2025 仰望盃全國科學 HomeRun 實作大賽

決賽成果報告書

隊 伍 名 稱 ： 餘音繞樑

作 品 名 稱 ： 隱形的聲音浮水印

科學概念 1　：聽覺掩蔽效應（Auditory Masking Effect）

聽覺掩蔽效應是指當一個聲音的頻率和強度足夠時，它會掩蓋其他聲音，讓人耳無法聽見。浮水印技術利用這一效應，將浮水印嵌入到人耳聽不到頻率區域。例如，可以將浮水印放在某個較高頻率範圍內，這些頻率在音樂或語音中常常會被其他聲音掩蔽，因此不容易被人耳識別。

科學概念 2　：頻譜特性（Frequency Domain Characteristics）

頻譜特性利用訊號在不同頻率上的分布來實現資訊的隱藏和提取。將一些資訊嵌入聲音訊號中，頻譜特性幫助我們找到適合的頻率區域來隱藏資訊。例如，人耳對高頻的敏感度較低，我們可以把浮水印嵌入高頻部分，這樣就不易被聽出來，但通過頻譜分析工具就可以輕鬆檢測出來。頻譜特性還能確保浮水印在經過壓縮或傳輸後依然存在，讓資訊可靠且不易丟失。總之，頻譜特性提供了聲音訊號的頻率分布資訊，讓浮水印能夠隱蔽地嵌入，既不影響聲音品質，又能有效保護聲音的版權或驗證其來源。

決賽成果報告書內文

(最多 10 頁)

壹、發想動機：

根據全球防詐聯盟（GASA）對台灣詐騙的統計，2023 年到 2024 年 4 月統計的冒名類型高風險電話和簡訊高達近 300 萬筆，電話詐騙已成為最常見的詐騙手法，2024 年光是該類詐騙案件的占比就高達六成。其中，假冒親友的詐騙手法排名第四，占其中的一成。且最常被騙的不是老人，反而是年輕人。因此，我們想要做一個可以防止聲音詐騙的裝置，用來過濾被惡意修改的聲音,識別潛在的詐騙風險。

貳、作品創意性：

聲音浮水印作品的創意在於技術性以及對音訊內容的應用價值。基本概念是將浮水印以難以察覺的方式嵌入聲音中。就像在水中作畫一樣，肉眼看不見卻始終存在。
此次作品的特色如下：

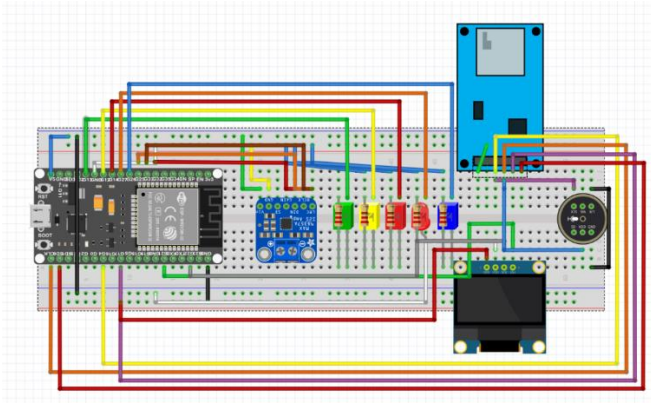
一、聲音浮水印允許以完全隱藏和有損壓縮的方式標記音訊內容，從而不會損害聲音品質和聽眾的感知。

二、聲音浮水印還可以與應用程式結合，從而追蹤音訊的真偽。這些技術的應用範圍甚至可以擴展到廣告、醫療領域的語音紀錄，及司法領域，通過加密音訊來比對和證明對話內容是否被篡改。

參、研究及運作及電路架構圖：

圖一為研究心智圖，內容是裝置功能以及裝置用途簡介。

圖一資料來源：研究者自行整理。



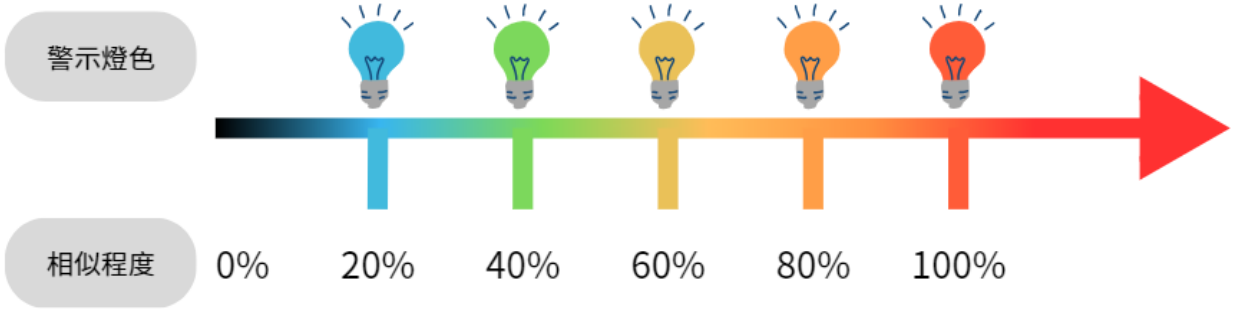
圖二為實體裝置電路圖，其中使用到的材料：

1. NodeMCU-32S 開發版
2. SSD1306 螢幕
3. SD 卡模組。
4. INMP441 錄音模組
5. MAX98357A 放音模組

圖二資料來源：研究者自行繪製

一、裝置介紹

本裝置會接收外在環境的語音訊息，並且從 SD 卡將真實聲音先進行噪音處理和解密，再將音檔輾平成一維陣列後一個個進行比對，確保該語音並非偽造之聲音，一旦判斷出是偽造會顯示在螢幕上面警示，圖三為警示程度的示意圖。



圖三資料來源：研究者自行繪製。

二、裝置運作實際流程

首先，處理流程分為四大部分：將音檔轉換成圖片、判斷音檔之間相似程度以及加解密音檔。由於都會遇到音檔上傳、儲存和顯示音檔或圖片部分，以下會先從音檔上傳開始說明。

(一)音檔上傳

1. 先從網頁上傳檔案。
2. 在從前端把檔案用二進位制方式傳到後端處理。
3. 後端在先使用 io.BytesIO 把二進位制檔轉變成一個假的 wav 檔，讓我們可可以是用 scipy.io.wavfile 去做讀取取樣率以 numpy 一維陣列的音檔，以下就是我們使用的音檔上傳的部分程式碼：

```
binary_data = await file.read()
wav_io = io.BytesIO(binary_data)

rate, audio = wavfile.read(wav_io)
```

```
def save_audio(audio, rate, num_channels=1, sample_width=2):
    file_name = f"waveform_{datetime.now().timestamp()}.wav"
    file_path = os.path.join("static", file_name)

    with wave.open(file_path, "wb") as wf:
        wf.setnchannels(num_channels)  # 設定聲道數量
        wf.setsampwidth(sample_width)  # 設定樣本寬度（例如 2 表示 16-bit）
        wf.setframerate(rate)          # 設定採樣率
        wf.writeframes(audio)          # 寫入音頻數據

    return file_name
```

圖四、五資料來源：研究者自行截圖。

(二)儲存和顯示音檔跟圖片

1. 音檔

(1) 先用 datetime 的時間戳記命名音檔使用。

(2) 再取出之前讀出的取樣率、numpy 一維陣列，還有聲道數量預設為 1 以及樣本寬度預設為 16-bit，用 wave 模組轉成一個真正的 wav 檔。

(3) 接著用 os 把音檔存進去 static 資料夾。

(4) 然後再把檔案路徑傳到前端，並用<img>顯示圖片。

(5) 最後可以選擇要不要把圖片傳到 esp32 上面顯示。

2. png 圖片

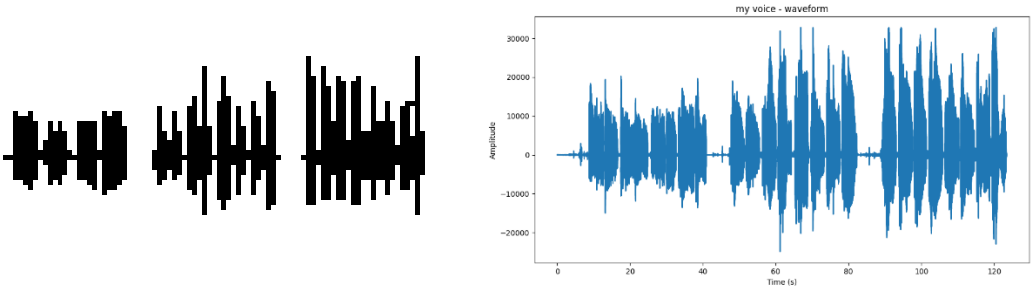
(1) 先用 datetime 的時間戳記命名音檔使用。

(2) 再用 matplotlib 模組儲存圖片到 static 裡面。

(3) 最後再把檔案路徑傳到前端，並用<img>顯示影片。

3. bmp 圖片

主要是用在 ssd1306 顯示上的，並用 Pillow 模組轉換成黑白且解析度為 128*64，因為 ssd1306 銀幕大小限制，所以只能顯示中間波的部分。如圖六及圖七。



圖六、圖七資料來源：研究者自行截圖

(三)將聲音轉換成圖片

1. 先將讀進來的音檔經過第一部份的轉換。

2. 接著會可以選擇是要波型圖還是頻譜圖。

3. 波型圖

(1) 直接用 matplotlib 內建功能轉換，並可以手動輸入圖片標題，但是限定只能是英文。

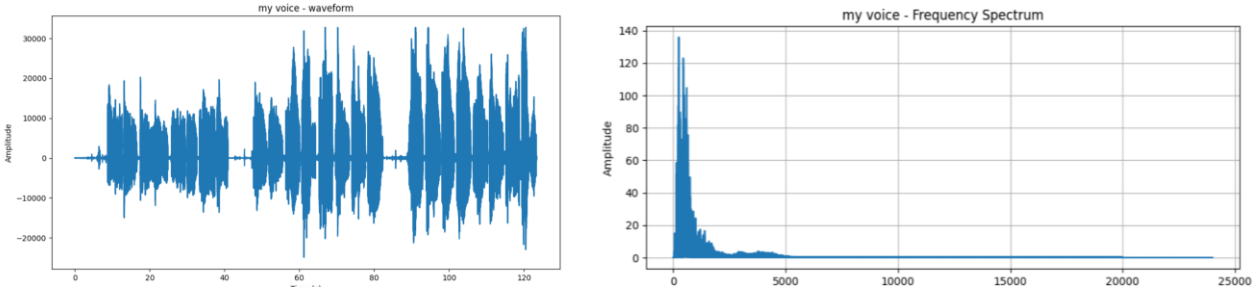
4. 頻率分析圖

(1) 先把音檔轉換成一維 numpy 陣列。

(2) 再裡用 numpy 內建的快速傅立葉轉換來取頻率域。

(3) 接著也是用 numpy 內建的快速傅立葉轉換來取出頻率，因為會產生對稱，所以我們只取前一半。

圖八和圖九是我們轉換出來的圖片。



圖八、圖九資料來源：研究者自行截圖。

(四)判斷兩個音檔相似程度

主要分為四個處理區塊：傅立葉轉換、MFCC、DTW、歐式距離與餘弦相似

1. 傅立葉轉換：把音訊中有含 ω 全取出來。

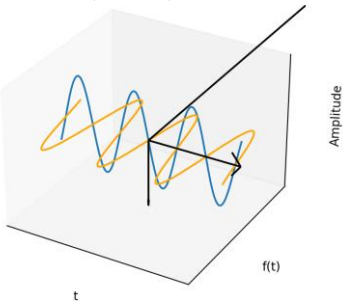
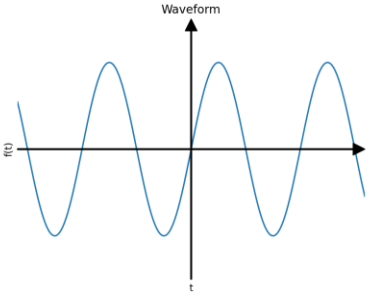
補充：

(1) 傅立葉級數$f(t)$是一個週期性函數表示為一組正弦與餘弦函數的線性組合。

(2) 歐拉公式 $e^{i\theta} = cos(\theta) + i\,sin(\theta)$，其中當 $\theta = \omega t$ 時，$e^{i\theta t}$ 為逆時針轉，$e^{-i\theta t}$。

圖八為傅立葉轉換公式，圖十及圖十一為轉換前後比較圖。

$$\hat{F}(t) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \begin{cases} = 0, not\ have\ \omega \\ \neq 0, have\ \omega \end{cases}$$

轉變前(圖十)	轉變後(圖十一)
	

圖十、圖十一資料來源：研究者自行截圖

2. MFCC

(1) 分幀與加窗：音訊訊號會被切割成數幀，分析局部頻率特徵。同時使用漢明窗平滑高噪音訊及避免邊界效應，公式如下。

$$x_w[n] = x[n] \cdot \left[0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right)\right], 0 \le n \le M-1$$

(2) 快速傅立葉變換：將時域信號轉換為頻域信號，以獲取頻率分佈特徵，時間複雜度僅 $O(NlogN)$，公式如下。

$$X(k) = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

(3) 梅爾頻率尺度轉換：使用梅爾頻率尺度（Mel Scale）來模擬人耳對低頻較細，高頻較疏的感知方式，公式如下。

$$Mel(f) = 2595log_{10}(1 + \frac{f}{700})$$

(4)梅爾濾波器組：這組三角形濾波器將頻譜映射到梅爾尺度。這些濾波器對低頻較密，對高頻較疏，公式如下。

$$Y_m = \sum_f |X(f)| \cdot h_m(f)$$

(5) 對數壓縮：人耳的聽覺響應對能量變化是對數性的，因此對上面濾波結果取對數來模擬這種性質，並降低頻率範圍內的巨大變化。這也可以增強較小能量信號的影響力。

(6) 離散餘弦變換：離散餘弦變換（DCT）能將頻域數據轉換為倒頻譜（Cepstrum），並且去除冗餘信息，使特徵更集中，公式如下。

$$C_k = \sum_{m=0}^{M-1} log(Y_m) cos(\frac{\pi k(2m+1)}{2M})$$

3. DTW (Dynamic Time Warping / 動態時間規整)

目的：在兩序列中找到最適合對應方式，使兩個序列對齊後差異最小。

(1) 距離矩陣：兩個序列 A、B，長度各為 N、M，有一個計算距離矩陣D會是 $N*M$ 的矩陣，$D[i][j]$ 為 $A[i]$、$B[j]$ 差異，公式如下。

$$D[i][j] = dist(A[i], B[j])$$

(2) 累積距離 ($C[i][j]$)：找到最小累積距離，公式如下。

$$C[i][j] = dist(A[i], B[j]) + min\{C[i-1][j], C[i][j-1], C[i-1][j-1]\}$$

(3) 回朔：從找到的終點再走回起點，找出最佳路徑。

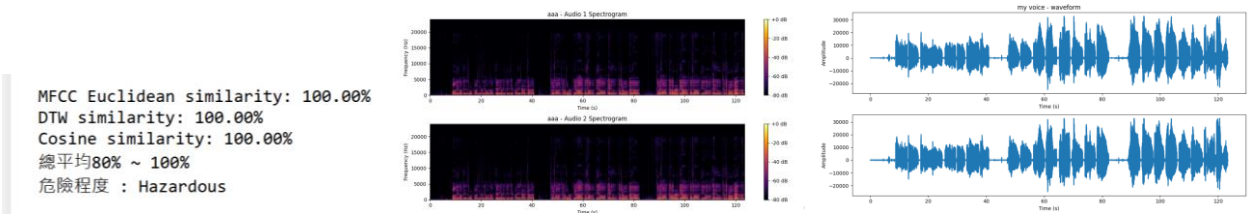
4. 歐式距離與餘弦相似

在 n 維歐幾里得空間中，點$P(x1,y1,\ldots,z1)$和點$Q(x2,y2,\ldots,z2)$之間的距離$d(P,Q)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$。而餘弦相似就是向量內積，公式：$A\cdot B=cosine\_similarity(A,B)|A||B|$。

顯示我們也分為三個部分相似程度分析、波型圖比較、頻譜圖必較：

1. 相似度分析是把上面算出的 MFCC、DTW、餘弦相似顯示出來，同時也顯示三者的平均，並用平均搭配圖三顯示燈號以及顯示警告文字。
2. 波型圖比較是用聲音轉換成圖片波型圖的方式，並把兩張圖併再一起。
3. 頻譜圖比較
   (1) 先把兩個陣列轉成 int16 類型。
   (2) 接著再運用 STFT 並取絕對值。
   (3) 最後在把單位轉換成 dB，並用 matplotlib 繪製成圖片 。

圖十二為相似度分析、圖十三為波型圖比較、圖十四圍頻譜圖比較。

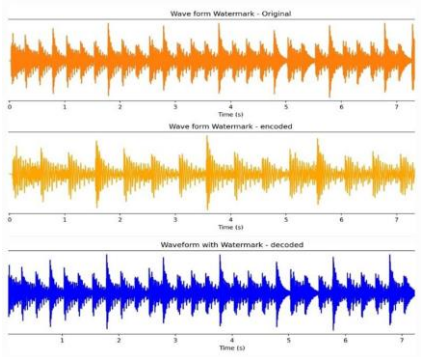


圖十二、圖十三、圖十四資料來源：研究者自行截圖。

(五)加密與生成浮水印

1. 用 AES-CBC 的加密方式，用 ADC 把音檔轉成 16 進位制，以 4 位或 8 位一組，生成 3 個 4 位或 8 位的 16 進位密碼組。
2. 用前一組跟三個 4 位或 8 位的 16 進位密碼組進行 xor 變成兩個 8 位 16 進位制字串。
3. 再把兩個 4 位或 8 位的 16 進位字串壓成一組，把它存在新字串裡。
4. 新字串即為加密完的音檔的密鑰，並輸出密鑰。

(六)解密音檔



1. 輸入正確的加密金鑰。
2. 將音檔放慢兩倍解密，再將加快輸出。

圖十五由上到下為原始、加密及解密真實聲音波形對照圖。

圖十五資料來源：研究者自行繪製。

其次，網頁分成以下四個部分：

(一)前端

為了讓使用者有更友善的操作介面，本研究開發一套網站作為前端使用平台，功能如下：

1. 可以上傳、顯示圖片和音檔以及顯示轉換完的數據，如圖十六。

圖十六資料來源：研究者自行截圖。

2. 如果是顯示圖片，可以自行命名圖片標頭，如圖十七。

圖十七資料來源：研究者自行截圖。

3. 負責傳輸圖片以及文字給 esp32 接收。

4. 發送指令到後端執行音檔或圖片處理，如圖十八。

圖十八資料來源：研究者自行截圖。

(二)後端

本系統後端框架為 python FastAPI，功能如下：

1. 前端資料請求以及回應。

2. 運用 __init__.py 來管理所有的檔案，方便我們可以同時引用多個檔案在 main.py 裡面運行。

3. 使 sqlalchemy 連接資料庫，儲存生成出的圖片以及音檔的路徑，方便檔案超過時間刪除檔案。

圖十九為 FastAPI 架設，圖二十為資料庫連接 python FastAPI。

```python
app = FastAPI()
UPLOAD_DIR = 'static'
os.makedirs(UPLOAD_DIR, exist_ok=True)
Base.metadata.create_all(bind=engine)

template = Jinja2Templates(directory='templates')
app.mount("/static", StaticFiles(directory="static"), name="static")

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

@app.get('/')
async def root(request: Request):
    return template.TemplateResponse('index.html', {"request": request})
```

```python
load_dotenv()
DB_URL = os.getenv("DB_URL") #讀取.env的URL

engine = create_engine(DB_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()

class Static(Base):
    __tablename__ = "static"
    id = Column(Integer, primary_key=True, index=True)
    static_url = Column(String(500), nullable=False)
    descriptiom = Column(Text)
    create_at = Column(DateTime, server_default=func.now())
```

圖十九、圖二十資料來源：研究者自行截圖。

(三)Docker

Docker 是一種開源的容器化平台，可用於打包、部署及執行應用程式，比傳統的虛擬機快上許多，使用 Docker 主要原因是要架在 railway 上以及可以整理 python 環境。

(四)資料庫

資料庫已經成為現在主流的資料結構，可以快速的查詢資料，也可以輕鬆存取、修改、管理資料等功能。大多數的資料庫都使用 sql（Structured Query Language）來編寫（Oracle, 2020 年）。

我們用的是 MySQL 搭配 python sqlalchemy 來管理我們的資料庫，因為我們之後要架在 railway 上面，無法手動刪除照片與音檔，所以我們需要寫一個資料庫連接 python 後端設定時間來刪除照片，防止超出儲存空間或是應用功能下降等。

至於後端搭配使用的 cloudinary 則是一個專給開發者的**雲**端媒體管理平台，提供一站式解決方案，涵蓋圖片與影片的眾多編輯功能。平台所提供的 API 和 SDK 更是可以協助開發者能快速整合媒體處理功能，無需自行建置複雜的前置作業，以下為核心技術的簡單介紹：

1. 即時編輯與轉換：透過 URL 參數調整圖片尺寸、裁切、浮水印、格式轉換（如 AVIF 自動優化），並保留原始檔案（Toolify.ai, 2024 年）（Astral Web 歐斯瑞有公司）。

2. AI 自動化處理：支援智能裁切、去背、視覺搜尋及內容感知壓縮，提升媒體品質與載入速度（Astral Web 歐斯瑞有限公司）（Evinco Editor）。

3. 全球 CDN 分發：透過內容傳遞網路加速交付，確保各地用戶快速存取（ysl0628, 2023 年）（Astral Web 歐斯瑞有限公司）。

4. 開發者工具：提供上傳元件（如 <CldUploadWidget/>）、Next.js 整合

套件（next-cloudinary）及詳盡 API 文件（ysl0628, 2023 年）（dfsgwe1231, 2020 年）。

透過 Cloudinary，開發者能專注於核心功能開發，同時確保媒體處理的效率與全球可及性（ysl0628, 2023 年）（Sarah Wilhelm, Shuting Fu）。

最後，再來介紹互聯網裝置，分成四個部分：

1. ESP32 微控制器

ESP32 為本系統的主要控制單元，負責：

(1) 連接 MQTT 通訊協定。

(2) 接收外部伺服器或感測裝置所傳送的音訊資料。

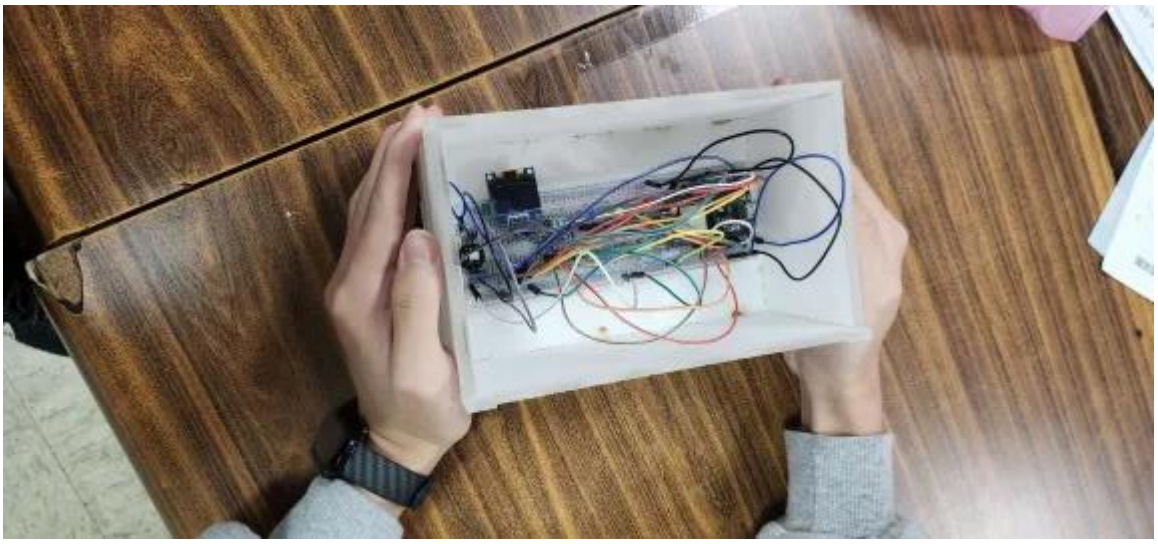
除了通訊功能外，ESP32 同時也兼具：

(1) 整合 OLED 螢幕、LED 指示燈等週邊設備

(2) 進行資料處理與控制輸出。

其具備 Wi-Fi 與藍牙功能，適合應用於物聯網音訊分析系統中。

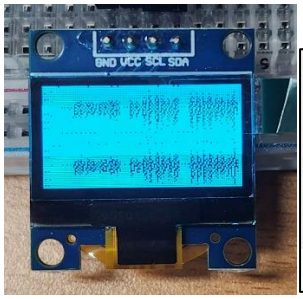
圖二十一為裝置實際展示圖。

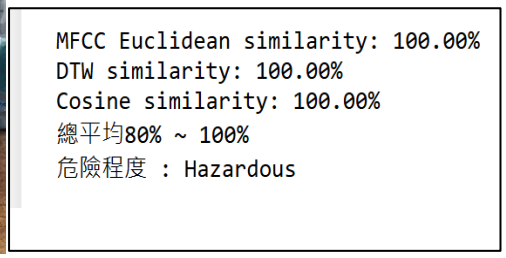


圖二十一資料來源：研究者自行拍攝

2．OLED 顯示模組

OLED 螢幕用來即時顯示音檔轉換後的圖像，例如透過將音訊轉換成頻率分析圖，視覺化呈現音訊在時間與頻率上的變化，如圖二十二，或是將音訊轉換成波型圖。系統也可同時顯示兩段音檔的圖像，便於使用者直接比較其視覺差異，進一步了解兩者的相似或不同之處，如圖二十三。還可以進行數據分析，在上面顯示相似程度以及我們計算相似程度的數據，如圖二十四。





圖二十二、二十三、二十四資料來源：研究者自行拍攝

3．LED 指示燈模組

為更直觀地反映音檔之間的相似程度，系統設計了五種顏色的 LED 指示燈，依據相似度百分比分為五個等級。當系統完成音檔分析與比對後，會根據相似程度自動點亮對應顏色的 LED 燈，具體分級如圖三。

4．錄放音模組

我們分成兩大部分：

(1) 錄音部分：使用 INMP441 將音訊信號（如語音）轉換為 I2S 數位音訊信號。你可以用開發板（例如 ESP32 或 Raspberry Pi）來接收來自 INMP441 的數位音訊信號，並處理或儲存這些數據。

(2) 播放部分：將錄製的數位音訊信號（I2S 格式）傳送給 MAX98357A，MAX98357A 會將這些數位信號轉換為模擬音訊並放大，通過喇叭播放出來。

肆、作品成果說明

一、裝置特色

1．輸出分級化：設計更直觀的多層次警報系統，以不同顏色顯示聲音的真偽級別。

2．音頻噪音處理：新增背景噪聲過濾功能，提升裝置在嘈雜環境中對聲音的辨度。

3．音頻轉換：把音頻轉換成圖片，並在網頁、esp32 上顯示，也提供下載照片。

二、應用延伸

1. 防偽認證：分析音頻真實性，應用於商業場景。

2. 司法鑑定：驗證音頻證據的真實性，支援法律用途。

3. 個人身分驗證：聲紋辨識技術確保授權使用，提升安全性。

六、參考文獻

朱航輝、李清坤(2005)。聲音浮水印之研究。大同大學:碩士論文。

Prezi。(2021)。DMA 運作&原理。https://prezi.com/p/lz8n__i0anap/dma/。

PanSci 泛科學。(2016)。強勢來襲的數位潮流：你不可不知的 DAC（上）—《音響入門誌》。https://pansci.asia/archives/106307。

AES 五种加密模式（CBC、ECB、CTR、OCF、CFB）。https://www.cnblogs.com/starwolf/p/3365834.html。

Whoscall。(2024 年 10 月 16 日)。30%台灣人遇到詐騙後一小時內受騙，發布亞洲詐騙調查!。https://whoscall.com/zh-hant/blog/articles/1400-the-state-of-scams-in-taiwan-2024。

歐斯瑞。Cloudinary - 雲端影像處理平台，線上圖片即時優化。https://www.astralweb.com.tw/cloudinary-introduction/。

Toolify.ai。(2024 年 2 月 13 日)。用 Cloudinary 將動態圖片轉換為 AVIF 進行優化。https://reurl.cc/EVgnkn。

Evinco。(2025 年)。只需使用這工具分析網站，找出最合適的圖片格式, 大小, 編碼，提升網站速度。 https://reurl.cc/VY0Er6。

ys10628。(2023 年)。營養師不開菜單的第十七天 - 在 Next.js 中使用 Cloudinary CDN 管理媒體資源。https://ithelp.ithome.com.tw/articles/10331579。

dfsgwe1231。(2020 年)。Cloudinary Upload API 簡介：輕鬆替代 AWS S3。https://blog.csdn.net/dfsgwe1231/article/details/105993907。

Sarah Wilhelm, Shuting Fu。CLOUDINARY – 基於雲端的映像和視訊管理。https://www.w-4.ch/zh/produkte/cloudinary。