Uma Plataforma IoT para Desenvolvimento de Sistemas de Comunicação Aumentativa e Alternativa

Thiago P. Lima[1], **Robson N. Fidalgo**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

{tpl,rdnf}@cin.ufpe.br

Abstract. *Augmentative and Alternative Communication (AAC) users with limited mobility are benefited from the use of triggers and screen scanners. Currently, the available resources are connected through USB or Bluetooth connections, which require physical proximity between the trigger and the application, restricting the possibilities of use to the same trigger and the same geographic location. This feature prevents the use of these tools remotely, as is the case with care in the current COVID-19 pandemic context. To fill this gap, this work presents an IoT-based platform that connects a AAC resource to a trigger remotely. Thus, the devices can be geographically distant and be used in real time, simply using an Internet connection.*

Resumo. *Usuários de Comunicação Aumentativa e Alternativa (CAA) com limitação motora são beneficiados pelo uso de acionadores e varredores de tela. Atualmente, os recursos disponíveis se conectam por meio de conexões do tipo USB ou Bluetooth, o que exige proximidade física entre acionador e aplicativo, restringindo as possibilidades de uso a um mesmo acionador e a mesma localização geográfica. Esta característica impede o uso dessas ferramentas de maneira remota, como é o caso dos atendimentos no atual contexto pandêmico da COVID-19. Para preencher esta lacuna, este trabalho apresenta uma plataforma baseada em IoT que conecta um recurso de CAA a um acionador de maneira remota. Assim, os dispositivos podem estar distantes geograficamente e serem usados, em tempo real, bastando para isso uma conexão de Internet.*

1. Introdução

A Tecnologia Assistiva (TA) [Ashton et al. 2009, Bersch 2008] é um termo utilizado para identificar todo o arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência. Dentre estes recursos e serviços temos uma área voltada para comunicação chamada Comunicação Aumentativa e Alternativa (CAA), que tem foco em pessoas sem a fala ou com limitação neste recurso de comunicação.

A CAA engloba recursos formados por figuras que representam palavras ou expressões e que permitam a interação e o desenvolvimento da linguagem. Os recursos mais comuns são as pranchas de comunicação e os jogos educativos. Estes recursos podem ser criados com papel ou outros materiais, porém versões digitais (e.g., aplicativos e sites) também existem [Lima et al. 2017, REAACT 2021]. No caso das pranchas, a

comunicação é estabelecida por meio da seleção dos símbolos que expressem o que a pessoa com limitação quer dizer ou, na versão digital, da indicação a partir do toque na tela ou em teclados convencionais (e.g., QWERT). O mesmo acontece com os jogos educativos. Estes tipos de aplicações não levam em consideração possíveis limitações motoras que um toque fino na tela possa requerer, exigindo o uso de acionadores específicos e varredores de tela (ver Seção 2.2).

O trabalho de Reis [Reis 2017] propôs um acionador multitoque construído em Arduino para ser integrado ao aplicativo aBoard através de uma conexão bluetooth. A solução resolve o problema da limitação motora e do toque fino, mas exige proximidade física entre acionador e aplicativo, restringindo as possibilidades de uso a um mesmo acionador e a mesma localização geográfica.

A Internet das Coisas (IoT, do inglês Internet of Things) [Ashton et al. 2009] a qual possibilita a integração de objetos – tanto software (e.g., recursos de CAA) quanto hardware (e.g., acionadores) – por meio da Internet pode ser utilizada para solucionar o problema da localização geográfica. Além disso, os sensores IoT podem repassar informações para o ambiente, viabilizando possíveis integrações com luzes, telas, relógios inteligentes e assistentes virtuais. O inverso também ocorre, ou seja, pode-se ter a leitura de informações do ambiente e mudar a lógica de acionamento como, por exemplo, emitir sons, ligar luzes ou mostrar informações coletadas no próprio dispositivo de acionamento. Atualmente, existem diversos serviços de servidores voltados para esta área [Thinger 2021, ThingsBoard 2021], porém o funcionamento destes não leva em consideração os requerimentos que uma plataforma de CAA requer (e.g., controle de conexão para garantir que a comunicação de outros dispositivos não interfira no funcionamento da CAA).

O objetivo deste trabalho é utilizar a Internet das Coisas para especificar e construir uma plataforma de CAA com acionador que possa ser utilizada em diálogos e atendimentos a distância (e.g., atendimentos remotos em tempos de COVID-19). A adaptação de serviços IoT para este uso é possível, porém, demanda um esforço inicial e que tende a crescer à medida que o suporte a mais aplicações de CAA seja incluído, podendo chegar a um ponto de incongruência. O que justifica a criação de um servidor específico para este fim. Como prova de conceito desta proposta, será utilizado um jogo Caça-Letra e um acionador multitoque, os quais também são contribuição do autor deste trabalho e pertencem à Plataforma Reaact [REAACT 2021]. O restante deste trabalho está organizado da seguinte forma:

- Seção 2 – apresenta os fundamentos teóricos necessários para o entendimento deste trabalho, os quais estão divididos em Internet das Coisas e Sistemas de Varredura e Acionadores;
- Seção 3 – apresenta a Plataforma Reaact, composta por recursos de CAA (aqui será utilizado um jogo Caça-Letras) e um acionador multitoque, os quais configuram a primeira contribuição desta proposta e serão utilizados como prova de conceito da plataforma apresentada;
- Seção 4 – detalha a proposta da plataforma IoT para CAA a qual configura a segunda contribuição desta proposta;
- Seção 5 – apresenta a prova de conceito desta proposta, onde o jogo Caça-Letras e o acionador multitoque são integrados à plataforma de IoT para CAA;

- Seção 6 – apresenta as considerações finais do trabalho, bem como os direcionamentos para trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta os conceitos de Internet das Coisas (IoT), os sistemas de varredura e acionadores, os quais são os fundamentos teóricos necessários para o entendimento deste trabalho.

2.1. Internet das Coisas

A Internet das Coisas é um ecossistema baseado na interconexão de objetos físicos por meio da Internet para coletar, trocar e armazenar dados por meio de software [Carrion and Quaresma 2019]. Pode ser visto como uma extensão da Internet que possibilita a conexão de objetos do cotidiano que tenham capacidade computacional. De maneira geral, o fluxo de dados na IoT passa por quatro componentes principais que se comunicam por meio de protocolos:

- Sensor – localizado nos objetos, percebem e coletam os dados do ambiente;
- Centro de dados – localizado na nuvem, armazena e analisa os dados coletados;
- Aplicação – é o software que controla os dados analisados e fornece o serviço para o usuário final;
- Consumidor – usuário final que consome e compartilha as informações com outros serviços e pessoas.

O protocolo WebSocket foi projetado para viabilizar a troca constante e persistente de dados entre cliente e servidor. De maneira geral, o protocolo de comunicação consiste em um canal bidirecional completo em uma única conexão [LUBBERS and GRECO 2021], garantindo que a comunicação entre cliente e servidor seja feita de forma não sincronizada, ou seja, ambos os lados podem enviar dados a qualquer momento enquanto a conexão estiver estabelecida. Conforme apresentado na Figura 1, este tipo de comunicação não é suportado no protocolo HTTP [FETTE 2021]. Além disso, o trabalho de Oliveira et al. [Oliveira et al. 2018] mostra que para aplicações que visam atingir o menor Round Trip Time (RTT) possível, este é o protocolo de comunicação mais adequado.

Atualmente, existem no mercado diversas placas de baixo custo para a prototipação de aplicações IoT. Um exemplo é a ESP8266 que, além do baixo custo mencionado, também conta com facilidades como conexão USB para a alimentação e a possibilidade de usar o protocolo WebSocket para a comunicação.

2.2. Sistemas de Varredura e Acionadores

Dado que usuários de CAA também podem ter limitações motoras severas, o uso da técnica de varredura e dos dispositivos acionadores deve ser considerado para facilitar a comunicação. A varredura pode ser independente ou dependente de um mediador. A varredura independente, consiste em um marcador que destaca sucessivamente uma série de itens (e.g., botões) que podem ser selecionados pelo usuário. Já na varredura dependente, o mediador faz, de maneira manual, o papel do marcador. Segundo Tetzchner [Tetzchner 1992], a varredura pode ser feita de três formas, as quais são ilustradas na Figura 2 e apresentadas a seguir:

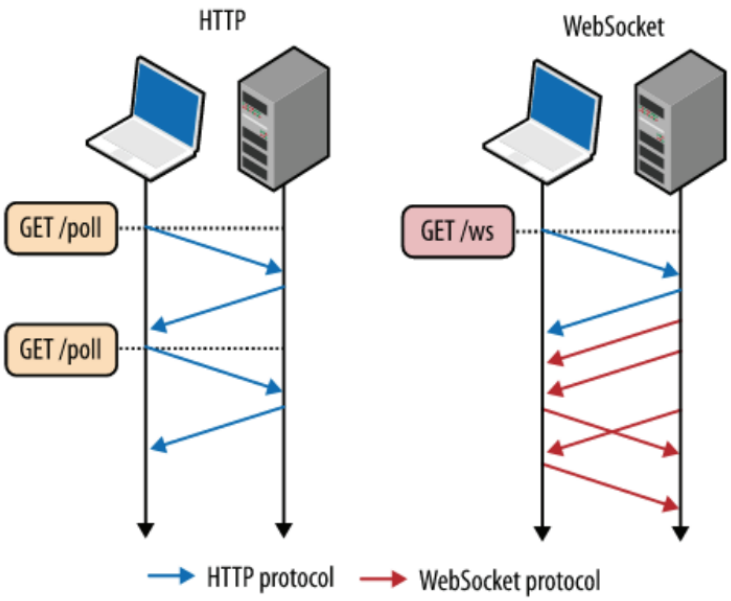

Figura 1. Fluxo de comunicação dos protocolos HTTP e WebSocket

- Linear – os itens são organizados de forma simples (e.g. matricial) e são destacados, um a um, da esquerda para a direita;
- Circular – os itens estão dispostos de forma circular e são destacados, um a um, no sentido horário;
- Por grupo – os itens são agrupados por alguma relação lógica ou espacial e destacam-se primeiramente os grupos e depois os itens do grupo destacado.

Figura 2. Tipos de Varredura

De acordo com Schirmer et al. [Schirmer et al. 2007], a seleção de um item usando a técnica de varredura pode ocorrer a partir de três modos de acesso:

- Varredura automática – inicia automaticamente e o marcador vai avançando, a partir de um tempo pré-configurado, até que o usuário ativa um acionador para selecionar o item desejado. Essa varredura automática exige menos movimentos do usuário;
- Varredura passo a passo – o usuário ativa o acionador repetidamente para mover o marcador até o item desejado e confirma a seleção do mesmo a partir de um

segundo acionador ou até um tempo predefinido seja alcançado. Essa varredura dá mais liberdade ao usuário, porém, exige mais desenvoltura do mesmo;
- Varredura inversa – o acionador deve ser mantido ativado até que o marcador selecione o item desejado. Essa varredura é mais indicada para indivíduos com movimentos muito tensos.

Os acionadores podem ser dispositivos de hardware ou software que podem ser ativados de diversas maneiras, por exemplo: pressão, tração, sopro ou sucção e contração muscular (cf. Figura 3). Para escolher um acionador, recomenda-se considerar os seguintes critérios propostos por Martins [Martins 2011]: a posição mais confortável para o usuário, o seu padrão de movimento mais consistente e voluntário e a melhor localização para facilitar a ativação do acionador.

Figura 3. Tipos de Acionadores

3. Plataforma Reaact

Esta seção apresenta parte da contribuição deste trabalho, ou seja, o software (i.e., jogo Caça Letras) e o hardware (i.e., Acionador Multitoque) que servem como prova de conceito da plataforma apresentada na Seção 4.

A Plataforma Reaact [REAACT 2021] surgiu da ideia de utilizar a aprendizagem significativa [Ausubel 1982] e a CAA para o ensino de vocabulário. A teoria da aprendizagem significativa de Ausubel [Ausubel 1982] diz que o aprendiz consegue se apropriar melhor dos conceitos quando a aprendizagem é baseada em conceitos previamente conhecidos por este. Assim, o Reaact utiliza o mesmo repertório (i.e., vocabulário) tanto para a prancha de comunicação, quanto para os jogos. Vale ressaltar que os jogos promovem o engajamento das crianças no aprendizado e que as aplicações baseadas em tablet aumentam o comportamento pró-social das crianças.

Atualmente, a plataforma conta com três ferramentas: a prancha de comunicação alternativa; um jogo de caça letras (cf. Figura 4) e um editor para a inserção de vocabulário. O jogo de Caça-Letras utiliza palavras do vocabulário do usuário e permite configurar características que podem aumentar o nível de dificuldade do jogo, são eles: 1) presença ou ausência de imagem; 2) presença ou ausência de texto; 3) somente vogais; 4) somente consoantes; e 5) ambos, vogais e consoantes. A partir do jogo é possível avaliar habilidades linguísticas como reconhecimento do símbolo que representa o objeto, das letras que compõe a palavra e do som associado a esta, bem como avaliar capacidades cognitivas como atenção e foco (e.g., o usuário se mantém atento? Consegue reconhecer a letra que falta?).

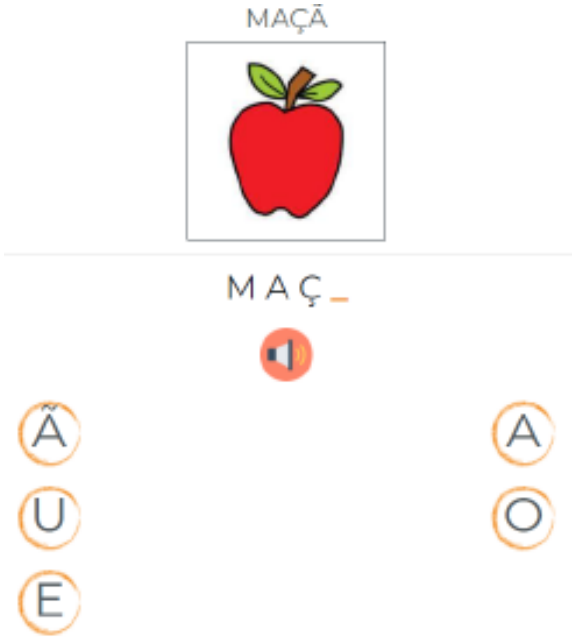

Figura 4. Jogo Caça Letras

Além do software, a Plataforma Reaact também possui integração com o hardware de um acionador multitoque que combina as vantagens de um acionador (i.e., não exige controle da motricidade fina) e de um varredor de tela (i.e., consegue acessar diversas áreas da tela). A Figura 5 apresenta o protótipo do acionador multitoque que consiste em 6 push-buttons coloridos dispostos linearmente em uma protoboard, juntamente com um micro-controlador ESP8266 que se conecta via Wi-Fi e é alimentada por um cabo USB. Os cinco primeiros botões são mapeados para áreas clicáveis do software e o último (i.e., voltar) é mapeado para repetir a fala do software.

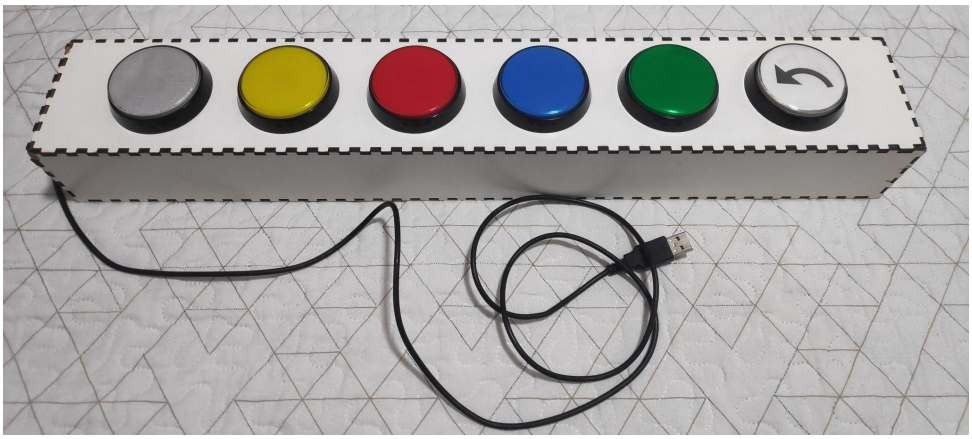

Figura 5. Acionador Multitoque

4. Plataforma IoT para CAA

Esta seção apresenta a segunda parte da contribuição deste trabalho, ou seja, a plataforma IoT para desenvolvimento de sistemas de CAA compostos por software e hardware que podem ser usados de maneira remota. A Figura 6 ilustra a visão geral desta proposta, a

qual é composta por quatro componentes, os quais são listados abaixo e detalhados nos próximos parágrafos:

- Software – representa o recurso de CAA. No escopo deste trabalho, será utilizado o jogo Caça-Letras apresentado na Seção 3;
- Hardware – representa o acionador externo ao recurso de CAA. No escopo deste trabalho, será utilizado o acionador multitoque apresentado na Seção 3;
- Servidor – representa o servidor WebSocket que serve como meio para a troca de mensagens entre Software e Hardware;
- Sala – representa o ambiente que contém toda a lógica para integração dos dispositivos Software e Hardware e que garante as condições necessárias para o funcionamento da plataforma.

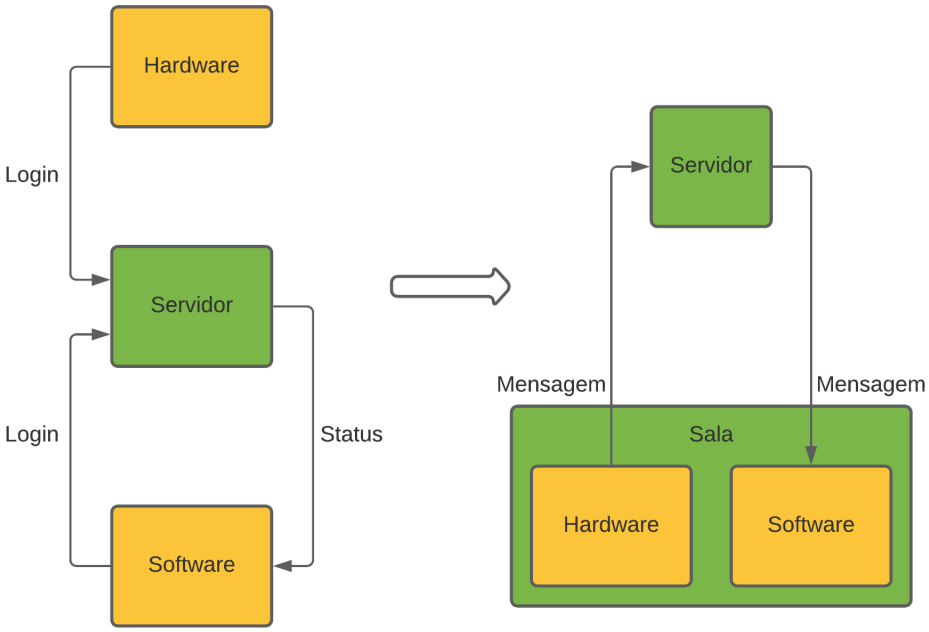

Figura 6. Visão Geral da Proposta

Conforme dito anteriormente, as condições de funcionamento da plataforma são garantidas pela Sala. Assim, para que uma Sala seja considerada funcional, esta precisa que, exatamente, 1 Hardware e 1 Software estejam conectados. Para isso, a Sala verifica periodicamente o status de conexão e garante que os dois componentes estão devidamente conectados e que podem se comunicar. Vale ressaltar que não é possível conectar 2 Hardwares ou 2 Softwares em uma mesma Sala.

O Servidor foi escrito em NodeJS, um framework Javascript para criação de servidores Web. Nele foram implementadas 2 camadas de comunicação, a WebSocket (WS) e a WebSocket Secure (WSS). De modo geral, *browsers* como o Google Chrome e o Mozilla Firefox tem como política de segurança que aplicações que rodam em uma camada de comunicação segura (e.g., HTTPS) não podem acessar camadas de comunicação menos seguras como o WS. Para lidar com essa política de segurança, faz-se necessária a implementação da camada WSS. Entretanto, alguns dispositivos de IoT não permitem a implementação dessa camada de segurança e só estabelecem conexão via protocolo WS. Para esses casos, faz-se necessário implementar os 2 protocolos para o servidor.

Para implementar as comunicações WS e WSS é necessário tratar os eventos *connection* e *close*. O evento *connection* ocorre quando um Cliente (i.e., o Software ou o Hardware) consegue estabelecer um canal de comunicação com o Servidor. A referência dessa conexão deve ser guardada para futuras comunicações entre Servidor e Cliente. O evento *close* ocorre quando a conexão é interrompida. Este não é um evento reativo e só é percebido quando o Servidor tenta executar alguma ação com o Cliente. No caso específico desta plataforma, o Servidor verifica se o Cliente está participando de alguma Sala, o remove e notifica a sua saída aos demais participantes.

De maneira geral, nesta plataforma, Cliente e Servidor podem trocar mensagens de 3 tipos, as quais são listadas a seguir e detalhadas nos próximos parágrafos:

- Login – tipo de mensagem que notifica o Servidor qual tipo de dispositivo está se conectando e para qual Sala deseja ir;
- Data – tipo de mensagem que notifica o Software o botão acionado no Hardware para que este processe a informação;
- Status – tipo de mensagem que notifica o Software sobre o status de conexão do Hardware e da Sala.

A Figura 7 ilustra um diagrama de sequência com os 3 tipos de mensagens trocadas entre Cliente e Servidor. Todas as mensagens são em forma de *string* onde os argumentos são separados por dois pontos (:), ou seja, na forma METODO:ARGUMENTO1:ARGUMENTO2. Dado o exposto, os argumentos podem ser qualquer *string* que não contenha dois pontos (:).

Antes do Login, Hardware e Software, precisam estabelecer uma conexão inicial com o Servidor para, só então, mandar a mensagem de Login na Sala. Esta mensagem segue o padrão LOGIN:DISPOSITIVO:SALA. Neste caso, DISPOSITIVO é um valor que pode ser HARDWARE ou SOFTWARE; e SALA é nome da sala que o DISPOSITIVO deseja se conectar.

A Sala é criada por meio de um dicionário onde a chave é o nome da Sala e o valor é um objeto composto pelos componentes Hardware e Software, os quais fazem referência ao *socket* criado após a conexão ao servidor. Caso não haja componentes ativos na Sala, o valor recebido será *null* ou *undefined*.

As mensagens de Status podem ser de três tipos: STATUS:NOT_ALLOWED, STATUS:HARDWARE_CONNECTED e STATUS:HARDWARE_DISCONNECTED. A primeira, ou seja, STATUS:NOT_ALLOWED é emitida sempre que um Software tente se conectar a uma Sala que já esteja ocupada por outro Software. Neste caso o acesso à Sala é negado. A segunda, ou seja, STATUS:HARDWARE_CONNECTED, comunica que existe um Hardware conectado e que a Sala está pronta para uso. Este evento é disparado logo após o Login de um Hardware ou quando um Software entra em uma Sala que já possua um Hardware conectado. Já a terceira, ou seja, STATUS:HARDWARE_DISCONNECTED, é disparada quando um o Software entra em uma Sala que não possui um Hardware conectado ou quando o Hardware se desconecta por algum motivo após a entrada de um Software na Sala.

Além das mensagens de Status descritas acima, foi implementado no Servidor um padrão de verificação de conexão que, a cada 10 segundos, envia uma mensagem de PING que deve ser respondida com um PONG. Caso não haja resposta, o Cliente

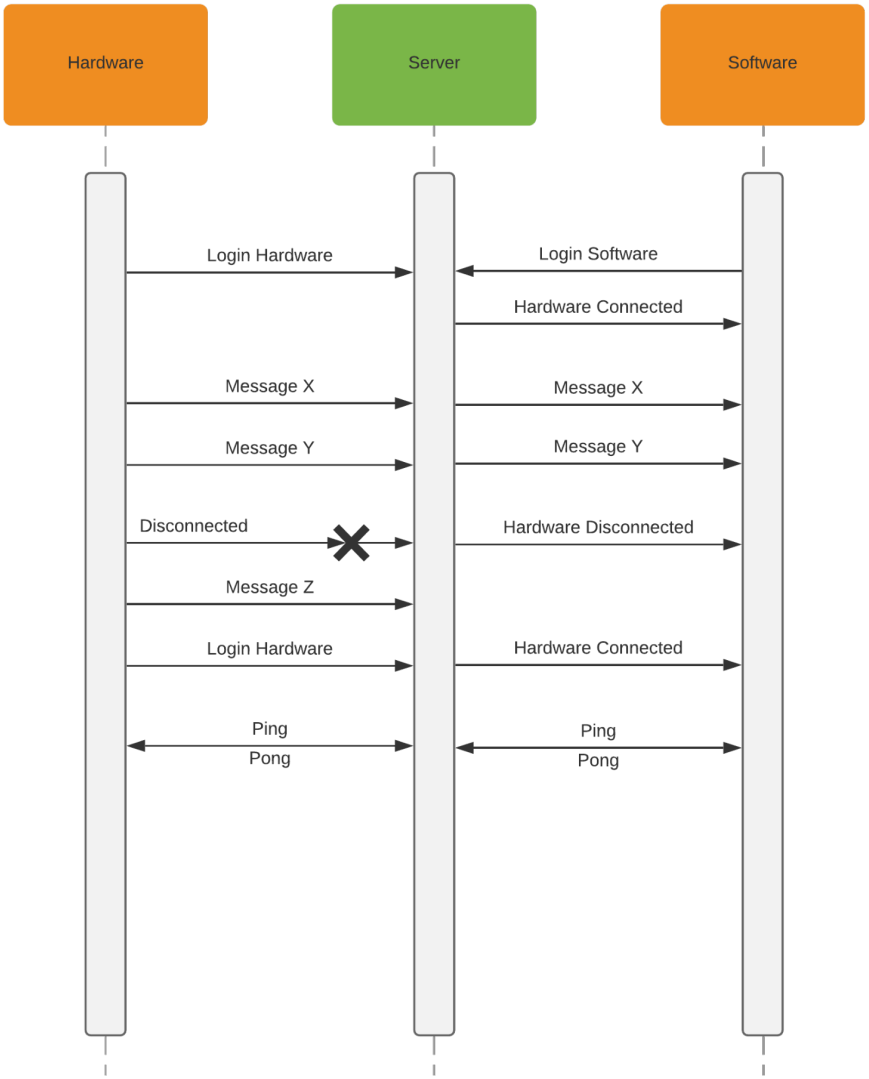

Figura 7. Diagrama de Sequência

que não respondeu será desconectado e removido da Sala. Esse padrão de verificação está disponível tanto no Hardware quanto no Software e algumas bibliotecas já o implementam por padrão.

A última classe de mensagem é a do tipo Data. Estas mensagens são geradas a partir do Hardware, são emitidas para o Software que ocupa a mesma Sala e seguem o padrão DATA:BUTTON. Neste caso BUTTON se refere ao botão acionado no Hardware e deve ser interpretado pelo Software que aciona a área da interface mapeada para aquele botão específico.

5. Prova de Conceito

Como prova de conceito, este trabalho vai usar os recursos do Reaact [REAACT 2021] que foram apresentados na Seção 3, ou seja, para o Hardware, será utilizado o acionador multitoque e, para o Software, o jogo Caça-Letras. Vale ressaltar que, apesar do acionador multitoque ter porta USB (cf. Figura 5), nenhum dado é enviado ou recebido por esta porta, a qual serve apenas como alimentação do dispositivo. Para facilitar o entendimento,

esta seção foi dividida em duas partes, a primeira para a integração do hardware e a segunda para a integração do software.

5.1. Integração com o Hardware

A integração do Hardware com o Servidor foi feita utilizando a Arduino *Integrated Development Environment* (IDE) [Arduino 2021], as bibliotecas ButtonDebounce [1] e WebSocketsClient [2] e a linguagem de programação utilizada foi C++ [Josuttis 2012]. A biblioteca ButtonDebounce lida com o controle de trepidação do botão e impede que este seja acionado diversas vezes em um curto espaço de tempo. Já a biblioteca WebSocketsClient é utilizada para conectar-se ao servidor utilizando o protocolo WebSocket.

O código do Hardware segue a sequência apresentada no pseudocódigo da Figura 8. Primeiro, conecta-se ao Wi-Fi e, logo em seguida, ao servidor. Vale ressaltar que tanto as credenciais da rede como a URL do servidor já foram previamente definidas no código do Hardware. Após a abertura desta conexão, o hardware envia a mensagem de login para se conectar a Sala REAACT (i.e., LOGIN:HARDWARE:REAACT). Com o login estabelecido, o Hardware entra em *loop* e fica apto a enviar o comando DATA:BUTTON com algum dos argumentos referentes aos botões disponíveis (i.e., BTN0, BTN1, BTN2, BTN3, BTN4 e BTN5) sempre que um deles seja pressionado. Caso nenhum botão seja pressionado ele fica em estado de *idle*.

```
1  INÍCIO
2      CONECTAR AO WIFI
3      CONECTAR AO SERVIDOR
4      ENVIAR COMANDO DE LOGIN (LOGIN:HARDWARE:REAACT)
5
6      REPETIR
7          SE BOTÃO PRESSIONADO ENTÃO
8              ENVIA COMANDO (DATA:BOTÃO)
9          SE NÃO
10             FAZ NADA
11     FIM REPETIR
12 FIM
```

Figura 8. Pseudocódigo do Hardware

Nota-se que a integração do Hardware é bem simples e que a complexidade fica do lado do Software que é responsável por implementar toda a lógica e máquina de estado da aplicação que for utilizar a Plataforma IoT.

5.2. Integração com o Software

A integração do Software com o Servidor foi feita utilizando a plataforma Angular [Angular 2021] e a biblioteca padrão para *browsers* WS [Docs 2021]. A conexão foi feita no protocolo WSS pois a aplicação hoje é acessada por HTTPS e, como citado anteriormente, *browsers* não permitem uma redução na segurança da conexão (cf. Seção 4).

[1]Disponível em: https://github.com/maykon/ButtonDebounce
[2]Disponível em: https://github.com/Links2004/arduinoWebSockets

```
INÍCIO
    CONECTAR AO SERVIDOR

    ESTADO DESCONECTADO:
        SE BOTAO LOGIN PRESSIONADO ENTÃO
            ENVIAR COMANDO DE LOGIN (LOGIN:SOFTWARE:REAACT)
            VAI PARA ESTADO CONECTADO
        FIM
        VAI PARA ESTADO DESCONECTADO

    ESTADO CONECTADO:
        REPETIR
            SE MENSAGEM RECEBIDA = STATUS:HARDWARE_DISCONNECTED ENTAO
                    VAI PARA ESTADO CONECTADO
            SE MENSAGEM RECEBIDA = STATUS:HARDWARE_CONNECTED ENTAO
                    VAI PARA ESTADO COM TECLADO

            ESTADO COM TECLADO:
                SE MENSAGEM RECEBIDA = DATA:BUTTON ENTAO
                    PROCESSA O COMANDO (BUTTON)
        FIM
FIM
```

Figura 9. Pseudocódigo do Software

O código do Software segue a sequência apresentada no pseudocódigo da Figura 9. Inicialmente, a implementação do Software se conecta ao servidor no instante que é aberto porém não faz Login ainda. O método de Login só é disparado quando o usuário informa no aplicativo, via botão nas configurações, que deseja utilizar o acionador multitoque. Em seguida, é criada uma máquina de estados que será modificada com o recebimento das mensagens do servidor. No exemplo da Figura 9 são implementados 2 estados referentes às respostas do Software às mensagens de STATUS:HARDWARE_CONNECTED e STATUS:HARDWARE_DISCONNECTED enviadas pelo Servidor. A primeira mensagem indica que a Sala está pronta para uso e o Software deve entrar em um estado para tratar os dados que o Hardware enviar. Já a segunda mensagem indica que o Hardware não se encontra na Sala e não haverá comunicação entre as partes (i.e., Software e Hardware). No primeiro caso temos uma Sala funcional e cada Software deverá implementar sua própria lógica de uso. No caso do Caça-Letras foi feito um mapeamento de cada um dos botões para cada uma das respostas garantindo, assim, o acionamento direto à área desejada.

A Figura 10 apresenta a interface do jogo quando este está desconectado do acionador multitoque, enquanto que a Figura 11 apresenta a interface do jogo quando este está conectado ao acionador multitoque. A parte A de cada figura mostra a tela de configuração e, a parte B, a tela do jogo. É possível notar na Figura 11-B que a interface se modifica para que cada opção clicável seja acessada por uma cor do acionador multitoque. Assim, o usuário tem a indicação visual de qual área foi mapeada para cada botão. Isso melhora a experiência de usuários com limitação motora, uma vez que estes podem, em poucos cliques, acionar a área desejada do Jogo, não sendo necessário esperar vários segundos para selecionar um elemento como ocorre com sistemas de varredura de tela.

Caso não seja possível fazer o mapeamento 1:1, ou seja, 1 botão do acionador correspondendo a 1 elemento clicável no software, deve-se fazer a divisão em subgrupos. A ideia é que cada subgrupo corresponda a um conjunto de elementos ou outros subgrupos de, no máximo, o número de botões clicáveis do acionador (i.e., no caso do

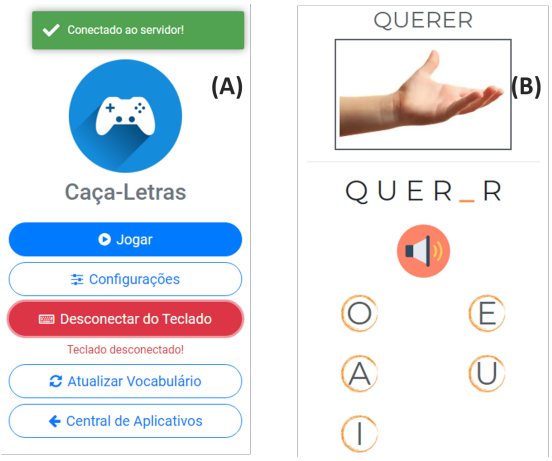

Figura 10. Recurso de CAA sem conexão com o acionador multitoque

Figura 11. Recurso de CAA conectado ao acionador multitoque

acionador multitoque deste trabalho, 5 botões). Assim, se o Jogo Caça-Letras tivesse, por exemplo, 25 opções de resposta, inicialmente, cada botão seria mapeado para um subconjunto de 5 elementos clicáveis. Depois da primeira seleção (i.e., do subconjunto), os 5 elementos do subconjunto escolhido seriam mapeados para os 5 botões clicáveis. Desta forma, no exemplo apresentado, a seleção da resposta entre 25 possibilidades se daria em duas etapas, onde a primeira seria para o subgrupo e a segunda para o elemento. A Equação 1 apresenta a generalização deste raciocínio, onde: $AreasClicaveis$ indica a quantidade de áreas clicáveis no software; $NBotoes$, a quantidade de botões do acionador; e $QuantidadeCliques$, a quantidade de cliques necessários. Note que a subtração de uma unidade em $NBotoes$ se dá porque o último botão do acionador será utilizado para desfazer a última seleção feita, caso $QuantidadeCliques$ seja maior que um.

$$AreasClicaveis = (NBotoes - 1)^{QuantidadeCliques} \quad (1)$$

Com relação à máquina de estado do Software, esta se manteve simples pois houve o mapeamento de 1:1 dos botões para as opções do Caça-Letras. Além disso, o Caça-Letras não precisa guardar nenhum estado entre as fases, o que simplifica ainda mais o processo. Caso mais cliques fossem necessários, a máquina de estado teria que ser estendida para mais combinações. Da mesma forma, se as opções fossem alteradas de acordo com as respostas anteriores (e.g., progressão em fases do jogo), a máquina de estados também seria mais complexa. A escolha do Caça-Letras foi feita visando diminuir a complexidade da máquina de estado, pois isso traria o benefício da validação rápida do protótipo.

6. Conclusão

Usuários de Comunicação Aumentativa e Alternativa (CAA) com limitação motora são beneficiados pelo uso de acionadores e varredores de tela ou, de maneira alternativa, por acionadores multitoque que permitem a seleção direta de áreas específicas da tela. Atualmente, os recursos de hardware disponíveis se conectam aos softwares por meio de conexões do tipo USB ou Bluetooth. Essas conexões exigem a proximidade física entre

acionador e aplicativo, restringindo as possibilidades de uso a um mesmo acionador e a mesma localização geográfica.

O atual contexto pandêmico da COVID-19 popularizou o atendimento remoto e fomentou o desenvolvimento de tecnologias que possam ser usadas para este fim. Neste contexto, este trabalho propôs uma plataforma, baseada em IoT, que conecta recursos de CAA com um acionador multitoque e que permite o uso remoto.

Este trabalho apresenta duas grandes contribuições: os recursos da Plataforma Reaact, isto é, o jogo Caça-Letras e o acionador multitoque; e a Plataforma IoT. Juntas, estas contribuições configuram um recurso de CAA e de ensino que pode ser utilizado em terapias remotas. Desta forma, o acionador multitoque pode estar na casa do paciente que interage, em tempo real, com seu terapeuta que pode estar distante geograficamente.

Apesar da contribuição, este trabalho ainda apresenta a limitação do hardware do acionador multitoque. Seria necessário produzir diversos acionadores para serem enviados aos pacientes. Em um trabalho futuro, pretende-se alterar a lógica da plataforma para que esta conecte o Software do recurso de CAA a um dispositivo acionador, que pode ser tanto um hardware quanto outro software (e.g., um aplicativo que possa ser baixado pelo usuário). Essa alteração na lógica da plataforma aumentaria sua escalabilidade e os benefícios desta solução. Além disso, pretende-se inserir metadados para dar suporte a outras funcionalidades. O uso de metadados pode ser adicionado aos eventos de login para que o outro dispositivo envolvido possa se beneficiar dessas informações, por exemplo, a quantidade de botões do acionador pode disparar um evento para que a interface do software se adapte a esta informação. Outro exemplo seria o uso dos metadados da sala (e.g., quanto tempo a criança brincou em um determinado jogo ou quantos acertos seguidos) podem ser compartilhadas com outros dispositivos de IoT como, por exemplo, um relógio inteligente ou o próprio acionador, o qual pode se utilizar de luzes coloridas que mudam de cor de acordo com as respostas da criança, incentivando o engajamento da mesma.

Referências

Angular (2021). Angular The modern Web developer's platform. https://angular.io. Accessed: 2021-06-01.

Arduino (2021). Arduino IDE 1.8.15. https://www.arduino.cc/en/software. Accessed: 2021-06-01.

Ashton, K. et al. (2009). That 'internet of things' thing. *RFID journal*, 22(7):97–114.

Ausubel, D. P. (1982). A aprendizagem significativa. *São Paulo: Moraes*.

Bersch, R. (2008). Introdução à tecnologia assistiva. *Porto Alegre: CEDI*, 21.

Carrion, P. and Quaresma, M. (2019). Internet da coisas (iot): Definições e aplicabilidade aos usuários finais. *Human Factors in Design*, 8(15):049–066.

Docs, M. W. (2021). WebSockets Documentation. https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API#documentation. Accessed: 2021-06-01.

FETTE, I. (2021). The websocket protocol. https://tools.ietf.org/html/rfc6455. Accessed: 2021-06-01.

Josuttis, N. M. (2012). The c++ standard library: a tutorial and reference.

Lima, T., Silva, E., Lima, A., Franco, N., and Fidalgo, R. (2017). aboard: uma plataforma computacional na nuvem para comunicação alternativa e educação inclusiva. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 6, page 102.

LUBBERS, P. and GRECO, F. (2021). HTML5 WebSocket: A Quantum Leap in Scalability for the Web. http://www.websocket.org/quantum.html. Accessed: 2021-06-01.

Martins, D. S. (2011). *Design de recursos e estratégias em tecnologia assistiva para acessibilidade ao computador e à comunicação alternativa*. Msc dissertation, Universidade Federal do Rio Grande do Sul.

Oliveira, G. M., Costa, D. C., Cavalcanti, R. J., Oliveira, J. P., Silva, D. R., Nogueira, M. B., and Rodrigues, M. C. (2018). Comparison between mqtt and websocket protocols for iot applications using esp8266. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 236–241. IEEE.

REAACT (2021). Reaact Comunicação Aumentativa e Alternativa. https://reaact.com.br. Accessed: 2021-06-01.

Reis, M. d. L. d. B. (2017). *Uma plataforma Arduino para construir acionadores para Sistemas de Comunicação Aumentativa e Alternativa*. Undergraduate thesis, Universidade Federal de Pernambuco.

Schirmer, C. R., Browning, N., Bersch, R., and Machado, R. (2007). Atendimento educacional especializado–deficiência física. *São Paulo: MEC/SEESP*, 1:130.

Tetzchner, EV e Martinsen, H. (1992). *Augmentative and Alternative Communication. IN.: Sign teaching & the use of communication aids*. Whurr Publishers, London.

Thinger (2021). Thinger.io – Open Source IoT Platform. https://thinger.io/. Accessed: 2021-08-29.

ThingsBoard (2021). ThingsBoard Open-source IoT Platform. https://thingsboard.io/. Accessed: 2021-08-29.