config step to play from the headphone jack. Please make sure that audio playback operates correctly by either using powered speakers (see the TA to borrow a set of speakers) or through headphones. Some useful references: Run 'mplayer –ao help' to list all possible audio drivers

To boost the audio volume run:

```
pi@netID-netID:~ $ amixer scontrols
Simple mixer control 'Master',0
```

```
Simple mixer control 'Capture',0
```

This shows the ‘simple controls’ for amixer. Master is master volume

The following command shows controls for Master volume

```
pi@ netID-netID:~ $ amixer -M sget Master
Simple mixer control 'Master',0
  Capabilities: pvolume pswitch pswitch-joined
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 65536
  Mono:
  Front Left: Playback 56210 [86%] [on]
  Front Right: Playback 56210 [86%] [on]
```

The following command sets volume to max

```
pi@netID-netID:~ $ amixer -M sset Master 100%
Simple mixer control 'Master',0
  Capabilities: pvolume pswitch pswitch-joined
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 65536
  Mono:
  Front Left: Playback 65536 [100%] [on]
  Front Right: Playback 65536 [100%] [on]
```

The above command sets volume to max

Some other useful audio controls:

```
aplay –l
```
lists info about sound cards
```
aplay /usr/share/sounds/alsa/Noise.wav
```
plays a built-in wav file to test headphone jack

At this point, the TFT has been successfully configured

Test from several consoles

- Test by starting mplayer directly on piTFT console window
- Run desktop on the monitor and start mplayer using a console window
- Remote login with ssh on your laptop. Start the mplayer video remotely

For each of the above tests, run the video on BOTH the piTFT and the monitor (framebuffers fb0 and fb1). Record results for video and audio output:

- Did video/audio playback?
- Where was video displayed

Are there any problems with any of these experiments (for example, does the video always run on the piTFT? Does the audio play? Is the audio synchronized correctly?) Let the video run and check as it progresses. Document your findings and discuss any issues with a TA. A table of results would be a great idea!

Audio Notes, Spring 2021, verified Fall 2021:
During testing, it seems that 'sudo' preceding audio commands could impact output to the headphone jack. For example:

- mplayer 'noise file' # plays correctly
- sudo mplayer 'noise file' # may not playback to headphone jack

So, for tests of mplayer with the video, you may see (**without** using sudo):

- Starting on piTFT; video on piTFT, audio on headphone jack
- Starting on desktop; video in window, audio on headphone jack
- Starting from ssh window; no video, audio on headphone jack

Please check the following with the TA:

- Demonstrate the piTFT screen by playing the video and audio
    - Launch mplayer from the console window on piTFT
    - Launch mplayer from a console window in startx
    - Launch mplayer from a remote session on your laptop
- Show the TA your backup files

Stop and Backup your SD card

Once this part of your lab has been completed, back up your SD card using instructions posted on canvas.

In previous semesters, we verified the backup process by restoring a backup file to the team's second SD card. The instructions are included below.

Plan this step for the beginning of Lab1, Week 2: Switch SD cards and launch mplayer from the switched card using the piTFT console window (demonstrate two running SD cards)

Once you have created the latest backup, restore the backup to the second SD card. Once restored, switch cards in the RPi and boot the second card. Confirm that this card functions identically to the initial card.

Between week 1 and week 2, we want you to restore your latest backup to the second SD card. At the beginning of week 2, you will demo the system booting and running the video to one of the TAs. This will provide a good check for your end-to-end system backup process, that is, your ability to both take a backup and restore a backup.

Demo at Start of Lab1, week 2:

Demo the system booting and running the video from the second SD card to one of the TAs. This will provide a good check for your end-to-end system backup process, that is, your ability to both take a backup and restore a backup. This should be a demo of the already restored SD card; The restore process must be completed prior to the start of your lab section.

Week 2

Demo at Start of Lab1, week 2:

Demo the system booting and running the video from the second SD card to one of the TAs. This will provide a good check for your end-to-end system backup process, that is, your ability to both take a backup and restore a backup. This should be a demo of the already restored SD card; The restore process must be completed prior to the start of your lab section.

A quick recheck: Make sure the output of the uname -a command on your system is as follows:

```
pi@netID-netID:~ $uname -a
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023 aarch64 GNU/Linux
```

This setting indicates that the 64-bit OS has been configured.

Capacitive piTFT configuration:

Run the command:

```
pi@netID-netID:~ $ dmesg | grep EP0110M09
[    9.984681] input: 1-0038 EP0110M09 as /devices/platform/soc/fe804000.i2c/i2c-1/1-0038/input/input3
```

Note from this sample command that the EP0110M09 driver is being used for ‘input3’ which if the piTFT touch screen. Using this information, please modify the following file which was created in week 1:

Edit /etc/udev/rules.d/95-ftcaptouch.rules and modify the line (note: this may already beset correctly but please check!):

```
SUBSYSTEM=="input", ATTRS{name}=="EP0110M09", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"
```

Once the file has been modified:

```
pi@netID-netID:~ $ cat /etc/udev/rules.d/95-ftcaptouch.rules
SUBSYSTEM=="input", ATTRS{name}=="EP0110M09", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchscreen"
                           |    |                |       |
```

Note that these are “curly braces” = “{ }”, NOT parentheses “ ( ) “. Sorry for the tiny font but I wanted to make sure that these examples show that the variables are entered on a single line with NO returns.

If this file was changed from week 1, please reboot the system then check the following:

To check the operation, run:

```
ls -l /dev/input/touchscreen
```

The touchscreen should now be associated with an event number as in the example below:

```
pi@netID-netID:~ $ ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Sep  4 08:55 /dev/input/touchscreen -> event3
```

These numbers MUST MATCH

WARNING! – Do NOT just run the above command if you are trying to resolve this issue! The value '`event3`' is an example from MY system. The settings on your system maty be different. In this case '`event3`' matches the setting '`input3`' obtained from the command below. In my case, the numbers for `input3` and `event3` match. Check your system to make sure the numbers match!

If there is **NO** entry for /dev/input touchscreen, please try the following:
Note the input number from the command above:

```
pi@netID-netID:~ $ dmesg | grep EP0110M09
[    9.984681] input: 1-0038 EP0110M09 as /devices/platform/soc/fe804000.i2c/i2c-1/1-0038/input/input3
```

In this case, on a sample pi, the device is **input3**. Your result may have a different input number. Noting this, run the following commands:

```
cd /dev/input
pi@netID-netID:/dev/input $ sudo ln -s event3 /dev/input/touchscreen
```

The above command creates a link between event3 (using the same number as the 'input' result from the dmesg command) and /dev/input/touchscreen.

Checking the result:

```
pi@netID-netID:/dev/input $ ls -l
total 0
drwxr-xr-x 2 root root     100 Feb 10 20:31 by-id
drwxr-xr-x 2 root root     160 Feb 10 20:31 by-path
crw-rw---- 1 root input 13,  64 Feb 10 20:31 event0
crw-rw---- 1 root input 13,  65 Feb 10 20:31 event1
crw-rw---- 1 root input 13,  66 Feb 10 20:31 event2
crw-rw---- 1 root input 13,  67 Feb 10 20:31 event3
crw-rw---- 1 root input 13,  68 Feb 10 20:31 event4
crw-rw---- 1 root input 13,  63 Feb 10 20:31 mice
crw-rw---- 1 root input 13,  32 Feb 10 20:31 mouse0
crw-rw---- 1 root input 13,  33 Feb 10 20:31 mouse1
lrwxrwxrwx 1 root root       6 Feb 10 20:50 touchscreen -> event3
```

Notice the link for /dev/input/touchscreen to event3.

Next , test that this worked by running evtest:

```
pi@netID- netID:~ $ 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: "1-0038 EP0110M09"
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      239
    Event code 1 (ABS_Y)
      Value      0
      Min        0
      Max      319
    Event code 47 (ABS_MT_SLOT)
      Value      0
      Min        0
      Max        1
    Event code 53 (ABS_MT_POSITION_X)
      Value      0
      Min        0
      Max      239
    Event code 54 (ABS_MT_POSITION_Y)
      Value      0
      Min        0
      Max      319
    Event code 57 (ABS_MT_TRACKING_ID)
      Value      0
      Min        0
      Max    65535
```

If this runs correctly, tap the piTFT screen a few times to confirm registration of a touch event. This will show up as text response in the command line window. Output from a sample system after tapping the screen:

```
Event: time 1707616529.037417, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value
13
Event: time 1707616529.037417, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value
56
Event: time 1707616529.037417, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value
168
Event: time 1707616529.037417, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1707616529.037417, type 3 (EV_ABS), code 0 (ABS_X), value 56
Event: time 1707616529.037417, type 3 (EV_ABS), code 1 (ABS_Y), value 168
Event: time 1707616529.037417, -------------- SYN_REPORT ------------
Event: time 1707616529.131088, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value
-1
Event: time 1707616529.131088, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1707616529.131088, -------------- SYN_REPORT ------------
```

Once this works, exit evtest using ctrl-C

Download sample files from the ECE5725 server:

Using the 'scp' command we discussed in class, download the sample files from the ece5725 server. The directory on the server is:

```
/home/Lab/lab1_files_f25
```

You will use 'scp' command to move these files from the server to you Raspberry Pi. Logged into the Raspberry Pi, as discussed in class, try the command:

```
scp -p -r my_netid@132.236.79.6:/home/Lab/lab1_files_f25 /home/pi
```

source for the files at the server — destination of the files

These sample files will be helpful for completing the work on Lab1, week 2

Controlling video playback with external devices:

Once the initial system is up and running, and video/audio playback are working, we want to explore playback control using external methods. You will be creating a few scripts for different operations. Save all scripts for later demonstration

Step 1: explore mplayer

The first step is to explore mplayer options for playback control. Type 'mplayer' with no options to see a list of video controls. There are controls for pause, skip ahead and back, and start and stop. Explore these options and decide which of these you might want to map into a control 'panel'.

Step 2: control mplayer with a FIFO

Create a fifo (called 'video_fifo'). Run mplayer and pass appropriate commands ('pause' for example) to the running instance of mplayer using echo commands from a command line screen. Make sure mplayer responds correctly to commands.

Note that one way to proceed would include:

- Boot the RPi to the console on the TFT

- Run startx so that the desktop starts on the monitor
- Open two console windows
- In one console window, run mplayer with appropriate arguments to use the fifo
- In the second window, issue commands to the fifo to control mplayer

For reference on how to use mplayer with a FIFO, check Canvas for the guide; 'using a fifo with mplayer'

Please note that this is an example page, however, keep in mind that the user these are example commands. You will have to make changes for your own system.

The following command lists all of the commands that may be sent to mplayer using a fifo. Search for examples of pause and quit in this list and check for additional useful mplayer commands:

```
mplayer -input cmdlist
```

Also, check this link for fifo commands that may be sent to mplayer. Search for pause and quit and check the descriptions of the commands:

http://www.mplayerhq.hu/DOCS/tech/slave.txt

Step 3: use Python to control mplayer with a FIFO

Write a python routine (called fifo_test.py) to experiment with sending some sample commands (pause is a simple command to begin with). The python routine should:

- Run in the foreground, waiting for some input from the keyboard
- Recognize a valid command (pause, for example) from the user
- Send the valid command(s) to the FIFO setup and used by mplayer
- Control mplayer as expected given the command descriptions
- Recognize a command which quits the script and returns to the command line

Note that this script should pass commands to a process instance of mplayer that is already running on the piTFT. Clearly, the mplayer process will have to monitor the FIFO for input from the script.

Step 4: Get input from a button connected to GPIO

There are 4 buttons on the piTFT. Using the schematic for the piTFT (see the 'Reference' section on the Canvas page), determine which GPIO pins these buttons are connected to and how they are connected.

With this information, write a python routine (called 'one_button.py') that will:

- Use the RPi.GPIO module in python
- Set GPIO numbering to Broadcom
- Initialize one of the buttons correctly
- Monitor the button for a press and display 'Button NN has been pressed' where NN is the pin number of the button

Step 5: Get input from four buttons connected to GPIO

Once the one_button.py routine is working, expand the routine to include all 4 buttons. This routine (called 'four_buttons.py') should

- Be based on one_button.py
- Extend the function to include checks for all 4 buttons being pressed by printing the message 'Button NN has been pressed' where NN is the pin number of the appropriate button.
- For one of the buttons on the 'edge' of the screen, print out 'Button NN has been pressed' and also quit the python program.

Step 6: Control mplayer through a FIFO using a python program

- Create a python program (named 'video_control.py') which will:
- Setup the 4 buttons correctly for detection when pressed
- Connect the following mplayer actions to the buttons:
    - Pause
    - Fast forward 10 seconds
    - Rewind 10 seconds
    - Quit mplayer
- In one command line window on the desktop, start the video_control.py python program.

- In a second command line window, Start an instance of mplayer which will play the video on the piTFT and read commands from the video_fifo

- Test correct button operation. If all is well, button presses on the piTFT buttons should control operation of the video on the piTFT.

This program should be demonstrated on the piTFT. Consider running video_control.py in the background then starting mplayer. Another method would be to start mplayer on the piTFT and run video_control.py using an ssh window from your laptop.

Step 7: bash script

Create a bash script (called 'start_video.sh') to launch mplayer and 'video_control.py'. You should be able to launch this from the command line on the piTFT where the video will begin, and you will be able to control the video with the piTFT buttons.
'start_video' should:

- Start 'video_control.py'
- Note: Consider paths for scripts, and for the FIFO
- Note: Consider foreground or background operation for 'video_control.py'
- Mplayer should be started next to execute on piTFT

At this point, backup your SD card for Lab 1, Week 2

Before completing the lab, demonstrate the following programs for the TA:

- fifo_test.py
- one_button.py
- four_buttons.py
- video_control.py
- start_video.sh

Take the TA through the operation of each piece of software and explain the function of each program indicating any features of the code.