無線網路基礎與應用 2015

TEAM07 Lab 1 Report

CamCom

蘇彥 高偉傑 黃子軒 楊孟翰 林蔚城

April 3, 2015

1 Introduction

" 可見光通訊" 為無線通訊技術的一種，利用燈光以肉眼無法辨識的高速明暗閃爍信號來傳輸資料數據。此次 Lab 1 利用簡易的裝置做出可見光通訊的環境，傳送端以 Zigduino 電路板控制三色 LED 燈的傳輸方式，藉由 8 PSK 上得到啟發，將 ASCII code 8 bits 分為兩個 4 bits，以 16 種頻段分別表示 0000 到 1111，再使用兩顆單色 LED 燈分別傳輸 4 bits ，以此方式傳達我們的訊息。接收端將 iphone 5s 鏡頭 (30 fps) 錄製的 LED 影片分解成多張 frame，以 OpenCV 分析計算每張 Frame 因 Rolling Shutter 產生出的黑色條紋週期，配合傳送端制定的頻率反推得到解碼後的字串訊息。

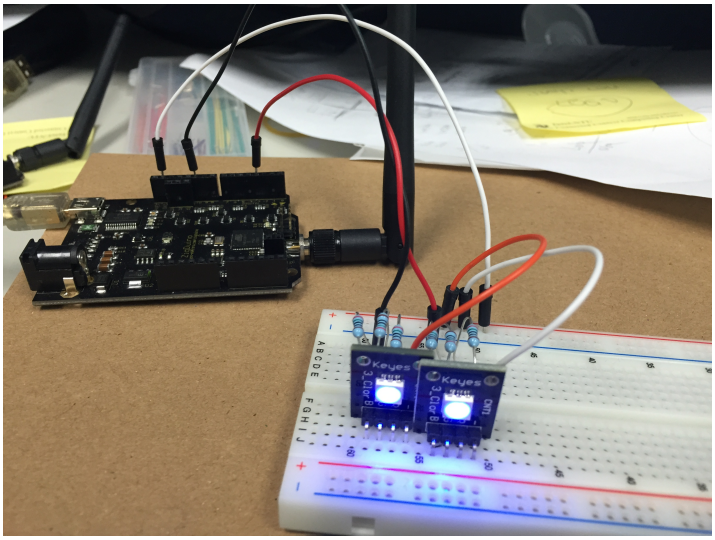

Figure 1: Zigduino 與 LED 配置圖

2 Implementation Method

- **TRANSMITTER ：**

  使用兩個 LED ，以不同頻率的閃爍來代表不同的訊息，其中一個 LED 使用 Timer 1，對應到 pin 9、10、11，另一個使用 Time 3，對應到 pin 3、4、6，Prescalar 皆設為 1，並且都是使用 CTC mode。
  ASCII code 一個字元為 8 bits，我們拆成前半 4 bits 和後半 4 bits，前半使用 Timer1，後半則是使用 Timer3。因 4 bits 共有 16 種可能，我們使用 16 種頻率來表示，經過與接收端的搭配測試，決定將頻率的範圍定在 350 Hz 到 650 Hz，讓 Receiver 端容易辨識，而每種頻率的間隔為 20 ms，也就是 350 Hz 代表 0000、370 Hz 代表 0001 ..... 以此類推，平時沒有資料在傳輸時的頻率則是設在 700 Hz 左右。根據公式，Frequency = 16 M / prescaler / OCRnA /2
  我們可以反推出 OCRnA 應該設定的值，舉例來說：假設我們要傳 A ，ASCII code 以二進位表示的話會是 0100 0001，因此將左邊的 LED 頻率設為 $350+4\times20=430$ Hz，右邊設為 $350+1\times20=370$ Hz。OCRnA 可由 16 M / 1 / Frequency / 2 算出分別約為 18604 和 21621，使用 delay 讓頻率持續一段時間後恢復成 700 Hz。

OCRnA = 16M / prescaler / Frequency / 2	Frequency(HZ)	Binary	Decimal
12307	650	1111	15
12698	630	1110	14
13114	610	1101	13
13559	590	1100	12
14035	570	1011	11
14545	550	1010	10
15094	530	1001	9
15686	510	1000	8
16326	490	0111	7
17021	470	0110	6
17777	450	0101	5
18604	430	0100	4
19512	410	0011	3
20512	390	0010	2
21621	370	0001	1
22857	350	0000	0

Figure 2: 頻率、傳送的資料和所對應的 OCRnA 表格

- **RECEIVER** ：

我們以 C++ 搭配 OpenCV 來辨識 LED 燈光所發出之訊號。透過相機 Rolling Shutter 的特性分析出燈光的頻率後，反推出傳送端所輸入的字元。分析單張 frame 之頻率的方式是窮舉所有週期的長度 (frame 中一條黑線加一條藍線的垂直長度)。首先將每一列每一個 pixel 的 RGB 值加總，接著去算出圖中任兩列值相差後取絕對值的總和，此值越低代表該長度是週期的倍數之可能性越大。以 Fig.3 中之方框為例，如果此方框在圖中的任一處其上緣與其下緣之總和最低，表示這方框之高度很有可能是此圖中週期之倍數。所以藉由找尋此函數之第一個低點即為一個週期長，我們依此來算出其所相對應的字元頻率。為了加速 data rate，我們裝了兩個 LED，也就是左邊的燈發送前 4 bits 的頻率，右邊的燈發送後 4 bits 的值，因此照片的圖案都會有兩個光源且各自發送不同頻率同 Fig.5。然而，原先

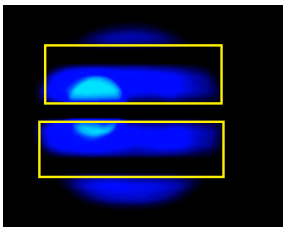

Figure 3:

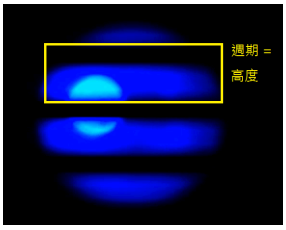

Figure 4:

只有一顆燈時燈源在影片中之比例夠大，但當擴充後單一燈源之畫面比例只剩下原先之四分之一，使得接收端面臨的挑戰有三個：
(1) 辨識正確率大大降低 (2) 所能接受之低頻門檻大幅提高
(3) 須找 Bounding Box。

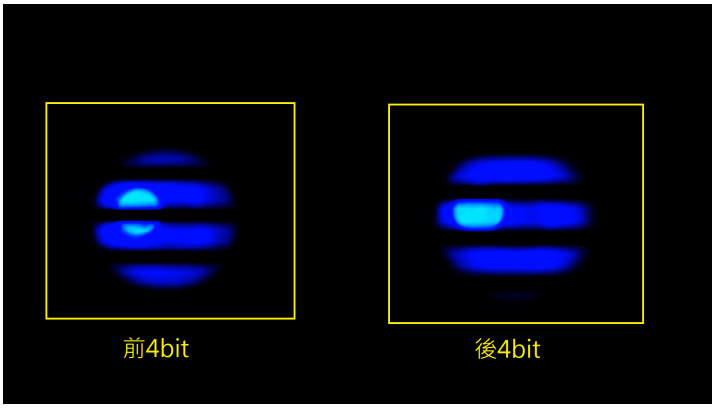

Figure 5:

程式執行流程如下:

(1) 讀取 frame(main)

(2) 找尋圓心 (find_2balls) 及其 BoundingBox (只有在一開始高頻時)

(3) 計算高度間隔之差並回傳頻率 (count_stripe)

(4) 判斷此 frame 是否該丟棄 (isDropable)(frame 中單一顆上下出現不同頻率)

(5) 輸出頻率在合理範圍中且不連續出現之字元。

3 Problem & sloved

- **TRANSMITTER**

  (1) 為何選擇使用 **frequency** 作為編碼方式？
  解決方法**:** 測試過使用 **PWM mode** 改變顏色及明亮後，發現皆是肉眼可辨識到的方式，需要更多技術上的突破，因此先選擇使用 **frequency** 作為我們的編碼方式

  (2) 一個字元為 8**bits**，要切成四個 2，兩個 4，還是一次傳 8**bits?**
  解決方法**:** 首先考量一次傳 **8bits** 是最好的情況，但是本次作業使用頻率來編碼，如果 1 次傳 8**bits,** 需要將頻率範圍切成 256 份 **(**或是 128 份，因為 **ASCII** 實際上只用到 7**bits)**，受限於 **Rx** 端在高頻率 **(**黑影太細**)** 以及低頻率 **(**黑影太粗**)** 無法有效辨識，因此退而求其次降到用兩個 4**bits** 傳輸

  (3) **delay** 若設在 30 時，一個字元需要用 4 個 **frame** 表示 **(**兩個 **frame** 傳送資料，兩個 **frame** 作為間隔**)**，但即使將 **delay** 降低，**frame** 數仍為 4 個 **frame**，一個 **frame** 傳送資料，但間隔卻變成 3 個
  解決方法**:** 經測試後發現為要印出資訊的 **Serial.print** 函數所造成。因此註解掉後便解決此問題。

  (4) 最佳的 **delay** 值為多少**?**
  解決方法**:** 在與接收端進行來回測試後，發現 **delay** 設為 33.5**ms** 不但可以將 1**byte** 限制在 1 個 **frame** 中，並且在大部分的測資下達到 100**%** 的準確率。

- **RECEIVER**

  (1) 一開始拍攝時，頻率的條紋十分不明顯 **Fig.**6，因此程式難以分辯。
  解決方法**:** 先讓手機對著亮光對焦，再近距離拍攝 **LED** 燈，即可明顯分辨出來 **Fig.**7。

  (2) 在偵測圓的時候，會被淺藍色光源所影響 **Fig.**8，導致找到錯誤的圓心，進而誤判圓的位置造成錯誤輸出。
  解決方法**:** 只在高頻率時，偵測圓心位置。

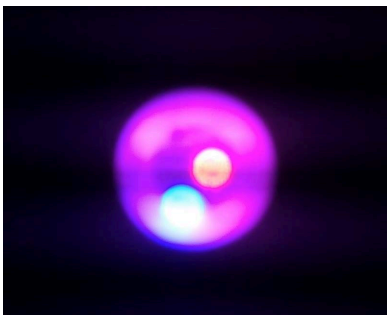

Figure 6:

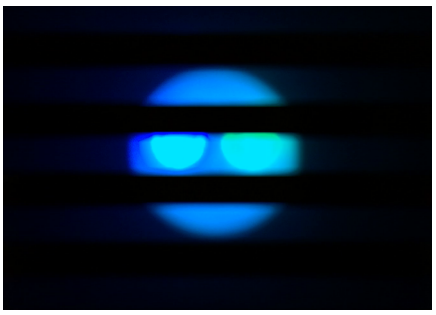

Figure 7:

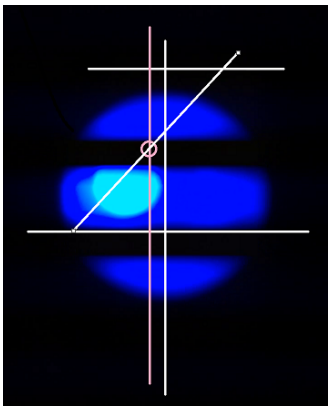

Figure 8:

(3) 傳輸端的頻率設定太高時，會導致條紋不明顯而產生錯誤輸出 Fig.9**，又設定太低時，也會導致條紋太粗而讓程式誤判 Fig.**10**。**
解決方法: 將頻率範圍設定在 350 **Hz 至** 650 **Hz。**

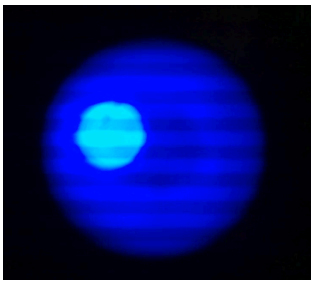

Figure 9:

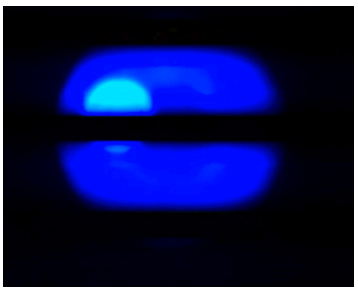

Figure 10:

(4) 當從一字元變成另一字元時，頻率也會有所不同，可能同時出現在同一張 **frame** 中 **Fig.**11，導致頻率判斷錯誤。
解決方法**:** 增加程式碼**:** 由中間往上掃一半，再由中往下掃一半，去判斷兩者頻率是否相同，不同的話則丟棄。

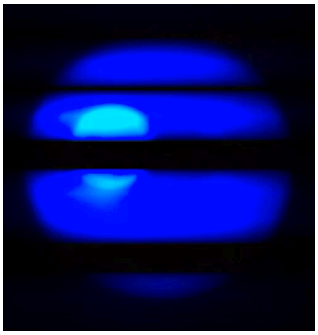

Figure 11:

(5) 當我們算出任意 **frame** 中的週期時，將之乘以 30**(**手機的 **fps)** 應當要等於傳輸端設定的頻率，但發現始終都差了 1.63 倍，後來探究原因是手機雖然為 30 **fps**，但其實一張相片並沒有真的拍滿 1/30 秒，因此每個算出來的頻率都會呈現一定倍數的差距。
解決方法**:** 頻率統一再乘上 1.63 倍使之相等。

4 Results

傳輸端輸入" abcdegfhijk " Fig.12，接著錄製影片，並在接收端去解碼，可得到結果 Fig.13。

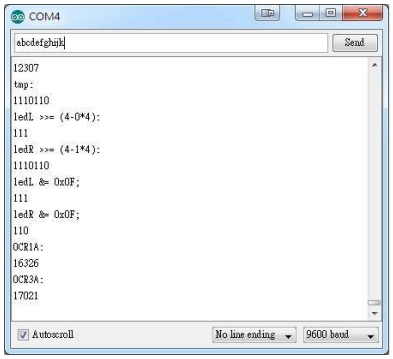

Figure 12:

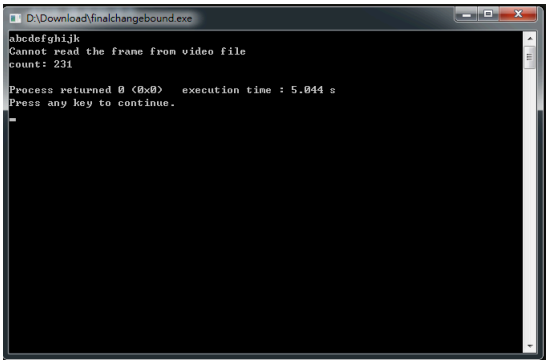

Figure 13:

傳輸速度:
由於拍攝手機是 iphone5s ，其照片都是 30 fps ，所以一張 frame 花了約 0.03 秒，每張 frame 表示 1 byte，因此可算出傳輸速度逼近 30 bytes/s 。

正確率:
經由反覆測試後，觀察到在正常一般情況下程式解碼正確率能保持在 95% 以上，唯在某些對於硬體進行特殊操作時 (如發光時間過久導致過熱等)，則正確率有可能降至 85% 左右。

5 Work Division

- **RECEIVER 端**

  林蔚城: Main Coder & Debugging
  黃子軒: 2nd Coder Testing & Debugging
  楊孟翰: Testing & Debugging & Discussing

- **TRANSMITTER 端**

  蘇彥: Method Design & Coding & Testing
  高偉傑: Detail Implement & Coding & Testing

- **REPORT**

  楊孟翰: Latex **整理排版**
  大家一起討論 Report **內容**

6 Reference

1. https://www.youtube.com/watch?v=zaq3iptX5Zc
2. http://opencv.org
3. http://www.cmlab.csie.ntu.edu.tw/ jsyeh/wiki/doku.php?id=%E8%91%89%E6%AD%A3%E8%81%96%E8%80%81%E5%B8%AB:%E6%95%99%E7%A0%94%E7%A9%B6%E7%94%9F%E5%AD%B8opencv