中華民國第 65 屆中小學科學展覽會

作品說明書

高級中等學校組　工程學科(一)

052306

智慧火場定位與危機應對系統

學校名稱： 新北市立新北高級中學

作者：	指導老師：
高二 洪睿妍 高三 張益誠 高三 連季宏	謝進生 蕭宇軒

關鍵詞： UWB、火場定位、危機地圖

摘要

本研究建構一套具即時定位、風險感知與智慧路徑規劃功能的火場應變系統，以提升緊急情境下的人員生存率與消防調度效率。針對火場中濃煙遮蔽與視線受限等問題，系統整合 UWB 無線定位技術、可拋式多感測探測球與自建危險係數模型，突破既有應變限制。探測球可即時回傳火場中氫氣、一氧化碳、瓦斯、粉塵與溫度等感測數據，並依據 IDLH 標準轉換為風險係數，生成即時危機地圖。

系統進一步結合風險模型與 Dijkstra 演算法進行加權路徑搜尋，避開高風險區域，提供最安全之撤離與救援路徑。資料處理採用 SQLite 本地儲存並搭配視覺化模組呈現，經模擬場景實測後，可即時更新環境資訊與路徑結果，展現高度準確性與實用性，未來可作為高風險場域智慧應變之技術基礎。

壹、前言

一、研究動機

2024 年 5 月，新竹晴空匯火災造成兩位消防人員殉職，事發當時，由於濃煙瀰漫、視線受阻，兩人迷失方向，最終因氧氣耗盡未能獲救。雖然他們曾發出求救訊號，但因呼救聲未被聽見、無法準確掌握其位置，錯失黃金救援時機。此事件凸顯目前災害救援在即時定位與火場資訊回傳上有著重大問題，也反映現場指揮與資訊整合能力的不足。

為改善此一問題，本研究提出一套智慧型火場危機應對系統，結合 UWB 無線定位、可拋式多感測探測球、IDLH 風險模型與 Dijkstra 演算法。系統能即時蒐集火場中溫度、氣體濃度與粉塵數據，透過 AI 演算法建構危機熱區地圖，並提供最短與最安全的避險路徑建議，協助指揮官於高壓環境下迅速做出精確決策，提升救援效率並降低人員傷亡風險。

二、研究目的

(一) 分析不同環境與現有救援技術的限制。

(二) 整合 UWB 無線定位與多感測器之可拋式探測球，進行即時定位與火場數據蒐集。

(三) 建危險係數模型，做出風險分級與決策資訊。

(四) 開發危機地圖並測試其在逃生與救援路徑規劃的應用。

三、文獻回顧

根據 ETtoday 新聞網(2024 年 12 月 19 日)報導: 台中全聯生鮮倉儲發生火災，釀成九人死亡、七人受傷的悲劇。主因為施工過程中產生火花引燃可燃物，導致火勢迅速蔓延且難以控制。

根據 Yahoo 奇摩報導(2024 年 5 月 28 日): 新竹市某大樓因地下一樓電路短路引發火災，釋放大量濃煙，因樓梯間缺乏通風與排氣系統失靈，導致人員逃生困難。

根據內政部消防署 111 年全國火災統計:全國火災共發生 1 萬 5890 次，死亡人數高達 152 人。而在其中建築物火災發生 5512 次，造成 130 人死亡。由於建築物的特殊結構和複雜性，救援行動難度增加，傳統的救援方式往往無法有效應對這些複雜的火場環境。圖 1 統計了 110、111 年全國火災死亡案件的火災類型分布，數據顯示建築火災造成的死亡人數遠高於其他類型，顯示建築物火災的高風險性。

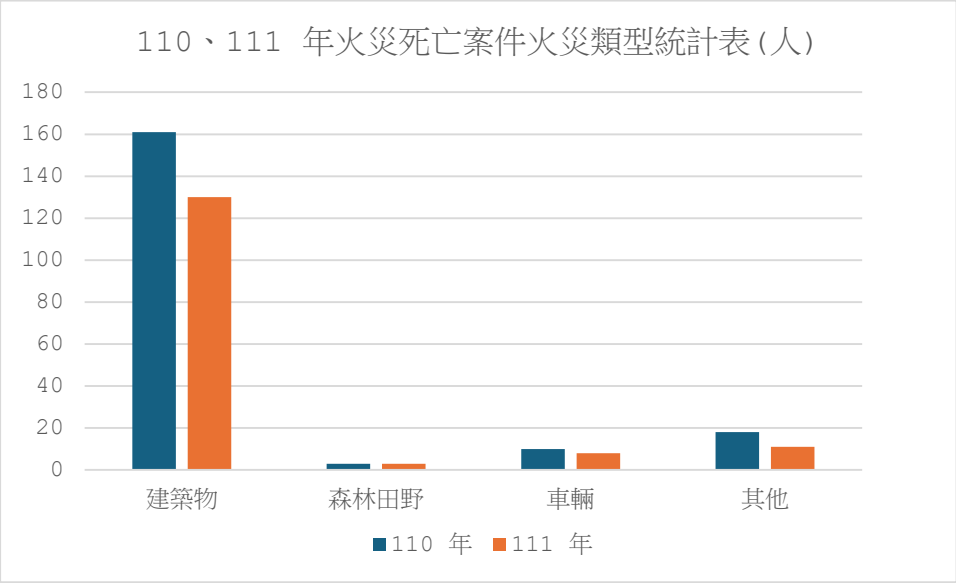


圖 1、 110、111 年火災死亡案件火災類型統計表(人) （由作者繪製）

根據內政部建築研究所(2020):在低樓層住宅中逃生路徑的單一性與避難設計的不完善，導致火災發生時救援人員救援受困人員的困難。許多老舊建築缺乏多方向的逃生出口，或因為鐵窗、狹窄樓梯等其他問題的阻礙，無法提供有效的救援。一旦火勢蔓延至出口，受困者無法迅速逃離，且消防隊在無法確定人員具體位置的情況下，容易錯過黃金救援時間。

根據內政部警政署(2023)：現行火災警報系統主要以固定式裝置為主，並以建築物內部煙霧和溫度感測器為主要偵測依據。然而此類系統多部署於建築內部，對於突發型、開放性或極端環境下的火災場域，則應變能力較為有限。

根據蘇韋倫(2022)指出：消防人員在火災現場執行救援任務時，面臨多種安全風險，如火場結構不穩定、濃煙與有毒氣體危害，以及設備故障等。這些風險不僅威脅救災人員的生命安全，也影響救援效率。為降低風險，許多研究強調建立完善的風險評估與管理系統，包含事前風險識別、現場監控與後續安全檢討。此外，透過訓練與標準作業程序，提高消防人員應對突發狀況的能力，也是減少意外的重要措施。結合歷史火災資料與風險管理策略，進一步發展火災預測模型，能有效協助指揮中心做出快速且準確的決策，提升整體消防救援的安全與效率 。

根據黃麗如(2023)指出:當建築物內發生火災，或因為消防控制不當而引發的人為災害時，常因逃生路線不清晰或標示不明確，導致人員無法及時疏散，進而危及人員生命安全。為了降低人員傷亡，逃生路線的規劃與火災控制已成為安全管理中的核心課題。根據災害發生頻率顯示，老舊建築的風險相對較高，特別是傳統建築的逃生通道仍有許多可改進之處。透過分析歷年火災的統計數據與特性，深入探討各項消防控制策略的成效，有助於優化逃生動線設計。結合現行法規與實際案例，更能凸顯清晰逃生路線與完善火災控制對於減少人命傷亡與財產損失的重要性。

目前的救災技術在定位受困人員時存在多項缺點，主要包括定位精度不足、訊號受阻、環境適應性差等問題。例如，GPS 技術在室內或地下空間常因訊號衰減而失效，而其他基於無線電或視覺的技術則可能受到濃煙、瓦礫或惡劣天氣的影響，導致搜救行動延誤。此外，救援人員需冒險進入危險區域進行手動搜尋，增加了生命安全的風險。

本研究提出結合 UWB 無線定位與 AI 模型，提升火場中受困人員的定位精度，確保即時掌握準確位置，縮短救援時間、提升效率，並降低救援人員暴露於高風險環境的機率

貳、研究設備及器材

表 1、研究設備與器材

名稱	規格	數量
樹梅派	Raspberry Pi4ModelB	1
粉塵感測器	LMS5003	16
瓦斯感測器	MQ-5	16
溫度感測器	LM35	16
氫氣感測器	MQ-8	16
一氧化碳感測器	MQ-9	16
開發版	ESP-WROOM32	16
室內型基站	Makerfabs ESP32 UWB	4
定位標籤	Makerfabs ESP32 UWB	1

參、研究過程與討論

本研究所有實驗相關相片都由作者所拍攝和繪製

一、初步探討系統運作之功能與流程

經過文獻探討和初步討論後，我們分析目前救援可能會面臨到的問題，並擬定相關計畫來改善。本系統初步規劃運作過程如下:

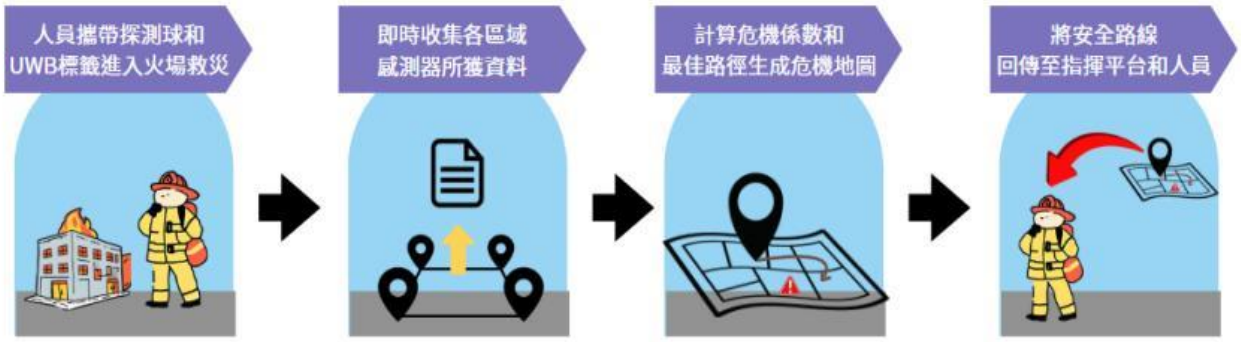


圖 2、初步系統運作流程（由作者使用 Canva 繪製）

二、研究架構與流程

根據上述的運作流程發想，我們將研究主幹分為四部份進行如下圖 3:A. 探測球製作及判斷 B. 地形探勘與繪製 C. UWB 無線定位系統與資料收集 D. 整合系統繪製危機地圖，最後再將系統進行修正，以完善運用智慧火場定位與危機應對系統。

A .探測球製作及判斷

我們將文獻討所發現的氣體查找各氣體的範圍來建立火災發生時產生危險氣體的資料庫。原本我們有嘗試過用監督式學習的方式來訓練閃燃模型，但發現需要大量實驗數據，而且在火場這種需要快速反應的情境下不夠即時。因此後來我們改用美國 NIOSH 公布的 IDLH 標準來將每種氣體的濃度轉換成危機係數 (0～9 級)。

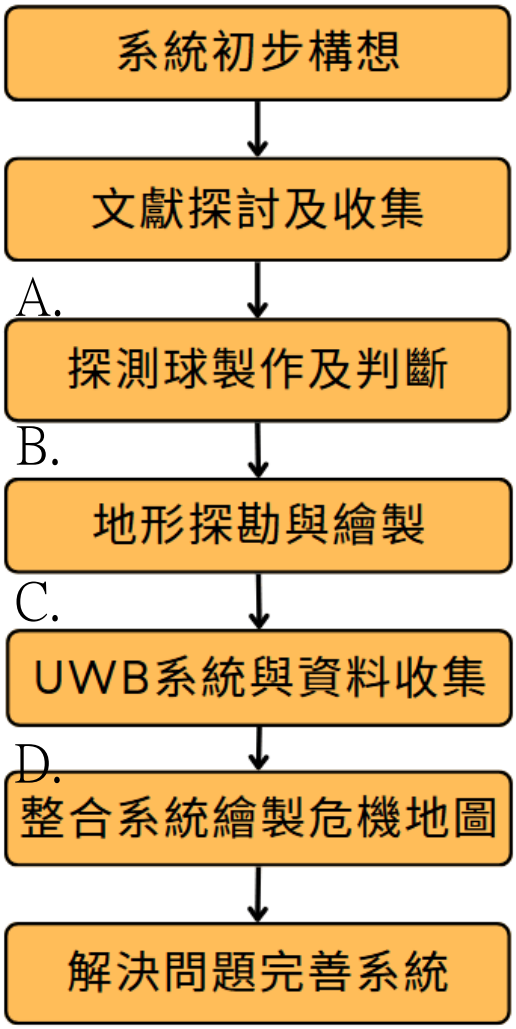


圖 3、研究流程圖（由作者使用 Canva 繪製）

B. 地形探勘與繪製

在此部分我們以已知地形為前提進行操作，當在事前我們就已得知火場內部地形結構，我們會將其地形圖匯入 UE4 中進行建模，以進行後續操作。

C. UWB 無線定位系統與資料收集

我們透過 UWB 無線定位來定位人員及感測器所在位置，再透過探測球或固定式感測器獲取火場內部資料。

D .整合系統繪製危機地圖

我們將感測器的資料透過危險係數模型判斷後，將判別結果依定位匯至地圖中，形成危機地圖，再根據現場人員的定位，讓系統透過最佳救災路徑模型規劃最安全且最短的救災路徑，以提高救災的安全性及效率。

三、探測球製作

(一) 初代球體外觀設計與製作

我們首先利用 tinkercad 進行 3D 建模設計出球體的外觀，然後利用雷射裁切木塊並組合，最後在球體表面安裝了 ESP32、UWB 標籤，以及多個感測器，包括溫度感測器、一氧化碳感測器、氫氣感測器……等，來完成我們的探測球。

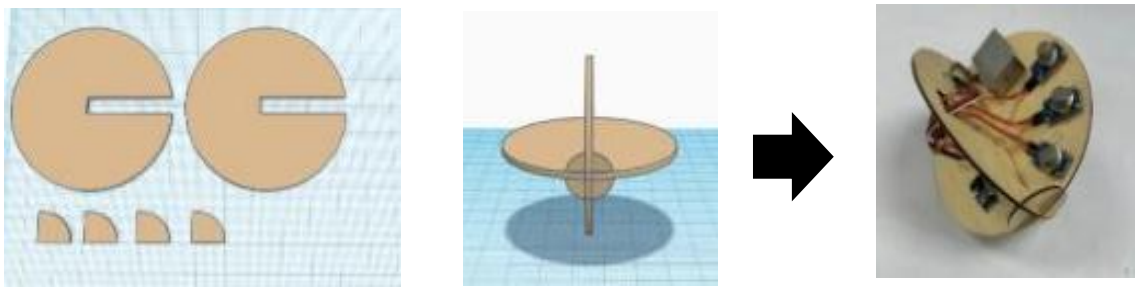


圖 4、球體設計及成品圖（由作者使用 tinkercad 建模和拍攝）

(二) 球體投擲實驗

原先我們使用木板進行雷切設計球體外框進行組裝，然而我們進行實際投擲時發現其偵測數值會因投擲時的旋轉造成數值的偏差

表 2、第一代探測球(如圖 4)測得數值表

狀態	二氧化碳	氫氣	瓦斯	一氧化碳	懸浮微粒	溫度
未投擲時	40.60	2.31	0.03	0.88	4.00	25.00
投擲後 1 秒	33.80	1.00	0.00	0.36	1.00	25.00
投擲後 2 秒	34.70	1.23	0.00	0.42	1.00	25.00
投擲後 3 秒	36.30	1.59	0.01	0.57	20.0	25.00
投擲後 4 秒	38.40	1.75	0.01	0.69	3.00	25.00
投擲後 5 秒	39.90	2.08	0.02	0.74	3.00	25.00
落地	40.50	2.31	0.02	0.87	4.00	25.00

我們經過詢問老師及查找資料後發現以下問題:

1. 球體採兩面相接設計，使得旋轉時球體的面會因面的作用力，而將周圍氣體排開
2. 感測器裝設在同一面造成感測器在旋轉時偵測範圍及方向單一

探測球經改良後，我們重測實驗如下表:

表 3、第二代探測球(如圖 7)測得數值

狀態	二氧化碳	氫氣	瓦斯	一氧化碳	懸浮微粒	溫度
未投擲時	40.40	2.17	0.02	0.85	3.00	26.00
投擲後 1 秒	40.30	2.17	0.02	0.85	3.00	26.00
投擲後 2 秒	40.30	2.16	0.02	0.85	3.00	26.00
投擲後 3 秒	40.40	2.17	0.02	0.86	2.00	26.00
投擲後 4 秒	40.40	2.18	0.01	0.86	2.00	26.00
投擲後 5 秒	40.40	2.17	0.02	0.85	3.00	26.00
落地	40.3	2.17	0.02	0.85	3	26

圖 5 實驗畫面

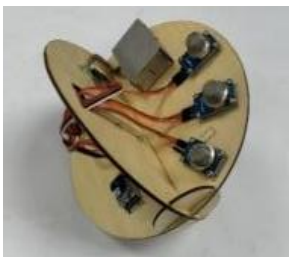


圖 6、第一代探測球

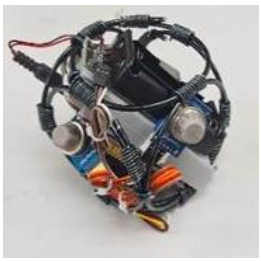


圖 7、第二代探測球

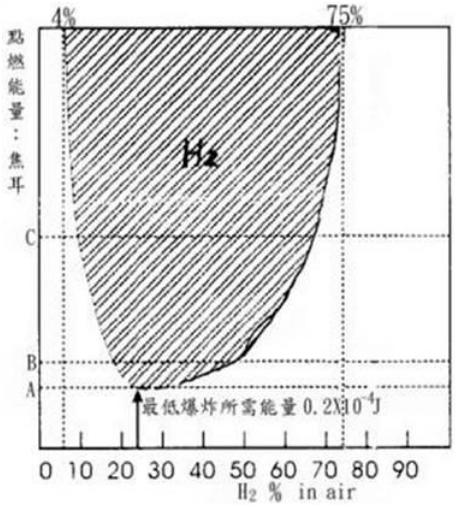
圖 8、第三代探測球

(皆由作者自行拍攝)

(三) 危險係數判斷模型演進

1. 初始閃燃模型判斷

原先在閃燃判斷中，我們會根據參考文獻(燃燒化學)中的爆炸範圍數據做為我們監督式學習的資料集，我們一共製作 500 筆資料集，並以訓練集和測試集 8:2 的比例進行訓練。



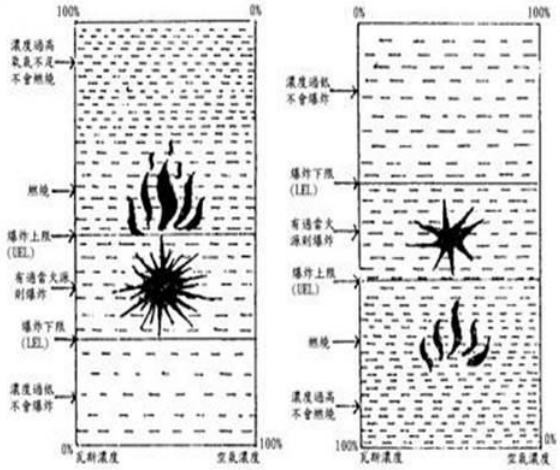
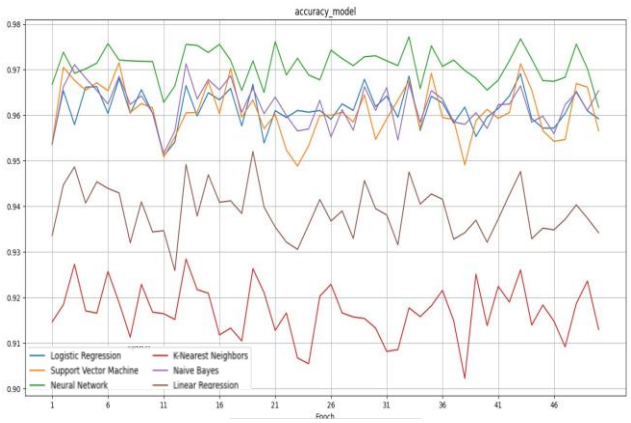


圖 9、氫氣之爆炸界限 圖 10、易燃瓦斯－空氣混合物之密度與爆炸難易之關係

(皆取自於燃燒化學)

接著我們使用機器學習的方式製作模型，我們測試了六種監督式學習方法，發現使用神經網路做出的召回率最高約為 95%。



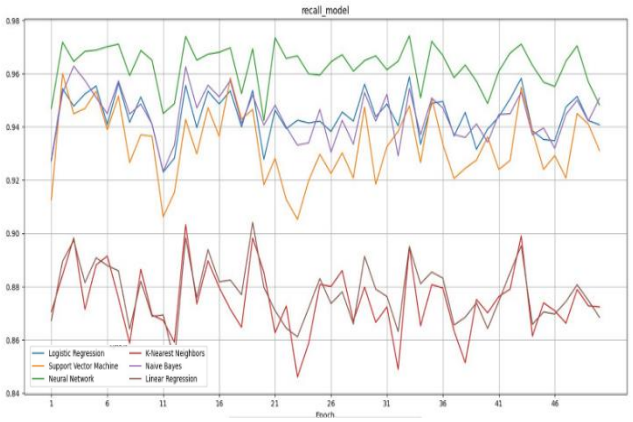


圖 11、各種機器學習準確率圖　　圖 12、各種機器學習召回率圖

(圖 11、12 皆由作者繪製)

為驗證模型實際預測能力，我們製作存有各種氣體比例的溫室箱，搭配各種溫度，並經等比例放大 10 倍以下實驗所偵測到的數值，用來模擬閃燃發生範圍，並判斷其預測的準確性。我們將溫度設定於 200℃ 至 560℃ 區間，並調整氧氣濃度至四種比例，模擬不同情境下的閃燃機率。

表 4、閃燃判斷實測實驗數據

固定氫氣濃度15%			固定瓦斯濃度15%			固定粉塵濃度20%			固定溫度360度			固定溫度360度			固定溫度360度		
溫度	實際	AI判斷	溫度	實際	AI判斷	溫度	實際	AI判斷	氫氣濃度	實際	AI判斷	瓦斯濃度	實際	AI判斷	粉塵濃度	實際	AI判斷
200	X	X	200	X	X	200	X	X	20	X	X	20	X	X	10	X	X
240	X	X	240	X	X	240	X	X	21	X	X	21	X	X	11	X	X
280	X	X	280	X	X	280	X	X	22	X	O	22	X	X	12	X	X
320	X	X	320	X	X	320	X	X	23	O	O	23	X	O	13	X	X
360	X	O	360	X	O	360	X	O	24	O	O	24	O	O	14	X	O
400	X	O	400	O	O	400	O	O	25	O	O	25	O	O	15	O	O
440	O	O	440	O	O	440	O	O	26	O	O	26	O	O	16	O	O
480	O	X	480	O	O	480	O	O	27	O	O	27	O	O	17	O	O
520	O	O	520	O	O	520	O	O	28	O	X	28	O	O	18	O	O
560	O	O	560	O	O	560	O	O	29	O	O	29	O	X	19	O	O

圖 13、閃燃判斷實測之溫室箱氣體濃度與溫度配置畫面(由作者自行拍攝)

透過逐步測量不同情境，觀察閃燃的發生頻率，結果如下表:

表 5、閃燃判斷實測結果之混淆矩陣

	閃燃	不閃燃
閃燃	29	7
不閃燃	3	21

最後我們將實驗結果繪製成混淆矩陣總結各組情境的預測準確度，而為了以更大的可能性抓到閃燃的情況，所以我們主要關注召回率，並得閃燃發生的總體預測召回率為 90%

然而，由於此模型是建立在我們自行建構的資料集之上，雖然準確率高，但其結果未必能完全貼近實際火災現場的複雜情況。因此，我們進一步重新設計下一版的危險係數判斷方法，改以採用美國 NIOSH 所制定的 IDLH 標準作為依據，以提升危險判定的真實性。

2. 第二版危險係數判斷

為了即時判斷火場中潛在的危險區域，我們將原本使用機器學習訓練的閃燃模型，改為採用已被廣泛認可的現有標準。系統中的危機係數（0～9 級）主要根據 NIOSH IDLH 標準進行轉換：

(1) 依據美國 NIOSH IDLH 標準判定氣體危險程度

本研究參考美國國家職業安全衛生研究所（NIOSH）公布的 IDLH 標準，該標準定義為「短時間內暴露即可能導致死亡、不可逆健康損害，或無法安全逃離現場的氣體濃度上限」，作為本系統評估各類氣體風險的依據。系統內建常見氣體之 IDLH 標準如下表所示：

表 6、常見氣體對應 IDLH 標準

氣體名稱	IDLH 濃度（ppm）	說明
氫氣（H_2）	40000	易燃、具爆炸性
一氧化碳（CO）	1200	高毒性，會導致缺氧
甲烷（CH_4）	5000	易燃氣體

當感測器所偵測的濃度超過對應 IDLH 標準值時，系統會自動將該區域標示為「極度危險區域」，並將危機係數設為第 9 級。為避險與路徑規劃的參考依據。若濃度介於安全值與 IDLH 之間，則依比例轉換為第 5～8 級不等的危機指數，用於危機地圖的風險標示與避險規劃。

四、地形探勘與繪製

(一) UE4 建模

1. 初始建模

首先我們透過 UE4 (Unreal Engine 4) 建立地形模型。在 UE4 中使用基本幾何體如 Cube、Plane 等物件，進行地形與障礙物的建模設計，構建模擬環境的基本框架，並且確保場景布局的邏輯性與整體結構的完整性。

2. 數據導出

接著為了進一步進行數值分析與規劃，我們將 UE4 中每個物件的座標位置與尺寸數據匯出至 Excel 檔案。這些數據包括物件名稱（如 Cube1、Cube2 等）、位置座標（X、Y、Z 軸）、尺寸（長、寬、高）以及相關的屬性。

3. 資料整理

而在 Excel 中我們會對匯出的數據進行清理與整理，以便後續的程式處理與平面圖繪製。例如，檢查數據完整性，刪除多餘的數據列，確保所有物件的屬性都正確無誤。

4. 平面圖生成

最後我們將整理後的數據導入程式進行二次處理，將三維模型轉化為二維平面圖。紅色框代表障礙物的邊界，並且在地圖上標註了物件的名稱與位置，形成一張具有障礙物、起點與終點的完整地圖。

	A	B	C	D	E	F	G
1	Name	X	Y	Z	Length	Width	Height
2	Floor	0	0	20	100	100	100
3	Cube_2	0	-500	20	1000	1	100
4	Cube2	0	500	20	1000	1	100
5	Cube3	500	0	20	1	1000	100
6	Cube4	-351.999	-458.957	-52.2416	100	100	100
7	Cube5	-500	-250	19.99999	1	500	100
8	Cube6	-500	149.9999	19.99999	1	100	100
9	Cube7	-500	399.9999	19.99999	1	200	100
10	Cube8	-460	-100	20	75	1	100
11	Cube9	-340	-100	20	75	1	100
12	Cube10	-300	-100	19.99999	1	200	100

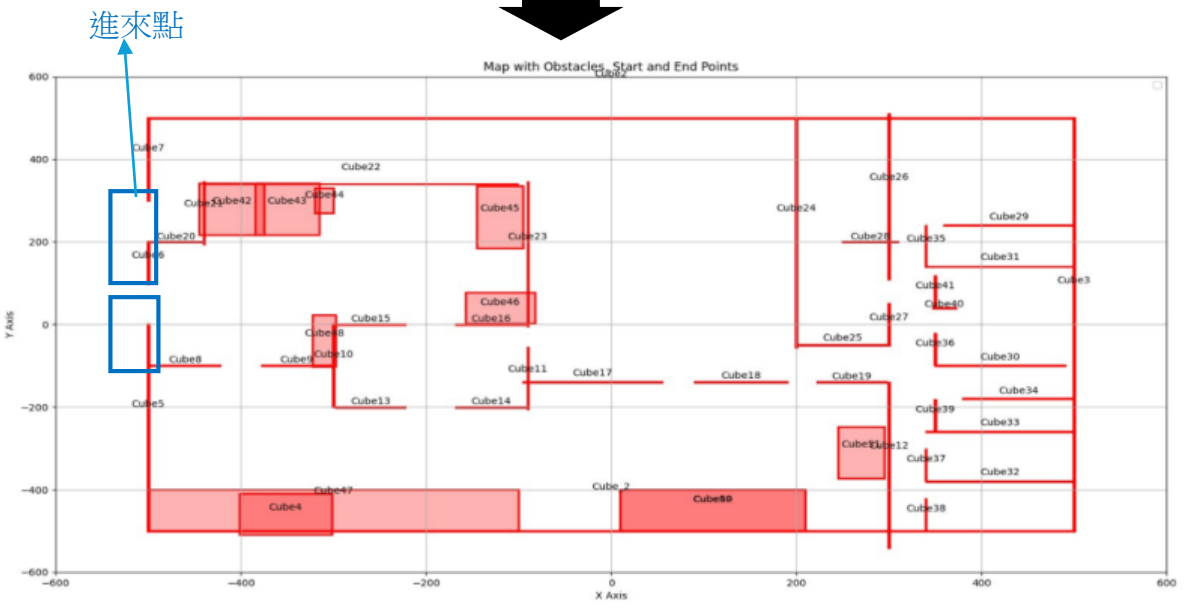


圖 14、UE4 平面圖(皆由作者自行繪製及拍攝)

五、UWB 無線定位系統與資料收集

(一) 定位系統選定及比較

針對本研究所需的定位系統我們做了前置比較，找出哪一種定位系統是最準確且適合搭配本系統。我們查詢 5 種市場上常見定位系統的資料，並繪成下表進行比較:

表 7、各室內定位特性之比較

	UWB	RFID	藍芽	WIFI	SLAM
定位精準	<1m	<2m	<1m	<2m	0.1 – 0.3 m
覆蓋距離	30m	2-5 m	10-20 m	20m	5 – 15 m
通訊頻率	3.1GHz~10.6 GHz	15kHz~5.8GHz	2.4GHz	2.4GHz、5GHz	60 – 150 Mbps
傳輸速度	480Mbps	106kbps	24Mbps	300Mbps	高(依處理器)
穿透性	強	中	中	中	低
抗干擾性	強	低	低	中	低

從比較結果可看出，在定位精準度方面，UWB 誤差小於 1 公尺，明顯優於其他技術。雖然光學 SLAM 在理想狀況下理論誤差可低至 30 公分，但實際應用中受環境干擾嚴重，穩定性遠不如 UWB。在覆蓋距離與傳輸速度方面，UWB 同樣表現出色，支援高速、高頻、穩定通訊，並具備強穿透與高抗干擾性，適合應用於多障礙、訊號阻隔嚴重的火場情境。

為進一步驗證 SLAM 在實務應用的可行性，我們實際搭建 SLAM 實驗平台，使用 Jetson Orin 作為運算核心，結合 ROS2 與 RViz 工具，並搭配光學雷達（LiDAR）與攝影機進行建圖與定位。然而於模擬火場環境中，SLAM 技術出現下列問題：

1. 感測干擾問題：SLAM 主要依賴光學影像與雷射反射建構環境地圖，然而火場中的濃煙易使鏡頭辨識失效、雷射訊號干擾，導致建圖不完整或定位誤差明顯。
2. 極端環境條件限制：高溫與強反光牆面亦干擾感測器回傳精度，進一步降低整體定位穩定性。
3. 系統成本與即時性不足：SLAM 系統需依賴高階硬體與複雜演算法，實作成本高，且處理延遲不利於高壓火場情境之即時應變。

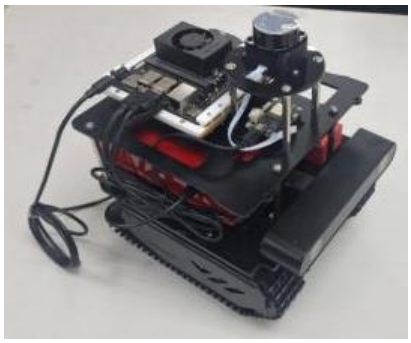


圖 15、SLAM 遙控車(由作者拍攝)

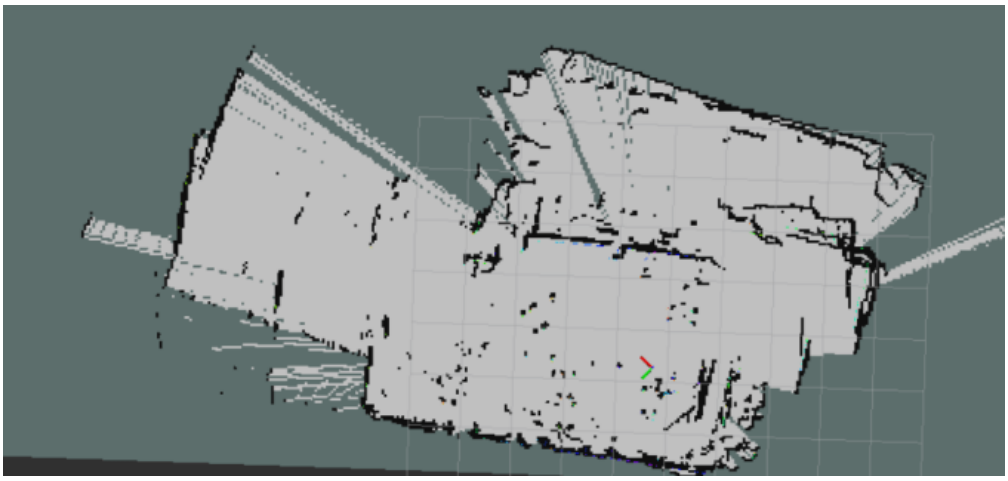


圖 16、SLAM 建圖(由作者繪製)

因此，我們最終決定棄用 SLAM，改採 UWB 無線定位，並與可拋式感測設備整合使用，提升整體應用彈性與實務價值。

(二) UWB 系統前測

我們首先在教室中設置 4 個基站（Anchor），而這些基站的座標是事先測量並固定的，同時確保它們在三維空間中的相對位置不變。接著我們透過 UWB 無線定位進行即時測距，當標籤（Tag）與這些基站進行通訊時，系統會回傳該標籤與各基站之間的距離，接著再利用最小平方法，我們透過至少 3 個基站的距離數據來計算標籤所在的位置座標（x, y）。而當其接收到 4 個基站的數據時，我們則會透過加權平均方式提高定位準確度，以減少測量誤差的影響。


```
valid_anchors = [(i, anc) for i, anc in enumerate(anchors) if uwb.list[i] > 0]
if len(valid_anchors) < 3:
    return False
x1, y1 = valid_anchors[0][1].x, valid_anchors[0][1].y
r1 = uwb.list[valid_anchors[0][0]]
for i in range(1, len(valid_anchors)):
    idx_i, anc_i = valid_anchors[i]
    xi, yi = anc_i.x, anc_i.y
    ri = uwb.list[idx_i]
    A.append([xi - x1, yi - y1])
    b.append(0.5 * ((xi**2 + yi**2 - ri**2) - (x1**2 + y1**2 - r1**2)))
A = np.array(A)
b = np.array(b).reshape(-1, 1)
pos = np.linalg.inv(A.T @ A) @ A.T @ b
uwb.set_location(int(pos[0][0]), int(pos[1][0]))
```

圖 17、實測圖、根據錨點距離定位 UWB 標籤程式(由作者繪製和拍攝)

本系統透過多個已知錨點與標籤之間的距離資訊進行定位。首先，將標籤與各錨點的圓形範圍方程式兩兩相減，化簡成線性方程組。接著利用最小平方法，將這些線性方程組轉換成矩陣運算形式求解。(A 是根據錨點位置與距離所建立的係數矩陣，x 是我們要求解的標籤位置向量，b 是距離與已知資料整理出的結果向量。)

$$E(x) = \|Ax - b\|^2 = (Ax - b)^T(Ax - b)$$

$$\frac{\partial E}{\partial x} = 2A^T(Ax - b)$$

$$A^T A^x = A^T b$$

$$x = (A^T A)^{-1} A^T b$$

當有效錨點不足或距離資料異常導致最小平方法無法求解時，系統採用兩圓估算法輔助定位。兩圓估算法透過計算兩個錨點距離圓的交點，或在不相交時取圓心連線上的比例值，估計標籤位置。系統分別計算三組錨點配對的兩圓位置，並將結果平均以提升估計穩定性。

```
def three_point(self, x1, y1, x2, y2, r1, r2):
    p2p = math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
    if r1 + r2 <= p2p:
        temp_x = x1 + (x2 - x1) * r1 / (r1 + r2)
        temp_y = y1 + (y2 - y1) * r1 / (r1 + r2)
    else:
        dr = p2p / 2 + (r1**2 - r2**2) / (2 * p2p)
        temp_x = x1 + (x2 - x1) * dr / p2p
                        temp_y = y1 + (y2 - y1) * dr / p2p
    return temp_x, temp_y
```

圖 18、計算兩個圓的交點(由作者繪製)

在獲取標籤的位置數據後，我們會將這些隨身標籤的座標存入數據庫，並記錄每筆數據的時間，方便後續分析與應用。

	name	posX	posY	createdDate
	過濾	過濾	過濾	過濾
1217	TAG 0	-153	-83	2025-02-21 16:08:02
1218	TAG 0	-164	-107	2025-02-21 16:08:03
1219	TAG 0	110	557	2025-02-21 16:08:03
1220	TAG 0	105	562	2025-02-21 16:08:04
1221	TAG 0	112	555	2025-02-21 16:08:04
1222	TAG 0	-167	-140	2025-02-21 16:08:05
1223	TAG 0	-166	-156	2025-02-21 16:08:06
1224	TAG 0	-156	-166	2025-02-21 16:08:06
1225	TAG 0	-148	-172	2025-02-21 16:08:07
1226	TAG 0	-140	-199	2025-02-21 16:08:08
1227	TAG 0	-126	-197	2025-02-21 16:08:08

圖 19、UWB 資料儲存(由作者繪製)

接著我們進一步處理這些數據，我們透過數學運算進行座標旋轉與座標偏移，使 Anchor 0 成為新的原點 (0,0)，並確保所有標籤的座標相對於這個新原點來顯示，使視覺化圖像更直觀易讀


```
def draw_uwb(uwb):
    pixel_x = int(uwb.x * cm2p + x_offset)
    pixel_y = SCREEN_Y - int(uwb.y * cm2p +
y_offset)
    if uwb.status:
        r = 10
        temp_str = uwb.name + " (" + str(uwb.x)
+ "," + str(uwb.y) + ")"
        font = pygame.font.SysFont("Consola",
24)
        surf = font.render(temp_str, True,
uwb.color)
        screen.blit(surf, [pixel_x, pixel_y])
        pygame.draw.circle(screen, uwb.color,
[pixel_x + 20, pixel_y + 50], r, 0)
```

圖 20、Anchor、Tag 的座標圖(由作者繪製和拍攝)

六、整合系統繪製危機地圖

(一) 危機地圖程式撰寫

我們基於 Dijkstra 演算法製作危機地圖模擬系統，該系統結合環境障礙物、火場風險評估及固定定點規劃，以模擬救援行動中的最佳路徑規劃。

1. 環境數據讀取與處理

我們主要透過 Python 進行開發，並運用 Pandas 讀取環境數據，利用 Matplotlib 繪製地圖，並使用 NumPy 計算風險係數。我們採用 Dijkstra 演算法，並運用 Heapq 實作其優先佇列處理。為了讓模擬情境更貼近真實情況，我們額外加入了隨機火源生成機制與風險評估模型。

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import heapq
import random
from shapely.geometry import box, Point, LineString
file_path = 'lol.csv'
data = pd.read_csv(file_path)
```

圖 21、讀取環境數據(由作者繪製)

2. 環境障礙物與探測球篩選

程式透過 pandas 讀取 CSV 檔案，其中包含環境中的障礙物（Cube）與探測球（Tag）座標資訊。程式根據 CSV 檔案中的資訊過濾並儲存障礙物及探測球的相關數據，建立出完整的環境地圖。

```
room_data = data[data['Name'].str.contains('Cube', na=False)].copy()
anchor_data = data[data['Name'].str.contains('Anchor', na=False)].copy()
```

圖 22、過濾並儲存相關數據(由作者繪製)

3. Dijkstra 演算法計算最佳路徑

程式採用 Dijkstra 演算法計算從起點至受災人員（目標點）之間的最短路徑。Dijkstra 演算法以累積移動成本作為判斷標準，確保最短的搜尋路徑。此外，程式針對環境中的障礙物進行檢測，避免搜尋路徑穿越障礙區域。

```
for dx, dy in directions:
    nxi, nyi = current.xi + dx, current.yi + dy
    if not (0 <= nxi < risk_map.shape[0] and 0 <= nyi < risk_map.shape[1]):
        continue
    current_x, current_y = index_to_coord(current.xi, current.yi)
    new_x, new_y = index_to_coord(nxi, nyi)
    if is_in_obstacle(new_x, new_y):
        continue
    if is_crossing_red_line(current_x, current_y, new_x, new_y):
        continue
    risk = risk_map[nxi, nyi]
    cost = current.cost + 10 + risk * 2
    heapq.heappush(open_list, Node(nxi, nyi, cost, current))
```

圖 23、最短的搜尋路徑(由作者繪製)

4. 視覺化地圖與路徑

本系統利用 matplotlib 將環境中的障礙物、火源、受災人員位置、固定定點（A 點）以及經由 Dijkstra 演算法計算出的最佳路徑進行可視化，確保使用者能夠直觀理解救援路線。

```
fig, ax = plt.subplots(figsize=(16, 12))
for xi in range(len(x_bins)-1):
    for yi in range(len(y_bins)-1):
        r = risk_map[xi, yi]
        color = (1, 1 - min(0.9, r / 10), 1 - min(0.9, r / 10))
        ax.fill_between([x_bins[xi], x_bins[xi+1]], y_bins[yi], y_bins[yi+1], color=color, alpha=0.3)
for poly in obstacle_polygons:
    ax.plot(*poly.exterior.xy, 'r-', linewidth=2)
ax.scatter(*zip(*fire_points), c='orange', s=100)
ax.scatter(*zip(*anchor_data[anchor_data['Risk'] > 7][['X', 'Y']].values), c='gold', s=200, marker='*')
if path:
    ax.plot(*zip(*path), 'g-', linewidth=3)
ax.scatter([start_x, goal_x], [start_y, goal_y], c='blue', s=150, marker=['o', 'P'])
plt.show()
```

圖 24、視覺化地圖與路徑(由作者繪製)

(二) 危機地圖歷代演進

1. 初始版本

在初始版本中，僅根據建築結構圖匯出資訊，進行靜態的最短路徑規劃，並根據該圖生成最短路徑。然而，該版本在規劃路徑時並未考慮實際場景中的牆壁或障礙物，導致繪製出的路徑雖然最短，但在實際操作中可能會穿越牆體或無法通行。

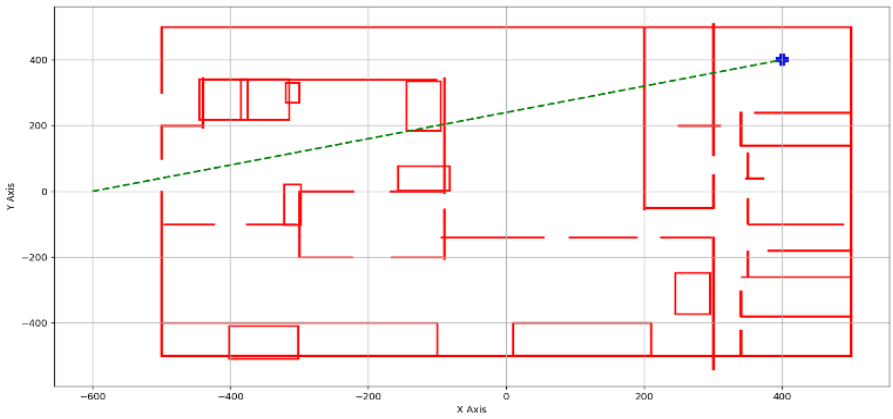


圖 25、初始危機地圖(由作者繪製)

2. 改良版本

基於初始版本中路徑規劃會穿牆的問題，我們對系統進行了改良，讓它能夠識別牆壁與障礙物，並規劃出合理且可行的通行路徑。然而，這個版本仍未考慮火災等危險因素，因此所規劃的路線仍有可能穿越高溫區域或氣體濃度異常區，對人員安全構成潛在風險。

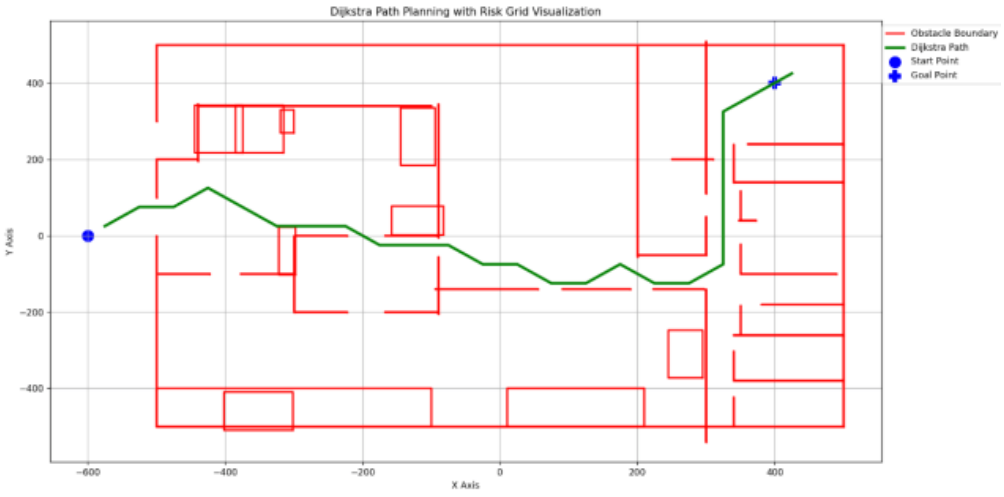


圖 26、改良危機地圖(由作者繪製)

3. 最終版本

最終版本系統整合危機係數模型與即時感測資料，實現危機地圖的動態更新與智慧路徑規劃。系統可根據環境中五項感測數據（溫度、瓦斯、氫氣、一氧化碳與粉塵），即時轉換為危險係數，並繪製風險熱區圖，有效呈現火場中高風險區域的空間分布情形。路徑模組同時搭配即時危險地圖，運行 Dijkstra 演算法，綜合考量距離與風險等級，動態規劃出最短且相對安全的避險路線。系統可即時接收來自探測球與固定式感測器的資料，自動排除高溫或有毒氣體濃度異常的區域，有效避免人員穿越潛在危險熱點。經過多場模擬與實地測試驗證，最終版本系統能在複雜且不斷變化的火場環境中穩定運作，準確避開高風險區域並選擇最佳安全路徑，顯著提升整體導航的準確性與安全性。

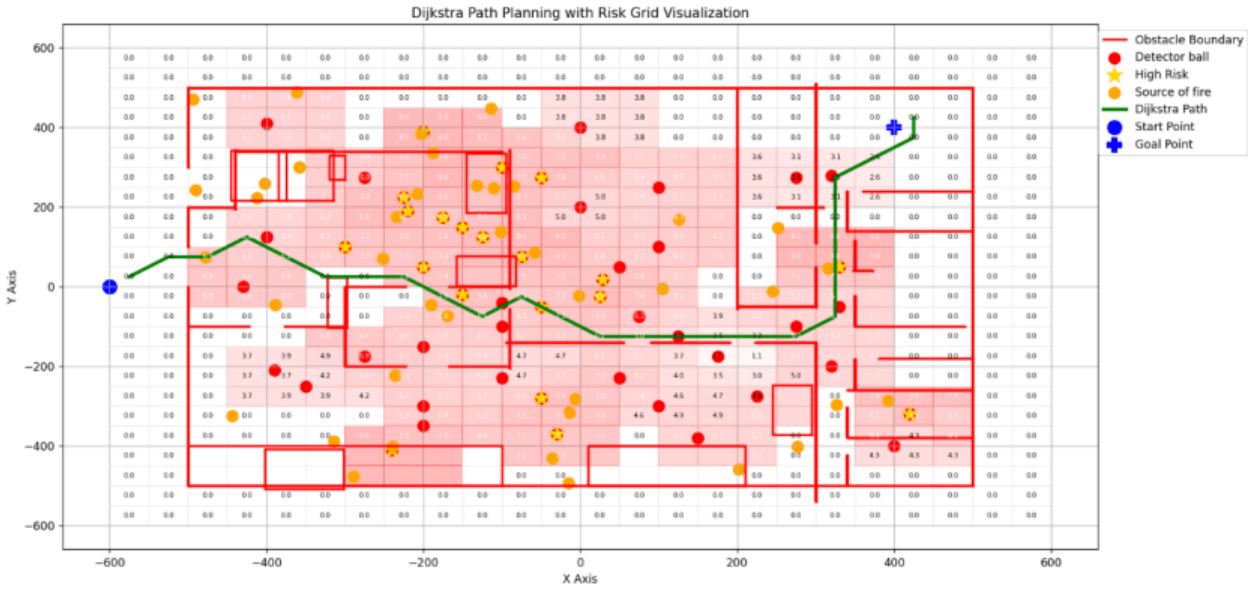


圖 27、最終危機地圖(由作者繪製)

透過以上三階段的測試與改良，我們逐步完成了危機地圖的優化設計，最終讓系統能夠以精準且智能的方式，並根據實際需求進行即時的路徑規劃，實現避險與高效的目標。

(三)危機地圖判斷流程

根據前測與實測結果，我們選用使用 Dijkstra 演算法來幫助規劃火場中的安全路線。這個方法會從起點出發，依照「目前累積距離和危險程度最低」的節點往下一步走，並依序檢查周圍八個方向的點。在這過程中，系統會自動排除有障礙物、已經走過，或危險係數太高的區塊，將符合條件的點加入待搜尋清單，直到找到可以到達終點的安全路徑。當成功找到路徑時，系統會回溯出一條最短又安全的撤離路線。如果整張地圖都無法通過，則會顯示「無法抵達」。這樣的設計可以幫助我們在危險環境中快速找出最安全的逃生方向，提升火場救援效率。

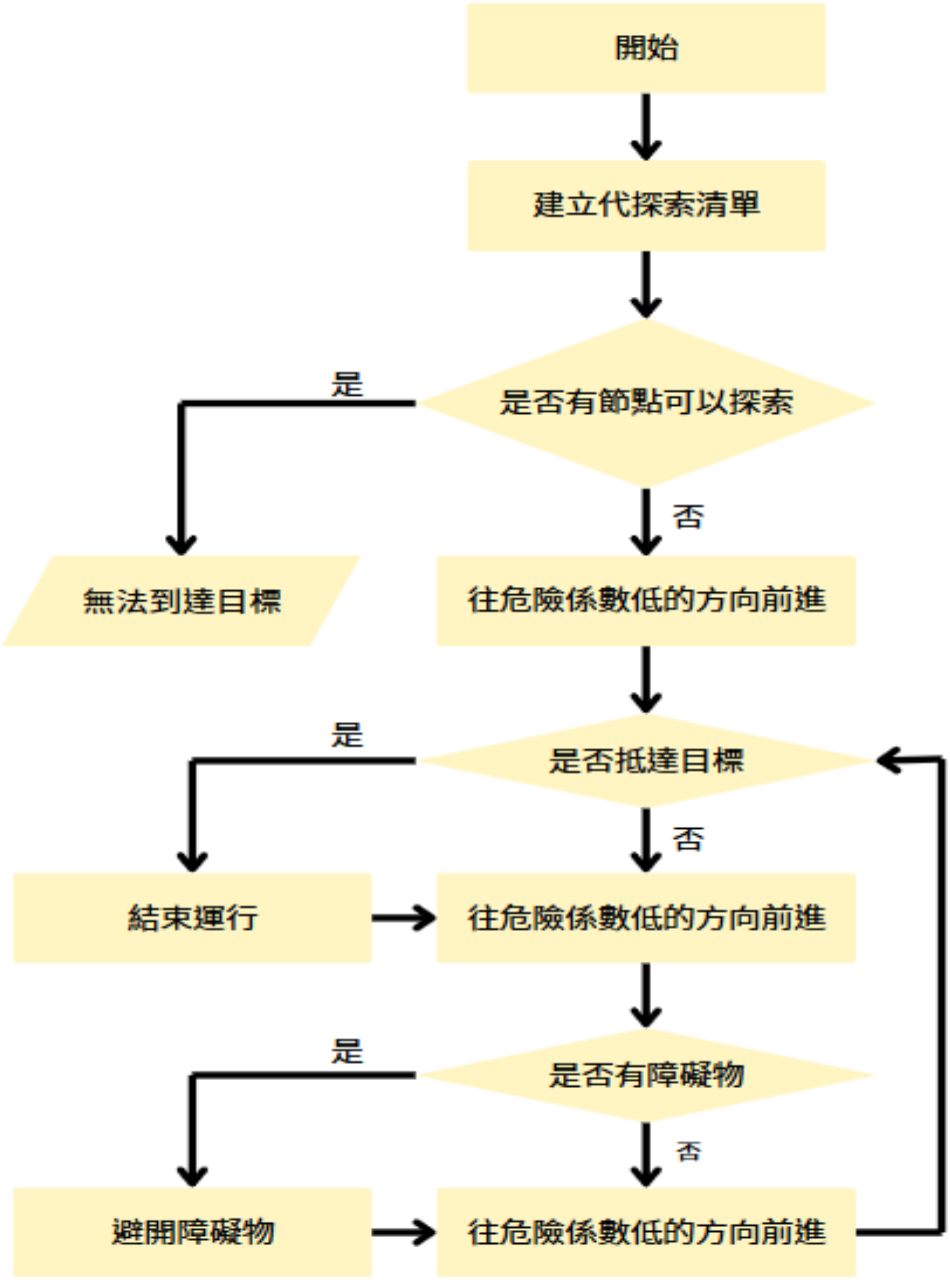


圖 28、危機地圖判斷流程圖（由作者使用 Canva 繪製）

肆、研究結果

一、系統發展之功能架構

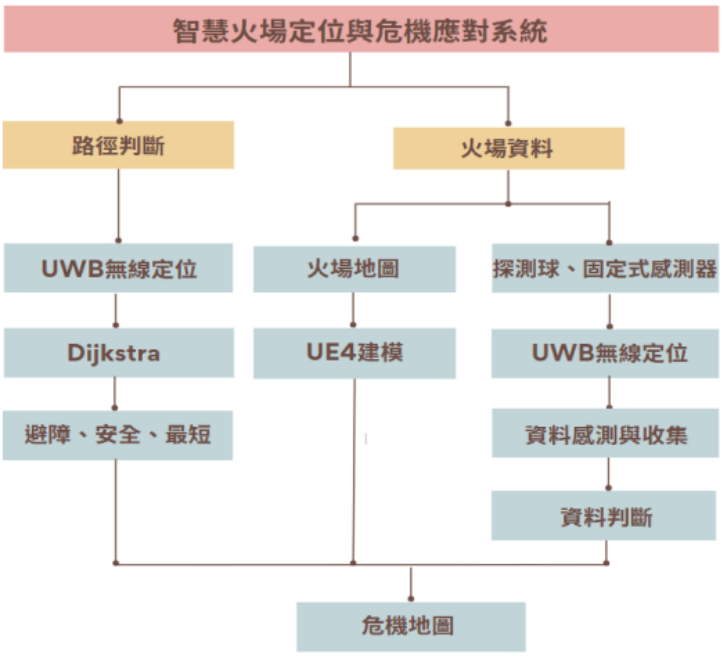


圖 29、系統架構圖（由作者使用 Canva 繪製）

我們設計了一套智慧型火場定位與危機應變系統，以 UWB 無線定位為核心，整合即時感測資料與模擬建築地圖，目的在於提升火場中人員救援的效率與安全性。整體系統可分為兩大模組：「路徑判斷模組」與「火場資料模組」。

(一)路徑判斷模組

1. UWB 無線定位：運用 UWB 的無線定位進行即時定位，確保人員或裝置位置的精準性。
2. Dijkstra 演算法：我們運用 Dijkstra 演算法，根據每個區塊的危險係數與距離，計算出一條既安全又快速的最佳逃生或救援路線，同時避開高風險區域與障礙物。

(二)火場資料模組

1. 火場地圖與建模：使用 UE4 進行火場空間建模，讓火場地圖更加具體化與視覺化，方便觀察各區域風險與結構。
2. 探測球與固定式感測器：我們設計了一款可投擲式探測球，並結合多點固定式感測器，能同時監測溫度、濃煙、一氧化碳、氫氣等多種危險因子。
3. 資料感測與判斷：系統會整合感測器的即時資料，再透過危機係數模型進行風險評估，判斷哪些區域較危險，作為危機地圖的依據。

最終，系統將兩大模組整合，生成危機地圖，即時呈現火場動態與安全路徑，提升救援判斷力與應變效率。

二、系統運作流程圖

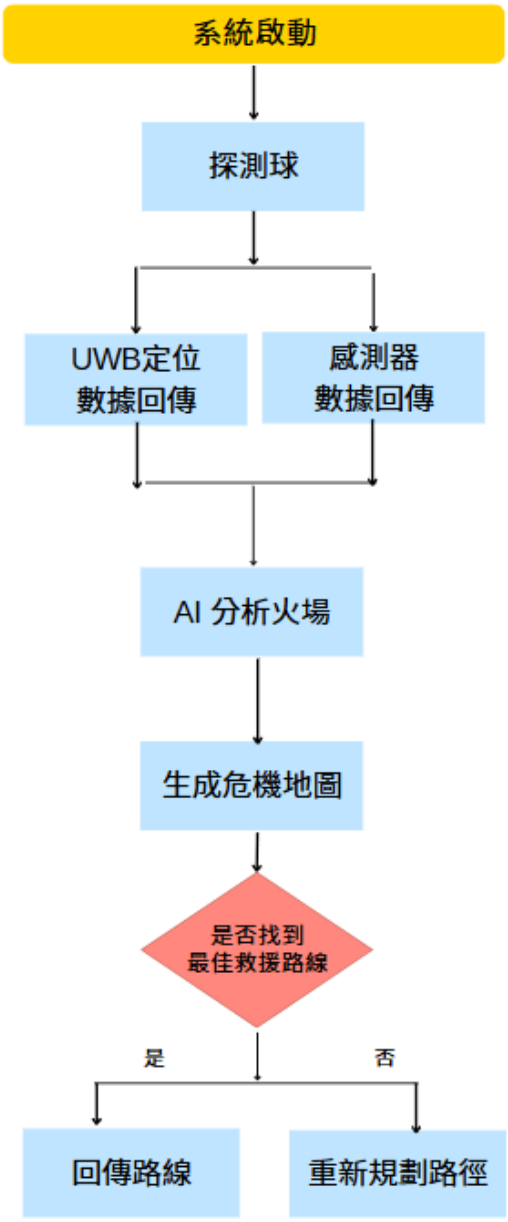


圖 30、系統運作流程圖（由作者使用 Canva 繪製）

在系統啟動後，探測球會被投入火場中進行資料蒐集。透過 UWB 模組，系統可即時取得探測球的定位資訊，同時整合感測器回傳的火場數據（如溫度、氣體濃度等）。

當系統接收到所有資料後，AI 模型將分析火場現況，並生成即時危機地圖。接著系統會嘗試規劃最佳的救援路線。若成功找到合適路徑，系統將即時回傳導航路線；若無法規劃出安全路徑，系統則會自動重新調整並進行路徑重算，以確保救援行動的安全性與效率。

三、危機地圖實測結果

(一) 16 宮格感測平台實測

為驗證本系統在實際感測部署下的整體效能，我們建置一組 4x4 感測模擬平台（共 16 格），每格模擬火場中不同區域的環境條件。每格搭載一組感測模組，包含溫度、瓦斯、氫氣、粉塵與一氧化碳感測器，以模擬火場中異常環境的空間分布。

圖 31、16 宮格感測裝置（由作者拍攝）

各個感測器經由 ESP32 即時傳輸數據至 Raspberry Pi，並儲存於 SQLite 資料庫中。為了辨識與管理這些資料，每一格皆配置專屬編號（1~16），並對應資料庫欄位。。

接下來，我們根據這些資料繪製出一張 16 宮格的危機地圖，將每格的即時感測數值轉換為對應的危險係數，用來模擬火場中各區域的風險分布狀況。為了測試危機地圖的動態路徑規劃能力，我們會持續讀取最新的 SQLite 資料，並即時更新安全路線。

如圖 32 所示，我們將地圖的左上角（0,0）設為起點，右下角（3,3）設為終點，顏色由淺至深分別對應低至高的危險係數。在路徑規劃時，我們不只考慮路徑的距離，同時也將每格的危險係數納入權重評估，可以看到圖中藍色路徑即為演算法所計算出的最短且相對安全的撤離路線，顯示系統可有效避開風險較高的區塊，並優先通過風險較低的區域，找出一條最短又相對安全的撤離路線，提供現場人員更即時、可靠的避難建議。

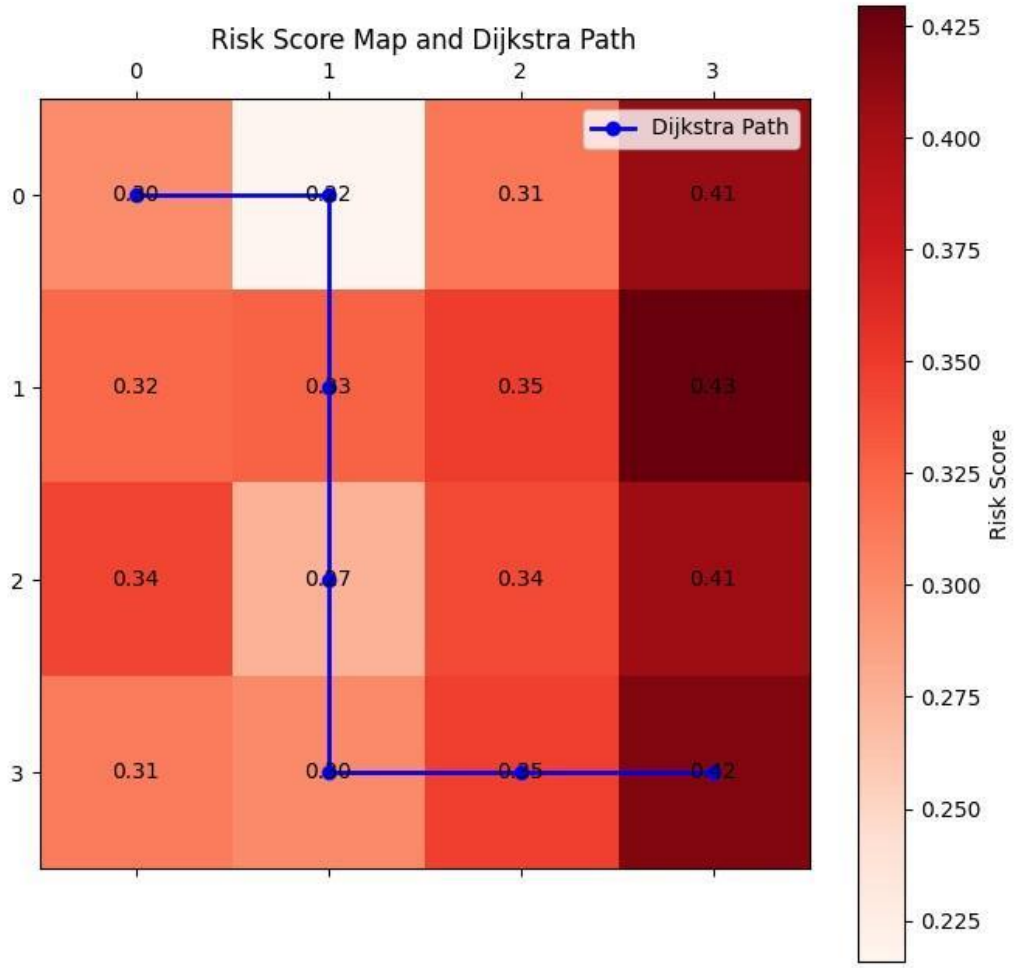


圖 32、16 宮格危機地圖(由作者繪製)

(二) 危機地圖模擬與路徑規劃測試

我們在模擬地圖中設計火場場景，自訂感測器(橘黃點)及探測球(紅點)位置，並標註各探測球所運算出的危險係數來模擬火場情境，如下圖 33。我們利用障礙物（紅線）將場地劃分為多個區域，探測點分佈於不同位置，並且各個區域存在不同程度的危險數據分布（火焰標記）。危險數據的熱區分布清晰顯示了高危險區域，為後續的安全路徑規劃提供了重要參考依據。

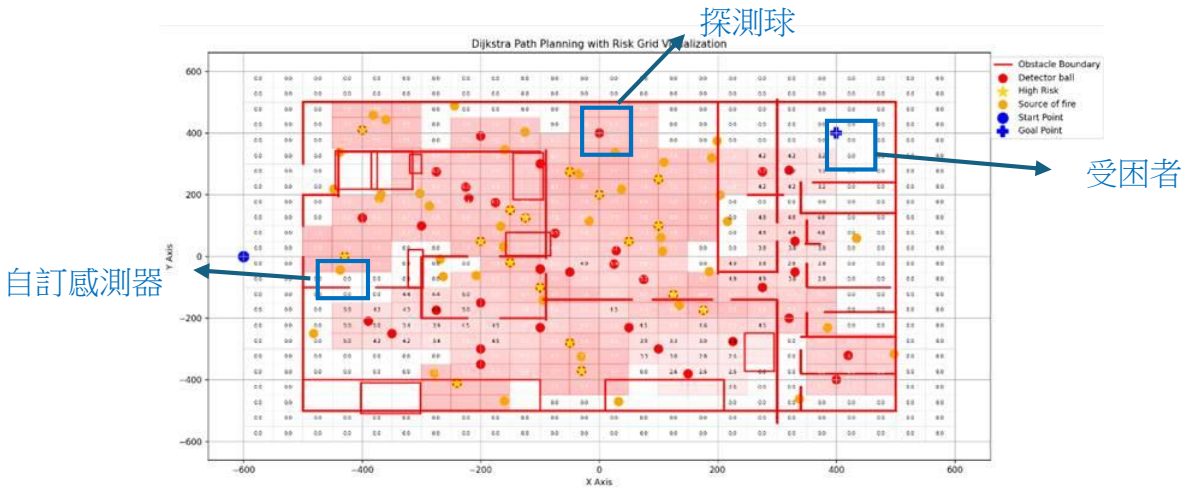


圖 33、危機地圖(由作者繪製)

接著利用 Dijkstra 演算法對模擬地圖進行路徑規劃，測試系統在該地圖中生成最佳的安全路徑。規劃的路徑有效避開高危險區域與障礙物，並優化了移動距離與時間。而經系統運算後最佳路徑如下圖 34，綠線為最佳路徑。

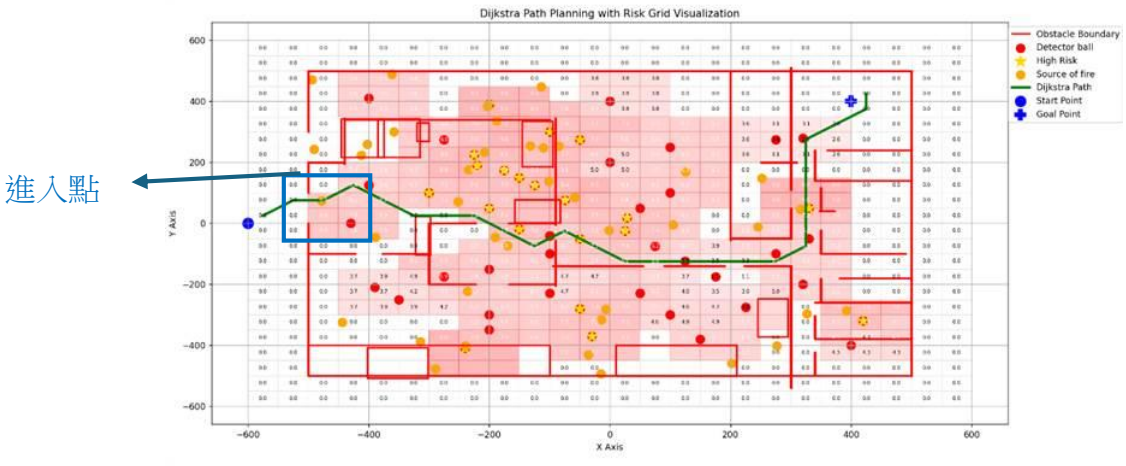


圖 34、最佳路徑的危機地圖(由作者繪製)

伍、結論

本研究整合超寬頻（UWB）無線定位技術、危機係數評估與多感測器資料，成功建立一套智慧型火場救援系統。經多次實測與系統優化，驗證本系統能有效掌握火場資訊，提升定位精度與救援效率，並提供安全、即時的撤離與救援路徑建議。其主要貢獻優勢如下：

(一) 即時定位與優化救援路徑：

應用 UWB 無線定位技術進行人員即時定位，結合危險係數評估模型與 Dijkstra 演算法，即時規劃最短且最安全之撤離與救援路徑，提升救援決策準確度與執行效率。

(二) 數據即時傳輸與決策輔助:

本研究透過探測球與固定式感測器收集火場內的關鍵數據（如溫度、一氧化碳、粉塵等），並將數據即時傳輸至雲端，結合 IDLH 模型生成危機係數地圖，再透過 AI 進行分析與判斷，生成即時危機地圖，提供消防人員準確的避險資訊，進一步提升救援決策的可靠性。

(三) 可與現有救援系統整合，降低部署成本:

本系統可與現有消防設備（如監視器、消防感測器等）整合，減少額外設備的安裝需求，並可利用既有的無線網路進行資料傳輸，降低部署與維護成本，使其具備良好的擴展性與實用價值。

(四) 系統可擴充與持續優化:

本系統可與既有的消防監測設施整合，部署成本低，具高度應用彈性，未來有望成為建築火災應變與災防規劃的重要輔助工具。

透過本研究的成果，我們希望能在未來進一步推動 UWB 無線定位技術於消防救援的應用，為災害應變提供更高效、安全的解決方案，降低火場救援的風險，保障人員生命安全，並減少財物損失。

陸、參考文獻

[1] Yahoo奇摩新聞。 (2024年5月28日).。6千萬晴空匯豪宅火災，3大致命錯誤同時發生、火不大濃煙卻超大嗆死2勇消…鑑定釀禍源頭是它. https://reurl.cc/0vRgbo

[2] 中華民國內政部消防署。 (2022年)取自 https://www.nfa.gov.tw/cht/index.php?code=list&ids=220

[3] 陳昱志（2019）。以 UWB 為輔助之高精度定位儀及其應用〔碩士論文，國立臺灣大學〕。華藝線上圖書館。https://doi.org/10.6342/NTU201902526。

[4] ETtoday 新聞網(2024 年 12 月 19 日) 全聯倉儲惡火 9 死名單　其中 7 名是台灣人，https://www.ettoday.net/news/20241219/2877093.htm。

[5] Feasycom(2024 年 5 月 11 日)。關於 10 公分精度 UWB 定位技術，您需要了解甚麼 https://www.feasycom.com/zh-TW/what-you-need-to-know-about-uwb-positioning-technology-with-10cm-accuracy.html

[6] 工具機與零組件雜誌，財團法人精密機械研究發展中心陳偉民, Tesfaye Wakessa Gussu, 賴嬿如, 與陳哲堅(2023 年 10 月 18 日)。超寬頻(UWB)室內定位系統應用於智能工廠的部署與最佳化 https://www.maonline.com.tw/article_inside.php?i=866。

[7] 內政部建築研究所(2022)，109 年度「前瞻建築防火避難及結構科技研發整合應用計畫(二)協同研究計畫」住宅防火對策之研究 https://ws.moi.gov.tw/001/Upload/404/relfile/9489/213919/3ea2ed7c-101a-4f9b-8fbb-fc508c301fc5.pdf。

[8] 內政部消防署全球資訊網(2017)，第十三章 搶救高層建築物火災安全指導原則

[9] 洪睿妍、張益城、連季宏(2024)。AI 消防閃燃偵測輔助系統。中華民國第 64 屆中小學科學展覽會高中組 工程組(一) 鄉土教材獎。新北市立新北高級中學。

[10] You-He Siao(蕭佑和)(2018 年 05 月 02 日) 群雄並起，室內定位技術大比拼｜大和有話說 https://dahetalk.com/2018/05/02/%E7%BE%A4%E9%9B%84%E4%B8%A6%E8%B5%B7%EF%BC%8C%E5%AE%A4%E5%85%A7%E5%AE%9A%E4%BD%8D%E6%8A%80%E8%A1%93%E。

[11] FireLeaks. (2021 年 2 月 22 日)。消防解密，火的一生-從起源到熄滅(解密你從沒看過的火災成長曲線)。 https://fireleaks.com/%E7%81%AB/

[12] 王榮豪、吳佳隆(2022)。智慧防火防災科技關鍵技術及應用規劃之研究

附錄一 Anchor 設置程式

```
/*
For ESP32S3 UWB AT Demo
Use 2.0.0 Wire
Use 1.11.7 Adafruit_GFX_Library
Use 1.14.4 Adafruit_BusIO
Use 2.0.0 SPI
Use 2.5.7 Adafruit_SSD1306
*/
// User config -----------------------------------------
#define UWB_INDEX 3
#define ANCHOR
#define UWB_TAG_COUNT 64
// User config end -----------------------------------------
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Arduino.h>
#define SERIAL_LOG Serial
#define SERIAL_AT mySerial2
HardwareSerial SERIAL_AT(2);
// ESP32S3
#define RESET 16
#define IO_RXD2 18
#define IO_TXD2 17
#define I2C_SDA 39
#define I2C_SCL 38
Adafruit_SSD1306 display(128, 64, &Wire, -1);
void setup()
{
pinMode(RESET, OUTPUT);
digitalWrite(RESET, HIGH);
SERIAL_LOG.begin(115200);
SERIAL_LOG.print(F("Hello! ESP32-S3 AT command V1.0 Test"));
SERIAL_AT.begin(115200, SERIAL_8N1, IO_RXD2, IO_TXD2);
SERIAL_AT.println("AT");
Wire.begin(I2C_SDA, I2C_SCL);
delay(1000);
// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V
internally
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))
{ // Address 0x3C for 128x32
SERIAL_LOG.println(F("SSD1306 allocation failed"));
for (;;)
; // Don't proceed, loop forever
}
display.clearDisplay();
logoshow();
sendData("AT?", 2000, 1);
sendData("AT+RESTORE", 5000, 1);
sendData(config_cmd(), 2000, 1);
sendData(cap_cmd(), 2000, 1);
// sendData("AT+GETCAP?/r/n", 2000, 1);
sendData("AT+SETRPT=1", 2000, 1);
sendData("AT+SAVE", 2000, 1);
sendData("AT+RESTART", 2000, 1);
}
long int runtime = 0;
String response = "";
//String rec_head = "AT+RANGE";
void loop()
{
// put your main code here, to run repeatedly:
while (SERIAL_LOG.available() > 0)
{
SERIAL_AT.write(SERIAL_LOG.read());
yield();
}
while (SERIAL_AT.available() > 0)
{
char c = SERIAL_AT.read();
if (c == '\r')
continue;
else if (c == '\n' || c == '\r')
{
SERIAL_LOG.println(response);
response = "";
}
else
response += c;
}
}
/*
void formatToJSON(String str) {
char *js = strtok(str.c_str(), ":");
}
*/
// SSD1306
void logoshow(void)
{
display.clearDisplay();
display.setTextSize(1); // Normal 1:1 pixel scale
display.setTextColor(SSD1306_WHITE); // Draw white text
display.setCursor(0, 0); // Start at top-left corner
display.println(F("MaUWB DW3000"));
display.setCursor(0, 20); // Start at top-left corner
// display.println(F("with STM32 AT Command"));
display.setTextSize(2);
String temp = "";
temp = temp + "A" + UWB_INDEX;
temp = temp + " 6.8M";
display.println(temp);
display.setCursor(0, 40);
temp = "Total: ";
temp = temp + UWB_TAG_COUNT;
display.println(temp);
display.display();
delay(2000);
}
String sendData(String command, const int timeout, boolean debug)
{
String response = "";
// command = command + "\r\n";
SERIAL_LOG.println(command);
SERIAL_AT.println(command); // send the read character to the
SERIAL_LOG
long int time = millis();
while ((time + timeout) > millis())
{
while (SERIAL_AT.available())
{
// The esp has data so display its output to the serial window
char c = SERIAL_AT.read(); // read the next character.
response += c;
}
}
if (debug)
{
SERIAL_LOG.println(response);
}
return response;
```

```
}
String config_cmd()
{
String temp = "AT+SETCFG=";
// Set device id
temp = temp + UWB_INDEX;
// Set device role
//x2:Device Role(0:Tag / 1:Anchor)
temp = temp + ",1";
// Set frequence 850k or 6.8M
temp = temp + ",1";
// Set range filter
temp = temp + ",1";
return temp;
}
String cap_cmd()
{
String temp = "AT+SETCAP=";
// Set Tag capacity
temp = temp + UWB_TAG_COUNT;
// Time of a single time slot 6.5M : 10MS 850K ： 15MS
temp = temp + ",10";
//X3:extMode, whether to increase the passthrough command when
transmitting
///(0: normal packet when communicating, 1: extended packet when
communicating)
temp = temp + ",0";
return temp;
}
```

【評語】052306

1. 本作品因應火場複雜環境、視線遮蔽等問題造成救災不便為動機，開發可拋式探測球、光達探測車、搭配無線定位技術，建構危險地圖，並用風險模型建議消防人員避開高風險區域之研究。

2. 研究中整合多種技術，顯示參賽選手對於該研究實驗方法熟悉、可清楚闡釋其開發系統之原理等，值得鼓勵。

3. 建議未來可多考量實際應用情境之可行性、各變量因果關係與感測器之使用極限，並藉由與實際火場之需求，設定應用情境、調整參數模型，應可提高風險評估的精確度與本作品之影響力。

作品海報

智慧火場定位與危機應對系統

摘要

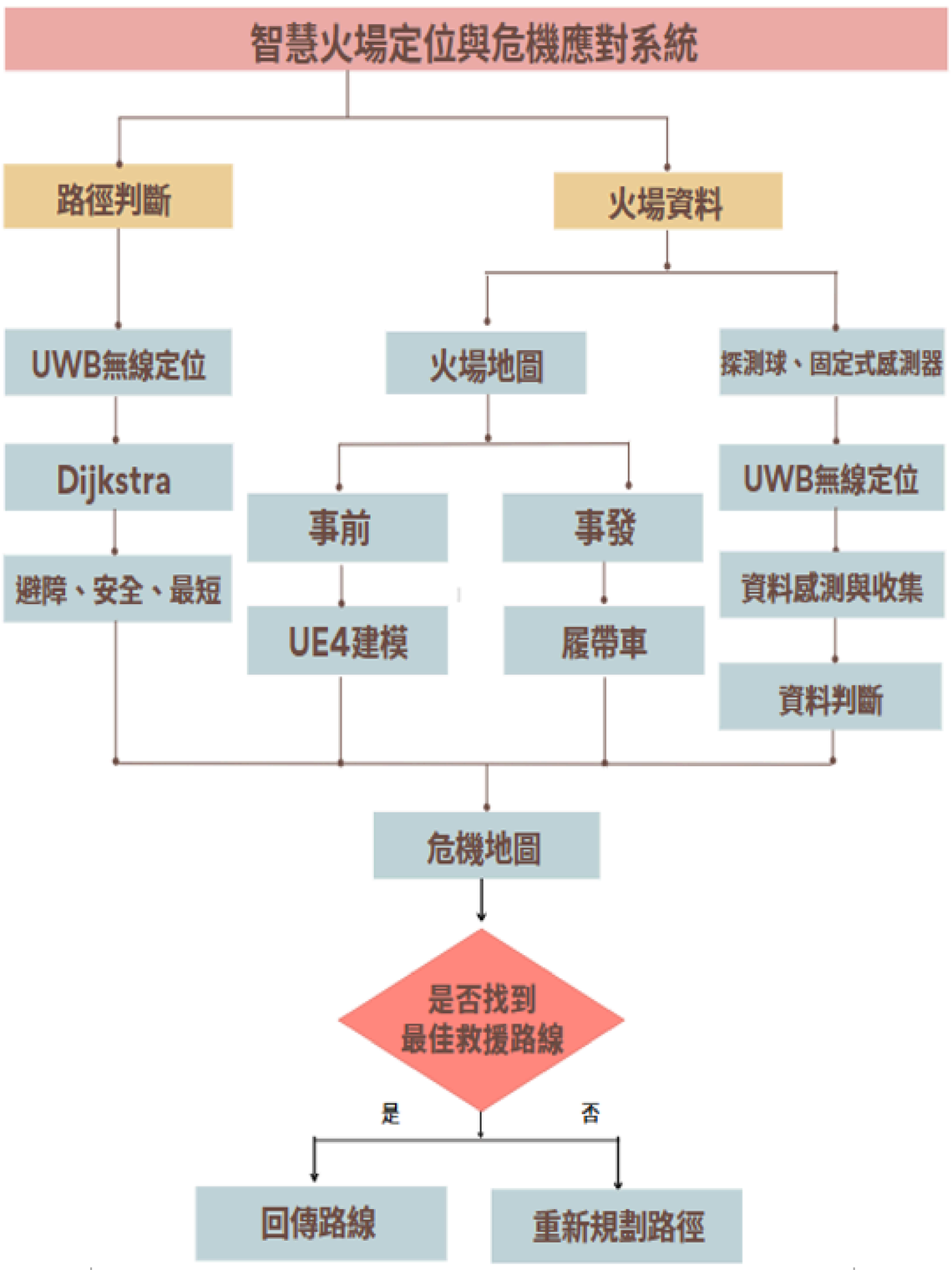
本研究建構一套結合UWB 無線定位技術、可拋式多感測探測球與風險係數模型的智慧火場應變系統，解決濃煙遮蔽與視線受限下難以定位與判斷的問題。探測球可即時回傳火場中氫氣、一氧化碳、瓦斯、粉塵與溫度數據，並依據 IDLH 標準轉換為危機係數，產出即時危機地圖。系統運用 Dijkstra 演算法進行風險加權路徑搜尋，規劃最短且最安全的撤離與救援路線。資料透過 ESP32 傳至 Raspberry Pi，使用 SQLite 儲存並搭配視覺化模組呈現。模擬實測結果顯示，本系統具備高準確性與實用性，未來可應用於火災、地震等高風險災害現場，作為智慧救援決策的輔助工具。

研究目的

(一) 分析不同環境與現有救援技術的限制。
(二) 整合UWB無線定位與多感測器之可拋式探測球，進行即時定位與火場數據蒐集。
(三) 建危險係數模型，做出風險分級與決策資訊。
(四) 開發危機地圖並測試其在逃生與救援路徑規劃的應用。

研究過程與結果

一、系統架構圖

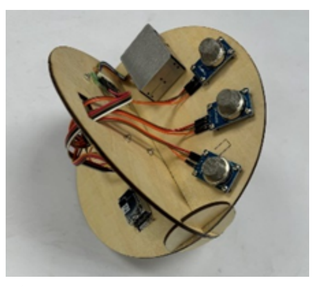


▲系統架構圖（由作者使用Canva繪製）

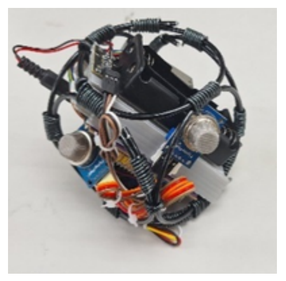
二、探測球製作

(一)探測球演化

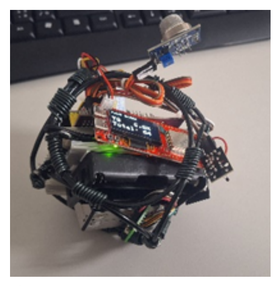
本研究經歷三代探測球設計。初代木質外殼因感測器配置不均造成誤差，第二代穩定性提升，第三代新增UWB無線定位模組，增強火場定位與資料回傳能力。



▲第一代探測球（作者自行拍攝）



▲第二代探測球（作者自行拍攝）



▲第三代探測球（作者自行拍攝）

▲實驗畫面（作者自行拍攝）

(二)危機係數

為能即時判斷火場中潛在高風險區域，本系統原本使用機器學習進行閃燃模型預測，然而因資料需求大且推論延遲高，難以滿足即時應變需求。

因此，我們改採 美國 NIOSH 所制定之 IDLH標準，做為危險判定依據。

表1、常見氣體對應 IDLH 標準（由作者繪製）

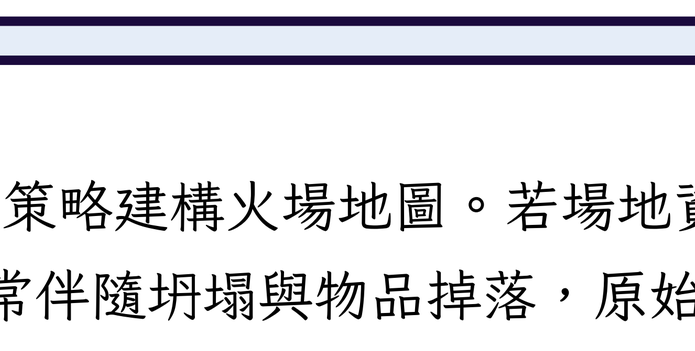
氣體名稱	IDLH 濃度（ppm）	說明
氫氣（H_2）	40000	易燃、具爆炸性
一氧化碳（CO）	1200	高毒性，會導致缺氧
甲烷（CH_4）	5000	易燃氣體

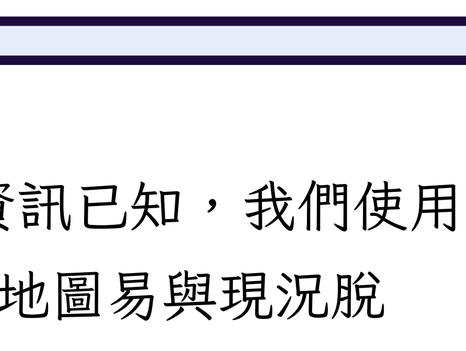
三、UE4建模 × SLAM建模

本系統採事前 UE4 建模與事發SLAM 動態建圖雙策略建構火場地圖。若場地資訊已知，我們使用 UE4 建立模型，作為初始地圖參考；但實際火場常伴隨坍塌與物品掉落，原始地圖易與現況脫節。因此，於事發當下，我們整合SLAM 技術，即時建圖補正空間變動，並結合UWB無線定位與感測器資料，產生即時危機地圖。

	A	B	C	D	E	F	G
1	Name	X	Y	Z	Length	Width	Height
2	Floor	0	0	20	100	100	1
3	Cube_2	0	-500	20	1000	1	1
4	Cube2	0	500	20	1000	1	1
5	Cube3	500	0	20	1	1000	1
6	Cube4	-351.999	-458.957	-52.2416	100	100	1
7	Cube5	-500	-250	19.99999	1	500	1
8	Cube6	-500	149.9999	19.99999	1	100	1
9	Cube7	-500	399.9999	19.99999	1	200	1
10	Cube8	-460	-100	20	75	1	1
11	Cube9	-340	-100	20	75	1	1
12	Cube10	-300	-100	19.99999	1	200	1

▲UE4平面圖(由作者拍攝、繪製)



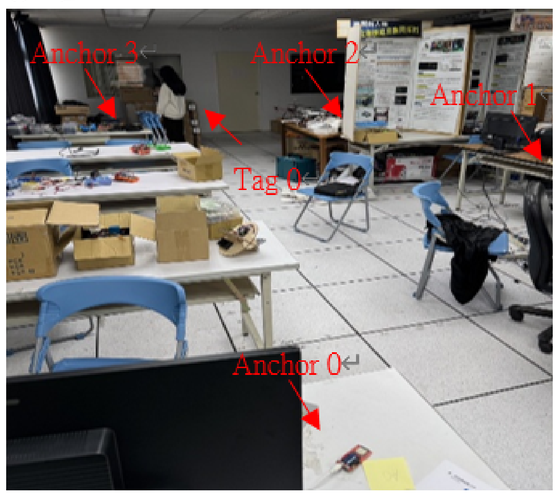


▲SLAM遙控車、SLAM建圖(由作者拍攝、繪製)

四、UWB系統前測

(一)設備設置

我們在教室裡設置了4個固定的基站，並確保它們的位置穩定。利用UWB無線定位系統進行即時測距，當標籤與基站通訊時，系統會回傳與各基站的距離。透過三角定位計算標籤的位置（x, y）。如果收到4個基站的數據，則使用加權平均方法提高定位準確度，減少誤差影響。



▲實測圖(由作者拍攝)

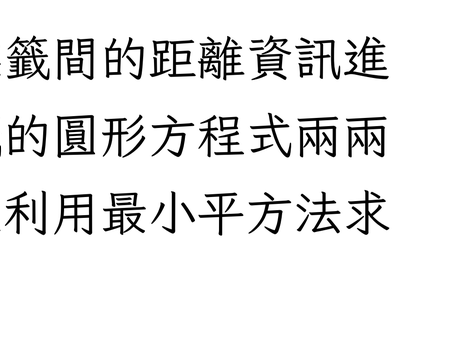
(二)座標化與視覺化

本系統透過多個已知錨點與標籤間的距離資訊進行定位。首先，將錨點與標籤的圓形方程式兩兩相減，轉換成線性方程組，並利用最小平方法求解位置向量x：

$$E(x) = \|Ax - b\|^2 = (Ax - b)^T(Ax - b)$$

當錨點不足或距離異常時，系統改用兩圓估算法，透過計算兩錨點圓的交點或圓心連線比例估算位置，並對多組配對結果取平均以提升穩定性。標籤座標會記錄至資料庫並附上時間戳記，後續再進行座標旋轉與偏移，以 Anchor 0 為原點，讓圖像呈現更直觀易讀。

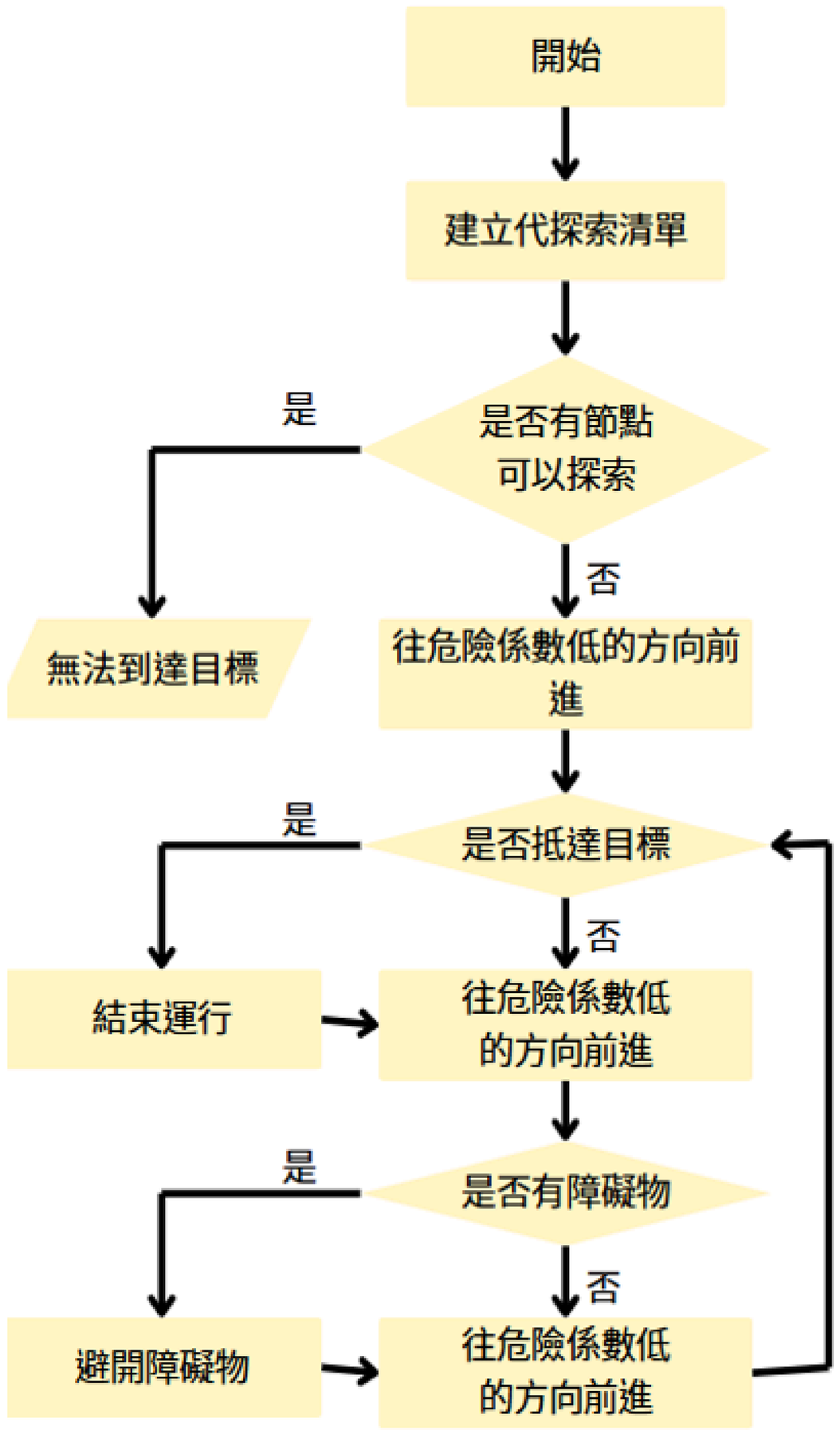
	name	posX	posY	createdDate
	過濾	過濾	過濾	過濾
1217	TAG 0	-153	-83	2025-02-21 16:08:02
1218	TAG 0	-164	-107	2025-02-21 16:08:03
1219	TAG 0	110	557	2025-02-21 16:08:03
1220	TAG 0	105	562	2025-02-21 16:08:04
1221	TAG 0	112	555	2025-02-21 16:08:04
1222	TAG 0	-167	-140	2025-02-21 16:08:05
1223	TAG 0	-166	-156	2025-02-21 16:08:06
1224	TAG 0	-156	-166	2025-02-21 16:08:06
1225	TAG 0	-148	-172	2025-02-21 16:08:07
1226	TAG 0	-140	-199	2025-02-21 16:08:08
1227	TAG 0	-126	-197	2025-02-21 16:08:08



▲ UWB資料儲存和Anchor、Tag的座標圖(由作者拍攝)

五、危機地圖判斷

本系統採用 Dijkstra 演算法規劃火場安全路線，從起點出發，優先搜尋累積距離與危險係數最低的節點，並檢查周圍八個方向。系統會自動排除障礙物、已走過區域與高危險區塊，直到找到可通往終點的路徑，再回溯出最短且相對安全的撤離路線。若無可通行路徑，則顯示「無法抵達」，有效提升逃生與救援效率。

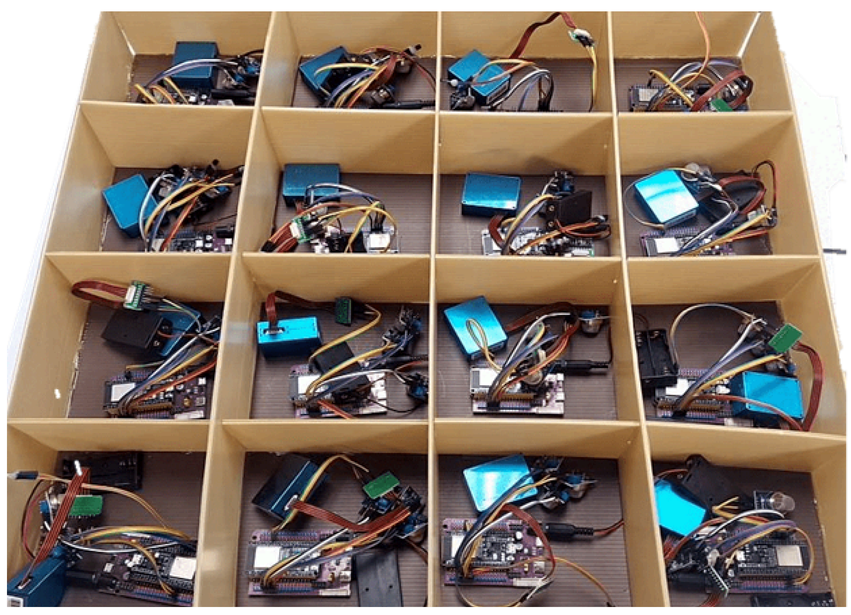


▲危機地圖判斷流程圖（由作者使用Canva繪製）

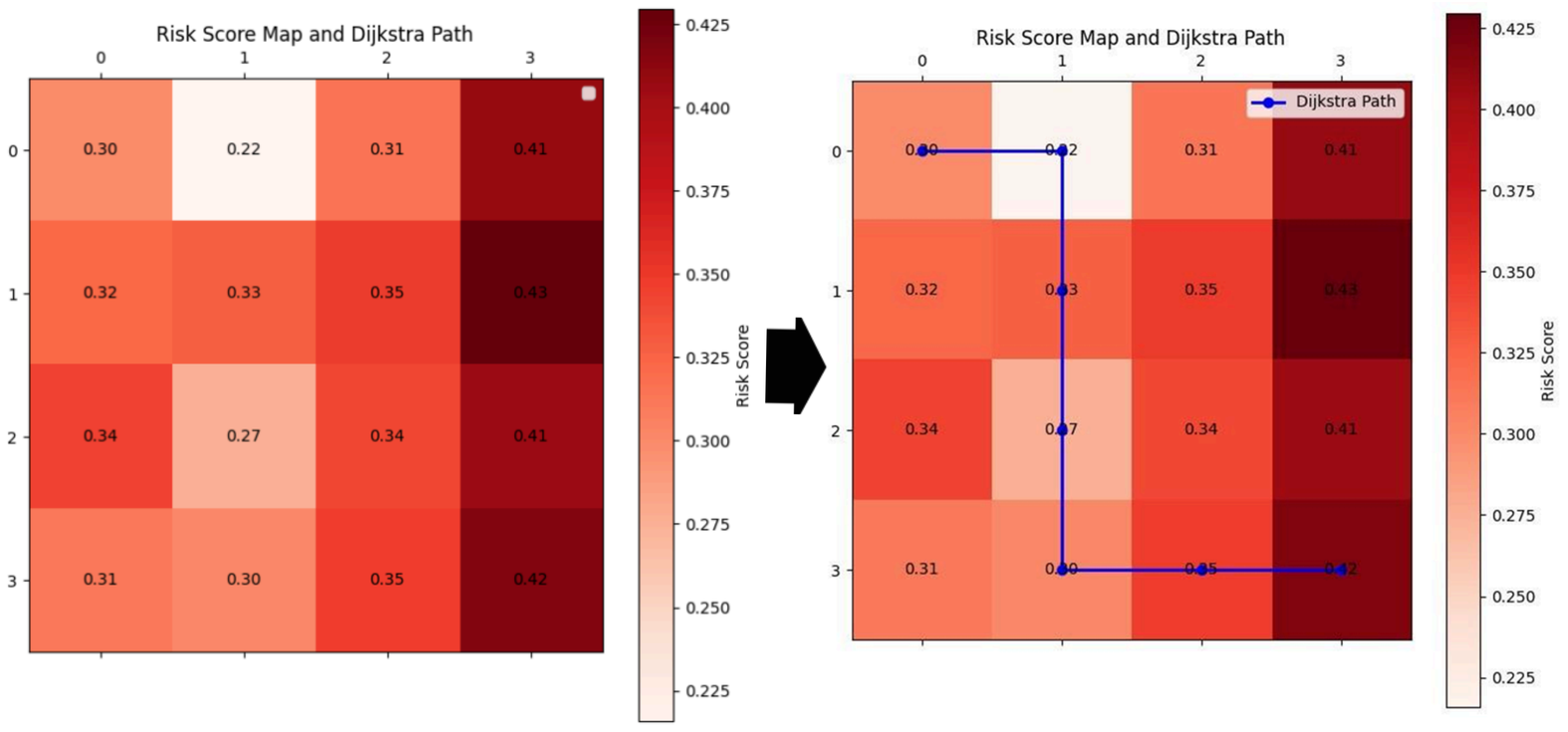
六、危機地圖實測結果

(一) 16宮格感測平台實測

我們建置一組 4×4 感測平台，模擬火場中不同區域的環境狀況。每格配置溫度、瓦斯、氫氣、粉塵與一氧化碳感測器，透過 ESP32 傳送數據至 Raspberry Pi 並儲存於 SQLite。系統根據即時數據繪製危機地圖，並以 (0,0) 為起點、(3,3) 為終點，透過 Dijkstra 演算法規劃避開高風險區的藍色撤離路線，提供即時且安全的逃生建議。



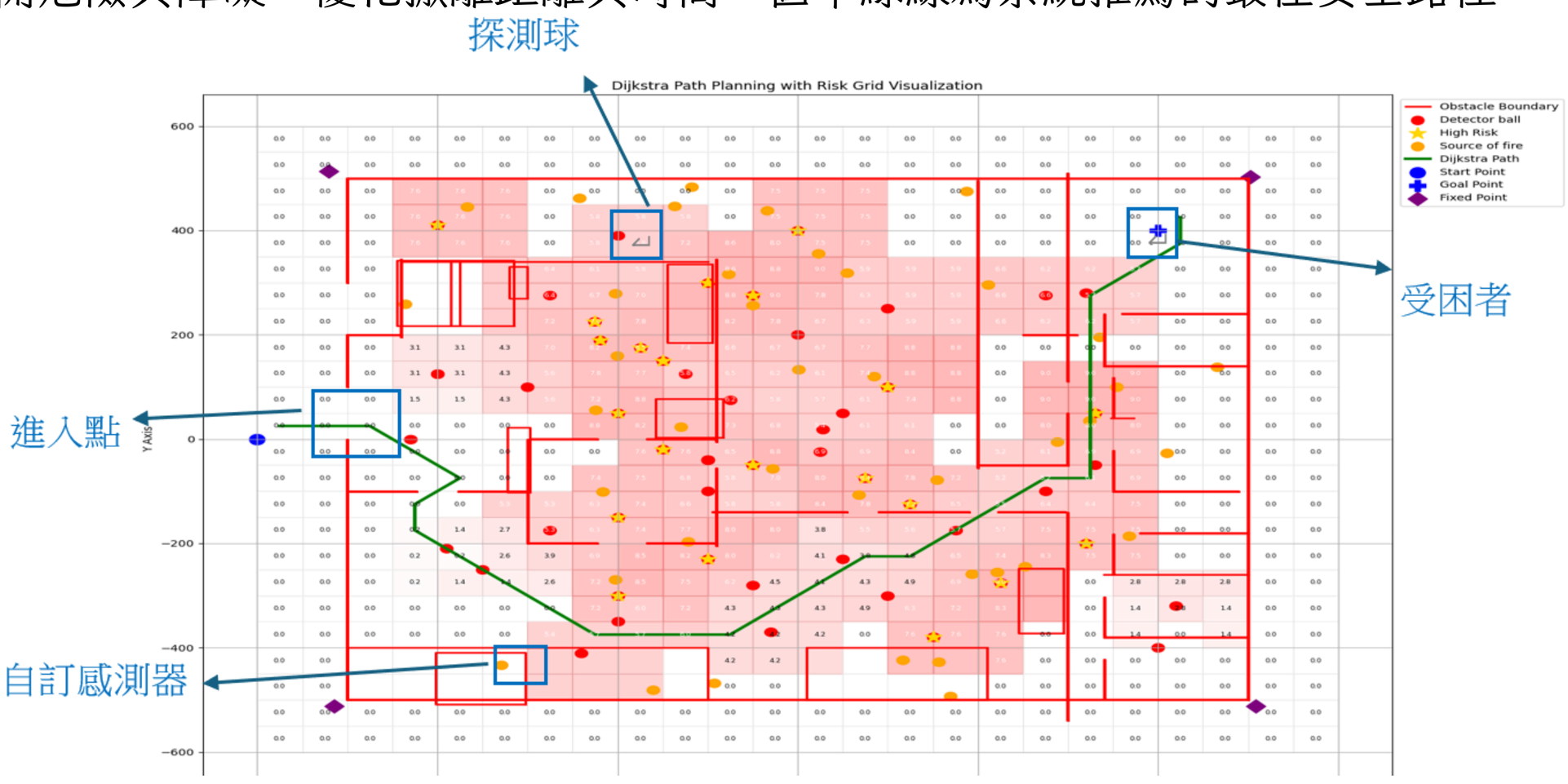
▲16宮格感測裝置（由作者拍攝）



▲16宮格危機地圖(由作者繪製)

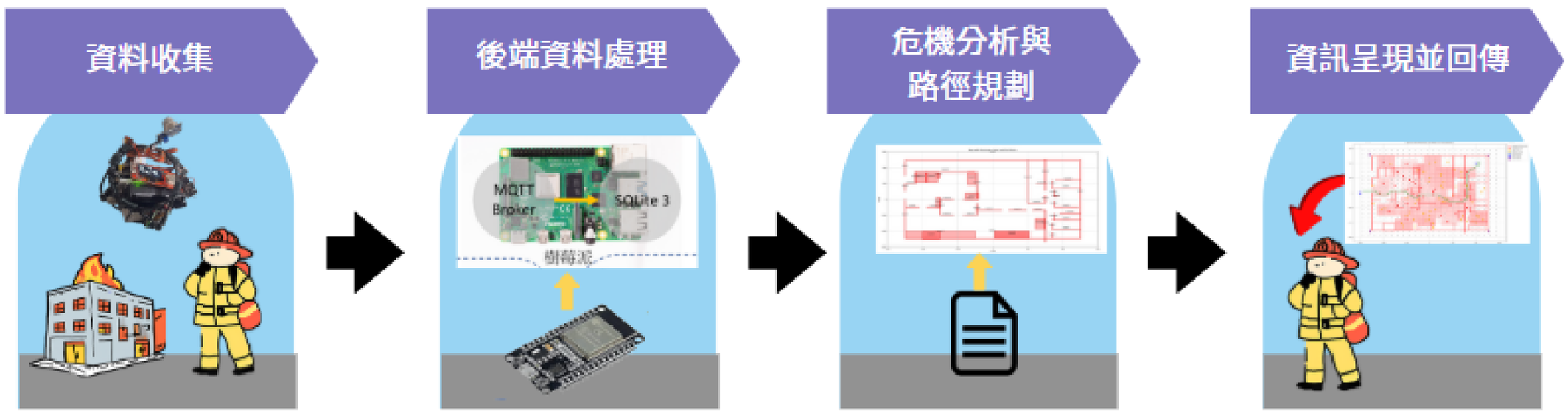
(二) 危機地圖模擬與路徑規劃測試

我們在模擬地圖中建立火場情境，配置感測器（橘黃點）與探測球（紅點），並標示各區的危險係數。紅線代表障礙物，火焰圖示顯示危險程度，有助於分析高風險區域。系統透過 Dijkstra 演算法規劃路徑，能有效避開危險與障礙，優化撤離距離與時間。圖中綠線為系統推薦的最佳安全路徑。



七、系統整合

本系統以四大部份運作，感測器結合可拋式探測球，針對火場內部即時回傳氫氣、一氧化碳、瓦斯、粉塵與溫度等關鍵數據，並搭配UWB無線定位模組同步標記感測座標。後端系統透過 IDLH 標準轉換危險係數，結合建築地圖與資料庫資料，生成危機地圖。路徑規劃則採用Dijkstra演算法，自動避開高風險區域，計算最短且安全之撤離路線。整體架構具備彈性擴充，可應用於多類型火場場域，支援即時決策與人員調度，提升救援效率與應變準確性。



▲16宮格感測裝置（由作者使用Canva繪製）

結論

(一) 即時定位與優化救援路徑：

應用UWB無線定位技術進行人員即時定位，結合危險係數評估模型與 Dijkstra 演算法，即時規劃最短且最安全之撤離與救援路徑，提升救援決策準確度 與執行效率。

(二) 數據即時傳輸與決策輔助:

本研究透過探測球與固定式感測器收集火場內的關鍵數據（如溫度、一氧化 碳、粉塵等），並將數據即時傳輸至雲端，結合IDLH模型生成危機係數地圖， 再透過AI進行分析與判斷，生成即時危機地圖，提供消防人員準確的避險資 訊，進一步提升救援決策的可靠性。

(三) 可與現有救援系統整合，降低部署成本:

本系統可與現有消防設備（如監視器、消防感測器等）整合，減少額外設備 的安裝需求，並可利用既有的無線網路進行資料傳輸，降低部署與維護成本，使 其具備良好的擴展性與實用價值。

(四) 系統可擴充與持續優化:

系統架構具彈性，未來可依照不同建築場域特性進行模組化擴充與優化，提升在建築火災應變與災防規劃上的實用性，成為災難管理中的重要輔助工具。

參考文獻

[1] 大濃煙卻超大嗆死2勇消…鑑定釀禍源頭是它. https://reurl.cc/0vRgbo

[2] 中華民國內政部消防署. (2022年)取自 https://www.nfa.gov.tw/cht/index.php?code=list&ids=220

[3] 陳昱志（2019）。以UWB為輔助之高精度定位儀及其應用〔碩士論文，國立臺灣大學〕。華藝線上圖書館。https://doi.org/10.6342/NTU201902526

[4} CDC NIOSH（2024）。IDLH Standards – Immediately Dangerous to Life or Health。取自：https://www.cdc.gov/niosh/idlh/intridl4.html