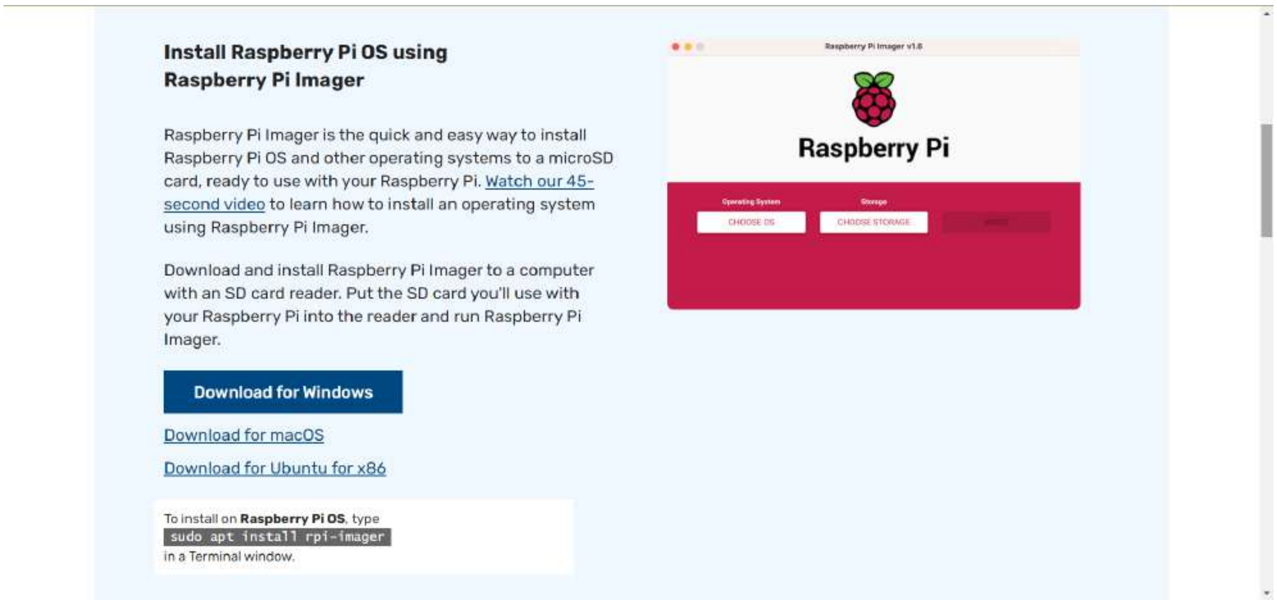
# FU740線上系統搭建文檔

## 樹莓派

### 0. 安裝系統

從官網下載相關軟件，點擊Computers->Software。

進入頁面後，在「Install Raspberry Pi OS using Raspberry Pi Imager」標題下方，可以看到不同系統的下載連結(此處以Windows為例)，根據自己的系統點擊下載Imager的安裝檔案。下載完成後安裝即可。



**Install Raspberry Pi OS using Raspberry Pi Imager**

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. Watch our 45-second video to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

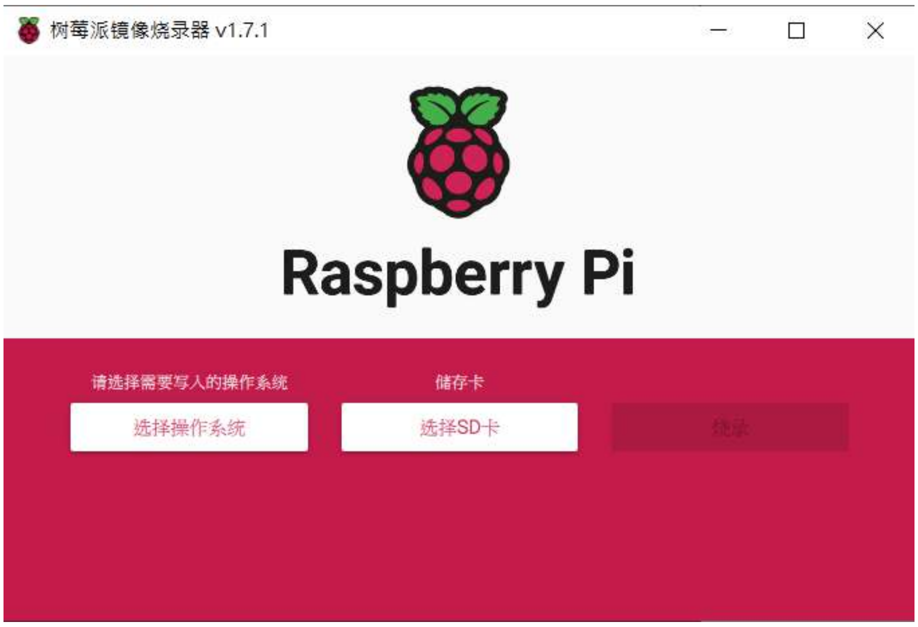
Download for Windows

Download for macOS

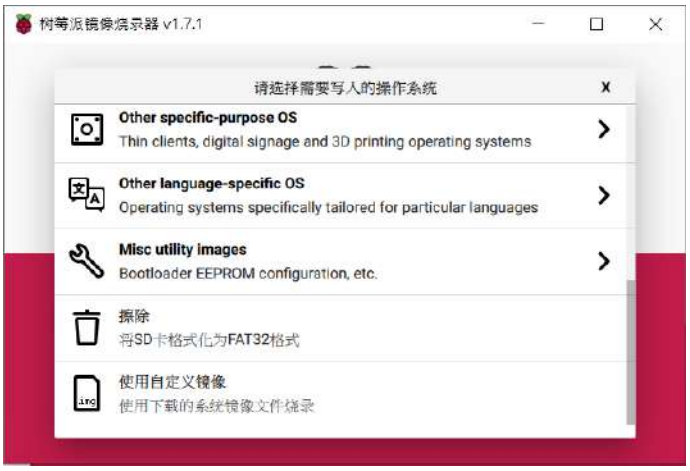
Download for Ubuntu for x86

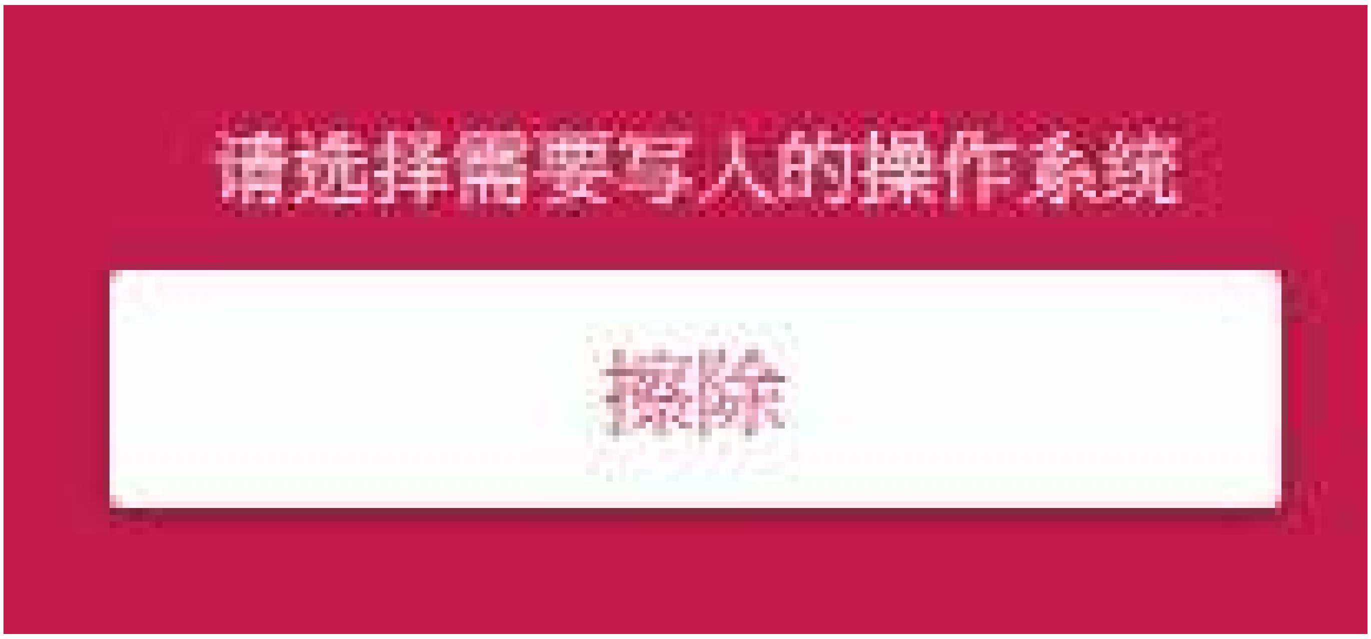
To install on **Raspberry Pi OS**, type
`sudo apt install rpi-imager`
in a Terminal window.

將SD卡利用讀卡器連接到你的裝置上，並打開Raspberry Pi Imager。

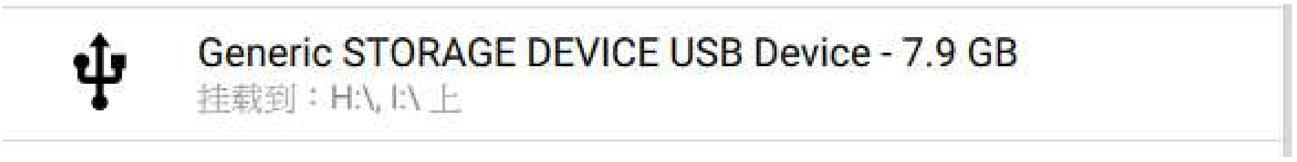


先進行格式化。先點擊選擇操作系統，選擇擦除。

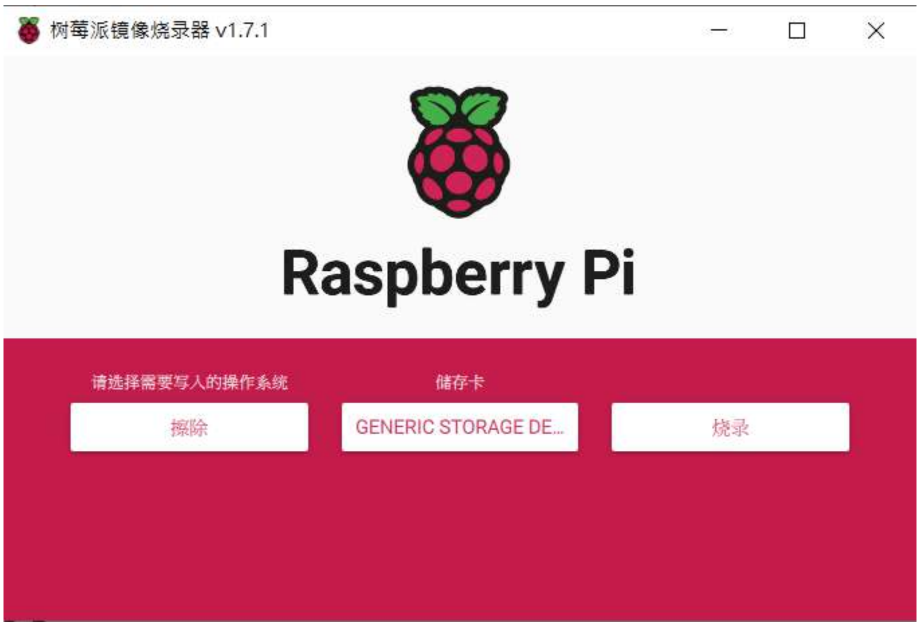

树莓派镜像烧录器 v1.7.1
请选择需要写入的操作系统
Other specific-purpose OS
Thin clients, digital signage and 3D printing operating systems
Other language-specific OS
Operating systems specifically tailored for particular languages
Misc utility images
Bootloader EEPROM configuration, etc.
擦除
将SD卡格式化为FAT32格式
使用自定义镜像
使用下载的系统镜像文件烧录



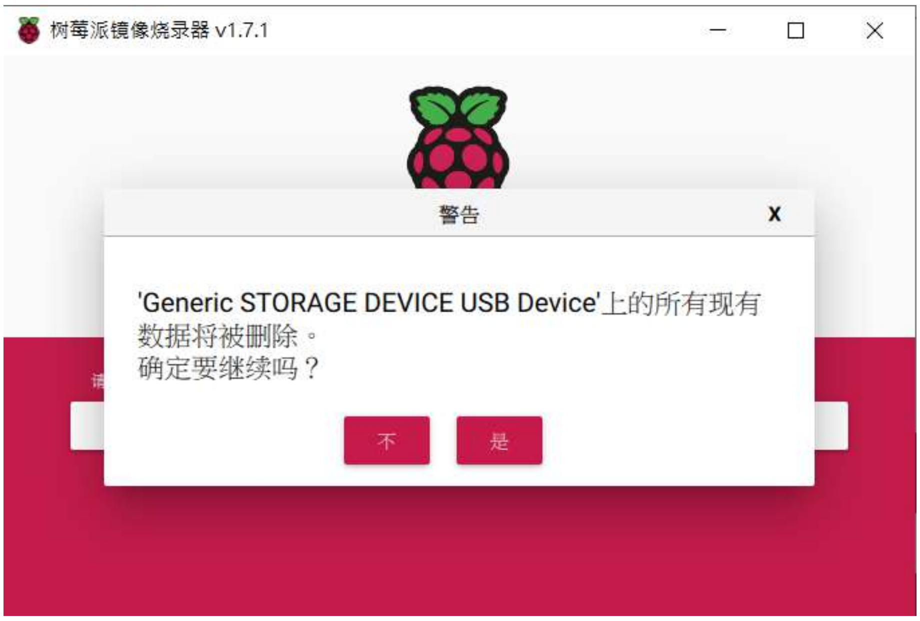
然後選擇相應的SD卡


Generic STORAGE DEVICE USB Device - 7.9 GB
挂載到：H:\, I:\ 上

點擊燒錄。

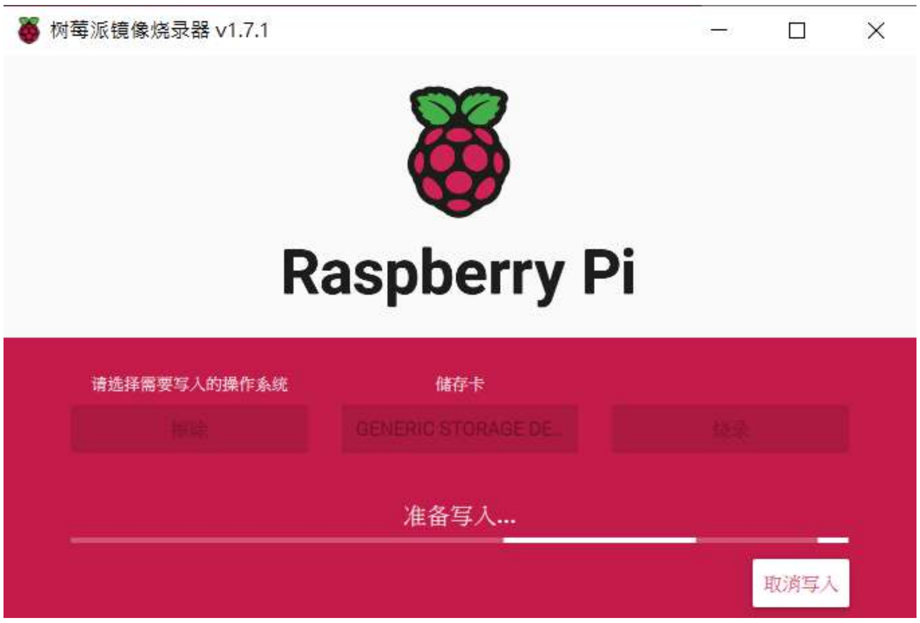


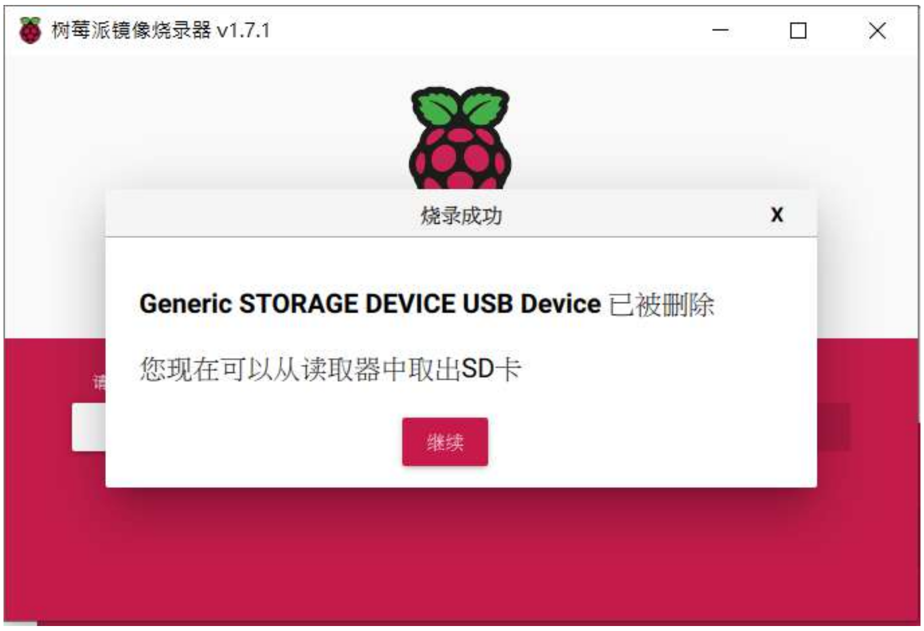
點選"是"。



'Generic STORAGE DEVICE USB Device'上的所有现有数据将被删除。
确定要继续吗？

等待擦除，擦除完畢後點擊繼續。

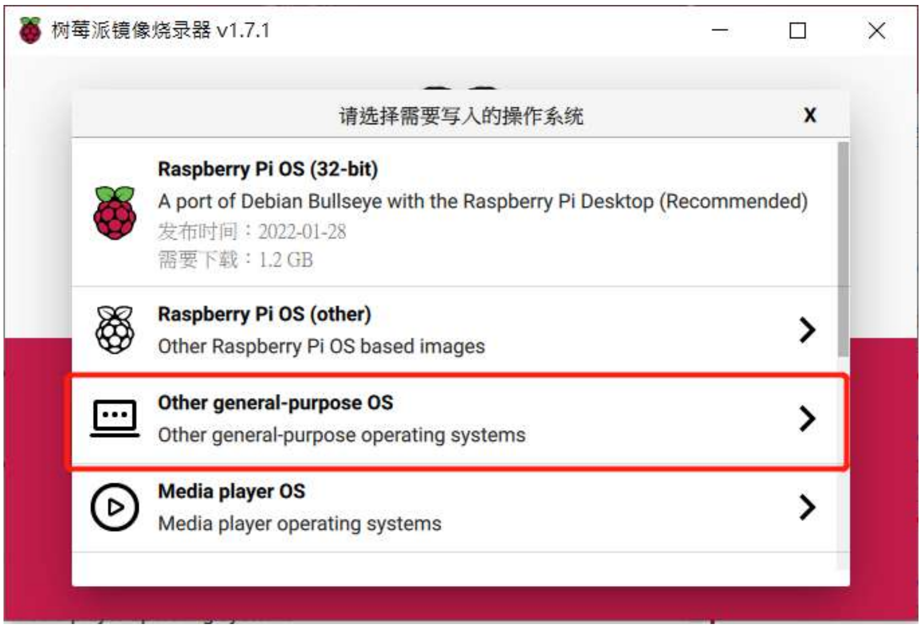




**Generic STORAGE DEVICE USB Device** 已被删除

您现在可以从读取器中取出SD卡

此時可以安裝操作系統，重新選擇適合的操作系統，此處以安裝Ubuntu 20.04版本為例。


树莓派镜像烧录器 v1.7.1

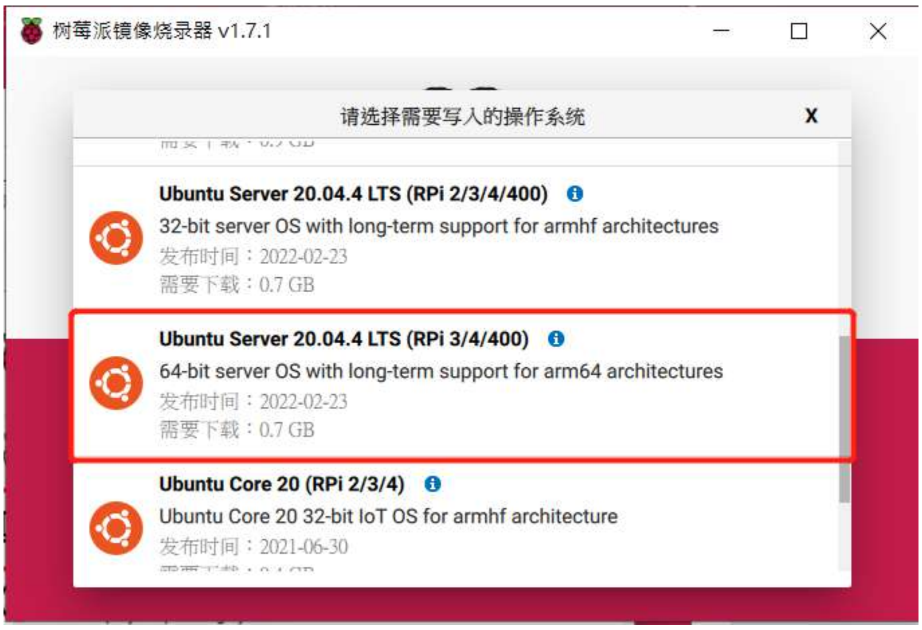
请选择需要写入的操作系统

Raspberry Pi OS (32-bit)
A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)
发布时间：2022-01-28
需要下载：1.2 GB

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS
Media player operating systems


树莓派镜像烧录器 v1.7.1

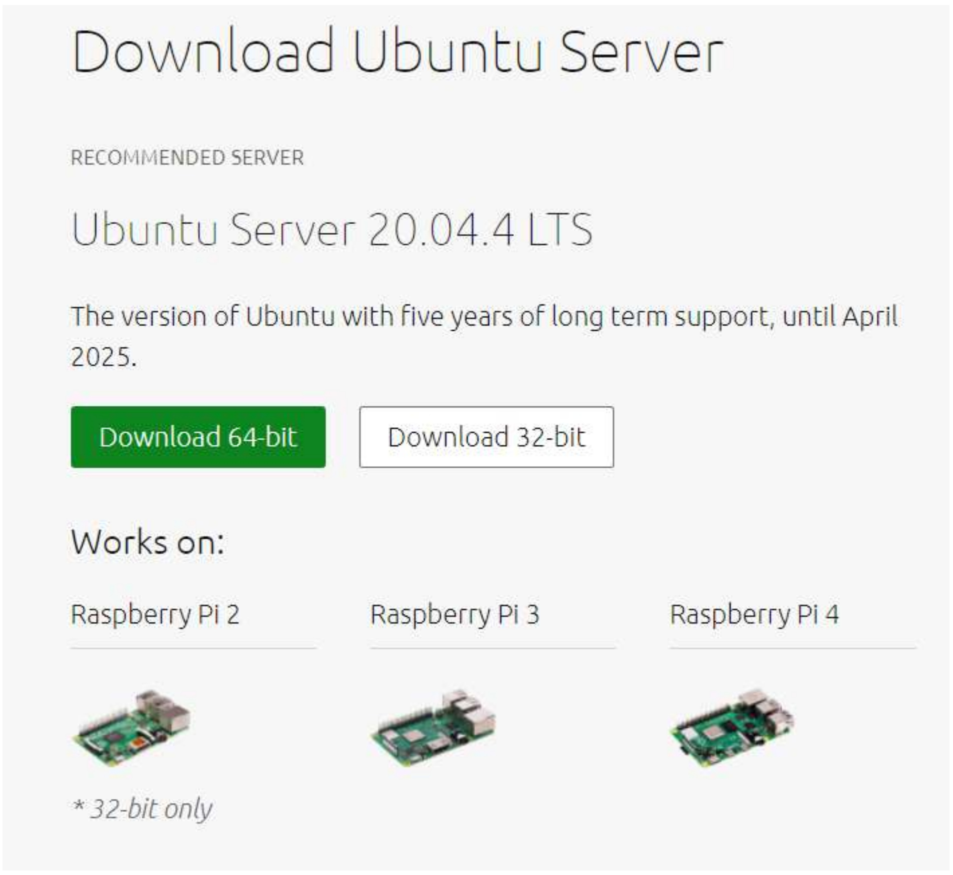
请选择需要写入的操作系统

Ubuntu Server 20.04.4 LTS (RPi 2/3/4/400)
32-bit server OS with long-term support for armhf architectures
发布时间：2022-02-23
需要下载：0.7 GB

Ubuntu Server 20.04.4 LTS (RPi 3/4/400)
64-bit server OS with long-term support for arm64 architectures
发布时间：2022-02-23
需要下载：0.7 GB

Ubuntu Core 20 (RPi 2/3/4)
Ubuntu Core 20 32-bit IoT OS for armhf architecture
发布时间：2021-06-30

或者可以在Install Ubuntu on a Raspberry Pi | Ubuntu 下載合適的鏡像，下載完成後直接點擊利用Raspberry Pi Imager打開即可。



Download Ubuntu Server

RECOMMENDED SERVER

Ubuntu Server 20.04.4 LTS

The version of Ubuntu with five years of long term support, until April 2025.

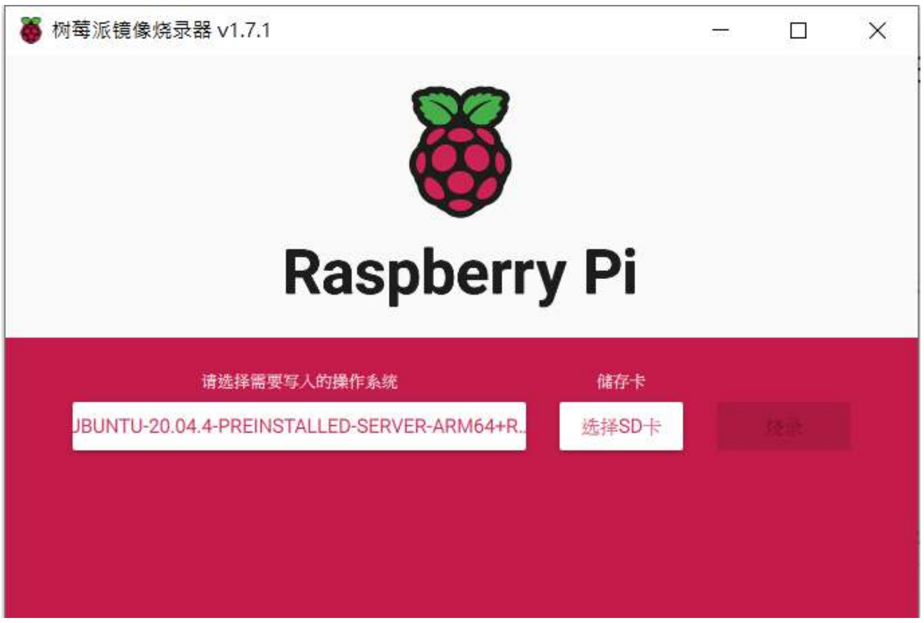
Works on:

Raspberry Pi 2

Raspberry Pi 3

Raspberry Pi 4

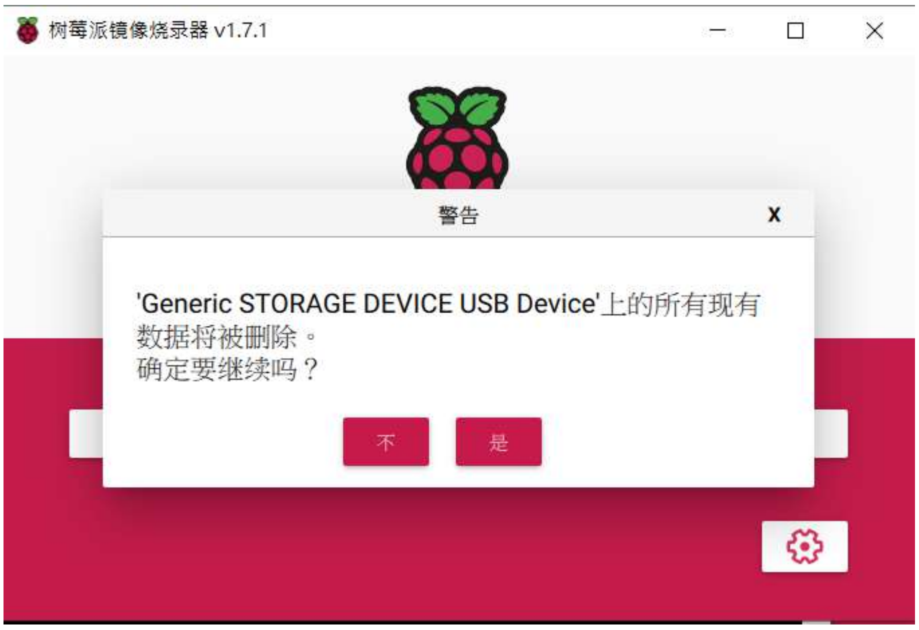
* 32-bit only



请选择需要写入的操作系统

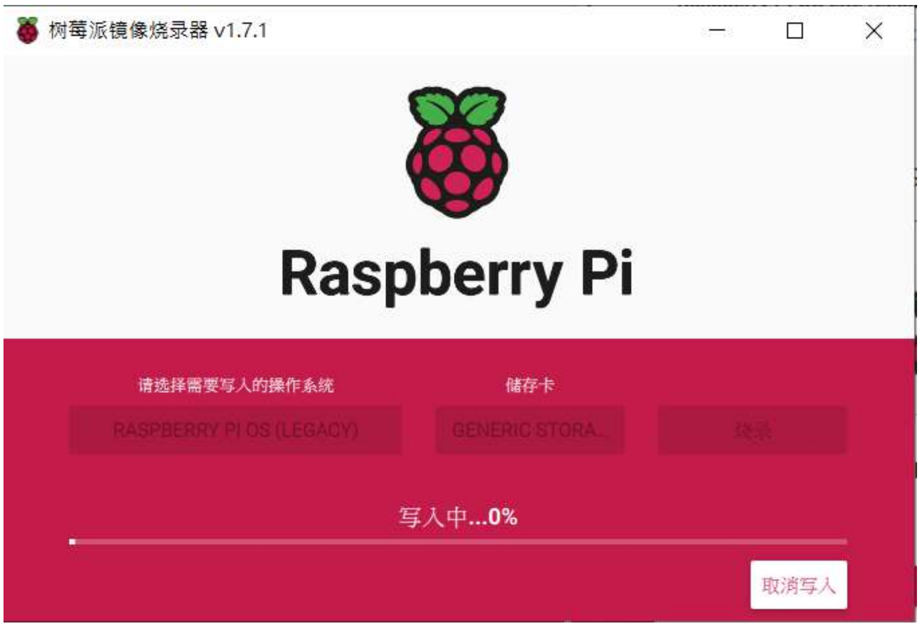
JBUNTU-20.04.4-PREINSTALLED-SERVER-ARM64+R..

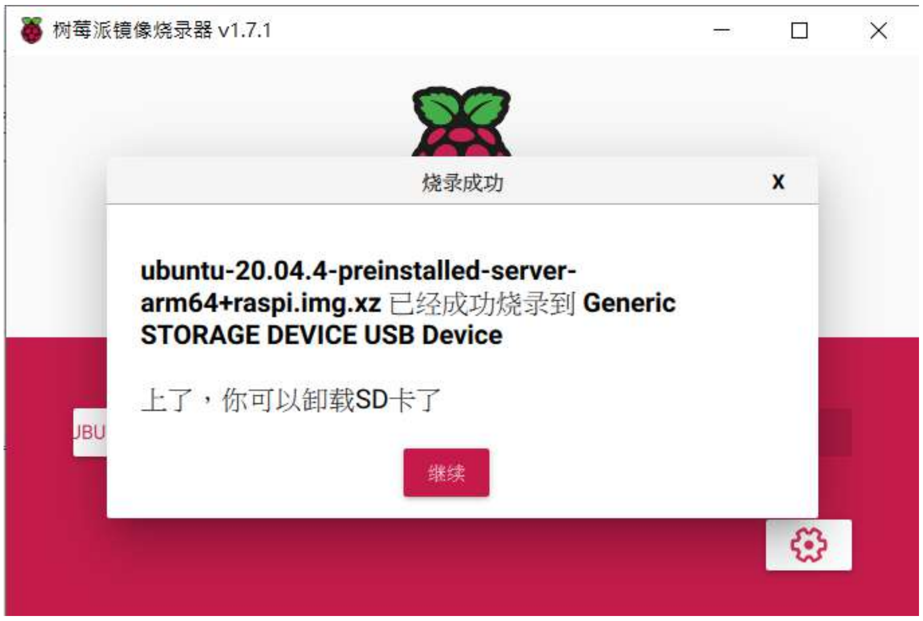
按照剛才的操作，重新選擇SD卡，並進行燒錄。



'Generic STORAGE DEVICE USB Device'上的所有现有数据将被删除。
确定要继续吗？

點擊燒錄，等待燒錄。安裝完成後把SD卡插到樹莓派上。





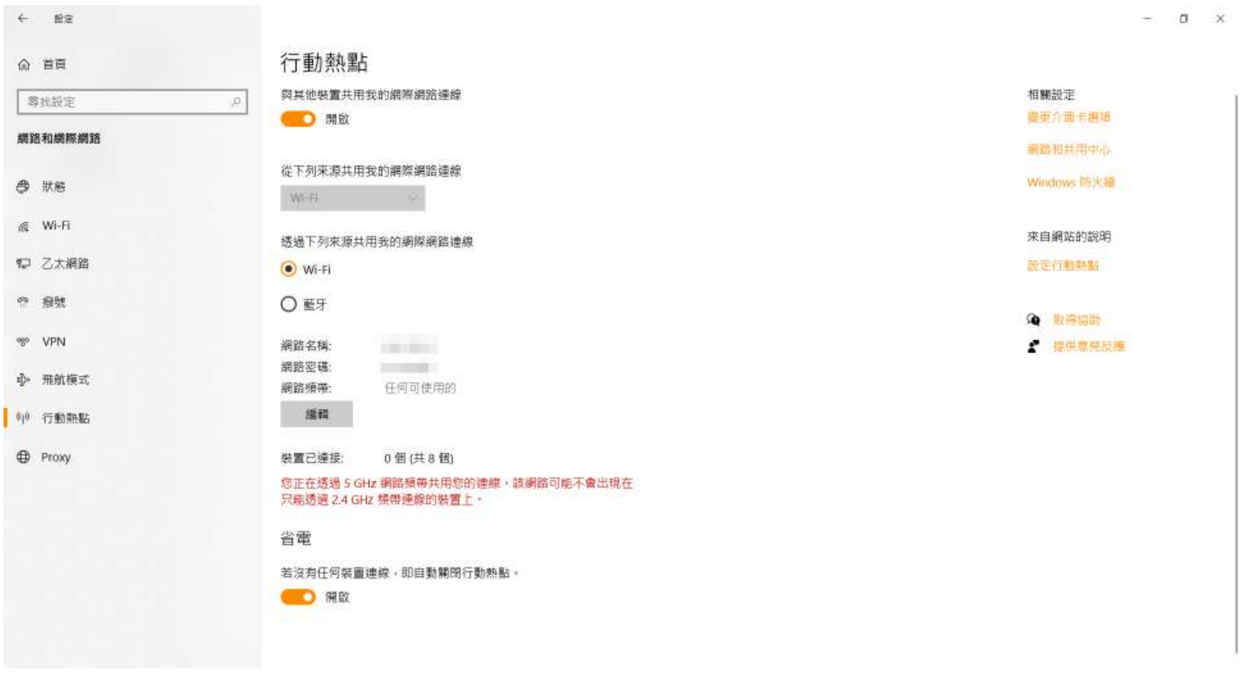
ubuntu-20.04.4-preinstalled-server-arm64+raspi.img.xz 已经成功烧录到 Generic STORAGE DEVICE USB Device

上了，你可以卸载SD卡了

### 1. 啟動樹莓派

對於開機設定，可以有兩種選擇

1. 如果有顯示器，直接接上，再接上電源線，鍵盤，鼠標等設備開機進行設置。
2. 如果沒有顯示器，便需要使用ssh來連接，對於Ubuntu第一次ssh登陸，需要有特定的操作
    1. 首先可以利用網線或者連接WIFI來讓樹莓派連上網絡，這裏是使用Windows台式機來作為初始連接，打開行動熱點頁面。



設定

首頁

尋找設定

**網路和網際網路**

- 狀態
- Wi-Fi
- 乙太網路
- 撥號
- VPN
- 飛航模式
- 行動熱點
- Proxy

## 行動熱點

與其他裝置共用我的網際網路連線

開啟

從下列來源共用我的網際網路連線

Wi-Fi

透過下列來源共用我的網際網路連線

[✓ Wi-Fi] 藍牙

網路名稱: [redacted]
網路密碼: [redacted]
網路頻帶: 任何可使用的

編輯

裝置已連接: 0 個 (共 8 個)

您正在透過 5 GHz 網路頻帶共用您的連線。該網路可能不會出現在只能透過 2.4 GHz 頻帶連線的裝置上。

### 省電

若沒有任何裝置連線，即自動關閉行動熱點。

開啟

相關設定

變更介面卡選項

網路和共用中心

Windows 防火牆

來自網站的說明

設定行動熱點

取得協助

提供意見反應

2. 在SD卡的 `/system-boot` 分區裏打開網絡配置文件 `network-config` (注意只有在初始boot時才可以使用此方法)

在最下方加入WiFi訊息，其中"wifi_name"為行動熱點的網絡名稱，"wifi_pwd"為行動熱點的網絡密碼。

```
# This file contains a netplan-compatible configuration which cloud-init will
# apply on first-boot (note: it will *not* update the config after the first
# boot). Please refer to the cloud-init documentation and the netplan
reference
# for full details:
#
# https://cloudinit.readthedocs.io/en/latest/topics/network-config.html
# https://cloudinit.readthedocs.io/en/latest/topics/network-config-format-
v2.html
# https://netplan.io/reference
#
# Please note that the YAML format employed by this file is sensitive to
# differences in whitespace; if you are editing this file in an editor (like
# Notepad) which uses literal tabs, take care to only use spaces for
# indentation. See the following link for more details:
#
# https://en.wikipedia.org/wiki/YAML
#
# Some additional examples are commented out below

   version: 2
   ethernets:
     eth0:
       dhcp4: true
       optional: true
   #wifis:
   #  wlan0:
   #    dhcp4: true
   #    optional: true
   #    access-points:
   #      myhomewifi:
   #        password: "S3kr1t"
   #      myworkwifi:
   #        password: "correct battery horse staple"
   #      workssid:
   #        auth:
   #          key-management: eap
   #          method: peap
   #          identity: "me@example.com"
   #          password: "passw0rd"
   #          ca-certificate: /etc/my_ca.pem
```

在最下方加入WiFi訊息，其中"wifi_name"為行動熱點的網絡名稱，"wifi_pwd"為行動熱點的網絡密碼。

```
wifis:
    wlan0:
      dhcp4: true
      dhcp6: true
      optional: true
      access-points:
        "wifi_name":
          password: "wifi_pwd"
```

3. 配置用戶名。ubuntu 留有一组默认的用户名, 要求用户第一次登陆的时候改掉它. 但是第一次接通电源的时候是无法连接到 WiFi 的, 需要断电再上电. 此时通过 SSH 登陆会遇到问题 `passwd: Authentication token manipulation error` 打开文件 `user-data`, 找到

   ```
   chpasswd:
     expire: true
     list:
     - ubuntu:ubuntu
   ```

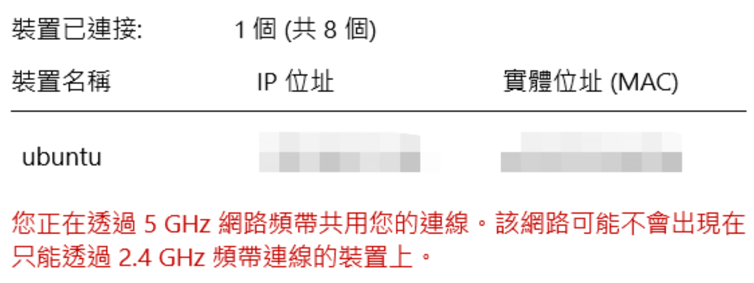
   把 `expire: true` 改为 `expire: false` 这样系统可以用这组默认用户名密码 `ubuntu:ubuntu` 登陆系统, 直到修改密码.

4. 保存文件，弹出SD卡

5. 把SD卡插入树莓派

6. 接上电源

7. 通过主机的热点界面观察树莓派是否接入

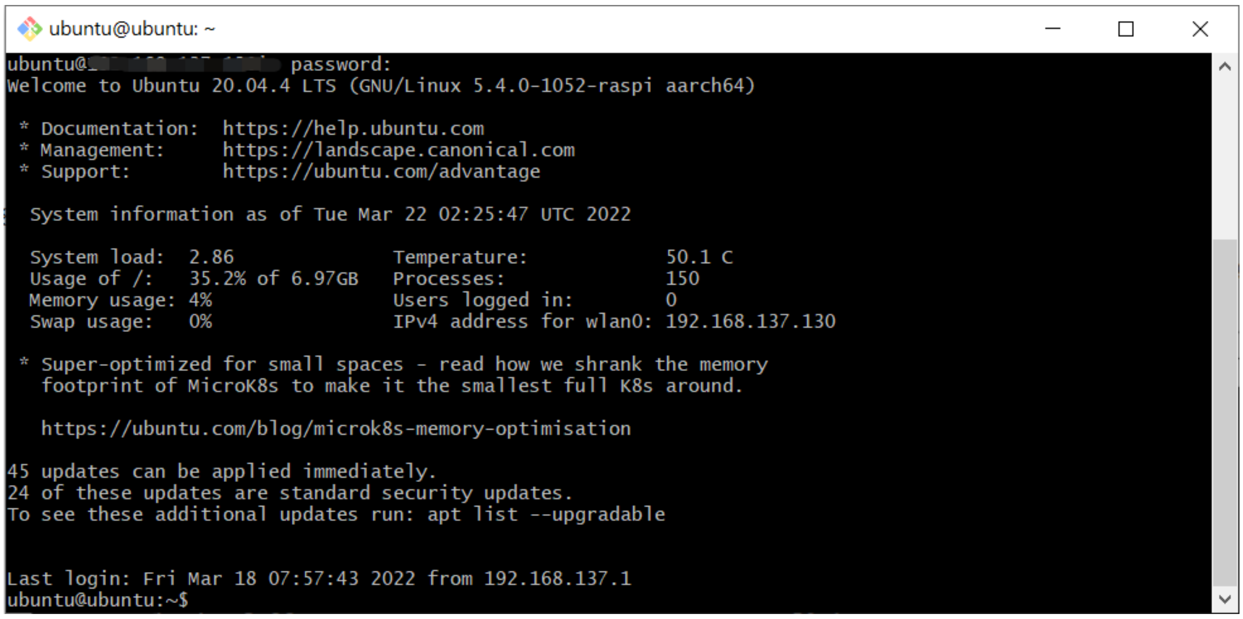


装置已連接: 1 個 (共 8 個)

| 裝置名稱 | IP 位址 | 實體位址 (MAC) |
|---|---|---|
| ubuntu | [redacted] | [redacted] |

您正在透過 5 GHz 網路頻帶共用您的連線。該網路可能不會出現在只能透過 2.4 GHz 頻帶連線的裝置上。

8. 然後可以通過ssh連接到樹莓派上，預設密碼為ubuntu



```
ubuntu@[redacted] password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-1052-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Mar 22 02:25:47 UTC 2022

  System load:  2.86              Temperature:           50.1 C
  Usage of /:   35.2% of 6.97GB   Processes:             150
  Memory usage: 4%                Users logged in:       0
  Swap usage:   0%                IPv4 address for wlan0: 192.168.137.130

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

45 updates can be applied immediately.
24 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Fri Mar 18 07:57:43 2022 from 192.168.137.1
ubuntu@ubuntu:~$
```

9. 可以利用公鑰免密SSH登入

    1. 在命行中輸入`ssh-keygen`，一直按`enter`即可。
    2. 利用`ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@pi_ip`，上傳金鑰到樹莓派上。
    3. 然後在下次登陸時便不需要輸入密碼了。

## 2. 連接Tsinghua-Secure

### 配置靜態IP

為了防止開機時IP可能會改變的情況，可以添加一個固定IP。

1. 修改dhcpcd設定，添加如下語句

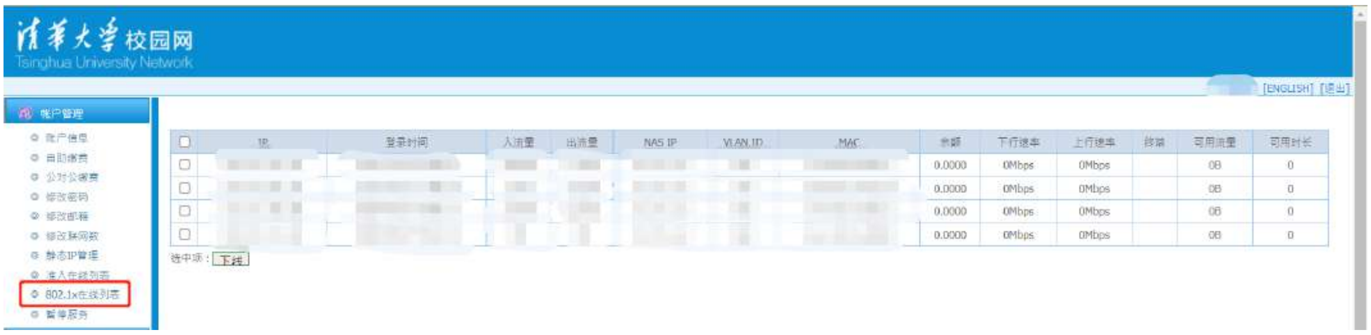
```
$ sudo vim /etc/dhcpcd.conf
```

```
interface wlan0
static ip_address= YOUR_STATIC_IP
static routers=YOUR_ROUTER_IP
static domain_name_servers=YOUR_DNS
```

### 動態IP方法

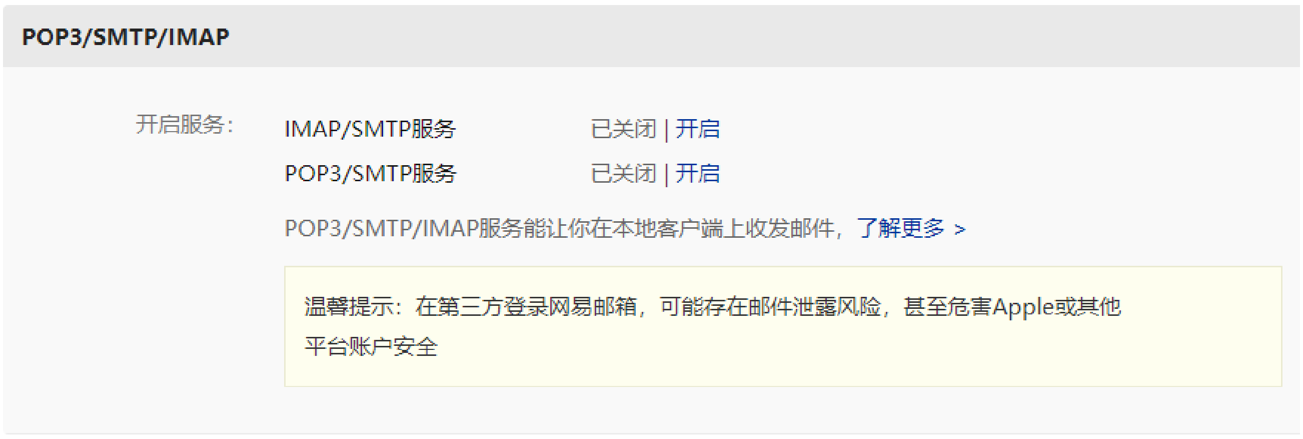
如果網絡條件不允許把IP設成固定的，可以利用這個方法。

為了能夠得知設備更換WiFi後的IP地址，撰寫一個簡單的開機發IP地址到郵件的腳本，參考[網址](使用 Python 将树莓派的 IP 发到邮箱 | 树莓派实验室 (nxez.com))。先安裝python `sudo apt` `install python` 、 `sudo apt install net-tools` 。(或者可以直接登上[這裏](清华大学校园网自助服务系统 (tsinghua.edu.cn))查看)



| | IP | 登录时间 | 入流量 | 出流量 | NAS IP | VLAN ID | MAC | 余额 | 下行速率 | 上行速率 | 终端 | 可用流量 | 可用时长 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | 0.0000 | 0Mbps | 0Mbps | | 0B | 0 |
| ☐ | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | 0.0000 | 0Mbps | 0Mbps | | 0B | 0 |
| ☐ | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | 0.0000 | 0Mbps | 0Mbps | | 0B | 0 |
| ☐ | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | [redacted] | 0.0000 | 0Mbps | 0Mbps | | 0B | 0 |

1. 以163mail為例，163 mail需要在設置裏，開啟smtp服務功能，然後記住所得到的授權密碼。

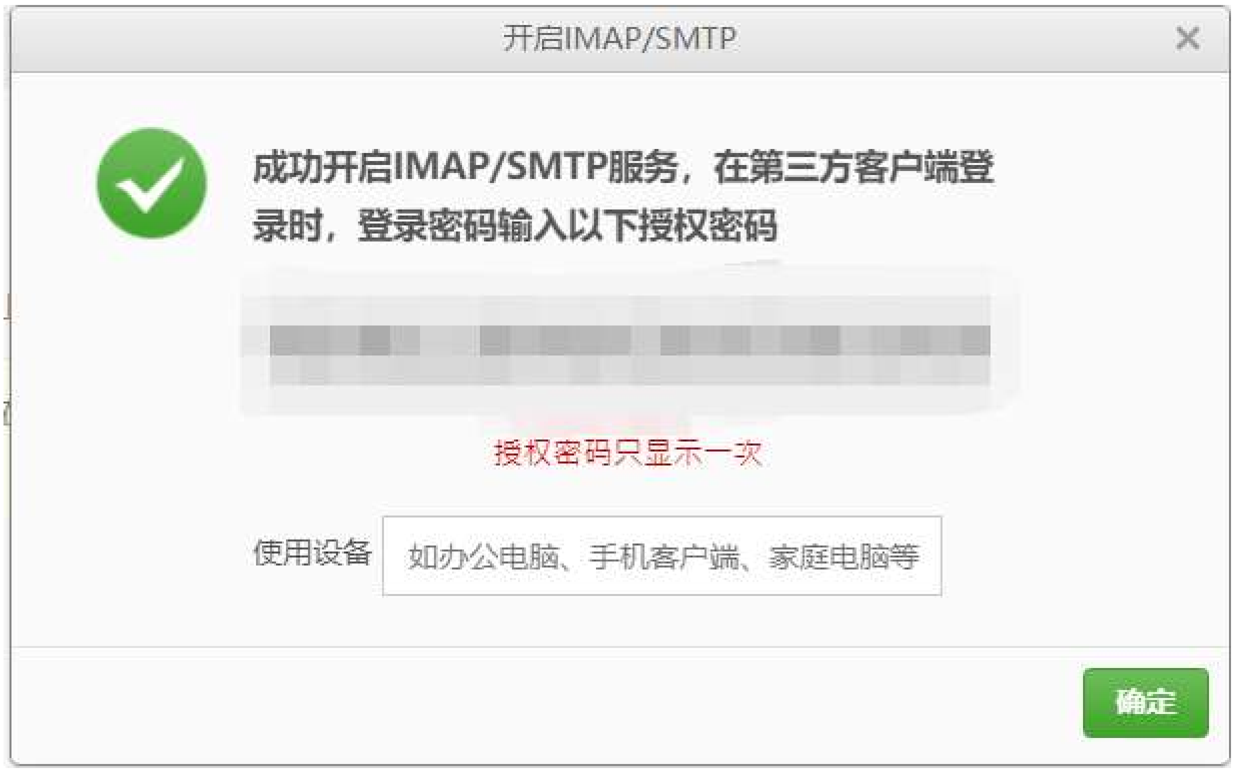


**POP3/SMTP/IMAP**

| | | |
|---|---|---|
| 开启服务： | IMAP/SMTP服务 | 已关闭 \| 开启 |
| | POP3/SMTP服务 | 已关闭 \| 开启 |

POP3/SMTP/IMAP服务能让你在本地客户端上收发邮件，了解更多 >

温馨提示：在第三方登录网易邮箱，可能存在邮件泄露风险，甚至危害Apple或其他平台账户安全



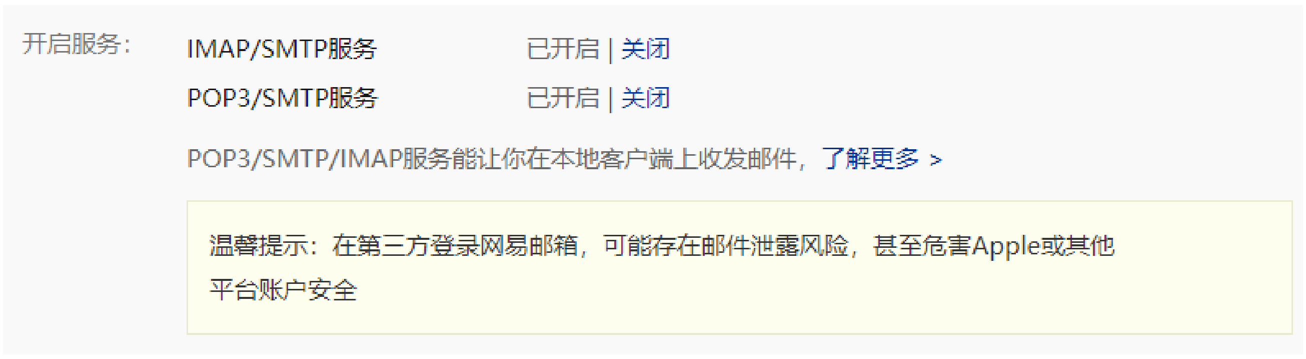
开启IMAP/SMTP

**成功开启IMAP/SMTP服务，在第三方客户端登录时，登录密码输入以下授权密码**

[redacted]

授权密码只显示一次

使用设备 如办公电脑、手机客户端、家庭电脑等


开启服务：IMAP/SMTP服务 已开启 | 关闭
POP3/SMTP服务 已开启 | 关闭
POP3/SMTP/IMAP服务能让你在本地客户端上收发邮件，了解更多 >
温馨提示：在第三方登录网易邮箱，可能存在邮件泄露风险，甚至危害Apple或其他平台账户安全

mail.py

```
#!/usr/bin/python
#-*- coding:utf-8 -*-
import smtplib
from email.mime.text import MIMEText
import sys
mail_host = 'smtp.163.com' #smtp地址
mail_user = 'username' #你的郵箱帳號
mail_pass = 'smtp_code' #smtp服務中提供的授權密碼
mail_postfix = '163.com'

def send_mail(to_list,subject,content):
      me = 'myPi'+"<"+mail_user+"@"+mail_postfix+">"
      msg = MIMEText(content)
      msg['Subject'] = subject
      msg['From'] = me
      msg['to'] = to_list

      try:
          s = smtplib.SMTP_SSL()
          s.connect(mail_host)
          s.login(mail_user,mail_pass)
          s.sendmail(me,to_list,msg.as_string())
          s.close()
          return True
      except Exception,e:
          print str(e)
          return False

  if __name__ == "__main__":
          send_mail(sys.argv[1], sys.argv[2], sys.argv[3])
```

ip.sh

```
#!/bin/bash
  sleep 20
  wlan0=`ifconfig  wlan0 | head -n2 | grep inet | awk '{print$2}'`
  eth0=`ifconfig  eth0 | head -n2 | grep inet | awk '{print$2}'`

  if ping -c 2 -W 3 www.baidu.com &> /dev/null ;then
  /home/ubuntu/mail.py striketong@163.com "Got raspberryPi IP!"
"WLAN0:$wlan0||ETH0:$eth0"
  fi
```

2. 給予兩個腳本權限

```
sudo chmod +x ip.sh
sudo chmod +x mail.py
```

3. 將腳本加入開機執行文件。這邊由於沒有rc.local，需要自己創建，參考網址

    1. 創建systemd的服务腳本，`sudo vim /etc/systemd/system/rc-local.service`

    ```
    [Unit]
     Description=/etc/rc.local Compatibility
     ConditionPathExists=/etc/rc.local

    [Service]
     Type=forking
     ExecStart=/etc/rc.local start
     TimeoutSec=0
     StandardOutput=tty
     RemainAfterExit=yes
     SysVStartPriority=99

    [Install]
     WantedBy=multi-user.target
    ```

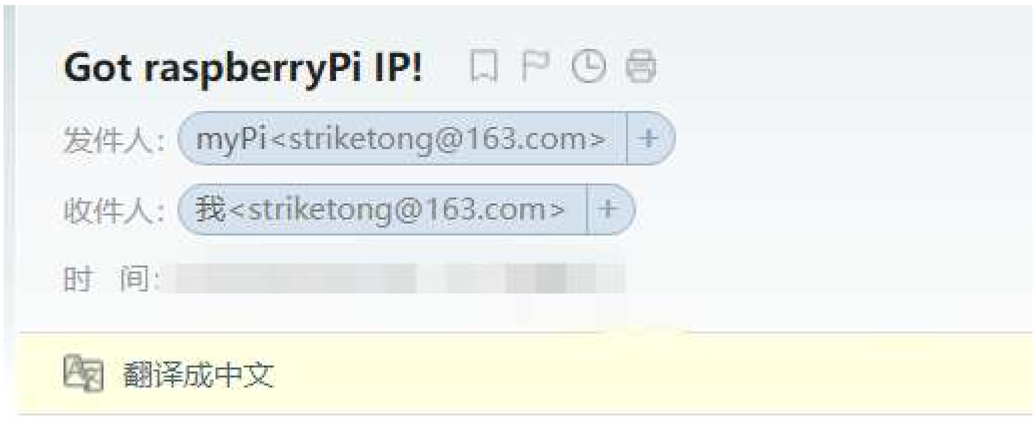
    2. 啟動服務

    ```
    sudo systemctl enable rc-local.service
    ```

    3. 編寫rc.local，`sudo vim /etc/rc.local`

```
#!/bin/bash

sudo /home/ubuntu/ip.sh
```

4. 給予權限 `sudo chmod +x /etc/rc.local`
5. 嘗試重啟設備 `sudo reboot`，看能否收到郵件
6. 成功收到帶有IP的郵件

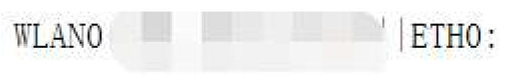


Got raspberryPi IP!

发件人: myPi<striketong@163.com>

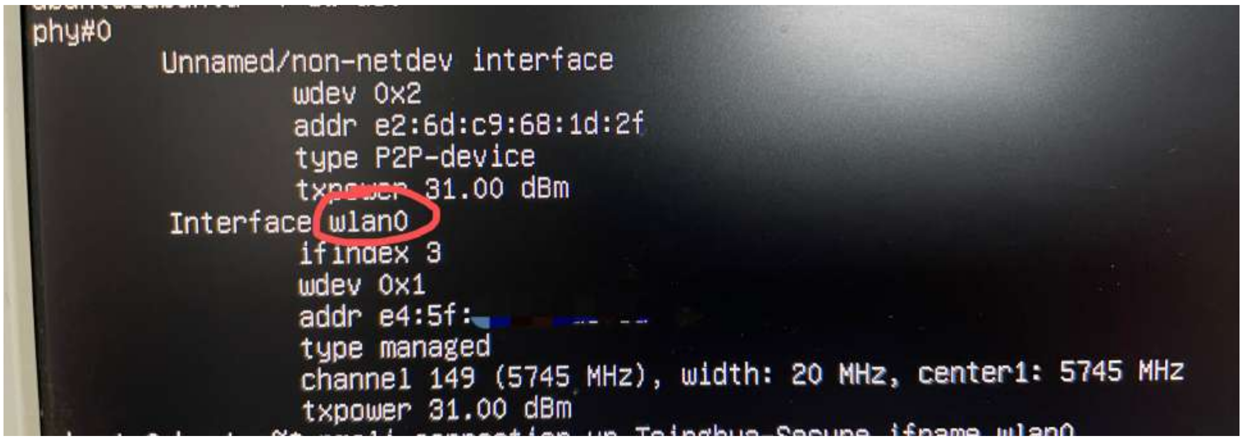
收件人: 我<striketong@163.com>

时 间: [redacted]



WLAN0 [redacted] |ETH0:

7. 根據學長們的[使用手冊](服务 - THU Services)，通過`wpa_supplicant`進行Tsinghua-Secure的配置
8. 通過顯示屏，直接在樹莓派的Terminal 中，通过命令“iw dev”查看无线网卡的标识：



```
phy#0
        Unnamed/non-netdev interface
                wdev 0x2
                addr e2:6d:c9:68:1d:2f
                type P2P-device
                txpower 31.00 dBm
        Interface wlan0
                ifindex 3
                wdev 0x1
                addr e4:5f:[redacted]
                type managed
                channel 149 (5745 MHz), width: 20 MHz, center1: 5745 MHz
                txpower 31.00 dBm
```

9. 输入如下命令，创建用于连接 802.1x 无线网络的配置文件：

编辑 `vim /etc/wpa_supplicant/wpa_supplicant-nl80211-wlan0.conf` ，其中 `wlan0` 是本机网卡名称，输入以下配置

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
        ssid="Tsinghua-Secure"
        proto=RSN
        key_mgmt=WPA-EAP
        pairwise=CCMP
        eap=PEAP
        identity="username"
        password="password"
        phase2="auth=MSCHAPV2"
        priority=9
}
```

其中 `username` 与 `password` 为自己帐号相应信息。

10. 重新編輯 `ip.sh`

```
#!/bin/bash
sudo killall wpa_supplicant
sleep 5
sudo wpa_supplicant -B -c /etc/wpa_supplicant/wpa_supplicant-nl80211-
wlan0.conf -i wlan0
sudo dhclient wlan0
sleep 30
wlan0=`ifconfig  wlan0 | head -n2 | grep inet | awk '{print$2}'`
eth0=`ifconfig  eth0 | head -n2 | grep inet | awk '{print$2}'`

if ping -c 2 -W 3 www.baidu.com &> /dev/null ;then
/home/ubuntu/mail.py striketong@163.com "Got raspberryPi IP!"
"WLAN0:$wlan0||ETH0:$eth0"
fi
```

11. 重啟測試 `sudo reboot` ，看是否能自動連上Tsinghua-Secure

12. 開機自動連接成功。

## 3. 修改SSH port

由於校園網禁止22端口，所以需要修改預設端口。改成你想要的端口。需注意，0-1024，8000-8100，3389，9100這些都不能用。

1. `sudo vim /etc/ssh/ssh_config`、`sudo vim /etc/ssh/sshd_config`，找到`#Port 22`，改成`Port 8122`。
2. 設置防火牆

   ```
   $ sudo ufw allow 8122
   ```

3. 重啟SSH服務

   ```
   $ sudo service sshd restart
   ```

4. 嘗試利用SSH連接`ssh -p 8122 ubuntu@ip`，成功。

### 4. 配置picocom

利用picocom來獲取串口的調試訊息，甚至可在串口直接為FU740安裝系統。

1. 安裝picocom

   ```
   $ sudo apt-get install picocom
   ```

### 5. 配置usbip

FU740裏有內建的UART/JTAG to USB可以直接利用usb訪問串口。而對於遠程訪問，可以利用usbip遠程訪問USB，繼而訪問串口。

1. 安裝usbip

   ```
   $ sudo apt install linux-tools-common
   $ sudo apt install linux-tools-5.4.0-1058-raspi
   ```

2. 配置

```
# server
# load usbip kernel module
$ sudo modprobe usbip_host
# start control daemon
$ sudo usbipd
# list available local devices
$ usbip list -l
# export USB device
$ sudo usbip bind -b <busid>
#sudo usbip bind -b 1-1.1
#sudo usbip bind -b 1-1.3
```

3. 若想在開機時自動啟動，編寫 `sudo vim /etc/systemd/system/usbip.service` ，$(uname -r)替換成相應的板本號。

   如

```
$ echo $(uname -r)
5.4.0-1058-raspi
```

   usbip.service

```
[Unit]
Description=usbip host daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/lib/linux-tools/$(uname -r)/usbipd &
ExecStartPost=/bin/sh -c "/usr/lib/linux-tools/$(uname -r)/usbip bind -b 1-1.1"
ExecStartPost=/bin/sh -c "/usr/lib/linux-tools/$(uname -r)i/usbip bind -b 1-1.3"
ExecStop=/bin/sh -c "/usr/lib/linux-tools/$(uname -r)/usbip unbind -b 1-1.1"
ExecStop=/bin/sh -c "/usr/lib/linux-tools/$(uname -r)/usbip unbind -b 1-1.3"

[Install]
WantedBy=multi-user.target
```

4. 啟用服務

```
$ sudo systemctl --system daemon-reload
$ sudo systemctl enable usbip.service
$ sudo systemctl start usbip.service
```

5. 用戶端配置

```
# client
# load vhci-hcd kernel module
$ sudo modprobe vhci-hcd
# list available remote devices
$ usbip list -r <server>
# attach remote device locally
$ usbip attach -r <server> -b <busid>
```

## 6. 配置tftp服務器

需要利用tftp來為FU740安裝系統。

1. 安裝tftpd，`sudo apt-get install xinetd tftpd-hpa`
2. 建立 TFTP 的配置文件，`sudo vim /etc/xinetd.d/tftp`

```
service tftp
{
    socket_type     = dgram
    protocol         = udp
    wait                    = yes
    user                        = root
    server          = /usr/sbin/in.tftpd
    server_args     = -s /srv/tftp -c
    disable     = no
    per_source    = 11
    cps             = 100 2
    flags              = IPv4
}
```

3. 更改配置文件，`sudo vim /etc/default/tftpd-hpa`，添加`--create`

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS=":8122"
TFTP_OPTIONS="-c -s -l"
```

4. 更改工作文件夾權限

```
$ sudo chown tftp:tftp /srv/tftp
```

5. 啟動服務

```
$ sudo /etc/init.d/xinetd restart
$ sudo /etc/init.d/tftpd-hpa restart
```

6. 查看服務是否正在運行

```
$ netstat -a |grep tftp
udp        0      0 0.0.0.0:tftp            0.0.0.0:*

$ sudo systemctl status tftpd-hpa

● tftpd-hpa.service - LSB: HPA's tftp server
     Loaded: loaded (/etc/init.d/tftpd-hpa; generated)
     Active: active (running) since Mon 2022-04-25 13:17:45 UTC; 2s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 2344985 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, statu>
      Tasks: 1 (limit: 9257)
     CGroup: /system.slice/tftpd-hpa.service
             └─2345011 /usr/sbin/in.tftpd --listen --user tftp --address 0.0.0>

Apr 25 13:17:45 rpiroot systemd[1]: Starting LSB: HPA's tftp server...
Apr 25 13:17:45 rpiroot tftpd-hpa[2344985]:  * Starting HPA's tftpd in.tftpd
Apr 25 13:17:45 rpiroot tftpd-hpa[2344985]:    ...done.
Apr 25 13:17:45 rpiroot systemd[1]: Started LSB: HPA's tftp server.
```

7. 客戶端請查看板本是否是tftp-hpa，如否請安裝tftp-hpa以配對服務器上的tftpd-hpa

```
$ dpkg -l tftp-hpa
$ sudo apt install tftp-hpa
```

## 7. 配置dhcp

1. 安裝dhcp

```
$ sudo apt-get install isc-dhcp-server
```

2. 啟用服務

```
$ sudo systemctl start isc-dhcp-server
$ sudo systemctl enable isc-dhcp-server
```

3. 配置DHCP服務器文件，`sudo vim /etc/dhcp/dhcpd.conf`，添加如下內容

```
subnet 10.0.1.0 netmask 255.255.255.0 {
   authoritative;
   range 10.0.1.5 10.0.1.255;
   default-lease-time 3600;
   max-lease-time 3600;
   option subnet-mask 255.255.255.0;
   option broadcast-address 10.0.1.255;
   option routers 10.0.1.5;
   option domain-name-servers 166.111.8.28; # 清華的DNS
}
```

4. 設定DHCP啟用的網絡卡，`sudo vim /etc/default/isc-dhcp-server`，`sudo vim /etc/default/dhcpd.conf`

```
INTERFACESv4="eth0"
```

5. 重啟DHCP服務

```
$ sudo systemctl restart isc-dhcp-server
```

6. 查看DHCP服務是否運行

```
$ sudo systemctl status isc-dhcp-server

lines 1--1...skipping...
● isc-dhcp-server.service - ISC DHCP IPv4 server
     Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor
preset:>
     Active: active (running) since Thu 2022-03-31 12:40:41 UTC; 12s ago
       Docs: man:dhcpd(8)
   Main PID: 91898 (dhcpd)
      Tasks: 4 (limit: 9257)
     CGroup: /system.slice/isc-dhcp-server.service
             └─91898 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-
server/dhcpd.pid >

Mar 31 12:40:41 ubuntu dhcpd[91898]: PID file: /run/dhcp-server/dhcpd.pid
Mar 31 12:40:41 ubuntu sh[91898]: Wrote 0 leases to leases file.
Mar 31 12:40:41 ubuntu dhcpd[91898]: Wrote 0 leases to leases file.
Mar 31 12:40:41 ubuntu dhcpd[91898]: Listening on
LPF/eth0/e4:5f:01:56:d8:57/10.0.1.0/24
Mar 31 12:40:41 ubuntu sh[91898]: Listening on
LPF/eth0/e4:5f:01:56:d8:57/10.0.1.0/24
Mar 31 12:40:41 ubuntu sh[91898]: Sending on
LPF/eth0/e4:5f:01:56:d8:57/10.0.1.0/24
Mar 31 12:40:41 ubuntu sh[91898]: Sending on   Socket/fallback/fallback-net
Mar 31 12:40:41 ubuntu dhcpd[91898]: Sending on
LPF/eth0/e4:5f:01:56:d8:57/10.0.1.0/24
Mar 31 12:40:41 ubuntu dhcpd[91898]: Sending on   Socket/fallback/fallback-net
Mar 31 12:40:41 ubuntu dhcpd[91898]: Server starting service.
~
~
```

7. 為樹莓派配置固定IP，修改DHCP服務器文件 `sudo vim /etc/dhcp/dhcpd.conf` ，添加host，此後在每次分配時樹莓派上的有線網絡便會得到固定的IP地址。

```
subnet 10.0.1.0 netmask 255.255.255.0 {
   authoritative;
   range 10.0.1.5 10.0.1.255;
   default-lease-time 3600;
   max-lease-time 3600;
   option subnet-mask 255.255.255.0;
   option broadcast-address 10.0.1.255;
   option routers 10.0.1.5;
}
host rpiroot {
    hardware ethernet e4:5f:01:56:d8:57;
    fixed-address 10.0.1.5;
}
host ubuntu { #在首次登入後，可根據板子的網卡MAC地址來分配固定IP
    hardware ethernet XX:XX:XX:XX:XX:XX;
    fixed-address 10.0.1.10;
}
```

### 8. 建立遠程響應網站

為了避免每次登入樹莓派才能啟動FU740，在此利用http請求實現一個更簡單的開機關機的方法。

1. 安裝Flask，`sudo apt-get install -y python3-flask`
2. 編輯`power.py`，利用Flask建立簡易網站用以接收開關機操作請求。

```python
from flask import Flask
import RPi.GPIO as GPIO
import gpiod
import time

def button(secs, line):
    with gpiod.Chip("gpiochip0") as chip:
        line = chip.get_line(line)
        line.request(consumer="powerd", type=gpiod.LINE_REQ_DIR_OUT)
        line.set_value(1)
        time.sleep(secs)
        line.set_value(0)

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"

@app.route("/poweron1")
def poweron1():
    button(1, 14)
    return "<p>Power on machine-1!</p>"

@app.route("/poweroff1")
def poweroff1():
    button(4, 14)
    return "<p>Power off machine-1!</p>"""

@app.route("/reboot1")
def reboot1():
    button(4, 14)
    time.sleep(1)
    button(1, 14)
    return "Rebooted machine-1!"

@app.route("/poweron2")
def poweron2():
    button(1, 15)
    return "<p>Power on machine-2!</p>"

@app.route("/poweroff2")
def poweroff2():
    button(4, 15)
    return "<p>Power off machine-2!</p>"""

@app.route("/reboot2")
def reboot2():
    button(4, 15)
    time.sleep(1)
    button(1, 15)
    return "Rebooted machine-2!"

if __name__ == '__main__':
    button(0, 14)
    button(0, 15)
    app.run(host='0.0.0.0', port=8182)
```

3. 運行程序 `python3 power.py` ，之後便可以通過 `http://IP:8182/operation` 進行開關機操作。
4. 配置服務文件，使得在樹莓派開機時自動運行 `power.py`
    1. 編寫服務文件 `sudo vim /etc/systemd/system/power.service`

    ```
    [Unit]
    Description=power control daemon
    After=network.target

    [Service]
    ExecStart=/usr/bin/python3 /home/ubuntu/test/power.py

    [Install]
    WantedBy=multi-user.target
    ```

    2. 啟動服務

    ```
    $ sudo systemctl enable power.service
    ```

### 9. 配置NAT

為了讓板子能連接外網，需要配置NAT。

1. 安裝 `iptables-persistent`

```
$ sudo apt install iptables-persistent
```

2. 配置NAT

```
$ sudo iptables -t nat -F POSTROUTING
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -s 10.0.1.0/24 -m conntrack --ctstate
NEW -j ACCEPT
$ sudo iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

3. 保存iptables，確保每次開機後都能加載NAT規則

```
$ sudo su
$ iptables-save > /etc/iptables/rules.v4
```

4. 開啟路由轉發，編輯`sudo vim /etc/sysctl.conf`文件

```
#net.ipv4.ip_forward=1 --> net.ipv4.ip_forward=1
```

5. 啟用服務

```
$ sudo sysctl -p
```

### 10. 配置Docker

根據[官網](Install Docker Engine on Ubuntu | Docker Documentation)上的教程來進行安裝

**設置倉庫**

1. 卸載舊板本

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. 設置倉庫

```
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

3. 添加官方GPG Key

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

4. 設置穩定版倉庫

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-
archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null*
```

安裝Docker Engine

1. 利用apt進行安裝

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. 測試Docker是否安裝成功，打印以下信息即為安裝成功

```
$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:10d7d58d5ebd2a652f4d93fdd86da8f265f5318c6a73cc5b6a9798ff6d2b2e67
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

3. 安裝Docker-compose

```
$ sudo apt install docker-compose
```

4. 安裝Portainer 2.0 Web GUI，之後打開http://IP:9000，便可以看到Docker管理

```
$ docker pull portainer/portainer-ce:linux-arm64
$ docker run -d -p 9000:9000 --restart unless-stopped --name portainer -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer-ce:linux-arm64 (進入網址後，可以修改容器的名稱為"protainer"，方
便之後直接啟動相同容器)
$ docker start portainer
```

5. 給予Docker權限

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

**編寫Dockerfile**

1. 編寫一個能夠連接串口的最低配置的Dockerfile

```
$ sudo mkdir docker
$ cd docker
$ vim Dockerfile
```

Dockerfile

```
FROM ubuntu:20.04

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update \
    && apt-get install -y iputils-ping \
    && apt-get install -y net-tools \
    && apt-get install -y picocom \
    && apt-get install -y vim \
    && apt-get install -y tftp-hpa \
    # for SSH
    && apt-get install -y openssh-server \
    && apt-get install -y openssh-client \
    && apt-get install -y ssh

CMD bash
```

2. 生成鏡像。

```
$ docker build -t ubuntu:custom .
Step 1/4 : FROM ubuntu:20.04
 ---> e9af1a920203
Step 2/4 : ARG DEBIAN_FRONTEND=noninteractive
 ---> Using cache
 ---> fca4a6a1e40d
Step 3/4 : RUN apt-get update     && apt-get install sudo     && apt-get install -
y iputils-ping     && apt-get install -y net-tools     && apt-get install -y
picocom     && apt-get install -y kmod     && apt-get install -y vim     && apt-
get install -y openssh-server     && apt-get install -y openssh-client     && apt-
get install -y ssh
...
...
Step 4/4 : CMD bash
 ---> Running in 0a9505ef0a64
Removing intermediate container 0a9505ef0a64
 ---> ee714030dec7
Successfully built ee714030dec7
Successfully tagged ubuntu:custom
```

**Docker遠程訪問**

**目前如果利用Docker訪問的話，需要停掉系統的usbip服務，直接把USB掛載到Docker上。**

配置一個自定義網絡，方便管理Docker的IP

1. 創建自定義網絡，並且指定網段：172.18.0.0/16

```
$ docker network create --subnet=172.18.0.0/16 testnet
$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
ca1cf709e0e0   bridge    bridge    local
3f1861d49981   host      host      local
0760c0a8c058   none      null      local
ba81bee543ca   testnet   bridge    local
```

2. 修改Dockerfile，並重新創建鏡像

```
FROM ubuntu:20.04

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update \
    && apt-get install -y iputils-ping \
    && apt-get install -y net-tools \
    && apt-get install -y picocom \
    && apt-get install -y vim \
    && apt-get install -y tftp-hpa \
    # for SSH
    && apt-get install -y openssh-server \
    && apt-get install -y openssh-client \
    && apt-get install -y ssh

RUN echo "root:root" | chpasswd

COPY startup_run.sh /root/startup_run.sh

RUN echo "# startup run" >> /root/.bashrc
RUN echo "if [ -f /root/startup_run.sh ]; then" >> /root/.bashrc
RUN echo "      /root/startup_run.sh" >> /root/.bashrc
RUN echo "fi" >> /root/.bashrc

RUN /bin/bash -c 'chmod +x /root/startup_run.sh'

RUN sed -i "s/#Port 22/Port 10008/g" /etc/ssh/sshd_config
RUN sed -i "s/#PermitRootLogin prohibit-password/PermitRootLogin yes/g"
/etc/ssh/sshd_config
RUN sed -i "s/UsePAM yes/UsePAM no/g" /etc/ssh/sshd_config

EXPOSE 10008

CMD bash
```

其中startup_run.sh文件，需要 `chmod +x /root/startup_run.sh`

```
#!/bin/bash

LOGTIME=$(date "+%Y-%m-%d %H:%M:%S")
echo "[$LOGTIME] startup run..." >>/root/startup_run.log
service ssh start >>/root/startup_run.log
```

3. 創建容器

```
$ docker run -itd -w /workspace -v /home/test1:/workspace -p 10008:10008 --net
testnet --ip 172.18.0.2 --name test1 --device=/dev/ttyUSB1 ubuntu:custom bash
```

指令解釋

-itd：讓容器在後台執行，並有可交互的終端

-w：工作目錄

-v：把樹莓派上的目錄掛載到容器工作目錄上，可以方便用戶在容器上建立自己的目錄同時會儲存在樹莓派上

-p：指定port

--net：所使用的網絡

--ip：指定該容器所使用的IP，方便ssh訪問和管理

--name：指定容器名稱，方便之後的使用

--device：把串口掛載到容器上

ubuntu:custom：鏡像名稱

4. 啟動容器

```
$ docker start test1
```

5. 可以通過ssh遠程訪問容器

```
$ ssh -J ubuntu@183.173.21.114 root@172.18.0.2 -p 10008
root@172.18.0.2's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-1059-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Apr 27 14:41:25 2022 from 172.18.0.1
root@84e3b56eec59:~#
```

### 11. 更改時區

1. 運行tzselect

   ```
   $ tzselect
   ```

2. 依次選擇Asia, China, Bejing

3. 複製文件

   ```
   $ sudo cp /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime
   ```

4. 查看時間

   ```
   $ date -R
   ```

### 12. 檢查Docker是否仍有SSH連接

為更好管理機器使用，可以定時檢查機器是否仍有人使用。

1. 安裝APSechduler庫，用於排程工作

   ```
   $ pip3 install APScheduler
   ```

2. 配合MySQL，編寫檢查腳本status.py與check.py

   status.py

```
from time import timezone
from apscheduler.schedulers.blocking import BlockingScheduler
from datetime import datetime
from time import localtime
import database
import docker_manage
from pathlib import Path
import check

# 服務器啟動時間
server_start_time = 9

# 服務器定期更新

# 每整點更新一次，用於Docker的更新
server_update_time_week = '0-6' # 0-6 代表 週一到週日工作
server_update_time_hour = '10-21' # 10-21 代表 10點開始第一次更新，21點最後一次更新
server_update_time_min = '0' # 代表 XX:00:XX 更新
server_update_time_sec = '1' # 代表 XX:XX:01 更新

# 每整點的55分檢查一次，用於提前5分鐘斷線的提示
server_check_time_week = '0-6'
server_check_time_hour = '10-21'
server_check_time_min = '55' # 代表 XX:55:XX 更新
server_check_time_sec = '1'

t = localtime()

log_name = str(t.tm_year) + '_' + str(t.tm_mon) + '_' + str(t.tm_mday) + '.log'

# 到時前五分鐘提示
def expired_remind():
    now_docker_names = database.get_status()[0]
    now_docker_names = [x for x in now_docker_names if x is not None]

    if now_docker_names:
        for now_docker_name in now_docker_names:
            check.expired_remind(now_docker_name)

# 更新Docker
def update_machine():
    t = localtime()
    period = t.tm_hour - 9

    # 第一個時間段不用特殊判斷
    if period == 1:
        docker_names = database.get_timetable(period)[0]
        machine_no = 1
        docker_names = [x for x in docker_names if x is not None]
```

```python
        # 如果預約時間表不為空，即有人預約
        if docker_names:
            for docker_name in docker_names:
                if docker_name is not None:
                    output = docker_manage.start_docker(docker_name)

                    t = localtime()

                    # 記錄Docker啟動時間
                    with open("./log/" + log_name, "a") as f:
                        f.write('Start docker ' + output + ' at ' + str(t.tm_hour)
 + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                    # 更新當前機器使用狀態
                    database.update_status(period, machine_no, docker_name)
                    machine_no += 1

    # 之後的時間段，需要判斷 下個時間段有否預約，是否為同一人預約，是否使用同一機器
    else:
        # 獲取下個時間段使用信息
        next_docker_names = database.get_timetable(period)[0]
        machine_no = 1

        # 判斷下個時間段是否有人使用，即時間表中是否為 'NULL'
        next_docker_names = [x for x in next_docker_names if x is not None]

        # 下個時間段有人預約
        if next_docker_names:
            now_docker_names = database.get_status()[0]
            now_docker_names = [x for x in now_docker_names if x is not None]

            # 如果目前有機器有分配用戶
            # 則檢查下個時間段該用戶是否使用同一機器
            # 若為同一機器則不中斷SSH連線，若不為同一機器則中斷SSH連線(停止 Docker)
            if now_docker_names:
                for now_docker_name in now_docker_names:
                    # 不為同一機器
                    if now_docker_name not in next_docker_names:
                        user_name = now_docker_name[0:len(now_docker_name)-2]

                        path = 'log/' + user_name
                        Path(path).mkdir(parents=True, exist_ok=True)

                        t = localtime()

                        # 停止Docker
                        output = docker_manage.stop_docker(now_docker_name)

                        # 記錄Docker停止時間
                        with open("./log/" + log_name, "a") as f:
                            f.write('Stop  docker ' + output + ' at ' +
str(t.tm_hour) + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                        # 保存用戶操作記錄
                        check.save_user_log(now_docker_name)

            # 根據預約時間表啟動下個時間段的Docker
            for next_docker_name in next_docker_names:
                if next_docker_name is not None:
                    output = docker_manage.start_docker(next_docker_name)

                    t = localtime()
                    with open("./log/" + log_name, "a") as f:
                        f.write('Start docker ' + output + ' at ' + str(t.tm_hour)
+ ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                    # 更新當前機器使用狀態
                    database.update_status(period, machine_no, next_docker_name)
                    machine_no += 1

        # 下個時間段沒有預約
        else:
            now_docker_names = database.get_status()[0]
            now_docker_names = [x for x in now_docker_names if x is not None]

            # 檢查當前分配的用戶是否正在使用 <-- 檢查SSH連線情況，如果SSH用戶為0，則
Docker沒人使用
            if now_docker_names:
                for now_docker_name in now_docker_names:
                    ssh_usr = check.check_alive(now_docker_name)

                    if ssh_usr == '0':
                        user_name = now_docker_name[0:len(now_docker_name)-2]

                        path = 'log/' + user_name
                        Path(path).mkdir(parents=True, exist_ok=True)

                        t = localtime()

                        # 停止Docker
                        output = docker_manage.stop_docker(now_docker_name)

                        with open("./log/" + log_name, "a") as f:
                            f.write('Stop  docker ' + output + ' at ' +
str(t.tm_hour) + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                        # 保存用戶操作記錄
                        check.save_user_log(now_docker_name)

                        # 更新當前機器使用狀態
                        database.update_status(period, machine_no, 'NULL')
                    machine_no += 1

if __name__ == '__main__':

    # 設定時間
    scheduler = BlockingScheduler(timezone="Asia/Shanghai")

    while True:
        t = localtime()

        # 當時間在服務器工作時間時啟動服務
        if t.tm_hour >= server_start_time:

            # 添加更新時間表排程服務
            scheduler.add_job(update_machine, 'cron',
                                              day_of_week=server_update_time_week,
                                              hour=server_update_time_hour,
                                              minute=server_update_time_min,
                                              second=server_update_time_sec
                                              )

            # 添加提示結束排程服務
            scheduler.add_job(expired_remind, 'cron',
                                              day_of_week=server_check_time_week,
                                              hour=server_check_time_hour,
                                              minute=server_check_time_min,
                                              second=server_check_time_sec
                                              )

            # 開啟排程服務
            scheduler.start()
```

check.py

```
import docker

# 返回Docker SSH連線數
def check_alive(docker_name):

    client = docker.from_env()

    container = client.containers.get(docker_name)
    output = container.exec_run(cmd='w | grep pts')
    output = str(output[1]).split(',')

    num = output[1].replace(' ', "")

    return num[0]

# 預約到時提醒
def expired_remind(docker_name):

    client = docker.from_env()

    container = client.containers.get(docker_name)
    output = container.exec_run(cmd="wall /etc/remind")

# 保存用戶操作記錄到主機
def save_user_log(docker_name):

    client = docker.from_env()
    usr_name = docker_name[0:len(docker_name)-2]
    docker_command = 'docker cp ' + docker_name + ':/etc/session.log
/docker_usr/info/' + usr_name + '/log/'
    subprocess.check_output([docker_command], shell=True)
```

## 13. 利用Python腳本自動生成Docker

1. 編寫docker_manage.py

```python
from flask import *
import time
import subprocess
import database
import time
import docker

# 機器數目
machine_num = 3

# 啟動Docker
def start_docker(usr_name):
    output = subprocess.check_output(["docker start " + str(usr_name)],
shell=True)
    output = str(output)
    output = output.replace('b\'', "")
    output = output.replace('\\n\'', "")
    return output

# 停止Docker
def stop_docker(usr_name):
    output = subprocess.check_output(["docker stop " + str(usr_name)], shell=True)
    output = str(output)
    output = output.replace('b\'', "")
    output = output.replace('\\n\'', "")
    return output

# 新建Docker
def make_docker(usr_name):
    user_num = database.check_size() - 1
    outputs=[]

    for id in range(machine_num):
        # Port從20000開始，每個用戶新增三個Docker，分配三個Port
        port = 20000 + 3 * user_num + id
        docker_command ="docker run -itd" + \
                        " -w /root/" + usr_name + \
                        " -v /docker_usr/user/" + usr_name + ":/root/" + usr_name
+ \
                        " -v /docker_usr/info/" + usr_name + "/ssh_keys/" +
":/root/.ssh" + \
                        " -p " + str(port) + \
                        ":10008" + \
                        " --net testnet" + \
                        " --name " + usr_name + "_" + str(id + 1) + \
                        " --device=/dev/ttyUSB" + str(id * 2 + 1) + \
                        " ubuntu:u740 bash"
        subprocess.check_output([docker_command], shell=True)
        time.sleep(1)
        docker_command ="docker stop " + usr_name + "_" + str(id + 1)
```

```
            output = subprocess.check_output([docker_command], shell=True)
            output = str(output)
            output = output.replace('b\'', "")
            output = output.replace('\\n\'', "")
            outputs.append(output)

        return outputs
```

### 14. 記錄用戶操作記錄

記錄用戶在Docker內的操作以及輸出記錄，利用系統內置的 `Script` 來實現此功能。

1. 新增 `.profile` 文件，用以替換Docker內的文件。

   `addate()` 函數為script記錄增加時間戳。

   `-fq` 參數使Script能夠刷新緩衝區以及以靜默狀態執行。

   `&& exit` 因為在執行script的情況下使用exit退出SSH的話會先退出script，因此多添加一個exit，使用戶在輸入exit能夠直接退出SSH不是只退出script。

```
# ~/.profile: executed by Bourne-compatible login shells.

if [ "$BASH" ]; then
  if [ -f ~/.bashrc ]; then
    . ~/.bashrc
  fi
fi

mesg n 2> /dev/null || true

adddate() {
    while IFS= read -r line; do
        printf '%s %s\n' "$(date)" "$line";
    done
}

/usr/bin/script -fq >( adddate >> /etc/session.log) && exit
```

### 15. 更新Dockerfile以及增加相關文件

通過不斷的更新後，需要更改Dockerfile來配合。

1. 修改Dockerfile

```
FROM ubuntu:20.04

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update \
    && apt-get install -y iputils-ping \
    && apt-get install -y net-tools \
    && apt-get install -y picocom \
    && apt-get install -y vim \
    && apt-get install -y tftp-hpa \
    # for SSH
    && apt-get install -y openssh-server \
    && apt-get install -y openssh-client \
    && apt-get install -y ssh

# 開機啟動SSH

RUN mkdir -p ~/.ssh

COPY startup_run.sh /etc/startup_run.sh
RUN /bin/bash -c 'chmod +x /etc/startup_run.sh'

RUN echo "# startup run" >> /root/.bashrc
RUN echo "if [ -f /etc/startup_run.sh ]; then" >> /root/.bashrc
RUN echo "      /etc/startup_run.sh" >> /root/.bashrc
RUN echo "fi" >> /root/.bashrc

RUN sed -i "s/#Port 22/Port 10008/g" /etc/ssh/sshd_config
RUN sed -i "s/#PermitRootLogin prohibit-password/PermitRootLogin yes/g"
/etc/ssh/sshd_config
RUN sed -i "s/#PasswordAuthentication yes/PasswordAuthentication no/g"
/etc/ssh/sshd_config
RUN sed -i "s/UsePAM yes/UsePAM no/g" /etc/ssh/sshd_config

RUN /bin/bash -c 'chmod -R go= /root/.ssh'
RUN /bin/bash -c 'chown -R root /root/.ssh'

# 結束提示語句
COPY remind /etc/remind
RUN /bin/bash -c 'chmod +x /etc/remind'

# 記錄用戶操作記錄配置
COPY .profile /root/.profile
RUN /bin/bash -c 'chmod +x /root/.profile'

#Expose Port
EXPOSE 10008

CMD bashh'
```

```
#Expose Port
EXPOSE 10008

CMD bash
```

2. remind文件

   到時提示語句

   ```
   :: SSH will shutdown in 5 minutes if you do not apply the same machine in next
   period, remember to save your files in time ::
   ```

3. config文件

   防止SSH閒置過久後自動斷線

   ```
   Host *
    ServerAliveInterval 60
    ServerAliveCountMax 3
   ```

### 16. 內嵌頁面自啟與Moodle網址動態更新

由於校園網每天IP都有可能不一樣，因此需要動態修改Moodle的訪問地址。修改ip.sh，在裏面加入啟動項。

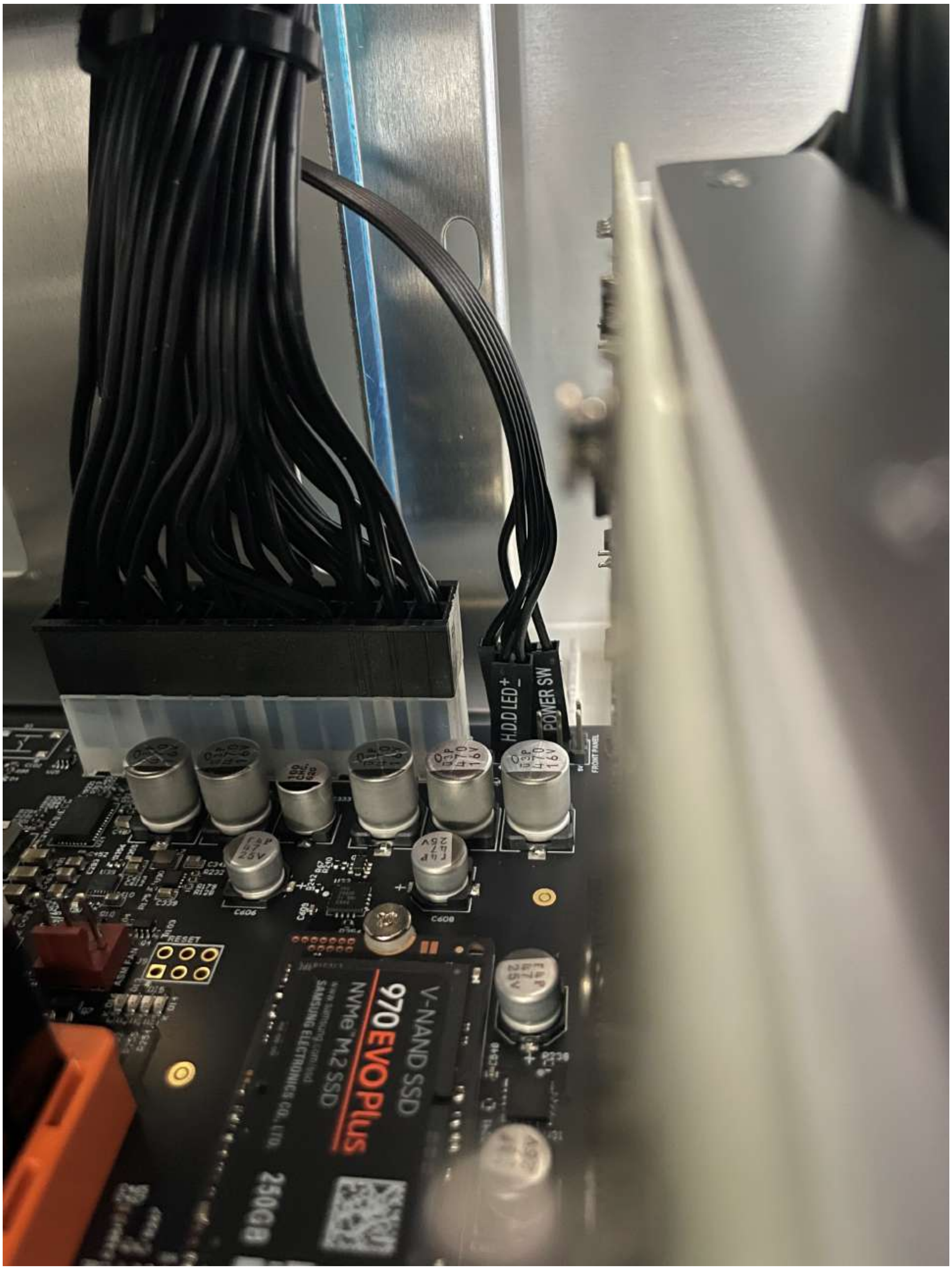
```
sudo sed -i "21c \$CFG->wwwroot   = 'http://$wlan0:8888/moodle';"
/var/www/html/moodle/config.php

/usr/bin/python3 /home/ubuntu/applysystem/moodle.py &
```

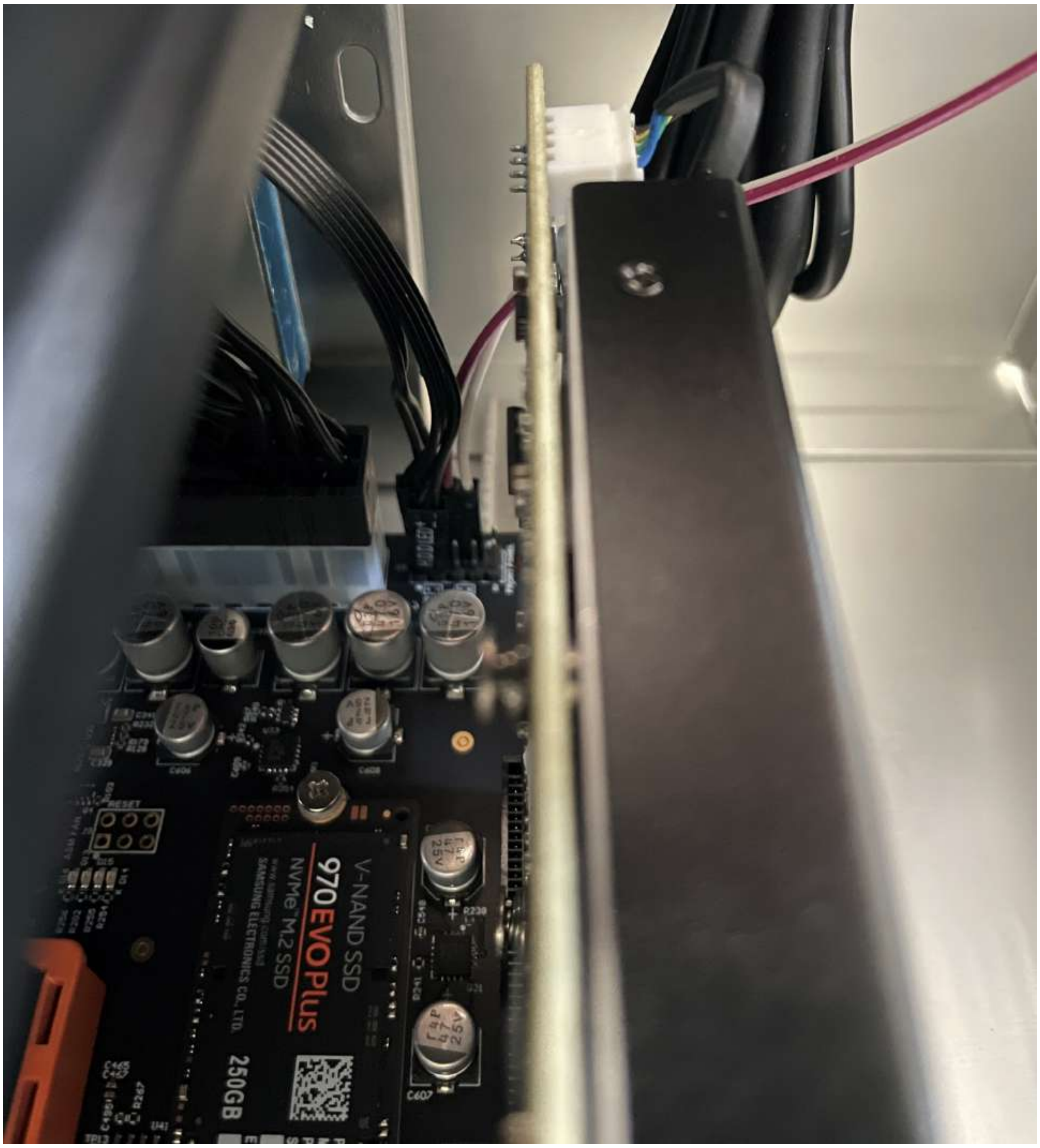
## FU740

### 1. 連接樹莓派與FU740

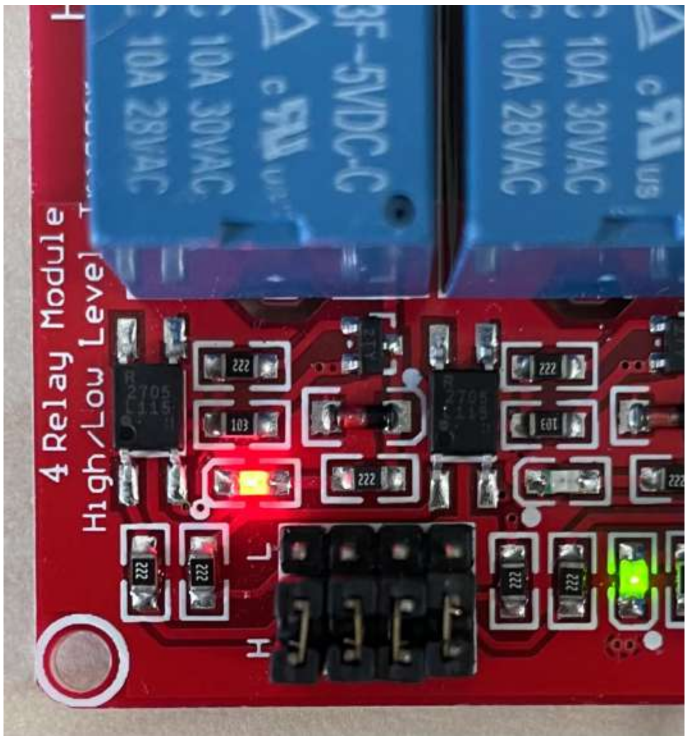
1. 利用杜邦線物理連接樹莓派與FU740

   1. 把原本版子上的 `POWER SW` 線換成自己的杜邦線。


H.D.D LED+
POWER SW
FRONT PANEL
RESET
V-NAND SSD
970 EVO Plus
NVMe M.2 SSD
250GB

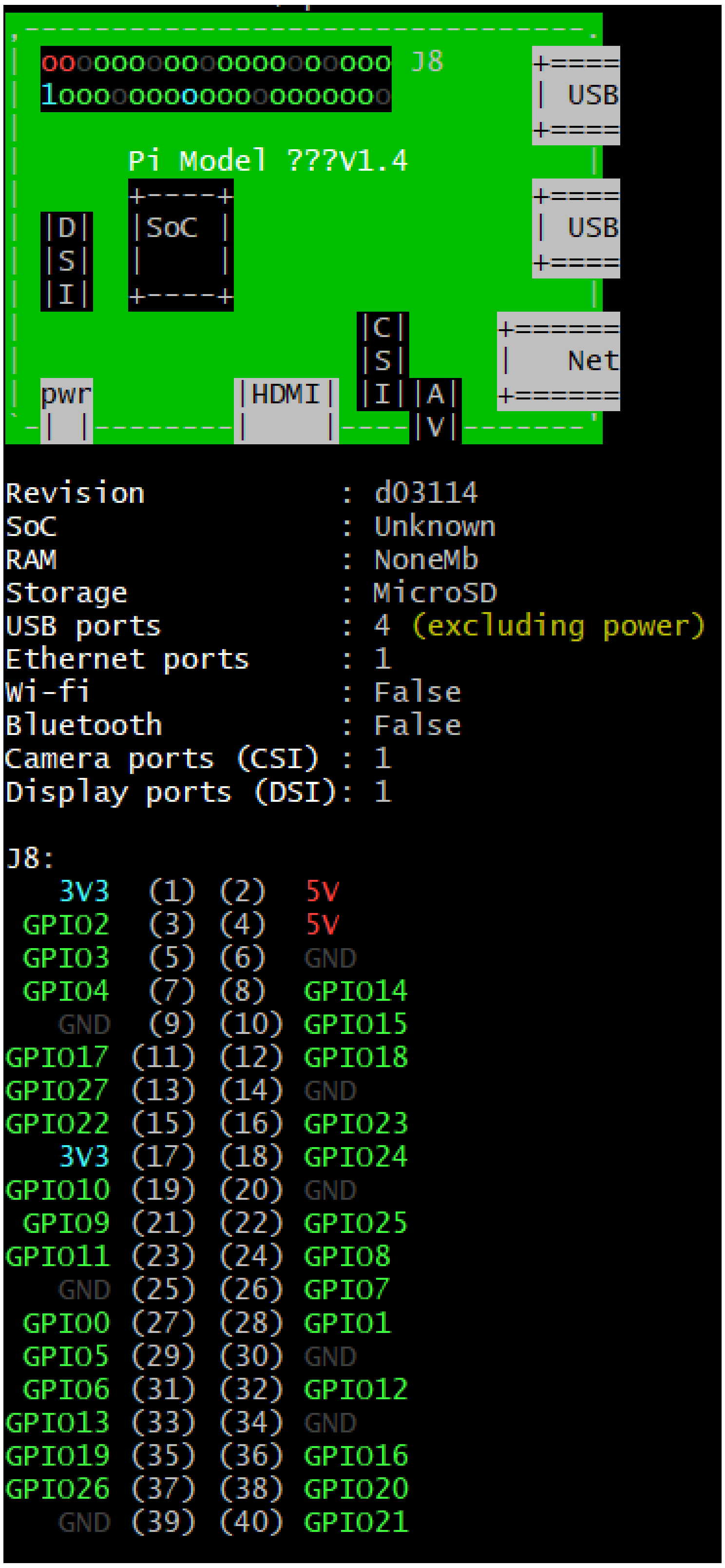


2. 把繼電器全部切換成高電平觸發

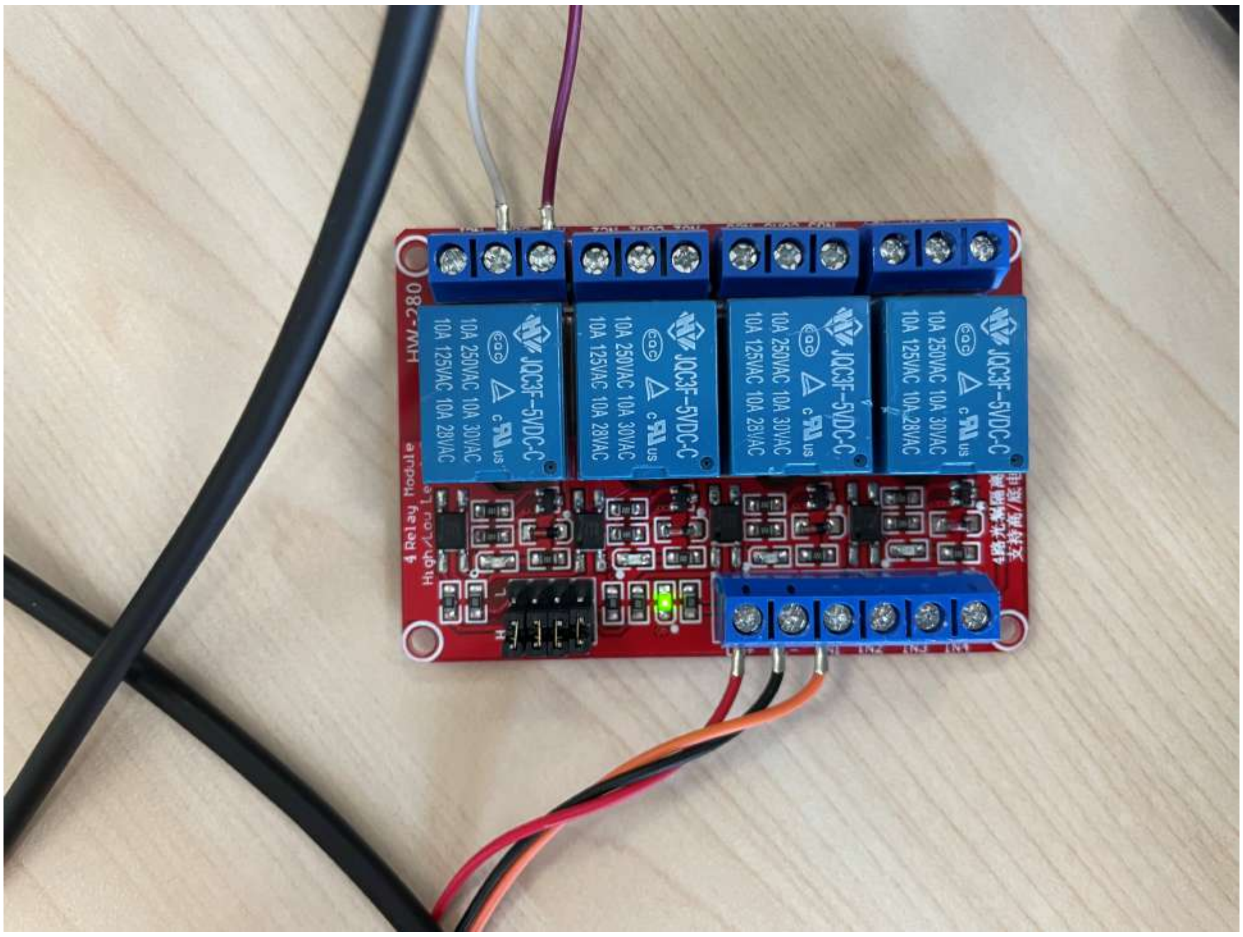
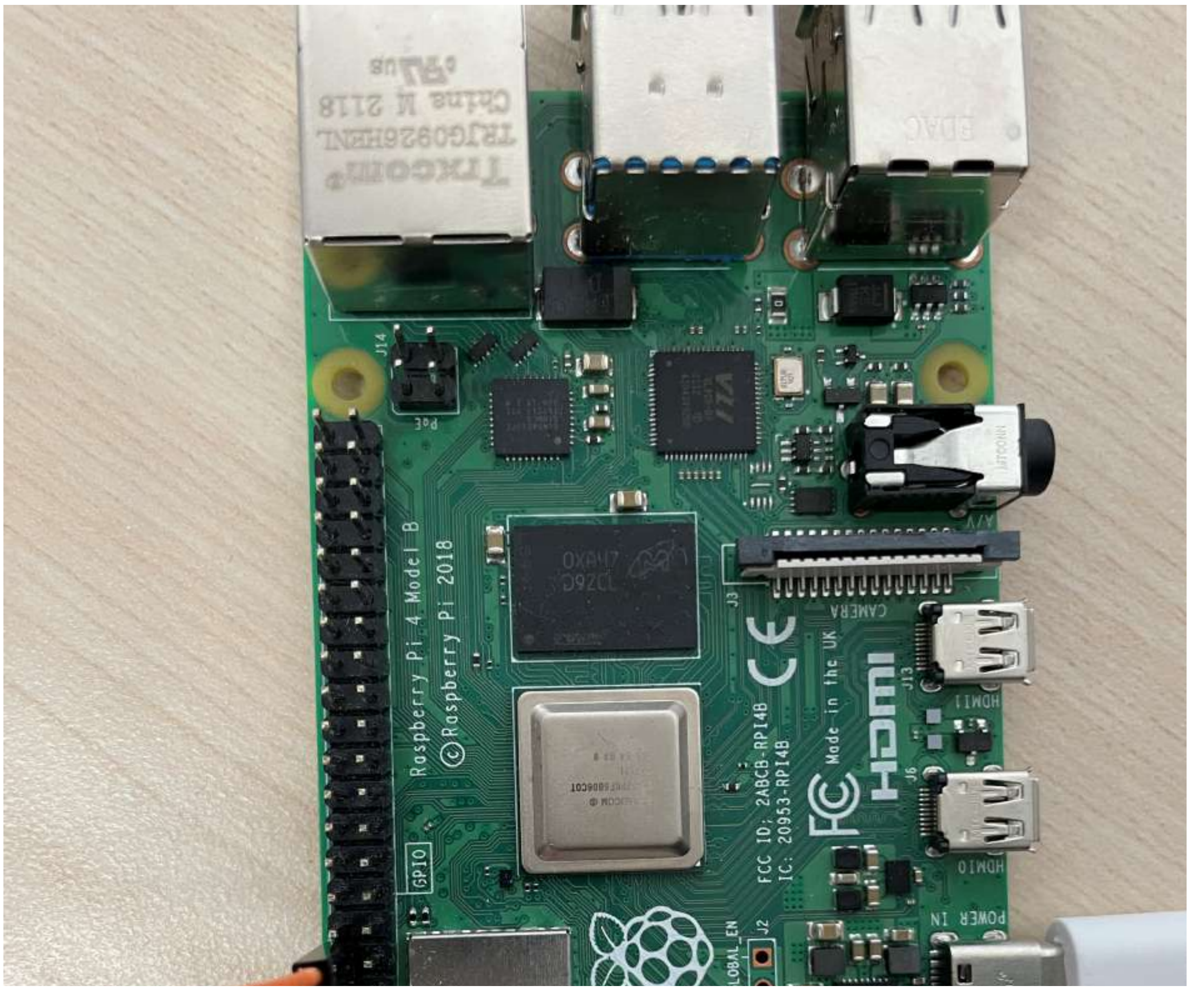


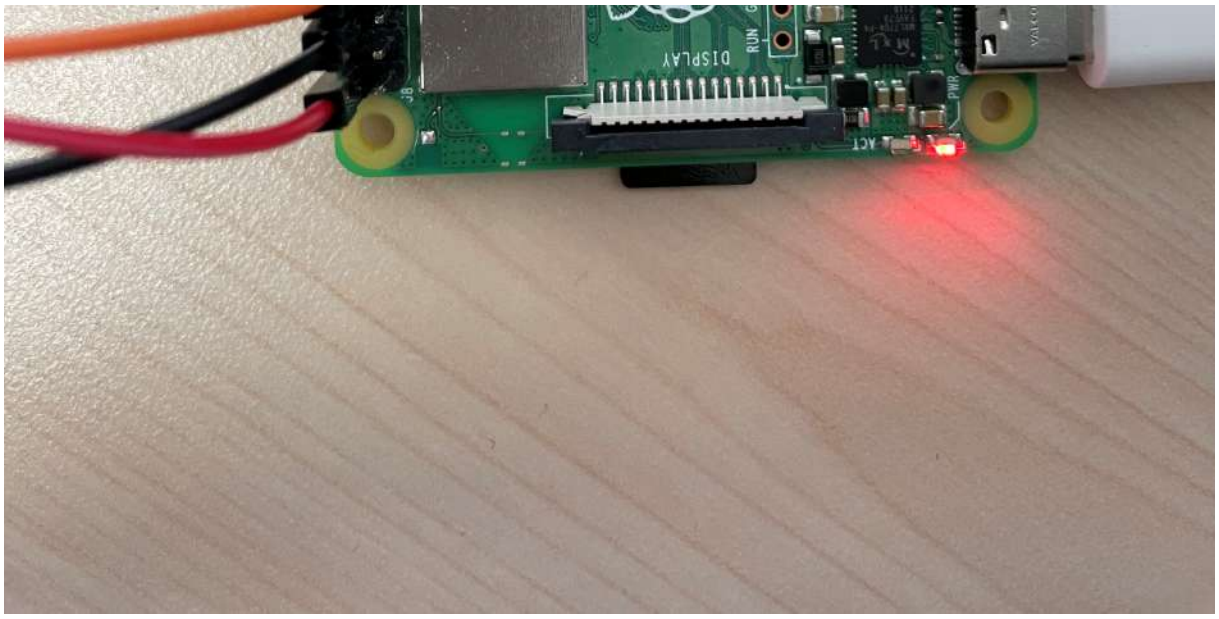
3. 將杜邦線連接到繼電器上。（切記不要弄反，紫色接到NO1，白色接到COM1）
4. 將樹莓派上的5V，GND以及訊號線(GPIO)接到繼電哭上的DC+，DC-，IN1上。接好後應如下圖所示。

   可以利用`pinout`來查看樹莓派上的對應接口。這裏接GPIO14。



```
,--------------------------------.
| oooooooooooooooooooo J8     +====
| 1ooooooooooooooooooo        | USB
|                             +====
|      Pi Model ???V1.4          |
|      +----+                 +====
| |D|  |SoC |                 | USB
| |S|  |    |                 +====
| |I|  +----+                    |
|                   |C|     +======
|                   |S|     |   Net
| pwr        |HDMI| |I||A|  +======
`-| |--------|    |----|V|-------'

Revision           : d03114
SoC                : Unknown
RAM                : NoneMb
Storage            : MicroSD
USB ports          : 4 (excluding power)
Ethernet ports     : 1
Wi-fi              : False
Bluetooth          : False
Camera ports (CSI) : 1
Display ports (DSI): 1

J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21
```



5. 接上FU740電源，進行下一步調試。

2. 在樹莓派上安裝python GPIO庫，`sudo apt-get install python3-dev python3-rpi.gpio`，撰寫GPIO測試程序`power.py`。

   給予python權限，`sudo chmod 4775 /usr/bin/python`，`sudo chmod 4775 /usr/bin/python3`

   - 1秒是開機，4秒是強制關機。

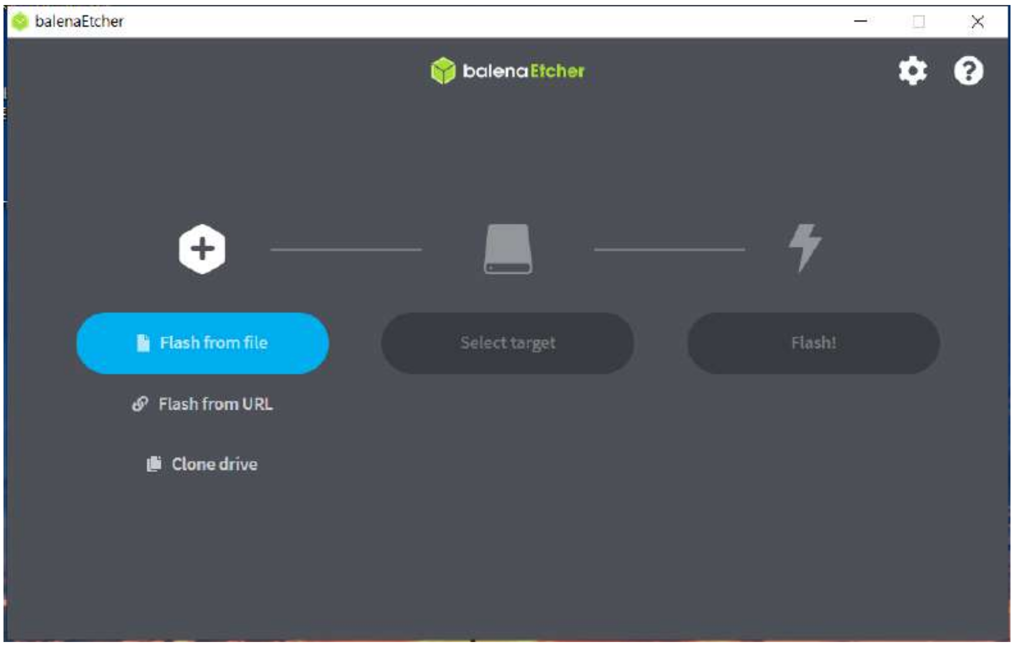
   ```
   import gpiod
   import time

   def button(secs, line):
       with gpiod.Chip("gpiochip0") as chip:
           line = chip.get_line(line)
           line.request(consumer="powerd", type=gpiod.LINE_REQ_DIR_OUT)
           line.set_value(1)
           time.sleep(secs)
           line.set_value(0)

   if __name__ == '__main__':
       button(1, 14)
   ```

   運行語句`python3 power.py`，運行後應該可以聽到十分清脆的繼電器啟動聲音以及十分響亮的風扇聲。這便說明成功利用樹莓派控制板子啟動了。

## 2. 安裝U-Boot

1. 在SD卡上寫入[U-boot](Release HiFive Unmatched SD Image & U-Boot SPI Installer · zhaofengli/nixos-riscv64 (github.com))

2. 利用[Etcher](balenaEtcher - Flash OS images to SD cards & USB drives)把鏡像燒錄到SD卡上



3. 連接樹莓派與FU740終端，然後查看是否已經接上

```
$ lsusb
```

如正確接入，應出下類似下面的訊息

```
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 0403:6010 Future Technology Devices International, Ltd
FT2232C/D/H Dual UART/FIFO IC
Bus 001 Device 005: ID 0403:6010 Future Technology Devices International, Ltd
FT2232C/D/H Dual UART/FIFO IC
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

其中的 Device 004 和 Device 005 便是接上的數據線。

4. 查看設備會話接口號

```
$ ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1
```

可以看到兩條數據線的名稱分別為 ttyUSB0 和 ttyUBS1。

5. 利用picocom連接到串口，然後進行串口調試

```
$ sudo picocom -b 115200 /dev/ttyUSB0

picocom v3.1

port is         : /dev/ttyUSB0
flowcontrol     : none
baudrate is     : 115200
parity is       : none
databits are    : 8
stopbits are    : 1
escape is       : C-a
local echo is   : no
noinit is       : no
noreset is      : no
hangup is       : no
nolock is       : no
send_cmd is     : sz -vv
receive_cmd is  : rz -vv -E
imap is         :
omap is         :
emap is         : crcrlf,delbs,
logfile is      : none
initstring      : none
exit_after is   : not set
exit is         : no

Type [C-a] [C-h] to see available commands
Terminal ready
```

若正常，會出現Terminal ready提示語句。

6. 把SD卡插回FU740上，訪問網站指令開機。

7. 應該會在命令行上看到如下輸出

```
...
Device 0: Vendor: 0x144d Rev: 2B2QEXM7 Prod: S4EUNX0R802232Z
            Type: Hard Disk
            Capacity: 238475.1 MB = 232.8 GB (488397168 x 512)
... is now current device
** No partition table - nvme 0 **
Couldn't find partition nvme 0:1
starting USB...
Bus xhci_pci: Register 4000840 NbrPorts 4
Starting the controller
USB XHCI 1.00
scanning bus xhci_pci for devices... 3 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found

Device 0: unknown device
switch to partitions #0, OK
mmc0 is current device
Scanning mmc 0:3...
Found U-Boot script /boot.scr
2236 bytes read in 4 ms (545.9 KiB/s)
## Executing script at 88100000

Firmware installer

Installing U-Boot to the SPI flash in 10 seconds!
devtype = mmc
devnum = 0
bootpart = 3

:: Starting flash operation

-> Initializing SPI Flash subsystem...
SF: Detected is25wp256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB

-> Reading new firmware from storage...
6291456 bytes read in 5423 ms (1.1 MiB/s)

-> Writing new firmware to SPI...
device 0 offset 0x0, size 0x600000
2691072 bytes written, 3600384 bytes skipped in 47.308s, speed 136770 B/s

✅ Flashing seems to have been successful!
You can now set MSEL[3:0] = 0110

Resetting in 5 seconds
resetting ...
System reset not supported on this platform
### ERROR ### Please RESET the board ###
```

8. 安裝完後把板子上的 MSEL碼 撥成(MSEL[3:0]=0110)，在下一次啟動時便不需要用到SD卡了

```
--------------- ON
|  | | o o |  |
|  o o | | o  |
--------------- OFF
```

9. 重新啟動機器，應該會出現如下輸出

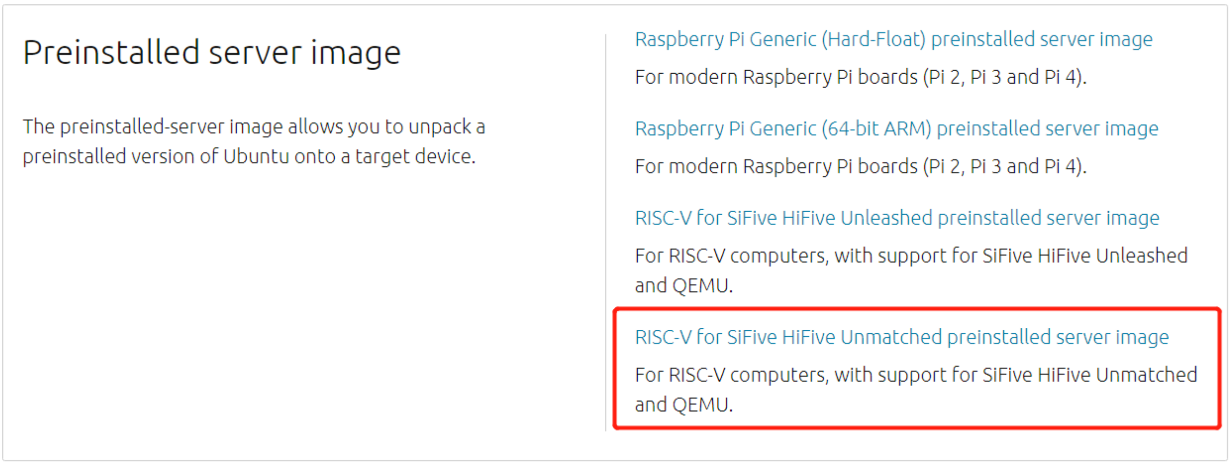
```
U-Boot SPL 2022.01 (Jan 01 1980 - 00:00:00 +0000)
Trying to boot from SPI

U-Boot 2022.01 (Jan 01 1980 - 00:00:00 +0000)

CPU:   rv64imafdc
Model: SiFive HiFive Unmatched A00
DRAM:  16 GiB
MMC:   spi@10050000:mmc@0: 0
Loading Environment from SPIFlash... SF: Detected is25wp256 with page size 256
Bytes, erase size 4 KiB, total 32 MiB
*** Warning - bad CRC, using default environment

EEPROM: SiFive PCB EEPROM format v1
Product ID: 0002 (HiFive Unmatched)
PCB revision: 3
BOM revision: B
BOM variant: 0
Serial number: SF105SZ212200785
Ethernet MAC address: 70:b3:d5:92:f9:e1
CRC: a2ae439d
In:    serial@10010000
Out:   serial@10010000
Err:   serial@10010000
Model: SiFive HiFive Unmatched A00
Net:   eth0: ethernet@10090000
Hit any key to stop autoboot:  0
PCIE-0: Link up (Gen1-x8, Bus0)

Device 0: Vendor: 0x144d Rev: 2B2QEXM7 Prod: S4EUNX0R802232Z
            Type: Hard Disk
            Capacity: 238475.1 MB = 232.8 GB (488397168 x 512)
... is now current device
** No partition table - nvme 0 **
Couldn't find partition nvme 0:1
starting USB...
Bus xhci_pci: Register 4000840 NbrPorts 4
Starting the controller
USB XHCI 1.00
scanning bus xhci_pci for devices... 3 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found

Device 0: unknown device
scanning bus for devices...

Device 0: unknown device
ethernet@10090000: PHY present at 0
ethernet@10090000: Starting autonegotiation...
ethernet@10090000: Autonegotiation timed out (status=0x7949)
ethernet@10090000: link down (status: 0x7949)
```

```
missing environment variable: pxeuuid
...
=>
```

10. 當出現`=>`時，便可進行下一步操作。

## 3. 安裝系統

1. 先下載一個系統鏡像，可以下載一個preinstalled的，在這裏使用[*Unmatched preinstalled server* image Ubuntu server 20.04](Ubuntu 20.04.4 LTS (Focal Fossa) (u-toyama.ac.jp))為例子。



Preinstalled server image

The preinstalled-server image allows you to unpack a preinstalled version of Ubuntu onto a target device.

Raspberry Pi Generic (Hard-Float) preinstalled server image
For modern Raspberry Pi boards (Pi 2, Pi 3 and Pi 4).

Raspberry Pi Generic (64-bit ARM) preinstalled server image
For modern Raspberry Pi boards (Pi 2, Pi 3 and Pi 4).

RISC-V for SiFive HiFive Unleashed preinstalled server image
For RISC-V computers, with support for SiFive HiFive Unleashed and QEMU.

RISC-V for SiFive HiFive Unmatched preinstalled server image
For RISC-V computers, with support for SiFive HiFive Unmatched and QEMU.

複製下載連結，然後在樹莓派上cd到/srv/tftp，下載鏡像到此文件夾。

```
$ cd /srv/tftp

$ sudo wget http://ubuntutym2.u-toyama.ac.jp/ubuntu-dvd/20.04/release/ubuntu-
20.04.4-preinstalled-server-riscv64+unmatched.img.xz
```

2. 下載後解壓，然後可以把文件改名，比如換成`ubuntu.img`

```
$ sudo xz -d ubuntu-20.04.4-preinstalled-server-riscv64+unmatched.img.xz

$ sudo mv ubuntu-20.04.4-preinstalled-server-riscv64+unmatched.img ubuntu.img
```

3. 查看鏡像的大小

```
$ stat ubuntu.img

File: ubuntu.img
  Size: 3758096384      Blocks: 5719456    IO Block: 4096   regular file
Device: b302h/45826d    Inode: 131263      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2022-04-01 09:14:52.854195481 +0000
Modify: 2022-03-25 04:52:26.000000000 +0000
Change: 2022-04-01 09:21:19.560816976 +0000
 Birth: -
```

記下Blocks的大小。

4. 打開需要安裝的機器，並利用picocom連接串口

```
$ sudo picocom -b 115200 /dev/ttyUSB1
```

5. 在啟動時會出現`Hit any key to stop autoboot:2`提示語句，按任意鍵停止autoboot後出現`=>`

6. 輸入如下語句來安裝系統，其中為上面記錄的Block大小，serverip為樹莓派上eth0的IP

```
=>dhcp
=>setenv serverip 10.0.1.5
=>tftp 0x100000000 ubuntu.img
=>pci enum
=>nvme scan
=>nvme write 0x100000000 0 <block cnt>
#nvme write 0x100000000 0 5719456
```

7. 等待系統安裝

8. 如安裝成功，重啟後應該會出現如下選項，輸入`1`

```
Device 0: Vendor: 0x144d Rev: 2B2QEXM7 Prod: S4EUNC0N929441N
            Type: Hard Disk
            Capacity: 238475.1 MB = 232.8 GB (488397168 x 512)
... is now current device
Scanning nvme 0:1...
Found /boot/extlinux/extlinux.conf
Retrieving file: /boot/extlinux/extlinux.conf
U-Boot menu
1:      Ubuntu 20.04.4 LTS 5.11.0-1029-generic
2:      Ubuntu 20.04.4 LTS 5.11.0-1029-generic (rescue target)
Enter choice:
```

9. 到此便成功安裝系統，可以登入系統進行調試。

## 4. HDMI輸出

1. 安裝所需庫

```
$ sudo apt install build-essential libevent-dev libjpeg-dev libbsd-dev
```

2. 安裝$ustreamer$

```
$ git clone --depth=1 https://github.com/pikvm/ustreamer
$ cd ustreamer
$ make
```

3. 運行$ustreamer$

```
$ ./ustreamer -r 1920x1080 -s :: -p 8080
```

4. 編寫開機服務文件`sudo vim /etc/systemd/system/streamer.service`

```
[Unit]
Description=streamer daemon
After=network.target

[Service]
ExecStart=/home/ubuntu/ustreamer/ustreamer -r 1920x1080 -s :: -p 8080

[Install]
WantedBy=multi-user.target
```

5. 啟用服務

   ```
   $ sudo systemctl enable streamer.service
   ```

6. 之後在瀏覽器中打開

   ```
   http:/IP:8080/stream
   ```

   便可以看到HDMI輸出。

## Moodle

### 1. 安裝Moodle

1. 安裝Apache/MySQL/PHP

   ```
   $ sudo apt install apache2 mysql-client mysql-server php7.4 libapache2-mod-php
   ```

2. 安裝額外軟件

   ```
   $ sudo apt install graphviz aspell ghostscript clamav php7.4-pspell php7.4-curl
   php7.4-gd php7.4-intl php7.4-mysql php7.4-xml php7.4-xmlrpc php7.4-ldap php7.4-zip
   php7.4-soap php7.4-mbstring
   ```

   重啟Apache

   ```
   $ sudo service apache2 restart
   ```

   安裝Git

   ```
   $ sudo apt install git
   ```

3. 下載Moodle

   在opt下載Moodle

   ```
   $ cd /opt
   ```

下載Moodle代碼與目錄

```
$ sudo git clone git://git.moodle.org/moodle.git
```

進入moodle

```
$ cd moodle
```

查看所有版本

```
$ sudo git branch -a
```

選用一個版本，以39_STABLE為例

```
$ sudo git branch --track MOODLE_39_STABLE origin/MOODLE_39_STABLE
$ sudo git checkout MOODLE_39_STABLE
```

4. 複製本地倉庫到 /var/www/html

```
$ sudo cp -R /opt/moodle /var/www/html/
$ sudo mkdir /var/moodledata
$ sudo chown -R www-data /var/moodledata
$ sudo chmod -R 777 /var/moodledata
$ sudo chmod -R 0755 /var/www/html/moodle
```

5. 設定MySQL服務器

```
$ sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
[mysqld]
#
# * Basic Settings
#
user                = mysql
# pid-file          = /var/run/mysqld/mysqld.pid
# socket            = /var/run/mysqld/mysqld.sock
# port              = 3306
# datadir           = /var/lib/mysql
default_storage_engine = innodb #add
innodb_file_per_table = 1 #add
```

重啟MySQL服務器

```
$ sudo service mysql restart
```

登入mysql

```
$ sudo mysql -u root -p
```

創建數據庫

```
mysql> CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

創建用戶

```
mysql> create user 'admin'@'localhost' IDENTIFIED BY 'admin';
```

給予用戶權限

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY
TABLES,DROP,INDEX,ALTER ON moodle.* TO 'admin'@'localhost';
```

退出

```
mysql> quit;
```

6. 登入網頁完成設定

```
http://IP/moodle
```

設定前先開啟讀寫權限

```
$ sudo chmod -R 777 /var/www/html/moodle
```

更改資料目錄

```
/var/moodledata
```

資料庫類型

```
mysqli
```

資料庫設定

```
資料庫主機 localhost
資料庫名稱 moodle
資料庫用戶名稱 admin
資料庫密碼 admin
資料表名稱的前置字元 mdl_
```

設定完後改回權限

```
$ sudo chmod -R 0755 /var/www/html/moodle
```

### 允許外網訪問

1. 修改apache2 port，將Listen 80改成Listen 8888

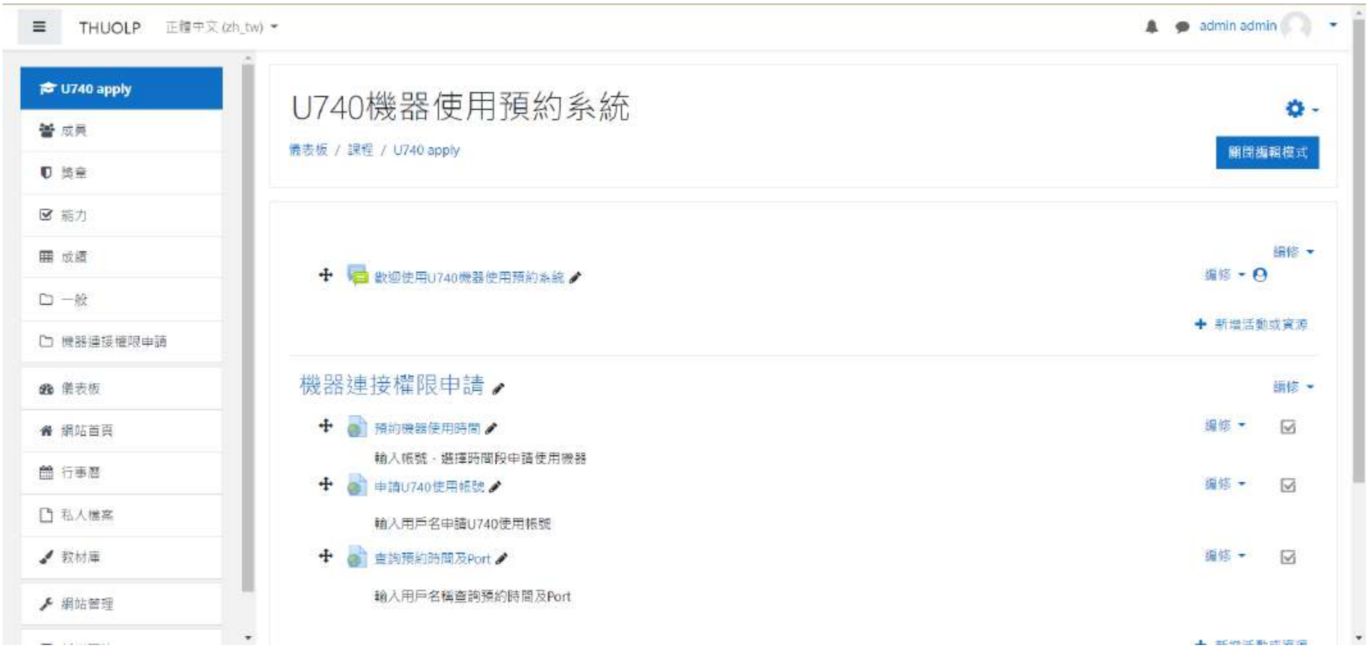
```
$ sudo vim /etc/apache2/ports.conf
#Listen 80
Listen 8888
```

2. 修改moodle的config.php，將http://IP/moodle，改成http://IP:8888/moodle

```
$ sudo vim /var/www/html/moodle/config.php
$CFG->wwwroot   = 'http://IP:8888/moodle';
```

## 2. 配置界面

先登入管理員帳號，點擊新增課程。進入課程後，點擊新活動或資源，添加各標題與內容如下：



輸入帳號，選擇時間段申請使用機器

輸入用戶名申請U740使用帳號

輸入用戶名稱查詢預約時間及Port

分別有三個標題：

1. 申請U740使用帳號
2. 預約機器使用時間
3. 查詢預約時間及Port

具體實現在**配置內嵌頁面**章節。

## 3. 配置數據庫

**配置用來記錄機器記錄預約/用戶docker訊息的實時數據庫。**

1. 利用MySQL創建數據庫

   利用之前創建的用戶登入MySQL，輸入密碼。

   ```
   $ mysql -u admin -p
   ```

2. 選擇數據庫

   ```
   mysql> USE moodle;
   ```

3. 創建新的數據表

### 創建記錄預約情況數據表

```
mysql> CREATE TABLE machine_timetable(
    -> period INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> machine_1 VARCHAR(255),
    -> machine_2 VARCHAR(255),
    -> machine_3 VARCHAR(255)
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

每欄代表： period：時間段 machine_1~3：該時間段該機器被分配的docker name

usrname_1~3：該時間段該機器被分配的username

### 創建記錄用戶Docker信息的數據表

```
mysql> CREATE TABLE docker_info(
   -> usr_id INT PRIMARY KEY NOT NULL,
   -> usr_name VARCHAR(255) NOT NULL,
   -> machine_port INT NOT NULL
   -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

每欄代表：

- usr_id：用戶id，與moodle上的相同，以便核對用戶
- usr_name：用戶申請時自行填寫的用戶名稱
- machine_port：系統生成的Docker Port

### 創建當前時間段機器使用情況的數據表

```
mysql> CREATE TABLE machine_status(
    -> period INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> machine_1 VARCHAR(255),
    -> machine_2 VARCHAR(255),
    -> machine_3 VARCHAR(255),
    -> port_1 INT NOT NULL,
    -> port_2 INT NOT NULL,
    -> port_3 INT NOT NULL
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

每欄代表：

- period：當前時間段

    - machine_13~~：正在使用1~~3號機的Docker名稱
    - port_13~~：正在使用1~~3號機的Docker Port

**編寫Python 查詢/更新腳本**

利用Python配合網站實現實時更新，對數據庫進行更新以及查詢操作。

1. 安裝MySQL驅動庫`MySQL Connector`，用來連接MySQL數據庫

   ```
   $ pip3 install mysql-connector-python
   ```

2. 編寫數據庫更新以及查詢腳本database.py

   database.py

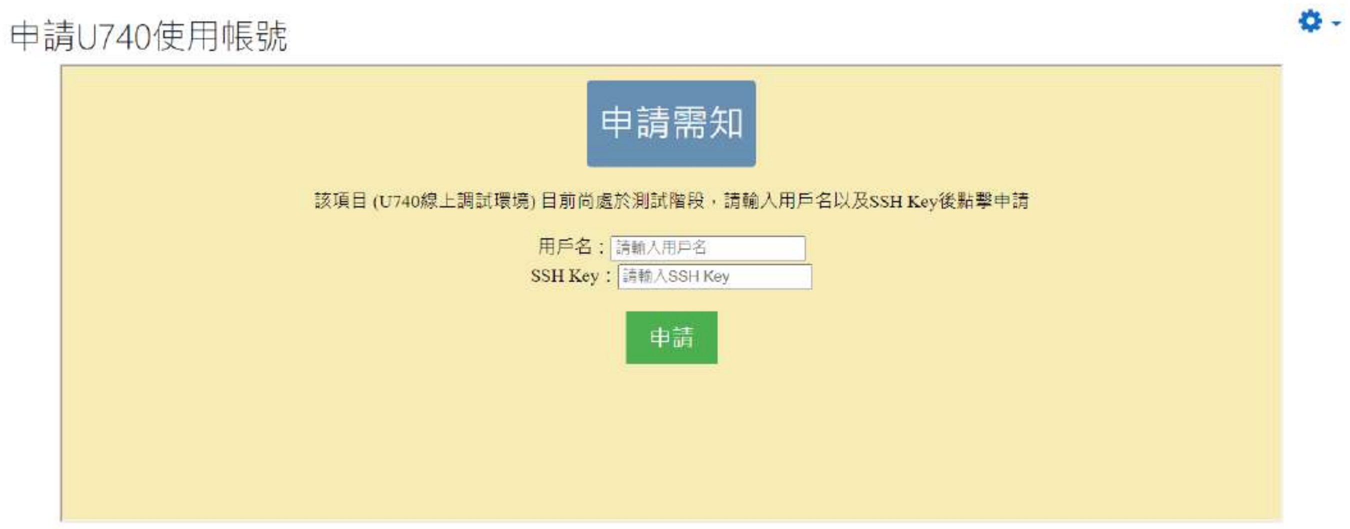
```
import mysql.connector

# 連接數據庫
mydb = mysql.connector.connect(
  host="localhost",
  user="admin",
  passwd="admin",
  database="moodle"
)

mycursor = mydb.cursor()

# 初始新建時間表項
def init():
    sql = "INSERT INTO machine_timetable (machine_1, machine_2, machine_3) VALUES
(%s, %s, %s)"
    val = (None, None, None)
    for i in range(12):
        mycursor.execute(sql, val)
    mydb.commit()

# 每天重置時間表
def init_daily():
    sql = "UPDATE machine_timetable SET machine_1=NULL, machine_2=NULL,
machine_3=NULL"
    mycursor.execute(sql)
    mydb.commit()

    rec=mycursor.rowcount
    sql = "UPDATE machine_status SET machine_1=NULL, machine_2=NULL,
machine_3=NULL, port_1=0, port_2=0, port_3=0"
    mycursor.execute(sql)
    mydb.commit()
    rec = rec + mycursor.rowcount

    print(rec, "record(s) affected")

# 返回用戶是否在數據庫中
def check(usr_name):
    sql = "SELECT * FROM docker_info WHERE usr_name ='" + usr_name + '\''
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    if myresult:
        return myresult[0][1]
    else:
        return None

def check_id(id):
    sql = "SELECT * FROM docker_info WHERE usr_id = " + str(id)
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    if myresult:
        return myresult[0][1]
    else:
        return None

# 返回用戶數量，用於分配Port
def check_size():
    sql = "SELECT * FROM docker_info"
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return len(myresult)

# 新增用戶到數據庫
def add(id, usr_name):
    user_num = check_size()
    port = 20000 + user_num
    sql = "INSERT INTO docker_info (usr_id, usr_name, machine_port) VALUES (%s, 
%s, %s)"
    val = (id, usr_name, port)
    mycursor.execute(sql, val)
    mydb.commit()
    return mycursor.rowcount

# 返回預約時間表空閒機器數量
def check_all_available():
    sql = "SELECT machine_1, machine_2, machine_3 FROM machine_timetable"
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    available = []
    for res in myresult:
        available.append(sum(x is None for x in res))
    return available

# 返回某一時間段的預約時間表空閒機器數量
def check_one_available(period):
    sql = "SELECT machine_1, machine_2, machine_3 FROM machine_timetable WHERE 
period=" + str(period)
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    available = sum(x is not None for x in myresult[0])
    return available

# 返回某一時間段的預約時間表空閒機器數量
def check_available_index(period):
    sql = "SELECT machine_1, machine_2, machine_3 FROM machine_timetable WHERE 
period=" + str(period)
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    available = [x is None for x in myresult[0]]
    machine_no = [i for i, x in enumerate(available) if x]
    #if machine_no:
    return int(machine_no[0]) + 1

# 更新預約時間表
def update_timetable(period, machine_no, docker_name):
    #machine_no = str(int(check_one_available(period)) + 1)
    sql = "UPDATE machine_timetable SET machine_" + str(machine_no) + '= \'' +
str(docker_name) + '\' WHERE period = ' + str(period)
    mycursor.execute(sql)
    mydb.commit()
    return mycursor.rowcount

# 更新當前時間段機器使用狀態
def update_status(period, machine_no, docker_name):

    port = 0

    if docker_name != "NULL":
        usr_name=docker_name #[0:len(docker_name)-2]
        port = get_port(usr_name)
        sql = "UPDATE machine_status SET machine_" + str(machine_no) + "= \'" +
str(docker_name) + "\'"
    else:
        sql = "UPDATE machine_status SET machine_" + str(machine_no) + "= NULL"

    mycursor.execute(sql)
    mydb.commit()

    sql = "UPDATE machine_status SET port_" + str(machine_no) + "= \'" + str(port)
+ "\'"
    mycursor.execute(sql)
    mydb.commit()

    sql = "UPDATE machine_status SET period=\'" + str(period) + "\'"
    mycursor.execute(sql)
    mydb.commit()

    return mycursor.rowcount

# 返回Port
def get_port(usr_name):
    sql = "SELECT machine_port FROM docker_info WHERE usr_name= '" + str(usr_name)
+"\'"
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return myresult[0][0]

# 返回當前機器使用狀態
def get_status():
    sql = "SELECT machine_1, machine_2, machine_3 FROM machine_status"
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return myresult

# 返回某個時間段的預約時間表
def get_timetable(period):
    sql = "SELECT machine_1, machine_2, machine_3 FROM machine_timetable WHERE
period= '" + str(period) + '\''
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return myresult

# 返回某個時間段的一個機器預約時間表
def get_one_timetable(period, machine_no):
    sql = "SELECT machine_" + str(machine_no) + " FROM machine_timetable WHERE
period= '" + str(period) + '\''
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return myresult[0]

# 返回當前機器使用Port
def get_status_port():
    sql = "SELECT port_1, port_2, port_3 FROM machine_status"
    mycursor.execute(sql)
    myresult = mycursor.fetchall()
    return myresult

# 返回用戶使用時間段以及Port
def get_status_port_queries(usr_name):
    result = []
    n_res = []
    queries = []

    for i in range(3):
        sql = "SELECT period, machine_" + str(i + 1) + " FROM machine_timetable
WHERE machine_" + str(i + 1) + "=\'" + str(usr_name) + "\'"
        mycursor.execute(sql)
        myresult = mycursor.fetchall()
        for x in myresult:
          result.append(list(x))

    for res in result:
        period = str(int(res[0]) + 9) + ':00 -- ' + str(int(res[0]) + 10) + ':00'
        usr_name = res[1]
        port = get_port(usr_name)
        queries.append([period, port])

    return queries

def update_url(ip):
    sql = "UPDATE mdl_url SET externalurl='http://" + ip + ":8188/apply\' WHERE id=1"
    mycursor.execute(sql)
    mydb.commit()

    sql = "UPDATE mdl_url SET externalurl='http://" + ip + ":8188/register\' WHERE id=2"
    mycursor.execute(sql)
    mydb.commit()

    sql = "UPDATE mdl_url SET externalurl='http://" + ip + ":8188/query\' WHERE id=3"
    mycursor.execute(sql)
    mydb.commit()

    return mycursor.rowcount
```

### 4. 配置內嵌頁面

通過python和html完成申請、預約與查詢網頁

**申請網站 register.py與register.html**

界面如下圖



申請U740使用帳號

申請需知

該項目 (U740線上調試環境) 目前尚處於測試階段，請輸入用戶名以及SSH Key後點擊申請

用戶名：請輸入用戶名
SSH Key：請輸入SSH Key

申請

輸入用戶名申請U740使用帳號

```python
from flask import *
import database
from my_docker import docker_manage
from pathlib import Path
import subprocess

app = Flask(__name__)

@app.route('/register', methods=['GET', 'POST'])
def register():
    id = None
    usr_name = None

    if request.method == 'POST':
        id = request.args.get('id')
        usr_name = request.form.get('usr_name')
        ssh_key = request.form.get('ssh_key')

        # 判斷用戶名以及SSH KEY是否為空
        if usr_name != '' and usr_name != ' ' and usr_name is not None and ssh_key !=
'':

            # 檢查用戶是否已在數據庫中
            existed = database.check_id(id)

            # 不存在則報錯
            if existed is not None:
                return render_template('register.html', existed=existed)
            else:
                path = "/docker_usr/info/" + usr_name + "/log"
                Path(path).mkdir(parents=True, exist_ok=True)

                path = "./my_docker/ssh_keys/" + usr_name
                Path(path).mkdir(parents=True, exist_ok=True)

                filename = "/authorized_keys"

                with open(path + filename, "w") as f:
                    f.write(ssh_key + '\n')

                # 數據庫中添加用戶
                result = database.add(id, usr_name)

                if result == 1:
                    outputs = docker_manage.make_docker(usr_name)

                    return render_template('register.html', success=usr_name)

                else:
                    return render_template('register.html', fail='fail')
        else:
            return render_template('register.html', wrong_info='wrong')

    return render_template('register.html')

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8184, debug=True)
```

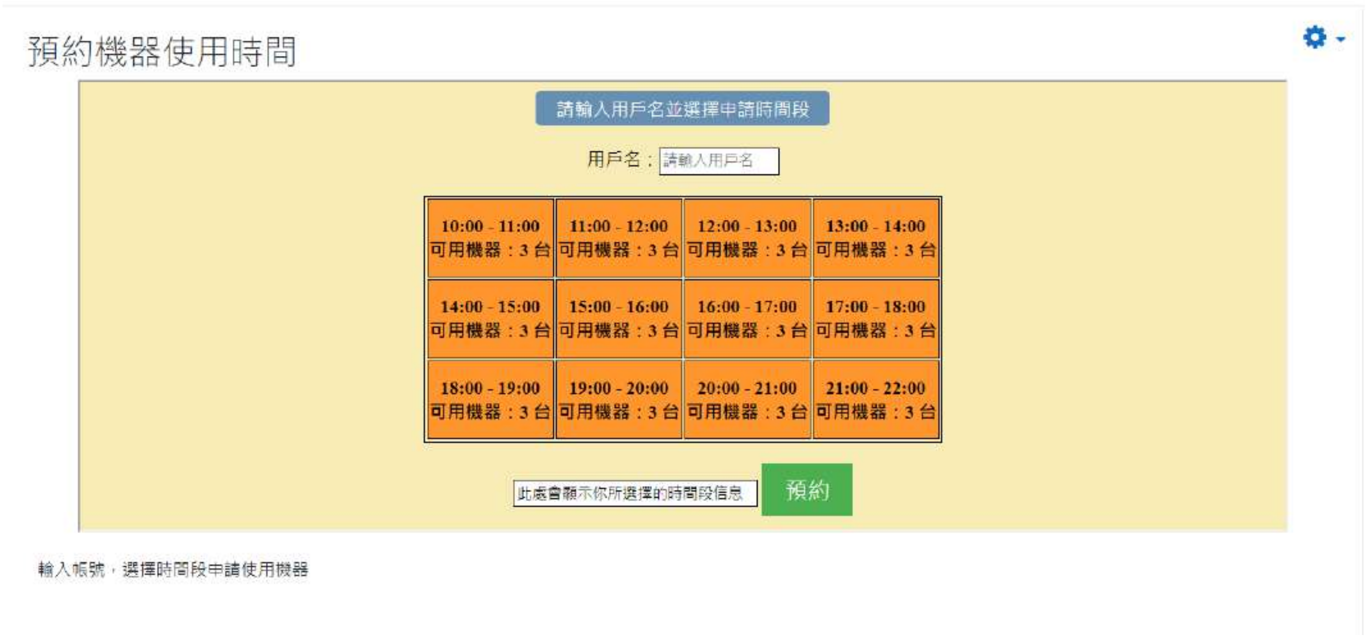
register.html

```
<html>
  <head>
    <title>U740 Register</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script>
      function showButtonEvent()
      {
        document.getElementById("Wait").style.display="block";
      }
    </script>
  </head>

  <style>
    .info {
      padding: 20px 10px;
      border:none;
      text-align: center;
      font-size: 200%;
      background: rgb(41, 105, 176, 0.7);
      color: rgb(255, 255, 255);
      border-radius: 4px;
    }

    .info2 {
      padding: 20px 10px;
      border:none;
      text-align: center;
      font-size: 100%;
      background: rgb(41, 105, 176, 0.7);
      color: rgb(255, 255, 255);
      border-radius: 4px;
    }

    .button {
      background-color: #4CAF50; /* Green */
      border: none;
      color: white;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 20px;
    }

  </style>
  <body style="background-color:rgb(248, 236, 182);">
    <center>
    <div class="container">
      <br><span class="info">申请需知</span><br><br><br>
      该项目 (U740线上调试环境) 目前尚处于测试阶段，请输入用户名以及SSH Key后点击申请
      <form action="" method="post"
onsubmit="document.getElementById('btn').disabled=true;">
        <br>
        用户名：<input type="text" placeholder="请输入用户名" name="usr_name" value="{{
request.form.usr_name }}">
        <br>
        SSH Key：<input type="text" placeholder="请输入SSH Key" name="ssh_key" value="
{{ request.form.ssh_key }}">
        <br>
        {% if success %}
          <p class="apply"><strong>用户 "{{ success }}" 申请成功！若要使用机器请预约时间
</strong>
        {% endif %}
        {% if fail %}
          <p class="apply"><strong>系统错误，请重新申请</strong>
        {% endif %}

        <br>
        {% if not success %}
        <input id="btn" class="button" type="submit" value="申请"
onclick="showButtonEvent()">
        {% endif %}

        {% if wrong_info %}
          <p class="apply"><strong>请输入用户名以及SSH Key！</strong>
        {% endif %}

        {% if existed %}
          <p class="apply"><strong>该Moodle帐号已注册过</strong>
        {% endif %}
      </form>

      <br>
      <div id="Wait" style="display: none;">
        <span class="info2">申请中，请稍等片刻</span>
      </div>


    </div>
    </center>
  </body>
</html>
```

**預約與查詢網站apply.py與apply.html, success.html, fail.html**

界面如下圖

預約



預約機器使用時間

請輸入用戶名並選擇申請時間段

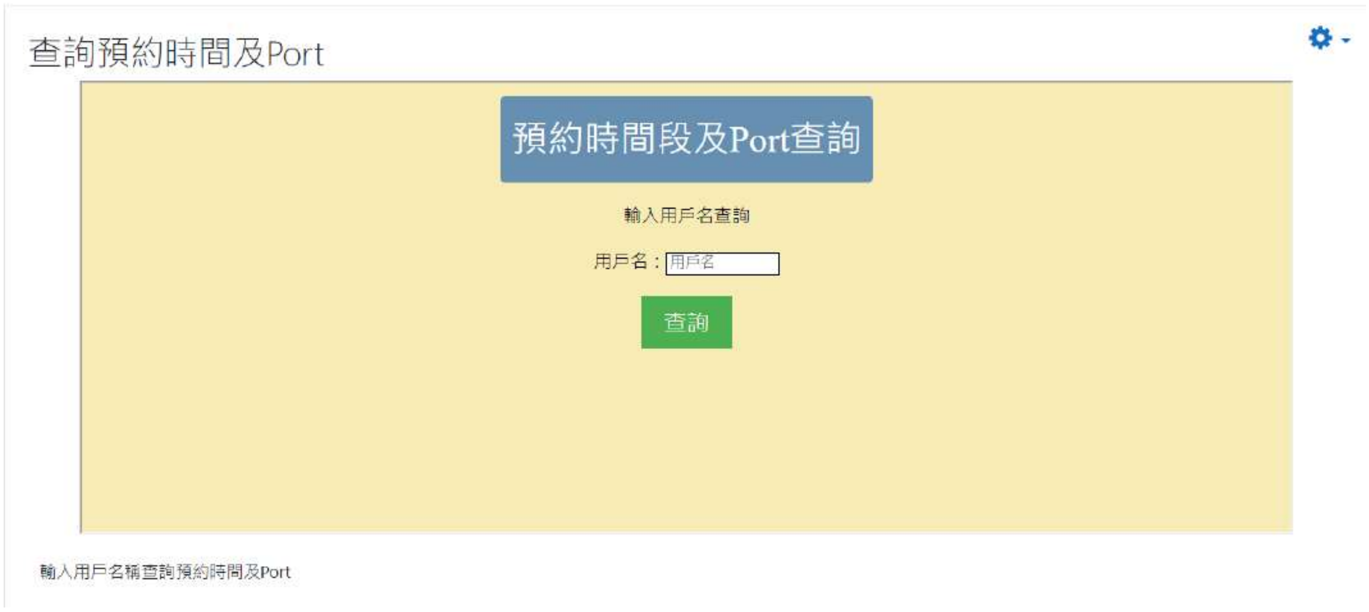
用戶名：請輸入用戶名

| | | | |
|---|---|---|---|
| 10:00 - 11:00 可用機器：3 台 | 11:00 - 12:00 可用機器：3 台 | 12:00 - 13:00 可用機器：3 台 | 13:00 - 14:00 可用機器：3 台 |
| 14:00 - 15:00 可用機器：3 台 | 15:00 - 16:00 可用機器：3 台 | 16:00 - 17:00 可用機器：3 台 | 17:00 - 18:00 可用機器：3 台 |
| 18:00 - 19:00 可用機器：3 台 | 19:00 - 20:00 可用機器：3 台 | 20:00 - 21:00 可用機器：3 台 | 21:00 - 22:00 可用機器：3 台 |

此處會顯示你所選擇的時間段信息 預約

輸入帳號，選擇時間段申請使用機器

查詢



查詢預約時間及Port

預約時間段及Port查詢

輸入用戶名查詢

用戶名：用戶名

查詢

輸入用戶名稱查詢預約時間及Port

apply.py

```python
from flask import *
from time import localtime
import subprocess
import numpy as np
import database

app = Flask(__name__)

# 查詢
@app.route('/query', methods=['GET', 'POST'])
def query():
    if request.method == 'POST':
        usr_name = request.form.get('usr_name')

        if usr_name != "":
            # 檢查用戶是否存在數據庫中
            usr_exist = database.check(usr_name)
            if usr_exist is not None:
                queries = database.get_status_port_queries(usr_name)
                # 如果查詢預約時間段不為空
                if queries:
                    return render_template('query.html', queries=queries,
usr_name=usr_name)
                # 為空
                else:
                    return render_template('query.html', no_query='yes',
usr_name=usr_name)
            # 用戶不存在
            else:
                return render_template('query.html', wrong_username=usr_name)
        # 查詢用戶名稱為空
        else:
            return render_template('query.html', no_username='no_username')
    else:
        return render_template('query.html')

# 申請
@app.route('/apply', methods=['GET', 'POST'])
def apply():
    if request.method == 'POST':
        # 獲取用戶名、時間段
        usr_name = request.form.get('usr_name')
        period = request.form.get('time_dialog')
        period = int(period[11]+period[12]) - 9
        usr_exist = database.check(usr_name)
        usr_appointment = usr_name in database.get_timetable(period)[0]

        #  檢查用戶名稱是否存在
        if usr_exist is not None and usr_appointment is False:
            previous_usrs = []
            if period != 1:
                previous_usrs = database.get_timetable(period - 1)
                previous_usrs = previous_usrs[0]
            if usr_name in previous_usrs:
                prev_machine_no = previous_usrs.index(usr_name) + 1
                now_machine_no = database.check_available_index(period)
                reserved_usr = database.get_one_timetable(period, prev_machine_no)[0]
                if reserved_usr is not None:
                    # switch
                    database.update_timetable(period, prev_machine_no, usr_name)
                    database.update_timetable(period, now_machine_no, reserved_usr)
                    port = database.get_port(usr_name)

                else:
                    database.update_timetable(period, prev_machine_no, usr_name)
                    port = database.get_port(usr_name)

            else:
                machine_no = database.check_available_index(period)
                port = database.get_port(usr_name)
                database.update_timetable(period, machine_no, usr_name)

            return render_template('success.html', port=port)
        else:
            return render_template('fail.html')
    else:
        available = []

        # 獲取空閒機器數量
        available = database.check_all_available()

        t = localtime()

        for i in range(len(available)):
            # 如果大於或等於預約時間即不可用
            if i + 10 <= t.tm_hour:
                available[i] = None

        # available1~12代表12個時段空閒機器數量
        return render_template('apply.html',
                               available1=available[0],
                               available2=available[1],
                               available3=available[2],
                               available4=available[3],
                               available5=available[4],
                               available6=available[5],
                               available7=available[6],
                               available8=available[7],
                               available9=available[8],
                               available10=available[9],
                               available11=available[10],
```

```python
                                        available12=available[11], )

if __name__ == '__main__':
    database.init_daily()

    t = localtime()

    log_name = str(t.tm_year) + '_' + str(t.tm_mon) + '_' + str(t.tm_mday) + '.log'

    with open("/home/ubuntu/applysystem/log/" + log_name, "w") as f:
        f.write('LOG for ' + log_name + '\n')

    app.run(host="0.0.0.0", port=8183, debug=True)
```

apply.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>U740 apply</title>
        <meta charset="UTF-8" ne="viewport" content="width=device-width, initial-
scale=1.0">
      </head>

<style>
    table, th, td {
        border:1px solid black;
    }

    .button {
        background-color: #4CAF50; /* Green */
        border: none;
        color: white;
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 20px;
    }

    .dialog {
        width:100px;
        height:20px;
        border:1px solid #000;
        text-align: left;
        background: rgb(255, 255, 255);
    }

    .info {
        width:210px;
        height:20px;
        border:1px solid #000;
        text-align: left;
        background: rgb(255, 255, 255);
    }

    tr{
        width:350px;
        height:70px;
        border:1px solid #000;
        text-align: center;
        background: rgb(255, 150, 45);
    }
    .active{
        background: rgb(77, 179, 233);
    }
```

```
</style>
<body style="background-color:rgb(248, 236, 182);">
    <center>
        <form class="form-horizontal" method="post" id="test">
            <div style="border-width: 2px ; width: 250px; height: 20px ; padding: 5px;
text-align: center; background-color: rgb(41, 105, 176, 0.7);border-radius: 4px;">
                <span style="color: rgb(255, 255, 255);">请输入用户名并选择申请时间段
</span>
            </div>
            <br>
            <span style="color: rgb(0, 0, 0);">用户名：</span><input class="dialog"
type="text" placeholder="请输入用户名" id="usr_name" name="usr_name" value="{{
request.form.usr_name }}">
            <br><br>

            <table id="myTable">

                <tr>
                    {% if available1 %}
                        <th>10:00 - 11:00 <br> 可用机器：{{ available1 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available2 %}
                        <th>11:00 - 12:00 <br> 可用机器：{{ available2 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available3 %}
                        <th>12:00 - 13:00 <br> 可用机器：{{ available3 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available4 %}
                        <th>13:00 - 14:00 <br> 可用机器：{{ available4 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}
                </tr>

                <tr>
                    {% if available5 %}
                        <th>14:00 - 15:00 <br> 可用机器：{{ available5 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available6 %}
```

```
                        <th>15:00 - 16:00 <br> 可用机器：{{ available6 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available7 %}
                        <th>16:00 - 17:00 <br> 可用机器：{{ available7 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available8 %}
                        <th>17:00 - 18:00 <br> 可用机器：{{ available8 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}
                </tr>

                <tr>
                    {% if available9 %}
                        <th>18:00 - 19:00 <br> 可用机器：{{ available9  }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available10 %}
                        <th>19:00 - 20:00 <br> 可用机器：{{ available10 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available11 %}
                        <th>20:00 - 21:00 <br> 可用机器：{{ available11 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}

                    {% if available12 %}
                        <th>21:00 - 22:00 <br> 可用机器：{{ available12 }} 台</th>
                    {% else %}
                        <th>该时间段已过或已约满</th>
                    {% endif %}
                </tr>

            </table>
            <br>
            <input class="info" type="text" id="time_dialog" name="time_dialog"
readonly="readonly" value="此处会显示你所选择的时间段信息">
            <input class="button" class="btn btn-default" name="sub" type="submit"
value="预约" id="sub">
```

```
    </center>

    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
    <script>

    document.getElementById('sub').onclick = function() {
        var invalid  = document.getElementById('time_dialog').value;
        var usr_name = document.getElementById('usr_name').value;

        if(usr_name == "") {
            alert('请输入用户名！');
        }

        if(invalid == '此处会显示你所选择的时间段信息') {
            alert('请选择时间段！');
        }

        if(invalid != '此处会显示你所选择的时间段信息' && usr_name != "") {
            return confirm('请确认你的用户名为: ' + usr_name + '\n' + invalid + '\n如有
错请点击取消');
        }
        return false;
    }

    $(function(){
        $('th').on('click',function(){
            var period = $(this).text();

            if (period != "该时间段已过或已约满") {
            $('th').removeClass('active').addClass();
            $(this).addClass('active');
            period_start = $(this).index() + $(this).closest('tr').index() * 4 + 10;
            period_end   = $(this).index() + $(this).closest('tr').index() * 4 + 11;

            document.getElementById("time_dialog").value = "你所选择的时间段为: " +
period_start + ":00 --" + period_end + ":00";
            }

        })
    })
    </script>

</body>
</html>
```

success.html

```html
<html>
  <head>
    <title>U740 apply success</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <style>
    .info {
      padding: 20px 10px;
      border:none;
      text-align: center;
      font-size: 200%;
      background: rgb(41, 105, 176, 0.7);
      color: rgb(255, 255, 255);
      border-radius: 4px;
    }

    .button {
      background-color: #4CAF50; /* Green */
      border: none;
      color: white;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 20px;
    }
  </style>
  <body style="background-color:rgb(248, 236, 182);">
    <center>
    <div class="container">
      {% if port %}
        <p class="info"><strong>分配成功，请使用该Port登陆Docker:</strong> {{ port }}
      {% endif %}
      <br><br>
      <input class="button" type="button" name="Submit"
onclick="javascript:history.back(-1);" value="返回上一页">
    </div>
    </center>
  </body>
</html>
```

fail.html

```html
<html>
  <head>
    <title>U740 apply</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <style>
    .info {
      padding: 20px 10px;
      border:none;
      text-align: center;
      font-size: 200%;
      background: rgb(41, 105, 176, 0.7);
      color: rgb(255, 255, 255);
      border-radius: 4px;
    }

    .button {
      background-color: #4CAF50; /* Green */
      border: none;
      color: white;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 20px;
    }
  </style>
  <body style="background-color:rgb(248, 236, 182);">
    <center>
    <div class="container">
      <p class="info"><strong>分配失败，请检查是否输入正确帐号或已预约该时段。</strong>
      <br><br>
      <input class="button" type="button" name="Submit"
onclick="javascript:history.back(-1);" value="返回上一页">
    </div>
    </center>
  </body>
</html>
```

query.html

```html
<html>
  <head>
    <title>U740 Query</title>
    <meta ne="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <style>
    .info {
      padding: 20px 10px;
      border:none;
      text-align: center;
      font-size: 200%;
      background: rgb(41, 105, 176, 0.7);
      color: rgb(255, 255, 255);
      border-radius: 4px;
    }

    .button {
      background-color: #4CAF50; /* Green */
      border: none;
      color: white;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 20px;
    }

    .dialog {
      width:100px;
      height:20px;
      border:1px solid #000;
      text-align: left;
      background: rgb(255, 255, 255);
    }

  </style>
  <body style="background-color:rgb(248, 236, 182);">
    <center>
    <div class="container">
      <br>
      <span class="info">预约时间段及Port查询</span><br><br><br>
      输入用户名查询
      <form action="" method="post" onsubmit="onsubmit()">
        <br>
        用户名：<input class="dialog" type="text" placeholder="用户名" name="usr_name"
value="{{ request.form.usr_name }}">
        <br><br>
        <input id="btn" class="button" type="submit" value="查询">
      </form>
```

```
        {% if queries %}
          <td>用户 "{{ usr_name }}" 预约情况如下：</td><br>
          {% for query in queries %}
            <tr>
              <td>时间段：{{ query[0] }} || Port: {{ query[1] }}</td><br>
            </tr>
          {% endfor %}
        {% endif %}

        {% if no_query %}
          <p>用户 "{{ usr_name }}" 今天没有任何预约！</p><br>
        {% endif %}

        {% if no_username %}
          <p>请输入用户名！</p><br>
        {% endif %}

        {% if wrong_username %}
          <p>请检查用户名称 "{{ wrong_username }}" 是否正确！</p><br>
        {% endif %}

      </div>
      </center>
    </body>
  </html>
```

**整合網站 moodle.py**

```python
from flask import *
from time import localtime
import subprocess
import numpy as np
import database
from time import timezone
from flask_apscheduler import APScheduler
from datetime import datetime
from my_docker import docker_manage
from pathlib import Path
import check
import socket

def get_host_ip():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(('8.8.8.8', 80))
        ip = s.getsockname()[0]
    finally:
        s.close()
    return ip


# 服務器定期更新

# 每整點更新一次，用於Docker的更新
server_update_time_week = '0-6' # * 代表 週一到週日工作
server_update_time_hour = '10-21' # 10-21 代表 10點開始第一次更新，21點最後一次更新
server_update_time_min = '00' # 代表 XX:00:XX 更新
server_update_time_sec = '01' # 代表 XX:XX:01 更新

# 每整點的55分檢查一次，用於提前5分鐘斷線的提示
server_check_time_week = '0-6'
server_check_time_hour = '10-21'
server_check_time_min = '55' # 代表 XX:55:XX 更新
server_check_time_sec = '01'

app = Flask(__name__)

class Config(object):
    SCHEDULER_API_ENABLED = True

scheduler = APScheduler()

# 到時前五分鐘提示
@scheduler.task('cron', id='expired_remind', day_of_week=server_check_time_week,
                                             hour=server_check_time_hour,
                                             minute=server_check_time_min,
                                             second=server_check_time_sec)
def expired_remind():
    now_user_names = database.get_status()[0]
    now_user_names = [x for x in now_user_names if x is not None]

    if now_user_names:
        for now_user_name in now_user_names:
            check.expired_remind(now_user_name)


# 更新時間表
@scheduler.task('cron', id='update_timetable',day_of_week=server_update_time_week,
                                              hour=server_update_time_hour,
                                              minute=server_update_time_min,
                                              second=server_update_time_sec)
def update_machine():
    t = localtime()
    period = t.tm_hour - 9


    # 獲取下個時間段使用信息
    next_user_names = database.get_timetable(period)[0]
    # 判斷下個時間段是否有人使用，即時間表中是否為 'NULL'
    next_user_exist = [x for x in next_user_names if x is not None]

    # 第一個時間段不用特殊判斷
    if period == 1:

        #user_names = [x for x in user_names if x is not None]

        # 如果預約時間表不為空，即有人預約
        if next_user_names:
            for next_user_name in next_user_names:
                if next_user_name is not None:
                    port = database.get_port(next_user_name)

                    machine_no = next_user_names.index(next_user_name) + 1

                    output = docker_manage.start_docker(next_user_name, port,
machine_no)

                    t = localtime()

                    # 記錄Docker啟動時間
                    with open("/home/ubuntu/applysystem/log/" + log_name, "a") as f:
                        f.write('Start docker ' + output + ' at ' + str(t.tm_hour) +
':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                    # 更新當前機器使用狀態
                    database.update_status(period, machine_no, next_user_name)

    # 之後的時間段，需要判斷 下個時間段有否預約，是否為同一人預約，是否使用同一機器
    else:
        now_user_names = database.get_status()[0]
        now_user_exist = [x for x in now_user_names if x is not None]

        # 下個時間段有人預約
        if next_user_exist:
            # 如果目前機器有分配用戶
            # 則檢查下個時間段該用戶是否使用同一機器
            # 若為同一機器則不中斷SSH連線，若不為同一機器則中斷SSH連線(停止Docker)
            if now_user_exist:
                for now_user_name in now_user_names:
                    if now_user_name is not None:
                        # 不為同一人
                        if now_user_name not in next_user_names:

                            t = localtime()

                            machine_no = now_user_names.index(now_user_name) + 1

                            # 停止Docker
                            output = docker_manage.stop_docker(now_user_name,
machine_no)

                            # 記錄Docker停止時間
                            with open("/home/ubuntu/applysystem/log/" + log_name, "a")
as f:
                                f.write('Stop  docker ' + output + ' at ' +
str(t.tm_hour) + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')
                        else:
                            now_machine_no = now_user_names.index(now_user_name) + 1
                            next_machine_no = next_user_names.index(now_user_name) + 1

                            if now_machine_no != next_machine_no:
                                output = docker_manage.stop_docker(now_user_name,
now_machine_no)

                                with open("/home/ubuntu/applysystem/log/" + log_name,
"a") as f:
                                    f.write('Stop  docker ' + output + ' at ' +
str(t.tm_hour) + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')


            # 根據預約時間表啟動下個時間段的Docker
            for next_user_name in next_user_names:
                if next_user_name is not None and next_user_name not in
now_user_names:

                    port = database.get_port(next_user_name)
                    t = localtime()

                    machine_no = next_user_names.index(next_user_name) + 1
                    output = docker_manage.start_docker(next_user_name, port,
machine_no)

                    with open("/home/ubuntu/applysystem/log/" + log_name, "a") as f:
                        f.write('Start docker ' + output + ' at ' + str(t.tm_hour) +
':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                    # 更新當前機器使用狀態
                    database.update_status(period, machine_no, next_user_name)

        # 下個時間段沒有預約
        else:
            # 檢查當前分配的用戶是否正在使用 <-- 檢查SSH連線情況，如果SSH用戶為0，則Docker
沒人使用
            if now_user_exist:
                for now_user_name in now_user_names:
                    if now_user_name is not None:
                        ssh_usr = check.check_alive(now_user_name)

                        if ssh_usr == '0':

                            t = localtime()

                            machine_no = now_user_names.index(now_user_name) + 1

                            # 停止Docker
                            output = docker_manage.stop_docker(now_user_name,
machine_no)

                            with open("/home/ubuntu/applysystem/log/" + log_name, "a")
as f:
                                f.write('Stop  docker ' + output + ' at ' +
str(t.tm_hour) + ':' + str(t.tm_min) + ':' + str(t.tm_sec) + '\n')

                            # 保存用戶操作記錄
                            # check.save_user_log(now_user_name, machine_no)

                            # 更新當前機器使用狀態
                            database.update_status(period, machine_no, 'NULL')

# 申請帳號
@app.route('/register', methods=['GET', 'POST'])
def register():
    id = None
    usr_name = None

    if request.method == 'POST':
        id = request.args.get('id')
        usr_name = request.form.get('usr_name')
        ssh_key = request.form.get('ssh_key')
        # 判斷用戶名以及SSH KEY是否為空
        if usr_name != '' and usr_name != ' ' and usr_name is not None and ssh_key !=
'':

            # 檢查用戶是否已在數據庫中
            existed = database.check_id(id)

            # 不存在則報錯
            if existed is not None:
                return render_template('register.html', existed=existed)
            else:
                path = "/docker_usr/info/" + usr_name + "/log"
                Path(path).mkdir(parents=True, exist_ok=True)

                path = "./my_docker/ssh_keys/" + usr_name
                Path(path).mkdir(parents=True, exist_ok=True)

                filename = "/authorized_keys"

                with open(path + filename, "w") as f:
                    f.write(ssh_key + '\n')

                # 數據庫中添加用戶
                result = database.add(id, usr_name)

                if result == 1:
                    outputs = docker_manage.make_docker(usr_name)

                    return render_template('register.html', success=usr_name)

                else:
                    return render_template('register.html', fail='fail')
        else:
            return render_template('register.html', wrong_info='wrong')

    return render_template('register.html')

# 查詢預約
@app.route('/query', methods=['GET', 'POST'])
def query():
    if request.method == 'POST':
        usr_name = request.form.get('usr_name')

        if usr_name != "":
            # 檢查用戶是否存在數據庫中
            usr_exist = database.check(usr_name)
            if usr_exist is not None:
                queries = database.get_status_port_queries(usr_name)
                # 如果查詢預約時間段不為空
                if queries:
                    return render_template('query.html', queries=queries,
usr_name=usr_name)
                # 為空
                else:
                    return render_template('query.html', no_query='yes',
usr_name=usr_name)
            # 用戶不存在
            else:
                return render_template('query.html', wrong_username=usr_name)
        # 查詢用戶名稱為空
        else:
            return render_template('query.html', no_username='no_username')
    else:
        return render_template('query.html')

# 預約時間
@app.route('/apply', methods=['GET', 'POST'])
def apply():
    if request.method == 'POST':
        # 獲取用戶名、時間段
        usr_name = request.form.get('usr_name')
        period = request.form.get('time_dialog')
        period = int(period[11]+period[12]) - 9
        usr_exist = database.check(usr_name)
        usr_appointment = usr_name in database.get_timetable(period)[0]

        #  檢查用戶名稱是否存在
        if usr_exist is not None and usr_appointment is False:
            previous_usrs = []
            if period != 1:
                previous_usrs = database.get_timetable(period - 1)
                previous_usrs = previous_usrs[0]
            if usr_name in previous_usrs:
                prev_machine_no = previous_usrs.index(usr_name) + 1
                now_machine_no = database.check_available_index(period)
                reserved_usr = database.get_one_timetable(period, prev_machine_no)[0]
                if reserved_usr is not None:
                    # switch
                    database.update_timetable(period, prev_machine_no, usr_name)
                    database.update_timetable(period, now_machine_no, reserved_usr)
                    port = database.get_port(usr_name)

                else:
                    database.update_timetable(period, prev_machine_no, usr_name)
                    port = database.get_port(usr_name)

            else:
                machine_no = database.check_available_index(period)
                port = database.get_port(usr_name)
                database.update_timetable(period, machine_no, usr_name)

            return render_template('success.html', port=port)
```

```python
        else:
            return render_template('fail.html')
    else:
        available = []

        # 獲取空閒機器數量
        available = database.check_all_available()

        t = localtime()

        for i in range(len(available)):
            # 如果大於或等於預約時間即不可用
            if i + 10 <= t.tm_hour:
                available[i] = None

        # available1~12代表12個時段空閒機器數量
        return render_template('apply.html',
                               available1=available[0],
                               available2=available[1],
                               available3=available[2],
                               available4=available[3],
                               available5=available[4],
                               available6=available[5],
                               available7=available[6],
                               available8=available[7],
                               available9=available[8],
                               available10=available[9],
                               available11=available[10],
                               available12=available[11], )

if __name__ == '__main__':

    ip = get_host_ip()

    database.update_url(ip)

    app.config.from_object(Config())
    scheduler.init_app(app)
    scheduler.start()

    database.init_daily()

    t = localtime()

    log_name = str(t.tm_year) + '_' + str(t.tm_mon) + '_' + str(t.tm_mday) + '.log'

    with open("/home/ubuntu/applysystem/log/" + log_name, "w") as f:
        f.write('LOG for ' + log_name + '\n')

    app.run(host="0.0.0.0", port=8188, debug=False)
```