火災警報系統

Fire Alarm System


楊家棠


張晉銘

火災警報系統

 楊家棠  張晉銘

系統介紹

這個功能利用 **DHT22 溫濕度感測器**和**MQ-7一氧化碳感測器**來監測火災時的溫度和氣體濃度。一旦檢測到溫度或一氧化碳濃度超過安全範圍，系統會立刻啟動**攝影機**和**蜂鳴器**。攝影機會拍攝現場並把影像發送到伺服器，用戶可以通過瀏覽器實時查看家中的狀況。

此功能設計能有效減少誤報，即使其中一個感測器出現故障，還是能持續進行監控，保證火災偵測的可靠性。

火災警報系統


楊家棠


張晉銘

流程圖

1. 初步偵測：**DHT22偵測溫度是否超過 45℃ 或是否MQ-7偵測有一氧化碳濃度超過 47%**。

2. 警報觸發：如果2個同時偵測到異常或1個偵測時間超過30秒，系統將**啟動蜂鳴器**，並**啟動攝影機拍攝及LED燈**(提供相機照明) 將影像傳送至伺服器。

3. 警報傳送：系統會將火災警報回傳至伺服器。

4. 連續監控：若未檢測到異常，系統會解除警報並回復相關設備繼續監控環境。

火災警報功能 | 設備圖


楊家棠

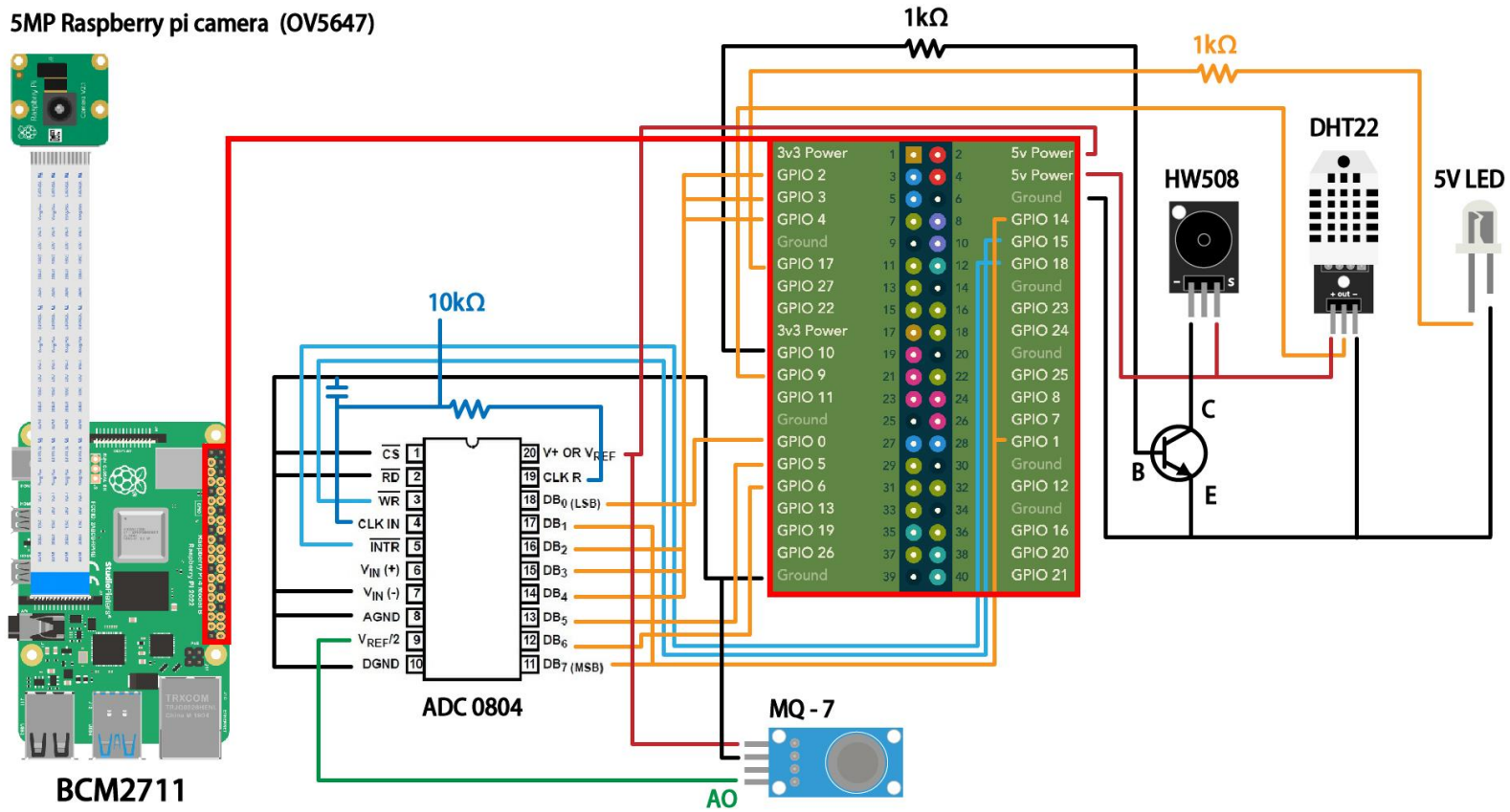

張晉銘

1. ADC 0804 腳位分配

由於 ADC 0804 需要佔用較多腳位，因此在開發初期便明確分配各設備所需的腳位，以避免整合時發生衝突。

2. 蜂鳴器硬體設計

若蜂鳴器的正極直接接回 GPIO pin 可能會導致 GPIO pin 燒毀。因此改用 NPN 電晶體，讓 GPIO 將訊號輸出至電晶體的基極 (B)，從而安全地啟動蜂鳴器。

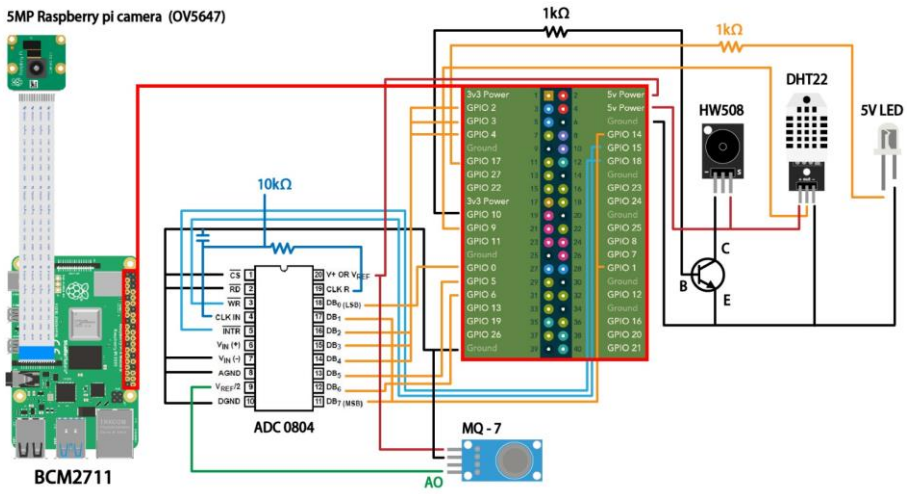


火災警報系統

設備表格對照


楊家棠


張晉銘

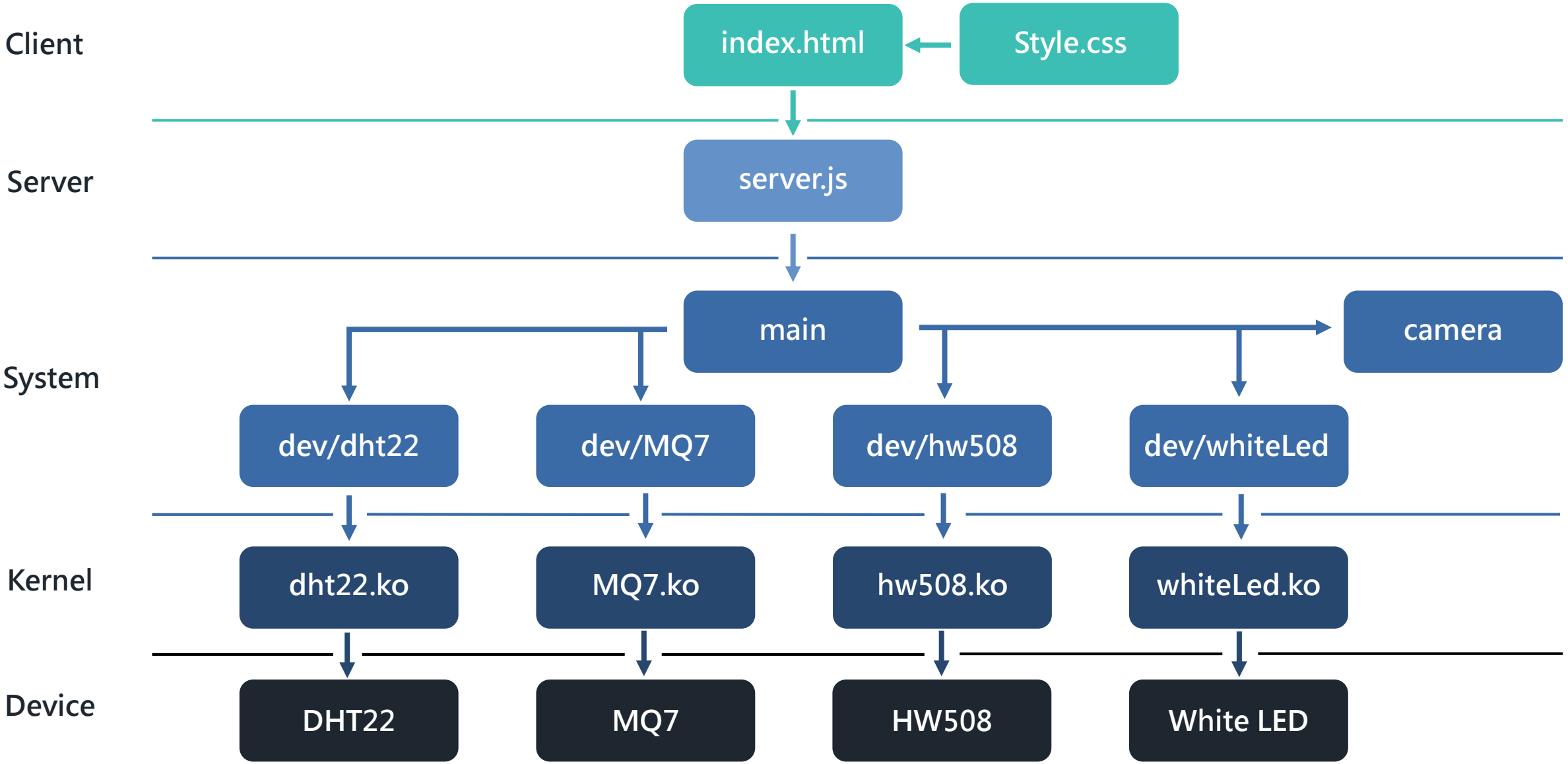


細部功能	使用元件/模組	介面	腳位
偵測溫度	DHT22	D Input D Output	GPIO9
偵測一氧化碳	MQ-7	A Output	V in(+) (ADC 0804)
類比數位轉換器	ADC 0804	A Intput D Input D Output	GPIO0 GPIO1 GPIO2 GPIO3 GPIO4 GPIO5 GPIO6 GPIO14 GPIO15 GPIO18
LED 照明	LED–W (5V)	D Output	GPIO17
蜂鳴器	HW-508 (buzzer)	D Output	GPIO10
相機	5 MP Raspberry Pi Camera (OV5647)	CSI	

火災警報系統 | 架構圖


楊家棠


張晉銘



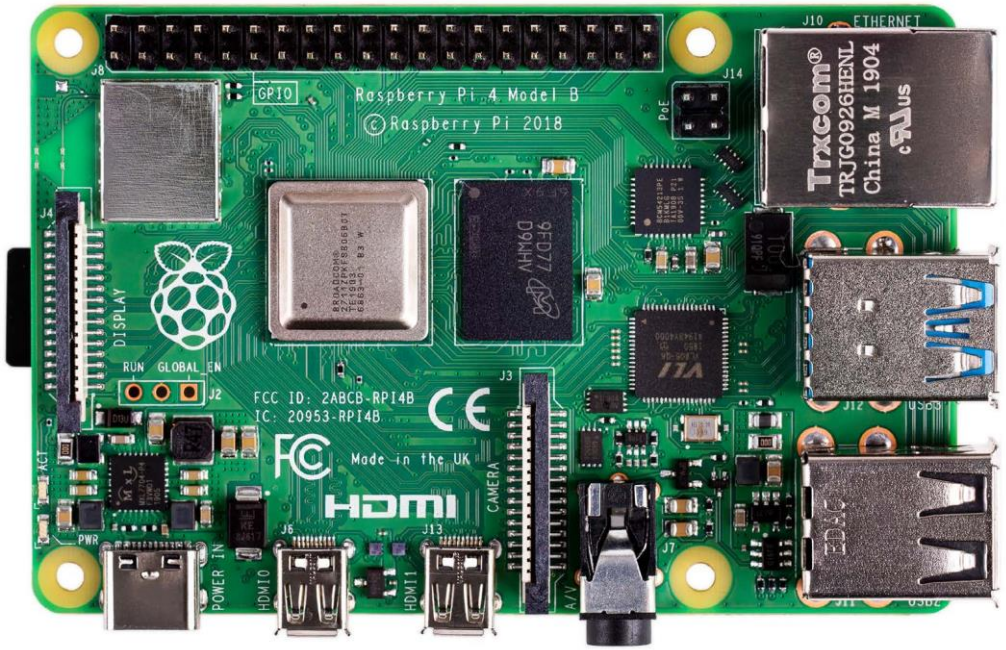
火災警報系統 | Device

 楊家棠  張晉銘

Raspberry pi 4B (BCM2711)

開發系統版本: Linux raspberrypi 5.10.103-v7l

開發系統架構: ARMv7



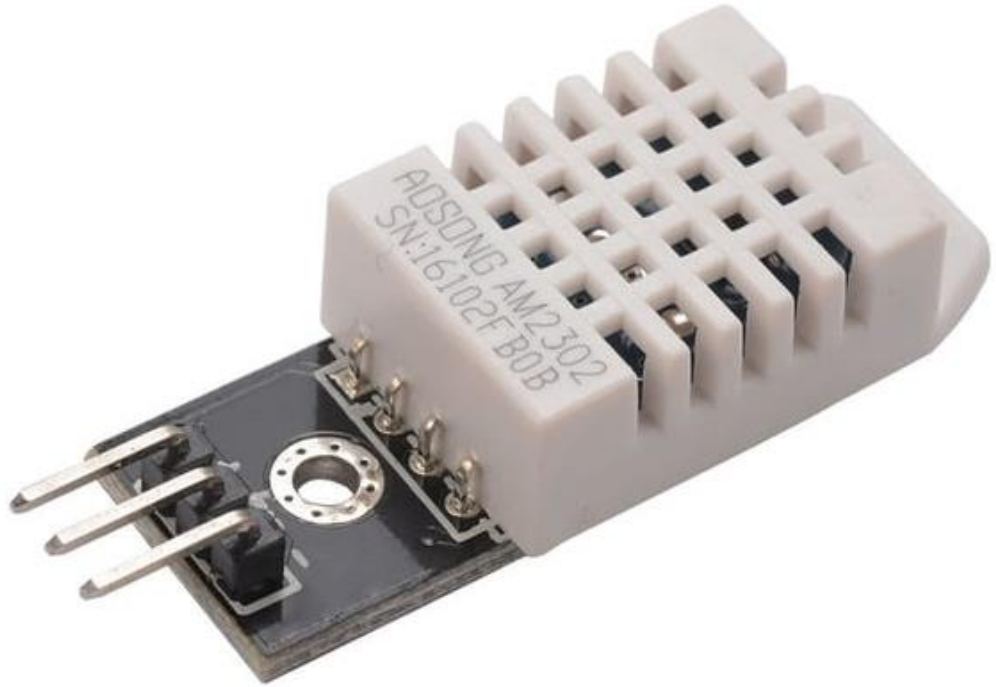
火災警報系統 | Device

楊家棠

DHT22 溫溼度感測器

DHT22 能精確測量環境溫度（-40°C 至 80°C）和濕度（0% 至 100% RH）。利用電容式濕度感測元件和熱敏電阻測量濕度和溫度，

根據 DHT22 感測器測得的溫度值來決定是否啟動火災警報。如果溫度超過設定的閾值，系統將觸發警報，以便及時提醒屋主有火災發生。

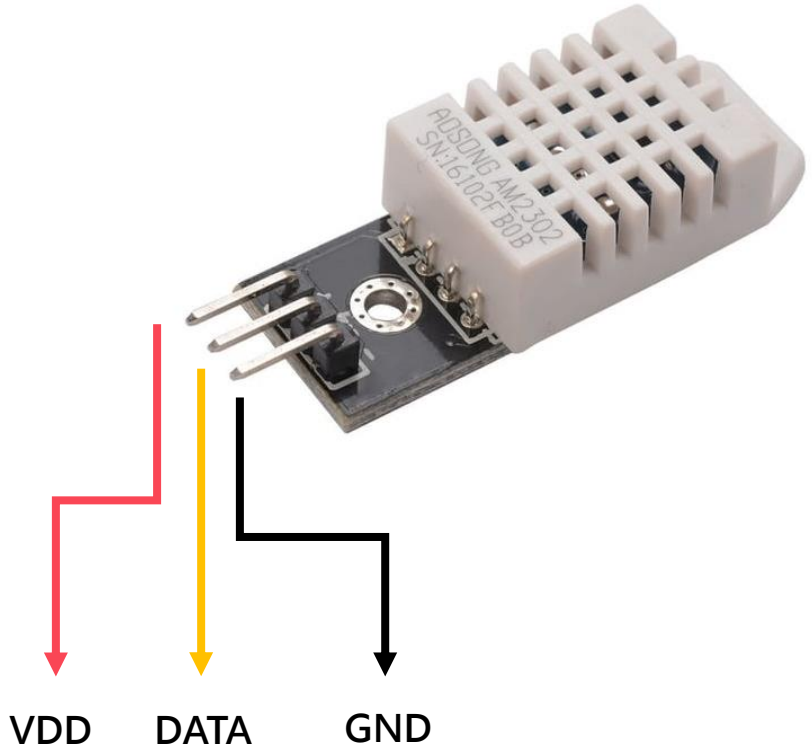


火災警報系統 | Device

楊家棠

DHT22 硬體規格

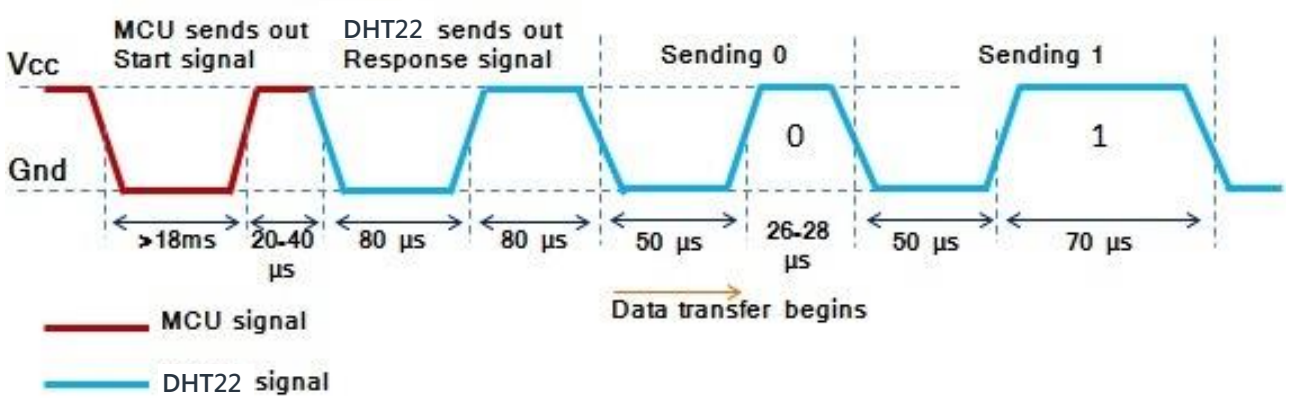
Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm



火災警報系統 | Device

楊家棠

DHT22 運作原理



DHT22單筆資料格式					
名稱	濕度		溫度		檢驗碼
單位	整數位(高位元)	小數位(低位元)	整數位(高位元)	小數位(低位元)	檢驗和
	byte0(起始)	byte1	Byte2	Byte3	Byte4
二進位	00101100	00000000	00001001	01110100	01000110
十進位	44.0		24.2		70

首先將 GPIO 設定為輸出模式，讓 BCM2711 發送啟動訊號給 DHT22。隨後將 GPIO 切換為輸入模式，當 BCM2711 接收完 DHT22 傳送的前導訊號來確認交握後，便開始接收 DHT22 的溫濕度資料。透過設立 flag 來判斷 DHT22 拉高電平的時間：若時間介於 26-28 us，則 bit 值為 0；若超過 70 us，則 bit 值為 1，共計 40 bit。

最後將前16+16bit的資料，經過換算後再存入buffer中。

2段 8bit 資料轉 16bit 資料溫度換算:

1/10 * ([高位元]*256 + [低位元])

火災警報系統 | Device

楊家棠

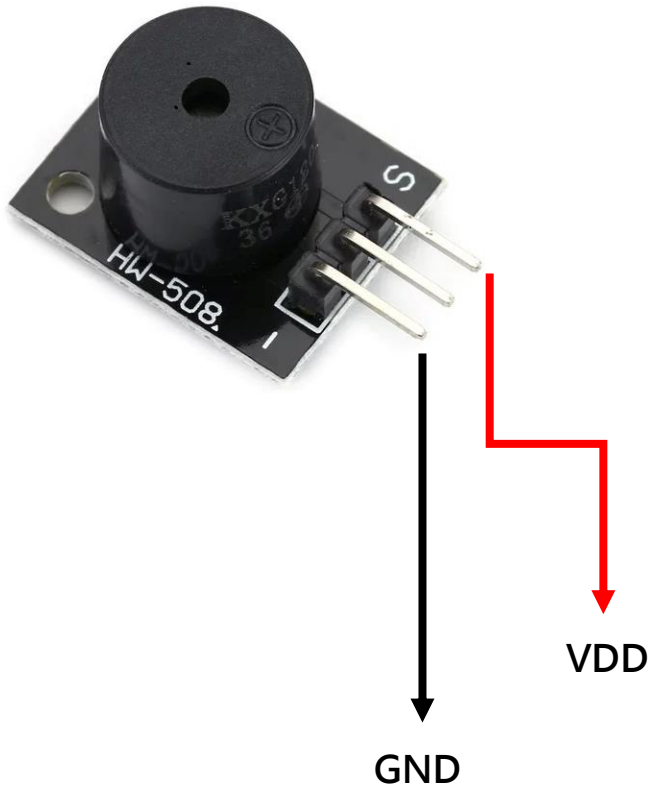
HW-508 有緣蜂鳴器

HW-508 內建振盪電路，只需接入直流電源即可發聲，而不需要外部音頻信號。

當火災警報觸發時，HW-508 蜂鳴器將立即響起，在第一時間發出警報以迅速通知屋主發生火災，提醒屋主及時採取行動。

火災警報系統 | Device

楊家棠



HW-508 硬體規格

- Operating Voltage : 3.3 ~ 5V DC
- Rated Current* : ≤30mA
- Sound Output at 10cm* : ≥85dB
- Resonant Frequency : 2300 ±300Hz
- Tone : Continuous
- Operating Temperature : -25°C to +80°C
- Storage Temperature : -30°C to +85°C
- Dimensions: 15.3 x 23.6 x 12.4mm

火災警報系統 | Device

楊家棠

樹莓派攝影機 (OV5647) & LED

支援高達 2592 x 1944 像素的靜態圖像和 1080p 30fps 的影像錄製，通過 CSI 接口提供高品質影像。小巧易安裝，適合家庭監控和自動化項目。

當火災警報觸發時，便會啟動攝影機並拍攝回傳影像至伺服器，有利於屋主及時判斷現況並採取相關行動。

相機啟動時便會一併啟動 LED ，提供攝影機拍攝時所需要的光源。

火災警報系統 | Device

楊家棠

樹莓派攝影機 & LED 硬體規格

5MP Raspberry Pi Camera 1 (OV5647)

- Fully Compatible with Both the Model A and Model B Raspberry Pi
- 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g
- Fully Compatible with many Raspberry Pi cases

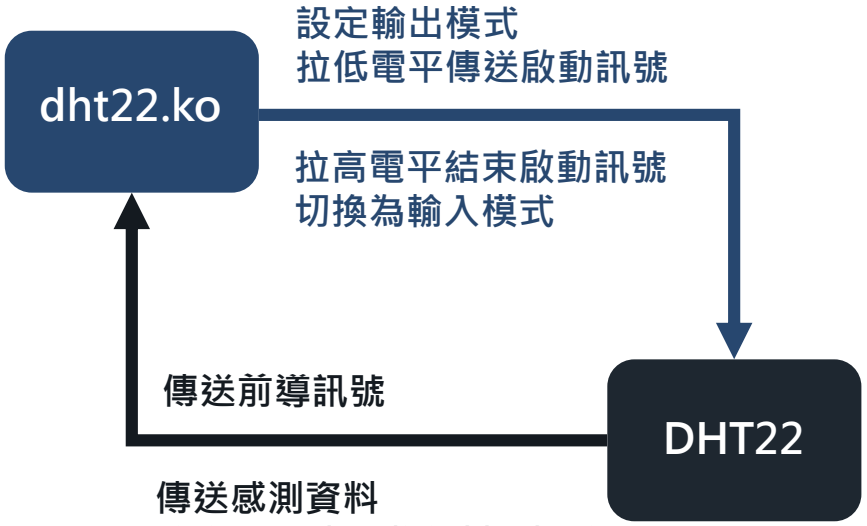
LED

- Diameter: 5φ
- Forward Voltage (VF): DC 5V (I = 20mA)
- Current: 20mA
- Wavelength: 520-530nm
- Luminous Intensity (IV): 5000-10000mcd
- Half Viewing Angle (2θ1/2): 30 degrees
- Dimensions: 5.0 x 8.7mm

火災警報系統 | Kernel - DHT22 driver

楊家棠

讀取溫度資料並進行運算



傳送感測資料
拉高電平時間做為判斷標準:
介於26-28us 值為0 ; 超過 70us 值為1

```
gpio_out(DHT22PIN, 0);
mdelay(30);
gpio_out(DHT22PIN, 1);
udelay(40);
if(data_in() == 0)
{
    while(!gpio_get_value(DHT22PIN))
    {
        udelay(5);
        i++;
        if(i>20)
        {
            printk("error data!\n");
            break;
        }
    }
```

```
for(i = 0;i < 5;i++)
{
    for(num = 0;num < 8;num++)
    {
        j = 0;
        while( !gpio_get_value(DHT22PIN) )
        {
            udelay(10);
            j++;
            if(j > 40)
            break;
        }
        flag = 0x0;
        udelay(28);
        if( gpio_get_value(DHT22PIN) )
        {
            flag = 0x01;
        }
```

```
temp = (buf[2] * 256 + buf[3]);

buf[2] = temp / 10;
buf[3] = temp % 10;
```

取得溫度的整數(高位元)與小數(低位元)後進行16bit運算
最後再÷10取得該值商數及餘數後分別放入陣列中

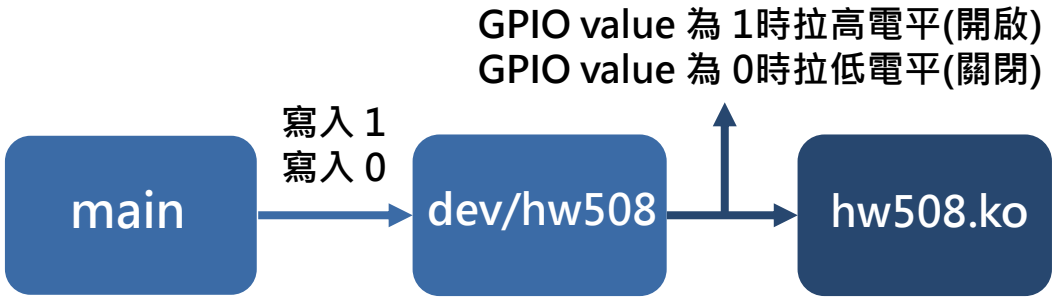
火災警報系統 | Kernel - HW508 driver

楊家棠

```
void control_buzzer(int value) {
    gpio_direction_output(HW508PIN, value);
}

static ssize_t buzzer_write(struct file *file, const char __user *b
    unsigned char value;
    int ret;

    if (size != 1) {
        return -EINVAL;  // Error: incorrect data size
    }

    ret = get_user(value, buffer);
    if (ret) {
        return ret;  // Return error code
    }

    if (value == '0') {
        control_buzzer(0);  // Turn off buzzer
    } else if (value == '1') {
        control_buzzer(1);  // Turn on buzzer
    } else {
        return -EINVAL;  // Error: unsupported data
    }

    return size;  // Return number of bytes written
```

開啟 / 關閉蜂鳴器

由 main 程式寫入字元設備檔來控制蜂鳴器，首先設定 GPIO 為輸出模式。當 buffer 內的字元為 1 時，將 GPIO 拉高電平以啟動蜂鳴器；當字元為 0 時，將 GPIO拉低電平以關閉蜂鳴器。

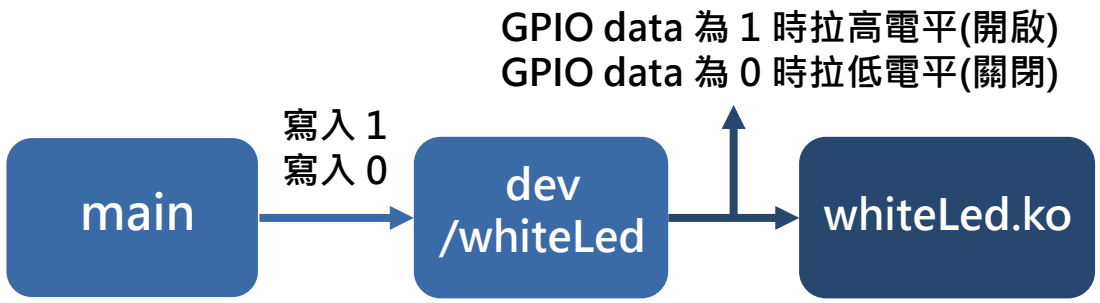


火災警報系統 | Kernel - LED driver

楊家棠

開啟 / 關閉 LED

由 main 程式寫入字元設備檔來控制 LED，首先設定 GPIO 為輸出模式。當 buffer 內的字元為 1 時，將 GPIO 拉高電平以打開 LED ；當字元為 0 時，將 GPIO拉低電平以關閉 LED。

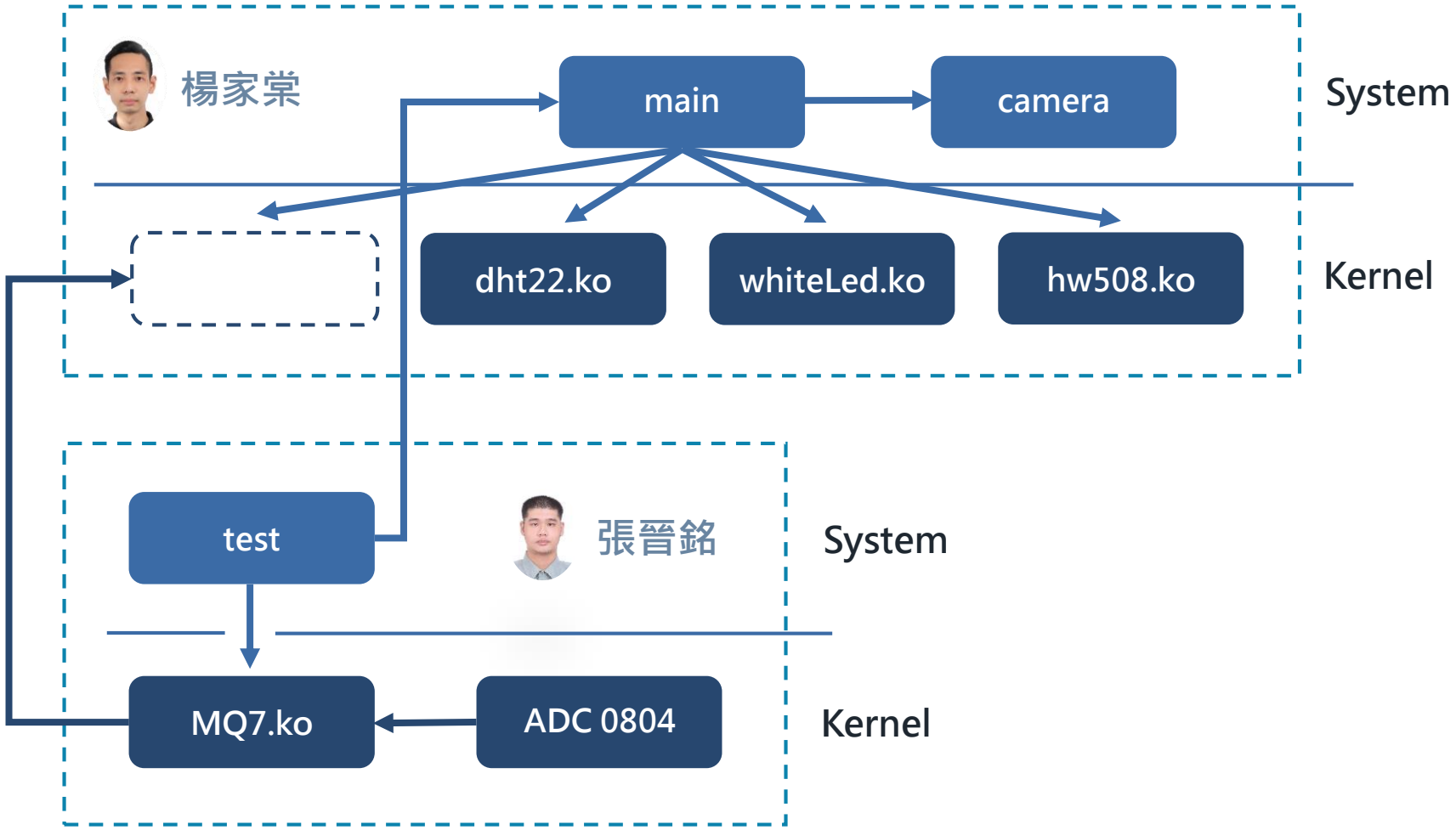



```c
static ssize_t whiteLed_write(struct file *file, const cha
    unsigned char value;
    int ret;

    if (size != 1) {
        return -EINVAL;  // Error: incorrect data size
    }

    ret = get_user(value, buffer);
    if (ret) {
        return ret;  // Return error code
    }

    if (value == '0') {
        control_led(0);  // Turn off LED
    } else if (value == '1') {
        control_led(1);  // Turn on LED
    } else {
        return -EINVAL;  // Error: unsupported data
    }

    return size;  // Return number of bytes written
```

火災警報系統 | System


楊家棠

系統整合示意圖



火災警報系統 | System - main program

楊家棠

1. 建立計時器

火災警報功能需調用多種設備，為確保DHT22及MQ7感測器在超過設定的閾值時能正常啟用其他設備並保持偵測，採用計時器為各項感測器及設備分配工作頻率，以避免產生衝突。

各項感測器 & 設備工作頻率

DHT22 溫度感測:		每2秒
MQ-7 一氧化碳感測:		每2秒
蜂鳴器(警報觸發時):	開/關間隔	每250毫秒
LED (警報觸發時):		持續開啟

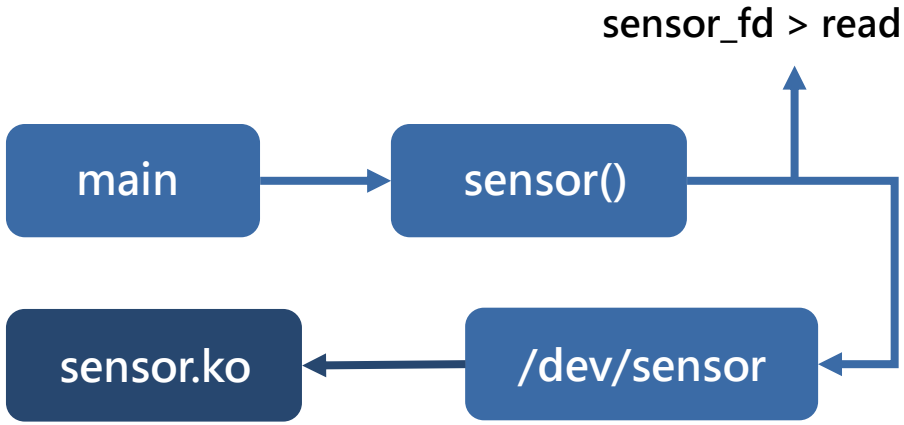
```
int main() {

while (1) {
    current_time = get_current_time();

    if (current_time - last_time_dht22 >= 2000) {  // 2s
        dht22();
        shm_flag[0] = tempz;
        last_time_dht22 = current_time;
    }

    if (current_time - last_time_MQ7 >= 2000) {  // 2s
        MQ7();
        shm_flag[1] = coVal;
        last_time_MQ7 = current_time;
    }

if (current_time - last_time_hw508 >= 250) {  // Buzzer frequency 250 ms
    if (buzzer_flag) {
        hw508_off();  // Buzzer OFF
    } else {
        hw508_on();   // Buzzer ON
    }
    buzzer_flag = !buzzer_flag;
    last_time_hw508 = current_time;
}
```

火災警報系統 | System - main program

楊家棠

2. 取得感測器偵測的數據

透過檔案描述子開啟 DHT22 和 MQ7 感測器的字元設備檔，再由 read 函數讀取 dht22.ko / MQ7.ko 傳送到 buffer 中的資料，將讀取資料回傳給 main 函數使用。




```c
int dht22(void) {
    int temperature_fd;
    int ret = 0;
    char buf[5];
    unsigned char tempx = 0;

    temperature_fd = open(DEVICE_DHT22, O_RDONLY);
    if (temperature_fd < 0) {
        perror("can not open device");
        exit(1);
    }

    ret = read(temperature_fd, buf, sizeof(buf));
    if (ret < 0) {
        printf("read err!\n");
    } else {
        tempz = buf[2];
        tempx = buf[3];
        printf("temperature! = %d.%d\n", tempz, tempx);
    }

    close(temperature_fd);
    return tempz;
}
```

```c
int MQ7() {
    int testfd;
    int ret = 0;
    unsigned char buf[2];

    testfd = open(DEVICE_BLTEST, O_RDONLY);
    if(testfd < 0)
    {
        perror("can not open device");
        exit(1);
    }
        ret = read(testfd,buf,sizeof(buf));
        if(ret < 0)
        {
            printf("read err!\n");
        }
        else
        {
            coVal = buf[1];
            printf("CO1! = %d\n", coVal);
        }
    if (testfd >= 0)
    {
        close(testfd);
    }

    return coVal;
}
```

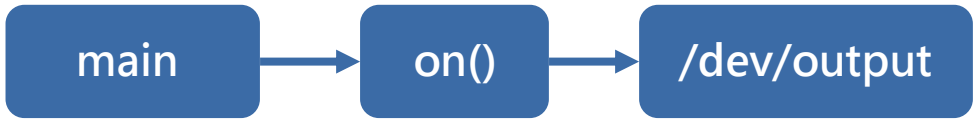
火災警報系統 | System - main program

楊家棠

3. 開關蜂鳴器 / LED

設定蜂鳴器(HW-508)及 LED 的 ON / OFF 函數:

Open dev/output ; Write : “1”



Write : “0” ; Close dev/output


```c
// Buzzer ON
void hw508_on(void) {
    if (buzzer_fd < 0) {
        buzzer_fd = open(DEVICE_HW508, O_WRON
        if (buzzer_fd < 0) {
            perror("can not open device");
            exit(1);
    if (write(buzzer_fd, "1", 1) < 0) {
        perror("write failed");
        close(buzzer_fd);
        buzzer_fd = -1;
        exit(1);
```

```c
// LED ON
void whiteLed_on(void) {
    if (whiteLed_fd < 0) {
        whiteLed_fd = open(DEVICE_WHITELED, O_WR
        if (whiteLed_fd < 0) {
            perror("can not open device");
            exit(1);
    if (write(whiteLed_fd, "1", 1) < 0) {
        perror("write failed");
        close(whiteLed_fd);
        whiteLed_fd = -1;
        exit(1);
```

```c
// Buzzer OFF
void hw508_off(void) {
    if (buzzer_fd >= 0) {
        if (write(buzzer_fd, "0", 1) < 0) {
            perror("write failed");
            close(buzzer_fd);
            buzzer_fd = -1;
            exit(1);
        }
        close(buzzer_fd);
        buzzer_fd = -1;  // not open
```

```c
// LED OFF
void whiteLed_off(void) {
    if (whiteLed_fd >= 0) {
        if (write(whiteLed_fd, "0", 1) < 0) {
            perror("write failed");
            close(whiteLed_fd);
            whiteLed_fd = -1;
            exit(1);
        }
        close(whiteLed_fd);
        whiteLed_fd = -1;  // not open
```

火災警報系統 | System - main program

楊家棠

4. 建立中斷

在測試階段，當 main 程式執行時被關閉，若 GPIO 仍處於高電平狀態，可能會導致蜂鳴器和 LED **仍持續開啟**。為了確保這些設備在程式結束時能夠正確關閉，因此加入中斷處理來**調用 OFF 函數**，以便在程式退出時自動關閉蜂鳴器和 LED。

5. 建立共享記憶體

由於樹莓派攝影機採用指令(system)來進行拍攝，執行時會**阻塞**其他函數的進行，因此將攝影機操作放入副程式中，並通過共享記憶體來**傳遞參數**。這麼做能夠避免阻塞主程式，確保程式的其他部分能繼續運行。

```
// interrupt (Ctrl+c)
void handle_signal(int signal) {
    if (signal == SIGINT) {
        hw508_off();
        whiteLed_off();
        exit(0);
    }
}

int main() {
//Shared Memory set
int shmid = shmget(SHM_KEY, ARRAY_SIZE * sizeof(int), 0644 | IPC_CREAT);
if (shmid == -1) {
    printf("shmget failed1");
    exit(1);
}

//Shared Memory put
shm_flag = (unsigned char *)shmat(shmid, NULL, 0);
if (shm_flag == (unsigned char *)(-1)) {
    printf("shmat failed2\n");
        exit(1);
}

signal(SIGINT, handle_signal);
```

火災警報系統 | System - main program

楊家棠

6. 判斷是否發生火災

若同時間，DHT22 感測溫度**超過 45 ℃**及 MQ-7 感測一氧化碳值**超過 120** 時，便會發布警報並啟動蜂鳴器及 LED，並設定共享記憶體變數陣列 (shm_flag) 遞給正在執行中的 camera 程式，camera 程式獲取該變數後將立即啟動樹莓派攝影機機進行拍攝。

而感測溫度或一氧化碳值達到閾值時，便啟動計數器，計數器每秒增加一次，**超過 30 次**便會發布警報並啟動相關設備，反之溫度及一氧化碳低於閾值時，計數器將歸零並根據條件重新記數。

```
//Meet the standard
if (tempz > 45 && coVal > 120 || counter_dht11 > 30 || counter_MQ7 > 30) {
    shm_flag[2] = 1;
    if (current_time - last_time_hw508 >= 250) {  // Buzzer frequency 250 ms
        if (buzzer_flag) {
            hw508_off();  // Buzzer OFF
        } else {
            hw508_on();   // Buzzer ON
        }
        buzzer_flag = !buzzer_flag;
        last_time_hw508 = current_time;
```

```
if (current_time - last_time_dht11_std >= 1000){  //1s
    if (tempz > 45) {
        counter_dht11++;
        // printf("counter_dht11 = %d\n", counter_dht11);
        last_time_dht11_std = current_time;
    }
    else {
        counter_dht11 = 0;
```

```
if (current_time - last_time_MQ7_std >= 1000){  //1s
    if (coVal > 120) {
        counter_MQ7++;
        // printf("counter_MQ7 = %d\n", counter_MQ7);
        last_time_MQ7_std = current_time;
    }
    else {
        counter_MQ7 = 0;
```

火災警報系統 | System - camera program

楊家棠

1. 共享記憶體參數 & 執行拍攝

a. 設定與讀取參數：

設定共享記憶體以獲取溫度和一氧化碳數據。
定期將這些數據寫入 CSV 檔。

b. 解決 CSV & 拍攝衝突：

由於樹莓派攝影機的拍攝指令與 CSV 檔寫入會發生衝突，因此將 CSV 檔案寫入功能整合至 camera 副程式中。

c. 警報處理：

當溫度或一氧化碳超過安全範圍時，觸發警報，執行拍攝指令。
拍攝完成後，將拍攝旗標（camera_write）寫入 CSV 檔案，通知伺服器定時上傳指定路徑的影像檔案。

```
int main() {
    //Shared Memory set
    int shmid = shmget(SHM_KEY, ARRAY_SIZE * sizeof(int), 0644 );
    if (shmid == -1) {
        perror("shmget failed");
        exit(1);

    //Shared Memory get
    shm_flag = (unsigned char *)shmat(shmid, NULL, 0);
    if (shm_flag == (unsigned char *)(-1)) {
        perror("shmat failed\n");
            exit(1);

    while (1) {
        tempz = shm_flag[0];
        coVal = shm_flag[1];
        dht22_write();
        MQ7_write();
        if (shm_flag[2]) {  // temperature > 45 & coVal > 60

        camera_capture();
        printf("Camera capture! flag = %d\n", shm_flag[2]);
        camera_write(1);
        sleep(1);  // 1s
```

火災警報系統 | System - camera program

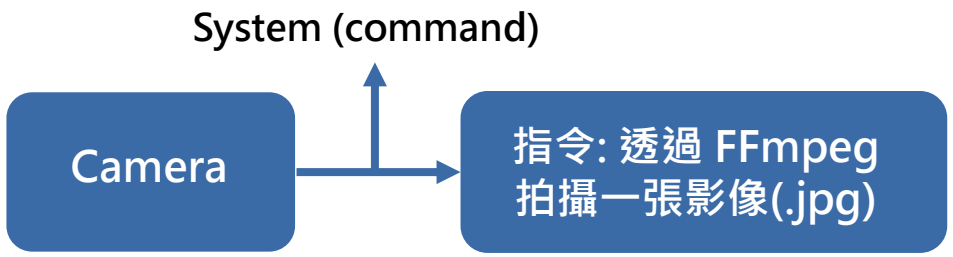
楊家棠

2. 攝影機拍攝程式碼

使用 system 函數來執行相機拍攝指令，在指令內透過 FFmpeg 開源軟體來調用樹莓派相機(dev/video0)拍攝一張解析度為 640x480 的jpg 影像檔(覆蓋)，並存放於指定路徑。

指令執行時間約為1秒。

```
void camera_capture() {
    int status;
    status = system("ffmpeg -f video4linux2 -i /dev/
    video0 -s 640x480 -vframes 1 -loglevel quiet
    -y ../excute/node_js/public/alert_image.jpg");
    if (status == -1){
        printf("Camera failed !!!");
    }
}
```



火災警報系統 | System - camera program

楊家棠

3. 將資料寫入 CSV 檔

camera 副程式執行後，會分別寫入不同的 CSV 檔以便 Server 獲取資料並將結果傳送至前端網頁。


```c
int dht22_write(void) {
    FILE *file;
    file = fopen("../excute/node_js/Data/temp_output.csv", "w");
```

```c
int MQ7_write(void) {
    FILE *file;
    file = fopen("../excute/node_js/Data/CO1_output.csv", "w");
```

```c
int camera_write(int a) {
    FILE *file;
    file = fopen("../excute/node_js/Data/camera_output.csv", "w")
```

結構

張晉銘

MQ-7 ➡ ADC 0804 ➡ dev/mq7 ➡ mq7.ko ➡ main ➡ Index.html

感測器本體，負責依照ppm濃度不同輸出0-255的數值

類比訊號轉為數位訊號可供判斷

建立設備檔，使此設備可以被module使用

掛載本module，使其可以將資料從kernel傳至user space

主要執行程式，藉由判斷不同數據決定要採取不同行動

將main輸出之資料傳至瀏覽器上供他人觀察

火災警報系統

張晉銘

MQ-7 一氧化碳感測器

MQ-7 常用於家庭、環境的一氧化碳偵測裝置，主要探測一氧化碳和煤氣。

此產品主要是利用二氧化錫(SnO2)在空氣中電導率較低的特性，當空氣中的一氧化碳濃度增高時，感測器中的電導率也會隨著濃度增高而變高，按照電導率的變化，轉換為與該氣體濃度相對應的輸出信號。

火災警報系統

張晉銘

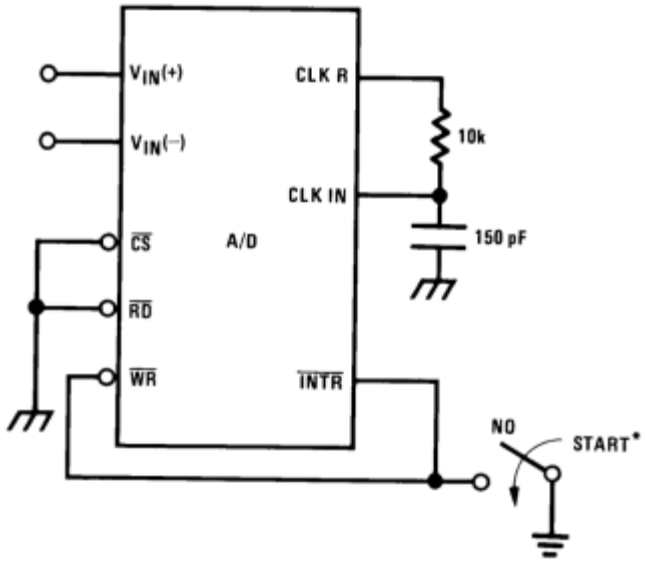
MQ-7 一氧化碳感測器

主要輸出方式，有分為 AO 和 DO 兩種，前者為 0-5V 類比(analog)輸出, 對應感測到的一氧化碳濃度，濃度越高電壓越高。

後者則是數位(digital)輸出，+5V 為 true, 代表一氧化碳氣體濃度過高觸發警報，0V 為 false, 代表正常。

火災警報系統

張晉銘



*After power up, a momentary grounding of the WR input is needed to ensure operation.

Figure 59. Self-Clocking in Free-Running Mode

ADC 0804

因為 Raspberry pi 中並無內建 ADC 轉換功能，因此無法直接將 AO 輸出到其中進行資料處理，因此需要另外將 AO 訊號輸入 ADC，由 ADC 轉為數位訊號之後再輸出至Raspberry pi，而接線的部分參考了規格書上的 Free – Running 接法，但 WR 和 INTR 的腳位則是由樹莓派控制取代原本開關的作用。

火災警報系統

張晉銘

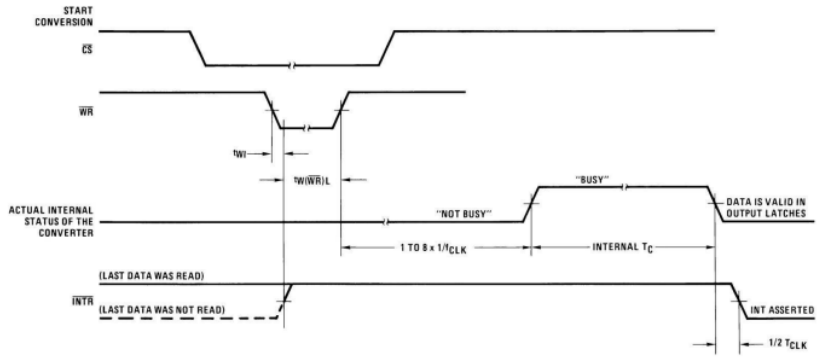
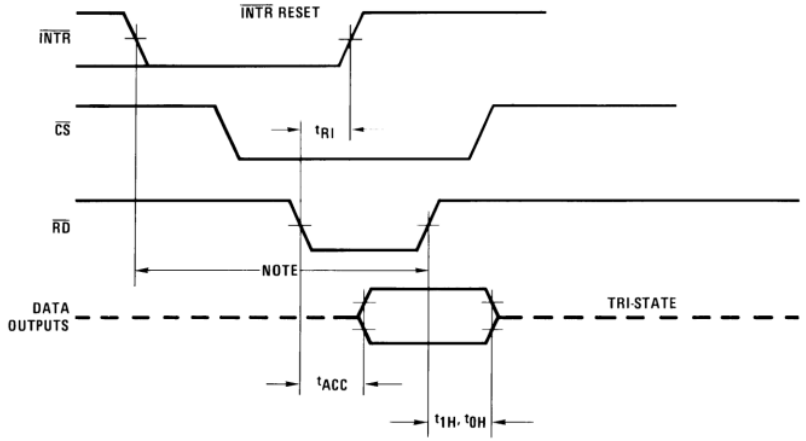


Figure 1. Start Conversion



Note: Read strobe must occur 8 clock periods ($8/f_{CLK}$) after assertion of interrupt to specify reset of $\overline{INTR}$.

Figure 2. Output Enable and Reset With $\overline{INTR}$

```
void read_data(void)
{

    gpio_out(WR,0);
        udelay(200);

    while(!gpio_in(INTR));

    buf[1] |= gpio_in(DB0) << 0;
    buf[1] |= gpio_in(DB1) << 1;
    buf[1] |= gpio_in(DB2) << 2;
    buf[1] |= gpio_in(DB3) << 3;
    buf[1] |= gpio_in(DB4) << 4;
    buf[1] |= gpio_in(DB5) << 5;
    buf[1] |= gpio_in(DB6) << 6;
    buf[1] |= gpio_in(DB7) << 7;


    gpio_out(WR,1);
```

火災警報系統

張晉銘

Node.js / html


```
function loadCoNumber(){
    $.ajax({
        url:"http://192.168.137.10:8181/watch/CO1",
        type:"GET",
        success:function(data){
            showall(data,"co1No");
            }
    });
}

function loadtempNumber(){
    $.ajax({
        url:"http://192.168.137.10:8181/watch/temp",
        type:"GET",
        success:function(data){
            showall(data,"tempNo");
            }
    });
}

function updateflag(){
    $.ajax({
        url:"http://192.168.137.10:8181/watch/camera",
        type:"GET",
        success:function(data){
            updateImage(data)
            }
    });
}
```

```
document.addEventListener("DOMContentLoaded", () =>{
    loadCoNumber();
    loadtempNumber();
    setTimeout(()=>{
        refresh();
    },1000)
});


function refresh(){
setInterval(loadCoNumber , 2000);
setInterval(loadtempNumber , 2000);
setInterval(updateflag ,1050);
}
```

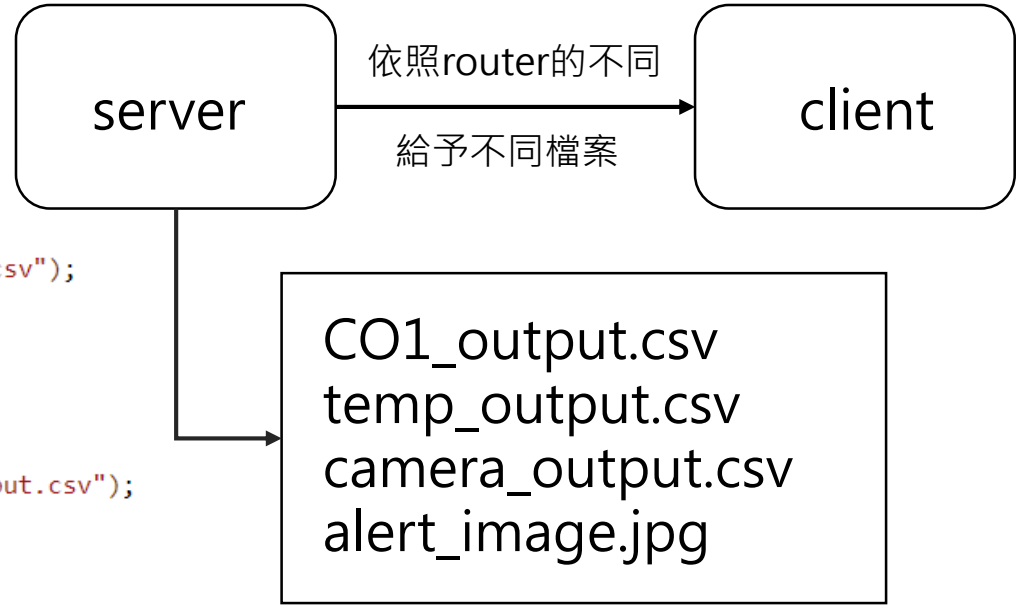
火災警報系統

Node.js / html

張晉銘

```
app.get('/watch/CO1',(req,res)=>{
    res.send(json);
});

app.get('/watch/temp',(req,res)=>{
    res.send(jsontemp);
});

app.get('/watch/camera',(req,res)=>{
    res.send(jsoncamera);
});

app.get('/alert_image.jpg',(req,res) => {
    res.sendFile(path.join(__dirname,'public','alert_image.jpg'))
});
```

```
app.use((req,res,next)=>{
    json=csvToJson.fieldDelimiter(',').getJsonFromCsv("./Data/CO1_output.csv");
    res.setHeader('Access-Control-Allow-Origin','*');
    next();
});

app.use((req,res,next)=>{
    jsontemp=csvToJson.fieldDelimiter(',').getJsonFromCsv("./Data/temp_output.csv");
    res.setHeader('Access-Control-Allow-Origin','*');
    next();
});

app.use((req,res,next)=>{
    jsoncamera=csvToJson.fieldDelimiter(',').getJsonFromCsv("./Data/camera_output.csv");
    res.setHeader('Access-Control-Allow-Origin','*');
    next();
});
```




楊家棠

張晉銘

Document

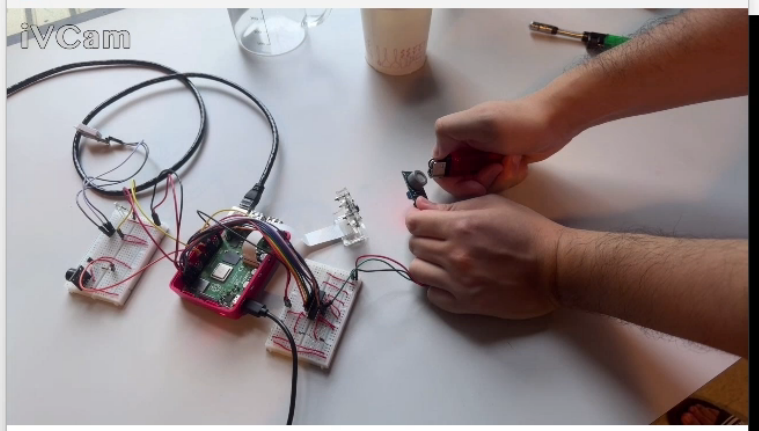
不安全 192.168.137.10:8181

Gmail YouTube 地圖

火災警報系統

一氧化碳指標	25	溫度(℃)	29

e2eSoft iVCam



iPhone 已連接

相機尚未開啟

Fire Alarm System Video

火災警報系統

 楊家棠  張晉銘

預計最佳化項目

1. ADC 改用其他元件以減少腳位占用。
2. 透過 Thread 來解決攝影機拍攝指令的阻塞問題。
3. 樹莓派攝影機從單張影像傳輸變更為透過 RTSP 進行影像串流。
4. 有緣蜂鳴器變更為無緣蜂鳴器，可隨情況調整鳴聲頻率。
5. 透過 APP(Android, iOS) 進行監控而非使用網頁瀏覽器。
6. 火災發生後 3 分鐘內仍持續響起警報，系統將會自動播打消防專線。
7. 將產品硬體拆分為 Server 端及 Function 端，兩者間透過 socket 溝通。
8. CSV 檔進行整合。

火災警報系統

 楊家棠  張晉銘

開發瓶頸&解決方案

1. Camera 執行拍攝時，因執行命令導致其他進程受到阻塞
   解決方案: 另外建立副程式來專門處理拍攝指令

2. Camera 執行拍攝時，會與寫入 CSV 檔產生衝突
   解決方案: 將寫入 CSV 檔程式放入副程式並與拍攝指令依順序執行

3. DHT22 讀取錯誤值問題
   解決方案: 更換硬體進行交叉比對

4. ADC 開發問題
   解決方案: 透過交叉比對以及控制變量達成硬軟體的檢查

5. 組員於不同 Linux 版本上進行開發導致 Node.js 安裝失敗
   解決方案: 使用 nvm 取代 apt install 以規避 GCC 版本過舊問題

火災警報系統

 楊家棠  張晉銘

Github QR Code