**Tên đề tài:**

**Thiết kế hệ thống thu phát tín hiệu băng tần tiếng nói dựa trên nền tảng phát triển ARM Cortex-M4F**

**Tóm tắt**

Thiết bị thu phát tín hiệu băng tần tiếng nói là thiết bị ghi lại tín hiệu tiếng nói ở dải tần số từ 300-3400 Hz, lưu trữ và phát lại đã lưu một […]. Cùng với sự phát triển của công nghệ bán dẫn, các thiết thu phát tín hiệu đã […]

Hệ thống thu phát số tín hiệu băng tần tiếng nói (DVR) về cơ bản bao gồm: Mạch lọc chống chồng phổ đầu vào (Input Aliasing filter), mạch xử lí số tín hiệu, thiết bị lưu trữ, và mạch tái tạo tín hiệu ở đầu ra. Các […]

Yêu cầu cơ bản của mạch lọc chống chống phổ của DVR đó là loại bỏ các thành phần tần số cao hơn tần số Nyquist [cite] của mạch chuyển đổi tín hiệu tương tự - số (ADC). Với tốc độ lấy mẫu 8000 mẫu một giây, mạch IAF phải có dải tần chuyển đổi hẹp 600Hz. Mạch lọc tích cực sử dụng opamp và các linh kiện thụ động khác để thiết kế và thi công với điều kiện trên không dễ dàng [fix]. Thay vào đó, giải pháp thiết kế trong báo cáo này áp dụng kĩ thuật xử lí tín hiệu đa tốc độ lấy mẫu (multirate signal processing) để thu được kết quả mong muốn như yêu cầu đề ra với mạch IAF đầu vào có dải tần chuyển đổi rộng hơn, cho phép thiết kế và thi công mạch dễ dàng hơn.

Ngoài ra, thiết bị […] vừa lấy mẫu dữ liệu tốc độ cao và đồng thời xử lý chúng trước khi ghi vào thiết bị lưu trữ. Một bộ đệm dữ liệu theo phương thức Vào trước – Ra trước (FIFO) được thực hiện nhằm đáp ứng yêu cầu này. […]

Fuck fuck fuck fuck […]

More fucks […]

Với những kết quả đạt được, blah blah […]

[Danh sách hình 5](#_Toc388968499)

[1. GIỚI THIỆU 6](#_Toc388968500)

[1.1. Cơ bản về thiết kế máy thu phát tín hiệu băng tần tiếng nói: 6](#_Toc388968501)

[1.2. Tổng quan về số hóa tín hiệu băng tần tiếng nói 6](#_Toc388968502)

[1.2.1. Tín hiệu băng tần tiếng nói: 6](#_Toc388968503)

[1.2.2. Lấy mẫu tín hiệu số tín hiệu băng tần tiếng nói. 7](#_Toc388968504)

[1.2.3. Lấy mẫu đa tốc độ: 7](#_Toc388968505)

[1.3. Tổng quan về lọc thông thấp 8](#_Toc388968506)

[1.3.1. Giới thiệu về bộ lọc số lọc thông thấp: 8](#_Toc388968507)

[1.3.2. Cơ bản về thiết kế bộ lọc số lọc thông thấp có Đáp ứng xung chiều dài hữu hạn (FIR) 8](#_Toc388968508)

[1.3.3. Ưu điểm của các bộ lọc số: 10](#_Toc388968509)

[1.4. Nền tảng phát triển ARM Cortex-M4F 10](#_Toc388968510)

[1.4.1. ARM Cortex-M4F[cortex m4 dgug] 10](#_Toc388968511)

[1.4.2. Đơn vị dấu phẩy động – Floating Point Unit (FPU) [cm4 dgug][tm4c123] 11](#_Toc388968512)

[1.4.3. Chuẩn giao tiếp phần mềm vi điều khiển Cortex (Cortex Microntroller Software Interface Standard – CMSIS) [cmsis] 12](#_Toc388968513)

[1.5. Kết luận 13](#_Toc388968514)

[2. THIẾT KẾ PHẦN CỨNG 14](#_Toc388968515)

[2.1. Sơ đồ khối thiết kế phần cứng [fix] 14](#_Toc388968516)

[2.2. Mạch nguồn 14](#_Toc388968517)

[2.3. Mạch tiền khuếch đại và mạch lọc chống chồng phổ IAF 15](#_Toc388968518)

[2.3.1. Electret Condenser Microphone (ECM) 15](#_Toc388968519)

[2.3.2. Mạch tiền khuếch đại và mạch lọc chống chồng phổ IAF 16](#_Toc388968520)

[2.3.3. Mạch tái tạo tín hiệu đầu ra: 17](#_Toc388968521)

[2.4. Mạch điều khiển chính - Kit phát triển Tiva C Launchpad nền tảng ARM Cortex-M4F 18](#_Toc388968522)

[3. THIẾT KẾ PHẦN MỀM 20](#_Toc388968523)

[3.1. Tổng quan: 20](#_Toc388968524)

[3.2. Kỹ thuật xử lí tín hiệu đa tốc độ lấy mẫu (Multirate Signal Processing) 20](#_Toc388968525)

[3.2.1. Hạn chế trong việc thiết kế bộ lọc tích cực cho IAF 20](#_Toc388968526)

[3.2.2. Cơ sở kĩ thuật xử lí đa tốc độ lấy mẫu: 21](#_Toc388968527)

[3.2.3. Thuật toán cho bộ Decimator: 22](#_Toc388968528)

[3.3. Hệ thống quản lí file theo chuẩn FAT 22](#_Toc388968529)

[3.3.1. Đặc điểm của FAT 22](#_Toc388968530)

[3.3.2. Thư viện FatFs[ref link fatfs] 23](#_Toc388968531)

[3.3.3. Ứng dụng thư viện FatFS để triển khai FAT32 trên thẻ nhớ SD 24](#_Toc388968532)

[3.4. Bộ đệm dữ liệu kiểu Vào trước ra trước (FIFO) 25](#_Toc388968533)

[3.4.1. Sự cần thiết của bộ đệm dữ liệu kiểu Vào trước ra trước (FIFO) 25](#_Toc388968534)

[3.4.2. Thiết kế bộ đệm FIFO [cần sửa lại title] 26](#_Toc388968535)

[3.4.3. Bộ đệm FIFO với μDMA cơ chế Ping-Pong 28](#_Toc388968536)

[3.5. Bộ lọc thông thấp Đáp ứng xung chiều dài hữu hạn (FIR) 30](#_Toc388968537)

[3.5.1. Thiết kế nhân bộ lọc h[i] với Scipy: 30](#_Toc388968538)

[3.5.2. Mô phỏng kết quả bộ lọc trên Scipy 31](#_Toc388968539)

[3.5.3. Nhúng bộ lọc vào phần mềm điều khiển 32](#_Toc388968540)

[4. KẾT QUẢ THỰC NGHIỆM 34](#_Toc388968541)

[4.1. Phần cứng và giao diện phần mềm: 34](#_Toc388968542)

[4.2. Các kết quả 36](#_Toc388968543)

[PHỤ LỤC 1 – NHÂN BỘ LỌC THÔNG THẤP FIR TRONG THIẾT KẾ 37](#_Toc388968544)

# Danh sách hình

[Hình 1 – Sơ đồ khối một thiết kế điển hình máy thu phát tín hiệu băng tần tiếng nói 7](#_Toc388918405)

[Hình 2 Đáp ứng tần số cơ bản của một bộ lọc số 9](#_Toc388918406)

[Hình 3 - Bộ lọc Đáp ứng xung chiều dài hữu hạn FIR 10](#_Toc388918407)

[Hình 4 - Đáp ứng xung bộ lọc FIR với M = 66 11](#_Toc388918408)

[Hình 5 - Đáp ứng tần số của bộ lọc thông thấp FIR 40001 điểm 11](#_Toc388918409)

[Hình 6 Cấu trúc cơ bản của kiến trúc ARM Cortex-M4F 12](#_Toc388918410)

[Hình 7 Cấu trúc của CMSIS [ref – cmsis] 14](#_Toc388918411)

[Hình 8 Sơ đồ khối thiết kế phần cứng 15](#_Toc388918412)

[Hình 9 Sơ đồ nguyên lí mạch nguồn 16](#_Toc388918413)

[Hình 10 - Cấu hình cơ bản của ECM 16](#_Toc388918414)

[Hình 11 - Mạch tiền khuếch đại và lọc chống chồng phổ 17](#_Toc388918415)

[Hình 12 - Đáp ứng tần số mạch lọc chống chồng phổ IAF 18](#_Toc388918416)

[Hình 13 Mạch tái tạo tín hiệu đầu ra 19](#_Toc388918417)

[Hình 14 – Đáp ứng bộ lọc thông thấp mạch tái tạo tín hiệu đầu ra 19](#_Toc388918418)

[Hình 15 – Sơ đồ khối Tiva C Launchpad 20](#_Toc388918419)

[Hình 16 – Sơ đồ kết nối các chân tín hiệu của Tiva C Launchpad trên mạch 21](#_Toc388918420)

[Hình 17 – Kĩ thuật xử lí đa tốc độ lấy mẫu 23](file:///E:\DOAN_SPRING_2014\WRITING%20THESIS\draft.docx#_Toc388918421)

[Hình 18- Cấu trúc của một phân vùng sử dụng FAT 24](#_Toc388918422)

[Hình 19 – Ví trí lớp thư viện FatFs trong các lớp của phần mềm 26](#_Toc388918423)

[Hình 20 – Cấu hình cơ bản của hệ thống nhúng với FatFs 26](#_Toc388918424)

[Hình 21 – Cấu trúc bộ đệm FIFO với 6 phần tử 28](#_Toc388918425)

[Hình 22 – Sơ đồ thuật toán yêu cầu phần tử trống từ bộ đệm FIFO 29](#_Toc388918426)

[Hình 23 – Sơ đồ thuật toán yêu cầu phần tử chứa dữ liệu từ bộ đệm FIFO 30](#_Toc388918427)

[Hình 24 – 𝜇DMA trạng thái sử dụng bộ đệm Ping 31](#_Toc388918428)

[Hình 25 – 𝜇DMA trạng thái sử dụng bộ đệm Pong 31](#_Toc388918429)

[Hình 26 – 𝜇DMA trở về trạng thái sử dụng bộ đệm Ping 32](#_Toc388918430)

[Hình 27 – Đáp ứng tần số của bộ lọc FIR 215 điểm 34](#_Toc388918431)

[Hình 28 – Bộ lọc với tín hiệu tổng hợp 100 Hz và 4100 Hz 34](#_Toc388918432)

[Hình 29 - Bộ lọc với tín hiệu tổng hợp 3200 Hz và 4050 Hz 35](#_Toc388918433)

[Hình 30 – Mạch kết nối với Kit Tiva C Launchpad 37](#_Toc388918434)

[Hình 31 – Kết nối mạch với Kit Tiva C Launchpad 38](#_Toc388918435)

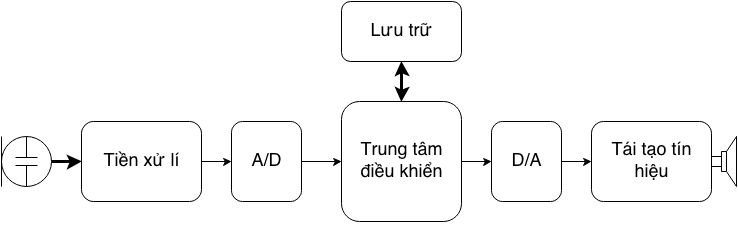
[Hình 32 – Giao diện dòng lệnh qua UART 39](#_Toc388918436)

[Hình 33 – Nhân bộ lọc thông thấp FIR trong thiết kế 41](#_Toc388918437)

# GIỚI THIỆU

* 1. Cơ bản về thiết kế máy thu phát tín hiệu băng tần tiếng nói:

Các thiết kế máy thu phát tín hiệu băng tần tiếng nói về cơ bản bao gồm thành phần xử lí trung tâm chịu trách nhiệm lấy mẫu tín hiệu đầu vào, xử lí và lưu trữ tín hiệu đó và cuối là tái tạo lại tín hiệu ở đầu ra. Dưới đây là sơ đồ khối cho một thiết kế điển hình:



Hình 1 – Sơ đồ khối một thiết kế điển hình máy thu phát tín hiệu băng tần tiếng nói

Tín hiệu tiếng nói được đưa qua bộ tiền xử lí, chịu trách nhiệm xử lí tín hiệu ban đầu như khuếch đại, lọc nhiễu, lọc chống hiện tượng chồng phổ tín hiệu (anti-aliasing). Tín hiệu tiếp tục được Trung tâm điều khiển lấy mẫu bằng bộ chuyển đổi tín hiệu tương tự - số ADC, xử lí và đưa vào thiết bị lưu trữ dưới dạng số. Khi có yêu cầu phát lại tín hiệu, Trung tâm điều khiển sẽ lấy tín hiệu số ở thiết bị lưu trữ và đưa đến bộ phận tái tạo tín hiệu thông qua bộ chuyển đổi tín hiệu số - tương tự DAC. Bộ tái tạo tín hiệu sẽ tại tạo lại tín hiệu ban đầu và đưa ra thiết bị nghe.

* 1. Tổng quan về số hóa tín hiệu băng tần tiếng nói
     1. Tín hiệu băng tần tiếng nói:

Tiếng nói của người lớn thông thường nằm ở tần số gốc là 120 Hz đối với nam và 210 Hz đối với nữ [cite - f0\_m&f]. Tuy nhiên, khi thực hiện truyền tín hiệu tiếng nói, nếu chỉ cung cấp băng tần vừa đủ, ví dụ 250 Hz, để truyền đi, khả năng nhận biết được giọng sẽ rất kém [Human Speech Spectrum]. Để nghe được, việc cần thiết là phải truyền thêm cả các hài của tần số cơ bản.

Trong công nghiệp truyền tín hiệu thoại, tín hiệu tiếng nói được truyền đi có băng thông 4000 Hz, trong đó đảm bảo các thành phần dưới 3400 Hz sẽ được truyền đi đầy đủ [B. P. Lathi modern digital and analog.]. Dải 600 Hz còn lại được dành ra cho vùng chuyển tiếp của bộ lọc tín hiệu.

Trong nhiều trường hợp, có thể lược bỏ cả các thành phần tần số từ DC đến 300 Hz của tín hiệu thoại. Lí do ở đây là một số thiết bị âm thanh công suất thấp có khả năng đáp ứng vùng tần số này rất kém. Việc lược bỏ đi các thành phần tần số từ DC đến 300 Hz không thay đổi khả năng nhận biết âm thanh thoại, hiệu ứng mất tần số gốc[subton.html#c2] cho thấy các tín hiệu bị loại bỏ thành phần tần số gốc vẫn cho cảm giác về cao độ của tần số đó.

* + 1. Lấy mẫu tín hiệu số tín hiệu băng tần tiếng nói.

Để thực hiện xử lí và lưu trữ tín hiệu số, tín hiệu tiếng nói ban đầu phải được số hóa.

Quá trình lấy mẫu tín hiệu phải tuân theo Định lí Nyquist – Shannon về lấy mẫu tín hiệu để tín hiệu ban đầu có thể được tái tạo lại hoàn chỉnh. Theo đó, nếu tần số cao nhất của tín hiệu là , tần số lấy mẫu là thì:

Như vậy, với băng thông tín hiệu là 4000 Hz, tần số lấy mẫu tối thiểu đối với tín hiệu băng tần tiếng nói là 8000 Hz.

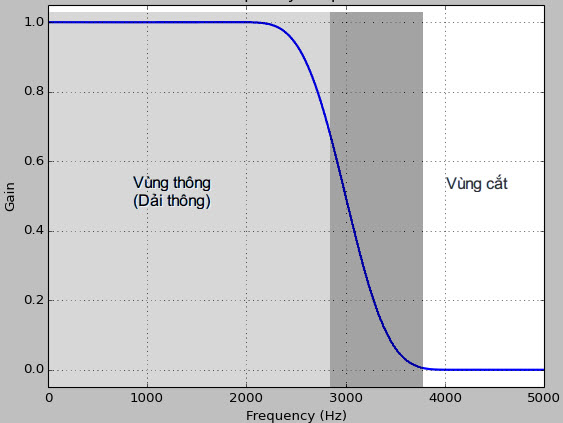
* + 3. Lấy mẫu đa tốc độ:

Một hệ thống lấy mẫu đa tốc độ (multirate system) cho phép tín hiệu lấy mẫu ở tốc độ ban đầu có khả năng tái lấy mẫu (resampling) với một tốc độ lấy mẫu khác. Lấy mẫu đa tốc độ là giải pháp linh động khi các thành phần khác nhau trong hệ thống yêu cầu xử lí tín hiệu ở tốc độ lấy mẫu khác nhau.

Có hai loại lấy mẫu đa tốc độ:

* Decimation: hạ tốc độ lấy mẫu. Tín hiệu ra có tốc độ lấy mẫu bằng số nguyên K lần so với tín hiệu vào.
* Interpolation: tăng tốc độ lấy mẫu. Tín hiệu ra có tốc độ lấy mẫu bằng nghịch đảo số nguyên K lần so với tín hiệu vào.

* 1. Tổng quan về lọc thông thấp
     1. Giới thiệu về bộ lọc số lọc thông thấp:



Hình 2 Đáp ứng tần số cơ bản của một bộ lọc số

Một bộ lọc số lọc thông thấp về cơ bản có ba vùng:

* Vùng thông (dải thông): vùng cho phép các thành phần tín hiệu có tần số bé hơn hoặc bằng tần số cắt đi qua. Tần số cắt là tần số mà tại đó độ lợi là về biên độ hay -3dB về công suất.
* Vùng chuyển tiếp: vùng chuyển giữa vùng thông sang vùng cắt. Độ lợi bé dần theo chiều tăng tần số cho đến khi đạt giá trị độ lợi vùng cắt.
* Vùng cắt: các thành phần tín hiệu có tần số trong vùng này đều bị loại bỏ.

Theo kiểu đáp ứng xung, có hai loại bộ lọc số lọc thông thấp là bộ lọc đáp ứng xung chiều dài hữu hạn (FIR) và bộ lọc đáp ứng xung chiều dài vô hạn (IIR). Về mặt tốc độ xử lí, các bộ lọc đáp ứng xung chiều dài vô hạn cho tốc độ cao hơn so với FIR. Tuy nhiên bản chất của IIR yêu cầu tính hồi tiếp, do đó bộ lọc IIR có khả năng không ổn định dẫn đến việc thiết kế phức tạp hơn so với FIR.

* + 1. Cơ bản về thiết kế bộ lọc số lọc thông thấp có Đáp ứng xung chiều dài hữu hạn (FIR)

Một bộ lọc số kiểu Đáp ứng xung chiều dài hữu hạn (Finite Impulse Response – FIR) luôn luôn ổn định. Tín hiệu đầu vào đi qua qua bộ lọc số FIR là hệ thống tuyến tính bất biến theo thời gian (LTI) sẽ được tích chập với đáp ứng xung của bộ lọc (nhân của bộ lọc) để cho tín hiệu đầu ra. Hình dưới đây mô tả điều đó:

Bộ lọc FIR

Hình 3 - Bộ lọc Đáp ứng xung chiều dài hữu hạn FIR

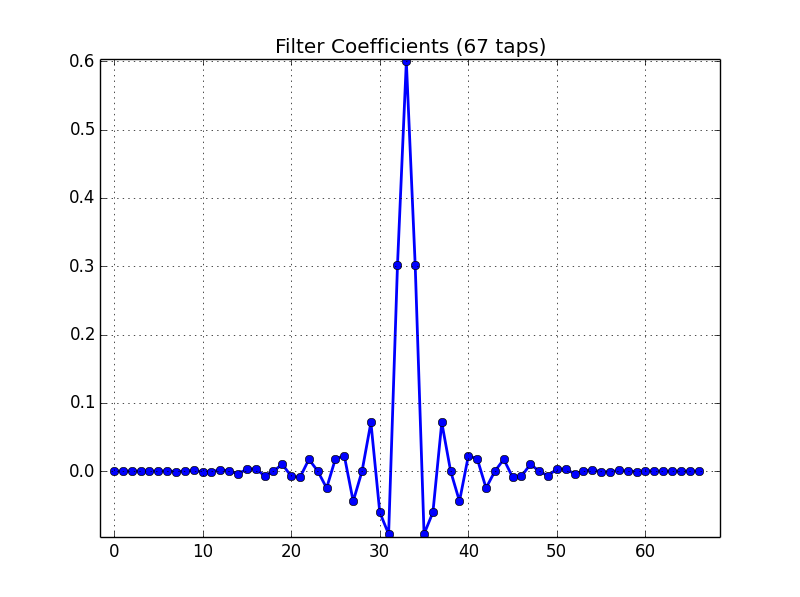
Thiết kế một bộ lọc FIR về cơ bản chính là tìm đáp ứng xung h[i] cho bộ lọc đó. Theo [ref scientist], chiều dài đáp ứng xung của bộ lọc phụ thuộc vào độ rộng vùng chuyển tiếp. Độ rộng vùng chuyển tiếp càng lớn thì chiều dài tối thiểu cần thiết càng nhỏ và ngược lại. Mối quan hệ trên có thể được diễn tả như sau [ref scientist]:

Với số nguyên M là chiều dài đáp ứng xung của bộ lọc và B là độ rộng vùng chuyển tiếp được tính theo tỉ lệ so với tốc độ lấy mẫu.

Ví dụ: Tốc độ lấy mẫu tín hiệu là 10 KHz, độ rộng vùng chuyển tiếp là 600 Hz thì B là 0.06. Khi đó:

Áp dụng cửa sổ Blackman, các giá trị của h[i] được tính theo công thức sau[ref scientist]:

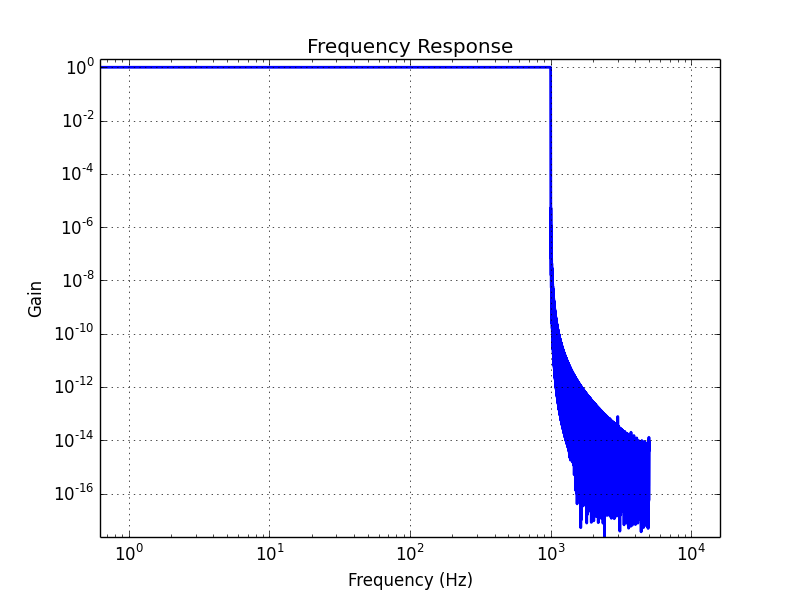
Tính toán với Scipy cho kết quả sau:



Hình 4 - Đáp ứng xung bộ lọc FIR với M = 66

* + 1. Ưu điểm của các bộ lọc số:

Các bộ lọc số được sử dụng với hai mục đích chủ yếu[ref the scientist]: (1) Tách các thành phần tín hiệu của một tín hiệu tổng họp, và (2) Khôi phục lại tín hiệu bị méo do ảnh hưởng nào đó. Bộ lọc tương tự cũng có thể được dùng cho những tác vụ như trên. Tuy nhiên, sử dụng các bộ lọc số cho những kết quả tốt hơn rất nhiều.

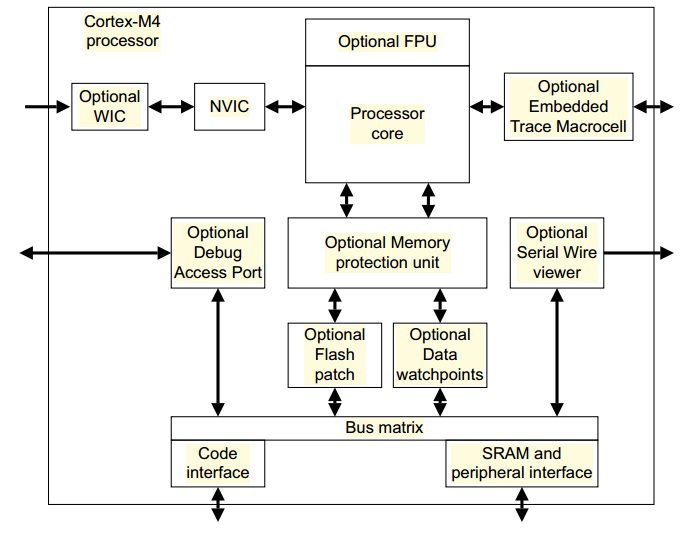


Hình 5 - Đáp ứng tần số của bộ lọc thông thấp FIR 40001 điểm

Ở trên đáp ứng của bộ lọc thông thấp FIR 40001 điểm, có độ lợi là 1 từ DC đến 1000 Hz và độ lợi bé hơn 10-5 từ 1001 Hz trở đi. Bộ lọc được thiết kế sử dụng Scipy [ref đến Scipy]. Một đáp ứng như vậy gần như là điều không thể đối với các bộ lọc tương tự.

* 1. Nền tảng phát triển ARM Cortex-M4F
     1. ARM Cortex-M4F[cortex m4 dgug]

ARM Cortex-M4F là nhân xử lí ARM Cortex-M4 với sự bổ sung của Đơn vị dấu phẩy động (FPU). Nhân xử lí ARM Cortex-M4 là nhân 32 bit được thiết kế để tích hợp vào các vi điều khiển. [need-to-fix]



Hình 6 - Cấu trúc cơ bản của kiến trúc ARM Cortex-M4F

Các vi điều khiển tích hợp nhân ARM Cortex-M4 vào bên trong tận dụng tốt các đặc điểm sau của nhân nói trên[cortex m4 dgug]:

* Năng lực xử lí tốt đi kèm với khả năng hỗ trợ ngắt nhanh.
* Cung cấp hệ thống gỡ lỗi mạnh hơn với các tính năng theo dõi.
* Cung cấp các chế độ siêu tiết kiệm điện (ULP).
* Tính năng bảo mật với Đơn vị bảo vệ bộ nhớ (MPU).
  + 1. Đơn vị dấu phẩy động – Floating Point Unit (FPU) [cm4 dgug][tm4c123]

FPU của ARM Cortex-M4F được dựa trên kiến trúc mở rộng FPv4-SP. FPU hỗ trợ đầy đủ các thao tác cộng, trừ, nhân, chia, phép nhân và tích lũy, và phép căn bậc hai cho số thập phân có độ chính xác đơn (single-precision). Nó còn hỗ trợ các thao tác chuyển đổi giữa số nguyên và số thập phân, và các chỉ lệnh trên các hằng số số thập phân.

FPU cung cấp khả năng tính toán hợp quy theo tiêu chuẩn *ANSI/IEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*.

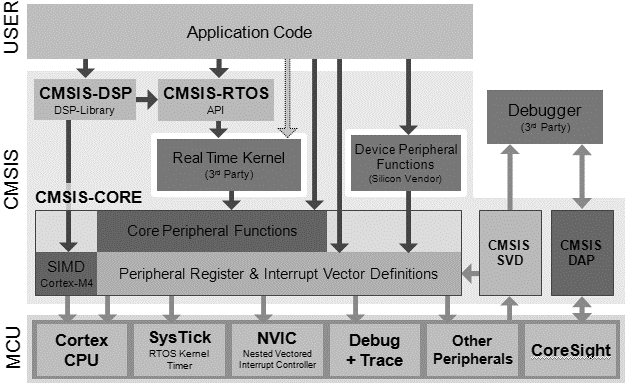
Đặc điểm của FPU:

* Các chỉ lệnh độ rộng 32 bit cho số thập phân độ chính xác đơn (kiểu C float) các thao tác xử lí dữ liệu.
* Kết hợp các chỉ lệnh nhân và tích lũy để tăng độ chính xác.
* Tăng tốc phần cứng cho các thao tác chuyển đổi, cộng, trừ, nhân với tùy chọn tích lũy, chia và căn bậc hai.
* 32 thanh ghi 32 bit độ chính xác đơn, có thể sử dụng như hai thanh ghi 16 bit.
* Ba tầng pipeline riêng biệt giúp tăng tốc độ xử lí đồng thời.
  + 1. Chuẩn giao tiếp phần mềm vi điều khiển Cortex (Cortex Microntroller Software Interface Standard – CMSIS) [cmsis]

CMSIS là Lớp trừu tượng hóa phần cứng (Hardware Abstraction Layer – HAL) cho các vi điều khiển bất kì thuộc dòng Cortex-M, không phụ thuộc vào nhà sản xuất phần cứng. CMSIS được phát triển bởi ARM nhằm cung cấp cho người phát triển phần mềm vi điều khiển các giao tiếp đến vi xử lí và các ngoại vi một cách đơn giản thông qua các Giao diện lập trình ứng dụng (Application Programming Interface). Qua đó, đơn giản hóa việc tái sử dụng phần mềm, giảm thời gian tìm hiểu và giảm thời gian phát triển phần mềm cho thiết bị.

CMSIS bao gồm các thành phần sau:

* CMSIS-CORE: API cho nhân xử lí Cortex-M và các ngoại vi. Cung cấp giao tiếp được chuẩn hóa cho các nhân xử lí Cortex-M.
* CMSIS-DSP: Thư viện xử lí số tín hiệu (Digital Signal Processing – DSP). Cung cấp các thao tác phục vụ cho xử lí số tín hiệu với tối ưu hóa cho các chỉ lệnh Đơn lệnh – Đa dữ liệu (Single Instruction Multiple Data – SIMD).
* CMSIS-RTOS API: Các API cho hệ điều hành thời gian thực (RTOS).
* CMSIS-SVD: Các đặc tả hệ thống cho ngoại vi. Đặc tả các ngoại vi của thiết bị trong một file XML để có thể sử dụng trong việc hỗ trợ cho bộ gỡ lỗi và các chỉ định ngắt.
* CMSIS-DAP: Debug Access Port. Cung cấp chuẩn hóa cho thiết bị gỡ lỗi.



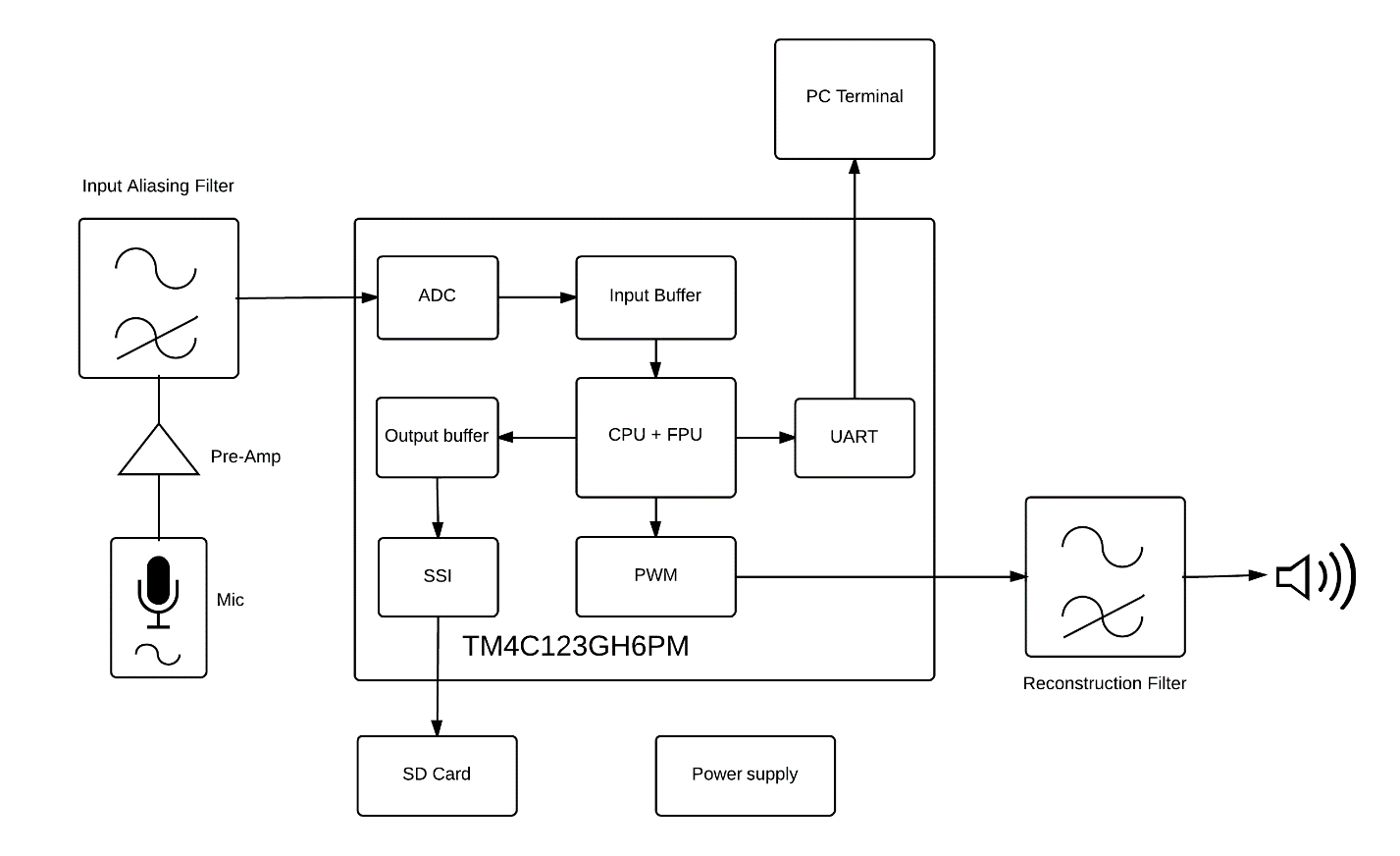
Hình 7 - Cấu trúc của CMSIS [ref – cmsis]

* 1. Kết luận

Chương này giới thiệu khái quát về tín hiệu băng tần tiếng nói. Cách

# THIẾT KẾ PHẦN CỨNG

* 1. Sơ đồ khối thiết kế phần cứng [fix]



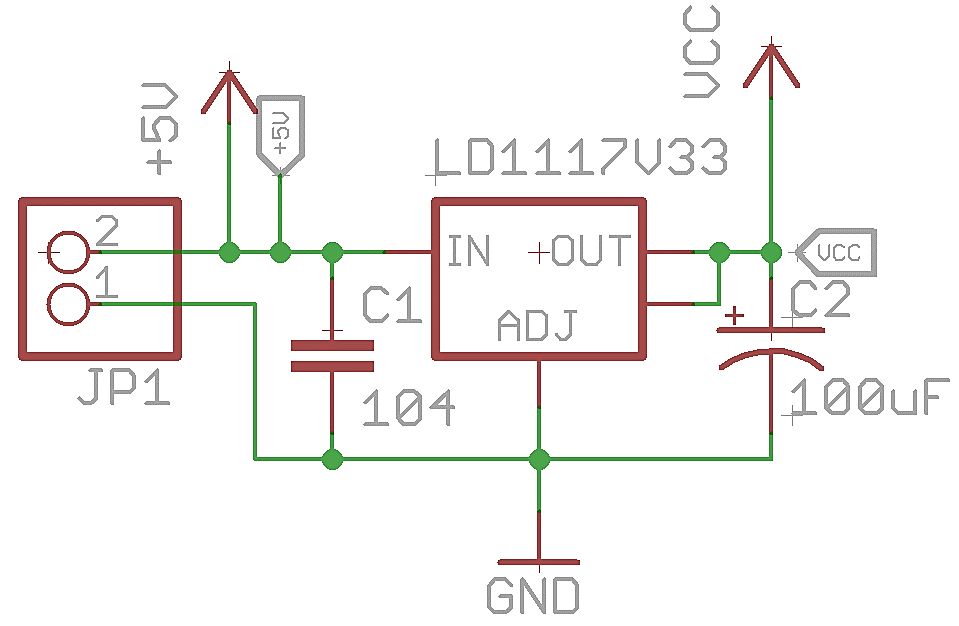
Hình 8 - Sơ đồ khối thiết kế phần cứng

* 1. Mạch nguồn

Mạch nguồn được thiết kế để cung cấp điện áp đơn +3.3V cho hoạt động của hệ thống. Việc sử dụng điện áp nguồn đơn cho phép hệ thống linh động hơn trong việc tận dùng các nguồn điện không cố định như Pin AA hay Pin Li-Ion, Li-Pol.

Một điểm cần lưu ý là thiết kế có sử dụng opamp. Nguồn cấp thông thường cho opamp là nguồn đôi với mức điện áp tham chiếu là GND. Với nguồn đơn, điện áp tham chiếu cho opamp không thể là GND mà phải thay đổi cho phù hợp[ref slyt189.pdf], thông thường chọn là ½ VCC để có độ rộng biên độ dao động tín hiệu ở mức cao nhất.

Sơ đồ mạch nguồn của thiết kể được thể hiện bên dưới:



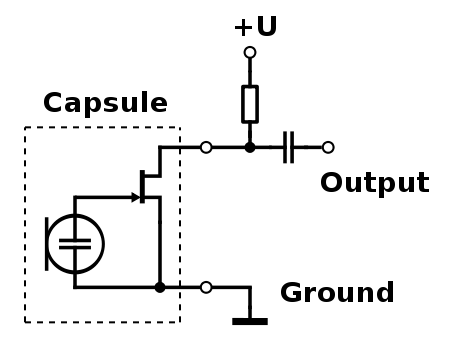
Hình 9 - Sơ đồ nguyên lí mạch nguồn

IC ổn áp được sử dụng là LD1117V33 với các đặc điểm sau [ref LD1117V33.pdf]:

* Điện áp dropout thấp, ~1V
* Dòng ra lớn nhất 800mA
* Tự động giới hạn dòng và giới hạn nhiệt
  1. Mạch tiền khuếch đại và mạch lọc chống chồng phổ IAF
     1. Electret Condenser Microphone (ECM)

ECM được sử dụng trong hầu hết các thiết bị âm thanh thông thường và trong các giải pháp truyền tín hiệu âm thanh [ref - snaa114.pdf].

Dưới đây là một cấu hình cơ bản để ECM hoạt động.



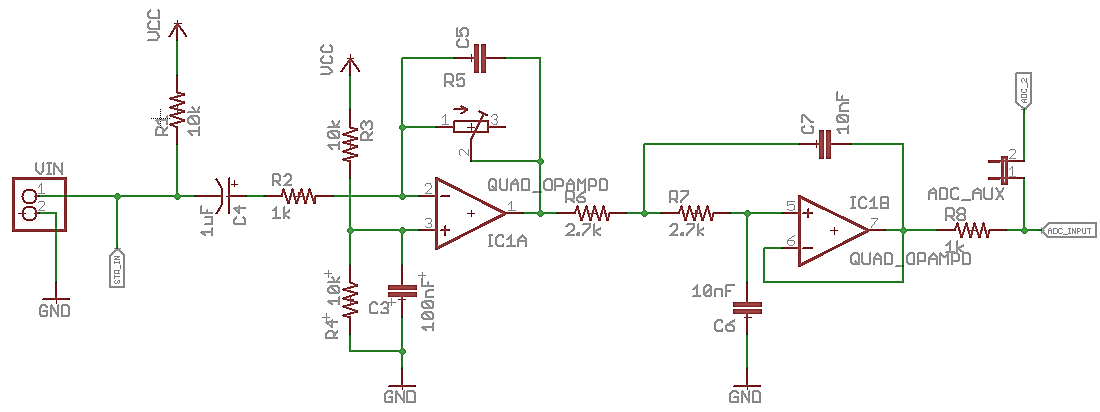
Hình 10 - Cấu hình cơ bản của ECM

Nguyên lí hoạt động của ECM dựa vào sự thay đổi điện dung giữa hai bản cực [ref – engineer garage html]. Bản cực thứ nhất được cố định, bản cực thứ hai có thể rung động được gọi là lớp màng, hai bản cực tạo thành một tụ điện. Tụ điện này đã được nạp sẵn điện [ref electret html]. Khi có âm thanh làm rung động lợp màng, điện dung của tụ điện thay đổi dẫn đến sự thay đổi điện áp giữa hai bản cực theo công thức:

Sự thay đổi này dẫn đến sự thay đổi dòng điện cực máng của JFET làm biến đổi tín hiệu ở đầu ra.

* + 1. Mạch tiền khuếch đại và mạch lọc chống chồng phổ IAF

Dưới đây là sơ đồ nguyên lý mạch tiền khuếch đại và mạch lọc chống chồng phổ IAF:



Hình 11 - Mạch tiền khuếch đại và lọc chống chồng phổ

Mạch tiền khuếch đại có hai nhiệm vụ cơ bản sau:

* Phân cực tín hiệu ECM dao động ở mức ½ VCC.
* Khuếch đại tín hiệu từ ECM.

Bởi vì tín hiệu đầu ra của ECM có biên độ rất nhỏ nên cần phải khuếch đại lên để đạt được biên độ mong muốn trước khi đưa vào ADC. Ngoài ra, từ mục [ref mạch nguồn], thiết kế này sử dụng nguồn đơn do đó tín hiệu từ ECM được phân cực để dao động ở mức ½ VCC. Như vậy biên độ đỉnh-đỉnh của tín hiệu từ ECM đã khuếch đại sẽ là VCC.

Độ lợi của bộ khuếch đại được tính theo công thức sau:

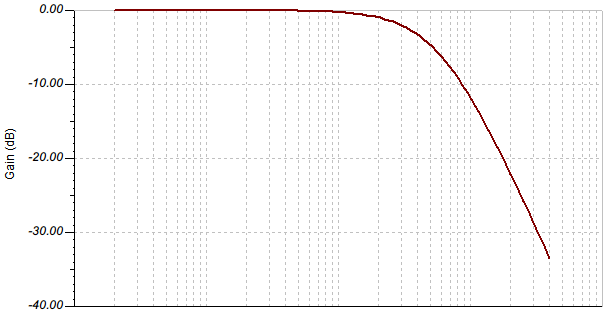
Độ lợi có thể được thanh đổi theo biến trở R5.

Opamp được sử dụng ở đây là OPA4340 của Texas Instruments. OPA4340 có các đặc điểm sau [ref – opa4340.pdf]:

* Phù hợp với nguồn đơn.
* Điện áp vào/ra tiệm cận điện áp nguồn.
* Nhiều thấp, phù hợp với các ứng dụng âm thanh và chuyển đổi A/D.

Mạch lọc chống chồng phổ là mạch lọc thông thấp theo cấu hình Sallen-Key với đáp ứng tần số Butterworth [ref opamp for everyone].

Dưới đây là đáp ứng tần số của mạch trên, được mô phỏng sử dụng phần mềm Tina-TI[ref Tina-TI]:



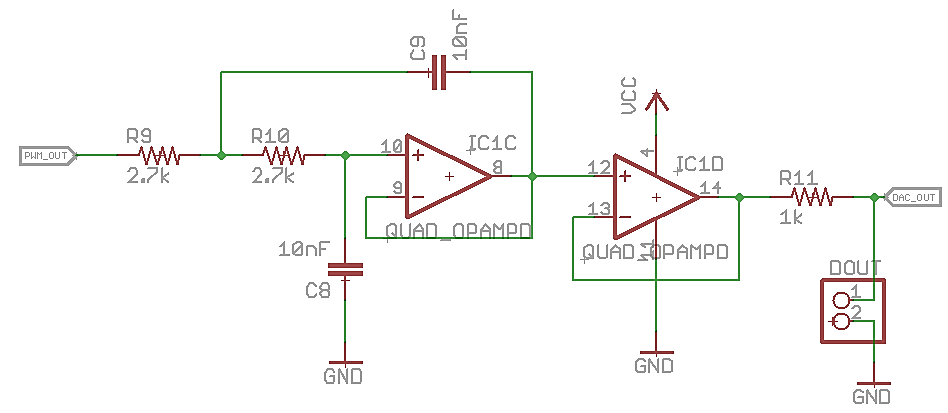
Hình 12 - Đáp ứng tần số mạch lọc chống chồng phổ IAF

Mạch có tần số cắt là 3.85 KHz và dải chuyển tiếp từ 3.85 KHz đến 16 KHz với suy hao tại 16 KHz là -20dB.

* + 1. Mạch tái tạo tín hiệu đầu ra:

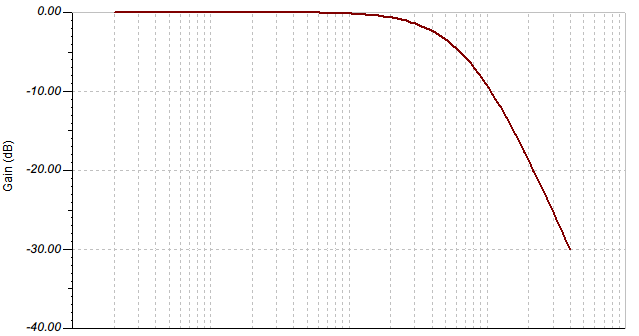
Mạch tái tạo tín hiệu đầu ra bao gồm một mạch lọc thông thấp và một bộ đệm tín hiệu với trở kháng ra đủ để cấp công suất cho một tai nghe.

Hình dưới là sơ đồ nguyên lí mạch tái tạo tín hiệu đầu ra:



Hình 13 Mạch tái tạo tín hiệu đầu ra

Tín hiệu PWM từ mạch điều khiển được đưa qua bộ lọc thông thấp với đáp ứng như sau:



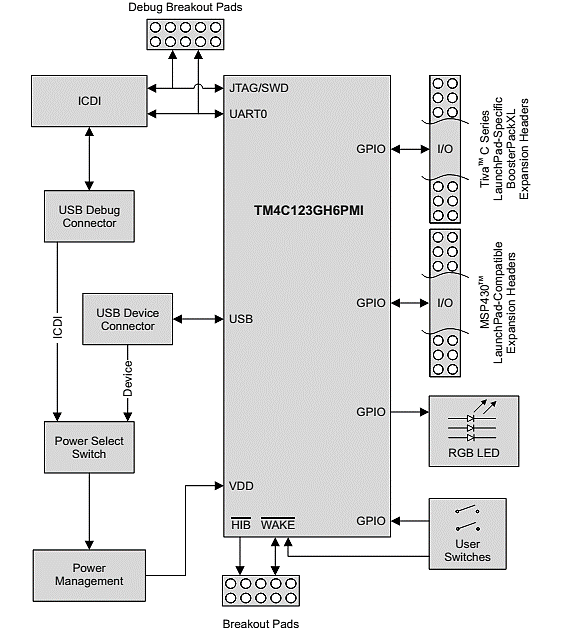
Hình 14 – Đáp ứng bộ lọc thông thấp mạch tái tạo tín hiệu đầu ra

Tần số cắt của mạch là 4.67 KHz.

Bộ đệm tín hiệu có chức năng cách li phần tín hiệu đầu ra từ mạch lọc với tải bên ngoại, trở kháng đầu ra của bộ đệm tín hiệu được xác định bởi R1 là 1000 Ω. Với tai nghe thông thường có trở kháng đặc trưng là 32 Ω. Công suất đầu ra đến tai nghe là:

* 1. Mạch điều khiển chính - Kit phát triển Tiva C Launchpad nền tảng ARM Cortex-M4F

Mạch điều khiển chính sử dụng kit phát triển Tiva C Launchpad của Texas Instrument [ti.com/launchpad].



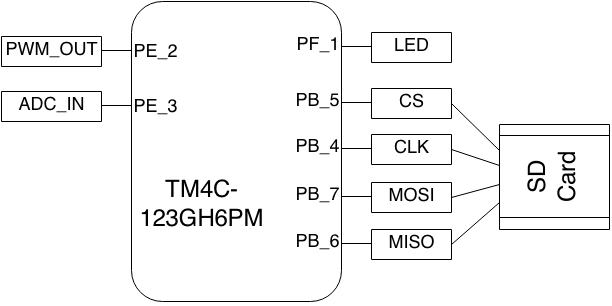
Hình 15 – Sơ đồ khối Tiva C Launchpad

Tiva C Launchpad sử dụng vi điều khiển TM4C123GH6PM với nhân ARM Cortex-M4F [ref to tiva c portal]. Kit cũng tích hợp phần debugger để hỗ trợ gỡ lỗi trong quá trình phát triển phần mềm.

Đặc điểm của TM4C123GH6PM [ref tm4c123gh6pm.pdf]:

* ARM Cortex-M4 CPU và FPU, tốc độ tối đa 80 MHz
* 256 KB Flash, 32 KB RAM và 2 KB EEPROM
* 2x 12 kênh ADC, tốc độ lấy mẫu tối đa 1 MSPS
* 16 kênh PWM
* 24 bộ hỗn hợp Timer/Capture/PWM
* 4 bộ truyền thông đồng bộ SSI, 8 bộ truyền thông không đồng bộ UART
* 43 I/O đa mục đích
* ROM trên chip được cài sẵn driver cho các ngoại vi

Sơ đồ kết nối các chân tín hiệu của Tiva C Launchpad trên mạch như sau:



Hình 16 – Sơ đồ kết nối các chân tín hiệu của Tiva C Launchpad trên mạch



# THIẾT KẾ PHẦN MỀM

* 1. Tổng quan:

Phần mềm điều khiển cần thực hiện các nhiệm vụ sau:

* Giao tiếp với người sử dụng thông qua phương thức truyền thông nối tiếp không đồng bộ UART.
* Cung cấp giao diện quản lí dữ liệu trên thẻ nhớ SD thông qua chuẩn FAT.
* Lấy mẫu tín hiệu và xử lí tín hiệu trong thời gian thực.
* Phát tín hiệu từ dữ liệu lưu trữ trên thẻ nhớ sử dụng PWM.
  1. Kỹ thuật xử lí tín hiệu đa tốc độ lấy mẫu (Multirate Signal Processing)
     1. Hạn chế trong việc thiết kế bộ lọc tích cực cho IAF

Với tốc độ lấy mẫu tín hiệu 8000 Hz, mạch IAF phải đảm bảo loại bỏ tất cả các thành phần tín hiệu có tần số trên 4000 Hz. Như đã trình bày ở [ref tới phần 600 Hz], dải chuyển tiếp của mạch IAF là 600 Hz. Thực hiện thiết kế mạch lọc tích cực cực sử dụng phần mềm FilterPro Desktop [ref link tới filterpro][ref tới file filter pro] với các thông số:

* Kiểu mạch lọc: Mạch lọc thông thấp
* Tần số cắt: 3400 Hz
* Tần số vùng suy hao: 4000 Hz
* Độ suy hao vùng chuyển tiếp: - 40 dB

Các tính toán trên phần mềm cho thấy bậc của mạch lọc tối thiểu là 10 với kiểu đáp ứng xung Chebyshev và cấu hình Sallen-Key. Với các kiểu đáp ứng khác, số bậc là lớn hơn 10. Cần phải ghép 5 tầng opamp với các linh kiện thụ động khác để tạo nên mạch trên. Điều này gây ra những khó khăn như sau:

* Phức tạp khi chọn lựa các linh kiện phù hợp với mạch lọc.
* Phức tạp khi thiết kế mạch in.
* Sai số linh kiện và sai số do nhiệt cùng với nhiễu tích lũy qua nhiều tầng gây ra ảnh hưởng đến đáp ứng và gây méo tín hiệu.
  + 1. Cơ sở kĩ thuật xử lí đa tốc độ lấy mẫu:

Để giải quyết vấn đề trên, kĩ thuật xử lí đa tốc độ lấy mẫu có thể được áp dụng. Mô hình cho hệ thống xử lí đa tốc độ lấy mẫu như sau:

Tín hiệu

IAF

ADC

Bộ lọc số FIR

Decimator

Tín hiệu

Hình 17 – Kĩ thuật xử lí đa tốc độ lấy mẫu

Tín hiệu đầu vào được lấy mẫu với tốc độ lấy mẫu là 32 KHz. Như vậy, bộ lọc chống chồng phổ có thể được thiết kế với tần số cắt 3.4KHz và dải tần số chuyển tiếp từ 3.4 KHz – 16 KHz. Điều này làm đơn giản hóa rất nhiều cho công việc thiết kế IAF. Tiếp theo, một bộ lọc số FIR có tần số cắt 3.4 KHz với dải chuyển tiếp 600 Hz sẽ thực hiện loại bỏ các thành phần tần số từ 4 KHz đến 16 KHz. Bộ Decimator về bản chất là bộ hạ tốc độ lấy mẫu. Decimator thực hiện hạ tốc độ lấy mẫu bằng cách trích 1 mẫu trong một nhóm 4 mẫu liên tục để tạo thành tín hiệu cuối cùng với tần số lấy mẫu 8 KHz và tần số cắt là 3.4 KHz.

* + 1. Thuật toán cho bộ Decimator:

Bộ Decimator với M = 4 có thể được diễn giải bằng thuật toán dưới:

*Input:* Mảng in[2048] chứa các mẫu liên tiếp của tín hiệu đầu với với tần số lấy mẫu 32 KHz.

*Output:* Mảng out[512], chứa các mẫu đầu ra với tần số lấy mẫu 8 KHz.

*Bước 1:* i ← 0; t ← 0.

*Bước 2:* out[t] = in[i].

*Bước 3:* t ← t + 1; i ← t \* 4

*Bước 4:* Nếu t = 512. Kết thúc.

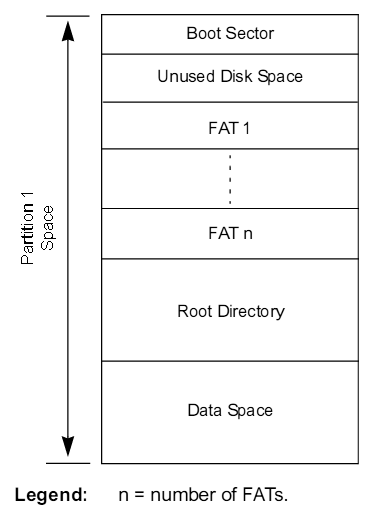
*Bước 5:* Quay về *Bước 2*.

* 1. Hệ thống quản lí file theo chuẩn FAT
     1. Đặc điểm của FAT

Dữ liệu ghi vào thiết bị lưu trữ cần được quản lí tốt để dễ dàng truy cập và sử dụng trong tương lai. FAT là một giải pháp tốt cho việc quản lí dữ liệu trên thiết bị lưu trữ như thẻ nhớ SD.

Bảng cấp phát tập tin (File Allocation Table – FAT) được Microsoft phát triển từ những năm 1970[ref fatgen103.doc], với phiên bản mới nhất là FAT32 được ra mắt năm 1996.

FAT cung cấp giải pháp quản lí tiện lợi, đơn giản và có thể tích hợp dễ dàng trong các phần mềm hệ thống nhúng [ref FAT File System (Windows Embedded CE 6.0)]. Cấu trúc của FAT trên một phân vùng như sau[ref 01045b.pdf microchip]:



Hình 18- Cấu trúc của một phân vùng sử dụng FAT

Thẻ nhớ SD thông thường chỉ có một phân vùng, bao gồm các thành phần sau:

* Boot Sector: Sector đầu tiên của phân vùng chứa các thông tin cơ bản.
* Các vùng FAT: Chứa bản ghi thể hiện các sắp xếp dữ liệu trên vùng dữ liệu. Thông thường có 2 bản FAT giống nhau trong vùng này để đề phòng trường hợp xảy ra mất mát dữ liệu.
* Root Directory: Chứa bảng tập hợp các thư mục con và các file ở thư mục cấp cao nhất (root). Thông tin về các thư mục con và file cấp thấp hơn được chứa tại thư mục cấp trên nó.

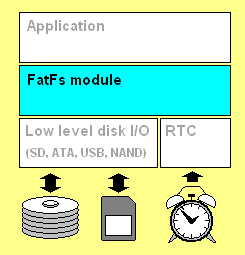
Vùng dữ liệu (Data Region): Chứa dữ liệu tệp hoặc thông tin các thư mục con. Dữ liệu của vùng này vẫn được giữ nguyên trong mọi trường hợp trừ trường hợp nó bị ghi đè lên.

* + 1. Thư viện FatFs[ref link fatfs]

FatFS là một module cung cấp giải pháp quản lí theo chuẩn FAT cho các ứng dụng nhúng. FatFs được viết theo quy chuẩn ANSI C và hoàn toàn tách biệt với lớp giao diện vào ra với thiết bị lưu trữ dữ liệu. Do đó FatFs không phụ thuộc vào kiến trúc phần cứng, nó có thể được tích hợp vào hầu hết các kiến trúc vi điều khiển mà không cần phải thực hiện bất kì thay đổi gì ở mã nguồn.

Các đặc trưng của FatFs [ref elm chan]:

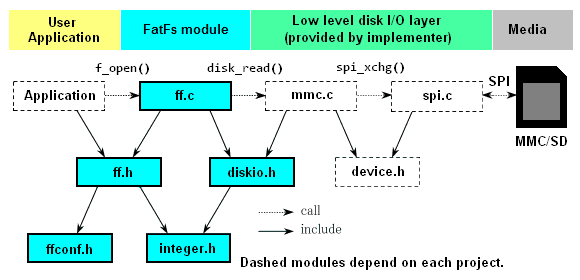
* Tương thích với chuẩn FAT của Windows
* Không phụ thuộc vào nền tảng phần cứng. Dễ dàng khi chuyển đổi nền tảng.
* Hỗ trợ kiểu tên nhiều kí tự theo chuẩn ANSI/OEM hay Unicode.
* Hỗ trợ hệ điều hành thời gian thực RTOS
* Hỗ trợ nhiều loại kích thước sector
* API được tối thiểu hóa, chế độ chỉ đọc, đệm vào/ra…



Hình 19 – Ví trí lớp thư viện FatFs trong các lớp của phần mềm

* + 1. Ứng dụng thư viện FatFS để triển khai FAT32 trên thẻ nhớ SD

Hình dưới đây mô tả cấu hình cơ bản của một hệ thống nhúng với FatFs [ref elmchan]:



Hình 20 – Cấu hình cơ bản của hệ thống nhúng với FatFs

Phần mềm hệ thống phải cung cấp trình điều khiển thiết bị lưu trữ dữ liệu ở lớp I/O cho module FatFs. Cụ thể là các hàm dưới đây:

* static void **power\_on**(): Khởi tạo các I/O và module SSI cần thiết cho việc thực hiện truyền dữ liệu theo phương thức SPI.
* static BYTE **send\_cmd**(): Gởi các lệnh điều khiển đến thẻ nhớ SD.
* static BOOL **rcvr\_block**(): Nhận một gói dữ liệu từ thẻ nhớ SD.
* static BOOL **xmit\_datablock**(): Gởi một gói dữ liệu đến thẻ nhớ SD.
* DSTATUS **disk\_initialize**(): Khởi tạo phân vùng sử dụng FatFs.
* DSTATUS **disk\_status**(): Kiểm tra trạng thái phân vùng.
* DRESULT **disk\_read**(): Đọc một hoặc nhiều sector.
* DRESULT **disk\_write**(): Ghi một hoặc nhiều sector.
* DRESULT **disk\_ioctl**(): Cung cấp các chức năng phụ

Ở lớp ứng dụng, FatFs cung cấp các thao tác trên file và thư mục thông qua các API dưới đây[elm-chan]:

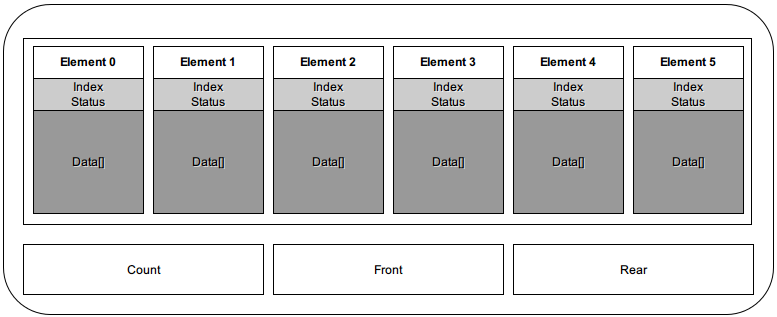
* **f\_mount** – Khởi tạo hay hủy bỏ phân vùng làm việc
* **f\_open** – Mở hay tạo mới một file
* **f\_close** – Đóng một file
* **f\_read** – Đọc một file
* **f\_write** – Ghi dữ liệu vào một file
* **f\_lseek** – Di chuyển con trong bên trong file để đọc ghi, mở rộng kích thước file.
* **f\_truncate** – Cắt giảm kích thước file
* **f\_stat** - Kiểm tra trạng thái của file hay thư mục
* **f\_opendir** – Mở một thư mục
* **f\_closedir** – Đóng một thư mục
* **f\_readdir** – Lấy thông tin về các file và thư mục con.
* **f\_mkdir** – Tạo thư mục con
* **f\_unlink** – Xóa một file hay một thư mục
  1. Bộ đệm dữ liệu kiểu Vào trước ra trước (FIFO)
     1. Sự cần thiết của bộ đệm dữ liệu kiểu Vào trước ra trước (FIFO)

Bộ đệm dữ liệu là một vùng dữ liệu được chia sẻ bởi các phần cứng hay các tiến trình khác nhau trong một phần mềm. Những thành phần này hoạt ở tốc độ khác nhau hay ở cấp độ ưu tiên khác nhau, bộ đệm được sử dụng để đảm bảo rằng hoạt động của tất cả các thành phần trong hệ thống không bị gián đoạn vì sự những khác biệt trên. Có hai kiểu đệm dữ liệu thông dụng đó là Vào sau ra trước (Last In Firt Out – LIFO) và Vào trước ra trước (First In First Out – FIFO), ứng với hai kiểu cấu trúc là ngăn xếp (stack) và hàng đợi (queue). Tính chất tự nhiên của hàng đợi cho thấy đây là kiểu đệm dữ liệu thích hợp cho một hệ thống thu phát tín hiệu thời gian thực.

* + 1. Thiết kế bộ đệm FIFO [cần sửa lại title]

Bộ đệm FIFO được thiết kế để đảm bảo sự lưu thông xuyên suốt của dữ liệu trong hai quá trình (1) Đệm dữ liệu cho bộ chuyển đổi A/D, (2) Đệm dữ liệu cho chức năng phát tín hiệu.

Bộ đệm FIFO trong thiết kế này cố định về dung lượng, dung lượng của bộ đệm được xác định tại thời điểm dịch chương trình. Kết cấu của bộ đệm về cơ bản bao gồm hai cấu trúc (C Structure) lồng nhau, với một cấu trúc chứa thông tin quản lí bộ đệm và cấu trúc còn lại chứa dữ liệu và thông tin của mỗi phần tử trong bộ đệm. Hình dưới mô tả cấu trúc của bộ đệm FIFO với 6 phần tử:



Hình 21 – Cấu trúc bộ đệm FIFO với 6 phần tử

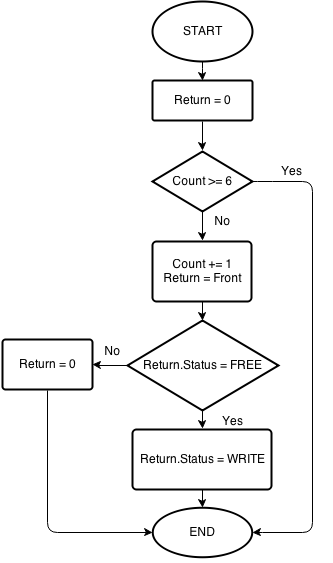
Ở cấu trúc quản lí:

* Count: đếm số lượng phần tử đang chiếm dụng. Giá trị của Count không bé hơn 0 và không lớn hơn tổng số phần tử được xác định của bộ đệm.
* Front: lưu giá trị Index của phần tử nằm ở đầu.
* Rear: lưu giá trị Index của phần tử nằm ở đuôi.

Ở cấu trúc phần tử:

* Index: chỉ thị vị trí của phần tử trong chuỗi các phần tử của bộ đệm.
* Status: chỉ thị trạng thái của phần tử. Có ba trạng thái:
  + FREE: phần tử được đang chờ được truy xuất.
  + READ: phần tử đang được đọc, không cho phép truy xuất tiếp theo.
  + WRITE: phần tử đang được ghi, không cho phép truy xuất tiếp theo.

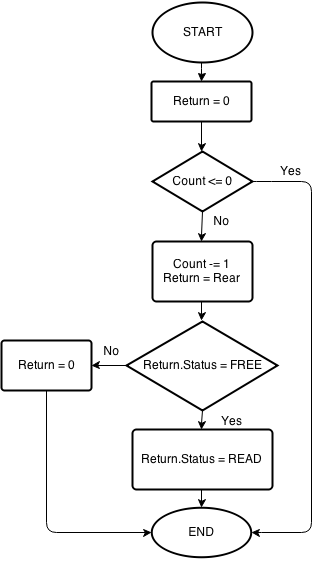
Quá trình yêu cầu một phần tử trống ở bộ đệm FIFO có 6 phần tử để thực hiện việc ghi dữ liệu lên được mô tả thông qua sơ đồ sau:



Hình 22 – Sơ đồ thuật toán yêu cầu phần tử trống từ bộ đệm FIFO

Kết quả trả về Return là một biến dạng con trỏ ban đầu được gán giá trị rỗng. Nếu tất cả các phần tử của bộ đệm đều đã chứa dữ liệu chưa sử dụng (Count >= 6), thì giá trị trả về sẽ là rỗng. Nếu không, giá trị của Front sẽ được gán cho Return. Công việc tiếp theo là kiểm tra phần này đang bị chiếm dụng hay không. Nếu trạng thái của nó là FREE thì phần tử sẽ được trả về, hoặc giá trị là rỗng nếu ngược lại.

Việc yêu cầu một phần tử đã chứa dữ liệu ở bộ đệm để cho tao tác đọc dữ liệu được thực hiện tương tự:

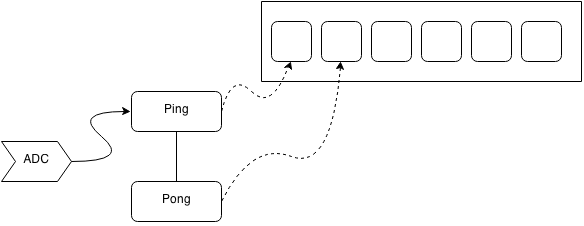


Hình 23 – Sơ đồ thuật toán yêu cầu phần tử chứa dữ liệu từ bộ đệm FIFO

* + 1. Bộ đệm FIFO với μDMA cơ chế Ping-Pong

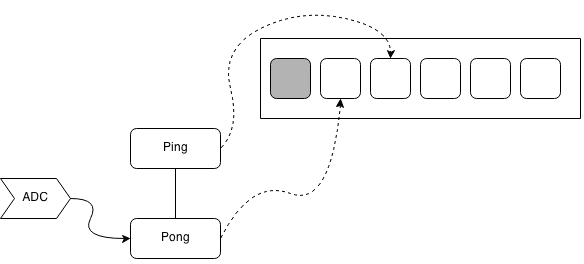
TM4C123GH6PM cung bộ điều khiển truy xuất bộ nhớ trực tiếp (DMA) với tên gọi μDMA[tm4c.pdf]. Chế độ hoạt động kiểu Ping-Pong của μDMA cho phép tự động chuyển đổi vùng đệm dữ liệu một cách tự động mà không cần sự can thiệp của phần mềm. Kết hợp giữa μDMA hoạt động chế độ Ping-Pong và bộ đệm FIFO giúp tối đa khả năng hoạt động thời gian thực của hệ thống.

Quá trình lấy mẫu tín hiệu với μDMA hoạt động chế độ Ping-Pong và bộ đệm FIFO được mô tả dưới đây:



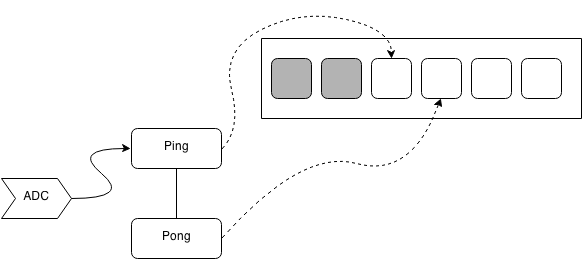
Hình 24 – 𝜇DMA trạng thái sử dụng bộ đệm Ping

Ban đầu, Ping và Pong lần lượt được trỏ đến hai phần tử đầu tiên của bộ đệm FIFO. ADC thực hiện lẫy mẫu và μDMA sẽ di chuyển dữ liệu vào Ping.



Hình 25 – 𝜇DMA trạng thái sử dụng bộ đệm Pong

Khi Ping đầy, μDMA tự độ chuyển dữ liệu sang Pong, sau đó sẽ phát cờ ngắt để chương trình thiết lập trỏ Ping đến phần tử tiếp theo của bộ đệm.



Hình 26 – 𝜇DMA trở về trạng thái sử dụng bộ đệm Ping

Dữ liệu đã ghi ở bộ đệm FIFO sẽ được các tiến trình phần mềm khác xử lí. Các phần tử của bộ đệm FIFO đã được xử lí sau đó sẽ được trả về trạng thái FREE để có thể tiếp tục nhận dữ liệu mới. Quá trình này lặp lại tạo ra sự liên tục trong việc lấy mẫu tín hiệu.

* 1. Bộ lọc thông thấp Đáp ứng xung chiều dài hữu hạn (FIR)
     1. Thiết kế nhân bộ lọc h[i] với Scipy:

Scipy là một hệ sinh thái dựa trên ngôn ngữ Python tập hợp các công cụ phần mềm hỗ trợ cho toán học, khoa học và kỹ thuật[scipy.org]. Về cơ bản Scipy bao các công cụ sau:

* NumPy: gói hỗ trợ xử lí mảng N-chiều.
* Scipy library: thư viện nền cho các chức năng tính toán trong khoa học.
* Matplotlib: thư viện đồ họa 2 chiều
* IPython: giao diện tương tác dòng lệnh
* Sympy: thư viện kí tự trong toán học
* pandas: thư viện hỗ trợ về cấu trúc và phân tích dữ liệu

Scipy cung cấp thư viện *signal* để hỗ trợ việc thiết kế bộ lọc số. Một bộ lọc FIR có thê được thiết kế thông qua hàm *firwin* với giá trị trả về là mảng chứa các hệ số của nhân bộ lọc FIR:

**scipy.signal.firwin**(*numtaps, cutoff, width=None, window='hamming', pass\_zero=True, scale=True, nyq=1.0*)

Trong đó:

* *numtaps*: số phần tử của mảng, M + 1
* *cutoff*: tần số cắt, tính theo tỉ lệ chuẩn hóa với tần số lấy mẫu, *cutoff* nằm trong khoảng từ 0 đến ½ tần số lấy mẫu.
* *width*: tham số này không sử dụng nếu dùng cửa sổ
* *window*: loại cửa sổ sử dụng (Blackman, Hamming, …)
* *scale*: mặc định là True để đáp ứng có độ lợi là 1 ở dải thông của bộ lọc
* *nyq*: mặc định là 1.0

Bộ lọc cần thiết kế có tần số cắt là 3400 Hz với dải chuyển tiếp có độ rộng 600 Hz. Với tần số lấy mẫu là 32000 Hz, từ 1.2.2 tính được chiều dài của nhân bộ lọc M là 214. Như vậy h[i] sẽ có tổng cộng 215 phần tử.

Sử dụng cửa sổ Blackman, việc tính toán h[i] được thực hiện thông qua mã nguồn sau:

sample\_rate = 32000.0

nyq\_rate = sample\_rate / 2.0

cutoff\_hz = 3400.0

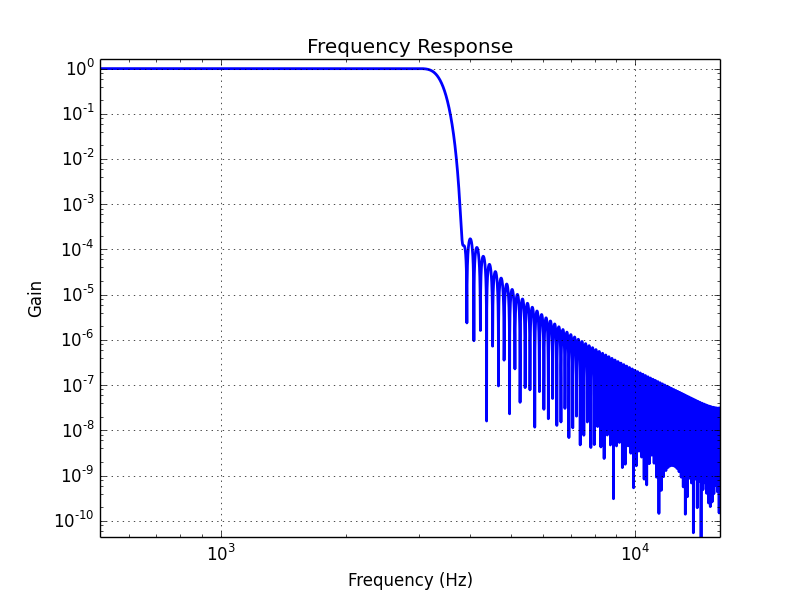
N = 215

taps = firwin(N, cutoff\_hz/nyq\_rate, window='blackman')

Kết quả được trình bày ở **Phụ lục 1**.

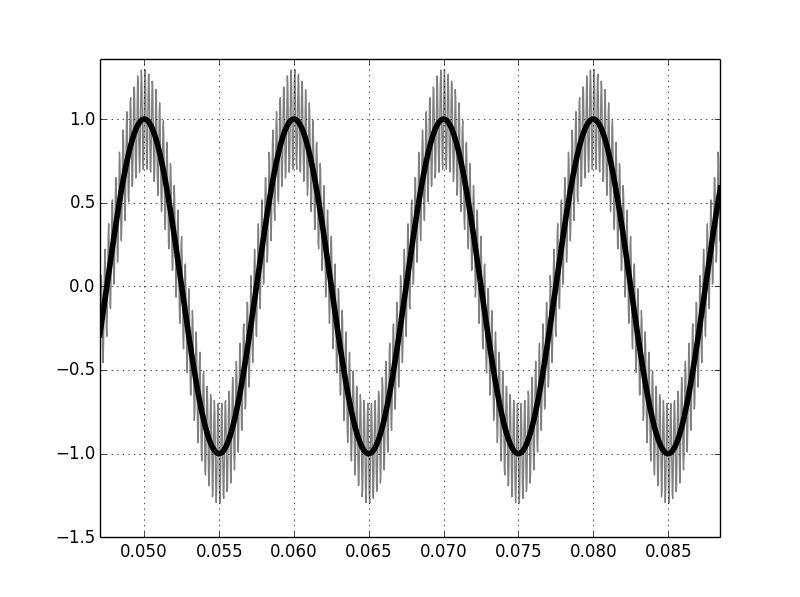
* + 1. Mô phỏng kết quả bộ lọc trên Scipy

Đáp ứng tần số của bộ lọc:



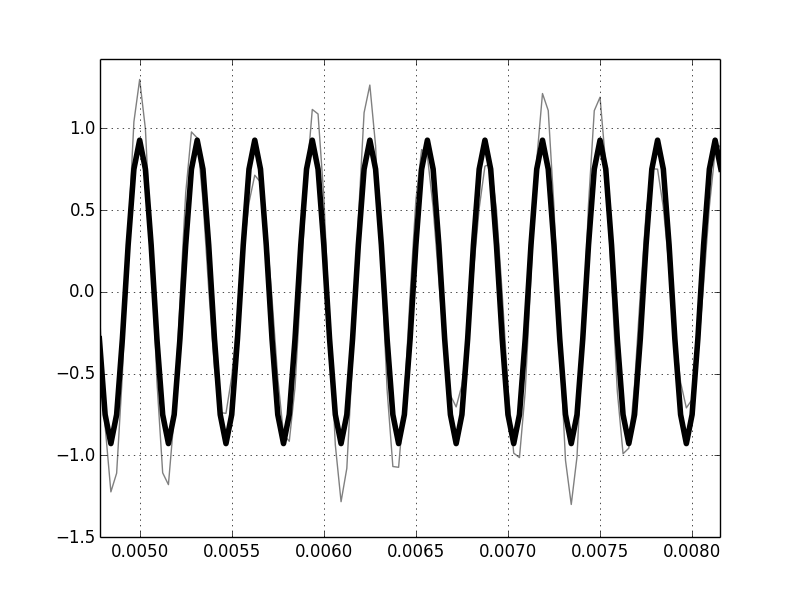
Hình 27 – Đáp ứng tần số của bộ lọc FIR 215 điểm

Kết quả mô phỏng bộ lọc với tín hiệu tổng hợp 100 Hz và 4100 Hz. Đường màu xám là tín hiệu ban đầu, đường màu đen là tín hiệu đã được lọc thành phần 4100 Hz.



Hình 28 – Bộ lọc với tín hiệu tổng hợp 100 Hz và 4100 Hz

Kết quả mô phỏng bộ lọc với tín hiệu tổng hợp 3200 Hz và 4050 Hz cũng cho thấy thành phần 4050 Hz đã được loại bỏ hoàn toàn.



Hình 29 - Bộ lọc với tín hiệu tổng hợp 3200 Hz và 4050 Hz

* + 1. Nhúng bộ lọc vào phần mềm điều khiển

CMSIS hỗ trợ các thao tác xử lí tín hiệu số thông qua thư viện DSP Lib. Bộ lọc FIR có thể được thực hiện thông qua hàm **arm\_fir\_f32**.

void **arm\_fir\_f32** ( *const arm\_fir\_instance\_f32 \* S,*

*float32\_t \* pSrc,*

*float32\_t \* pDst,*

*uint32\_t blockSize*

)

Trong đó:

* arm\_fir\_instance\_f32: trỏ tới mảng S chứa nhân của bộ lọc.
* pSrc: trỏ tới mảng chứa dữ liệu đầu vào
* pDst: trỏ tới mảng chứa dữ liệu đầu ra
* blockSize: số mẫu được xử lí trong một lần thực hiện hàm.

Các thao tác của **arm\_fir\_f32** là các thao tác trên dấu phẩy động kiểu độ chính xác đơn (single-precision floating point), trong khi đó giá trị lấy mẫu từ ADC của TM4C123GH6PM là kiểu nguyên. Do đó, để sử dụng hàm trên, việc cần thiết là phải chuyển đổi các giá trị dữ liệu tín hiệu vào từ số nguyên thành dấu phẩy động. Mặc dù DSP Lib cũng cung cấp các hàm xử lí tín hiệu số kiểu số nguyên, tuy nhiên sử dụng kiểu dấu phẩy động có các lợi thế cơ bản sau:

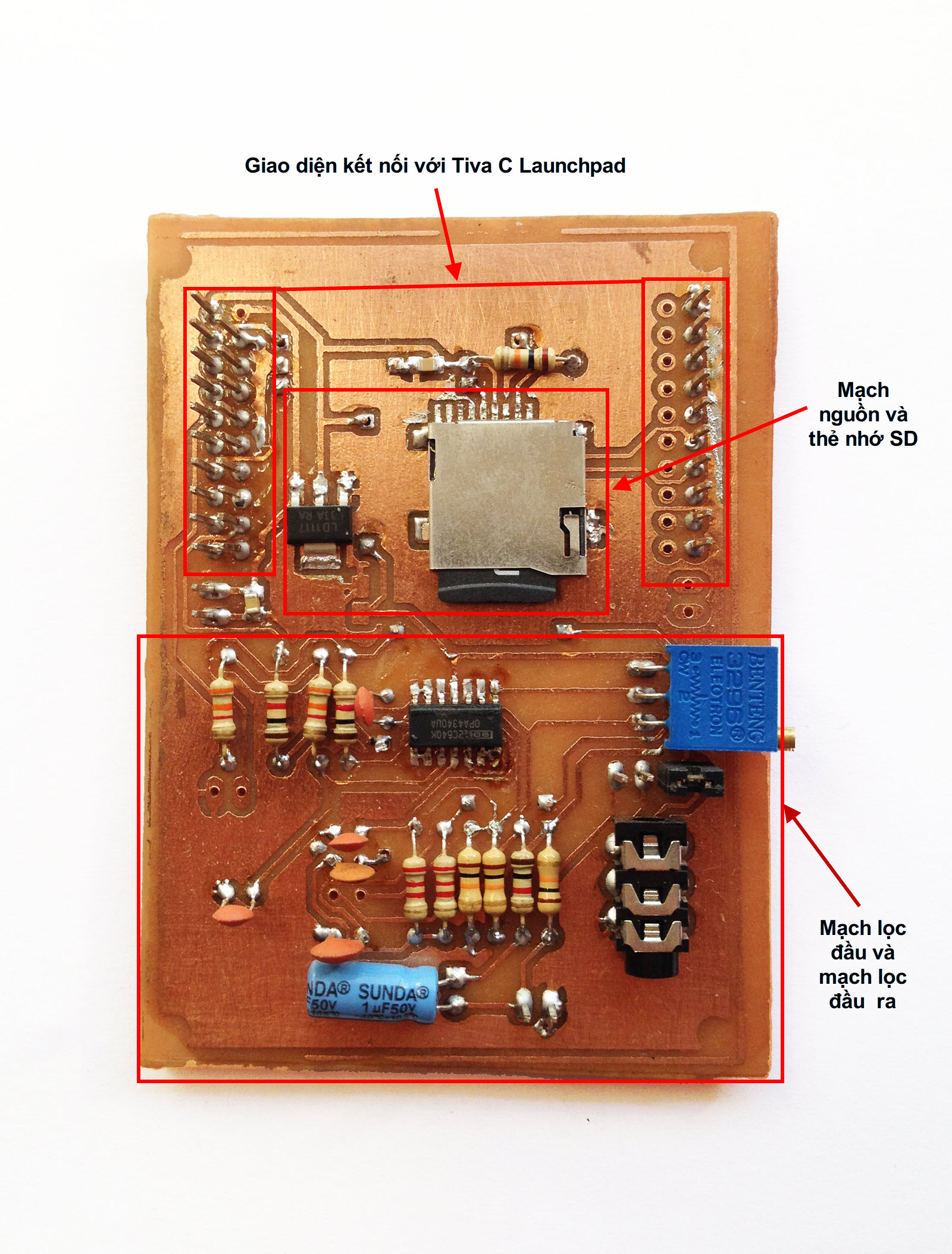
* Tỉ lệ Tín hiệu trên nhiễu (SNR) cao, nhiễu do lượng tử hóa thấp hơn 30000 lần so với việc dùng kiểu số nguyên. [ref dspguide]
* Tận dụng được khả năng hỗ trợ phần cứng của Đơn vị dấu phẩy động FPU để tăng tốc xử lí.



# KẾT QUẢ THỰC NGHIỆM

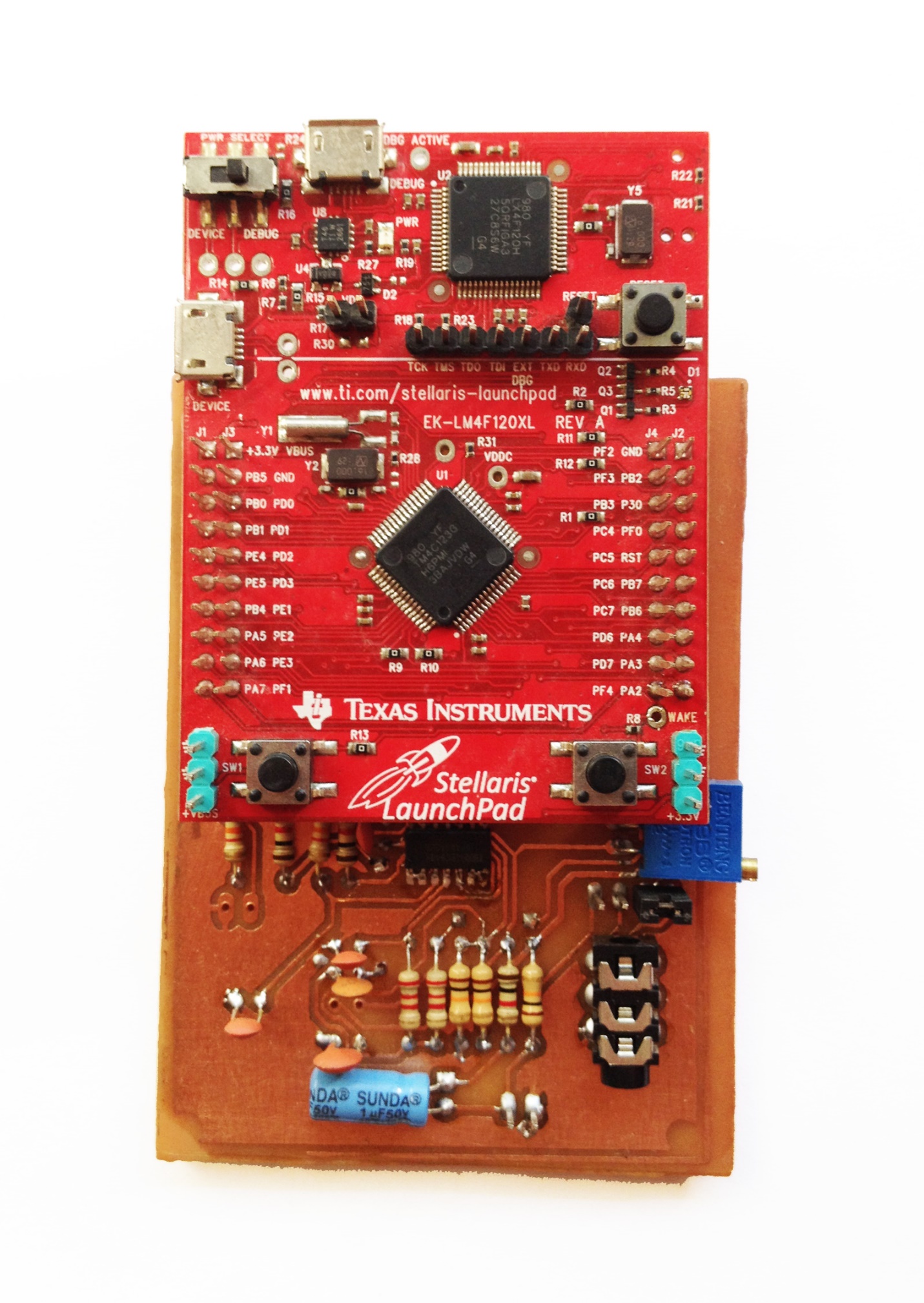
* 1. Phần cứng và giao diện phần mềm:

Phần mạch kết nối với Kit Tiva C Launchpad:



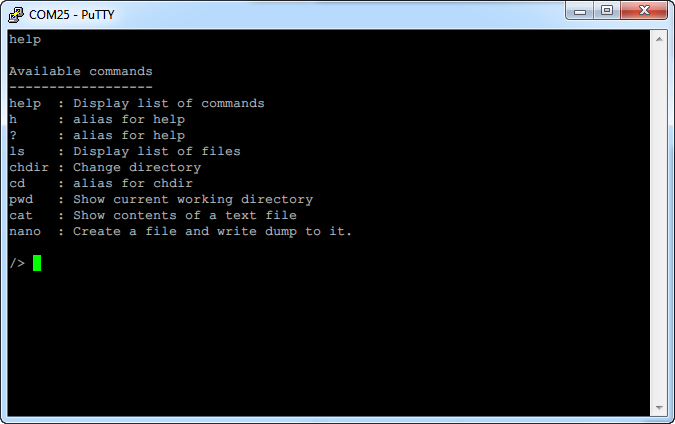
Hình 30 – Mạch kết nối với Kit Tiva C Launchpad

Kết nối hoàn chỉnh với Kit Tiva C Launchpad:



Hình 31 – Kết nối mạch với Kit Tiva C Launchpad

Giao diện điều khiển dòng lệnh thông qua truyền thông không đồng độ UART:



Hình 32 – Giao diện dòng lệnh qua UART

* 1. Các kết quả

# PHỤ LỤC 1 – NHÂN BỘ LỌC THÔNG THẤP FIR TRONG THIẾT KẾ

-0.0000000000f, +0.0000002323f, +0.0000007832f, +0.0000006623f,

-0.0000013344f, -0.0000052010f, -0.0000088389f, -0.0000086861f,

-0.0000019135f, +0.0000109305f, +0.0000246171f, +0.0000306960f,

+0.0000217068f, -0.0000036617f, -0.0000374144f, -0.0000636747f,

-0.0000652397f, -0.0000329458f, +0.0000267322f, +0.0000906192f,

+0.0001266479f, +0.0001085528f, +0.0000312376f, -0.0000812237f,

-0.0001809153f, -0.0002150066f, -0.0001514953f, -0.0000000000f,

+0.0001840694f, +0.0003175232f, +0.0003249826f, +0.0001776711f,

-0.0000833346f, -0.0003538823f, -0.0005057655f, -0.0004446306f,

-0.0001617323f, +0.0002468301f, +0.0006083376f, +0.0007434267f,

+0.0005508558f, +0.0000685616f, -0.0005221625f, -0.0009603081f,

-0.0010166269f, -0.0006069718f, +0.0001462305f, +0.0009412564f,

+0.0014130646f, +0.0012957560f, +0.0005612871f, -0.0005343108f,

-0.0015316759f, -0.0019550690f, -0.0015320035f, -0.0003469578f,

+0.0011513379f, +0.0023113232f, +0.0025549538f, +0.0016547575f,

-0.0001171513f, -0.0020530995f, -0.0032832550f, -0.0031571081f,

-0.0015695511f, +0.0009256536f, +0.0032924294f, +0.0044315507f,

+0.0036777162f, +0.0011550680f, -0.0021882817f, -0.0049191898f,

-0.0057191265f, -0.0039998024f, -0.0002553880f, +0.0040390115f,

+0.0069873984f, +0.0070881679f, +0.0039629010f, -0.0013417470f,

-0.0066640387f, -0.0095781522f, -0.0084634882f, -0.0033352546f,

+0.0039649501f, +0.0103786814f, +0.0128614278f, +0.0097586557f,

+0.0017315711f, -0.0082336439f, -0.0158512626f, -0.0172755251f,

-0.0108842665f, +0.0016640160f, +0.0156537114f, +0.0248954297f,

+0.0241880976f, +0.0117573361f, -0.0093288647f, -0.0314689127f,

-0.0446569404f, -0.0398304505f, -0.0123105452f, +0.0359236943f,

+0.0960510962f, +0.1545391329f, +0.1969944767f, +0.2125004224f,

+0.1969944767f, +0.1545391329f, +0.0960510962f, +0.0359236943f,

-0.0123105452f, -0.0398304505f, -0.0446569404f, -0.0314689127f,

-0.0093288647f, +0.0117573361f, +0.0241880976f, +0.0248954297f,

+0.0156537114f, +0.0016640160f, -0.0108842665f, -0.0172755251f,

-0.0158512626f, -0.0082336439f, +0.0017315711f, +0.0097586557f,

+0.0128614278f, +0.0103786814f, +0.0039649501f, -0.0033352546f,

-0.0084634882f, -0.0095781522f, -0.0066640387f, -0.0013417470f,

+0.0039629010f, +0.0070881679f, +0.0069873984f, +0.0040390115f,

-0.0002553880f, -0.0039998024f, -0.0057191265f, -0.0049191898f,

-0.0021882817f, +0.0011550680f, +0.0036777162f, +0.0044315507f,

+0.0032924294f, +0.0009256536f, -0.0015695511f, -0.0031571081f,

-0.0032832550f, -0.0020530995f, -0.0001171513f, +0.0016547575f,

+0.0025549538f, +0.0023113232f, +0.0011513379f, -0.0003469578f,

-0.0015320035f, -0.0019550690f, -0.0015316759f, -0.0005343108f,

+0.0005612871f, +0.0012957560f, +0.0014130646f, +0.0009412564f,

+0.0001462305f, -0.0006069718f, -0.0010166269f, -0.0009603081f,

-0.0005221625f, +0.0000685616f, +0.0005508558f, +0.0007434267f,

+0.0006083376f, +0.0002468301f, -0.0001617323f, -0.0004446306f,

-0.0005057655f, -0.0003538823f, -0.0000833346f, +0.0001776711f,

+0.0003249826f, +0.0003175232f, +0.0001840694f, -0.0000000000f,

-0.0001514953f, -0.0002150066f, -0.0001809153f, -0.0000812237f,

+0.0000312376f, +0.0001085528f, +0.0001266479f, +0.0000906192f,

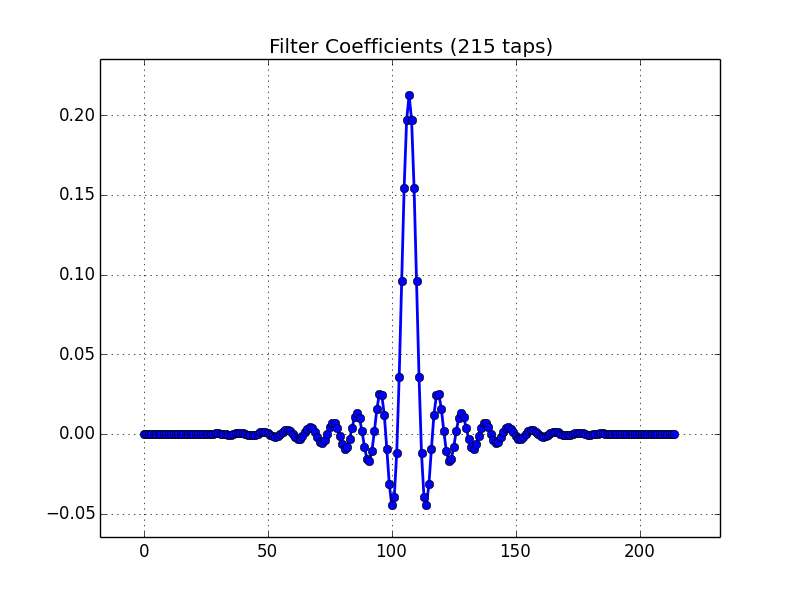
+0.0000267322f, -0.0000329458f, -0.0000652397f, -0.0000636747f,

-0.0000374144f, -0.0000036617f, +0.0000217068f, +0.0000306960f,

+0.0000246171f, +0.0000109305f, -0.0000019135f, -0.0000086861f,

-0.0000088389f, -0.0000052010f, -0.0000013344f, +0.0000006623f,

+0.0000007832f, +0.0000002323f, -0.0000000000f



Hình 33 – Nhân bộ lọc thông thấp FIR trong thiết kế