一、專題動機

傳統自學工具常以純文字或影片呈現，容易造成使用者缺乏參與感與持續動力。本專題希望打造一個結合 **AI 語言模型、語音合成（TTS）、實體互動裝置（PIC18F4520）、馬達與燈光反應** 的「互動式學習 AI 夥伴」。
使用者能透過實體按鈕作答，由實體 AI 夥伴提供題目、回饋、鼓勵與動作／燈光效果，使學習從被動輸入提升為 **具情緒、聲音與肢體反應的沉浸式互動體驗**。

二、功能與原理說明

本系統是一個人機互動式智慧問答系統，結合：

- 大型語言模型（LLM）
- LangGraph 多 Agent 狀態機
- PIC18F 微控制器（UART）
- 前端 UI（Streamlit / Web）
- 語音輸出（TTS）

讓使用者可以透過實體按鈕（PIC18F）與 AI 互動，完成：

- 主題選擇
- 問題回答
- 對錯回饋
- 硬體行為回應（燈號 / 動作 / 語音）

★ 核心功能之原理:

1. LangGraph = 系統的大腦（State Machine）

- LangGraph 被用來描述「流程狀態」而非單次對話
- 每一個節點（Node）代表一個明確階段：
  - 選主題
  - 出題
  - 等待回答
  - 評估結果
- 狀態轉移是可視化、可追蹤、可 debug 的

非「一問一答」，而是完整互動流程

2. PIC18F = 實體輸入 / 輸出代理（Physical Agent）

PIC18F 負責：

輸出（UART TX）

- 使用者按下實體按鈕
- PIC18F 透過 UART 傳送：
- "A\n" / "B\n" / "C\n"
- Python Orchestrator 解析為使用者選擇

輸入（UART RX）

Python 端依系統狀態送控制碼給 PIC：

UART Code	意義
010\n	說話 / 互動中
111\n	錯誤回饋（震動 / 燈 / 動作）
000\n	停止 / reset

PIC18F 依控制碼驅動：

- LED
- 馬達
- 其他實體反饋裝置

3. Steamlit 前端:

以 JSON 作為「前後端同步狀態層」

current_state.json 是整個系統的狀態來源。

為什麼用 JSON？

- 前端只需 poll
- 不需要 WebSocket
- 狀態可直接 debug（打開檔案就能看）
- 與 LangGraph 的「狀態概念」天然契合

這些狀態完美切合，本專案的 “後端驅動”

★ 功能流程

1. Pick Domain（主題選擇）

- LLM 產生三個主題（A/B/C）
- 寫入 JSON
- 播放 TTS
- UART TX：010 → 000
- 進入等待狀態

2. Wait Domain Choice（等待 PIC 輸入）

- Python 阻塞讀 UART
- 收到 A/B/C
- 決定使用者選擇的主題

3. Generate Question（出題）

- LLM 根據主題產生：
  - 問題
  - 三個選項

- 正確答案

- 更新 JSON
- 播放問題語音
- UART TX：010 → 000

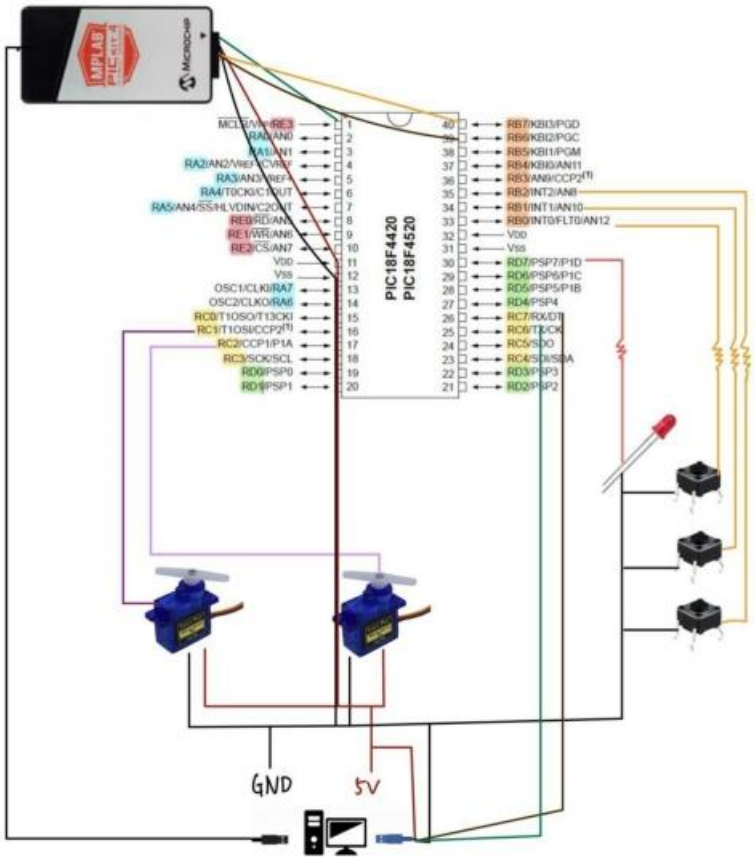
4. Wait Answer（等待回答）

- 再次等待 PIC18F 傳入 A/B/C
- 記錄使用者答案

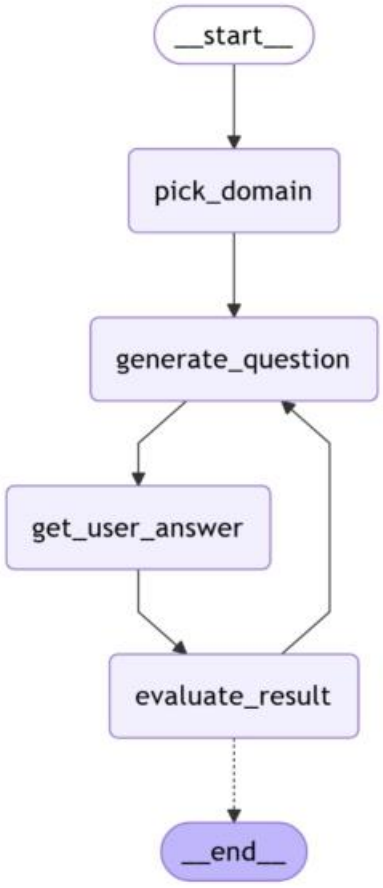
5. Evaluate Result（評估與回饋）

- 比對 user_answer 與 correct_answer
- 更新 JSON：
  - status = correct / wrong
- 播放語音回饋
- UART 行為：
  - 正確：010
  - 錯誤：111
  - 最後一定送：000
- 回到出題節點（形成循環）

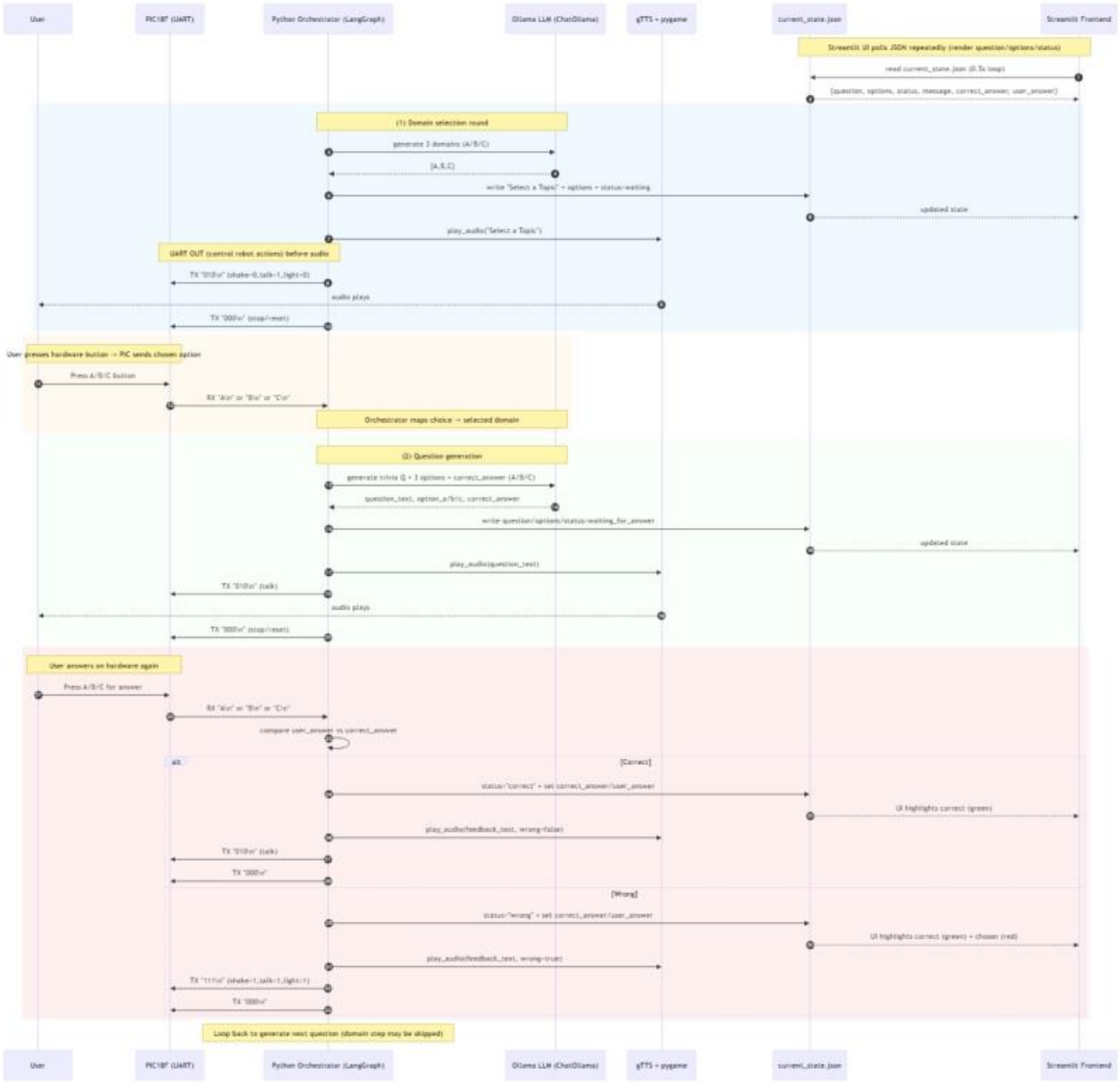
★ 電路圖



★ **LangGraph 流程圖**



★ 整合流程示意（以序列圖表示）



三、實際組員之分工項目

成員	負責範圍
陳柏巨	PIC 通訊 + 中斷 + UART + 按鍵輸入處理
蕭捷晨	馬達動作控制（AI實體動作情緒表達）
黃書璿	LED 光效果控制（AI實體燈光情緒表達）
余沛承	Python Orchestrator + AI 模型串接 + TTS
馮立忻	UI 或遊戲介面設計

四、基本單元項目列舉:

1. UART

使用目的

作為 **PIC18F 與 Python Orchestrator（AI 系統）之間的通訊介面**。

實作內容

- PIC18F 透過 UART 傳送使用者按鍵輸入：
  - "A\n" / "B\n" / "C\n"
- Python 端接收後：
  - 解析使用者選擇
  - 驅動 LangGraph 狀態轉移
- Python 端反向透過 UART 傳送控制碼至 PIC18F：
  - 010\n → 互動 / 說話中
  - 111\n → 錯誤回饋（燈 / 動作）
  - 000\n → 停止 / reset (等待輸入)

2. Interupt

使用目的

確保 **UART 接收不會因主程式阻塞而遺失資料**。

實作內容

- PIC18F 使用 **UART RX Interrupt,**
  - 當 UART 收到資料時：
    - 立即觸發中斷
    - 讀取 RCREG
    - 將資料存入 buffer 或狀態變數
- UART TX **Interrupt,**
  - 使用者透過 **實體按鈕** 與系統互動，
    PIC18F 讀取按鍵狀態後，**主動透過 UART 傳送指令至電腦端**。
- Button Interrupt

3. PWM

使用目的

本系統利用 PWM 控制馬達與致動元件，將 AI 系統的判斷結果轉換為不同強度的實體回饋，提升人機互動的直覺性與即時性。

錯誤回饋（Error Feedback）

- Python 判斷答錯
- UART 傳送：
  - 111\n
- PIC18F：
  - 開啟 PWM
  - 提高 duty cycle
  - 驅動馬達（震動 / 擺動）

強烈的回饋

讓使用者**「感覺得到自己答錯」**

正確回饋（Correct / Neutral）

- UART 收到：
- 010\n 或 000\n
- PIC18F：
  - 關閉 PWM 或降低 duty
  - 馬達停止或平穩

五、進階單元項目舉例

1. AI 與 LangGraph 多 Agent 系統整合

使用技術

- **大型語言模型（LLM API）**
- **LangGraph 狀態機框架**
- **Python Orchestrator**

說明

本專題並非僅使用單一 AI 回應，而是將整個互動流程建模為 多狀態、多節點的 LangGraph 系統。

每個 Agent / Node 對應一個明確功能：

- 主題選擇（Domain Selection）
- 問題生成（Question Generation）
- 使用者輸入處理（User Input Handling）
- 結果評估與回饋（Evaluation & Feedback）

此設計使 AI 成為流程控制核心，而非單純的聊天工具。

加分理由

- 使用「特殊演算法與 API（LangGraph）」
- 高於一般單晶片或單次 AI 呼叫的複雜度
- Pompt Design 讓題目以及回饋可以更加隨機以及貼合主題
- **可視化、可擴充的系統設計**

2. TTS 語音輸出整合（✓）

使用技術

- Text-to-Speech API
- 音訊播放模組（如 gTTS + pygame）

說明

系統會根據不同狀態（出題、回饋、錯誤）即時產生語音，
搭配 PIC18F 的實體動作回饋，形成多感官互動。

加分理由

- 整合外部 API
- 提升人機互動體驗
- 非單純文字輸出
-

3. 整合性

本專題整合：

- PIC18F 微控制器

- UART 通訊
- Timer / Interrupt / PWM
- AI 語言模型
- LangGraph 多 Agent 架構
- 語音輸出
- 前端 UI 顯示

從硬體端結合軟體，加入網頁前後端
形成完整的**跨平台人機互動系統**。

4. 專題完整性

系統具備完整流程：

1. 使用者實體輸入
2. AI 邏輯處理
3. 狀態同步
4. 視覺與實體回饋
5. 循環互動

非單一功能展示，而是可長時間運作的完整系統，並且設計方式讓主題可變，型成多樣性。

5. 創新性

- 將 **LangGraph 狀態機概念引入嵌入式專題**
- AI 與 硬體明確分工、低耦合
- 實體硬體參與 AI 決策流程

並且設計方式讓整體，更多元化，可擴充
將概念延伸可以讓學習，互動化，更適合用在低年齡教育
具備創新互動模式與教學示範價值。

加分項目參考

加分項目	**本專題對應**
外部周邊 / API	LLM API、LangGraph、TTS
效率	計算分工、事件驅動
可擴充性	LangGraph 可以依需求快速擴充
整合性	軟硬體 + AI + UI
完整性	全流程互動
創新性	AI 狀態機 + 硬體回饋式學習

補充:

遇到的困難與問題:

在本系統開發初期，原定目標為同時控制三軸伺服馬達以達成多維度的運動控制，然而，受限於 PIC18F4520 微控制器之硬體架構，該晶片僅配置兩組硬體脈波寬度調變（CCP）模組，導致第三軸馬達必須改採軟體模擬 PWM 的方式進行驅動，此一架構上的折衷方案在隨後的整合測試中引發了嚴重的訊號干擾問題，成為本次開發最主要的技術瓶頸，首先，軟體模擬 PWM 必須仰賴 CPU

進行精確的迴圈計數與腳位翻轉來產生控制訊號，然而，本系統同時啟用了 UART 非同步傳輸功能以接收外部指令，這導致了「中斷服務程式（ISR）」與「軟體 PWM 時序」之間的資源衝突，當系統在產生軟體 PWM 的高電位訊號期間，若恰逢 UART 接收中斷觸發，CPU 必須暫停當前的計時工作轉而處理通訊資料，在 Baud rate 較低且系統時脈僅為 125 kHz 的環境下，中斷處理過程造成的延遲顯著拉長了 PWM 的脈波寬度，此非預期的訊號變形導致第三軸馬達將錯誤的脈寬解讀為極大的角度變化，進而產生劇烈的物理抖動與異常的高頻噪音，嚴重影響控制的精確度與馬達壽命，其次，由於系統運行於 125 kHz 的極低時脈頻率，指令週期長達 32 微秒，這使得 C 語言中涉及除法的角度換算運算變得極為耗時，在缺乏硬體浮點數運算單元的情況下，繁重的數學運算進一步佔用了 CPU 資源，導致軟體 PWM 的頻率難以穩定維持在標準的 50 Hz，面對上述硬體限制與軟體干擾的雙重挑戰，為確保系統的絕對穩定性與訊號純淨度，最終決策將架構調整為雙馬達控制，完全捨棄受 CPU 負載影響的軟體模擬方案，轉而全數採用硬體 CCP 模組直接驅動，從根本上解決了時序衝突所導致的失控現象。