逢　甲　大　學

資訊工程學系專題報告

VB到VB.NET

以遠端家電控制系統開發為例

劉 原 孝 (四乙)

學　　生：張 駿 淵 (四乙)

李 佳 容 (四乙)

指導教授：竇 其 仁

中華民國九十二年十二月

目　　錄

圖 表 目 錄

摘要

本專題為「VB 到 VB.NET－以遠端家電控制系統開發為例」，重點在於藉由升級遠端家電控制系統升級實做，將 VB 專案升級到 VB.NET 平台，實際了解 VB 與 VB.NET 之間差異與探討，可分為下列四個部分。

第一章 導論

說明專題的動機，目的以及實做的分工情形

第二章 技術背景

介紹遠端家電控制系統中所使用的硬體技術，以及.NET 的背景知識概述。

第三章 系統架構與升級

說明遠端家電系統的軟、硬體架構，本系統的功能介紹，以及升級方法－利用升級精靈升級的評估。

第四章 系統升級實做

說明專題實際升級過程，以及在升級過程中所發現的困難與差異。

第五章 結論

對系統升級可再用程式碼以及升級結果總整理，提出遭遇的困難、解決方法、系統的未來展望以及專題心得。

第一章 導論

1.1 動機

隨著網際網路的蓬勃發展，近幾年由全球個人電腦軟體的霸主微軟，花了好幾年的開發投入大量的人力與資金，所研發出的.NET 技術，.NET 最終實現讓所有使用者可以在任何地方利用任何裝置，來獲得使用者的資料或應用服務的願景。在網際網路非常盛行的現在，以及微軟積極的推動.NET，.NET 非常有可能成為下一代應用程式發展的主要的平台，在過去完全沒有接觸過.NET 的我們，非常有興趣一探究竟，希望能對於這個完全嶄新的開發環境，能有所了解。

根據我們對.NET 的初步了解，發現.NET 與過去 Windows 平台式截然不同的，因此若.NET 技術要完全替代傳統的開發環境，必定要經過一番革命，在這過程中，原本架構在傳統的開發環境的系統，應該經過哪些改革，才能在新的.NET 環境下使用；同樣由微軟所開發的語言 VB，為了不在.NET 平臺中被淘汰，因此微軟在 VB.NET 中建立了升級精靈，來協助程式設計者升級，升級精靈號稱能將 VB 6.0 程式碼升級至百分之 95，是否真的如此功能強大？應用升級精靈會處理那些問題，及產生那些錯誤和警告訊息呢?

再者，近年來隨著資訊家電的風行，由學長所研發出來一套結合了資訊家電以及保全的系統---遠端家電控制與智慧型保全系統，在此

系統中，擁有非常良好的使用者介面，功能也非常的齊全，但隨著科技之變遷，系統必須再更新，才能使系統符合環境的改變，而繼續存留下來，這也是在軟體工程中一個重要的議題。

1.2 目的

倘若.NET 技術能廣大的推行，勢必有許多原本架構在傳統開發環境的系統需要更新，才能在新的.NET 環境下使用，因此，本著實驗的精神，具有前瞻性且創新的想法，決定將 VB 既有系統－遠端家電控制系統，轉移到.NET 的平台，藉由系統升級實作，了解系統升級到.NET 所需過程，探討同樣由微軟開發的 VB 語言與 VB.NET 之間的差異，VB.NET 升級精靈號稱能將系統升級 95%，實際上能夠升級的比例是多少；此外，系統升級也是一項良好學習與訓練，學習到系統再利用的概念，訓練我們軟體整合的能力，在未來職場上，系統升級是常見的，所以我們決定了我們的專題題目－VB 到 VB.NET 以遠端家電控制系統開發為例，開始了我們的專題實做。

海報製作內容

1.3 分工情形

工作項目	負責人員
面板	李佳容
Adam 控制程式	劉原孝
定時功能	張俊淵
電器控制功能	劉原孝
虛擬感應開關功能	張俊淵 劉原孝
錄影管理功能	劉原孝
登錄使用者管理	李佳容
專題書面編寫	李佳容 張俊淵 劉原孝
投影片製作	李佳容 張俊淵 劉原孝

第二章 技術背景

在本章將介紹.NET 的相關背景知識，以及在遠端家電控制系統所使用到的硬體裝置，包括 Adam 控制模組、網路攝影機以及固體繼電器。

2.1 .NET

2.1.1 .NET 的基本概念

以下的解釋來自於台灣微軟網頁

簡單地說，.NET 就是 Microsoft 為 XML Web 服務所提供的平台。XML Web 服務可讓多個應用程式透過 Internet 彼此通訊並共用資料，不論其作業系統或程式語言為何。

Microsoft® .NET 平台包含一系列功能齊備的產品，這些產品都採用 XML 及 Internet 業界標準所建置，可讓您全方位的開發、管理、使用並體驗 XML Web 服務的操作環境。XML Web 服務將會成為您目前所使用的 Microsoft 應用程式、工具和伺服器中的一部份；並且將會內建於新產品中，以因應您在商務上所有的需求。

更明確地說，Microsoft 目前建構中的 .NET 平台分成五大範圍，包括：工具、伺服器、XML Web 服務、用戶端和 .NET 操作環境。

簡單的講，是一種平台，也是一種規格，符合此規格的平台，不管是用神麼設備，神麼作業系統，通通都是.NET 平台，都可以跑.NET 程式，包括一般的視窗應用程式（windows form）.ADO 或者 ASP 程式（支援 XML 規格網頁程式），只要是微軟.NET 產品下，就代表可支援.NET 架構，可以寫網頁程式。

理論上，我們如果用 Studio.NET 開發程式，那就不需要考慮這個程式的平台是神麼，也不需要顧慮使用哪一種程式語言或程式庫，只要它支援.NET 架構，通常可以執行 Studio.NET 開發的應用程式，所以所寫的程式，可能是在個人電腦 PDA 汽車液晶螢幕手機也可以在 UNIX Windows Linux Mac 作業系統下跑，當然，前提是這些平台與作業系統通通要支援.NET。

2.1.2 .NET Framework

.NET Framework 又是神麼呢？以下的答案來自微軟 MSDN 網站：

.NET Frameworke 是所有.NET 平台的基礎結構。.NET Framework 包含了兩個主要的部分: Common Language Runtime 及類別庫(包含 ASP. NET、 Windows Form、 .ADO.NET)，相互結合以提供再不同系統中可以輕易整合的服務和解決方案。.NET Framework 提供完整管理.

保護和眾多功能的應用程式執行環境，簡化的開發與部署的方式，並且可以和各類語言無接縫的整合。

上面的意思是說.NET Framework 這個基礎結構提供對於軟體開發和 Windows 作業系統的新支援，不管用 Studio.NET 裡面的神麼工具開發，都需要.NET Framework 支援。因為.NET Framework 使用自己本身的程式碼包裹作業系統，所以 VB.NET 程式只須專責和.NET Framework 溝通即可，不需要去處理應付作業系統的細節 。

.NET Framework 包含兩大部分執行通用語言(Common Language Runtime)以及基底類別庫（Base Class Library）後者又包含 ASP.NET 、Windows Form、 .ADO.NET 三大部，主要是程式庫，分別用來開發 Windows 應用程式(表單程式) 及資料庫應用程式與支援 XML 協定的 ASP 程式。

2.1.2.1 Common Language Runtime

存在.NET Framework 基層的就是 Common Language Runtime (CLR)。CLR 是真正執行 VB.NET 應用程式的模組。各種語言包含 VB6 轉換到 VB.NET，造成之間差異的原因都是由於 CLR 需求的緣故。傳統的執行檔稱為未處理碼(unmanaged code)，只能在特定的系統上運作，而在.NET 架構下，當建立 VB.NET 應用程式時，其實是將程式

編碼譯成 CLR 的內部語言(稱之為 MSIL 或 IL)，稱為 Managed Code，並非機器碼，而是一種中介語言，它十分相似於 JAVA 中的 Bytecode。當在執行應用程式時，.NET 程式會呼叫 JLT(Just-In-Time)編譯器，這種編譯器會產生實際的原生碼以執行程式。透過這種機制，Microsoft 日後可以針對其他非 Windows 的作業系統建立 CLR，那麼 VB.NET 應用程式就可以執行於這些安裝 CLR 的非 Windows 作業系統之上。由於是轉介之後再執行，所以效率會比直接執行的機器碼來的差，因此不同機器、不同系統上都必須安裝.NET Framework，其中的程式庫直接提供符合該系統的程式碼，多少彌補了效率上的缺憾。

雖然使用 Managed codes 的第一次載入程式時的效率比較差，但它有以下的優點：

(1). 自動配置、清除記憶體。

(2). 自動執行垃圾收集（garbage collection），集中零碎的記憶體，組成效率較佳的叫大區塊。

(3). 程式較穩定、安全。

(4). 提供嚴謹的型別安全的檢查，確保資料、類別可以正確無誤的在不同的系統間轉換。這個機制稱之為 Common Type System。

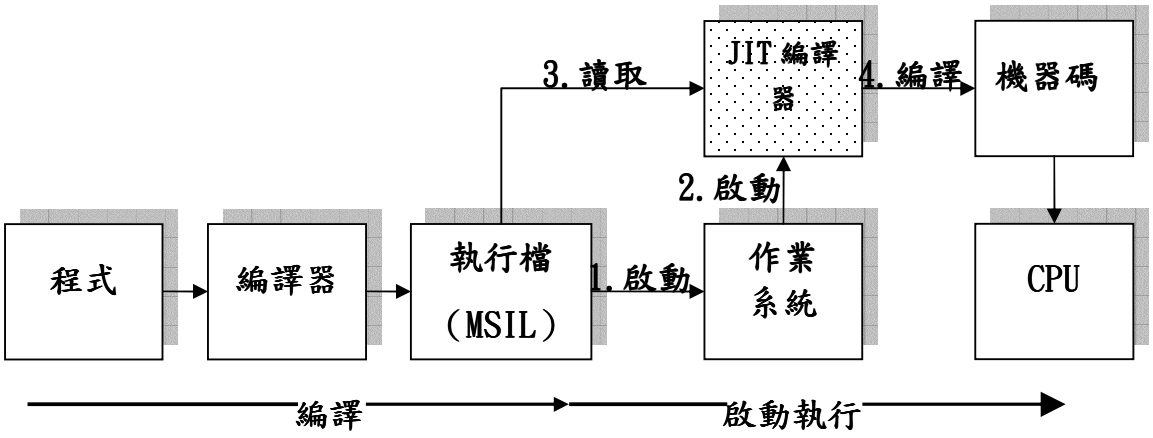


圖 2-1.NET 程式的編譯及執行模式

2.1.2.2 基底類別庫

基底類別庫是.NET Framework 第二個主要的部分，它是一個非常龐大的，並且已經預先寫好程式碼的集合，可以透過 C# 及 Visual Basic 及 Visual C++和其他 Visual Studio 語言使用基底類別庫以開發應用程式。基底類別庫自然而然會提供寫程式的需求。舉例來說，若程式需要去處理建立 Form 的許多細節，而基底類別庫就會提供 From 類別，因此就不需要去處理建立 Form 的許多細節了。程式碼只須宣告一個新的 Form，CLR 編譯器就會從.NET Framework 的基底類別庫取得 Form 所需的程式碼。透過這種方式寫程式，應用程式的體積會比以往的 Windows 應用程式小上許多；所需要的幾千行程式碼早已寫

在基底類別庫中了，因此應用程式執行檔(.EXE)並不需要包含所有的資訊。

在 VB.NET 應用程式中所需的所有元素－包含表單、按鈕功能表及其他要項，全部都是來自於基底類別庫。因為其他的 Visual Studio 應用程式也是使用相同的基底類別庫，所以如果你要在應用程式中混合不同的語言也十分的簡易。同時，分散式應用程式的部署也十分簡單，因為應用程式所需的支援訊息早已都安裝在機器上了。

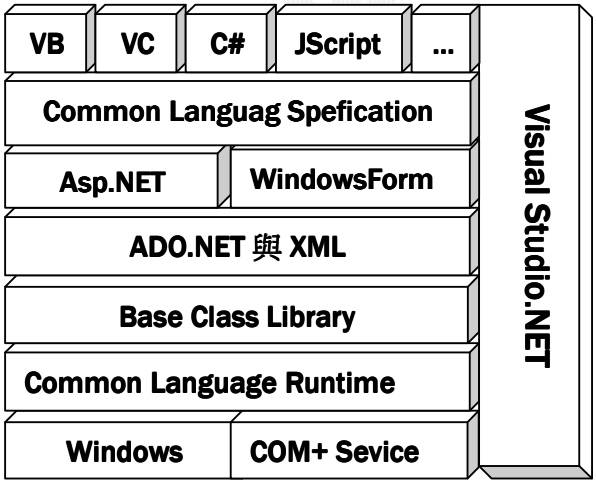


圖 2-2 .NET 整體架構

2.2 Adam

2.2.1 Adam 介紹

Adam 系列產品為研華科技股份有限公司所生產之產品，是

一內建微處理器之智慧型電腦介面感應器模組　　，透過簡單的 ASCII 格式的指令和 RS-485(雙向性傳輸線路標準)的協定做到遠程的控制，對 RS-485 網路傳輸只需要 2 條線路：DATA+和 DATA -。Adam 提供信號判斷、A/D 和 D/A 轉換、資料比較、和數位通訊函式。有些模組提供繼電器和 TTL 裝置之數位輸出/輸入線。

Adam 模組沒有開關。使用者可以僅僅藉由從電腦傳出的指令來改變一類比輸入信號去接受電壓、電偶或 RTD 輸入的幅度。遠程結構可藉由使用以選單為主之軟體或指令型式來完成。

2.2.2 Adam 4522

ADAM 4520/4522 轉換器如圖 2-3 所示，可以讓使用者使用較大的傳輸速度、傳輸範圍和能使現今具有 RS-232 介面的網路裝置。在 ADAM 4520/4522 內的電路系統自動地感應資料流的方向、在網路上的設備間交換信號 ，且可省去網路中 handshaking 的要求，然後將 RS-485 網路的線路需求數量減到只用 2 條。傳輸速度可達 115.2Kbps。

圖 2-3　Adam 4522

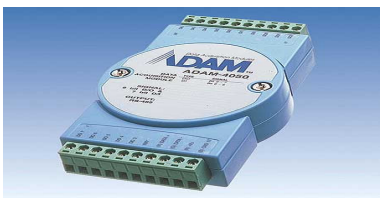


圖 2-4　Adam 4050

2.2.3 Adam 4050

Adam 4050 如圖 2-4 所示，具有 7 個數位輸入孔和 8 個數位輸出孔。電腦可使用模組的數位輸入來決定狀況、遠端的數位信號；可透過繼電器 (Solid-state relay)來使用數位輸出控制各式電器設備。

2.3 網路攝影機

本系統採用 AXIS 2100 Network Camera 如圖 2-5 所示，來作即時監控。在任何的標準瀏覽器可觀看 JPEG 格式影像都能被用 、取得、轉移而且看實況影像並且設定裝配是容易的 ，沒有額外的硬件或軟體被需要。

AXIS2100 網路照相機是為一個單獨的照相機，與一內建的網頁伺服器。你使用照相機的 URL 位址去擷取影像。而不需要個人電腦，它有它自己的 IP 位址。一個快速攝影或影像串流被觸發來自瀏覽器的請求即時攝影被 CAMERA 所擷取且被壓縮成 JPEG 格式並透過網路傳回瀏覽器中。高達 10 張/秒的傳輸速度，這幾乎是依賴網路的傳輸速度與 PC 的執行速度而定。

圖 2-5　AXIS 2100 Network Camera

2.4 Solid State Relay , SSR

固體繼電器(Solid State Relay)如圖 2-6 所示，具有開關速度快、靈敏度高、接點電流大、體積小、無接點火花等優點，正逐步廣泛地應用在各方面。固體繼電器包含以下四點特性：

1. 輸入觸發電壓由 3 至 32VDC。
2. 輸入與輸出開關之間用光電耦合隔離，安全可靠。
3. 輸出負載最大電流為 10A (80 至 280VAC)。
4. 輸出有防振蕩保護電路。

圖 2-6　Solid State Relay

第三章系統架構與升級

在本章將說明遠端家電系統的軟、硬體架構、系統的功能介紹，以及升級方法－利用升級精靈升級的評估。

3.1 硬體架構

本系統的硬體架構如圖 3-1 所示，遠端遙控電器可透過以下硬體控制步驟來達成，

步驟 1. Client 端傳送控制命令至 server 端。

步驟 2. Server 端接收到之後，透過 RS-232 將控制指令送至 ADAM 控制模組。

步驟 3.透過繼電器(SSR, Solid-State Relay)啟動電器。

步驟 4.遠端使用者可透過 Network Camera 檢視屋內電器的啟動情形。

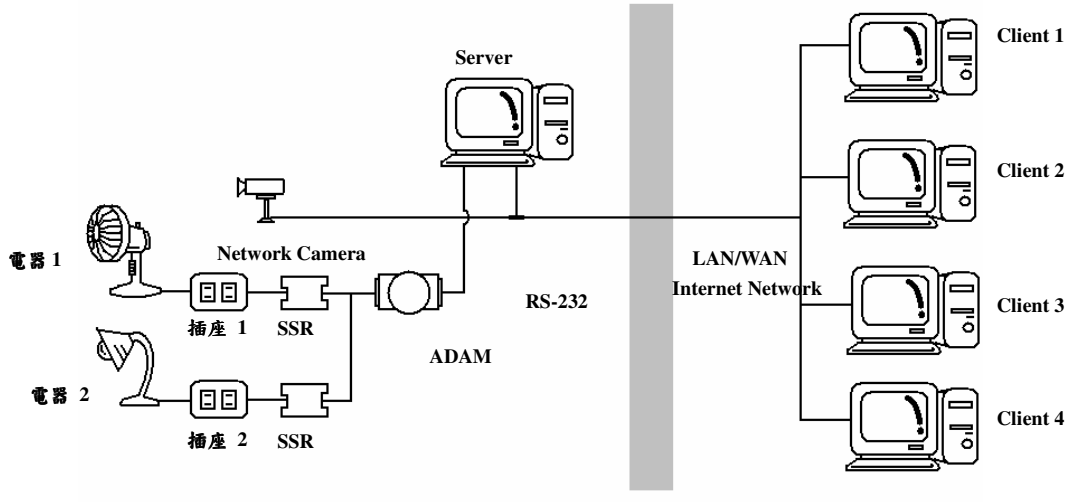


圖 3-1 硬體架構圖

3.2 軟體架構

本系統結合了資訊家電與數位監控系統並透過網際網路達到遠端遙控家電以及自動化的監控保全。系統軟體架構如圖 3-2 所示可分成**資訊家電**和**數位監控**兩個部分，並透過網際網路加以整合應用。

1、資訊家電：配合硬體裝置，讓使用者可在遠端控制屋內電器，或是透過定時裝置，讓電器在指定的時間內啟動，此外使用者在屋內也可透過簡單的肢體動作來控制電器。

2、數位監控部：配合攝影機可及時監控屋內狀態，當屋內出現可疑物體時，系統會自動追蹤物體並分析物體行為，若判斷出偵測的可疑物體為竊賊時，會自動啟動錄影功能，並同時對屋內的行動電話發出手機簡訊，顯示屋內遭入侵的訊息。另外，使用者在遠端也可透過攝影機觀看屋內狀態。

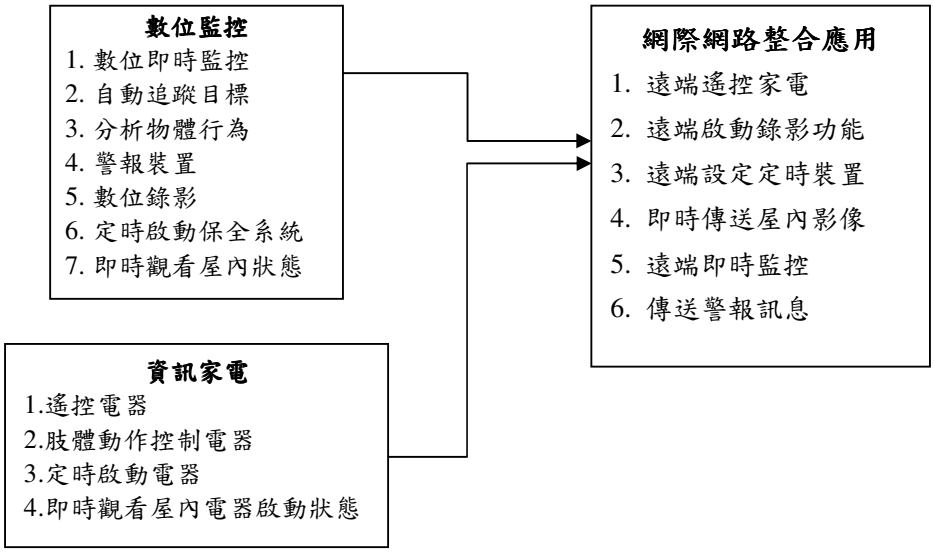


圖 3-2 軟體架構圖

3.3 軟體功能

遠端家電控制：本功能主要是提供使用者不在家中時也可以透過網路來控制屋內的電器，只要預先在家中將電器安裝在硬體控制裝置上，並透過 Network Camera 傳送屋內影像，位於遠端的使用者便能透過軟體接收影像並可直接在畫面上點取欲控制的家電來改變該它的狀態。所有在家電端與遠端之間的資料傳送皆以對稱式的編碼方式來確保系統的安全性。在家電端的 server 模式中可以管理所有登錄使用者的帳號、密碼、連線的來源 IP 及控制權限。

智慧型保全系統：外出時，使用者可利用保全功能對屋內作有效的監控。系統啟動時會自動調整與設定保全裝置，使用者不需作任何設定即可有效地偵測出所有可疑物體並加以分析與追蹤。當氣候或天色改變時，即使外在環境透過窗戶對屋內影像產生變化，系統也會自

動調整設定，不會影響到系統的正常運作。當系統判斷出偵測到的可疑物體為竊賊時，會自動啟動錄影功能，將整個行竊過程記錄在家電端與遠端的電腦中，並同時對屋主的行動電話發出手機簡訊，顯示屋內遭入侵的警告訊息。

定時啟動功能：系統中並包含了定時功能，提供使用者在指定時段啟動指定的物件，如家電、錄影功能或保全功能。例如您可以在下班前啟動家中的電鍋，到家前關閉保全系統以避免自己觸發警報，甚至在特定時段啟動系統的錄影裝置記錄保姆育兒狀況。

錄影管理功能：此功能主要是可以自動儲存與管理系統中錄影功能所錄的影片，其中可分為手動錄影、保全錄影和定時錄影，使用者可自行設定影片的儲存品質，系統除了會自動儲存影片並提供完整的播放功能，讓使用者可以很方便地檢視所有記錄的影片。

虛擬感應開關：虛擬感應開關主要是提供使用者在屋內的特定區域以手部揮動的感應方式來改變單一或多項物件的狀態，如電器、錄影功能或保全功能。使用者可以利用主工具列中的新增物件功能產生虛擬感應開關，並使用虛擬感應開關功能中的設定功能來編輯欲啟動的物件，完成之後使用者可在屋內所設定的區域中以手部來回揮動一次改變連結物件的狀態。另外，虛擬感應開關可以直接設在電器物件上,使用者可以直接在電器前以手部來回揮動一次改變該電器的狀態。

3.4 平台

作業系統 : winsows xp

硬體 : 筆記型電腦、Adam 控制模組、SSR(Solid-State Relay) 、Network Camera

軟體 : Visual Studio.NET 2002

3.5 升級方法

VB6.0 和 VB.NET 的差異很大，VB.NET 已完全物像導向，並且要從以桌上電腦為中心的模式，轉變成以全球資訊網為中心 .NET 模式，本非易事，因此微軟提供升級精靈，為了協助程式設計師找出程式碼中的問題，然後設法將 VB6.0 專案升級至 VB.NET 專案，因此我們先嘗試利用 “ 升級精靈“ 來執行專案的升級,看是否能順利升級成功，若否，那麼應用升級精靈會處理那些問題及產生那些錯誤和警告訊息呢?對程式設計師是否有幫助呢?

3.5.1 使用升級精靈

當利用 VB.NET 試圖開啟 VB6.0 的專案時，就會自動啟動該精靈，如下圖所示。

圖 3-3 升級精靈開始畫面

升級精靈會自動轉換 VB6.0 專案為 VB.NET 專案，升級精靈會修改語言中的語法變更，並將 Visual Basic 6.0 表單轉換為 Microsoft Windows® Form。此外，升級精靈還會產生報告，警告程式設計者任何需要在程式碼中進行的手動變更。註解會以「待辦」工作的形式顯示在新的 [工作清單] 視窗中，只要連按兩下某工作項目，便可瀏覽到程式碼陳述式。

升級精靈報告:

```
Public Sub ClearSetBars()
        ClearSet()
        'UPGRADE_ISSUE: PictureBox 方法 TotalTime.Cls 未升級。
按一下以取得詳細資訊:
```

```
'ms-help://MS.VSCC.2003/commoner/redir/redirect.htm?keyword="vb
up2064"'
        TotalTime.Cls()
        'UPGRADE_ISSUE: PictureBox 方法 rule2.Cls 未升級。 按一
下以取得詳細資訊:
'ms-help://MS.VSCC.2003/commoner/redir/redirect.htm?keyword="vb
up2064"'
        rule2.Cls()
    End Sub
```

3.5.2 升級精靈缺點

從 3.5.1 節中，升級精靈似乎對程式碼的升級有很大的幫助，即使沒辦法完全升級成功，也提供了報告來幫助程式開發者，但實際上，VB6.0 和 VB.NET 差異實在太大，升級精靈並沒有辦法全部涵蓋，雖然它會轉換一些程式碼，或許可以使我們在改寫程式碼時，能夠節省一些時間，並且注意一些細節，但經過我們實際的操作後發現有幾個缺點：

3.5.2.1 控制項轉換

在轉換控制項上，尤其是有圖片的控制項， VB.NET 完全無法升級，如圖 3-4 原圖片與轉換後的結果相差甚多。

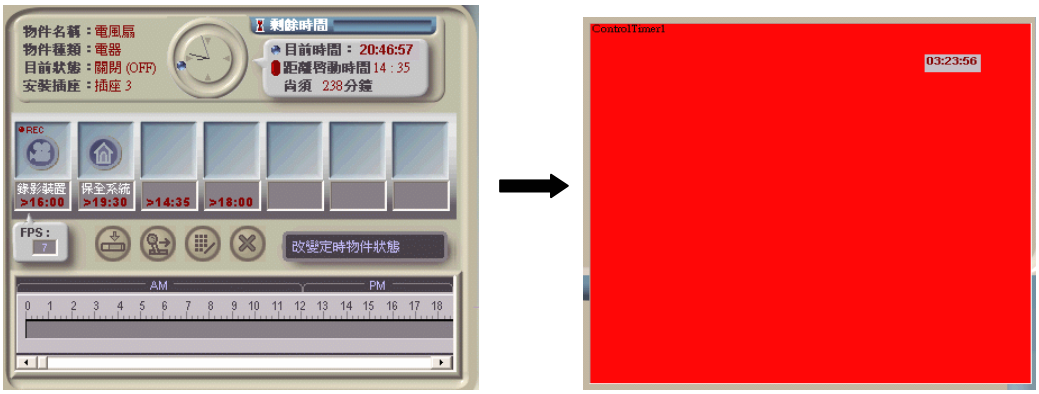


圖 3-4 圖片升級前後

3.5.2.2 隱性宣告轉換

在一般 VB6.0 中的“Explicit Off”即所謂的隱性宣告，它允許程式中的變數不用宣告，然而被升級到 VB.NET 後，因 VB.NET 會將 Explicit On ，所以這些之前在 VB6.0 沒有被宣告的變數就會被升級到 Object ，雖然這樣的改寫看起來並無不當，但因為編譯器無法得知此變數在設計上的意圖，所以編譯器會插入額外的邏輯操作在程式執行時聯繫物件的方法或是物件，因此這樣的執行會比較沒有效率並且拖慢程式的執行速度，甚至在大的程式碼中會出現不可預期的錯誤。

3.5.2.3 VB6.0 程式未標準化

對於不符合原 VB6.0 語法的程式碼，雖然在 VB6.0 中依然可以被正確編譯執行，但對 VB.NET 升級精靈的升級時，無法找出適當對應的轉換語法，而無法升級，又或者它會轉換成與原語意不相符的程式碼。

3.5.3 對“升級精靈”評估

由 3.5.2 節中各點看來，升級精靈應用在專案簡單且標準化之 VB6.0 專案時，是可完全升級成功，但應用在複雜且龐大且原 VB 語法使用不符合規格的專案，使用升級精靈後，得到的是一堆排山倒海的錯誤，反而讓程式變得更複雜，且除錯困難；而且在我們專案中，圖檔控制項佔了很大的一部分，但升級精靈也無法將具有圖檔的控制項升級，所以使用升級精靈結果，僅可升級 30-40%程式碼。

3.5.4 C#與 VB.NET 的評估

由於升級精靈無法順利升級，因此在這階段，竇老師希望我們能評估一下，既然面對的都是陌生的語言，並且 VB.NET 升級精靈無法順利升級，若將系統升級到 VB.NET 或 C#兩者間做個衡量，用何種語言對專題及未來的發展較有益處，在經過我們的討論及評估之後，決定還是使用 VB.NET 語言，由於雖然升級精靈無法直接升級，但升級精靈的報告對我們再重新寫程式時依然有參考的價值，並且雖然無法直接升級，但經過評估之後發現，可再使用的程式碼依然有 40%，會比使用 C#語言從零開始來的有益。

3.5.5 決定升級方法

因此，我們決定了我們的升級方法與語言，不使用修改使用升級

精靈後的程式，改用參考升級精靈的報告，一步步改寫的方式，從設計階段開始，一方面比修改使用升級精靈後的程式來的有效率，一方面可對 VB.NET 的語法有更深的了解。

3.5.6 2002 與 2003 升級精靈的比較

在開始專題實作的時候，我們是利用 Visual Studio 2002 的版本來升級，但在專題製作的後期，微軟推出了新的版本 Visual Studio 2003，在我們使用新版本的時候發現，2003 加進了一些新的改變，比以往更容易移轉為 Visual Basic .NET。

一、 **圖片控制項**：對於圖片的控制項在 2002 版本中是完全無法升級，但在 2003 的版本已可升級 85%的面板，剩下的 15%，只須靠手動調整圖片的位置與大小。

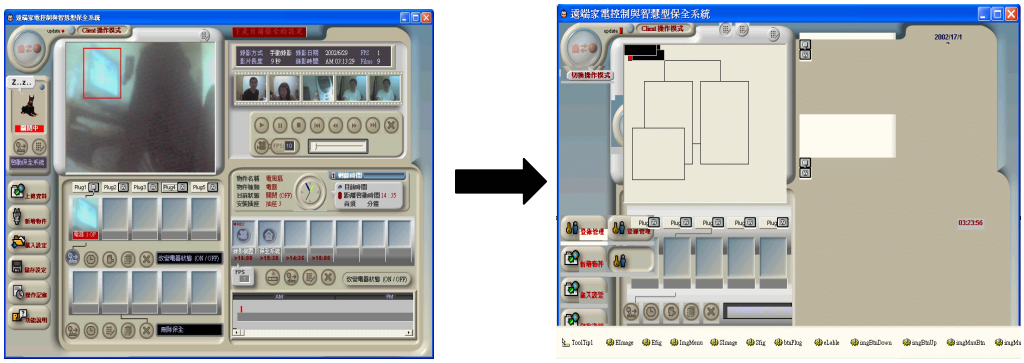


圖 3-5 2003 升級圖片控制項

二、 **控制項陣列**：在.NET 中不支援控制項陣列的使用，因此原本在 2002 版本須手動升級，但 2003 版本以可自動升級這部分，使用方式直接使用_sLable_1，_sLable_2…..操作控制項陣列各元

素，.NET 所加入的方法如下。

1. 先宣告各物件

```
Public WithEvents _sLable_0 As System.Windows.Forms.Label
Public WithEvents _sLable_1 As System.Windows.Forms.Label
Public WithEvents _sLable_2 As System.Windows.Forms.Label
```

2. 宣告 sLabel 為一物件陣列

```
Public WithEvents sLable As
Microsoft.VisualBasic.Compatibility.VB6.LabelArray
```

3. 將各控制項加入 Form 中

```
Me._sLable_2 = New System.Windows.Forms.Label
Me._sLable_1 = New System.Windows.Forms.Label
Me._sLable_0 = New System.Windows.Forms.Label
```

4. 將 sLable 加入 Form 中

```
Me.sLable = New
Microsoft.VisualBasic.Compatibility.VB6.LabelArray(Me.components
)
```

5. 開始化

```
CType(Me.sLable,
System.ComponentModel.ISupportInitialize).BeginInit()
```

6. 結束化

```
CType(Me.sLable,
System.ComponentModel.ISupportInitialize).EndInit()
```

三、 在整合開發環境啟動速度加快，輸入程式碼時快速、自動的格式化處理；更容易處理方法、屬性和程式錯誤的改良。

第四章 系統升級實做

本章節主要在描述在系統升級實做的過程，我們以模組的分類方式，將我們由實作中發現的 VB 與 VB.NET 各種差異提出說明，若在各模組升級中都會用到的改變，會獨立出來說明。

4.1 Event 在 VB 6.0 與.NET 之間的改變

Event 在許多模組升級中皆有使用到，因此先在此提出說明:

- Visual Basic 6 的事件（Event）
  - Visual Basic 6 使用名稱和事件之間建立關聯性，方式是：＜物件名稱＞_＜事件名稱＞;例如和 CommandButton 指令相關的點選事件：

```
Private Sub Command1_Click()
```

  - 假如你改變 Vb6 中的事件名稱，讓它成為不和任何事件相符的子程式名稱，它就只是一個簡單的子程式。因此，名稱樣式決定了一個子程式是否能成為一個事件。
- Visual Basic .NET 的事件（Event）
  - VB.NET 並不使用名稱和事件建立關聯性；如果一個子程式包含 Handles 子句的話，它就和事件有所關聯；子程式的名稱可以是任何名稱，它所觸發的事件則在 Handles 子句中。例如和 VB.NET 按鈕相關的點選事件：

```
Private Sub Command1_Click(ByVal sender As System.Object,
```

```
ByVal e As__  System.EventArgs) Handles Command1.Click
```

- VB.NET 也將 Command1 的 Click 事件建立名為 Command1_Click。但是，名稱只是讓程式碼更容易辨識罷了。事實上，喜歡為事件常式取什麼名稱都可以；像是可以將 Command1_Click 事件修改為如下的 YouClickedMyButton：

```
Private Sub YouClickedMyButton (ByVal sender As
System.Object, __ ByVal e As System.EventArgs) Handles
Command1.Click
```

- 另一個關於事件的改變是 VB6 和 VB.NET 中事件的參數並不相同；在 VB6 中，事件子程式包含了每個參數的名稱和類型，而在 VB.NET 中，參數被包裹在 EventArgs 中，並以關聯性的方法來傳遞到物件；當然，在 VB.NET 事件的事件子程式中，也包含觸發事件的物件。

- 以下是兩個版本的 VB 中，處理不同事件參數的例子；考慮一個有 ListBox 空項的表單，必須撰寫程式碼來展示以選取的項目。VB6 中，需要撰寫如下列的程式碼：

```
Private Sub List1_ItemCheck(Item As Integer)
    MsgBox ” You Checked Item：” & Item
End Sub
```

在 VB.NET 中對應的程式碼為：

```
Private Sub CheckedListBox1_ItemCheck(ByVal sender As __
```

```
System.Object, ByVal e As __
System.Windows.Forms.ItemCheckEventArgs)__
Handles CheckedListBox1.ItemCheck
    MsgBox（” You Checked Item：” & e.Index）
End Sub
```

- 觀察 VB 6 例子中被選取的項目，是如何直接以參數進行傳遞 (Item As Integer)；在 VB.NET 中，它則是以傳遞 ItemCheckEventArgs 物件 e 的成員來進行。

4.2 面板的升級

由於拿到原始碼時，並無附上當初在設計這些面板的原始資料，單單只有去得原始程式碼的部分，以致於我們需要使用螢幕抓圖，再影像編輯軟體製成可用的面板，製作的步驟雖然都是反覆的作一樣的動作，但是過程卻是相當麻煩的，現在介紹步驟如下：

Step1:首先，開啟一個模組的原始碼及介面

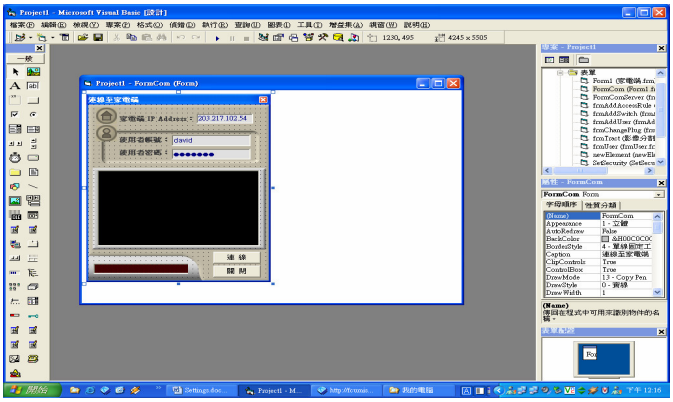


圖 4-1 開啟模組的介面

Step2:切換到介面模式，選定要製成新圖形的部分後，將其餘的部分通通刪除，並且將此模組的原始碼也通通刪除，避免再執行時產生錯誤訊息。

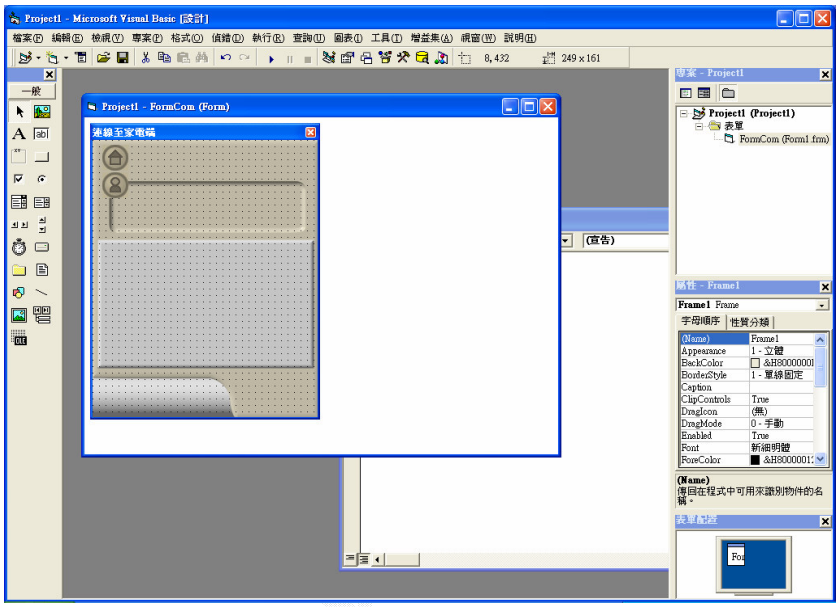


圖 4-2 切換介面模式

Step3:執行起來後，畫面上只會顯示我們需要的部分，此時按下 ALT+PrintScreen 以擷取螢幕畫面。

圖 4-3 執行後畫面

Step4:開啟影像編輯軟體，開一個新檔案，檔案的 SIZE 要符合所抓圖形的大小。

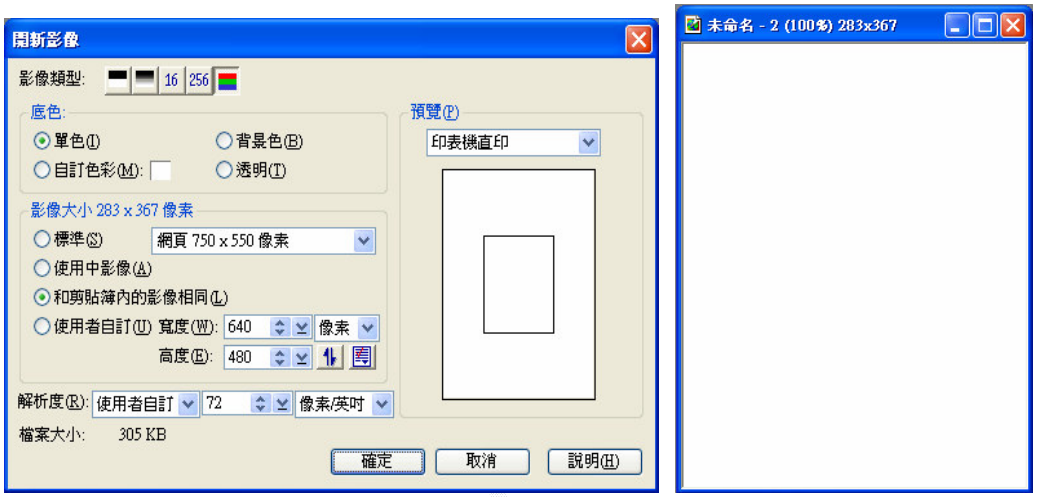


圖 4-4 開啟影像編輯軟體

Step5:按貼上，將所擷取的畫面貼上，剪裁需要的部分，另存成新的圖形，便完成圖形的製作。

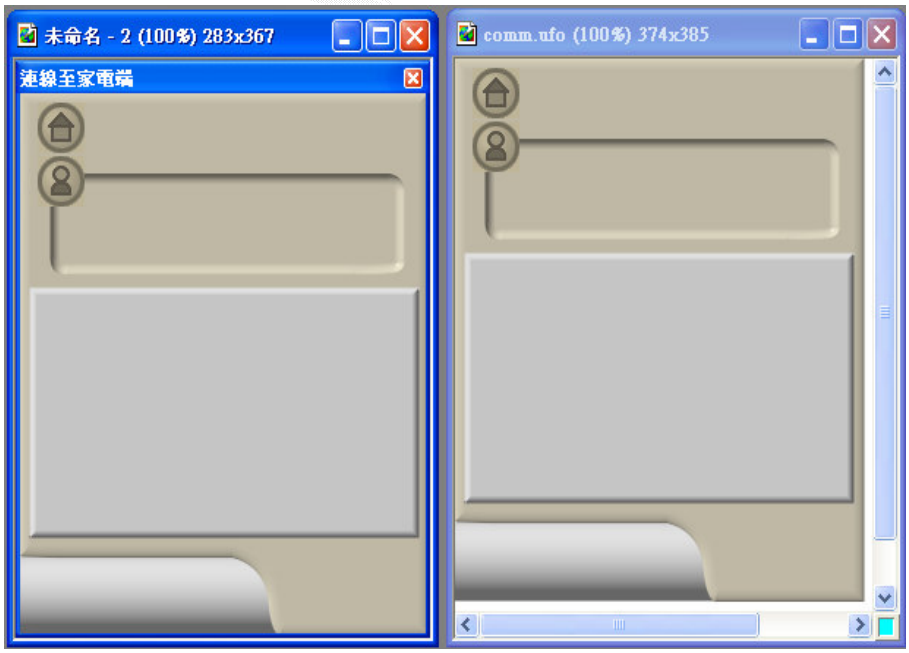


圖 4-5 貼上、裁剪圖片

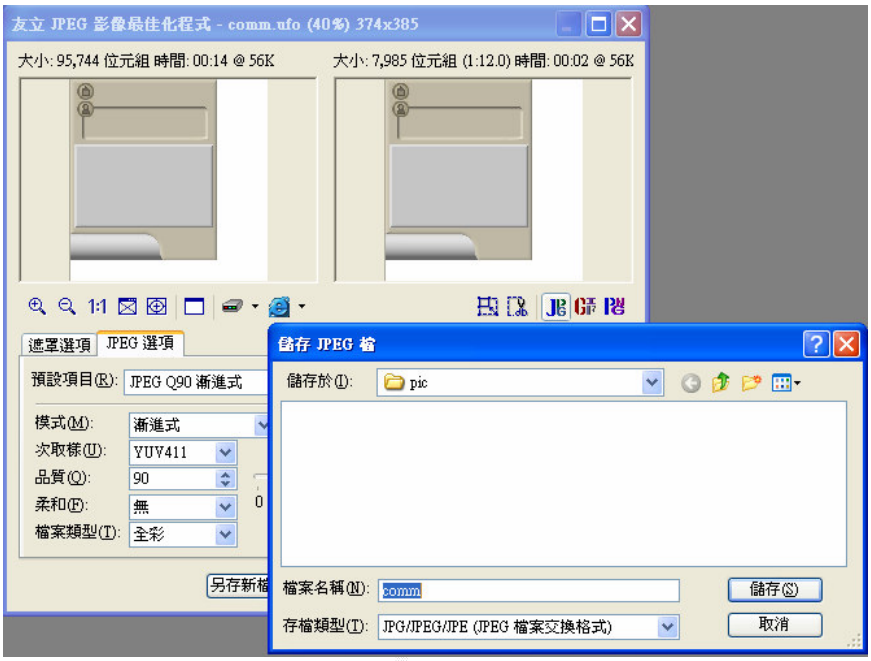


圖 4-6 儲存 JPEG 檔

Step6:最後，就可以在 VB.NET 中將圖形載入使用。

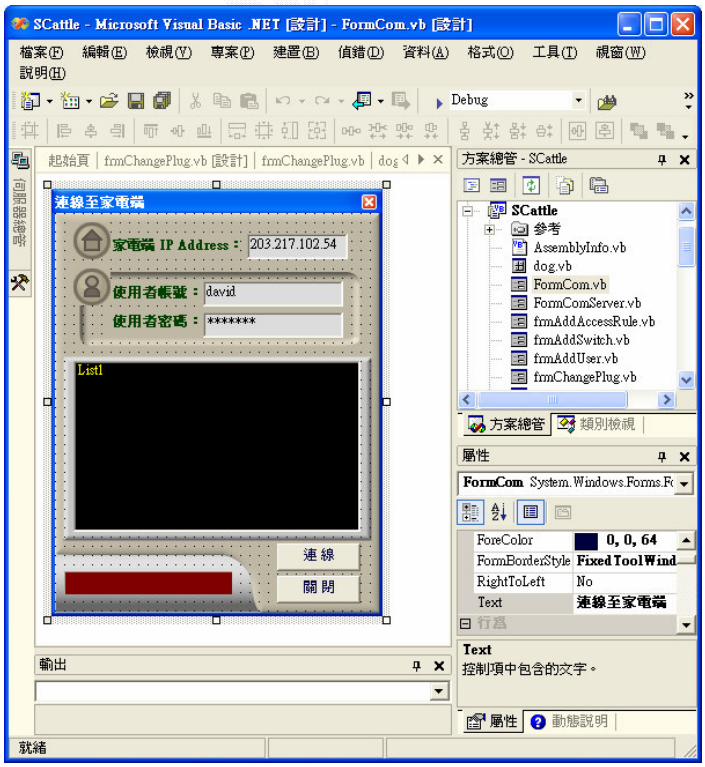


圖 4-7 載入 VB.NET 圖形

4.3 Dog

- 視窗名稱：無
- 表單名稱：Dog
- 模組檔名：Dog.ctl
- 升級至 VB.NET 後之檔名：Dog.vb
- 升級到 VB.NET 和 VB 之不同處：
  - 由於 Dog 這個使用者控制項當初在設計時，需要呈現狗會搖頭，會吠的動畫效果，但是在傳統的 VB 環境下無法使用設計好的 GIF 動畫檔來達成所需要的效果，雖然能讀取 GIF 檔案，但是卻只能顯示出第一個畫格，而無法呈現 GIF 檔案本身動畫的效果。
  - 所以在 VB 環境下實做這個使用者控制項時，便必須採用將多張分解動作圖片配合使用迴圈的方式，設定每 0.3 秒中更換下一張圖，達成動畫的效果，但是此種方法卻是非常麻煩和不符合設計師期望的需求。
  - VB 程式碼：利用迴圈控制圖片的播放

```
Private Sub Timer1_Timer()
    Static count As Integer
    If Mode = 1 Then
        imgDog.Picture = DogWatch(count).Picture
        imgLight.Picture = imgGreen(count Mod 2).Picture
    Else
```

```
            imgDog.Picture = DogBark(count).Picture
            imgLight.Picture = imgRed(count Mod 2).Picture
        End If
        count = count + 1
        If count = 8 Then count = 0
End Sub
```

- 於是在新版的 VB.NET 裡，改善了這種情況，對於圖片的使用就無此限制，所以在升級至 VB.NET 時，便將原本包含在 VB 原始程式裡的動畫分鏡圖片利用圖片擷取的方式，把一張一張的圖片先做成單格的 GIF 圖檔，再利用 PhotoImpact Gif Aninmator 將多張圖片製作成具動畫效果的圖檔，讀入 VB.NET 中使用，完成此部分的升級。
- 相較之下，VB.NET 在圖檔上的使用較 VB 優秀許多，能夠直接使用具動畫效果的 GIF 檔，使得這個部份的程式碼變的相當簡潔，只須控制需要的 GIF 圖檔出現的時間即可。
- 雖然在製作這部份圖檔呈現上，在 VB 和 VB.NET 使用不同的方式，但是使用者在使用的時候，畫面上所呈現的效果，卻是感覺不出有任何差異。

- 模組畫面：
  - VB 裡的設計畫面：
    - 在 VB 環境裡，可以看出圖片是拆解開的，需利用迴圈的

控制來達到動畫的效果，並且使用的圖檔很多。

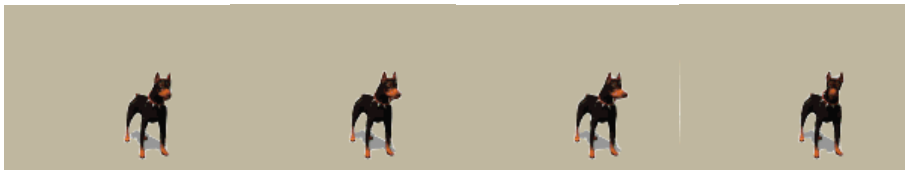


圖 4-8 VB 中 Dog 分解動畫 a

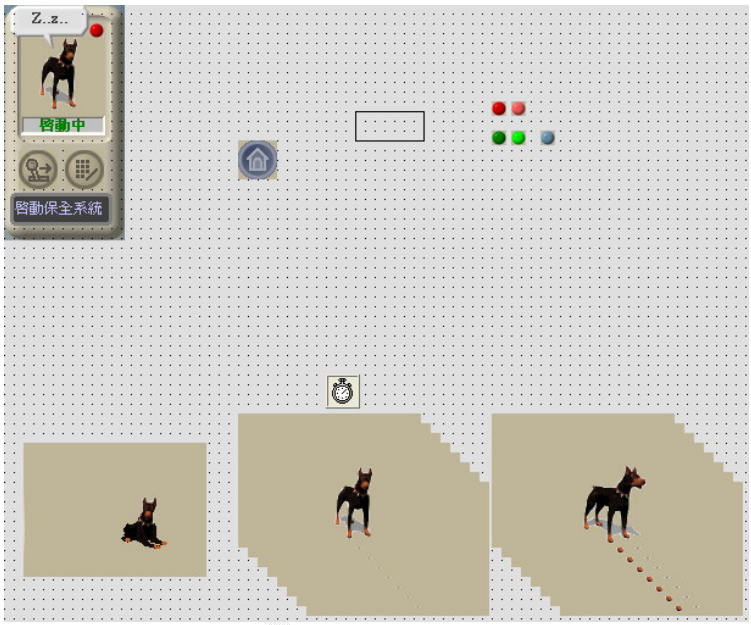


圖 4-9 VB 中 Dog 分解動畫 b

- VB.NET 裡的設計畫面：
  - VB.NET 裡可以直接使用 GIF 檔案，呈現動畫的效果，使得程式碼和使用的圖檔大大的減少。

圖 4-10 VB.NET 直接使用 GIF 檔

■ 實際顯示之畫面：

圖 4-11 實際顯示畫面

● VB 和 VB.NET 在圖檔讀取時語法上的差異介紹：

■ 在 VB 下使用圖檔的方式：

◆ VB 圖檔的使用可以使用 PictureBox 和 Image，2 種物件來顯示圖片，但 PictureBox 物件更可以載入圖片、顯示文字、畫圖，比 Image 功能來的強很多。2 個物件在載入圖檔上的方式是在屬性一欄的 Picture 屬性，這一個屬性專門在指定所要放的圖片，在 Picture 屬性後的〝...〞按鈕，點一下選擇想要載入的圖片，可以接受的圖片格式有：BMP、DIB、JPG、GIF、WMF、EMF、CUR、ICO 等。選好按〝確定〞鈕。2 種物件有支援 GIF 的透明背景，但不支援 GIF 的動畫。

圖 4-12 屬性視窗　　**圖 4-13 載入圖片視窗**

- 在 VB.NET 下使用圖檔的方式：
  - 在 VB.NET 的環境下已經將 PictureBox 取代 Image 物件的功能，可以接受的圖片的格式和 VB 一樣，在載入圖檔上的方式和 VB 不一樣，改成 Image 這個屬性來選取所要的圖檔，並且支援 GIF 的透明背景和動畫，因為此種改變，使得在升級這個部份的模組時得到了許多好處，減少圖檔使用和簡化程式碼撰寫。另外，在 VB.NET 可以透過 Picture.Image.FromFile(“路徑\檔名”)，由磁碟機中載入圖片到 PictureBox 控制項中。

- 程式碼可再利用之程度：60%
- 製作至完成升級所需時間：2 小時

4.4 Settings

- 視窗標題：設定
- 表單名稱：Settings
- 模組檔名：Settings.frm
- 升級至 VB.NET 檔名：Settings.vb
- 模組畫面：

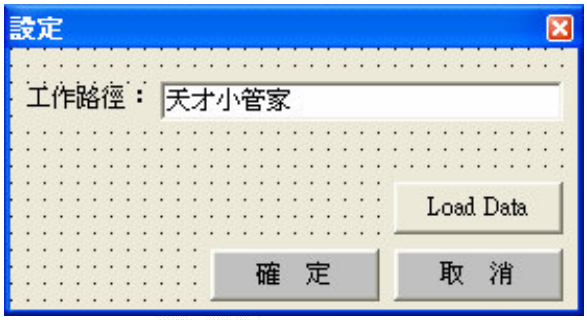


圖 4-14 “設定”視窗

- 升級到 VB.NET 與 VB 之不同處：
  - 由於這個模組純粹是用來設定工作路徑，藉以儲存程式產生的設定資料，將路徑存在 CurrentPath 這個變數裡，供程式呼叫。
  - 但因表單的介面簡潔以及的程式碼的內容相當的少，所以在升級時可以直接升級，不需變更，所以這個模組的升級是相當快速的。
- 製作至完成升級所需時間：10 分鐘
- 程式碼可再利用之程度：100%

4.5 LightText

- 表單標題：無
- 表單名稱：LightText
- 模組檔名：UserControl1.ctl
- 升級至 VB.NET 後之檔名：LightText.vb
- 模組畫面：

圖 4-15　LightText 畫面

- 升級到 VB.NET 和 VB 之不同處：
  - 在 VB 要撰寫一個物件的屬性會利用以下的程式碼，其中 Property Get 是用來讓使用者讀取屬性，而 Property Set/Let 則是用來設定屬性，至於是要使用 Set 還是 Let 則要看賦予的屬性是物件還是一般的變數型別，而每設定一個 Get/Set/Let 關鍵字就利用一組 Public Property … End Property 隔開，個別設定自己的屬性，程式碼如下：

```
Public Property Get Text() As String
    Text = pText
End Property

Public Property Let Text(ByVal vNewValue As String)
    pText = vNewValue
End Property
```

- 而在 VB.NET 中已經將 Set 和 Let 關鍵字合併只使用 Set 關鍵字，並且現在 VB.NET 中已經將 Get/Set 集成到一個 Public Property … End Property 結構中去，不分開使用了，程式碼如下：

```
Public Property Caption() As String
    Get                                          //傳回屬性質
        Return pText
    End Get
    Set(ByVal vNewValue As String)               //設定屬性
質
        pText = vNewValue
    End Set
End Property
```

- 製作至完成升級所需時間：1 小時
- 程式碼可再利用之程度：50%

4.6 FormCom

- 表單標題：連線至家電端
- 表單名稱：FormCom
- 模組檔名：Form1.frm
- 升級至 VB.NET 後之檔名：FormCom.vb
- 模組畫面：

圖 4-16 “連線至家電端”視窗

- 升級到 VB.NET 和 VB 之不同處：
  - Label 文字標籤控制項的功能最主要在〝顯示資料〞，要設定或是改變 Label 裡顯示的顯示資料時，VB 和 VB.NET 裡的用法已經改變。在 VB 裡是用 Caption 屬性來設定；VB.NET 裡則是改用 Text 屬性。
  - ListBox 控制項用來提供一串列的文字項目清單來供選擇。ListBox 控制項許多屬性在 VB 和 VB.NET 裡已經變更，以下

列出在此模組使用到不同的部份：

Visual Basic	Type	Visual Basic .Net
AddItem	Method	Items.Add
Clear	Method	Items.Clear
Click	Event	SelectedIndexChanged
ListIndex	Property	SelectedIndex

表 4-1 ListBox 控制項差異

- Variant 的資料型別如果沒有明確指定變數的型態，它可以代表任意資料型別；在 VB.NET 中，則以 Object 類別來替代 Variant 宣告，它綜合了 VB6 中的 Object 和 Varient 類別；如同 VB6 中的的 Object 一樣，VB.NET 的 Object 類別可以被指派為另一個物件的執行個體，也如同 VB6 中 Variant 宣告一樣，可以指派任何宣告為任何型態。
- 在 VB 裡有著一些存在著一些常用的函式，當然在 VB.NET 裡也保留下來了，但是呼叫方式卻有相當的改變。以在這個模組是用到 Right 函式為例：
  - 在 VB 下的用法是

```
Recieve_SubString = Right$(ReciveTxt, Len(ReciveTxt) - 1)
```

  - 在 VB.NET 下要使用的函式時，由於這些函式都已經被包含在 Microsoft.VisualBasic 函式庫裡，所以直接使用下列語法即可達成與 VB 裡一樣的效果

```
Imports VB = Microsoft.VisualBasic
Recieve_SubString = VB.Right(ReciveTxt, Len(ReciveTxt) - 1)
```

或是

```
Recieve_SubString = Microsoft.VisualBasic.Right(ReciveTxt,
Len(ReciveTxt) - 1)
```

- 製作至完成升級所需時間：2 小時
- 程式碼可再利用之程度：60%

4.7 Module1

- 表單標題：無
- 表單名稱：無
- 模組檔名：Module1.bas
- 升級至 VB.NET 後之檔名：Module1.vb
- 模組畫面：無
- 升級到 VB.NET 和 VB 之不同處：
  - 這個 Module 裡包含了相當多使用者定義型別（User Defined Type）的結構（Structure），其宣告方式在 VB 和 VB.NET 已經變更，介紹如下：
  - VB 中要使用結構的宣告方式以 Type 這個關鍵字開頭，介紹如下：

```
Type 結構型別名稱
    欄位名稱 As 資料型別
```

```
End Type
```

■ VB.NET 中使用結構的宣告方式則改以 Structure 這個關鍵字開頭，介紹如下：

```
Structure 結構型別名稱

    Dim 欄位名稱 As 資料型別
End Structure
```

■ VB 和 VB.NET 之間，使用者自訂型別還存在一個差異：VB 在使用者自訂型別裡支援固定長度的陣列宣告；然而 VB.NET 卻無法詳細指定使用者定義類別的陣列大小。為了解決這個問題,可以在使用者自訂型別裡定義一個建構式來將陣列初始化成固定長度。以這個 Module 的一個 Structure 為例，介紹如何解決這個問題：

■ 在 VB 裡的原始程式碼

```
Type CUser
    AccessRule(5) As AccessRule
End Type
```

■ 將 VB 升級至 VB.NET 後相對應的程式碼

```
Structure CUser
        <VBFixedArray(5)> Dim AccessRule() As AccessRule
        Public Sub Initialize()
            ReDim AccessRule(5)
        End Sub
End Structure
```

■ VB.NET 中的結構內加入了 VBFixedArray 屬性，可以詳細指

定陣列的長度。另外，就是使用了一個稱為 Initialize 的公開（Public）方法，這個 Initialize 方法裡的程式碼包含了 ReDim 呼叫，以用來初始化被包含在結構內的固定長度陣列。

- 基本上這樣是完成了 VB 到 VB.NET 中使用者自訂型別一些升級的問題，但實際上配合其他模組執行後會出現一個錯誤訊息”NullReferenceException「Object reference not set to an instance of an instance of an object・」”到 MSDN 上查詢後發現以上問題的原因是結構成員 CUser 並不包含任何陣列元素，為了初始化此陣列與其中的元素，我們必須在有用到 Module 裡的使用者自訂型別的模組中，在其模組中宣告一個結構變數之後，立刻呼叫其結構內 Initialize 的方法，即可解決這個問題。例子如下（假設這是在一個有呼叫使用 CUser 結構的模組中的 Main 部分）：

```
Public Sub Main（）
        Dim User As CUser
        User. Initialize()
End Sub
```

- 製作至完成升級所需時間：2 小時
- 程式碼可再利用之程度：80%

4.8 frmAddUser

- 表單標題：新增使用者
- 表單名稱：frmAddUser
- 模組檔名：frmAddUser.frm
- 升級至 VB.NET 後之檔名：frmAddUser.vb
- 模組畫面：

圖 4-17 “新增使用者”視窗

- 升級到 VB.NET 和 VB 之不同處：
  - 在模組畫面中（上圖）的四個 CheckBox 是屬於一個名為 Check1 控制項陣列的元素
  - 原始 VB 程式碼中利用下列程式碼來清除還原 Check1 控制項陣列元素的屬性（0）

```
For i = 0 To 3
        Check1(i).Value = 0
Next i
```

- 在 VB.NET 中只是將其控制項陣列取消掉，將原本的四個元素重新命名，拆成獨立的控制項使用，相當簡單地解決在 VB 控制項陣列升級到 VB.NET 的問題，程式碼如下：

```
Check0.Checked = 0
Check1.Checked = 0
Check2.Checked = 0
Check3.Checked = 0
```

- 製作至完成升級所需時間：1 小時
- 程式碼可再利用之程度：85%

4.9 frmAddAccessRule

- 表單標題：新增限制來源 IP 規則
- 表單名稱：frmAddAccessRule
- 模組檔名：frmAddAccessRule.frm
- 模組畫面：

圖 4-18 “新增限制來源 IP 規則”視窗

- 升級到 VB.NET 和 VB 之不同處：
  - 在 VB 中，模組畫面（上圖）中的 2 個按鈕（確定、取消）是一個簡單的控制項陣列，在這個模組中利用 Command1_Click(Index As Integer)的 Index 來決定是哪一個按鈕被選取，程式碼如下：

```
Private Sub Command1_Click(Index As Integer)
    If Index = 0 Then
        If optSelectType(0).Value = True Then
            frmAddUser.lstGoIP.AddItem txtIP.Text
        Else
            frmAddUser.lstStopIP.AddItem txtIP.Text
        End If
    End If
    txtIP.Text = ""
    frmAddAccessRule.Hide
End Sub
```

  - 而升級至 VB.NET 的方法，是採用將控制項陣列拆開的方法來升級，由於原本 VB 中的這個控制項陣列只有 2 個元件，所以決定分開來升級，比較方便，程式碼如下：

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
e                         As System.EventArgs) Handles
Button1.Click
        If optSelectType1.Checked = True Then
            AddUser.lstGoIP.Items.Add(txtIP.Text)
        ElseIf optSelectType1.Checked = True Then
            AddUser.lstStopIP.Items.Add(txtIP.Text)
        End If
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal
e                         As System.EventArgs) Handles
```

```
Button2.Click
        txtIP.Text = " "
        Me.Hide()
End Sub
```

- 製作至完成升級所需時間：1 小時
- 程式碼可再利用之程度：50%

4.10 ControlTimer

- 表單標題：無
- 表單名稱：ControlTimer
- 模組檔名：rul.ctl
- 升級至 VB.NET 後之檔名：ControlTimer.vb
- ControlTimer.vb 模組畫面：

圖 4-19 “ControlTimer.vb”視窗

4.10.1 升級問題一

控制項陣列是長期存在 Visual Basic 中的特色，一個控制項陣列容許多個控制項分享相同的事件和初始的屬性設定，可以使程式設計師套用相同的行為到控制項陣列的控制項中，下圖就是一個控制項陣列，分別從 ControlImg(0) ～ ControlImg(6)

圖 4-20 控制項陣列畫面

在副程式 btn_Click（）中就有其應用，

```
btn.Picture = btnAdd(1).Picture
```

其中 btnAdd(1) 就是在 btnAdd 控制項陣列中的其中一項，在升級過程中，最開始在 VB.NET 的畫面設計就遇到了不能將物件名稱設定為 btnAdd(1)，只因如此設定會出現錯誤訊息“ 無效的屬性值“ ，所以只能先將物件設定為 btnAdd_1，之後的控制項名稱，也是用類似的方式來命名。

- **VB 與 VB.NET 在控制項陣列的不同處**

VB.NET 中要把控制項陣列想像成包含控制項的標準陣列，必須要利用撰寫程式碼宣告此控制項陣列，而不是像 VB6.0 可以在設計初期就將控制項陣列設定完成，因此 VB.NET 中就要花更多時間來處理程式碼在控制項陣列上，例如在此時控 ControlTimer_Load（）的副程

式中，便要將 btnAdd(0)和 btnAdd(1)利用程式碼來宣告，如下

```
btnAdd.Load(0)
btnAdd.Load(1)
```

由此可見，若是只有一兩種控制項陣列的話，程式設計上不會那麼的複雜，但是在這程式中，常利用到控制項陣列還有各個控制項都有其圖片還有屬性，要是一一用程式碼寫下來，不僅費時，還有出現錯誤的機會也相對提升。

利用程式碼來將寫出每個控制項陣列，本是一件複雜的工作，也失去了 VB 設計的方便性，所以利用另一種方式，利用控制項的 Index，在 VB6.0 中的控制項陣列，可以利用 Index 來參考，例如：

```
Private Sub imgBtnUp_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, x As Single, y As Single)
    imgBtnUp(Index).Top = 198
End Sub
```

此段副程式用來控制 imgBtnUp 控制項陣列在滑鼠壓下(Down)時的觸發動作，其中 Index 就是用來操縱控制項陣列，Index = 1 就表示控制 imgBtnUp(1)的動作，以此類推，如此就能輕鬆的控制此陣列，但在 VB.NET 中控制項並沒有 Index 屬性，所以利用了 Tag 屬性來替代，如此就可以在 VB.NET 設計初期使用控制項陣列，而不用在程式碼中一一宣告，以下程式碼就是升級 VB 的控制項陣列應用：

原 VB 程式碼:

```
Private Sub imgBtnUp_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, x As Single, y As Single)
    Select Case Index     ‘利用 Index 來操縱各控制項
    Case 1
        labTip.Caption = "改變定時物件狀態"
    Case 2
        labTip.Caption = "修改物件啟動時段"
    Case 3
        labTip.Caption = "刪除定時物件"
    End Select
End Sub
```

其中 Index 能表示成 imgBtnUp(index)，而控制 imgBtnUp(0)、imgBtnUp(1)、imgBtnUp(2)、imgBtnUp(3) 。

升級 VB.NET 後程式碼:

```
Private Sub imgBtnUp_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles
imgBtnUp_1.MouseMove, imgBtnUp_2.MouseMove,
imgBtnUp_3.MouseMove        ‘利用 Handles 來處理各個物件
        Select Case sender.tag   ‘sender 就是被 MouseMove 操縱的物件
            Case 1
                LabTip.Text = "改變定時物件狀態"
            Case 2
                LabTip.Text = "修改物件啟動時段"
            Case 3
                LabTip.Text = "刪除定時物件"
        End Select
```

```
End Sub
```

- **升級重點1**

由以上的 VB.NET程式可看出來這撰寫事件處理程序為主的程式設計方法，使整個程式由事件處理程序組成，當某事件發生時，自然會有人負責驅動相關的事件處理程序來處理，但另一個跟VB 6.0很大的不同在於“Handles“的關鍵字，經由這關鍵字來為事件及事件處理程序來作媒,例如,經由上例可看出,程序名稱 imgBtnUp_MouseMove 在宣告後面有 Handles imgBtnUp_1.MouseMove，這才是事件處理的重點所在，imgBtnUp_1.MouseMove 代表 imgBtnUp_1物件的 MouseMove 事件,因此只要我們在“Handles“ 關鍵字後面的“ 物件名稱“ 與“ 事件名稱“ 的組合沒變，VB.NET就不會叫錯事件處理程序，由此可知,”Handles”的存在好處就是將以一個事件處理程序，處理多個事件。

- **升級重點2:**

在設計初期就要將各控制項的 Tag 屬性設定好，例如：imgBtnUp_0 物件 Tag 屬性就是 0 、 imgBtnUp_1 物件 Tag屬性 1 ，依此類推，當事件被觸發，副程式所宣告的 “ByVal sender As Object” ，sender就會反映到被觸發的物件上，此時，sender 就代表了接下來要動作的物件並包含了其所有屬性，所以我們就可以利用sender來在程式中控制所要的控制項陣列，例如要使用控制項陣列中的imgBtnUp_1有所動作，必須先將imgBtnUp_1的Tag屬性設定為 1，以

替代在VB 6.0中設計初期的 Index屬性，然後事件被觸發後，sender就代表了imgBtnUp_1，因此我們在程式中可以利用sender.Tag來控制各個控制項陣列，當sender.Tag = 1時，就是可以來控制 imgBtnUp_1的動作，跟VB 6.0中利用Index控制，可說是有異曲同工之妙，但在控制上比較起來，VB.NET可是豐富了許多。

4.10.2 升級問題二

- **在結構的初始化與控制項陣列不同處:**

在 VB 6.0 中，使用者定義型態 Type …... End Type，將會在 VB.NET 的升級精靈中，被升級到 Structure ……. End Structure ，這點升級是我們不用去擔心的，在結構中，我們可以去定義任何型態的成員，還可以定義方法與內容，至少讓使用者覺得在 VB.NET 中的結構使用，比較跟類別的使用相似，其中跟 VB6.0 的差異在於，無法建立結構的實體，並且所有的結構成員都包含 Shared 屬性，這表示結構可以存取被定義在類別或模組的共享成員，但是無法存取實體成員，因此在以下的程式碼中，可以看到結構中包含 Initial 的公用方法，這使得宣告為結構的變數，能夠被分配到記憶體空間，雖然在使用者型態方面沒有 VB6.0 來的容易使用，但這表示 VB.NET 更符合物件導向的程式語言。

VB6.0 可以在使用者定義型態裡支援固定長度的陣列宣告，但是

VB.NET 則無法詳細指定使用者型態的陣列大小，因此為了使這一限制能夠運作，必須在使用者型態中定義一個建構子，來將陣列初始化成固定長度。以下就是結構初始化範例，利用 Module1.vb 的程式碼來說明：

VB 程式碼：

```
    Type TimeControlElement
      ON_time(MAX_TIME_COUNT) As TimeElement
      OFF_time(MAX_TIME_COUNT) As TimeElement
      TimeCounts As Integer
      Types As Integer
      Index As Integer
End Type
```

由升級精靈改寫後的 VB.NET ：

```
    Structure TimeControlElement
        <VBFixedArray(MAX_TIME_COUNT)> Dim ON_time() As
TimeElement
        <VBFixedArray(MAX_TIME_COUNT)> Dim OFF_time() As
TimeElement

        Dim TimeCounts As Short '工作數量

        Dim Types As Short
        Dim Index As Short

'UPGRADE_TODO: 必須呼叫 "Initialize" 以初始化此結構的執行個

體。 按一下以取得詳細資訊:

'ms-help://MS.VSCC/commoner/redir/redirect.htm?keyword="vbup1026"'
        Public Sub Initialize()
            ReDim ON_time(MAX_TIME_COUNT)
            ReDim OFF_time(MAX_TIME_COUNT)
        End Sub
    End Structure
```

因此，經由升級精靈的改寫，我們並不用去花很多時間在結構的升級上，在升級後的VB.NET程式碼中，可看出升級精靈把VBFixedArray屬性加入結構內，並且詳細指定陣列的長度，雖然在改寫程式的過程中，並沒有發現這項升級對其他程式有所影響，但在一些書籍中寫明了這功能會將此結構傳給一個VB的檔案函式，例如Get或是Put的應用上，而檔案函式將使用此屬性來決定是否需要讀出或是寫入與陣列相關的額外標投資料。

升級精靈造成的另一個改變是在結構內插入一個 Initialize 的方法公開方法，由上面的程式碼可看出它包含了 ReDim 的呼叫，以用來結構內的固定長度陣列之初始化。

- **升級重點1**

當在改寫 ControlTimer 的程式碼中，第一個會遇到的錯誤就是NullReferenceException「Object reference not set to an instance of an object」會發生以上的錯誤，就是結構成員 tElement 並沒有包含任何陣列元素，為了初始化此陣列與其中的元素，必須先呼叫結構內的Initialize 方法，如此才可以使用結構內容。

- **升級重點2**

第二個會發生的錯誤，就是將放置初始化的程式碼位置，一般來說，將初始化的程式碼放在最先用到此結構變數的前面，例如：在

ControlImg_Click此副程式中，這是處理 ControlImg_0 -- ControlImg_6 的控制項陣列中的click事件，裡面的 tElement 已在 Module1.vb 中宣告為 TimeControlElement 的結構變數，在此副程式中可以看到 tElement 變數，因此我便將初始化的宣告放在這個副程式中，但是在編譯時，還是會出現“System.NullReferenceException為處理例外發生於時控並未將物件參考設定為物件的執行個體，試了幾次後，要將此結構變數的初始化放在時控 Load副程式中才能順利編譯，還有另一個需要注意，一開始將初始化程式碼寫成 tElenment(MaxElement).Initialize，但這樣也會發生錯誤，原以為只要像類似宣告陣列的方法一樣，在結構變數中，把MaxElement載入，如此就可以產生出相對的記憶體空間，但顯然此方法不能使用，因此在參考 MSDN 還有 debug 中的訊息後，寫出以下的程式碼：

```
For j = 0 To MaxElement - 1     '初始化
    tElement(j).Initialize()
Next
```

4.10.3 升級問題三

發生在撰寫 rule.vb 上，這是一個使用者控制項，表面上看起來並沒有那麼複雜，但在升級到 VB.NET 過程中，可是遇到一堆問題，最先遇到的問題就是容器控制項的問題，在 VB6.0 中，PictureBox 就是一個容器控制項(Container)，所謂的 Container 就是可以做為其他控

制項的容器，並且提供焦點管理，它的好處就是在移動這 Container 時，所包含的各個控制項都會一起移動，還有當此容器控制項有事件發生時，例如： MouseDown ，其它的控制項也會有所反應，因此使程式的運作上能夠達到一致性，但在VB.NET中 PictureBox 並不是一個容器控制項，但經由升級精靈的改寫，可看出 VB.NET 已將 PictureBox 這控制項改由 Panel 來替代， Panel 就是面板控制項，可以用來建立控制項的群組關係，而且移動此面板時，其中在面板中的控制項也都會跟著移動。

- **升級重點1**

在 VB.NET 升級精靈改寫後，原本在 rule.vb 中的 rule 使用者控制項卻不能運作，經過討論後，第一個想到的問題應該是原先 VB6.0 中有個控制項 — ARProgressBar 不能升級的關係，這個控制項能將所拖曳的路徑留下標記，但這一控制項不是 VB6,0 中所內建的，因此無法直接升級，但就算是上網去找，也沒有能給 VB.NET 使用的，因此在撰寫此段程式碼時，就打算將這 ARProgressBar 控制項放著不管，先處理其他控制項還有程式碼，然而在將所有的控制項設計好並且將程式碼改寫後，至少期望能看到使用者控制項 rule 能夠運作，但當時無論怎麼 debug ，就是無法找出錯誤來。

- **升級重點2**

在 VB6.0 中，當我們操縱一個容器控制項時，

圖4-21 容器控制項

如上圖，當這容器控制項 TotolTime 接收到 MouseDown 的訊號時，它會將此訊號傳給所有它包含的控制項，因此在這 rule 中的所有控制項都能夠依照程式碼中 MouseDown 的訊息來運作，但此時就會有一個問題，若是所有被包含的控制項同時動作，那麼怎判斷哪個控制項要動作或是停止，所以在程式碼中利用 AddStep 來控制它們，

VB6.0程式碼:

```
Private Sub TotalTime_MouseDown(Button As Integer, Shift As
Integer, x As Single, y As Single)
        If AddStep = 1 Or AddStep = 3 Then
                selButton(0).BorderColor = selButton(1).BorderColor
        End If
       Select Case AddStep
           Case 1:
                FirstPos = x
                AddStep = 2
              Case 3:
                SecondPos = x
                AddStep = 4
           End Select
    End If
  End Sub
```

- **升級重點3:**

在設計階段依各容器控制項中的控制項來製作，且程式碼也改寫

過，但是無論怎麼操作，rule 使用者控制項就是沒有動作，

```
Private Sub TotalTime_MouseDown(ByVal sender As Object, ByVal e
As System.Windows.Forms.MouseEventArgs) Handles
TotalTime.MouseDown
End Sub
```

以上是升級到 VB.NET 後，操縱容器控制項 MouseDown 的副程式，問題的癥結就出在 “Handles”後面所接的物件名稱，也許是因為VB.NET已經升級為一個完全以物件導向的程式語言，所以連對各物件的宣告也變的嚴格，我們要直接在“Handles”後面接上所要動作的控制項，而不是如同 VB6.0 對容器控制項的操作，

```
Private Sub TotalTime_MouseDown(ByVal sender As Object, ByVal e
As System.Windows.Forms.MouseEventArgs) Handles
Shape1.MouseDown , frmSmallTime.MouseDown
End Sub
```

由上程式碼可以看出，TotalTime 包含了 Shape1 和 frmSmallTime 等控制項，但是在 VB.NET 不能對 TotolTime 操作，而要直接操縱被包容物 Shape1 和 frmSmallTime 。

- **程式碼可再利用之程度：**40%
- **製作至完成升級所需時間： 20 小時**

4.11 objClock

- 表單名稱：objClock
- 模組檔名：時鐘.ctl
- 升級至 VB.NET 後之檔名: objClock.vb

- 升級至 VB.NET 與 VB 不同處：

VB6.0 中的 Line 和 Image 控制項在 VB.NET 中並無相對應的控制項，VB.NET 並不支援在任何表單上以任何方法或是 PictureBox 控制項進行繪圖工作，兩者都已經被 VB.NET 併入到 GDI+ (Graphics Design Interface) 裡面了，必須使用 System.Drawing.Graphics 物件進行傳遞工作，要對 Paint 事件有所回應，雖然 VB.NET 上是宣稱了繪畫功能更為強大，但這也使得程式設計上更加複雜，畢竟原本在 VB 6.0 中設計初期就可以使用的控制項，現在卻要用程式碼來表達出來，這樣增添了不少設計的困難度；在 VB.NET 中 Image 類別是一個抽象基礎類別，提供圖形存取與方法，既然抽象(Abstract)，就不能實體化或直接使用它的方法，必須在程式碼中宣告所要使用的圖檔，例如以下的程式碼：

```
Dim pic As Image
Pic = Image.FromFile("圖檔路徑")
```

雖然程式碼不多，但總是比起在 VB6.0 中設計階段就能看到圖形還有其位置，來的麻煩許多，況且在這改寫家電端系統中，有許多的圖檔，要是每個都用程式碼來撰寫，而且要等到編譯後才能看到圖檔的位置，還有其中可能的錯誤，這樣只會增加程式設計上的困難，所以改用另一個方法來替代：PictureBox，這是一個在 VB6.0 和 VB.NET 中都有的控制項，把 PictureBox 中的 Image 屬性直接設定為我們所想

要的 Image 圖檔，如此一來就可以在設計階段看到所有的圖檔，當然這麼一來，所消耗的記憶體還有程式整體執行速度更是不在話下，但以現今電腦的運算速度，這些所消耗的記憶體還有增加的執行時間，都是能被允許的。

- **升級重點**

Line 控制項的改變，升級後包含在 VB.NET 的 System.Drawing 裡面，利用基本的繪圖物件來表現出 Line 的方法—DrawLine，但在這之前，必須瞭解有哪些物件能夠表現出 Line，例如：我們繪製時要決定繪製時物體的線條顏色(Pen 物件)、粗細、效果(Brush 物件)，還有要繪製在哪個部分(Point 物件)、區域大小(Rectangle 物件)…等。

在 Clock.vb 中，在 VB6.0 的程式中，利用 Line 控制項直接來繪畫出時針、分針、秒針等 Clock 物件，也因為在 Line 的控制項中有其屬性能夠在撰寫程式碼使用，但在 VB.NET 中 Line 只能用 DrawLine 來表現，所以在這 Clock.vb 中的程式碼幾乎已經重新寫過了。

首先，在 VB6.0 設計階段，已經把時針、分針、秒針都已經設計它們的顏色、長短和粗細，因此在 VB.NET 中，要先把這些利用 Pen 物件來替代：

```
Dim SecPen As New Pen(Color.Black, 1)
Dim MinPen As New Pen(Color.DarkBlue, 3)
Dim HrPen As New Pen(Color.GreenYellow, 3)
```

使用GDI+來繪圖，可以想像我們在所指定的畫布上作畫，當然，

要作畫的第一件事，就是挑選所需要的畫筆或是筆刷，因此由上面的程式碼可看出，利用Pen類別來產生出我們所需要的畫筆，

```
Dim SecPen As New Pen(Color.Black, 1)
```

SecPen代表秒針，顏色黑色、畫筆寬度為一，同理，MinPen代表分針，顏色深藍、畫筆寬度為三，HrPen代表時針，顏色黃綠、畫筆寬度為三，把這些所要用到的畫筆設定好後，猶如在VB6.0中在設計階段把 Line 控制項加入，並且設定屬性；至於畫布，也就是要繪圖的地方，則要先建立Graphics物件，Graphics 物件代表的是 GDI+ 描繪介面，而且是用來建立圖形影像的物件，利用以下兩步驟來表現所要的圖形：

1.建立 Graphics 物件：

方法一： Dim g As Graphics

方法二： 利用Paint 事件處理常式中的 PaintEventArgs，若要從 Paint 事件中的 PaintEventArgs 取得 Graphics 物件的參考，使用以下三步驟：

Step1：宣告 Graphics 物件。

Step2：指派變數來參考當作 PaintEventArgs 一部份傳遞的 Graphics 物件

Step3：插入程式碼來繪製表單或控制項。

例如：

```
Private Sub objClock_Paint(sender As Object, pe As PaintEventArgs)
```

```
Handles MyBase.Paint
        Dim g As Graphics = pe.Graphics
    End Sub
```

2.使用 Graphics 物件來描繪線條和形狀、呈現文字或顯示和管理影像。

因此在 objClock.vb 中將在 PictureBox (Name = Pic) 上繪出各個時針，首先要將此 PictureBox 當作是畫布，以 UpdateSec() 副程式為例，

```
g = Pic.CreateGraphics
```

然後就可以利用 g 此物件來在此畫布上繪圖，

```
g.Clear(Color.Silver) '將此畫布清除並把顏色設為 Silver
g.DrawLine(SecPen, x1, y1, secx2, secy2)
g.DrawLine(SecPen, x1, y1, secx2, secy2)
'利用 SecPen 畫筆在 Point( x1, y1) 到 ( secx2, secy2) 畫線
g.DrawImage(shpHourPos.Image, Hrsx2, Hrsy2)
'將shpHourPos.Image此圖片放在 Point ( Hrsx2, Hrsy2)
```

- **程式碼可再利用之程度：**90%
- **製作至完成升級所需時間：**10 小時

4.12 frmServerMode

- 表單標題：家電端 Sever
- 表單名稱：formServerMode
- 模組檔名：frmServerMode.frm

- 升級至 VB.NET 後之檔名：formServerMode.vb
- 模組畫面：

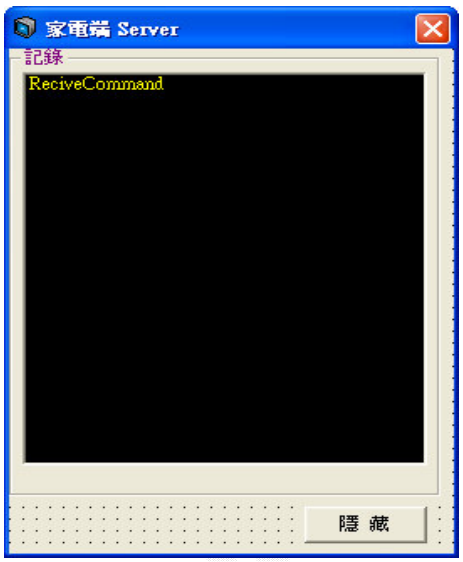


圖 4-22 “家電端 sever” 視窗

- 升級到 VB.Net 和 VB 之不同處：
  - ADAM 的控制部份就是全部都是寫在這個模組，當然 ADAM 這部份的程式碼隨著 VB 升級至 VB.NET 也有些小小變更的地方，先列出原先 VB 部分的程式碼：

```
Private Sub Comm1_OnComm()
    Select Case Comm1.CommEvent
        Case comEvCD
        Case comEvCTS
        Case comEvDSR
        Case comEvRing
        Case comEvReceive
        Case comEvSend
    End Select
End Sub
```

  - 在 VB.NET 中的程式碼：

```
Private Sub Comm1_OnComm(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
Comm1.OnComm
        Select Case Comm1.CommEvent
            Case MSCommLib.OnCommConstants.comEvCD
            Case MSCommLib.OnCommConstants.comEvCTS
            Case MSCommLib.OnCommConstants.comEvDSR
            Case MSCommLib.OnCommConstants.comEvRing
            Case
MSCommLib.OnCommConstants.comEvReceive
              Case
MSCommLib.OnCommConstants.comEvSend
        End Select
    End Sub
```

- 製作至完成升級所需時間：2 小時
- 程式碼可再利用之程度：60%

4.13 frmUser

- 表單標題：使用者資訊
- 表單名稱：frmUser
- 模組檔名：frmUser.frm
- 升級至 VB.NET 後之後檔名：frmUser.vb
- 模組畫面：

圖 4-23 “使用者資訊”視窗

- 升級到 VB.Net 和 VB 之不同處：
  - 這個模組會將新增使用者的資訊儲存至檔案中，而 VB 和 VB.NET 開、讀檔的方式也有一些不一樣的改變。
  - 在 VB 中要開啟一個檔案時，可以任意指定一個編號的檔案緩衝區使用，開啟另一個檔案時，可以隨意再開啟一個檔案編號檔案緩衝來使用，但不同的檔案在未關閉時，不得使用相同的檔案緩衝來處理，也就是說每一個檔案只能用一個檔案緩衝來存取資料，不同的檔案不可以用同一個檔案緩衝來存取資料。而開啟一個檔案的方法如下：

Open：開啟一個檔案，Open 檔名 For 存取模式 As #FreeFile .....

  - 舉個例子來說：

```
Private Sub Command1_Click()
        Open "C:\天才小管家\setup.ini" For Input As #1

        Open "C:\天才小管家\Served.dat" For Output As #2
        LenB(MyStruct)
        …
End Sub
```

存取模式	說明
Append	將資料新增至檔案的最後一筆。
Output	將資料重新寫入檔案。
Input	將資料由檔案裡讀出來。
Binary	使用二進數資料來存取檔案。

表 4-2 存取模式說明

■ 使用 Close 將開啟的檔案關閉，將檔案緩衝歸還(釋放)。

```
Private Sub Command1_Click()
        Open "C:\天才小管家\setup.ini" For Input As #1
        LenB(MyStruct)
        Close #1
End Sub
```

■ 在 VB.NET 中的改變，以相同例子來介紹，在整理出其改變，如下：

```
FileOpen(1, "C:\天才小管家\setup.ini", OpenMode.Input)
```

```
                LenB(MyStruct)
FileClose(1)
```

原 VB 語法	VB.NET 對應之改變
Open	FileOpen
Close	FileClose

表 4-3 VB 與 VB.NET 對應語法

原 VB 存取模式	VB.NET 對應之改變
Append	OpenMode. Append
Output	OpenMode. Output
Input	OpenMode.Input
Binary	OpenMode. Binary

表 4-4 VB 與 VB.NET 對應存取模式

- 另外，VB.NET 的檔案開、讀的功能也更加強大，完整介紹如下：

Opens a file for input or output.

```
Public Sub FileOpen( _
    ByVal FileNumber As Integer, _
    ByVal FileName As String, _
    ByVal Mode As OpenMode, _
    Optional ByVal Access As OpenAccess=OpenAccess.Default
Optional ByVal Share As OpenShare = OpenShare.Default,
Optional ByVal RecordLength As Integer = -1
```

- 製作至完成升級所需時間：2 小時

- 程式碼可再利用之程度：70%

4.14 frmChangePlug

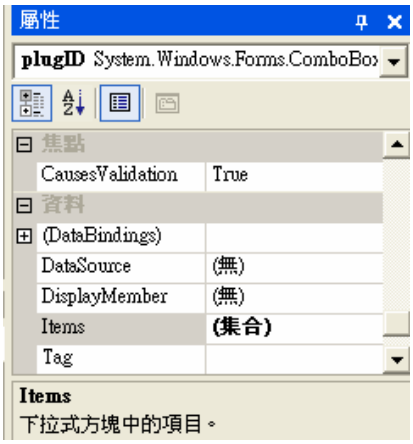
- 表單標題：變更插座編號
- 表單名稱：frmChangePlug
- 模組檔名：frmChangePlug.frm
- 升級至 VB.Net 後之檔名：frmChangePlug.vb
- 模組畫面：

圖 4-24 “變更插座編號”視窗

- 升級到 VB.NET 和 VB 之不同處：
  - 這個模組中有一個 ListBox 裡面包含五個項目（插座 1～5）在原來的 VB 原始程式碼是利用迴圈去建立這些項目

```
For i = 1 To 5
        plugID.AddItem "插座  " & Str$(i)
Next i
```

  - 但在 VB.NET 可以直接在設計階段輸入要有幾個項目，下面利用實際的畫面來介紹：



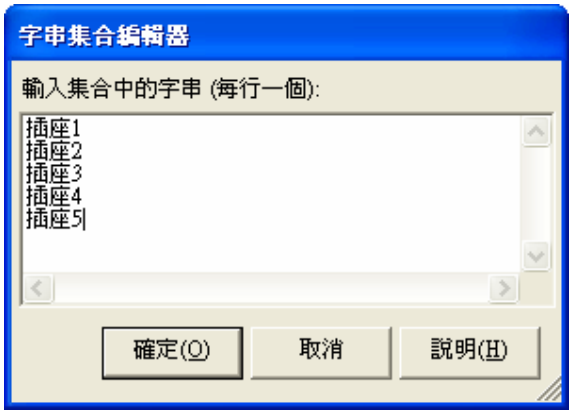


圖 4-25 屬性畫面　　**圖 4-26 字串集合編輯器**

- 先選取 ListBox 控制項然後檢視其屬性欄，按一下 Items 一欄，會出現一個”字串集合編輯器”，此時把希望顯現出的項目填入即可。

- 製作至完成升級所需時間：1 小時
- 程式碼可再利用之程度：0%

4.15 frmAddSwitch

- 表單標題：設定虛擬開關
- 表單名稱：frmAddSwitch
- 模組檔名：frmAddSwitch.frm
- 升級至 VB.NET 後之檔名：frmAddSwitch.vb
- 模組畫面：

圖 4-27 “設定虛擬開關”視窗

- 升級到 VB.NET 和 VB 之不同處：
  - 通常在 ListBox、TextBox、Label…等控制項中的屬性欄中通常都有著一個屬性 ForeColor 用來顯示控制項文字和圖片的前景顏色，如果要設定它的顏色，在 VB 中用下列的方法可以改變顏色：

```
ObjStatue.ForeColor = &H8000&
```

- H8000 在 VB.NET 是屬於 Integer 值，而 VB.NET 中 ForeColor 中所接受的顏色設定值必須是 Color 值才行，但可以用下列的方法接受 H8000：

```
ObjStatue.ForeColor =
System.Drawing.ColorTranslator.FromOle(&H8000)
```

- 製作至完成升級所需時間：2 小時
- 程式碼可再利用之程度：80%

第五章 結論

在本章將為整個專題做總結，包括系統升級總整理、系統 REUSE 比例、專題實作心得以及實作所遭遇的困難與解決方式。

5.1 系統可再用比例

系統項目	可再用比例
各部分面板	55%
Adam 控制程式	100%
網路程式	70%
電器控制功能	70%
虛擬感應開關功能	30%
錄影管理功能	60%
定時功能	50%
登錄使用者管理	50%
智慧型保全系統	30%

表 5-1 各模組可再用程式碼整理

以上為我們整個系統升級，可再用的程式碼的整理，最後整個系統的可在用程度約為 57%。

5.2 系統升級結論

我們在升級的過程中發現，升級並非想像中的容易，有下列 3 點

原因：

一、 升級精靈僅可升級約 57%，並非能夠升級到 95%；進一步的手動驗證、評估、更新、除錯以及測試都需要花費許多時間。

二、 VB.NET 完全物向導向化，架構在 CLR 平台上，且包含龐大的函式類別庫，要充分掌握 VB.NET，必須對.NET 執行機制以及每一個物件的方法屬性及參數資訊充分了解。

三、 對於 VB.NET 不支援的方法，必須找出替代的方式，來達到相同功能，由於是蠻新的技術，翻閱書籍或上網請教都很難得到解決的方式，大多是靠我們自己摸索以及參考微軟 MSDN 技術文件，才想出解決的方式。

升級實作後發現，系統轉換到 VB.NET 所需付出的成本取決於下列兩點：

一、 轉換精靈的設計：像在 VB.NET 2002 無法升級圖片控制項，2003 版本已可升級，若對於不支援的部分升級精靈都能自動升級，提供替代的方式，那使用升級精靈可升級的比例將可大大的提高。

二、 程式碼的品質：原系統程式碼的設計對系統的升級是很重要的，若原系統對在.NET 不支援的部分使用較少，例如不支援表單的圖形方法或形狀和直線控制項。不使用隱含性宣告，程式

碼符合標準 VB6 的語法，那使用升級精靈升級的比例也可大大的提升。

5.3 遭遇困難與解決方法

硬體的錯誤:由於我們題目為系統的升級，所以一開始老師要求我們，要熟悉系統各步驟的操作，由於是第一次接觸，硬體與軟體的安裝就花了我們不少時間去熟悉，在實際操作時，有發生無法動作的情形，原本以為是我們在硬體或軟體的安裝上出現問題，經過多次請教學姊與學長與確認之後，發現我們在安裝與操作上並無錯誤，軟體的部分是沒有問題的，所以我們猜測應該是 Adam 發生錯誤，因為學姊之前在使用時，Adam 模組就常出現問題，後來在經過仔細電路的測試之後，發現原來是用了錯誤的變壓器，才導致無法動作，只是小小的一個錯誤，就讓我們折騰了許久。

系統升級:在實際著手升級時，也有很多難處，最大的困難在於我們面對的是兩種不熟悉的語言－VB 與 VB.NET，原本以為 VB.NET 的升級精靈應該可以順利的升級，但是發現並非如此，VB 與 VB.NET 語法差異非常大，VB.NET 已經變成物件導向的程式語言，用法也迥然不同。

在升級過程中，我們發現很多改變，在很多書本中都未寫明，必須靠我們自己去摸索，編譯時程式常常出現一堆錯誤，要發現這錯誤，

還必須花了一堆時間來試驗，但程式語言不就是在最細微的地方往往最容易去忽略，因此實際將 VB 6.0改寫成 VB.NET 才能發現兩者真的有許多地方的不同，往往坊間有許多人將VB.NET稱作是 VB 7.0 ，顯然是對VB.NET沒有深入的研究，表面上看起來差不多的產品，但是程式語言的精神已經不一樣了。

5.4 系統未來展望

此系統是使用 Network Camera，未來在視訊的部分，使用支援一般大樓監視器的系統，可降低所花費的攝影機成本，並多處放置攝影機來達到完全監控，使其能落實無死角的監控理念。

在 Adam 控制模組部份，目前只能控制簡單電器，在增加電器各數部分較有限；未來可改用 8051 單晶片來控制，可操控較高檔位的家電，在增加電器個數部分也較為容易 。

在遠端操作部分，未來可使用 Windows CE.NET 改寫系統，由於同樣是.NET 架構下的，不須改變太多程式碼；讓使用者可隨時透過小型行動裝置由遠端操控本系統，監視家中畫面，讓本系統的即時性更為提升。

5.5 專題實作心得

從專題實作上，使我們體會到在升級一專案上的困難，原以為都是 Microsoft Visual Basic 的產品，利用升級精靈就可以達到順利升級

的結果，沒想到單單在升級各個使用者控制項上，就深刻體會到 VB 6.0 和 VB.NET 上的差別，尤其是程式碼上可以瞭解到 VB.NET 已經改變成一個物件導向的程式語言，而必須瞭解先前的 VB6.0 程式碼的設計含意後，才能改寫成 VB.NET ，不僅如此，遇到比較大的困擾就是 VB.NET 對許多人還是很陌生，在改寫的過程中，就算遇到了問題，上網請教對這方面有研究的先進，但還是無法得到一個正確的結果，因此在 VB.NET 的摸索時間就增加了許多，但也讓我們印象深刻，尤其是遇到一些參考資料無法解決辦法的時候，往往自己想個好幾天是很正常的，所以讓我們在寫專題報告時能夠就遇到的問題，寫上個好幾頁的研究心得，這是由我們專題實作中，才能有所學習到的，由於.NET 整個架構非常龐大，我們不敢說對.NET 完全透徹的了解，但讓我們對 VB.NET 已有更近一步的認識，我們寫程式的功力也大大的邁進了一大步。

從大三下開始，歷經了一年的專題實作，從初步的找專題老師、決定題目、專題實作、報告編寫以及專題發表，都讓我們從中學習了許多課堂上學習不到經驗。一開始由於我們對 Visual Basic 語言不熟悉，讓一開始的進度落後，不過大家都能互相幫忙，遇到困難時大家都會互相討論，組員間都能互相體諒，最後，很感謝竇老師的引導以及學長們的指導，讓我們專題可以順利的完成。

參考資料

[1]林建仁， Visual Basic .NET 新世代高手之基礎篇，pp.180 –319 ，碁峰資訊，February 2003

[2]正序工作室， Visual Basic .NET 程式設計典範，pp. 4-2 – 13-44，金禾資訊，August 2002

[3]陳清木， Visual Basic .NET 程式設計入門與實務，pp.78 – 320，文魁資訊，May 2002

[4]Ed Robinson & Michael Bond & Robert lan Oliver，Upgrading MicroSoft Visual Basic 6.0 to MicroSoft Visual Basic .NET， pp.2-2 – 13-24，文魁資訊，October 2002

[5]柯溫釗， Visual Basic .NET 學習經典，pp.45 – 210，知城數位科技，2002

[6]洪國勝， Visual Basic 程式設計實務與應用，松岡電腦圖書資訊，1999

[7]林建仁， Visual Basic 新世代高手，碁峰資訊，1998

[8]廖榮貴研究室， Visual Basic 程式設計入門與實務，文魁資訊，2002

附錄 A－系統安裝說明

A.1 硬體需求

(一)、586 以上, 64MB ram, 硬碟 1GB 以上, 100MB 網路卡。

(二)、AXIS 2100 Network Camera。

(三)、電源控制裝置(含 Adam4522 X 1、Adam4050 X 1、Solid State Relay X 5)。

A.2 作業平台

Windows XP/2000/Me/98/NT

A.3 硬體安裝程序

硬體安裝如圖 A-1 所示，其詳細安裝步驟如下(請對照圖 A-1 中的標示)：

步驟 1：將網路攝影機的網路線接至主機的 RJ-45 連接埠。

步驟 2：接上網路攝影機的電源。

步驟 3：將電源控制器的 RS-232 接頭接至主機的 RS-232 串列埠。

步驟 4：接上電源控制器的電源。

步驟 5：將電器插頭插在電源控制器的插座上。

其中，電源控制器的插座上方皆有編號如圖 A-2 所示。軟體在設定時

會要求使用者輸入電器所安裝的插座編號，因此當使用者將電器安裝於電源控制器時，請務必記住電器所安裝的插座編號。

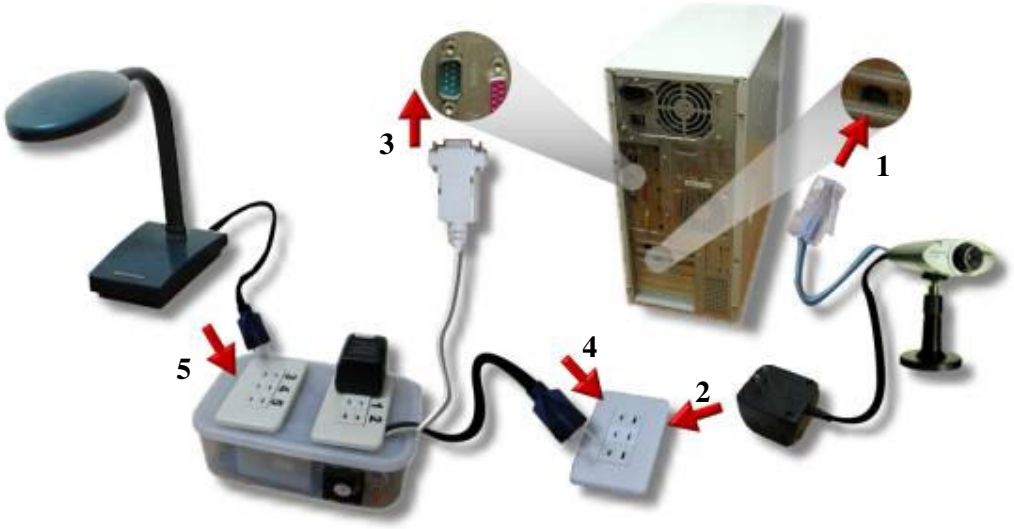


圖 A-1 硬體安裝說明圖

A.4 啟動電源控制器

將電源控制器右側的開關切到的 ON 位置如圖 A-3 所示，即可啟動電源控制器。

圖 A-2 電源控制器上方的插座編號

圖 A-3 電源控制器開關的位置

A.5 軟體安裝程序

步驟 1：安裝.NET Framework

（在未來，Windows 作業系統內建此步驟可省略）

步驟 2：遠端家電控制系統安裝

步驟 3：在 C 槽新增兩個資料夾－天才小管家與天才小管家 Sever

步驟 4：啟動遠端家電控制系統即可使用

系統操作說明，請見附錄 B

附錄 B－系統操作說明

B.1 主操作畫面

遠端家電控制與智慧型保全系統的主操作畫面如圖 B-1 所示。操作介面中的各項功能說明請參照標示編號。

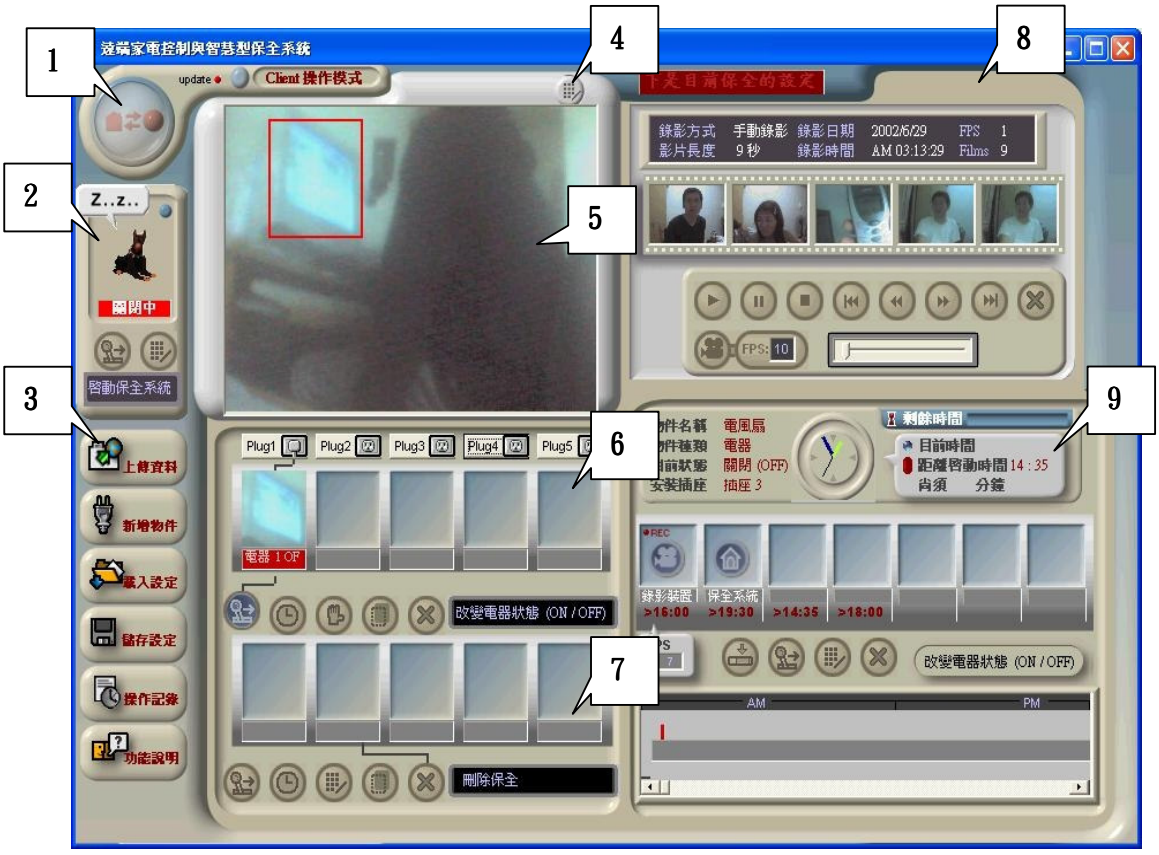


圖 B-1 系統主操作畫面

1. 切換操作模式：可將系統切換至 Server 和 Client 兩種操作模式，分別使用於家電端及遠端。

2. 保全系統：顯示目前屋內監控狀態，並可設定警報裝置。

3. 主工具列：包含了建立各項物件、儲存載入設定、操作記錄以及功能說明等功能操作。

4. 操作功能切換鈕：切換操作介面為複雜操作功能或簡易操作功能 。如圖 B-2 所示。

5. 攝影畫面：透過網路攝影機顯示目前屋內狀態。

6. 電器控制面板：提供使用者有效地操作已設定的電器物件。

7. 虛擬感應開關面板：提供使用者在屋內的特定區域以手部揮動的感應方式來啟動電器、錄影裝置或保全系統。

8. 錄影控制面板：可以自動儲存與管理系統中錄影功能所錄的影片 。

9. 定時控制面板：提供使用者在指定的時段內啟動指定的物件，包含家電、虛擬感應開關、錄影裝置以及保全系統。

圖 B-2 簡易操作功能畫面

B.2 主工具列

資料上傳：使用者可在遠端設定系統中的各項功能，並透過此工具鈕將設定好資料上傳至家中的主機並加以更新。(此功能用於 Client 操作模式中)

登錄管理：此系統主要是對登入的使用者加以管理，並設定登入者的操作權限，保護系統免於被有心人士利用。(此功能用於 Server 操作模式中)

新增物件：系統中包含了電器、虛擬開關、錄影裝置、定時裝置和保全系統共五種控制物件，其中電器和虛擬開關兩種物件，可利用此工具鈕直接從攝影畫面中選取物件所有位置來產生。其詳細的操作步驟在 B.3 節會詳細介紹。

載入設定：由工作路徑中載入指定的設定檔至系統中的各項功能設定。

儲存設定：將目前系統中所有的功能設定儲存成設定檔於工作路徑中。

操作記錄：記錄系統中各項控制物件包括電器、虛擬開關、錄影裝置、定時裝置以及保全系統的啟動關閉時間。(Server 操作模式中還包括記錄遠端使用者的帳號、登錄時間和來源 IP 等相關登錄資料)

功能說明：根據系統中各項功能包括遙控電器、虛擬開關控制、錄影裝置、定時裝置以及保全系統的操作步驟加以說明。

B.3 系統操作模式

本系統包含了 Server 和 Client 兩種操作模式，分別使用於家電端及遠端。切換鈕位於圖 B-1 中標示 1 的位置，使用者在家中必須使用 Server 操作模式與電源控制器連線；在遠端時則將必須使用 Client 操作模式與家中主機連線。

B.3.1 Server 操作模式

當系統切換至 Server 操作模式時，便可以接收遠端使用者所送的控制指令，再依控制指令啟動或關閉家中的電器。同時 Server 操作模式透過網路攝影機所輸入的影像對屋內作監控的動作。另外，在 Server 操作模式下可使用虛擬感應開關功能，提供使用者在屋內的特定區域以手部揮動的感應方式來啟動電器、錄影裝置或保全系統。欲將系統切換至 Server 操作模式，可直接按下主操作畫面左上方的切換鈕，此時切換鈕中會顯示 Server 操作模式圖示如右圖所示。切換至 Server 操作模式後，系統會自動和電源控制連線，若未連線則會出現警告訊息視窗。另外在 Server 操作模式下的主工具列會出現登錄管理鈕 ，用來管理所有登錄的

使用者，詳細說明會在 B.8 節中。

B.3.2 Client 操作模式

當系統切換至 Client 操作模式時，使用者可在遠端遙控家中的電器，並可透過網路攝影機所傳來的影像檢視屋內電器的啟動情形。另外 ，由於網路攝影機內建了一個 web server，因此即使屋內的主機未開啟也可以在遠端透過 Client 操作模式執行保全監控。欲將系統切換至 Client 操作模式，可直接按下主操作畫面左上方的切換鈕，此時切換鈕中會顯示 Client 操作模式圖示如右圖所示。切換至 Client 操作模式後，系統會自動出現連線視窗如圖 B-3 所示。使用者只要輸入帳號、密碼和 Server 端的位址，即可連線至 Server 端。

圖 B-3 連線視窗

B.4 電器控制

當電器安裝於電源控制裝置，使用者尚須在軟體上設定該電器在攝影畫面中的位置以及安裝的插座編號等資訊，才可以透過軟體由家電端或遠端控制該電器。首先，使用者必須新增控制電器至電器控制面板，接下來可以透過電器控制面板或直接由攝影畫面控制安裝的電器，詳細操作說明如下。

B.4.1 新增控制電器

步驟 1：按下主工具列中的新增物件鈕。

步驟 2：將游標移至攝影畫面，此時游標會變成十字形。接著按下滑鼠左鍵不放拖曳，選取該電器在影像上的位置如圖 B-4 所示 (拖曳時若按下滑鼠右鍵可取消選取)。

步驟 3：當選取完成放開滑鼠左鍵，即出現新增物件視窗如圖 B-5 所示。

步驟 4：於物件名稱輸入框中輸入電器名稱。

步驟 5：於物件種類下拉式清單中選取控制家電。

步驟 6:於安裝插座下拉式清單中選取該家電所安裝的插座編號(若選擇的插座已被其他電器所使用，則會顯示警告視窗如圖 B-6 所示)。

步驟 7：按下確定鈕，電器即新增至電器控制面板上。

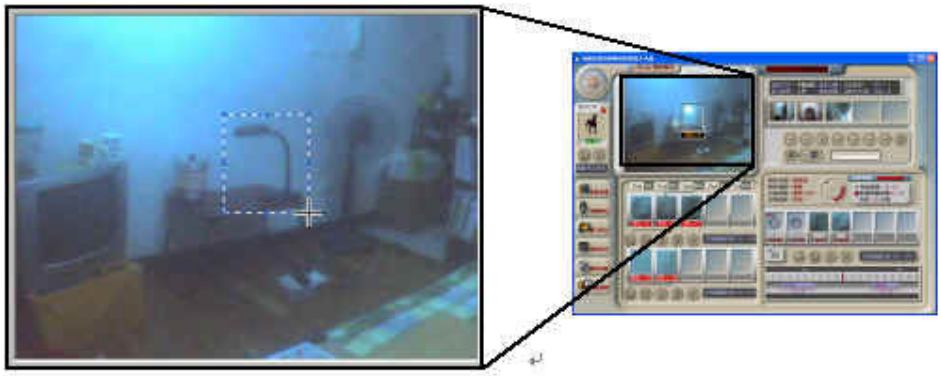


圖 A-4 使用滑鼠拖曳選取電器於攝影畫面

圖 B-5 新增物件視窗

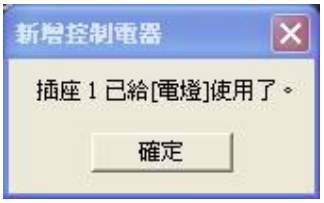


圖 B-6 插座重複使用警告視窗

B.4.2 使用電器控制面板

電器控制面板如圖 B-7 所示，其主要功能是提供使用者有效地操作已設定的電器物件。面板中會顯示所有安裝的電器圖示以及該電器的名稱與目前狀態，當滑鼠游標移至圖示上方時會自動顯示該電器所安裝的插座編號。電器控制面板中的功能鈕分別表示如下：

改變電器狀態 (ON / OFF)

將電器加入至定時設定

電器可由虛擬感應開關控制

重新設定電器所在位置

 從電器控制面板中刪除電器

欲修改電器在電器控制面板上所設定的插座編號，使用者可直接按下電器所安裝的**插座編號鈕**，出現變更插座編號視窗如圖 B-8 此時在變更插座編號下拉式清單中選取重新設定的插座編號，若該插座編號已被其他電器使用，則這兩個插座編號會直接互換。

另外，使用者也可以直接從攝影畫面中改變電器的狀態，只要將滑鼠游標移至畫面中欲控制的電器上，該電器周圍便浮現成按鈕狀如圖 A-9a 所示，按鈕下方會標示電器名稱即目前狀態，此時在電器鈕上按下滑鼠左鍵即可改變該電器的狀態如圖 B-9b 所示。

圖 B-7 電器控制面板

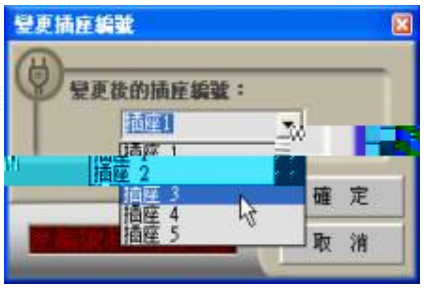


圖 B-8 直接從攝影畫面中改變電器狀態

圖 B-9a 游標移至電器上
電器周圍浮現成按鈕

圖 B-9b 直接按下電器按鈕即改變

B.5 虛擬感應開關功能

虛擬感應開關主要是提供使用者在屋內的特定區域以手部揮動的感應方式來改變單一或多項物件的狀態，包含電器、錄影裝置或保全裝置，如圖 B-10 所示。使用者可以利用主工具列中的新增物件功能產生虛擬感應開關，並使用虛擬感應開關功能中的設定功能來編輯欲啟動的物件，完成之後使用者可在屋內所設定的區域中以手部來回揮動一次改變連結物件的狀態。另外，虛擬感應開關可以直接設在電器物件上，使用者可以直接在電器前以手部來回揮動一次改變該電器的狀態。

圖 B-10 以手部揮動的感應方式來改變電器的狀態

B.5.1 新增虛擬感應開關

步驟 1：按下主工具列中的新增物件鈕。

步驟 2：將游標移至攝影畫面，此時游標會變成十字形。接著按下滑鼠左鍵不放拖曳，選取該虛擬感應開關在影像上的位置 (拖曳時若按

下滑鼠右鍵可取消選取)。

步驟 3：當選取完成放開滑鼠左鍵，即出現新增物件視窗如圖 B-5 所示。

步驟 4：於物件名稱輸入框中輸入虛擬感應開關的名稱。

步驟 5：於物件種類下拉式清單中選取虛擬感應開關 。

步驟 6：按下確定扭，該虛擬感應開關即新增至虛擬感應開關控制面板上。

B.5.2 使用虛擬感應開關控制面板

虛擬感應開關面板如圖 B-10 所示,其主要功能是提供使用者有效地操作已設定的虛擬感應開關物件 。面板中會顯示所有已設定的虛擬感應開關圖示以及該虛擬感應開關的名稱與目前狀態。一個虛擬感應開關可以控制改變單項或多項物件的狀態 ，如電器、錄影功能或保全功能。當滑鼠游標移至圖示上方時會自動顯示該虛擬感應開關所連結的電器物件如圖 B-13 所示。而虛擬感應開關控制面板中的功能鈕分別表示如下：

 改變虛擬感應開關狀態

 將虛擬開關設為定時啟動

 設定虛擬感應開關的控制物件

 重新設定感應位置

 刪除虛擬感應開關

使用者可利用電器控制面板中的 功能鈕將電器物件同時設成虛擬感應開關物件，此時該電器物件會新增至虛擬感應開關控制面板上，因此該虛擬感應開關所感應的區域即該電器在影像上的區域。使用者可以直接在電器前以手部揮動改變該電器的狀態如圖 B-11 所示。

圖 B-11 直接在電器前以手部揮動改變該電器的狀態

B.5.3 設定虛擬感應開關的控制物件

一個虛擬感應開關可以同時啟動單一或多項物件的狀態，包含電器、錄影裝置或保全裝置，因此使用者可使用虛擬感應開關控制面板中的 功能鈕啟動虛擬感應開關設定視窗如圖 B-12 所示，來設定欲連結的控制物件。設定視窗左方會顯示現存物件項目，使用者可按 鈕加入至該開關所啟動的物件項目，或按 鈕從啟動物件項目中移除。當滑鼠游標移至面板中已設定的虛擬感應開關圖示，會自

動顯示所有啟動電器的連結如圖 B-13 所示。

圖 B-12 虛擬感應開關設定視窗

圖 B-13 虛擬感應開關設定視窗

B.6 錄影管理功能

此功能主要是可以自動儲存與管理系統中錄影功能所錄的影片，使用者可自行設定影片的儲存量，最高可達 10 FPS (Frames Per Second)。錄影管理功能除了會自動儲存影片，並提供完整的播放功能，讓使用者可以很方便地檢視所有記錄的影片。錄影管理功能所錄的影片可分為手動錄影、保全錄影和定時錄影三種不同的錄影方式分別表示如下：

手動錄影：使用者可自行用錄影管理功能中的  鈕錄影。按鈕中含有一個輸入框可以讓使用者設定影片的 FPS。

定時錄影：在使用者設定的時間內錄影，相關操作會在 B.8 節加以說

明。

保全錄影：此錄影方式和前面兩者最大的不同在於保全錄影是當保全系統偵測到可疑物體時所自動錄下的影片，非使用者所控制。

B.6.1 使用錄影管理面板

錄影控制面板如圖 B-14 所示。已錄過的影片會以縮圖的方式將第一格的畫面顯示於面板上。當用滑鼠點選影片圖示時，面板上方的影片資訊會顯示該部影片的錄影方式、影片長度、錄影日期、錄影時間、影片的格數和 FPS。錄影時，攝影畫面的左上角會出現 ●REC 閃爍表示正在錄影，錄完後會自動將影片圖示顯示於面板上。另外在面板下方的功能鈕可用來檢視影片圖示中的影片，其功能分別表示如下：

 播放影片

 暫停播放影片

 停止播放影片

 移至影片開端

 影片倒轉

 影片快轉

 移至影片尾端

 刪除影片

手動錄影

 影片播放位置

圖 B-14 錄影管理面板

B.7 智慧型保全系統說明

使用者可利用保全功能對屋內作有效的監控，保障屋內安全。系統啟動時會自動調整與設定保全裝置，使用者不需作任何設定即可有效的偵測出所有可疑物體並加以分析與追蹤。當系統判斷出偵測到的可疑物體為竊賊時，會自動啟動錄影功能，將整個行竊過程紀錄在家電端與遠端的電腦中，並同時對屋主的行動電話發出手機簡訊，顯示屋內遭入侵的警告訊息。

B.7.1 保全控制面板

使用者可利用面板中的 鈕啟動保全系統以及 鈕設定警報裝置。其中，面板裡的看門狗會以動畫的方式顯示目前監控狀態。當保全系統啟動時，看門狗會左右環顧如圖 B-15a 所示，表示正在監

控屋內環境。當發現竊賊時，看門狗便會開始吠如圖 B-15b 所示。當保全系統關閉時，看門狗會自行趴下休息如圖 B-15c 所示。

監控保全功能啟動後，在監控過程中當系統偵測到屋內遭入侵，會自動警報裝置。使用者可以使用  鈕開啟警報裝置設定視窗如圖 A-17 所示，視窗中包含了**警報設定**、**分析輸入影像**和**背景狀態圖**三個部分。

圖 B-15a

圖 B-15b

圖 B-15c

B.7.2 警報設定

當警報被觸發時,保全功能有三種反應方式供使用者選擇來達到警報功用，此三個選項的表示如下：

(1)傳送警告簡訊至屋主手機：

勾選此項目當後警報被觸發時，系統會自動依使用者所輸入 E-Mail 將簡訊內容傳送到屋主的行動電話中(行動電話必須可以接收電子郵

件)。

(2)啟動錄影裝置：

勾選此項目當後警報被觸發時會自動啟動錄影裝置，根據使用者所設定的 FPS 值錄下竊賊的犯罪過程。

(3)啟動警報聲：

勾選此項目之後，當後警報被觸發時，系統會自動播放使用者預設選定的音效檔來嚇阻竊賊。

B.7.3 分析輸入影像

假設保全系統啟動時如圖 B-16a 所示，當有人走進畫面中時如圖 A-16b 所示，影像分析畫面會顯示出偵測到的物體如圖 B-17 所示，並加以分析追蹤到的物體。影像分析畫面下方有兩個選項分別表示如下：

(1) 顯示分析曲線：勾選此項目則會顯示追蹤物體的分析曲線，系統會自行由分析曲線判斷偵測到的物體是否為關節物體還是外頭天色變化所造成的光影變化。

(2) 顯示入侵物位置：勾選此項目則會顯示目前系統所鎖定的偵測物

體，並擷取出追蹤物體的所在位置。

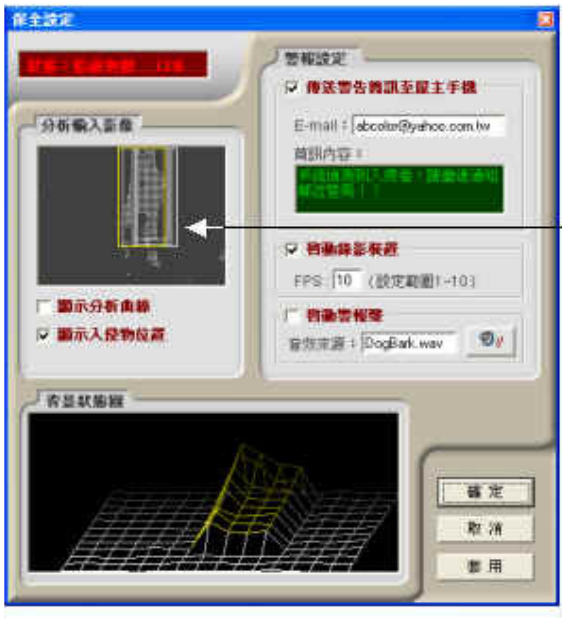


圖 B-17 警報設定視窗

B.8 定時功能

定時功能如圖 B-18 所示，主要是提供使用者在指定的時段內啟動指定的物件，包含家電、虛擬感應開關、錄影裝置以及保全系統。使用者首先把要定時啟動的物件加入定時控制面板中。接下來，使用者可用定時控制面板中的功能鈕，設定物件的啟動時段。

圖 B-18 定時控制面板

新增定時控制物件：電器控制面板和虛擬感應開關面板中均包含了一個 功能鈕，使用者可透過它將電器物件和虛擬感應開關物件加入定時控制面板中，此時該物件也同時成為定時控制物件，可以在指

定的時段中執行。另外，錄影裝置和保全系統本身也是定時控制物件之一，使用者可直接設定啟動時段。

使用定時控制面板：定時控制面板中包含了定時物件資訊，定時物件圖示和物件時間軸，如圖 B-18 所示和當滑鼠游標移至定時物件圖示時，會自動顯示該物件的資訊以及在時間軸上的啟動時間。其中物件資訊中除了顯示物件名稱、種類以及狀態外，還會顯示目前時間、下次啟動時間，以及距離下次被啟動所剩下的時間。另外在物件資訊的右上方則顯示該定時物件在工作佇列中的位置，此功能會在下一節詳細說明。定時控制面板中含有四個功能鈕，分別表示如下：

 新增物件啟動時段

 改變定時物件狀態（啟動或關閉）

 修改物件啟動時段

 刪除定時物件

時間軸與剩餘時間：時間軸主要是用來顯示定時物件的啟動時段，時間軸上所標示的是一天的時間，由 00 時至 24 時。在圖 B-18 中，當游標移至電風扇圖示時，時間軸所顯示電風扇的啟動時段分別在 3 點 25 分時啟動 233 分鐘，以及在 14 點 35 分時啟動 185 分鐘。另外，面板中的剩餘時間顯示目前物件距離被啟動所剩餘的時間，若該物件正被啟動則會顯示物件距離被關閉所剩餘的時間

新增物件啟動時段：假設現在要為電風扇新增一個啟動時段，由 14 點 35 分開啟家電至 17 點 40 分關閉電器，則設定步驟如下：

步驟 1：游標移至電風扇圖示上方，此時圖示周圍會以紅框圍住。

步驟 2：按下 鈕。

步驟 3：當游標移至時間軸時會出現粗調鈕和粗調鈕所在的時間 。

(粗調鈕會隨著滑鼠游標左右移動，移動一格表示移動 15 分鐘，因此它顯示每個鐘頭的 00 分、15 分、30 分、45 分。)

步驟 4：將粗調鈕移至 PM 2 : 30 的位置。

步驟 5：按下粗調鈕不放，出現微調滑桿如右圖所示。

步驟 6：此時移動一格表示移動一分鐘，可由粗調鈕所在的時間往前往後各微調 15 分鐘。

步驟 7：微調至 PM 2 : 35 時放開滑鼠左鍵，即設定啟動時間為 14 點 35 分。

步驟 8：往右移動滑鼠將粗調鈕移至接近關閉時間 PM 5：45。

步驟 9：按下粗調鈕並利用微調滑桿將時間微調至 PM 5：40。

步驟 10：放開滑鼠左鍵則啟動時段會顯示於時間軸上如圖所示 ，

啟動時段會顯示啟動時間和啟動時間長度 。

設定完成電風扇將會在 14 點 35 分啟動，17 點 40 關閉，長 185 分鐘。

A.9 登錄使用者管理

近年來隨著網路快速發展與普及，使得網路安全更顯得重要。遠端家電控制與智慧型保全系統由於可以透過網路來控制家中電器並取得屋中影像，因此系統必須多一層保護對登入的使用者加以管理，防止被有心人士利用。使用者管理介面如圖 B-19 所示，提供使用者在 server 模式下，對所有登入的使用者會給予不同的使用權限，並檢查使用者的來源 IP 位址是否合法。按下主工具列中的登錄管理鈕 ，即可進入登錄使用者管理視窗。

新增登入使用者

步驟 1：按下使用者清單下方的 鈕，即出現新增使用者的對話視窗，如圖 B-20 所示。

步驟 2：輸入新增的使用者帳號和密碼。

步驟 3：設定操作權限，包含 4 個核選方塊(可複選)

(1) 新增物件：選取此選項，則該使用者可對系統新增各種物件。

(2) 控制物件：選取此選項，則該使用者可控制系統中所有的物件。

(3) 修改物件：選取此選項，則該使用者可對系統中物件任意修改。

(4) 刪除物件：選取此選項，則該使用者可刪除系統中的物件。

步驟 4：設定限制來源 IP 規則。

步驟 5：按下確定鍵即完成新增使用者，並將使用者帳號加入使用者管理介面中的使用者帳號清單中。

圖 B-19．使用者管理介面

圖 B-20．新增使用者的對話視窗

修改登入使用者資料：可直接點選使用者帳號清單中的使用者帳號，檢視該使用者的各項設定內容。修改完欲修改的項目後按下 套用 即可完成修改。

刪除登入使用者

步驟 1：在使用者管理介面中的使用者帳號清單中選取欲刪除的使用者帳號。

步驟 2：按下 鈕，即可刪除此使用者。

限制來源 IP：限制來源 IP 主要是用來限制使用者只能由特定的網域登錄，避免有心人士利用他人的帳號密碼登錄 server 端。透過**限制來源 IP 規則**，可以設定使用者由指定網域登錄或限制由指定網域登錄。新增限制來源 IP 規則步驟如下：

步驟 1：按下 新增限制來源 IP 規則 按扭，即出現新增限制來源 IP 規則視窗如圖 B-21 所示。

步驟 2：選擇限制方式，有以下兩種方式：

(1)允許進入規則：所設定的來源網域可允許登錄 server 端。

(2)限制進入規則：所設定的來源網域被禁止登錄 server 端。

步驟 3：在輸入框中輸入限制來源 IP (字元'*'表示 0 至 255，假設 IP 設為 140.134.26.* 則表示的網域涵蓋 140.134.26.0 至 140.134.26.255)。

步驟 4：確認內容後按下 取 消 即可新增限制來源 IP 規則，否則按 取 消 取消新增限制來源 IP 規則。

圖 B-21. 新增限制來源 IP 規則

附錄 C－ADAM

ADAM-4522

General / isolate RS-422 / 485 Repeater

Isolated / General RS-232 to 422 / 845 converter

Specifications

- **Power requirement：** Unregulated +10~+30 V_{DC} . Module protected from power reversals
- **Case：**ABS with captive mounting hardware
- **Accessorise (supplied)：** ABS DIN- rail mounting adapter, SECC panel mounting bracker
- **Plug- in screw terminal wiring：**Accepts AWG1-#14~#22(0.5 to 2.5 mm^2)wires
- **Operating temperature：** -10 to 70°C (14 to 158°F)
- **Dimensions：** 60mm*120mm(2.36”*4.41”)
- **Transmission speed (bps)：**1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K, RTP control and RS-4522 mode (switchable)
- **Isolation voltage：**3000V_{DC}(ADAM-4522 only)
- **RS-232 interface connector：**Female DB-9
- **RS422/RS-485 interface connector：** Plug-in screw terminal
- **Power consumption：**1.2W

Features

- Automatic internal RS-485 bus supervision
- No external flow control signals required for RS-485
- 3000VDC isolation protection(ADAM-4520 only)
- Transient suppression on RS-485 data lines
- Speed up to 115.2 kbps
- Networking up to 4000 feet
- Reserved space for termination resistors
- Power and data flow indicator for troubleshooting
- Power requirement： +10 to+30V_{DC}
- Mounts easily on a DIN-rail or panel

ADAM-4050

Digital I / O Module

Specifications

Digital Input

- **Channels：7**
  Logic level 0：+1 V max.
  Logic level 1：+3.5 V ~ +30 V
- **Pull up current Logic：**0.5 mA, 10 K resistor to +5 V

Digital Output

- **Channels：**8
  Open collector to 30 V , 30 mA max. load
- **Power dissipation：**300 mW

Watchdog Timer

- **Built - in：**

Power

- **Power Requirements：**+10~+30 VDC(non-regulated)
- **Power consumption：**0.4 W