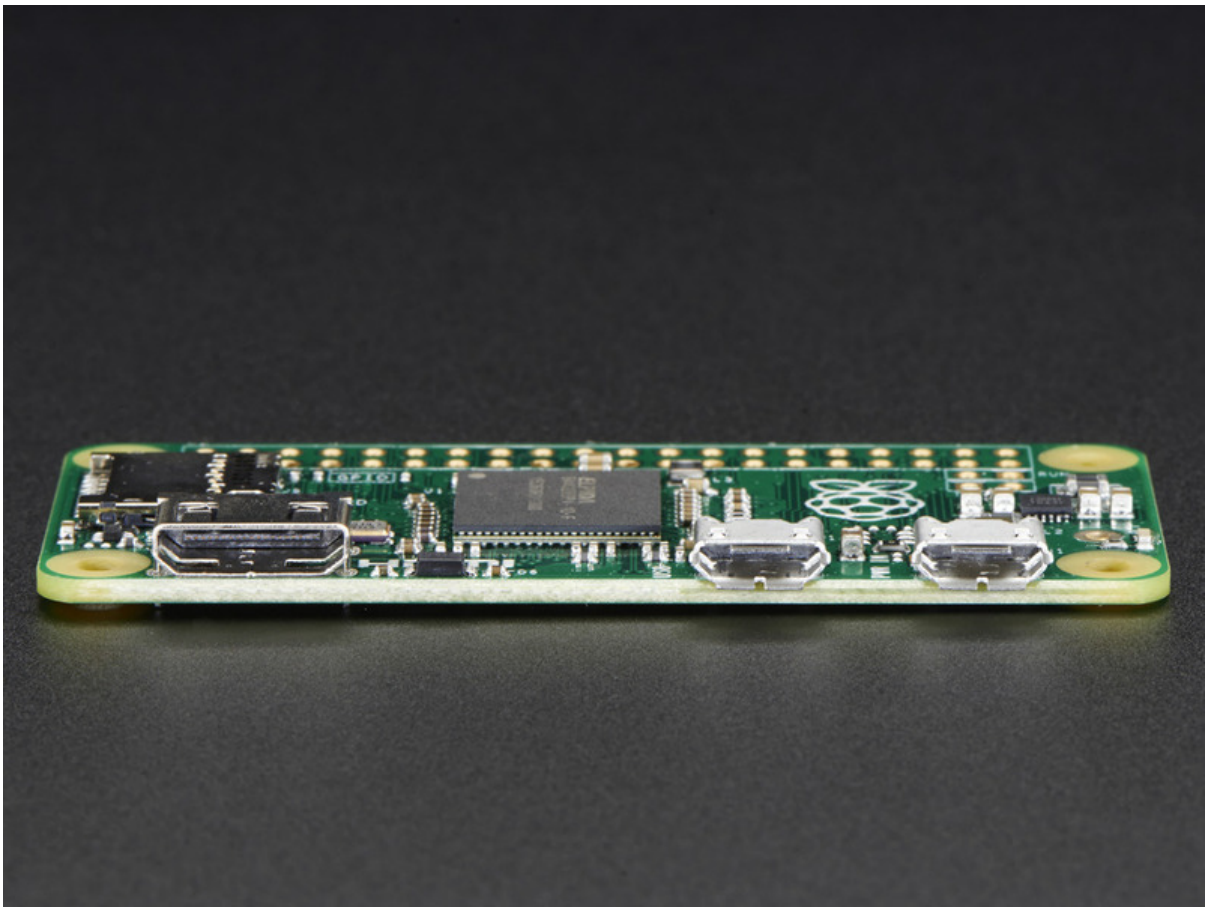
Turning your Raspberry Pi Zero into a USB Gadget

Created by lady ada

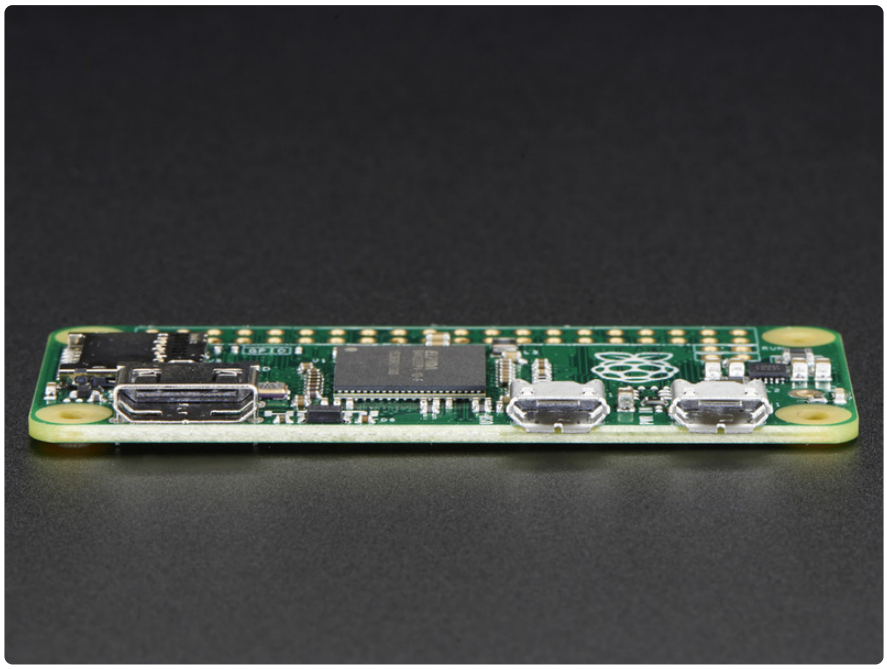


https://learn.adafruit.com/turning-your-raspberry-pi-zero-into-a-usb-gadget

Last updated on 2024-03-08 02:21:10 PM EST

Table of Contents

Overview



When the Pi Zero came out, one of the downsides (!) of the low-cost design was swapping the 'standard' USB A-port for a micro-B port. Now you have to use an 'OTG' cable instead of just plugging in a device.

There was also the matter of, if you didn't have anything connected to USB, and powered up the Pi Zero with an old Raspbian image, you'd get a strange warning (https://adafru.it/khe)

```
WARN::dwc_otg_handle_mode_mismatch_intr:68: Mode Mismatch Interrupt: currently in Device mode
```

Basically, the Pi sorta-trying to become a **usb device** rather than a **usb host**

Some awesome people on github (https://adafru.it/khf) sorted out that if you used the DWC2 USB driver, and patched a few files, you could get the Pi to act like a USB device (in linux-land this is called the **USB Gadget** system)

Thx for the tips from Andrew, as of May 2016, Raspbian Jessie does not require a new kernel (https://adafru.it/q1d)

This tutorial is basically just a writeup of how you can follow along and turn your Pi zero into a **USB Serial** device or **Ethernet** device. That's two whole ways of being able to connect to your Pi zero just by plugging in a micro B cable! You don't even need to power your Pi seperately, as power is provided from your computer.

As of May 2016, Raspbian Jessie has built in kernel support - this tutorial is way easier!

Yeah the gadget system can do a lot more, but these are the two modules we've tested so far. The compiled kernel package has just about every USB gadget compiled in as a module if you'd like to try others

Before You Begin

This tutorial isn't terribly difficult but you should have some raspberry Pi experience. In particular you will want to do the following **before anything else**

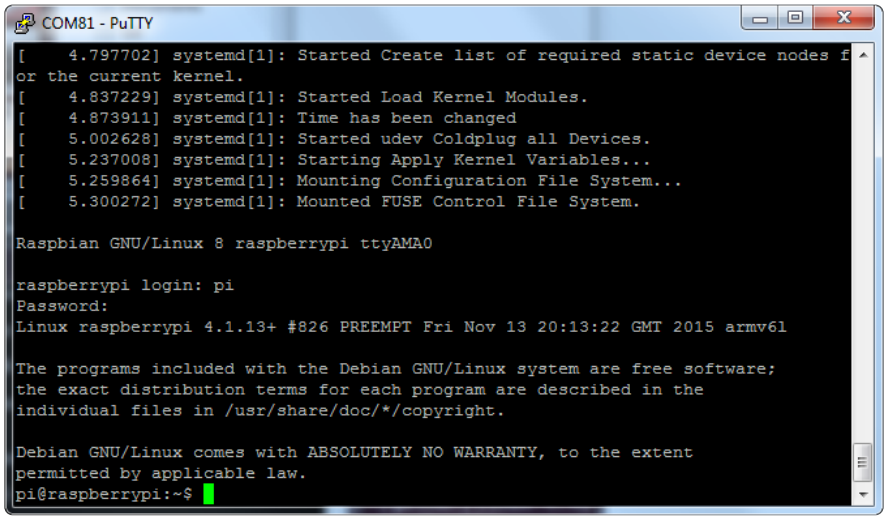
- Burn a copy of Rasbian Jessie Lite (or just plain Jessie) to a 4G or 8G SD card. (https://adafru.it/dDL)
- Micro USB cable

For Gadget serial you'll also want

- Solder in a 2x20 male header (http://adafru.it/2822) or somehow be able to connect a console cable to your Pi Zero
- Have a USB console cable and be able to log into your Pi over serial from a desktop computer (https://adafru.it/kgF)

While you don't need a console cable, it's a lot easier to copy & paste the commands into a terminal than to type into a keyboard + monitor.

Basically, get your Pi zero to a point you can log in. Power it from the Power USB port, leave the Data USB port 'empty'



OK **now** you can continue!

Serial Gadget

We'll start with Serial Gadget, which is the 'simplest' of the USB gadgets. This one basically makes it so when you plug in the Pi Zero to your computer, it will pop up as a **Serial (COM) Port** - the nice thing about this technique is you can use the pi with any computer and operating system and it doesnt require special drivers or configuration.

Thx for the tips from Andrew, as of May 2016, Raspbian Jessie does not require a new kernel (https://adafru.it/q1d)

Step 0. Download and install latest Jessie

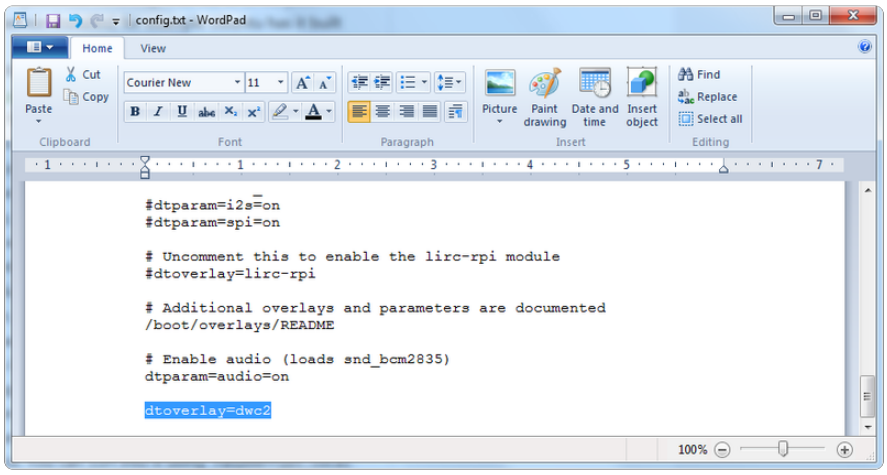
We're using Jessie Lite but plain Jessie Raspbian should work too! You need May 2016 or later (tested with 2016-05-27)

This tutorial has the details (https://adafru.it/dDL)

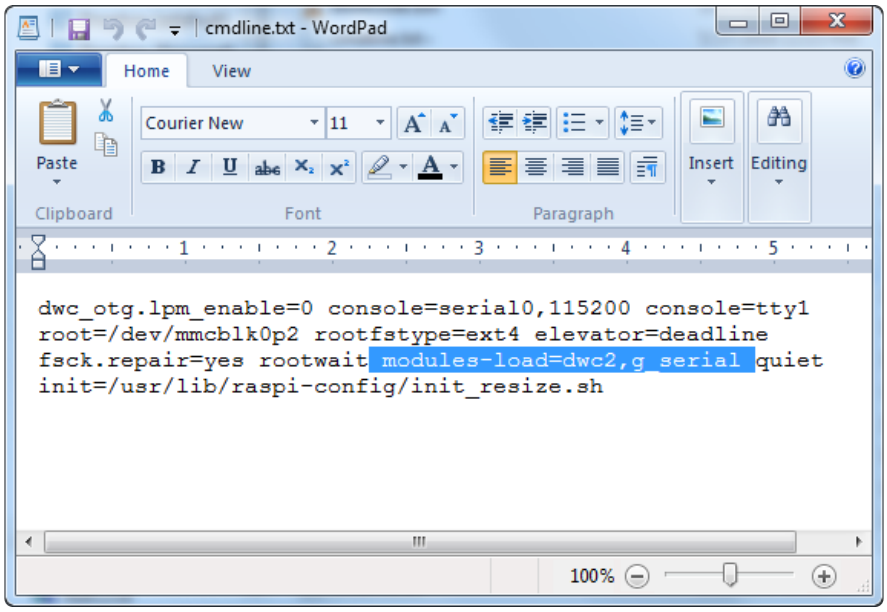
Step 1. Edit config.txt & cmdline.txt

After burning the SD card, do not eject it from your computer! Use a text editor to open up the **config.txt** file that is in the SD card post-burn.

Go to the bottom and add `dtoverlay=dwc2` as the last line:



Save the config.txt file as plain text and then open up cmdline.txt After **rootwait** (the last word on the first line) add a space and then `modules-load=dwc2,g_serial`

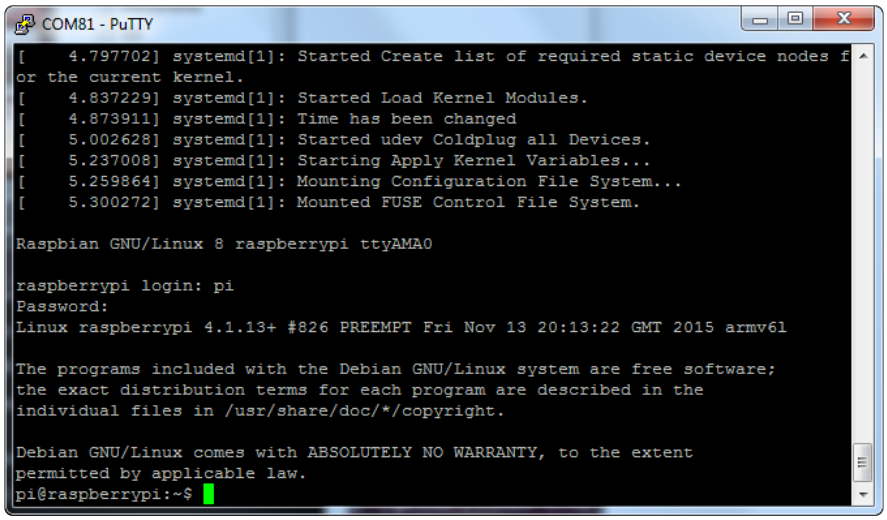


At the time of writing, this is the full cmdline.txt contents (in case you need to start over). Note it is one very long line

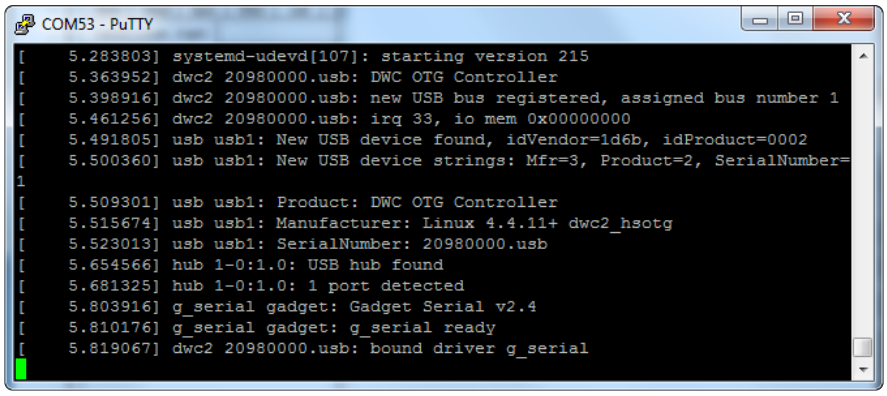
```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait modules-
load=dwc2,g_serial quiet init=/usr/lib/raspi-config/init_resize.sh
```

Log into your Pi Zero

Insert the SD into your Pi Zero, connect the console cable, power the Pi & log into via the USB console.



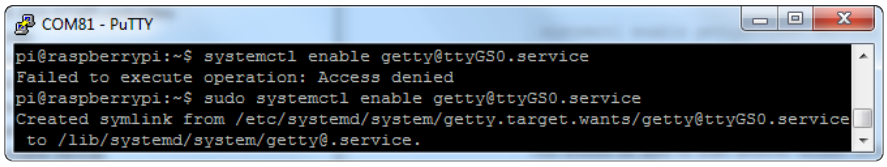
While booting, or later when runing **sudo dmesg** you can see that it bound driver **g_serial**



Set up logging in on Pi Zero via Serial Gadget

OK just cuz you have a Serial port doesn't mean you can log in with it yet. The Pi knows it has a Serial port but you have to tie it to a console. You can do that very easily with:

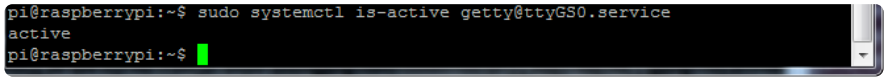
- **sudo systemctl enable** getty@ttyGS0.service



(don't forget the sudo like i did at first!)

You can then verify its running with

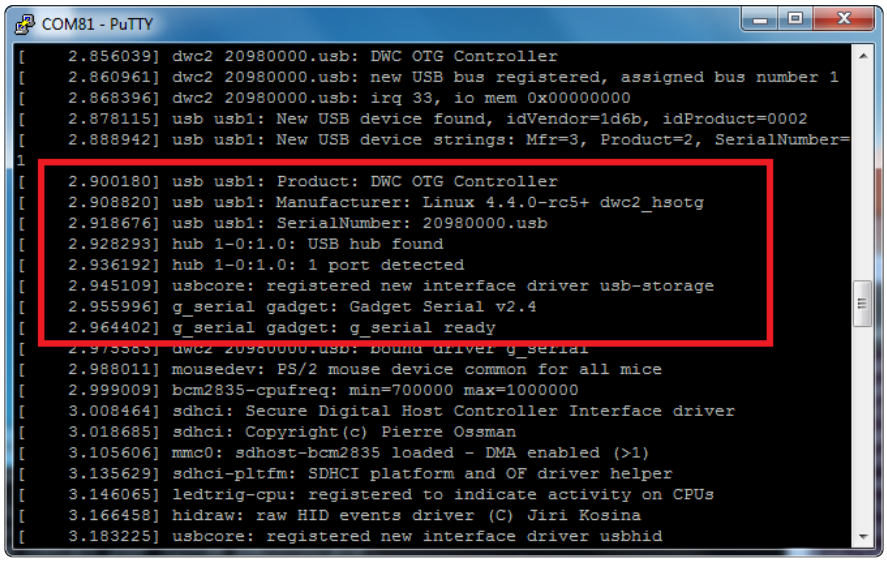
- **sudo systemctl is-active getty@ttyGS0.service**



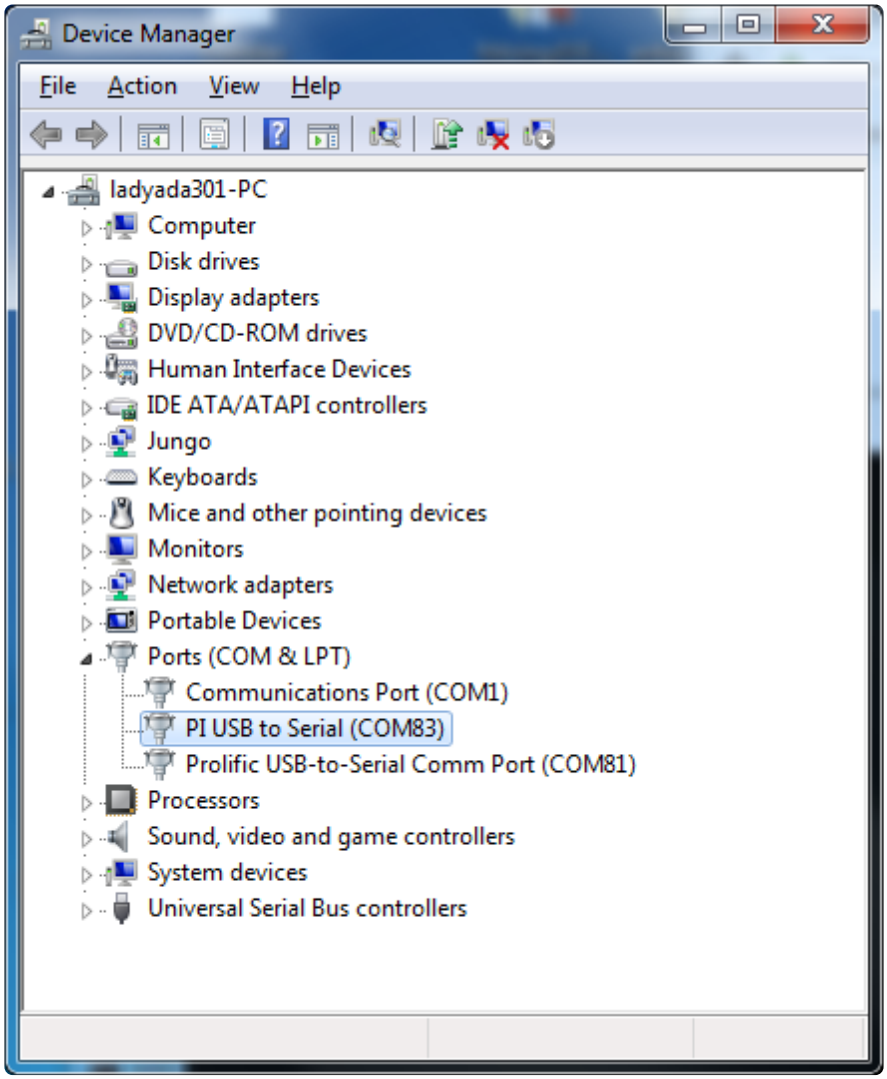
Thats...pretty much it. run **sudo reboot** to start up your Pi Zero. Plug in a USB Micro cable from your computer to the Pi Zero.

Don't forget to plug in the USB cable from your computer to the "USB" connector port on the Pi Zero, not the PWR connector.

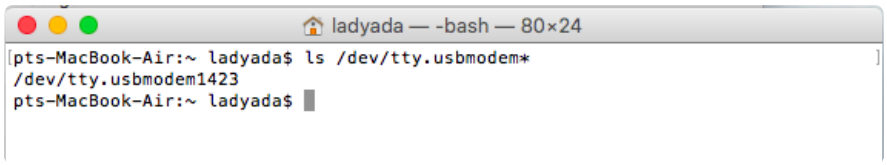
While the Zero is rebooting you can see that it loads the **g_cdc** module which provides "CDC USB Serial support" (CDC stands for 'communications device class' (https://adafru.it/kha))



On your computer you'll see a new Serial port is created. Check the Windows device driver:

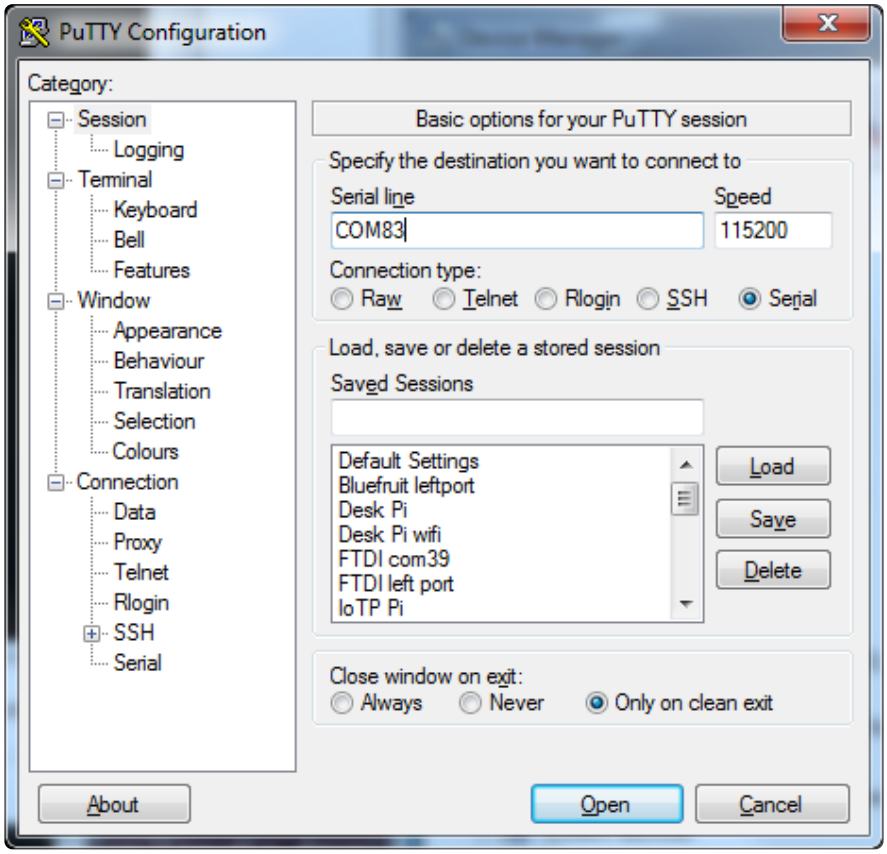


On mac, it will be a new device called **/dev/tty.usbmodemNNNN** where NNNN can be any number



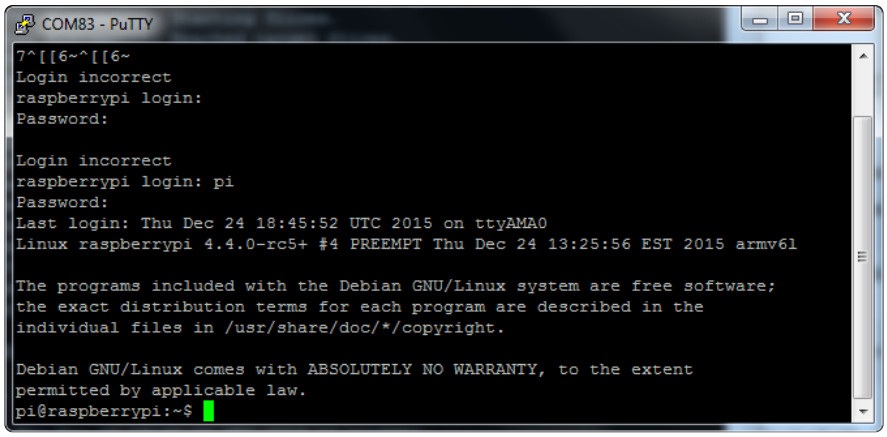
Log into your Pi using Serial Port Software

OK now that your Pi is rebooted and you get that USB serial device again, you can connect to it at **115200** baud (8N1 8-bit No-parity 1-stop if you need to set that)



you can disconnect the console cable, so you dont mix up the USB console cable and the direct-console connection (since they both have COM/Serial ports)

You can also remove the power cable to the 'power USB' port, since the desktop computer will be powering the Pi thru the USB gadget port.



You may have to hit return a few times to get it to come up with the login prompt. But that's it! You're now connected to your Pi Zero directly

---

Ethernet Gadget

The Ethernet Gadget is a little more difficult to set up, but is a lot more powerful because you can tunnel networking, VNC, ssh and scp files, etc. Basically you get the ability to log in to the console as well as anything else you could want to do over a network connection

Note that even though it's called "Ethernet Gadget" you do not use an Ethernet cable! The only cable is the USB micro-B cable from your computer to your Pi Zero. The Pi 'appears' like an Ethernet device.

You can even share your desktop computer's network setup so your Pi can access the internet through your computer via the USB cable! Cool huh?

Thx for the tips from Andrew, as of May 2016, Raspbian Jessie does not require a new kernel & has raspberrypi.local setup by default so it's a lot easier (https://adafru.it/q1d)

Step 0. Download and install latest Jessie

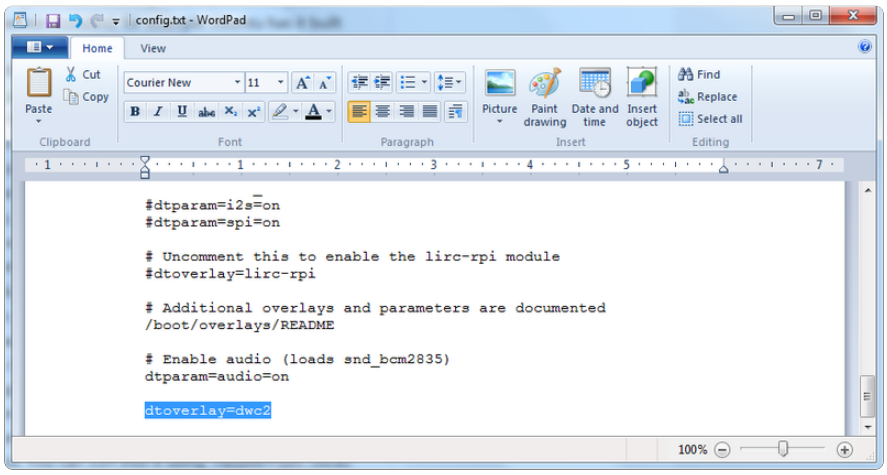
We're using Jessie Lite but plain Jessie Raspbian should work too! We're using Jessie Lite but plain Jessie Raspbian should work too! You need May 2016 or later (tested with 2016-05-27)

This tutorial has the details (https://adafru.it/dDL)

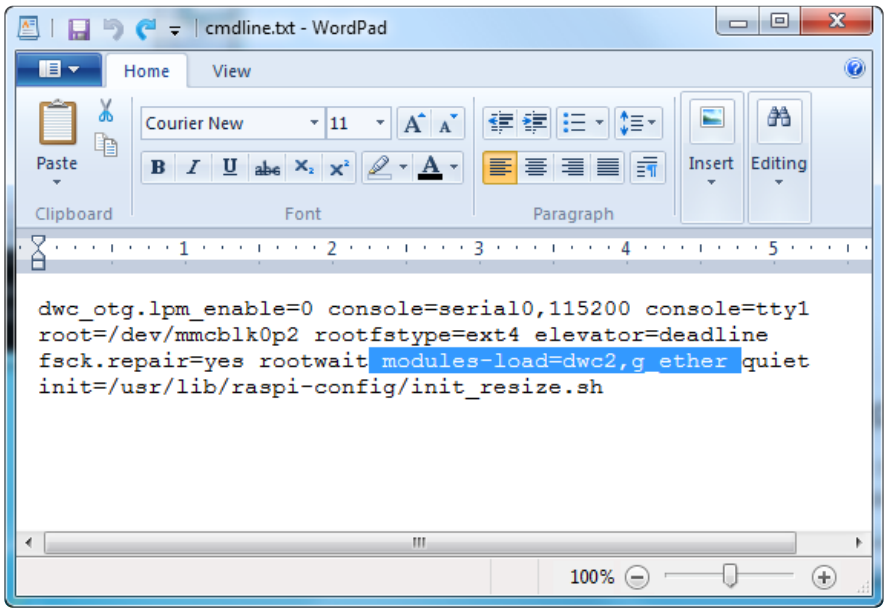
Step 1. Edit config.txt & cmdline.txt

After burning the SD card, do not eject it from your computer! Use a text editor to open up the **config.txt** file that is in the SD card post-burn.

Go to the bottom and add `dtoverlay=dwc2` as the last line:



Save the config.txt file as plain text and then open up cmdline.txt After **rootwait** (the last word on the first line) add a space and then `modules-load=dwc2,g_ether`

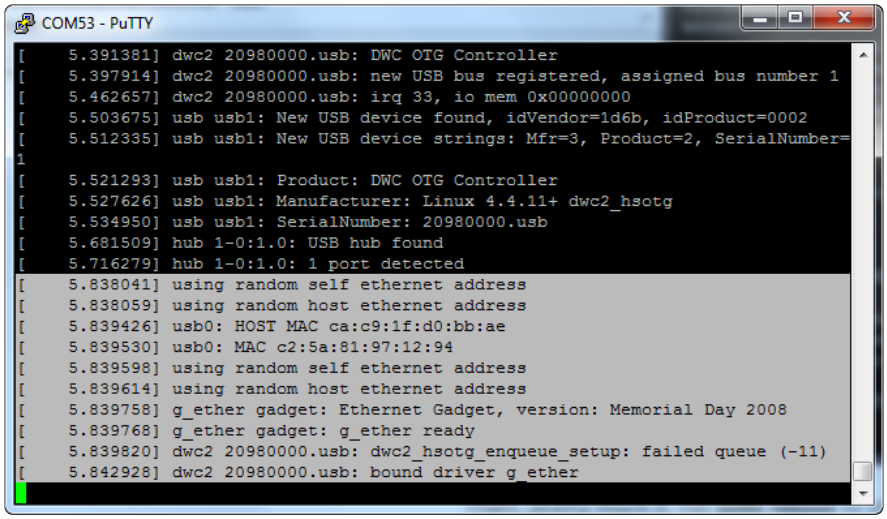


Boot Your Pi with USB

Plug in a MicroUSB cable from your Pi Zero's USB port to your computer

Don't forget to plug in the USB cable from your computer to the "USB" connector port on the Pi Zero, not the PWR connector.

If you have a console cable you can watch the Zero's console to see it enable the **g_ether** device:

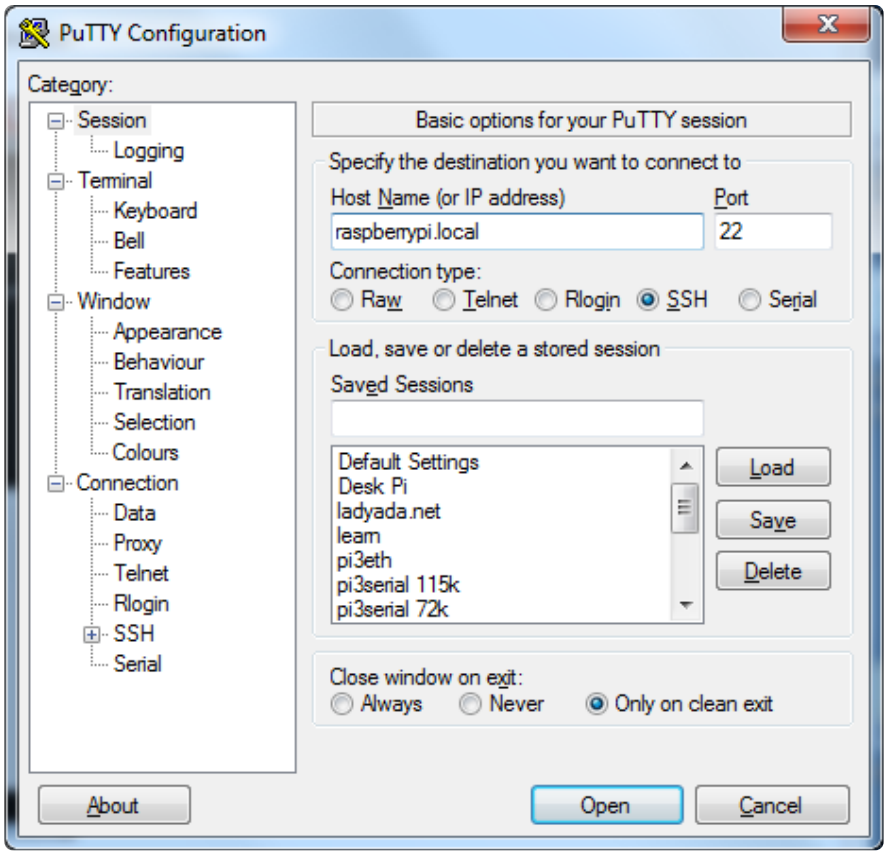


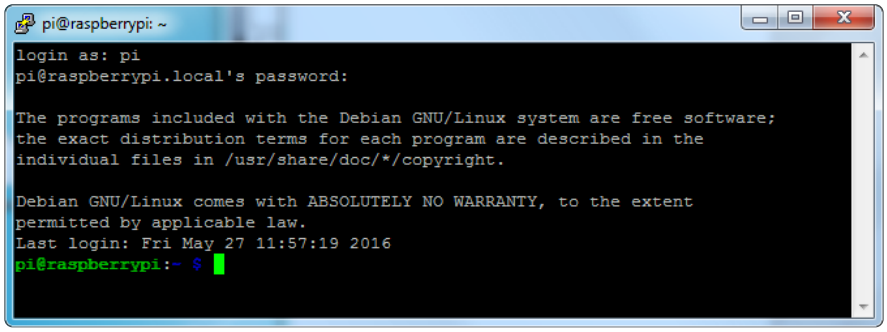
SSH!

If you enable SSH on your Pi, you can then also SSH in to **raspberrypi.local**

Start by enabling SSH (https://adafru.it/vbC)

If you are using a Mac or Linux chances are you have Bonjour already installed. On Windows, you may need to add Bonjour support so it knows what to do with .local names (https://adafru.it/q1e)



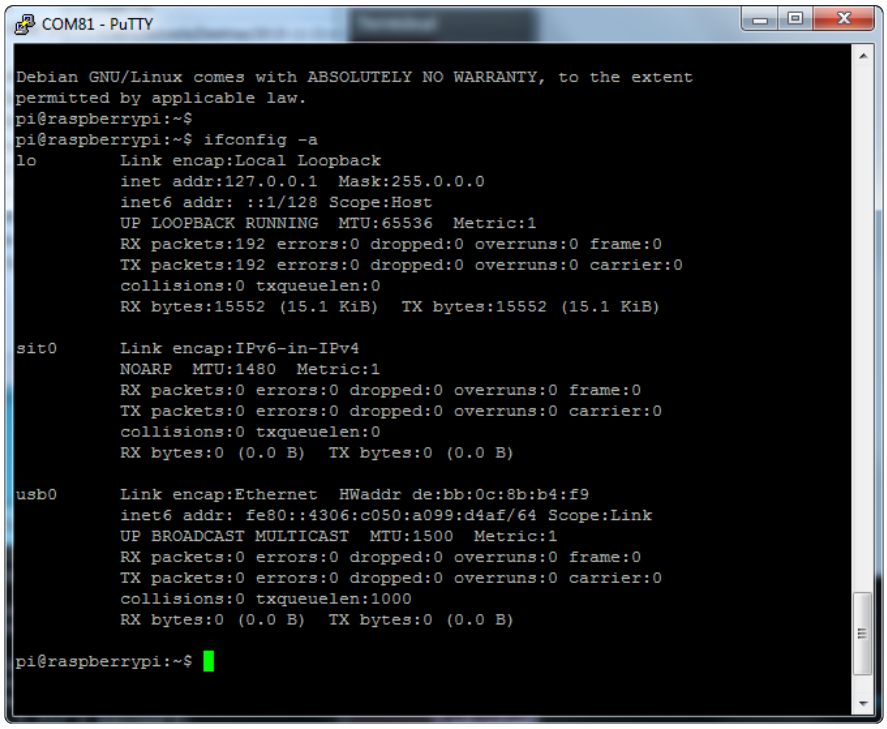


Advanced Networking (Fixed IP)

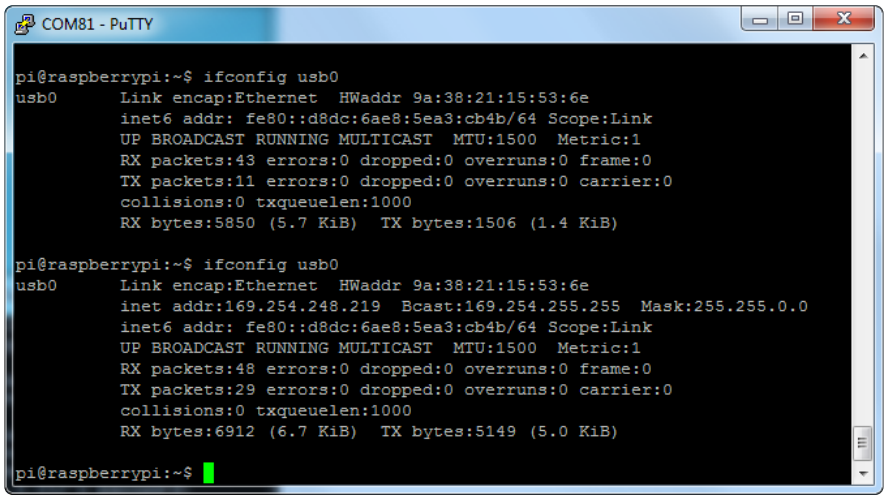
If you need to manage fixed IP addresses for some reason - here's some useful techniques for managing your Pi's Gadget Ethernet device. Otherwise, you can always just keep using **raspberrypi.local**

You can now log in and check that you have a new network device called **usb0**

- **sudo ifconfig -a**



Try plugging the Pi Zero into your computer now. For example, on a Mac, we plugged it in



As you can see above, between the first ifconfig and second, the network came up with an address. The problem this is a arbitrary (Bonjour/Zero Conf assigned) address, and we dont want to have to guess it.

We can configure this device to have a fixed address (this makes it easier to find on a network!)

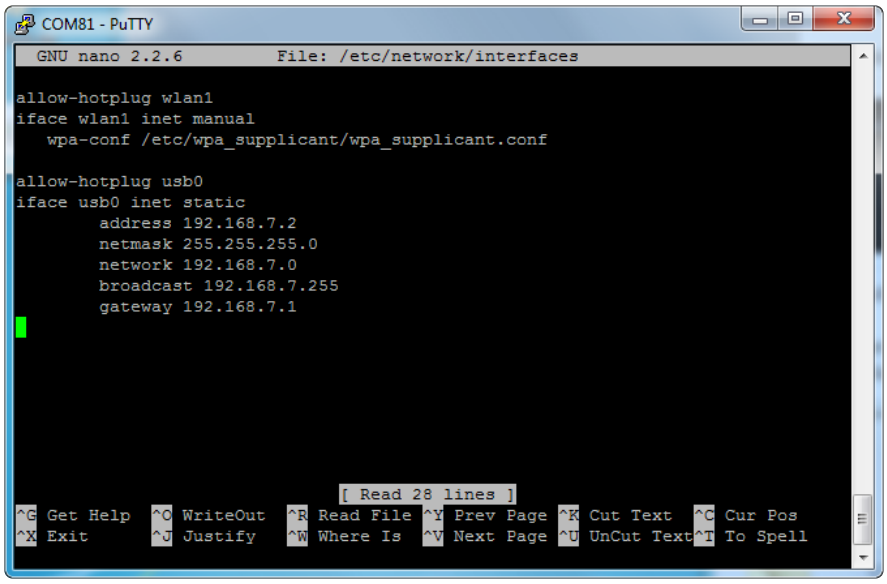
- **sudo nano /etc/network/interfaces**

and add at the end

```
allow-hotplug usb0
iface usb0 inet static
        address 192.168.7.2
        netmask 255.255.255.0
        network 192.168.7.0
        broadcast 192.168.7.255
        gateway 192.168.7.1
```

This will give the **Raspberry Pi the IP Address 192.168.7.2**

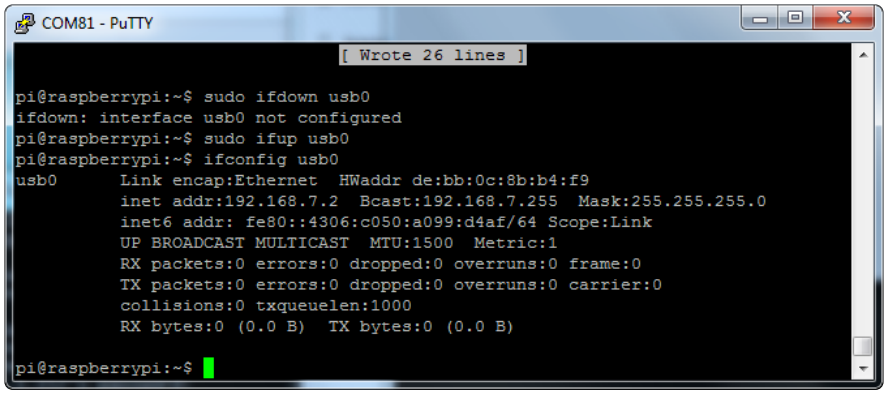
you can change this to a different address but unless you're sure that 192.168.7.* is unavailable, keep it as above for now.



Save the file and run

- **sudo ifdown usb0** (this may fail, its fine)
- **sudo ifup usb0**
- **ifconfig usb0**

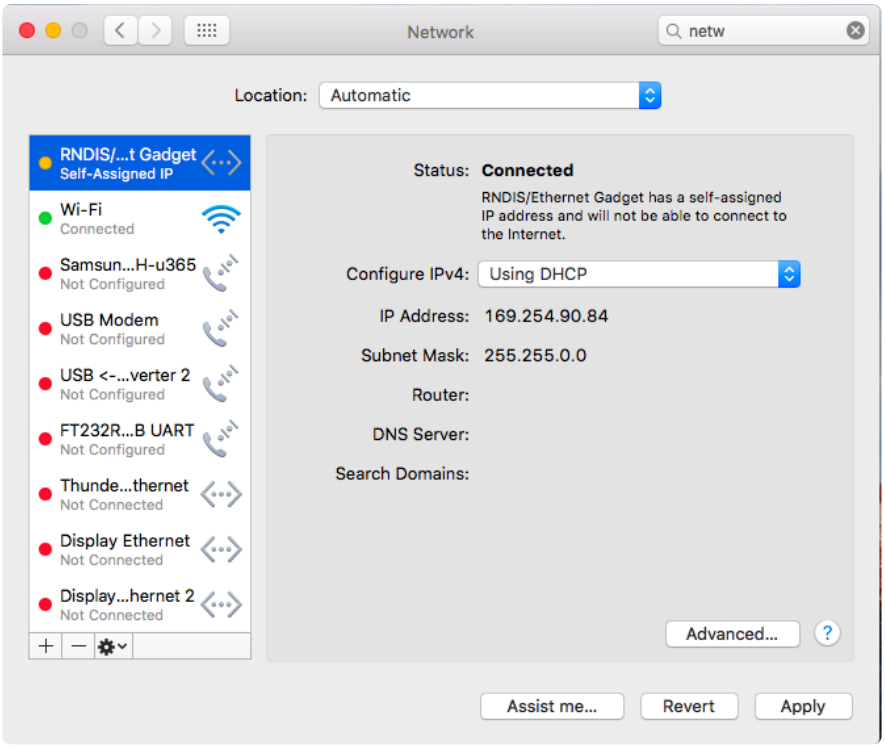
to verify it now has the 192.168.7.2 address



Now on your **computer** you'll need to set it up as well.

If you are using a Mac as the Host Computer

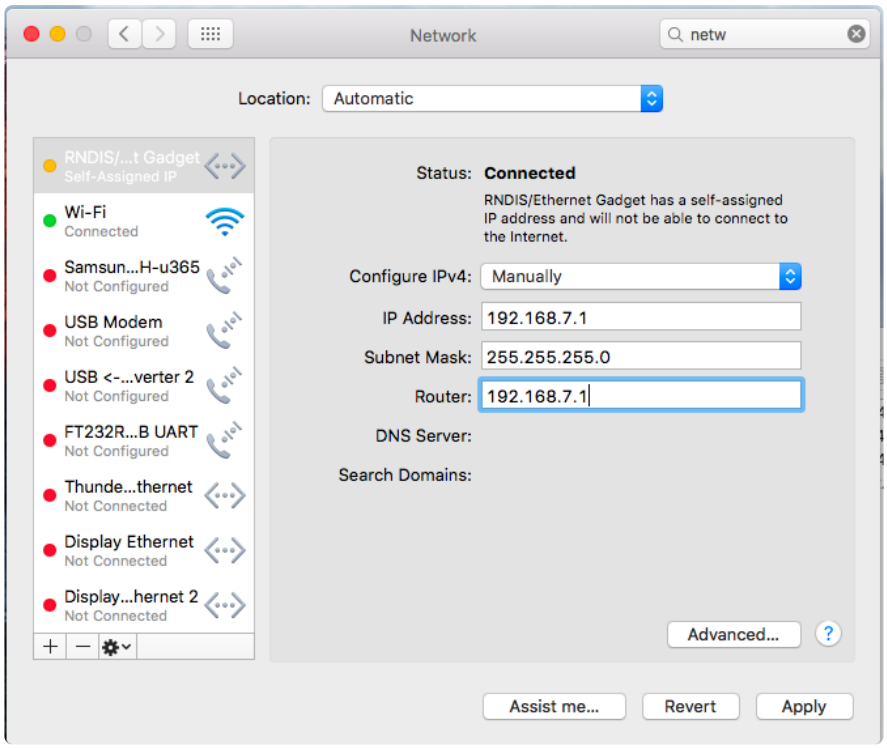
On a Mac OS X machine, open up the **System Preferences -> Network** box.



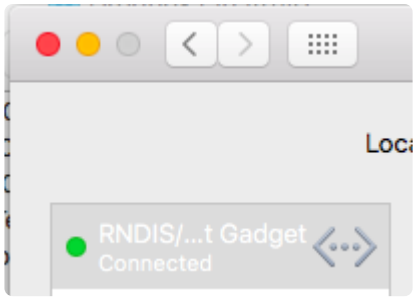
You'll see the device show up as an RNDIS/Ethernet Gadget. it'll probably be set up for DHCP by default so change it to **Configure IP4 Manually**

- For the IP address pick **192.168.7.1** (note that this is not the same as the Pi Zero's address!)
- For the subnet mask, use **255.255.255.0** (same as Pi)
- For the router/gateway use **192.168.7.1** (same as Pi)

If you didnt use our suggested netconfig above on the Pi, you may have to adjust this one to match



Click **Apply** when done, and wait a minute or so you will get a green dot:



If you're still having issues, a reader reported some Mac's need a special option on the g_ether device. While logged into your Pi with a console cable, run `sudo nano/etc/modprobe.d/g_ether.conf`
and add: `options g_ether use_eem=0`

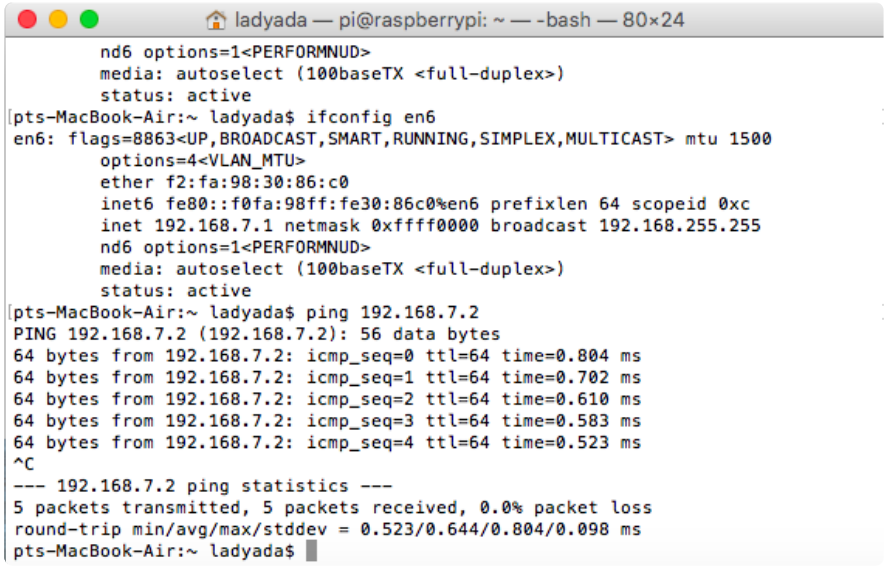
on it's own line, at the end.

After a reboot or manual load of the module, the the RNDIS/CNC gadget will turn yellow then green after assigning an IP.

You can use a terminal on the computer to check the IP address was set, your device will be called **enX** where X is some number, use **ifconfig -a** to see a list of all devices, chances are the Pi is the last one.

Once you can see that the IP address is set, try pinging the pi with

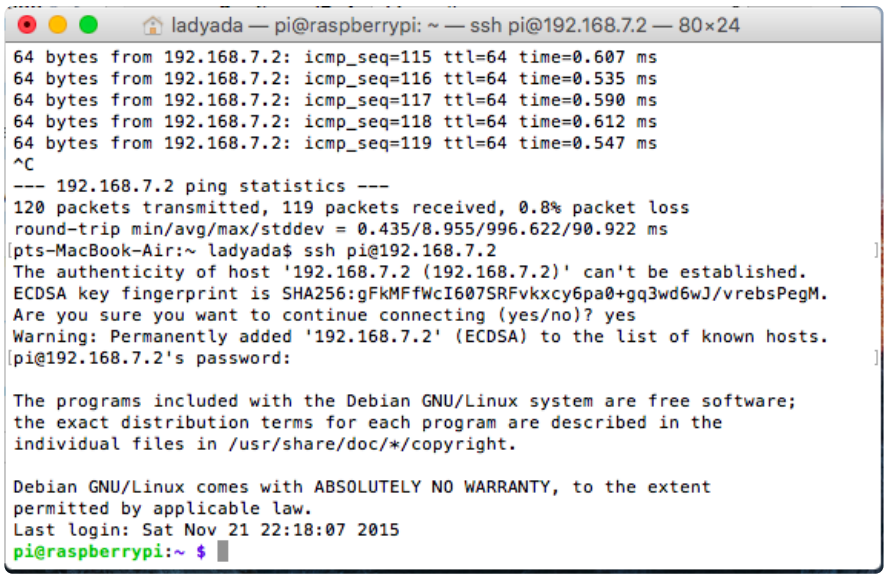
- **ping 192.168.7.2**



To be honest, I rebooted the Pi after setting up the network config file, so if it doesnt work at first, try that.

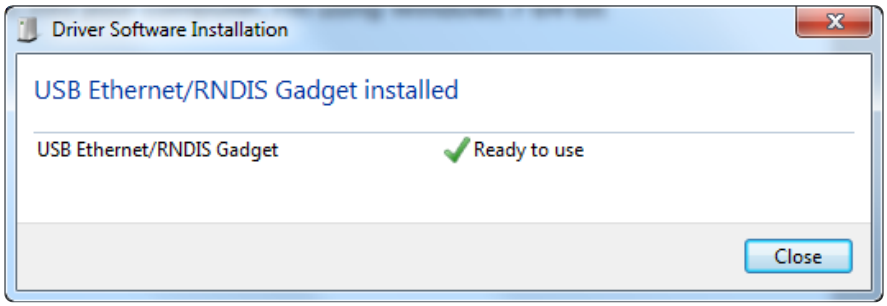
Next up you can ssh into your pi from your Mac!

- **ssh pi@192.168.7.2**



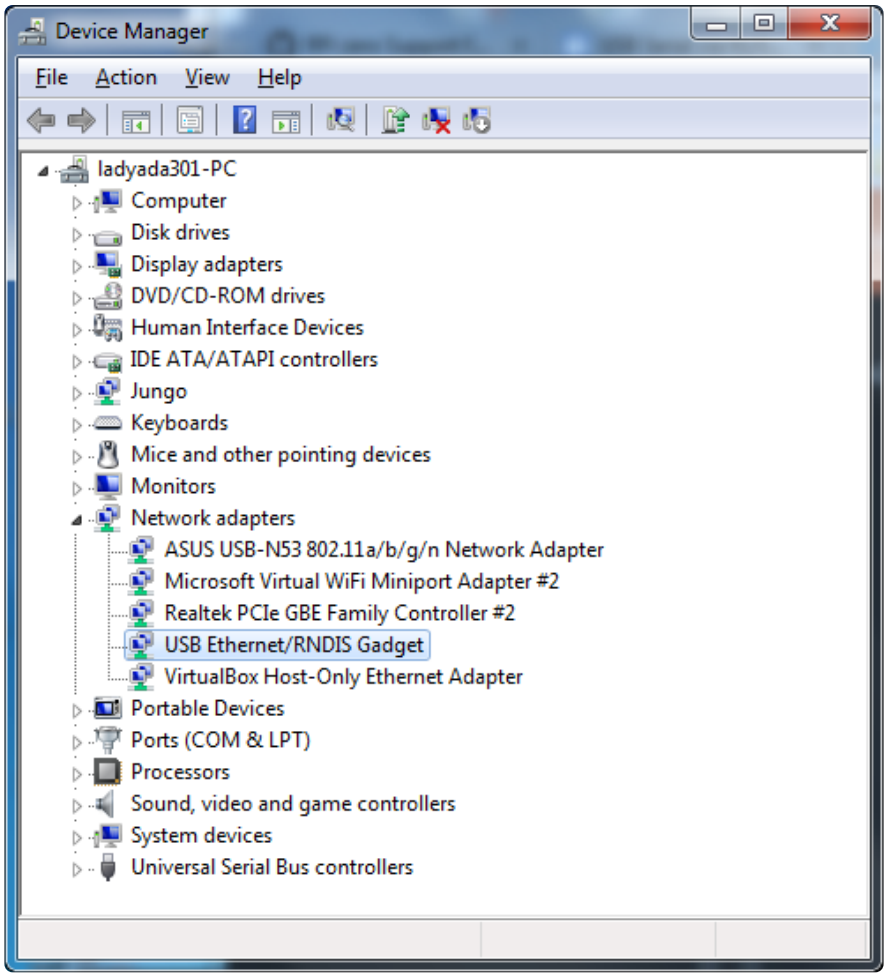
If you are using Windows as the Host Machine

Plug in the Pi Zero into your computer, I'm using Windows 7 64-bit. It will automatically download and install the RNDIS Ethernet drivers

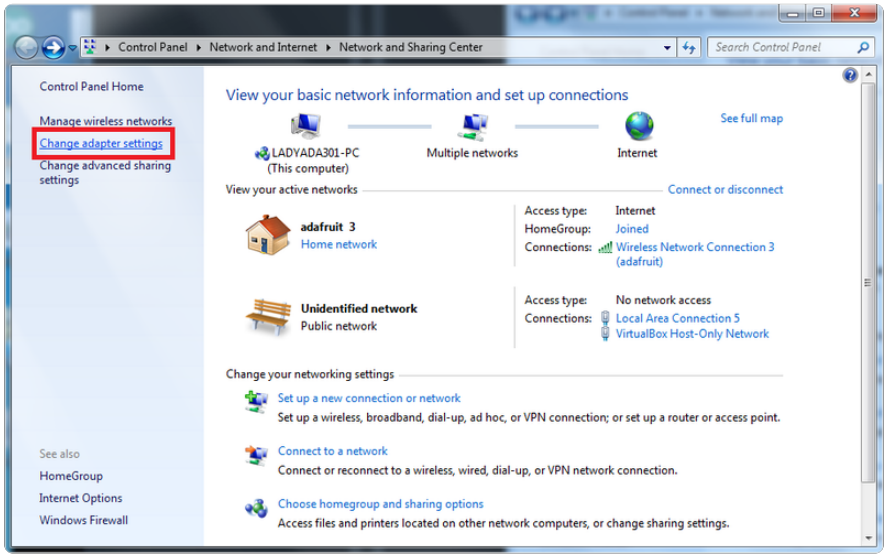


Some versions of windows may mis-interpret the PI as a COM port and you must manually force or install Microsoft RNDIS driver usage in Device Manager by right-click>Update Driver Software>Browse my computer>Pick from a list>Network Adapters>Microsoft>Remote NDIS compatible device.

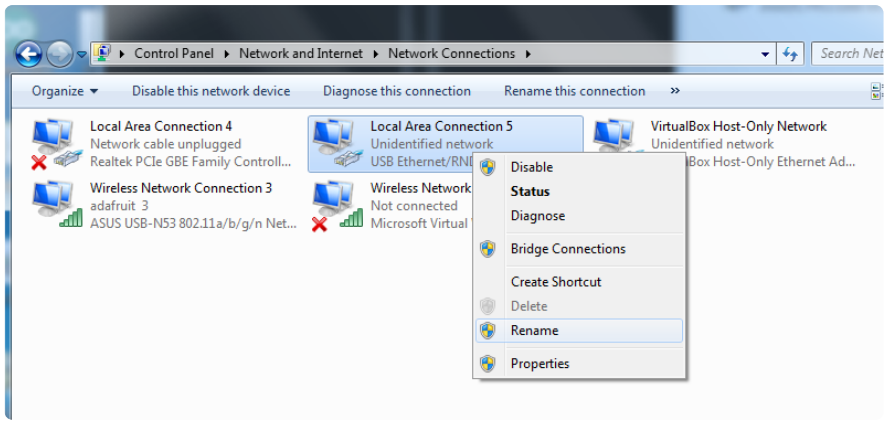
Check the Device Manager to check that it is a new network adapter



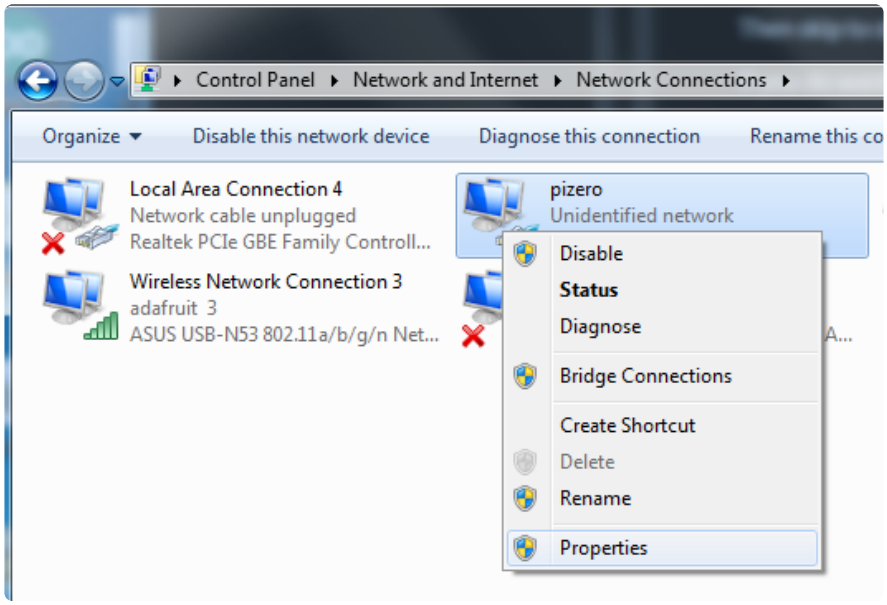
Open up **Network and Sharing Center** and click on **Change Adapter Settings**



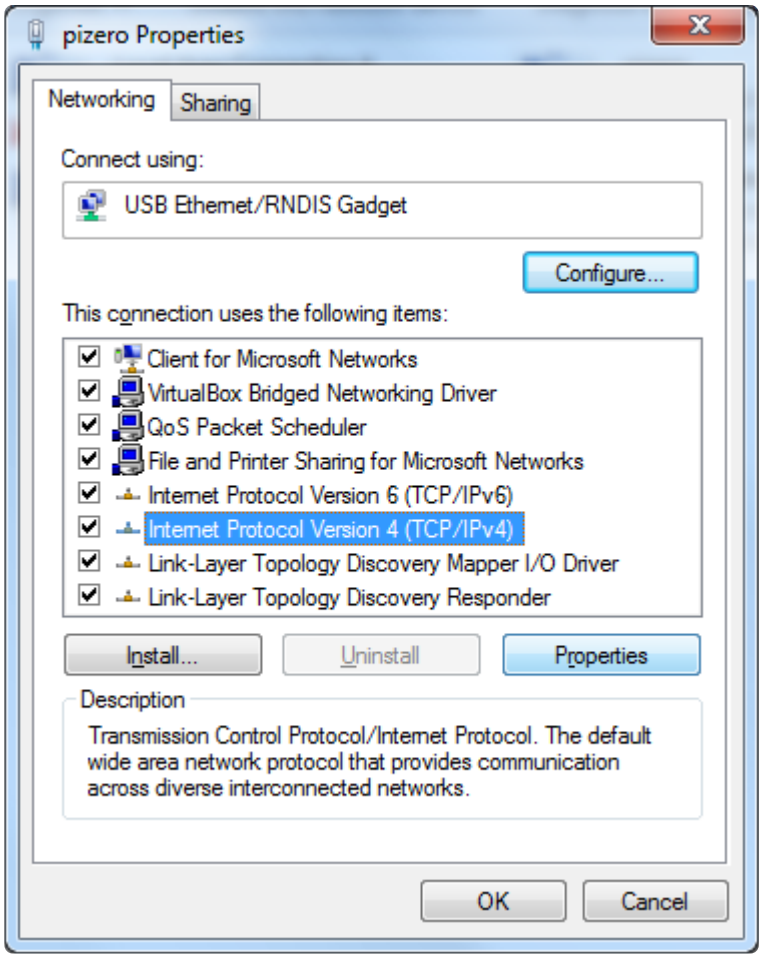
You'll see a list of all the myriad adapters you have. I have a lot but you'll likely only have 2 or 3. Find the RNDIS adapter and rename it **pizero** (makes it easier to find)



Then right-click and select **Properties...**

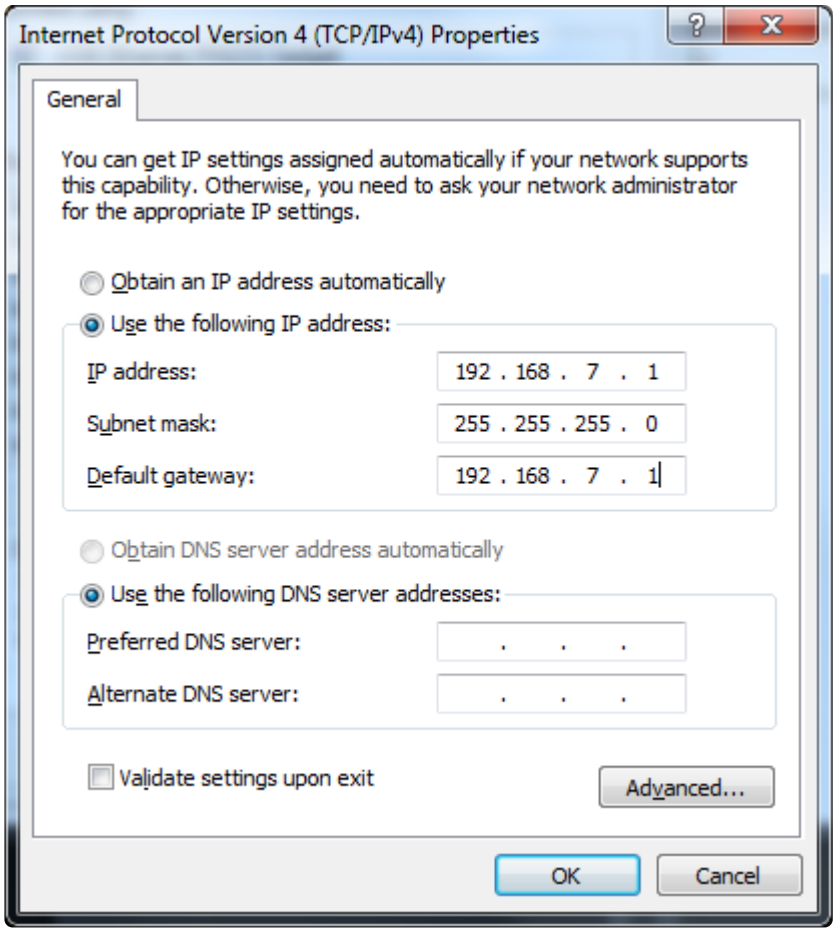


And select the **Internet Protocol Version 4 (TCP/IPv4)** from the connection list and click **Properties**

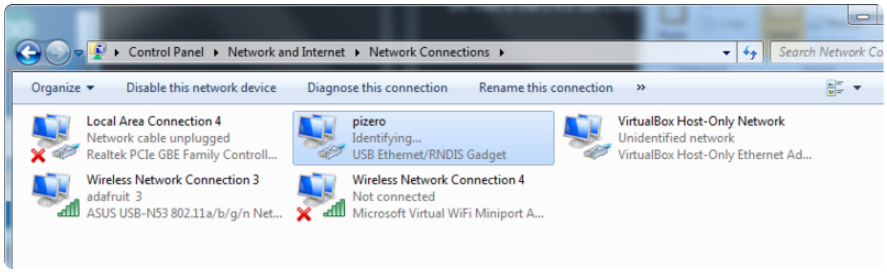


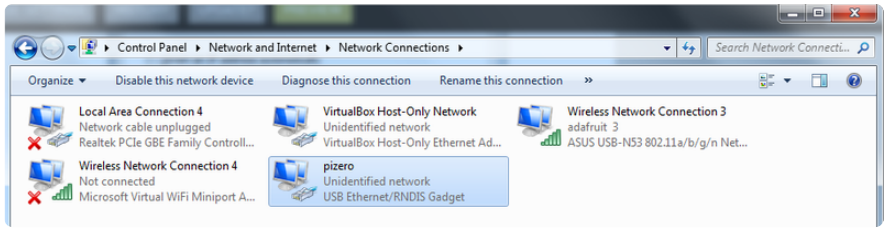
Enter in **192.168.7.1** as the computer's IP address and gateway (the gateway got erased later, I think Windows just automatically uses the IP address if they're the same) the subnet mask is **255.255.255.0** same as the Pi's

There's no DNS address

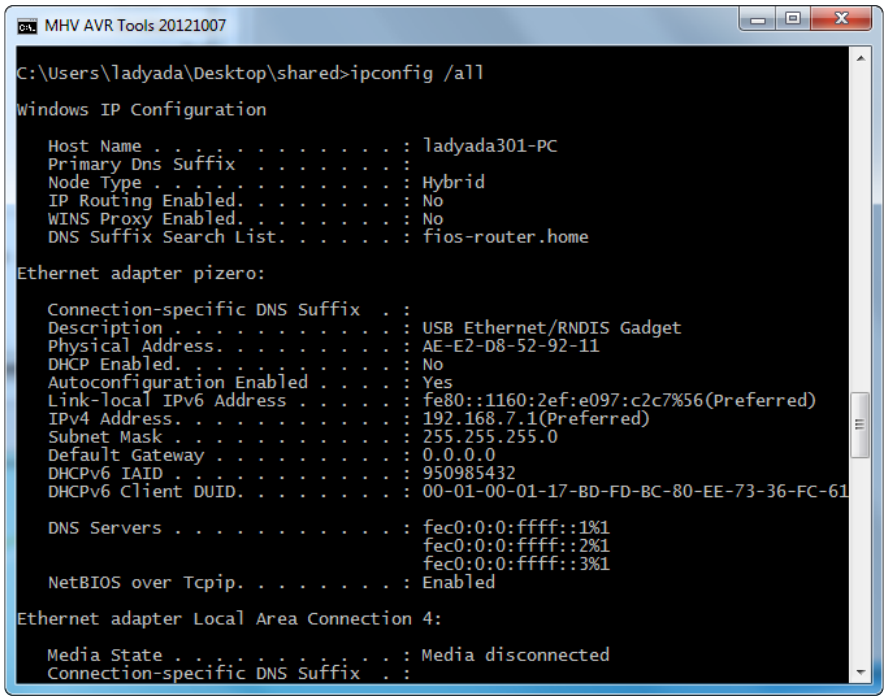


I unplugged & replugged in the Pi Zero, Windows will then identify the network.

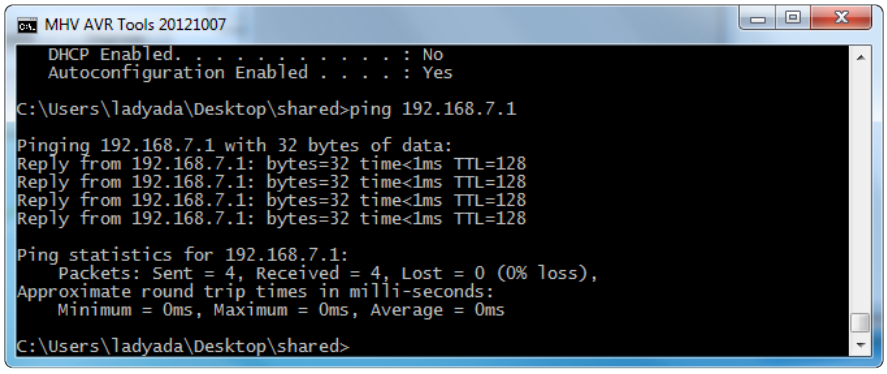




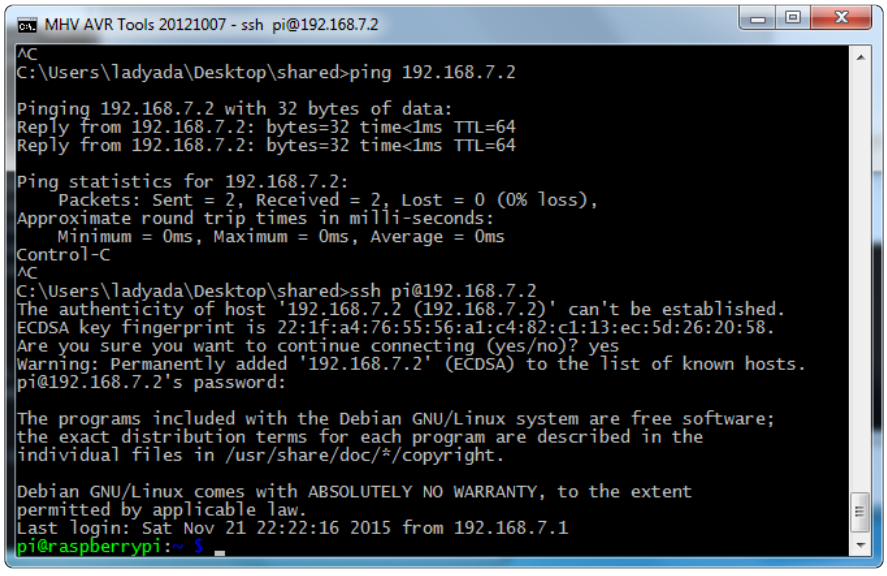
Now you can use a command box to run **ipconfig /all** if you want to check out the stats on the connection



and **ping 192.168.7.2** (the pi)



and even **ssh!**



Ethernet Tweaks

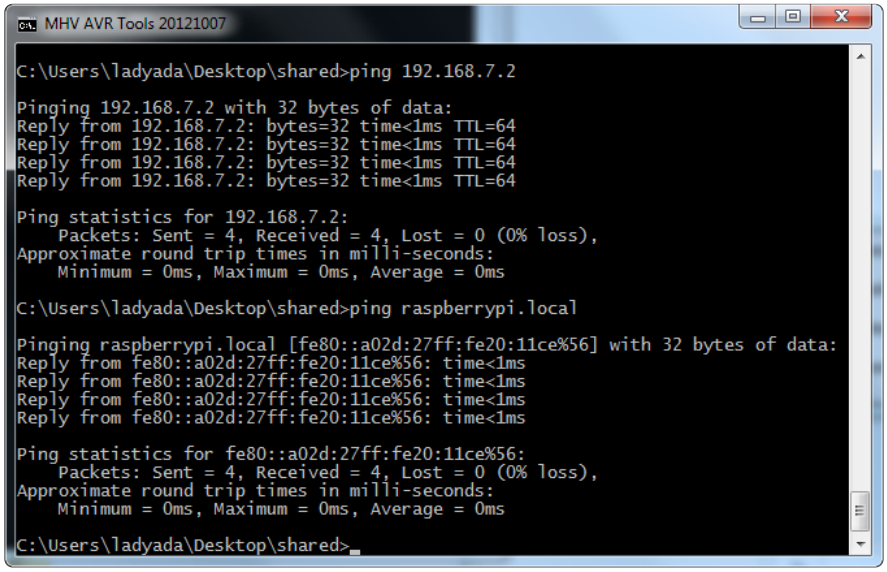
Using mDNS/Bonjour Naming

If you don't want to have to remember your Pi's IP address, you don't have to! Jessie Lite includes and automatically enables **avahi** which lets you use names like **raspberrypi.local**

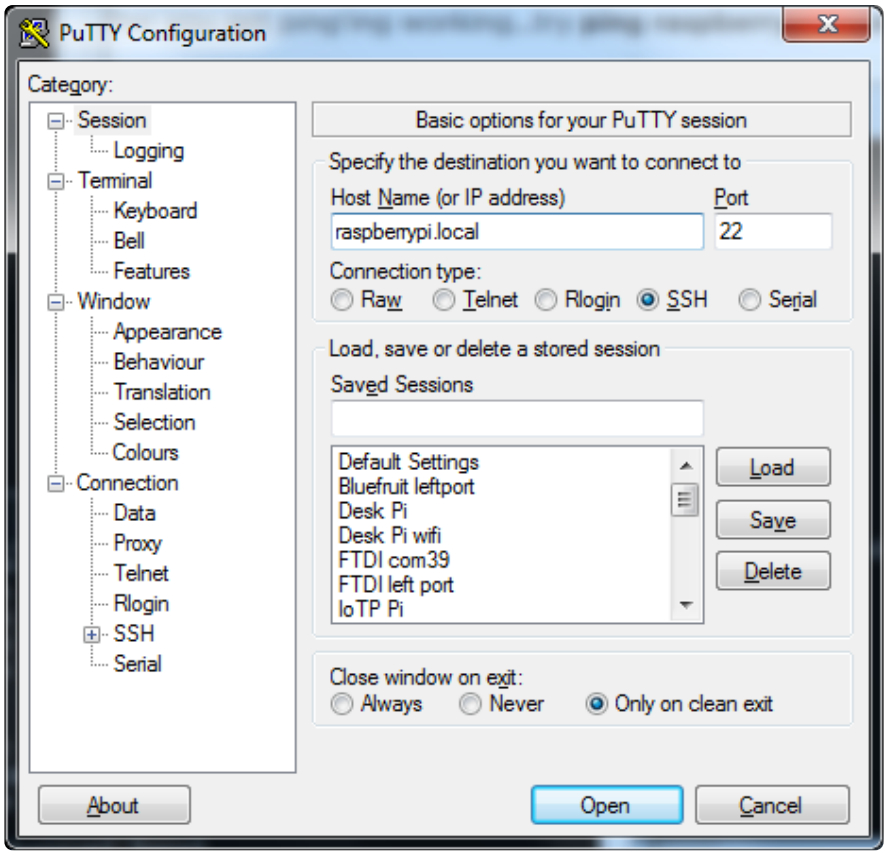
If for some reason its not activated, we have a full tutorial that will help you get set up. (https://adafru.it/khB)

Don't forget, Windows doesn't have native Bonjour support, so download & install Bonjour Print Services!
(check the tutorial above for a link on where/how to install, you only have to do it once)

So, after you get ping'ing working...try **ping raspberrypi.local**

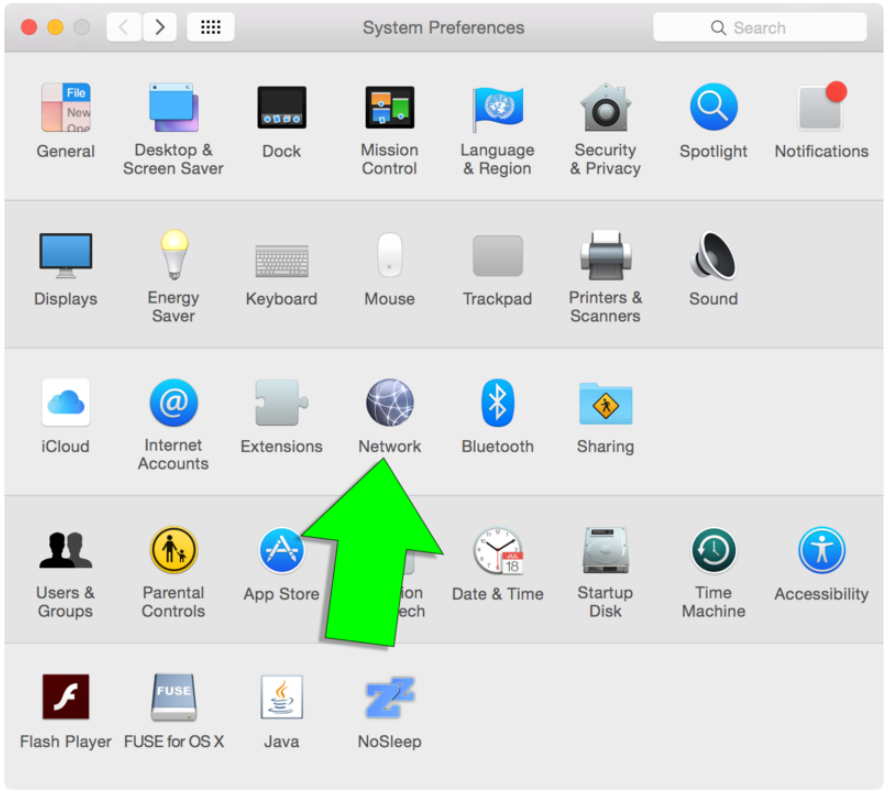


Or for ssh, it's also perfectly fine:

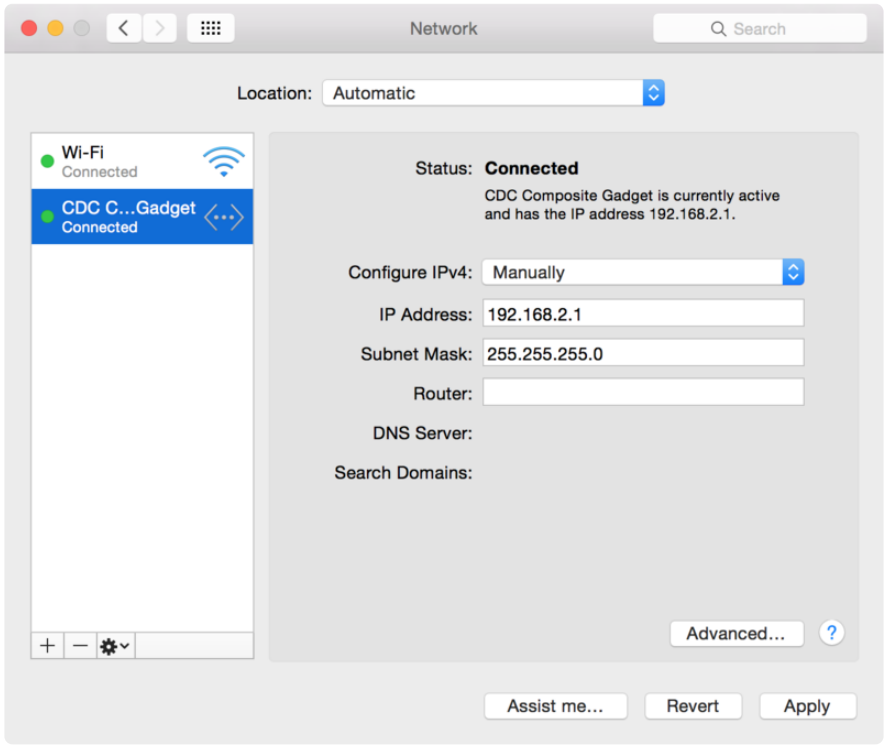


Sharing Network Access to Your Pi

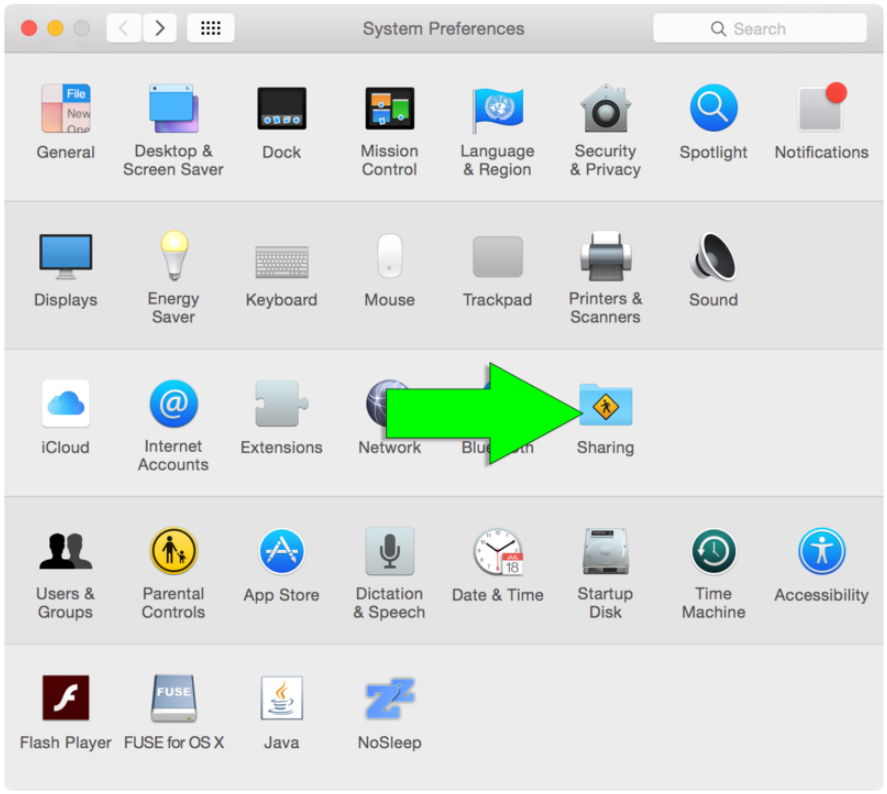
On OS X, open the **Network** tab of System **Preferences.**



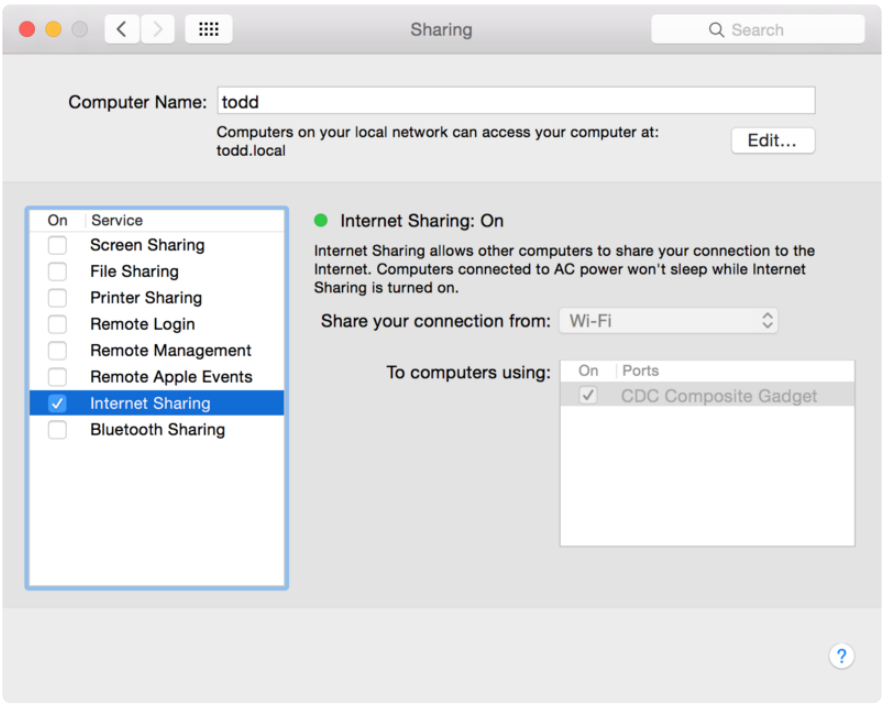
Select the existing **CDC** or **RNDIS** USB connection to your Raspberry Pi by selecting **Manually** from the Configure IPv4 menu. Use **192.168.2.1** for the IP Address, and **255.255.255.0** for the Subnet Mask. Click Apply to save your changes.



Then, open the **Sharing** tab in System Preferences.



Turn on **Internet Sharing** to share your existing internet connection from Wi-Fi or ethernet with the **CDC** or **RNDIS** Raspberry Pi connection.



Edit your /etc/network/interfaces file on your Pi to match the one below.

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo usb0
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug usb0
iface usb0 inet manual
```

The important lines are:

```
auto lo usb0
```

and also:

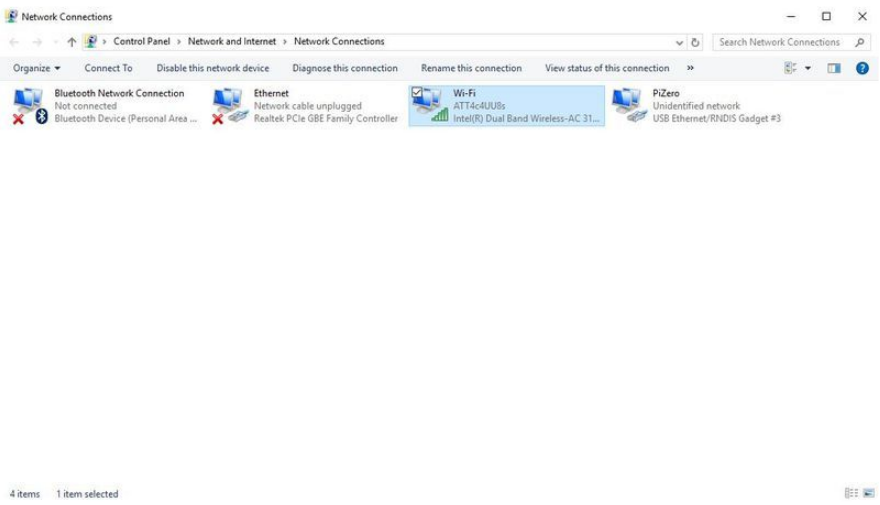
```
allow-hotplug usb0
iface usb0 inet manual
```

Restart your Pi using **sudo reboot**, and SSH back in to it using **ssh pi@raspberrypi.local**. You can then attempt to **ping google.com.**

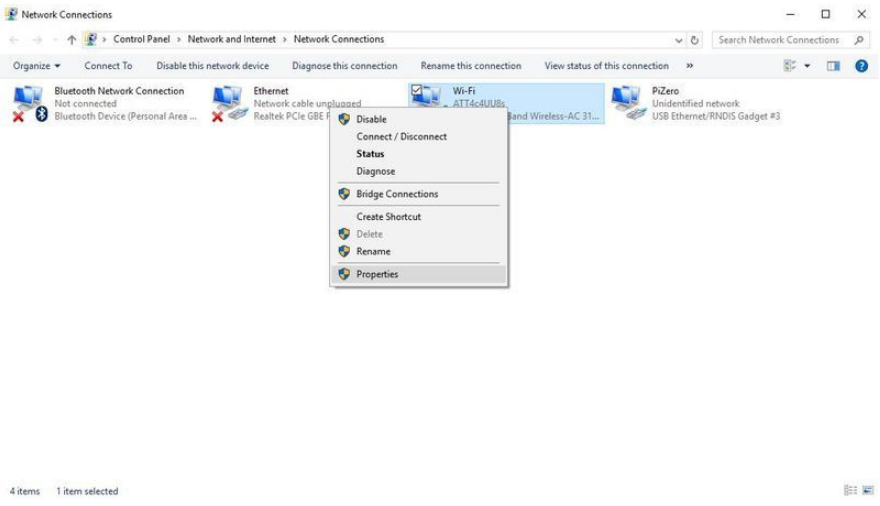
```
$ ping -c 5 google.com
PING google.com (216.58.219.238): 56 data bytes
64 bytes from 216.58.219.238: icmp_seq=0 ttl=55 time=20.975 ms
64 bytes from 216.58.219.238: icmp_seq=1 ttl=55 time=20.904 ms
64 bytes from 216.58.219.238: icmp_seq=2 ttl=55 time=20.646 ms
64 bytes from 216.58.219.238: icmp_seq=3 ttl=55 time=20.401 ms
64 bytes from 216.58.219.238: icmp_seq=4 ttl=55 time=20.379 ms

--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.379/20.661/20.975/0.247 ms
```

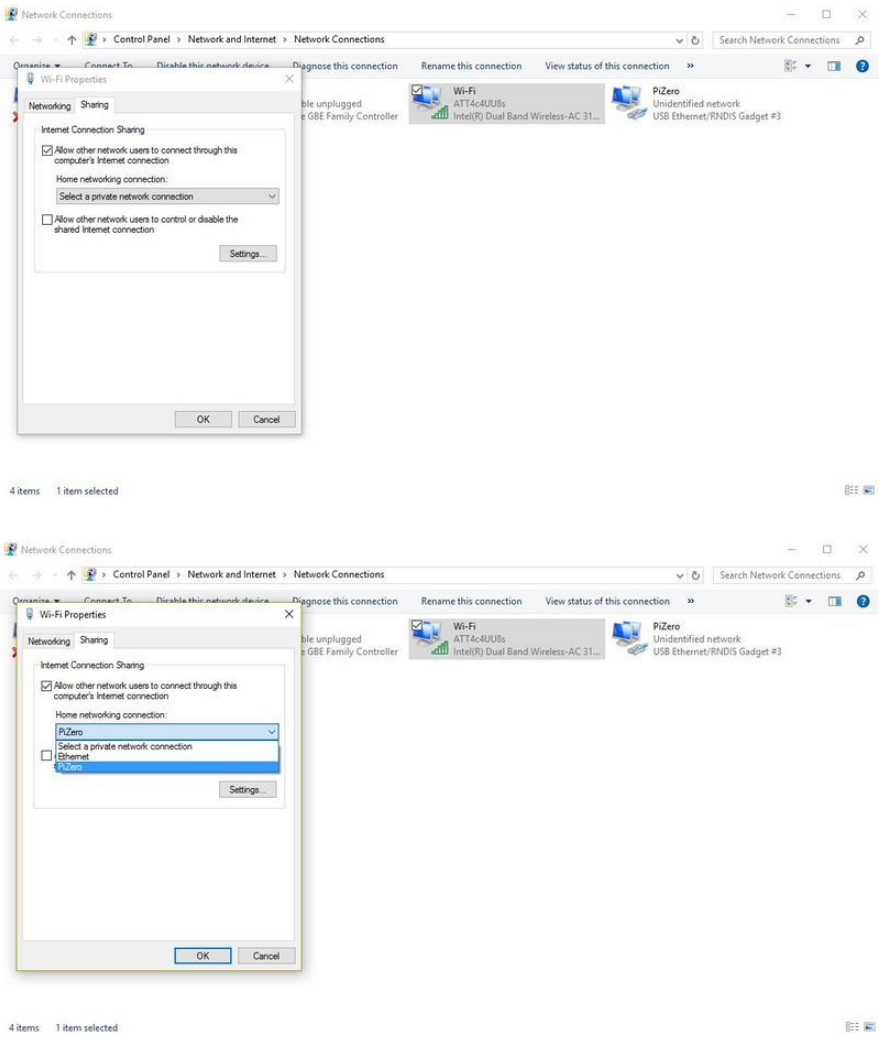
If using Windows, open **Network and Sharing Center** and click on **Change Adapter Settings**



Right-Click on your internet connection and select **Properties**.



Select the **Sharing** tab. Click the checkbox if it is not already checked. Then click on **Select a private network connection** and select **PiZero** from the dropdown.



Restart your Pi using **sudo reboot**, and SSH back in to it using **ssh pi@raspberrypi.local**. You can then attempt to **ping google.com.**

```
$ ping -c 5 google.com
PING google.com (216.58.219.238): 56 data bytes
64 bytes from 216.58.219.238: icmp_seq=0 ttl=55 time=20.975 ms
64 bytes from 216.58.219.238: icmp_seq=1 ttl=55 time=20.904 ms
64 bytes from 216.58.219.238: icmp_seq=2 ttl=55 time=20.646 ms
64 bytes from 216.58.219.238: icmp_seq=3 ttl=55 time=20.401 ms
64 bytes from 216.58.219.238: icmp_seq=4 ttl=55 time=20.379 ms

--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.379/20.661/20.975/0.247 ms
```

IP Addressing Options

On newer versions of Raspbian, the IP addressing for all network cards is done on the Pi via the program called dhcpcd. If you just want to set a static IP address, you can edit the **/etc/dhcpcd.conf** file, but we're going to take a different approach.

This page in the guide will walk you through:

- Disabling dhcpcd
- Setting your IP address on usb0 manually
- Setting up the l0 and wlan0 interfaces to act normally
- Run your own DHCP server on the usb0 port, so your Pi can provide an address to your Linux or Windows PC or Mac without any additional software on your desktop or laptop.

Disabling dhcpcd

First, let's disable dhcpcd. This is non-destructive, but when we run this command dhcpcd won't be able to assign addresses anymore, so you should be logged in locally for this with a monitor attached.

```
sudo systemctl disable dhcpcd
```

Setting up the interfaces

Now let's setup your interfaces manually since dhcpcd won't be doing it anymore. Go ahead and run:

```
sudo nano /etc/network/interfaces
```

In there you will probably see something in the file that says:

```
source-directory /etc/network/interfaces.d
```

Go ahead and leave that in. Below it, add all of this text, paying attention to the indents:

```
auto lo
iface lo inet loopback
```

```
auto usb0
allow-hotplug usb0
iface usb0 inet static
address 10.77.77.77
netmask 255.255.255.0

allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

The interfaces we've assigned are l0 (loopback, which is needed, just not in scope of this guide), usb0 (which we assigned a static IP address of 10.77.77.77), and wlan0, which will still connect to WiFi normally. Go ahead and save the file and close the editor.

Next, let's make sure your phone's hotspot connection is in **/etc/wpa_supplicant/wpa_supplicant.conf**.

Run:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

In there, each network should be listed like this:

```
network={
ssid="Your-Home-SSID"
psk="yourpassphrase"
}
```

Modify it to match your home network's wireless info - if it's already there, you don't need to add it. You can add this section for each wireless network you'll want the Pi to connect to.

Run your own DHCP Server

Next we'll install dnsmasq, which will let us use DHCP to assign IP addresses to PCs or Macs that connect to the USB port on the Pi. Simply run:

```
sudo apt-get install -y dnsmasq
```

Configuration is easy - just run:

```
sudo nano /etc/dnsmasq.conf
```

Add the following lines at the bottom:

```
dhcp-range=10.77.77.78,10.77.77.99,12h
dhcp-option=3
dhcp-option=6
```

The DHCP range will need to match the interface IP address we assign to the usb0 interface, and this option will assign addresses between 10.77.77.78 and .99, with a 12 hour lease. That should be more than enough. If you need to change the IP range for some reason, make sure to match the configuration of usb0 with these items. We also us DHCP options 3 and 6 - they are annotated in the config file, but they prevent dnsmaq from advertising a default route or DNS - we don't need this Pi to be a DNS server or a router for this tutorial.

Go ahead and save and exit from the file, we won't start dnsmasq just yet though. When rebooting, please give your Pi time to start all the services, get a WiFi address, and assign one to your PC. The Pi Zero W is a little slower.

Go ahead and safely shut the Pi down with the following command:

```
sudo halt
```

Checking it out

Once the Pi is halted, you should be able to see nothing going on with the display and safely unplug it from power. Next you can simply plug a USB data cable (make sure it's not a charging cable) to the micro USB port closest to the center of the Pi, and the other end to your PC. You'll hear a sound and see drivers installing on Windows 10 and newer, and on all systems you'll see a new network card. You should be able to simply connect with:

```
ssh -l pi 10.77.77.77
```

From your PC and get the login prompt on the Pi. The Pi will also independently connect to WiFi, which can be handy if you're testing a different wireless network, connecting to a WiFi hotspot, etc.

Other Modules!

Serial and Ethernet are the easiest to get going but they are far from the only gadgets the Linux kernel supports. You can also try such options as:

- **Mass storage** (you can have the Pi appear as a 'USB key' disk drive) - note, we didn't get this up and running smoothly, it enumerated but disk access to the backing file didnt work on our windows machine
- **MIDI** - shows up as a 'native' USB MIDI audio device
- **HID** - appear to the host computer as a mouse/keyboard/joystick
- **Audio** - Show up as an audio/speaker device & line in as well?
- **Composite** - a mix of serial/ethernet/mass storage composite devices is available. Note that this may work on a Mac or Linux but for windows you'd need a custom driver
- **Printer, webcam, etc** - There's about a dozen more options

For more details, check out the USB gadget API framework page (https://adafru.it/klc)

Sunxi also has a handy page (https://adafru.it/kld)

We compiled all of the available USB gadget modules into the December 25, 2015 (or later) kernel tgz. You can enable them by using **modprobe** or editing the **/etc/modules** file to enable. If they need options, creating a new file for those options in **/etc/modprobe.d/usbgadget.conf** or similar

In particular, here's the modules that are available:

```
#
# USB Peripheral Controller
#
# CONFIG_USB_FUSB300 is not set
# CONFIG_USB_FOTG210_UDC is not set
# CONFIG_USB_GR_UDC is not set
# CONFIG_USB_R8A66597 is not set
# CONFIG_USB_PXA27X is not set
# CONFIG_USB_MV_UDC is not set
# CONFIG_USB_MV_U3D is not set
# CONFIG_USB_M66592 is not set
# CONFIG_USB_BDC_UDC is not set
# CONFIG_USB_NET2272 is not set
# CONFIG_USB_GADGET_XILINX is not set
# CONFIG_USB_DUMMY_HCD is not set
CONFIG_USB_LIBCOMPOSITE=m
CONFIG_USB_F_ACM=m
CONFIG_USB_F_SS_LB=m
CONFIG_USB_U_SERIAL=m
CONFIG_USB_U_ETHER=m
CONFIG_USB_F_SERIAL=m
CONFIG_USB_F_OBEX=m
```

```
CONFIG_USB_F_NCM=m
CONFIG_USB_F_ECM=m
CONFIG_USB_F_EEM=m
CONFIG_USB_F_SUBSET=m
CONFIG_USB_F_RNDIS=m
CONFIG_USB_F_MASS_STORAGE=m
CONFIG_USB_F_FS=m
CONFIG_USB_F_UAC1=m
CONFIG_USB_F_UAC2=m
CONFIG_USB_F_UVC=m
CONFIG_USB_F_MIDI=m
CONFIG_USB_F_HID=m
CONFIG_USB_F_PRINTER=m
CONFIG_USB_CONFIGFS=m
CONFIG_USB_CONFIGFS_SERIAL=y
CONFIG_USB_CONFIGFS_ACM=y
CONFIG_USB_CONFIGFS_OBEX=y
CONFIG_USB_CONFIGFS_NCM=y
CONFIG_USB_CONFIGFS_ECM=y
CONFIG_USB_CONFIGFS_ECM_SUBSET=y
CONFIG_USB_CONFIGFS_RNDIS=y
CONFIG_USB_CONFIGFS_EEM=y
CONFIG_USB_CONFIGFS_MASS_STORAGE=y
CONFIG_USB_CONFIGFS_F_LB_SS=y
CONFIG_USB_CONFIGFS_F_FS=y
CONFIG_USB_CONFIGFS_F_UAC1=y
CONFIG_USB_CONFIGFS_F_UAC2=y
CONFIG_USB_CONFIGFS_F_MIDI=y
CONFIG_USB_CONFIGFS_F_HID=y
CONFIG_USB_CONFIGFS_F_UVC=y
CONFIG_USB_CONFIGFS_F_PRINTER=y
CONFIG_USB_ZERO=m
CONFIG_USB_AUDIO=m
# CONFIG_GADGET_UAC1 is not set
CONFIG_USB_ETH=m
CONFIG_USB_ETH_RNDIS=y
CONFIG_USB_ETH_EEM=y
# CONFIG_USB_G_NCM is not set
CONFIG_USB_GADGETFS=m
CONFIG_USB_FUNCTIONFS=m
CONFIG_USB_FUNCTIONFS_ETH=y
CONFIG_USB_FUNCTIONFS_RNDIS=y
CONFIG_USB_FUNCTIONFS_GENERIC=y
CONFIG_USB_MASS_STORAGE=m
CONFIG_USB_G_SERIAL=m
CONFIG_USB_MIDI_GADGET=m
CONFIG_USB_G_PRINTER=m
CONFIG_USB_CDC_COMPOSITE=m
CONFIG_USB_G_ACM_MS=m
CONFIG_USB_G_MULTI=m
CONFIG_USB_G_MULTI_RNDIS=y
CONFIG_USB_G_MULTI_CDC=y
CONFIG_USB_G_HID=m
CONFIG_USB_G_DBGP=m
# CONFIG_USB_G_DBGP_PRINTK is not set
CONFIG_USB_G_DBGP_SERIAL=y
CONFIG_USB_G_WEBCAM=m
# CONFIG_USB_LED_TRIG is not set
# CONFIG_UWB is not set
CONFIG_MMC=y
# CONFIG_MMC_DEBUG is not set
```

Compiling your own kernel? Here's the v4.4 .config we used

https://adafru.it/kle

You'll also have to patch the 'common' rpi overlay as shown here (https://adafru.it/khf)

Old Kernel Install

This is the older, no longer required technique - documented in case you need it!

Step 0. Download new Kernel Package

Download the following onto your desktop computer:

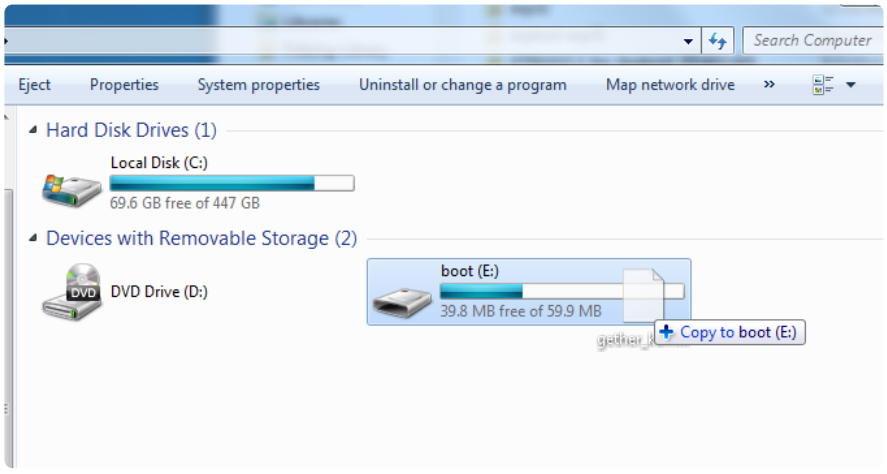
Download the modular Gadget Kernel TGZ file

https://adafru.it/klb

and rename it **gadgetkernel.tgz**

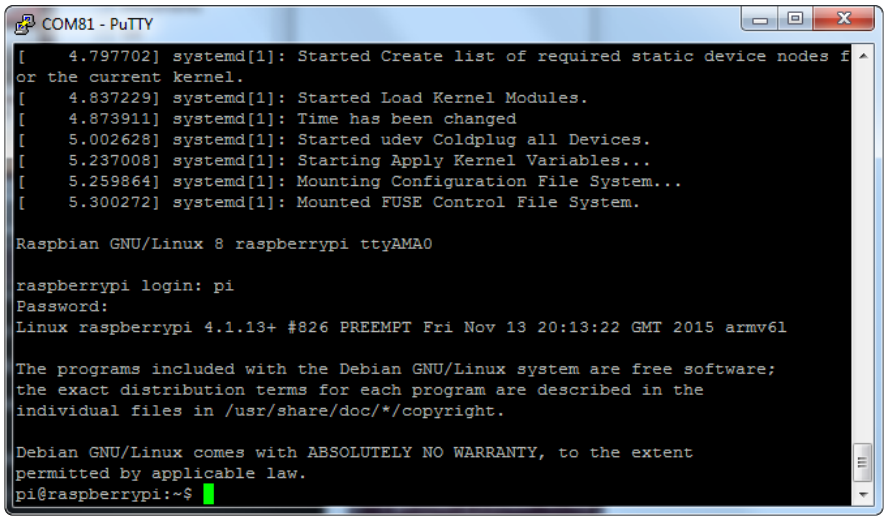
Step 1. Copy New Kernel to SD Card

Copy the new kernel file over to the **boot** directory of the Jessie Lite card. After you're done burning the SD image, don't eject it just yet. Drag the **kernel.tgz** file over to the SD card. This way you can ferry the kernel into your Pi without needing network



Step 2. Log into your Pi Zero

Insert the SD into your Pi Zero, connect the console cable, power the Pi & log into via the USB console.

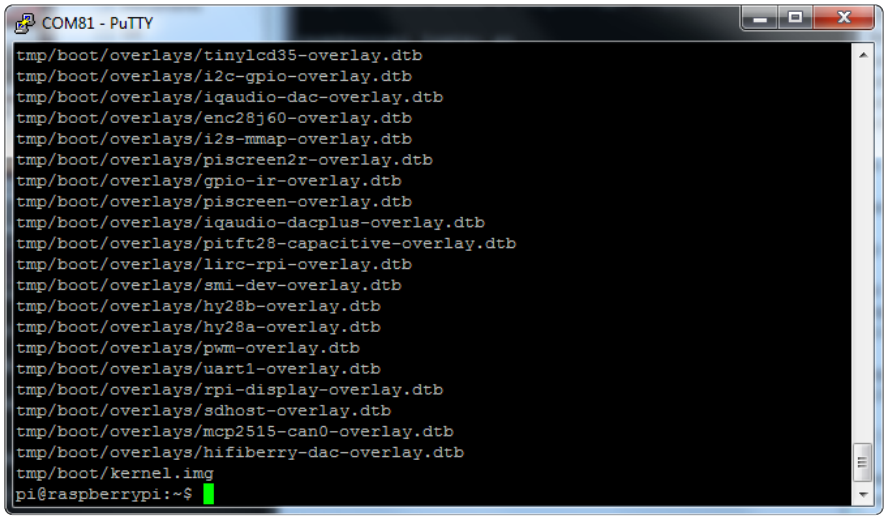


Step 3. Uncompress new kernel package

Uncompress and install the **kernel .tgz file**

run the following commands:

- **cd ~**
- **sudo mv /boot/gadgetkernel.tgz .**
- **tar -xvzf gadgetkernel.tgz**



You'll see a long stream of file names ending with **tmp/boot/kernel.img**

You may see a bunch of complaints about timestamps being in the future, this is totally OK

Step 4. Backup and Install new Kernel

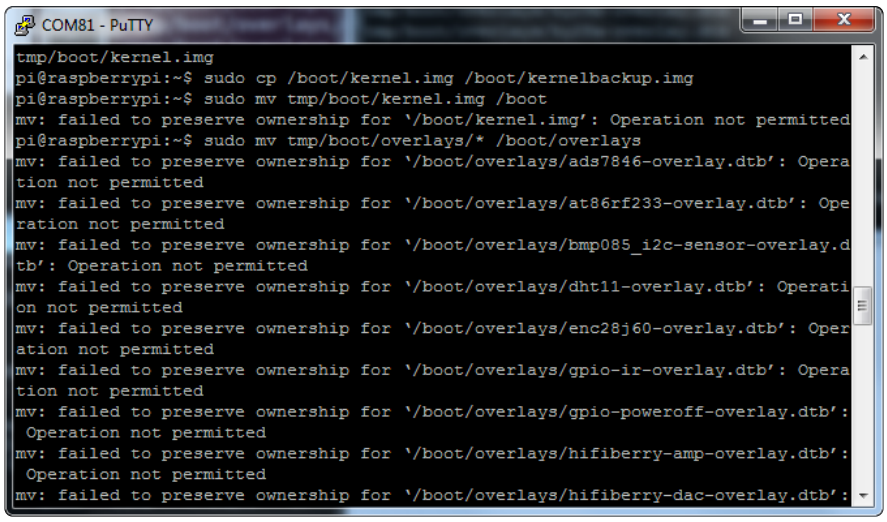
Run

- **sudo mv /boot/kernel.img /boot/kernelbackup.img**

to make a backup of the current kernel. Now run

- **sudo mv tmp/boot/kernel.img /boot**

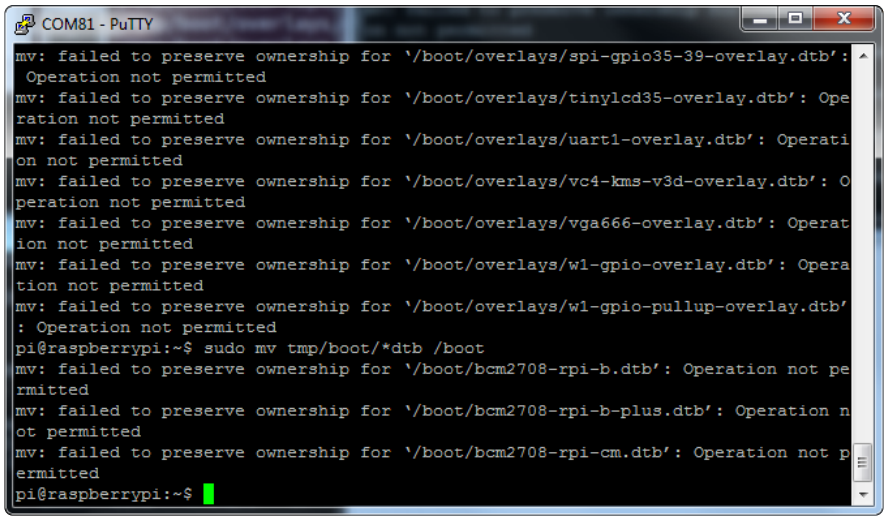
You may see complaints about preserving ownership, you can ignore them

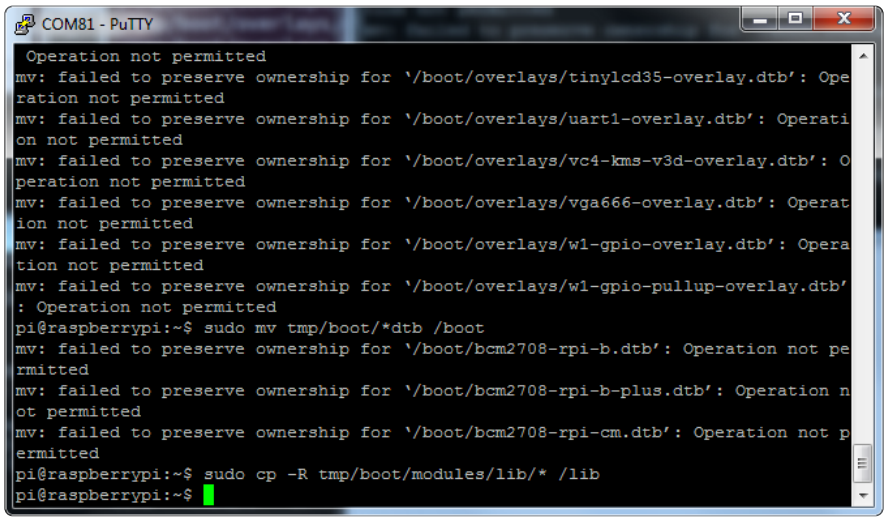


Step 5. Install Overlays & Modules

Run the commands to install the new overlays & modules

- **sudo mv tmp/boot/overlays/* /boot/overlays**
- **sudo mv tmp/boot/*dtb /boot**
- **sudo cp -R tmp/boot/modules/lib/* /lib**





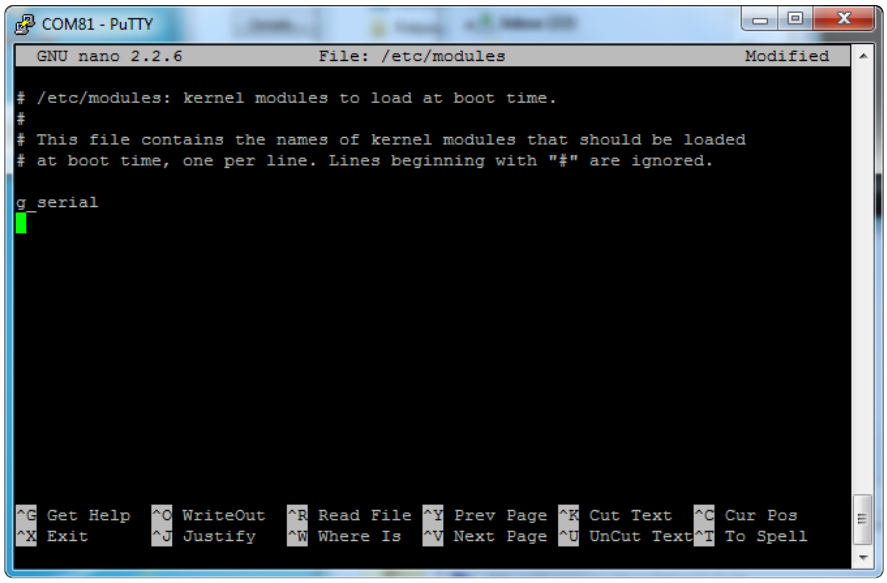
Gadget Serial!

Now we'll tell the Pi we want to use the **g_serial** module

Run

- **sudo nano /etc/modules**

and add **g_serial** on a single line at the end, then save



[Continue from this step for the rest of Serial Gadget setup and testing (https://adafru.it/q1c)](https://adafru.it/q1c)

Gadget Ethernet!

Now we'll tell the Pi we want to use the **g_ether** module

Run

- **sudo nano /etc/modules**

and add **g_ether** on a single line at the end, then save