國立中央大學工學院
酷派爭霸讚—程式設計菁英賽
成果報告書

『自動化工具機聲音檢測系統』

參賽隊員1：陳薇如

參賽隊員2：李芳誼

參賽隊員3：馬欣妤

目錄

一、作品簡介

在工業界，加工機的運轉聲音常伴隨著加工過程出現異常而有不尋常的聲音，例如撞刀時，常會伴隨著尖銳高頻的金屬聲，以及較平常分貝還大的聲響。因此，「自動化工具機聲音檢測系統」是一款使用 Python 進行編寫的聲音檢測系統，藉由分析聲音中的分貝和頻率來判斷加工機運作過程是否出現異常。目前常見到檢測加工異常的方法，除了有傳統靠人力監督機器運作之外，有些較新穎的設備會附有透過刀具震動或刀具與工件接觸產生的物理波形來檢測刀具加工是否異常。因此，透過「分析聲音」來達到「檢測加工是否異常」的想法非常嶄新，極具創意性。

在應用方面，除了檢測加工機運作是否正常之外，我們更期待將它延伸應用在「放電加工」等，加工聲音非常微小，且肉眼不易辨識的製程中。由於放電加工製程需在工作液底下進行，肉眼不易辨識，只能透過工程師長期累積的經驗，用聽覺辨別加工聲音是否異常，加工是否成功。若將辨別交由電腦完成，不僅降低出錯率，也降低人力成本。

程式的可塑性極大，在實際應用範圍也非常廣，以電腦取代人耳，打破人類先天對頻率接收範圍的限制。這項作品主要是應用在工業界，根據不同的工具機可以調整不同檢測的分貝和頻率範圍。若將程式碼與工具機結合，在檢測到異常聲音而無人回應，便可以立即發出警告，將工具機停機以降低損害。在程式完善方面，可以加入機器學習，讓電腦自動判別造成異常加工聲音的可能原因，先從電腦分析的原因著手解決，盡速讓加工回歸正常。

二、專題團隊及任務分工

1. 團員個人經歷

陳薇如

機械系設計組三年級，擅長 Python 程式語言，在大學的課程中，已經修習完畢數個 Python 程式語言課程，能夠快速搜索需要的資訊，有效利用手邊的資源，並進行統整和應用。

李芳誼

機械系設計組三年級，升大學前先修 Python，對 Python 很感興趣，會利用課餘時間精進 Python。時常有許多創意、新穎的想法，擅長將所學知識和實際應用結合，迸發各種不同靈感。

馬欣妤

機械系光機電組三年級，機械系有關程式語言的課程都非常高分(例如:微控制器 100 分、光機電介面實驗 100 分)，非常擅長 C 語言，因為邏輯很好，在 Python 程式語言方面也毫不遜色。

2. 任務分工方式

陳薇如

主要負責檢測音量和頻率大小、設計系統介面的程式碼、程式 debug、海報排版、剪輯影片。

李芳誼

主要負責整體概念發想、檢測音量大小程式碼、程式 debug、海報設計、拍攝影片、成果設計報告書。

馬欣妤

主要負責檢測頻率大小、程式 debug、海報內容構想、拍攝影片、成果設計報告書。

三、成果報告

專題名稱: 自動化工具機聲音檢測系統

陳薇如，李芳誼，馬欣妤

摘要：

由於工具機自動化趨勢，我們用 python 設計一款「自動化工具機聲音檢測系統」，用來取代人力監控機器，對加工聲音的音量和頻率進行嚴謹的分析以及即時可視化數據。在加工異常時，立刻發出警告通知，降低加工過程的損失風險。

一、 動機

由於在三月分時，參加了 2023 年台北國際工具機展覽 TIMTOS，得到許多啟發。在展場中我們看見許多新型加工工具機，配合機械手臂來達到自動化生產的過程，以及只要將工件放入一台自動加工的工具機，不管是車床、銑床、切削等多種加工，一次就能完成。在這樣自動化生產的趨勢下，卻有一個無法避免的缺失，也就是自動化生產雖然提升了生產效率，但加工過程的異常與否卻需要靠人力來監督，倘若將監督的工作交由電腦完成，必定能為工業界帶來更多效益。

此外，我們也考慮到有些加工製程是必須在肉眼不易辨識的情況下進行，最典型的例子就是放電加工，因為需要在工作液底下進行加工，因此加工過程是否異常，大部分都須仰賴人為聽力進行辨認。有鑑於放電加工所產生的聲音很小，不容易聽清楚，我們也可以讓這項工作交由電腦來完成，相信一定可以提高加工過程辨別的準確率。

二、 研究方法

由於此次作品是用 python 針對聲音進行分析，我們希望能夠寫出一個「自動化工具機聲音檢測系統」，應用在加工過程中，監聽是否出現異常聲音，並即時繪出圖形。一旦聲音出現異常，必定有高機率是加工過程出現了異常，系統必須即時跳出警告通知，並在製程結束後將音檔分析資料儲存，以便事後追查情況。

我們先瞭解基本聲音參數的設定，包括對聲音取樣本數、樣本格式、聲道數量以及如何開啟麥克風。在對 pyaudio 聲音數據有一定的了解後，我們開始思考對程式功能的分配。因為需要使用的套件及功能眾多，需要使用物件導向的概念分配，由流程圖著手，如 Figure1，將每一個模塊清楚分配後，才開始撰寫程式碼。

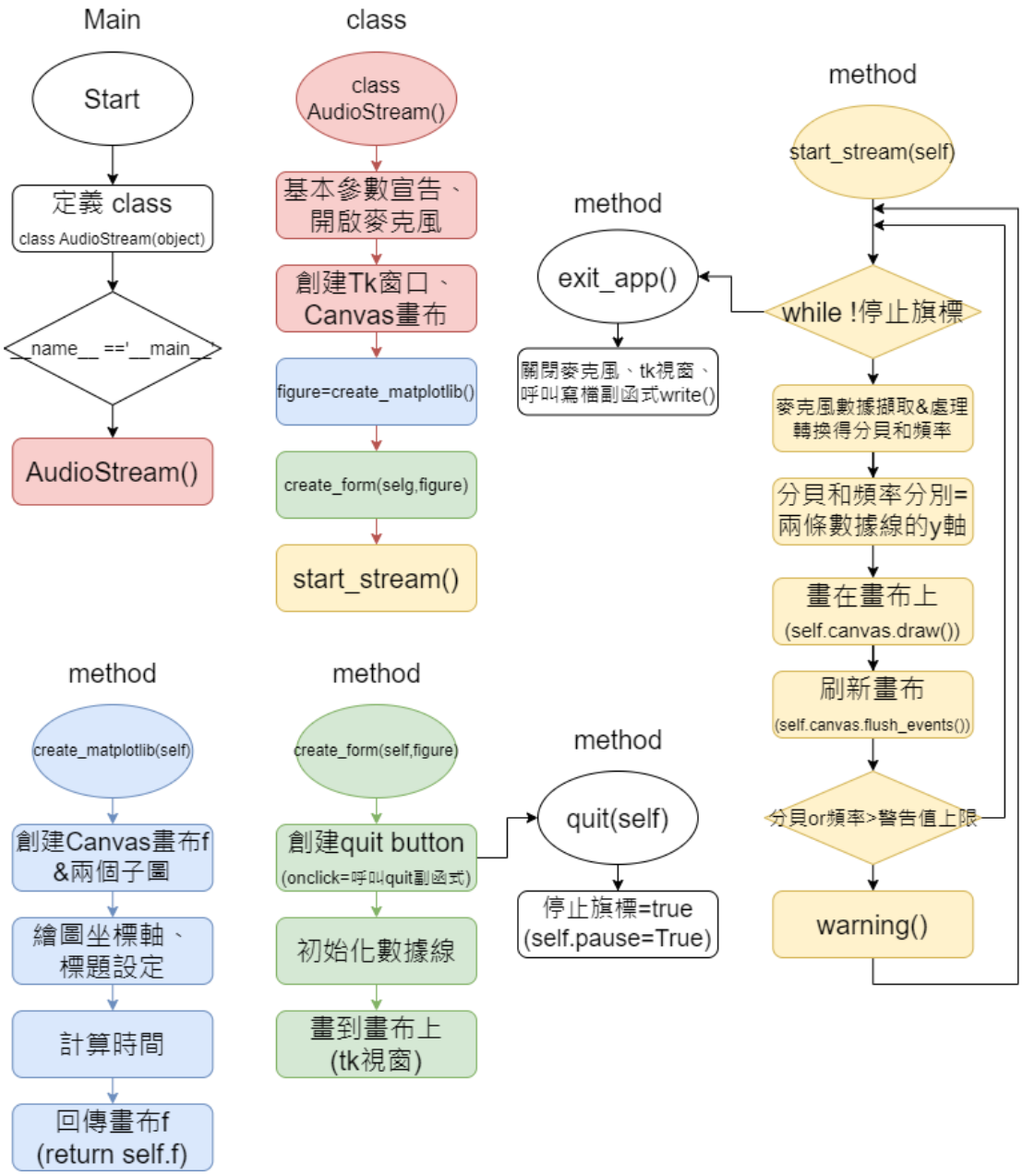

Figure1. 流程圖

我們共有 8 個函式，分別為 create_matplotlib(建立 matplotlib 圖)、create_form(建立 tk 窗口)、start_stream(開始錄音)、exit_app(離開系統)、find_max(找最大值)、warning(警告)、write(寫檔案)、quit(按鈕函數)。

其中，create_matplotlib 會回傳一張圖給 create_form，在有初始值的情況下，start_stream 打開麥克風，開始錄音和數據轉換，並且不斷刷新畫布，讓式窗上的數據更加即時。

在需要警告時，我們會呼叫 warning。在按下按鈕 buttom 時，呼叫 quit 去停止 start_stream，接著 exit_app 便會啟動，將數據儲存、找最大值、寫檔案等等。完整程式碼如下：

```
class AudioStream(object):
    def __init__(self):                                    # 常數
```

```
        self.CHUNK = 1024 * 2                       # 每幀樣本數
        self.FORMAT = pyaudio.paInt16               # 樣本格式
        self.CHANNELS = 1                           # 聲道數量
        self.RATE = 44100                           # 一秒的樣本數
        self.pause = False                          # 停止變數
        self.frames = []                            # 紀錄數據 list
        self.p = pyaudio.PyAudio()                  # 建立 pyaudio 物件
        self.filename = "record.csv"                # 數據紀錄處
        self.max = 0                                # 最大值
        self.warning_db = 75                        # 分貝上限(警告用)
        self.warning_freq = 1000                    # 頻率上限(警告用)
        self.frame_count = 0                        # 數幀數

        self.root = Tk()                            # 創建窗口
        self.canvas = Canvas()                      # 創建顯示圖形的畫布
        self.figure = self.create_matplotlib()      #  figure 為 matplotlib 所畫圖形對象
        self.create_form(self.figure)               # 將 figure 顯示在 create_form 視窗上

        self.stream = self.p.open(                  # 從麥克風中獲取音訊數據
            format=self.FORMAT,
            channels=self.CHANNELS,
            rate=self.RATE,
            input=True,
            output=True,
            frames_per_buffer=self.CHUNK            # 每一個 buffer 中的幀數
        )
        self.start_stream()                         # 開始錄音加畫圖函式
        self.root.mainloop()

## matplot initiall setting
def create_matplotlib(self):
        # 創建 matplotlib 繪圖對象 f 及兩個子圖
        self.f = plt.figure(figsize=(160,120), dpi = 100)
        self.ax = self.f.add_subplot(2,1,1)
        self.ax2 = self.f.add_subplot(2,1,2)

        ##分貝圖 xy 軸範圍
        self.ax.set_ylim(0,100)
        self.ax.set_xlim(1,2048)
```

```
        ##頻率圖 xy 軸範圍
        self.ax2.set_ylim(0,10)
        self.ax2.set_xlim(20,20000)

        ##標題&xy 軸標題設定
        self.ax.set_title('AUDIO WAVEFORM')
        self.ax.set_xlabel('samples')
        self.ax.set_ylabel('volume(DB)')
        self.ax2.set_xlabel('frequency(Hz)')
        self.ax2.set_ylabel('strength')

        # 計算初始時間
        self.start_time = time.time()                    # 紀錄開始時間

        # matplot 回傳 figure 給繪製視窗函式
        return self.f

## 繪製視窗函式
def create_form(self,figure):
        button = Button(master=self.root, text="Quit", command=self.quit)
        button.pack(side=RIGHT,ipadx=50)

        # 初始化數據線
        x = np.arange(0,self.CHUNK)                      #  x 的數值
        xf = np.linspace(0, self.RATE, self.CHUNK)
        self.line, = self.ax.plot(x,np.random.rand(self.CHUNK),'r')  # 先隨機出一條線
        self.line2, = self.ax2.plot(xf,np.random.rand(self.CHUNK),'b')

        # 把繪製的圖形(數據)顯示到 tkinter 視窗上
        self.canvas=FigureCanvasTkAgg(figure,self.root)
        self.canvas.draw()
        self.canvas.get_tk_widget().pack(side=BOTTOM)

# 開始錄音加畫圖
def start_stream(self):
        print('stream started')                                  # 開始錄音
        # 把繪製的圖形顯示到 tkinter 窗口上
        while not self.pause :
            # 數據擷取&處理
```

```
            data = self.stream.read(self.CHUNK)                    # 讀取 stream 資料
            data_int = struct.unpack(str(self.CHUNK) + 'h', data)  # byte to interger
            data_dB = [20 * math.log10(abs(x)) if x != 0 else 0 for x in data_int] # 轉換為分貝
            yf = np.abs(fft(data_int)[0:self.CHUNK])/ (128 * self.CHUNK)

            #分貝&頻率的 y 軸數據畫成線
            self.line.set_ydata(data_dB)
            self.line2.set_ydata(yf)

            #丟到圖上
            self.canvas.draw()                          # 畫在 canvas 畫布 fig 圖上
            self.canvas.flush_events()                  # 刷新畫布
            self.frames.append(list(data_dB))

            #若分貝>設定上限則 warning
            if max(data_dB) > (self.warning_db):   # 若分貝大於設定上限
                self.warning()
            if max(yf) > (self.warning_freq):       # 若頻率大於設定上限
                self.warning()
        else:
            self.exit_app()                         # 當停止時，呼叫 exit_app

    # 離開 app
    def exit_app(self):
        self.p.close(self.stream)                   # 關閉麥克風
        print('stream closed')                      # 提示字元
        self.find_max()                             # 找最大值
        self.write()                                # 寫入檔案
        self.root.quit()
        self.root.destroy()

    # 找最大值函數
    def find_max(self):                                     # 找最大值函數
        self.max= max(max(sig) for sig in self.frames)      # 用 for 找二維 list
        print("max vol(dB) = ",self.max)                    # 提示字元

    # 警告函數
    def warning(self):
        t = time.time() - self.start_time                   # 經過秒數
```

```
            print(f"warning,time={t:.4f}")                    # 警告字元

        #寫檔函數
        def write(self):
            wf = open(self.filename, 'w')                    # 讀寫聲音記錄檔
            writer = csv.writer(wf)                          # 使用 csv.writer()建立一個 csv 寫入器
            writer.writerows(self.frames)
            # writer.writerows()將二維 list 中的每一個子 list 寫入到檔案

            wf.close()                                       # 關掉檔案
            print("file has been saved")                     # 資料寫入提示

        # Buttom 結束
        def quit(self):
            self.pause = True                                # 停止參數=True


if __name__ == '__main__':
    AudioStream()
```

三、 結果討論

此系統將對加工製成聲音的由電腦讀取的數據轉換成音量分貝和頻率赫茲進行分析，在程式中我們會設置分貝以及赫茲的上限值，若製程中的聲音超出設定的上限值，則在 terminal 會跳出警告通知，告知加工異常的時間、分貝或頻率值，並在按下 quit 鍵後關閉介面，將整個資料儲存程 csv 檔，在日後需要調閱資料時可供參考。

四、 結論

因應現在製造生產自動化趨勢，每個自動加工製程雖然自動化提升了效率，卻仍需要靠人力控管機台。以我們設計的「自動化工具機聲音檢測系統」取代人力，來監督自動加工的過程，或是應用在肉眼難以辨認的加工製程，可改善工程問題中的缺失。有鑑於程式的可塑性非常高，面對不同加工製程和加工機的不同，可以調整需要檢測的分貝以及頻率範圍。

因為次系統能夠即時繪出聲音相關參數圖形，意即將聲音進行可視化。如此一來，即便是超出人耳能聽見的頻率範圍，也可以輕鬆在圖片中顯示。在機械行業，有許多加工是無法用肉眼看見的，如放電加工，往往是在液體中進行，或是許多綜合型加工機，會完全沒有透明窗口，因此，若人工用監控，也是只能依靠聲音。若將聲音監控系統裝置在加工機內，如此一來可以降低外界工廠噪音，將分析更加精準精確化。若經過長時間的使用也可以用來分析工具機機況，例如是否有震動過大的問題，而去檢查零件的磨損或變形程度，或是螺絲的

鬆動等等。若加工過程出現異常加工聲音，系統可以即時跳出警告資訊，倘若將此功能和工具機結合，便可以在出現異常聲音時即時停止工具機的運轉，最後不管加工是否成功，系統都會將整個製程聲音檔案儲存，以便往後需要資料時可供參考。

這項設計是一個創新且新穎的想法，以往辨認加工是否正常都是利用刀具接觸到工件所產生的物理波行進行分析，或是工具機已經附有檢測加工是否正常的系統，用聲音檢測加工製程運作的想法幾乎可以說是空前絕後，因此我們非常期待往後能將這個系統真正實行，改善加工品質。

四、專題製作心得

陳薇如:

一開始我和芳誼是在 2023 台北國際工具機展上，因為參觀了各式各樣與自動化機械的相關系統，所以有了這個用聲音做機器監測的想法。不過當時只是單純一個念頭而已。後來，我們在機械系舉辦的程式競賽時想到這個題目，開始去研究如何使用 python 去蒐集聲音，並且研究 python 收音的方法，還有對音訊資料處理方法。那次比賽我們用 matplotlib 製作了即時音量視覺化的程式，也是這個想法的開端，形成了整個系統的雛形。

後來因得名而推薦來參加此次比賽，我們更認真的去研究 python 在介面製作的方法，如何在 tk 介面上置入 matplotlib 圖形。起初我認為並不複雜，因為之前寫過 c#.net 和 python 的 tk 介面，應當可以快速上手，但是後來才發現，由於網路上提供的資料並不多，許多都只是一些簡單的例題，再加上我自己很少寫這麼大型的程式，對物件導向的程式並沒有很多的設計經驗，因此剛開始在摸索各種參數的傳遞，尤其在 matplotlib 和 python tk 的連接上，一直在試錯和思考，如何將兩者放在一起、何時刷新、應該在迴圈內或外等等，最後順利的將頻率的圖及整個系統完善了。另外我們在上次的比賽中，教授有向我們提問，我們音量的單位是什麼，我們這次也有將這個問題解除，將聲音的音量改為分貝，頻率的單位為赫茲。

李芳誼:

這次題目我們將工具機運轉聲音和程式結合，這個概念在市面上幾乎查不太到有人這樣做，是一個非常新穎的想法。所以當初想到這個題目的時候，我真的熱血沸騰，心想著一定要將這個酷點子做出來! 程式的撰寫固然非常重要，但能夠應用程式語言才是更勝一籌。

在參加這次比賽之前，我一直對自己的程式很沒有自信，因為大一必修的 C++課程我都聽不懂老師在說什麼，這門課還因此差點被當掉，讓我有好長一段時間非常抗拒寫程式，大學部關於程式的課程也都刻意避開不修。但是在後來某一個暑假，因為在家太無聊，我想起高中自己有先修過 python，我想著如果要和程式語言重修舊好，應該要先從自己熟悉的語言著手。於是在我自學 python 一段時間後，便也學出一點滋味來了!

這次的比賽也是鼓起很大的勇氣才報名參加，因為我的隊友都比我還要會寫程式，我很怕自己拖累隊友。幸好之前自學 python 的那些基礎概念都有派上用場，在這次競賽中，我寫的程式碼也有被採用，真的非常感動!為了寫出能用的程式碼，自己也上網做了很多功課，最後在和組員一起 debug 的同時，也學到很多。

雖然這次比賽還有很多改進的空間，像是等待跑程式的時間真的有點久，但這個創新的點子才只是一開始！由於人耳的先天限制無可避免，相信將程式優化後，這個想法能幫助許多在工廠被噪音淹沒的耳朵！

馬欣妤:

這次被芳誼和薇如找來一起比賽，一開始其實有點擔心，因為 python 一直是我的罩門，加上工具機機械加工也不是我的專業，所以很擔心我會幫不上忙。但另外兩位隊員一直鼓勵我參賽，我也想說那就試試看吧，總不能一直跟 python 過不去！

硬著頭皮來研究程式之後發現我真的不是普通的討厭 python，因為上高中之前最一開始學習的語言是 C++，後來高中電腦課接觸到 python 後，就覺得語法太不一樣了，讓我很不習慣，寫得很挫折。加上 python 是直譯語言，相較 C 和 C++等編譯語言，python 每次執行程式的時候都要讓我等好久，所以我真的好不喜歡它！之前我也比較少這種需要繪圖分析數據的需求，大部分都是在寫微處理器的程式，使用 C 語言較多，所以相較之下對 python 的依賴程度就比較低。

在這次競賽中，我上網找了大量的資源。一開始我其實連 class 要怎麼寫怎麼用都不知道，稍微惡補了這方面的知識，才開始找各種套件的使用方法、範例程式碼之類的來參考。其中遇到最大的挫折還是呼叫 method 的時候，該在哪裡呼叫，這樣寫它會怎麼遞迴，哪些變數會隨著遞迴被刷新.....等等。

而對於這次的作品，如果有更多時間，我其實有兩個想改善的地方：第一就是我想研究能讓整個程式執行時更省時的方法。這學期我有修資工系開的程式語言課程，就有介紹到各種類型的變數在執行時的時間差（例如靜態變數、堆疊動態變數等等），尤其 python 這種直譯語言，每次迴圈（遞迴）的這些微小時間差加總起來都會影響很大。因此，想讓程式更有效率執行的話，這些小細節就顯得更加重要了。

第二點想改善的地方是，我們這次使用的 tk 介面，感覺有很多功能都還沒用到，例如我們可以加更多的功能按鈕，讓整支程式的功能變得更強大，像是暫停和繼續鍵、計時鍵，或是即時顯示分貝值（直接以數字顯示在介面上那種）。