# 在 MA35D1 Buildroot 上實現 ChatGPT

NuMicro® 64 位元系列微控制器範例程式碼介紹

## 文件資訊

| | |
|---|---|
| 應用簡述 | 本程式在 Nuvoton Buildroot 平台上，利用 OpenAI API 來實現 ChatGPT，包括聊天功能和圖像生成功能等。 |
| BSP 版本 | MA35D1 Series Buildroot 2024 |
| 開發平台 | NuMaker-IoT-MA35D16F90 V2.4 |

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

*For additional information or questions, please contact: Nuvoton Technology Corporation.*

www.nuvoton.com

# 1. 概述

此範例程式利用 OpenAI API 與 OpenAI 伺服器進行溝通，進而提供 ChatGPT 功能，包括聊天和圖像生成等。它提供了一個簡單而方便的方式，讓使用者可以通過命令列介面與 OpenAI 的眾多 AI 模型進行互動。

這個程式的目的是在嵌入式平台上實現強大的 AI 能力，為使用者提供自然流暢的對話體驗和圖像生成功能。它可以應用於各種應用場景，如智慧家居、機器人等，為設備帶來更智慧的互動體驗。

此範例程式是基於 Nuvoton Buildroot 2024 的環境來進行開發，需先建立 Nuvoton Buildroot 2024 開發環境 (https://github.com/OpenNuvoton/buildroot_2024)，才能夠執行此範例程式。

雖然在這篇文章中，是以 NuMaker-IoT-MA35D16F90 V2.4 開發板來舉例。但是這篇文章的內容適用於所有支援 Nuvoton Buildroot 2024 的開發板，包含 NUC970、NUC980、N9H30、MA35D0、MA35D1 和 MA35H0 等系列。

## 1.1 原理

OpenAI API 是一個開放的應用程式介面，用於訪問 OpenAI 的語言模型。而 ChatGPT 是 OpenAI API 中的一個特定模型，專門用於聊天和對話式應用。

ChatGPT 的特點是可以進行對話式的互動，它能夠理解上下文並生成相應的回應。這是通過將用戶的輸入和之前的對話作為模型的輸入，並使用生成式的方法來生成回應。ChatGPT 還具有一定的自主性，可以在生成回應時表現出一定的創造性。

OpenAI API 支援多種不同方式的函式庫，用以跟 OpenAI 伺服器進行溝通，其包括 curl、python、node.js 和 csharp，本程式是使用 curl 方式來與 OpenAI 伺服器進行溝通。當使用者在命令列介面上輸入指令時，程式會使用 curl 函式庫來建立 HTTP 請求，將指令作為請求的內容發送到 OpenAI 伺服器的 API 端點。伺服器在接收到請求後，使用 ChatGPT 或其他使用者指定的模型進行處理，並生成相應的回應或圖像。然後，伺服器將生成的結果作為 HTTP 響應返回給程式，程式再將響應解析並顯示在命令列介面上供使用者查看。這樣，使用者就可以通過這個程式使用 curl 方式與 OpenAI 伺服器進行溝通，實現 ChatGPT 功能。

## 1.2 環境設定

此範例程式主要是以 bash 的語法所撰寫，在程式中會使用到許多 Linux 工具。因此在執行前，必須要確保這些工具是可以正常運行的。因為在 Buildroot 中，可以很容易的支持這些所需的工具。所以在本篇文章中，就以 Buildroot 為開發環境。

MA35D1 環境架設，產生映像檔可參考新唐 MA35D1 Buildroot Project User Manual。

- **Buildroot**

於 Buildroot 的 menuconfig 中啟用此範例程式所需的套件，請按照以下步驟進行：

1. 透過執行 make menuconfig 命令來打開 Buildroot 的配置選單。在套件選擇選單中，可以選擇相應的選項並按下空格鍵來啟用套件。也可以直接使用「/」鍵，用套件的名稱來搜索套件。

2. 範例程式主要是以 bash 的語法所撰寫，而且在輸入的文字處理上會使用到 readline 和 Unicode 等功能。所以需要啟用相關套件。

   - Target packages ---> [*] Show packages that are also provided by busybox
   - Target packages ---> Shell and utilities ---> [*] bash
   - Target packages ---> Libraries ---> Text and terminal handling ---> [*] ncurses
   - Target packages ---> Libraries ---> Text and terminal handling ---> [*] enable wide char support

3. 範例程式是使用 curl 的方式來與 OpenAI 伺服器進行溝通。雙方在進行溝通時，會使用 JSON 格式。而且必須確保系統的時間是準確的，否則 curl 會出現異常。所以需要啟用 curl、jq 和 ntp 等相關套件。

   - Target packages ---> Libraries ---> Networking ---> [*] libcurl
   - Target packages ---> Libraries ---> Networking ---> [*] curl binary
   - Target packages ---> Libraries ---> Networking ---> SSL/TLS library to use (OpenSSL) --->
   - Target packages ---> Libraries ---> Crypto ---> [*] CA Certificates
   - Target packages ---> Development tools ---> [*] jq
   - Target packages ---> Networking applications ---> [*] ntp

4. 對於輸出的文字，我們會使用 gawk 來對其進行排版處裡。所以需要啟用 gawk 套件。

   - Target packages ---> Development tools ---> [*] gawk

   需要注意到的是，若要處理 Unicode，則 C 函式庫環境需要支持 UTF-8。在 Buildroot 預設是使用 glibc 函式庫，你可以選擇改用 musl 函式庫。因為 musl 函式庫預設就有支持 UTF-8，所以直接就能夠支持處理 Unicode。

   - Toolchain ---> C library (musl) --->

   若要維持原先的設定使用 glibc 函式庫，則需要為 glibc 函式庫建立 locale 環境。以下是以 C.UTF-8 為例。

   - System configuration ---> [*] Purge unwanted locales
   - System configuration ---> (C.UTF-8) Locales to keep
   - System configuration ---> (C.UTF-8) Generate locale data

5. 除了範例程式之外，我們還要將相關的系統設定檔案放到裝置之中。所以在配置選單內，我們必須要指定一個根檔案系統覆蓋目錄。而之後我們所新增或修改的檔案，就可以直接放到這個目錄來。

   - System configuration ---> (board/nuvoton/ma35d1/rootfs_overlay) Root filesystem overlay directories

6. 啟用所有需要的套件後，退出 menuconfig 並保存配置。

7. 將 *rootfs_overlay* 整個目錄拷貝到 *board/nuvoton/ma35d1/*目錄下，以符合前面定義的根檔案系統覆蓋目錄。這樣能夠確保範例程式和系統設定檔案，都擺放在 Buildroot 系統的正確位置。

8. 通過 make 命令使用更新的配置來構建 Buildroot 系統。

編譯完成後，使用 NuWriter 將映像檔燒錄到 NuMaker-IoT-MA35D16F90 開發板。

## 1.3 執行結果

本範例程式的 Buildroot 映像檔執行編譯結果，如圖 1-1 所示。在編譯完成後，可以在 *output/images/* 目錄下找到產生的映像檔。

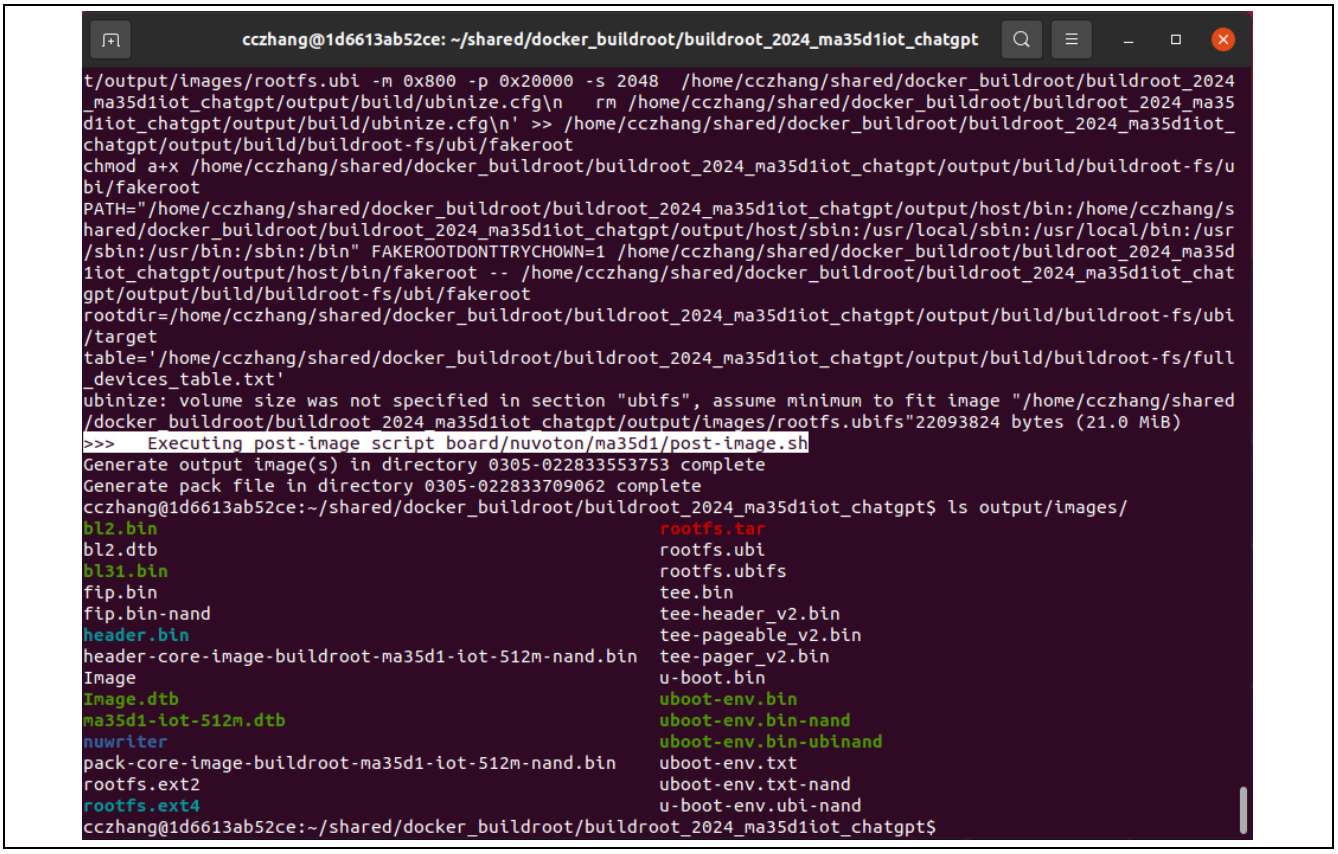


圖 1-1 Buildroot 執行編譯結果

本範例程式的 Buildroot 映像檔執行燒錄結果，如圖 1-2 所示。

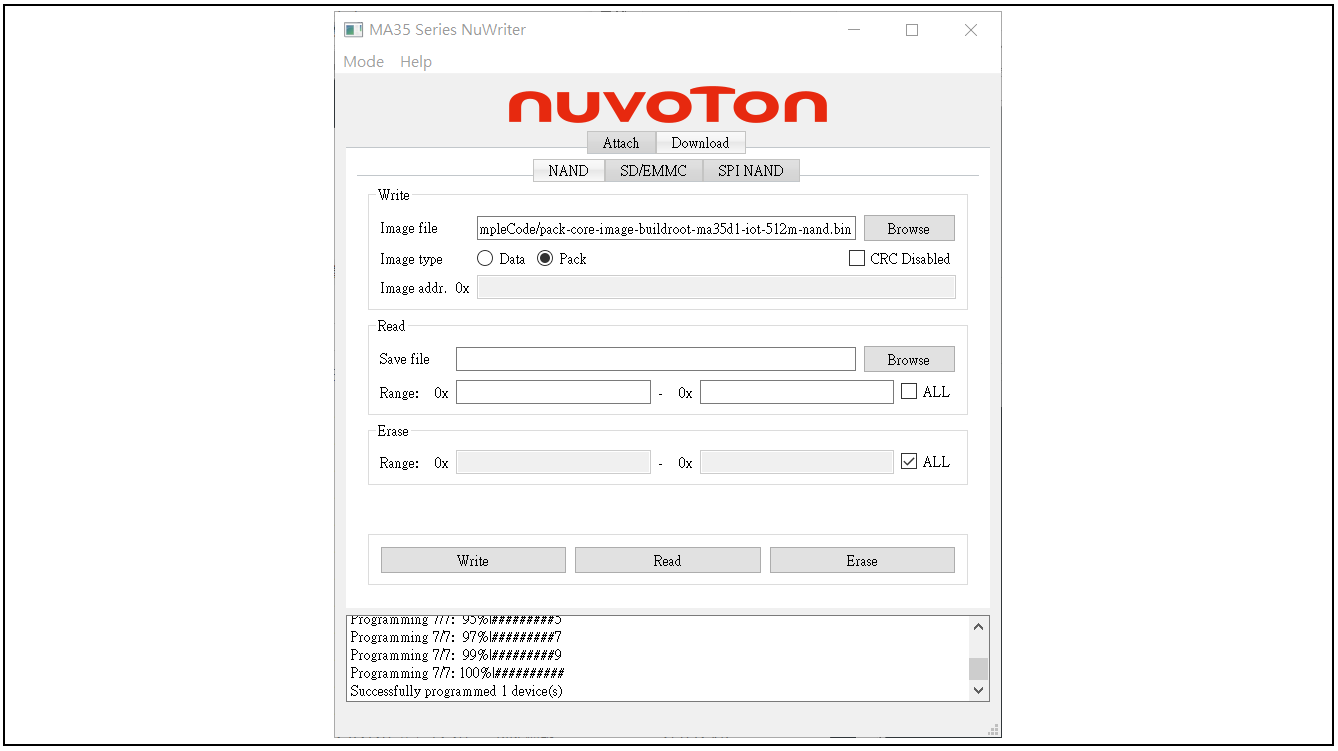


圖 1-2 使用 NuWriter 燒錄 Buildroot 映像檔

本範例程式在終端軟體下的聊天功能執行結果，以 Putty 為例，如圖 1-3 和圖 1-4 所示。在聊天的過程中，可以和 ChatGPT 進行互動。

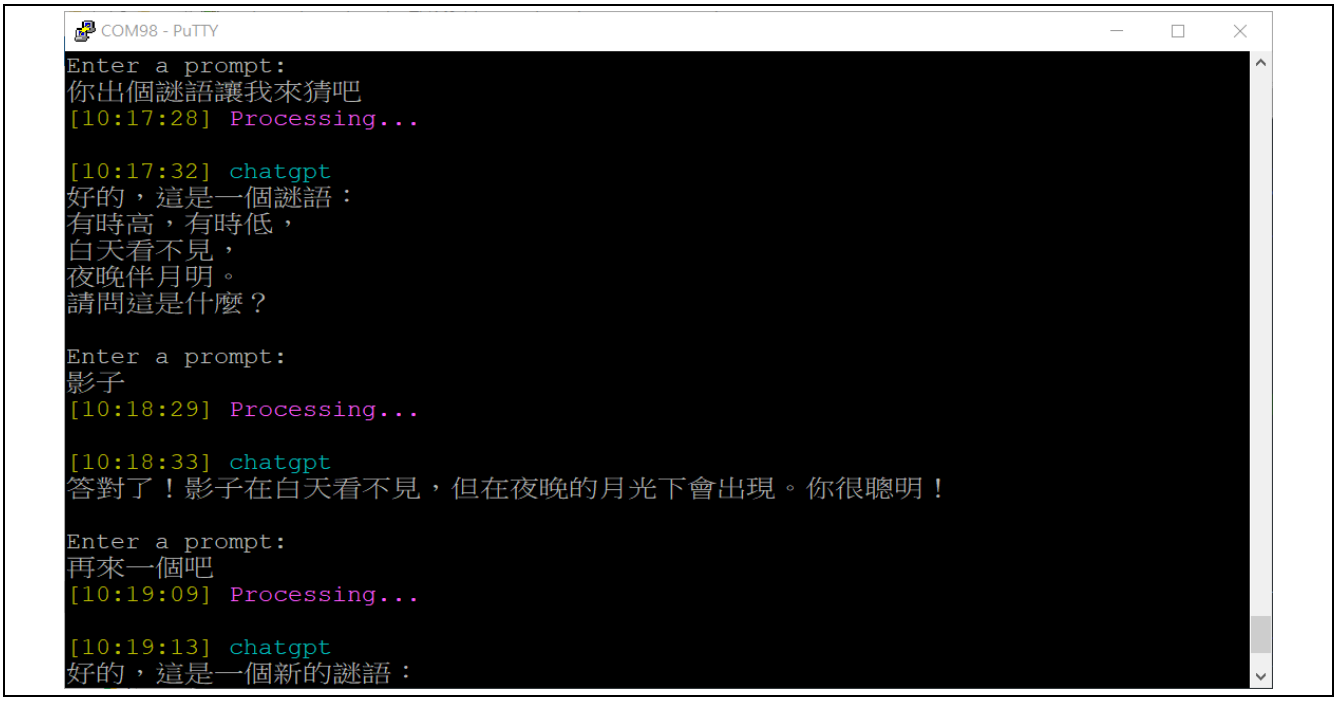


圖 1-3 ChatGPT 聊天功能範例 1

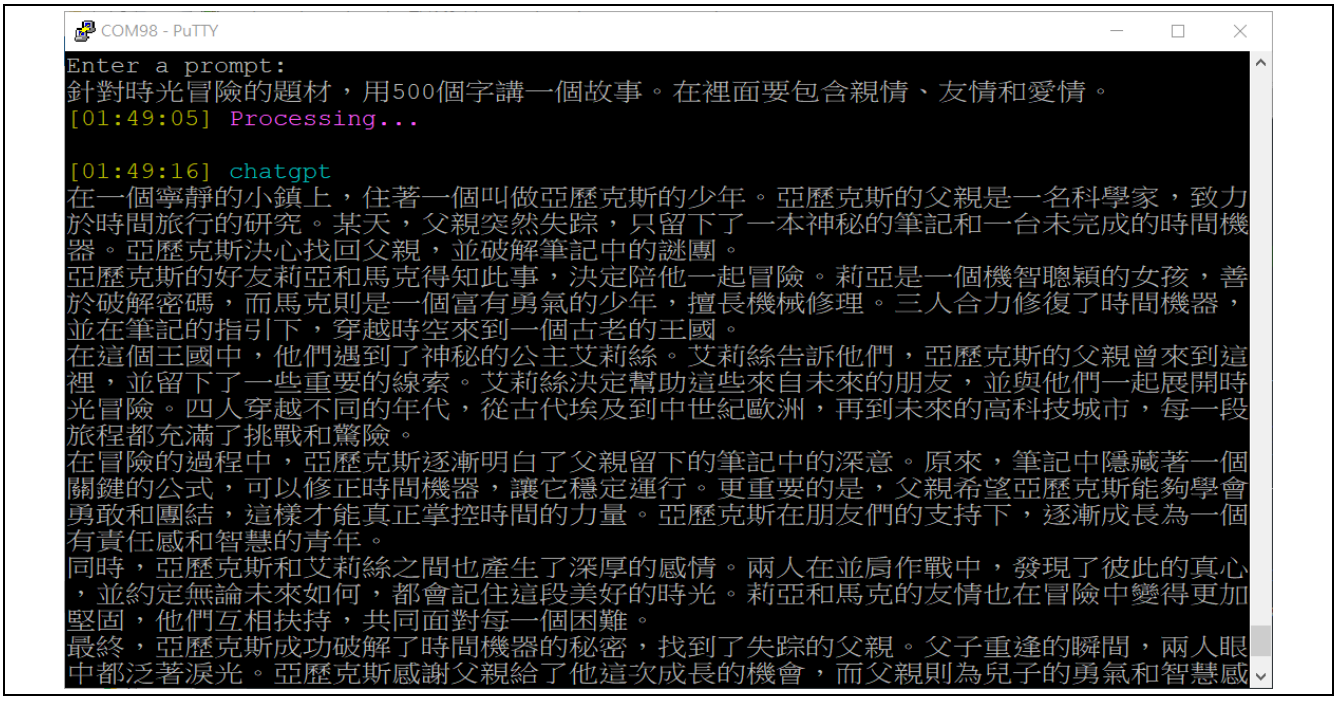


圖 1-4 ChatGPT 聊天功能範例 2

本範例程式的圖像生成功能執行結果，如圖 1-5 所示。圖像生成的指令為「image:」，在其之後輸入圖像生成的提示，ChatGPT 就會生成對應的圖像。在生成的過程結束時，在視窗上會顯示生成圖像的 URL。之後也會將此圖像下載至本地來，圖檔的名稱為 *image.png*。

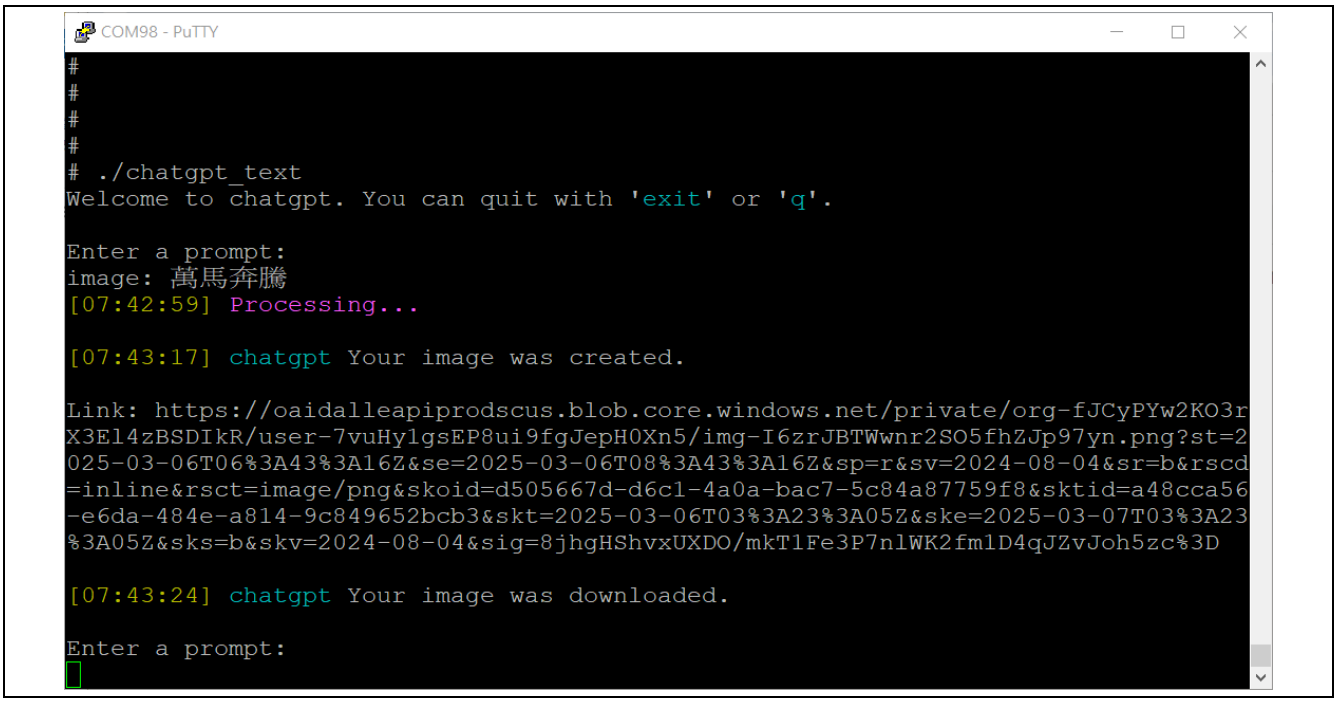


圖 1-5 ChatGPT 圖像生成功能範例

在圖像生成時，提示可以是一個簡單的詞語。比如：

**萬馬奔騰**

其結果如圖 1-6 所示。

圖 1-6 ChatGPT 生成圖像 1

提示也可以是一整段較複雜的文字段落，比如：

**在這場激烈的戰爭中，精靈、矮人和人類聯合起來，面對邪惡的魔族和黑暗的半獸人。彼此的勇士奮勇作戰，劍與斧相交，箭矢如雨。精靈的弓箭手在樹梢上射下死亡的雨幕，矮人的戰斧劈開敵人的盾牌，人類的勇士衝鋒陷陣。火焰和魔法交織，為這片戰場增添了驚心動魄的景象。種族間的血戰持續著，但他們的團結和勇氣使他們能夠抵抗黑暗勢力，保衛著自由和和平的希望。**

其結果如圖 1-7 所示。

圖 1-7 ChatGPT 生成圖像 2

提示也可以是一段對圖像生成的描述，其中包含著許多不同的條件要求，比如：

**以小白兔和狐狸在打架為主題，畫一個 3D 風格的四格漫畫。**

其結果如圖 1-8 所示。

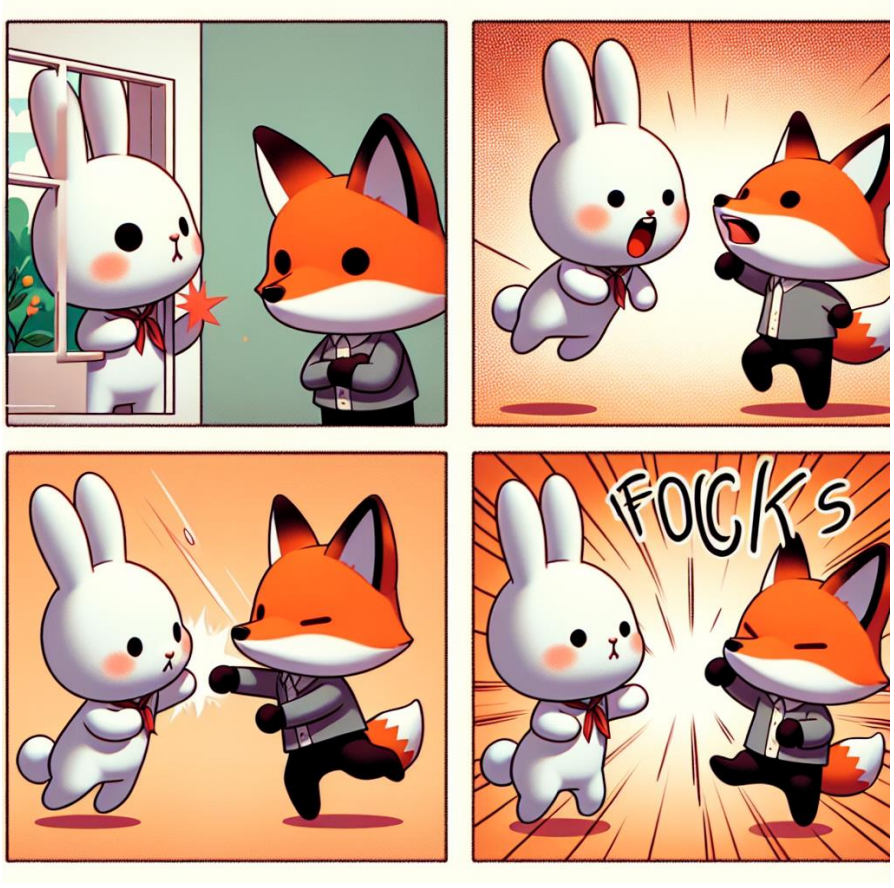


圖 1-8 ChatGPT 生成圖像 3

## 2. 程式碼介紹

因為在生成式AI的文本中，常常會使用到一些特殊字元。而這些特殊字元，在使用OpenAI API時會造成格式解析的錯誤。因此必須要對其進行事先處理，以確保不會影響到程式運行。

```
# escapes special characters to ensure the string can be safely used
# in a JSON-compatible format
# $1 should be the message(s) to do escape
escape() {
    echo "$1" | jq -Rrs 'tojson[1:-1]'
}
```

當使用者輸入訊息給 ChatGPT 後，在呼叫 OpenAI API 之前，必須要將其先打包成 ChatGPT 模型的標準格式。另外 ChatGPT 的互動回應也會跟之前的對話相關，所以這段新的使用者訊息會加在對話消息列表的最後，再一起作為模型的輸入文本。

```
# build user chat message function for /chat/completions (gpt models)
# builds chat message before request,
# $1 should be the escaped request prompt,
# it extends $chat_message
build_user_chat_message() {
    local escaped_request_prompt="$1"
    if [ -z "$chat_message" ]; then
        chat_message="{\"role\": \"user\", \"content\": \"$escaped_request_prompt\"}"
    else
        chat_message="$chat_message, {\"role\": \"user\", \"content\":
\"$escaped_request_prompt\"}"
    fi
}
```

ChatGPT 的互動回應跟之前的對話相關，這裡面除了使用者的輸入訊息之外，也包含了 ChatGPT 之前的回應。所以當 ChatGPT 有了回應之後，這段回應也要加在對話消息列表的最後。

另外在 OpenAI API 中，token 是指文本的最小單位，而 API 的價格和限制是基於 token 數量計算的。在程式中會設定一個最大 token 數量的限制，如果輸入文本的 token 數量超過了這個限制，在這邊也會進行縮減。其規則是從最舊的資料開始刪除，直到符合限制值為止。這個限制值預設是 4096，使用者可以根據自己的需求來進行調整。

```
# adds the assistant response to the message in (chatml) format
# for /chat/completions (gpt models)
# keeps messages length under max token limit,
# $1 should be the escaped response data
# it extends and potentially shrinks $chat_message
add_assistant_response_to_chat_message() {
    local escaped_response_data="$1"
```

```
    # add response to chat context as answer
    chat_message="$chat_message, {\"role\": \"assistant\", \"content\":
\"$escaped_response_data\"}"

    # check prompt length, 1 word =~ 1.3 tokens
    # reserving 100 tokens for next user prompt
    while (($(echo "$chat_message" | wc -c) * 13 / 10 > (MAX_TOKENS - 100))); do
        # transform to json array to parse with jq
        local chat_message_json="[ $chat_message ]"
        # remove first/oldest QnA from prompt
        local chat_message_removed=$(echo "$chat_message_json" | jq -c '.[2:]')
        # remove the json array format
        chat_message=${chat_message_removed:1:-1}
    done
}
```

下列為聊天功能的程式碼，其呼叫 OpenAI API 的 curl 函式庫，向 OpenAI 伺服器請求使用 ChatGPT 模型並給定包含對話的訊息列表，之後模型將會傳回回應。在 OpenAI API 中，參數 model 為 ChatGPT 模型，預設為 gpt-4o。參數 messages 是迄今為止的對話消息列表，也就是此函式的輸入參數再經由一些處理，以確保符合 OpenAI API 的參數格式。參數 max_completion_tokens 指定了生成文本的最大 token 數量，用於控制生成結果的長度，其預設值為 4096。參數 temperature 的作用，是控制生成文本的隨機性和多樣性。其值的可設定範圍介於 0 到 2 之間，預設值為 0.7。當 temperature 設置為較高的值時，生成的文本會更加隨機和多樣化。這意味著模型有可能會選擇概率較低的選項，並產生一些更加出人意料的回應。相反，當 temperature 設置為較低的值時，生成的文本會更加重視確定性和一致性。若要進一步了解 OpenAI API 內的所有參數，可以參考 OpenAI API reference。

```
# request to OpenAPI API chat completion endpoint function
# $1 should be the message(s) formatted with role and content
request_to_chat() {
    local message="$1"
    escaped_system_prompt=$(escape "$SYSTEM_PROMPT")

    curl https://api.openai.com/v1/chat/completions \
     -sS \
     -H 'Content-Type: application/json' \
     -H "Authorization: Bearer $OPENAI_KEY" \
     -d '{
         "model": "'"$MODEL"'",
         "messages": [
             {"role": "system", "content": "'"$escaped_system_prompt"'"},
             '"$message"'
         ],
         "max_completion_tokens": '$MAX_COMPLETION_TOKENS',
         "temperature": '$TEMPERATURE'
     }'
}
```

因為 ChatGPT 模型的回應文本，可能會包含多國語言。這邊針對一些常見的語言，例如英文、中文、日文等，在顯示回應文本到終端視窗時會進行換行的處裡，以利於閱讀。

```
# function to wrap text considering different languages
# $1 should be the response body
text_display() {
echo "$1" | gawk -v max_width=$MAX_WIDTH '
{
    # initialize variables
    line = ""  # current line content
    len = 0  # current line length

    # split the line into words by spaces
    n = split($0, words, " ")

    # loop through each word
    for (i = 1; i <= n; i++) {
     word = words[i]  # extract the current word
     word_len = 0  # length of the current word (consider ASCII and non-ASCII characters)

     # calculate the length of the word (English words and non-ASCII characters)
     for (j = 1; j <= length(word); j++) {
          char = substr(word, j, 1)

          # check if the character is an ASCII character (1 space)
          if (char ~ /[ -~]/) {
               word_len += 1
          } else {
               # non-ASCII characters (Chinese, Japanese) take 2 spaces
               word_len += 2
          }
     }
    …
    …
}
```

如果在使用 OpenAI API 時，OpenAI 伺服器有回報錯誤的話，這邊會將錯誤的相關訊息顯示在終端視窗上。

```
# error handling function
# $1 should be the response body
handle_error() {
    if echo "$1" | jq -e '.error' >/dev/null; then
     echo -e "Your request to Open AI API failed: \033[0;31m$(echo "$1" | jq -r
'.error.type')\033[0m"
     echo "$1" | jq -r '.error.message'
     exit 1
    fi
}
```

下列為圖像生成功能的程式碼，其呼叫 OpenAI API 的 curl 函式庫，向 OpenAI 伺服器請求使用 dall-e-3 模型並給定提示，之後模型將會生成新圖像。在 OpenAI API 中，參數 model 為圖像生

成模型，預設為 dall-e-3。參數 prompt 為圖像生成的提示，此函式的輸入參數為圖像生成的指令「image:」再加上圖像生成的提示。在將圖像生成的指令移除後，僅保留圖像生成的提示來作為此項參數設定。參數 n 為生成圖像的數量，預設值為 1。參數 size 為生成圖像的尺寸，預設值為 1024x1024。

```
# request to OpenAI API image generations endpoint function
# $1 should be the prompt
request_to_image() {
    local prompt="$1"
    image_response=$(curl https://api.openai.com/v1/images/generations \
    -sS \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $OPENAI_KEY" \
    -d '{
        "model": "dall-e-3",
        "prompt": "'"${prompt#*image:}"'",
        "n": 1,
        "size": "'"$SIZE"'"
    }')
}
```

除了 dall-e-3 模型之外，另外還可以選擇舊版的 dall-e-2 模型，以下表格為上述參數在兩者間的差別。使用者可以根據自己的需求，來選擇圖像生成的模型。

| OpenAI API 參數 | dall-e-2 | dall-e-3 |
|---|---|---|
| prompt | 最大為 1000 字元 | 最大為 4000 字元 |
| n | 1~10 | 1 |
| size | 256x256<br>512x512<br>1024x1024 | 1024x1024<br>1792x1024<br>1024x1792 |

表 2-1 dall-e-2 與 dall-e-3 模型之參數比較

下列是一些 OpenAI 模型的參數預設值，設定使用者可以進行修改以符合其需求。在其中只有最後一項 SIZE 是跟圖像生成功能有關，其餘的參數皆是跟聊天功能有關。需要注意到的是，參數 MAX_COMPLETION_TOKENS 只有在 GPT 4 以後的模型，才會有支援。

```
# set defaults
MODEL=${MODEL:-gpt-4o}
MAX_TOKENS=${MAX_TOKENS:-4096}
MAX_COMPLETION_TOKENS=${MAX_COMPLETION_TOKENS:-4096}
TEMPERATURE=${TEMPERATURE:-0.7}
MAX_WIDTH=${MAX_WIDTH:-80}
SIZE=${SIZE:-1024x1024}
```

在程式最下方的 while 迴圈，則是整個聊天功能和圖像生成功能的主要程式碼。在提示可輸入時，若使用者輸入「exit」或「q」，則結束程式運行。若使用者輸入「image:」開頭的字串，則進行圖像生成功能。除此之外，皆進行聊天功能。在進行圖像生成功能時，會將生成圖像結果的遠端 URL 顯示在視窗中，也會將圖像下載至本地目錄。

```
while $running; do

    …
    …

    if [[ $prompt =~ ^(exit|q)$ ]]; then
         running=false
    elif [[ "$prompt" =~ ^image: ]]; then
         request_to_image "$prompt"
         handle_error "$image_response"
         image_url=$(echo "$image_response" | jq -r '.data[0].url')
         if [ "$time_info" = true ]; then
             echo -ne "\n\033[33m["`date "+%H:%M:%S"`"] \033[0m"
         fi
         echo -e "${CHATGPT_CYAN_LABEL}Your image was created. \n\nLink: ${image_url}\n"

         if [ -f image.png ]; then
             rm image.png
         fi
         curl -sS $image_url -o image.png
         if [ "$time_info" = true ]; then
             echo -ne "\033[33m["`date "+%H:%M:%S"`"] \033[0m"
         fi
         echo -e "${CHATGPT_CYAN_LABEL}Your image was downloaded."
    else
         # escape quotation marks, new lines, backslashes...
         request_prompt=$(escape "$prompt")

         build_user_chat_message "$request_prompt"
         response=$(request_to_chat "$chat_message")
         handle_error "$response"
         response_data=$(echo "$response" | jq -r '.choices[].message.content')

         if [ "$time_info" = true ]; then
             echo -ne "\n\033[33m["`date "+%H:%M:%S"`"] \033[0m"
         fi
         echo -e "${CHATGPT_CYAN_LABEL}"
         text_display "${response_data}"
         echo -e "${CHATGPT_CYAN_LABEL}${response_data}" > response.txt
         add_assistant_response_to_chat_message "$(escape "$response_data")"

         timestamp=$(date +"%Y-%m-%d %H:%M")
         echo -e "$timestamp $prompt \n$response_data \n" >>~/.chatgpt_history
    fi
done
```

# 3. 軟體與硬體需求

## 3.1 軟體需求

- BSP 版本
  - MA35D1 Series Buildroot 2024

## 3.2 硬體需求

- 電路元件
  - NuMaker-IoT-MA35D16F90 V2.4

# 4. 目錄資訊

目錄結構如下圖所示。

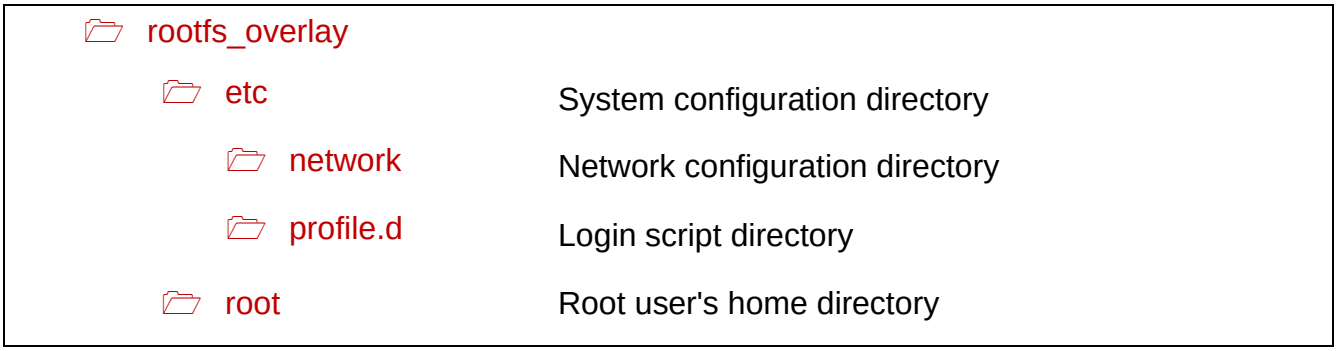


圖 4-1 目錄資訊

# 5. 範例程式執行

1. 用終端軟體連接上 NuMaker-IoT-MA35D16F90 開發板的 VCOM。

2. 在 Buildroot 開機後，需要確保裝置是可以正常連接上網路，並且系統時間是準確的，這樣才能夠正常執行 ChatGPT 範例程式。可以利用 ping 和 date 命令來進行確認。

```
# ping -c 1 api.openai.com
PING api.openai.com (172.66.0.243): 56 data bytes
64 bytes from 172.66.0.243: seq=0 ttl=56 time=28.304 ms

--- api.openai.com ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 28.304/28.304/28.304 ms
# date
Thu Mar  6 05:43:21 UTC 2025
```

3. 在系統環境中，設定自行申請的 OpenAI API key。

```
# export OPENAI_KEY=your_key_here
```

4. 執行 ChatGPT 範例程式。關於這個部分，可以參考下列描述。

   - 在執行範例程式之前，可以使用「-h」參數來查看幫助訊息。

```
# ./chatgpt_text -h
A simple, lightweight shell script to use OpenAI's Language Models and DALL-E from the
terminal.

Commands:
  image: - To generate an image and store it named "image.png" in local, you also can open
the the image in your browser.

Options:
  -p, --prompt              Provide prompt instead of starting chat

  -l, --list                List available openAI models

  -m, --model               Model to use

  -t, --temperature         Temperature

  -s, --size                Image size. (The sizes that are accepted by the OpenAI API is
                            dall-e-3 model: 1024x1024, 1792x1024 or 1024x1792,
                            dall-e-2 model: 256x256, 512x512 or 1024x1024)
```

   - 可以直接執行範例程式進入交談模式，如同前述章節之執行結果。在對話中輸入一般文字時，會呼叫聊天功能。但是如果在文字前面加上「image:」時，就會呼叫圖像生成功能。

```
# ./chatgpt_text
```

- 在執行範例程式時，也可以直接指定參數的設定值，用以取代在程式碼中之預設值。以下為改成使用聊天模型 gpt-3.5-turbo 的範例。要注意到的是，許多的聊天模型其訊息格式與 gpt-4o 並不相容。若要使用這些模型，必須要先分析其訊息格式，並且修改範例程式的相關內容。

```
# ./chatgpt_text -m gpt-3.5-turbo
```

- 在執行時使用「-l」參數，可以列出官方目前所有支持的模型清單。

```
# ./chatgpt_text -l
```

- 在執行時使用「-p」參數，可以直接指定聊天提示並且得到回應，不用進入到交談模式。要注意到的是，此種方法只適合一次性的獨立提問，所以其回應不會與之前的文本有所互動。

```
# ./chatgpt_text -p "地球上總共有多少海水?"

[07:23:32]
地球上的海水總量約為13.32億立方千米，占地球表面約71%的面積。
#
```

# 6. 修訂紀錄

| Date | Revision | Description |
|---|---|---|
| 2025.04.25 | 1.00 | 初始發布。 |

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**

*Please note that all data and specifications are subject to change without notice.*
*All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*