ICS - Trabalho II
Síntese Aditiva

Juarez Aires Sampaio Filho 11/0032829

Universidade de Brasília

4 de Junho de 2014

Requisitos

- Desenvolver uma interface gráfica em Java que implemente três instrumentos aditivos(ver figura a seguir) e seja capaz de tocar melodias e notas com ele. A interface deve possuir operações de:
  - Música: tocar uma melodia ou ruídos interessantes
  - Nota

Dispositivo RAN

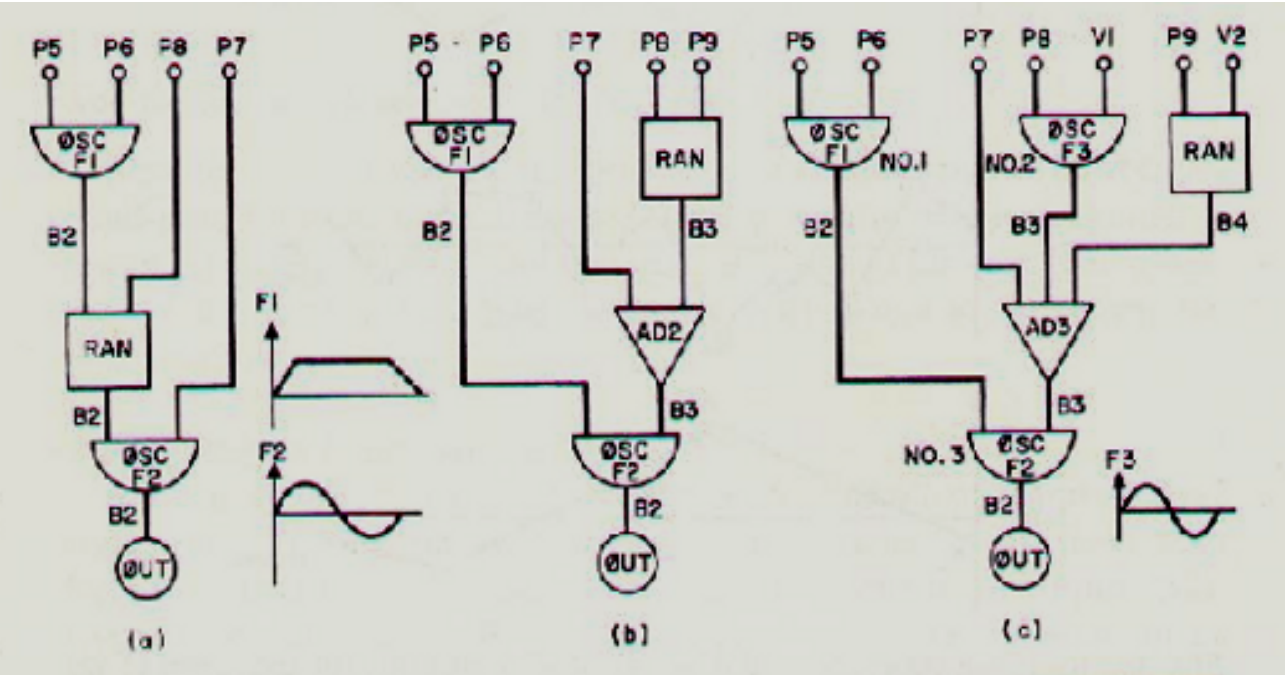

Figura: Instrumentos a serem implementados

RAN

- Vemos pelas especificações que **RAN precisa aceitar** como entrada tanto uma **constante** como um **dispositivo**.
- Dentre as classes disponíveis na API SomA, escolhemos fazer isso utilizando como base a classe **Oscilador**
- Essa é a escolha natural para ter como classe base
- Gostaríamos de poder setar a tabela SIN da classe oscilador para aquela tabela gerada com números aleatório. Como isso não é possível, **foi necessário uma gambiarra**.
- **A frequência do sin é setada para 0 e sua fase para 90**. Desta forma temos um sinal constante em $+1$
- a entrada da amplitude é um dispositivo multiplicador
- as entradas desse multiplicador são duas envoltórias: Uma é gerada aleatoriamente em -1 e $+1$ e a outra é uma envoltória de amplitude gerada pelo usuário

Dispositivo RAN

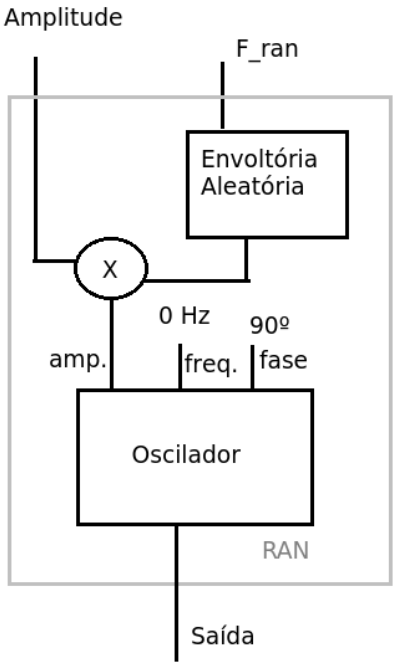

Figura: Dispositivo RAN: esquemático da construção

RAN

Classe desenvolvida para gerar a envoltória aleatória.

- extends **Oscilador**
- entradas controláveis:
  - f_ruido: define o número de amostras aleatórias geradas no intervalo
  - a: define a amplitude da envoltória gerada. Isto é, os números aleatórios vão de -A até +A. Pode ser uma constante ou um objeto Envoltoria.
- fórmula utilizada para gerar números entre -A e +A :

```
float random = 2f*A*((float)Math.random()) - A;
```

- possui método de visualização gerado com o pacote **jmathtool**

Dispositivo RAN - Detalhes

- A frequência f_ruido nos dá o intervalo com que os pontos aleatório são gerados em uma tabela onde a última posição é 720.
- Isto é, os pontos são gerados em intervalos de $\frac{720}{f\_ruido}$
- Ou seja, f_ruido define o número de amostras aleatórias presentes na duração de **toda** a envoltória.
- **A sensação do ruído, no entanto, é sentida pela quantidade de pontos aleatórios em 1 segundo**.
- Para mantermos a mesma sensação, é **necessário alterar a frequência do ruído** quando a duração da nota for alterada, **de modo que dentro de 1 segundo tenhamos o mesmo número de pontos aleatórios** que tínhamos antes.
- Usamos então de uma variável para f_ran_atual e outra para f_ran_base

Dispositivo RAN

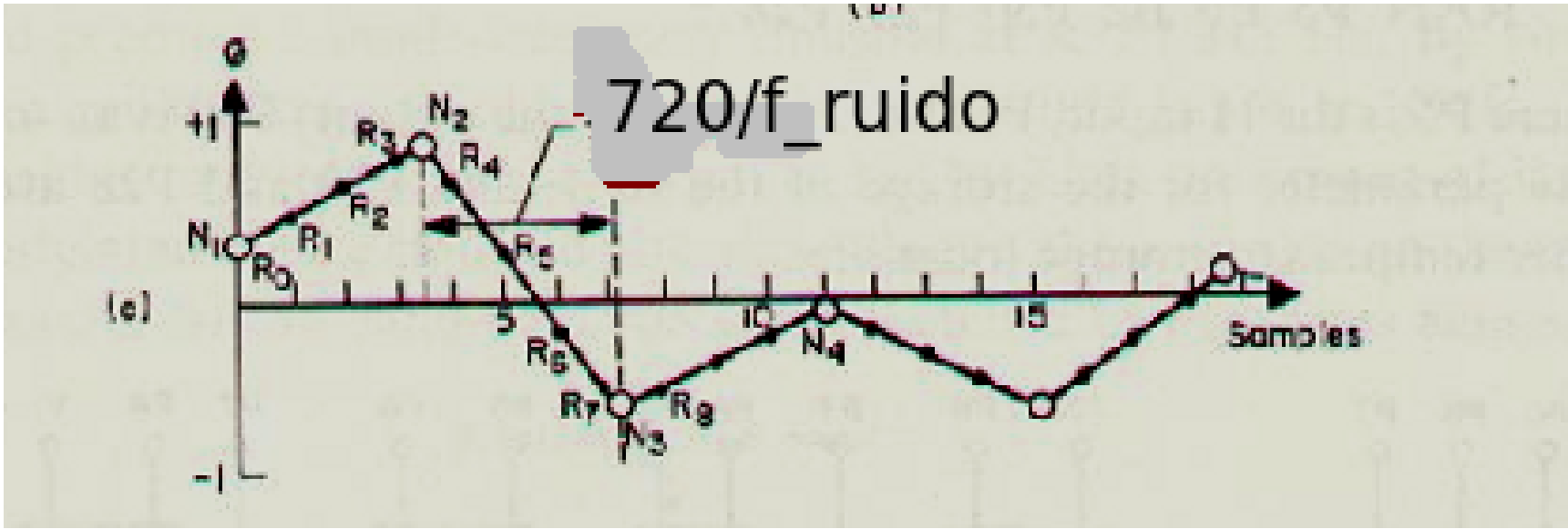

Figura: f_ruido define o intervalo com que os pontos aleatório são gerados

Dispositivo RAN

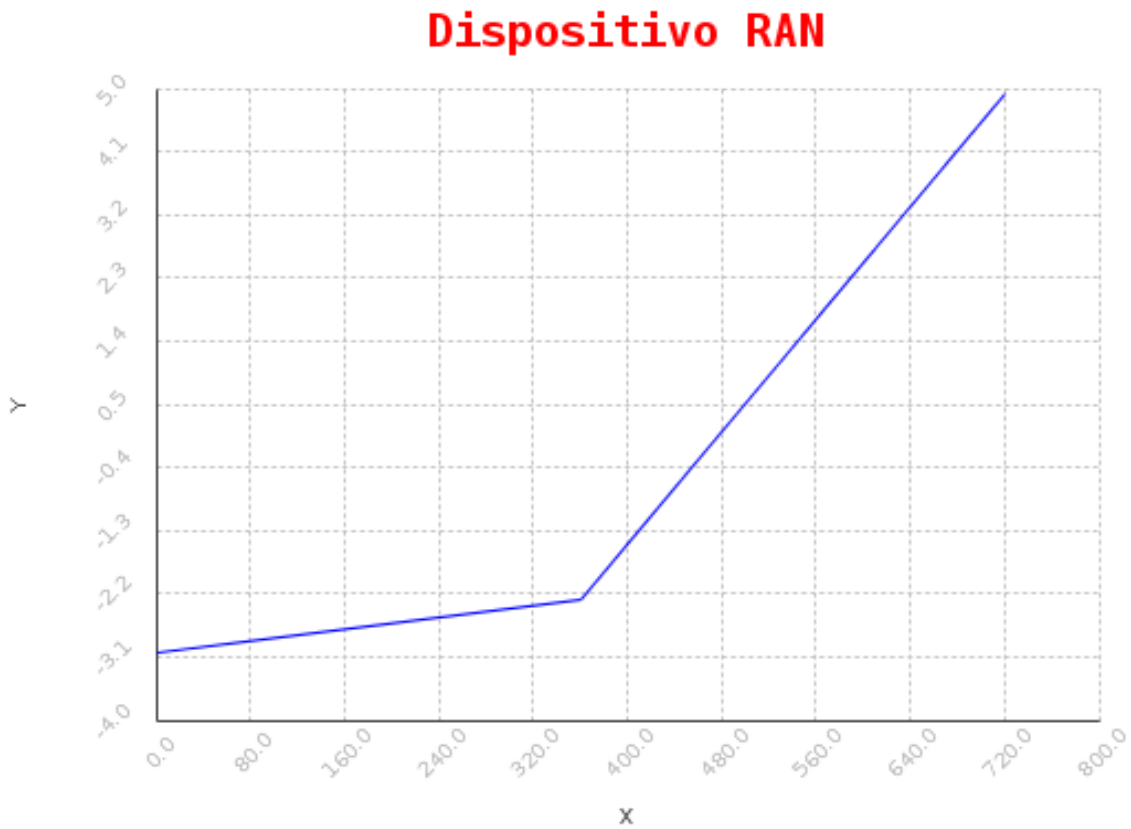

Figura: Dispositivo RAN, f_ran = 2, A = 10

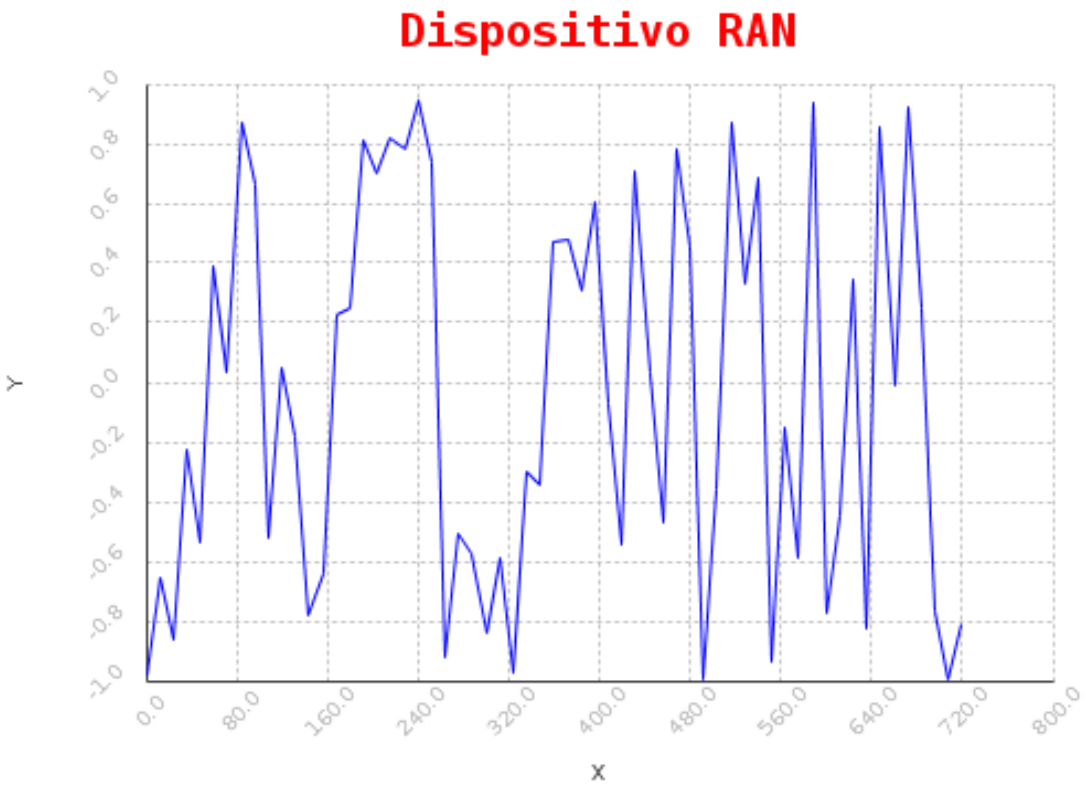

Figura: Dispositivo RAN, f_ran = 60, A = 1

RAN - detalhes

```
private void setRAN(){
setFrequencia(0);
setFase(90);
envFinal = new Multiplicador(randomEnv, ganhoEnv);
setDispositivoAmplitude(envFinal);
}
```

Figura: detalhes do método de configuração

RAN - detalhes

```
public void setDuracao(float d){
ganhoEnv.setDuracao(d);
this.duracao = d;
f_ruido = d*f_ruido_base;
generateRandomEnv();
randomEnv.setDuracao(d);
}
```

Figura: detalhes do método para configurar duração

Instrumento 1

- Extends **UnidadeH**
- Como a unidade H é o menor instrumento possível, nossos instrumentos são todos descendentes dela.
- Criamos uma unidadeH com **super()** e pegamos o seu oscilador.
- setamos então a entrada de ganho o oscilador para um dispositivo **RAN**
- redefinimos então os métodos **setGanho** e **setDuracao** para atuarem no objeto **RAN**

Instrumento 1 - construção

```
public Instrumento1(){
super();
initialize(); //inicializa as variaveis

setH(1.0f);
setGanho(1f);
setLambda(0.5f);

ran.setGanho(env);
ran.setGanho(10);
ran.setF_ruido(60);
osci.setDispositivoAmplitude(ran);
osci.setFrequencia(440);
osci.setDuracao(5f);
}
```

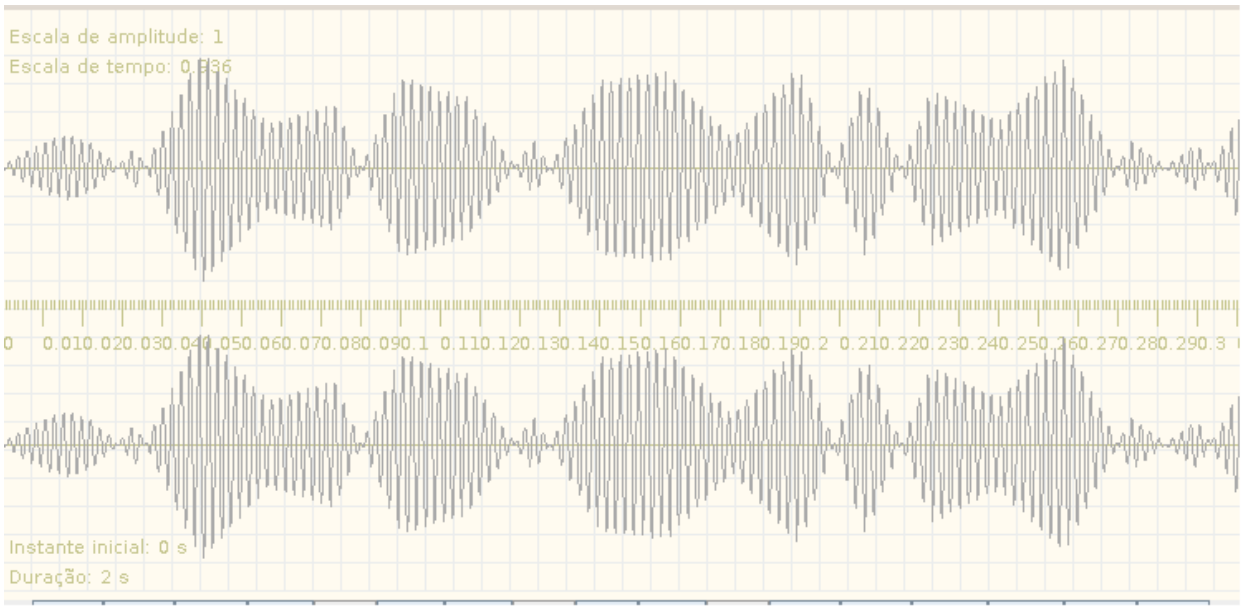

Figura: Saída do Instrumento 1

Instrumento 2

- Extends **UnidadeH**
- **Basta conectar os blocos**.
- A figura a seguir ilustra a conexão entre os blocos.

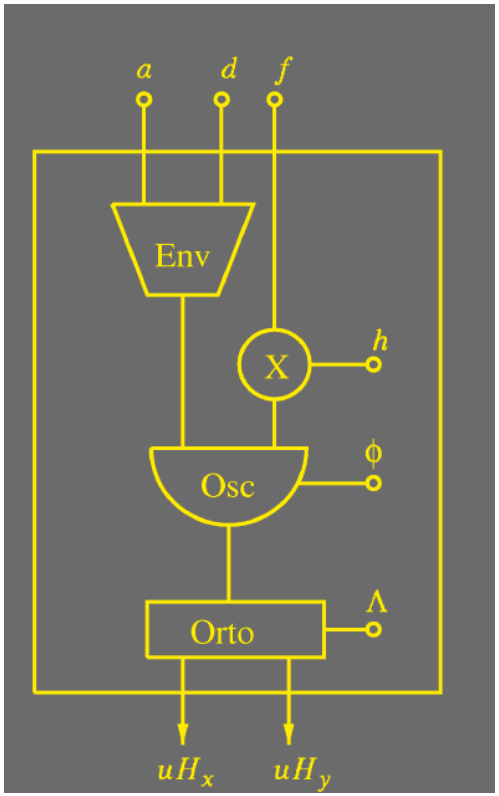

Figura: Unidade H padrão: no trabalho desenvolvido, não utilizamos a env padrão da unidade H

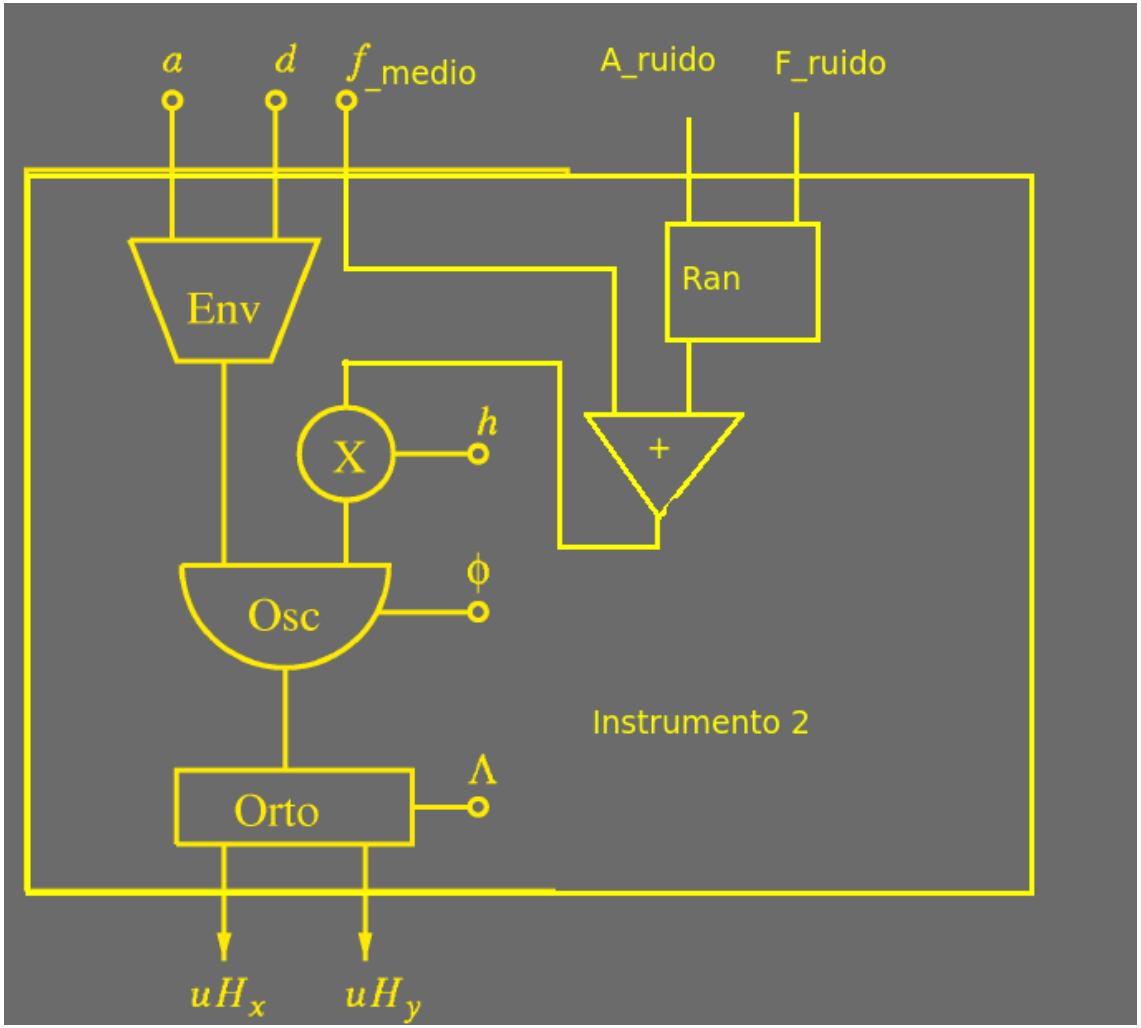

Figura: Esquemático do instrumento 2 tendo como base a unidade H

Instrumento 2 - construção

```
public Instrumento2(){
super();
setLambda(0.5f);
ran = new RAN(20f, 10);
f_medio = 440;
f_medio_env = constantEnvoltoria(f_medio);
sum = new Somador(f_medio_env, ran);

//define envoltoria padrao
Curva curva = new Curva(720);
curva.addPonto(0f, 0f);
curva.addPonto(60f, 1000f);
curva.addPonto(450f, 1000f);
curva.addPonto(720f, 0f);
ganhoEnv = new Envoltoria();
ganhoEnv.setCURVA(curva);


osci = getOscilador();
osci.setDispositivoAmplitude(ganhoEnv);
osci.setDispositivoFrequencia(sum);
}
```

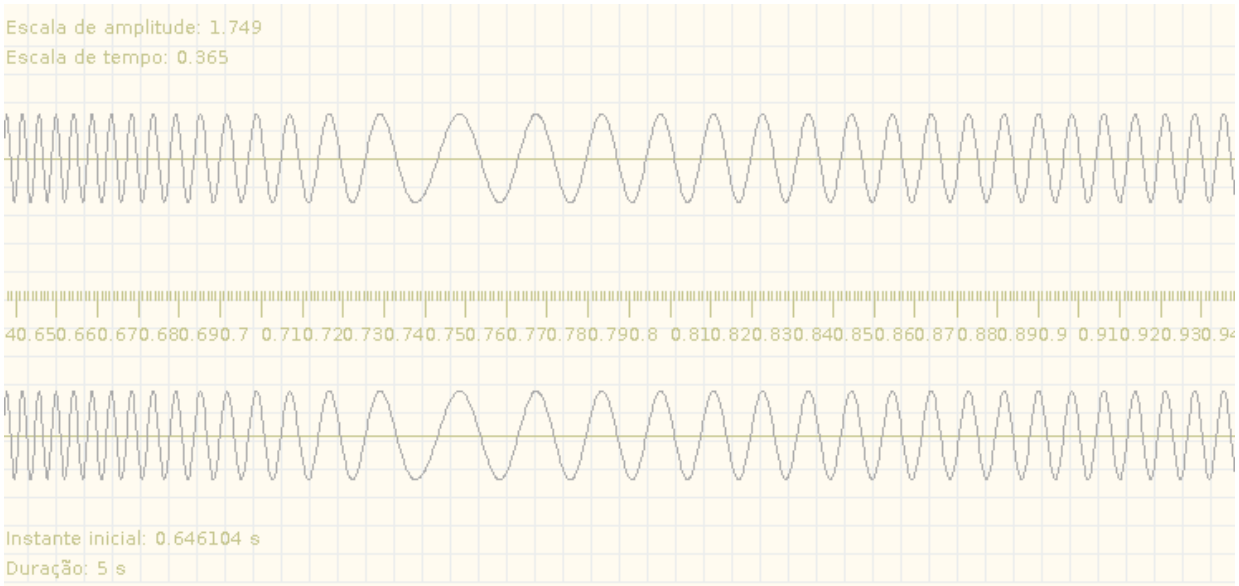

Figura: Saída do Instrumento 2

Instrumento 3

- Extends **UnidadeH**
- Como no item anterior, **basta conectar os blocos necessários**.
- Pode ser visto como uma extensão do Instrumento 2. As classes são muito semelhantes.
- Apenas acrescentamos um somador e um oscilador ao instrumento 2.

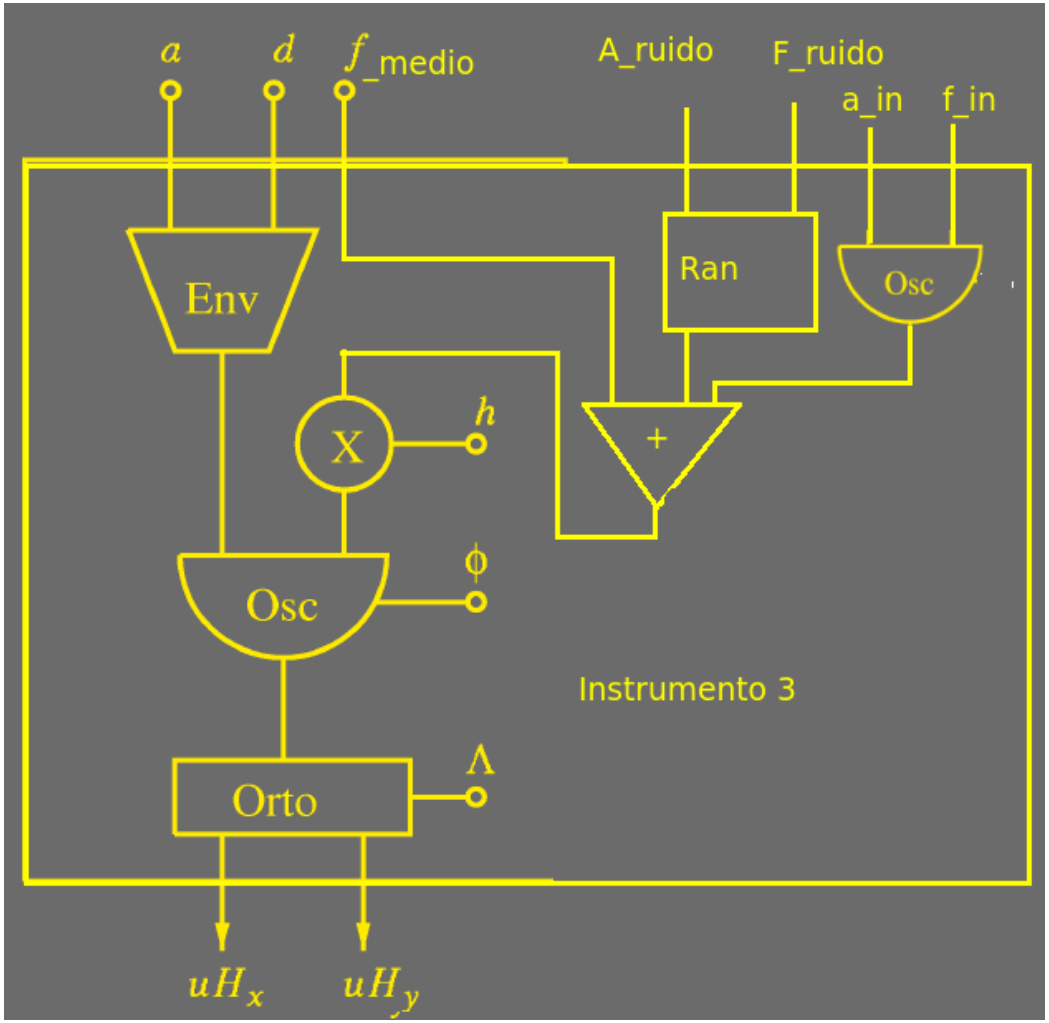

Figura: Esquemático do instrumento 3 tendo como base a unidade H

Instrumento 3 - construção

```
public Instrumento3(){
//propriedades da unidadeH
super();
setLambda(0.5f);
//RAN
ran = new RAN(100f, 10);//RAN(float amplitude, float new_f_ran)
//Frequencia media
f_medio = 440;
f_medio_env = constantEnvoltoria(f_medio);
//F medio + RAN
sum1 = new Somador(f_medio_env, ran);
osciF = new Oscilador(1, 1, 0); //Oscilador(float a, float f, float p)
sum2 = new Somador(sum1, osciF);
//define envoltoria padrao
Curva curva = new Curva(720);
curva.addPonto(0f, 0f);
curva.addPonto(60f, 1000f);
curva.addPonto(450f, 1000f);
curva.addPonto(720f, 0f);
ganhoEnv = new Envoltoria();
ganhoEnv.setCURVA(curva);
osci_out = getOscilador();
osci_out.setDispositivoAmplitude(ganhoEnv);
osci_out.setDispositivoFrequencia(sum2);
}
```

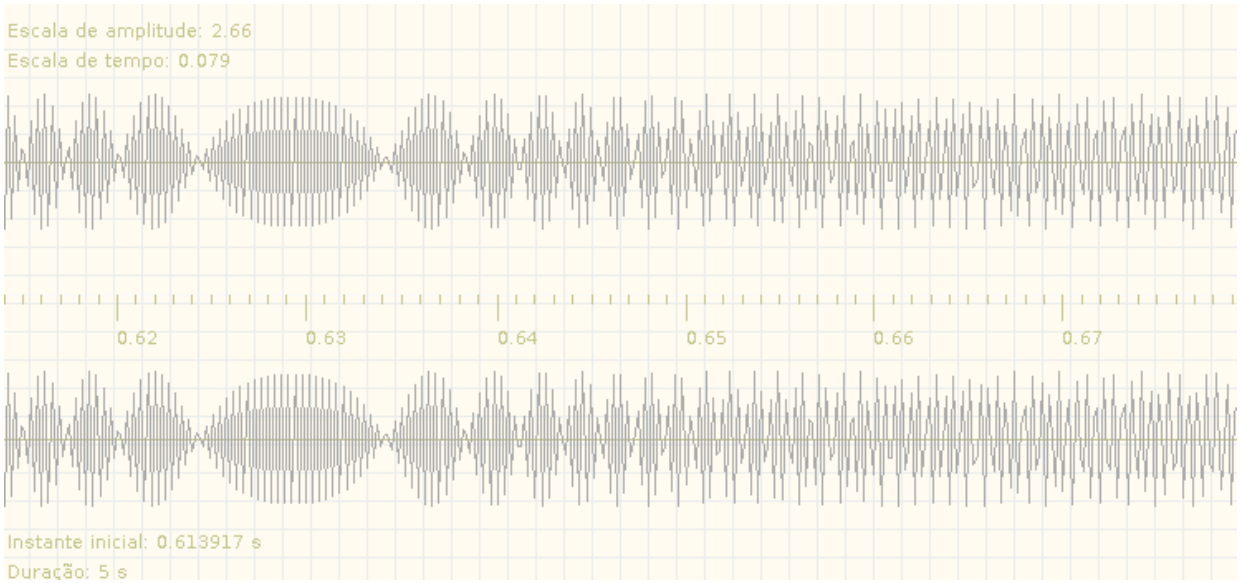

Figura: Saída do Instrumento 3

Instrumento

- Uma interface que os dispositivos anteriores devem interpretar
- Possui métodos básicos que todos os instrumentos devem possuir
- Facilita a manipulação de diferentes instrumentos dentro da interface gráfica

Interface Gráfica

Desenvolveu-se uma interface simples capaz de setar todos os parâmetros dos instrumentos, selecionar diferentes instrumentos e setar parâmetros pre-definidos para gerar sons interessantes

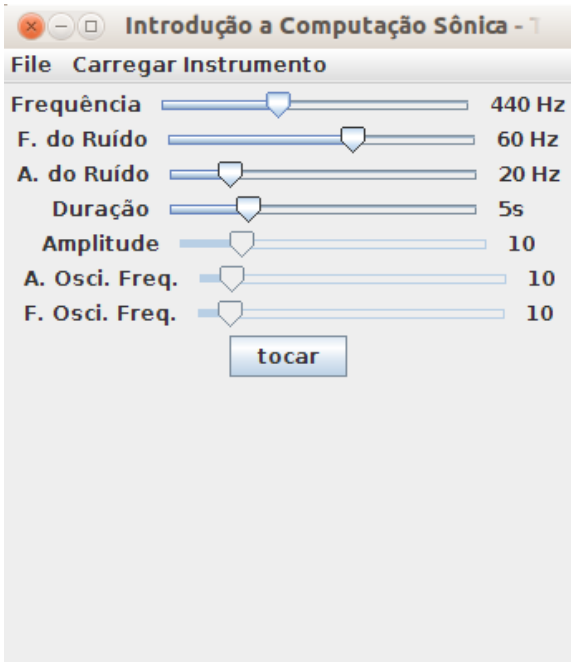

Figura: Interface gráfica desenvolvida

Documentação e Referências

- Documentação produzida com javadoc
- Documentação da API sintese
- API jmathplot