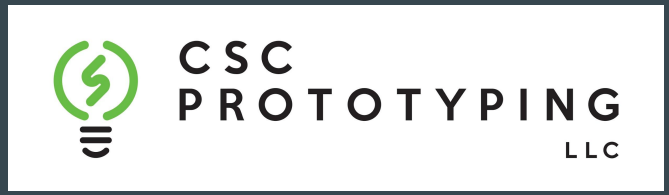
Rescue Repeater Software Installation & Configuration

•••

NXP HoverGames



dbrummund@brummundtechnologies.com

nwhitehouse@cscprototyping.com

Prepare to Program the Arduino Pro Mini

NOTE: The Arduino Pro Mini must be programmed prior to hardwiring it to the Pocket Beagle. The Pro Mini UART port used to communicate with the Pocket Beagle is the same port needed to connect to the Arduino IDE for programming.

Hardware needed to program the Pro Mini:

i) USB to serial converter (DSD Tech or equivalent)

ii) 22 AWG Solid Hookup Wire

iii) 9V Battery or other 6-12V DC Power source

iv) Windows, Mac or Linux PC

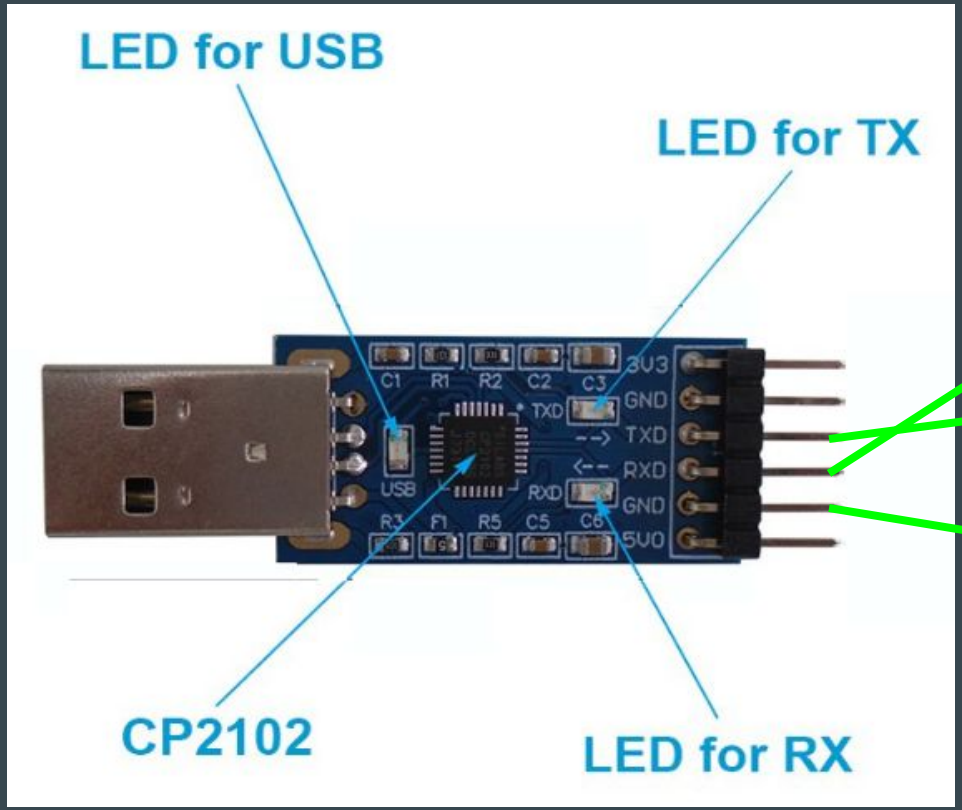
v) Arduino Pro Mini (5V Model used for demo)

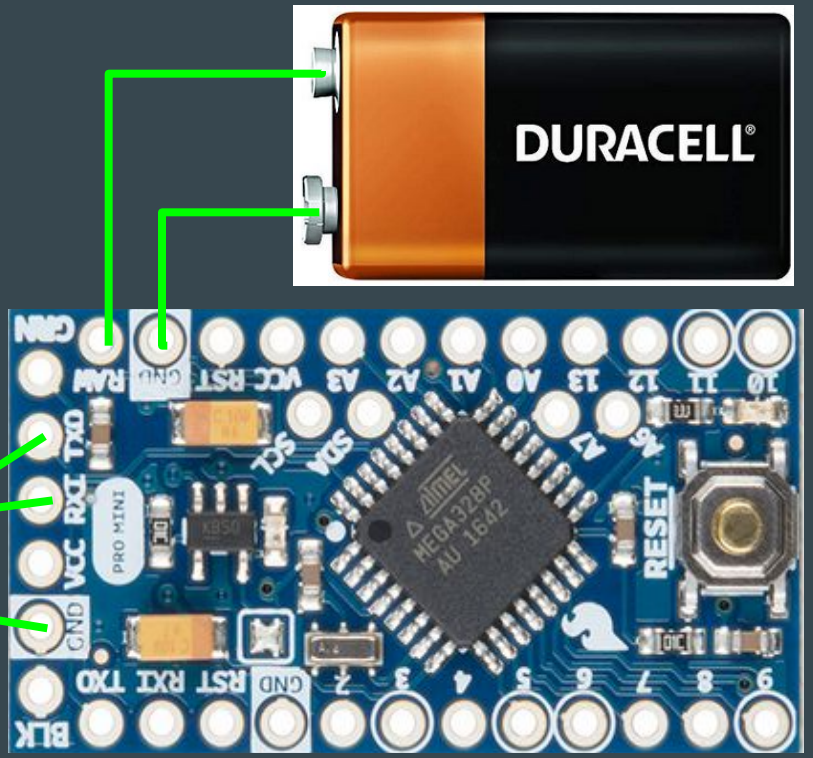
Hardware Setup

Solder 6” sections of hookup wire to the Pro Mini RXI, TXO, and RAW positions, and one of the GND positions. The destinations of these wires will be described in the following slides.

Note: The DSD Tech USB to serial converter has built in 5V and 3.3V voltage regulators. Since the 5V Arduino Pro Mini was used, it was decided to use a 9V battery to power the Pro Mini during programming. The 5V from the DSD Tech may be sufficient to use as the power source during programming (the Pro Mini voltage regulator has a very low dropout voltage) - but the risk of poor voltage regulation while programming is probably best avoided.

Setup Continued: Strip about ½” of insulation from the ends of the wires coming from the Arduino Pro Mini. The 22 AWG Solid core mates nicely with the female 2.54mm connections of the DSD Tech’s cable.





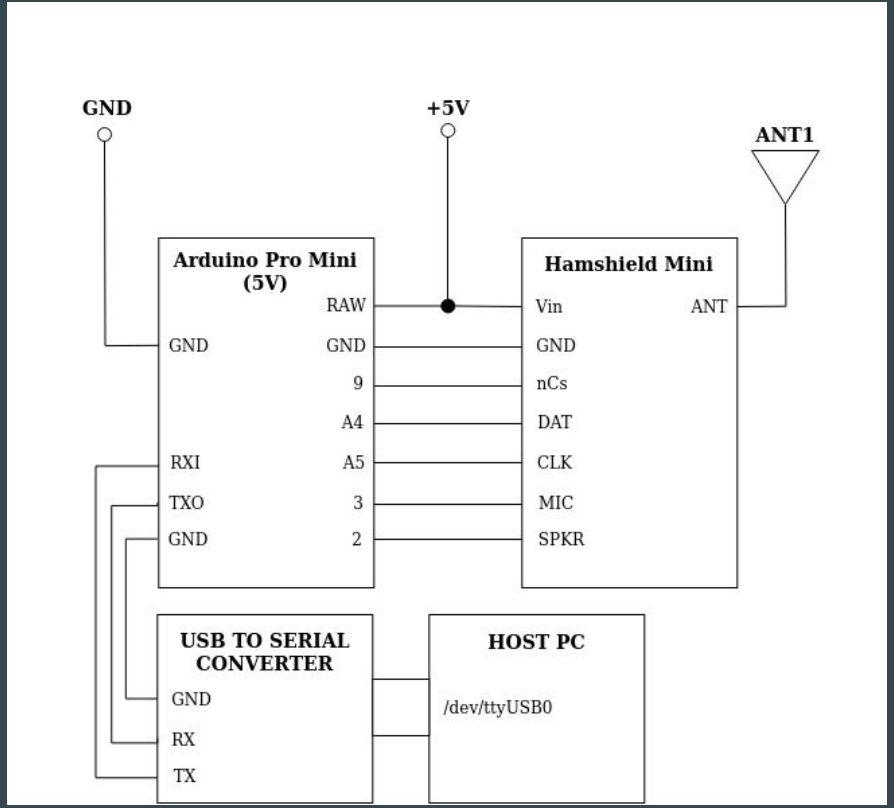
Setup Continued: Connect the Arduino Pro Mini, DSD Tech USB to serial converter and DC power source as shown

Program the Arduino Pro Mini!

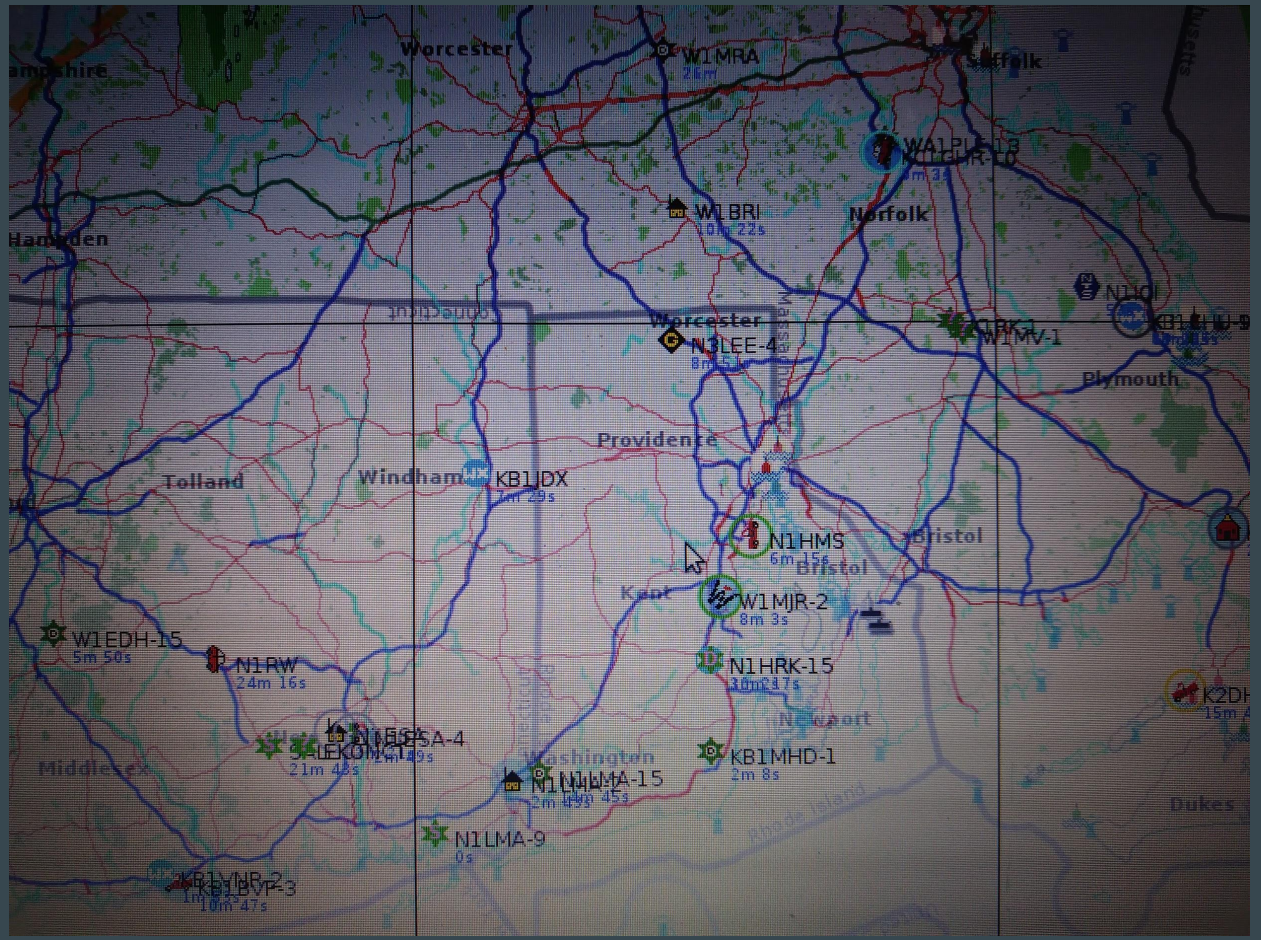
1) Download the HamShield_KissTNC project files from the github project page.
2) Download and install the Arduino IDE following the instructions: https://www.arduino.cc/en/main/software
3) Connect the USB to serial to your PC (you may need to reboot afterwords)
4) Start the Arduino IDE and open the Hamshied_KissTNC.ino project file
5) Confirm the board selected under the tools menu is the Arduino Pro Mini
6) Upload the file. Don't forget to hit the reset button on the arduino after hitting upload. Check your connections and see the Arduino IDE documentation on the web page for further instructions and assistance

KISS TNC Test

The Arduino Pro Mini and the HamShield Mini can be wired as a standalone KISS TNC. The USB to serial converter can be used to connect the Pro Mini to a host PC running a common APRS application to test the functionality of the hardware. I used the YAAC application on my host PC : https://www.ka2ddo.org/ka2ddo/YAAC.html



Our KISS TNC
Test Results:



APRS PACKETS ARE COMING AND GOING!

Prepare to Program the Pocket Beagle

Hardware needed to program the Pocket Beagle:

i) Pocket Beagle

ii) USB A to micro USB cable (supplied with Pocket Beagle)

iii) DSD Tech USB to Serial converter (or equivalent)

iv) 32GB Micro SDCard

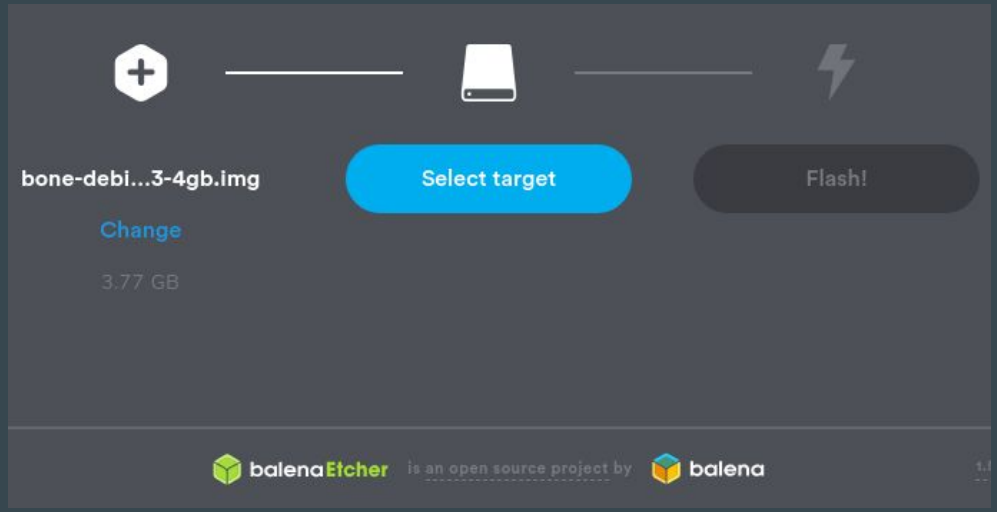
iv) Windows, Mac or Linux PC

- This how-to is written for linux (tested on Ubuntu 18.04) host PC

Download and Flash the OS

Download the Debian 9.9 2019-08-03 4GB SD IoT image from: http://beagleboard.org/latest-images

Follow the instructions here: http://beagleboard.org/getting-started to flash the image to the SD Card using the balenaEtcher program



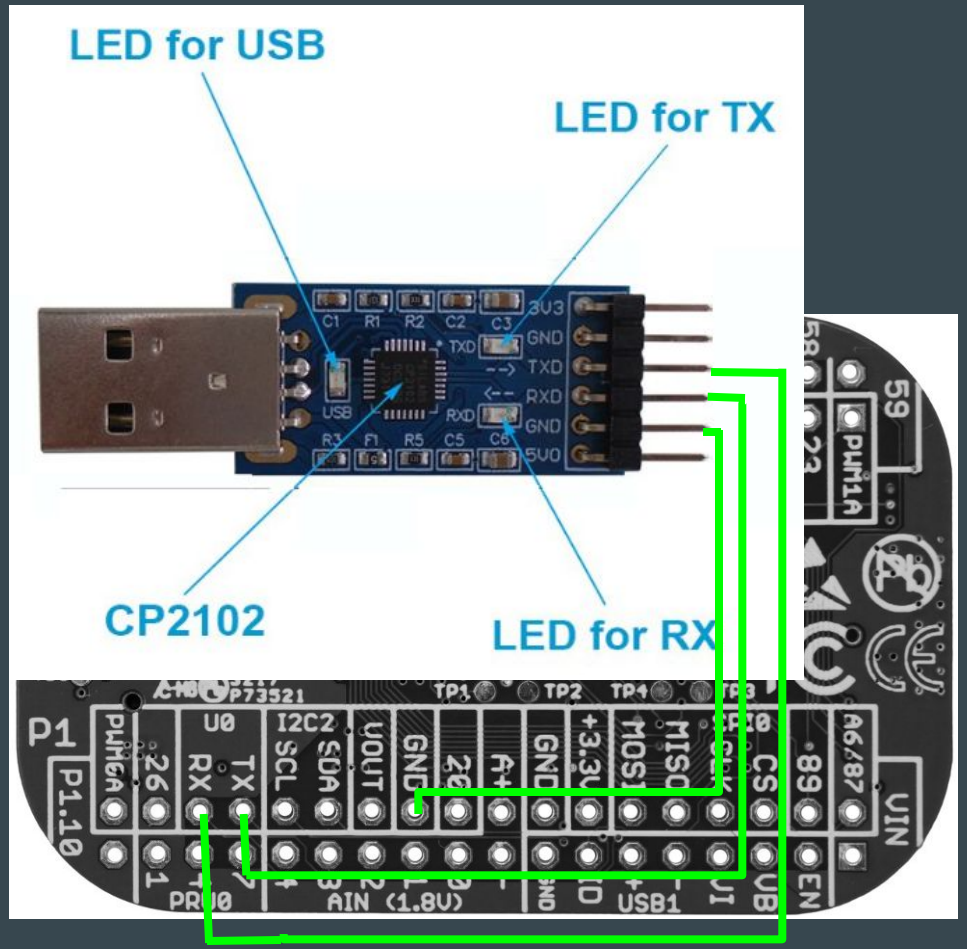
Hardware Setup

Normally the host PC connects to the Pocket Beagle via the micro-USB port.

By default the Pocket Beagle has no network connection. In order to download the required packages for the Rescue Repeater, we need network access to the repositories.

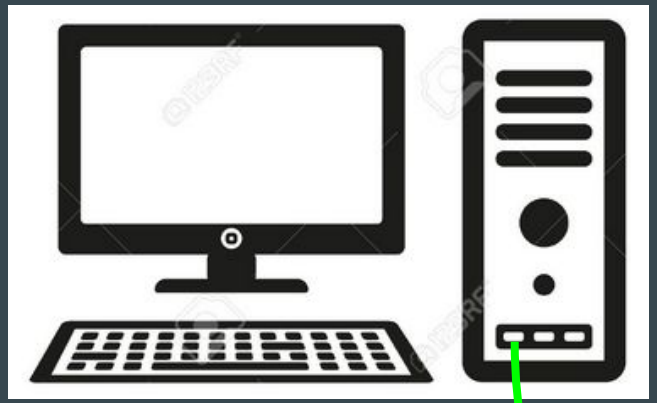
Setup continued: One solution for networking is to use the DSD Tech USB to serial converter to communicate with the Pocket Beagle via U0, and forward IP packets between the Pocket Beagle micro-USB port and the host PC

Begin by connecting the USB to serial converter to U0 as shown



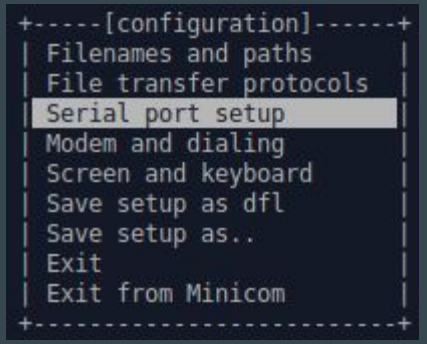
Setup continued: Next, connect Pocket Beagle micro-USB port to the host PC using the supplied cable.

The host will establish a connection to the Pocket Beagle over the USB port. We need to install the minicom package on the host PC to control the Pocket Beagle via U0.



Setup continued: On the host, start minicom from the terminal with “sudo minicom -s”

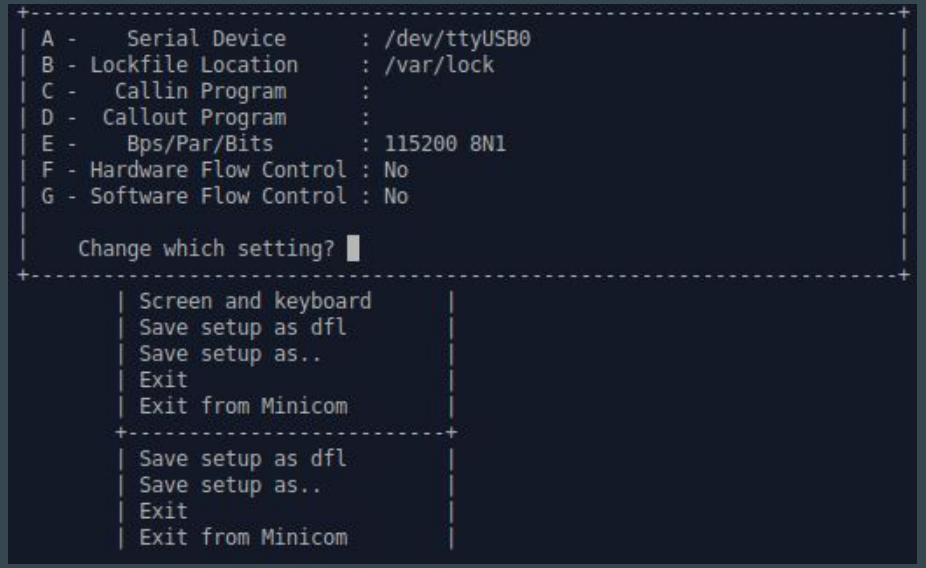
Minicom will start and present the setup menu. Select serial port setup.



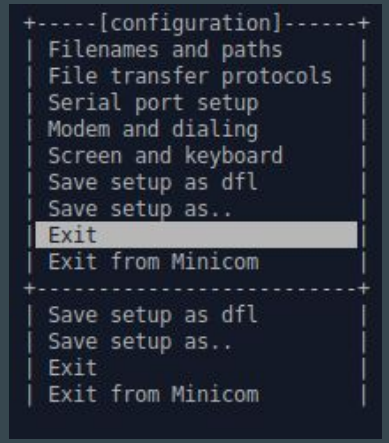
Setup continued: Set the serial device to the location of the USB to serial converter. (option A)

Set the Bps/Par/Bits to 115200 8N1 (option E)

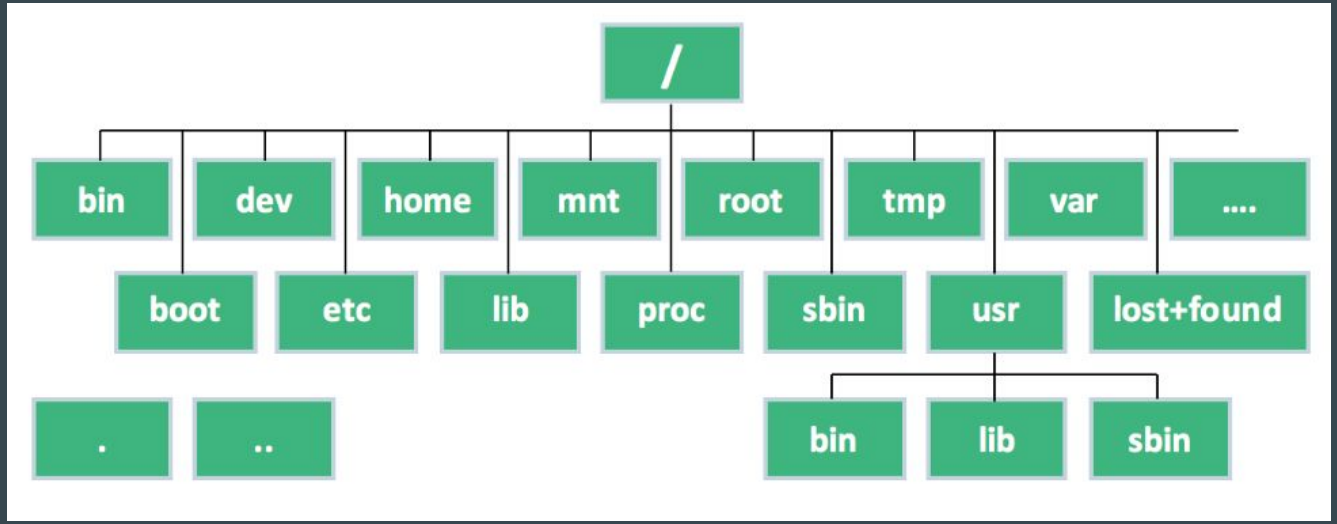
Press enter to return to the main menu



Setup continued: Select exit on the minicom menu to enter the terminal emulator. You may need to hit enter several times to sync up with the Pocket Beagle, but you should be presented with the Pocket Beagle command line



WE NOW HAVE AN ALTERNATE COMMUNICATION INTERFACE (U0) TO THE POCKET BEAGLE. THE POCKET BEAGLE USB IS FREE TO USE FOR PORT FORWARDING



Setup continued: The remaining steps require placing and configuring the Rescue Repeater project scripts within the linux filesystem. Please reference the plentiful linux support documentation, tutorials and forums across the web to supplement the instruction herein

-The project scripts are within the /etc and /home directories

Internet Tethering

There is a tether_host.sh script for the host PC and a thether_client script to support tethering an internet connection from the host PC to the Pocket Beagle.

Place the tether_client script in the Pocket Beagle /home/debian/ directory

> Ensure the script is executable. Use “sudo chmod + tether_client” to set the script permissions to allow execution

Place the tether_host.sh somewhere on the host filesystem. The location is up to the user’s convenience. Ensure it is also executable.

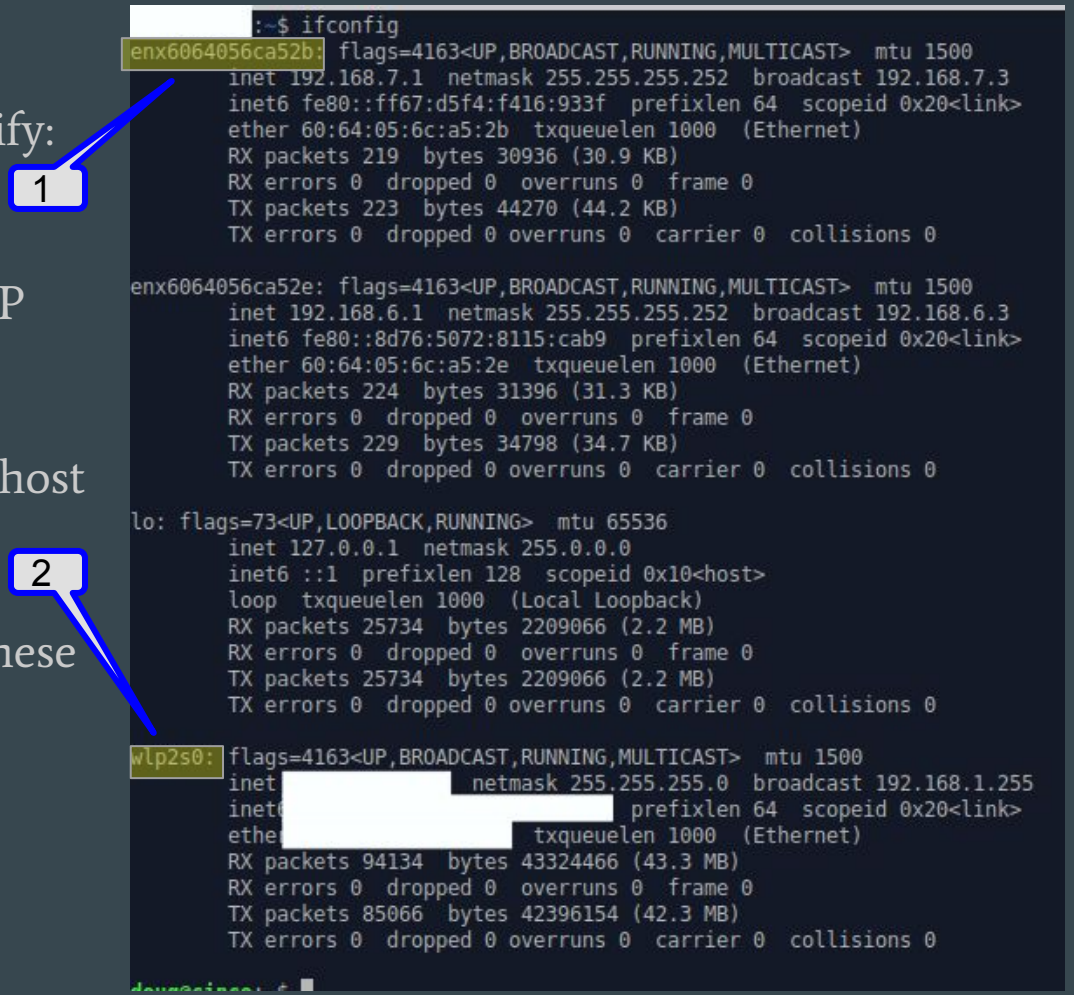
We need to set a few parameters within the tether_host.sh script...

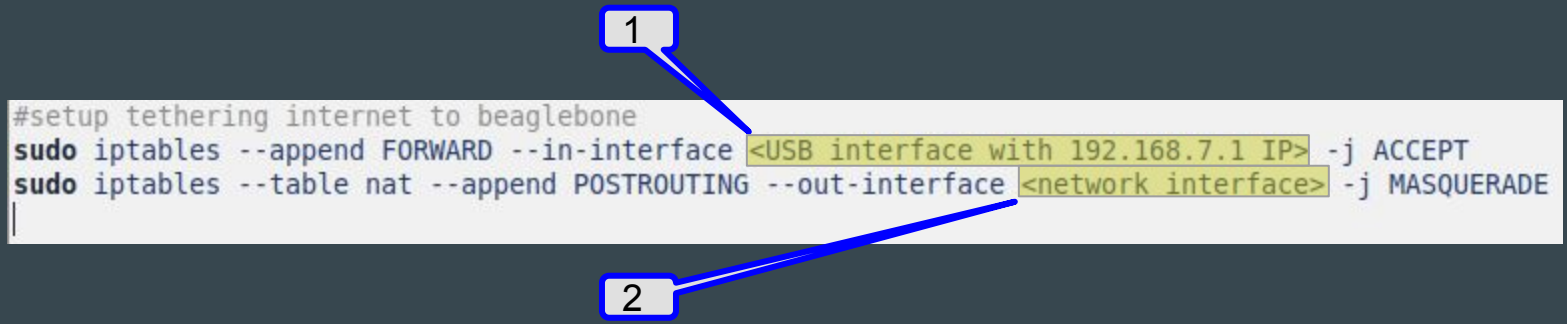
Tethering continued: Within the tether_host.sh script, we need to specify:

1) the host network interface that is connected to the Pocket Beagle (the IP address will be 192.168.7.1)

2)The interface which is used for the host internet connection.

Use the “ifconfig” command to find these interfaces





Tethering continued:
Update the interface definitions in the tether_host.sh script

Tether!

On the host PC, run the tether_host.sh script: “sudo ./tether_host.sh”

On the Pocket Beagle (Via the Minicom terminal), run the tether_client script: “sudo ./tether_client”

The Pocket beagle should now be tethered to the host PC internet connection. Test connectivity by pinging google DNS server: “ping 8.8.8.8”

You should see replies to the ping messages. Terminate the ping test with Ctl+c

One more step...

Tether! continued

The Pocket Beagle still needs to be configured to use a DNS server to resolve URL addresses.

Edit the /etc/resolv.conf file: “sudo nano /etc/resolv.conf”

Add the following line:

“nameserver 8.8.8.8”

Save the changes by writing-out the file with Ctrl+o

Test DNS: “ping www.google.com”

```
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=19.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=22.4 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 19.809/21.128/22.447/1.319 ms
debian@beaglebone:~$ ping www.google.com
PING www.google.com (172.217.6.228) 56(84) bytes of data.
64 bytes from lga25s55-in-f228.1e100.net (172.217.6.228): icmp_seq=1 ttl=55 time
=15.1 ms
64 bytes from lga25s55-in-f228.1e100.net (172.217.6.228): icmp_seq=2 ttl=55 time
=17.3 ms
64 bytes from lga25s55-in-f228.1e100.net (172.217.6.228): icmp_seq=3 ttl=55 time
=19.6 ms
^C
--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 15.137/17.364/19.602/1.829 ms
debian@beaglebone:~$
```

THE POCKET BEAGLE NOW HAS AN INTERNET CONNECTION TO INSTALL THE REQUIRED PACKAGES

Install the Required Packages

Use apt to install the required packages on the Pocket Beagle: “sudo apt install <package>”

- ax25-apps
- ax25-tools
- libax25 (This may have already been installed for the other packages)
- aprx

WE ARE NOW READY TO CREATE/EDIT THE CONFIG FILES

Configure

Add the following scripts from the project files to the /etc/ directory:

- Edit or create a modules file. This file sets what kernel modules will be loaded at boot. Ensure all the modules in the example file are included. The ax25 services and the aprx digipeater require the kernel modules to be loaded, or they won't start correctly during boot.
- dhcpcd-ax0.conf (This file prevents dhcp requests or wpa supplicant authentication on the KISS interface ax0 that gets created by aprx)
- aprx.conf (This is the main configuration file for the aprx APRS digipeater)

The aprx.conf file provided configures aprx to function as an RF only APRS digipeater. Only a few parameters need to be edited...

Configure continued: There are many configuration possibilities, see the aprx documentation for details: https://thelifeofkenneth.com/aprx/

Only 2 parameters require editing for the Rescue Repeater configuration

Enter Callsign and SSID

Enter Location (See the Example on the next line)

```
#
# Simple sample configuration file for the APRS Digipeater using
# the Resue Repeater (HamShield Mini, Arduino Pro Mini & Pocket Beagle)
# This configuration is structured with Apache HTTPD style tags
# which then contain subsystem parameters.
#

#
# For simple case, you need to adjust 4 things:
#   - Mycall parameter
#   - passcode parameter in APRS-IS configuration
#   - Select correct type of interface (ax25-device or serial-device)
#   - Optionally set a beacon telling where this system is
#   - Optionally enable digipeater with or without tx-igate
#


#
#
# Define the parameters in following order:
#   1)  <aprsis>     ** zero or one
#   2)  <logging>    ** zero or one
#   3)  <interface>  ** there can be multiple!
#   4)  <beacon>     ** zero to many
#   5)  <telemetry>  ** zero to many
#   6)  <digipeater> ** zero to many (at most one for each Tx)
#


#
# Global macro for simplified callsign definition:
# Usable for 99+% of cases.
#

mycall MYCALL-n
#
# Global macro for simplified "my location" definition in
# place of explicit "lat nn lon mm" at beacons. Will also
# give "my location" reference for "filter m/100".
#
myloc lat ddmm.mmN lon dddmm.mmW
#Example:4125.00N lon 07130.00W
```

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#

vhf     <MYCALL-n>      9600    255     2       144.390 MHz (1200  bps)
#2       OH2BNS-9        38400   255     7       TNOS/Linux  (38400 bps)
```

Configure continued:

Add or replace the /etc/ax25/axports file:

The provided axports file creates an ax25 port called vhf.

Replace <MYCALL-n> with your callsign.

Configure continued:

Copy the following scripts to /etc/ax25/ (ensure these scripts are executable!)

- ax25-up
- ax25-down

Note on ax25-up and ax25-down scripts: These scripts are used by the /etc/init.d/ax25ifs service (this script also provided with project files) which is a soft dependency for the aprx startup script. The ax25ifs, ax25-up and ax25-down scripts were copied from examples from N9LOY: http://www.sneaky.net/blog/2015/07/08/n9loy-packet-radio/

Initially, the ax25-up script included a kissattach command to bind the ax25 port (vhf in our case) and ax0. This prevented aprx from accessing the vhf port. In our ax25-up script, the kissatach line is commented out and ax25-up ultimately does nothing. ax25-down may be used by aprx for killing kiss interfaces on exit? For dependency sake, the ax25-up, ax25-down and the ax25ifs scripts are included. Confirmation is needed on whether the ax25ifs “should start” dependency is appropriate in the aprx startup service LSB header.

Configure continued:

Copy the following scripts to /etc/init.d/ (ensure they are executable)

- aprx
- ax25ifs

POCKET BEAGLE AND KISS TNC ARE READY! THE RESCUE REPEATER SOFTWARE COMPONENTS WILL AUTOMATICALLY START DURING BOOT - MANUAL CONTROL FROM A HOST PC IS NO LONGER REQUIRED