修 平 科 技 大 學

資 訊 網 路 技 術 系

專題實務報告書

救難自走車

指 導 老 師：麥毅廷 教授

專題製作學生：四技資網四甲 劉懿寬 BN97022

四技資網四甲 李玟毅 BN97024

四技資網四甲 吳冠霆 BN97041

四技資網四乙 李明遠 BN97066

中華民國 100 年 12 月 28 日

目錄

圖目錄

摘要

隨著時代不斷的進步，網際網路在現代人的生活中也越來越重要了，很多行業也開始與網路做結合來提升工作效率與便利性。隨著網際網路的發達，無線網路的發展也有所進步，處處都有供Wi-Fi讓人們使用，而且許多行業開始將無線網路融入其中提升服務品質。

由於電視新聞、報紙時常報導火災的發生，常因為沒辦法第一時間救火，導致人民財產化為烏有，所以本組的構思以救火為主。當災難發生的第一瞬間，我們並無法準確的馬上了解發生的地點， 而且往往發現時也許災難點都發生了一陣子，假如當屋主不再家時，剛好電線走火本來應該能夠進行快速撲滅的，到最後都是要拖到火勢猛烈濃煙四起，才發現才進行通報，這時對屋主來說也許損失慘重。

而本系統的功能就是能夠即時通知救難人員，當有災難發生時，就會觸發到感測器，感測器會馬上透過網路主動通知救難人員，在利用本系統快速尋找出距離災難點最近的救難人員，並通知距離近的救難隊，請儘速前往災難點進行救災，在有限的時間內來降低災難的危害。

第一章 前言

1-1 簡介

由於資訊的進步，網際網路的發展，以至於讓消費者可以在家中就可得到想要的服務。網際網路的發展伴隨著無線網路的興起，讓 Zigbee、藍芽、WIFI 等無線網路，在各種場合下展現出自己優越的便利性，由於無線網路具有:不必佈線、和不受場地影響的優點，可在許多環境下使用網際網路而不受限制，許多業者都開始使用無線網路來取代現有的有線網路。

由於災難不定時的發生，雖然我們無法防範災難，但是我們至少能為發生災難時做應對，當災害發生時且救難人員又無法直接進入災害現場時，這時在災害現場只有無線網路不受限制，只要在災難現場觸動到感應器時，中心控制端會擷取離災難現在最近的救難車的訊號值(RSSI)，然後透過訊號值(RSSI)的大小讓中心端去指派最近的救難車去距離災害現場來救援救援，這樣就可以節省寶貴的時間，省去掉不必要的時間花在尋找災難點，和救災現場，減少時間上的浪費來增加救援的時間，這樣的系統優點是在於錯綜複雜的地形，或是通報者即使不熟悉附近環境都能夠迅速地知道事發地點。

1-2 目的

我們模擬了一套救難自走車系統，利用普特企業公司的機器車與Xbee (Zigbee)無線通訊模組做結合，讓監控端(中心端)來判斷距離失火車最近的救難車，並且指派最近的救難車前往失火端，並在網頁上進行更新動作以完成以下的動作:

1. 當車子失火時會透過震動片的震動來觸發，再透過Xbee (Zigbee)無線通訊模組傳訊息給予中心端。

2. 這時候中心端就會收集附近的救難車所傳的RSSI值不斷的給予中心端讓中心端坐接收的動作。

3. 這時電腦端會來接收中心端所收到的RSSI值的大小，透過比對來進行判斷距離失火端，最近的救難車。
4. 網頁當電腦端進行距離判斷完畢的同時，會將資料在網頁上進行更新讓人也能看到。
5. 當中心端收到電腦端判斷出救難車的RSSI的最大值時，就會送出距離失火車最近的救難車前往指定的地點來進行救援的動作。

第二章 文獻探討

2-1、ZigBee 無線通訊技術

何謂Zigbee?

圖2-1-1為實體Zigbee模組

ZigBee (Xbee)就是一種無線網路模組，主要由ZigBeeAlliance制定，底層是採用IEEE 802.15.4 標準規範的媒體存取層與實體層。主要特色有低速、低耗電、支援大量網路節點、支援多種網路拓撲。

ZigBee協定層從下到上分別為實體層(PHY)、媒體存取層(MAC)、網路層(NWK)、應用層(APL)等網路裝置的角色可分為ZigBeeCoordinator、ZigBeeRouter、ZigBeeEndDevice等三種。支援網路拓撲有Star、Tree、Mesh等三種。

ZigBee是非常受歡迎的2.4GHz XBee模組(Digiformally Maxstream)由於ZigBee很平價、且低功率無線感測網路。這個模組為IEEE 802.15.4 推疊(以Zigbee 為基礎) 及包覆到簡易使用的serial command 設置中。這個模組容許微控制器、電腦、系統、任何東西含有serial port 間非常可靠的及簡單的通訊，支援點對點及點對多點。

特性：

一. 省電、二.可靠度高、三.高擴充性、四.成本低

1. 傳輸速率低、傳輸資料量亦少，所以訊號的收發時間短。在非工作模式時ZigBee處於睡眠模式，而在工作與睡眠模式之間的轉換時間，一般睡眠啟動時間只有 15ms，而設備搜索時間為30ms。透過上述方式，使得ZigBee十分省電，電池則可支援ZigBee長達6個月到2 年左右的使用時間。

2. ZigBee 之MAC 層採用talk-when-ready之碰撞避免機制：當有資料傳送需求時則立即傳送，每個發送的資料封包都由接收方確認收到並進行確認訊息回覆，若沒有得到確認訊息的回覆就表示發生了碰撞，將再傳一次，以此方式大幅提高系統資訊傳輸之可靠度。

3. 一個ZigBee的網路最多包括有255個ZigBee網路節點。若是透過Network Coordinator則整體網路最多可達到6500個ZigBee網路節點，再加上各個Network Coordinator可互相連接，使整體ZigBee 網路節點數目將十分可觀。

IEEE802.15.4/ZigBee是一種相當先進的短距離傳輸技術標準，從家用無線通訊規格HomeRF聯盟中所分出來，ZigBee聯盟成立於2002年，在不同區域有不同的定義的頻段(全球2.4GHz、美國915MHz、歐洲868MH)，2.4GHz ~ 2.4835GHz，全球通用之頻段。ZigBee聯盟成立於2002年，目前全球已有不少廠商加入成為ZigBee 聯盟會員，其成員包括IC設計、家電、通訊設備及玩具等廠商。就像ZigBee的標誌(Logo)所呈現，ZigBee聯盟以感測與控制為主要應用，而定 義出簡單、成本低，又容易實現的無線通訊標準。互相連接，使整體ZigBee網路節點數目將十分可觀。

2-2、ZigBee點對點功能(Point-to- Point)

點對點(Point-to- Point)是802.15.4的ZigBee(XBee)支援主從式或點對點、點對多方式運作(圖2-1)，同時最多可有255 個裝置鏈結，具有高擴充性。主要應用的方向在於家庭裝置自動化、環境安全與控制、以及個人醫療照護等功能，逐漸成為產業共通的短距離無線通訊技術之一。

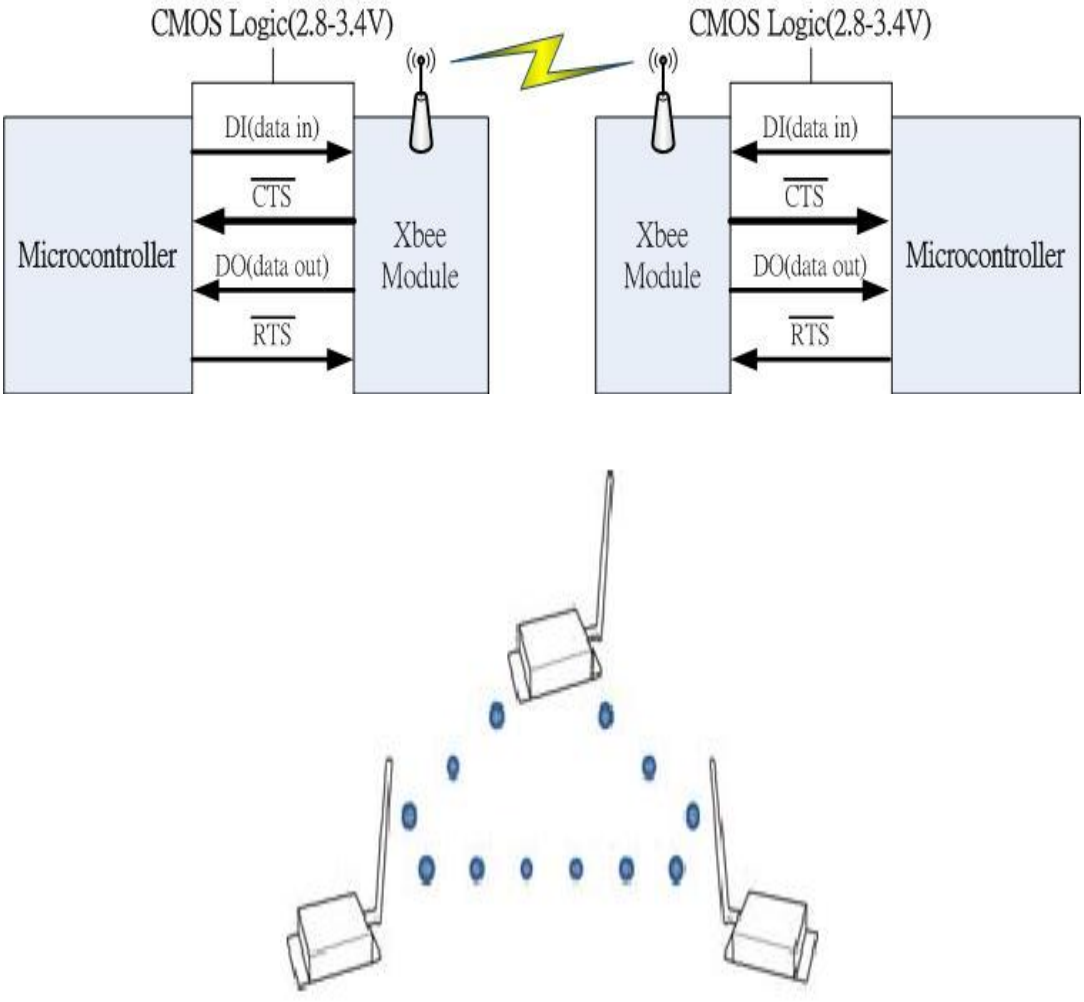


圖2-2-1點對點或點對多的方式

這是一個非常受歡迎的2.4GHz XBee模組(Digiformally Maxstream)並且是一個平價、低功率無線感測網路。這個模組Zigbee為基礎及包覆到簡易使用的serial command 設置中。這個模組容許微控制器、電腦、系統、任何東西含有serial port 間非常可靠的及簡單的通訊。點對點及點對多點往例支援。

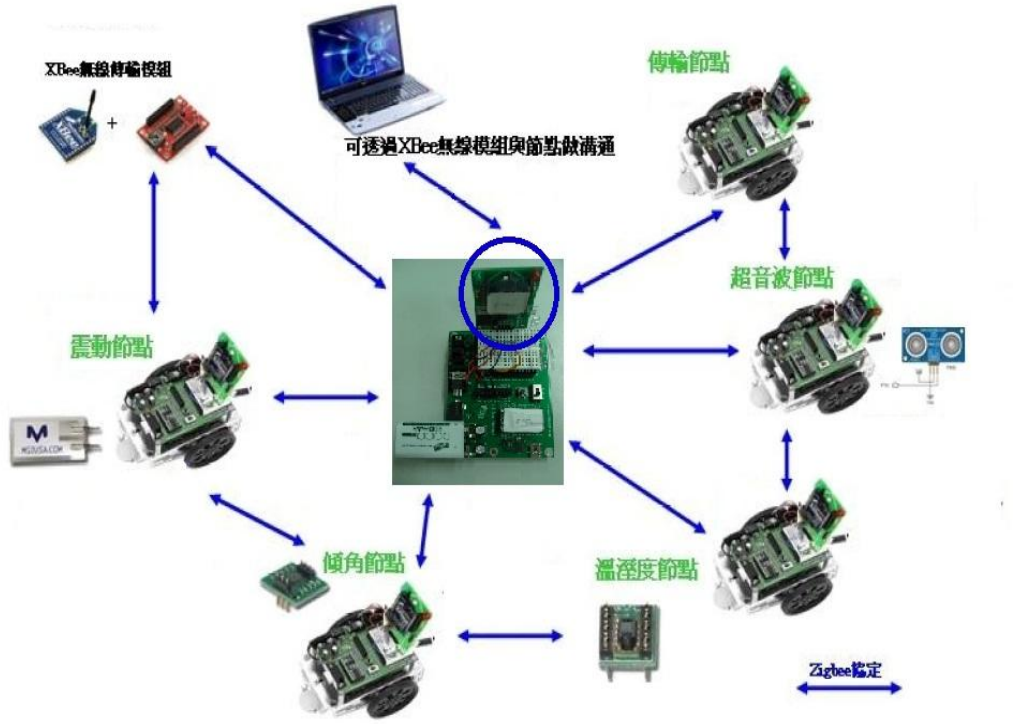


圖2-2-2 Xbee的許多模組，可融入日常生活中

2-3、訊號強度(RSSI)值

訊號強度(RSSI)值:Received Signal Strength Indicator(RSSI)是接收訊號的強度，亦即訊號再傳送訊息當中，所測量的無線訊號強度值，依照強度大小來加以判別所判斷的發射電波與接收端的距離，每個鄰近的感測點和無線感測模組都可以透過發射的無限電波強弱，加以判斷各個接收端的相對位置，以現代的進步來說，RSSI的好處是不需要外加電阻與模組了，因為無線感測模組的晶片也已經有偵測腳位的電路，因此使用RSSI比其他方式省成本以及更加節省電源、容易建立。

但是在RSSI中，主要的有個誤差影響:訊號失真(Signal Attenuation)，在任意空間中是有能量衰減損耗，則傳送端和接收端之間的距離，多重訊號會有不同大小強度和相位傳送到接收端，因此頻率會因為訊號的重建和破壞，而造成頻段選擇的干擾變化，使用展頻接收可以有較佳的接收訊號及降低干擾。

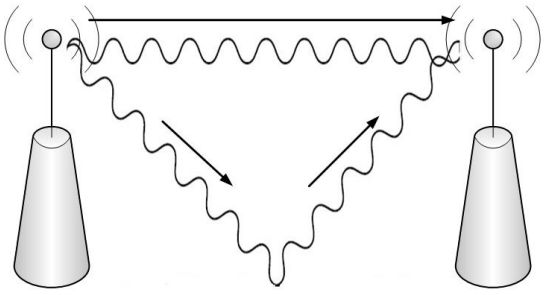
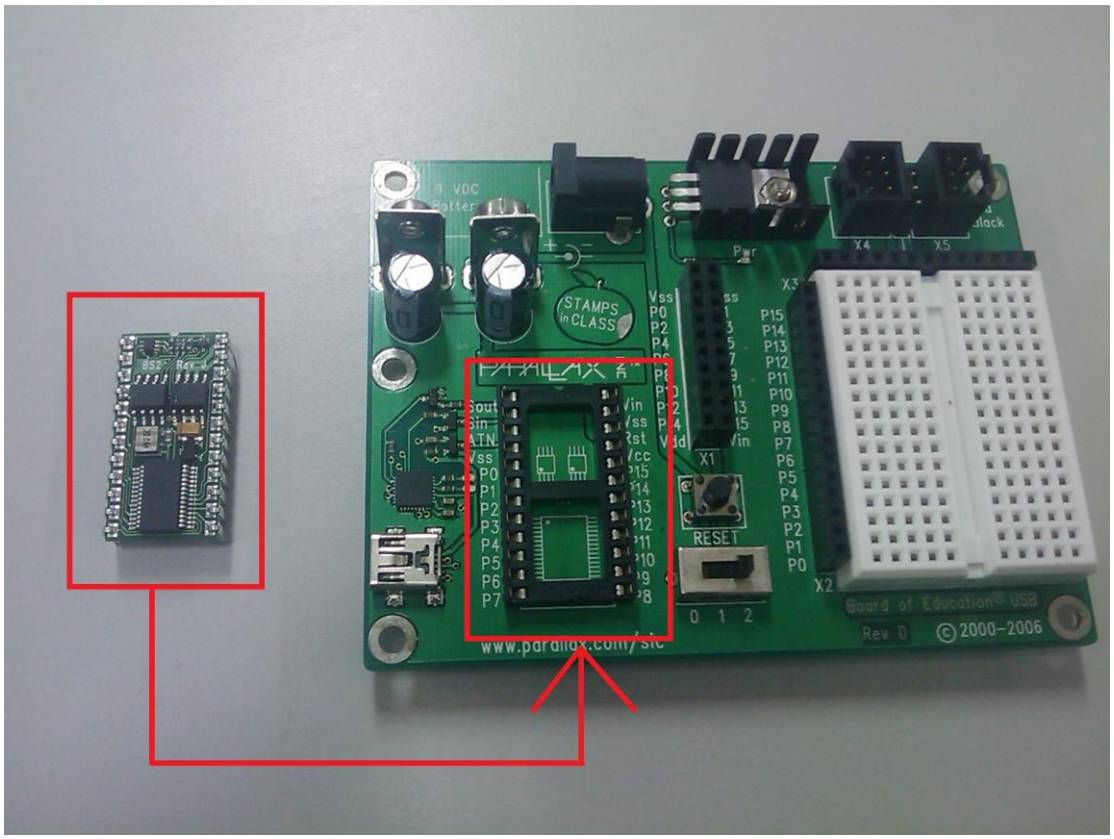


圖2-3-1空間中是有能量衰減損耗

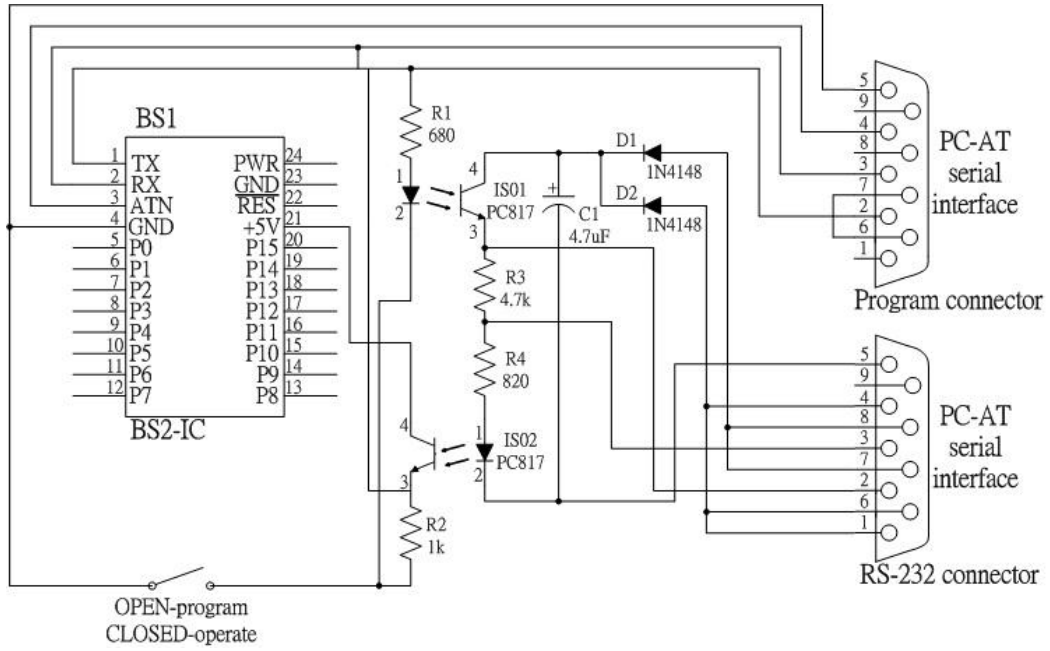
第三章 系統建置

3-1、元件介紹

在這一節中，我們將描述救難系統的架構，它是使用購買來，現成Parallax零件與Zigbee。Zigbee是一種無線模組產品，是由美國加州大學柏克萊分校(UC Berkeley)所設計，並委託Crossbow Technology 公司所生產，而在台灣是以PlayRobot 飆機器人/普特企業有限公司BASIC Stamp，我們使用BASIC Stamp 2微控制器教育套件(USB)來實作我們救難車傳送訊號強度(RSSI值)的部分。



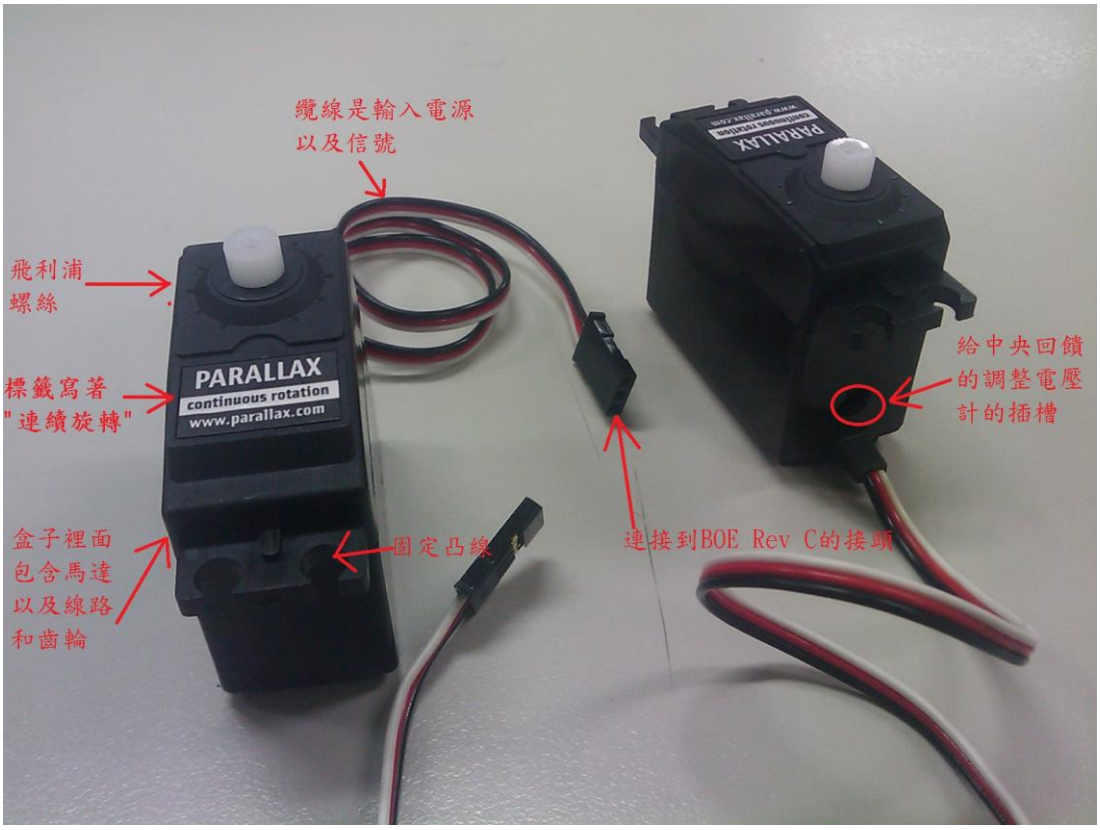
如圖3-1-1 BASIC Stamp 2微控制器電路板



如圖3-1-2 微控制器的電路板設計圖

這種微控制器皆提供計算、通訊與感測的功能。

為了使無線模組具備傳送訊號的能力，我們利用Parallax 公司的Basic Stamp 2 單晶片提供本系統功能的運算能力，並結合Parallax 的步進驅動伺服馬達(RC Servo如下圖3-1-3)。



如圖3-1-3為步進驅動伺服馬達(RC Servo)

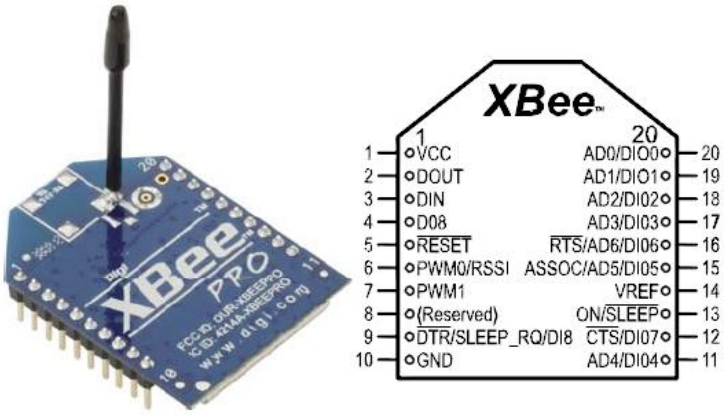


圖3-1-4 ZigBee的內部構造

救難系統其中ZigBee(如圖3-1-4)是最主要的訊息傳送元件，負責提供救難車的通訊能力。而主要的計算能力由BASIC Stamp 2 單晶片提供，它可算從感測電路板(sensor boards)所感測到的資料，以及透過Parallax。

伺服馬達電路板(Servo Controller)控制馬達的轉動。為了增進系統的穩定度，我們使用鋰電池來提供電力來源，分別供給ZigBee、BASIC Stamp2 與步進驅動伺服馬達。

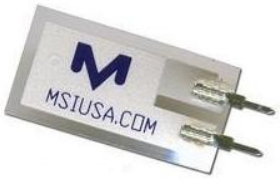


圖3-1-5 震動感知器(vibration)

圖 3-1-5 這種壓電感測器通常用作開關 (switch)或震動(vibration)感知器，應用範圍包括 ：

1. 警告系統 sensor

2. 產品震動/或損壞的警告 sensor (detect vibrate and shock)

3. 加速度計

4. Tab sensor (標籤式感測器)

我們所使用的是飆機器人車公司的 Boe-Bot Robot Kit(通稱 BB) 智慧型機器人教學平台作為模擬建置平台來建置模擬環境，將 BS2 微處理器結合 BOE 多功能電路板並在上面加上各式零件配件組合成 BB 自走車。

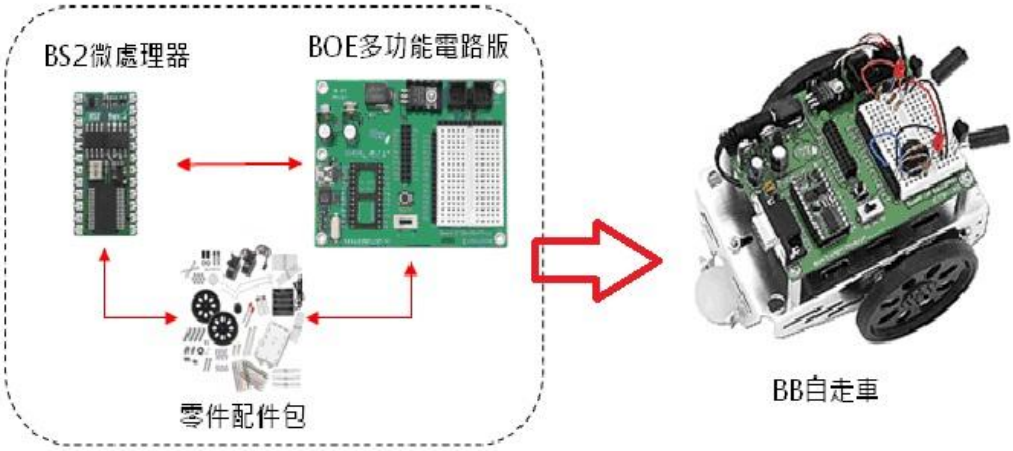


圖 3-1-6 組合完成後的 BB 自走車

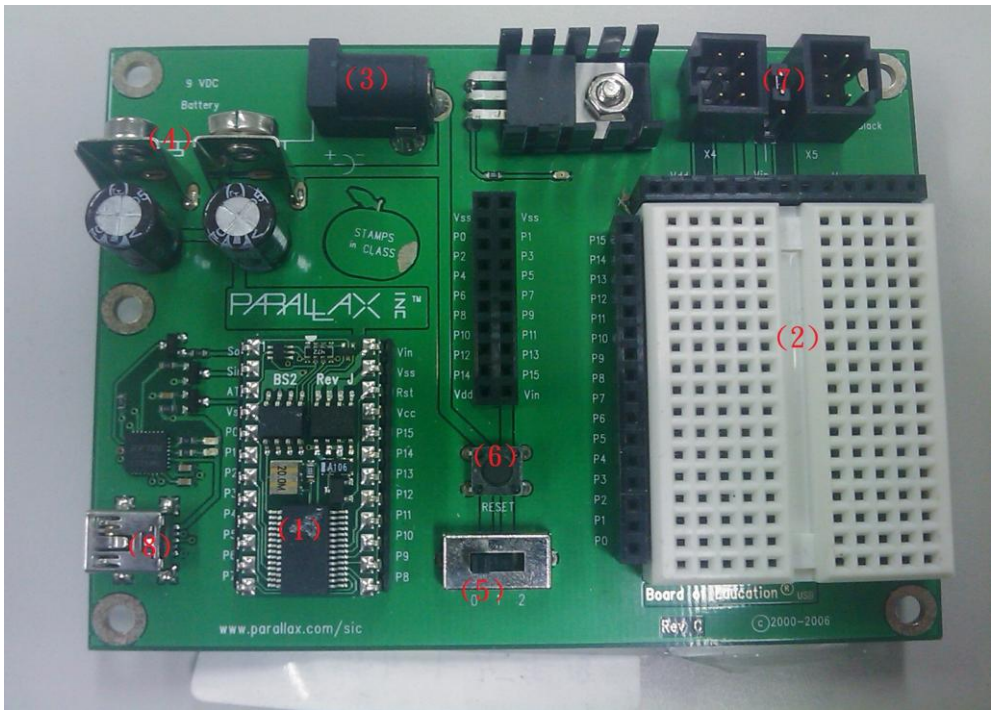


圖 3-1-7 BOE 多功能電路板平面圖

上圖3-1-7為BS2多功能電路板平面圖

(1)BS2微處理器插槽: 插入BS2微處理器的插槽。

(2)麵包板: 插入零件配件。

(3)電源輸入端: BOE多功能電路板所使用的電源為6-9V的電壓，可使用USB或3號電池組供電。

(4)電池插槽: 同為電源輸入端，是用來插入9V方型電池的插槽。

(5)開關: 位於 0 時不供電，位於 1 時只有麵包板供電，位於 2 時 BOE 多功能電路板全面板供電。

(6)重置按鈕(Reset):將寫於 BS2 微處理器上的程式重新運作一次。

(7)3PIN 插槽:此處用來連結 BB 自走車伺服機。

(8)USB 插槽:用來與電腦連結將程式寫入 BS2 微處理器中。

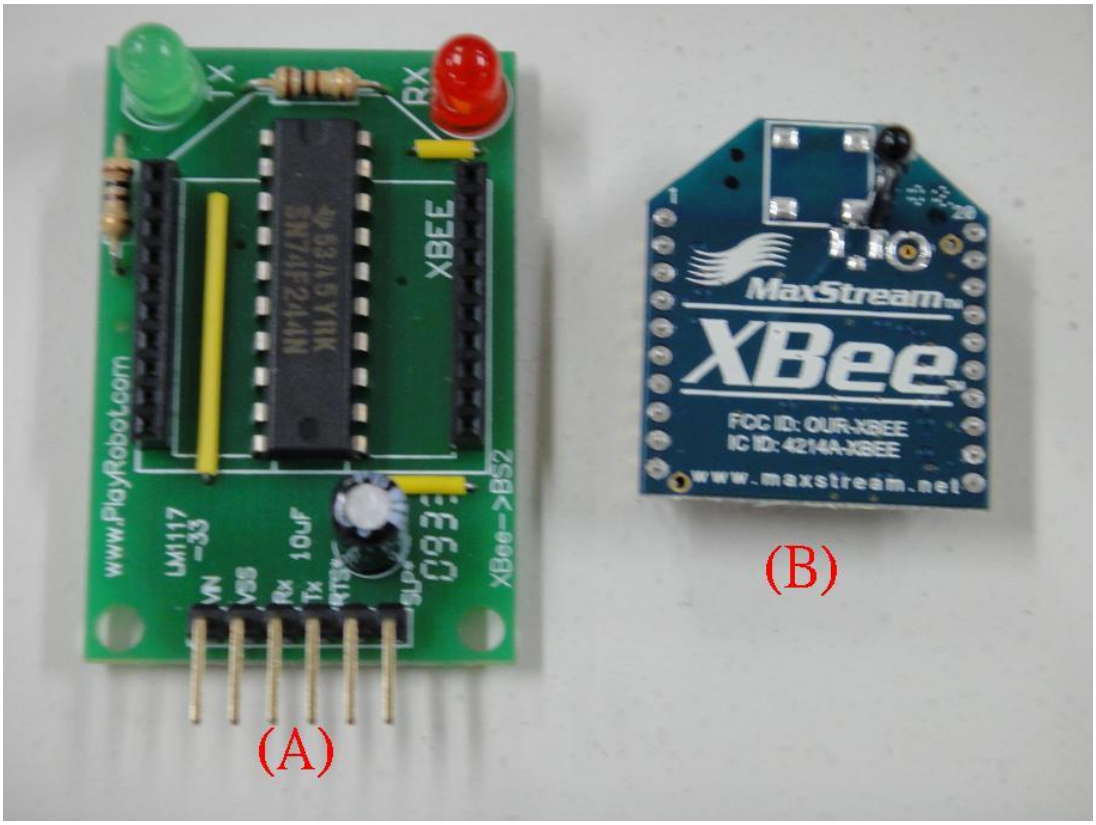


圖 3-1-8 Xbee (Zigbee)無線通訊模組

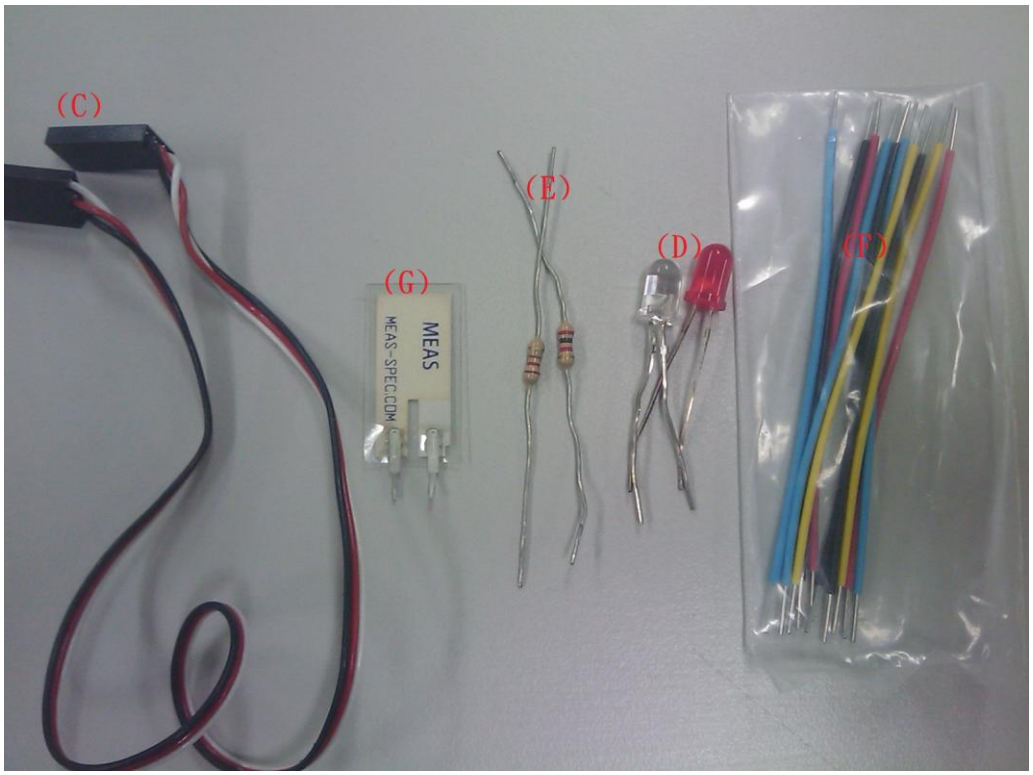


圖 3-1-9 各式零組件

(A) TTL標準介面:讓XBee可以與BOE多功能電路板連結。

(B) XBee:利用ZigBee無線通訊協定與微處理器、電腦與系統之間交換傳輸資料。

(C)3Pin 延長線: 為解決麵包板大小問題所使用的延長線。

(D)LED: 發光二極體 （Light-Emitting Diode，簡稱 LED） 是一種能發光的電子元件。

(E)220Ω & 2KΩ :電阻用來接 LED 和震動感測器。

(F)接單心線線包:用於增加硬體配件所需用品。

(G)壓電式薄膜震動感測器: 這種壓電感測器通用作開關 (switch)或震動(vibration)感知器。

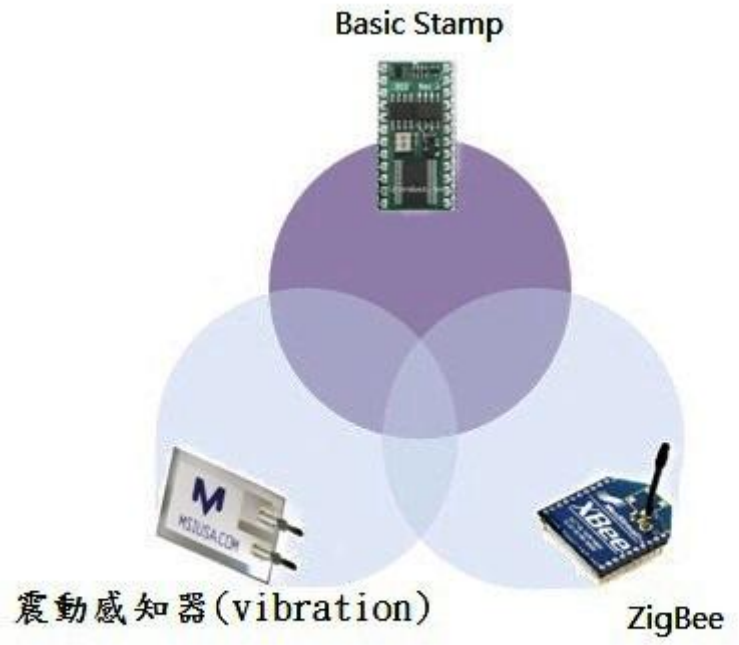


圖3-1-10為本系統主要的主角

透過Basic Stamp 、ZigBee與震動感知器（vibration）做結合，以Basic Stamp所寫的程式，讓震動感知器（vibration）作為一個觸發點，當觸發時ZigBee就會以網路的方式互相傳送訊息來達到我們所要的功能來完成我們的概念。

3-2 環境建置

一、 BASIC Stamp 建置

我們所用的是PBASIC_BASIC Stamp Windows開發編輯器v2.5.2 (支援 Windows 2K/XP/Vista/7)，BASIC Stamp的語法是一種用來對BS2微處理器下達指令的程式語言。

點擊 圖示後來執行，接著會出現下圖3-2-1

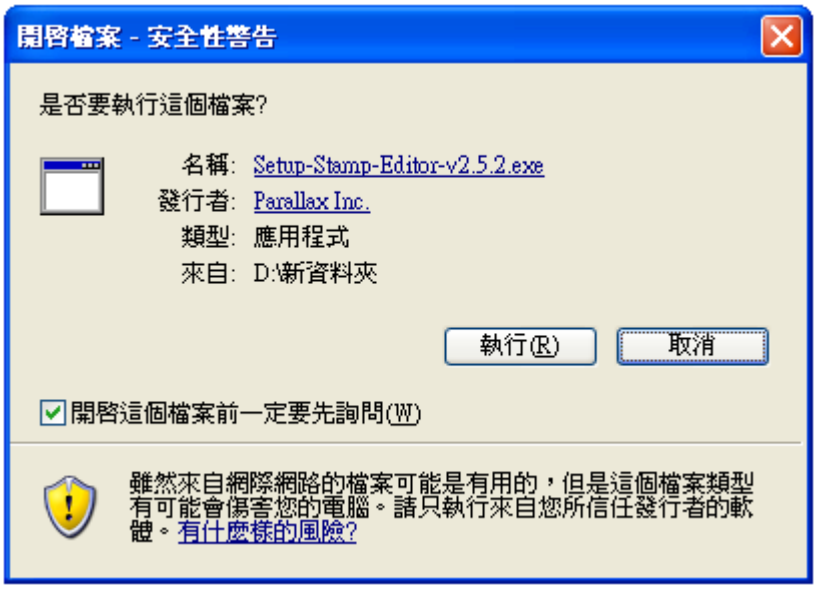


圖3-2-1為”是否要執行此程式 ”

按執行後會直接進到安裝程式起始的畫面

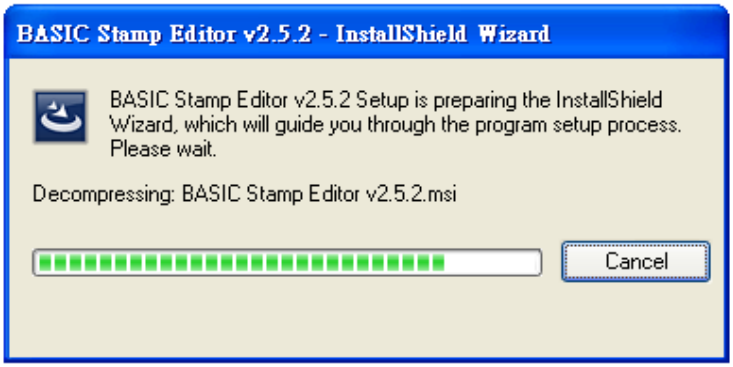


圖3-2-2為安裝程式的進度

圖3-2-3 安裝程式起始畫面

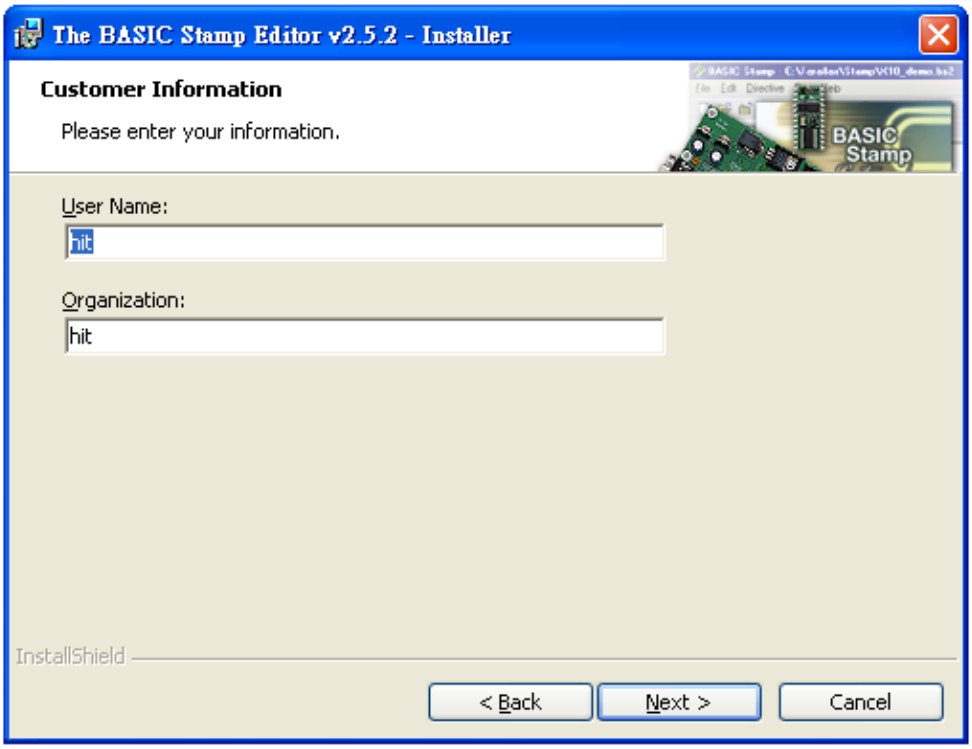


圖3-2-4輸入安裝者身分

如圖3-2-4 所示，User Name是使用者名稱，Organization是組織名稱，預設會去讀取電腦上的設定，無特別需求請直接點選NEXT進入下一步。

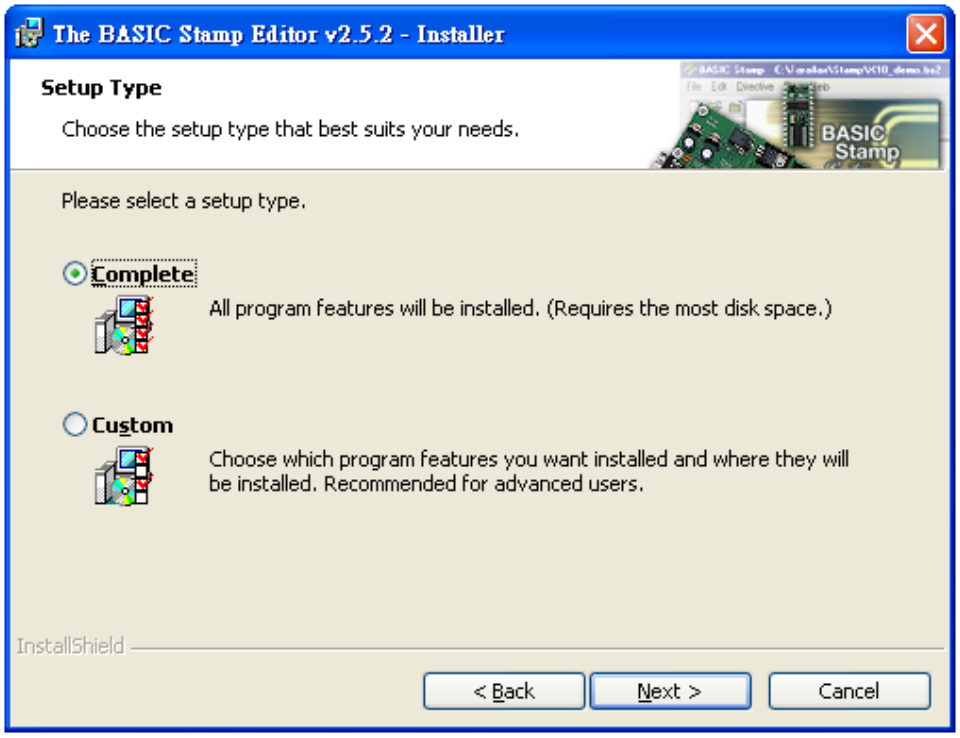


圖3-2-5 一般安裝或進階安裝

接著如圖3-2-5會詢問是否Complete :一般安裝(所有檔案都安裝到電腦上)或是Custom:進階安裝(只選擇想要的檔案安裝)，同上一步驟；若沒特殊需求直接選擇Complete就可點選NEXT往下繼續進行。

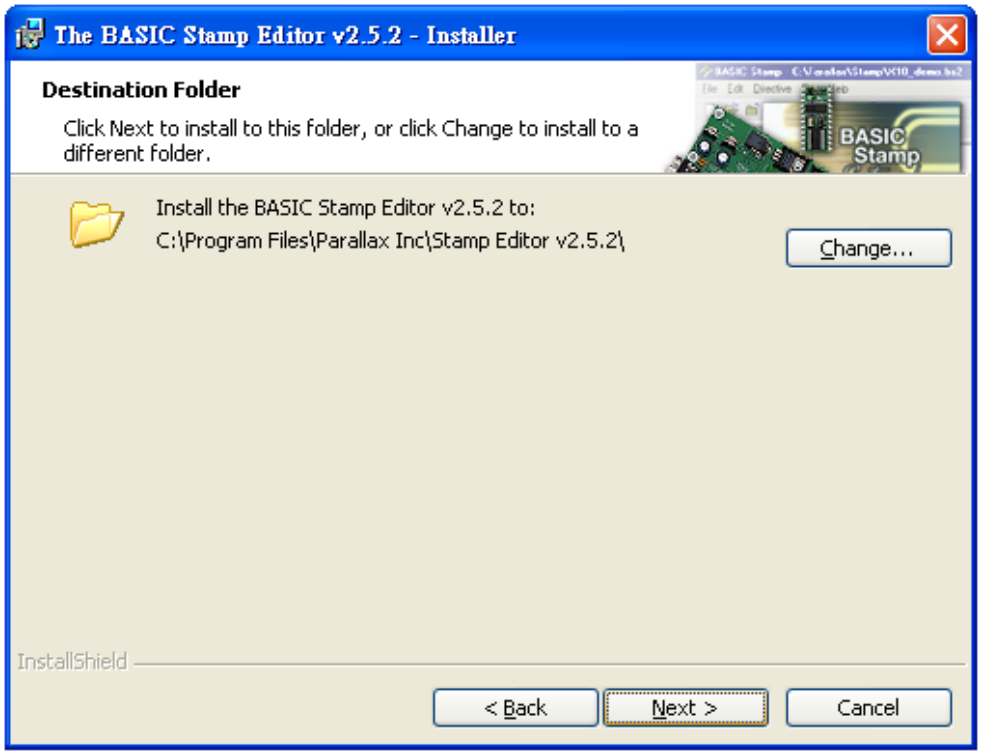


圖3-2-6 指定安裝路徑

接著圖3-2-6會詢問你安裝的路徑，預設在C槽的Program Files底下，可自行選擇安裝位置，選擇好請按NEXT下一步。

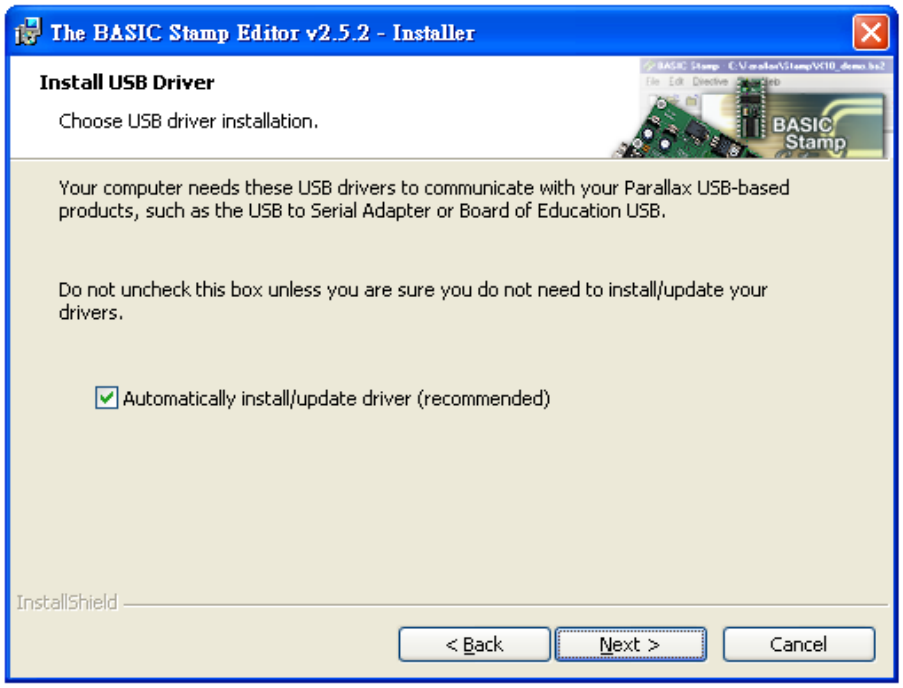


圖3-2-7 是否自動更新

如圖3-2-7這時會詢問是否要自動更新(建議)，讓程式維持最新版本。

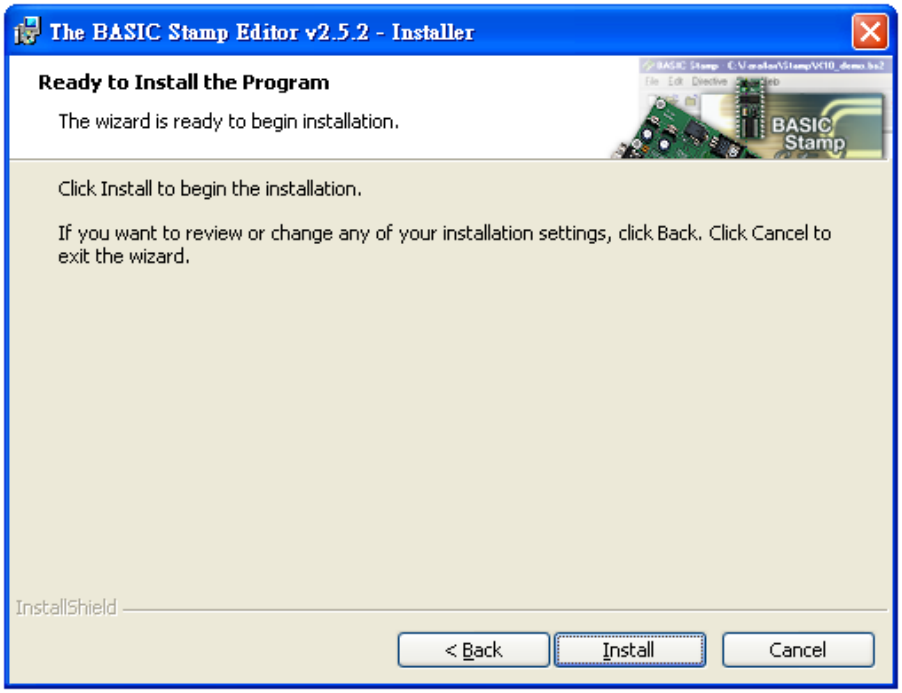


圖3-2-8 確認安裝

上圖3-2-8確認以上選擇無誤之後點選Install就會開始安裝檔案了。

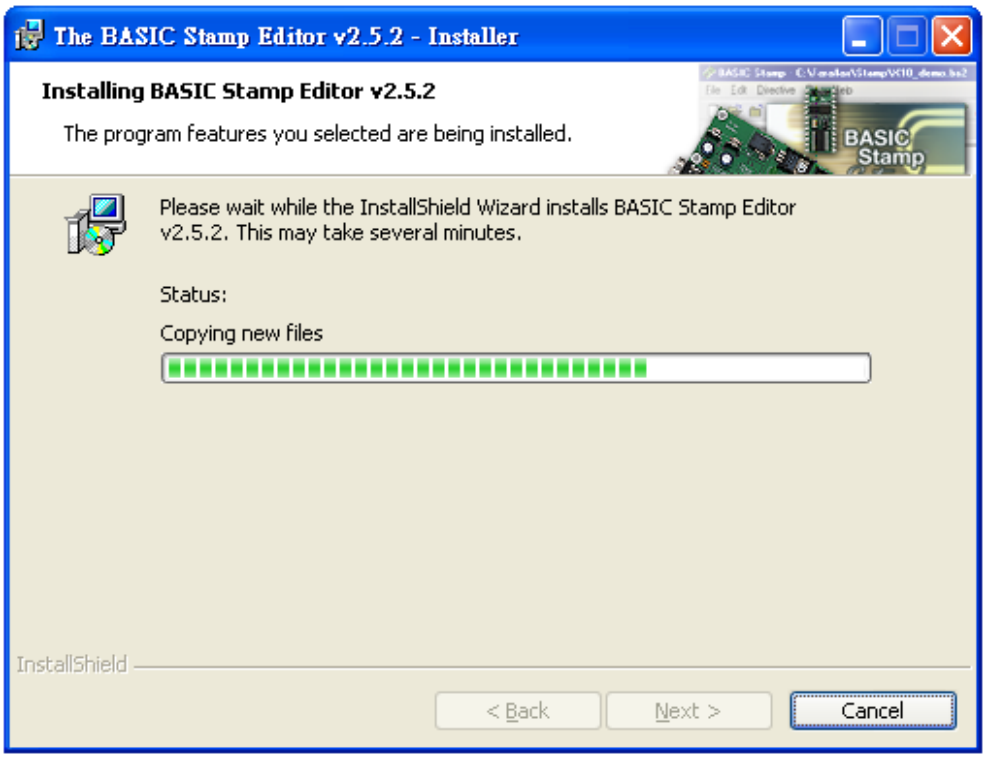


圖3-2-9 安裝檔案中

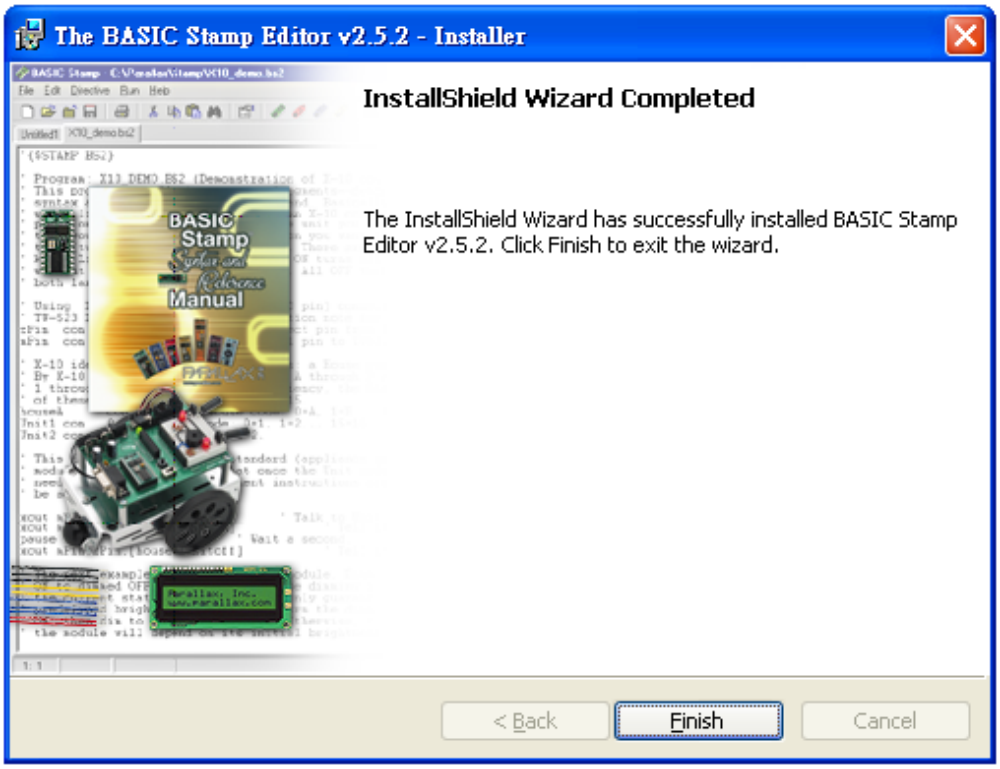


圖3-2-10 安裝完成

安裝完成後如圖3-2-10點選”Finish”即可完成安裝。

安裝完成之後桌面上會出現捷徑 

，點選之後進入編譯程式。

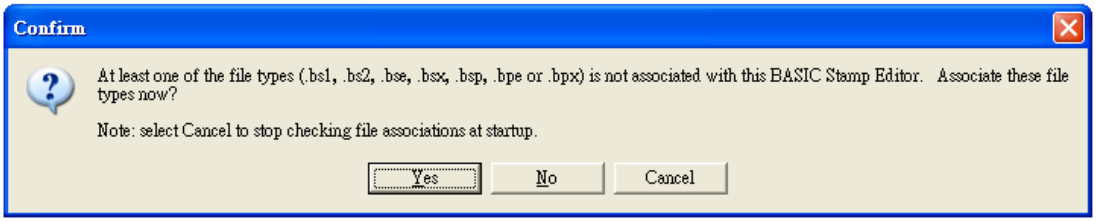


圖3-2-11 預設副檔名.BS2用此軟體開啟

第一次進入編譯程式他會詢問你是否要將副檔名.bs2的檔案預設用 BASIC Stamp Editor 開啟檔案，如圖3-2-11所示，選擇是即可。

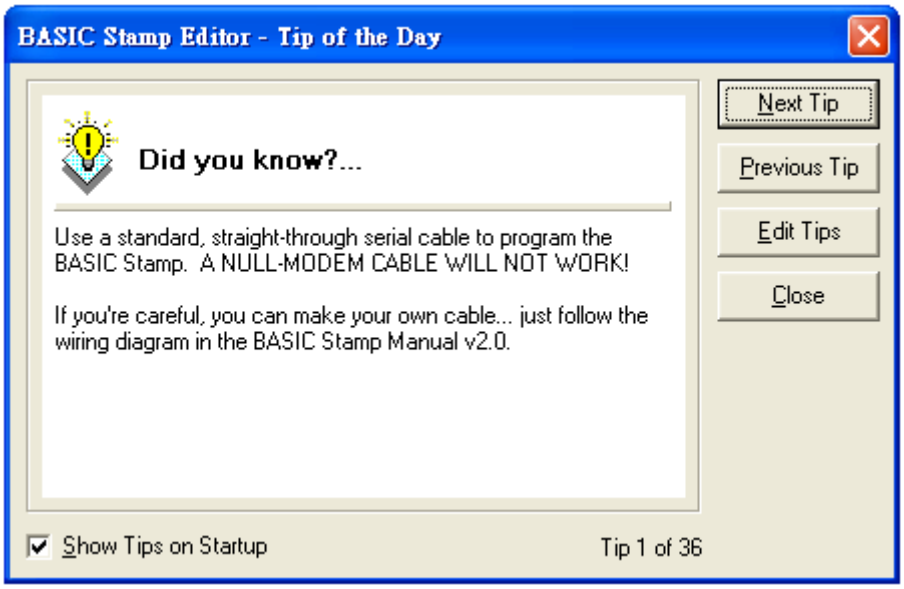


圖3-2-12 小技巧提示

接著會出現上圖3-2-12小技巧提示，若不需要點選Close即可。

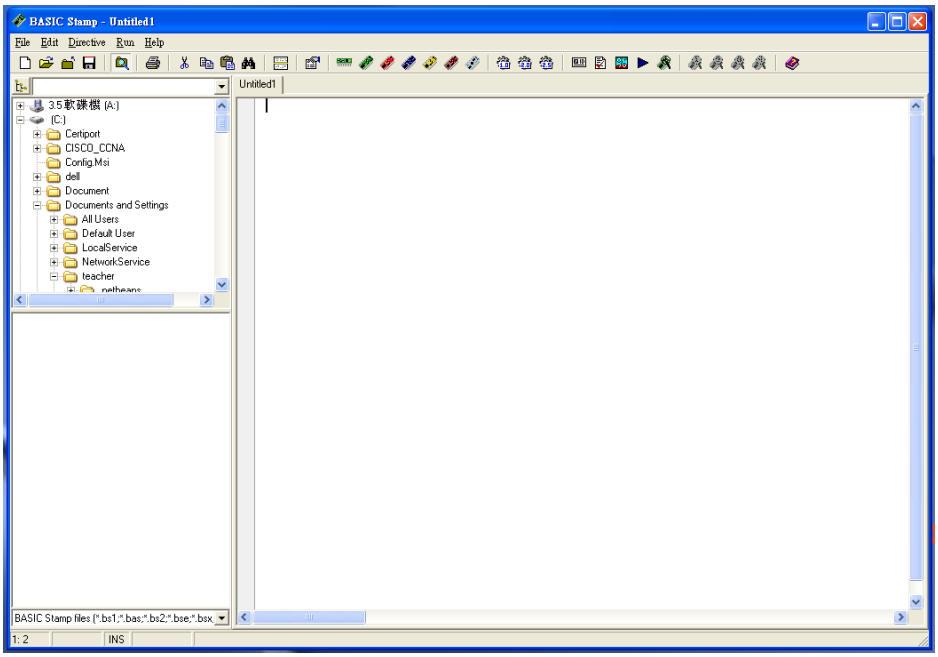


圖3-2-13 程式編譯軟體主視窗

接下來就可以看到上圖3-2-13畫面，這就代表安裝完成，環境即建置完成。

圖 3-2-14 選擇 BS2 與 PBASIC Language:2.5

按下後會在程式一開始上出現**' {$STAMP BS2} ' {$PBASIC 2.5}**

第一個型別叫做$STAMP 型別，它會告訴BASIC Stamp Editor 你將會載入程式至BASIC Stamp 2。

第二個型別叫做$PBASIC 型別，它會告訴BASIC Stamp Editor 你現在使用的是 2.5 PBASIC 程式語言。

3-3 Ubuntu 環境建置

系統建置:

使用 Oracle VM VirtualBox 安裝 Ubuntu Server 11.04

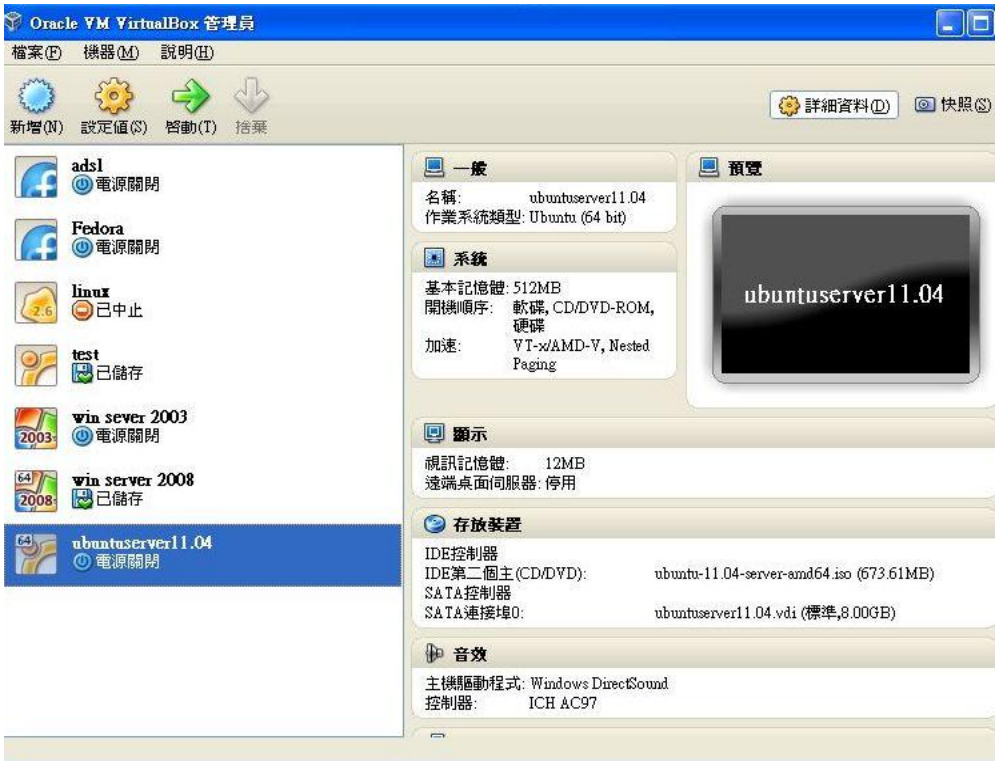


圖 3-3-1 為初始介面

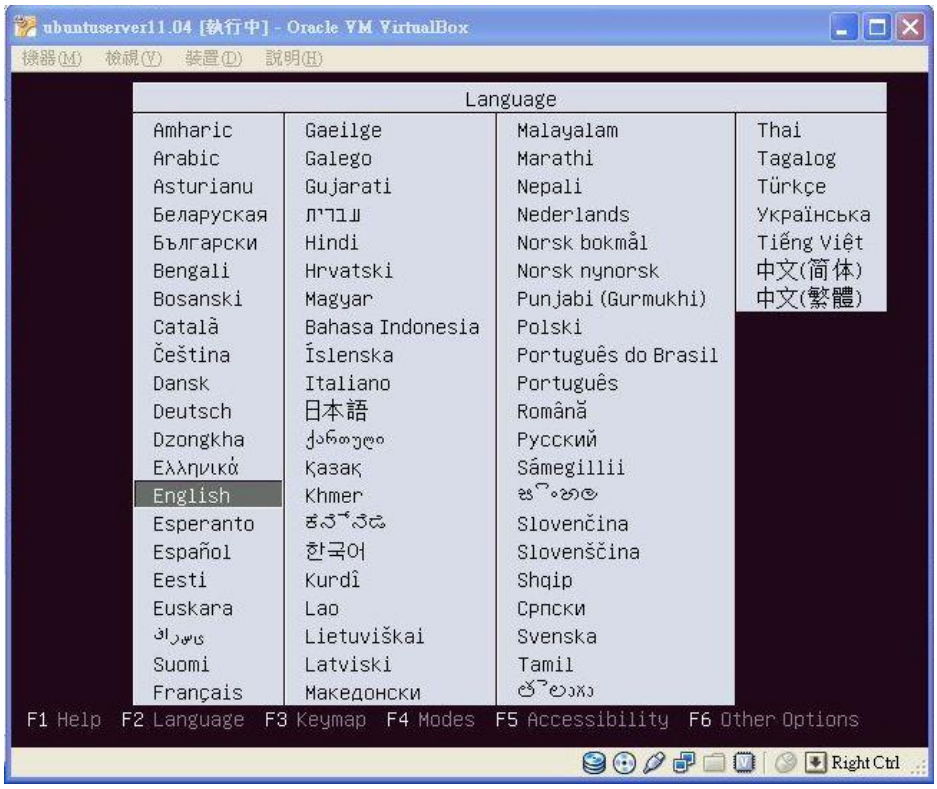


圖 3-3-2 選擇英文介面安裝

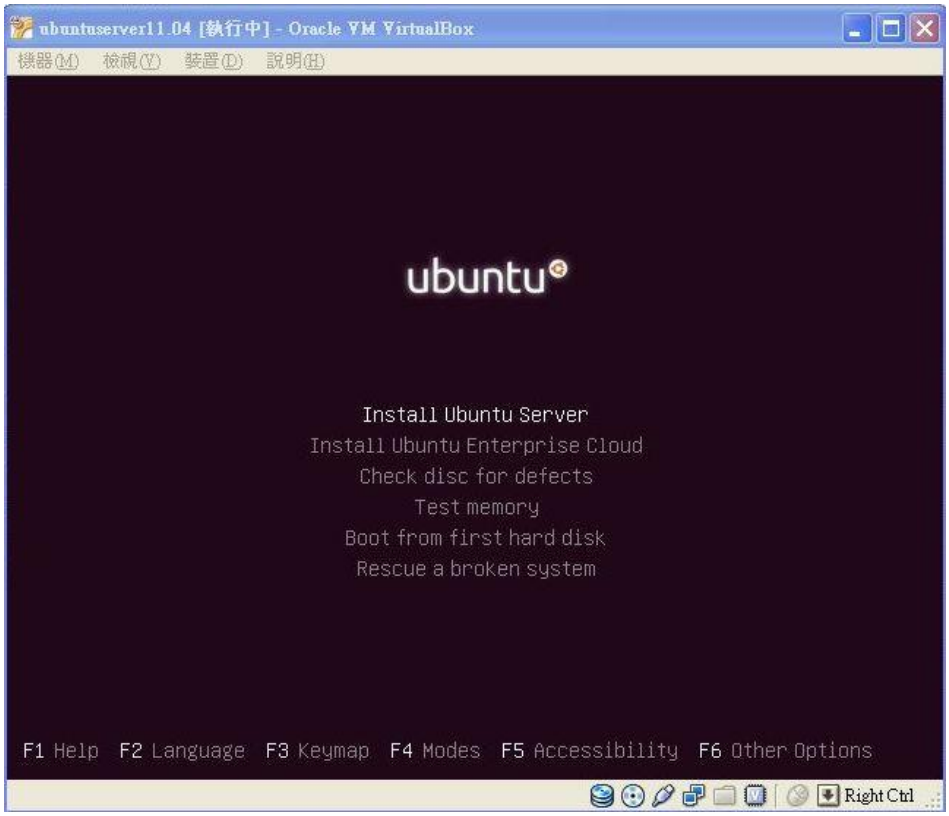


圖 3-3-3 開始安裝

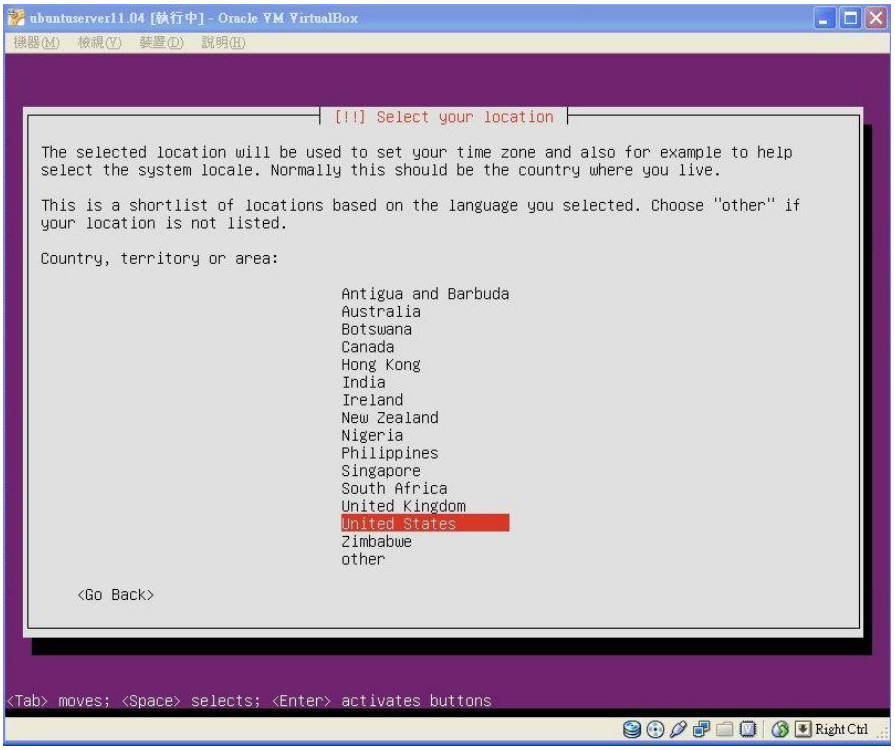


圖 3-3-4 以安裝英文版為主

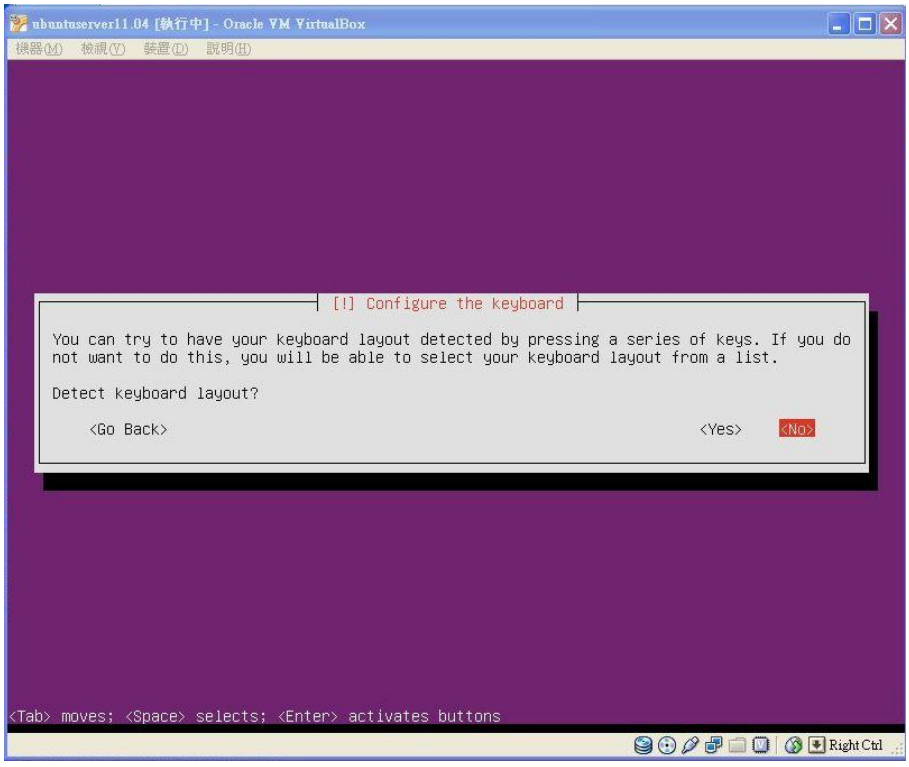


圖 3-3-5 選”YES”會幫你偵測鍵盤類型，”NO”會自行選擇

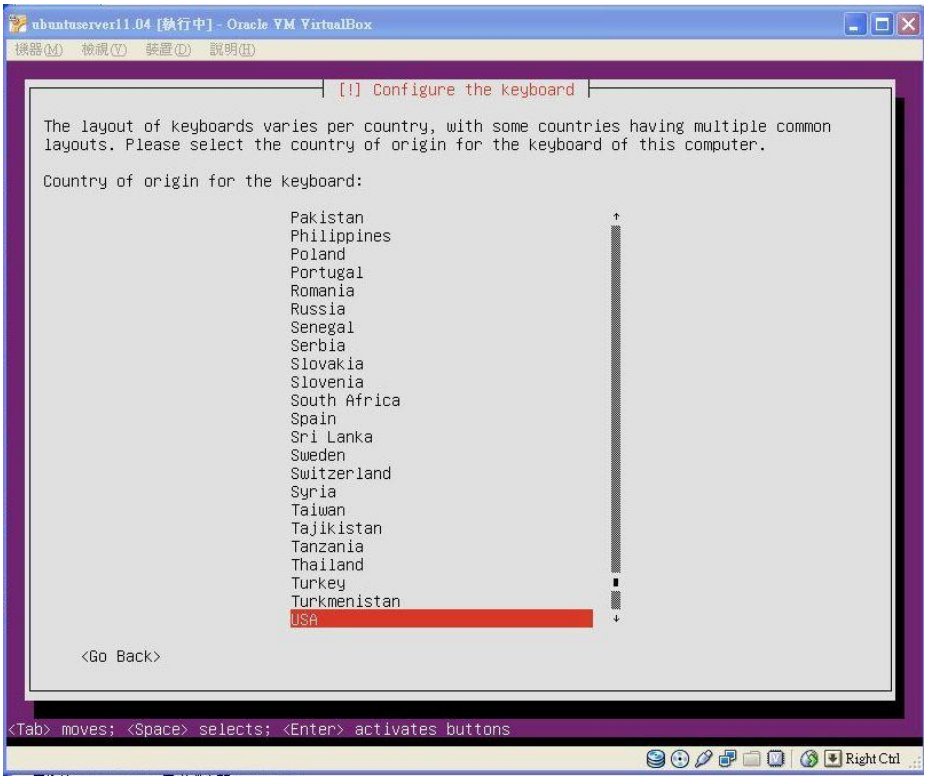


圖 3-3-6 設定你的 keyboard

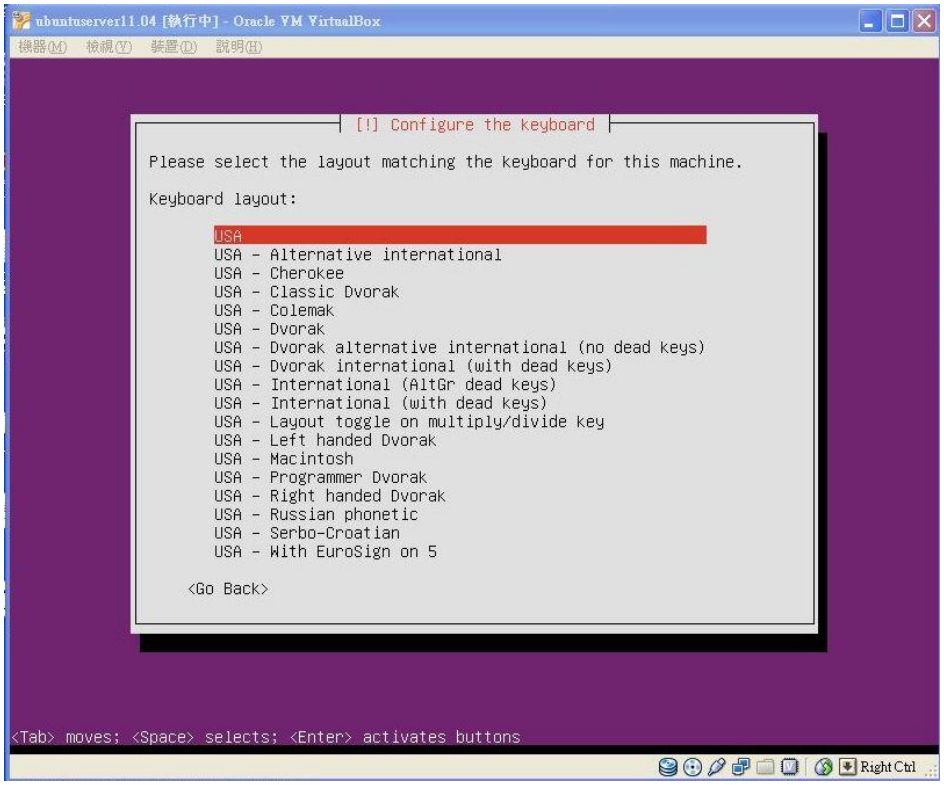


圖 3-3-7 選擇 USA

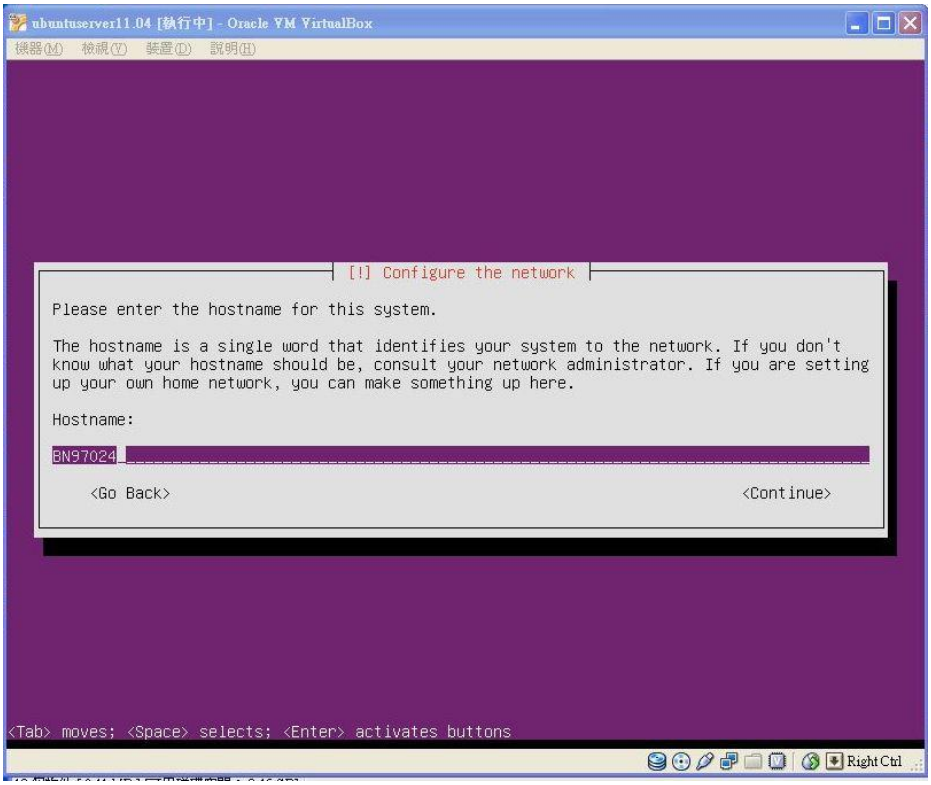


圖 3-3-8 設定電腦名稱

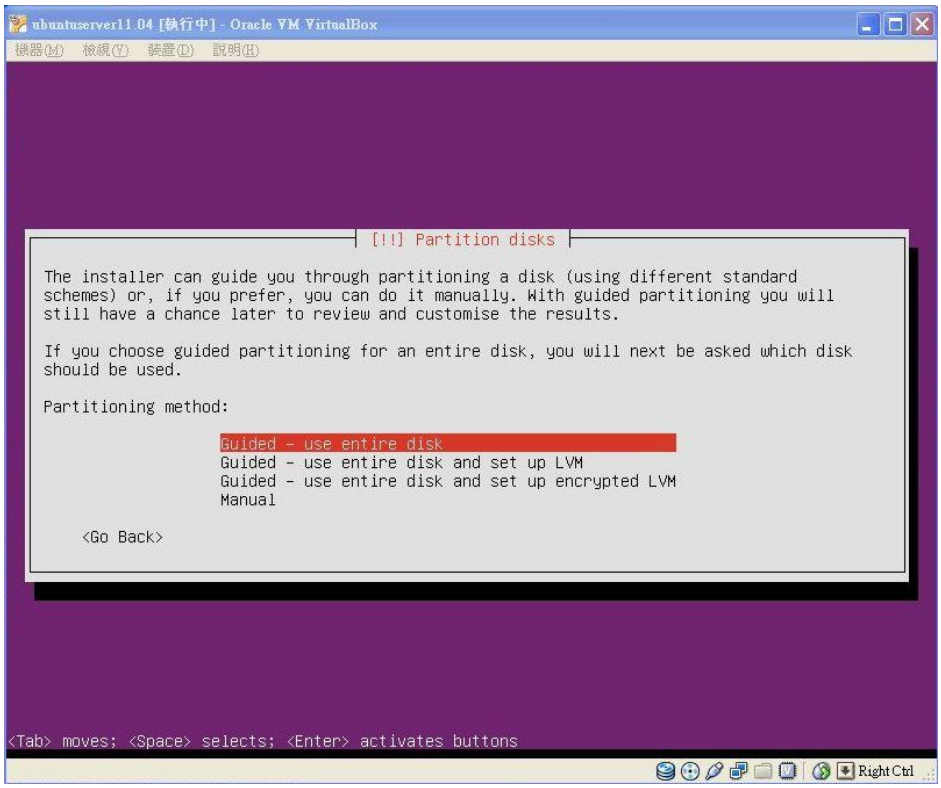


圖 3-3-9 使用建議來分割硬碟

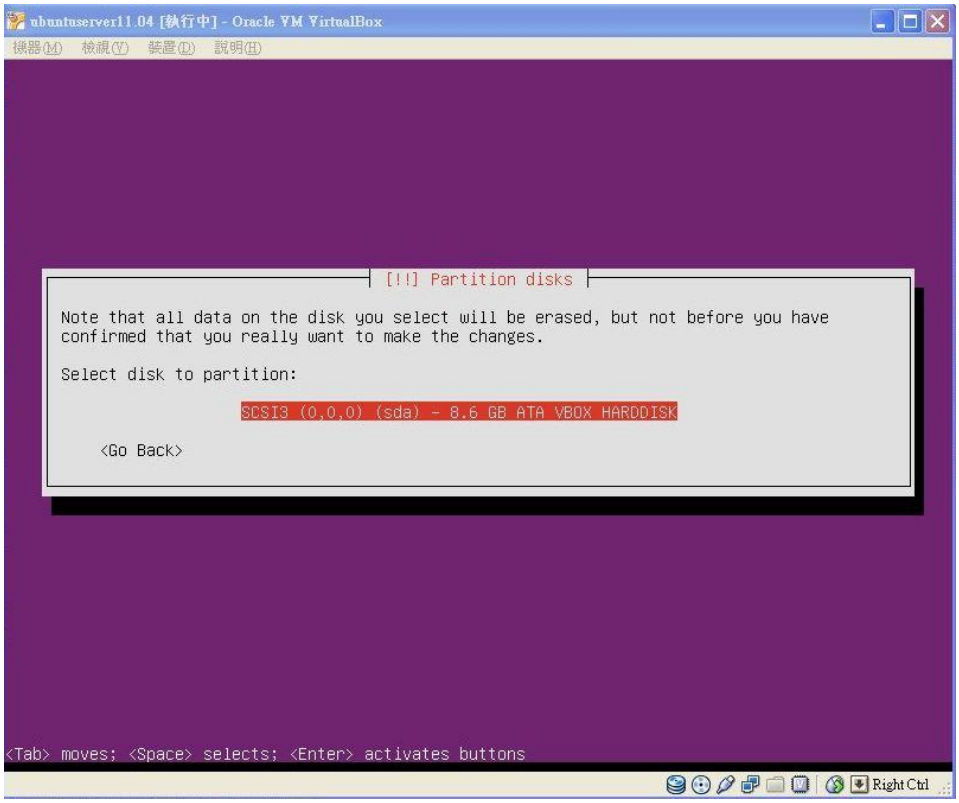


圖 3-3-10 選擇一個硬碟

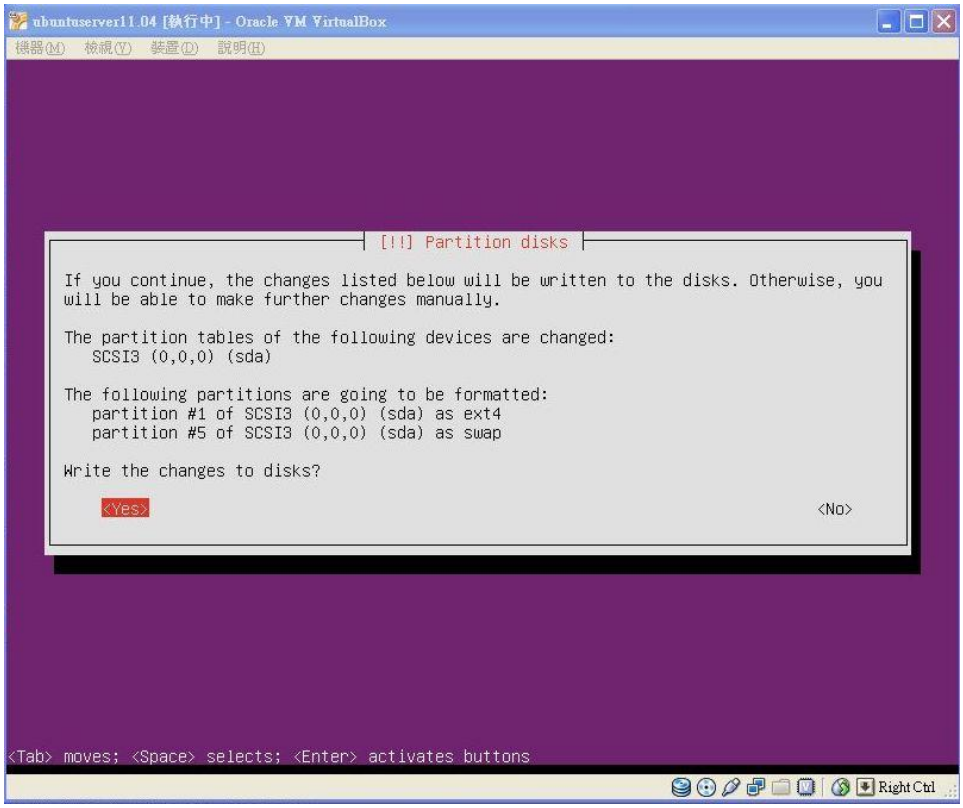


圖 3-3-11 預覽分割的情況，選”YES”就確認此分割

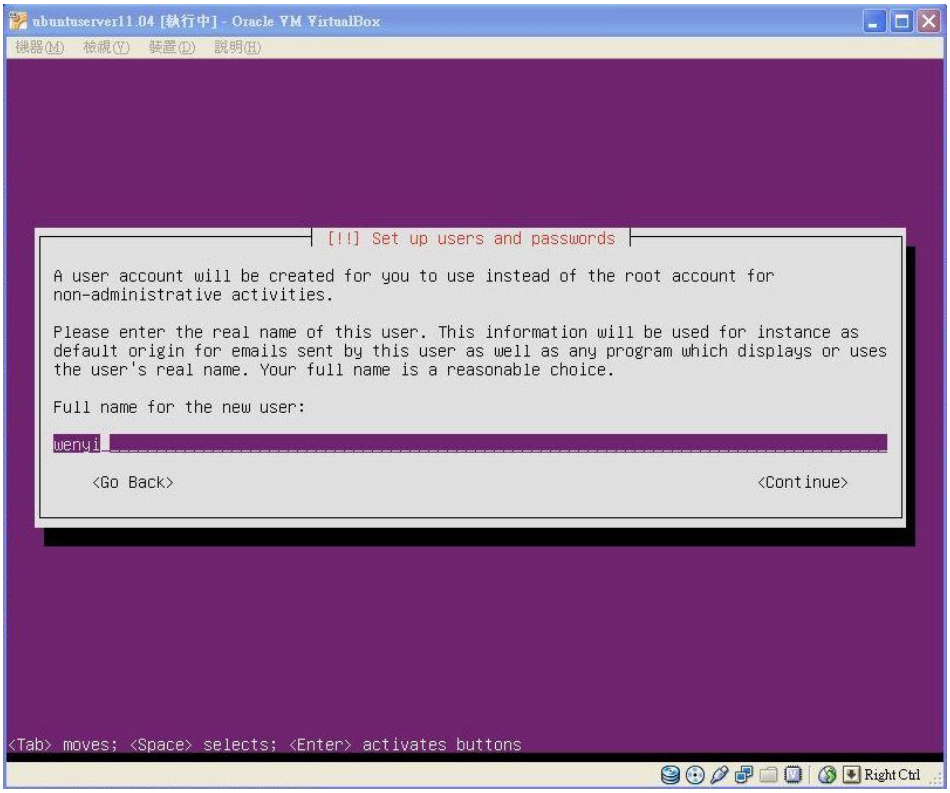


圖 3-3-12 使用者名稱

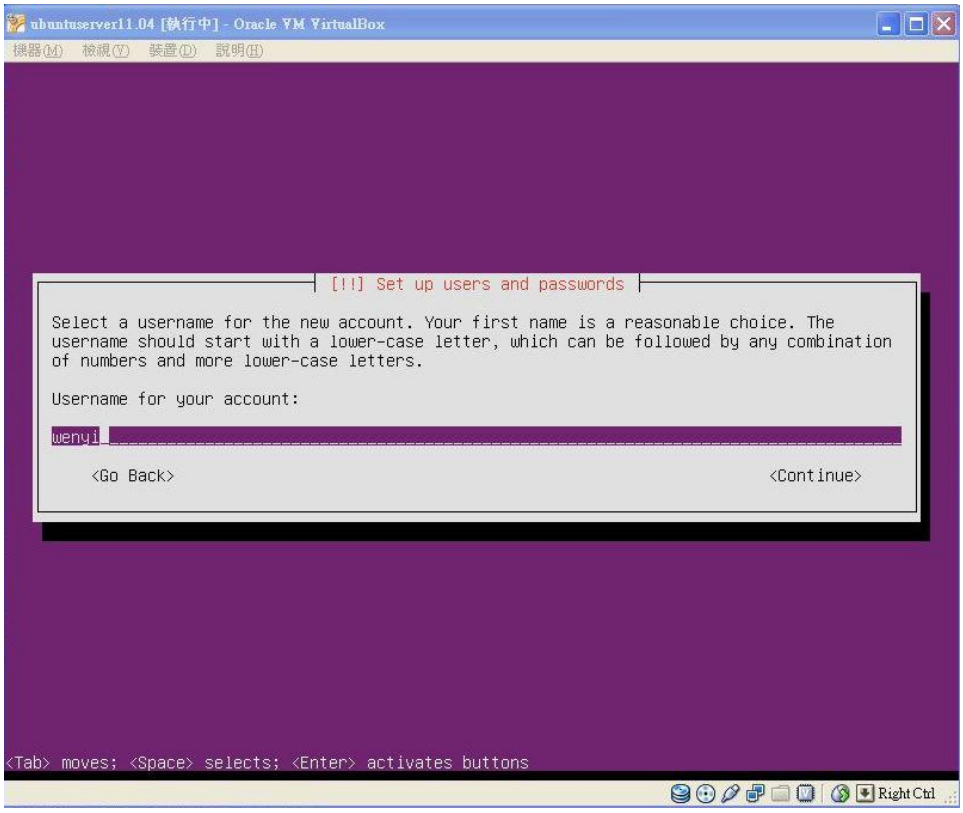


圖 3-3-13 使用者帳號

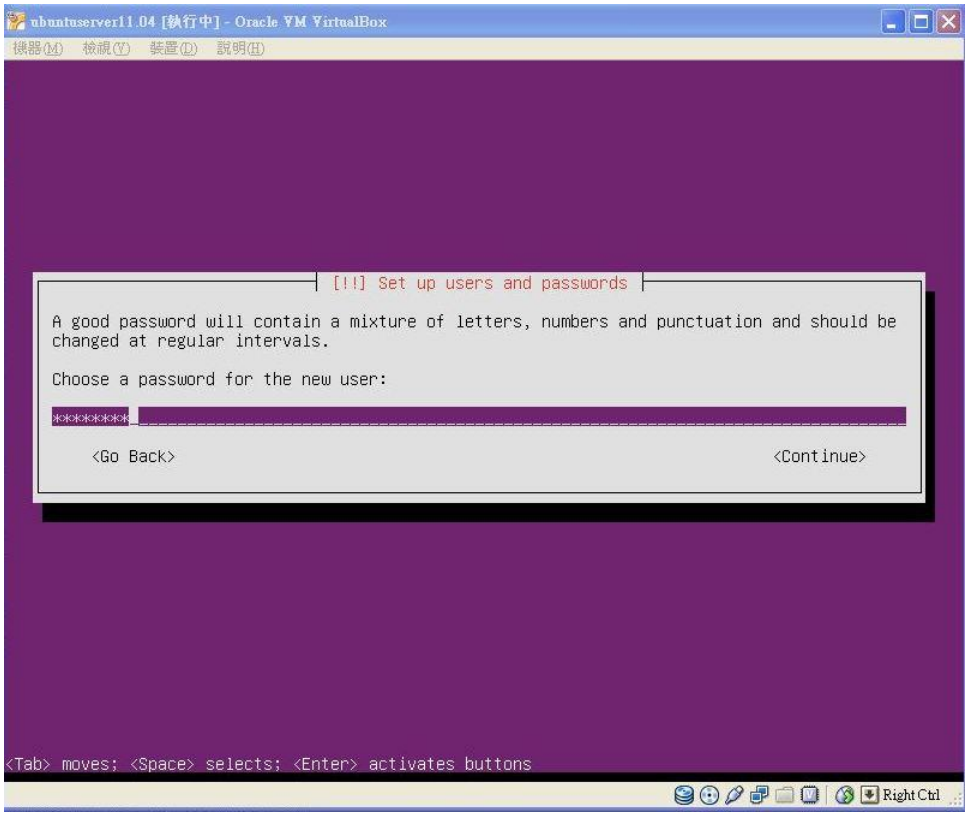


圖 3-3-14 設定密碼

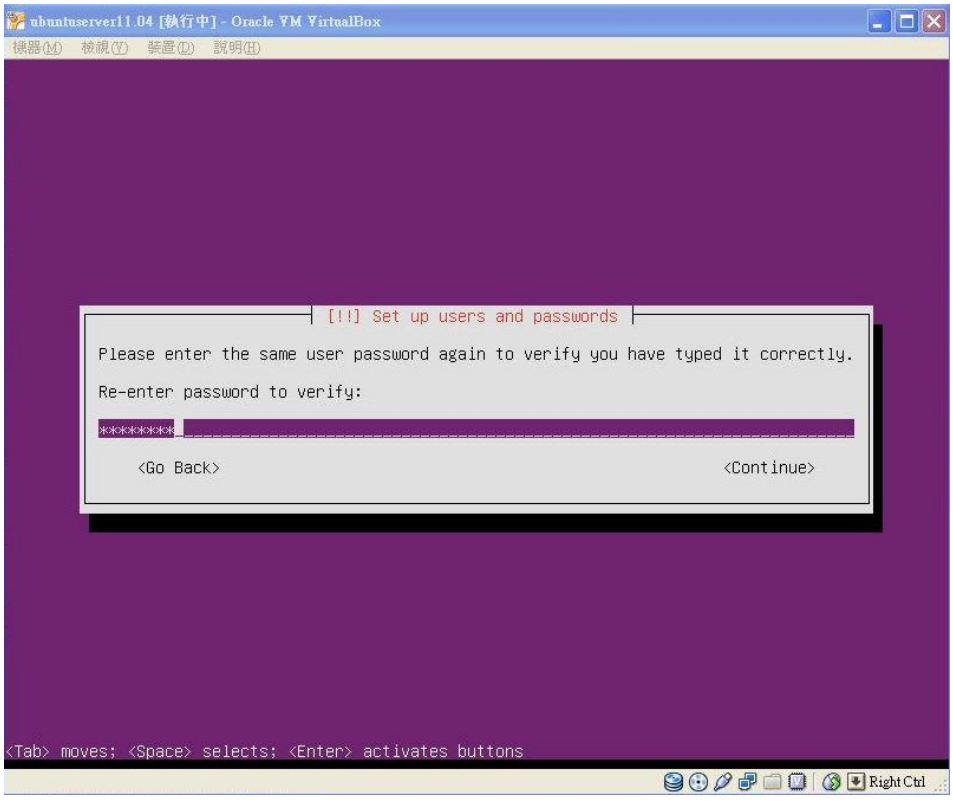


圖 3-3-15 再輸入一次密碼

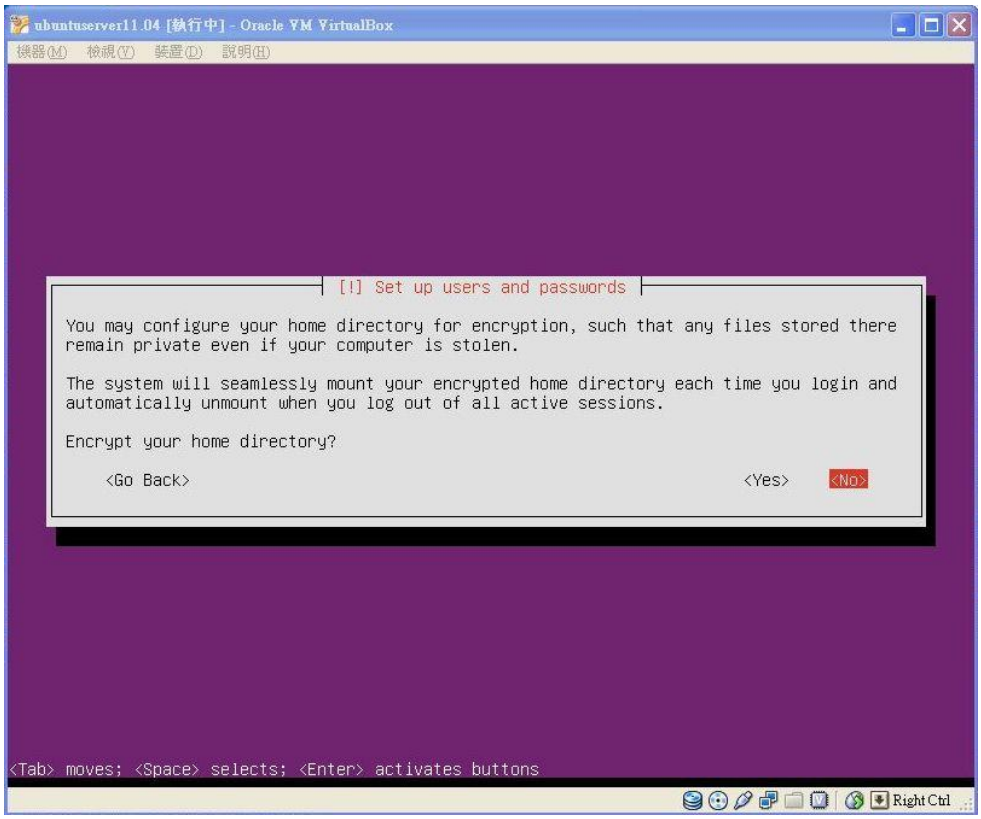


圖 3-3-16 家目錄是否加密

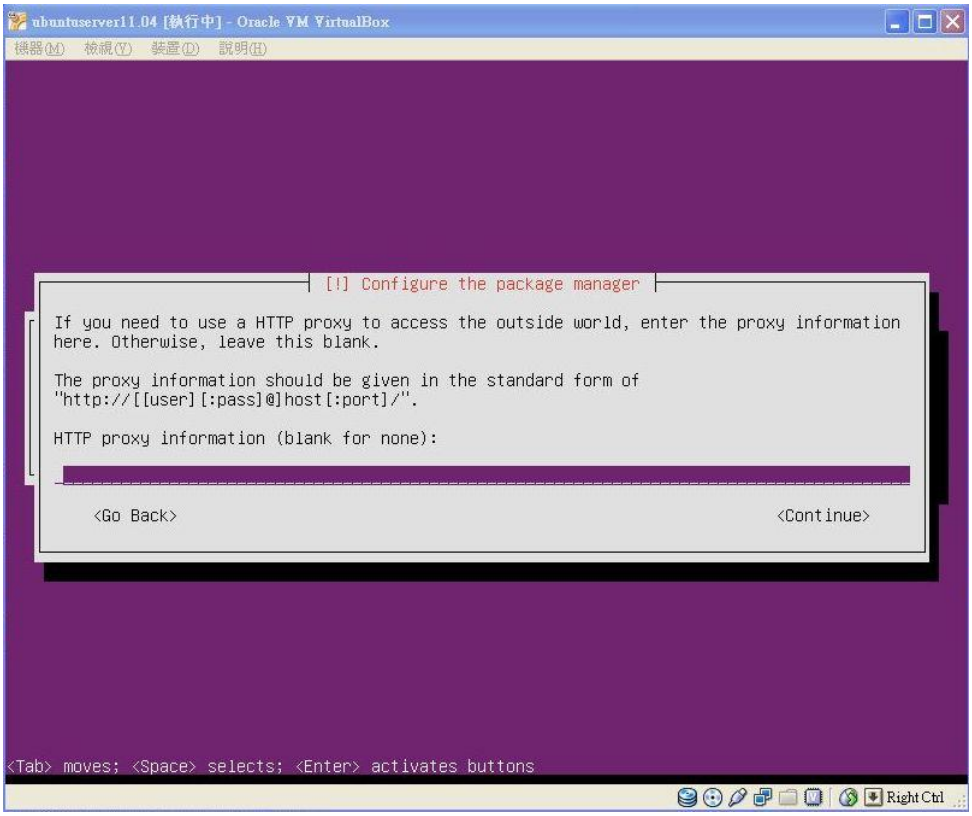


圖 3-3-17 有 proxy 可在此設定

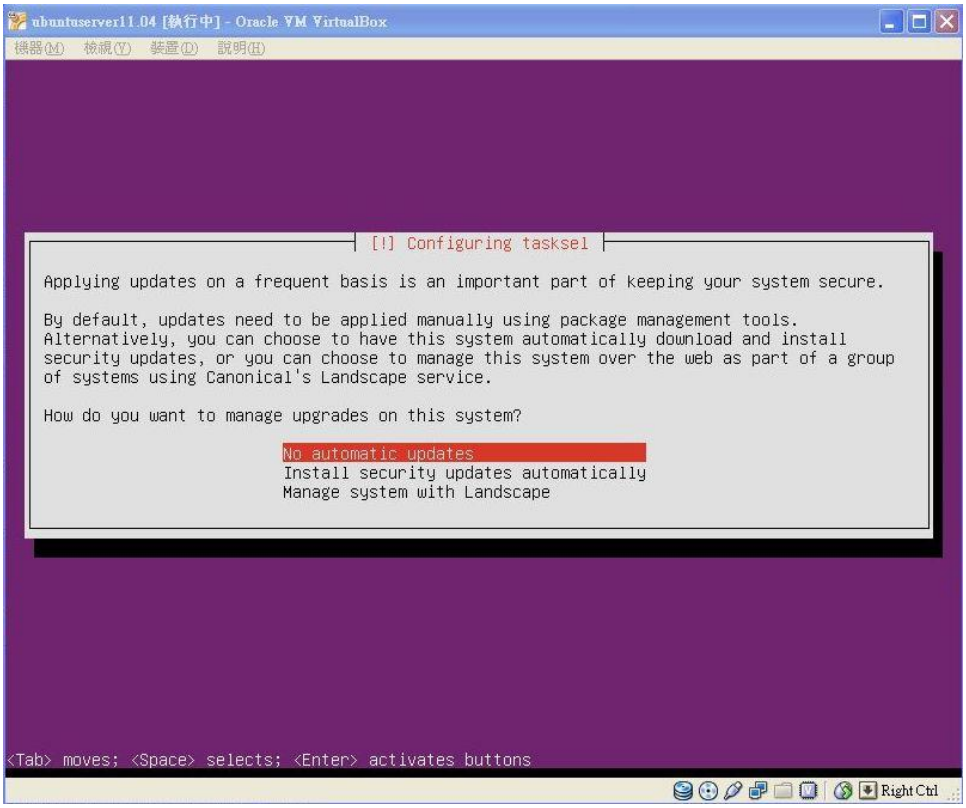


圖 3-3-18 需不需要馬上更新，這裡選不更新

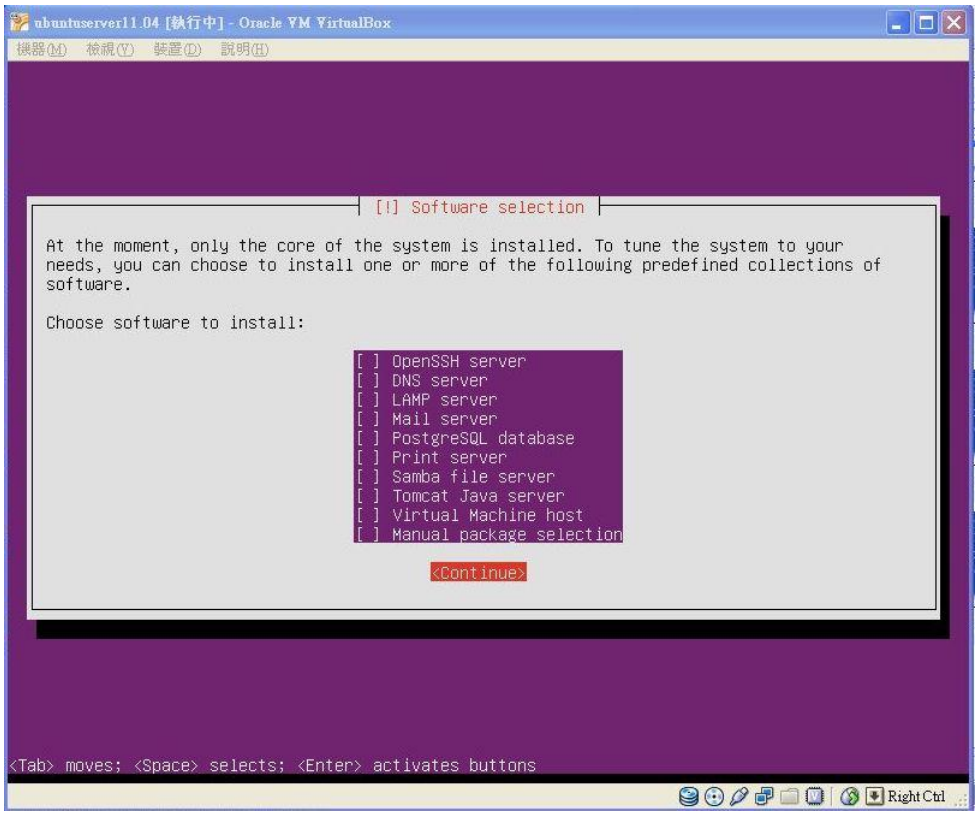


圖 3-3-19 安裝的套件，先不安裝，進系統之後再安裝

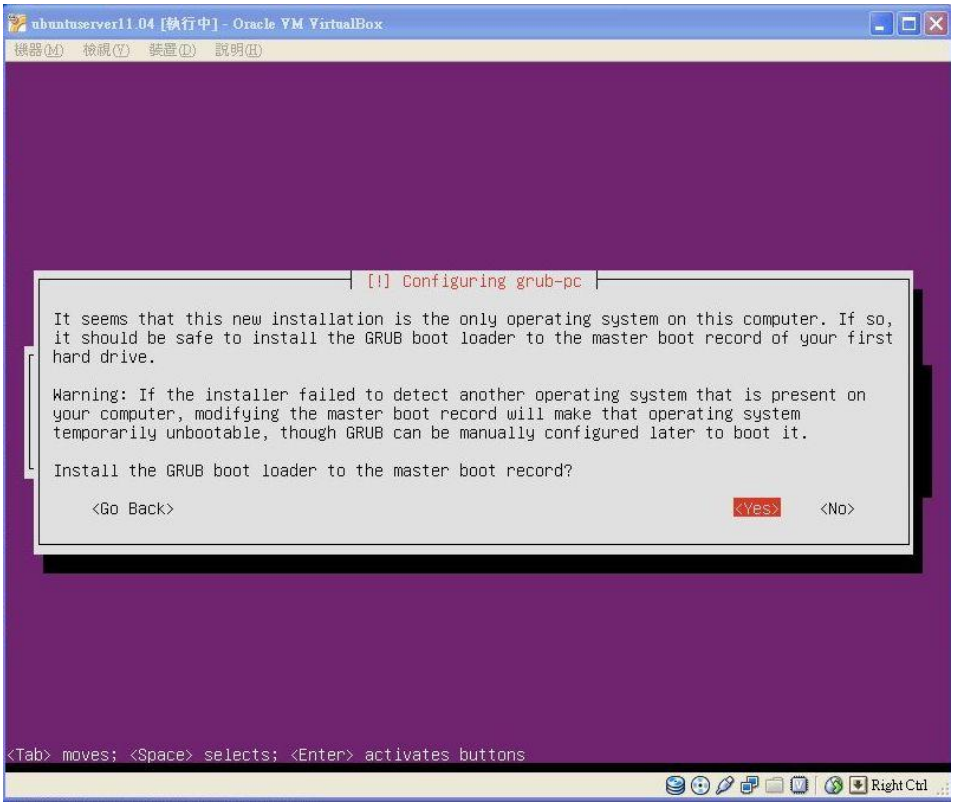


圖 3-3-20 使用 GRUB 開機選單

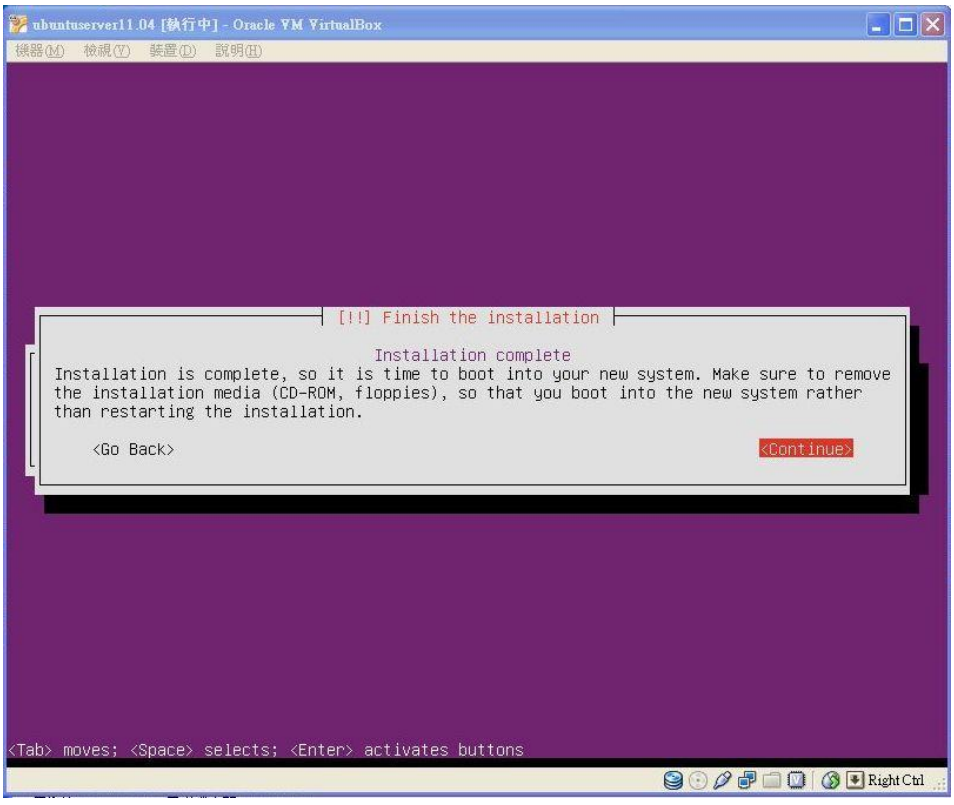


圖 3-3-21 提醒重開機前先拿出光碟

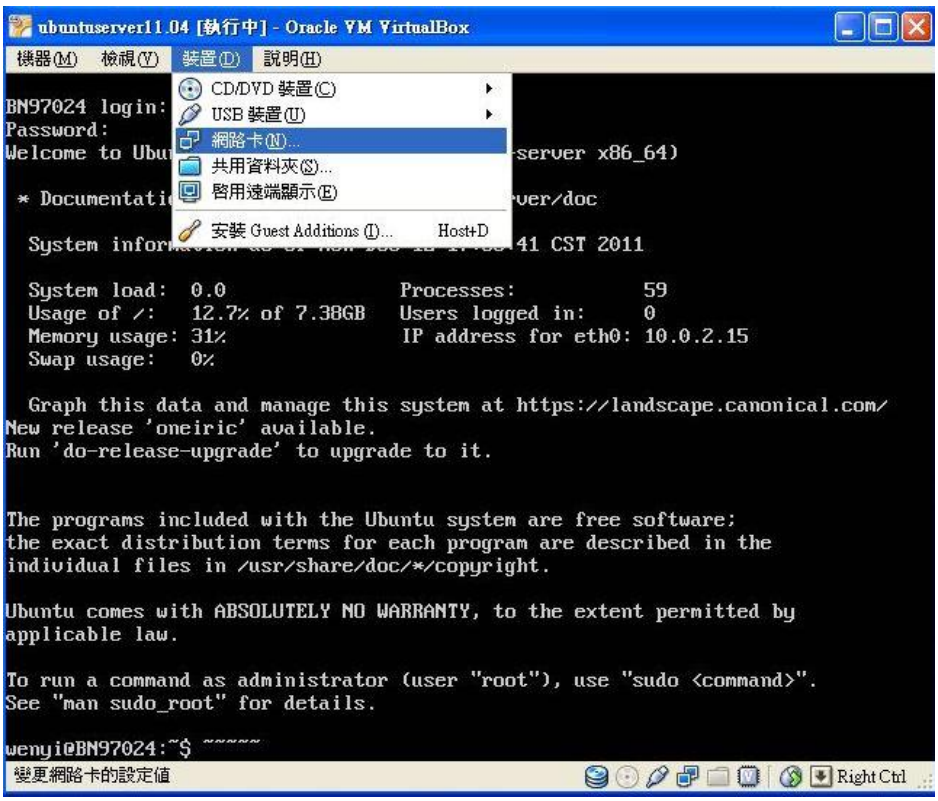


圖 3-3-22 將網路模式改成 bridge 模式，這樣便可使用實體 IP

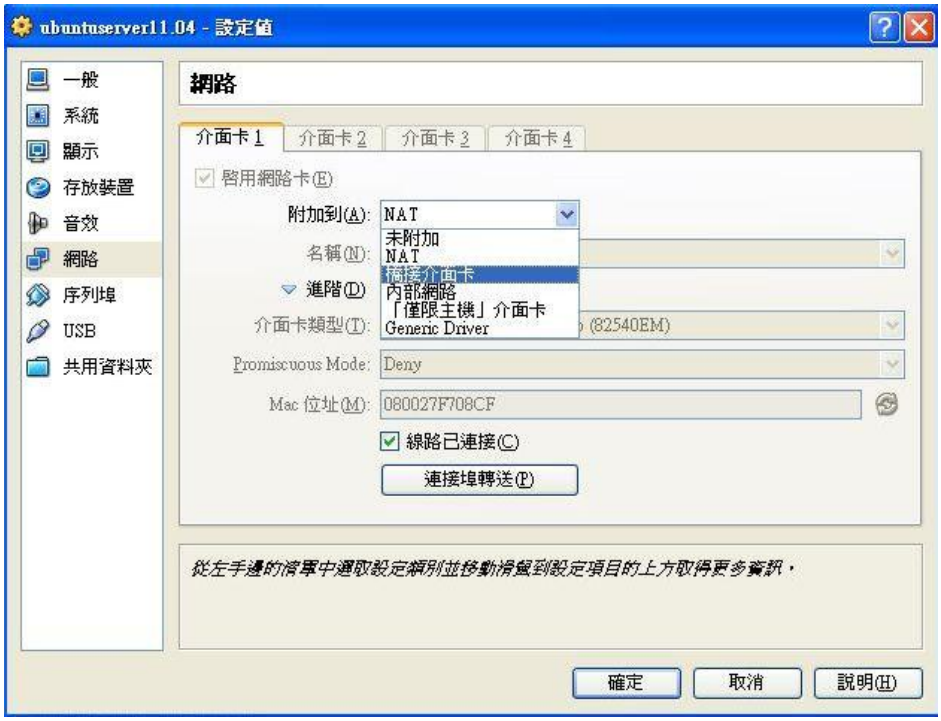


圖 3-3-23 裝置網路卡

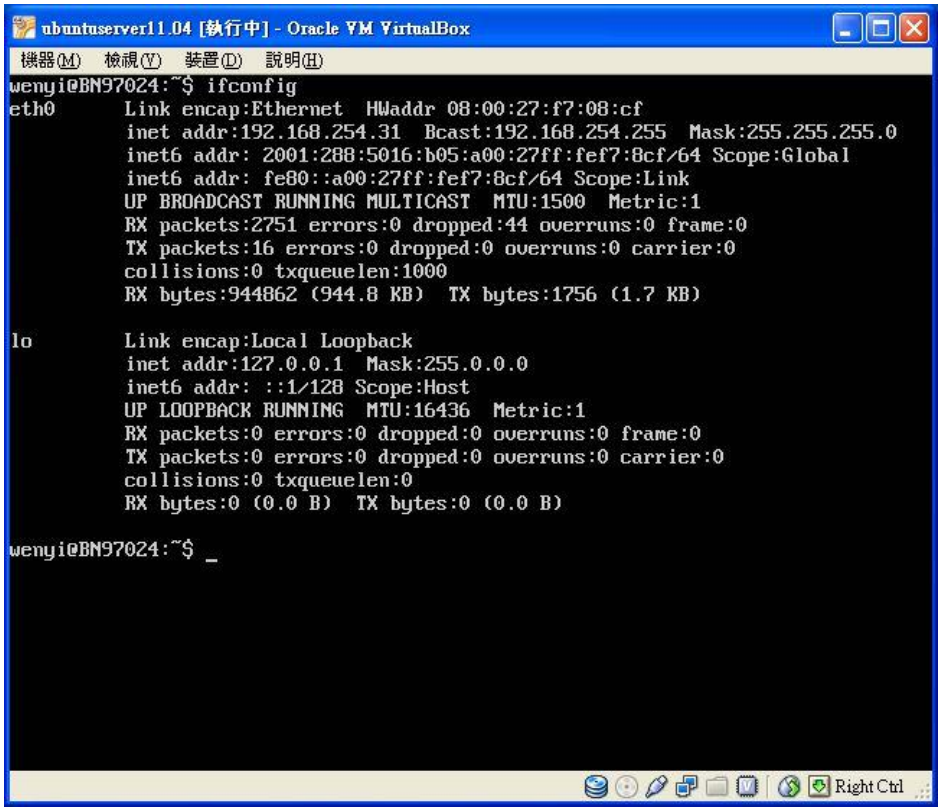


圖 3-3-24 改成 Bridge 後還需將網路介面作設定

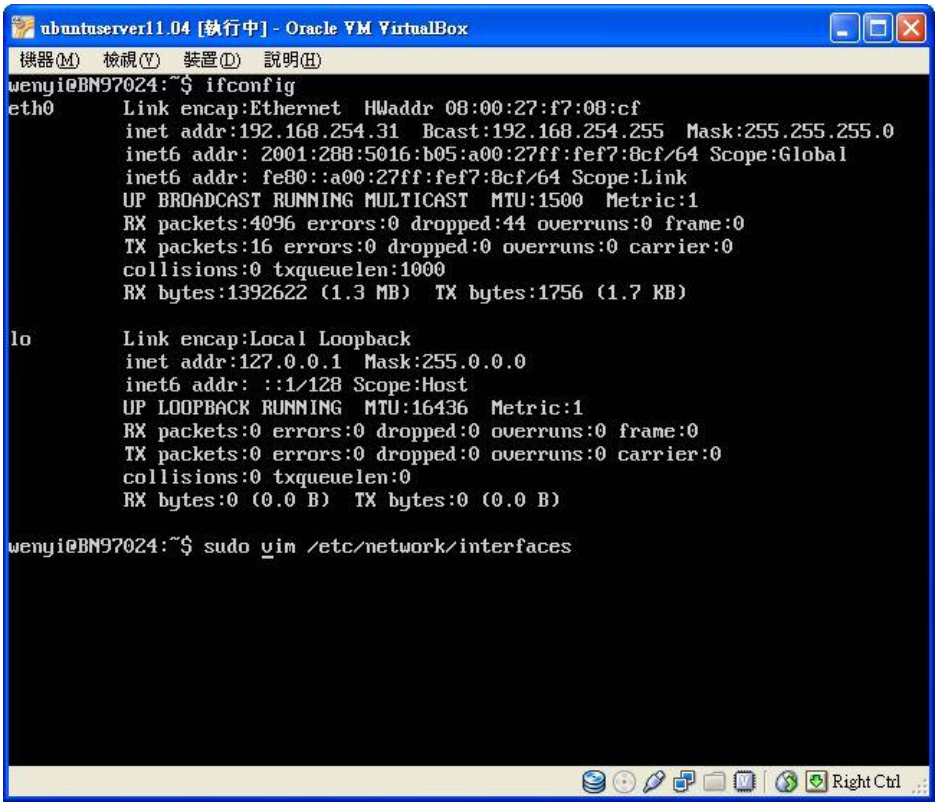


圖 3-3-25 ubuntu 初始介面，更改/etc/network/interfaces

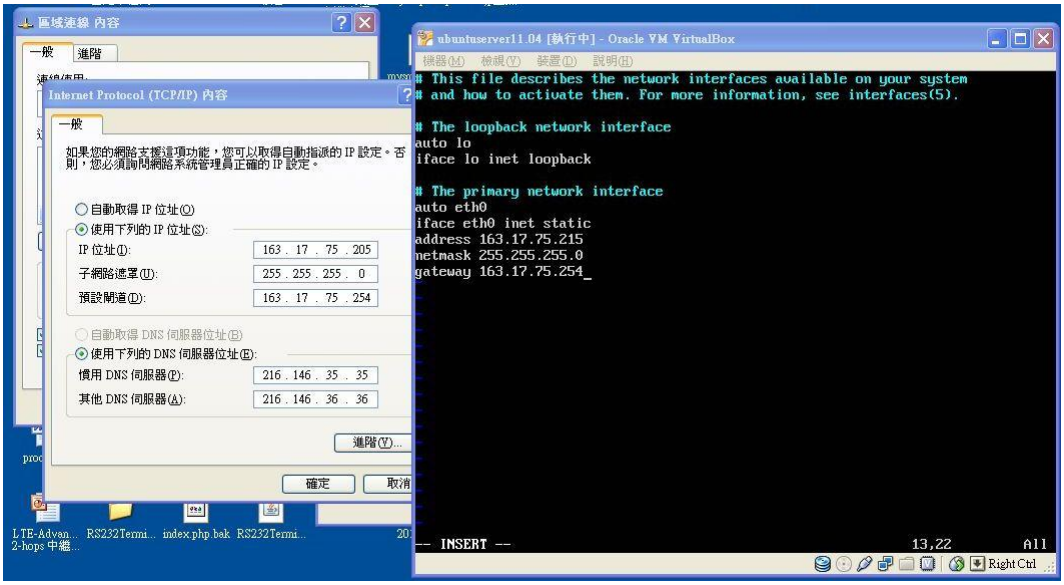


圖 3-3-26 將自動取得 IP 改成靜態並加上 IP、網路遮罩、閘道

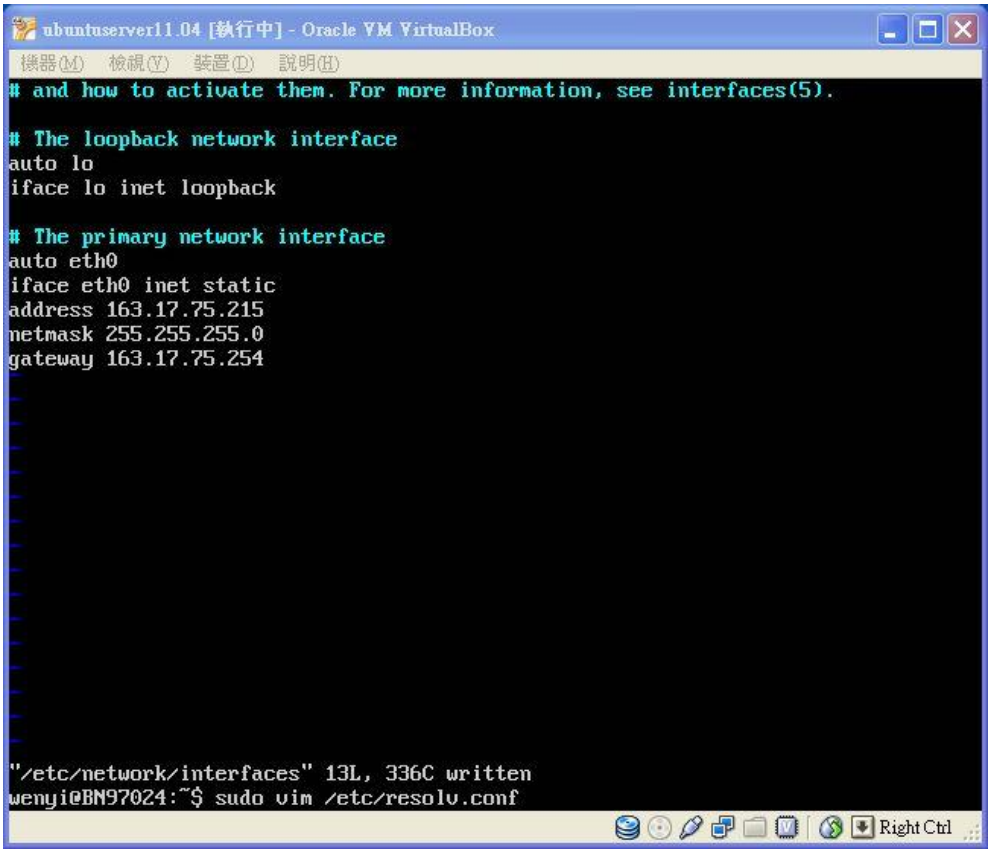


圖 3-3-27 修改/etc/resolv.conf

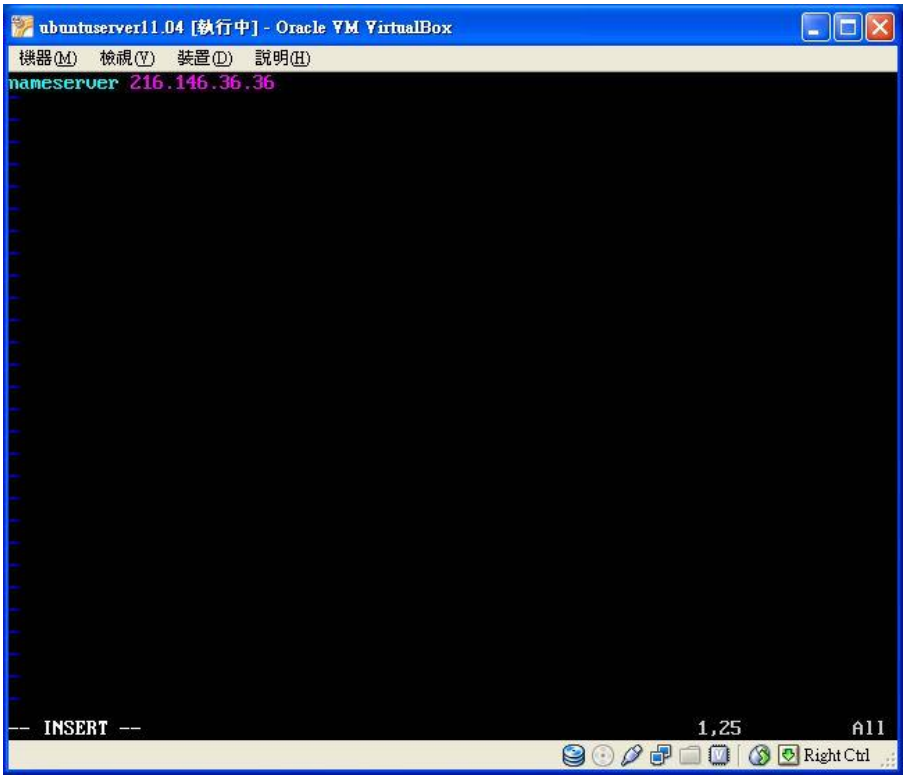


圖 3-3-28 設定 DNS Server 使用上圖之 DNS Server IP

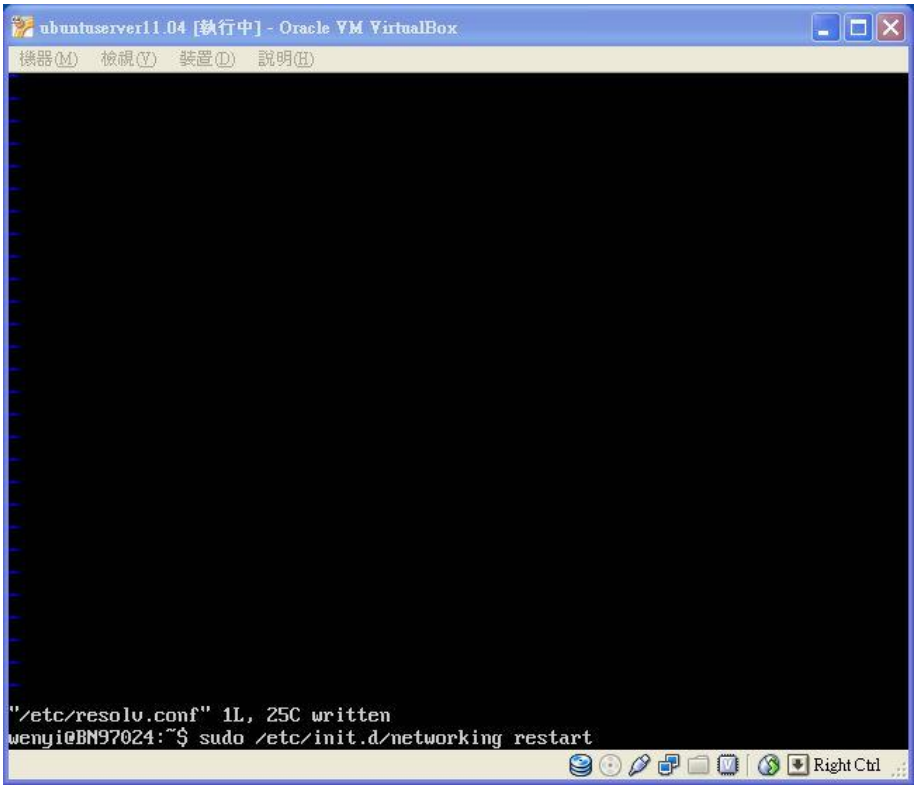


圖 3-3-29 設定好之後，需重新啟動網路使其更新

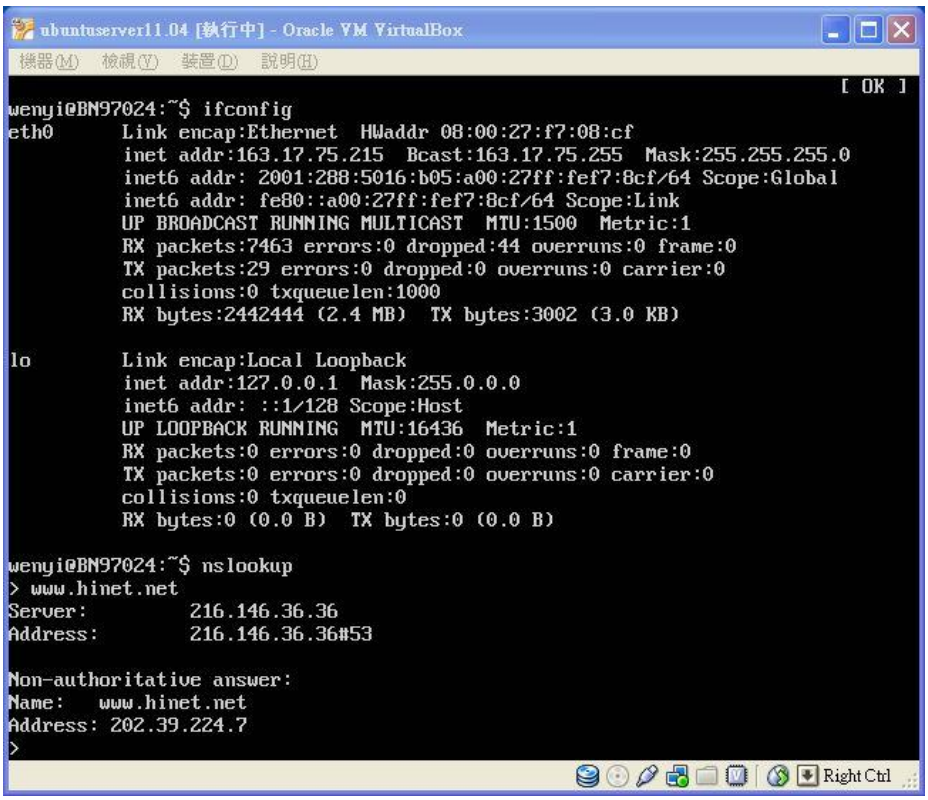


圖 3-3-30 設定完成並測試

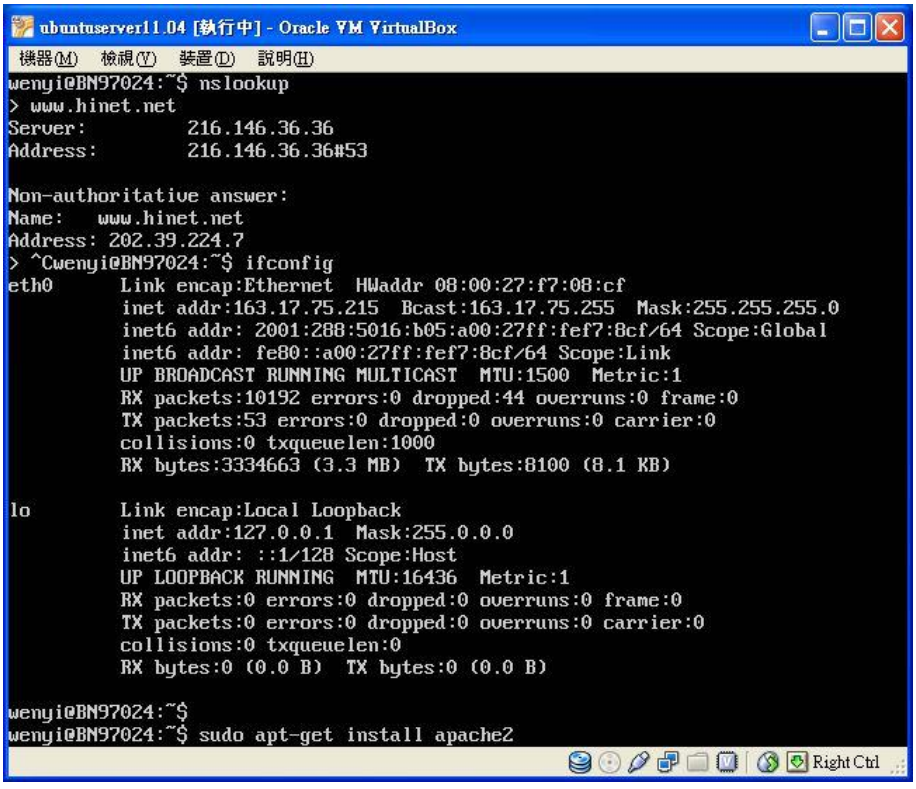


圖 3-3-31 安裝 Apache

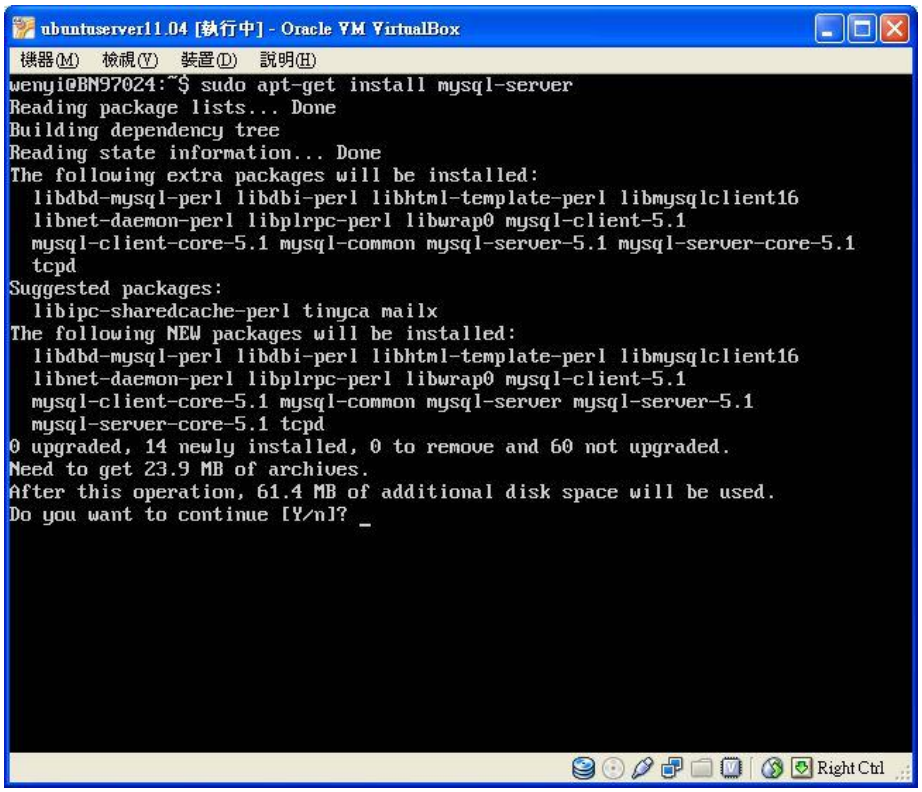


圖 3-3-32 安裝 SQL-Server

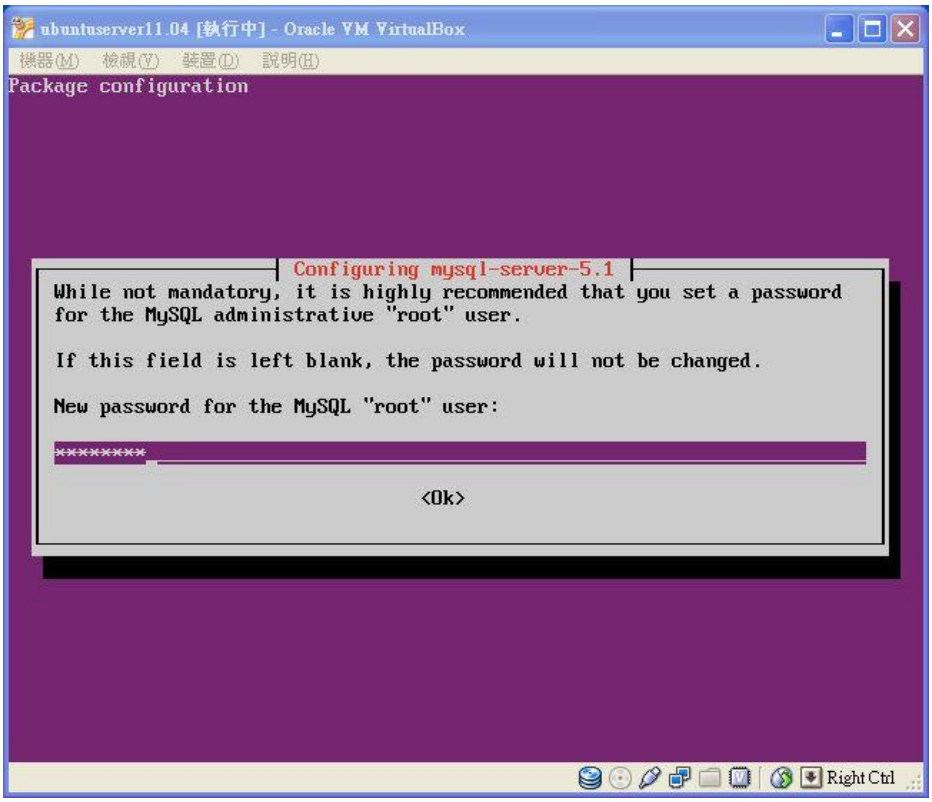


圖 3-3-33 設定 MYSQL 的 ROOT 密碼

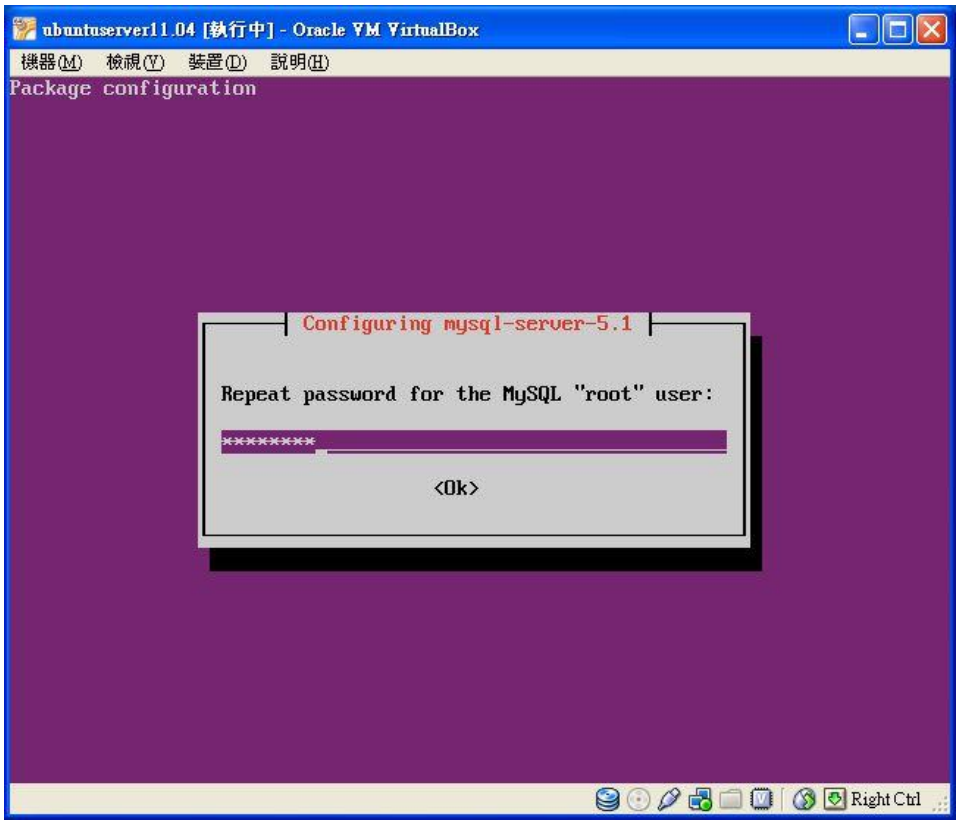


圖 3-3-34 再輸入一次

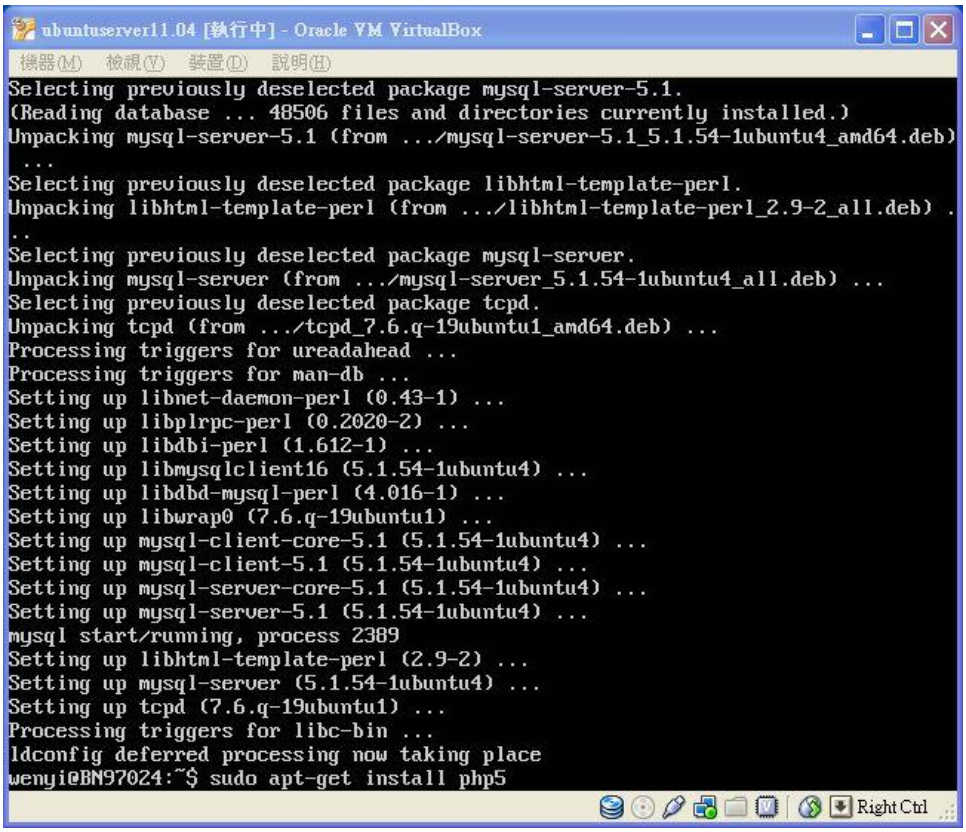


圖 3-3-35 安裝 PHP

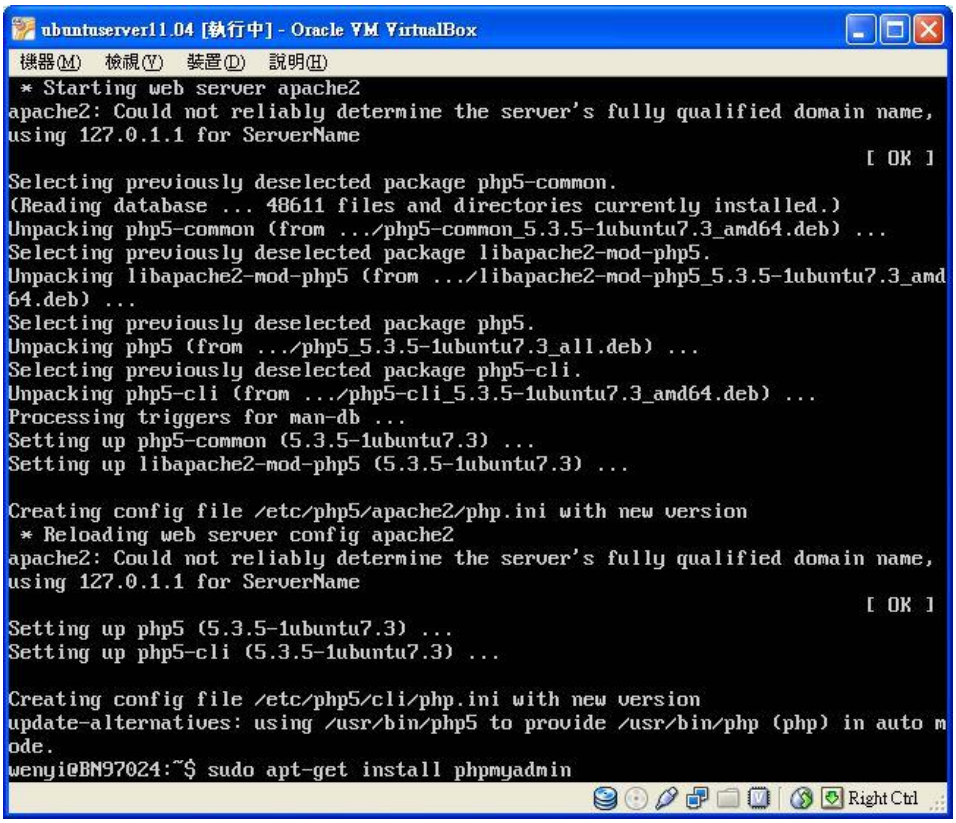


圖 3-3-36 安裝資料庫管理套件-phpmyadmin

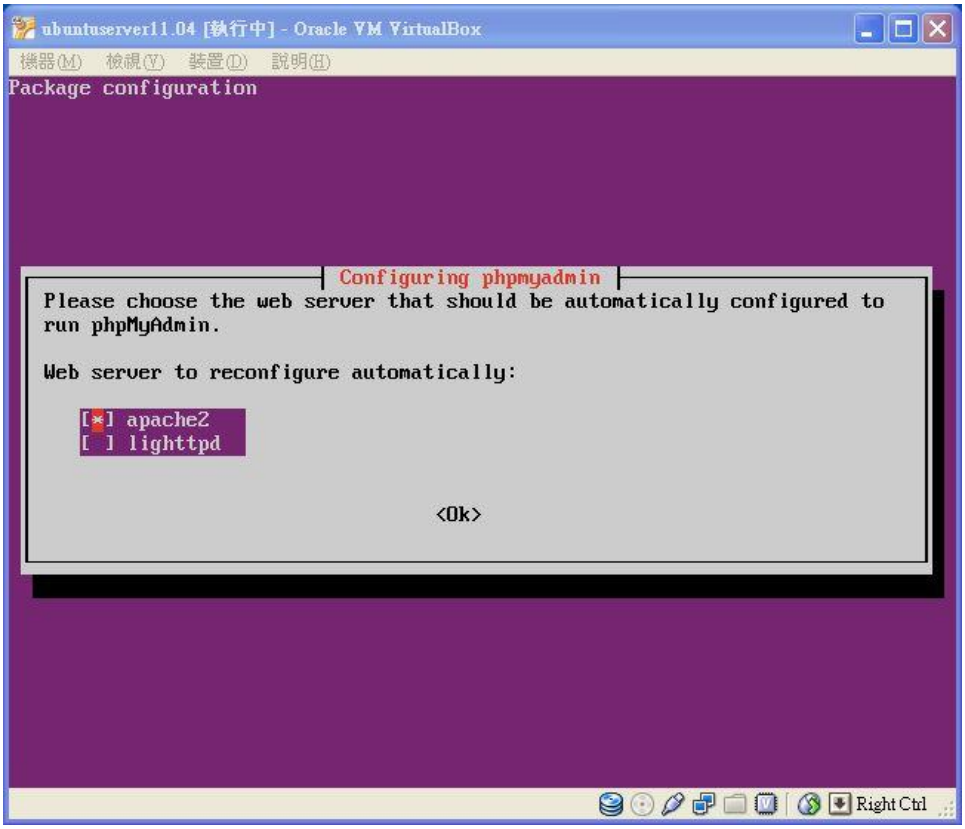


圖 3-3-37 選擇 apache2

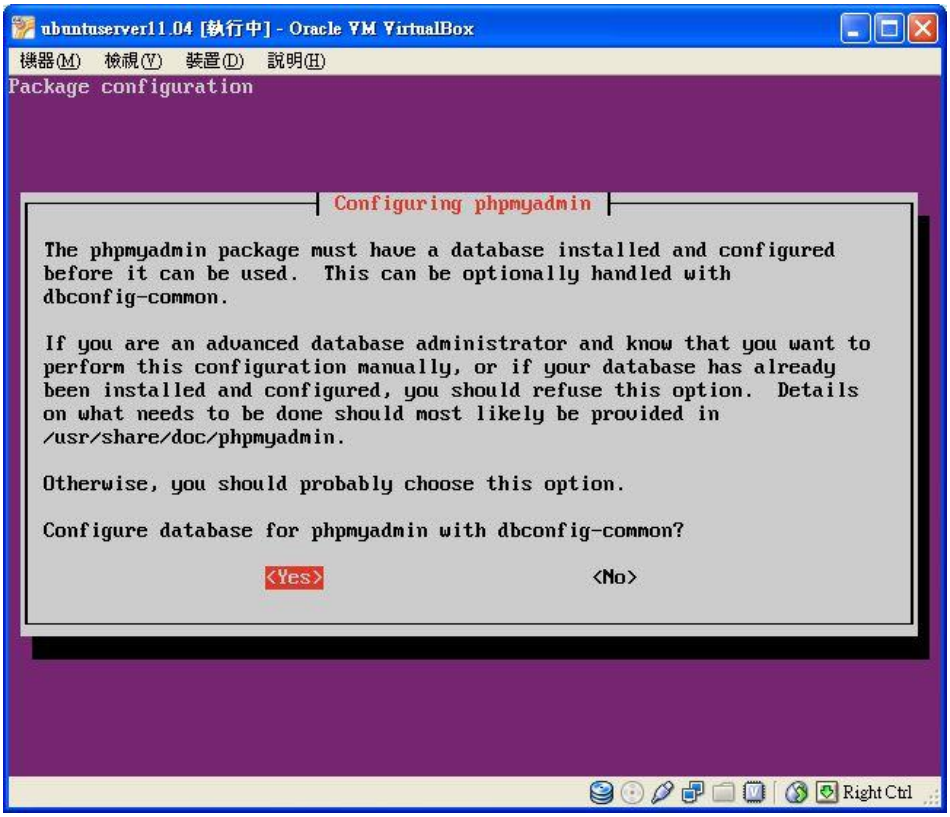


圖 3-3-38 是否使用 dbconfig-common 來設定資料庫

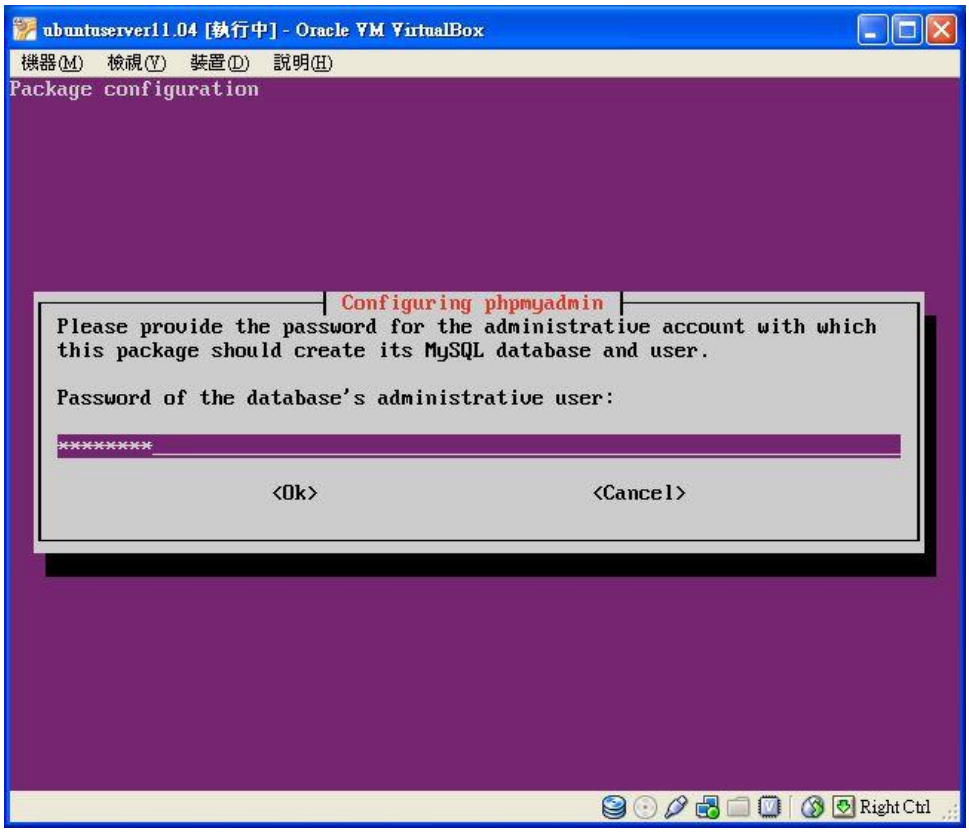


圖 3-3-39 設定資料庫管理者密碼

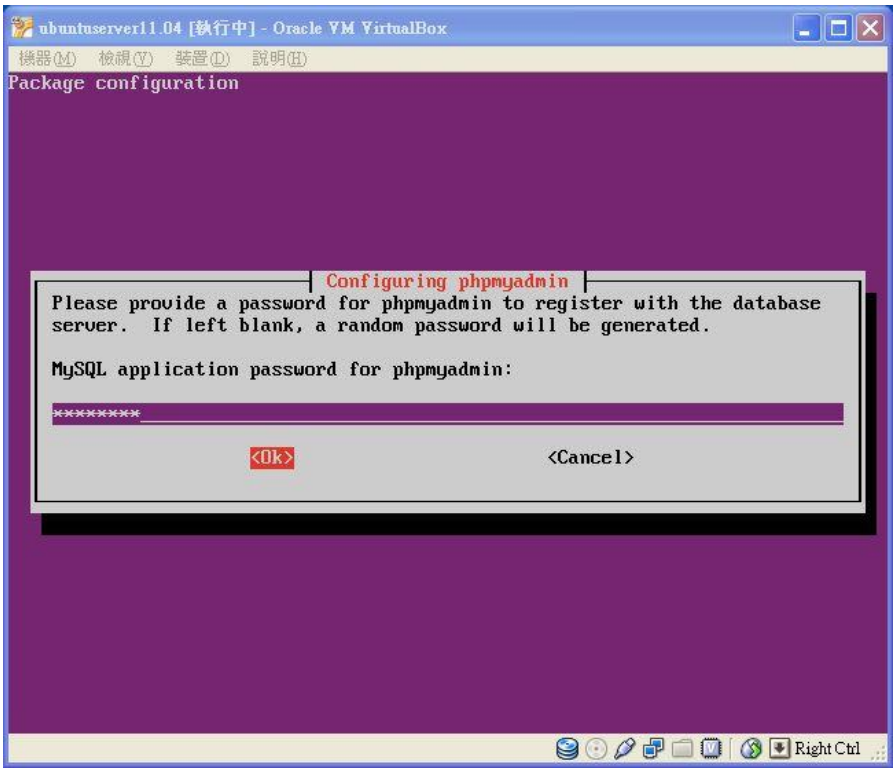


圖 3-3-40 設定密碼註冊資料庫，若是空白則會隨機設定一組

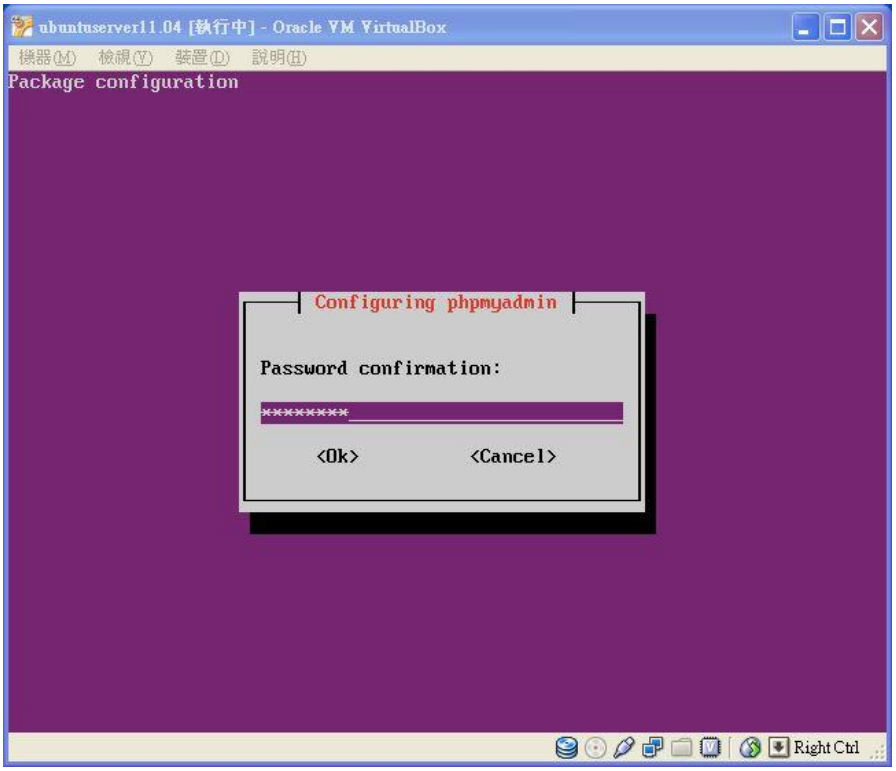


圖 3-3-41 再次確認剛剛輸入的密碼

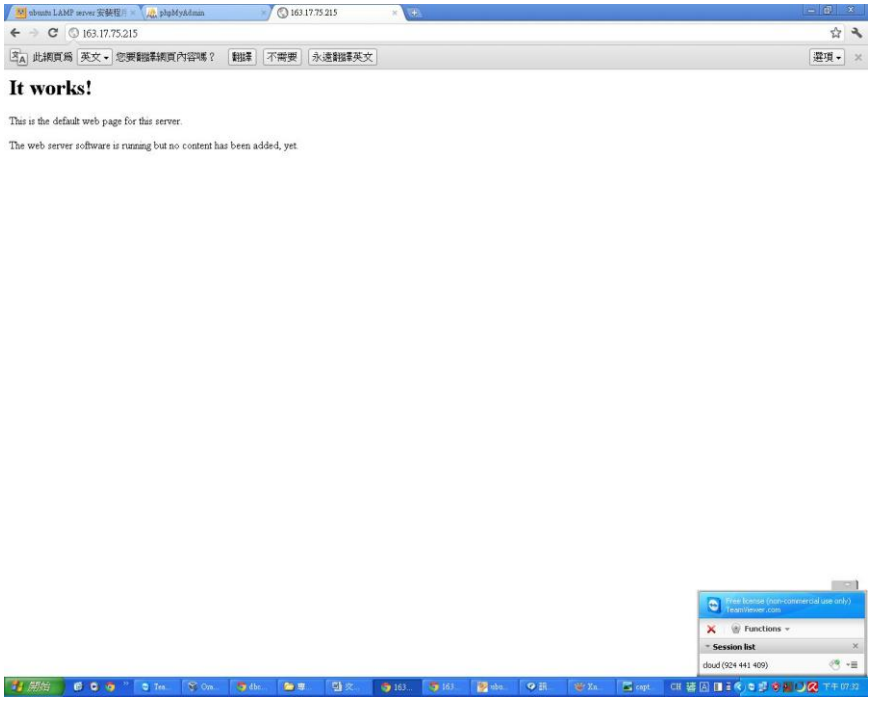


圖 3-3-42 若都安裝完開啟瀏覽器輸入虛擬機器的 IP，會看到 Apache(Web Server)已經可以使用

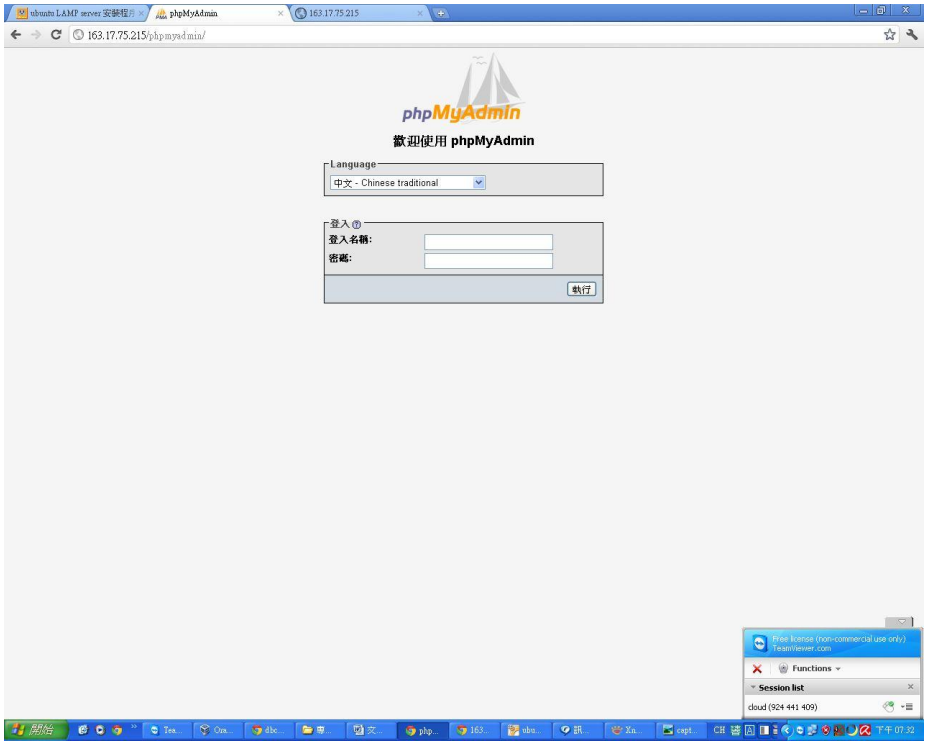


圖 3-3-43 在網址列輸入的 IP 加上/phpmyadmin，則會出現管理資料庫的頁面

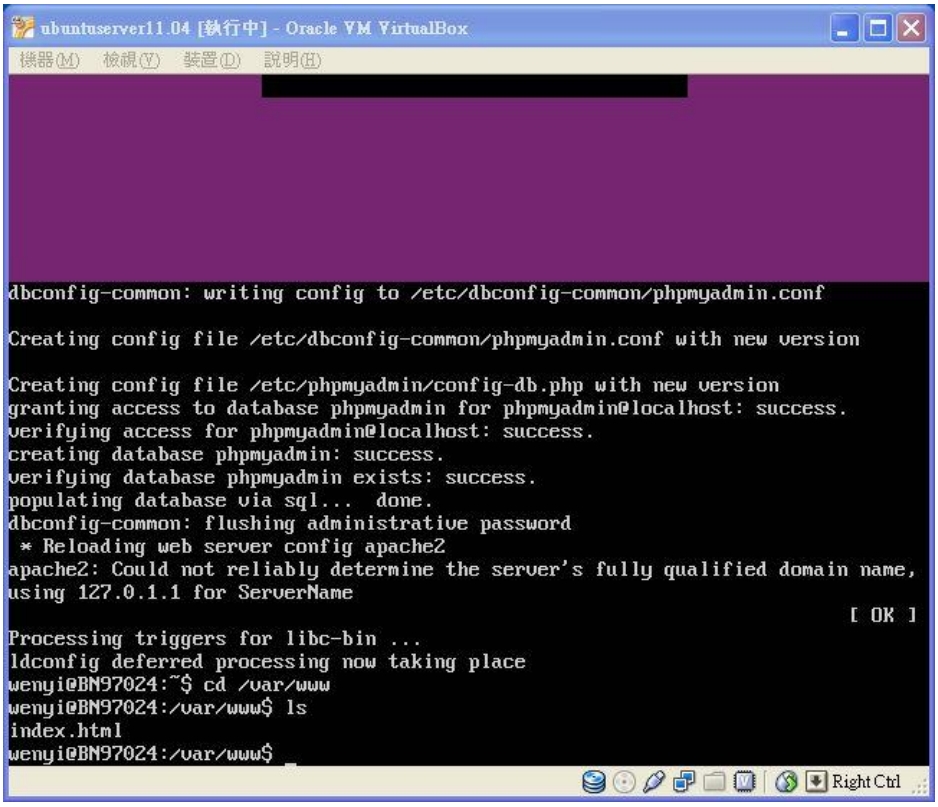


圖 3-3-44 測試 apache WWW 網頁的根目錄 /var/www/index.html

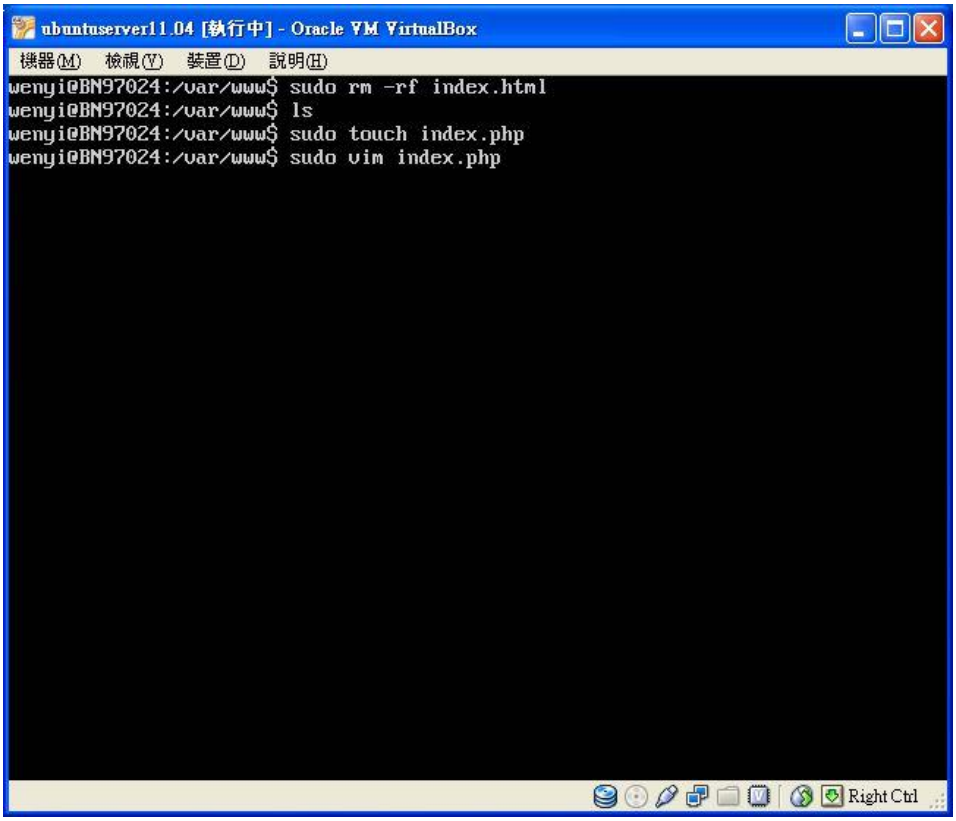


圖 3-3-45 刪除 index.html 新增 index.php

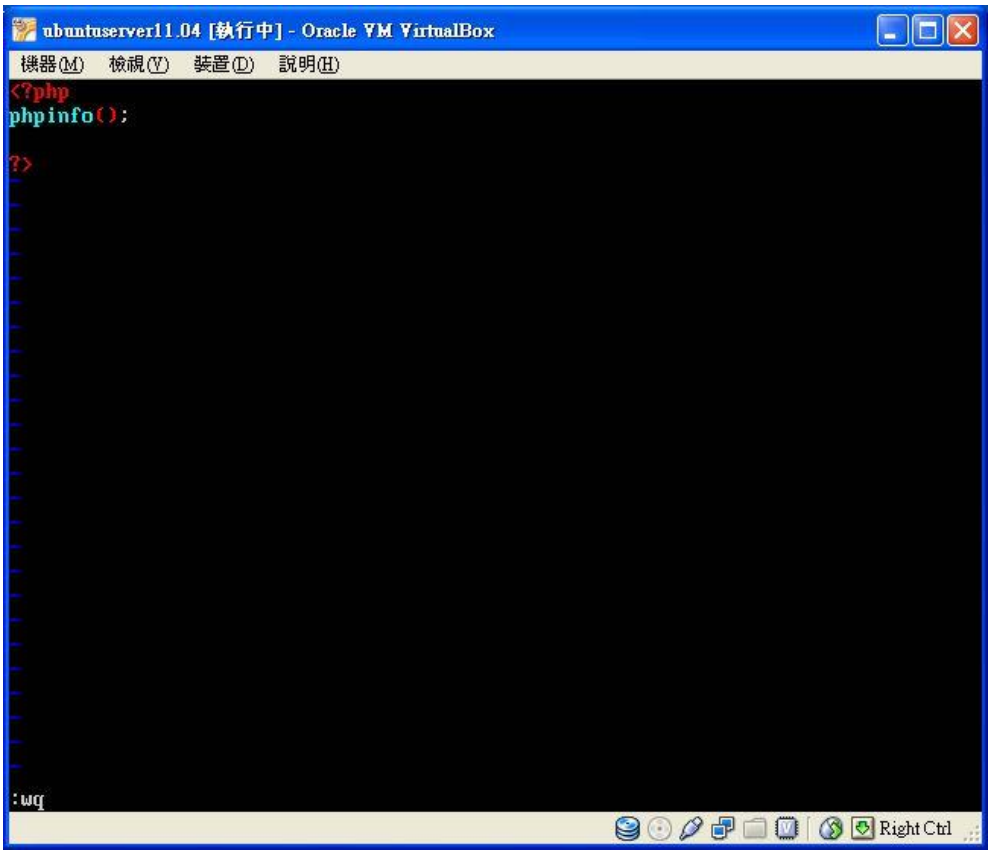


圖 3-3-46 編輯 index.php 使用內建函式，並寫入儲存

圖 3-3-47 上圖寫的函式呈現畫面

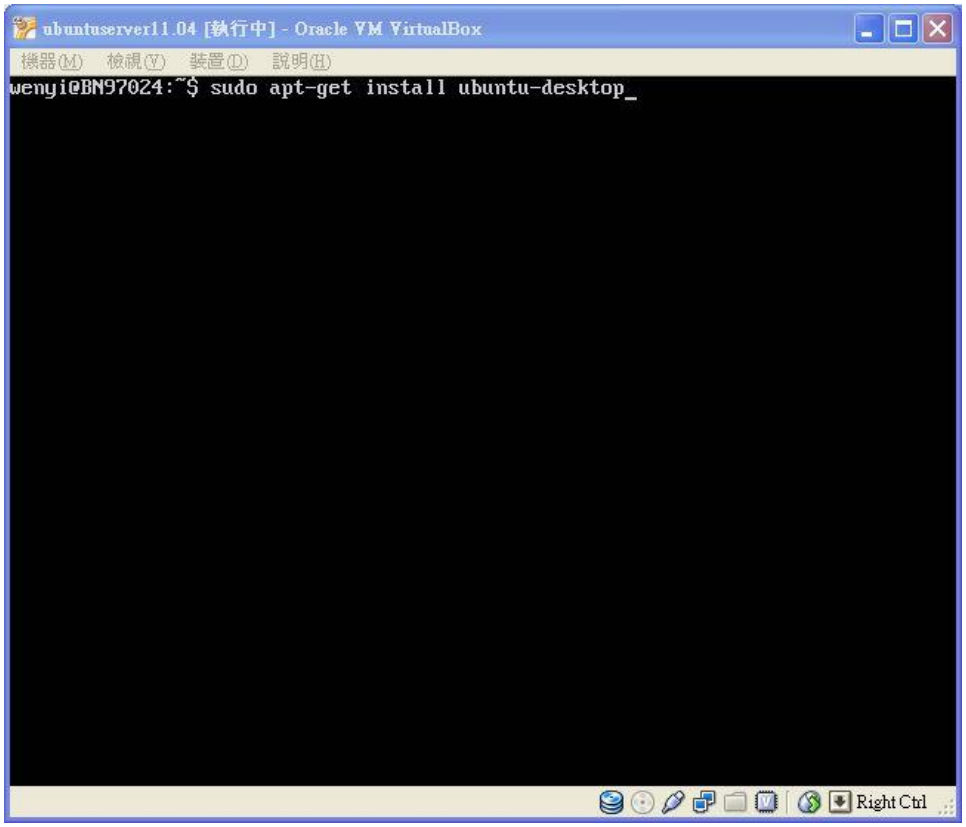


圖 3-3-48 安裝圖形化桌面

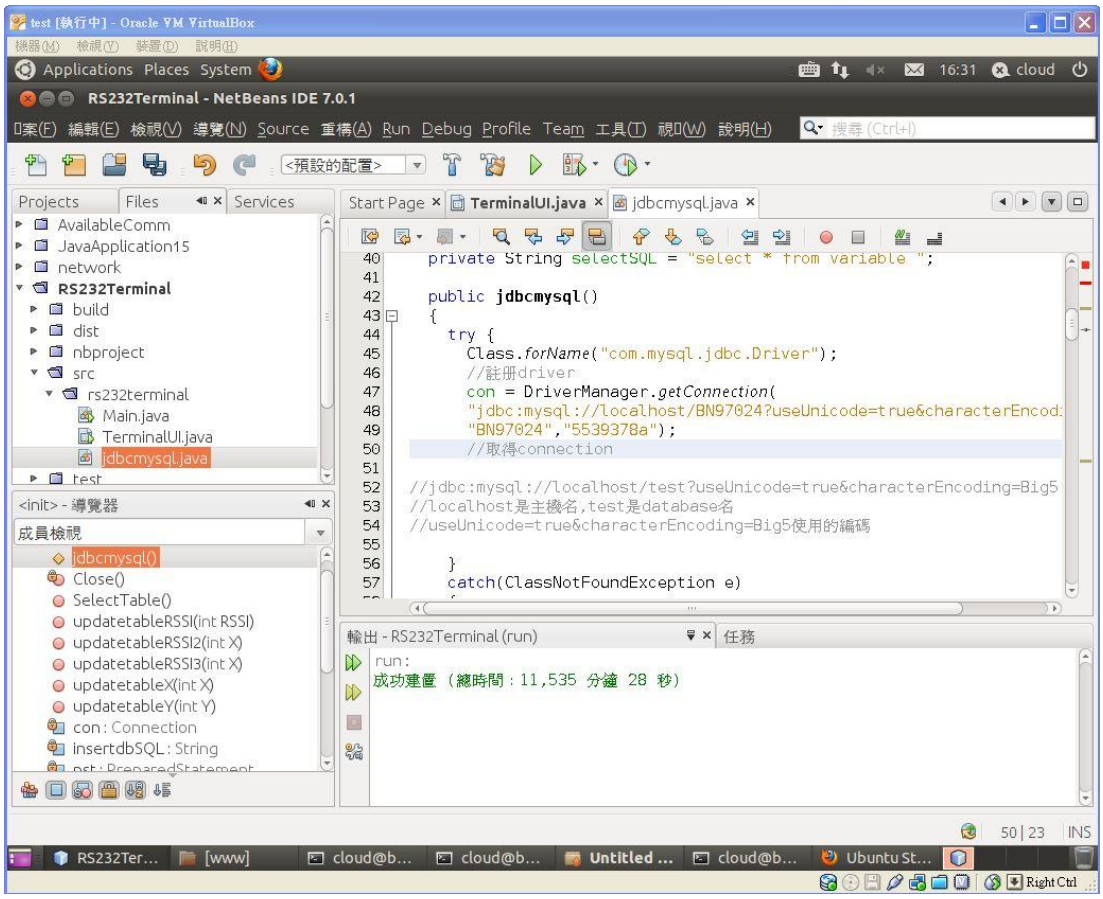


圖 3-3-49 安裝完重新開機會自動使用圖型化介面開機

若要切換回文字介面，按 Ctrl+Alt+F1~F7，切換圖形化 Ctrl+Alt+F8，這樣若要編輯程式碼或是要找檔案會比較方便，需要穩定跑時就切換成文字介面。

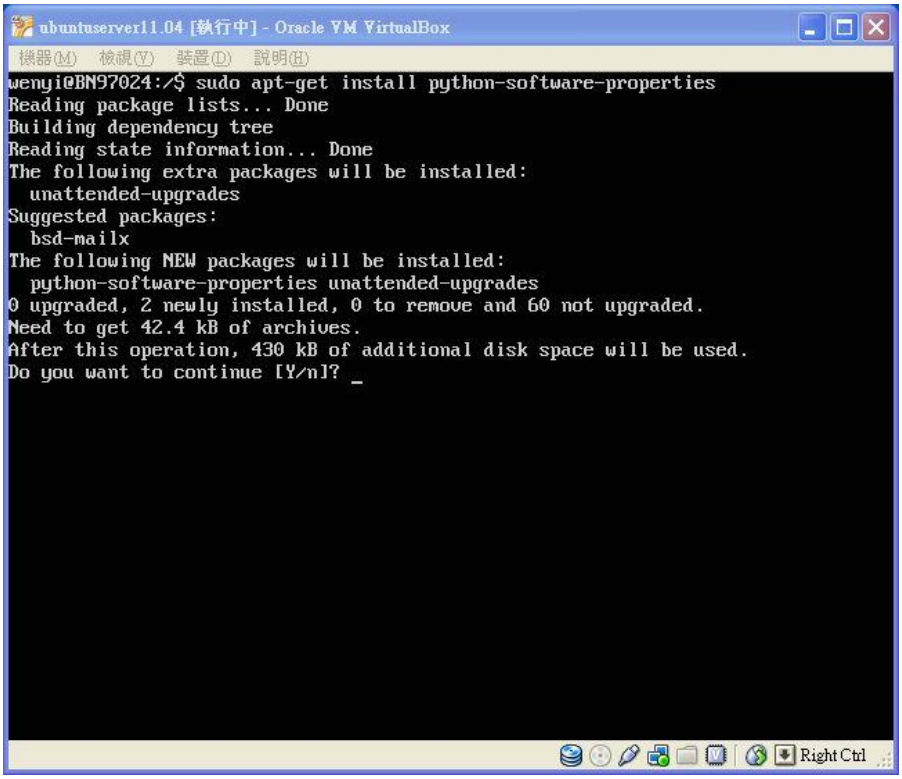


圖 3-3-50(使用圖型化介面編輯程式)安裝 oracle JDK

因為 apt-get 上的列表沒有 JDK，所以需要加入並更新下載站點，但是預設也沒有更新站點的套件，所以要先安裝才能夠更新列表。

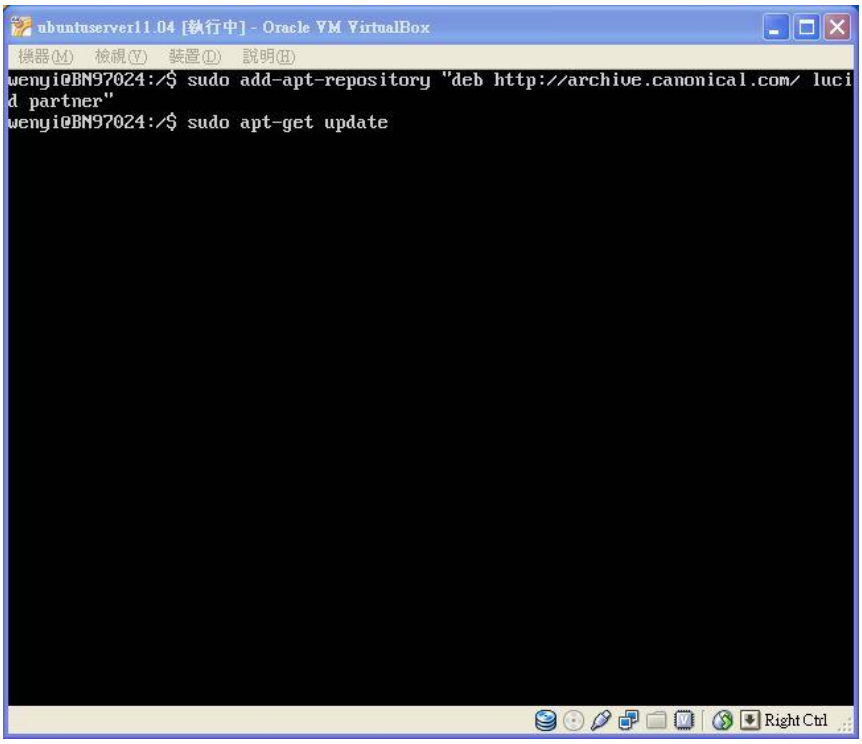


圖 3-3-51 安裝完後使用 apt-get-repository 來新增下載站點並更新套件列表。

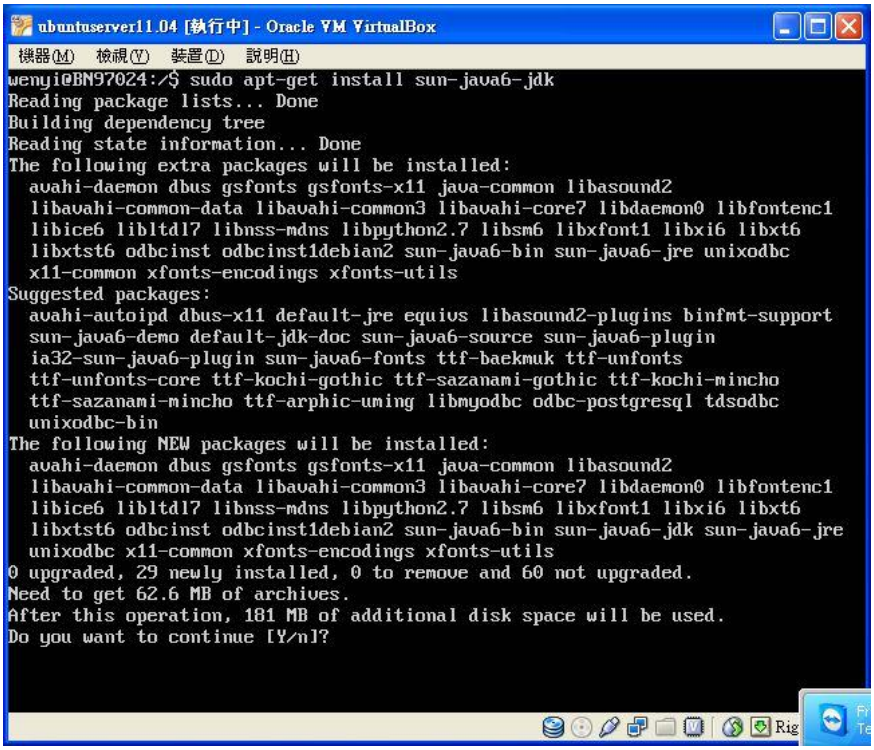


圖 3-3-52 就可以使用 apt-get 安裝 JDK

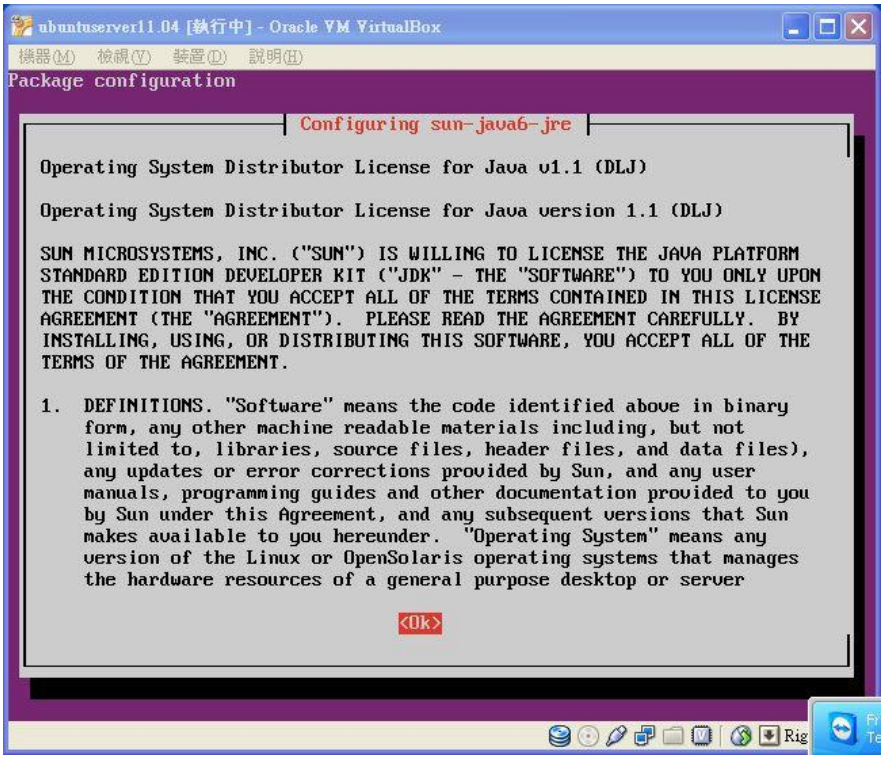


圖 3-3-53 安裝 JDK，會問你是否要繼續。

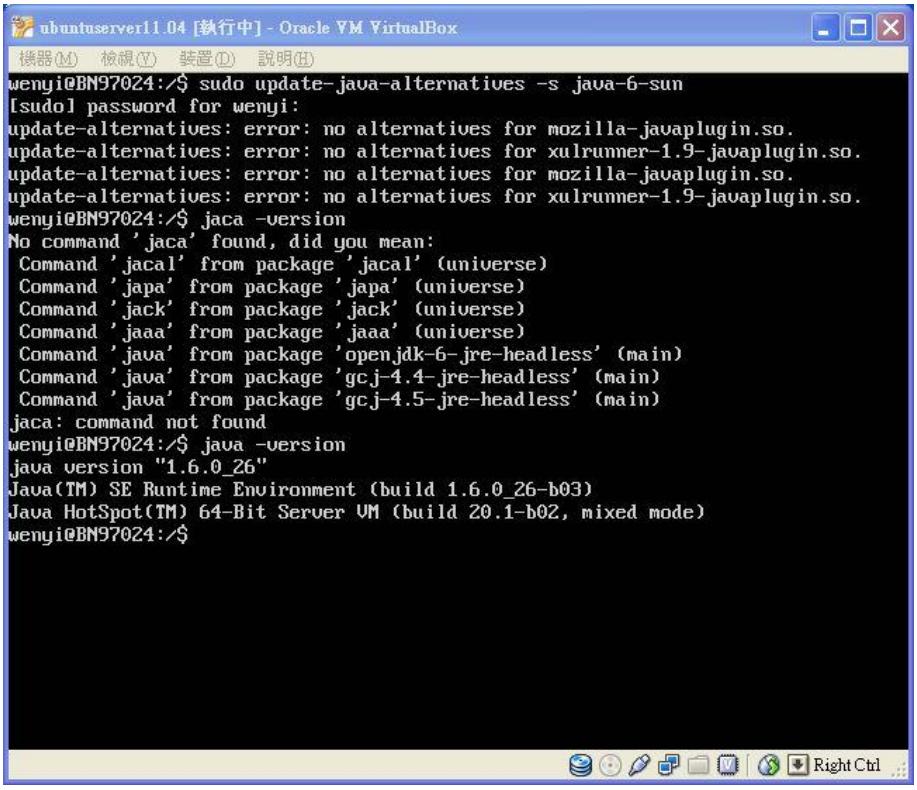


圖 3-3-54 安裝完成之後使用 java –version 檢查是否成功

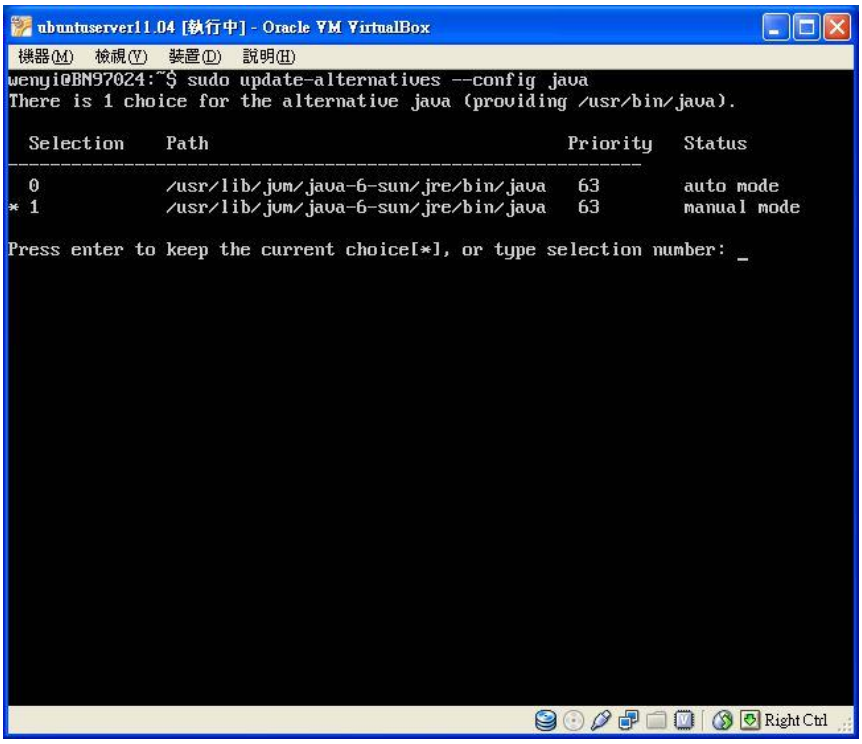


圖 3-3-55 需要將本來預設的 openjdk 改成剛才安裝的 Jdk。

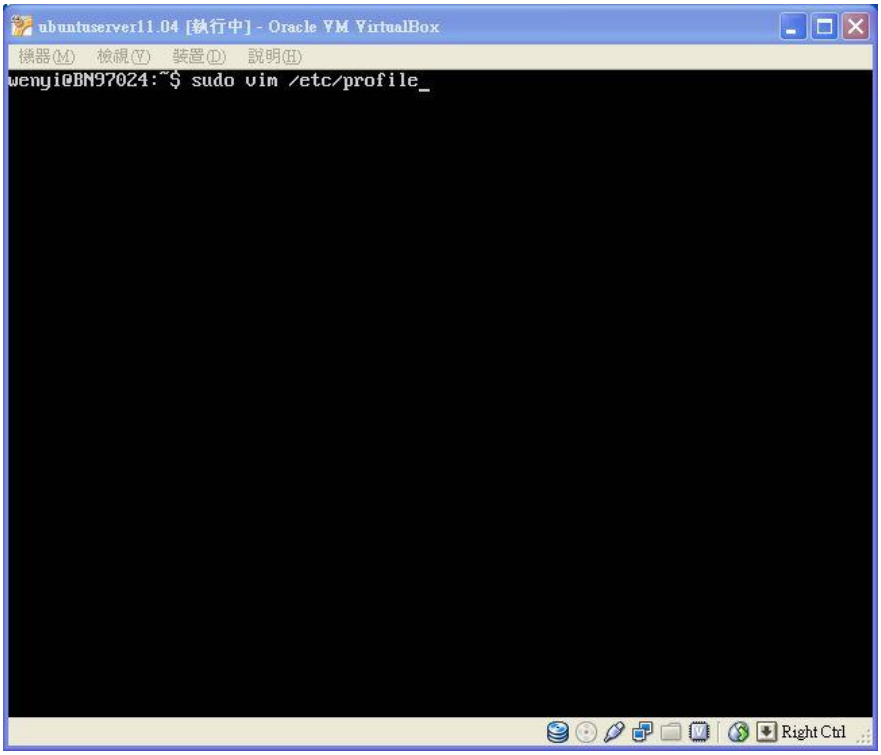


圖 3-3-56 Vim /etc/profile 編輯檔案，設定 JAVA 環境變數

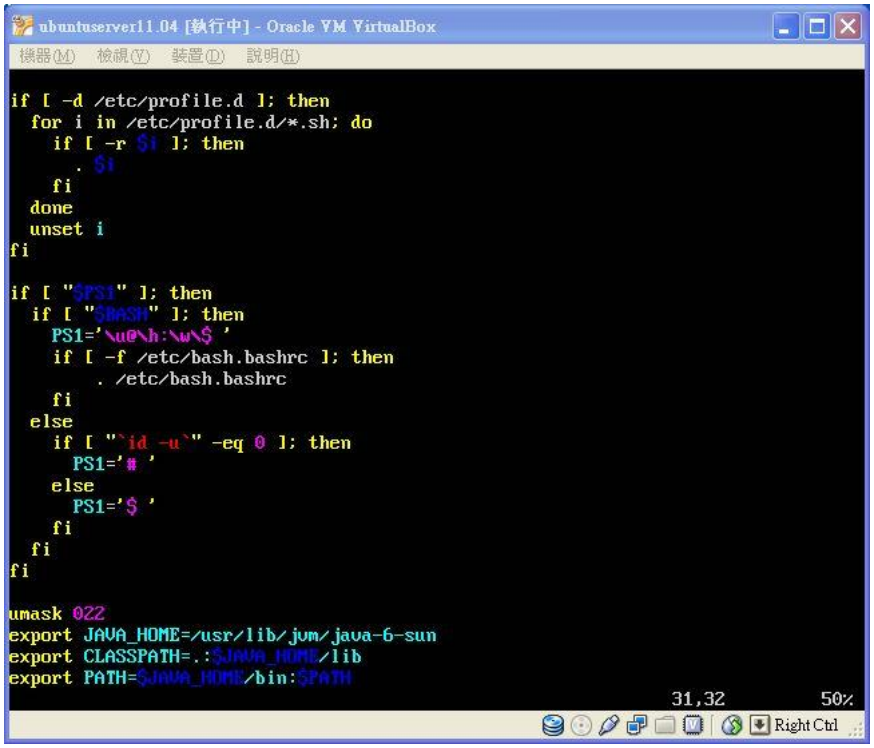


圖 3-3-57 在最後面加入 export JAVA_HOME=/usr/lib/jvm/java-6-sun

export CLASSPATH=.:$JAVA_HOME/lib

export PATH=$JAVA_HOME/bin:$PATH

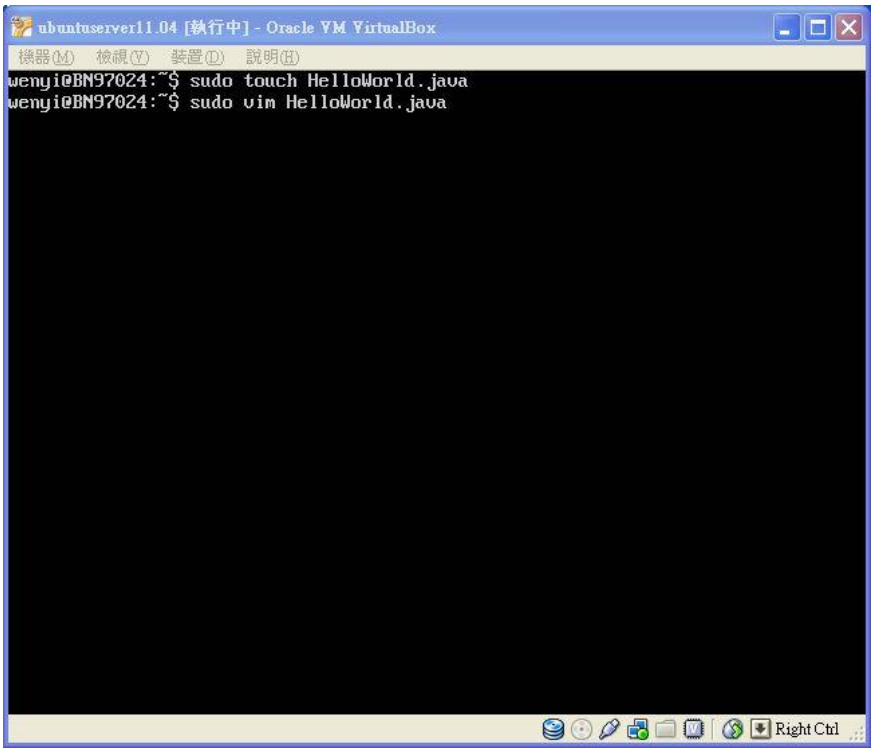


圖 3-3-58 新增 HelloWorld.java

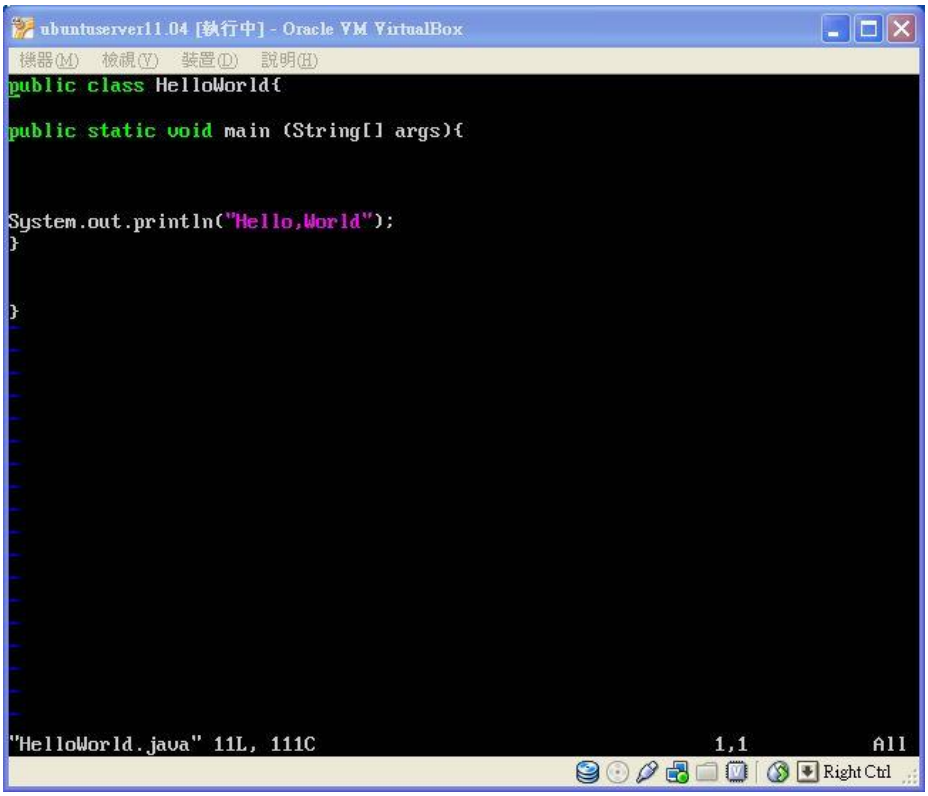


圖 3-3-59 編輯 Helloworld.java

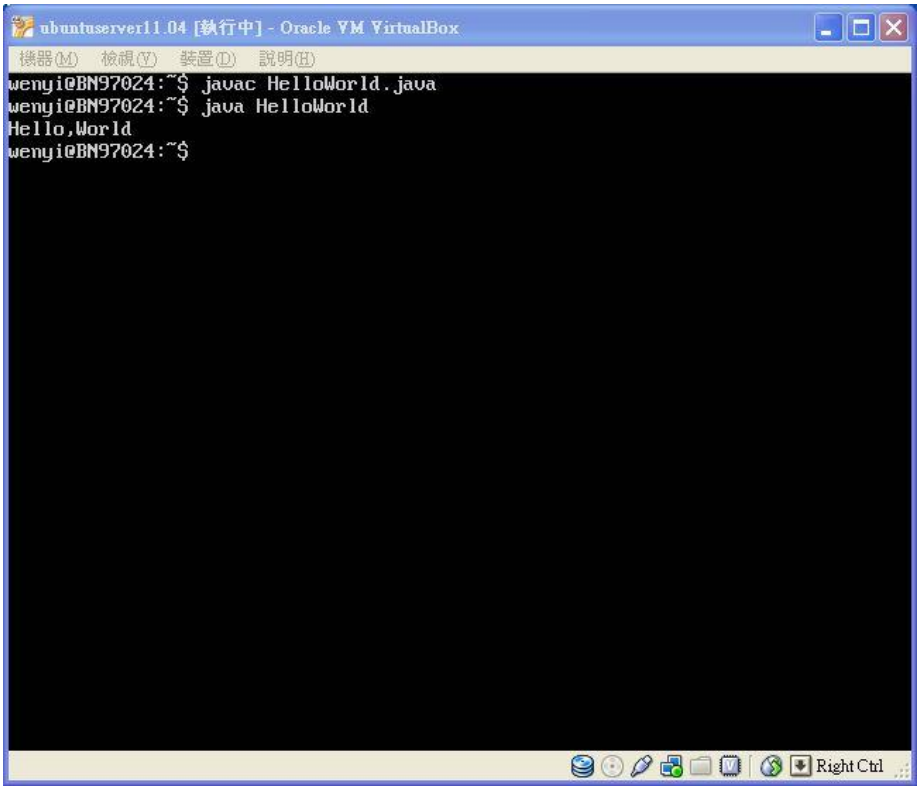


圖 3-3-60 測試執行 JAVA(HelloWorld)

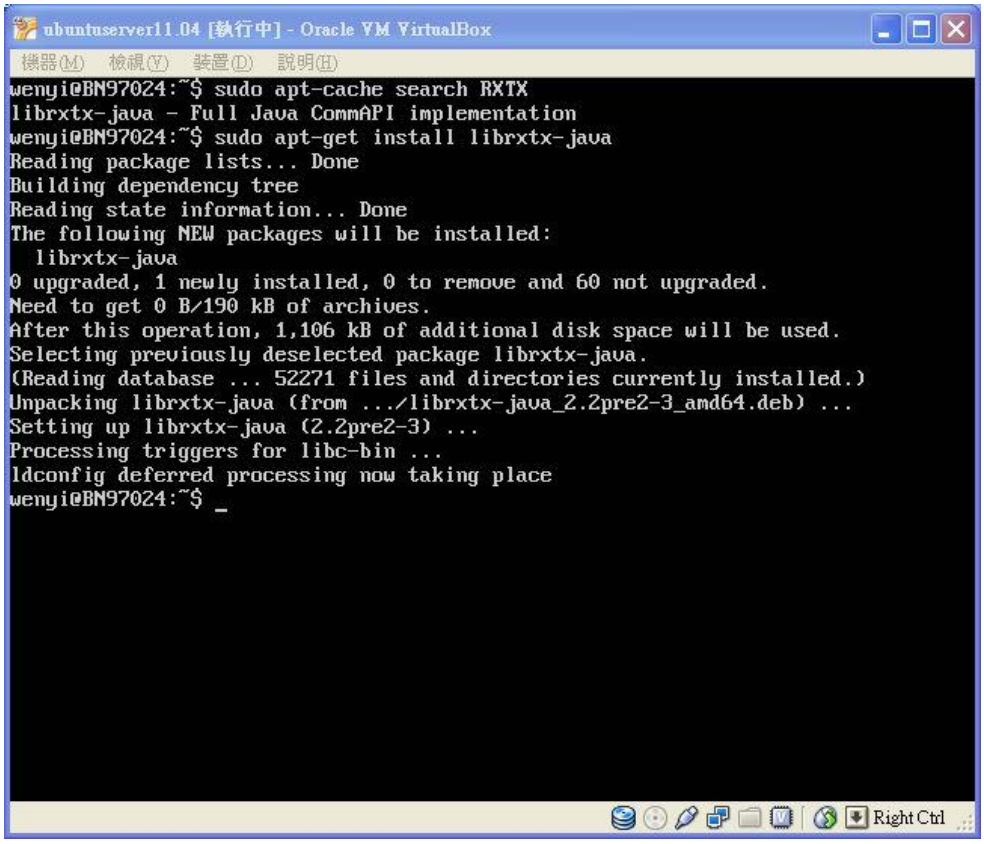


圖 3-3-61 安裝 RXTX，安裝完會增加下列檔案:

/usr/lib/jni/librxtxI2C-2.1-7.so
/usr/lib/jni/librxtxI2C.la
/usr/lib/jni/librxtxI2C.so
/usr/lib/jni/librxtxParallel-2.1-7.so
/usr/lib/jni/librxtxParallel.la
/usr/lib/jni/librxtxParallel.so
/usr/lib/jni/librxtxRS485-2.1-7.so
/usr/lib/jni/librxtxRS485.la
/usr/lib/jni/librxtxRS485.so
/usr/lib/jni/librxtxRaw-2.1-7.so
/usr/lib/jni/librxtxRaw.la
/usr/lib/jni/librxtxRaw.so
/usr/lib/jni/librxtxSerial-2.1-7.so
/usr/lib/jni/librxtxSerial.la
/usr/lib/jni/librxtxSerial.so
/usr/share/doc/librxtx-java/AUTHORS
/usr/share/doc/librxtx-java/README.Debian.gz
/usr/share/doc/librxtx-java/RMISecurityManager.html

/usr/share/doc/librxtx-java/SerialPortInstructions.txt.gz
/usr/share/doc/librxtx-java/changelog.Debian.gz
/usr/share/doc/librxtx-java/changelog.gz
/usr/share/doc/librxtx-java/copyright
/usr/share/java/RXTXcomm.jar

其中需要將
/usr/share/java/RXTXcomm.jar
/usr/lib/jni/librxtxParallel.so
/usr/lib/jni/librxtxSerial.so

將/usr/share/java/RXTXcomm.jar
複製到/usr/lib/jvm/java-6-sun/jre/lib/ext
將/usr/lib/jni/librxtxParallel.so
/usr/lib/jni/librxtxSerial.so
複製到/usr/lib

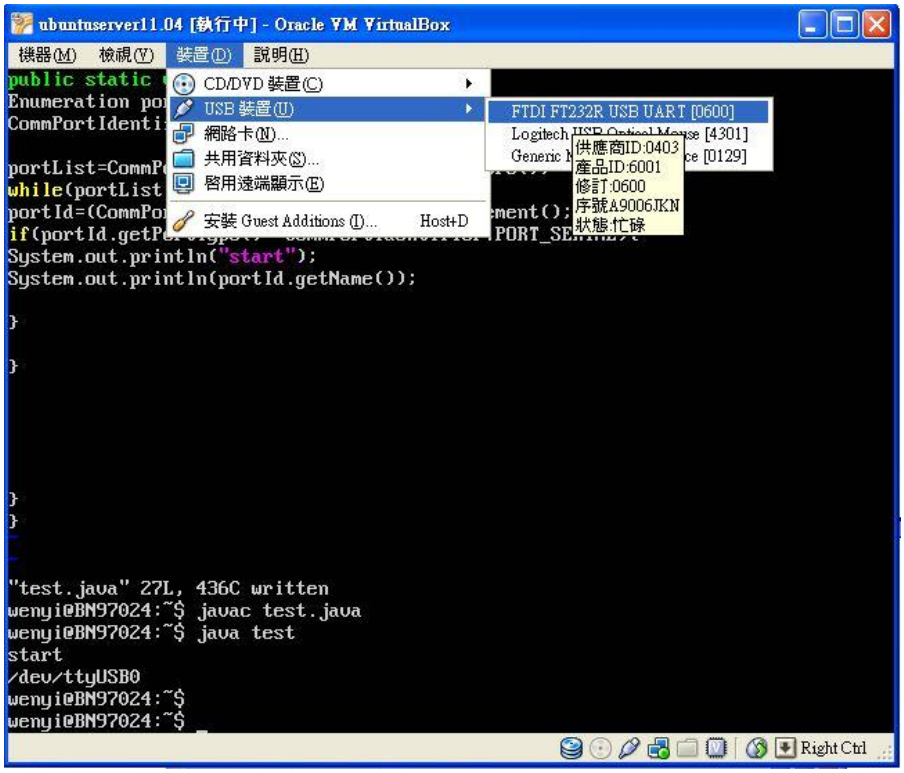


圖 3-3-62 在虛擬機器上的裝置=>USB 裝置=>找到 BOE 板裝置

將 BOE 板和電腦上的 USB 連結，在 Virtual BOX 的裝置=>USB 裝置=>FTDI FT232R USB UART ，安裝驅動，這樣實體 USB 就可以跟 Virtual BOX 上的系統連結使用。

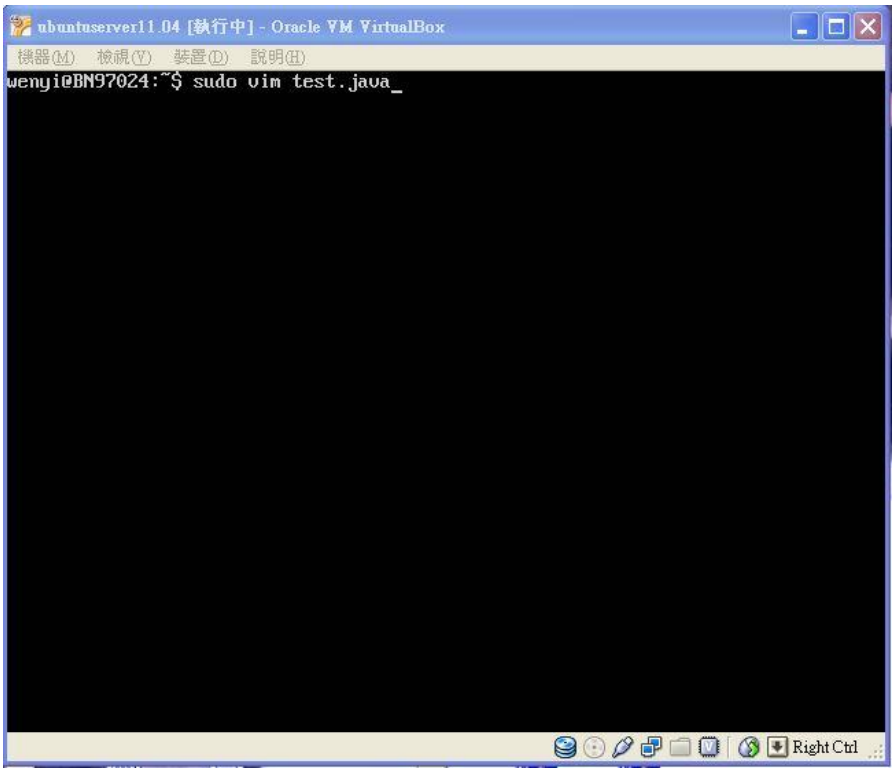


圖 3-3-63 編輯 test.java

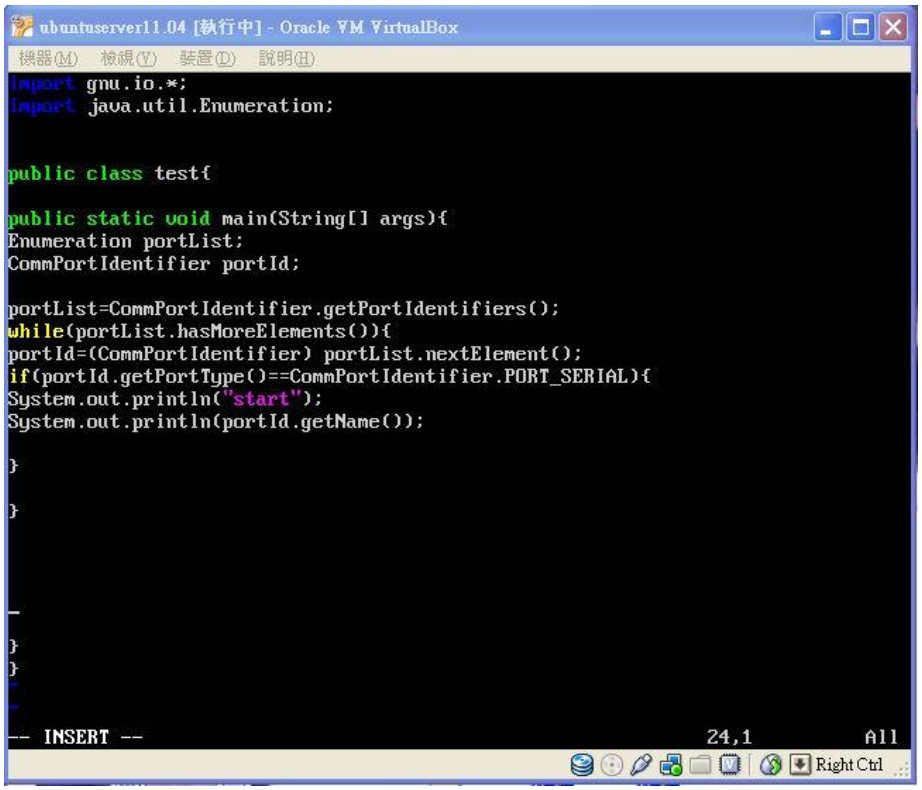


圖 3-3-64 Import RXTX 包的套件 gnu.io.*鍵入程式碼，功用為找到插入 USB 的 RS232 硬體

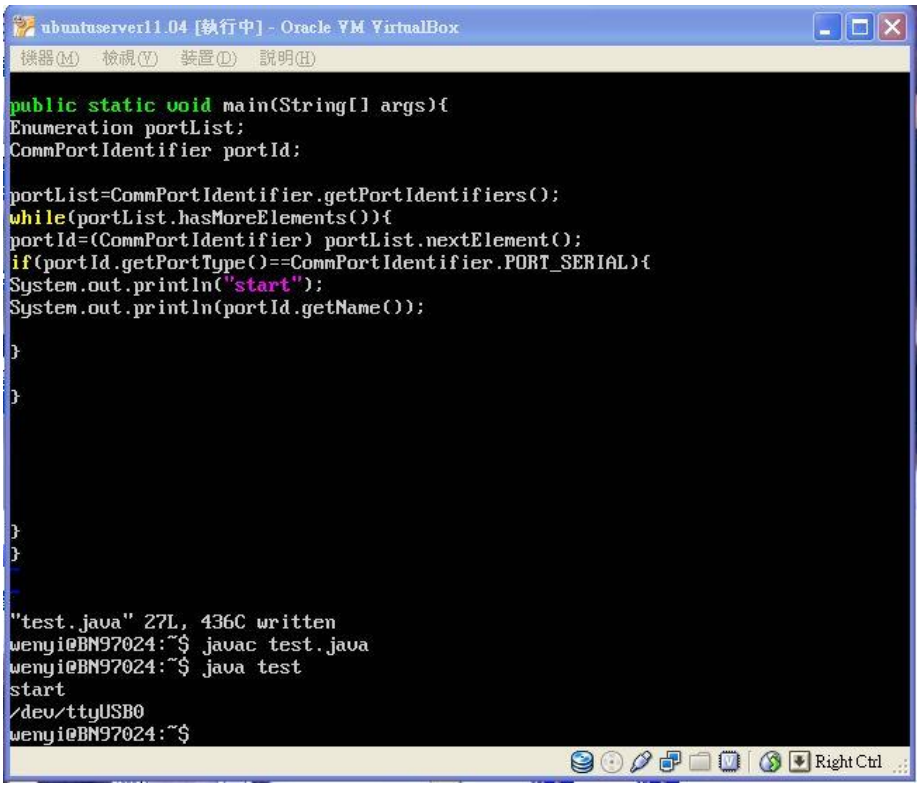


圖 3-3-65 編譯後執行 test，可以看到有顯示出/dev/ttyUSB0，就是該硬體的編號。

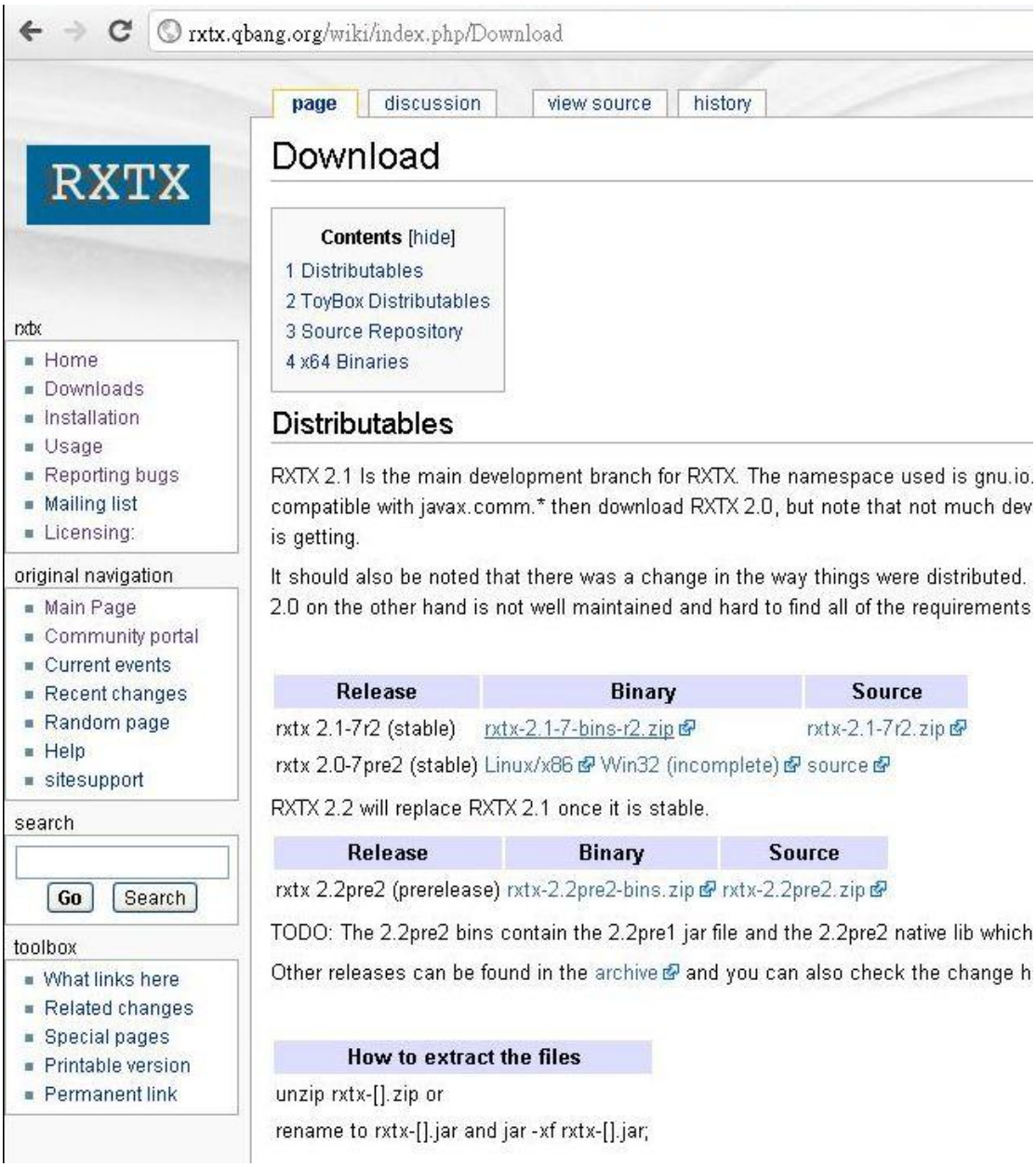


圖 3-3-66 在 Windows 安裝 RXTX，下載 rtx-2.1-7bins-r2.zip

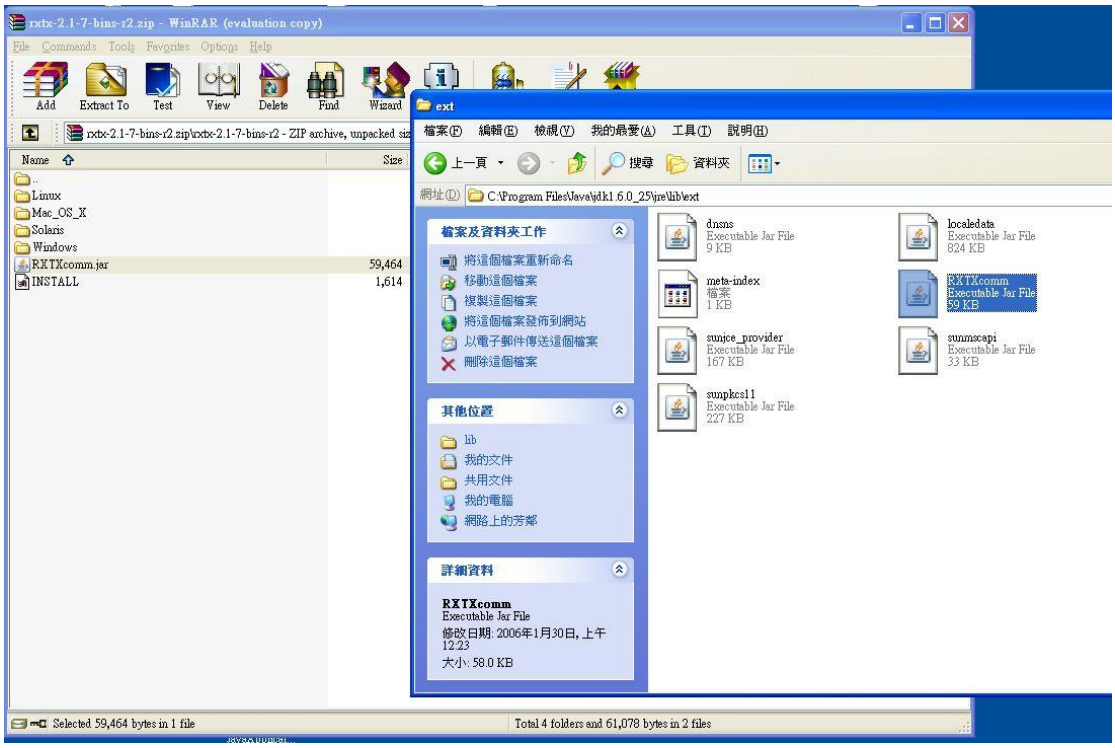


圖 3-3-67 將壓縮檔打開

將 RxTxcomm.jar 解壓縮到 C:\Program Files\JAVA\jdk1.6.0_25\jre\lib\ext

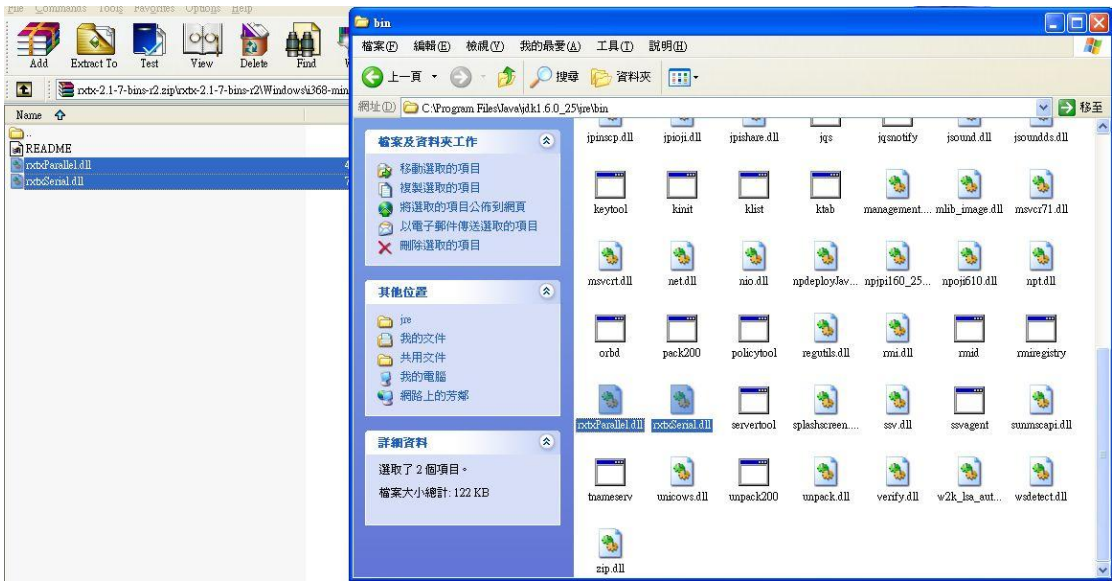


圖 3-3-68 將 rxtxParallel.dll 跟 rxtxSerial.dll 解壓縮到 C:\Program Files\Java\jdk1.6.0_25\jre\bin

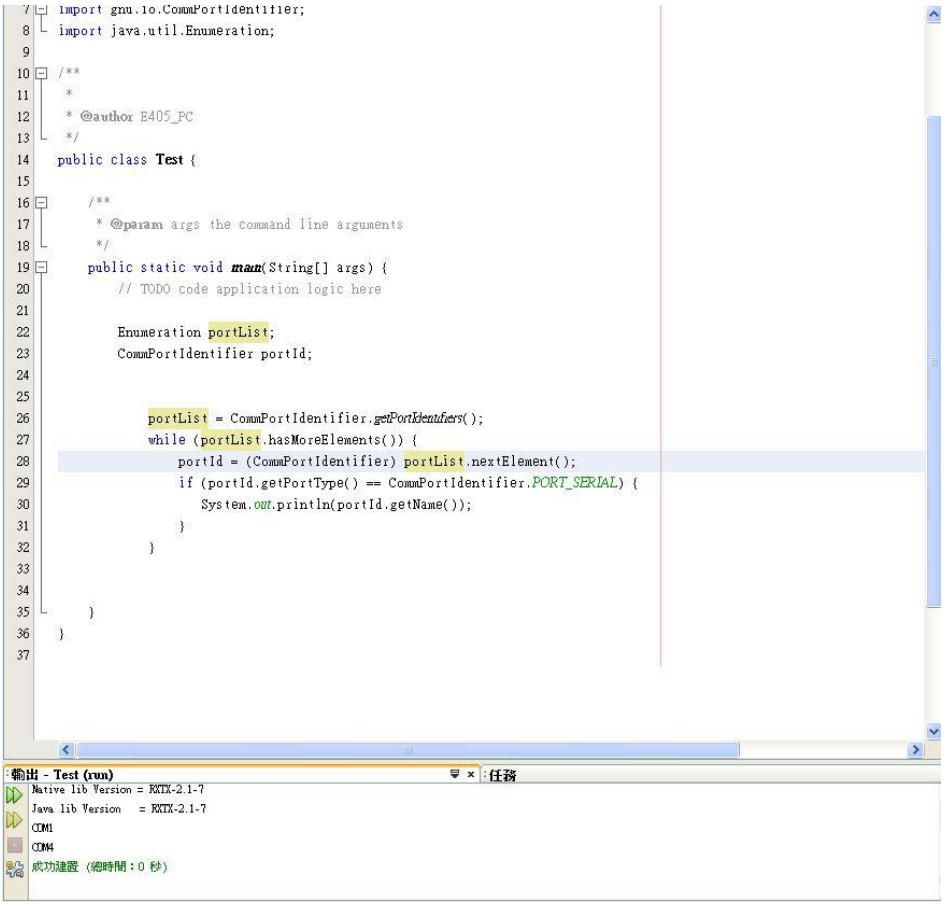


圖 3-3-69 測試:使用 NetBeans IDE 編輯程式(同 Ubuntu 測試 RXTX)並編譯執行

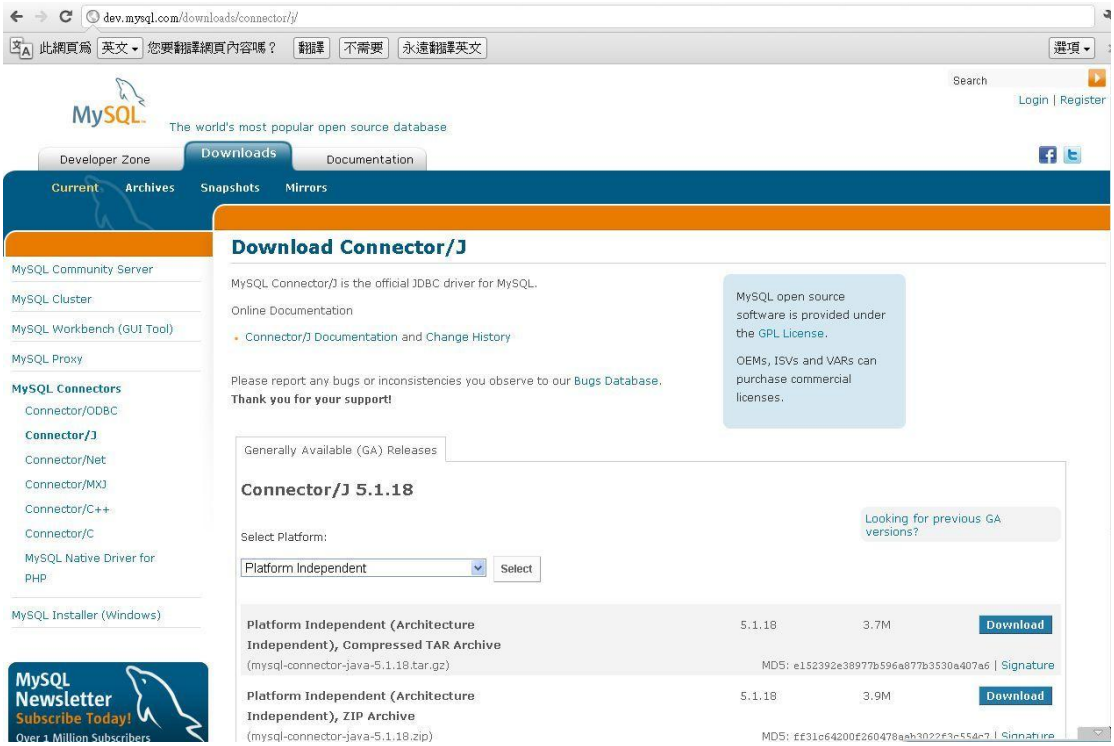


圖 3-3-70 下載 JDBC 的套件，在 Mysql 的網站下載

```
14
15
16  import java.sql.*;
17
18  public class JavaApplication3 {
19      public static void main(String[] args) {
20          String driver = "com.mysql.jdbc.Driver";
21          try {
22              Class.forName(driver);
23              Connection conn = DriverManager.getConnection(
24                      "jdbc:mysql://localhost:3306/BN97024",
25                      "[redacted]", "[redacted]");
26
27              if(conn != null && !conn.isClosed()) {
28                  System.out.println("資料庫連線測試成功！");
29                  conn.close();
30              }
31
32          }
33          catch(ClassNotFoundException e) {
```

輸出 - JavaApplication3 (run) | 任務

```
run:
資料庫連線測試成功！
成功建置 (總時間：1 秒)
```

圖 3-3-71 測試:將下載下來的檔案放到

(Windows) C:\Program Files\JAVA\jdk1.6.0_25\jre\lib\ext
(Ubuntu)/usr/lib/jvm/java-6-sun-1.6.0.26/jre/lib/ext
同樣使用 NetBeans IDE 編輯程式並編譯執行

第四章 救難自走車系統架構與運作說明

本章要點:

本系統係將 ZigBee 裝載在機器車上，透過 ZigBee 傳輸的訊號強度(RSSI)，來測得其大約距離，再把訊號強度(RSSI)回傳到電腦(RS232-RXTX)，電腦裡有虛擬機器並建立 Web Server，系統收到後傳至虛擬機器內的系統,則系統內建置需要的軟體並能將收到的訊號強度(RSSI)呈現在網頁上，並且在 PHP 上比對訊號強度(RSSI)的大小，比對完畢後，會傳回於監控端，最後依其判斷，距離失火端最近的救難車，來前往救火。

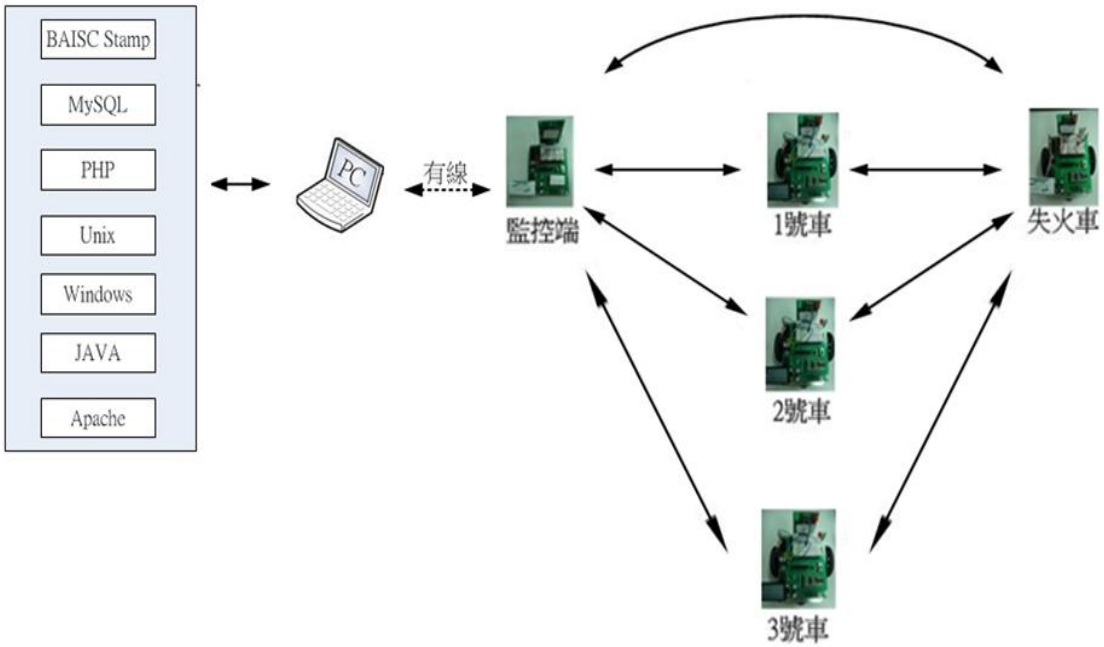


圖 4-0-1 整體圖

左邊為電腦會使用到的軟體,右邊為機器車使用 ZIGBEE 來傳輸 RSSI 及其他資料。

4-1 系統功能

本系統共分成四個角色：監控端（中心端）、失火端、救難端（我們使用小型機器車來模擬，後面章節將統稱為失火車與救難車）、PC端四個角色。

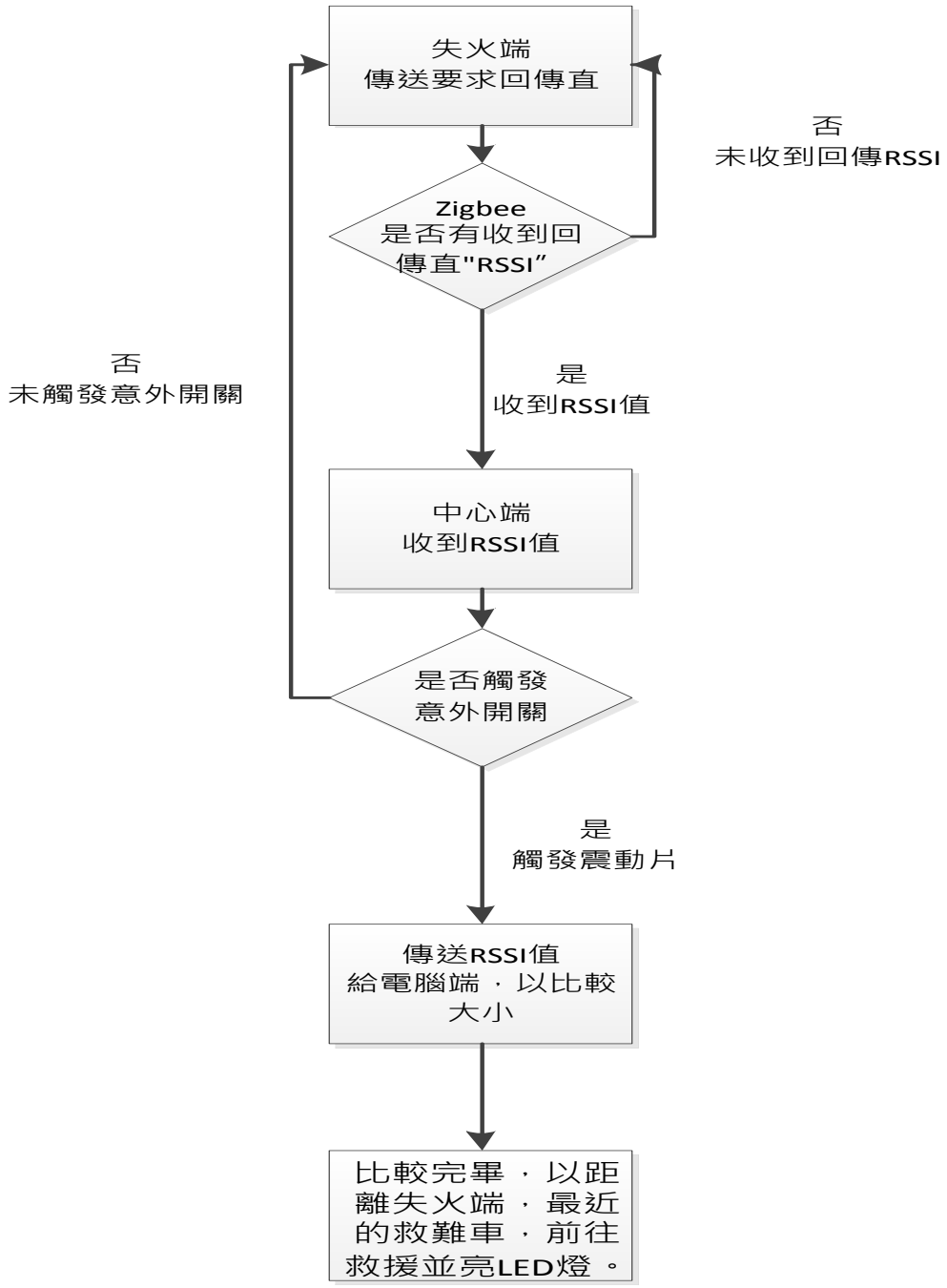


圖 4-1-1 為整體流程圖

監控端（中心端）：

用來監控失火車與三台救難車的訊號強度和失火車是否著火了，當失火車發生意外，著火了！著火了！監控端將會收到失火車所發出的求救訊號，並且 LED 亮起（警示燈），然而發出救援訊號給距離最近的救難車，如圖 4-1-2。

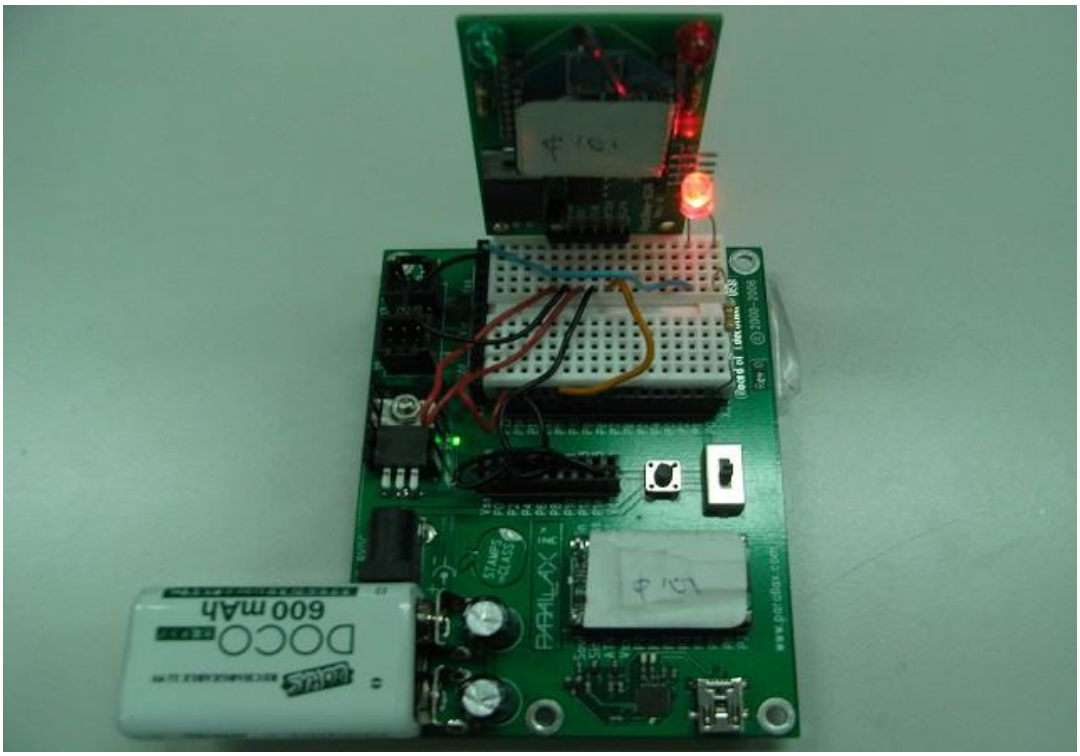


圖 4-1-2 收到失火車求救訊號亮 LED。

主要中心端流程圖

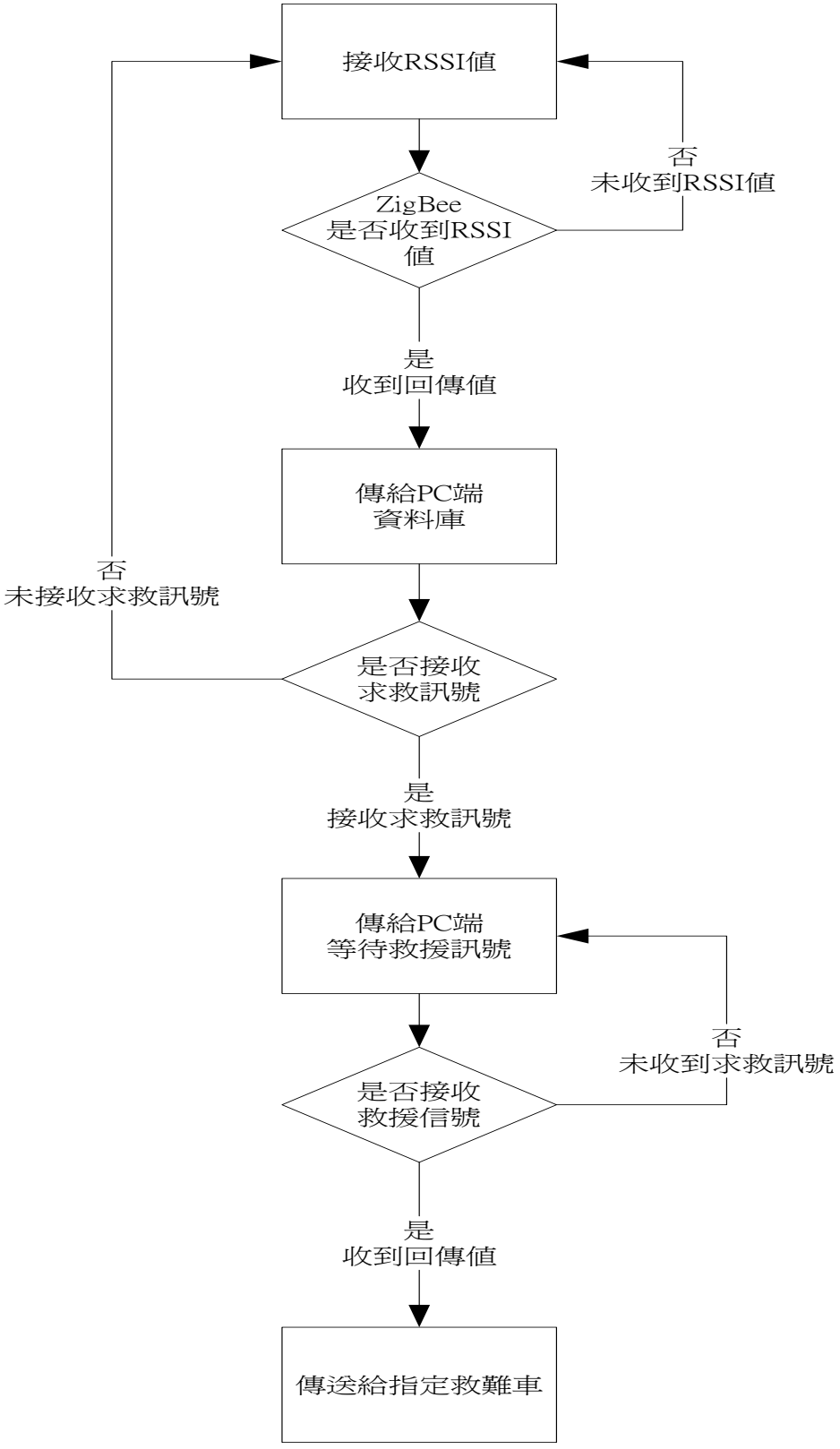


圖 4-1-3 監控端流程圖。

失火車：

用來發出要求回傳訊號給三台救難車，接收救難車回傳的訊號強度（RSSI 值），並將其值傳送給監控端，一直不斷重複執行，以及觸碰意外開關（震動片），用來表示失火車已經著火了，並透過發出求救訊號給監控端，等待求救，如圖 4-1-4。

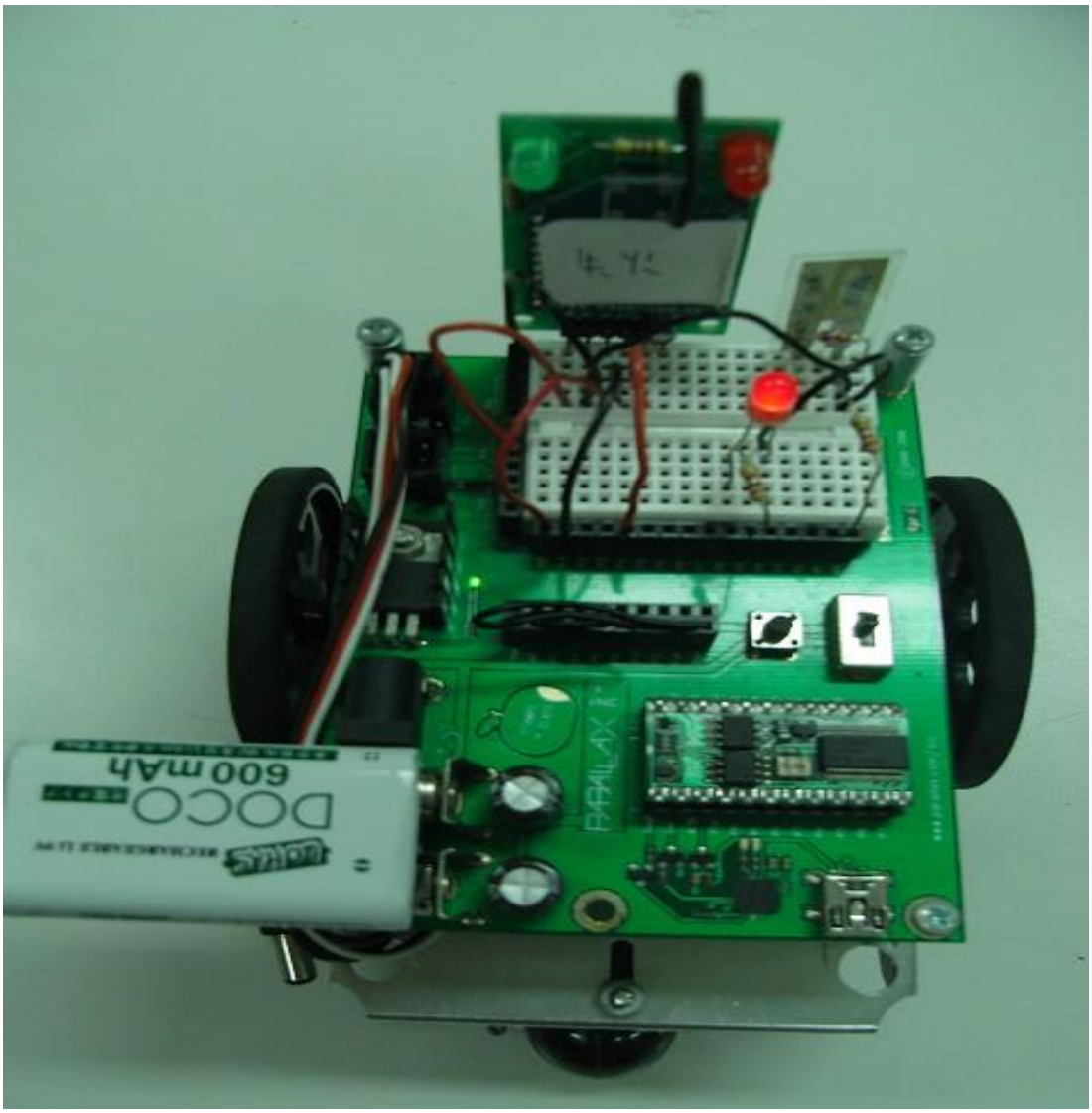


圖 4-1-4 觸碰開關啟動，表示失火，亮 LED。

主要失火車流程圖

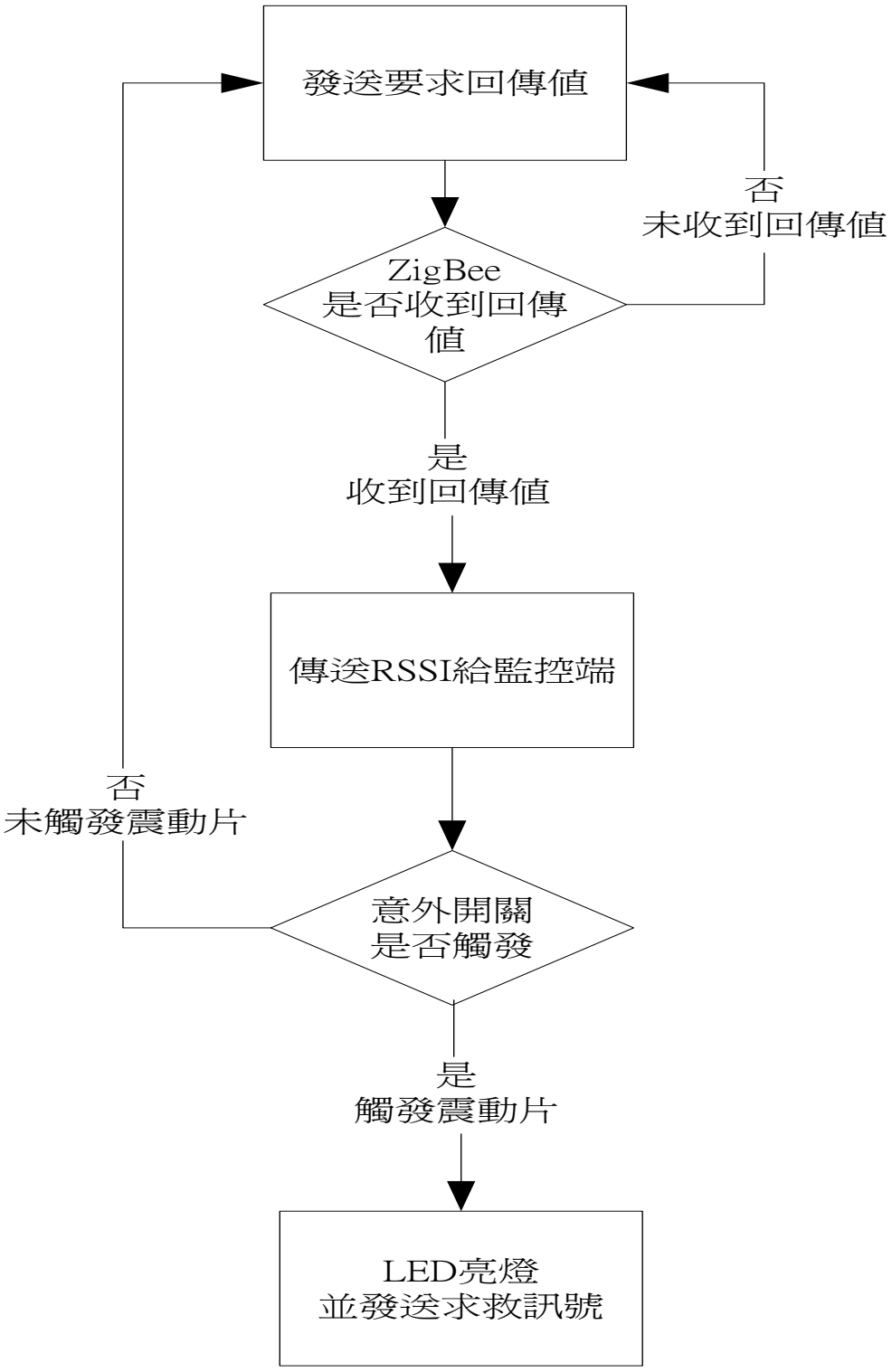


圖 4-1-5 失火車流程圖。

救難車：

用來接收失火車所發出的訊息，並將其回傳，等待下個訊息，一直重複執行，直到接收到監控端所發出的救援訊息，然而 LED 亮起，表示收到救援訊息，如圖 4-1-6。

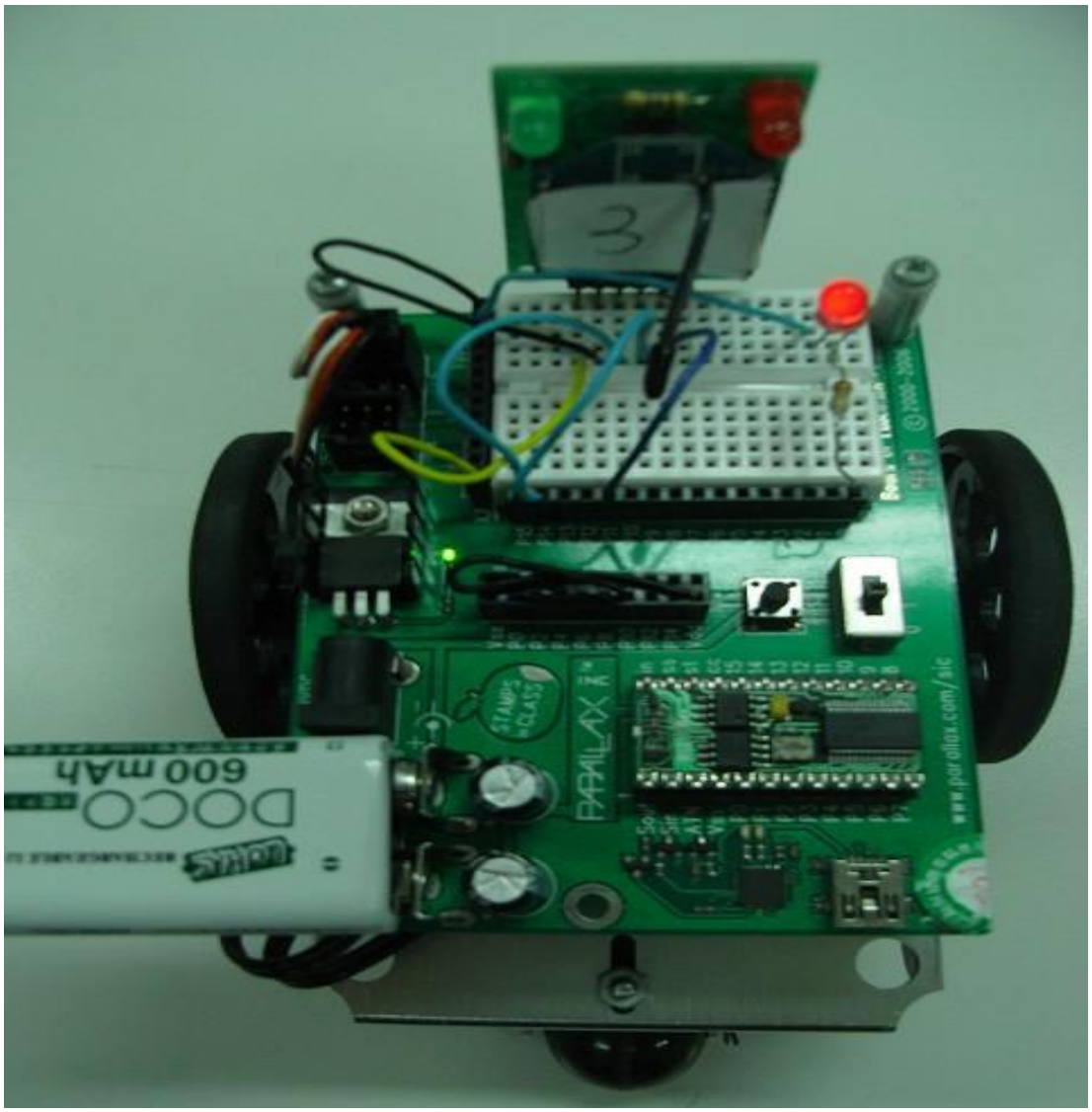


圖 4-1-6 收到救援訊息，亮 LED。

主要救難車流程圖

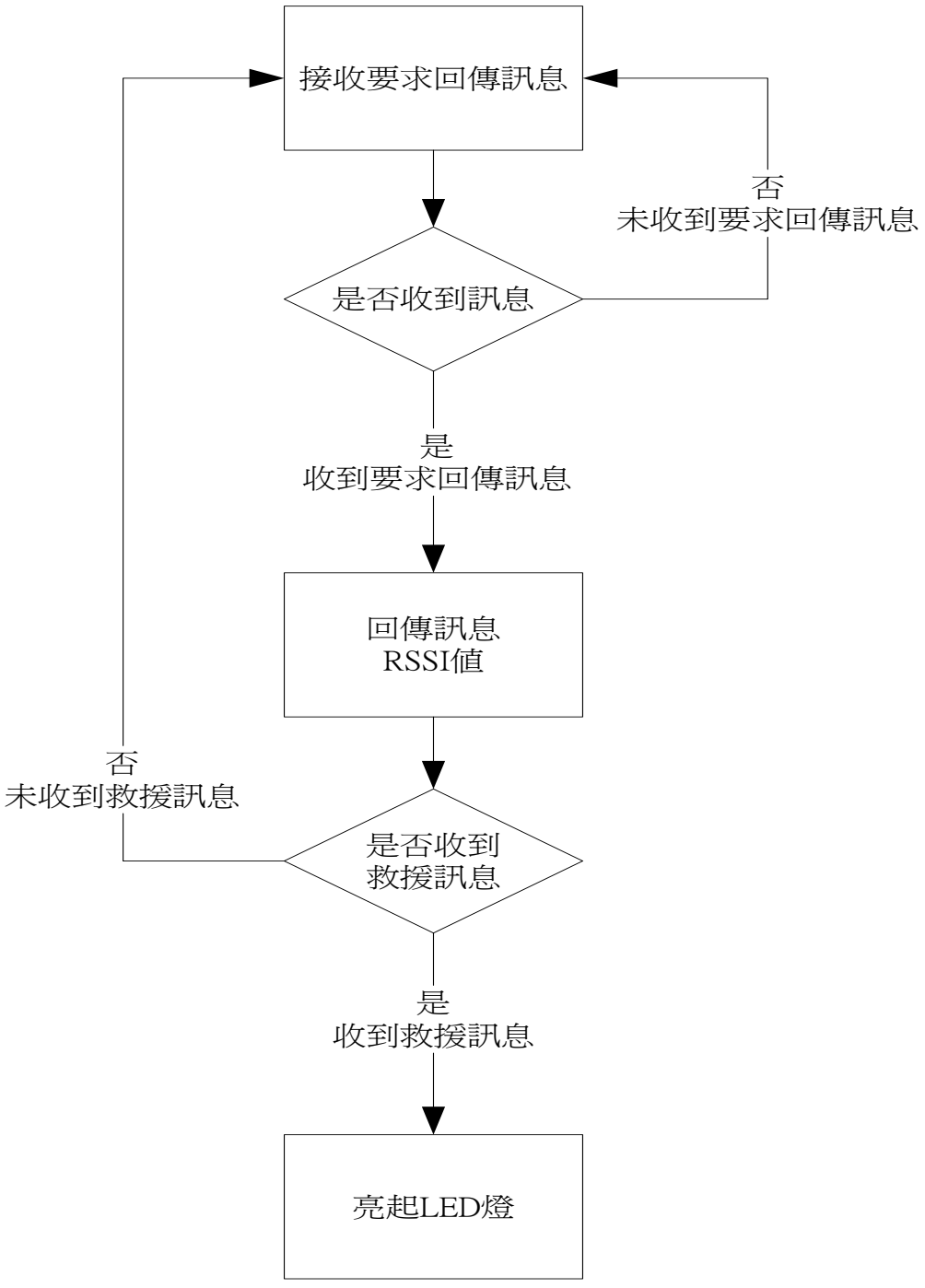


圖 4-1-7 救難車流程圖。

PC 端：

用來記錄監控端所收到的訊號強度（RSSI 值），並且顯示在網頁上，如圖 4-1-8，以及接收監控端送出的求救訊息，判斷三台救難車與失火車的訊號強度（RSSI 值），何者為最近，將此訊息傳給監控端，如圖 4-1-9。

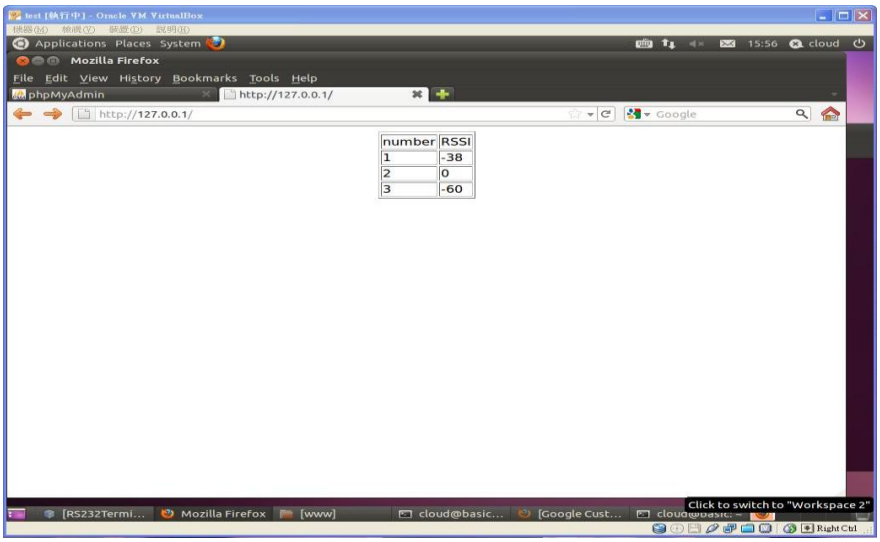


圖 4-1-8 PHP 畫面

圖 4-1-9 將救援訊息傳給監控端

4-2 訊號強度擷取

我們是採用 ZigBee 無線通訊模組來做傳輸，透過 BASIC Stamp 撰寫程式，將 ZigBee 與 ZigBee 之間的傳送中擷取訊號強度（RSSI 值），如圖 4-2-1、4-2-2。

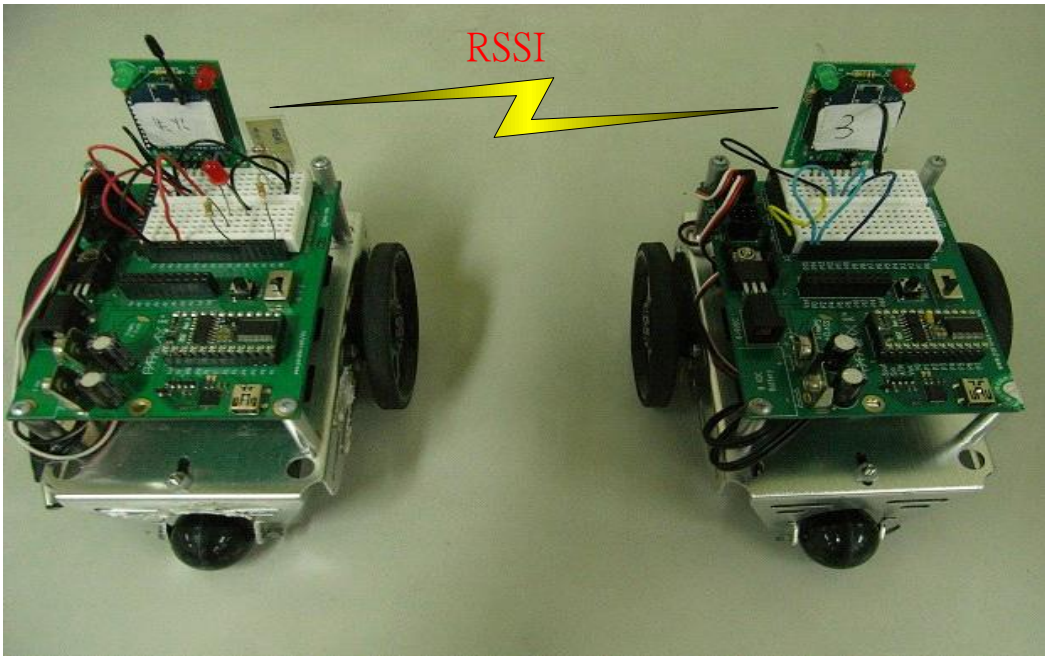


圖 4-2-1 ZigBee 之間傳送訊息

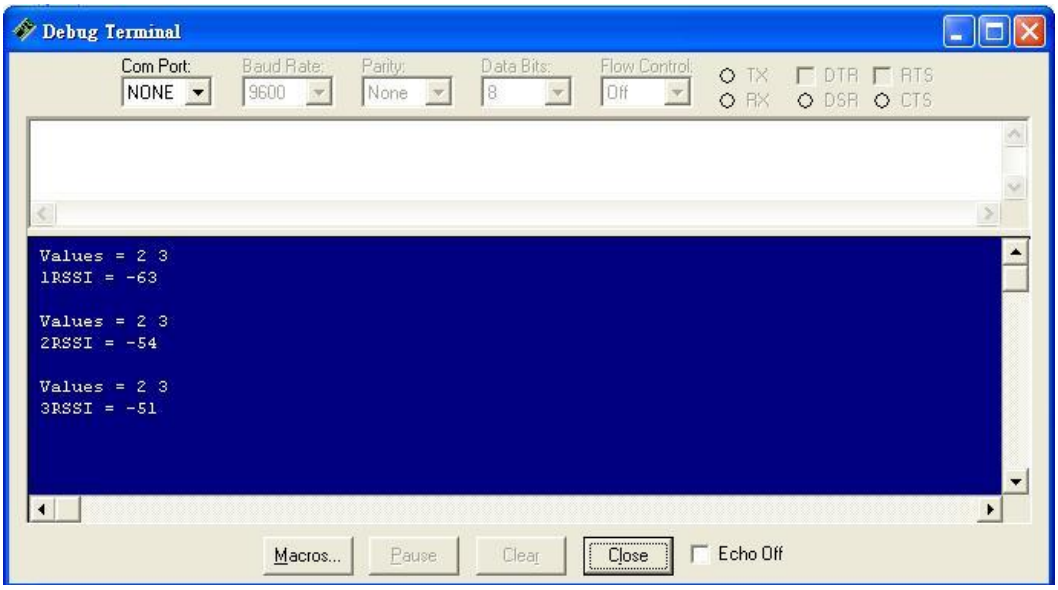


圖4-2-2 失火車收到三台救難車RSSI，用BAISC Stamp軟體顯示畫面。

4-3 監控端建置說明

監控端如圖4-3-1，我們透過ZigBee模組來做通訊，在監控端上裝有LED，用來顯示是否有收到求救訊號，監控端主要透過ZigBee接收失火車所傳來的RSSI值，並傳到PC端存進資料庫，如圖4-3-2，以及接收求救訊息及發送救援訊號。

圖 4-3-1 監控端與 PC 端做連結

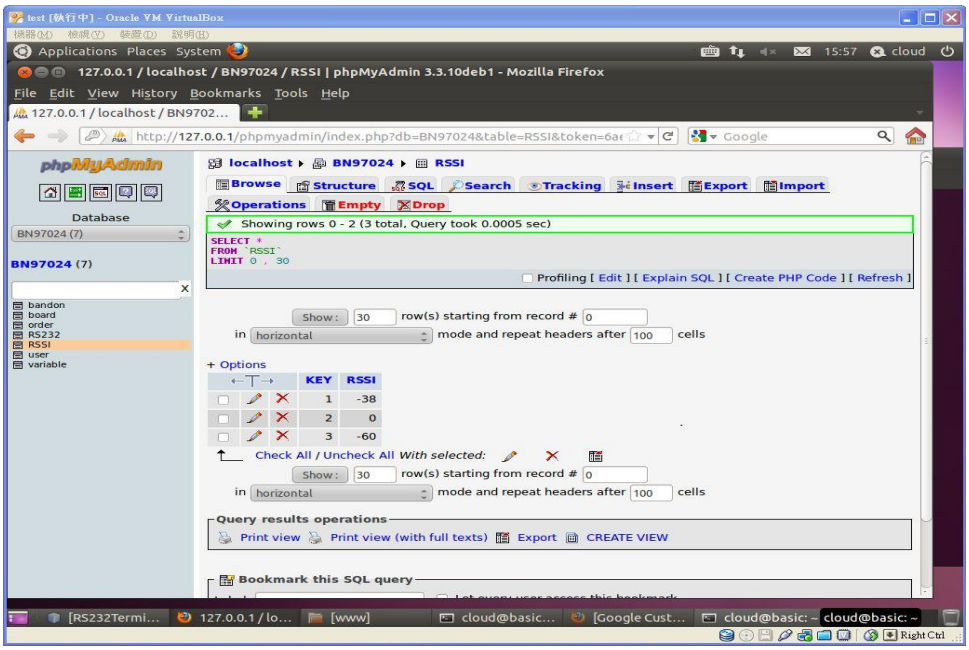


圖4-3-2 資料庫畫面

4-4 電腦控制端(PC端)

電腦控制端:透過RS232(JAVA)接收監控端傳送的資訊(RSSI)，然後透過socket送資訊到遠端，遠端利用JDBC更新資料庫，把資料庫建置在ubuntu上，ubuntu裡安裝了apache(web server)及mysql(資料庫)，使用php(網頁)來呈現，在需要的時候，計算分別收到的資訊(RSSI)，將處理完的資訊再沿原通道送回機器車，達到雙向通訊。

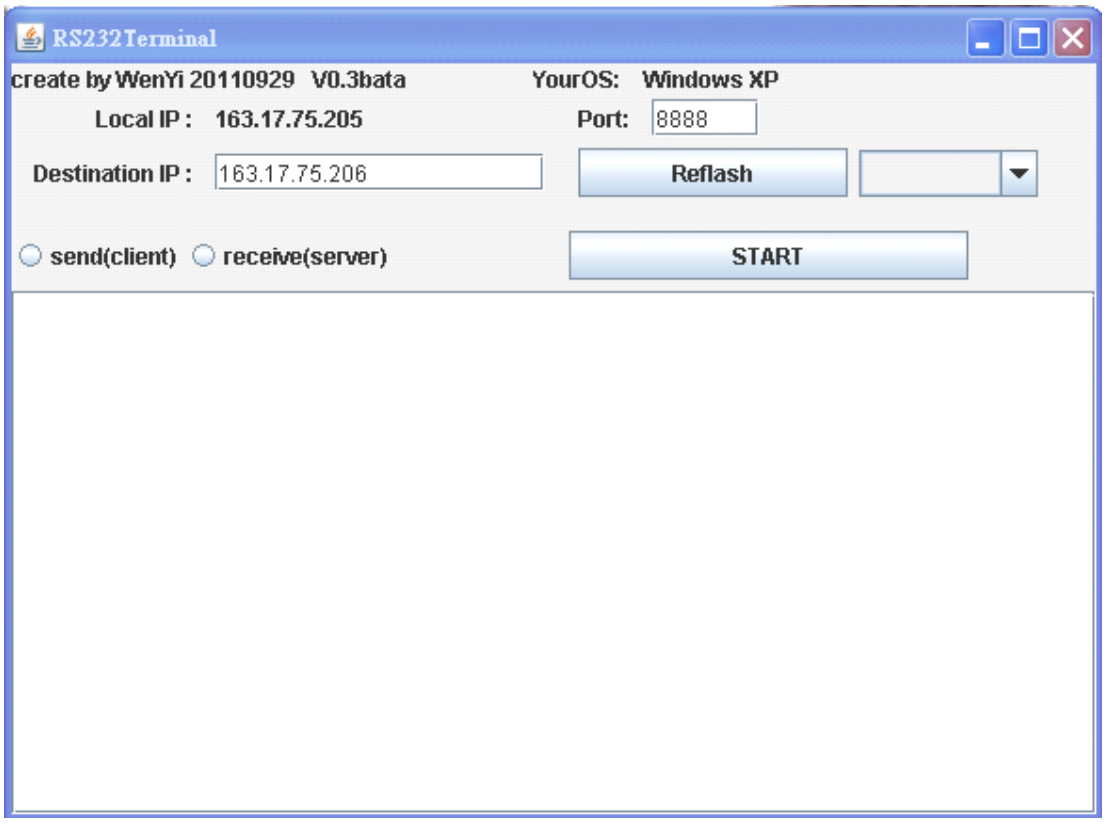


圖4-4-1 RS232 Terminal

我們將socket的server端與client端和RS232、JDBC寫成一個UI，Client端的運作方式為接收RS232的資訊，將資訊篩選後送到 Server端，Server端接收後依照資訊將資料庫裡的資料更新。

Server端為Server模式，負責接收Client端送來的資訊，並依資訊的不同更，來更新資料庫，需要的時候，將資料庫的訊息回傳給Client端。Client端為Client模式，負責接收機器車的訊息，並將訊息送出到Server端，若接收到機器車特定的訊息，則會去收Server端的訊息並回傳給機器車。

操作與程式碼分為兩種Client模式和Server模式:

一、Client模式:

首先要確認連接到電腦的usb號碼，點選1的Reflash會出現目前PC上所有已連接的usb comport，選擇連接BB自走車的comport，如下圖:

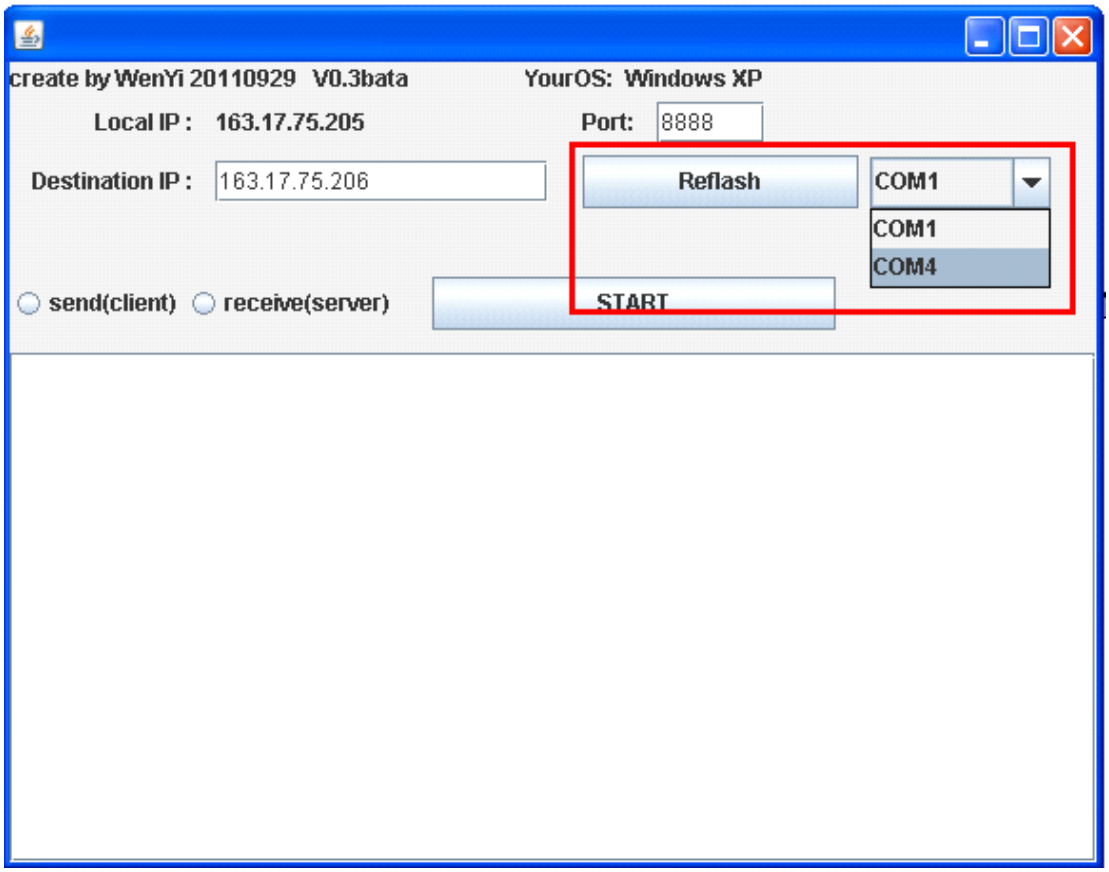


圖4-4-2 選擇連接BB自走車的comport

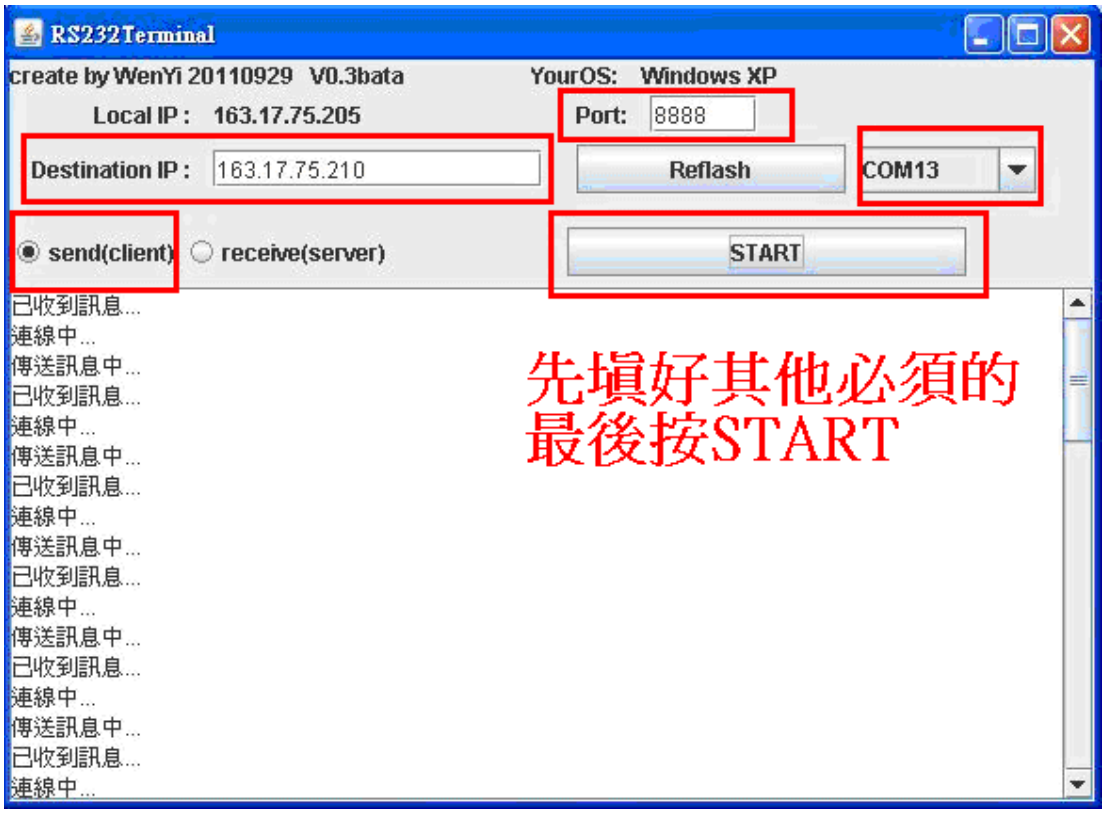


圖4-4-3 連入Client需填寫的區塊

然後要知道server端的ip位址，若不會看可以直接看左上角的local IP它會顯示目前本機IP(遠端電腦執行時)位址，接著設定port number，Server與Client 需要同一埠號才能正常通訊，選擇Server模式或是Client模式，這裡為Client模式，最後按START開始執行。

二、Server模式:

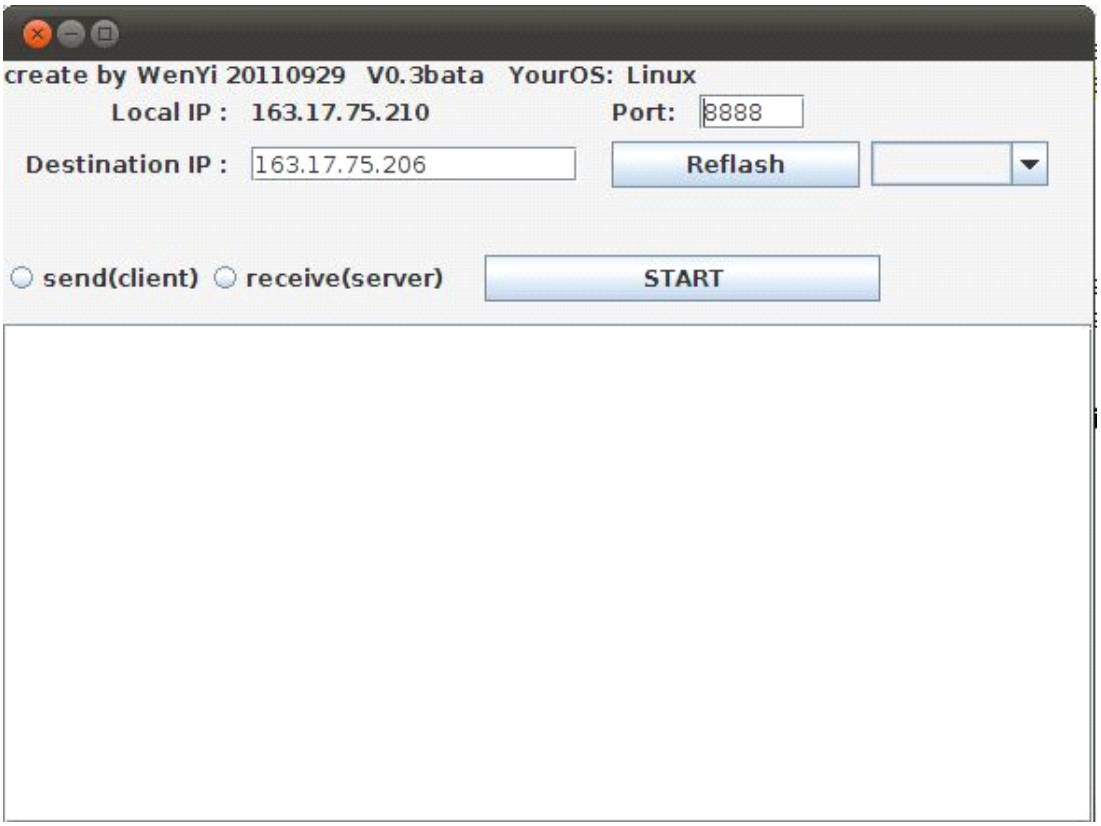


圖4-4-4 直接選取server再按START

若client送資料來，則會顯示在上面，並更新資料庫裡的內容，若需要找出最近的救難車，會取出資料庫內的最近車號，並送回client端。

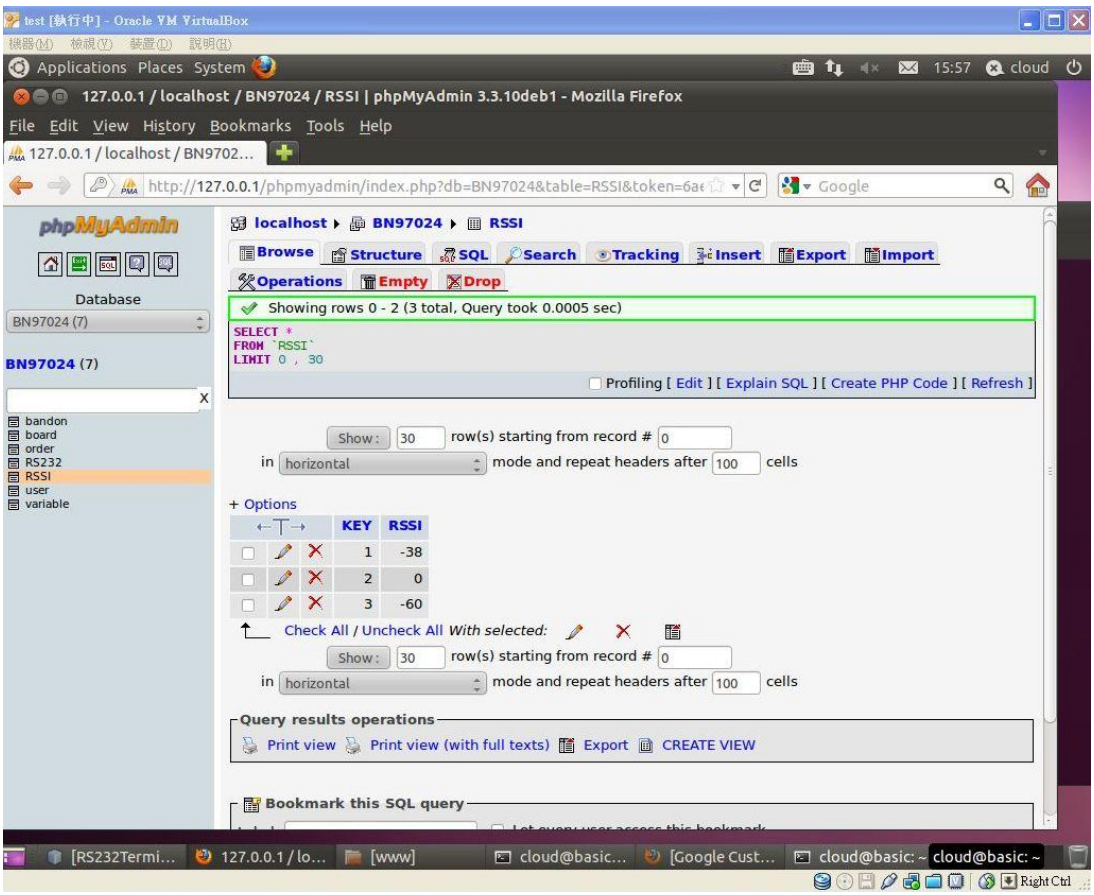


圖4-4-5 管理資料庫的介面

利用Apache建置Web Server，我們再撰寫php網頁，將mysql資料庫的資料，以其他形式呈現在網頁上，這樣就能透過網際網路隨時監看訊息。

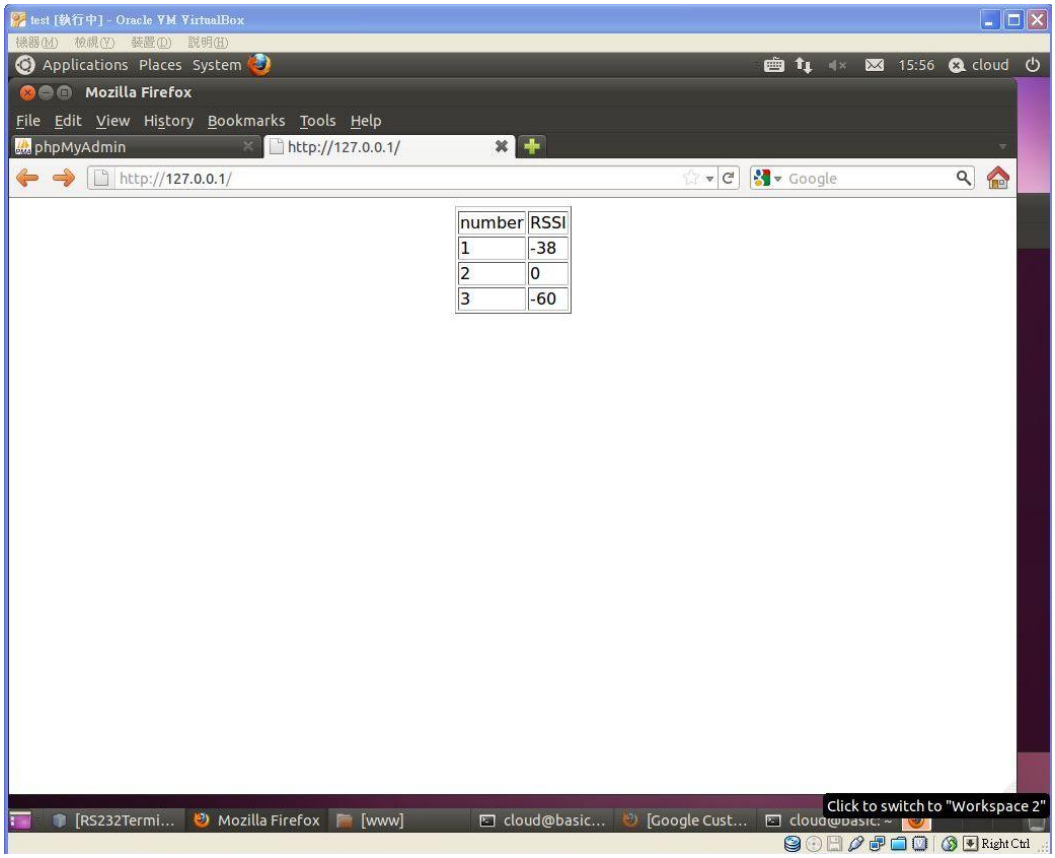


圖4-4-6 網頁上的訊息、在本機觀看的情形

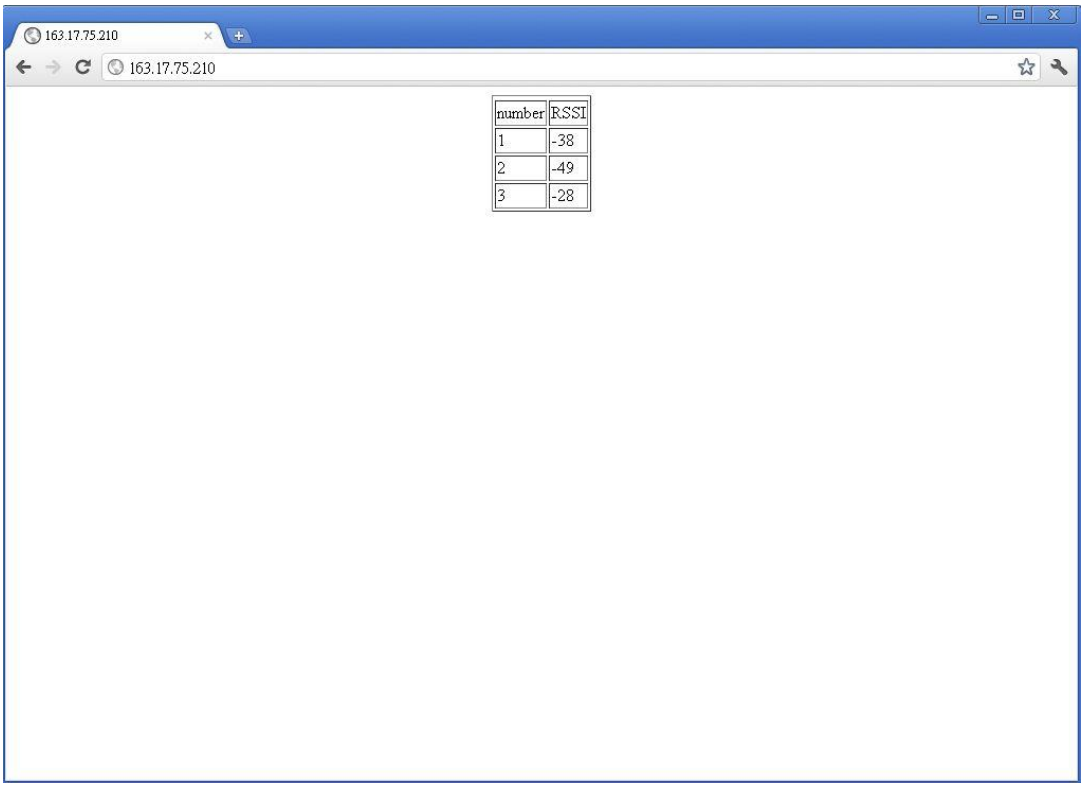


圖4-4-7 網頁上的訊息、透過網際網路觀看的情形

程式碼部分:

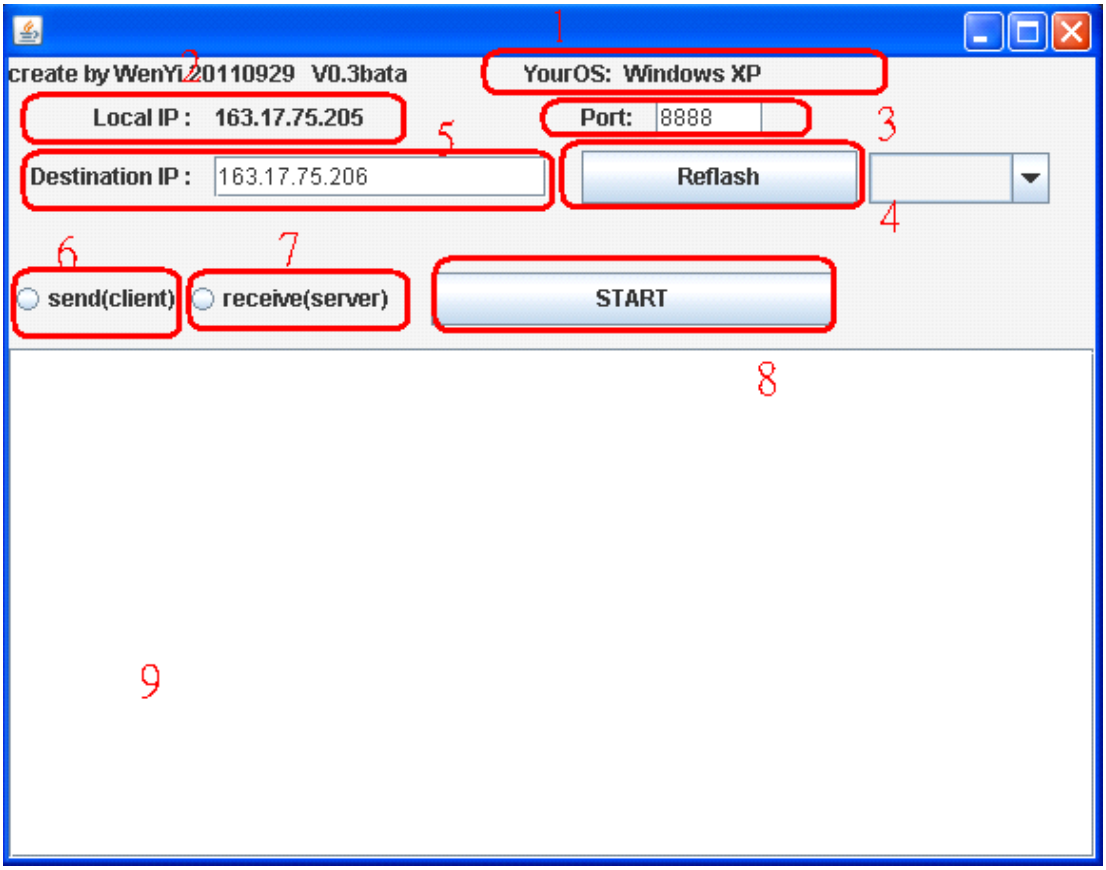


圖4-4-8 以程式流程的方式，將程式碼分成幾個部份來說明。

第(1)的部分:

利用JAVA提供的方法，System.getProperty(“os.name”)，將會回傳一個字串，為系統的名稱，再將該Label改成系統名稱，因為需要抓取本機IP，而Windows和unix的方法會不一樣，藉此來辨明需要用哪些方法去抓取本機IP(詳見(2))。

```
49          String os = System.getProperty("os.name");
50          jLabel7.setText(os);
```

圖4-4-9 第(1)的部分的回傳一個字串

第(2)的部分:

抓取本機IP，配合(1)獲得的字串，利用indexOf來比對是否為windows的系統，將會回傳一個值，代表指定字串第一次出現的值，因為字串定為”Windows”，所以理想值為0，若不是則為unix系統。

隨著作業系統的不同,有不一樣的抓取系統IP的方法，所以利用if…else來判別indexOf方法的回傳的值是否為0，表示為Windows系統，小於0(else)則為unix系統，Windows系統抓取本機IP的方法,JAVA的lib有提供，在有InetAddress屬性的物件使用getLocalHost()方法，此方法將會回傳電腦名稱/ip，再使用getHostAddress，回傳的是String型別的IP，而unix系統若使用上述方法會得到127.0.0.1，這不是我們想要的，所以就另外換方法，Runtime.getRuntime().exec(“ ”)為JAVA lib提供的方法，表示在目前作業系統的環境使用文字命令模式，” ”內為輸入的指令，由於在這裡因為是unix系統，所以使用了”ifconfig”，再將此輸出輸入到JAVA裡接收，使用BufferReader來接收，將接收的資訊使用while迴圈將資訊送出來，然後使用indexOf比對出第一個”inet addr”位置，因為通常目前正在使用的網路會是第一個網路連線，再使用substring方法，抓取出IP字串，將它設置到Label上。

```
51          int a = os.indexOf("Windows");
52
53          if (a >= 0) {
54              try {
55
56                  InetAddress inet = InetAddress.getLocalHost();
57                  jLabel5.setText(inet.getHostAddress());
58              } catch (IOException ex) {
59                  Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
60              }
61          } else {
62              try {
63                  String mac = null;
64                  BufferedReader bufferedReader = null;
65                  Process process = null;
66                  process = Runtime.getRuntime().exec("ifconfig");
67                  bufferedReader = new BufferedReader(new InputStreamReader(process.getInputStream()));
68                  String line = null;
69                  int index = -1;
70                  while ((line = bufferedReader.readLine()) != null) {
71                      index = line.indexOf("inet addr:");
72                      if (index >= 0) {
73                          mac = line.substring(index + 10, 33);
74                          jLabel5.setText(mac);
75
76                          break;
77                      }
78                  }
79
80              } catch (IOException ex) {
81                  Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
82              }
83
84          }
```

圖4-4-10 第(2)的部分的程式碼

第(4)的部分:

CommPortIdentifier 為RXTX提供的屬性，Enumeration為物件的集合，因為CommPortIdentifier的屬性也是Enumeration，所以在使用的時候就不用再強制轉換型別，先將comboBOX(圖4-4-8的(4))裡的物件清空，使用getPortIdentifiers()可以知道目前電腦連接的實體port，使這些port成為一個集合，因為RS232的類型是SERIAL，所以使用if…else判斷只要SERIAL介面，用while()將集合裡的元素，一個一個放到放到comboBox裡。

```
507         Enumeration portList;
508         CommPortIdentifier portId;
509         try {
510             portid.removeAllItems();
511             portList = CommPortIdentifier.getPortIdentifiers();
512             while (portList.hasMoreElements()) {
513                 portId = (CommPortIdentifier) portList.nextElement();
514                 if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
515                     portid.addItem(portId.getName());
516                 }
517             }
518 
519         } catch (Exception ex) {
520                  Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
521         }
```

圖4-4-11 第(4)的部分的程式碼

第(3)、(5)、(6)、(7)、(8)的部份:

我們除了使用JAVA預設的lib，還用了RXTX的lib，它可以幫助我們使用JAVA去和RS-232連結溝通，RXTX是open souce，讓Java的工具包（JDK）與serial和parallel溝通，因為使用了Socket，所以會依照使用者選了Server(7)或者Client(6)模式來進行。

使用 client 模式:

在 client 模式中，必須先填入 port，數字從 1-65536，一般是填 1024 以上，還要避開其他應用程式用到的號碼，要與 server 端相符才可以運作，然後是目的 IP，IP 需要實體的 IP，這樣才能將資訊送達 server 端，若不是實體 IP，則需其他技術(如 port forwarding)來達成，最後按下 START 鍵，依照所選模式開始運作。

若選擇的為 client 模式:

Client 用到了 RXTX(RS-232)、socket、RXTX 需要 COM port，socket 需要埠號(port)和 IP 位址，就跟上述的一樣(client 需要填入的欄位)，有了上述的部份，就可以使用 getText()等方法取得輸入的資訊再使用 toString()方法轉成字串，就能夠使用在其他方法。

```
private void STARTActionPerformed(java.awt.event.ActionEvent evt) {

    if (send.isSelected()) {

        try {
            new RS232Reader().connect(portid.getSelectedItem().toString());
        } catch (Exception ex) {
            Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else if (receive.isSelected()) {
```

圖 4-4-12 選擇的 client 模式的程式碼

在開啟該 COM port 之前要先看有沒有其他程序在使用，開啟的時候還要設定速率等相關數據，要兩邊速率符合(PC-BOE 板)，這樣才能讀到正確的資訊，讀取資料的時候，用 getInputStream()取得資訊再用 CommPortReceiver()來接收，由於 CommPortReceiver()有繼承執行緒，所以能夠不斷的接收資訊。

```
202     class RS232Reader {
203
204         public void connect(String portName) throws Exception {
205             CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
206             if (portIdentifier.isCurrentlyOwned()) {
207                 System.out.println("Port in use!");
208             } else {
209
210                 SerialPort serialPort = (SerialPort) portIdentifier.open(this.getClass().getName(), 2000);
211
212
213                 serialPort.setSerialPortParams(
214                         9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
215
216
217                 new CommPortReceiver(serialPort.getInputStream()).start();
218                 CommPortSender.setWriterStream(serialPort.getOutputStream());
219
220             }
221         }
222     }
```

圖 4-4-13 選擇的為 send(client)端的程式碼

將收到的資訊一個 byte 一個 byte 讀取直到沒有東西可以讀取並交由 protocolImpl 的 onReceive 處理。

```
171     class CommPortReceiver extends Thread {
172
173         InputStream in;
174         Protocol protocol = new ProtocolImpl();
175
176         public CommPortReceiver(InputStream in) {
177             this.in = in;
178         }
179
180         public void run() {
181             try {
182                 int b;
183                 while (true) {
184
185                     // if stream is not bound in.read() method returns -1
186                     while ((b = in.read()) != -1) {
187                         protocol.onReceive((byte) b);
188                     }
189                     protocol.onStreamClosed();
190
191                     // wait 10ms when stream is broken and check again
192                     sleep(10);
193                 }
194             } catch (IOException e) {
195                 e.printStackTrace();
196             } catch (InterruptedException e) {
197                 e.printStackTrace();
198             }
199         }
200     }
```

圖 4-4-14 onReceive 的處理程式碼

protocolImpl()是 protocol 介面的實作方法加上其他自定義方法，protocolImpl()的 onReceive()持續收到資訊，若內容不是'\r'就一直將內容存在陣列裡,若是則代表換行,在 onMessage()會呼叫 getMessage()將陣列裡的資訊轉成 String 並將資訊用 socket 傳出。

```
90      class ProtocolImpl implements Protocol {
91
92          Socket s;
93          byte[] buffer = new byte[1024];
94          int tail = 0;
95
96          @Override
97          public void onReceive(byte b) {
98              // simple protocol: each message ends with new line
99              if (b == '\r') {
100                 onMessage();
101             } else {
102                 buffer[tail] = b;
103                 tail++;
104             }
105         }
106
107         public String getMessage(byte[] buffer, int len) {
108             return new String(buffer, 0, tail);
109         }
110
111         private void onMessage() {
112
113             if (tail != 0) {
114                 // constructing message
115                 String message = getMessage(buffer, tail);
116
117                 //System.out.println(" RECEIVED MESSAGE:" + message);
118                 Socket(message);
119
120                 tail = 0;
121             }
122         }
```

圖 4-4-15 為轉換 String，將資訊用 socket 傳出

Socket()收到資訊後，將輸入的 port 及 IP 位址轉成需要的型別後建立連線，以 OutputStream()將資料存放，再呼叫子函式 write 將資料送出到 Server 端。

```
136         private void Socket(String msg) {
137             try {
138                 String Destinnation, cport;
139                 Destinnation = DIP.getText().toString();
140                 cport = port.getText().toString();
141                 s = new Socket(Destinnation, Integer.parseInt(cport));
142                 message.append("連線中..." + "\n");
143                 BufferedWriter bw = new BufferedWriter(
144                         new OutputStreamWriter(s.getOutputStream()));
145                 bw.write(msg + "\n");
146                 message.append("傳送訊息中..." + "\n");
147                 System.out.printf("傳送訊息中..." + "\n");
148                 bw.flush();
149             } catch (UnknownHostException ex) {
150                 Logger.getLogger(ProtocolImpl.class.getName()).log(Level.SEVERE, null, ex);
151                 System.out.printf("連線失敗..." + ex.getMessage() + "\n");
152             } catch (IOException ex) {
153                 Logger.getLogger(ProtocolImpl.class.getName()).log(Level.SEVERE, null, ex);
154                 System.out.printf("連線失敗..." + ex.getMessage() + "\n");
155             }
156         }
157     }
```

圖 4-4-16 連線訊息的成功與失敗的程式碼

Server:

若選擇的為 receive(server)，會將 port 抓取成為 int 型別，並建立 ServerSocket 以 thread 去持續 listen 該 port 是否有資訊傳入， 再用 thread 去不斷接收資訊，將資訊交給 processClientMessage( )處理。

```
422         } else if (receive.isSelected()) {
423             int sport = Integer.parseInt(port.getText().toString());
424             try {
425                 ss = new ServerSocket(sport);  // ServerSocket --> bind --> listen
426                 new Thread() {
427                     public void run() {
428                         while (true) {
429                             try {
430                                 final Socket s = ss.accept();
431                                 new Thread() {
432                                     public void run() {
433                                         processClientMessage(s);
434                                     }
```

圖 4-4-17 連線訊息的成功與失敗的程式碼

processClientMessage()是承接上一張圖接收資訊的 thread 裡的程式，用迴圈判斷連線是否中斷來執行處理資料的工作，把收到的資訊用 getInputStream()抓取，再用 BufferReader 儲存，宣告 String 變數(msg)用 readLine()把儲存的資料一行行取出，並用迴圈判斷資料是否讀取完畢，若是則跳出，然後將資料加到 textArea 圖 4-4-8(9)，並將此訊息傳給 jdbcmysql()的 updatetableRSSI()更新資料庫。

```
426                         private void processClientMessage(Socket s) {
427                             while (!s.isClosed()) {
428                                 try {
429                                     BufferedReader br = new BufferedReader(
430                                             new InputStreamReader(s.getInputStream()));
431                                     String msg = br.readLine();
432                                     if (msg == null) {
433                                         break;
434                                     }
435                                     message.append(msg + "\n");
436                                     //System.out.printf(msg);
437                                     jdbcmysql test = new jdbcmysql();
438                                     test.updatetableRSSI(msg);
439 
440                                 } catch (IOException ex) {
441                                     Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
442                                     try {
443                                         s.close();
444                                     } catch (IOException ex1) {
445                                         Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex1);
446                                     }
447                                 }
448                             }
449                         }
450                     }.start();
451                 } catch (IOException ex) {
452                     Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
453                 }
454             }
455         }
456     }.start();
457 } catch (IOException ex) {
458     Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE, null, ex);
459 }
```

圖 4-4-18 處理 socket 的 thread

更新資料庫:

用的是 JDBC 包,跟 RXTX 不同的是它為 sun 釋出的,能夠在 JAVA 使用 mysql 或者其他資料庫語言,對資料庫送出 SQL 語句,作新增、刪除、修改、查詢的動作。

因為考慮不讓資料過於龐大的緣故,用的是更新的語句,先宣告 String 變數,內容為各資訊所需使用的 SQL 語句,再載入對應的資料庫驅動,取得對資料庫的連線,利用這個連線(物件),對資料庫送出 SQL 語句。

```
public class jdbcmysql {
  private Connection con = null; //Database objects
  //連接object
  private Statement stat = null;
  //執行,傳入之sql為完整字串
  private ResultSet rs = null;
  //結果集
  private PreparedStatement pst = null;
  //執行,傳入之sql為預儲之字申,需要傳入變數之位置
  //先利用?來做標示
 private String updateSQLX="update RS232 set `X`= ? where `EQ`=100"   ;
 private String updateSQLY="update RS232 set `Y`= ? where `EQ`=100"   ;

 private String updateSQLRSSI="update RSSI set `RSSI`= ? where `KEY`=1"   ;
  private String updateSQLRSSI2="update RSSI set `RSSI`= ? where `KEY`=2"   ;
  private String updateSQLRSSI3="update RSSI set `RSSI`= ? where `KEY`=3"   ;


  private String selectSQL = "select * from variable ";

  public jdbcmysql()
  {
    try {
      Class.forName("com.mysql.jdbc.Driver");
      //註冊driver
      con = DriverManager.getConnection(
      "jdbc:mysql://localhost/BN97024?useUnicode=true&characterEncoding=utf8",
      "BN97024","5539378a");
      //取得connection
```

圖 4-4-19 在 JAVA 使用 JDBC 需要取得的連線與定義好的 SQL 語句

updatetableRSSI()會依據收到的值，先將收到的字串看第一個值為何，依此來判斷該使用哪一個 SQL 語句，再將此 SQL 語句和所要更新的值，用連線(物件)送出，來更新資料庫。

```
public void updatetableRSSI(String RSSI) {
    try {
        String F2 = RSSI.substring(0, 1);
        int F3=0;
        if (F2.equals("1")) {
        F3 = Integer.parseInt(RSSI.substring(8,11));
        pst = con.prepareStatement(updateSQLRSSI);
        }
        else if (F2.equals("2")) {
        F3 = Integer.parseInt(RSSI.substring(8,11));
        pst = con.prepareStatement(updateSQLRSSI2);
        }
        else if (F2.equals("3")) {
        F3 = Integer.parseInt(RSSI.substring(8,11));
        pst = con.prepareStatement(updateSQLRSSI3);
        }

        System.out.printf("SQL" + RSSI);
        pst.setInt(1, F3);
        pst.executeUpdate();
        pst.clearParameters();
    } catch (SQLException e) {
        System.out.println("updateDB Exception :" + e.toString());
    } finally {
        Close();
    }
}
```

圖 4-4-20 更新資料庫的程式

使用SQL語句將RSSI資料表中存放的RSSI值取出放入陣列裡，利用JAVA內建方法Math.max()找出最小值(因為負值，但是實際上我們要的為加上絕對值後的最大值)，再與陣列逐個元素比對找出其對應的車號，將該車號對資料表更新。

```
public void  SelectRSSI() {
   int[] min = new int[3];
   String mini = "null";
   try {
       stat = con.createStatement();
       rs = stat.executeQuery(selectRSSI);
        int a=0;
       while (rs.next()) {
           int aa;
          aa= rs.getInt("RSSI");
          min[a]=aa;
          a++;}
       int Max;
       Max=min[0];
       for(int i=1;i<a;i++){
     Max=Math.max(Max, min[i]);     }
       System.out.printf(Max+"\n");
       for(int i=0;i<a;i++){
       if(Max==min[i]){
        if(i+1==1){
       mini="A";
        }else if(i+1==2){
        mini="B";
        }else if (i+1==3){
        mini="C";
        } }}
      pst = con.prepareStatement(updatemin);
       pst.setNString(1, mini);
       pst.executeUpdate();
      pst.clearParameters();
  } catch (SQLException e) {
      System.out.println("DropDB Exception :" + e.toString());
  }
```

圖4-4-21 找出資料表中所有RSSI值，比對出最小值並回傳車號

當失火車發出求救訊息時，監控端會收到並送特定訊息給Client端，Client一樣會將訊息送給Server端，當Server端收到此特定訊息會去資料庫找出(上一步計算出的資料)最近的車號並送回Client端

```
} else if (msg.equals("min")) {

    min = test.SelectTable();
    message.append(min + "\n");
    test.Close();
    bws.write(min + "\n");
    bws.flush();
```

圖4-4-22 透過JDBC找出資料庫中的資訊並送回Client端。

```
public String SelectTable() {
    String min = null;
    try {
        stat = con.createStatement();
        rs = stat.executeQuery(selectmin);

        while (rs.next()) {
           // System.out.println(rs.getString("min"));
           min= rs.getString("min");
        }
    } catch (SQLException e) {
        System.out.println("DropDB Exception :" + e.toString());
    }
 return min;
}
```

圖4-4-23 找出最近車號的資料表所呼叫的方法。

Client端這裡使用Thread來接收Server端回傳的車號(字串)，使用與Server端連線時同一個Socket物件，來取得BufferReader物件，以readLine()方法逐行讀取，最後將車號用CommPortSender()傳出到機器車。

```
new Thread() {

    @Override
    public void run() {

        String msg2;
        try {

            brc = new BufferedReader(
                    new InputStreamReader(s.getInputStream()));


            msg2 = brc.readLine();
            System.out.printf(msg2 + "\n");
            message.append(msg2 + "\n");

            CommPortSender.send(getMessage(msg2));
        } catch (IOException ex) {
            Logger.getLogger(TerminalUI.class.getName()).log(Level.SEVERE,
        }

    }
}.start();
```

圖4-4-24 用Thread接收Server回傳的車號，呼叫CommPortSender()。

```
class CommPortSender {

    static OutputStream out;

    public static void setWriterStream(OutputStream out) {
        CommPortSender.out = out;
    }

    public static void send(byte[] bytes) {
        try {


            // sending through serial port is simply writing into OutputStream
            out.write(bytes);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
public byte[] getMessage(String message) {
    return (message + "\n").getBytes();
}
```

圖4-4-25 將車號利用getMessage()轉成Byte型別後，以OutputStream()物件的子方法write()送出到機器車。

4-5 通知救難車

監控端收到 PC 端給的訊息（距離最近的救難車），假設 1 號救難車為最近，監控端會透過 ZigBee 對 1 號救難車發出救援訊息（前往失火車救難），1 號救難車收到前往失火車的訊息後，會亮起車上的 LED，表示收到救援訊息。

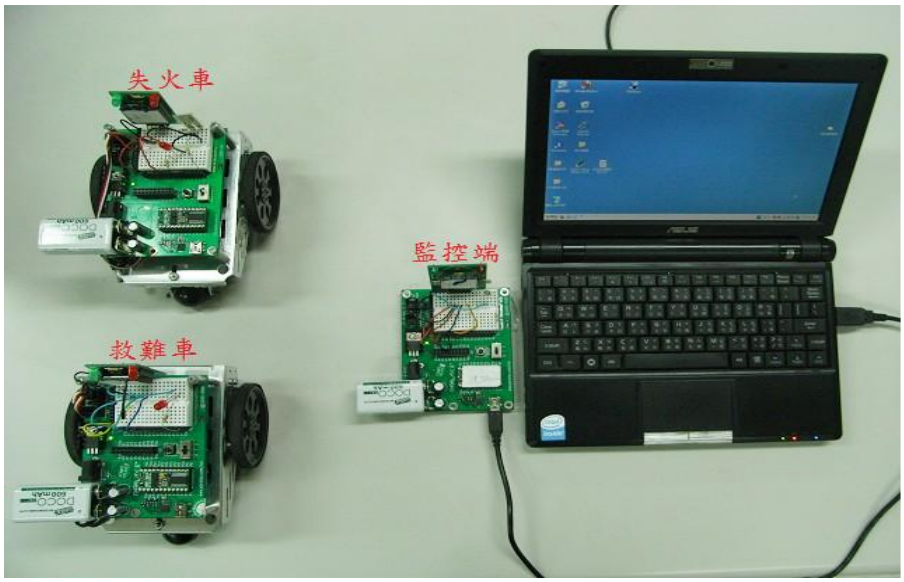


圖 4-5-1 沒亮 LED

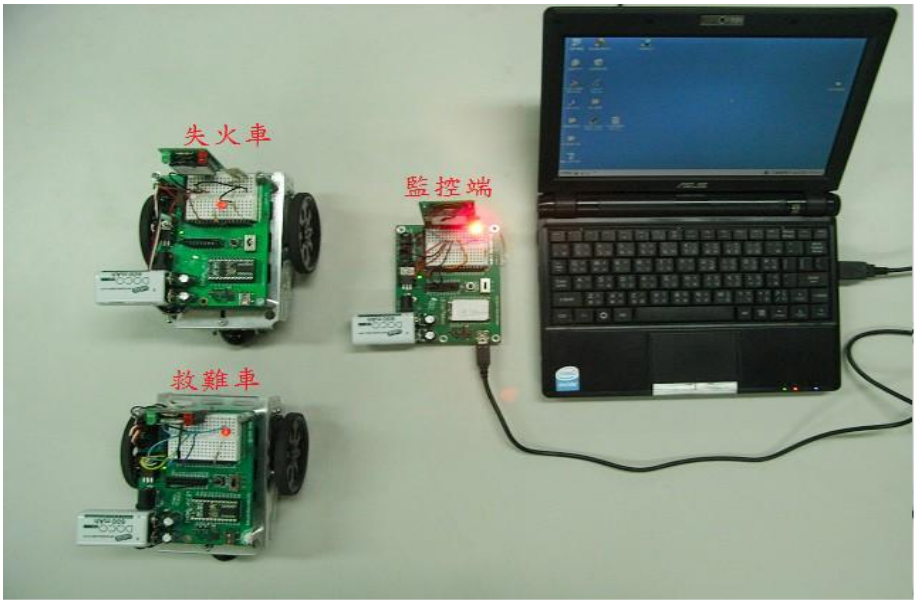


圖 4-5-2 亮 LED

4-6 實作總覽

首先我們假設將失火車放在 A 點，而救難車放在 A 點以外的其他地方，並將監控端與 PC 端做連結，此刻失火車會發出訊息給三台救難車，要求三台救難車回傳一個訊息，表示有收到訊號，而救難車收到訊息後，會回應失火車所需要的訊息，則失火車收到救難車回傳的訊息後，會擷取此訊號強度（RSSI 值），並將訊息傳送給監控端，則監控端會將訊息傳給 PC 端，PC 端透過 RS232（JAVA）接收監控端傳來的訊息（RSSI 值），再透過 socket 送訊號到遠端，遠端利用 JDBC 更新資料庫，資料庫建置在 ubuntu 上，ubuntu 裡安裝了 apache(web server)及 mysql(資料庫)，使用 php(網頁)來呈現，為了更新訊息，以上的動作會一直做循環。

當失火車上的意外開關（震動片）觸碰了，則失火車會發出求救訊號，監控端收到失火車的求救訊號後，會告知 PC 端，當 RSSI 值越接近”0”，表示距離失火車越近，再將比較結果（假設為 1 號救難車最接近）傳給監控端，則監控端接收到此訊息，會傳救援訊息給 1 號車，當 1 號車收到救援訊息後，會亮起 LED 燈，表示已收到救援訊息。

3車							
			A				
					1車		
	2車						

圖 4-6-1 場地位置

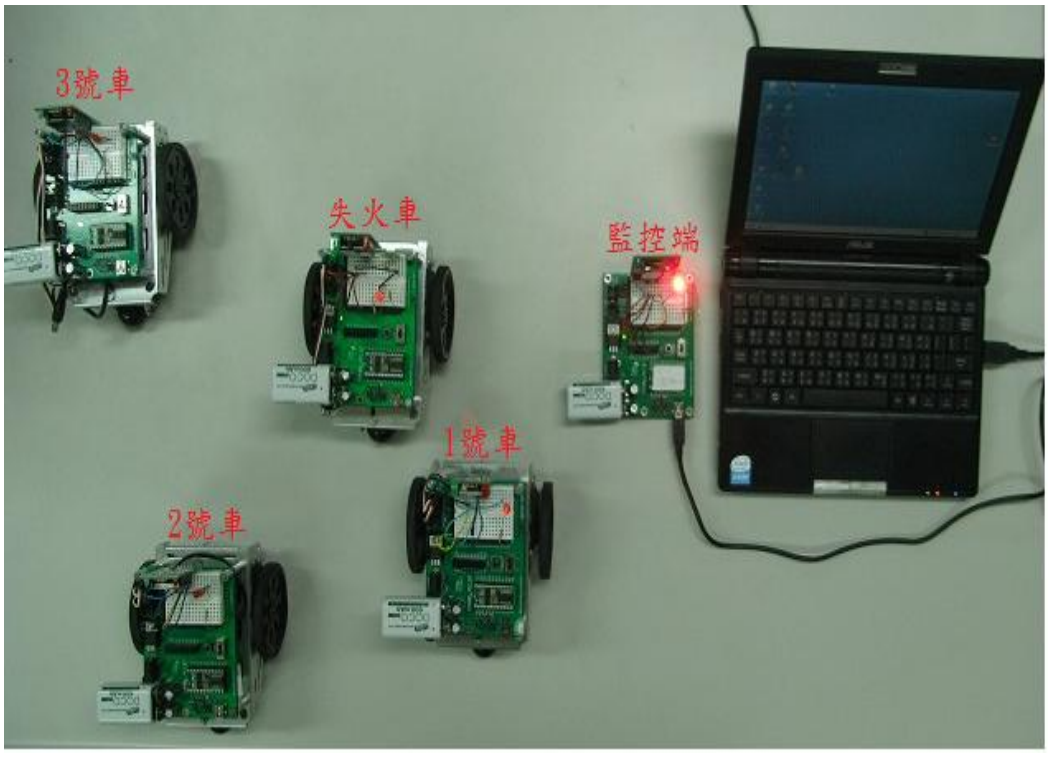


圖 4-6-2 最後結果

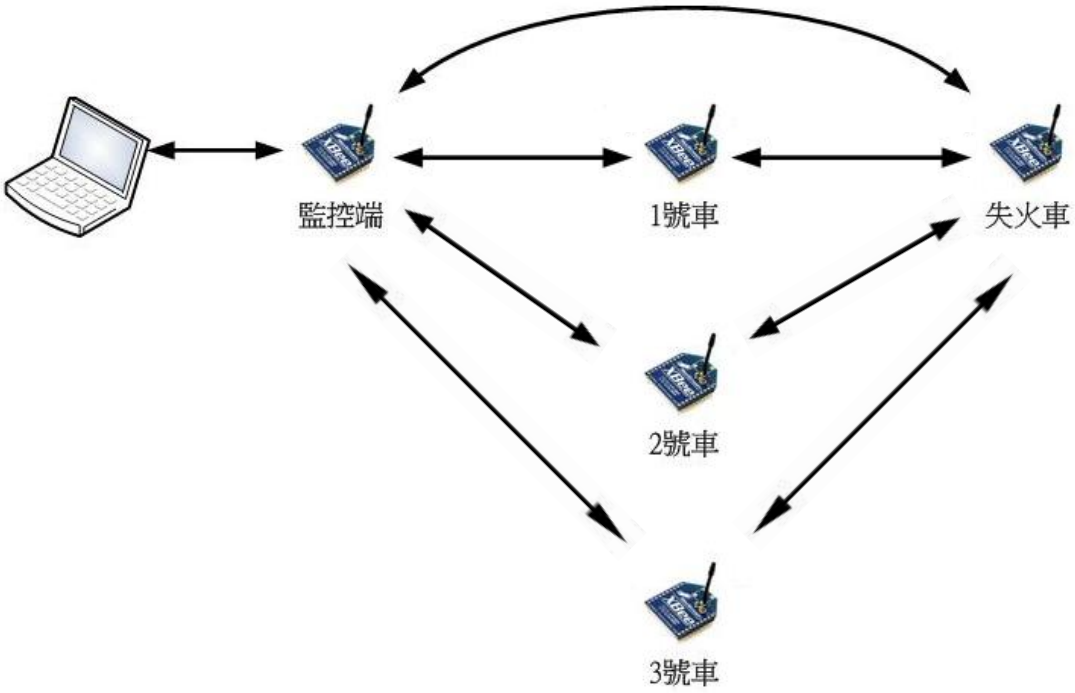


圖 4-6-3 ZigBee 通訊模組通訊狀況圖

第五章 結論與未來研究方向

結論:

在地方災害發生的時候，通常有很多手段來得知災害是否發生，後續的通報救災也是依照當地最近的救難局來處理，但是若災害是載具上發生的時候，就需要持續得知載具所在的地點，這樣災害發生時才能夠最快的來救難，這時就必須使用無線通訊來獲知載具地點，並能夠將災難訊息發送到最近的救難局，將這些作到自動化且精準得到災害是否發生以及最近的救難局在哪，就能夠更快來援救，並且若能夠透過感測器偵測災難類型及強度就能夠作到最佳化救災資源分配，在多處需要救難且救災資源不充足時能夠立即判斷。

未來研究方向:

一、可增加三軸定位系統、GPS、Google　Map，讓救難車可以很精準地到達失火地點，並且會策劃到達失火點的最佳路徑走向。

二、可以加裝攝影機，已利用遠端看到現在的情境。

三、未來不一定要用飆機器車公司的 Boe-Bot Robot Kit(通稱 BB)電路板，可以使用 P8X32A-Q44 Propeller 微控制器晶片的多功能八核心板，來增加更多的作用。

四、未來會改以更穩定的網路傳輸設備，增加他傳送訊息的穩定度以確保在發生災害時不受其他環境因素，而受到干擾。

第六章 參考文獻

[1] 魏忠必，柯中益，涂柏村,“ZigBee技術應用於路燈監控系統之研究” in Proceedings of the 5th Intelligent Living Technology (ILT2010), pp. 210-217, Taichung, Taiwan, June 2010.

[2] 周哲宇，吳世琳, "基於大尺度路網動態路況資訊的車輛路徑規劃演算法,”in Proceedings of theTaiwan Academic Network Conference 2010 (TANet 2010), Tainan, Taiwan, Oct. 2010, pp. 1-6.

[3] 王能中，袁華建，黃紹偉，陳子龍,”基於無線感測網路之Google Map資料庫整合系統,” in Proceedings of the 15th Mobile Computing Workshop (MC2010), pp. 1-10, Taichung, Taiwan, May 2010.

[4] 李鴻成，在車載隨意網路以IP傳遞與暫存實現快速換手機制，國立高雄應用科技大學資訊管理研究所碩士論文, 2010年。

[5] Chin-Ling Chen, Jinn-Ke Jan, and Mei-Li Chong, “Using RFID To Design An Intelligent Parking Lot Management System,” in Proceedings of the 15th Mobile Computing Workshop (MC2010), pp. 1-5, Taichung, Taiwan, May 2010.

[6] 王煌城、李彥廣、張妙琦、陳彥良、程鴻文， 全民警察時代-以嵌入式技術建置贓車搜尋系統，國立宜蘭大學電子工程學系，碩士論文, 2010年