Sistema Vestível para Portadores de Deficiência Auditiva

Sensory Expansion Vest for the Hearing Impaired

Caique R. Marques
Federal University of Santa Catarina
Florianópolis, Brazil
c.r.marques@grad.ufsc.br

Lucas R. Neis
Federal University of Santa Catarina
Florianópolis, Brazil
lucas.neis@grad.ufsc.br

Rafael L. Cancian
Federal University of Santa Catarina
Florianópolis, Brazil
cancian@lisha.ufsc.br

Roberto A. P. Martins
Federal University of Santa Catarina
Florianópolis, Brazil
robertophi@gmail.com

RESUMO

Este artigo foca na análise técnica de um Sistema Vestível para Portadores de Deficiência Auditiva. Usando motores de vibração presentes no colete, usamos comandos enviados de um smartphone para especificar um padrão de vibração para os motores (`e.g.` um sinal de bússola ou um áudio). Como é um sistema embarcado, ele é projetado para ser minimalista, portanto, ele tem que manter suporte às funcionalidades sem adição de hardware externo. Como este projeto usa uma FPGA, durante o desenvolvimento, nós focamos em fazer o software caber em uma memória *in-chip*, como resultado, o software tem um tamanho de 30 KB.

ABSTRACT

This paper is focused in the technical analysis of a Sensory Expansion Vest for the Hearing Impaired. Using vibration motors attached to a vest, we use commands sent from a smartphone to specify a pattern of vibration to the motors (`e.g.` a compass signal or audio). As it is a embedded system, it is thought to be minimalist so it can support its features without the need of any extra hardware. As this project uses a FPGA, during development, we focused in making the software fit in the in-chip memory. As a result, the software has a size of 30KB.

1. INTRODUÇÃO

O desenvolvimento tecnológico possibilitou diversas melhorias ao estilo de vida, no entanto, ainda há problemas de adoção de inovações provindas de tal desenvolvimento por pessoas com deficiências porque a tecnologia não fora criada para atendê-las. Assim, ferramentas que fazem o uso de áudios acabam tendo difícil adoção por deficientes auditivos ou ferramentas com imagens e vídeos podem ser mais inacessíveis a deficientes visuais. Este projeto, baseado na proposta de David Eagleman[2], visa quebrar as barreiras de comunicação usando o que qualquer pessoa é capaz de notar: o tato. Graças ao tato, podemos transformar sinais que são impossíveis de serem notados por alguém com alguma deficiência sensorial em um sinal facilmente entendível usando vibrações. Através de um colete de motores, um sinal é enviado a partir de um smartphone e o usuário é capaz de sentir o comando que ele enviou através de um padrão de vibração estabelecido (um certo tipo de vibração para recepção do áudio ou um certo tipo de vibração para a localização da coordenada norte).

Atualmente, o projeto já tem suporte a áudio, giroscópio e bússola. Nesta versão do software empregado, infelizmente, a adição de novos atributos é trabalhosa, mas uma nova versão do software está sendo trabalhada para que isso não seja uma tarefa árdua. Graças ao uso de uma FPGA, o hardware, por ora, é maleável o bastante para podermos mexer com as configurações e obtermos um melhor resultado. Em seu estado atual, o hardware conta com um controlador para a matriz de motores, um processador Nios II, uma UART e um timer.

Nas próximas seções iremos detalhar quais os componentes de hardware usados para a realização deste projeto e como a comunicação é feita entre software e hardware. Em seguida, cada componente de software é explicado. Por fim, apresentaremos as ideias futuras do projeto, que envolve a reimplementação do software, que já está em seus estados iniciais.

2. HARDWARE E PERIFÉRICOS

O hardware é, em essência, uma FPGA (*Altera DE2*) com diversos componentes.[1]

3. SOFTWARE

O software é, neste momento, simples e busca apenas o controle dos mecanismos da camada de hardware. No entanto, o código lida com instruções diferentes e, portanto,

[1]Esta seção é um esboço. A adesão de explicações por parte dos responsáveis pelo hardware é necessária.

SBSI 2016, May 17th-20th, 2016, Florianópolis, Santa Catarina, Brazil

podemos separar seus blocos de funcionamento.

Apesar de ingênuo, o software em seu estado atual tem dois níveis para controlar os componentes de hardware. A primeira camada, mais próxima ao hardware — doravante chamada de camada de mediação — e a segunda camada, acima da camada de mediação — doravante chamada de camada de controle e aplicação.

A camada de mediação é responsável pelas interações com o hardware e, portanto, quaisquer usos de componentes passam por esta seção. Junto aos drivers provindos pela Altera[3] para controle de componentes, a camada de mediação, se torna a nossa *Hardware Abstraction Layer* (*HAL*) ou Camada de Abstração de Hardware.

Já a camada de controle e aplicação é responsável pela interpretação de sinais provindos de comandos do smartphone e geração de padrões para vibração. Atualmente, há apenas três padrões de vibração estabelecidos: Áudio, Bússola e Giroscópio.

3.1 A Camada de Mediação

Na camada de mediação, se faz o controle dos seguintes componentes de Hardware: *Fast Fourier Transform* (FFT), *Motores* e *Wi-Fi*. Cada um desses componentes só pode ser acessado através dessa camada e pelos seus devidos controladores (ou mediadores). Nesta seção explicamos cada um dos mediadores.

3.1.1 FFT

Infelizmente o componente para a execução da Transformada de Fourier ainda não foi implementado. Como consequência, o mediador da FFT atua apenas como um mediador para um componente Dummy.

Sendo assim, recebendo um arranjo de n amostras, este mediador apenas retorna metade dos mesmos sem qualquer escolha ou processamento.

3.1.2 Motores

A matriz de motores no colete recebe e realiza comandos provindos de uma classe específica, que é o mediador dos motores. Os comandos, são de tamanho de 32 bits, têm quatro campos que são: comando (bits 31 a 24), linha (bits 23 a 16), coluna (bits 15 a 8) e valor (bits 7 a 0). A seguir, mais detalhes de cada campo:

- **Comando**: Especifica como deve ser a vibração pelos motores, se a vibração deve ser gradual até a um ponto específico ou se a vibração deve diminuir gradualmente até parar. Segue-se os cinco comandos criados:
  - Especificar potência (`0x0`): este comando especifica ou muda a potência de vibração de um dado motor alvo, especificado pelos campos ”linha”e ”coluna”do comando;
  - Especificar taxa de aumento de potência (`0x1`): a potência vai aumentando gradualmente até atingir o limite especificado no campo `Valor` (veja mais adiante);
  - Especificar taxa de decaimento (`0x2`): a potência vai decaindo gradualmente até atingir o limite especificado no campo `Valor` (veja mais adiante);
  - Cópia de uma linha a outra (`0x3`): este comando copia todos os valores dos motores de uma linha para a de baixo;
  - Cópia de uma coluna a outra (`0x4`): este comando copia todos os valores dos motores de uma coluna para a coluna à direita;
- **Linha**: Especifica qual a linha, da matriz de motores, deve ser escrita. Se o valor for 255, toda a linha é selecionada para escrita;
- **Coluna**: Especifica qual a coluna, da matriz de motores, deve ser escrita. Se o valor for 255, toda a linha é selecionada para escrita;
- **Valor**: Define qual a intensidade máxima de vibração dos motores, caso o valor do comando seja `0x02`, o limite especificado neste campo é o máximo que a vibração deve alcançar.

Ressalta-se aqui que o controle desses comandos é todo feito em hardware.

3.1.3 Wi-Fi

A comunicação entre o smartphone e o colete de motores é via rede Wi-Fi. Conforme especificado na seção anterior, o hardware suporta recebimento de pacotes via Wi-Fi que é interpretado pelos mediadores e respondido pela matriz de motores, assim, cada pacote recebido contém um caractere correspondente que identifica qual mediador deve ser ativado e quais os parâmetros que ele deve obedecer.

3.2 A Camada de Controle e Aplicação

Como mencionado na seção 3, esta camada lida com os padrões de vibração e a análise de dados recebidas pelo smartphone. Todos os dados são enviados via Wi-Fi, sendo essa camada a responsável por iniciar a espera por um novo sinal de dados, a escolha de qual é o processamento necessário e padrão de vibração a ser executado. Cada tipo de dados é identificado por uma letra enviada junto ao sinal de dados. Nesta seção explicamos como funcionam cada um dos padrões e os processamentos usados.

3.2.1 Áudio

Quando recebido do mediador do Wi-Fi, a string de dados de tamanho n recebida contém $n - 1$ amostras de áudio enviadas pelo smartphone e o seu primeiro caractere é `a` que identifica os dados como áudio. A primeira medida é enviar os dados para o mediador da FFT (aprofundado na subseção 3.1.1). O retorno do mediador é um arranjo de $\frac{n-1}{2}$ inteiros — supostamente, frequências em um intervalo de $[0, n - 1]$ — que deve ser enviado para o gerador de vibração de áudio (GVA).

O padrão de vibração do áudio é simples. Uma vez que o primeiro arranjo de frequências é enviado para o GVA, este divide o número $n - 1$ de frequências pelo número l de motores em uma linha. Este valor é o número de frequências que cada motor é responsável.

$$s = \frac{n-1}{l}$$

Portanto, cada motor da linha administra s frequências. Assim, o primeiro motor da linha (motor da coluna 0) lida com as frequências em $[0, s)$, já o segundo lida com as frequências em $[s, 2 \times s)$ e assim em diante.

Para definir a potência em que o motor deve vibrar, é feito então a média aritmética dos valores das s frequências em que o motor i é responsável.

Quando um arranjo chega ele é, inicialmente, aplicado na primeira linha de motores de cima para baixo (`i.e.` a linha 0). Os dados que estavam sendo tocados anteriormente são empurrados para a linha de baixo. Com isso, gera uma linha de áudios que foram tocados ao longo do tempo.

A taxa de aumento de potência é de 5 e a taxa de decaimento é de 1.

3.2.2 Bússola

Como sabemos, uma bússola nos mostra onde está o norte, portanto, o que é recebido via Wi-Fi do smartphone é a direção, em graus, em que o norte está. Além disso, também é enviado o caractere identificador `c` (de *compass*).

Diferente do áudio, nenhum pré-processamento é necessário. No entanto, cálculos ainda são necessários para escolher quais motores devem ser ativados. Existem duas diferentes versões para o padrão de vibração da bússola: uma usando cinco motores (quatro em uma mesma linha e mais um na linha acima desta) e um segundo que usa nove motores (oito em uma mesma linha e mais um na linha acima desta) chamadas de `compass4` e `compass8`, respectivamente. Ambas são explicadas a seguir:

- **compass4:** Esta versão é usada quando há no mínimo quatro e no máximo sete motores em uma linha de motores. No entanto, são necessários cinco motores. Um motor para indicar que o ângulo está entre 355° e 5° e quatro outros para indicar os quadrantes em que o ângulo pode está. O motor que indica que o usuário já olha para o norte (cujo chamamos de motor `N`) se localiza na linha de motores acima da última linha(número de motores por coluna -1) que é onde os demais motores estão.

  A imagem a seguir ilustra a ideia:

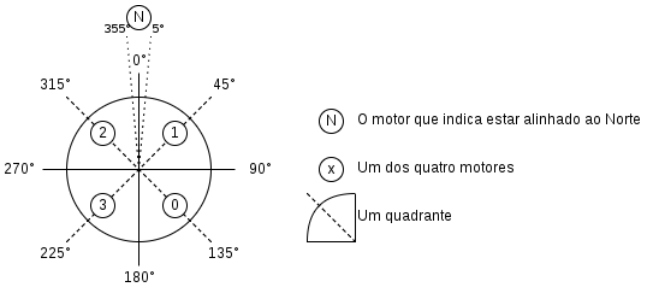

Figura 1: Representação dos quadrantes que os motores assumem na arquitetura `compass4`

  Então, conforme a figura 1, se o ângulo recebido estiver no intervalo (0°, 90°] o motor na coluna 1 deve ser ativado. Caso o ângulo estiver em [270°, 0°), o motor na coluna 2 deveria ser ativado.

  No entanto, se o motor estiver em [355°, 0°) o motor `N` será ativado juntamente ao motor na coluna 2. Similarmente, se o motor estiver em (0°, 5°] o motor `N` será ativado juntamente ao motor na coluna 1. O motor `N` só será ativado sozinho caso o ângulo seja exatamente 0°. Com isso, um usuário consegue saber se precisa virar mais para a esquerda ou mais para a direita para localizar o norte.

- **compass8:** Esta implementação é uma adaptação da versão `compass4` e deve ser usada caso hajam pelo menos 8 motores por linha. A diferença dessa versão é o melhor uso do maior número de motores para identificar um maior número de intervalo de ângulos. A imagem a seguir ilustra a ideia:

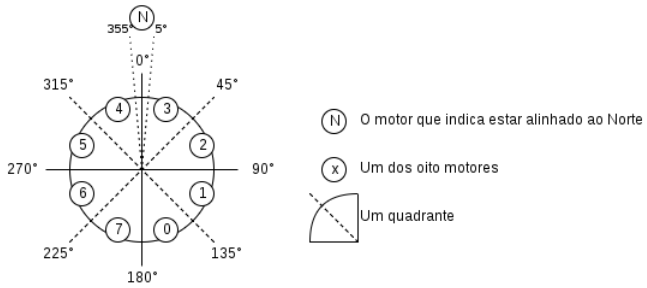

Figura 2: Representação dos setores que os motores assumem na arquitetura `compass8`

  O método usado é selecionado em tempo de compilação, através do uso de um `if-then-else` metaprogramado.

  A taxa de aumento de potência é de 5 e a taxa de decaimento é de 1. Além disso, a potência do motor é fixa, originalmente, em 25.

3.2.3 Giroscópio

Giroscópios nos fornecem a aceleração angular em três eixos. Para tornar isso em um padrão de vibração em uma matriz de duas dimensões, usamos a aceleração em dois desses eixos para selecionar um motor e a terceira se torna a potência em que o motor deve vibrar.

Mas para fazermos isso, temos alguns problemas. Se o número de motores em uma das dimensões for par, ele deverá ser tratado diferente do que seria se o número de motores fosse ímpar. Além disso, precisamos definir um intervalo de atuação.

Para resolver isso, separamos em dois métodos. O primeiro é usado para quando temos um conjunto de motores ímpar em certa dimensão. Este funciona por encontrar o motor localizado no centro desta dimensão. Com isso, definimos nosso intervalo de atuação para valores entre -100 e 100. Separamos, então, dois motores para representar valores fora dos limites e o restante dos motores dividem os 200 de intervalo entre si.

$$intvpmtr = \frac{200}{m-2}$$

Onde m é o número de motores na dimensão onde trabalhamos. Com isso, o intervalo por motor é definido e podemos então saber que o motor 0 da dimensão em questão lida com o intervalo $(-100, intvpmtr - 100]$ e assim por diante. Mais importante, podemos definir que o motor localizado no centro da dimensão lida com $[-\frac{intvpmtr}{2}, \frac{intvpmtr}{2}]$.

Logo, iniciamos testando se o valor está dentro do intervalo do motor central. Se sim, sabemos que o motor está na parte central da dimensão procurada. Caso contrário, testamos se o valor é negativo. Se sim, procuramos o motor que deve lidar com este valor entre os motores negativos, do contrário testamos entre os motores que lidam com os valores positivos.

No entanto, a versão com um número par de motores na dimensão não tem um motor no centro, então é feita uma busca simples para encontrar o motor que lida com o intervalo em que o valor está compreendido.

Sendo assim, basta tratar a dimensão remanescente. Esta é tratada de maneira simples. Pegamos o valor absoluto e então testamos para verificar se esta é menor que 240 (pois 255 é a maior potência que um motor sabe lidar), caso positivo, apenas somamos 10 a esse valor e temos, portanto, todos os valores necessários. Por outro lado, se for maior que 240, apenas reduzimos para 240 e somamos 10.

A taxa de aumento de potência é de 15 e a taxa de decaimento é de 5.

3.2.4 *Escalonamento de Padrões*

Um problema persistente neste projeto é o caso em que dois sinais são enviados a um mesmo motor. Uma das formas de contornar o problema é reservando fatias de tempo para cada sinal usar o motor, método chamado de multiplexação temporal. A forma implementada no projeto consiste num timestamp que é iniciado, e após meio segundo, o sinal é alternado para usar o motor.

3.2.5 *Adição de Novos Padrões*

A atual implementação torna difícil a adição de novos padrões. No entanto, a remodelagem do software pensa em tornar o processo fácil e sem complicações. No estado corrente, é necessário entender verdadeiramente o que acontece na camada de controle e aplicação.

4. REMODELAGEM DO SOFTWARE

Uma proposta de remodelagem do software foi feita e os primeiros passos foram dados. A ideia é a reformulação do projeto em camadas, onde cada uma tem suas funções. O projeto será dividido em três camadas: hardware, sistema operacional e aplicação.

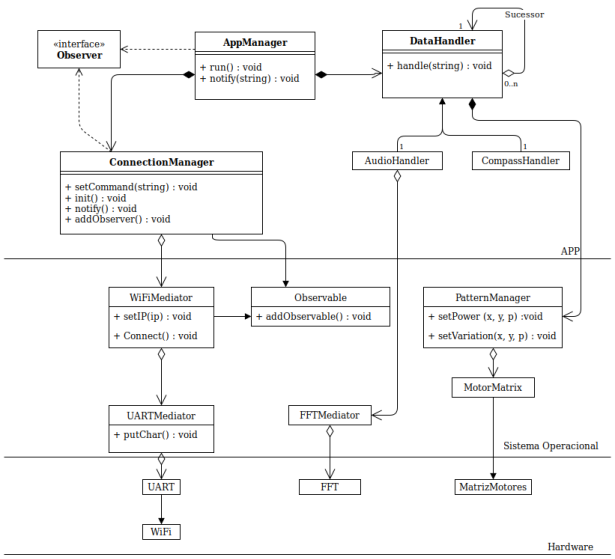

Figura 3: Diagrama de classes do projeto reformulado

- **Hardware**: Parte mais baixo nível, correspondendo toda a parte de hardware do projeto, logo, componentes desta camada são a UART, o Wi-Fi e a matriz de motores;
- **Sistema operacional**: Será a camada responsável em fazer a comunicação entre a camada de aplicação com o hardware, portanto, componentes nesta camada são os mediadores;
- **Aplicação**: Camada responsável com a recepção dos dados especificados pelo usuário, tal qual padrão usar, e da administração dos padrões, como o pacote recebido deve ser tratado pela aplicação e respostas vindas da conexão Wi-Fi estabelecida.

O diagrama de classes na figura 3 descreve a estrutura da remodelagem do projeto.

A meta é que cada componente realize apenas uma, e muito bem, dada tarefa. Além disso, a proposta é deixar o projeto extensível, que o torne mais fácil de adicionar novos recursos, como por exemplo, ao adicionar um novo componente de coleta de dados - simplesmente fazer com que a classe nova seja filha de `DataHandler`, assim, esta classe, através do padrão de cadeia de responsabilidade, pergunta a qual das classes filhas é apta para executar a tarefa que recebeu do aplicativo de smartphone (ou seja, pelo usuário).

5. REFERÊNCIAS

[1] Altera. *Nios II Classic Software Developer's Handbook*. Altera, 2015. Acessado em 01/10/2016.

[2] D. Eagleman. Can we create new senses for humans? Acessado em 29/11/2016, 2015.

[3] I. FPGA. Download center. Acessado em 18/08/2016.