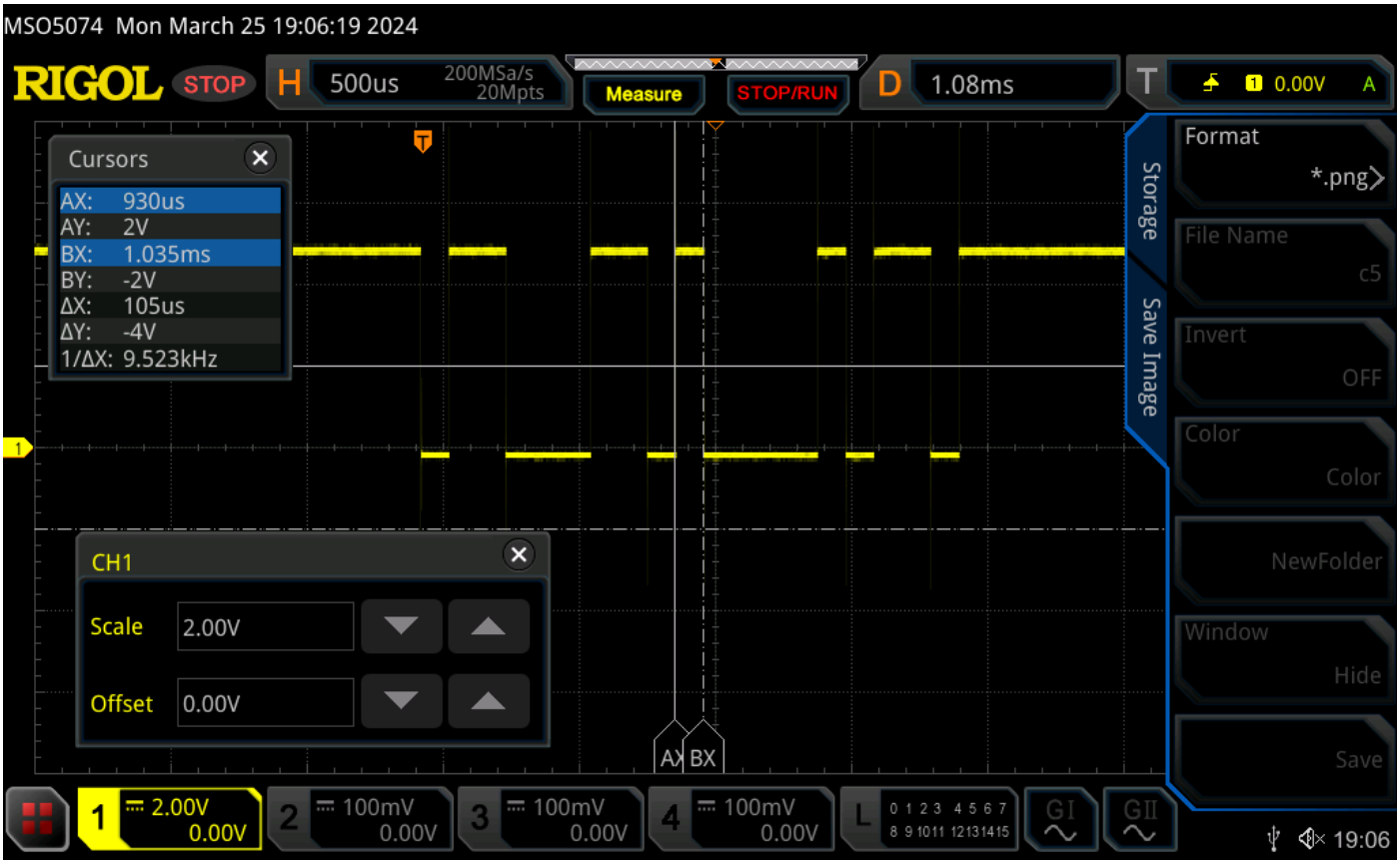
REPORT

Experiment 1: UART protocol

English name I am using:

CHOU,CHIEN-AN

1. UART frame waveform



2. Frame content (Fill the blank with 0 or 1)

	START	Bit0 (LSB)	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7 (MSB)	STOP
1st frame	0	1	1	0	0	0	1	1	0	1
2nd frame	0	0	0	0	1	0	1	1	0	1

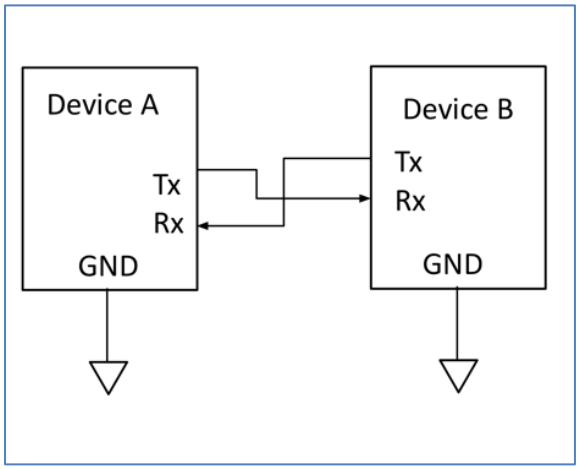
3. The interval of a bit is 1/9523=105.008u (second) which means the Barud rate is equal to 9523 (bps).

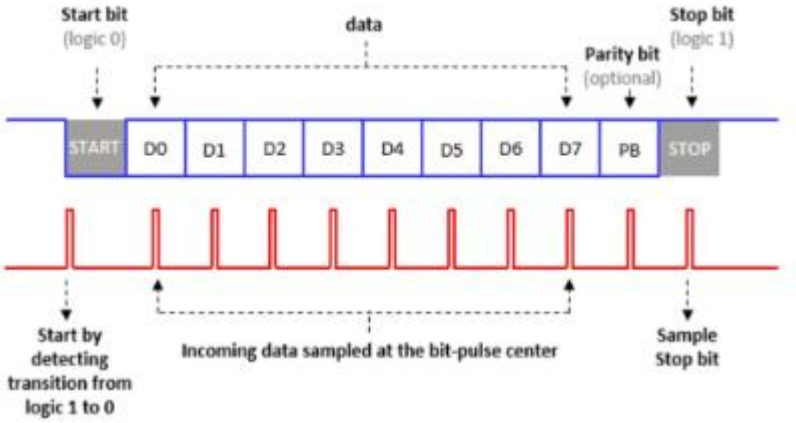
UART protocol 介紹與電路圖分析：

UART 為通用異步(非同步)收發傳輸器（Universal Asynchronous Receiver/Transmitter)。其功能是將要傳輸的資料在串列通信與並行通信之間加以轉換，通常用於兩個裝置之間的溝通，作為把並行輸入信號轉成串列輸出信號的晶片，UART 通常被集成於其他通訊接口的連結上。

而 UART protocol 特色為雙向非同步(裝置之間不需要互相傳遞或共用 clock 信號來進行同步)，資料內容之間的傳遞是使用序列的方式傳遞，而由於非同步的緣故，所以互傳資料的兩個裝置必須事先設定好彼此的傳輸速率，並確保一致。

而裝置一般有兩個狀態，分別是 Idle(閒置)狀態跟傳輸狀態，而為了區分兩個狀態，因此傳遞的訊號頭尾會含有起始位元和終止位元，兩者之間才是真正被傳遞的訊號(如右下圖)。





左上圖為兩個用 UART protocol 傳遞訊號的裝置，可以看到兩者有 Tx、Rx 和 gnd 三個對外的接線，而彼此的輸出(Tx)會連接到對方的輸入(Rx)。

Question:

為何沒訊號時，Tx 設定為高電位?

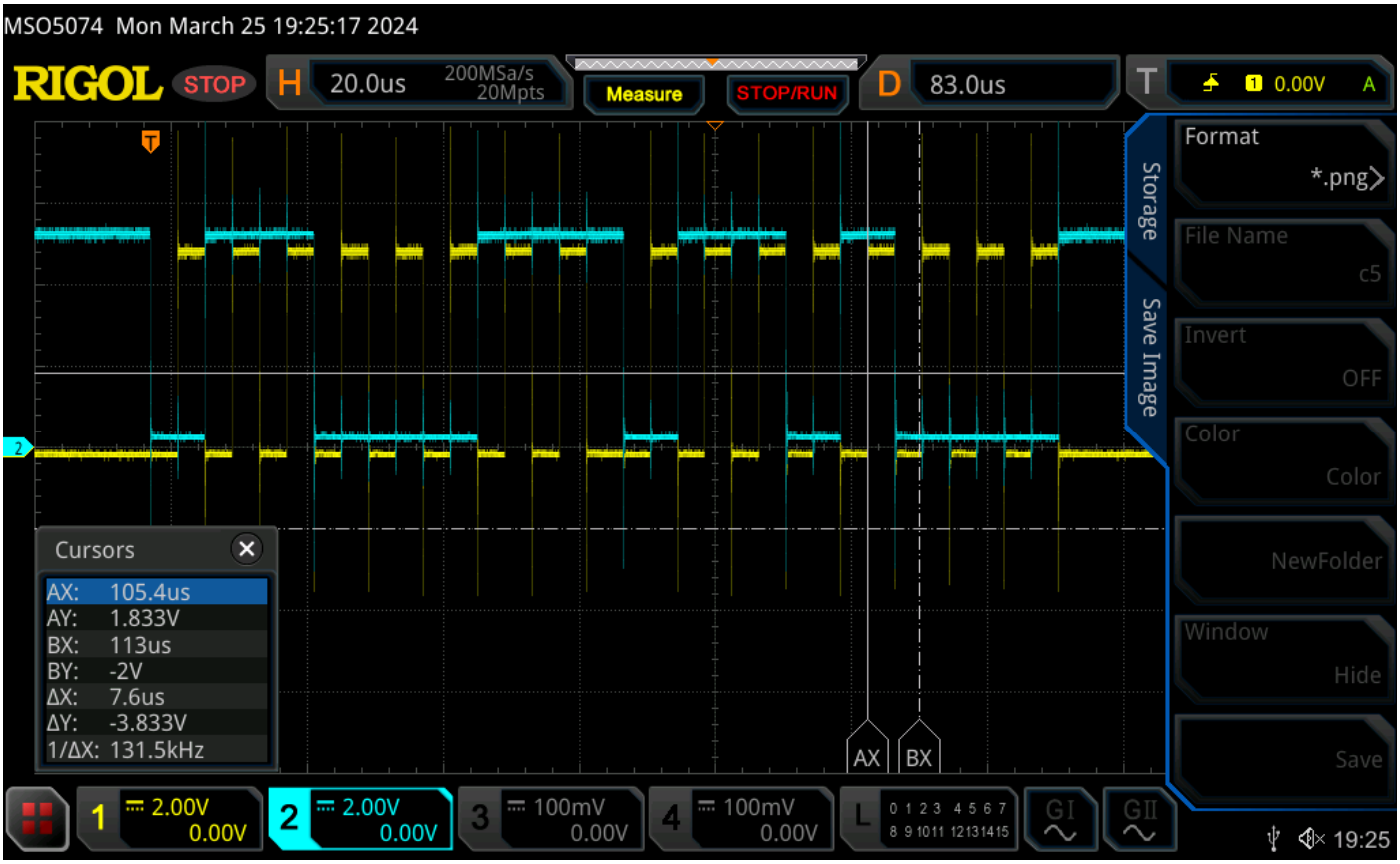
這樣的設計,最主要的原因是為了方便檢測損壞的線路或發射器,因為如果將 Idle 設計成低電位，雖然能降低功耗，但是這卻會使得檢查通訊裝置之間是否有損壞更加困難，因此設計為高電位，即可確認裝置是處於 Idle 狀態，還是損壞狀態。

為什麼不可以先接好電路，才燒 code 到 arduino?

由於 Arduino uno 中，pin 0 跟 pin 1 也是內部晶片在初始化時(燒錄 code 時)會用來接收要燒錄的資訊的所用的針腳，所以在傳輸程式的時候若把 pin0 或 pin1 接上線路，就會影響 Arduino 燒錄的運作，導致程式傳輸失敗。

Experiment 2: SPI protocol

1. SPI frame waveform



2. Frame content (Fill the blank with 0 or 1)

	Bit7 (MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (LSB)
1st frame	0	1	1	0	0	0	1	1
2nd frame	0	1	1	0	1	0	0	0

3. The frequency of the SCK is equal to 131.5k Hz

SPI protocol 介紹與電路圖分析：

SPI 序列週邊介面（Serial Peripheral Interface）以主從方式工作(master and slave)，這種模式通常有一個主(master)設備和一個或多個從(slave)設備，需要至少 4 條連接線路，或是 3 根（單向傳輸時），分別是 SDI（數據輸入）、SDO（數據輸出）、SCLK（時鐘）、CS（片選）。

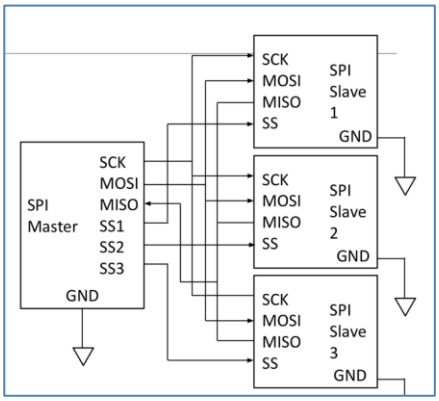
（1）SDI – SerialData In,串列數據輸入。

（2）SDO – SerialDataOut,串列數據輸出。

（3）SCLK – Serial Clock,時鐘信號，由主設備產生。

（4）CS – Chip Select, 從晶片是否被主晶片選中的控制信號，由主設備控制。

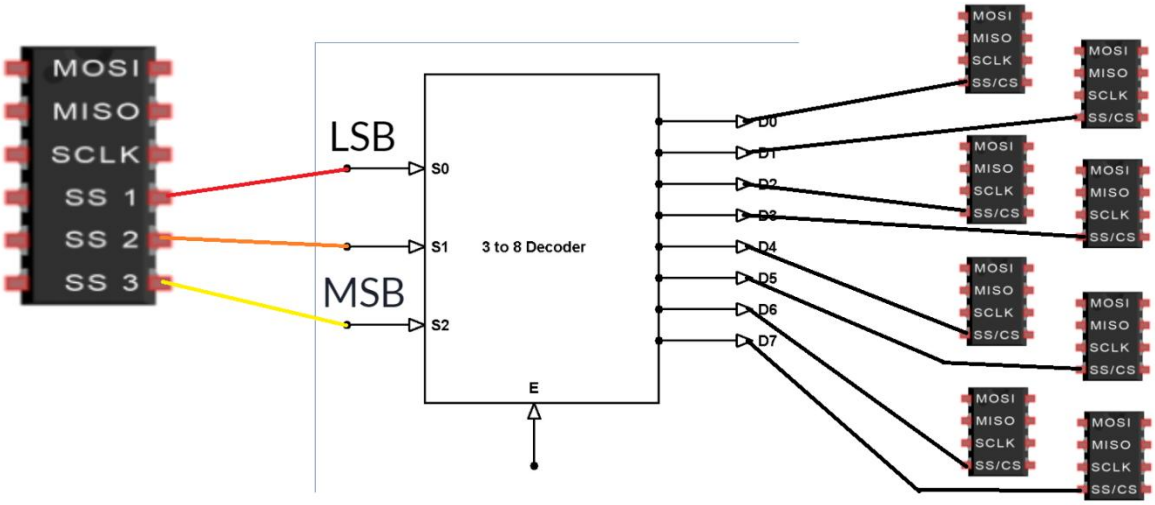
由右方的線路圖可以看出，Master 的 CS 端口數量，必須與 Slave 的數量一致，這是因為當 Master 的 SDI 和 SDO 接對接著多個 Slave 的 SDO 和 SDI，因此要透過每個 Slave 獨立對應到 Master 上的 CS 接腳之電位來決定這時候的 Master 是在與哪一個 Slave 進行訊號的溝通。



Question:

請用一個 3-to-8 decoder, 設計只用 3 支 pin 產生 8 個 SS 訊號，控制 8 個從機的電路?

以下為簡易設計圖，其中當 Master 要呼叫第 1 個 Slave 時，只需要將 SS1~SS3 的訊號拉低，就會產生 000 的 bits 訊號，經由 Decoder 轉成 0 號，而當要呼叫第 2 個 Slave 時，只需要將 SS1 拉高，SS2 和 SS3 拉低，就會產生 001 的 bits 訊號，經由 Decoder 轉成 1 號，以此類推會有 000 001 010 011 100 101 110 111 八種組合，對應 1~8 號 Slave。

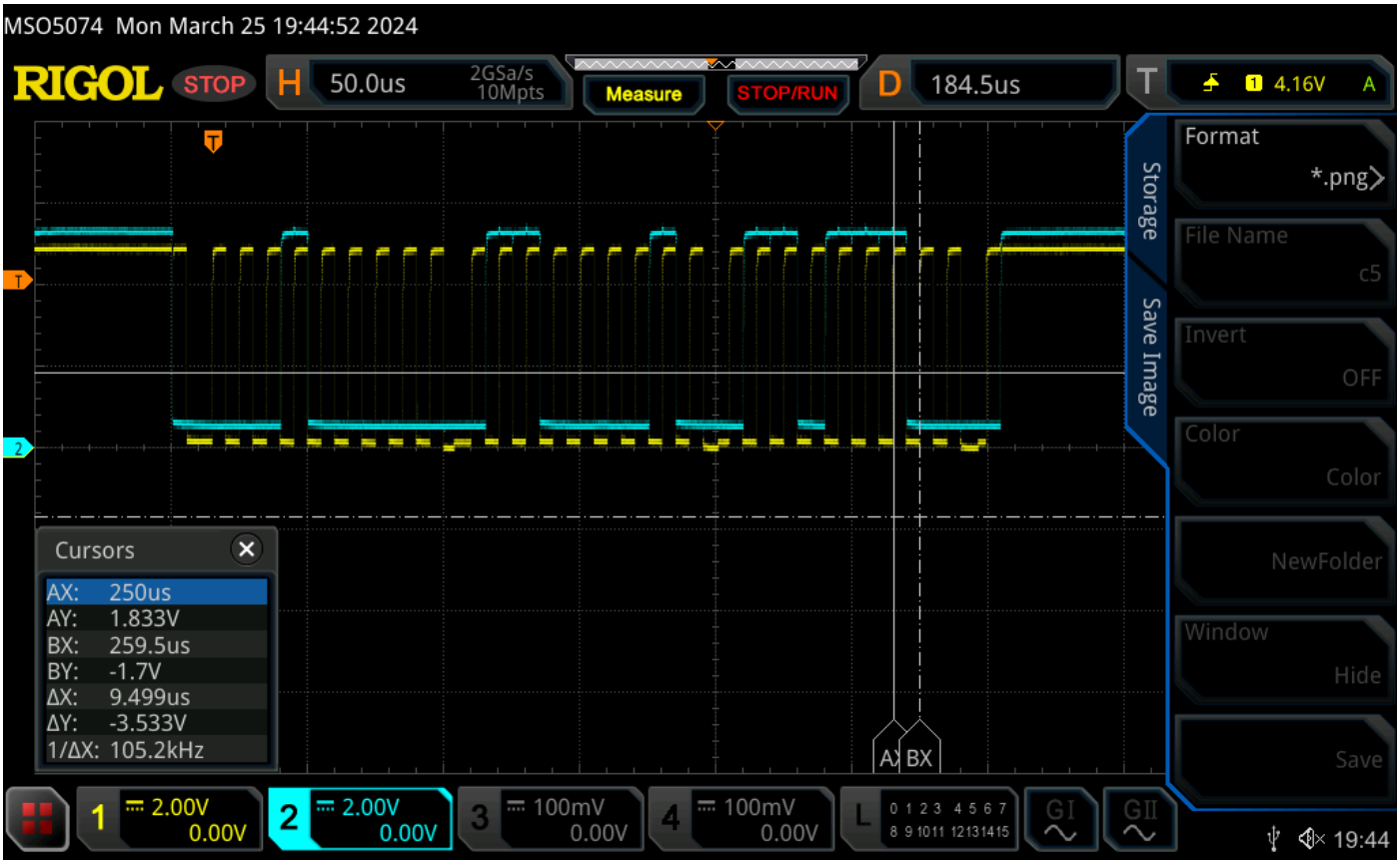


為什麼 IC2 的實驗不能跟 UART 或 SPI 一樣只使用一個 arduino 板?

這是由於 I2C 使用的 ACK 來進行 Slave 的定位與溝通，因此需要使用不同位置的裝置才可以使 ACK 訊號正常運作，不然都是自己的話可能會出現位置錯誤的問題。

Experiment 3: I2C protocol

1. I2C frame waveform

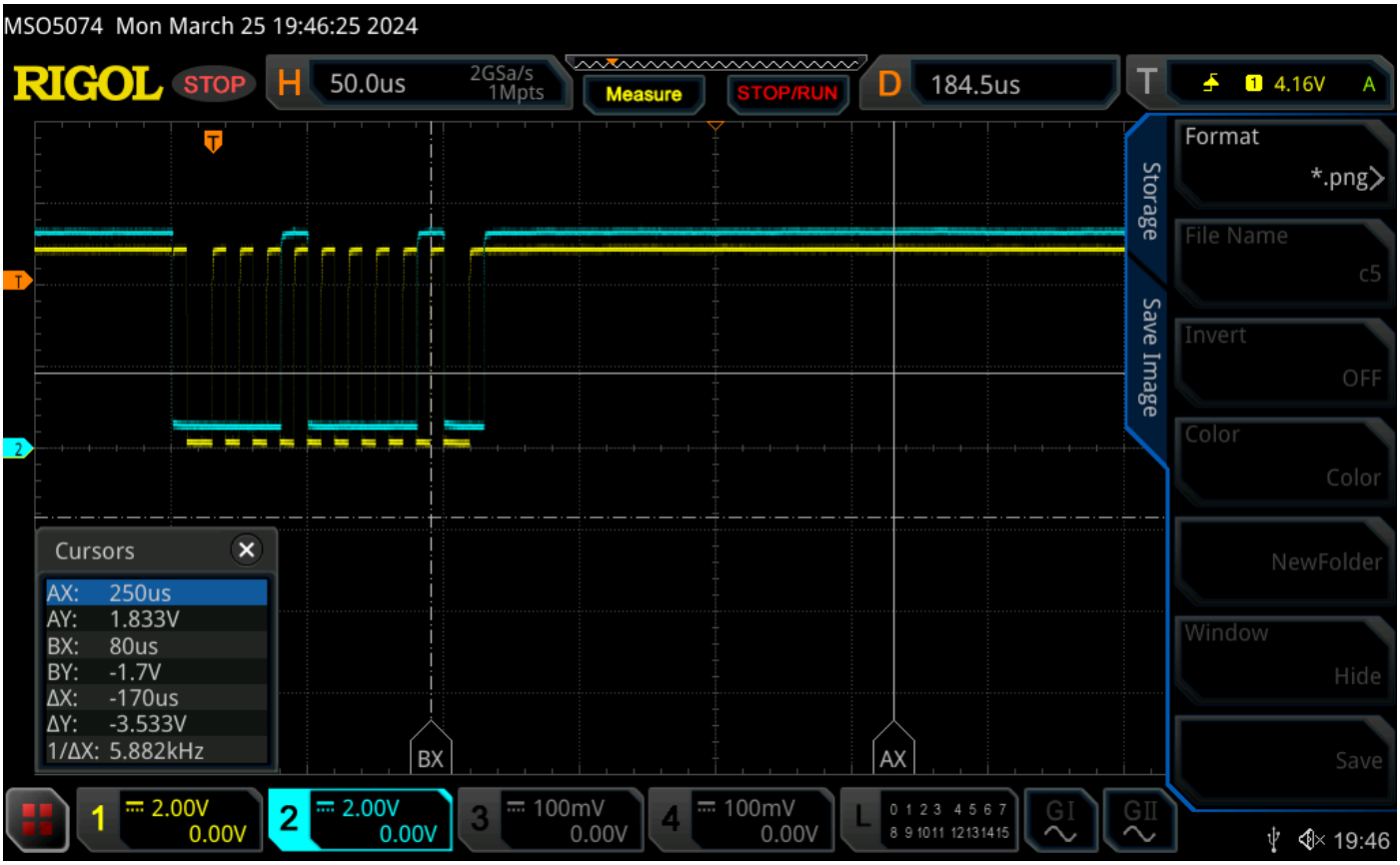


2. Frame content (Fill the blank with 0 or 1)

	Address6	Address5	Address4	Address3	Address2	Address1	Address0	Read /Write	ACK
1st frame	0	0	0	1	0	0	0	0	0
	Bit7 (MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (LSB)	ACK
2nd frame	0	1	1	0	0	0	0	1	0
3rd frame	0	1	1	0	1	1	1	0	0

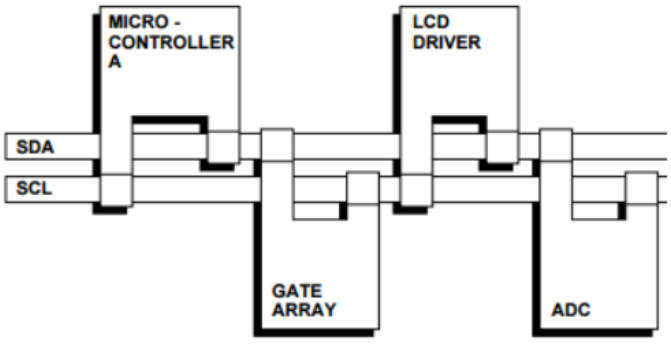
3. The frequency of the SCL is equal to 105.2k Hz

若將 Slave 拔掉，可以看到在 ACK 訊號那邊就變成了高電位，代表 Master 沒有找到對應的 Slave，如下波形圖:



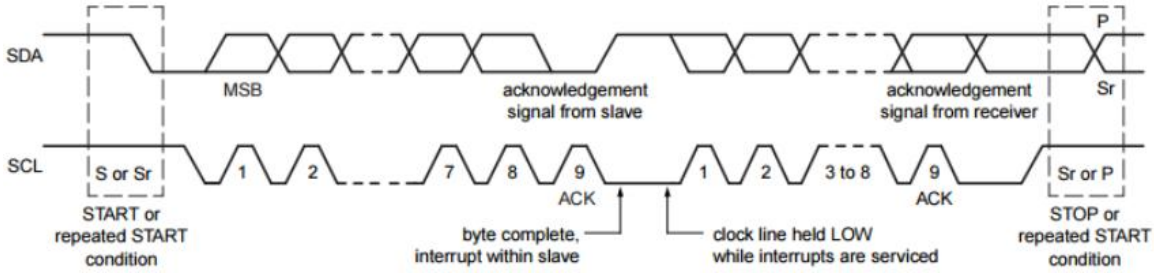
I2C protocol 介紹與電路圖分析：

I2C(Inter-Integrated Circuit)，唸做 I-square-C，為 NXP(前身為飛利浦)開發的通訊協定，主要用來做為 IC 之間的通訊，其非常適合拿來設定 IC 初始值，或是 IC 之間的控制訊號傳輸，又由於 I2C 只用兩條線通訊 SDA(data)/SCL(clock)，因此非常省空間，其架構圖如下左圖:

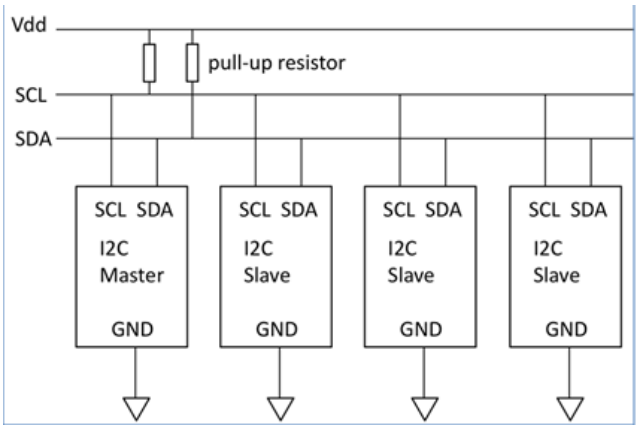


I2C 是Master/Slave架構

I2C 傳送資料也是透過序列式的傳輸，只需要兩條線，一個叫做 SDA 專門用來送資料，另一個叫做 SCL 是用來傳 clock，其資料格式如下圖，依序是由 start condition 所開始，然後開始傳資料，最後 stop condition 結束。所謂 start condition 就是這兩條線的某種狀態的組合可以拿來認定傳輸的開始。



本實驗介紹圖(如下)為 1 對 3 的簡易配置圖，因為 I2C Protocol 的 SDA 都是接在同一條線上，通過位置去區分不同的裝置，因此 slave 會去分辨 Master 所傳送的位置是不是指到自己的位置，如果是自己的位置就會去做回應，所以不需要因為 Slave 數量增加而增加 SC 腳位。



Question:

為甚麼 I2C 接線要使用上拉電阻?

上拉電阻被用來確保信號線（SDA 和 SCL）在不被主設備或從設備主動拉低時，保持在一個已知的狀態，也幫助防止信號線因為受雜訊干擾而導致 floating。

可以將實驗的 PNP BJT 改成 NPN BJT 嗎?為什麼?

可以，因為在這個實驗中只是當開關使用，而不管是 npn 還是 pnp 都存在啟動電壓，因此都可以作為電壓相依的開關，但要值得注意的是 Vcc 與 Gnd 可能需要更改方向。

Experiment 4: 2-digit 7-segment display

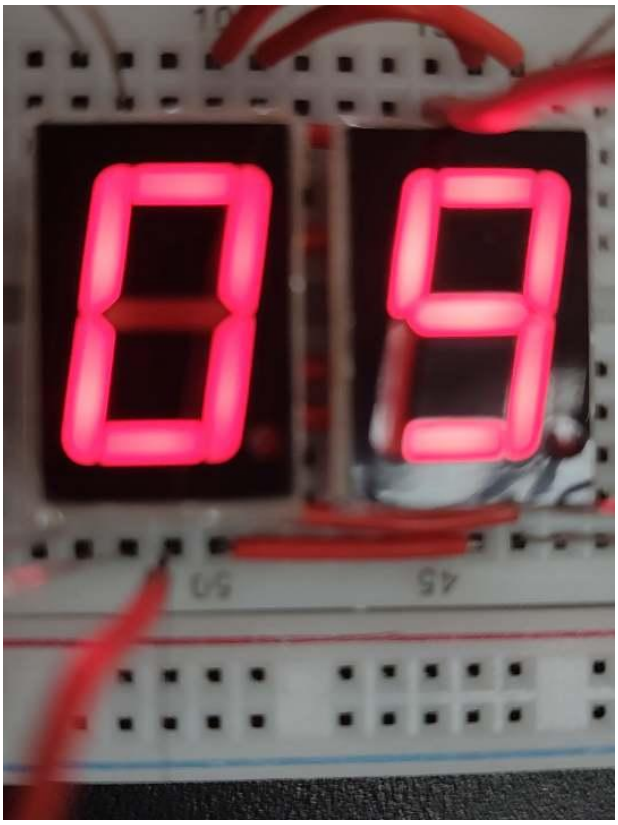
The sketch of your design: (copy from the Arduino IDE window and paste here)

```
int pin[8] = {13,12,11,10,9,8,7,6};
int left = 5;
int right = 4;

void setup()
{
   for(int i = 0;i < 8;i++)
   {
      pinMode(pin[i],OUTPUT);
   }
   pinMode(left, OUTPUT);
   pinMode(right, OUTPUT);
   digitalWrite(left, HIGH);
   digitalWrite(right, HIGH);
}

void loop()
{
   display(true, 0);
   display(false, 9);
}

void display(bool digit, int num){
   if(degit){
      digitalWrite(left, HIGH);
      digitalWrite(right, LOW);
   shownum(num);
   }
   else{
      digitalWrite(left, LOW);
      digitalWrite(right, HIGH);
   shownum(num);
   }

   delay(1);;
}
```

```
void showSevenSeg(byte A, byte B, byte C, byte D, byte E, byte F, byte G, byte P)
{
  digitalWrite(pin[0], A);
  digitalWrite(pin[1], B);
  digitalWrite(pin[2], C);
  digitalWrite(pin[3], D);
  digitalWrite(pin[4], E);
  digitalWrite(pin[5], F);
  digitalWrite(pin[6], G);
  digitalWrite(pin[7], P);
}

void shownum(int num)
{
  switch(num)
  {
    case 0:
      showSevenSeg(0,0,0,0,0,0,1,1);
      delay(1);
      break;
    case 1:
      showSevenSeg(1,0,0,1,1,1,1,1);
        delay(1);
      break;
    case 2:
      showSevenSeg(0,0,1,0,0,1,0,1);
        delay(1);
      break;
    case 3:
      showSevenSeg(0,0,0,0,1,1,0,1);
        delay(1);
      break;
    case 4:
      showSevenSeg(1,0,0,1,1,0,0,1);
        delay(1);
      break;
    case 5:
      showSevenSeg(0,1,0,0,1,0,0,1);
        delay(1);
      break;
    case 6:
```

```
            showSevenSeg(0,1,0,0,0,0,0,1);
                delay(1);
            break;
        case 7:
            showSevenSeg(0,0,0,1,1,1,1,1);
                delay(1);
            break;
        case 8:
            showSevenSeg(0,0,0,0,0,0,0,1);
                delay(1);
            break;
        case 9:
            showSevenSeg(0,0,0,0,1,0,0,1);
                delay(1);
            break;
        case 10:
            showSevenSeg(1,1,1,1,1,1,1,1);
                delay(1);
            break;
    }
}
```

Take a photo of your display content.



電路分析：

實驗四的電路使用兩個七段顯示器，並且 Vcc 的訊號透過兩個 BJT 設計的開關來決定 on 或 off，透過足夠快的切換閃爍，讓視覺上看起來是兩個數字同時在亮，也就是利用視覺暫留來達成好似一起亮起的情況。

Question:

試寫出 **void display(bool digit, int num)** 的自訂函數?

```
void display(bool digit, int num){
   if(degit){
      digitalWrite(left, HIGH);
      digitalWrite(right, LOW);
   shownum(num);
   }
   else{
      digitalWrite(left, LOW);
      digitalWrite(right, HIGH);
   shownum(num);
   }

   delay(1);;
}
void shownum (bool digit, int num){
   switch(num)
   {
      case 0:
         showSevenSeg(0,0,0,0,0,0,1,1);
         delay(1);
         break;
      case 1:
         showSevenSeg(1,0,0,1,1,1,1,1);
            delay(1);
         break;
      case 2:
         showSevenSeg(0,0,1,0,0,1,0,1);
            delay(1);
         break;
      case 3:
         showSevenSeg(0,0,0,0,1,1,0,1);
```

```
            delay(1);
        break;
    case 4:
        showSevenSeg(1,0,0,1,1,0,0,1);
            delay(1);
        break;
    case 5:
        showSevenSeg(0,1,0,0,1,0,0,1);
            delay(1);
        break;
    case 6:
        showSevenSeg(0,1,0,0,0,0,0,1);
            delay(1);
        break;
    case 7:
        showSevenSeg(0,0,0,1,1,1,1,1);
            delay(1);
        break;
    case 8:
        showSevenSeg(0,0,0,0,0,0,0,1);
            delay(1);
        break;
    case 9:
        showSevenSeg(0,0,0,0,1,0,0,1);
            delay(1);
        break;
    case 10:
        showSevenSeg(1,1,1,1,1,1,1,1);
            delay(1);
        break;
  }
}
```

實驗心得:

這次實驗介紹了三種通訊協議，之前在網路程式設計的時候也有接觸到相關的知識，而這次透過電路與示波器觀察的方式，實際將傳輸的訊號透過波形圖讀出，也讓我對數位訊號的傳遞更加了解，並且這次的助教介紹的非常用心，因此也讓我更快可以上手此次的實驗，總而言之，通訊示電機系的一個核心知識，因此我很高興可以在實驗課了解到這方面相關的知識。

實驗結論:

實驗一介紹的 UART 通訊協議，重點為其非同步的特性，因此需要設定好彼此的 bund rate 才可以正確傳遞訊號。

實驗二介紹的 SPI 通訊協議，重點為其同步且一對多的特性，但由於每多一個 Slave 都會是 Master 多一個 SC 接腳，因此不省空間。

實驗三介紹的 I2C 通訊協議，重點為其體積小(只需用到兩個接腳)且同步的特性，其使用 ACK 特性也完美的克服了 SPI 在一對多時面對的問題

實驗四使用 BJT 當作開關來控制 7 段顯示器，重點為 BJT 當作開關來使用，以及使用 function 簡化實驗所使用的 Arduino code。

參考資料:

實驗一:

中文百科。 (未知) UART。 檢自: https://www.newton.com.tw/wiki/UART (2024/03/31)

ROHDE&SCHWARZ. (Unknown) Understanding UART. retrieve from: https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html (2024/03/31)

實驗二:

中文百科。 (未知) SPI。 檢自: https://www.newton.com.tw/wiki/SPI (2024/03/31)

StrongPiLab. (2016/06/26) I2C-協定用法原理簡介-晶片溝通的橋樑。檢自: https://www.strongpilab.com/i2c-introduction/ (2024/03/31)

實驗三:

中文百科。 (未知) I2C 匯流排(I2C)。 檢自: https://www.newton.com.tw/wiki/I2C (2024/03/31)