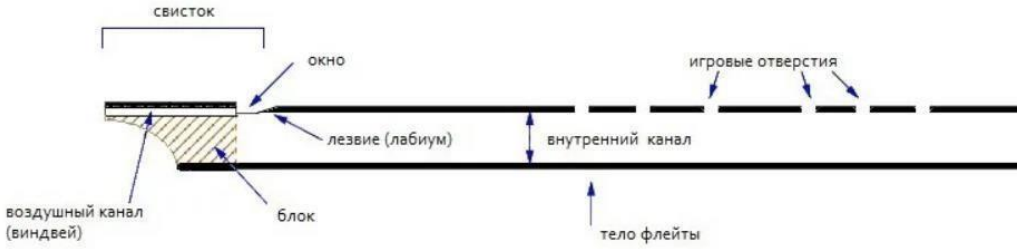
Дз

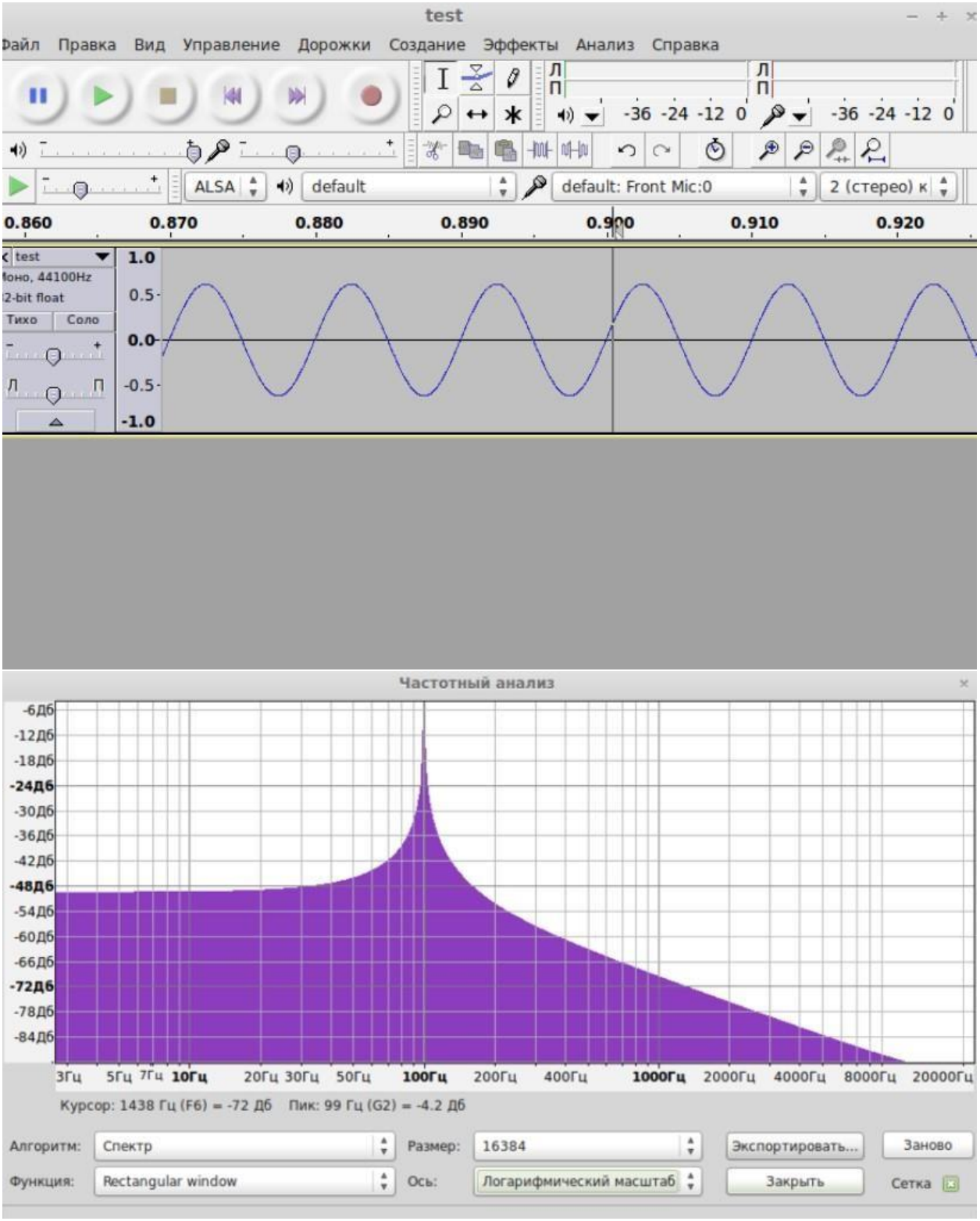
1. аккордеон: при сжатии/разжатии мех поступает воздух. Он становится причиной колебаний язычков, для которых играющий человек открыл клапаны(открываются в зависимости от нажатой клавиши). В язычке есть металические планки, как раз они и колеблются. в зависимости от размера язычка будут разные ноты: высокие при маленьком язычке, низкие при толстых и тяжелых пластинах
   флейта: Свисток состоит из воздушного канала и лезвия. Канал формируется блоком и телом флейты или кольцом, а на лезвие попадает поток воздуха из канала. Из за этого происходит звук. Итак, попадая на лезвие (лабиум), струя воздуха рассекается: часть воздуха вылетает наверх, а другая начинает вибрировать, создавая колебания воздуха, который находится в канале флейты. Воздух в канале работает примерно как струна: закрыв все игровые отверстия, получается базовый тон флейты. Открывая игровые отверстия последовательно снизу, мы укорачиваем звучащую длину воздушного столба в канале — звук становится выше. Так же как и лады на струнных инструментах, игровые отверстия флейты могут располагаться в разных местах, тем самым определяя звукоряд инструмента.



   Саксофон: воздух поступает когда музыкант дует в мундштук, Трость после этого прижимается к краям мундштука. Пошла первая фаза колебаний трости. Распространяясь по трубке саксофона, воздушная волна доходит до звуковых отверстий или раструба инструмента. Звуковые отверстия открывает музыкант нажатием. Чем шире открываемое отверстие, тем ниже звук. Затем, не изменяя своей фазы, волна отражается в обратном направлении. В канале саксофона образуется стоячая звуковая волна. Дойдя до трости, вибрирующий воздух заставляет ее возвратиться в перво-начальное положение, завершая тем самым вторую фазу и весь цикл колебательного процесса. За это время воздушная волна успевает пробежать по каналу саксофона путь в прямом и обратном направлении.
2. Реальная звуковая волна в треке отличается от просто гармонической волны, это будет комбинация нескольких волн с разными частотами. Однако для распознавания трека не нужно хранить всю эту информацию(иначе поиск был бы слишком долгим). Необходимо разложить сложную волну на отдельные частоты и узнать громкость(амплитуду) каждой из них, для этого используют преобразования Фурье. Получаем спектограмму, обрезаются частоты свыше 5000 – 4000 Гц, потому что основная информации о мелодии, нужная для распознавания, находится именно в этих пределах. Затем самые яркие частоты в определенный момент времени помечаются, это называется recorder fingerprint. Для каждой песни в базе данные сделан такой отпечаток. Когда пользователь записывает кусочек песни, который хочет распознать, то приложение получает его fingerprint, после

чего база данных находит трек с похожим fingerprint(это делается с помощью хеш таблицы для оптимизации, ключ - частота), он и будет результатом

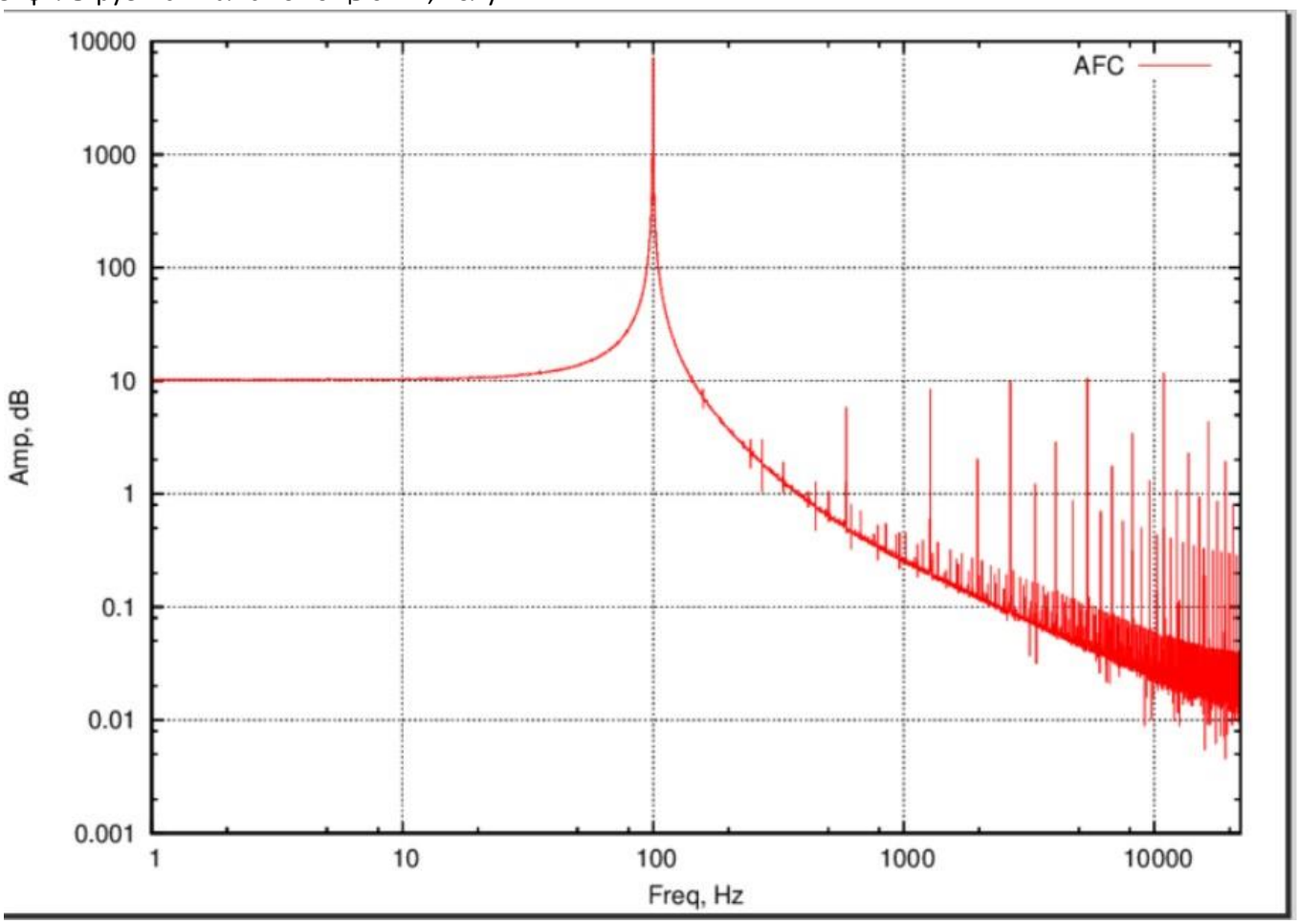
3. Буду делать это с помощью программы на си, которая пишет информацию в wav файл, анализировать его и строить спектограмму буду с помощью audacity



Добавила шум:

```c
int main(int argc, char * argv[])
{
    int i;
    //float amplitude = 32000;
    float amplitude = 32000; // 32767/(10%amp+5%amp+100%amp)
    float freq_Hz = 100;

    srand( _Seed: (unsigned int)time( _Time: 0));

    //short int meandr_value=32767;

    /* fill buffer with a sine wave */
    for (i=0; i<BUF_SIZE; i++)
    {

        //buffer[i] = (int)(amplitude/10 * sin((float)(2*M_PI*i*freq_Hz/S_RATE))); //10%amp
        //buffer[i] +=(int)(amplitude/20 * sin((float)(2*M_PI*i*10*freq_Hz/S_RATE))); //5% amp
        buffer[i] +=(int)(amplitude * sin( _X: (float)(2*M_PI*i*freq_Hz/S_RATE))); //100% amp


        //buffer[i] +=(int)amplitude/10*d_random(-1.0, 1.0); //nois

    }
    write_wav( filename: "test.wav",  num_samples: BUF_SIZE, buffer, S_RATE);
```

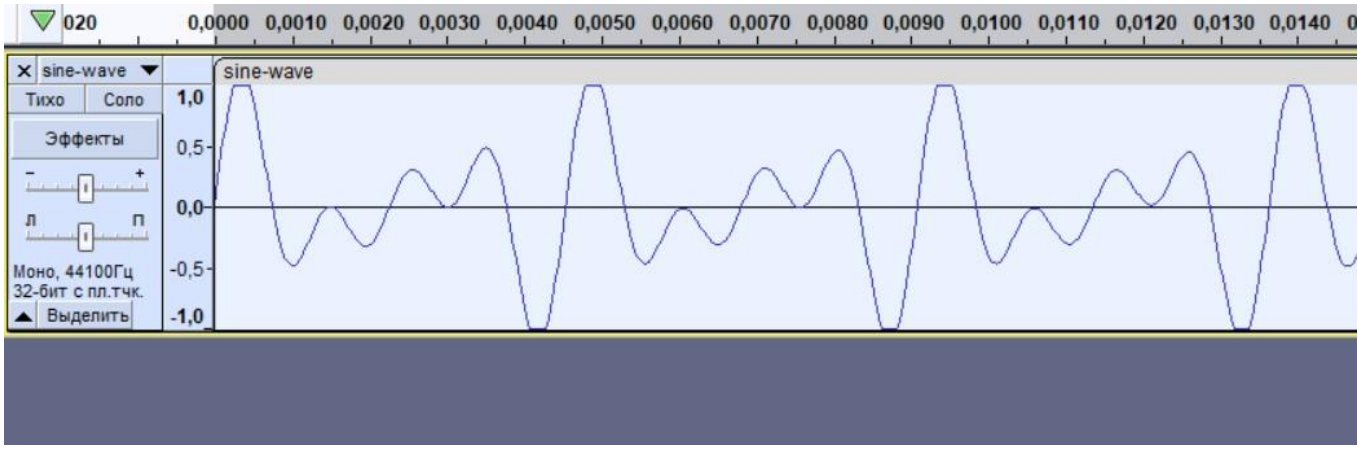
Отфильтруем сигнал с помощью ФП, получим:



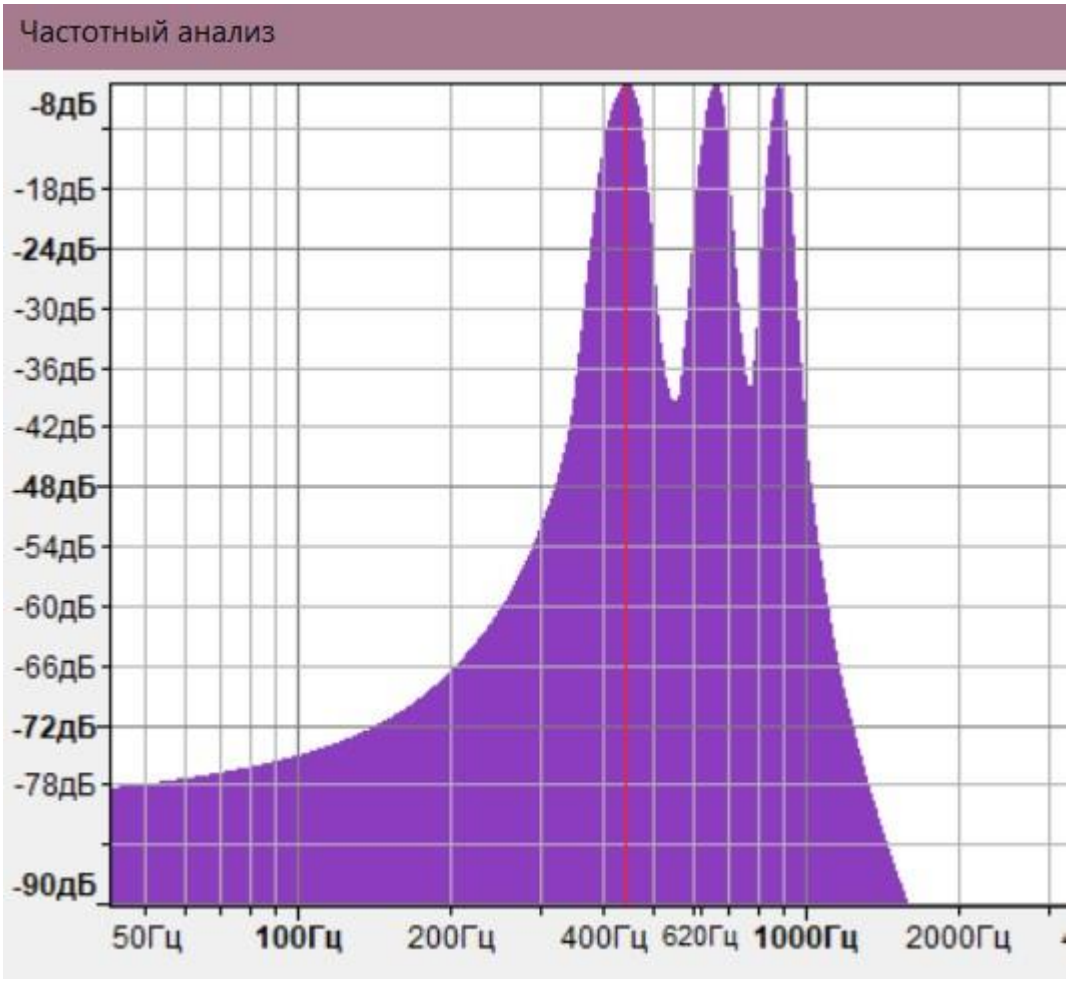
Сумма трех гармоник с одинаковой амплитудой и разными частотами соответственно 660, 880, 440 Гц

```
/* fill buffer with a sine wave */
for (i=0; i<BUF_SIZE; i++)
{

    buffer[i] = (int)(amplitude * sin( _X (float)(2*M_PI*i*(freq_Hz+220)/S_RATE)));
    buffer[i] +=(int)(amplitude * sin( _X (float)(2*M_PI*i*2*freq_Hz/S_RATE)));
    buffer[i] +=(int)(amplitude * sin( _X (float)(2*M_PI*i*freq_Hz/S_RATE)));

    //buffer[i] +=(int)(amplitude * sin((float)(2*M_PI*i*freq_Hz/S_RATE))); //100% amp

    //buffer[i] +=(int)amplitude/10*d_random(-1.0, 1.0); //pois
```

Получилась волна вида:

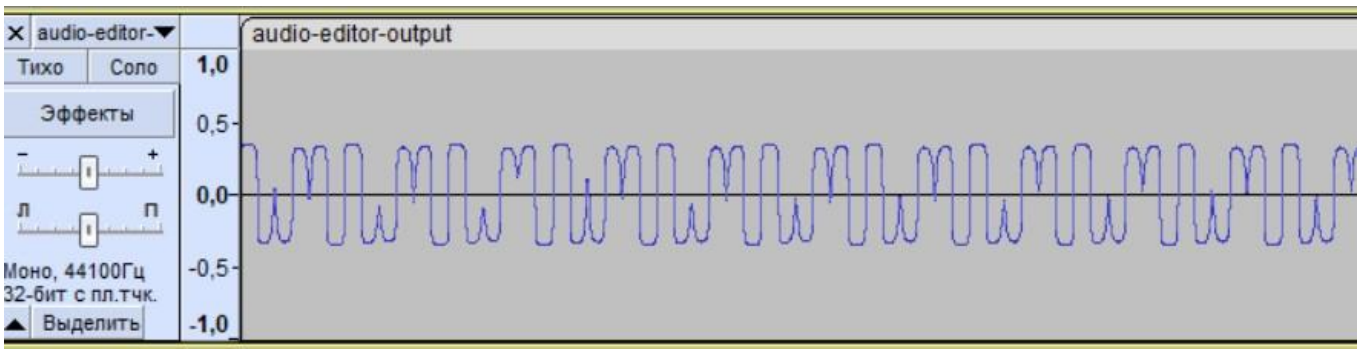


Анализ спектра:



Добавила шум:

```
//float amplitude = 32000;
float amplitude = 12000; // 32767/(10%amp+5%amp+100%amp)
float freq_Hz = 440;

srand( _Seed: (unsigned int)time( _Time: 0));

/* fill buffer with a sine wave */
for (i=0; i<BUF_SIZE; i++)
{

    buffer[i] = (int)(amplitude * sin( _X: (float)(2*M_PI*i*(freq_Hz+220)/S_RATE)));
    buffer[i] +=(int)(amplitude * sin( _X: (float)(2*M_PI*i*2*freq_Hz/S_RATE)));
    buffer[i] +=(int)(amplitude * sin( _X: (float)(2*M_PI*i*freq_Hz/S_RATE)));

    buffer[i] +=(int)amplitude/10*d_random( min: -1.0,  max: 1.0); //nois
```



Отфильтруем сигнал с помощью ФП, получим: