Instructions for Creating a Scopify Pi

Thursday, April 28, 2016 10:21 PM

These steps will allow a Raspberry Pi of nearly any form to become a WiFi->Serial Bridge. The main purpose of this code is to enable telescopes to be controlled wirelessly from a phone, tablet, or computer. The capability of a WiFi bridge is certainly useful in other fields but that is not the main goal of this project.

With these instructions, I will go into the maximum detail that I think someone who has a Raspberry Pi will need. This means that I won't spoon feed, but I will give every step required. If you think I should add or subtract, please leave a comment in the discussion board.

1. Download and unzip Raspbian Jesse Lite from: https://www.raspberrypi.org/downloads/raspbian/
2. Using your program of choice, image an SD-card that fits your Pi with that the Raspbian Jesse .bin file. I use Win32Disk Imager on my Windows 10 PC.
3. With Raspbian on the SD Card, put it into the Pi. Hook the Pi up to a keyboard, WiFi adapter (if not built in), monitor, and finally power.
4. The Pi will boot up and ask for a login and password. These will be the defaults: Login = pi , Password = raspberry
5. This is a good time to remind you that case counts in Linux. "Raspberry" wouldn't work for the password. Similarly as we are making changes to files, case counts. I spent considerable time diagnosing a problem where I had typed: ttyama0 where the correct term was ttyAMA0.
6. The first thing to do is to enable WiFi.
7. Let's talk about WiFi. We'll be creating a soft WiFi Access point using the WiFi adapter. This is do-able with many of the RPi compatible adapters, but with the current version of Raspbian it requires recompiling drivers, recompiling hostapd, and installing a bunch of required libraries. I've done it, it works, it's a pain. For a few bucks more than the cheap Edimax adapter that everybody has, you can get the "Official" Raspberry Pi adapter that uses a Broadcom Chipset that just works. It's the same chipset as the Raspberry Pi 3. It makes life much easier, so that's what I'll be assuming that you have. If you'd like help in getting other adapters to work, leave a comment in the discussion forum.
8. So, back to enabling WiFi: Type:
    a. **sudo nano /etc/wpa_supplicant/wpa_supplicant.conf**
    b. When that opens in nano (the program that we'll use to edit files), use the keyboard to scroll to the end of the text and add the following lines:
        1. **network={**
            ssid="ssid"
            psk="password"
            }
        2. **NOTE!!! We haven't yet changed the keyboard layout and the " is not in the normal place if you are using a U.S. keyboard. For me quote was shift+2 (usually @). We'll fix this later.**
            i. Use the quotes but replace ssid and password with the correct info for whatever WiFi connection you have right now. This is NOT where we are creating a WiFi access point for use in the field. The idea here is that you'll be able to configure the RPi via WiFi

without having to have a monitor or keyboard connected to it.

   c. With the network info added to wpa_supplicant.conf, type: "**ctrl+o**" (the control button and the letter o at the same time). Nano will then prompt you to save wpa_supplicant.conf. Hit enter to accept. The type "**ctrl+x**" to exit nano.
   d. You should now be able to connect to WiFi. For good measure, let's reboot the Pi to make sure wpa_supplicant.conf get's loaded: Type: **sudo reboot**.
   e. Once the RPi reboots, type: **ifconfig wlan0** you should see an ip address in the inet addr field. If you don't please consult the googles for help on getting WiFi up.

9. With WiFi enabled, let's update the repositories. Type: **sudo apt-get update** this may take a while. Accept any thing that it asks about adding additional data due to the update. Once that is done type: **sudo apt-get upgrade** . This will also take a while. Again say yes (type Y) to whatever it asks.
10. Ok now that we can access the apt-get repositories, let's install some software:
11. Type: **sudo apt-get install ser2net dnsmasq hostapd** . Again, type Y for whatever it asks.
12. We now need to disable dnsmasq and hostapd from coming up on their own upon reboot. We want to control when they operate:
13. Type: **sudo systemctl disable dnsmasq** then **sudo systemctl disable hostapd**
14. Next we'll configure a bunch of OS features using the built in RPi configuration GUI.
15. Type: **sudo raspi-config**
16. We'll go down the list of options:
    1. Expand Filesystem. Do it.
    2. Change User Password: Your call here. I changed the password to something easier to type than raspberry.
    3. Boot Options: No changes here
    4. Wait for Network at Boot: Change to Yes. This makes the script to discover if we are connected to a network work correctly.
    5. Internationalization Options: Go into the sub menu
        i. I1 Change Local: Scroll Down choose the correct local for where you are. I'm in the US and choose all the "us" options.
        ii. I2 Change Timezone. Do it.
        iii. I3 Change Keyboard Layout. Pick what you have. I picked Generic 104-key PC. Then choose the layout. If you don't live in Great Brittan choose other and then pick the country you are in. I'm in the U.S. and chose English (US)
    6. Enable Camera: No change here
    7. Add to Rastrack: No change here
    8. Overclock: No Change here
    9. Advanced Options: Go into the sub menu.
        i. I changed the HOSTNAME to: scope for easy identification on the network.
        ii. Enable SSH
        iii. **DISABLE Serial** (I know this doesn't make sense, but if you have it enabled here, it prevents ser2net from operating)
        iv. No further changes.
    10. Click finish and then **sudo reboot**
    11. Use your new password to log in. Or fire up your PC / Mac and start using it to configure the RPi. I use Putty to SSH into the RPi. Raspbian has zero configuration network enabled by default, so you don't need to know the IP address of your RPi to SSH into it. Simply type

"yourhostname.local" into putty where it asks for the address. In my case it was: scope.local. Easy.

12. If you are at a PC you can now transfer the files provided in Github to your RPi. You can use secure copy (scp) or in Windows you can use WinSCP. I like WinSCP.
13. Transfer the files to the home directory of the Pi which is what comes up when you connect with WinSCP. That directory is /home/pi
14. With the files transferred over, go back to your RPi's keyboard or start your SSH session.
15. We'll move the files to the correct place now.
16. Make sure you are in your home directory. Type: **cd /home/pi**
17. Type the following:
    - i. **sudo mv ser2net.conf /etc/ser2net.conf**
    - ii. **sudo mv rc.local /etc/rc.local**
    - iii. **sudo mv hostapd.conf /etc/hostapd/hostapd.conf**
    - iv. **sudo mv access_point.conf /etc/dnsmasq.d/access_point.conf**
    - v. **sudo mv hostapd-systemd.service /lib/systemd/system/hostapd-system.service**
    - vi. **sudo mv checker.sh /usr/sbin/checker.sh**
    - vii. **cd /etc**
    - viii. **sudo chmod +x rc.local**
    - ix. **sudo chown root:root rc.local**
    - x. **sudo reboot**
18. That's it folks. The rest is just details. You may want to modify how I've got the WiFi access point setup. You can change the name of the AP, the pre shared key, and whether it's an open or close network in hostapd.conf. You can change the DHCP IP addressing scheme in access_point.conf . You can completely forgo the attempt to connect to your local WiFi network by removing the info from wpa_supplicant.conf that we initially setup, checker.sh will then always bring up the hostapd and dnsmaq services upon boot. Alternately, you can remove checker.sh completely from rc.local, then enable the hostapd and dnsmasq services using the command: systemctl enable (your service here).
19. checker.sh runs at the end of the boot up cycle. It checks if the WiFi adapter has an assigned IP address. If if does, the script does nothing and allows the wlan0 interface to continue to be connected to the network it joined. I find it very convenient to have the RPi connect to my local network so that my PC's, Phones, and tablets can all connect to the scope without having to connect to the scopes WiFi access point.
20. If the wlan0 interface does not have an IP address, then checker.sh starts the hostapd and dnsmasq services which starts the AP and the DHCP server. This will bring up a new WiFi network that you'll need to connect to in order to talk to the scope / mount.
21. Since zero-conf networking is enabled. Set your astro software to look for your hostname.local, again scope.local for me. This way it doesn't matter if the RPi is connected to your home WiFi or if it created it's own AP, the address in your software (like Sky Safari) will always be scope.local
22. Let's talk about ser2net. The configuration is pretty straight forward. In the last lines of the /etc/ser2net.conf file it configures which ports to listen to for serial data. I choose port 4030 since that was a Sky Safari default. If you want to change ports or need to change the baud rate or other serial configurations, then use nano to modify ser2net.conf to your telescopes requirements. The comments in ser2net.conf are pretty good, you'll figure it out.

17. Setting up a Serial port from the RPi to your mount / telescope