國立臺中科技大學資訊工程系

實務專題報告書

Asar——輕鬆打造聊天機器人

指導教授：張家瑋　教授

洪啟舜　教授

學生：蔡詳羿

陶昱瑋

中華民國 111 年 11 月

摘要

本專題旨在開發一款名為 Asar 的輕量級聊天機器人設計平台，將聊天機器人與樹莓派結合，提供一條龍的聊天機器人服務部署流程。開發者僅需一片微型單板電腦，就能輕鬆創造專屬於自己的聊天機器人。

以往，開發者需要掌握多種程式語言及專業技術，才有能力建立聊天機器人服務。而 Asar 平台的出現大幅降低了此門檻。

Asar 的系統架構採用容器化設計，實現快速建構和部署。在自然語言處理方面，Asar 使用 Transformer、ALBERT 等近代深度學習技術，使聊天機器人具備自然語言理解能力。在開發者體驗方面，Asar 提供專屬的視覺化設計工具，讓開發者以流程圖的思維來設計聊天劇本；以拖拉方塊的方式來編寫程式，進而控制樹莓派的周邊設備。此外，Asar 提供了各大通訊平台的接口，簡化將聊天機器人整合至聊天室的步驟。在隱私方面，Asar 平台完全運行於樹莓派上，無須依賴外部服務，降低個人資料外流的風險。受益於樹莓派的自由度、擴充性等優勢，Asar 能夠應用於各種場景。

關鍵詞：Asar、聊天機器人、樹莓派、微型單板電腦。

目次

摘要.....i
目次.....ii
圖次.....iv
第 1 章 緒論.....1
1.1. 研究動機.....1
1.2. 研究目標.....2
第 2 章 現有相關研究概況及比較.....3
2.1. 聊天機器人的 NLU 流程.....3
2.2. 近代自然語言處理技術.....4
2.2.1. Transformer 模型.....4
2.2.2. BERT 與 ALBERT 模型.....5
2.3. 聊天機器人的設計模式.....6
2.4. 視覺化程式語言.....7
第 3 章 研究方法.....8
3.1. 系統架構.....8
3.2. 前端網頁服務.....10
3.2.1. 架構.....10
3.2.2. 功能.....10
3.2.2.1. 聊天機器人設計工具.....10
3.2.2.2. 視覺化劇本設計工具.....11
3.2.2.3. 視覺化程式設計工具.....11
3.3. 後端 API 服務.....12
3.3.1. 架構.....12
3.3.2. 功能.....13
3.3.2.1. 儲存訓練資料.....13
3.3.2.2. 轉換訓練資料格式.....13
3.3.2.3. 呼叫子服務訓練、執行對話模型.....13

3.4. 聊天機器人服務.....14
3.4.1. 架構.....14
3.4.2. 功能.....14
3.4.2.1. 自然語言理解（NLU）.....14
3.4.2.2. 訓練與執行對話模型.....15
3.4.2.3. 串接通訊平台.....15
3.5. 動作代理服務.....16
3.5.1. 架構.....16
3.5.2. 功能.....16
3.5.2.1. 執行非對話型任務.....16
第 4 章 研究成果.....17
4.1. 部署聊天機器人.....17
4.2. 設計聊天機器人.....18
4.2.1. 專案管理工具.....18
4.2.2. 訓練資料管理工具.....19
4.2.3. 視覺化劇本設計工具.....20
4.2.4. 視覺化程式設計工具.....21
4.2.5. 聊天室.....22
4.3. 轉換訓練資料格式.....22
4.4. 串接通訊平台.....24
4.5. 實際案例——居家管家.....25
第 5 章 結論與未來展望.....30
5.1. 結論.....30
5.2. 未來展望.....31
第 6 章 參考文獻.....32

圖次

圖 2-1 NLU 流程圖 3
圖 2-2 Transformer 模型架構 4
圖 2-3 BERT 的用例 5
圖 2-4 Google Dialogflow CX 6
圖 2-5 Amazon Lex 6
圖 2-6 Scratch 範例程式碼——判斷數字奇偶數 7
圖 2-7 MIT APP Inventor 範例程式碼——判斷數字奇偶數 7
圖 3-1 Asar 系統架構 8
圖 3-2 管理員與使用者的操作流程圖 9
圖 3-3 前端網頁服務架構 10
圖 3-4 Vue Flow 11
圖 3-5 Google Blockly 11
圖 3-6 後端 API 服務架構 12
圖 3-7 Route-Model 軟體架構模式 12
圖 3-8 後端 API 服務的功能 13
圖 3-9 聊天機器人服務架構 14
圖 3-10 NLU 流程 14
圖 3-11 訓練並執行對話模型 15
圖 3-12 接收並回覆使用者的聊天訊息 15
圖 3-13 動作代理服務架構 16
圖 3-14 動作代理服務執行非對話型任務 16
圖 4-1 Docker Compose 模板 17
圖 4-2 專案管理工具 18
圖 4-3 訓練參數調整介面 18
圖 4-4 各類訓練資料的管理介面 19
圖 4-5 視覺化劇本設計工具介面及節點種類 20
圖 4-6 開關風扇的劇本 20

圖 4-7 擴充方塊 21
圖 4-8 控制繼電器與串接 Covid-19 資訊 API 的程式碼 21
圖 4-9 聊天室 22
圖 4-10 訓練資料轉換前（左）、後（右）對照圖 23
圖 4-11 程式碼轉換前（左）、後（右）對照圖 23
圖 4-12 credentials.yml 24
圖 4-13 串接至各個通訊平台的聊天室 24
圖 4-14 居家管家案例的硬體設備 25
圖 4-15 居家管家案例的界面導覽 25
圖 4-16 讀書情境 26
圖 4-17 睡眠情境 27
圖 4-18 外出情境 28
圖 4-19 控制電器設備 29
圖 4-20 Covid-19 的即時資訊 29

第1章 緒論

1.1. 研究動機

近年來，聊天機器人（Chatbot）迅速崛起。在日常生活中，人們使用虛擬助理（Virtual assistant）來處理各種瑣事，例如查詢天氣、安排行程，甚至能在外出時控制家中的電器，虛擬助理帶來的便利大大提高了人們的生活品質。在商業上，企業引入虛擬客服來節省成本，同時也改善了營運效率與用戶體驗。根據 IT 研究權威公司 Gartner 的預測[1]，在未來五年內，企業使用聊天機器人作為客服的方式將會成為主流。目前，聊天機器人正逐漸被應用在各個領域，它將成為未來生活的趨勢。

隨著人工智慧的發展，在自然語言處理（NLP）的領域中，已經出現如 Transformer、ALBERT 輕量級的深度學習模型，如今，執行 NLP 任務不再需要效能強大的電腦，僅需一片微型單板電腦就能運行整個聊天機器人服務。

樹梅派（Raspberry Pi）是當今熱門的微型單板電腦，具有價格低廉、可擴充性高的優勢，適合運行聊天機器人服務。然而，在樹梅派上部署一個聊天機器人服務並不容易，流程繁瑣且耗時，目前尚無完整的解決方案，這對於開發者而言無疑是一大阻礙。因此，本專題將提出一套解決方案來簡化部屬流程，並降低技術門檻，讓開發者能輕鬆地在樹莓派上部署聊天機器人服務。

1.2. 研究目標

本專題的目的是開發一款名為 Asar，運作於樹莓派的輕量級聊天機器人設計平台，提供能輕鬆部署聊天機器人服務的解決方案，Asar 有以下目標：

(1) 具備 NLU 能力的聊天機器人

使用者不必提供明確的指令，聊天機器人就能理解出正確的意圖。例如使用者說「我很熱」時，聊天機器人會詢問使用者「要開風扇嗎？」

(2) 簡化部署流程：

Asar 平台將以容器的方式運行，並整合設計聊天劇本、訓練對話模型、控制樹莓派周邊設備、串接外部 API、串接通訊平台等重點功能，讓開發者能快速部署並將服務上線。

(3) 人性化的設計工具：

Asar 平台將提供視覺化的設計工具，取代傳統文字編輯器，協助開發者更有效率地管理訓練資料及對話模型。

(4) 控制樹梅派周邊設備及串接外部 API：

Asar 平台將提供專屬的視覺化程式設計工具，開發者僅需具備基本程式設計能力，就能輕鬆控制樹梅派周邊設備及串接外部 API。

(5) 注重個人隱私，降低敏感資訊外流的風險：

Asar 平台將完全運行於樹莓派上，聊天訊息的推論由本地端處理，無須依賴外部服務，從而避免敏感資訊外流。

第2章 現有相關研究概況及比較

2.1. 聊天機器人的 NLU 流程

根據 Rasa 論文[2]、Rasa Blog 文章[3]的研究，NLU 流程（圖 2-1）依序為斷詞、特徵化、實體提取、意圖識別、回覆決策，各個步驟的功能如下：

(1) 斷詞（tokenize）：將輸入的句子分割成單詞

(2) 特徵化（featurize）：將單詞轉換成向量或數值，以便電腦進行處理和分析。

(3) 實體提取（entity extraction）：從句子中提取重要的訊息，例如：人名、地點。

(4) 意圖識別（intent detection）：判斷使用者在對話中的意圖或目的。

(5) 回覆決策（response selection）：根據使用者的意圖，做出相對應的回覆。

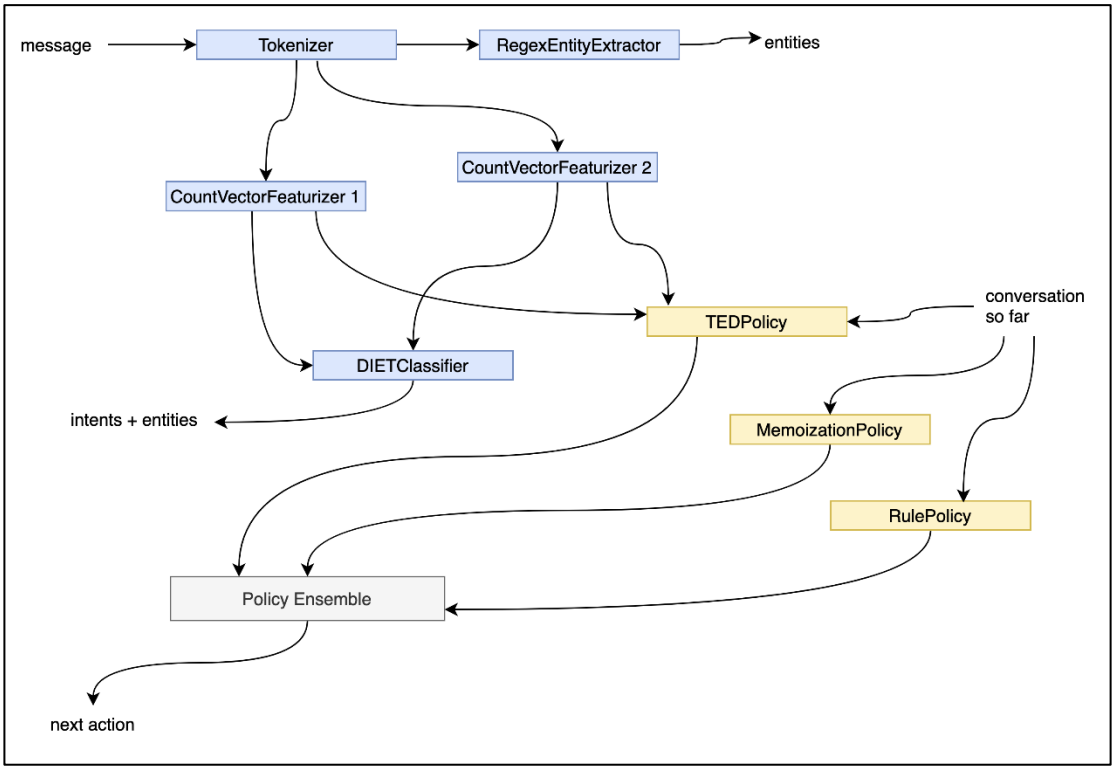

圖 2-1 NLU 流程圖

圖片來源：Rasa Blog 文章[3]

2.2. 近代自然語言處理技術

2.2.1. Transformer 模型

Transformer[4]是一種編碼器-解碼器（encoder-decoder）架構的深度學習模型，它使用自注意力機制（Self-attention），解決了以往 LSTM、RNN 模型中梯度消失、無法並列處理文本的問題。Transformer 模型能夠執行各種自然語言處理任務，例如機器翻譯（Machine Translation）、命名實體辨識（Named Entity Recognition），目前 Transformer 模型是解決自然語言處理問題的首選模型之一。

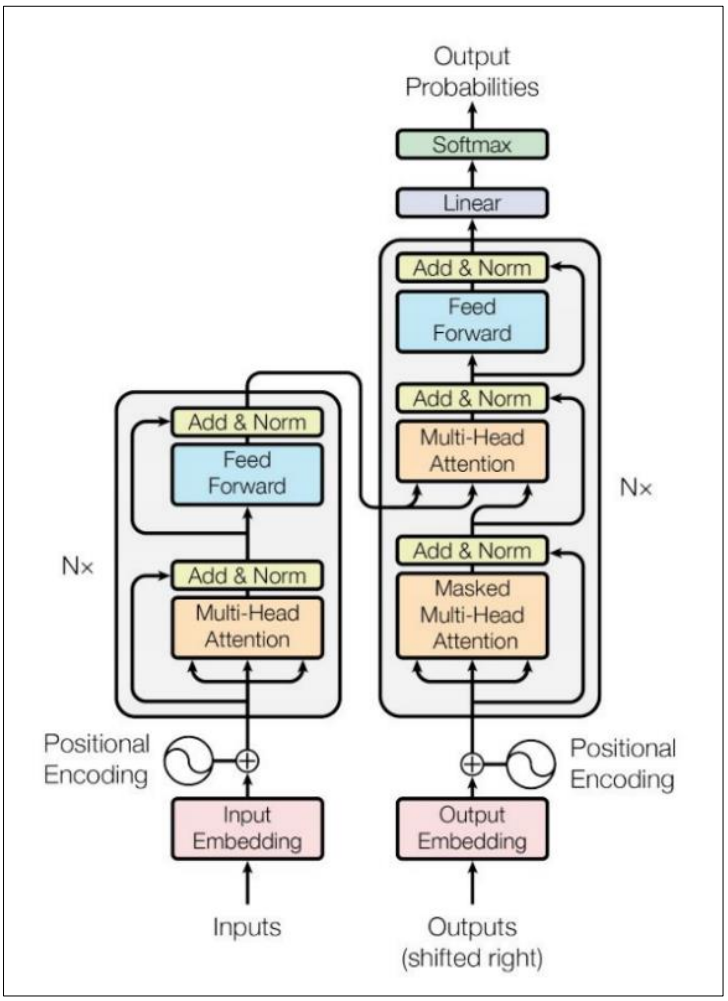

圖 2-2 Transformer 模型架構

圖片來源：Attention is all you need[4]

2.2.2. BERT 與 ALBERT 模型

BERT[5]全名 Bidirectional Encoder Representations from Transformers，採用 Transformer 模型中的 Encoder 技術。BERT 預訓練模型是 Google 以無監督學習，並使用大量純文字語料庫訓練成的語言模型。在自然語言處理的領域中，常被用來當作遷移學習（Transfer learning）的模型，在 BERT 論文中提到了四種用例（圖 2-3），包括成對句分類、單一句分類、問答任務以及標記任務。

ALBERT（A Lite BERT）[6]是一個基於 BERT 的輕量化模型，利用共享參數、句序預測，改善其模型大小與訓練成本，適合運作在資源有限的系統上。

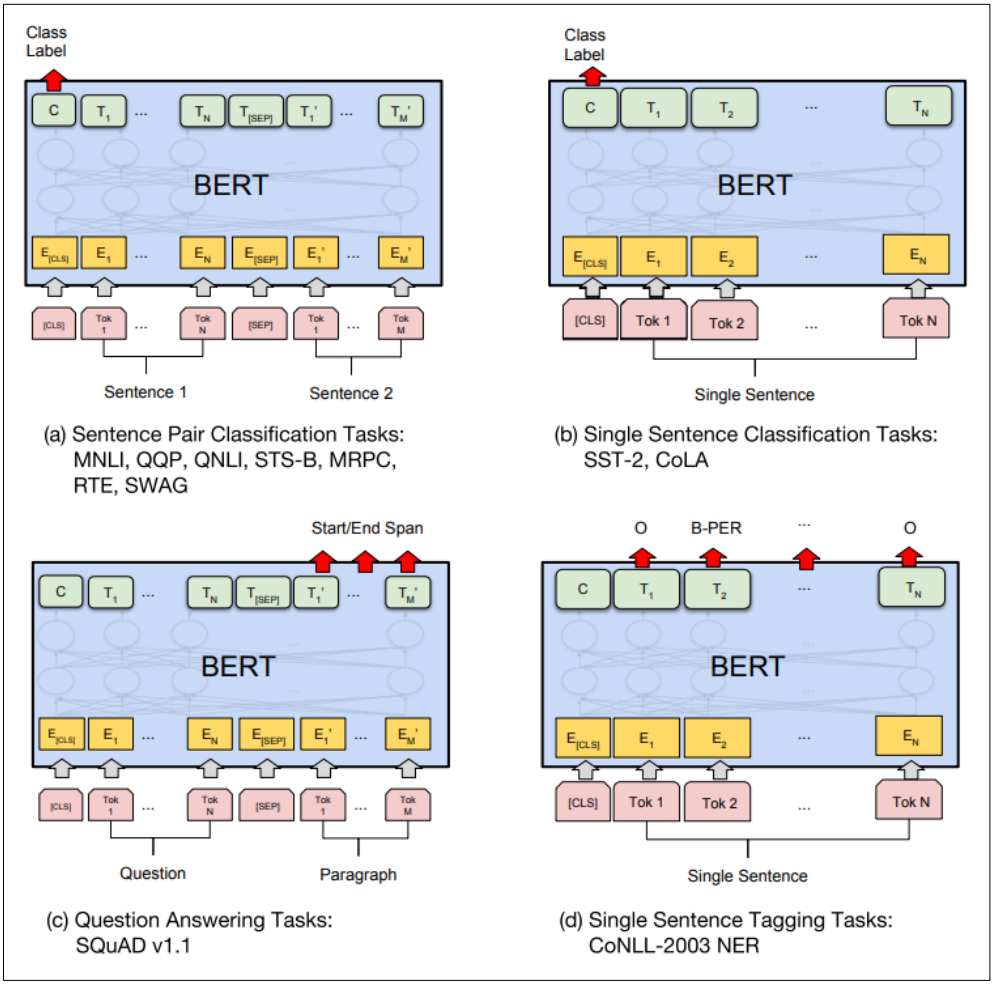

圖 2-3 BERT 的用例

圖片來源：Bert[5]

2.3. 聊天機器人的設計模式

現行的商業聊天機器人開發平台，如 Google Dialogflow CX（圖 2-4）、Amazon Lex（圖 2-5）。皆採用以「邊與節點」為概念的視覺化設計模式來建構聊天機器人，概念包括流程圖、有向圖等。

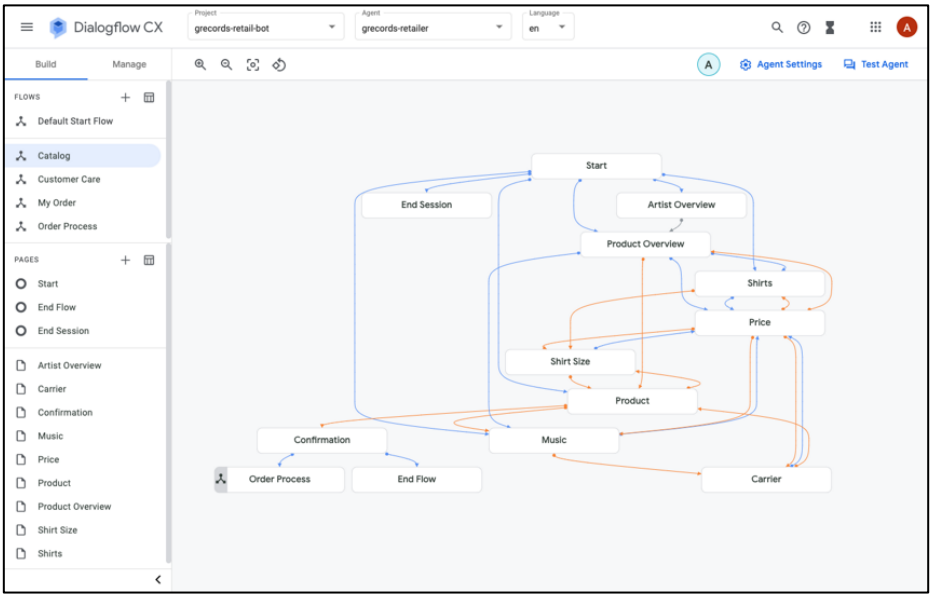

圖 2-4 Google Dialogflow CX

圖片來源：Dialogflow CX documentation

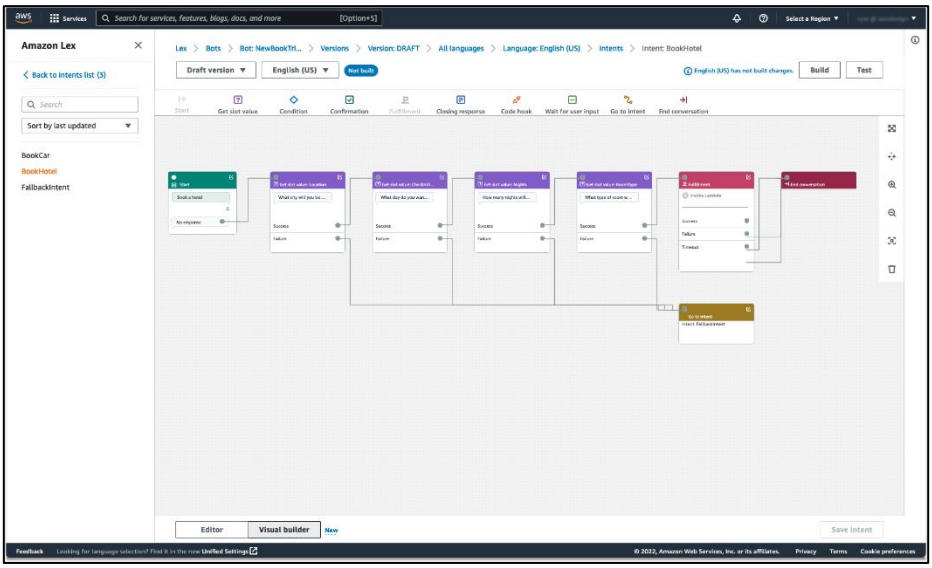

圖 2-5 Amazon Lex

圖片來源：Amazon Lex Documentation

2.4. 視覺化程式語言

視覺化程式語言（Visual programming language）是一種以「方塊」為概念的程式語言，讓使用者透過圖形化元素進行程式設計，相較於文字式程式語言更為直覺、簡單，因此有許多面向初學者的程式設計工具採用視覺化程式語言，例如 Scratch（圖 2-6）、MIT App Inventor（圖 2-7）。

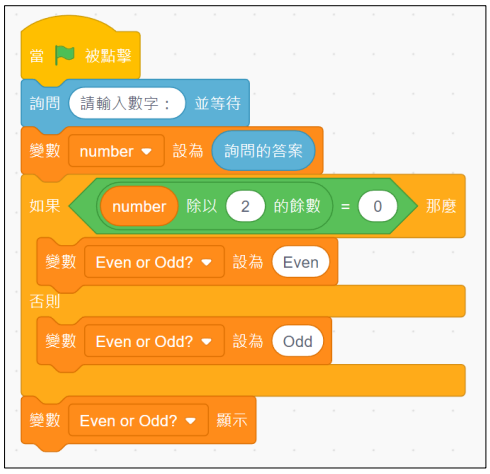

圖 2-6 Scratch 範例程式碼——判斷數字奇偶數

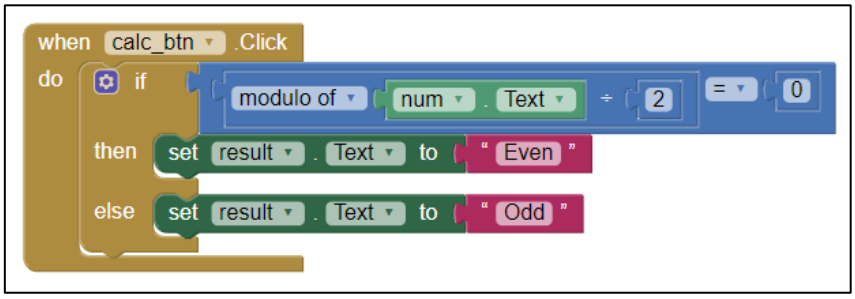

圖 2-7 MIT APP Inventor 範例程式碼——判斷數字奇偶數

第3章 研究方法

3.1. 系統架構

Asar 平台採用容器化的設計，依據功能分成四個子服務，包括前端網頁服務、後端 API 服務、聊天機器人服務、動作代理服務，系統架構如圖 3-1。

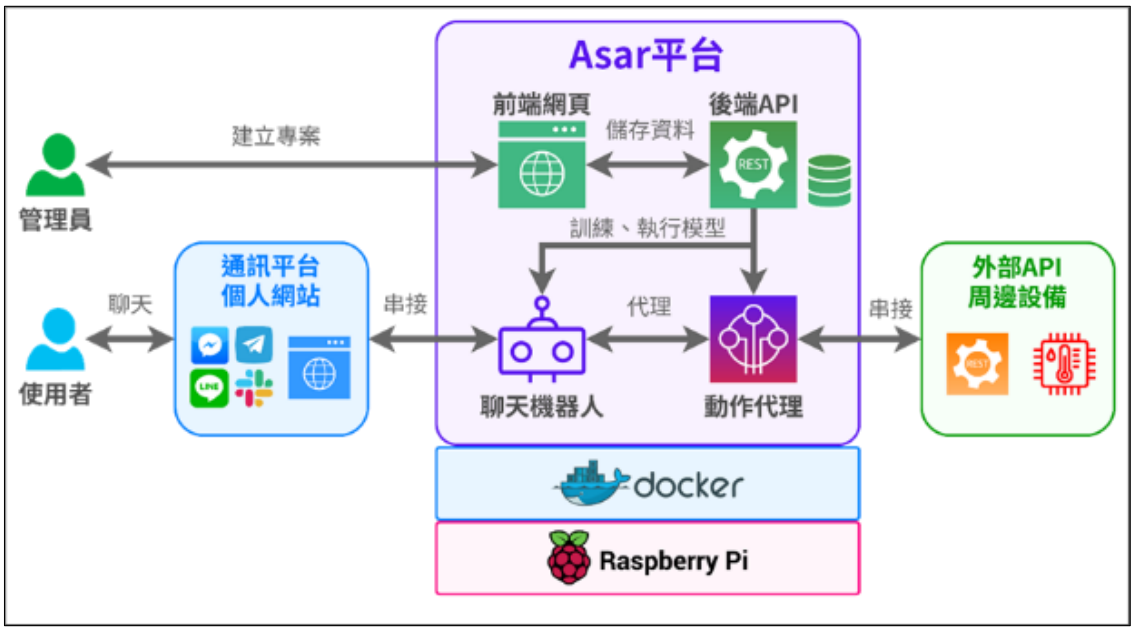

圖 3-1 Asar 系統架構

管理員（開發者）的部屬流程如圖 3-2 左圖。首先，管理員使用 Docker 安裝 Asar 平台，隨後進入前端網頁服務，建立聊天機器人專案及撰寫訓練資料。與此同時，後端 API 服務會接收、驗證並儲存資料。在訓練對話模型時，後端 API 服務會將訓練資料傳送至聊天機器人服務及動作代理服務執行訓練。最後，管理員將聊天機器人串接至通訊平台，即完成部署。

使用者的互動流程如圖 3-2 右圖。使用者經由通訊平台與聊天機器人對話，訊息會傳送至聊天機器人服務執行 NLU 流程，在對話過程中，如果使用者試圖執行非對話型任務，例如呼叫外部 API、控制周邊設備。將會觸發動作代理服務代為執行。最後，若使用者停止對話，則互動結束。

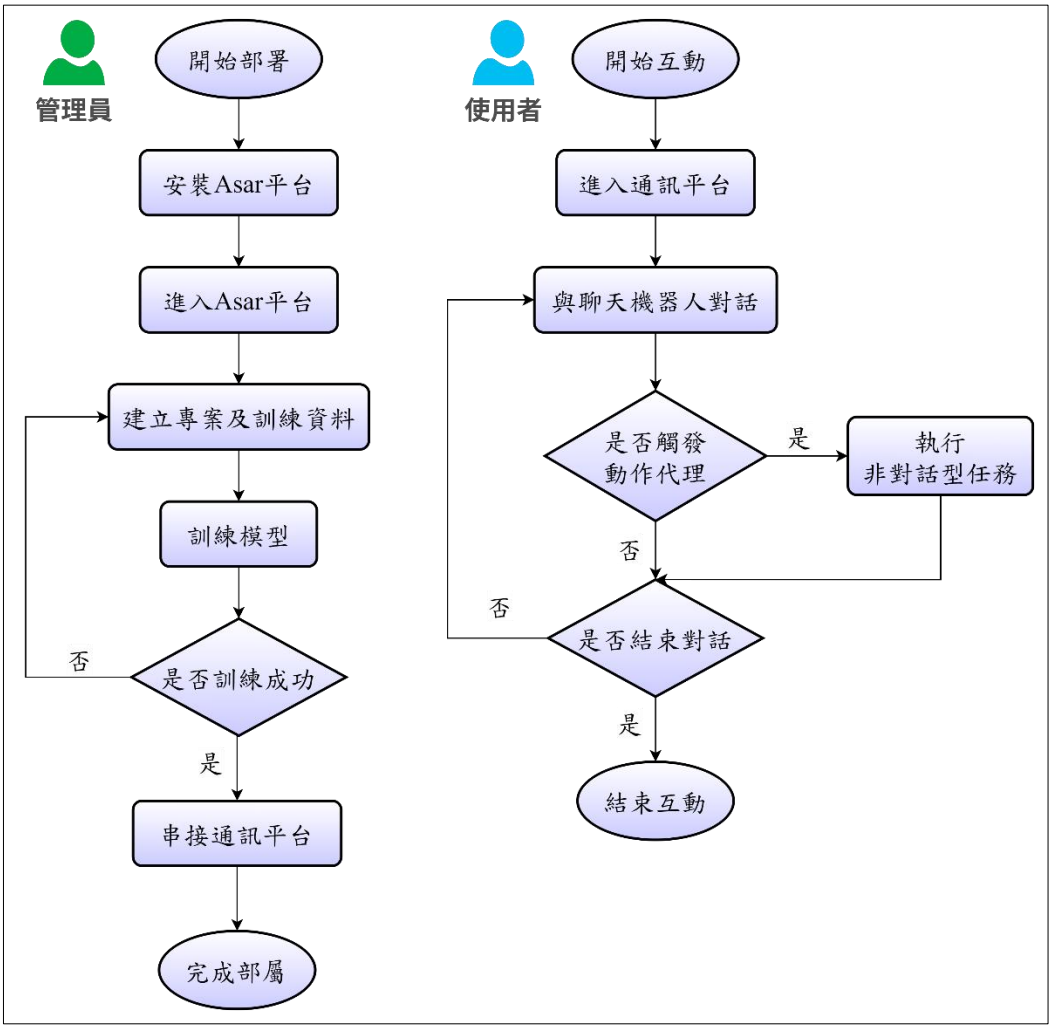

圖 3-2 管理員與使用者的操作流程圖

3.2. 前端網頁服務

3.2.1. 架構

本服務使用 Nginx 作為網頁伺服器，網頁使用 Vue 框架設計，並結合 Prime Vue、Vue flow、Google Blockly 來開發設計工具，架構圖如圖 3-3。

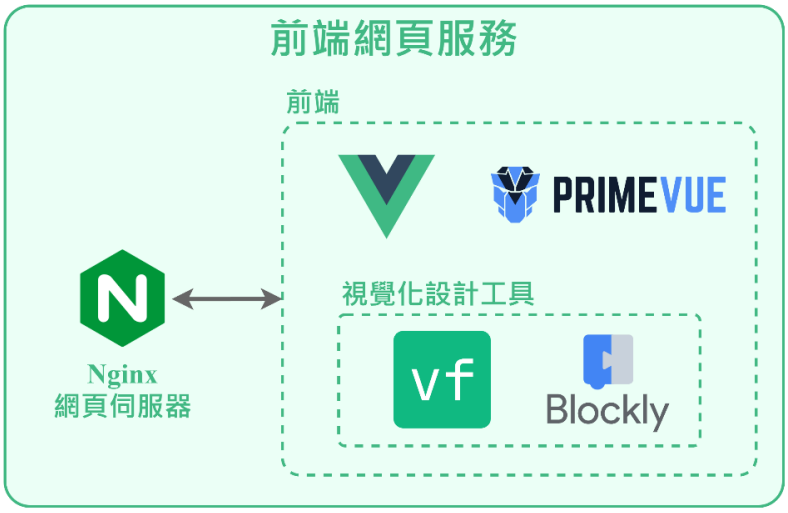

圖 3-3 前端網頁服務架構

3.2.2. 功能

本服務提供了多種設計工具，管理員能透過網頁瀏覽器來使用。設計工具包括：

3.2.2.1. 聊天機器人設計工具

此工具提供專案管理、訓練資料管理功能，內建一個測試用的聊天室，採用組件化設計來使頁面的版型整齊一致，並透過響應式設計來實現動態網頁效果，進而優化使用者體驗。此工具使用 Vue 框架、Prime Vue 組件庫開發。

3.2.2.2. 視覺化劇本設計工具

此工具是一個以流程圖為概念的劇本設計工具，讓管理員能以直觀的對話流程思維來設計劇本。此工具使用 Vue flow 函式庫開發（圖 3-4）。

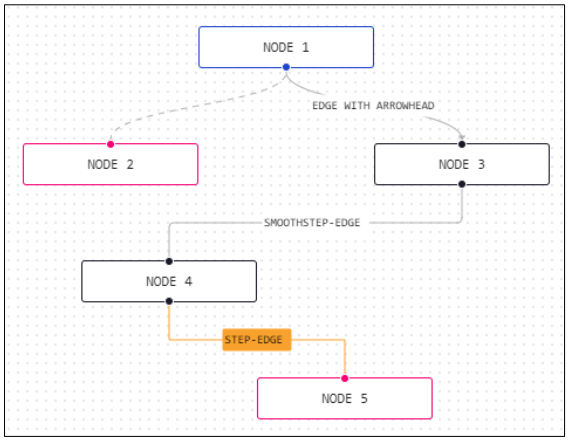

圖 3-4 Vue Flow

3.2.2.3. 視覺化程式設計工具

此工具是一個以方塊為概念的程式設計工具，讓管理員能以簡單的方式來控制樹莓派的周邊設備、串接外部 API。此工具使用 Google Blockly 函式庫開發（圖 3-5）。

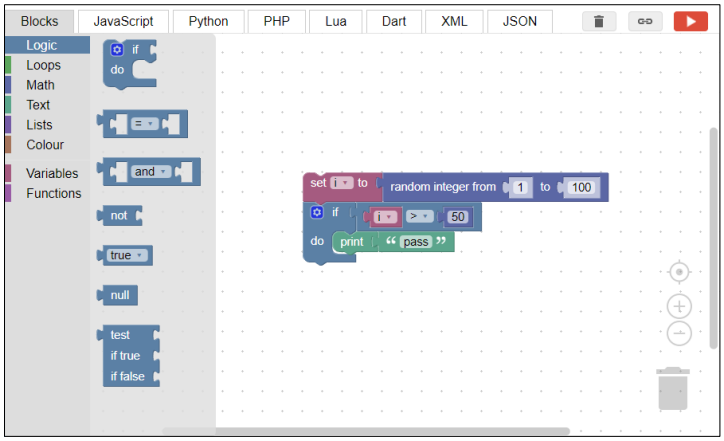

圖 3-5 Google Blockly

3.3. 後端 API 服務

3.3.1. 架構

本服務使用 gunicorn 作為伺服器，SQLite 作為資料庫管理系統。API 服務使用 Flask 框架設計，並採用 Jinja 模板引擎來設計程式碼轉換工具，架構圖如圖 3-6。

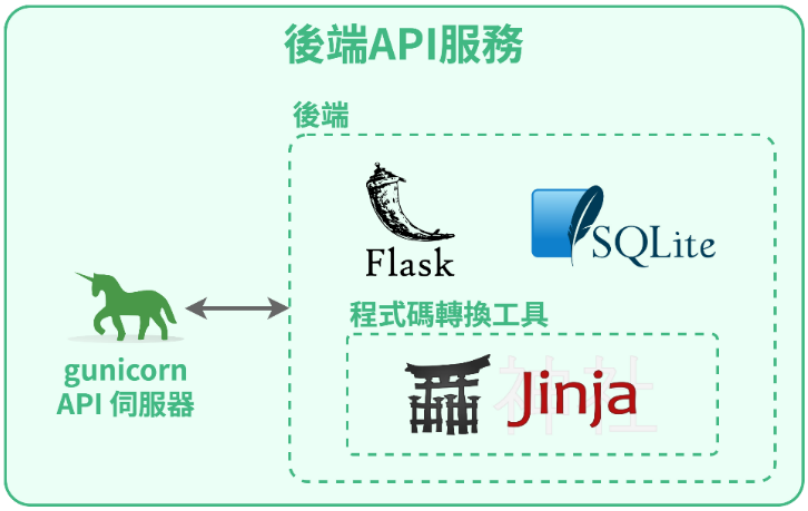

圖 3-6 後端 API 服務架構

API 服務使用 Route-Model 設計模式（圖 3-7），將訊息處理與資料掌管的程式碼分離，進而簡化程式碼的複雜度、提高可重用性，而程式碼的設計方式採用基於物件導向的 Class-based Views 模式。

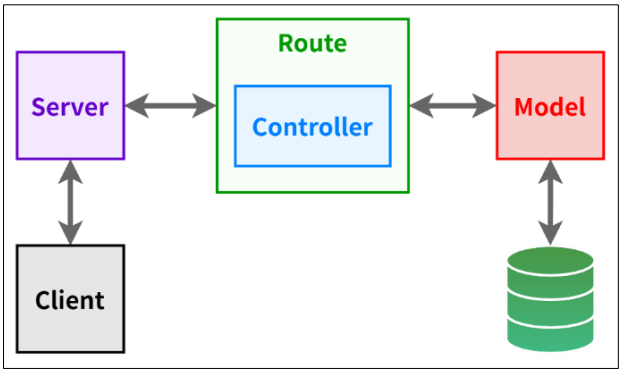

圖 3-7 Route-Model 軟體架構模式

3.3.2. 功能

本服務是 Asar 平台中所有子服務之間的溝通橋樑（圖 3-8），讓各個子服務可以相互交流和協調工作，本服務提供的功能如下：

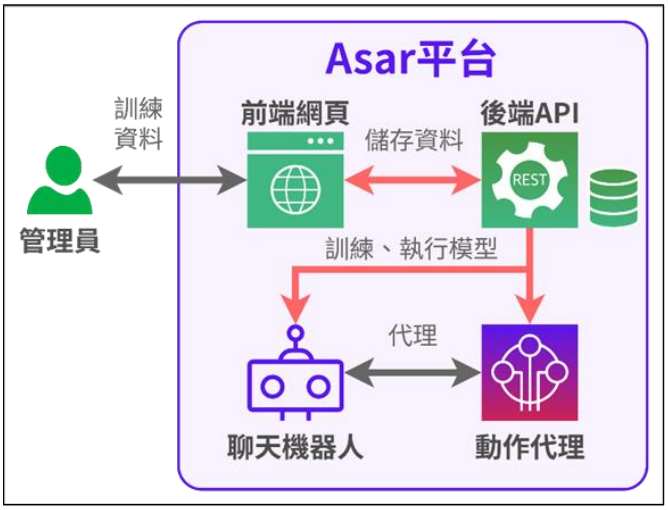

圖 3-8 後端 API 服務的功能

3.3.2.1. 儲存訓練資料

接收管理員從前端網頁提供的訓練資料。儲存至後端檔案系統和資料庫。

3.3.2.2. 轉換訓練資料格式

訓練模型之前，必須將原始的訓練資料轉換成指定的格式，此工具使用 Python、Jinja 模板引擎開發。

3.3.2.3. 呼叫子服務訓練、執行對話模型

將訓練資料傳送至聊天機器人服務和動作代理服務進行訓練，並執行對話模型。

3.4. 聊天機器人服務

3.4.1. 架構

本服務基於 Rasa 機器學習框架開發，並整合 CKIP Transformers 的 ALBERT 語言模型、斷詞器與實體提取器等自然語言處理工具，來支援繁體中文。架構圖如圖 3-9。

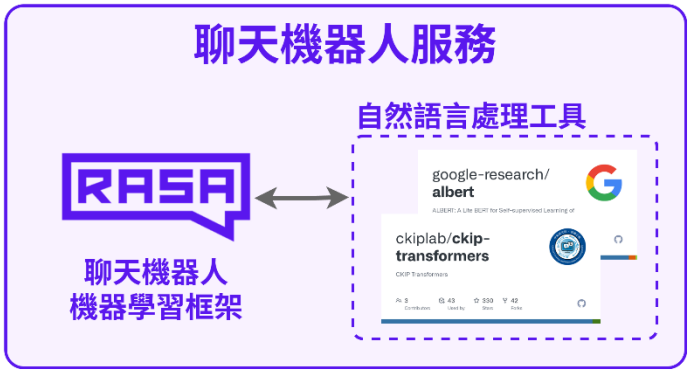

圖 3-9 聊天機器人服務架構

3.4.2. 功能

3.4.2.1. 自然語言理解（NLU）

本服務以管線化（pipeline）的方式管理 NLU 流程（圖 3-10），依序為斷詞、特徵化、實體提取、意圖識別、回覆決策。

圖 3-10 NLU 流程

3.4.2.2. 訓練與執行對話模型

本服務接收由管理員提供的訓練資料，負責訓練、執行對話模型（圖 3-11）。

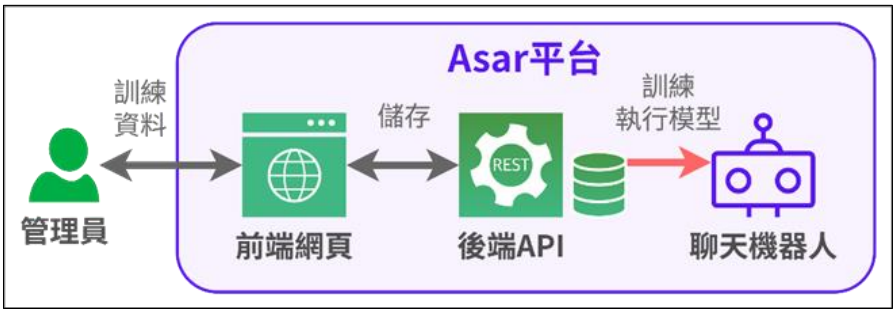

圖 3-11 訓練並執行對話模型

3.4.2.3. 串接通訊平台

本服務提供 Line、Facebook Messenger、Telegram 通訊平台接口，以及 Web API，能夠接收並回覆使用者的聊天訊息（圖 3-12）。

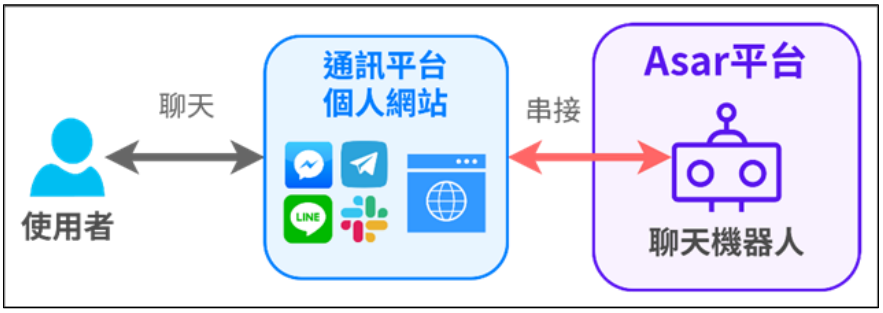

圖 3-12 接收並回覆使用者的聊天訊息

3.5. 動作代理服務

3.5.1. 架構

本服務基於 Rasa SDK 開發，Rasa SDK 屬於 Rasa 機器學習框架生態系的一部份，用於設計非對話型任務。本服務透過 RPi.GPIO Python 套件、映射主機設備功能，讓容器內的程式能夠存取樹莓派的周邊設備。架構圖如圖 3-13。

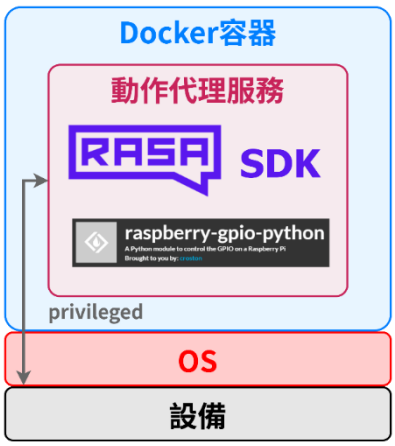

圖 3-13 動作代理服務架構

3.5.2. 功能

3.5.2.1. 執行非對話型任務

非對話型任務包含呼叫外部 API、控制樹莓派周邊設備，如果使用者在聊天過程中試圖執行非對話型任務，聊天機器人服務會請求動作代理服務來代理執行。

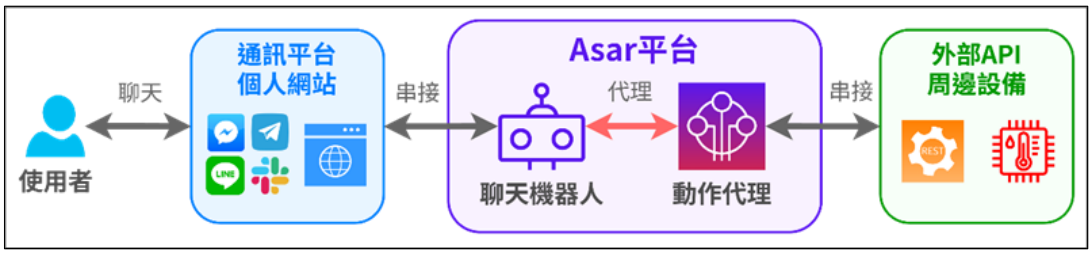

圖 3-14 動作代理服務執行非對話型任務

第4章 研究成果

4.1. 部署聊天機器人

Asar 平台採用 Docker 容器化設計，可透過 Docker Compose（圖 4-1）一鍵啟動所有 Asar 子服務，實現快速部署。

```yaml
version: "3.8"

networks: # 創建Asar專用的Docker Network
  default:
    name: asar-prod

volumes:
  rasa:    # 存放聊天機器人服務的資料
  actions: # 存放動作代理服務的資料
  data:    # 存放後端API服務的資料

services:
  rasa: # 聊天機器人服務
    image: devilhyt/rasa:custom
    volumes:
      - rasa:/app
      - data:/data
    ports:
      - 5005:5005

  action: # 動作代理服務
    image: devilhyt/rasa-sdk:custom
    volumes:
      - actions:/app/actions
    ports:
      - 5055:5055
    privileged: true

  api: # 後端API服務
    image: devilhyt/asar-api:latest
    volumes:
      - actions:/actions
      - data:/data
    environment:
      SECRET_KEY: # Flask密鑰
      # 密鑰可用此指令生成
      # python -c 'import secrets; print(secrets.token_hex())'
      RASA_API_HOST: rasa # 聊天機器人服務的別名
      ASAR_API_HOST: api # 後端API服務的別名
    ports:
      - 5500:5500

  web: # 前端網頁服務
    image: devilhyt/asar-web:latest
    environment:
      RASA_API_HOST: rasa # 聊天機器人服務的別名
      ASAR_API_HOST: api # 後端API服務的別名
    ports:
      - 80:80
    depends_on:
      - api
```

圖 4-1 Docker Compose 模板

4.2. 設計聊天機器人

4.2.1. 專案管理工具

管理者可以創建多個聊天機器人專案，每個專案代表一種聊天機器人服務類型，左側選單包含訓練、載入和設定選項，管理者能選擇指定的專案來執行（圖 4-2），並且能自由調整對話模型的訓練參數（圖 4-3）。

圖 4-2 專案管理工具

圖 4-3 訓練參數調整介面

4.2.2. 訓練資料管理工具

訓練資料管理工具提供人性化的介面，協助管理者建立訓練資料，圖 4-4 為各類訓練資料的管理介面，由上至下依序為意圖、回覆、實體、槽位，左側為管理選單，右側為編輯器。

圖 4-4 各類訓練資料的管理介面

4.2.3. 視覺化劇本設計工具

視覺化劇本設計工具中有八種節點（圖 4-5），包含開始、結束、意圖、回覆、動作節點，以及具有邏輯判斷功能的「進行中的迴圈」、「槽位被設置」、「表單」節點。透過這些節點的組合與連接，管理者能創造出各種不同的劇本。

圖 4-5 視覺化劇本設計工具介面及節點種類

以「開關風扇」劇本為例（圖 4-6），開始節點的左分支為想開風扇，右分支反之，而左分支又有兩個子分支，當聊天機器人認為使用者想開風扇時，它會再次詢問是否要執行，左子分支為接受、右子分支為拒絕，而右分支同樣有兩個子分支，此劇本共有四個分支。

圖 4-6 開關風扇的劇本

4.2.4. 視覺化程式設計工具

在視覺化程式設計工具中，除了提供 Blockly 內建的基本方塊，還擴充了針對 Rasa 和樹莓派的專屬方塊（圖 4-7），開發者能使用這些方塊來存取聊天內容、控制樹莓派的周邊設備、串接外部 API。

對於一些複雜的程式操作或無法確定的資料格式，開發者可以使用 Advanced Code area 方塊，以撰寫文字程式碼的方式來實現功能，圖 4-8 左側為「控制繼電器短暫閉路」的程式碼；右側為串接「Covid-19 資訊 API」的程式碼。

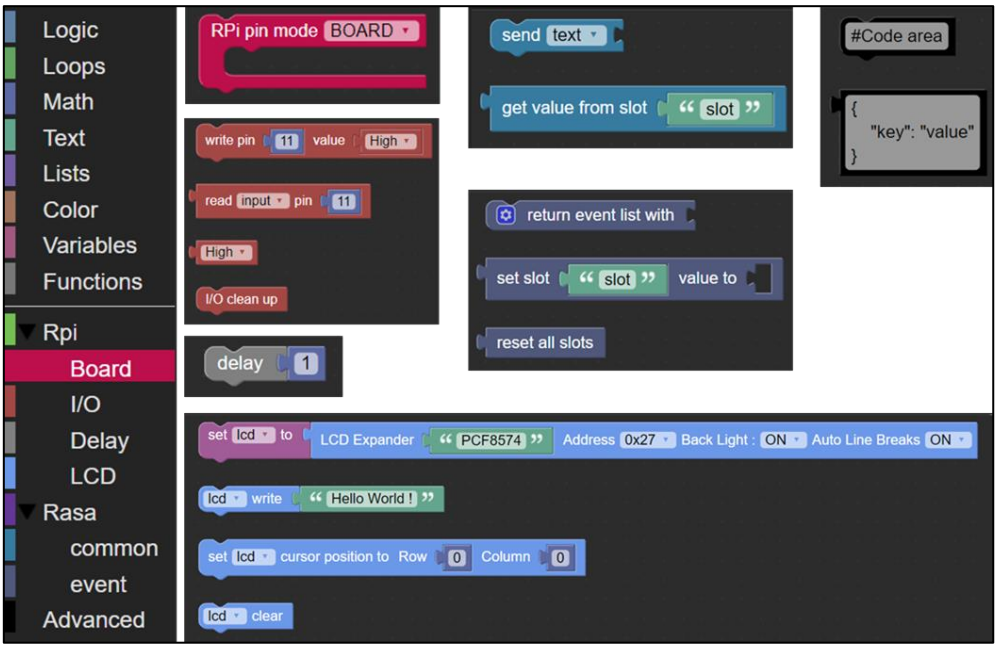

圖 4-7 擴充方塊

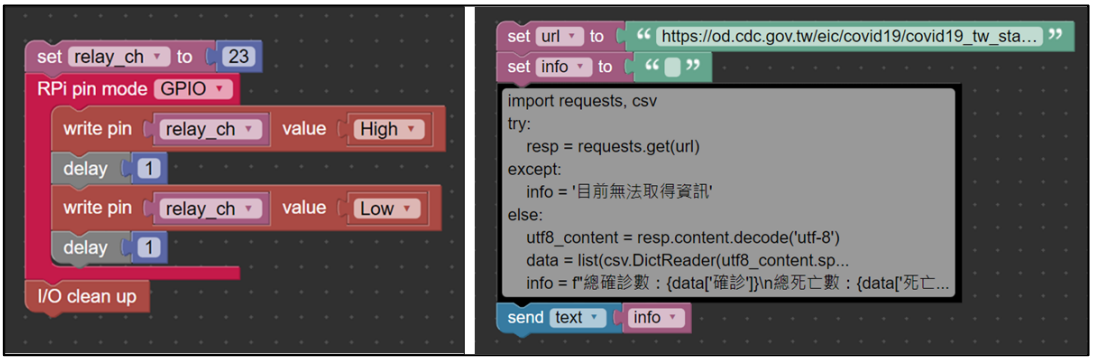

圖 4-8 控制繼電器與串接 Covid-19 資訊 API 的程式碼

4.2.5. 聊天室

管理者可以透過內建的聊天室（圖 4-9）來跟聊天機器人進行對話，檢查其回覆內容的準確性，並在必要時進行調整，以確保聊天機器人的行為符合預期。

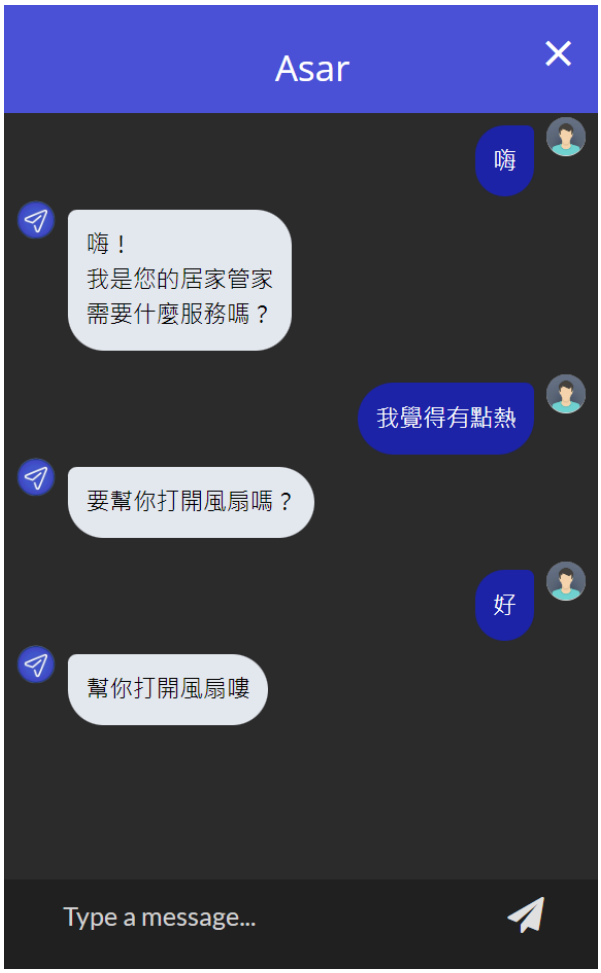

圖 4-9 聊天室

4.3. 轉換訓練資料格式

在訓練模型之前，後端會先將訓練資料轉換成 Rasa 支援的資料格式，再傳送給聊天機器人服務進行訓練。以 4.2.3 節的「開關風扇」劇本為例，圖 4-10 為資料轉換的前、後對照圖，轉換的方式是使用 DFS 演算法來分析流程圖結構，生成相應的 YAML 格式訓練資料。

圖 4-10 訓練資料轉換前（左）、後（右）對照圖

同樣地，後端會先將視覺化程式碼轉換成 Rasa SDK 支援的程式語言，再傳送給動作代理服務執行。以 4.2.4 節的「控制繼電器短暫閉路」的程式碼為例，圖 4-11 為程式碼轉換的前、後對照圖，轉換的方式是使用 Jinja 模板引擎，將 Google Blockly 生成的 Python 程式碼包裝成 Rasa SDK 的 Action Class 格式。

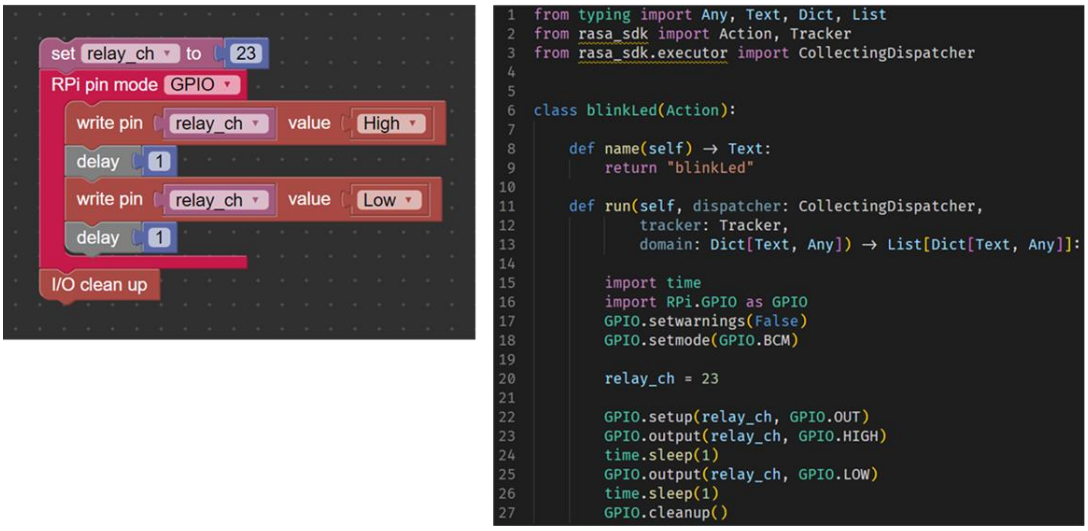

圖 4-11 程式碼轉換前（左）、後（右）對照圖

4.4. 串接通訊平台

Asar 提供 Line、Facebook、Telegram 通訊平台接口，將通訊平台的驗證資訊填入聊天機器人服務中的/app/credentials.yml 檔案（圖 4-12），重啟服務後即完成串接。

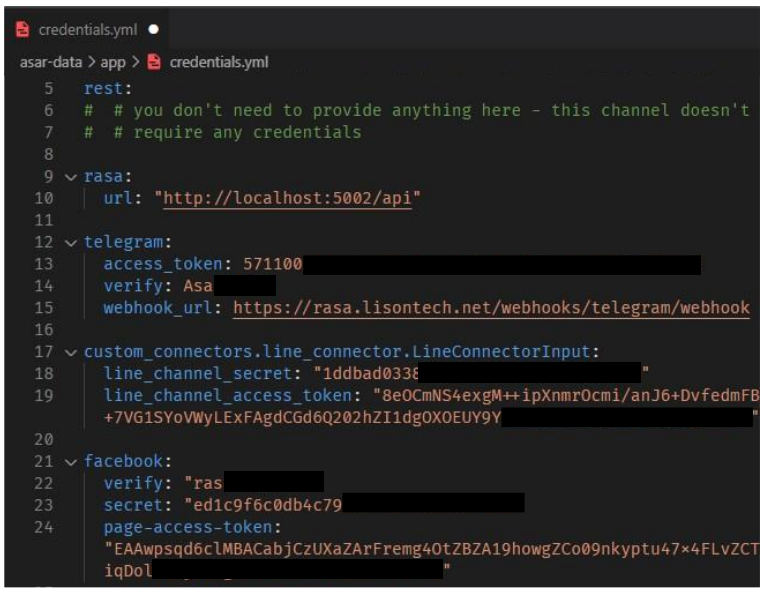

圖 4-12 credentials.yml

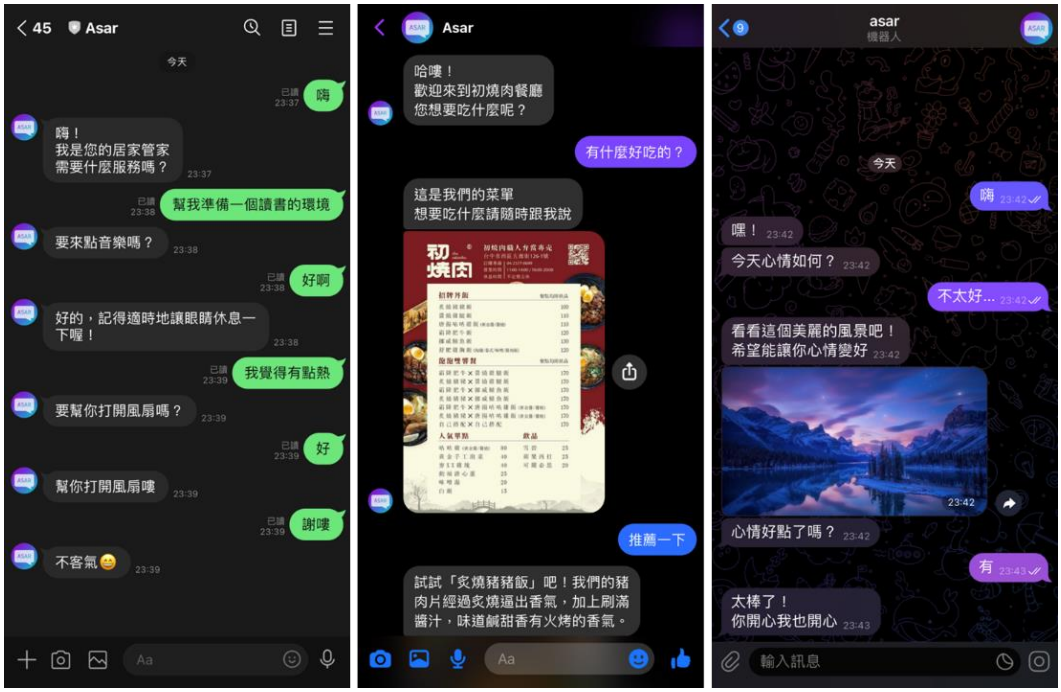

圖 4-13 串接至各個通訊平台的聊天室

4.5. 實際案例——居家管家

在居家管家案例中，提供了三種居家情境，包括讀書、外出、睡眠情境，並使用小型設備及感測器來模擬家中的電器設備，包括數位留言板（LCD 顯示器）、電風扇（小風扇及繼電器）、電燈（全彩 LED 燈）、溫度計（DHT11）。

圖 4-14 居家管家案例的硬體設備

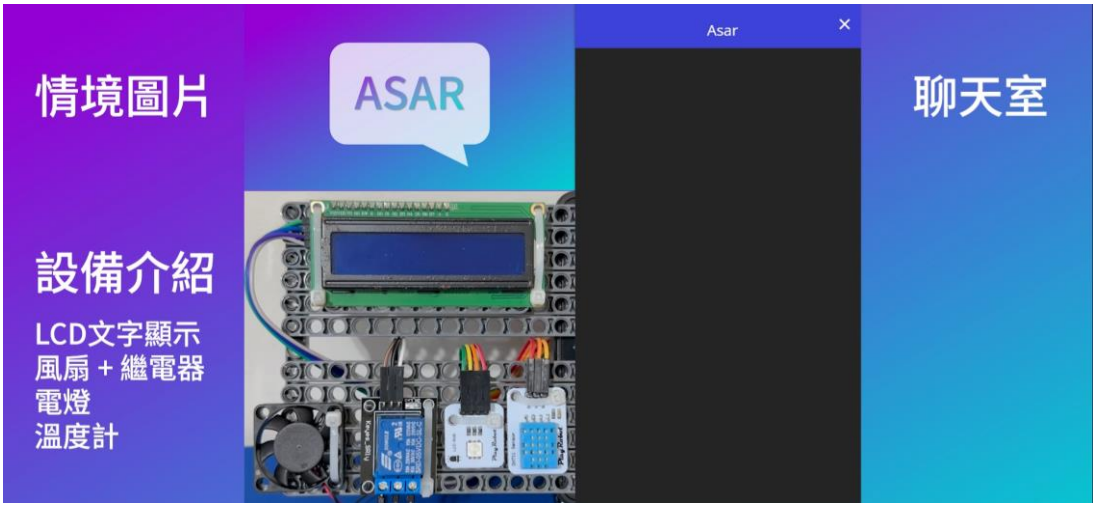

圖 4-15 居家管家案例的界面導覽

當使用者向管家問候，要求設置讀書環境時，管家會給予正確的回應，主動詢問使用者是否需要播放音樂來提高讀書環境的品質，並提醒使用者要適時地休息（圖 4-16）。

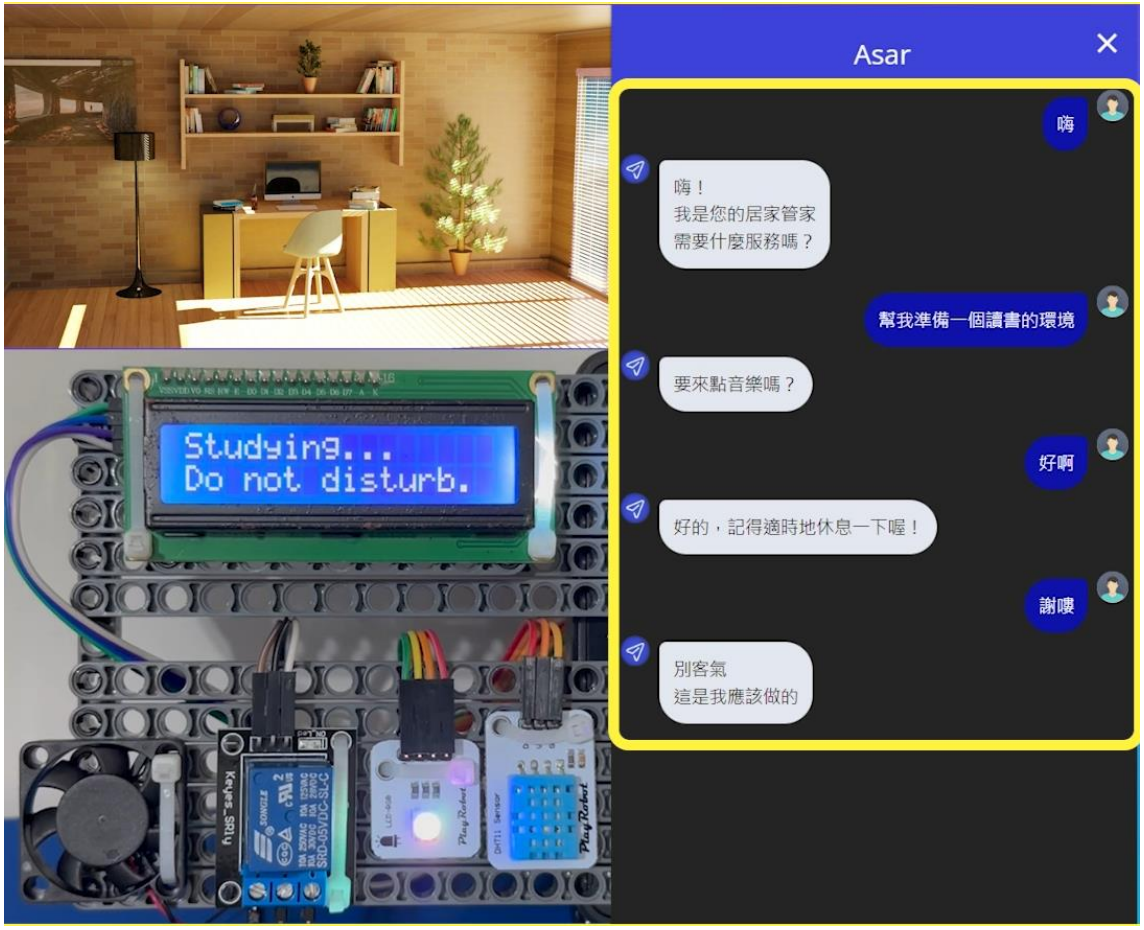

圖 4-16 讀書情境

使用者要睡覺時，管家會開啟紅色夜燈，並偵測室溫，管家會根據室溫主動詢問使用者是否要開、關電風扇。

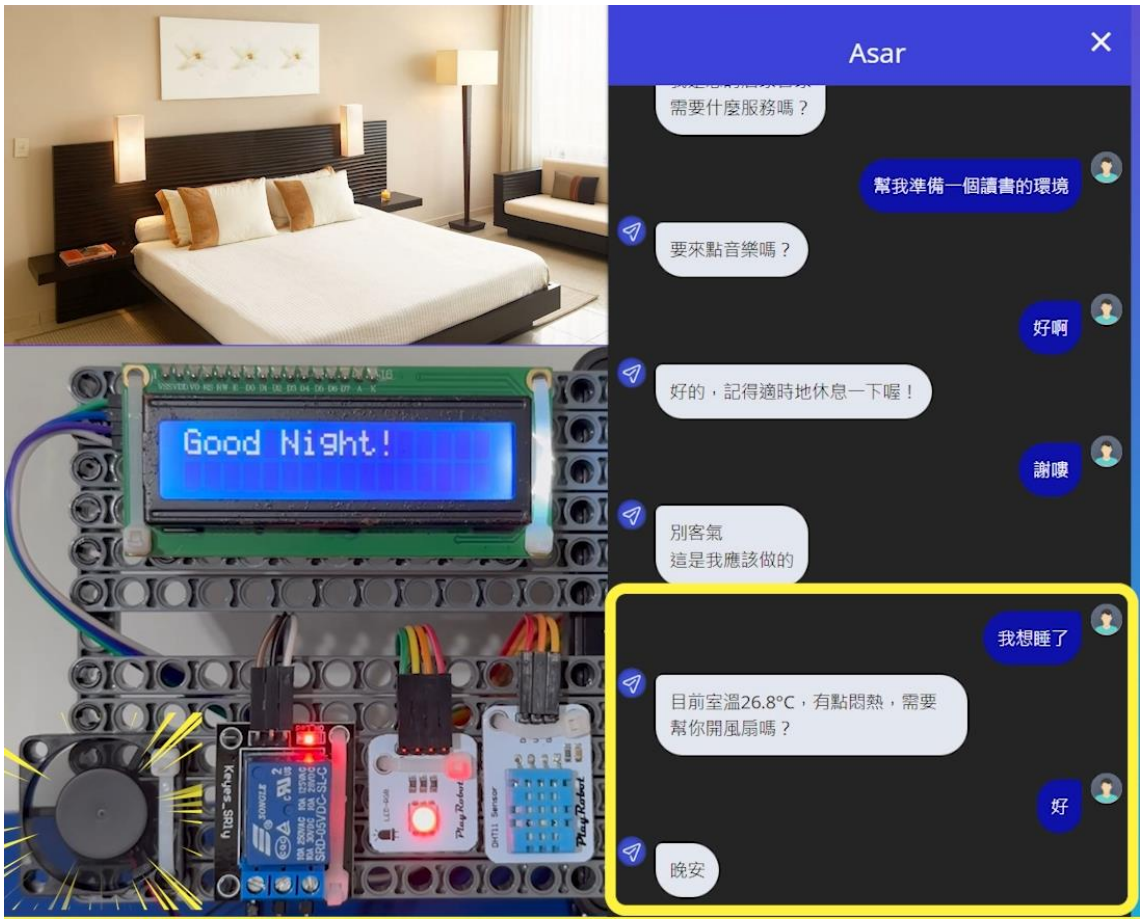

圖 4-17 睡眠情境

使用者外出時，管家會將家中的電器設備關閉，並且更改數位留言板的文字訊息，提醒訪客目前主人不在家（圖 4-18）。

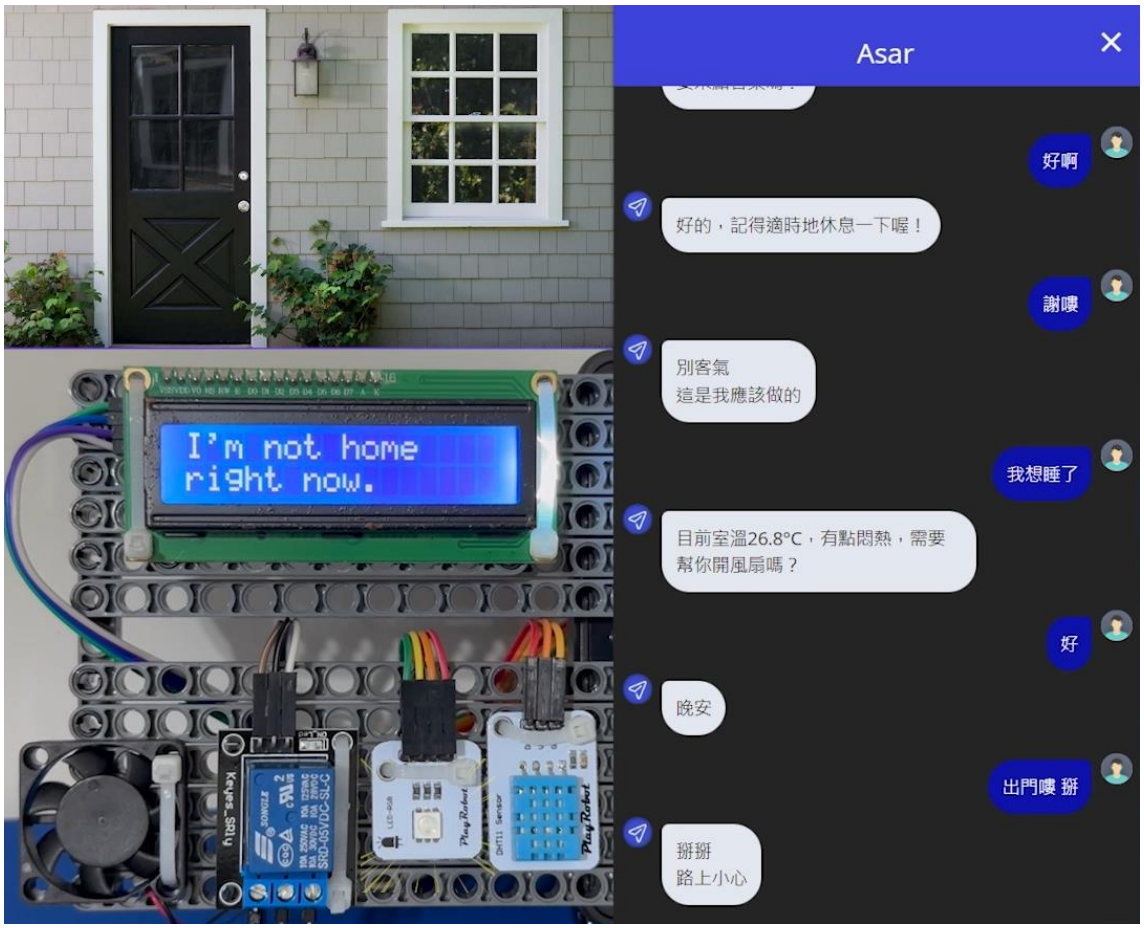

圖 4-18 外出情境

使用者可以請管家控制家中的電器設備，就算沒有提供明確的指令，管家也能識別出正確的意圖（圖 4-19）。此外，管家還能透過外部 API 提供使用者網路上的即時資訊，例如 Covid-19 的即時資訊（圖 4-20）。

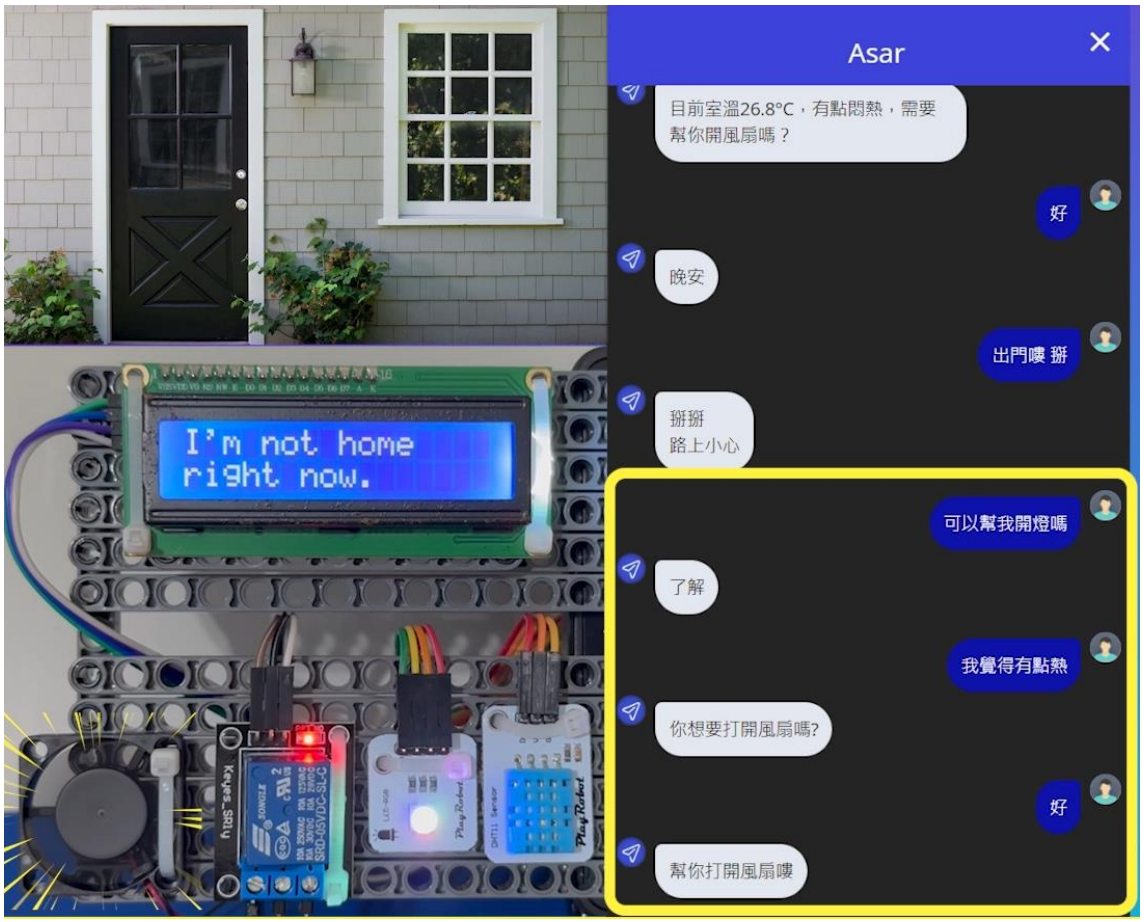

圖 4-19 控制電器設備

圖 4-20 Covid-19 的即時資訊

第5章 結論與未來展望

5.1. 結論

本專題提出並實現了能夠輕鬆部署聊天機器人服務的解決方案，開發出一款名為 Asar，運作於樹梅派的輕量級聊天機器人設計平台，並實現了以下目標：

(1) 具備 NLU 能力的聊天機器人

應用 Transformer、ALBERT 深度學習模型，並透過一系列的 NLU 流程，包括斷詞、特徵化、實體提取、意圖識別、回覆與動作決策，使聊天機器人具備 NLU 的能力。

(2) 簡化部署流程

Asar 平台以容器的方式安裝與運行，並提供一條龍的聊天機器人服務部署流程，包括建立聊天劇本、訓練對話模型、控制樹莓派周邊設備、串接外部 API、串接通訊平台，各項功能分別由四個子服務提供。聊天機器人服務負責訓練、執行對話模型，以及串接通訊平台；動作代理服務負責執行非對話任務；前端網頁服務提供人性化的設計工具；後端 API 服務是所有子服務的溝通橋樑。

(3) 人性化的設計工具

以人性化的設計工具取代文字編輯器，提供齊全的設計工具讓管理員能更有效率地設計聊天機器人，設計工具包括專案管理工具、訓練資料建立工具、視覺化劇本設計工具、視覺化程式設計工具，管理員透過網頁瀏覽器來設計聊天機器人。

(4) 控制樹梅派周邊設備與串接外部 API

透過專屬的視覺化程式設計工具，管理員能夠使用以「方塊」為概念的程式設計模式，更加直覺、簡單地撰寫非對話任務的程式碼，輕鬆將樹莓派周邊設備與外部 API 串接至聊天機器人服務中。

(5) 注重個人隱私，降低敏感資訊外流的風險

聊天訊息的推論完全在本地端執行，不必依賴雲端運算，從而提高個人隱私的安全性。

5.2. 未來展望

最初 Asar 的目標客群是針對臺灣人所開發，未來 Asar 將與國際接軌，擴大目標客群實現國際化，以下是未來目標：

(1) 支援多國語言

目前 Asar 支援中文與英文，未來將陸續新增各國語言，初步以日文與韓文等鄰近國家的語言為優先。

(2) 提供多元的通訊平台接口

目前 Asar 提供了臺灣人常用的 Line、Facebook Messenger、Telegram 接口，未來將新增 WhatsApp、Twitter、Discord 等國際熱門通訊平台的接口，讓各個國家、地區的使用者能夠使用自己習慣的通訊平台與聊天機器人交談。

(3) 推廣到教育市場

目前 Asar 主要用於個人與商業用途，未來可推廣到教育用途，並針對各個教育階段提供合適的學習教材。

第6章 參考文獻

[1] STAMFORD, Conn. (2022). Gartner Predicts Chatbots Will Become a Primary Customer Service Channel Within Five Years. Gartner newsroom. https://www.gartner.com/en/newsroom/press-releases/2022-07-27-gartner-predicts-chatbots-will-become-a-primary-customer-service-channel-within-five-years

[2] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. (2017). Rasa: Open Source Language Understanding and Dialogue Management. arXiv preprint arXiv:1712.05181

[3] Vincent Warmerdam. Computing machinery and intelligence. (2021). Bending the ML Pipeline in Rasa 3.0. Rasa. https://rasa.com/blog/bending-the-ml-pipeline-in-rasa-3-0/

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. (2017). Attention is all you need. Advances in neural information processing systems. arXiv preprint arXiv:1706.03762.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

[7] Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. (2020). Diet: Lightweight language understanding for dialogue systems. arXiv preprint arXiv:2004.09936.

[8] Vladimir Vlasov, Johannes EM Mosig, and Alan Nichol. (2019). Dialogue transformers. arXiv preprint arXiv:1910.00486.

[9] Custom Graph Components. (2022). Rasa & Rasa Pro Documentation. https://rasa.com/docs/rasa/custom-graph-components/