第一堂課：ARM ↔ FPGA 操作教學

LED 控制 · 完整逐步操作文件

照著操作流水線走 6 步：接線 → 找 IP → SSH → 用 C+mmap 點 LED、跑馬燈 → 看板上綠燈動

CIS 影像辨識專案 · 後段辨識環境上手（今天用 LED 預習之後讀影像的方法）

淡江大學電機工程學系　AI 機器學習實驗室　·　2026-07

淡江大學
Tamkang University

AI 機器學習實驗室
AI & Machine Learning Lab
淡江大學 電機工程學系

課程目標

【導論】上完這堂課你會做到的事

- **認識 DE10-Nano：一塊板子裡其實有「兩個腦」——ARM 與 FPGA**
- 學會用 SSH，從你的 PC 連進板子的 Linux 下指令
- **★ 學會：還沒設定固定 IP 前，怎麼找出板子現在的 IP**
- 親手跑三支程式，讓 ARM 控制 FPGA 的 LED（看得到燈在動）
- 理解這一切對「影像辨識專案」的意義與對應

學習方法：照著「操作流程 6 步」一步一步做，看到板子有反應就懂了。

這堂課在整個專案的哪裡？（為什麼要學）

【導論】先看大方向，細節後面再學就不會迷路

★ 今天這堂課：練「② 橋接」這個動作——ARM 用 mmap 過橋讀寫 FPGA。
先用 LED 練手，學會了再把「LED 位址」換成「影像位址」。

所以你今天看到的 LED，就是之後「影像資料」的預習版——一模一樣的方法。

板子有「兩個腦」：ARM 與 FPGA

【觀念】最容易混淆、但最重要的觀念

比較項目	ARM (HPS)	FPGA (Fabric)
跑什麼	Linux + 軟體 (C / Python)	硬體邏輯 (Verilog → 位元流)
像什麼	一台小電腦	一塊可程式化電路
這專案誰負責	後段團隊：寫辨識軟體	前段團隊：CIS 擷取硬體
怎麼溝通	透過「橋接」：ARM 用 mmap /dev/mem 讀寫 FPGA 的暫存器 (位址 0xFF20_xxxx)	

✓ 我們寫的程式都跑在【ARM】這顆；FPGA 裡的 led_pio / LED 是現成硬體（GHRD），我們只是去「戳」它。

名詞速查（看到不懂的字，翻這頁）

【觀念】完全沒接觸過也沒關係，先有印象就好

FPGA：可以「重新設定線路」的晶片（硬體可改）

ARM / HPS：板子裡的 CPU，這裡跑 Linux

SoC：把 CPU 和 FPGA 包在同一顆晶片裡

Linux：作業系統（像 Windows，常用在裝置）

暫存器 register：硬體的「控制開關格子」，寫數字進去就控制硬體

位址 address：每個格子的「門牌號碼」，如 0xFF203000

hex 十六進位：0x 開頭的數字寫法，比二進位短

bitstream .sof：FPGA 的「設計檔」，燒進去才知道怎麼接線

JTAG：燒 FPGA 用的通道（Mini-USB Blaster）

SSH：從 PC 遠端登入板子 Linux 的加密通道

序列 / COM：不靠網路、直接跟板子說話的線（COM5）

IP 位址：板子在網路上的門牌（192.168.50.199）

mmap：把硬體位址對映成程式可直接讀寫的記憶體

三個一定要懂的基礎：位址、暫存器、hex

【觀念】看懂後面所有程式的根基

1

位址 (address) = 門牌號碼

硬體的每個控制點都有一個號碼，例如 LED 在 0xFF203000。

2

暫存器 (register) = 開關格子

在那個位址「寫一個數字」= 操作硬體；「讀」= 看硬體狀態。

3

hex (十六進位) = 數字的簡寫

0x 開頭；0x7F 其實就是二進位 1111111（7 個 1）。

串起來：在「位址 0xFF203000」的「暫存器」寫「0x7F」= 7 顆 LED 全亮。後面所有程式都在做這件事。

比喻：位址 = 門牌、暫存器 = 房間裡的開關面板、hex = 開關設定的簡寫。

二進位 ↔ 十六進位 (hex) 換算小抄

【觀念】關鍵：1 個 hex 符號 = 剛好 4 個 bit

hex	二進位	十進位
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

hex	二進位	十進位
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

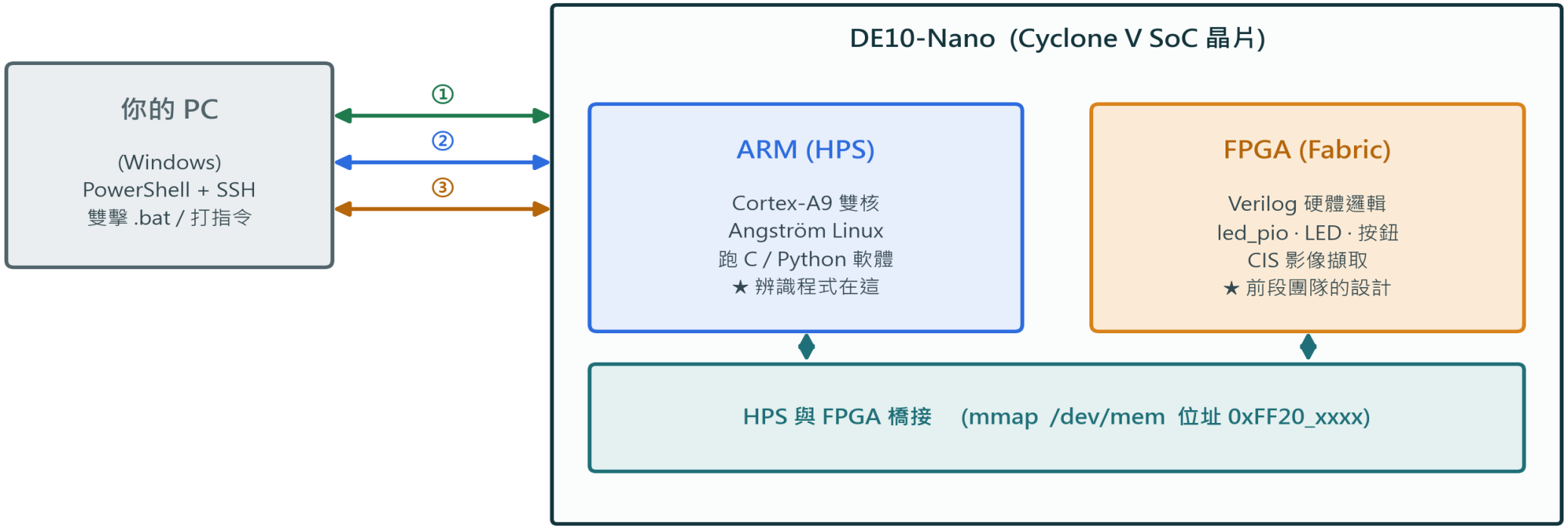
拆一個 byte（兩個 hex 位）：0x7F → 7=0111、F=1111 → 0111 1111 → 7 顆 LED 全亮（最高位沒接到燈）。

0x55 → 5=0101、5=0101 → 0101 0101 → 交替亮（第 1/3/5/7 顆）。

系統方塊圖

【觀念】整台板子的組成一圖看懂

圖一　系統方塊圖：你的 PC 如何透過 SSH 操控 DE10-Nano（ARM + FPGA）



三條連接線：

① Ethernet 網路線 → SSH / 網頁　　② Micro-USB(OTG) → 序列 COM5 / 網路　　③ Mini-USB → JTAG 燒 FPGA

重點：我寫的程式跑在【ARM】(軟體)，透過【橋接】讀寫【FPGA】的暫存器；FPGA 硬體邏輯是現成的 GHRD。

先學會用「終端機 / 命令列」

【準備】就是一個「用打字下指令」的視窗，不是用滑鼠點

- **命令列 = 你打一行字、按 Enter，電腦就照做（比點圖示更直接）**
- Windows 開法：按 Win 鍵 → 打 powershell → 按 Enter，出現一個視窗
- 看到開頭那串提示字在閃 = 換你打字了（那串提示不用自己打）
- 打完一條指令 → 按 Enter 執行；想清空畫面打 cls（或 clear）

複製貼上 & 省力鍵（很重要）

- **貼上：在視窗裡按「滑鼠右鍵」= 貼上（這裡 Ctrl+V 不一定有用）**
- ↑ / ↓ 方向鍵：叫回剛剛打過的指令，不用整條重打
- Tab 鍵：打一半按 Tab，自動補完檔名 / 路徑
- 打錯不會壞：看到 not recognized / command not found = 打錯字，重打就好

進板子後最常用的 Linux 指令

【準備】用 SSH 連進去之後，就是在板子的 Linux 裡打這些

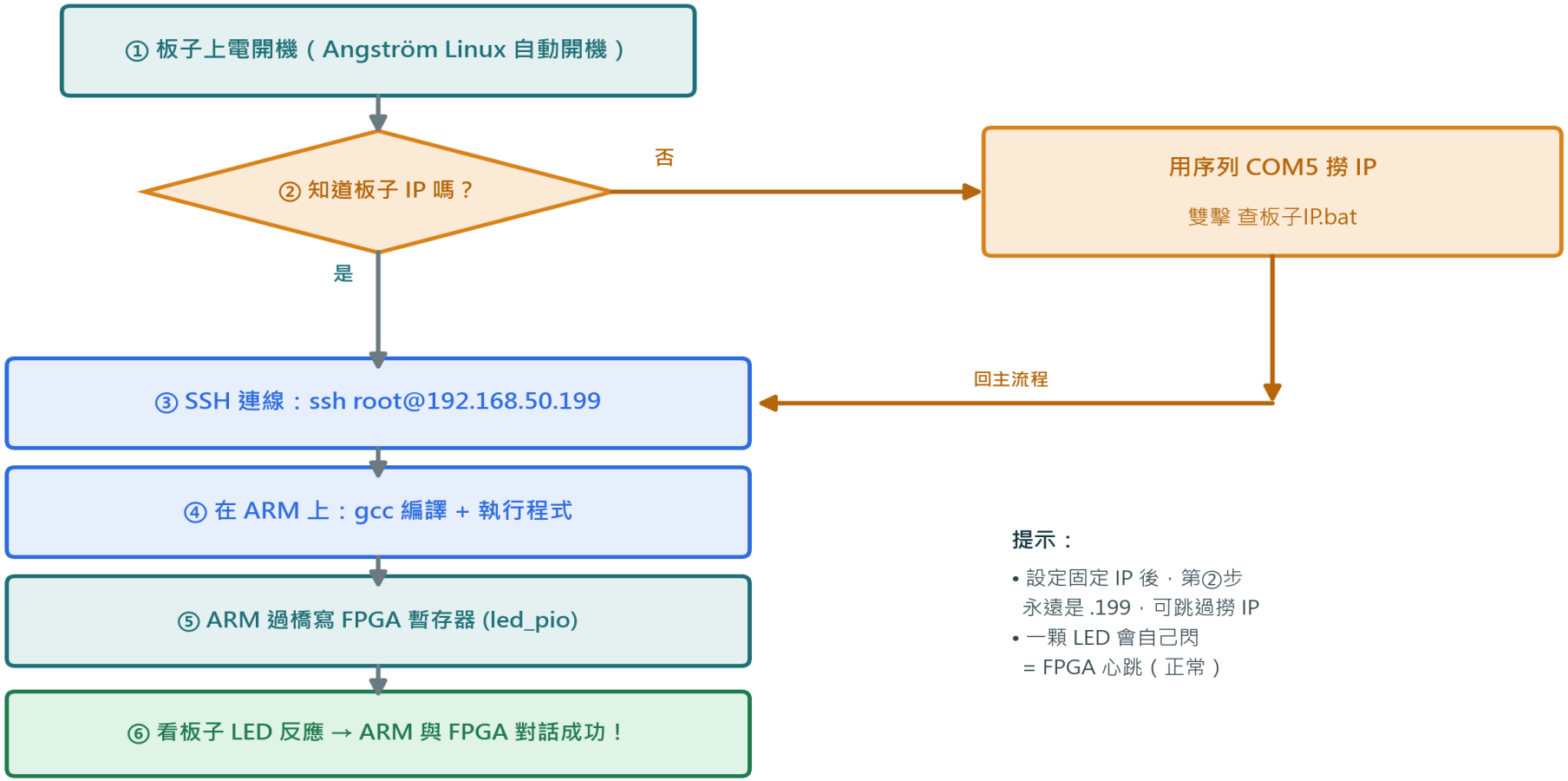
指令	作用	例子
`pwd`	我現在在哪個資料夾	`pwd`
`ls`	列出這個資料夾有什麼	`ls`　　`ls -l`
`cd`	進到某個資料夾	`cd /home/root`
`cd ..`	回上一層資料夾	`cd ..`
`cat`	把檔案內容印出來	`cat readme.txt`
`./名稱`	執行這支程式	`./fpga_led 0x7F`
`gcc`	把 C 原始碼編成程式	`gcc -O2 -o fpga_led fpga_led.c`
`reboot`	重新開機板子	`reboot`

小提醒：Linux 區分大小寫（LS ≠ ls）；打一半按 Tab 會自動補全；每條指令打完都要按 Enter。

操作流程總覽（接下來照這 6 步走）

【全景】接線 → 找IP → SSH → 點燈 → 跑馬燈 → 看結果

圖二　操作流程圖：從開機到讓 ARM 控制 FPGA

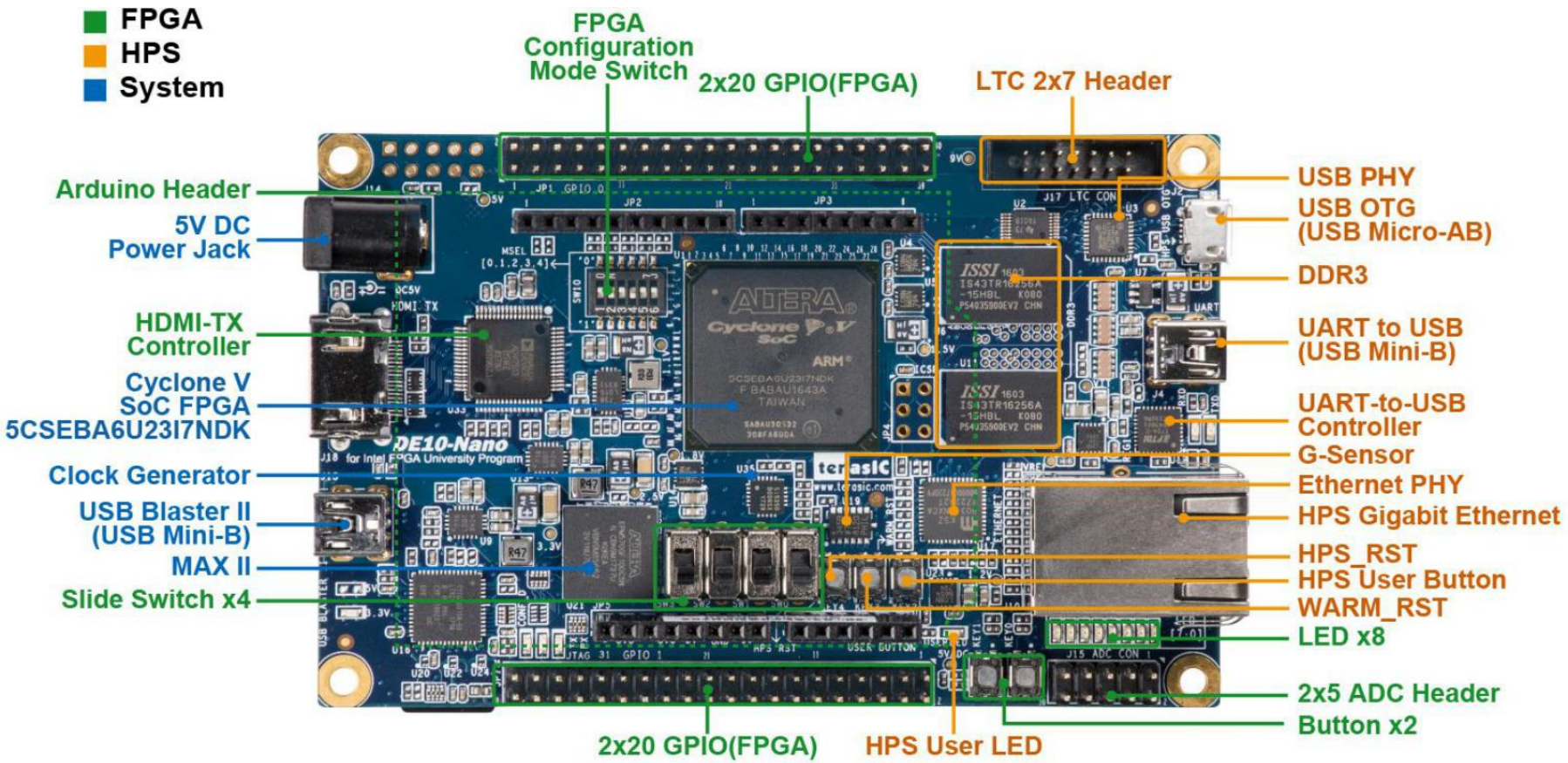


提示：

- 設定固定 IP 後，第②步永遠是 .199，可跳過撈 IP
- 一顆 LED 會自己閃 = FPGA 心跳（正常）

接線、開機

操作流程　第 1 步 / 共 6 步　·　認識板子俯視照 · 該插哪幾個孔



⚠ 操作下指令：插「USB OTG (Micro-AB)」+「HPS Gigabit Ethernet」+ 電源；要燒 FPGA 才插「USB Blaster II (Mini-B)」。開機等約 30 秒。

找板子的 IP

STEP 2 / 6

操作流程　第 2 步 / 共 6 步　·　IP 會變，但 SSH 要先知道 IP 才能連（雞生蛋）

問題：IP 一變，SSH 就連不上 → 你「沒辦法用 SSH 查 IP」

解法：用「序列線 COM5」——它不經過網路，直接問板子！

方法 A　手動（學原理）

- 用序列終端機開 COM5（115200, 8N1）
- 登入：帳號 root、密碼直接 Enter（空的）
- 打指令：ifconfig eth0
- 看這一行 → 那就是 IP：

```
inet addr: 192.168.50.199
```

方法 B　懶人（最快）

- **雙擊 查板子IP.bat（.ps1 不能雙擊）**
- 它自動：開 COM5 → 登入 → 跑 ifconfig
- 直接印出板子 IP + 連線網址：

```
Board IP = 192.168.50.199
SSH : ssh root@192.168.50.199
Web : http://192.168.50.199:8080
```

補充：也可登入家用路由器後台 → 看「DHCP 用戶端清單」找 hostname de10-nano 的 IP。

工具程式：get_board_ip.ps1

STEP 2 / 6

操作流程　第 2 步 / 共 6 步　·　把上一頁「手動撈 IP」整包自動化

跑在：你的 PC（PowerShell）

用途：透過序列 COM5 自動撈 IP

```
$p = New-Object System.IO.Ports.SerialPort('COM5',115200,'None',8,'One')
$p.Open()                          # 開啟序列埠 COM5
$p.Write("root`r")                # 送出帳號 root
$p.Write("`r")                   # 送出空密碼 (直接 Enter)
$p.Write("ifconfig eth0`r")  # 叫板子印出網路設定
$out = $p.ReadExisting()          # 讀回結果
# 再用正則從 $out 抓出  inet addr: 192.168.50.xx
```

- **核心觀念：序列線不靠網路 → 板子掉網/換IP 都能進去問**
- 等於把手動「開 COM5 → 登入 → ifconfig」整包自動化
- 回傳值直接給你 ssh 指令，複製就能連

SSH 連進板子的 Linux

STEP 3 / 6

操作流程　第 3 步 / 共 6 步　·　Secure Shell：你的鍵盤接到板子 Linux 上

- **SSH = 一條「加密的遠端登入通道」，從你的 PC 連進板子**
- 連上後，你在 PC 打的指令，其實是在「板子上」執行

格式：ssh 使用者@板子IP

```
ssh root@192.168.50.199
```

- **開 PowerShell：按 Win 鍵 → 打 powershell → Enter**
- 貼上指令：在視窗按滑鼠右鍵 = 貼上 → 按 Enter
- 第一次問 (yes/no)？→ 打 yes；root 沒密碼，不會問密碼
- 小撇步：按 ↑(上方向鍵) 可叫回上一條指令

一句話

「我的鍵盤接到板子的 Linux 上」

```
你的 PC ──加密通道──▶ 板子 Linux
(打指令)                (真正執行)
```

點燈：fpga_led.c（mmap 寫 led_pio）

操作流程　第 4 步 / 共 6 步　·　讀身份 System ID + 設定 LED（ARM→FPGA）

跑在：ARM（Linux，用 gcc 編譯）

用途：讀 System ID + 設定 LED

```
int fd = open("/dev/mem", O_RDWR | O_SYNC);        // 開實體記憶體
void *v = mmap(NULL, 0x04000000, PROT_READ|PROT_WRITE,
               MAP_SHARED, fd, 0xFC000000);        // 映射橋接區
volatile uint32_t *led =
        v + ((0xFF200000 + 0x3000) & 0x03FFFFFF);  // led_pio 位址
led[0] = 0x7F;                  // 寫 → 7 顆 LED 全亮 (ARM→FPGA)
printf("LED = 0x%02X\n", led[0]);   // 讀回確認
```

- **板上 python3 沒有 mmap 模組、devmem 也沒有 → 用 C + mmap 最穩**

編譯 + 執行：gcc -O2 -o fpga_led fpga_led.c　然後　./fpga_led 0x7F

- 0xFF203000 = led_pio，接到板上 LED[7:1]（GHRD 硬體決定的）

LED 數字怎麼設？位址怎麼來？

操作流程　第 4 步 / 共 6 步　·　看懂 0x7F、0x55 和 0xFF203000 是什麼意思

圖四　LED 值怎麼看：7 個 bit，每個 bit 控制一顆 LED（1 = 亮、0 = 暗）

hex 值	二進位（低 7 bit）	LED7	LED6	LED5	LED4	LED3	LED2	LED1	
0x7F	1 1 1 1 1 1 1	●	●	●	●	●	●	●	全部亮
0x55	1 0 1 0 1 0 1	●	○	●	○	●	○	●	交替亮 4 顆
0x2A	0 1 0 1 0 1 0	○	●	○	●	○	●	○	交替亮 3 顆
0x01	0 0 0 0 0 0 1	○	○	○	○	○	○	●	只亮 LED1
0x00	0 0 0 0 0 0 0	○	○	○	○	○	○	○	全部暗

hex 只是二進位的簡寫：led = 0x55 → 1010101 → LED 1、3、5、7 亮。改數字 = 改亮哪幾顆。

位址怎麼來： `0xFF203000` = `0xFF200000`（HPS 輕量橋接起點） + `0x3000`（led_pio 偏移，來自 GHRD 位址表）

雙向：fpga_blink.c（讀開關 + 跑馬燈）

操作流程　第 5 步 / 共 6 步　·　FPGA→ARM 讀、ARM→FPGA 寫，兩個方向都示範

跑在：ARM（Linux）

用途：讀開關/按鈕 + LED 跑馬燈

```
// 讀 (FPGA → ARM)：讀板上實體指撥開關
printf("DIPSW = 0x%X\n", *dipsw & 0xF);

// 寫 (ARM → FPGA)：LED 跑馬燈
for (i = 0; i < 7; i++) {
    *led = (1 << i);    // 一次亮一顆，往左移動
    msleep(90);         // 停 90 毫秒
}
*led = 0x55;            // 最後停在「交替亮」
```

- **兩個方向都示範：**
  - 讀：DIPSW / BUTTON = FPGA→ARM
  - 寫：LED 跑馬燈 = ARM→FPGA
- **這就是辨識端「從 FPGA 拿資料、回寫結果」的雛形**
- 懶人版：雙擊 LED_BLINK.bat 也能看

看結果：板上綠燈動了！

STEP 6 / 6

操作流程　第 6 步 / 共 6 步　·　完整 6 步一次 recap（照著做就會成功）

1	**接線**	Micro-USB ＋ 網路線 ＋ 電源；開機等約 30 秒
2	**找 IP**	雙擊 查板子IP.bat → 得到 192.168.50.199
3	**SSH 連線**	ssh root@192.168.50.199（第一次問 yes/no → yes）
4	**點燈**	./fpga_led 0x7F　或　雙擊 LED_ON.bat
5	**跑馬燈**	./fpga_blink　或　雙擊 LED_BLINK.bat
6	**看結果**	板上那排綠燈跟著亮/動 → ARM 成功控制 FPGA！

板上 SW/KEY 旁那排綠燈亮/跑起來 = ARM 透過 mmap 成功控制了 FPGA。恭喜，橋接練成！

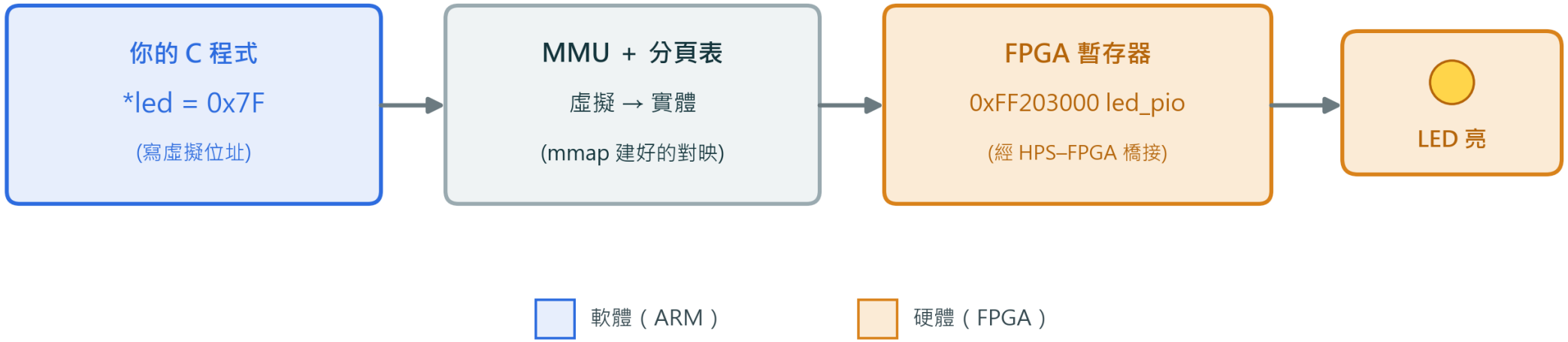
深入：mmap 是什麼？

【深入】為什麼一句 *led = 0x7F 就能控制硬體

圖三　mmap：把「虛擬位址」對映到 FPGA 的「實體暫存器」

① 設定（只做一次）：mmap() 在「分頁表」建立 虛擬位址 → 實體位址 的對映

② 之後每次 *led 存取（CPU 直接走，不再呼叫 OS）：



一句話：mmap 先建好「虛擬位址 → 實體暫存器」的對映；之後寫那個虛擬位址，CPU 就直接寫進 FPGA 暫存器，不再麻煩作業系統。

存取 FPGA 暫存器的方法（這台板上）

【深入】要戳硬體只能走「mmap 派」；read/write 派在這台會 EFAULT

方法	能用？	說明
C + mmap	可 · 最穩	gcc 板上就有、編一次到處跑、最貼近正式碼
Python 2 + mmap	可	py2 有 mmap 模組 → 想用腳本快速試也行（py2 較舊）
Python 3 + mmap	不可	這台 python3 沒編進 mmap 模組 → import 不了
devmem 指令	不可	沒裝；opkg 在這台舊系統多半也連不到
dd / os.read·write	不可 · EFAULT	走 read/write，硬體暫存器這台核心不支援

✓ 我們的工具都走「C + mmap」：fpga_led（點燈）、fpga_blink（雙向）、reg（通用讀寫任何暫存器）。

（進階）設定固定 IP，一勞永逸

【工程實務】讓板子永遠是 192.168.50.199

- **為什麼會變：DE10-Nano 每次開機隨機產生新 MAC → DHCP 就配到不同 IP**
- 解法：用板子內建的 connman 把它鎖成「靜態 IP」

在板子上（SSH 進去後）執行一次：

```
connmanctl config ethernet_000000000000_cable \
    --ipv4 manual 192.168.50.199 255.255.255.0 192.168.50.1
```

- **設定後重開機也不變 → 之後第 2 步永遠是 .199，可跳過撈 IP**
- 要還原成自動取得：把 --ipv4 manual ... 改成 --ipv4 dhcp

本案已設定完成：板子固定在 192.168.50.199。

「燒錄 FPGA」是什麼？什麼時候別亂燒

【工程實務】今天其實沒燒 FPGA——但要先知道安全規矩

- **燒錄 FPGA = 把一份「線路設計檔（.sof / 位元流）」灌進 FPGA，讓它照那張藍圖接線**
- 像「給 FPGA 換一張電路藍圖」；換了，硬體的行為就跟著變
- 今天我們沒燒：FPGA 裡的線路（GHRD）開機就在，我們只是用 ARM 去戳它的暫存器

⚠ 安全鐵則（不照做會害板子重開、IP 跑掉）

- **別在「正在跑 Linux」的板子上、用 JTAG 燒「純 FPGA」設計 → 會拆掉 ARM↔FPGA 橋接、板子重開機**
- 要改 FPGA：燒「含 HPS 的 SoC 設計（GHRD）」，或從 Linux 用 fpga_manager 載入 → 才不會自爆
- 燒錄要插「USB Blaster II（Mini-USB）」那個孔；平常下指令是「USB OTG（Micro-USB）」
- 燒錄 / 開機到一半，別拔電源

一句話：今天不需要燒 FPGA；真要燒，先確認那份設計是「含 HPS」的，再燒。

排錯 / 常見問題

【工程實務】卡住時先看這頁

Q 連不上板子 / ssh 逾時？

A 雙擊 查板子IP.bat 用序列撈現在的 IP；檢查網路線、電源有沒有鬆。

Q 看不到 LED 在動？

A 看「SW/KEY 旁那排綠燈」；有一顆會自己每秒閃一次 = FPGA 心跳，正常。

Q IP 每次開機都變？

A DE10-Nano 隨機 MAC 造成；設定固定 IP（前一頁）就一勞永逸。

Q 可以隨便燒 FPGA 嗎？

A 別在「正在跑 Linux」的板子上 JTAG 燒「純 FPGA」設計 → 會害板子重開機。

總結 & 對辨識專案的意義

第一堂課一頁記住

- 今天學會：板子有兩個腦、用 SSH 連線、（沒固定 IP 時）用序列找 IP、讓 ARM 用 mmap 控制 FPGA

對「影像辨識專案」的對應

前段團隊	→ 橋接 →	後段團隊
FPGA 硬體 CIS 掃描圖片、存影像	mmap /dev/mem 0xFF20_xxxx	ARM 軟體 讀影像 → 影像分類 / 辨識

下一步（第二堂課）：把今天的「led_pio 位址」換成「影像緩衝區位址」，同一套 mmap 就能把影像讀進辨識程式。

板子固定網址： `ssh root@192.168.50.199` | `http://192.168.50.199:8080`

第一堂課完 · 有問題隨時問