

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

— * —



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

ỨNG DỤNG TẠO TIẾNG NÓI TIẾNG VIỆT TỪ VĂN
BẢN TRÊN KIT MINI2440

Giáo viên hướng dẫn:

ThS. Dư Thanh Bình

Sinh viên thực hiện:

Phí Tùng Lâm 20071670 KTMT - K52

Nguyễn Trung Dũng 20070583 KTMT - K52

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: **Phí Tùng Lâm**

MSSV:20071670

Điện thoại liên lạc : 0975253758

Email: king_lam_2110@yahoo.com

Lớp: KTMTK52

Hệ đào tạo: Đại học

Họ và tên sinh viên: Nguyễn Trung Dũng

MSSV:20070583

Điện thoại liên lạc : 01682699970

Email: nguyendung_9002@yahoo.com

Lớp: KTMTK52

Hê đào tạo: Đại học

Thời gian làm ĐATN: Từ ngày 21/02 /2012 đến 01/06 /2012

2. Mục đích nội dung của ĐATN

Xây dựng ứng dụng chuyển chữ viết thành tiếng nói trên KIT phát triển Mini2440.

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu lý thuyết
- Tìm hiểu lý thuyết về hệ thống nhúng
- Tìm hiểu về Kit phát triển Mini2440
- Tìm hiểu về hệ điều hành Android
- Phân tích yêu cầu của đề tài
- Thiết kế và cài đặt ứng dụng
- Kiểm thử

4. Lời cam đoan của sinh viên:

Chúng tôi - Phí Tùng Lâm - NguyễnTrung Dũng - cam kết ĐATN là công trình nghiên cứu của bản thân chúng tôi dưới sự hướng dẫn của ThS. Dư Thanh Bình.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày 01 tháng 06 năm 2012

Tác giả ĐATN

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:.....

.....
.....
.....
.....

Hà Nội, ngày tháng năm
Giáo viên hướng dẫn

Ths. Dư Thanh Bình

LỜI NÓI ĐẦU

Những năm gần đây, sự phát triển vượt bậc của khoa học kỹ thuật nói chung và công nghệ thông tin nói riêng đã tác động tới mọi mặt của đời sống xã hội, kinh tế, chính trị... Sự hiện diện của những ứng dụng tin học đã góp phần giải quyết những khó khăn do việc thiếu thông tin cũng như nâng cao năng suất lao động, góp phần tạo ra một lượng lớn của cải vật chất và thúc đẩy sự phát triển của xã hội.

Gần đây, hệ điều hành Android đang dành được nhiều sự quan tâm nghiên cứu và thử nghiệm. Trong đó thiết kế hệ thống nhúng cũng là một lĩnh vực rộng lớn mà Android có thể khai thác.

Trong quá trình thực tập tại viện CNTT em được giới thiệu một đề tài rất thú vị là hệ thống TTS, hệ thống có rất nhiều ứng dụng như đọc báo online, đọc tiếng việt cho người nước ngoài,... hệ thống này rất phát triển với các ngôn ngữ khác nhưng chưa thực sự phát triển với tiếng Việt. Từ thực tế đó em nảy ra ý tưởng viết một hệ thống TTS tiếng việt trên hệ thống nhúng sử dụng hệ điều hành Android.

Với đề tài “**ứng dụng tạo tiếng nói tiếng việt từ văn bản trên kit mini2440**” em cũng đã đạt được một số kết quả nhất định. Đó sẽ là hành trang giúp em rất nhiều trong thời gian học tập và làm việc sau này.

Để có thể hoàn thành được đồ án là nhờ sự giúp đỡ to lớn của **các thầy cô giáo trong trường Đại học Bách Khoa Hà Nội** nói chung và **các thầy cô trong khoa Công nghệ Thông tin, bộ môn Kỹ Thuật Máy Tính** nói riêng. Các thầy cô đã tận tình giảng dạy, truyền đạt cho em những kiến thức, những kinh nghiệm quý báu trong suốt 5 năm học tập và rèn luyện tại trường. Xin được gửi tới các thầy, các cô lời cảm ơn chân thành nhất đặc biệt đến **thầy Đur Thanh Bình - Giảng viên bộ môn Kỹ Thuật Máy Tính, khoa Công nghệ thông tin, trường Đại học Bách Khoa Hà Nội** đã hết lòng giúp đỡ, hướng dẫn và chỉ bảo tận tình trong quá trình em làm đồ án tốt nghiệp.

Em cũng xin cảm ơn sự giúp đỡ tận tình của anh **Vũ Tất Thắng** và các thành viên của phòng tiếng nói và hình ảnh viện CNTT trong quá trình xây dựng phần mềm.

Cuối cùng, em xin được gửi lời cảm ơn chân thành tới **gia đình, bạn bè** đã động viên, chăm sóc, đóng góp ý kiến và giúp đỡ em trong quá trình học tập, nghiên cứu và hoàn thành đồ án tốt nghiệp.

Hà Nội, tháng 5 năm 2012

Sinh viên

Mục lục

CHƯƠNG I: MỞ ĐẦU	8
1.1. Nhiệm vụ của đồ án.....	8
1.2. Môi trường thực hiện đồ án	8
1.3. Bố cục đồ án.....	8
CHƯƠNG II: ĐẶT VẤN ĐỀ VÀ GIẢI PHÁP	9
2.1. Nhu cầu thực tế	9
2.2. Phạm vi và nhiệm vụ.....	10
CHƯƠNG III: NỀN TẢNG CÓ SẴN	11
3.1. Hệ thống nhúng	11
3.2. Giới thiệu về Android	12
3.2.1. Lịch sử Android	12
3.2.2. Tính năng mở của hệ điều hành Android	13
3.2.3. DEIVING và máy ảo DALVIK	13
3.2.4. Kiến trúc Android.....	14
3.2.5. Các thành phần của một Project Android.....	16
3.2.6. Chu kì của một ứng dụng Android.....	16
3.2.7. Các thành phần giao diện của Andorid	19
3.3. Giới thiệu về KIT Mini2440	22
3.3.1. Cấu hình kỹ thuật	22
3.3.2. Các chú ý về cổng giao tiếp	23
3.3.3. Nguồn hỗ trợ.....	25
3.3.4. Mạch khởi động lại hệ thống (System Reset).....	26
3.3.5. LEDs.....	26
3.3.6. Nút Bấm	26
3.3.7. A/D input test.....	27
3.3.8. Speaker.....	27
3.3.9. Serial Port	28
3.3.10.Nối tiếp USB	28
3.3.11.LCD interface	29
3.3.12.EEPROM	29
3.3.13.Network Interface	30
CHƯƠNG IV: XÂY DỰNG ỨNG DỤNG	31
4.1. Mô hình hệ thống	31

4.2.	Cài đặt trên KIT.....	32
4.2.1.	Sửa nhân mini2440 cho màn hình X35 Sony.....	32
4.2.2.	Dịch nhân android	34
4.2.3.	Dịch file system.....	36
4.2.4.	Cài đặt Android cho mini2440	37
4.3.	Giao diện chương trình.....	38
4.3.1.	Thiết kế giao diện chương trình.....	38
4.3.2.	Giao diện chi tiết	40
4.4.	Làm bộ gõ tiếng việt với Android	43
4.4.1.	Cấu tạo chung của tiếng việt	43
4.4.2.	Các kiểu gõ tiếng việt.....	45
4.4.3.	Thuật toán để lập trình bộ gõ	47
4.5.	Giao tiếp với server Isolar	51
4.5.1.	Gửi yêu cầu đến server	51
4.5.2.	Stream file âm thanh.....	52
4.6.	Chia sẻ mạng của Ubuntu qua dây Ethernet.....	53
4.6.1.	Mô hình hoạt động.....	54
4.6.2.	Cấu hình NAT	54
4.6.3.	Cấu hình routing	55
4.6.4.	Cấu hình bên máy nhận(mini2440)	55
CHƯƠNG V: KẾT LUẬN.....		57
DANH MỤC TÀI LIỆU THAM KHẢO.....		58

Mục lục các hình

Hình 1 - Kiến trúc Android	14
Hình 2 - Activity Stack	17
Hình 3 - chu kì sống của một Activity	18
Hình 4 - Sử dụng Linear Layout	20
Hình 5 - bố trí các widget trong Frame Layout	20
Hình 6 - bố trí các widget trong Table Layout.....	21
Hình 7 - Kit Mini2440.....	23
Hình 8 - Kết nối SDRAM	24
Hình 9 - kết nối NAND, NOR Flash.....	25
Hình 10 - sơ đồ nguồn.....	25
Hình 11 - mạch reset.....	26
Hình 12 - Vị trí và kết nối nút bấm	27
Hình 13 - ADC input	27
Hình 14 - Kết nối loa ngoài	27
Hình 15 - Kết nối cổng nối tiếp.....	28
Hình 16 - Nối tiếp USB	28
Hình 17 - Giao tiếp LCD	29
Hình 18 – EEPROM.....	29
Hình 19 - Giao diện mạng	30
Hình 20 - Mô hình hệ thống	31
Hình 21 - Biên dịch chéo gcc	34
Hình 22 - menuconfig.....	35
Hình 23 - tạo zImage.....	35
Hình 24 - dịch root file.....	36
Hình 25 - giao diện dnw.....	37
Hình 26 - kết quả dịch nhân.....	37
Hình 27 - Biểu đồ giao diện	38
Hình 28 - mô hình phần nhập văn bản.....	39
Hình 29 - mô hình phần chạy tệp âm thanh	40
Hình 30 - giao diện chung	40
Hình 31 - thanh điều khiển nhập text.....	41
Hình 32 - màn hình nhập văn bản.....	41
Hình 33 - bàn phím ảo	42
Hình 34 - Tuỳ chỉnh của chương trình.....	42
Hình 35 – danh sách tập tin âm thanh	43
Hình 36 - thanh điều khiển phát âm thanh.....	43
Hình 37 - lưu đồ xử lý tiếng việt	51
Hình 38 - Mô hình kết nối xử lý tiếng nói	52
Hình 39 - Mô hình hệ thống chương trình	53
Hình 40 - chia sẻ mạng qua dây ethernet	54
Hình 41 - kết quả chia sẻ mạng.....	56

CHƯƠNG I

MỞ ĐẦU

1.1.Nhiệm vụ của đồ án

Đồ án nhằm mục đích xây dựng một ứng dụng chuyển văn bản chữ viết tiếng việt thành tiếng nói (Text to Speech) trên một thiết bị nhúng là KIT phát triển Mini2440.

1.2.Môi trường thực hiện đồ án

Đồ án thực hiện tại Viện công nghệ thông tin Việt Nam - số 18 Hoàng Quốc Việt - Hà Nội.

1.3.Bố cục đồ án

Đồ án gồm năm phần chính:

- Phần một là cái nhìn tổng quan về đồ án.
- Phần hai là vấn đề và cách giải quyết vấn đề: bài toán TTS trong thực tế là gì, các vấn đề gặp phải , và đề ra hướng giải quyết.
- Phần hai là các nền tảng sẵn có.
- Phần bốn là quá trình thực hiện và kết quả, phần đã thực hiện được và chưa thực hiện được.
- Phần năm là các kết luận, ưu nhược điểm và hướng phát triển của vấn đề.

CHƯƠNG II

ĐẶT VẤN ĐỀ VÀ GIẢI PHÁP

2.1.Nhu cầu thực tế

Hiện nay trên thế giới, vấn đề sử dụng phần mềm về âm thanh không còn là quá mới mẻ, tuy nhiên do đặc thù các ngôn ngữ là khác nhau nên không có một mô hình chung cho việc xử lý cho tất cả các ngôn ngữ. Trên thực tế chúng ta chưa có mô hình xử lý tiếng nói tiếng việt ở mức chung cho tất cả cùng sử dụng dù sử dụng tiếng nói tiếng việt là một nhu cầu không hề nhỏ. Tương tự như Tiếng Anh, viện CNTT đã xây dựng được phần mềm chuyên biệt về TTS đối với tiếng việt, hiện tại đã có một server TTS đang hoạt động. Vậy sử dụng hệ thống đó như thế nào.

Chúng em lựa chọn xây dựng một ứng dụng như một hệ thống ứng dụng thử nghiệm cho hệ thống TTS trên, mở rộng cho các ứng dụng TTS sau này. Hệ thống cho phép phát tiếng nói từ đoạn văn bản tiếng việt nhập vào. Hệ thống này xuất phát từ một nhu cầu thực tế là cần tạo ra một thiết bị cầm tay có khả năng phát ra tiếng nói dựa trên đoạn văn bản nhập vào, có tác dụng trong rất nhiều lĩnh vực:

- Người nước ngoài, việt kiều có nhu cầu nghe và học tiếng việt.
- Các bệnh nhân bị chấn thương, bệnh tật ảnh hưởng đến chức năng nói có thể dễ dàng giao tiếp.
- Đọc tin nhắn, đọc báo Online trên các thiết bị di động.
- ...

Từ thực tế đó cho thấy cần xây dựng hệ thống trên một thiết bị cầm tay, có thể cài đặt phần mềm TTS lên đó, có khả năng kết nối mạng để có thể sử dụng server TTS. Có thể xây dựng hệ thống trên các điện thoại smart phone hiện đang rất phổ biến trên thị trường hoặc các thiết bị xách tay khác như máy tính bảng, laptop ,... đều sử dụng được, trong quá trình thực tập chúng em đã chọn hệ điều hành nhúng Android, và sử dụng KIT phát triển Mini2440 để làm phần cứng cơ bản cho thiết bị cầm tay.

Mặc dù xây dựng trên phần cứng của Mini2440 nhưng ứng dụng hoàn toàn có thể cài đặt trên bất cứ thiết bị nào có sử dụng hệ điều hành Android, đó là một lĩnh vực rộng lớn tại thực tế môi trường tại Việt Nam.

2.2. Phạm vi và nhiệm vụ

Các nhu cầu cần có của đề tài:

- Nghiên cứu Kit Mini2440.
- Nghiên cứu hệ điều hành Android và cách cài đặt sử dụng Android trên Mini2440.
- Cách xây dựng phần mềm trên android.
- Xây dựng ứng dụng thực tế TTS trên hệ thống trên.

Các nhiệm vụ cần thực hiện:

- Xây dựng kết nối từ KIT Mini2440 đến server TTS.
- Xây dựng hệ thống phát tiếng nói trên Mini2440.
- Xây dựng hệ thống nhập liệu tiếng việt cho Android.

CHƯƠNG III

NỀN TẢNG CỐ SẴN

3.1. Hệ thống nhúng

Hệ thống nhúng là một hệ thống hoàn chỉnh được sử dụng cho một thiết bị, bao gồm Linux kernel và các ứng dụng kèm theo (kết hợp này gọi là một **distribution**). Cụm từ “nhúng”(embedded) thường được đề cập trong nhân nhưng thật chất không có một phiên bản nhân Linux nào dành riêng cho hệ thống nhúng. Mã nguồn nhân Linux được sử dụng chung để biên dịch cho mọi thiết bị, từ các thiết bị nhúng, đến máy PC và cả các server lớn, đối với mỗi nền tảng sẽ có những hiệu chỉnh phù hợp và đặc biệt dành cho nền tảng đó.

Các hệ thống nhúng Linux có thể mua được từ các nhà sản xuất (**Monta Vista, Wind River System...**), các distribution này đã được phát triển hoàn chỉnh có thể cài đặt như cài đặt một hệ điều hành cho một máy PC thông thường, đi kèm với distribution là các công cụ phát triển (toolchain, debugger, project management software và image builder...). Tuy nhiên, việc xây dựng một hệ thống nhúng từ các phần tử rời rạc ban đầu sẽ giúp chúng ta có sự thấu hiểu sâu về hoạt động của hệ thống cũng như không phải tốn chi phí chi trả cho nhà cung cấp. Các thành phần cấu tạo nên hệ thống nhúng bao gồm boot loader, nhân Linux, các ứng dụng. Tất cả các thành phần này đều có thể tìm thấy phiên bản mã nguồn mở và chúng ta có thể tự mình chỉnh sửa, thay đổi cho phù hợp với thiết bị của mình.

Hệ thống nhúng không thể chạy trên các processor có kiến trúc nhỏ hơn 32-bit. Tuy nhiên trong thời gian gần đây, công nghệ system-on-chip (SOC) phát triển mạnh, dẫn đến việc hạ giá thành sản xuất các microprocessor, đồng thời bộ nhớ RAM và flash cũng rẻ và có dung lượng lớn hơn tạo nên thuận lợi cho việc chuyển sang phát triển hệ thống nhúng. Các hệ thống nhúng ngày nay bên cạnh các chức năng cần thiết còn có thể hỗ trợ thêm các chức năng phụ (web server, firewall, nghe nhạc...) thông qua hệ thống nhúng.

Việc sử dụng Hệ thống nhúng cho hệ thống nhúng còn giúp giảm thời gian thiết kế và phát triển, do bản thân Linux kernel được thiết kế theo module. Chúng ta có thể dễ dàng tìm được nhiều module có sẵn và hiệu quả như TCP/IP stack, X-server cho ứng dụng GUI, hoặc có thể tìm thấy driver cho thiết bị nhúng của mình đã được viết sẵn trong nhân.

Một điểm mạnh khác của hệ thống nhúng là mã nguồn mở, điều này cho phép người thiết kế can thiệp sâu hơn vào các dịch vụ và module mà hệ điều hành cung cấp. Người thiết kế có thể hiểu rõ hơn về những hàm mà họ gọi và thậm chí có thể thay đổi, tối ưu các hàm này cho thiết bị mình sử dụng. Người thiết kế còn có thể dựa vào các module driver có sẵn để tham khảo cho các driver sắp viết. Tính mã nguồn mở còn giúp code hỗ trợ bởi nhân có tính tin cậy cao, trước khi được đưa vào cây mã nguồn(kernel source tree), code này đã được thử nghiệm rất nhiều trong cộng đồng và thậm chí nếu có lỗi xảy ra cũng sẽ có phần sửa thay thế trong thời gian ngắn.

Tính sẵn sàng của hệ thống nhúng là rất cao. Ít có hệ điều hành nào hỗ trợ được nhiều nền tảng và thiết bị như Linux. Ngoài hỗ trợ phần cứng Linux còn hỗ trợ các giao thức tiêu chuẩn (wifi, bluetooth, ...).

Tất cả những điều trên cho chúng ta thấy tính năng hiệu quả của hệ thống nhúng và cũng những điểm mạnh đó làm cho xu hướng phát triển hệ thống nhúng ngày càng trở nên quan trọng, càng nhiều công ty đầu tư vào lĩnh vực này và càng nhiều người yêu thích hệ thống nhúng hơn.

3.2. Giới thiệu về Android

3.2.1. Lịch sử Android

Ban đầu Android là hệ điều hành dựa trên lõi Linux cho các thiết bị cầm tay của công ty Android Inc thiết kế. Vào 2005, Google mua lại công ty này và bắt đầu xây dựng Android Platform. Những nhà đồng sáng lập của Android chuyển sang làm việc tại Google gồm có Andy Rubin (đồng sáng lập công ty Danger), Rich Miner (đồng sáng lập công ty Wildfire Communications), Nick Sears (từng là phó chủ tịch của T-Mobile), và Chris White (trưởng nhóm thiết kế và phát triển giao diện tại WebTV). Và sau đó vào năm 2007 thuộc về liên minh các thiết bị cầm tay (Open Handset Alliance), mục tiêu của liên minh này là nhanh chóng đổi mới để đáp ứng tốt cho nhu cầu người tiêu dùng, kết quả đầu tiên của nó chính là nền tảng Android. Android được sản xuất nhằm đáp ứng nhu cầu của các nhà sản xuất, các nhà khai thác và các lập trình viên cho thiết bị cầm tay.

Phiên bản đầu tiên được ra đời vào tháng 11 năm 2007, hãng T –mobile công bố chiếc điện thoại Android đầu tiên đó là T-mobile G1, vài ngày sau Google lại công bố phiên bản Android SDK release Candidate 1.0. Trong tháng 10/2008 Google được cấp giấy phép mã nguồn mở cho Android Platform.

Khi android được phát hành thì một trong số các mục tiêu trong kiến trúc của nó là cho phép các ứng dụng có thể tương tác với nhau và có thể sử dụng lại các thành phần của ứng dụng khác. Việc tái sử dụng không chỉ được áp dụng cho các dịch vụ mà còn cho các thành phần dữ liệu và giao diện người dùng.

Cuối năm 2008, Google cho phát hành một thiết bị cầm tay có tên là Android Dev Phone 1, có thể chạy được ứng dụng Android mà không phụ thuộc vào các nhà cung cấp điện thoại di động. Mục đích của thiết bị này là cho phép các nhà phát triển có thể thực hiện các cuộc thí nghiệm trên một thiết bị thực chạy Android mà không phải kí bất cứ một bản hợp đồng nào. Cùng thời gian thì Google cho ra phiên bản vá lỗi 1.1 tuy nhiên nó vẫn chưa hỗ trợ self-key mà vẫn dùng bàn phím vật lý. Đến tháng 4/2009, SDK 1.5 đã giải quyết vấn đề cùng với một số các nâng cấp: nâng cao khả năng ghi âm, vật dụng, live folder,...

3.2.2. Tính năng mở của hệ điều hành Android

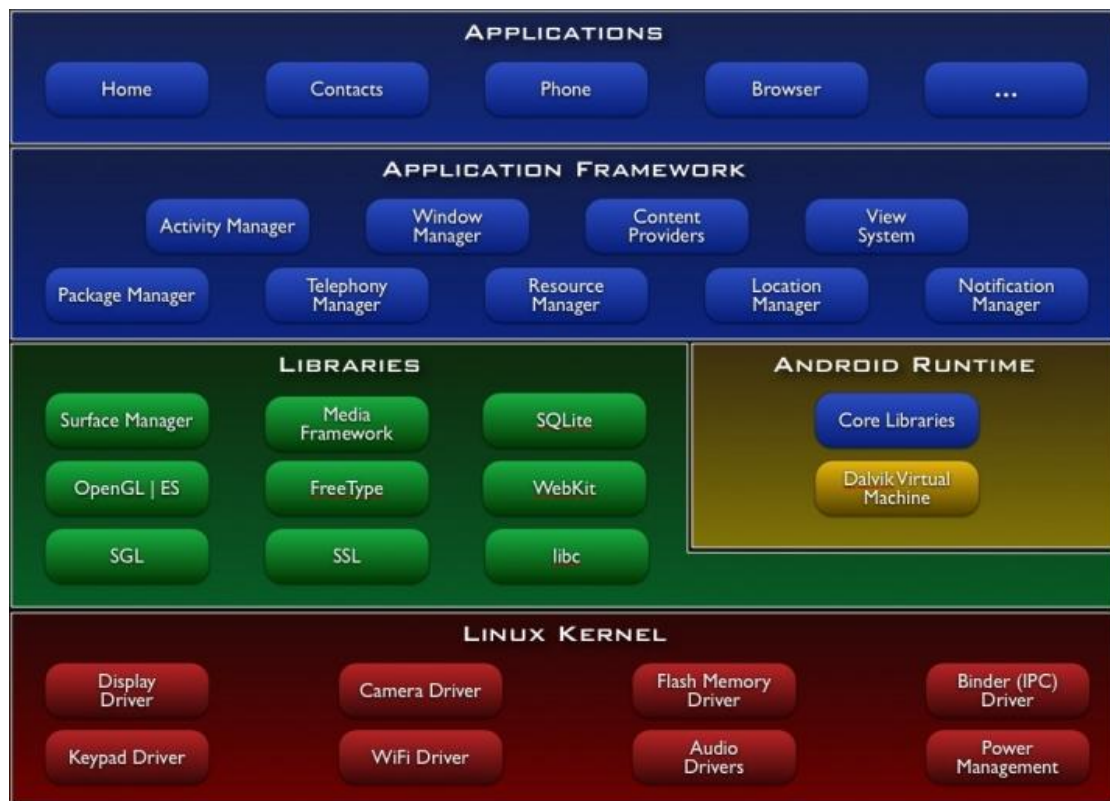
Android được xây dựng để cho phép các nhà phát triển tạo ra các ứng dụng di động hấp dẫn tận dụng tất cả tính năng của chiếc điện thoại. Nó được xây dựng thực sự mở. Ví dụ, một ứng dụng có thể kêu gọi bất kỳ chức năng lõi nào của điện thoại như thực hiện cuộc gọi, gửi tin nhắn văn bản, hoặc sử dụng máy ảnh, cho phép các nhà phát triển tạo ra phong phú hơn và nhiều hơn ứng dụng theo những yêu cầu của người dùng. Android được xây dựng trên Linux Kernel. Hơn nữa, nó sử dụng một máy ảo tùy chỉnh được thiết kế để tối ưu hóa bộ nhớ và tài nguyên phần cứng trong một môi trường di động.

3.2.3. DEVING và máy ảo DALVIK

Dalvik là máy ảo cho phép chạy các ứng dụng Java trên thiết bị Android. Nó chạy các ứng dụng được chuyển đổi thành một file thực thi Dalvik (dex). Định dạng phù hợp cho hệ thống và thường bị hạn chế về bộ nhớ và tốc độ xử lý.

Khi viết một ứng dụng cho Android sẽ được dịch sang bytecode của Java, sau đó để thực thi ứng dụng này trên Android nhà phát triển cần một công cụ là dx, công cụ này sẽ chuyển code sang một dạng là dex bytecode đóng vai trò là cơ chế ảo thực thi ứng dụng Android.

3.2.4. Kiến trúc Android



Hình 1 - Kiến trúc Android

Tầng ứng dụng

Là tầng dành để viết các ứng dụng người dùng được viết tất cả trên nền Java. Có một số các phần có sẵn như: browser, camera, phone,...

Application Framework

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển một nền tảng có khả năng xây dựng nên các ứng dụng rất phong phú và sáng tạo. Nhà phát triển được tự do vận dụng phần cứng, các thiết bị chạy nền, các dịch vụ hệ thống, ...

Cơ bản mỗi ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm:

- Một tập hợp rất nhiều các View có khả năng kế thừa lẫn nhau dùng để thiết lập phần giao diện ứng dụng: gridview, table view, ...
- Một “Content Provider” cho phép các ứng dụng có thể truy suất từ các ứng dụng khác hoặc chia sẻ giữa các ứng dụng đó.
- Một “Resource Manager” cung cấp tới các tài nguyên không phải là mã nguồn: localized strings, graphis, and layouts files.

- Một “Notification Manager” cho phép các ứng dụng hiển thị các custom alerts trong status bar.

Activity Manager được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

Library

Android sử dụng nhiều thư viện của C/C++, một số thư viện như sau:

- System C Library
- Media library
- Surface Manager
- LibWebCore
- SGL
- 3D Library
- Free type

Android Runtime

Android bao gồm một tập hợp các thư viện cơ bản mà cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Tất cả các ứng dụng Android đều chạy trên tiến trình riêng. Máy ảo Dalvik được thiết kế cho thiết bị có thể chạy nhiều máy ảo hiệu quả. Các VM Dalvik thực thi các tập tin thực thi Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là dựa trên register-based, và chạy các lớp java đã được biên dịch sang định dạng dex. Các VM Dalvik dành cho các chức năng cơ bản như luồng và quản lý bộ nhớ thấp.

Linux Kernel

Android dựa trên Linux 2.6 với các hệ thống dịch vụ cốt lõi như security, memory manager, process manager, network stack, và driver model, nó hoạt động như lớp trừu tượng hóa giữa phần cứng và phần còn lại của ngăn xếp phần mềm.

Android Emulator

Android SDK và Plugin Eclipse là một bộ Android Developer Tool dùng để viết, debug testing cho các ứng dụng. Nó được trang bị đầy đủ các phần tùy nhiên đôi chỗ bị hạn chế như USB, camera, video, nguồn giả lập,...

3.2.5. Các thành phần của một Project Android

AndroidManifest.xml

File này dùng để định nghĩa các screen sử dụng, các permission và các theme cho ứng dụng, thông tin về phiên bản SDK và main activity sẽ chạy đầu tiên. File này có 3 thành phần chính:

- **Application** chứa các thuộc tính được định nghĩa cho ứng dụng:
 - Android:icon = <drawable resource>: icon cho ứng dụng.
 - Android :name = <string> chứa tên của ứng dụng.
 - Android:theme = <drawable theme> chứa theme của ứng dụng.
- **Permission** chứa các thuộc tính chỉ định quyền truy xuất sử dụng tài nguyên của ứng dụng, ví dụ:

```
<uses-permission android:name =  
    "android.permission.READ_PHONE_STATE"/>
```

- **Version** chứa các thông tin về phiên bản thấp nhất của SDK đang được ứng dụng sử dụng:

```
<uses-sdk android:min SdkVersion=7/>
```

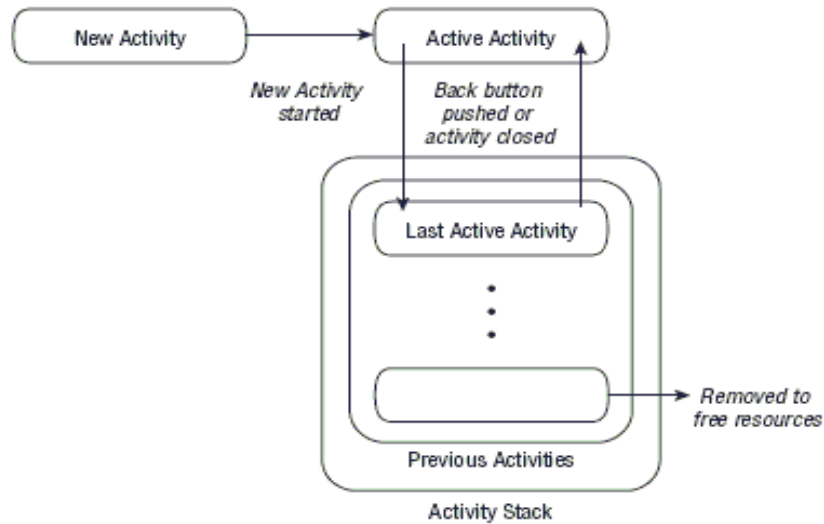
File R.java

File này quản lý các thuộc tính khai báo trong file xml của ứng dụng và tài nguyên hình ảnh, mỗi khi có thay đổi về giao diện như thêm sửa các đối tượng thì code của file này tự động thay đổi , nói chung không cần đụng chạm đến file này khi xây dựng ứng dụng.

3.2.6. Chu kì của một ứng dụng Android

Chu kì sống thành phần: Các thành phần ứng dụng có một chu kì sống từ lúc bắt đầu đến lúc kết thúc, giữa quá trình nó có thể inactive /active hoặc có thể visible/invisible trong khi đang active.

Activity stack : bên trong hệ thống Activity được quản lý như một stack, khi một activity mới được chạy nó sẽ nằm trên đỉnh của stack, activity đang chạy trước sẽ được đặt xuống dưới nó trong stack và sẽ không thấy trong suốt quá trình chạy của activity hiện tại cho đến khi người dùng ấn back thì nó sẽ được đẩy lên và trở thành activity được active.

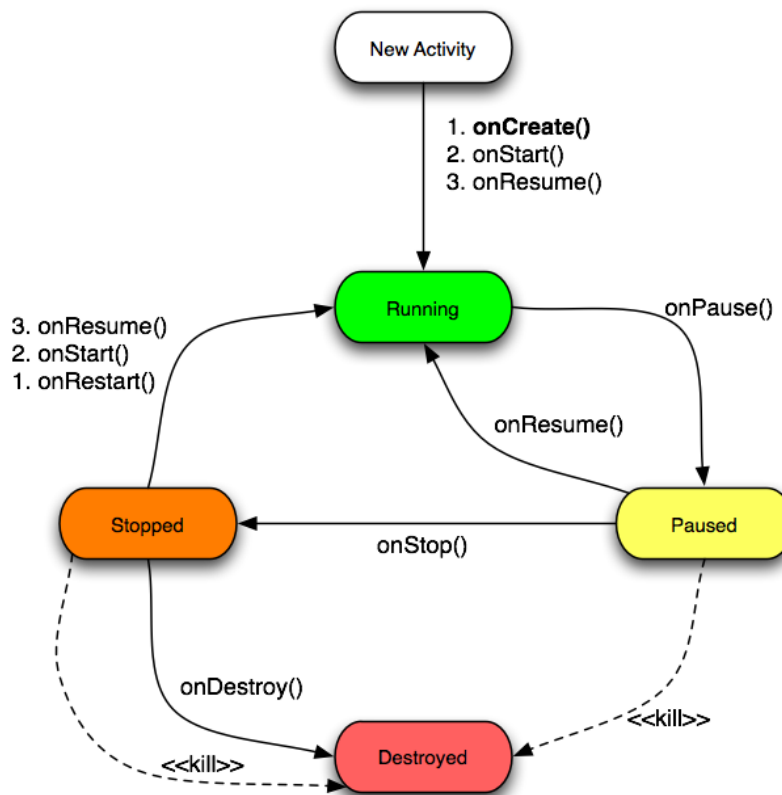


Hình 2 - Activity Stack

Các trạng thái của chu kỳ sống: một ứng dụng có 3 quá trình chính sau:

- Active(running) là khi đang chạy trên màn hình tập trung vào các thao tác của người sử dụng.
- Paused : là khi nó đang được tạm dừng nhưng vẫn trông thấy, tức là có một activity khác chạy trên nó nhưng không đầy màn hình nên có thể thấy được, lúc này activity vẫn còn sống nhưng có thể bị kết thúc nếu thiếu vùng nhớ.
- Stopped: nếu nó hoàn toàn bị bao phủ bởi activity khác, nó vẫn còn trạng thái và thông tin thành viên, và thường bị loại bỏ khi thiếu vùng nhớ.

Activity Lifecycle



Hình 3 - chu kì sống của một Activity

Một ứng dụng kết thúc khi mà mọi thành phần của nó kết thúc, khi activity kết thúc tức là người dùng không còn giao tiếp với ứng dụng nhưng không có nghĩa là ứng dụng đó kết thúc vì ngoài ra còn có Service, Broadcast ,... có nghĩa là các thành phần không tương tác người dùng vẫn chạy dưới sự quản lý của hệ điều hành cho đến khi người dùng tắt ứng dụng.

Một ứng dụng sẽ sống từ khi có lần đầu tiên gọi `onCreate()` cho đến trạng thái cuối cùng gọi `onDestroy()`, và hiển thị giữa một lần gọi `onStart()` đến một lần gọi `onStop()`, các phương thức của một chu trình sống:

- `onCreate()`:
 - thực hiện tất cả cài đặt tĩnh, tạo các view kết nối dữ liệu đến list
 - phương thức này được gửi qua đối tượng Bundle chứa đựng từ trạng thái trước của Activity.
 - Luôn theo sau bởi `onStart()`.

- **onRestart():**
 - được gọi sau khi ứng dụng đã dừng, và khởi động lại lần nữa
 - luôn theo sau bởi onStart().
- **onStart():**
 - được gọi trước khi activity hiện ra với người dùng.
 - theo sau bởi onResume() nếu activity đến trạng thái foreground hoặc onStop() để nó trở nên ẩn.
- **onResume():**
 - được gọi trước khi activity tương tác với người dùng.
 - tại đây activity nằm trên đỉnh của stack activity.
 - luôn theo sau bởi onPause().
- **onPause():**
 - Gọi khi hệ thống resume activity khác
 - Diễn hình cho việc bảo toàn dữ liệu
 - Theo sau bởi onResume() nếu activity trở về từ trước hoặc onStop() nếu nó trở nên hiện (visible) với người dùng.
- **onStop():**
 - gọi khi activity trở nên ẩn với người dùng (invisible).
 - Dừng khi nó bị hủy, theo sau là onDestroy() hoặc bị activity khác bao phủ, theo sau sẽ là onRestart().
- **onDestroy():**
 - gọi trước khi activity bị hủy
 - là lần gọi cuối cùng đối với activity này
 - dùng khi activity được hoàn thành hoặc bị hủy để tiết kiệm vùng nhớ.

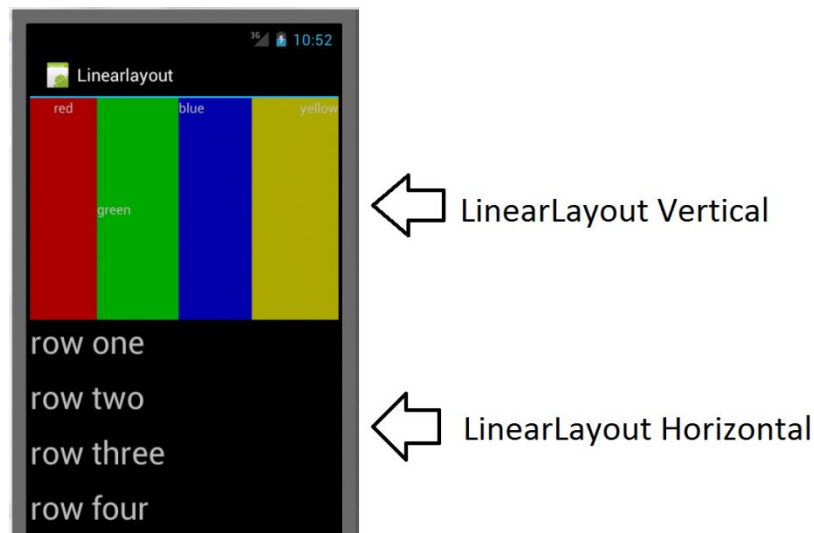
Trạng thái của activity có thể bị hệ thống kill.

3.2.7. Các thành phần giao diện của Android

View là các đối tượng xây dựng nên giao diện người dùng, có nhiều loại tất cả đều kế thừa từ lớp view, được gọi là các widget, các thuộc tính chung bao gồm vị trí, background, lề,...

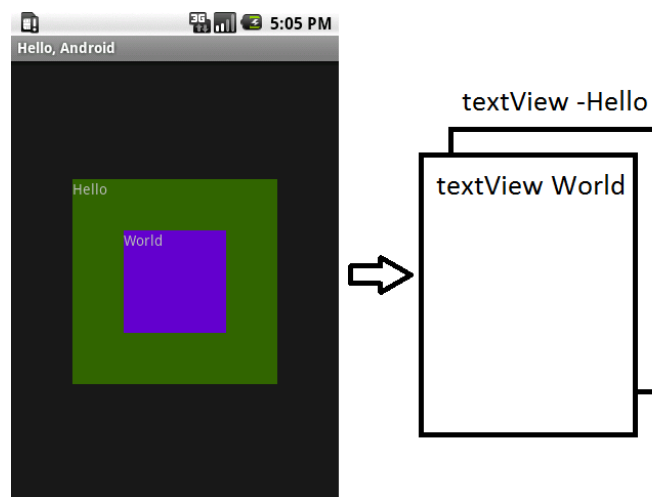
View group dùng để bố trí các đối tượng khác như button, text,...

Linear layout dùng bố trí các thành phần con theo chiều ngang hay dọc và không có xuống dòng. Các thành phần trong linear layout không phụ thuộc vào kích thước màn hình mà phụ thuộc vào quan hệ tương ứng giữa các thành phần.



Hình 4 - Sử dụng Linear Layout

Frame layout bố trí các đối tượng kiểu layout như photoshop, các đối tượng thuộc layout dưới sẽ bị che khuất bởi đối tượng thuộc layout trên, dùng cho các đối tượng muốn có khung hình bên ngoài như contact image button.

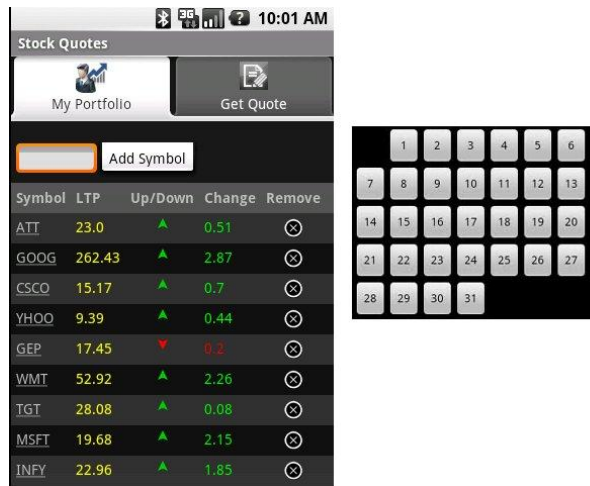


Hình 5 - bố trí các widget trong Frame Layout

Absolute layout bố trí các đối tượng con ở bất kỳ vị trí nào thông qua tọa độ x,y tuy nhiên nó không thay đổi theo khi màn hình thay đổi nên ít được sử dụng.

Retalive layout bố trí các thành phần con đối xứng dựa vào các vị trí trên, dưới, trái, phải của một đối tượng thuộc layout parent, bởi thế cũng không phụ thuộc vào kích thước màn hình.

Table layout dùng khi tạo một table chứa dữ liệu hoặc bố trí các widget theo kiểu hàng cột.



Hình 6 - bố trí các widget trong Table Layout

Button dùng tương tác với ứng dụng và một thành phần quan trọng và phổ biến, **Image button** có thêm thuộc tính image cho button. Các sự kiện xảy ra khi thực hiện các thao tác Click, LongClick,...

Image View hiển thị các image.

List view hiển thị thông tin theo từng dòng, mỗi dòng có một số các thông tin cố định. Có thể trong mỗi dòng của List lại có các thành phần khác như checkbox, layout khác, ...

Text view hiển thị văn bản nhưng không cho phép chỉnh sửa, **Edit text** cho phép chỉnh sửa nội dung cho các văn bản.

Check box chỉ nhận 2 giá trị true hay false, sử dụng trong nhiều trường hợp.

3.3. Giới thiệu về KIT Mini2440

Kit Mini2440 có kích thước 100mm vuông dựa trên nền tảng ARM9, sử dụng họ vi xử lý s3c2440, kit được ứng dụng cho việc phát triển hệ thống nhúng, điều khiển các thiết bị công nghiệp, phát triển trên thiết bị PDA và định vị GPS. Các hệ thống system on chip được sử dụng nhiều trong các thiết bị cầm tay như smartphone và PDA.

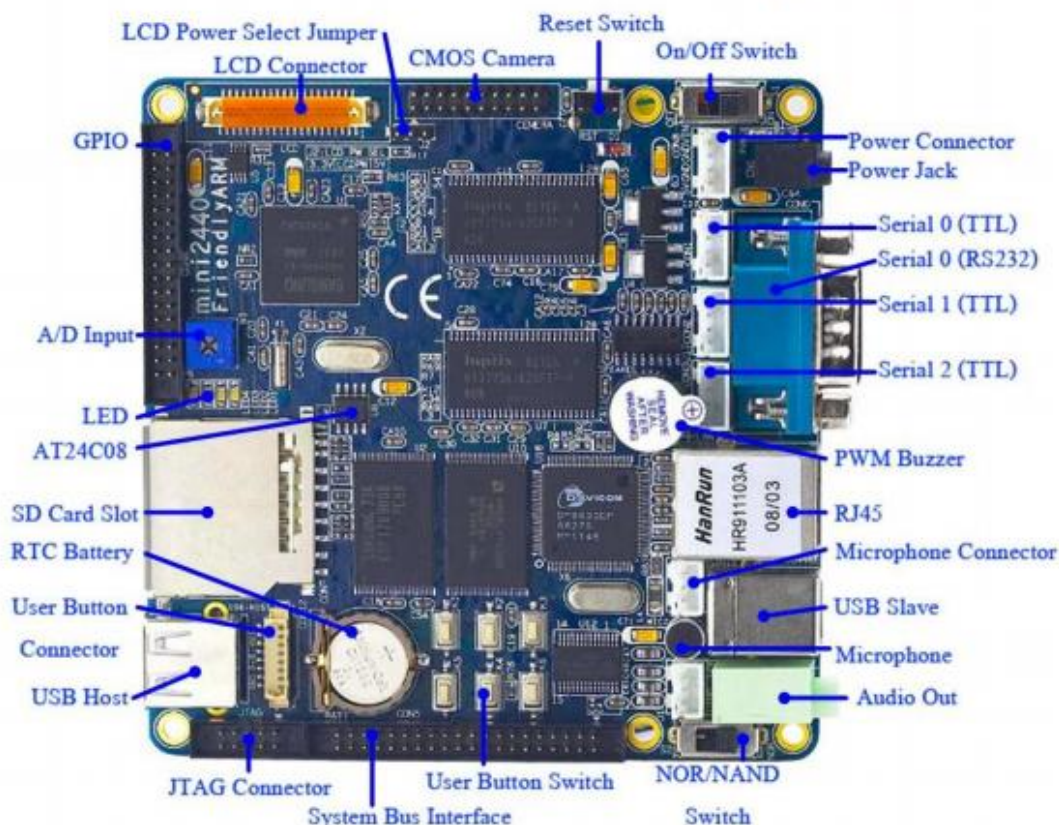
Kit Mini2440 có kích thước 3.9 x 3.9 inches (100 x 100mm). Mạch được thiết kế 4 lớp, được thiết kế đảm bảo các yêu cầu toàn vẹn tín hiệu đối với mạch tần số cao. Chip Samsung s3c2440 có lõi là cấu trúc ARM920T với tốc độ 400MHz (tần số thường dùng) và 533 MHz (tần số đỉnh).

Thành phần của kit Mini2440 gồm có các I/O port, Ethernet, USB host và slave, ba cổng nối tiếp, có thể chọn thêm module Wifi, camera CMOS và camera USB.

3.3.1. Cấu hình kỹ thuật

Vi xử lý	Samsung s3c2440 (lõi ARM920T)tần số 400MHZ, tần số đỉnh 533MHz
Memory	<ul style="list-style-type: none"> – 64MB SDRAM – 32 bit data bus – tốc độ 100MHz
Flash	<ul style="list-style-type: none"> – 256MB NAND flash – 2MB NOR flash
Flash mở rộng	1 x giao tiếp thẻ SD
Màn hình	<ul style="list-style-type: none"> – Màn hình 3.5 inch cảm ứng – Phân giải 1024x768 pixels – Hỗ trợ các chế độ đen trắng, 4,16 mức xám, 256, 4096 màu – Cấu hình chuẩn NEC 256K color 240x320/3.5; TFT True Color LCD
Giao tiếp mạng	1 x 10/100 giao tiếp Ethernet RJ45 (DM9000 chip) module Wifi
USB	1 x USB Host 1 x USB Slave (chuẩn giao tiếp loại B)
Serial	1x DB9 connector (RS232) có 3 cổng TTL
Audio	1 cổng ra stereo; 1 x mic
Camera	1 x 20-pin (kích thước 2.0 mm) kết nối camera; CMOS or USB cameras
Other I/O	<ul style="list-style-type: none"> – 1 x 10-pin (loại 2.0mm) chuẩn JTAG – 4 x LEDs

	<ul style="list-style-type: none"> – 6 x nút nhấn – 1 x PWM điều khiển loa – 1 x Biến trở để thử ADC – 1 x I2C bus AT24C08 chip, để kiểm tra I2C bus – 1 x 34-pin 2.0mm giao tiếp GPIO – 1 x 40-pin 2.0mm giao tiếp bus hệ thống
Pin RTC	
Nguồn	5V
Hệ điều hành	Linux-2.6.xx + Qtopia , Windows CE 5.0/6.0 và Android Hỗ trợ cài đặt hệ điều hành từ cổng USB

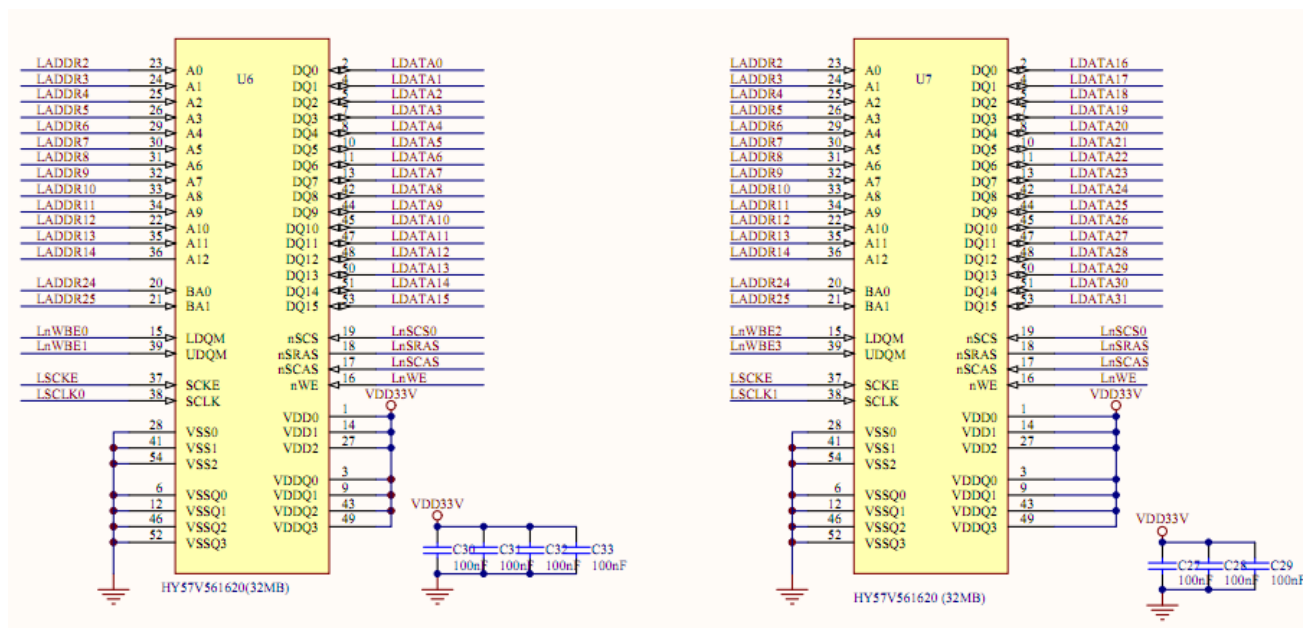


Hình 7 - Kit Mini2440

3.3.2. Các chú ý về cổng giao tiếp

a) SDRAM

Mini2440 sử dụng 2 bộ nhớ ngoài 32MB tổng cộng là 64 MB SDRAM chip (model: HY57V561620FTP), nối tiếp với nhau sẽ tạo thành data bus 32 bit tăng cao tốc độ truy cập, địa chỉ bắt đầu là 0x30000000, sơ đồ nguyên lý như sau:



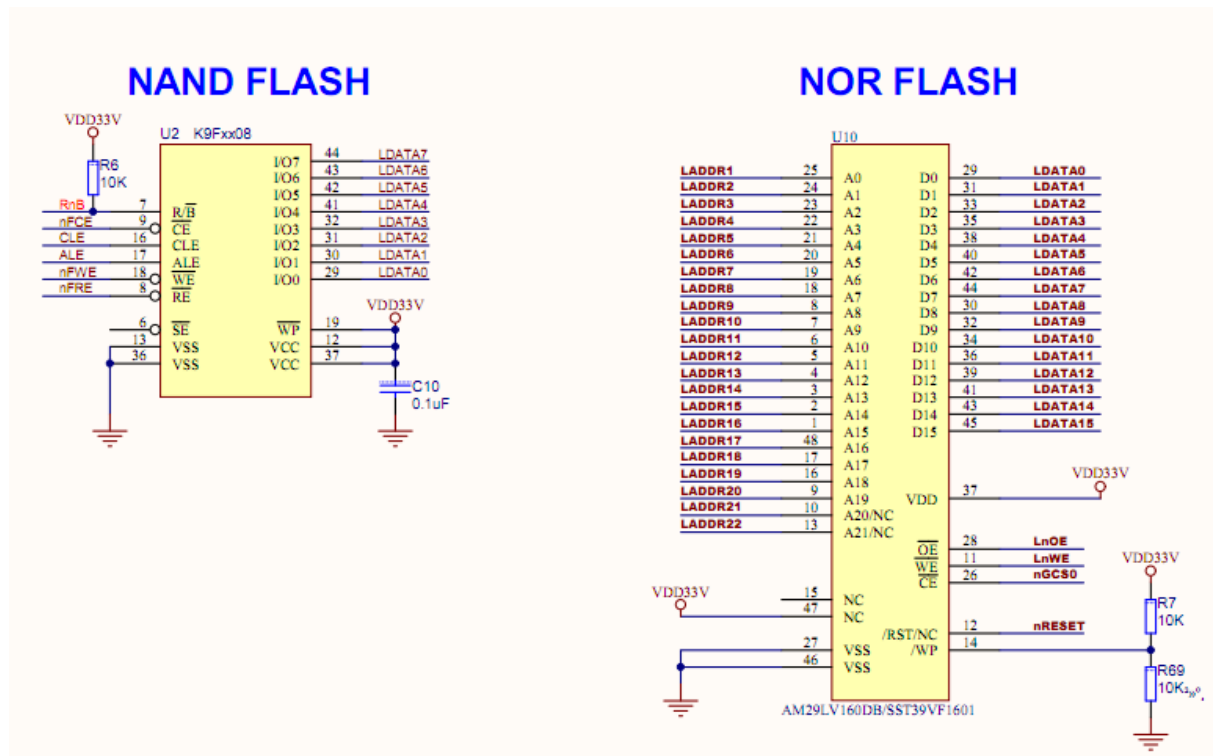
Hình 8 - Kết nối SDRAM

b) Flash

Mini2440 có 2 bộ nhớ Flash: NOR Flash (SST39VF1601, 2 Mbytes) và NAND Flash (K9F1208, 64 Mbytes), lựa chọn Boot Flash thông qua switch S2.

NAND Flash không sử dụng address line, dành riêng để kết nối giao diện điều khiển với CPU, sử dụng 8 bit data bus. Hầu hết các USB và Sdcard được sử dụng khi NAND Flash được bật.

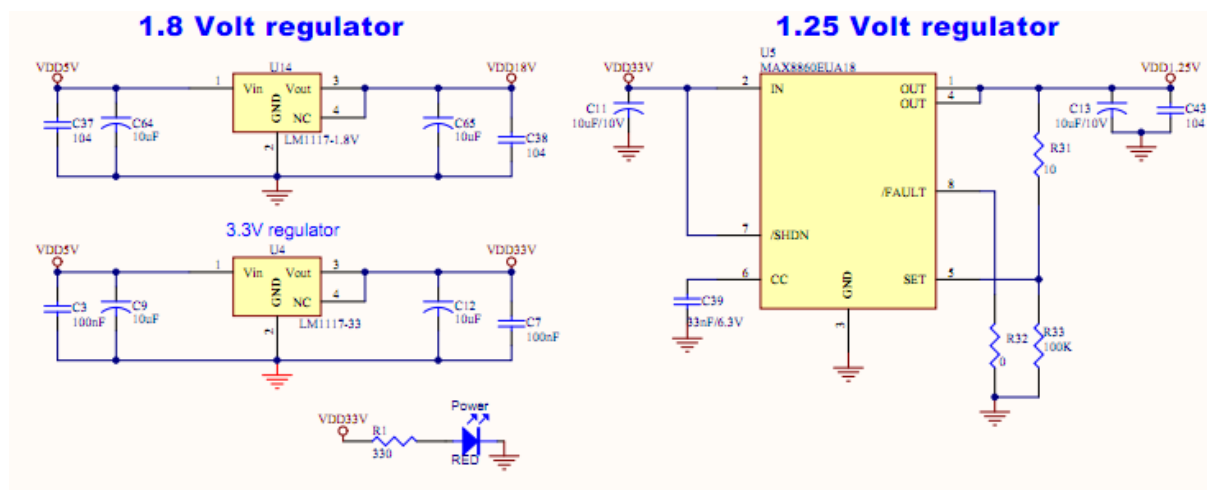
NOR Flash sử dụng A1-A22 chân địa chỉ và 16 chân dữ liệu. trên thực tế sơ đồ nguyên lý chỉ sử dụng có 20 chân địa chỉ, A21 và A22 có kết nối nhưng không sử dụng.



Hình 9 - kết nối NAND, NOR Flash

3.3.3. Nguồn hỗ trợ

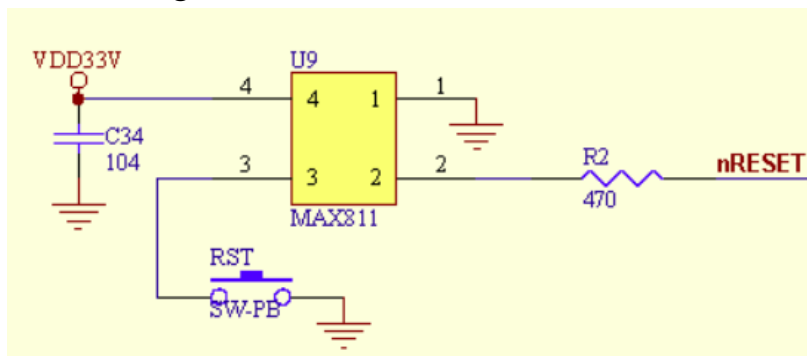
Mini 2440 sử dụng nguồn 5V, tuy nhiên do các đặc tính khác mà cần sử dụng thêm các mức: 3.3V, 1.8V, and 1.25V được tạo ra trực tiếp từ nguồn cấp 5V ở trên. Các nguồn được cấp thông qua switch S1 cấp cho toàn mạch, tuy nhiên cần chú ý là KIT không phải là một thiết bị di động nên đây không phải là cách quản lý nguồn tốt nhất.



Hình 10 - sơ đồ nguồn

3.3.4. Mạch khởi động lại hệ thống (System Reset)

Board sử dụng MAX881 để reset lại CPU:



Hình 11 - mạch reset

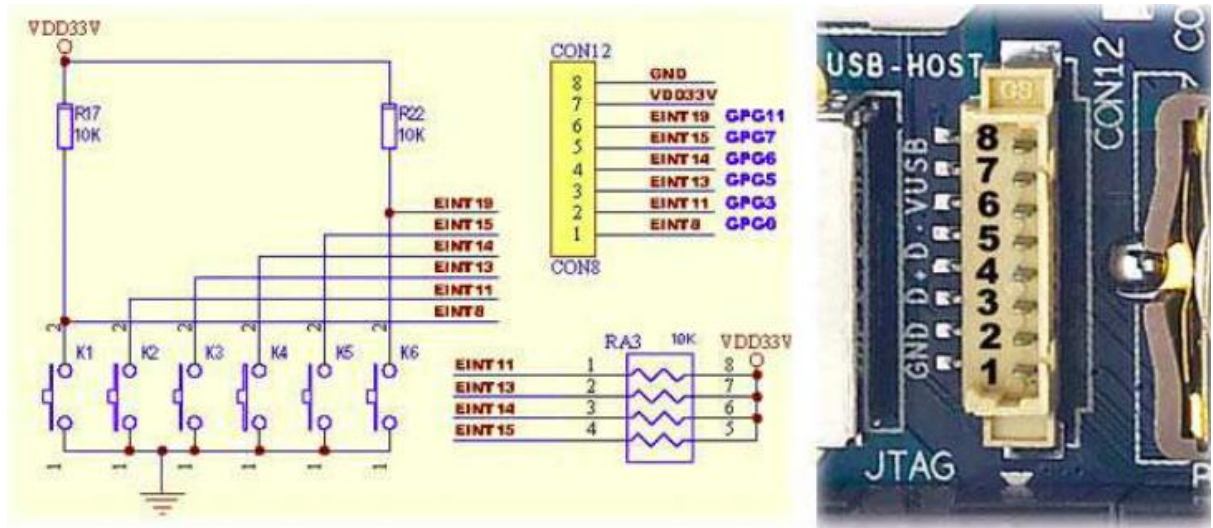
3.3.5. LEDs

	LED1	LED2	LED3	LED4
GPIO	GPB5	GPB6	GPB7	GPB8
Reusable for	nXBACK	nXREQ	nXDACK1	nDREQ1
Network Name	nLED_1	nLED_2	nLED_3	nLED_4

3.3.6. Nút Bấm

Có 6 nút bấm được đưa vào trên KIT, nối trực tiếp với các chân ngắt của Chip và là chân hoạt động tích cực mức thấp. các chân có thể sử dụng tùy mục đích khác nhau của người sử dụng, các button này được nối với CON12, sơ đồ như sau:

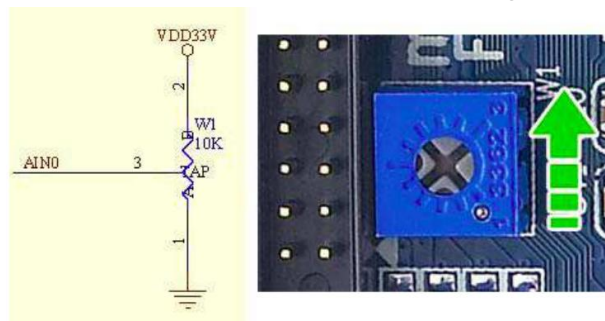
	K1	K2	K3	K4	K5	K6
Corresponding interrupt	EINT8	EINT11	EINT13	EINT14	EINT15	EINT19
GPIO Reuse	GPG0	GPG3	GPG5	GPG6	GPG7	GPG11
Special function 1	nothing	nSS1	SPIMISO1	SPIMOSI1	SPICLK1	TCLK1
CON12 corresponding pin	CON12.1	CON12.2	CON12.3	CON12.4	CON12.5	CON12.6



Hình 12 - Vị trí và kết nối nút bấm

3.3.7. A/D input test

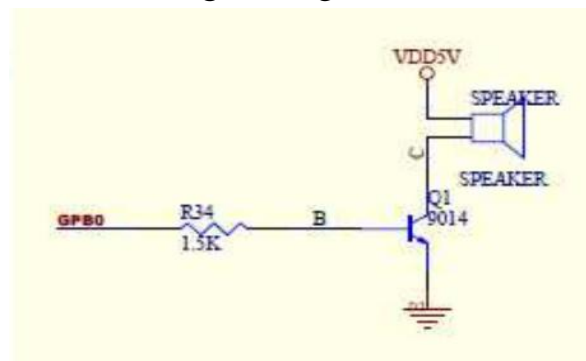
Có tổng cộng 4 kênh A/D được nối với CON4 GOIP. Để thuận lợi cho quá trình test AIN0 được kết nối với một biến trở R0 để thực nghiệm quá trình test.



Hình 13 - ADC input

3.3.8. Speaker

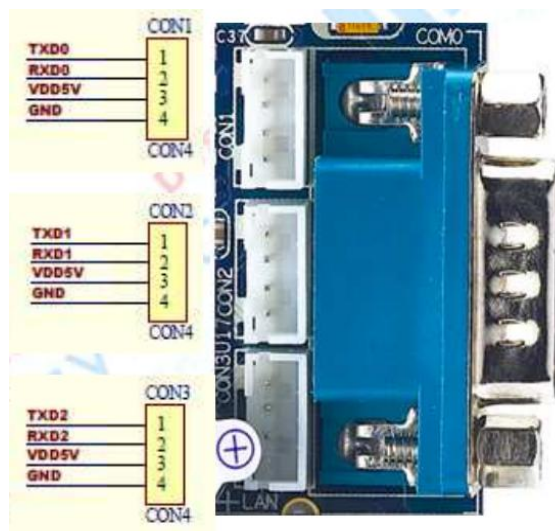
Mini2440 có một chân ra Analog nối với loa ngoài theo sơ đồ như sau, tín hiệu ra có thể sử dụng cho các loa thông thường.



Hình 14 - Kết nối loa ngoài

3.3.9. Serial Port

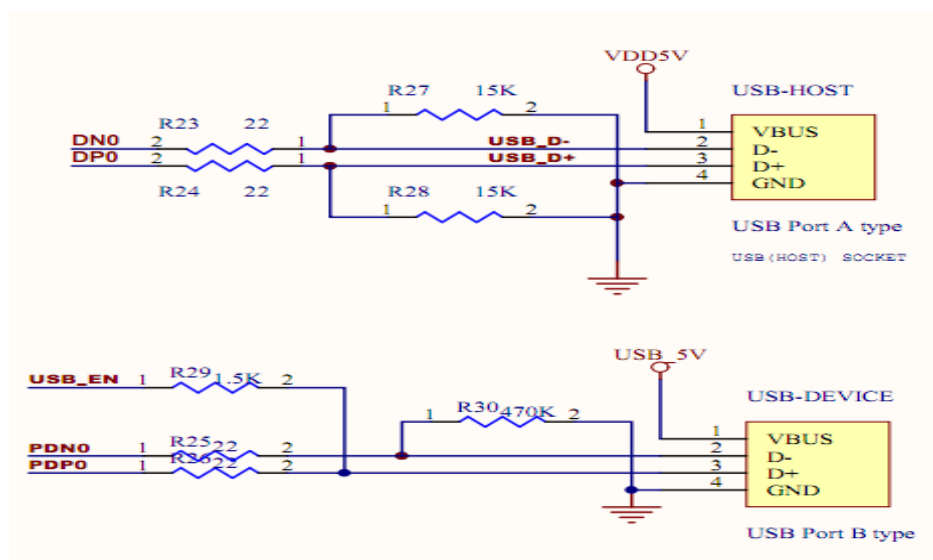
Có tổng cộng 3 cổng Serial trên board UART0,1,2. Trong hầu hết các ứng dụng, chỉ sử dụng đến 3 chức năng đơn giản như truyền và nhận dữ liệu, sẽ tương ứng với CON1,2,3 trên board. Để cho thuận tiện thì cổng COM0 được để trực tiếp dưới dạng RS232 converter.



Hình 15 - Kết nối cổng nối tiếp

3.3.10. Nối tiếp USB

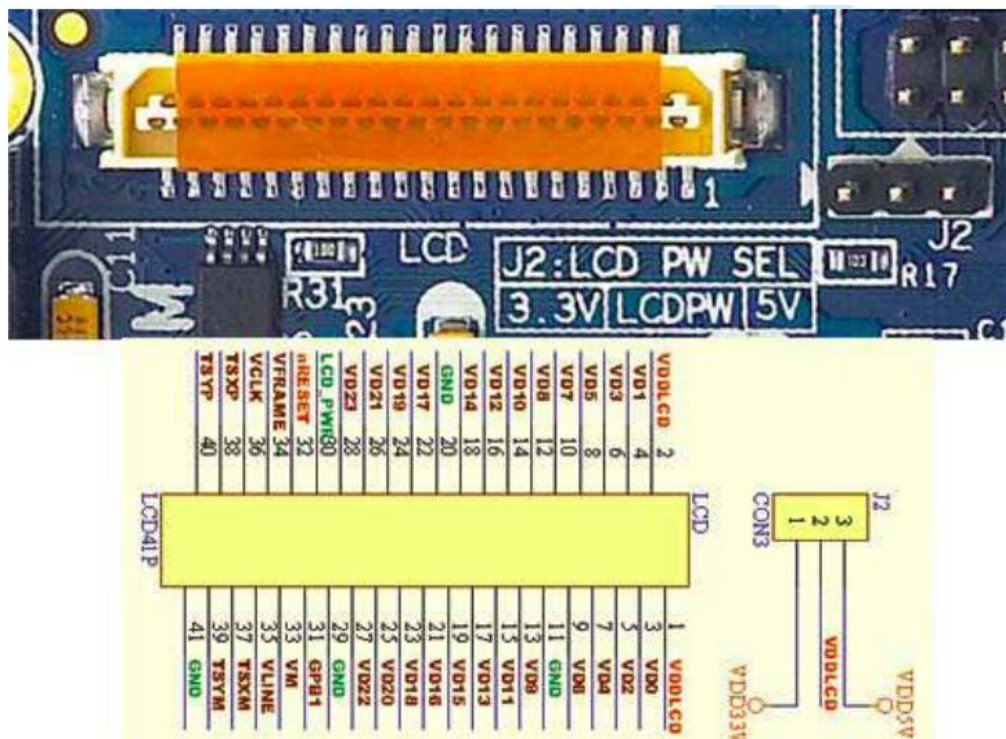
Có hai giao diện USB, một USB host tương tự như PC, có thể cắm USB camera, USB keyboard, USB mouse,...còn lại 1 USB slave dùng để download đến board.



Hình 16 - Nối tiếp USB

3.3.11. LCD interface

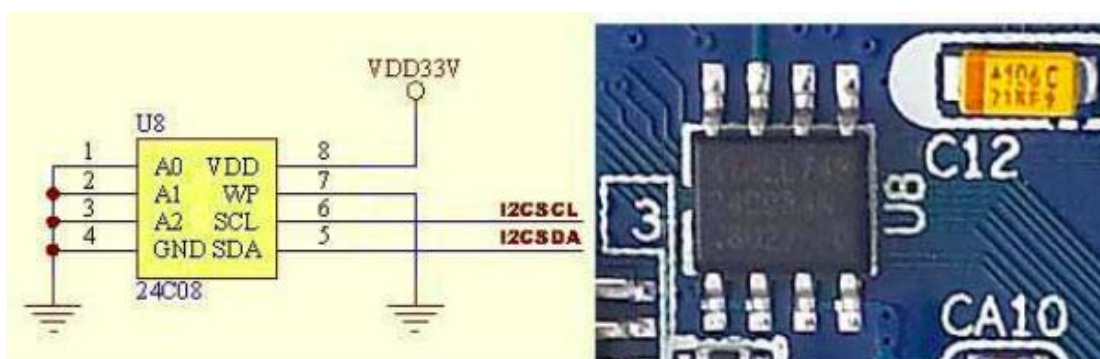
Giao diện LCB của board là loại 41-pin 0.5 mm pitch block. Dữ liệu ra là khối dữ liệu RGB 8:8:8, có thể hỗ trợ tối đa 16 triệu màu. Các chân 37,38,39,40 là các chân dành cho cảm ứng (touch screen).



Hình 17 - Giao tiếp LCD

3.3.12. EEPROM

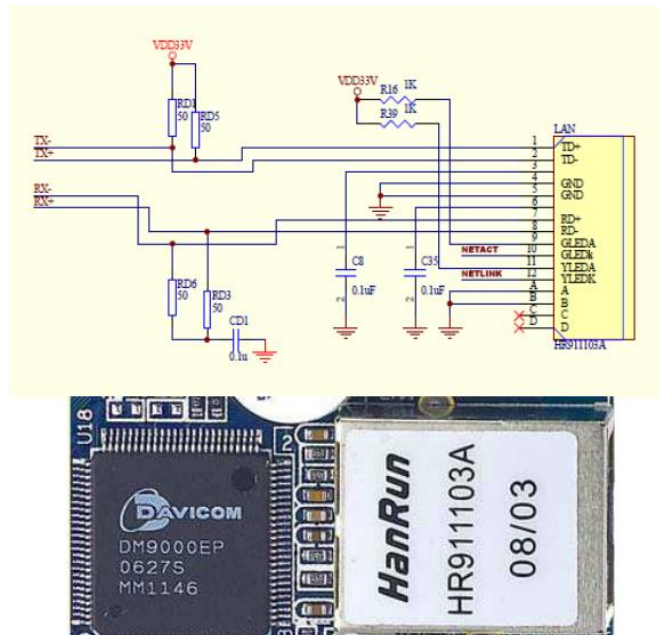
Board có thể kết nối tín hiệu I2C, thông qua chip AT24C08, đây chỉ là thử nghiệm kết nối I2C bus nên không có các thông số cụ thể



Hình 18 – EEPROM

Trang 30

Board sử dụng chip mạng DM900, có thể cắm trực tiếp board vào mạng LAN thông thường khi OS đã có driver cho DM900.



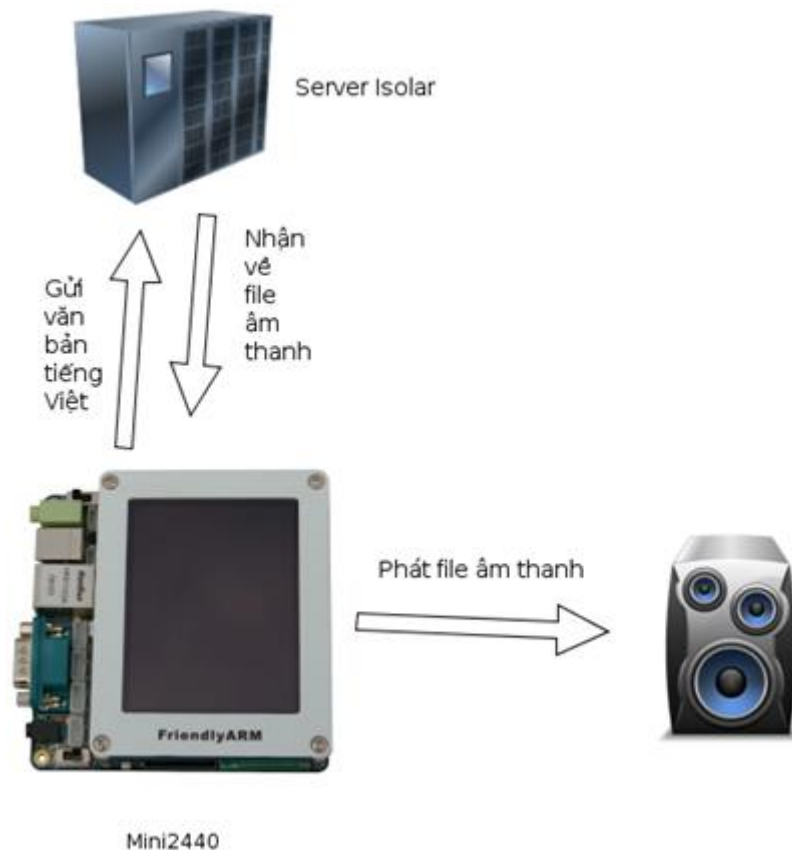
Hình 19 - Giao diện mạng

Ngoài ra còn có Audio Interface, JTAG Interface, GPIO, CMOS Camera Interface là các ứng dụng mở rộng thêm tùy theo mục đích của người sử dụng đã được tích hợp sẵn trên Board.

CHƯƠNG IV XÂY DỰNG ỨNG DỤNG

4.1.Mô hình hệ thống

Mô hình như sau:



Hình 20 - Mô hình hệ thống

Người dùng sẽ nhập văn bản tiếng việt qua KIT mini2440,sau khi nhập xong,KIT sẽ gửi yêu cầu bao gồm cả phần văn bản người dùng đã nhập lên server của isolar,sau đó nhận file âm thanh mà server đã xử lý về và phát ra loa

Những công việc cần làm trong hệ thống:

- Cài đặt nhân android lên KIT mini2440
- Viết chương trình cho phép nhập tiếng việt trên hệ thống KIT mini2440 qua bàn phím ảo,bàn phím thật hoặc qua file văn bản có sẵn tiếng Việt
- Viết chương trình tạo yêu cầu gửi đến server và nhận file âm thanh từ server gửi về
- Phát file âm thanh ra loa

4.2. Cài đặt trên KIT

4.2.1. Sửa nhân mini2440 cho màn hình X35 Sony.

Bản thân android được xây dựng phục vụ chính cho màn hình T35 của mini2440, màn hình X35 không được hỗ trợ mặc định, bởi thế để sử dụng chúng ta cần sửa và dịch lại nhân cho android để có thể sử dụng cho KIT mini2440X35. Hệ điều hành Android trong mini2440 gồm những thành phần sau:

Bootloader (Supervivi128Mb hoặc nboot)
Android Kernel (X35)
Android-Fsroot (tạo file .img)

Vì T35 và X35 đều có các thiết bị ngoại vi giống nhau nên điều cần thiết duy nhất đó là sửa lại driver của màn hình T35 để có thể chạy được trên X35.

Để làm được điều đó, đầu tiên ta phải sửa cấu hình của màn hình:

```
# cd /usr/local/android/kernel
# cd drivers/video/
# gedit Kconfig
```

Bắt đầu từ dòng 69, thay đổi như sau (từ trái thành phải)

<pre>choice prompt "LCD select" depends on FB_S3C2410 help S3C24x0 LCD size select config FB_S3C2410_T240320 boolean "3.5 inch 240X320 Toppoly LCD" depends on FB_S3C2410 help 3.5 inch 240X320 Toppoly LCD</pre>	<pre>choice prompt "LCD select" depends on FB_S3C2410 help S3C24x0 LCD size select config FB_S3C2410_X240320 boolean "3.5 inch 240X320 SONY LCD" depends on FB_S3C2410 help 3.5 inch 240x320 SONY LCD</pre>
--	--

Như ta thấy, màn hình của T35:s3c2440_T240320 đã được sửa lại thành của X35: S3C2410_X240320

Tiếp theo đó, chúng ta phải sửa lại driver cho màn hình X35:

cd /usr/local/android/kernel/arch/arm/mach-s3c2440/

gedit mach-mini2440.c

Từ dòng 169, thay đổi như sau:

<pre>#elifdefined(CONFIG_FB_S3C2410_T240320) #define LCD_WIDTH 320 #define LCD_HEIGHT 240 #define LCD_PIXCLOCK 170000 #define LCD_RIGHT_MARGIN 68 #define LCD_LEFT_MARGIN 4 #define LCD_HSYNC_LEN 5 #define LCD_UPPER_MARGIN 10 #define LCD_LOWER_MARGIN 4 #define LCD_VSYNC_LEN 1</pre>	<pre>#elif defined(CONFIG_FB_S3C2410_X240320) #define LCD_WIDTH 240 #define LCD_HEIGHT 320 #define LCD_PIXCLOCK 170000 #define LCD_RIGHT_MARGIN 25 #define LCD_LEFT_MARGIN 0 #define LCD_HSYNC_LEN 4 #define LCD_UPPER_MARGIN 0 #define LCD_LOWER_MARGIN 4 #define LCD_VSYNC_LEN 9 #define LCD_CON5 (S3C2410_LCDCON5_FRM565 S3C2410_LCDCON5_INVVDEN S3C2410_LCDCON5_INVVFRAME S3C2410_LCDCON5_INVVLINE S3C2410_LCDCON5_INVVCLK S3C2410_LCDCON5_HWSWP S3C2410_LCDCON5_PWREN)</pre>
--	--

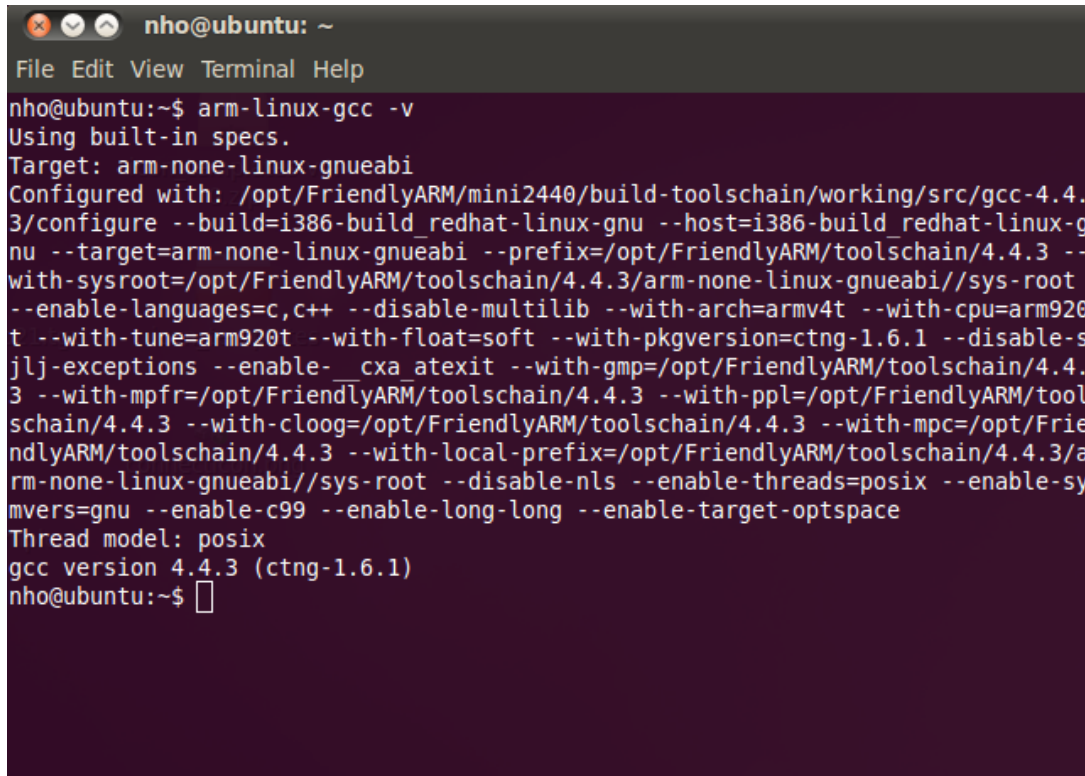
Tiếp theo, từ dòng 322, thay đổi như sau:

<pre>static void __init smdk2440_machine_init(void) { //s3c24xx_fb_set_platdata(&smdk2440_fb_info); #if defined (LCD_WIDTH) s3c24xx_fb_set_platdata(&mini2440_fb_info); #endif platform_add_devices(smdk2440_devices, ARRAY_SIZE(smdk2440_devices)); smdk_machine_init(); }</pre>	<pre>static void __init smdk2440_machine_init(void) { //s3c24xx_fb_set_platdata(&smdk2440_fb_info); #if defined (LCD_WIDTH) s3c24xx_fb_set_platdata(&mini2440_fb_info); #endif platform_add_devices(smdk2440_devices, ARRAY_SIZE(smdk2440_devices)); s3c2410_gpio_cfgpin(S3C2410_GPG4, S3C2410_GPG4_OUTP); s3c2410_gpio_setpin(S3C2410_GPG4, 1); smdk_machine_init(); }</pre>
--	--

Đó là tất cả những gì mà nhân android phải sửa để có thể chạy trên thiết bị KIT mini2440X35. Công việc tiếp theo là dịch nhân để tạo ra file zImage(nhân linux) nạp riêng cho X35 và tạo file img(file system) để nạp vào KIT

4.2.2. Dịch nhân android

Trước hết phải đảm bảo là đã có công cụ arm-linux-gcc để biên dịch chéo, Gõ lệnh arm-gcc-v để kiểm tra xem đã cài đặt công cụ trên chưa



```
nho@ubuntu: ~  
File Edit View Terminal Help  
nho@ubuntu:~$ arm-linux-gcc -v  
Using built-in specs.  
Target: arm-none-linux-gnueabi  
Configured with: /opt/FriendlyARM/mini2440/build-toolschain/working/src/gcc-4.4.3/configure --build=i386-build_redhat-linux-gnu --host=i386-build_redhat-linux-gnu --target=arm-none-linux-gnueabi --prefix=/opt/FriendlyARM/toolschain/4.4.3 --with-sysroot=/opt/FriendlyARM/toolschain/4.4.3/arm-none-linux-gnueabi//sys-root --enable-languages=c,c++ --disable-multilib --with-arch=armv4t --with-cpu=arm920t --with-tune=arm920t --with-float=soft --with-pkgversion=ctng-1.6.1 --disable-sjlj-exceptions --enable-cxa-atexit --with-gmp=/opt/FriendlyARM/toolschain/4.4.3 --with-mpfr=/opt/FriendlyARM/toolschain/4.4.3 --with-ppl=/opt/FriendlyARM/toolschain/4.4.3 --with-cloog=/opt/FriendlyARM/toolschain/4.4.3 --with-mpc=/opt/FriendlyARM/toolschain/4.4.3 --with-local-prefix=/opt/FriendlyARM/toolschain/4.4.3/arm-none-linux-gnueabi//sys-root --disable-nls --enable-threads=posix --enable-symvers=gnu --enable-c99 --enable-long-long --enable-target-optspace  
Thread model: posix  
gcc version 4.4.3 (ctng-1.6.1)  
nho@ubuntu:~$
```

Hình 21 - Biên dịch chéo gcc

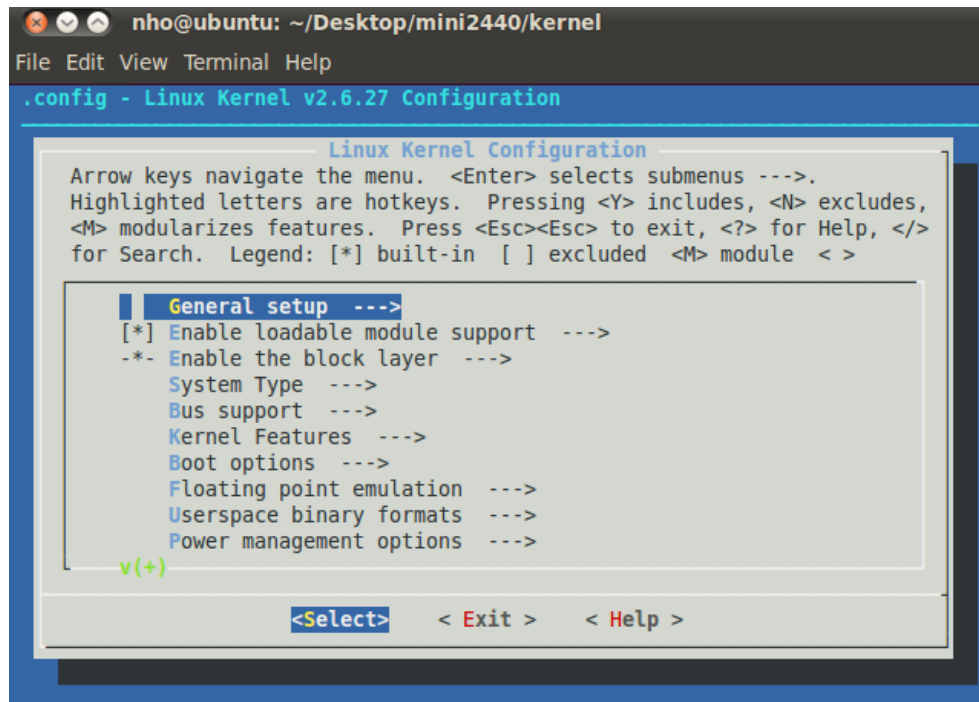
\$ cd /usr/local/android/kernel/

\$ cp config_mini2440 .config

\$ make menuconfig

Ta có được menuconfig hiện lên,ta chọn:

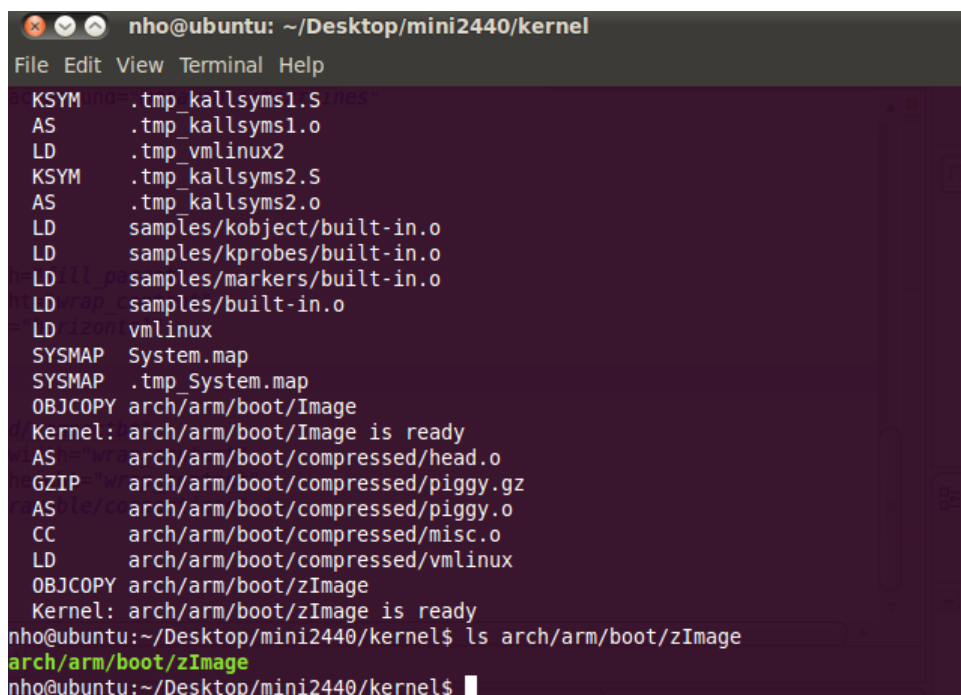
Device Drives -> Graphic support -> Support for frame buffer devices -> S3C2410 LCD framebuffer support -> (X) 3.5 inch 240x320 Samsung LCD



Hình 22 - menuconfig

Sau khi chọn, ấn exit và tạo file zImage, file này chứa nhân của mini 2440X35:

\$ make zImage



Hình 23 - tạo zImage

Sẽ mất vài phút để dịch tất cả. File zImage sẽ nằm tại đường dẫn:

/usr/local/android/kernel/arch/arm/boot/zImage

4.2.3. Dịch file system

Trong file system, có vài thông số ta cần thay đổi, thứ nhất là ngôn ngữ, mặc định trong file hệ thống fsroot android 1.5 thì sẽ sử dụng tiếng Trung Quốc, ta phải thay đổi vài thông số để có thể sử dụng tiếng Anh:

```
$ cd /usr/local/android/fs/
```

```
$ gedit default.prop
```

Thay đổi các dòng sau để nhân trở thành tiếng Anh

```
persist.sys.country = CN
```

```
persist.sys.language = zh
```

thành

```
persist.sys.country = US
```

```
persist.sys.language = en
```

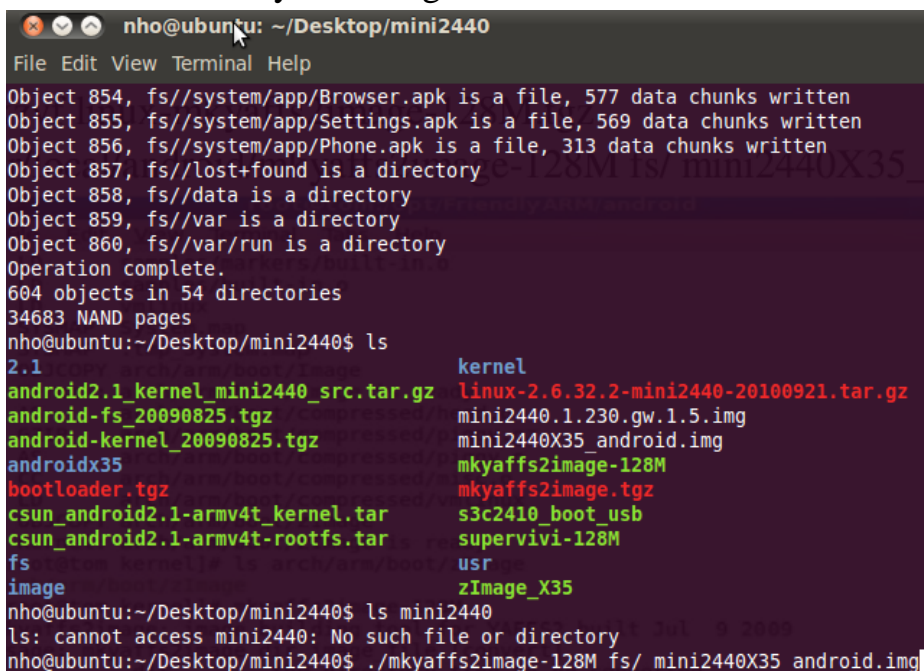
Sau đó, mặc định cổng ethernet của KIT sẽ sử dụng địa chỉ IP là 192.168.1.230 và default gateway là 192.168.1.1. Để thay đổi địa chỉ này theo ý muốn, ta chỉ việc sửa file :

```
$ gedit /usr/local/android/fs/system/etc/shine/net.conf
```

Cuối cùng là dịch fsroot thành file img tương ứng, ở đây ta sử dụng 1 phần mềm tên **mkyaffs2image** để dịch

```
$ tar xzvf linux-mkyaffs2image-128M.tgz
```

```
$ ./usr/local/android/mkyaffs2image-128M fs/ mini2440X35_android.img
```



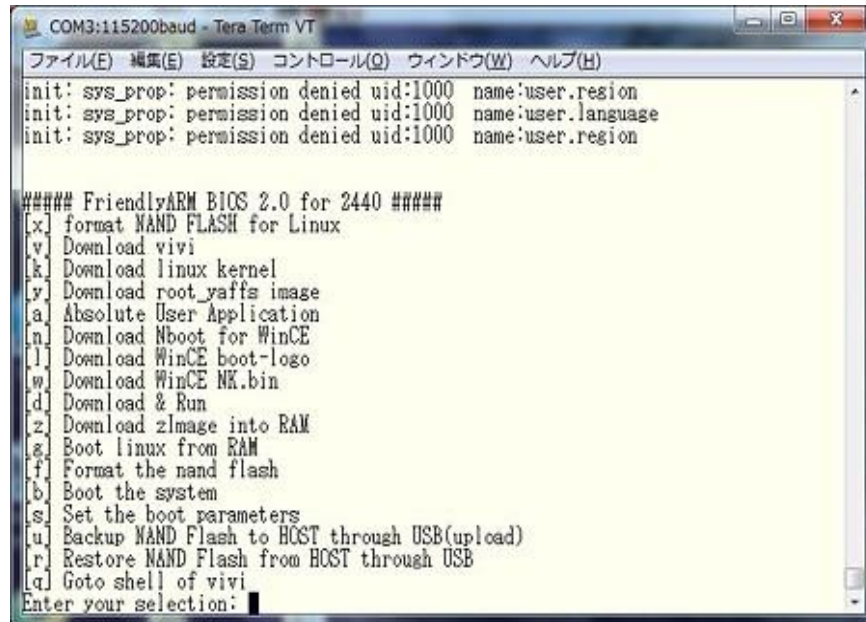
```
nho@ubuntu: ~/Desktop/mini2440
File Edit View Terminal Help
Object 854, fs//system/app/Browser.apk is a file, 577 data chunks written
Object 855, fs//system/app/Settings.apk is a file, 569 data chunks written
Object 856, fs//system/app/Phone.apk is a file, 313 data chunks written
Object 857, fs//lost+found is a directory
Object 858, fs//data is a directory
Object 859, fs//var is a directory
Object 860, fs//var/run is a directory
Operation complete.
604 objects in 54 directories
34683 NAND pages
nho@ubuntu:~/Desktop/mini2440$ ls
2.1
android2.1_kernel_mini2440_src.tar.gz
android-fs_20090825.tgz
android-kernel_20090825.tgz
androidx35
bootloader.tgz
csun_android2.1-armv4t_kernel.tar
csun_android2.1-armv4t-rootfs.tar
fs
image
kernel
linux-2.6.32.2-mini2440-20100921.tar.gz
mini2440.1.230.gw.1.5.img
mini2440X35_android.img
mkyaffs2image-128M
mkyaffs2image.tgz
s3c2410_boot_usb
supervivi-128M
usr
zImage_X35
nho@ubuntu:~/Desktop/mini2440$ ls mini2440
ls: cannot access mini2440: No such file or directory
nho@ubuntu:~/Desktop/mini2440$ ./mkyaffs2image-128M fs/ mini2440X35_android.img
```

Hình 24 - dịch root file

File system có tên là mini2440X35_android.img nằm tại
/usr/local/android/

4.2.4. Cài đặt Android cho mini2440

Trước hết, hãy chuyển đổi switch của KIT mini2440 sang NOR Flash, sau đó mở dnw,ta sẽ có màn hình:



Hình 25 - giao diện dnw

- Đầu tiên cần nên format lại bộ nhớ NAND chọn [x] và [f].
- Sau đó, chọn [v] tải superVivi128M hoặc vboot.
- Sau đó chọn [k] để tải zImage_android .
- Cuối cùng chọn [y] để port mini2440T35_android.img.

Cuối cùng,chuyển switch sang NAND flash và reboot KIT
Kết quả:



Hình 26 - kết quả dịch nhân

4.3. Giao diện chương trình

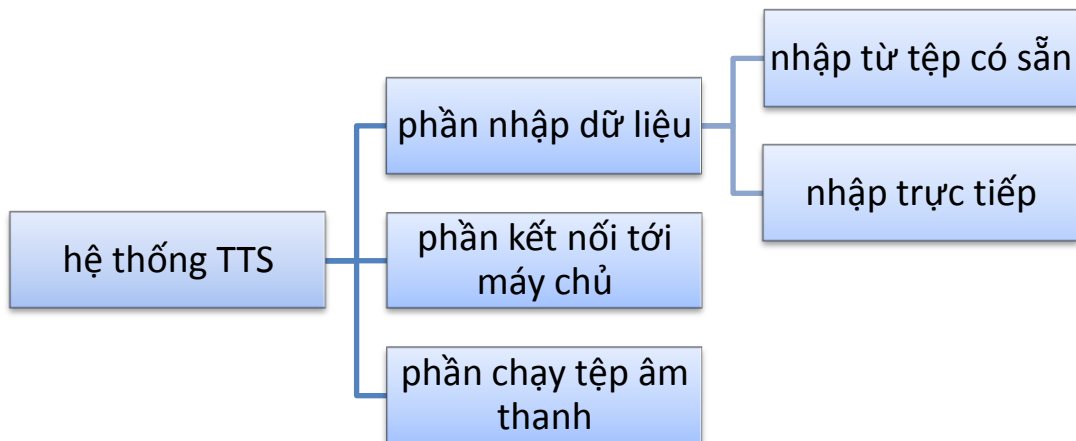
4.3.1. Thiết kế giao diện chương trình

❖ Biểu đồ thiết kế giao diện

Biểu đồ sẽ phân cấp và xác định các mục đích nhất định, đảm bảo tính nhất quán và dễ dùng cho ứng dụng, đồng thời tạo định hướng dễ dàng thiết kế chi tiết.

Có thể tóm tắt quy trình hoạt động của thiết bị như sau:

- Người dùng nhập đoạn text cần thiết để phát âm vào, có 2 cách để nhập là nhập từ file và nhập trực tiếp.
- Sau đó hệ thống sẽ gửi đoạn text lên server, server sẽ thực hiện chuyển văn bản thành tiếng nói và gửi trả về file âm thanh.
- Hệ thống thực hiện phát file âm thanh. Ngoài ra hệ thống cho phép phát lại các file âm thanh đã dịch trước đó, và thực hiện các thao tác như với một file âm thanh thông thường.



Hình 27 - Biểu đồ giao diện

❖ Thiết kế giao diện người dùng

- Giao diện nhập dữ liệu từ tệp

Nhập dữ liệu từ tệp cần một màn hình chứa đoạn văn bản trong tệp và một nút bấm khi ấn vào sẽ mở đường dẫn đến thư mục chứa các tệp văn bản chứa các đoạn đã nhập sẵn hay nhập trước đó.

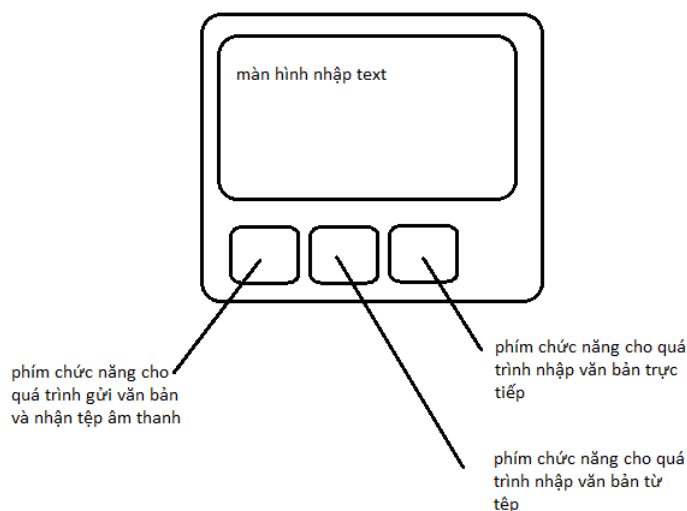
- Giao diện nhập dữ liệu trực tiếp

Nhập dữ liệu trực tiếp cần có một bàn phím để nhập các kí tự và các hộp thoại chọn các thao tác khác như xóa toàn bộ, chọn kiểu gõ, ... phần văn bản đã nhập có thể hiển thị luôn trên phần màn hình của giao diện nhập dữ liệu từ tệp. Tuy nhiên do phần này cần khá nhiều diện tích, nếu xây dựng cùng với phần giao diện nhập dữ liệu từ tệp sẽ chiếm quá nhiều diện tích nên phần này sẽ được tách làm một lớp giao diện riêng có màn hình nhập liệu riêng, sau khi trở về từ giao diện nhập trực tiếp sẽ đưa phần văn bản có trong đó vào phần màn hình hiển thị ở giao diện nhập từ tệp. Đồng thời cần thêm một phím chức năng để mở ra giao diện nhập dữ liệu bằng tay.

- **Giao diện kết nối**

Giao diện kết nối đơn giản chỉ cần một phím khi ấn thực hiện chuyển tải đoạn văn bản và sau đó tải tệp âm thanh về lưu vào thẻ nhớ hay bộ phận lưu trữ của thiết bị. Cần có thêm một phím chức năng ngắt kết nối nếu thời gian tải về quá lâu do đoạn văn bản quá dài hoặc do kết nối chậm mà người dùng muốn bỏ qua.

Do giao diện chỉ gồm hai phím chức năng đơn giản nên thực hiện đưa nó vào bảng chức năng của phần nhập liệu để giảm thiểu diện tích cho ứng dụng.

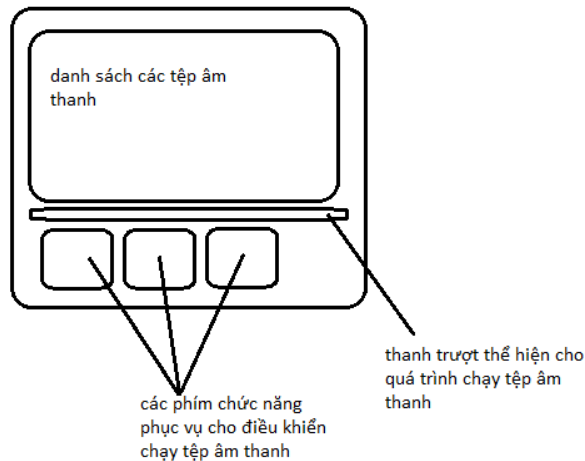


Hình 28 - mô hình phần nhập văn bản

- **Giao diện chạy tệp âm thanh**

Phần chạy tệp âm thanh sẽ cần một danh sách các tệp âm thanh đã được chạy trước đó, phục vụ cho chức năng nghe lại nếu cần, đoạn âm thanh vừa thực hiện chuyển từ văn bản cũng nằm tại vị trí cuối cùng trong danh sách này.

Bảng điều khiển chạy tệp âm thanh, tương tự như một phần mềm chạy thông thường với thanh trượt thể hiện phần đã chạy, các nút bấm tương ứng với các chức năng chạy, tạm dừng, ... được bố trí bên dưới.



Hình 29 - mô hình phần chạy tệp âm thanh

4.3.2. Giao diện chi tiết

Giao diện chương trình được xây dựng trong 5 layout xếp dọc, các thành phần trong mỗi layout được sắp xếp như sau:



Hình 30 - giao diện chung

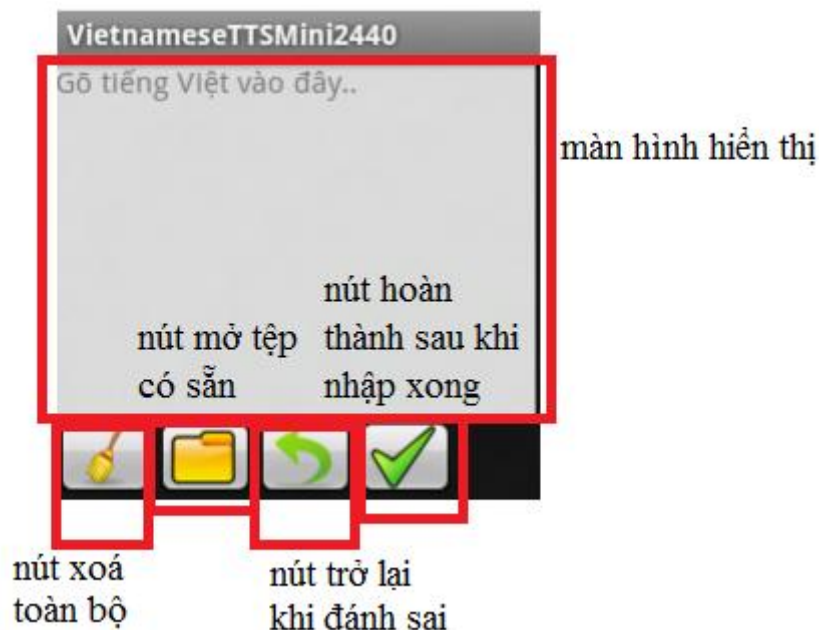
Linear layout thứ nháp chứa một edit text, ta sẽ nháp tiếng việt vào đây để thực hiện chuyển đoạn văn bản này thành tiếng nói và phát ra.

Linear layout thứ hai chứa 3 image button và một check box, nút submit sẽ thực hiện chuyển text sang server, nút Choose sẽ thực hiện nhập text từ một file tiếng việt có sẵn (như một văn bản chẳng hạn), nút Clear sẽ xóa trắng toàn bộ nội dung trong edit text, và check box sẽ cho phép người dùng nhập text thông thường.

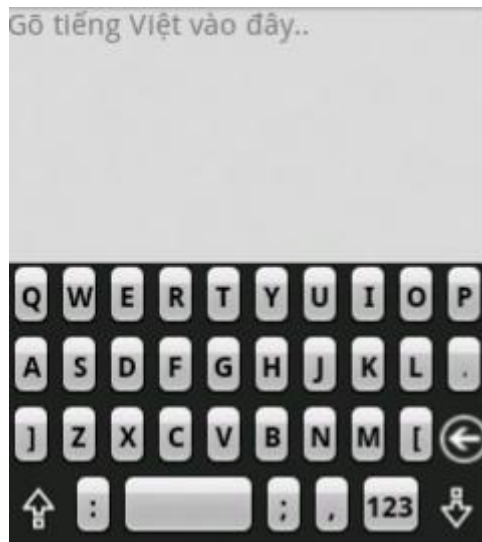


Hình 31 - thanh điều khiển nhập text

Trên thanh điều khiển nhập văn bản có checkbox để mở ra giao diện nhập văn bản bằng tay, khi ấn vào checkbox này giao diện nhập văn bản sẽ hiện ra. Giao diện nhập văn bản bằng tay gồm có một TextEdit là màn hình hiển thị đoạn text nhập bằng tay, khi ấn vào màn hình này bản phím ảo sẽ hiện ra phục vụ cho việc nhập văn bản, khi hoàn thành xong đoạn văn bản chỉ cần nhấp lại vào nút hoàn thành để trở về giao diện phát âm thanh. Ngoài ra cũng có thể mở các tệp có sẵn để chỉnh sửa rồi mới sử dụng.



Hình 32 - màn hình nhập văn bản



Hình 33 - bàn phím ảo

Tại giao diện nhập văn bản ấn vào menu, chọn mục **tuỳ chỉnh** sẽ là các lựa chọn cho việc nhập văn bản như bật tắt chế độ gõ tiếng việt, chọn kiểu gõ, chọn chế độ bỏ dấu thông minh, chế độ đánh dấu kiểu cổ điển.



Hình 34 - Tuỳ chỉnh của chương trình

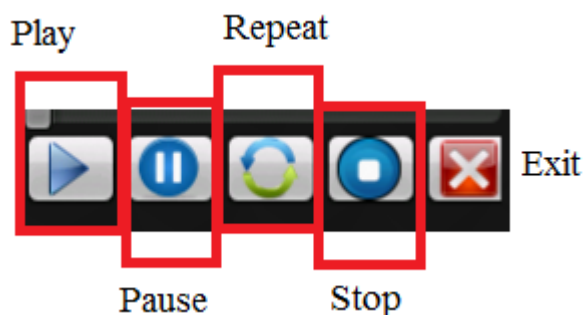
Linear layout thứ ba có một List view chứa các file được tải về để có thể chọn chạy lại chỉnh sửa hay xóa,... ô check box là để cho chọn các file để thao tác, bên cạnh là hai text view chứa tên và đường dẫn đến file tương ứng, ấn menu để chọn xóa.



Hình 35 – danh sách tập tin âm thanh

Linear layout thứ tư là thanh seebar thể hiện cho các đoạn âm thanh đang được chạy, có thể tùy chỉnh các đoạn âm thanh tại các vị trí muốn nghe.

Linear layout thứ năm là các nút bấm chạy, dừng, tạm dừng, chạy lại từ đầu cho các đoạn âm thanh đã được liệt kê trong List view ở trên, muốn xóa các đoạn âm thanh có thể bấm vào menu rồi chọn delete.



Hình 36 - thanh điều khiển phát âm thanh

4.4.Làm bộ gõ tiếng việt với Android

4.4.1. Cấu tạo chung của tiếng việt

Đầu tiên, để xử lý tiếng việt cần phải mô tả được các thành phần của một chữ tiếng việt. Hệ thống âm vị tiếng việt có:

- 11 nguyên âm đơn (monothong): **a, ă, â, e, ê, i, o, ô, ơ, u, ư**. Chữ y đứng một mình và chữ i đứng một mình là hai lỗi viết của cùng một nguyên âm, vì thế không tính y.
- 30 nhị trùng âm (diphthong): **ai, ao, au, ay, âu, ây, eo, êu, ia, iê, iu, oa, oă, oe, oi, ôi, ơi, ua, uă, uê, ui, uô, uơ, uy, ura, uri, uơ, uru, oo, ôô**; Hai

âm sau có nhưng rất ít dùng: oo (cái soong), ôô (ôôc đôôc, tức là ồồ dồồ nói giọng Quảng Bình);

- 12 tam trùng âm (triphthong): **iêu, oai, oay, uây, uôi, uyê, uyu, uoi, ươu, uya, oao, oeo**; yêu không khác gì iêu về âm nên không tính.
- 15 phụ âm đơn (consonant): **b, c(k,qu), d, đ, g, h, l, m, n, p, r, s, t, v, x**;
- 8 phụ âm kép: **ch, gi, kh, nh, ng, ph, th, tr**.

Tổng cộng số âm và vần là 76.

- 6 thanh (accent) **không, huyền, hỏi, sắc, ngã, nặng**. Cộng là 82 âm vị (phoneme).
- 17 phụ âm chỉ đứng đầu: **b, d, đ, g, h, k, l, r, s, v, x, qu, gi, kh, ph, th, tr**;
- 8 phụ âm có thể đứng đầu và cuối: **c, m, n, p, t, ch, nh, ng**; p thoát đầu không phải phụ âm đầu nhưng khi du nhập các từ nước ngoài, nó được tính là phụ âm đầu.
- 10 nguyên âm không thể kết thúc chữ (nguyên âm lưng): **ă, â, iê, oă, uâ, uô, oo, ôô, ươ, uyê**;
- 28 nguyên âm luôn kết thúc chữ, tức là không có âm gì có thể đứng sau chúng (nguyên âm cuối): **ai, ao, au, ay, âu, ây, eo, êu, ia, iu, oi, ôi, oí, ui, ua, ui, uu, iêu, uôi, uyu, uoi, ươu, oai, oay, uây, uya, oeo, oao**;

Một số luật do thói quen.. Khi xây dựng lý thuyết ta không áp dụng các luật này mà chỉ đưa vào sau cùng như một lựa chọn.

- **g** và **ng** khi đứng liền trước **e, ê, i** viết là **gh** và **ngh**
- **c, k, q** là một âm vị đồng nhất, theo thói quen, viết là:
 - c khi đứng trước:
 - + a (ca hát), ă (cắt đứt), â (cắt giầu)
 - + o (co quắp), ô (cô gái), ơ (cơ khí)
 - + u (cu Tèo), ư (cư trú)
 - + ai (cai quản), ao (cao xa), au (cau trầu)
 - + ay (cay đắng), âu (câu cá), ây (cây cối), oi (coi ngó)
 - + ôi (côi cút), ơi (cơi trầu)
 - + ua (cua ghe), ui (củi lửa), ư (cư gổ), ưi (khung cửi)
 - + ươ (cương quyết), ươ (cư mang)
 - + ươi (cười cợt), uôi (cuối cùng)
 - k khi đứng trước:

- + e (kẻ thù), ê (hạt kê)
- + i hay y (kí sự, ký sự)
- + eo (keo kiệt),
- + êu (kêu gào)
- + ia (kia kìa), iê (kiên quyết),
- + iu (kiu kiu - tiếng chó con kêu), iêu (kiêu sa);
- qu khi ứng trước:
 - + oa (qua loa-> qu + oa),
 - + oă (quăn tít -> qu + oăn), oe (que kem -> qu + oe), uy (vu quy -> qu + uy),
 - + uơ (quơ quào -> qu + uơ), uô (tổ quốc -> qu + uốc), uê (quê hương -> qu + uê),
 - + uâ (quân nhân -> qu + uân), oai (quai xách -> qu + oai), oay (quay tròn -> qu + oay), uây (quây quần -> qu + uây),
 - + uyê (chim quyên -> qu + uyên),
 - + uya (giày quya -> qu + uya - phiên âm tiếng Pháp cuir), oeo (chết queo -> qu + oeo)
 - + oao (quơ quào -> qu + oào)
 - + Luật viết lược chữ u hay chữ o khi gặp "qu + u..." hay "qu + o..." là cách giải thích cho các tranh cãi về
 - + qua = qu + a hay q + ua (hai cách này không đúng, "qua" cùng vần với "loa" trong "qua loa", thế nên qua
 - + = qu + oa.

4.4.2. Các kiểu gõ tiếng việt

Hiện nay có 3 cách gõ tiếng việt được sử dụng rộng rãi, đó là các kiểu gõ **VIQR**, **VNI** và **TELEX**, ngoài ra có thể sử dụng kiểu gõ **Auto** có thể gõ dấu tất cả các cách gõ dấu của 3 kiểu trên.

Các kiểu gõ chi tiết như sau:

Accents vs. Vowels Dấu với nguyên âm	Telex Input Method Cách gõ Telex	VNI Input Method Cách gõ VNI	VIQR Input Method Cách gõ VIQR
a circumflex - â	aa	a6	a^
e circumflex - ê	ee	e6	e^
o circumflex - ô	oo	o6	o^
a breve - ă	aw	a8	a(
o horn - ơ	ow	o7	o+
u horn - ư	uw	u7	u+
d stroke - đ	dd	d9	Dd
acute - sắc	s	1	'
grave - huyền	f	2	`
dot below - nặng	j	5	.
hook above - hỏi	r	3	?
tilde - ngã	x	4	~
remove diacritics - xóa dấu	z	0	-
Ví dụ: Tiếng Việt	Tieesng Vieejt	Tie61ng Vie65t	Tie^'ng Vie^.t

Để tiện xây dựng trong chương trình chỉ sử dụng quy định dấu như kiểu gõ VNI, các kí tự như dấu mũ, các thanh sẽ được biểu diễn thông qua các số từ 0 đến 9, các kiểu gõ còn lại sẽ được chuyển sang tương ứng với kiểu gõ này. Đối với mỗi kiểu gõ sẽ xây dựng một lớp đại diện cho kiểu gõ đó kế thừa từ interface InputMethod, lớp này sẽ thực hiện chuyển các kí tự dấu của kiểu gõ tương ứng sang các kí tự dấu của kiểu gõ VNI. Tất cả các lớp kế thừa InputMethod sẽ sử dụng chung phương thức getAccentMark ghi đè từ interface để xuất ra kí tự dấu tương ứng.

Cụ thể với kiểu gõ Telex ta có:

'S'	's'	accent = '1'
'F'	'f'	accent = '2'
'R'	'r'	accent = '3'
'X'	'x'	accent = '4'
'J'	'j'	accent = '5'
'A'	'a'	
'E'	'e'	

'O'	'o'	accent = '6'
'W'	'w'	accent = '7'
'D'	'd'	accent = '9'
'Z'	'z'	accent = '0'

Với kiểu gõ này kí tự dấu dù viết hoa hay không được coi như nhau, “S” hay “s” đều có nghĩa là dấu “sắc”.

Tương tự với kiểu gõ Viqr:

'\'	accent = '1'
'^'	accent = '2'
'?'	accent = '3'
'~'	accent = '4'
'.'	accent = '5'
'^'	accent = '6'
'*'	
'+'	accent = '7'
'('	accent = '9'
'_'	accent = '0'

4.4.3. Thuật toán để lập trình bộ gõ

Nguyên tắc chung của bộ gõ tiếng việt là sử dụng hook bàn phím – tức là chặn các thông điệp về bàn phím: trạng thái bàn phím, mã phím, các thông điệp,... Sau đó dùng các thuật giải riêng của mình để xử lý chuỗi nhập vào thành chuỗi tiếng việt tương ứng và xuất ra các thiết bị tương ứng.

Đối với Android ta bắt sự kiện phím thông qua hàm OnKey() của bàn phím. Hàm này xử lý sự kiện khi ấn một phím thuộc bàn phím cứng, các thông tin nhập vào bao gồm keycode – mã của phím được ấn, thông qua chỉ số này xác định phím được ấn là phím gì, thứ hai là keyevent là sự kiện phím bấm, ở đây là sự kiện phím được ấn ACTION_DOWN. Các thông tin này được chuyển qua xử lý trên lớp **VietKeyListener** qua hàm setKey để xuất ra kí tự tiếng việt.

Ngoài ra đối với các thiết bị không có bàn phím cứng mà sử dụng bàn phím ảo sẽ xử lý tương tự thông qua sự kiện addtext() của bàn phím ảo xây dựng kèm theo chương trình, và sử dụng hàm setkey3 của lớp **VietKeyListener** để xuất tiếng việt. Do cơ chế hoạt động có nhiều điểm khác nhau nên cần xây dựng nên các hàm khác nhau đối với từng loại bàn phím để chương trình có thể chạy được trên nhiều môi trường nhất.

Hoạt động của một chương trình gõ tiếng việt là thu nhận các phím do người dùng gõ vào, sau đó xử lý và cho ra chuỗi đã được xử lý. Cuối cùng là cần phải xuất đến thành phần đang chứa chuỗi đã xử lý đó. Nhưng nếu chỉ xuất thôi thì trong cửa sổ sẽ tồn tại cả chuỗi ban đầu và chuỗi đã xử lý. Vậy nên trước hết phải xóa các ký tự ban đầu đi. Sau đó mới xuất chuỗi đã xử lý sau.

Việc xóa chuỗi ban đầu tương đối phức tạp. Chúng ta sẽ căn cứ vào chuỗi bộ đệm mà ta đã nhận và thêm các ký tự từ bàn phím vào để tìm "từ cuối cùng trong chuỗi" để xử lý.

Ví dụ ta có được chuỗi bộ đệm như sau : “Câu lacj”. Vì từ “Câu” đã được xử lý rồi nên không phải chạy qua để kiểm tra nữa mà chỉ lấy từ cuối cùng thôi. Đó là chữ “lacj”. Có rất nhiều cách để lấy ra từ cuối cùng trong câu. Một cách là ta viết sẵn một hàm xác định vị trí đầu tiên của từ cuối cùng của một chuỗi. Sau đó mỗi lúc cần ta gọi hàm và truyền tham số để xác định được từ cuối cùng.

Trong chương trình, hàm xác định vị trí của từ tại vị trí hiện tại là hàm **getCurrentWord** của lớp **VietkeyListener**. Hàm này trả về chuỗi là từ đang tại vị trí hiện tại của cửa con trỏ, đồng thời xác định được vị trí bắt đầu và kết thúc của từ đó với các biến **start** và **end**.

Các biến đầu vào bao gồm pos là vị trí của con trỏ hiện tại và source là chuỗi đầu vào cần xác định từ tại vị trí của con trỏ. Ở ví dụ hàm trả về chuỗi “lacj” khi con trỏ nằm tại vị trí sau ký tự j là “lacj”, **start** là vị trí bắt đầu của từ này là 4 và **end** là 8. Sau đó dựa vào hai biến start và end này để tiến hành thay thế các ký tự nằm trong khoảng được xác định bằng các ký tự đã được đưa về dạng tiếng việt. Chuỗi trả về của hàm chính là chuỗi được đưa vào bộ đệm để xử lý. Sau đó tùy vào thủ tục cài đặt để xử lý chuỗi nguyên âm tương ứng.

Ví dụ: Ta xử lý dấu câu (kiểu Telex) thì mỗi khi ta thu nhận được một trong các ký tự "f,s,r,x,j" thì ta gọi hàm này. Trong hàm ta cần lấy ra chuỗi nguyên âm để xác định vị trí đặt dấu. Nếu không tìm thấy thì ta chỉ đặt tiếp ký tự nhận được vào bộ đệm. Nếu tìm thấy, trong ví dụ trên chữ hiện tại đang là chữ "a" và ta nhận được ký tự gõ là "j". Vì đây là 1 nguyên âm nên đặt dấu ngay tại vị trí của nguyên âm này. Vậy ta được chữ ạ. Ta phải thay chữ "a" trong chuỗi bộ đệm ban đầu bằng "ạ". Sau đó xác định số ký tự cần xóa trong cửa sổ Focus căn cứ vào chuỗi bộ đệm, để sau đó dùng một hàm hay thủ tục để xóa các ký tự ban đầu trong cửa sổ Focus. Cụ thể trong ví dụ trên - từ "lacj" và chuỗi bộ đệm là "Câu lacj". Chuỗi có độ dài 8 và vị trí tìm thấy chữ "a" là 6. Vậy số ký phím cần xóa bằng độ dài chuỗi bộ đệm trừ đi vị trí chữ "a" (vị trí chữ đầu tiên của chuỗi

nguyên âm tìm được). Sau khi cho ra chuỗi bộ đệm là "Câu lạc" và một chuỗi bộ đệm khác là "ạ" đồng thời ghép nốt các ký tự còn lại sau chữ "a" trong chuỗi bộ đệm vào chuỗi này. Cụ thể là "ạc". Bây giờ ta xóa chuỗi "acj" trong cửa sổ Focus và xuất chuỗi "ạc" đã xử lý. Thế là trong cửa sổ sẽ được chuỗi "Câu lạc".

Toàn bộ công việc nhận biết và thay thế chuỗi sẽ được thực hiện trong hàm **setkey()** của lớp **VietkeyListener**. Hàm này được gọi khi có sự kiện ấn phím xảy ra, nó thực hiện nhận và các ký tự và xuất ra các ký tự tiếng việt đã thay thế vào EditText của chương trình.

Đầu tiên cần lấy vào các biến đầu vào bao gồm: vị trí con trỏ hiện tại (lấy qua hàm **getSelectionStart()**), chuỗi đang có - doc (lấy qua hàm **getText()**), ký tự tại vị trí hiện tại (vị trí tại con trỏ của chuỗi đang có), ký tự nhập vào (lấy qua biến **keyCode**).

Sau đó tiến hành xử lý viết tắt, chương trình sẽ tìm kiếm những tổ hợp ký tự tương ứng trong bảng viết tắt và thay thế nó thành từ hoàn chỉnh ngay khi người dùng gõ vào. Ví dụ, nếu trong bảng viết tắt quy định “nhưng” được viết là “nu” thì khi gõ “nu” vào chương trình sẽ tự động thay thế “nu” thành “nhưng” trong đoạn văn bản hiển thị trên ô nhập.

Tiếp theo cần tiến hành lấy vị trí đặt dấu cho các bộ nguyên âm, nếu nguyên âm đơn thì dấu sẽ được đặt trực tiếp lên nguyên âm đó, nhưng nếu là nguyên âm đôi hay ba thì cần xác định vị trí đặt dấu cho nguyên âm đó. Và theo kiểu bỏ dấu hiện tại thì dấu được đặt kiểu “òà” thay vì “òa”,... nếu âm đầu là “gi” hoặc “qu” thì tự động để vị trí đặt dấu lên một tức là coi “gi” và “qu” là một âm đầu và không đặt dấu vào i và u của các âm đầu này.

Hàm **shiftAccent** xây dựng trong lớp **VietkeyInput** có tác dụng xác định vị trí của dấu sẽ được đánh trong từ tương ứng, với **curWord** là từ tại vị trí hiện tại và **keyChar** là ký tự được nhập vào.

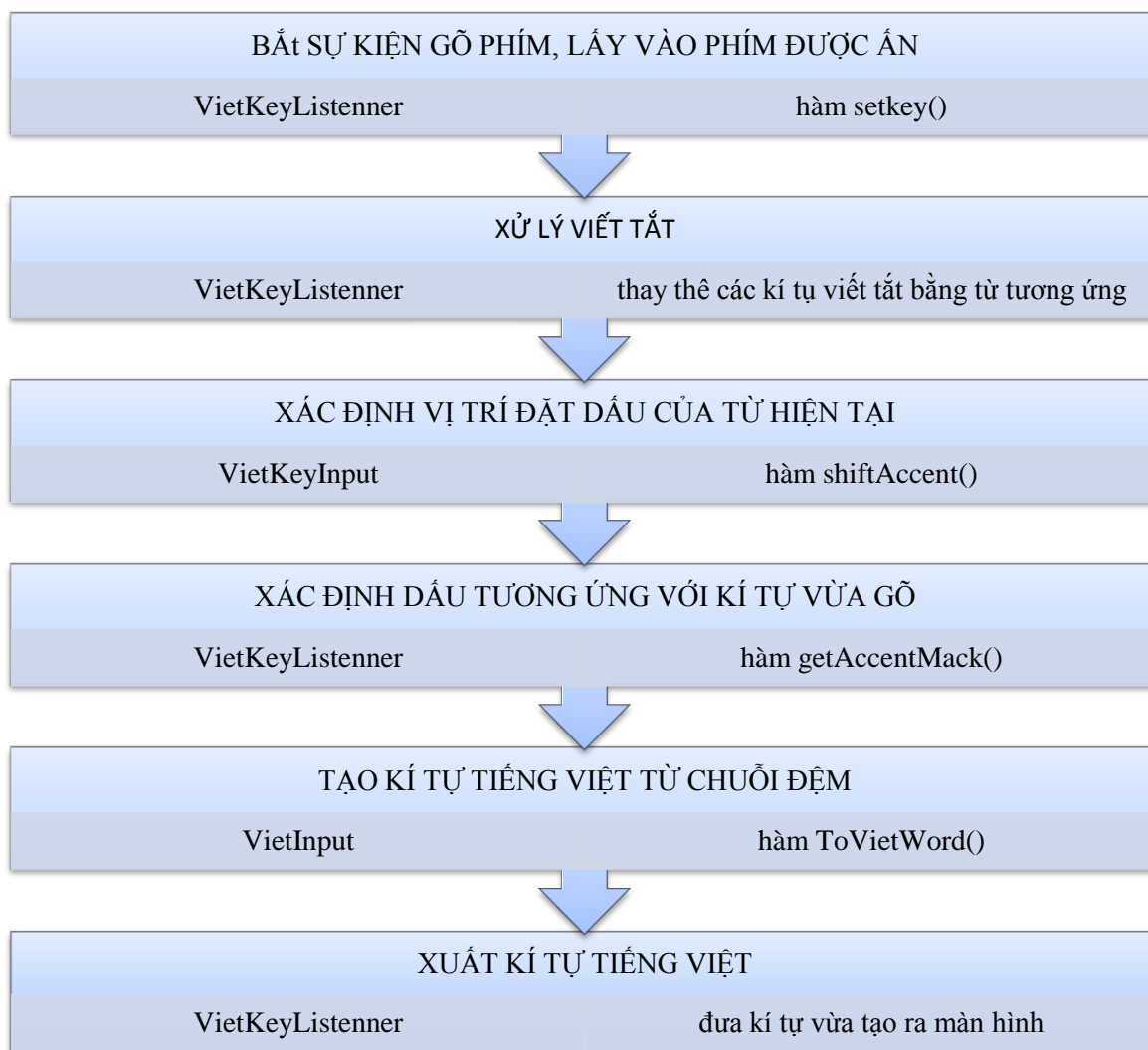
Cuối cùng là tiến hành đặt dấu cho nguyên âm dựa vào ký tự dấu được đưa vào. Việc tạo các ký tự dấu cho nguyên âm sẽ được thực hiện bởi hàm **ToVietWord** của lớp **VietkeyInput**, sau khi đã xử lý ta cần tiến hành thay thế tương ứng từ đệm thành từ đã được xử lý thêm dấu:

Lớp **Vietkeylistener** phụ trách xuất các ký tự ra EditText còn xử lý các ký tự nhập vào là do lớp **VietKeyInput** xử lý. Lớp **VietKeyInput** cung cấp các hàm

xây dựng kí tự tiếng việt, các hàm chính của lớp này bao gồm **shiftAccent** và **toVietword**,

Trong chương trình bộ gõ được xây dựng riêng bên trong phân lớp để nhập tiếng việt vào chứ không được xây dựng bên ngoài, điều đó sẽ đảm bảo chương trình sẽ chạy được ngay sau khi cài đặt mà không cần cài đặt thêm bất cứ thành phần nào khác kèm theo. Trên thực tế cũng có một số bộ gõ tiếng việt được xây dựng nhưng có thể các bộ gõ đó sẽ không chạy được ở các phiên bản Android cũ, cách trên sẽ được giải quyết vấn đề này. Chương trình được xây dựng trên nhân Android thấp nhất là 1.5 đảm bảo chương trình có thể chạy được trên hầu hết các thiết bị chạy nền Android.

Ngoài ra chương trình có một bàn phím ảo riêng hỗ trợ nhập tiếng việt, có thể cài đặt cho các thiết bị khó có thể lắp đặt bàn phím cứng hoặc cho những người dùng không muốn sử dụng bàn phím cứng, đồng thời đảm bảo tiêu chí chương trình có thể chạy mà không cần cài thêm chương trình bàn phím ảo vào nhân Linux đảm bảo tài nguyên sử dụng.



Hình 37 - lưu đồ xử lý tiếng việt

4.5. Giao tiếp với server Isolar

4.5.1. Gửi yêu cầu đến server

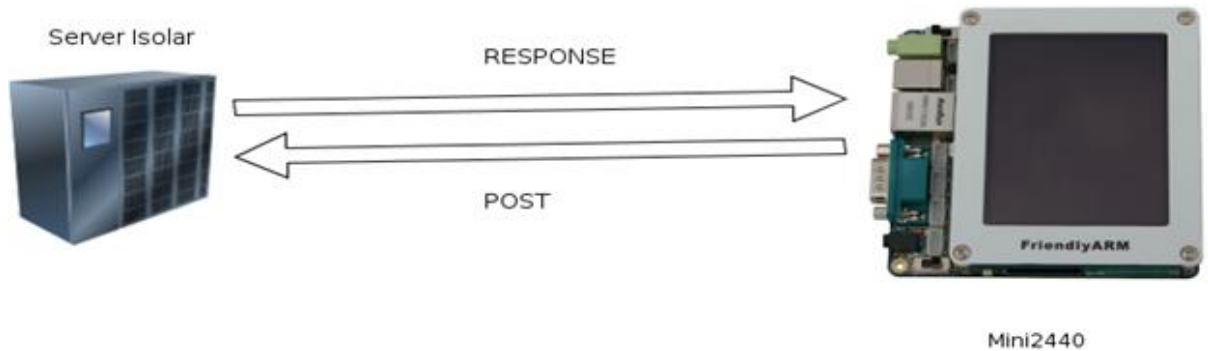
Sau khi nhập được văn bản tiếng việt, chúng ta phải gửi văn bản này đến server để yêu cầu chuyển thành file âm thanh.

Quá trình yêu cầu thực chất là gửi gói tin POST đến server với dữ liệu yêu cầu là đoạn văn bản tiếng việt mà người dùng đã nhập:

```
Data_Sent = "voice=male1&SSinput=" + data + "&formSubmit=Submit";
```

Data_sent là dữ liệu sẽ được gửi đi cùng với gói tin POST và data là văn bản tiếng việt mà người sử dụng nhập từ bàn phím

Mô hình như sau:



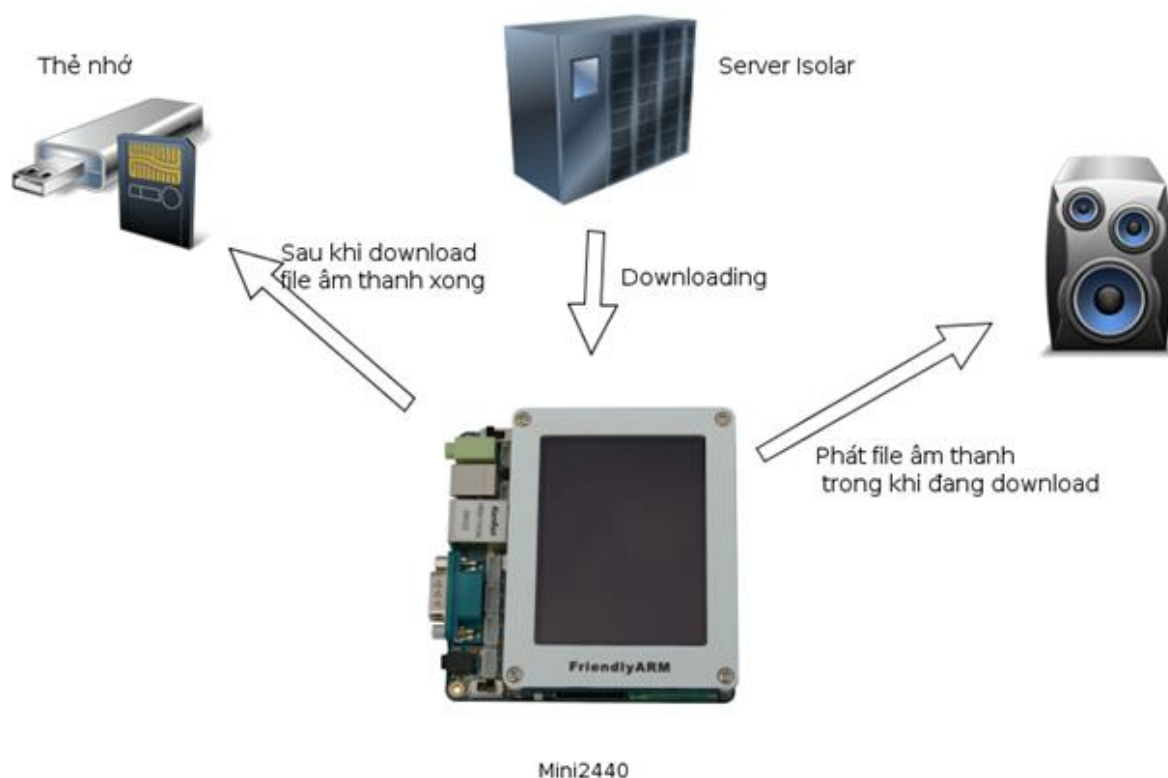
Hình 38 - Mô hình kết nối xử lý tiếng nói

Sau khi gửi gói tin POST, server sẽ gửi về một gói tin phản hồi, trong đó sẽ có link chứa đường dẫn của file âm thanh tiếng việt, nhiệm vụ tiếp theo là phải tải file âm thanh đó về và phát ra loa

4.5.2. Stream file âm thanh

Với những đoạn văn bản nhỏ, khi chuyển sang tiếng nói, dung lượng có thể sẽ nhỏ và không đáng kể, khi đó chương trình sẽ có thể download về hết và phát ra loa bình thường. Nhưng đối với đoạn văn bản lớn thì khả năng sau khi người sử dụng ấn nút yêu cầu đến server cho đến khi download một file âm thanh lớn về sẽ mất rất nhiều thời gian, vì vậy, chương trình sẽ stream file âm thanh về và chạy trong lúc đang download

Mô hình như sau:



Hình 39 - Mô hình hệ thống chương trình

Từ server, chương trình download file âm thanh về buffer. Khi buffer đủ số lượng nhất định, chương trình sẽ copy buffer sang chỗ khác và chạy file copy đó. Đến khi chạy hết file copy, tiếp tục copy file buffer đã download về sang file khác và tua đến đoạn file copy trước đã chạy. Khi đã play hết cả file, copy file đã download đầy đủ ra thẻ nhớ và xóa hết các file buffer còn lại. Giải thuật này gọi là double-buffer

Sở dĩ phải sử dụng giải thuật này vì những file đang sử dụng để lưu dữ liệu download từ server, nếu chạy luôn file này thì sẽ gây xung đột hệ thống, vì vậy, chương trình sẽ làm theo hướng an toàn hơn là lưu ra file buffer copy khác để chạy.

4.6. Chia sẻ mạng của Ubuntu qua dây Ethernet

Chia sẻ mạng (Internet Connection Sharing - ICS) là khả năng mà một máy tính dùng mạng có thể chia sẻ băng thông mạng với một hay nhiều máy tính khác. Để được như vậy, máy tính có kết nối mạng phải được cấu hình như một Cổng Internet (Internet Gateway). Máy tính được chia sẻ sẽ kết nối mạng trực tiếp thông qua cổng internet đó.

4.6.1. Mô hình hoạt động



Hình 40 - chia sẻ mạng qua dây ethernet

Để chia sẻ qua dây ethernet, bên phía máy ubuntu, địa chỉ IP phải được đặt là tĩnh:

```
sudo ip addr add 192.168.1.5/24 dev eth0
```

Hoặc :

```
sudo ifconfig eth0 192.168.1.5 netmask 255.255.255.0 up
```

Sau đó, ta phải cấu hình bảng ip bên máy chia sẻ để gói tin có thể từ máy được chia sẻ đi thẳng qua Ubuntu gateway

4.6.2. Cấu hình NAT

```
sudo iptables -A FORWARD -o eth0 -i eth1 -s 192.168.0.0/24 -m  
conntrack --ctstate NEW -j ACCEPT
```

```
sudo iptables -A FORWARD -m conntrack --ctstate  
ESTABLISHED,RELATED -j ACCEPT
```

```
sudo iptables -A POSTROUTING -t nat -j MASQUERADE
```


Lệnh thứ nhất và thứ hai cho phép chuyển tiếp gói tin, lệnh thứ ba giúp dịch địa chỉ mạng(NAT),khi đó,gói tin từ máy được chia sẻ sẽ được đi thẳng qua Gateway trung gian và đến thẳng địa chỉ được yêu cầu,sau đó lúc gói tin đi về cũng đi theo đường ngược lại.

Lưu lại bảng IP để lần sau ta không phải cấu hình lại gateway nữa:

```
sudo iptables-save | sudo tee /etc/iptables.sav
```

Sửa file /etc/rc.local và thêm vào trước dòng "exit 0" :

```
iptables-restore < /etc/iptables.sav
```

4.6.3. Cấu hình routing

Cấu hình cổng gateway để có thể truyền dẫn gói tin giữa hai cổng bằng cách:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Sửa file /etc/sysctl.conf, thêm vào các dòng sau:

```
net.ipv4.conf.default.forwarding=1  
net.ipv4.conf.all.forwarding=1
```

Khi đó là đã xong cấu hình bên phía gateway.

4.6.4. Cấu hình bên máy nhận(mini2440)

Bên phía mini2440,ta cần cấu hình địa chỉ ip tĩnh cho cổng ethernet và cấu hình để cho các gói tin đi ra ngoài mạng qua cổng này
Cấu hình địa chỉ ip tĩnh cho mini2440,trên màn hình boot:

```
ifconfig eth0 192.168.1.230 netmask 255.255.255.0 up
```

Cấu hình default gateway để các gói tin ra ngoài mạng sẽ đi qua cổng máy chia sẻ

```
route add default gw 192.168.1.5
```

Và kết quả:



Hình 41 - kết quả chia sẻ mạng

CHƯƠNG V

KẾT LUẬN

Với mức độ phát triển hệ thống nhúng và hệ điều hành Android hiện nay ở Việt Nam, các ứng dụng cho lĩnh vực này sẽ ngày càng phát triển rộng rãi, và ứng dụng TTS trên hệ thống nhúng Android sẽ được sử dụng rộng rãi. Ứng dụng cho chúng em xây dựng sẽ là bước mở đầu cho chúng em tham gia vào lĩnh vực mới mẻ đầy tiềm năng này

Mặc dù đã rất cố gắng nghiên cứu, nhưng do chưa có kinh nghiệm trong việc thiết kế và phát triển phần mềm nhúng và cũng do lĩnh vực nghiên cứu còn khá lạ lẫm nên chắc chắn không tránh khỏi những sai sót, mong các thầy cô và các bạn đóng góp ý kiến để em rút ra được các kinh nghiệm cho công việc sau này.

- **Hướng phát triển:**

- Do chưa có phần cứng cụ thể nên ứng dụng được xây dựng trên nền tảng KIT Mini2440, trong tương lai có thể xây dựng một hệ thống nhúng chuyên biệt như một thiết bị cầm tay nhỏ gọn để có thể di chuyển dễ dàng cho các công việc cần đến TTS tiếng việt.
- Tiếp tục hoàn thiện và sửa chữa những lỗi ứng dụng mắc phải khi đưa vào thử nghiệm và hoạt động.
- Có thể xây dựng một bộ TTS cỡ nhỏ trực tiếp trên thiết bị nhúng cho các thành phần tiếng việt cơ bản thì di chuyển đến những khu vực không kết nối được mạng đảm bảo hoạt động thông suốt của thiết bị tuy nhiên chất lượng có thể giảm sút đôi chút.

DANH MỤC TÀI LIỆU THAM KHẢO

1. **Vietandroid.com**; *Hướng dẫn lập trình cơ bản với Android*.
2. **www.friendlyarm.net**; *Hướng dẫn cơ bản về KIT mini2440*.
3. **Google Developer**; *Google Android Project*;
4. **Mark .L Murphy**; *BeginningAndroid*.
5. **Reto Meier**; *Professional Android Application Development*; US 2008.
6. **Wei Meng Lee**; *Beginning Android Application Development*.