National Cheng Kung University

Institute of Space and Plasma Sciences

2022 Annual Report

專題生 ： 陳德岳

指導教授 ： 張博宇 博士

日期 ： 2023/1/13

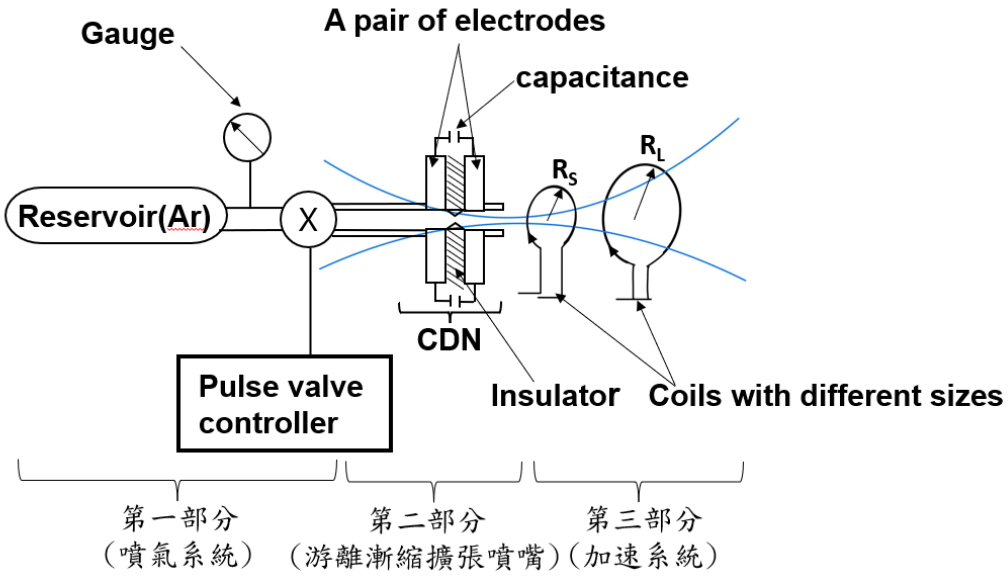
摘要

2022 年下半年正式加入成大電漿所脈衝電漿實驗室(PPL)製作專題，專題的主題是要開發出一個用不平衡的角向捏縮產生推力的電漿推進器，此電漿推進器主要有三個部分:進氣的噴氣系統，游離漸縮擴張噴嘴(德拉瓦噴嘴)，以及透過磁場壓力產生推力的加速系統。在噴氣系統中，我會設計一個脈衝電磁閥門控制器來控制進氣量，當氣體進入第二部分，由一對跨有高電壓的電極所構成的漸縮擴張噴嘴時，除了透過電弧放電產生電漿之外，同時形成準直向後移動的電漿噴流，流到下游的加速系統，由一對大小不同的線圈所組成，提供脈衝的磁場壓力差將電漿噴流再往後推來產生更大的推力。在這份年度報告當中，我將會從第一部分噴氣系統中的設計閥門控制器(電路板)開始，其中包括 Arduino 的練習，畫 Layout 圖，焊接電路板，另外我也使用了 COMSOL 做了噴出氣體的模擬及練習，本次報告將會詳細的紀錄各項的執行狀況。

目錄

一、 不平衡角向捏縮的脈衝電漿推進器之開發

專題所要開發的〝不平衡角向捏縮的脈衝電漿推進器〞如圖一所示，總共會分成三個部分，第一個部分是噴氣系統(gas-puff system)的設置，此階段使用一個灌有氬氣的鋼瓶，脈衝電磁閥，和控制電磁閥的控制器(Pulse valve controller)；第二個部分是由一對跨有高壓的電極和絕緣體進行加工設置而成的游離漸縮擴張噴嘴(Ionization Convergent Divergent Nozzle (ICDN))；第三個部分是加速系統，由一對不同大小的線圈設置而成。



圖一 電漿推進器簡圖。

本推進器的運作原理從第一部分開始，當氣瓶送出氣體時，由脈衝電磁閥提供脈衝氣體到第二部分的 ICDN 當中，誘發組成 ICDN 並跨有一對高壓的電極之間的電弧放電，將氣體游離成電漿態。除此之外，ICDN 可以讓電漿噴流以超音速的速度流進入第三部分的線圈中。同時，我們將利用電容放電產生的脈衝電流驅動線圈，產生磁場 B_{coil} 及相對應的磁場壓力 P_B:

$$B_{coil} = \frac{\mu_0 I}{2R} \quad ,(1)$$

$$P_B = \frac{B^2}{2\mu_0} \quad ,(2)$$

其中 I 為電流，R_S、R_L 分別為圖一中大小不同的線圈半徑。因為兩個線圈的大小不同，產生的磁場不同，產生磁場壓力差ΔP_B:

$$\Delta P_B = \frac{\mu_0 I^2}{8}\left(\frac{1}{R_S^2} - \frac{1}{R_L^2}\right) \quad ,(3)$$

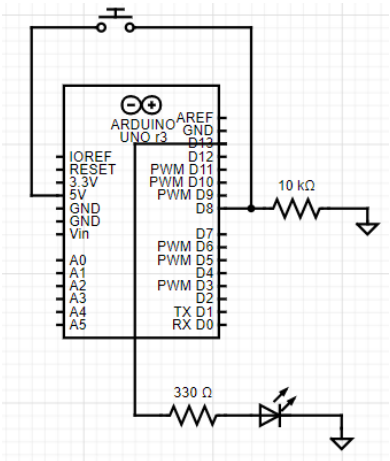
將第二部分產生的電漿向後推出，根據牛頓第三運動定律，推進器便會得到向前的推力。而其中第一部分噴氣系統中的脈衝電磁閥將會由控制器來控制，此控制器會使用 Arduino nano 板來控制，因此在下一章節我將會詳細介紹我練習 Arduino 的細節。

二、 Arduino 練習

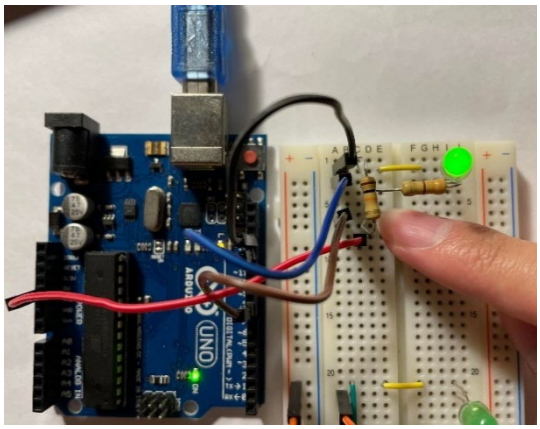
這次的 Arduino 練習主要是以輸出方波為目標，因為我們電漿推進器氣體噴射閥門控制就是要用 Arduino 輸出的方波來控制。這次的練習範圍參考了「超圖解 Arduino 互動設計第 4 版」的第一章到第五章，包含了基本的 Arduino 設置，麵包版實驗和連接阜的設定，經由練習 LED 燈的開關，了解使用 Arduino 輸出方波的作法，接下來將會詳細介紹所做的各個實驗。

2-1 用麵包板組裝開關電路

剛開始的練習都以簡單的電路為主，電路圖如圖二，紅線接 5V 輸出端，棕線接按鈕的輸入端(D8 腳)，藍線接 LED 的輸出端(D13 腳)，黑線接地(GND)。實驗結果:當按下按鈕時，LED 燈亮起，鬆開按鈕則熄滅，見圖三。



圖二 2-1 電路圖



圖三 2-1 實際組裝

程式碼如圖四，第 1 至第 2 行分別宣告 LED 燈和開關(SW)的腳位為 13 及 8，setup 函式中的 pinMode 則是個別設定接腳的模式，LED 設為輸出，SW 則設為輸入。loop 函式中的 digitalRead 功能是讀取開關 SW 的數值，digitalWrite 則是依照 SW 的狀態點亮或熄滅 LED 燈。

```
1  const byte LED = 13;
2  const byte SW = 8;

3  void setup()
   {
4    pinMode(LED, OUTPUT);
5    pinMode(SW, INPUT);
   }

6  void loop()
   {
7    bool val = digitalRead(SW);
8    digitalWrite(LED,val);

   }
```

圖四 2-1 程式碼

2-2 LED 切換開關及其修正

2-2 的電路組裝沿用 2-1 的電路圖，如圖二所示。實驗目的是，按一下按鈕開關點亮 LED 燈，再按一下則熄滅 LED 燈，與 2-1 最大的不同就是不用一直按著開關就可以持續讓 LED 燈亮起。

程式碼如圖五，第 1、2 行 LED 跟開關 SW 宣告的腳位不變，而為了讓 LED 燈能持續亮起，在第 3、4 行多宣告了 lastState 和 toggle 兩個布林變數(bool)，用來記錄上一次的開關狀態和輸出給 LED 的訊號。另外，setup 函式的功能與 2-1 描述相同。

loop 函式中寫了兩個 if 的條件程式:

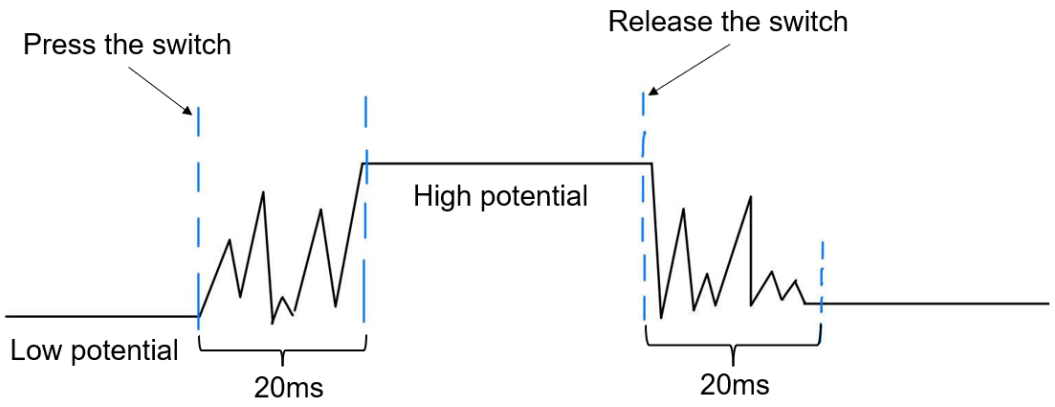
第 10 行第一個 if: 先讀取(digitalRead)開關的狀態，如果是高電位(按下那一刻)則讓狀態暫存(lastState)。

第 12 行第二個 if: 開關狀態恢復(按完鬆開)，呈現低電位時，將變數 toggle 的內容反轉，使本來應該不亮的 LED 燈反轉燈號使之持續亮起；或原本亮的 LED 燈反轉為不亮，最後輸出呈現實驗結果(digitalWrite)。

```
1  const byte LED = 13;
2  const byte SW = 8;
3  bool lastState = LOW;
4  bool toggle = LOW;

5  void setup()
   {
6    pinMode(LED, OUTPUT);
7    pinMode(SW, INPUT);
   }

8  void loop()
   {
9    bool b1 = digitalRead(SW);

10   if(b1)
     {
11    lastState = b1;
     }
12   if(b1 != lastState)
     {
13     toggle = !toggle;
14     digitalWrite(LED, toggle);
15     lastState = LOW;
     }
   }
```

圖五 2-2 程式碼

實驗結果:當開關連續按時，無法立即顯示出效果，即有 LED 無法改變燈號的現象。這個現象是機械式開關彈跳問題導致如圖六，簡單來說就是開關按下那一刻會出現一個微小的雜訊波干擾。書裡提供兩個改善無法改變燈號的現象的方法，一個是直接修改程式碼，第二個是組一個新的 RC 濾波電路來消除，接下來會一一介紹。



圖六 按鈕的機械式開關雜訊

2-2-1 使用程式修正 LED 切換開關

為了消除機械式開關彈跳的雜訊波干擾，有一個最簡單的方法是直接忽略掉圖六中那段訊號，在讀取開關數值前先暫停(delay)個 20 毫秒，再讀取開關值，這樣就能跳過那段小雜訊。

程式碼如圖七，只需在第 4 行 loop 函式中加入 delay 函式，再存到 lastState，其餘程式一樣故省略。

實驗結果:更改完之後，連續按開關即可順利切換 LED 燈號，接下來的章節 2-2-2 則是不更動程式的狀況下(沿用 2-2 的程式)更改電路來達到像 2-2-1 一樣的效果。

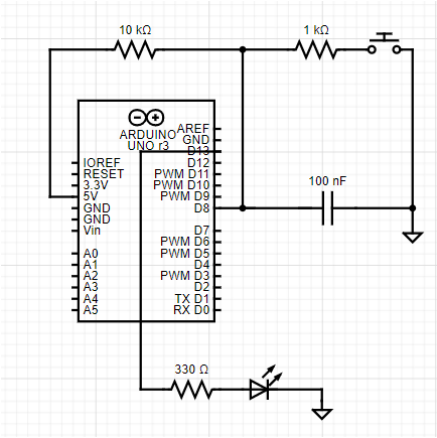
```
1  void loop()
   {
2    bool b1 = digitalRead(SW);

3    if(b1)
     {
4     delay(20);
5     bool b2 = digitalRead(SW);

6     if(b1 == b2)
      {
7      lastState = b1;
       }
```

圖七 2-2-1 程式碼

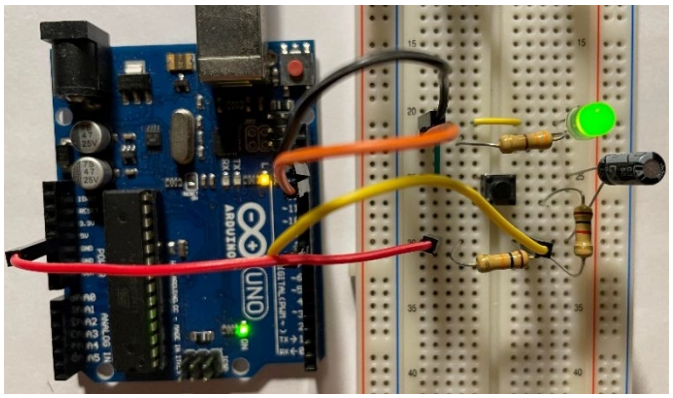
2-2-2 使用 RC 電路消除開關彈跳訊號

這個實驗主要是增加一個電容來達到消除開關彈跳訊號來達到如章節 1-2-1 一樣的效果，圖八為電路圖，紅線接 5V 輸出端，黃線接按鈕的輸入端(D8 腳)，橘線接 LED 的輸出端(D13 腳)，黑線接地(GND)。

實驗結果:按一下開關燈號亮起，再按一下關閉，不會有開關不敏感的問題，圖九為按下開關後之情況。

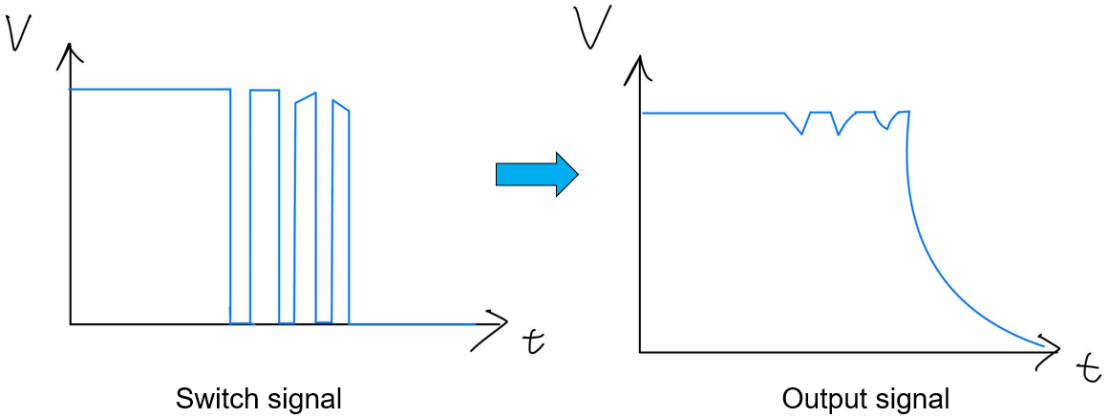


圖八 2-2-2 電路圖



圖九 2-2-2 實際操作

程式碼如圖五，不需要修正，使用 RC 電路的原因是，電容在充放電的過程中，訊號會變成比較平滑的曲線如圖十，也就是開關的高頻雜訊被過濾掉了，所以能夠把這種 RC 電路稱為低通濾波器。如此一來，便能解決 LED 無法改變燈號的現象。



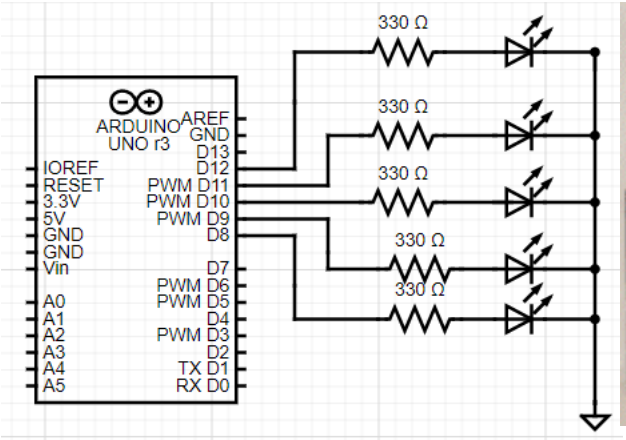
圖十 RC 開關訊號輸出比較圖

2-3 LED 跑馬燈

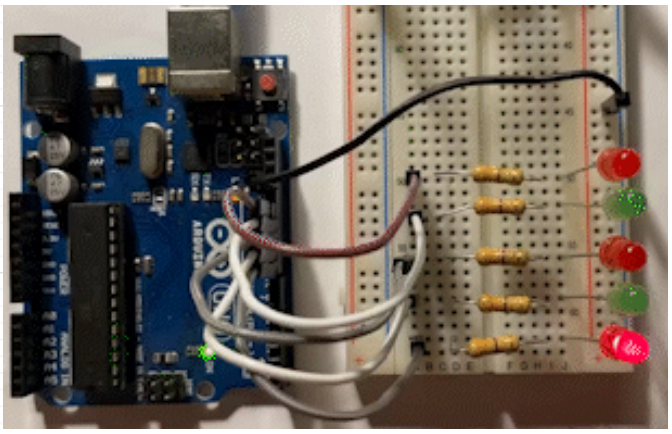
這個章節主要是練習讓 LED 燈輪流亮起和熄滅，產生跑馬燈的效果，主要分成單向跑馬燈和雙向跑馬燈(詳見 2-3-1，2-3-2)亦練習了不同的程式書寫方式來讓整體更有效率。

2-3-1 單向 LED 跑馬燈

電路圖如圖十一，五條線分別接在 LED 的輸出端(D8~D12 腳位)，黑線接地(GND)。實驗結果:LED 燈泡輪流亮起，如圖十一。



圖十一 2-3-1 電路圖



圖十二 2-3-1 實際操作

程式碼如圖十三，先宣告每一個 LED 輸出的腳位(D8~D12 腳位)，在 setup 函式中把模式設為輸出(OUTPUT)，而 loop 函式中則開始分別對每個 LED 燈輸出高電位和低電位，各持續 0.1 秒(delay)以達到跑馬燈的效果，圖十三的 loop 函式有省略 LED3~5 高電位的程式，不過皆和上述相同。

```
const byte LED1 = 8;
const byte LED2 = 9;
const byte LED3 = 10;
const byte LED4 = 11;
const byte LED5 = 12;

void setup() {

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);
}
```

```
void loop() {
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(LED5, LOW);

  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(LED5, LOW);

  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  digitalWrite(LED4, LOW);
  digitalWrite(LED5, LOW);

  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, HIGH);
  digitalWrite(LED5, LOW);
```

圖十三 單向跑馬燈程式碼

圖十三程式的寫法雖然簡單直觀，不過當 LED 燈數量多時，程式就會變得龐大而且效率低，因此接下來使用了 for 迴圈，使程式乾淨整潔，如圖十四。這次宣告的方式跟前一個個別宣告不一樣，而是直接在第 1、2 行宣告起始腳位(starpin)和結束腳位(endpin)。

在 setup 函式中，使用 for 迴圈來設定輸出的模式，從 startpin 開始，每次加 1 一直加到 endpin 為止(第 5 行)，這樣寫就可以省略一個一個設定腳位輸出模式的寫法。

而 loop 函式內的原理也與 setup 函式大同小異，第 3 行一開始設定 lightPin 為 startPin(第 8 腳位的 LED 燈)，每一次輪迴結束就代表一個燈號的亮及熄滅，然後每一個迴圈的 lightPin 都會加一，若 lightPin 與 endPin 相同時將其回到 startPin(第 15 行)這樣輪迴以達到跑馬燈的效果。

```
1  const byte startPin = 8;
2  const byte endPin = 12;
3  byte lightPin = startPin;

4  void setup()
   {
5    for(byte i = startPin; i<= endPin; i++)
     {
6     pinMode(i,OUTPUT);
7     digitalWrite(i,LOW);

       }
   }
```

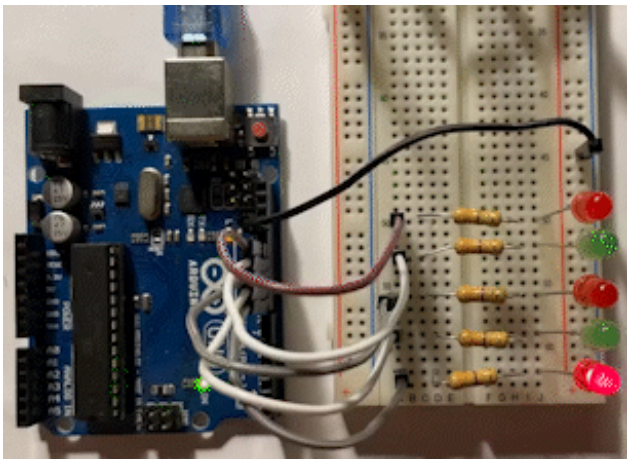
```
8  void loop()
   {
9    digitalWrite(lightPin, HIGH);
10   delay(100);
11   digitalWrite(lightPin, LOW);

12   if(lightPin < endPin)
     {
13     lightPin ++;
14   }else
     {
15    lightPin = startPin;
     }

   }
```

圖十四 for 迴圈改善的程式

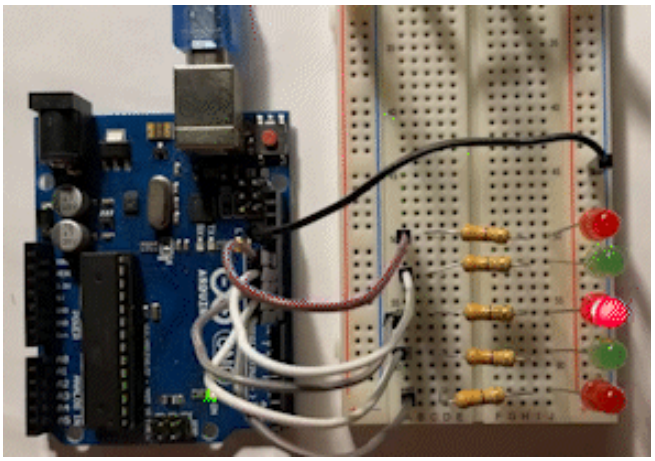
2-3-2 LED 陣列變數來回跑馬燈

本節跟 2-3-2 非常相似，不過此節的跑馬燈是來回跑的，雖然可以直接用類似圖十三的程式來撰寫，不過本節使用陣列變數來撰寫。電路圖則跟 2-3-1 一樣，見圖十一。

實驗結果:LED 燈來回跑，下圖十五與十六分別顯示出與前者普通跑馬燈的差異



圖十五 單向跑馬燈



圖十六 雙向跑馬燈

程式碼如圖十七，在宣告腳位時使用陣列的方式來定義，而 sizeof 函式功能能傳回陣列 LEDs 位元組的大小，以這題為例，LEDs 陣列的元素為 8~12 五個元素見第 1 行，變數值為 5，因此 total 的值為 5(第 2 行)。

setup 函式的程式跟圖十四是一樣的，就是單純把 LED 燈的腳位設為輸出，且輸出的值為 Low 唯一的差別在於使用陣列的格式書寫。

loop 函式中的兩個 for(第 8 行及第 12 行)，可以把它想像成兩個方向，簡單來說就是去跟回來，腳位輸出從 8 到 12，再從 12 回到 8，除了有兩個方向陣列書寫外，其餘的概念皆跟單向跑馬燈相似。

```
1  const byte LEDs[] = {8,9,10,11,12};
2  const byte total = sizeof(LEDs);


3  void setup()
   {
4    for(byte i=0; i<total; i++)
     {
5      pinMode(LEDs[i], OUTPUT);
6      digitalWrite(LEDs[i], LOW);
     }

   }
```

```
7  void loop()
   {
8    for(byte i=0; i<total-1; i++)
     {
9      digitalWrite(LEDs[i],HIGH);
10     delay(100);
11     digitalWrite(LEDs[i],LOW);
     }

12   for(byte i=total-1; i>0; i--)
     {
13     digitalWrite(LEDs[i],HIGH);
14     delay(100);
15     digitalWrite(LEDs[i],LOW);
     }
   }
```

圖十七 2-3-2 程式碼

另外，除了使用 for 迴圈寫之外，我也練習了使用 if 條件式搭配邏輯運算子的方式書寫，圖十八是要加的程式和調整。

先宣告一個陣列元素指引，然後定義跑馬燈的方向，dir 為 1 或-1。

loop 函式中則改成用 if 條件式寫，當 i=0 或 i=4(total=5)時，改變跑馬燈跑的方向。要注意的是，這裡的 0~4 指的是陣列的第 0~4 個元素，不是腳位，對應腳位則是 8~12。由於改方向要加個負號，所以 dir 的初始值是-1 而不是 1，否則 i 會變成負的，腳位會對不上導致燈號不會亮。

這種寫法雖然直觀上可能會有一點不好理解，不過程式又精簡了一些，但兩者的測驗結果都是一樣的，就是讓跑馬燈來回跑。

```
uint8_t i = 0;
int8_t dir = -1;


void loop()
{
  digitalWrite(LEDs[i],HIGH);
  delay(100);
  digitalWrite(LEDs[i],LOW);
  if(i==0||i==total-1)
  {
    dir=-dir;
  }
  i += dir;

}
```

圖十八 2-3-2 程式碼調整

2-4 UART 序列阜通信

這個章節主要練習的部分是 Arduino 與電腦的通訊部分，底下將練習兩個與 Arduino 通訊的簡單程式碼。

這裡 include 了 Serial 的程式庫，使用其中的函式便可以和 Arduino 通訊，9600 是鮑率，就是兩者之間訊號的傳遞速率，必須一致才可以進行通訊，而接下來就是簡單的字串輸出如圖十九。

```
byte ledPin = 13;

void setup() {
  Serial.begin(9600);
  Serial.println("Hello,");
  Serial.print("\tLED pin is: ");
  Serial.print(ledPin);
  Serial.print("\nBYE!");

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

圖十九 字串輸出程式碼

而另一個程式碼也是大同小異，這裡要練習的是輸入字元來控制 Arduino 板上的小 LED 燈，當輸入 1，LED 燈亮，當輸入 0，LED 燈熄滅，同樣重要的是，鮑率必須相同，否則結果會是一串亂碼。

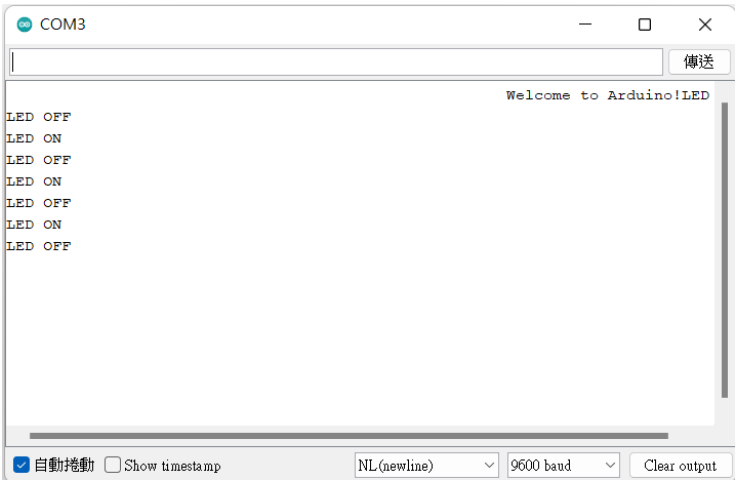
```
char val;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);

  Serial.print("Welcome to Arduino!"

}
```

```
void loop() {
  if(Serial.available()){
    val = Serial.read();
    if(val == '1'){
      digitalWrite(LED_BUILTIN,HIGH);
      Serial.println("LED ON");
    }else if(val == '0'){
      digitalWrite(LED_BUILTIN,LOW);
      Serial.println("LED OFF");
    }
  }
```

圖二十 字元控制程式碼

結果如圖二十一，打開序列阜監控視窗後，當我們輸入數字 1 則視窗顯示 LED ON，反之當我們輸入 0 視窗顯示 LED OFF，即為 Arduino 與電腦通訊。

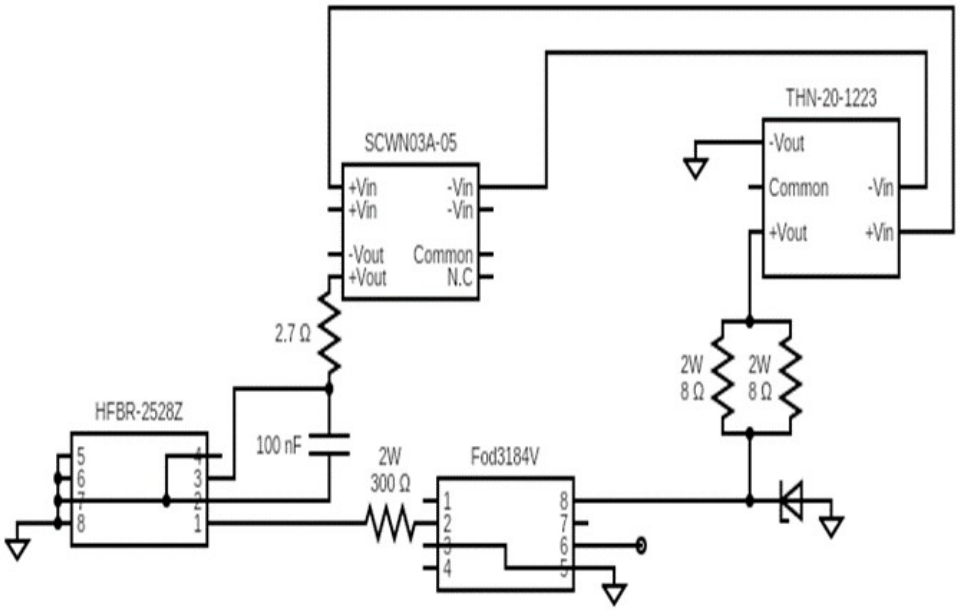
結論:到目前為止，第一章到第五章的 Arduino 練習就告一段落了，下一章節將會進入到閥門控制器電路板的設計及實作焊接。



圖二十一 序列阜監控視窗

三、 脈衝電磁閥門控制器

為了要控制氣體噴出的閥門開關，我們需要設計一個電路來控制閥門，圖一電漿推進器簡圖中的 Pulse-valve controller 就是我們的目標控制器，為了設計出電路板我參考了致賢學長的電路圖(圖二十二)再加以修改。與學長的目標不同，我的控制器將與電推系統共同組成一個運作系統，因此產生的訊號端是由 Arduino 產出，跟學長接收光纖訊號的系統是不同的。



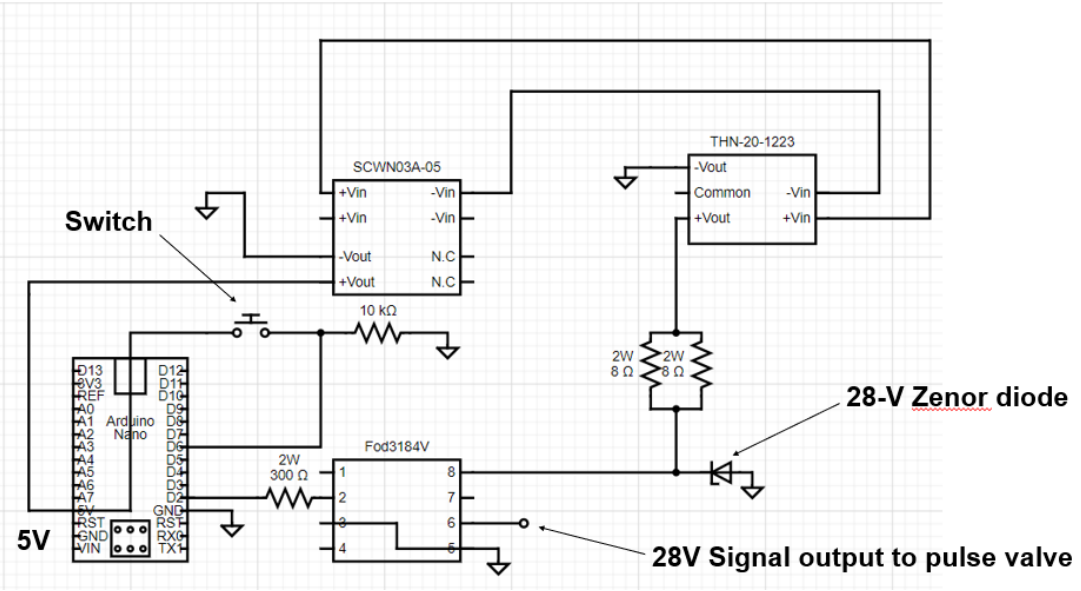
圖二十二 致賢學長的電路圖

最後設計的閥門控制器電路板上包含了許多部份，包含了兩個直流轉換器(DC-DC Converter)，分別供應給 Arduino nano 板所需的 5V 的 SCWN03A-05，以及供應給訊號輸出所需的 30V 的 THN-20-1223；為了要符合驅動脈衝電磁閥(009-1669-900)所需的 28V，加設了用 5 個 5V、1 個 3V 所組成的 28V 稽納二極體，提供 28V 給 Fod3184V 這個閘極驅動器上，最後，我也加了一個按鈕以方便我來測試此控制器，詳細的規格如表一。

表一

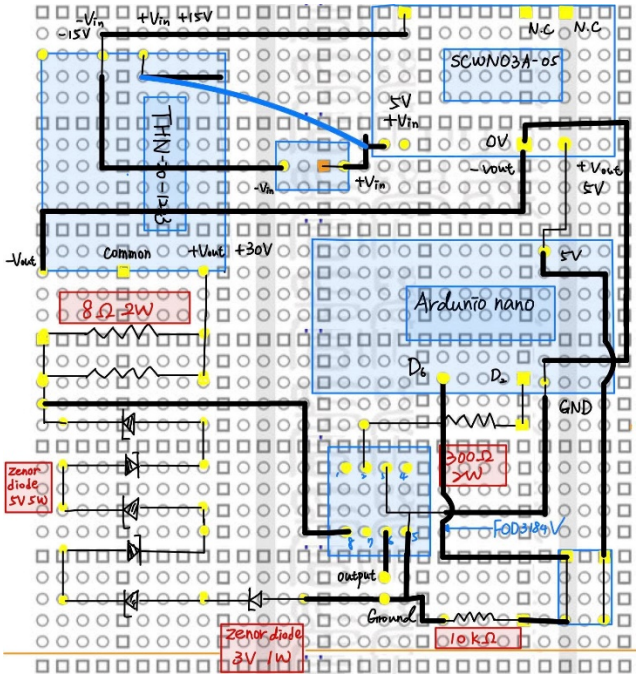
部件	規格
直流轉換器	SCWN03A-05、THN-20-1223
閘極驅動器	Fod3184V
電阻	8Ω*2、300Ω*1、10kΩ*1
稽納二極體	5V*5、3V*1
其他部件	Arduino nano board、開關*1

根據上面的規格，我們設計出圖二十三的電路圖，經由並聯讓 12V 的電池分別供電給兩個直流轉換器，再經各自的線路接地。



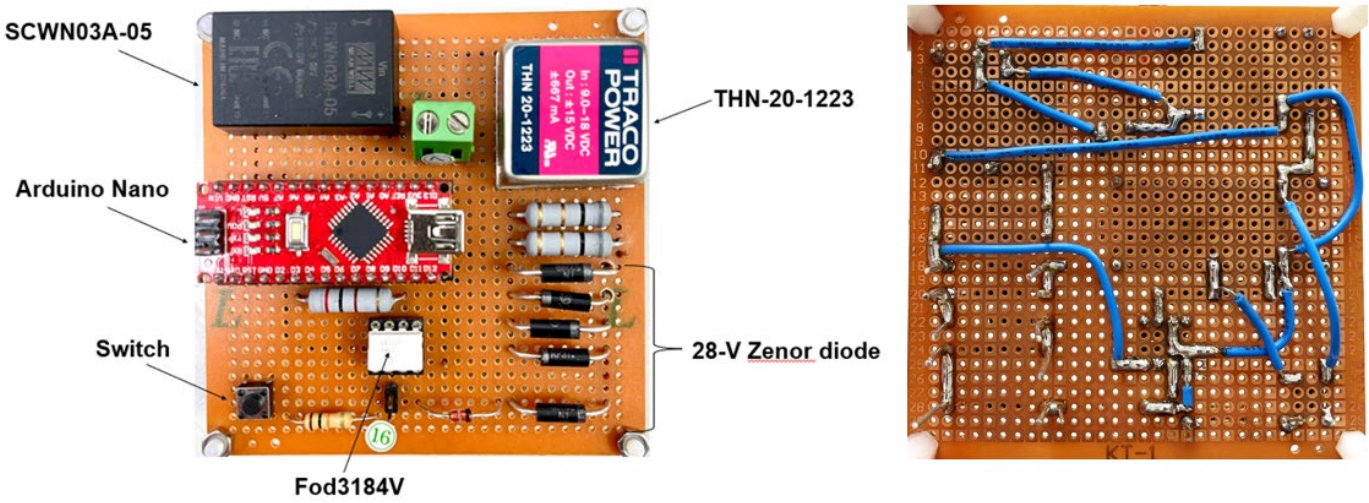
圖二十三 閥門控制器電路圖

有了電路圖之後，我們就可以畫焊接需要用的佈線圖(Layout)，如圖二十四



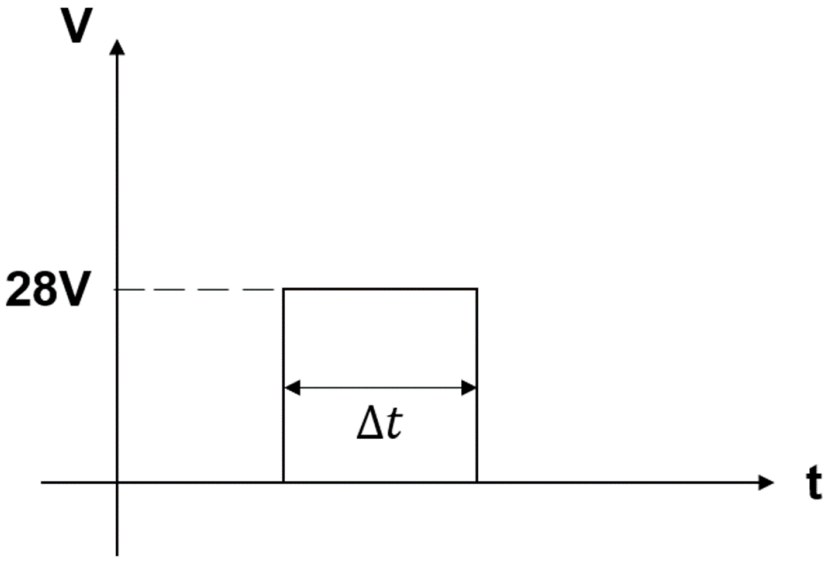
圖二十四 Layout 圖

焊接完的電路板如圖二十五，即脈衝電磁閥門控制器整體。



圖二十五 脈衝電磁閥門控制器

這個控制器，最重要的作用就是用 Arduino nano 板的訊號，產生一個 28V 的方波，如圖二十六，其中的Δt是可以根據我們的實驗去調整。



圖二十六 28V 方波

做出來的控制器後續會來測試看它是否能正常運作，預計使用圖五的程式碼來執行。

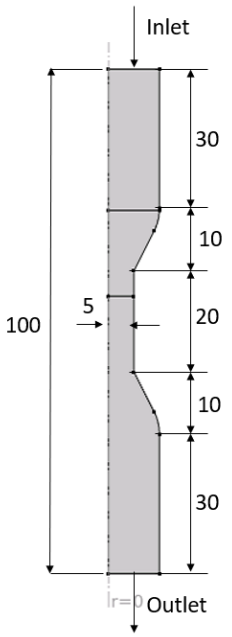
四、 COMSOL 模擬練習及應用

為了確保噴出的氣體能準直的穿越線圈中心，如圖一，我們希望先透過 COMSOL 模擬來了解氣體噴出的狀況是否符合我們的需求。本章節將會練習 COMSOL 模擬程式，主要分成兩個部分都會先以穩態的狀況進行模擬:第一個部分，為了熟悉 COMSOL 的介面及操作，我參考了皮托科技編著的 COMSOL 有限元素分析快易通」中的漸縮擴張管路內的流體分析，在這個部分主要是以層流模擬為主;而第二部分我會嘗試來模擬高馬赫數流體，並搭配我們電漿推進器的系統模型，會使用高馬赫數來進行是因為我們預期我們的系統推出的氣體是在超音速的情況。

4-1 穩態層流模擬

4-1-1 Geometry

為了模擬漸縮擴張管路內的層流模擬，我們首先必須先繪出結構圖。這裡使用的是 2D 軸對稱的方式，使用的研究是穩態層流分析，根據圖十五中的規格，進入 COMSOL 內的 Model Builder 視窗，在 Geometry 內，我們可以新增所需的線段即可以畫出我們想要漸縮擴張管路。



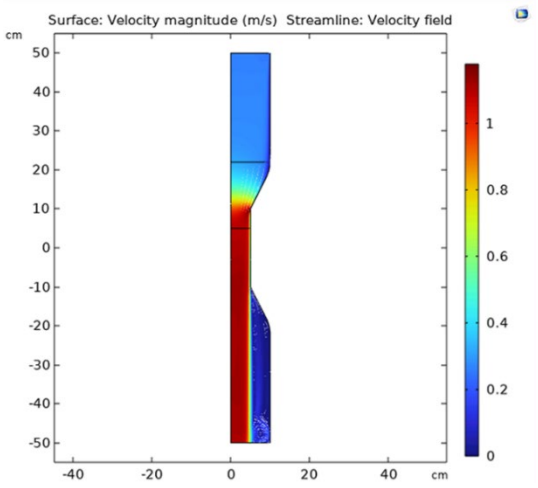
Materials	Argon
Inlet Pressure	1Pa
Outlet Pressure	0Pa
Equation	Q=VA

Unit : cm

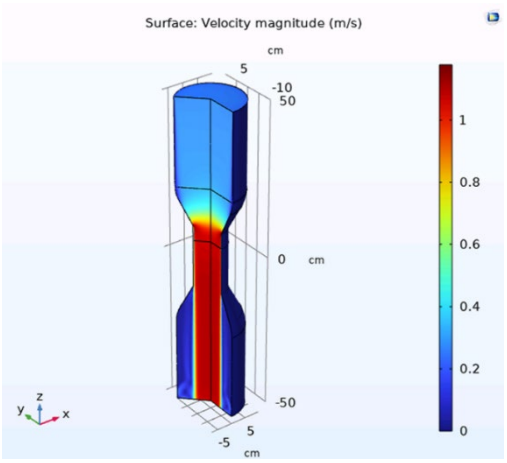
圖二十七 漸縮擴張管路規格

4-1-2 Material and Boundary Condition

畫完 2D 軸對稱圖後，我們就可以開始選擇模擬的邊界條件。在 Model Builder 中的 Material 選擇氬氣，氣體進入的入口和出口分別將壓力設為 1Pa 和 0Pa，如圖二十七。根據流量公式，流量等於速度乘上截面面積，我們可以預估得知，在管子漸縮處流體會因為截面積縮減而加速，等一切都設定完之後，按下開始研究進行模擬，模擬結果如圖二十八、圖二十九。

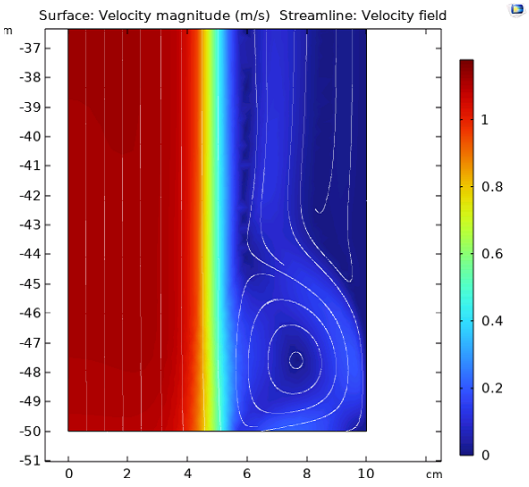


圖二十八 2D 速度分布

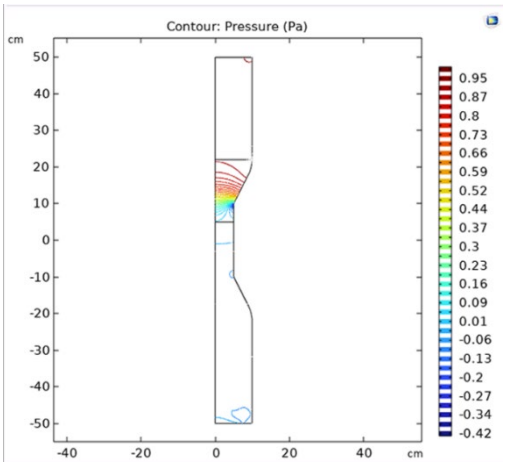


圖二十九 3D 速度分布

圖二十八及圖二十九，分別為模擬 2D 平面圖及 3D 立體圖的情況，從這兩張圖我們可以發現，速度的確在漸縮處加快，這說明我們之前的假設是對的。放大圖二十八我們也可以觀察到流體在出口附近的渦流情形，如圖三十。而圖三十一則顯示出管路內的壓力分佈。



圖三十 流體在出口附近的渦流



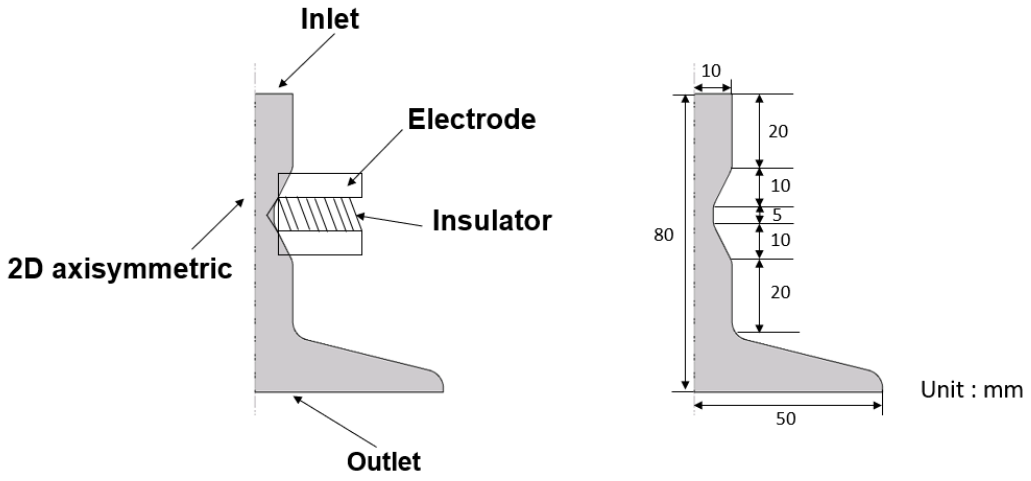
圖三十一 壓力分佈

4-2 穩態高馬赫數流體模擬

在我們電漿推進器系統中的模擬，因為噴出的氣體是以聲速數量級的速度擴散，所以用超音速的狀況才可以讓氣體準直的噴射出去因此，我們使用的研究是高馬赫數流體的模擬，然後依照圖三十二的規格畫出理想的 2D 軸對稱結構圖。

4-2-1 Geometry

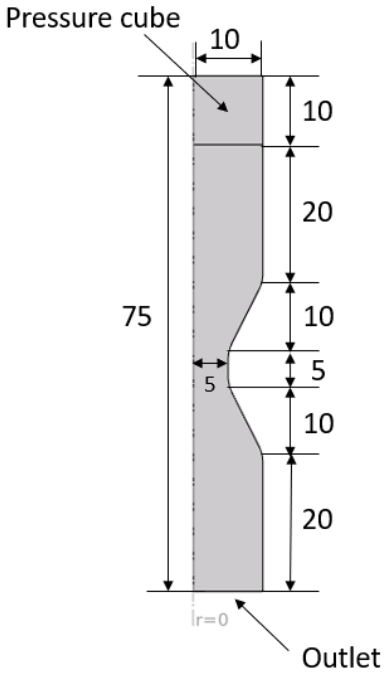
圖二十就是以圖九電推的簡圖下去繪製出來的，中間漸縮處設定為 5㎜ 是為了要讓那對電極進行電弧放電所以要在 5㎜ 以內，出口的擴增處則是為了要模擬氣體推出環境也就是真空艙的情況。



圖三十二 穩態高馬赫數流體模擬規格

4-2-2 Material and Boundary Condition

在執行這個模擬的時候出現了錯誤，推估原因可能是因為邊界條件的問題，像是入口和出口的壓力設定不能為 0，或是出口的擴增處構型問題導致。為了找出錯誤，我修改了圖三十二的構型，把擴增的區域拿掉，變成只有簡單的漸縮擴增管道，如圖二十七那樣。另外，我也把邊界條件改成使用壓力塊的方式，整體規格和構型如圖三十三。

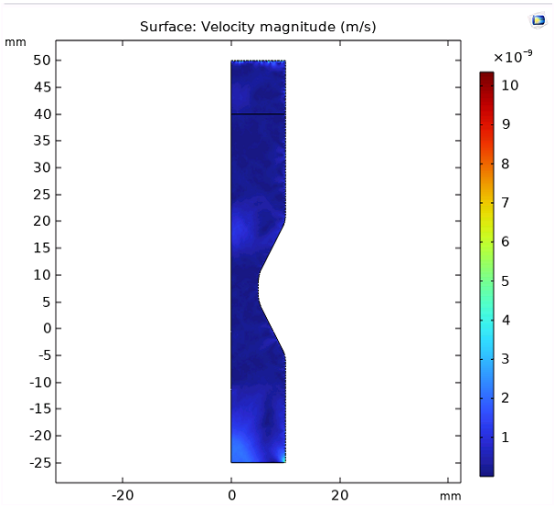


Materials	Argon
Inlet Pressure	5atm
Outlet Pressure	1atm
Temperature	293.15K

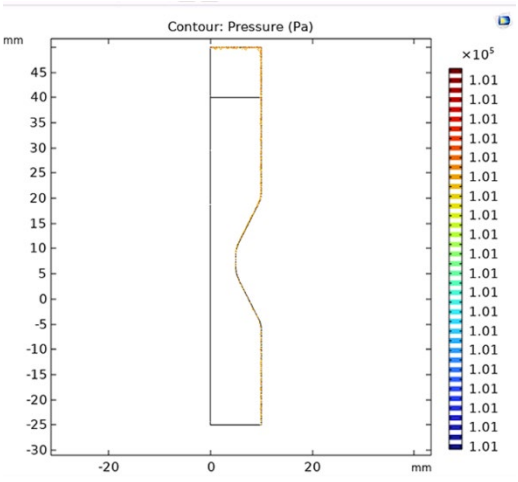
Unit : mm

圖三十三 規格

另外為了避免邊界條件壓力值有 0 的狀況出現，把 inlet 和 outlet 的值改為 1atm 跟 5atm，跑出來的結果如圖三十四與三十五。



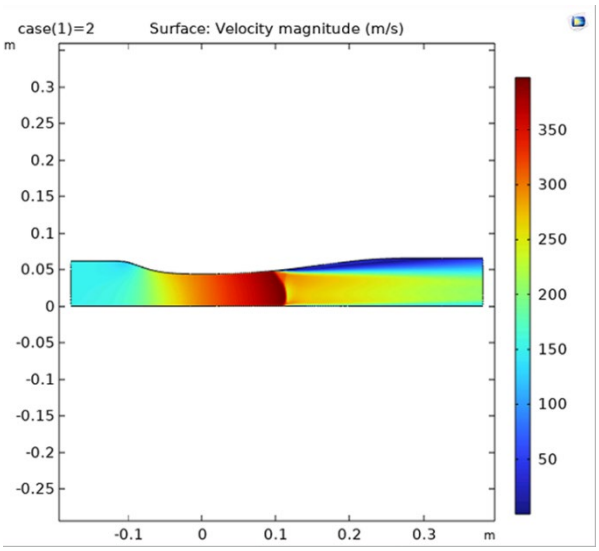
圖三十四 速度分布圖



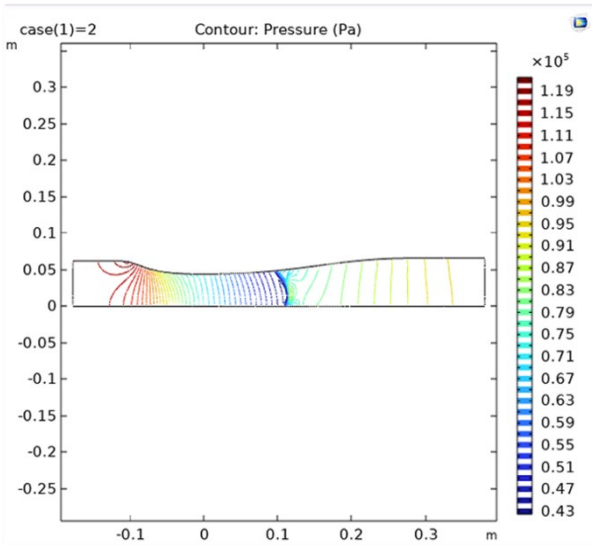
圖三十五 壓力分布圖

從圖三十四跟圖三十五，我們可以發現到，速度和壓力的分布在整個管道是沒有變化的，而且也維持 0。這個原因是因為，在系統達到穩態的情況時，剛開始我們設置的壓力方塊，在一段時間過後氣體就流光了，所以這整個模擬

才會看不出速度的變化。後續的改善方式可以朝向使用時變的研究方式，或者更改邊界的設定條件等等，如圖二十四，圖二十五，就是我們預期想要看到的模擬結果(穩態高馬赫數流體模擬)，也是未來努力的目標。



圖三十六 預期速度分佈模擬



圖三十七 預期壓力分佈模擬

總結來說，這個章節的模擬後續會持續的進行，預計會從上圖的 COMSOL 範例繼續下去做修改，並看能不能套用到我們的構型上，如圖三十二。

五、 未來工作

未來的工作，在寒假期間，我會繼續嘗試 COMSOL 模擬，另外也會開始測試脈衝電磁閥門控制器，也就是使用 Arduino nano 板送出訊號，產生方波來控制電磁閥，測試我的程式以及我焊接的電路板是否能順利的開啟閥門。

二月初我會開始準備申請大專生計畫，包括整理之前做的專題資料以及撰寫企劃書，預計二月下旬送件，開學後會繼續專題的進度開始製作電漿推進器的第二階段，也就是 Convergent Divergent Nozzle(CDN)的部分。

六、 總結

2022 年的下半年，開始做電漿推進的專題，從一開始的 Arduino 練習打基礎，目的是要銜接製作脈衝電磁閥門控制器，用 5V 的 Arduino 訊號轉成 28V 的輸出方波，控制器電路板的部分在這學期已全部焊接完畢，等之後測試才會知道結果。而使用 COMSOL 模擬的部份，完成了基礎的層流分析，高馬赫數流體的分析模擬仍在進行當中，預計會和下學期的 CDN 製作銜接。

七、 參考資料

趙英傑《超圖解 Arduino 互動設計入門（第四版）》台灣 旗標出版社 2020 年三月出版

皮托科技《COMSOL 有限元素分析快易通》

COMSOL Application Gallary : Transonic Flow in a Sajben Diffuser (https://www.comsol.com/model/transonic-flow-in-a-sajben-diffuser-10407)