中華民國第 61 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

052506

自然語言結合推薦算法開發程式碼生成器之研究

學校名稱：中山學校財團法人高雄市中山高級工商職業學校

作者：	指導老師：
職三 林頡辰 職三 簡齊君	楊鎮澤

關鍵詞：推薦演算、程式學習、自然語言

摘要

本研究希冀提供學生另一種程式的學習方式，減少學生對於程式的排斥感，同時亦能讓學生在學寫程式之前，先訓練其溝通能力，使其說話較具邏輯性的結構，如此一來將能更有效的表達其所要陳述的事件，我們自行開發一套軟體系統，可經由口語的方式與電腦溝通，同時電腦經過關鍵詞的辨識後，能更精準的透過推薦演算推出幾種可能的目標事件，透過選擇的方式，一步步引導使用者完成其想要達到的動作，最終目標是透過這個軟體自動產生程式碼，如此一來將可減少學生對於程式學習的恐慌，提高學生對於程式設計的學習成就，同時，請老師利用我們自行開發的程式學習輔助軟體實際教學後進行問卷調查，亦有超過百分之 64 的同學認為，這樣的軟體使他們對於程式學習更感興趣，也提升他們撰寫程式的成功率。

壹、 研究動機

這幾年全世界皆在倡導程式教育的重要，當然我們也看到很多的程式學習工具，其中最常見的便是以拼圖方式引導學生寫程式，如 Scratch 軟體，此軟體雖然操作簡單，也提供防呆的功能，但其實大部分學生的學習方式仍是把老師示範的畫面拍下來，然後照著老師的完成圖，找尋一樣的色塊一塊塊的拼上去，很多學生以為只要完成作業便是學習程式，但其實邏輯觀念仍然沒有進步。程式教育最重要的目地莫過於訓練學生的邏輯概念，運用良好的邏輯觀念訓練學生處理事務的能力，也訓練學生對事件描述的溝通能力，這兩者才是程式教育最重要的精神，至於是否會寫出程式，其實是其次，但現在很多學生的認知似乎不是如此，所以我們擬定「自然語言結合推薦算法開發程式碼生成器之研究」的計畫，開發一套軟體，讓學生練習講話，透過邏輯化的敘述，讓軟體來運用其演算法產生推薦的方塊，一步步引導學生講出他想要處理的事件，最後軟體將依據學生的敘述自動產生程式，學生只需要將程式複製使用，便能驗證其對於事件處理的描述是否正確，如此一來將能訓練學生的表達能力，對於日後在生活上或是工作上將能夠達到有效率的溝通。

本研究主題運用到的學校課程有以下幾門課：

A、程式設計實習：此課程中我們學習變數的定義及函數的運用，以及要設計一個程式時如何繪製流程圖，當然這門課裡面學習最多的就是邏輯的觀念，我覺得要能夠設計出一個

可幫助解決問題的程式，那麼邏輯思考能力要強，所以整個問題弄清楚很重要，如此才能夠想出解決問題的方法。 當然也因為這門課程，讓我們對程式產生興趣，透過日後的持續練習，繼續增進程式的開發能力。而在這個研究裡面，用較多的程式是 JavaScript 和 Python 。

B、單晶片微處理器實習 ：我們學校學到的是 arduino 開發板，因此我們利用它來當成我們程式驗證的對象，我們希望透過口語溝通的方式與電腦溝通，讓電腦自動產生符合 arduino 的程式語言，進而透過把程式上傳，將最終的結果呈現在 arduino 開發板上 。

C、人工智慧原理 ：這堂課裡面介紹了自然語言，因此我們對於自然語言的斷詞產生興趣，我們讓電腦學習如何找出我們講一句話裡的主詞，並且將一段話裡面重要的部分斷詞處理，也就是找出重點，去除贅詞，再將這些有用的詞去對應程式，並透過推薦程式的方式引導使用者逐步完成程式的開發 。

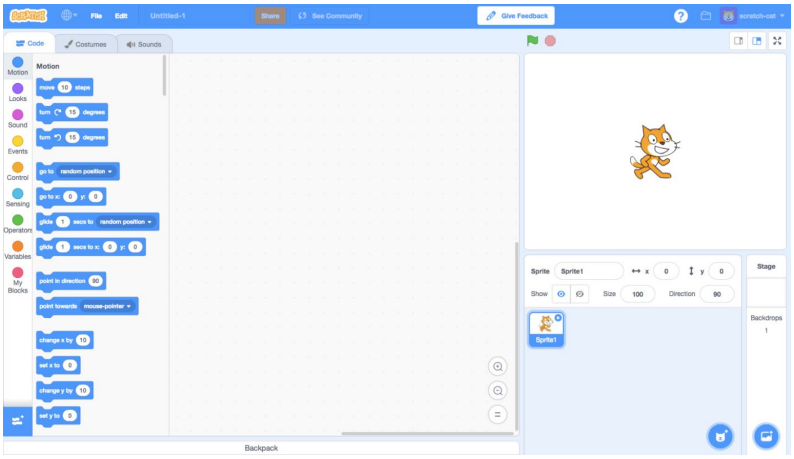


圖 1　Scratch 軟體

貳、 研究目的

(一)、 了解拼圖式的程式學習軟體

(二)、 了解語音辨識的原理

(三)、 了解推薦算法

(四)、 了解 Python 的撰寫

(五)、 了解 Electron Framework

(六)、 了解 Arduino CLI

參、 研究設備及器材

一、 筆記型電腦

圖 2 筆記型電腦

二、 Arduino UNO

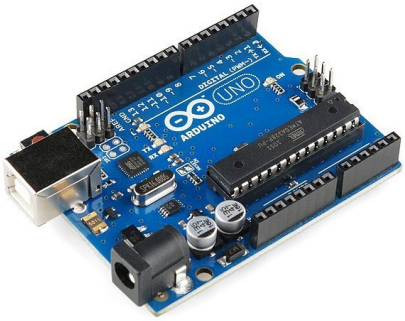


圖 3 Arduino UNO

三、 麥克風

圖 4 麥克風

四、 LED

圖 5 LED

五、　麵包板

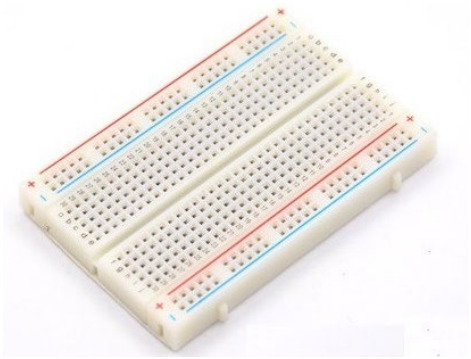


圖 6　麵包板

肆、　研究過程或方法

一、語音辨識（Speech Recognition）的原理

語音辨識，也被稱之為（Speech To Text，STT），是一種將聲音訊息轉換為文字的技術，如今被廣泛運用在各項科技領域，例如語音撥號、語音助手（例如：Siri、Google Assistant ）、物聯網控制（Google Nest、Homekit）、語音文件檢索，以及最為普遍的語音輸入。語音辨識技術涉及到訊號處理、圖型識別、概率論、發聲機理、資訊理論和人工智慧等等。

(一)原理

現在主流的大詞彙量語音辨識系統多數使用統計圖型識別技術。典型的統計圖型識別技術，由幾個基本模組組成：

1. 訊號處理及特徵辨識：主要任務是從輸入訊號中提取特徵，再經由聲學模型處理運算，同時也包括了一些訊號處理技術，以儘可能降低環境中的噪點、頻道衝突、說話人口齒不清等因素對特徵所造成的影響。
2. 聲學模型：多數識別系統採用基於一階隱馬爾科夫模型進行建模。
3. 發音辭典：發音辭典包含系統所理解的詞彙及對應的發音，提供了聲學模型建模單元和語言模型單元間的對應參考。

4. 語言模型：語言模型對系統所針對的語言建模，理論上所有正規語言，亦或者上下文無關文法在內的各種語言模型皆可作為語言模型，但主流還是採用基於統計的 N 元文法及其變種。
5. 解碼器：語音辨識的核心所在，主要任務是針對輸入的音訊，根據上述聲學、語言模型及詞典中找出最為相近的訊號，藉此輸出最高概率的正確文字。

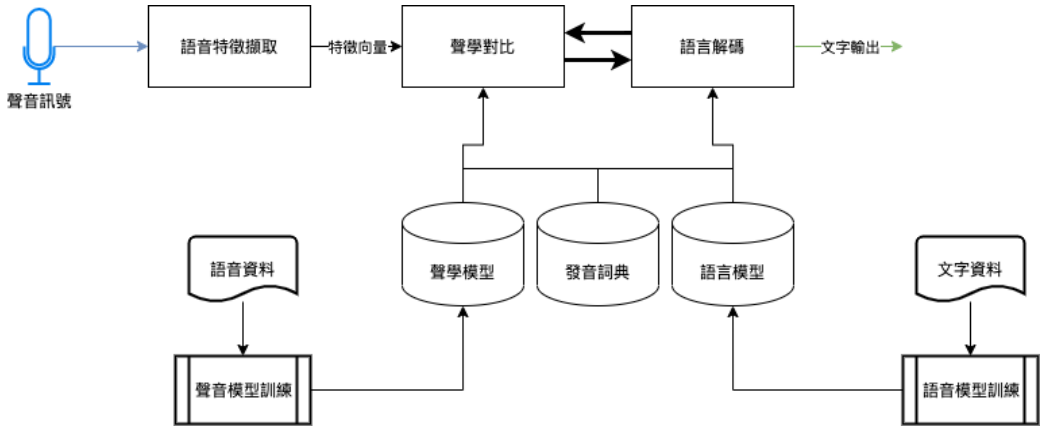


圖 7 語音辨識架構

(二)應用與開發過程

我們測試專案中採用的套件是 Python 支援的 SpeechRecognition 套件（pip）來實作 Google SpeechRecognition API。SpeechRecognition 內部支援各種語音辨識的 API，例如：（資料來源來自於官網：https://pypi.org/project/SpeechRecognition/）

- CMU Sphinx
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text
- Snowboy Hotword Detection (works offline)

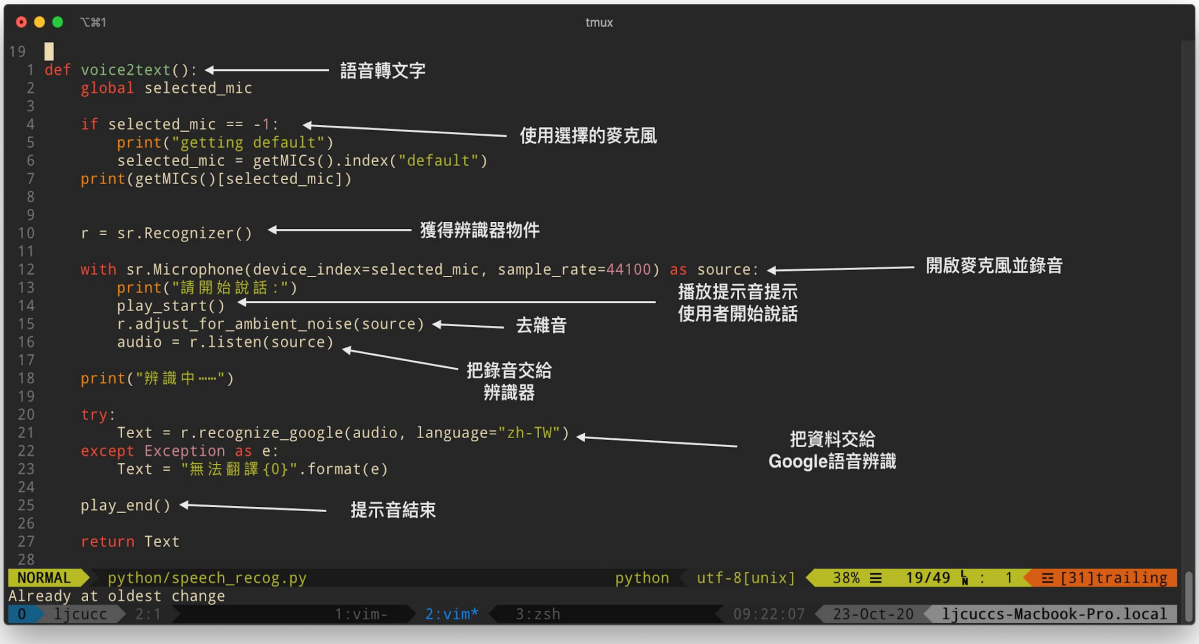


圖 8　語音辨識應用程式碼

二、減少運算成本的推薦算法

(一)針對使用者使用情況製作排序資料

所有的程式資料和程式模板都是預先組合好的版本（如圖 9），依照格式排序。其排序演算法使用 Pyc 實作之類型為 Quicksort，最壞的情況下需要$O(n^2)$次完成排序動作；普通情況下則需要$O(n\ log\ n)$次完成排序。

排序的順序，我們可以定義str為輸入字串，其是一由$word_{in}$所組成的有順序性之單字陣列。N集合為所有的關鍵字組，而一個關鍵字組可以$word_d = (W[size], id)$的方式存在（W為一有順序性之單字陣列，$size$為陣列大小），$result$是一個空陣列，用來紀錄與排列推薦結果。

演算法大略可以看成：

$$for\ each\ words_d\ in\ N, for\ each\ words_{in}\ in\ str:$$

$$if\ words_{in}\ is\ contained\ by\ words_d\ then$$

$$set\ words_{in}\ to\ rank\ and\ decrease\ \boldsymbol{rank}$$

$$and\ append\ words_{in}\ into\ result$$

並且我們還可以知道，單一使用者的輸入在經過推薦器的運算所產生的時耗$Lost_T$ 。如果 Pyc 一筆資料的處理時間是c，那我們可以推導出在普通情況下的時耗$Lost_T$：

$$n = size\ of\ result$$

$$Lost_T = c \cdot (size\ of\ N) \cdot (size\ of\ str) + c \cdot O(n\ log\ n)$$

```
狀態,腳位,設定,設為,按鈕,輸出,輸入:set_pin_state
輸出,設定,設為,LED,腳位,燈,高電位,低電位:set_pin_output
取得,獲得,讀取,輸入,按鈕,腳位:get_pin_input
閃爍,腳位:set_pin_blink
變數,設定,設為:set_var
新增,變數:new_var
重複,執行,迴圈,幾次,當:loop
```

圖 9　欲處理之資料模型結構

(二)預先生成資料庫

此外，我們也寫一個程式特別生成這種格式，使用純文字檔的目的是為了讓教育者（例如老師或營隊助教）方便安裝與修改為所需之程式（前中文字為引索，後為 ID）。接下來 ID 就會被用來查詢程式模板如下圖：

```
suggestion:
    undefined:
        type: undefined
        title: 未找到相關程式區塊
    set_pin_state:
        title: 設定腳位狀態
        description: 當程式開始時，讓電腦知道哪一個腳位要做什麼動作，例如讓腳位「3」「輸出」。
        type: set_pin_state
        fields:
            pin:
                title: 腳位編號
                type: select
                data: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1,A2,A3,A4,A5
            state:
                type: select
                title: 腳位類型
                data: 輸出, 輸入
        requirements:
            - arduino
    set_pin_output:
        title: 設定腳位輸出
        description: 設定某個腳位的電位輸出，共有兩種狀態— 高電位、低電位。
        type: set_pin_output
        fields:
            pin:
                title: 腳位編號
                type: select
                data: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1,A2,A3,A4,A5
            state:
                title: 輸出狀態
                type: select
                data: 高電位, 低電位
        requirements:
            - arduino
```

圖 10　推薦器前導資料

經過語音辨識的關鍵字，會被我們利用文字解析提取出來（如圖 11），且進行類似撞庫的比對運算，隨後把資料送進推薦器，其會將資料庫之初始資料載入到記憶體中，依照上述演算法運算完後，再利用 Flask 伺服 response 到 client 端的 Vue.js 做顯示，便會顯示已排序的程式片段供使用者做篩選。

```python
def tokenize(inputString):
    global keywords
    print(list(keywords))
    # print(list(code_id))

    print(inputString)

    result = []

    for key_obj in keywords:
        [keys, codeID] = key_obj

        id_added = False

        for index, keyword in enumerate(keys):
            # print(keyword)
            if(inputString.find(keyword) > -1):
                if(not id_added):
                    result.append([100-index, codeID])
                    id_added = True
                else:
                    result[len(result)-1][0]+=100-index

    def sort_func(item):
        return item[0]

    result = [item for item in result if item[0] >= 100]
    result.sort(key=sort_func, reverse=True)

    print(list(result))
    codeID_docs(result)

    return result

```

圖 11　詞法分析器

```python
def init():
    global keywords, codebase
    keywords = []

    print("init tokenizing ... ")

    keywordsFile = open(get_root()+"/assets/keywords.txt",  encoding="utf8")

    while True:
        line = keywordsFile.readline().replace("\n", "")

        if not line:
            break

        if(line.find(":")>-1):
            values = line.split(":")
            keywords.append([values[0].split(","), values[1]])
    print(list(keywords))

    with open(get_root()+"/assets/codebase.yml", "r",  encoding="utf8") as stream:
        codebase = yaml.load(stream, Loader=yaml.FullLoader)

```

圖 12　推薦器之啟動程式

```
def codeID_docs(id_list):
    global codebase

    for index, obj in enumerate(id_list):
        id_list[index] = obj[1]

    print(id_list)

    if(len(id_list) == 0):
        id_list.append(codebase['suggestion']['undefined'])
        return

    for index in range(len(id_list)):
        id_list[index] = codebase['suggestion'][id_list[index]]
```

圖 13　推薦器回傳之程式片段資訊

```
@app.route('/api/v1/load_suggestion')
def loadSuggestion():
    searchString = request.args.get('search')
    return jsonify(result=True, data=asc.tokenize(searchString))

@app.route('/api/v1/get_blocklist')
def getBlockList():
    return jsonify(result=True, data=blocks.getBlockList())

@app.route('/api/v1/get_blocks')
def getDisplayBlock():
    block_ids = request.args.get('q')
    return jsonify(result=True, data=blocks.getBlocks(block_ids.split(",")))

@app.route('/api/v1/get_block_template_list')
def getBlockTemplateList():
    return jsonify(result=True, data=blocks.getTemplateList())

@app.route('/api/v1/get_block_template')
def getBlockTemplate():
    block_ids = request.args.get('q').split(",")
    return jsonify(result=True, data=blocks.getTemplates(block_ids))

@app.route('/api/v1/get_code_template')
def getCodeTemplate():
    return jsonify(result=True, data=blocks.getCodeTemplate())

```

圖 14　推薦器被呼叫之入口—後端伺服器程式

三、前端軟體的溝通

我們的主要可以分為兩個部分，以 Electron Framework 實作而成的前端與由 Python

15）。

(一)Electron Framework

什麼是 Electron？Electron 是一個以 Chromium 作為 core 實作的一個簡易的網頁框架，可以允許我們利用網頁為一個程式建構優質的介面。我們就是利用 Electron 來為我們的應用程式實作 Blockly 的搭配。

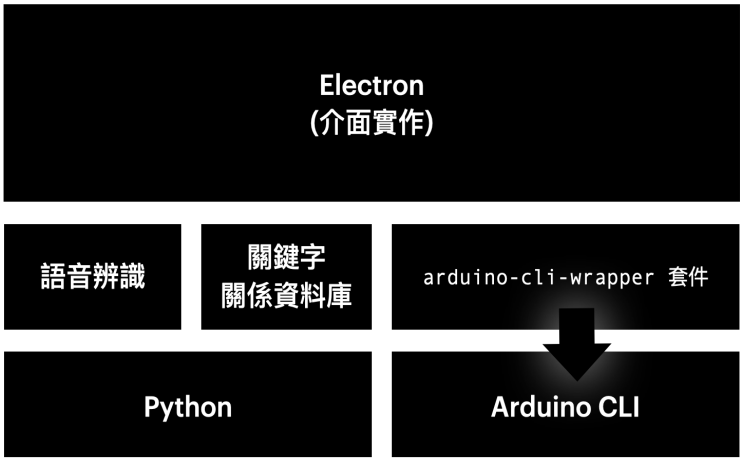


圖 15　軟體結構圖

1.語音辨識的替代

在先前的應用程式中，我們採用的是 Chrome 瀏覽器內建的 Web Speech API 將語音轉換成文字，因此我們必須先在伺服器上部署（Deploy）我們的網頁應用程式，並且在特定的瀏覽器與裝置上開啟，因為 iOS 的 Safari 和 Chrome 皆是使用 Webkit 作為 JavaScript 的運作技術，所以並不支持 Web Speech API。此外在 macOS 上的 Safari 也同為 Webkit，皆不支援 Web Speech API，必須使用支援 Web Speech API 或基於 Chromium 內核與其所衍生之瀏覽器。在後續為了改善使用者體驗，我們將網頁應用程式以及各個伺服器端程式全部打包且直接嵌在 Electron 的程式中，讓程式具有一體性，以利後續在其他教育組織推廣的時，可以一次性安裝全部的套件，沒有絲毫阻礙。

語音辨識的實作

我們會先把使用者的請求（GET method）利用 AJAX 發送至使用者電腦上部署的一個應用程式伺服器（這個應用程式伺服器會在應用程式剛啟動的時候就會開始運行），而應用程式伺服器就會利用 Python 的 SpeechReg 來辨識使用者的聲音（這個部分會使用到 PyAudio，後續會接著提到）。接下來就會直接 response（回傳）使用者所說的話（排序中最高的結果），然後結束 GET request。（如圖 16）

(二)前端介面

目前前端介面包含的功能有：Blockly 編輯介面、方塊建構介面（使用 Vue.js 實作與建構）、推薦器介面 （使用 Vue.js 實作與建構）。

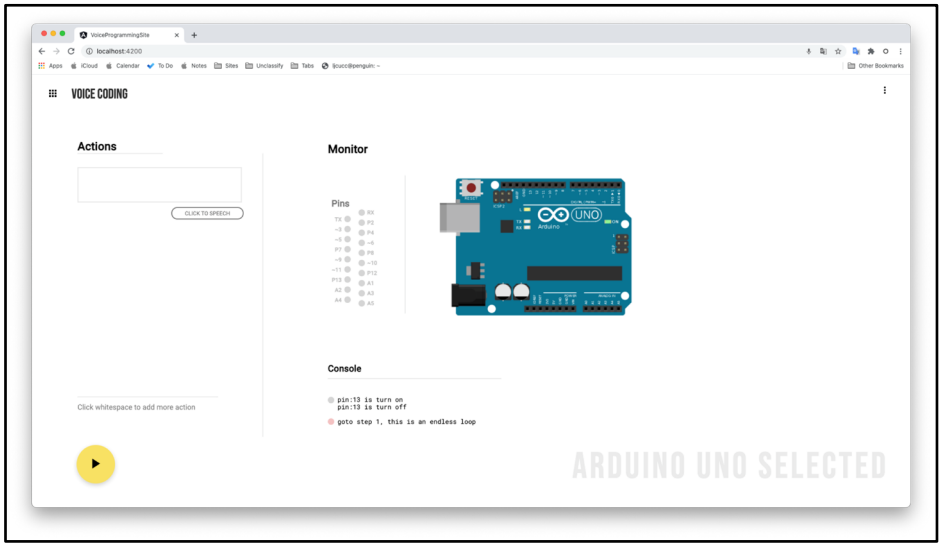


圖 16　前端介面之截圖與介紹（純網頁執行版本）

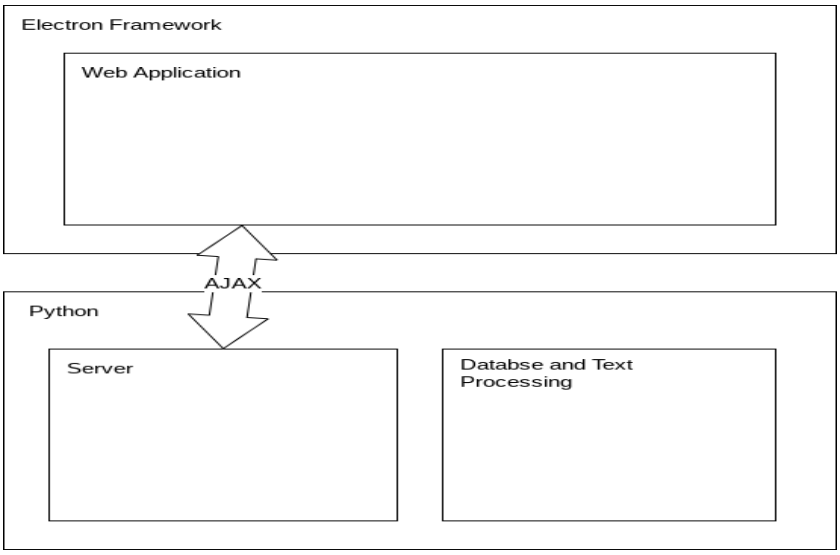


圖 17　前端與後段之間的關係

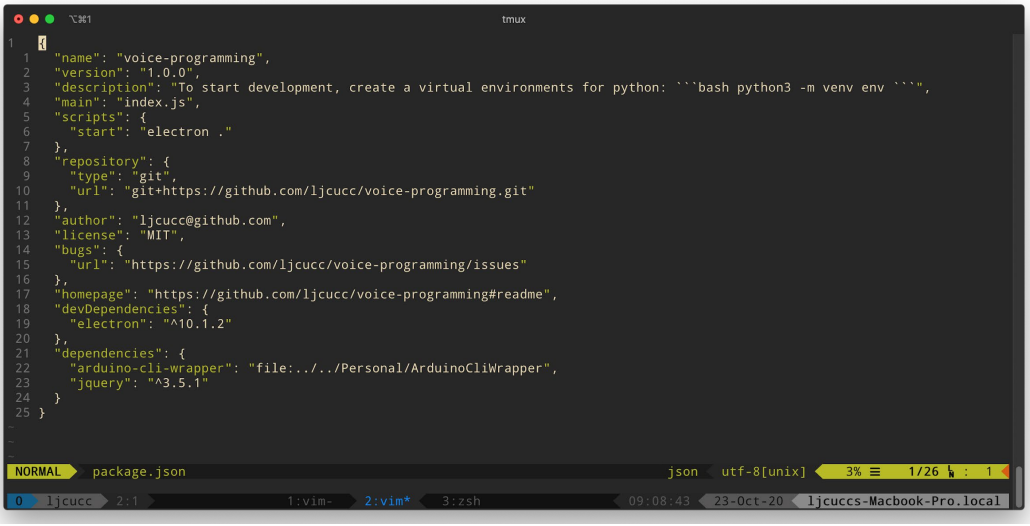


圖 18　Electron 的啟動設定檔

四、開發過程

(一)使用 git 管理程式版本

在這次的專案中我們致力於使用 git 來幫助我們管理專案，我們利用不用的 branch 去測試不同的功能。例如我們先前版本的介面就是分別於不同的 branch 做開發的。

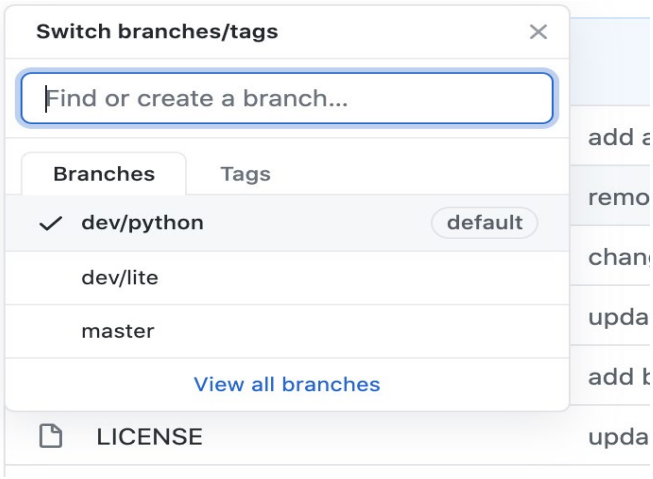


圖 19　此的 repository 中的所有 branch

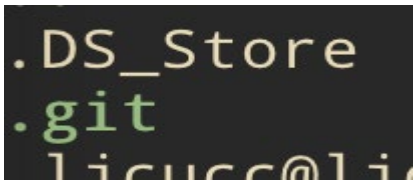


圖 20　.git 資料夾

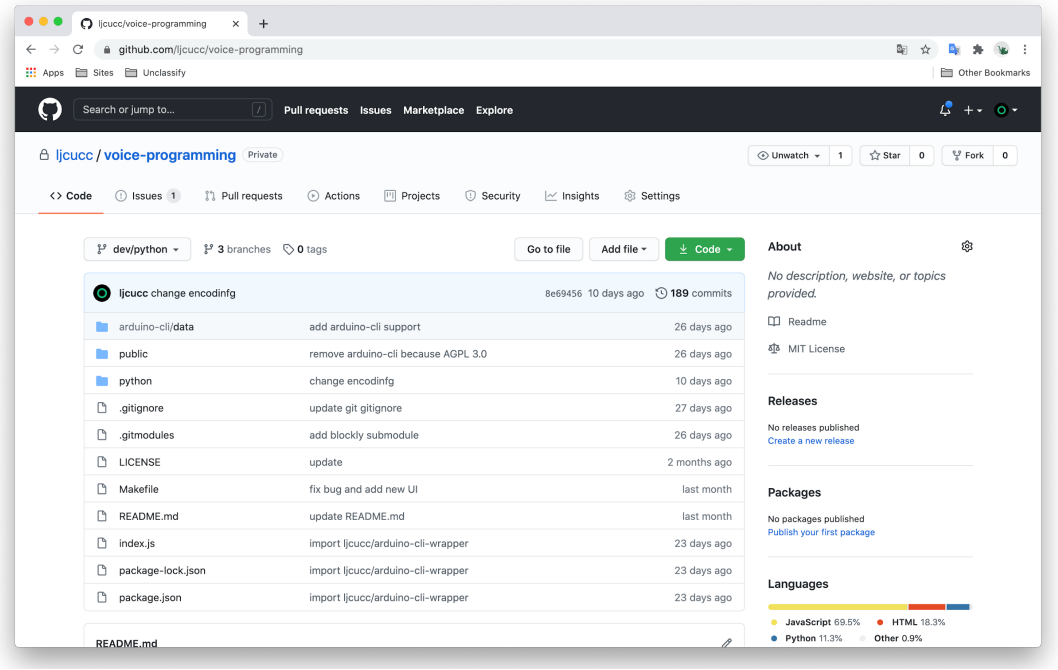


圖 21　此專案的 Github 頁面（Private）

(二)Makefile 搭配 vim 加快開發效率

使用 vim 可以加速開發，可是搭配 vim 的套件更可以讓效率加倍。為了同時使用 git 和 run python script，我們於是使用 Makefile 作為我們專案的自動化工具。在 vim 中有一個 make 指令可以幫助我們快速的執行 Makefile 而不用額外開 tmux 跳出 vim 另外執行指令並且造成動作上的負擔，除此之外我們還可以自動化繁雜的步驟（例如下圖的 make git 幫我們直接 commit、push 我們的 repository）。

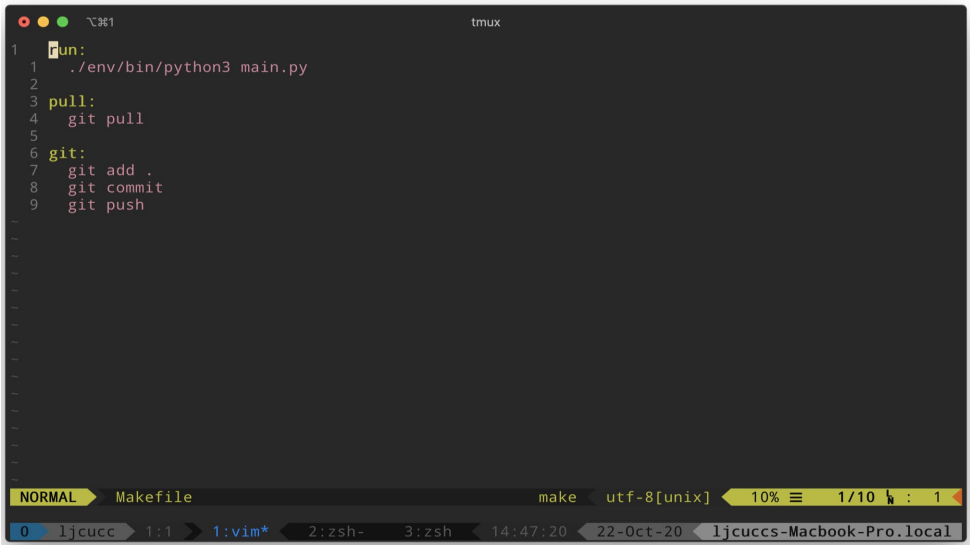


圖 22　在 vim 中開啟的 Makefile 檔案

(三)遭遇的問題的處理方法

所有我們遭遇的開發問題都會被寫進 README 中，方便接下來的 debug 和設計調整，避免出現與預期不符的功能及核心思想。

1.在 git 中安裝 Blockly

因為 Blockly 是隸屬於其他 repository 的，所以如果我們想要使用 Blockly 的話可以透過以下指令來完成：

git submodule init

git submodule update

2.Pyaudio 在 mac 上的安裝問題

因為 PyAudio 在 macOS 上會使用到一個 Objective-C 的轉譯器，所以在 macOS 上開發的時候我們都會特地另外設置一個 requirements.txt 的套件清單。安裝指令如下：

pip install pyobjc

(1)Python 在 Xcode 12 所產生的問題

因為 Xcode 12 為了配合全新的 macOS 版本—Big Sur 而把 Python 的版本從 Python 3.7 升級到 python 3.8，這造成了我們很多依賴於 Python 3.7 的程式庫（例如 PyAudio 或 dlib）無法執行。在遇到問題的起初我們先利用時光機（Time Machine）恢復到舊版本，之後經過調查發現是 Python 版本問題。最後的解決方法是用 brew 隔離安裝了 Python 3.7，使 Xcode 的升級不影響整體程式的運行。

五、上傳程式碼技術

我們為了讓對 Arduino IDE 不太熟悉的初學者可以快速的開始開發，所以我們決定使用 arduino-cli 去自動化上傳和配對步驟。

(一)Arduino CLI

Arduino CLI 是 Arduino IDE 官方為 CLI(Command Line Interface)所開發的工具集。CLI 是許多注中效率的開發者熟悉的選擇，CLI 的工具主要運行在 Terminal（或 Terminal Simulation）中。而所有 CLI 的好處就是相容性（寬容度）高，可以接受 bash 引流或是可以簡易的實 terminal 編輯器的 wrapper（例如可以寫一個簡單的腳本來實作 wrapper 並且在 vim 中觸發等等），許多 GUI 編輯器也參考了 terminal 編輯器的特性（例如在 vscode 中開發 arduino 所設定的 command path 就是設定指令的位置）。

相同的我們程式也可以使用 Arduino-CLI 相同的特性，把 CLI 工具引導到我們的程式中（這也是許多程式在擴充其程式語言之功能的方式之一，例如 ImageMagick），在 Electron 的框架中我們使用 Node.js 預設的 API——child-process 去觸發 arduino-cli（當然使用者也要預載在其電腦中，這部分如果是在教學現場老師會幫忙完成，或者我們後續會開發自動安裝檔完成安裝）。

這部分的程式（Arduino-CLI wrapper）原本是採用其他開源專案 https://www.npmjs.com/package/arduino-cli?activeTab=readme，但是因為 License 的問題最終沒有使用，在無奈找不到其他 Library 的情況下終於找到 ljcucc 以 MIT License 宣告開源的 repository。https://github.com/ljcucc/arduino-cli-wrapper

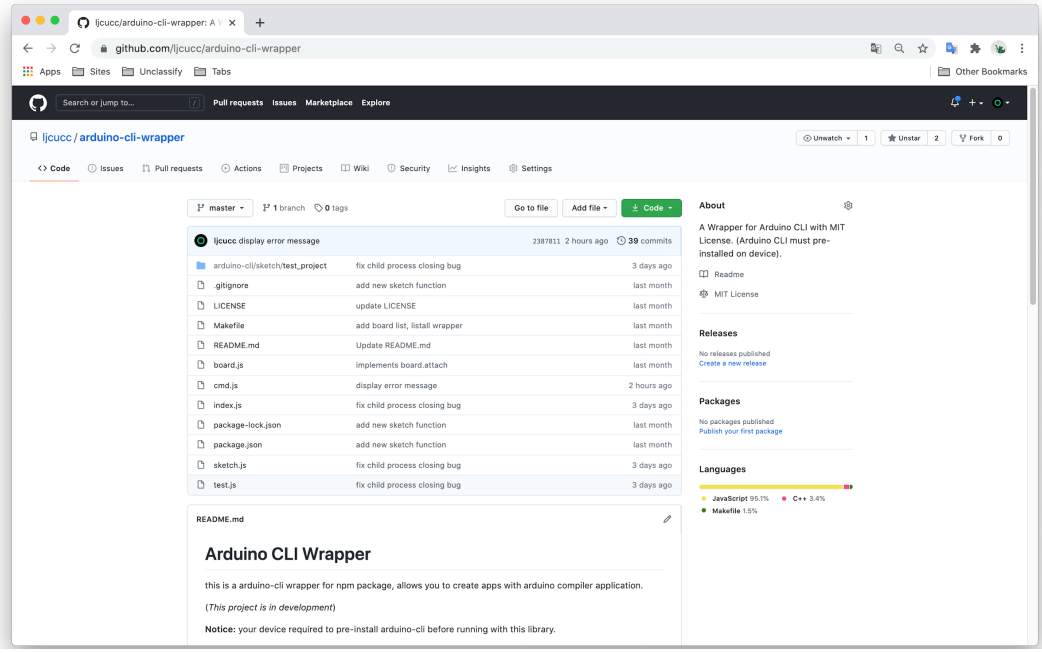


圖 23　arduino-cli-wrapper 開源之專案畫面

(二)在專案中使用 ljcucc 的 arduino-cli-wrapper

使用 npm 安裝此套件：npm install ljcucc/arduino-cli-wrapper

六、程式操作說明

(一)程式操作介面

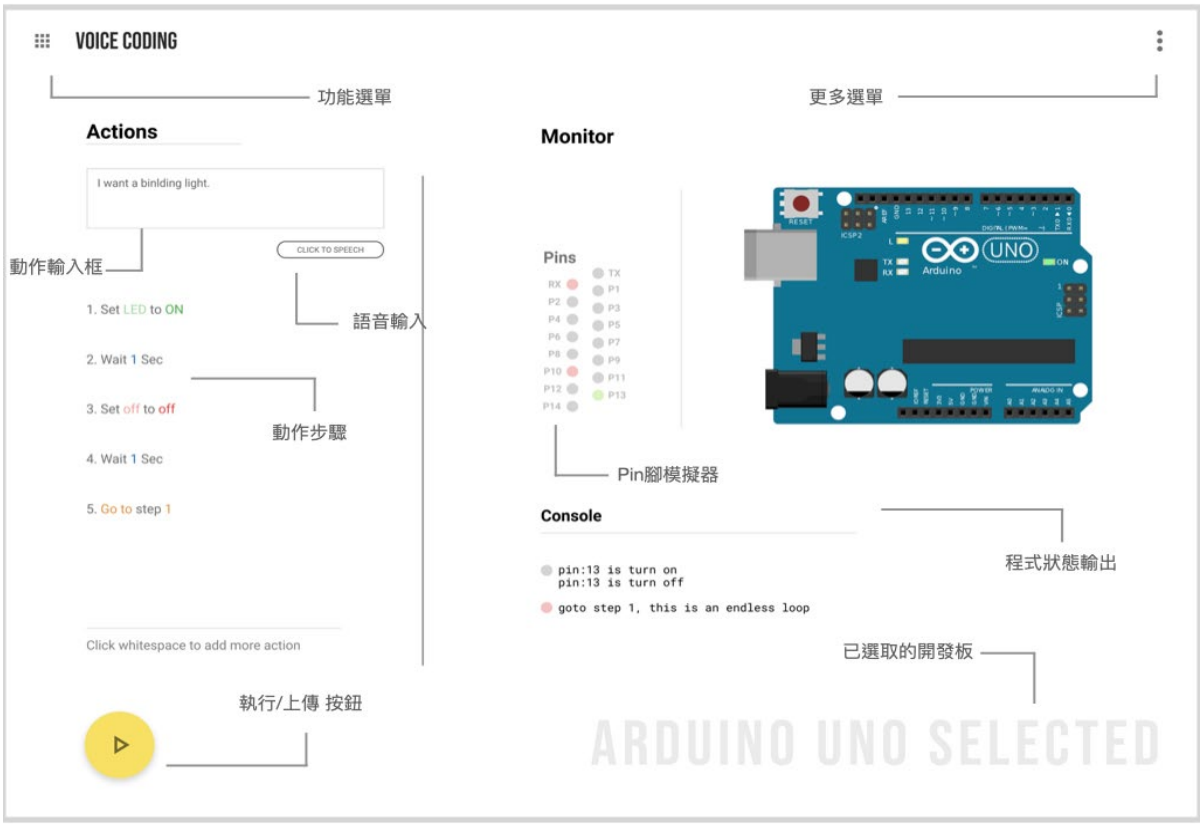


圖 24　軟體介面

Apple 的 Human Design 設計團隊在 WWDC 2015 的演講上曾經說過一句話 —— 「當我們在設計介面，我們會直接顯示 20%的功能到使用者面前，剩下的 80%都會隱藏於是使用者介面」我們也嘗試要利用這種設計理念，在經過幾次的重構後終於找出適當的介面與模式。

Voice Coding 啟動後可以分為幾個個部分：

●動作步驟程式工作區（Actions） — 讓使用者進行程式最後編輯的地方。

●監控顯示器（Monitor） — 進一步讓使用者了解程式的運作原理。

在 Action 中所設定的所有指令都會顯示在《動作步驟》的編輯區中，《動作步驟》的概念就有點像是食譜，告訴使用者程式會如何執行著。

(二)創建自定義積木的詳細使用步驟

創建自定義積木的使用者操作如下：

1.點選《click to speech》，聽到提示音後開始說話，聽到另一個提示音後結束。或者直接在空欄中填寫您想要的程式（口述語言）。

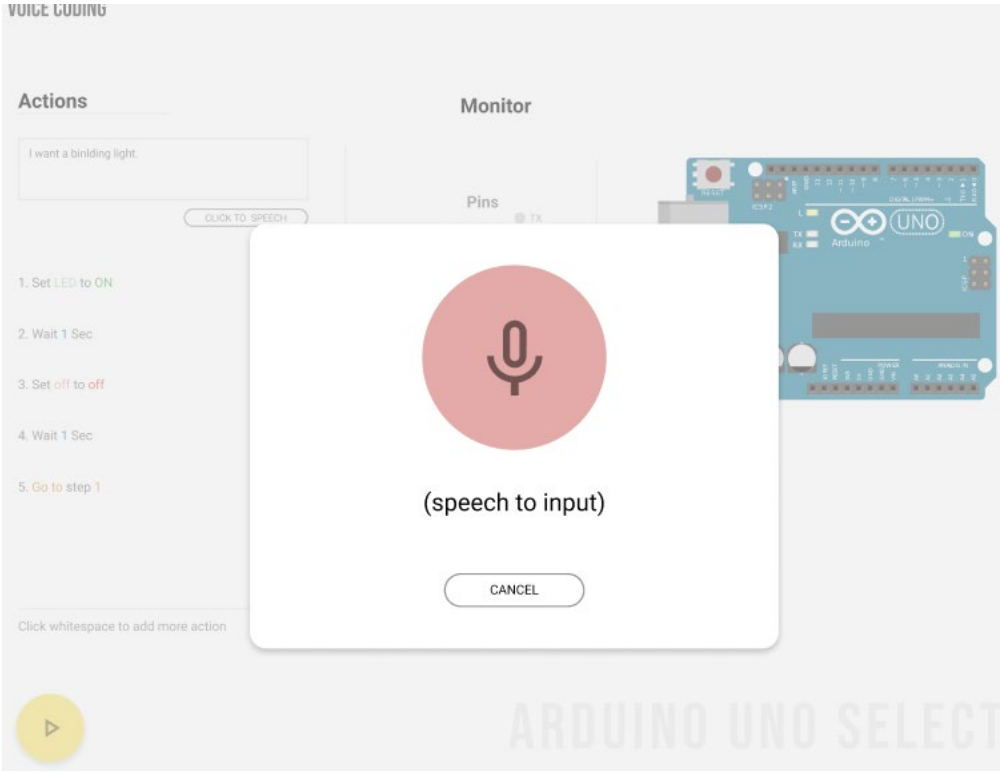


圖 25　語音輸入

如果使用者要使用語音輸入，需要設定麥克風來源。請點選在三點選單中點選《Select external sound input》，即會顯示音源驅動清單。請在其中選擇正確的麥克風再開始語音輸入。（如下圖）

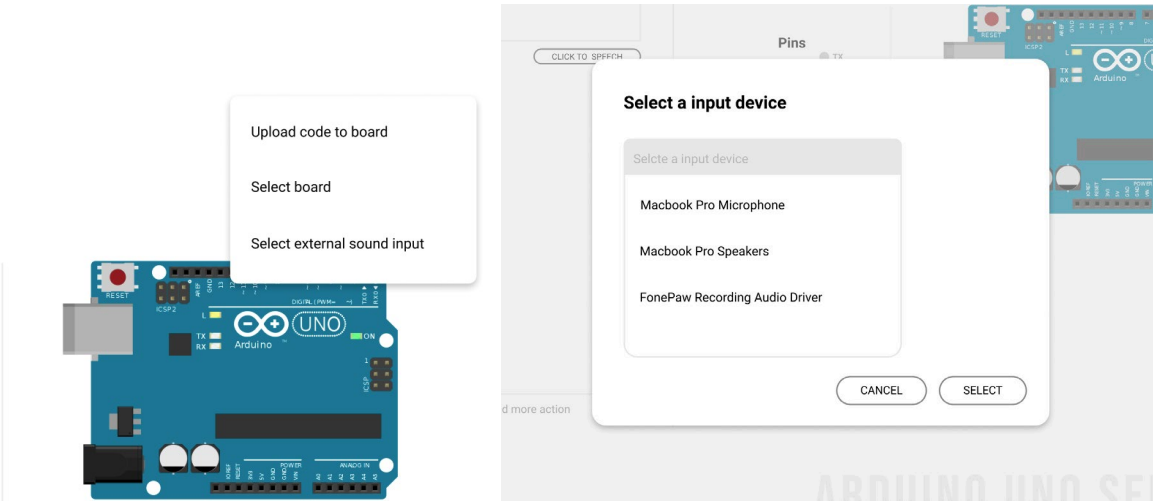


圖 26　麥克風選單圖示，滑鼠滑過上方即會顯示提示標題

2.接著如果使用者是使用語音輸入，當結束語音提示響起，使用者所說的話（經由文字轉語音的結果）將會顯示在空欄中。

3.如果是使用者手動輸入於空欄中，推薦結果將會即時的顯示在下方的清單中，如果未找到或無法理解相關結果，將會顯示「未找到相關程式碼」。

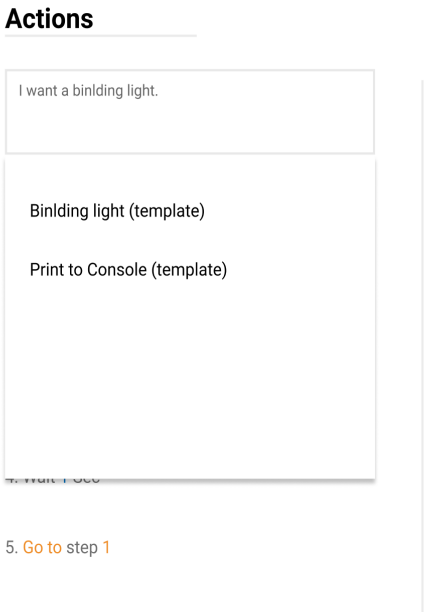


圖 27　程式會推薦使用者幾個「方案動作選項」，選擇其選項

4.選擇選項後提示視窗會消失，而你所選擇的方案則會在下方的 Action 列表中顯示。

5.接下來如果您所選動作模板是需要設定的，程式將會跳出填寫表單，完成進一步的敘述，上方會保留著你當初敘述的內容

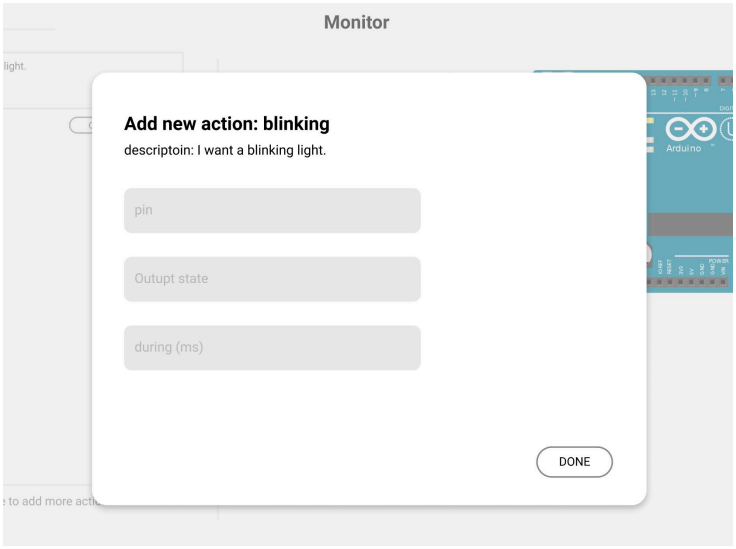


圖 28　設定方案

6.如果你想要在後續設定相同的動作步驟，可以在動作編輯區點選步驟，即可編輯。

7.在編輯各個項目時，程式會依據你的敘述來推薦相關的選項（圖 29）。

圖 29　程式會依據敘述推薦相關的選項

(三)上傳 Blockly 程式碼至 Arduino

當你寫完程式後，就需要上傳程式碼至 Arduino。在這裡請記得如果您還沒安裝

arduino-cli 於您的電腦請先安裝，目前有實際測試過的系統為 macOS、Ubuntu。

1.首先請選擇您插入電腦連結的 Arduino 板子，在右上角的三點圖示中點選「開發板」圖選項。

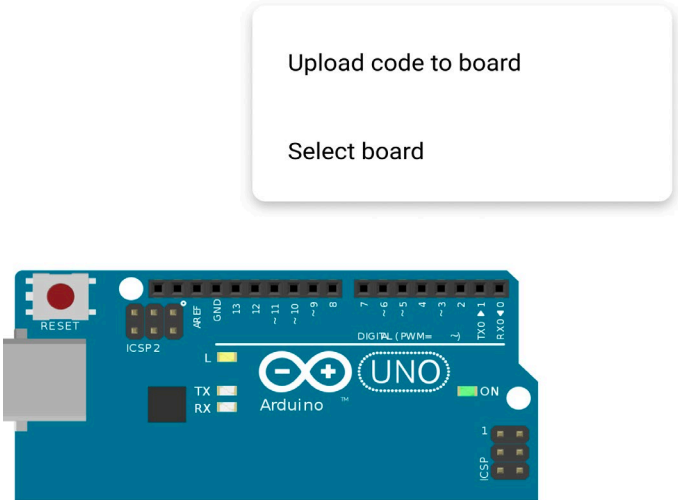
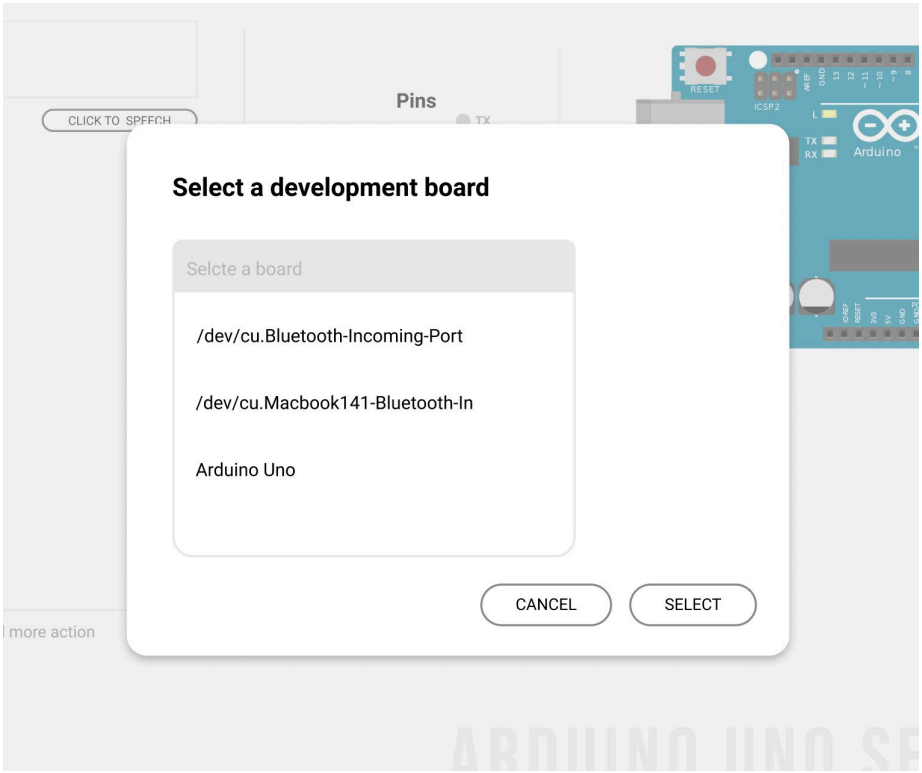


圖 30　選單介面

2.點選後會跳出選擇清單，如果程式無法辨識裝置會顯示/dev/*的系統位置，如果能辨識則會顯示裝置名稱。



31　顯示裝置名稱

3.接下來您就可以按下「播放鍵圖示按鈕」上傳您的 Arduino 程式。

注意：這個動作會複寫您原本在 Arduino 中的所有程式與資料（包括 MCU 內建的 EEPROM）。

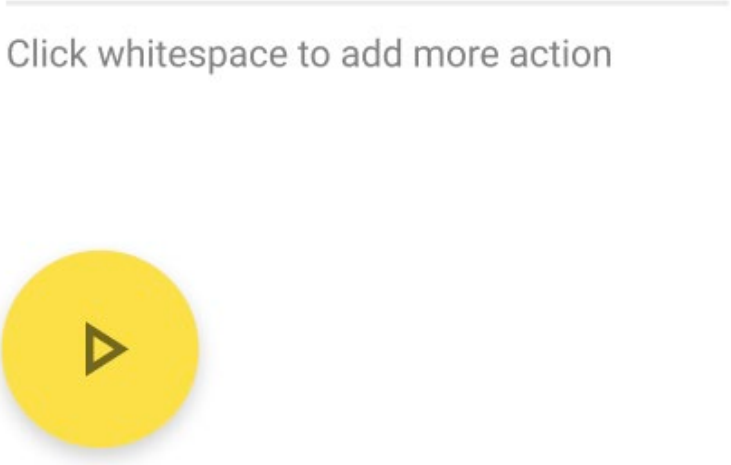


圖 32　上傳您的 Arduino 程式

(四)上傳程式碼

如果您希望可以進階的更改程式碼，我們可以使用「上傳按鈕(播放鍵)」來直接取得上傳至 Arduino 的原始碼。

點選「上傳按鈕(播放鍵)」後程式碼就會顯示在下方。在程式碼上點選程式碼並且按下複製（Copy）按鈕來複製，如果您的電腦有安裝 arduino-cli，也可以按上傳（upload）直接上傳您的程式碼。

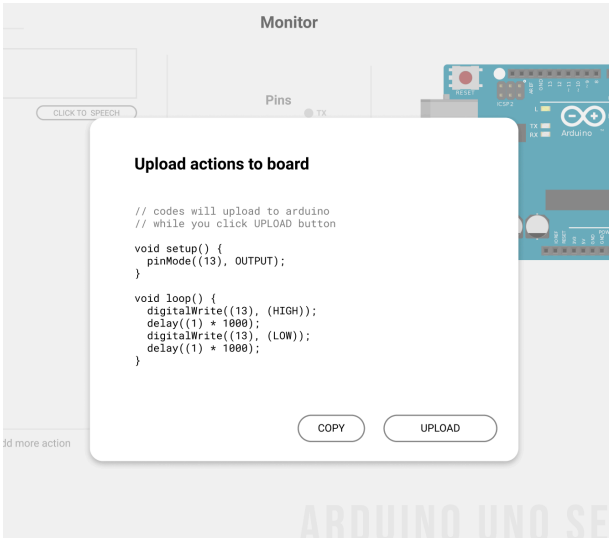


圖 33　上傳或複製程式碼

2.複製完程式碼後，打開 Arduino 應用程式，您可以在 Lunchpad（Mac）、應用程式選單、或者開始列中的程式集中（Windows）找到 Arduino 圖示（如果您沒有安裝 Arduino IDE，可以依照這個網址安裝：https://www.arduino.cc/en/software，如果您是使用 Linux 且您的電腦中已安裝 snapstore，例如 Ubuntu，請到 snapstore 或 Ubuntu Store 中搜尋 Arduino IDE 安裝：https://snapcraft.io/arduino）。

(1)首先把程式碼貼在程式編輯區中。

(2)接著按下上傳按鈕。

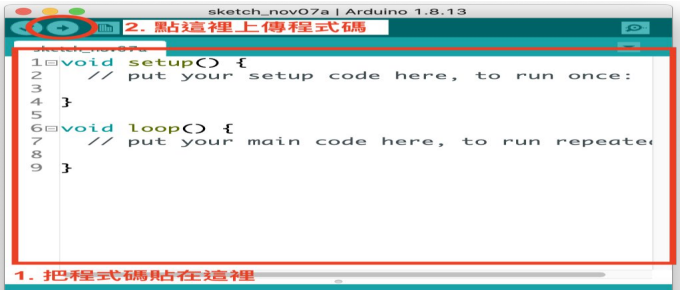
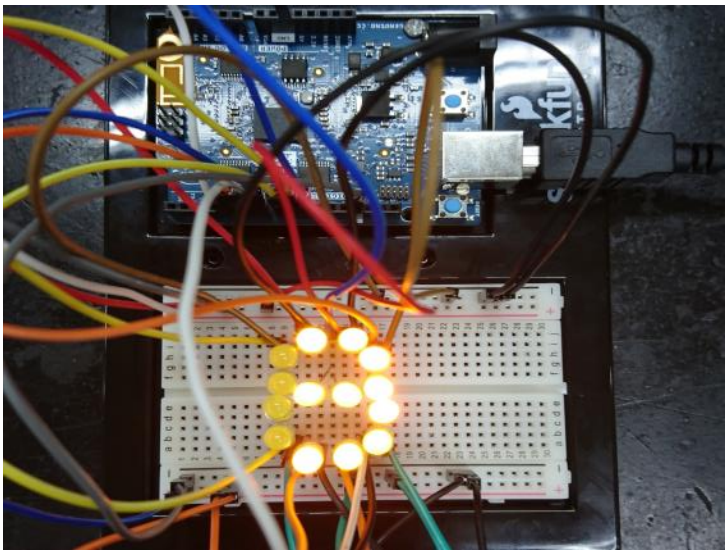


圖 34　上傳程式到開發板

(3)如果您有 Touchbar 點選這個按鈕上傳：

35　Touchbar

(4)接下來您就會看到您的 Arduino 會依照您先前所設定的指示執行。



36　程式執行結果

七、學生上課實際操作軟體後問卷調查

為了印證我們的程式對於程式邏輯教育的幫助，我們實際讓一個班的學生(41 人)來

使用本軟體，並在結束之後，請參與的同學幫忙填寫意見回饋，如此一來便有以下的數據：

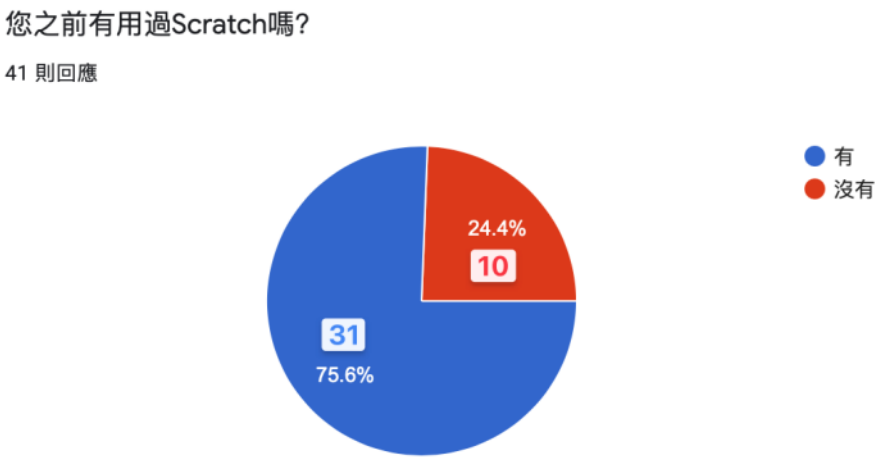
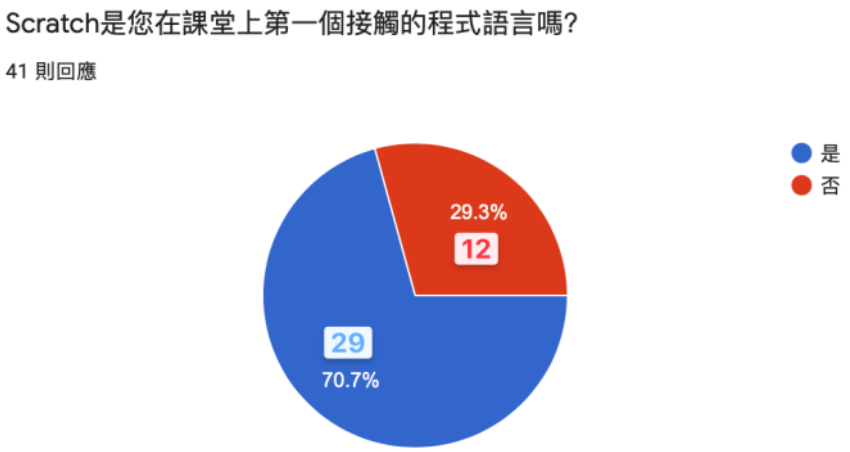


圖 37　問題 1 問卷結果

第一個問題，我們調查了 41 位學生在此之前有多少人使用過 Scratch，數據表示有 31 人在本次之前使用過 Scratch，此外，有 10 人沒有使用過 Scratch。



38　問題 2 問卷結果

接下來我們調查了，Scratch 是否是學生在課堂上觸及到的第一種程式語言，數據顯示：對於全班 29 人來說，Scratch 是課堂上第一次接觸到的程式設計軟體，另外 12 人則是接觸到如 Mblock 等大同小異的程式設計軟體。

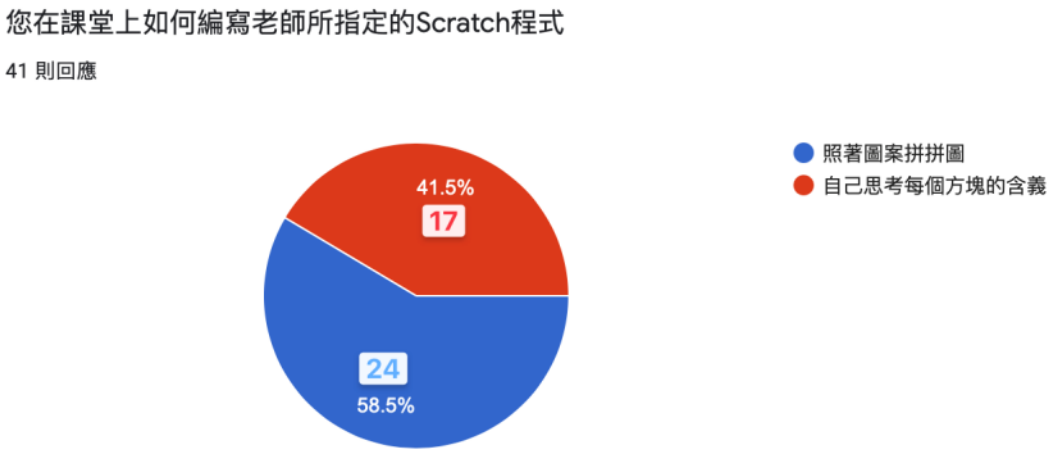
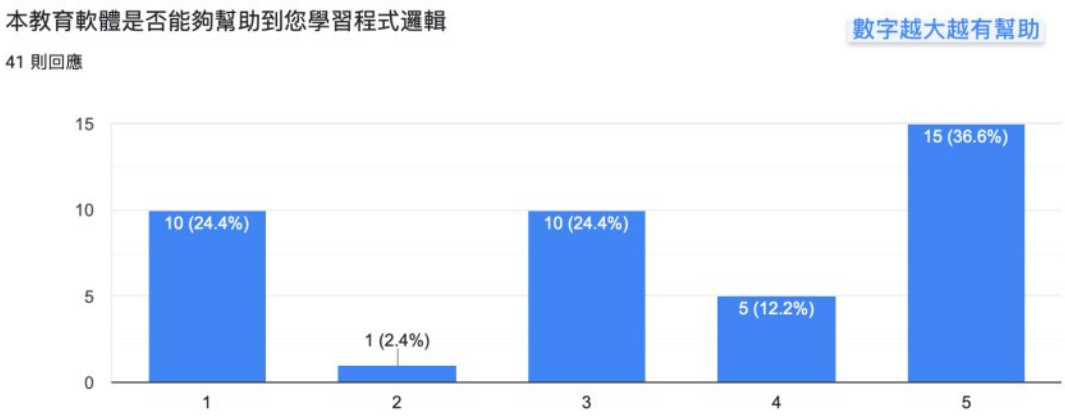


圖 39　問題 3 問卷結果

我們詢問了這些操作邏輯近似 Scratch 的軟體，同學在課堂時是如何完成老師安排的進度的，數據結果表示，有 58.5%的學生，都是在玩拼圖遊戲，而沒有思考這些 block 的具體作用。



40　問題 4 問卷結果

以上的數據顯示，大多數的學生都喜歡利用口述的方式來進行程式邏輯的學習，不過也有 26.8%(得分 1、2)的人認為對於自身的程式能力提升效益不大。

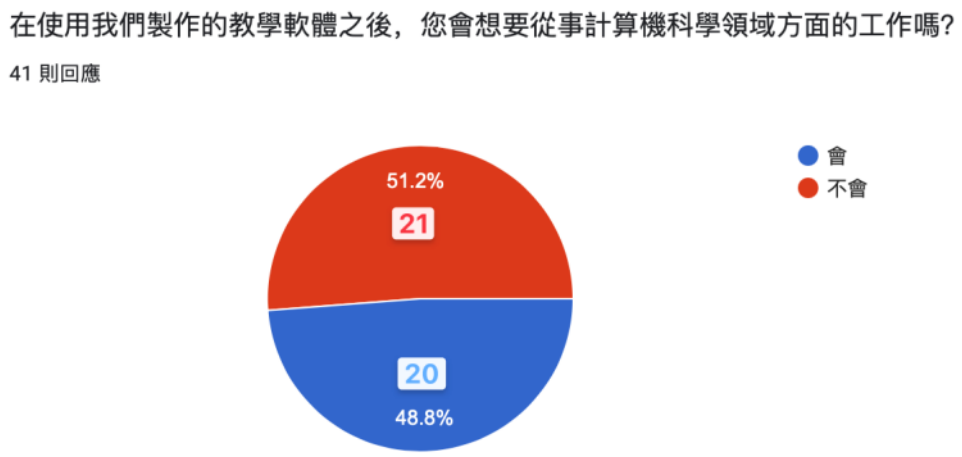
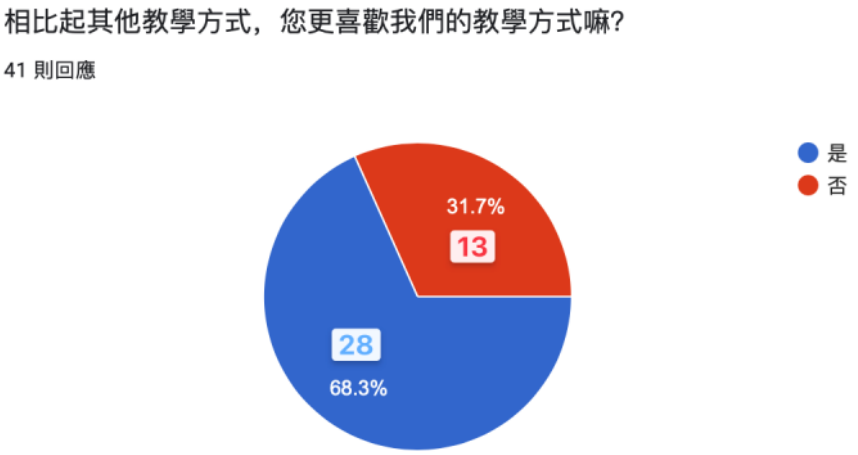


圖 41　問題 5 問卷結果

儘管是資訊科，但是對於半數的人來說，在未來規劃，其實沒有想要成為工程師，繼續寫程式。所以我們認為，程式設計課程的意義在於培養學生的處理事情的邏輯，而不是培養寫程式的能力。



42　問題 6 問卷結果

數據顯示：對於大多數人，我們的教學方式明顯比 Scretch 而言，更加有趣及易懂。

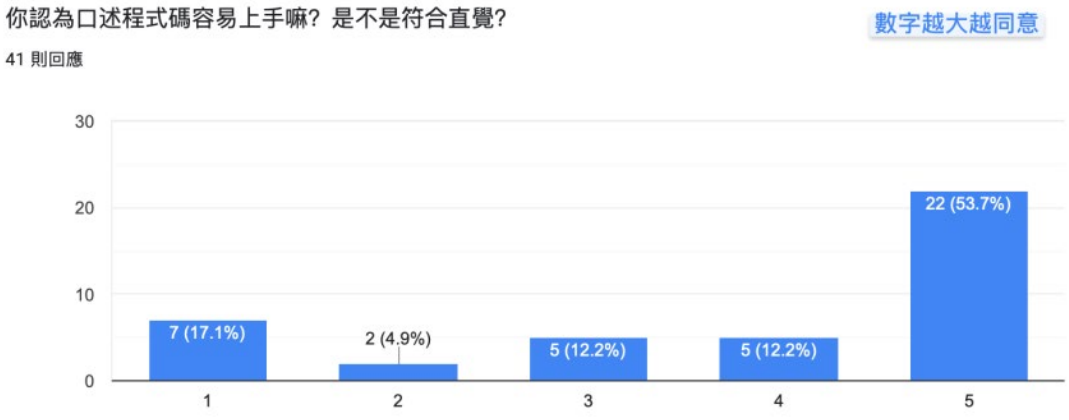


圖 43　問題 7 問卷結果

最後，67.9%(得分 4、5)的人非常同意本程式開發計畫所陳述的教學方式，非常的符合直覺，能讓人幾乎無障礙的編寫程式，且透過逆向思考來學習每一句程式碼所代表的意義。

伍、　研究結果

目前我們的研究僅限於產生 Arduino 程式，因為其為目前學校最常見的硬體電路，經由實驗，我們能夠透過口說或是像寫作一樣描述我們所要實現的事件，在告訴電腦我們想要做甚麼之後，電腦便會自動產生推薦的下一步動作供我們選擇，一步步引導我們完成整個事件的描述，最終可以自動將程式傳進 Arduino IDE 或是讓使用者自行複製程式到 Arduino IDE 做後續的修改動作，如此一來將可以縮短程式寫作的時間，同時對於初學者來說，更不需要寫程式，只需要透過符合邏輯化的敘述便能夠設計其所要實現的電路，可說是相當方便，對於已熟悉程式的使用者來說，亦可以縮短其開發時間，只需要做點程式修正即可；且透過問卷我們可以發現，我們自行開發的程式軟體，確實適合用在教學上，同時，也可以輔助學生學習程式設計。

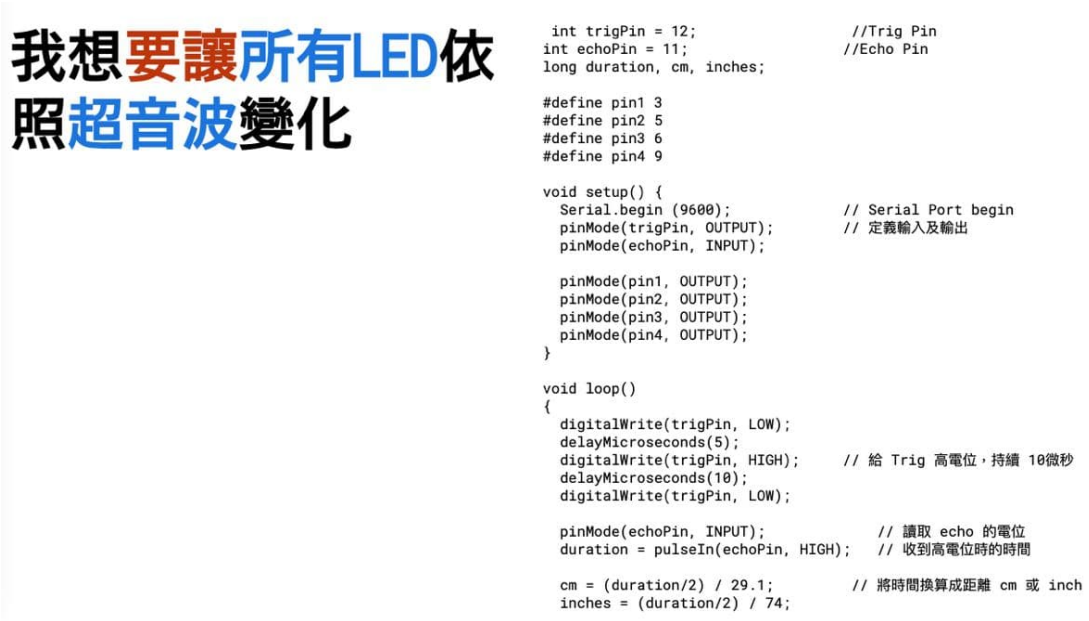


圖 44　語音程式碼產生器程式生成畫面

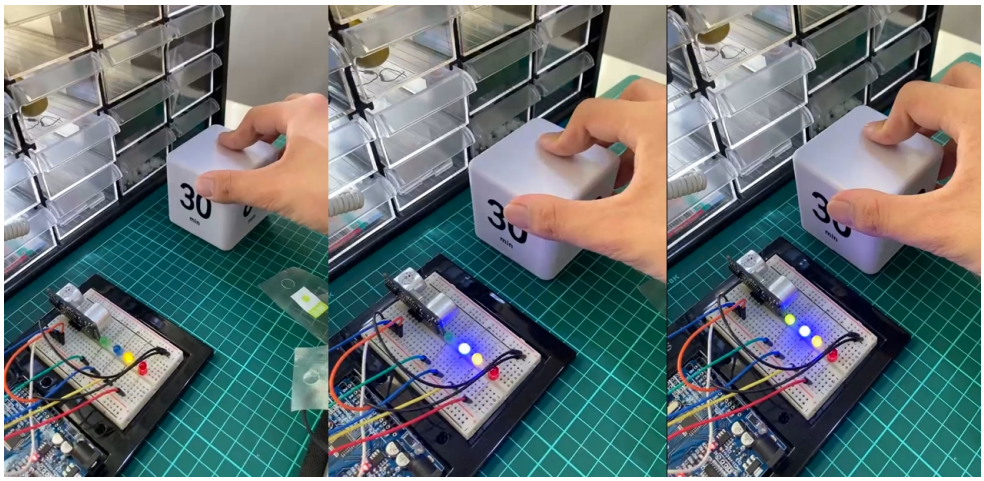


圖 45　程式上傳後 arduino uno 執行結果

陸、　討論

(一)這樣的設計是否符合程式教育的精神？

目前很多人誤以為程式教育就是要學寫程式，於是造成很多學生的排斥，認為以後沒有要從事該行業，為何要寫程式，我們的設計便是希望學生可以經由此工具訓練其邏輯化的口說能力，將欲實踐的事件表達清楚，如此一來，日後不管從事任何領域的工作，都能夠準確有效的與人溝通。

(二)學生透過此工具的學習是否能夠對程式產生興趣？

目前我們先開發出此程式，當然還在優化中，但我們實際找一個班級學生來驗證此軟體是否能夠對學生學習程式有實質的幫助，經由問卷結果，確實對學生有幫助。

(三)軟體是否能夠支援函式庫自動匯入，例如：講到超音波時能夠自動匯入超音波的函式庫？

目前這些功能正在優化中，期待能夠加入幾個學校裡面常見的感測器函式庫，方便學校教學使用。

柒、 結論

自然語言結合推薦算法開發程式碼生成器之研究，目前開發進度已是可試用階段，我們也期待能夠讓更多人測試，如此將能讓這個研究計畫更加完善，我們希望除了軟體的完整性外，也期待能夠用社會學的角度來探討這樣的研究方法是否能夠達到訓練其邏輯概念的目的，同時對於初學程式者來說，是否能夠提高其學習成就，待有一定的邏輯觀念後，再進入拼圖式的程式設計軟體學習，這個研究能夠提供程式設計另一種開發方式，當我們要解決一件事情時，可以用口語的方式告訴電腦，由電腦自動生成一個程式來解決使用者當前遇到的問題，如此一來，將能夠協助很多人解決一些問題，所以此研究的概念相當的創新，我們也非常熱衷在此計畫的研究，希望有朝一日，能夠在各個校園看見學生使用此軟體來輔助學習程式。

捌、 參考資料及其他

(一) 劉漢偉(2019)。Vue.js 從入門到項目實戰。清華大學出版社。

(二) 史蒂文·金尼，譯：徐曙光(2019)。Electron 跨平台開發實戰。清華大學出版社。

(三) Shelley Powers，譯：楊尊一(2016)。Node 學習手冊第二版。歐萊禮。

(四) David Flanagan(2012)。JavaScript 大全(第六版)。歐萊禮。

(五) 勁樺科技(2017)。入門首選 Python 程式設計。台北市：台科大圖書股份有限公司。

【評語】052506

此作品想用口述的方式且使用語言辨認的技術來協助產生 scratch 程式，具有創意且有實作。對於從語音自動轉成程式的細節可以有更詳盡的說明。建議要在作品書中詳細敘述整個研究流程，並提供實際應用的範例說明。作品書有提供一範例呈現在 LED 上顯示 3 的結果，建議應該更進一步說明清楚如此做對邏輯實驗的實際幫助。

作品簡報

自然語言結合推薦算法開發程式碼生成器之研究

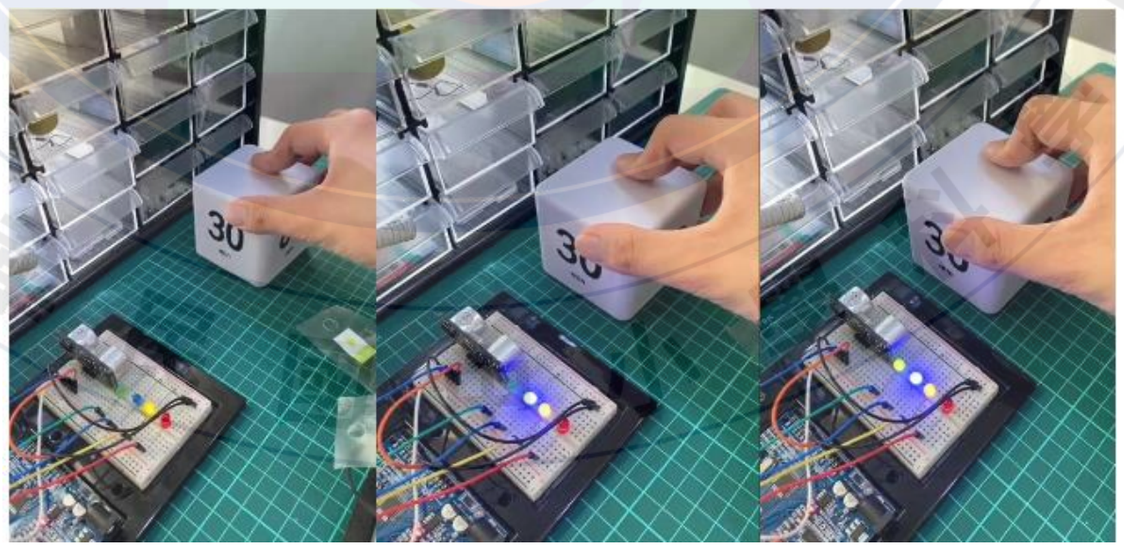
編號：
組別：高級中等學校組
科別：電腦與資訊科學

前言

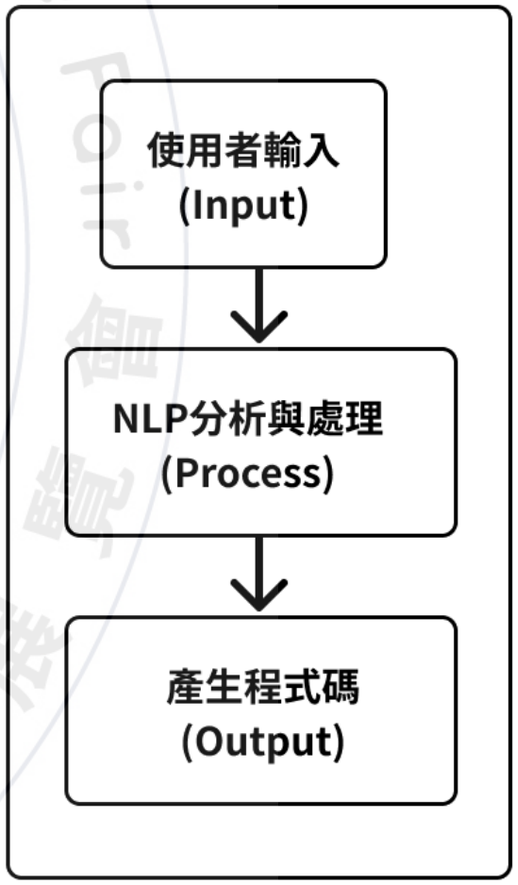
- **摘要**
  - 我們自行開發一套軟體系統，可經由口語的方式與電腦溝通， 同時電腦經過關鍵詞的辨識後，能更精準的透過推薦演算推出幾種可能的目標事件，透過選擇的方式，一步步引導使用者完成其想要達到的動作，最終目標是透過這個軟體自動產生程式碼。
  - 透過針對自然語言的分析來產生C/C++程式碼，利用說話邏輯來培養學生的表達能力。超過64%的同學認為，對於程式學習更感興趣，也提升他們撰寫程式的成功率。

- **研究目的：**
  1. 了解拼圖式的程式學習軟體
  2. 了解語音辨識的原理
  3. 了解推薦算法
  4. 了解 Python 的撰寫
  5. 了解 Electron Framework
  6. 了解 Arduino CLI

- **整個作品的研究內容可以分為4大類：**
  - 使用者輸入
  - NLP分析與處理
  - 產生程式碼
  - 程式實作



程式實作 (Implementation)

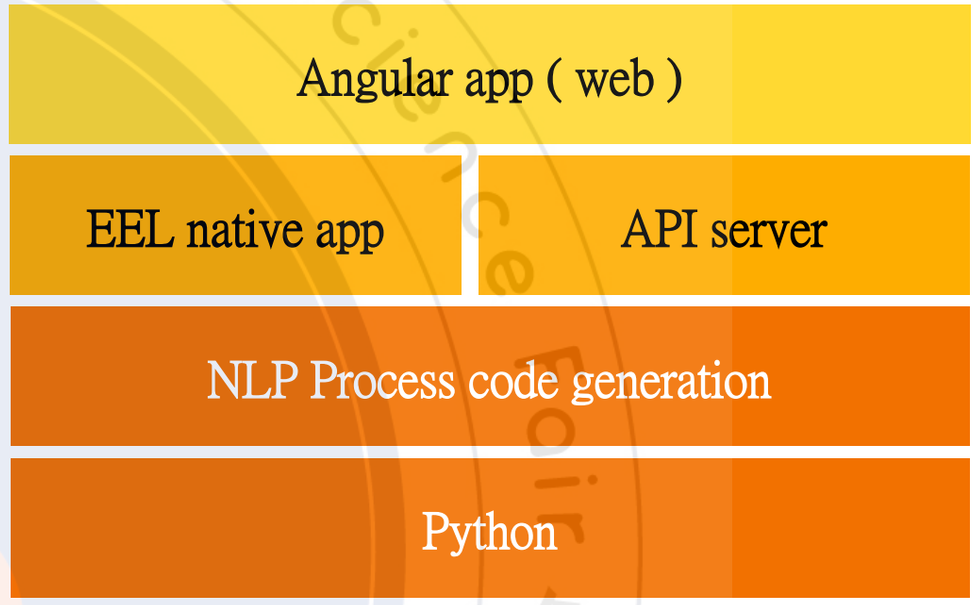


註：圖為研究流程圖

研究方法

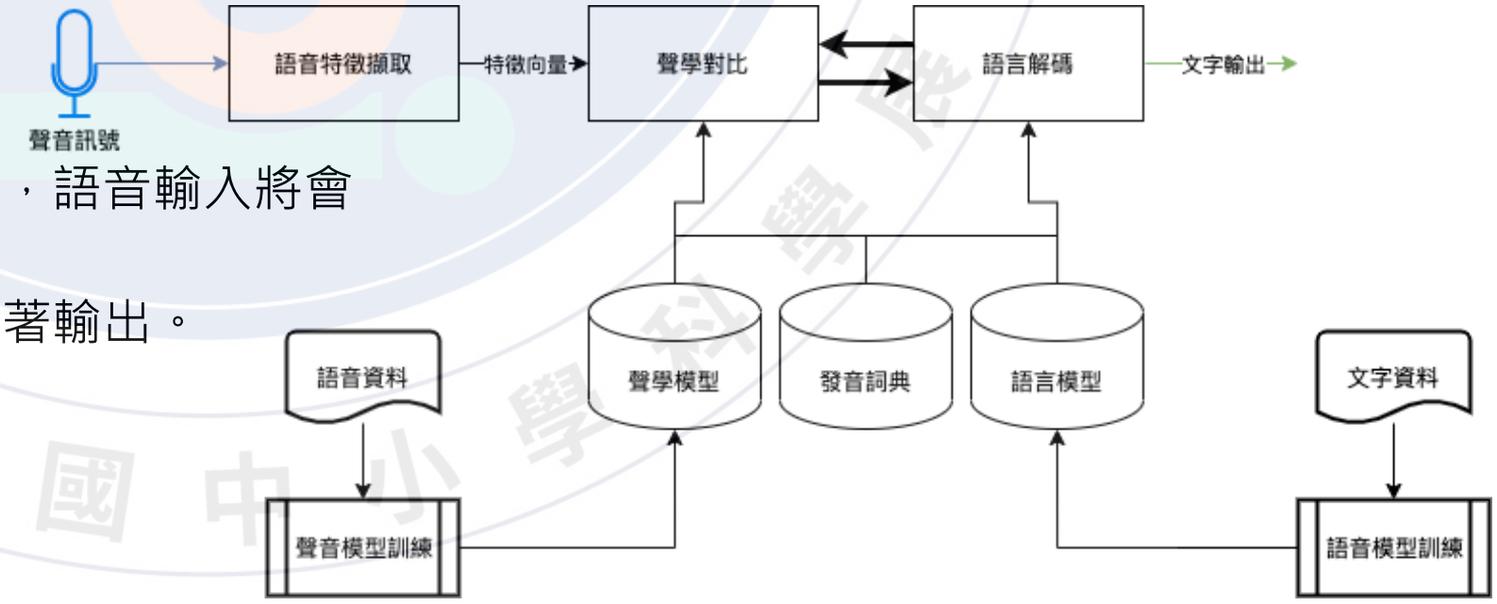
自然語言的輸入 (Input)

- **輸入方式：**
  - 目前採用語音轉文字方案 (Speech to text solution)。
  - 替代方案使用Safari或Chrome/Chromium內建的Speech API作為語音輸入的方式。
- **實作方式**
  - 使用者的輸入會顯示在前端介面 Angular + Electron / Chromium (EEL) 的地方做處理，接著利用Python Flask作為API端口作為輸入。
  - 後端程式Python (Flask 當作 API端口)接收到資料後就會進行下一個步驟。



註：上圖為整體程式架構與運作關係

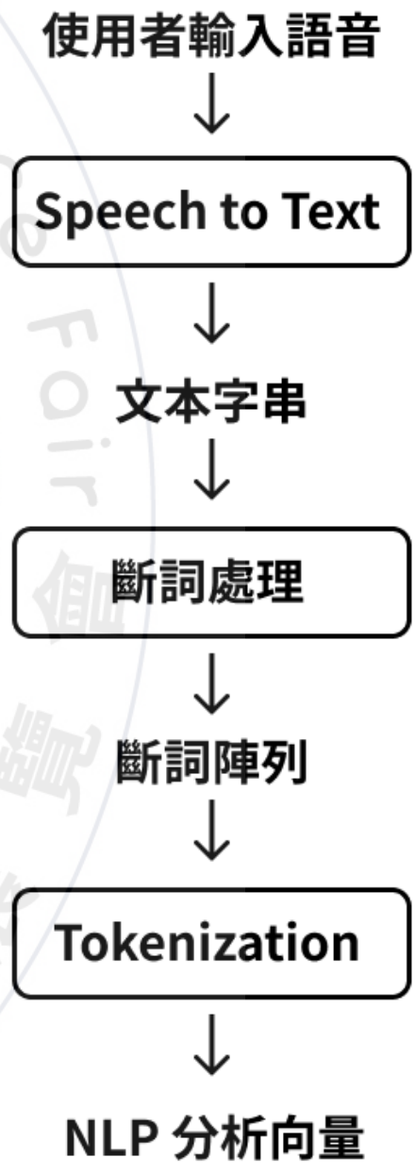
- **使用方式**
  - 當使用者點擊「語音輸入」的按鈕後，語音輸入將會被開啟，使用者即可輸入語音。
  - 輸入完後訊息就會送到後端處理，接著輸出。



研究方法

自然語言斷詞原理

- 我們使用jieba所提供的自然語言斷詞作為斷詞工具。
- **為什麼要使用jieba？**
  - 因為jieba提供了大量的中文斷句資料庫。和英文不一樣的是，我們中文不是以字作為詞的單位（也就是以空白作為每個字的分割，所以英文沒有必要做斷詞）。
- **Jieba的斷詞模式**
  - Jieba的斷詞模式共分為：精確、全斷詞、搜尋模式，主要是利用規則斷詞和統計斷詞的方式實現中文斷詞原理。
  - 在這次的研究中，我們使用jieba的原因是因為他的詞庫廣泛，所以我選擇用全斷詞來取得最原始的斷詞資料。
- **如何讓程式去理解輸入語句？**
  - 斷詞處理的本質就是把一串字串利用現有的自然語言語法規則轉換為多個不定長度的語詞向量。
- **什麼是NLP 分析向量？**
  - NLP 分析向量是一組由自然語言所產生的分析資料，用於產生程式碼之間的語法分析部分以利於分析語句。

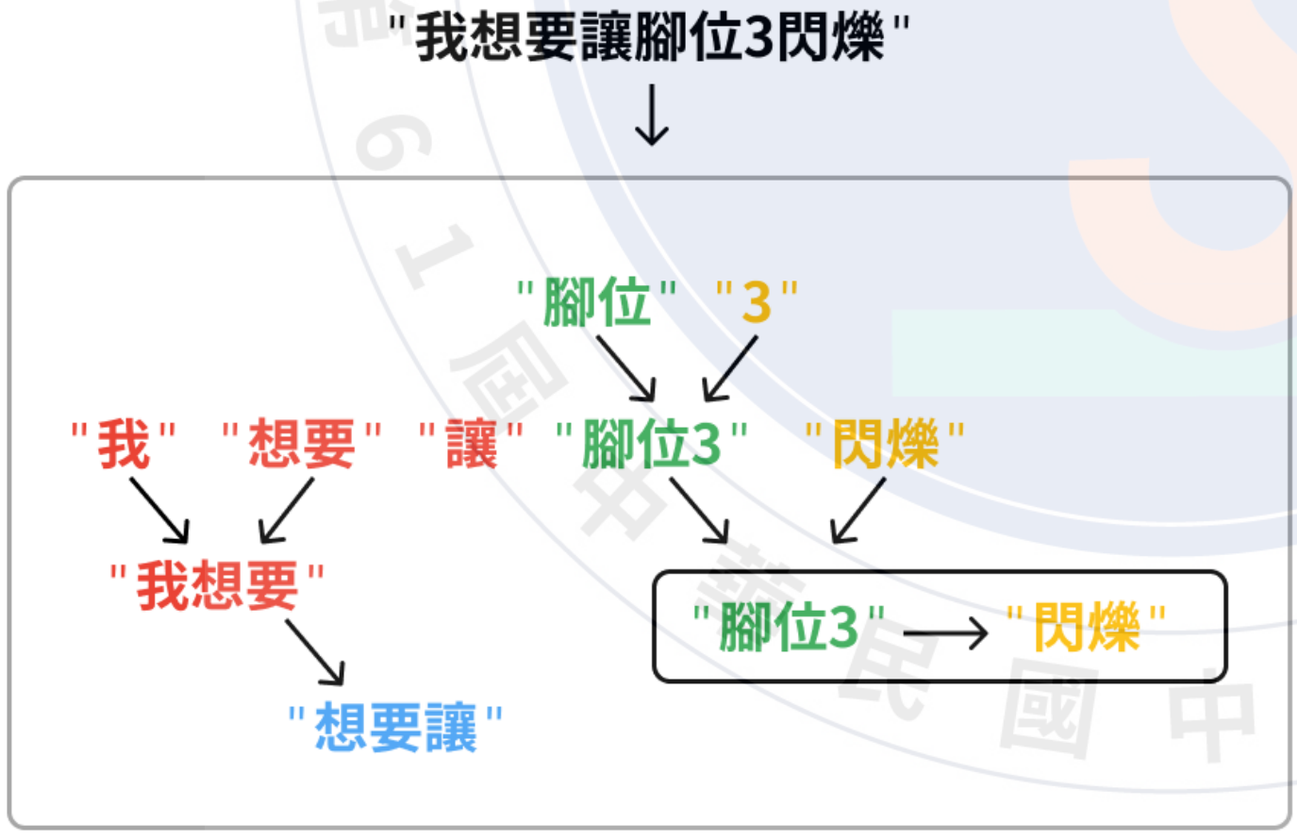


研究方法

語意分析與處理原理

- **如何讓程式去理解輸入語句？**
  - 我們利用jieba和推薦演算法來實現理解學生的語句（如右圖所示之原理）。

1. 利用jieba做斷句分詞處理。
2. 利用交叉比對驗證結果（如下圖圖所示）最終產生一個型別標示向量。
3. 藉由向量在資料庫裡做推薦排序取得最終向量。



$$n = size\ of\ result$$

$$Lost_T = c \cdot (size\ of\ N) \cdot (size\ of\ str) + c \cdot O(n \log n)$$

上圖為推薦演算法損失延遲之算式

研究方法

程式碼產生原理 (Output)

- **什麼是AST（Abstract syntax tree）？**
  - AST - 抽象描述語法樹，一種用於描述程式結構的一種樹狀結構。

- **轉換NLP 分析向量為AST**
  - 當語意分析處理完後，就會生成「NLP 分析向量」，在這個步驟程式就會生成AST，為後續程式生成做準備。
  - 當程式接收到NLP就會依照斷詞結構在資料庫中尋找適當的轉換關鍵字，有如斷詞原理和推薦演算法的變形（如圖三）。
  - 接著程式會尋找一個AST object可能的關鍵字（如圖一，為AST 資料結構轉換資料庫）。
  - 接著再利用遞迴結構解析（parse）AST。
  - 程式碼產生器（Code generator）會利用AST進行程式碼產生（如圖二）

圖一

```
def merge_target_ref(ast_list):

    def sentance_list(acc, index, lis):
        if(index == len(lis)):
            return acc

        cur = lis[index]
        cur = sentance(acc=[], index=0, lis=cur)
        acc.append(cur)

        return sentance_list(acc, index+1, lis)

    def sentance(acc, index, lis):
        if(index == len(lis)):
            return acc

        cur = lis[index]

        if(len(acc) > 0 and
```

圖二

```
suggestion:
    undefined:
        type: undefined
        title: 未找到相關程式區塊
    set_pin_state:
        title: 設定腳位狀態
        description: 當程式開始時，讓電腦知道哪一個腳位要做什麼動作，例如讓腳位「3」「輸出」。
        type: set_pin_state
        fields:
            pin:
                title: 腳位編號
                type: select
                data: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1,A2,A3,A4,A5
            state:
                type: select
                title: 腳位類型
                data: 輸出, 輸入
        requirements:
            - arduino
    set_pin_output:
        title: 設定腳位輸出
        description: 設定某個腳位的電位輸出，共有兩種狀態— 高電位、低電位。
        type: set_pin_output
        fields:
            pin:
                title: 腳位編號
                type: select
                data: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1,A2,A3,A4,A5
            state:
                title: 輸出狀態
                type: select
                data: 高電位, 低電位
        requirements:
            - arduino
```

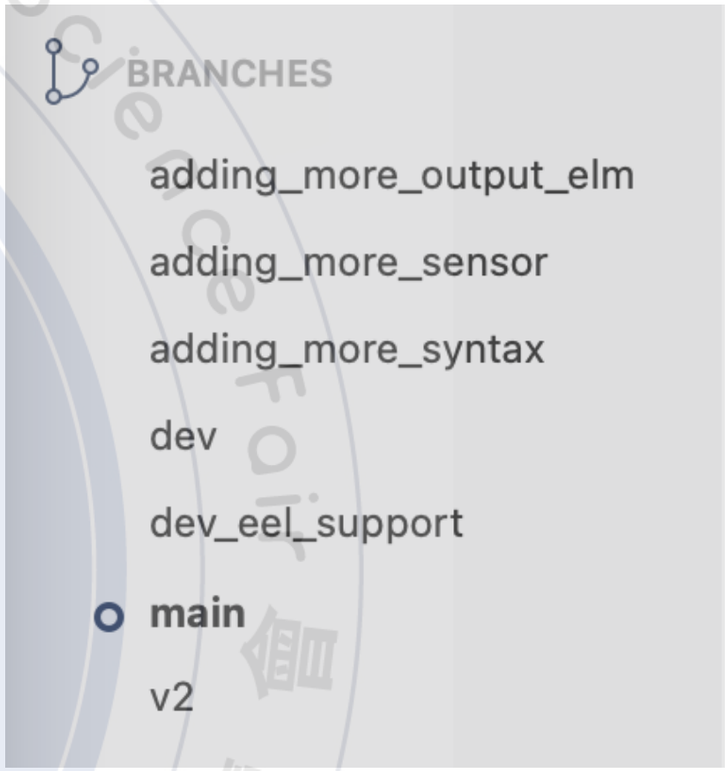
圖三

```
狀態,腳位,設定,設為,按鈕,輸出,輸入:set_pin_state
輸出,設定,設為,LED,腳位,燈,高電位,低電位:set_pin_output
取得,獲得,讀取,輸入,按鈕,腳位:get_pin_input
閃爍,腳位:set_pin_blink
變數,設定,設為:set_var
新增,變數:new_var
重複,執行,迴圈,幾次,當:loop
```

研究方法

程式實作與開發 (Implementation)

- **研究方向**
  - 使用 Python+Angular(web)開發。
    - 在這裏我們為了方便開發和管理，我們選擇使用Angualr作為我們開發前端介面的開發工具。
    - 在Python 後端方面我們也選擇使用Flask作為我們的開發框架。
  - 使用git管理版本。
    - 在第一次分區賽後，因為程式的結構過於複雜，再加上新的程式功能要求下，我們決定採用branch開發策略，增加功能開發的穩定性。
  - 配合使用arduino-cli來自動化上傳程式（但考慮到學生的操作後已移除）。
  - 在開發過程中，我們也同時學習如何使用vim (key-bind) 開發，透過使用vim 的 key-bind來加快開發效率，讓我們更容易把重心擺在試驗及研究上。



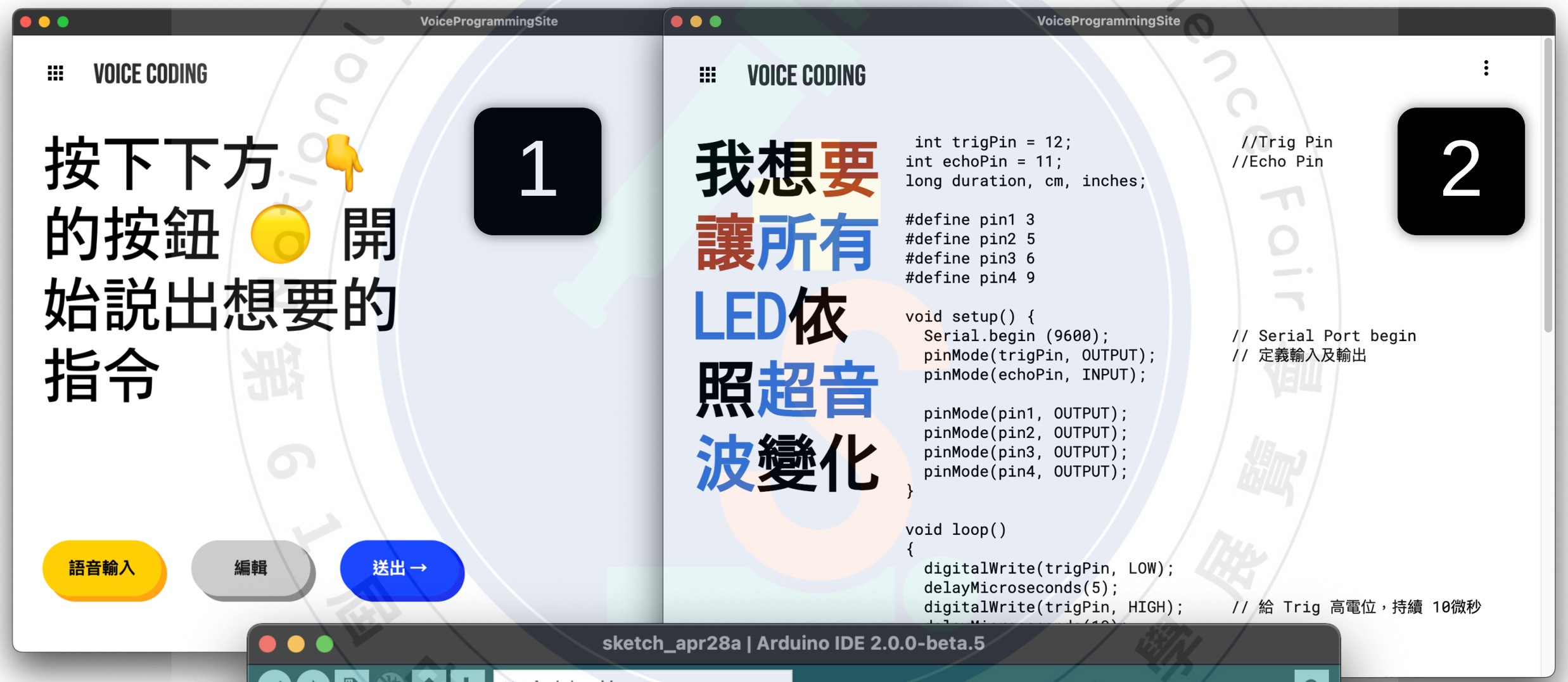
註：圖為開發過程中git的branch

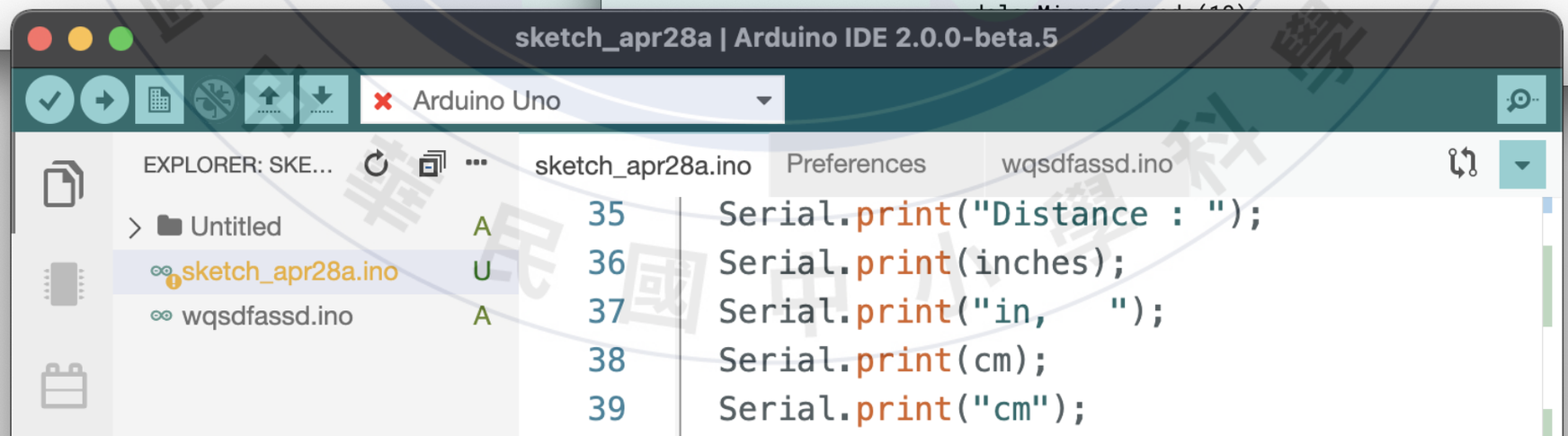
註：圖為程式關係架構圖

研究結果

實驗模組測試 - 軟體

- 斷詞NLP測試（軟體）

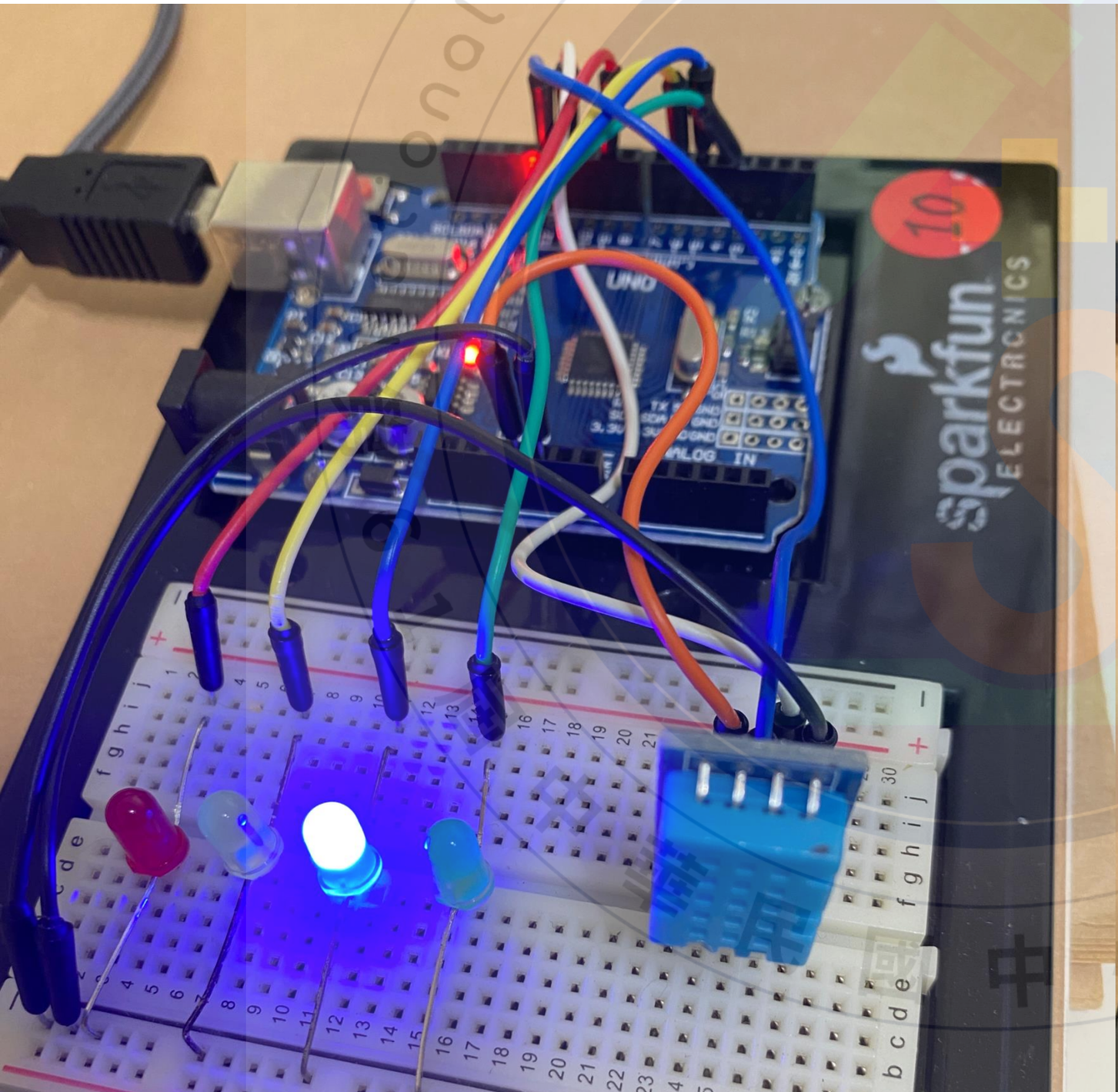


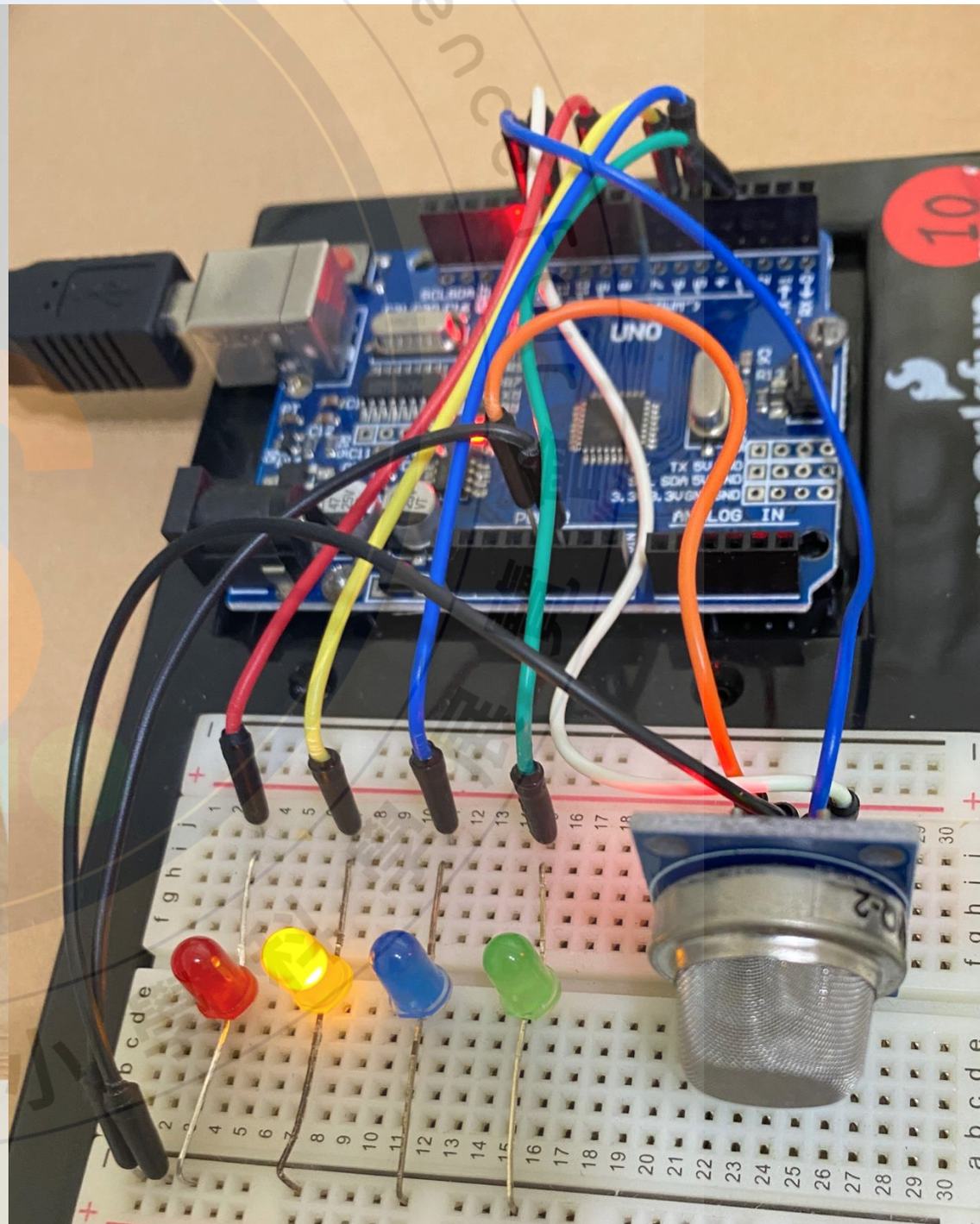


研究結果

實驗模組測試 - 硬體與韌體

- 下圖為實驗模組測試的例圖

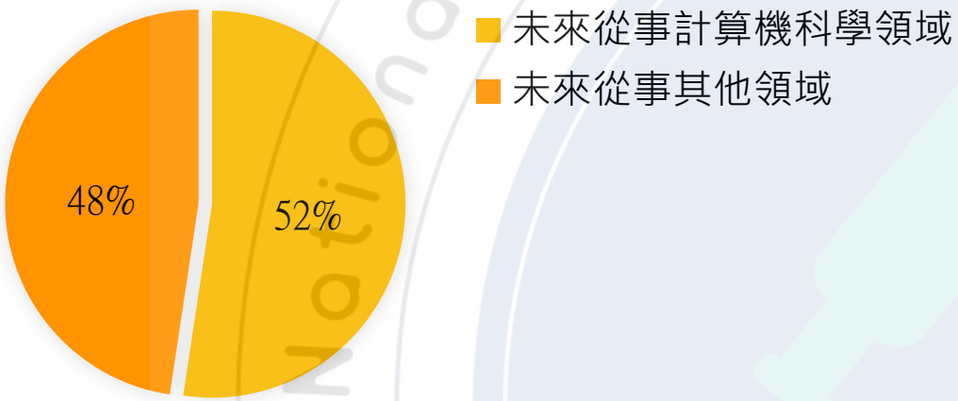




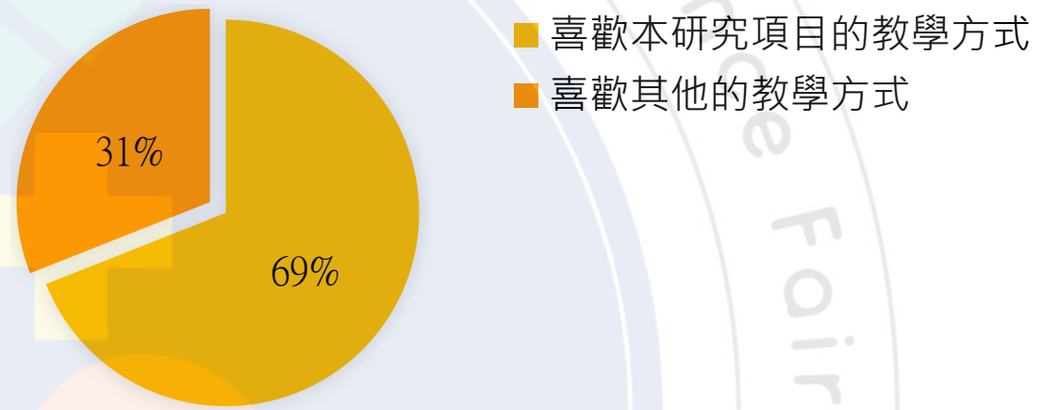
研究結果解釋

問卷分析與解釋

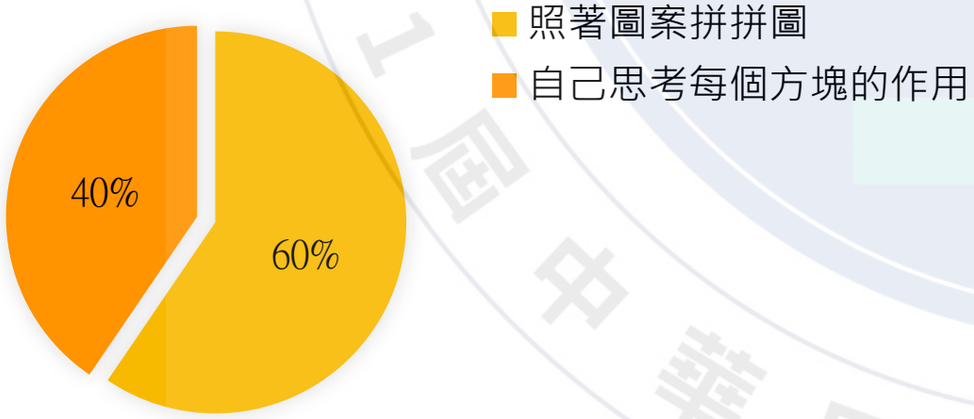
- **意見調查**（資訊科42人班級）



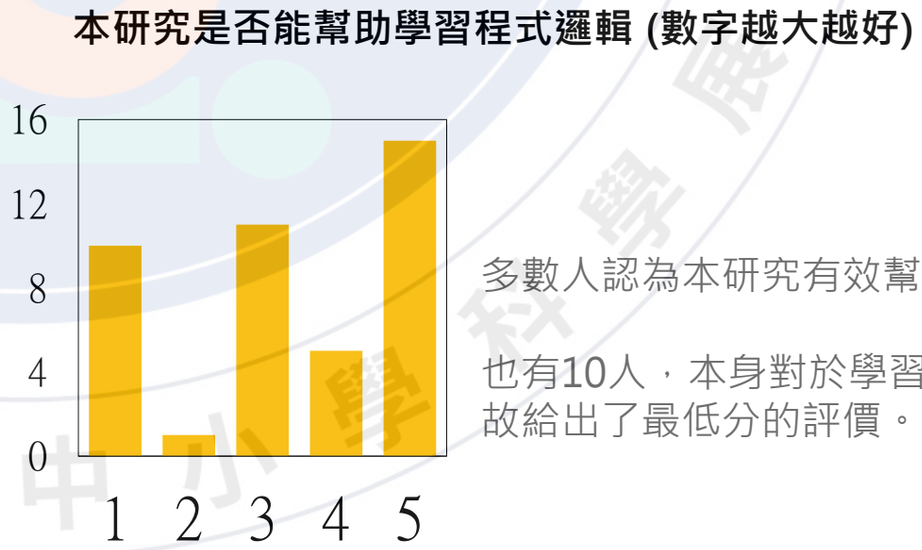
雖為資訊科，但是有一半的人，改變心意未來不會在計算機領域從事工作。也就是說，近半數人未來大概不需寫程式。



29人認為，透過口述產生程式碼的方式，比起傳統的教學方式來的有趣，更能被接受。



25人在使用拼圖式程式設計軟體，都只是將其當作純粹的拼圖遊戲。



多數人認為本研究有效幫助學習程式。

也有10人，本身對於學習程式並不感興趣，故給出了最低分的評價。

結論

- **目前的功能適配進度**
  - 自然語言結合推薦算法開發程式碼生成器之研究，比起分區賽，增加了控制超音波、溫濕度傳感器的功能，並且將原先的LED功能做好了完整的適配，可以透過口語化的方式操作所有的LED。
  - 依照目前的進度，預估再六個月的時間，本研究將能覆蓋Arduino所支持的大部分元件。

- **本研究項目對於程式教育的幫助**
  - 根據學生的意見回饋，我們認為本計畫能夠幫助學生練習自己的表達能力，且同時增添趣味性，使學生感到興趣，能夠有效延長專注時間。

- **未來目標**
  - 經過這個實驗之後，實踐利用口語化的方式產生程式碼，學習其他的程式語言（例如Java、VB、Python），將可能被實現。

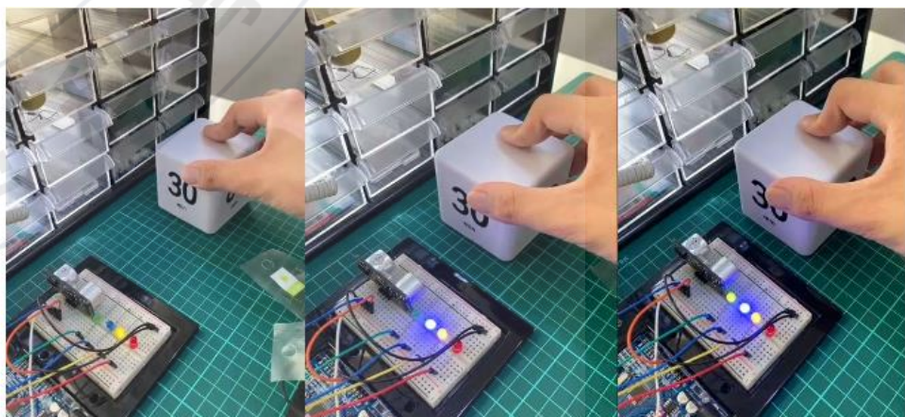
我想要讓所有LED停

```
 int trigPin = 12;
int echoPin = 11;
long duration, cm,

#define pin1 3
#define pin2 5
#define pin3 6
#define pin4 9
```

```
Serial.print(inches);
Serial.print("in,   ");
Serial.print(cm);
Serial.print("cm");
Serial.println();

digitalWrite(pin1, LOW);
```



參考資料

- 劉漢偉(2019)·Vue.js 從入門到項目實戰·清華大學出版社。
- 史蒂文金尼·譯:徐曙光(2019)·Electron 跨平台開發實戰·清華大學出版社。
- Shelley Powers·譯:楊尊一(2016)·Node 學習手冊第二版·歐萊禮。
- David Flanagan(2012) ·JavaScript大全(第六版)·歐萊禮。
- 勁樺科技(2017)·入門首選 Python 程式設計·台科大圖書股份有限公司。
- Kenny Chen(2020)·Jieba 原理介紹·取自：https://blog.kennycoder.io/2020/02/12/Python-知名Jieba中文斷詞工具教學/