Application Note

Establishing a Gateway Connection Through PPP

Bailey Looper, Schuyler Patton

ABSTRACT

This document describes the process of establishing a gateway by configuring Point-to-Point Protocol (PPP) on a cellular modem connected to an AM62x TI processor. PPP is a data link layer protocol communication protocol that establishes a direct connection between two network nodes or endpoints. PPP enables a processor running Linux to exchange IP traffic through a modem, which then bridges to a GSM or LTE network.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Use Case

In many applications, a reliable gateway to the internet is necessary to enable communication, monitoring, or control of remote devices. Cellular modems provide an effective design when a traditional wired connection is not feasible, or when wireless (for example, 802.11) coverage is inconsistent. By leveraging PPP, a processor can seamlessly integrate a cellular modem as a network interface.

One significant application is in electric vehicle charging stations in dense urban environments. Building structures often attenuate or block Wi-Fi signals, limiting connectivity for payment processing or grid negotiation. Ethernet can be used in principle, but physical access to wired infrastructure is frequently restricted by security concerns or cost. A cellular modem provides an attractive alternative, offering a secure and dedicated path to the internet.

Building automation is another sector where PPP over cellular is useful. Many installations require coverage in remote corners of a facility where Wi-Fi cannot reach. Cellular connectivity provides a reliable and maintainable means of enabling communication without dependency on wired backhaul (e.g Ethernet or fiber optic cables). PPP is also widely adopted in transportation and logistics, where long-haul trucking and delivery fleets depend on cellular links to maintain reliable communication across wide geographic areas.

2 Platform and Products

2.1 Hardware

The required hardware is a processor, a cellular modem with either a UART or USB connection between them, and a means of console-based development.

This implementation was developed on the TI AM625 processor using the BeaglePlay single-board computer, together with a Quectel BG95 multimode LPWA cellular module. However, the steps discussed in this application note can be used with other TI ARM processors running Linux as the procedures and protocols are processor agnostic.

Connectivity between the processor and modem was achieved through the microBUS click header for UART communication, though USB is also supported in typical designs. Since UART is used here, this implementation is restricted to low bit rate applications. An active Hologram SIM card was inserted into the modem. Development and debugging were performed on a Linux host system connected to the BeaglePlay.

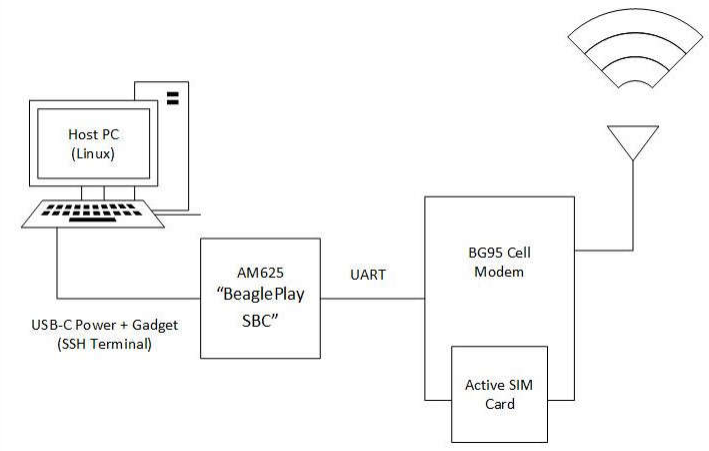

Figure 2-1. Necessary Hardware

The entire implementation can be done with the above connections to the BeaglePlay and cell modem. Once the BeaglePlay board is powered, the USB Type-C port serves dual purposes as both the primary power source and USB gadget, allowing for a development access point. Through this connection, the host PC can establish an SSH session (see Connect WiFi - step 2 ssh debian@192.168.8.1) with the board, creating a convenient environment for configuring the modem and initiating PPP connections without requiring additional interfaces.

Connection to the BG95 from the BeaglePlay is limited to UART when utilizing the mikroBUS click header, but many times USB is used between the processor and the modem.

The following additional setup is helpful for further debugging.

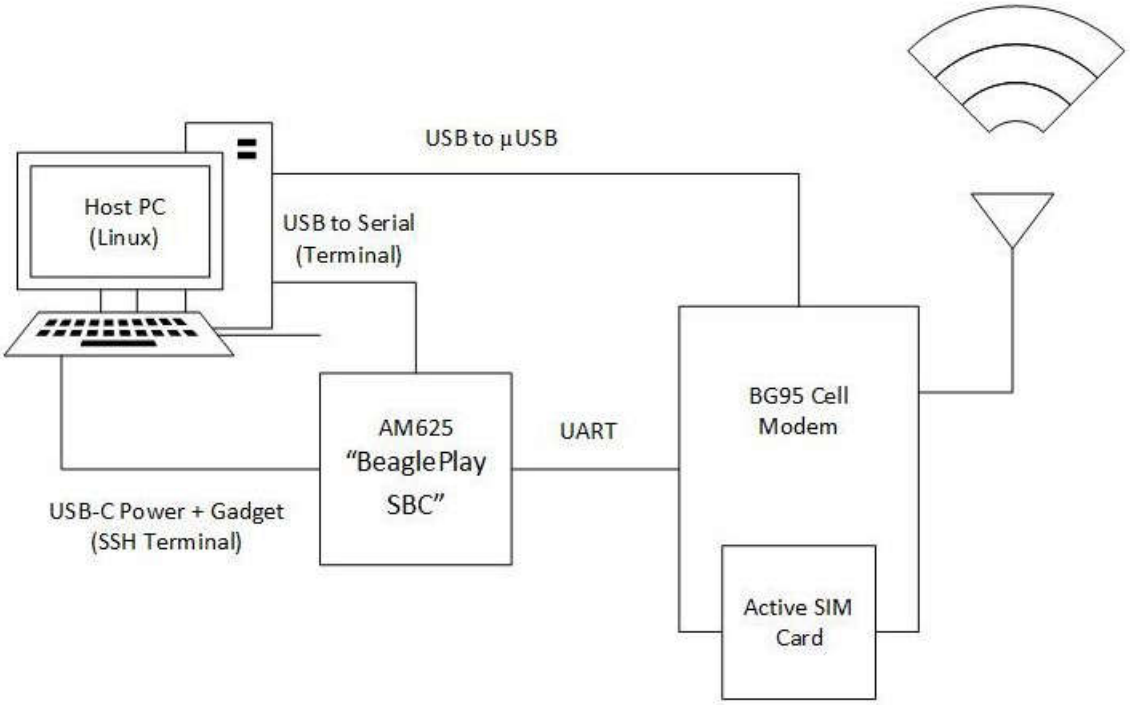

Figure 2-2. Additional Hardware

The UART connection between the BeaglePlay and the host PC allows an interactive boot console and access to boot logs during startup. After a successful boot, the same logs can be accessed over an SSH session using the dmesg command. This connection is optional.

A USB-to-Micro-USB connection between the host PC and the BG95 provides direct access to the modem to verify functionality before attempting to communicate through the BeaglePlay. This connection is optional.

2.2 Software

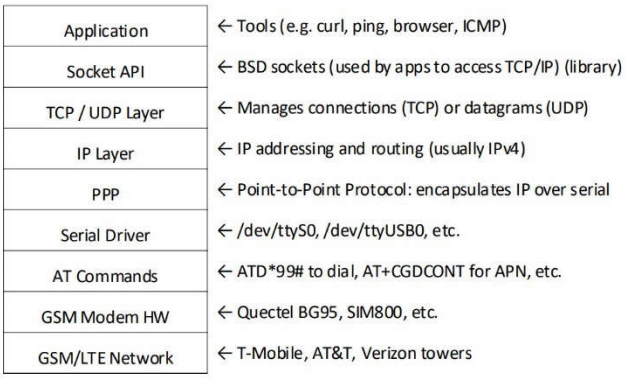

Figure 2-3. Network Stack

Once hardware connectivity is in place, the PPP provides the bridge between the Linux networking stack and the modem. The PPP daemon (pppd) is responsible for establishing and managing a Point-to-Point Protocol session over the serial port of the modem. This negotiates the PPP session, performs authentication if required (such as CHAP or PAP), and sets up the ppp0 network interface on the host. This includes assigning an IP address, configuring routing, and setting DNS servers so that the system can send and receive IP traffic over the connection.

The modem is controlled through AT commands to register with the cellular network and start a data session. AT commands are an industry standard, providing a text-based interface to the functionality of the modem. For example, AT+CGDCONT=1,"IP","<APN>" configures the PDP context with the desired APN, and ATD*99# initiates the actual connection. Once the modem returns CONNECT, the PPP session becomes active and the modem functions as a transparent link-layer pipe, carrying encapsulated IP packets over the cellular network.

Data from the operating system flows through the ppp0 network interface. The IP packets are encapsulated by PPP and transmitted over the serial link to the modem. The modem converts these packets into cellular data and communicates with the GSM or LTE network, sending and receiving data to the wider internet. In this way, the PPP session bridges the IP layers of the host with the cellular network through the serial interface to the modem.

3 Methods

PPP is often used for remote network access and internet connections. In this application, PPP provides a lightweight, serial-based mechanism for transporting IP packets.

The Linux kernel includes PPP support natively. The PPP Daemon (pppd) manages PPP connections. pppd works with the PPP driver of the kernel to set up, maintain, and configure PPP connections, including negotiating IP addresses and handling authentication.

Before PPP can operate, the modem must be configured over a serial port with AT commands. Once PPP is dialed up, the serial port becomes the data path between the processor and the network connection. Commonly used AT commands include:

AT Command	Description
AT+CPIN?	Checks whether the SIM card is unlocked and ready for use
AT+CEREG?	Reports the registration status of the device on the cellular network
AT+COPS?	Queries the currently selected network operator
AT+CSQ	Returns the received signal strength indication (RSSI) and bit error rate (BER)
AT+CGDCONT?	Displays the defined PDP context(s), including APN configuration
ATD*99#	Initiates a PPP data call to establish a network connection
AT+QNWINFO	Reports the active Radio Access Technology (RAT) and frequency band (Quectel Specific)

Once the modem is confirmed to be registered with a network and a data profile is active, PPP can be launched. A common workflow involves verifying the state of the modem through AT commands, confirming signal strength and registration, and then initiating PPP through a peer configuration file that invokes both chat and pppd. The chat utility issues the dial-up AT commands, while pppd latches onto the session once CONNECT is returned. The chat script automates the dial up process, avoiding timing issues do present if dialing were performed manually. To learn more about AT commands and the use for Quectel Modems and beyond, see *Quectel_BG95&BG77&BG600L_Series_AT_Commands_Manual_V2.0.*

4 Overview of Steps to Bring Up the Interface and Establish A Connection

For a step-by-step implementation of establishing a gateway connection through Point-to-Point Protocol, please see *Establishing A Gateway with a Cell Modem* on the beagleboard website.

The beagleboard article documents:

- Powering on the cell modem.
- Establishing a serial connection with the modem.
- AT commands: configuring the modem, checking signal strength, and confirming the modem is registered to a network.
- Linux Commands: automating modem dial up, launching PPP, and testing network connection.

5 Troubleshooting

If the modem does not dial up or connect to the network as expected, debug steps include:

5.1 Power and Initialization First

First, confirm that the modem has sufficient power, is out of reset, and is responding to AT commands. A quick check is to issue AT+CPIN? to make sure the SIM is detected and ready before attempting to dial.

5.2 Verify Network Before PPP

Before launching PPP, verify that the modem has good signal and is properly registered to the network.

Start with AT+CSQ to check signal strength; aim for an RSSI value of 15 or higher for reliable operation. If signal is weak, try moving to a different physical location or adjusting the antenna. For more details, see the section further down on antenna signal strength.

Next, confirm network registration with AT+COPS? and AT+CEREG?. If AT+COPS? continues to return +COPS: 0 even after waiting a minute or two, force a network scan using AT+COPS=? though note that this process can take several minutes, during which the modem does not respond to other AT commands. The current radio access technology (RAT) and frequency band can be checked with AT+QNWINFO to confirm the modem is attached to the expected technology (for example LTE Cat-M1). See *AT+COPS: Modes, parameters, and examples* for more information on the operator selection mode, operator, and radio access technology (RAT) and *+CREG vs. +CGREG vs. +CEREG* for more details on the registration status of the device.

5.3 Test Manual Dial First

Before relying on PPP, attempt to dial manually with ATD*99#. If this does not result in a CONNECT, PPP does not succeed either.

5.4 Timing When Dialing Up

When transitioning from modem configuration to PPP negotiation, timing is critical. The PPP daemon must attach immediately after the modem enters PPP mode. Manually issuing AT commands from a terminal and then switching to launch pppd often fails due to race conditions. Instead, use a chat script to automate the configuration and dialing sequence; this makes sure that the PPP attaches seamlessly.

5.5 Serial Communication

When communicating over UART, use a terminal emulator like tio to verify command/response behavior. To debug commands used in a chat script, open two terminals: one running cat /dev/ttyS0 to listen, and the other sending commands with echo -e "AT\r" > /dev/ttyS0. This setup allows for validation of the command sequence and modem responses before integrating with pppd.

If issues persist, double-check serial port configuration. A known working configuration is:

sudo stty -F /dev/ttyS0 115200 cs8 -cstopb -parenb -crtscts

The configuration can differ depending on the modem or processor. To learn more about serial port configuration, see the Linux manual page. To confirm the configuration: reboot, reconfigure, and retest with cat and echo before retrying PPP. More details on serial configuration are available in the stty man page.

Note that only one program can access a serial port at a time. Do not run cat /dev/ttyS0 or tio concurrently with pppd or chat.

5.6 Chat Script

In the peer file, the line

connect "/usr/sbin/chat -v -f /etc/ppp/peers/bg95-chat"

handles dialing and Packet Data Protocol (PDP) setup. If the AT commands fail, inspect the chat script closely for formatting issues.

For debugging, enable ECHO ON in the script, or run chat manually with the command below to watch the exchange without involving pppd.

chat -v -f script > /dev/ttyS0 < /dev/ttyS0

Note that stray or invisible characters in chat scripts can cause unexpected failures. Always edit in a plain text editor using UNIX line endings. Aborting conditions are strict: any matching abort string immediately stops the script. During experimentation, consider commenting out aborts to avoid premature exits.

5.7 PPP Options

Enable full logging during troubleshooting by adding *debug* and *nodetach* to the peer file. To make sure operation is smooth, configure PPP to automatically set DNS and routing with *usepeerdns* and *defaultroute*.

Negotiation can be streamlined by disabling unused protocols, which reduces delays. Protocols that can be disabled include:

- noipv6
- nobsdcomp
- nodeflate
- novj

A full description of available options can be found in the PPP daemon documentation.

5.8 Signal Strength and Antenna Setup

AT+CSQ reports received signal strength and bit error rate in the format:

+CSQ: <RSSI>,<BER>

RSSI is the Received Signal Strength Indication, valid RSSI values are 0–31, where higher values indicate stronger signals. A value of 99 means no signal. Aim for an RSSI of at least 15. BER is the Bit Error Rate and can generally be ignored in this context.

Physical location of the modem can significantly impact signal. If signal strength remains under 10, try moving the test set up near a window. If that fails to improve signal try moving the test setup close to a cell tower.

Band locking can further improve connection stability by forcing the modem to remain on a preferred band instead of hopping between weaker ones. To check which band the device is on currently, perform the command *AT+QNWINFO* which indicates network information such as the access technology selected, the operator, and the band selected. Commands for locking the band are:

- Lock to a band: AT+QCFG=*BAND*,0,<mask>,0
  - To undo the lock and restore all bands: AT+QCFG=*BAND*,0,0,0
- Reset modem after making changes: AT+CFUN=1,1

If this fails, another method is to factory reset RAT/bands and put the modem in full auto:

- Auto mode (LTE-M + NB-IoT + GSM fallback): AT+QCFG=*nwscanmode*,0,1
- Try the RATs in this order -> Cat-M/LTE-M first, then GSM, then NB-IoT: AT+QCFG=*nwscanseq*,020301,1
- Auto select between CAT-M & NB-IoT: AT+QCFG=*iotopmode*,2,1
- Enable all bands: AT+QCFG=*BAND*,0,0,0

Always make sure that antennas are connected. If RSSI remains low, experiment with antenna placement and orientation.

5.9 Modem Reset

If the modem consistently fails to register to a network or loses responsiveness, a reset can be required. Conditions that warrant a reset include:

- Failure to register after several minutes (AT+COPS? and AT+CEREG? never indicate success).
- Failure to register even after a full network scan (AT+COPS=?).
- Continuous reports of no signal (AT+CSQ = 99,99 or AT+QNWINFO = No Service).
- Loss of responsiveness to AT commands after a PPP session ends.

Reset options include:

- AT command reset: AT+CFUN=1,1
- Hardware reset via GPIO toggling, for example
  - gpioset --mode=exit $(gpiofind MIKROBUS_GPIO1_12)=0
  - gpioset --mode=exit $(gpiofind MIKROBUS_GPIO1_12)=1

6 Summary

Establishing a gateway connection through PPP on a cellular modem provides a simple and robust method for enabling remote devices to communicate with the internet when wired connectivity is impractical. A TI processor such as the AM625 can interface with a modem over UART or USB, and with Linux PPP support, the modem is treated as a standard network interface once the connection is active.

By combining modem initialization through AT commands with session management through pppd, the system creates a transparent bridge from IP-based applications to the cellular network. This approach is widely applicable in markets ranging from electric vehicle charging infrastructure to building automation, where reliable and remotely managed connectivity is essential.

7 References

1. Beagleboard, *Connect Wifi*, forum.
2. Quectel, *Quectel_BG95&BG77&BG600L_Series_AT_Commands_Manual_V2.0*, manual.
3. Beagleboard, *Establishing a Network Gateway with a Cell Modem Through PPP*, blog.
4. Onomondo, *AT+COPS: Modes, parameters, and examples*, blog.
5. Onomondo, *+CREG vs, +CGREG vs. +CEREG*, blog.
6. Linux, *stty(1)- Linux Manual Page*, webpage.
7. Samba, *https://ppp.samba.org/pppd.html*, webpage.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265