

Walkthrough LTE Priv

Para la configuración de LTE Priv, empleamos una USRP B200.

Una vez conectada al PC, para poder ser reconocida necesitamos instalar las dependencias del SW de la radio:

- `sudo apt install libuhd-dev uhd-host`

Una vez conectada la radio, tratamos de encontrarla mediante el comando:

- `sudo uhd_find_devices`

Tratando de hacer esto, surge el problema del directorio de imágenes: por defecto no viene instalado por lo que fue necesario hacer:

- `sudo uhd_images_downloader`

Ahora sí podemos ver la radio conectada:

```
pi@raspberrypi:~ $ sudo uhd_find_devices
[INFO] [UHD] linux; GNU C++ version 12.2.0; Boost_107400; UHD_4.3.0.0+ds1-5
[INFO] [B200] Loading firmware image: /usr/share/uhd/images/usrp_b200_fw.hex...
-----
-- UHD Device 0
-----

Device Address:
  serial: 325D8ED
  name: B200mini
  product: B200mini
  type: b200
```

Comprobamos que es esta sacando las características de la radio con el comando:

- `uhd_usrp_probe`

```
pi@raspberrypi:~ $ sudo uhd_usrp_probe
[INFO] [UHD] linux; GNU C++ version 12.2.0; Boost_107400; UHD_4.3.0.0+ds1-5
[INFO] [B200] Loading firmware image: /usr/share/uhd/images/usrp_b200_fw.hex...
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize radio control...
[INFO] [B200] Initialize radio control...
[INFO] [B200] Performing register checksum test...
[INFO] [B200] Register checksum test passed
[INFO] [B200] Setting master clock rate selection to 'automatic'.
[INFO] [B200] Asking for clock rate 30.000000 MHz...
[INFO] [B200] Actually got clock rate 30.000000 MHz.

Device: B-Series Device
  Name: B200mini
  serial: 325D8ED
  name: B200mini
  product: B200
  revision: 0
  M-Series: B.0
  FPGA Version: 7.0

Time sources: none, internal, external
Clock sources: internal, external
Outputs: not locked

  RX DSP: 0
    Freq range: -8.000 to 8.000 MHz

  RX Board: A
    RX Frontend: A
      Name: FE-TX1
      Antennas: TX/RX
      Sensors: temp, io_locked
      Freq range: 50.000 to 6000.000 MHz
      Gain range PGA: 0.0 to 89.8 step 0.2 dB
      Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
      Connection Type: IQ
      Uses LO offset: No

  RX Codec: A
    Name: B200mini B2 dual ADC
    Gain Elements: None

  TX DSP: 0
    Freq range: -8.000 to 8.000 MHz

  TX Dboard: A
    TX Frontend: A
      Name: FE-TX1
      Antennas: TX/RX
      Sensors: temp, io_locked
      Freq range: 50.000 to 6000.000 MHz
      Gain range PGA: 0.0 to 89.8 step 0.2 dB
      Bandwidth range: 200000.0 to 56000000.0 step 0.0 Hz
      Connection Type: IQ
      Uses LO offset: No

  TX Codec: A
    Name: B200mini TX dual DAC
    Gain Elements: None

pi@raspberrypi:~ $
```

Para configurar la radio programable, necesitamos instalar srsRAN:

- `mkdir build`
- `cd build`
- `sudo cmake ../`
- `sudo make`
- `sudo make install`
- `sudo ldconfig`

Necesita varias dependencias que no venían instaladas por defecto en la Raspberry:

- `sudo apt install -y libmbdts-dev`
- `sudo apt install -y libsctp-dev`
- `sudo apt install -y libconfig++-dev`

Una vez tenemos esto, debemos configurar srsRAN: crearemos un directorio `.config/srsran` en `~/`, copiamos la carpeta `usr/local/etc/srsran/*` en ella y probamos la instalación ejecutando:

- `sudo srsepc` - (inicia core de la radio)
- `sudo srseNB` - (inicia estación base)

Parámetros que configuraremos:

- Frecuencia: 3.6-3.8 GHz (usaremos 3600Mhz)
- Ancho de banda: podríamos hasta 56Mhz pero usaremos 5/10/20Mhz
- Ganancia de transmisión y recepción
- Potencia transmisión
- Clock rate: 5/10/20Mhz

Por tanto: En el EPC: (`enb.conf`)

- `rf_freq = 3600000000`
- `nof_prb = 25`
- `tx_gain = 40`
- `rx_gain = 30`

- `enb_id = 1`
- `tracking_area_code = 1`
- `mme_addr = <IP_MME> # Dirección IP del MME (EPC)`
- `s1c_bind_addr = <IP_eNB> # IP de la eNB`

En la rasp: `sudo ifconfig eth0 <tu_IP_fija> netmask <tu_máscara>` (facilita conexión)

A mayores, `rf.device_name = "uhd"` y `log_level` para debug

A mayores del `epc.conf` y el `enb.conf`, necesitamos otros archivos como el `ue_db.csv` (donde ponemos los parámetros de las sims permitidas), `rr.conf` y `rb.conf`. Estos archivos se referencian con sus paths en los ficheros de configuración para que estación base y radio lo tengan en cuenta.

Una vez tenemos el `epc.conf` y el `enb.conf` creado, ejecutamos en el PC con `sudo srsepc /home/pi/.config/srsran/epc.conf` y `sudo srseNB /home/pi/.config/srsran/enb.conf`

```
pi@raspberrypi:~ $ sudo srsepc /home/pi/.config/srsran/epc.conf

Built in Release mode using commit ec29b0c1f on branch master.

--- Software Radio Systems EPC ---

Reading configuration file /home/pi/.config/srsran/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf001, MNC: 0xff01
SPGW GTP-U Initialized.
```

```

--- exiting ---
pi@raspberrypi:~$ sudo srsepc /home/pi/.config/srsran/epc.conf

Built in Release mode using commit ec29b0c1f on branch master.

--- Software Radio Systems EPC ---

Reading configuration file /home/pi/.config/srsran/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf001, MNC: 0xff01
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:001, MNC:01
S1 Setup Request - TAC 1, B-PLMN 0xf110
S1 Setup Request - Paging DRX v128
Sending S1 Setup Response

```

En la raspberry para dar conectividad para LTE teníamos un script que era pppd call provider para establecer la conexión una vez el pincho estuviera introducido. Para LTE Privado creamos otro que es similar, por lo que correremos con pppd call ltepriv.

Al inicio, como corrimos en la raspberry nos daba errores porque no tenía suficiente memoria. Al ejecutarlo conectada al PC funcionó a la primera. En el proceso de debugging de cual era el error, estuvimos probando ejecutando los comandos de la sim y mandándolos poco a poco, para debuggear. Entre otros, fuimos viendo cosas como

```
echo -e "AT+CPIN=\"6907\"r" > /dev/ttyUSB2
```

```
echo -e "ATr" > /dev/ttyUSB2 | cat /dev/ttyUSB2
```

Cuando lo arreglamos, encontramos un error de autenticación en la radio:

```

--- exiting ---
pi@raspberrypi:~$ sudo srsepc /home/pi/.config/srsran/epc.conf

Built in Release mode using commit ec29b0c1f on branch master.

--- Software Radio Systems EPC ---

Reading configuration file /home/pi/.config/srsran/epc.conf...
HSS Initialized.
MME S11 Initialized
MME GTP-C Initialized
MME Initialized. MCC: 0xf001, MNC: 0xff01
SPGW GTP-U Initialized.
SPGW S11 Initialized.
SP-GW Initialized.
Received S1 Setup Request.
S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b
S1 Setup Request - MCC:001, MNC:01
S1 Setup Request - TAC 1, B-PLMN 0xf110
S1 Setup Request - Paging DRX v128
Sending S1 Setup Response
Initial UE message: LIBLTE_PPE_PSG_TYPE_ATTACH_REQUEST
Received Initial UE message -- Attach Request
Attach request -- IMEI: 860000000000000/3
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 11110000
Attach Request -- PS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 1
PDN Connectivity Request -- ESM Information Transfer requested: false
Downlink NAS: Sending Authentication Request
UE NAS: Authentication Failure
MME code failure
Received UE Context Release Request. MME-UE S1AP Id 1

```

Finalmente, arreglamos un error de configuración que teníamos en la autenticación en el csv y una vez hecho esto, pudimos usar nuestros scripts para darle conectividad entre estación base y raspberry del barco.

Sin embargo, esto tenía un problema todavía: nuestro script se desconectaba de forma semialeatoria, por lo que decidimos cambiarlo por otra forma mucho más simple que debimos usar desde el principio:

- apt-get install network-manager -y
- service network-manager restart
- nmcli connection delete LTEConnection
- nmcli connection add type gsm ifname cdc-wdm0 con-name LTEConnection apn test123
- nmcli connection modify LTEConnection connection.autoconnect yes
- nmcli connection up LTEConnection --ask
- nmcli connection show
- nmcli device status

Una vez hecho esto, podemos ver que tendremos una nueva interfaz de red wwan0 en la raspberry, y en la estación base se nos informará de que se le asignó la ip a la raspberry. Ahora ya tenemos interconectividad entre estación base y raspberry a través del LTE Privado, pero todavía no tenemos conectividad a internet a través de esta interfaz.

Para conseguirla, debemos ir a la estación base y aplicar el script de masquerading que se encuentra dentro de las librerías de srsRAN: `/lib/srsRAN/srsepc/srsepc_if_masq.sh` a la interfaz. Aplicando este script, permitiremos que se haga el setup el NAT dinámico y masquerading en esa interfaz lo que permitirá enrutar paquetes y que los UE puedan acceder al resto de la red fuera de la LTE Privada. A mayores, debimos añadir en nuestro caso que en las tablas de enrutamiento del pc permitiese el hacer forwarding the paquetes de la interfaz en cuestión, pero este paso no debería ser necesario en principio.

Una vez tenemos esto, podemos comprobar la conectividad a internet haciendo un ping a google usando la interfaz `wwan0` desde la raspberry, comprobando así la conectividad final.