用 Python 透過 Ethernet 遠端控制 EtherCAT 馬達：QEC-M-01 + Moon's 驅動器

開發指南

版本 1.2 | 2026 年 5 月
昭營科技股份有限公司

修訂記錄

版本	日期	變更說明
1.2	2026 年 5 月	初次對外發行。
1.0	2026 年 1 月	內部工程版本。

目錄

1. 概覽

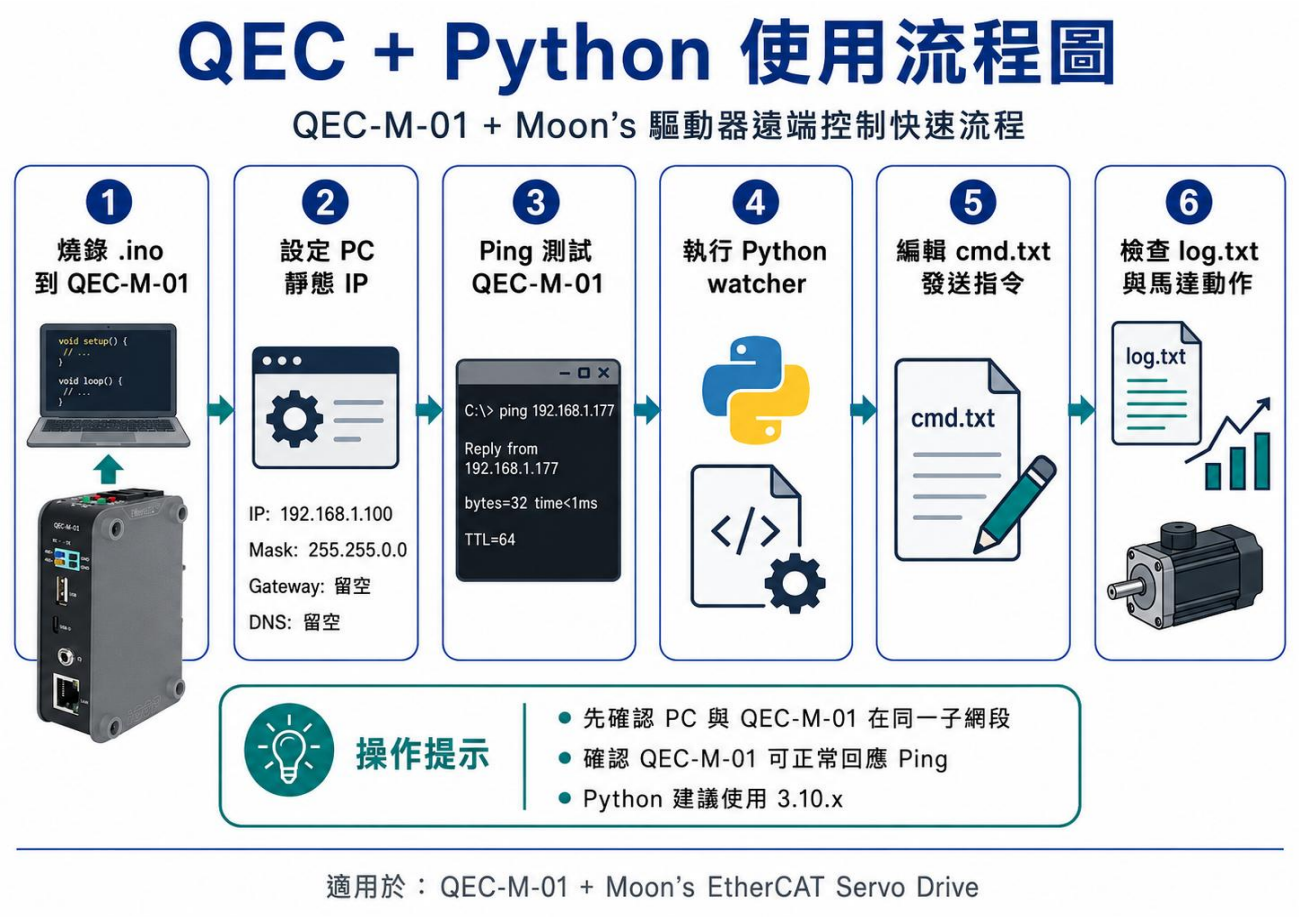
這份指南展示一個最小可用的遠端馬達控制架構：上位 PC 用 Python 透過標準 Ethernet IP，對執行 86Duino 程式的 QEC-M-01 送出控制指令，同時持續記錄馬達位置回報。

整套系統由兩個程式組成：

- Telnet_Ethercat_MOONS_Log.ino：燒入 QEC-M-01，負責 EtherCAT 馬達控制與 Telnet 伺服器
- watch_dir_telnet_log.py：在 Windows PC 上執行，監看指令檔並透過 Telnet 送出指令、記錄 Log

這個架構不需要任何特殊 SDK 或工業軟體，只需要標準 Python 3 和一條網路線。

本指南適合需要將 QEC-M-01 整合至上位 PC、Python 測試程式、自動化測試流程或客製化 HMI 的使用者。PC 端不需要直接處理 EtherCAT，只需透過 Ethernet 傳送簡單控制指令。



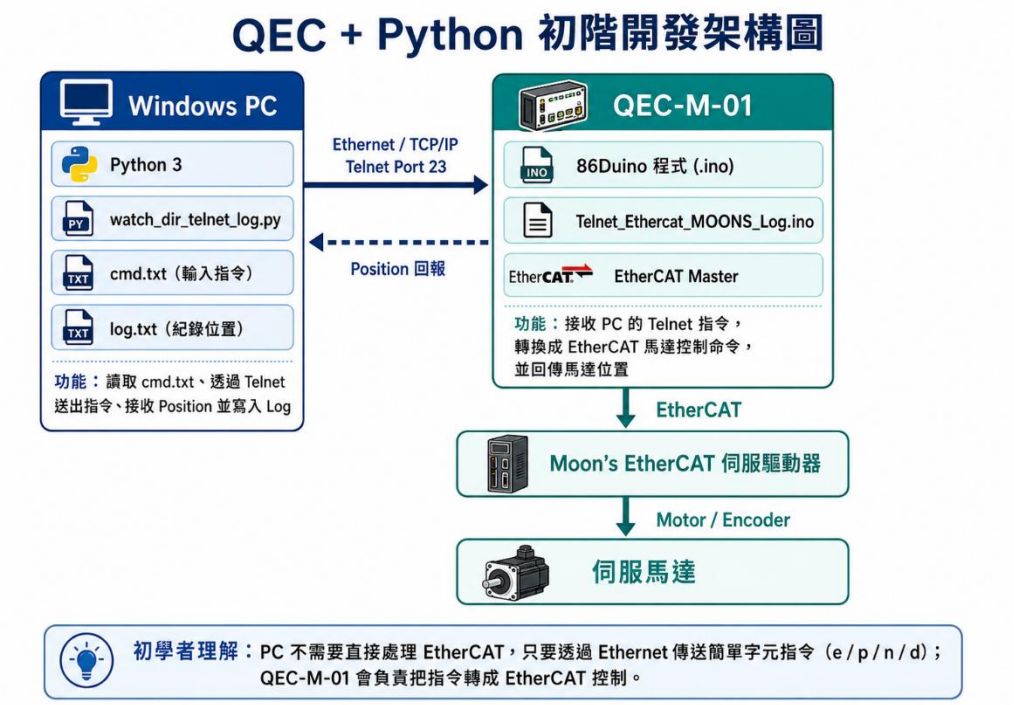
1.1 系統架構

關於此系統架構所需要的硬體及軟體。

1.1.1 硬體需求

元件	型號	說明
EtherCAT 控制器	QEC-M-01	執行 86Duino 程式，內建 EtherCAT Master
EtherCAT 伺服驅動器	Moon's（CiA402 相容）	連接至 QEC-M-01 EtherCAT 端口
上位 PC	Windows（任意版本）	執行 Python 3 控制程式
網路	同一區域網路	PC 與 QEC-M-01 在同一子網段

1.1.2 通訊架構



QEC-M-01 扮演雙重角色：對下透過 EtherCAT 控制伺服驅動器，對上作為 Telnet 伺服器接收 PC 指令。PC 端不需要任何 EtherCAT 知識，只需要能建立 TCP 連線。

> **【注意】**
> 本範例使用 Telnet 作為教學與驗證用途，適合在封閉式區域網路或實驗環境中使用。若用於正式設備或跨網段系統，建議改用具備權限控管、封包格式檢查或加密機制的 TCP/Socket 通訊架構。

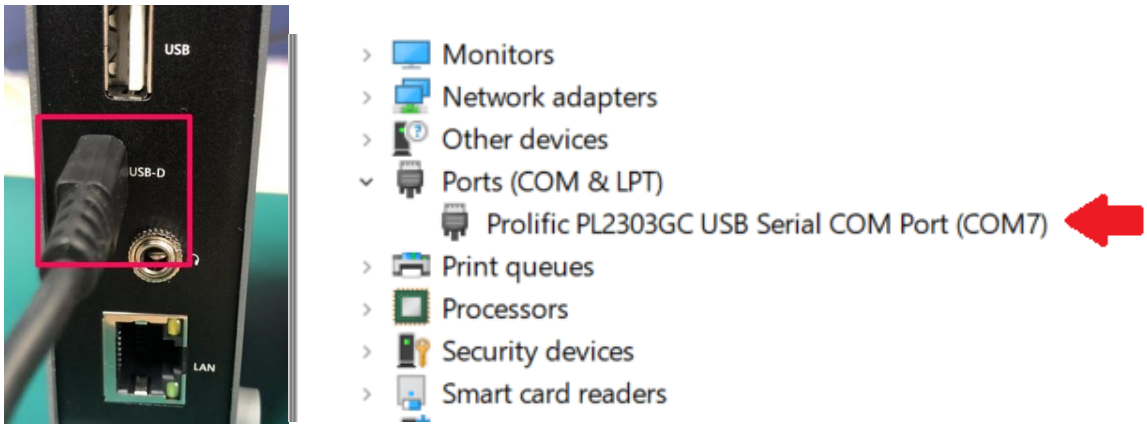
2. 設定 QEC-M-01(燒入 .ino)

將 Telnet_Ethercat_MOONS_Log.ino 燒入 QEC-M-01 中。

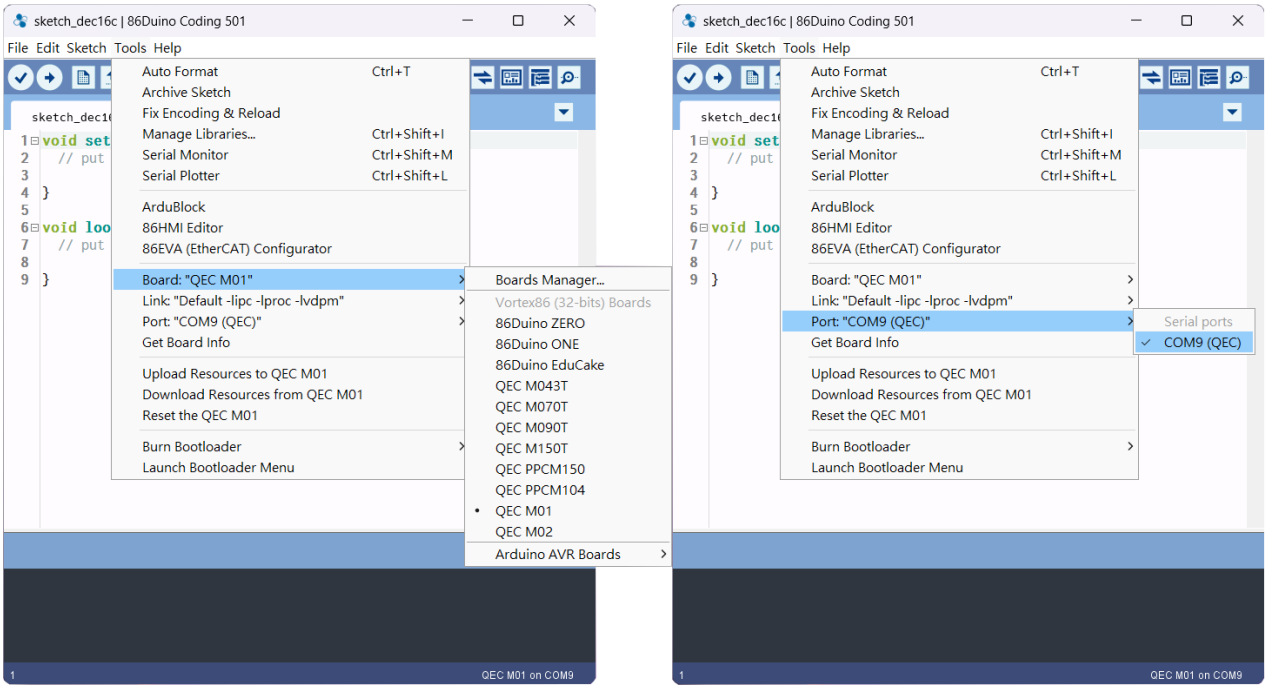
2.1 86Duino 環境設定

請依照以下步驟設定環境：

1. 使用 Micro USB 轉 USB 線將 QEC-M-01 連接到您的電腦（需安裝 86Duino IDE）。
2. 開啟 QEC 電源。
3. 打開電腦上的"裝置管理員"（按 Win+X 後在選單中選擇）->"連接埠 (COM 和 LPT)"，展開連接埠；您應該可以看到"Prolific PL2303GC USB 串列埠 COM 連接埠 (COMx)"；如果沒有，您需要安裝所需的驅動程式。(Windows PL2303 驅動程式可在此 下載)



4. 開啟 86Duino IDE。
5. 選擇正確的開發板：在 IDE 選單中，選擇「工具」>「開發板」>「QEC-M-01」（或您使用的 QEC MDevice 型號）。
6. 選擇連接埠：在 IDE 的選單中，選擇"工具">"連接埠"，然後選擇要連接到 QEC MDevice 的 USB 連接埠（在本例中為 COM9 (QEC)）。



2.2 Telnet_Ethercat_MOONS_Log.ino 程式碼說明

針對 QEC-M-01 上燒入的程式碼做說明，可分為 Ethernet 網路設定、EtherCAT 馬達初始化、和指令解析與位置回報，三個部分。

2.2.1 網路設定

在 .ino 開頭確認以下設定符合你的網路環境:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);       // QEC-M-01 的固定 IP
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
```

【注意】

IP 位址必須與你的 PC 在同一子網段,且不與其他設備衝突。修改後需重新燒錄。

2.2.2 EtherCAT 馬達初始化

`setup()` 中依序完成 EtherCAT 初始化:

```
master.begin();                          // 啟動 EtherCAT Master
motor.attach(0, master);                 // 掛載第 0 個 Slave(Moon's 驅動器)
master.start(1000000, ECAT_SYNC);        // 週期 1ms,同步模式
motor.setCiA402Mode(CIA402_PP_MODE);     // Profile Position 模式
motor.enable();                          // 使能馬達
motor.pp_SetVelocity(100000);            // 速度:100000 pulse/s
motor.pp_SetAcceleration(5000);          // 加速度
motor.pp_SetDeceleration(5000);          // 減速度
```

CIA402_PP_MODE 是 CiA402 標準的 Profile Position 模式,適合點對點定位控制。
Moon's 驅動器預設支援此模式,不需要額外設定。

2.2.3 指令解析與位置回報

`loop()` 做兩件事:接收 Telnet 指令、每秒廣播一次馬達位置。

```
char thisChar = client.read();
if (thisChar == 'p') motor.pp_Run(100000);   // 正轉至 +100000
else if (thisChar == 'n') motor.pp_Run(-100000);  // 反轉至 -100000
else if (thisChar == 'd') motor.disable();
else if (thisChar == 'e') motor.enable();

// 每秒回報位置
if (millis() - t0 >= 1000) {
    server.print("Position: ");
    server.println(motor.getPositionActualValue());
    t0 = millis();
}
```

【注意】

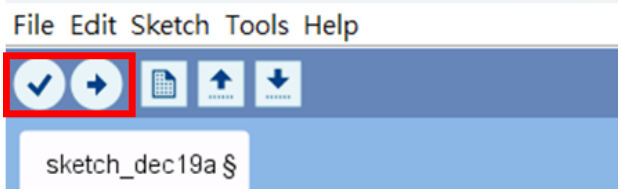
`pp_Run()` 的參數單位是取決於你的驅動器解析度設定。

【小訣竅】

燒錄完成後,用任何 Telnet 客戶端(例如 Windows 的 PuTTY)連接 192.168.1.177:23,應該看到 "Hello, client!" 回應,代表網路連線正常。

2.3 上傳程式碼

程式碼寫完成後，點選工具列的 進行編譯，確認編譯完成且無錯誤後，即可點選 上傳。

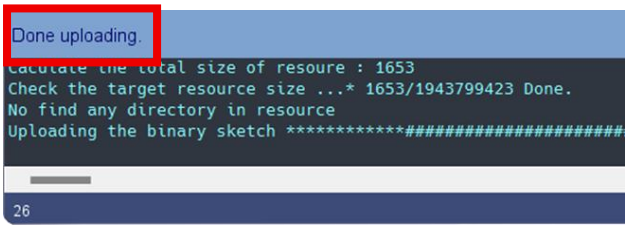


(圖中 sketch 名僅為示意)

【注意】

上傳期間 IDE 視窗請勿關閉、USB 線勿拔除;一般 sketch 上傳約 10~30 秒。

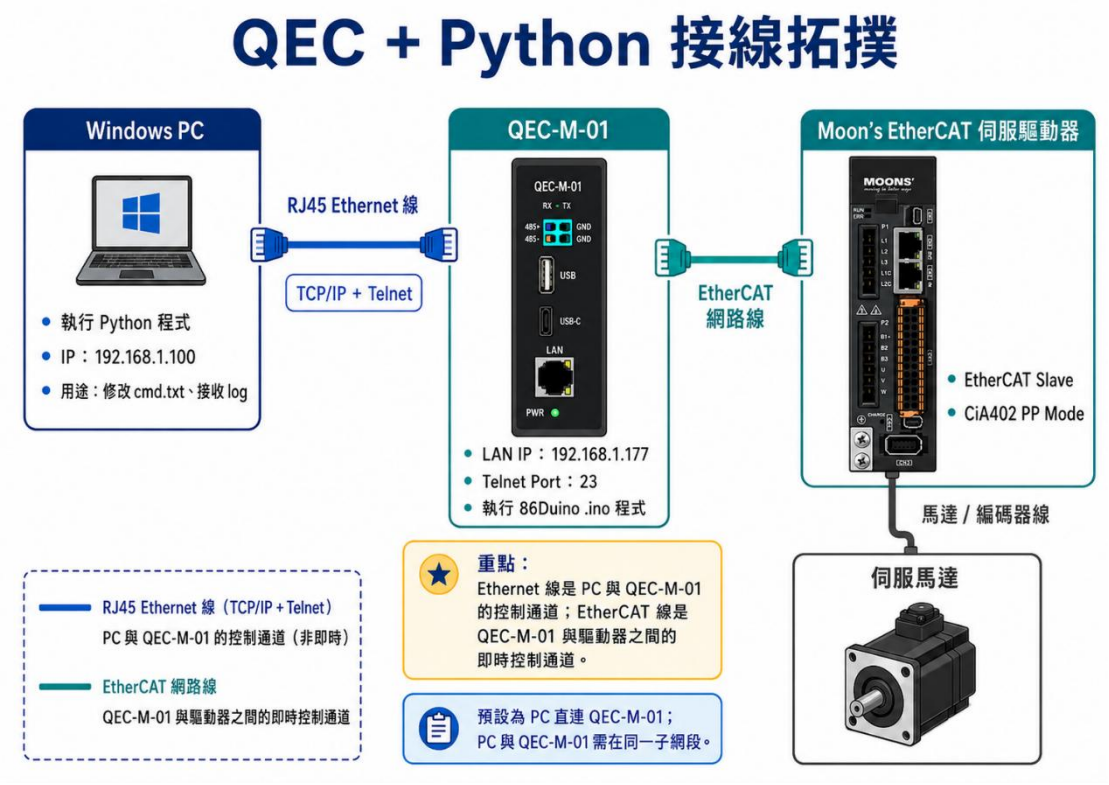
一般 sketch 上傳約 10~30 秒。視窗最後出現 Done uploading 即代表上傳成功；若出現紅字錯誤，則針對錯誤處去調整。如有任何其他異常，請與我們聯繫。



3. PC 上的網路設定

預設 PC 直連 QEC-M-01 正前方的 LAN (一條 RJ45 網路線、無中間 Switch / Router),PC 端必須手動設定靜態 IP 才能與 QEC-M-01 (預設 192.168.1.177) 同網段通訊。

3.1 接線拓撲



3.2 Windows 11 設定靜態 IP (推薦做法)

1. 按 Win + I 開啟「設定」(Settings)。
2. 左側選單選「網路和網際網路」(Network & Internet)。
3. 點選「乙太網路」(Ethernet);若有多張網卡,選擇接到 QEC-M-01 的那張。
4. 找到「IP 指派」(IP assignment) 項目,按右側「編輯」(Edit) 按鈕。
5. 把下拉選單從「自動 (DHCP)」改為「手動」(Manual),「IPv4」開關打開。
6. 填入下列數值並按「儲存」(Save):
   - IP 位址:192.168.1.100
   - 子網路遮罩 / 子網路前置長度:255.255.0.0 (或填 16)
   - 閘道 (Gateway):留空
   - 慣用 / 替代 DNS:留空;DNS over HTTPS 維持「關閉」

【注意】
Windows 11 22H2 之後部分版本欄位名稱改為「子網路前置長度 (Subnet prefix length)」,直接填數字 16 即代表 255.255.0.0。

3.3 傳統 ncpa.cpl 路徑 (Windows 10 / 11 通用,作為備用)

1. 按 Win + R,輸入 ncpa.cpl, Enter 開啟「網路連線」視窗。
2. 找到接到 QEC-M-01 的網卡,右鍵「內容」(Properties)。
3. 選「網際網路通訊協定第 4 版 (TCP/IPv4)」,按「內容」。
4. 勾「使用下列 IP 位址」,填:IP 位址 192.168.1.100、子網路遮罩 255.255.0.0、預設閘道留空、DNS 全部留空,按「確定」儲存。

3.4 驗證 — ipconfig

1. 驗證 IP 設定生效: cmd 執行

```
ipconfig
```

   - 對應網卡的「IPv4 位址」應顯示 192.168.1.100,「子網路遮罩」應顯示 255.255.0.0。
   - 若顯示 169.254.x.x,代表設定未套用 / 仍在 DHCP 模式 → 回頭檢查上述步驟。

4. 設定 PC 端 Python 環境

4.1 環境需求

Python 3.7 以上版本,不需要額外安裝套件。
watch_dir_telnet_log.py 只使用 Python 標準函式庫 (telnetlib、pathlib、argparse、datetime)。

> 【注意】
> telnetlib 在 Python 3.11 起被標記為 deprecated,Python 3.13 已移除。若你的 Python 版本 >= 3.11,建議改用 socket 模組替代,或鎖定 Python 3.10 以下版本。

4.2 準備指令檔

在任意路徑建立 cmd.txt,初始內容為空白或任意文字。例如:

```
C:\my\folder\cmd.txt
```

4.3 啟動監控程式

開啟 Windows CMD,切換至 watch_dir_telnet_log.py 所在目錄,執行:

```
python watch_dir_telnet_log.py --dir "C:\\my\\folder"
```

4.4 完整參數說明

參數	預設值	說明	範例
--dir	(必填)	cmd.txt 所在資料夾	--dir "C:\folder"
--file	cmd.txt	監看的檔名	--file ctrl.txt
--host	192.168.1.177	QEC-M-01 的 IP	--host 192.168.1.100
--port	23	Telnet 連接埠	--port 2323
--interval	0.1(秒)	檔案輪詢間隔	--interval 0.05
--log	log.txt	位置記錄輸出檔	--log output.txt

> 【注意】
> log.txt 預設輸出在執行指令時的當前目錄(即 watch_dir_telnet_log.py 所在位置),不是 --dir 指定的資料夾。

5. 遠端控制馬達

5.1 發送指令

程式啟動後,只需要修改 cmd.txt 的內容就能控制馬達:

cmd.txt 內容	觸發動作	說明
e	`motor.enable()`	馬達使能,啟動前必須先送此指令
p	`pp_Run(100000)`	移動至絕對位置 +100000 pulse
n	`pp_Run(-100000)`	移動至絕對位置 -100000 pulse
d	`motor.disable()`	馬達去使能,停止並釋放

Python 程式每 0.1 秒讀取一次 cmd.txt,只有當檔案內容發生變化時才會送出指令,避免重複觸發。

【小訣竅】
最快的操作方式是在 Windows 檔案總管開啟 cmd.txt,用記事本編輯後存檔(Ctrl+S)。Python 偵測到 mtime 變化後會在 0.1 秒內送出指令。

6. 讀取 Log

6.1 Log 格式

log.txt 每行格式為:

```
[YYYY-MM-DD HH:MM:SS.mmm] <Telnet 回傳內容>
```

實際範例(來自測試執行):

```
[2025-10-22 16:27:27.373] Position: -100000
[2025-10-22 16:27:28.356] Position: -100000
[2025-10-22 16:26:01.873] Position: 99429
[2025-10-22 16:26:02.853] Position: 94534
[2025-10-22 16:26:03.943] Position: 84646
```

馬達移動中,位置數值會從目前位置連續變化至目標位置,可以從 Log 重建馬達的完整運動軌跡。

6.2 Log 的已知行為

由於 Telnet 是基於 TCP 的串流協議,`server.print()` 和 `server.println()` 的輸出有時會在緩衝邊界處被分割成兩行:

```
[2025-10-22 16:26:09.848] -
[2025-10-22 16:26:09.958] 56404
```

這兩行合在一起才是完整的位置值「-56404」。這是 TCP 分片行為,不影響數值的正確性,但若需要做後處理分析,需要在 parsing 時考慮到這個情況。

7. 操作教學:首次端到端驅動

本章整合上述步驟,以「Purpose / Prerequisites / Steps / Expected Results / Verification / Common Issues」結構,引導使用者完整跑完一次 e → p → n → d 驅動週期,並驗收 log 寫入正確。

7.1 目的

以 Python watch_dir_telnet_log.py 完成一輪 e → p → n → d 驅動,驗證 cmd.txt 監看與 Telnet 送出機制工作正常,且馬達位置紀錄被正確寫入 log.txt。

7.2 前置條件

- PC 網卡設為 192.168.1.100 / 255.255.0.0,ipconfig 確認生效,ping 192.168.1.177 通,PuTTY 手動測試確認馬達會依字元動作。
- Python 已安裝,watch_dir_telnet_log.py 與 cmd.txt 放置完成 (例如 C:\qec\watch_dir_telnet_log.py、C:\qec\cmd\cmd.txt)。
- Telnet_Ethercat_MOONS_Log.ino 已燒入 QEC-M-01,Serial Monitor 顯示:

```
Begin: 1
Slave: 1
Start: 1
Enable: 1
Chat server address: 192.168.1.177
```

【注意】
(若已上驅動器主電源) 急停按鈕已釋放、機構淨空、確認馬達轉動不會撞機。

7.3 步驟

1. 開啟 cmd 並切換到工作目錄:

```
cd /d C:\qec
```

2. 執行 Python:

```
python watch_dir_telnet_log.py --dir "C:\qec\cmd"
```

3. Console 應顯示 [TELNET] Connecting → [TELNET] Connected → [INFO] Watching file...。
4. 用記事本開啟 C:\qec\cmd\cmd.txt,內容改為單一字元 e, Ctrl + S 儲存。
5. Console 應在 0.1 秒內顯示 [SEND] 'e' sent。
   此時驅動器若有 Alarm 應清除。
6. 把 cmd.txt 內容改為 p,儲存;Console 顯示 [SEND] 'p' sent,馬達正轉至接近 +100000 pulse。
7. 把 cmd.txt 內容改為 n,儲存;Console 顯示 [SEND] 'n' sent,馬達反轉至接近 -100000 pulse。
8. 把 cmd.txt 內容改為 d,儲存;Console 顯示 [SEND] 'd' sent,馬達 Disable 狀態。
9. Ctrl + C 結束 Python, Console 顯示 [INFO] Stopped by user.。
10. 檢查 log.txt (位於 cd 後的當前目錄,即 C:\qec\),內容應為形如 [YYYY-MM-DD HH:MM:SS.mmm] Position: <value> 的多行紀錄。

7.4 預期結果

- 每次儲存 cmd.txt, Console 都在 0.1 秒內顯示對應 [SEND] 訊息。
- 馬達依字元做出 enable / 正轉 / 反轉 / disable 動作。
- log.txt 每秒至少累積一行 Position 紀錄。
- Position 數值在運動過程中由目前位置平滑變化至 ±100000 附近。

7.5 驗收方式

- Console 訊息順序:[SEND] 'e' → 'p' → 'n' → 'd'。
- Serial Monitor 與 log.txt 中的 Position 數值同步遞變 (允許 ± 1 秒誤差)。
- 操作完成後 log.txt 行數 ≈ 操作秒數 (含 TCP 分片偶爾分兩行的情況)。

7.6 常見問題

- [SEND] 訊息出現但馬達不動:回頭做步驟二的 PuTTY 手動測試,可定位問題在 Python 端或驅動器端。
- cmd.txt 改了沒反應:某些 IDE 採 write-and-rename 寫檔導致 mtime 不變;改用記事本即可。
- ImportError: No module named 'telnetlib':Python 為 3.13+,參見「常見問題排查」。
- log.txt 看不到 Position 行,只有空白:確認 Python 仍在執行;若 Console 顯示 [LOOP] Error,通常為 socket 異常,程式會自動重連。

8. 完整程式碼

本章節錄完整原始碼供讀者對照閱讀。如欲直接下載完整測試檔案(免於手動複製貼上)。
請參考下方連結:

【下載】完整測試檔案

- 86Duino 端 (.ino 與相關函式庫):
  https://www.qec.tw/wp-content/uploads/2026/05/QEC_Python_Telnet_86DuinoCode.zip
- PC 端 (watch_dir_telnet_log.py + cmd.txt 範本):
  https://www.qec.tw/wp-content/uploads/2026/05/QEC_Python_Telnet_PythonCode.zip

8.1 Telnet_Ethercat_MOONS_Log.ino

```
/*
  Chat  Server

  A simple server that distributes any incoming messages to all
  connected clients.  To use telnet to  your device's IP address and type.
  You can see the client's input in the serial monitor as well.
  Using an Arduino Wiznet Ethernet shield.

  Circuit:
   Ethernet shield attached to pins 10, 11, 12, 13
   Analog inputs attached to pins A0 through A5 (optional)

  created 18 Dec 2009
  by David A. Mellis
  modified 9 Apr 2012
  by Tom Igoe

*/

#include <SPI.h>
#include <Ethernet.h>
#include "Ethercat.h"
EthercatMaster master;
EthercatDevice_CiA402 motor;

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network.
// gateway and subnet are optional:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);
IPAddress dnsserver(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);



// telnet defaults to port 23
EthernetServer server(23);
boolean alreadyConnected = false; // whether or not the client was connected previously
int32_t pos = 0;
static uint32_t t0   = 0;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // EtherCAT
  Serial.print("Begin: "); Serial.println(master.begin());
  Serial.print("Slave: "); Serial.println(motor.attach(0, master));
  Serial.print("Start: "); Serial.println(master.start(1000000, ECAT_SYNC));
  motor.setCiA402Mode(CIA402_PP_MODE);
  Serial.print("Enable: "); Serial.println(motor.enable());
  motor.pp_SetMotionProfileType(0); // Linear ramp (trapezoidal profile)
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
```

```
  // initialize the ethernet device
  Ethernet.begin(mac, ip, dnsserver, gateway, subnet);
  // start listening for clients
  server.begin();

  Serial.print("Chat server address:");
  Serial.println(Ethernet.localIP());

  // Set Timing
  t0 = millis();
}

void loop() {
  // Read Position
  pos = motor.getPositionActualValue();

  // wait for a new client:
  EthernetClient client = server.available();

  // when the client sends the first byte, say hello:
  if (client) {
    if (!alreadyConnected) {
      // clead out the input buffer:
      client.flush();
      Serial.println("We have a new client");
      client.println("Hello, client!");
      alreadyConnected = true;
    }

    if (client.available() > 0) {
      // read the bytes incoming from the client:
      char thisChar = client.read();
      if (thisChar == 'p')
        motor.pp_Run(100000);
      else if (thisChar == 'n')
        motor.pp_Run(-100000);
      else if (thisChar == 'd')
        motor.disable();
      else if (thisChar == 'e')
        motor.enable();
    }
  }

  // Error Log / per second
  if (millis() - t0 >= 1000) {
    // echo the bytes back to the client:
    server.print("Position: ");
    server.println(pos);
    // echo the bytes to the server as well:
    Serial.print("Position: ");
    Serial.println(pos);
    t0 = millis();
  }
}
```

8.2 watch_dir_telnet_log.py

```
# -*- coding: utf-8 -*-
"""
每 0.1 秒監看資料夾中的指定檔案內容，內容改變才送指令：
  'p' -> motor.pp_Run(100000)
  'n' -> motor.pp_Run(-100000)
  'd' -> motor.disable()
  'e' -> motor.enable()

同時，每輪把 Telnet 回傳的所有資料（數值/文字/字串）追加到 log.txt（若無則自動建立）。
- 非阻塞讀取 telnet 緩衝；每行附上本地時間戳。
- 連線中斷會自動重連。
"""

import argparse
import time
import telnetlib
from pathlib import Path
from typing import Optional
from datetime import datetime

VALID_CMDS = {'p', 'n', 'd', 'e'}

def connect_telnet(host: str, port: int, retry_seconds: float = 2.0) -> telnetlib.Telnet:
    """連線 telnet；失敗則每 retry_seconds 秒重試直到成功。"""
    while True:
        try:
            print(f"[TELNET] Connecting to {host}:{port} ...")
            tn = telnetlib.Telnet(host, port, timeout=5)
            print("[TELNET] Connected.")
            return tn
        except Exception as e:
            print(f"[TELNET] Connect failed: {e} ; retry in {retry_seconds}s")
            time.sleep(retry_seconds)

def send_char(tn: telnetlib.Telnet, ch: str) -> bool:
    """送出單一字元（不含換行）。成功回 True。"""
    try:
        tn.write(ch.encode("ascii"))
        print(f"[SEND] '{ch}' sent")
        return True
    except Exception as e:
        print(f"[SEND] Failed: {e}")
        return False

def read_command_char(file_path: Path) -> Optional[str]:
    """
    讀取檔案第一個非空白字元；若非 p/n/d/e 或檔案不存在，回傳 None。
    """
    if not file_path.exists() or not file_path.is_file():
        return None
    try:
        text = file_path.read_text(encoding="utf-8", errors="ignore")
    except Exception:
        return None
    for c in text:
        if not c.isspace():
            c = c.lower()
            return c if c in VALID_CMDS else None
    return None

def append_log_line(log_path: Path, line: str) -> None:
    """把單行（含換行）追加到 log.txt（若無會自動建立）。"""
    with open(log_path, "a", encoding="utf-8", errors="ignore") as f:
        f.write(line)

def drain_telnet_to_log(tn: telnetlib.Telnet, log_path: Path) -> None:
```

```
    """
    非阻塞讀取目前 telnet 緩衝的所有資料並寫入 log.txt（逐行加上時間戳）。
    """
    try:
        chunk = tn.read_eager()  # 非阻塞；可能為空
    except EOFError:
        # 連線被關閉，交由外層處理重連
        raise
    if not chunk:
        return
    text = chunk.decode(errors="ignore").replace("\r\r\n", "\n").replace("\r\n", "\n")
    ts = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
    for line in text.splitlines():
        append_log_line(log_path, f"[{ts}] {line}\n")

def main():
    ap = argparse.ArgumentParser(description="Watch file content change and send 'p'/'n'/'d'/'e'
via Telnet; log all Telnet outputs.")
    ap.add_argument("--dir", required=True, help=r"要監看的資料夾")
    ap.add_argument("--file", default="cmd.txt", help="要監看的檔名（位於 --dir 下，預設 cmd.txt）
")
    ap.add_argument("--host", default="192.168.1.177", help="Telnet 主機（預設 192.168.1.177）")
    ap.add_argument("--port", type=int, default=23, help="Telnet 連接埠（預設 23）")
    ap.add_argument("--interval", type=float, default=0.1, help="輪詢秒數（預設 0.1）")
    ap.add_argument("--log", default="log.txt", help="Telnet 回傳內容寫入檔名（預設 log.txt，寫在
執行目錄）")
    args = ap.parse_args()

    folder = Path(args.dir).resolve()
    if not folder.is_dir():
        raise SystemExit(f"--dir 不是資料夾：{folder}")
    file_path = (folder / args.file).resolve()
    log_path = Path(args.log).resolve()

    # 建立 telnet 連線
    tn = connect_telnet(args.host, args.port)

    last_mtime: Optional[float] = None
    last_cmd: Optional[str] = None

    print(f"[INFO] Watching file: {file_path} (Ctrl+C 結束)")
    try:
        while True:
            try:
                # 1) 檔案內容變化才送指令
                if file_path.exists():
                    mtime = file_path.stat().st_mtime
                    if last_mtime is None or mtime != last_mtime:
                        cmd = read_command_char(file_path)
                        if cmd is not None and cmd != last_cmd:
                            if not send_char(tn, cmd):
                                # 斷線則重連再送一次
                                try:
                                    tn.close()
                                except Exception:
                                    pass
                                tn = connect_telnet(args.host, args.port)
                                send_char(tn, cmd)
                            last_cmd = cmd
                        last_mtime = mtime
                else:
                    last_mtime = None  # 檔案重建時會被偵測

                # 2) 把 telnet 端回傳的所有資料寫入 log.txt
                try:
                    drain_telnet_to_log(tn, log_path)
                except EOFError:
```

```
                    # 連線被關閉，重連
                    try:
                        tn.close()
                    except Exception:
                        pass
                    tn = connect_telnet(args.host, args.port)

                time.sleep(args.interval)
            except Exception as e:
                print(f"[LOOP] Error: {e}")
                time.sleep(0.5)
    except KeyboardInterrupt:
        print("\n[INFO] Stopped by user.")
    finally:
        try:
            tn.close()
        except Exception:
            pass

if __name__ == "__main__":
    main()
```

9. 常見問題排查

現象	可能原因與處理方式
Connect failed 一直重試	(1) IP 設定錯誤,確認 QEC-M-01 與 PC 在同一子網段。 (2) QEC-M-01 尚未完成開機,等待 Serial Monitor 顯示 IP 位址後再啟動 Python。
送出指令但馬達沒反應	(1) 先確認 'e' 指令已送出 (馬達需要先 enable)。 (2) 檢查 EtherCAT 連線,Serial Monitor 應顯示 Begin / Slave / Start 均回傳正值。
Log 中位置數值被分兩行	正常現象,TCP 分片所致。合併相鄰兩行的數字字串即為完整數值。
telnetlib ImportError	Python 版本 >= 3.13,已移除 telnetlib。改用 Python 3.10 或以下,或改寫為 socket 版本。
指令重複送出多次	cmd.txt 被意外多次更新 (如自動備份工具)。 Python 程式設計上只有 cmd 值不同時才送指令,同一指令不會重複觸發。

10. 下一步

這個範例展示的是最基礎的單軸遠端控制架構。

QEC 平台的擴充方向包括:

- 多軸控制:增加 `motor2.attach(1, master)`,依此類推,最多支援多個 EtherCAT Slave
- 更精細的指令協議:在 Telnet 字串中傳遞目標位置數值,而不是固定的 ±100000
- SubDevice I/O 整合:透過 QEC SubDevice 模組讀取感測器或控制 Digital Output
- HMI 整合:將 Python 控制邏輯包裝成 GUI,搭配 tkinter 或 PyQt 做可視化操作介面

前往 QEC 產品頁了解 QEC-M-01 規格與支援的 EtherCAT 伺服驅動器品牌:https://www.qec.tw/

11. 聯絡我們

若您在依本指南實作的過程中遇到任何技術問題,或對 QEC 系列產品有採購或客製化需求,歡迎透過下列任一管道與我們聯繫:

業務洽詢 / 採購	info@qec.tw (QEC 產品線(EtherCAT MDevice 等))
技術支援 / RD	info@icop.com.tw (ICOP Technology Inc.)