UFRN

Relatório de Sistemas Embarcados

Arturo Fonseca de Souza

26/07/2022

Som imaginário

Aprendendo sobre a física do som brincando

---

1 Introdução

O uso de tecnologias como ferramentas auxiliares de ensino em atividades experimentais está se tornando uma prática cada vez mais comum. Elas vem se mostrando muito eficientes em cativar o interesse de estudantes em disciplinas de ciências nas escolas ao redor do mundo, particularmente na área de Física [1, 2]. De acordo com Petry et al, essas atividades proveem oportunidades para discussão e estabelecem ligações com situações da vida real, sendo importantes para um aprendizado significativo.

Aliado a essas tecnologias, a gamificação, isto é, a prática de associar técnicas e recursos de jogos e aplicá-las em outros contextos, de acordo com Souza, possibilita a criação de dinâmicas de grupos, ao aplicar um novo conceito na educação, com tecnologias de jogos digitais abrangendo desafios na execução das atividades escolares, e tornando a prática do ensino mais lúdica [3].

O intuito deste projeto é, então, se utilizar da gamificação para desenvolver um dispositivo lúdico, de baixo custo, voltado ao ensino de Física, mais especificamente sobre o comportamento de ondas sonoras e como o ser humano consegue discernir as suas origens no espaço.

Esse dispositivo seria utilizado em salas de aula tanto para estudantes no ensino fundamental como de ensino médio, colocando-os numa posição de maior participação e autonomia no processo de aprendizado. Nesse processo também seria promovido, naturalmente, a interdisciplinaridade entre as áreas de Física, Matemática, Eletrônica e Informática.

2 Descrição do sistema

O sistema consiste basicamente em um jogo formado por um dispositivo, com dois sensores de som, e um aplicativo de celular. Aliado ao dispositivo há uma tela LCD, para exibir informações pertinentes, um servomotor, que irá apontar para a origem de um som alto e curto (como um bater de palmas), um potenciômetro, botão para interações do usuário e um microcontrolador para controlar todos esses componentes.

Antes de explicar o jogo e suas diferentes modalidades é necessário explicar como o sistema consegue localizar a origem de um certo som:

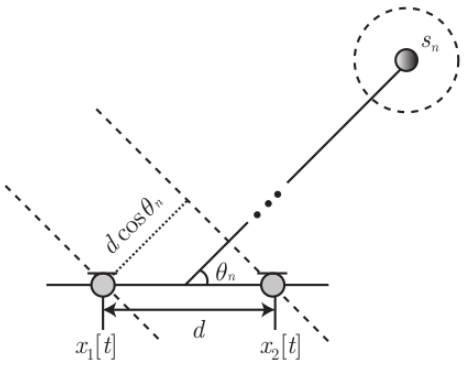


Figura 1: Calculando origem do som. Fonte: Frebregat et al. [4]

Considere os dois sensores de som do dispositivo, $m = 1, 2$ a uma distância d um do outro e uma fonte de som, S_n, emitindo som de um ângulo $\theta \in [0, \pi]$ com respeito ao centro dos dois microfones (veja Figura 1). A origem do som é calculada considerando a diferença de tempo em que o som chega em cada sensor. Essa diferença de tempo pode ser dada por

$$\Delta t = \frac{d \cdot cos(\theta)}{c} [4] \tag{1}$$

Com c sendo a velocidade do som, aproximadamente 343 m/s. Logo, a direção do som em relação aos sensores, isto é, θ, pode ser dada por

$$\theta = cos^{-1}(\frac{\Delta t \cdot c}{d}) \tag{2}$$

Essa equação é usada pelo dispositivo nos dois modos de jogos disponíveis e é esperado que os estudantes a conheçam e entendam-a:

2.1 Modo I

O Modo I é o responsável pelo nome do projeto, Som Imaginário. Nesse modo o dispositivo mostraria em sua tela a diferença de tempo em que um certo som "imaginário"chegou em cada sensor e em qual sensor ele chegou primeiro.
Cada estudante, então, calcularia a direção desse som utilizando a equação (2) e tentaria girar um ponteiro associado a um potenciômetro no dispositivo, submetendo sua resposta junto com seu nome pelo aplicativo do celular. Quando o último estudante submetesse sua tentativa, o sistema apontaria para a direção correta da origem do som "imaginário"e mostraria o nome da pessoa vencedora em sua tela.

2.2 Modo II

O Modo II consiste na escolha de um estudante para ficar de olhos vendados junto ao dispositivo. Depois, alguma outra pessoa é escolhida para bater palmas. O estudante perto do dispositivo teria que apontar para a direção do som, ao mesmo tempo em que o dispositivo também tentaria apontar para a direção do som com seu servomotor. Como o dispositivo só consegue reconhecer e apontar sons a sua frente, um som produzido atrás dele levaria ele a apontar para o ângulo espelhado no plano dos microfones. Essas inconsistências poderiam ser discutidas em aula, com orientações de um(a) professor(a).

3 Requisitos funcionais e não funcionais

RFH01	Detectar a direção de sons de curta duração e alto volume (amplitude).
RFH02	Indicar direção de um som através de um servomotor.
RFH03	Mostrar tempos de chegada das primeiras incidências de sons nos sensores em displays LCD
RFH04	Capturar tentativas de acerto da direção do "som imaginário".
RFH05	Enviar tentativas de acerto da direção de um "som imaginário"(modo I) com o pressionamento de um botão para o aplicativo.
RNH06	Identificar sons com precisão suficiente para capturar a diferença de tempo entre dois sensores de som com num mínimo 20cm de distância um do outro.
RNH07	Apontar origem de um som pontual com precisão de 98% (ângulo), desde que a origem do som esteja entre os primeiros 180° graus considerando a frente do dispositivo.
RNH08	Possuir autonomia de bateria de pelo menos 2 horas.
RNH09	Custar menos de R$140,00.

Tabela 1: Requisitos do dispositivo.

RFS01	Permitir seleção de modo de jogo.
RFS02	Enviar informações de descisões para dispositivo via Bluetooth.
RFS03	Exibir informações básicas do funcionamento de ondas sonoras assim como regras dos dois modos de jogo.
RFS04	Registrar distância entre sensores passada pelo usuário e envia-la para dispositivo via Bluetooth.
RFS05	Registrar identificação de tentativas de acerto.
RNS06	Possuir menos de 10MB.
RNS07	Funcionar em dispositivos Android e IOS.
RNS08	Gastar menos que 1% de bateria a cada 3 minutos (desconsiderando gasto de energia da tela).

Tabela 2: Requisitos do aplicativo.

4 Modelagem

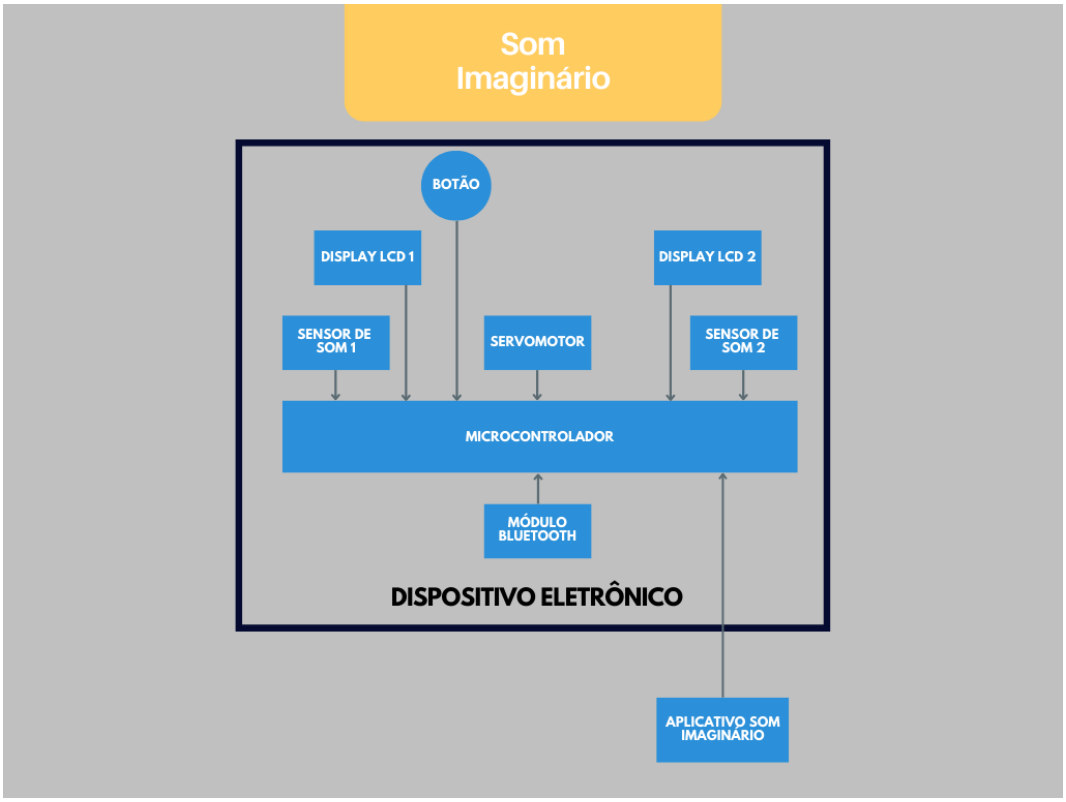


Figura 2: Diagrama de blocos do sistema.

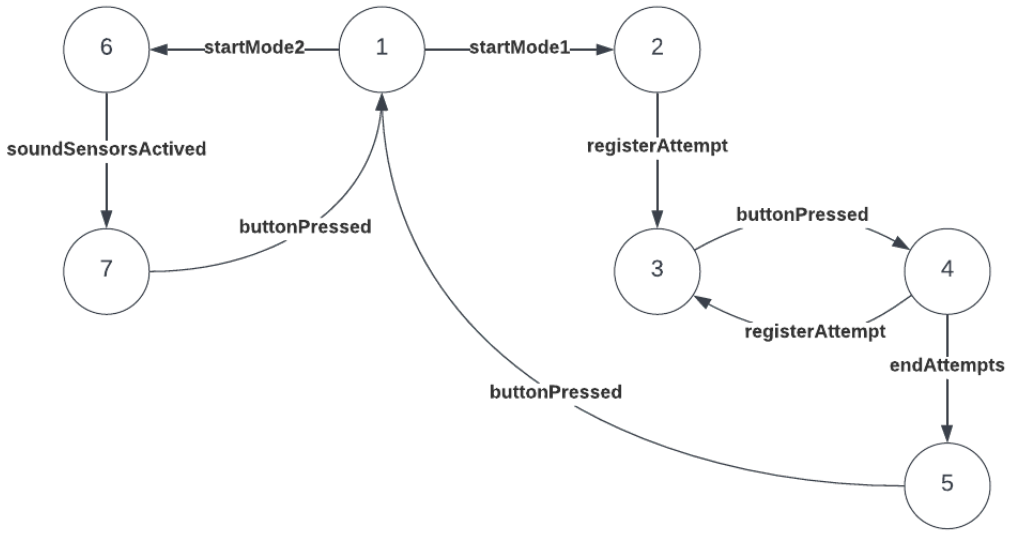


Figura 3: Diagrama de estados do dispositivo.

Os estados do diagrama de estados do dispositivo (Figura 3) seriam:
1: Espera sinal de início;
2: Informa duas marcas de tempo;
3: Espera entrada de tentativa;
4: Envia tentativa para aplicativo;
5: Mostra direção do "som imaginário";
6: Espera som de palmas;
7: Mostra direção do som das palmas.

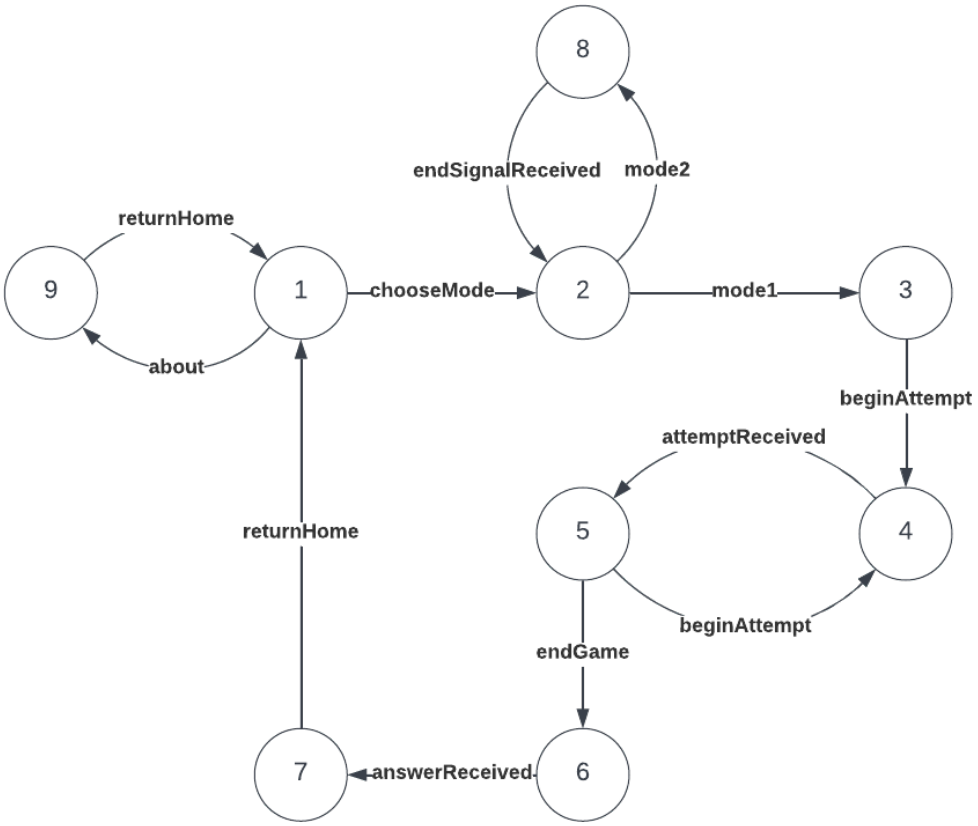


Figura 4: Diagrama de estados do aplicativo.

Os estados do diagrama de estados do aplicativo (Figura 4) seriam:
1: Exibe tela de home;
2: Mostra tela de opções de modos de jogo;
3: Envia sinal para dispositivo informando início do modo 1 e mostra tela de registro de tentativas;
4: Registra nome da pessoa e envia sinal de começo de tentativa para dispositivo. Aguarda resposta do dispositivo;
5: Registra tentativa associada a pessoa;
6: Envia sinal de fim de jogo para dispositivo junto com valor da distância entre sensores e aguarda reposta do dispositivo;
7: Exibe nome da pessoa vencedora;
8: Envia sinal para dispositivo informando início do modo 2 e aguarda sinal de término do disposi-

tivo;
9: Exibe informações sobre a física do jogo e instruções do dois modos.

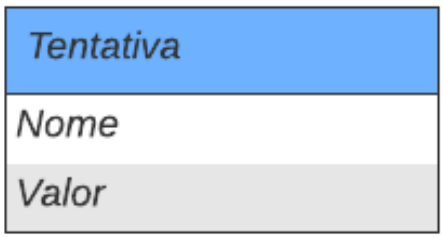


Figura 5: Diagrama de dados relacional (aplicativo).

Figura 6: Telas do aplicativo.

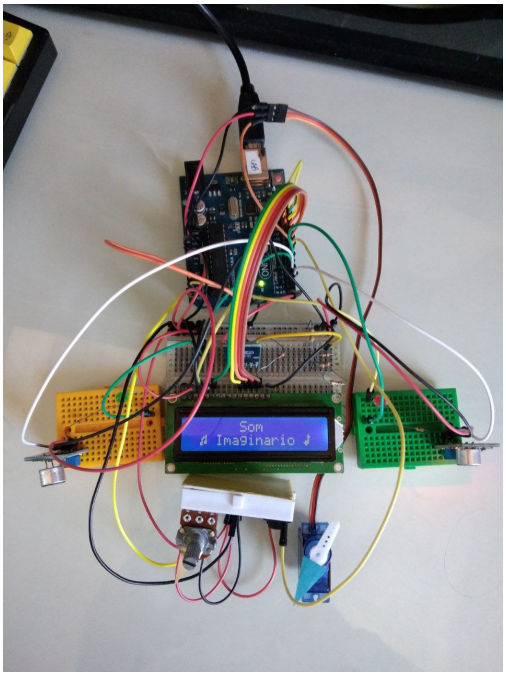
5 Implementação

5.1 Aplicativo

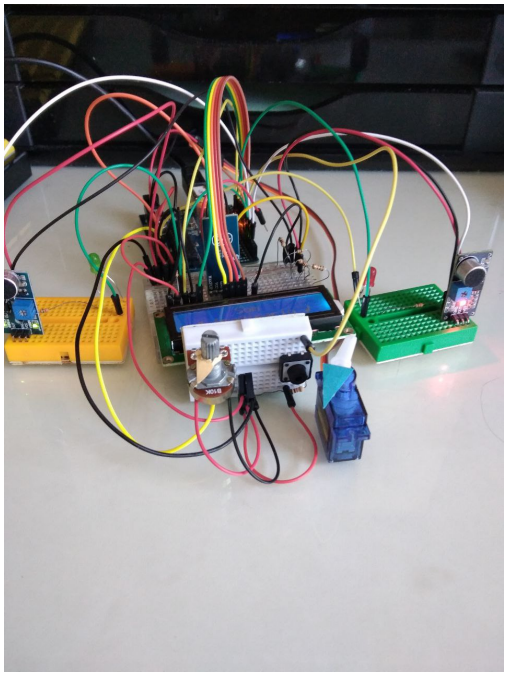
Devido a problemas de organização de tempo, não foi possível implementar o aplicativo que gerenciaria os modos de jogo e o registro de tentativas no Modo I, assim como mostraria informações relevantes a cerca do jogo e a física envolvida. Logo, nenhum requisito associado ao aplicativo foi atendido.

5.2 Dispositivo

O componente central do dispositivo, o microcontrolador, passou por umas mudanças durante a implementação. Inicialmente, o módulo ESP32 foi escolhido devido sua dimensões compactas e por possuir módulo Bluetooth e Wi-Fi integrado. Mas ele não era muito compatível com outros componentes por possuir saídas de 3.3V em suas portas lógicas. Logo, a plataforma Arduino foi escolhida, mais especificamente sua placa Arduino Uno, muito semelhante ao ESP32, mas não possui os módulos de comunicação sem fio integrados, além ter um tamanho um pouco maior. Em compensação, por ser um microcontrolador conhecido, é menos custoso, mantendo o objetivo de baixo custo do projeto.
Como o aplicativo não foi desenvolvido, a parte de comunicação Bluetooth do dispositivo também não.
A falta do aplicativo influenciou algumas mudanças no comportamento do sistema como um todo. A seleção de modos de jogo ficou dentro do próprio dispositivo. O Modo I do jogo só registra uma tentativa, anunciando instantaneamente se ela está parcialmente correta ou não.
Quase todos requisitos funcionais do dispositivo foram cumpridos (veja Tabela 2). O requisito RFH05 não foi cumprido devido a falta do aplicativo.
O requisito não-funcional RNH07 não foi cumprido completamente devido a problema no código que serão explicados posteriormente.
Os requisitos não-funcionais RNH08 e RNH09 não foram testados / calculados.



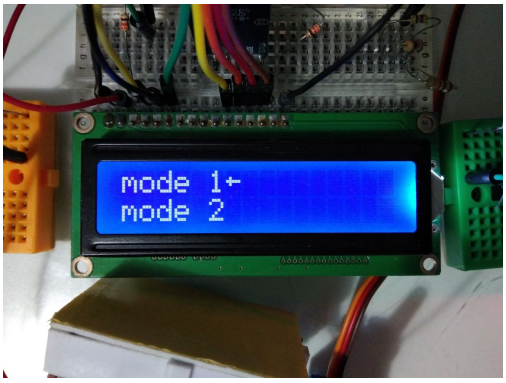
(a) Vista superior.



(b) Vista frontal.

Figura 7: Fotos do dispositivo

Figura 8: Foto da tela inicial.



(a) Foto da tela de escolha do modo de jogo.

(b) Foto da tela de informações do som.

Figura 9: Fotos das telas.

A língua escolhida para interface do sistema foi o inglês, por ser uma língua mais difundida internacionalmente, permitindo que a divulgação do projeto fosse facilitada. Mas como a ideia inicial era desenvolver o sistema para ser usado por escolas públicas brasileiras, foi planejado, inicialmente, uma opção de mudanças entre as línguas português brasileiro e inglês, que acabou não sendo implementada.

A tela inicial (ver Figura 8) é a primeira tela mostrada quando o sistema é iniciado. Quando o botão é pressionado, a tela de escolha de modo de jogo (ver Figura 9a) é mostrada. Selecionando o Modo I (*mode 1*), a tela de informações do som (Figura 9b) é mostrada. Nela é exibido qual sensor recebeu primeiro o som e também a diferença de tempo entre os sensores em micro-segundos.

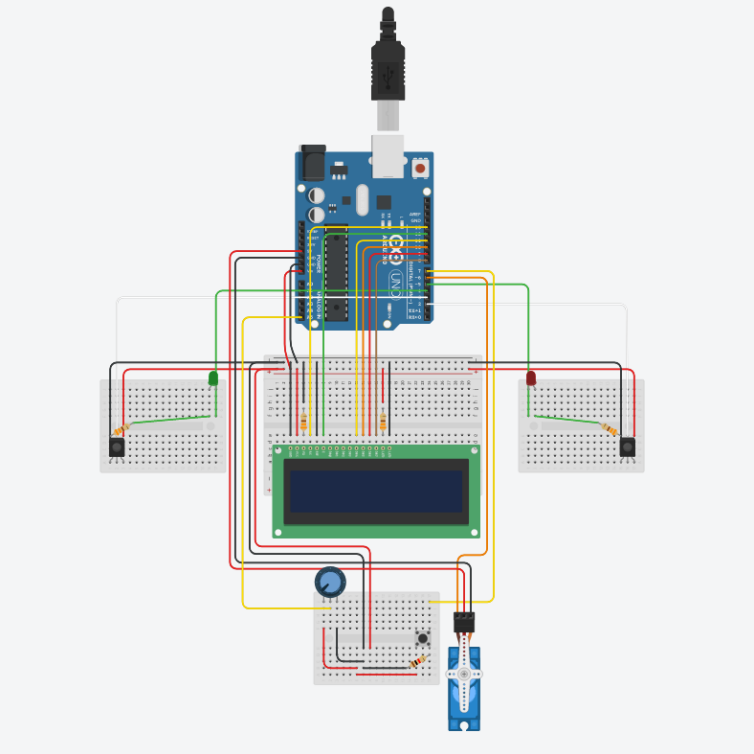


Figura 10: Esquemática do circuito do dispositivo.

5.3 Código do dispositivo

A linguagem que é usada para programar o microcontrolador Arduino Uno é baseada em C++. O seu código é compilado e enviado através da Arduino IDE. Abaixo é mostrado um resumo do código e suas funções.

```
309 void loop(){
310   if(gameMode == 2) {
311     mode2();
312   }
313   else if(gameMode == 1) {
314     mode1();
315   }
316
317   int buttonVal = digitalRead(button);
318   if(buttonVal == HIGH && !gameRunning) {
319     gameMode = showMenu();
320     gameRunning = true;
321   }
322 }
```

Figura 11: Função principal.

A Figura 11 mostra a função *loop*, que será pecorrida muitas vezes por segundo na execução do código no dispositivo. Ela consiste basicamente na chamada de duas funções: *mode1* e *mode2*, que representam, respectivamente, os dois modos do jogo. Ao primeiro aperto do botão, a tela de seleção de modos de jogo é mostrada, representada pela função *showMenu*.

```
165   while(true) {
166     int buttonVal = digitalRead(button);
167
168     //First moment that the button is pressed
169     if(buttonVal == HIGH && !buttonPressed) {
170       buttonPressed = true;
171       buttonTimePressed = millis();
172     }
173     /*If the button is pressed for greater or equal than the time defined by buttonAction2Time,
174     then the option selected is toggled*/
175     else if(buttonVal == HIGH && buttonPressed && (millis() - buttonTimePressed) >= buttonAction2Time) {
176       option = (option == 1)? 2 : 1;
177       buttonPressed = false;
178       buttonTimePressed = 0;
179       drawMenu(option);
180       delay(2000);
181     }
182     //If the button was pressed for less than buttonAction2Time, then the current option is selected
183     else if(buttonVal == LOW && buttonPressed) {
184       //drawSplashScreen();
185       lcd.clear();
186       lcd.setCursor(0, 0);
187       lcd.print("     mode ");
188       lcd.print(option);
189       startMode1 = (option == 1);
190       return option;
191     }
192     else if(buttonVal == LOW) {
193       buttonPressed = false;
194       buttonTimePressed = 0;
195     }
196   }
```

Figura 12: Função *showMenu*.

Dentro da função showMenu existe um "loop infinito"em que aguarda o pressionamento do botão. Se o botão é pressionado por mais de 2 segundos, a opção atual (modo de jogo selecionado) é alterada. Se o botão for "despressionado"antes desse tempo, a opção atual é selecionada e seu valor é retornado e uma das duas funções de modo de jogo é chamada na *loop*.

```
270  void mode1() {
271    if(startMode1) {
272      imaginaryFirstSide = (random(0, 2) == 0)? "left" : "right";
273      imaginaryDeltaS = random(0, 401);
274      drawMode2(imaginaryDeltaS, imaginaryFirstSide);
275      delay(2000);
276      startMode1 = false;
277    }
278
279    int buttonVal = digitalRead(button);
280    if(buttonVal == HIGH) {
281      int potentiometerVal = analogRead(potentiometer);
282  //    unsigned long imaginaryAngle = acos(((imaginaryDeltaS / 1000000L) * 343L) / 0.168L);
283  //    //Convert angle from radians to degrees
284  //    imaginaryAngle = imaginaryAngle * 57296L / 1000L;
285  //    if(imaginaryFirstSide == right)
286        Serial.println(potentiometerVal);
287        Serial.println(imaginaryFirstSide);
288      if((potentiometerVal < 430) && (imaginaryFirstSide == "right")){
289        lcd.clear();
290        lcd.print("Correct!!!");
291        delay(3000);
292      }
293      else if((potentiometerVal > 430) && (imaginaryFirstSide == "left")){
294        lcd.clear();
295        lcd.print("Correct!!!");
296        delay(3000);
297      }
298      else {
299        lcd.clear();
300        lcd.print("Try again!");
301        delay(3000);
302      }
303    }
304  }
```

Figura 13: Função *mode1*.

Nas linhas iniciais da função *mode1* (Figura 13, linhas 271-277) é escolhido valores aleatórios para a direção do som e da diferença do tempo de sua chegada nos sensores. A máxima diferença possível entres os sensores foi calculada por volta de 400 micro-segundos, que seria quando a origem do som estivesse na mesma reta em que os sensores no espaço (180° ou 0°).
O restante dessa função registra a posição do potenciômetro quando o botão é pressionado e exibe na tela do dispositivo se foi uma tentativa correta ou não.
Devido a problemas no código, provavelmente relacionado a tipagem de variáveis, não foi possível utilizar a equação (2) sem que ela exibisse resultados diferentes dos calculados manualmente utilizando os mesmos valores gerados aleatoriamente. Portanto, a tentativa é marcada como correta se o potenciômetro apontar para o mesmo quadrante da origem do som (entre 0°-90° pra direita e entre 91° 180° pra esquerda).

```
204   if(!firstS1) {
205     if(rightSensorVal == HIGH) {
206       if(!calculatingOrigin && (micros() - (firstSignal + deltaS) > timeToWaitAnotherCapture)){
207         firstSignal = micros();
208         firstS1 = true;
209         calculatingOrigin = true;
210       }
211       else if(firstS2) {
212         deltaS = micros() - firstSignal;
213         calculatingOrigin = false;
214         firstS1 = false;
215         firstS2 = false;
216         drawMode2(deltaS, "left");
217         deltaS = 12;
218         unsigned long servoAngle = acos(((deltaS / 1000000L) * 343L) / 0.168L);
219         //Convert angle from radians to degrees
220         servoAngle = servoAngle * 57296L / 1000L;
221         Serial.println(deltaS / 1000000L);
222         Serial.println(servoAngle);
223         myservo.attach(servo);
224         //Convert angle from radians to degrees
225         myservo.write(servoAngle);
226         delay(200);
227         myservo.detach();
228       }
229
230       startT1 = millis();
231       digitalWrite(led1, HIGH);
232     }
233     else if(millis() > (startT1 + deltaT1)){
234       digitalWrite(led1, LOW);
235     }
236   }
```

Figura 14: Função *mode2*.

Na função *mode2* (Figura 14) existem diversas variáveis de controle para que o algoritmo consiga identificar que sensor recebeu primeiro o som e, em seguida, a diferença de tempo de chegada desse som entre os sensores. No trecho escolhido é possível ver a lógica para um só sensor. Assim, que o sensor emite um sinal *HIGH* primeiro, o algoritmo impede que ele entre novamente nessa parte do código. Se ele for o segundo a receber o sinal (linha 211), então a diferença de tempo é calculada, juntamente com o ângulo da origem do som, seguindo a equação (2). Mas, como dito anteriormente, esses cálculos não funcionam como esperado, com *acos* retornando sempre um valor muito próximo de 1.

6 Conclusão

O projeto, então, que visa funcionar como um jogo educativo para incentivar o aprendizado de certos conteúdos de Física (intimamente interligados com as áreas de Matemática, Biologia e Computação/Eletrônica) é aparentemente implementável e funcional. Infelizmente, uma parte fundamental à questão educativa do projeto, o aplicativo, com suas informações acerca do funcionamento de todo sistema, não foi implementado a tempo.
É sempre importante lembrar que a tecnologia não é o fim quando se trata de metodologias de ensino nos ambientes escolares, como Samra nos diz [5]. Ela é apenas uma ferramenta auxiliar (mas poderosa) no ensino.

Referências

[1] C. A. Petry, F. S. Pacheco, D. Lohmann, G. A. Correa, P. Moura, Project teaching beyond physics: Integrating arduino to the laboratory, in: 2016 Technologies Applied to Electronics Teaching (TAEE), 2016, pp. 1–6. doi:10.1109/TAEE.2016.7528376.

[2] L. C. Farcas, O. F. Caltun, Experimental set-ups using microcontrollers and sensors realized by students to be used in physics lessons, Journal of Physics: Conference Series 1929 (1) (2021) 012075. doi:10.1088/1742-6596/1929/1/012075.

[3] M. R. Souza, Scratch: Gamification na educação, Revista CNEC Educação 2 (2019) 58–81. URL http://sys.facos.edu.br/ojs/index.php/cneceducacao/article/view/322/293

[4] G. Fabregat, J. A. Belloch, J. M. Badía, M. Cobos, Design and implementation of acoustic source localization on a low-cost iot edge platform, IEEE Transactions on Circuits and Systems II: Express Briefs 67 (12) (2020) 3547–3551. doi:10.1109/TCSII.2020.2986296.

[5] S. Samra, Technology in the classroom: Target or tool, Procedia - Social and Behavioral Sciences 81 (06 2013). doi:10.1016/j.sbspro.2013.06.484.