

DECT NR+ Python 範例程式教學

版本 1.2.0，2026-10。BANFi-TG-PYS。譜羅科技(股)公司 BANFi Semiconductor 產品；通訊協定堆疊與韌體由 FLSEMI 提供。

1. 這些範例在示範什麼

DECT NR+ 網狀網路 (mesh) 在無線電之間傳送資料。閘道器在它上面再加一層 App：手機或電腦透過藍牙低功耗 (Bluetooth LE) 連到自己的閘道器，經由網狀網路收送資料。接在 USB 序列埠上的開發板 (燒有 DECT NR+ 韌體的 nRF9151 DK、nRF9131 EK) 也可以直接加入。

```
App (手機/PC) <-- Bluetooth LE --> 閘道器 <-- DECT NR+ 網狀網路 --> 閘道器 <-- Bluetooth LE --> App
程式 (PC) <-- USB 序列埠 ----> 開發板 <-- DECT NR+ 網狀網路 --> 任一節點
```

在這之上提供的功能：

- **文字訊息**：傳給某一台閘道器，或傳給所有人 (群組)。
- **群組對講 (按住說話)**：半雙工，一次一人說話。網路的 sink (匯聚節點) 把每一框在每一跳只轉送一次給所有成員。
- **一對一通話**：兩台無線電在通話期間離開網狀網路，改走一個私有網路，結束後再回來。

所有語音編碼 (Opus) 與 App 內加密都在 App 裡完成；閘道器與無線電只負責傳遞看不懂內容的資料框。

1.1.0 起新增 Talk：給 USB 序列埠開發板用、有視窗介面的對講機與訊息工具，涵蓋群組頻道、點對點語音、訊息、找出網路中的無線電，以及加入與離開網路。它的引擎 (`talk_engine.py`) 與網路控制 (`hif_net.py`) 可以不靠視窗單獨使用。**1.2.0 起** Talk 也能像手機 App 一樣透過閘道器的藍牙連線使用。詳見 **Talk 使用與開發手冊** (BANFi-UM-TALK)。

2. 需要準備的東西

閘道器：藍牙範例需要同一網路中至少兩台 DECT NR+ 閘道器，其中一台是網路的 sink。通話範例需要兩台**不是** sink 的閘道器 (總共三台)：sink 不接受通話。

開發板 (選用)：序列埠範例需要一台或多台燒有 DECT NR+ 韌體的開發板。開發板接上 USB 會出現兩個序列埠，主機介面在第二個 (1 Mbaud)。

電腦：已在 macOS (Apple silicon) 上測試：所有範例以 Python 3.14 測試；序列埠範例、編解碼器與加密也以 Python 3.9 測試。Windows 與 Linux 使用相同的函式庫 (bleak、pyserial、libopus)，但尚未經過我們測試。

- macOS、Windows 或 Linux；
- Python 3.9 或更新版本（Talk 視窗需要 Python 3.10 以上與 Tk 8.6 以上）；
- 藍牙低功耗。

Opus 函式庫

- macOS：`brew install opus`
- Debian/Ubuntu：`sudo apt install libopus0`
- Windows：把 `opus.dll` 放在 PATH 中

Python 套件：`python3 -m pip install -r requirements.txt`（bleak、pyserial、cryptography；Talk 的麥克風與喇叭另需 `sounddevice`）。

語音輸入：一個語音 WAV 檔，16 kHz、單聲道、16 位元。其他格式可用以下指令轉換：`ffmpeg -i in.mp3 -ac 1 -ar 16000 -sample_fmt s16 speech_16k.wav`

在 macOS 上，請從「終端機」（或其他有藍牙權限的 App）執行藍牙範例。

3. 群組金鑰

App 內加密使用每個群組一把 16 位元組的金鑰。只有 App 持有這把金鑰；閘道器、relay 與 sink 都沒有。

```
cd examples
python3 make_group_key.py group_key.json
```

請透過你信任的管道，把同一個檔案交給每位成員的 App。每個範例都用 `--key` 指定它。不加 `--key` 時，訊息與語音以明文傳送；網狀網路本身的連線加密仍會保護每一跳，但 relay 與 sink 能讀到內容。

群組可以在不中斷通訊的情況下更換金鑰。每把金鑰帶有世代編號（0–7）；App 以新世代傳送，同時保留上一個世代用來接收。

4. 閘道器配對

閘道器的藍牙名稱是 `BFi53-` 加上它的 ID，例如 `BFi53-1a2b3c4d`。

1. 在閘道器上按一下 **Button 1**，LED 呈藍色呼吸燈：配對視窗已開啟。
2. 執行任一藍牙範例並指定 `--gw BFi53-...`，作業系統會進行配對（LE Secure Connections）。
3. 之後閘道器會自動重新連到這台電腦。已配對的閘道器廣播時不帶名稱，因此程式會記住第一次連線時的位址（`~/config/flsemi_samples/known_gateways.json`），下次用位址找到它。

請注意：

- **一次一個連線：**每台閘道器同時只服務一個藍牙連線，請先關閉其他占用它的 App。
- **按住 Button 1 三秒會清除所有配對：**之後如果電腦無法重新連線，請在作業系統的藍牙設定中移除該閘道器，再重新配對。

5. 範例程式

每個範例都可以用 `--help` 顯示用法。

5.1 文字訊息：`ble_message.py`

```
# 在連到閘道器 B 的電腦上
python3 ble_message.py --gw BFi53-5e6f7a8b --key group_key.json listen 120
# 在連到閘道器 A 的電腦上
python3 ble_message.py --gw BFi53-1a2b3c4d --key group_key.json send --to all "meet at the gate"
```

B 端預期輸出：`from 0x1a2b3c4d: meet at the gate`。

目的地：`--to all` 送到網路中每一台閘道器；`--to 0x...` 只送到某一台。

大小：一則訊息最多 200 位元組；加密增加 16 位元組，文字可用 184 位元組。

接收端顯示：

- 未加密送出的訊息顯示 `[in clear]`；
- 以接收端沒有的金鑰封裝的訊息顯示 `[encrypted, cannot open]`。

5.2 群組對講：`ble_group_voice.py`

```
python3 ble_group_voice.py --talk BFi53-5e6f7a8b --listen BFi53-1a2b3c4d \
    --key group_key.json --wav speech_16k.wav --out heard.wav
```

一台電腦同時連兩台閘道器，扮演兩位成員：

1. 兩者都加入群組。
2. 一台以即時速度播出 WAV：Opus 12 kbit/s、每框 40 ms。
3. 另一台把收到的內容錄成 `heard.wav`，遺失的框會自動補齊。

最後一行摘要顯示送出、收到與補齊的框數。

封包大小上限：群組語音框必須能放進 sink 的一個廣播資源時段，因此範例會限制每個 Opus 封包的大小；加密後的語音框不超過 76 位元組。你自己的 App 也請保留這個上限。

5.3 一對一通話：`ble_call.py`

```
python3 ble_call.py --caller BFi53-5e6f7a8b --callee BFi53-1a2b3c4d \
    --key group_key.json --wav speech_16k.wav --out heard_call.wav
```

使用 `--key` 時的通話建立流程：

1. 撥號方的 App 產生一把通話金鑰，並以群組金鑰封裝。

2. 它的閘道器只把封裝後的內容送過網狀網路。
3. 接聽方的 App 打開封裝，接聽時把金鑰交給自己的閘道器。
4. 兩台無線電接著移到一個私有網路。

建立通話需要幾秒鐘；掛斷後兩台無線電都回到網狀網路。

sink 不接受通話：通話會讓兩台無線電暫時離開網狀網路，而網路不能失去 sink。在 sink 上發起的通話會立即被拒（錯誤 -1）；打給 sink 的通話會以「the network's sink takes no calls」結束。hello 回應會告訴你的 App 它的閘道器是不是 sink（flags 位元 0），App 可以據此在 sink 上隱藏通話按鈕。

5.4 開發板之間廣播：uart_broadcast.py

```
python3 uart_broadcast.py --from /dev/cu.usbmodemAAAA3 --to /dev/cu.usbmodemBBBB3  
/dev/cu.usbmodemCCCC3
```

第一塊板子廣播五個資料封包，其他列出的板子各自回報收到幾個。Windows 上序列埠名稱為 COM...。

5.5 從開發板傳訊息：uart_message.py

格式與閘道器 App 相同，所以開發板和手機可以互傳訊息：

```
python3 uart_message.py --port /dev/cu.usbmodemBBBB3 --key group_key.json listen 60  
python3 uart_message.py --port /dev/cu.usbmodemAAAA3 --key group_key.json send --to all  
"board here"
```

5.6 開發板加入群組對講：uart_group_voice.py

```
python3 uart_group_voice.py --port /dev/cu.usbmodemAAAA3 --key group_key.json listen 30  
python3 uart_group_voice.py --port /dev/cu.usbmodemAAAA3 --key group_key.json talk  
speech_16k.wav
```

開發板每 20 秒向 sink 報到一次，之後就像手機一樣收聽群組。收聽時會錄成 heard_group_uart.wav。

5.7 兩塊開發板之間的語音：uart_p2p_voice.py

```
python3 uart_p2p_voice.py --talk /dev/cu.usbmodemAAAA3 --listen /dev/cu.usbmodemBBBB3 \  
--key group_key.json --wav speech_16k.wav
```

語音框經由網狀網路送到另一塊板子的節點 ID。

點對點語音只用在相鄰（一跳）的板子之間。每秒 25 框的路由單播串流在超過一跳時跟不上：在我們的測試網路上，兩跳只送達約三分之一的框。跨多跳的語音請使用群組對講（sink 每跳只轉送一次），或一對一通話（兩台無線電改走直接連線）。

5.8 有視窗的 Talk：apps/nrplus_talk.py

```
cd apps
python3 nrplus_talk.py
```

把上面的序列埠範例整合成一個有視窗的程式：群組頻道、點對點語音、訊息、成員清單，以及加入、離開與重新加入網路。當它的無線電是網路的 sink 時，Talk 也負責群組頻道。也能透過閘道器的藍牙連線使用。需要 Tk 8.6 以上；要使用麥克風與喇叭需要 `sounddevice`。詳見 Talk 使用與開發手冊。

6. 測試結果

以下數據於 2026-10-05 以本套件在測試網路上量得：Band 9、九台無線電、五跳、兩台閘道器，並以一 Mac 透過藍牙低功耗代替手機。這些是實驗室數據，不是保證值。

範例	結果
<code>ble_message.py</code>	B → A：加密群組訊息、加密直送訊息與明文訊息，三者都正確收到並顯示
<code>ble_group_voice.py</code>	加密，B → A，一跳：301 框收到 301 框，無補齊
<code>ble_call.py</code>	加密邀請：7.5 秒接通，之後 301 框收到 301 框
<code>uart_broadcast.py</code>	從一塊板子：距離 3、4、5 跳的節點各收到 5/5
<code>uart_message.py</code>	加密群組訊息在 5 跳外的板子上成功開啟；板子送出的加密訊息在手機 App（ <code>ble_message.py</code> ）成功開啟
<code>uart_group_voice.py</code>	一塊板子說話、另一塊在 sink 下方 4 跳收聽：302 框收到 275 框
<code>uart_p2p_voice.py</code>	一跳：300/301 與 297/301；兩跳：95/301（見 5.7）

7. 開發你自己的 App

7.1 閘道器連線（Bluetooth LE）

- **服務與特徵值：** – 服務 `f1e5e000-7c1a-4b9e-9b1d-46a0b6f9d300`； – 請求寫入 `...e001`，使用 write without response； – 訂閱 `...e002` 的通知。
- **配對：**訂閱加密特徵值 `da2e7828-fbce-4e01-ae9e-261174997c48` 會讓作業系統進行配對。
- **封包格式：**雙向皆為 `0xF7 | 類型 | seq | 長度 u16 LE | 內容`。請求的回應帶相同的 `seq`；來自網狀網路的事件 `seq` 為 0。

類型	方向	內容
0x01 hello	送往閘道器	— → 0x81 info：協定版本、flags（位元 0 = sink）、ID u32、韌體文字
0x02 send	送往閘道器	目的地 u32 + 資料 → 0x82 result（0 = 已排入佇列）
0x03 who	送往閘道器	— → 0x83 成員清單
0x10 call start	送往閘道器	被叫方 u32、載波 u16（0 = 自動）[、通話金鑰 16、封裝 33]
0x11 call accept	送往閘道器	[通話金鑰 16]
0x12 reject / 0x13 hang up	送往閘道器	—
0x16 call voice	送往閘道器	語音框（除非被拒，否則無回應）
0x18 group join	送往閘道器	1 = 加入，0 = 離開
0x19 group voice	送往閘道器	語音框（除非被拒，否則無回應）
0x90 message	來自閘道器	來源 u32 + 資料
0x91 call state	來自閘道器	狀態、通話 ID u32、對方 u32、info i16
0x93 call voice	來自閘道器	語音框
0x94 group voice	來自閘道器	發話者 u32 + 語音框
0x95 floor	來自閘道器	發話權持有者 u32、是否被拒 u8
0x96 call invite	來自閘道器	通話 ID u32、撥號方 u32、封裝（空 = 未封裝）
0xFF error	來自閘道器	請求類型、錯誤 i16

數值格式：連線上所有多位元組數值皆為小端序（little-endian），內容中註明 BE 者除外。

目的地 ID：0xFFFFFFFF 是所有人，0 是 sink。來自 sink 的訊息，來源顯示為 0xFFFFFFFFE。

7.2 語音框

flags u8（位元 0 第一框，位元 1 最後一框）| seq u16 LE | codec u8（1 = 0pus）| 0pus 封包

- Opus 使用 16 kHz 單聲道、12 kbit/s、40 ms 一框、VOIP 模式。
- 以一個設定「最後一框」位元的空框結束一段通話。
- sink 只讀 flags 這個位元組，用來管理發話權。

7.3 App 內加密

作法是 AES-128-CTR，加上以明文計算、並與明文一起加密的 5 位元組 CMAC 完整性碼。每把子金鑰都是 CMAC(群組金鑰, 標籤)。確切的格式與標籤請見 [flsemi_samples/app_crypto.py](#)。

語音框

- flags、seq 與 codec 三個位元組保持明文；codec 位元組的位元 7 設為 1，位元 4–6 放金鑰世代。
- 接著是 4 位元組的封包計數器，然後是密文。
- IV 由發話者 ID、群組 ID 與計數器組成。

訊息

- 0xE2 | 1 | 世代 | 發送者 ID | 計數器 | 密文。
- 發送者 ID 放在標頭中，因為 sink 的廣播不帶來源位址。只要網路有回報來源，接收端就必須與它比對。

通話邀請的封裝

- 世代 | nonce 8 | 加密後的通話金鑰 | 標籤 8。
- 標籤綁定撥號方與被叫方的 ID。

封包計數器

- 以 10 ms 時鐘刻度計數，永不倒退。請保持這個做法，App 重新啟動後才不會被誤判為重送。
- 接收端對每個發送者保留最近 1024 個計數器的視窗。

7.4 移植建議

Bluetooth LE

- iOS/macOS：CoreBluetooth。
- Android：BluetoothGatt；請要求 247 以上的 MTU。
- Windows：WinRT GattSession。
- 寫入請求以 MTU - 3 位元組分段。

Opus

- iOS/Android：連結 libopus。
- 網頁：WebCodecs，或 libopus 的 WASM 版本。

AES-CTR 與 AES-CMAC

- iOS：AES 用 CommonCrypto。CryptoKit 沒有 CMAC，請依 RFC 4493 以 AES-ECB 實作 CMAC。
- Android：API 31 以上用 javax.crypto 的「AESCMAC」Mac，或使用 Bouncy Castle。

音訊時序：以即時速度送出語音，每 40 ms 一框；播放端使用約 3 框的小型抖動緩衝。

8. 限制

- 群組一次一人說話：第二位發話者會收到 floor 事件，且不會被轉送。
- 每台閘道器一個藍牙連線。
- 訊息大小：每則 200 位元組（加密時文字 184 位元組）。
- 群組語音框大小：必須在封包上限之內（見 5.2）。
- 開發板之間的點對點語音：僅限一跳（見 5.7）。
- 通訊距離與節點數量：本套件不提供這兩項數據。你的佈建規劃請洽 BANFi 技術窗口。

9. 授權

原始碼依 FLSEMI 5-Clause License 提供（見 LICENSE）。你可以修改、整合進商業產品並再散布，但僅限用於內含 FLSEMI 或 BANFi DECT NR+ 裝置的產品。請保留原始碼中的著作權聲明。

© 2026 譜羅科技(股)公司 BANFi Semiconductor Co., Ltd. 與 FL Semiconductor LLC 共同擁有本文件著作權。問題請洽 BANFi 技術窗口，banfiww.com。