AIoT

feat. Jetson Nano, Raspberry Pi, ESP32

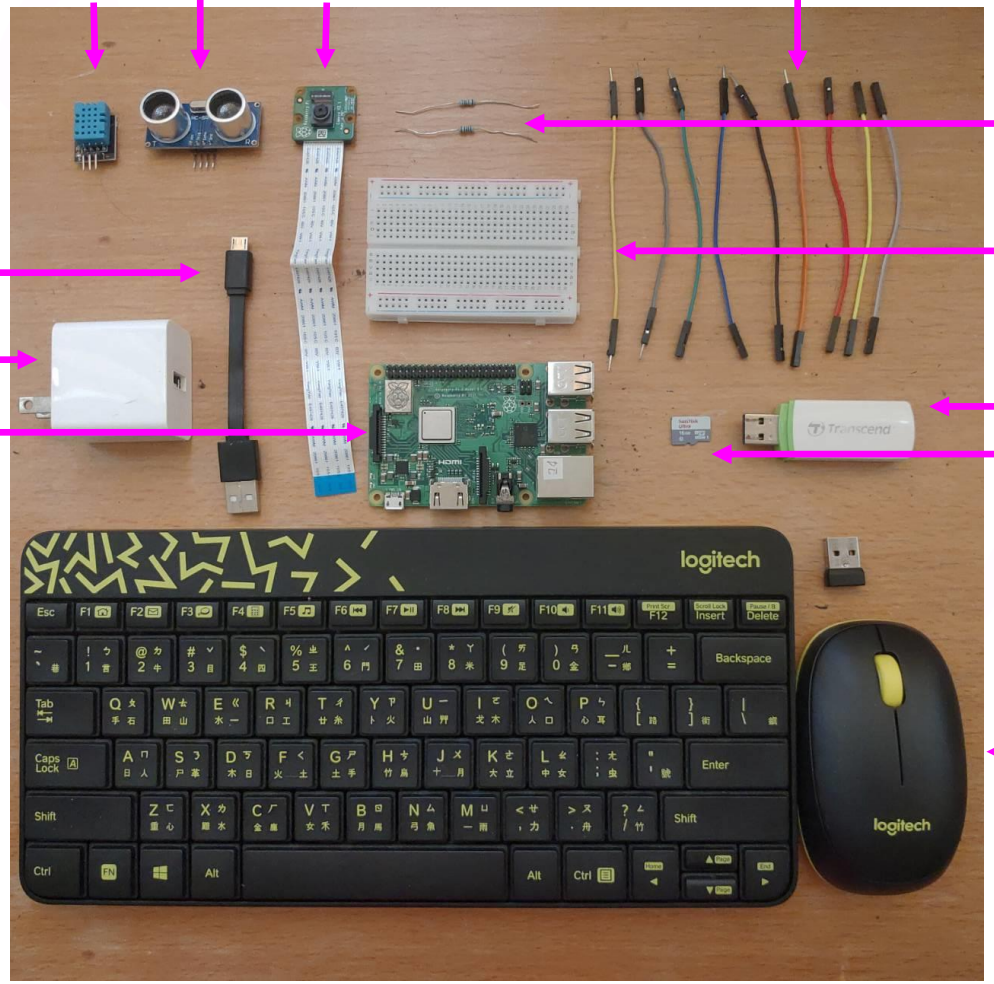
Enos Chou

2022/11/08

Enos' Steps

2022/10/18(二) 09:00~16:30	專題主題講座 by Enos
2022/10/25(二) 09:00~16:30	專題輔導 by Teams feat. Enos
2022/11/03(四) 09:00~16:30	專題初報 by Teams feat. Enos
2022/11/08(二) 09:00~16:30	AIoT by Enos
2022/11/22(二) 09:00~16:30	專題輔導 by Teams feat. Enos
2022/11/29(二) 09:00~16:30	GCP AI 應用 by Enos
2022/12/05(一) 09:00~16:30	GCP AI 應用 by Enos
2022/12/13(二) 09:00~16:30	專題輔導 by Teams feat. Enos
2022/12/22(四) 09:00~16:30	產業趨勢職涯講座 by Enos
2022/12/27(二) 09:00~16:30	專題輔導 by Teams feat. Enos
2022/12/30(五) 09:00~16:30	專題預報 by Teams feat. Enos

Before Raspberry AIoT



Before Raspberry AIoT

1. Edge

Board & Peripherals

- Raspberry Pi 3B+
- **USB-A** Power Adapter **5V2.5A**
- USB-A to **Micro-USB** Cable
- microSD 16G/ **UHS-I**/ U1

擇一
- CSI-2 Camera
- USB-A Webcam

- USB-A Keyboard
- USB-A Mouse
- HDMI Monitor
- HDMI Cable

Sensors

- Dupont Lines (F-F/ M-F)
- DHT-11 or DHT-22 (3 pins)
- HC-SR04
- Breadboard
- Resistors 1KΩ & 2KΩ

Before Raspberry AIoT

2. Environment

Laptop or Desktop

- Windows 10
- USB-A microSD Card Reader Dongle/ microSD Card Reader built-in

3. Backend

Cloud VM

選 • GCP Account Ready

Before Raspberry AIoT

4. Software

Windows 10

Python Environment

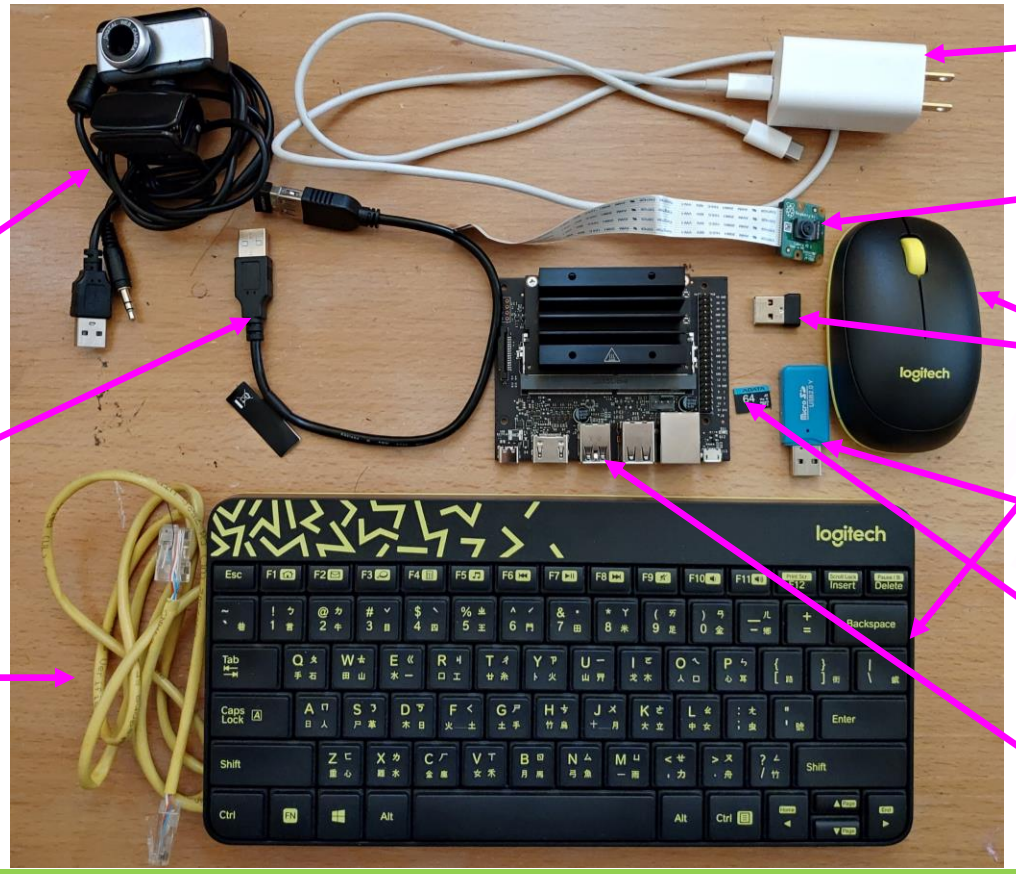
① Python 3.6.8 or 3.6.*

② TensorFlow 2.4.4

③ Jupyter Notebook latest

Before Jetson AIoT

1. Jetson Nano 2GB 的 USB-A 埠僅 3 組，須妥善安排
2. Camera 採用 CSI-2 介面可節省 USB-A 埠
3. 鍵盤滑鼠共用 USB-A
   可節省 1 組 USB-A 埠



USB-A Webcam

USB-A WiFi Dongle

RJ45 網路線

USB-C 5V3A 電源

CSI-2 Camera

無線鍵盤滑鼠，共用一組
USB-A 接收器

USB-A microSD 讀卡機

microSD 64G

Jetson Nano 2GB (NVIDIA)

Before Jetson AIoT

1. Edge

Board & Peripherals

- NVIDIA Jetson Nano 2GB
- **USB-C** Power Adapter **5V3A**
- microSD 64G/ **UHS-I**/ U1
- USB-A WiFi Dongle
- 擇一
  - CSI-2 Camera
  - USB-A Webcam
- USB-A Keyboard
- USB-A Mouse
- HDMI Monitor
- HDMI Cable

Before Jetson AIoT

2. Environment

Laptop or Desktop

- Windows 10
- 選 USB-A microSD Card Reader Dongle/ microSD Card Reader built-in
- 選 USB-A Ethernet Dongle or Ethernet built-in
- RJ45 Ethernet Cable

Before Jetson AIoT

3. Backend

Cloud VM

選 • GCP Account Ready

Before Jetson AIoT

4. Software

Windows 10

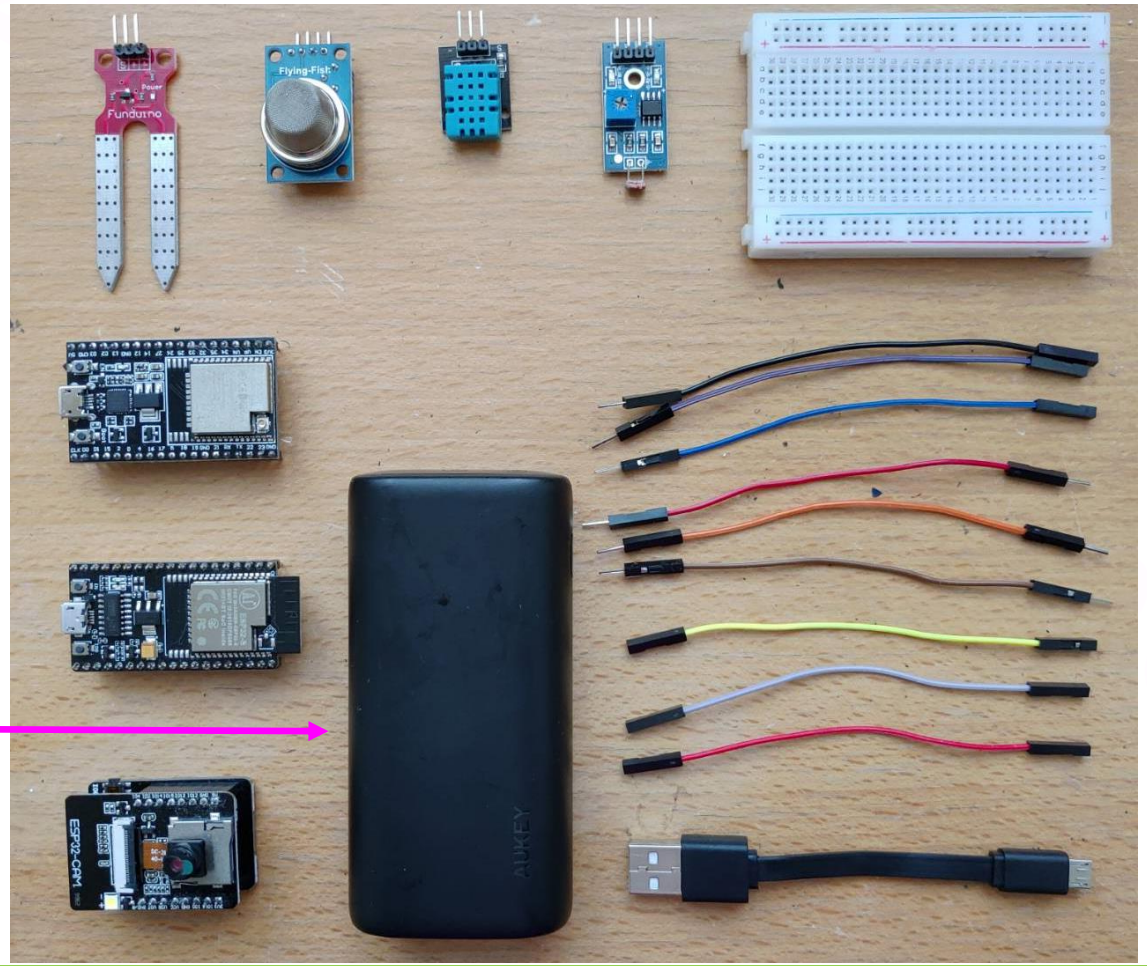
Jetson Nano 2GB Image (JetPack 4.6)

① 以瀏覽器搜尋 Jetson Download Center >
SEARCH: Jetson Nano 2GB >
Jetson Nano 2GB Developer Kit SD Card Image 4.6 >
Jetson Nano 2GB Developer Kit SD Card Image

② jetson-nano-2gb-jp46-sd-card-image.zip (6.08GB)

Before ESP32 AIoT

土壤溼度
MQ135
DHT-11
光敏電阻
麵包板



ESPRESSIF
ESP32-WROOM-32U

Ai-Thinker
NodeMCU-32

USB-A
小電流行動電源

ESP32-CAM

杜邦線 公-母
(搭配麵包板使用)
(數量視 sensor 而定)

杜邦線 母-母
(不接麵包板使用)
(數量視 sensor 而定)

USB-A to Micro-USB
Cable

Before ESP32 AIoT

1. Edge

Board & Peripherals

擇一
- ESPRESSIF ESP32-WROOM-32
- Ai-Thinker **NodeMCU-32**

- ESP32-CAM
- USB-A to **Micro-USB** Cable

選
- USB-A 小電流行動電源

Sensors

- Dupont Lines (F-F)

選
- Dupont Lines (F-M)
- Breadboard
- DHT-11 or DHT-22 (3 pins)

選
- Other Sensors

Before ESP32 AIoT

2. Environment

Laptop or Desktop

- Windows 10

3. Backend

Cloud VM

選 • GCP Account Ready

Topics

1. AIoT & Edge
   - Glance of AIoT & Edge
2. Raspberry Pi 3B+
   - Raspberry Pi Installation
   - Raspberry Faces (IoT, MQTT)
   - Raspberry Trees (TensorFlow Lite)
   - Raspberry Temperature (DHT-11)
   - Raspberry Distance (HC-SR04)
3. Jetson Nano 2GB
   - Jetson Installation
   - Jetson Faces (IoT, MQTT)
   - Jetson Trees (TensorFlow, TensorRT)
4. ESP32
   - MicroPython Installation on ESP32
   - ESP32 Sensors feat. MicroPython
   - ESP32CAM Stream
5. More

Glance of AIoT & Edge

IoT, AIoT, Edge

Terminology

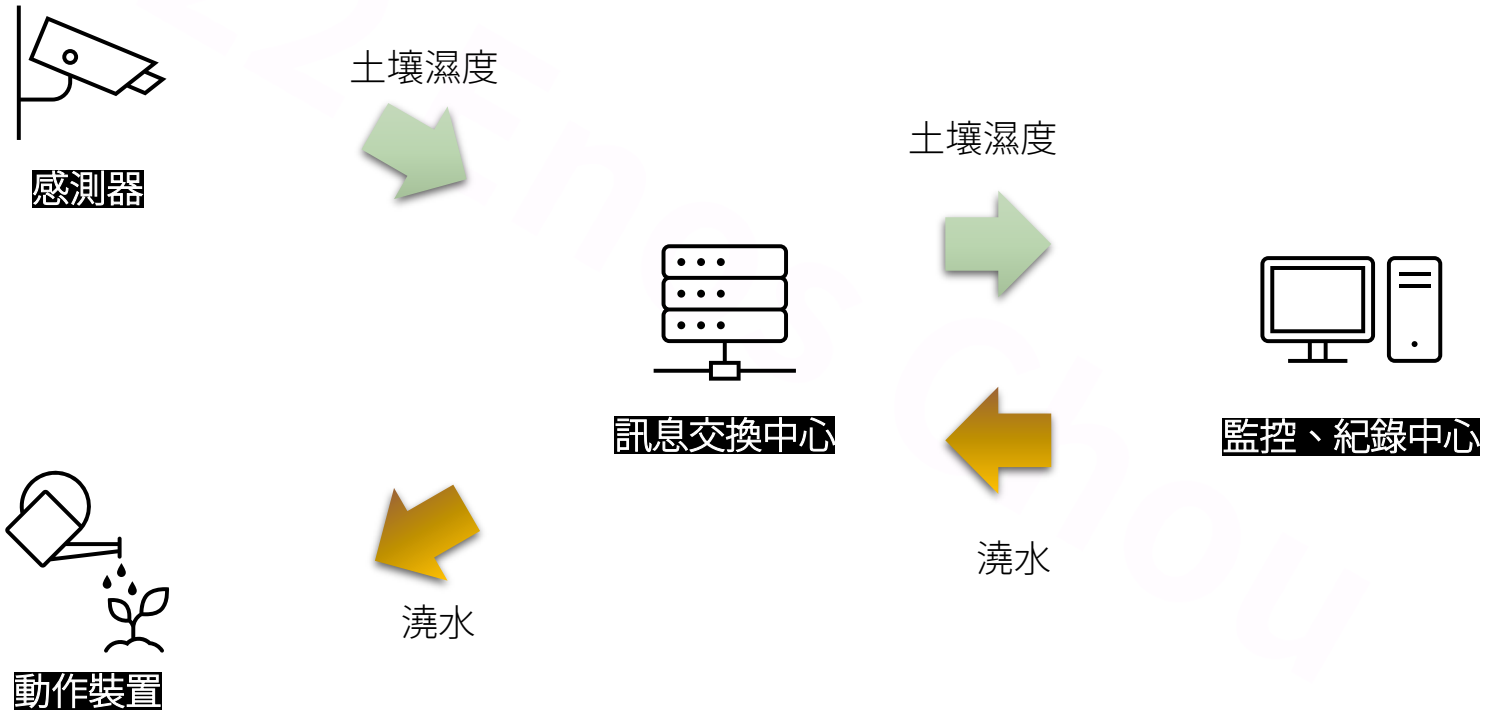
AI = Artificial Intelligence 人工智慧

IoT = Internet of Things 物聯網

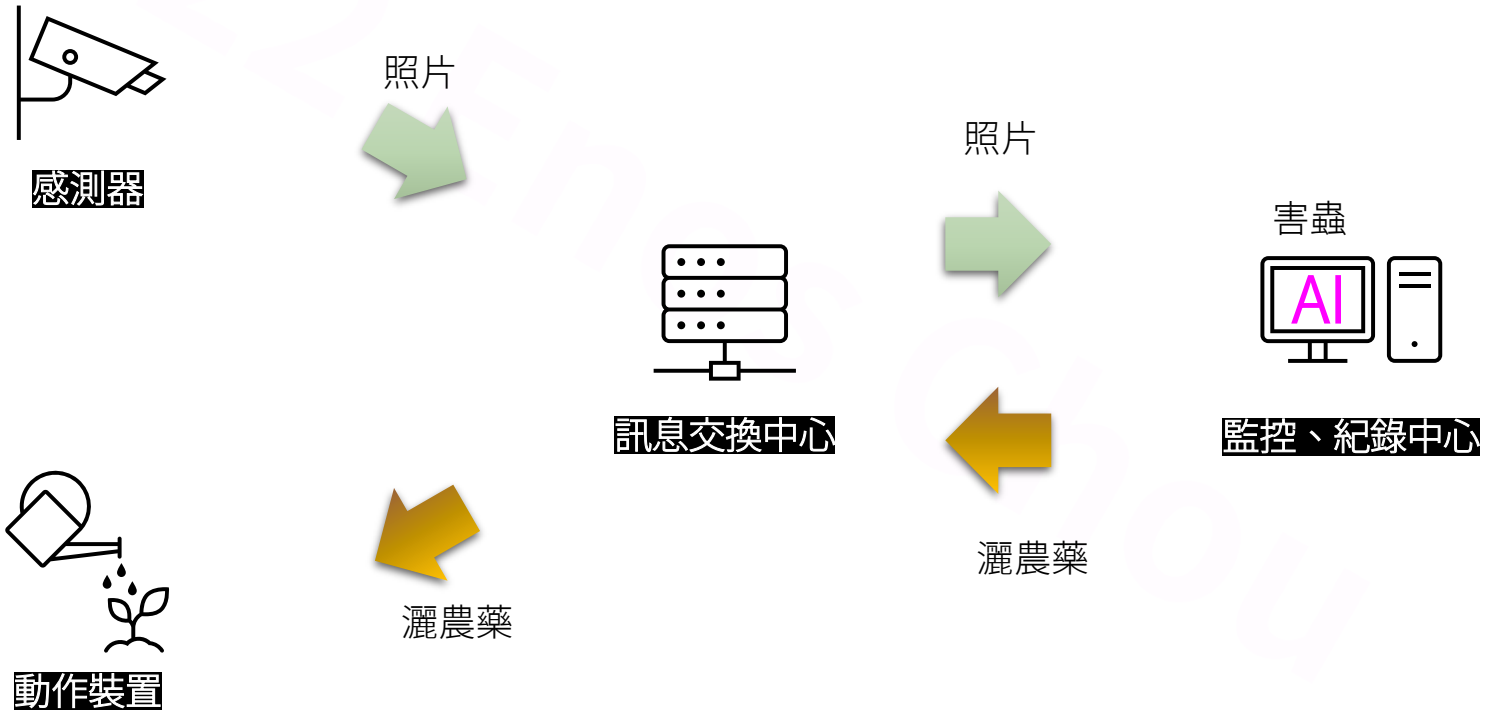
AIoT = AI + IoT

Edge = 邊緣運算裝置

IoT



AIoT without Edge

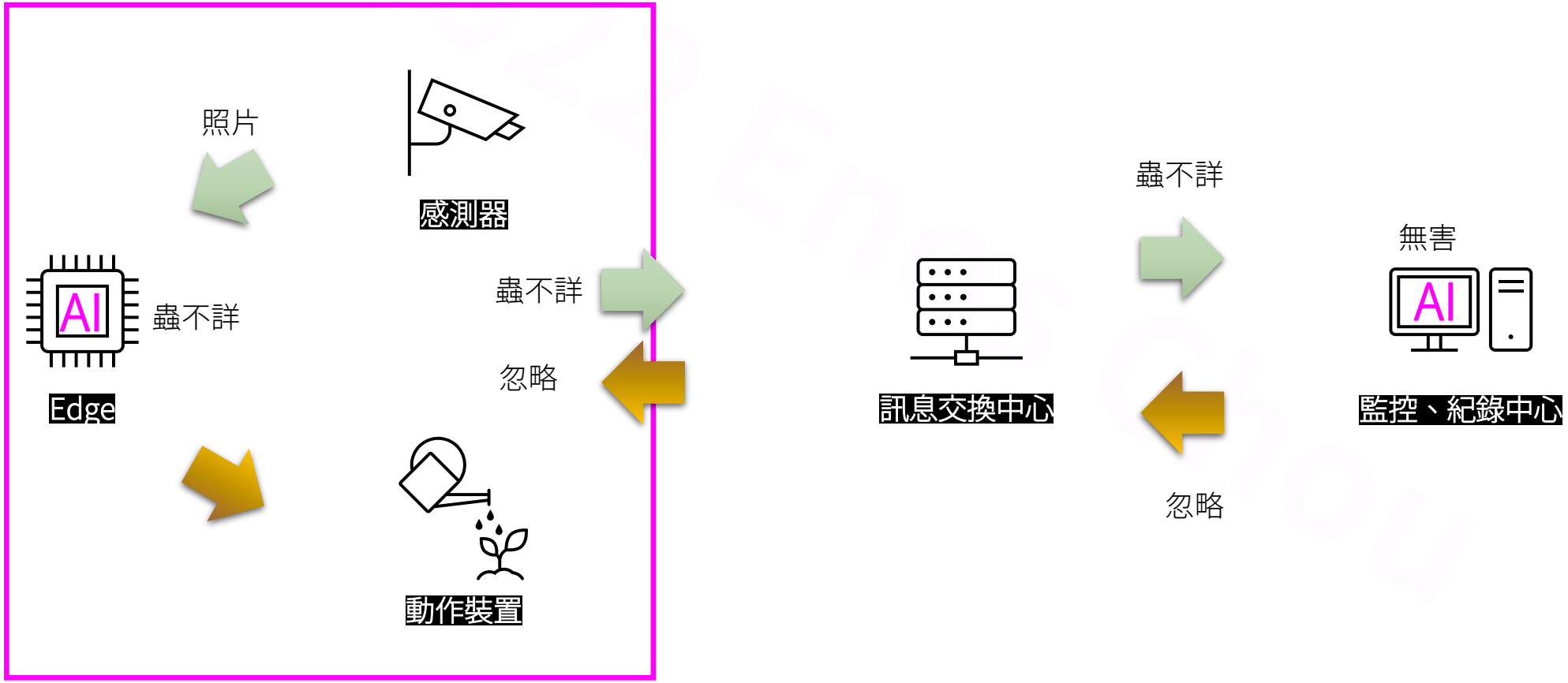


Edge AI

訊息交換中心

監控、紀錄中心

AIoT with Edge



Raspberry Pi Installation

Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

2. **組裝並開機** (on Raspberry Pi)

3. **遠端登入** (on Windows 10)

Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

a. 下載並安裝 SD Memory Card Formatter 5.0.2 for SD/SDHC/SDXC

① https://sdcard.org/ >
Downloads: SD Memory Card Formatter >
For Windows >
Accept

② SDCardFormatterv5_WinEN.zip (6.12MB)

③ 解壓縮後點擊 SD Card Formatter 5.0.2 Setup EN.exe 開始安裝

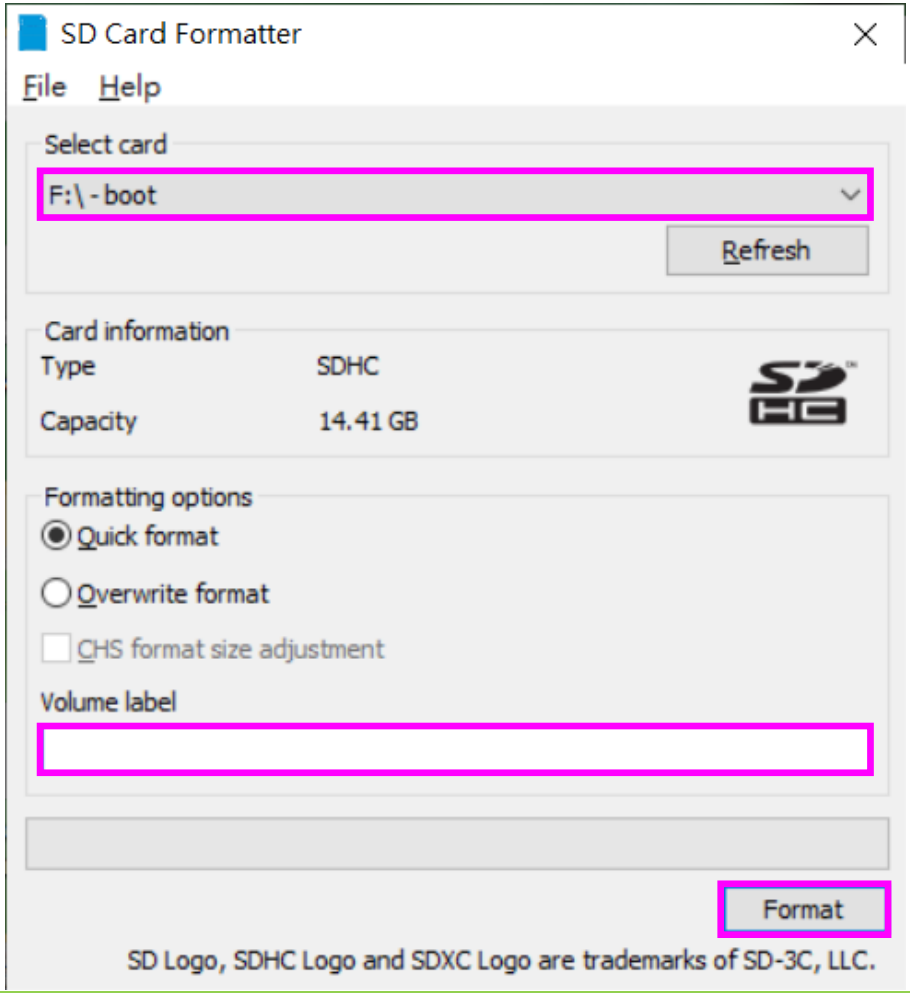
④ 以預設值安裝完畢
Next > > ◉I accept the terms in the license agreement > Next > > Next > > Install > Finish

Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

b. 格式化 microSD

① 插入 microSD
② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format

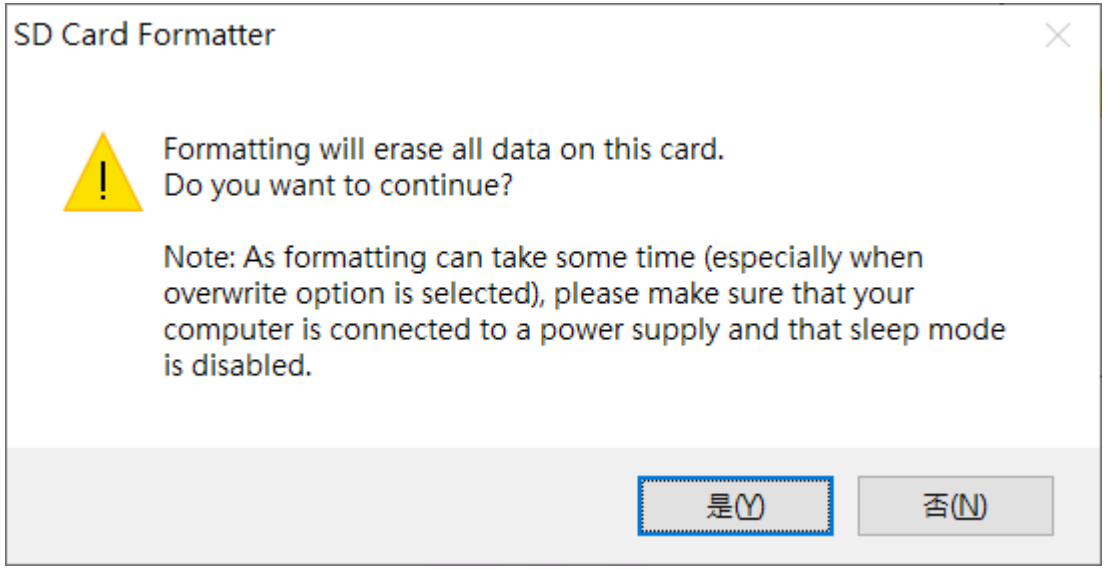


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

b. 格式化 microSD

① 插入 microSD
② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format
③ 是(Y) 開始快速格式化

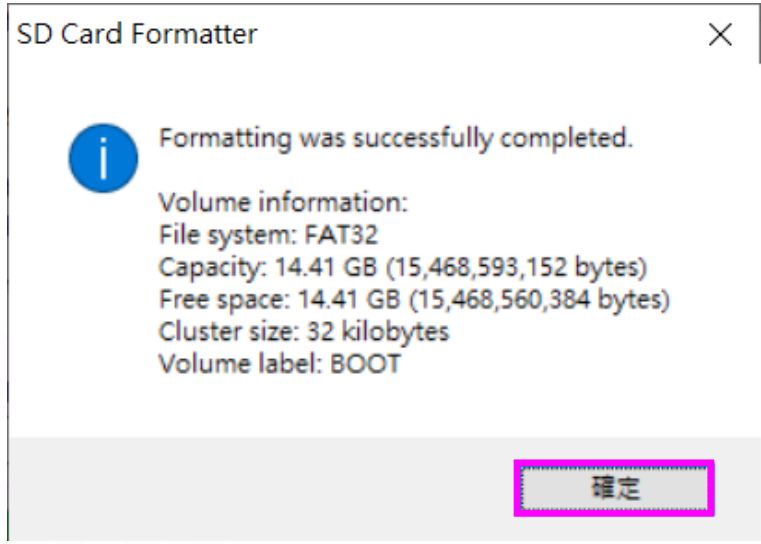


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

b. 格式化 microSD

① 插入 microSD
② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format
③ 是(Y) 開始快速格式化
④ 已完成， 確定



Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

c. 下載 Raspberry Pi OS with Desktop

① https://www.raspberrypi.com >
Software >
(往下捲) Manually install an operating system image: See all download options >
(往下捲) Raspberry Pi OS with desktop (32-bit): Download

② 2022-09-22-raspios-bullseye-armhf.img.xz (893 MB)

Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   d. 以瀏覽器下載 Etcher 並安裝

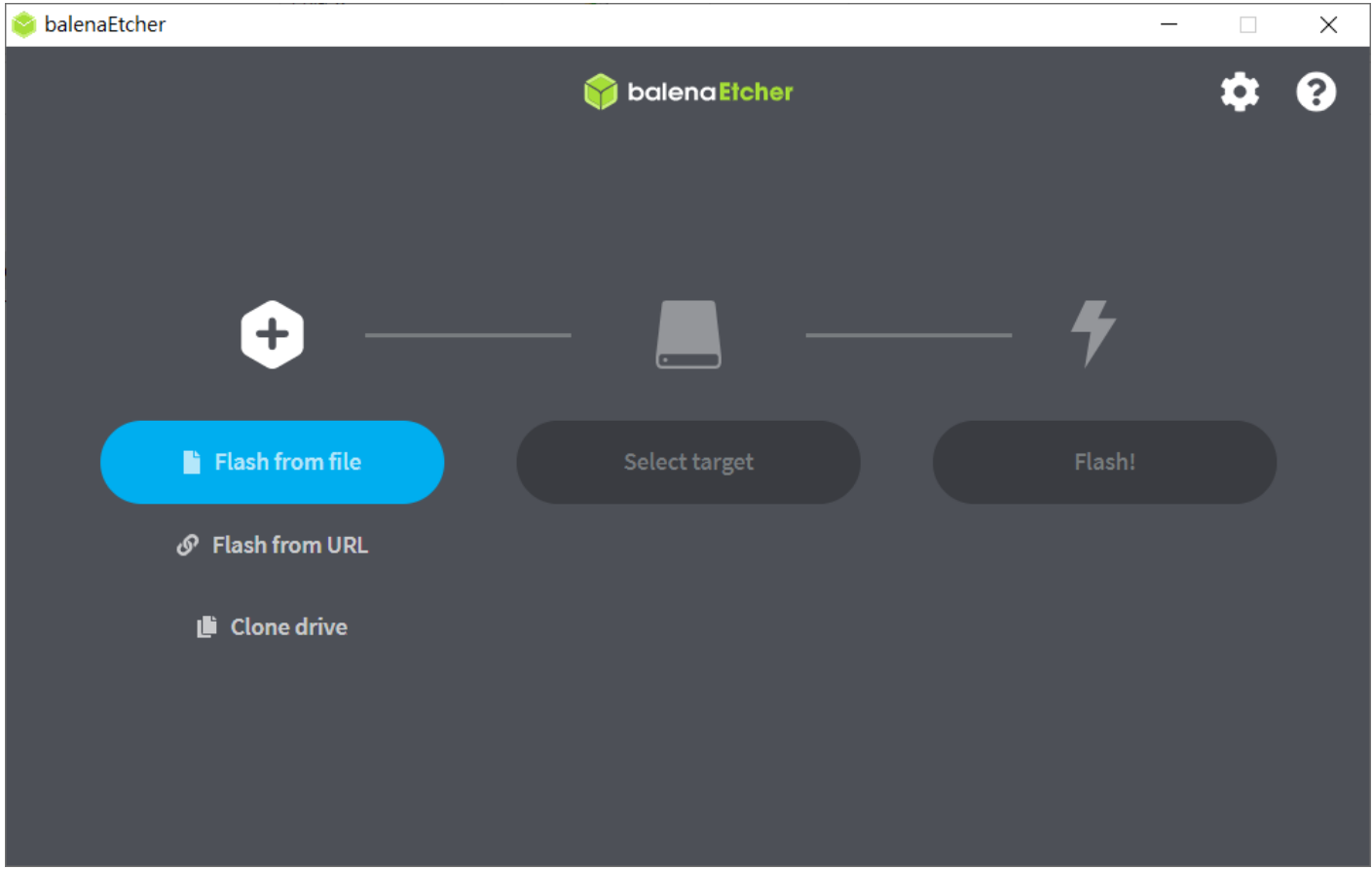
      ① https://www.balena.io/ >
      More products >
      balenaEtcher >
      (往下捲) Get your assets: Etcher for Windows (x86|x64) (Portable) Download
      ② balenaEtcher-Portable-1.7.9.exe (Windows)

Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   e. 燒錄 Raspberry Pi OS

      ① 啟動 Etcher

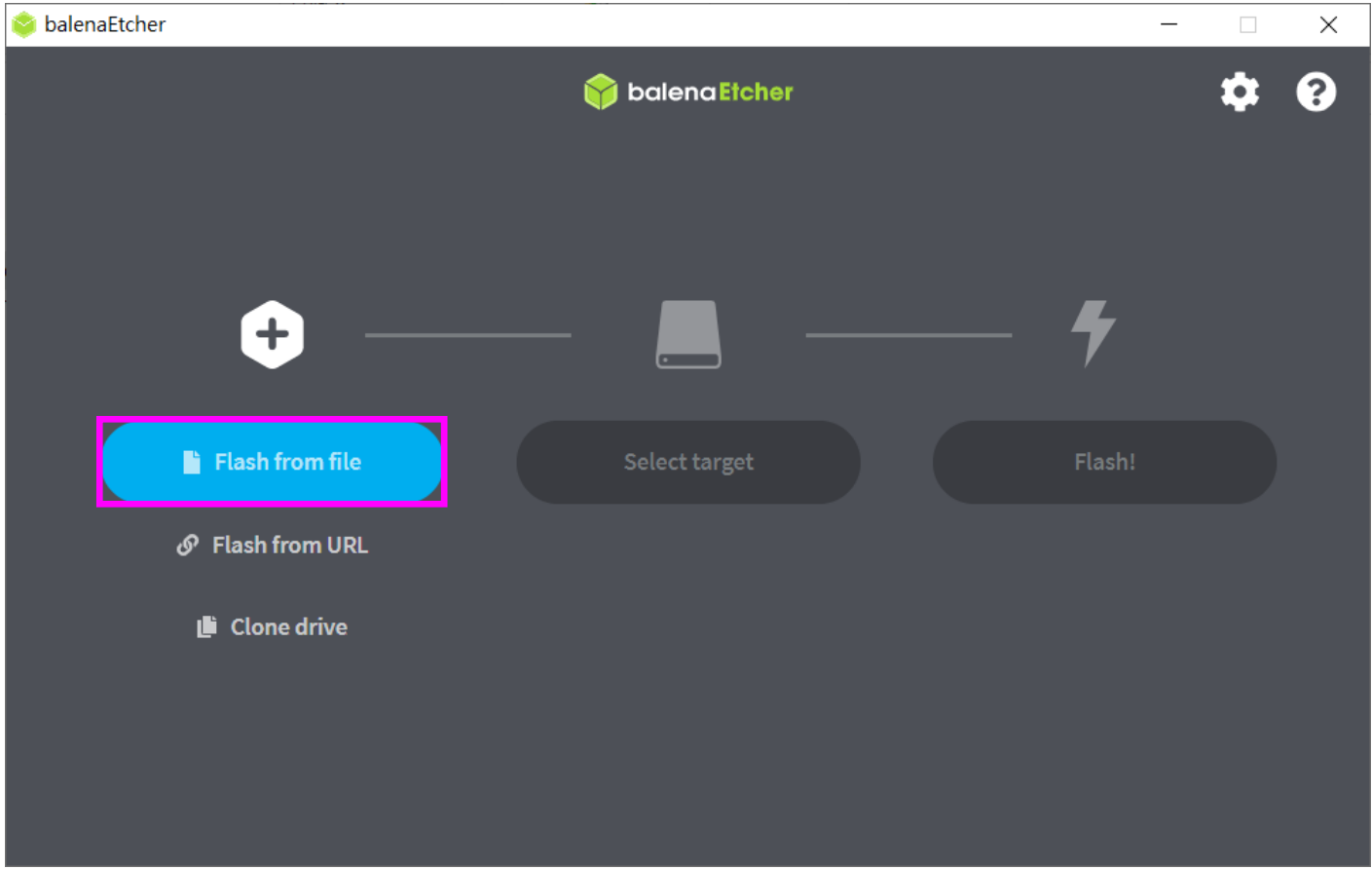


Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   e. 燒錄 Raspberry Pi OS

      ① 啟動 Etcher
      ② Flash from file

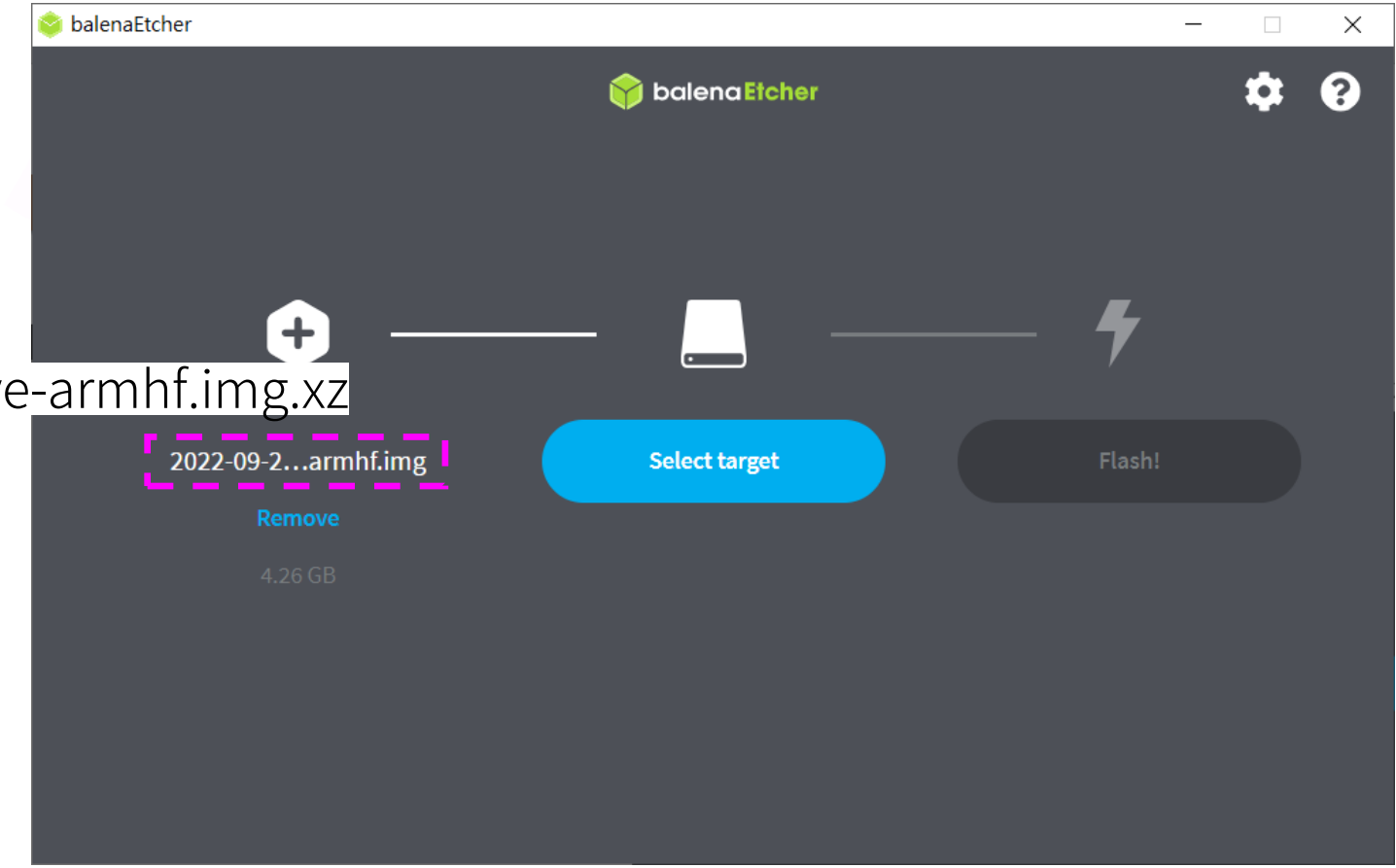


Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   e. 燒錄 Raspberry Pi OS

      ① 啟動 Etcher
      ② Flash from file
      ③ 2022-09-22-raspios-bullseye-armhf.img.xz

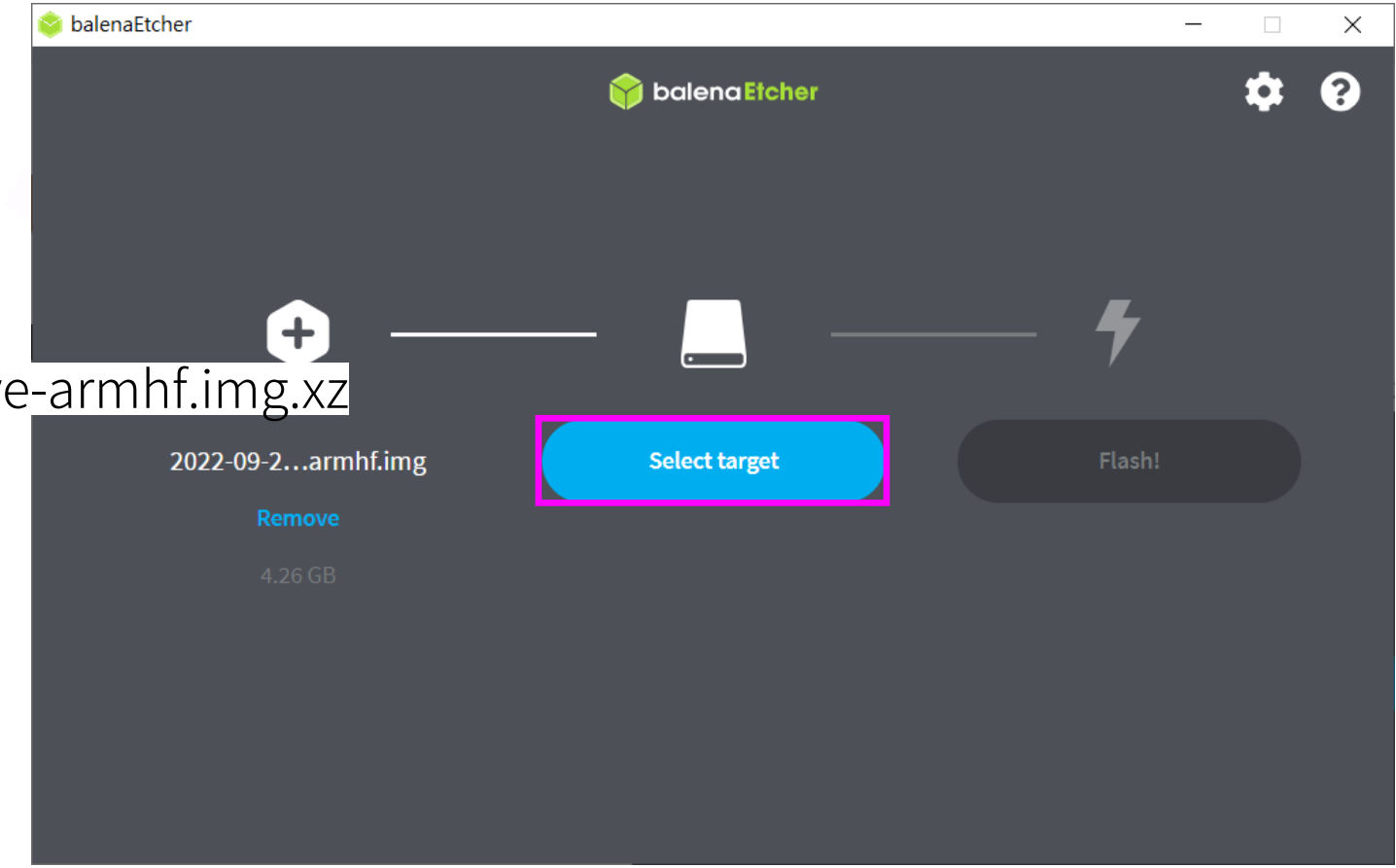


Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   e. 燒錄 Raspberry Pi OS

      ① 啟動 Etcher
      ② Flash from file
      ③ 2022-09-22-raspios-bullseye-armhf.img.xz
      ④ Select target

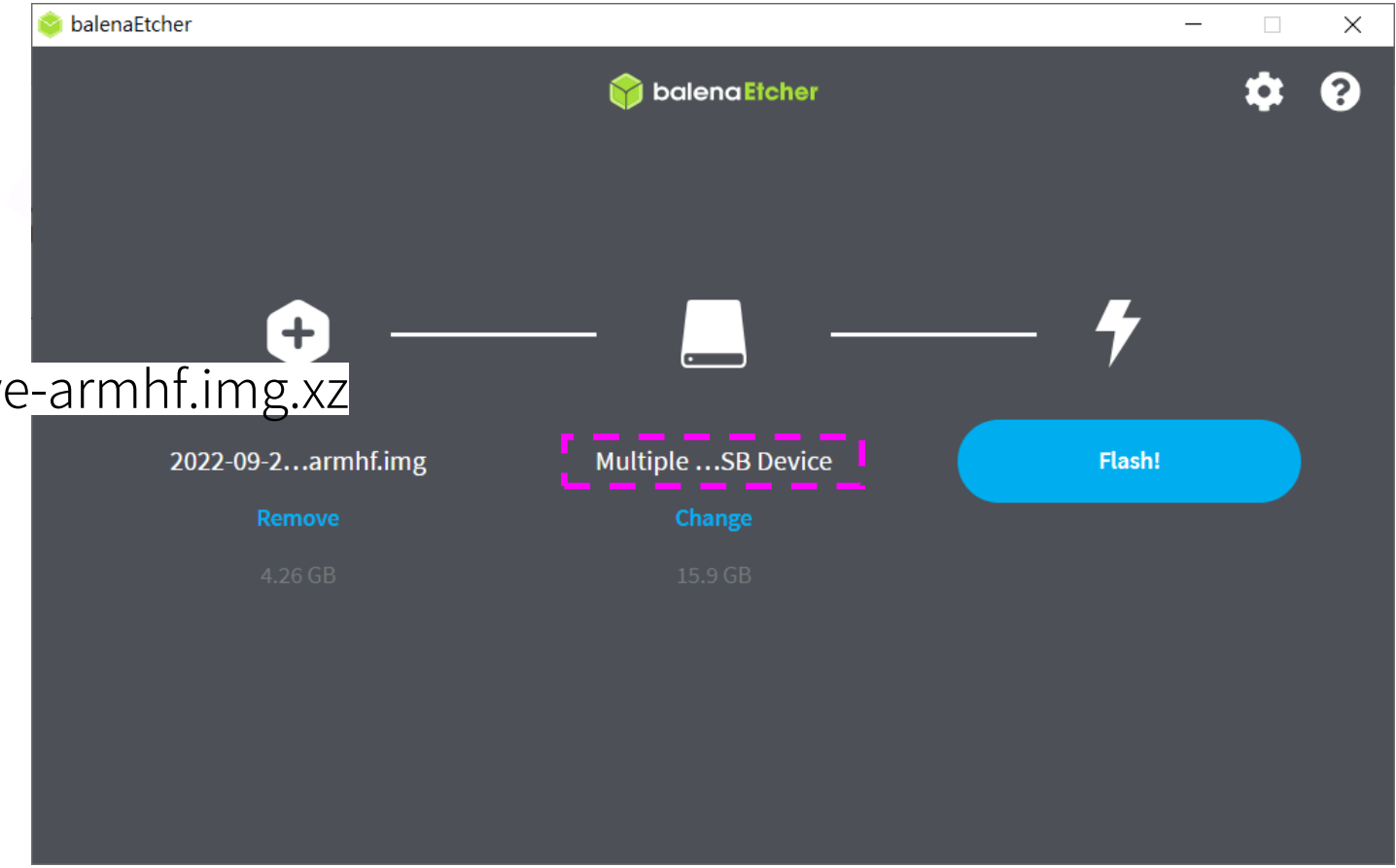


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 Raspberry Pi OS

① 啟動 Etcher
② Flash from file
③ 2022-09-22-raspios-bullseye-armhf.img.xz
④ Select target
⑤ *YOUR_MICROSD_DRIVE*

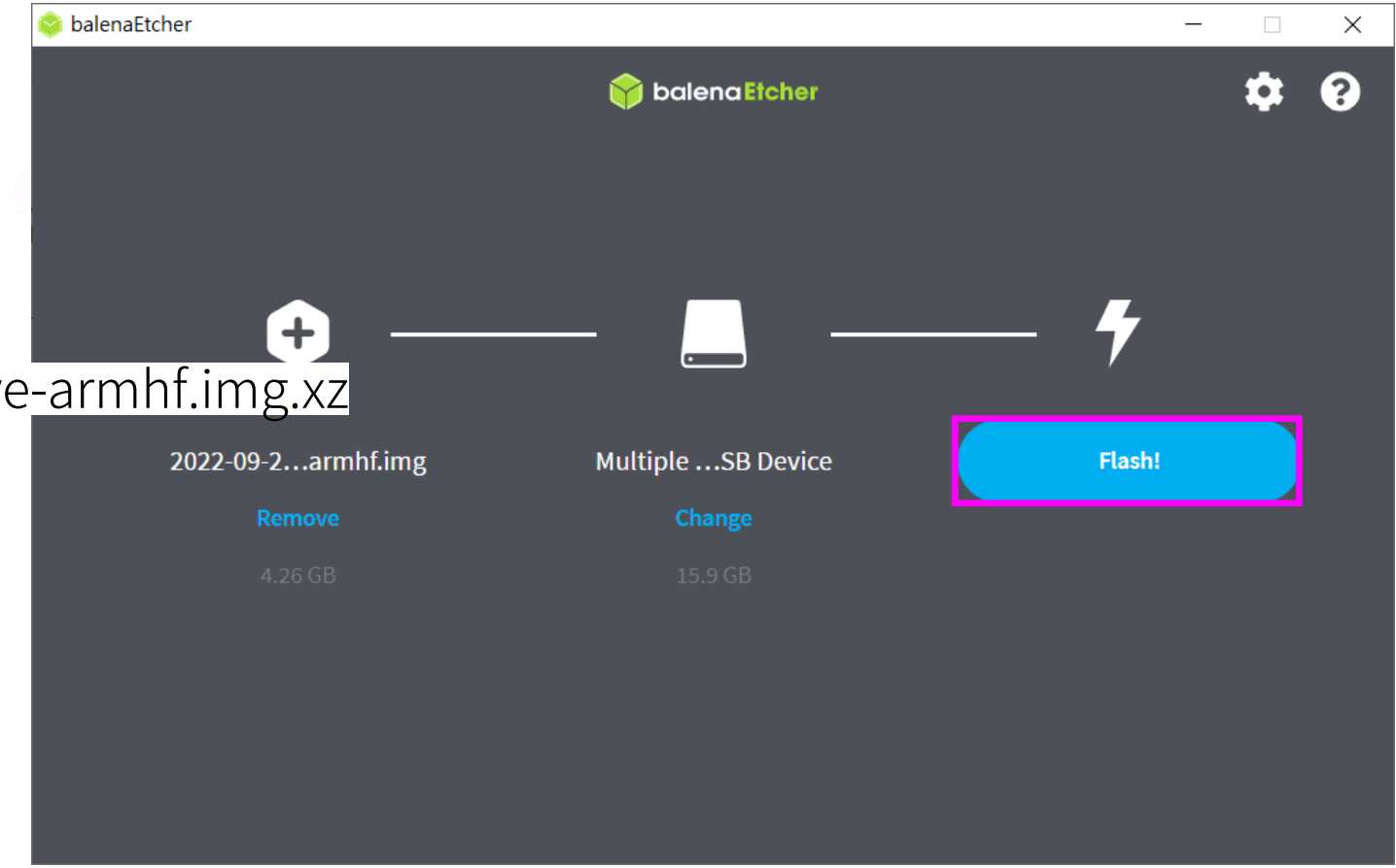


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 Raspberry Pi OS

① 啟動 Etcher
② Flash from file
③ 2022-09-22-raspios-bullseye-armhf.img.xz
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 5 分)

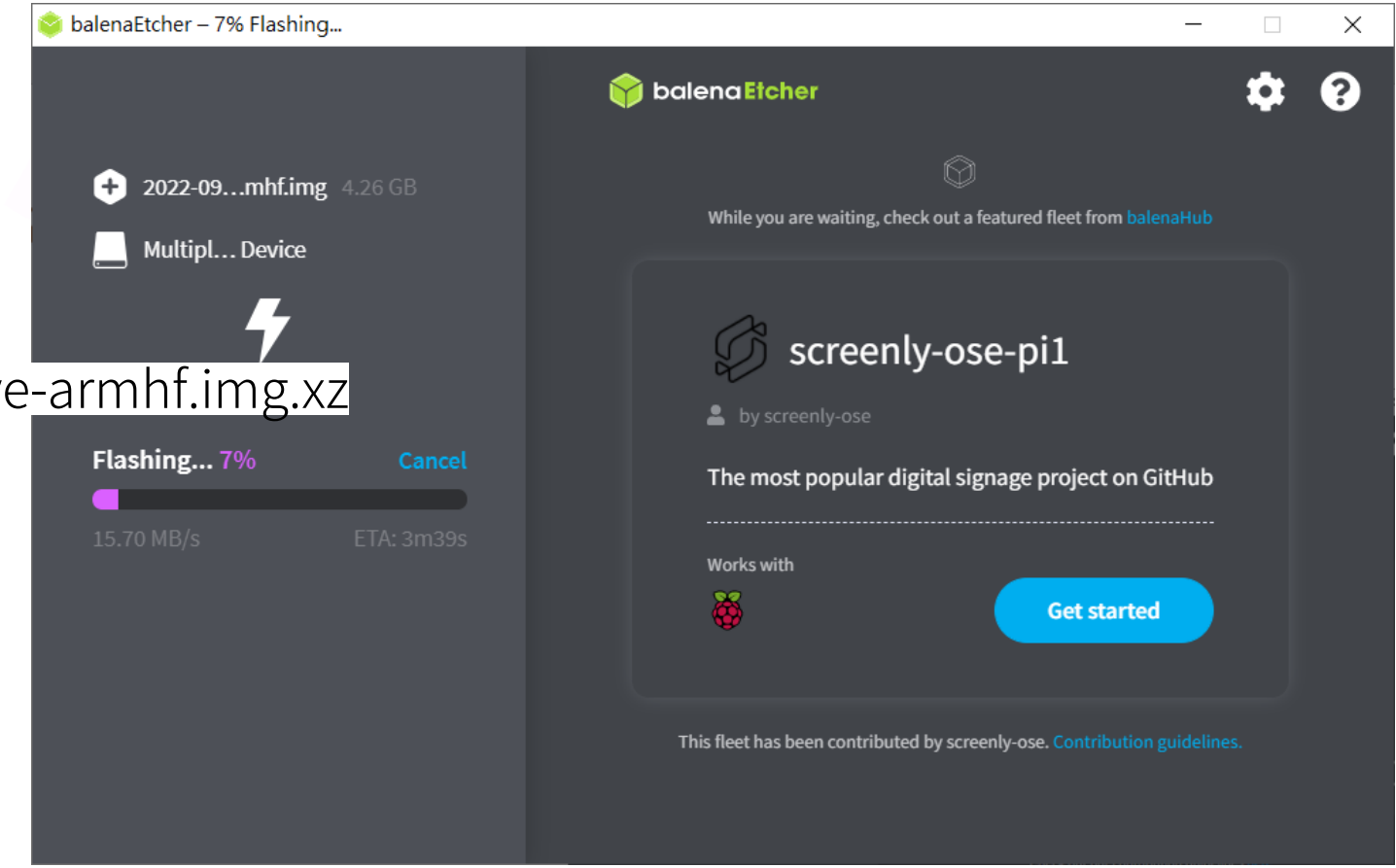


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 Raspberry Pi OS

① 啟動 Etcher
② Flash from file
③ 2022-09-22-raspios-bullseye-armhf.img.xz
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 5 分)

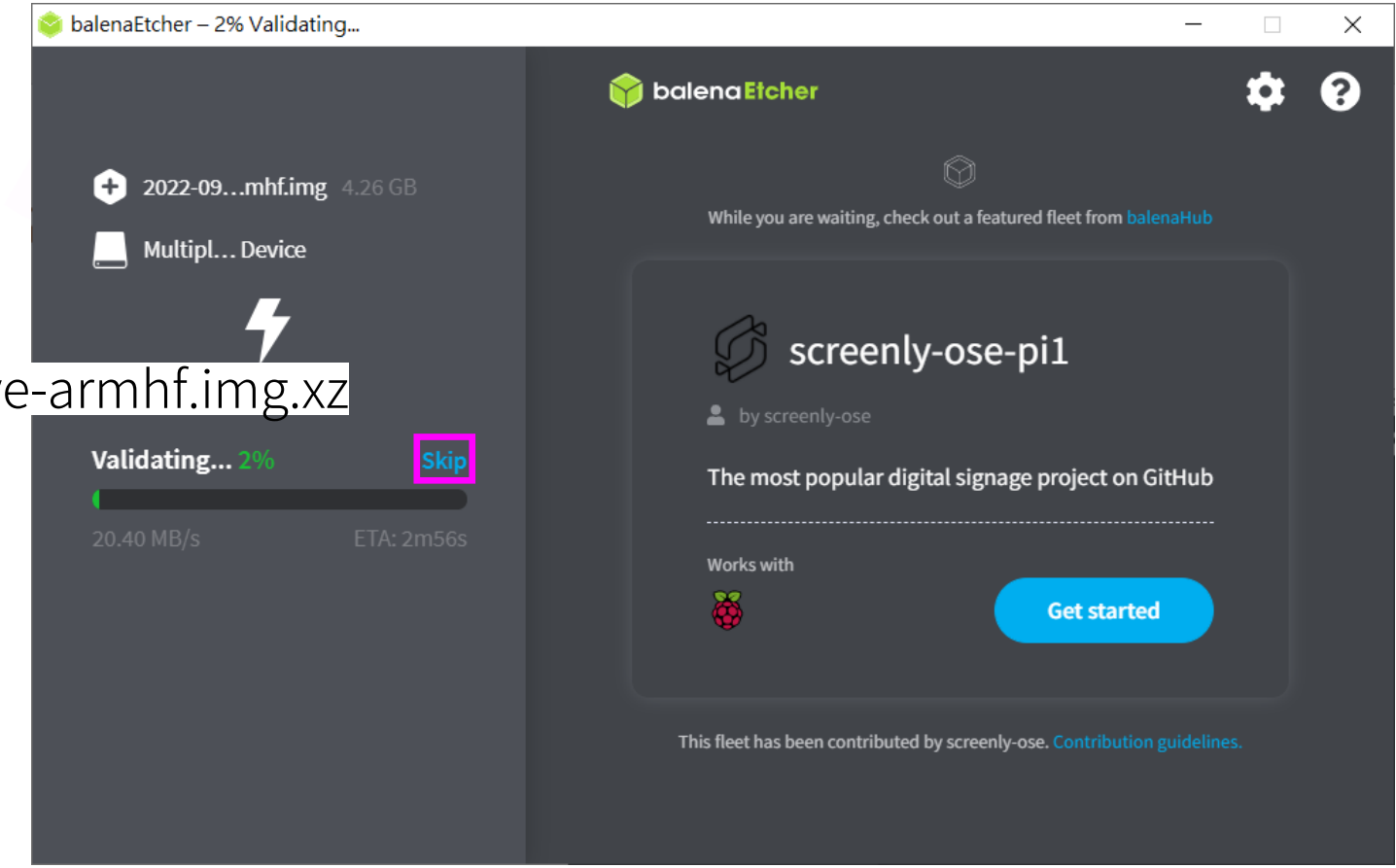


Raspberry Pi 3B+ Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 Raspberry Pi OS

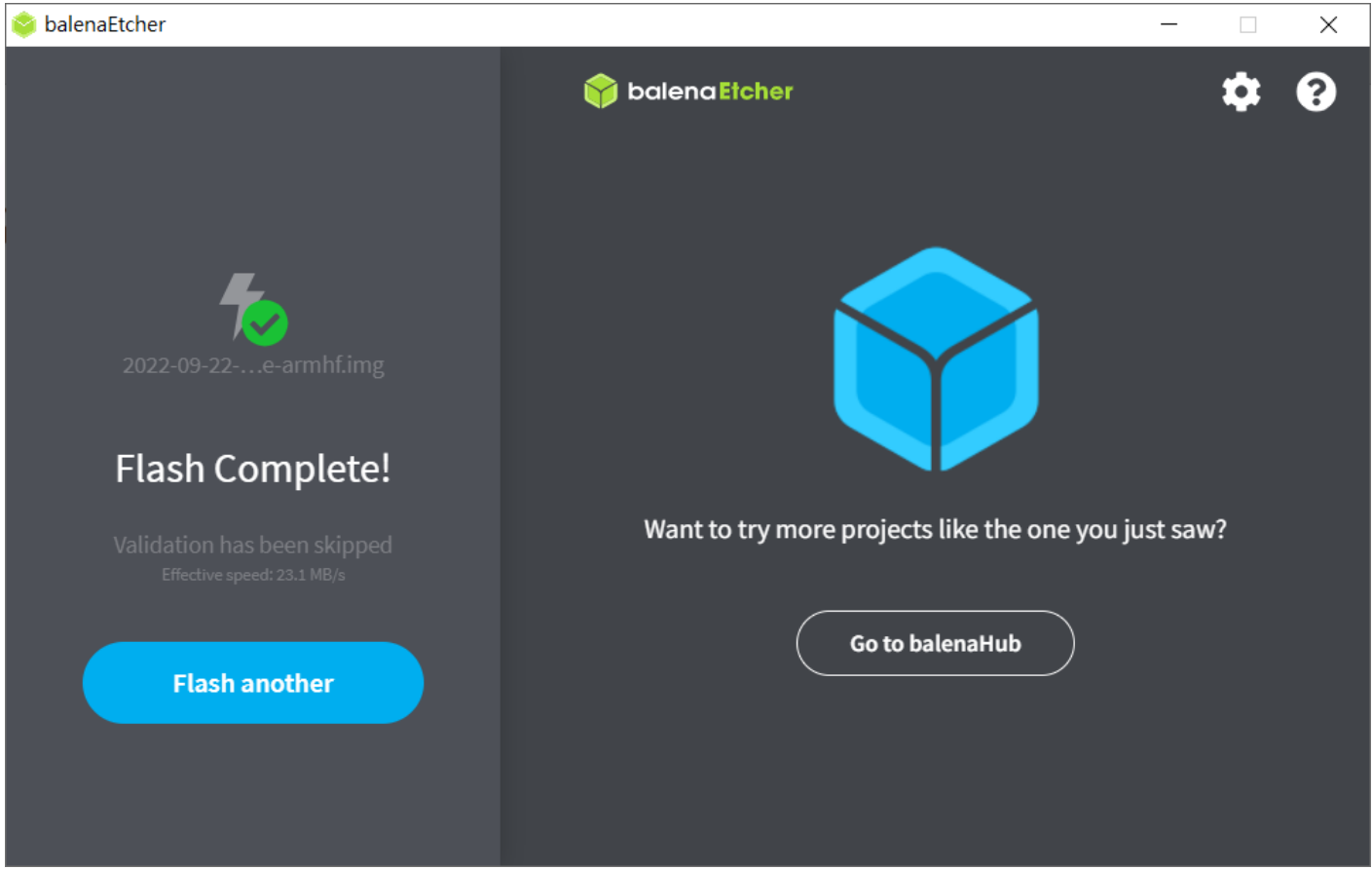
① 啟動 Etcher
② Flash from file
③ 2022-09-22-raspios-bullseye-armhf.img.xz
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 5 分)
⑦ 驗證，Skip



Raspberry Pi 3B+ Installation

1. **製作開機磁碟** (on Windows 10)

   f. 退出 microSD



Raspberry Pi 3B+ Installation

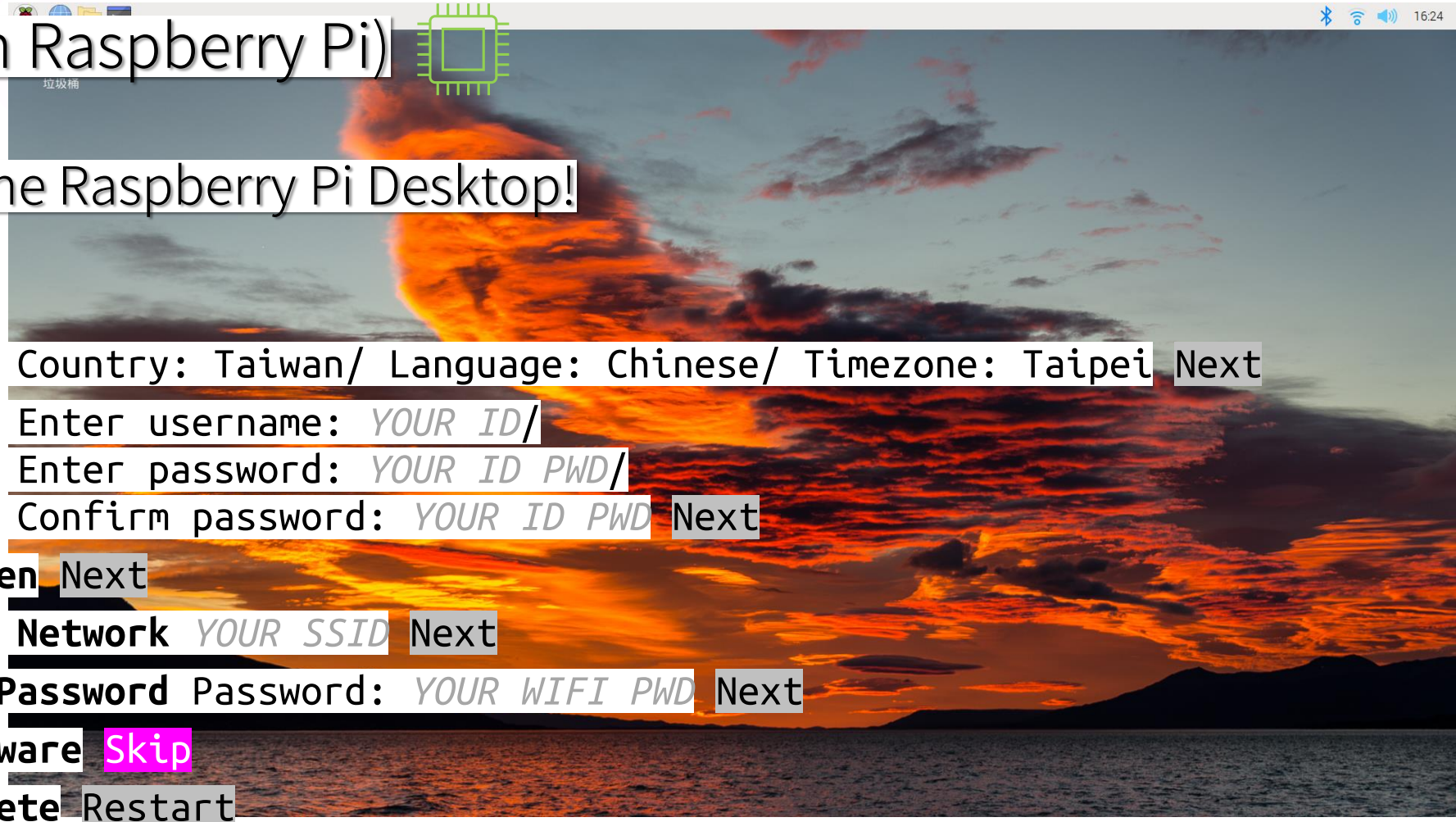
2. 組裝並開機 (on Raspberry Pi)

a. 硬體組裝

① HDMI 螢幕
② USB-A 鍵盤
③ USB-A 滑鼠
④ microSD
⑤ Micro-USB Power Adaptor 5V2.5A

Raspberry Pi 3B+ Installation

2. 組裝並開機 (on Raspberry Pi)



b. Welcome to the Raspberry Pi Desktop!

① Next

② **Set Country** Country: Taiwan/ Language: Chinese/ Timezone: Taipei Next

③ **Create User** Enter username: *YOUR ID*/
Enter password: *YOUR ID PWD*/
Confirm password: *YOUR ID PWD* Next

④ **Set Up Screen** Next

⑤ **Select WiFi Network** *YOUR SSID* Next

⑥ **Enter WiFi Password** Password: *YOUR WIFI PWD* Next

⑦ **Update Software** Skip

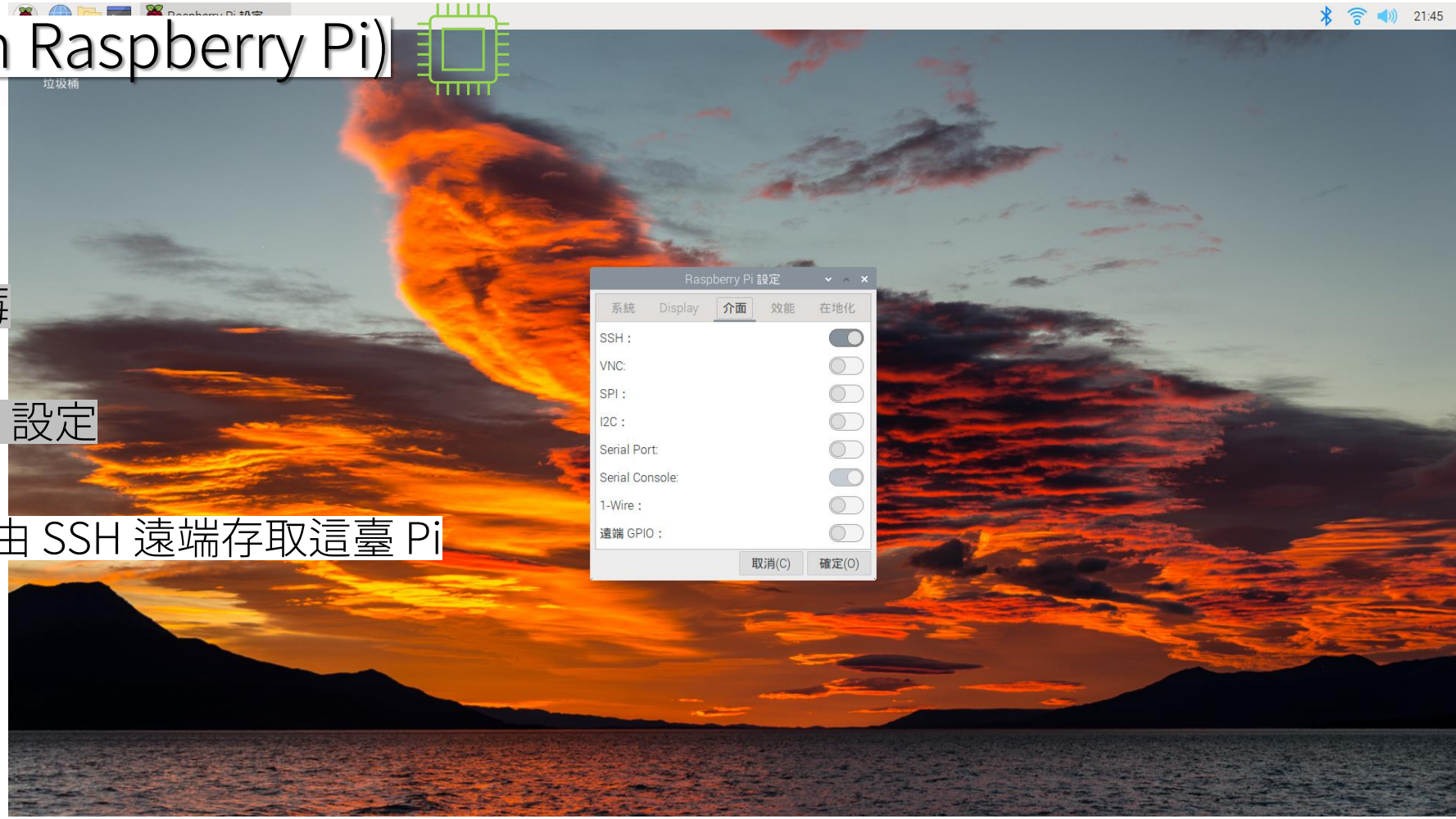
⑧ **Setup Complete** Restart

Raspberry Pi 3B+ Installation

2. 組裝並開機 (on Raspberry Pi)

c. 啟動遠端登入

① (左上角) 樹莓
② 偏好設定
③ Raspberry Pi 設定
④ 介面
⑤ SSH: 啟用經由 SSH 遠端存取這臺 Pi
⑥ 確定(O)

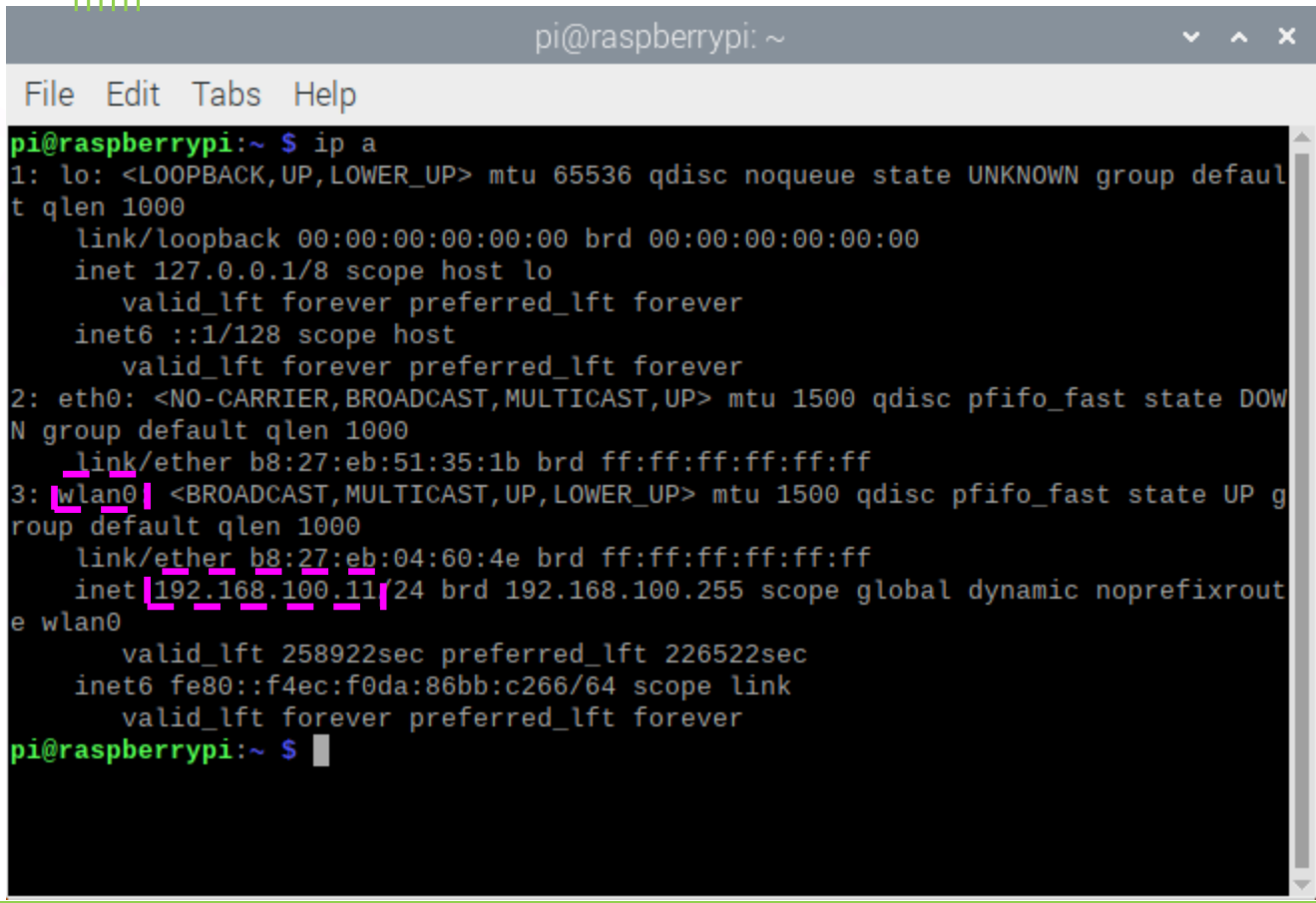


Raspberry Pi 3B+ Installation

2. 組裝並開機 (on Raspberry Pi)

d. 取得 IP 位址

① 開啟 LX 終端機
② `ip a` ENTER
③ 取得 wlan0 介面之 IP 位址



Raspberry Pi 3B+ Installation

3. 遠端登入 (on Raspberry feat. Windows 10) 

a. 下載並安裝 MobaXterm Home Edition v22.1 (Portable Edition)

① https://mobaxterm.mobatek.net/ >
Download >
Home Edition: Download now >
MobaXterm Home Edition v22.1 (Portable edition)

② MobaXterm_Portable_v22.1.zip

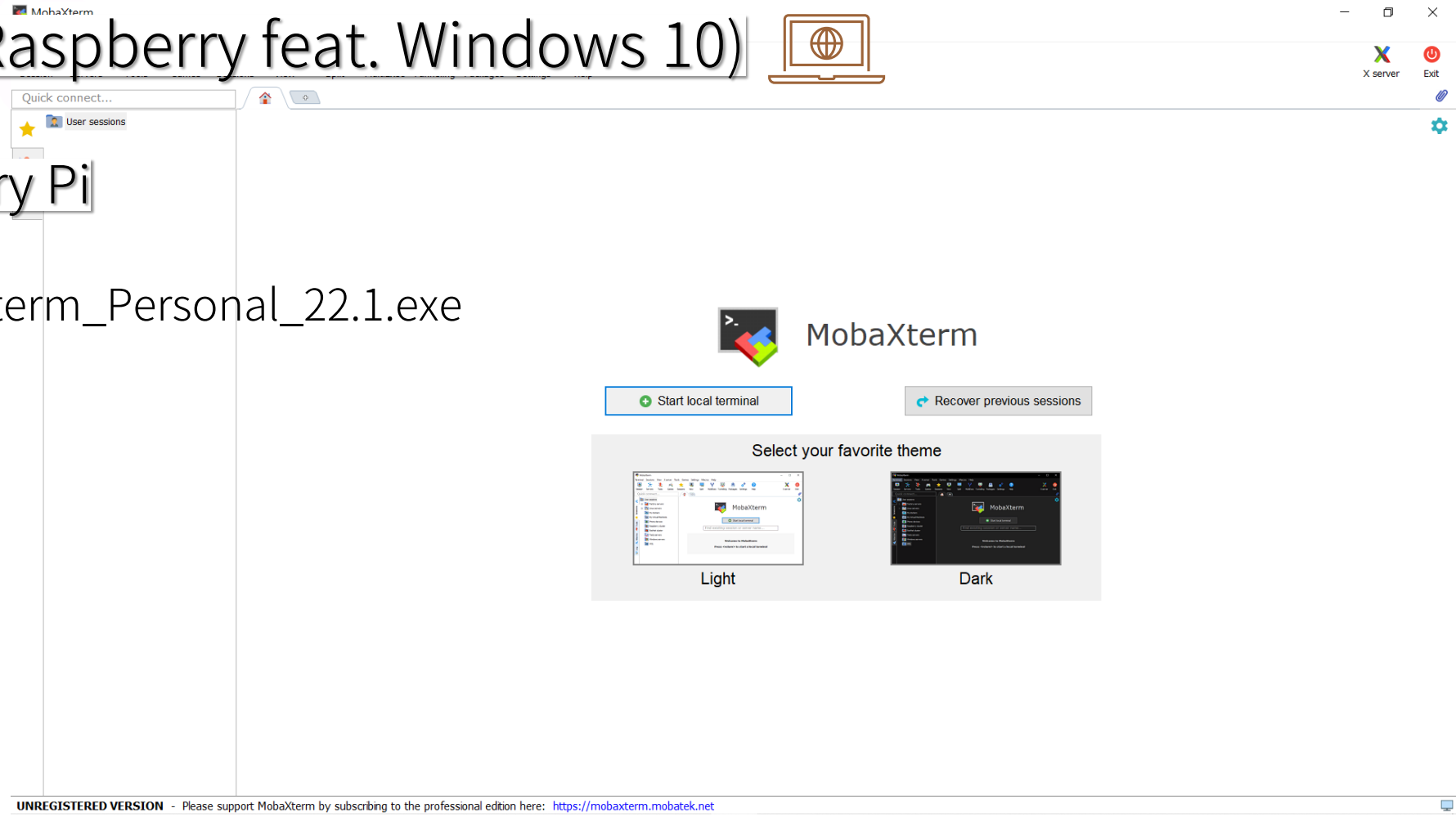
③ 解壓縮 MobaXterm_Portable_v22.1.zip ⇨ MobaXterm_Personal_22.1.exe

Raspberry Pi 3B+ Installation

3. 遠端登入 (on Raspberry feat. Windows 10)

b. 連入 Raspberry Pi

① 啟動 MobaXterm_Personal_22.1.exe



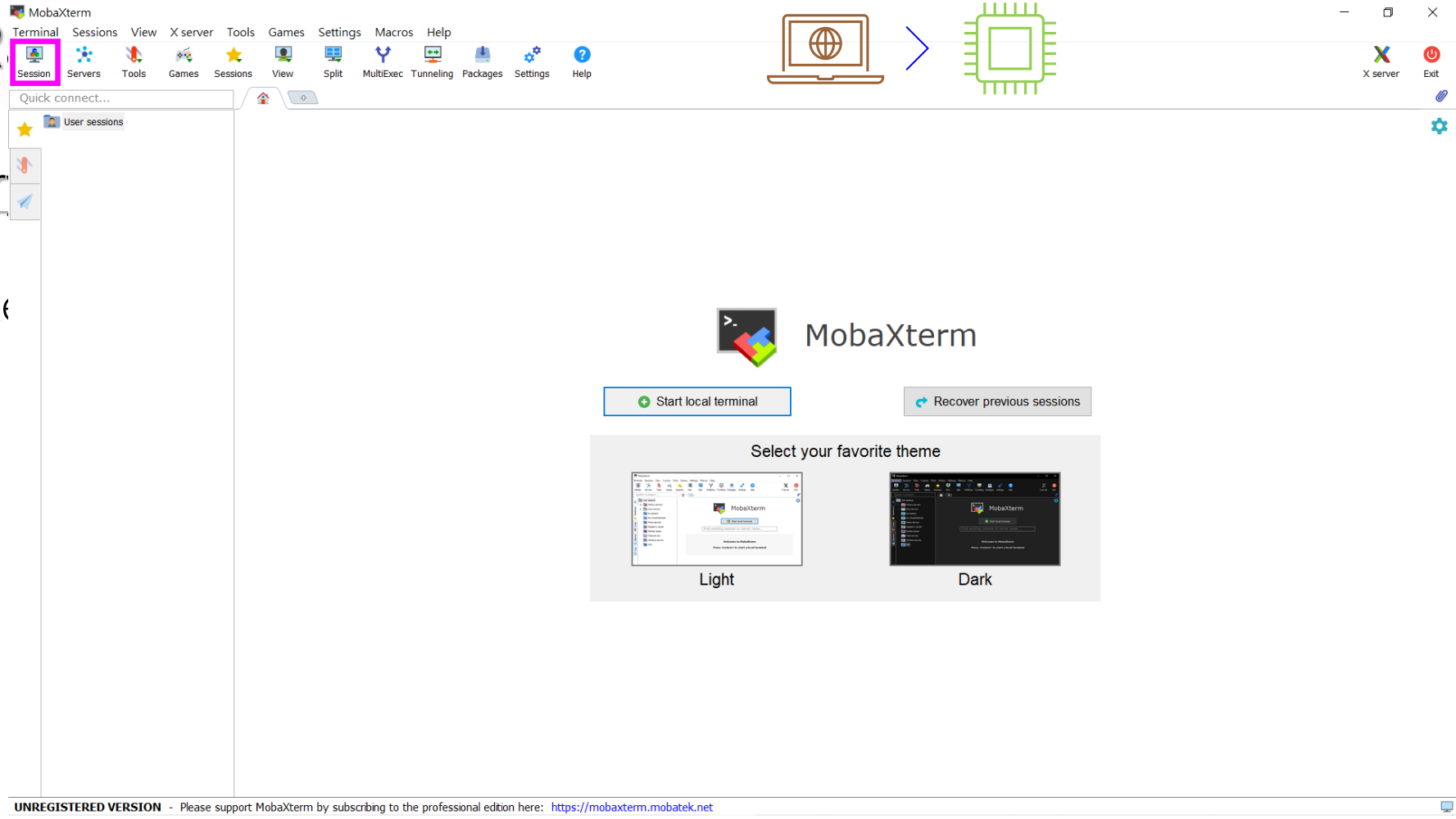
Raspberry Pi 3B+ Installation

3. 遠端登入 (on R

b. 連入 Raspberr

① 啟動 MobaXt

② Session

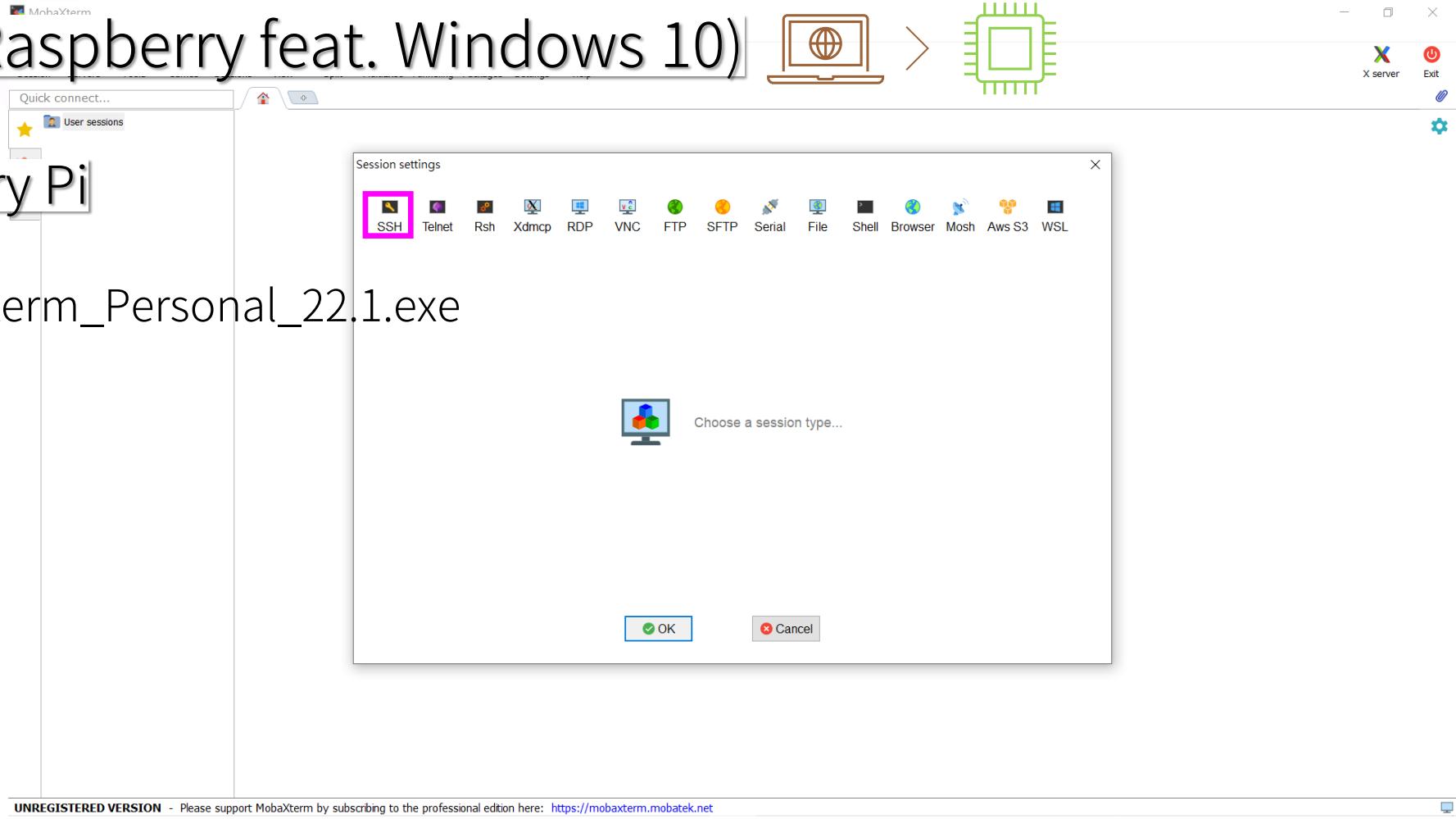


Raspberry Pi 3B+ Installation

3. 遠端登入 (on Raspberry feat. Windows 10)

b. 連入 Raspberry Pi

① 啟動 MobaXterm_Personal_22.1.exe
② Session
③ SSH



Raspberry Pi 3B+ Installation

3. 遠端登入 (on Raspberry feat. Windows 10)

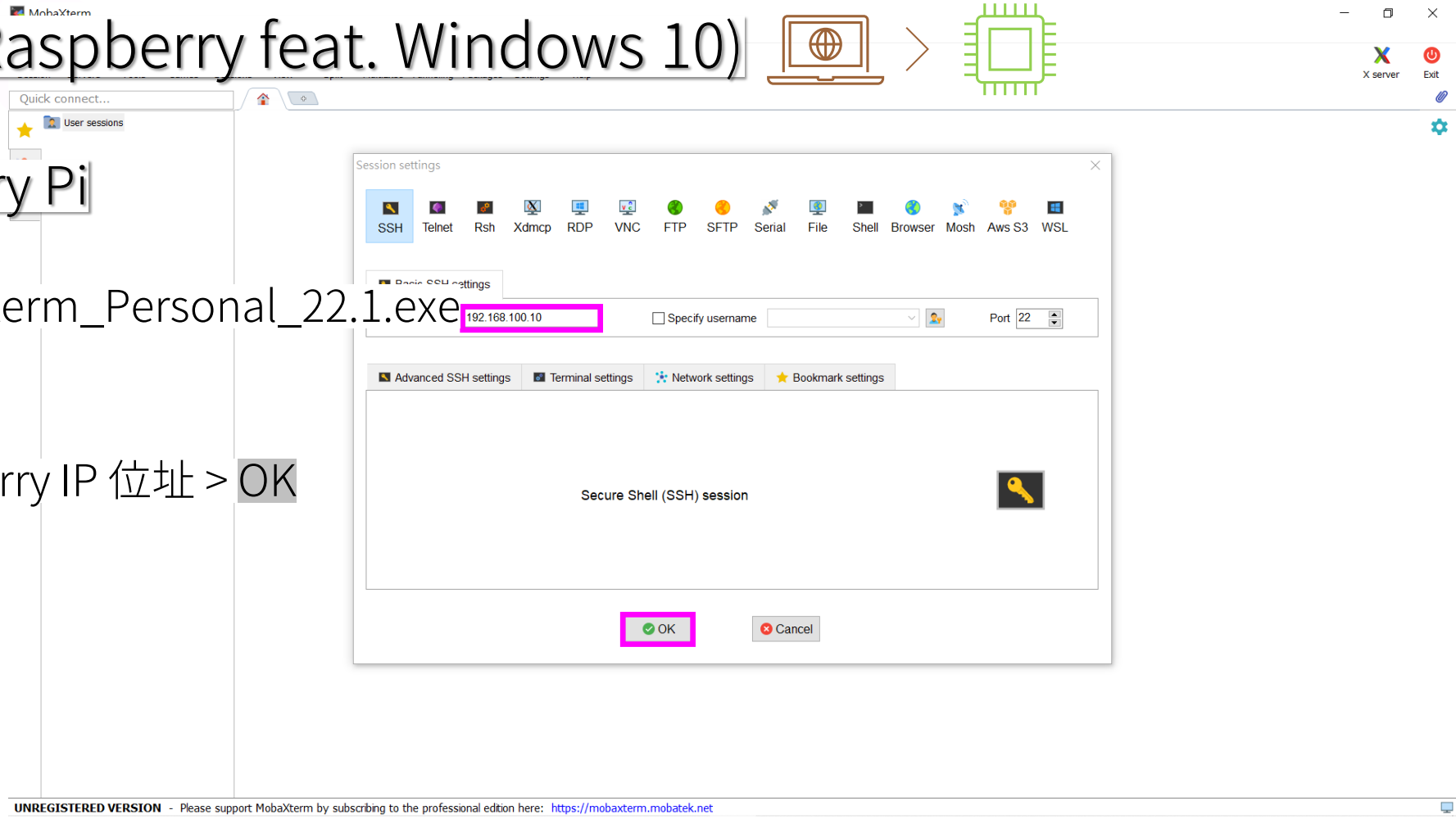
b. 連入 Raspberry Pi

① 啟動 MobaXterm_Personal_22.1.exe

② Session

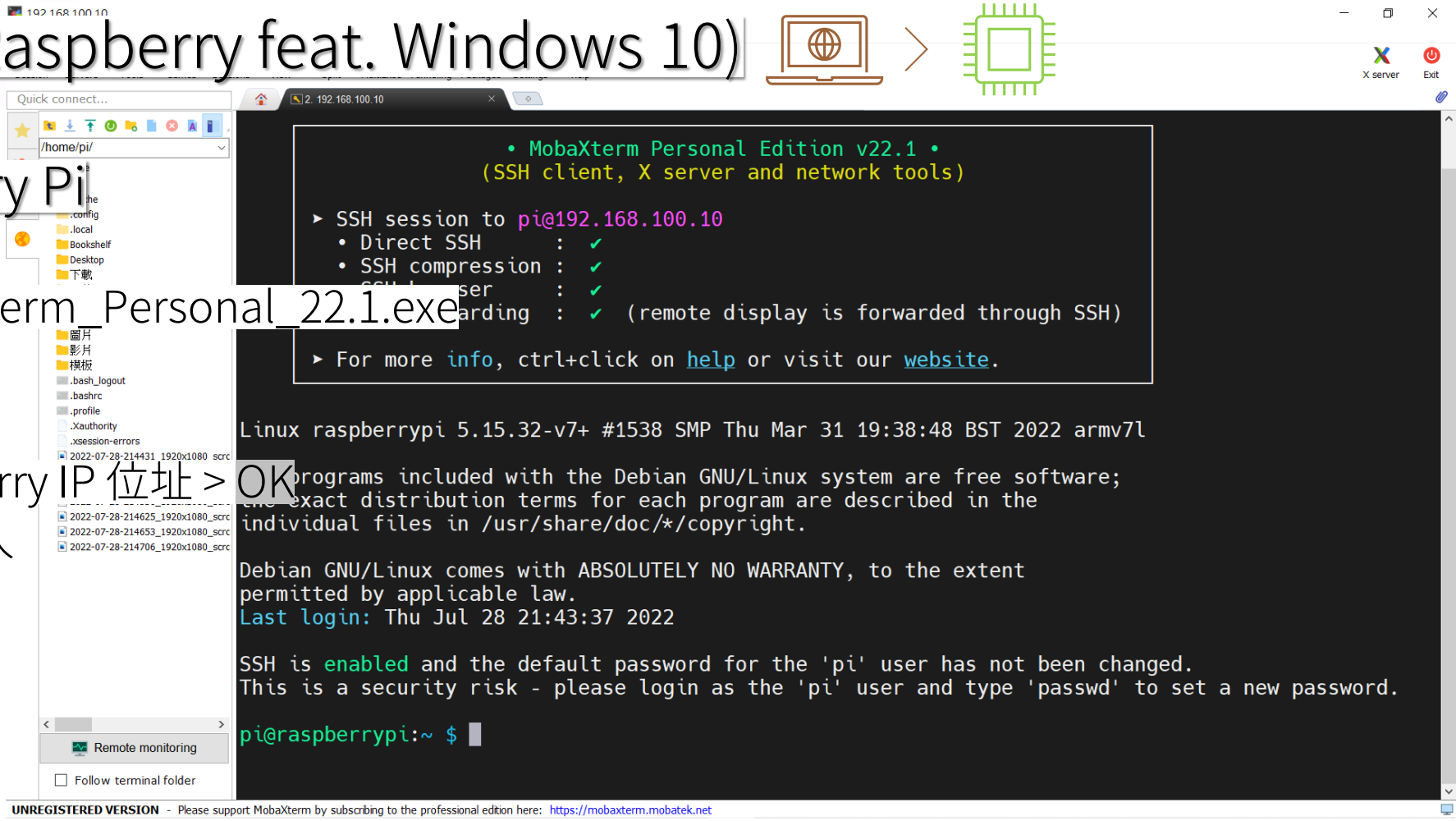
③ SSH

④ 填入 Raspberry IP 位址 > OK



Raspberry Pi 3B+ Installation

3. 遠端登入 (on Raspberry feat. Windows 10)



b. 連入 Raspberry Pi

① 啟動 MobaXterm_Personal_22.1.exe
② Session
③ SSH
④ 填入 Raspberry IP 位址 > OK
⑤ 帳號密碼登入

Raspberry Faces

Raspberry Faces

1. 安裝套件 (on Raspberry Pi)
2. 註冊與點名 (on Raspberry Pi)
3. 物連網的註冊與點名

Raspberry Faces

1. 安裝套件 (on Raspberry Pi) 

a. 套件 dlib & face_recognition

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② 安裝 (約 2 分)

```
# targets: face_recognition (1.3.0)
# dependencies: Pillow (built-in 8.1.2), dlib>=19.7 (19.24.0), numpy
(built-in 8.1.2)
sudo apt-get update; python3 -m pip install --no-warn-script-location
face_recognition
```

Raspberry Faces

1. 安裝套件 (on Raspberry Pi) 

b. 點名應用 rollcall_edge

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② 安裝 (約 2 分)

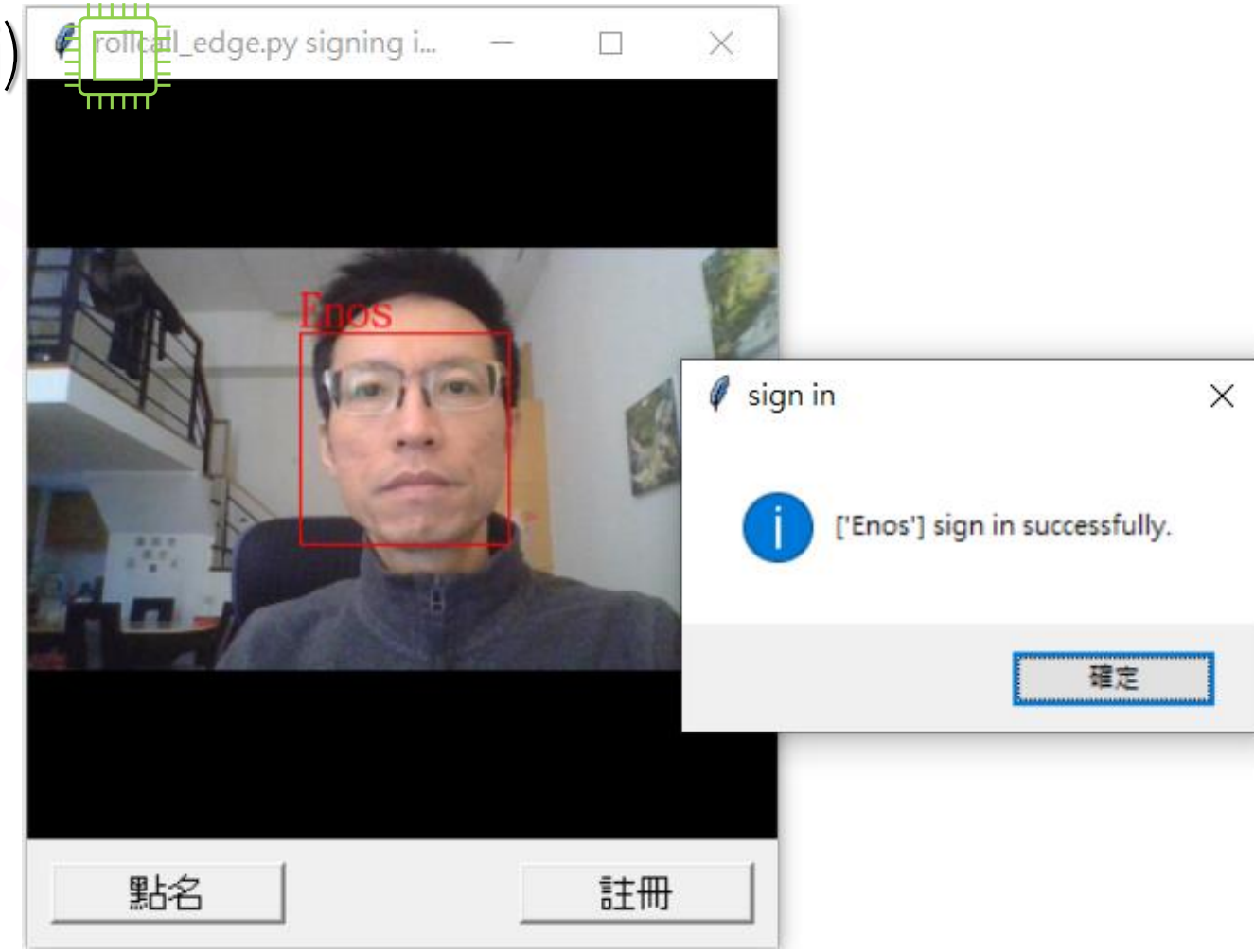
```
# targets: pil.imgetk, opencv (4.5.1), paho-mqtt (1.5.1), rollcall_edge
sudo apt-get update; sudo apt-get install -y libatlas-base-dev python3-
pil.imagetk python3-opencv python3-paho-mqtt; cd; git clone
https://github.com/catchsob/facecam; wget --no-check-certificate
'https://drive.google.com/u/0/uc?id=1eGAsTN1HBpJAkeVM57_C7ccp7hbgSz3_&e
xport=download' -O facecam/TaipeiSansTCBeta-Regular.ttf
```

Raspberry Faces

2. 註冊與點名 (on Raspberry Pi)

a. 啟動 rollcall_edge

① 開啟 LX 終端機
② 進入目錄
```
cd; cd facecam
```
③ 執行
```
python3 rollcall_edge.py
```
④ 先 註冊，再 點名



Raspberry Faces

2. 註冊與點名 (on Raspberry Pi) 

b. Trace rollcall_edge.py

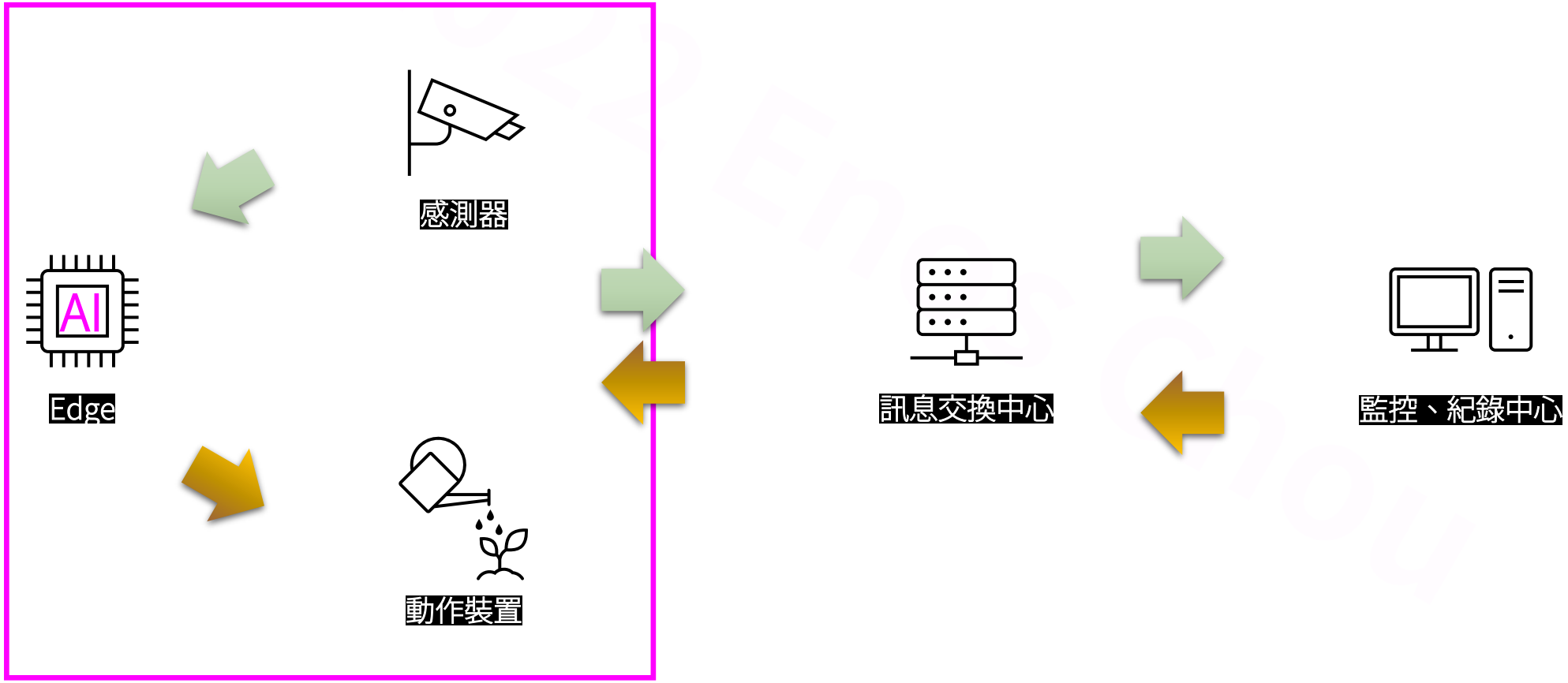
face_recognition

```
ens = face_recognition.face_encodings(face)  # encode the faces
locs = face_recognition.face_locations(face)  # get the face frame
distance = face_recognition.face_distance(faces.encodes, ens[e])  # compute the
distance with each face
facei = distance.argmin()  # select the most likely face
```

Raspberry Faces

3. 物連網的註冊與點名

Recap AIoT with Edge



Faces AIoT

Raspi with rollcall_edge
MQTT Publisher

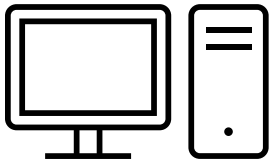
MQTT publish

GCE

mosquitto

MQTT Broker

MQTT subscribe



Application
MQTT Subscriber

MQTT subscribe

Node-RED

Management Panel
MQTT Subscriber

Raspi with rollcall_edge
MQTT Publisher

MQTT publish

Faces AIoT

MQTT Topic Example

- `rollcall/signin`
- `rollcall/register`
- `rollcall/#`
- `#`

MQTT Message Example

- `Someone on 2022/02/15`

AIoT with MQTT

1. Server as MQTT Broker
2. Server as MQTT Subscriber
3. Client as MQTT Publisher

AIoT with MQTT

1. Server as MQTT Broker
    a. mqttgo.io

AIoT with MQTT

1. Server as MQTT Broker

b. Ubuntu 18.04 on GCE

① 登入 console.cloud.google.com
② 建立並切換至新專案
③ 啟用 Compute Engine
④ Compute Engine > VM 執行個體 > 建立執行個體
- 名稱 mqtt
- 區域 us-west1/ 區域 us-west1-b
- 機器類型 e2-micro
- 開機磁碟 > 作業系統 Ubuntu/ 版本 Ubuntu 18.04 LTS x86/64
- 進階選項 > 網路 > 網路介面 > 外部 IPv4 位址 > 建立 IP 位址 > mqttip

AIoT with MQTT

1. Server as MQTT Broker

b. Ubuntu 18.04 on GCE

⑤ 虛擬私有雲網路 > 防火牆 > 建立防火牆規則
- 名稱 mqttfw
- 目標 網路中的所有執行個體
- 來源 IPv4 範圍 0.0.0.0/0
- 通訊協定和埠 > 指定的通訊協定和埠 > TCP 1883

⑥ (optional) DNS 指向外部 IP 位址 (mqttip)

⑦ SSH 連線 GCE VM 執行個體 mqtt

```
sudo apt-get update; sudo apt-get install -y mosquitto; sudo
timedatectl set-timezone Asia/Taipei
```

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

套件 paho-mqtt

```
pip install paho-mqtt
```

AIoT with MQTT

2. Server as MQTT Subscriber

Python 3 Environment

官方範例

https://github.com/eclipse/ > Repositories: paho.mqtt.python

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("rollcall/#")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
```

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

```
YOUR_BROKER = 'YOUR_BROKER'

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(YOUR_BROKER, 1883, 60)
client.loop_forever()
```

AIoT with MQTT

3. Client as MQTT Publisher 

Python 3 Environment

```
from time import strftime

from paho.mqtt import publish

YOUR_BROKER = 'YOUR_BROKER'
topic = 'rollcall'
msg = f'Mary sign in on {strftime("%Y/%m/%d-%H:%M:%S")}'
publish.single(topic, msg, hostname=YOUR_BROKER)
```

AIoT with MQTT

3. Client as MQTT Publisher 

Python 3 Environment

```
msgs = [['rollcall/register', f'Jerry on {strftime("%Y/%m/%d-%H:%M:%S")}'],
        ['rollcall/signin', f'Eric on {strftime("%Y/%m/%d-%H:%M:%S")}']
       ]
publish.multiple(msgs, hostname=YOUR_BROKER)
```

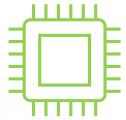
Raspberry Faces

3. 物連網的註冊與點名

建立 MQTT Broker

請參閱

Raspberry Faces

3. 物連網的註冊與點名 

啟動 rollcall_edge AIoT 模式 (on Raspberry Pi)

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② `cd; cd facecam`

③ `python3 rollcall_edge.py -m YOUR_BROKER`

Raspberry Trees

Trees Powered by TensorFlow Lite

1. **安裝套件** (on Raspberry Pi) 

    a. TensorFlow Lite Runtime (2.5.0)

        ① 開啟 Terminal (on LX 終端機/ MobaXterm)

        ② `echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list; curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -; sudo apt-get update; sudo apt-get install python3-tflite-runtime`

Trees Powered by TensorFlow Lite

2. **轉換模型** (on Windows 10) 

b. HDF5 ⇨ TensorFlow Lite

① 準備 Python 3 環境
`TensorFlow 2.4.4 (~ 2.5.3)`

② 下載檔案

轉換程式	`hdf5_to_tflite.py`
樹木模型	`trees17V1.h5`
標籤檔	`treeset_labels.txt`
測試圖檔	台灣欒樹.jpg 楓香.jpg ...

Trees Powered by TensorFlow Lite

2. 轉換模型 (on Windows 10) 

b. HDF5 ⇨ TensorFlow Lite

③ 轉換模型，開啟命令提示字元

```
python hdf5_to_tflite.py YOUR_HDF5.h5
python hdf5_to_tflite.py trees17V1.h5 ⇨ trees17V1.tflite
```

Trees Powered by TensorFlow Lite

3. 叫用模型 (on Raspberry Pi) 

c. Inference

① 上傳檔案 (to Raspberry Pi)，開啟 MobaXterm

測試程式　`trees-infer.py`
樹木模型　`trees17V1.tflite`
標籤檔　`treeset_labels.txt`
測試圖檔　台灣欒樹.jpg 楓香.jpg ...

② Inference，開啟 Terminal (on LX 終端機/ MobaXterm)

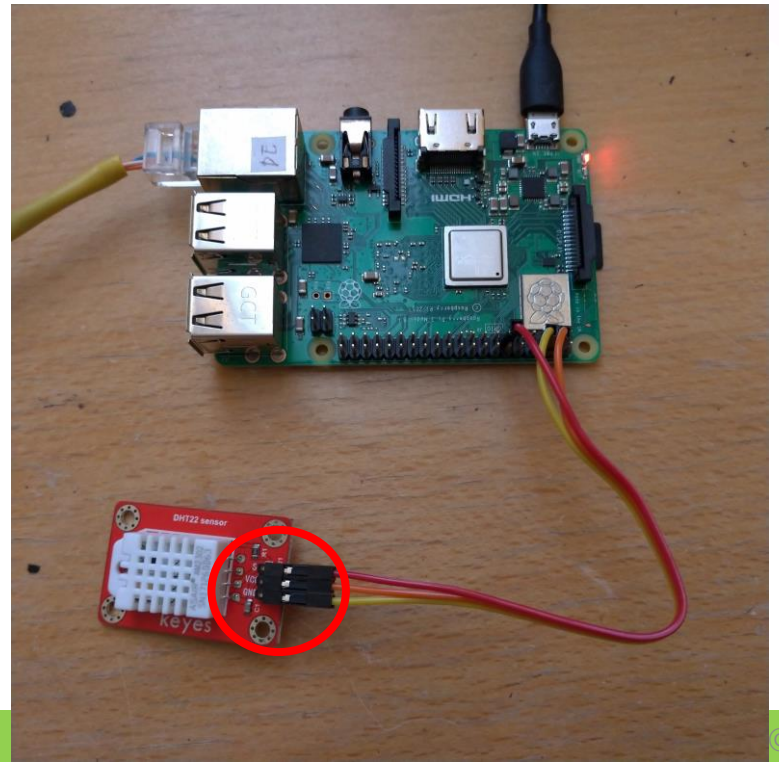
```
python3 trees-infer.py YOUR_TFLITE.tflite YOUR_TREES -l YOUR_LABELS
python3 trees-infer.py trees17V1.tflite 楓香.jpg -l treeset_labels.txt
```

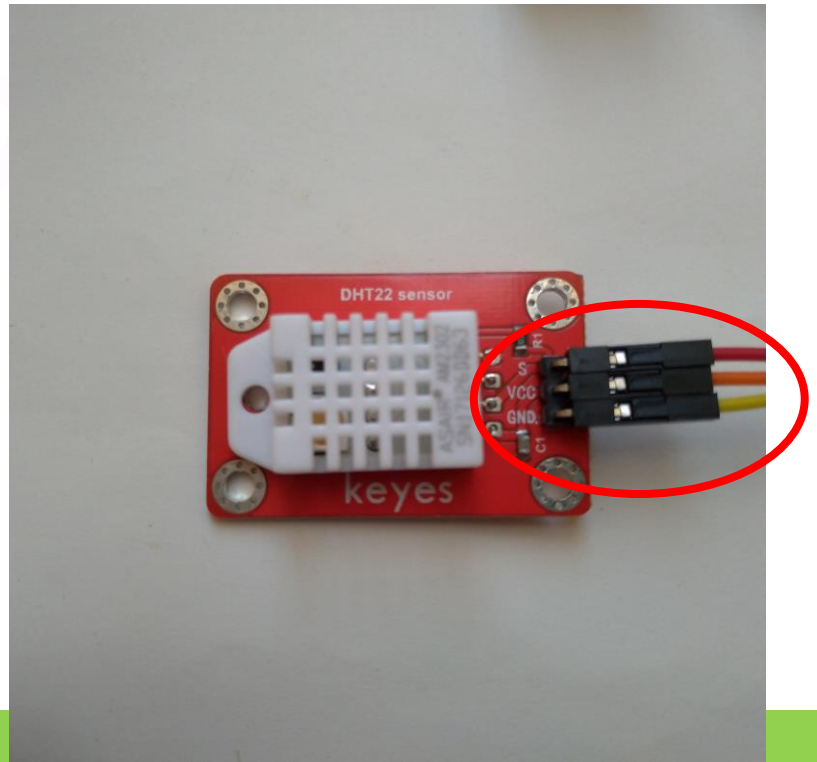
Raspberry Temperature

Get Temperature

1. 感測器界接

DHT-22 or DHT-11



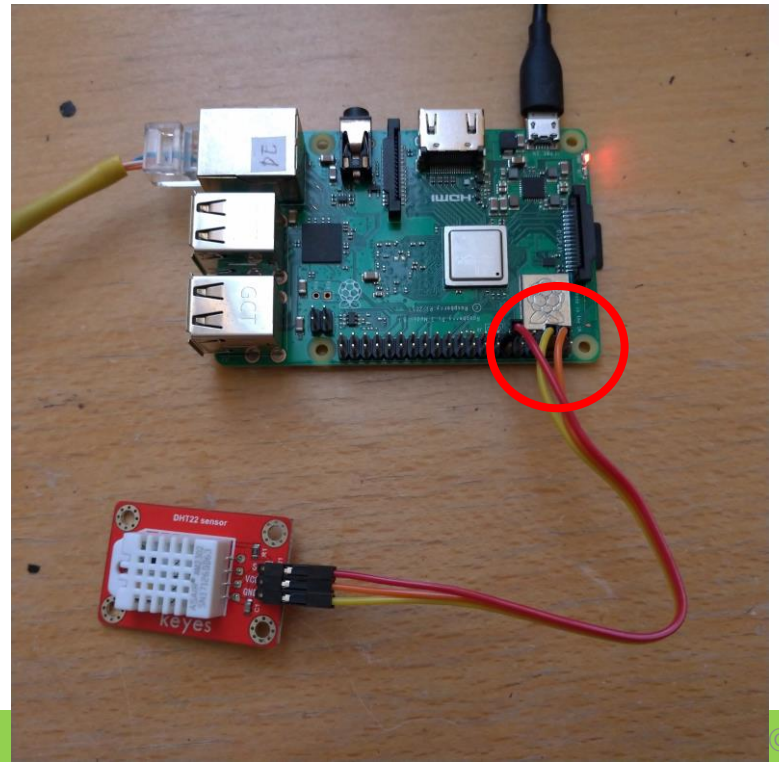


S
VCC
GND

Get Temperature

1. 感測器界接

DHT-22 or DHT-11



連接 pinout 請參閱 https://pinout.xyz

S	➔ 任接 GPIO
VCC	➔ 任接 5V 或 3V3
GND	➔ 任接 Ground

S	➔ 任接 GPIO
+	➔ 任接 5V 或 3V3
-	➔ 任接 Ground

Get Temperature

2. 安裝套件 (on Raspberry Pi) 

套件 Adafruit_Python_DHT

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② `python3 -m pip install adafruit-circuitpython-dht; sudo apt-get install libgpiod2`

Get Temperature

3. 應用開發 (on Raspberry Pi) 

a. 叫用套件

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② 取得 `dht.py`

③ `python3 dht.py -g YOUR_GPIO`

```
python3 dht.py -g 27
```

Get Temperature

3. **應用開發** (on Raspberry Pi) 

    b. 整合 MQTT

    請參閱

Your Turn - Watching the Temperature

以 Raspberry Pi 實作溫度監控機制，當溫度高於 25°C 以 MQTT 告警

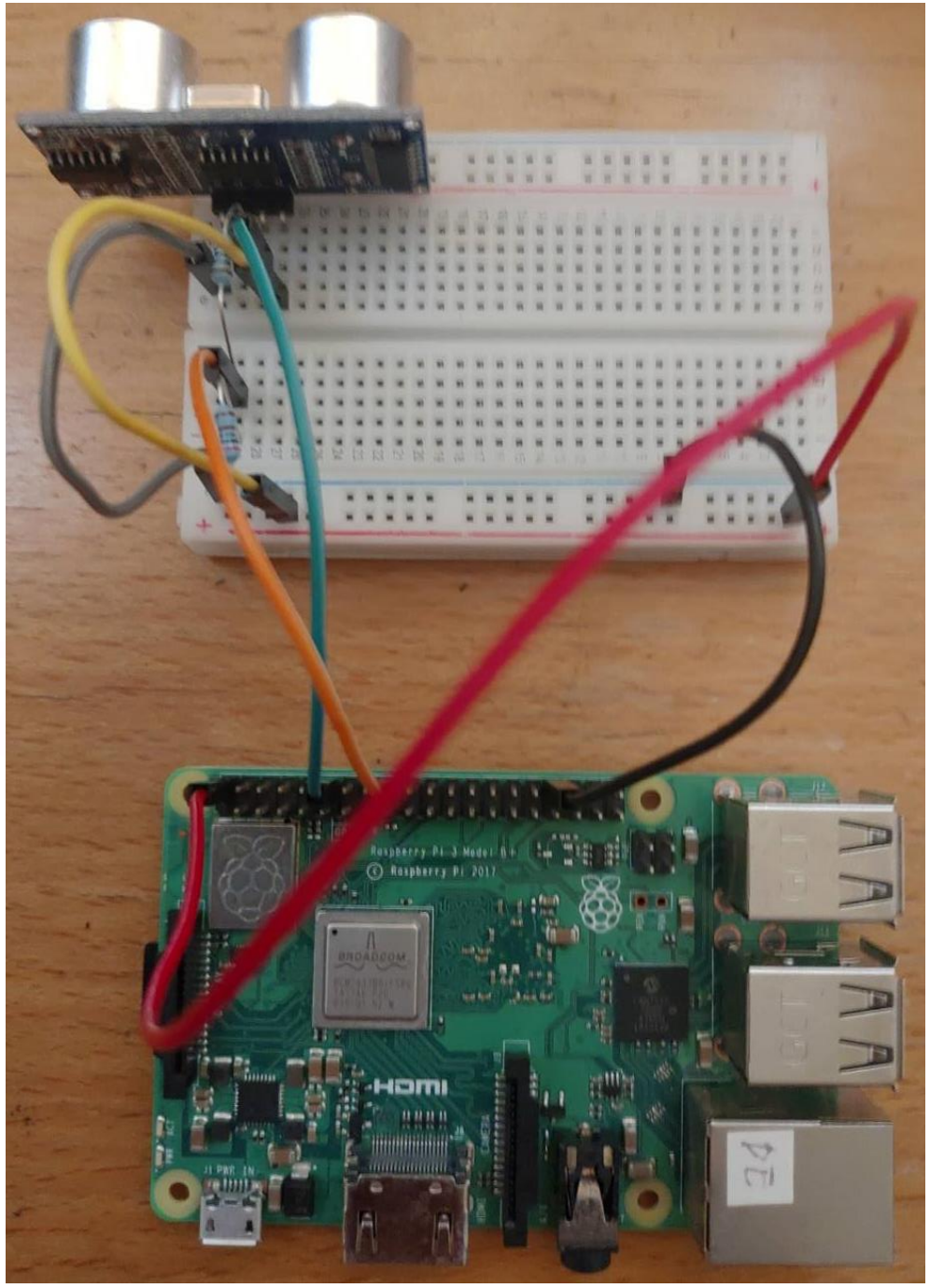
Raspberry Distance

Get Distance

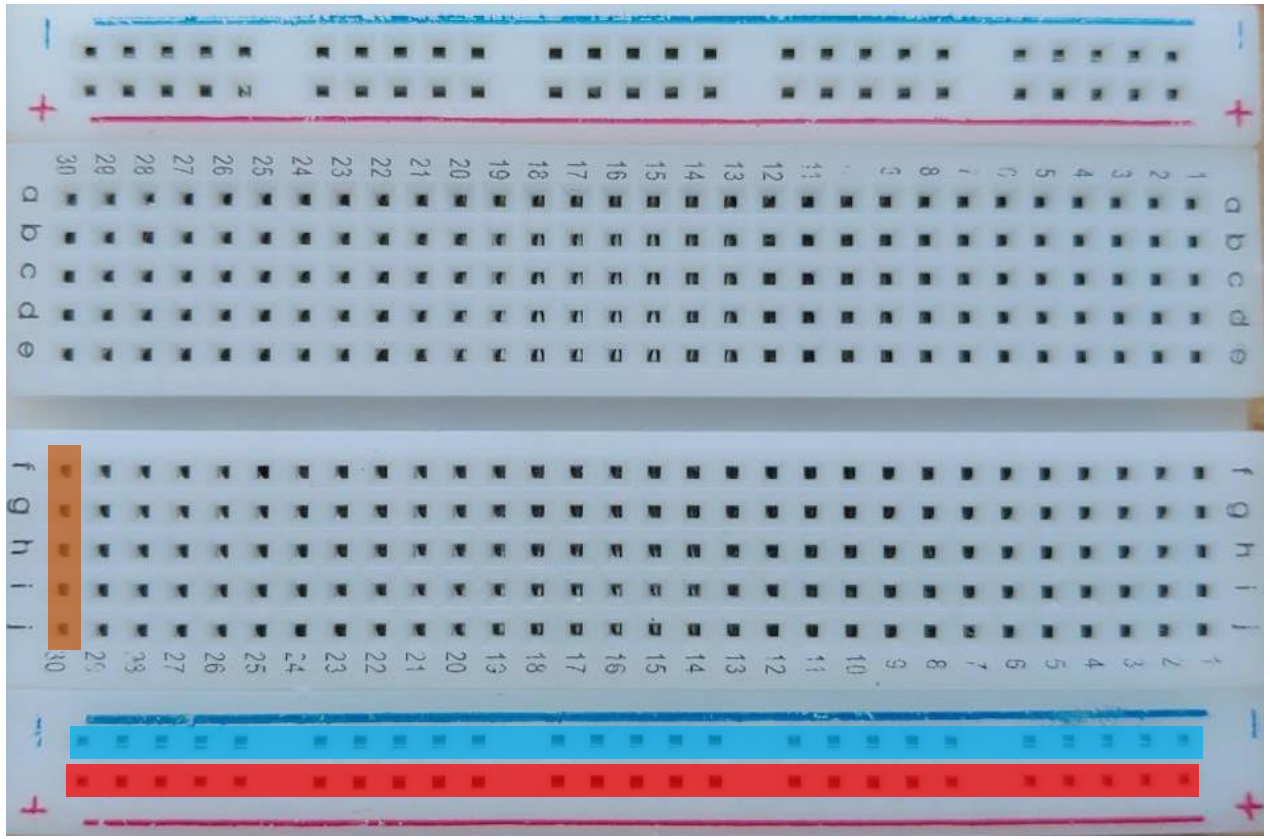
1. 聲納界接

HC-SR04 to Raspberry Pi

```
HC-SR04      Breadboard      Raspi
Vcc      -------->------>  5V
Trig     -------->------>  GPIO
Echo     -> 1KΩ --+->--->  GPIO
                  -> 2KΩ ->  GND
GND      -------->------>  GND
```



Breadboard Conecpt



Get Distance

2. 應用開發 

測試程式

① 開啟 Terminal (on LX 終端機/ MobaXterm)

② 取得 `hcsr04.py`

③ `python3 hcsr04.py -e YOUR_ECHO -t YOUR_TRIGGER`

```
python3 hcsr04.py -e 17 -t 27
```

Your Turn - Watching the Distance

以 Raspberry Pi 實作距離監控機制，當距離小於 10cm 以 MQTT 告警

Jetson Installation

Jetson Nano 2GB Installation

1. **製作開機磁碟** (on Windows 10)

2. **組裝並開機** (on Jetson Nano)

3. **初始化** (on Jetson Nano)

官方指引

https://developer.nvidia.com/ >
SOULTIONS > Intelligent Machines: Embedded and Edge AI >
DEVELOPER KITS: Getting Started >
Jetson Nano 2GB Developer Kit: Getting Started Guide

Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10) 

a. 下載 Jetson Nano 2GB 映像檔

① 以瀏覽器搜尋 Jetson Download Center >
SEARCH: Jetson Nano 2GB >
Jetson Nano 2GB Developer Kit SD Card Image 4.6 >
Jetson Nano 2GB Developer Kit SD Card Image

② jetson-nano-2gb-jp46-sd-card-image.zip (6.08GB)

Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10) 

b. 下載並安裝 SD Memory Card Formatter 5.0.2 for SD/SDHC/SDXC

① https://sdcard.org/ >
Downloads: SD Memory Card Formatter >
For Windows >
Accept

② SDCardFormatterv5_WinEN.zip (6.12MB)

③ 解壓縮後點擊 SD Card Formatter 5.0.2 Setup EN.exe 開始安裝

④ 以預設值安裝完畢

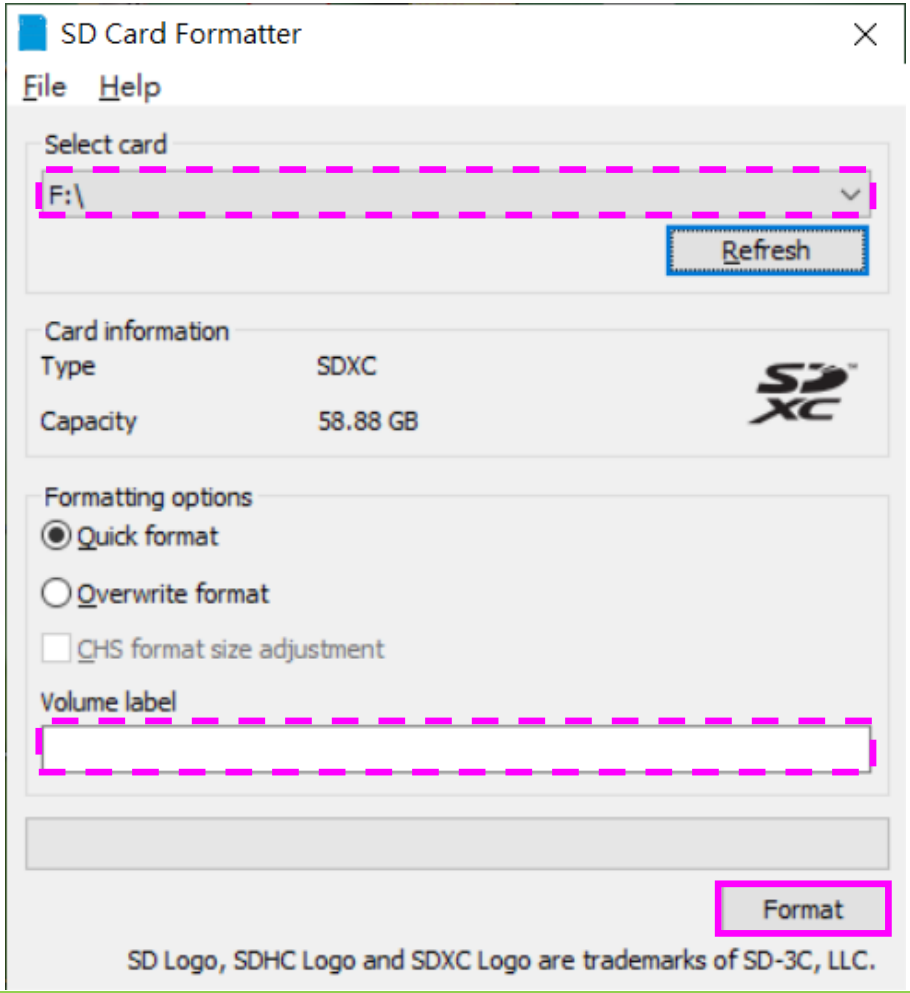
Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

c. 格式化 microSD

① 插入 microSD

② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format

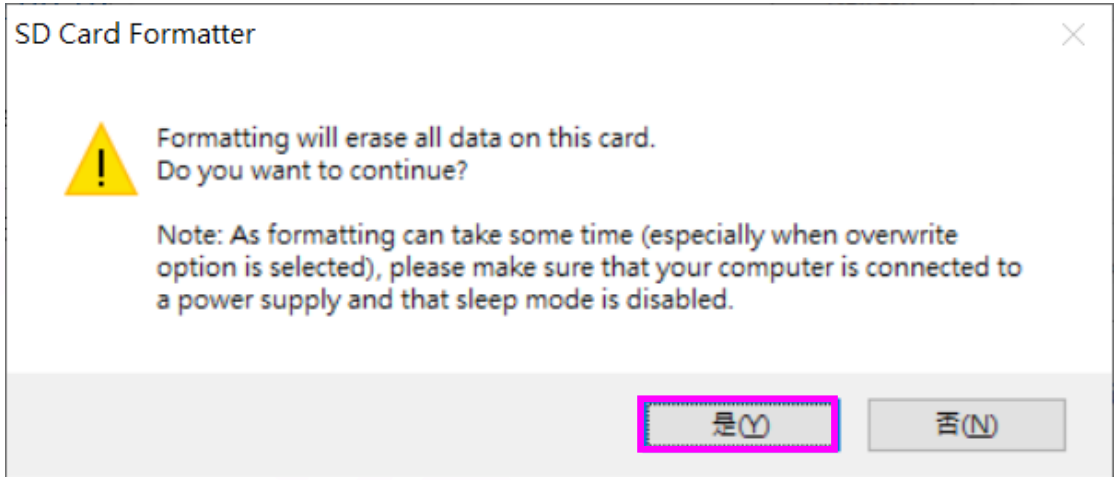


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

c. 格式化 microSD

① 插入 microSD
② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format
③ 是(Y) 開始快速格式化

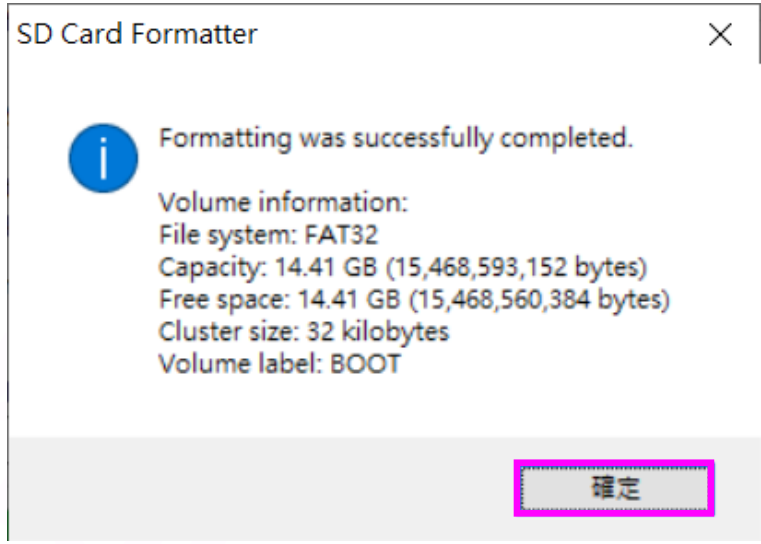


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

c. 格式化 microSD

① 插入 microSD
② 開啟 SD Card Formatter > Select card 選擇正確磁碟 > Volume label 清空 > Format
③ 是(Y) 開始快速格式化
④ 已完成， 確定



Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

d. 下載並安裝 Etcher

① https://www.balena.io/ >
More products >
balenaEtcher >
(往下捲) Get your assets: Etcher for Windows (x86|x64) (Portable) Download

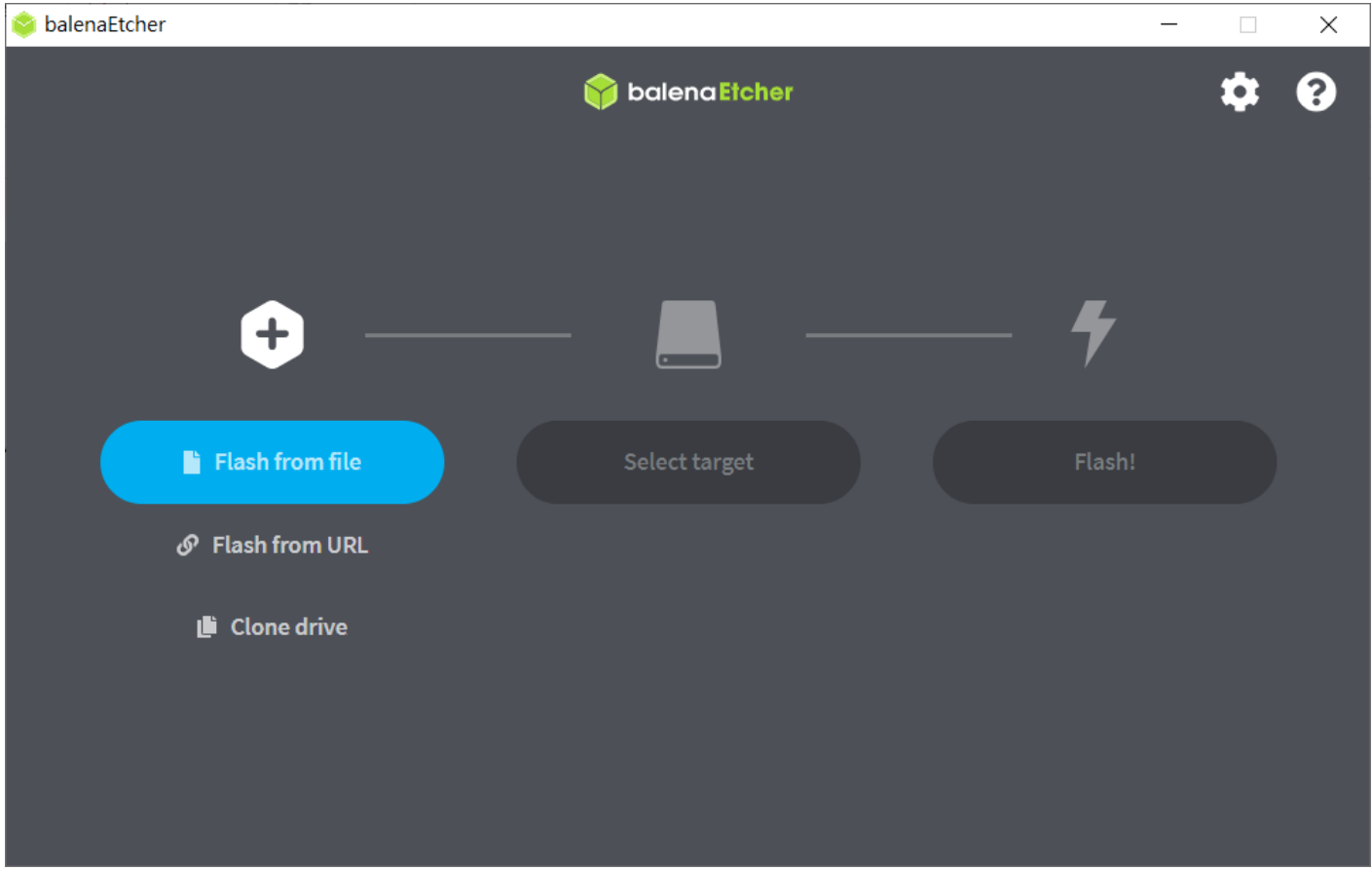
② balenaEtcher-Portable-1.7.9.exe (Windows)

Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher

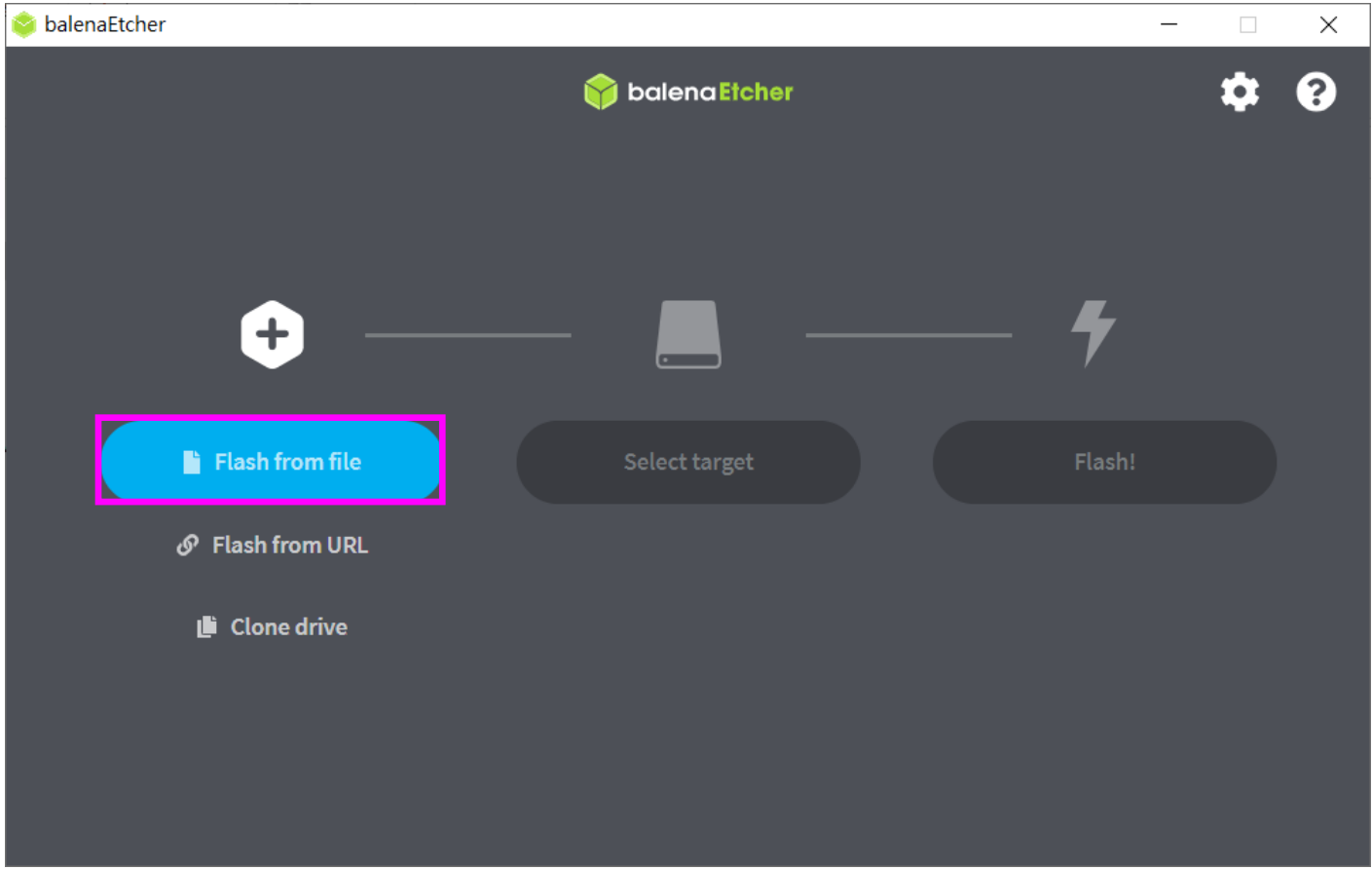


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file

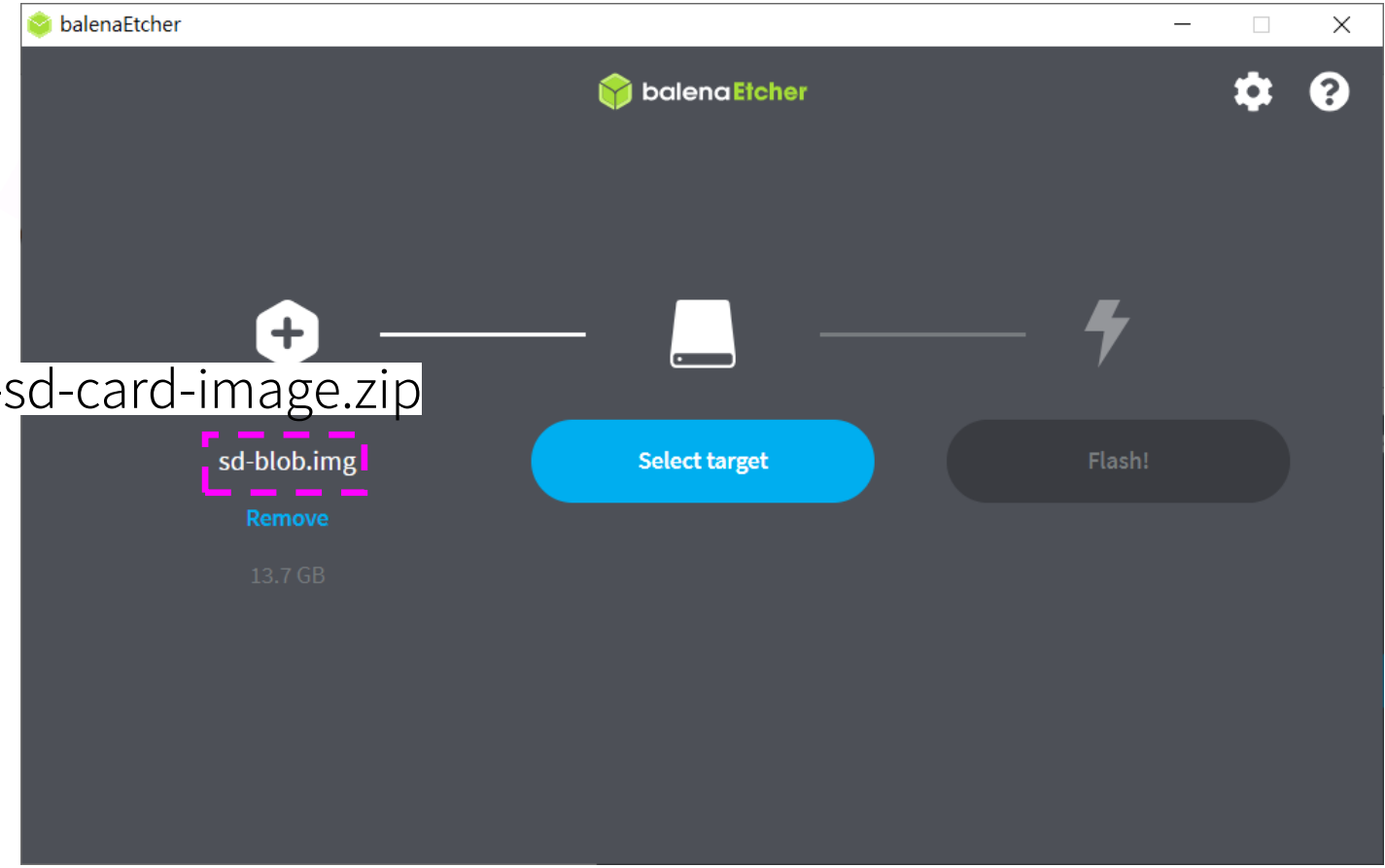


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip

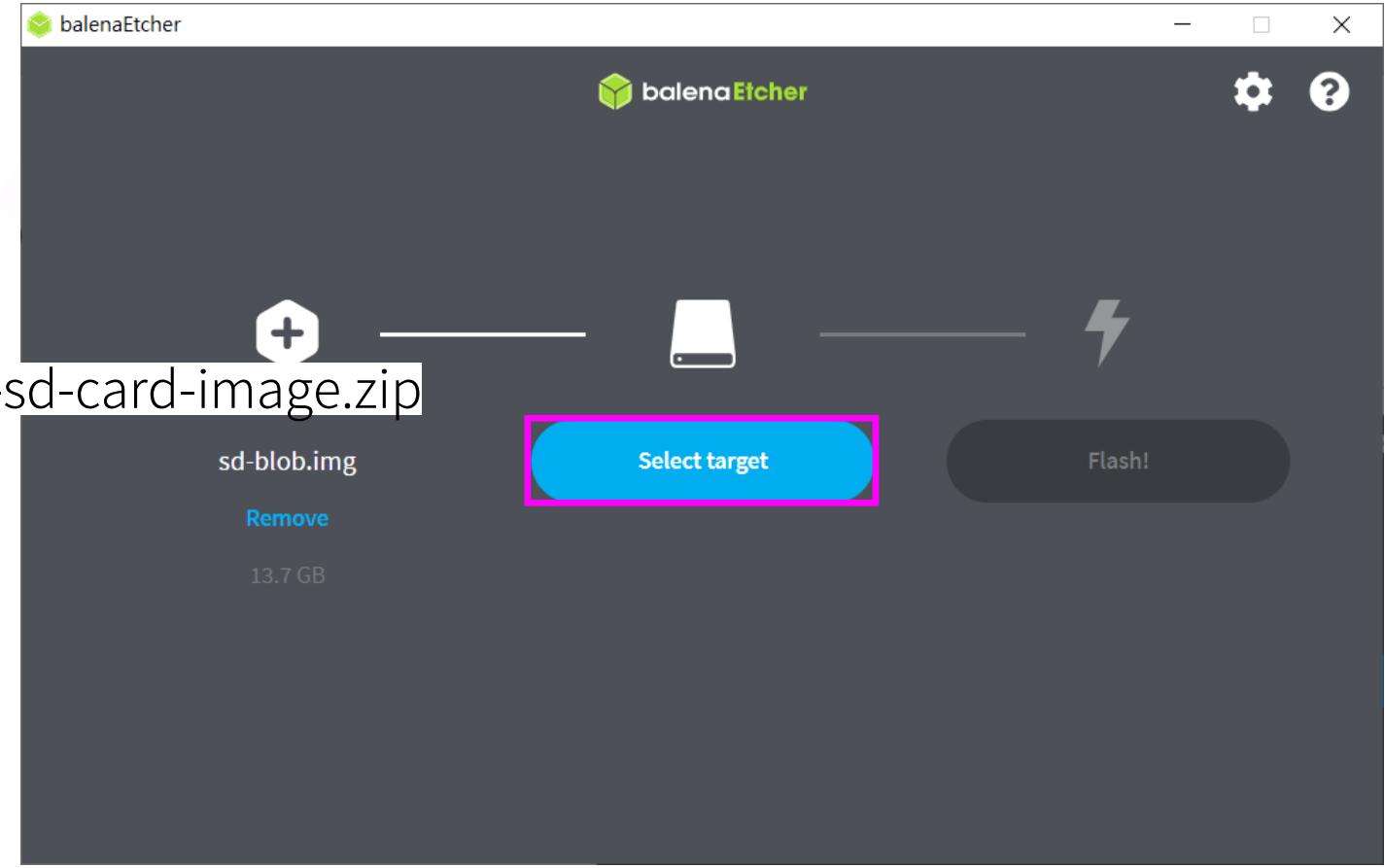


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip
④ Select target

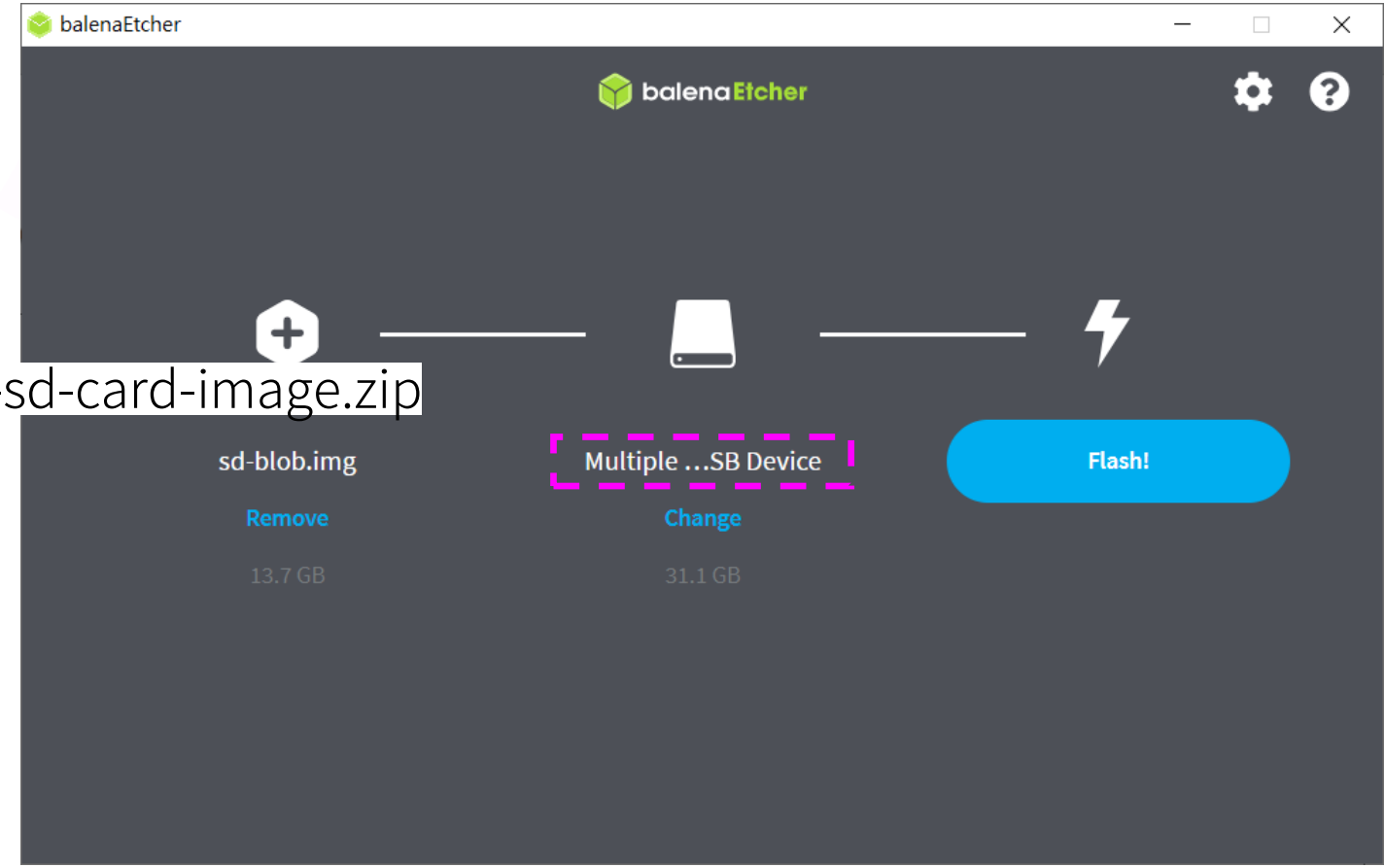


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip
④ Select target
⑤ *YOUR_MICROSD_DRIVE*

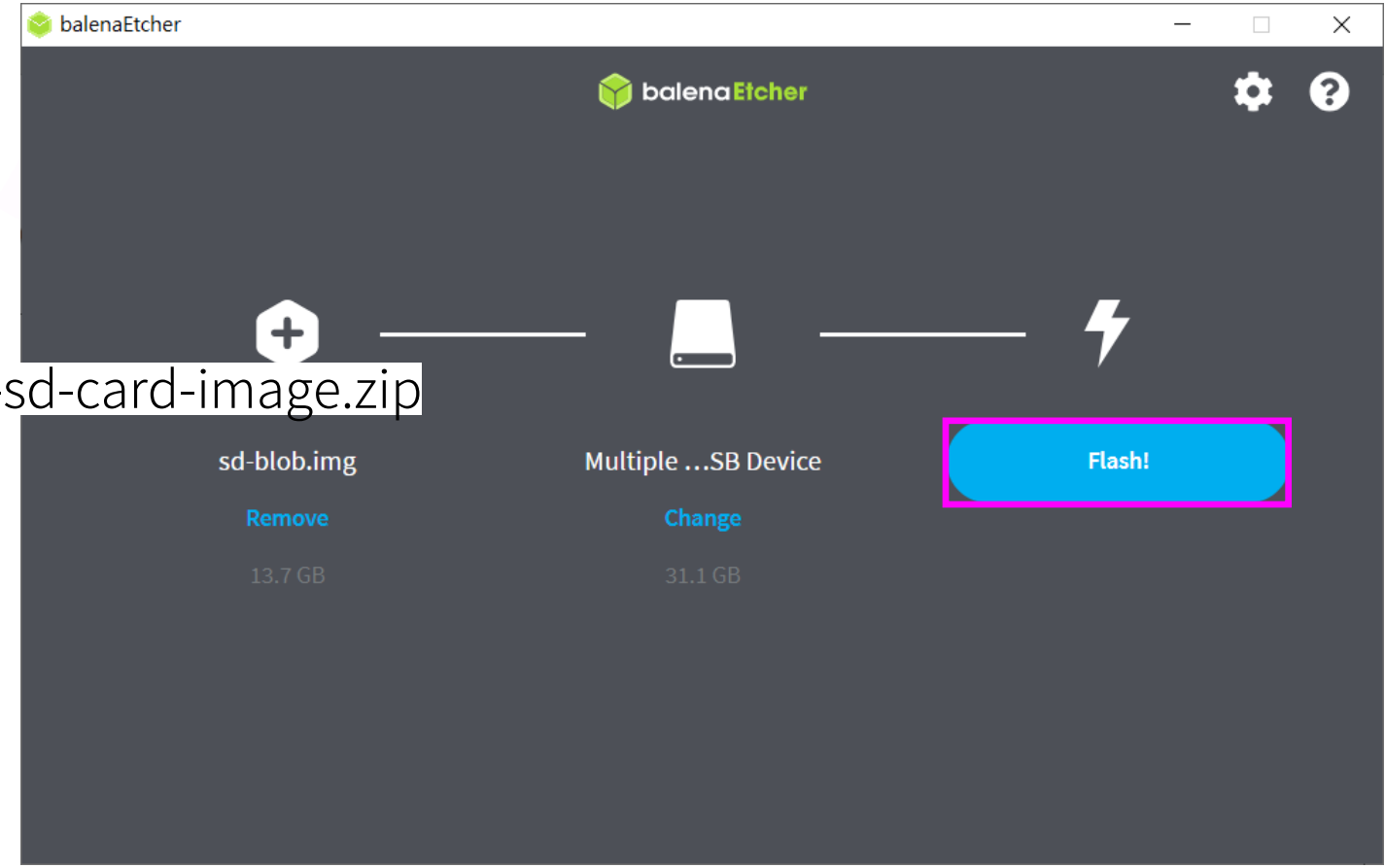


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 23 分)

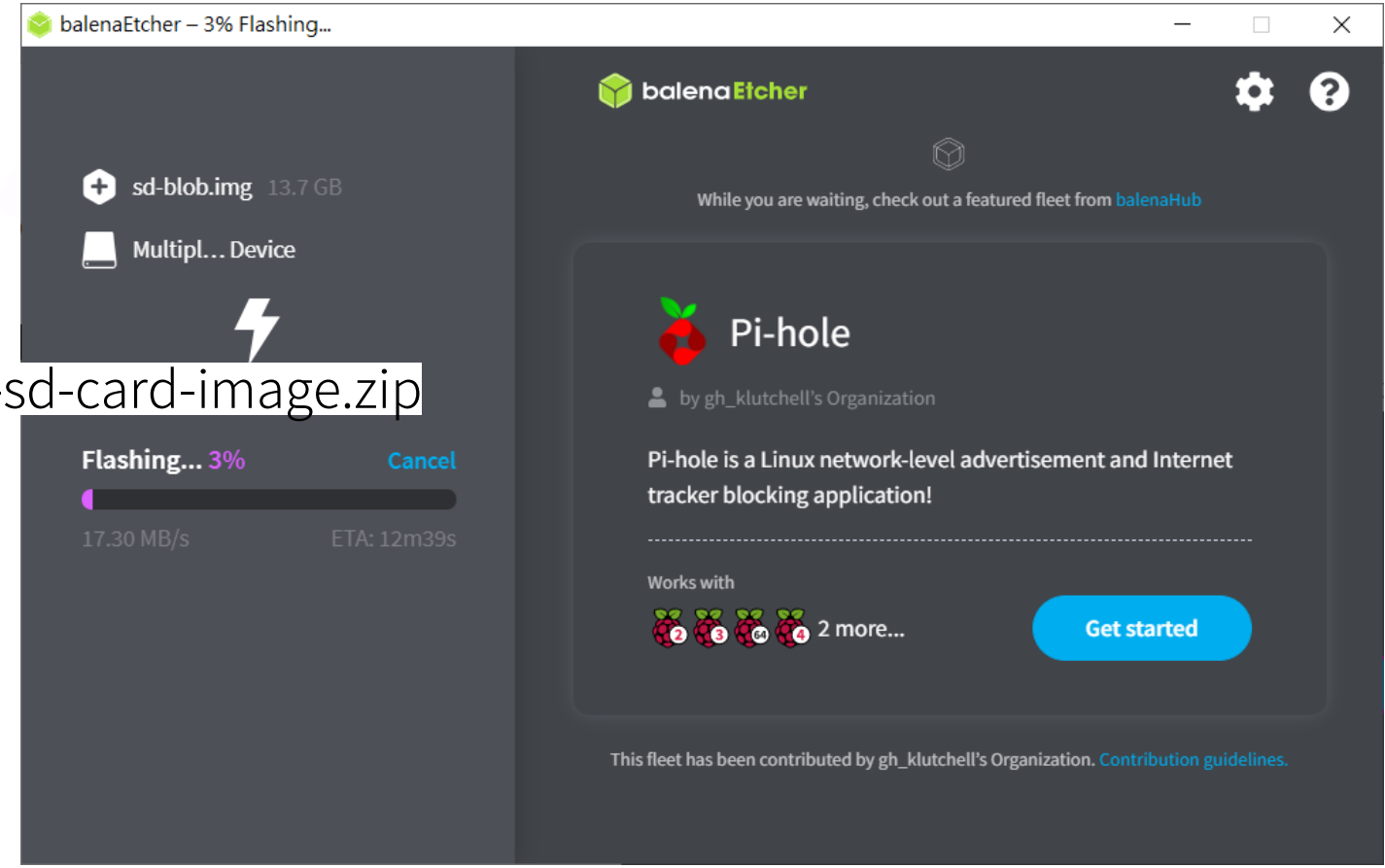


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 23 分)

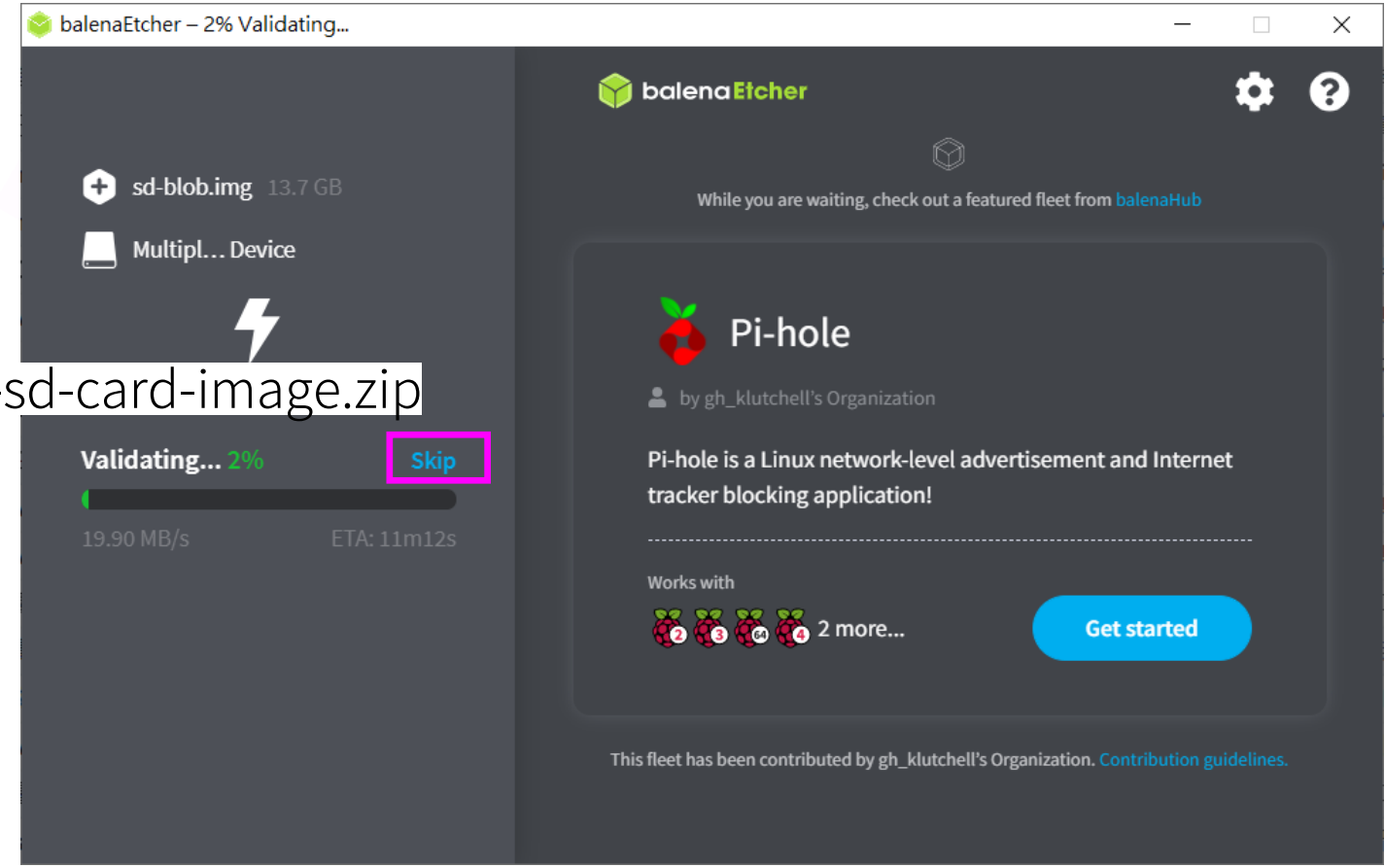


Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

e. 燒錄 JetPack 4.6

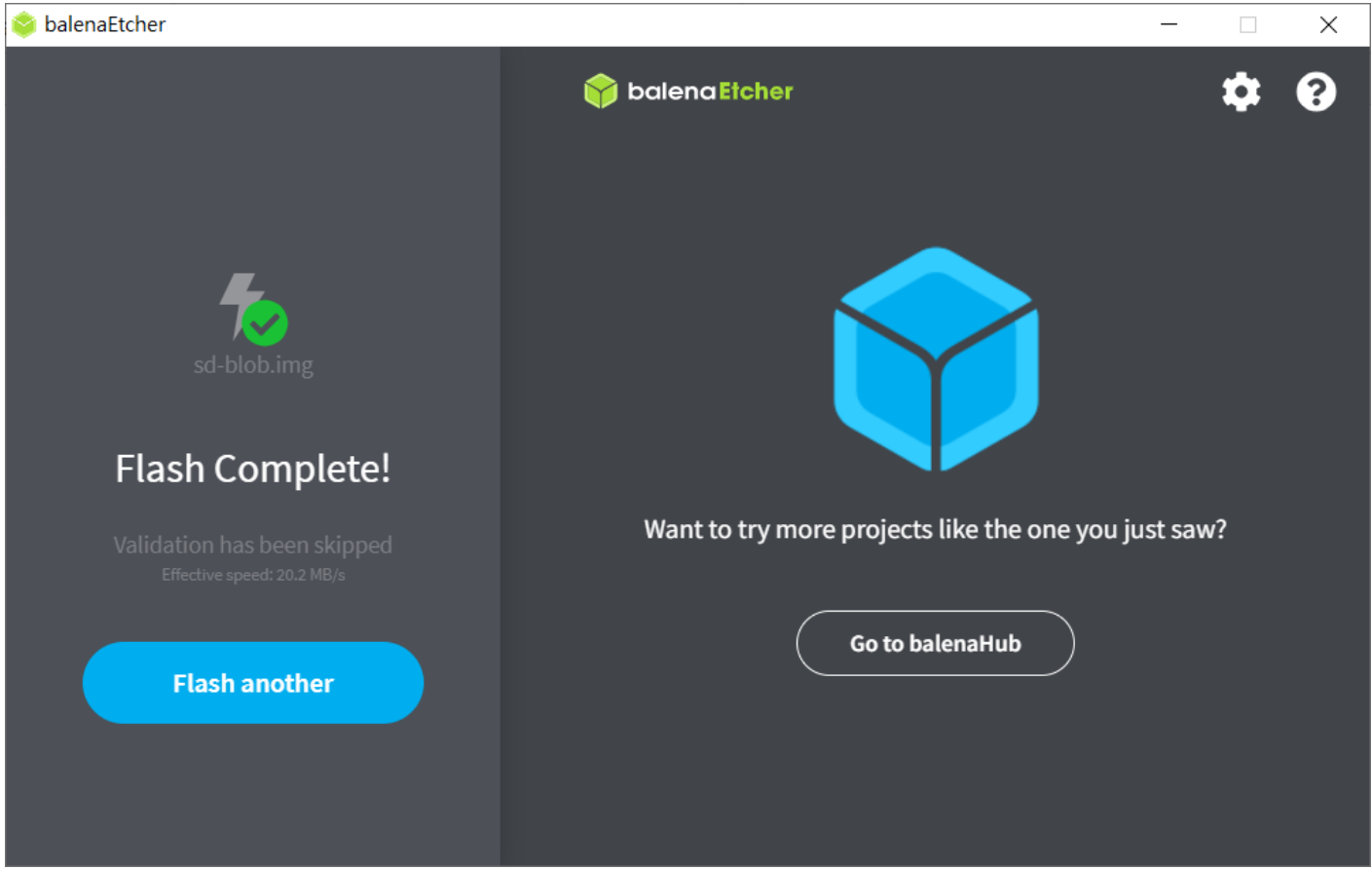
① 啟動 Etcher
② Flash from file
③ 選擇 jetson-nano-2gb-jp46-sd-card-image.zip
④ Select target
⑤ *YOUR_MICROSD_DRIVE*
⑥ Flash! 燒錄 (約 23 分)
⑦ 驗證，Skip



Jetson Nano 2GB Installation

1. **製作開機磁碟** (on Windows 10)

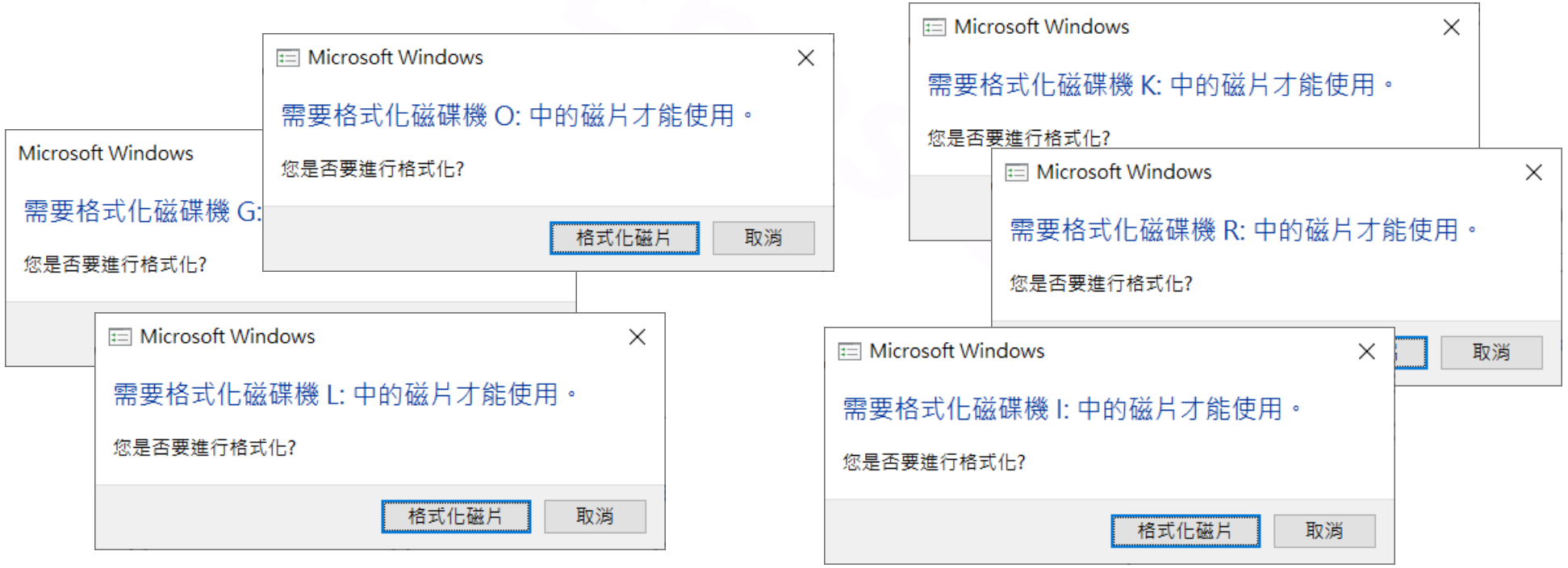
    f. 退出 microSD



Jetson Nano 2GB Installation

1. 製作開機磁碟 (on Windows 10)

過程中若跳出一堆 (十多個) 對話方塊請 取消



Jetson Nano 2GB Installation

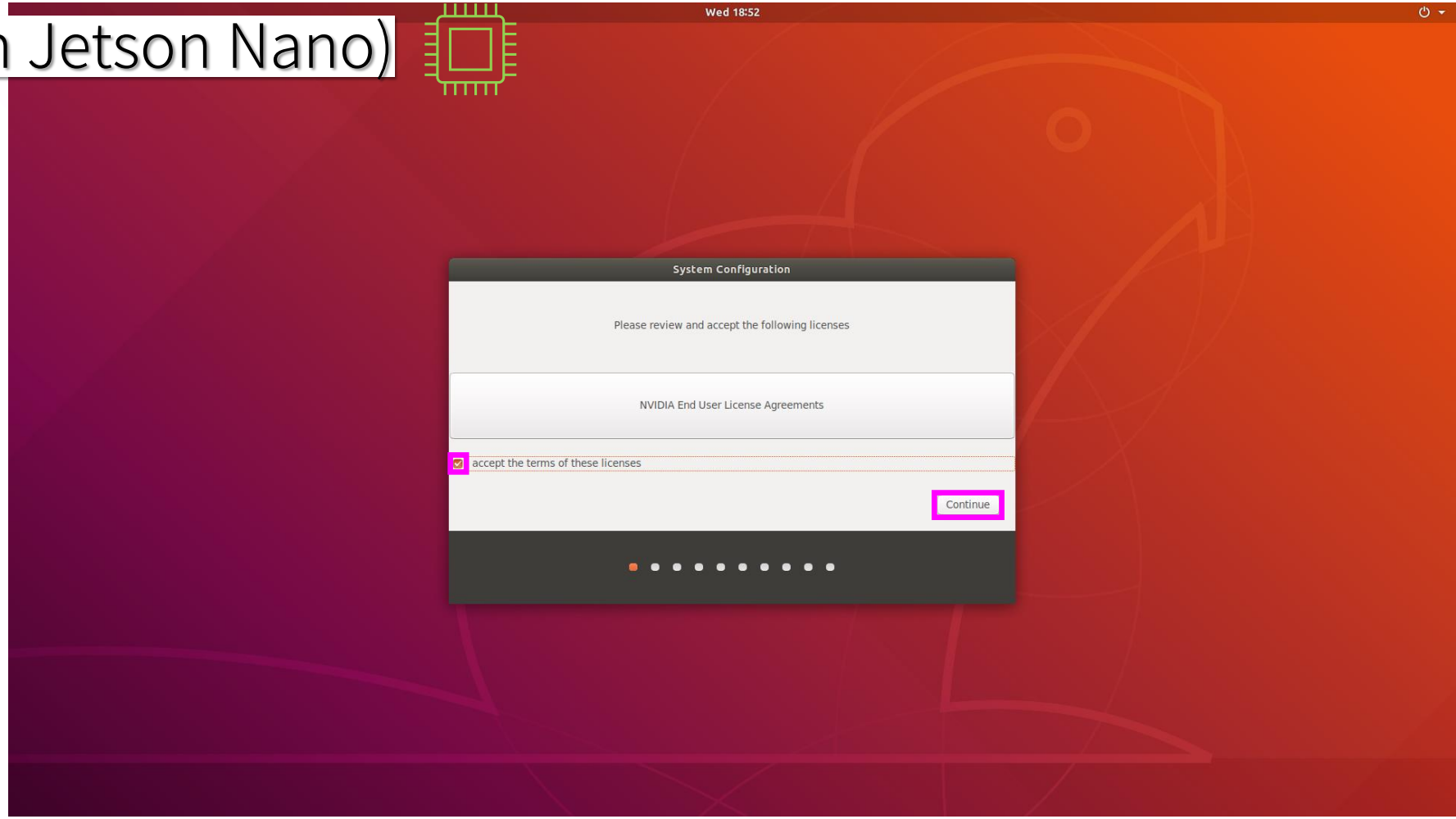
2. 組裝並開機 (on Jetson Nano)

插入 microSD
接上 USB-A WiFi Dongle
接上 CSI-2 or USB-A Camera
接上 USB-A 鍵盤滑鼠
接上 HDMI 螢幕
接上 USB-C 5V3A 電源

Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

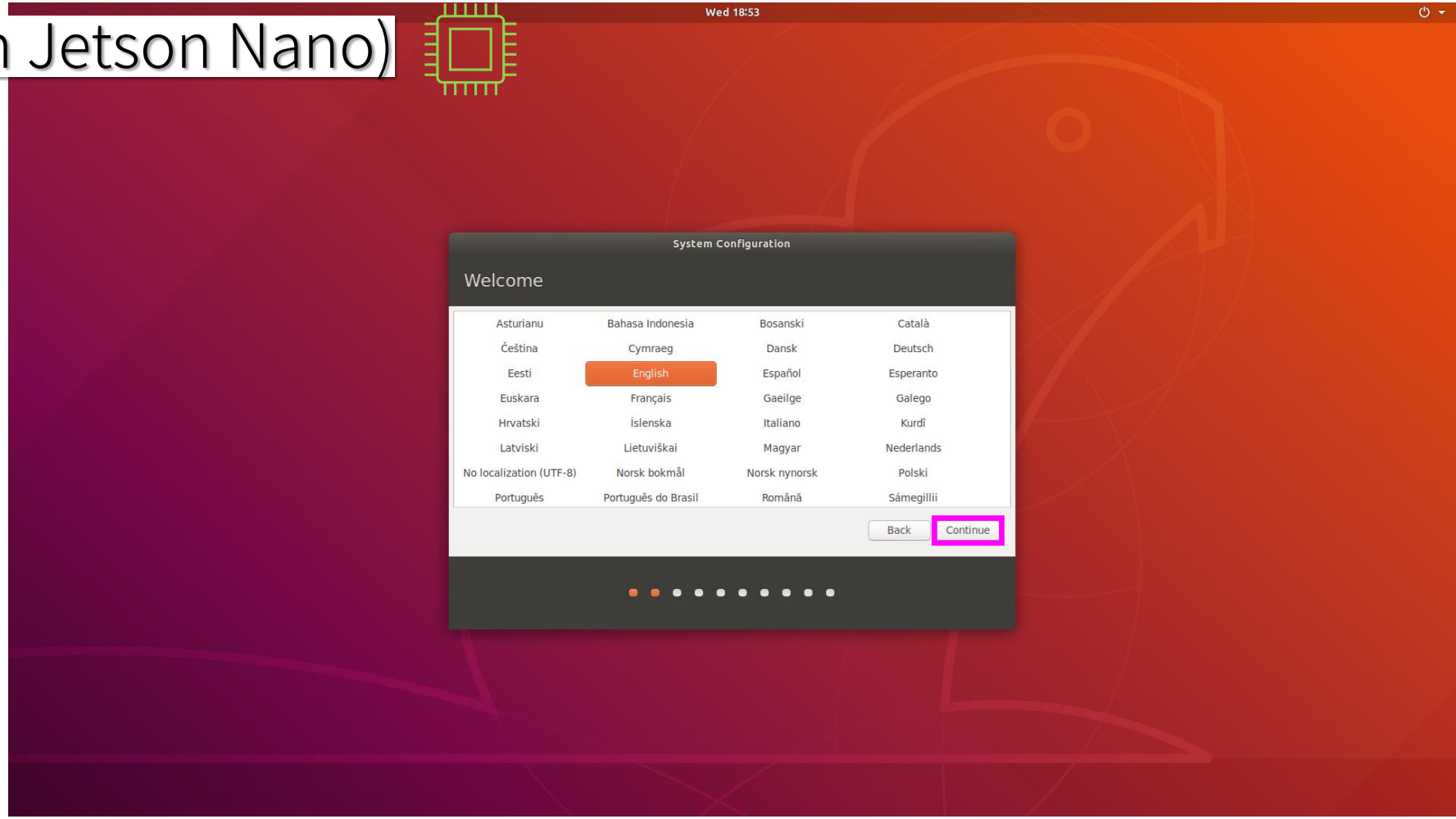
接受授權合約



Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

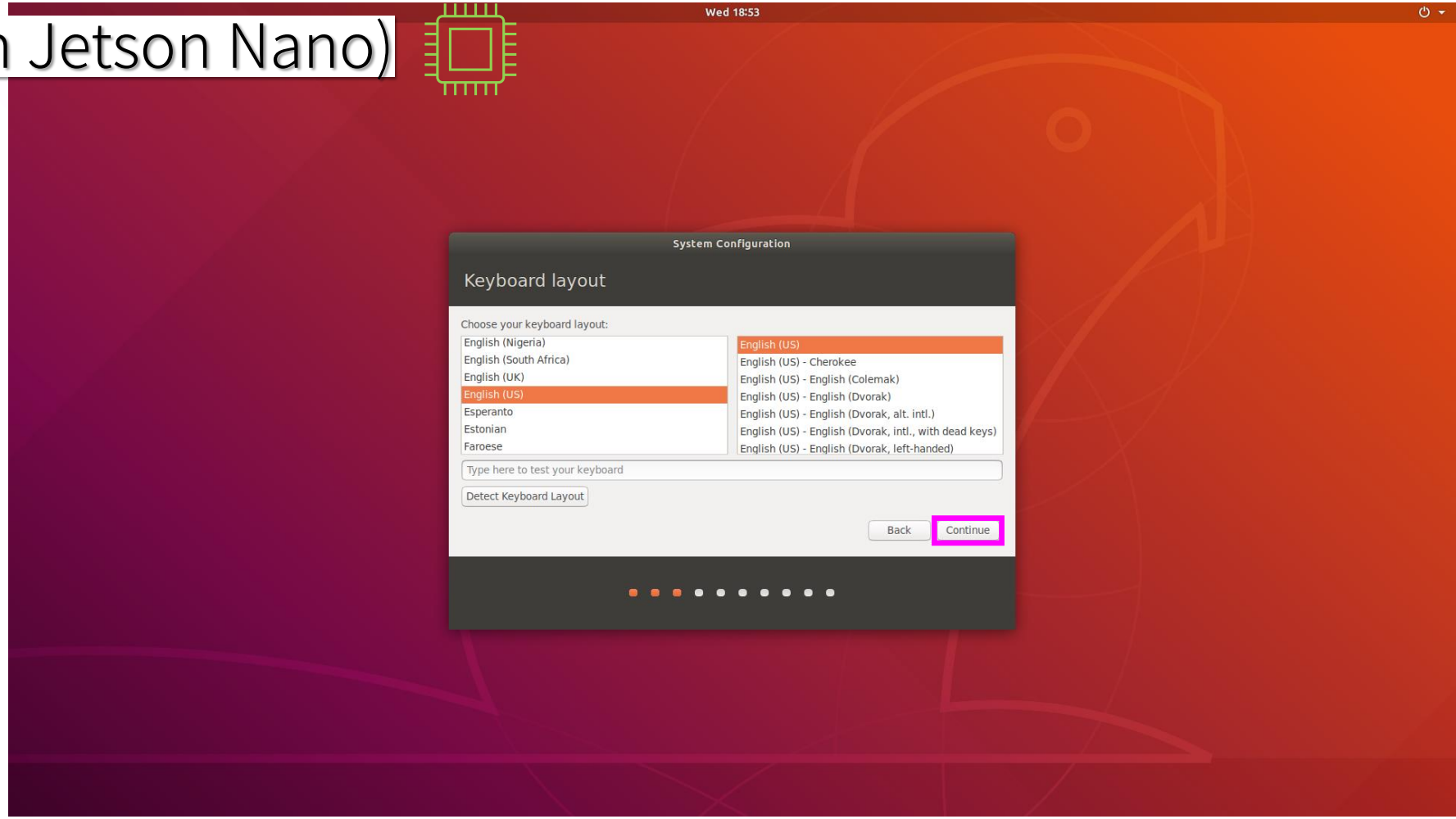
選擇顯示語言



Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

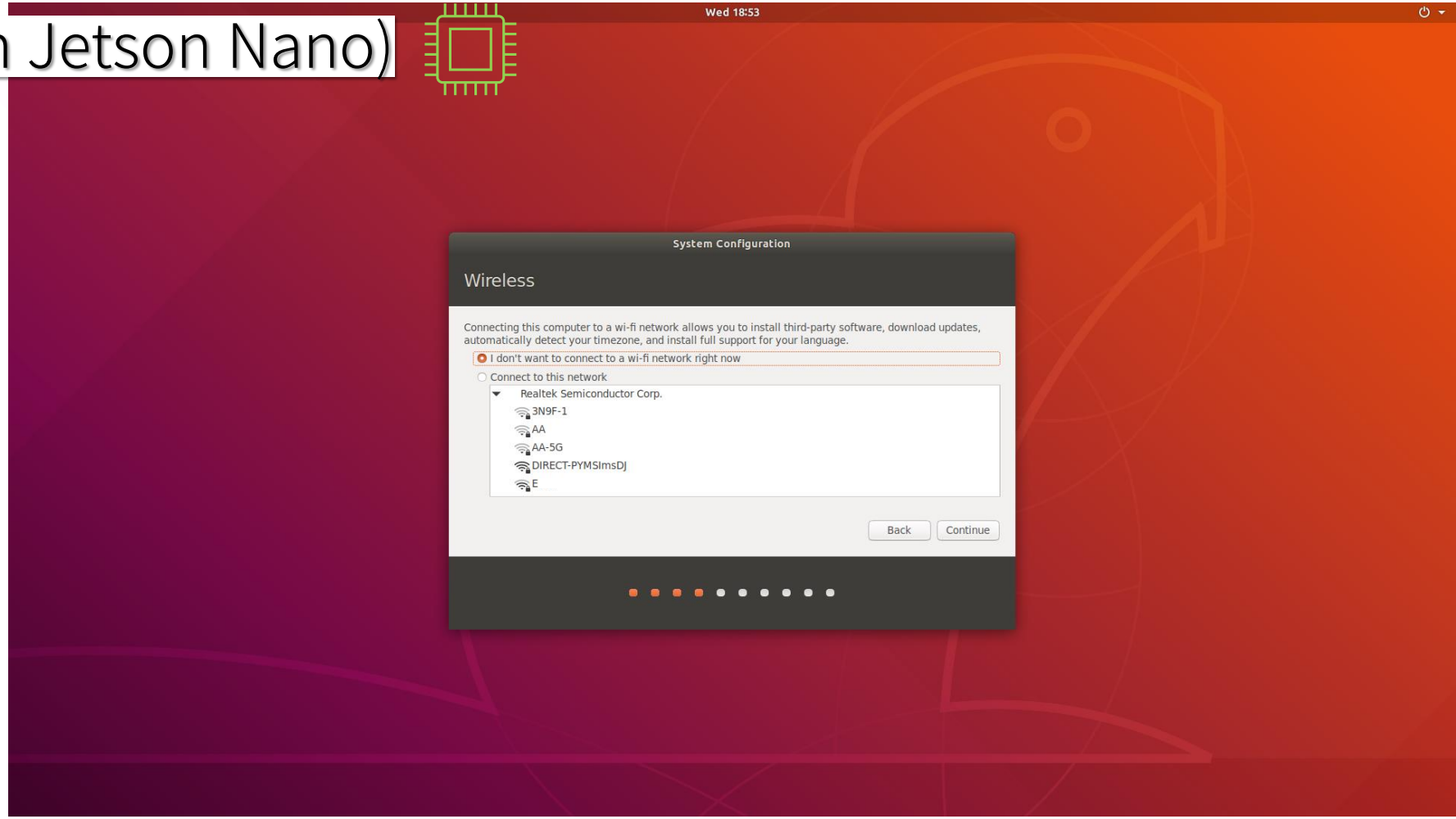
選擇鍵盤



Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

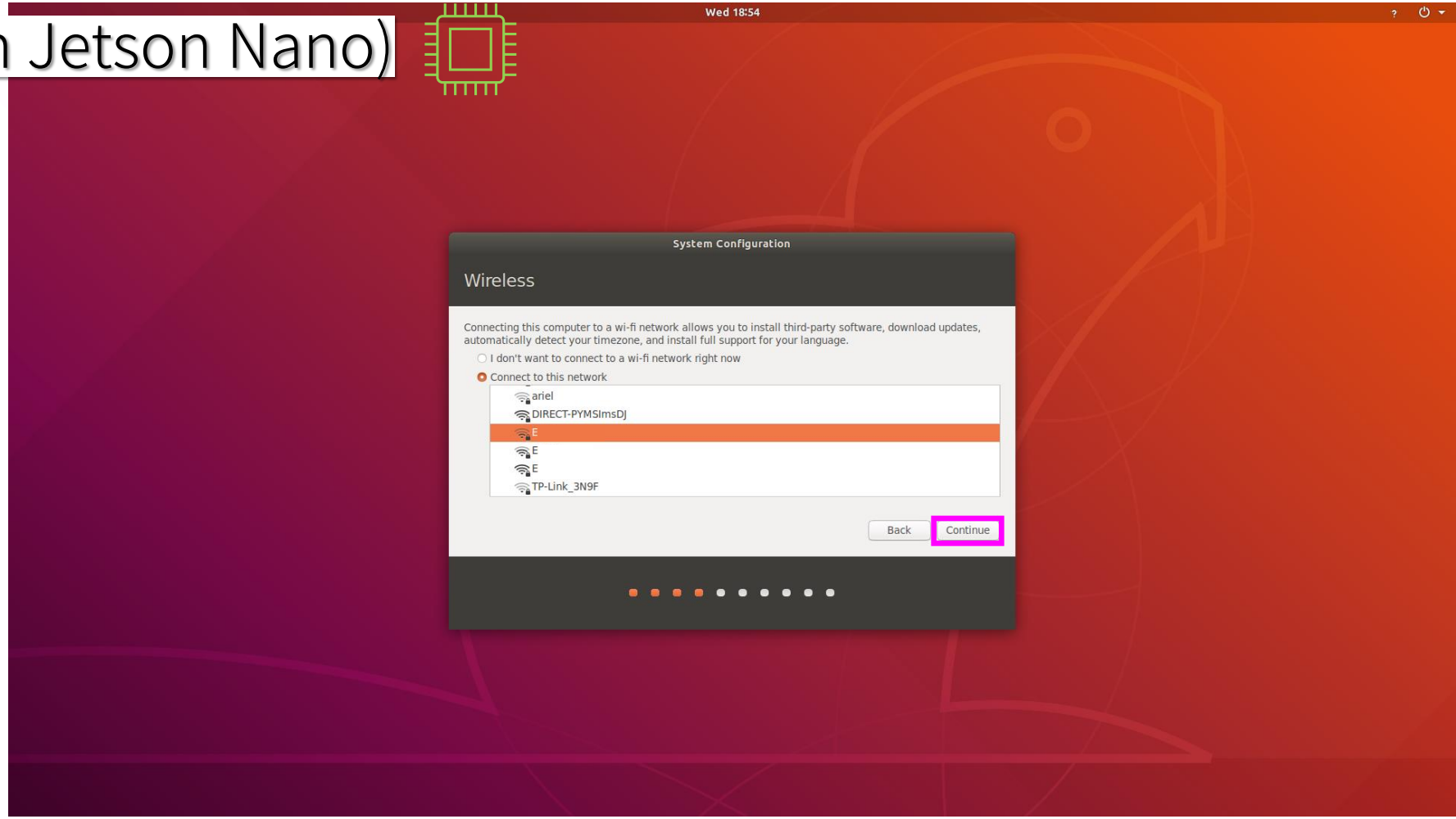
設定 WiFi



Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

設定 WiFi



Jetson Nano 2GB Installation

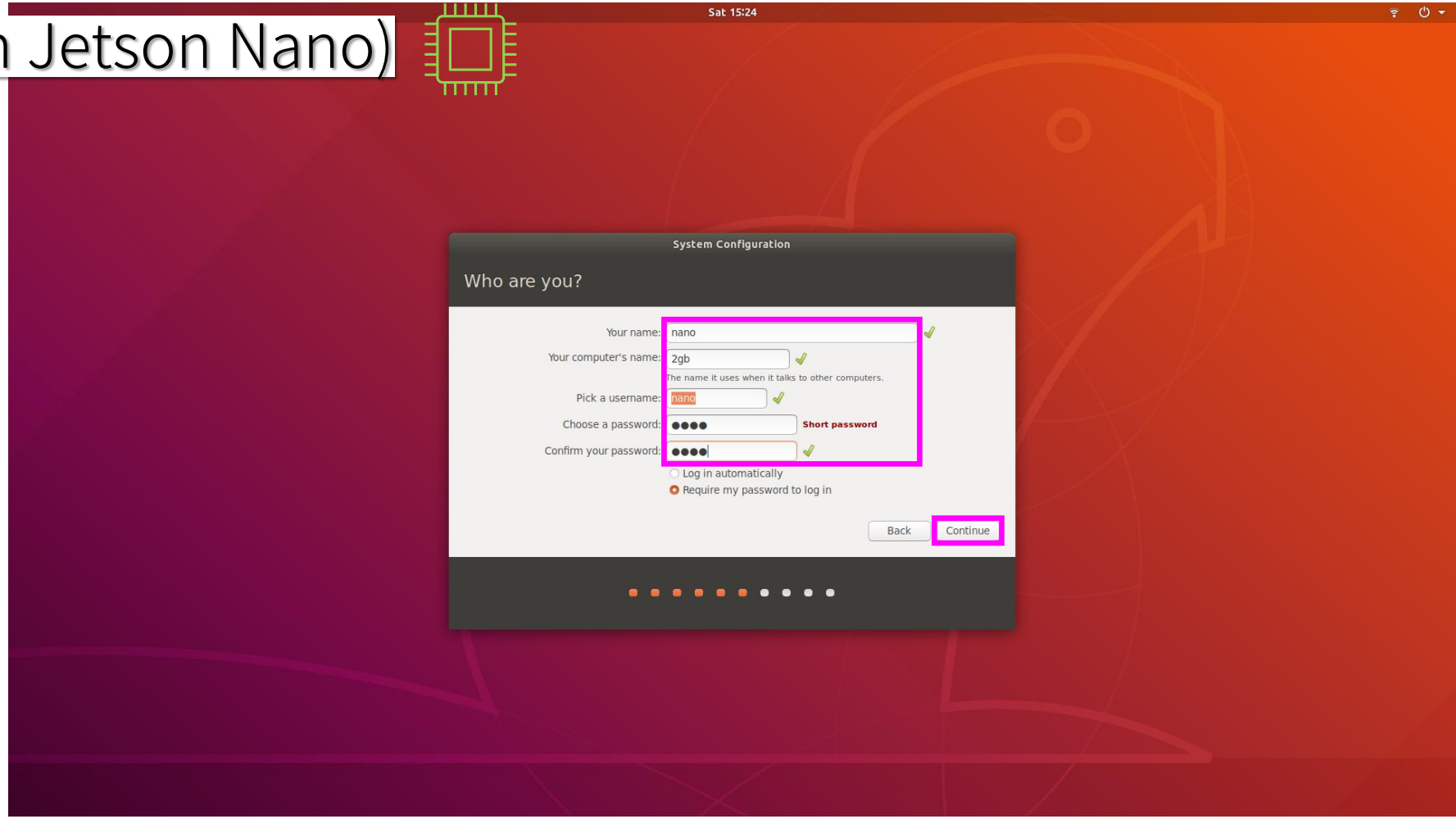
2. 組裝並開機 (on Jetson Nano)

設定時區

Jetson Nano 2GB Installation

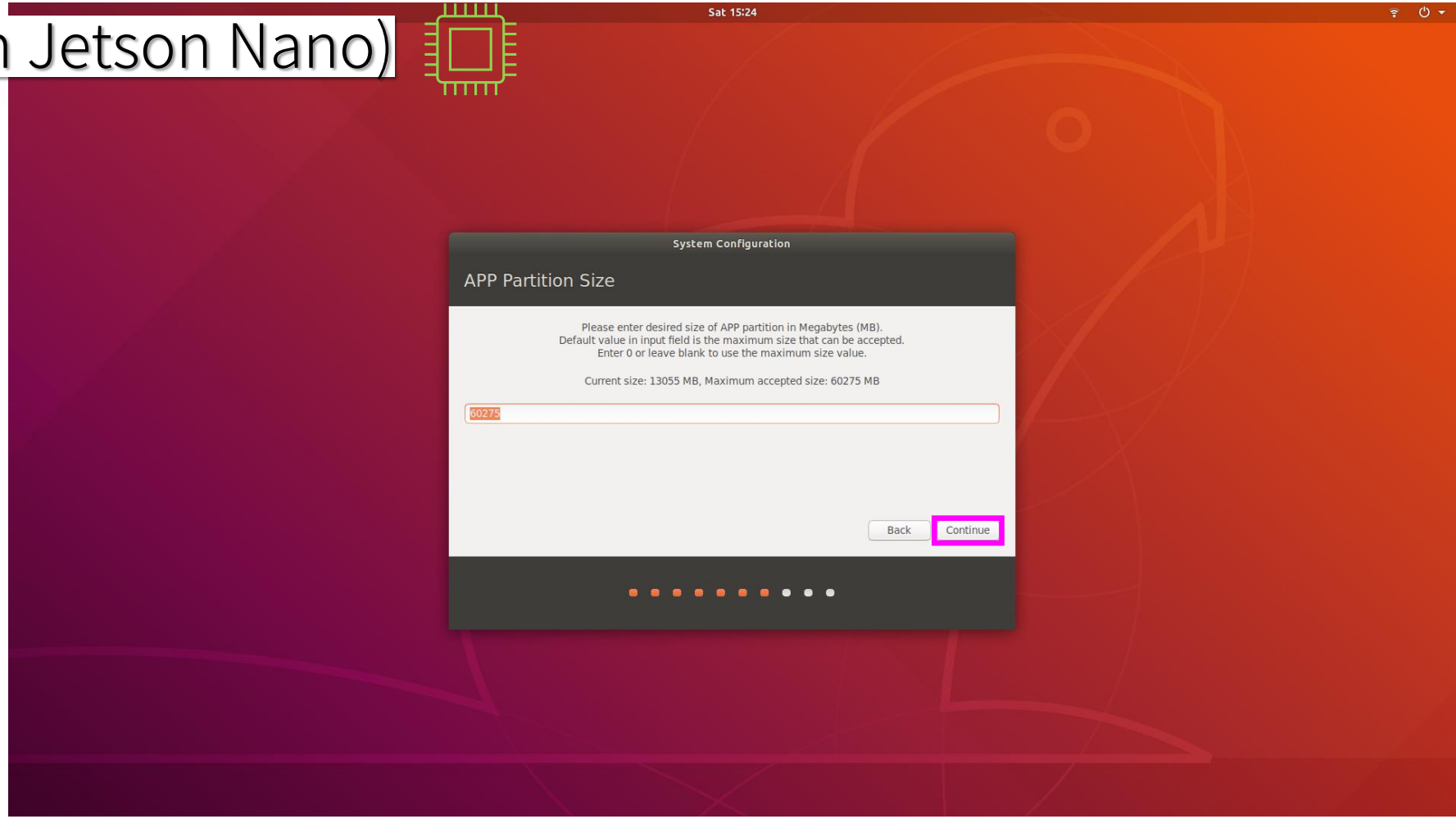
2. 組裝並開機 (on Jetson Nano)

設定帳號密碼



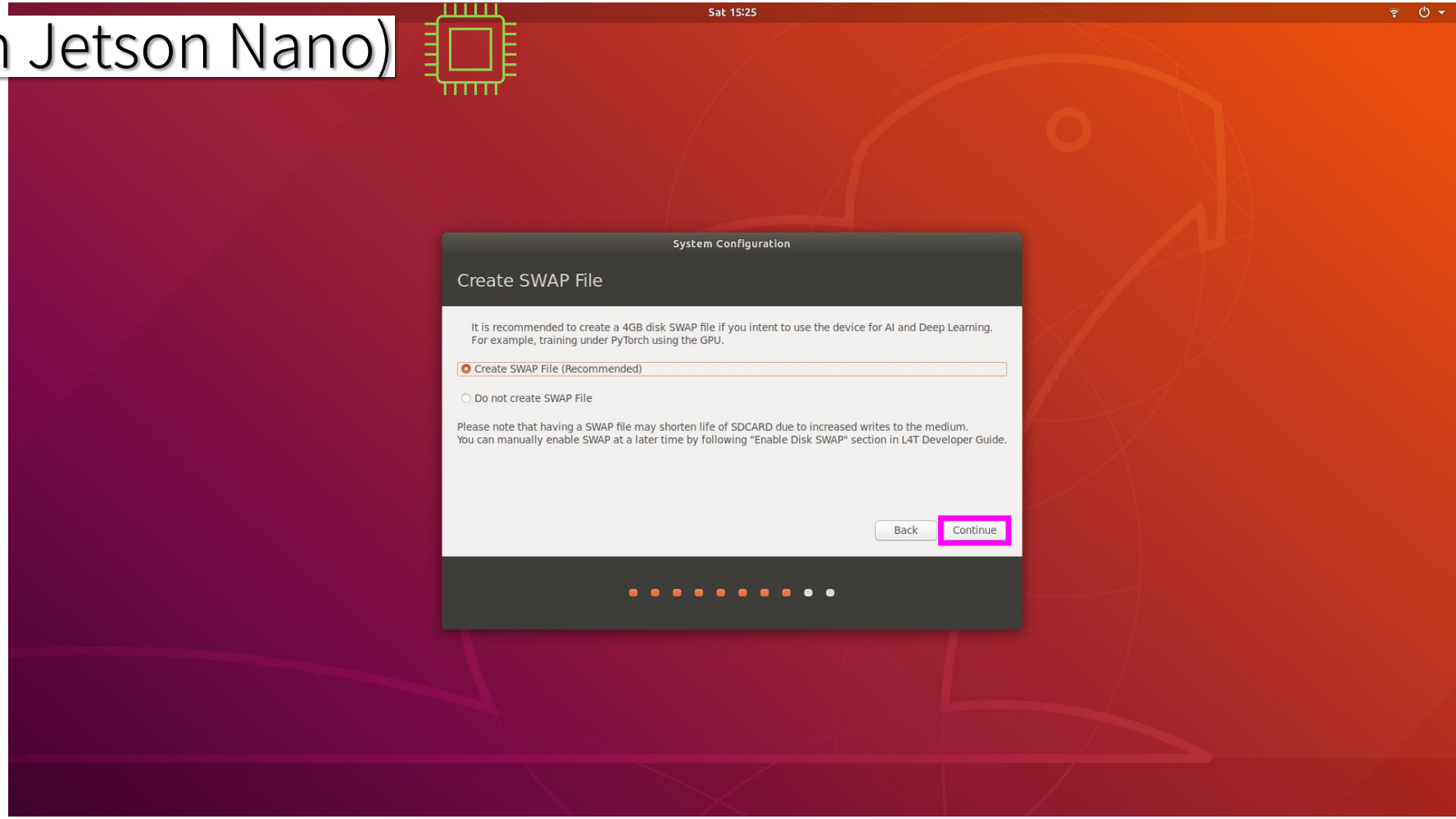
Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)



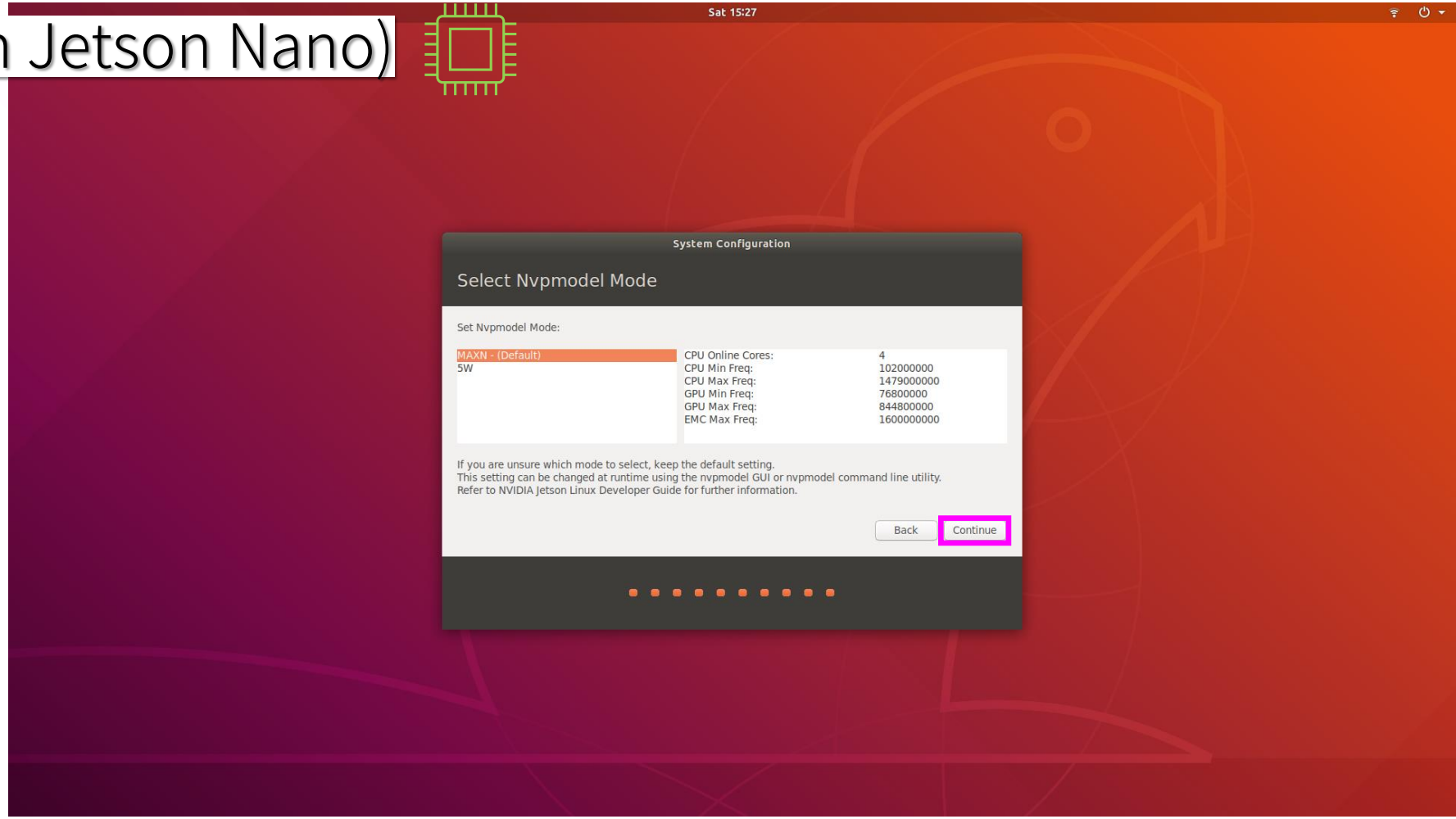
Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)



Jetson Nano 2GB Installation

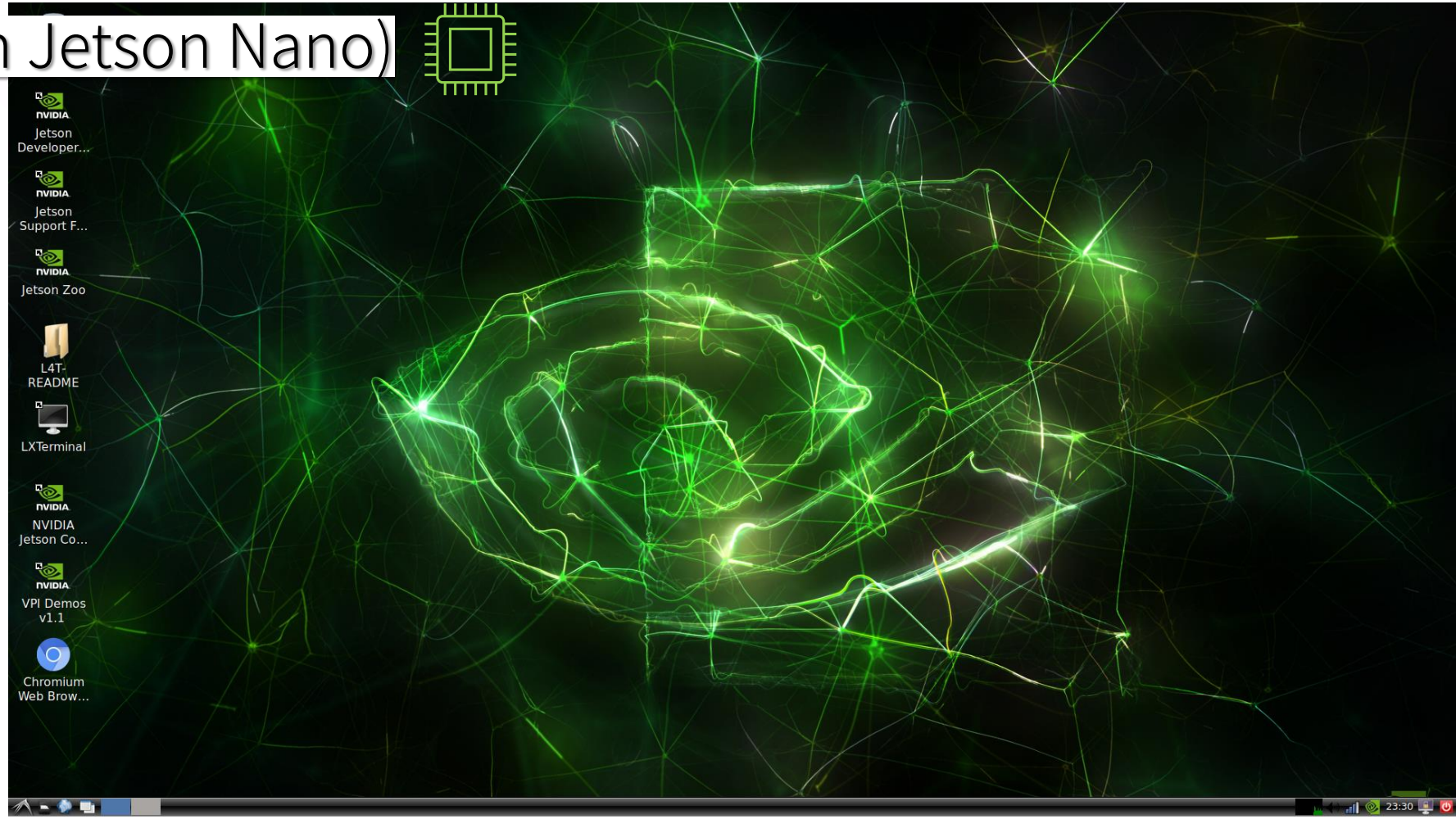
2. 組裝並開機 (on Jetson Nano)



Jetson Nano 2GB Installation

2. 組裝並開機 (on Jetson Nano)

完成開機



Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano)

   a. 取得 Jetson Nano 2GB IP 位址

      ① 開啟 LXTerminal

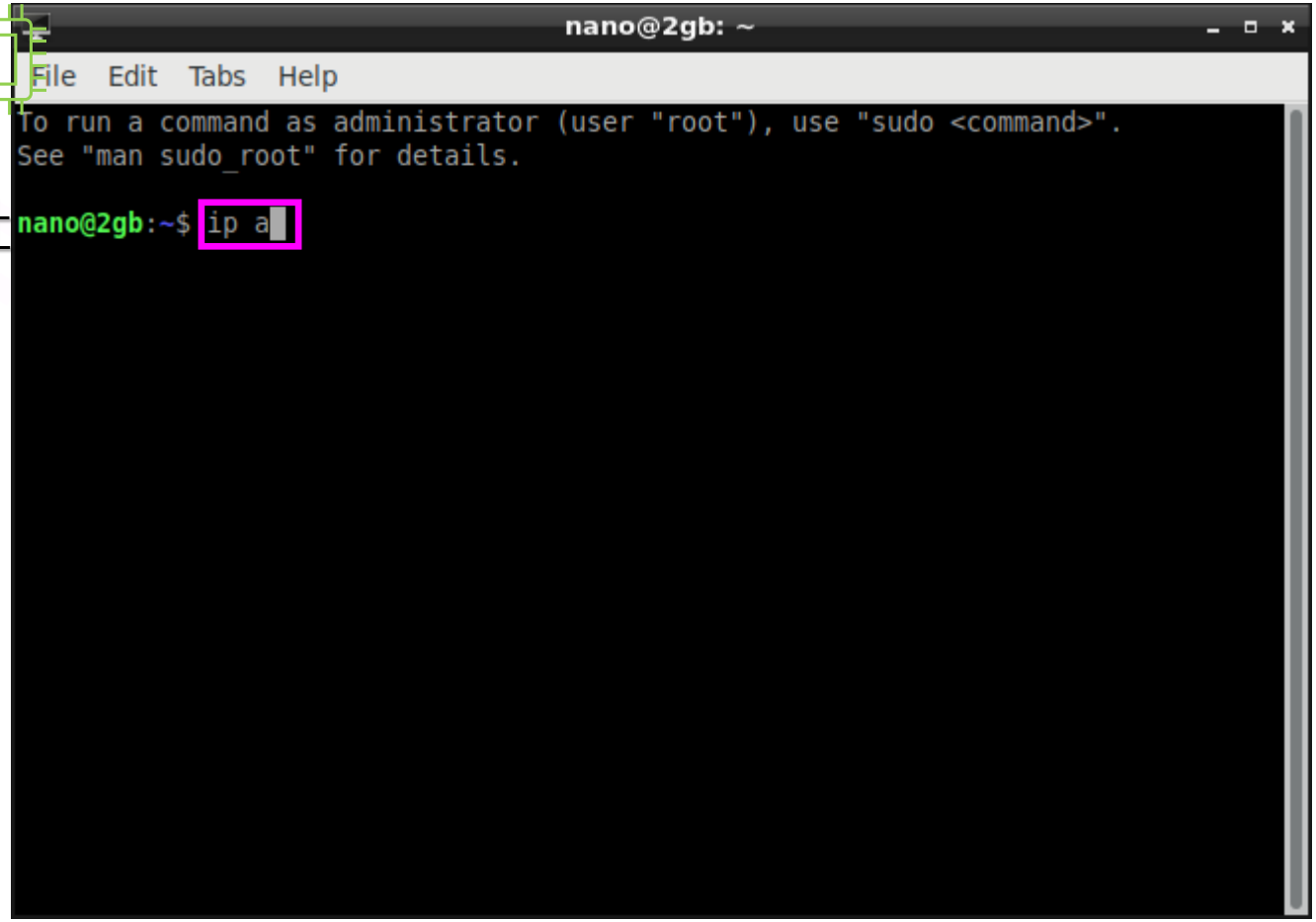
Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano)

a. 取得 Jetson Nano 2GB IP 位址

① 開啟 LXTerminal

② `ip a` ENTER



Jetson Nano 2GB Installation

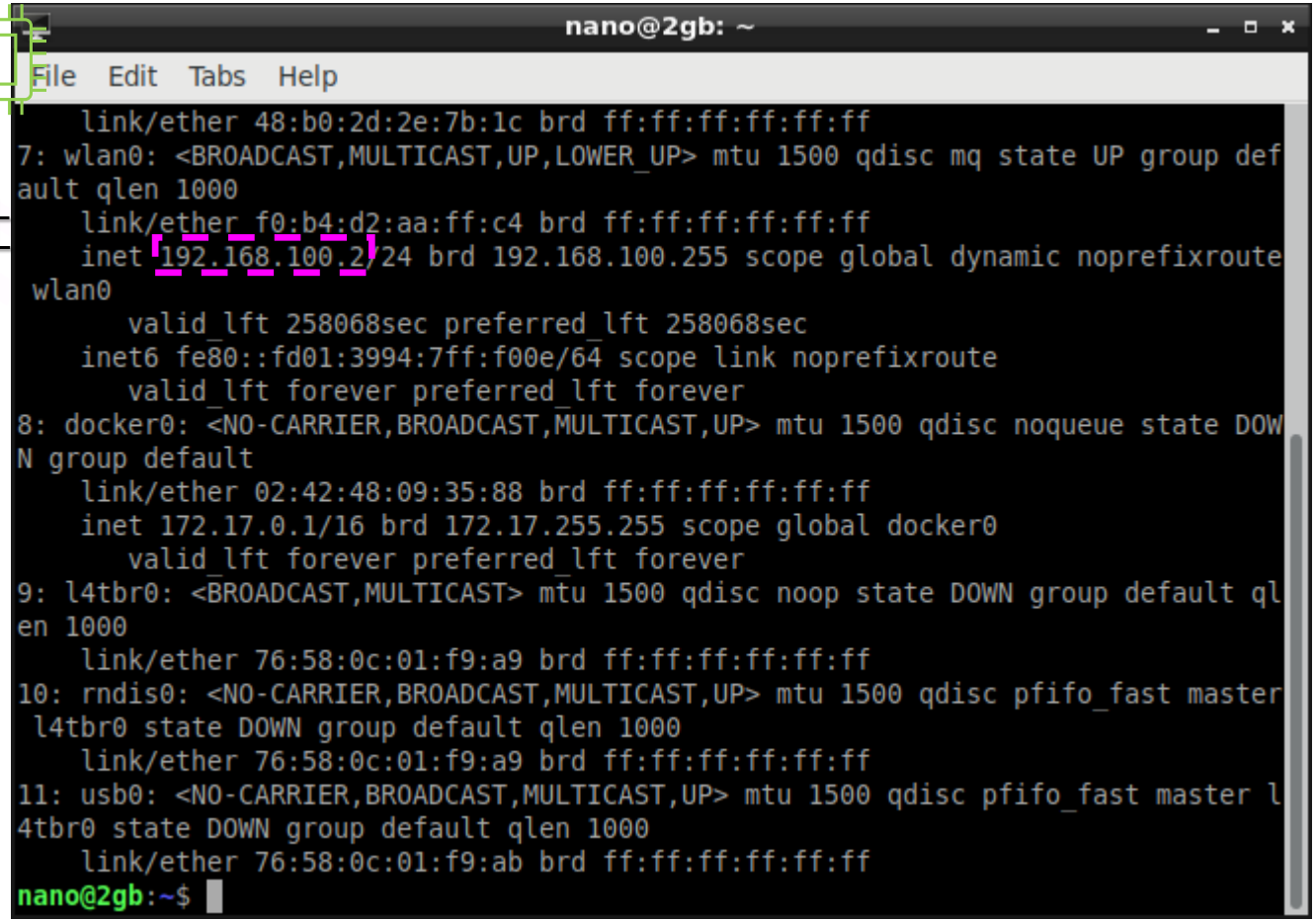
3. 初始化 (on Jetson Nano)

a. 取得 Jetson Nano 2GB IP 位址

① 開啟 LXTerminal

② `ip a`

③ 取得 IP 位址



Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano) 

b. 遠端連線 (feat. Windows 10)

下載 MobaXterm Home Edition (Portable edition)

① https://mobaxterm.mobatek.net/ >
Download >
Home Edition: Download now >
MobaXterm Home Edition v22.1 (Portable edition)

② MobaXterm_Portable_v22.1.zip

③ 解壓縮 MobaXterm_Portable_v22.1.zip ⇨ MobaXterm_Personal_22.1.exe

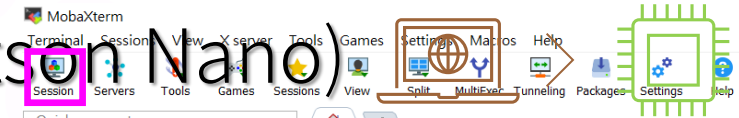
Jetson Nano 2GB Installation

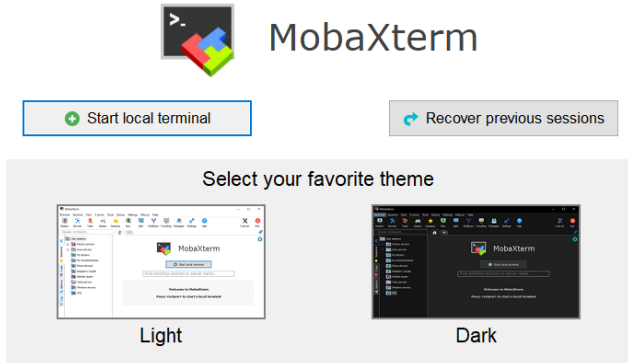
3. 初始化 (on Jetson Nano)

b. 遠端連線 (feat. Windows 10)

① 執行 MobaXterm_Personal_22.1.exe

② Session



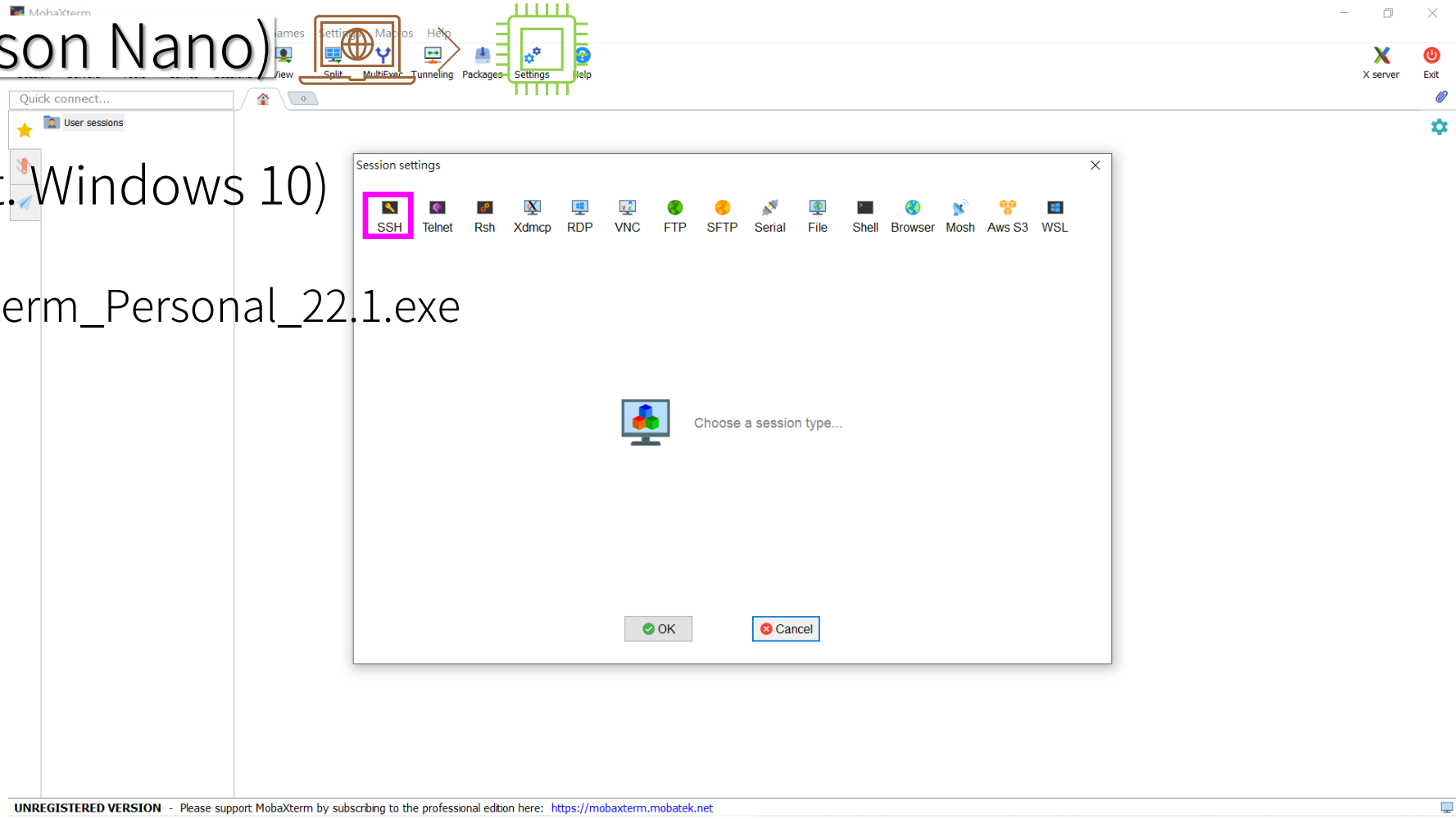


Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano)

b. 遠端連線 (feat. Windows 10)

① 執行 MobaXterm_Personal_22.1.exe
② Session
③ SSH



Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano)

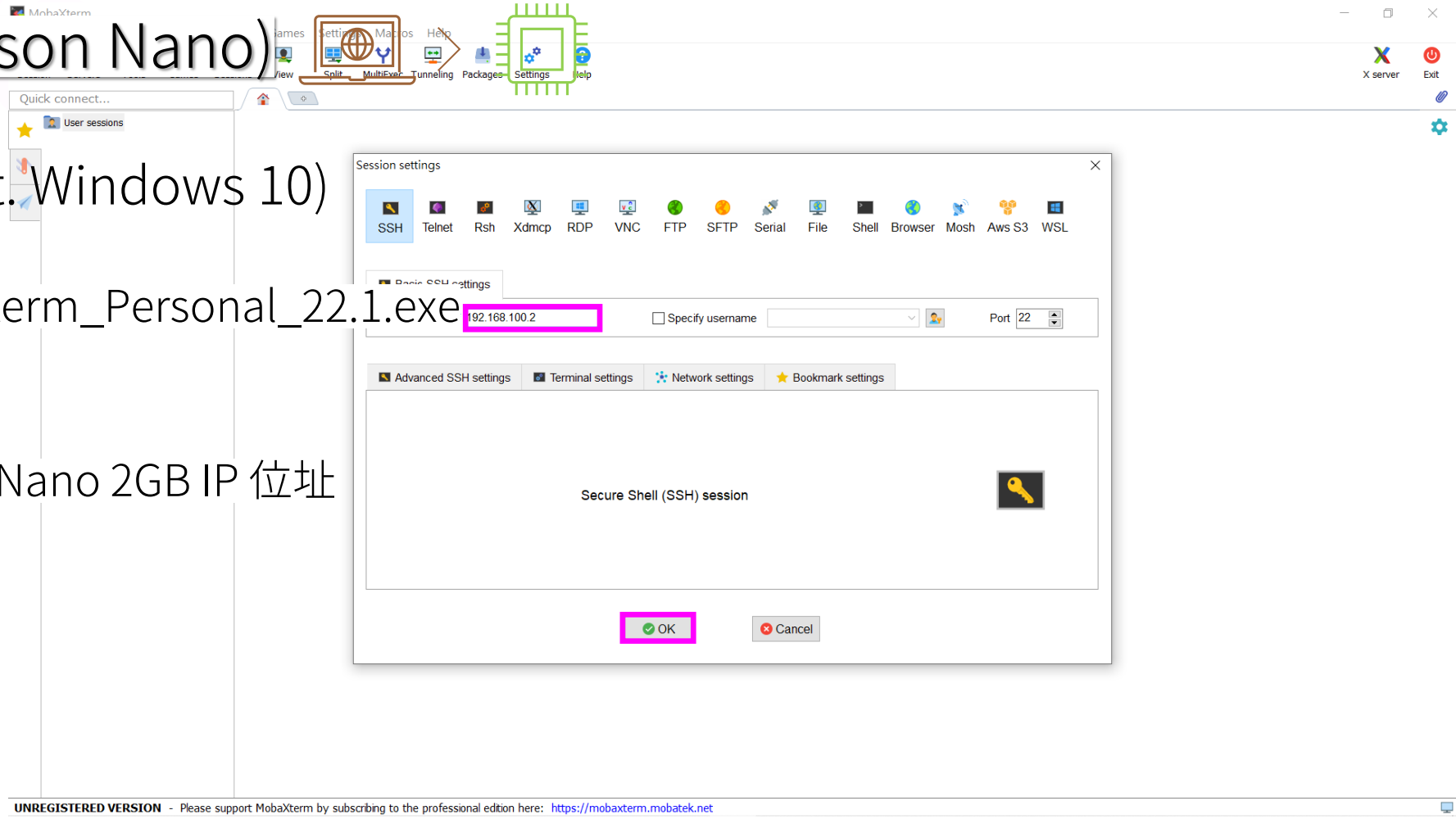
b. 遠端連線 (feat. Windows 10)

① 執行 MobaXterm_Personal_22.1.exe

② Session

③ SSH

④ 填入 Jetson Nano 2GB IP 位址

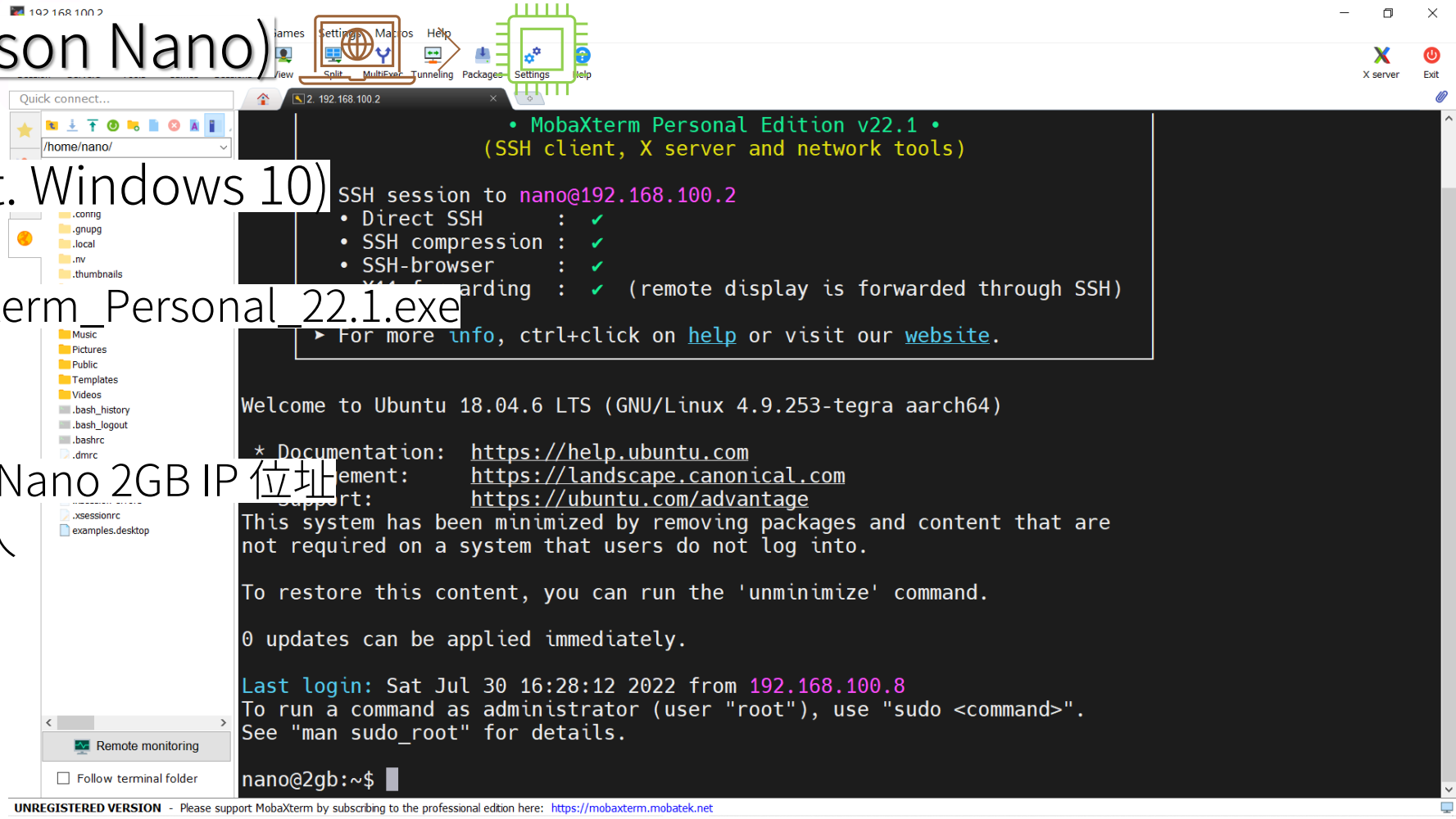


Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano)

b. 遠端連線 (feat. Windows 10)

① 執行 MobaXterm_Personal_22.1.exe
② Session
③ SSH
④ 填入 Jetson Nano 2GB IP 位址
⑤ 帳號密碼登入



Jetson Nano 2GB Installation

3. 初始化 (on Jetson Nano) 

c. 安裝基礎套件 (約 2 分)

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② `sudo apt-get update; sudo apt-get install -y python3-pip; pip3 install --upgrade pip`

Jetson Faces

Jetson Faces

1. 安裝套件 (on Jetson Nano)
2. 註冊與點名 (on Jetson Nano)
3. 物連網的註冊與點名

Jetson Faces

1. **安裝套件** (on Jetson Nano) 

    a. 套件 dlib & face_recognition (約 44 分)

        ① 開啟 Terminal (on LXTerminal/ MobaXterm)

        ②
```
# targets: Cython (0.29.32), face_recognition (1.3.0)
# dependencies: dlib >= 19.7 (19.24), Pillow (8.4.0)
sudo apt-get install -y libopenblas-dev liblapack-dev libjpeg-dev
libfreetype6-dev libcanberra-gtk-module; python3 -m pip install --no-
warn-script-location Cython face_recognition
```

Jetson Faces

1. 安裝套件 (on Jetson Nano) 

b. 點名應用 rollcall_edge

① 開啟 Terminal (on LXTerminal/ MobaXterm)

②
```
# targets: paho-mqtt (1.3.1), rollcall
sudo apt-get install -y python3-paho-mqtt; cd; git clone
https://github.com/catchsob/facecam; wget --no-check-certificate
'https://drive.google.com/u/0/uc?id=1eGAsTN1HBpJAkeVM57_C7ccp7hbgSz3_&exp
ort=download' -O facecam/TaipeiSansTCBeta-Regular.ttf
```

Jetson Faces

2. 註冊與點名 (on Jetson Nano)

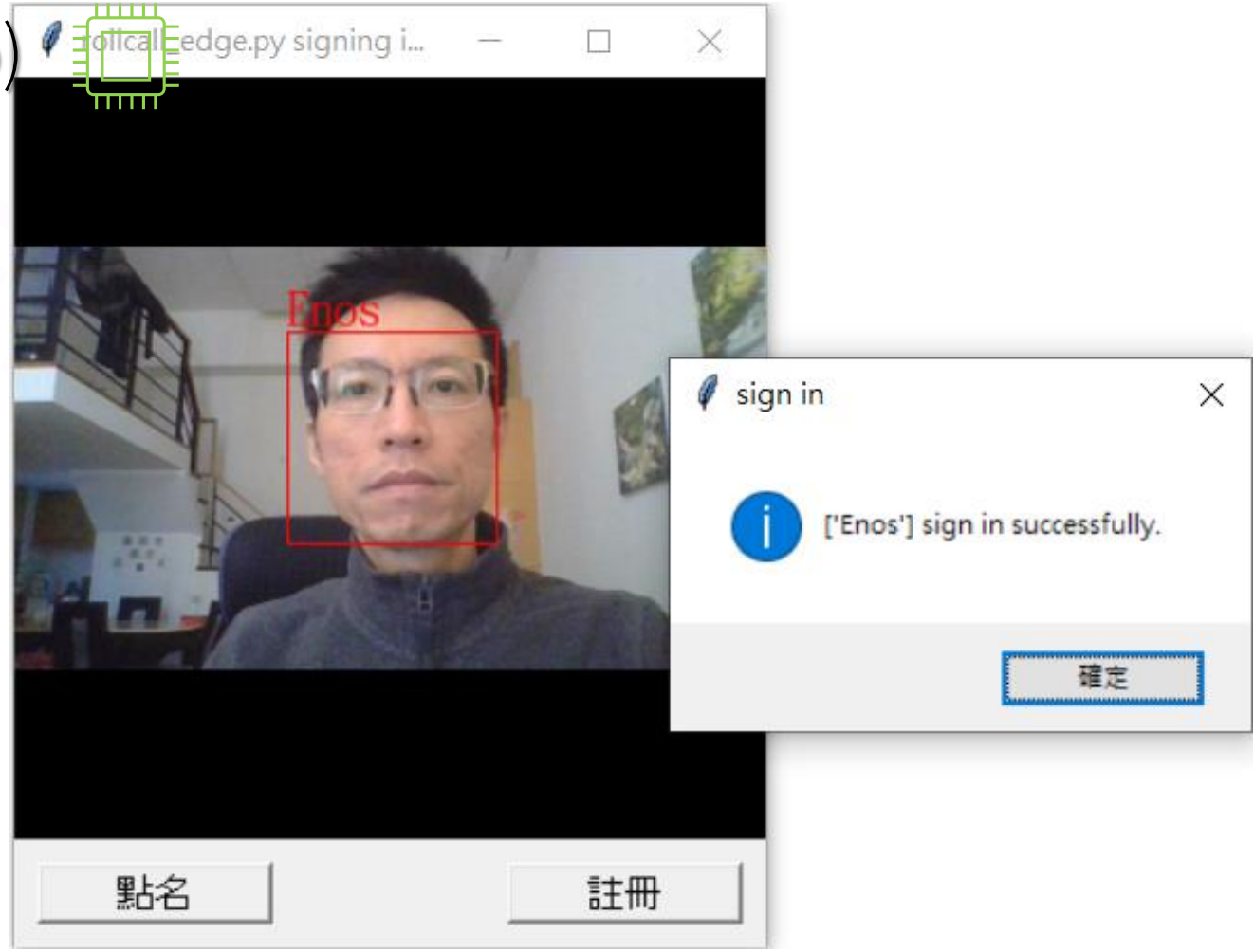
a. 啟動 rollcall_edge

① 開啟 LXTerminal

② `cd; cd facecam`

③ `python3 rollcall_edge.py`
`python3 rollcall_edge.py -c`

④ 先 註冊，再 點名



Jetson Faces

2. 註冊與點名 (on Jetson Nano) 

b. trace rollcall_edge.py

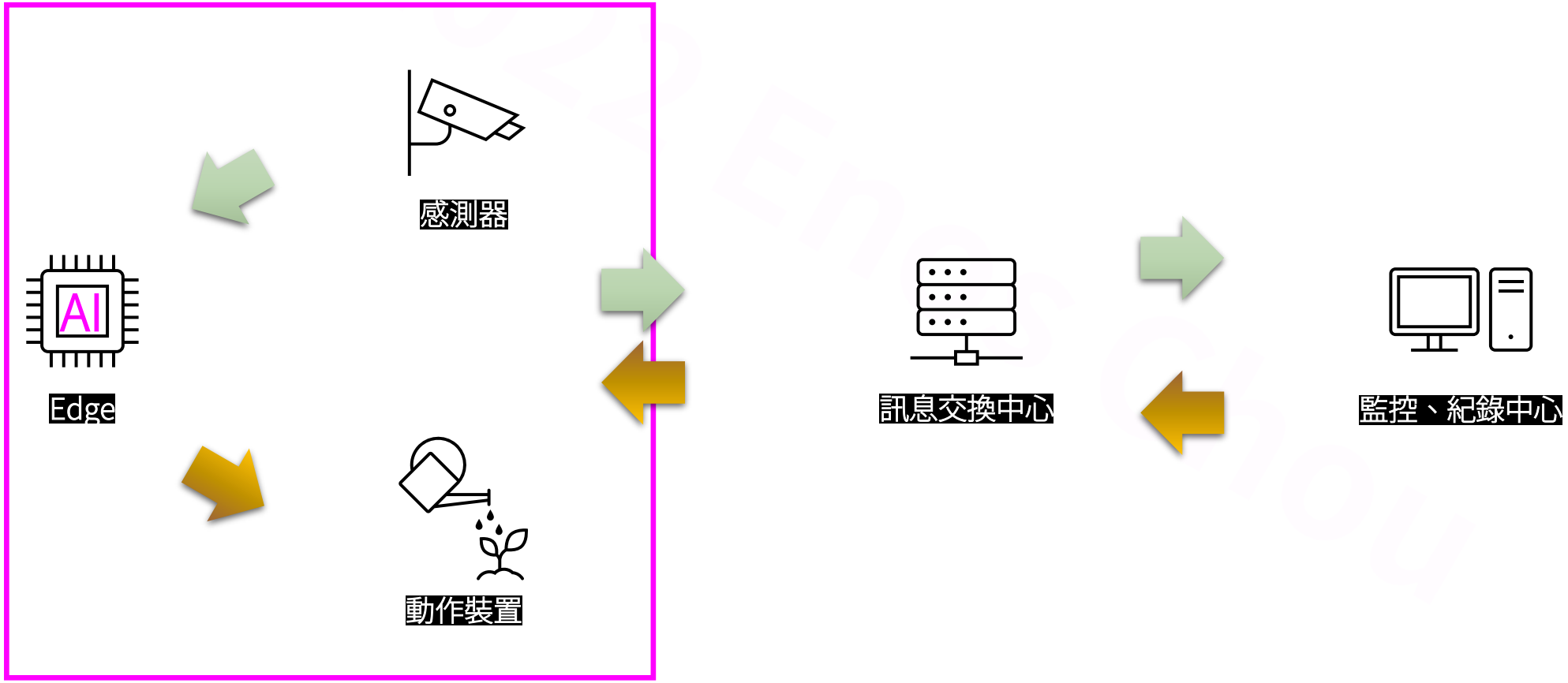
face_recognition

```
ens = face_recognition.face_encodings(face)  # encode the faces
locs = face_recognition.face_locations(face)  # get the face frame
distance = face_recognition.face_distance(faces.encodes, ens[e])  # compute the
distance with each face
facei = distance.argmin()  # decide the most likly face
```

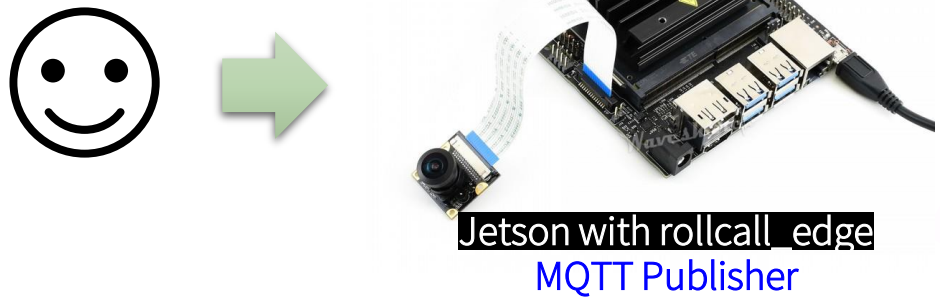
Jetson Faces

3. 物連網的註冊與點名 

Recap AIoT with Edge



Faces AIoT



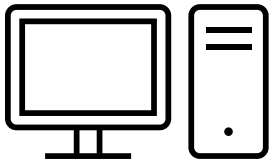
Jetson with rollcall_edge
MQTT Publisher

MQTT publish

GCE

mosquitto
MQTT Broker

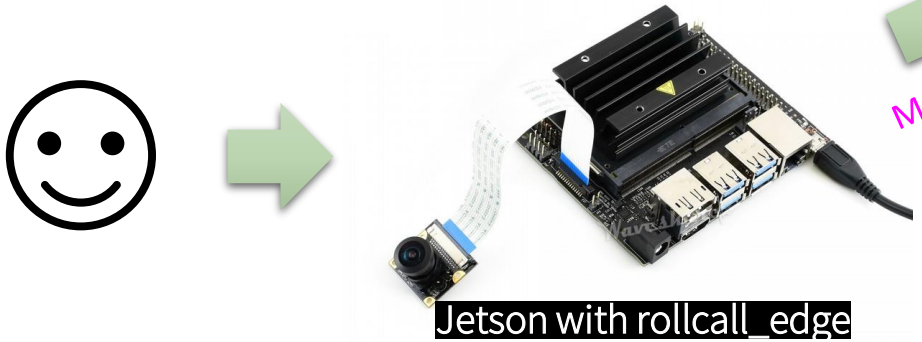
MQTT subscribe



Application
MQTT Subscriber

MQTT subscribe

Node-RED
Management Panel
MQTT Subscriber



MQTT publish

Jetson with rollcall_edge
MQTT Publisher

Faces AIoT

MQTT Topic Example

- `rollcall/signin`
- `rollcall/register`
- `rollcall/#`
- `#`

MQTT Message Example

- `Someone on 2022/02/15`

AIoT with MQTT

1. Server as MQTT Broker
2. Server as MQTT Subscriber
3. Client as MQTT Publisher

AIoT with MQTT

1. Server as MQTT Broker
    a. mqttgo.io

AIoT with MQTT

1. Server as MQTT Broker

b. Ubuntu 18.04 on GCE

① 登入 console.cloud.google.com
② 建立並切換至新專案
③ 啟用 Compute Engine
④ Compute Engine > VM 執行個體 > 建立執行個體
- 名稱 mqtt
- 區域 us-west1/ 區域 us-west1-b
- 機器類型 e2-micro
- 開機磁碟 > 作業系統 Ubuntu/ 版本 Ubuntu 18.04 LTS x86/64
- 進階選項 > 網路 > 網路介面 > 外部 IPv4 位址 > 建立 IP 位址 > mqttip

AIoT with MQTT

1. Server as MQTT Broker

b. Ubuntu 18.04 on GCE

⑤ 虛擬私有雲網路 > 防火牆 > 建立防火牆規則
- 名稱 mqttfw
- 目標 網路中的所有執行個體
- 來源 IPv4 範圍 0.0.0.0/0
- 通訊協定和埠 > 指定的通訊協定和埠 > TCP 1883

⑥ (optional) DNS 指向外部 IP 位址 (mqttip)

⑦ SSH 連線 GCE VM 執行個體 mqtt

```
sudo apt-get update; sudo apt-get install -y mosquitto; sudo
timedatectl set-timezone Asia/Taipei
```

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

套件 paho-mqtt

```
pip install paho-mqtt
```

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

官方範例

https://github.com/eclipse/ > Repositories: paho.mqtt.python

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("rollcall/#")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
```

AIoT with MQTT

2. Server as MQTT Subscriber 

Python 3 Environment

```
YOUR_BROKER = 'YOUR_BROKER'

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(YOUR_BROKER, 1883, 60)
client.loop_forever()
```

AIoT with MQTT

3. Client as MQTT Publisher 

Python 3 Environment

```
from time import strftime

from paho.mqtt import publish

YOUR_BROKER = 'YOUR_BROKER'
topic = 'rollcall'
msg = f'Mary sign in on {strftime("%Y/%m/%d-%H:%M:%S")}'
publish.single(topic, msg, hostname=YOUR_BROKER)
```

AIoT with MQTT

3. Client as MQTT Publisher

Python 3 Environment

```
msgs = [['rollcall/register', f'Jerry on {strftime("%Y/%m/%d-%H:%M:%S")}'],
        ['rollcall/signin', f'Eric on {strftime("%Y/%m/%d-%H:%M:%S")}']
       ]
publish.multiple(msgs, hostname=YOUR_BROKER)
```

Jetson Faces

3. 物連網的註冊與點名

建立 MQTT Broker (on GCE)

請參閱

Jetson Faces

3. 物連網的註冊與點名

啟動 rollcall_edge AIoT 模式 (on Jetson Nano)

① 開啟 LXTerminal

② `cd; cd facecam`

③ `python3 rollcall_edge.py -m YOUR_BROKER`
`python3 rollcall_edge.py -c -m mqttgo.io`

Jetson Trees

TensorFlow, TensorRT

Trees Inference

Powered by TensorFlow

1. 安裝軟體 (on Jetson Nano)
2. 叫用模型 (on Jetson Nano)

Powered by TensorRT

1. 轉換模型 (on Jetson Nano)
2. 叫用模型 (on Jetson Nano)

Trees Powered by TensorFlow

1. **安裝軟體** (on Jetson Nano) 

    a. 套件 TensorFlow 2.5.0 (約 20 分)

        ① 開啟 Terminal (on LXTerminal/ MobaXterm)

        ②
```
# targets: tensorflow 2.5.0
# dependencies: protobuf 3.19.4, grpcio 1.34.1, h5py-2.10.0
# numpy 1.19.5 would cause core dump, adopt 1.19.4 instead
sudo apt-get install -y python3-testresources libhdf5-serial-dev hdf5-tools libhdf5-dev; python3 -m pip install --no-warn-script-location numpy==1.19.4
https://developer.download.nvidia.com/compute/redist/jp/v46/tensorflow/tensorflow-2.5.0+nv21.8-cp36-cp36m-linux_aarch64.whl
```

Trees Powered by TensorFlow

1. **安裝軟體** (on Jetson Nano) 

   b. 模型與應用程式

      ① 開啟 Terminal (on LXTerminal/ MobaXterm)

      ② `cd; git clone https://github.com/catchsob/trees17`

Trees Powered by TensorFlow

2. 叫用模型 (on Jetson Nano) 

Inference

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② `cd; cd trees17`

③ `python3 trees-infer.py YOUR_HDF5.h5 YOUR_TREES -l YOUR_LABELS`

```
python3 trees-infer.py trees17V1.h5 構樹.jpg -l treeset_labels.txt
```

Trees Powered by TensorRT

1. **轉換模型** (on Jetson Nano) 

   HDF5 ⇨ ONNX ⇨ TensorRT

Trees Powered by TensorRT

1. 轉換模型 (on Jetson Nano) 

HDF5 ⇨ ONNX ⇨ TensorRT

套件 tf2onnx (約 1 分)

① 開啟 Terminal (on LXTerminal/ MobaXterm)

②
```
# targets: onnxruntime-gpu 1.11.0
cd; wget https://nvidia.box.com/shared/static/pmsqsiaw4pg9qrbeckcbymho6c01jj4z.whl -O onnxruntime_gpu-1.11.0-cp36-cp36m-linux_aarch64.whl; python3 -m pip install --no-warn-script-location onnxruntime_gpu-1.11.0-cp36-cp36m-linux_aarch64.whl
```

③
```
# targets: onnx 1.11.0, tf2onnx (1.12.1); packaging (17.1) is newly added
sudo apt-get install -y protobuf-compiler libprotobuf-dev python3-packaging; python3 -m pip install onnx==1.11.0 tf2onnx
```

Trees Powered by TensorRT

1. 轉換模型 (on Jetson Nano) 

HDF5 ⇨ ONNX ⇨ TensorRT

HDF5 轉 ONNX (約 6 分)

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② `cd; cd trees17`

③ `python3 -m tf2onnx.convert --keras YOUR_HDF5.h5 --output YOUR_ONNX.onnx`

```
python3 -m tf2onnx.convert --keras trees17V1.h5 --output trees17V1.onnx
```

Trees Powered by TensorRT

1. 轉換模型 (on Jetson Nano) 

HDF5 ⇨ ONNX ⇨ TensorRT

ONNX 轉 TensorRT (約 4 分)

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② `cd; cd trees17`

③ `/usr/src/tensorrt/bin/trtexec --onnx=`*`YOUR_ONNX`*`.onnx --shapes=`*`YOUR_INPUT_SHAPE`*` --workspace=`*`YOUR_WORKSPACE`*` --saveEngine=`*`YOUR_TENSORRT`*`.trt`

```
/usr/src/tensorrt/bin/trtexec --onnx=trees17V1.onnx --
saveEngine=trees17V1.trt
```

Trees Powered by TensorRT

2. 叫用模型 (on Jetson Nano) 

a. 套件 pycuda (約 5 分)

① 開啟 Terminal (on LXTerminal/ MobaXterm)

②
```
# 直接執行 pip3 install pycuda 會報錯，須先行指定路徑
# 安裝完畢會報錯 typing-extensions 版本衝突，不須理會
export CPATH=$CPATH:/usr/local/cuda-10.2/targets/aarch64-linux/include;
export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/cuda-10.2/targets/aarch64-
linux/lib; python3 -m pip install pycuda
```

Trees Powered by TensorRT

2. 叫用模型 (on Jetson Nano) 

b. Inference

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② `cd; cd trees17`

③ `python3 trees-infer.py YOUR_TENSORRT.trt YOUR_TREES -l YOUR_LABELS`

```
python3 trees-infer.py trees17V1.trt 鳳凰木.jpg -l treeset_labels.txt
```

Trees Powered by TensorRT

2. 叫用模型 (on Jetson Nano) 

c. Trace trees-infer.py

```
runtime = trt.Runtime(trt.Logger(trt.Logger.WARNING))  # initialize TersorRT
with open(args.engine, "rb") as f:
    engine = runtime.deserialize_cuda_engine(f.read())  # create TensorRT engine
inputs, outputs, bindings, stream = common.allocate_buffers(engine)  # buffer
inputs[0].host = np.reshape(img, [-1])  # prepare data
with engine.create_execution_context() as context:
    trt_outputs = common.do_inference_v2(context, bindings=bindings,
inputs=inputs, outputs=outputs, stream=stream)  # inference
preds = trt_outputs[0].reshape(1, -1)  # convert result
```

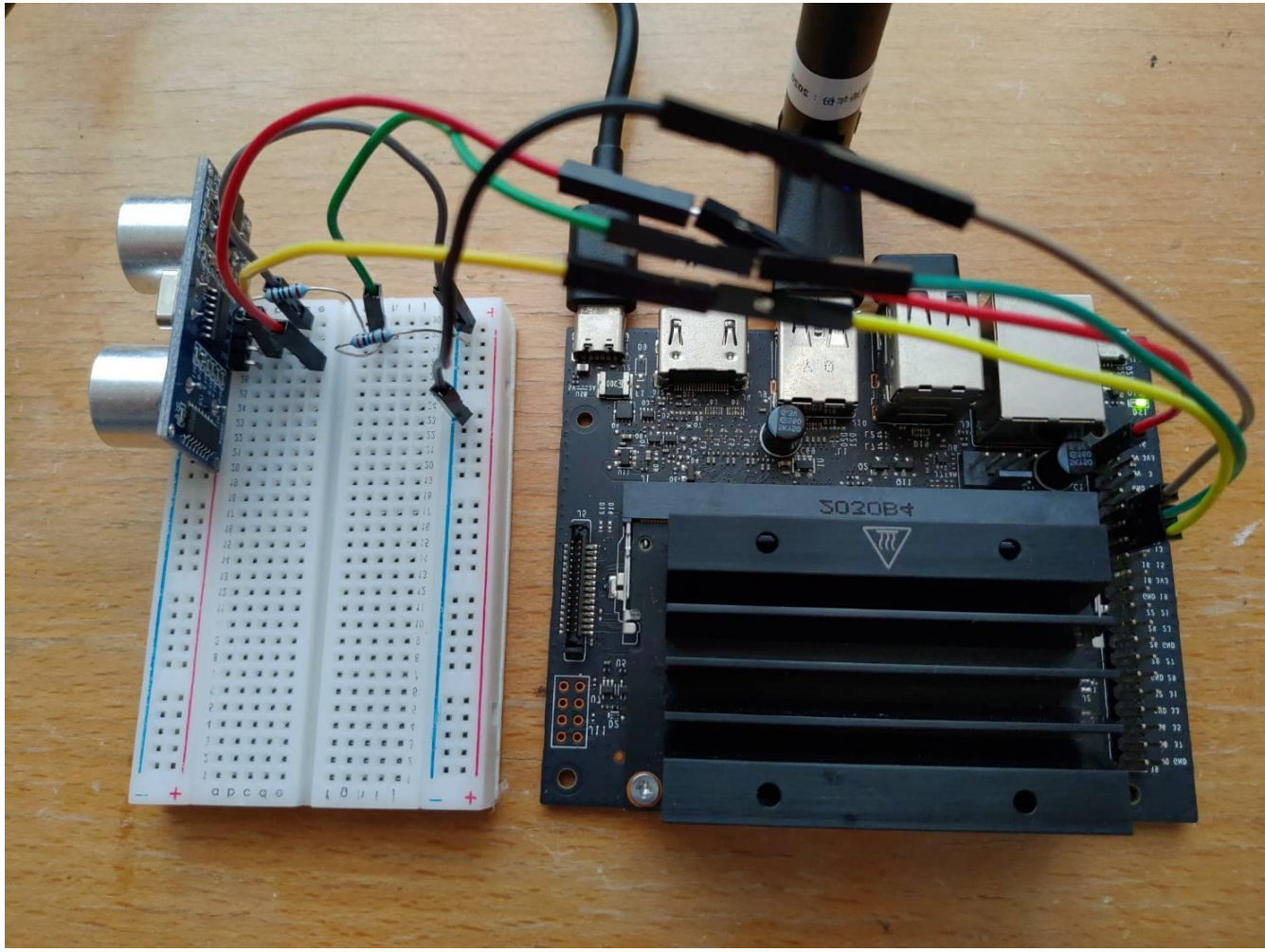
Jetson Distance

Get Distance

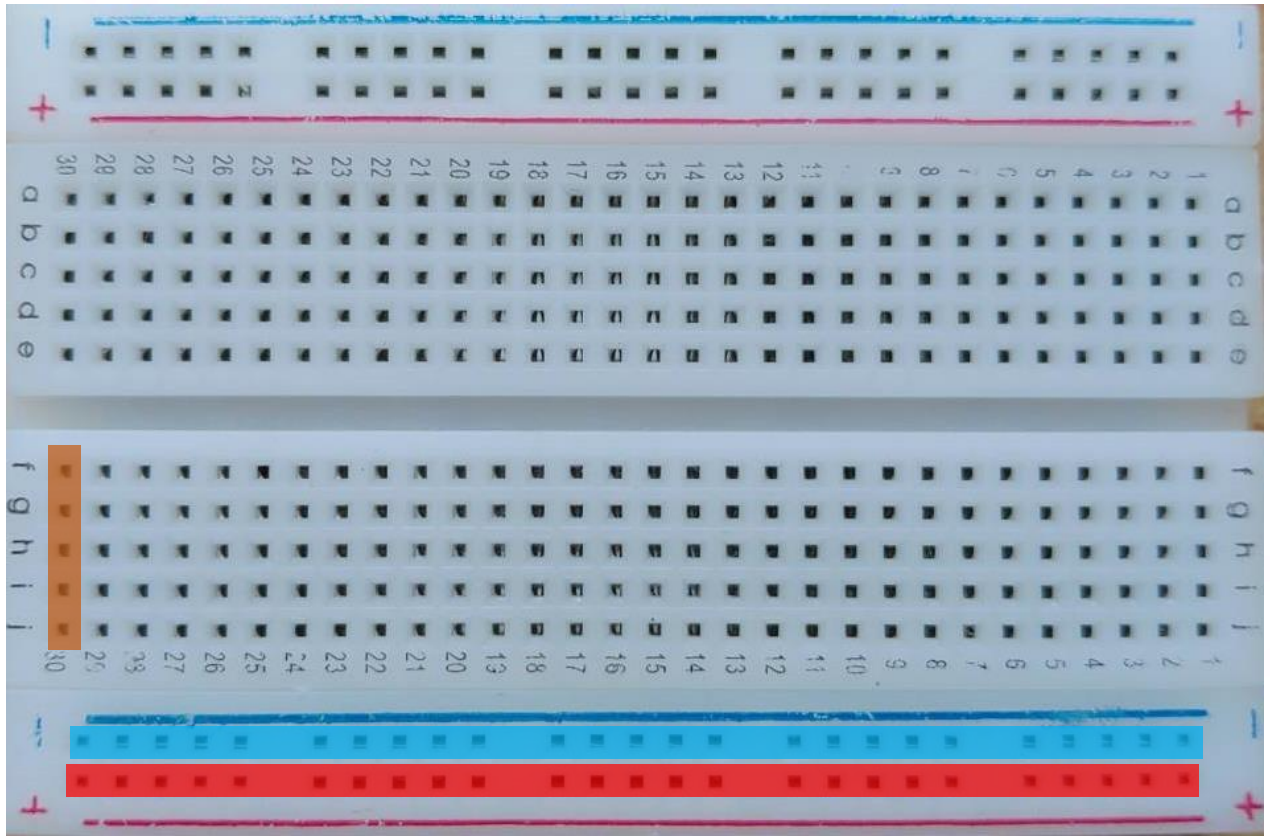
1. 聲納界接

HC-SR04 to Jetson Nano

```
HC-SR04      Breadboard      Jetson
Vcc      -------->------>  5V
Trig     -------->------>  GPIO
Echo     -> 1KΩ --->--->   GPIO
                -> 2KΩ ->  GND
GND      -------->------>  GND
```



Breadboard Conecpt



Get Distance

2. 應用開發 

測試程式

① 開啟 Terminal (on LXTerminal/ MobaXterm)

② 取得 `hcsr04.py`

③ `python3 hcsr04.py -e YOUR_ECHO -t YOUR_TRIGGER`

```
python3 hcsr04.py -e 17 -t 27
```

Your Turn - Watching the Distance

以 Jetson Nano 實作距離監控機制，當距離小於 10cm 以 MQTT 告警

MicroPython Installation on ESP32

ESPRESSIF ESP32-WROOM-32, Ai-Thinker NodeMCU-32

MicroPython Installation on ESP32

系統安裝

a. 下載韌體 (on Windows 10)

b. 準備工具 (on Windows 10)

c. 寫入韌體 (on ESP32 feat. Windows 10)

MicroPython Installation on ESP32

系統安裝

a. 下載韌體 (on Windows 10)

① https://micropython.org/ > DOWNLOAD > ESP32 > Firmware: Releases: v1.19.1 (2022-06-18).bin

② esp32-20220618-v1.19.1.bin

MicroPython Installation on ESP32

系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool (3.3.1)

```
pip install esptool
```

MicroPython Installation on ESP32

系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 於 Windows 10 以 USB-A to MicroUSB Cable 接 ESP32 開發板

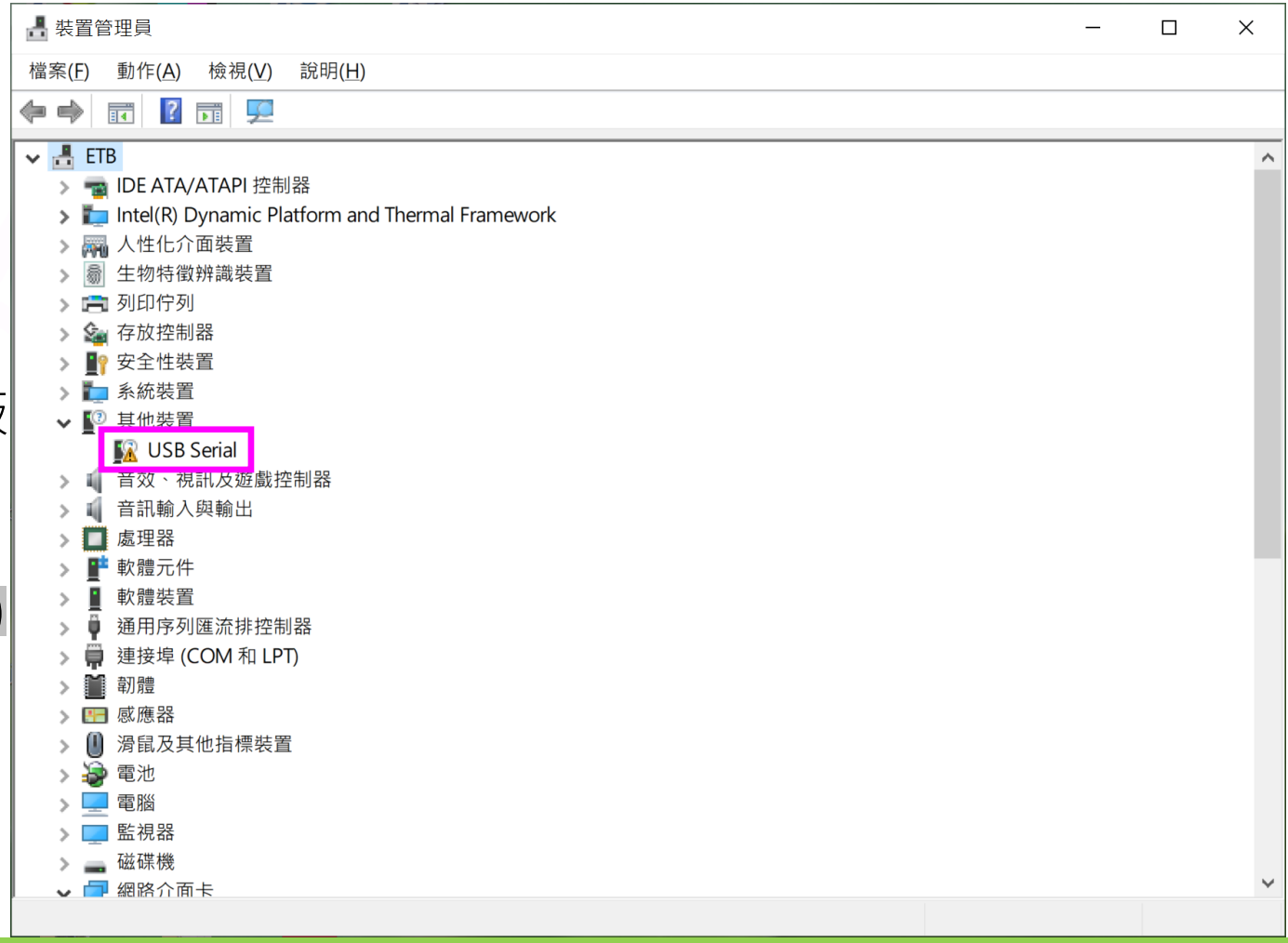
MicroPython Installation on ESP32

系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 以 MicroUSB 接 ESP32 開發板
③ 安裝 USB UART Driver

Windows Key + X > 裝置管理員(M)



MicroPython Installation on ESP32

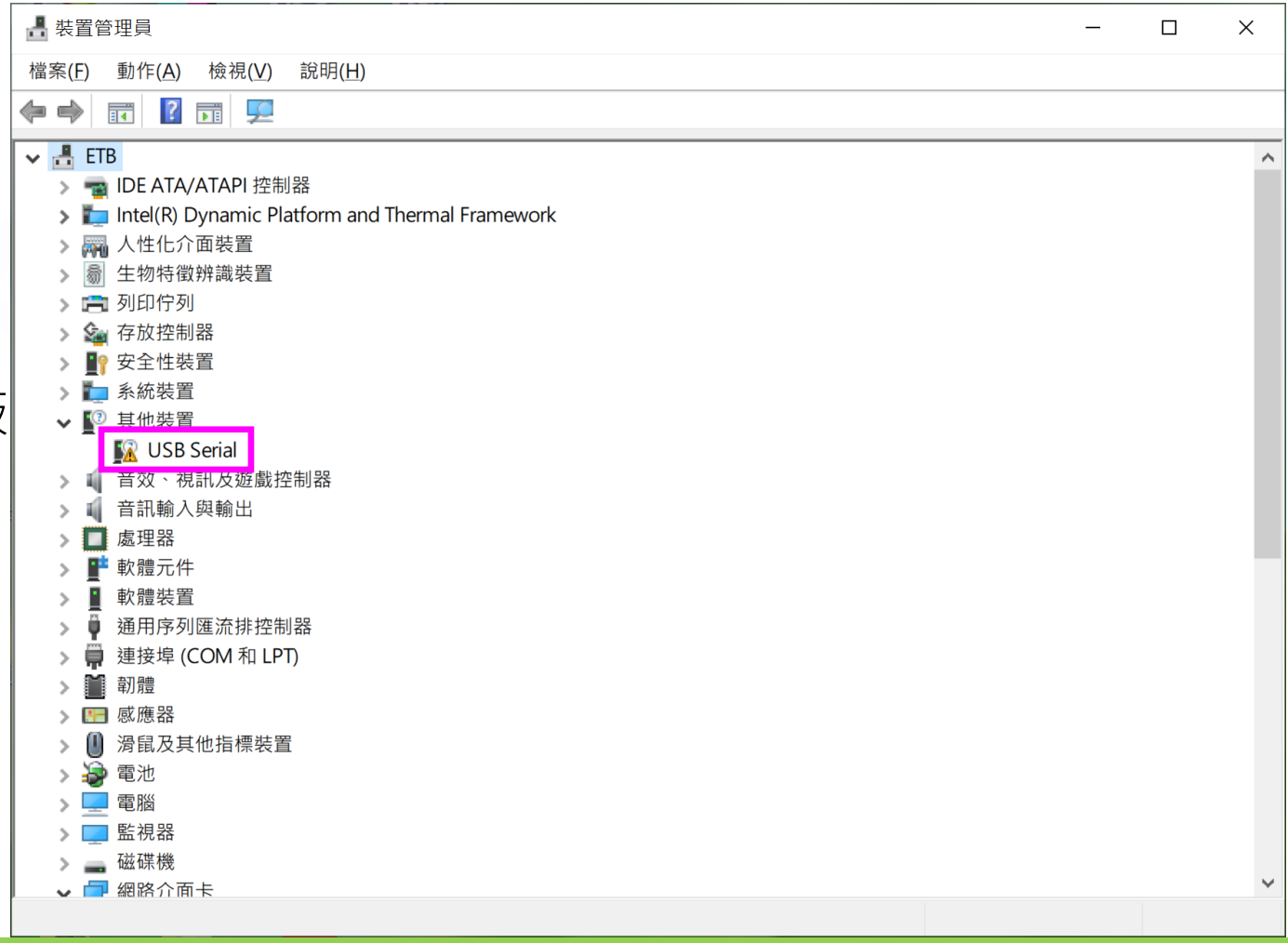
系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 以 MicroUSB 接 ESP32 開發板
③ 安裝 USB UART Driver

檢視晶片型號: CH340C

搜尋、下載、安裝對應 Driver



MicroPython Installation on ESP32

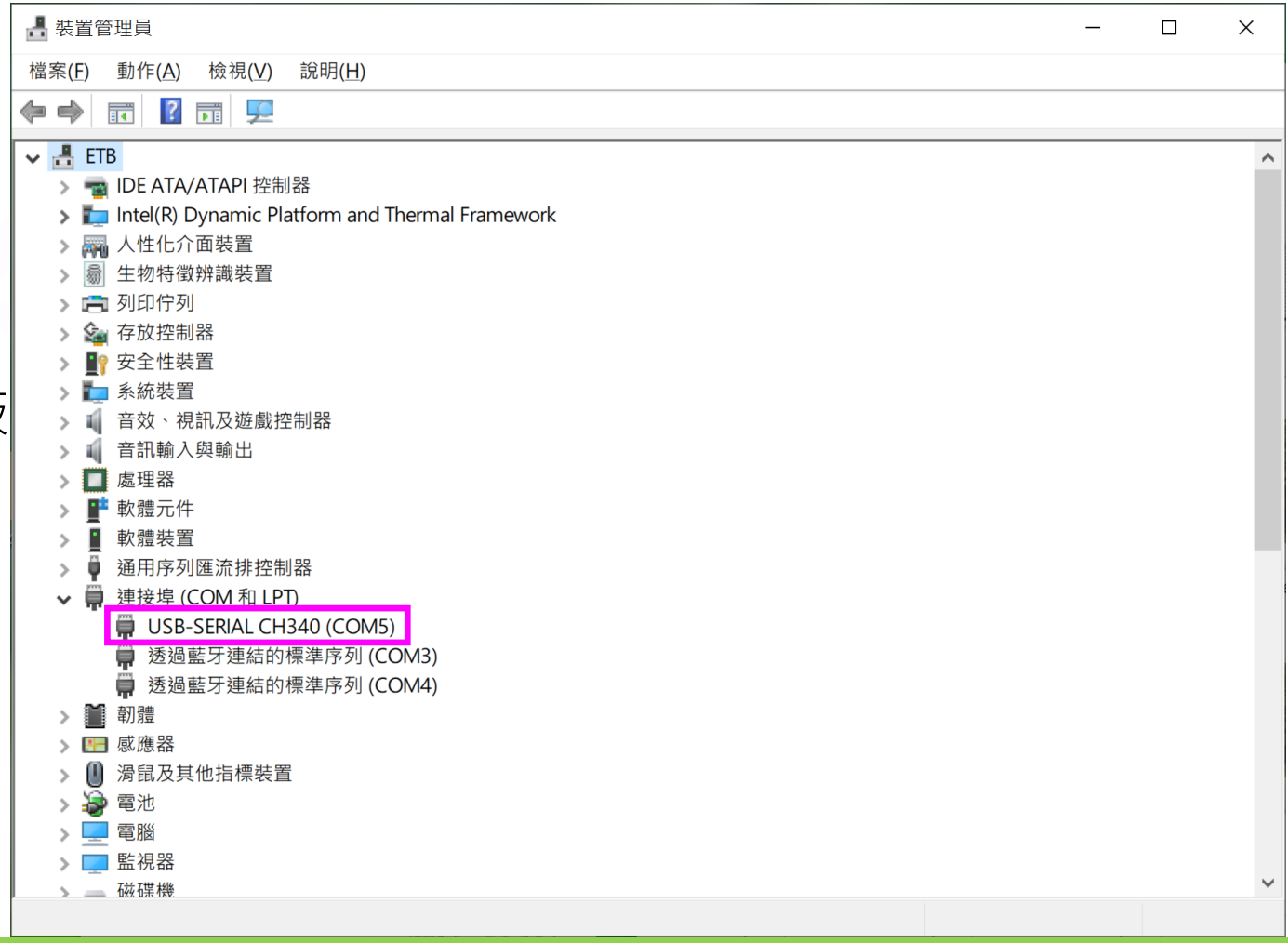
系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 以 MicroUSB 接 ESP32 開發板
③ 安裝 USB UART Driver

檢視晶片型號: CH340C

搜尋、下載、安裝對應 Driver



MicroPython Installation on ESP32

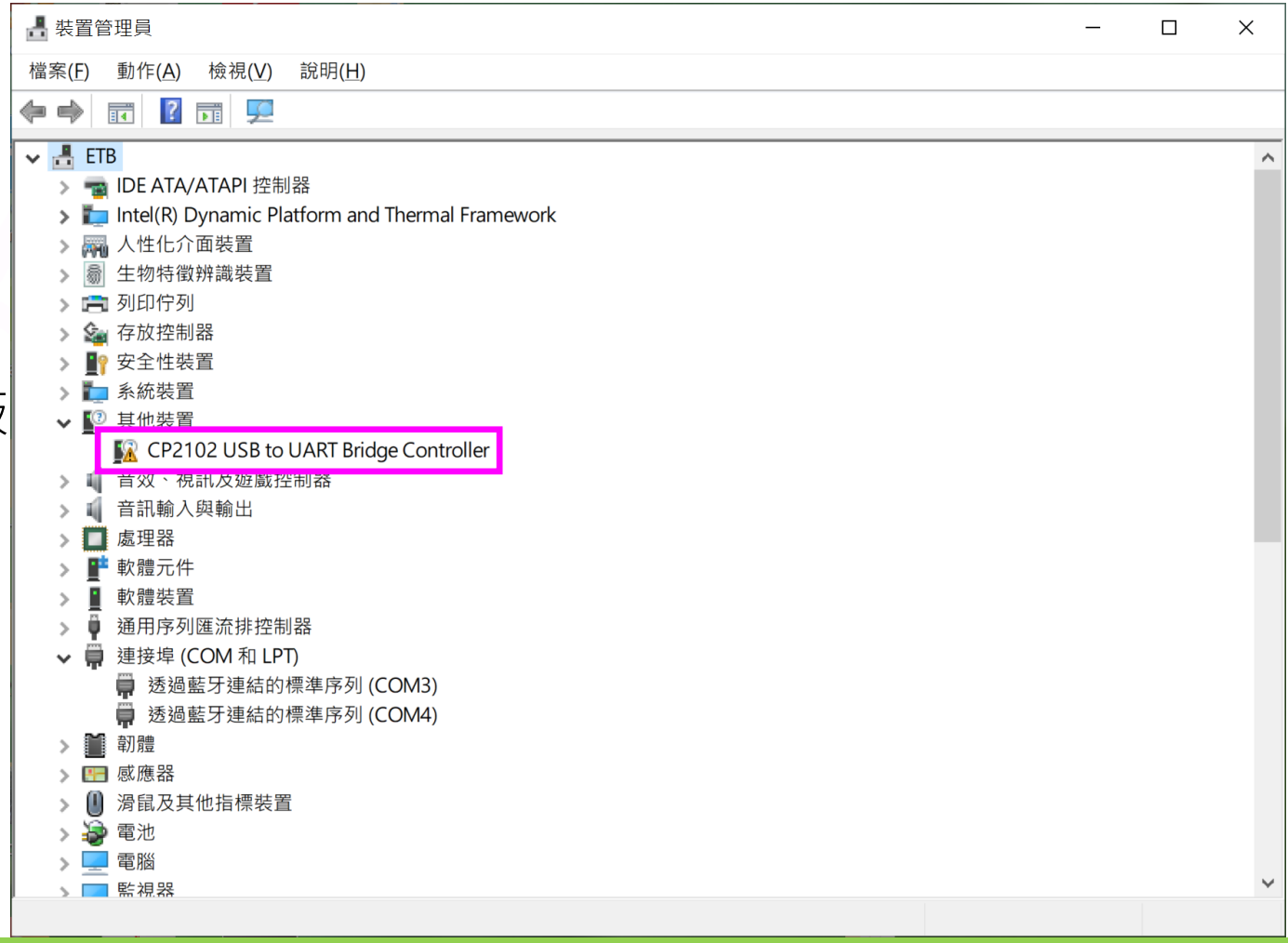
系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 以 MicroUSB 接 ESP32 開發板
③ 安裝 USB UART Driver

檢視晶片型號: CP2102

搜尋、下載、安裝對應 Driver



MicroPython Installation on ESP32

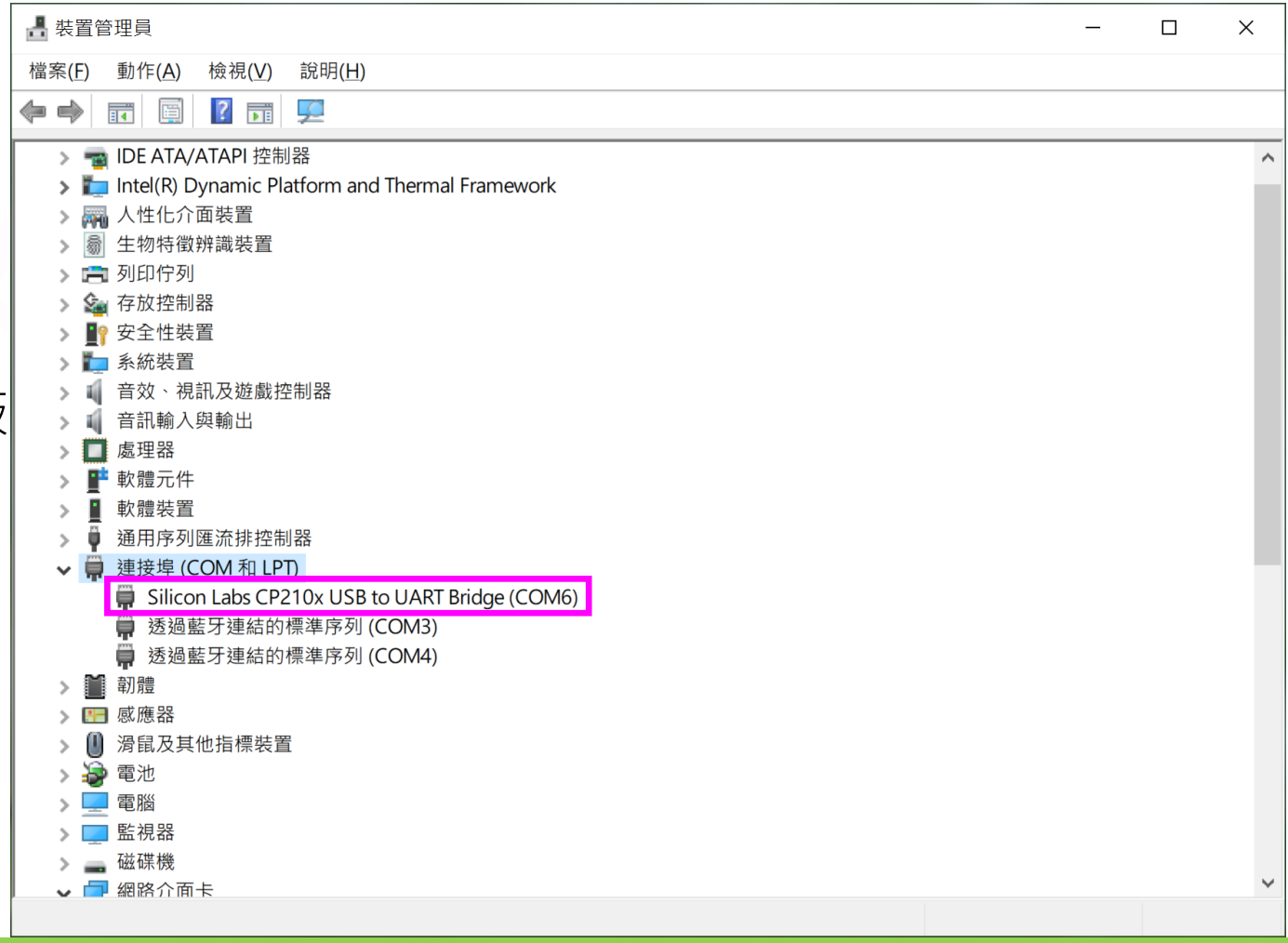
系統安裝

b. 準備工具 (on Windows 10)

① 套件 esptool
② 以 MicroUSB 接 ESP32 開發板
③ 安裝 USB UART Driver

檢視晶片型號: CP2102

搜尋、下載、安裝對應 Driver



MicroPython Installation on ESP32

系統安裝

c. 寫入韌體 (on ESP32 feat. Windows 10)

① 清除 flash

```
esptool --chip esp32 --port YOUR_COM erase_flash
esptool --chip esp32 --port com6 erase_flash
```

MicroPython Installation on ESP32

系統安裝

c. 寫入韌體 (on ESP32 feat. Windows 10)

① 清除 flash
② 寫入 flash

```
esptool --chip esp32 --port YOUR_COM --baud YOUR_BAUDRATE write_flash -z
0x1000 YOUR_FIRMWARE.bin
esptool --chip esp32 --port com6 --baud 460800 write_flash -z 0x1000
esp32-20220618-v1.19.1.bin
```

MicroPython Installation on ESP32

開發工具安裝

a. 下載與安裝 Thonny

- https://thonny.org > Download version 3.3.13 for Windows
- thonny-3.3.13.exe
- 點擊 thonny-3.3.13.exe 安裝 Thonny

MicroPython Installation on ESP32

開發工具安裝

b. 設定 Thonny

- 工具 > 選項 > 直譯器 > Thonny 應該使用哪一個直譯器或設備來執行你的程式？ MicroPython (ESP32) > 確認
- 工具 > 選項 > 直譯器 > 選擇連接埠或是 WebREPL > *YOUR_COM* > 確認

MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

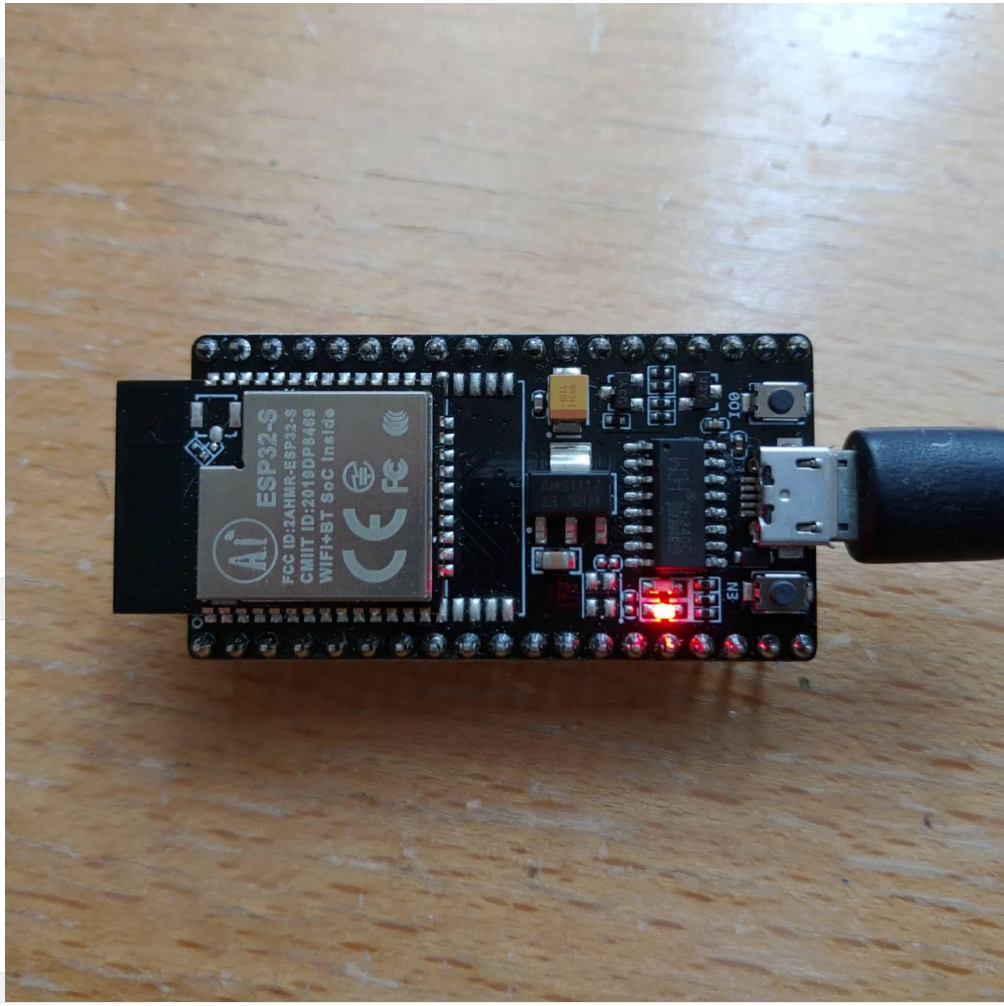
① 啟動 Thonny
② 若互動環境無法 ready 則 STOP

MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32



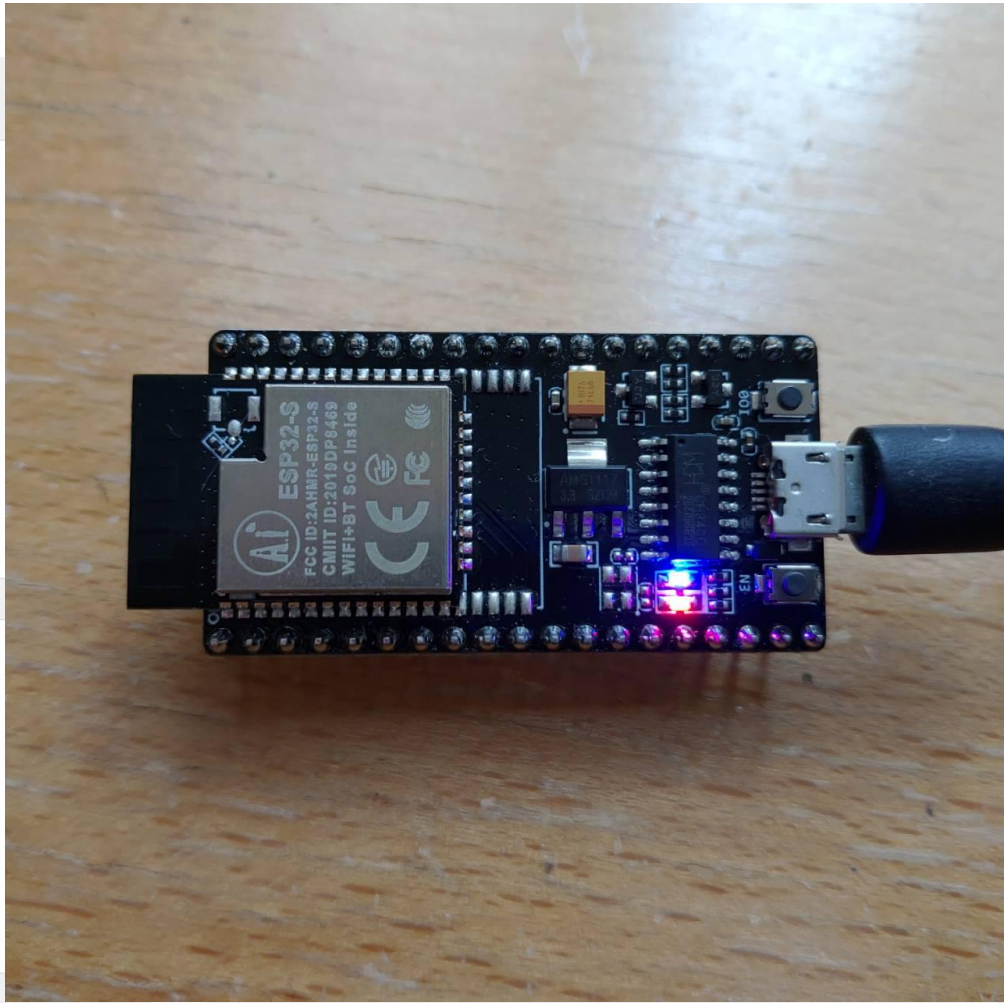
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32

點亮 LED (適用 NodeMCU-32)



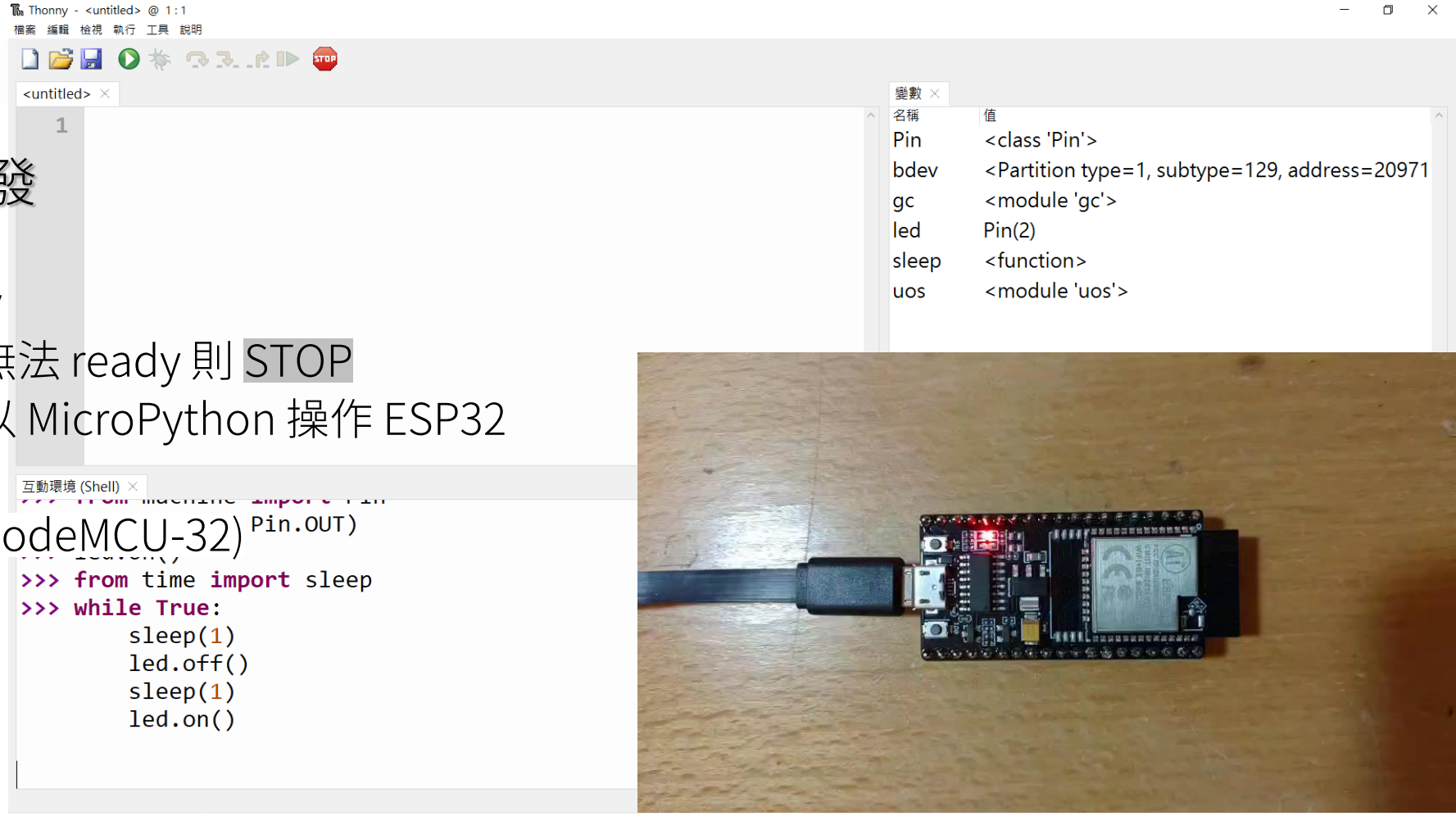
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32

閃爍 LED (適用 NodeMCU-32)



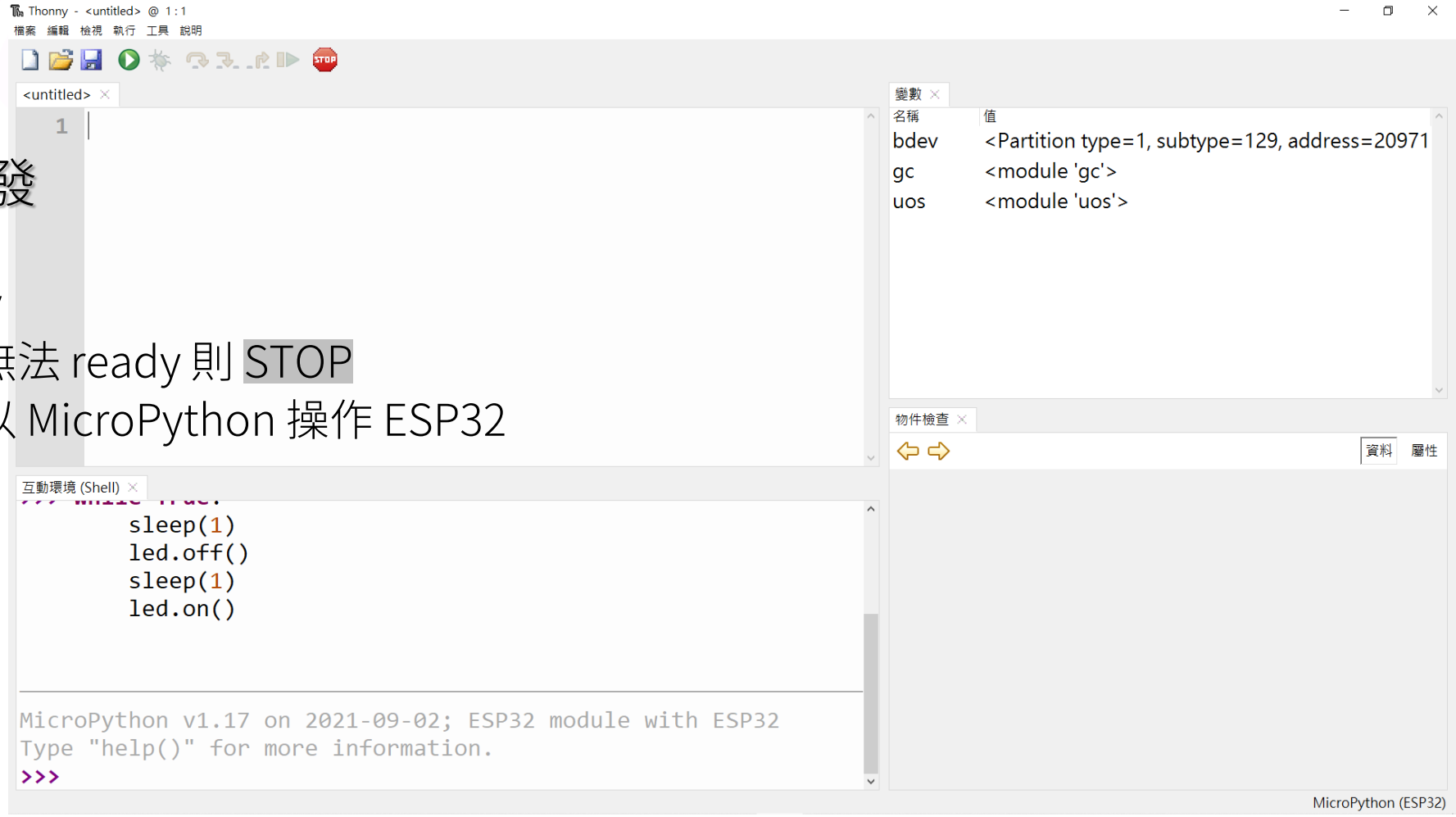
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32

STOP 結束



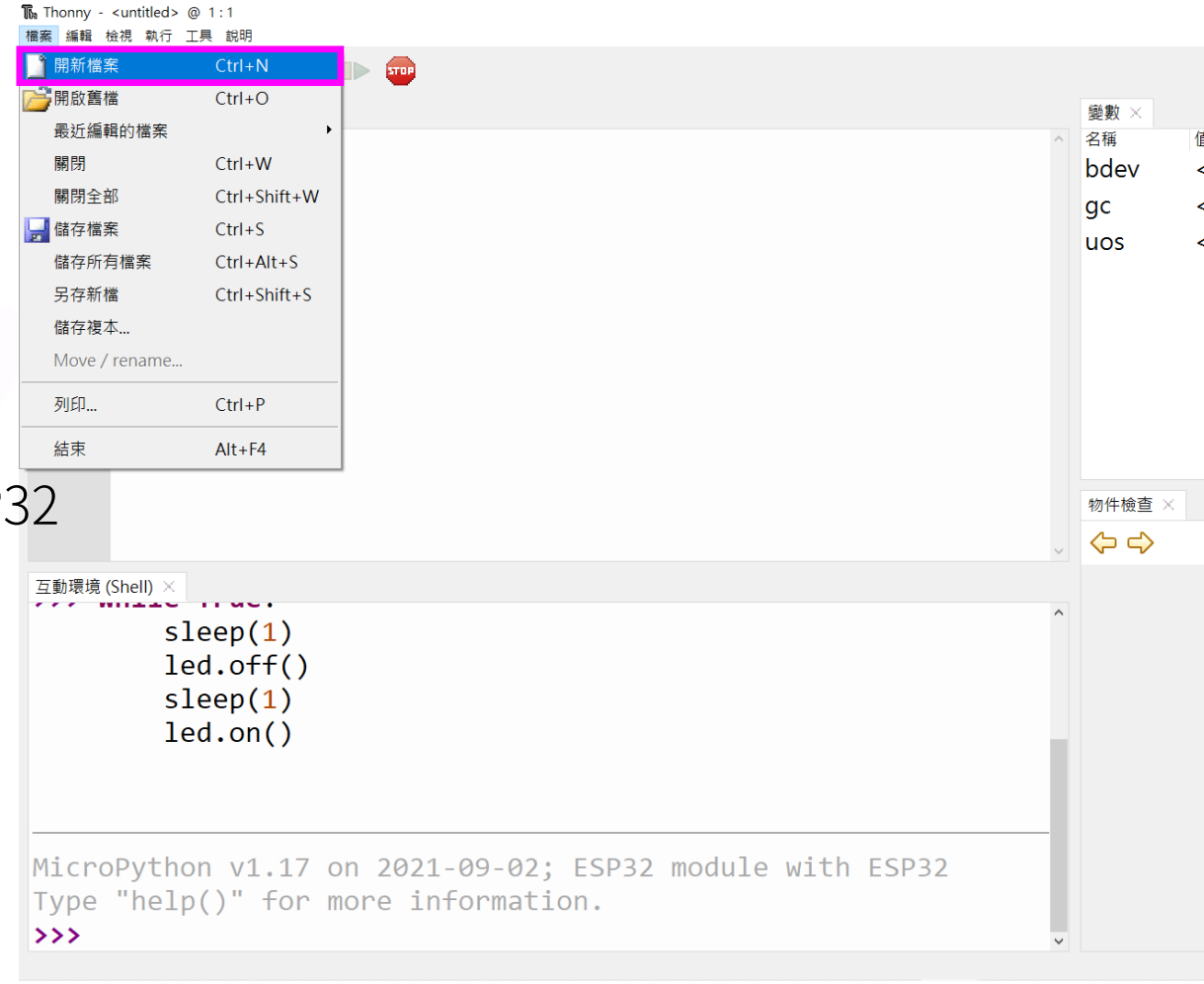
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案

檔案 > 開新檔案



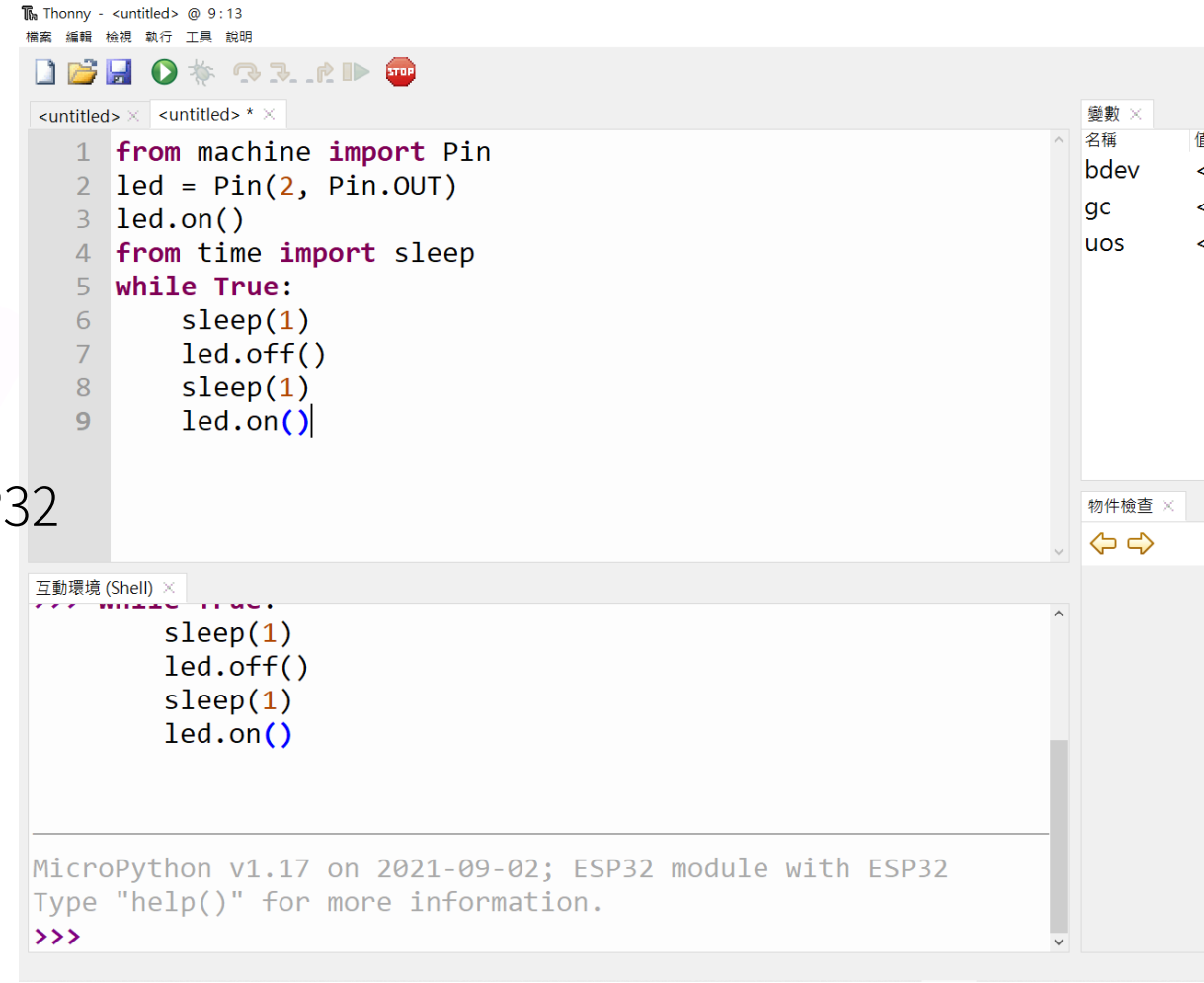
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案

以 MicroPython 撰寫程式



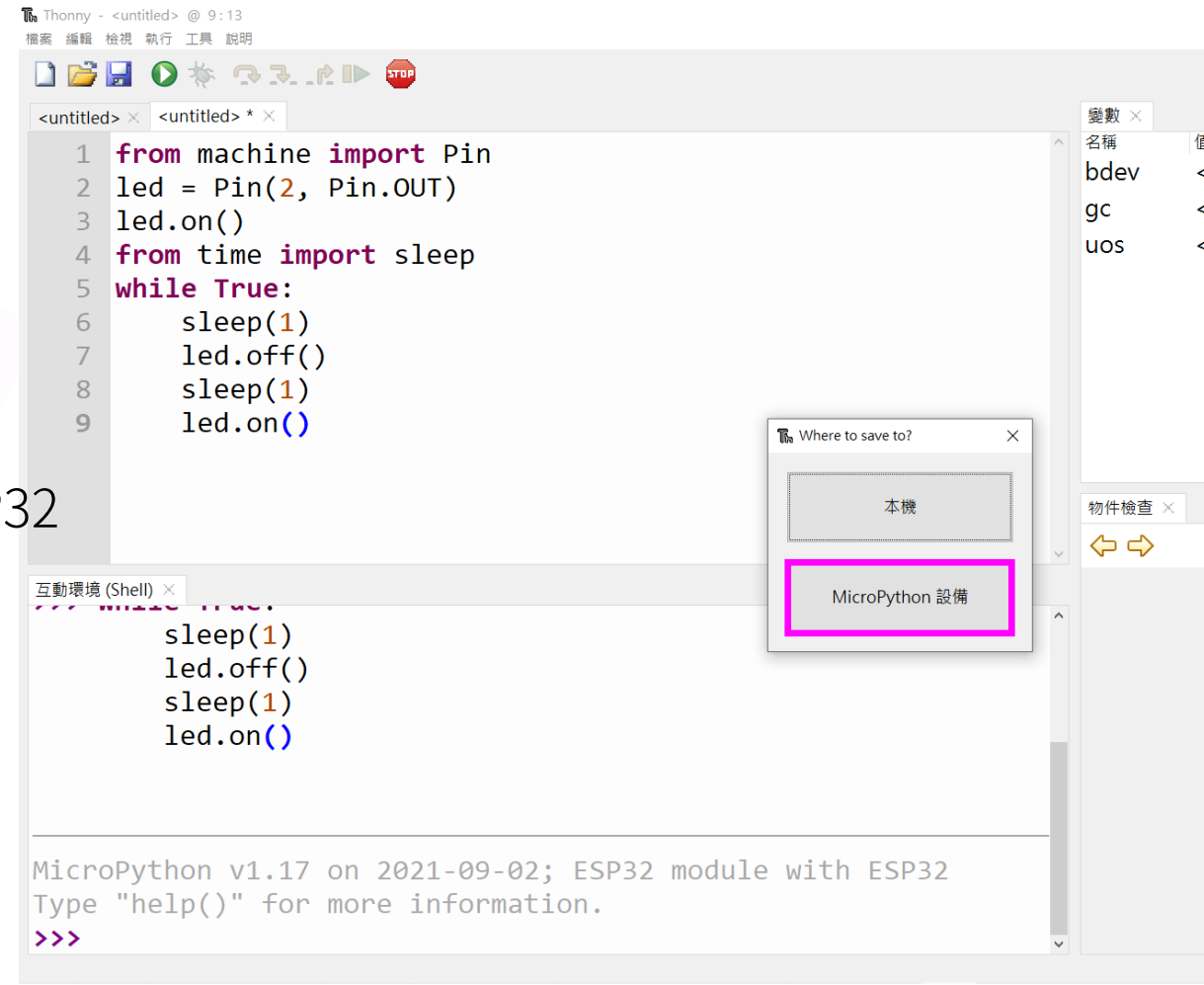
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案

可選擇於 本機 或 MicroPython 設備 儲存
請選擇 MicroPython 設備



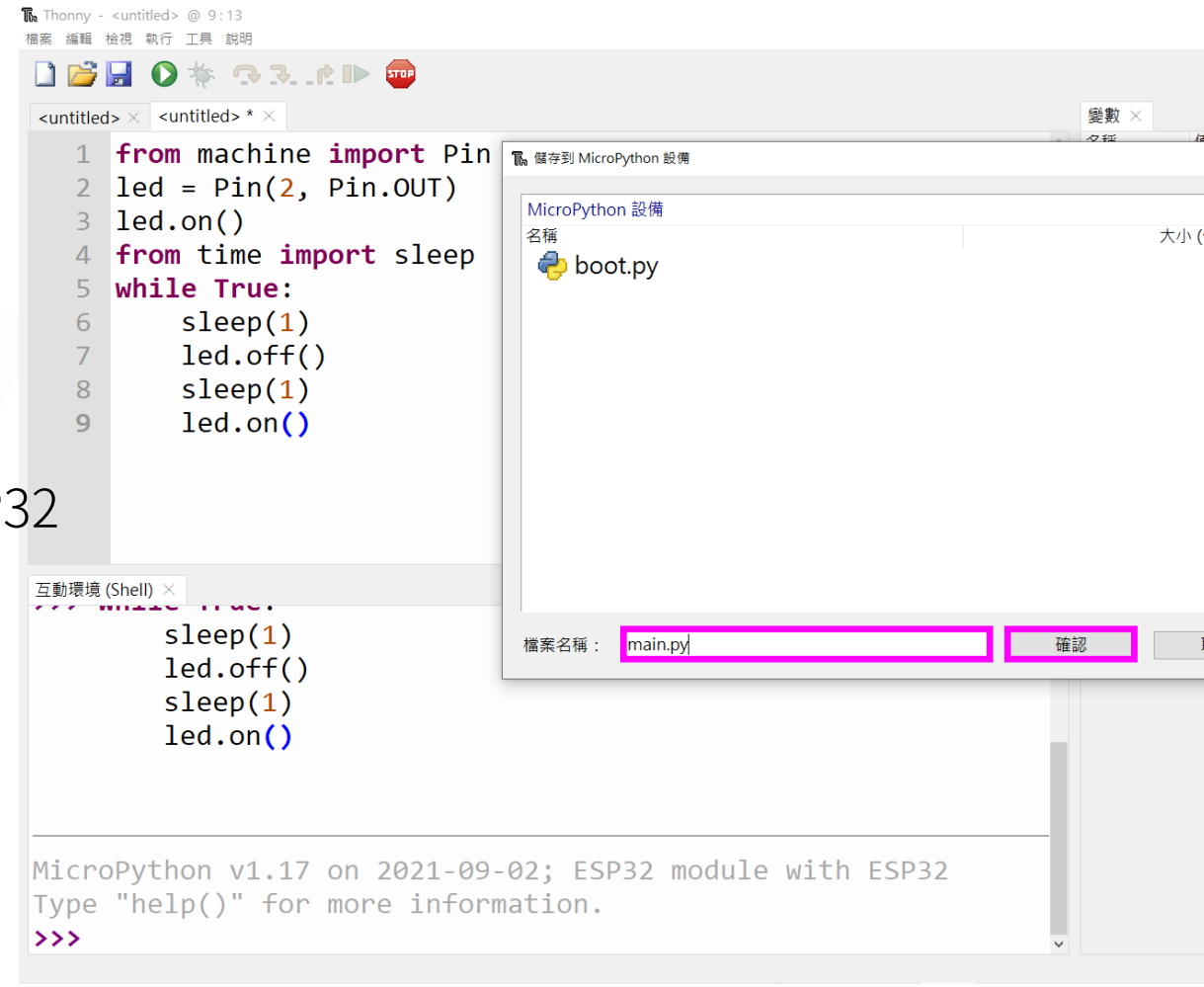
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案

儲存為 main.py > 確認



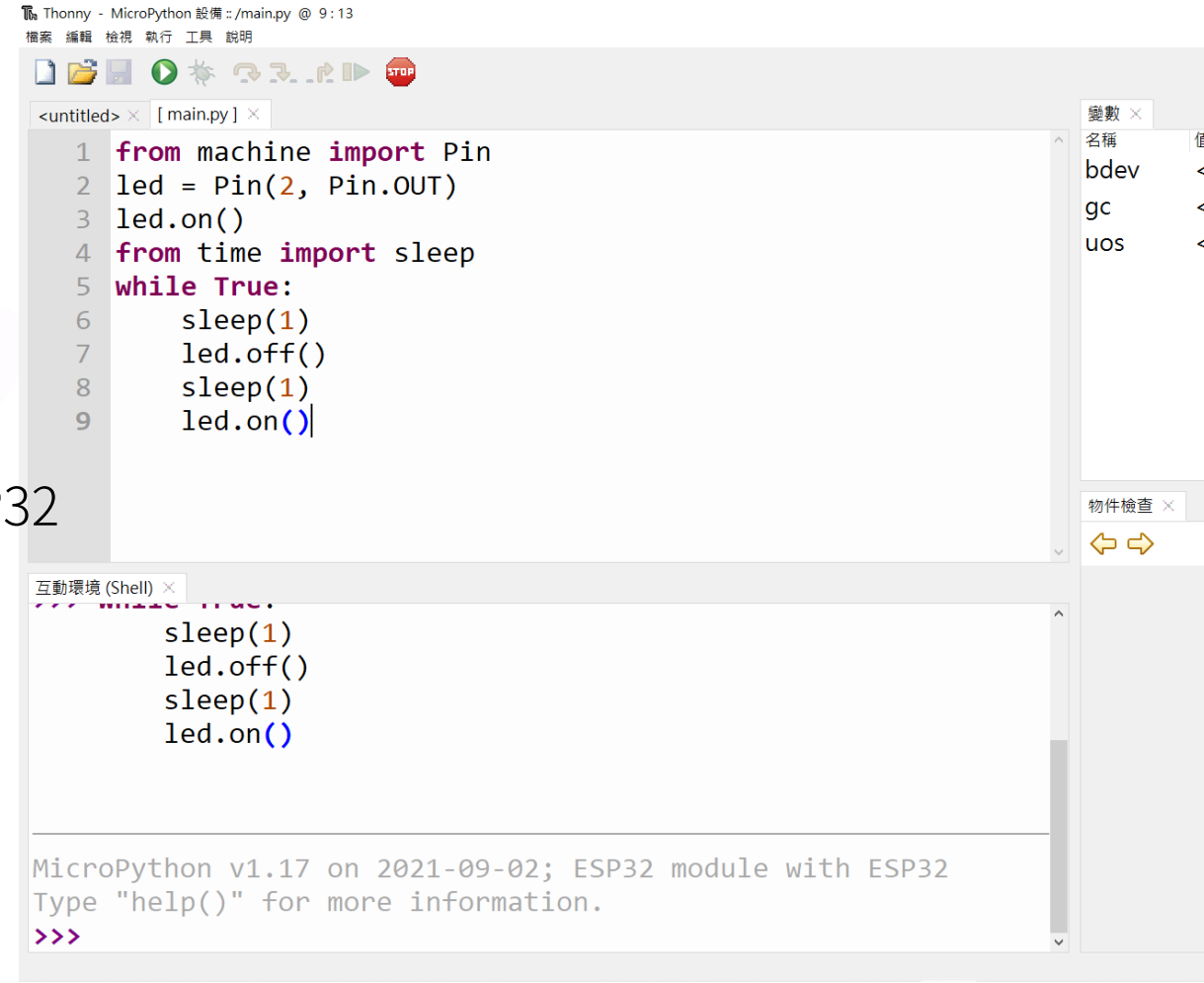
MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案

儲存完畢，ESP32 改接行動電源測試



MicroPython Installation on ESP32

開發工具安裝

c. 以 Thonny 開發

① 啟動 Thonny
② 若互動環境無法 ready 則 STOP
③ 於互動環境以 MicroPython 操作 ESP32
④ 開新檔案
⑤ 開啟舊檔

檔案 > 開啟舊檔
可選擇開啟 本機 或 MicroPython 設備 檔案

ESP32 Sensors feat. MicroPython

DHT-11, ADC

ESP32 Pinout

ESP32 Pinout 設計

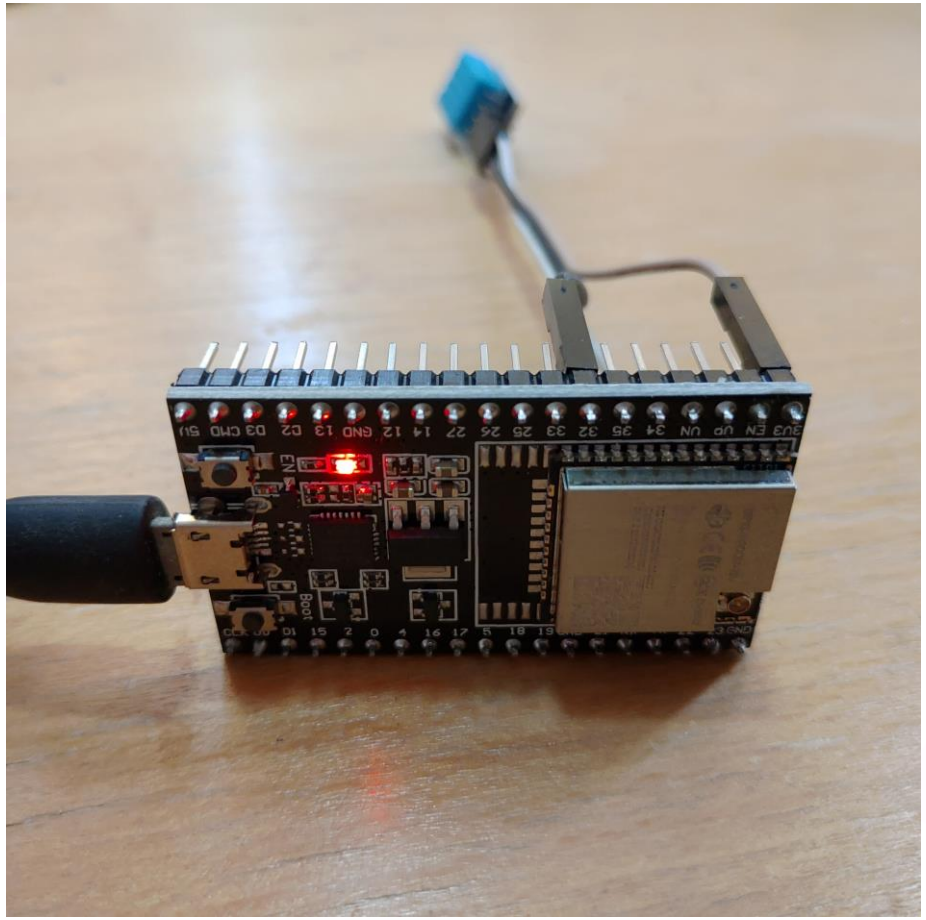
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/esp32-devkitC-v4-pinout.png

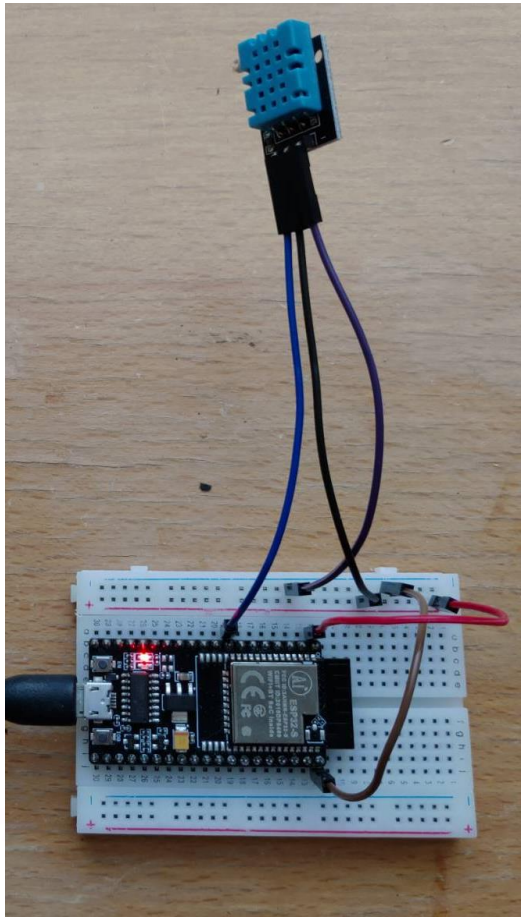
ESP32 DHT-11 feat. MicroPython

溫溼度感測器接線

DHT-11 to ESP32

```
DHT-11          ESP32
-       ----> GND
        ----> 5V or 3V3
S       ----> GPIO
```





ESP32 Sensors feat. MicroPython

boot.py

- 開機後最先執行的 Python 檔
- WiFi 等初始化程序宜置於此

main.py

- boot.py 執行完畢後接續執行的 Python 檔
- 通常以 loop 實作確保持續執行
- 一般應用宜置於此

ESP32 Sensors feat. MicroPython

boot.py

WiFi

```
import network

wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect('YOUR_SSID', 'YOUR_KEY')
while not wifi.isconnected():
    pass
print(wifi.ifconfig())
```

NTP

```
import ntptime

ntptime.settime()  # UTC
```

ESP32 Sensors feat. MicroPython

main.py

DHT-11

```
from machine import Pin
from dht import DHT11

p = DHT11(Pin(YOUR_PIN))
p.measure()
print(f'{p.temperature()}, {p.humidity()}')
```

ADC

```
from machine import Pin, ADC

p = ADC(Pin(YOUR_PIN))
p.atten(ADC.ATTN_11DB)
print(p.read())
```

ESP32 Sensors feat. MicroPython

main.py

MQTT Client

```
from umqtt.simple import MQTTClient

client = MQTTClient(client_id='YOUR_CLIENT', server='YOUR_BROKER')
client.connect()
client.publish('YOUR_TOPIC', 'YOUR_CONTENT')
client.disconnect()
```

ESP32 DHT-11 feat. MicroPython

boot.py

```
import network
import ntptime

wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect('YOUR_SSID', 'YOUR_KEY')

while not wifi.isconnected():
    pass

print(wifi.ifconfig())

ntptime.settime()
```

ESP32 DHT-11 feat. MicroPython

main.py

```
from time import sleep, localtime, time

from machine import Pin
from dht import DHT11
from umqtt.simple import MQTTClient

import esp
esp.osdebug(None)

mqtt_broker = 'YOUR_BROKER'
mqtt_client = MQTTClient(client_id='YOUR_CLIENT', server=mqtt_broker)

p = DHT11(Pin(YOUR_PIN))
```

ESP32 DHT-11 feat. MicroPython

main.py

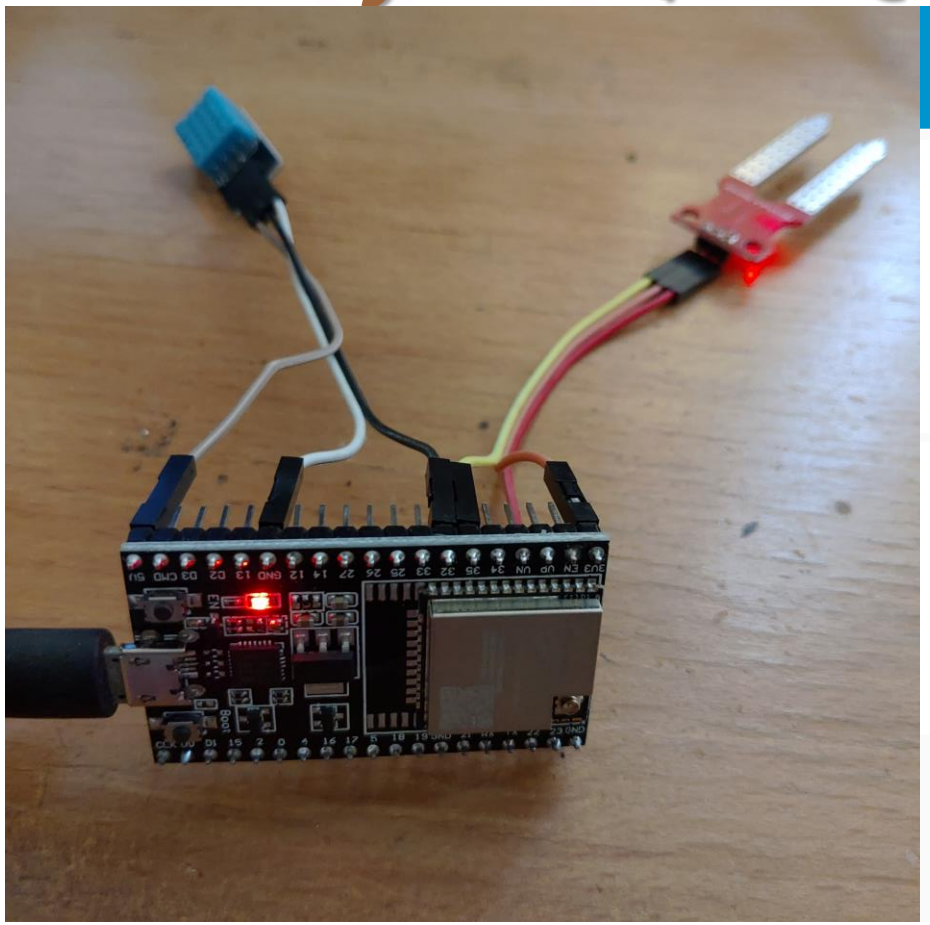
```
    while True:
        mqtt_client.connect()
        p.measure()
        temp = p.temperature()
        hum = p.humidity()
        dt = localtime(time() + 28800)
        dt = f'{dt[0]:04}{dt[1]:02}{dt[2]:02}-{dt[3]:02}{dt[4]:02}{dt[5]:02}'
        topic = 'plant/soya/dht11/1'
        content = f'{topic}|{dt}|{temp}|{hum}'
        print(content)
        mqtt_client.publish(topic, content)
        mqtt_client.disconnect()
        sleep(YOUR_SECS)
```

ESP32 Sensors feat. MicroPython/ MQTT

建立並啟動 MQTT Broker

- 請參閱

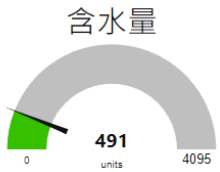
ESP32 DHT-11 & Moisture feat. MicroPython/ MQTT



Moisture

RAW

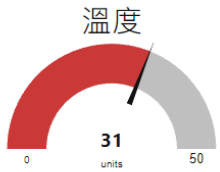
plant/soya/moisture/1|20211007-141528|491

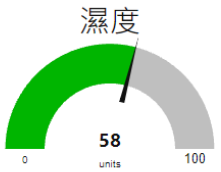


DHT-11

RAW

plant/soya/dht11/1|20211007-141528|31|58



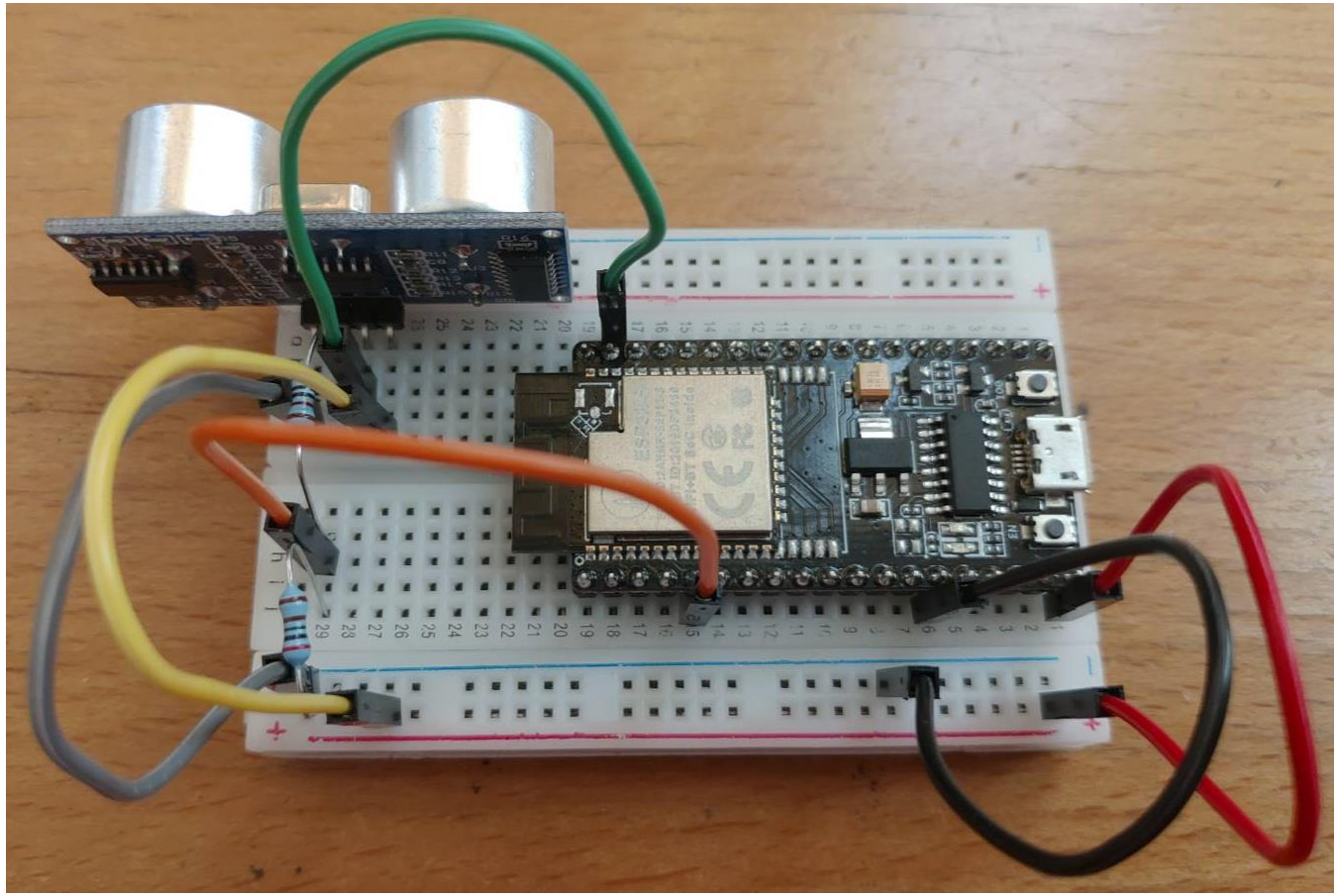


ESP32 HC-SR04 feat. MicroPython

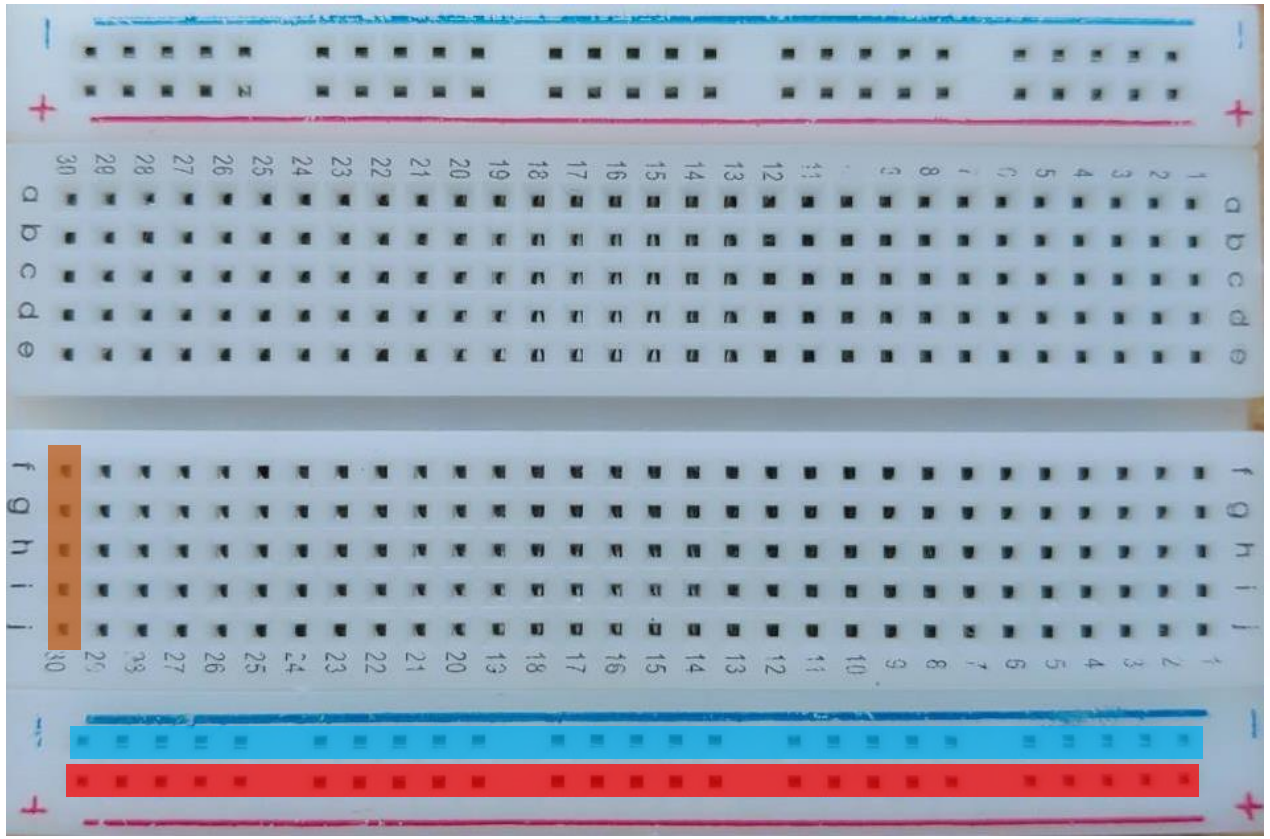
聲納界接

HC-SR04 to ESP32

```
HC-SR04      Breadboard      ESP32
Vcc      -------->------>    5V
Trig     -------->------>    GPIO
Echo     -> 1KΩ --+->--->    GPIO
                  |
                  -> 2KΩ ->  GND
GND      -------->------>    GND
```



Breadboard Conecpt



ESP32 HC-SR04 feat. MicroPython

開發測試

a. 函式庫 hcsr04.py

① 於 GitHub https://github.com/rsc1975/micropython-hcsr04 取得 hcsr04.py

② 將 hcsr04.py 儲存至 ESP32

ESP32 HC-SR04 feat. MicroPython

開發測試

b. 主程式 main.py

```
from hcsr04 import HCSR04
from time import sleep

sensor = HCSR04(trigger_pin=23, echo_pin=34, echo_timeout_us=10000)
try:
    while True:
        print(f'Distance: {sensor.distance_cm()}cm')
        sleep(1)
except OSError as e:
    print('ERROR getting distance:', e)
```

ESP32 Notes

1. **對時**
   - ESP32 不儲存時間，每次開機需要對時，程式應放在 boot.py
   - 對時無法設定時區，計算台灣時間需自行加 8 HR
2. **PINs for ADC**
   - PIN (GPIO) OK: 32, 33, 34, 35
   - PIN (GPIO) Failure: 25, 26, 27
3. **多 Sensor 供電**
   - 若外接超過兩個 sensor 因供電 PIN 不足，需靠麵包版延展電源

Your Turn - Watching the Temperature

以 ESP32 實作溫度監控機制，當溫度高於 25°C 以 MQTT 告警

Your Turn - Watching the Distance

以 ESP32 實作距離監控機制，當距離小於 10cm 以 MQTT 告警

ESP32CAM Stream

ESP32-CAM, ESP32-CAM-MB

ESP32CAM Stream

1. 安裝 Arduino IDE (on Windows 10)

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

3. 擷取畫面 (on Windows 10)

ESP32CAM Stream

1. 安裝 Arduino IDE (on Windows 10)

安裝 Arduino IDE

ESP32CAM Stream

1. 安裝 Arduino IDE (on Windows 10)

安裝 Arduino IDE

① https://www.arduino.cc/ > SOFTWARE > DOWNLOAD OPTIONS: Windows Win 7 and newer > JUST DOWNLOAD
② 獲得 arduino-1.8.19-windows.exe
③ 點擊 arduino-1.8.19-windows.exe 進行安裝 > I Agree > Next > > Install > Close
④ 啟動 Arduino IDE

ESP32CAM Stream

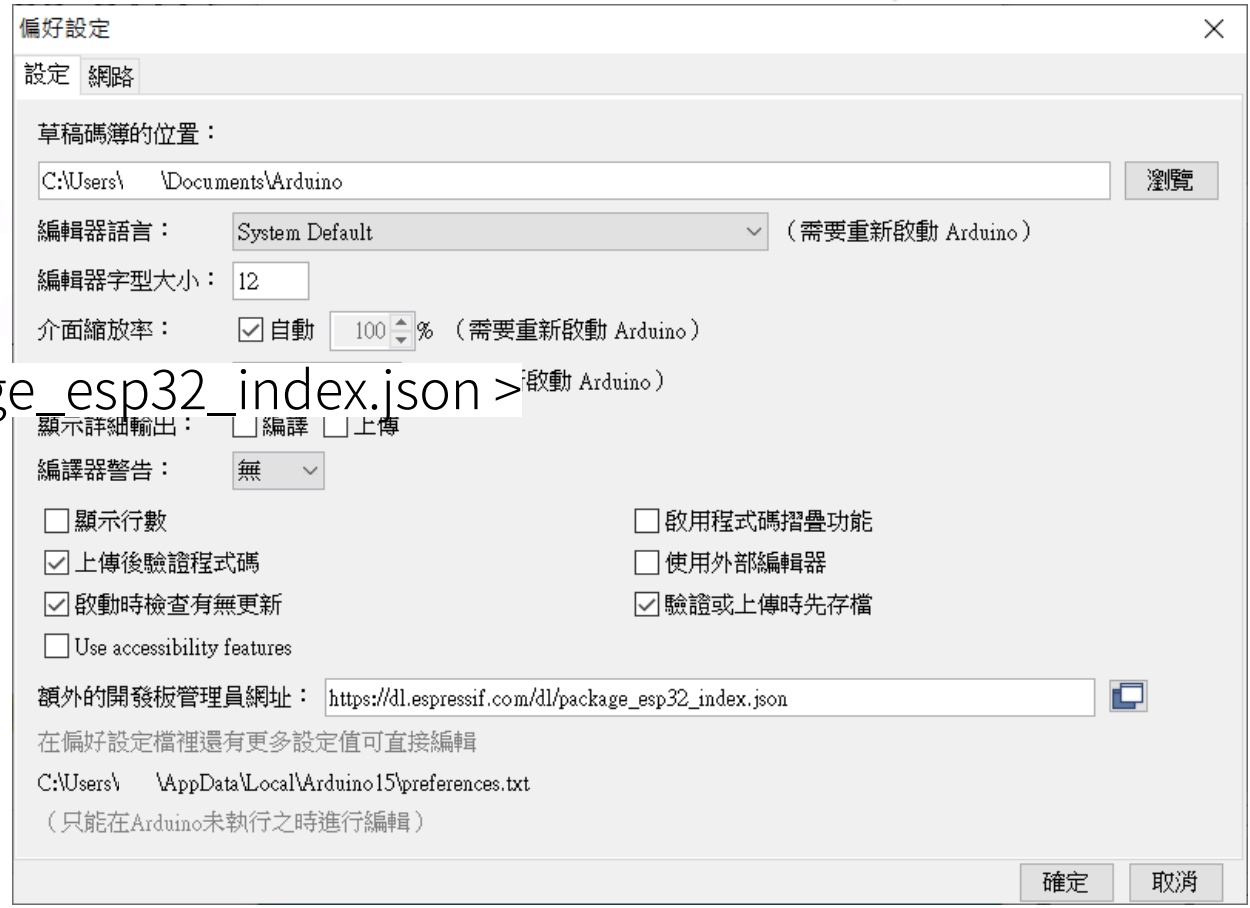
2. **製作 CameraWebServer** (on ESP32-CAM feat. Windows 10)
    a. 設定 Arduino IDE for ESP32-CAM
    b. 連接 ESP32-CAM
    c. 安裝 CameraWebServer

ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

a. 設定 Arduino IDE for ESP32-CAM

① Arduino IDE > 檔案 > 偏好設定 > 額外的開發板管理員網址：https://dl.espressif.com/dl/package_esp32_index.json > 確定



ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

a. 設定 Arduino IDE for ESP32-CAM

① Arduino IDE > 檔案 > 偏好設定 > 額外的開發板管理員網址：https://dl.espressif.com/dl/package_esp32_index.json > 確定

② Arduino IDE > 工具 > 開發版: "Arduino Uno" > 開發版管理員... > 尋找 "esp32" 並點選 > 安裝 > 關閉

ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

a. 設定 Arduino IDE for ESP32-CAM

① Arduino IDE > 檔案 > 偏好設定 >
額外的開發板管理員網址：
https://dl.espressif.com/dl/package_esp32_index.json >
確定

② Arduino IDE > 工具 > 開發版: "Arduino Uno" >
開發版管理員... > 尋找 "esp32" 並點選 >
安裝 > 關閉

③ Arduino IDE > 工具 > 開發版: "Arduino Uno" >
ESP32 Arduino >
尋找 "AI Thinker ESP32-CAM" 並點選

ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

b. 連接 ESP32-CAM

① ESP32-CAM 連接 Windows 10

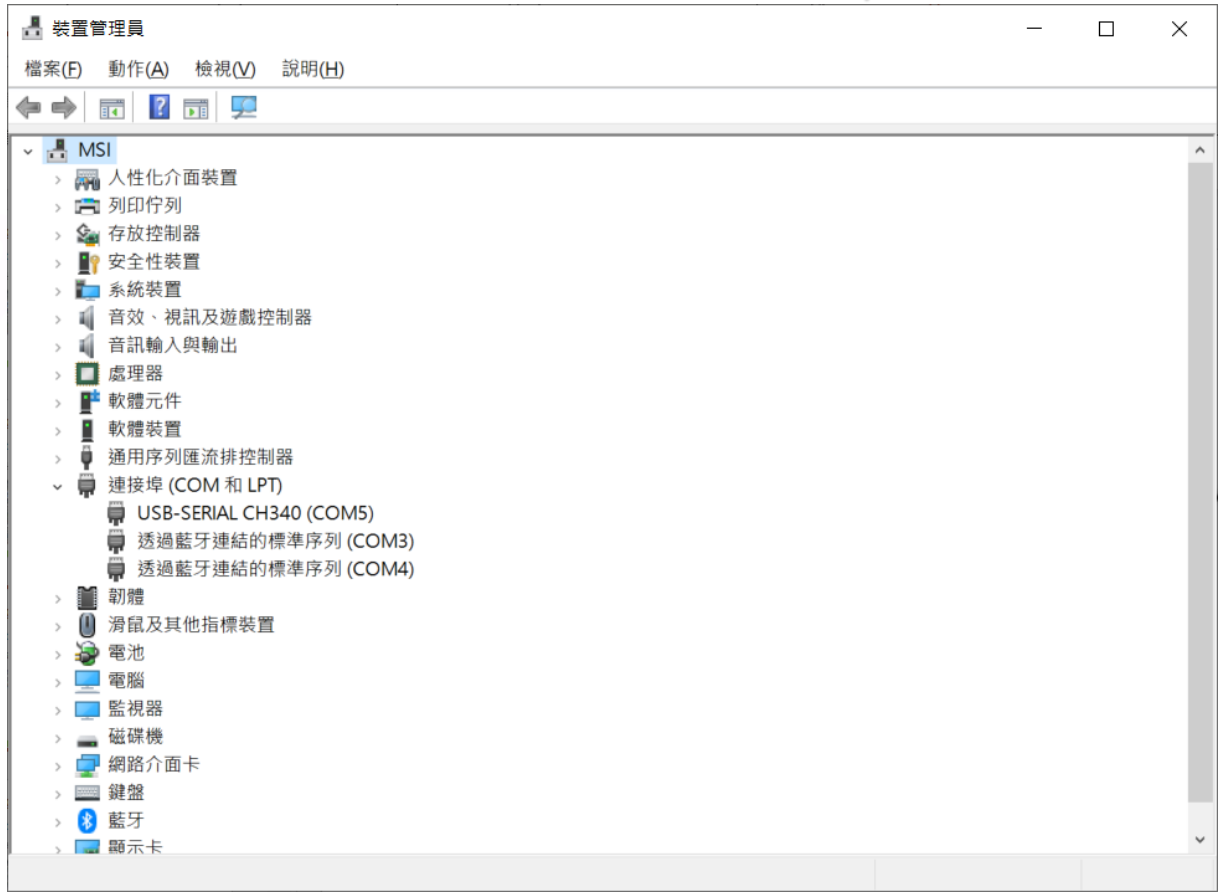
ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

b. 連接 ESP32-CAM

① ESP32-CAM 連接 Windows 10

② 確認 USB UART Driver 正確安裝
安裝方式參閱



ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

b. 連接 ESP32-CAM

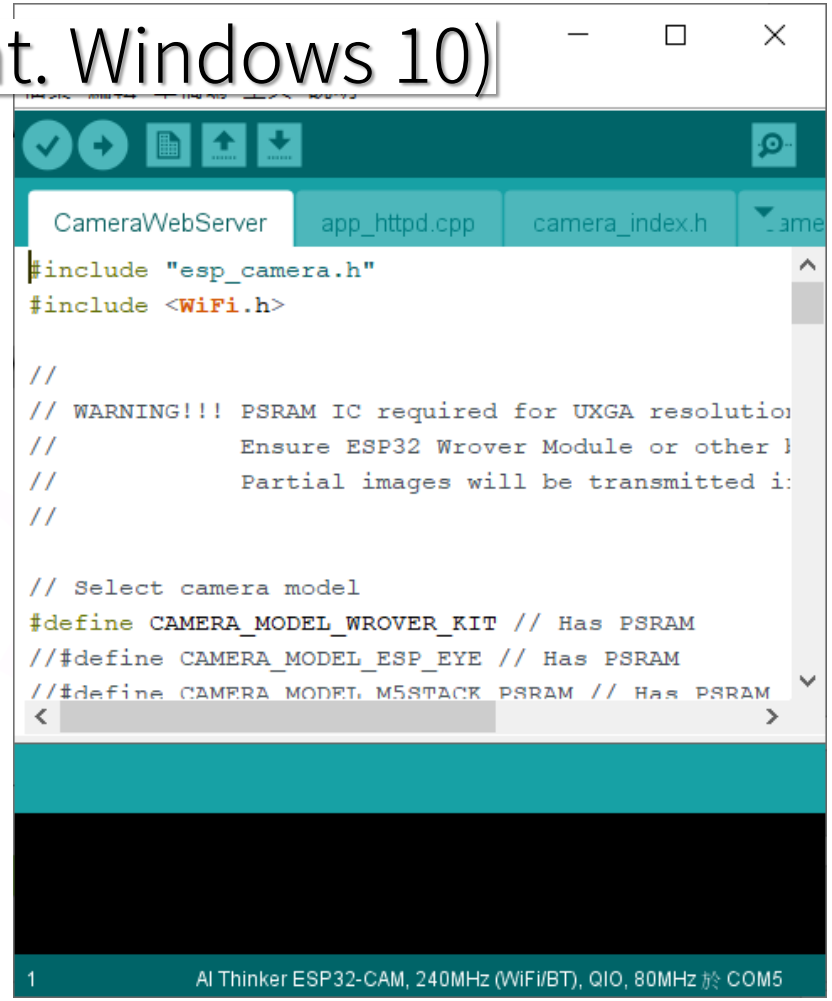
① ESP32-CAM 連接 Windows 10
② 確認 USB UART Driver 正確安裝
③ 設定 COM 埠
Arduino IDE > 工具 > 序列埠 > *YOUR_COM*

ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer



ESP32CAM Stream

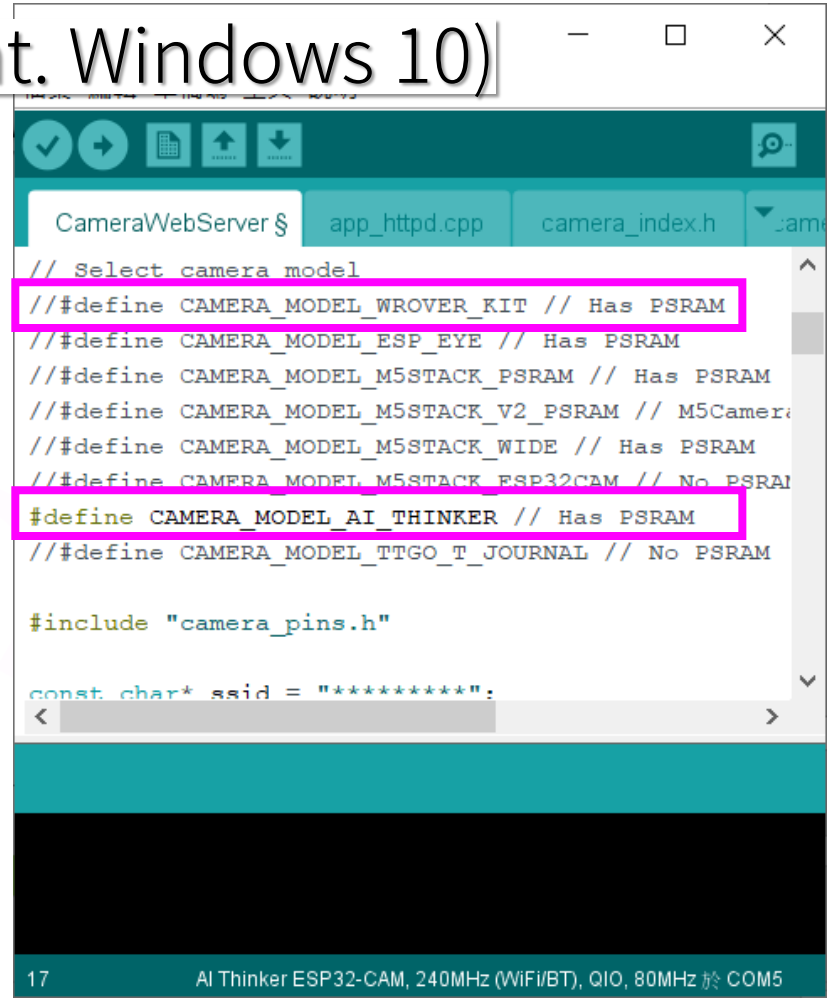
2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer

② 修改 AI_THINKER

```
//#define CAMERA_MODEL_WROVER_KIT
#define CAMERA_MODEL_THINKER
```



ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

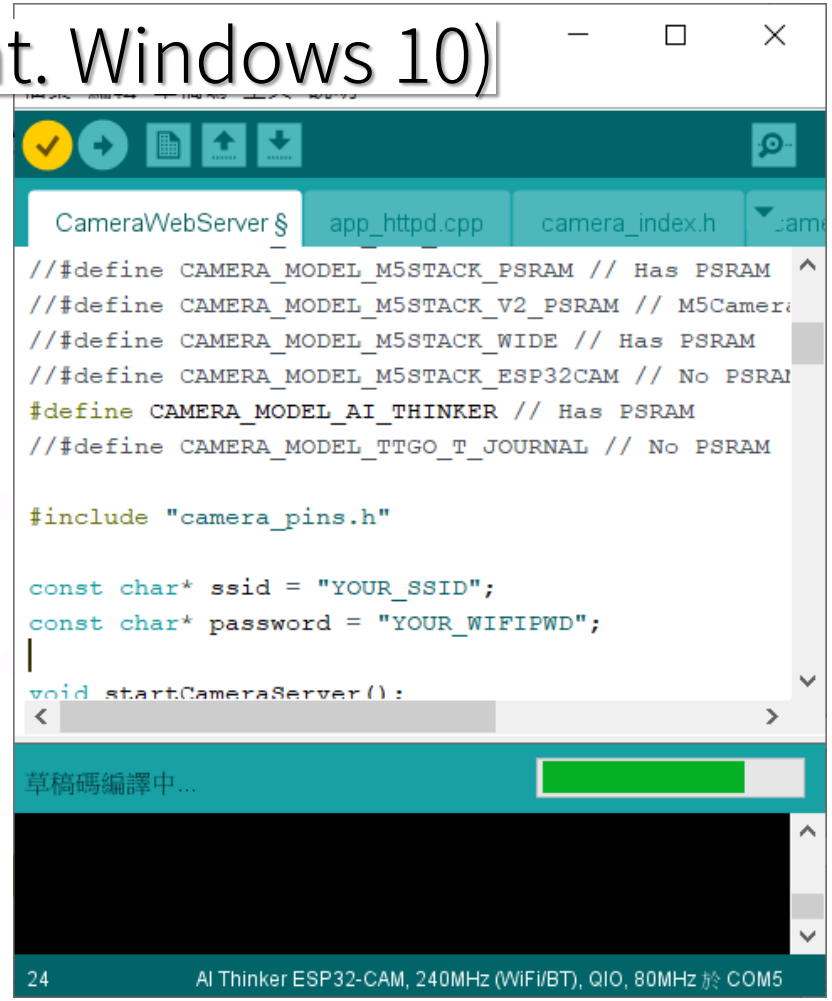
① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer
② 修改 AI_THINKER
③ 修改 WiFi SSID & PWD

ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer
② 修改 AI_THINKER
③ 修改 WiFi SSID & PWD
④ 驗證

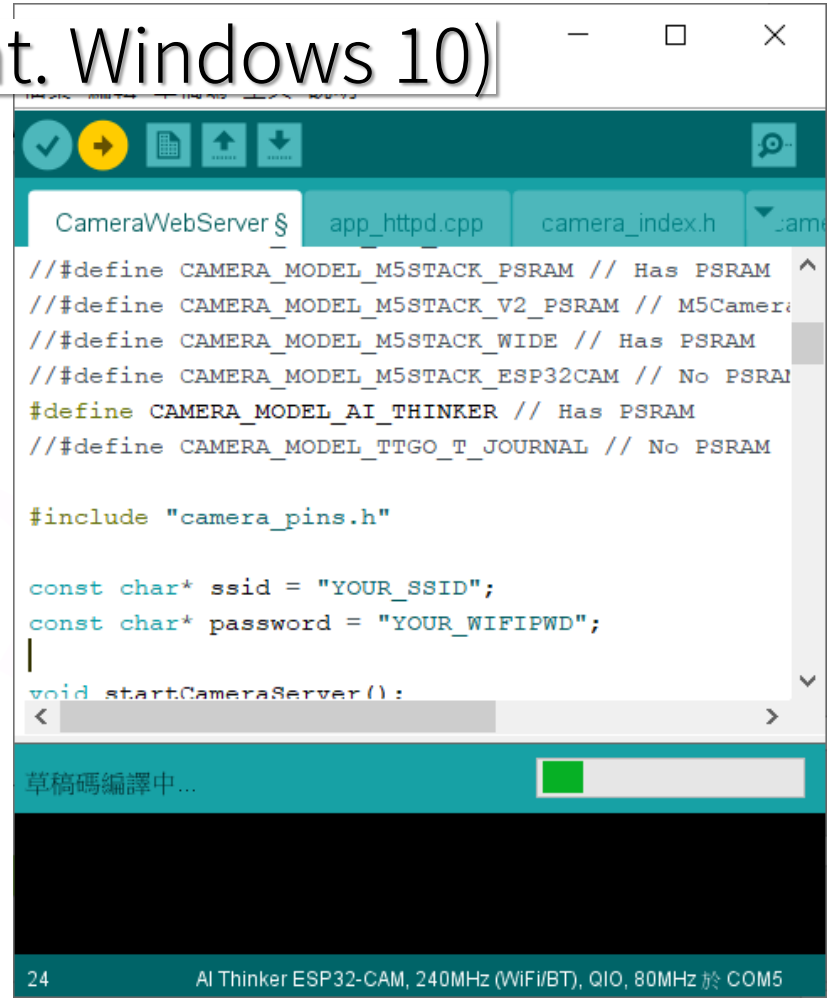


ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer
② 修改 AI_THINKER
③ 修改 WiFi SSID & PWD
④ 驗證
⑤ 上傳

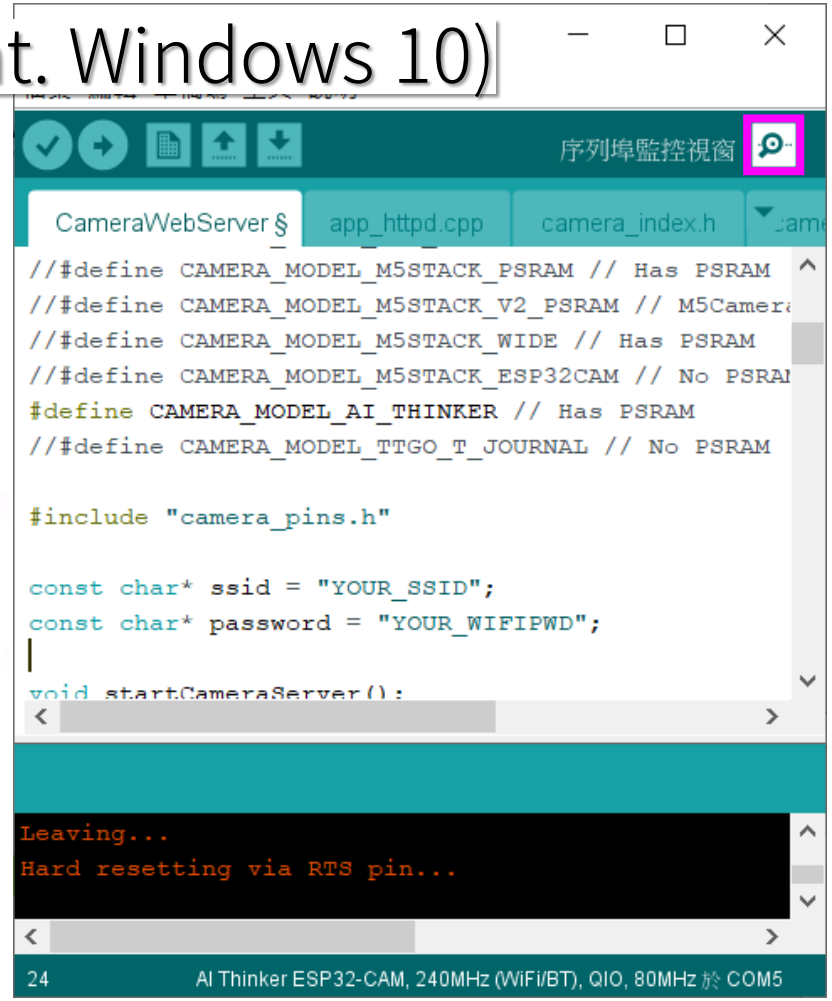


ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer
② 修改 AI_THINKER
③ 修改 WiFi SSID & PWD
④ 驗證
⑤ 上傳
⑥ 確認
序列埠監控視窗 > 硬體 reset

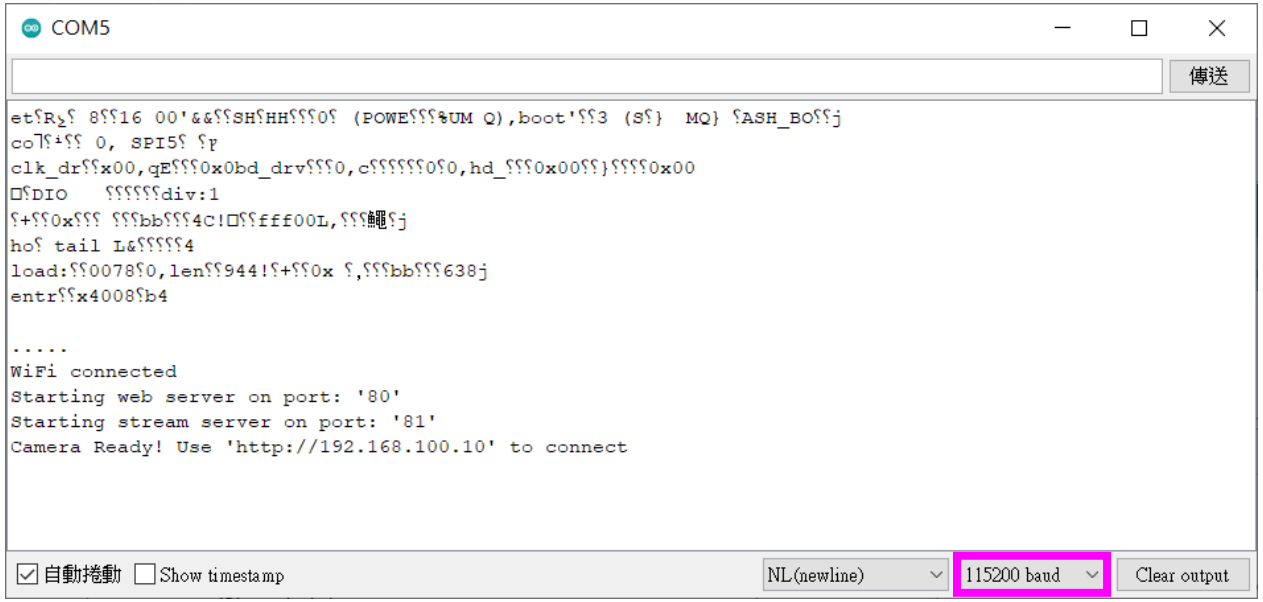


ESP32CAM Stream

2. 製作 CameraWebServer (on ESP32-CAM feat. Windows 10)

c. 安裝 CameraWebServer

① Arduino IDE > 檔案 > 範例 > ESP32 > Camera > CameraWebServer
② 修改 AI_THINKER
③ 修改 WiFi SSID & PWD
④ 驗證
⑤ 上傳
⑥ 確認
序列埠監控視窗 > 硬體 reset



ESP32CAM Stream

3. **擷取畫面** (on Windows 10)

    a. 標準測試

    b. 擷取測試

ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

a. 標準測試

① ESP32-CAM 接獨立電源

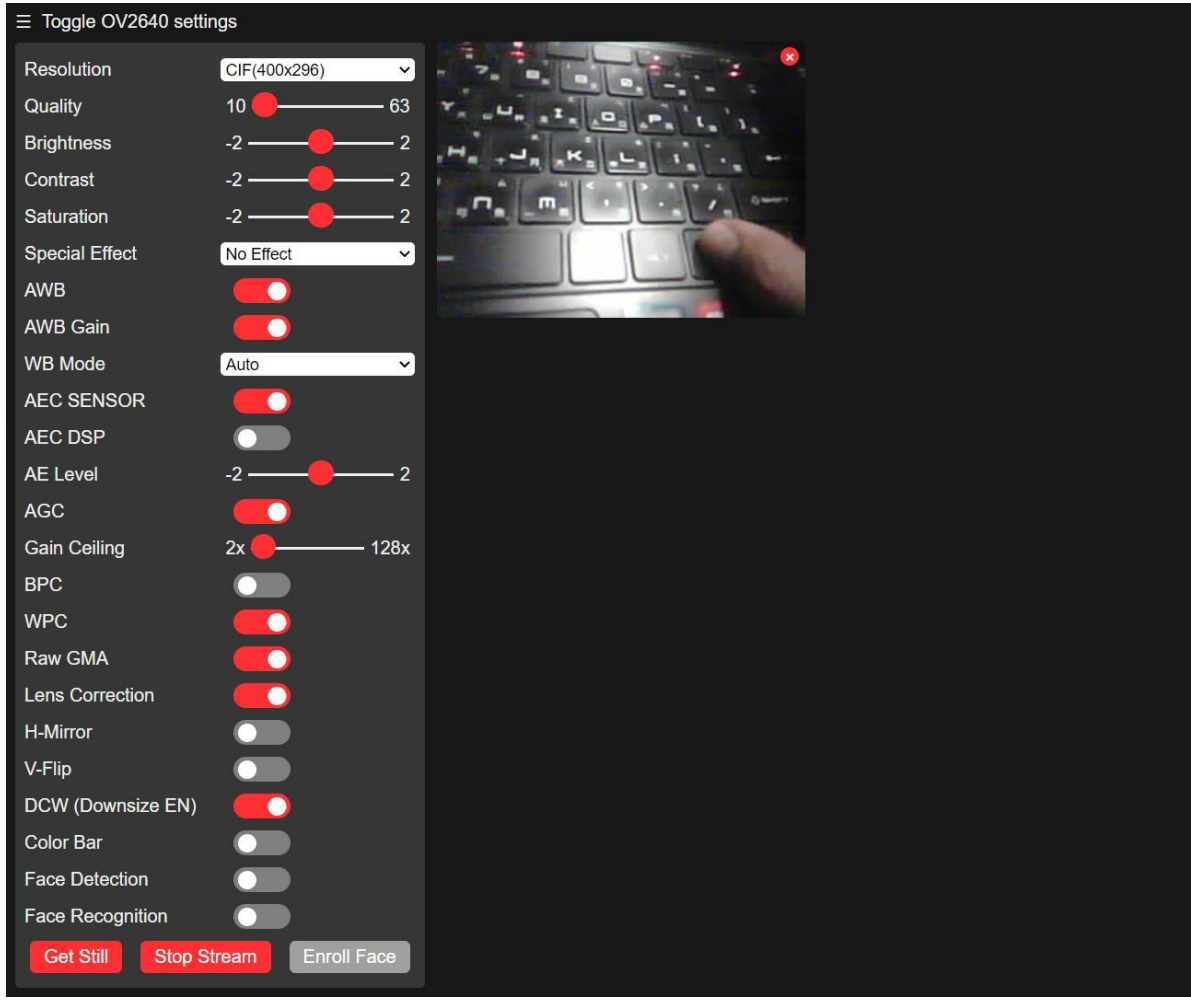
ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

a. 標準測試

① ESP32-CAM 接獨立電源

② 以瀏覽器瀏覽 http://x.x.x.x:80
(x.x.x.x 為 ESP32-CAM IP)

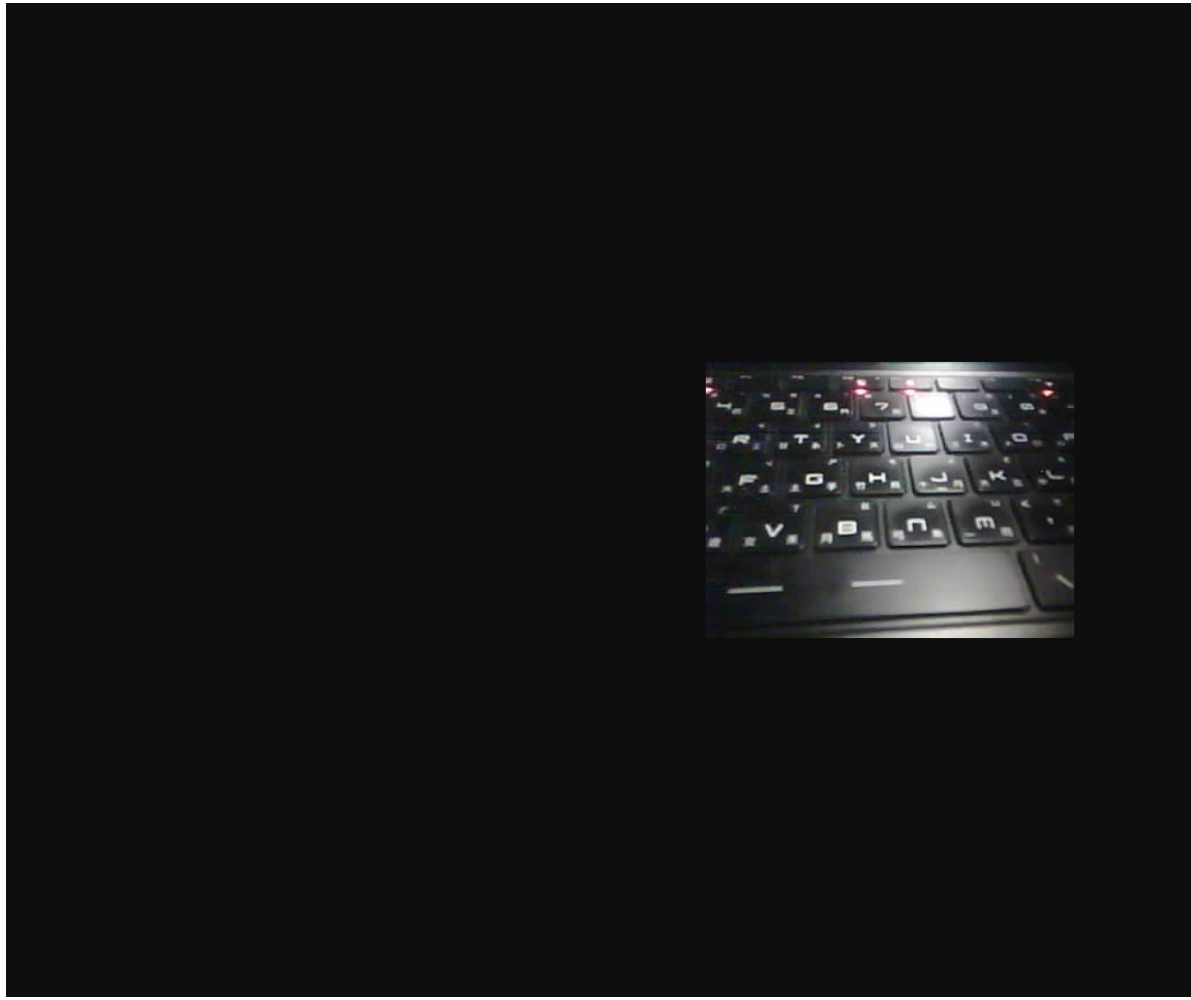


ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

a. 標準測試

① ESP32-CAM 接獨立電源
② 以瀏覽器瀏覽 http://x.x.x.x:80
(x.x.x.x 為 ESP32-CAM IP)
③ 以瀏覽器瀏覽 http://x.x.x.x:81/stream



ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

a. 標準測試

① ESP32-CAM 接獨立電源
② 以瀏覽器瀏覽 http://x.x.x.x:80
(x.x.x.x 為 ESP32-CAM IP)
③ 以瀏覽器瀏覽 http://x.x.x.x:81/stream
④ Note
ESP32-CAM WebCameraServer 僅限單 session 連入，連入前須先關閉其他來源連線

ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

b. 擷取測試

① ESP32-CAM 接獨立電源
② 準備測試環境

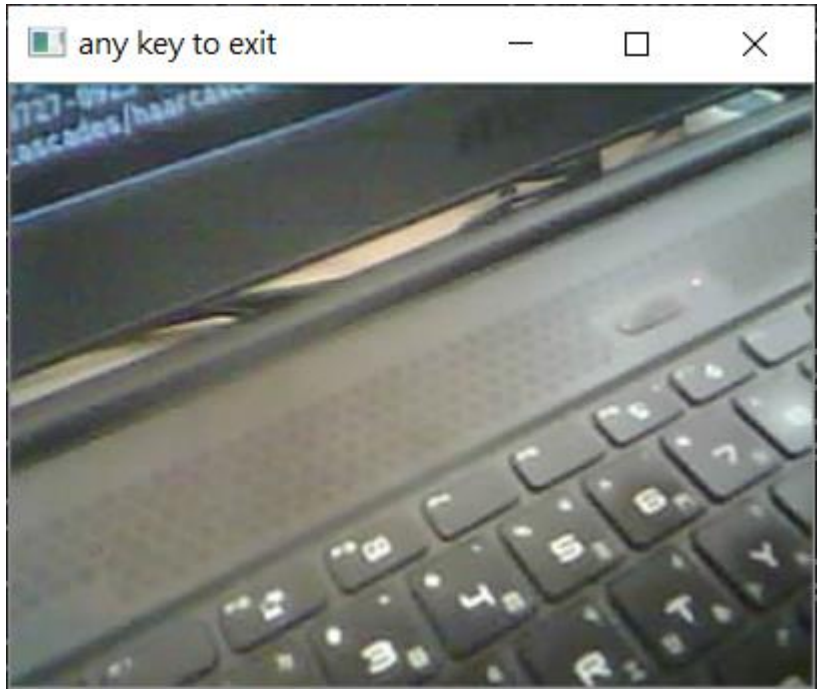
```
Python 3.6.8+
numpy
opencv-python
```

ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

b. 擷取測試

① ESP32-CAM 接獨立電源
② 準備測試環境
③ 測試




```
python playesp32cam.py -i 192.168.100.12

python playesp32cam.py -i 192.168.100.12 -g

python playesp32cam.py -i 192.168.100.12 -c haarcascade_frontalface_default.xml
```

ESP32CAM Stream

3. 擷取畫面 (on Windows 10)

b. 擷取測試

① ESP32-CAM 接獨立電源
② 準備測試環境
③ 測試

```
stream = urlopen(url)
bts = stream.read(4096)
jpghead = bts.find(b'\xff\xd8')
jpgend = bts.find(b'\xff\xd9')
jpg = bts[jpghead:jpgend+2]
img = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8), cv2.IMREAD_UNCHANGED)
```

More

LED

1. LED on Raspberry/ Jetson

a. LED to Raspberry Pi

```
LED          Breadboard     Raspi
LONG     ----> 100Ω ----> GPIO
SHORT    --------->------> GND
```

b. Command (Raspberry only)

```
raspi-gpio set YOUR_GPIO op dh
raspi-gpio set YOUR_GPIO op dl
raspi-gpio help
```

c. Code in Python

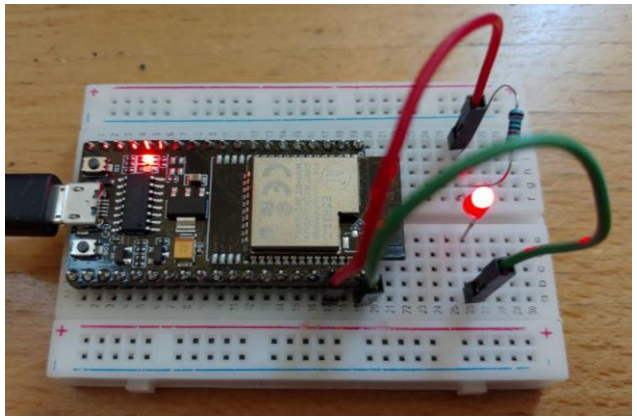
```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
YOUR_PIN = 17
GPIO.setup(YOUR_PIN, GPIO.OUT)
GPIO.output(YOUR_PIN, GPIO.HIGH)
sleep(3)
GPIO.output(YOUR_PIN, GPIO.LOW)
GPIO.cleanup()
```

LED

2. LED on ESP32

a. LED to ESP32

```
LED          Breadboard     ESP32
LONG     ----> 100Ω ----> GPIO
SHORT    --------->------> GND
```



b. Code in MicroPython

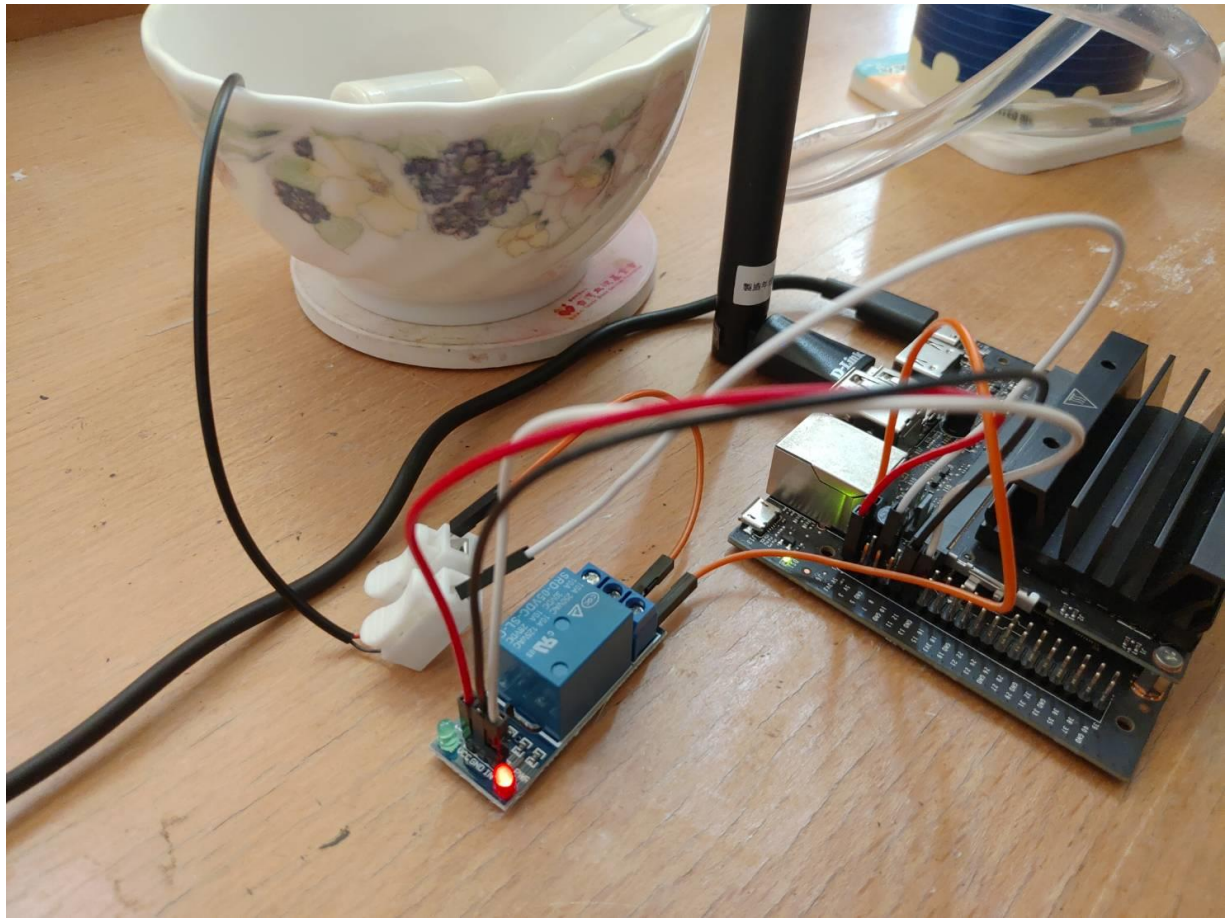
```
from machine import Pin
from time import sleep

YOUR_PIN = 17
led = Pin(YOUR_PIN, Pin.OUT)
led.value(1)
sleep(3)
led.value(0)
```

Pump

1. Pump on Raspberry/ Jetson

    a. Pump to Raspberry Pi

```
Pump          Relay          Raspi
-          ---------------->  GND
+          ----> COM
                 NC ------->  5V
                 IN ------->  GPIO
                 VCC ----->   3V3
                 GND ----->   GND
```



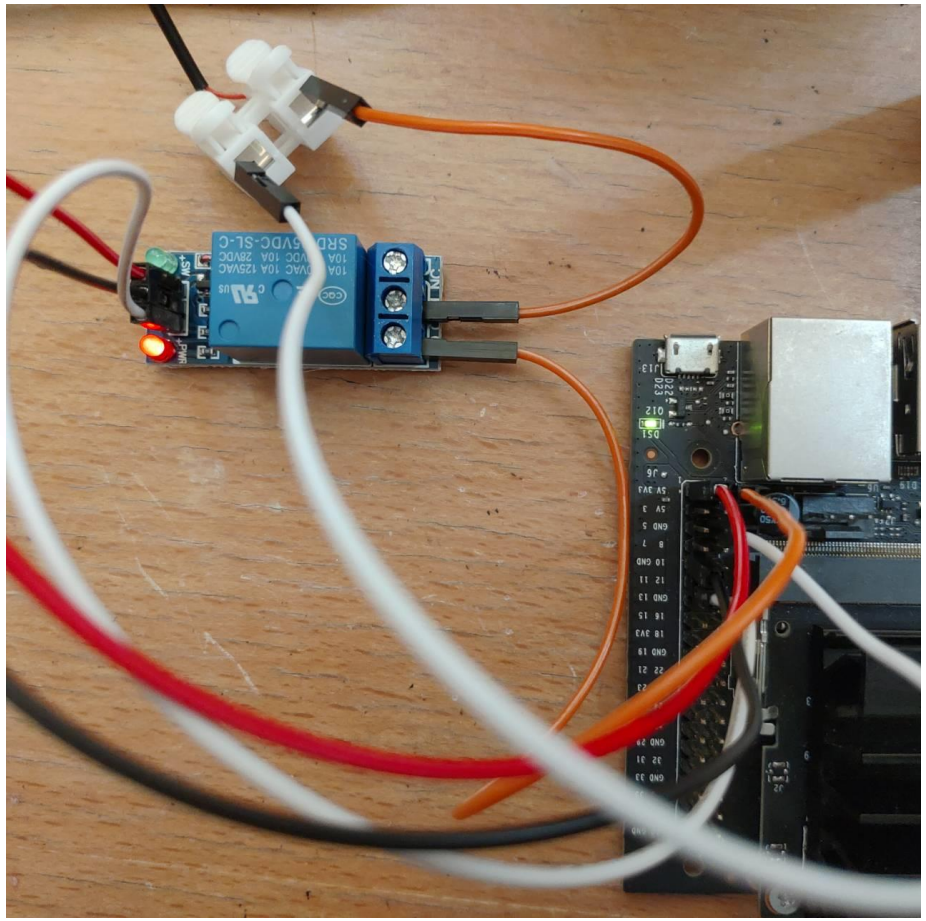
Note
NC: Normally Close，預設連通
NO: Normally Open，預設斷路
COM: Common，公共點

Pump

1. Pump on Raspberry/ Jetson

a. Pump to Raspberry Pi

```
Pump          Relay          Raspi
-          ---------------->  GND
+          ----> COM
                 NC ------->  5V
                 IN ------->  GPIO
                 VCC ------>  3V3
                 GND ------>  GND
```

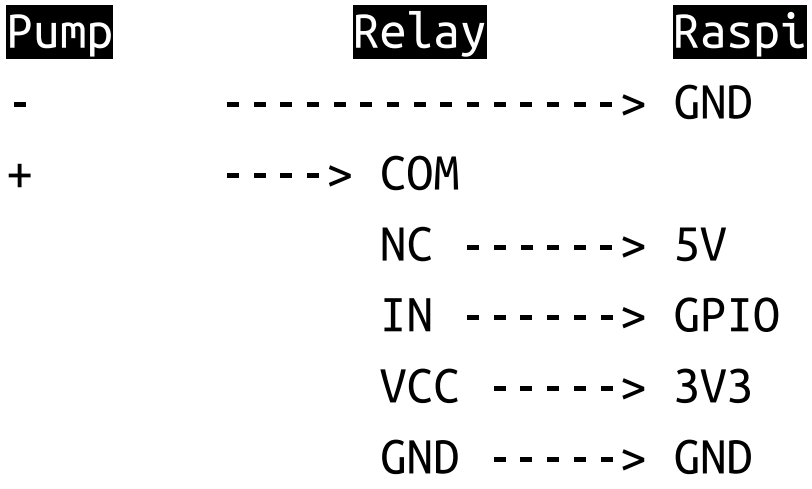


Note
NC: Normally Close，預設連通
NO: Normally Open，預設斷路
COM: Common，公共點

Pump

1. Pump on Raspberry/ Jetson

a. Pump to Raspberry Pi



Note
NC: Normally Close，預設連通
NO: Normally Open，預設斷路
COM: Common，公共點

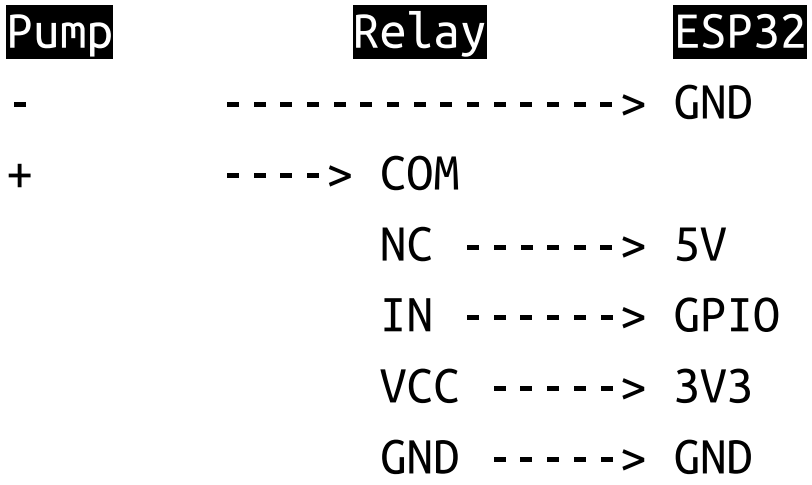
b. Code in Python

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
YOUR_PIN = 17
GPIO.setup(YOUR_PIN, GPIO.OUT)
GPIO.output(YOUR_PIN, GPIO.LOW)
sleep(3)
GPIO.output(YOUR_PIN, GPIO.HIGH)
GPIO.cleanup()
```

Pump

2. Pump on ESP32

a. Pump to ESP32



b. Code in MicroPython

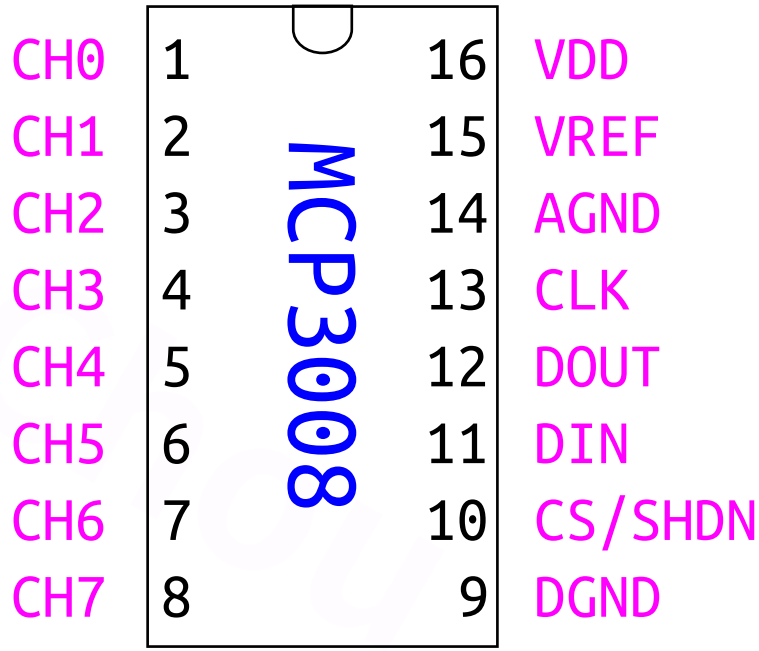
```
from machine import Pin
from time import sleep

YOUR_PIN = 23
pump = Pin(YOUR_PIN, Pin.OUT)
pump.value(0)
sleep(3)
pump.value(1)
```

Note
NC: Normally Close，預設連通
NO: Normally Open，預設斷路
COM: Common，公共點

Sensors feat. MCP3008

1. Sensors feat. MCP3008 on Raspberry
   a. Analog to Raspberry

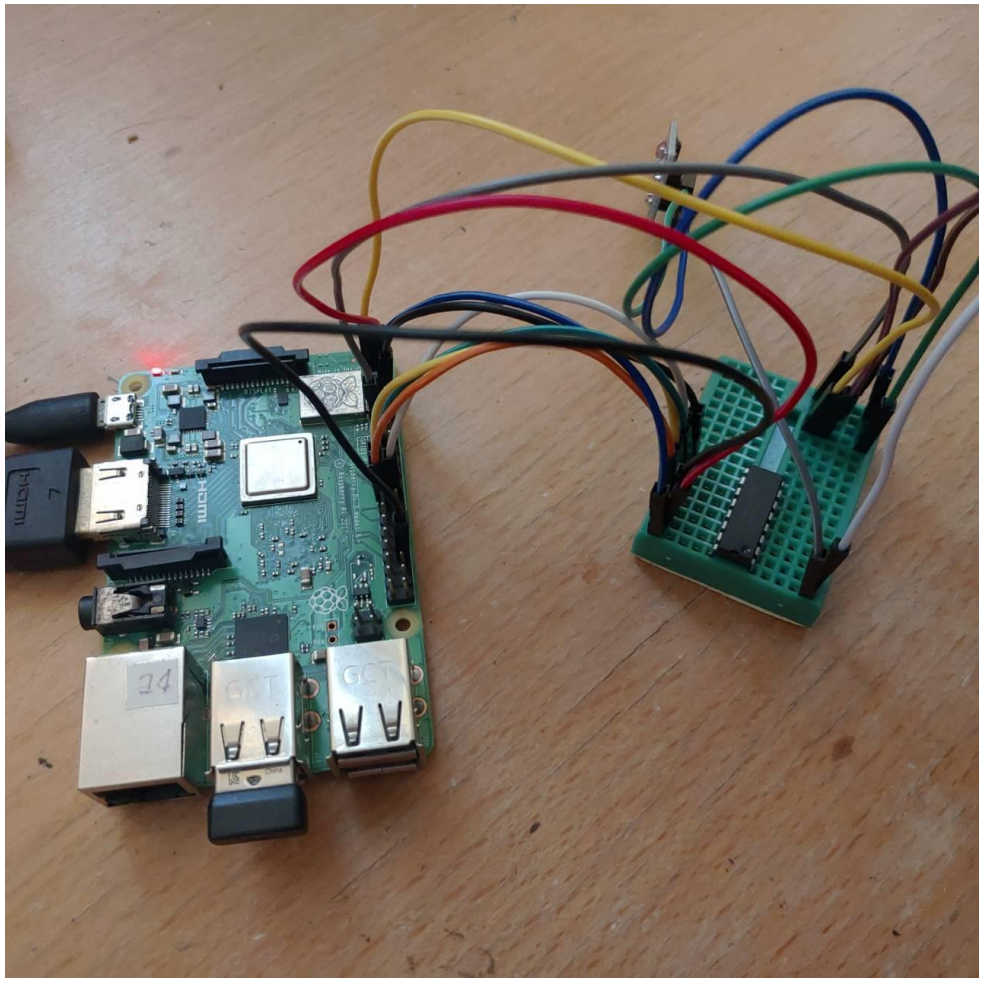
```
Sensor        MCP3008        Raspi
-          ---------------> GND
+          ---------------> 3V3
S          ----> CH0
                 VDD  ----> 5V
                 VREF ----> 5V
                 AGND ----> GND
                 CLK -----> 3V3
                 DOUT ----> SPI_MISO
                 DIN -----> SPI_MOSI
                 CS/SHDN -> SPI_CE0
                 DGND ----> GND
```



Sensors feat. MCP3008

1. Sensors feat. MCP3008 on Raspberry
   a. Analog to Raspberry

```
Sensor        MCP3008        Raspi
-             ---------------> GND
+             ---------------> 3V3
S             ----> CH0
                    VDD  ----> 5V
                    VREF ----> 5V
                    AGND ----> GND
                    CLK -----> 3V3
                    DOUT ----> SPI_MISO
                    DIN -----> SPI_MOSI
                    CS/SHDN -> SPI_CE0
                    DGND ----> GND
```



Sensors feat. MCP3008

1. Sensors feat. MCP3008 on Raspberry

   b. Enable SPI

Sensors feat. MCP3008

1. Sensors feat. MCP3008 on Raspberry

c. Module in Python

https://github.com/luxedo/ >
Repository RPi_mcp3008 >
下載 mcp3008.py

d. Code in Python

```
import mcp3008

with mcp3008.MCP3008() as adc:
    #r = adc.read_all()
    r = adc.read([mcp3008.CH0])
    print(r)
```

The End