藍芽無線電話系統與服務搜尋協定實作

指導老師：侯廷昭

參賽隊員：許宏凱 張逸豪 施富仁 賴振德

國立中正大學電機工程研究所

摘要：

Bluetooth 的發展解決許多短距離無線連結的需求。但在 Bluetooth 相關產品陸續推出時卻有一技術最為複雜、相關技術背景知識要求最多，且目前尚無產品問世的領域 — Bluetooth Telephony。在本專題中我們開發出先進的 “藍芽無線電話系統”，完全實現 Bluetooth Telephony 功能。

在專題中軟體部份我們同時發展 Bluetooth 協定堆疊中，幾個最主要的核心技術： TCS (Telephony Control Protocol Specification) 協定、SDP (Service Discovery Protocol) 協定、 SDAP (Service Discovery Application Profile)、以及 L2CAP (Logic Link Control and Adaptation Protocol) 協定。另外，為了彌補 Bluetooth 規格未定義詳盡的地方，我們在系統及程式發展中創造了新的輔助函式，使系統程式能完整正常的運作。

在專題中的硬體部份我們採取自行製作之數位語音電路，作為日後繼續發展各項應用之基礎。這個電路，我們稱之為 "Bluetooth Phone Emulation Board"，我們運用 PCM Codec 晶片、SLIC (Subscriber Line Interface Circuit) 晶片來完成此一電路，並整合 Bluetooth Module 於其上。我們將此電路與一般市售有線電話搭配，實現藍芽無線電話系統之功能。

關鍵詞：藍芽無線電話系統、Bluetooth、TCS、SDP、L2CAP、PCM、SLIC。

一、前言

Bluetooth 的發展解決了許多短距離無線連結的需求，目前 Bluetooth Special Interest Group (SIG) 廠商也陸續推出各式 Bluetooth 產品。在 Bluetooth 的各種應用技術中，Bluetooth Telephony 技術最為複雜 相關技術背景知識要求最多、目前也尚無產品問世。我們的主題 —“藍芽無線電話系統”完全實現 Bluetooth Telephony 功能，此項應用需結合 Telecommunication、 Data communication、Embedded System、及語音與數據整合等多項技術。

由於在 Bluetooth network 中每一台 Bluetooth 設備皆可自由移動，因此在 piconet 中可提供的服務隨時都不同，這部分和傳統的網路環境具有很明顯的差異。為了能夠解決這一部份的問題，Bluetooth 制定一個很獨特的協定 － 服務搜尋協定（SDP）。 SDP 可讓每一個應用程式隨時、即時的找到目前 piconet 中所有提供服務的設備以及其服務的特性。由於 SDP 在 Bluetooth 協定中極為重要，因此 “服務搜尋協定 (Service Discovery Protocol) 實作”，也是我們在此次競賽中的主題。

我們的專題在技術創新部分，以更簡潔且更為快速的程式碼實現了複雜的 Telecommunication Signaling 功能；在程式語法上，我們比傳統實現 Q.931 Finite State Machine 的方法更為精簡，並達成相同的目的；我們創造新的輔助函式，使 Bluetooth protocol 不足之處更為完備；SDP 部分，我們自建符合 SDP 環境的資料庫與搜尋方式，讓 SDP 讀寫、搜尋資料庫能夠更為快速；我們將 Bluetooth protocol stack 實現成 Embedded system module，可以隨時依需要抽換；我們在資源有限的情形下，使用一般的市售電話來實現無線手機雛型，這種方式同時也讓一般市售電話機只要接上了我們的藍芽無線通訊裝置，就能馬上具有無線電話與對講機之功能；我們在 Ericsson Bluetooth Module 功能不全，不能建立 Piconet 的情形下，設計新的方法完成 WUG Group 建立與 Fast Inter-Member Access；在應用創新部分，透過 Bluetooth Gateway 先進的 Signaling 程序，我們可以建立起多對通話連線，可以選擇同時獨立通話而沒有數量的限制，這是目前家用無線電話無法做到的。

藍芽無線電話系統與搜尋服務協定的實作成果在第五節將會做更詳細的說明。

二、研究目的

由於 Bluetooth 技術具有不受方向限制、可穿透障礙物、具有比傳統電話更大傳輸量、更多方面應用、以及更快傳輸速率等優點，同時 Bluetooth 也是第一個可將通訊、資訊、消費性電子產品這三類廠商結合在一起的技術；再者，Bluetooth 提出後，世界各大廠一致看好並相繼加入聯盟；更何況臺灣廠商一向在消費性產品有著強大的競爭力，所以我們認為 Bluetooth 是一個非常適合台灣的研究領域，我們希望能協助廠商進行先期的研究開發，並作為他們的助力。

在專題題目選取方面，我們發現在一般辦公室的環境中，通常以有線電話作為電話分機，因而需要大量的佈線工程和許多昂貴的交換機設備，而有線電話由於必須固接於一處，不具可攜性，因此常會發生打電話到某分機卻找不到人，或是常需轉接數次才能找到通話對象的情形。

目前市面上販售的家庭用無線電話，容易受到雜訊干擾，而且並無加密的機制，有被竊聽的危險。一般的無線電話子機有數目的限制，而且只能使用於同一廠牌的母機，無法任使用者將其帶到其他場所與不同廠牌之無線母機通話。

由發展 Bluetooth 技術的構想與解決上述問題的想法，我們研發出了這套系統 --- ”藍芽無線電話系統 ”以及“ 藍芽服務搜尋協定 ”（SDP）。我們根據 Bluetooth 規格書中的 Cordless Telephony Profile、 Intercom Profile、及 Service Discovery Application Profile 來規劃我們的系統。

我們將 Bluetooth 規格書中的 TCS Profile 加以延伸，加強了 TCS 的功能，開發以藍芽為無線傳輸媒介的無線電話系統。這樣的無線電話系統可以裝設於辦公環境中，取代行之有年的有線電話分機系統，節省佈線經費及維護複雜度。同時由於 Bluetooth 技術的公開性，不同廠牌的無線子機和無線母機間的互通將成為可能。

藉由 TCS 中 Group Management 的功能，藍芽無線電話系統使辦公室分機可以隨身攜帶，而隨著使用者的移動加入不同 Bluetooth Gateway 的群組中，因此可達成分機的可攜性。

除了一般無線電話的應用之外，藍芽無線電話系統也支援無線子機之間互相通話的對講機功能，透過 Bluetooth Gateway 先進的 Signaling 程序，我們可以建立起無限多對通話連線，沒有數量的限制。

本專題所發展之藍芽無線電話系統與服務搜尋協定將來可與 PDA 或其他小型無線裝置整合，發展出能整合語音通訊、資料傳遞、資訊家電控制的多功能無線視訊電話系統。若將我們所開發的藍芽無線電話系統與行動電話系統整合在一起，更可發展出結合行動電話、室內無線電話、對講機功能於一身的三用電話。

三、原理與分析

1 TCS 結構圖

在 Bluetooth 的協定架構中，Bluetooth 無線電話系統使用到的核心，也是最主要的部分叫做 TCS Binary （Bluetooth Telephony Control protocol Specification Binary），它是根據 ITU-T Q.931 所制定出來的, 與 Q.931 不同的是, TCS 並沒有利用使用者和網路端來作區別，只以發話端 （Outgoing side） 與受話端 （Incoming side） 表示。如圖 3.1 所示，TCS 位於 L2CAP 上層，利用 L2CAP 傳送訊息及建立連結。

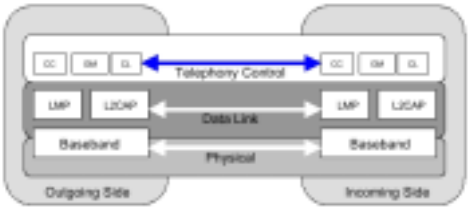


圖 3.1 TCS within the Bluetooth stack

TCS 功能如下：

Call Control （CC）：利用 signaling 的方式建立和釋放 Bluetooth 機器之間的語音和數據通道。

Group Management （GM）：利用 signaling 方式減輕管理 Bluetooth 機器所需的負擔。

1.1 CALL CONTROL （CC）

Call Control 主要目的是利用訊息傳遞以建立和釋放兩個藍芽裝置間的語音及資料通道；其包含了兩種程序, 通話建立程序 （CALL ESTABLISHMENT）及通話清除程序 （CALL CLEARING）。

1.1.1 CALL ESTABLISHMENT

1.1.1.1 通話的要求

由發話端（Outgoing side）傳送 SETUP 訊號給受話端（Incoming side）來啟動整個通話建立（call establishment）的程序。

發話端在傳送 SETUP MESSAGE 後，發話端的 CC 狀態機進入 Call initiated 狀態，而受話端接收到 SETUP MESSAGE，受話端的 CC 狀態機會進入 Call Present 的狀態。

受話端傳送 CONNECT 訊息來告訴發話端它所呼叫的電話已被接通，並且停止鈴聲，開始計時 （T313 計時器）。發話端收到受話端所傳來的 CONNECT 訊息後，會停止所有的計時，並且完成 SCO 或 ACL Link 兩端的連接，之後再傳送 CONNECT ACKNOWLEDGE 表示已建立起資料傳送的連結，而且進入 Active 狀態。而受話端在收到這個訊息後立即停止 T313 計時器，並進入 Active 狀態，之後兩端即可傳送資料。假使 T313 計時器逾時，則發話端會啟動啟動 Call Clearing 程序。

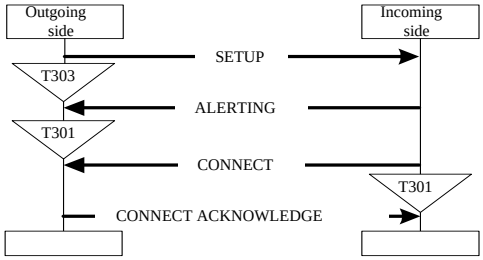


圖 3.2 整個通話建立訊息傳遞的流程

1.1.2 CALL CLEARING

發話端結束通話時，會傳送 DISCONNECT 訊息，結束與受話端之間的語音或資料通道，進入 Disconnect Request 狀態，而受話端進入 Disconnect Indication 狀態。

DISCONNECT 訊息告訴受話端要結束彼此之間的通訊，若彼此通訊已經結束，則受話端傳送 RELEASE 訊息給發話端會進入 Release Request 狀態。發話端收到 RELEASE 訊息後，釋放通道並傳送 RELEASE COMPLETE 訊息，回到 Null 狀態。受話端收到 RELEASE COMPLETE 訊息後，回到 Null 狀態。

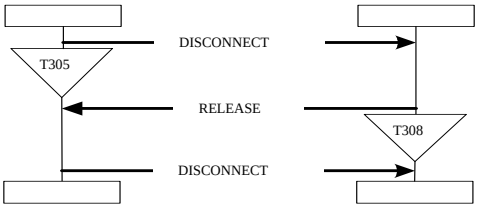


圖 3.3 通話清除程序訊息傳遞流程

1.2 GROUP MANAGEMENT （GM）

1.2.1 Wireless User Group

在 GM 的協定中，多台支援 TCS 的 Bluetooth 裝置可以組成一個 WUG。其中有一台裝置作為 WUG master，其它裝置則為 WUG member。

每個 WUG 中的成員都擁有所有 WUG member 或是 WUG master 裝置的資訊。WUG master 會藉由 Configuration Distribution 將這些資訊送給所有 WUG member。

1.2.2 概觀

Group Management 的功能分成三個不同的程序，分別為 Obtain access rights、Configuration distribution 和 Fast inter-member access，主要功能是在建立並維護 Wireless User Group (WUG) 的成員資訊。

1.2.2.1 Obtain Access Rights

經由使用 Obtain Access Rights 的程序之後，此台裝置就具有使用位於同一個 WUG 中其它裝置所提供的電話服務的權利。

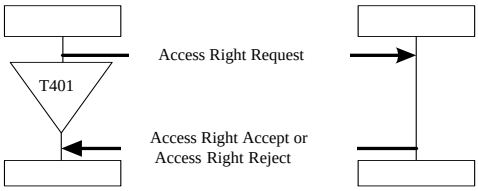


圖 3.4 Obtain Access Rights Message Flow

1.2.2.2 Configuration Distribution

當 WUG 中的狀況有了改變（例如，有裝置加入或退出、WUG 結構改變） 而且有必要通知在 WUG 中的 member 時就會進行 Configuration Distribution 程序。

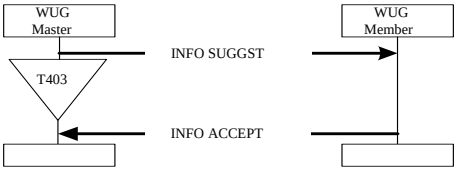


圖 3.5 Configuration Distribution Message Flow

1.1.2.3 Fast Inter-Member Access

當兩台 WUG member 在 WUG 中都處於可使用的狀況下，其中一台 WUG member 可利用 Fast Inter-Member Access 的程序取得和另一台之間的連接。在 Fast Inter-Member Access 的程序中，發話端會取得受話端的 clock 資訊，同時強迫受話端進入 Page Scan 模式並持續一段特定的時間。

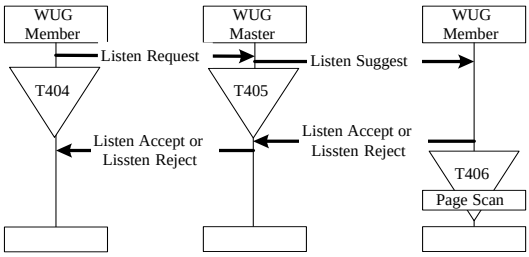


圖 3.6 Fast Inter-Member Access Message Flow

2 Service Discovery Protocol（SDP）

SDP 的架構及流程是運用 Client－Server 的模式在運作，如果 Bluetooth 設備欲在 piconet 中提供服務的話就必須具備 SDP Server 機制，必須注意的是每一台 Bluetooth 設備上無論有多少種服務項目，最多只能有一個 SDP Server。

所有由 SDP Server 提供的服務項目資訊都儲存在 SDP Server 的 Service Record 中，每個服務項目都會儲存在個別的 Service Record 中並且有一個唯一的 Service Record handle 對應，在圖 3.7 中每個 Service Record 均包含由 SDP Server 所提供的其中一種 Service 之全部 attribute，經由此 Service Record 可將一個服務項目的所有特性描述清楚。在 Service Record 中每個 Service Attribute 均分為兩個部份，*Attribute ID* 是用來區分此 Service Attribute 的功用，經過比對便可得知 *Attribute Value* 存放的內容形式。

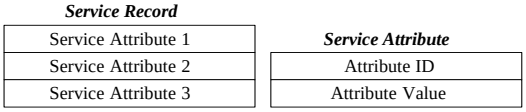


圖 3.7 Service Record and Service Attribute

每個服務項目均由許多 Service Class 組成，在 Service Class 中有分為 superclass 和 subclass 兩部份，subclass 會保留 superclass 的全部 attribute 並會另外定義新的 attribute。在 SDP 中每個 Service Class 都由一個 UUID（Universally Unique Identifier） 代表不同的 Service Class，而這些 UUID 是在 attribute ID 為 0x0001 的 Service Attribute 中。

在 SDP 中 UUID 是由 128 bit 所組成的數值，為了增進程式效能，UUID 也可以由 16-bit 或 32-bit 的數值代表。而 16-bit 或 32-bit UUID 的組成方法是由一個 Bluetooth Base UUID 再加上 32-bit 或 16-bit 數值所組成。其組成方式為：

$$128\_bit\_value = 16\_bit\_value * 2^{96} + BT\_Base\_UUID$$

$$128\_bit\_value = 32\_bit\_value * 2^{96} + BT\_Base\_UUID$$

Base UUID = 00000000-0000-1000-7007-00805F9B34FB

當 SDP Client 上層之 Application 要求尋找某一 Service 時會發出 SDP_ServiceSearch Request PDU 向 SDP Server 尋找，其中有一個參數是由特別的 UUID 所組成的 Service search pattern，當 SDP Server 收到此 PDU 後會和每個 Service Record 中 attribute 裡面的 UUID 進行比對。如果 Service search pattern 中的 UUID 完全符合 Service Record ，此時 SDP Server 會將符合條件的 Service Record 中的 ServiceRecordHandle 放入 SDP_Service SearchResponse PDU 內傳回給 SDP Client。接著 SDP Client 端會針對其中一個 Service Record 發出 SDP_ServiceAttributeRequest PDU，進一步向 SDP Server 端查詢所需之 attribute。此外，SDP 可將 ServiceSearch Pattern 和要找的 Attribute ID 直接放入 SDP_ServiceSearchAttributeRequest PDU 送往 SDP Server。利用這種做法可節省一次封包所需的傳送時間，只需經過一次查詢後 SDP Server 就會將前面所提的兩步驟做完才傳回一個 SDP_ServiceAttributeResponse PDU。除了直接向 SDP Server 要求符合的 Service Record 之方法外，SDP Client 也可藉由 Browsing 的方式向 SDP Server 查詢在 SDP Server 上有提供的服務項目。而 SDP Client 經由 Browsing 後可得到表 3.1 的資料，經由表 3.1 便可由 SDP Client 組成圖 3.8 的架構。

Service Name	Service Class	Attribute Name	Attribute Value
SDP	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	SDP_Group
L2CAP	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	L2CAP_Group
TCS-Bin	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	TCS_Group
SDAP	ServiceDiscoveryServerServiceClassID	BrowseGroupList	SDP_Group
Cordless	CordlessTelephony	BrowseGroupList	TCS_Group
Intercom	Intercom	BrowseGroupList	TCS_Group

表 3.1 Browse Table

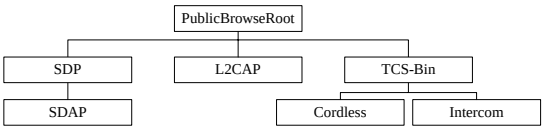


圖 3.8、 Browse Hierarchy

3 Logical Link Control and Adaptation Protocol （L2CAP）

3.1 L2CAP 簡介

The Logical Link Control and Adaptation Layer Protocol 簡稱 L2CAP，歸屬於 Bluetooth Protocol Stack 中之資料鍊結層 (Data Link Layer)，主要功能在提供上層的 Protocols 兩種 Data Services “ Connection-Oriented ” 及 “ Connectionless “，除此之外 L2CAP 亦具有 Protocols Multiplexing 、 Segmentation and Reassembly 、 Quality of Service 及 Group Management 的功能。

3.2 L2CAP 層功能介紹

3.2.1 Protocol Multiplexing

Bluetooth 的規範中允許多個上層 Protocol 同時運作但兩個藍芽裝置之間最多只能有一條 ACL Link，因此 L2CAP 必須具有 Protocols Multiplexing 的能力，並能辨別出上層不同的資料封包。

L2CAP 層於規範中定義了三種類型的 Logical channel，分別為 Connection-Oriented、Connectionless 以及 Signaling，其中前兩種 Channel 的建立，都必須先經過 Signaling 的動作來完成。Connectionless Channel 是提供 Point to Multi-Point 之連結，而 Connection-Oriented Channel 則提供 Point to Point 之連結。

3.2.2 Segmentation and Reassembly

在規範中規定 L2CAP 層所能接收上層之封包最大為 64K Bytes，但是 Baseband 接收來自 L2CAP 層之封包最大只有 341Bytes，因此 L2CAP 層必須對上層較大的封包做切割的動作，使其符合 Baseband 所能傳送之封包大小。同樣地， L2CAP 層也必須具備將多個 Baseband 封包重組還原的能力，圖 3.9 為 L2CAP 層進行封包的切割與重組的示意圖。如果 L2CAP 層與 Baseband 之間有 Host Controller Interface （HCI） Protocol 存在時，封包之切割與重組工作將由 HCI 來執行。

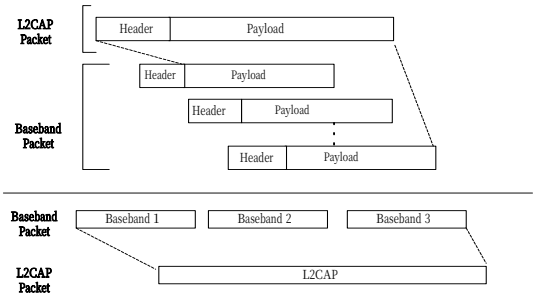


圖 3.9 Segmentation and Reassembly service 的示意圖

3.3 State Machine

如圖 3.10 所示為 L2CAP 層與上下層 Event 與 Action 關係的示意圖。圖中的符號 L2CA_ 表示上層 Protocol 與 L2CAP 層間訊息的互動，符號 LP_ 表示 L2CAP 層與下層 Protocol 間訊息的互動，符號 L2CAP_ 表示 Peer to Peer 間訊息的互動。且從上層 Protocol 收到的訊息稱作 Request ，相對應的回應叫 Confirm ，而從下層 Protocol 得到的訊息稱作 Indication ，相對應的回應則稱為 Response。L2CAP 與上下層彼此間訊息的溝通，即經由一序列的 Request， Indication、 Response 及 Confirm 的動作來完成。

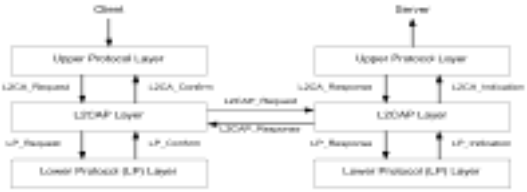


圖 3.10 L2CAP Layer Interactions

以下為一簡單的 State Machine 例子，說明 L2CAP 層是如何根據 Event 以及 Channel 的 State 來觸發相關的 Action ，並最後進行 State 的轉換。

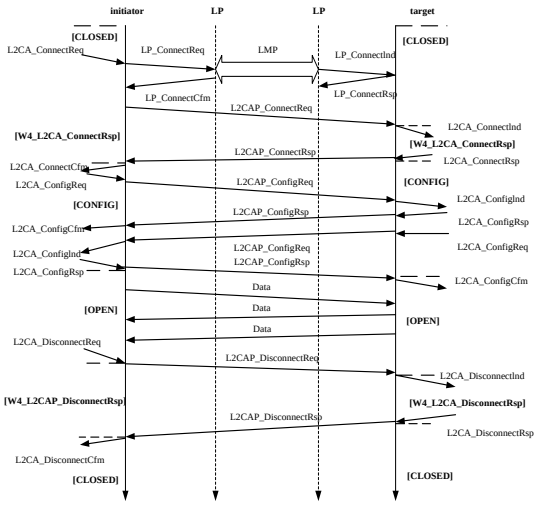


圖 3.11 Message Sequence Chart Of Basic Operation

四、軟硬體系統

1 軟體架構

圖 4.1 所示為本專題中 Bluetooth 軟體部分的架構。其中 TCS、 SDP、 L2CAP 之功能已於第三章中描述。(LMP 及 Baseband 是由 Bluetooth module 的硬體及韌體來實現。)

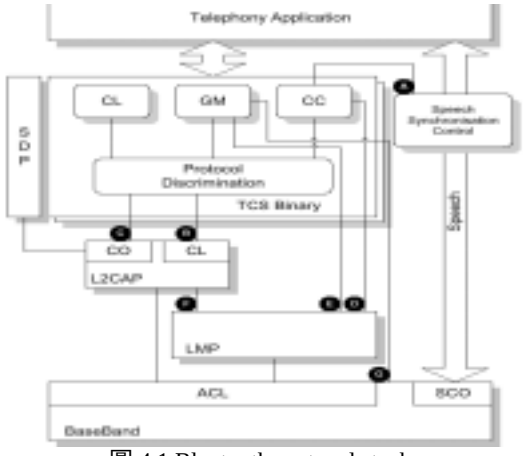


圖 4.1 Bluetooth protocol stack

以下我們就圖 4.1 中，各部份間之介面分別說明：

介面 (A)： 在 TCS Profile 中，電話應用層 (Telephony Application) 與底層 (Baseband) 間存在著提供數位語音傳送的路徑 (speech path) 。TCS 中的 CC 透過此介面控制該語音路徑的連接與中斷。

介面 (B)： 此介面使 TCS 能夠利用 L2CAP 所提供的非連結導向通道，點對多點 (point-to-multipoint) 地傳送 signaling 訊息。Gateway (GW) 透過此介面傳送 TCS 的廣播訊息，而各 Terminal (TL) 透過此介面接收該訊息。

介面 (C)： 此介面使 TCS 能夠透過 L2CAP 的連結導向通道，點對點 (point-to-point) 地傳送 signaling 訊息。

介面 (D)： 此介面使 TCS 中的 CC 能夠直接控制 LMP 以建立或釋放 SCO link。

介面 (E)： 此介面使 TCS 中的 GM 在啟始和 key-handling 時能夠使用 LMP 的功能。

介面 (F)： L2CAP 與 LMP 間的介面。

介面 (G)： 此介面使 GM 能夠直接控制 LC/Baseband，並使用 Baseband 中的 inquiry、paging 和 pairing 等動作。

在本專題中，我們透過了 HCI 與 Bluetooth Module 溝通，因此 (D)、(E)、(F)、(G) 等介面的實現我們是以透過 HCI 間接對 LMP、Baseband 下控制指令的方式來達成。在程式的實作上， Bluetooth 協定堆疊中各層協定間我們以函式的呼叫與傳回值作為其介面，並完整地定義了上下層間所需提供的服務、回應、及所需傳遞的變數。這種作法不但使我們能夠在開發軟體時能夠讓各層專注於其功能的實現及正確性，同時並帶來了模組化的便利性，讓我們能在相同介面上開發新的程式碼而免去牽一髮而動全身的危險，加快了測試、整合的進度。

在軟體的開發過程中為了讓上層的 TCS 及 SDP 可以和下層 L2CAP 分開且各自獨立發展。我們規劃了兩個方案，一是直接使用 Bluetooth module 配合軟體的開發；或是利用軟體模擬資料傳送動作，利用 Socket 作為傳送資料的通道進行開發。這兩種方式的優缺點如下：在軟體方面若以 Socket 方式模擬下層傳送資料的動作，可避免每一部份的開發都需要 Bluetooth Module 而造成硬體不足的困擾，也可讓各層協定獨力發展，在發生問題或流程驗證時可減少變數，但在將來整合時必須將軟體模擬的部分改為可用於真正硬體上的架構，但這不需要花太多時間。若直接使用 Bluetooth module 發展，則未來不需面對更改架構的問題，但相對的，就沒有使用軟體模擬方法的優點，而且在我們的操作經驗中，每次測試時 Bluetooth module 要進行硬體初始的動作，而這個動作將會比用軟體模擬多花費將近十秒的時間，這一點也將造成開發人員之困擾。因此，在比較這兩種方式的優缺點後，我們決定選擇以軟體模擬方式先進行開發，等開發完成後時再與硬體整合，進行完整的測試動作。而這也如我們預期的一樣，減少了很多的困擾，同時讓開發速度加快。

我們將 Bluetooth 協定堆疊以 Linux Kernel Module 的方式實現在 Linux OS 上，並整合進 Embedded Linux Kernel 中。由於在 Linux Kernel Mode 中，除錯相當困難，稍有不慎即會造成系統當機，所以我們將發展過程分為兩個階段，初期在 Linux User Mode 下發展系統，成熟後再將所有 Protocol 改寫為 Kernel Module。另外，整合到 Embedded Linux System 的原因是藍芽無線電話系統為一嵌入式系統，我們必須將 Linux 作業系統的核心做大量的修改與微小化，以便在嵌入式系統中與硬體整合。

另外，由於 Bluetooth Core Specification 並

沒有針對介面詳加定義，所以第四章中描述的程式架構除了標準流程外，大部分都是我們自行發展的輔助函式，與實作的技巧和方法。

1.1 Group Management （GM）

我們在程式實作上將 WUG master 與 WUG member 分別撰寫，其流程圖分別如圖 4.2、圖 4.3 所示。至於其動作可以分為加入 WUG 及進行 Fast Inter-Member Access 流程兩部份。

加入 WUG 部份可分為處理 Bluetooth 裝置加入 WUG 之動作及進行 WUG member 資料更新兩步驟。首先，若 WUG master 收到 ACCESS_RIGHTS_REQUEST 訊息後，會呼叫 access_rights_request_server() 判斷是否要讓提出要求之 Bluetooth 裝置加入 WUG。若允許則送出 ACCESS_RIGHTS_ACCEPT 訊息，否則送出 ACCESS_RIGHTS_REJECT 訊息。當 Bluetooth 裝置成功加入 WUG 之後，代表 WUG 之資料已經有所更動, 因此 WUG master 會進行 Configuration Distribution 之動作, 對每台 WUG member 送出 INFO_SUGGEST 訊息，藉此將目前 WUG member 之資料進行更新。

在執行 Fast Inter-Member Access 動作時，若 WUG master 收到 LISTEN_REQUEST 訊息會先判斷 Incoming 端是否為 WUG member, 若是則將 LISTEN_SUGGEST 訊息送給 Incoming 端，代表有 WUG member 要和它通話，否則送出 LISTEN REJECT 給 Outgoing 端同時將拒絕原因一併送出。當 Incoming 端收到 LISTEN_SUGGEST 訊息後若有能力和 Outgoing 端通話便送出 LISTEN ACCEPT 訊息；否則送出 LISTEN REJECT 訊息。WUG master 收到由 Outgoing 端送出之訊息後會將訊息再轉送給 Incoming 端。

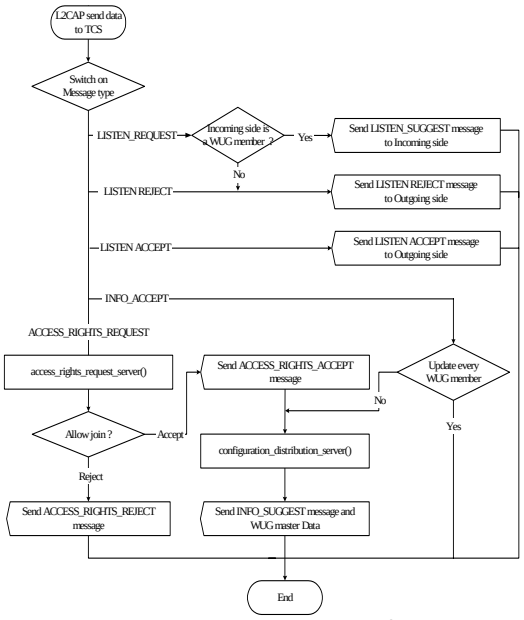


圖 4.2 TCS WUG Master 流程圖

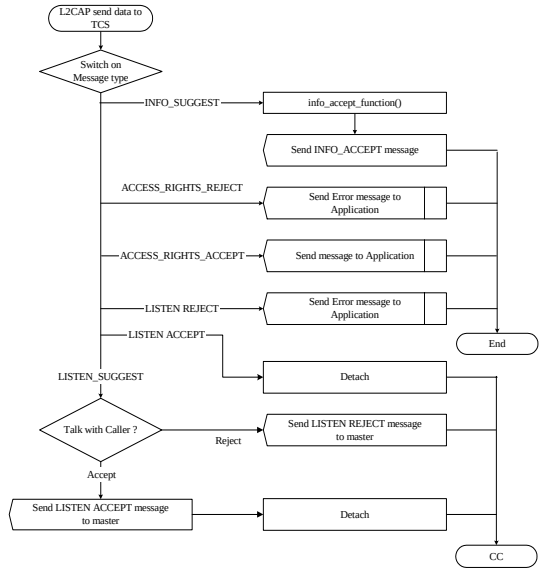


圖 4.3 TCS WUG Member 流程圖

1.2 Call Control （CC）

Call Control entity 是一種 Finite State Machine 的機制。而 Finite State Machine 在有事件發生時，會依本身的狀態 （CC State） 和發生的事件 (Event)，來決定該做的動作。假使在寫 C code 時單純使用 switch 來判斷，會需要兩層 swtich，第一層判斷是哪種 message，而第二層判斷 state。此種方法造成 debug 困難，而且最主要的，就是程式碼會變的非常複雜。

因此我們將 CC 所有的狀態和所有會發生的事件互相配對，而以一個二維陣列來記載不

同狀態與事件的組合下所應有的動作，我們稱此二維陣列為狀態表。

使用狀態表的好處有： 1. 可以清楚了解整個 Finite State Machine 的動作流程。 2. 除錯容易。 3. 可依據實際需求，輕易增加或修改程式碼。

藉由圖 4.4，程式可以查出應該呼叫的函式以處理發生的事件。

```
int call_control_state[11][18] = {
                    /*  <--  state  -->  */
    /*0  1  2  3  4   5  6  7  8  9   10 11 12 13 14 15 16 17*/  /* events */
    {0, 7, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*0 ALERTING*/
    {0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*1 CALL_PROCEEDING*/
    {0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*2 CONNECT*/
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0},   /*3 CONNECT_ACKNOWLEDGE*/
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*4 PROGRESS*/
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*5 SETUP*/
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},   /*6 SETUP_ACKNOWLEDGE*/
    {0, 0, 0,10,10,10,10,10,10, 0,10,18,19, 0, 0,10, 0, 0},   /*7 DISCONNECT*/
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,11, 0, 0, 0, 0, 0, 0},   /*8 RELEASE*/
    {0, 0,12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,12, 0, 0, 0, 0},   /*9 RELEASE_COMPLETE*/
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,13, 0, 0},   /*10 INFORMATION*/
};
```

圖 4.4、 State Table

動作代碼與函式間的對應，我們以一 Action Vector 來完成。 Action Vector 是一個一維陣列，其中各元素皆為指向不同函式的指標(pointer)，存放著各函式的位址。以動作代碼為此陣列之 index，即可查出動作代碼所對應的函式位址，並呼叫動作代碼所對應的函式。這種實作方式是 Specification 上面沒有提到，卻又必須要有的，我們稱之為輔助函式。

```
static int (*act_vec[20]) (struct cc_status * cc,struct TCS_message * temp_p) =
{act00, act01, act02, act03, act04, act05, act06, act07, act08, act09,
act10, act11, act12, act13, act14, act15, act16, act17, act18, act19};
```

圖 4.5、 Action Vector

以下之流程圖說明在不同狀態及事件的組合下 CC 的動作，圖 4.6 為通話建立程序所經過之流程，圖 4.7 為中斷通話所經過之流程，而圖 4.8 則是 timeout 發生時應作之處理。

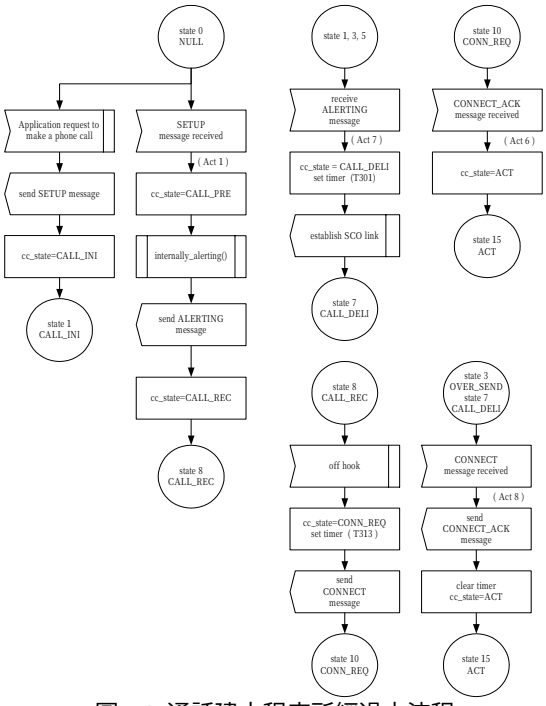


圖 4.6 通話建立程序所經過之流程

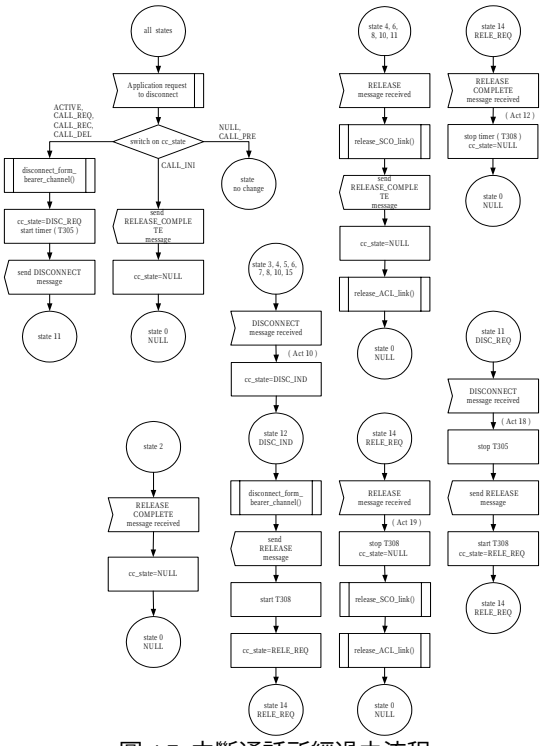


圖 4.7 中斷通話所經過之流程

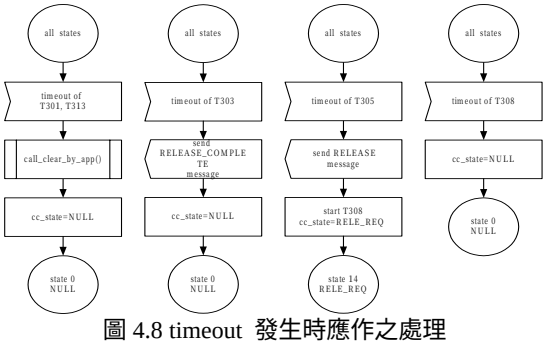


圖 4.8 timeout 發生時應作之處理

1.3 L2CAP

1.3.1 CLOSED State

如圖 4.9 所示，當上層要與 L2CAP 層溝通時必需借由 L2CA_init 事件把參數 (Protocol Service Multiplexer-PSM 與 Return_ADDR) 告知 L2CAP 層，以作為往後區別及回傳到上層之依據。如果 L2CAP 層收到上層 L2CA_ConnectReq 事件時，會先判斷參數 PSM 與 Bluetooth Device Address (BD_ADDR) 是否正確，接著確認 Baseband ACL Link 是否存在；如果ACL Link已存在, 則L2CAP層建立Logical Channel，同時儲存相關參數，接著進行 L2CAP_ConnectReq 的動作且始動 RTX 計數器，以確認傳送狀況，進入 W4_L2CAP_CONNECT_RSP State。這種實作方式為新增的輔助函式。

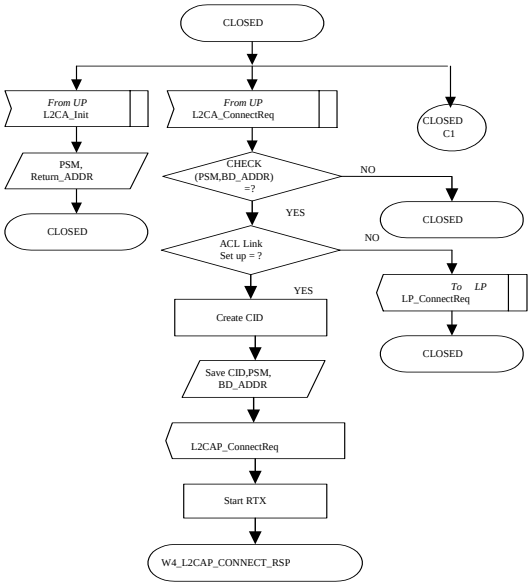


圖 4.9 CLOSED State 流程圖

如圖 4.10 所示，如果收到下層的 LP_ConnectInd 事件，L2CAP 層會判斷 BD_ADDR 是否正確，如果正確則回應 LP_ConnectRsp 動作至下層，回到 CLOSED State。若 L2CAP 層收到下層 LP_Connect_Cfm 事件，首先會先判斷 Status 參數是否成功，接著進行 L2CAP_ConnectReq 的動作且起始 RTX 計數器，進入 W4_L2CAP_CONNECT_RSP State；否則以 L2CA_ConnectCfm 告知上層無法建立 Channel，並回到 CLOSED State。若收到 Peer 端的 L2CAP_ConnectReq 事件，L2CAP 層會儲存相關參數且產生對應於 Peer 端之 CID 值，接著會進行 L2CA_ConnectInd 動作告知上層連線狀態及 CID 值，且進入 W4_L2CA_CONNECT_RSP State。

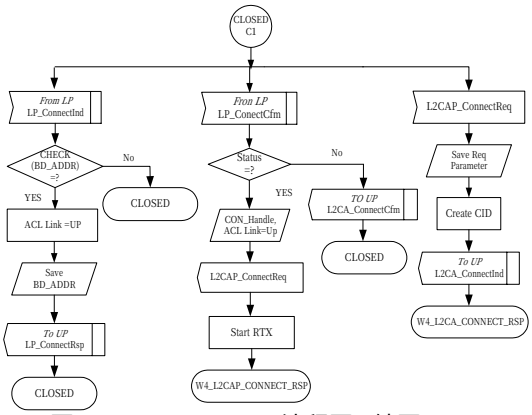


圖 4.10 CLOSED State 流程圖 (續圖 4.9)

1.3.2 W4_L2CA_CONNECT_RSP

如圖 4.11 所示，L2CAP 收到上層的 L2CA_ConnectRsp 事件並藉由判斷 Response 參數來進行 L2CAP_ConnectRsp 的動作，進入 CONFIG State；或傳送 L2CAP_ConnectRsp，告知 Peer 端失敗的原因，且清除 CID，進入 CLOSED State。

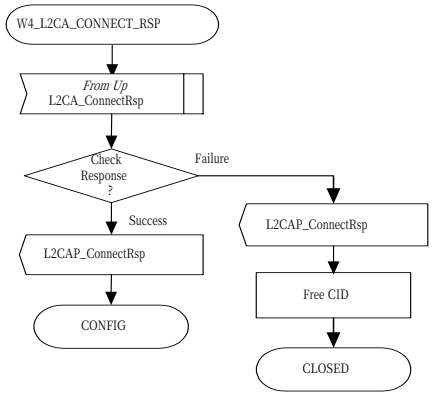


圖 4.11 W4_L2CA_CONNECT_RSP State 流程圖

1.3.3 W4_L2CAP_CONNECT_RSP State

如圖 4.12 所示，若收到 Peer 端所送之 L2CAP_ConnectRsp 事件，L2CAP 層會先終止 RTX 計數器，然後判斷 Result 參數之結果，如果為 Pending 則傳送 L2CA_ConnectPnd 通知上層，並啟動 ERTX 計數器，回到 W4_L2CAP_CONNECT_RSP State；如果成功則傳送 L2CA_ConnectCfm 至上層，告知建立連線的狀態及 CID 值，進入 CONFIG State；如果失敗則傳送 L2CA_ConnectCfm 至上層，告知上層失敗的原因，進入 CLOSED State。

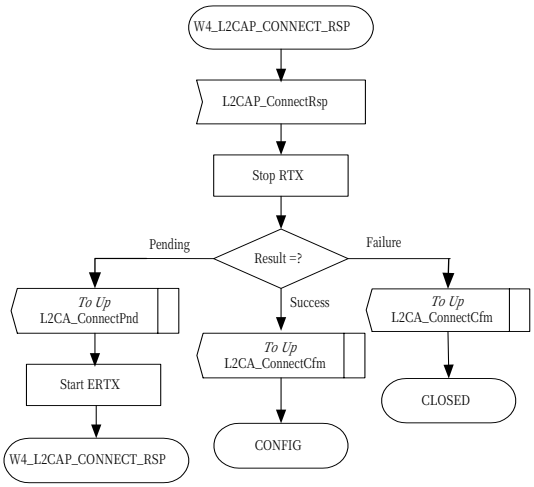


圖 4.12 W4_L2CAP_CONNECT_RSP State 流程圖

1.3.5 OPEN State

如圖 4.15 所示，若收到上層 L2CA_DataWrite 事件時進行 L2CAP_Data 的動作，將資料傳送到 Peer 端；若收到 Peer 端所送的 L2CAP_Data 事件時，會將所收到資料利用 L2CA_DataRead 傳送到上層。若收到上層 L2CA_DisconnectReq 事件要求中止通道會進行 L2CAP_DisconnectReq 動作, 將訊息傳遞至 Peer 端，同時起動 RTX Timer 並進入 W4_L2CAP_DISCONNECT_RSP Sate。若收到 Peer 端送 L2CAP_DisconnectReq 事件則利用 L2CAP_DisconnectInd 通知上層，告知 Peer 端要求中斷通道，進入 W4_L2CA_DISCONNECT_RSP State。

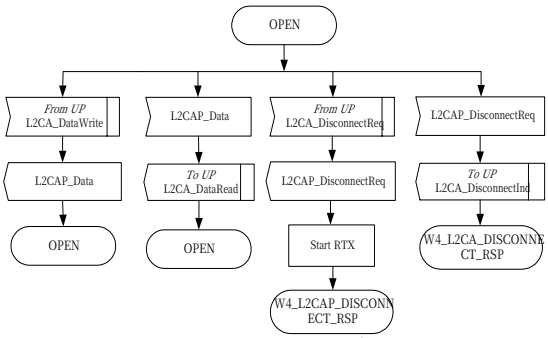


圖 4.15 OPEN State 流程圖

1.3.6 W4_L2CA_DISCONNECT_RSP

如圖 4.16 所示，若收到上層的 L2CA_DisconnectRsp 事件時，會進行 L2CAP_DisconnectRsp 動作，並把欲中止的 CID 清除，接著進入 CLOSED State 。

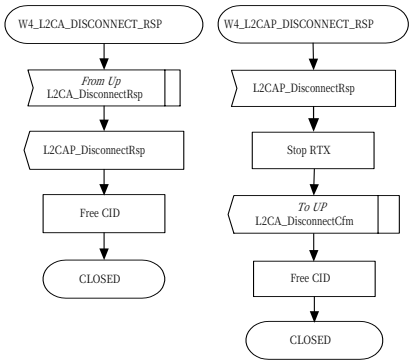


圖 4.16 (W4_L2CAP/W4_L2CA) _DISCONNECT_RSP State 流程圖

1.3.7 W4_L2CAP_DISCONNECT_RSP State

如圖 4.16 所示，若收到 Peer 端的 L2CAP_DisconnectRsp 事件時，會先中止 RTX Timer, 然後進行 L2CA_DisconnectCfm 動作通知上層通道中斷成功，同時 L2CAP 層會將 CID 給釋放掉，進入 CLOSED State 。

1.4 服務搜尋協定 (Service Discovery Protocol)

本專題之另一重點在 Service Discovery Protocol (SDP) 的實現。關於 SDP 之實作，Bluetooth 規格書中定義了 Service Discovery Application Profile (SDAP)，其架構如圖 4.17 所示，我們根據此架構進行程式的設計與撰寫。我們將圖 4.17 中 BT_module_Cntrl 部份包含在應用程式裡，因此應用程式可以經由 HCI，控制 Baseband 執行 Inquiry 等動作, 以找尋附近之其他 Bluetooth 裝置。

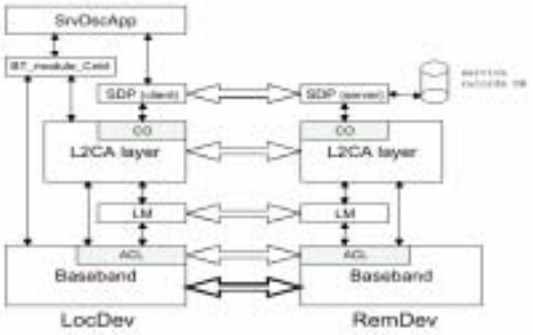


圖 4.17、 SDAP 之架構圖

當一個 Bluetooth 裝置 (如圖 4.17 中之 LocDev) 欲查詢周遭其他 Bluetooth 裝置 (如圖 4.17 中之 RemDev) 是否提供某特定服務時， LocDev 上的應用程式會呼叫 SDP 部分之程式, 以 SDP Client 的身份發出 Request 給 RemDev，而 RemDev 上的 SDP Server 依此 Request 在其 Service Record 資料庫中搜尋, 隨後發出 Response 將搜尋結果傳回。 SDP 之基本協定時序圖如圖 4.17 所示。

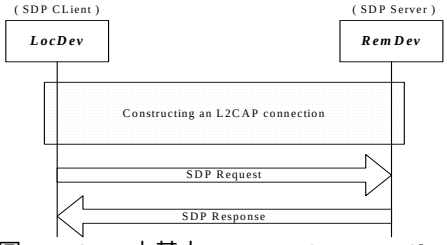


圖 4.17 SDP 之基本 Message Sequence Chart

同一個 Bluetooth 裝置可同時扮演 SDP Client 及 SDP Server 之角色，因此我們以圖 4.18 的方式規劃及撰寫程式。

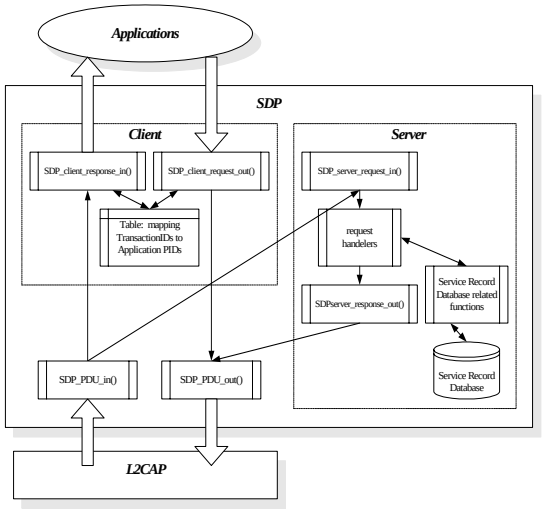


圖 4.18 SDP 程式規劃

1.4.1 SDP Server 的實現

我們將 SDP 的 Service Record 資料庫以鏈結串列的方式來實現，此種方式的好處是簡單、記憶體管理較靈活，雖然在資料搜尋速度上不如其他資料格式 (如二元樹、 Hash Table 等)，但因 SDP Server 通常不需維護龐大之 Service Record 資料庫，考量其程式實現之便利性及複雜度，我們選擇了較簡易的單向鏈結串列來進行 Service Record 資料庫之實現。

SDP Server 僅負責在收到 Request 時搜尋資料庫產生 Response，圖 4.19 所示為我們撰寫之 SDP Server 程式流程圖。

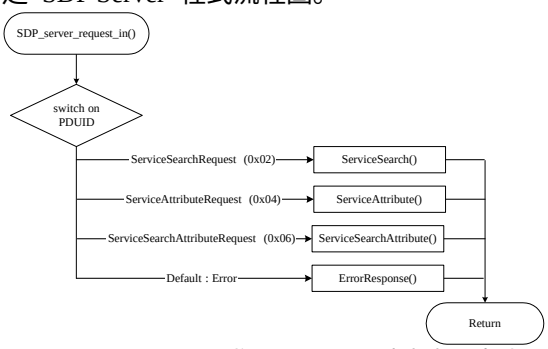


圖 4.19 SDP Server 收到 Request 時之處理方式

1.4.2 SDP Client 之實現

當上層應用程式要搜尋或詢問某項服務和其屬性時，會呼叫 SDP 之 Client 部分程式，發出 Request 向 SDP Server 查詢；而當 Server 回應 Response 時， SDP Client 需將其中所記載之各項服務及其屬性等，進行整理並向上送到相對應之應用程式。

為支援多個應用程式同時使用 SDP 的服務查詢功能，我們利用 SDP 協定中每次查詢都有不同 TransactionID 的特性，建立起應用程式與 TransactionID 之對應表。當應用程式呼叫 SDP Client 發出 Request 時，將此應用程式之 PID 及對應之 TransactionID 填入此表，如此即可在收到 Response 時找到此次查詢所對應之應用程式。

圖 4.20、圖 4.21 為我們所撰寫之 SDP Client 程式流程圖。

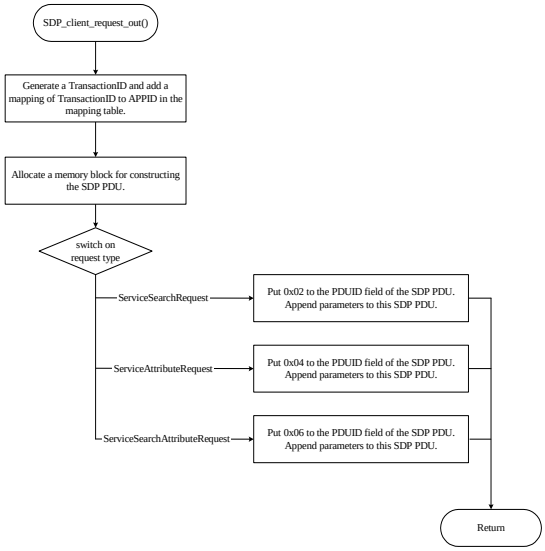


圖 4.20 SDP Client 發出 Request 之程式流程圖

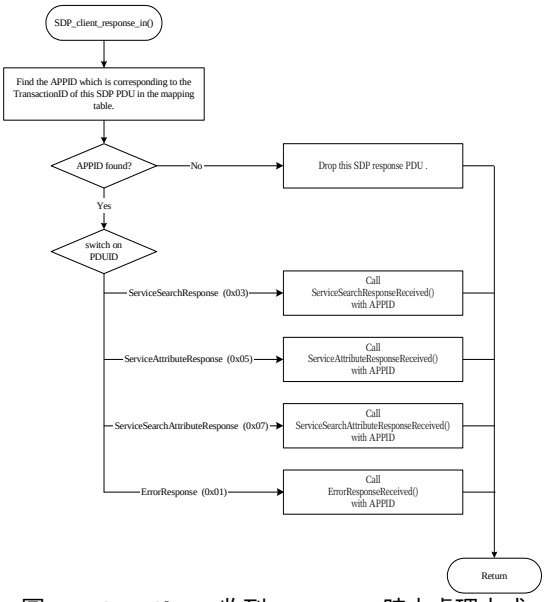


圖 4.21 SDP Client 收到 Response 時之處理方式

1.4.3 SDP Browsing 的實現

當 SDP Client 欲瀏覽 SDP Server 所提供之服務時，可藉由 Browsing 之功能取得 SDP Server 上的 Service Record 資料庫內容。由於此功能在 Specification 上完全沒有定義如何完成，故實作方法完全由我們自行以輔助函式研發。

SDP Client 使用 Browsing 功能時，首先會送出內含 ServiceSearchAttributeRequest 訊息的封包，在此封包中包含 ServiceSearchPattern 及 AttributeIDList 兩種參數。其中在 ServiceSearchPattern 這一參數欄位填入 BrowseGroupDescriptorServiceClassID 作為搜尋目標；而在 AttributeIDList 中填入 BrowseGroupList 和 GroupID 兩種 AttributeID。藉此即可獲得 SDP Server 所有 GroupDescriptor 之資料。

SDP Client 接著會繼續搜尋所有 GroupDescriptor 下之各 Service Records，其藉著送出 ServiceSearchAttributeRequest 訊息，並在此訊息之 ServiceSearchPattern 欄位中放入欲搜尋之 GroupDescriptor 之 GroupID，而 AttributeIDList 欄位放入 BrowseGroupList。這是因為我們要搜尋的對象是屬於特定 Group 的 Service Records，將 BrowseGroupList 填入 AttributeIDList 欄位可將屬於該 Group 的 Service Record 通通找出。

由於在 Browsing 的第一次詢問動作中我們可能會得到不只一個 GroupID，所以搜尋每個 Group 的 Service 可能需經過多次才可完成。圖 4.22 所示為我們所設計之 SDP Browsing 程式流程。

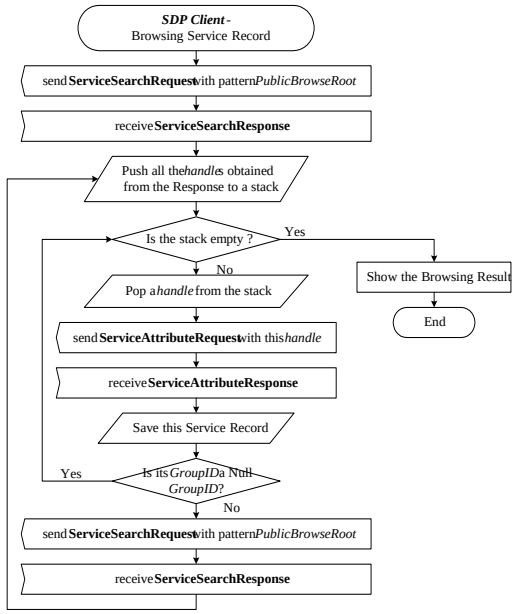


圖 4.22 SDP Browsing 之程式流程圖

2 硬體系統架構

本專題的硬體可分為兩種不同之裝置：藍芽無線電話 Gateway、藍芽無線電話分機。兩者皆能支援 Bluetooth 之 SDP 功能。其中 Gateway 負責管理 Wireless User Group（WUG），而無線電話分機可加入此 WUG 成為其成員。加入 WUG 後各分機可透過 Bluetooth TCS Profile 中 GM 部分所定義的 Fast Inter-Member Access 程序，與其他分機以藍芽無線傳輸直接通話。圖 4.23 為本專題硬體部分之示意圖，包含一部藍芽無線電話 Gateway 及兩部藍芽無線電話分機。

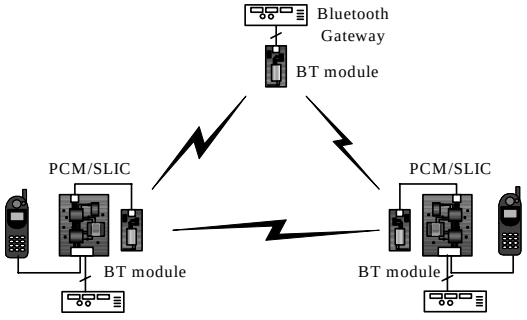


圖 4.23、 System Infrastructure topology

我們使用 Two-CPU 之解決方案來實現藍芽無線通訊功能：在 Host PC 上執行 Bluetooth 協定程式，而在 Bluetooth Module 上執行 LMP、BaseBand 等功能。我們採用 Ericsson 公司所開發出之 Bluetooth Application Kit 模組以完成上述功能。此模組提供 UART 介面 (Serial

Port) 及 USB 介面以供控制，並提供一組 PCM 信號的介面。

我們將 Bluetooth Module 和電話語音介面整合於自行設計之電路板上，以實作藍芽無線電話分機. 藍芽協定程式及應用程式在 Linux 平台上開發，於工業級電腦上執行，而透過電腦的 Serial Port 傳送 HCI Command 到 Bluetooth Module 以進行控制。

本專題中硬體關鍵部分為自行製作之電路，我們稱之為 "Bluetooth Phone Emulation Board"，其功能方塊圖如圖 4.24 所示。我們使用 PCM Codec、 Subscriber Line Interface Circuit (SLIC) 晶片，設計出電話語音介面，提供 RJ-11 插座使一般家用電話能夠直接將電話線接上而變成藍芽無線話機。

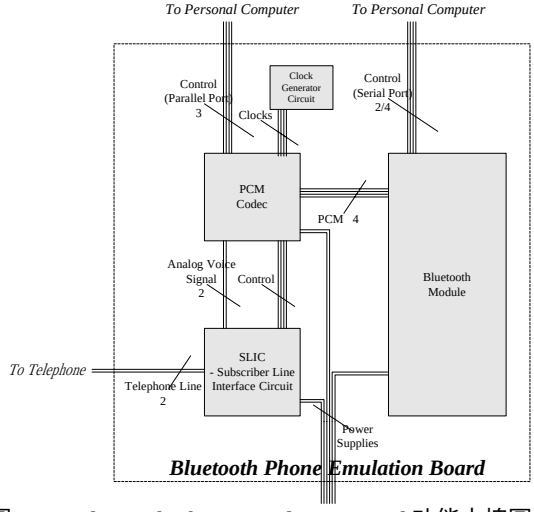


圖 4.24、Bluetooth Phone Emulation Board 功能方塊圖

在通話中其中一方的使用者對著電話機的話筒講話而產生語音訊號，我們的電話語音介面將此類比語音訊號自電話回路中取出，送入 PCM Codec 進行取樣、量化及編碼，以成為數位語音資料。PCM Codec 所輸出之數位資料串流經由 PCM 介面送入 Bluetooth Module，再透過藍芽無線傳輸到達另一通話方。當對方透過藍芽模組接收到數位之 PCM 資料串流後，由其電話語音介面中的 PCM Codec 解出類比語音訊號，再由 SLIC 將訊號載於電話線上，最後傳至話機而呈現於聽筒。

我們使用 National Semiconductor 公司的 TP3070 - COMBO II Programmable PCM CODEC/Filter，作為我們的 PCM Codec。此晶片需先透過其所提供之 Control Port，將晶片內之控制用暫存器填入所需之設定值 (如： 工作時脈頻率、 PCM 介面所使用之 Channel、TimeSlot 等)，方能正確工作。我們以工業級電腦之 Parallel Port 與此晶片之 Control Port 連接，撰寫程式使 Parallel Port 產生控制信號以存取晶片內之暫存器，進行設定。

SLIC 的主要功能是提供電話公司局端的一般用戶線路介面，提供用戶端饋電、振鈴訊號，並將類比之語音訊號載於電話線路之上。目前已有許多晶片設計公司將 SLIC 做成現成的晶片，我們採用的是 Ericsson 公司的 PBL 38621/1。

一般市售電話機提供了聽筒、麥克風、按鍵等裝置之控制，透過 SLIC 提供之介面，一般電話得以與我們的 Bluetooth Phone Emulation Board 連接，如此便能使用其各項現成之功能。這種方式同時也讓一般市售電話機只要接上了我們的藍芽無線通訊裝置，就能馬上具有無線電話與對講機之功能。

3 Piconet 的建立

由於 Ericsson Bluetooth Module 不支援建立 Piconet 功能。所以為了解決此問題，除了規格書中所訂定的 Intercom 正常程序之外，我們還設計了解決此問題的機制。

因為 Ericsson Bluetooth Module 無法讓 Gateway 與兩台 Member 同時建立連線，因此在 Gateway 完成與某個 Member 的訊息交換後，我們需先將 Physical ACL Link 中斷，Gateway 才能與另一個 Member 建立連線，並進行下一步動作。

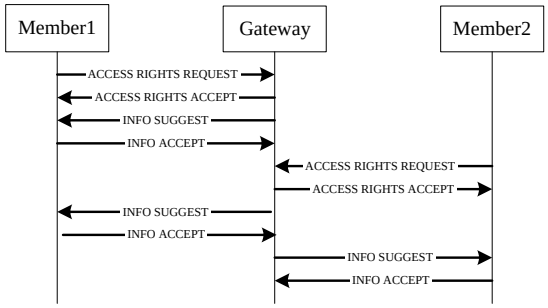


圖 4.25 加入 WUG、更新 WUG 資訊時之切換情形

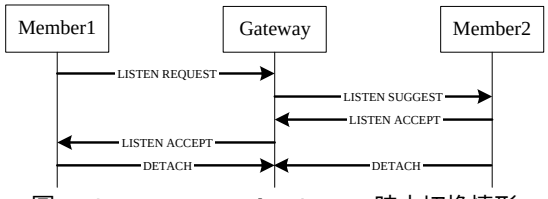


圖 4.26 Fast Inter-Member Access 時之切換情形

我們以 Intercom 的 Message Sequence Chart 來說明我們的解決方法，例如，當 Member 1 回

應 Gateway Info Accept 後，按照正常的程序，Member 2 可以直接與 Gateway 進行連線，但由於硬體的限制，Gateway 必須先將與 Member 1 之間的 Physical ACL Link 中斷，Member 2 才能與 Gateway 建立連線。同理，當 Gateway 回應 Member 2 ACCESS RIGHTS ACCEPT 後, 需中斷與 Member 2 之間的 Physical Link，才能與 Member 1 進行連線的建立，接下來進行 INFO Suggest 的動作。類似動作在 Member 1 與 Member 2 皆已完成與 Gateway 之間的訊息交換後才結束。

五、實驗結果與比較

1 協定程式執行畫面

1.1 在 SDP 部份我們先展示 Service Search 的實驗結果：

－ 當 SDP Server 收到 SDP Client 所送來之 ServiceSearchRequest 訊息後會根據所要尋找之 Pattern （在此為搜尋 BrowseGroupDescriptorServiceClassID 之 UUID）在資料庫中進行搜尋，搜尋完成會將結果傳回 SDP Client。

－ SDP Client 經由 SDP Server 所傳回之結果可知有三個 ServiceRecordHandle 是符合所要找之條件。接下來可根據所要搜尋之 ServiceRecordHandle 尋找其 Service Attribute。

SDP Server	SDP Client
PDU ID : SDP_ServiceSearchRequest Transaction ID : 55 Parameter Length : 8 UUID (16-bit) : 0x1001 MaximumServiceRecordCount : 100 ContinuationState : 0 RecordSearch	PDU ID : SDP_ServiceSearchResponse Transaction ID : 55 Parameter Length : 17 TotalServiceRecordCount : 3 CurrentServiceRecordCount : 3 Handle 1 : 0x00000000 Handle 2 : 0x12345678 Handle 3 : 0x25436543 ContinuationState : 64

圖 5.1

－ 在此 SDP Server 會根據 SDP Client 所要搜尋之某一 ServiceRecordHandle 及要找之 AttributeID 到資料庫中去搜尋。並將結果回傳 SDP Client。

－ SDP Client 在回傳之結果中可得到所要找的 ServiceRecord 之 AttributID 內容。在此結果中是對 ServiceRecordHandle 為 0x25436543 之 ServiceRecord 搜尋其 0x0005 及 0x0200 的 AttributeID。因此在結果中可得知在 0x0005 及 0x0200 中的內容為 0x00001002 及 0x01029999。

SDP Server	SDP Client
PDU ID : SDP_ServiceAttributeRequest Transaction ID : 56 Parameter Length : 15 Request Handle : 0x25436543 MaximumAttributeByteCount : 100 Unsigned Integer (2 Bytes) : 0x0005 Unsigned Integer (2 Bytes) : 0x0200 ContinuationState : 0 RecordSearch	PDU ID : SDP_ServiceAttributeResponse Transaction ID : 56 Parameter Length : 23 AttributeListByteCount : 20 Unsigned Integer (2 Bytes) : 0x0005 UUID (32-bit) : 0x00001002 Unsigned Integer (2 Bytes) : 0x0200 UUID (32-bit) : 0x01029999 ContinuationState : 64

圖 5.2

1.2 接下來展示 Browsing 部份的實驗結果：

－ 第一次執行 Browsing 動作後，SDP Server 將所有 GroupDescriptor 之資料傳回 SDP Clent 。在結果中可得知有 SDP 、 TCS-Bin 及 L2CAP 三個 Group。

－ 第二次執行 Browsing 動作並在發出之封包中加入 SDP 的 GroupID 後所得到之結果。SDP 中可得知有 SDAP 這個 Service。

－ 第三次執行 Browsing 動作並在發出之封包中加入 TCS-Bin 的 GroupID 後所得到之結果。TCS-Bin 中可得知有 Cordless 及 Intercom 兩個 Service。

－ 第四次執行 Browsing 動作並在發出之封包中加入 L2CAP 的 GroupID 後所得到之結果。由於 L2CAP 尚未提供 Service，因此在 L2CAP 之 Group 下無資料顯示。

第一次

```
==============
PublicBrowseRoot
  +SDP
  +TCS-Bin
  +L2CAP
==============
```

第二次

```
==============
PublicBrowseRoot
  +SDP
    +SDAP
  +TCS-Bin
  +L2CAP
==============
```

第三次

```
================
PublicBrowseRoot
  +SDP
    +SDAP
  +TCS-Bin
    +Cordless
    +Intercom
  +L2CAP
================
```

第四次

```
==============
PublicBrowseRoot
  +SDP
    +SDAP
  +TCS-Bin
    +Cordless
    +Intercom
  +L2CAP
==============
```

圖 5.3

1.3 在 TCS-Bin 部份，我們先展示 Group Management 實驗結果：

－ 圖 5.4 為加入兩台 WUG member 後在 WUG master 所儲存之資料, 而此資料也將藉由送出 INFO SUGGEST 之訊息時讓所有 WUG member 將其儲存關於 WUG 之資料進行更

新。當 WUG membser 完成資料更新後會收到 INFO ACCEPT 訊息。當 WUG member 收到 INFO SUGGEST 時會將儲存關於 WUG 之資料進行更新完成後會送出 INFO ACCEPT 訊息給 WUG master。

```
CONFIGURATION_DISTRIBUTION for No.1
**********************************************************************
GW_BD_ADDR No.1 : 00:D0:B7:03:48:ED  Internal number : 0000
----------------------------------------------------------------------
TL_BD_ADDR No.1 : 00:D0:B7:03:48:F5  Internal number : 4671
TL_BD_ADDR No.2 : 00:D0:B7:03:49:23  Internal number : 3119
TL_BD_ADDR No.3 : 00:00:00:00:00:00  Internal number : 0000
TL_BD_ADDR No.4 : 00:00:00:00:00:00  Internal number : 0000
TL_BD_ADDR No.5 : 00:00:00:00:00:00  Internal number : 0000
TL_BD_ADDR No.6 : 00:00:00:00:00:00  Internal number : 0000
TL_BD_ADDR No.7 : 00:00:00:00:00:00  Internal number : 0000
**********************************************************************

Bluetooth_TCS_Group_management : INFO_ACCEPT
```

圖 5.4

1.4 接下來是 CC 部份的實驗結果：

發話端 (outgoing side)

－ 傳送 SETUP 訊息給受話端 CC (Call Control) 狀態為 call initialed 狀態。

```
send TCS message:SETUP
_____________The State of Call Control_____________
call initialed State.
```

－ 收到 ALERTING 訊息。CC (Call Control) 狀態為 call delivered 狀態。

```
Bluetooth_TCS_Call_Control : ALERTING
_____________The State of Call Control_____________
call delivered state.
```

－ 建立 SCO 連線。

```
establkish sco link.
tcs_add_sco_link():SCO conhandle = 2
```

－ 收到 CONNECT 訊息，傳送 CONNECT ACKNOWLEDGE 訊息。CC (Call Control) 狀態為 active 狀態，之後即可通話。

```
Bluetooth_TCS_Call_Control : CONNECT
send TCS message:CONNECT_ACKNOWLEDGE
_____________The State of Call Control_____________
active state.
```

受話端 (incoming side)

－ 受話端收到 TCS 的 SETUP 訊息，改變 CC (Call Control) 狀態為 call present 狀態，而開始電話設備響鈴。傳送 ALERTING 訊息，之後 cc (Call Control) 狀態為 call received 狀態，等待使用者拿起電話 (接受電話) 。

```
Bluetooth_TCS_Call_Control : SETUP
_____________The State of Call Control_____________
call present state.
_____________________________________________________
ring...

send TCS message:ALERTING
_____________The State of Call Control_____________
call received state.
```

－ 發話端建立起 SCO 連線。

```
CONNECTION_REQUEST for SCO LINKbt_write_lower_driver
tcs_add_sco_link():SCO conhandle = 2
```

－ 假使使用者拿起電話，CC (Call Control)會傳送 connect 訊息給發話端。 CC (Call Control) 狀態為 connection request 狀態。

```
send TCS message:connect
_____________The State of Call Control_____________
connection request state.
```

－ 收到 CONNECT_ACKNOWLEDGE 訊息，CC (Call Control) 狀態為 active 狀態，開始通話。

```
Bluetooth_TCS_Call_Control : CONNECT_ACKNOWLEDGE
_____________The State of Call Control_____________
active state.
```

1.5 L2CAP 部份

L2CAP 部分程式執行的畫面

－ TL 端:建立 Logical Channel

```
_____________________ CLOSED STATE _____________________________
L2CA_Connect_Req(): PSM= 1:BD_ADDR= 0:D0:B7: 3:48:ED
L2CAP_Connect_Req(): Local CID:40, PSM = 1

_____________________ W4_L2CA_CONNECT_RSP ____________________
L2CAP_Connect_Rsp():LCID=40:RCID=40:RESULT=0:STATUS=0
L2CA_Connect_Cfm():LCID=40 Result=0 Status=0
```

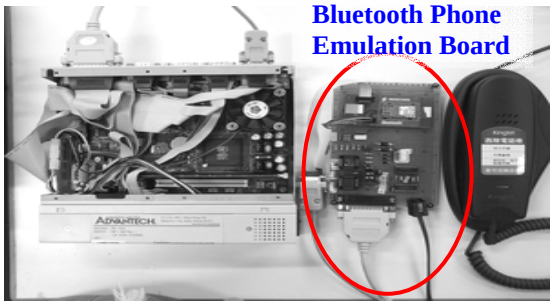
－ GW 端: 建立 Logical Channel

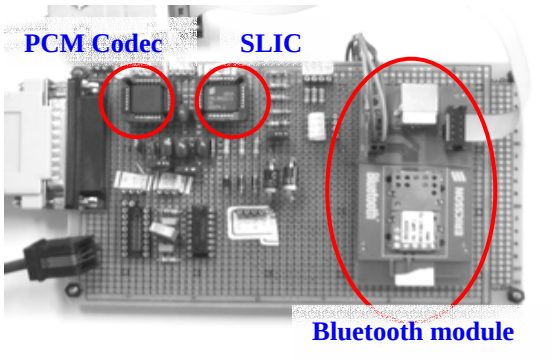
```
_____________________CLOSED STATE_____________________________
L2CAP_Connect_Req():PSM=1,RCID=40
L2CA_Connect_Ind():CID=40,PSM=1,Identifier=1,BD_ADDR=0:D0:B7: 3:48:F5

_____________________W4_L2CA_CONNECT_RSP_____________________
L2CA_Connect_Rsp():LCID=40,Response=0,Status=0
L2CAP_Connect_Rsp():RCID=40,LCID=40,Result=0,status = 0
```

由以上結果我們可以確認 SDP、TCS、L2CAP 等協定堆疊均執行無誤。

2 硬體電路及測試結果





在 Telephone Application 藉由 SDP、TCS、L2CAP 等協定建立 SCO 連線後，兩台藍芽無線電話分機藉由 Phone Emulation Board，將類比語音轉為數位 PCM 訊號經由 Bluetooth module 作溝通。

我們使用 Bluetooth module 所提供的 PCM clock 及 sync，作為 PCM codec 的所需之 BitClock (BCLK) 及 FrameSync (FS)。測試結果，PCM codec 所編成的 PCM 數位語音訊號成功地由 Bluetooth module 接收，並透過 SCO 連線傳送到另一個 Bluetooth module 後解回類比語音訊號。我們實地對著話筒講話，能夠在另一端的聽筒清楚地聽到語音，這證明了我們的硬體設計正確且運作正常，整個軟硬體系統能夠順利配合。

六、結論

我們的 "藍芽無線電話系統" 完全實現 Bluetooth Telephony 功能，此項應用結合了 Telecommunication、 Data communication、Embedded System、及語音與數據整合等多項技術。

在 Bluetooth 協定堆疊中，我們同時發展了核心技術中最主要的 TCS (Telephony Control Protocol Specification) 協定、SDP (Service Discovery Protocol) 協定、SDAP (Service Discovery Application Profile)、以及 L2CAP (Logic Link Control and Adaptation Protocol) 協定。另外，為了彌補 Bluetooth 規格未定義詳盡的地方，我們在系統及程式發展中創造了新的輔助函式，使系統程式能完整正常的運作。

我們的藍芽無線電話系統使辦公室分機可以隨身攜帶，而隨著使用者的移動加入不同 Bluetooth Gateway 的群組中，因此能夠達成分機的可攜性，不會發生有線分機系統中常找不到人的情況。除了一般無線電話的應用之外，藍芽無線電話系統也支援無線子機之間互相通話的對講機功能。

在技術創新部分，我們以更簡潔且更為快速的程式碼實現了複雜的 Telecommunication Signaling 功能；在程式語法上，我們比傳統實現 Q.931 Finite State Machine 的方法更為精簡，並達成相同的目的；我們創造新的輔助程式，使 Bluetooth protocol 不足之處更為完備；SDP 部分，我們自建符合 SDP 環境的資料庫與搜尋方式，讓 SDP 讀寫、搜尋資料庫能夠更為快速；專題中的軟體部份，我們將 Bluetooth 協定堆疊以 Linux Kernel Module 的方式實現在 Linux OS 上，並把成果整合進 Embedded Linux Kernel 中，可以隨時依需要抽換；而硬體部份，在資源有限的情形下，我們運用 PCM Codec 晶片、SLIC (Subscriber Line Interface Circuit) 晶片來完成 Phone Emulation Circuit，使用一般的市售電話來實現無線手機雛形，並整合 Bluetooth Module 於其上，這種方式同時也讓一般市售電話機只要接上了我們的藍芽無線通訊裝置，就能馬上具有無線電話與對講機之功能；我們在 Ericsson Bluetooth Module 功能不全，不能建立 Piconet 的情形下，設計新的方法完成 WUG Group 建立與 Fast Inter-Member Access。在應用創新部分，透過 Bluetooth Gateway 先進的 Signaling 程序，我們可以建立起多對通話端，可以選擇同時獨立通話而沒有數量的限制，這是目前家用無線電話無法做到的，這些都是我們專題的創新。

只要一機在手，使用者將擁有功能強大的電話、存取使用所有資訊家電的中樞、以及小型語音結合資料處理裝置。我們希望藉由開發本專題中的系統，能厚植我國在 Bluetooth 技術研發的能力，並協助本國廠商提升其競爭力。

七、參考文獻

[1] Bluetooth Core Protocol Specification 1.0b
[2] Bluetooth Profile Specification 1.0b
[3] International Telecommunication Union, "ITU-T Recommendation Q.931"