國立台北大學資訊工程學系專題報告

智慧路燈系統 2.0

專題組員:李岳烜、李國均、陳澔偉

專題編號:PRJ-NTPUCSIE-104-007
執行期間:104 年 9 月 至 105 年 6 月

1. 摘要

隨著交通越來越發達，鋪的路越來越多，國人又越來越重視自身的用路安全，從高速公路到平面道路到各個大街小巷內，以往部分區域是不會設置路燈的，但伴隨著各種交通意外、行人夜歸被騷擾的事件的發生，目前在臺灣已經可以看到就連巷子內都會設置路燈了。然而，路燈的狀況以及報修卻是相當不方便的一件事。

本計畫旨在改善原先學長姐專題製作過的「智慧路燈系統」，以降低成本和提升支援度為主要目標，將原先使用的 beaglebone 換成 arduino 接在路燈上，並使用 Power Line Communication(PLC)的方式進行訊息傳遞，如此只要在有電的地方都能傳遞訊息。同時，我們加入了 ios 版本的 app，讓維修人員想要使用 app 時，不再侷限於 android 手機。

關鍵詞:智慧路燈、降低成本、Power Line Communication(PLC)、arduino、ios

2. 簡介

我們在每盞路燈上接上一個 arduino，每 10 盞路燈會接上一個 beaglebone，其功能為節點控制(node controller)，路燈之間皆是以 Power Line Communication(PLC)的方式串接[圖 1]，只要在有電力的地方便能使用，而每個區域會有一個 beaglebone 負責區域控制(zone controller)。此系統讓路燈可以自己回報自己的狀況，同時維修人員在安裝路燈時，可以使用 app 對路燈進行經緯度定位，並依照我們的編號編碼方式輸入到資料庫中，增加了許多便利和安全性。

若是有需求要對特定區域開關路燈也可從 app 進行任意不規則形狀選取或單擊開關，維修頁面也提供了導航功能，讓維修人員能夠更方便了解路燈狀況並更快速抵達現場維修。

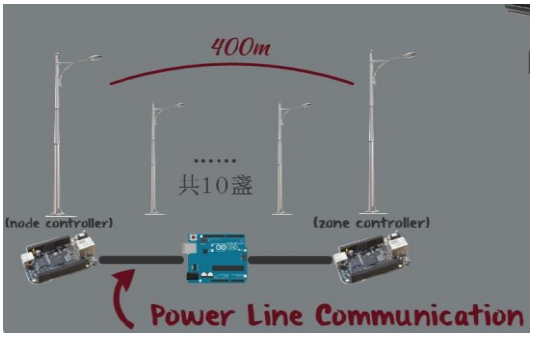


圖 1: 系統架構示意圖

3. 專題進行方式

3.1 Protocol 介紹

為了能夠處理足夠大的路燈數量，再加上訊息傳遞時的需要，必須要有一個完整的路燈 id 協定方便日後開發，而我們的格式是:(distinct id, zone id, segment id, node id)，第 1 個表示地區，第 2 個表示區域，第 3 個和第 4 個則為一個路燈的 tree，表示該區域路燈的結構，每個 id 皆為 4byte，以 16 進位方式表示，例如:(00,00,A0,10)。[圖 2]

- For Example, ID:

Distinct	Zone	Segment	Light
新北市 三峽區	NTPU	Tree	

圖 2: 路燈 ID 協定 示意圖

我們定義區分 arduino 跟 beagle bone 的方式是在 Light id 有差別，每個 beaglebone 都算是一個母親，並且擁有 9 個 beaglebone 孩子，在 ID 的配置上，Light ID 的第二個字元是'0'者為 beaglebone，意即 Light ID 為"00"、"10"、"20"、"A0"都是 beaglebone，而每個 beaglebone 的第一個字元都是母親 beaglebone 的第一個字元，意即如果母親 beaglebone 的 Light ID 為"20"，孩子 arduino 就是"2*"，*只有 1~9。

3.2 系統架構圖

Server 目前架設在臺北大學社會科學院 7F，而我們會以距離 server 最近的一盞路燈將它裝上 beaglebone，並使其成為 zone controller，再依序路燈接上 9 個 arduino 後，會再接上一個 beaglebone 為 node controller 然後再同樣接上 9 個 arduino，每個 beaglebone 都會負責管理其後方的 9 盞燈，監控它們的好壞狀況，並且轉傳開/關燈的訊息封包給它們，像個父母在管理他的 9 個小孩一樣，然而小孩之間不能互相溝通，只能和他們的父母溝通[圖 3]。若是任何系統上的路燈發生故障，則會依序回傳直到回傳到 zone controller，再由它轉交給 server。

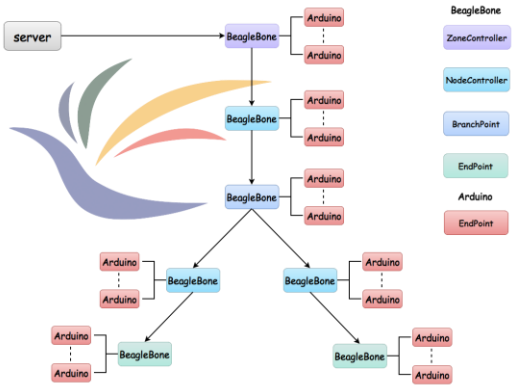


圖 3: 硬體系統架構圖

部分路口會遇到分岔狀況，而該 beaglebone 就會成為 branch point，其功能必須能轉傳給 2 條分支的 node controller。

3.3 轉傳與偵錯機制

轉傳狀況可以分為兩種 beagle bone 對 beaglebone & beaglebone 對 arduino，任一狀況當發生傳輸失敗時，我們的系統會採取「反覆連線 5 次」的方式進行偵錯，如果 5 次都失敗，beaglebone 對 beaglebone 的狀況會將封包轉交給下一個 beagle bone(距離 800 公尺)，並把錯誤訊息往前回傳至 server；beaglebone 對 arduino 的狀況則會直接將錯誤訊息往前回傳至 server，沒有再轉傳至下一個的問題。

3.4 板子運作模式

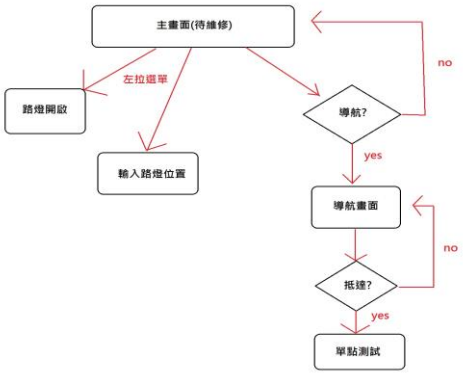
3.4.1 Beaglebone

插上電後會主動執行路燈程式，路燈程式會先去看設定檔裡這個點是哪類型的點，然後設定燈、網路，然後進入迴圈等待收到資料，收到資料會先判定是否是給自己的，是的話執行開關燈或偵測；如果不是給自己的，會檢查資料裡是不是要給自己身後的孩子 arduino，是的話會傳給特定 ID 的 arduino；如果都沒有的話，就會依傳來的方向往下一個點傳

3.4.2 Arduino

插上電後也會主動執行路燈程式，會一直等待母親 beaglebone 傳資料，收到資料後會截取需要的參數，去執行開關燈或偵測，執行完後回到迴圈等待下一次收資料

3.5 軟體架構與簡介



我們在軟體方面做了許多的改善與優化，整理如下:

(一)待維修:

(1)新增通知功能，只要有新的壞掉路燈就會通知維修人員。

(2)導航加入計算路程距離和時間，讓維修人員更能規劃路線和時間。

(3)加入單點測試，不必再另外切到開燈頁面做測試，app 會直接跳出該路燈狀態並提供測試。

(二)路燈開啟:

(1)新增不規則形狀圈選繪圖模式，並加入新的射線法演算法

(2)樣式修改，將單點開和範圍圈選整合在同一頁

(三)輸入路燈位置:

(1)新增個人位置定位

3.5 主要困難與解決之道

硬體方面由於是承接學長作品，所以在交接時常常有問題必須得問學長，再加上 trace 別人的 code 較花時間且沒有註解，解決方法就是不斷地問學長。而且硬體還有困難是板子本身有不少問題，一開始 beaglebone 的外接 wifi 晶片腳位資料不見，索性換成 wifi dongle，但原本的 beaglebone 的 OS 又無法使用，所以最後重灌了另一種 OS；arduino 板也蠻多問題的，因為我們不是買原廠的板子，我們是買副廠的來作實驗，到後期時 arduino 的網路擴充板還莫名壞掉，最後只好買原廠的網路擴充板來試。

軟體方面，ios 端由於使用的是最新的 swift 而非 obj-c，所以語言會不斷地被更新，網路上的資源也相對的少，甚至有時候一更新，語法的用法就完全不同，就得再找官方文件修改，相當麻煩。

3.7 人員配置與職責 - 時程規劃

起初因為完全沒有 ios 版本，所以我們 3 個人，其中岳烜負責硬體端，剩下國均和澔偉則是學習 swift 來開發 ios，以原先 android 的模樣為設計方向盡力模仿，但後來因為 android 方面也必須有所提升和優化，所以 2 個人便拆開分別為國均負責 ios 和澔偉負責 android。

在寒假時，ios 端基本上已經完全跟上 android 版本，下學期便是不斷討論並修改成我們想要的樣子，並致力於整合軟硬體，同時有空的時間開發網頁。

4. 主要成果與評估

本次專題成果可以分成三個部分，分別為硬體端、伺服器端、行動裝置端。

一、硬體端:

我們以 PLC 方式進行串接，在這方面是以窄頻方式傳輸，其最遠可傳輸距離為 1 公里，我們估計其實際值為 800 公尺，然而為了避免我們的系統中有任何一個環節發生故障，我們限制 beaglebone 之間的距離為 400 公尺，如果傳輸失敗，我們會對該路燈進行傳輸測試 5 次，如果 5 次都失敗，則會轉傳給下一個 800 公尺的 beaglebone[圖 4]，並回傳該路燈的故障訊息給 zone controller，再交由 zone controller 將錯誤訊息傳給 server。軟體端則會收到故障路燈位置，並通知維修人員進行維修。

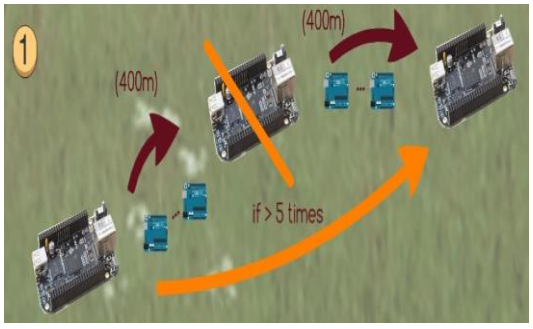


圖 4: 轉傳偵錯示意圖

另外在硬體端我們也降低了大量的成本，原本的系統是全部以 beagle bone 實作，換成 arduino 之後，每組路燈平均可以降低 55%的成本，數量越多則降低越多，如表 1。

系統	第一代	第二代	
電燈總數	430	430	
BBB 數目	430	40	
BBB 價錢	2000	2000	
ARDUINO 數目	0	390	
ARDUINO 價錢	800	800	
總價	860000	392000	
差距			468000
比率			54.419%

表 1:兩代系統價錢差異

二、伺服器端

Server 上存放有許多轉傳/編碼用的程式還有資料庫，資料庫則分為「路燈資料庫」和「動作資料庫」。路燈資料庫負責存放所有路燈的資訊，包含其 ID、經緯度、路燈狀態、是否故障等等。動作資料庫則負責記錄手機發送給 server 的所有要求，目前來說都是開燈/關燈的要求，而 server 上會有一支程式在背景不斷地偵測這個動作資料庫是否有新的資料，如果有則將這些資料打包送給 zone controller，再由 zone controller 轉傳給 node controller 開啟/關閉相對應的路燈，同時 server 上會將這些動作刪除，表示已將要求送出。

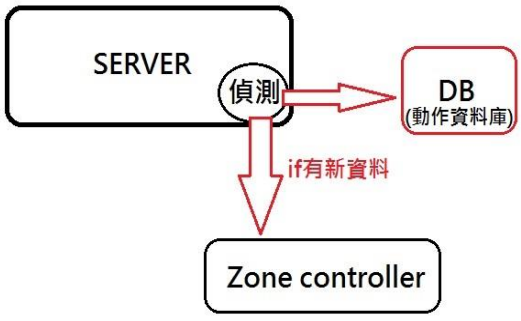


圖 6: 動作偵測與轉傳示意圖

三、行動裝置端:

行動裝置我們除了支援 android 系統，在這版本我們也加入了 ios 版本，軟體功能上主要分為三大功能：待維修、路燈開啟、輸入路燈位置。**「待維修」**頁面會列出所有壞掉的路燈，並且只要有新的壞掉路燈就會提供通知，可以選擇進行導航維修，導航時可以計算路程距離和時間，並可對壞掉的路燈進行單點測試。

圖 7:待維修清單(左) 單點測試(右)

「路燈開啟」我們在此版本加入新的演算法(射線法)，觀察所有路燈和圈選形狀的邊線交點是否為奇數個，此方法可支援各種不規則形狀圈選路燈，如圖 8。

圖中星星符號代表路燈，黑色線為圈選的形狀，我們會從路燈做 2 條射線，分別為 x 軸正向和 y 軸正向，若路燈在圈選形狀外則交點會是「偶數」個；若在形狀內則為「奇數」個。

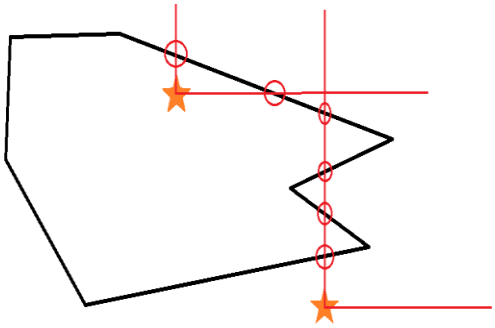


圖 8: 射線法示意圖

同時我們也為此演算法做了實際測試，其測試結果如圖 9。嚴重失誤率之所以為 10%，是因為我們認為在進行區域開燈時，若是不小心遺漏了零星幾盞燈未正確開啟是可以被接受的。

	次數	百分比	總失誤率	嚴重失誤率
完全正確	15 次	75%	25%	10%
未正確開啟	2 次	10%		
邊緣遺漏 1~2 盞燈	3 次	15%		
總試驗次數	20 次			

圖 9: 射線法正確率分析

輸入路燈位置

維修人員在安裝路燈時，必須要有一支程式供他們快速安裝，而本系統最重要的正是路燈的定位，目前我們是採取使用手機 GPS 的方式定位，誤差大約 50 米以內。[圖 10]

圖 10: 輸入路燈位置畫面

四、預期與實際成效差距

原本我們期望軟硬體的支援度可以達到100%，但過程中碰到許多問題，再加上我們三人各自都有自己要忙的事情，所以最後僅能支援單點開啟可以使硬體端開/關燈。希望能夠支援多點開啟，但是多點開啟在server端進行封包打包時，會碰到許多的問題，諸如開的燈擴及不同區域、同時有開燈/關燈的動作等等，牽涉到很多排序及偵錯的問題，硬體面也必須要有較高的技術來支援，所以無法完成。

4. 結語與展望

依我們目前的能力可能沒辦法把這個「智慧路燈系統」做的很完善，但我相信只要持續的改善下去，有一天它一定能夠為市民及路燈維修人員們帶來很大的便利。而我們目前希望未來能夠實現以下三點：

(一)實現校園路燈系統：

目前我們的系統僅為測試狀態，而真正要推廣到市場上，一般來說成本必須降低到一盞路燈300元台幣，然而我們目前成本是800多元台幣，還有很大一段距離，但如果只是校園內的數量，我們認為是可行的，且整體校園路燈結構也較不複雜，實作起來相對容易。

(二)建立雲端平台(商業模型):

我們可以提供系統給需要的廠商，然後幫他們設置一個維修網站，並幫他們定期維護，我們只需要收取維護費用和路燈系統維護費用，基本的維修我們不必干涉，並為廠商個別設立他們的路燈資料庫，讓他們便於使用，而我們也省事。

(三)成本再降低:

承接第一點所說，目前成本仍然過高，若要再推廣出去，勢必要再將成本降低，使用單晶片的硬體。

5. 銘謝

首先要先感謝的是我們的指導老師張玉山老師，多虧有老師每個禮拜跟我們開會並激勵我們去思考更好的解決方式，和我們一起討論，讓我們學習到怎麼和夥伴們一起把一個專題做好。再來要感謝張景棠學長，因為我們是承接學長的作品，讓學長花了不少時間接受我們的問題轟炸，學長仍然很有耐心地為我們解答，除了非常感謝之外，也因為學長的指導讓我們學到了很多。

最後，也要感謝曾經幫助過我們的同學們，有時候問題找不到人問時，一些比較厲害的同學們願意幫忙我們，使我們專題更順利！

6. 參考文獻

[1] Vandad Nahavandipoor,"iOS 8 Swift Programming 錦囊妙計",黃銘偉譯,2015

[2] 蔡明志,"學會Swift程式設計的18堂課",碁峰出版,2014

[3] Kevin Tatroe, Peter MacIntyre, Rasmus Lerdorf,"PHP程式設計",林耕溥譯,2014

[4] Martin Evans, Joshua Noble, Jordan Hochenbaum, "Arduino完全實戰手冊", 王冠勛譯, 2014

[5] Simon Monk, "給邪惡天才的30個 Arduino 專題",謝瑩霖,蔡睿烝譯,2014