

Relatório Técnico

Dalle Pad - O gadget que te transforma em um DJ

Leonardo W. Pereira – leonardowinterpereira@gmail.com

Lucas Z. Cordeiro – lukeszc@gmail.com

Luis Felipe. M. Ortiz – luisfmazzu@gmail.com

Junho de 2016

Resumo

O trabalho consiste no desenvolvimento de um equipamento musical eletrônico. Atualmente com a evolução da área musical é possível gerar sons artificiais e criar músicas a partir deste tipo de equipamento. Portanto, há a necessidade de desenvolver novas tecnologias para a produção musical, para contemplar o que já é produzido atualmente e o que surgirá no futuro. Os componentes utilizados nesse projeto são de baixo custo, consistindo em um invólucro de madeira, botões, potenciômetros, um microcontrolador, módulo *Bluetooth* e um shield MIDI. As principais funções do projeto são enviar mensagens MIDI, de maneira que qualquer *software* possa reconhecê-los, e desenvolver um aplicativo para Android que reproduz áudio ao receber essas mensagens. Como resultado, tem-se uma estrutura que apresenta todas as funcionalidades básicas para um músico editar e tocar músicas da maneira que desejar.

1 Introdução

A música é a arte de combinar sons de maneira agradável ao ouvido e a sensibilidade emocional utilizando elementos como melodia, harmonia e ritmo. Atualmente, não se conhece nenhuma civilização ou agrupamento que não possua manifestações musicais próprias. A criação, o desempenho, o significado e até mesmo a definição de música variam de acordo com a cultura e o contexto social, como composições fortemente organizadas e improvisadas [1].

Os instrumentos musicais até o século XIX baseavam-se em um mesmo princípio de produção sonora, todo som era proveniente da vibração de algum material elástico (as cordas do violão e do piano, por exemplo) que gerava ondas que se propagam pelo ar até atingirem o sistema auditivo do ouvinte. Entretanto, o surgimento de novas tecnologias baseadas na eletricidade e no uso de sinais eletromagnéticos abriu a possibilidade da geração de sons artificiais, sem a utilização de instrumentos mecânicos [2] [3].

Atualmente, pode-se contar com a Tecnologia MIDI. Desde seu lançamento no mercado no inicio da década de 1980, o protocolo MIDI tem tido um papel

de grande importância na indústria da música. MIDI permitiu, pela primeira vez, um meio de comunicação de informações musicais de um dispositivo para outro de uma forma que foi aceito e adotado por toda uma indústria [4]. Este protocolo continua sendo amplamente utilizado na área musical, e é de interesse deste projeto compreender a teoria e a prática referente a este tipo de aplicação, não somente sua utilização no produto final. Esta é a motivação para o desenvolvimento do controlador DALLE PAD.

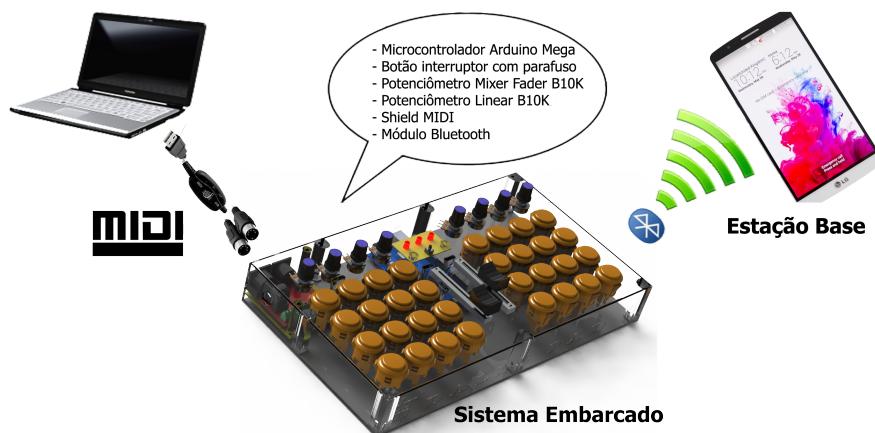


Figura 1: Visão Geral do projeto

O trabalho consiste em desenvolver um controlador MIDI que funciona transmitindo informações tanto via interface MIDI, para se comunicar com qualquer *software* que possua esta comunicação, quanto via *bluetooth*, para se comunicar com um aplicativo para o sistema operacional Android desenvolvido pela equipe. Como pode ser visto na Figura 1, esta comunicação é feita a partir de um sistema embarcado Arduino Mega, que recebe informações vindas de botões e potenciômetros e as converte em mensagens seriais a serem transmitidas pelos seus pinos de Tx/Rx. No *software* para Android, sons pré definidos disponíveis no aplicativo podem ser relacionados com cada botão, que ao ser pressionado, envia esta informação para o celular que reproduz o áudio. Na comunicação via interface MIDI, cada botão pressionado envia uma mensagem informando qual botão está se comunicando, e os potenciômetros enviam seu valor para as entradas analógicas do Arduino. Com isso, estes *softwares* são capazes de mapear quaisquer funções disponíveis ao DallePad, permitindo que ele funcione como instrumento para remixagens, criação de conteúdo musical novo, efeitos sonoros e performance ao vivo.

No desenvolvimento do projeto, alguns requisitos, que servem de guia para todas as modificações e aperfeiçoamentos, foram levantados. Tais requisitos estão definidos nas Tabelas 1 e 2.

Requisitos Funcionais
O <i>software</i> deverá permitir ao usuário final controlar dados MIDI
O <i>software</i> deverá permitir que o usuário altere as configurações do Dalle Pad
O <i>software</i> deverá realizar funções básicas, como trocar efeitos e tocá-los
O <i>hardware</i> deverá possibilitar uma fácil interação do usuário com o <i>software</i> através de botões e potenciômetros
O <i>hardware</i> deverá realizar uma ação ao ser pressionado um botão ou modificado um potênciometro

Tabela 1: Requisitos Funcionais do projeto

Requisitos Não-Funcionais
O projeto deverá possuir um relatório técnico
O <i>hardware</i> deverá possuir PCB's para facilitar o interfaceamento e evitar o acúmulo de fios
O aplicativo deverá possuir uma Interface gráfica funcional (em um estado inicial deve ser necessário ao menos a edição de som para cada botão, efeitos e volume, bem como uma interface simples de aprendizado)
O projeto deverá apresentar conexão entre ambas as partes através de USB, MIDI e <i>Bluetooth</i>
O invólucro deverá ser de plástico (para que possa ser impresso em uma impressora 3D) ou de um material que possa ser desenvolvido pela equipe
O <i>software</i> deverá ser desenvolvido na plataforma Android

Tabela 2: Requisitos Não-Funcionais do projeto

Ao longo do desenvolvimento do projeto, o problema que se procura resolver é: Existe como confeccionar um controlador MIDI que utilize os requisitos citados e que possua custo acessível?

2 Fundamentação Teórica

Robert Guérin, autor do livro *MIDI Power!* define MIDI como: interface que permite que músicos, engenheiros de som e iluminação, entusiastas da computação, ou qualquer outra pessoa que utilize computadores e instrumentos musicais eletrônicos para criar, ouvir e aprender sobre música, oferecendo uma linguagem comum entre dispositivos e *software* compatíveis. [4]

Com esta definição, pode-se perceber que MIDI é, por si só, incapaz de reproduzir som. Seus dados podem ser gravados em um programa de *software* (conhecido como sequenciador) ou dispositivo de *hardware*, onde estes podem ser editados e transmitidos para instrumentos eletrônicos ou outros dispositivos para criar a música ou controlar qualquer parâmetro desejado. Além disso, MIDI pode ser separado em três entidades diferentes: a) A linguagem utilizada, também conhecida como protocolo de comunicação; b) a interface de *hardware*

utilizada para transmitir e receber essas informações (tais como conectores e cabos); c) os formatos de distribuição, tal como SMF (*Standard MIDI Files*).

Fundamentalmente, MIDI é uma linguagem de descrição de música em forma binária, em que cada palavra binária descreve um evento de uma música [5]. Cada instrumento musical realiza um som, o qual está totalmente sobre controle do músico. A mensagem MIDI em si não contém o real som tocado pelo instrumento, mas sim a ação de tocar uma determinada nota em um determinado momento e durante um determinado período [6] [7].

O protocolo de comunicação MIDI é transmitido de forma serial ao invés de paralela. Em uma transmissão paralela, como o nome sugere, as informações são transmitidas simultaneamente. A quantidade de dados que podem ser transmitidos simultaneamente depende na capacidade física do fio e da velocidade com que os dispositivos podem enviar suas informações. Na Figura 2, podemos perceber que um *byte* (oito *bits*) é transmitido simultaneamente utilizando uma transmissão paralela. A transmissão serial, por outro lado, consegue enviar apenas um *bit* após o outro.

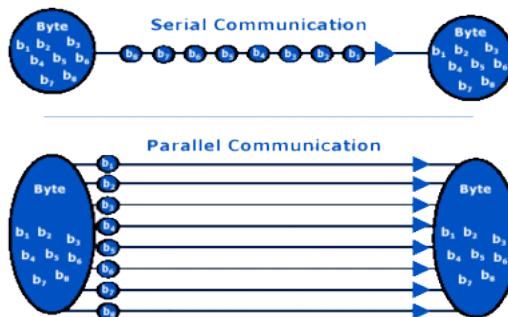


Figura 2: Comparação entre transmissão serial e paralela

MIDI envia informação a uma taxa de 31250 bps (*bits* por segundo). Essa velocidade é chamada de taxa de transmissão. Uma vez que o protocolo utiliza transmissão serial, o mesmo envia apenas um *bit* por vez. Cada *byte* em uma mensagem MIDI contém 10 *bits* de dados (8 *bits* para as informações e 2 *bits* para correção de erro). Isso significa que MIDI envia cerca de 3125 *bytes* de dados cada segundo.

Quando comparamos esse valor com a taxa de transmissão de 176400 *bytes* necessária para transmitir áudio digital (reprodução e gravação) em formato de CD de áudio, MIDI pode parecer incrivelmente devagar. Entretanto, neste último não é necessário enviar tanta informação quanto o áudio digital, sendo capaz de, teoricamente, transmitir até 500 mensagens MIDI por segundo.

MIDI tem a vantagem de ser compacto, uma vez que o som tocado por determinado instrumento não é gravado, mas sim apenas as informações de cada evento. Devido a este fato, arquivos MIDI apresentam um tamanho cerca de 300 vezes menor do que um arquivo de música atual (MP3). Entretanto, esse

valor pode ser muito maior se compararmos com formatos descompactados de músicas.

O fato de MIDI não gravar som, entretanto, pode também ser uma desvantagem. Quando o som de um determinado instrumento é gravado, o usuário tem total controle sobre o resultado final, enquanto com MIDI, este mesmo resultado depende do dispositivo ou *software* que será utilizado para reproduzir este mesmo som [8].

Quando se trabalha com música, muitas vezes deseja-se que diversos instrumentos participem da mesma composição. Com MIDI, pode-se incluir instruções para cada um deles através dos chamados **canais MIDI**. Um canal MIDI pode ser visto como um canal de televisão (veja a Figura 3). Um determinado instrumento conecta-se apenas a um único canal, assim como um usuário o faz em sua televisão. Isso não significa que os outros canais não estão disponíveis, mas sim que o conteúdo disponível (no nosso caso, a mensagem MIDI) foi filtrado para processar apenas o relevante para aquele canal.

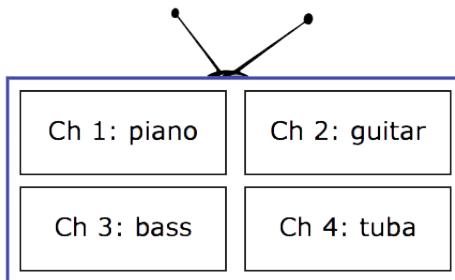


Figura 3: Exemplo de canais MIDI [6]

Cada porta MIDI suporta até 16 canais simultâneos. Em outras palavras, um usuário pode escolher um determinado canal para transmitir as mensagens MIDI ou possuir até 16 instrumentos conectados simultaneamente, cada um realizando diferentes eventos.

Para transmitir informações através de dispositivos, é utilizado uma **mensagem MIDI**. O conteúdo de uma mensagem MIDI apresenta um componente chamado de *status byte*. Este componente é acompanhado por um valor que é definido por um ou mais *bytes* de dados, como pode ser visto na Figura 4.

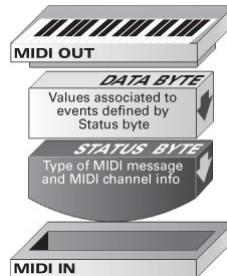


Figura 4: Byte de estado e de dados [4]

- A porção referente ao *byte* de estado serve para identificar o tipo de informação que está sendo enviada. Ela informa ao dispositivo receptor a qual canal MIDI determinado evento pertence e o que este evento é.
- A porção referente ao *byte* de dados informa ao dispositivo receptor o valor que está associado ao evento determinado na outra porção da mensagem MIDI.

Os valores utilizados pelo *byte* de estado variam entre 128 e 255 (1000 0000 a 1111 1111 em números binários), enquanto para o *byte* de dados estes variam entre 0 a 127 (0000 0000 a 0111 1111 em números binários). É fácil perceber que, para um dispositivo MIDI reconhecer qual o tipo de mensagem que se está sendo transmitida, basta verificar o MSB (Most Significant Bit) (*Bit* mais significativo, do inglês). Se este valor for 1, trata-se de um *byte* de estado e, caso contrário, de um *byte* de dados.

Digamos que um dispositivo MIDI receba três valores: 175, 43 e 125. Ele saberá que 175 é um *byte* de estado, enquanto os outros dois valores são os *bytes* de dados que estão acompanhando o *byte* de estado. Caso este dispositivo receba outros valores, digamos: 200, 220 e 90. Ele saberá que os dois primeiros valores são *bytes* de estados, enquanto o último é um *byte* de dados que acompanha apenas o último *byte* de estado, não ambos.

3 Estrutura

Este projeto procura considerar diversos quesitos, tais como resistência, momento, peso e tamanho do produto, além de manter os custos de produção o mais baixo possível. Para que isso fosse possível, decidiu-se projetar todos os componentes em uma ferramenta específica, podendo assim ser consideradas todas as variáveis para desenvolver um projeto de qualidade.

Inicialmente, foram definidos os componentes terceirizados do projeto, para que então fosse possível dar início ao mesmo. Estes componentes são:

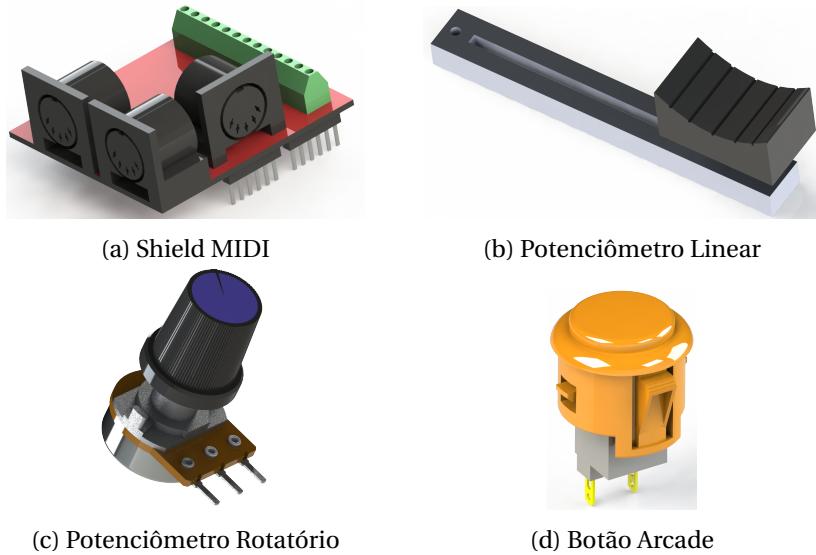


Figura 5: Componentes terceirizados do projeto

Possuindo todas as medidas dos componentes, pôde-se planejar e desenhar o invólucro, utilizando a ferramenta SOLID WORKS ©. O material escolhido foi a madeira, devido ao seu baixo custo e de fácil manuseio. O desenho final da estrutura pode ser vista na Figura 6.

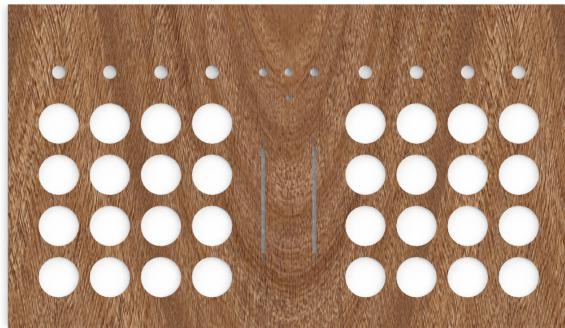


Figura 6: Invólucro de madeira - Vista Superior

Considerando a estética, este protótipo foi projetado para uma simetria bilateral. Desta forma, 16 botões, 4 potenciômetros rotativos e 2 potenciômetros lineares foi a melhor combinação levando em consideração o microcontrolador escolhido, a complexidade do projeto e o custo do mesmo.

Para a confecção do invólucro, foi contratado um serviço terceirizado. O resultado encontra-se na Figura 7. O produto fechado e pronto para uso encontra-

se na figura 8.



Figura 7: Invólucro de madeira confeccionado



Figura 8: Produto final

4 Sistema embarcado e estação base

A parte elétrica do projeto pode ser compreendida através da figura 9. O sistema embarcado gera sinais tanto para o shield MIDI quanto para o módulo bluetooth e estes se comunicam com suas respectivas plataformas (*software* com interface MIDI e sistema operacional Android, respectivamente). Os botões e potenciômetros enviam as informações a serem interpretadas como comandos e transmitidas aos dispositivos.

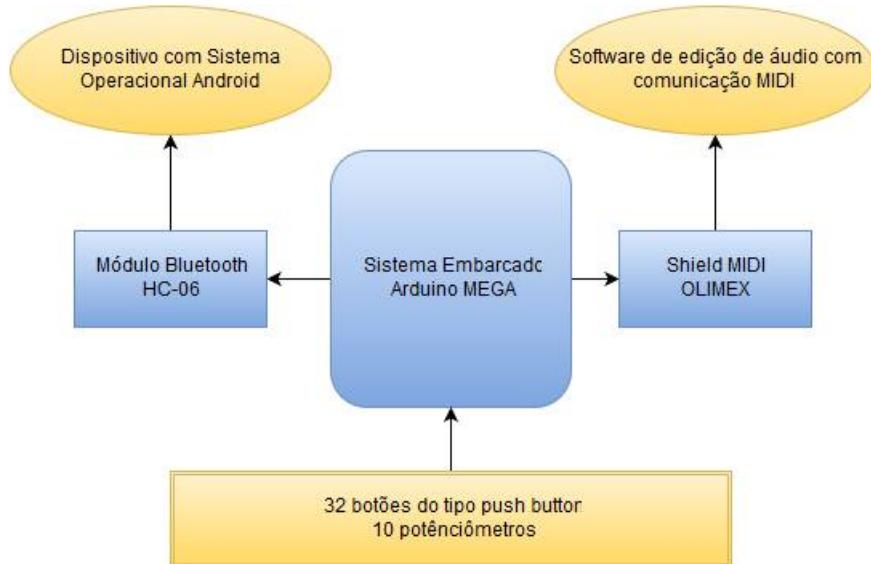


Figura 9: Diagrama de blocos com a representação dos componentes do projeto

O sistema embarcado escolhido foi o Arduino Mega, pois ele possui um número suficiente de pinos de entrada saída digital para ligarmos um botão por pino, suprimindo assim a necessidade de uma configuração mais complexa para a ligação deles como, por exemplo, matriz ou multiplexação[9].

A ligação entre os componentes externos (botões, potenciômetros, shield MIDI e módulo *Bluetooth*) e o sistema embarcado foi feita utilizando uma placa universal, com conectores MODU soldados nela e roteados com jumpers, desta maneira, não é preciso ligar diretamente estes componentes ao Arduino, o que tornou a montagem mais organizada e facilitou a correção de eventuais problemas. A ligação entre os componentes presos ao invólucro e a placa foi feita com o uso de cabos *flat* (Figura 10), confeccionados pela própria equipe. Para os cabos ligados aos botões, foram utilizados conectores do tipo *faston* (Figura 11), que se encaixavam adequadamente aos seus terminais. Para os potenciômetros, cabos *flat* regulares de 3 vias foram usados, mas feitos de tal maneira que pudessem ser encaixados nos potenciômetros regulares, com ajuda de uma barra de pinos MODU neles soldada.



Figura 10: Cabos flat utilizados para conectar os botões às placas



Figura 11: Conectores flats para serem conectados nos terminais dos botões

Originalmente o roteamento entre os botões e o sistema embarcado seria feito utilizando uma placa de circuito impresso, cujo *layout* foi feito com o *software* Eagle (Figura 12), porém surgiram desafios na passagem do *layout* do circuito impresso para a placa de cobre. Como o cronograma precisava ser seguido e problemas adicionais ainda podiam ocorrer, o projeto foi mudado para a utilização da placa universal. O layout confeccionado no Eagle foi utilizado como orientação para o local dos componentes a serem soldados na placa utilizada.

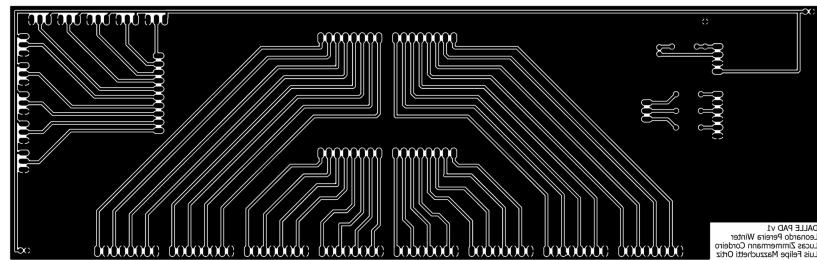


Figura 12: Layout da placa de circuito impressa que seria confeccionada

Outro problema enfrentado foi o fato de ter sido confeccionado um invólucro com uma altura baixa para o projeto e os cabos para ligação dos botões à placa serem compridos, o que dificultou o fechamento do produto em si. Este problema foi mitigado colando os cabos à carcaça interna do produto para reduzir o comprimento deles que fica solto dentro da caixa. O resultado das ligações (sem os cabos *flat*) pode ser visto na figura 13, um *layout* organizado e simples de conectar os cabos e corrigir eventuais erros.

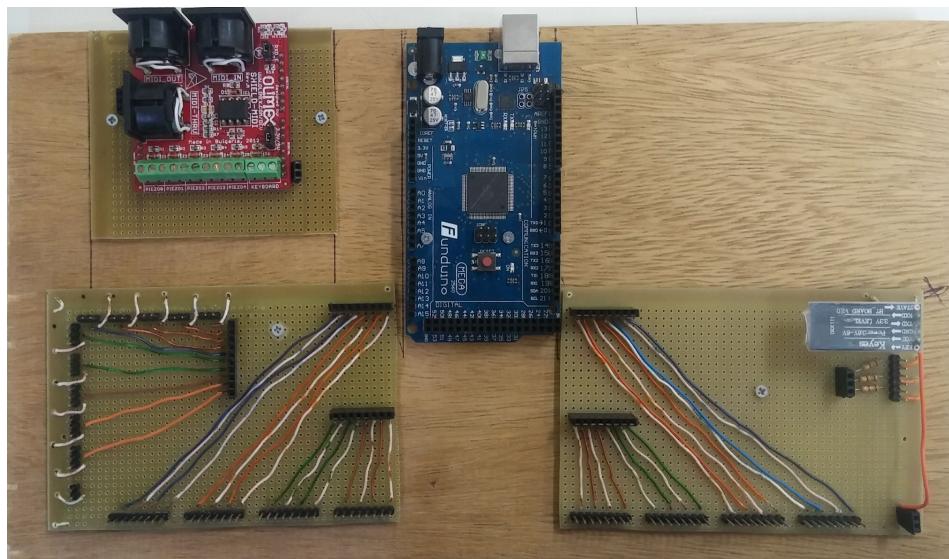


Figura 13: Componentes e placas universais com conectores roteados

O aplicativo foi desenvolvido para Android com a linguagem JAVA e consiste em apresentar funções básicas que um *software* manipulador de áudio possui. Como um *software* completo geraria grandes dificuldades quanto à manipulação de ondas e arquivos MIDI, decidiu-se em utilizar a estação base como um reprodutor de áudio, onde também seria possível trocar os efeitos utilizados. Portanto, o aplicativo somente recebe mensagens do arduino, sem a necessidade de respondê-las.

A primeira tela, presente na figura 14, é responsável pela habilitação do *Blu-*

uetooth no dispositivo. Após estabelecida a conexão, o aplicativo reconhece mensagens MIDI enviadas pelo Arduino, reproduzindo um som pré-definido de acordo com o botão pressionado. Também é possível trocar efeitos de cada botão ao tocar em um deles na tela do aplicativo, como pode ser visto na figura 15.

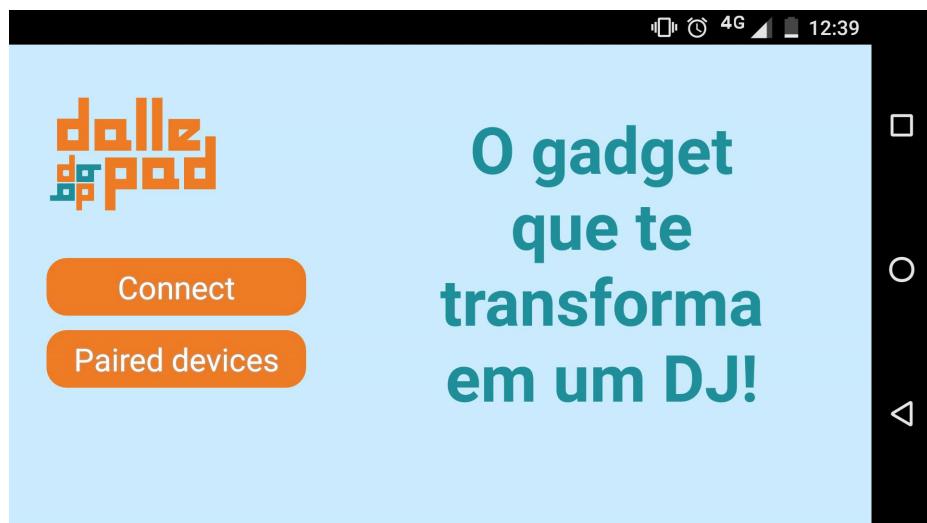


Figura 14: Tela inicial do aplicativo - *MainView*

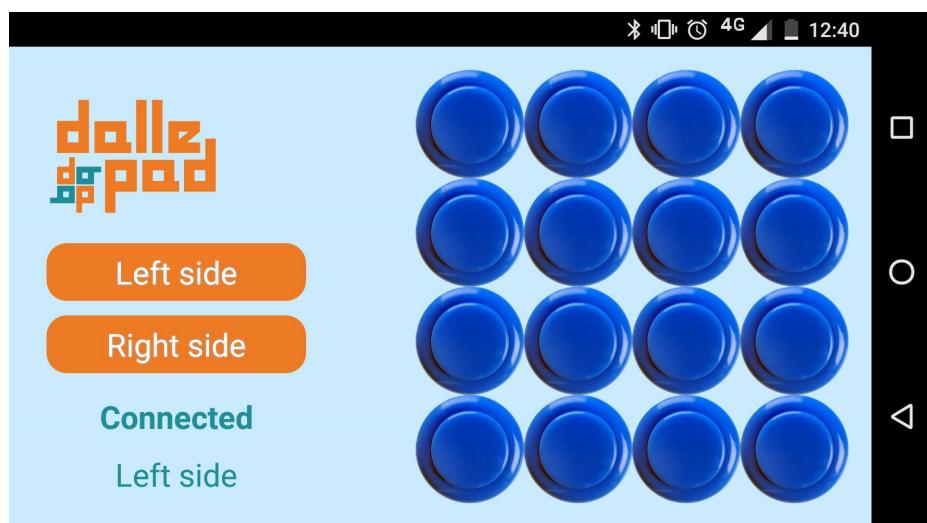


Figura 15: Aplicativo conectado com o arduino

5 Algoritmos

A figura 16 ilustra o diagrama de classes desenvolvido no começo do projeto, onde sua elaboração foi dada através do modelo MVC (*Model-view-controller*). Dessa maneira, foram separadas classes responsáveis pelos dados (*Model*), pelo controle dos dados (*controller*) e pelo design do aplicativo (*view*). As principais funções utilizadas estão contidas nas classes de controle, BtConnector e MidiController.

Na primeira tela (*MainView*), o reconhecimento entre o *software* e o sistema microcontrolado é estabelecida com ajuda da classe JAVA *BluetoothAdapter*. Essa classe auxilia na identificação dos dispositivos pareados com o Android que está sendo utilizado. Assim que um dispositivo pareado é selecionado, há a transição para a tela principal (*ConnectedView*). Logo no início da criação dessa classe, uma conexão TCP entre o aplicativo e a estação base é estabelecida através de um *BluetoothSocket*. Assim que a conexão é sucedida, o aplicativo cria uma *thread ConnectionListener* onde passa a esperar as mensagens do Microcontrolador. Cada mensagem recebida é transformada em *string* e dividida em *substrings* na classe *MidiTranslator* para identificar qual botão foi pressionado e se o botão foi pressionado ou solto naquele momento, assim como uma mensagem MIDI.

Para evitar *delay* entre o pressionamento de um botão e a execução de um som, foi utilizado a classe JAVA *Soundpool*. Ao mesmo tempo que o aplicativo tenta conectar-se com o arduino, sons previamente adicionados no aplicativo são carregados na memória para que a execução do som seja feita da maneira mais rápida possível[10]. As classes responsáveis pelos efeitos não foram utilizadas no projeto por terem sido consideradas parte do aprimoramento e não do projeto em si.

Para o arduino, as conexões MIDI e *bluetooth* são iniciadas e os botões e potenciômetros são indentificados na inicialização[11]. Ao entrar na função *loop*, o arduino passa a esperar pelo pressionamento dos botões e ajustes dos potenciômetros. Isso é realizado através da leitura dos estados desses componentes através das funções *digitalRead* e *analogRead*, assim como o uso de condicionais para identificar quanto tempo estes foram pressionados.[12]

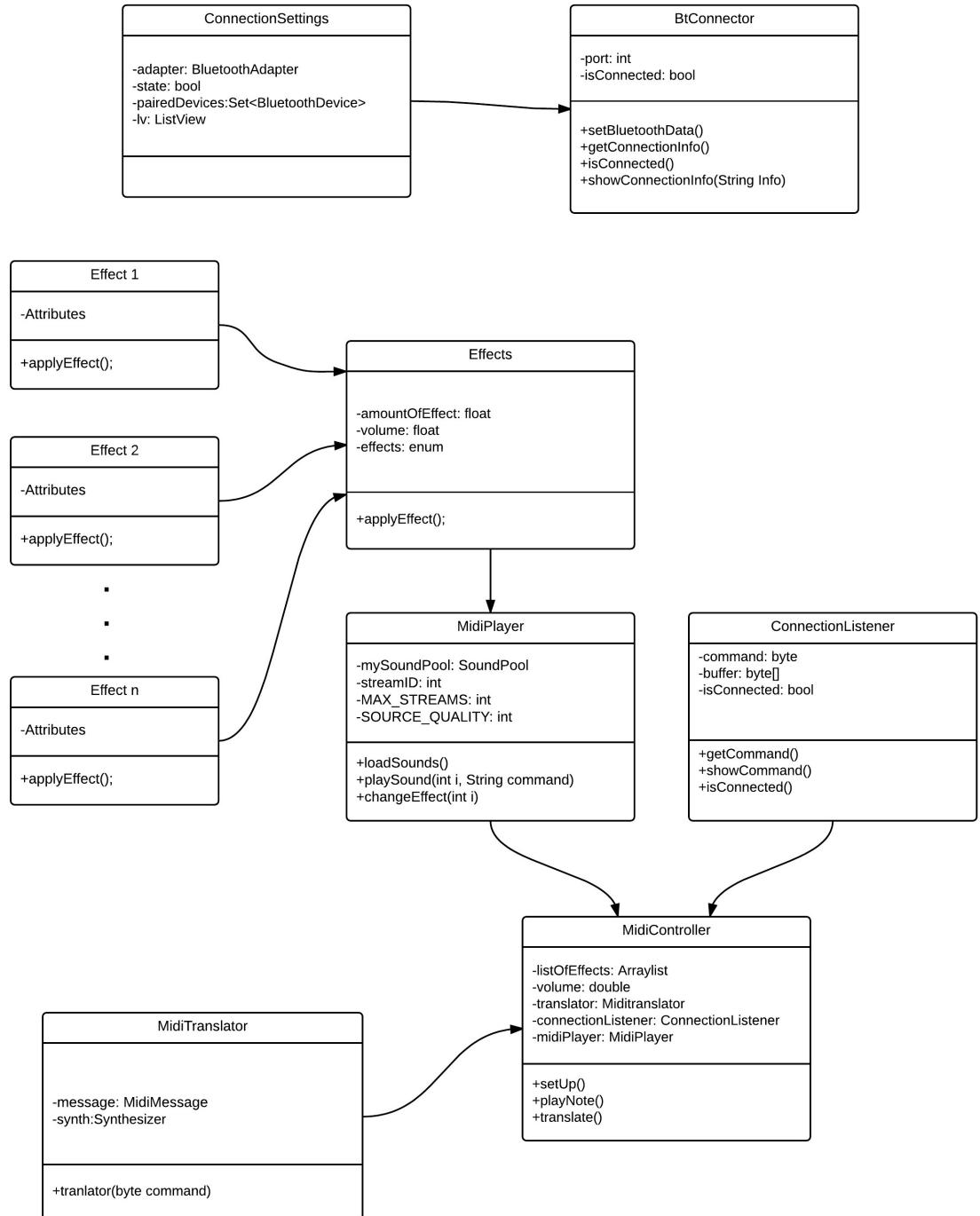


Figura 16: Diagrama de classes do aplicativo Android

Por fim, no momento que o microcontrolador identifica o pressionamento de um botão, as funções *MidiMessageToPC* e *MidiMessageToMobile* são chama-

das. Essas funções recebem um valor de *status byte*, um de dados contendo qual botão foi pressionado e outro contendo o *data byte*. O mesmo acontece com os potenciômetros, porém essas funções serão chamadas diversas vezes durante seu ajuste. As mensagens enviadas podem ser identificadas por qualquer *software* que utilize MIDI.

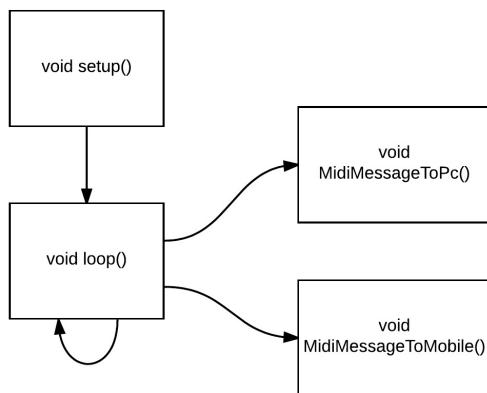


Figura 17: Funções do Arduino

6 Custos

Os custos do projeto realizado estão descritos na Tabela 3.

Material	Especificação	Valor / un	Qtde	Valor Total
Arduino	Arduino Uno R3 ATmega	13.90€	1	R\$ 57.50
Saída MIDI	OLIMEX Shield-MIDI	19.00€	1	R\$ 78.50
Bluetooth	<i>SunFounder Bluetooth HC-06 RS232</i>	8.99€	1	R\$ 37.20
Adaptador MIDI	Adaptador USB para Midi In-Out	7.70€	1	R\$ 34.00
<i>continua na próxima página</i>				

<i>continuação da página anterior</i>				
Material	Especificação	Valor / un	Qtde	Valor Total
Potenciômetro Linear	Potenciômetro Linear B10K	0.33€	8	R\$ 11.00
Potenciômetro Deslizante	Potenciômetro Deslizante B10K	1.17€	2	R\$ 9.75
Botão	Botão Interruptor ARCADE 24mm	R\$ 2.50	32	R\$ 80.00
Envoltório Potenciômetro Linear	<i>Potentiometer Drehknopf</i> 6mm	0.33€	8	R\$ 11.00
Envoltório Potenciômetro Deslizante	<i>Knob</i> para Potenciômetro 75mm	R\$ 1.50	2	R\$ 3.00
Madeira	Chapa de madeira compensada 10mm	R\$ 8.00	2	R\$ 16.00
Madeira	Chapa de madeira compensada 4mm	R\$ 2.00	4	R\$ 8.00
Mão-de-Obra	Terceirizada	R\$ 126.00	-	R\$ 126.00
Outros	Materiais para confecções dos cabos e das placas	-	-	R\$ 50,00
Total				R\$ 521.95

Tabela 3: Custos do projeto

Outros controladores MIDI podem ser encontrados no mercado na faixa de R\$400,00 a R\$500,00, porém estes possuem menos botões e funcionalidades que o projeto desenvolvido. Pode-se concluir que o custo do Dalle Pad é inferior ao de produtos similares.

7 Considerações finais

Ao final, tem-se uma estrutura funcional que apresenta todas as funcionalidades básicas que um músico precisa para editar e tocar músicas da maneira que desejar. O projeto apresenta compatibilidade com qualquer tipo de *software* que utilize comunicação MIDI e apresentou um custo mais acessível se comparado com outros produtos similares presentes no mercado. A estação base consiste em um aplicativo de fácil manuseio, sendo que, assim que o mesmo conecta-se ao Dalle Pad, o usuário poderá tocar e mudar os efeitos de cada botão. Para

trabalhos futuros, sugere-se aprimorar o *software* para que este conte cole as classes não utilizadas nesse projeto, como a classe *Effects* que teria a função de adicionar efeitos para os áudios tocados. Também seria interessante desenvolver uma função para carregar sons presentes na nuvem ou no próprio celular, além daqueles presentes no aplicativo.

Agradecimentos

Primeiramente gostaríamos de agradecer aos nossos amigos Cesar Figueiredo Lula e Jonas Ariel Berner, por nos ajudarem com ideias e auxilio técnico no desenvolvimento do projeto. Também agradecemos à Bollauf Marcenaria e Carpintaria pelo ótimo trabalho realizado com o invólucro e à UTFPR pela disponibilização dos laboratórios. Por fim, somos gratos pelos professores Gustavo Borba e Guilherme Schneider pelas diversas orientações no decorrer do projeto.

Referências

- [1] Rose Marie Santini. *Admirável Chip Novo: A música na era da internet*. Rio de Janeiro: E-Papers Serviços Editoriais, Pág. 27 a 29, 2005.
- [2] Artists House Music. A beginner's guide to sampling, 2015.
- [3] Michael Hewitt. *Music Theory for Computer Musicians*. Course Technology, 368, 2008.
- [4] Robert Guerin. *Midi Power!* Course Technology PTR, 368, 2009.
- [5] Luciano Alves. *Fazendo Música no Computador*. Editora Campus, 368, 2009.
- [6] John Gibson. Introduction to midi and computer music: The midi standard, 2013.
- [7] Jeffrey Hass. Indiana university - introduction to computer music, 2013.
- [8] Glen Ballou. *Handbook for Sound Engineers, 5th Edition*. Focal Press, 368, 2015.
- [9] Arduino. Arduino reference, 2014.
- [10] Oracle. Java sound programmer guide, 2015.
- [11] Instructables. Arduino and bluetooth hc-05 connecting easily, 2016.
- [12] Instructables. Arcade button midi controller, 2015.