# 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 the grounded mats on the lab bench
- Make sure you ground yourself by staying in contact with the mat.

Personal safety

- Safety goggles are REQUIRED for soldering

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

Notes before you start:

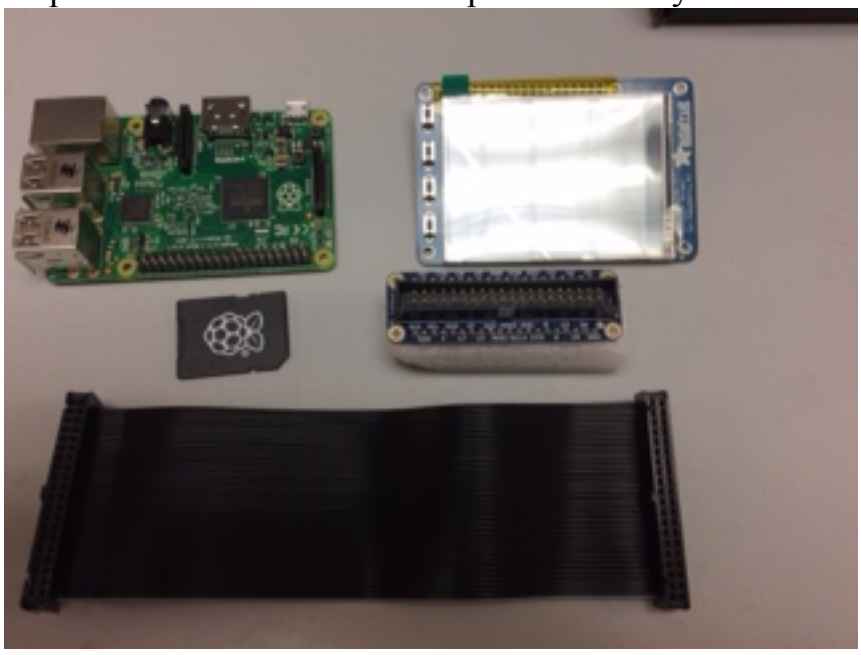
- The R-Pi is powered up at the END of this assembly AFTER you show your assembly to the TA.
- The 40-pin GPIO connectors are NOT keyed. This means they will fit in a number of different orientations. Check that your assembly matches the photos AND check your assembly with the TA before applying power.

Initial assembly and start up:
In the following order, assemble all the components (please don't forget to work on the grounded mat). Here is a photo of the initial system elements:

Assemble your system using the following steps:

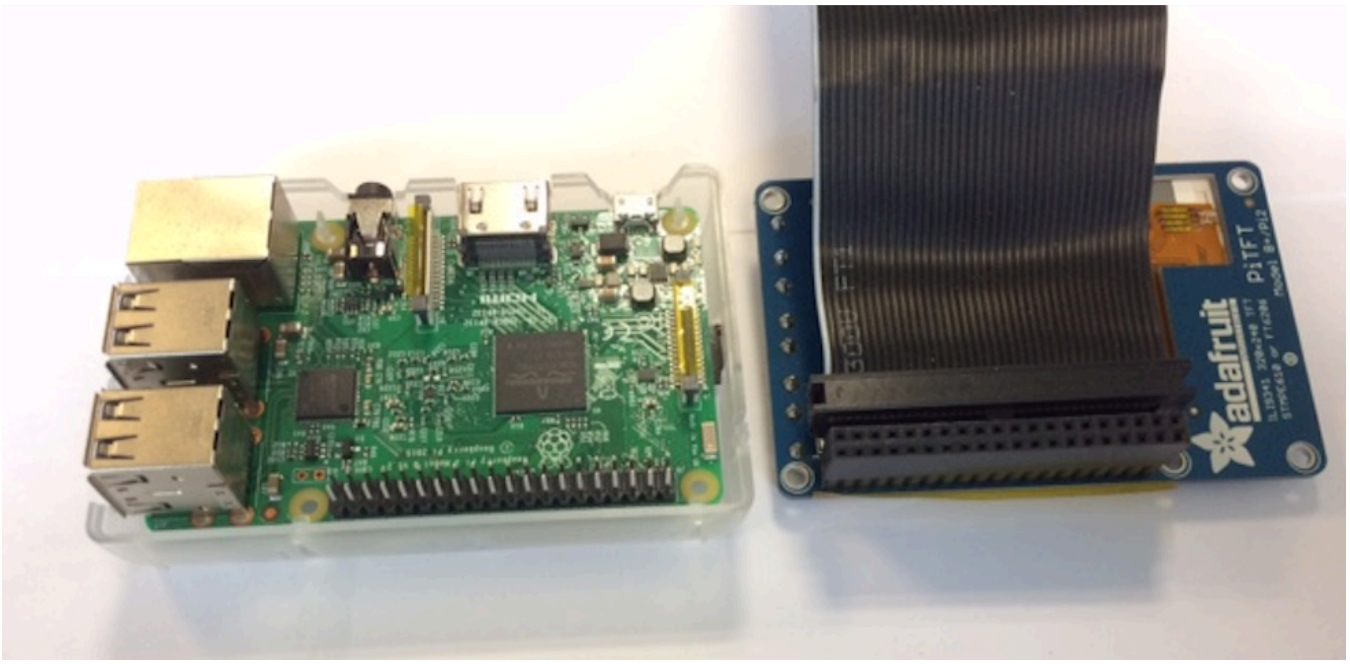
- Unpack the R-Pi
- Install the R-Pi in the bottom of the R-Pi case. To install the RPi in the case, slide one edge under the plastic tabs of the case, then snap the 2 clips through the RPi mounting holes. Note that the top of the case will not be used. The photos below show the RPi and case:

- Unpack the TFT
- Unpack the breakout cable.  A photo of the system elements:



- 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:

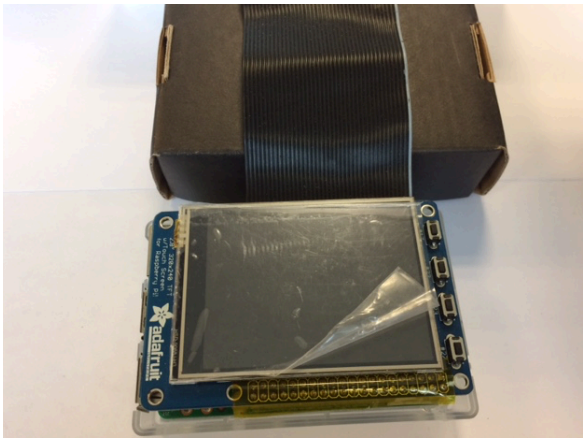

- Plug the TFT into the RPi 40 pin GPIO connector.
  Here is a photo of the piTFT, with the cable correctly plugged, shown next to the RPi.

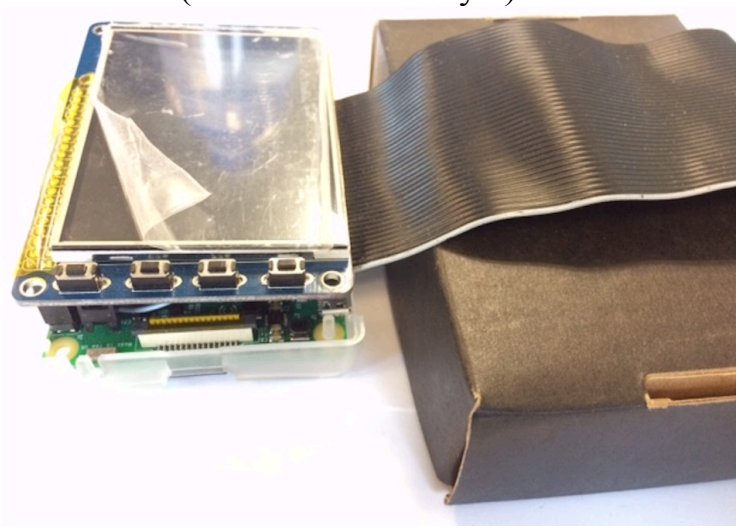



Flip the screen onto the RPi and carefully plug the screen into the 40 pin connector on the RPi. Your completed R-Pi, PiTFT screen, and breakout cable should look like the following photos.

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 (do not install this yet):





- Extract the miniSD Card from the carrier and insert in the bottom slot of the R-Pi. Note the correct position of the mini-SD.
- Plug the display into the HDMI port using the DVI-HDMI adapter and DVI cable.
- Plug into the USB ports:
  - Mouse
  - Keyboard
- Plug in the Ethernet cable

**Before you power-up your system:**

- Show the TA your setup and get a power adapter
- Plug in the mini-USB connector from the power adapter
- Plug adapter into 110V outlet

You must cycle thru display settings using the on-display buttons to make sure ‘DVI’ is selected for display.

You should see the R-Pi booting

**Important Note: 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 all LEDs on the R-Pi stop 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

As part of Homework 1, you should have setup your SD card with the correct version of the Raspbian kernel; We will be using the latest Buster release, 4.14.58.

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

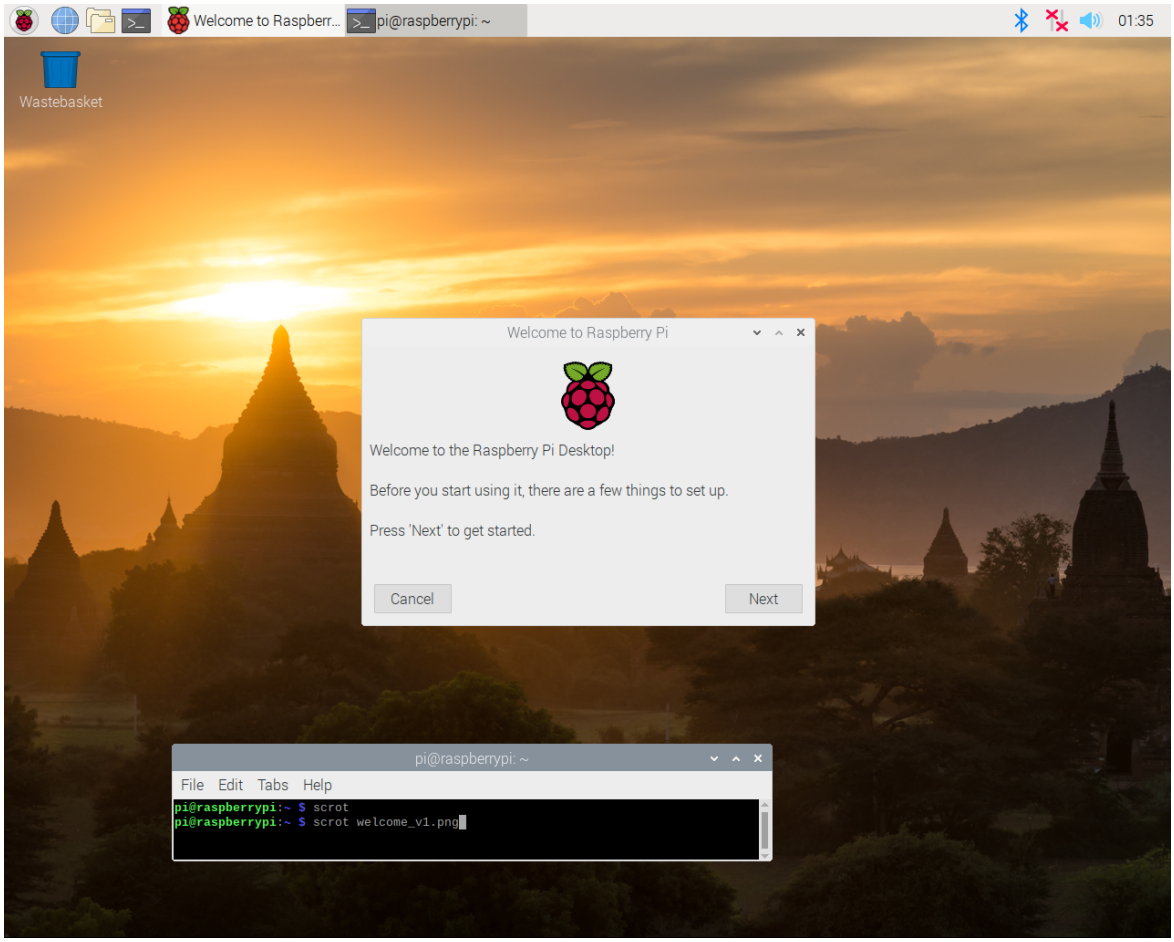
There are some raspi-config details (for reference only) in the following links:

https://www.raspberrypi.org/documentation/configuration/raspi-config.md

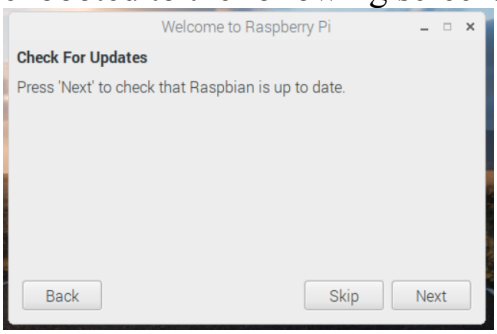
general configuration link:
https://www.raspberrypi.org/documentation/configuration/

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.

An initial install of the Raspbian Buster kernel booted directly to the Raspbian Desktop. DO NOT hit ‘NEXT’ in the prompt that appears on Reboot. Hit CANCEL instead:




Another instance of the kernel booted to the following screen. In this case, hit ‘SKIP’:




**If you see this screen, or a similar ‘helper’ screen, please hit ‘skip’.**

You want to skip any auto-config screens so you will have a better idea of what files are changing to configure the Linux kernel.

Once the system boots, if you have a desktop, find the terminal icon and open a terminal window. If your system boots to a command line interface, proceed with command below.

- 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 4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25
BST 2019 armv7l GNU/Linux
pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

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

- issue the command ‘sudo raspi-config’
- In the ‘boot options’ select ‘Desktop/CLI’ then ‘B1 Console’

This change will boot directly to the console window on the next RPi reboot. Since you are already in raspi-config, continue with the notes below for more configuration.

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

- In ‘Change User Password’, Change the pi user password to something you will remember!
  - **Important Note:** If your keyboard has a keypad, DO NOT use this to enter any Password information

- In ‘Network Options’, select ‘Hostname’: set a unique hostname to ‘netid_netid’ where ‘netid_netid’ are the Cornell netids of the two team members in your group.
- For Localization options:
    - Change locale:
        - Use the Spacebar and tab to navigate
        - De-select ‘en_GB.UTF-8 UTF-8’. Set the Locale ‘en-us.UTF8 UTF8’. When prompted, select ‘None’ for the Default Locale
    - Change Timezone
        - Set the local time to America
        - Set the location to ‘New York’

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

**Restart sudo raspi-confing and continue with localization options:**

- Change Keyboard Layout
    - ‘Generic 101 key PC’ keyboard
    - select ‘Other’, then ‘English US’
    - ‘configuring keyboard-configuration’
        - select ‘English (US)’
        - Select ‘English (US)’ again on the next seceen
        - next, select ‘The default for the keyboard layout’
        - 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

- Interfacing Options: enable ssh
- In the ‘Advanced’ settings
    - A1, expand the file system
    - A4 Audio: Force 3.5 mm jack
- Update: update this 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@raspberrypi:~ $ 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

Run ‘ifconfig –a’ and check the entry for eth0 to determine the Ethernet address; save the Ethernet address so you can log in remotely using putty or a terminal window (for example, iTerm on Mac)

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

- `'sudo apt-get update'`
- `'sudo apt-get 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 Note: 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.

Once upgrade is completed:

## 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 –r 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 RPi-jfs9 4.19.58-v7+ #1245 SMP Fri Jul 12 17:25:51 BST 2019 armv7l
GNU/Linux
```

Note - 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 R-Pi pages

Load an update for the vim editor:

`'sudo apt-get install vim'`

From the ecommand line issue the command 'startx'

Once in the desktop, exit back to the command line window using 'ctrl-ALT-BACKSPACE' which you setup earlier using the raspi-config tool

Some simple checks:

- 'hostname'
- run 'startx' and make sure the desktop starts
- try 'ctrl-alt-backspace' to exit startx
- 'ifconfig –a' to check network connections
- 'cat /proc/cpuinfo' to check the number of cores
- 'time date'
- 'htop'  # Note, exit htop with 'CTRL-C'

On the R-pi, run `'ifconfig -a'`. Note the IP address for the 'eth0' Ethernet adapter. Using this ip address, you can log in remotely using your laptop. An example of the ssh command: `'ssh pi@nnn.mmm.aa.bb'` Where 'pi' is the user name and 'nnn.mmm.aa.bb' is the ip address for the RPi.

From ifconfig –a, record the MAC addresses of the eth0 (wired Ethernet) and wlan0 (WiFi adapter). Using this link:

https://dnsdb.cit.cornell.edu/dnsdb-cgi/mycomputers.cgi

Add in both devices by selecting 'Add Device with no Browser' at the bottom of the page. This will add both network adapters to the list of devices you are using at Cornell, allowing access to features used on the Cornell networks.

Using 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 Ethernet (eth0) connection.

Note: The raspberry Pi does indeed have a built-in WIFI adapter. In general, wired Ethernet will be much faster and is preferred for downloading big kernels, patches, or other large hunks of software.

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 us later……

## TFT Screen Install

In past classes, we have used a detailed set of instructions posted on the Adafruit site to install the piTFT. When Stretch was released (mid-August, 2017), the instructions failed as the techs at Adafruit had not yet had a chance to update the installation instructions. As a result, for the Fall, 2017 version of the class, we downgraded to the last stable version of Jessie to be able to use the piTFT.

This semester, the instructions on the Adafruit site still do not work. However, there have been some discussions on for forums and there seems to be a workable solution to install the piTFT. The following instructions provide details:

**Install software to support the piTFT:**

Run:

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

The python-pip application is used to install python packages on the Pi.
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@RPi-jfs9:~ $ sudo apt-get install -y bc fbi git python-pip python-smbus python-
spidev 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@RPi-jfs9:~ $ sudo apt-get install -y bc fbi git python-pip python-smbus python-
spidev evtest  libts-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.20.1-2).
python-smbus is already the newest version (4.1-1).
python-pip is already the newest version (18.1-5+rpt1).
python-spidev is already the newest version (20170223~145721-2).
The following package was automatically installed and is no longer required:
  rpi.gpio-common
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  evemu-tools ghostscript gsfonts libevemu3 libts0
Suggested packages:
  imagemagick ghostscript-x
The following NEW packages will be installed:
  bc evemu-tools evtest fbi ghostscript gsfonts libevemu3 libts-bin libts0
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.


### Many more lines of installation
```

## Run the command:

```
sudo pip install evdev
```

## Example:

```
pi@RPi-jfs9:~ $ sudo pip install evdev
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting evdev
  Downloading
https://files.pythonhosted.org/packages/fe/6e/3fa84a43571dec4d00dc26befccc9867b6b32
63651531cbc1345f718860f/evdev-1.2.0.tar.gz
Building wheels for collected packages: evdev
  Running setup.py bdist_wheel for evdev ... done
  Stored in directory:
/root/.cache/pip/wheels/05/f9/96/72f2a2385b675af9e80b05582472178948c2d36aab8c0380df
Successfully built evdev
Installing collected packages: evdev
Successfully installed evdev-1.2.0
```

**Add piTFT info to config.txt**

Edit the file /boot/config.txt and add the following lines to the end of the file. 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
[pi2]
device_tree=bcm2710-rpi-3-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft28-resistive,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:

```
[    2.724715] random: systemd: uninitialized urandom read (16 bytes read)
[    2.740661] random: systemd: uninitialized urandom read (16 bytes read)
[    2.746520] systemd[1]: Created slice system-systemd\x2dfsck.slice.
[    2.746806] random: systemd: uninitialized urandom read (16 bytes read)
[    2.748174] systemd[1]: Created slice User and Session Slice.
[    2.748332] systemd[1]: Reached target Swap.
[    2.748441] systemd[1]: Reached target Slices.
[    2.886203] i2c /dev entries driver
[    3.539274] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[    3.670884] systemd-journald[108]: Received request to flush runtime journal
from PID 1
[    4.279025] vc_sm_cma: module is from the staging directory, the quality is
unknown, you have been warned.
[    4.282061] bcm2835_vc_sm_cma_probe: Videocore shared memory driver
[    4.282086] [vc_sm_connected_init]: start
[    4.285890] media: Linux media interface: v0.10
[    4.287583] [vc_sm_connected_init]: installed successfully
[    4.323770] videodev: Linux video capture interface: v2.00
[    4.339235] stmpe-spi spi0.1: stmpe610 detected, chip id: 0x811
[    4.354872] bcm2835_mmal_vchiq: module is from the staging directory, the
quality is unknown, you have been warned.
[    4.375323] stmpe-gpio stmpe-gpio: DMA mask not set
[    4.380909] bcm2835_v4l2: module is from the staging directory, the quality is
unknown, you have been warned.
[    4.383874] bcm2835_codec: module is from the staging directory, the quality is
unknown, you have been warned.
[    4.413495] bcm2835-codec bcm2835-codec: Device registered as /dev/video10
[    4.413509] bcm2835-codec bcm2835-codec: Loaded V4L2 decode
[    4.432301] bcm2835-codec bcm2835-codec: Device registered as /dev/video11


[    6.206084] stmpe-ts stmpe-ts: DMA mask not set
[    6.219502] input: stmpe-ts as
/devices/platform/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-ts/input/input2
[    6.219849] fbtft: module is from the staging directory, the quality is unknown,
you have been warned.
[    6.272523] fb_ili9340: module is from the staging directory, the quality is
unknown, you have been warned.
[    6.277651] fbtft_of_value: buswidth = 8
```

```
[    6.277704] fbtft_of_value: debug = 0
[    6.277712] fbtft_of_value: rotate = 90
[    6.277721] fbtft_of_value: fps = 30
[    6.465585] graphics fb1: fb_ili9340 frame buffer, 320x240, 150 KiB video
memory, 4 KiB buffer memory, fps=33, spi0.0 at 64 MHz
[    7.293874] random: crng init done
[    7.293891] random: 7 urandom warning(s) missed due to ratelimiting
[    7.429022] uart-pl011 3f201000.serial: no DMA platform data
[    7.701356] 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"
```

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

Run:

```
sudo modprobe stmpe_ts
```

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

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@RPi-jfs9:~ $ ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Aug  8 14:14 /dev/input/touchscreen -> event2
```

**Use evtest to verify touchscreen operation**

With the above changes, the evtest utility may be used to check the operation of the touchscreen. Run:

```
sudo evtest /dev/input/touchscreen
```

Once evtest runs, tap the screen several times and you should see data associated with each tap.

Here is an example of evtest starting:

```
pi@RPi-jfs9:~ $ 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)
```

And here is an example of information from PiTFT screen taps once evtest is running:

```
Event: time 1565288113.825992, type 3 (EV_ABS), code 0 (ABS_X), value 2064
Event: time 1565288113.825992, type 3 (EV_ABS), code 1 (ABS_Y), value 2306
Event: time 1565288113.825992, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 109
Event: time 1565288113.825992, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1565288113.825992, -------------- SYN_REPORT ------------
Event: time 1565288113.833006, type 3 (EV_ABS), code 0 (ABS_X), value 2056
Event: time 1565288113.833006, type 3 (EV_ABS), code 1 (ABS_Y), value 2283
Event: time 1565288113.833006, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 128
Event: time 1565288113.833006, -------------- SYN_REPORT ------------
Event: time 1565288113.840022, type 3 (EV_ABS), code 0 (ABS_X), value 2059
Event: time 1565288113.840022, type 3 (EV_ABS), code 1 (ABS_Y), value 2302
Event: time 1565288113.840022, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 123
Event: time 1565288113.840022, -------------- SYN_REPORT ------------
Event: time 1565288113.847018, type 3 (EV_ABS), code 0 (ABS_X), value 2060
Event: time 1565288113.847018, type 3 (EV_ABS), code 1 (ABS_Y), value 2296
Event: time 1565288113.847018, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 119
```

Note that ‘ctrl-c’ followed by the ‘esc’ key exits the evtest script

**Set initial piTFT calibration**

Next, setup information for initial calibration of the pitft. Edit the file /etc/pointercal (this will create a new file…. Note, you need to used sudo to edit the file) and add the line:

```
33 -5782 21364572 4221 35 -1006432 65536
```

There is a space between each of these numbers.

**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:

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

just after the 'rootwait' entry. Notes on this change:

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

```
pi@RPi-jfs9:~ $ cat /boot/cmdline.txt
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=3c4c1ba5-02
rootfstype=ext4 elevator=deadline 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="6x12"
```

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

```
pi@RPi-jfs9:~ $ 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="6x12"

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@RPi3-jfs9:~ $ 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 of the install steps that previously existed at the Adafruit site.

One set of original Adafruit instructions that remains working describes how to run a video on the piTFT. These instructions are here:

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

Instructions on this page are correct. Please work through these instructions for playing a sample video. In short, the instructions describe:

- Load the sample video
- Install mplayer
- Test video playback on the piTFT

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 for a set) or through headphones. Some useful references:

Run 'mplayer –ao help' to list all possible audio drivers

To boost the audio volume run:

```
amixer sset PCM,0 100%
```

At this point, the TFT has been successfully configured

## Test from several consoles

- Test by starting mplayer directly on piTFT
- Run 'startx' to run desktop on external display. Start mplayer using a console window
- Remote login using your laptop. Start the mplayer video remotely

Are there any problems with any of these experiments (for example, does the video always run on the piTFT? Is the audio synchronized correctly?) Let the video run and check as it progresses. Document your findings and discuss any issues with a TA.

## Stop and Backup your SD card

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

**Once your SD card is backed up, please check 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
- Switch SD cards and launch mplayer from the switched card using the piTFT console window (demonstrate two running SD cards)

Demo at Start of Lab1, week 2:

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 week1 and week2, 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.

## 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.

## 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 number of 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 blackboard 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 link describes mplayer commands which may be sent using a fifo.  Search for examples of pause and quit in this document and check for additional useful mplayer 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 blackboard 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 script

- Create a python routine (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
- Start an instance of mplayer and test for correct button operation

This program may be demonstrated on the piTFT (consider running video_control.py in the background then starting mplayer) or by using ctrl-alt-F2 to open a second console on the monitor, stating video_control, then mplayer.

Step 7: bash script

Create a bash script (called 'start_video') 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, Save an image of your SD card for Lab 1, Part B

---

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

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