**111下數位科技專案實作**

**期末報告**

109033132 動機系 李臻茵

2023.07.13

**Contents**

1. Idea generation & Product definition ………………………………………………………………….3
2. Hardware Structure ……………………………………………………………………………………. 3
3. Software technology …………………………………………………………………………………… 4
   1. Version 1 ..……………………………………………………………………………………….. 4
      1. Architecture……………………………………………………………………………….. 4
      2. Key Details ……………………………………………………………………………….. 5
   2. Version 2 ………………..……………………………………………………..…………………. 6
      1. Key Details ……………………………………………………………………………….. 6
      2. Incomplete Verification ……………………………………………………..…………… 6
4. Future Work …………………………………………………………………………………..………… 7
5. References……………………………………………………………………………………………

\*Github link: <https://github.com/teresasa0731/Home_system_HOST_esp32>

1. **Idea generation & Product definition**

透過這次修課的機會，希望可以網路傳輸協定與物聯網技術有基本的了解，故選擇學習開發一套居家系統，而這次的子專案則從建立整個系統的控制中心 System Host開始，配合基本的三個功能來架設，分別為：

1. Control：以LED燈呈現對開關的控制訊號輸出。
2. Monitor：分別以Switch Button+LED燈呈現靜態訪問的訊號輸入；以HC-SR501來呈現及時狀態的偵測。
3. Live Streaming

為了將問題模型化，並將焦點放在網路的連線溝通上，以上三個功能都以最基本的元件模擬；同時為了將來能夠對系統有完整的控制，包括記憶體分配與特殊硬體模式等的設計，因此選擇直接使用ESP-IDF (Espressif IoT Development Framework) 做為此次ESP32硬件開發的軟體環境。

1. **Hardware Structure**

|  |  |
| --- | --- |
| **HC-SR501**   * Fig 1. Hardware circuit wiring diagram[[1]](#footnote-1) | * Fig 2-1. HC-SR501 with mask[[2]](#footnote-2)   一張含有 電子工程, 網球, 球拍, 室內 的圖片  自動產生的描述   * Fig 2-2. Hardware circuit wiring diagram |

1. **Software technology**

前提及此次專題的重點是在學習網路傳輸的理論背景與應用，故在研究過由esp32的開發商espressif所提供的編譯環境ESP-IDF與範例後，先由單向訪問的web server技術開始建立(ver.1)，完成對周邊元件訊號的邏輯判斷後，加上HTML文本來將其架設成網站形式，開始透過網站做訪問。此版本中遇到的幾個瓶頸：

1. HTML文本難以維護，因此階段的按鈕指令發送給伺服器是透過連接至不同位址做發送，但因網域時常更動而需更改文本內容，而此文本為內嵌於主程式中，故每次更新都須重新編譯(build)及燒錄(flash)，故在下一版加入SPIFFS檔案系統技術。
2. 感測模組的訊號有即時性問題，這裡的做法是每100ms做輪詢，其過度消耗CPU的資源，故二版使用web socket的方式架構雙向溝通的伺服器。

**3.1 version 1**

將硬體端esp32當static web server，由web(client 端)對伺服器發起 HTTP request進行單向訪問，伺服器經由網路接受HTTP請求後，透過HTTP協定傳送回復給客戶端。

**3.1.1 Architecture**

**一張含有 圖表, 文字, 方案, 工程製圖 的圖片

自動產生的描述**

* + - Fig 3-1. Program Flow chart[[3]](#footnote-3)

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述 ◀ Fig 3-2. Server-Client connection

**3.1.2 Key Details**

1. Multitasking

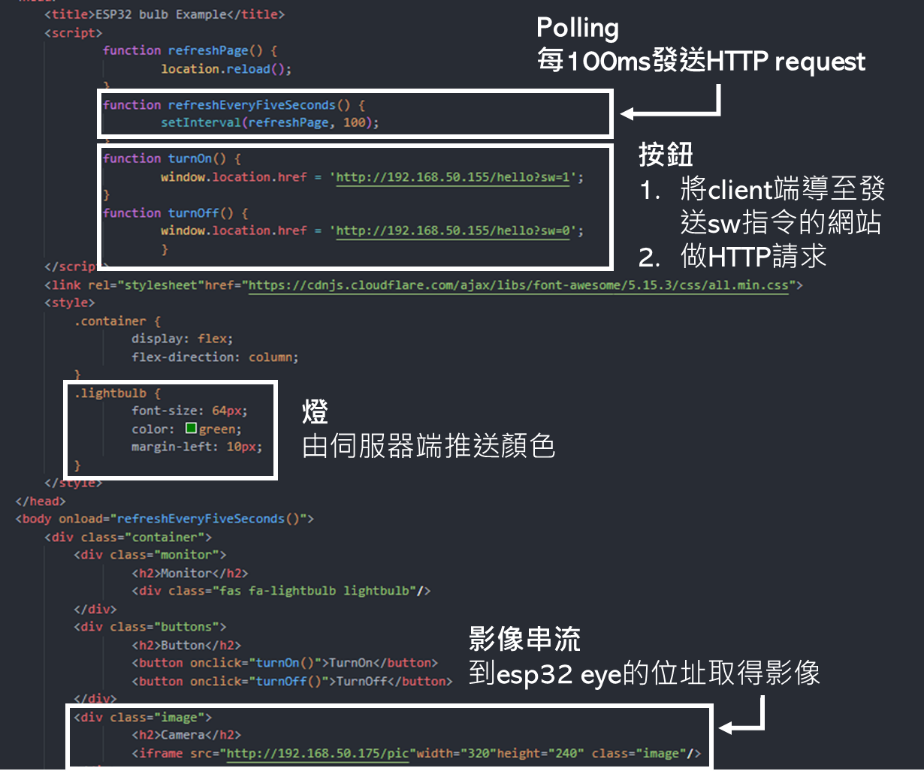
因當CPU進入server狀態後可視為在主程式架設好參數後，到function中服務，故為了同時能接受人體感測模組的訊號，對其做multitasking來分割事件。



1. HTML文本

網站的部分希望能對前提及的三個功能提供資訊參考：

|  |  |
| --- | --- |
|  | 1. 人體感測模組狀態   在硬體上以綠燈做為訊號顯示   1. 透過網站按鈕控制硬體狀態   此處為控制紅燈做訊號顯示   1. 來自esp32eye的影像資訊[[4]](#footnote-4)   目前先以照片的形式取代，因其易過熱  ◀Fig 4. Website |

 ◀ Fig 4. Ver1.html[[5]](#footnote-5)

**3.2 version 2**

改成使用 web socket + SPIFFS的模式，除了將發送文檔打包而非內嵌於程式碼，在對HTML文本進行優化時能獨立處理，也實現client與server的雙向溝通機制，可以即時的交換資訊，不用等待client端發出請求，或使用佔用資源的輪詢方式來實現，同時因Handshake後會保持連線狀態，所以每次通訊時所發出的請求就可省略掉一些關於連線狀態的資訊，提高連線效率。

**3.2.1 Key Details**

1. SPIFFS

一張含有 文字, 螢幕擷取畫面, 字型, 陳列 的圖片

自動產生的描述 一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

可以看到在partition table中，相較於舊版多要了一塊flash的空間作為SPIFFS的儲存位置(紅框處)，就可將推送資訊放於額外的空間，在實作時也不需重新編譯，只需燒錄更動的檔案即可(以此次專案而言即可以單獨優化HTML文本)。

1. web socket vs web server

在功能2(monitor)中，舊版的觸發方式來自於client端頻繁的發送請求，同時因監測的功能是分割在task2(前所提及的multitasking)，故當client發送請求時，其實並非感測器當下的狀態，加上此次選擇的感測器訊號並非非常穩定，故在此部份是第一版比較缺失的地方；但透過web socket的方式，可以透過訊號觸發來讓伺服器端向用戶端推送資訊，且舊版若是在不同平台上操作，儘管接收同一伺服器的資訊，在未更新網頁前是不會收到最新的狀態的，改成用web socket後就可以同步刷新網站資訊。

**3.2.2 Incomplete Verification[[6]](#footnote-6)**

因功能2的推送觸發仍在修正，故舊第二版預期能達到的目標而言，剩伺服器端在某些條件下(此專案即為當感測器觸發時要改變網頁端綠燈的狀態)主動發送資訊此功能未完善，其他諸如獨立HTML文本、建立web socket連線等皆已完成。

1. **Future work**
2. 完善網頁架構

考量到未來欲監控的電器，須對網頁做動態調整。

目前還無法將esp32 eye的影像同步在web socket的網頁上更新，僅ver.1能以polling詢問esp32 eye伺服器的影像，故仍需再研究。

1. 硬件方面修正與設計

完成ver.2後即可開始時做硬體的部分，大致如如何將市電控制與開發板結合、設計相應應用場景的模式等。

1. 影像處理

學習影像辨識，設計並取得所需資訊(ex.人體偵測/環境辨識等)，與居家系統的應用情境結合

1. 修課同學提及之感測模組仍在測試中，尚未能展示。 [↑](#footnote-ref-1)
2. 透過增加錐狀遮罩(暫時以紙捲代替)來提高人體感測模組的指向性。 [↑](#footnote-ref-2)
3. 主程式碼請參照資料夾 systemHost\_Ver1/main.c [↑](#footnote-ref-3)
4. esp32 eye 程式來源為官方提供之範本(<https://github.com/espressif/esp-iot-solution>)，可見資料夾pc\_server/ [↑](#footnote-ref-4)
5. 可見資料夾 systemHost\_Ver1/ ver1.html [↑](#footnote-ref-5)
6. 此版本因web socket的程式推送仍在修正，目前1.5版本較為完善(可見資料夾systemHost\_Ver1.5/) [↑](#footnote-ref-6)