|  |
| --- |
| **VIETNAMESE VOICE ASSISTANT** |

MỤC LỤC

[CHƯƠNG 1. GIỚI THIỆU 8](#_Toc362676620)

[1.1 Đặt vấn đề. 8](#_Toc362676621)

[1.2 Khảo sát các thống có chức năng tương tự. 9](#_Toc362676622)

[1.3 Dự án Viva 11](#_Toc362676623)

[1.3.1 Giới thiệu 11](#_Toc362676624)

[1.3.2 Mục tiêu 11](#_Toc362676625)

[1.3.3 Chức năng 11](#_Toc362676626)

[1.4 Kế hoạch xây dựng Viva 14](#_Toc362676627)

[1.5. Nội dung trình bày trong báo cáo đồ án 15](#_Toc362676628)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT & KIẾN TRÚC HỆ THỐNG VIVA. 16](#_Toc362676629)

[2.1 Sơ đồ khối hệ thống 16](#_Toc362676630)

[2.1.1 Sơ đồ tổng quan 16](#_Toc362676631)

[2.1.2 Mô tả các khối 17](#_Toc362676632)

[2.2 Khối tổng hợp tiếng Việt 19](#_Toc362676633)

[2.2.1. Cơ sở lý thuyết 19](#_Toc362676634)

[2.2.1.1.Lý thuyết tổng hợp tiếng 19](#_Toc362676635)

[2.2.1.2.Tổng hợp tiếng theo phương pháp ghép nối 20](#_Toc362676636)

[2.2.2 Chương trình Hoa Súng – Mica 23](#_Toc362676637)

[2.2.2.1 Giới thiệu chương trình 23](#_Toc362676638)

[2.2.2.2 Biểu đồ lớp của chương trình 24](#_Toc362676639)

[2.2.2.3 Quá trình tổng hợp tiếng nói của chương trình 25](#_Toc362676640)

[2.2.3 Chuyển đổi sang thư viện dùng cho Android 25](#_Toc362676641)

[2.2.3.1 Quá trình chuyển đổi 25](#_Toc362676642)

[2.2.3.1 Sử dụng trong Android project 27](#_Toc362676643)

[2.2.3.3 API 28](#_Toc362676644)

[2.3. Module nhận dạng tiếng Việt. 29](#_Toc362676645)

[2.3.1. Cơ sở lý thuyết. 29](#_Toc362676646)

[2.3.1.1 Phương pháp trích chọn đặc trưng MFCC 29](#_Toc362676647)

[2.3.1.2 Mô hình Markov ẩn liên tục 30](#_Toc362676648)

[2.3.2. Triển khai module nhận dạng trong Viva. 32](#_Toc362676649)

[2.3.2.1 Xây dựng mô hình nhận dạng trên Sphinx4 32](#_Toc362676650)

[2.3.2.2 Xây dựng chương trình nhận dạng 32](#_Toc362676651)

[2.4. Module hiểu ngữ nghĩa. 34](#_Toc362676652)

[2.4.1 Cơ sở lý thuyết. 34](#_Toc362676653)

[2.4.1.1 Tách từ tiếng Việt 34](#_Toc362676654)

[2.4.1.1 Automat 35](#_Toc362676655)

[2.4.1.2 Automat hữu hạn đơn định – DFA (Deterministic Finite Automata) 37](#_Toc362676656)

[2.4.2 Triển khai module hiểu ngữ nghĩa trong Viva. 38](#_Toc362676657)

[2.5. Kiến trúc VIVA trên android. 38](#_Toc362676658)

[2.5.1. Giới thiệu kiến trúc Android. 38](#_Toc362676659)

[2.5.2. Kiến trúc ứng dụng Viva. 38](#_Toc362676660)

[2.5.2.1. Biểu đồ lớp 38](#_Toc362676661)

[2.5.2.2. Mô tả lớp 41](#_Toc362676662)

[CHƯƠNG 3. KHỐI CHỨC NĂNG TIN NHẮN 48](#_Toc362676663)

[3.1 Chức năng gửi tin nhắn. 48](#_Toc362676664)

[3.1.1 Mô tả. 48](#_Toc362676665)

[3.1.2 Biểu đồ hoạt động. 49](#_Toc362676666)

[3.1.3 Biểu đồ tuần tự. 50](#_Toc362676667)

[3.2 Chức năng đọc tin nhắn đến. 52](#_Toc362676668)

[3.2.1 Mô tả. 52](#_Toc362676669)

[3.2.2 Biểu đồ hoạt động. 53](#_Toc362676670)

[3.2.3 Biểu đồ tuần tự. 54](#_Toc362676671)

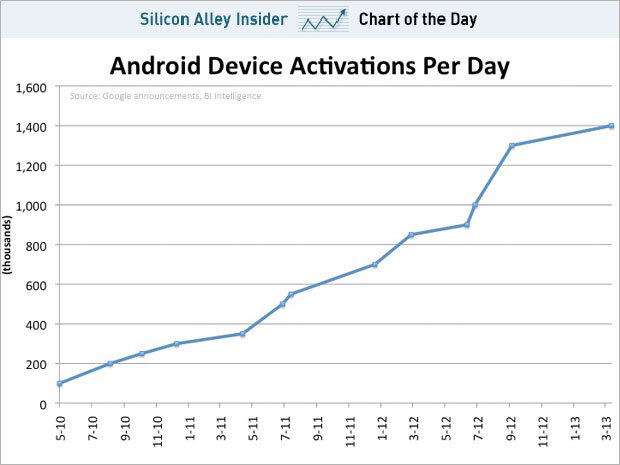
# GIỚI THIỆU

## 1.1 Đặt vấn đề.

Tương tác người – máy bằng tiếng nói là mảng đề tài rộng, mặc dù không mới nhưng luôn phát triển từng ngày về công nghệ, với mục tiêu chung là làm cho máy móc nghe được, hiểu được yêu cầu của con người và thực hiện.

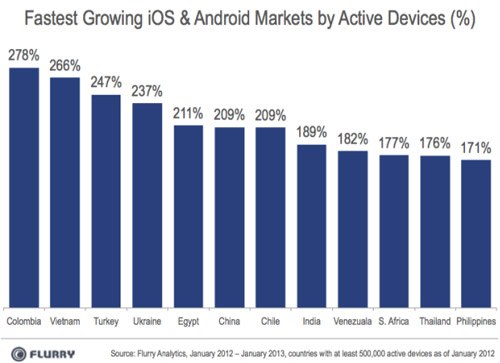
Các thiết bị di động thông minh như Smartphone, Tablet đang phát triển với tốc độ vũ bão và dần trở thành một phần không thể thiếu trong cuộc sống của con người hiện đại. Chiếc điện thoại giờ đây không chỉ là một thiết bị nghe gọi, những chiếc điện thoại thông minh có thể phục vụ con người trong tất cả các lĩnh vực giải trí, công việc, học tập, cuộc sống thường nhật. Giao tiếp với điện thoại thông minh bằng tiếng nói là một đề tài thú vị, một xu hướng công nghệ hiện nay, với mục tiêu biến chiếc điện thoại thành thư ký ảo cho người dùng, giúp đỡ người dùng thực hiện các thao tác mà không cần chạm vào màn hình, tạo ra sự tiện lợi và trải nhiệm thú vị.

Theo thống kê [1], chỉ tính riêng hệ điều hành Android, tính đến tháng 03/2013 đã có tổng cộng 1.230.000.000 thiết bị Android được kích hoạt, và hiện tại mỗi ngày có thêm 1.400.000 thiết bị Android được kích hoạt.



Hình 1.1 Số thiết bị Android được kích hoạt mỗi ngày [1]

Tính đến tháng 11/2013 Việt Nam đang là nước có tốc đột tăng trưởng về thiết bị Android & iOS đứng thứ hai thế giới [2]



Hình 1.2 Các nước có tốc độ tăng trưởng thiết bị Android & iOS cao nhất [2]

Rất nhiều hãng sản xuất điện thoại thông minh, công ty công nghệ phần mềm hàng đầu trên thế giới đang đầu tư, nghiên cứu phát triển các hệ thống tương tác với smartphone bằng tiếng nói như Apple, Google, Nuance, Samsung… với các sản phẩm Siri, Google Now, Nuance Talks, S Voice…

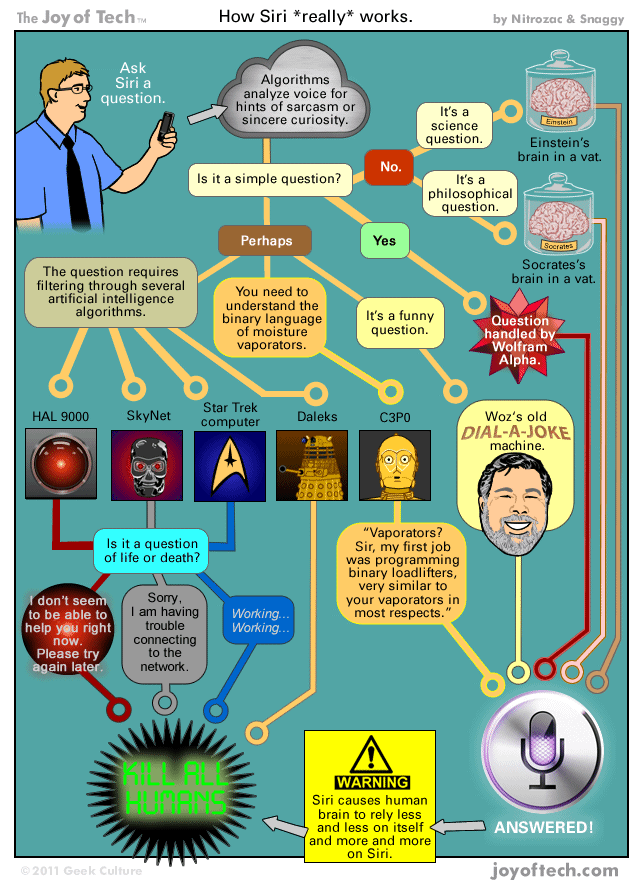
Tuy nhiên chưa hề có một hệ thống giúp người dùng tương tác với điện thoại bằng tiếng Việt, chính vì vậy chúng em đã lựa chọn đề tài này làm đồ án tốt nghiệp.

## 1.2 Khảo sát các thống có chức năng tương tự.

Trên thế giới các hãng công nghệ đã xây dựng các hệ thống tương tác với điện thoại qua tiếng nói như thế nào? Trước hết chúng ta hãy khảo sát các hệ thống hiện có trên thế giới để trả lời câu hỏi này.

a. Siri [3]

* Công bố vào tháng 11/2011, do Apple phát triển, chạy trên nền tảng hệ điều hành iOS.
* Ngôn ngữ hỗ trợ: English, French, German, Japanese, Spanish, Italian, Korean, Madarin, Cantonese.
* Các chức năng chính: bản đồ, tìm kiếm trên web, ghi chú, gửi Twitter tweet , tìm liên hệ trong danh bạ, mở ứng dụng, tìm địa danh, gọi điện, gửi email, SMS, đặt báo thức, truy vấn thông tin: thời tiết, thể thao, phim, kiến thức (Wolfram Alpha).
* Sơ đồ hoạt động:



Hình 1.3 Mô phỏng sơ đồ hoạt động của chương trình Siri [4]

b. Google Now [5]

Phát triển bởi GooglFigure 1e, được tích hợp với Android OS 4.1, công bố vào tháng 07/2012.

* Ngôn ngữ hỗ trợ: English.
* Chức năng chính: thống kê thông số sức khỏe người dùng khi chạy bộ, đạp xe, tìm kiếm ngày sinh, sự kiện, lịch bay, thao tác với Gmail, tìm kiếm thông tin: phim, tin tức, cuộc họp, hình ảnh, địa điểm, đề tài nghiên cứu, thể thao, chứng khoán, thông tin giao thông, du lịch, dịch máy, thông tin thời tiết.

c. Một số hệ thống khác: TellMe (Microsoft), S-Voice (Samsung), Iris (Dexetra), Speaktoit Assistant (Speaktoit, Inc), Vlingo (Vlingo, Inc), Evi (Evi Technologies Ltd).

## 1.3 Dự án Viva

### 1.3.1 Giới thiệu

Để thực hiện đề tài tương tác với smartphone bằng tiếngViệt, dự án Viva (**VI**etnamese **V**oice **A**ssistant) được thành lập tại viện nghiên cứu Mica. Dự án được thực hiện bởi các sinh viên: Nguyễn Duy Bình, Nguyễn Hữu Hiển, Nguyễn Thanh Tùng, Nguyễn Văn Hiếu, Nguyễn Văn Bảo, dưới sự dẫn dắt của TS.Trần Đỗ Đạt, TS.Nguyễn Quốc Cường và TS.Mạc Đăng Khoa.

### 1.3.2 Mục tiêu

Dự án Viva được thành lập với mục tiêu xây dựng hệ thống có thể:

* Nghe được: nhận dạng tiếng nói (tiếng Việt).
* Hiểu được: hiểu được câu lệnh của người dùng.
* Nói được: phản hồi lại người dùng bằng tiếng nói.
* Làm thư ký: giúp người dùng thực hiện các thao tác trên điện thoại mà không cần chạm vào bàn phím hay mèn hình để điều khiển, các tương tác sẽ được yêu cầu, trả lời kết quả, thông tin qua tiếng nói.

Dựa trên sự hướng dẫn của các giảng viên trên Mica cùng với một số kết quả khoa học đã được nghiên cứu trên Mica, nhóm Viva có nhiệm vụ:

* Xây dựng mới khối nhận dạng tiếng nói tiếng Việt.
* Xây dựng mới khối hiểu câu lệnh tiếng Việt.
* Áp dụng kết quả tổng hợp tiếng nói từ chương trình Hoa Súng để xây dựng module tổng hợp tiếng.
* Xây dựng ứng dụng thư ký ảo Viva trên điện thoại thông minh, kết nối ba khối trên cùng với các module thực thi chức năng.
* Xây dựng Viva theo hướng mở: có thể mở rộng được miền nhận dạng, khối hiểu câu lệnh, thêm mới các module chức năng Viva có thể thực hiện.

### 1.3.3 Chức năng

Dựa trên những khảo sát từ các hệ thống tương tự đang có, Viva sẽ được phát triển trước tiên trên hệ điều hành Android với những chức năng sau:

1. Giao tiếp đơn giản.
   1. Thư ký ảo có khả năng giao tiếp với người dùng các câu tiếng Việt thông dụng.
      * Chào hỏi
      * Trả lời tên, tuổi
      * Trả lời tâm trạng...
   2. Các câu giao tiếp có ý nghĩa phù hợp với ngữ cảnh giao tiếp.
   3. Cùng một câu nói của người dùng, thư ký ảo có thể đưa ra 1 hoặc nhiều câu trả lời dạng khác nhau để việc giao tiếp không trở nên nhàm chán.
   4. Thư ký ảo có thể chủ động giao tiếp với người dùng:
      1. Chào, hỏi người người dùng khi mở ứng dụng.
         * Hôm nay tâm trạng bạn thế nào? Bạn có muốn nghe một bản nhạc không?
         * Tôi có thể giúp gì cho bạn?
         * Bạn đang có 1 email, tin nhắn chưa đọc.Bạn có muốn tôi mở email/ trả lời tin nhắn không?
      2. Đặt câu hỏi khi không nhận dạng được tiếng nói hay không thể phân tích chính xác nghĩa của từ mà người dùng nói.
         * Ý của bạn là .... phải không ?
         * Tôi không nghe rõ, bạn vui lòng nhắc lại
   5. Hệ thống có khả năng tiếp nhận, mở rộng thêm dữ liệu về các câu giao tiếp để có thể làm cho khả năng giao tiếp của thư ký ảo ngày càng phong phú, chuẩn xác hơn.
2. Tương tác với ứng dụng.
   1. Đóng, mở ứng dụng theo tên.
   2. Tương tác với các ứng dụng nền tảng có sẵn trên hệ điều hành Android.
      1. Ứng dụng nhắn tin:
         1. Soạn, gửi tin nhắn tới số điện thoại hoặc tên người nhận trong danh bạ.
         2. Thông báo, đọc, trả lời tin nhắn tới.
      2. Ứng dụng gọi điện:
         1. Gọi điện tới số điện thoại.
         2. Gọi điện tới tên liên lạc trong danh bạ
         3. Tra cứu cước viễn thông, số tiền trong tài khoản
      3. Ứng dụng Email:
         1. Soạn, gửi email
         2. Thông báo, đọc, trả lời email đến
      4. Ứng dụng bản đồ
         1. Tìm một địa danh theo tên trên bản đồ
         2. Trả lời khoảng cách giữa hai địa điểm
         3. Tìm các địa điểm theo từ khóa gần người dùng: cây ATM, quán cà phê, nhà hàng, khách sạn, sân bóng…
      5. Ứng dụng báo thức
         1. Đặt báo thức
         2. Hủy báo thức
      6. Ứng dụng Google Search
         1. Mở Google tìm kiếm theo từ khóa
      7. Ứng dụng chơi nhạc, video
         1. Mở bài hát trong playlist
         2. Mở video trong playlist
      8. Notification.
         1. Đọc các thông báo của hệ điều hành.(Nếu ngôn ngữ hệ điều hành đang sử dụng khác tiếng Việt thì qua một bước dịch sang tiếng Việt).
   3. Các ứng dụng phổ biến có cung cấp API (Application Programming Interface)
      1. Mạng xã hội Facebook
         1. Đăng bài viết
         2. Chụp hình và đăng lên trang cá nhân
      2. Mạng xã hội Twiter
         1. Tweet - đăng tin nhắn lên Twiter
      3. Mạng xã hội Instagram
         1. Chụp hình và đăng lên trang cá nhân
3. Truy vấn thông tin: người dùng hỏi, thư ký ảo tìm & trả lời các thông tin.
   1. Nguồn thông tin trên internet:
      1. Tỉ giá tiền tệ.
      2. Thời tiết.
      3. Giá vàng.
      4. Chứng khoán.
      5. RSS tin tức từ các trang báo mạng tiếng Việt.
      6. Thông tin về con người, sự vật, sự việc từ nguồn Wikipedia tiếng Việt.
   2. Nguồn thông tin trên điện thoại:
      1. Thời gian, ngày tháng.

Trong khuôn khổ đồ án tốt nghiệp, nhóm đặt mục tiêu và hoàn thành các khối nhận dạng tiếng nói, khối hiểu câu lệnh, khối tổng hợp, thiết kế và xây dựng cở sở hạ tầng cho ứng dụng Viva trên nền Android 2.3 cùng với các chức năng cơ bản: **2.2.1, 3.1.1, 3.1.2, 3.1.3**, việc xây dựng thêm các chức năng còn lại sẽ được triển khai trong thời gian tới.

## 1.4 Kế hoạch xây dựng Viva

Mỗi thành viên trong Viva được phân công một mảng, học tập và nghiên cứu một mặt để xây dựng Viva, một mặt phát triển làm đồ án tốt nghiệp. Nhiệm vụ của mỗi thành viện được liệt kê trong bảng sau:

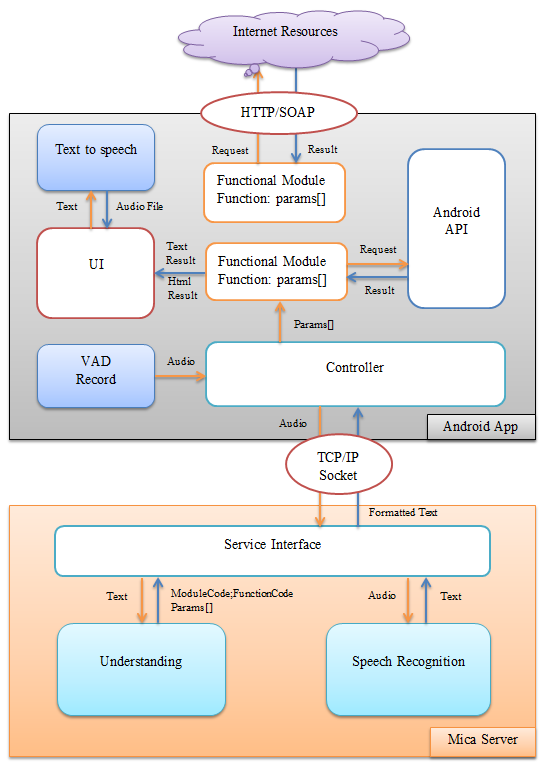
Bảng 1.1 Bảng phân công công việc cho các thành viên nhóm Viva

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Công việc** | **Tên đồ án tốt nghiệp** |
| Nguyễn Duy Bình | 1. Phân tích, thiết kế cơ sở hạ tầng cho ứng dụng Viva. 2. Áp dụng kết quả từ chương trình Hoa Súng để xây dựng module tổng hợp tiếng của Viva, chạy trên điện thoại. 3. Xây dựng module chức năng tin nhắn. 4. Chuyển nội dung văn bản tiếng Việt không dấu thành có dấu phục vụ cho chức năng đọc tin nhắn. 5. Viết chương trình thu âm để tạo cơ sở dữ liệu huấn luyện khối nhận dạng. | Tương tác với điện thoại thông minh bằng tiếng nói |
| Nguyễn Hữu Hiển | 1. Xây dựng module thu tiếng nói, tiền xử lý làm đầu vào cho module nhận dạng tiếng nói. 2. Xây dựng module phát hiện tiếng nói, tự động bắt đầu thu và ngắt thu âm khi người dùng nhập câu lệnh bằng tiếng nói. | Ứng dụng thuật toán phát hiện tiếng nói trong hệ thống nhận dạng tiếng nói. |
| Nguyễn Thanh Tùng | 1. Xây dựng module nhận dạng tiếng nói dựa trên Sphinx 4. | Xây dựng hệ thống nhận dạng online trên nền tảng sphinx4 |
| NguyễnVăn Hiếu | 1. Xây dựng khối hiểu câu lệnh. 2. Ghép nối các module vào ứng dụng Viva. |  |
| Nguyễn Văn Bảo | 1. Nâng cao chất lượng module tổng hợp tiếng. 2. Xây dựng module chức năng truy vấn thông tin. |  |

# CƠ SỞ LÝ THUYẾT & KIẾN TRÚC HỆ THỐNG VIVA.

## 2.1 Sơ đồ khối hệ thống

### 2.1.1 Sơ đồ tổng quan



Hình 2.1 Sơ đồ khối hệ thống Viva

### 2.1.2 Mô tả các khối

Hệ thống Viva bao gồm các khối chính sau :

1. ***Khối VAD :***

* **Mô tả :** Tiếng nói của người dùng được thu âm qua khối VAD, khối VAD tự động phát hiện tiếng nói để bắt đầu ghi âm, tự động ngắt ghi âm khi người dùng ngừng nói quá một khoảng thời gian đặt trước (0.5giây). Sau các bước lọc nhiễu, nén file, khối VAD gửi file âm thanh san khối điều khiển chính để gửi lên server nhận dạng.
* **Đầu vào :** Tiếng nói
* **Đầu ra :** File thu âm

1. ***Khối Controller (khối điều khiển):***

* **Mô tả:** Khối Controller làm nhiệm vụ giao tiếp với Server chạy khối nhận dạng và khối hiểu qua TCP/IP socket. Khối Controller gửi file ghi âm đã được thu và xử lý bởi khối VAD lên Server Mica, sau khi nhận được kết quả là đoạn text có định dạng, chứa thông tin nội dung câu nói hệ thống nhận dạng được, mã module chức năng, mã lệnh và các giá trị tham số sẽ thực hiện. Từ kết quả trả về, khối Controller thực hiện tìm tới module chức năng và gọi thực thi lệnh tương ứng với các tham số đầu vào nếu có.
* **Đầu vào:** File ghi âm từ khối VAD.
* **Đầu ra:** Kích hoạt module chức năng và lệnh tương ứng.

1. ***Khối Service Interface (Khối giao diện dịch vụ đặt trên Server Mica):***

* **Mô tả:** Khối làm nhiệm vụ tiếp nhận các request TCP/IP Socket từ các client do khối Controller gửi lên. Từ các file ghi âm gửi lên, khối gọi tới khối nhận dạng để thu về câu nói dạng text tương ứng, tiếp theo khối gọi tới khối hiểu để phân tích yêu cầu của người dùng và trả lại kết quả về client.
* **Đầu vào:** File ghi âm.
* **Đầu ra:** Đoạn text chứa câu nhận dạng được và kết quả phân tích bởi khối hiểu.

1. ***Khối nhận dạng tiếng nói:***

* **Mô tả:** Khối làm nhiệm vụ nhận dạng các file ghi âm và trả về câu nói dạng text tương ứng.
* **Đầu vào:** File ghi âm.
* **Đầu ra:** Text.

1. ***Khối hiểu câu lệnh:***

* **Mô tả:** Khối làm nhiệm vụ phân tích ngữ nghĩa câu lệnh của người dùng, từ đó xác định xem hệ thống phải gọi tới module chức năng nào, thực thi lệnh với các tham số trích xuất được trong câu lệnh để đáp ứng lại người dùng.

Thí dụ: người dùng yêu cầu “tôi muốn xem thông tin thời tiết ngày mai tại hà nội”, khối hiểu sẽ phân tích và trả về:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *ModuleCode* | QUERYINFO | *FunctionCode* | WEATHER | *Param* | Hà nội | *Param* | Ngày mai |

* **Đầu vào:** Text.
* **Đầu ra:** Text có định dạng.

1. ***Các khối chức năng:***

* **Mô tả:** Các khối chức năng bao gồm các function làm nhiệm vụ thức thi yêu cầu của người dùng, các khối này được xây dựng theo danh sách các chức năng của Viva liệt kê tại mục **1.3.3**.

Các chức năng sau khi được kích hoạt bởi khối Controller sẽ chạy theo một kịch bản được lập trình sẵn, trong quá trình thực thi có thể sẽ phải gọi tới API của hệ điều hành để thực hiện thao tác (nhắn tin, gọi điện, gửi email…) hoặc truy vấn tới nguồn thông tin trên Internet để lấy thông tin trả lời người dùng (thời tiết, tỉ giá, giá vàng, chứng khoán…)

Sau khi thực thi xong, sẽ gọi tới khối UI để hiển thị kết quả dạng text hoặc dạng Html view.

* **Đầu vào:** Tham số.
* **Đầu ra:** Text, Html result.

1. ***Khối UI (giao diện):***

* **Mô tả:** Khối làm nhiệm vụ hiển thị kết quả text – câu trả lời của Viva và các dữ liệu trực quan dạng Html (giá vàng, thời tiết…) cho người dùng. Đồng thời với việc hiển thị câu trả lời của Viva lên màn hình, khối UI sẽ gọi tới khối tổng hợp tiếng nói để thu về file tiếng và phát trả lời người dùng.
* **Đầu vào:** Câu trả lời dạng text, dữ liệu hiển thị dạng Html.
* **Đầu ra:** Kết quả hiện lên màn hình.

1. ***Khối tổng hợp tiếng nói:***

* **Mô tả:** Khối làm nhiệm vụ tổng hợp câu nói dạng text đưa vào và trả về file tiếng nói dạng âm thanh.
* **Đầu vào:** Text.
* **Đầu ra:** File tiếng nói.
* Khối nhận dạng tiếng nói và khối hiểu ngữ nghĩa phức tạp về cấu trúc, khi thực thi chiếm dụng tài nguyên (không gian ổ cứng, bộ nhớ, xử lý của CPU) rất lớn nên chưa thể tích hợp vào ứng dụng trên điện thoại mà được triển khai trên server của Mica, cho các client kết nối tới bằng TCP/IP socket. Chính vì lý do này, hệ thống Viva khi chạy luôn luôn cần kết nối mạng internet.

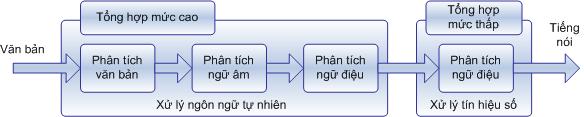
## 2.2 Khối tổng hợp tiếng Việt

### 2.2.1. Cơ sở lý thuyết

#### 2.2.1.1.Lý thuyết tổng hợp tiếng

Một nhu cầu rất quan trọng trong lĩnh vực tổng hợp tiếng nói là tổng hợp tiếng nói từ văn bản (Text To Speech – TTS). Quá trình này được chia làm hai mức xử lý:

* High Level Synthesis: Tổng hợp mức cao.
* Low Level Synthesis: Tổng hợp mức thấp.



Hình 2.2 Mô hình tổng hợp tiếng nói

* [**Tổng hợp mức cao**](#_Toc60680202)

Tổng hợp mức cao là giai đoạn đầu của quá trình tổng hợp, giai đoạn chuyển đổi các văn bản text thành các đơn vị tiếng nói (ví dụ như diphone). Văn bản được nhập hoặc sao chép vào, sau đó qua tổng hợp mức thấp sẽ thành tiếng nói.

Tổng hợp mức cao gồm 3 bước:

* Xử lý trước văn bản với các chữ số, các ký tự đặc biệt, chữ viết tắt, và những từ viết tắt được ghép bằng các chữ đầu của các từ đầy đủ...
* Phân tích cách phát âm của từ, kể cả từ đồng âm khác nghĩa và các tên riêng.
* Phân tích ngữ điệu của tiếng nói.

Sau khi tổng hợp mức cao, thông tin được cung cấp cho hệ thống mức thấp để điều khiển. Chẳng hạn, với bộ tổng hợp formant thì cần các thông tin như tần số cơ bản, tần số formant, khoảng thời gian, và biên độ của mỗi đoạn âm thanh.

* **[Tổng hợp mức thấp](#_Toc60680206)**

Tổng hợp mức thấp là quá trình tạo ra tín hiệu tiếng nói từ các thông tin được đưa tới từ tổng hợp mức cao (chuỗi các đơn vị âm và thông tin ngữ điệu).

Có 3 phương pháp cơ bản được thực hiện ở mức tổng hợp mức thấp:

* Phương pháp mô phỏng hệ thống phát âm.
* Phương pháp tổng hợp formant.
* Phương pháp tổng hợp ghép nối .

#### 2.2.1.2.Tổng hợp tiếng theo phương pháp ghép nối

Trong tổng hợp tiếng nói theo kiểu ghép nối, một đơn vị âm được tổng hợp đơn giản bằng cách lấy ra đoạn sóng âm tương ứng với âm vị tương ứng. Một lời phát biểu được tổng hợp bằng cách ghép nối đồng thời nhiều đơn vị âm lại với nhau. Tuy nhiên, nếu ta ghép nối 2 đoạn âm không liền kề nhau sẽ dẫn đến việc không liên tiếp về phổ và không liên tiếp về ngôn điệu. Sự không liên tiếp về phổ xảy ra khi formant tại điểm nối không hợp với nhau. Sự không liên tiếp về ngôn điệu xảy ra khi cao độ tại điểm nối là không hợp với nhau. Chất lượng âm thanh tổng hợp sẽ tồi khi mà có nhiều điểm không liên tiếp trong ghép nối, cho dù các đơn vị âm là hoàn toàn tự nhiên.

Thường thì, các hệ thống tổng hợp tiếng nói theo kiểu kết nối hay có sự thay đổi về chất lượng, có thể một hệ thống có chất lượng tốt, và hệ thống khác có thể là chất lượng tồi. Nếu có đủ lượng đơn vị âm, chất lượng âm thanh tổng hợp có thể tốt như âm thanh lúc thu. Tuy nhiên nếu xuất hiện một vài sự không liên tục sẽ gây cho chất lượng âm thành tổng hợp sẽ kém đi nhiều.

Chính vì vậy, khi thiết kế một hệ thống tổng hợp tiếng nói, ta cần quan tâm đến các vấn đề quan trọng sau:

* Lựa chọn loại đơn vị âm tối ưu
* Xây dựng cơ sở dữ liệu của các đơn vị âm
* Tìm kiếm đơn vị âm tối ưu để ghép nối
* Ghép nối các đơn vị âm
* **Lựa chọn loại âm vị tối ưu:**

Tiếng Việt là ngôn ngữ đơn âm tiết. Âm tiết là đơn vị âm thanh nhỏ nhất có thể cảm nhận được. Cấu trúc của âm tiết trong tiếng Việt gồm 5 phần: âm đầu, âm đệm, âm chính, âm cuối và thanh điệu.

Tổng quan về các loại đơn vị âm trong tiếng Việt, các nghiên cứu trên Viện MICA cho thấy rằng, đối với tiếng Việt tồn tại bốn kiểu đơn vị âm (âm vị kép, bán âm tiết, âm đầu/vần, âm tiết) có thể sử dụng cho quá trình tổng hợp. Tuy nhiên kiểu đơn vị âm âm đầu/vần sẽ không phải là loại đơn vị âm tối ưu do sự không liên tục giữa âm đầu (initial part) và vần (final part). Còn đối với trường hợp đơn vị âm là âm tiết, để một hệ thống có thể tổng hợp được bất kỳ đoạn văn bản nào thì đòi hỏi phải có một cơ sở dữ liệu rất lớn. Do vậy mà âm vị kép và bán âm tiết hiện vẫn đang là những loại đơn vị âm được sử dụng phổ biến nhất trong các hệ thống tổng hợp tiếng nói bằng phương pháp ghép nối. Viện nghiên cứu MICA cũng đang nghiên cứu và sử dụng chủ yếu hai loại đơn vị âm này trong chương trình tổng hợp tiếng nói được phát triển tại Viện và trong đó, cơ sở dữ liệu cho loại đơn vị âm là bán âm tiết được xây dựng và phát triển đầy đủ nhất.

* **Xây dựng cơ sở dữ liệu của các đơn vị âm**

Tiếng Việt là ngôn ngữ có thanh điệu, ngoài việc ảnh hưởng của khớp nối giữa các đơn vị âm cạnh nhau, đơn vị âm cũng bị ảnh hưởng bởi thanh điệu của âm tiết. Trong ngôn ngữ có thanh điệu đơn vị âm được chia thành hai loại: đơn vị âm không thanh điệu và đơn vị âm có thanh điệu. Đơn vị âm không thanh điệu là đơn vị âm thu được nhờ việc tách ra từ các âm tiết có thanh ngang. Đơn vị âm có thanh điệu thu được từ các âm tiết chứa thông tin về thanh điệu. Với đơn vị âm không thanh điệu, sau khi ghép nối để tổng hợp nên các âm tiết không thanh điệu, thanh điệu được thêm vào bằng cách điều khiển trực tiếp sự thay đổi của F0 của âm tiết. Trong khi đó, với đơn vị âm có thanh điệu, những thông tin về thanh điệu được lưu trong đơn vị âm, vì vậy ta không cần phải điều khiển đường cong F0.

* **Tìm kiếm đơn vị âm tối ưu để ghép nối**

Bước tìm kiếm đơn vị âm tối ưu nhằm tìm kiếm trong cơ sở dữ liệu các đơn vị âm tốt nhất sao cho sau khi ghép nối ta thu được chất lượng tiếng nói là tốt nhất có thể. Các đơn vị âm tốt nhất là các đơn vị thỏa mãn sao cho độ méo tiềm tàng (về ngữ điệu, về phổ) giữa chúng là thấp nhất.

Có hai phương pháp để chọn lựa ra các đơn vị âm tối ưu cho quá trình tổng hợp:

* *Chọn lựa dựa trên mô hình cây quyết định:* dữ liệu đọc được nhóm lại trong một cây bằng cách phân đoạn mỗi nút thành các nút cấp dưới dựa trên dữ liệu âm học, bằng cách sử dụng các tiêu chuẩn được đề xuất bởi nhãn ngữ cảnh của dữ liệu. Điều này tạo ra một số lượng nhóm, mỗi nhóm chứa các phân đoạn giống nhau ở cùng một mức ngữ cảnh và âm học. Nhóm được sử dụng cho một ngữ cảnh nào đó trong quá trình tổng hợp có thể lấy ra từ cây tương ứng hoặc sử dụng các kết quả giống về ngữ cảnh nhằm mục đích so sánh nhãn của ngữ cảnh yêu cầu với nhãn của các nhóm có thể sử dụng được.
* *Chọn lựa dựa trên việc tối ưu hóa hàm giá trị:* phương pháp này sẽ tìm ra các units dựa trên một lượng các instances cho trước với mục đích là tối thiểu hóa hàm giá trị, hay tương ứng với việc các loại méo và nhiễu xảy ra trong quá trình ghép nối là nhỏ nhất.
* **Ghép nối các đơn vị âm**

Vấn đề lớn nhất đối với kỹ thuật tổng hợp ghép nối là ghép nối các đơn vị âm. Chất lượng của tiếng nói tổng hợp được nhờ kỹ thuật này phụ thuộc rất nhiều vào phương pháp hay thuật toán được sử dụng để ghép nối các đơn vị âm. Thông thường, việc ghép nối các đơn vị âm dễ dẫn đến hiện tượng không liên tục tại điểm ghép nối giữa các đơn vị âm (về cao độ, về phổ, về pha). Những sự không liên tục này gặp phải có thể là do có sự khác nhau về ngữ cảnh của các đơn vị âmhay do chính chất lượng không tốt của các đơn vị âm gây ra bởi quá trình thu âm, quá trình phân đoạn…

Đối với kỹ thuật tổng hợp ghép nối của tổng hợp tiếng nói, mô hình tổng hợp tiếng nói trong miền thời gian là mô hình cho phép thu được tiếng nói với chất lượng cao mà chi phí tính toán lại thấp hơn rất nhiều. Sự phát triển của tổng hợp tiếng nói trong miền thời gian gắn liền với sự phát triển của phương pháp PSOLA, một phương pháp cho phép điều khiển các tham số ngữ điệu của tiếng nói và ghép nối các đơn vị âm.

***Phương pháp TD-PSOLA***

PSOLA (Pitch Synchronous Overlap and Add) còn được gọi là kỹ thuật chồng đồng bộ cao độ tần số cơ bản. Nó thực hiện việc biến đổi trường độ và cao độ [âm thanh](http://vi.wikipedia.org/wiki/%C3%82m_thanh) hay là quá trình thay đổi cao độ [tần số âm cơ bản](http://vi.wikipedia.org/wiki/T%E1%BA%A7n_s%E1%BB%91_%C3%A2m_c%C6%A1_b%E1%BA%A3n) F0 và độ dài tiếng nói ở miền thời gian trực tiếp trên dạng sóng tiếng nói.Kỹ thuật này được sử dụng trong [tổng hợp giọng nói](http://vi.wikipedia.org/wiki/T%E1%BB%95ng_h%E1%BB%A3p_gi%E1%BB%8Dng_n%C3%B3i).

Kỹ thuật PSOLA được diễn ra qua ba bước là:

* Phân tích tín hiệu gốc thành các chuỗi âm ngắn theo các cửa sổ đồng bộ với các điểm đánh dấu pitch (còn gọi là epoch).
* Biến đổi các âm phân tích được ở trên thành các âm ngắn mới theo các biến đổi độ dài và tần số âm cơ bản.
* Tổng hợp lại các âm ngắn này theo phương pháp cộng chồng, đồng bộ lại tín hiệu.

Giải thuật PSOLA có nhiều phiên bản như FD-PSOLA, LP-PSOLA, TD-PSOLA. Trong các phiên bản này TD-PSOLA được sử dụng rộng rãi do chi phí tính toán thấp và chất lượng âm thanh tổng hợp đem lại.

TD-PSOLA là phiên bản trên miền thời gian của giải thuật PSOLA (Pitch Synchronous Overlap-Add). Với PSOLA, tín hiệu tổng hợp được tạo nên bằng cách cộng xếp chồng (Overlap-Add) các đoạn tín hiệu thành phần. Giải thuật này cho phép thao tác trực tiếp với tín hiệu tiếng nói trên miền thời gian, thay đổi tần số cơ bản và độ dài của tín hiệu.

### 2.2.2 Chương trình Hoa Súng – Mica

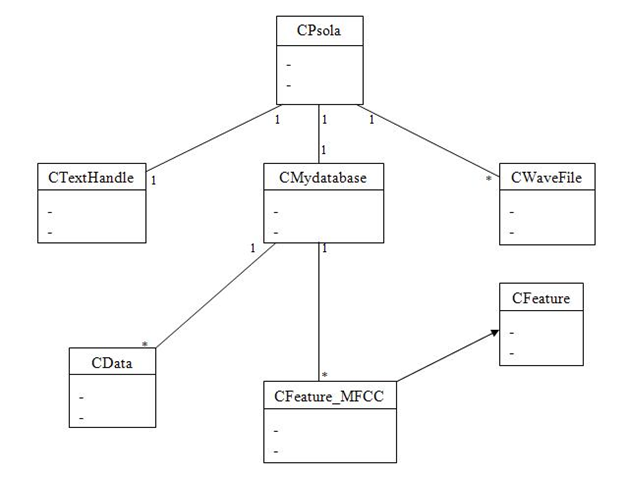
#### 2.2.2.1 Giới thiệu chương trình

Chương trình tổng hợp tiếng nói Hoa Súng là chương trình tổng hợp tiếng Việt có chất lượng khá cao, dễ hiểu và tương đối tự nhiên trên PC được phát triển bởi các chuyên gia nghiên cứu của Viện MICA. Chương trình được viết dựa trên nền tảng lập trình hướng đối tượng bằng công cụ lập trình Microsoft Visual C++ 6.0. Phiên bản đầu tiên của chương trình (phiên bản demo 1.0) được hoàn thành vào năm 2007 với sự tham gia của Trần Đỗ Đạt và Lê Xuân Hùng.

Chương trình cho phép điều khiển các tham số ngữ điệu (tần số cơ bản, trường độ) của tiếng nói cần tổng hợp theo ý muốn đồng thời còn cho phép người dùng có thể lựa chọn loại đơn vị âm để tổng hợp như: âm vị kép (Diphone), bán âm tiết (Half-Syllable), âm tiết (Base Syllable), âm đầu/vần (initial/final part) và dùng kết hợp các loại đơn vị âm để tổng hợp.

#### 2.2.2.2 Biểu đồ lớp của chương trình

Chương trình tổng hợp tiếng nói Hoa Súng bao gồm các lớp chính (các lớp liên quan đến các module tổng hợp tiếng nói) như trên biểu đồ sau đây:



Hình 2.3 Biểu đồ lớp của chương trình Hoa Súng

* CPsola: chứa các phương thức dùng cho việc điều khiển các tham số về ngữ điệu của tiếng nói và ghép nối các đơn vị âm, làm trơn năng lượng giữa các âm tiết, đây có thể coi là lớp trung tâm của chương trình.
* CTextHandle: chứa các phương thức về phân tích văn bản đầu vào, lấy ra các thông tin về âm tiết…
* CMyDatabase: chứa các phương thức để thao tác với các file cơ sở dữ liệu (tìm kiếm, lấy ra các thông tin về đơn vị âm…) cũng như các phương thức về chọn lựa đơn vị âm tối ưu.
* CWaveFile: chứa các phương thức thao tác với file .wav như tạo mới một file .wav, thêm dữ liệu vào một file .wav đã tồn tại cũng như lấy ra các dữ liệu từ file .wav.
* CData, CFeature và CFeature\_MFCC: chứa các thuộc tính và phương thức về xử lý tín hiệu số (tính FFT, các hệ số MFCC…) và trích chọn đặc tính (tính khoảng cách phổ, khoảng cách về ngữ cảnh…).

#### 2.2.2.3 Quá trình tổng hợp tiếng nói của chương trình

Chương trình tổng hợp tiếng nói HOA SÚNG sẽ có nhiệm vụ nhận văn bản đầu vào (gõ trực tiếp từ bàn phím hoặc từ file văn bản) và tạo ra file tiếng nói ở đầu ra là file .wav. Quá trình tổng hợp tiếng nói theo chương trình sẽ chia làm hai giai đoạn (block) lớn sau đây:

* Giai đoạn 1: đây là giai đoạn phân tích văn bản đầu vào, sử dụng các mô hình ngữ điệu và mô hình thanh điệu để lấy ra các thông tin cần thiết (các đơn vị âm cấu thành và các tham số ngữ điệu) về các âm tiết cần tổng hợp. Bước tiếp theo của giai đoạn này là tìm kiếm trong cơ sở dữ liệu các đơn vị âm tối ưu (sử dụng phương pháp tối ưu hóa hàm chi phí) để phục vụ cho việc ghép nối.
* Giai đoạn 2: đây là giai đoạn điều khiển các tham số ngữ điệu thích hợp theo thuật toán TD - PSOLA và ghép nối các đơn vị âm đã được chọn lựa ở giai đoạn 1. Các đơn vị âm được ghép nối để tạo ra từng âm tiết và từng âm tiết sẽ được ghi lần lượt vào file .wav, mỗi khi ghi một âm tiết mới vào thì làm trơn mức năng lượng giữa âm tiết đó với âm tiết trước nó.



Hình 2.4 Biểu đồ hoạt động tổng quát của chương trình Hoa Súng

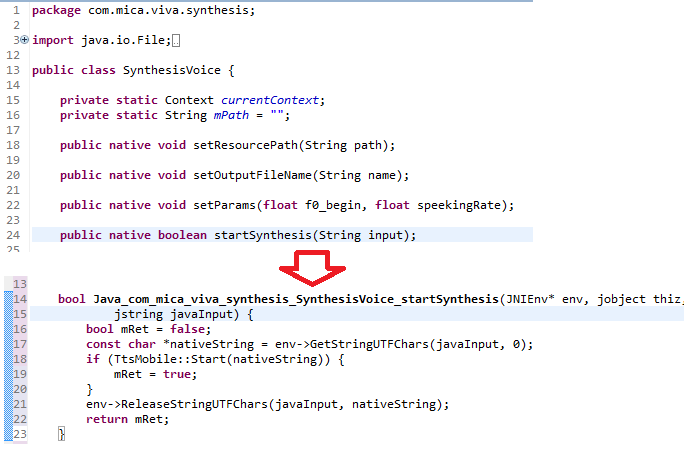
### 2.2.3 Chuyển đổi sang thư viện dùng cho Android

#### 2.2.3.1 Quá trình chuyển đổi

Việc triển khai trên hệ điều hành Android khá phức tạp, do các ứng dụng Android được viết bằng mã nguồn Java, được biên dịch thành bytecode rồi chuyển thành các file thực thi do Davik Virtual Machine nên ta không thể gọi trực tiếp tới mã nguồn C++ của thư viện tổng hợp Tiếng Việt ở trên được. Đế có thể sử dụng lại mã nguồn C++ trong chương trình Hoa Súng cho khối tổng hợp trên Viva, chúng ta cần thực hiện các bước sau:

**Bước 1:** Chuẩn hóa các thư viện dùng trong mã nguồn C++ của Hoa Súng: thay thế thư viện xử lý chuỗi CString, CArray của thư viện MFC (Microsoft Foundation Class) chỉ hộ trợ trên nền Windows bằng thư viện mã nguồn mở Standard Lib C (STL C).

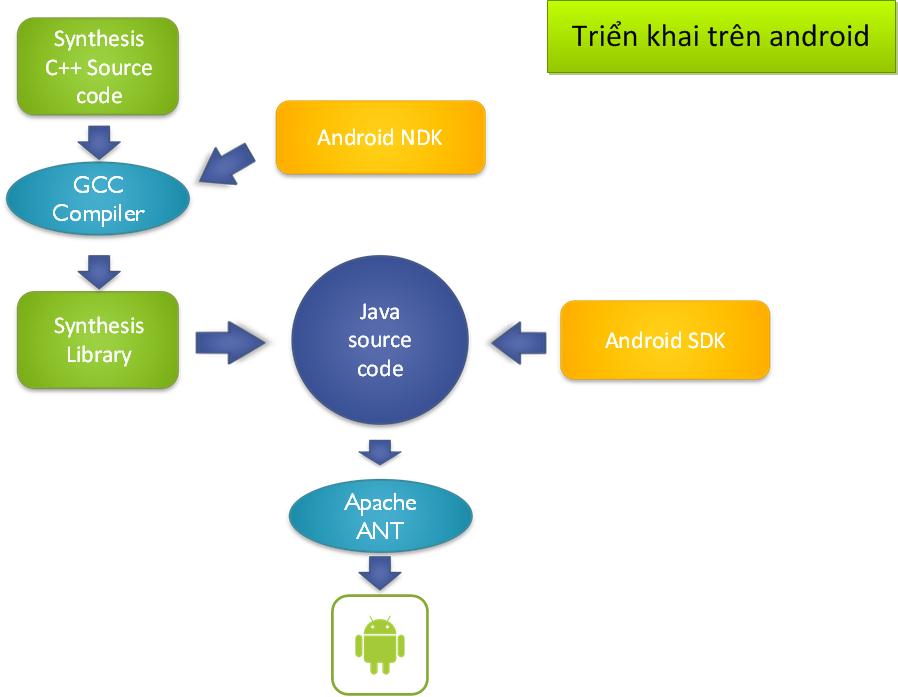
**Bước 2:** Khai báo các phương thức dạng native trong lớp SynthesisVoice và viết các phương thức thực thi tương ứng trong file main.cpp trong mã nguồn C++.



**Bước 3:** Biên dịch mã nguồn C++ thành thư viện liên kết động ( lib \*.so) rồi trong mã nguồn Java của chương trình , các thư viện này được nạp vào qua lời gọi **System.LoadLibrary().**

**E:\Study\study\DoAnTiengNoi\MyWorks\Resource\load-library.png**

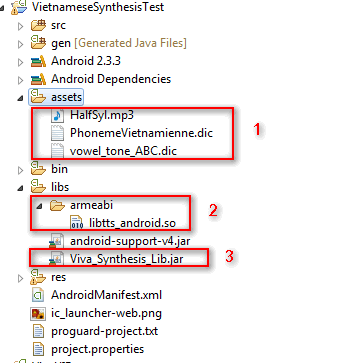
**Bước 4:** Đóng gói module tổng hợp tiếng com.mica.viva.synthesis thành thư viện \*.jar có thể dùng trong các project android.



Hình 2.5 Triển khai module tổng hợp tiếng trên Android

#### 2.2.3.1 Sử dụng trong Android project

**Bước 1:** thêm thư viện & dữ liệu.



Hình 2.6 Copy thư viện, dữ liệu module tổng hợp tiếng vào project Android

1. Copy các ***file dữ liệu*** vào thư mục ***assets***.
2. Copy thư viện ***libtts\_android.so*** vào thư mục ***libs/armeabi***.
3. Copy thư viện ***Viva\_Synthesis\_Lib.jar*** vào thư mục ***libs***.

**Bước 2:** sử dụng module tổng hợp tiếng.

Import package:

**import** com.mica.viva.synthesis.SynthesisVoice;

Khởi tạo đối tượng, gọi hàm để cấu hình (tự động) các tham số module tổng hợp:

SynthesisVoice syn = **new** SynthesisVoice();

syn.prepareToSynthesis(getApplicationContext());

Tổng hợp & phát câu nói:

syn.readText(text);

#### 2.2.3.3 API

Bảng 2.1 API module tổng hợp tiếng trên Android

|  |  |
| --- | --- |
| **package**  **public** **class** | com.mica.viva.synthesis  SynthesisVoice |
| **Public methods** | **Mô tả** |
| **void** setResourcePath(String path); | Thiết đặt đường dẫn tới các file dữ liệu.  Example:  "/data/data/" + getPackageName() + "/" |
| **void** setOutputFileName(String name ); | Thiết đặt đường dẫn + tên file tiếng nói được tổng hợp.  Example:  "/data/data/" + getPackageName() + "/result.wav" |
| **void** setParams(**float** f0\_begin, **float** speekingRate); | Thiết đặt tham số tổng hợp tiếng nói:  f0\_begin: cao độ (0-500).  speekingRate: trường độ (0-1). |
| **boolean** startSynthesis(String input); | Tổng hợp tiếng nói cho câu đưa vào, kết quả tổng hợp lưu tại file thiết đặt trước bởi hàm setOutputFileName  Return: true – tổng hợp thành công  false – tổng hợp không thành công |
| **void** prepareToSynthesis(Context mContext) | Tự động thiết đặt các tham số nêu trên:  Resource path:  "/data/data/" + getPackageName() + "/"  Output file:  "/data/data/" + getPackageName() + "/result.wav"  f0 begin: 210  speaking rate: 1 |
| **void** readText(String message) | Tổng hợp câu nói, đồng thời phát câu nói tổng hợp được. |

## 2.3. Module nhận dạng tiếng Việt.

Module nhận dạng tiếng Việt được xây dựng trên nền tảng Sphinx4 đặt trên server (server MICA) phục vụ cho nhiều client kết nối tới qua internet.

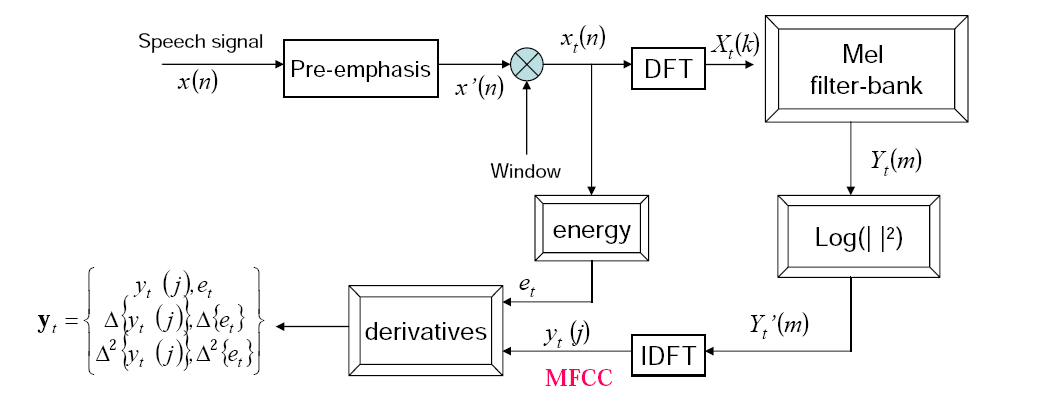
Sphinx4 là công cụ nhận dạng tiếng nói liên tục với bộ từ vựng lớn, không phụ thuộc vào người nói do trường đại học Carnegie Mellon University phát triển. Nó là tập hợp của các công cụ miễn phí và mã nguồn mở cho phép nhà nghiên cứu và phát triển xây dựng các hệ thống nhận dạng cho riêng mình. Sphinx4 là công cụ viết bằng ngôn ngữ lập trình Java, được phát triển trên nền tảng Sphinx2, Sphinx3. Nó cung cấp mô hình linh hoạt hơn trong việc phát triển hệ thống nhận dạng tiếng nói.

Mô hình được chọn để xây dựng hệ thống nhận dạng là mô hình Markov ẩn liên tục (Continous Density HMM). Phương pháp trích chọn đặc trưng được sử dụng là MFCC (Mel-scale Frequency Cepstral Coefficients).

### 2.3.1. Cơ sở lý thuyết.

#### 2.3.1.1 Phương pháp trích chọn đặc trưng MFCC

Phương pháp trích chọn đặc trưng MFCC có quy trình được thể hiện ở mô hình sau:



Hình 2.7 Sơ đồ thuật toán MFCC

* Speech signal: Tín hiệu âm thanh đầu vào, quá trình thu âm lời nói thực chất là quá trình biến đổi một tín hiệu liên tục thành một tín hiệu rời rạc (qua microphone).
* Pre - Emphasis:tăng cường năng lượng của các âm thanh ở tần số cao.
* Window: phân khung và cửa sổ hóa tín hiệu.
* DFT (Discrete Fourier Transform): chuyển đổi tín hiệu thời gian thành tín hiệu tần số.
* Mel filter-bank: dùng những cửa sổ lọc – filterbank, được sắp xếp một cách đồng đều để tính toán mel – spectrum.
* Logarit bình phương biên độ: tính logarit của bình phương biên độ tín hiệu đầu ra sau khi qua bộ lọc tần số Mel.
* IDFT: chuyển đổi logarit của mel spectrum về trục thời gian, kết quả được gọi là "mel- frequency cepstrum coefficients (MFCC)".
* Energy: tính toán hệ số đặc trưng của năng lượng : 
* Derivatives: tính toán hệ số để mô tả sự thay đổi của các hệ số đặc trưng theo thời gian. Các hệ số đó gọi là đặc trưng delta và đặc trưng double-delta

Một vectơ đặc trưng MFCC điển hình có định dạng như sau :

* Độ dài mỗi khung (frame) : 25 ms
* Khoảng cách các frame : 10 ms
* Hệ số pre-emphasis : 0.97
* Số lượng các hệ số đặc trưng :
  + 12 MFCC ( hệ số thang tần số Mel)
  + 12 hệ số delta MFCC
  + 12 hệ số delta bình phương MFCC
  + 1 hệ số đặc trưng năng lượng
  + 1 hệ số delta năng lượng
  + 1 hệ số delta bình phương của năng lượng

Như vậy, tổng cộng có 39 hệ số đặc trưng trong MFCC.

#### 2.3.1.2 Mô hình Markov ẩn liên tục

* **Mô hình Markov ẩn**

Mô hình Markov ẩn là mô hình thống kê trong đó hệ thống được mô hình hóa được cho là một quá trình Markov với các tham số không biết trước và nhiệm vụ là xác định các tham số ẩn từ các tham số quan sát được, dựa trên sự thừa nhận này. Các tham số của mô hình được rút ra sau đó có thể sử dụng để thực hiện các phân tích kế tiếp.

Trong một mô hình Markov điển hình, trạng thái được quan sát trực tiếp bởi người quan sát, và vì vậy các xác suất chuyển tiếp trạng thái là các tham số duy nhất. Mô hình Markov ẩn thêm vào các đầu ra: mỗi trạng thái có xác suất phân bổ trên các biểu hiện đầu ra có thể. Vì vậy, nhìn vào dãy của các biểu hiện được sinh ra bởi HMM không trực tiếp chỉ ra dãy các trạng thái.

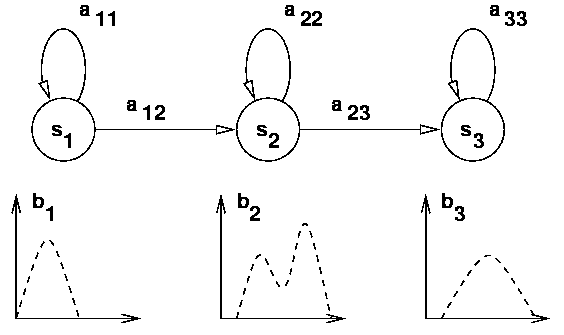
Mô hình Markov ẩn đặc trưng bởi các tham số sau :

* O = {o1, o2,..., oM} : Một bảng kí tự quan sát ở đầu ra. Các kí tự này đại diện cho đầu ra các đại lượng vật lí của hệ thống.
* Ω {1,2,..., N} : Tập hợp các trạng thái để biểu diễn một không gian trạng thái. st biểu diễn trạng thái tại thời điểm t.
* Aij = {aij} : Ma trận xác suất chuyển trạng thái. aij  là xác suất chuyển trạng thái từ i sang j.
* B = {bi(k)} : Ma trận xác suất đầu ra. bi(k) là xác suất đầu ra của kí tự ok khi trạng thái i được gia nhập.
* π = {πi} là phân bố trạng thái khởi tạo. πi = P (s0 = i)
* Chuỗi vecto quan sát X (X1, X2,..., XT).

Áp dụng mô hình HMM vào nhận dạng tiếng nói, ta phải giải quyết 3 vấn đề chính sau :

* Vấn đề đánh giá : Cung cấp  cho mô hình các tham số ( chuỗi quan sát O = (o1,o2, ..., oT) và một mô hình HMM (Φ = (A, B,π)) , tính xác suất của dãy đầu ra cụ thể (P (X | Φ). Giải bằng thuật toán tiến trước.
* Vấn đề giải mã : Cung cấp cho mô hình các tham số Φ và X , tìm dãy các trạng thái (ẩn) S (s0,s1,...,sT) có khả năng lớn nhất mà có thể sinh ra dãy đầu ra đã cung cấp. Giải bằng thuật toán Viterbi.
* Vấn đề học : Cung cấp dãy đầu ra X và mô hình Φ , tìm tập hợp có khả năng nhất của chuyển tiếp trạng thái và các xác suất đầu ra. Giải bằng thuật toán Baum-Welch.
* **Mô hình Markov ẩn liên tục (Continuous density HMM)**

Để đảm bảo cấu trúc ban đầu của tín hiệu nguyên thủy, người ta dùng mô hình Markov liên tục :



Hình 2.8 Mô hình Markov ẩn liên tục

Mô hình HMM liên tục là một mô hình tốt cho lí thuyết HMM. Do khối lượng tính toán khá lớn nên đặt hệ thống nhận dạng trên 1 máy chủ (server) là phù hợp. Mỗi trạng thái trong HMM liên tục cần có M vector riêng đặc trưng. Mỗi vector có thêm 1 ma trận hội tụ tương ứng với nó. Vector có số chiều càng lớn thì kích thước ma trận hội tụ càng lớn, tính toán càng phức tạp.

### 2.3.2. Triển khai module nhận dạng trong Viva.

Trong phạm vi đề tài, nhóm tập trung xây dựng cơ sở dữ liệu xoay quanh một số chủ đề chính : tin nhắn (soạn tin nhắn bằng tiếng nói, ra lệnh gửi, hủy... ), truy vấn thông tin (thời tiết, giá vàng,ngoại tệ, chứng khoán...) trên Internet. Mô hình nhận dạng dựa trên bộ từ vựng nhỏ (từ điển khoảng 250 từ), các kịch bản huấn luyện được xây dựng từ trước tập trung vào các chủ đề trên.

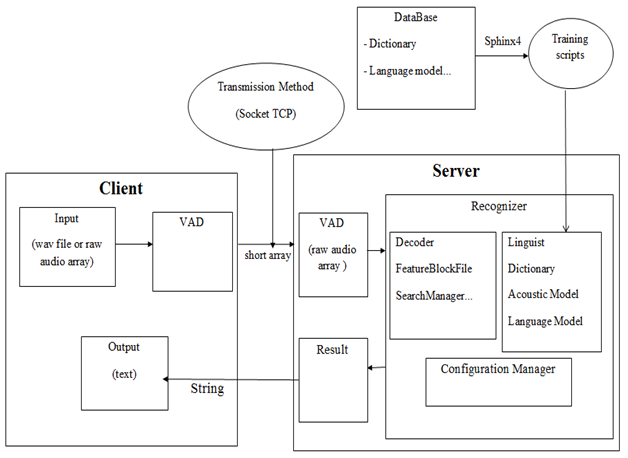
#### 2.3.2.1 Xây dựng mô hình nhận dạng trên Sphinx4

Việc xây dựng mô hình nhận dạng trên Sphinx4 được thực hiện qua 2 bước:

* Chuẩn bị cơ sở dữ liệu
* Xây dựng thư viện cơ sở dữ liệu và các mô hình

#### 2.3.2.2 Xây dựng chương trình nhận dạng

* **Sơ đồ khối chương trình**



Hình 2.9 Sơ đồ khối chương trình nhận dạng

* **Cách thức hoạt động**
* Môi trường truyền là mạng: Internet (mạng dây, 3G), LAN ( mạng dây, wifi )
* Kiểu kết nối: Multi Client – Server
* Client gửi cho server một mảng kiểu short là tín hiệu âm thanh đã được lấy mẫu
* Server nhận dạng và trả lại kết quả dạng text
* Phương thức truyền : Socket TCP

Cách thức hoạt động của cả hệ thống Multi client – server nhận dạng tiếng nói được mô tả như sau :

Sau khi thu âm ở client, client sẽ tự động ngắt thu âm và mở kết nối socket với server, số mẫu tiếng nói ban đầu được và được lưu vào một mảng short và được gửi lên server.

Server khi chạy luôn mở socket để lắng nghe và kết nối khi có một client muốn kết nối. Với mỗi một client đến server tạo một thread riêng để đáp ứng với client đó :

* Nhận mảng short client gửi lên
* Chuyển mảng short thành mảng int
* VAD mảng int để bỏ đi khoảng lặng và những xung âm thanh ngắn
* Ghi thành file âm thanh định dạng wav làm đầu vào cho khối nhận dạng
* Nhận dạng file âm thanh
* Trả ra kết quả là một string
* Gửi lại string xuống client
* Đóng kết nối với client tương ứng và kết thúc thread

Quá trình nhận dạng qua một số bước chính: sau khi qua bộ VAD để bỏ các khoảng lặng, các file âm thanh sẽ được trích chọn đặc trưng MFCC để tạo ra các hệ số đặc trưng như đã mô tả ở các phần trước, rồi đưa đến bộ giải mã (decoder). Các hệ số đặc trưng đưa vào bộ giải mã để ánh xạ đến mô hình ngôn ngữ (file dict và DMP) và mô hình âm học (folder an4.ci\_cont) đã thu được sau quá trình huấn luyện. Bộ giải mã sẽ hoạt động và đưa ra kết quả nhận dạng.

## 2.4. Module hiểu ngữ nghĩa.

Trong giai đoạn đầu của dự án, xây dựng chương trình Viva phiên bản demo, khối hiểu ngữ nghĩa được xây dựng đơn giản để hiểu câu lệnh người dùng yêu cầu, và tập trung vào các chức năng: tin nhắn, truy vấn thông tin. Khối hiểu được xây dựng trên mô hình một automat hữu hạn, đơn định, câu đầu vào được tách từ, các từ được gán nhãn theo bộ nhãn quy định trước, danh sách từ đã gán nhãn được đưa vào automat, nếu được đoán nhận, automat sẽ trả về mã khối chức năng, mã chức năng và các tham số.

Phần cơ sở lý thuyết sau đây trình bày tóm lược về phương pháp tách từ và khái niệm automat, automat hữu hạn đơn định sử dụng trong khối hiểu.

### 2.4.1 Cơ sở lý thuyết.

#### 2.4.1.1 Tách từ tiếng Việt

Tách từ là một quá trình xử lý nhằm mục đích xác định ranh giới của các từ trong câu văn, cũng có thể hiểu đơn giản rằng tách từ là quá trình xác định các từ đơn, từ ghép… có trong câu. Đối với xử lý ngôn ngữ tự nhiên, để có thể xác định cấu trúc ngữ pháp của câu, xác định từ loại của một từ trong câu, yêu cầu nhất thiết đặt ra là phải xác định được đâu là từ trong câu. Vấn đề này tưởng chừng đơn giản với con người nhưng đối với máy tính, đây là bài toán rất khó giải quyết.

Chính vì lý do đó tách từ được xem là bước xử lý quan trọng đối với các hệ thống Xử Lý Ngôn Ngữ Tự Nhiên, đặc biệt là đối với các ngôn ngữ thuộc vùng Đông Á theo loại hình ngôn ngữ đơn lập. Ví dụ: tiếng Trung Quốc, tiếng Nhật, tiếng Thái, và tiếng Việt. Với các ngôn ngữ thuộc loại hình này, ranh giới từ không chỉ đơn giản là những khoảng trắng như trong các ngôn ngữ thuộc loại hình hòa kết như tiếng Anh…, mà có sự liên hệ chặt chẽ giữa các tiếng với nhau, một từ có thể cấu tạo bởi một hoặc nhiều tiếng. Vì vậy đối với các ngôn ngữ thuộc vùng Đông Á, vấn đề của bài toán tách từ là khử được sự nhập nhằng trong ranh giới từ.  
3.2.1 Các hướng tiếp cận giải quyết

Có 3 phương pháp tiếp cận chính trong bài toán tách từ :

* Tiếp cận dựa vào từ điển cố định.
* Tiếp cận dựa vào thống kê thuần túy
* Tiếp cận dựa trên cả hai phương pháp trên.

Các phương pháp được sử dụng:

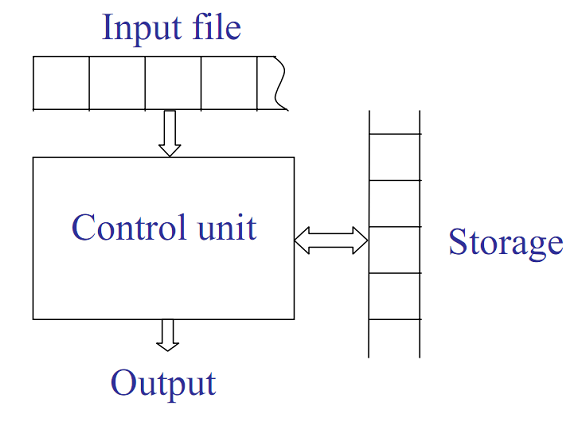
* So khớp từ dài nhất (Longest Matching)
* So khớp cực đại (Maximum Matching)
* Mô hình Markov ẩn (Hidden Markov Models- HMM)
* Học dựa trên sự cải biến (Transformation-based Learning – TBL)
* Chuyển đổi trạng thái trọng số hữu hạn (Weighted Finite State Transducer – WFST)
* Độ hỗn loạn cực đại (Maximum Entropy – ME)
* Máy học sử dụng vectơ hỗ trợ (Support Vector Machines)
* Trường xác xuất có điều kiện (CRFs)

Tài nguyên hỗ trợ:

* Từ điển tiếng Việt
* Ngữ liệu tiếng Việt được tách từ hỗ trợ quá trình huấn luyện

#### 2.4.1.1 Automat

Automat là máy dịch tự động. Đây là một mô hình trừu tượng của một automat bao gồm các thành phần sau :



Hình 2.10 Mô hình của một automat

* Đầu vào (input file) : là nơi mà các chuỗi nhập (input string) được ghi lên và được automat đọc nhưng không thay đổi được nội dung. Nó được chia thành các ô, mỗi ô giữ một ký hiệu.
* Cơ cấu nhập(input mechanism): là bộ phận có thể đọc input file từ trái sang phải, một ký tự tại một thời điểm. Nó cũng có thể dò tìm được điểm kết thúc của chuỗi nhập
* Bộ nhớ tạm(temporary storage) : là thiết bị gồm một số không giới hạn ô nhớ(cell), mỗi ô có thể giữ 1 ký hiệu từ một bảng chữ cái (không cần giống bảng chữ cái nhập). Automat có thể đọc và thay đổi nội dung của các ô nhớ này.
* Đơn vị điều kiển (Control Unit): mỗi automat có một đơn vị điều khiển, cái mà có thể ở trong một trạng thái bất kỳ trong một số hữu hạn các trạng thái nội, và có thể chuyển đổi trạng thái trong một kiểu được định nghĩa sẵn có nào đó.

**Một số khái niệm:**

* Trạng thái nội: là một trạng thái của đơn vị điều khiển mà nó có thể đạt được
* Trạng thái kế: Là một trạng thái nội của đơn vị điều khiển mà nó sẽ đạt tới ở thời điểm kế tiếp
* Hàm chuyển trạng thái: là hàm đưa ra trạng thái kế của automat dựa trên trạng thái hiện hang, kí hiệu hiện thành được quét và thông tin hiện hành trong bộ nhớ tạm.

**Hoạt động của automat:**

* Tại một thời điểm bất kỳ đã cho, đơn vị điều khiển đang ở trong một trạng thái nội (internal state) nào đó, và cơ cấu nhập là đang quét (scanning) một ký hiệu củ thể nào đó trên Input file.
* Trạng thái nội của đơn vị điều khiển tại thời điểm kế tiếp được xác định bởi trạng thái kế (next state) hay bởi hàm chuyển trạng thái.
* Trong suốt quá trình chuyển trạng thái từ khoảng thời gian này tới khoảng thời gian kế, kết quả có thể được sinh ra và lưu trong bộ nhớ lưu trữ có thể thay đổi.

#### 2.4.1.2 Automat hữu hạn đơn định – DFA (Deterministic Finite Automata)

Otomat hữu hạn đơn định gồm một tập hữu hạn các trạng thái và một tập các phép chuyển trạng thái tới trạng thái khác trên các ký hiệu nhập (input symbols) được chọn từ một bộ chữ cái ∑ nào đó. Mỗi ký hiệu nhập có đúng một phép chuyển khỏi mỗi trạng thái. Nó cũng có thể chuyển về chính nó. Có trạng thái bắt đầu và một số trạng thái được thiết kế như trạng thái kết thúc hay trạng thái chấp nhận.

Otomat hữu hạn đơn định gồm 5 thành phần:

* Q : là tập hữu hạn các trạng thái
* ∑ : là bộ chữ cái nhập hữu hạn
* δ : Là hàm chuyển ánh xạ từ Q x ∑ 🡪 Q, tức là δ(q,a) là một trạng thái được cho phép bởi phép chuyển từ trạng thái q trên ký hiệu nhập a
* q0 là trạng thái bắt đầu
* F là tập các trạng thái kết thúc

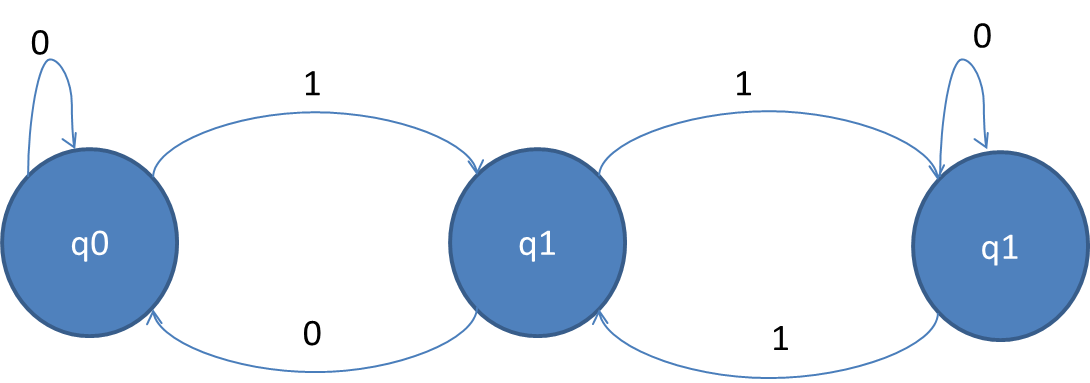
Hoạt động của một DFA:

* Tại thời điểm khởi đầu, nó được giả thiết ở trong **trạng thái khởi đầu q0** với cơ cấu nhập đang ở ký hiệu đầu tiên bên trái của chuỗi nhập
* Mỗi lần di chuyển, cơ cấu nhập tiến về phía phải 1 ký hiệu và lấy ra
* Khi gặp ký hiệu kết thúc, chuỗi được chấp nhận nếu automat đang ở vào 1 trong **các trạng thái chấp nhận được** của nó. Ngược lại thì chuỗi bị từ chối.

Để biểu diễn trực quan cho DFA ta có thể dùng đồ thị chuyển trạng thái:

Ví dụ đồ thị chuyển trạng thái của Một DFA : M = (Q, ∑, δ, q0, F)

Q = {q0,q1,q2}, ∑ = {0,1}, F{q1}



Hình 2.11 Đồ thị chuyển trạng thái của một automat hữu hạn đơn định

### 2.4.2 Triển khai module hiểu ngữ nghĩa trong Viva.

* Bước 1 : Tách từ

Sử dụng công cụ có sẵn là bộ tách từ  *vnTokenizer 4.1.1c (04/08/2010)* của dự án VLSP.[10]

* Bước 2 : Gán nhãn keyword cho từ

Xác định loại liên quan của từ được tách ra.

Ví dụ: nhắn, nhắn tin 🡪 tomessage ; thời tiết, trời 🡪toweather.

Đây chính là xác định các “ký tự nhập” của automat.

* Bước 3 : Thực hiện dịch automat

Kết quả chạy theo là các tham số Function xác định được.

* Bước 4: Xuất kết quả ra, trả về hàm điều khiển ở hệ thống.

Trong hệ thống hiện tại, gửi về client để xử lý.

## 2.5. Kiến trúc VIVA trên android.

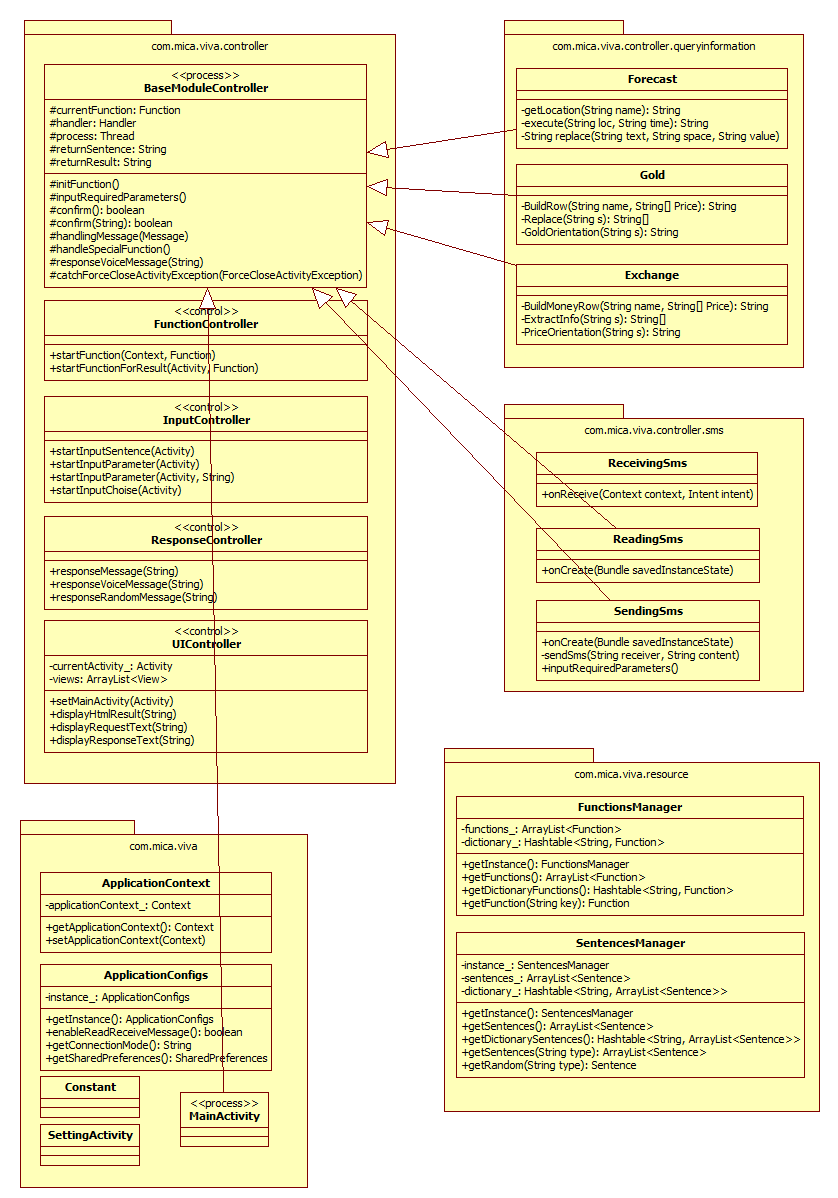
Ứng dụng Viva được thiết kế và xây dựng trên android, một hệ điều hành mã nguồn mở mạnh mẽ dành cho thiết bị di động. Hệ điều hành android ngày càng phát triển mạnh mẽ và số lượng thiết bị cài đặt hệ điều hành này cũng không ngừng tăng cao.

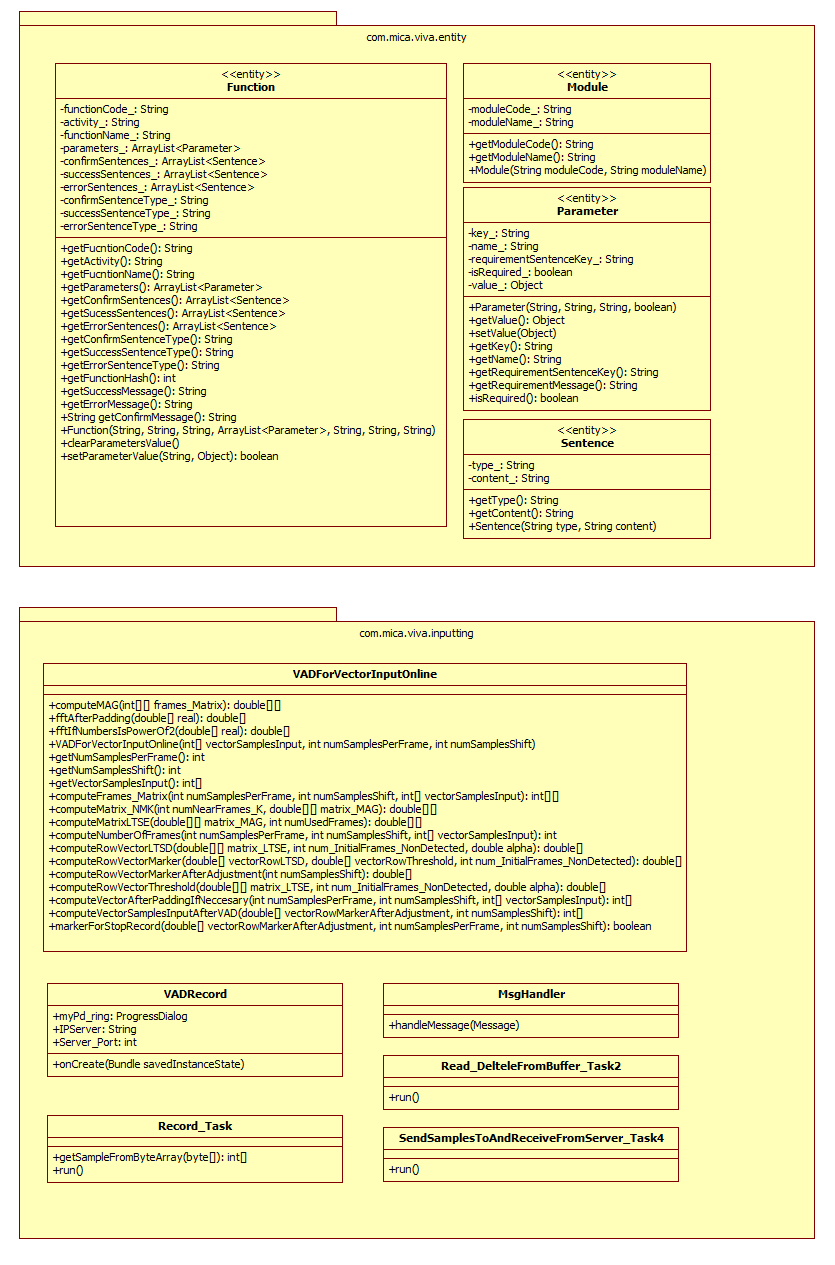
### 2.5.1. Giới thiệu kiến trúc Android.

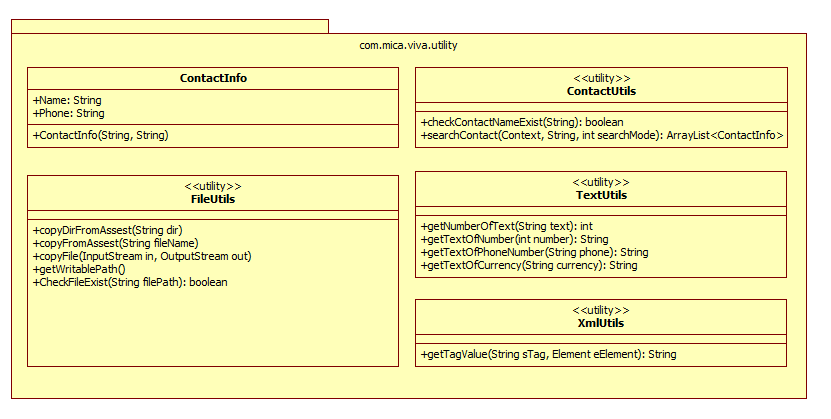
### 2.5.2. Kiến trúc ứng dụng Viva.

#### 2.5.2.1. Biểu đồ lớp

Viva được phân tích và thiết kế hướng đối tượng. Kiến trúc Viva gồm các package và các lớp được thể hiện trong hình vẽ dưới đây.







Hình 2.12 Biểu đồ lớp của chương trình Viva

#### 2.5.2.2. Mô tả lớp

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller | |
| Class | Đặc tả | |
| BaseModuleController | Lớp cha | android.app.Activity |
| 1. Là lớp cơ sở của các lớp thực thi chức năng (gửi tin, đọc tin nhắn, trả lời thời tiết, giá vàng…) 2. Có thuộc tính **protected** Thread process là luồng thực thi chức năng. Các lớp con sẽ khai báo kịch bản, tiến trình thực hiện trong hàm **public** **void** run() của process | |
| UIController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển giao diện chính của ứng dụng. * Cung cấp các phương thức public để thao tác với giao diện chính: * **void** displayRequestText(String requestText) hiển thị câu nói của người dùng nhập vào. * **void** displayResponseText(String responseText) hiển thị câu trả lời của Viva * **void** displayHtmlResult(String htmlContent) hiển thị thông tin dạng html | |
| InputController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc nhập câu nói. * Các phương thức public để thực hiện quá trình nhập câu nói: * **void** startInputSentence(Activity caller) bắt đầu nhập vào câu lệnh * **void** startInputParameter(Activity caller) bắt đầu nhập vào tham số (thí dụ: tham số tên ngươi nhận trong chức năng gửi tin nhắn) * **void** startInputChoise(Activity caller) bắt đầu nhập vào lựa chọn có hay không? | |
| ResponseController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc trả lời bằng tiếng nói của Viva. * Cung cấp các phương thức public thực hiện việc đọc và hiển thị câu trả lời: * **void** responseVoiceMessage(String message) đọc câu trả lời. * **void** responseMessage(String message) đọc câu trả lời đồng thời hiện câu trả lời ra màn hình. * **void** responseRandomMessage(String type) tìm câu ngẫu nhiên theo kiểu định trước trong cơ sở dữ liệu và trả lời người dùng. | |
| FunctionController | Lớp cha | java.lang.Object |
| * Là lớp tiếp nhận dữ liệu trả về từ khối hiểu và kích hoạt chức năng tương ứng. * Các phương thức public quan trọng: * Function getFunctionFromData(String st) nhận vào đoạn text có định dạng trả về từ khối hiểu và đưa ra chức năng tương ứng cùng các tham số nếu có. * **void** startFunction(Context context, Function function) Kích hoạt một chức năng. * **void** startFunctionForResult(Activity activity,Function function) kích hoạt một chức năng và trả về kết quả sau khi thực hiện xong. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.entity | |
| Class | Đặc tả | |
| Module | Lớp cha | java.lang.Object |
| 1. Lưu thông tin module chức năng | |
| Function | Lớp cha | java.lang.Object |
| * Lưu thông tin từng chức năng (gửi tin nhắn, đọc tin nhắn, trả lời thời tiết…) | |
| Parameter | Lớp cha | java.lang.Object |
| * Lưu thông tin tham số của chức năng (tham số nội dung trong chức năng gửi tin nhắn) | |
| Sentence | Lớp cha | java.lang.Object |
| * Lưu thông tin câu trả lời cho Viva | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.resource | |
| Class | Đặc tả | |
| FunctionsManager | Lớp cha | java.lang.Object |
| 1. Đọc dữ liệu các chức năng từ file data và lưu vào danh sách các đối tượng. 2. ArrayList<Function> getFunctions() Lấy danh sách các chức năng 3. Function getFunction(String key) Lấy đối tượng chức năng - Function theo mã. | |
| SentencesManager | Lớp cha | java.lang.Object |
| * Đọc dữ liệu các câu nói từ file data, lưu vào danh sách câu nói. * ArrayList<Sentence> getSentences() Lấy danh sách các câu nói * Sentence getRandom(String type) Lấy ra một câu ngẫu nhiên theo kiểu truyền vào | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller.sms | |
| Class | Đặc tả | |
| SendingSms | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng gửi tin nhắn | |
| ReadingSms | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng đọc tin nhắn | |
| ReceivingSms | Lớp cha | android.content.BroadcastReceiver |
| * Thực hiện bắt bản tin Broadcast khi có tin nhắn đến, đọc các thông tin và kích hoạt chức năng đọc tin nhắn. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller.queryinformation | |
| Class | Đặc tả | |
| ForeCast | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin thời tiết. * Thông tin thời tiết được lấy từ trang *wunderground.com* qua API Webservice. | |
| Gold | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin giá vàng. * Thông tin giá vàng được lấy từ trang *24h.com.vn* | |
| Exchange | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin tỉ giá. * Thông tin tỉ giá được lấy từ trang *24h.com.vn* | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva | |
| Class | Đặc tả | |
| ApplicationConfigs | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Lưu và hỗ trợ truy cập đến các tham số setting của chương trình. | |
| ApplicationContext | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Lưu ngữ cảnh hoạt động của chương trình. | |
| Constant | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Chứa các trường giá trị constant của trương trình | |
| MainActivity | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Activity chính, mang giao diện tương tác của ứng dụng với người dùng | |
| SettingActivity | Lớp cha | android.preference. PreferenceActivity |
| * Activity cho phép người dùng tùy chọn các trường setting của chương trình. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.inputting | |
| Class | Đặc tả | |
| VADRecord | Lớp cha | android.app.Activity |
| * Là lớp điều khiển việc thu âm, tự ngắt thu âm, gửi mảng mẫu tiếng nói lên server nhận dạng, nhận kết quả text từ server và hiển thị lên màn hình * Có thuộc tính: Thread thread1, Thread thread2, Thread thread3 * Thread 1: thu âm, đưa vào buffer và vào hàng đợi * Thread 2: lấy ra từ buffer lần lượt 8000 mẫu để áp dụng thuật toán VAD và đánh dấu có tự ngắt hay không việc thu âm ở Thread 1 * Thread 3: được tạo ra sau khi ngắt thu âm, lấy từ hàng đợi và gửi lên server nhận dạng và nhận kết quả text | |
| MsHandler | Lớp cha | android.os.Handler |
| * Phục vụ việc trao đổi giữa thread chính: UI và thread 2, thread 3 và UI. Khi ngắt thu âm, thread 2 gửi tín hiệu cho UI, UI sinh ra thread 3 để gửi lên server nhận dạng. Khi nhận được text từ server thread 3 báo tín hiệu cho UI để hiển thị text trên lên màn hình. | |
| Record\_Task | Thực thi | java.lang.Runnable |
| * Phục vụ thu âm, thực hiện bởi thread 1 | |
| Read\_DelteleFromBuffer\_Task2 | Thực thi | java.lang.Runnable |
| * Phục vụ việc tự ngắt thu âm, thực hiện bởi thread 2 | |
| SendSamplesToAndReceive FromServer | Thực thi | java.lang.Runnable |
| * Phục vụ việc gửi lên server và nhận text từ server, thực hiện bởi thread 3 | |
| VADForVectorInputOnline | Lớp cha | java.lang.Object |
| * Là lớp cung cấp các methode để áp dụng thuật toán phát hiện tiếng nói LTSD với đầu vào là một vector các mẫu tiếng nói kiểu int * Phục vụ việc: phân khung, phát hiện tiếng nói hay không theo từng khung, thiết lập thời gian ngắt thu âm, đánh dấu việc ngắt thu âm * Có các methode: * int getNumSamplesPerFrame() : trả về số mẫu trên 1 khung sau khi phân khung * int getNumSamplesShift() : trả về khoảng cách 2 khung liên tiếp ( tính theo số mẫu ) * int[] getVectorSamplesInput(): trả về vector các mẫu tiếng nói ban đầu * int[][] computeFrames\_Matrix(int numSamplesPerFrame, int numSamplesShift, int[] vectorSamplesInput) : trả về mảng 2 chiều sau khi phân | |

|  |  |
| --- | --- |
| Package | com.mica.viva.utility |
| * Package chứa các lớp utility với các hàm static để hỗ trợ thao tác như xử lý text (TextUtils), đọc danh bạ (ContactUtils), đọc ghi file (FileUtils)… | |

# KHỐI CHỨC NĂNG TIN NHẮN

## 3.1 Chức năng gửi tin nhắn.

### 3.1.1 Mô tả.

Chức năng gửi tin nhắn cho phép người dùng gửi tin nhắn với hai tham số: tên hoặc số điện thoại người nhận và nội dung tin nhắn.

Nếu trong câu lệnh người dùng chưa có hai thông tin này, Viva sẽ yêu cầu người dùng nhập vào thông tin còn thiếu.

Trước khi gửi tin nhắn, Viva yêu cầu người dùng xác nhận có muốn gửi tin nhắn hay không, nếu xác nhận có, Viva thực hiện gửi tin nhắn và thông báo khi gửi thành công.

Khi người yêu cầu gửi tin nhắn cho tên một người trong danh bạ, Viva sẽ tìm kiếm trong danh bạ theo tên này, các trường hợp xảy ra:

* Không có tên người nhận trong danh bạ: Viva thông báo và yêu cầu nhập lại.
* Có 1 liên hệ tương ứng: gửi tin nhắn cho người này.
* Có nhiều hơn 1 liên hệ tương ứng: Viva yêu cầu người dùng chọn theo số thứ tự “một” / “hai” / “ba”…

### 3.1.2 Biểu đồ hoạt động.



Hình 3.1 Biểu đồ hoạt động của chức năng gửi tin nhắn

### 3.1.3 Biểu đồ tuần tự.





Hình 3.2 Biểu đồ tuần tự của chức năng gửi tin nhắn

## 3.2 Chức năng đọc tin nhắn đến.

### 3.2.1 Mô tả.

Android là hệ điều hành mở cho phép các ứng dụng bắt các bản tin Broadcast các sự kiện trong đó có sự kiện có tin nhắn đến. Lớp ReceivingSms trong module tin nhắn của Viva làm nhiệm vụ bắt bản tin Broadcast này và kích hoạt chức năng đọc tin nhắn mỗi khi có tin nhắn đến.

Chức năng đọc tin nhắn bao gồm 3 thao tác sau:

* Thông báo tin nhắn mới: Viva thông báo cho người dùng “bạn có tin nhắn từ số điện thoại…”. Nếu số điện thoại người gửi được lưu trong danh bạ, Viva sẽ tìm trong danh bạ để lấy ra tên liên hệ tương ứng và thông báo cho người dùng.
* Confirm và đọc nội dung tin nhắn: Viva hỏi người dùng có muốn nghe nội dung tin nhắn không, nếu có, Viva thực hiện đọc nội dung tin nhắn.
* Confirm trả lời tin nhắn: Viva hỏi người dùng có muốn trả lời tin nhắn không, nếu có, Viva chuyển sang chức năng gửi tin nhắn đã trình bày trên.

### 3.2.2 Biểu đồ hoạt động.



Hình 3.3 Biểu đồ hoạt động của chức năng đọc tin nhắn

### 3.2.3 Biểu đồ tuần tự.



Hình 3.4 Biểu đồ tuần tự của chức năng đọc tin nhắn