KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN ĐIỆN TỬ - VIỄN THÔNG

ĐOÀN NGỌC PHƯƠNG

BÀI GIẢNG:
LẬP TRÌNH TRÊN ĐIỆN THOẠI DI ĐỘNG

TẬP BÀI GIẢNG
(Lưu hành nội bộ)

THÁI NGUYÊN THÁNG 08/NĂM 2010

CHƯƠNG 2: CÁC CÔNG NGHỆ SỬ DỤNG TRÊN THIẾT BỊ DI ĐỘNG

2.1 Công nghệ mạng thông tin di động GSM

2.1.1 Quá trình phát triển của mạng thông tin di động GSM

Những năm đầu 1980, hệ thống viễn thông tế bào trên thế giới phát triển mạnh mẽ đặc biệt là ở Châu Âu mà không được chuẩn hóa về các chỉ tiêu kỹ thuật. Điều này đã thúc giục Liên minh Châu Âu về Bưu chính viễn thông CEPT (Conference of European Posts and Telecommunications) thành lập nhóm đặc trách về di động GSM (Groupe Spécial Mobile) với nhiệm vụ phát triển một chuẩn thống nhất cho hệ thống thông tin di động để có thể sử dụng trên toàn Châu Âu.

Ngày 27 tháng 3 năm 1991, cuộc gọi đầu tiên sử dụng công nghệ GSM được thực hiện bởi mạng Radiolinja ở Phần Lan (mạng di động GSM đầu tiên trên thế giới).

Năm 1989, Viện tiêu chuẩn viễn thông Châu Âu ETSI (European Telecommunications Standards Institute) quy định chuẩn GSM là một tiêu chuẩn chung cho mạng thông tin di động toàn Châu Âu, và năm 1990 chỉ tiêu kỹ thuật GSM phase I (giai đoạn I) được công bố.

Năm 1992, Telstra Australia là mạng đầu tiên ngoài Châu Âu ký vào biên bản ghi nhớ GSM MoU (Memorandum of Understanding). Cũng trong năm này, thỏa thuận chuyển vùng quốc tế đầu tiên được ký kết giữa hai mạng Finland Telecom của Phần Lan và Vodafone của Anh. Tin nhắn SMS đầu tiên cũng được gửi đi trong năm 1992.

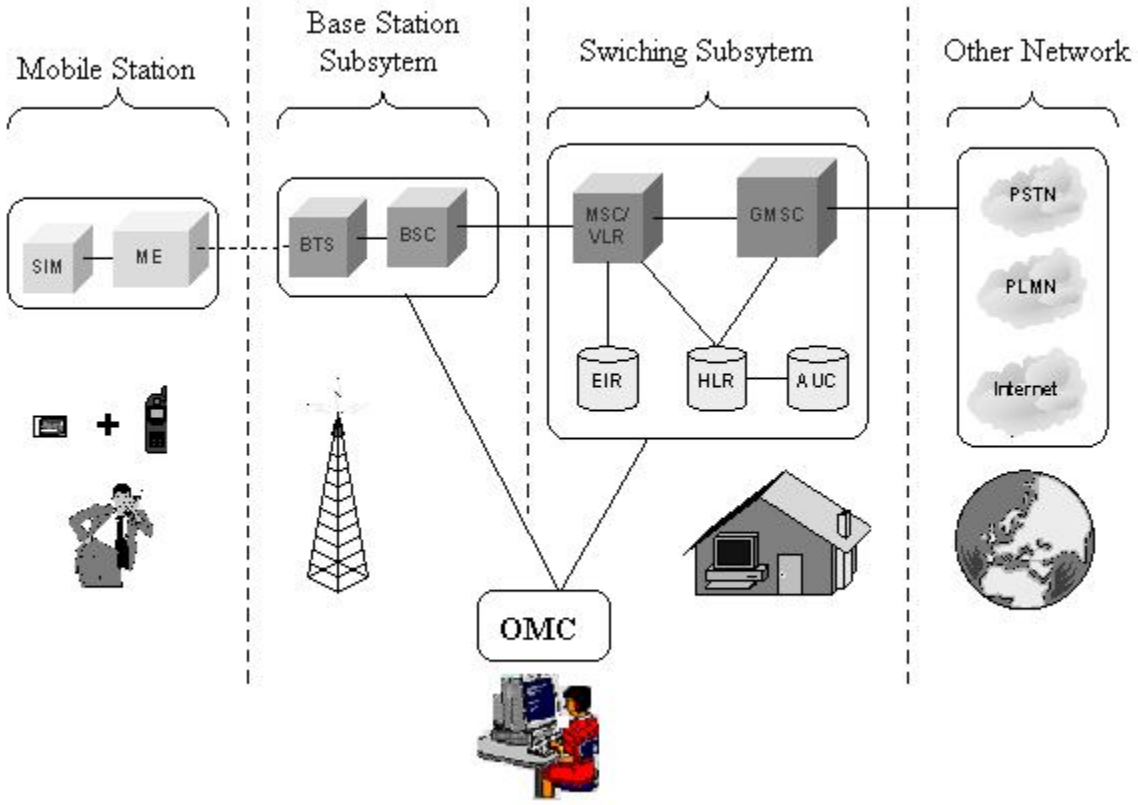
Những năm sau đó, hệ thống thông tin di động toàn cầu GSM phát triển một cách mạnh mẽ, cùng với sự gia tăng nhanh chóng của các nhà điều hành, các mạng di động mới, thì số lượng các thuê bao cũng gia tăng một cách chóng mặt.

Năm 1996, số thành viên GSM MoU đã lên tới 200 nhà điều hành từ gần 100 quốc gia. 167 mạng hoạt động trên 94 quốc gia với số thuê bao đạt 50 triệu.

Năm 2000, GPRS được ứng dụng. Năm 2001, mạng 3GSM (UMTS) được đi vào hoạt động, số thuê bao GSM đã vượt quá 500 triệu. Năm 2003, mạng EDGE đi vào hoạt động.

Cho đến năm 2006 số thuê bao di động GSM đã lên tới con số 2 tỉ với trên 700 nhà điều hành, chiếm gần 80% thị phần thông tin di động trên thế giới.

2.1.2. Kiến trúc tổng quát



Hình 2-1 Mô hình hệ thống thông tin di động GSM

Các ký hiệu:

OSS	: Phân hệ khai thác và hỗ trợ	BTS	: Trạm vô tuyến gốc
AUC	: Trung tâm nhận thực	MS	: Trạm di động
HLR	: Bộ ghi định vị thường trú	ISDN	: Mạng số liên kết đa dịch vụ
MSC	: Tổng đài di động	PSTN (Public Switched Telephone Network):	
BSS	: Phân hệ trạm gốc	Mạng chuyển mạch điện thoại công cộng	
BSC	: Bộ điều khiển trạm gốc	PSPDN	: Mạng chuyển mạch gói công cộng
OMC	: Trung tâm khai thác và bảo dưỡng	CSPDN (Circuit Switched Public Data Network):	
SS	: Phân hệ chuyển mạch	Mạng số liệu chuyển mạch kênh công cộng	
VLR	: Bộ ghi định vị tạm trú	PLMN	: Mạng di động mặt đất công cộng
EIR	: Thanh ghi nhận dạng thiết bị		

Các thành phần chức năng trong hệ thống

Mạng thông tin di động công cộng mặt đất PLMN (Public Land Mobile Network) theo chuẩn GSM được chia thành 4 phân hệ chính sau:

- Trạm di động MS (Mobile Station)
- Phân hệ trạm gốc BSS (Base Station Subsystem)
- Phân hệ chuyển mạch SS (Switching Subsystem)

- Phân hệ khai thác và hỗ trợ (Operation and Support Subsystem)

Trạm di động (MS - Mobile Station)

Trạm di động (MS) bao gồm thiết bị trạm di động ME (Mobile Equipment) và một khối nhỏ gọi là mođun nhận dạng thuê bao (SIM-Subscriber Identity Module). Đó là một khối vật lý tách riêng, chẳng hạn là một IC Card hoặc còn gọi là card thông minh. SIM cùng với thiết bị trạm (ME-Mobile Equipment) hợp thành trạm di động MS. SIM cung cấp khả năng di động cá nhân, vì thế người sử dụng có thể lắp SIM vào bất cứ máy điện thoại di động GSM nào truy nhập vào dịch vụ đã đăng ký. Mỗi điện thoại di động được phân biệt bởi một số nhận dạng điện thoại di động IMEI (International Mobile Equipment Identity). Card SIM chứa một số nhận dạng thuê bao di động IMSI (International Subcriber Identity) để hệ thống nhận dạng thuê bao, một mật mã để xác thực và các thông tin khác. IMEI và IMSI hoàn toàn độc lập với nhau để đảm bảo tính di động cá nhân. Card SIM có thể chống việc sử dụng trái phép bằng mật khẩu hoặc số nhận dạng cá nhân (PIN).

Trạm di động ở GSM thực hiện hai chức năng:

– Thiết bị vật lý để giao tiếp giữa thuê bao di động với mạng qua đường vô tuyến.

– Đăng ký thuê bao, ở chức năng thứ hai này mỗi thuê bao phải có một thẻ gọi là SIM card. Trừ một số trường hợp đặc biệt như gọi cấp cứu… thuê bao chỉ có thể truy nhập vào hệ thống khi cắm thẻ này vào máy.

Phân hệ trạm gốc (BSS - Base Station Subsystem)

BSS giao diện trực tiếp với các trạm di động MS bằng thiết bị BTS thông qua giao diện vô tuyến. Mặt khác BSS thực hiện giao diện với các tổng đài ở phân hệ chuyển mạch SS. Tóm lại, BSS thực hiện đấu nối các MS với tổng đài và nhờ vậy đấu nối những người sử dụng các trạm di động với những người sử dụng viễn thông khác. BSS cũng phải được điều khiển, do đó nó được đấu nối với phân hệ vận hành và bảo dưỡng OSS. Phân hệ trạm gốc BSS bao gồm:

- TRAU (Transcoding and Rate Adapter Unit): Bộ chuyển đổi mã và phối hợp tốc độ.
- BSC (Base Station Controler): Bộ điều khiển trạm gốc.

- BTS (Base Transceiver Station): Trạm thu phát gốc.

Khối BTS (Base Tranceiver Station):

Một BTS bao gồm các thiết bị thu /phát tín hiệu sóng vô tuyến, anten và bộ phận mã hóa và giải mã giao tiếp với BSC. BTS là thiết bị trung gian giữa mạng GSM và thiết bị thuê bao MS, trao đổi thông tin với MS qua giao diện vô tuyến. Mỗi BTS tạo ra một hay một số khu vực vùng phủ sóng nhất định gọi là tế bào (cell).

Khối TRAU (Transcode/Rate Adapter Unit):

Khối thích ứng và chuyển đổi mã thực hiện chuyển đổi mã thông tin từ các kênh vô tuyến (16 Kb/s) theo tiêu chuẩn GSM thành các kênh thoại chuẩn (64 Kb/s) trước khi chuyển đến tổng đài. TRAU là thiết bị mà ở đó quá trình mã hoá và giải mã tiếng đặc thù riêng cho GSM được tiến hành, tại đây cũng thực hiện thích ứng tốc độ trong trường hợp truyền số liệu. TRAU là một bộ phận của BTS, nhưng cũng có thể được đặt cách xa BTS và thậm chí còn đặt trong BSC và MSC.

Khối BSC (Base Station Controller):

BSC có nhiệm vụ quản lý tất cả giao diện vô tuyến thông qua các lệnh điều khiển từ xa. Các lệnh này chủ yếu là lệnh ấn định, giải phóng kênh vô tuyến và chuyển giao. Một phía BSC được nối với BTS, còn phía kia nối với MSC của phân hệ chuyển mạch SS. Giao diện giữa BSC và MSC là giao diện A, còn giao diện giữa BTS và BSC là giao diện A.bis.

Các chức năng chính của BSC:

1. Quản lý mạng vô tuyến: Việc quản lý vô tuyến chính là quản lý các cell và các kênh logic của chúng. Các số liệu quản lý đều được đưa về BSC để đo đạc và xử lý, chẳng hạn như lưu lượng thông tin ở một cell, môi trường vô tuyến, số lượng cuộc gọi bị mất, các lần chuyển giao thành công và thất bại...
2. Quản lý trạm vô tuyến gốc BTS: Trước khi đưa vào khai thác, BSC lập cấu hình của BTS (số máy thu/phát TRX, tần số cho mỗi trạm...). Nhờ đó mà BSC có sẵn một tập các kênh vô tuyến dành cho điều khiển và nối thông cuộc gọi.

3. Điều khiển nối thông các cuộc gọi: BSC chịu trách nhiệm thiết lập và giải phóng các đấu nối tới máy di động MS. Trong quá trình gọi, sự đấu nối được BSC giám sát. Cường độ tín hiệu, chất lượng cuộc đấu nối được ở máy di động và TRX gửi đến BSC. Dựa vào đó mà BSC sẽ quyết định công suất phát tốt nhất của MS và TRX để giảm nhiễu và tăng chất lượng cuộc đấu nối. BSC cũng điều khiển quá trình chuyển giao nhờ các kết quả đo kể trên để quyết định chuyển giao MS sang cell khác, nhằm đạt được chất lượng cuộc gọi tốt hơn. Trong trường hợp chuyển giao sang cell của một BSC khác thì nó phải nhờ sự trợ giúp của MSC. Bên cạnh đó, BSC cũng có thể điều khiển chuyển giao giữa các kênh trong một cell hoặc từ cell này sang kênh của cell khác trong trường hợp cell này bị nghẽn nhiều.

4. Quản lý mạng truyền dẫn: BSC có chức năng quản lý cấu hình các đường truyền dẫn tới MSC và BTS để đảm bảo chất lượng thông tin. Trong trường hợp có sự cố một tuyến nào đó, nó sẽ tự động điều khiển tới một tuyến dự phòng.

Phân hệ chuyển mạch (SS - Switching Subsystem)

Phân hệ chuyển mạch bao gồm các khối chức năng sau:

- Trung tâm chuyển mạch nghiệp vụ di động MSC
- Thanh ghi định vị thường trú HLR
- Thanh ghi định vị tạm trú VLR
- Trung tâm nhận thực AuC
- Thanh ghi nhận dạng thiết bị EIR

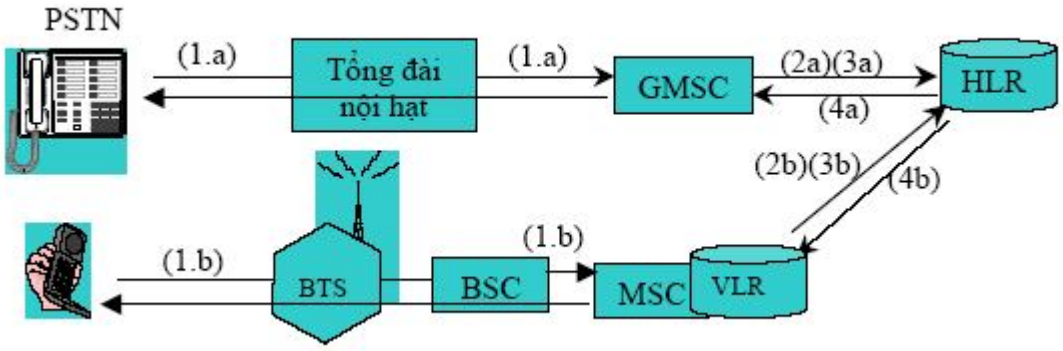
Phân hệ chuyển mạch (SS) bao gồm các chức năng chuyển mạch chính của mạng GSM cũng như các cơ sở dữ liệu cần thiết cho số liệu thuê bao và quản lý di động của thuê bao. Chức năng chính của SS là quản lý thông tin giữa những người sử dụng mạng GSM với nhau và với mạng khác.

Trung tâm chuyển mạch di động MSC:

Tổng đài di động MSC (Mobile services Switching Center) thường là một tổng đài lớn điều khiển và quản lý một số các bộ điều khiển trạm gốc BSC. MSC thực hiện các chức năng chuyển mạch chính, nhiệm vụ chính của MSC là tạo kết nối và xử lý cuộc gọi đến những thuê bao của GSM, một mặt MSC giao tiếp với phân hệ BSS và mặt khác giao tiếp với mạng ngoài qua tổng đài cổng GMSC (Gateway MSC).

Chức năng chính của tổng đài MSC:

- Xử lý cuộc gọi (Call Processing)
- Điều khiển chuyển giao (Handover Control)
- Quản lý di động (Mobility Management)
- Tương tác mạng IWF(Interworking Function): qua GMSC



Hình 2-2 Chức năng xử lý cuộc gọi của MSC

(1): Khi chủ gọi quay số thuê bao di động bị gọi, số mạng dịch vụ số liên kết của thuê bao di động, sẽ có hai trường hợp xảy ra :

- (1.a) – Nếu cuộc gọi khởi đầu từ mạng cố định PSTN thì tổng đài sau khi phân tích số thoại sẽ biết đây là cuộc gọi cho một thuê bao di động. Cuộc gọi sẽ được định tuyến đến tổng đài cổng GMSC gần nhất.
- (1.b) – Nếu cuộc gọi khởi đầu từ trạm di động, MSC phụ trách ô mà trạm di động trực thuộc sẽ nhận được bản tin thiết lập cuộc gọi từ MS thông qua BTS có chứa số thoại của thuê bao di động bị gọi.

(2): MSC (hay GMSC) sẽ phân tích số MSISDN (The Mobile Station ISDN) của thuê bao bị gọi để tìm ra HLR nơi MS đăng ký.

(3): MSC (hay GMSC) sẽ hỏi HLR thông tin để có thể định tuyến đến MSC/VLR quản lý MS.

(4): HLR sẽ trả lời, khi đó MSC (hay GMSC) này có thể định tuyến lại cuộc gọi đến MSC cần thiết. Khi cuộc gọi đến MSC này, VLR sẽ biết chi tiết hơn về vị trí của MS.

Như vậy có thể nối thông một cuộc gọi ở mạng GSM, đó là chức năng xử lý cuộc gọi của MSC.

Để kết nối MSC với một số mạng khác cần phải thích ứng các đặc điểm truyền dẫn của mạng GSM với các mạng này. Các thích ứng này gọi là chức năng tương tác IWF (Inter Networking Function). IWF bao gồm một thiết bị để thích ứng giao thức và truyền dẫn. IWF có thể thực hiện trong cùng chức năng MSC hay có thể ở thiết bị riêng, ở trường hợp hai giao tiếp giữa MSC và IWF được để mở.

Bộ ghi định vị thường trú (HLR - Home Location Register):

HLR là cơ sở dữ liệu tham chiếu lưu giữ lâu dài các thông tin về thuê bao, các thông tin liên quan tới việc cung cấp các dịch vụ viễn thông. HLR không phụ thuộc vào vị trí hiện thời của thuê bao và chứa các thông tin về vị trí hiện thời của thuê bao.

HLR bao gồm:

- Các số nhận dạng: IMSI, MSISDN.
- Các thông tin về thuê bao
- Danh sách các dịch vụ mà MS được sử dụng và bị hạn chế
- Số hiệu VLR đang phục vụ MS

Bộ ghi định vị tạm trú (VLR - Visitor Location Register):

VLR là một cơ sở dữ liệu chứa thông tin về tất cả các MS hiện đang ở vùng phục vụ của MSC. Mỗi MSC có một VLR, thường thiết kế VLR ngay trong MSC. Ngay cả khi MS lưu động vào một vùng MSC mới. VLR liên kết với MSC sẽ yêu cầu số liệu về MS từ HLR. Đồng thời HLR sẽ được thông báo rằng MS đang ở vùng MSC nào. Nếu sau đó MS muốn thực hiện một cuộc gọi, VLR sẽ có tất cả các thông tin cần thiết để thiết lập một cuộc gọi mà không cần hỏi HLR, có thể coi VLR như một HLR phân bố. VLR chứa thông tin chính xác hơn về vị trí MS ở vùng MSC. Nhưng khi thuê bao tắt máy hay rời khỏi vùng phục vụ của MSC thì các số liệu liên quan tới nó cũng hết giá trị.

Hay nói cách khác, VLR là cơ sở dữ liệu trung gian lưu trữ tạm thời thông tin về thuê bao trong vùng phục vụ MSC/VLR được tham chiếu từ cơ sở dữ liệu HLR.

VLR bao gồm:

- Các số nhận dạng: IMSI, MSISDN, TMSI.
- Số hiệu nhận dạng vùng định vị đang phục vụ MS
- Danh sách các dịch vụ mà MS được và bị hạn chế sử dụng
- Trạng thái của MS (bận: busy; rỗi: idle)

Thanh ghi nhận dạng thiết bị (EIR - Equipment Identity Register):

EIR có chức năng kiểm tra tính hợp lệ của ME thông qua số liệu nhận dạng di động quốc tế (IMEI-International Mobile Equipment Identity) và chứa các số liệu về phần cứng của thiết bị. Một ME sẽ có số IMEI thuộc một trong ba danh sách sau:

1. Nếu ME thuộc danh sách trắng (White List) thì nó được quyền truy nhập và sử dụng các dịch vụ đã đăng ký.
2. Nếu ME thuộc danh sách xám (Gray List), tức là có nghi vấn và cần kiểm tra. Danh sách xám bao gồm những ME có lỗi (lỗi phần mềm hay lỗi sản xuất thiết bị) nhưng không nghiêm trọng tới mức loại trừ khỏi hệ thống
3. Nếu ME thuộc danh sách đen (Black List), tức là bị cấm không cho truy nhập vào hệ thống, những ME đã thông báo mất máy.

Khối trung tâm nhận thực AuC (Aunthentication Center)

AuC được nối đến HLR, chức năng của AuC là cung cấp cho HLR các tần số nhận thực và các khoá mật mã để sử dụng cho bảo mật. Đường vô tuyến cũng được AuC cung cấp mã bảo mật để chống nghe trộm, mã này được thay đổi riêng biệt cho từng thuê bao. Cơ sở dữ liệu của AuC còn ghi nhiều thông tin cần thiết khác khi thuê bao đăng ký nhập mạng và được sử dụng để kiểm tra khi thuê bao yêu cầu cung cấp dịch vụ, tránh việc truy nhập mạng một cách trái phép.

Phân hệ khai thác và bảo dưỡng (OSS)

OSS (Operation and Support System) thực hiện 3 chức năng chính:

1) Khai thác và bảo dưỡng mạng.
2) Quản lý thuê bao và tính cước.
3) Quản lý thiết bị di động.

Khai thác và bảo dưỡng mạng:

- **Khai thác:**

Là hoạt động cho phép nhà khai thác mạng theo dõi hành vi của mạng như tải của hệ thống, mức độ chặn, số lượng chuyển giao giữa hai cell.v.v.. Nhờ vậy nhà khai thác có thể giám sát được toàn bộ chất lượng dịch vụ mà họ cung cấp cho khách hàng và kịp thời nâng cấp. Khai thác còn bao gồm việc thay đổi cấu hình để giảm những vẫn đề xuất hiện ở thời điểm hiện thời, để chuẩn bị tăng lưu lượng trong tương lai và mở rộng vùng phủ sóng. Ở hệ thống viễn thông hiện đại, khai thác được thực hiện bằng máy tính và được tập trung ở một trạm.

- **Bảo dưỡng:**

Có nhiệm vụ phát hiện, định vị và sửa chữa các sự cố và hỏng hóc, nó có một số quan hệ với khai thác. Các thiết bị ở hệ thống viễn thông hiện đại có khả năng tự phát hiện một số các sự cố hay dự báo sự cố thông qua kiểm tra. Bảo dưỡng bao gồm các hoạt động tại hiện trường nhằm thay thế các thiết bị có sự cố, cũng như việc sử dụng các phần mềm điều khiển từ xa.

Hệ thống khai thác và bảo dưỡng có thể được xây dựng trên nguyên lý của TMN (Telecommunication Management Network - Mạng quản lý viễn thông). Lúc này, một mặt hệ thống khai thác và bảo dưỡng được nối đến các phần tử của mạng viễn thông (MSC, HLR, VLR, BSC, và các phần tử mạng khác trừ BTS). Mặt khác hệ thống khai thác và bảo dưỡng được nối tới máy tính chủ đóng vai trò giao tiếp người - máy. Theo tiêu chuẩn GSM hệ thống này được gọi là trung tâm vận hành và bảo dưỡng (OMC - Operation and Maintenance Center).

Quản lý thuê bao:

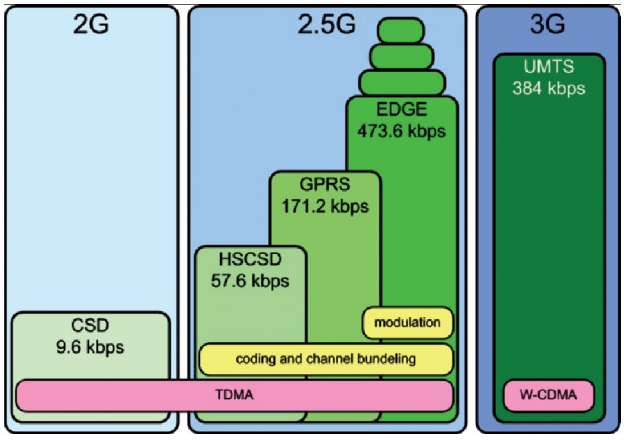
Bao gồm các hoạt động quản lý đăng ký thuê bao. Nhiệm vụ đầu tiên là nhập và xoá thuê bao khỏi mạng. Đăng ký thuê bao cũng có thể rất phức tạp, bao gồm nhiều dịch vụ và các tính năng bổ sung. Nhà khai thác có thể thâm nhập được các thông số nói trên. Một nhiệm vụ quan trọng khác của khai thác là tính cước các cuộc gọi rồi gửi đến thuê bao. Khi đó HLR, SIM-Card đóng vai trò như một bộ phận quản lý thuê bao.

Quản lý thiết bị di động:

Quản lý thiết bị di động được bộ đăng ký nhận dạng thiết bị EIR thực hiện. EIR lưu trữ toàn bộ dữ liệu liên quan đến trạm di động MS. EIR được nối đến MSC qua đường báo hiệu để kiểm tra tính hợp lệ của thiết bị. Trong hệ thống GSM thì EIR được coi là thuộc phân hệ chuyển mạch NSS.

2.1.4 Công nghệ GPRS

Để truyền thông tin số từ điện thoại di động đến mạng, mạch chuyển đổi dữ liệu CSD (Circuit Switched Data) được sử dụng trong thế hệ mạng di động 2G, CSD hỗ trợ tốc độ lên tới 9.6 kbps. GSM (Global System Mobile Communication) – hệ thống truyền thông di động toàn cầu là hệ thống đa truy nhập phân chia theo thời gian (TDMA), sử dụng khe thời gian để kết nối, mỗi khe đại diện cho một kênh người dùng. Để tăng tốc độ truyền dữ liệu trong hệ thống GSM, công nghệ HSCSD (High Speed Circuit Switched Data) - dữ liệu chuyển mạch tốc độ cao được sử dụng. HSCSD là tiêu chuẩn GSM cuối cùng để sử dụng chuyển mạch thay vì truyền dữ liệu chuyển mạch gói, khi sử dụng HSCSD một kết nối thường trực được thiết lập giữa các bên gọi và được gọi để trao đổi dữ liệu. Dịch vụ thông tin di động vô tuyến chuyển mạch gói GPRS (General Packet Radio Service) là sự mở rộng của CSD và HSCSD. GPRS tốt hơn HSCSD, nó được đưa ra ở thế hệ 2.5G và có thể coi là bước phát triển đến công nghệ 3G.



Hình :

Sự phát triển của hệ thống truyền thông cellular.

Khác với CSD và HSCSD là các công nghệ chuyển mạch kênh, GPRS là công nghệ chuyển mạch gói. Nói một cách đơn giản hơn, GPRS là dịch vụ truyền tải dữ liệu thông qua tín hiệu vô tuyến, được phát triển dựa trên nền tảng GSM. Ứng dụng chính của GPRS là hỗ trợ dịch vụ email và duyệt web, vì GPRS truyền tải dữ liệu giữa điện thoại di động với nhà cung cấp (sau đó nhà cung cấp truyền tải tiếp dữ liệu đó đến mạng internet) nên nếu máy được cài đặt các phần mềm/ứng dụng phù hợp, bạn sẽ kết nối được với thế giới internet bên ngoài. Để tăng cường tốc độ truyền dữ liệu cho GPRS, EDGE đã được phát triển, công nghệ EDGE hay còn gọi là EGPRS, là một công nghệ di động được nâng cấp từ GPRS cho phép truyền dữ liệu với tốc độ có thể lên đến 384 Kbps cho người dùng cố định hoặc di chuyển chậm và 144 Kbps cho người dùng di chuyển tốc độ cao. EDGE dùng phương thức điều chế 8-PSK để tăng tốc độ dữ liệu truyền. Chính vì thế, để triển khai EDGE, các nhà cung cấp mạng phải thay đổi trạm phát sóng BTS cũng như là thiết bị di động so với mạng GPRS.

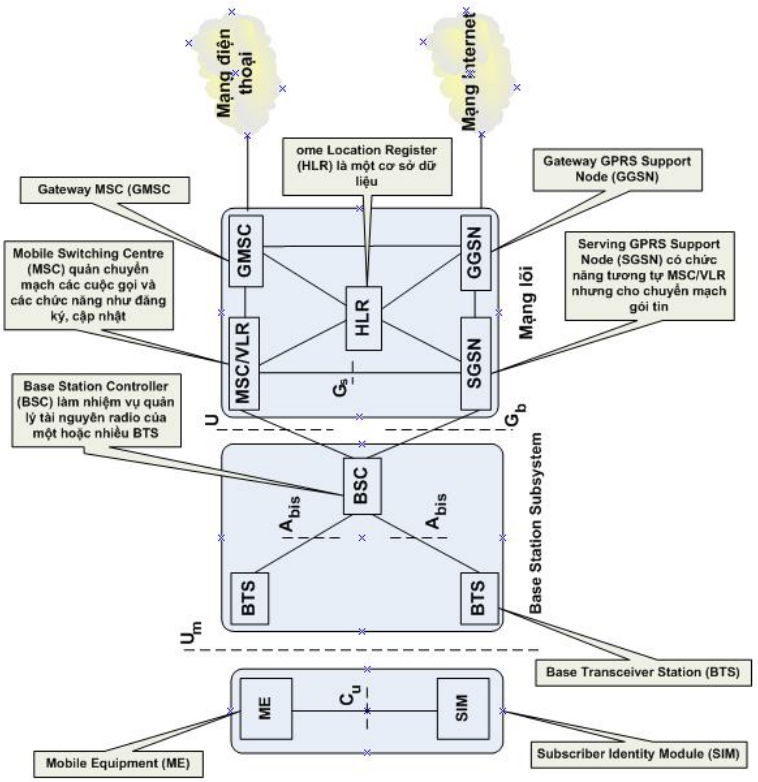
Các yếu tố chính của kiến trúc GPRS [3GPP-23.060] được thể hiện trong hình dưới Một trạm di động GPRS được phân loại theo khả năng của mình để hỗ trợ đồng thời các chế độ hoạt động cho mạng GSM và GPRS [3GPP-22.060] như sau:

- Class A: các trạm di động hỗ trợ sử dụng đồng thời dịch vụ GSM và GPRS (Đính kèm, kích hoạt, theo dõi, truyền tải, ...). Một lớp trạm di động có thể được thiết lập hoặc nhận cuộc gọi vào hai dịch vụ đồng thời. Sự phức tạp cao của việc thiết kế các lớp làm cho sản xuất thiết bị tốn kém. Đo đó, các thiết bị này thường không có sẵn cho thị trường.

- Class B: các trạm điện thoại di động được gắn vào cả hai dịch vụ GSM và GPRS. Tuy nhiên, trạm di động chỉ có thể hoạt động một trong hai dịch vụ tại một thời điểm.

- Loại C: các trạm điện thoại di động được gắn với dịch vụ hoặc dịch vụ GSM hoặc GPRS nhưng không thuộc cả hai dịch vụ đồng thời. Trước khi thành lập hoặc nhận cuộc gọi trên một trong hai dịch vụ, trạm điện thoại di động đã được gắn liền với dịch vụ mong muốn.

Trước khi một trạm di động có thể truy cập dịch vụ GPRS, nó phải thực thi một thủ tục đính kèm GPRS để cho biết sự hiện diện của nó vào mạng. Sau khi đính kèm GPRS của chính nó, các trạm di động kích hoạt một giao thức chuyển dữ liệu gói (PDP - Packet Data Protocol) phù hợp với mạng để có thể truyền hoặc nhận dữ liệu. Thủ tục này được gọi là kích hoạt ngữ cảnh PDP. Giao diện tầng không của GPRS giống hệt giao diện tầng không của mạng GSM (điều chế sóng radio, tần số băng thông và cấu trúc khung). GPRS dựa trên cơ sở phát một triển hệ thống con của GSM. Tuy nhiên, mạng lõi GPRS dựa trên một hệ thống mạng GSM trong đó được tích hợp bổ sung hai thành phần: phục vụ và các nút hỗ trợ cổng GPRS. Ngoài ra, dịch vụ EDGE (Enhanced Data Rate for Global Evolution) có thể được hỗ trợ nâng cao hiệu năng của GPRS.



Hình. Kiến trúc mạng GPRS

Nút hỗ trợ phục vụ GPRS

Node hỗ trợ phục vụ GPRS viết tắt là SGSN (Serving GPRS Support Node) được kết nối với một hoặc nhiều trạm con gốc. Nó hoạt động như một bộ định tuyến gói dữ liệu cho tất cả các trạm di động hiện hành trong một khu vực địa lý. Nó cũng theo dõi vị trí của trạm di động và thực hiện chức năng bảo mật và kiểm soát truy cập.

Nút hỗ trợ Gateway GPRS

Các nút hỗ trợ Gateway GPRS viết tắt là GGSN (Gateway GPRS Support Node) cung cấp các điểm gắn kết giữa miền GPRS với các mạng dữ liệu như Internet hoặc mạng PSTN. Một Access Point Name (APN) được sử dụng bởi người dùng di động để thiết lập kết nối đến các mạng đích được yêu cầu.

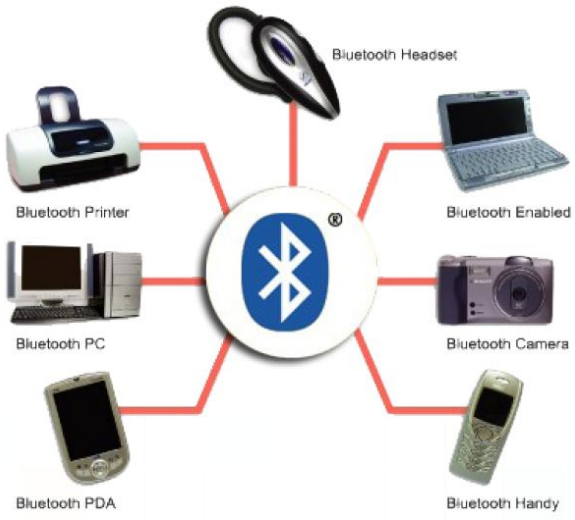
2.1.5. Công nghệ 3G

Thế hệ di động thứ 3 (3G): Mạng 3G đặc trưng bởi tốc độ dữ liệu cao, capacity của hệ thống lớn, tăng hiệu quả sử dụng phổ tần và nhiều cải tiến khác. Có một loạt các chuẩn công nghệ di động 3G, tất cả đều dựa trên CDMA, bao gồm: UMTS (dùng cả FDD lẫn TDD), CDMA2000 và TD-SCDMA.

Công nghệ UMTS (Universal Mobile Telecommunication System) được sử dụng, UMTS được chuẩn hóa bởi tổ chức 3GPP và được phát triển lên từ các nước sử dụng GSM. Để tăng tốc độ truyền dữ liệu cho mạng 3G, UMTS được nâng cấp lên với chuẩn HSPDA (High Speed Downlink Packet Access), gói đường truyền tốc độ cao, cho phép các mạng hoạt động trên hệ thống UMTS có khả năng truyền tải dữ liệu với tốc độ cao hơn hẳn. Công nghệ HSDPA hiện nay cho phép tốc độ download đạt đến 1.8, 3.6, 7.2 và 14.4 Mbps, và trong tương lai gần, tốc độ hiện nay có thể được nâng lên gấp nhiều lần. Song song với công nghệ HSDPA là công nghệ HSUPA (High Speed Uplink Packet Access Tương tư HSDPA), HSUPA cải thiện tốc độ tải dữ liệu lên, về lý thuyết tốc độ upload dữ liệu của công nghệ HSUPA có thể đạt đến 5.76Mbps. Cụm từ 3GPP LTE (The Third Generation Partnership Project Long Term Evolution) được dùng để để nói về một công nghệ di động mới đang được phát triển và chuẩn hóa bởi 3GPP. Với LTE tốc độ truyền dữ liệu có thể đạt tới 100Mbps với download và 50 Mbps với upload.

2.1 Công nghệ Bluetooth

Đặc tả Bluetooth được phát triển đầu tiên bởi Ericsson vào năm 1999 và sau đó được chuẩn hoá bởi Bluetooth Special Interest Group (SIG). Bluetooth là một đặc tả công nghiệp cho truyền thông không dây tầm gần giữa các thiết bị điện tử. Công nghệ này hỗ trợ việc truyền dữ liệu qua các khoảng cách ngắn giữa các thiết bị di động và cố định, tạo nên các mạng cá nhân không dây (Wireless Personal Area Network-PANs). Bluetooth cho phép kết nối và trao đổi thông tin giữa các thiết bị như điện thoại di động, điện thoại cố định, máy tính xách tay, PC, máy in, thiết bị định vị dùng GPS, máy ảnh số, và video game console. Bluetooth ngày càng được quan tâm nhiều hơn bởi nó cho phép người dùng được kết nối không dây miễn phí, tiêu hao ít năng lượng, giá thành thiết bị lại rẻ và công nghệ này được rất nhiều tập đoàn lớn hỗ trợ. Tuy nhiên sự phát triển của Bluetooth còn hạn chế do sự giới hạn của khoảng cách truyền, số lượng kết nối, khả năng chống nhiễu và khả năng bảo mật của nó. Trong tương lai Bluetooth sẽ trở nên mạnh hơn với công nghệ siêu băng rộng UWB (ultra wide-band). UWB cho phép tăng tốc độ truyền thông tầm gần lên tới 400Mbps.

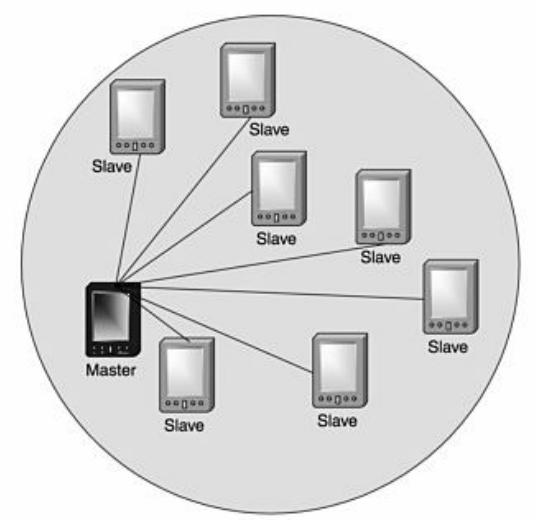


Hình:
Ứng dụng của bluetooth

Cấu trúc Bluetooth

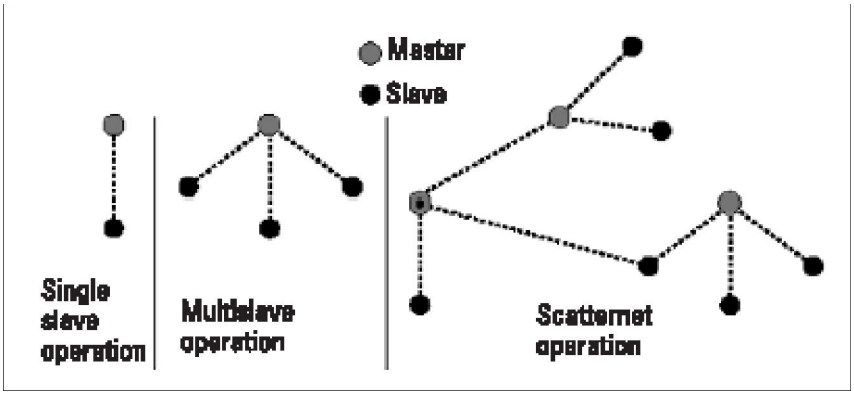
Bluetooth được thiết kế để sử dụng trong thông tin vô tuyến giữa hai hay nhiều trạm di động. Hệ thống cung cấp một kết nối điểm-điểm giữa hai trạm hoặc kết nối điểm-đa điểm tại nơi chia sẻ đường truyền của một vài trạm. Như thế chúng ta sẽ có một piconet (Picotnet là tập hợp các thiết bị được kết nối thông qua kỹ thuật Bluetooth theo mô hình Ad-Hoc) ở nơi có hai hay nhiều trạm cùng chia sẻ đường truyền.

Trong một piconet, một trạm sẽ đóng vai trò là chủ, những thành phần khác sẽ là tớ. Bất kỳ đơn vị Bluetooth nào (tất cả chúng đều giống nhau) đều có thể đóng một trong vai trò chủ, tớ khi có yêu cầu đòi hỏi. Trạm chủ được định nghĩa là nơi khởi tạo kết nối (tới một hay nhiều trạm tớ). Một piconet có thể là chủ và có tới 7 trạm tớ trong trạng thái hoạt động của nó. Trạng thái hoạt động có nghĩa là một trạm tớ đang giao tiếp với một trạm chủ; một trạm có thể trong trạng thái khoá (parked state) nếu nó được đồng bộ hoá tới trạm chủ, nhưng nó không hoạt động trên kênh. Cả trạm tích cực và trạm tam dừng đều được kiểm soát bởi trạm chủ.



Mô hình piconet

Một trạm tớ có thể được đồng bộ hoá với piconet khác: một trạm có thể là chủ trong piconet này nhưng lại là tớ trong một piconet khác. Theo cách này, nhiều piconet chồng lấn lên nhau (khi không đồng bộ về thời gian hoặc tần số) sẽ tạo thành một scatternet. Những đặc điểm khác nhau này được tổng hợp trong hình:



Hình : Các kiểu kết nối khác nhau giữa các trạm

Đặc điểm chính của hệ thống Bluetooth được đưa ra trong bảng bên dưới. Hệ thống Bluetooth sử dụng phương thức truy cập khe thời gian. Một gói tin có thể sử dụng trên 5 khe thời gian nhưng ít nhất phải dùng một khe. Hệ thống Bluetooth có thể truyền một kênh dữ liệu không đồng bộ, hơn ba kênh tiếng nói đồng thời hoặc một kênh hỗ trợ truyền đồng thời dữ liệu và tiếng nói không đồng bộ.

Spectrum	2.4 GHz
Maximum physical rate (symbol rate)	1 Mbps
Access method	TDMA/FDMA-TDD

Các loại kết nối khác nhau được hỗ trợ bởi Bluetooth là:

- Một kết nối đồng bộ tốc độ 64Kbps trong kết nối trực tiếp cho kênh tiếng nói;
- Tốc độ không cân đối tối đa cho một kết nối trực tiếp là 723.2Kbps (có thể tăng 57.6Kbps nữa theo phương truyền ngược lại) hoặc tốc độ 433.9Kbps cân đối cho đồng bộ liên kết.

Cách thức hoạt động của Bluetooth.

Cơ chế truyền và sửa lỗi

Kỹ thuật Bluetooth thực sự là rất phức tạp. Nó dùng kỹ thuật nhảy tần số trong các timeslot (TS), được thiết kế để làm việc trong môi trường nhiễu tần số radio, Bluetooth dùng chiến lược nhảy tần để tạo nên sức mạnh liên kết truyền thông và truyền thông thông minh. Cứ mỗi lần gửi hay nhận một packet xong, Bluetooth lại nhảy sang một tần số mới, như thế sẽ tránh được nhiễu từ các tín hiệu khác.

So sánh với các hệ thống khác làm việc trong cùng băng tần, sóng radio của Bluetooth nhảy tần nhanh và dùng packet ngắn hơn. Vì nhảy nhanh và packet ngắn sẽ làm giảm va chạm với sóng từ lò vi sóng và các phương tiện gây nhiễu khác trong khí quyển.

Có 3 phương pháp được sử dụng trong việc kiểm tra tính đúng đắn của dữ liệu truyền đi:

- Forwad Error Corrrection: thêm 1 số bit kiểm tra vào phần Header hay Payload của packet.
- Automatic Repeat Request: dữ liệu sẽ được truyền lại cho tới khi bên nhận gửi thông báo là đã nhận đúng.
- Cyclic Redundancy Check: mã CRC thêm vào các packet để kiểm chứng liệu Payload có đúng không.

Bluetooth dùng kỹ thuật sửa lỗi tiến FEC (Forward Error Correction) để sửa sai do nhiễu tự nhiên khi truyền khoảng cách xa. FEC cho phép phát hiện lỗi, biết sửa sai và truyền đi tiếp (khác với kỹ thuật BEC-Backward Error Control chỉ phát hiện, không biết sửa, yêu cầu truyền lại).

Giao thức băng tần cơ sở (Baseband) của Bluetooth là sự kết hợp giữa chuyển mạch và chuyển đổi packet. Các khe thời gian có thể được dành riêng cho các packet phục vụ đồng bộ. Thực hiện bước nhảy tần cho mỗi packet được truyền đi. Một packet

trên danh nghĩa sẽ chiếm 1 timeslot, nhưng nó có thể mở rộng chiếm đến 3 hay 5 timeslot.

Bluetooth hỗ trợ 1 kênh dữ liệu bất đồng bộ, hay 3 kênh tín hiệu thoại đồng bộ nhau cùng một lúc, hay 1 kênh hỗ trợ cùng lúc dữ liệu bất đồng bộ và tín hiệu đồng bộ.

Quá trình hình thành Piconet

Một Piconet được tạo bằng 4 cách:

- Có Master rồi, Master thực hiện Paging để kết nối với 1 Slave.
- Một Unit (Master hay Slave) lắng nghe tín hiệu (code) mà thiết bị của nó truy cập được.
- Khi có sự chuyển đổi vai trò giữa Master và Slave.
- Khi có một Unit chuyển sang trang thái Active

Để thiết lập một kết nối mới, tiến trình INQUIRY hay PAGE sẽ bắt đầu. Tiến trình Inquiry cho phép 1 Unit phát hiện các Unit khác trong tầm hoạt động cùng với địa chỉ và đồng hồ của chúng.

Tiến trình Paging mới thực sự là tạo kết nối. Kết nối chỉ thực hiện giữa những thiết bị mang địa chỉ Bluetooth. Unit nào thiết lập kết nối sẽ phải thực hiện tiến trình paging và tự động trở thành Master của kết nối.

Trong tiến trình paging, có thể áp dụng vài chiến lược paging. Có một chiến lược paging bắt buộc tất cả các thiết bị Bluetooth đều phải hỗ trợ, chiến lược dùng khi các Unit gặp trong lần đầu tiên, và trong trường hợp tiến trình paging theo ngay sau tiến trình inquiry. Hai Unit sau khi kết nối nhờ dùng chiến lược bắt buộc này, sau đó có thể chọn chiến lược paging khác.

Sau thủ tục Paging (PAGE), Master thăm dò Slave bằng cách gửi packet POLL

thăm dò hay packet NULL rỗng theo như Slave yêu cầu.

Chỉ có Master gửi tín hiệu POLL cho Slave, ngược lại không có.

Các vai trò của thiết bị trong Piconet là:

- Stand by : Không làm gì cả.
- Inquiry : Tìm thiết bị trong vùng lân cận.
- Paging :Kết nối với 1 thiết bị cụ thể.
- Connecting : Nhận nhiệm vụ.

Hình 2-14 Quá trình truy vấn tạo kết nối.

Mô hình truy vấn các thiết bị trong thực tế:

Hình 2-15 Truy vấn tạo kết nối giữa các thiết bị trong thực tế.

Khi thiết bị tạo paging muốn tạo các kết nối ở các tầng trên, nó sẽ gửi yêu cầu kết nối host theo nghi thức LMP (Link Manament Protocol). Khi Unit quản lý host này nhận được thông điệp, nó thông báo cho host biết về kết nối mới. Thiết bị từ xa có thể chấp nhận (gửi thông điệp chấp nhận theo nghi thức LMP) hoặc không chấp nhận kết nối (gửi thông điệp không chấp nhận theo nghi thức LMP).

Khi thiết bị không yêu cầu bất kỳ thủ tục thiết lập liên kết từ xa nào cả, nó sẽ gửi thông điệp "thiết lập hoàn thành". Thiết bị này vẫn nhận được yêu cầu từ các thiết bị khác. Khi một thiết bị khác đã sẵn sàng tạo liên kết, nó cũng gửi thông điệp "thiết lập hoàn thành". Sau đó 2 thiết bị có thể trao đổi packet trên kênh logic khác với LMP.

Quá trình hình thành Scatternet

Một Master hay Slave của Piconet này có thể thành Slave của Piconet khác nếu bị Master của piconet khác thực hiện tiến trình paging với nó. Có nghĩa là bất kỳ unit nào cũng có thể tạo 1 Piconet mới bằng cách paging một unit đã là thành viên của một Piconet nào đó. Ngược lại, bất kỳ unit nào tham gia trong 1 Piconet, đều có thể thực hiện paging

lên Master hay Slave của Piconet khác. Điều này có thể dẫn đến việc chuyển đổi vai trò giữa Master và Slave trong kết nối mới này.

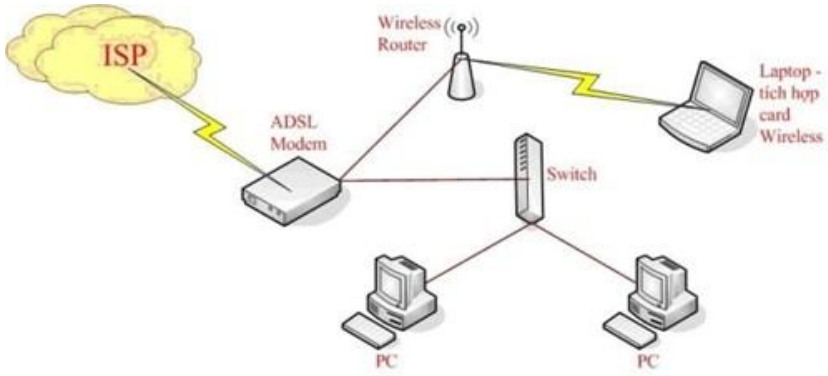
Hình 2-16 Minh hoạ một Scatternet.

Các kết nối bên trong một Piconet được thiết lập thông qua các unit chia sẻ, unit này thuộc về 2 hay nhiều Piconet, nó dùng kỹ thuật phân chia thời gian để chuyển đổi qua lại giữa các Piconet.

2.1 Công nghệ Wifi

WiFi là gì?

Wi-Fi viết tắt từ Wireless Fidelity hay mạng 802.11 là hệ thống mạng không dâysử dụng sóng vô tuyến, giống như điện thoại di động, truyền hình và radio.Hệ thống này đã hoạt động ở một số sân bay, quán café, thư viện, trường đại học hoặc khách sạn. Hệ thống cho phép truy cập Internet tại những khu vực có sóng của hệ thống này, hoàn toàn không cần đến cáp nối. Ngoài các điểm kết nối công cộng(hotspots), WiFi có thể được thiết lập ngay tại nhà riêng.Tên gọi 802.11 bắt nguồn từ viện IEEE (Institute of Electrical and ElectronicsEngineers). Viện này tạo ra nhiều chuẩn cho nhiều giao thức kỹ thuật khác nhau, vànó sử dụng một hệ thống số nhằm phân loại chúng; 3 chuẩn thông dụng của WiFi hiệnnay là 802.11a/b/g.



Hình :

Mô hình mạng Wifi

Hoạt động

Truyền thông qua mạng không dây là truyền thông vô tuyến hai chiều. Cụ thể:

- Thiết bị adapter không dây (hay bộ chuyển tín hiệu không dây) của máy tính chuyển đổi dữ liệu sang tín hiệu vô tuyến và phát những tín hiệu này đi bằng một ăng-ten.
- Thiết bị router không dây nhận những tín hiệu này và giải mã chúng. Nó gửi thông tin tới Internet thông qua kết nối hữu tuyến Ethernet.Quy trình này vẫn hoạt động với chiều ngược lại, router nhận thông tin từInternet, chuyển chúng thành tín hiệu vô tuyến và gởi đến adapter không dây của máytính

Các chế độ bảo mật trong WiFi

Các chế độ bảo mật của router thường có:

- Wired Equivalency Privacy (WEP) sử dụng công nghệ mã hóa 64 bit hoặc 128 bit. Mã hóa 128 bit an toàn hơn.
- WiFi Protected Access (WPA) là một bước tiến của WEP và hiện giờ là một phần của giao thức mạng bảo mật không dây 802.11i. Nó sử dụng giao thức mã hóa toàn bộ bằng một khóa tạm thời. Giống như WEP, bảo mật WPA cũng phải đăng nhập bằng một mật khẩu. Hầu hết các điểm truy cập không dây công cộng hoặc là mở hoàn toàn hoặc bảo mật bằng WPA hay WEP 128 bit.
- Media Access Control (MAC) bảo mật bằng cách lọc địa chỉ của máy tính. Nó không dùng mật khẩu đối với người sử dụng, nó căn cứ vào phần cứng vật lý của máy tính. Mỗi một máy tính đều có riêng một địa chỉ MAC độc nhất. Việc lọc địa chỉ MAC chỉ cho phép những máy đã đăng ký mới được quyền truy cập mạng. Cần đăng ký địa chỉ của máy tính khi thiết lập trong router.

Sóng Wifi

Các sóng vô tuyến sử dụng cho WiFi gần giống với các sóng vô tuyến sử dụng cho thiết bị cầm tay, điện thoại di động và các thiết bị khác. Nó có thể chuyển và nhận sóng

vô tuyến, chuyển đổi các mã nhị phân 1 và 0 sang sóng vô tuyến và ngược lại.Tuy nhiên, sóng WiFi có một số khác biệt so với các sóng vô tuyến khác ở chỗ:

- Chúng truyền và phát tín hiệu ở tần số 2.5 GHz hoặc 5GHz. Tần số này cao hơn so với các tần số sử dụng cho điện thoại di động, các thiết bị cầm tay và truyền hình. Tần số cao hơn cho phép tín hiệu mang theo nhiều dữ liệu hơn.
- Chúng dùng chuẩn 802.11.

Sơ lược về chuẩn 802.11

Chuẩn 802.11b là phiên bản đầu tiên trên thị trường. Đây là chuẩn chậm nhất và rẻ tiền nhất, và nó trở thành ít phổ biến hơn so với các chuẩn khác. 802.11b phát tín hiệu ở tần số 2.4 GHz, nó có thể xử lý đến 11megabit/giây, và nó sử dụng mã CCK (complimentary code keying).

Chuẩn 802.11g cũng phát ở tần số 2.4 GHz, nhưng nhanh hơn so với chuẩn 802.11b, tốc độ xử lý đạt 54 megabit/giây. Chuẩn 802.11g nhanh hơn vì nó sử dụng mã OFDM (orthogonal frequency-division multiplexing), Một công nghệ mã hóa hiệu quả hơn.

Chuẩn 802.11a phát ở tần số 5 GHz và có thể đạt đến 54 megabit/giây. Nó cũng sử dụng mã OFDM. Những chuẩn mới hơn sau này như 802.11n còn nhanh hơn chuẩn 802.11a, nhưng 802.11n vẫn chưa phải là chuẩn cuối cùng.

WiFi có thể hoạt động trên cả ba tần số và có thể nhảy qua lại giữa các tần số khác nhau một cách nhanh chóng. Việc nhảy qua lại giữa các tần số giúp iảm thiểu sự nhiễu sóng và cho phép nhiều thiết bị kết nối không dây cùng một lúc.

Ưu điểm và nhược điểm của mạng không dây

Ưu điểm

Sự tiện lợi: Mạng không dây cũng như hệ thống mạng thông thường. Nó cho phép người dùng truy xuất tài nguyên mạng ở bất kỳ nơi đâu trong khu vực được triển khai(nhà hay văn phòng). Với sự gia tăng số người sử dụng máy tính xách tay(laptop), đó là một điều rất thuận lợi.

Khả năng di động: Với sự phát triển của các mạng không dây công cộng, người dùng có thể truy cập Internet ở bất cứ đâu. Chẳng hạn ở các quán Cafe, người dùng có thể truy cập Internet không dây miễn phí.

Hiệu quả: Người dùng có thể duy trì kết nối mạng khi họ đi từ nơi này đến nơi khác.

Triển khai: Việc thiết lập hệ thống mạng không dây ban đầu chỉ cần ít nhất 1 access point. Với mạng dùng cáp, phải tốn thêm chi phí và có thể gặp khó khăn trong việc triển khai hệ thống cáp ở nhiều nơi trong tòa nhà.

Khả năng mở rộng: Mạng không dây có thể đáp ứng tức thì khi gia tăng số lượng người dùng. Với hệ thống mạng dùng cáp cần phải gắn thêm cáp.

Nhược điểm

Bảo mật: Môi trường kết nối không dây là không khí nên khả năng bị tấn công của người dùng là rất cao.

Phạm vi: Một mạng chuẩn 802.11g với các thiết bị chuẩn chỉ có thể hoạt động tốt trong phạm vi vài chục mét. Nó phù hợp trong 1 căn nhà, nhưngvới một tòa nhà lớn thì không đáp ứng được nhu cầu. Để đáp ứng cần phải mua thêm Repeater hay access point, dẫn đến chi phí gia tăng.

Độ tin cậy: Vì sử dụng sóng vô tuyến để truyền thông nên việc bị nhiễu, tín hiệu bị giảm do tác động của các thiết bị khác (lò vi sóng,….) là không tránh khỏi. Làm giảm đáng kể hiệu quả hoạt động của mạng.

Tốc độ: Tốc độ của mạng không dây (1- 125 Mbps) rất chậm sovới mạng sử dụng cáp (100Mbps đến hàng Gbps)

CHƯƠNG 3: LẬP TRÌNH J2ME TRÊN ĐIỆN THOẠI DI ĐỘNG

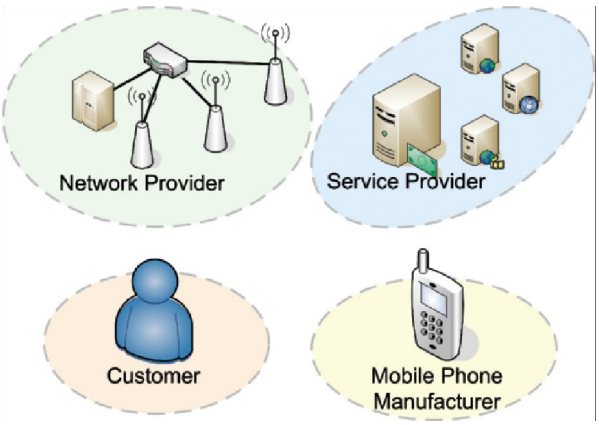
3.1 Tổng quan về điện thoại di động và J2ME

3.1.1 Giới thiệu về điện thoại di động

a. Mobile Phone Family

Điện thoại di động được gọi với tên gọi như vậy vì ứng dụng đầu tiên và quan trọng nhất của nó là cho phép người dùng có thể di chuyển trong khi đang thực hiện cuộc gọi.

Trong kiến trúc của mạng di động, các trạm cơ sở (base station) chính là những nhân tố quan trọng, các trạm này làm nhiệm vụ kết nối với điện thoại di động, đảm bảo sự tồn tại cho mạng di động, và cho phép mạng di động kết nối với những thành phần truyền thông hữu tuyến khác. Bên cạnh đó, các trạm cơ sở và mạng xương sống còn cho phép người dùng ở các vùng cell khác nhau có thể truyền thông được với nhau. Mạng di động đã trải qua nhiều thế hệ, đầu tiên là thế hệ 1G, đây là hệ thống truyền tín hiệu tương tự (analog), thế hệ này không cung cấp nhiều dịch vụ, chủ yếu là các dịch vụ về âm thanh. Thế hệ thứ hai 2G, chuyển đổi từ việc sử dụng tín hiệu analog sang tín hiệu số, từ đó thêm vào rất nhiều dịch vụ mới như SMS và trao đổi dữ liệu với mạng Internet, nhưng sự thay đổi quan trọng nhất của 2G so với 1G là sự phân chia giữa nhà mạng và nhà cung cấp dịch vụ. Trước đó nhà cung cấp mạng là những người độc quyền, họ là người cung cấp mạng di động cũng là người quyết định những dịch vụ nào sẽ được cung cấp ở máy di động của người dùng, sau khi mạng 2G ra đời và có sự phân chia giữa nhà cung cấp mạng và nhà cung cấp dịch vụ, thị trường di động đã được chia ra làm bốn thành phần chính: Nhà cung cấp mạng, nhà cung cấp dịch vụ, người sử dụng và nhà sản xuất điện thoại di động.



Hình 3.1:

Nhà cung cấp dịch vụ, người sử dụng và nhà sản xuất điện thoại di động trong thị trường di động.

Giữa các thành viên này trong mạng di động có nhiều rằng buộc lẫn nhau, quan trọng nhất là mối quan hệ giữa nhà cung cấp mạng và nhà sản xuất điện thoại di động. Để thu hút khách hàng, nhà mạng đã thỏa thuận với nhà sản xuất điện thoại di động, để điện thoại di động được bán với số lượng lớn hơn và cũng có nhiều người sử dụng mạng điện thoại di động hơn, mặt khác nhà mạng cũng đưa ra được một loạt các dịch vụ sẵn có cho điện thoại di động. Xa hơn nữa, nhà cung cấp mạng còn điều khiển việc dịch vụ nào được đi qua mạng của họ và làm cho những nhà cung cấp dịch vụ phải phụ thuộc vào quyết định của họ.

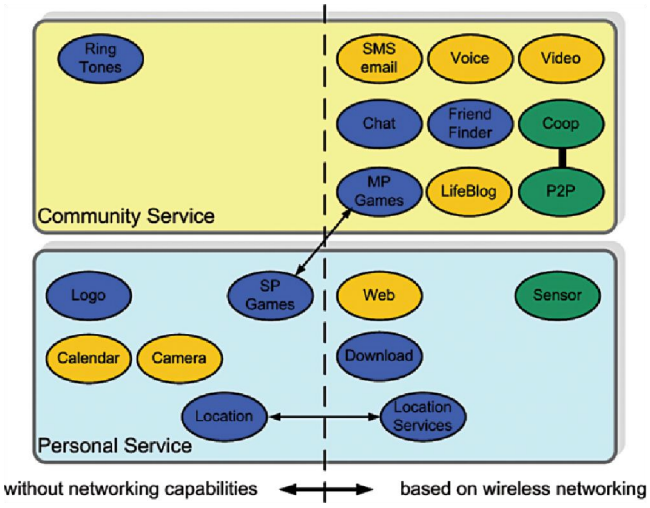
Sự độc quyền của nhà cung cấp mạng sẽ bị suy giảm đi cùng với công nghệ không dây như WLAN và Bluetooth. Các dạng truyền thông này cho phép tạo ra những kiểu dịch vụ mới mà không cần phải phụ thuộc nhiều vào nhà cung cấp mạng. Tuy nhiên, do vấn đề lợi nhuận, sự phát triển tự do của các dạng truyền thông trên cũng bị hạn chế bởi

nhà sản xuất điện thoại di động. Nhà cung cấp dịch vụ dường như đã tự do hơn trong lập trình để tạo ra các sản phẩm dịch vụ mới, tuy nhiên, nhiều khi việc tạo ra ứng dụng trên điện thoại di động lại động chạm tới lợi ích của nhà cung cấp mạng, do đó sự linh hoạt của chiếc điện thoại di động lại bị kìm chế, ví dụ như hệ thống VoIP chỉ có thể truyền theo kiểu bán song công, hay khi kết hợp giữa giao thức IP với module GPRS, IP thông qua Bluetooth hay WLAN cũng khó được hỗ trợ hay khó sử dụng ở một vài dòng điện thoại, nhưng sự hạn chế này cũng không ngăn được việc tạo ra các dịch vụ thú vị, đó chỉ là vấn đề thời gian.

Thế hệ mạng di động 3G. 3G là công nghệ truyền thông thế hệ thứ ba, cho phép truyền cả dữ liệu thoại và dữ liệu ngoài thoại (tải dữ liệu, gửi email, tin nhắn nhanh SMS, hình ảnh,…). Hệ thống 3G yêu cầu một mạng truy cập radio hoàn toàn khác so với hệ thống 2G hiện nay. Trong các dịch vụ của 3G, cuộc gọi video thường được mô tả như một dịch vụ trọng tâm của sự phát triển.

Do chi phí cho bản quyền các tần số phải trang trải trong nhiều năm trước khi đạt tới các thu nhập do 3G đem lại, nên việc xây dựng mạng 3G đòi hỏi một khối lượng đầu tư khổng lồ. Cũng vì vậy nhiều nhà cung cấp dịch vụ viễn thông đã rơi vào khó khăn về tài chính, càng khiến cho việc triển khai 3G tại nhiều nước bị chậm trễ, ngoại trừ ở Nhật Bản và Hàn Quốc – những nước tạm bỏ qua các yêu cầu về bản quyền tần số, mà đặt ưu tiên cao việc phát triển hạ tầng công nghệ thông tin – viễn thông quốc gia. Nhật Bản là nước đầu tiên đưa 3G vào khai thác thương mại một cách rộng rãi. Năm 2005, khoảng 40% các thuê bao tại Nhật Bản là thuê bao 3G, khiến cho mạng 2G dần biến mất tại nước này. Sự thành công của 3G tại Nhật Bản chỉ ra rằng điện thoại video không phải là ứng dụng hủy diệt (killer application). trong thực tế, việc sử dụng điện thoại video thời gian thực chỉ chiếm một phần nhỏ trong các dịch vụ của 3G. Mặt khác, việc tải về tệp âm nhạc lại được sử dụng nhiều nhất, nhất là giới trẻ. Sự phát triển của 3G đã mở ra rất nhiều triển

vọng về việc xây dựng các dịch vụ mới cho các nhà phát triển phần mềm, các nhà cung cấp dịch vụ, tuy nhiên rất nhiều dự án bị đổ vỡ bởi chính người sử dụng cũng không biết họ mong muốn những gì, vì vậy chúng ta cần phải phân tích cẩn thận trước khi tiến hành một dự án phần mềm cho điện thoại di động. Ứng dụng trên điện thoại di động chia làm hai loại: Dịch vụ cá nhân và dịch vụ truyền thông.



Hình 3.2:

Các loại hình dịch vụ dành cho điện thoại di động

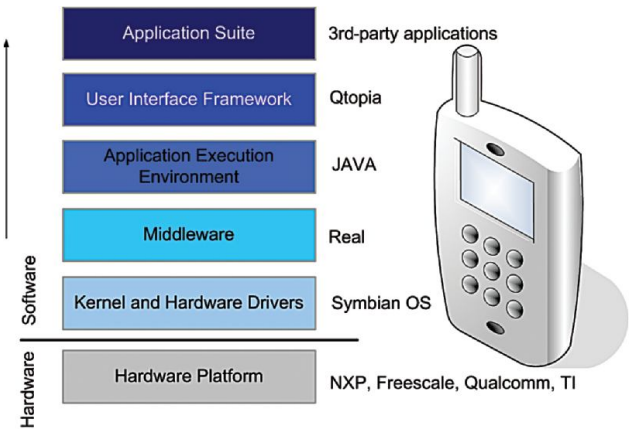
Dịch vụ cá nhân là tất cả các dịch vụ được sử dụng bởi khách hàng trên điện thoại di động với sự giới hạn hay không tương tác với những khách hàng khác. Dịch vụ truyền thông thì ngược lại, đó là các dịch vụ tạo ra sự tương tác giữa những người sử dụng. Dịch vụ cá nhân bao gồm các dịch vụ như là: lịch, máy ảnh, tải về logo, nhạc trên điện thoại di động hay chơi trò chơi như chơi rắn, bóng bàn... Cả dịch vụ cá nhân lẫn dịch vụ truyền thông đều được chia ra làm hai loại dịch vụ có và không có sự hỗ trợ của mạng không

dây. Mạng không dây được hỗ trợ bởi chuẩn kết nối GSM, GPRS, EDGE, 3G, bluetooth, hay WLAN. Như hình trên, các dịch vụ có màu cam là các dịch vụ đã có sẵn trên hầu hết các điện thoại di động, các dịch vụ màu xanh da trời là các dịch vụ có thể được cài đặt thêm trên điện thoại di động, và cuối cùng là các dịch vụ màu xanh lá cây, là những dịch vụ của tương lai. Dịch vụ truyền thông với sự hỗ trợ của mạng không dây được hứa hẹn sẽ tạo ra nhiều điều thú vị, vì con người chúng ta luôn sống trong sự tương tác với những người khác, các dịch vụ đang phát triển như SMS, dịch vụ thoại video, tải WEB, xem phim trực tuyến... và các trò chơi tương tác đang là một hướng phát triển đầy triển vọng cho các nhà cung cấp dịch vụ.

Ngoài ra trong tương lai, khi mạng 4G và 5G phát triển, chúng ta lại có thêm nhiều loại dịch vụ mới như dịch vụ peer to peer, dịch vụ truyền thông ngang hàng, dịch vụ sensor... Điều đó cho thấy rằng điện thoại di động và mạng di động là mảnh đất màu mỡ cho các lập trình viên.

b. The flexible Mobile Phone

Điện thoại di động ngày càng trở nên linh hoạt trong con mắt của các nhà phát triển ứng dụng, thông qua các ngôn ngữ lập trình, với ngôn ngữ tiên phong là Java với phiên bản nhỏ gọn J2me, điện thoại di động đã trở thành một môi trường lập trình hấp dẫn đối với các lập trình viên.



Hình 3.3:

Kiến trúc của chiếc điện thoại có thể lập trình

Nhìn vào hình trên ta thấy rằng các ứng dụng di động (như các chương trình Java) chạy trên tầng ứng dụng *application suite*, khung giao diện người dùng *user interface framework* cung cấp tất cả các chức năng cho phép ta điều khiển điện thoại di động. Các nhà sản xuất điện thoại di động thường truy nhập vào nền tảng của họ (với một vài hạn chế) bằng cách sử dụng lõi và trình điều khiển phần cứng *kernel and hardware drivers* như là hệ điều hành Symbian để truy nhập vào nền tảng phần cứng. Gần như mọi thứ thuộc về phần mềm đều có thể truy cập được, chỉ có một phần tĩnh không thể thay đổi được của chiếc điện thoại là nền tảng phần cứng.

Hình 3.4:

Giao diện người dùng, giao diện kết nối và tài nguyên có sẵn

của điện thoại di động

Ngoài khả năng lập trình được, điện thoại di động còn có rất nhiều các khả năng và chức năng khác, chúng ta sẽ nhóm các đặc tính này của điện thoại di động ra thành ba nhóm là giao diện người dùng *user interface*, giao diện kết nối *communication interface,* và tài nguyên sẵn có *buit-in resources*. Giao diện người dùng bao gồm loa, microphone, camera, màn hình, các bộ cảm ứng, và bàn phím. Tài nguyên có sẵn bao gồm pin, bộ xử lý trung tâm và bộ nhớ. Chúng ta đặc biệt quan tâm tới nhóm thứ ba là nhóm giao diện kết nối, giao diện kết nối thường bao gồm khả năng kết nối celular và short-range.

Thay vì việc sở hữu một chiếc điện thoại thông minh với đầy đủ các chức năng và chất lượng tốt, chúng ta nên tìm hiểu cách sử dụng và chia sẻ một cách thông minh những khả năng của điện thoại di động. Thực vậy, chính sự khác biệt trong thiết kế của các chủng loại điện thoại di động đã tạo ra nhiều tính năng đặc biệt, từ một chiếc điện

thoại đơn giản với khả năng thoại có thể trở thành một thiết bị đầu cuối với âm nhạc và hình ảnh sống động. Công nghệ mạng không dây cho phép chúng ta làm điều đó.

Những nhà phát triển ứng dụng có thể bắt đầu nhìn nhận chiếc điện thoại như là một tập hợp các khả năng có thể được gộp lại vì những mục đích nào đó. Tuy nhiên điện thoại di động không chỉ bị giới hạn bởi sự kết hợp của các thành phần phần cứng. Thậm chí phần mềm cũng được sử dụng một cách hết sức linh hoạt. Một trong những ý tưởng đầu tiên là thiết kế lớp trung gian (cross-layer). Giao thức cross-layer là ý tưởng của việc thay đổi các lớp theo nhu cầu. Một số lớp giao thức có thể thậm chí được bỏ qua hay được nhóm lại theo một cách mới. Bước cuối cùng mang đến tính linh hoạt cho những mức thấp hơn của chồng nghi thức (protocol stack). Trong khi tính linh hoạt trên các thiết bị di động thương mại bị hạn chế đối với những phương pháp đa mô hình (multi-modality), tương lai sẽ được thống trị bởi những cách tiếp cận software-defined radio. Trong khi multi-modality chỉ đang chọn trong số những công nghệ không dây sẵn có khác nhau như Bluetooth hay WLAN, software-defined radio có thể tự động thay đổi các phần radio theo các chức năng cần thiết. Ở mức cao nhất, nó cũng có thể thay đổi từ truyền thông short-range sang truyền thông cellular, trao đổi chồng giao thức.

3.1.2 Kiến trúc điện thoại di động hỗ trợ J2ME

Một điện thoại di động muốn có khả năng chạy được các ứng dụng viết bằng J2ME cần phải có cấu hình tối thiểu như sau :

- Tối thiểu 512KB bộ nhớ để chạy Java
- Tối thiểu 256KB bộ nhớ dành cho phân bổ bộ nhớ thực thi chương trình
- Kết nối mạng thường trực

Chủ yếu các thiết bị điện thoại di động ngày nay là các thiết bị kết nối có giới hạn(CLDC). CLDC là cấu hình dành cho các thiết bị với khả năng xử lý,dung lượng bộ nhớ, khả năng hiển thị và lưu trữ tài nguyên hạn chế. Vì vậy ,CLDC cũng có những hạn chế khác biệt so với định nghĩa trong java:

- Không hỗ trợ dấu chấm động toán học,phép tính này đòi hỏi nhiều tài nguyên, CPU và phần lớn các CPU cho các thiết bị di động không hỗ trợ phép tính này. CLDC không hỗ trợ biến, toán tử, hằng, hàm…. Liên quan đến dấu chấm động.
- Không hỗ trợ phương thức hủy finalize(), việc "dọn dẹp" tài nguyên trước khi đối tượng bị xóa được đẩy về phía các lập trình viên.
- Hỗ trợ hạn chế các xử lý ngoại lệ lỗi, do để thực hiện quản lý ngoại lệ hệ thống cần phải có một cấu hình mạnh mà các thiết bị CLDC lại không đáp ứng được. Hơn nữa, các hệ thống nhúng luôn có cơ chế xử lý lỗi của riêng nó, thông thường là khởi động lại phần cứng.
- Quá trình xác minh lớp: khác biệt của quá trình xác minh lớp của CLDC là nó thực hiện xác minh lớp qua hai bước, để giảm bớt những yêu cầu tài nguyên trong quá trình xác minh.

Ngoài những hạn chế trên, CLDC cũng có những lớp riêng, được thiết kế cho phù hợp với đặc thù của môi trường các thiết bị phần cứng mà nó quan tâm, trong đó, đáng chứ ý nhất là bộ khung kết nối – GCF (Generic Connection Framework):

Kết nối mạng với J2me khá dễ dàng với gói java.net, gói này có kích thước vào khoảng 200 kilobytes, bao gồm khoảng 20 lớp khác nhau. Kích cỡ đó quả là "quá khổ" đối với các thiết bị di động, hơn nữa J2me còn cần phải hỗ trợ một tập hợp lớn các thiết bị có nhiều sự khác nhau về kích cỡ, hình dáng, khả năng mạng và yêu cầu nhập xuất file. Để phù hợp với các thiết bị, J2me đưa ra mô hình khung kết nối chung trong cấu hình CLDC.

Ý tưởng của nó là: định nghĩa một cách trừu tượng các hoạt động mạng và nhập xuất file để dùng chung cho một số lượng lớn các thiết bị, giống như một bộ khung nền vậy. Ở mức độ CLDC, J2me chỉ định nghĩa và tạo kết nối, giao thức thực sự và các phương thức trao đổi dữ liệu của giao thức đó được trao cho tầng Profile.

3.1.3 Giới thiệu về J2ME

J2me là một thành viên bé nhỏ trong gia đình họ Java, hướng đến lập trình trên những thiết bị thông tin gia dụng từ Internet-cho đến các máy TV, máy chụp ảnh, điện thoại di

động, Pocket PC… J2me được phát triển từ kiến trúc Java Card, Embeded Java và Personal Java của phiên bản Java 1.1. Đến sự ra đời của Java 2 thì Sun quyết định thay thế Personal Java và được gọi với tên mới là Java 2 Micro Edition, hay viết tắt là J2me. Đúng với tên gọi, J2me là nền tảng cho các thiết bị khách hàng có tính chất nhỏ, gọn với sức mạnh xử lí và tài nguyên có hạn. Có rất nhiều thiết bị dạng này như điện thoại di động, máy quay phim, chụp hình. Những thiết bị này không có tùy chọn để tải xuống và cài đặt phần mềm từ xa như PC. Phần mềm điều khiển thiết bị được cài sẵn trong quá trình sản xuất ra thiết bị áp đặt bởi nhà sản xuất. Với sự giới thiệu của J2me những thiết bị này không còn là "tĩnh" nữa, chúng hoàn toàn có thể được người dùng tự lập trình để thêm vào những tính năng mới. J2me cài trên thiết bị đã sẵn có tùy chọn để duyệt, tải xuống và cài đặt cũng như thực thi những ứng dụng Java.

J2me không phải là một ngôn ngữ mới, nhưng sự xuất hiện của nó đã cho phép công nghệ Java có thể chạy trên các thiết bị nhúng, bằng cách lược bỏ đi các chức năng mà các thiết bị này không thể hỗ trợ và thêm vào các chức năng mà các thiết bị này cần. Với sự ra đời và giới thiệu của Java dùng cho môi trường những thiết bị di động nhỏ gọn như vậy, chúng ta giờ đây đã có khả năng tận dụng hàm thư viện và sức mạnh lập trình của Java vào cuộc sống đời thường.

3.1.4 Kiến trúc J2ME

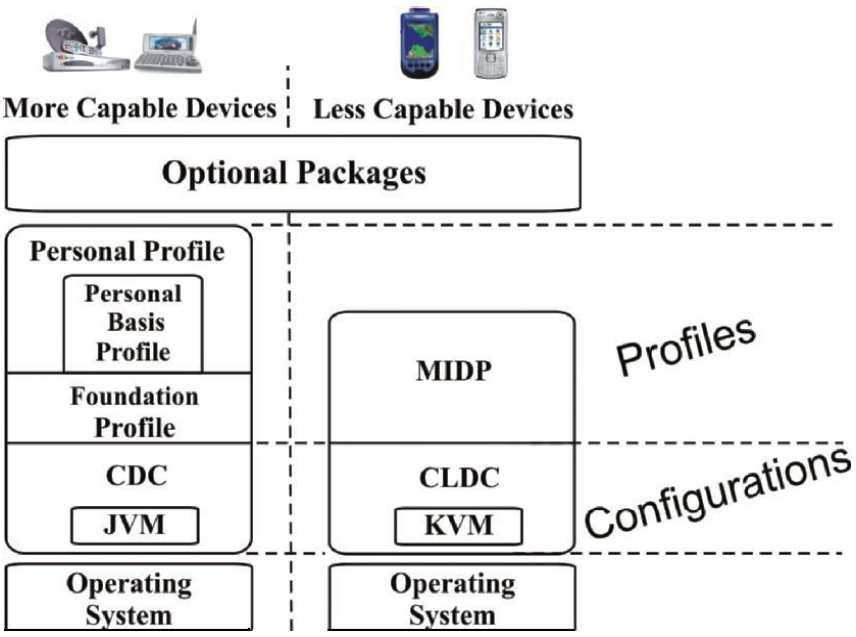
J2me được xây dựng để phát triển ứng dụng chạy trên nền các thiết bị nhúng, nhóm các thiết bị này rất phong phú và giữa chúng thường có nhiều khác biệt về đặc tính (bộ nhớ, tốc độ xử lý, khả năng kết nối mạng…), ví dụ như các thiết bị điện thoại, Screenphone, PDA, máy chụp ảnh… có thể có sự khác nhau về kích cỡ màn hình, hay giữa những chiếc điện thoại với màn hình có độ lớn như nhau nhưng lại có sự khác nhau về độ phân giải… Sự phong phú và phức tạp này khiến cho việc xây dựng một nền tảng J2me chung cho tất cả các thiết bị là không thể, thay vào đó J2me được xây dựng với một kiến trúc khác, đó là kiến trúc phân tầng, trong đó các thiết bị sẽ được phân nhóm theo đặc tính (bộ nhớ, tốc độ xử lý, khả năng kết nối mạng…) mỗi nhóm thiết bị đó tương đương với một nhóm configuration-cấu hình J2me, cấu hình sẽ đặc tả nền tảng Java trên

nhóm các thiết bị này, hay nói cách khác, cấu hình sẽ định nghĩa các chức năng chung cơ bản nhất cho các thiết bị cùng nhóm.

Tuy nhiên, với những giới hạn về phần cứng như vậy, khả năng phát triển ứng dụng dựa trên configuration là không lớn, hơn nữa, giữa các thiết bị có cùng cấu hình cũng có nhiều đặc điểm, những khả năng nổi trội khác nhau, để cung cấp môi trường lập trình cho các thiết bị chuyên biệt trong cùng một nhóm cấu hình và để linh hoạt hơn khi công nghệ thay đổi, các thiết bị lại một lần nữa được phân nhóm, nhóm cấu hình sẽ được phân nhỏ hơn và được gọi là các profile.

Profile là định nghĩa mở rộng thêm cho một phân loại cấu hình, nó cung cấp một tập các thư viện lập trình cho phép tạo ra các ứng dụng chạy trên một kiểu thiết bị đặc biệt.

Hình dưới mô tả kiến trúc phân tầng của J2ME:



3.1.5 Môi trường phát triển , công cụ lập trình ứng dụng trên điện thoại di động với J2ME.

a.Java Development Kit

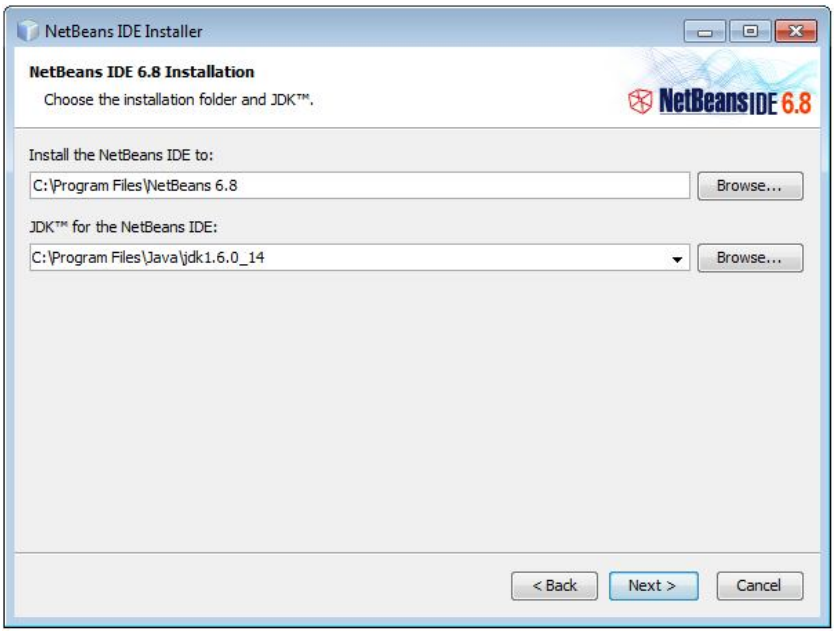
Chúng ta cần JDK để sử dụng trình biên dịch và thông dịch của nó cho các ứng dụng mà chúng ta tạo ra. JDK có thể dùng cho cả CDC và CLDC, tuy nhiên phải có phiên bản từ sdk 1.6 trở lên. Bạn download JDK từ trang Web của sun:

http://java.sun.com

Sau đó tiến hành cài đặt như bất kỳ phần mềm nào khác, trong quá trình đó chương trình sẽ hỏi bạn nơi đặt thư mục, nếu bạn không chọn (browse) đường dẫn mới, chương trình sẽ sử dụng đường dẫn mặc định.

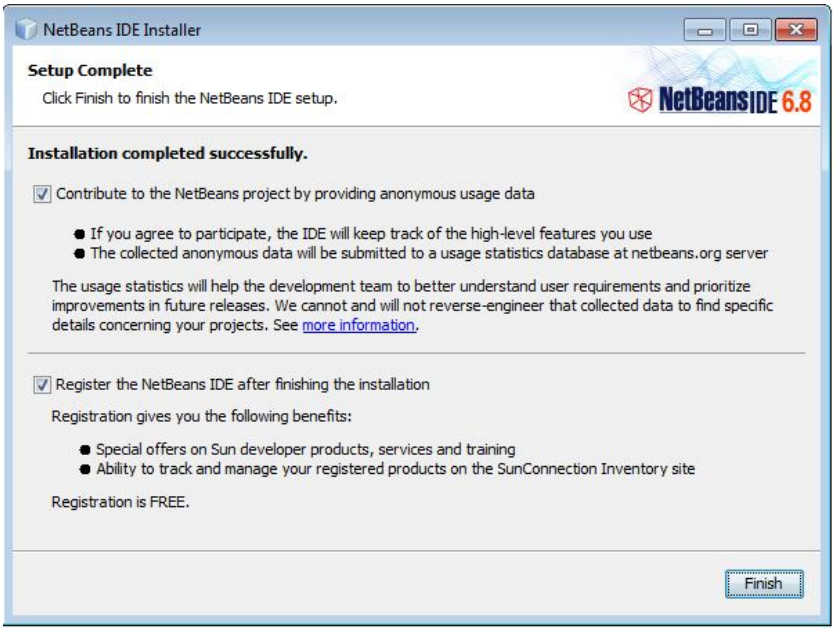
b. NetBeans

Việc cài đặt NetBeans cũng được tiến hành như các phần mềm khác, tuy nhiên cần phải chú ý đến việc NetBeans tìm folder cài đặt Java, nếu như NetBeans không tự tìm được người dùng cần cung cấp cho NetBeans đường dẫn đến thư mục đó.



Hình 3.5:Tìm đường dẫn đến thư mục đã cài đặt Java

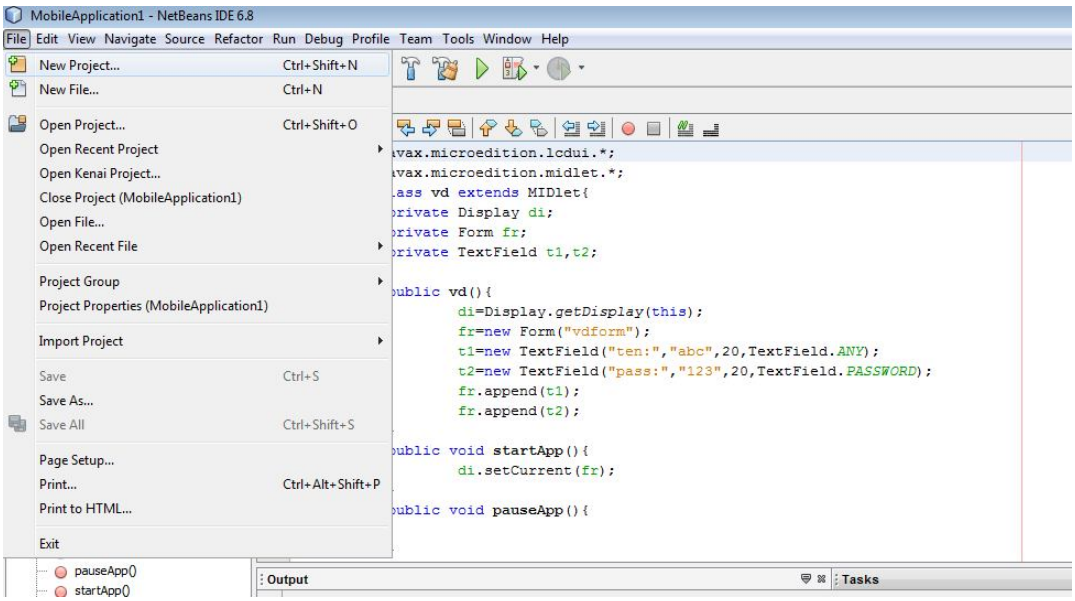
Khi việc cài đặt hoàn tất, cửa sổ sau sẽ được hiện lên, nhấn nút Finish để kết thúc. Trong cửa sổ này có tùy chọn đăng ký NetBeans, vì là một sản phẩm mã nguồn mở, bạn có thể bỏ qua việc đăng ký này mà vẫn được sử dụng phần mềm một cách bình thường :



Hình 3.6:Kết thúc việc cài đặt NetBeans

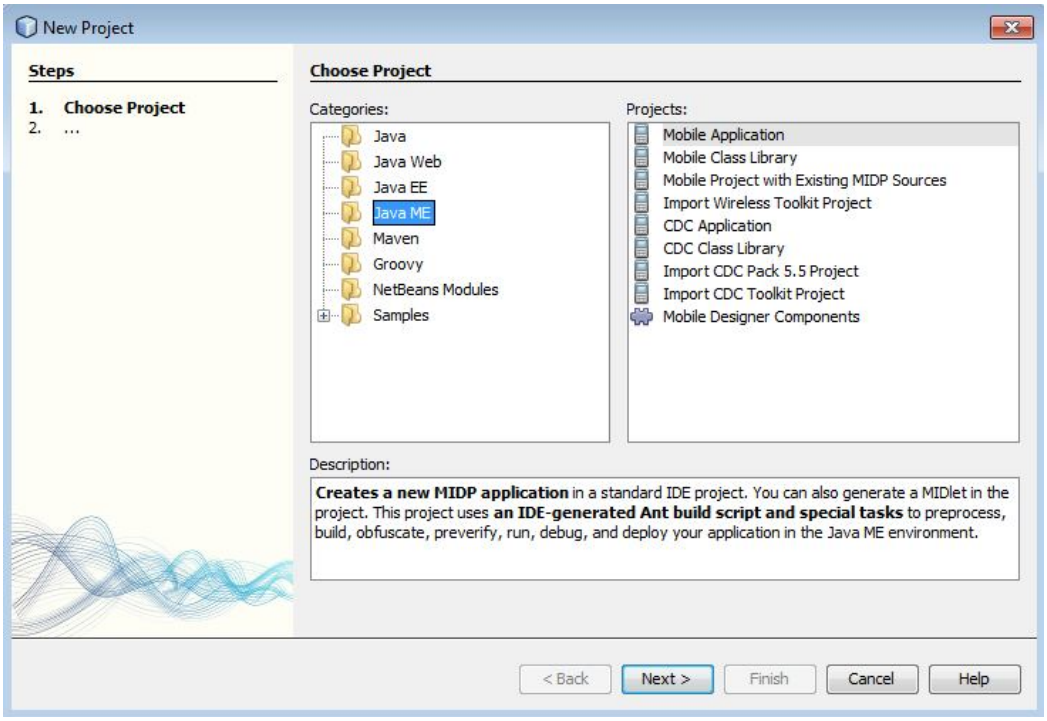
c. Tạo một ứng dụng MIDP với Netbean

Để tạo một dự án mới, hãy nhấn vào tùy chọn File và chọn New Project :



Hình 3.7:Tạo một dự án mới

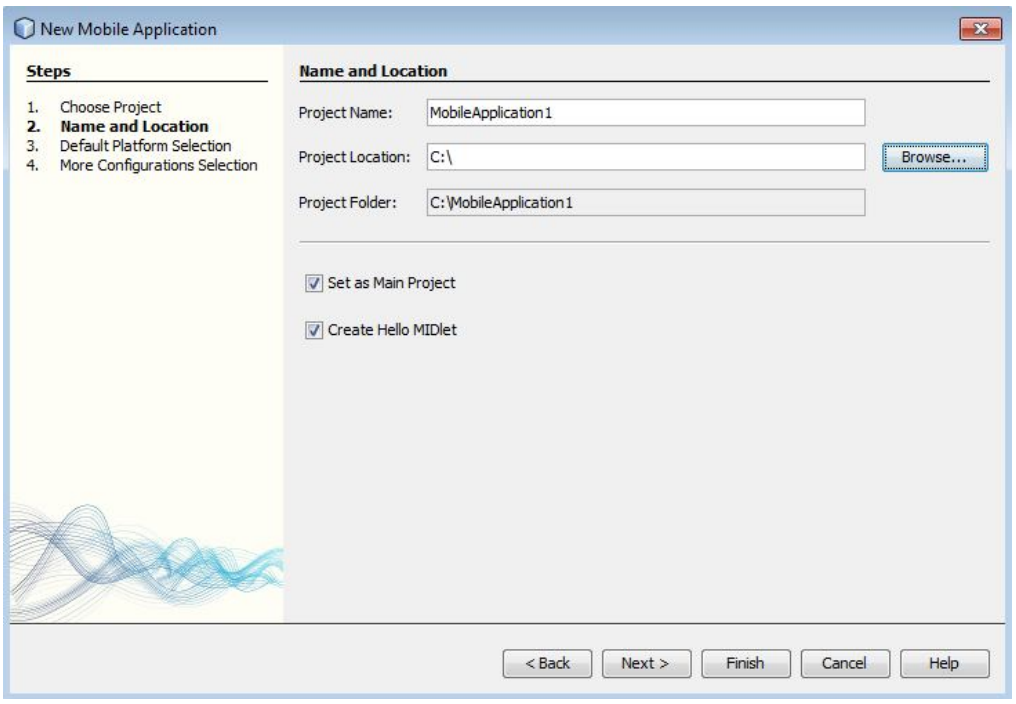
Sau đó, hãy chọn kiểu của dự án : dự án Java, Java Web hay JavaME... để tạo một dự án J2me bạn hãy chọn JavaME :



Hình 3.8:Tạo dự án J2me

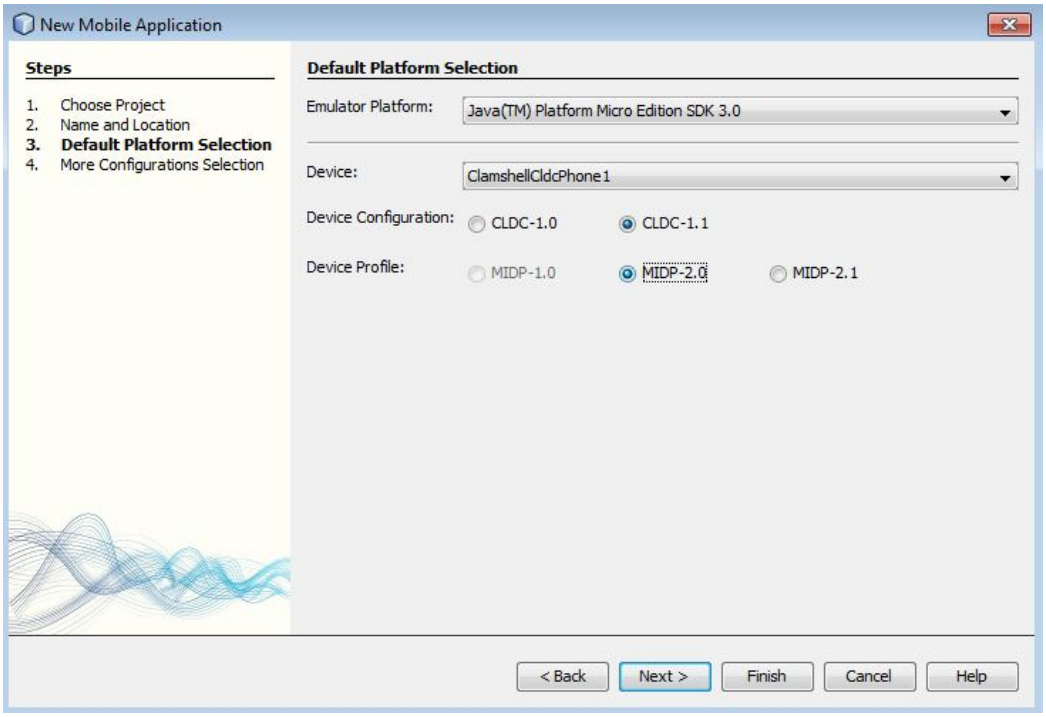
Bước tiếp theo NetBeans sẽ yêu cầu bạn đặt tên cho dự án, trong cửa sổ này có tùy chọn Set as Main Project và Create Hello MIDlet, nếu bạn muốn dự án mới này là dự án chính hãy chọn tùy chọn Set as Main Project, nếu cần tạo một lớp hello world hãy chọn tùy chọn còn lại.

Bên cạnh đó, bạn cần phải chọn nơi lưu giữ Project của mình trong phần Project Location :



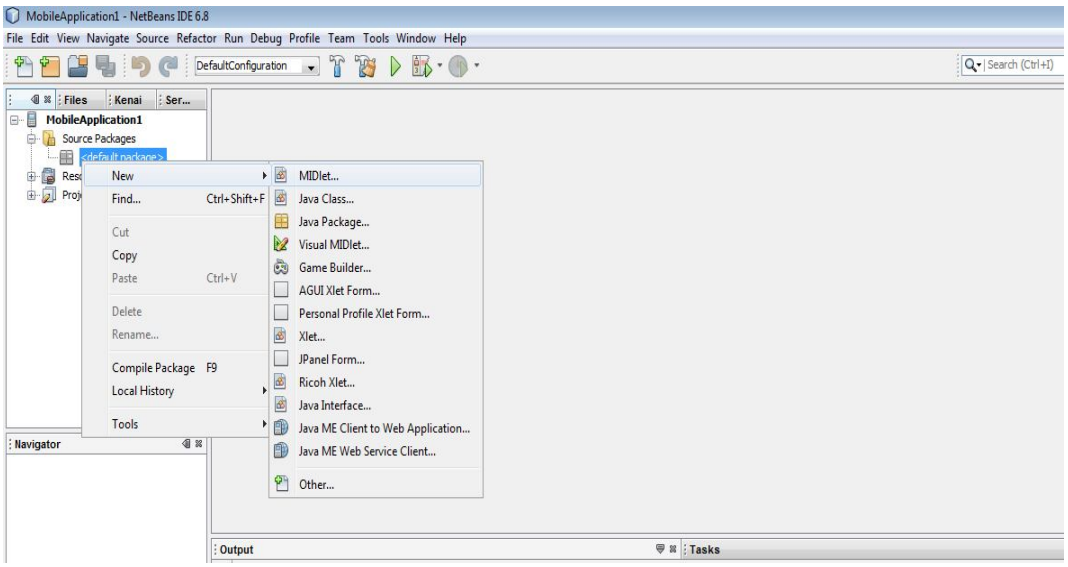
Hình 3.9:Đặt tên cho dự án

Cửa sổ tiếp theo sẽ yêu cầu bạn chọn phiên bản CLDC và MIDP phù hợp, việc này còn tùy thuộc vào việc bạn lựa chọn phát triển dự án của bạn trên những dòng sản phẩm nào ?:



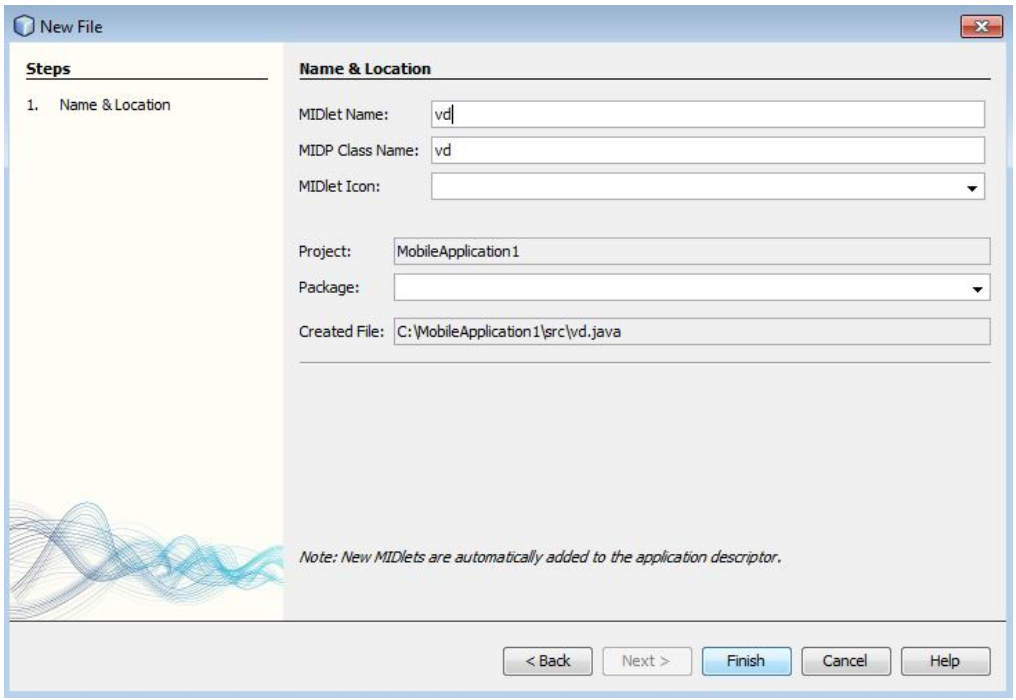
Hình 3.10:Lựa chọn phiên bản CLDC và MIDP phù hợp

Sau khi có dự án mới, việc tiếp theo cần phải làm là viết code cho dự án, trước tiên hãy tạo một file mới, nếu là file thực thi chính, bạn hãy chọn MIDlet :



Hình 3.11:Tạo file mới cho dự án

Đặt tên cho file mới đó (lưu ý rằng tên file này phải trùng với tên lớp được tạo ra trong file) :

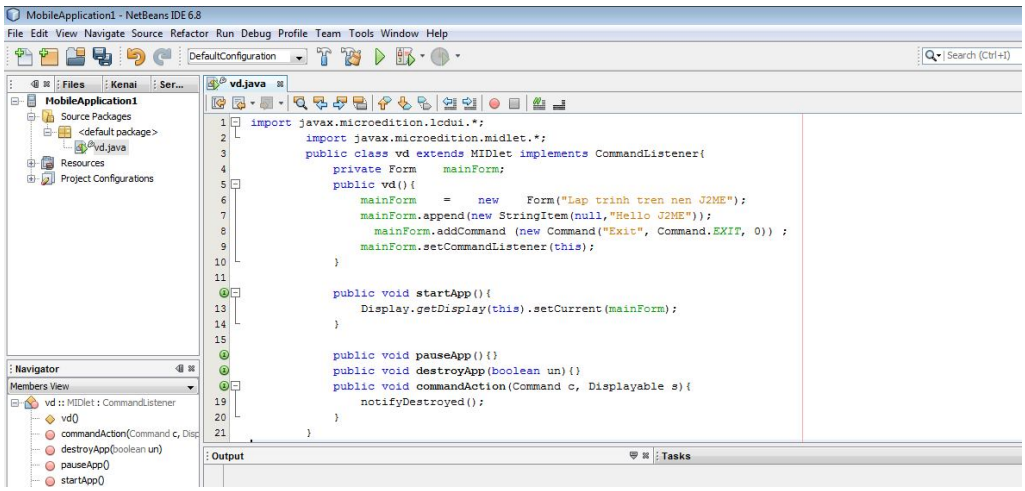


Hình 3.12 :Đặt tên cho file

Dưới đây là một đoạn code:

```
import javax.microedition.lcdui.*;
    import javax.microedition.midlet.*;
    public class vd extends MIDlet implements CommandListener{
      private Form mainForm;
      public vd(){
        mainForm = new Form("Lap trinh tren nen J2ME");
        mainForm.append(new StringItem(null,"Hello J2ME"));
        mainForm.addCommand (new Command("Exit",    Command.EXIT, 0));
        mainForm.setCommandListener(this);
      }
      public void startApp(){
        Display.getDisplay(this).setCurrent(mainForm);
      }
      public void pauseApp(){}
      public void destroyApp(boolean un){}
      public void commandAction(Command c, Displayable s){
        notifyDestroyed();
      }
    }
```

Bạn hãy copy đoạn code này, vào file vd vừa được tạo ra:



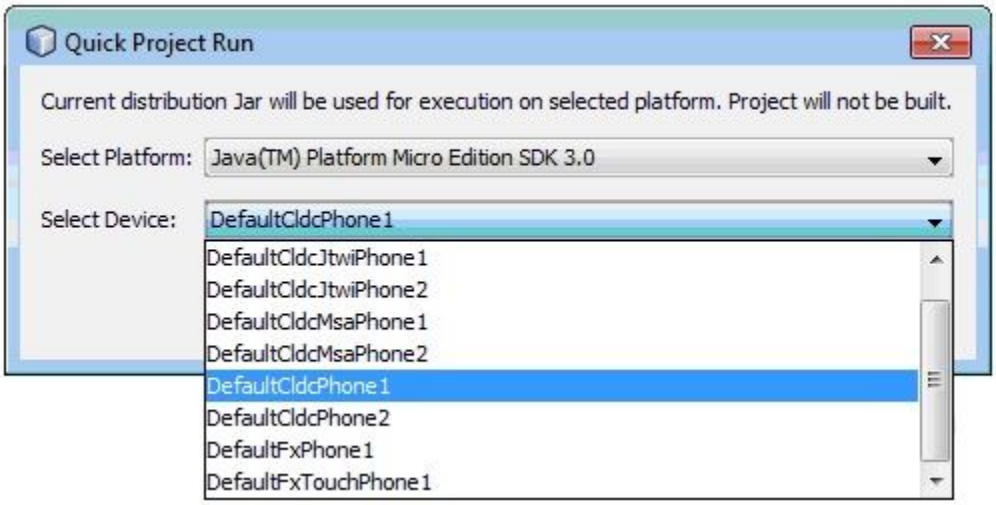
Hình 3.13 :Soạn code cho dự án

Sau khi có code rồi bạn hãy nhấn ctrl+s để lưu và cũng là dịch chương trình. Bây giờ bạn chú ý nút mà xanh có hình mũi tên nằm trên thanh công cụ, hãy nhấn nó để chạy ứng dụng trên bộ mô phỏng (nếu dự án của bạn là dự án chính – Main Project), nếu không bạn hãy kích chuột phải vào tên dự án và chọn run.

Sau đây là kết quả thực thi của chương trình trên bộ mô phỏng:

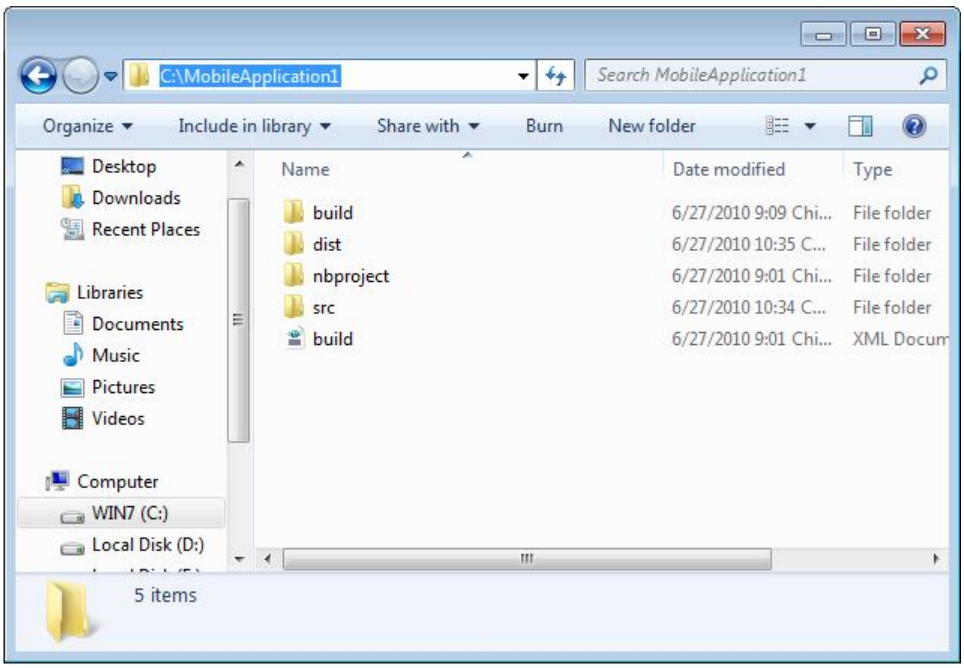
Hình 3.14:Kết quả thực thi của chương trình vd

Một chương trình có thể chạy trên nhiều bộ mô phỏng khác nhau, sau khi chương trình đã được build, bạn hãy nhấn chuột phải vào tên của dự án và chọn run with, một cửa sổ sẽ hiện ra cho phép bạn chọn các mẫu mô phỏng điện thoại khác nhau (điều này rất có ích trong việc kiểm tra phạm vi thực thi của chương trình trên các dòng điện thoại):



Hình 3.15:Lựa chọn bộ mô phỏng để thực thi chương trình

Đối với mỗi dự án J2me được tạo ra, trong folder mà bạn đã chọn để lưu dự án sẽ có cấu trúc cây thư mục như sau:



Hình 3.15: Cấu trúc thư mục của một Project

Trong đó:

- Folder src sẽ chứa mã nguồn của dự án (cùng với các tài nguyên hình ảnh, âm thanh… nếu có).
- Folder dist sẽ chứa các sản phẩm cuối cùng của dự án. Bạn chỉ cần copy 2 file .JAR và .JAD trong thư mục này vào điện thoại di động, cài đặt từ file .JAD là chương trình của bạn có thể sử dụng được.

3.2 Phát triển ứng dụng trên điện thoại di động với J2ME

3.2.1 Hiện trạng thiết bị di động và MIDlet

Hiện trạng thiết bị di động (MIDP) là một Đây là Profile được định nghĩa dành riêng cho các thiết bị di động và là thành phần chính trong J2me. MIDP cung cấp các chức năng cơ bản cho hầu hết các dòng thiết bị di động, phổ biến nhất như các máy điện thoại di động và các máy PDA. Tuy nhiên MIDP được thiết kế cho các máy di động có cấu hình rất thấp, cho nên nó không phải không có những hạn chế. Trước khi nói về những hạn chế và những khác biệt của MIDP so với J2se, chúng ta sẽ cùng điểm qua những yêu cầu phần cứng, phần mềm của MIDP:

Các yêu cầu phần cứng:

- Màn hình phải hỗ trợ ít nhất 96×54 pixel
- Phải có ít nhất một kiểu nhập liệu cho người dùng
- Tối thiểu 128 kilobytes ký ức dùng để chạy các thành phần của thiết bị di động
- Tối thiểu 8 kilobytes ký ức trống dùng cho ứng dụng lưu trữ dữ liệu cấu hình riêng cho ứng dụng
- Tối thiểu 32 kilobytes ký ức để chạy Java
- Kết nối mạng không dây

Về yêu cầu phần mềm, hệ điều hành trên thiết bị phải cung cấp tối thiểu cơ chế định thời, xử lý ngoại lệ và xử lý ngắt, có khả năng chạy KVM và:

- Phần mềm phải hỗ trợ cách ghi ảnh bitmap ra màn hình
- Phần mềm phải chấp nhận ký tự nhập vào và chuyển thông tin cho máy ảo Java
- Hệ thống cũng phải có khả năng đọc, ghi vào vùng cứng của thiết bị.
- Phải có khả năng truy nhập đến chức năng nối mạng trên thiết bị.

Hạn chế và khác biệt của MIDP:

- Phần lớn các thư viện đồ họa trong J2se không thể dùng trong MIDP, do khả năng hiển thị và xử lý hạn chế của các thiết bị MIDP.
- Không hỗ trợ quản lý file và thư mục, thay vào đó để lưu trữ dài hạn, J2me cung cấp cho bạn một chức năng khác tương đương là Record Manager System – RMS, hệ thống quản lý bản ghi.

Những chức năng đặc biệt mà MIDP cung cấp:

- Download qua mạng an toàn qua việc hỗ trợ giao thức HTTPS
- Kiểm soát việc kết nối giữa máy di động và server: ví dụ, chương trình không thể kết nối tới server nếu thiếu sự chấp thuận của người sử dụng
- Thêm các API hỗ trợ Multimedia
- Mở rộng các tính năng của Form
- Hỗ trợ kiểu ảnh RBG

3.2.2 Thành phần cơ bản của một ứng dụng MIDlet.

MIDlet là một ứng dụng được thiết kế cho MIDP profile, vì lý do đó, một MIDlet không chỉ tham chiếu được đến các lớp của MIDP, mà còn có thể tham chiếu được đến các lớp của CLDC.

giống các ứng dụng Java Destop truyền thống, ứng dụng MIDlet không dùng một hàm main để làm điểm truy nhập, mà dùng một lớp con của lớp *javax.microedition.midlet.MIDlet* làm điểm truy nhập. Lớp này định nghĩa một số hàm trừu tượng (abtract), các hàm này sẽ được gọi đến khi trạng thái của MIDlet thay đổi.

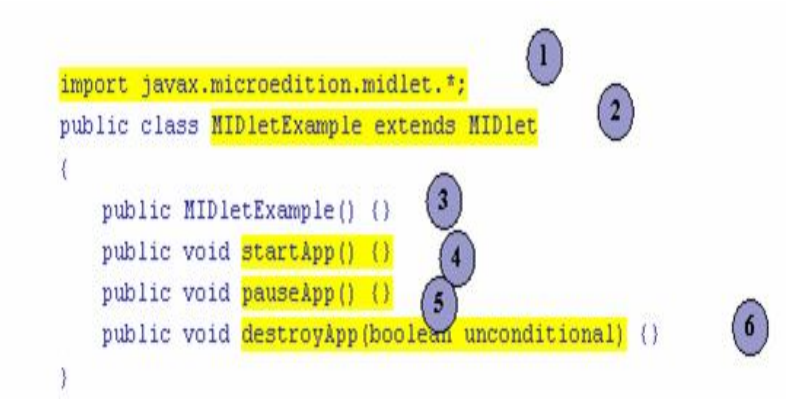
Toàn bộ các lớp, các file cần thiết cho ứng dụng MIDlet sẽ được đóng gói vào trong một bộ được gọi là MIDlet suite, nó bao gồm hai file .JAR và .JAD chuẩn. File lưu trữ .JAR bao gồm các file .class, các file tài nguyên của ứng dụng và một file thống kê manifest mang thông tin mô tả về file JAR này. File .JAD hay còn gọi là bộ mô tả ứng dụng là file chứa thông tin về MIDlet, nó làm hai nhiệm vụ chính:

- Cung cấp thông tin cho bộ quản lý ứng dụng về nội dung file .JAR, qua đó hệ thống sẽ quyết định xem MIDlet này có thích hợp để chạy trên thiết bị hay không.
- Cung cấp phương tiện chuyển tham số cho MIDlet mà không phải thực hiện thay đổi file .JAR

Midlet gồm 3 phương thức chính :

- **Phương thức StartApp(boolean b) :** Là phương thức cho phép lập trình và cấu hình điện thoại di động . Các hành động chức năng chính, các thao tác lên trình ứng dụng chủ yếu được viết trong StartApp. StartApp là phương thức được kích hoạt khởi chạy cùng với ứng dụng trên thiết bị di động. Có thể coi nó chính là phần cốt lõi trong một chương trình viết bằng J2ME cho điện thoại di động.
- **Phương thức PauseApp(Boolean b)** : là một phương thức hỗ trợ công việc sử lý cho người lập trình khi thiết bị di động vì một lý do nào đó ứng dụng cần phải tạm dừng để nhường quyền hoạt động cho một ứng dụng khác. Phương thức PauseApp cho phép người lập trình thực hiện các công việc xử lý dữ liệu , lưu trữ thông tin, cấu hình hệ thống… trước khi ứng dụng được tạm dừng.
- **Phương thức DestroyApp(Boolean b)** : Là một phương thức hỗ trợ người lập trình sử lý dữ liệu,lưu trữ thông tin… như phương thức PauseApp nhưng công việc sử lý này sẽ được thực hiện trước khi đóng ứng dụng chứ không phải là tạm dừng như đối với PauseApp.

Một Midlet đơn giản cần phải có đầy đủ 3 phương thức cơ bản : StartApp,PauseApp,DestroyApp.



Ngoài 3 phương thức cơ bản trên, Midlet còn có các phương thức khác như :

Phương thức	**Mô tả**
abtract void destroyApp (boolean unconditional)	MIDlet chuẩn bị đóng
abtract void pauseApp ( )	MIDlet chuẩn bị dừng
final void startApp( )	Midlet được đặt vào trạng thái kích hoạt
final void notifyDestroyed( )	MIDlet yêu cầu được shutdown
final void notifyPaused( )	MIDlet yêu cầu được dừng lại
final void resumeRequest( )	MIDlet yêu cầu được kích hoạt
final String getAppProperty(String key)	Lấy thuộc tính từ file JAR hoặc JAD

– **Vòng đời của một MIDlet**

Một MIDlet đi qua nhiều chu trình của vòng đời hoạt động và luôn ở trong một trong ba trạng thái sau:

Paused (tạm ngừng): một MIDlet được đặt trong trạng thái paused sau khi phương thức khởi dựng đã được gọi, nhưng trước khi khởi động bởi bộ quản lý ứng dụng. Khi MIDlet đã được khởi động, nó có thể chuyển đổi xen kẽ giữa trạng thái pause và active (kích hoạt) bất kỳ thời điểm nào trong suốt vòng đời của nó.

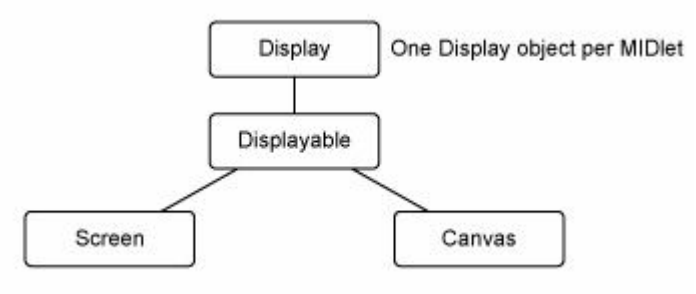
Active (kích hoạt): MIDlet đang chạy.

Destroyed (hủy): MIDlet chấm dứt, giải phóng tài nguyên và bị bộ quản lý ứng dụng đóng lại.

3.2.3 Giao diện người dung

A.Giao diện người dùng cấp cao

- **Đối tượng Display**



Hình 2.16:Phân cấp lớp Display

Mỗi một MIDlet có một tham chiếu tới đối tượng Display, đối tượng này có thể trả về thông tin màn hình hiện thời: phạm vi màu hỗ trợ, độ phân giải…Và chứa những chức năng để truy vấn các đối tượng có thể hiển thị trên màn hình, là các đối tượng Displayable. Đối tượng Display có thể xem là bộ quản lý màn hình điều khiển những thông tin nào sẽ được hiển thị trên thiết bị và hiển thị khi nào. Mặc dù chỉ có duy nhất một đối tượng Display cho một MIDlet, nhưng có thể có nhiều đối tượng Displayable bên trong MIDlet cùng hiển thị ra màn hình.

Chúng ta thường sử dụng các phương thức sau của lớp Display:

- Static Display getDisplay(MIDlet m): lấy về đối tượng Display cho MIDlet.
- Displayable getCurrent(): lấy về đối tượng Displayable hiện thời của màn hình.
- Void setCurrent(Displayable d): hiển thị đối tượng Displayable.

Các hàm API của Display:

Phương thức	**Mô tả**
static Display getDisplay (MIDlet m)	Lấy về đối tượng Display object cho

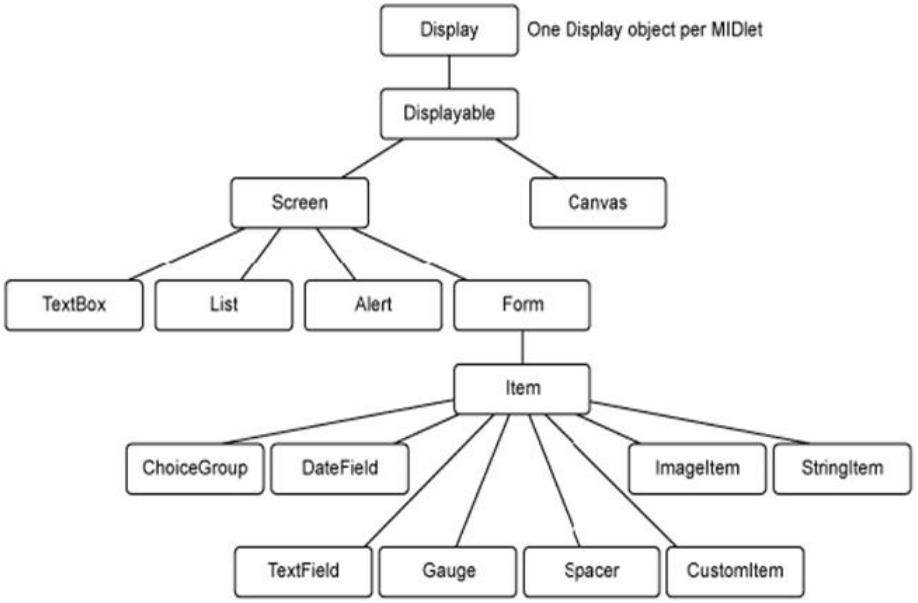
	MIDlet
Displayable getCurrent ( )	Lấy về đối tượng Displayable hiện hành
void setCurrent (Alert alert, Displayable nextDisplayable)	Hiển thị cảnh báo trên đối tượng Displayable hiện hành
void nextCurrent (Displayable nextDisplayable)	Hiển thị một đối tượng Displayable mới
boolean isColor ( )	Thiết bị có hỗ trợ màu
int numColors ( )	Số màu mà thiết bị hỗ trợ
void callSerially ( )	Yêu cầu gọi đối tượng Runnable sau khi vẽ lại

ví dụ về tạo đối tượng Display cho một MIDlet:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet {
        private Display di;
        private Form fr;
        public vd(){
           di=Display.getDisplay(this);
           fr=new Form("vdform");
        }

        public void startApp(){
          di.setCurrent(fr);
        }
        public void pauseApp(){}
        public void destroyApp(boolean un){}
```

}

- **Đối tượng Dislayable và Screen**



Hình 3.17:Phân cấp lớp Displayable

Một đối tượng Displayable là một thành phần được hiển thị trên thiết bị. MIDP chứa 2 lớp con của lớp Displayable là Screen và Canvas.

Một đối tượng Screen không phải là một cái gì đó hiện ra trên thiết bị, mà lớp Screen này sẽ được thừa kế bởi các thành phần hiển thị ở mức cao, chính các thành phần này sẽ được hiển thị ra trên màn hình.

Canvas hay còn gọi là khung vẽ, là thành phần giao diện cấp thấp của J2me. Những thành phần giao diện cấp thấp như Canvas sẽ cho phép ta vẽ trực tiếp lên màn hình để tạo ra các thành phần đồ họa sinh động hay xây dựng Game.

Ngoài MIDlet và màn hình ra, cái cần phải quan tâm khi tạo giao diện cho chương trình đó chính là các đối tượng có thể hiển thị được trên màn hình Displayable, trong các đối tượng này, đối tượng mà chúng ta sẽ sử dụng nhiều nhất chính là Form, Form là một trong những khung chứa cơ bản, chỉ khi tạo được khung chứa cơ bản, người lập trình mới

có thể đưa được các đối tượng giao diện bậc cao mong muốn vào Form và xuất lên màn hình.

- **Thành phần Form và Item**

Một Form giống như một khung chứa, mọi thành phần giao diện mà ta muốn hiển thị ra màn hình đều nằm trong "khung chứa" này, và các phương thức của Form sẽ quyết định hiển thị các thành phần này khi nào và như thế nào.

Một Item là một thành phần giao diện có thể thêm vào Form. Nói chung Một Form chỉ đơn giản là một khung chứa các thành phần, mà mỗi thành phần được thừa kế từ lớp Item, bao gồm: ChoiceGroup, DateField, Gauge, ImageItem, StringItem, TextField.

– **Form**

Một Form không khác gì một cửa sổ, cửa sổ đó có cơ chế cuộn thả, có thể cất giữ bất cứ thành phần giao diện nào.

Form ban đầu được tạo ra sẽ chỉ là một cửa sổ trống, để có thể thêm các thành phần giao diện vào trong Form bạn cần dùng phương thức append của đối tượng Form, phương thức append này sẽ trả về một chỉ số báo nơi thành phần được định vị, thành phần đầu tiên được đưa vào cửa sổ sẽ có chỉ số là 0. Các thành phần giao diện sau khi được đưa vào trong Form sẽ được tự động sắp xếp theo thứ tự từ trái qua phải, từ trên xuống dưới.

Form có các phương thức để xác định số thành phần trên một Form và truy cập chỉ số (vị trí) của một thành phần nào đó hiện có trong Form. Form có những phương thức để chèn (insert), thay thế (replace) và xóa (delete) những đối tượng thành phần. Để có thể chèn, thay thế và xóa các thành phần một cách chính xác thì người lập trình cần xác định chỉ số (vị trí) của thành phần cần xóa, chèn hay thay thế.

Các API thường dùng với Form:

Phương thức	**Mô tả**
Form (String s)	Tạo Form với nhãn là s, nhãn này giống như title của Form.
Form (String s, Item[] items)	Tạo Form và thêm các mục Item trong

	mảng vào Form.
int append(Displayable d)	Thêm thành phần Displayable vào Form.
void delete(int i)	Xóa thành phần Displayable tại vị trí thứ i
void insert(int i1, Item i)	Chèn Item i vào vị trí i
String getLabel( )	Lấy về nhãn cho Item
void setLabel (String s)	Gán nhãn cho Item

Ví dụ tạo ra một Form:

Hình 3.18:

Kết quả thực thi chương trình tạo ta một Form

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet{
    private Display di;
    private Form fr;
```

```
public vd(){
    di=Display.getDisplay(this);
        // tạo Form với nhãn là vdform
    fr=new Form("vdform");
}
public void startApp(){
        // hiển thị form đã được tạo ra trong phương thức khởi dựng
        //ngay khi ứng dụng bắt đầu hoạt động.
    di.setCurrent(fr);
}
public void pauseApp(){
}
public void destroyApp(boolean b){
}
}
```

– **Các thành phần Item**

Item là một lớp trừu tượng, do đó ta không thể tạo những thể hiện của lớp Item. Thay vào đó chúng ta tạo ta các đối tượng từ lớp con của Item. Dưới đây sẽ giới thiệu về các thành phần Item và cách sử dụng chúng.

- **StringItem**

Thành phần StringItem được dùng để hiển thị một nhãn hay chuỗi văn bản. Người dùng không thể thay đổi nhãn hay chuỗi văn bản khi chương trình đang chạy. StringItem không nhận ra sự kiện, nghĩa là người dùng không thể tương tác được với StringItem.
Một StringItem gồm hai phần là phần nhãn (lable) và phần văn bản (text), bạn có thể dùng phương thức riêng cho lable và text để thay đổi hay lấy về nội dung hiện thời của chúng.

Những phương thức hay được sử dụng của lớp StringItem:

Phương thức	**Mô tả**
StringItem(String label, String text)	Phương thức khởi dựng
void setText(String text)	Thay đổi nội dung text của StringItem
void setLable(String lable)	Thay đổi lable của StringItem
String getText( )	Lấy về giá trị hiện thời của text
String getLable( )	Lấy về giá trị hiện thời của lable

Ví dụ tạo ra một StringItem:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class vd extends MIDlet {
   private Display display;
   private Form fmMain;
   private StringItem siMsg;
   public vd() {
      display = Display.getDisplay(this);
      //tạo một đối tượng StringItem có tên là siMsg
     //với nhãn là "tên trường" và văn bản là "khoa công nghệ thông tin"
      siMsg = new StringItem("tên trường: ", "Khoa công nghệ thông tin");
      fmMain = new Form("StringItem Test");
      //thêm đối tượng siMsg vào Form
      fmMain.append(siMsg);
```

```
}
public void startApp() {
   display.setCurrent(fmMain);
}
public void pauseApp() { }
public void destroyApp(boolean unconditional) { }
}
```

Hình 2.19:Kết quả thực thi chương trình tạo ra một StringItem

- **TextField**

Một thành phần TextField là một ô nhập liệu với một dòng duy nhất, giống như các ô nhập tên tài khoản và password bạn vẫn nhìn thấy khi đăng nhập vào các trang web.

TextField có một thuộc tính rất quan trọng đó là ràng buộc (constraint), ràng buộc cho biết người dùng chỉ được phép nhập loại dữ liệu nào vào TextField. Có bốn loại ràng buộc hỗ trợ đặc biệt cho các kiểu định dạng: địa chỉ email, URLs, số và số điện thoại. Bảng dưới đây liệt kê những ràng buộc nhập liệu sẵn có. Ngoài những ràng buộc, khi bạn tạo ra một TextField, bạn có thể chỉ rõ có bao nhiêu ký tự mà bạn cần trong ô nhập liệu.

Ràng buộc	**Kiểu hỗ trợ**
CONSTRAINT_MASK	Sử dụng mặt nạ này khi bạn cần xác định giá trị hiện thời của ràng buộc
ANY	Cho phép tất cả ký tự
EMAIL_ADDR	Cho phép dạng nhập liệu hợp lệ là một địa chỉ email
NUMERIC	Cho phép duy nhất số. Bao gồm số dương lẫn số âm
PASSWORD	Nhập dữ liệu dưới dạng password, các ký tự nhập vào sẽ bị che đi bởi ký tự mặt nạ.
PHONENUMBER	Dạng nhập liệu theo định dạng số điện thoại
URL	Chỉ cho phép nhập các ký tự hợp lệ theo dạng địa chỉ web URL

Phương thức dựng của lớp TextField:

TextField(String label, String text, int maxSize, int constraints). Tạo ra một TextField với nhãn là label, văn bản có sẵn trong TextField là text, số ký tự tối đa là maxSize và dạng dữ liệu được phép nhập vào là constraints.

Sau khi người dùng tương tác vào TextField, bạn sẽ cần lấy lại các thông tin được người dùng nhập vào và xử lý chúng, để làm được việc đó bạn hãy sử dụng các phương thức được giới thiệu sau đây của TextField:

Phương thức	**Mô tả**
TextField (String label, String text, in maxSize, int constraints)	Tạo ra một TextField mới
void delete(int offset, int length)	Xóa các ký tự trong TextField: Với vị trí bắt đầu xóa là offset, và số ký tự cần xóa là length.
void insert(char[] data, in offset, int length, int position)	Chèn ký tự từ mảng vào TextField tại một vị trí xác định
void insert(String src, int position)	Chèn ký tự vào TextField
void setChars(char[] data, int offset, int length, int position)	Thay thế ký tự từ mảng
void setString(String text)	Thay thế nội dung TextField với chuỗi
int getChars(char[] data)	Lấy về nội dung TextField
String getString( )	Lấy về nội dung chứa trong TextField
int getConstraints( )	Lấy về ràng buộc định nghĩa của TextField
void setConstraints(int constraints)	Đặt ràng buộc cho TextField
int getMaxSize( )	Trả về số ký tự tối đa của TextField
int setMaxSize(int maxSize)	Đặt số kí tự tối đa
Int getCaretPosition( )	Lấy vị trí con trỏ nhập liệu
int size( )	Lấy về số ký tự hiện có của TextField

Ví dụ tạo ra một TextField:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet{
    private Display di;
    private Form fr;
    private TextField t1,t2;
    public vd(){
        di=Display.getDisplay(this);
        fr=new Form("vdform");
                //tạo textfield với tên là t1. Nhãn của t1 là "ten :",
                //văn bản có sẵn trong TextField là "abc",
               //số ký tự tối đa là 20, kiểu ràng buộc là ANY
        t1=new TextField("ten:","abc",20,TextField.ANY);
                //tạo textfield với tên là t2. Nhãn của t2 là "pass :",
                //văn bản có sẵn trong TextField là "123",
               //số ký tự tối đa là 20, kiểu ràng buộc là PASSWORD
        t2=new TextField("pass:","123",20,TextField.PASSWORD);
        fr.append(t1);
                fr.append(t2);
    }
    public void startApp(){
        di.setCurrent(fr);
    }
    public void pauseApp(){
    }
    public void destroyApp(boolean b){
    }
    }
```

Hình 3.20:Ví dụ tạo ra các TextField

- **Image & ImageItem**

Hai lớp được dùng để hiển thị hình ảnh là: Image và ImageItem. Image được dùng để tạo ra một đối tượng hình ảnh và giữ thông tin như là chiều cao và chiều rộng của hình ảnh. Lớp ImageItem mô tả một tấm ảnh sẽ được hiển thị như thế nào, ví dụ tấm ảnh sẽ được đặt ở trung tâm, hay đặt về phía bên trái, hay bên trên của màn hình.

MIDP đưa ra 2 loại hình ảnh là loại ảnh không biến đổi và ảnh biến đổi. Một tấm ảnh không biến đổi thì không thể bị thay đổi kể từ lúc nó được tạo ra. Đặc trưng của loại ảnh này là được đọc từ một tập tin. Một tấm ảnh biến đổi về cơ bản là một vùng nhớ. Điều này tùy thuộc vào việc bạn tạo nội dung của tấm ảnh bằng cách ghi nó lên vùng nhớ.

Một vài phương thức khởi dựng hay dùng của lớp Image và ImageItem

Image createImage(String name): Tạo ra một đối tượng hình ảnh từ một ảnh có sẵn, ảnh này nằm trong thư mục src của dự án, và người lập trình chỉ cần đưa ra tên chính xác của ảnh (cả phần mở rộng) để chương trình nạp ảnh đó vào.

Image createImage(Image source)

Image createImage(byte[] imageDate, int imageOffset, int Length)

Image createImage(int width, int height): Tạo ra một ảnh với chiều rộng và chiều cao xác định.

Những phương thức hay được sử dụng của lớp Image:

Phương thức	**Mô tả**
static Image createImage (String name)	Tạo image không thay đổi được từ resource.
static Image createImage (Image resource)	Tạo ảnh từ một ảnh có sẵn.
static Image createImage (byte [] imageData, int imageOffset, intimageLength)	Tạo ảnh dữ liệu mảng.
static ImagecreateImage (int width, int height)	Tạo ảnh theo chiều rộng và cao xác định.
Int getHeight()	Lấy về chiều cao.
Int getWidth()	Lấy về chiều rộng.
Boolean isMutable	Xác định ảnh có thay đổi được hay không.

Chú ý: PNG là loại ảnh được hỗ trợ bởi bất kỳ thiết bị MIDP nào.

Đoạn mã dưới đây mô tả làm thế nào tạo một tấm ảnh từ một tập tin, và gắn nó với một đối tượng ImageItem và thêm một bức ảnh vào một Form:

```
Form fmMain = new Form("Images");
Image img = Image.createImage("/house.png");
```

fmMain.append(new ImageItem(null, img, ImageItem.LAYOUT_CENTER, null));

Đoạn mã dưới đây mô tả việc sử dụng đối tượng Image và đối tượng ImageItem:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet implements CommandListener{
    Display dis;
    Form fr;
    Command c;
    public vd(){
        dis=Display.getDisplay(this);
        fr=new Form("hi you!");
        c=new Command("exit",Command.EXIT,1);
        try{
             //tạo ra một đối tượngImage có tên là i
            //được nạp vào từ một ảnh với  tên là "leaf.png"
        Image i=Image.createImage("/leaf.png");
            //thêm đối tượng ảnh này vào Form
            //ảnh được đặt ở vị trí chính giữa: LAYOUT_CENTER
        fr.append(new
ImageItem(null,i,ImageItem.LAYOUT_CENTER,null));
        dis.setCurrent(fr);
        }
        catch(java.io.IOException e){
             System.out.print("ko nap duoc anh");
        }
        fr.addCommand(c);
        fr.setCommandListener(this);
    }
```

```
    public void startApp(){
        dis.setCurrent(fr);
    }
    public void pauseApp(){ }
    public void destroyApp(boolean b){ }
    public void commandAction(Command c,Displayable db){
        destroyApp(true);
        notifyDestroyed();
    }
}
```

Hình 2.22:Ví dụ tạo Image

Bạn đừng quan tâm tới những đối tượng mà bạn chưa từng học qua, hãy thử để biết cách đưa một ảnh lên màn hình. Bạn hãy copy file leaf.png vào cùng folder với file vd.java, hoặc đổi tên leaf.png trong mã nguồn thành một tên khác, và copy file ảnh có tên như vậy vào cùng thư mục với file vd.java.

- **Thành phần List, TextBox, Alert, Ticket**

Trong phần này chúng ta sẽ xem xét đối tượng List, TextBox, Alert, Ticket trong các thành phần giao diện cấp cao của ứng dụng MIDP, các lớp này cùng cấp với lớp Form, hay nói cách khác chúng đều là mở rộng của lớp Screen, tương tự như lớp Form. Sự khác nhau quan trọng giữa các thành phần này với các thành phần ở mục 3.5.2 là cách hiển thị trên màn hình, Form có thể giữ bất kỳ đối tượng Item nào nhưng đối tượng này thì không, khi đối tượng này được đặt là thành phần hiển thị thì nó là thành phần duy nhất được hiển thị.

– **List**

Một List cũng là một danh sách chọn như ChoiceGroup, tuy nhiên các List đuợc dùng để thể hiện một thực đơn các chọn lựa. List có ba dạng đó là : Đa lựa chọn và chọn loại trừ và danh sách không tường minh.

List đa lựa chọn hay List chọn loại trừ, khi List được tạo ra dưới dạng này thì bên cạnh các nút chọn sẽ có hình ảnh của nút radio hay checkbox, hơn nữa không có sự kiện nào phát sinh khi người dùng thay đổi một phần tử trong List.

Tuy nhiên List không tường minh lại khác, List không tường minh không có nút radio hay checkbox bên cạnh, khi người dùng nhấn vào các nút chọn trong List thì sẽ có sự kiện phát sinh.

Phương thức khởi dựng cơ bản của List:

List (String title, int listType): Tạo ra một List với nhãn là title và kiểu là listType.

Dưới đây là bảng liệt kê các listType có thể có của một đối tượng List:

Giá trị	**Mô tả**
EXCLUSIVE	Duy nhất một lựa chọn được phép
MULTIPLE	Có thể chọn cùng lúc nhiều giá trị
IMPLICIT	Danh sách không tường minh

Các phương thức thường dùng trong lớp List:

Phương thức	**Mô tả**
List (String title, int listType)	Tạo ra một danh sách mới không có phần tử nào
List (String title, int listType, String[] stringElements, Image[] imageElements)	Tạo ra danh sách List từ các mục dữ liệu chứa trong mảng
int append (String stringPart, Image imagePart)	Thêm phần tử
void delete(int elementNum)	Xóa một phần tử tại chỉ số xác định
void insert(int elementNum, String stringPart, Image imagePart)	Chèn phần tử tại chỉ số xác định
void set(int elementNum, String stringPart, Image imagePart)	Thay thế phần tử ở tại một chỉ số xác định
String getString(int elementNum)	Lấy về nội dung văn bản của phần tử tại chỉ số xác định
Image getImage(int elementNum)	Lấy về hình ảnh kết hợp với phần tử ở tại chỉ số xác định

int getSelectedIndex()	Chỉ số hiện hành
void setSelectedIndex(int elementNum, boolean selected)	Đặt chỉ số hiện hành
int getSelectedFlags(boolean[] selectedArray_return)	Lấy về tình trạng của các phần tử
void setSelectedFlags(boolean[] selectedArray)	Đặt tình trạng chọn cho phần tử
boolean isSelected(int elementNum)	Xác định xem phần tử được chọn hay không
int size()	Xác định số phần tử bên trong danh sách List

Đoạn mã dưới đây minh họa việc sử dụng một List – không tường minh:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet{
    Display dis;
    List li;
    public vd(){
        dis=Display.getDisplay(this);
        try{
          Image i[]={
```

```
                    Image.createImage("/down.png"),
                    Image.createImage("/up.png"),
                    Image.createImage("/help.png")
            };
            String s[]={
                    "down",
                    "up",
                    "help"
            };
        li=new List("list",List.IMPLICIT,s,i);
    }
    catch(java.io.IOException e){
            System.out.print("ko load dc anh");
    }
}
public void startApp(){
    dis.setCurrent(li);
}
public void pauseApp(){}
```

```
    public void destroyApp(boolean b){}
}
```

Hình 3.21:Vd tạo một List

- **Xử lý sự kiện**

Phần lớn các đối tượng đồ họa chúng ta đã học từ phần trước đều có khả năng đón nhận các tương tác từ phía người dùng, ví dụ như việc nhập thông tin vào TextField hay việc chọn Item từ một ChoiceGroup hay một List... Mỗi khi một tương tác xảy ra trên một thành phần giao diện, chương trình phải có những hành động trả lời tương ứng với những tương tác đó, ví dụ khi bạn nhấn nút đăng nhập hay đăng xuất trên một website, hệ thống sẽ phải có những hành động để cho phép bạn tham gia hay thoát khỏi hệ thống.

Những tương tác xảy ra trên các thành phần giao diện như vậy chúng ta sẽ gọi là sự kiện và việc tạo ra những hành động trả lời cho các tương tác đó chúng ta sẽ gọi là xử lý sự kiện.

– **Mô hình xử lý sự kiện**

Việc đầu tiên khi xử lý sự kiện đó là chúng ta phải có cơ chế để nhận biết xem sự kiện đó xảy ra ở đâu (trên thành phần giao diện nào) và sau đó là đưa ra các hành động trả lời cho các sự kiện đó. Về cơ bản, chúng ta có ba bước chính để quản lý thành công một sự kiện.

- Phần cứng (thiết bị vật lý) phải đoán nhận được điều gì đó đã xuất hiện: Một nút đã được nhấn hay một nút được thả ra, dây nối pin nguồn adapter đang cắm vào hay tháo ra.
- Phần mềm trên thiết bị (bộ quản lý ứng dụng) cần được thông báo về sự kiện này.
- Đây là điểm nơi chúng ra sẽ ứng dụng vai trò là một người phát triển MIDlet. Một thông báo từ bộ quản lý ứng dụng sẽ được gửi cho MIDlet. Thông báo này sẽ chứa thông tin về sự kiện sao cho chúng ta có thể đưa ra những quyết định xử lý (sự kiện có thể là một yêu cầu hiển thị thông báo giúp đỡ trên màn hình thiết bị)

Để MIDlet có thể nhận biết được các sự kiện chúng ta phải gắn vào nó một bộ lắng nghe sự kiện, còn được gọi với thuật ngữ là Listener. Có hai giao diện Listener chính cho mỗi MID Profile là: CommandListener và ItemListener.

Khi MIDlet đã nhận biết được sự kiện rồi, việc tiếp theo người lập trình cần phải làm là đưa ra các hành động trả lời, để làm được việc đó bạn cần một lớp cài đặt hoặc cả hai giao diện interface này. Khi cài đặt interface, bạn sẽ xây dựng nội dung cho phương thức commandAction() và itemStateChanged(). Đây là nơi bạn đặt mã để thực hiện những công việc đúng với yêu cầu của sự kiện xuất hiện.

– **Đối tượng COMMAND**

Trong phần này chúng ta sẽ cùng nhau tìm hiểu cách tạo ra và sử dụng một thành phần đồ họa rất đơn giản, nhưng gần như không thể thiếu trong các giao diện phần mềm đó là nút nhấn. Nút nhấn trong J2me được gọi là Command, nút nhấn có hai loại:

- **Nút phím** thường là những nút nằm ở góc dưới cùng bên trái và bên phải trên màn hình chương trình (nằm trên thanh Menu ở cuối màn hình), gọi nút này là nút phím bởi vì nó có khả năng ánh xạ lên các nút thật trên bàn phím điện thoại.
- Những nút không thể tham chiếu đến các nút thật trên bàn phím gọi là **nút mềm.** Để nhấn các nút này người dùng chỉ cần nhấn nút **Ok/Menu** trên bàn phím điện thoại (phím tương đương với phím **Enter** trên bàn phím máy vi tính).

Việc xử lý sự kiện cho những nút này gồm các bước:

1. Tạo ra các đối tượng Command.
2. Đưa Command vào một Form, TextBox, List hoặc Canvas.
3. Thêm một CommandListener vào Form, TextBox, List hoặc Canvas ở bước trên.

Khi bắt được một sự kiện, Listener sẽ được gọi (gửi một thông báo sự kiện từ bộ quản lý ứng dụng đến Listener). Kết quả là phương thức commandAction() mà Listener quản lý sẽ được gọi. Bên trong phương thức này bạn có thể xác định Command nào bắt đầu hoạt động và xử lý sự kiện tương ứng.

Dưới đây là ví dụ cho phép tạo ra một nút thoát khỏi MIDlet:

```
import javax.microedition.lcdui.*;

import javax.microedition.midlet.*;
```

```
//cài đặt giao tiếp lắng nghe sự kiện CommandListener
public class vd extends MIDlet implements CommandListener{
    private Display di;
    private Form fr;
    private Command c1;
    public vd(){
        di=Display.getDisplay(this);
        fr=new Form("vdform");
            //Tạo ra nút nhấn
        c1=new Command("exit",Command.EXIT,1);
            //thêm nút nhấn vào form
        fr.addCommand(c1);
            //gắn bộ lắng nghe sự kiện vào form
        fr.setCommandListener(this);
    }
    public void startApp(){
        di.setCurrent(fr);
    }
    public void pauseApp(){}
```

```
public void destroyApp(boolean b){ }

public void commandAction(Command com,Displayable db){

        //xác định Command gây ra sự kiện và xử lý

    if(com==c1){

        destroyApp(true);

        notifyDestroyed();

    }

    }

}
```

– **Command và CommandListener**

Khi tạo một đối tượng lệnh Command mới chúng ta có ba tham số: nhãn (label), kiểu (type) và quyền ưu tiên (priority).

phương thức khởi dựng:

- Command (String label, int commandType, int priority): Tạo một nút nhấn có nhãn là label, loại nút nhấn là commandType, quyền ưu tiên là priority.

ví dụ:

- c1=new Command("exit",Command.EXIT,1);

Trong đó:

1. Nhãn (label): Đây là phần chỉ rõ nội dung văn bản mà bạn kết hợp với Command. Nhãn có thể được hiển thị trực tiếp trên màn hình thiết bị hoặc hiển thị bên trong một menu.

2. Kiểu (type): Xác định xem Command là nút mềm hay nút phím, nếu là nút phím thì là loại nút phím nào?. Dưới đây là bảng các giá trị kiểu của Command:

Giá trị	**Mô tả**
BACK	Yêu cầu di chuyển tới màn hình trước đó
CANCEL	Yêu cầu hủy bỏ một thao tác. Ví dụ, khi hiển thị một màn hình nhắc nhở nhập vào một địa chỉ Web, bạn có thể có cả nút OK lẫn Cancel làm những nút tùy chọn trên màn hình
EXIT	Yêu cầu thoát khỏi MIDlet
HELP	Yêu cầu hiển thị thông tin giúp đỡ
ITEM	Yêu cầu ánh xạ trên Command lên một Item trên màn hình. Ví dụ, khi sử dụng thành phần danh sách List, bạn có thể mô phỏng hoạt động của một menu ngữ cảnh bằng cách ánh xạ lệnh Command cho từng mục trong danh sách List.
OK	Xác nhận ra quyết định từ phía người dùng. Ví dụ, sau khi dữ liệu tải xuống, bạn có thể đưa ra một màn hình thông báo "Download completed" với một lệnh kiểu OK.
SCREEN	Dành cho những lệnh không có ánh xạ lên các phím đặc biệt. Ví dụ, bạn có thể có những lệnh khởi động quá trình chuyển tập tin từ máy di động sang máy tính lớn hay tải xuống dữ liệu từ Internet hoặc từ các máy PC. Nhãn "Upload" và "Download" không có phím ánh xạ trực tiếp trên thiết bị.
STOP	Yêu cầu dừng một thao tác. Ví dụ, nếu đang tải dữ liệu xuống từ mạng Internet, tùy chọn này có thể hiện sẵn giúp người dùng có thể

	chấm dứt quá trình download mà không phải đợi thao tác hoàn tất

3. Quyền ưu tiên: Quyền ưu tiên được xác định bằng một con số, số có giá trị càng cao thì quyền ưu tiên càng thấp. Những giá trị này có thể giúp ích cho bộ quản lý ứng dụng khi sắp xếp các Item xuất hiện trong menu lựa chọn hoặc sắp thứ tự các nút mềm trên màn hình. Quyền ưu tiên này làm một yêu cầu do bạn chỉ định. Phím thực sự được ánh xạ và quyền ưu tiên trên phím thực tế sẽ được quyết định và điều bởi thiết bị.

Những phương thức thường được dùng của Command:

Phương thức	**Mô tả**
Constructor	
Command (String label, int commandType, int priority)	Tạo ra đối tượng Command
Phương thức	
int getCommandType()	Lấy về kiểu gán cho lệnh
String getLabel()	Lấy về nhãn gán cho lệnh
int getPriority()	Lấy về quyền ưu tiên gán cho lệnh

Phương thức của CommandListener:

Phương thức	**Mô tả**
void commandAction(Command c, Displayable d)	Được gọi khi lệnh Command "c" trên một đối tượng Displayable "d" bắt đầu một sự kiện

ví dụ dưới đây sẽ mô tả cách sử dụng Command và CommandListener:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet implements CommandListener{
    private Display di;
    private Form fr;
        private TextBox tb;
    private Command c1,c2,c3;
    public vd(){
        di=Display.getDisplay(this);
        fr=new Form("vd");
         tb=new TextBox("help","Đây là nội dung help...",50,0);
                //tạo ra các nút nhấn
        c1=new Command("exit",Command.EXIT,1);
         c2=new Command("help",Command.HELP,2);
         c3=new Command("back",Command.BACK,3);
                //thêm các nút nhấn vào form
        fr.addCommand(c1);
         fr.addCommand(c2);
         tb.addCommand(c3);
                //gắn bộ lắng nghe sự kiện vào các thành phần khung chứa
         fr.setCommandListener(this);
```

```
      tb.setCommandListener(this);
}
public void startApp(){
     di.setCurrent(fr);
}
public void pauseApp(){ }
public void destroyApp(boolean b){ }
public void commandAction(Command com,Displayable db){
     if(com==c1){
        destroyApp(true);
        notifyDestroyed();
     }
          else
          if(com==c2){
               di.setCurrent(tb);
          }
          else
          if(com==c3){
               di.setCurrent(fr);
          }
 }
```

}

Hình 2.29:ví dụ về Command và CommandListener

Đây là một ví dụ đơn giản, giao diện chính của chương trình được gắn với hai nút exit và help, exit để thoát khỏi MIDlet, help để bật lên một cửa sổ trợ giúp dạng TextBox, cửa sổ help này được gắn với một nút back để quay lại giao diện chính của chương trình.

- Để tạo được chương trình như trên, đầu tiên ta phải tạo ra các nút nhấn:

c1=new Command("exit",Command.EXIT,1);

c2=new Command("help",Command.HELP,2);

c3=new Command("back",Command.BACK,3);

- Rồi gắn các nút nhấn vào các đối tượng khung chứa thích hợp:

```
fr.addCommand(c1);

fr.addCommand(c2);

tb.addCommand(c3);
```

- Gắn đối tượng lắng nghe sự kiện vào các khung chứa. Vì ở đây MIDlet của chúng ta được cài đặt để thực thi giao diện CommandListener (implements CommandListener) cho nên đối tượng lắng nghe sự kiện cũng chính là MIDlet:

```
fr.setCommandListener(this);

tb.setCommandListener(this);
```

- Bước cuối cùng là xử lý sự kiện:

```
public void commandAction(Command com,Displayable db){
    if(com==c1){
        destroyApp(true);
        notifyDestroyed();
        }
            else
            if(com==c2){
                di.setCurrent(tb);
            }
            else
            if(com==c3){
                di.setCurrent(fr);
            }
     }
```

Ở đây để nhận biết được đối tượng Command nào là đối tượng gây ra sự kiện, chúng ta sử dụng tham số **com** của phương thức CommandAction với câu lệnh kiểm tra đơn giản như sau:

```
if (com==tên_Command){
//thực hiện công việc tương ứng
}
```

- **Item và ItemStateListener**

Một kiểu xử lý sự kiện thứ hai được thực hiện thông qua đối tượng Item. Item chỉ có thể truy xuất từ bộ phận của Form trong khi lệnh Command có thể truy xuất được từ Form, TextBox, List, hoặc Canvas.

Khi bạn thêm một Item vào Form, tượng tự như với Command, bạn phải thêm một bộ ItemStateListener vào chung để lắng nghe sự kiện.

Khi có sự thay đổi tác động đến Item (như Gauge tăng mức đếm hoặc DateField thay đổi dữ liệu), đối tượng ItemStateListener sẽ được thông báo bằng một thông điệp.

Bạn có thể xem thông báo này như một lời gọi tới phương thức itemStateChanged(). Phương thức này có thể giải mã thông điệp xem Item đã thay đổi những gì và xử lý những thay đổi này như thế nào nếu cần thiết.

Lưu ý:StringItem và ImagItem cũng là những lớp con của Item. Tuy nhiên, khi được cấp phát, những đối tượng này là tĩnh và do đó không nhận được thông báo về sự kiện phát sinh.

Ví dụ dưới đây sẽ cho phép tạo ra một TextField và một StringItem. Bất cứ khi nào người dùng thay đổi nội dung của TextField thì nội dung của StringItem cũng bị thay đổi theo:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet implements CommandListener,
ItemStateListener{
    private Display di;
    private Form fr;
        private TextField tf;
        private StringItem st;
    private Command c1;
    public vd(){
        di=Display.getDisplay(this);
        fr=new Form("vd");
        c1=new Command("exit",Command.EXIT,1);
                //TextField và StringItem là các đối tượng Item
                tf=new TextField("họ tên:","",50,TextField.ANY);
                st=new StringItem("Xin chào:","");
                fr.append(tf);
                fr.append(st);
```

```
    fr.addCommand(c1);
    fr.setCommandListener(this);
            //gắn bộ lắng nghe vào form chứa Item
            fr.setItemStateListener(this);
}
public void startApp(){
    di.setCurrent(fr);
}
public void pauseApp(){         }
public void destroyApp(boolean b){ }
public void commandAction(Command com,Displayable db){
    if(com==c1){
       destroyApp(true);
       notifyDestroyed();
    }
}
  //xử lý sự kiện
  public void itemStateChanged(Item it){
```

```
        if(it==tf){
            st.setText(tf.getString());
        }
    }
}
```

Để cho mỗi khi TextField thay đổi nội dung, nội dung của StringItem cũng bị thay đổi thì ta phải gắn bộ lắng nghe sự kiện vào TextField:

```
fr.setItemStateListener(this);
```

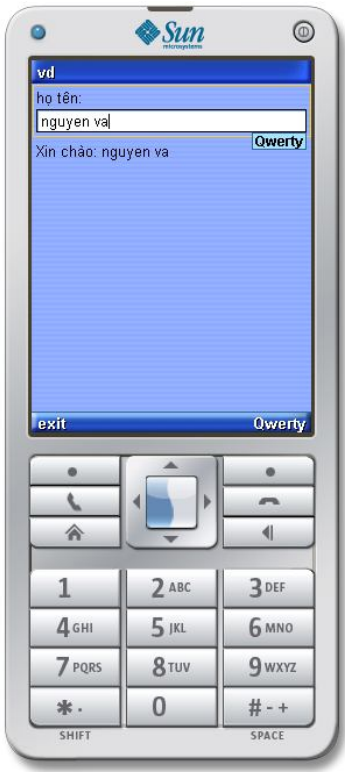
Bộ lắng nghe hay đối tượng lắng nghe sự kiện ở đây là MIDlet (this) của chúng ta vì MIDlet được cài đặt để thực thi giao diện ItemStateListener (implements CommandListener, ItemStateListener).

Sau đó tiến hành xử lý sự kiện trong phương thức itemStateChanged:

```
public void itemStateChanged(Item it){
        if(it==tf){
            st.setText(tf.getString());
        }
    }
```

Tương tự như xử lý với CommandListener, ở đây chúng ta cũng sử dụng tham số **Item** của phương thức itemStateChanged để xác định đối tượng Item đã gây ra sự kiện với câu lệnh kiểm tra đơn giản:

```
if(it==tên_Item){
        //đoạn mã xử lý tương ứng
    }
```



Hình 2.30:Ví dụ sử dụng ItemStateListener

Trừ StringItem và ImageItem, các Item khác đều có thể nhận biết tương tác với người dùng. Khi thêm Item vào Form, bạn tạo một Listener để đón bắt sự kiện người dùng (cho

tất cả các Item trên Form). Khi có sự thay đổi phát hiện, phương thức itemStateChanged() sẽ được gọi. Bên trong phương thức này bạn có thể xác định được thay đổi diễn ra trên Item nào và quyết định cách xử lý chúng. Không cần phải gọi itemStateChanged() mỗi khi có thay đổi diễn ra. Tuy nhiên, nó theo những quy tắc sau:

- Nếu một Item đã thay đổi thì itemStateChanged() phải được gọi trên Item thay đổi trước khi thông báo về thay đổi của mình cho những Item kế tiếp
- Nếu một MIDlet thực hiện thay đổi cho một Item (theo cách tương tự như tương tác người dùng) thì itemStateChanged() sẽ không được gọi. ví dụ, nếu bạn viết mã bên trong MIDlet thay đổi giá trị của DateField, điều này sẽ không làm phát sinh sự kiện.
- Nếu thiết bị chạy MIDlet có thể nhận biết khi người dùng đã đi chuyển từ Item này sang Item khác (thay đổi focus), itemStateChanged() sẽ phải được gọi khi bắt đầu rời bỏ Item hiện hành trước khi đến Item tiếp theo.

phương thức của ItemStateListener:

Phương thức	**Mô tả**
void itemStateChanged(Item item)	Được gọi khi Item thay đổi

– **Xử lý sự kiện với các đối tượng giao diện**

Các phần trên đã giới thiệu tới bạn cách tạo ra các thành phần giao diện và cách xử lý sự kiện, để chi tiết hơn, dưới đây giáo trình sẽ đưa ra các ví dụ về cách dùng các phương thức của các đối tượng đồ họa và cách xử lý sự kiện đối với các đối tượng đó.

A) TextField

Chúng ta sẽ cùng nhau quay lại ví dụ trong phần ItemStateListener để nói về cách xử lý sự kiện với TextField. TextField là một Item cho nên ta hoàn toàn có thể dùng ItemStateListener với nó. TextField có rất nhiều phương thức để xử lý trực tiếp các thông tin người dùng nhập vào ngay trên ô nhập thông tin, bạn hãy thử sử dụng các phương

thức đó để thấy được sự linh hoạt của TextField, còn bây giờ, chúng ta hãy làm một ví dụ đơn giản với việc lấy lại thông tin trên TextField ngay thời điểm người dùng nhập vào bằng đối tượng ItemSateListener:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class vd extends MIDlet implements CommandListener,
ItemStateListener{
    private Display di;
    private Form fr;
        private TextField tf;
        private StringItem st;
    private Command c1;
    public vd(){
        di=Display.getDisplay(this);
        fr=new Form("vd");
        c1=new Command("exit",Command.EXIT,1);
                tf=new TextField("họ tên:","",50,TextField.ANY);
                st=new StringItem("Xin chào:","");
                fr.append(tf);
```

```
            fr.append(st);
    fr.addCommand(c1);
    fr.setCommandListener(this);
            fr.setItemStateListener(this);
}
public void startApp(){
    di.setCurrent(fr);
}
public void pauseApp(){          }
public void destroyApp(boolean b){ }
public void commandAction(Command com,Displayable db){
    if(com==c1){
      destroyApp(true);
      notifyDestroyed();
    }
}
  public void itemStateChanged(Item it){
     if(it==tf){
```

```
            st.setText(tf.getString());
        }
      }
    }
```

TextField là một Item mà ItemStateListener là Listener dành riêng cho các Item, vì vậy khi TextField có bất cứ tương tác gì từ phía người dùng, nó đều được đón nhận từ ItemStateListener. Kết quả, bạn sẽ thấy là bất cứ điều gì xảy ra với nội dung trong ô văn bản của TextField sẽ đều được lặp lại ở phần Text của StringItem.

b) List

List không giống các thành phần đã giới thiệu ở trên, List không phải là một Item, do đó ta chỉ có thể dùng bộ lắng nghe CommandListener với List. Vậy dùng CommandListener với List như thế nào?, chúng ta sẽ cùng xét ví dụ dưới đây với một List không tường minh:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.IOException;
public class vd extends MIDlet implements CommandListener{
    Display dis;
    List li;
    Command c1;
    public vd(){
```

```
        dis=Display.getDisplay(this);
        li=new List("list",List.IMPLICIT);
        li.append("muc1",null);
        li.append("muc2",null);
        li.append("muc3",null);
        li.append("muc4",null);
        li.append("muc5",null);
        c1=new  Command("exit",Command.EXIT,1);
        li.addCommand(c1);
        li.setCommandListener(this);
    }
    public void startApp() {
        dis.setCurrent(li);
    }
    public void pauseApp(){ }
    public void destroyApp(boolean b){ }
    public void commandAction(Command c,Displayable db){
        boolean b[]=new boolean[li.size()];
        int i;
```

```
        if(c==c1){
                destroyApp(true);
                notifyDestroyed();
        }
        else
                 if(c==li.SELECT_COMMAND){
        switch(li.getSelectedIndex()){
                    case 0: System.out.println("muc 1");break;
                    case 1: System.out.println("muc 2");break;
                    case 2: System.out.println("muc 3");break;
                    case 3: System.out.println("muc 4");break;
                    case 4: System.out.println("muc 5");break;
                 }
            }
      }
}
```

Ví dụ này cũng không có gì đặc biệt, chúng ta chỉ đơn giản là đưa ra màn hình output của netbeans tên của thành phần trong List được lựa chọn. Để làm được việc đó, trong phương thức cài đè commandAction ta phải kiểm tra xem có phải List đã gây ra sự kiện không:

```
if(c==li.SELECT_COMMAND)
```

Vì là Listener của các Command nên ở đây ta phải dùng thuộc tính tĩnh SELECT_COMMAND với kiểu trả về là một Command của List để kiểm tra thành phần nào của List được chọn. Sau đó ta chỉ cần hiển thị tên của thành phần đó lên là được:

```
switch(li.getSelectedIndex()){
                case 0: System.out.println("muc 1");break;
                case 1: System.out.println("muc 2");break;
                case 2: System.out.println("muc 3");break;
                case 3: System.out.println("muc 4");break;
                case 4: System.out.println("muc 5");break;
            }
```

Trong đó hàm getSelectedIndex() sẽ trả về chỉ số của thành phần List đã được người dùng chọn lựa.

Hình :Ví dụ xử lý sự kiện với List

2.2.5 Kết nối internet với J2ME

A. Khung kết nối chung GCF

CLDC cung cấp cho chúng ta một khung tổng quát để thiết lập kết nối mạng, đó là GCF. Ý tưởng của nó là: định nghĩa một cách trừu tượng các hoạt động mạng và nhập xuất file để dùng chung cho một số lượng lớn các thiết bị, giống như một bộ khung nền vậy. Ở mức độ CLDC, J2me chỉ định nghĩa và tạo kết nối, giao thức thực sự và các phương thức trao đổi dữ liệu của giao thức đó được trao cho tầng Profiles.

Với GCF, để kết nối chúng ta sử dụng một lớp có khả năng mở mọi loại kết nối bao gồm: file, http, datagram, … Tên của lớp này là Connector. Như vậy nếu sử dụng

Connector để mở kết nối, chúng ta chỉ cần gọi một phương thức open cố định dạng như sau:

Connector.Open(“protocol:address; parameter”)

Cơ chế mà GCF dùng để mở nhiều loại giao tiếp chỉ bằng một phương thức chung duy nhất này đã chứng minh tính uyển chuyển của GCF. Cơ chế này hoạt động như sau:

1. Trong thời gian thực thi, mỗi khi có yêu cầu mở một giao thức, Connector sẽ tìm đến lớp tương ứng cài đặt giao thức ấy. Quá trình tìm kiếm này được thực hiện thông qua phương thức Class.forName(). Ví dụ như để yêu cầu mở kết nối HTTP trong J2ME, yêu cầu đó sẽ được viết như sau: Class.forName(“com.sun.midp.io.j2me.http.Protocol”);
2. Khi tìm thấy lớp tương ứng, Class.forName() sẽ trả về một đối tượng có cài đặt giao diện Connection (trong đó lớp Connector và giao diện Connection đã được định nghĩa sẵn trong CLDC). Sau khi kết nối thành công, mọi công việc giao tiếp, trao đổi dữ liệu sẽ đều được thực hiện với đối tượng cài đặt giao diện Connection này.

Ta cần nhớ rằng: cài đặt thật sự của các giao thức đều nằm ở mức profiles. Trong MIDP 1.0, HttpConnection hỗ trợ một tập con HTTP phiên bản 1.0. Do đó khi lớp này mở rộng ContentConnection, nó đã được cung cấp sẵn hơn 20 phương thức chuyên biệt để giao tiếp thông qua giao thức HTTP.

Ví dụ, để tạo một kết nối HTTP ta sẽ làm như sau:

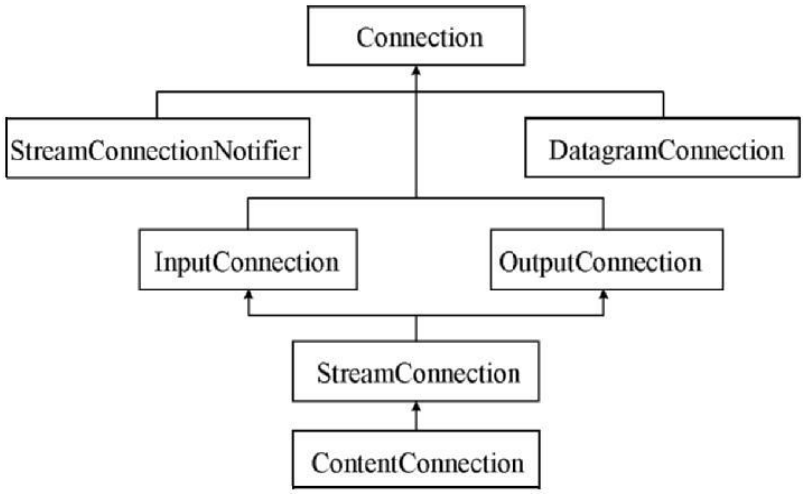
```
//Tạo chuỗi kết nối

String url = "http://www.corej2me.com";

// HttpConnection là một đối tượng thuộc MIDP

// lớp cài đặt thực sự giao thức HTTP
```

```
HttpConnection http=null;

//Dùng đối tượng Connector để mở kết nối, phương thức open của đối

//tượng này trả về một đối tượng Connection.

//Ép kiểu đối tượng Connection thành HttpConnection

HttpConnection http = (HttpConnection) Connector.open(url);
```

B. Các giao thức được hỗ trợ trong GCF

Như ta đã biết, các thiết bị di động nói chung và điện thoại di động nói riêng có khả năng kết nối rất lớn, bởi vì nó được người sử dụng dùng để trao đổi với thế giới bên ngoài tại bất cứ nơi đâu và bất cứ lúc nào. Những khả năng kết nối ấy không chỉ là khả năng thoại hay SMS mà còn là kết nối không dây. Do đó, Để J2me có thể hỗ trợ một số lượng lớn các thiết bị di động khác nhau với nhiều khả năng kết nối và yêu cầu nhập xuất khác nhau GCF đã được thiết kế. GCF định nghĩa kỹ thuật kết nối chung nhất có thể cho mọi loại thiết bị, được dùng chung bởi cả hai loại cấu hình là CDC và CLDC, vì vậy GCF hỗ trợ rất nhiều loại giao thức, nhiều kiểu kết nối khác nhau, ví dụ: Giao thức HTTP, FTP, UDP, Datagram, Socket... Nhưng đối với điện thoại di động, đối với hiện trạng MIDP, giao thức luôn luôn được hỗ trợ là giao thức HTTP, vì vậy trong giáo trình này chúng ta sẽ học cách tạo kết nối HTTP với J2me.

Để bắt đầu, chúng ta sẽ cùng xem GCF cung cấp cho ta những công cụ gì để tiến hành kết nối và trao đổi dữ liệu thông qua đối tượng Connection, đối tượng mà chúng ta đã được làm quen ở mục trên.



Hình 2.42:Phân cấp lớp Connection

Sau đây là mô tả các giao diện kết nối được định nghĩa trong CLDC:

- Giao diệnStreamConnectionNotifier: được dùng khi đợi một kết nối phía server được thiết lập. Phương thức acceptAndOpen() bị chặn cho đến khi client thiết lập kết nối.
- Giao diện InputConnection dùng để thực hiện một luồng nhập tuần tự dữ liệu chỉ đọc.
- Giao diện OutputConnection dùng để thực hiện một luồng xuất dữ liệu chỉ viết.
- Giao diện StreamConnection là kết hợp của cả hai giao diện InputConnection và OutputConnection. Nó dùng cho các thiết bị di động có truyền thông hai chiều.
- Giao diện ContentConnection kế thừa giao diện StreamConnection và thêm vào các phương thức getType(), getEncoding(), và getLength(). Nó cung cấp cơ sở cho giao diện HttpConnection của MIDP.

- Giao diện HttpConnection được định nghĩa trong MIDP và kế thừa giao diện ContentConnection của CLDC. Giao diện này cung cấp các phương thức thiết lập một kết nối HTTP.

Việc sử dụng những đối tượng này như thế nào, trong trường hợp nào để tiến hành giao tiếp và trao đổi dữ liệu chúng ta sẽ được học trong các ví dụ cụ thể dưới đây.

C. Hỗ trợ giao thức HTTP trong MIDP

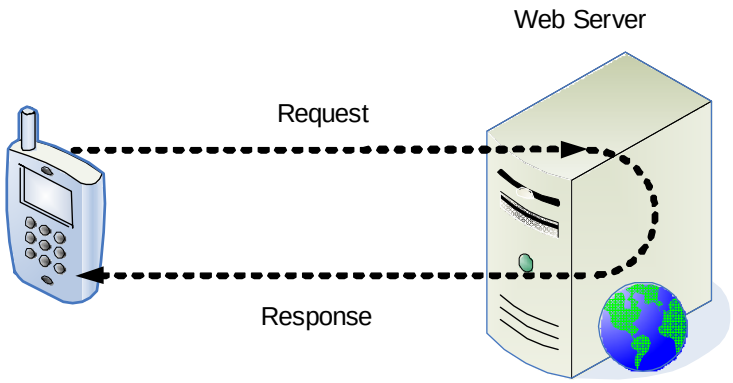
Đối với công nghệ J2ME, cần phải quan tâm đến sự hạn chế của cả kết nối mạng và tài nguyên của thiết bị, không giống như môi trường thông thường của máy tính cá nhân với kết nối mạng cố định. Điều này có nghĩa là nhà phát triển nên lường trước được các khoảng thời gian trễ dài trên băng thông hạn chế. Hơn nữa, bất kỳ trong tình huống nào cũng không nên cho rằng thiết bị di động luôn luôn có kết nối. Về tài nguyên, ta phải đối mặt với vấn đề khả năng tính toán hạn chế cùng với khả năng lưu trữ tương đối của thiết bị. Do đó, trước khi phát triển một ứng dụng phân tán cho client di động, ta cần phải xem xét kỹ các yếu tố trước khi chọn giao thức, bởi vì quyết định này có thể có ảnh hưởng lớn đến hiệu suất của ứng dụng.

HTTP là một giao thức liên lạc client/server lý tưởng cho ứng dụng Java di động. Đối với mỗi đặc tả, thiết bị tương thích MIDP 1.0 phải hỗ trợ HTTP. Các giao thức khác như TCP hay UDP là tùy chọn. Bởi vì không phải tất cả thiết bị MIDP đều hỗ trợ truyền thông socket hay datagram, do đó triển khai HTTP trên thiết bị di động cho phép tối ưu khả năng chuyển đổi giữa các thiết bị từ các nhà sản xuất khác nhau.

Một lợi điểm khác nữa là giao thức HTTP được hưởng truy xuất không lỗi (trouble-free access) thông qua tường lửa. Bởi vì server và client di động hầu như được tách biệt bằng firewall, HTTP không cần phải cấu hình thêm. Mặc dù vậy, ta cũng nên quan tâm đến các rủi ro bảo mật có thể có khi mở kết nối HTTP ra thế giới bên ngoài. Java cung cấp API lập trình mạng, hỗ trợ giao thức HTTP 1.1. Ta dễ dàng tạo ra các request GET, POST, và HEAD trong ứng dụng Java.

- **Sơ lược về giao thức HTTP**

HTTP là giao thức truyền siêu văn bản và là giao thức tầng ứng dụng cho web. Nó hoạt động theo mô hình Client/Server.



Hình 3.2:

Mô hình Client - Server

- **Client:** Yêu cầu, nhận, hiển thị các đối tượng Web.
- **Server:** Web server gửi các đối tượng trả lời cho các yêu cầu từ phía Client.

HTTP là giao thức phi trạng thái, trong đó Server không lưu lại các yêu cầu của client. Điều này có nghĩa là: sau mỗi lần yêu cầu được gửi lên Server và Server kết xuất kết quả trả về cho Client thì kết nối sẽ hoàn toàn bị ngắt, để tiếp tục yêu cầu Server trả lời cho các yêu cầu khác, Client lại phải tiến hành kết nối lại từ đầu.

HTTP sử dụng giao thức TCP của tầng giao vận. Các bước tiến hành từ khi Client kết nối tới Server sau đó gửi và nhận kết quả từ Server gửi về như sau:

- Client khởi tạo kết nối TCP với Server
- Server chấp nhận kết nối TCP từ Client
- các thông điệp HTTP được trao đổi giữa browser và web server.

- đóng kết nối TCP

Có hai kiểu thông điệp HTTP là yêu cầu (Request) và trả lời (Response). Các thông điệp được định dạng kiểu mã ASCII.

Cả HTTP và HTTPS đều gửi request và response. Máy Client gửi request, còn Server sẽ trả về response.

– **Client Request**

Client request bao gồm 3 phần sau:

- Request method
- Header
- Body

Request method định nghĩa cách mà dữ liệu sẽ được gửi đến Server. Có 3 phương thức được cung cấp sẵn là GET, POST, HEADER. Khi sử dụng Get, dữ liệu cần request sẽ nằm trong URL. Với Post dữ liệu gửi từ client sẽ được phân thành các stream riêng biệt. Trong khi đó, Header sẽ không gửi dữ liệu yêu cầu lên server, thay vào đó header chỉ request những meta information về server. GET và POST là hai phương thức request khá giống nhau, tuy nhiên do GET gửi dữ liệu thông qua URL nên sẽ bị giới hạn, còn POST sử dụng những stream riêng biệt nên sẽ khắc phục được hạn chế này.

Ví dụ về việc mở HTTP Connection thông qua GET

```
String url = "http://www.corej2me.com?size=large";

HttpConnection http = null;

http = (HttpConnection) Connector.open(url);

http.setRequestMethod(HttpConnection.GET);
```

Những **Header field** sẽ cho phép ta truyền các tham số từ Client đến Server. Các header field thường dùng là If-Modified-Since, Accept, and User Agent. Bạn có thể đặt các field này thông qua phương thức setRequestProperty(). Dưới đây là ví dụ dùng setRequestProperty(), chỉ có những dữ liệu thay đổi sau ngày 1 tháng 1 năm 2005 mới được gửi về từ server:

```
String url = "http://www.corej2me.com\somefile.txt";

HttpConnection http = null;

http = (HttpConnection) Connector.open(url);

http.setRequestMethod(HttpConnection.GET);

// Đặt header field

http.setRequestProperty("If-Modified-Since", "Sat, 1 Jan 2005 12:00:00 GMT");
```

Body chứa nội dung mà bạn muốn gửi lên server. Ví dụ về sử dụng POST và gửi dữ liệu từ client thông qua stream:

```
String url = “http://www.corej2me.com”;

tmp = "test data here";

OutputStream ostrm = null;

HttpConnection http = null;

http = (HttpConnection) Connector.open(url);

http.setRequestMethod(HttpConnection.POST);

// Gửi dữ liệu
```

```
ostrm = http.openOutputStream();

byte bytes[] = tmp.getBytes();

for(int i = 0; i < bytes.length; i++) {

    os.write(bytes[i]);

}

os.flush();
```

Sau khi nhận được và xử lý yêu cầu từ phía client, server sẽ đóng gói và gửi về phía client.

- **Server Response**

Cũng như client request, server cũng gồm 3 phần sau:

- Status line
- Header
- Body

Status line sẽ thông báo cho client kết quả của request mà client gửi cho server. HTTP phân loại status line thành các nhóm sau đây:

- 1. xx Thông tin
- 2. xx Thành công
- 3. xx Chuyển hướng
- 4. xx Máy khách Client lỗi
- 5. xx Máy chủ Server lỗi

Status line bao gồm version của HTTP trên server, status code, và đoạn text đại diện cho status code.

Ví dụ: "HTTP/1.1 200 OK" "HTTP/1.1 400 Bad Request" "HTTP/1.1 500 Internal Server Error" 102

Không giống như **header** của client, server có thể gửi data thông qua header. Sau đây là những phương thức dùng để lấy thông tin Header mà server gửi về:

- String getHeaderField(int n), lấy giá trị header field thông qua chỉ số
- String getHeaderField(String name), lấy giá trị header field thông qua tên
- String getHeaderFieldKey(int n), lấy header field key thông qua chỉ số

Server có thể trả về nhiều Header field. Trong trường hợp này, phương thức đầu tiên sẽ cho lấy header field thông qua index của nó. Còn phương thức thứ hai lấy nội dung header field dựa vào tên của header field. Còn nếu muốn biết tên (key) của header field, có thể dùng phương thức thứ 3 ở trên.

Sau đây là ví dụ về 3 phương thức trên, trong trường hợp server gửi về chuỗi "content-type=text/plain":

Phương thức	**Giá trị**
http.getHeaderField(0)	"text-plain"
http.getHeaderField("content-type")	"text-plain"
http.getHeaderFieldKey(0)	"content-type"

Body: Cũng giống như Client, Server gửi hầu hết những thông tin trong phần body cho Client. Client dùng input stream để đọc kết quả trả về từ Server:

```
//con là một đối tượng HttpConnection

InputStream in=con.openInputStream();
```

```
int length=(int)con.getLength();
byte data[];
if(length!=-1){
        data=new byte[length];
        in.read(data);
}
else
{
        ByteArrayOutputStream br=new ByteArrayOutputStream();
        int ch;
        while((ch=in.read())!=-1)
                br.write(ch);
        data=br.toByteArray();
        br.close();
}
```

- **Các hàm API HttpConnection**

3.3.2.1. Một vài hàm thường dùng trong HttpConnection

Phương thức	**Mô tả**
long getDate()	Lấy về ngày tháng từ Header

long getExpiration()	Lấy về thời gian hết hạn
String getFile()	Lấy tên file từ địa chỉ URL
int getHeaderField(int n)	Lấy về giá trị trường header bằng chỉ số
String getHeaderField(String name)	Lấy về giá trị trường header bằng tên
long getHeaderFieldDate(String name, long def)	Lấy về trường ngày tháng theo tên
int getHeaderFieldInt(String name, int def)	Lấy về trường int theo tên
String getHeaderFieldKey(int n)	Lấy về khóa của trường sử dụng chỉ số
String getHost()	Lấy về tên host từ địa chỉ URL
long getLastModified()	Lấy về giá trị trường last-modified
String getPort()	Lấy về cổng từ địa chỉ URL
String getProtocol()	Lấy về giao thức từ địa chỉ URL
String getQuery()	Lấy về chuỗi truy vấn (chỉ dùng với GET)
String getRef()	Lấy về phần tham chiếu của địa chỉ URL
String getRequestMethod()	Lấy về phương thức hiện thời đang sử dụng (GET, POST hay HEAD)
String getRequestProperty(String key)	Lấy về cấu hình hiện thời của request

	property
int getResponseCode()	Lấy về mã phản hồi (numeric value)
String getResponseMessage()	Lấy về thông điệp phản hồi (text value)
String getURL()	Lấy về toàn bộ địa chỉ URL
void setRequestMethod(String method)	Đặt phương thức truyền thông (GET, POST or HEAD)
void setRequestProperty(String key, String value)	Đặt thuộc tính yêu cầu (header information)

3.3.3.2.Ví dụ về kết nối Internet với J2me

Bạn biết rằng khi bạn kết nối đến các địa chỉ website như "http://google.com.vn" thì trên trình duyệt bạn sẽ nhận được kết quả trả về là các đối tượng web, như hình ảnh, nút nhấn, links... Tuy nhiên, một chương trình J2me không phải là trình duyệt, nó không hiểu được các mã HTML, đối với nó mọi thứ mà Server của website đó gửi về đều là các byte dữ liệu nhị phân, bạn có thể chuyển đổi nó về các đối tượng như chuỗi ký tự, số, hình ảnh... để sử dụng, điều này còn phụ thuộc vào việc server gửi về những dữ liệu gì, có chuyển đổi về các kiểu dữ liệu bạn mong muốn hay không?.

Để làm rõ hơn vấn đề này, chúng ta sẽ cùng nhau đi xây dựng một chương trình J2me đơn giản, trong đó chương trình sẽ cung cấp cho người dùng một TextField để người dùng nhập địa chỉ kết nối. Sau khi người dùng nhập song địa chỉ và nhấn nút "connect" trên màn hình, chương trình sẽ tiến hành kết nối và nhận kết quả trả về từ phía server, chuyển kết quả nhận được đó thành dạng chuỗi và hiển thị lên màn hình:

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
```

```
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
public class vd extends MIDlet implements Runnable,
CommandListener{
    HttpConnection con;
    InputStream in;
    OutputStream out;
    Thread th;
    Form fr;
    Display dis;
    TextField t1;
    StringItem it;
    Command c1;
    String url;
    public vd(){
        th=new Thread(this);
        dis=Display.getDisplay(this);
        fr=new Form("vdform");
        t1=new TextField("Địa chỉ URL:","",50,TextField.URL);
        it=new StringItem("kết quả kết nối:","");
        fr.append(t1);
        fr.append(it);
        c1=new Command("connect",Command.OK,1);
        fr.addCommand(c1);
        fr.setCommandListener(this);
    }
    public void startApp() {
       dis.setCurrent(fr);
```

```
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void connect(){
                try{
                 con=(HttpConnection)Connector.open(url);
                 con.setRequestMethod(HttpConnection.GET);
                 con.setRequestProperty("User-Agent","Profile/MIDP-2.0
                 Configuration/CLDC-1.1");
                 con.setRequestProperty("Content-Language", "en-US");
                 con.setRequestProperty("Content-Type", "application/x-www-
                 form-urlencoded");
                 }
                 catch(Exception e){
                System.out.println("kết nối lỗi:"+e.getMessage());
                 }
      }
      public void in(){
       String str="nothing";
                try{
                        in=con.openInputStream();
                        int length=(int)con.getLength();
                        byte server[];
                        if(length!=-1){
                                server=new byte[length];
                                in.read(server);
```

```
                    str=new String(server);
                }
                else
                {
                    ByteArrayOutputStream br=new
                   ByteArrayOutputStream();
                    int ch;
                    while((ch=in.read())!=-1)
                        br.write(ch);
                    str=new String(br.toByteArray());
                    br.close();
              }
              it.setText(str);
            }
            catch(Exception e){
            }
      }
    public synchronized void run(){
       this.connect();
       this.in();
    }
    public void commandAction(Command c, Displayable d){
        url=t1.getString();
        th=new Thread(this);
        th.start();
    }
}
```

Hình:

Ví dụ kết nối Internet với J2me

Giả sử như chúng ta kết nối đến địa chỉ "http://google.com.vn", bạn sẽ ngạc nhiên khi thấy rằng kết quả trả về không phải là biểu tượng google và các thành phần quen thuộc khác của trang web này, mà là một chuỗi các ký tự khá rườm rà, đó chính là mã HTML mà máy chủ của trang web google.com.vn đã trả về cho chương trình của bạn.

Qua ví dụ này chúng ta sẽ học được cách viết một chương trình kết nối Internet với J2me với ba thao tác cơ bản:

- Tạo kết nối đến server
- Gửi thông tin lên server

- Nhận và xử lý thông tin trả lời từ server

Mã để tạo ra một kết nối đến máy chủ Server từ xa cần chỉ rõ địa chỉ URL và một vài thông tin header quan trọng cho yêu cầu request từ máy khách, như sử dụng kiểu kết nối nào, dùng ngôn ngữ gì...:

```
con=(HttpConnection)Connector.open(url);
con.setRequestMethod(HttpConnection.GET);
con.setRequestProperty("User-Agent","Profile/MIDP-2.0
Configuration/CLDC-1.1");
con.setRequestProperty("Content-Language", "en-US");
con.setRequestProperty("Content-Type", "application/x-www-
form-urlencoded");
```

Sau đó, để nhận thông tin trả về từ máy chủ ta cần phải mở một luồng InputStream và đọc dữ liệu từ máy chủ Server trả về vào trong một mảng byte. Trong ví dụ này, chúng ta chưa có nhu cầu gửi hoặc nhận lệnh HTTP:

```
//Tạo InputStream connection
in=con.openInputStream();
int length=(int)con.getLength();
byte server[];
if(length!=-1){
        server=new byte[length];
        in.read(server);
        str=new String(server);
        }
else
{
        ByteArrayOutputStream br=new
         ByteArrayOutputStream();
        int ch;
```

```
        while((ch=in.read())!=-1)
            br.write(ch);
        str=new String(br.toByteArray());
        br.close();
    }
```

Khi ContentConnection và InputStream đã được thiết lập, chúng ta thử tham chiếu đến chiều dài length của dữ liệu được gửi về từ Server. Nếu length có thể xác định, chúng ta đọc kết quả vào trong mảng server chỉ bằng một lệnh:

```
in.read(server);
```

Nếu length không có sẵn, chúng ta cần phải đọc từng ký tự một. Java cung cấp một công cụ tiện lợi và hiệu quả giúp bạn điều khiển công việc này. Sử dụng ByteArrayOutputStream, các byte được đọc vào trong một mảng byte nội, Java sẽ điều khiển và tự thay đổi kích thước của mảng cho phù hợp. Khi Java đọc xong, chúng ta đơn giản chuyển nội dung stream vào trong mảng byte của chương trình:

```
ByteArrayOutputStream br=new ByteArrayOutputStream();
int ch;
while((ch=in.read())!=-1)
    br.write(ch);
str=new String(br.toByteArray());
br.close();
```

Cuối cùng chúng ta chuyển đổi kết quả trả về thành dạng chuỗi và in ra màn hình điện thoại nhờ đối tượng ImageItem:

```
str=new String(server);//nếu tham chiếu được length
str=new String(br.toByteArray());//nếu không tham chiếu được length
it.setText(str);//hiển thị kết quả lên màn hình điện thoại
```

Dưới đây là một ví dụ tương tự, tuy nhiên, thay vì kết nối đến một địa chỉ trang web nào đó và nhận về mã HTML, chúng ta sẽ dùng chương trình J2me kết nối đến một địa chỉ cho phép download một ảnh định dạng PNG cỡ nhỏ (nhỏ hơn 48*48 pixel, lớn

hơn có thể vượt quá khả năng hiển thị của thiết bị). Sau khi download được ảnh về ta sẽ hiển thị ảnh đó lên màn hình điện thoại, dưới đây là mã nguồn:

Hình :

Ví dụ kết nối Intenet với J2me

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
```

```
public class vd extends MIDlet implements Runnable,
CommandListener{
    HttpConnection con;
    InputStream in;
    OutputStream out;
    Thread th;
    Form fr;
    Display dis;
    TextField t1;
    StringItem it;
    Command c1;
    String url;
    Image im;
    public vd(){
        th=new Thread(this);
        dis=Display.getDisplay(this);
        fr=new Form("vdform");
        t1=new TextField("Địa chỉ URL:","",50,TextField.URL);
        t1.setString("http://www.iconspedia.com/dload.php?up_id=3574");

        it=new StringItem("kết quả kết nối:","");
        fr.append(t1);
        fr.append(it);
        c1=new Command("connect",Command.OK,1);
        fr.addCommand(c1);
        fr.setCommandListener(this);
    }
    public void startApp() {
```

```
    dis.setCurrent(fr);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void connect(){
                try{
                 con=(HttpConnection)Connector.open(url);
                 con.setRequestMethod(HttpConnection.GET);
                 con.setRequestProperty("User-Agent","Profile/MIDP-2.0
                  Configuration/CLDC-1.1");
                 con.setRequestProperty("Content-Language", "en-US");
                 con.setRequestProperty("Content-Type", "application/x-
                 www-
                 form-urlencoded");
                 }
                 catch(Exception e){
                System.out.println("kết nối lỗi:"+e.getMessage());
                 }
      }
      public void in(){
       String str="nothing";
       im=null;
      try{
                in=con.openInputStream();
                int length=(int)con.getLength();
                byte server[];
```

```
            if(length!=-1){
                    server=new byte[length];
                    in.read(server);
            }
            else
            {
                    ByteArrayOutputStream br=new
                      ByteArrayOutputStream();
                    int ch;
                    while((ch=in.read())!=-1)
                            br.write(ch);
                    server=br.toByteArray();
                    br.close();
              }
              im=Image.createImage(server,0,server.length);
              ImageItem ii=new
              ImageItem(null,im,ImageItem.LAYOUT_DEFAULT,null);
              if(fr.size()!=0)
                 fr.set(0, ii);
              else
                 fr.append(ii);
             }
            catch(Exception e){
            }
   }
public synchronized void run(){
    this.connect();
    this.in();
```

```
        }
        public void commandAction(Command c, Displayable d){
            url=t1.getString();
            th=new Thread(this);
            th.start();
        }
    }
```

Đoạn mã này không có nhiều khác biệt với đoạn mã trên, Thay vì chuyển đổi kết xuất từ server trả về thành dạng chuỗi, ta lại chuyển đổi kết quả trả về từ server thành dạng ảnh và hiển thị lên màn hình:

```
im=Image.createImage(server,0,server.length);
ImageItem ii=new
ImageItem(null,im,ImageItem.LAYOUT_DEFAULT,null);
if(fr.size()!=0)
    fr.set(0, ii);
else
    fr.append(ii);
```

Trong đoạn mã trên đối tượng "server" là đối tượng mảng byte dùng để chứa kết quả trả về từ server. Bạn chú ý, có thể links download ảnh này sẽ không còn tồn tại sau một khoảng thời gian nhất định nào đó, bạn nên thay link này bằng một link khác, chú ý, kiểu ảnh được J2me hỗ trợ là PNG.

CHƯƠNG 4: ĐIỀU KHIỂN KẾT NỐI GIỮA ĐIỆN THOẠI VÀ MẠNG VIỄN THÔNG VỚI TẬP LỆNH AT

4.1. Giới thiệu chung về tập lệnh AT (GSM 07.07)

Đầu những năm 1980, công ty Hayes đã bắt đầu sản xuất loại thiết bị Smartmodem với tốc độ 1200. Sau đó không lâu công ty này đã nhanh chóng phát hành ra thị trường loại Smartmodem với tốc độ 2400 baud/bps trong khi loại Smartmodem 1200 lại đang được thịnh hành trên thị trường. Khi đó Hayes không có thời gian để thay đổi lại thiết kế giữ hai loại Smartmodem 1200 và 2400. Chính vì thế mà các nhà thiết kế của Hayes đã tạo ra một chuẩn modem tồn tại cho đến ngày nay.

Cả hại loại Smartmodem 1200 và 2400 đều được coi là các loại modem thông minh thời đó, nó cho phép con người có thể gọi tới các số điện thoại khác từ điện thoại mà nó được gắn trên đó, khi người nghe từ modem bên kia có thể nhấn nút để chấp nhận hoặc từ chối cuộc gọi. Quy trình quay số chung giữa các modem được sử dụng như trong mạng điện thoại công cộng POTS/PSTN (Plain Old Telephone Service/Public Switched Telephone Network) được giới thiệu vào những năm 1980.

Các Smartmodem 1200 và 2400 đã không nhận biết được tín hiệu bận rộn hoặc phát hiện ra âm thanh quay số cũng như hàng chục các tính năng khác hiện nay dùng trong modem sản xuất cuối những năm 1980.

Những gì đã làm cho Smartmodem 1200 và 2400 là duy nhất tại thời điểm đó là một nhà sản xuất cung cấp hai mô hình tốc độ khác nhau của modem POTS/PSTN mà chúng chấp nhận các lệnh lập trình, được gọi là tập lệnh AT (ATtention). Tuy nhiên không phải bất kỳ phần mềm nào dùng trên các Smartmodem 1200 cũng sẽ làm việc trên Smartmodem 2400.

Một số nhà sản xuất modem đột nhiên nhận ra lợi thế của mô hình modem mới hỗ trợ các lệnh tương tự như các mô hình trước đây. Một số nhà sản xuất thường xuyên phát triển các lệnh hoàn toàn khác nhau phù hợp với thiết kế modem mà họ sản xuất ra.

Hãng Hayes kịp thời đã kiện các công ty này bằng cách sử dụng từ "Hayes" trong các tài liệu sản phẩm và bao bì của họ. Từ đó các nhà sản xuất khác bắt đầu gọi modem của

họ là sử dụng tập lệnh AT tương thích và tiếp tục để bắt chước các thiết lập lệnh của Hayes. Tới năm 1986, hầu hết các modem đều hỗ trợ tập lệnh AT.

Ngày nay hầu hết các modem chuyên dụng và các điện thoại di động hỗ trợ công nghệ GSM đều sử dụng tập lệnh AT cơ bản hoặc tập lệnh AT mở rộng để truyền thông trong mạng.

4.1. Các lệnh AT kiểm tra cấu hình

Các lệnh AT để kiểm tra cấu hình bao gồm các lệnh để kiểm tra cấu hình hệ thống, cấu hình cliens. Cung cấp các thông tin nhận thực từ cliens đến server..

4.1.1. AT+CGMI nhận dạng nhà sản xuất

Ý nghĩa

Cho phép kiểm tra nhà sản xuất modem GSM đang được sử dụng.

Cú pháp

AT+ CGMI

Giải thích

- AT là một từ khóa bắt đầu của bất cứ câu lệnh nào thuộc tập lệnh AT (AT command).
- CGMI là yêu cần xác thực nhà sản xuất mà cliens gửi đến server.
- Sau khi nhận được yêu cầu, server sẽ kiểm tra nếu đúng thì nó sẽ trả về nhà sản xuất của mình hay là chuẩn làm việc của nó theo nhà sản xuất nào.

Thường thì đối với chuẩn GSM 07.07 kết quả trả về đối với câu lênh CGMI sẽ là một chuỗi là tên nhà sản xuất như :" Ubinetics Ltd".

2. AT+ CGMM kiểm tra model hệ thống

Ý nghĩa

Cho phép người lập trình kiểm tra model làm việc của modem GSM.

Cú pháp

AT+CGMM

Giải thích

Khi modem nhận được một cú pháp lệnh đúng như thế này nó sẽ gửi trả về một chuỗi ký tự là tên của các model.

Có một số model thông dụng như:GDC200,GC201,GA100.

3. AT+CGMR xác thực nhà sản xuất.

Ý nghĩa

Xác thực thông tin chính xác về nhà sản xuất. Kết quả nhận được la chi tiết hơn so với AT+CGMI.

Cú pháp

AT+ CGMR

Giải thích

Modem sẽ trả về một cách chính xác nhất nhà sản xuất của mình. Việc xác nhận lại đôi khi vẫn thường hay sử dụng khi mà kết quả trả về từ "AT CGMI" chưa được rõ dàng và chính xác.

Kết quả trả về với câu lệnh này sẽ là một "revision" chính xác về nhà sản xuất của modem ở dạng chuỗi ký tự.

4. AT+CGSN kiểm tra serial modem

Ý nghĩa

Cho phép lấy được số serial của từng modem giúp ích cho việc nhận thực modem.

Cú pháp

AT+CGSN

Giải thích

Sau khi lệnh thực hiện thành công modem sẽ trả về chính xác số serial của mình dưới dạng chuỗi ký tự. Số serial của modem trả về thường có dạng :

02-GDC200-xxxxxxx

02-GC201-xxxxxxx

02-GA100-xxxxxxx

Trong đó xxx sẽ là serial của modem, phi trước sẽ là model mà nó sử dụng.

5. AT+CSCS thiết lập ký tự thiết bị đầu cuối.

Ý nghĩa

Lựa chọn bộ ký tự được hỗ trợ bởi modem, chuyển dổi ký tự giữa modem và thiết bị đầu cuối.

Có một số kiểu, chuẩn ký tự được sử dụng như : GSM alphabet(GSM), international reference alphabet (IRA), hay kiểu HEX.

Cú pháp

– Kiểm tra danh sách bộ ký tự được modem hỗ trợ

AT+CSCS=?

Nếu modem nhận được câu lênh đúng thì nó sẽ trả về danh sách các kiểu ký tự mà nó hỗ trợ theo dạng " CSCS: <danh sách kiểu được hỗ trợ > "

– Kiểm tra bộ ký tự đang được sử dụng

AT+CSCS?

Kết quả của câu lệnh này là modem sẽ trả về kiểu ký tự mà modem đang sử dụng có dạng " CSCS: <kiểu đang được sử dụng> "

– Thiết lập bộ ký tự sử dụng cho modem

AT CSCS=<kiểu thiết lập>

<kiểu thiết lập> có thể là "GSM","IRA","HEX".

6. AT+CIMI Nhận dạng thuê bao di động

Cú pháp

AT+CIMI

Giải thích

Kết quả trả về của lệnh này sẽ là số nhận dạng thuê bao di động quốc tế của modem mà chúng ta sử dụng(imsi). Kết quả này chỉ nhận được khi nó tồn tại trong một mạng cụ thể, hay cũng có thể hiểu là thuê bao đang ở tình trạng online.

7. AT+WS46 Lựa chọn mạng

Ý nghĩa

Cho phép lựa chọn một mạng di động để modem hoạt động.

Cú pháp

– Kiểm tra mạng di động tồn tại

AT+WS46=?

– Lựa chọn mạng cho modem hoạt động

AT+WS46=<n>

Giải thích

Trên thực tế mạng di động của chúng ta bao gồm các nhà cung cấp khác nhau, mỗi thuê bao sử dụng một nhà cung cấp mạng của mình. Vì vậy mà trước khi kết nối thuê bao di động luôn cần phải tìm được sự hiện diện của các nhà cung cấp dịch vụ qua đó lựa chọn nhà cung cấp dịch vụ cho mình. Và thuê bao cần phải lựa chọng kiểu mạng tế bào cho mình.

Trước hết để tìm kiếm, kiểm tra xem tại khu vực của thuê bao có những kiểu mạng tế bào nào ta sử dụng lệnh :

" AT+WS46=? "

Câu lệnh này sẽ trả về danh sách các mạng tế bào dưới dạng số tự nhiên. Đối với mạng GSM thì kết quả sẽ là số 12.

Tương tự để lựa chọn mạng ta sử dụng lệnh:

"AT+WS46=<n>"

Với "n" là số hiệu của mạng. GSM=12.

4.2 Các lệnh AT điều khiển cuộc gọi

4.2.1. ATD quay số thiết lập cuộc gọi

Ý nghĩa

Cho phép quay số thiết lập cuộc gọi đến một thuê bao khác với tín hiệu điều khiển tùy chọn.

Cú pháp

ATD<n>[<msg>][;]

Giải thích

Với <n> là số thuê bao đích cần gọi tới,

<msg> là các ký tự điều khiển như :

- “,” dừng quay số.
- “T” : bỏ qua chuông quay số
- “P” : bỏ qua xung quay số.
- “!” : đăng ký gọi lại.
- “W” : chờ chuông quay số.
- “@” : chờ trả lời.
- “I

” : hạn chế .

Các cuộc gọi nếu ký tự được gửi sau <cr> nhưng trước khi nhận được bản tin kết nối “OK” thì cuộc gọi sẽ bị hủy.

Khi thực hiện lệnh “ ATD<n>[<msg>][;] “ ta có thể nhận được một số mã trả lời dưới dạng số từ 0 đến 8. Trong đó

- 0 (OK) : lệnh thực hiện thành công, không có lỗi.
- 1 (CONNECT) : kết nối được thiết lập (ATX=0).
- 2 (RING) : chuông báo
- 3 (NO CARRIER) : cuộc gọi thất bại hoặc mất kết nối.
- 4 (ERROR) : lệnh không hợp lệ hoặc quá dài.
- 7 (BUSY) : thuê bao gọi tới đang bận không thể tiếp nhận cuộc gọi.
- 8 (NO ANSWER) : cuộc gọi thất bại do thời gian chờ kết nối quá lâu

4.2.2. ATD> quay số từ danh bạ điện thoại

Ý nghĩa

Cho phép quay số thiết lập cuộc gọi với đích đến là một số đích được lưu trong bộ nhớ danh bạ điện thoại.

Cú pháp

ATD><mem><n>[l][;]

Giải thích

Câu lệnh : “ ATD><mem><n>[l][;] “ sẽ thực hiện việc quay số thiết lập một cuộc gọi đến một thuê bao mà số địa chỉ được lấy trong danh bạ điện thoại đã được ghi rõ . Nếu

ký tự gửi sau <cr> trước khi nhận được thông báo kết nối hoặc bản tin "OK" thì cuộc gọi sẽ bị hủy.

Với câu lệnh trên tùy chọn mem có thể là các số khẩn cấp hay là các số được lưu trong dạnh bạ trên SIM. Nó có thể nhận các giá trị là "EN"(Enmergency number) hoặc "AD" (SIM phone book).

Tùy chọn <n> chính là vị trí bộ nhớ của số điện thoại dùng để quay số

4.2.3. Quay số hiện thời trên danh bạ điện thoại

Để quay số thiết lập cuộc gọi đến một số điện thoại hiện hành trên danh bạ điện thoại ta sử dụng cú pháp lệnh:

" ATD><n>[l][;] "

Tương tự như việc quay số được lựa chọng từ điện thoại ta chỉ cần chỉ ra địa chỉ hay vị trí của số điện thoại cần gọi đến trong danh bạ điện thoại tương ứng với tùy trọn <n> trong câu lệnh trên để tiến hành quay số đến một thuê bao đã được lưu trữ trong danh bạ điện thoại.

4.2.4. ATD+CHUP Ngắt cuộc gọi

Ý nghĩa

Cho phép kết thúc cuộc gọi đang được kích hoạt.

Cú pháp

ATD+CHUP

Giải thích

Khi thực hiện lệnh thành công, tất cả các cuộc gọi đang hoạt động sẽ bị treo đồng thời tiết hành kết thúc cuộc gọi một cách luân phiên.

4.2.5. AT+CBST Lựa chọn kiểu dịch vụ

Ý nghĩa

Cho phép thiết lập các thông số dùng cho cuộc gọi tới, các giá trị này cũng được dùng để chấm dứt một cuộc gọi dữ liệu.

Cú pháp

- Kiểm tra danh sách được hỗ trợ

AT+CBST=?

- Kiểm tra trạng thái hiện tại

  AT+CBST

- Thiết lập các tham số dùng cho dịch vụ

  AT+CBST=:[<speed>[,<name>[,<ce>]]]>

Giải thích

"AT+CBST=?"

Khi thực hiện lệnh thành công sẽ trả về danh sách các cấu hình được hỗ trợ dưới dạng xâu như sau : " CBST:<speed><name><ce> "

Trong đó speed có thể là 7 (9600bps_V32), 12(9600bps_V34), 14(14400bps_V34), 71(9600bps_V110), 75(14400bps_V110).

Name = 0, kết nối dữ liệu không đồng bộ.

<ce> có thể nhận giá trị 0 hoặc 1 để thiết lập cho quá trình sửa lỗi diễn ra transparent hay non-transparent.

Đối với kết nối gửi dữ liệu FAX chỉ được hỗ trợ ở tốc độ 9600 bps.

4.2.6. AT+CRLP Giao thức liên kết vô tuyến

Ý nghĩa

Cho phép kiểm tra, thiết lập các tham số dùng cho cuộc gọi dữ liệu không trong suốt.

Cú pháp

- Kiểm tra danh sách các tham số được hỗ trợ

  AT+CRLP=?

- Kiểm tra trạng thái hiện tại hay cấu hình các tham số

  AT+CRLP?

  AT+CRLP=<parameter>

4.2.7. AT+CR Điều khiển phản hồi dịch vụ

Ý nghĩa

Cho phép tùy chọn nhận hoặc không nhận phản hồi dịch vụ. Nếu ở trạng thái kích hoạt modem sẽ truyền các phản hồi dịch vụ về thiết bị đầu cuối dữ liệu khi đã thiết lập

tốc độ truyền, cấu hình truyền trước khi các mã kết nối cuối cùng được trả về.

Cú pháp

AT+CR=?

Giải thích

Giống như các lệnh điều khiển cuộc gọi khác, việc điều khiển phản hồi dịch vụ cũng cho phép ta kiểm tra danh sách hỗ trợ, kiểm tra trạng thái hiện tại, thiết lập chế độ.

"AT+CR=?" nếu thực hiện thành công lệnh này sẽ trả về các chế độ được hỗ trợ dưới dạng xâu theo dạng "CR: <danh sách mode>"

Thường thì có 2 mode chính là : 0 disnable reporting

1 enable reporting

Để kiểm tra mode hiện tại ta chỉ việc sử dụng cú pháp lệnh : "AT+CR?"

Tương tự để thiết lập mode ta chỉ việc gửi đến modem lệnh : "AT+CR=<mode>"

4.2.8. AT+CEER Báo cáo lỗi mở rộng

Ý nghĩa

Lệnh này trả về thông tin dưới dạng text, nó là các thông tin mở rộng chi tiết hơn về các lỗi gây ra việc thất bại trong một cuộc gọi hay là các lỗi làm thay đổi cuộc gọi.

Cú pháp

AT+CEER

Giải thích

Khi lệnh được thực hiện thành công bản tin trả về sẽ có dạng: " CEER:<report>".

Giả sử khi quay số gọi tới một thuê bao khác mà thuê bao đó không thể tiếp nhận cuộc gọi vì đang có một cuộc gọi khác thì bản tin trả về của lệnh này sẽ là:

" CEER: user busy " , "user busy" chính là <report>.

4.2.9. AT+CRC Mã kết quả

Ý nghĩa

Lệnh dùng để điều khiển cho người sử dụng báo cáo các định dạng mở rộng thông qua việc thiết lập cuộc gọi của một thiết bị di động.

Cú pháp

AT+CRC=<mode>

<mode> ở đây có thể là 0 hoặc 1, tương ứng với việc cho phép hoặc không.

4.3 Các lệnh liên quan đến dịch vụ mạng

4.3.1. AT+CNUM,subscriber number

Ý nghĩa

Khi thực hiện lệnh này sẽ trả về MSIDNS liên quan đến thuê bao. Nếu thuê bao có các MSIDNS khác nhau cho các dịch vụ khác nhau thì mỗi MSIDNS sẽ được trẻ về trên một dòng.

Cú pháp

AT+CNUM

Giải thích

Nếu thực hiện thành công sẽ nhận được kết quả trả về có dạng:

”CNUM: [<alpha>],<number>,<type>,<speed>,<service>[,<itc>]<cr><lf>

Trong đó alpha: là tùy trọn về kiểu chuỗi số lien quan đến số thuê bao

Number: chuỗi số thuê bao điện thoại được dịnh dạng theo kiểu của tùy trọng <type>.

Type: kiểu địa chỉ ở dạng số nguyên.

Speed: thông tin về tôc độ

Service: thông tin về dịch vụ ,chế độ hoạt động đồng bộ hay không đồng bộ….

4.3.2. AT+CREG Đăng ký mạng

Ý nghĩa

Hiển thị trạng thái đăng ký mạng.

Cú pháp

– Kiểm tra các chế độ được hỗ trợ

AT+CREG=?

- Kiểm tra chế độ hiện tại

  AT+CREG?

- Thiết lập chế độ hiện tại

  AT+CREG=<n>

3.3.3. AT+COPS Lựa chọn nhà cung cấp

Ý nghĩa

Cho phép đăng ký, hiển thị các nhà cung cấp mạng hợp lệ.

Cú pháp

- Kiểm tra danh sách được hỗ trợ

  AT+COPS=?

- Kiểm tra trạng thái hiện tại

  AT+COPS?

- Thiết lập, đăng ký nhà cung cấp mạng.

  AT+COPS=[<mode>[,<format>[,<oper>]]]

Giải thích

Tham số "*mode*" là kiểu đăng ký mạng mà modem sẽ sử dụng. Nó có thể là tự động hoặc thủ công tùy thuộc vào thiết lập của người quản trị.

Tham số "*format*" là định dạng ký tự của "*oper*"

"*oper*" thể hiện nhận dạng của nhà cung cấp với định dạng được thiết lập bởi "*format*"

4.3.4. AT+CLCK Khóa hoặc mở khóa

Ý nghĩa

Cho phép khóa hoặc mở mạng cho modem, mật khẩu là cần thiết cho hành động này.

Cú pháp

AT+CLCK=<fac>,<mode>[,<password>]

Giải thích

<fac> cho phép lựa chọn hành động điều khiển.

<mode> chế độ khóa/mở khóa đối với <fac>

<password> mật khẩu cần thiết dùng cho thao tác thay đổi.

4.4. Các lệnh AT điều khiển tin nhắn SMS

4.4.1 AT+CMGF Định dạng SMS

Ý nghĩa

Lệnh này cho phép thiết lập định dạng SMS mà modem sử dụng, để thiết lập một định dạng SMS cho modem thì định dạng đó cần được hỗ trợ bởi modem.

Cú pháp

- Kiểm tra các định dạng SMS được hỗ trợ

  AT+CMGF=?
- Kiểm tra định dạng SMS mà modem đang sử dụng

  AT+CMGF?
- Thiết lập định dạng SMS cho modem

  AT+CMGF=<mode>

Giải thích

<mode> ở đây chính là chế độ định dạng SMS mà modem sẽ sử dụng. Có 2 <mode> cơ bản là 0 (PDU) và 1(Text).

Trước khi có một thao tác điều khiển liên quan đến SMS ta cần cấu hình chế độ SMS cho modem trước khi điều khiển.

4.4.2. AT+CMGW Ghi SMS vào bộ nhớ sim card.

Ý nghĩa

Cho phép ghi một tin nhắn văn bản vào bộ nhớ lưu trữ của sim card. Nếu thành công sẽ trả về bị trí lưu tin nhắn.

Cú pháp

AT+CMGW=<number>

Giải thích

Với <number> là số thuê bao cần gửi đến, khi thực hiện lệnh thành công modem

sẽ trả về ký tự ">". Sau đó ta có thể nhập nôi dung tin nhắn cần lưu trữ và kết thúc bằng "Ctrl+z".

4.4.3. AT+CMSS Gửi tin nhắn từ bộ nhớ Sim card

Ý nghĩa

Cho phép điều khiển modem GSM gửi một tin nhắn được lưu trong bộ nhớ Sim card.

Cú pháp

AT+CMSS=<index>,<number><cr>

Giải thích

Với AT+CMSS, <index> chính là chỉ số của tin nhắn được lưu trong bộ nhớ Sim card, <number> là số thuê bao cần gửi tin nhắn,<cr> mã kết thúc câu lệnh.

Ví dụ để gửi tin nhắn đến số "0123456789" đã được lưu trong bộ nhớ tại vị trí số 2 ta thực hiện cú pháp lệnh:

AT+CMSS=2,"0123456789"<cr>

4.4.4. AT+CMGD Xóa tin nhắn trong bộ nhớ lưu trữ

Ý nghĩa

Xóa một tin nhắn đang lưu trữ trong bộ nhớ điện thoại. Vị trí bộ nhớ mà tin nhắn lưu trữ là cần thiết cho câu lệnh.

Cú pháp

AT+CMGD=<index>

Trong đó <index> chính là chỉ số, vị trí lưu trữ của tin nhắn.

Ví dụ để xóa tin nhắn tại vị trí lưu trữ số 3 ta sử dụng cú pháp lệnh:

AT+CMGD=3

4.4.5. AT+CMGL Liệt kê tin nhắn

Ý nghĩa

Cho phép liệt kê toàn bộ các tin nhắn SMS chưa đọc trên bộ nhớ.

Cú pháp

AT+CMGL

Lưu ý

Ta có thể sử dụng tùy chọn "ALL" để liệt kê toàn bộ tin nhắn có trong bộ nhớ hiện tại. ví dụ: *AT+CMGL="ALL"*

4.5. Giới thiệu module GSM MC35i

4.5.1 Giới thiệu

Module MC35i là một modem GSM hoạt động trên nền mạng 900MHz và 1800MHz do Sieamen sản xuất. Nó hỗ trợ GPRS và các tinh năng cơ bản của một điện thoại di động như nghe, gọi và nhắn tin.

GSM Modem loại Siemens MC35i có kích thước nhỏ - gọn, lắp đặt đơn giản nhanh chóng, tích hợp nhiều tính năng và khả năng hoạt động lâu dài trong điều kiện bình thường là những điểm nổi bật của Modem MC35i. Điều này mang lại những tiện ích đối với các thiết bị điện thoại di động, máy tính xách tay, thiết bị đa phương tiện... và đặc biệt là khả năng tích hợp dễ dàng với PDA, các thiết bị di động thu nhỏ...

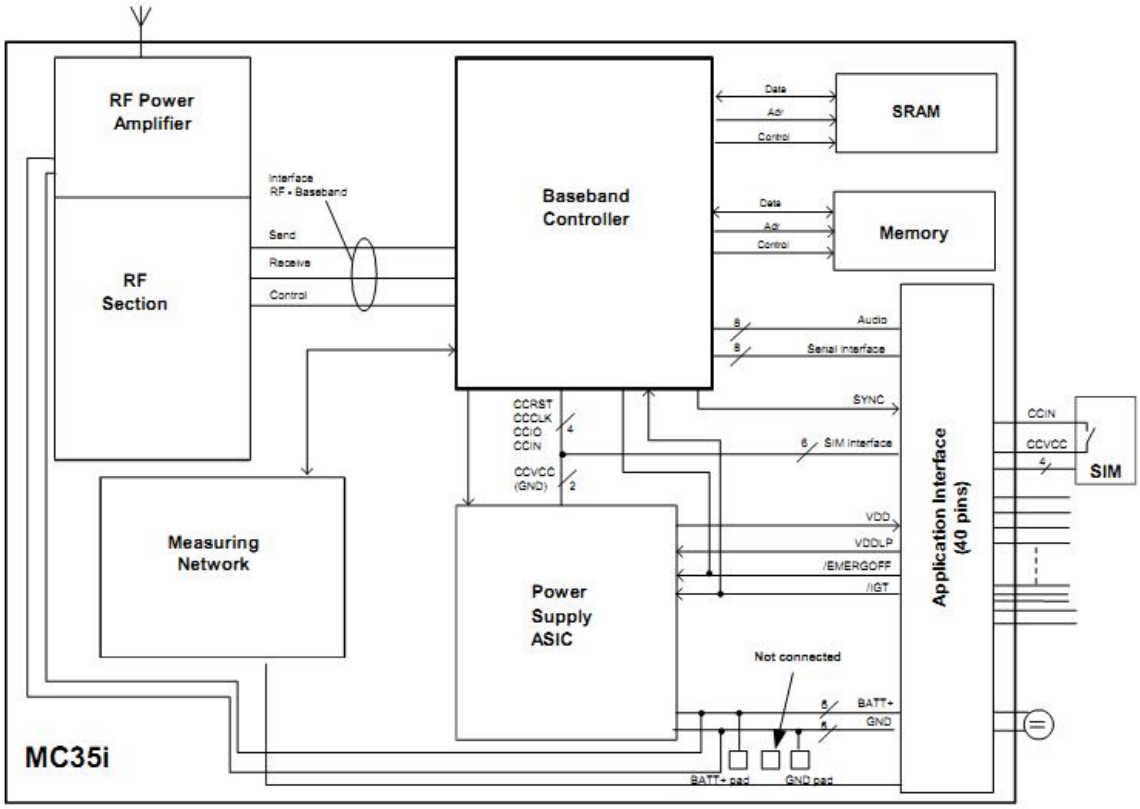
Module MC35i đáp ứng được giải pháp GSM/GPRS cho hiệu suất cao với: Vi xử lý băng tần cơ sở, điện áp cung cấp ASIC, tần số vô tuyến điện bao gồm một bộ khuếch đại công suất và giao diện anten. Các phần mềm MC35i được lưu trữ trong một thiết bị nhớ flash. Bộ nhớ bổ sung SRAM cho phép MC35i đáp ứng yêu cầu kết nối GPRS.

Các giao diện vật lý cho các ứng dụng di động được thực hiện thông qua một kết nối ZIF. ZIF gồm có 40 chân, cho phép kiểm soát các khối, truyền dữ liệu và tín hiệu âm thanh, các đường điện áp cung cấp. Ngoài ra, module GSM MC35i cung cấp giao diện nối tiếp tích hợp với giao diện Man-Machine (MMI), điều khiển bởi tập lệnh AT và hỗ trợ tốc độ truyền lên tới 230 kbps.

Module gsm mc35i

4.5.2. Cấu trúc, chức năng Module GSM MC35i

4.5.2.1 Sơ đồ khối MC35i



Sơ đồ khối module gsm mc35i

Dựa vào sơ đồ khối ta có thể thấy module gsm mc35i đáp ứng được các tính năng có bản của một điện thoại di động như nghe, gọi, nhắn tin.

Khối RF Power Amplifier và RF secsion giúp module mc35i khuếch đại tín hiệu vô tuyến thu được, lựa chon giải tần hoạt động với Baseband Controller. Bộ nhớ SRAM hay bộ nhớ ngoài có thể dùng để lưu trữ dữ liệu, chương trình điều khiển, thông tin mạng, thông tin thuê bao....

4.5.2.2. Thành phần, chức năng Module GSM MC35i

Dựa vào sơ đồ khối ta có thể thấy module GSM MC35i gồm các khối chính sau:

– Khối RF

- Khối băng tần cơ sở
- Nguồn
- Bộ nhớ
- Khối giao tiếp(ZIF,SIM...)

** Khối RF của module GSM MC35i:*

Khối này có chức năng thu phát tín hiệu RF trên dải tần của GSM 900MHz/1800MHz.

- Khối khuếch đại công suất RF:

IC công suất phát được điều khiển thay đổi công suất phát thông qua lệnh APC (Auto Power Control) ra từ IC cao trung tần. Mạch APC có khả năng tự động điều chỉnh công suất phát.

- Khối kết nối anten:

Khối này có chức năng bức xạ hoặc thu nhận sóng điện từ. Phục vụ hoạt động của module GSM MC35i.

** Khối băng tần cơ sở GSM:*

- Bộ điều khiển hoạt động GSM ở tần số 26MHz:

Bao gồm CPU (Center Processor Unit – Đơn vị xử lý trung tâm). CPU thực hiện các chức năng như: Điều khiển tắt mở nguồn chính; chuyển nguồn giữa chế độ thu và phát; Điều khiển đồng bộ sự hoạt động giữa các IC; Điều khiển khối thu phát sóng; Quản lý các chương trình trong bộ nhớ; Điều khiển truy cập SIM Card...

- Nguồn cung cấp:

Khối nguồn có chức năng chính là: Điều khiển tắt mở nguồn; Chia nguồn thành nhiều mức nguồn khác nhau để phục vụ cho các khối của module; Ổn định nguồn cung cấp cho các tải tiêu thụ...

- Bộ nhớ:

CPU hoạt động theo các mã lệnh được lập trình sẵn nạp vào trong bộ nhớ. CPU sẽ không hoạt động được nếu không có phần mềm nạp trong bộ nhớ. Vi xử lý khi hoạt động sẽ truy cập và lấy ra các phần mềm điều khiển trong IC nhớ FLASH, thực hiện

giải mã tạo ra các lệnh điều khiển để điều khiển các bộ phận khác của module hoạt động.

- SRAM: (Syncho Radom Acccess Memory):

Là bộ nhớ trung gian lưu trữ tạm các dữ liệu trong quá trình xử lý của CPU. Nếu bộ nhớ SRAM hỏng thì CPU sẽ không hoạt động được, khi ta tắt nguồn thì dữ liệu trong SRAM sẽ mất.

- Giao tiếp ứng dụng (ZIF connector)

MC35i được trang bị một kết nối ZIF gồm 40 chân cho việc kết nối với những ứng dụng di động. Các chân chức năng của ZIF gồm:

- Nguồn cung cấp
- Giao tiếp RS232
- Hai giao diện âm thanh
- Giao tiếp SIM

Chân	***In / Out***	***Ký hiệu***	***Mô tả chức năng***
1-5	I	BATT+	Nguồn cung cấp
6-10	Ground	GND	Nối đất
11-12	I	POWER	Sạc nguồn
13	O	VDD	Nguồn cung cấp ngoài
14	I	BATT_TEMP	Nhiệt độ pin
15	I	/IGT	Cháy
16	O	RING0_TXD1	Kết nối RS232
17	O	/RING0_TXD1	Kết nối RS232
18	O	/RxD0	Kết nối RS232
19	I	/TxD0	Kết nối RS232
20	O	/CTS0	Kết nối RS232

21	I	/RTS0	Kết nối RS232
22	I	/DTR0	Kết nối RS232
23	O	/DCD0	Kết nối RS232
24	I	CCIN	SIM
25	O	CCRST	SIM
26	IO	CCIO	SIM
27	O	CCCLK	SIM
28	O	CCVCC	SIM
29	Ground	CCGND	SIM
30	I/O	VDDLP	Sao lưu RTC
31	I	/EMERGOFF	Ngắt nguồn
32	O	SYNC	Đồng bộ
33	O	EPP2	Audio
34	O	EPN2	Audio
35	O	EPP1	Audio
36	O	EPN1	Audio
37	I	MICP1	Audio
38	I	MICN1	Audio
39	I	MICP2	Audio
40	I	MICN2	Audio

Bảng Chức năng chân của ZIF

Đối với giao tiếp SIM: Bộ vi xử lý băng tần cơ sở có một giao diện SIM tích hợp tương thích với tiêu chuẩn ISO 7816. Đây là đường nối vào các giao diện chủ (ZIF nối) để được kết nối với một thẻ SIM bên ngoài. Sáu chốt kết nối ZIF được dành riêng cho giao diện SIM. Các pin CCIN có nhiệm vụ phát hiện xem có sim trong khay hay không.

Tín hiệu	Mô tả chức năng
CCGND	Nối đất
CCCLK	Thẻ chip đồng hồ, với tốc độ xung nhịp khác nhau có thể được đặt trong bộ vi xử lý băng tần cơ sở.
CCVCC	Cung cấp điện áp từ PSU-ASIC
CCIO	Dòng dữ liệu đầu vào và đầu ra.
CCRST	Thẻ chip được thiết lập lại, được cung cấp bởi bộ xử lý băng tần cơ sở.
CCIN	- Nhập vào bộ xử lý băng gốc để phát hiện một khay thẻ SIM vào ngăn chứa. - Các pin CCIN là bắt buộc cho các ứng dụng cho phép người – sử dụng để loại bỏ các thẻ SIM trong quá trình hoạt động. Các pin CCIN là chỉ sử dụng cùng với một thẻ SIM. Nó không phải được sử dụng cho bất kỳ mục đích khác.

Bảng Chức năng chân của giao tiếp SIM

4.5.2.3 Hoạt động của MC35i

MC35i gồm 3 chế độ hoạt động:

- Chế độ bình thường
- Chế độ tắt nguồn
- Chế độ báo động

Chế độ hoạt động bình thường

- GSM / GPRS SLEEP

Thiết lập các chế độ khác nhau để tiết kiệm năng lượng với tập lệnh AT + CFUN. Chế độ này đang hoạt động ở mức tiết kiệm năng lượng đến mức tối thiểu. Nếu module này đã được đăng ký với mạng GSM trong chế độ IDLE, nó vẫn còn trong chế độ SLEEP, đăng ký và phân trang từ các trạm BTS.

Tiết kiệm điện có thể được lựa chọn ở các cấp độ khác nhau như vô hiệu hóa giao diện AT hoặc kích hoạt lại giao diện AT cho phép người sử dụng có thể truy cập thường xuyên các lệnh AT.

- GSM IDLE

  Khi kích hoạt chế độ này, sau khi đăng ký với mạng GSM, module có thể được phân trang từ các trạm BTS và sẵn sàng để gửi và nhận tín hiệu.

- GSM TALK

  Kết nối giữa hai thuê bao được tiến hành. Công suất tiêu thụ phụ thuộc vào các thiết lập mạng như DTX on/off, FR/EFR/HR, anten...

- GPRS IDLE

  Module đã sẵn sàng để chuyển dữ liệu GPRS, nhưng không có dữ liệu hiện đang được gửi hoặc nhận được. Công suất tiêu thụ phụ thuộc vào các thiết lập mạng và cấu hình GPRS.

- Quá trình truyền dữ liệu GPRS được tiến hành. Công suất tiêu thụ phụ thuộc vào các thiết lập mạng (ví dụ như kiểm soát mức năng lượng), đường xuống đường lên / dữ liệu giá và cấu hình GPRS.

Chế độ tắt nguồn

Bình thường thiết kế sẽ tắt sau khi gửi lệnh AT^SMSO hay chế độ khẩn cấp ra thông qua pin /EMERGOFF. Các nguồn ASIC (PSU_ASIC) ngắt kết nối cung cấp điện áp từ phần baseband của mạch điện. Chỉ có một điều chỉnh điện áp trong PSU_ASIC đang hoạt động để tạo năng lượng cho RTC

Chế độ báo động

Hạn chế hoạt động theo chức năng đưa ra cảnh báo RTC trong khi module này là ở chế độ Power Down. Trong chế độ báo thức, module không được đăng ký từ mạng GSM, do đó sẽ giới hạn số lệnh AT có thể truy cập.

4.6 Lập trình giao tiếp MC35i với vi điều khiển

Như đã trình bày, MC35i cung cấp cho chúng ta các giao diện ghép nối, điều khiển cơ bản. Việc giao tiếp mc35i với các microprocess có thể thực hiện thông qua phương

pháp truyền thông nối tiếp.

Trên thực tế có nhiều loại modem gsm hỗ trợ các chuẩn truyền thông khác nhau, dựa vào đó ta có thể lựa chọn một phương pháp giao tiếp phù hợp để cấu hình cho modem gsm.

Khi sử dụng vi điều khiển giao tiếp với mc35i để điều khiển cấu hình cho modem mc35i thực chất là việc truyền và nhận các tin hiệu điều khiển, tín hiệu phản hồi tới và từ mc35i trả về.

Vậy khi giao tiếp MC35i với vi điều khiển ta cần chú ý các đặc điểm sau:

- Chuẩn truyền thông sử dụng cho việc giao tiếp : RS232.
- Điều khiển dựa trên tập lệnh AT theo chuẩn GSM 07.07 hoặc GSM 07.05.

Một số quy tắc khi sử dụng tập lệnh AT trong lập trình giao tiếp modem GSM:

Quy tắc 1:

Tất cả các dòng lệnh phải bắt đầu với "AT" và kết thúc bởi một ký tự trả về nào đó kí hiệu là <CR>, thông thường là ký tự Enter. Trong chương trình HyperTerminal của Microsoft Windows khi kiểm tra một thiết bị được kết nối với máy tính sử dụng tập lệnh AT thì để kết thúc một lệnh AT có thể nhấn phím Enter trên bàn phím.

Thí dụ: Để xem danh sách tất cả các tin nhắn SMS trong chưa đọc được lưu trữ trong bộ nhớ của thiết bị, gõ "AT+CMGL" và cuối cùng nhấn kí tự kết thúc thì dạng cú pháp như sau:

AT + CMGL <CR>

Quy tắc 2:

Một dòng lệnh có thể chứa nhiều hơn một lệnh AT, nhưng chỉ có lệnh đầu tiên được bắt đầu bằng "AT". Mỗi lệnh được cách nhau bởi dấu chấm phẩy.

Thí dụ: Để liệt kê tất cả các tin nhắn SMS chưa được đọc trong bộ nhớ và lấy tên nhà sản xuất thiết bị thì gõ lệnh như sau:

AT+CMGL; + CGMI <CR>

Lỗi sẽ xảy ra nếu cả hai lệnh AT được bắt đầu với "AT" như sau:

AT+CMGL; AT+ CGMI <CR>

Quy tắc 3: Một chuỗi được đặt giữa cặp dấu nháy kép.

Thí dụ: đọc tất cả các tin nhắn SMS từ bộ nhớ lưu trữ trong chế độ văn bản cần phải chỉ định chuỗi "ALL" trong lệnh AT+CMGL, như sau:

AT +CMGL = "ALL" <CR>

Quy tắc 4:

Thông tin trả lời và mã kết quả luôn luôn bắt đầu và kết thúc với một khí tự trả về và ký tự linefeed (kí hiệu là LF).

Thí dụ: Sau khi gửi dòng lệnh "AT+ CGMI <CR>" đến các thiết bị di động, các thiết bị di động sẽ trả lại thông tin phản hồi dạng như sau:

< C R > < L F > N o k i a < C R > < L F > <CR><LF>OK<CR><LF>