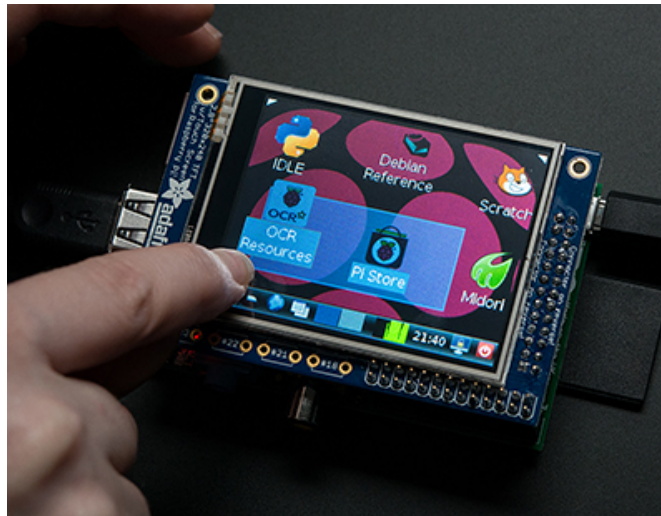


## Adafruit PiTFT - 2.8" Touchscreen Display for Raspberry Pi

Created by lady ada



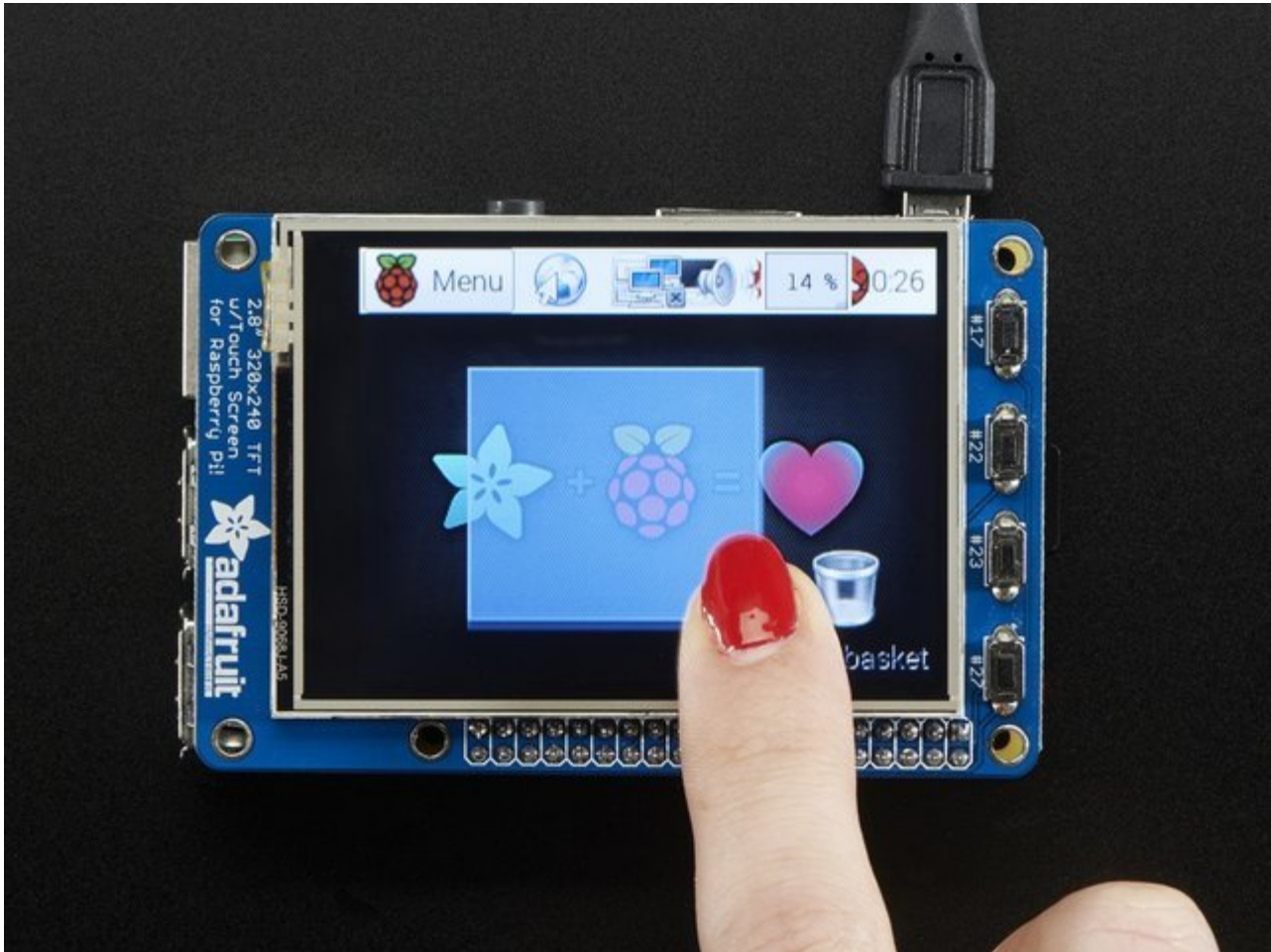
Last updated on 2017-09-22 05:38:06 PM UTC

## Guide Contents

Guide Contents	2
Overview	4
Original PiTFT	4
PiTFT Plus	5
Assembly	9
Easy Install	16
Ready to go image	16
DIY Installer script	16
Step 1. Expand Filesystem	17
Step 2. Install new Kernel	17
Step 3. Enable & Configure the PiTFT	19
Detailed Installation	21
Before you start	22
Download & Install Kernel	22
Resistive Touchscreen Manual Install & Calibrate	30
Setting up the Touchscreen	30
Running evtest	32
AutoMagic Calibration Script	33
Manual Calibration	35
X Calibration	37
Console Configuration	40
Turn off Console Blanking	43
Raspbian Jessie	43
Raspbian Wheezy	43
Userspace Tools	45
Download, Test and Install	45
Resistive Touchscreen Support	46
HELP! (FAQ)	48
Playing Videos	55
How To Play Videos	55
Converting/Resizing Videos	57

Displaying Images	61
Using FBCP	63
Backlight Control	64
PWM Backlight Control with GPIO 18	64
On / Off Using STMPE GPIO	65
For older versions of PiTFT Kernel	65
PiTFT PyGame Tips	67
Install pip & pygame	67
Ensure you are running SDL 1.2	68
Extras!	71
Tactile switch as power button	71
Making it easier to click icons in X	72
Boot to X Windows on PiTFT	73
Right-click on a touchscreen	74
Gesture Input	76
Installation	76
Usage	76
Downloads	80
2.8" PiTFT Plus Schematic & Layout	80
PiTFT 3.2" Plus Schematic	82
Original PiTFT 2.8" Schematic & Layout	82

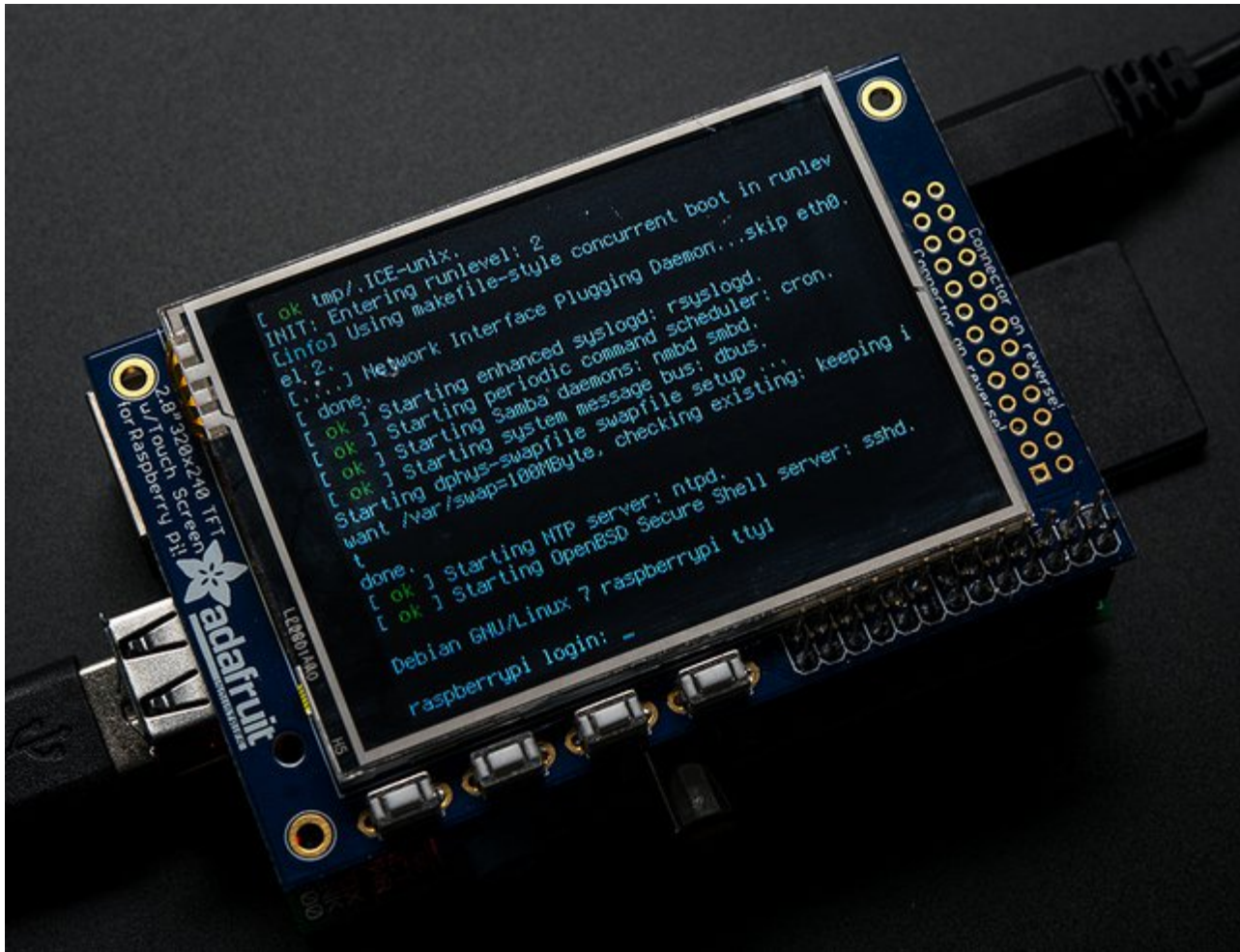
# Overview



Is this not the cutest little display for the Raspberry Pi? It features a 2.8" display with 320x240 16-bit color pixels and a resistive touch overlay. The plate uses the high speed SPI interface on the Pi and can use the mini display as a console, X window port, displaying images or video etc. Best of all it plugs right in on top!

## Original PiTFT

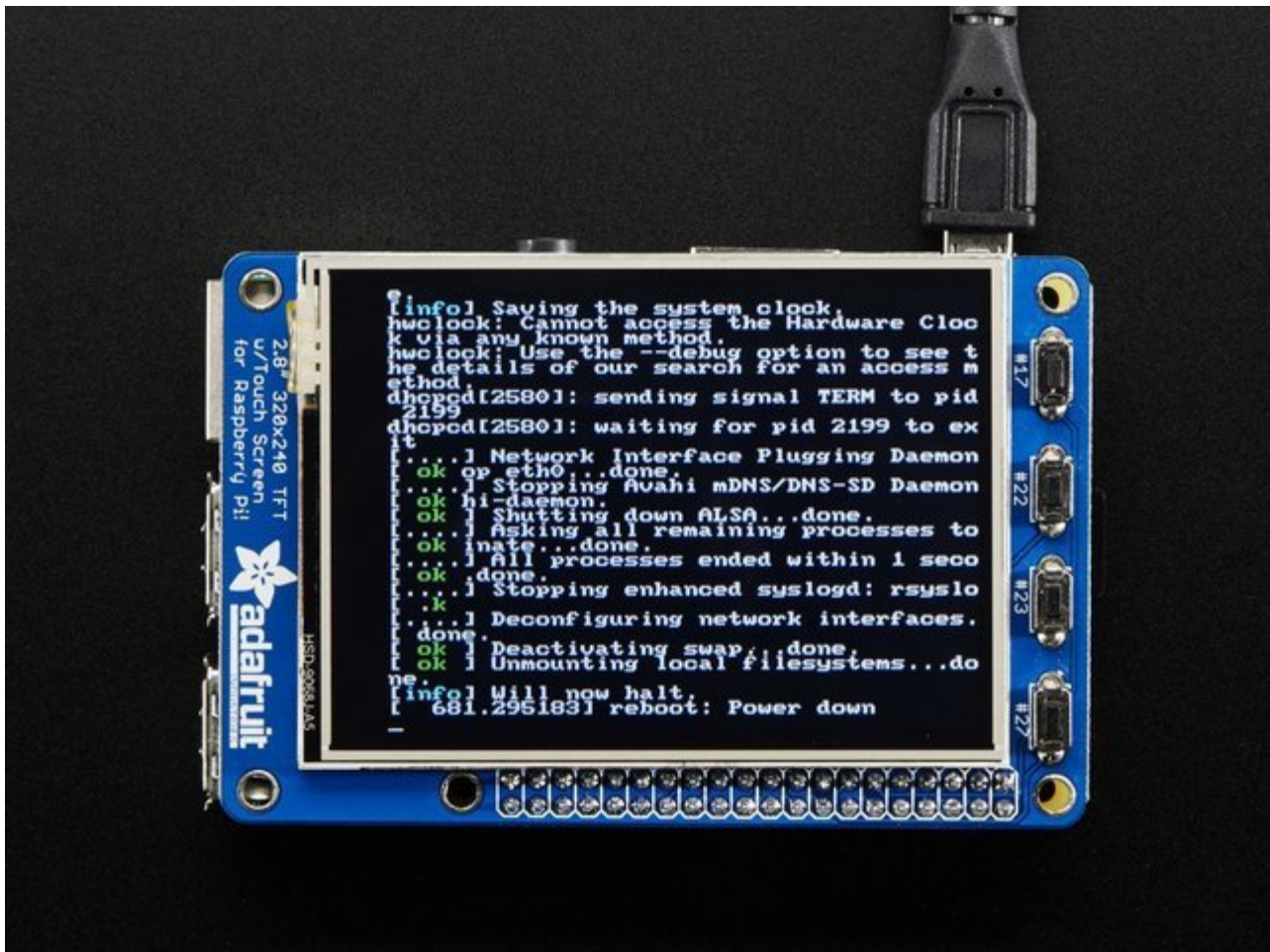
The original version PID 1601 is designed to fit nicely onto the Pi Model A or B but also works perfectly fine with the Pi Zero, Pi 2, Pi 3 or Pi 1 Model A+ or B+ as long as you don't mind the PCB overhangs the USB ports by 5mm



## PiTFT Plus

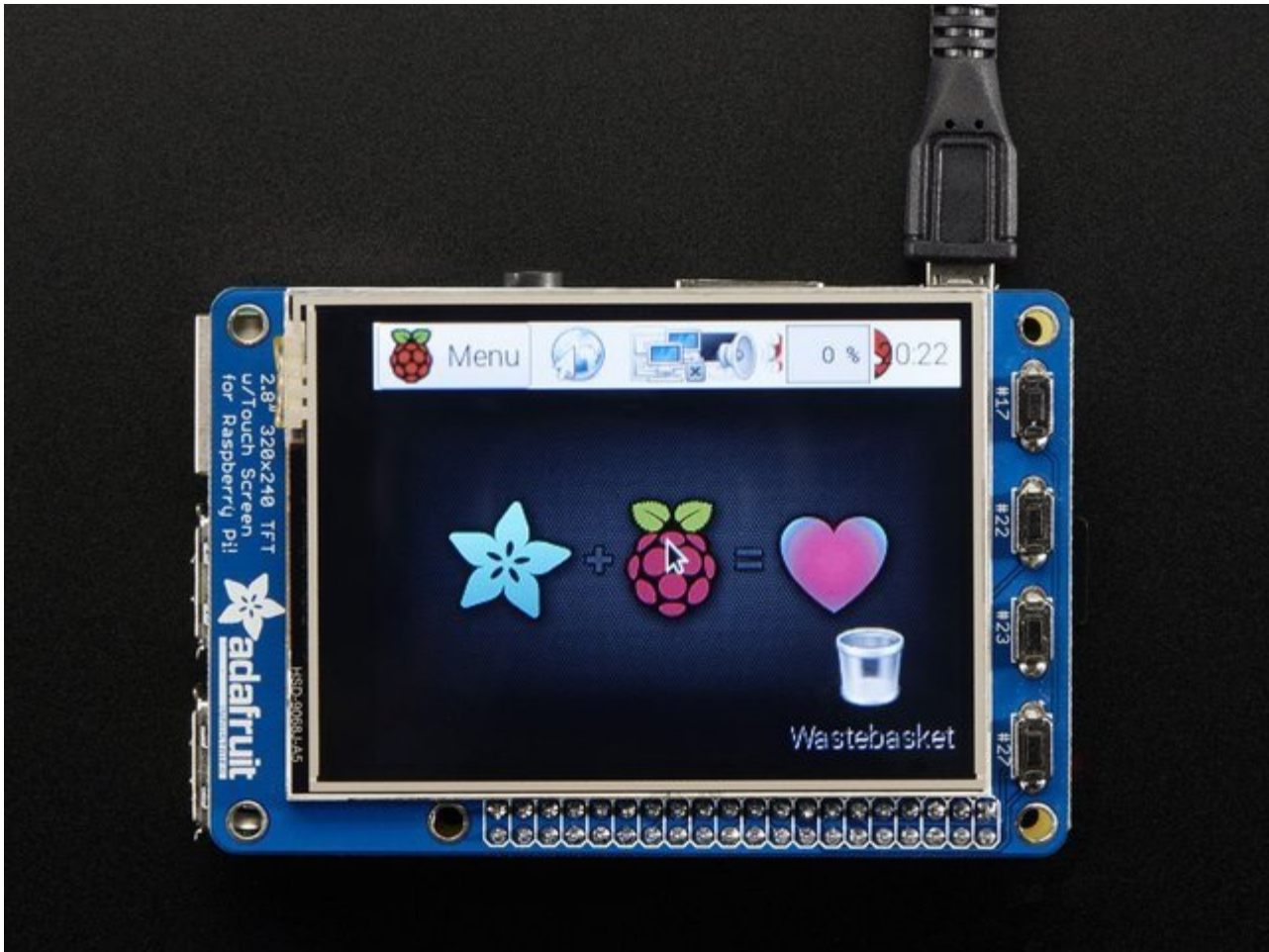
The newer PiTFTs are updated to fit perfectly onto the Pi Zero, Pi 3, Pi 2 or Model A+, B+!  
(Any Pi with a 2x20 connector) **Not for use with an old Pi 1 with 2x13 connector**



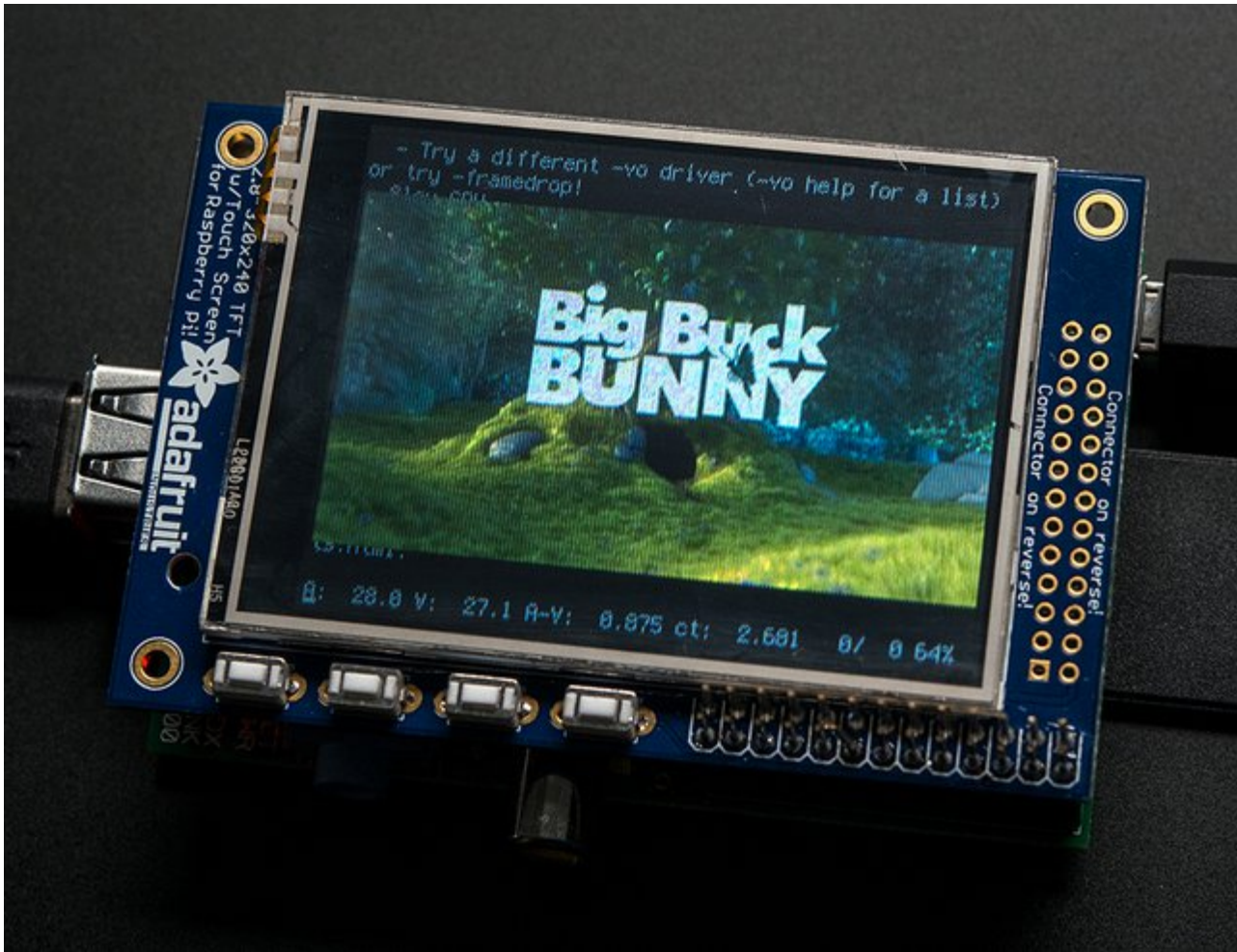


This design uses the hardware SPI pins (SCK, MOSI, MISO, CE0, CE1) as well as GPIO #25 and #24. All other GPIO are unused. Since we had a tiny bit of space, there's 4 spots for optional slim tactile switches wired to four GPIOs, that you can use if you want to make a basic user interface. For example, you can use one as a power on/off button.

We bring out GPIO #23, #22, #21, and #18 to the four switch locations!



To make it super easy for use: we've created a custom kernel package based off Notro's awesome framebuffer work, so you can install it over your existing Raspbian (or derivative) images in just a few commands.

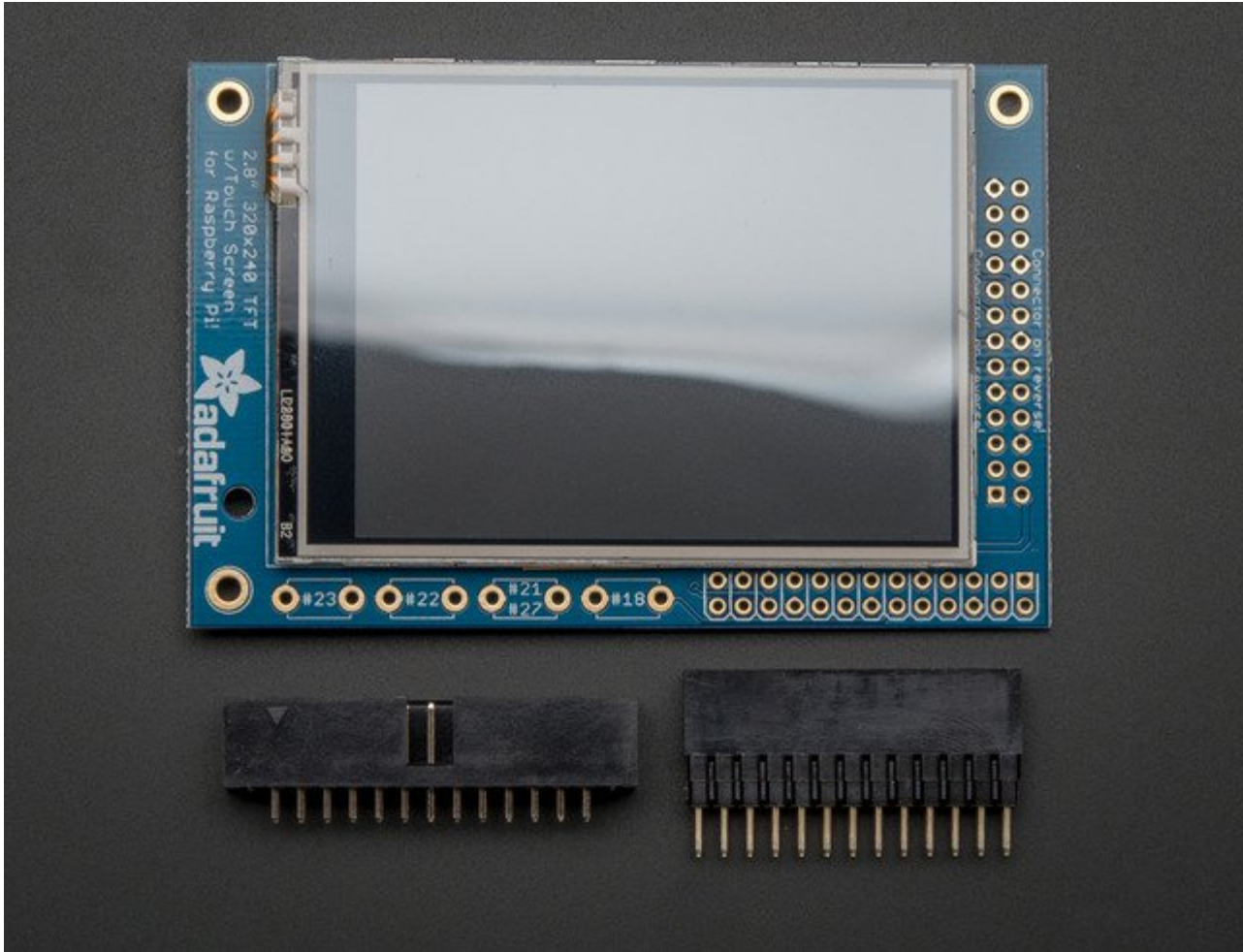


This tutorial series shows you how to install the software, as well as calibrate the touchscreen, splay videos, display images such as from your PiCam and more!



# Assembly

This tutorial page is for PiTFT that came as a kit. If your PiTFT is already assembled, skip this step!



Before you start check that you have the parts you need: an assembled PiTFT plate with the 2.8" screen, extra tall female header and the 2x13 IDC socket. Note that it is normal for the screen to be 'loose' - this is so its easier for you to solder the connector on!



Check also on the back that the TFT is attached and that the flex connector is seated into the onboard FPC socket.



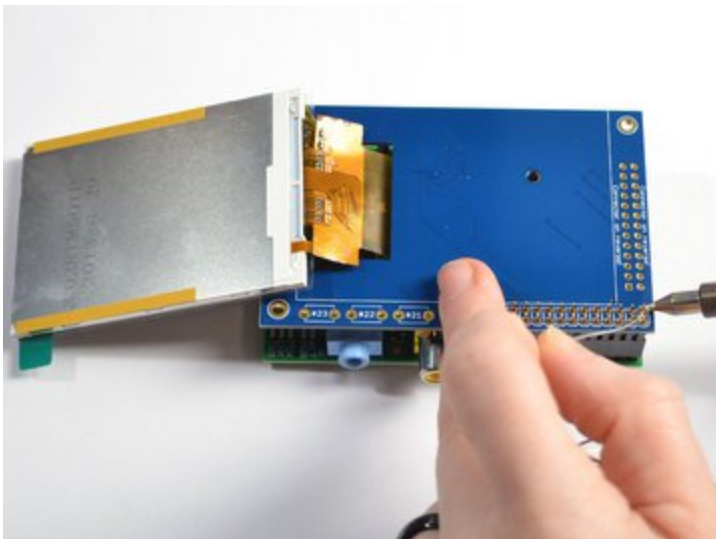
The easiest way to attach the header is if you have a Raspberry Pi as a 'stand' - make sure its powered off & unplugged!



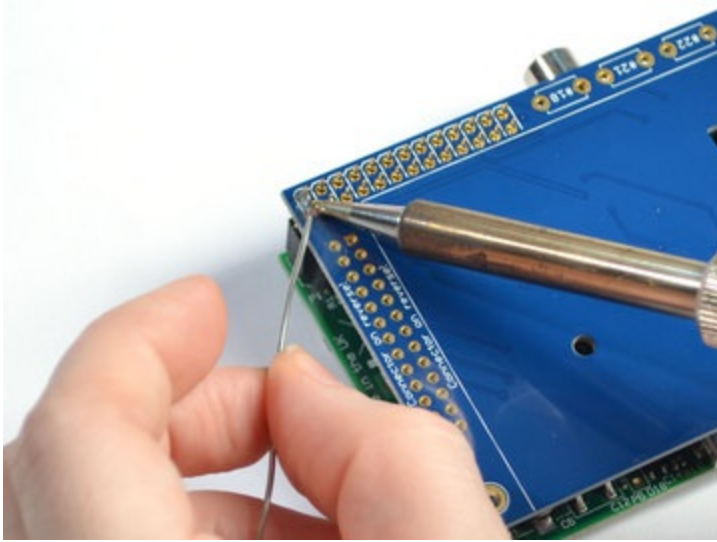
Plug the extra tall female header into the GPIO port on the Pi as shown. Make sure its seated nice and flat



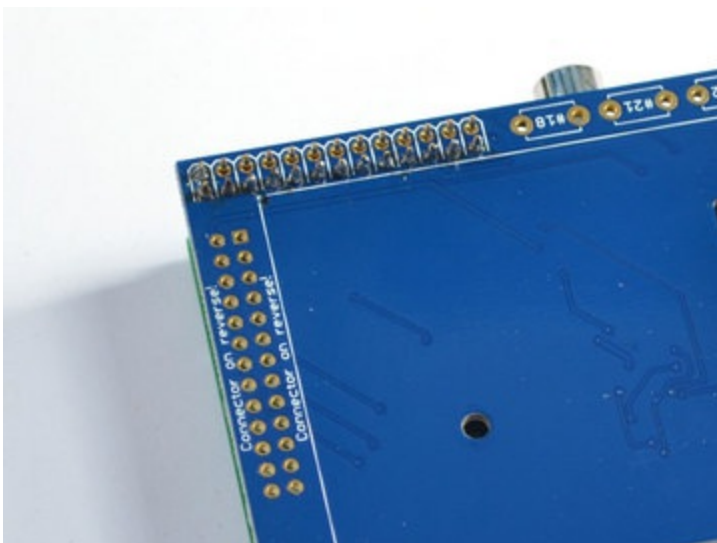
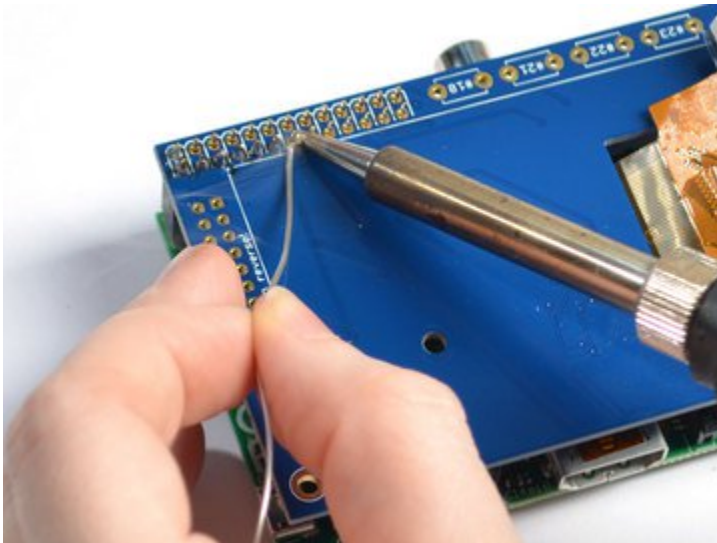
Place the PiTFT shield on top so all the pins stick through the connector on the side. Gently flip the TFT so its off to the side and wont be in your way while you solder







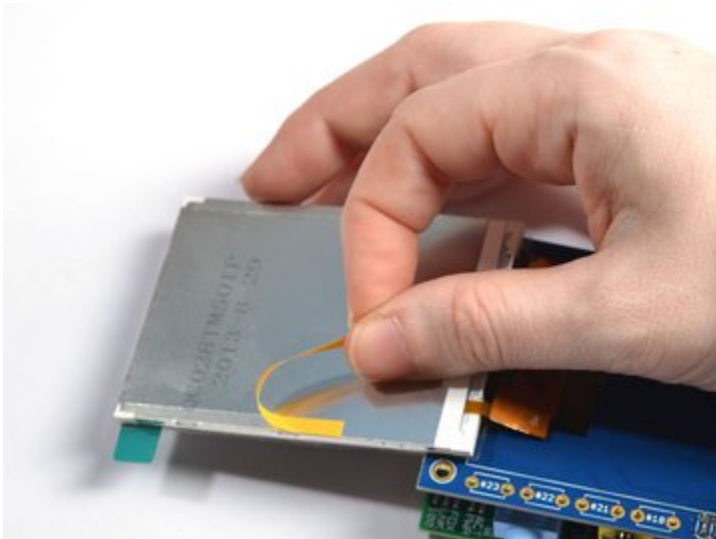
Heat up your soldering iron, and grab some solder. Start by tack-soldering one of the corners while pressing on the plate to make it sit flat. Once you have one or two pins done you can continue to solder each of the pins.





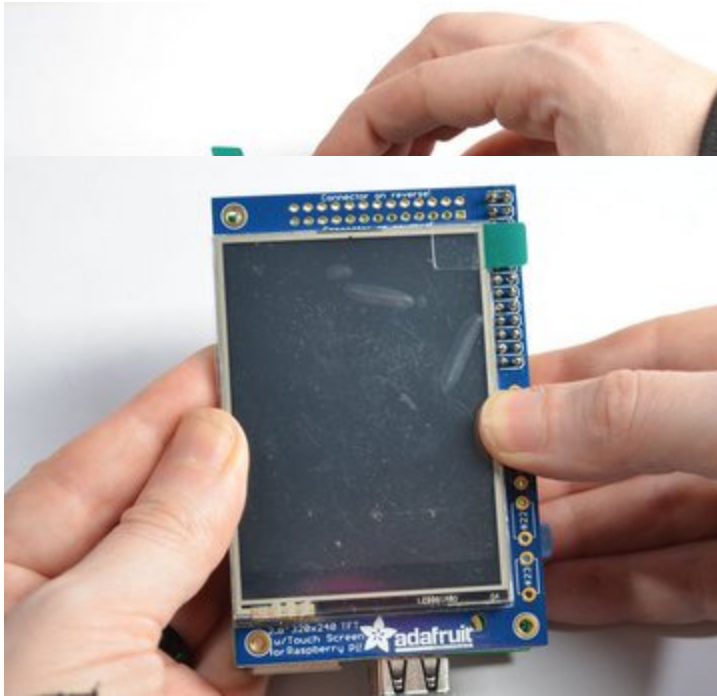


Before attaching the display, check that all the pins are soldered nicely and there's no bridging, cold solder, shorts, or unsoldered pins.

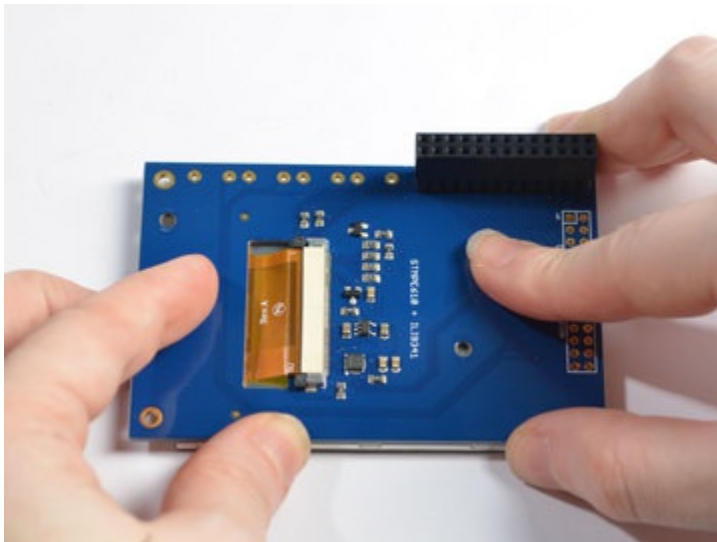


Now we can attach the screen. Remove the two thin tape cover strips.

Line up the screen on the white outline, make sure there's some space from the header you just soldered in and the metal sides of the screen. As long as you don't really press down on the screen you can reposition it once or twice.



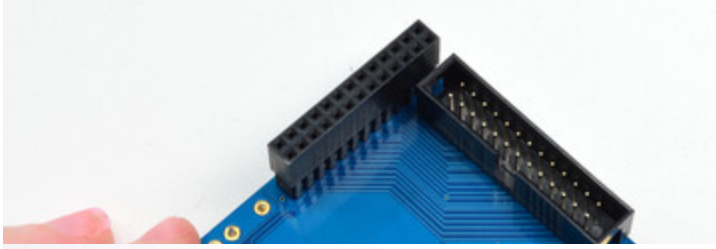
Once you have the screen so it is definitely not touching the header, you can gently press on the sides to secure the tape.



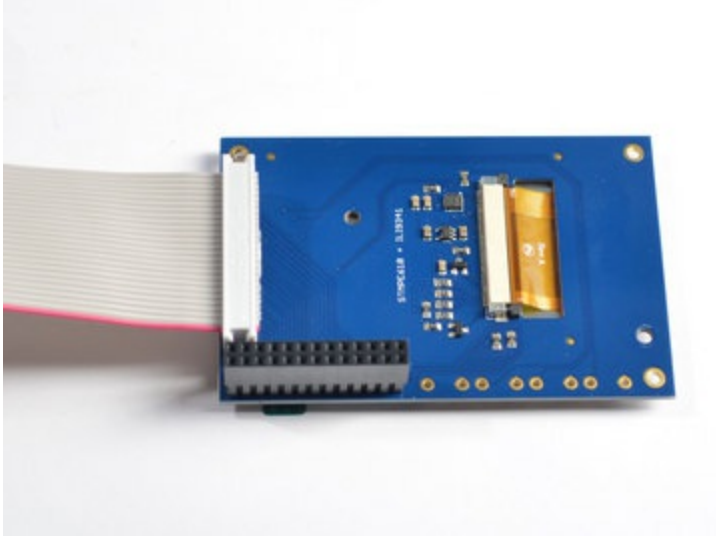
If the protective plastic cover is still on the screen you can press it against a clean table from above. That way you will really securely attach it!

If you want to attach an Adafruit Cobbler or similar, you can solder in the optional 2x13 IDC on the **bottom** of the screen as shown here. This will keep the top side clean and flat. Solder in all 26 pins

The picture shows a 2x13 male header. We've since updated this



product to include an IDC socket so it's easier to add a cobbler. Both will work, though!



You can attach a 26-pin IDC cable just make sure the pin 1 indicator is on the right as indicated in this photo - there's also a #1 marking on the PCB!

# Easy Install

The PiTFT requires kernel support and a couple other things to make it a nice stand-alone display. We have a detailed step-by-step setup for hackers who want to tweak, customize or understand the PiTFT setup. If you just want to get going, check out the following for easy-install instructions!

## Ready to go image

If you want to start with a fresh image, we have two for Raspbian. There's the larger 'classic Jessie' image that will boot into X by default, and requires a 8G image, it has a lot more software installed. There's also the smaller 'Jessie Lite' that will boot into the command line, and can be burned onto a 2G card! Click below to download and install into a new SD card.

[Unzip and follow the classic SD card burning tutorials \(https://adafru.it/aMW\)](https://adafru.it/aMW)

These images are customized for the RESISTIVE touch 2.8" PiTFT, also known as PID #1601 and #2298 or the Resistive 2.4" HAT, a.k.a PID #2455 - These are not for use with 3.5" PiTFT or Capacitive Touch PiTFT

[Download Jessie-based PiTFT 2.4", 2.8" and 3.2" Resistive Image for Pi 1, 2, 3, Zero \(Sept 23, 2016\)](https://adafru.it/s7f)

<https://adafru.it/s7f>

[Download Jessie Lite-based PiTFT 2.4", 2.8" and 3.2" Resistive Image for Pi 1, 2, 3, Zero \(Sept 23, 2016\)](https://adafru.it/s7A)

<https://adafru.it/s7A>

Previous images:

- [Raspbian Jessie 2016/03/25-based image \(https://adafru.it/mA9\)](https://adafru.it/mA9)
- [Raspbian Jessie Lite 2016/03/25-based image \(https://adafru.it/mAa\)](https://adafru.it/mAa)
- [Raspbian Jessie 2015/09/24-based image \(https://adafru.it/iDA\)](https://adafru.it/iDA)
- [Raspbian Wheezy 2015/09/09-based image \(https://adafru.it/idJ\)](https://adafru.it/idJ)
- [Raspbian 2014/06/20-based image \(https://adafru.it/dSM\)](https://adafru.it/dSM)
- [Raspbian 2014/09/09-based image \(https://adafru.it/e12\)](https://adafru.it/e12)

## DIY Installer script

If you don't want to download an image, you can run our installation package helper from



inside your existing Raspbian install. It will download the kernel add-ons, and configure your Pi for PiTFT joy

[The helper is available for perusal here \(https://adafru.it/eIn\)](https://adafru.it/eIn) if you are interested in how it works

## Step 1. Expand Filesystem

Start by expanding the filesystem **This is required!!!**

```
sudo raspi-config  
(expand filesystem)  
sudo reboot
```

## Step 2. Install new Kernel

Then, once the filesystem is expanded, download and install the new kernel by running the following commands:

```
curl -SLs https://apt.adafruit.com/add-pin | sudo bash  
sudo apt-get install raspberrypi-bootloader adafruit-pitft-helper raspberrypi-kernel
```

and type **y** (yes) when prompted

The first command adds **apt.adafruit.com** to your repository list, so you can grab code directly from adafruit's servers

```
pi@raspberrypi ~ $ curl -SLs https://apt.adafruit.com/add-pin | sudo bash
```

The next line does the actual download and installation, it'll take a while because there's a lot of software to replace for PiTFT support.

```
pi@raspberrypi ~/Adafruit-Occidentalis $ sudo apt-get install raspberrypi-bootloader
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
The following packages will be upgraded:
  libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0 raspberrypi-bootloader
5 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
Need to get 61.5 MB of archives.
After this operation, 12.7 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

It's normal for the Pi to pause and/or take a while at this step for many minutes, there's a lot of kernel software to replace

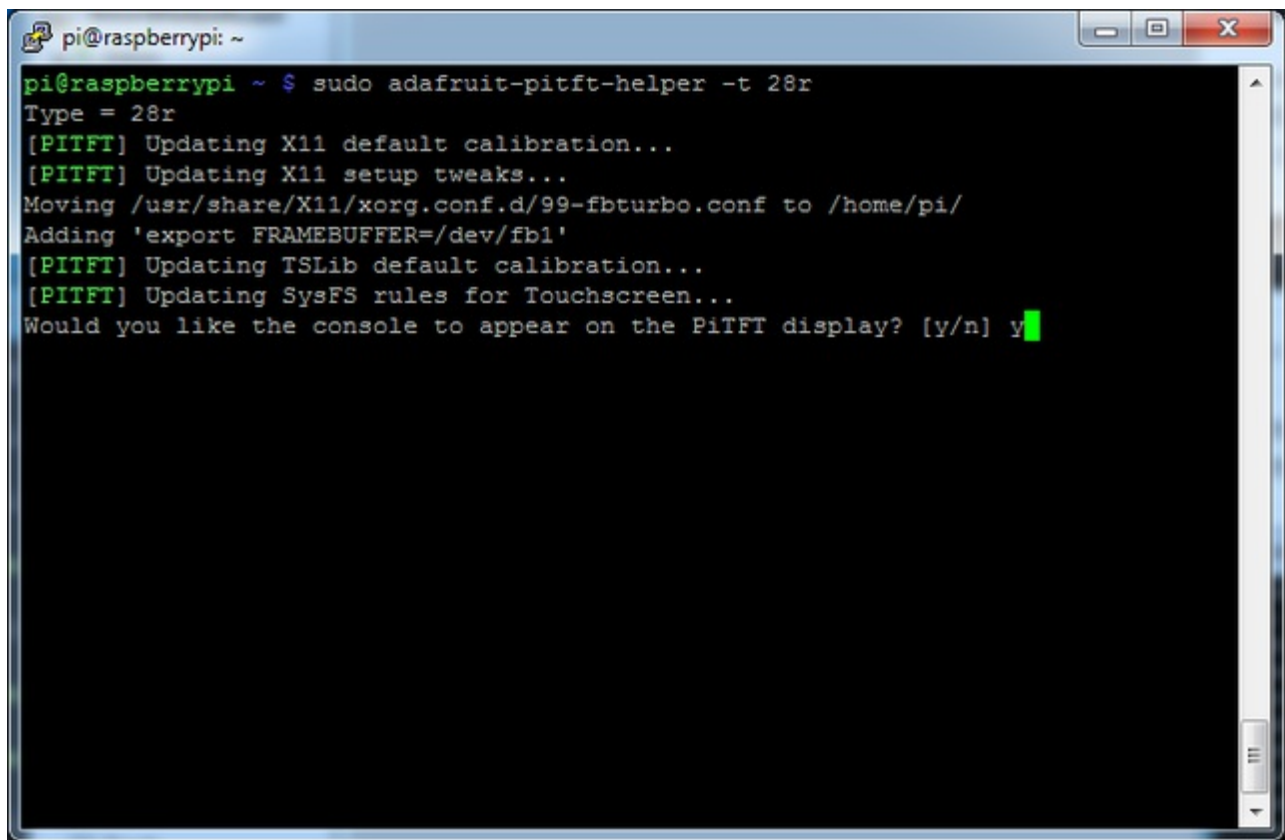
## Step 3. Enable & Configure the PiTFT

OK now the kernel and helper are installed, all you have to do is run the helper which will configure the kernel device tree overlays and add the few configurations to make the console show up, etc.

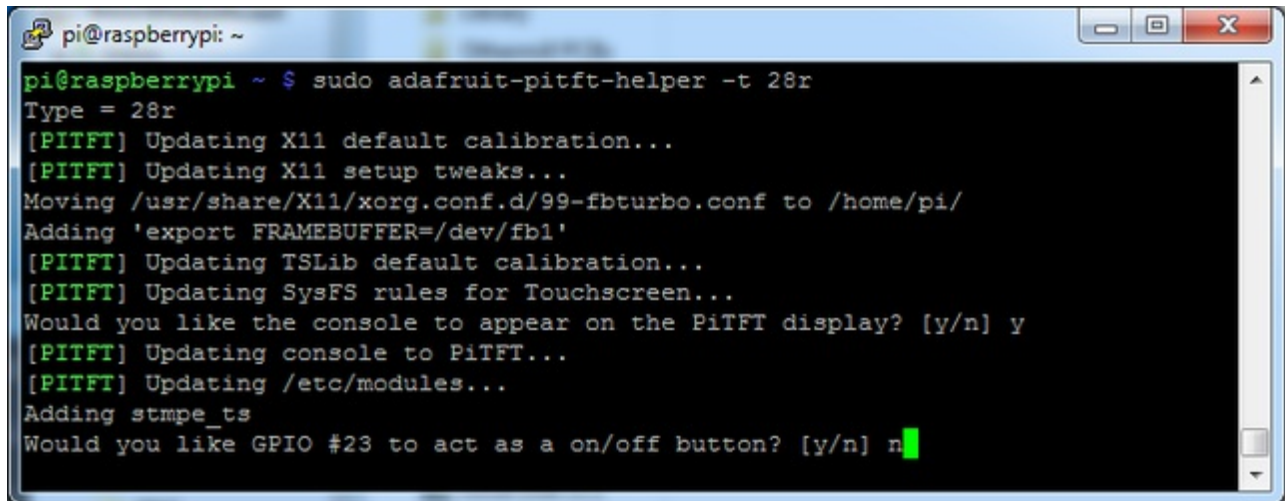
```
sudo adafruit-pitft-helper -t 28r
```

This will install the "2.8 Resistive" type of PiTFT into the current install. This is the same as the 3.2" and 2.4" Resistive screen too (same resolution, pinout, etc.)

At the end you will be prompted on whether you want the text console to appear on the PiTFT. Answer Y or N depending on your personal desires!

A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The terminal shows the command 'sudo adafruit-pitft-helper -t 28r' being executed. The output includes: 'Type = 28r', '[PITFT] Updating X11 default calibration...', '[PITFT] Updating X11 setup tweaks...', 'Moving /usr/share/X11/xorg.conf.d/99-fbturbo.conf to /home/pi/', 'Adding \'export FRAMEBUFFER=/dev/fb1\'', '[PITFT] Updating tslib default calibration...', '[PITFT] Updating SysFS rules for Touchscreen...', and a prompt 'Would you like the console to appear on the PiTFT display? [y/n] y' with a green cursor on the 'y'.

You will also be prompted on whether you want one of the tactile buttons to act as an 'on off' switch. Answer Y or N depending on your personal desires!

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The terminal shows the command 'sudo adafruit-pitft-helper -t 28r' being executed. The script performs several updates: X11 default calibration, X11 setup tweaks (moving a config file and adding an export), TSLib default calibration, and SysFS rules for the touchscreen. It then asks if the console should appear on the PiTFT display, which is answered 'y'. Next, it updates the console to PiTFT and updates /etc/modules. It adds 'stmpe\_ts' and asks if GPIO #23 should act as a button, which is answered 'n'.

```
pi@raspberrypi ~ $ sudo adafruit-pitft-helper -t 28r
Type = 28r
[PITFT] Updating X11 default calibration...
[PITFT] Updating X11 setup tweaks...
Moving /usr/share/X11/xorg.conf.d/99-fbturbo.conf to /home/pi/
Adding 'export FRAMEBUFFER=/dev/fb1'
[PITFT] Updating TSLib default calibration...
[PITFT] Updating SysFS rules for Touchscreen...
Would you like the console to appear on the PiTFT display? [y/n] y
[PITFT] Updating console to PiTFT...
[PITFT] Updating /etc/modules...
Adding stmpe_ts
Would you like GPIO #23 to act as a on/off button? [y/n] n
```

Thats it!

Run **sudo reboot** to try out your fancy new PiTFT :)



# Detailed Installation

If you've grabbed our Easy Install image, or use the script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the kernel install

In the next few steps we'll cover the **detailed** installation procedure. Chances are, you should grab the Easy Install image or script. If you have some interest in the details of how we install the PiTFT setup, read on!



In order to add support for the 2.4" or 2.8" TFT and touchscreen, we'll need to install a new Linux Kernel. Lucky for you, we created a kernel package that you can simply install *over* your current Raspbian (or Raspbian-derived) install instead of needing a whole new image. This makes it easier to keep your install up-to-date.

To use our kernel .deb files you must be using Raspbian or a derivative. This wont work with

Arch or other Linux flavors. As Raspbian is the official OS for the Pi, that's the only Linux we will support! [Others can recompile their own kernel using our patchfile \(https://adafru.it/cY2\)](https://adafru.it/cY2), but we have no tutorial or support or plans for such.

## Before you start

You'll need a working install of Raspbian with network access. [If you need help getting that far, check out our collection of Pi tutorials \(https://adafru.it/aWq\)](https://adafru.it/aWq).

We'll be doing this from a console cable connection, but you can just as easily do it from the direct HDMI/TV console or by SSH'ing in. Whatever gets you to a shell will work!

Also, run **sudo apt-get update** !

To run these all the setup and config commands you'll need to be logged into a proper Terminal - use ssh, a console cable, or the main text console (on a TV). The WebIDE console may not work.

## Download & Install Kernel

The only way we're distributing the PiTFT kernel packages right now is thru apt.adafruit.com so you'll still need to run:

```
curl -SLs https://apt.adafruit.com/add-pin | sudo bash
```

To add apt.adafruit.com to your list of software sources

```
pi@raspberrypi ~ $ curl -SLs https://apt.adafruit.com/add-pin | sudo bash
```

Then install the kernel with

```
sudo apt-get install raspberrypi-bootloader
```

This will take a up to 20 minutes so go make a sandwich or coffee. It takes longer than it used to because there's now 2 kernels (v6 and v7 arm) and 2 kernel module directories.

Don't use rpi-update!

```
pi@raspberrypi ~/Adafruit-Occidentalis $ sudo apt-get install raspberrypi-bootloader
```

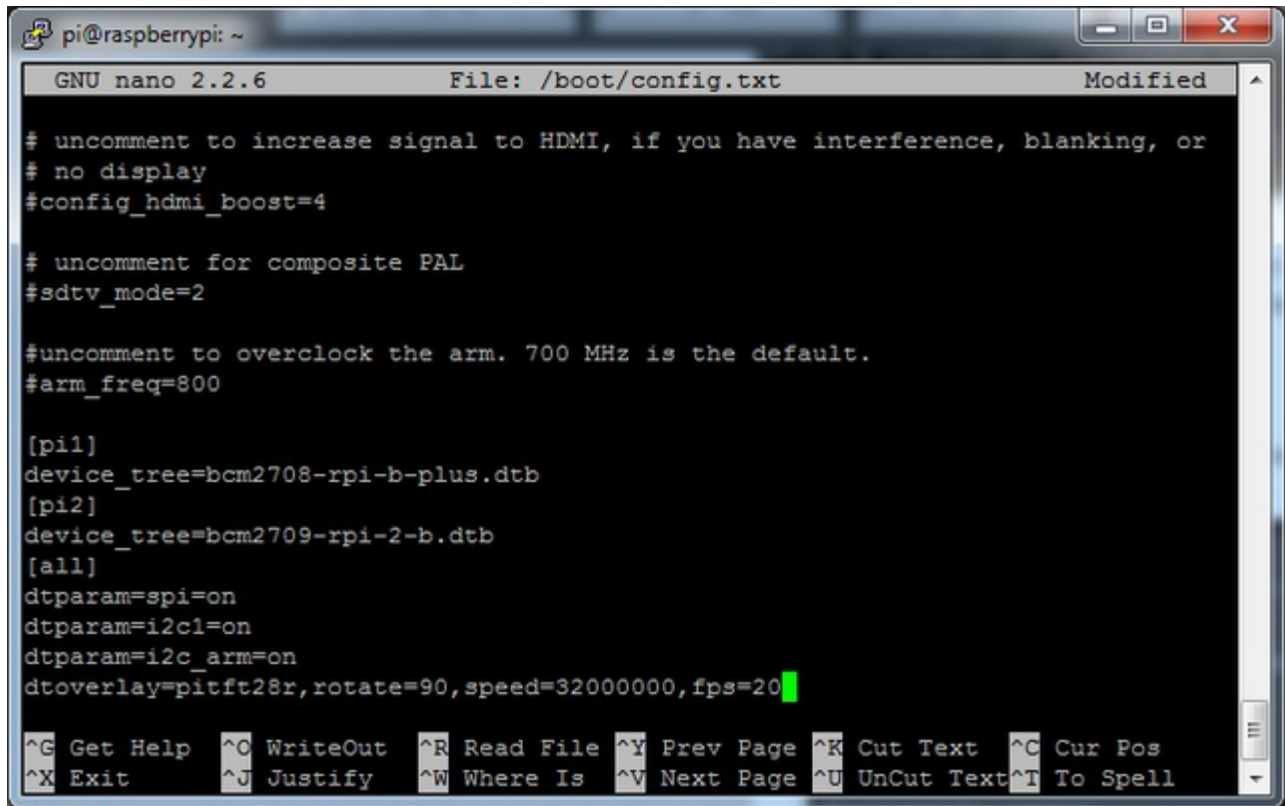
```
pi@raspberrypi ~/Adafruit-Occidentalis $ sudo apt-get install raspberrypi-bootloader
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
The following packages will be upgraded:
  libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0 raspberrypi-bootloader
5 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
Need to get 61.5 MB of archives.
After this operation, 12.7 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

OK since you're not going to run the helper, lets add the device tree overlay manually. Edit `/boot/config.txt` with

**`sudo nano /boot/config.txt`**

and add the following lines at the end:

```
[pi1]
device_tree=bcm2708-rpi-b-plus.dtb
[pi2]
device_tree=bcm2709-rpi-2-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20
```



```
pi@raspberrypi: ~
GNU nano 2.2.6      File: /boot/config.txt      Modified

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

[pi1]
device_tree=bcm2708-rpi-b-plus.dtb
[pi2]
device_tree=bcm2709-rpi-2-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

The **rotate=** variable tells the driver to rotate the screen **0 90 180** or **270** degrees.

**0** is portrait, with the bottom near the USB jacks

**90** is landscape, with the bottom of the screen near the headphone jack

**180** is portrait, with the top near the USB jacks

**270** is landscape, with the top of the screen near the headphone jack.

You can change this file with **nano** and reboot to make the change stick.

The **speed=** variable tells the driver how fast to drive the display. 32MHz (**32000000**) is a good place to start but if your screen is acting funny, try taking it down to 16MHz (**16000000**) *especially* if you're doing something like using a GPIO extender to put the screen away from the Pi.

Save the file. Now we'll just reboot to let it all sink in.

**sudo shutdown -h now** (if you don't have the TFT installed, shutdown, place the TFT on the Pi and re-power)

or

**sudo reboot** (if you have the TFT plate installed already)

When the Pi restarts, the attached PiTFT should start out all white and then turn black. That means the kernel found the display and cleared the screen. If the screen did not turn black, that means that likely there's something up with your connection or kernel install. Solder

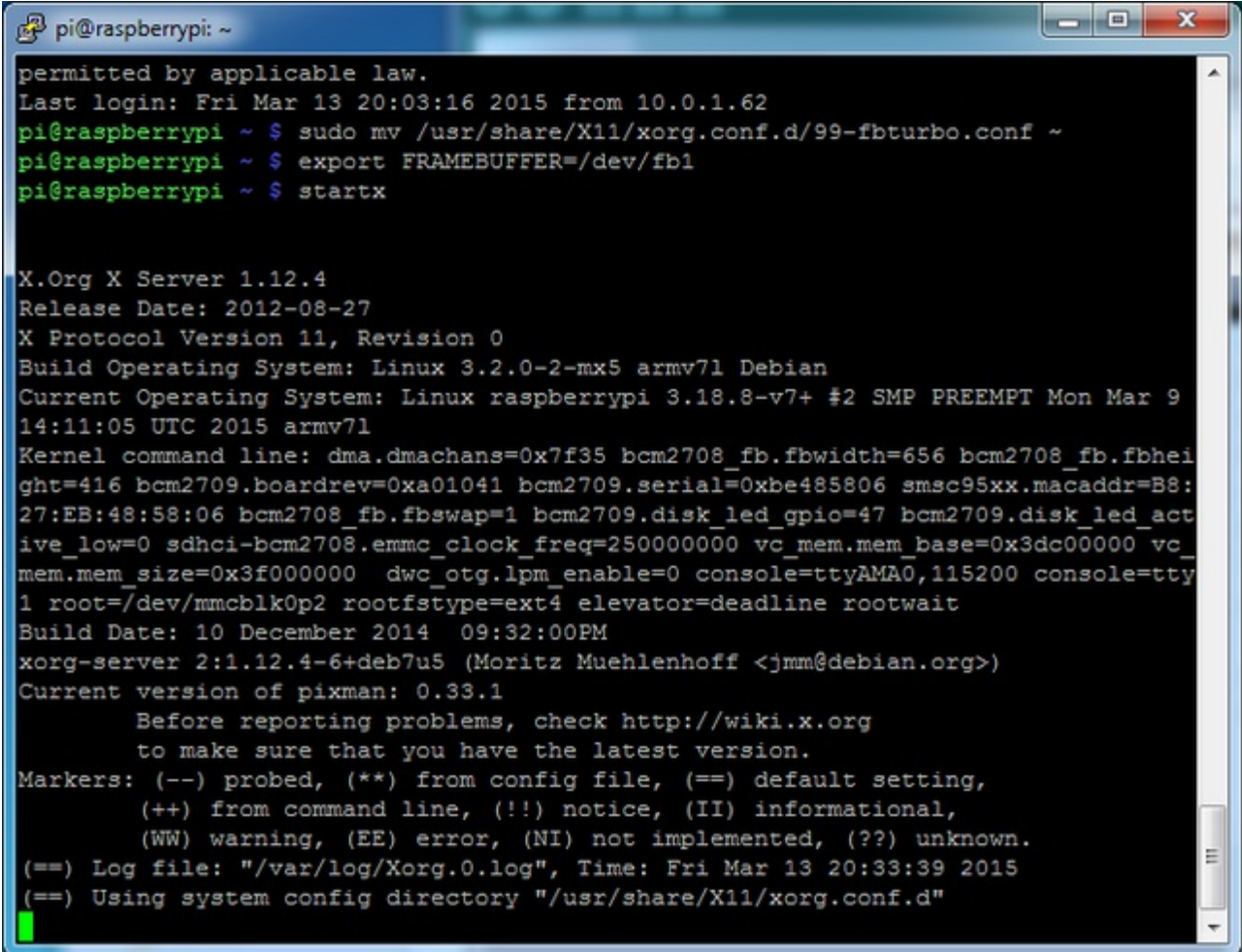


anything that needs resoldering!

Now that you're rebooted, log back in on the console/TV/SSH. There's nothing displayed on the screen yet, we'll do a test to make sure everything is perfect first!

Run the following commands to startx on the **/dev/fb1** framebuffer, a.k.a PiTFT screen:

```
sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
export FRAMEBUFFER=/dev/fb1
startx
```



```
pi@raspberrypi: ~
permitted by applicable law.
Last login: Fri Mar 13 20:03:16 2015 from 10.0.1.62
pi@raspberrypi ~ $ sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
pi@raspberrypi ~ $ export FRAMEBUFFER=/dev/fb1
pi@raspberrypi ~ $ startx

X.Org X Server 1.12.4
Release Date: 2012-08-27
X Protocol Version 11, Revision 0
Build Operating System: Linux 3.2.0-2-mx5 armv7l Debian
Current Operating System: Linux raspberrypi 3.18.8-v7+ #2 SMP PREEMPT Mon Mar 9
14:11:05 UTC 2015 armv7l
Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.fbwidth=656 bcm2708_fb.fbhei
ght=416 bcm2709.boardrev=0xa01041 bcm2709.serial=0xbe485806 smsc95xx.macaddr=B8:
27:EB:48:58:06 bcm2708_fb.fbswap=1 bcm2709.disk_led_gpio=47 bcm2709.disk_led_act
ive_low=0 sdhci-bcm2708.emmc_clock_freq=250000000 vc_mem.mem_base=0x3dc00000 vc_
mem.mem_size=0x3f000000 dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty
1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
Build Date: 10 December 2014 09:32:00PM
xorg-server 2:1.12.4-6+deb7u5 (Moritz Muehlenhoff <jmm@debian.org>)
Current version of pixman: 0.33.1
    Before reporting problems, check http://wiki.x.org
    to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
        (++) from command line, (!!) notice, (II) informational,
        (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Fri Mar 13 20:33:39 2015
(==) Using system config directory "/usr/share/X11/xorg.conf.d"
```

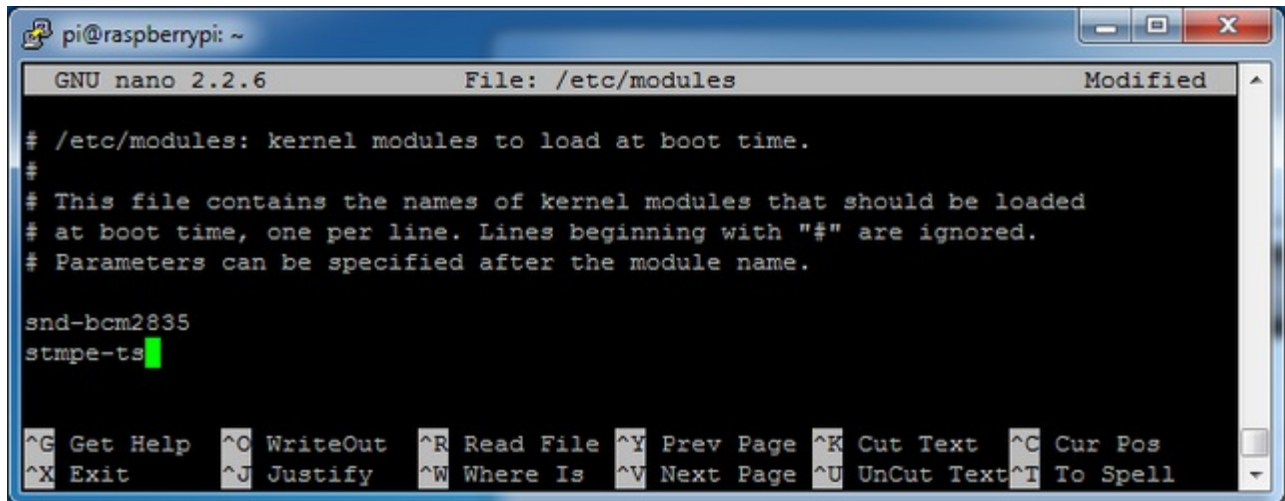
You should see the Pi desktop show up on the TFT! Congrats, you've completed the first test perfectly.

Hit Control-C in the console to quit the X server so we can continue configuration

Next up we'll add support for the touch screen automatically on boot. Edit the module list with

**sudo nano /etc/modules**

and add **stmpe-ts** on a line at the end



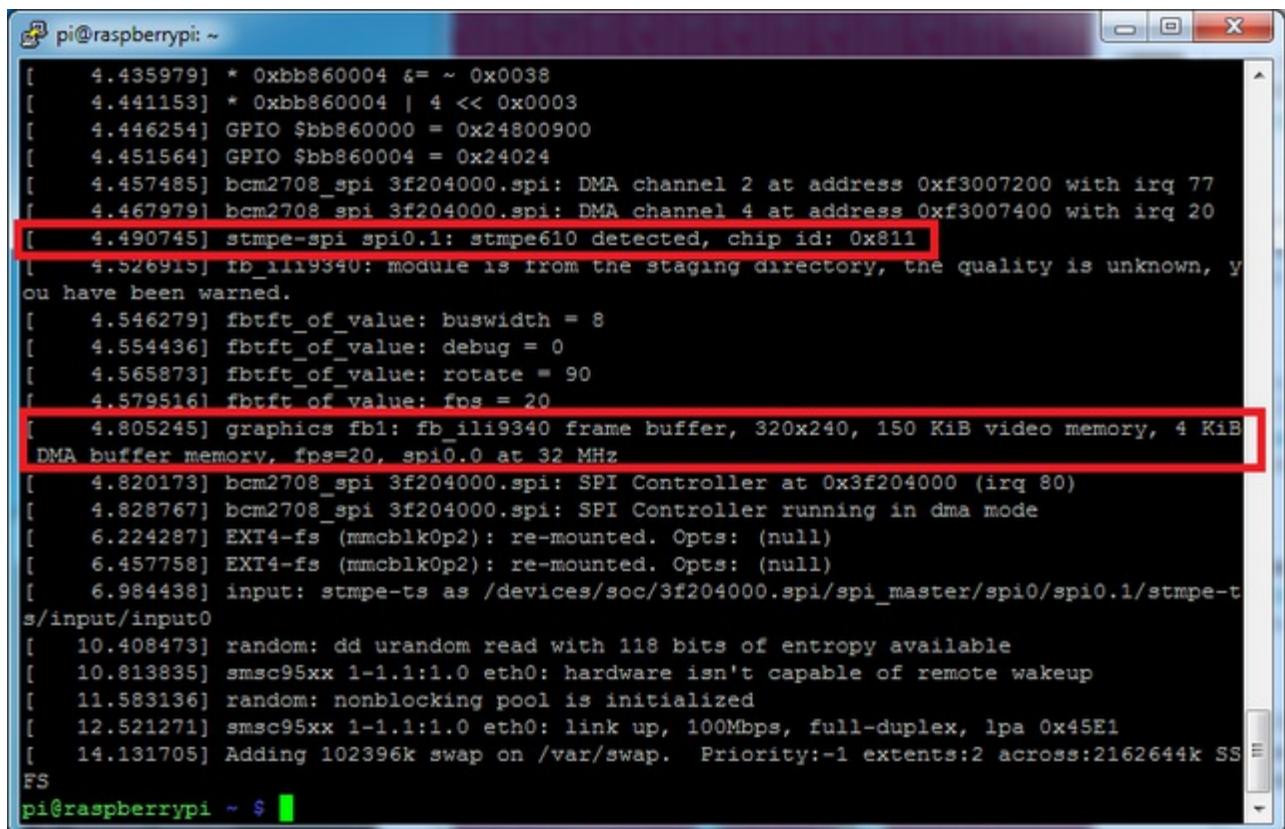
```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/modules Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
stmpe-ts

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Save the file and reboot the Pi with **sudo reboot** and look at the console output (or run **dmesg** in the console window after logging in) you will see the modules install. Look in particular for the STMPE610 detection and the ILI9340 screen frequency as highlighted here



```
pi@raspberrypi: ~
[ 4.435979] * 0xbb860004 &= ~ 0x0038
[ 4.441153] * 0xbb860004 | 4 << 0x0003
[ 4.446254] GPIO $bb860000 = 0x24800900
[ 4.451564] GPIO $bb860004 = 0x24024
[ 4.457485] bcm2708_spi 3f204000.spi: DMA channel 2 at address 0xf3007200 with irq 77
[ 4.467979] bcm2708_spi 3f204000.spi: DMA channel 4 at address 0xf3007400 with irq 20
[ 4.490745] stmpe-spi spi0.1: stmpe610 detected, chip id: 0x811
[ 4.526915] fb_ili9340: module is from the staging directory, the quality is unknown, you have been warned.
[ 4.546279] fbtft_of_value: buswidth = 8
[ 4.554436] fbtft_of_value: debug = 0
[ 4.565873] fbtft_of_value: rotate = 90
[ 4.579516] fbtft_of_value: fps = 20
[ 4.805245] graphics fb1: fb_ili9340 frame buffer, 320x240, 150 KiB video memory, 4 KiB DMA buffer memory, fps=20, spi0.0 at 32 MHz
[ 4.820173] bcm2708_spi 3f204000.spi: SPI Controller at 0x3f204000 (irq 80)
[ 4.828767] bcm2708_spi 3f204000.spi: SPI Controller running in dma mode
[ 6.224287] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[ 6.457758] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[ 6.984438] input: stmpe-ts as /devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-ts/input/input0
[ 10.408473] random: dd urandom read with 118 bits of entropy available
[ 10.813835] smsc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
[ 11.583136] random: nonblocking pool is initialized
[ 12.521271] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[ 14.131705] Adding 102396k swap on /var/swap. Priority:-1 extents:2 across:2162644k SS
pi@raspberrypi ~ $
```

We can set up the touchscreen for **rotate=90** configuration by doing the following (for more delicate calibration or for other rotate=XX values, see the next section)

Create the directory and new calibration configuration file:

**sudo mkdir /etc/X11/xorg.conf.d**

**sudo nano /etc/X11/xorg.conf.d/99-calibration.conf**

and enter in the following lines, then save.

Section "InputClass"

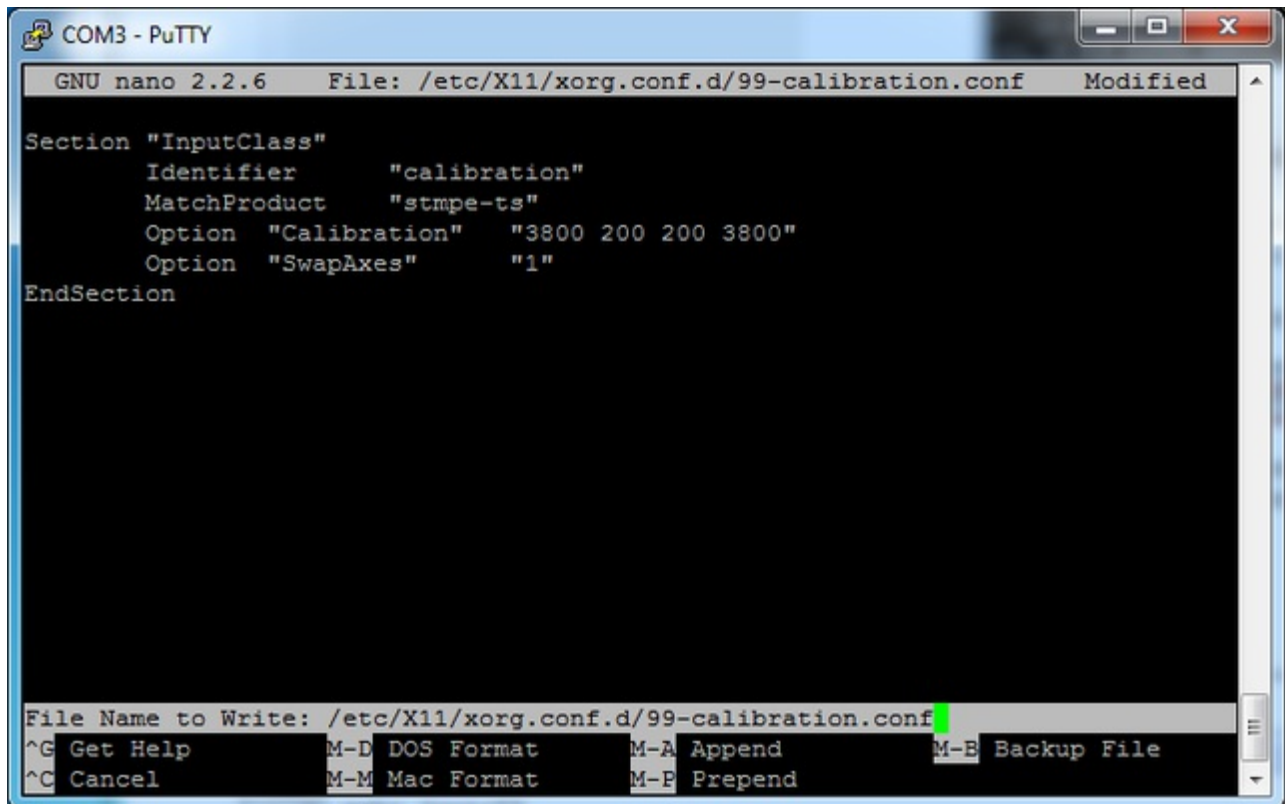
Identifier "calibration"

MatchProduct "stmpe-ts"

Option "Calibration" "3800 200 200 3800"

Option "SwapAxes" "1"

EndSection



```
COM3 - PuTTY
GNU nano 2.2.6 File: /etc/X11/xorg.conf.d/99-calibration.conf Modified
Section "InputClass"
  Identifier      "calibration"
  MatchProduct    "stmpe-ts"
  Option "Calibration" "3800 200 200 3800"
  Option "SwapAxes"  "1"
EndSection
File Name to Write: /etc/X11/xorg.conf.d/99-calibration.conf
^G Get Help      M-D DOS Format   M-A Append      M-B Backup File
^C Cancel        M-M Mac Format   M-P Prepend
```

You can now try to run X again with

**FRAMEBUFFER=/dev/fb1 startx**

Type Control-C to quit X

If you don't ever want to have to type **FRAMEBUFFER=/dev/fb1** before **startx**, you can make it a default state by editing your profile file: **sudo nano ~/.profile** and adding

**export FRAMEBUFFER=/dev/fb1**

near the top and saving the file. Then reboot to reload the profile file. It will now always assume you want to use /dev/fb1

```
COM3 - PuTTY
GNU nano 2.2.6      File: /home/pi/.profile

# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

export FRAMEBUFFER=/dev/fb1

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

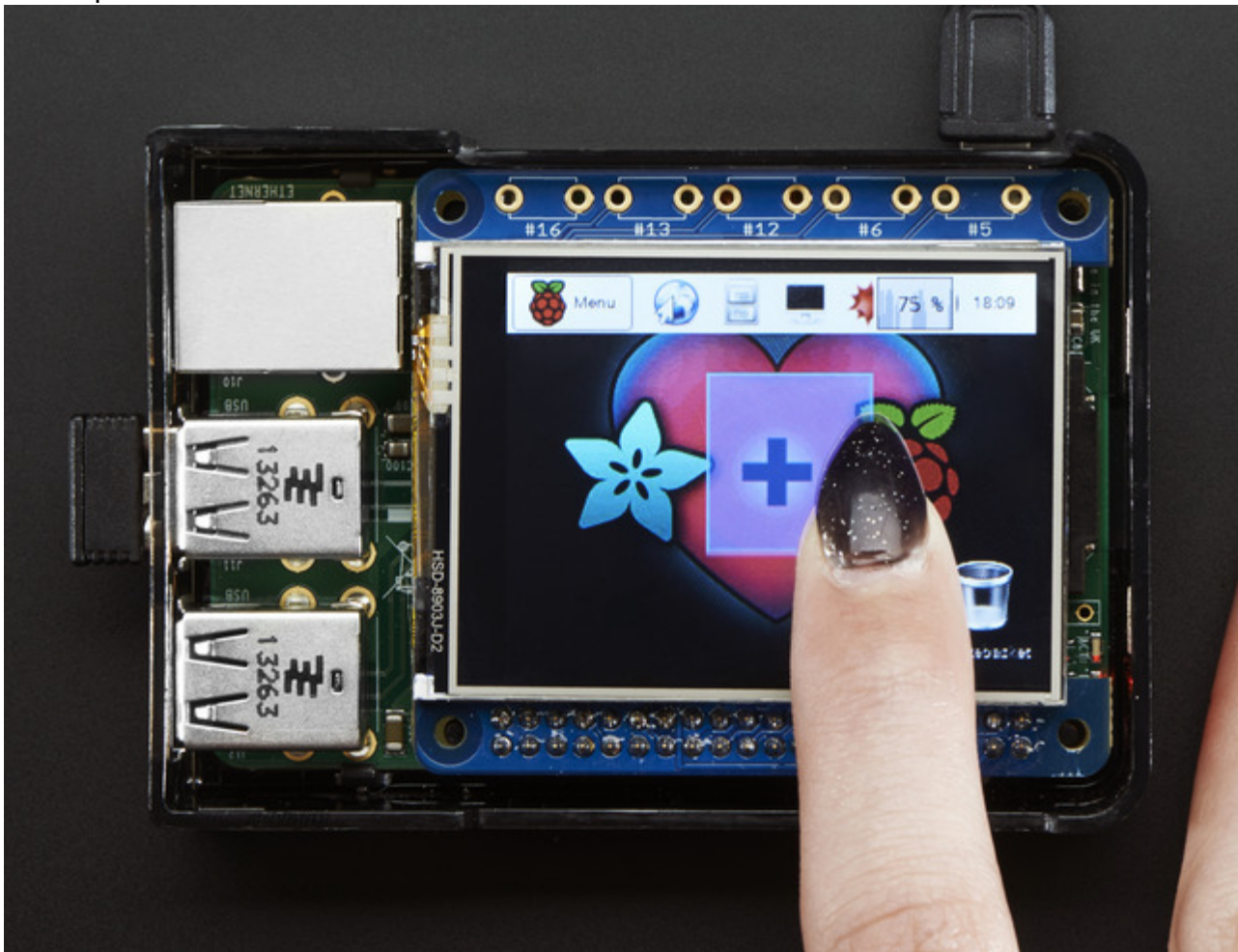
[ Read 24 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```



# Resistive Touchscreen Manual Install & Calibrate

If you've grabbed our Easy Install image, or used the installer script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the touchscreen

This procedure is identical for the 2.4", 2.8", 3.2" and 3.5" Resistive PiTFTs. Not for use with the Capacitive PiTFT!



## Setting up the Touchscreen

Now that the screen is working nicely, we'll take care of the touchscreen. There's just a bit of calibration to do, but it isn't hard at all.



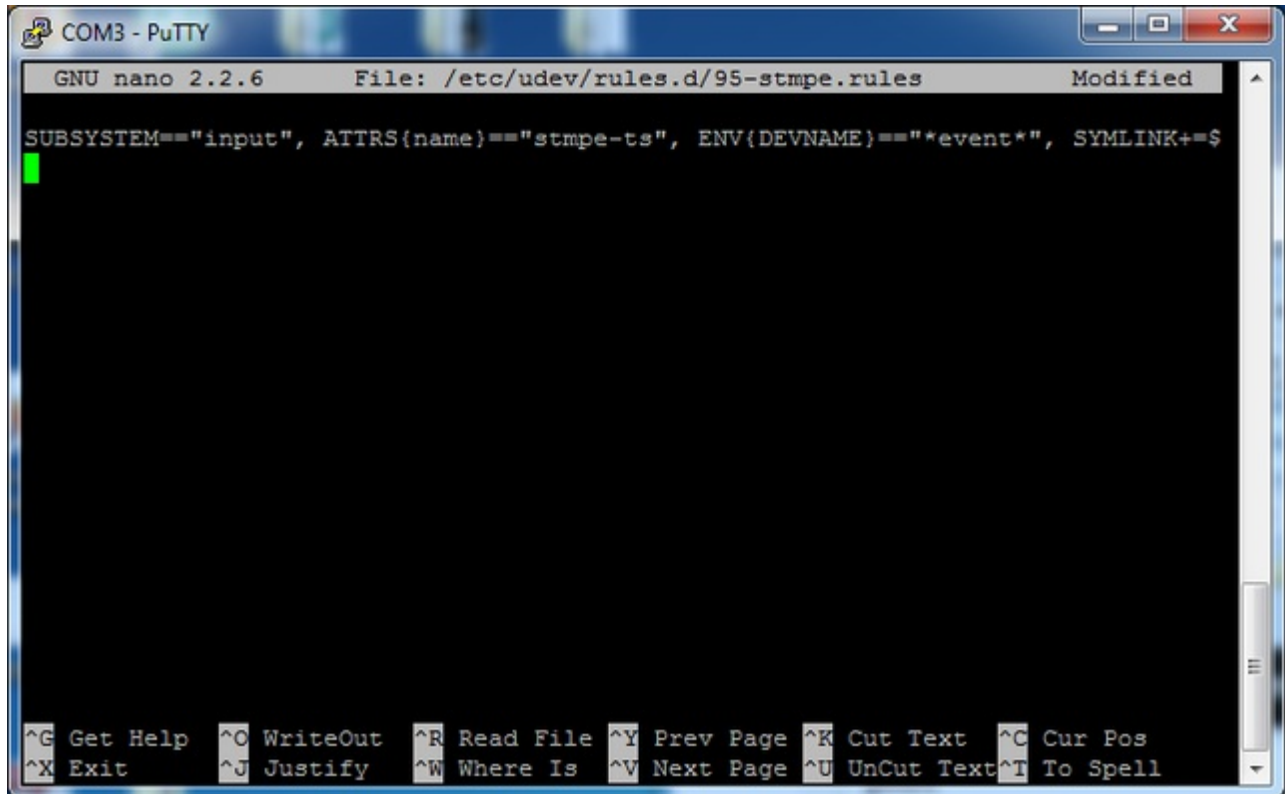
Before we start, we'll make a **udev** rule for the touchscreen. That's because the **eventX** name of the device will change a lot and its annoying to figure out what its called depending on whether you have a keyboard or other mouse installed.

Run

```
sudo nano /etc/udev/rules.d/95-stmpe.rules
```

to create a new **udev** file and copy & paste the following line in:

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

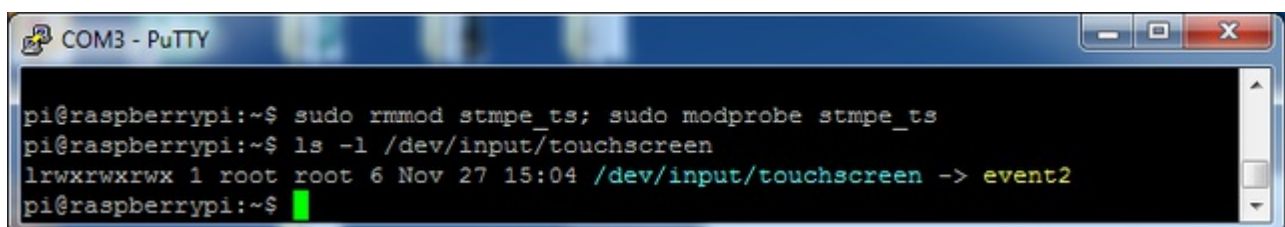
A screenshot of a PuTTY terminal window titled 'COM3 - PuTTY'. The terminal shows the GNU nano 2.2.6 text editor editing the file /etc/udev/rules.d/95-stmpe.rules. The file content is: SUBSYSTEM=="input", ATTRS{name}=="stmpe-ts", ENV{DEVNAME}=="\*event\*", SYMLINK+=\$. The bottom status bar of the nano editor shows various keyboard shortcuts like ^G Get Help, ^O WriteOut, ^R Read File, etc.

Remove and re-install the touchscreen with

```
sudo rmmod stmpe_ts; sudo modprobe stmpe_ts
```

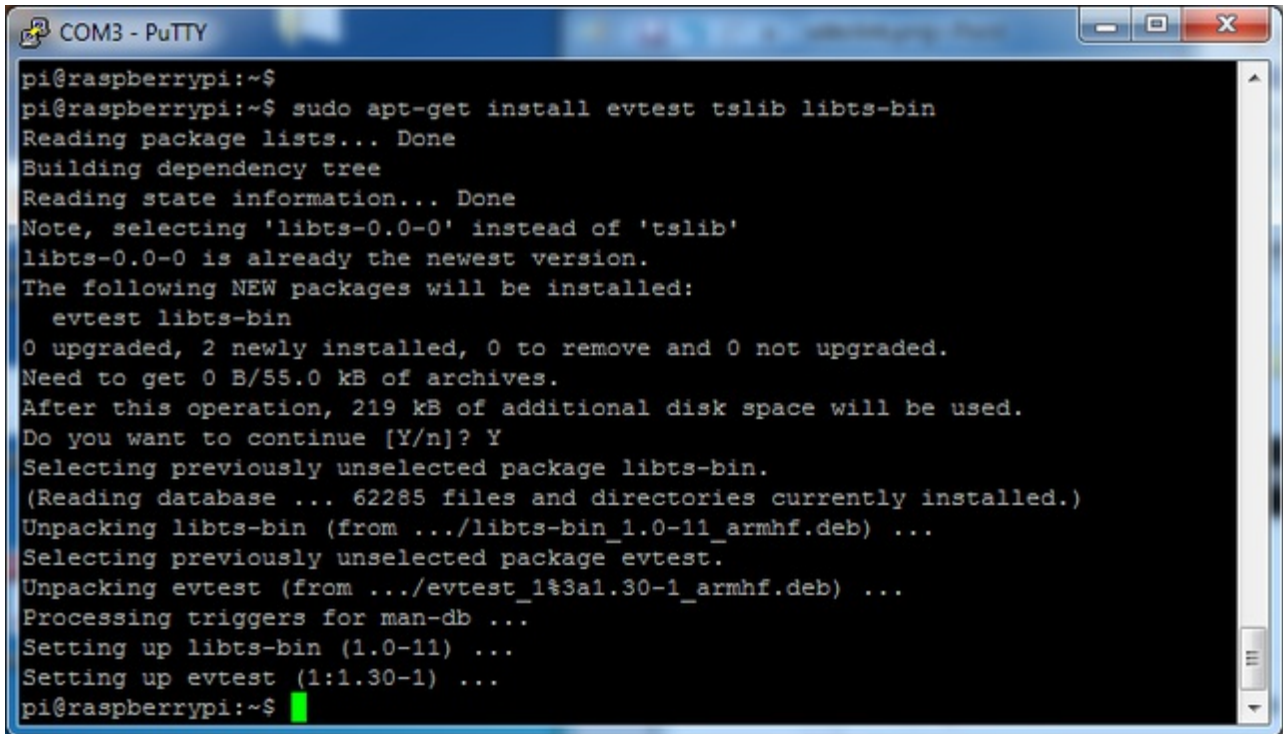
Then type **ls -l /dev/input/touchscreen**

It should point to **eventX** where X is some number, that number will be different on different setups since other keyboards/mice/USB devices will take up an event slot

A screenshot of a PuTTY terminal window titled 'COM3 - PuTTY'. The terminal shows the following commands and output: pi@raspberrypi:~\$ sudo rmmod stmpe\_ts; sudo modprobe stmpe\_ts; pi@raspberrypi:~\$ ls -l /dev/input/touchscreen; lrwxrwxrwx 1 root root 6 Nov 27 15:04 /dev/input/touchscreen -> event2; pi@raspberrypi:~\$

There are some tools we can use to calibrate & debug the touchscreen. Install the "event test" and "touchscreen library" packages with

```
sudo apt-get install evtest tslib libts-bin
```



```
pi@raspberrypi:~$  
pi@raspberrypi:~$ sudo apt-get install evtest tslib libts-bin  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Note, selecting 'libts-0.0-0' instead of 'tslib'  
libts-0.0-0 is already the newest version.  
The following NEW packages will be installed:  
  evtest libts-bin  
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.  
Need to get 0 B/55.0 kB of archives.  
After this operation, 219 kB of additional disk space will be used.  
Do you want to continue [Y/n]? Y  
Selecting previously unselected package libts-bin.  
(Reading database ... 62285 files and directories currently installed.)  
Unpacking libts-bin (from .../libts-bin_1.0-11_armhf.deb) ...  
Selecting previously unselected package evtest.  
Unpacking evtest (from .../evtest_1%3a1.30-1_armhf.deb) ...  
Processing triggers for man-db ...  
Setting up libts-bin (1.0-11) ...  
Setting up evtest (1:1.30-1) ...  
pi@raspberrypi:~$
```

## Running evtest

Now you can use some tools such as

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

which will let you see touchscreen events in real time, press on the touchscreen to see the reports.

```
COM3 - PuTTY
pi@raspberrypi:~$ 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)

COM3 - PuTTY
Event: time 1385565357.639692, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 149
Event: time 1385565357.639699, ----- SYN_REPORT -----
Event: time 1385565357.645965, type 3 (EV_ABS), code 0 (ABS_X), value 1580
Event: time 1385565357.645973, type 3 (EV_ABS), code 1 (ABS_Y), value 1846
Event: time 1385565357.645980, ----- SYN_REPORT -----
Event: time 1385565357.652293, type 3 (EV_ABS), code 0 (ABS_X), value 1634
Event: time 1385565357.652301, type 3 (EV_ABS), code 1 (ABS_Y), value 1864
Event: time 1385565357.652305, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 143
Event: time 1385565357.652310, ----- SYN_REPORT -----
Event: time 1385565357.658614, type 3 (EV_ABS), code 0 (ABS_X), value 1658
Event: time 1385565357.658622, type 3 (EV_ABS), code 1 (ABS_Y), value 1877
Event: time 1385565357.658626, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 139
Event: time 1385565357.658631, ----- SYN_REPORT -----
Event: time 1385565357.664919, type 3 (EV_ABS), code 0 (ABS_X), value 1748
Event: time 1385565357.664928, type 3 (EV_ABS), code 1 (ABS_Y), value 1888
Event: time 1385565357.664935, ----- SYN_REPORT -----
Event: time 1385565357.671199, type 3 (EV_ABS), code 0 (ABS_X), value 1778
Event: time 1385565357.671207, type 3 (EV_ABS), code 1 (ABS_Y), value 1895
Event: time 1385565357.671211, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 134
Event: time 1385565357.671216, ----- SYN_REPORT -----
Event: time 1385565357.698600, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 0
Event: time 1385565357.698607, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1385565357.698610, ----- SYN_REPORT -----
```

## AutoMagic Calibration Script

If you rotate the display you need to recalibrate the touchscreen to work with the new screen

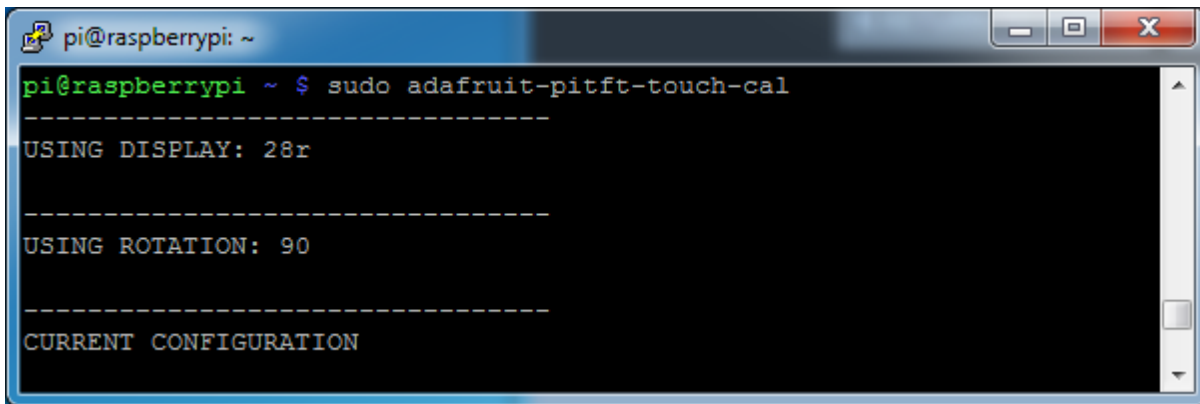
orientation. You can manually run the calibration processes in the next section, or you can run a small Python script which will automatically set a default touchscreen calibration based on the screen orientation.

[This helper is automatically installed for you but if you'd like you can check it out here on github \(https://adafru.it/elu\)](https://adafru.it/elu)

Run it at the command line with

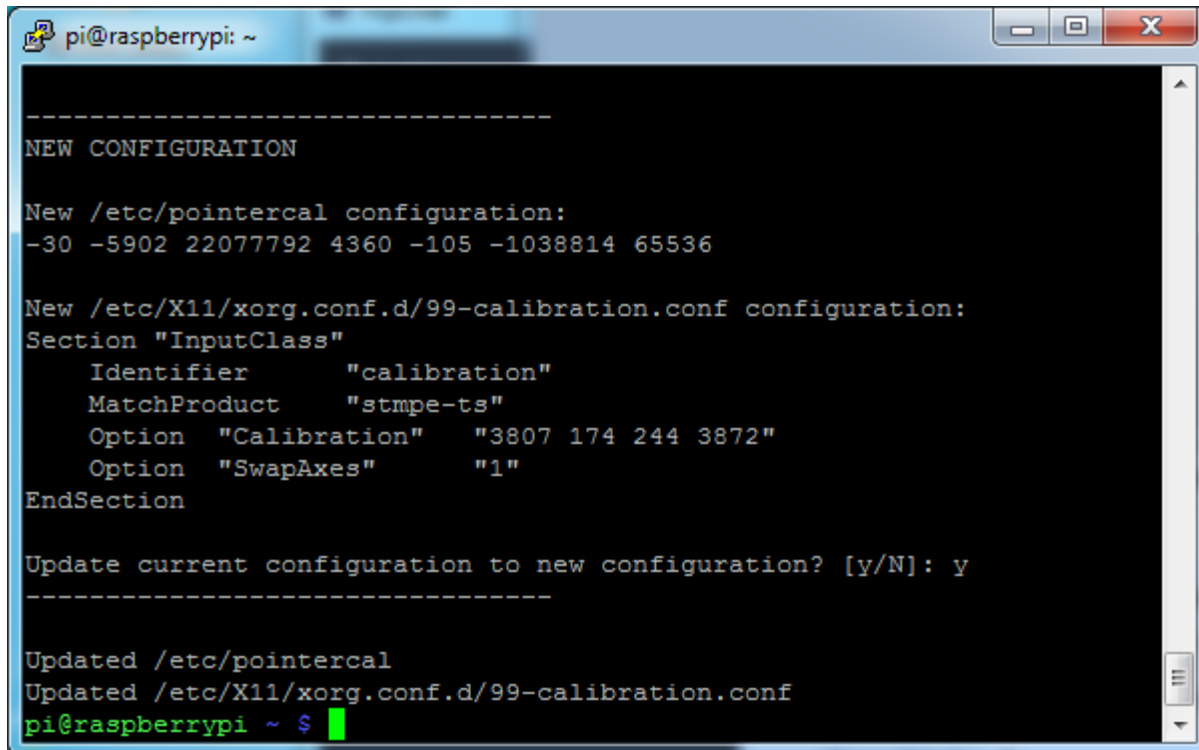
```
sudo adafruit-pitft-touch-cal
```

it will try to figure out what display you have installed and the rotation it's set up for

A terminal window titled 'pi@raspberrypi: ~' showing the execution of the 'sudo adafruit-pitft-touch-cal' command. The output is as follows:

```
pi@raspberrypi ~ $ sudo adafruit-pitft-touch-cal
-----
USING DISPLAY: 28r
-----
USING ROTATION: 90
-----
CURRENT CONFIGURATION
```

By default the script will attempt to read the screen orientation by examining the PiTFT module configuration with modprobe. If the script can read the orientation it will print out the current orientation, the current touchscreen calibration values, and the new touchscreen calibration values based on the current orientation. Before updating the calibration the script will ask you to confirm that you'd like to make the change. Press **y** and enter to confirm.



```
pi@raspberrypi: ~  
-----  
NEW CONFIGURATION  
  
New /etc/pointercal configuration:  
-30 -5902 22077792 4360 -105 -1038814 65536  
  
New /etc/X11/xorg.conf.d/99-calibration.conf configuration:  
Section "InputClass"  
    Identifier      "calibration"  
    MatchProduct    "stmpe-ts"  
    Option  "Calibration"    "3807 174 244 3872"  
    Option  "SwapAxes"      "1"  
EndSection  
  
Update current configuration to new configuration? [y/N]: y  
-----  
  
Updated /etc/pointercal  
Updated /etc/X11/xorg.conf.d/99-calibration.conf  
pi@raspberrypi ~ $
```

Try using this default calibration script to easily calibrate your touchscreen display. Note that the calibration values might not be exactly right for your display, but they should be close enough for most needs. If you need the most accurate touchscreen calibration, follow the steps in the next section to manually calibrate the touchscreen.

## Manual Calibration

If the "automagic" calibration technique isn't working for you, or you have some other setup where you need to carefully calibrate you can do it 'manually'

You will want to calibrate the screen once but shouldn't have to do it more than that. We'll begin by calibrating on the command line by running

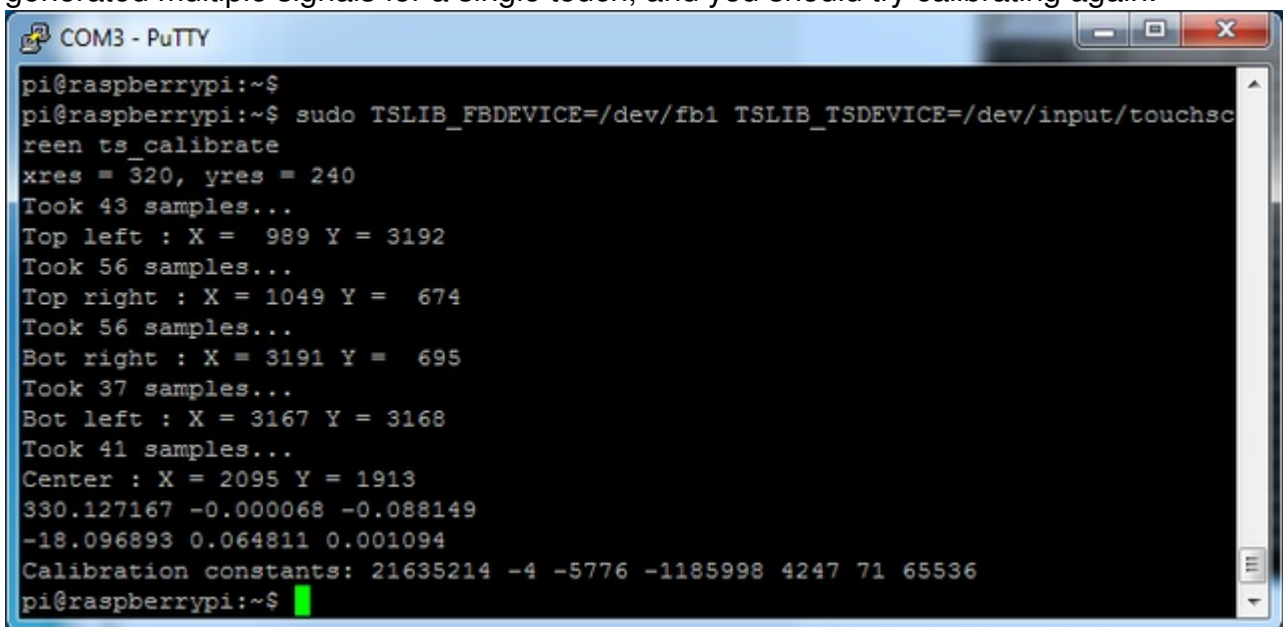
```
sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen ts_calibrate
```

follow the directions on the screen, touching each point. Using a stylus is suggested so you get a precise touch. Don't use something metal, plastic only!



```
┌─┐
└─┘ TSLIB calibration utility
    Touch crosshair to calibrate
```

You should see five crosshair targets. If you see less than that, the touchscreen probably generated multiple signals for a single touch, and you should try calibrating again.



```
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen ts_calibrate
xres = 320, yres = 240
Took 43 samples...
Top left : X = 989 Y = 3192
Took 56 samples...
Top right : X = 1049 Y = 674
Took 56 samples...
Bot right : X = 3191 Y = 695
Took 37 samples...
Bot left : X = 3167 Y = 3168
Took 41 samples...
Center : X = 2095 Y = 1913
330.127167 -0.000068 -0.088149
-18.096893 0.064811 0.001094
Calibration constants: 21635214 -4 -5776 -1185998 4247 71 65536
pi@raspberrypi:~$
```

Next you can run

```
sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen ts_test
```

which will let you draw-test the touch screen. Go back and re-calibrate if you feel the screen isn't precise enough!



## X Calibration

You can also calibrate the X input system but you have to use a different program called **xinput\_calibrator**

You can do this if the calibration on the screen isn't to your liking or any time you change the **rotate=XX** module settings for the screen. Since the screen and touch driver are completely separated, the touchscreen doesn't auto-rotate

Normally you'd have to compile it but we have a ready to go package for you so run:

```
sudo apt-get install -y xinput-calibrator
```

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo apt-get install -y xinput-calibrator
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libatkmm-1.6-1 libcairomm-1.0-1 libglibmm-2.4-1c2a libgtkmm-2.4-1c2a libpangomm-1.4-1
Suggested packages:
  xinput
The following NEW packages will be installed:
  libatkmm-1.6-1 libcairomm-1.0-1 libglibmm-2.4-1c2a libgtkmm-2.4-1c2a libpangomm-1.4-1 xi
0 upgraded, 6 newly installed, 0 to remove and 37 not upgraded.
Need to get 1,697 kB of archives.
After this operation, 5,606 kB of additional disk space will be used.
Get:1 http://apt.adafruit.com/raspbian/ wheezy/main xinput-calibrator armhf 0.7.6-1 [64.9
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libglibmm-2.4-1c2a armhf 2.
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libatkmm-1.6-1 armhf 2.22.6
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libcairomm-1.0-1 armhf 1.10
Get:5 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libpangomm-1.4-1 armhf 2.28
Get:6 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libgtkmm-2.4-1c2a armhf 1:2
Fetched 1,697 kB in 1s (1,150 kB/s)
Selecting previously unselected package libglibmm-2.4-1c2a:armhf.
(Reading database ... 80372 files and directories currently installed.)
```

Before you start the `xinput_calibrator` you will need to delete the old calibration data so run

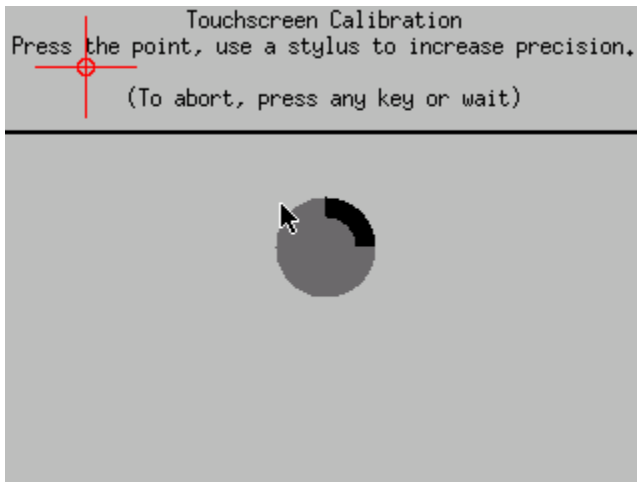
```
sudo rm /etc/X11/xorg.conf.d/99-calibration.conf
```

Before running `startx` and the calibrator - otherwise it gets really confused!

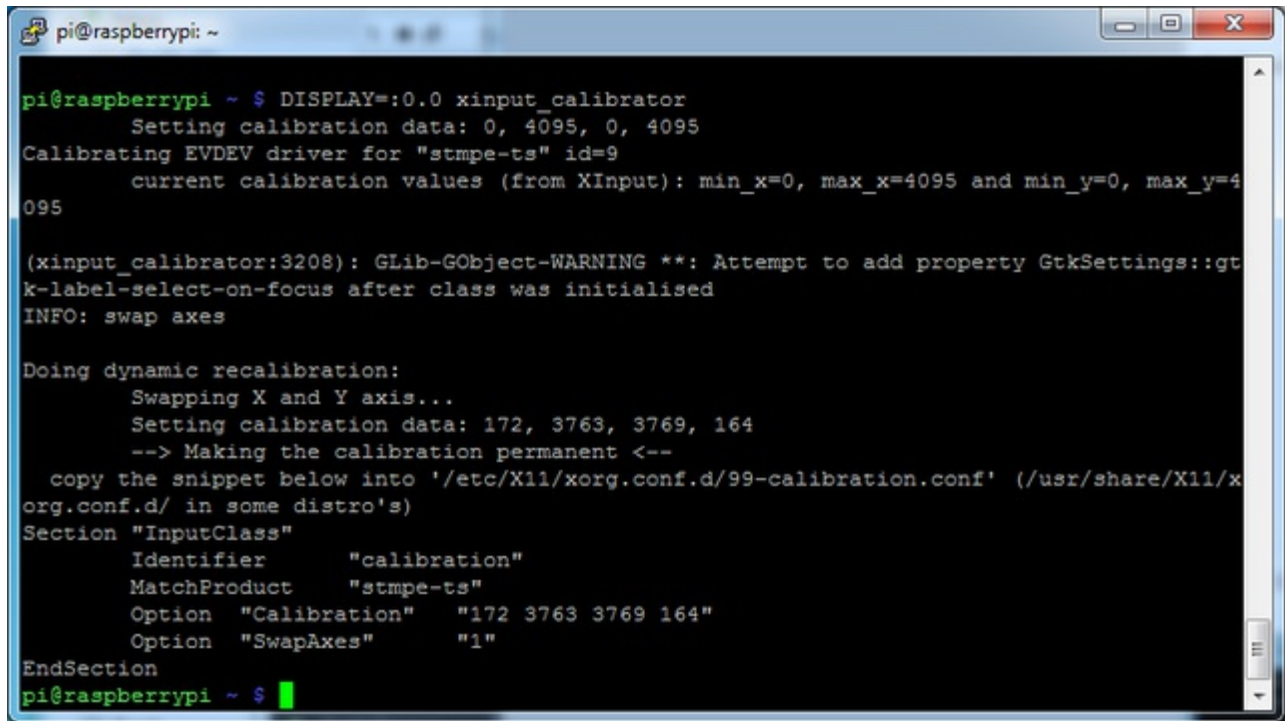
Now you'll have to run the xcalibrator while also running X. You can do this by **startx** and then opening up the terminal program and running the **xinput\_calibrator** command (which is challenging to do on such a small screen) OR you can do what we do which is run startx in a SSH/Terminal shell and then run the xinput\_calibrator from the same shell, which requires the following command order:

```
FRAMEBUFFER=/dev/fb1 startx &
DISPLAY=:0.0 xinput_calibrator
```

Follow the directions on screen



Once complete you'll get something like:



```
pi@raspberrypi: ~  
pi@raspberrypi ~ $ DISPLAY=:0.0 xinput_calibrator  
Setting calibration data: 0, 4095, 0, 4095  
Calibrating EVDEV driver for "stmpe-ts" id=9  
current calibration values (from XInput): min_x=0, max_x=4095 and min_y=0, max_y=4095  
  
(xinput_calibrator:3208): GLib-GObject-WARNING **: Attempt to add property GtkSettings::gtk-label-select-on-focus after class was initialised  
INFO: swap axes  
  
Doing dynamic recalibration:  
Swapping X and Y axis...  
Setting calibration data: 172, 3763, 3769, 164  
--> Making the calibration permanent <--  
copy the snippet below into '/etc/X11/xorg.conf.d/99-calibration.conf' (/usr/share/X11/xorg.conf.d/ in some distro's)  
Section "InputClass"  
Identifier      "calibration"  
MatchProduct    "stmpe-ts"  
Option "Calibration" "172 3763 3769 164"  
Option "SwapAxes"   "1"  
EndSection  
pi@raspberrypi ~ $
```

Run **sudo nano /etc/X11/xorg.conf.d/99-calibration.conf** and copy the

Section "InputClass"

```
Identifier      "calibration"  
MatchProduct    "stmpe-ts"  
Option "Calibration" "172 3763 3769 164"  
Option "SwapAxes"   "1"
```

EndSection

or whatever you got, into there. You can quit X if you want by typing **fg** to bring that command into the foreground, and then Control-C to quit.

Depending on the 'rotation' of the screen, when you do this calibration, you may need to comment out the SwapAxes part with a # and/or swap the numbers around so looks like:

```
Option "Calibration" "119 3736 3850 174"
```

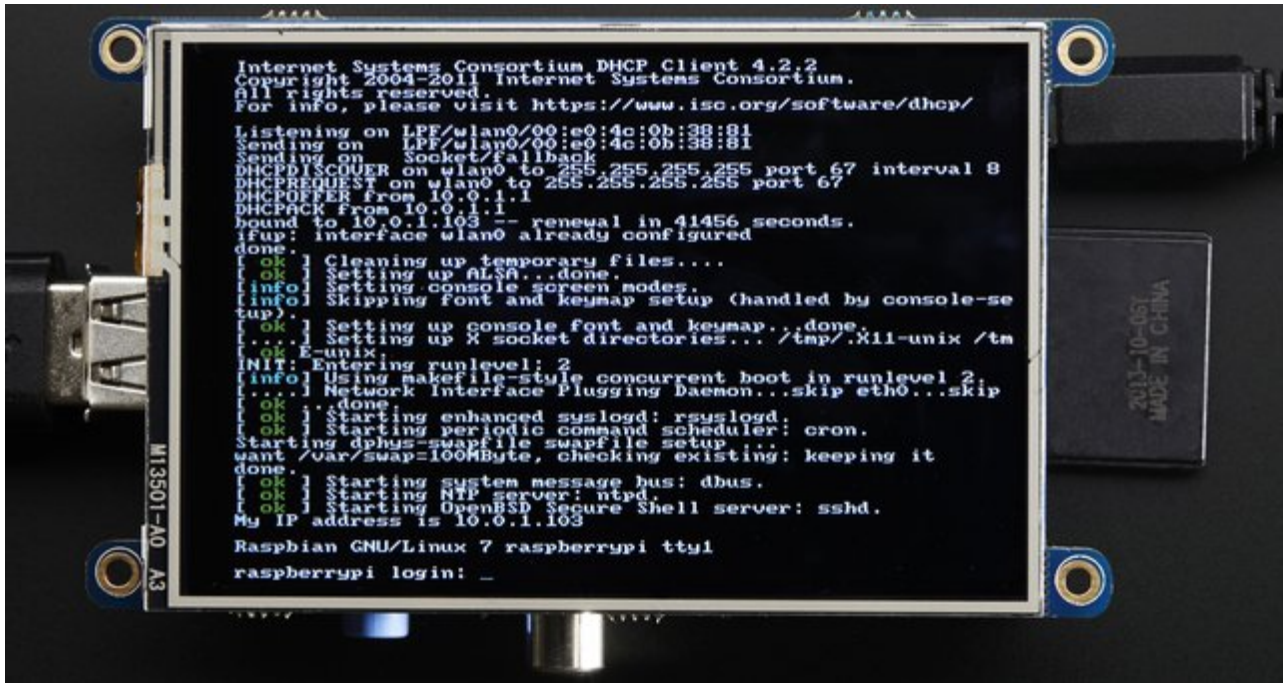
to

```
Option "Calibration" "3736 119 174 3850"
```

Your touchscreen is now super calibrated, hurrah!

# Console Configuration

If you've grabbed our Easy Install image, or use the script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the console



One fun thing you can do with the display is have it as your main console instead of the HDMI/TV output. Even though it is small, with a good font you can get 20 x 40 of text. For more details, check out <https://github.com/notro/fbtf/wiki/Boot-console> (<https://adafru.it/cXQ>)

First up, we'll update the boot configuration file to use the TFT framebuffer `/dev/fb1` instead of the HDMI/TV framebuffer `/dev/fb0`

```
sudo nano /boot/cmdline.txt
```

you can also edit it by putting the SD card into a computer and opening the same file.

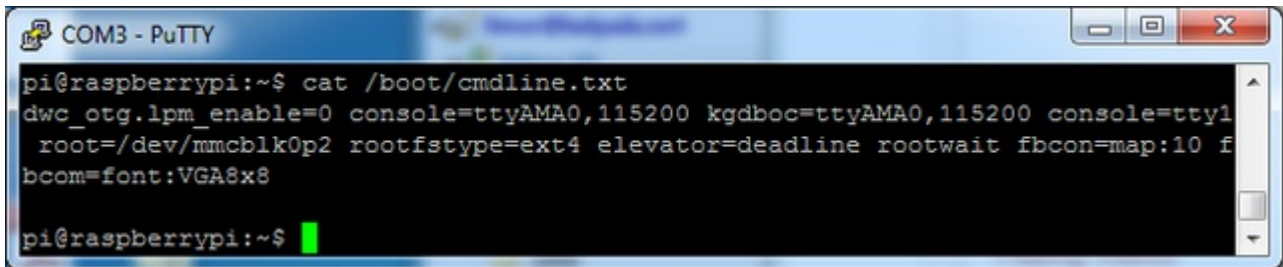
At the end of the line, find the text that says `rootwait` and right after that, enter in:  
`fbcon=map:10 fbcon=font:VGA8x8` then save the file.

On the next boot, it will bring up the console.

**Note that the kernel has to load up the display driver module before it can display anything on it so you won't get the rainbow screen, a NooBs prompt, or a big chunk of**



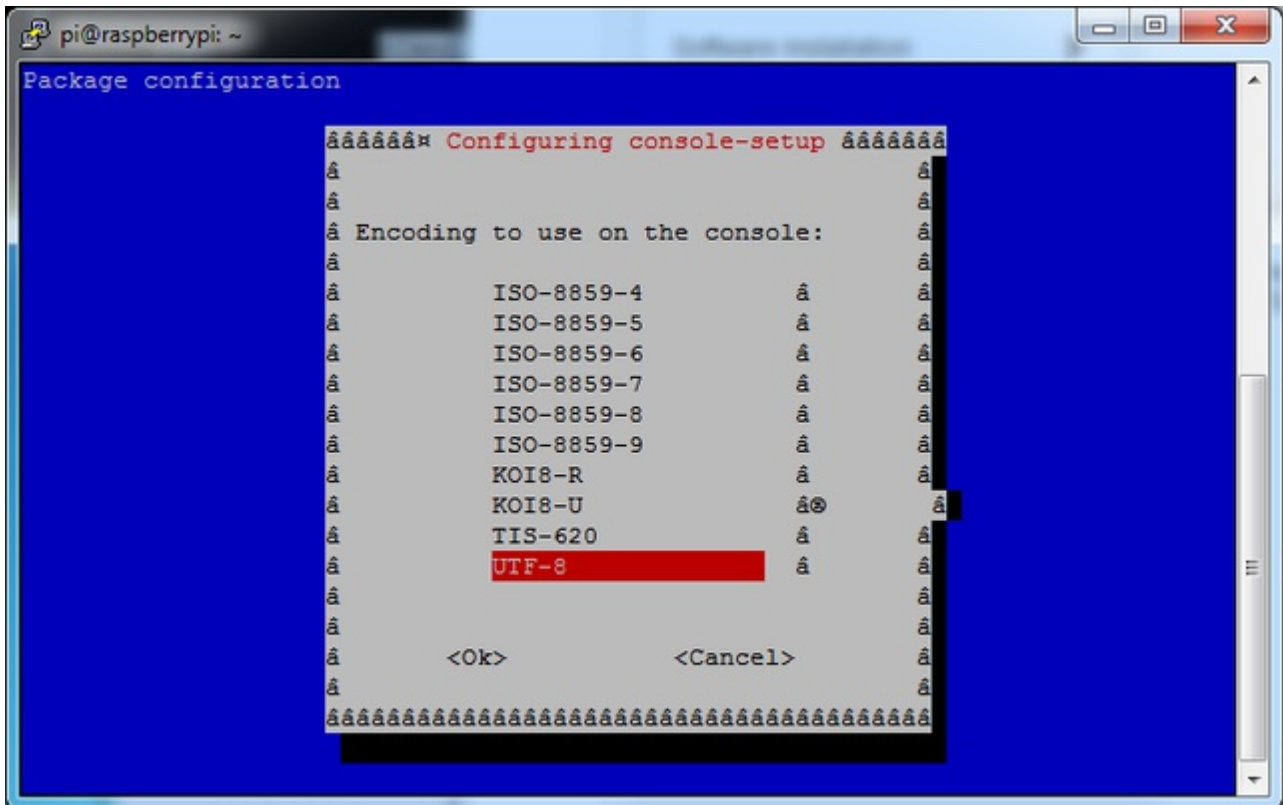
the kernel details since the module is loaded fairly late in the boot process.

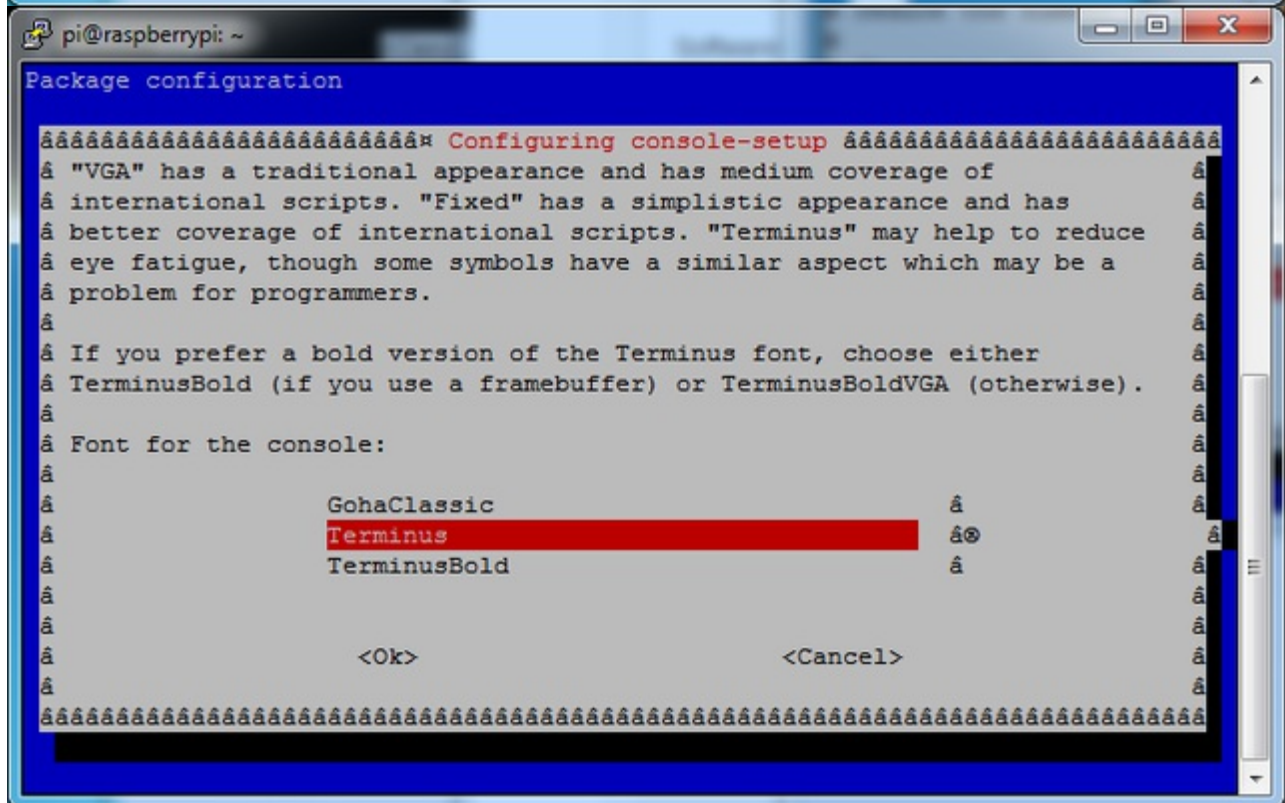
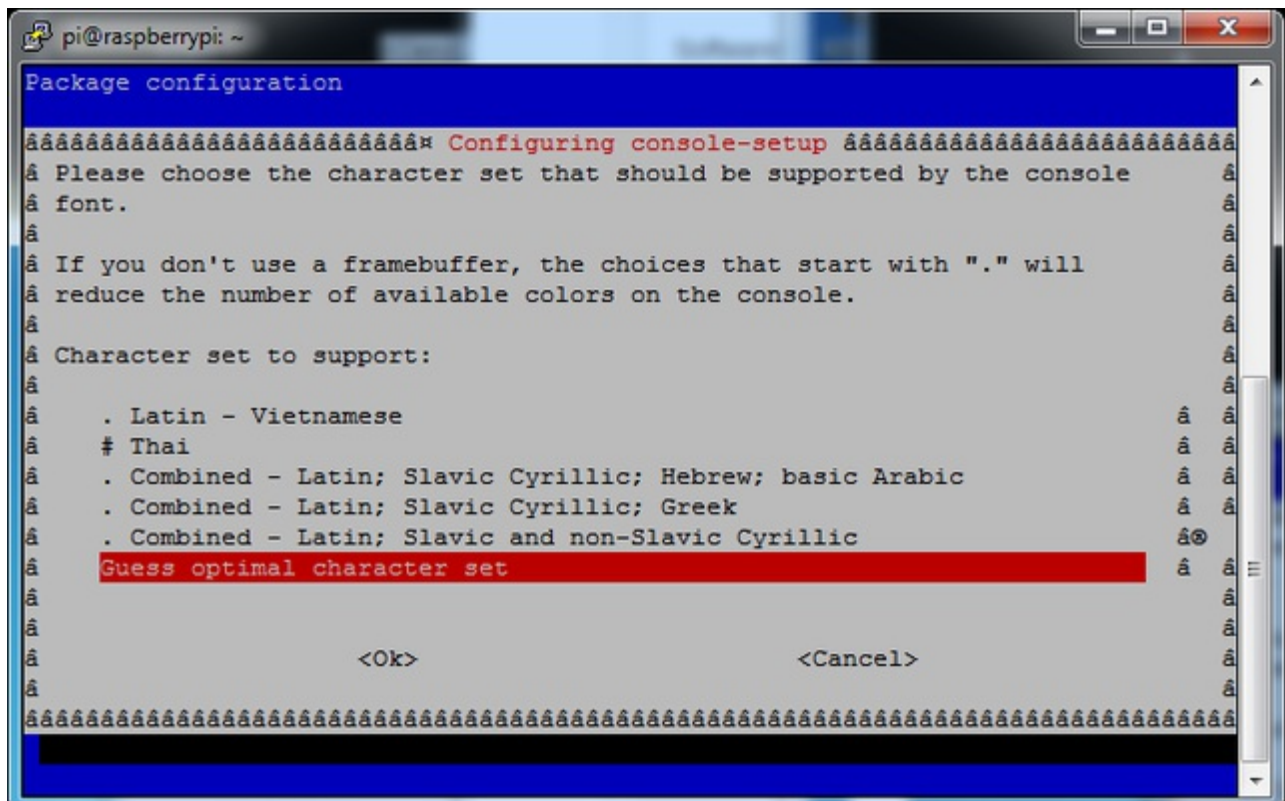


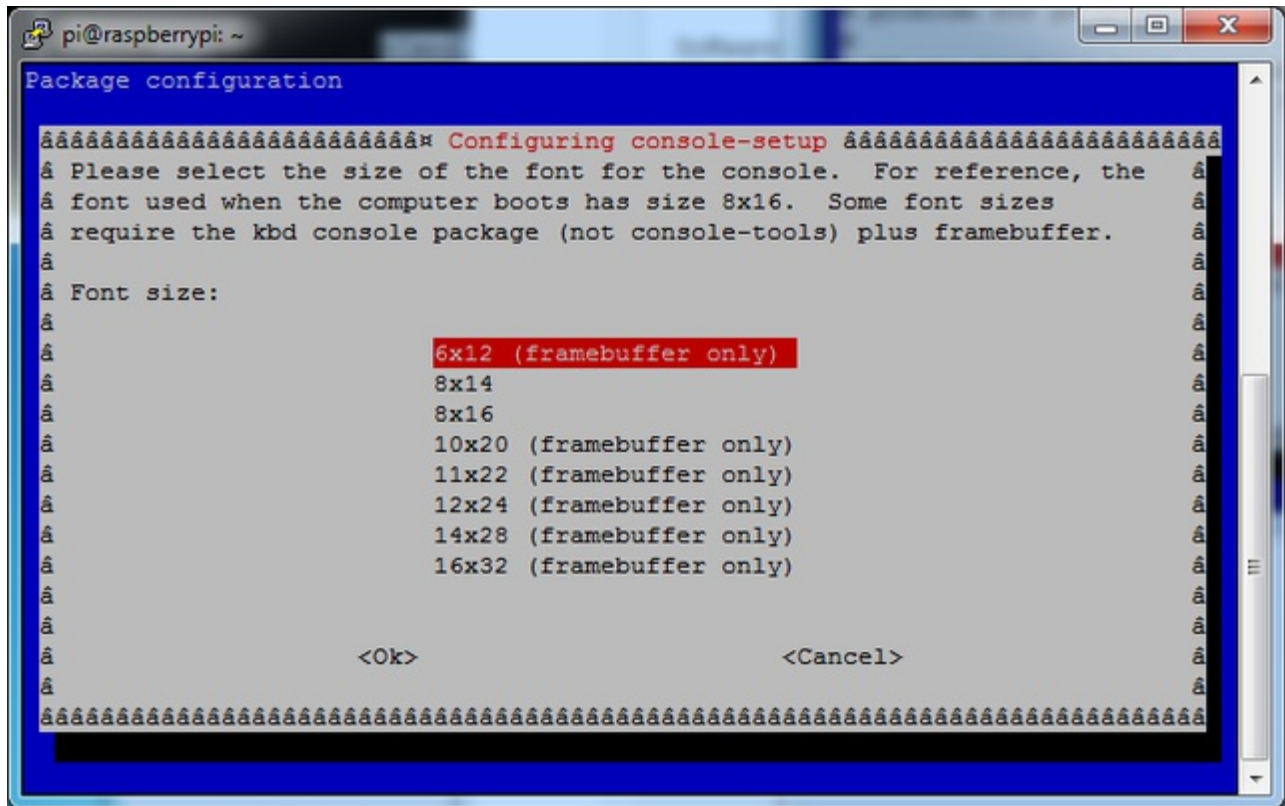
```
pi@raspberrypi:~$ cat /boot/cmdline.txt
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait fbcon=map:10 f
bcom=font:VGA8x8

pi@raspberrypi:~$
```

I think the VGA8x8 font is a bit chunky, you probably want 12x6 which is what is shown in the photo above. To change the font, run **sudo dpkg-reconfigure console-setup** and go thru to select Terminus 6x12







## Turn off Console Blanking

You may notice the console goes black after 30 minutes, this is a sort of 'power saving' or 'screensaver' feature.

### Raspbian Jessie

Add the following line to `/etc/rc.local`

```
sudo sh -c "TERM=linux setterm -blank 0 >/dev/tty0"
```

on the line before the `finalexit 0`

### Raspbian Wheezy

You can disable this by editing `/etc/kbd/config` and looking for

```
BLANK_TIME=30
```

and setting the blank time to 0 (which turns it off)

```
BLANK_TIME=0
```





# Userspace Tools

Major updates to Raspbian often **break PiTFT support**. The PiTFT kernel package doesn't work across different OS releases, and it's a *lot* of work to prepare a new one.

We've experimented with an alternate approach that **doesn't rely on a custom kernel** — it instead works in “**user space**.” So far it's worked well regardless of the OS version being used!

There are tradeoffs. The code is still in a rough state with many features yet to be implemented, and also the performance is slightly less than the kernel approach. It's typically adequate though, even for game emulation (RetroPie, etc.), so give it a try if you've had trouble with the “classic” approach.

This currently requires a bit of Linux-y knowledge, editing files and such...

## Download, Test and Install

PiTFT displays use **SPI** to communicate, so make sure that's enabled using the **raspi-config** utility:

```
sudo raspi-config
```

Menu options move around from time to time...at the time of this writing, SPI is under “Interfacing Options.”

Then retrieve the software using *wget*...

```
wget https://github.com/adafruit/Adafruit_Userspace_PiTFT/archive/master.zip
unzip master.zip
```

And then a quick test...

```
cd Adafruit_Userspace_PiTFT-master
sudo ./tftcp
```

The PiTFT should mirror the contents of the Raspberry Pi's HDMI output at this point. Text and everything will be microscopic, but we're just checking that the program runs. If not, confirm that the file **/dev/spidev0.0** exists — this should happen when SPI is enabled. Double-check *raspi-config* and it never hurts to reboot.

Does it run? Good. Press control+c to kill the program, and we'll set it up to run automatically



on boot.

First, copy the **tftcp** executable to **/usr/local/bin**:

```
sudo cp tftcp /usr/local/bin
```

Then edit the file **/etc/rc.local** as **root** (you can substitute your editor of preference for nano):

```
sudo nano /etc/rc.local
```

Just above the final “exit 0” line, insert the following line:

```
/usr/local/bin/tftcp &
```

The screen looks best if the HDMI resolution exactly matches the PiTFT resolution, so the final step is to configure the system for 320x240 video:

```
sudo nano /boot/config.txt
```

Append the following lines to the bottom of the file:

```
disable_overscan=1
hdmi_force_hotplug=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt=320 240 60 1 0 0 0
```

OPTIONAL: you can also use “640 480” in place of “320 240” above. This is exactly twice the PiTFT native resolution, and the tftcp utility will perform a smooth 2:1 filtering of the image. Any larger though and the image isn’t as sharp (and text becomes tiny, like when we first tested it).

Now **reboot** and the PiTFT should activate toward the end of the boot process.

## Resistive Touchscreen Support

This is even more experimental than the **tftcp** utility...it only works with the resistive screen, and there’s no calibration support yet, but if you’d like to try it out...

First there’s some prerequisite software to install:

```
sudo apt-get update
sudo apt-get install python-pip python-smbus python-dev
sudo pip install evdev
```

“cd” to the same directory where the software was downloaded earlier, and try it out...

```
cd Adafruit_Userspace_PiTFT-master
```

```
sudo python touchmouse.py
```

Whether you're in X11 or in text console mode (e.g. Raspbian Lite), the cursor should move in response to touch, which is emulating a mouse.

If that seems OK, press control+c to stop it and we'll use the same steps to make it auto-run on boot:

```
sudo cp touchmouse.py /usr/local/bin  
sudo nano /etc/rc.local
```

Insert this line just above the "exit 0" at the end of the file:

```
/usr/bin/python /usr/local/bin/touchmouse.py &
```

**reboot** and both PiTFT and touch should be active now.

# HELP! (FAQ)

My PiTFT used to work, now it doesn't!

Did you do an `apt-get upgrade` or `rpi-update`? This command will blow away our PiTFT kernel which means that you will no longer have PiTFT support, you will have to redo the **easy-install** steps to reinstall the kernel.

If you had already made a working PiTFT setup, you may be able to reinstall the Adafruit kernel like so:

```
sudo apt-get install raspberrypi-kernel=1.20161027-1
```

If it tells you that the latest version is already installed, try this instead:

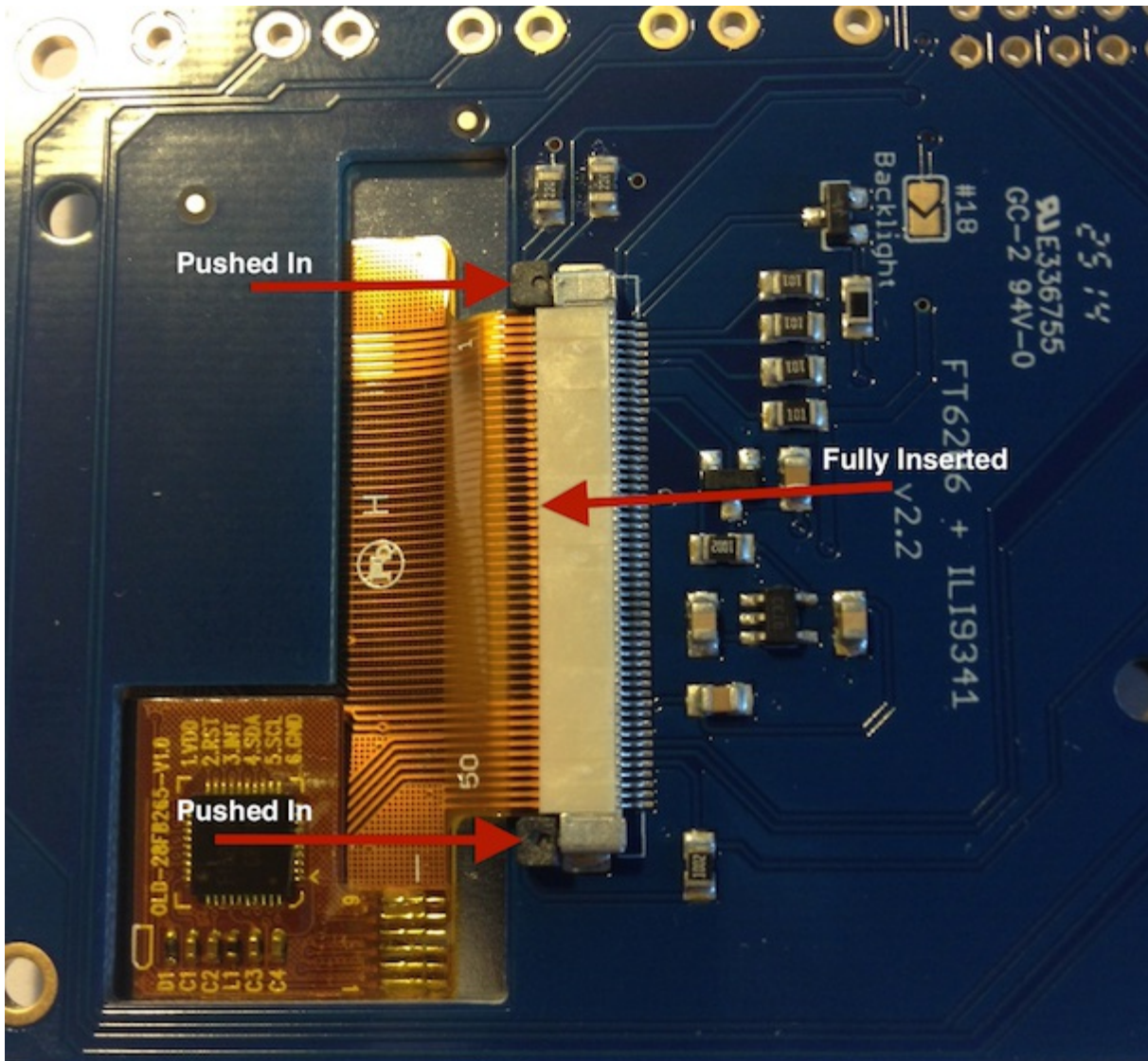
```
sudo apt-get install --reinstall raspberrypi-kernel=1.20161027-1
```

...you can [check here](https://adafru.it/eUK) (<https://adafru.it/eUK>) and substitute the most recent version you see in the `=1.20161027-1` part.

I'm booting my Pi with the PiTFT and the HDMI output 'locks up' during boot!

**It looks like the Pi is 'halting' or 'locking' up during boot** but what is really happening is the console is switching from the HDMI output to the PiTFT output.

Check your PiTFT connections, particularly make sure you seated the PiTFT on the Pi properly, nothing is in the way, and the TFT flex connector is seated properly.



My PiTFT works for a bit and then I get a black screen with a short line of white pixels in one corner

Sounds like you tried to configure your Pi to 'boot straight to X', that is, start up the graphics interface on boot. This doesn't work by default because the Pi operating system is not expecting a PiTFT so it boots to the HDMI output. See below for how to set up your Pi to boot to X on the PiTFT

To 'fix' this, you can either connect an HDMI monitor, then in a terminal window run **sudo raspi-config** and configure the Pi to boot to the command line not X! If you do not have an HDMI monitor, you can also try a console cable

How can I force the Pi to bring up X on the HDMI/TV monitor?

There's two ways to do it. In older Pi installs, use the **fb0** framebuffer when you want to display stuff on the HDMI/TV display, for example:

**FRAMEBUFFER=/dev/fb0 startx**

will use the HDMI/TV framebuffer for X windows instead of the PiTFT

On Jessie Pi installs, run

**sudo nano /usr/share/X11/xorg.conf.d/99-fbdev.conf**

to edit the configuration file and make sure it contains:

```
Section "Device"
  Identifier "display"
  Driver "fbdev"
  Option "fbdev" "/dev/fb0"
EndSection
```

change the Option "fbdev" "/dev/fb0" line to Option "fbdev" "/dev/fb1" if you want the xdisplay on the PiTFT

I'm trying to run startx and I get FATAL: Module g2d\_23 not found.

don't forget you have to remove the turbo file!

**sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~**

How come OMX-Player and Minecraft and other programs don't appear on the PiTFT display?

Some programs are graphics-optimized, particularly the video playback tools and some other programs like Minecraft. They write 'directly' to the HDMI output, and cannot write to the PiTFT so there is no way to directly make them work. However, you *can* have the output go to HDMI and then mirror the HDMI onto the PiTFT with **fbcp**. [See this tutorial for more details \(https://adafru.it/fbe\)](https://adafru.it/fbe)

Why doesn't the tactile button on GPIO #21 work?

On some older PiTFTs we had one of the buttons labeled #21 - that's the original RasPi name for that pin. If you're using a V2 (chance is, you are!) that is now called #27.

All the PiTFT's we ship now have the button labeled #21 and #27

I want better performance and faster updates!

You can change the SPI frequency (overclock the display) by editing **boot/config.txt** and changing the **dtoverlay** options line to:

**dtoverlay=pitft28r,rotate=90,speed=62000000,fps=25**

Or whatever you like for speed, rotation, and frames-per-second. BUT, here's the thing, the Pi only supports a *fixed number* of SPI frequencies. So tweaking the number a little won't do anything. The kernel will round the number to the closest value. You will always get frequencies that are 250MHz divided by an even number. Here's the only SPI frequencies this kernel supports

- 15,625,000 (a.k.a 16000000 = 16 MHz)
- 17,857,142 (a.k.a. 18000000 = 18 MHz)
- 20,833,333 (a.k.a 21000000 = 21 MHz)



- 25,000,000 (= 25 MHz)
- 31,250,000 (a.k.a 32000000 = 32MHz)
- 41,666,666 (a.k.a 42000000 = 42MHz)
- 62,500,000 (a.k.a 62000000 = 62MHz)

So if you put in 48000000 for the speed, you won't actually get 48MHz, you'll actually only get about 42MHz because it gets rounded down. We tested this display nicely with 32MHz and we suggest that. But you can put in 42MHz or even try 62MHz and it will update faster

You can tweak fps (frames per second) from 20 to 60 and frequency up to 62MHz for tradeoffs in performance and speed. Reboot after each edit to make sure the settings are loaded properly. There's a trade off that if you ask for higher FPS you're going to load the kernel more because it's trying to keep the display updated.

How can I take screenshots of the little screen?

[We took the screenshots for this tutorial with](#)

[fbgra](https://adafru.it/diV) ([b](https://adafru.it/diV)) ([b](https://adafru.it/diV))

```
wget http://fbgrab.monells.se/fbgrab-1.2.tar.gz (https://adafru.it/diW)
tar -zxvf fbgrab*.gz
cd fbgrab/
make
./fbgrab screenshot.png
```

```
COM3 - PuTTY
pi@raspberrypi:~$ wget http://fbgrab.monells.se/fbgrab-1.2.tar.gz
--2014-04-21 19:26:22-- http://fbgrab.monells.se/fbgrab-1.2.tar.gz
Resolving fbgrab.monells.se (fbgrab.monells.se)... 66.33.214.148
Connecting to fbgrab.monells.se (fbgrab.monells.se)|66.33.214.148|:80... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 12836 (13K) [application/x-tar]
Saving to: `fbgrab-1.2.tar.gz'

100%[=====>] 12,836      --.-K/s   in 0.03s

2014-04-21 19:26:22 (497 KB/s) - `fbgrab-1.2.tar.gz' saved [12836/12836]

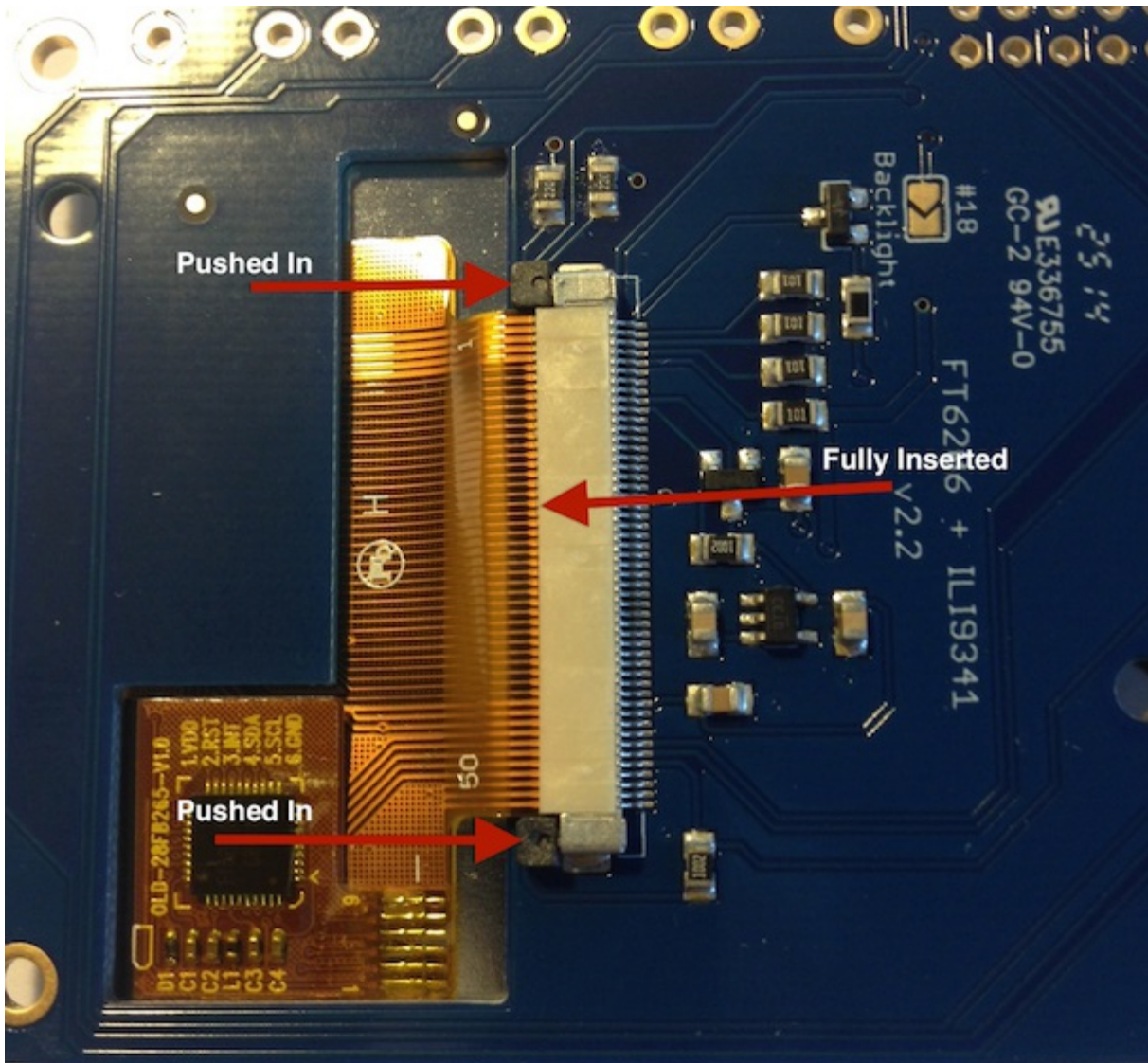
pi@raspberrypi:~$ tar -zxvf fbgrab-1.2.tar.gz
fbgrab/
fbgrab/fbgrab.c
fbgrab/INSTALL
fbgrab/fbgrab.1.man
fbgrab/COPYING
fbgrab/Makefile
pi@raspberrypi:~$ cd fbgrab/
pi@raspberrypi:~/fbgrab$ make
cc -g -Wall   fbgrab.c -lpng -lz -o fbgrab
gzip --best --to-stdout fbgrab.1.man > fbgrab.1.gz
pi@raspberrypi:~/fbgrab$ ./fbgrab
Usage:   ./fbgrab      [-hi] [-{C|c} vt] [-d dev] [-s n] [-z n]
          [-f fromfile -w n -h n -b n] filename.png
pi@raspberrypi:~/fbgrab$ ./fbgrab filemanager.png
Resolution: 320x240 depth 16
Converting image from 16
Now writing PNG file (compression -1)
```

How do I automatically boot to X windows on the PiTFT?

Once you have a PiTFT installation setup you can add a custom X windows configuration to use the PiTFT by default. Then you can use the normal raspi-config boot to console/desktop options to control if the Pi boots to a console or desktop. [See the detailed instructions on this page of the guide \(https://adafruit.it/ird\)](https://adafruit.it/ird) for more information.

My screen isn't working/works erratically/looks funny

Check to make sure that the flat flex cable is fully seated in the connector and the 'ears' are pushed in to secure it. See the picture for what it should look like:



On my first run of startx I get a window saying "GDBus Error.org.Freedesktop Policy Kit1 Error: Failed Cannot determine user of subject"

This happens on the Raspberry Pi the first time you run startx, no matter what display. You can just re-start X and it wont appear again.

Can I get a right-click from the touch-screen?

Yes! Please see this post:

<https://forums.adafruit.com/viewtopic.php?f=47&t=77528&p=393280#p393322> (<https://adafru.it/fH3>)

I'm having difficulties with the STMPE resistive touch screen controller

[Here's a hack for the device tree overlay that can force different SPI modes, sometimes that helps!](https://adafru.it/fEw) (<https://adafru.it/fEw>)

My PiTFT's rotation/calibration isn't working in X11

X11 (the graphical system) has changed how it gets touchscreen input, so if you rotate the display and the calibration isn't being picked up:

Check `/usr/share/X11/xorg.conf.d` for a file called `10-evdev.conf`

If you don't see that file

1. You need to `sudo apt-get install xserver-xorg-input-evdev`, and then...
2. If you do have a `40-libinput.conf` in that same directory, you must remove it even if/once `evdev` is installed, since it will override the `10-evdev.conf` otherwise.

[Thanks to cerebrate in the forums for the hint!](https://adafru.it/fEw)(<https://adafru.it/fEw>)



# Playing Videos



## How To Play Videos

You can play many types of videos on the screen, using mplayer you don't even need to run X and you can script the movies to play using Python. We'll show you how to just play one video for now.

To demo, we'll use an mp4 of Big Buck Bunny for 320 pixel wide screens. Below we show you how to create/resize videos, but to make it easy, just download our version with:

```
wget http://adafruit-download.s3.amazonaws.com/bigbuckbunny320p.mp4 (https://adafru.it/cXR)
```

The video is 30MB which is a lot if you haven't expanded your SD card yet. Before you do this, run `sudo raspi-config` to expand the SD card so you don't run out of space!

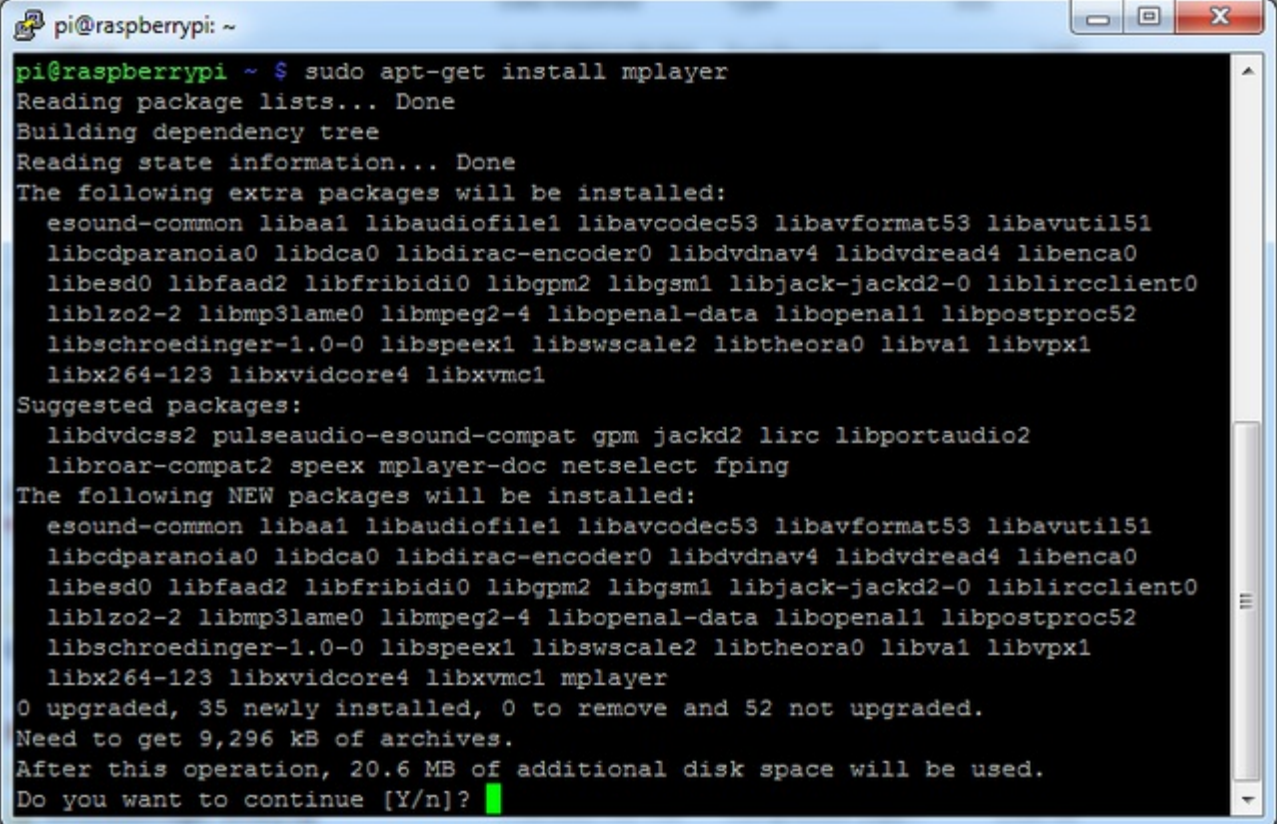


If you don't have **mplayer** yet, run

```
sudo apt-get update
```

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

to install it. It may take a few minutes to complete

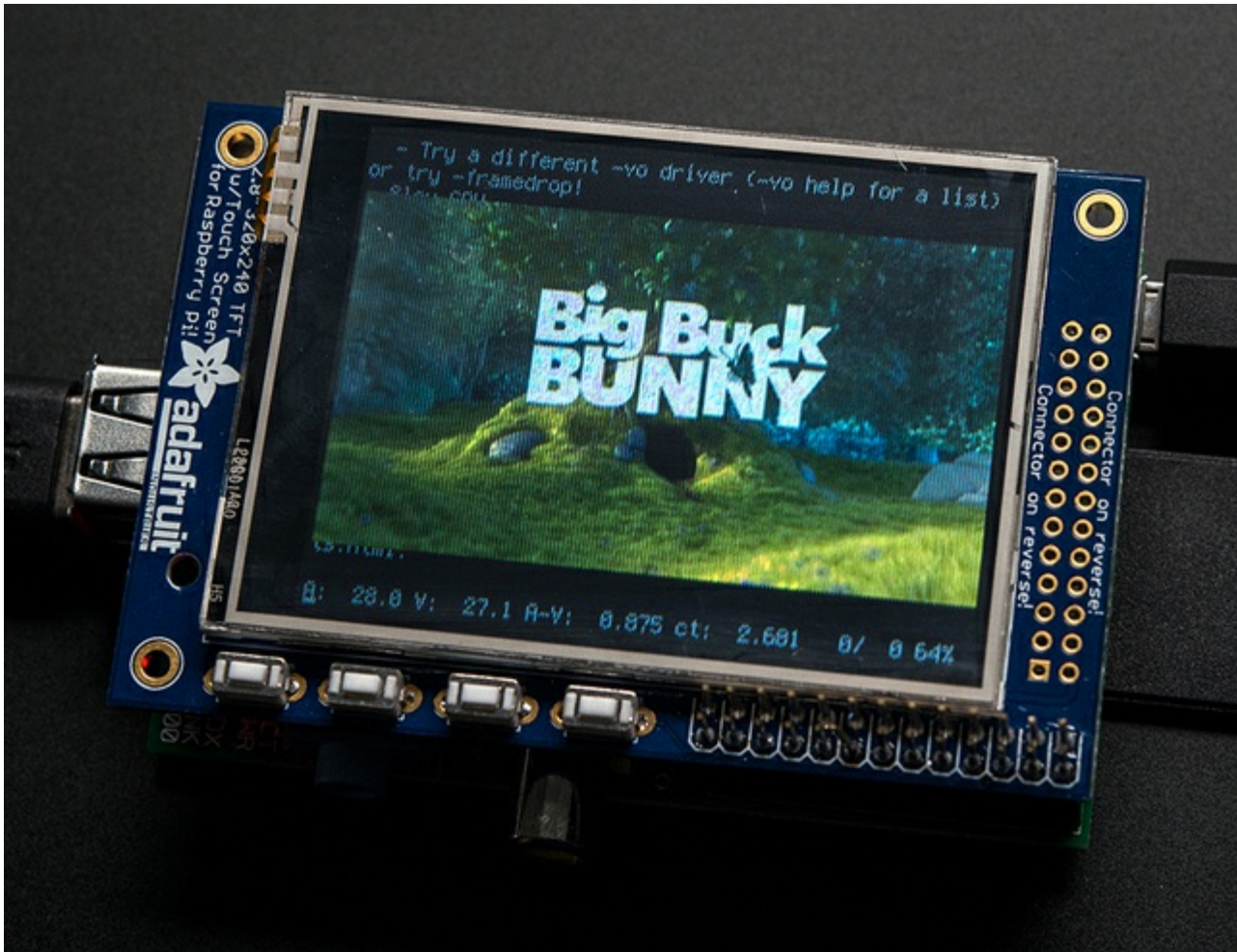
A terminal window titled 'pi@raspberrypi: ~' showing the command 'sudo apt-get install mplayer' and its output. The output lists various extra packages to be installed, suggested packages, and the new packages to be installed. It also shows the disk space requirements and asks for confirmation to continue.

```
pi@raspberrypi ~ $ sudo apt-get install mplayer
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  esound-common libaa1 libaudiofile1 libavcodec53 libavformat53 libavutil51
  libcdparanoia0 libdca0 libdirac-encoder0 libdvdnav4 libdvdread4 libenca0
  libesd0 libfaad2 libfribidi0 libgpm2 libgsm1 libjack-jackd2-0 liblircclient0
  liblzo2-2 libmp3lame0 libmpeg2-4 libopenal-data libopenal1 libpostproc52
  libschroedinger-1.0-0 libspeex1 libswscale2 libtheora0 libva1 libvpx1
  libx264-123 libxvidcore4 libxvmc1
Suggested packages:
  libdvdcss2 pulseaudio-esound-compat gpm jackd2 lirc libportaudio2
  libroar-compat2 speex mplayer-doc netselect fping
The following NEW packages will be installed:
  esound-common libaa1 libaudiofile1 libavcodec53 libavformat53 libavutil51
  libcdparanoia0 libdca0 libdirac-encoder0 libdvdnav4 libdvdread4 libenca0
  libesd0 libfaad2 libfribidi0 libgpm2 libgsm1 libjack-jackd2-0 liblircclient0
  liblzo2-2 libmp3lame0 libmpeg2-4 libopenal-data libopenal1 libpostproc52
  libschroedinger-1.0-0 libspeex1 libswscale2 libtheora0 libva1 libvpx1
  libx264-123 libxvidcore4 libxvmc1 mplayer
0 upgraded, 35 newly installed, 0 to remove and 52 not upgraded.
Need to get 9,296 kB of archives.
After this operation, 20.6 MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

OK now you just have to run:

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

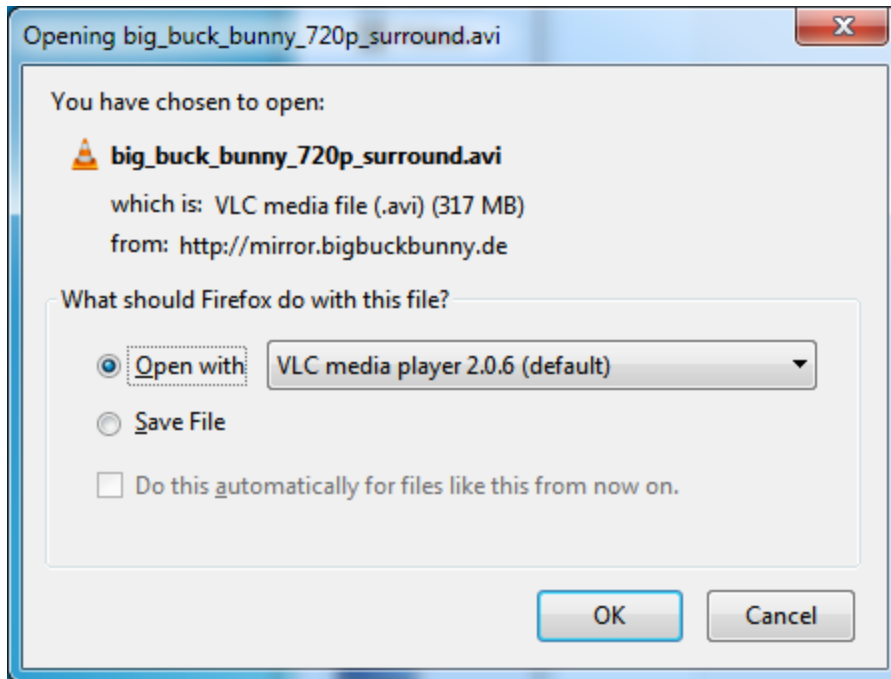
If your video is not sized for 320 wide, you may need to add `a-zoom` after `-framedrop` so that it will resize - note that this is quite taxing for the Pi, so it may result in a choppy or mis-synced video!



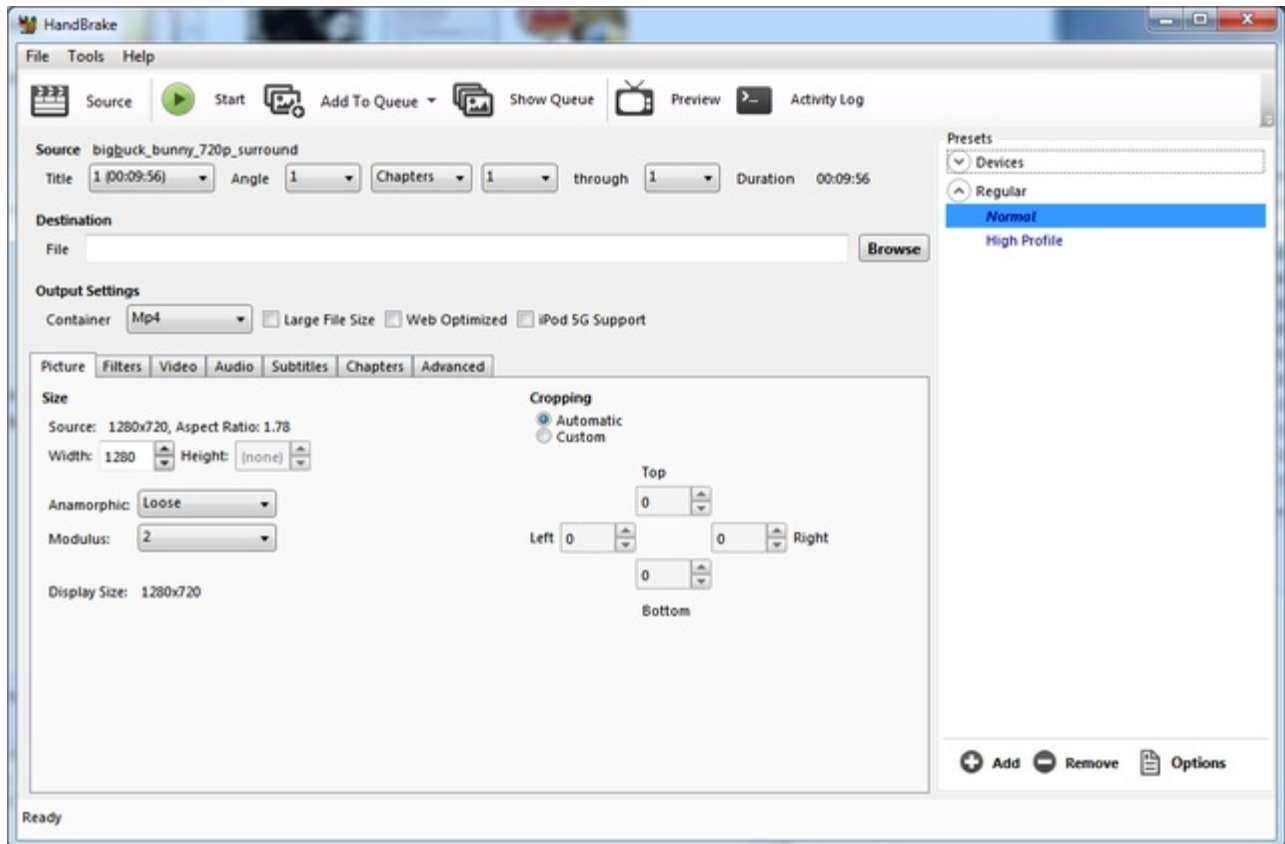
## Converting/Resizing Videos

It's possible to play full length videos on the TFT plate, but since the screen is small and the Pi can't use hardware acceleration to play the videos it's best to scale them down to 320x240 pixels. This will be easier for the Pi to play and also save you tons of storage space. For this demo, we'll be using the famous [Big Buck Bunny](https://adafru.it/cXS) (<https://adafru.it/cXS>) video, which is creative commons and also very funny!

You can download it from the link above, we'll be using the 720p AVI version.

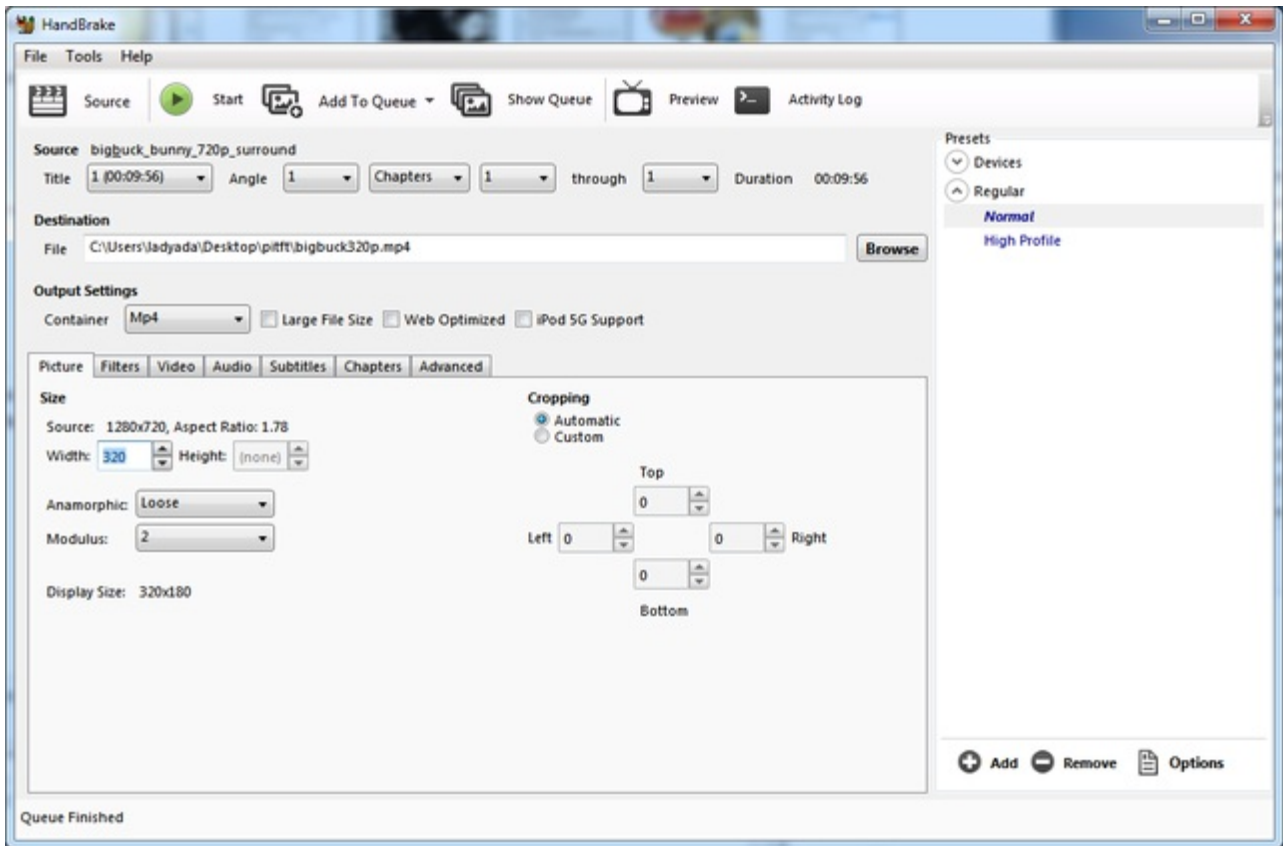


To do the conversion itself, we suggest [HandBrake](https://adafru.it/cXT) (<https://adafru.it/cXT>) which works great and is open source so it runs on all operating systems! Download and install from the link. Then run the installed application and open up the AVI file from before. The app will pre-fill a bunch of information about it.

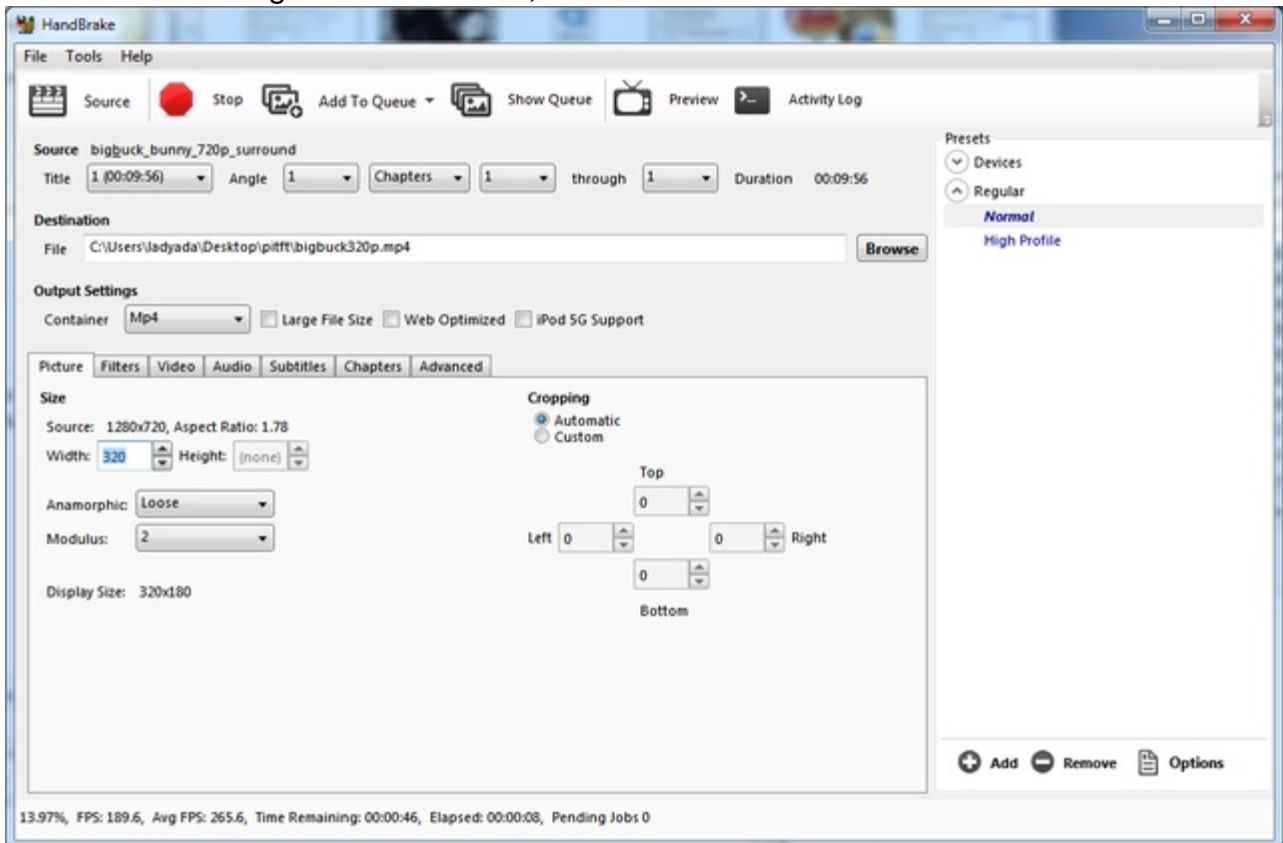


Under **Destination** click **Browse...** to select a new MP4 file to save. Then under **Picture** change the **Width** to 320 (the height will be auto-calculated)





Click **START** to begin the conversion, it will take a minute or two.



That's it! You now have a smaller file. Don't forget to play it on your computer to make sure it

plays right before copying it to your Pi

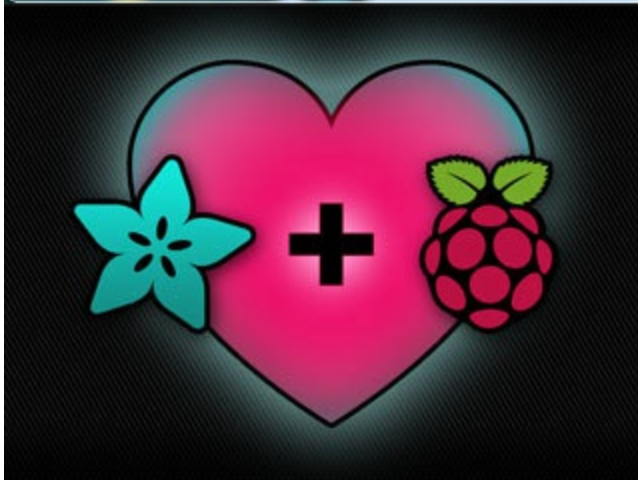


# Displaying Images

You can display every day images such as GIFs, JPGs, BMPs, etc on the screen. To do this we'll install **fbi** which is the **frame buffer image** viewer (not to be confused with the FBI agency!)

**sudo apt-get install fbi** will install it

```
COM3 - PuTTY
pi@raspberrypi:~$ sudo apt-get install fbi
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  imagemagick
The following NEW packages will be installed:
  fbi
0 upgraded, 1 newly installed, 0 to remove and 52 not upgraded.
Need to get 59.7 kB of archives.
After this operation, 157 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main fbi armhf 2.07-10 [59.7 kB]
Fetched 59.7 kB in 1s (40.0 kB/s)
Selecting previously unselected package fbi.
(Reading database ... 64758 files and directories currently installed.)
Unpacking fbi (from .../archives/fbi_2.07-10_armhf.deb) ...
Processing triggers for mime-support ...
Processing triggers for man-db ...
Setting up fbi (2.07-10) ...
pi@raspberrypi:~$
```



Grab our lovely wallpapers with

wget <http://adafruit-download.s3.amazonaws.com/adapiluv320x240.jpg>  
 wget <http://adafruit-download.s3.amazonaws.com/adapiluv480x320.png> (<https://adafru.it/cXU>)

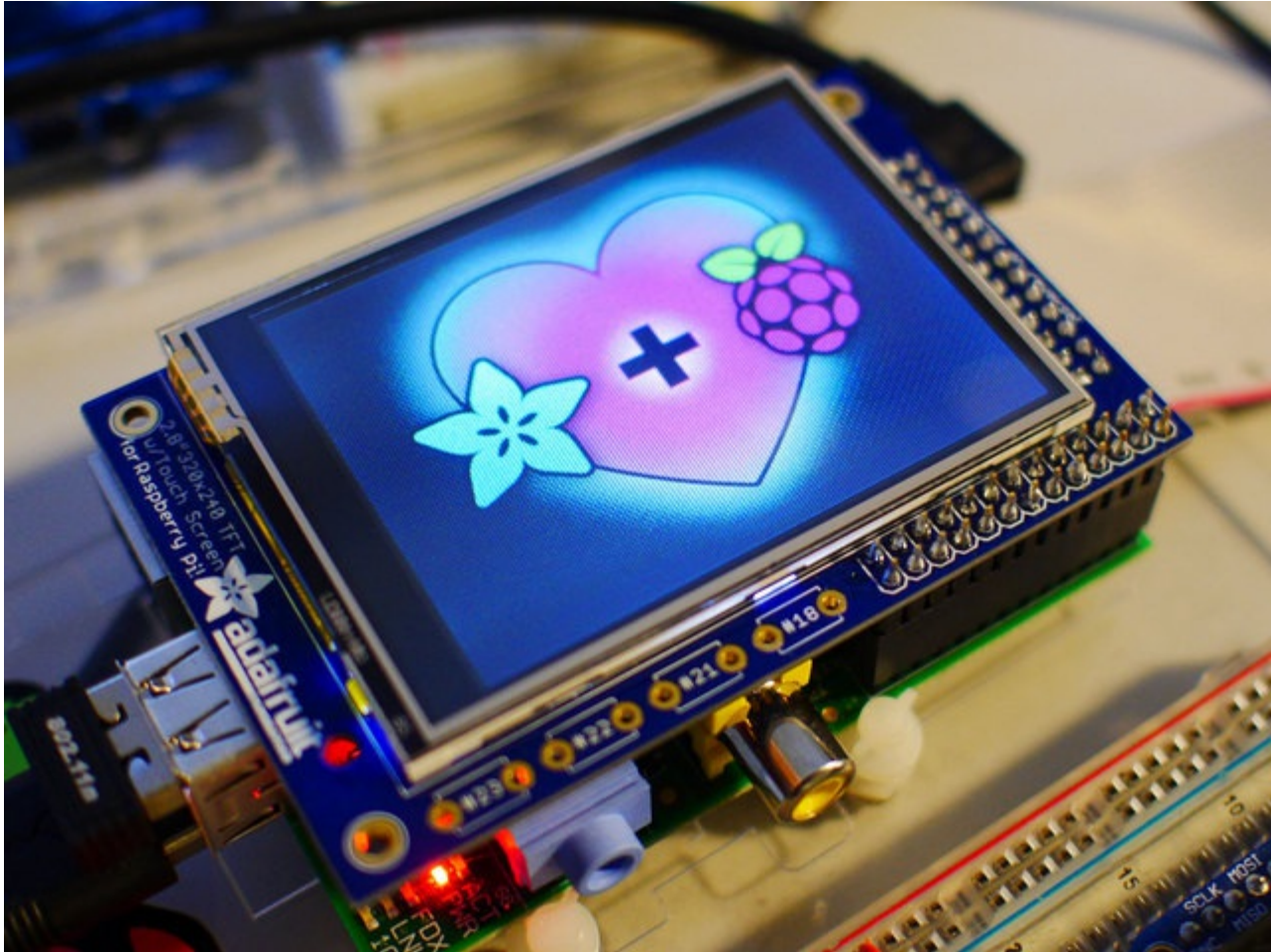
For 320x240 PiTFTs (2.2", 2.4", 2.8" or 3.2") view it with

```
sudo fbi -T 2 -d /dev/fb1 -noverbose -a adapiluv320x240.jpg
```

or for 3.5" PiTFTs:

```
sudo fbi -T 2 -d /dev/fb1 -noverbose -a adapiluv480x320 (https://adafru.it/cXU).jpg
```

That's it!



# Using FBCP



**The Ideal:** Adafruit's PiTFT displays are razor sharp. Whereas small composite screens on the Raspberry Pi usually require some video scaling (resulting in blurriness), PiTFT uses the GPIO header, digitally controlled pixel-by-pixel for a rock steady image. Though not a *lot* of pixels, it works great for retro gaming (and the display neatly stacks above the board, no side protuberances for video cables).

**The Downside:** this GPIO link entirely bypasses the Pi's video hardware, including the graphics accelerator. Many games and emulators *depend* on the GPU for performance gains. So the PiTFT has traditionally been limited to just a subset of specially-compiled emulators that can work and run well enough without the GPU.

**The Solution:** our latest PiTFT drivers, along with a tool called *fbcp* (framebuffer copy), careful system configuration, and (optionally) the more potent Raspberry Pi 2 board open the doors to many more gaming options. Existing emulator packages (such as RetroPie, with *dozens* of high-performance emulators and ports) — previously off-limits to the PiTFT — can run quite effectively now!

[Click here to go to our FBCP tutorial!](https://adafruit.com/blog/2016/07/20/fbcp-tutorial/)

<https://adafruit.com/blog/2016/07/20/fbcp-tutorial/>



# Backlight Control

The backlight of the 2.8" PiTFT has 4 LEDs in series and it draws ~75mA at all times, controlled by a transistor. The PiTFT 3.5" display has 6 LEDs in a row, and we use a boost converter to get the 5V from the Pi up to the ~20V needed to light up all the LEDs.

There might be times you'd like to save some power and turn off the backlight. The screen and touchplate will still work, you just can't see anything. We designed the board with the STMPE610 touchscreen controller which has 2 extra GPIO and tied one of them to control the backlight. You can use the command line to control the backlight.

By default, the backlight's on...but you can control it in two ways!

## PWM Backlight Control with GPIO 18

If you want precise control, you can use the PWM output on GPIO 18. There's python code for controlling the PWM but you can also just use the kernel module and shell commands.

You'll need to make sure the STMPE control is not 'active' as the STMPE GPIO overrides the PWM output.

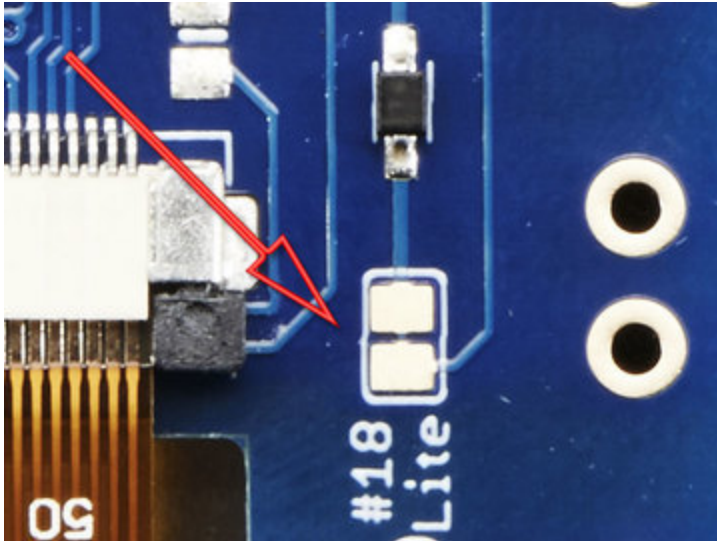
```
sudo sh -c 'echo "1" > /sys/class/backlight/soc\:\backlight/brightness'
```

(Or if you are running an old kernel before the backlight object, try **`sudo sh -c "echo 'in' > /sys/class/gpio/gpio508/direction"`**)

OK now you can set the GPIO #18 pin to PWM mode using WiringPi's **`gpio`** command

With these basic shell commands, you can set the GPIO #18 pin to PWM mode with 1000 Hz frequency, set the output to 100 (out of 1023, so dim!), set the output to 1023 (out of 1023, nearly all the way on) and 0 (off)

```
gpio -g mode 18 pwm
gpio pwmfc 1000
gpio -g pwm 18 100
gpio -g pwm 18 1023
gpio -g pwm 18 0
```



If you'd like to not have #18 control the backlight, simply cut the solder jumper, the tiny trace between the two large gold pads marked **Lite #18**

## On / Off Using STMPE GPIO

Another option is to just turn it on and off using the extra GPIO created by the touchscreen driver

Thanks to the raspberry Pi overlay system, this GPIO is already set up for you in a file called **/sys/class/backlight/soc:backlight/brightness**

To turn the backlight off run

```
sudo sh -c 'echo "0" > /sys/class/backlight/soc:backlight/brightness'
```

To turn it back on, run

```
sudo sh -c 'echo "1" > /sys/class/backlight/soc:backlight/brightness'
```

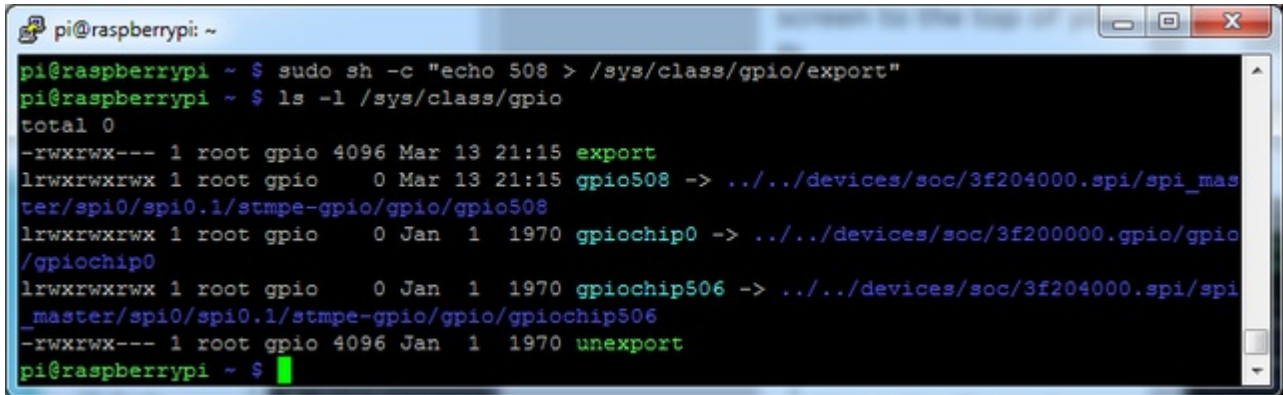
## For older versions of PiTFT Kernel

On older versions of the PiTFT kernel/overlay, the GPIO was not tied to the backlight device. Start by getting access to the GPIO by making a device link

```
sudo sh -c "echo 508 > /sys/class/gpio/export"
ls -l /sys/class/gpio
```

For some *really* old versions, the GPIO pin was #252 not #508 so substitute that if you're running something from 2014 or earlier



A terminal window on a Raspberry Pi. The prompt is 'pi@raspberrypi: ~'. The user enters 'sudo sh -c "echo 508 > /sys/class/gpio/export"'. The prompt changes to 'pi@raspberrypi ~ \$'. The user enters 'ls -l /sys/class/gpio'. The output shows a directory listing for /sys/class/gpio, including files like 'export', 'gpio508', 'gpiochip0', 'gpiochip506', and 'unexport'.

```
pi@raspberrypi ~ $ sudo sh -c "echo 508 > /sys/class/gpio/export"
pi@raspberrypi ~ $ ls -l /sys/class/gpio
total 0
-rwxrwx--- 1 root gpio 4096 Mar 13 21:15 export
lrwxrwxrwx 1 root gpio  0 Mar 13 21:15 gpio508 -> ../../devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-gpio/gpio/gpio508
lrwxrwxrwx 1 root gpio  0 Jan  1 1970 gpiochip0 -> ../../devices/soc/3f200000.gpio/gpio/gpiochip0
lrwxrwxrwx 1 root gpio  0 Jan  1 1970 gpiochip506 -> ../../devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-gpio/gpio/gpiochip506
-rwxrwx--- 1 root gpio 4096 Jan  1 1970 unexport
pi@raspberrypi ~ $
```

Once you verify that you see GPIO #508, then you can set it to an output, this will turn off the display since it will output 0 by default

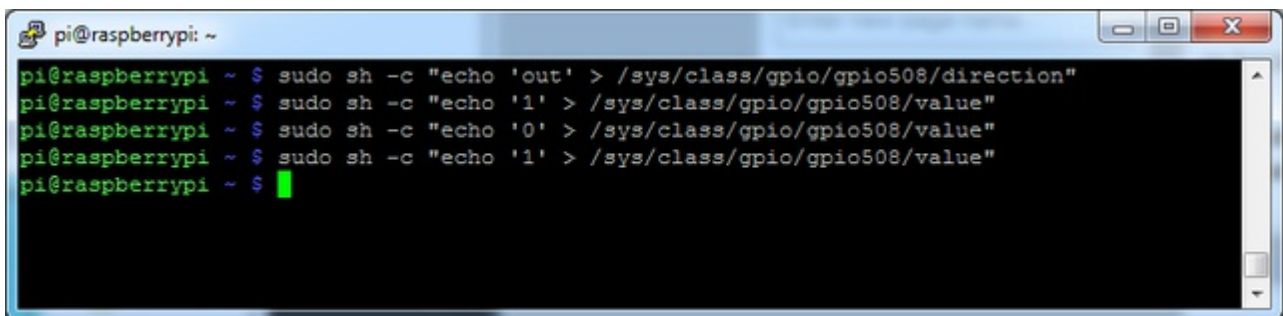
```
sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"
```

Then turn the display back on with

```
sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
```

or back off

```
sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"
```

A terminal window on a Raspberry Pi showing a sequence of commands. The prompt is 'pi@raspberrypi: ~'. The user enters 'sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"', then 'sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"', then 'sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"', and finally 'sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"'.

```
pi@raspberrypi ~ $ sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"
pi@raspberrypi ~ $ sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $ sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $ sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $
```

# PiTFT PyGame Tips

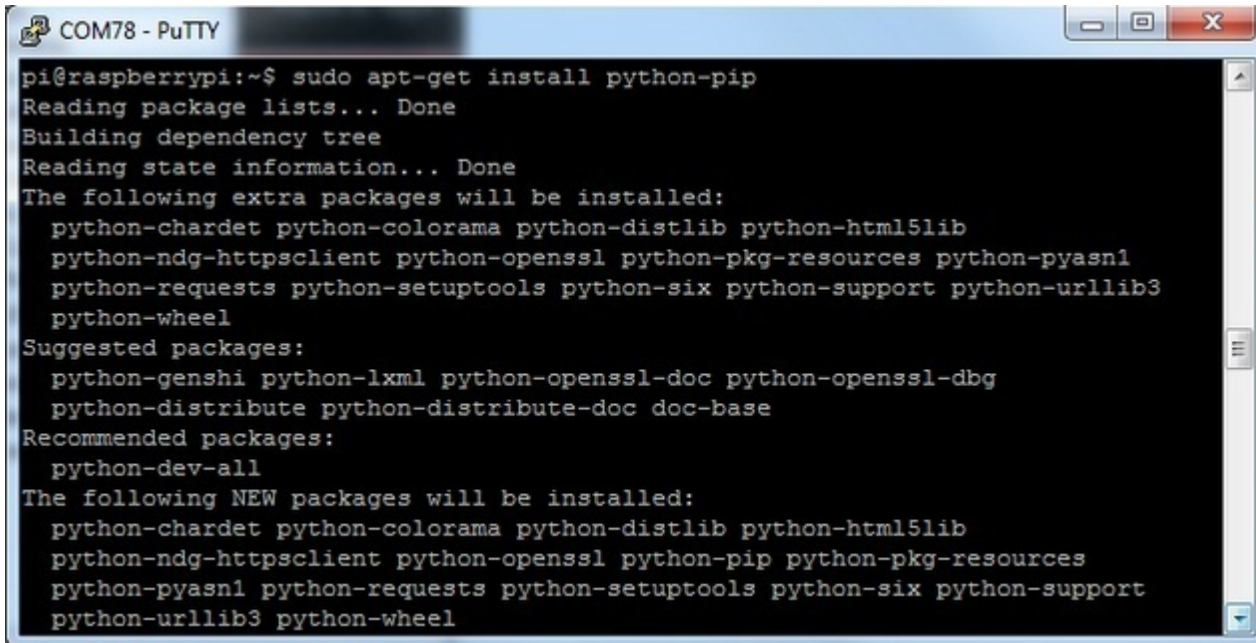
Since the PiTFT screen is fairly small, you may need to write custom UI programs. Pygame is the easiest way by far to do this.

[Jeremy Blythe has an excellent tutorial here on getting started.](https://adafru.it/saw) (<https://adafru.it/saw>)

However, *before* you follow that link you'll want to set up pygame for the best compatibility:

## Install pip & pygame

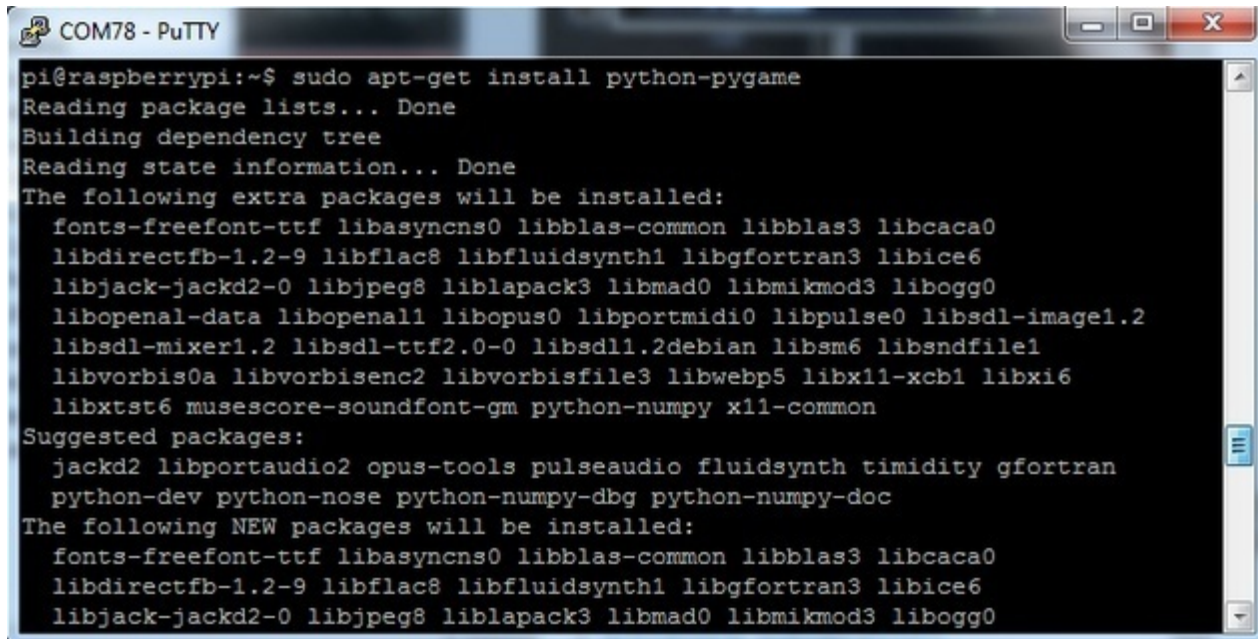
Install Pip: **sudo apt-get install python-pip**



```
COM78 - PuTTY
pi@raspberrypi:~$ sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  python-chardet python-colorama python-distlib python-html5lib
  python-ndg-httpsclient python-openssl python-pkg-resources python-pyasnl
  python-requests python-setuptools python-six python-support python-urllib3
  python-wheel
Suggested packages:
  python-genshi python-lxml python-openssl-doc python-openssl-dbg
  python-distribute python-distribute-doc doc-base
Recommended packages:
  python-dev-all
The following NEW packages will be installed:
  python-chardet python-colorama python-distlib python-html5lib
  python-ndg-httpsclient python-openssl python-pip python-pkg-resources
  python-pyasnl python-requests python-setuptools python-six python-support
  python-urllib3 python-wheel
```

Install Pygame: **sudo apt-get install python-pygame**

(this will take a while)



```
pi@raspberrypi:~$ sudo apt-get install python-pygame
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  fonts-freefont-ttf libasyncns0 libblas-common libblas3 libcaca0
  libdirectfb-1.2-9 libflac8 libfluidsynth1 libgfortran3 libice6
  libjack-jackd2-0 libjpeg8 liblapack3 libmad0 libmikmod3 libogg0
  libopenal-data libopenal1 libopus0 libportmidi0 libpulse0 libSDL-image1.2
  libSDL-mixer1.2 libSDL-ttf2.0-0 libSDL1.2debian libsm6 libsndfile1
  libvorbis0a libvorbisenc2 libvorbisfile3 libwebp5 libx11-xcb1 libxi6
  libxtst6 musescore-soundfont-gm python-numpy x11-common
Suggested packages:
  jackd2 libportaudio2 opus-tools pulseaudio fluidsynth timidity gfortran
  python-dev python-nose python-numpy-dbg python-numpy-doc
The following NEW packages will be installed:
  fonts-freefont-ttf libasyncns0 libblas-common libblas3 libcaca0
  libdirectfb-1.2-9 libflac8 libfluidsynth1 libgfortran3 libice6
  libjack-jackd2-0 libjpeg8 liblapack3 libmad0 libmikmod3 libogg0
```

## Ensure you are running SDL 1.2

SDL 2.x and SDL 1.2.15-10 have some serious incompatibilities with touchscreen. You can force SDL 1.2 by running a script. ([Thanks to heine in the forums!\(https://adafru.it/sax\)](https://adafru.it/sax))

Edit a new file with **sudo nano installsdl.sh**  
and paste in the following text:

```
#!/bin/bash

#enable wheezy package sources
echo "deb http://archive.raspbian.org/raspbian wheezy main"
"> /etc/apt/sources.list.d/wheezy.list

#set stable as default package source (currently jessie)
echo "APT::Default-release \"stable\"";
"> /etc/apt/apt.conf.d/10defaultRelease

#set the priority for libsdl from wheezy higher then the jessie package
echo "Package: libsdl1.2debian
Pin: release n=jessie
Pin-Priority: -10
Package: libsdl1.2debian
Pin: release n=wheezy
Pin-Priority: 900"
"> /etc/apt/preferences.d/libSDL

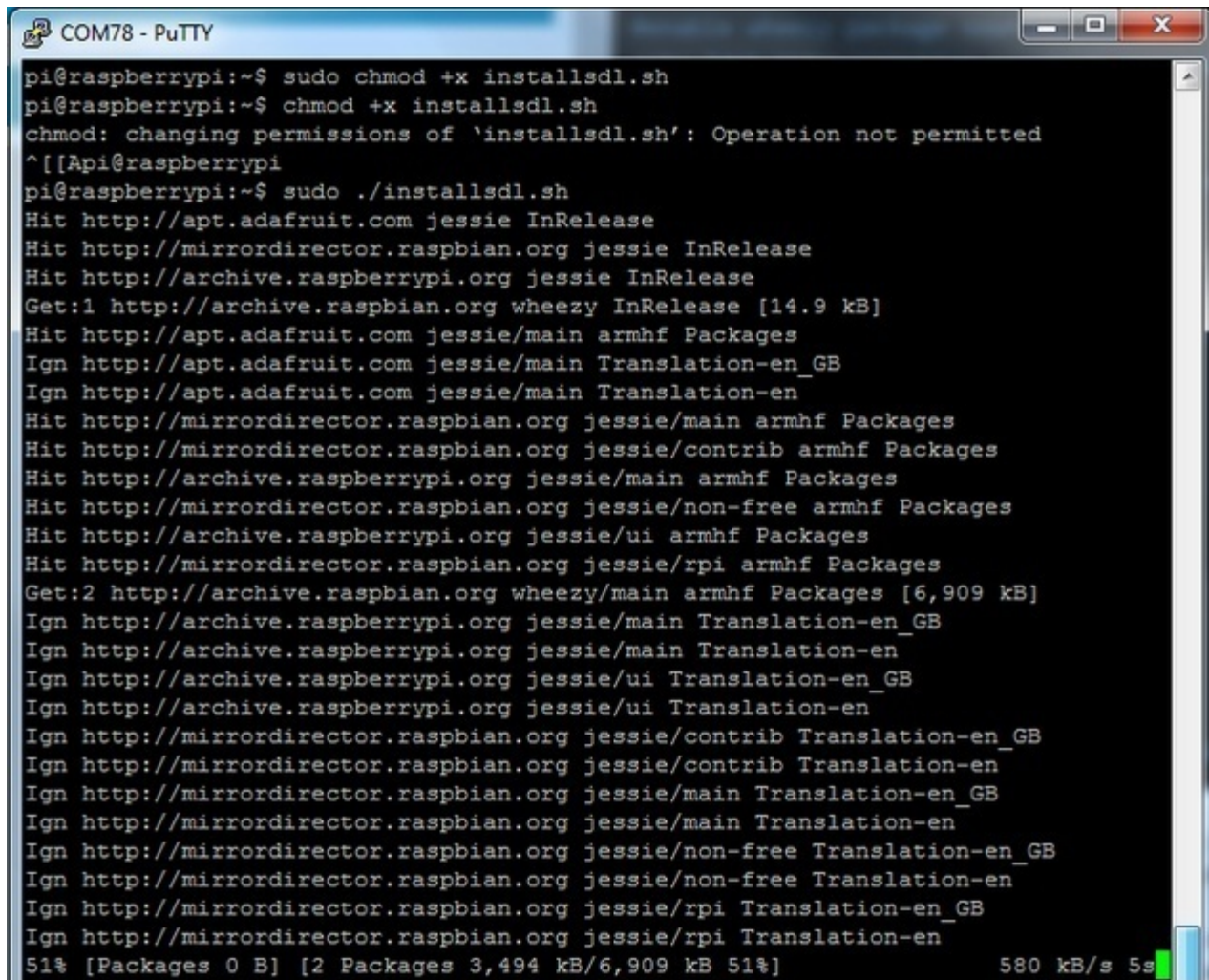
#install
apt-get update
apt-get -y --force-yes install libsdl1.2debian/wheezy
```



run

**sudo chmod +x installsdl.sh**

**sudo ./installsdl.sh**



```
pi@raspberrypi:~$ sudo chmod +x installsdl.sh
pi@raspberrypi:~$ chmod +x installsdl.sh
chmod: changing permissions of 'installsdl.sh': Operation not permitted
^[[Api@raspberrypi
pi@raspberrypi:~$ sudo ./installsdl.sh
Hit http://apt.adafruit.com jessie InRelease
Hit http://mirrordirector.raspbian.org jessie InRelease
Hit http://archive.raspberrypi.org jessie InRelease
Get:1 http://archive.raspbian.org wheezy InRelease [14.9 kB]
Hit http://apt.adafruit.com jessie/main armhf Packages
Ign http://apt.adafruit.com jessie/main Translation-en_GB
Ign http://apt.adafruit.com jessie/main Translation-en
Hit http://mirrordirector.raspbian.org jessie/main armhf Packages
Hit http://mirrordirector.raspbian.org jessie/contrib armhf Packages
Hit http://archive.raspberrypi.org jessie/main armhf Packages
Hit http://mirrordirector.raspbian.org jessie/non-free armhf Packages
Hit http://archive.raspberrypi.org jessie/ui armhf Packages
Hit http://mirrordirector.raspbian.org jessie/rpi armhf Packages
Get:2 http://archive.raspbian.org wheezy/main armhf Packages [6,909 kB]
Ign http://archive.raspberrypi.org jessie/main Translation-en_GB
Ign http://archive.raspberrypi.org jessie/main Translation-en
Ign http://archive.raspberrypi.org jessie/ui Translation-en_GB
Ign http://archive.raspberrypi.org jessie/ui Translation-en
Ign http://mirrordirector.raspbian.org jessie/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/contrib Translation-en
Ign http://mirrordirector.raspbian.org jessie/main Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/main Translation-en
Ign http://mirrordirector.raspbian.org jessie/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/non-free Translation-en
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en
51% [Packages 0 B] [2 Packages 3,494 kB/6,909 kB 51%] 580 kB/s 5s
```

it will force install SDL 1.2

```
COM78 - PuTTY
Ign http://archive.raspbian.org wheezy/main Translation-en_GB
Ign http://archive.raspbian.org wheezy/main Translation-en
Fetched 6,924 kB in 42s (162 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Selected version '1.2.15-5' (Raspbian:7.0/oldstable [armhf]) for 'libsdl1.2debian'
The following packages will be DOWNGRADED:
  libsdl1.2debian
0 upgraded, 0 newly installed, 1 downgraded, 0 to remove and 21 not upgraded.
Need to get 203 kB of archives.
After this operation, 12.3 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian/ wheezy/main libsdl1.2debian armhf 1.2.15-5 [203 kB]
Fetched 203 kB in 1s (134 kB/s)
dpkg: warning: downgrading libsdl1.2debian:armhf from 1.2.15-10+rpil to 1.2.15-5
(Reading database ... 33729 files and directories currently installed.)
Preparing to unpack .../libsdl1.2debian_1.2.15-5_armhf.deb ...
Unpacking libsdl1.2debian:armhf (1.2.15-5) over (1.2.15-10+rpil) ...
Setting up libsdl1.2debian:armhf (1.2.15-5) ...
Processing triggers for libc-bin (2.19-18+deb8u1) ...
pi@raspberrypi:~$
```

OK **now** you can continue with pygame



## Extras!

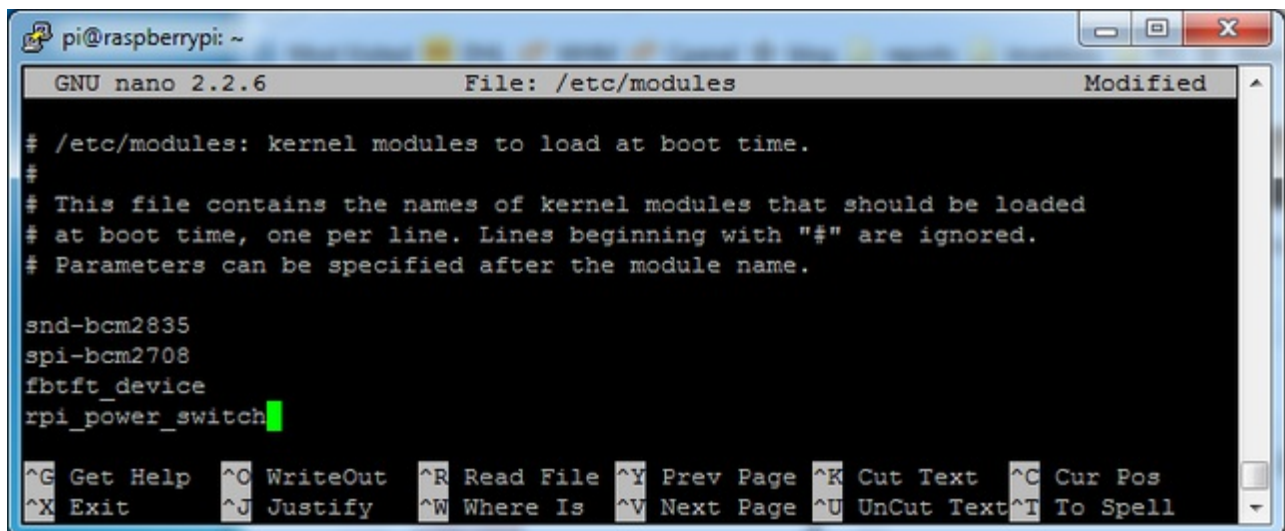
# Tactile switch as power button

Its a good idea to safely turn off your Pi with a good **sudo shutdown -h now** but that often means pulling out a keyboard or connecting to the console. With our kernel we added a cool module that will let you turn any GPIO into a power button. Since there's a couple of tactile switches right there on the front, lets turn one into a power button. Press once to properly turn off the pi, press again to start it up. Isn't that nice?

We'll be using GPIO #23, the left-most button on a PiTFT 2.8", on the 2.4" HAT, #16 is a good choice since its on a tactile button. But, you can use any GPIO you want, really!

You will have to grab a pack of slim tactile switches(<http://adafru.it/1489>) or otherwise solder in a button

Add **rpi\_power\_switch** to **/etc/modules** and save



```

pi@raspberrypi: ~
GNU nano 2.2.6      File: /etc/modules      Modified
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
spi-bcm2708
fbtft_device
rpi_power_switch
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell

```

Now create a new conf file or edit our existing one with

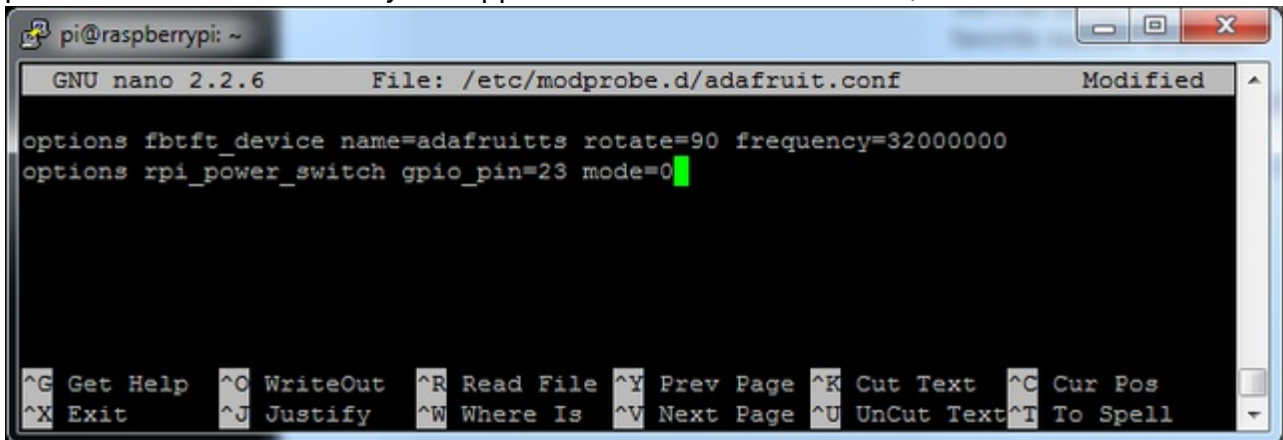
**sudo nano /etc/modprobe.d/adafruit.conf**

and enter in the line

**options rpi\_power\_switch gpio\_pin=23 mode=0**

Of course, change the **gpio\_pin** setting to some other # if you wish. **mode=0** means its a

pushbutton *not* a switch. If you happen to install an on/off switch, use **mode=1**

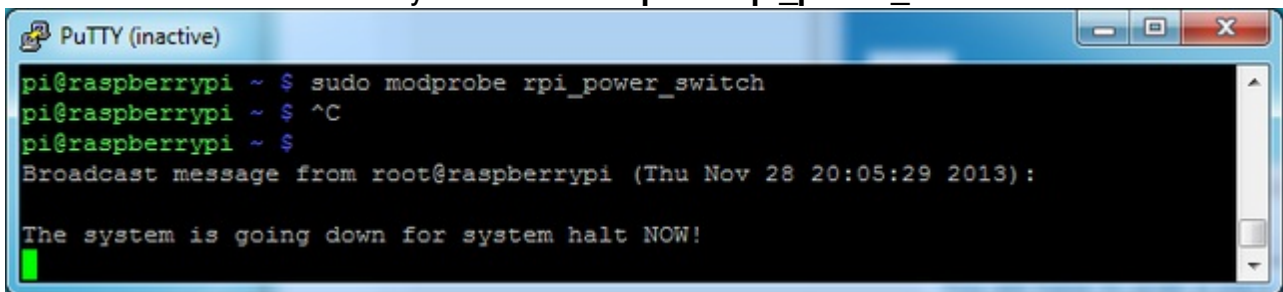


```
pi@raspberrypi: ~
GNU nano 2.2.6      File: /etc/modprobe.d/adafruit.conf      Modified

options fbtft_device name=adafruitts rotate=90 frequency=32000000
options rpi_power_switch gpio_pin=23 mode=0

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

To make it active immediately run **sudo modprobe rpi\_power\_switch**



```
PuTTY (inactive)
pi@raspberrypi ~ $ sudo modprobe rpi_power_switch
pi@raspberrypi ~ $ ^C
pi@raspberrypi ~ $
Broadcast message from root@raspberrypi (Thu Nov 28 20:05:29 2013):

The system is going down for system halt NOW!
```

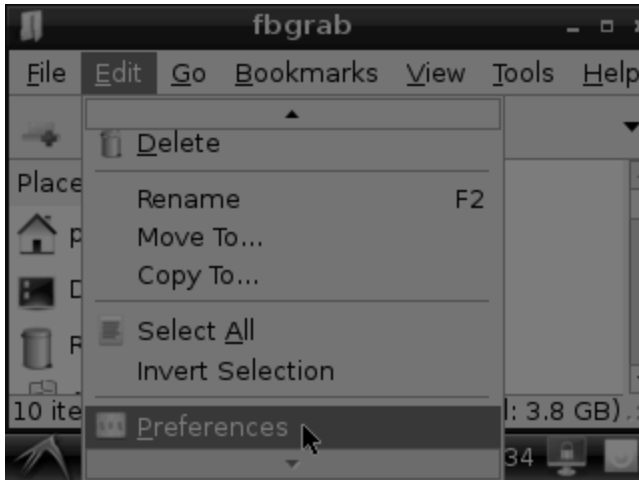
## Making it easier to click icons in X

If you want to double-click on icons to launch something in X you may find it annoying to get it to work right. In LXDE you can simply set it up so that you only need to single click instead of double.

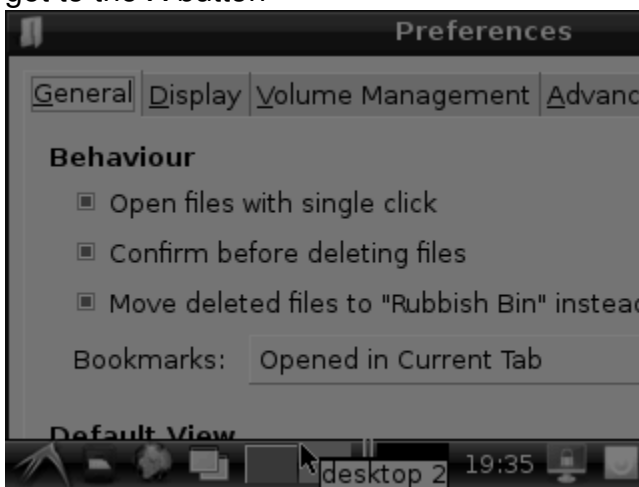
From LXDE launch the file manager (sorry these pix are grayscale, still figuring out how to screenshot the framebuffer!)



Then under the **Edit** menu, select **Preferences**



Then select **Open files with single click** and close the window (you'll need to drag it over to get to the X button)



## Boot to X Windows on PiTFT

To enable booting straight to X windows on the PiTFT follow the steps below. First make sure a display configuration which would conflict is **not** present by executing in a terminal on the Pi:

```
sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
```

Don't worry if the command fails with an error that the file doesn't exist. This failure is normal and should happen on a good PiTFT install. You can ignore it and move on.

Next run the command below to open the nano text editor as root and create the file **/usr/share/X11/xorg.conf.d/99-pitft.conf**:

```
sudo nano /usr/share/X11/xorg.conf.d/99-pitft.conf
```

When the editor loads to a blank file, copy in the text below:

```
Section "Device"
  Identifier "Adafruit PiTFT"
  Driver "fbdev"
  Option "fbdev" "/dev/fb1"
EndSection
```

Then save the file by pressing **Ctrl-O** and then **enter**, and finally exit by pressing **Ctrl-X**.

The step above will create a configuration file which tells X windows that it should use the PiTFT framebuffer (located at /dev/fb1) by default when it runs.

At this point you can use the raspi-config tool to enable booting to desktop just like normal on the Pi. Run the following command:

```
sudo raspi-config
```

Then pick the **Enable Boot to Desktop/Scratch** option and choose if you want to boot to the console, desktop, or scratch environment. After exiting the tool and rebooting you should see the Pi load X windows on the PiTFT after (be patient it can take around 30 seconds to load).

If you want to disable booting to X, just use the raspi-config command again to choose the console boot option.

## Right-click on a touchscreen

Obviously if you have a touchscreen, it cannot tell what finger you are pressing with. This means that all 'clicks' are left clicks. But if you want a right-click, you *can* do it.

Just add the following lines into your InputClass of **/etc/X11/xorg.conf.d/99-calibration.conf** after the calibration section

```
Option "EmulateThirdButton" "1"
Option "EmulateThirdButtonTimeout" "750"
Option "EmulateThirdButtonMoveThreshold" "30"
```

So for example your file will look like:

```
Section "InputClass"
  Identifier "calibration"
  MatchProduct "stmpe-ts"
  Option "Calibration" "3800 120 200 3900"
  Option "SwapAxes" "1"
  Option "EmulateThirdButton" "1"
  Option "EmulateThirdButtonTimeout" "750"
  Option "EmulateThirdButtonMoveThreshold" "30"
EndSection
```

This makes a right mouse click emulated when holding down the stylus for 750 ms.

([Thx adamaddin!](https://adafru.it/fH3) (<https://adafru.it/fH3>))



# Gesture Input

With the PiTFT touchscreen and [xstroke](https://adafru.it/dD0) (<https://adafru.it/dD0>) you can enter text in applications by drawing simple character gestures on the screen! Check out the video below for a short demonstration and overview of gesture input with xstroke:

## Installation

Unfortunately xstroke hasn't been actively maintained for a few years so there isn't a binary package you can directly install. However compiling the tool is straightforward and easy with the steps below. Credit for these installation steps goes to [mwilliams03 at ozzmaker.com](https://adafru.it/dD1) (<https://adafru.it/dD1>).

First install a few dependencies by opening a command window on the Pi and executing:

```
sudo apt-get -y install build-essential libxft-dev libxpm-dev libxtst-dev
```

Now download, compile, and install xstroke by executing:

```
cd ~
wget http://mirror.egtvendt.no/avr32linux.org/twiki/pub/Main/XStroke/xstroke-0.6.tar.gz
tar xfv xstroke-0.6.tar.gz
cd xstroke-0.6
./configure
sed -i '/^X_LIBS = / s/$/ -lXrender -lX11 -lXext -ldl/' Makefile
make
sudo make install
```

If the commands above execute successfully xstroke should be installed. If you see an error message, carefully check the dependencies above were installed and try again.

Once xstroke is installed you will want to add a couple menu shortcuts to start and stop xstroke. Execute the following commands to install these shortcuts:

```
wget https://github.com/adafruit/PiTFT_Extras/raw/master/xstroke.desktop
wget https://github.com/adafruit/PiTFT_Extras/raw/master/xstrokekill.desktop
sudo cp xstroke*.desktop /usr/share/applications/
```

## Usage

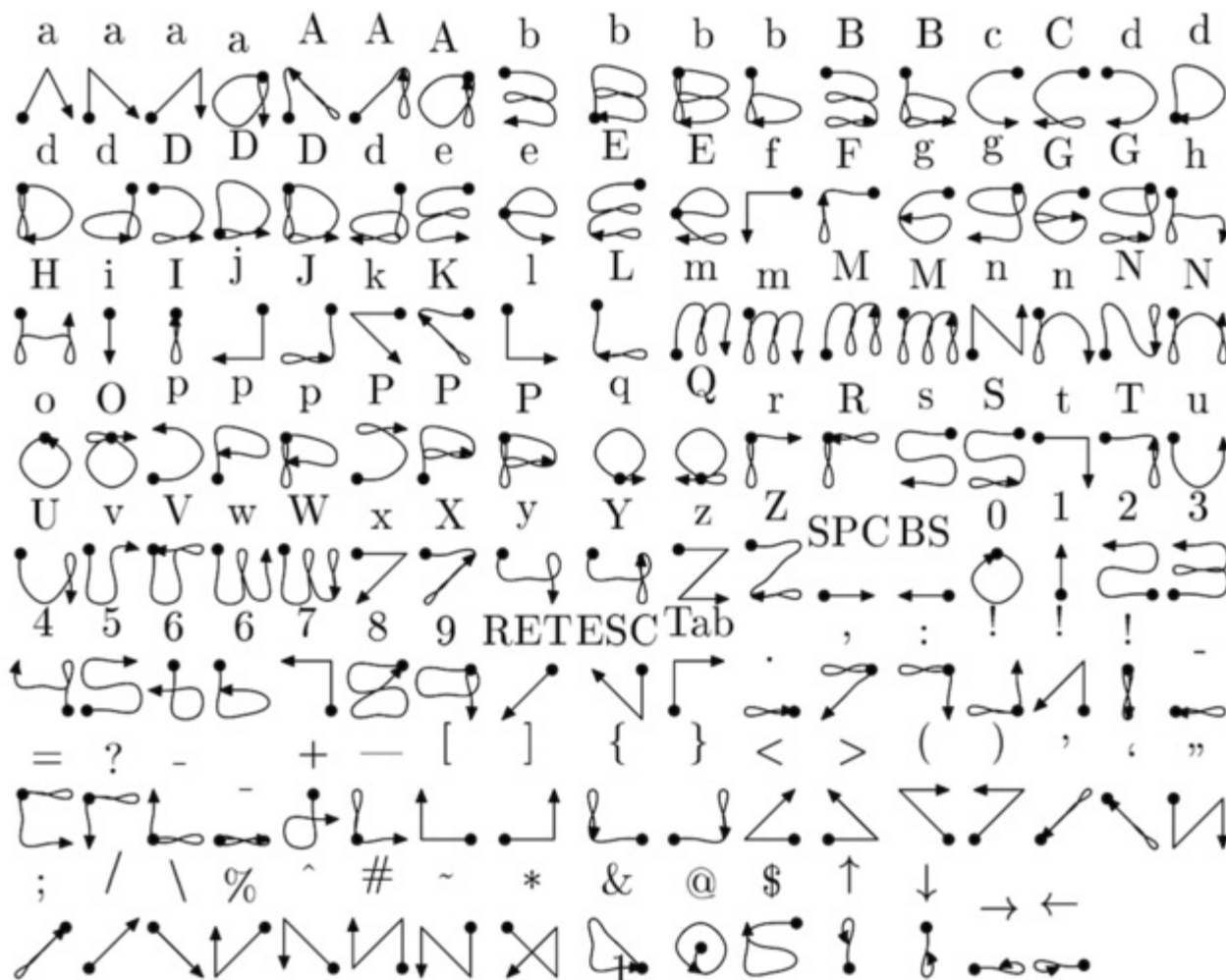
To use xstroke I highly recommend using a plastic stylus instead of your finger. Also [calibrate the touchscreen for X-Windows](https://adafru.it/dD2) (<https://adafru.it/dD2>) so you have the best control over the cursor possible.

Don't use a ballpoint pen or sharp metal stylus as it could scratch or damage the touchscreen! Start X-Windows on the PiTFT and open the LXDE menu by clicking the icon in the lower left corner. Scroll up to the **Accessories** menu at the top and notice the new **XStroke** and **XStroke Kill** commands.

Click the **XStroke** menu option to start xstroke. You should see a small pencil icon appear on the bottom right side of the screen. The pencil icon means xstroke is running, however by default it's not yet looking for gesture input.

Open an application that takes text input, such as LXTerminal. To enable gesture input click the xstroke pencil icon. You should see the pencil turn green and the text 'abc' written over top of the icon. You might need to click the icon a few times to get the click to register in the right spot.

When xstroke is looking for gesture input you can drag the mouse cursor in a gesture anywhere on the screen to send specific key strokes. Here's a picture of the possible gestures you can send:



(credit to Carl Worth for the image above)

To draw a gesture from the above image, press anywhere on the screen, start from the circle in the gesture, and follow the gesture pattern towards the arrow. As you draw a gesture you should see a blue line displayed that shows what you've drawn. Lift up the stylus when you get to the end of the gesture at the arrow. If xstroke recognizes the gesture it will send the appropriate key press to the active window. Try drawing a few characters from the image above to get the hang of writing gestures.

A few very useful gestures are backspace (which deletes a character), return/enter, and space. To draw a backspace gesture just draw a line going from the right side of the screen to the left side. The gesture for return/enter is a diagonal line from the top right to bottom left. Finally a space is a straight line from the left to the right.

Note that when xstroke is looking for gestures you might not be able to click or control the cursor as you normally would expect. To stop xstroke's gesture recognition carefully press the xstroke pencil icon again until the 'abc' text disappears. I've found this process can be a little

finicky as the icon is very small and any movement will be interpreted as a gesture. Use a light touch and try a few times to click the icon.

If you get stuck completely and can't disable xstroke by clicking the icon, connect to the Raspberry Pi in a terminal/SSH connection and run 'killall xstroke' (without quotes) to force xstroke to quit. The normal way to stop xstroke is to navigate to the **Accessories** -> **XStroke Kill** command, but you might not be able to do that if xstroke is listening for gesture input.

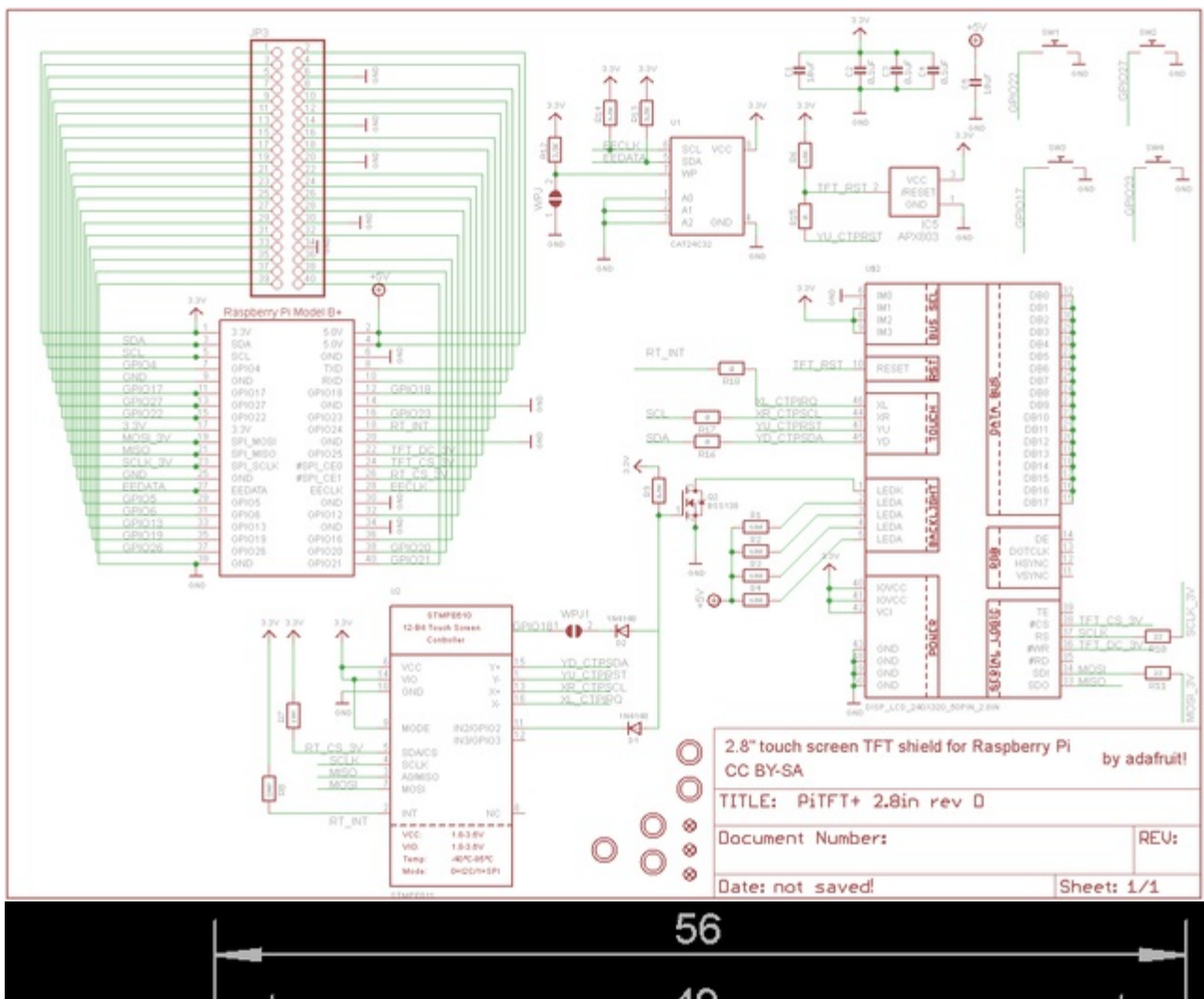
Have fun using xstroke to control your Pi by writing gestures on the PiTFT screen!

# Downloads

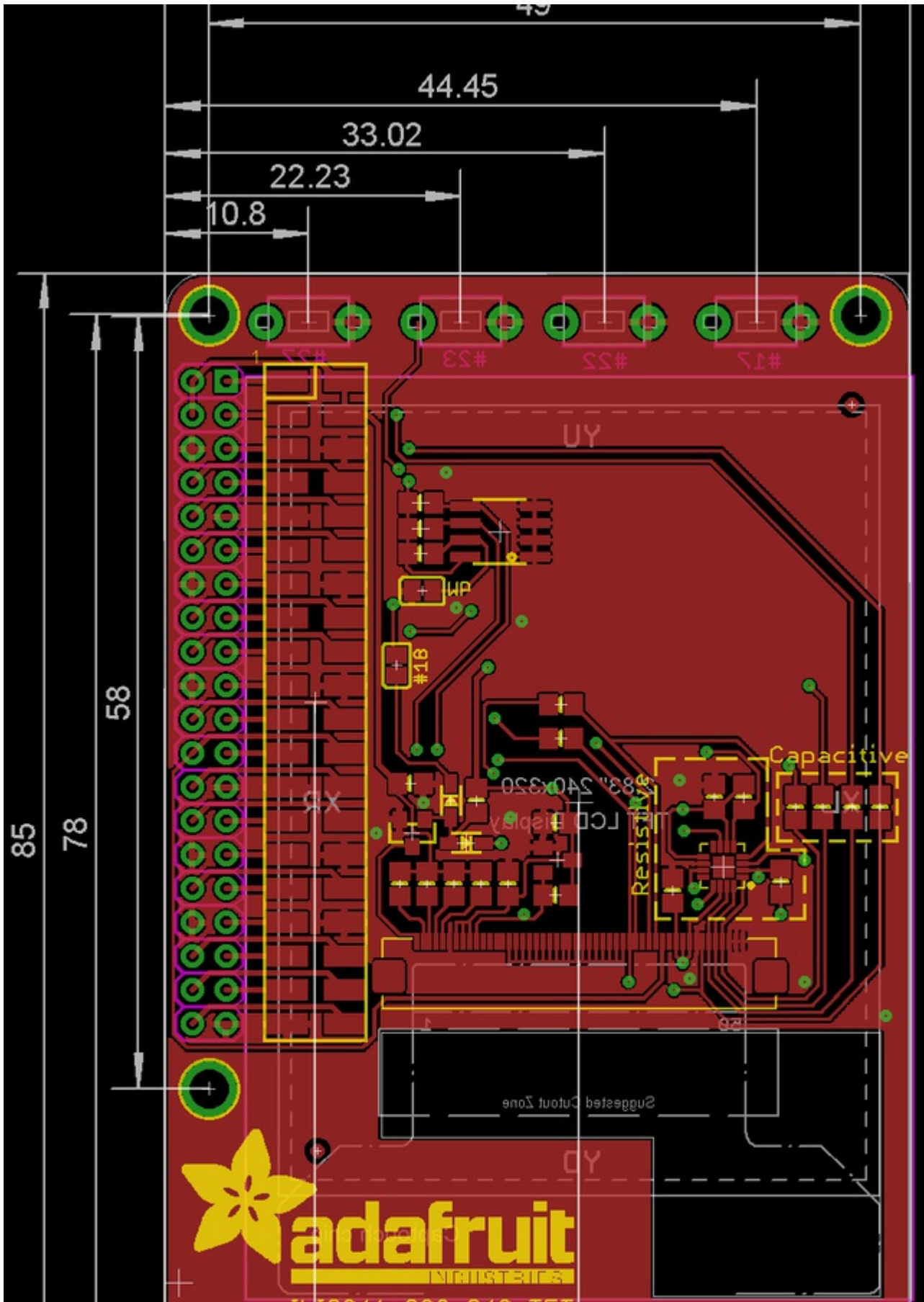
- The latest kernel fork that adds all the TFT, touchscreen, and other addons is here on [github](https://adafru.it/dcA) (<https://adafru.it/dcA>)
- Datasheet for the 'raw' 2.8" TFT display (<https://adafru.it/sEt>)
- Original 2.8" PiTFT EagleCAD PCB Files on GitHub (<https://adafru.it/oYB>)
- PiTFT Plus 2.8" EagleCAD PCB Files on GitHub (<https://adafru.it/oYC>)
- PiTFT Plus 3.2" EagleCAD PCB Files on GitHub (<https://adafru.it/rDv>)
- Fritzing Files in the Adafruit Fritzing Library (<https://adafru.it/aP3>)

## 2.8" PiTFT Plus Schematic & Layout

For the Pi B+ & Pi 2 version (2x20 header)

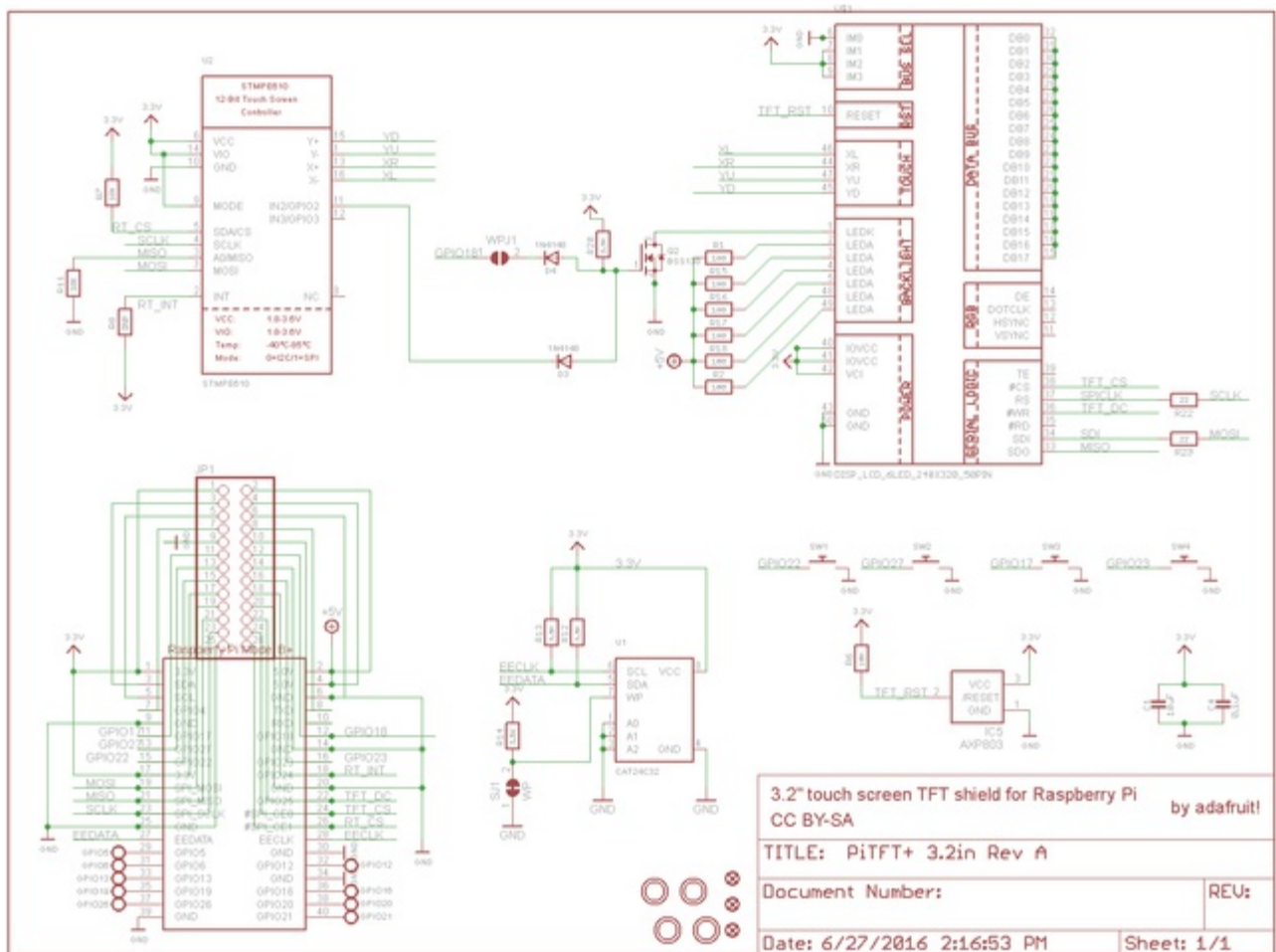








## PiTFT 3.2" Plus Schematic



## Original PiTFT 2.8" Schematic & Layout

For the Original Pi 1 version (2x13 header)

