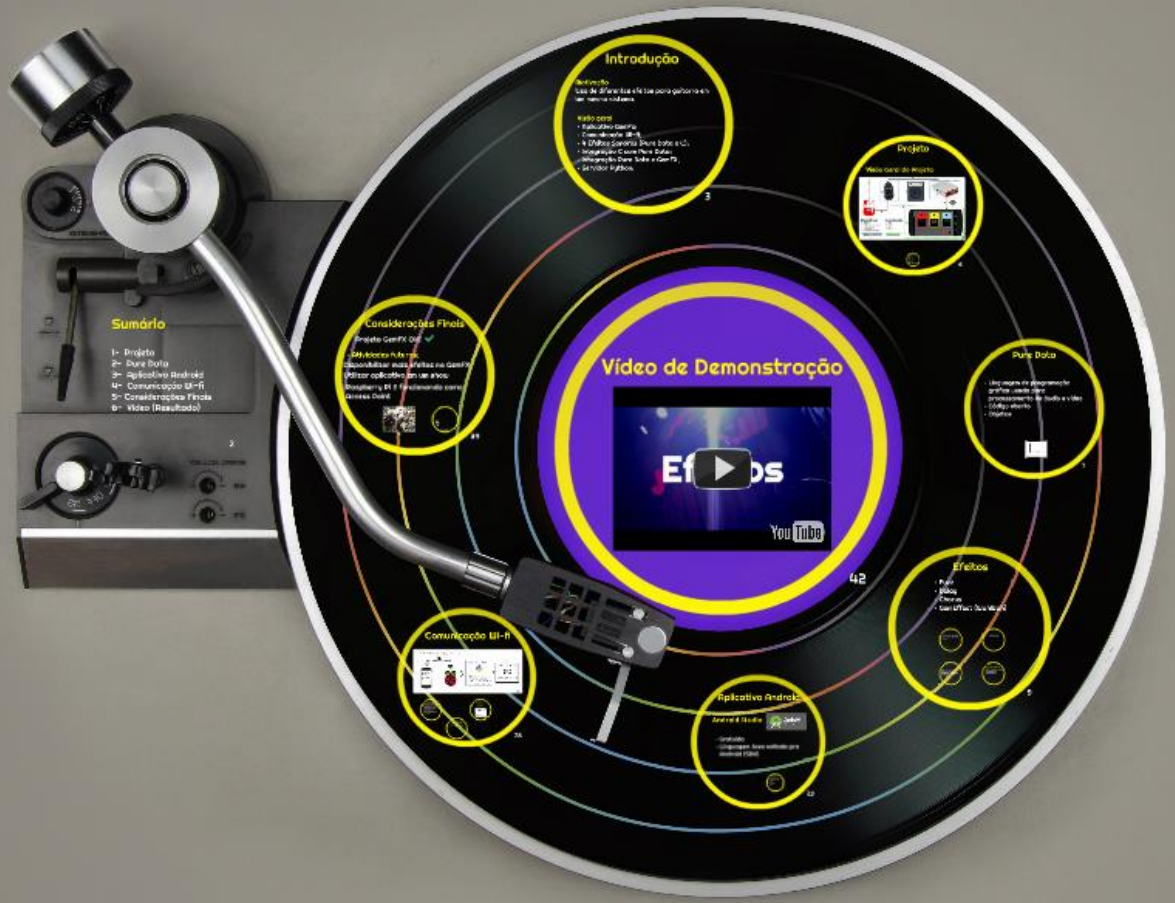
GemFx

Geanine Inglat
Mariana Carrião
Mariana Cabral



Sumário

Introdução

Motivação

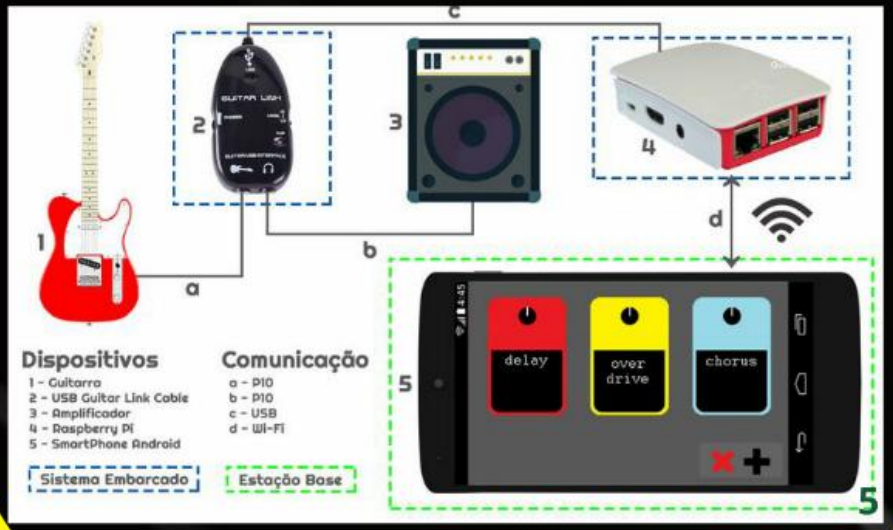
Uso de diferentes efeitos para guitarra em um mesmo sistema.

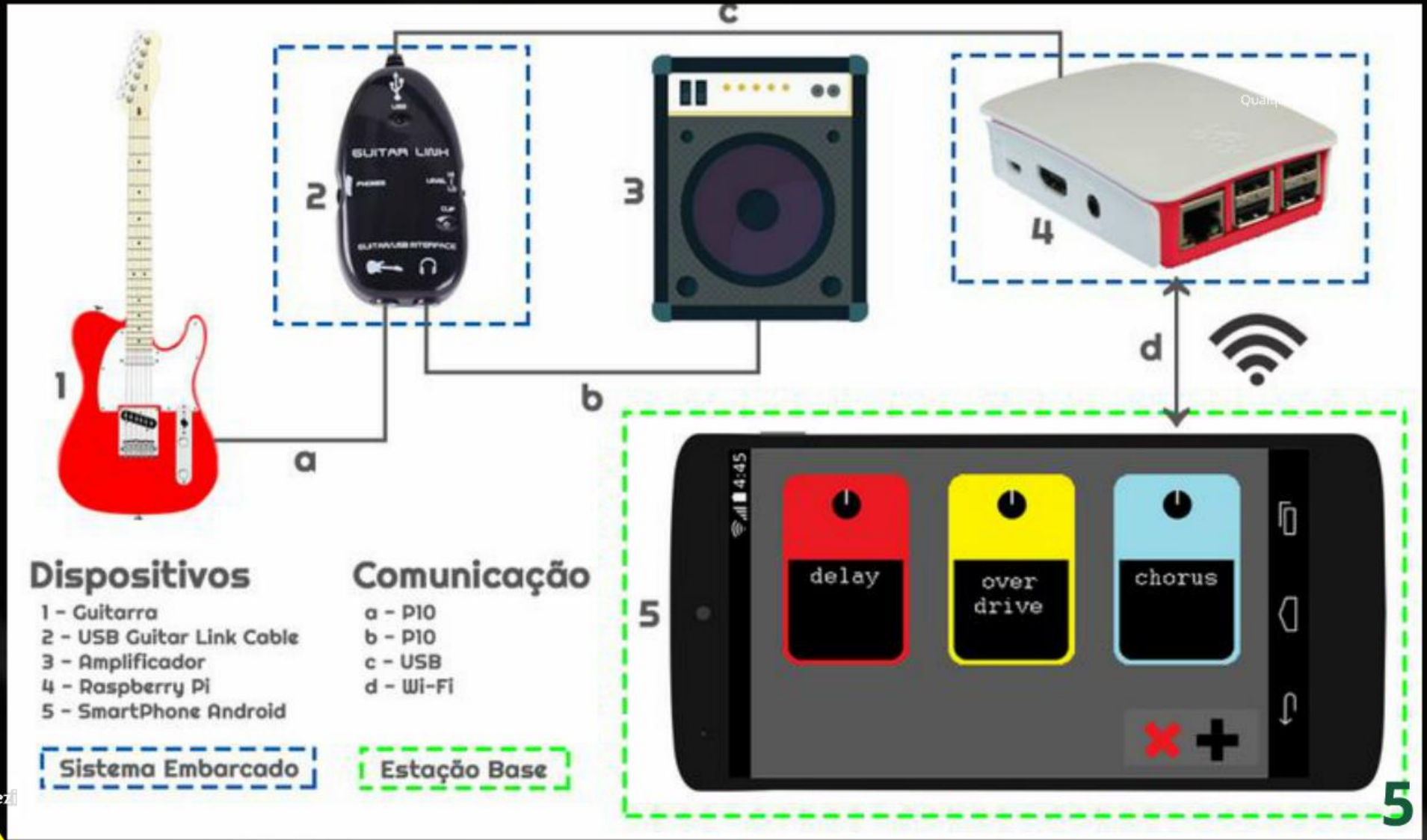
Visão geral

- Aplicativo GemFx;
- Comunicação Wi-fi;
- 4 Efeitos Sonoros (Pure Data e C);
- Integração C com Pure Data;
- Integração Pure Data e GemFX;
- Servidor Python;

Projeto

Visão Geral do Projeto




c
2
3
4
1
b
a
d
5
delay
over drive
chorus
4:45
Dispositivos
1 - Guitarra
2 - USB Guitar Link Cable
3 - Amplificador
4 - Raspberry Pi
5 - SmartPhone Android
Comunicação
a - P10
b - P10
c - USB
d - Wi-Fi
Sistema Embarcado
Estação Base
5

Efeitos Desenvolvidos

Quatro efeitos:

- Delay (Pure Data)

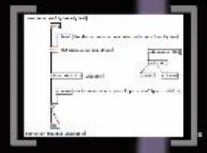
- Distortion (Pure Data)

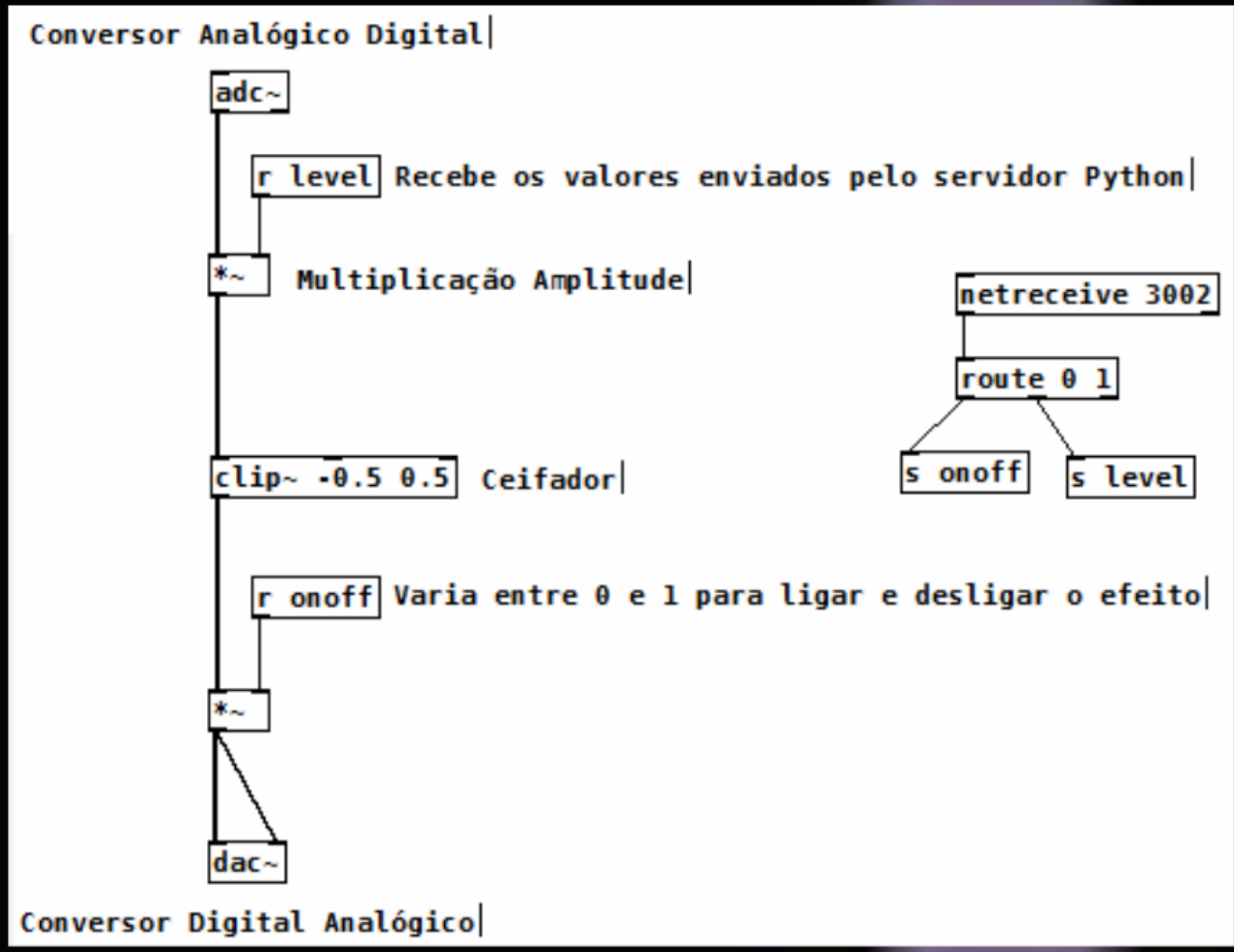
- Chorus (Pure Data)

- WahWah - GemEffect (Linguagem C)

Pure Data

- Linguagem de programação gráfica usada para processamento de áudio e vídeo
- Código aberto
- Objetos



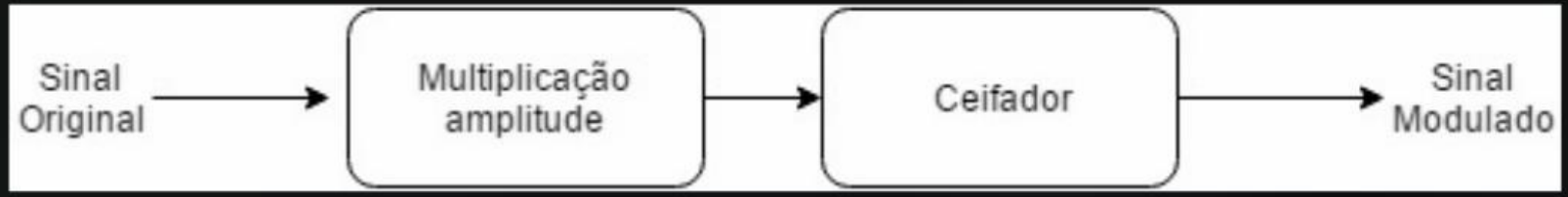

Conversor Analógico Digital
adc~
r level
Recebe os valores enviados pelo servidor Python
*~
Multiplicação Amplitude
netreceive 3002
route 0 1
s onoff
s level
clip~ -0.5 0.5
Ceifador
r onoff
Varia entre 0 e 1 para ligar e desligar o efeito
*~
dac~
Conversor Digital Analógico

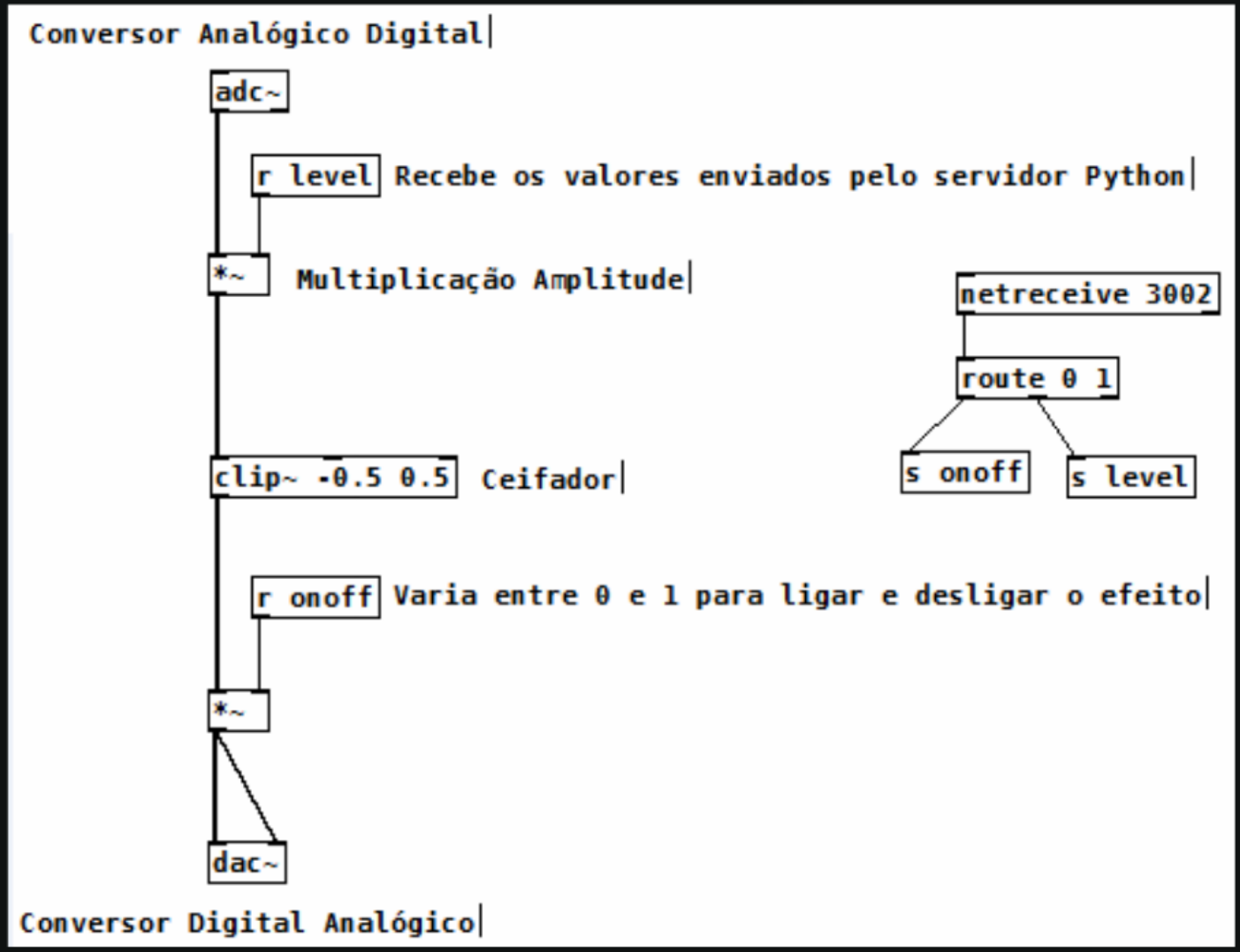
42

Efeitos

- Fuzz
- Delay
- Chorus
- GemEffect (WahWah)

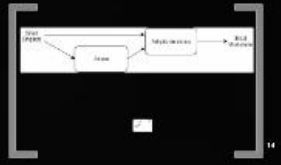
9

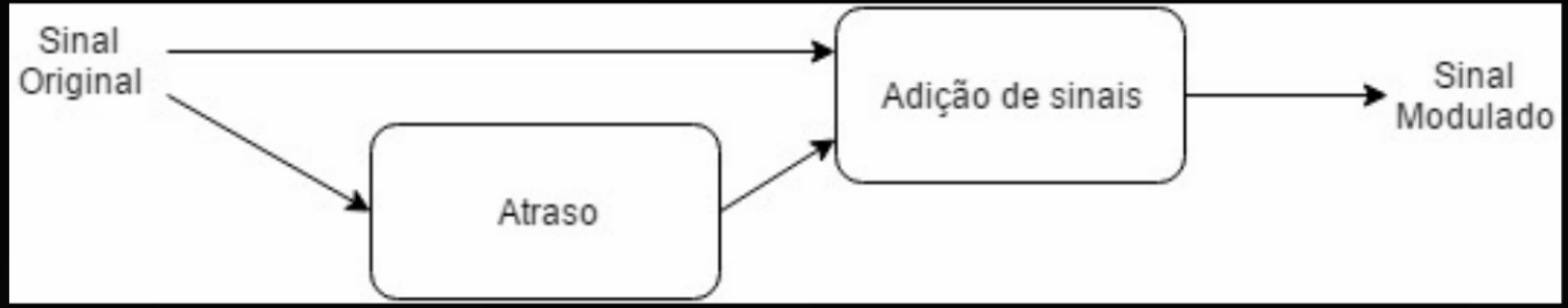

Sinal Original
Multiplicação amplitude
Ceifador
Sinal Modulado

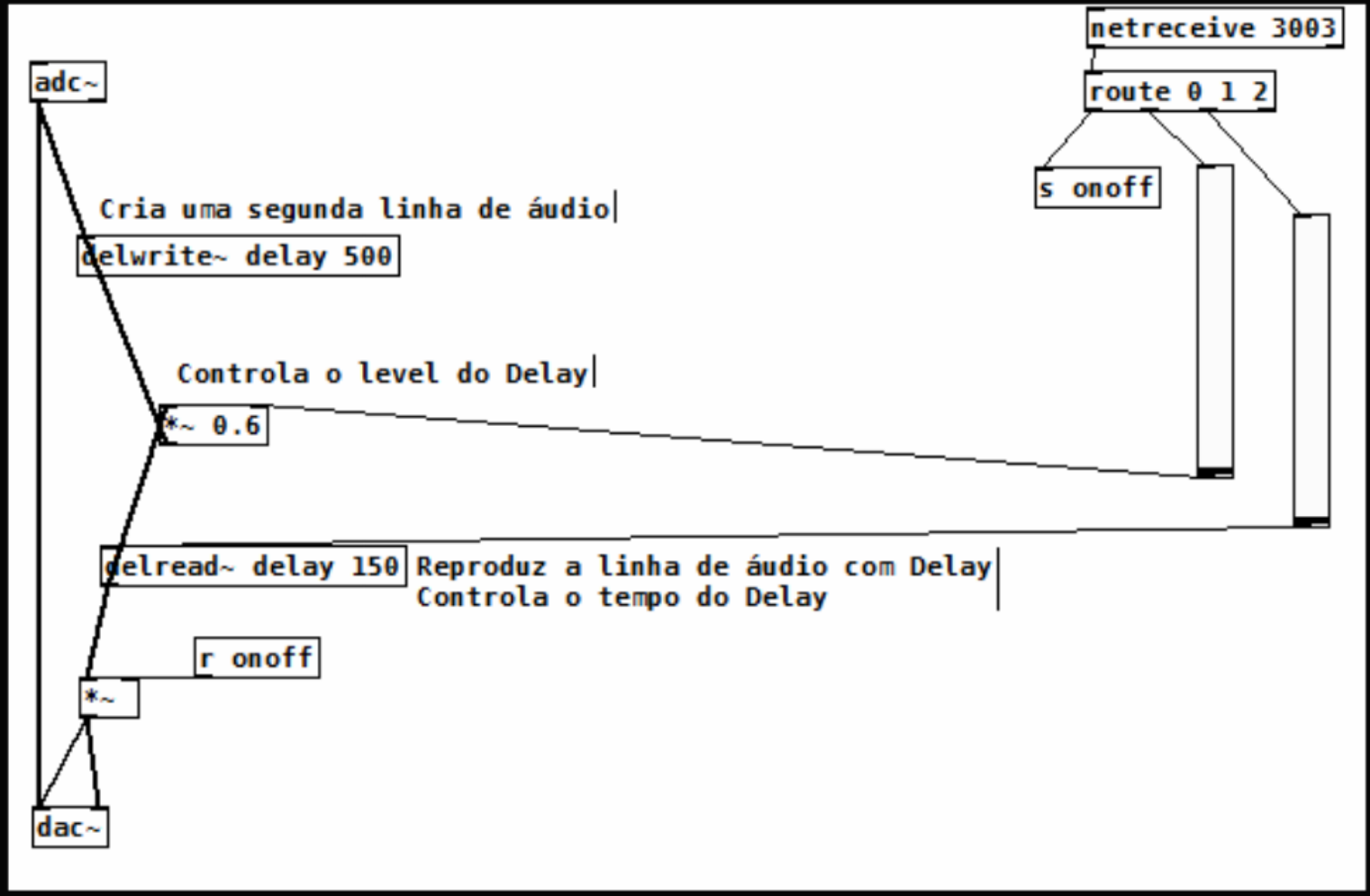

Conversor Analógico Digital
adc~
r level
Recebe os valores enviados pelo servidor Python
*~
Multiplicação Amplitude
netreceive 3002
route 0 1
s onoff
s level
clip~ -0.5 0.5
Ceifador
r onoff
Varia entre 0 e 1 para ligar e desligar o efeito
*~
dac~
Conversor Digital Analógico

Delay

Efeito de atraso

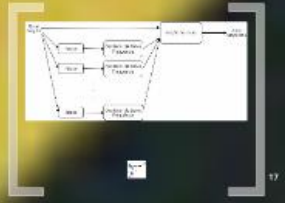


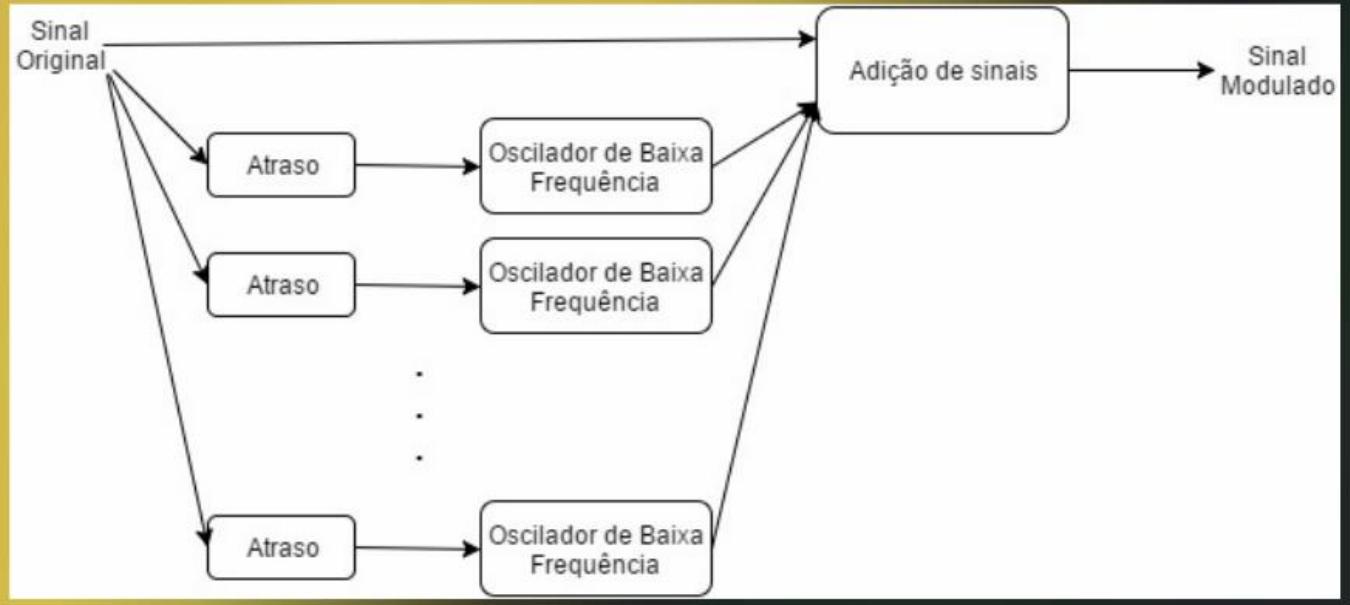

Sinal
Original
Atraso
Adição de sinais
Sinal
Modulado

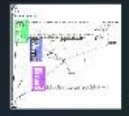

netreceive 3003
adc~
route 0 1 2
s onoff
Cria uma segunda linha de áudio
delwrite~ delay 500
Controla o level do Delay
*~ 0.6
delread~ delay 150
Reproduz a linha de áudio com Delay
Controla o tempo do Delay
r onoff
*~
dac~

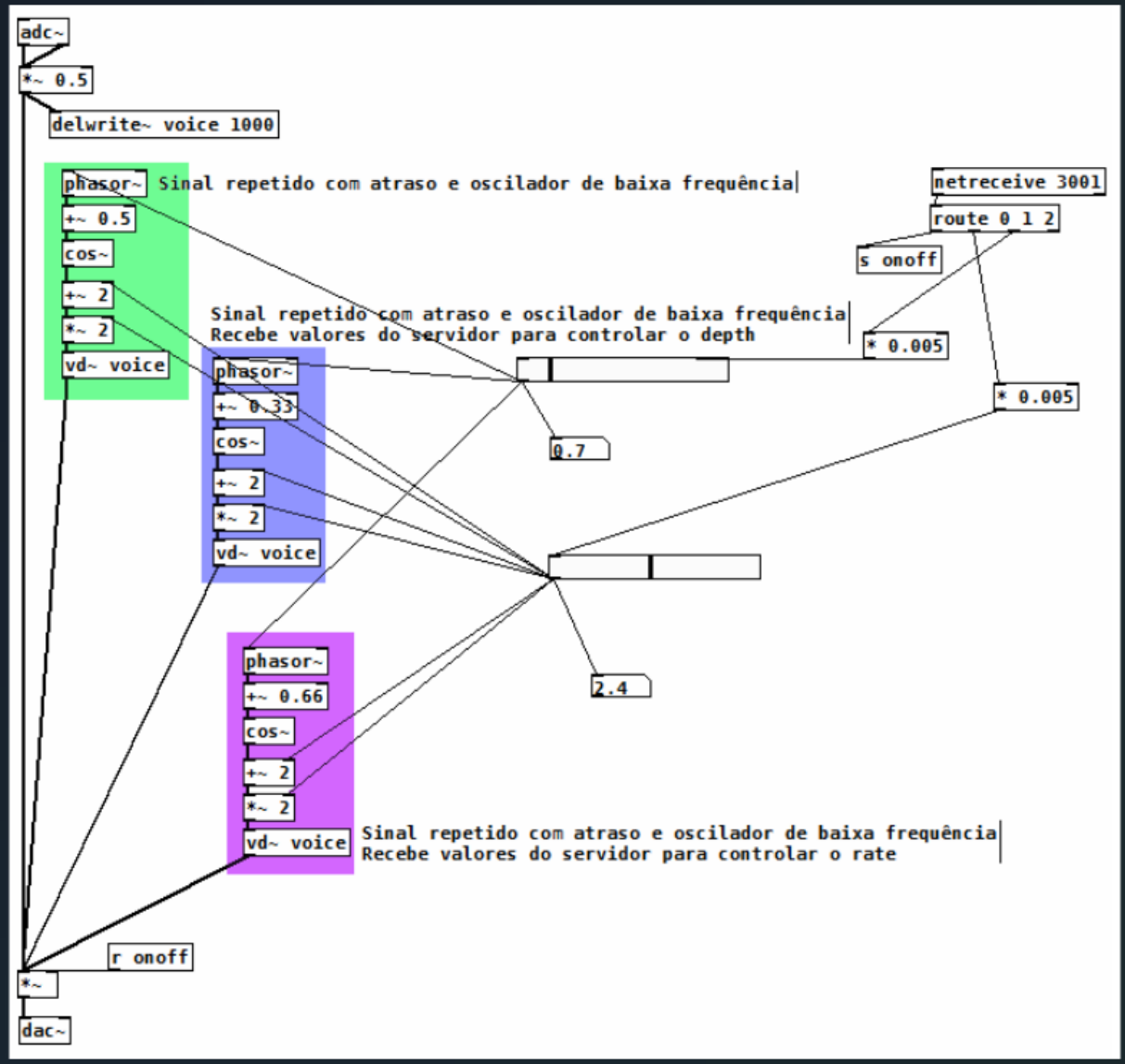
Chorus

Efeito que ocorre quando dois ou mais sinais de frequência e timbre aproximados são percebidos como apenas um sinal



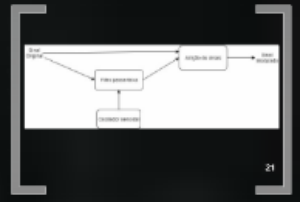

Sinal
Original
Atraso
Atraso
Atraso
Oscilador de Baixa
Frequência
Oscilador de Baixa
Frequência
Oscilador de Baixa
Frequência
Adição de sinais
Sinal
Modulado




adc~
*~ 0.5
delwrite~ voice 1000
phasor~
Sinal repetido com atraso e oscilador de baixa frequência
+~ 0.5
cos~
+~ 2
*~ 2
vd~ voice
netreceive 3001
route 0 1 2
s onoff
Sinal repetido com atraso e oscilador de baixa frequência
Recebe valores do servidor para controlar o depth
* 0.005
* 0.005
phasor~
+~ 0.33
cos~
+~ 2
*~ 2
vd~ voice
0.7
2.4
phasor~
+~ 0.66
cos~
+~ 2
*~ 2
vd~ voice
Sinal repetido com atraso e oscilador de baixa frequência
Recebe valores do servidor para controlar o rate
r onoff
*~
dac~

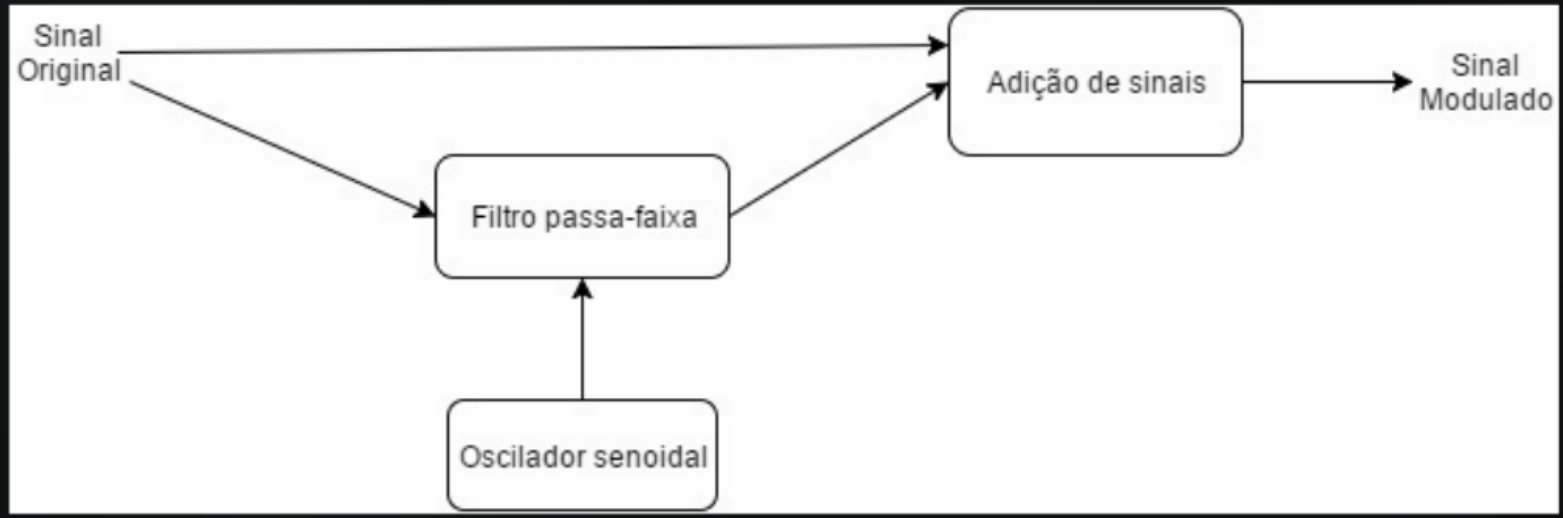
GemEffect (WahWah)

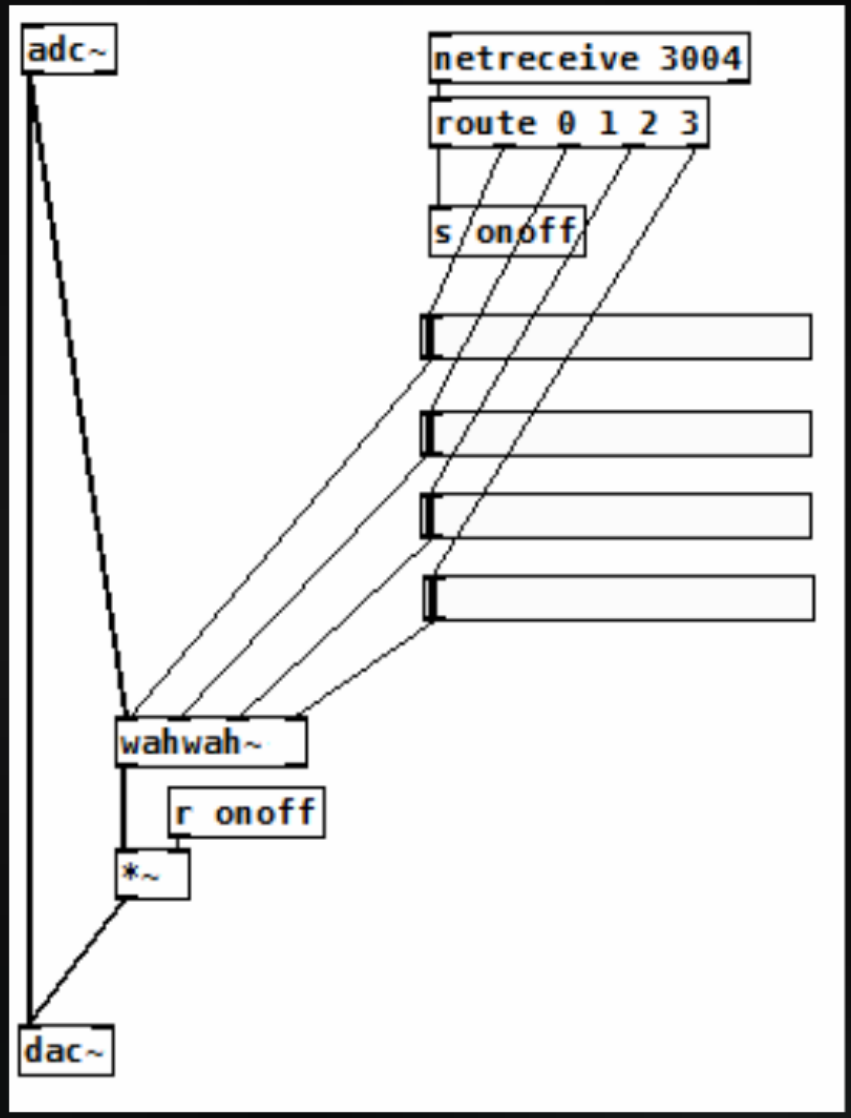
- Efeito de variação no tempo
- Filtro passa-faixa
- Desenvolvido em C



PD-Externals

Objetos externos programados em C ou C++ que podem ser integrados ao PureData


Sinal
Original
Adição de sinais
Sinal
Modulado
Filtro passa-faixa
Oscilador senoidal


adc~
netreceive 3004
route 0 1 2 3
s onoff
wahwah~
r onoff
*~
dac~

Aplicativo Android

Android Studio

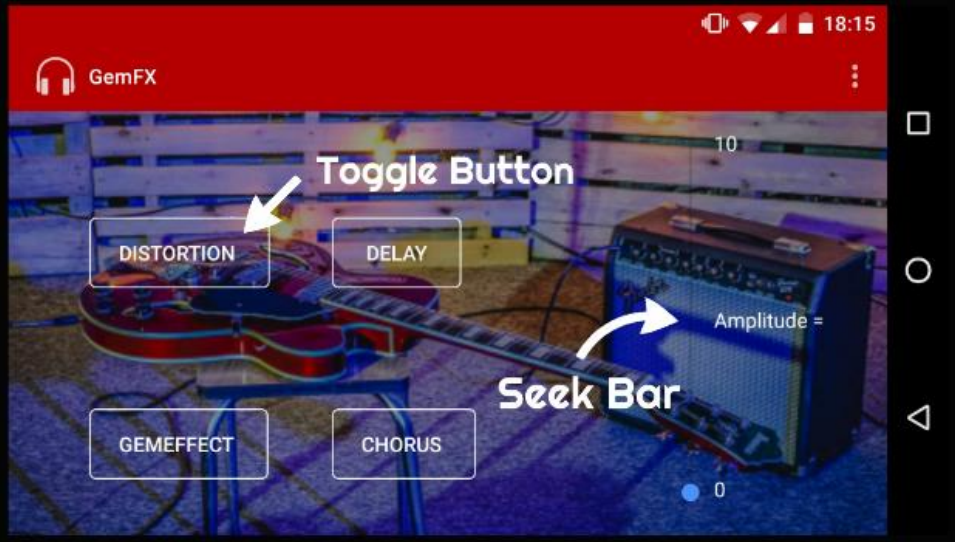
- Gratuido
- Linguagem Java voltada pra Android (SDK)

Aplicativo GemFX

Cada tela de um aplicativo Android possui o nome de Activity. O aplicativo desenvolvido possui 3 activities:

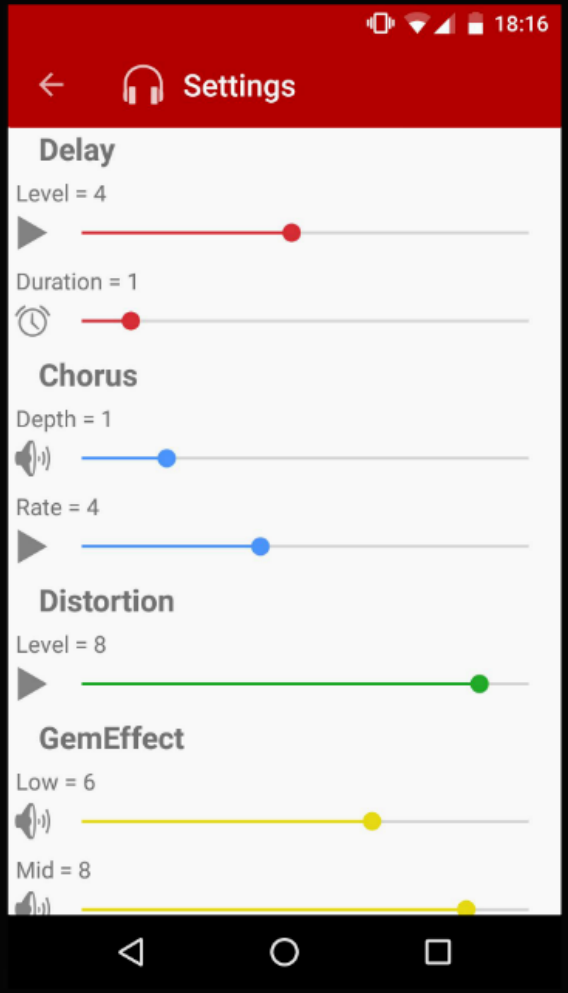
- main_activity
- settings_activity
- toobal

Main Activity

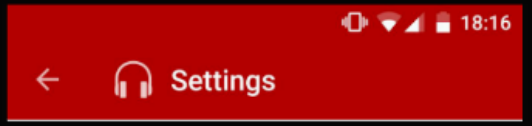


Toggle Button States

Settings Activity

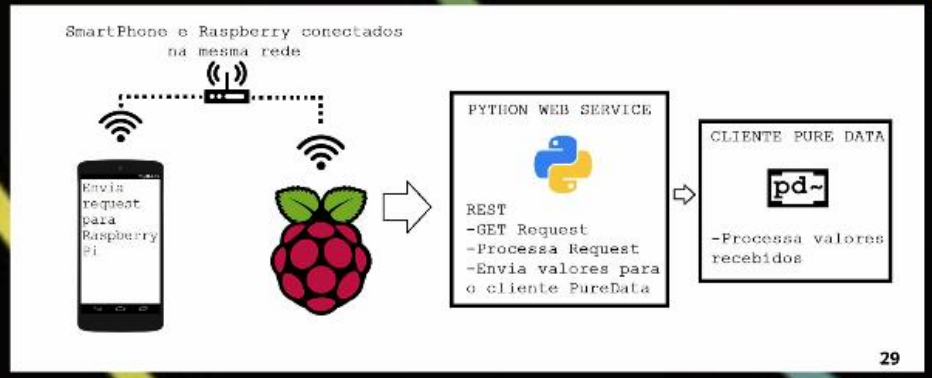


Toolbar

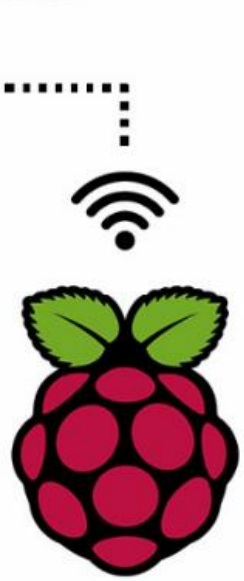


A activity toolbar é diferente das anteriores. Ela é uma activity integrada as outras, fazendo papel de menu. É com essa activity que é possível "caminhar" entre a tela principal e a tela de configuração

Comunicação Wi-fi



SmartPhone e Raspberry conectados na mesma rede



PYTHON WEB SERVICE

REST
-GET Request
-Processa Request
-Envia valores para o cliente PureData

CLIENTE PURE DATA

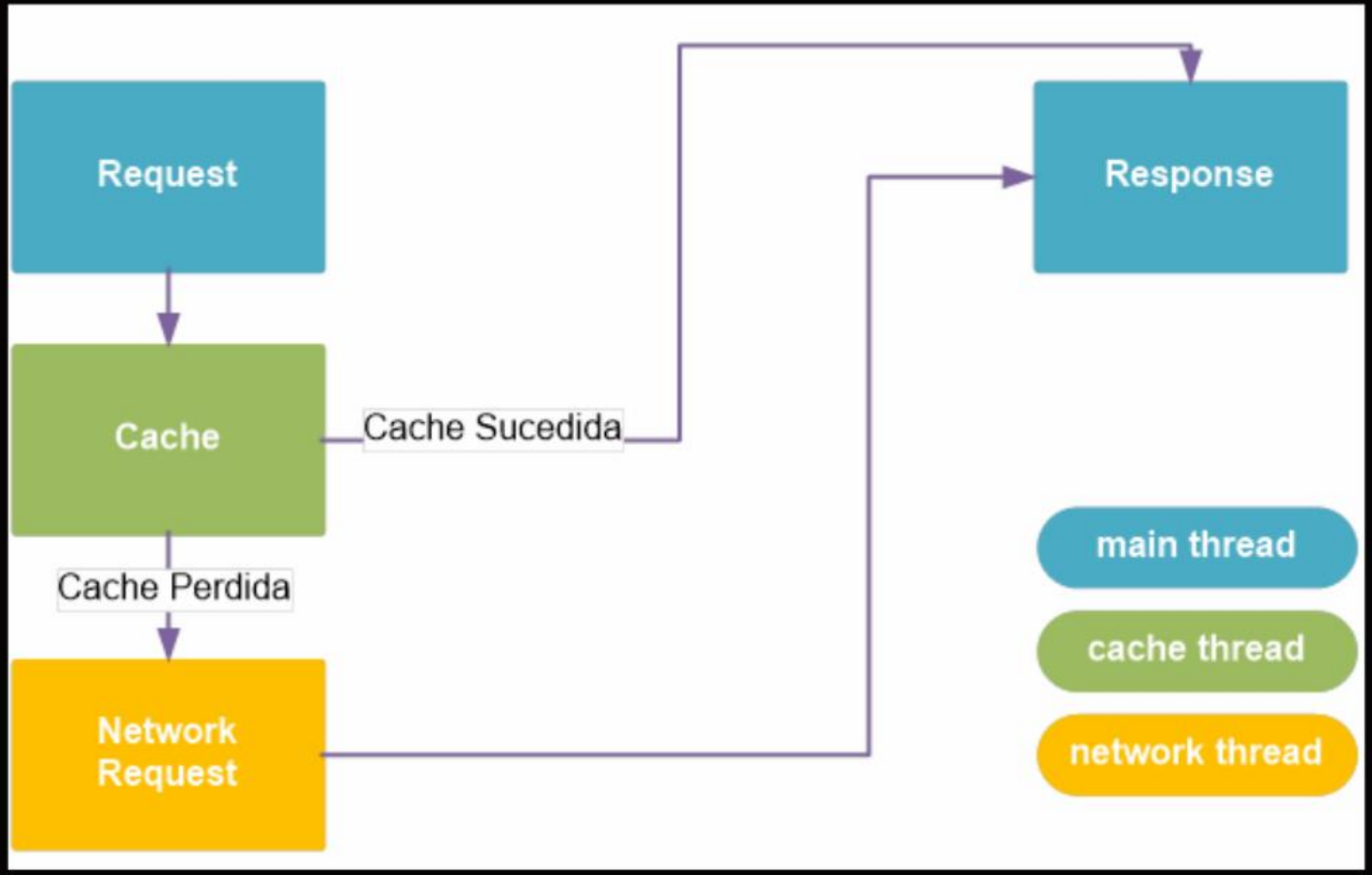
pd~

-Processa valores recebidos

Biblioteca Volley

Volley é uma biblioteca de HTTP (Hypertext Transfer Protocol) desenvolvida inteiramente pela Google. A Google desenvolveu essa biblioteca devido a deficiência das classes de networking do Android Studio.

Funcionamento:



Vantagens:

- Priorização de requests
- Programa automaticamente requests
- É capaz de suportar múltiplas conexões
- Automaticamente manda para a cache qualquer request
- É possível cancelar requests individualmente

No Android Studio:


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final RequestQueue queue = Volley.newRequestQueue(this);
```

Primeiramente é declarada uma fila de Requests

```
//envia o novo level
String url_level = host + "settings/distortion/level/" + Integer.toString(dil);
sendRequest(url_level, queue);
```

Dependendo do efeito selecionado, seu parâmetro é salvo numa url e é chamado o método para enviar o request

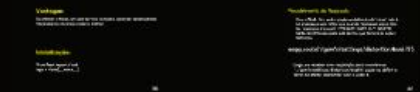
```
public void sendRequest(String url, RequestQueue queue)
{
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            (response) → {
                    //exibe a resposta no toast
                    Toast.makeText(getApplicationContext(), response, Toast.LENGTH_LONG).show();
            }, (error) → {
            //caso falhe
            //Toast.makeText(getApplicationContext(), "Falha de Comunicação! :(", Toast.LENGTH_LONG).show();
    });
    queue.add(stringRequest);
}
```

Em todos os requests feito no nosso aplicativo é utilizado o método GET, o qual espera-se uma resposta. Assim que a resposta chegar é exibido um Toast para dar feedback ao usuário. Após isso o request é adicionado a fila de requests

Assim que o request entra na fila (queue.add(stringRequest)), ele então é tratado como explicado anterirormente

Python Web Server

- micro-framework "Flask".



Vantagem

Ao utilizar o Flask, um web service completo pode ser desenvolvido inteiramente em único arquivo Python

Inicialização:

```
from flask import Flask
app = Flask(__name__)
```

Recebimento de Requests

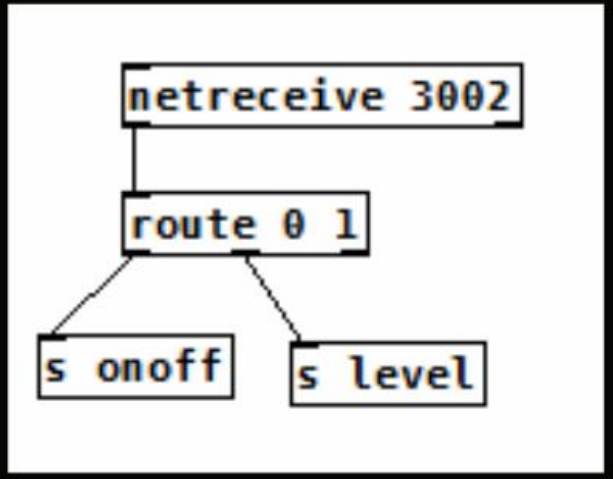
Com o Flask, fica muito simples a definição de 'rotas', isto é, os endereços web (URIs) que quando receberem algum tipo de requisição (request) HTTP (GET, POST, PUT, DELETE) serão identificados pelo web service que tomará as ações definidas.

```
@app.route('/gemfx/settings/distortion/level/5')
```

Logo, ao receber uma requisição para o endereço: '../gemfx/settings/distortion/level/5' pode-se definir o 'level' do efeito 'distortion' com o valor 5.

Pure Data Client

- netreceive
- route



Considerações Finais

Projeto GemFX OK!

- **Atividades futuras:**

Disponibilizar mais efeitos no GemFX

Utilizar aplicativo em um show

Raspberry Pi 2 funcionando como Access Point

40

Custos

USB Guitar Link Cable

US$8

Raspberry Pi 2 Model B - 1GB

US$ 42

Wi-fi Dongle

US$ 9

Dólar R$3,36

Guitarra

R$600

Amplificador

R$250

Cabo P10

R$30 (x2)

Custo total: R$ 1110

ões Finais

eitos no GemFX
um show
onando como

39

Vídeo de Demonstração

42

Pu

- Linguagem d
gráfica usad
processame
- Código aber
- Objetos

Efeitos

- Fuzz
- Delay
- Chorus
- GemEffect (WahWah)

Prezi

Perguntas?

GemFx

Geanine Inglat
Mariana Carrião
Mariana Cabral