

ESP32 MicroPython 中文 OLED 跑馬燈顯示系統

專案報告

1. 專案名稱與目的

專案名稱

ESP32 MicroPython 中文 OLED 跑馬燈顯示系統

專案目的

- **克服技術限制**：解決 SSD1306 OLED 原生不支援中文字元的限制
- **動態字型渲染**：利用外部 Flask API 動態生成中文字點陣圖
- **多元顯示模式**：在 ESP32 上實現中文文字的靜態顯示與跑馬燈滾動效果
- **彈性字體調整**：提供可調整的字體大小功能
- **實用性應用**：為嵌入式系統中的中文資訊顯示提供完整解決方案

應用場景

- 中文提示看板
 - IoT 中文顯示器
 - 即時通知裝置
 - 智能家居資訊顯示
-

2. 使用元件與工具

硬體元件

- **ESP32 開發板**：主控制器，負責Wi-Fi連接與程式執行
- **SSD1306 OLED 顯示模組**：0.96吋/128x64像素，I2C 接口
- **杜邦線**：I2C 連接線材 (SDA, SCL, VCC, GND)
- **USB 傳輸線**：供電與程式燒錄

軟體與工具

- **MicroPython 韌體**：運行在 ESP32 上的 Python 環境
 - **Python 3**：運行 Flask 伺服器的主機環境
 - **Flask 框架**：提供字型 API 服務
 - **Pillow (PIL)**：Python 圖像處理庫，用於生成點陣圖
 - **ssd1306.py**：OLED 驅動程式庫
 - **urequests**：ESP32 HTTP 客戶端庫
 - **network**：ESP32 Wi-Fi 連接庫
 - 中文字型檔案：NotoSansTC-Regular.ttf (支援繁體中文)
 - 開發工具：Thonny IDE
-

3. 程式設計說明

系統架構

本系統採用前後端分離設計：

後端 (Flask 字型伺服器)

- 檔案：`font_server.py`
- 運行平台：PC (Python 3 + Flask)
- 主要功能：
 - 載入中文字型檔案
 - 提供 RESTful API 端點 `/api/font`
 - 接收文字並轉換為點陣圖
 - 回傳 JSON 格式的點陣圖數據

前端 (ESP32 MicroPython)

- 檔案：`oled_display.py`, `main.py`
- 運行平台：ESP32 (MicroPython)
- 主要功能：
 - Wi-Fi 連線管理
 - HTTP 請求發送
 - OLED 驅動控制
 - 跑馬燈顯示邏輯

核心程式模組

1. Flask 字型伺服器 (`font_server.py`)

python

```
def text_to_bitmap(text, font_size=FONT_SIZE):  
    """將文字轉換為點陣圖"""  
    # 載入字型檔案  
    # 計算文字尺寸  
    # 渲染為黑白圖像  
    # 轉換為點陣圖陣列  
    return {'bitmap': bitmap, 'width': width, 'height': height}
```

關鍵特性：

- 支援空白字元處理
- 動態字體大小調整
- 錯誤處理與回饋機制
- URL 編碼處理

2. OLED 顯示類別 (`oled_display.py`)

python

```
class OledChineseDisplay:  
    def __init__(self, scl_pin, sda_pin, font_api_url, scroll_mode=True):  
        # 初始化 I2C 和 OLED  
  
    def connect_wifi(self, ssid, password):  
        # Wi-Fi 連線處理  
  
    def display(self, texts):  
        # 主要顯示函數
```

關鍵功能：

- **Wi-Fi 連線管理**：自動連接並顯示連線狀態
- **URL 編碼處理**：手動實作中文字元 URL 編碼
- **點陣圖渲染**：支援靜態顯示與跑馬燈模式
- **記憶體管理**：垃圾回收機制防止記憶體溢出

3. 跑馬燈顯示邏輯

python

```
def _render_bitmap(self, bitmap_data, speed=0.08):
    if self.scroll_mode:
        # 跑馬燈模式：文字從右到左滾動
        total_scroll_width = self.width + width
        for offset in range(total_scroll_width):
            # 重新繪製每一幀
    else:
        # 靜態居中顯示
```

通訊協議

- 協議：HTTP/RESTful API
- 資料格式：JSON
- API 端點：`GET /api/font?text={encoded_text}`
- 回應格式：

json

```
{
    "bitmap": [[0,1,0,1], [1,0,1,0], ...],
    "width": 48,
    "height": 24,
    "success": true
}
```

4. 測試過程與結果

測試環境

- 硬體：ESP32 DevKitC + 0.96吋 OLED (I2C)
- 軟體：Python 3.x + MicroPython
- 網路：Wi-Fi 區域網路

測試流程

1. Flask 伺服器測試

- 啟動 `font_server.py`
- 確認字型檔案存在於正確路徑
- 測試 API 端點回應

2. ESP32 程式上傳

- 燒錄 MicroPython 韌體
- 上傳 `oled_display.py`, `main.py`
- 配置 Wi-Fi 憑證和 API 位址

3. 整合測試

- ESP32 重啟並觀察連線狀態
- 測試不同長度的中文字串
- 驗證跑馬燈滾動效果

測試案例與結果

API 測試

測試內容	API 請求	結果
基本中文	<code>/api/font?text=你好</code>	✓ 正常回傳點陣圖
中英混合	<code>/api/font?text=Hello世界</code>	✓ 混合顯示成功
空白處理	<code>/api/font?text=你好%20世界</code>	✓ 空白正確呈現
純空白	<code>/api/font?text=%20%20%20</code>	✓ 三個空白顯示

ESP32 顯示測試

測試項目	測試內容	結果
Wi-Fi 連線	自動連接指定網路	✓ 連線穩定
中文顯示	「你好 世界」	✓ 清晰顯示
跑馬燈效果	文字從右到左滾動	✓ 平滑滾動
多段文字	連續顯示多個字串	✓ 自動切換
空白測試	「 三個空白 」	✓ 正確處理

性能表現

- 字型轉換速度：約 200-500ms (依文字長度)
- 網路延遲：約 100-300ms (區域網路)
- 滾動流暢度：每幀 80ms，視覺效果流暢
- 記憶體使用：約 15-25KB (依文字長度)

5. 遇到的問題與解決方式

問題 1：OLED 通訊不穩定

問題描述：

- 出現 `OSError: [Errno 19] ENODEV` 或 `ETIMEDOUT` 錯誤
- OLED 模組無法正常初始化

解決方案：

1. 檢查物理連接：確保 SDA/SCL/VCC/GND 接線正確牢固
2. 調整 I2C 頻率：設定為 400kHz 提升穩定性

python

```
self.i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=400000)
```

3. 穩定供電：確保 ESP32 電源供應充足
4. I2C 掃描診斷：使用掃描工具確認設備可見性

問題 2：中文顯示亂碼

問題描述：

- OLED 顯示無法辨識的白色長條圖案
- 點陣圖資料異常

解決方案：

1. 字型檔案路徑：確認 `NotoSansTC-Regular.ttf` 與 `font_server.py` 在同一目錄
2. 字型載入檢查：

python

```
if os.path.exists(FONT_PATH):  
    font = ImageFont.truetype(FONT_PATH, font_size)  
else:  
    print(f"警告：找不到字型檔案 {FONT_PATH}")
```

3. API 測試：獨立測試 Flask API 確認點陣圖正確生成

問題 3：跑馬燈滾動邏輯錯誤

問題描述：

- 文字無法完整從右側進入
- 滾動過程出現跳動或不完整顯示

解決方案：

1. 座標計算修正：

```
python

for offset in range(total_scroll_width):
    px = self.width - offset + x # 從螢幕右側外開始
```

2. 邊界檢查：確保只繪製可見範圍內的像素

3. 滾動範圍調整：`total_scroll_width = self.width + width`

問題 4：空白字元處理

問題描述：

- Flask 無法正確處理 URL 編碼的空白字元 `%20`
- Pillow 對空白字元回傳寬度為 0

解決方案：

1. Flask 自動解碼：

```
python

text = request.args.get('text', '') # Flask 自動解碼 %20
```

2. 空白寬度估算：

```
python

if not text.strip(): # 純空白處理
    estimated_space_width = font_size // 3
    width = estimated_space_width * len(text)
```

問題 5：記憶體管理

問題描述：

- 長文字或小字體導致記憶體不足
- 程式執行中途停止

解決方案：

1. 垃圾回收：

```
python

import gc
gc.collect() # 在關鍵位置進行記憶體回收
```

2. 記憶體監控：

```
python

print(f"可用記憶體: {gc.mem_free()} bytes")
```

3. 資源釋放：及時釋放不再使用的變數

6. 結論與心得

專案成果

- 技術突破：成功克服 SSD1306 OLED 不支援中文的限制
- 系統整合：實現了完整的前後端分離架構
- 功能豐富：支援靜態顯示、跑馬燈效果、多種字體大小
- 穩定性佳：具備完善的錯誤處理和記憶體管理機制
- 可擴展性：架構設計便於後續功能擴展

技術收穫

- 軟硬體協同：深入理解了軟體程式與硬體電路的協同工作原理
- 嵌入式限制：體驗了在有限資源下進行程式設計的挑戰
- API 設計**：學習了 RESTful API 的設計與實作
- 圖像處理：掌握了點陣圖生成與處理技術
- 除錯技巧：培養了系統性的問題診斷與解決能力

學習心得

- 問題導向學習：每個技術難點都促使深入學習相關知識
- 迭代開發重要性：透過不斷測試與改進達成最終目標
- 文檔記錄價值：詳細的開發記錄有助於問題追蹤與知識累積
- 社群資源利用：開源函式庫和社群討論大大加速了開發進程

未來展望

1. 雲端整合：整合雲端 API 實現遠端訊息推送
 2. 多語言支援：擴展支援更多語言字型
 3. 圖形顯示：加入簡單圖形和圖標顯示功能
 4. 觸控互動：結合觸控感測器實現互動式顯示
 5. 商業應用：發展為商用的資訊顯示解決方案
-

7. 執行結果展示


系統運行流程

1. 初始化階段：ESP32 啟動，OLED 顯示 "System Booting..."
2. Wi-Fi 連線：顯示 "Connecting WiFi" 及 SSID
3. 連線成功：顯示 "WiFi Connected!" 及 IP 位址
4. 文字顯示：依序顯示設定的中文字串
5. 跑馬燈效果：文字從右到左平滑滾動

顯示效果特色

- 中文清晰：24號字體下中文字符清晰可辨
- 滾動流暢：80ms 間隔提供流暢的視覺效果
- 空白處理：正確顯示文字間的空白字元
- 多段切換：自動切換顯示多個文字段落
- 錯誤提示：API 連線失敗時顯示錯誤訊息

執行結果影片

 [點此觀看專案執行影片](#)

8. 程式碼結構

檔案清單

```
ESP32_Chinese_OLED/  
├─ font_server.py      # Flask 字型 API 伺服器  
├─ oled_display.py     # ESP32 OLED 顯示類別  
├─ main.py             # ESP32 主程式  
├─ NotoSansTC-Regular.ttf # 中文字型檔案  
└─ README.md           # 專案說明文件
```

主要功能模組

- 字型轉換模組：`text_to_bitmap()` 函數
 - 網路通訊模組：`connect_wifi()`, `_fetch_font_bitmap()` 函數
 - 顯示控制模組：`_render_bitmap()` 函數
 - 系統管理模組：記憶體管理、錯誤處理
-

附錄

A. 環境設定需求

- Python 3.7+
- Flask 2.0+
- Pillow 8.0+
- MicroPython 1.18+
- ESP32 開發板
- SSD1306 OLED 模組

B. 安裝說明

- 安裝 Python 依賴套件：`pip install flask pillow`
- 下載中文字型檔案至專案目錄
- 燒錄 MicroPython 韌體至 ESP32
- 上傳 Python 檔案至 ESP32

C. 故障排除

- 無法顯示中文：檢查字型檔案路徑
 - Wi-Fi 連線失敗：確認 SSID 和密碼正確
 - OLED 無反應：檢查 I2C 連線和電源
 - API 連線失敗：確認 Flask 伺服器運行狀態
-

報告製作日期：2025年6月18日

專案開發者：[您的姓名]

聯絡方式：[您的聯絡資訊]