RFIDUSB9 Python SDK

RFIDUSB9 UHF RFID 讀寫器的 Python SDK，讓你用幾行程式
就能盤點、讀寫 EPC/USER/TID、寫流水號，大幅縮短開發時間。

- 裝置：WenShing UHF RFID Reader，USB **HID**，VID `2202` / PID `1007`
- 介面：純 Python，跨平台（Windows 實測）

安裝

```
pip install hidapi          # 唯一相依
# 把 rfidusb9/ 資料夾放到你的專案旁，或：
pip install -e .            # 在 sdk/ 目錄執行，安裝成套件
```

5 行上手

```
from rfidusb9 import RFIDReader, BANK_USER

r = RFIDReader()
r.connect()                                  # 自動偵測裝置
for epc, rssi in r.inventory(0.8):           # 盤點
    print(epc, rssi, "dBm")
r.write_and_verify(bytes.fromhex(epc), BANK_USER, 0, b'\x12\x34')  # 寫+驗證
r.close()
```

範例程式在 `examples/`（01 連線 → 06 多卡挑最近），直接執行即可。

API 參考

連線

方法	說明
`RFIDReader(vid=None, pid=None)`	建立物件；不給則自動偵測
`.connect()`	開啟裝置；成功回 True，失敗丟 `RFIDError`
`.close()`	關閉
`.is_open()`	是否已連線
`list_devices()`	回 `[(vid, pid, name), ...]`
`.test()`	連線測試，回 bool
`.set_power(dbm)`	設定功率 17~26 dBm，回實際值

盤點

方法	回傳
`.inventory(duration=0.6)`	`[(epc_hex, rssi_dBm), ...]`（單次，自動 start→stop）
`.inventory_epcs(duration)`	`[epc_hex, ...]`（只要 EPC 時用）
`.start_inventory()`	進入連續串流（送一次）
`.poll_inventory(max_ms=80)`	收串流，回 `[(epc, rssi), ...]`
`.retrigger_inventory()`	連續盤點 keepalive
`.stop()`	停止盤點

> RSSI 為 dBm 負值，**越大（越接近 0）= 距離越近**。

讀取

```
ok, data, status = r.read_bank(epc_bytes, bank, word_addr, word_count, access_pwd=(0,0,0,0))
```

- `bank`：`BANK_RESERVED(0)` / `BANK_EPC(1)` / `BANK_TID(2)` / `BANK_USER(3)`
- `word_addr`：起始字位址（1 word = 2 bytes）。EPC 資料從 word 2 開始
- `word_count`：要讀幾個字（>=1，不可為 0）
- 回 `(ok: bool, data: bytes, status: str)`

寫入（皆含自動讀回驗證）

```
res = r.write_and_verify(epc_bytes, bank, word_addr, data_bytes, access_pwd=(0,0,0,0))
res = r.write_epc_and_verify(select_epc, new_epc_bytes, access_pwd=(0,0,0,0))
```

- `data_bytes` 長度需為偶數（word 對齊）
- **寫 EPC 區請用** `write_epc_and_verify`：會自動更新 PC 長度字， 寫 10-byte EPC 讀回即 10 bytes（`write_and_verify` 對 EPC@word2 也會自動轉呼叫它）
- 回傳 dict：

```
{
  "ok": bool, "verified": bool,
  "write_status": str, "readback": "讀回hex", "expected": "預期hex",
  "message": "中文結果說明",
}
```

- 底層單次操作（不驗證）：`.write_bank(...)` 回 `(ok, status)`

流水號

```
from rfidusb9 import SerialGenerator
gen = SerialGenerator(prefix_hex="E2000017", start=1, digits=8, step=1, base=16)
hex_str, display = gen.next()     # 取號並遞增
```

常數

- `BANK_RESERVED / BANK_EPC / BANK_TID / BANK_USER`
- `BANK_NAMES`：代碼→名稱
- `READ_STATUS` / `WRITE_STATUS`：狀態碼→中文說明

- `DEFAULT_VID = 0x2202` / `DEFAULT_PID = 0x1007`

實務注意事項

1. **多卡同場**：用指定 EPC 去 `read_bank` / `write_*` 即可鎖定單張； 先 `inventory()` 依 RSSI 挑最近那張（見 `examples/06_pick_nearest.py`）。
2. **寫入比讀取耗能**：寫不進時把 Tag 靠近天線、功率設 26 dBm。
3. **回應偶爾遺失**：`write_and_verify` 已內建「不論寫入回應一律讀回比對 + 重試」，以讀回是否相符為準。
4. **狀態碼**：讀成功 status=成功(底層 0xDE)；`0x09` 表該 Tag 無此區/晶片無 USER。
5. **連續密集操作**：指令間建議留些間隔；本 SDK 已內建必要的安定延遲與重試。