Group 07

Group 07 Disclosure Document

Caleb Gannon gannonc 400137271

Ben Stephens stephb5 400135219

CLI Commands

Open terminal in src folder.

make

cat test1.raw | **exe CLI** | aplay -c 2 -f S16_LE -r 48000

exe CLI options:

./experiment – mono, mode 0, binary input

./experiment 0 – mono, mode 0, binary input

./experiment 0 -s – stereo, mode 0, binary input

./experiment 0 -s -f – stereo, mode 0, float32 input

./experiment 1 – mono, mode 1, binary input

./experiment 1 -s – stereo, mode 1, binary input

./experiment 1 -s -f – stereo, mode 1, float32 input

Initial Group Comments

Ben and Caleb recognize that there is conflict between the perspective of each section in this group. Our greatest disinclination about the division of efforts stems in our eyes from the fact that John and Sawoud: failed to take a group-oriented programming approach, built methods on the side that were not functional in the larger context of the working model; with disregard to all recent updates.

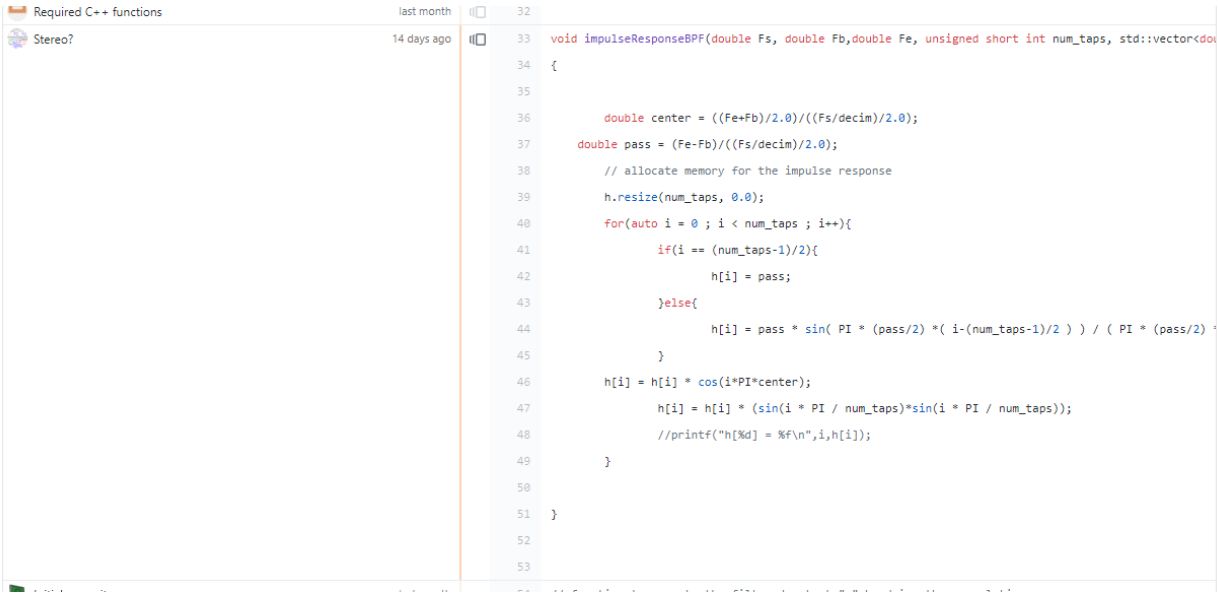
Please Note:

Our group mates are claiming responsibility for implementing the bandpass filter.

BandPass Filter	src/filiter.cpp	970a86af1b5b3833765b87f7daaec4a578b88ad0

Several times throughout the duration of the project, Ben and Caleb's branch would be merged into John and Sawoud's branch.

As you can see in the branch history of filter.cpp between the two branches: Caleb is responsible for uploading the impulseResponseBPF method 2 days before the commit made by John and Sawoud. The code is identical, and it is dishonest for them to claim it as their own:



Required C++ functions — last month
Stereo? — 14 days ago

```
void impulseResponseBPF(double Fs, double Fb,double Fe, unsigned short int num_taps, std::vector<dou
{

        double center = ((Fe+Fb)/2.0)/((Fs/decim)/2.0);
    double pass = (Fe-Fb)/((Fs/decim)/2.0);
        // allocate memory for the impulse response
        h.resize(num_taps, 0.0);
        for(auto i = 0 ; i < num_taps ; i++){
                if(i == (num_taps-1)/2){
                        h[i] = pass;
                }else{
                        h[i] = pass * sin( PI * (pass/2) *( i-(num_taps-1)/2 ) ) / ( PI * (pass/2) *
                }
        h[i] = h[i] * cos(i*PI*center);
                h[i] = h[i] * (sin(i * PI / num_taps)*sin(i * PI / num_taps));
                //printf("h[%d] = %f\n",i,h[i]);
        }

}
```

Ben and Caleb's branch (Commit 14 days ago). **Note** the curly bracket stylization matches Caleb's typical approach.

Notice the methods striking resemblance to Ben and Caleb's Lab 2 impulseResponseLPF:

main — lab2-group07-tuesday / src / blockProcessing.cpp — Go to file

gannoCal Final - 1st — Latest commit a19da9c on Jan 30 — History

2 contributors

258 lines (216 sloc) | 9.61 KB — Raw | Blame

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>

#define PI 3.14159265358979323846

// function for computing the impulse response (reuse from previous experiment)
void impulseResponseLPF(float Fs, float Fc, unsigned short int num_taps, std::vector<float> &h)
{

        float cutoff = Fc/(Fs/2);
        // allocate memory for the impulse response
        h.resize(num_taps, 0.0);
        for(auto i = 0 ; i < num_taps ; i++){
                if(i == (num_taps-1)/2){
                        h[i] = cutoff;
                }else{
                        h[i] = cutoff * sin( PI * cutoff *( i-(num_taps-1)/2 ) ) / ( PI * cutoff *( i-(num_taps-1)/2 ) );
                }
                h[i] = h[i] * (sin(i * PI / num_taps)*sin(i * PI / num_taps));
                printf("h[%d] = %f\n",i,h[i]);
        }

}
```

Group 07

John and Sawoud's branch (Commit 12 days ago).

Also

1. John and Sawoud are claiming contributions the RDS path. Please note that these methods are not included in the final version.
2. John and Sawoud are claiming contribution to Front end/Mono audio path. Note that if inspected in the GitHub timeline, Ben and Caleb were ultimately responsible for the final working intermediary model (pre-threading), alongside the final front end model.
    a. See `cc3ba62a914b93715616d30b261e924b7a3864e4` and `c4bce3ef5e74570d94a0f6ed6df06c07887d5500` for pre-threading
    b. See latest commit to main for final front end model.

Project Timeline

Early stages of project:

1. Both groups met as a whole, occasionally.
2. Mode 0 Mono initially had contributions from both groups.
    a. John and Sawoud separated IQ into I and Q samples.
    b. Ben and Caleb were responsible for the decimator.
    c. Ben and Caleb were responsible for block processing.
    d. Ben and Caleb were responsible for the debugging, optimization and integration of the source code.
3. Mode 0 stereo was modeled, built, optimized, and integrated in its entirety by Ben and Caleb.

   a. Ben and Caleb add command line inputs.
4. The re-sampler used in mode 1 was created by John and Sawoud.
   a. Ben and Caleb were responsible for its integration to the source code.

Mid stages of project:

1. Ben and Caleb were entirely responsible for implementing Front-end to Audio threading.

At this point it was March 23rd. The project group met and decided to divide and conquer RDS.

Late stages of the project:

1. Ben and Caleb completed the entire RDS python model by the 25th.
2. John and Sawoud did not reach out until the 28th.
3. At this point, Ben and Caleb had built an un-optimized version of C++.
4. Ben and Caleb asked John and Sawoud to help optimize and debug the current working model.
5. **John and Sawoud have made no commits containing updates or fixes to the current working model (as of April 1st).**
6. Ben and Caleb continued to independently finish RDS C++ (Including threading, and a redesigned re-sampler).

Final stages of the project.

1. Ben and Caleb are responsible for integrating all existing models into a real-time source code bundle and optimizing parameters for improved performance on the Pi.

Contribution Spreadsheet

Project Section	Contribution	Included in Final Draft?	File Location in SC	Branch History
Mono 0	Python Model	Yes	/model	Provided by Prof. Nicolici
	C++ Lowpass & Convolution	Yes	filter.cpp	From Labs.
	Debugging			Caleb & Ben
	Integration w/ source code			Caleb & Ben
Mono 1	Python Model	No	/model	
	C++ Resample	Yes	filter.cpp	John&Sawoud

	Debugging			Caleb & Ben
	Integration w/ source code			Caleb & Ben
Stereo	Python Model	Yes	/model	Caleb & Ben
	C++ Band-pass	Yes	filter.cpp	Caleb & Ben
	C++ PLL	Yes	fmPll.cpp	Caleb & Ben
	Debugging			Caleb & Ben
	Integration w/ source code			Caleb & Ben
RDS	Python Model	Yes	/model	Caleb & Ben
	C++ Extraction + Recovery	Yes	experiment.cpp	Caleb & Ben
	C++ Demodulation	Yes	experiment.cpp	Caleb & Ben
	C++ Data Processing	Yes	filter.cpp	Caleb & Ben
	Debugging			Caleb & Ben
	Integration w/ source code			Caleb & Ben
Threading	C++ Methods	Yes	experiment.cpp	Caleb & Ben
	Debugging			Caleb & Ben
	Integration w/ source code			Caleb & Ben
Major Debugging	Final Integration			Caleb & Ben
	Code and package cleanup			Caleb & Ben