Cornestick: Um Joystick de Sopro para Jogos Digitais

Victor Travassos Sarinho*

Universidade Estadual de Feira de Santana – UEFS, Lab. de Entretenimento Digital Aplicado – LEnDA, Brasil

RESUMO

Diferentes interfaces de controle podem ser usadas para fornecer entrada de comandos para jogos digitais. O processamento de sinais fisiológicos e de detecção de movimentos se apresenta como uma abordagem alternativa para a comunicação de seres humanos com jogos digitais. Este trabalho apresenta o desenvolvimento de um joystick de sopro para jogos digitais denominado *Cornestick*. Para tal, movimentos direcionais e de sopro em uma corneta são capturados por sensores Arduino e convertidos em comandos equivalentes de controle emitidos por joysticks tradicionais em jogos digitais. Como resultado, obteve-se um joystick capaz de executar comandos básicos necessários para a execução de partidas em jogos que utilizem movimentos direcionais e apenas um botão de tiro. Testes preliminares realizados com crianças e adultos também confirmaram *Cornestick* como uma solução de desempenho e de jogabilidade satisfatórios, capaz de garantir uma nova perspectiva de diversão, interação e reabilitação de pacientes através de jogos clássicos existentes.

Palavras-chave: jogos, joystick, sopro, arduino.

1 INTRODUÇÃO

Diferentes interfaces de controle podem ser usadas para fornecer entrada de comandos para jogos digitais. Tais interfaces possuem interruptores analógicos ou digitais que fornecem sinais elétricos que podem ser mapeados para comandos específicos a depender do jogo digital escolhido [1].

O processamento de sinais fisiológicos e de detecção de movimentos tem se apresentado como uma abordagem alternativa para a comunicação natural e intuitiva de seres humanos com aplicações computacionais diversas [2]. Entre as medidas fisiológicas comuns já utilizadas em jogos digitais ou na Interação Homem-Computador (HCI), pode-se destacar: cardiovascular, tensão eletrodérmica, muscular, ocular, temperatura da pele, atividade cerebral e medidas respiratórias [2].

Com o objetivo de fornecer uma interface de controle de jogos digitais baseada em sinais fisiológicos e de detecção de movimentos, este trabalho apresenta o desenvolvimento de um joystick de sopro para jogos digitais. Para tal, movimentos direcionais e de sopro em uma corneta são capturados por sensores Arduino [3] e convertidos em comandos equivalentes de controle emitidos por joysticks tradicionais em jogos digitais. A ideia é ampliar as formas de interação de jogadores com jogos clássicos existentes, permitindo seu uso futuro no tratamento e na reabilitação de pacientes em geral através da estimulação de sinais fisiológicos monitorados pelo joystick proposto.

Para fins de explanação do joystick desenvolvido, este artigo apresenta na seção 2 trabalhos relacionados a joysticks baseados no sopro e na respiração, e em formas de interpretação de sinais de sopro humano. A seção 3 descreve a abordagem de implementação hardware/software aplicada em sensores Arduino para desenvolver o joystick proposto. A seção 4 apresenta os resultados preliminares de desempenho e de jogabilidade obtidos com o uso do joystick desenvolvido. Finalmente, a seção 5 apresenta as conclusões e os trabalhos futuros deste projeto.

*e-mail: vsarinho@uefs.br

2 TRABALHOS RELACIONADOS

Diversas abordagens têm sido aplicadas na monitoração de sinais relacionados ao sopro humano produzido por um jogador.

Como exemplo, Al-Junaid, Saif e Yacoop [4] desenvolveram um algoritmo para a identificação de sons de sopro humano, através de um método de eliminação de sons baseado na distribuição única dos níveis de energia dos sinais de voz e de não-voz emitidos por um ser humano.

Patel e Abowd [5] desenvolveram uma interface que interpreta o ato de soprar em um laptop ou tela de computador para controlar diretamente algumas aplicações interativas. Usando apenas um microfone comum, esta interface realiza estimativas de localização do sopro na tela, tendo como base a resposta de frequência do sopro em certas regiões da tela do computador.

Arroyo-Palacios e Romano [2] projetaram uma interface respiratória-computador (RCI), a qual foi avaliada por um mini-jogo onde os jogadores participavam de uma corrida para explodir balões virtuais em 3D.

Zielasko et al. [6] desenvolveram o *BlowClick*, uma ferramenta que captura o som da respiração do usuário com um microfone e efetua "disparos" quando ela estiver acima de um determinado limite previamente calibrado.

Chen [7] desenvolveu o *Blowatch*, um método de entrada para *smartwatches* que utiliza sopros e gestos para invocar várias operações do dispositivo, tais como ajustar o volume da música, tirar uma foto ou atender uma chamada telefônica.

Finalmente, Chen e Osman [1] propuseram uma interface para detecção da orientação e da magnitude de sopro da respiração de um usuário, através do acompanhamento do resíduo termal da respiração e do uso do histórico de imagens térmicas do sopro.

3 DESENVOLVIMENTO

Para o desenvolvimento do *Cornestick*, foram utilizados 3 sensores Arduino para monitoração do sopro e de movimentos direcionais na corneta. Um sensor de vibração e um sensor de intensidade de som aplicados na parte interna da corneta ficaram responsáveis pela captura de dados indicadores de sopro do jogador. Um acelerômetro na ponta frontal da corneta ficou responsável pela captura de movimentos de inclinação da corneta, de modo a simular os comandos direcionais de um joystick a serem emitidos. A Figura 1 ilustra a distribuição dos sensores conectados ao Arduino em um protótipo inicial do joystick proposto.

3.1 Esquema de Hardware

Cornestick tomou como base comandos de teclado para simular comandos equivalentes de controle emitidos por joysticks tradicionais usados em jogos digitais. Desta forma, se fez necessário uma placa Arduino compatível com a biblioteca de envio de comandos de teclado para computadores/dispositivos conectados. Entretanto, apenas algumas versões mais atuais do Arduino, tais como *Duo*, *Leonard* e *Micro*, apresentam suporte para tal biblioteca. Outro fator a ser considerado é a compatibilidade com as portas *SCL* e *SDA* para fins de monitoração do

acelerômetro, as quais não estão presentes em todas as versões Arduino disponíveis. Assim, para o desenvolvimento do protótipo inicial do *Cornestick*, foi escolhido o Arduino *Micro R3 ATmega32u4*, devido ao seu tamanho reduzido para o encaixe na parte interna da corneta, compatibilidade com a biblioteca *Keyboard* para simulação de comandos de um joystick, e a compatibilidade com as portas SCL e SDA para a monitoração de dados do acelerômetro.

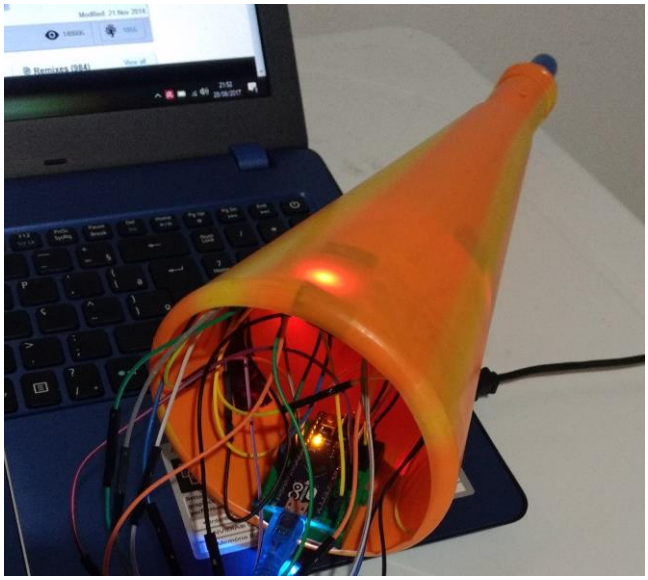

Figura 1: Protótipo inicial do Cornestick.

Com relação aos sensores escolhidos, a combinação dos sensores de vibração *SW-420* e de intensidade de som *KY-038* busca evitar falsos-positivos na confirmação de sopros na corneta, causados pelo simples chacoalhar da corneta ou pela detecção de sons extras presentes no ambiente. Ambos os sensores são monitorados por entradas digitais e analógicas da placa Arduino, os quais indicam a presença digital de vibração e a intensidade analógica do som detectado. As calibrações dos sensores foram realizadas manualmente, de modo a detectar variações significativas apenas com o soprar da corneta.

Como relação ao acelerômetro *ADXL345*, este permite detectar variações de posição da corneta nos eixos X, Y e Z do espaço através da coleta de sinais das portas SCL e SDA monitoradas. Para a versão atual do *Cornestick*, apenas os dados de dois eixos foram utilizados de modo a enviar comandos direcionais de teclado para fins de simulação do joystick proposto, tais como *KEY_UP*, *KEY_DOWN*, *KEY_LEFT* e *KEY_RIGHT*.

Para fins de ilustração, a Figura 2 apresenta o esquema de integração dos sensores Arduino utilizados no *Cornestick* com as portas indicadas na placa Arduino escolhida.

3.2 Software Gerador de Comandos

Sistemas Arduino realizam a ativação e o monitoramento de sensores a partir da execução de dois procedimentos base: *setup* e *loop*. *Setup* é responsável pela ativação do acelerômetro, pela definição dos pinos de entrada dos sinais de vibração e de intensidade de som, e pela inicialização das bibliotecas *Serial* e *Keyboard*. *Loop* efetua a monitoração e o tratamento cíclico dos sinais obtidos pelos sensores Arduino. Neste processo, a cada 50 milissegundos, a implementação de *loop* no *Cornestick* verifica os valores obtidos por cada sensor e decide quais comandos simuladores de joystick devem ser enviados para o dispositivo/computador conectado.

Para os valores de som, estipulou-se um sinal com intensidade duas vezes maior do que o apresentado durante a captação do som sem ruído para indicar a presença de um som de sopro. Já para os valores de vibração, a detecção do sinal positivo no pino digital torna possível indicar a vibração da corneta após um sopro. Com a junção das detecções dos sensores confirmando um sopro na corneta, o procedimento *loop()* efetua o envio de um espaço em branco via *Keyboard* para o computador/dispositivo conectado a USB do Arduino, simulando assim o pressionar de um botão de um joystick em um jogo.

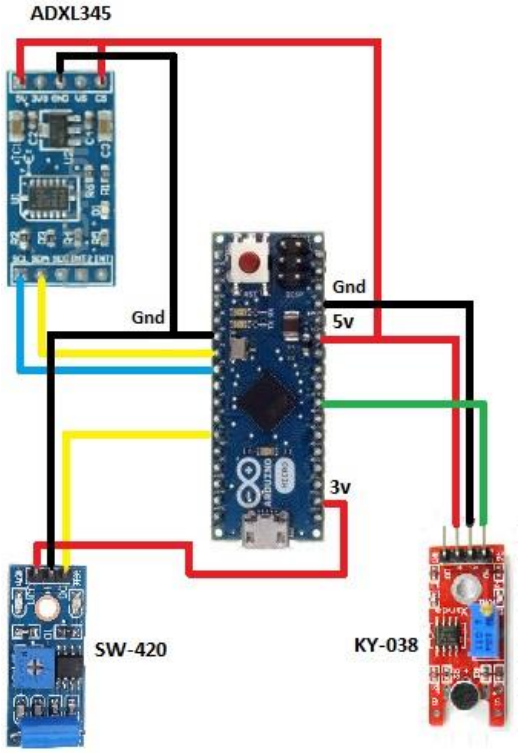

Figura 2: Esquema de integração dos sensores com a placa Arduino.

Para o acelerômetro, definiu-se inclinações de 45º (+/- 5 m/s²) no eixo Y para indicar movimentos direcionais para a direita e para a esquerda em um joystick. Caso a inclinação seja maior do que 45º para qualquer dos lados neste eixo, efetua-se o envio dos respectivos *KEY_LEFT* e *KEY_RIGHT* via biblioteca *Keyboard*, através dos comandos *Keyboard.press(KEY_LEFT_ARROW)* e *Keyboard.press(KEY_RIGHT_ARROW)* por exemplo. Para o eixo X, definiu-se inclinações de 36º (+/- 4 m/s²) como indicadores de movimentações direcionais para cima e para baixo em um joystick. Se a inclinação for maior do que 36º para qualquer dos lados neste eixo, efetua-se o envio dos respectivos *KEY_UP* e *KEY_DOWN* via biblioteca *Keyboard*, através dos comandos *Keyboard.press (KEY_UP_ARROW)* e *Keyboard.press(KEY_DOWN_ARROW)* por exemplo. Vale salientar que a definição de quais eixos monitorar (X, Y ou Z) depende apenas do posicionamento do acelerômetro aplicado na parte frontal da corneta.

A Figura 3 descreve os códigos *setup* e *loop* para inicialização e monitoração de sensores aplicado na placa Arduino, responsáveis pela definição de quais comandos de controle enviar para o computador/dispositivo conectado.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <Keyboard.h>

int pin_noise = A4;
int pin_vibration = 7;

int v_noise = 0;
int v_vibration = 0;

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void setup(void)
{
  Serial.begin(9600);

  if(!accel.begin()) {
    Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");
    while(1);
  }
  accel.setRange(ADXL345_RANGE_16_G);

  pinMode(pin_noise, INPUT);
  pinMode(pin_vibration, INPUT);

  Keyboard.begin();
}

void loop(void)
{
  Keyboard.releaseAll();

  sensors_event_t event;
  accel.getEvent(&event);

  v_noise = analogRead(pin_noise);
  v_vibration = digitalRead(pin_vibration);

  if (v_noise > 100 && v_vibration > 0)  Keyboard.write(' ');

  if (event.acceleration.y > 5) Keyboard.press(KEY_LEFT_ARROW);
  if (event.acceleration.y < -5) Keyboard.press(KEY_RIGHT_ARROW);
  if (event.acceleration.x > 4) Keyboard.press(KEY_DOWN_ARROW);
  if (event.acceleration.x < -4) Keyboard.press(KEY_UP_ARROW);

  delay(50);
}
```

Figura 3: Setup de inicialização e loop de monitoração e tratamento dos sensores Arduino.

4 Resultados e Discussão

Fazendo um breve comparativo entre o *Cornestick* desenvolvido e os trabalhos relacionados previamente descritos, não se produziu um algoritmo novo de interpretação de sons de sopro conforme Al-Junaid, Saif e Yacoop [4] porque os sensores Arduino aplicados já traziam este tipo de informação conforme os sinais de vibração e sons obtidos. Não se utilizou do microfone do laptop para interpretar a localização do som do sopro conforme Patel e Abowd [5] porque esta abordagem não apresenta uma ergonomia adequada para seu uso em jogos digitais, sem contar a necessidade constante de reconfiguração do dispositivo para cada laptop utilizado. As estratégias utilizadas por Chen [7], Zielasko et al. [6] e Arroyo-Palacios e Romano [2] se mostraram bem interessantes, porém se limitaram apenas a captura do "movimento de disparo", limitando assim sua capacidade de representar todos os movimentos desejados por um jogador em uma partida. Finalmente, a proposta de Chen e Osman [1] é bastante interessante, contudo, além de exigir o uso de um hardware mais complexo, ela não apresenta o conceito de brinquedo tangível para o jogador, algo que deve ser considerado em pacientes com deficiências tratáveis através de estimulação do sopro.

Com relação aos testes de usabilidade no protótipo inicial do *Cornestick* desenvolvido, estes foram realizados com um pequeno grupo de 3 crianças e 2 adultos em partidas aleatórias de 2 jogos clássicos distintos disponíveis na web, *Space Invaders* [8] e *Missile Command* [9]. Tratam-se de jogadores que já tiveram alguma interação em algum momento com jogos casuais clássicos, sendo assim capazes de efetuar um comparativo de jogabilidade do *Cornerstick* com a abordagem de interação tradicional neste estilo de jogo.

Para a metodologia de execução dos testes, definiu-se com 3 partidas ou duração máxima de 10 minutos para cada jogador em cada jogo utilizando o joystick proposto. Após a execução das partidas nos dois jogos, efetuou-se o questionamento e a coleta das respostas dos jogadores envolvidos. Para fins de ilustração, a Figura 4 apresenta a utilização do *Cornestick* durante uma partida de *Space Invaders* em sua fase inicial de testes.

Figura 4: Uso do Cornestick em uma partida de Space Invaders.

A coleta de opiniões qualitativas dos jogadores teve como base os grupos de questões propostos por Lund [10], tais como utilidade, facilidade de uso, facilidade de aprendizagem e satisfação com o joystick proposto. Dentre as opiniões coletadas, percebeu-se que o peso, o desempenho e a ergonomia do *Cornestick* se mostraram adequados para a realização das partidas. Entretanto, a movimentação de inclinação da corneta como direcional para a esquerda e direita não foram efetivamente naturais/espontâneas para os jogadores nos primeiros momentos.

Houve também algumas reclamações com relação não detecção de sopros em alguns "disparos" durante algumas partidas jogadas em versões web dos jogos indicados. O ritmo frenético de alguns jogos também se tornou um dificultador para o uso do *Cornestick*, uma vez que se tornou cansativo efetuar vários disparos contínuos com o uso do sopro em uma partida. Para ambos os casos, tem-se como possíveis soluções a implementação futura de uma calibração dinâmica dos sensores para aumentar a repetição do envio de comandos e o desenvolvimento de jogos específicos para o ritmo de disparos com sopros.

Finalmente, com relação aos custos e esquemas desenvolvidos para o *Cornestick*, estes se mostraram simples, baratos, viáveis e amplamente disponíveis para a comunidade, podendo ser facilmente validado e replicado por terceiros conforme esquema ilustrado na Figura 2 a um custo de aproximadamente $10 cada.

5 Conclusões e Trabalhos Futuros

Este artigo apresentou o desenvolvimento e os testes iniciais de um joystick de sopro denominado *Cornestick*. Trata-se de um joystick capaz de executar comandos básicos necessários para a execução de partidas em jogos clássicos e casuais que usem movimentos direcionais e apenas um botão de tiro.

Cornestick foi desenvolvido tendo como base a plataforma Arduino, garantindo assim uma solução de baixo custo dentro do conceito de open hardware compartilhado. O desempenho e a jogabilidade proposta pelo joystick também apresentaram resultados satisfatórios nos testes preliminares realizados com crianças e adultos.

Dentre as limitações encontradas no *Cornestick*, se faz necessário acrescentar um interruptor e um botão de pressão na corneta para fins de liberação do envio de sinais do acelerômetro. Isto é necessário para melhorar a jogabilidade do joystick, uma vez que a versão atual envia continuamente os sinais do acelerômetro quando plugado, gerando assim uma movimentação contínua e ininterrupta do jogador mesmo em momentos indesejados durante uma partida corrente. Outra limitação é a ausência de uma calibração dinâmica dos sensores via software, a qual exige a produção de uma interface futura para a definição dos limiares de captação desejados conforme jogador/jogo/ambiente corrente.

Como trabalhos futuros, pretende-se efetuar o desenvolvimento de jogos específicos para um melhor aproveitamento das funcionalidades e das características físicas do joystick proposto, ao invés de usá-lo apenas como um substitutivo de controles atualmente disponíveis.

Outros joysticks baseados no sopro também serão desenvolvidos, os quais irão monitorar mudanças de temperatura, pressão, umidade e movimentos causados pelo sopro em si. A ideia é utilizá-los em conjunto com o *Cornestick* em terapias fonoaudiológicas para fins de validação dos mesmos como ferramentas de apoio a jogos sérios voltados para saúde de crianças e adultos em geral.

REFERÊNCIAS

[1] R. Chen and S. Osman. Blow tracking user interface system and method. *U.S. Patent* 8,786,698, issued July 22, 2014.

[2] J. Arroyo-Palacios and D. M. Romano. Exploring the use of a respiratory-computer interface for game interaction. In *Games Innovations Conference*, 2009. *ICE-GIC* 2009. *International IEEE Consumer Electronics Society's*, pp. 154-159. IEEE, 2009.

[3] Arduino. What is Arduino? http://arduino.cc/, September 2017.

[4] H. Al-Junaid, A. M. Saif and F. Y. AlWazzan. Design of Digital Blowing Detector. *International Journal of Information and Electronics Engineering* 6, no. 3, pp. 180, 2016.

[5] S. N. Patel and G. D. Abowd. Blui: low-cost localized blowable user interfaces. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pp. 217-220. ACM, 2007.

[6] D. Zielasko, S. Freitag, D. Rausch, Y. C. Law, B. Weyers and T. W. Kuhlen. BlowClick: A Non-Verbal Vocal Input Metaphor for Clicking. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, pp. 20-23. ACM, 2015.

[7] W. H. Chen. Blowatch: Blowable and Hands-free Interaction for Smartwatches. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on human factors in computing systems*, pp. 103-108. ACM, 2015.

[8] Space Invaders. Space Invaders remixed by ifugu. https://scratch.mit.edu/projects/1979494/, August 2011.

[9] Missile Command. Missile Command by wkelly42. https://scratch.mit.edu/projects/63635586/, May 2015.

[10] A. M. Lund. Measuring Usability with the USE Questionnaire12. *Usability interface*, 8(2), 3-6, 2001.