Technical Brief
AN201 Rev A2

Connecting to the Internet from Linux using the Daisy Cellular Modem

By Adam Hickerson *Raveon Technologies Corp*

Adapted from https://wiki.ubuntu.com/NetworkManager/Hardware/3G/Probing *and* http://elinux.org/RPi_Serial_Connection *for the Daisy Cellular Modem*

Summary

The Daisy-Cellular modem may be used as a drop-in for other Daisy Modems used as part of a DART network. However, the Daisy-Cellular also implements a pass-through mode that allows the modem to operate identical to other cellular modems on the market, functioning as a software-compatible drop-in replacement for such a device.

Most cellular modems implement a set of AT commands and protocols that allow a desktop OS (or advanced microcontroller OS) to connect to the internet as though they were connected through a dial-up landline or even an Ethernet connection. The Daisy-Cellular modem also provides this interface, allowing an OS such as Linux (the focus of this guide) or Windows to connect to the internet seamlessly.

The benefit of connecting is this manner is that it allows the programmer to write most software using high-level interfaces such as TCP Sockets, even testing them on a normal Ethernet or Wi-Fi connection along with the cellular connection.

This guide describes the configuration and steps necessary to connect in this manner.

Setting up the Connection

At its core, the connection will use `pppd` to create a point-to-point connection. There are numerous way to configure how the connection will come up. This guide focuses on using `wvdial` which simplifies much of the process. Verify everything required is available by issuing `sudo apt-get install wvdial`.

Once `wvdial` is installed, it must be configured to connect the Daisy-Cellular and create a PPP connection. This is done by editing /etc/wvdial.conf to include specific dial commands relevant to the Daisy-Cellular as well as the wireless carrier to be used. This guide cannot address all possible wireless carrier configurations, so the script used makes a few assumptions. The comments within the script indicate the assumptions and possible other actions to take.

```
[Dialer cellular]
ISDN = 0
```

```
New PPPD = yes

# Your modem device path.
Modem = /dev/ttyAMA0

Baud = 38400

# If your SIM card has a PIN, comment this line, uncomment the next one, and
# change the PIN shown to your PIN.
Init = ATZ
# Init = ATZ+CPIN="0000"

# By default it is assumed that you can dial out without an APN.
# If this is not the case, comment out the following line and uncomment the
# next and place the APN insead of YOUR_ISP_APN.
Init2 = AT+CGDCONT=1,"IP"
#Init2 = AT+CGDCONT=1,"IP","YOUR_ISP_APN"

# Most services/devices dial with *99# . A few seem to require *99***1#
Phone = *99#

# These often suffice, but your ISP might require different details. They're
# often dummy details used for all users on the ISP, frequently the ISP's
# name, but some ISP's do require you to use a real username and password.
Username = dummy
Password = dummy
```

This script was adapted from https://wiki.ubuntu.com/NetworkManager/Hardware/3G/Probing. Editing the script will require root priveleges. Assuming no other `wvdial` connections are needed, the existing `/etc/wvdial.conf` can be replaced.

Dialing Out

To dial out, simply issue the command `sudo wvdial cellular`. This should handle all the necessary steps to dial out and set up the OS to use the connection.

If successful, the terminal will show the IP addresses and DNS addresses being assigned. To verify the connection, a ping command such as `ping raveon.com` can be issued. Ensure no other network connections are connected (disconnect the Ethernet and any other USB network connection). If the ping succeeds, the Linux computer is online via the Daisy-Cellular modem. Any networking function (a graphical browser, `lynx`, `ssh`, `wget`). Keep in mind that cellular data is being used at this point, so be careful about what actions are taken.

To bring down the connection, Ctrl+C can be used to kill the command in the terminal window, or another method of killing the command may also be used such as `sudo killall wvdial`. True software integrations will use other methods of performing these actions.

The following section shows a successful dial-out being performed for reference.

Appendix A: Successful wvdial Dial-Out

```
~ $ sudo wvdial cellular
--> WvDial: Internet dialer version 1.61
--> Initializing modem.
--> Sending: ATZ
ATZ
OK
--> Sending: AT+CGDCONT=1,"IP"
AT+CGDCONT=1,"IP"
OK
--> Modem initialized.
--> Sending: ATDT*99#
--> Waiting for carrier.
ATDT*99#
CONNECT
--> Carrier detected.  Waiting for prompt.
~[7f]}#@!}!}!} }9}"}&} } } } }#}%B#}%}%}&$B^i}'}"}(}"}!N~
--> PPP negotiation detected.
--> Starting pppd at Sun Sep 27 21:37:51 2015
--> Pid of pppd: 4420
--> Using interface ppp0
--> pppd: [08]��[01]p|�[01]
--> pppd: [08]��[01]p|�[01]
--> pppd: [08]��[01]p|�[01]
--> pppd: [08]��[01]p|�[01]
--> local  IP address 10.84.8.14
--> pppd: [08]��[01]p|�[01]
--> remote IP address 10.84.8.14
--> pppd: [08]��[01]p|�[01]
--> primary   DNS address 172.17.1.101
--> pppd: [08]��[01]p|�[01]
--> secondary DNS address 172.17.1.102
--> pppd: [08]��[01]p|�[01]
```

----- Ctrl-C Issued Here -----

```
^CCaught signal 2:  Attempting to exit gracefully...
--> Terminating on signal 15
--> pppd: [08]��[01]p|�[01]
--> Connect time 0.3 minutes.
--> pppd: [08]��[01]p|�[01]
--> pppd: [08]��[01]p|�[01]
--> Disconnecting at Sun Sep 27 21:38:11 2015
```

Appendix B: Raspberry Pi Signal Connections

The Daisy-Cellular is connected to the UART provided on the GPIO header of the Raspberry Pi. Power must be supplied to both the Pi and the Daisy-Cellular, keeping in mind that the Daisy-Cellular requires a supply capable of sourcing up to a 2A surge (under rare conditions).

Raspberry Pi GPIO	Daisy-Cellular Universal Modem Connector
Pin 8 (GPIO15/TxD)	Pin 5 (RX_IN)
Pin 10 (GPIO16/RxD)	Pin 6 (TX_OUT)

The serial port used will show as `/dev/ttyAMA0` in Linux.

Handshaking lines may increase the performance of the connection, but they are not available in hardwsare on the Raspberry Pi unless the P5 header is present.

Appendix C: Raspberry Pie Configuration of the Serial Port

In any system, it is important that the serial port is not configured for use by any other process, such as a login prompt.

By default, the Raspberry Pi configures the serial port to show boot messages and a login shell. To disable these on a modern installation of Raspbian, the `raspi-config` tool can be used. Start the tool with `sudo raspi-config`. Versions of the tool may have different menu options, so the specific numbers are not included here. Locate the option for Serial and disable shell login on the serial port.

To verify that the command was successful, two locations must be checked.

1. Inspect the contents of /boot/cmdline.txt and ensure there are no references to ttyAMA0
2. Verify that no uncommented references to `ttyAMA0` exist in `/etc/inittab`. This is a longer file, so it is wise to use a command such as `grep ttyAMA0 /etc/inittab` to view only relevant lines. Comment lines in this file are preceeded by a hash mark (`#`). In particular, the following line will likely appear, ensure it is preceeded by the hash mark:

```
#T0:23:respawn:/sbin/getty –L ttyAMA0 115200 vt100
```

The existing processes controlling `ttyAMA0` must be stopped. The simplest way to do this is to simply reboot the system.

See the guide at http://elinux.org/RPi_Serial_Connection for more information on this step.