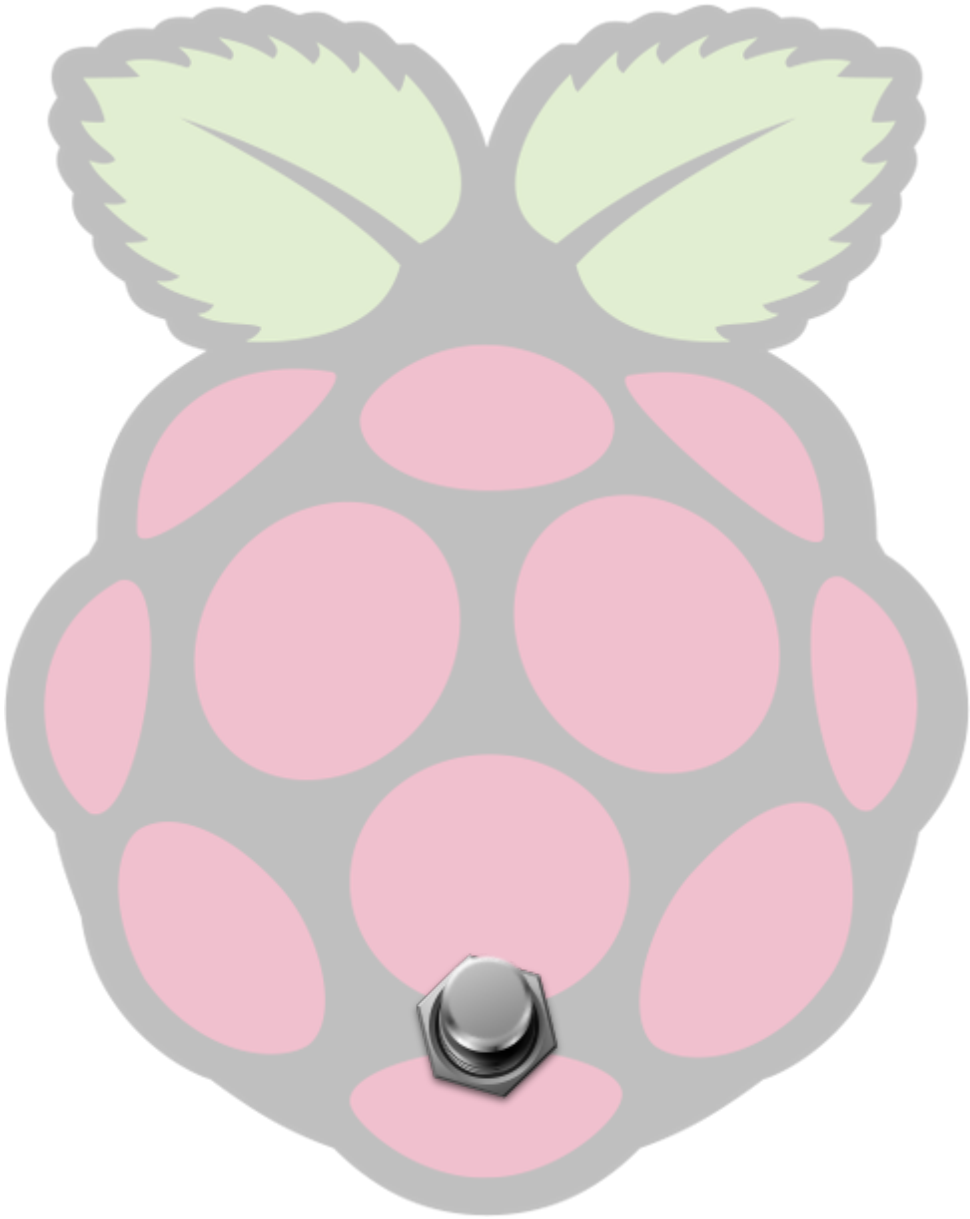

Pedal Pi - Documento de Requisitos

de Pedal Controller Projects

Versão 1.0.0
18/03/2016

Desenvolvido por Paulo Mateus Moura da Silva

Sumário

1. Prefácio

Como forma de padronizar o processo de desenvolvimento do Pedal Pi, este documento fora desenvolvido e está acessível para todos os leitores, sejam estes clientes (instrumentistas) e desenvolvedores (colaboradores, seja, engenheiros de sistema, engenheiros de teste, engenheiros de manutenção ou especialistas em marketing e divulgação).

Versão 1.0.0

18/03/2016 - Paulo Mateus Moura da Silva

Em sua versão inicial, o documento de requisitos concentra-se em aspectos básicos, visando o desenvolvimento contínuo incremental de forma sustentável. Em relação ao uso, a entrega de um produto viável mínimo que ofereça um fácil controle inicial em forma de *WebService* também é vislumbrada. Por fim, metas de segurança, praticidade e uso são definidas.

Futuras versões

Futuras versões deverão ser lançadas para aprofundar os requisitos de sessão de usuário, instalação de efeitos, automatização na atualização de sistema, suporte a periféricos.

Estrutura do documento

O documento é composto por seis sessões, cujo seus propósitos estão a seguir descritos:

1. **Prefácio**: Lista as versões desta documentação, enumerando os objetivos e motivos que levaram a criação de determinada versão. A estrutura do documento também é exposta como auxílio no entendimento deste documento;

2. **Introdução**: A necessidade da construção do sistema, uma visão superficial do minimundo em que o sistema operará, os serviços oferecidos pelo sistema e o impacto esperado no mercado musical com a introdução da solução desenvolvida;

3. **Glossário**: Termos técnicos - seja do minimundo, quando do sistema - descritos para auxiliar o leitor;

4. **Requisitos**: Requisitos - sejam de usuário, quanto de sistema - descritos inspirado no modelo *PWS - SISP Lista de Requisitos v0.2*;

5. **Arquitetura e modelos do sistema**: Uma visão geral de alto nível da arquitetura do sistema. O espaço está também reservado para a adição de modelos do sistema em versões futuras;

6. **Evolução do sistema**: Pressupostos fundamentais em que o sistema se baseia descritos para nortear alterações no sistema, seja por decorrência da evolução do software, quanto por manutenção.

2. Introdução

2.1 Ambientação

Dos protótipos de guitarra elétrica de Leo Fender ao presente momento, a busca por novos modos de expressão por meio da modificação sonora do instrumento sempre estiveram presentes na vida da guitarra.

Saturação por amplificação (extrema) do sinal, furos em falantes de amplificadores, válvulas fritando, equalizadores de frequência, moduladores de sinal, emuladores e simuladores de equipamentos *vintage* e instrumentos são algumas das experiências e recursos utilizados como parte da dinâmica de criação e reprodução musical.

Esforços na área de modulação e simulação de sinais foram introduzidos no mercado como forma de compactar equipamentos - como pedais analógicos e amplificadores, diminuir o investimento, facilitar o acesso a sons propiciados por equipamentos raros, ampliar as possibilidades de uso. Como resultado, processadores de multi-efeitos obteram parte do mercado de unidades de efeito.

Da estratégia mercadológica, a indústria de processadores multi-efeitos dispõe de produtos de forma cíclica. “O mais moderno e realista equipamento já construído”, “n vezes mais potente que o anterior” ou alguma variação é quase sempre utilizada no material de propaganda. Como por vezes a frequência de lançamentos é curta, consumidores por vezes questionam da veracidade das propagandas, como também reclamam da necessidade de comprar ciclicamente hardware fechados para obter - as vezes - pequenas melhorias.

O brasileiro Gianfranco Ceccolini, através do MOD Quadra e (em fase de testes com usuários finais) o MOD Duo, trouxe para o meio musical uma novidade ao embarcar computadores de propósito geral e utilizar neles um Sistema Operacional open-source: O poder da comunidade open-source. Seguindo o padrão para *plugins* de áudio *LV²*, os equipamentos da MOD Device possibilitam a criação e melhoria dos algoritmos utilizados pelos efeitos. Sua arquitetura fora projetada visando a expansão de hardware através de periféricos que podem ser desenvolvidos por entusiastas, programadores ou empresas. Deste modo, com a já obtenção do produto, um upgrade no som deixou de estar atrelado em um gasto em um hardware totalmente novo.

Entretanto, os equipamentos da MOD Device também possuem suas restrições, seja por hardware (custo do computador com placa de áudio integrada > $ 349.00), quanto por software. Uma análise comparativa está fora do escopo deste documento.

Por outro ambiente, Raspberry Pi Foudation é uma fundação que visa dispor sistemas computacionais de propósito geral portáteis, baratos e robustos para divulgação da inclusão digital, desenvolvimento do aprendizado e até a introdução de campos da Ciência da Computação para crianças. Seu hardware recém lancado, Raspberry Pi 3,

dispõe de confuguração relativamente próxima ao MOD Duo por um preço acessível ($35.00). Por rápida comparação, Pi somente não oferece uma interface de áudio aceitável.

O projeto Pedal Pi, especificado neste documento de requisitos, visa oferecer um processador de multi-efeitos como forma de solução de sistema embarcado acessível, integrável com interfaces de áudio compatíveis com o *OS Raspian* e controlável a partir de periféricos de baixo custo.

Seu desenvolvimento se dará a partir de um processo incremental orientado a planos. Entretanto, por se tratar de um projeto open-source, este deverá atender características de processos ágeis. O conjunto deverá possibilitar um desenvolvimento colaborativo descentralizado sustentável.

2.2 Descrição

Pedal Pi destina-se a prover uma solução para instrumentistas que desejam utilizar efeitos de áudio de qualidade em seus projetos artísticos com um investimento mínimo.

Sua organização lógica baseia-se em pedais efeitos: Uma cadeia de efeitos de pedais são ligados em série de forma lógica para obter um resultado sonoro desejável. A entrada é - em grande parte das vezes - o instrumento e a saída, equipamentos como mesas de som, fones de ouvido, sistemas amplificadores de áudio, interface de áudio.

Imagem 1 - Cadeia de pedais ligados.

Através da abstração de cadeias de pedais, o conceito de *patch* ou *set de efeitos* fora introduzido pelos processadores multi-efeitos: conjunto de efeitos conectados logicamente com seus parâmetros definidos. A persistência e troca nas apresentações trouxe maior versatilidade de uso aos equipamentos. *Bancos* referem-se a uma lista ordenada de *patchs* e são utilizados para fins organizacionais.

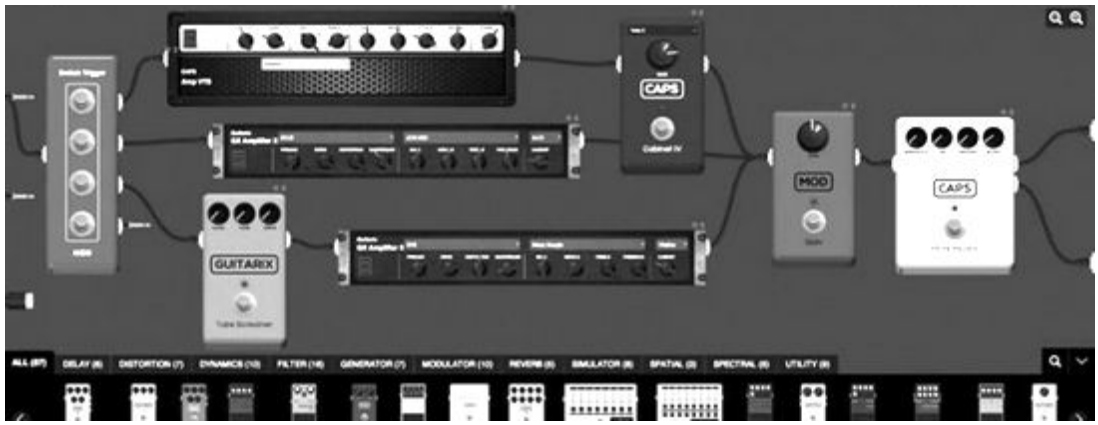

Imagem 2 - Exemplo de aplicação para configuração da MOD Devices.

Pedal Pi deve oferecer suporte a configuração de bancos, *patches* e seus efeitos através de programas consumidores de *WebService*, como também por periféricos controladores (Imagem 3.4). A conexão de entrada (para instrumentos - Imagem 3.1) e saída (para equipamentos de áudio - Imagem 3.5) deve ser possível através de interfaces de áudio suportadas pelo Sistema Operacional *Raspian* (Imagem 3.2). Os efeitos estarão disponíveis inicialmente pelo software do equipamento. Atualizações poderão ser adquiridas através de um ponto central de distribuição (site).

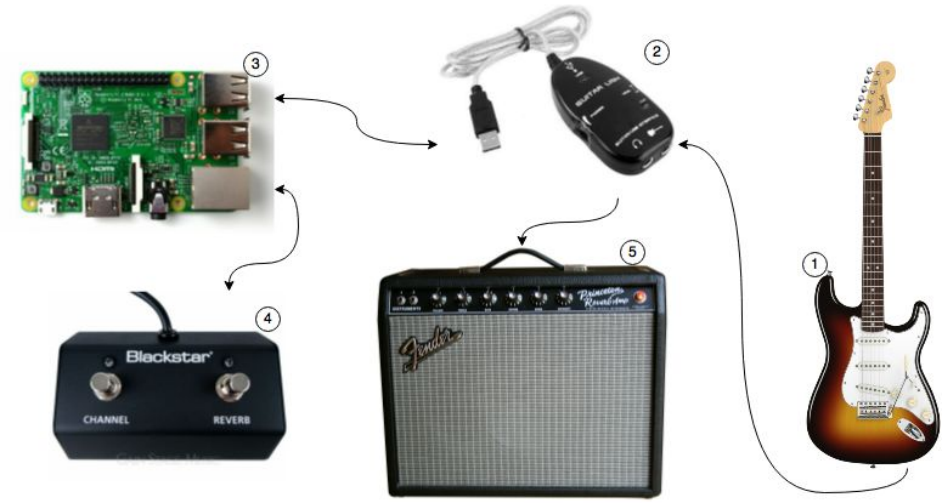

Imagem 3 - Organização lógica dos equipamentos. Ondas sonoras analógicas de um instrumento (1) são convertidos para um formato digital por interface de áudio (2). Há o processamento do áudio através do Raspberry Pi pelos algoritmos dos efeitos previamente configurados (3). Configurações rápidas poderão ser realizadas através de periféricos externos (4). O áudio processado é enviado para a placa de áudio que o converte em sinal digital. Por fim, o som é enviado para um equipamento de som (5).

2.3 Impacto

Ao introduzir uma ferramenta econômica que atenda uma boa parcela do mercado musical, espera-se: 1. que empresas que trabalham com processamento digital de áudio revisem seus planos estratégicos de negócio, de forma a beneficiar mais os usuários finais, e 2. que a comunidade open-source de áudio obtenha maior visibilidade e maior força ao oferecer mais um produto de qualidade.

3. Glossário

Backup: Cópia de segurança.

Megabytes (MB): Corresponde a 1.000.000 de *bytes*, onde um *byte* corresponde a um *bit*, unidade atômica de medida de espaço de armazenamento.

Vintage: Diz-se de quaisquer produtos antigos e de excelente qualidade. Equipamentos *Vintage* geralmente são caros e raros de se obter pela pouca produção e pelo tempo de fim de produção.

Interface de áudio: Equipamento que faz a ponte entre instrumentos e computador: Converte o sinal analógico do equipamento para digital, como também o inverso.

Linguagem de programação: Conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias.

LV²: Padrão aberto para plugins de áudio. Possui uma interface simples e estável, porém, vem acompanhado por extensões que agregam funcionalidade para suportar as necessidades de software de áudio cada vez mais poderoso.

Open source: Código-fonte de um software aberto.

Patch: Conjunto de efeitos (na forma de plugins) conectados logicamente entre sí com seus parâmetros definidos.

Pedais de Efeitos: Equipamento eletrônico utilizado para alterar o som natural de um instrumento elétrico. Geralmente ficam no chão próximo ao músico e são conectados diretamente ao seu instrumento. São comumente acionados pelo pé do instrumentista através de um interruptor. Exemplos de pedais são *Distortion*, *Overdrive*, *Wah-wah*, *Reverb*, *Delay*.

Processador multi-efeitos: Sistema embarcado em tempo real com finalidade de processar através de *DSPs* (*digital signal processors*) ou microcomputadores de propósito geral, sinais digitais obtidas de um instrumento musical através de conversões AD.

Raspian: Sistema Operacional distribuído pela Raspberry Pi Foundation com suporte oficial para as versões do Raspberry Pi.

Sistema Operacional (SO), Operational System (OS): Conjunto de programas cuja função é gerenciar os recursos do sistema, como: definir qual programa recebe atenção do processador, gerenciar memória, criar um sistema de arquivos.

WebService: Uma solução utilizada na integração de sistemas e na comunicação entre aplicações distintas. Com esta, é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

4. Requisitos

Requisitos funcionais são descritos em uma estrutura inspirada no *Modelo SISP: Lista de Requisitos v0.2*. A apresentação deste será conforme o seguinte padrão:

[CATEGORIZAÇÃO DO REQUISITO - SIGLA - NUMERO]	*Uma descrição sobre o requisito* **Prioridade**: *Classificação da prioridade, seja Essencial (alta) ou Desejável (média).* **Observações**: *(Somente se houver) Informações adicionais, como requisitos com que se relaciona e justificativa de sua existência.*

Onde:

- **[CATEGORIZAÇÃO DO REQUISITO - SIGLA - NUMERO]**: Nome do requisito, composto pelos campos:
    - **Categorização do Requisito**: RF (Requisito funcional) ou RNF (requisito não funcional);
    - **Sigla**: Sigla do tipo de requisito (o tipo de requisito será descrito na subseção no qual o requisito se encontra);
    - **Número**: Número do requisito.

Requisitos não funcionais são descritos de forma similar aos Requisitos funcionais exceto pelo campo **Prioridade**, que torna-se opcional, tendo em vista que **todos são Essenciais**, exceto os ditos de forma explícita.

Neste documento, não há uma clara diferenciação de *Requisitos de Usuário* e *Requisitos de Sistema*. Pela simplicidade da aplicação, foram tomados esforços para que os requisitos fossem descritos com o nível mais alto possível, de forma em que clientes e gerentes possam entender, mas com nível de detalhes o suficiente para que o documento também dê suporte as desenvolvedores e analistas.

Tendo ainda em vista a divisão da aplicação em partes como *WebService* (Seção 4.2.1) e *Aplicação controladora* (Seção 4.2.3), perceberá uma aparente duplicação de Requisitos Funcionais. Entretanto, lembre-se que cada requisito trás um valor de agregação na camada no qual este se enquadra e as camadas se relacionam entre si. O projeto como um todo será evidente nas definições de Caso de Uso.

4.1 Requisitos não funcionais

4.1.1 Requisitos não funcionais de Processo

[RNF - PROC - 01] A metodologia de desenvolvimento adotada será orientada a planos, com adição de características presentes em métodos ágeis.

[RNF - PROC - 02] Partes críticas do sistema deverão ser implementadas em C ou C++.

Observação:

Considere partes críticas a nível de processamento

[RNF - PROC - 03] WebServices podem ser implementados em Python, Node, Ruby ou uma outra linguagem de programação interpretada. Uso de Java deve ser fortemente evitado (ver **[RNF - DES - 01]**).

[RNF - PROC - 04] O processo de desenvolvimento deve favorecer: uma arquitetura consistente modelada em camadas, esforços contínuos de refatoração e desenvolvimento conforme as boas práticas de código limpo.

4.1.2 Requisitos não funcionais de Segurança

[RNF - SEG - 01] Usuário equipamento pode configurar para que o consumo do *WebService* para gerenciamento das configurações do dispositivo seja permitido somente de forma autenticada.

[RNF - SEG - 02] Atualizações do software devem ser realizadas através de uma funcionalidade do sistema somente para quando a rede de energia elétrica estiver estável para evitar corrompimentos.

Observação:

O Sistema Operacional Raspian vem um programa que analisa o consumo e a disponibilidade de energia pelo Raspberry.

[RNF - SEG - 03] *Backups* de configurações poderão ser realizadas a partir de clientes que consumam os *WebService* oferecidos pela aplicação.

4.1.3 Requisitos não funcionais de Desempenho

[RNF - DES - 01] A aplicação - que permite o controle do hardware e execução dos *plugins* de efeito - deve consumir no máximo 10% do poder de processamento e 256 MB de RAM (onde 128 MB é minimamente destinado ao Sistema Operacional).

[RNF - DES - 02] Alterações de configurações simples de efeitos realizadas através do WebService de controle de alterações deve consumir no máximo 20ms.

[RNF - DES - 03] Alterações de configurações simples efeitos realizadas através de Hardware por periféricos deverão ser realizadas no máximo em 10ms.

[RNF - DES - 04] Alterações de configurações complexas deverão ser realizadas no máximo em 100ms.

[RNF - DES - 05] O período de inicialização do sistema não deverá ultrapassar um minuto.

Observações

Alterações de configurações	
Simples	Complexas
Troca de estado de efeito (ativação/desativação)	Troca de uso de *patch* atual
Troca de valor de algum parâmetro de um efeito	Cópia de banco
	Cópia de *patch*

4.1.4 Requisitos não funcionais de Confiabilidade

[RNF - CON - 01] A atualização da aplicação não deve resultar em perda de configurações previamente salvas.

[RNF - CON - 02] Em caso de interrupção de energia, no uso do equipamento *ao vivo*, configurações de *patch* e *banco* não devem ser perdidas e/ou corrompidas. Entretanto, alterações de configurações simples de efeitos realizadas nos últimos 30 minutos são toleráveis.

[RNF - CON - 03] Interrupções por completo do funcionamento do sistema são totalmente indesejáveis, de modo em que o limite deve ser de uma vez por hora nas versões iniciais.

[RNF - CON - 04] O sistema pode interromper o funcionamento de *plugins* com consumo de processamento instável (com taxa de variação de 50%) ou gulosos (com taxa de uso superior aos 50% do ofertado pelo hardware).

4.1.5 Requisitos não funcionais de Usabilidade

[RNF - USA - 01] Para usuários experientes, caso estes queiram utilizar o equipamento como um computador de propósito geral, a case que envolve o hardware não deve bloquear entradas *HDMI* e *USB*.

Observações

Requisitos não funcionais de Usabilidade relacionados ao controle do dispositivo serão introduzidas em documentos posteriores.

4.1.6 Requisitos não funcionais de Manutenibilidade

[RNF - MAN - 01] A manutenção deverá do sistema deverá ser simples, conforme os parâmetros estabelecidos em **[RNF - PROC - 04]**

[RNF - MAN - 02] Problemas de usuários deverão ser relatados a desenvolvedores através de um e-mail próprio para isto e o acompanhamento deverá ser realizado com o uso das *Issues* da ferramenta de compartilhamento de código *Github*.

4.1.7 Requisitos não funcionais de Documentação

[RNF - DOC - 01] A documentação do código (funções, métodos, parâmetros) deverá ser mínima e da arquitetura detalhada (ver **[RNF - MAN - 01]** e **[RNF - PROC - 04]**).

[RNF - DOC - 02] Manuais ilustrados para usuários com as funções do sistema deverão estar disponíveis através do site oficial do produto. Através destes, instrumentistas pouco familiarizados com equipamentos de áudio deverão se familiarizar com o sistema em um tempo mínimo de 6 horas.

4.1.8 Requisitos não funcionais de Restrições Econômicas

[RNF - ECO - 01] O custo final do equipamento a ser vendido - em sua forma mais simples - não deve passar de $60.00 ($35.00 do Raspberry Pi, $10.00 de placa de áudio genérica, $5.00 de case para Raspberry e $10.00 para periférico controlador de efeito).

[RNF - ECO - 02] Destinado a ser um software livre, o projeto não possui uma fonte de financiamento fixa. Deste modo, os investimentos (seja financeira, seja força de trabalho) devem ser aplicados de melhor forma possível.

4.2 Requisitos Funcionais

4.2.1 Requisitos funcionais - WebService

Os seguintes requisitos serão utilizados por uma aplicação cliente controladora. Deste modo, iterações com usuários finais são desnecessárias, bem como restrições referentes às ações de usuários. Para detalhes dos serviços que devem ser oferecidos por uma aplicação controladora, visite a seção *4.2.3 Requisitos funcionais - Aplicação controladora*.

[RF - WS - 01] Disponibilidade das informações de versão das aplicações referentes ao Pedal Pi.

Prioridade: Desejável

Observação: Este requisito será utilizado para verificações de atualizações com um servidor remoto por intermédio de um programa cliente.

[RF - WS - 02] Listagem de todos os efeitos instalados na plataforma com as possíveis configurações dos parâmetros.

Prioridade: Essencial

Observação: Para adicionar um efeito a algum patch (**[RF - WS - 08]**), o cliente deverá saber quais são os efeitos instalados.

[RF - WS - 03] Listagem dos bancos já configurados e persistidos.

Prioridade: Essencial

Observação: Para configurar um banco (**[RF - WS - 06]**), selecioná-lo para uso (**[RF - WS - 12]**) ou listar os *patches* deste

(**[RF - WS - 04]**), é necessário a aplicação cliente conheça quais são os bancos já persistidos.

[RF - WS - 04] Listagem dos *patches* já configurados e persistidos de um banco informado pela aplicação cliente. Para cada *patch*, suas informações vinculadas (listadas em **[RF - WS - 05]**) deverão ser também retornadas.

Prioridade: Essencial

Observação: Para configurar um *patch* (**[RF - WS - 07]**) ou selecioná-lo para uso (**[RF - WS - 13]**), é necessário a aplicação cliente conheça quais são os *patches* já persistidos.

[RF - WS - 05] Para determinado *patch*, listar 1. as informações dos efeitos selecionados com seus parâmetros, 2. as conexões entre efeitos e portas de entrada e saída e 3. as configurações de periféricos com suas ações vinculadas deverão ser também listadas.

Prioridade: Desejável (desde que **[RF - WS - 04]** já retorne estas informações)

Observação: **[RF - WS - 08]**, **[RF - WS - 09]**, **[RF - WS - 10]** e **[RF - WS - 11]** necessitam destas informações.

[RF - WS - 06] Configurar bancos, seja: adicionar novos, remover bancos já persistidos ou alterar suas informações.

Prioridade: Essencial

Observação: O limite máximo de bancos depende da memória livre disponível.

[RF - WS - 07] Configurar *patches*, seja: adicionar novos, remover *patches* já persistidos ou alterar suas informações.

Prioridade: Essencial

Observação: O limite máximo de patches depende da memória livre disponível.

[RF - WS - 08] Adicionar efeitos para o *patch* em uso. A adição de efeitos deve respeitar os limites de consumo de memória e de processamento, de forma em que a soma de todos os efeitos, juntamente com os

processos do sistema não ultrapasse 90% do processamento e 95% da memória principal.

Prioridade: Essencial

Observação: O limite máximo de bancos dependerá da quantidade de memória livre disponível para persistir.

[RF - WS - 09] Trocar estado de um efeito (*ligado para desligado* ou *desligado para ligado*) do *patch* em uso.

Prioridade: Essencial.

[RF - WS - 10] Alterar valores valores dos parâmetros de um efeito do *patch* em uso.

Prioridade: Essencial

[RF - WS - 11] Conexão e desconexão entre efeitos ou entradas e saídas.

Prioridade: Essencial

Observação: Criação de *loops* (como ligar a saída de um efeito na entrada do mesmo) deve ser evitada, de forma que esta operação deve ser rejeitada, caso a aplicação cliente erroneamente requisite.

[RF - WS - 12] Um dos bancos já persistidos sera carregado como *banco atual* (ou *banco em uso*). Deste modo, as configurações do primeiro *patch* salvo será carregado para uso (**[RF - WS - 13]**) e as outras estarão disponíveis para serem utilizadas ao vivo.

Prioridade: Essencial

[RF - WS - 13] Um dos *patchs* do banco em uso (ver **[RF - WS - 12]**) virará o *patch* atual (conforme parâmetro passado na aplicação cliente). Deste modo, os recursos utilizados pelo *patch* anteriormente utilizado deverão ser liberados e os recursos requisitados pelos efeitos do *patch*, com seus parâmetros e conexões, deverão ser devidamente alocados.

Prioridade: Essencial

4.2.2 Requisitos funcionais - Sistema Operacional

[RF - OS - 01] Atualização do Sistema Operacional via programas oferecidos pelo próprio Raspian.

Prioridade: Desejável

Observação: O próprio sistema operacional utilizado oferecerá por padrão estas configurações.

[RF - OS - 02] Instalação de efeitos não testados a partir do terminal ou outros métodos sem suporte pelo Pedal Pi devem ser permitidas.

Prioridade: Desejável

Observação: O usuário se responsabilizará por qualquer problema provindo de suas ações.

4.2.3 Requisitos funcionais - Aplicação controladora

Requisitos funcionais relacionados a aplicação controladora serão contemplados em versões posteriores deste documento.

4.2.4 Requisitos funcionais - Conexão de periféricos controladores

Requisitos funcionais relacionados aos periféricos serão contemplados em versões posteriores deste documento.

4.2.5 Requisitos funcionais - Interface de áudio externa

[RF - AU - 01] Interfaces de áudio compatíveis com RaspianOS devem ser automaticamente reconhecidas, de forma em que a primeira reconhecida será a utilizada pelo sistema conforme sua configuração padrão (**[RF - AU - 02]**).

Prioridade: Desejável

Observação: O uso de múltiplas interfaces de áudio simultaneamente ou escolha de alguma poderá entrar como um futuro requisito.

[RF - AU - 02] Arquitetura deve estar preparada para um conjunto definido de múltiplas interfaces de áudio, de modo em que a seleção de

configuração de interface deva ser automática (assim como ocorre **[RF - AU - 01]**).

Prioridade: Desejável

Observação: Possibilidade de configuração manual é desejável. Entretanto, isto poderá entrar como um futuro requisito.

5. Arquitetura e modelos do sistema

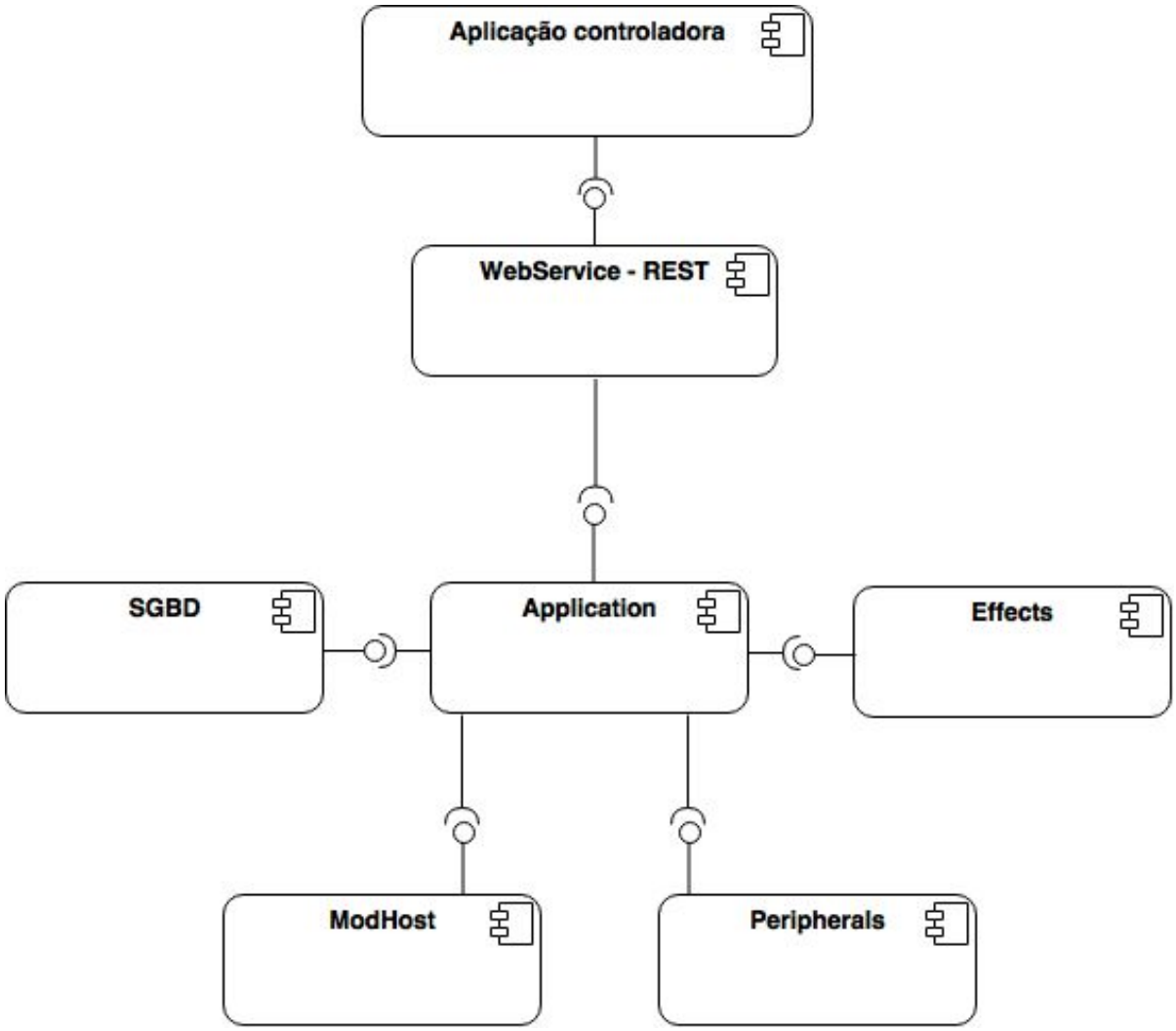

Imagem 4 - Diagrama de componentes para Pedal Pi

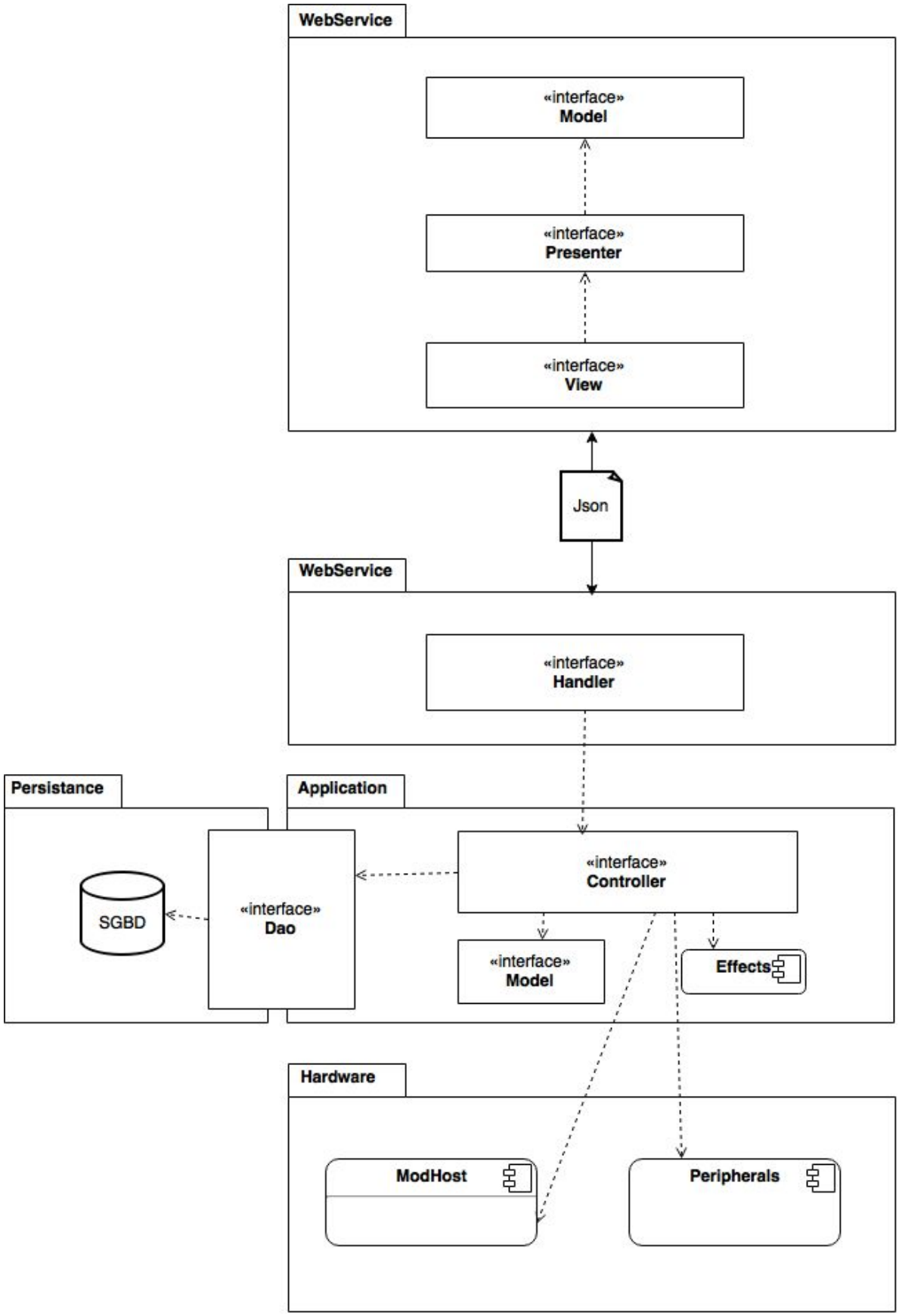

Imagem 5 - Diagrama de pacotes para Pedal Pi

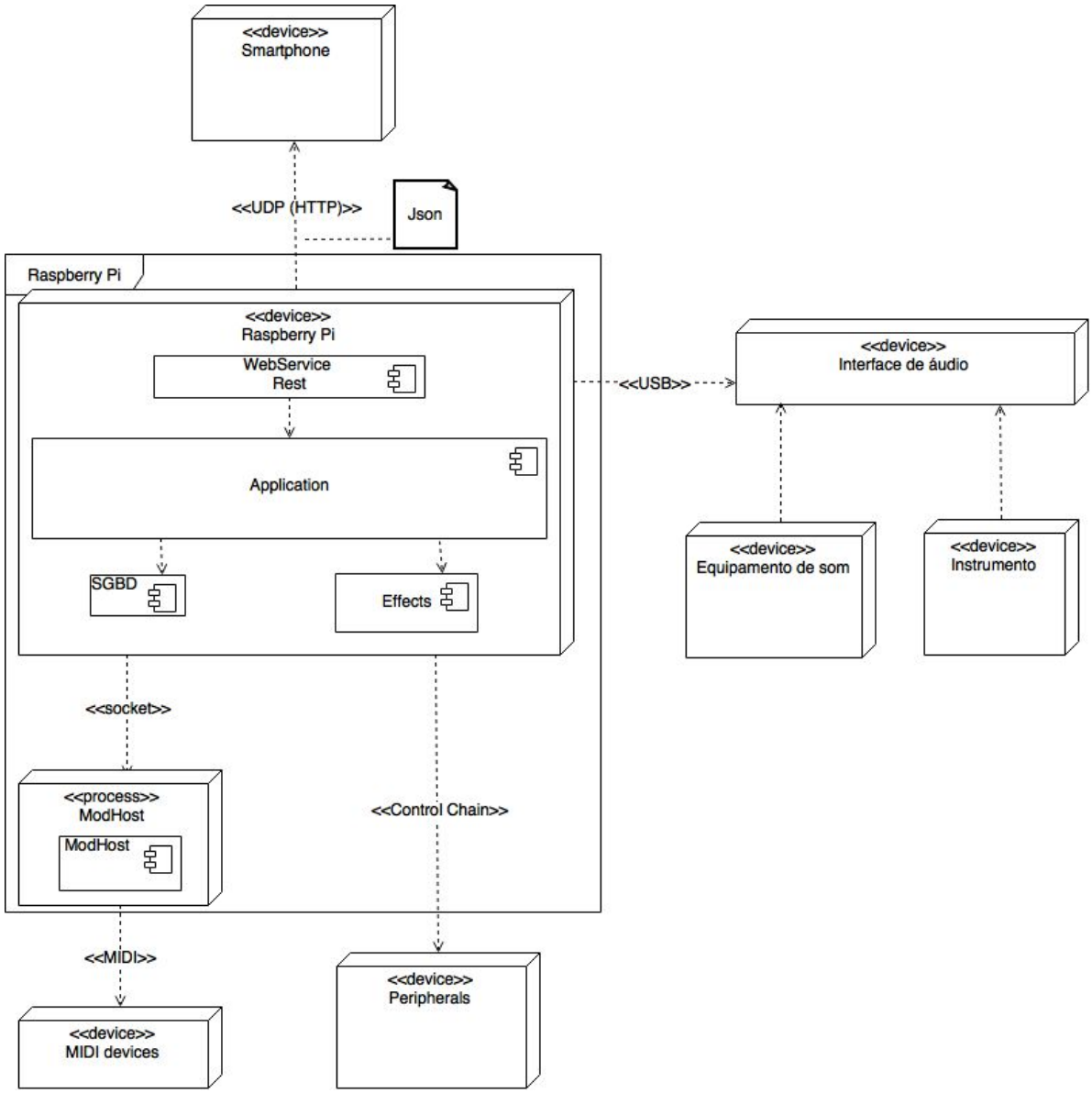

Imagem 6 - Diagrama de implementação para Pedal Pi

6. Evolução do sistema

Em quatro anos, a Raspberry Pi Foundation ofereceu uma evolução em computadores de propósitos gerais embarcados. Espera-se que ainda hajam maiores investimentos nesta área, de modo em que, futuramente, computadores portáteis com maior capacidade de processamento possam ser utilizados no propósito deste sistema.

Das interfaces de áudio, espera-se das fabricantes um maior suporte para Sistemas Operacionais Linux. Criações de novos protocolos e padrões para áudio pela comunidade também não estão descartados.

O sistema fora desenvolvido pensando na utilização de plugins de áudio LV². De fato, existem várias outras tecnologias, como *VST* (Virtual Studio Technology), *AU* (AudioUnits), *RTAS* (Real-Time Audio Suite), *LADSPA* (Linux Audio Developers Simple Plugin API).

Percebe-se que a variabilidade dos principais componentes utilizados pela arquitetura são fortes candidatos a sofrerem evoluções em um período de tempo relativamente pequeno. Desta forma, a arquitetura do sistema exalta separação de componentes por camadas bem definidas por interface simples e maleáveis.