CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
25 a 29 de novembro de 2019

CONTROLE REMOTO DE MÚSICAS NO SMARTPHONE VIA BLUETOOTH ENTRE APLICATIVO ANDROID E ARDUINO

Júlia Tannús de Souza*[1]

[1]FEELT – Universidade Federal de Uberlândia

Resumo **- Este artigo descreve a construção de um controle remoto baseado no microcontrolador Arduino e que conta também com botões, potenciômetro, *display* LCD e Bluetooth. Seu propósito é controlar um aplicativo Android de músicas que estejam no *smartphone* e que também foi desenvolvido neste trabalho. Atualmente, é possível utilizar o celular como controle remoto de televisões, caixas de som e outros dispositivos que possuam Bluetooth. Porém, não foram encontradas propostas que usam o próprio celular como fonte de som e Arduino como controle remoto. Isto é vantajoso, pois um celular moderno possui bons autofalantes, é um item que as pessoas sempre levam consigo e, portanto, ele possibilita ouvir música em qualquer lugar sem custo adicional de uma caixa de som. Neste sentido, este projeto difere das soluções apresentadas no mercado. O sistema final provou ser funcional e cumprir sua proposta.**

Palavras-Chave **- Aplicativo Android, Comunicação Bluetooth, Controle Remoto, Microcontrolador Arduino**

SMARTPHONE MUSIC REMOTE CONTROL VIA BLUETOOTH COMMUNICATION BETWEEN ANDROID APPLICATION AND ARDUINO

Abstract **- This article describes the construction of a remote control based on the Arduino microcontroller that also has buttons, potentiometer, LCD display and Bluetooth. Its purpose is to control an Android application of music that is on the smartphone that was also developed in this work. Currently, is possible to use the smartphone as a remote control for televisions, speakers, and other Bluetooth enabled devices. However, no proposals were found that use the cell phone itself as a sound source and Arduino as a remote control. This is advantageous because a modern mobile phone has good speakers, it is an item that people always carry with them, so it makes it possible to listen to music anywhere at no extra cost from a speaker. In this sense, this project differs from the solutions presented in the market. The final system proved to be functional and fulfill its proposal.**

*julia.tannus95@gmail.com

Keywords **- Android Application, Arduino Microcontroller, Bluetooth Communication, Remote Control**

I. INTRODUÇÃO

Em 1994, Ericsson criou a ideia do Bluetooth. O nome foi tirado do rei Viking Harald Bluetooth, que uniu a Noruega e a Dinamarca [1]. A visão por trás dele era criar uma interface de rádio de baixo consumo de energia e baixo custo.

Um dos principais aspectos do Bluetooth é que ele elimina os cabos e interconecta os dispositivos de computação e comunicação. Uma maneira de descrever a ideia por trás disso é que o espaço de conectividade pessoal é como uma bolha de comunicação. Ele acompanha as pessoas e permite que elas se conectem a outros dispositivos que entram na bolha. Não é tão limitado quanto a tecnologia de infravermelho, onde você precisa de uma linha de visão para se conectar. Outra característica importante é que oferece conectividade *ad hoc* entre dispositivos pessoais. Isso possibilita formar grupos sem a necessidade de qualquer infraestrutura [2].

As contribuições mais estreitamente relacionadas ao trabalho aqui apresentado são as que foram feitas no uso de dispositivos Bluetooth para controlar aplicativos de computador. Um exemplo é o trabalho em [3], que mostra uma possível implementação de um controle remoto Bluetooth. Eles criaram um sistema de controle remoto universal. Um de seus principais objetivos era controlar dispositivos de consumo através da Internet.

Ademais, existem alguns exemplos de aplicativos de controle remoto Bluetooth já existentes no mercado, disponíveis em lojas de aplicativo [4]. Outro produto facilmente encontrado para venda é o controle remoto de aparelhos de som que possuam comunicação Bluetooth.

Porém, ao pesquisar-se trabalhos e produtos relacionados, não foram encontradas propostas similares a aqui apresentada, ou seja, a de um controle remoto com base em Arduino para controlar um celular. Neste caso, o celular não é o controle remoto, mas sim é controlado remotamente pelo Arduino.

Um celular moderno atual possui bons autofalantes, é um item que as pessoas sempre levam consigo e, portanto, ele possibilita ouvir música em qualquer lugar sem custo adicional de uma caixa de som. Além disso, a principal vantagem do celular é o aplicativo de música. Ele oferece uma melhor organização das músicas, possibilita encontrar as músicas mais rápido, pode classificá-las por álbum, artista,

ano, selecionar a partir de qual segundo o usuário deseja escutar e salva os dados das músicas escutadas, podendo criar listas de reprodução com as músicas preferidas, as mais ouvidas, as adicionadas recentemente, etc.

Outra vantagem é a versatilidade de um aplicativo de celular: poderia, por exemplo, neste trabalho, também ser implementada a funcionalidade de listar e acessar arquivos de vídeo. Assim, o *smartphone* reproduziria tanto vídeos quanto músicas remotamente, funcionando como uma estação multimídia.

Portanto, o controle baseado em Arduino e aplicativo de música aqui apresentados podem ser utilizados em qualquer ambiente para controlar som à distância. Neste sentido, ele difere das soluções apresentadas no mercado. Ademais, o projeto provou ser funcional e cumprir sua proposta.

II. MATERIAIS E MÉTODOS

Nesta seção, serão descritos o projeto, componentes utilizados, construção e programação para implementar o controle remoto e aplicativo de celular.

A. Projeto

Projetou-se o controle para funcionar de acordo com os passos seguintes:

1) Baixa-se o aplicativo desenvolvido.

2) Configura-se o aplicativo, indicando para ele onde estão suas músicas (digitando o endereço interno, por exemplo, storage/0F19-C7FD, que indica o endereço do cartão SD neste celular utilizado).

3) O aplicativo fará uma lista de suas músicas.

4) Após isto, será possível interagir com suas músicas tocando em qualquer uma da lista e usando os botões da interface. É possível dar Play/Pause, Next (próxima música da lista), Prev (música anterior). O botão configuração permite trocar o endereço de suas músicas. O botão "atualizar" atualiza a lista. Também é possível pesquisar músicas pelo nome.

5) Conecta-se com o módulo bluetooth do Arduino por meio do botão "Conectar".

6) Agora, também é possível controlar a reprodução das músicas no celular através do Arduino:

6.1) Os botões push do Arduino têm as funções de Play/Pause, Next e Prev.

6.2) Por meio do potenciômetro, é possível controlar o volume da música.

6.3) O nome da música que atualmente está tocando aparece no *display* LCD.

B. Lista de materiais

O controle remoto é composto dos seguintes elementos:

1. 1 Microcontrolador Arduino UNO
2. 6 resistores de 300Ω
3. 3 push buttons
4. 1 potênciômetro de 10K
5. 1 LCD do tipo Hitachi hd44780 (16x2)
6. 1 módulo Bluetooth hc-05

Todos os componentes foram de fácil aquisição em lojas de eletrônica locais.

C. Diagrama esquemático do hardware

A seguir, são apresentados o diagrama esquemático do hardware utilizado, com as ligações correspondentes entre os *chips* (Figura 1) e o diagrama visão Protoboard (Figura 2), com o desenho dos componentes e suas ligações.

Figura 1: Diagrama esquemático do hardware

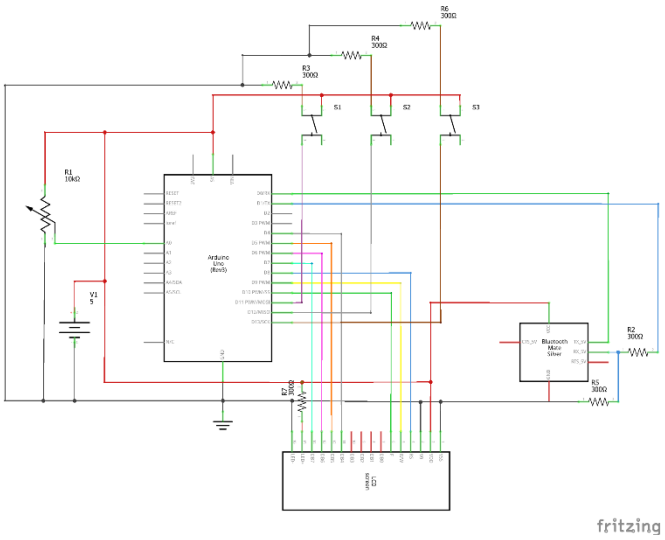
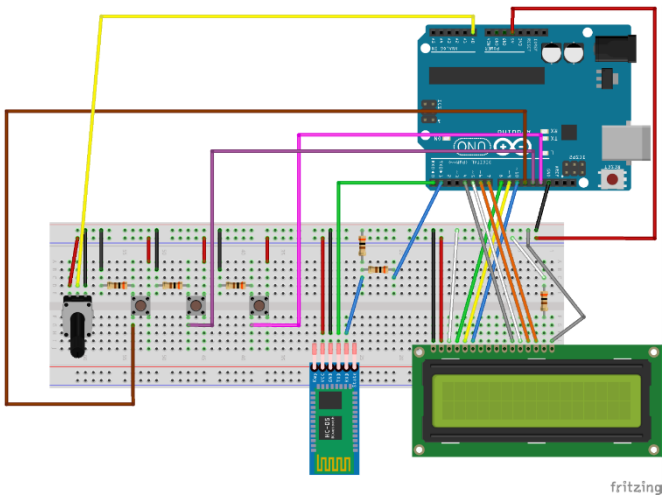


Figura 2: Diagrama esquemático - visão Protoboard



D. Montagem dos componentes

1) *Botões*

Primeiro, foram utilizados 3 *push buttons*. Foi determinado que o que está ligado ao pino 11 do Arduino teria a função "Play/Pause". O do pino 12 teria a função "Next". O do pino 13 teria a função "Prev".

Para cada botão, foi necessário conectar três fios à Protoboard. Conectou-se uma perna vertical do botão na alimentação +5V. Na outra perna, conectou-se um fio até o pino digital escolhido no Arduino. O outro lado da mesma perna foi conectada por meio de um resistor pull-down (300Ω) ao GND. Este resistor adiciona uma resistência ao fio que liga ao GND, para que, quando a chave seja fechada, a corrente "prefira" passar pelo fio de menor resistência, que

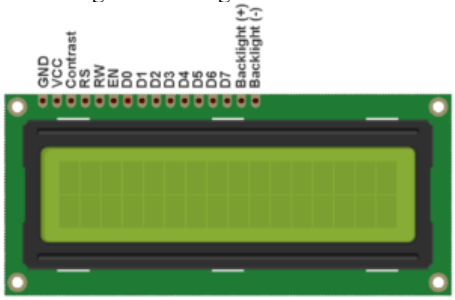
é o que liga ao +5V, indicando ao Arduino que o botão foi pressionado.
Quando o botão *push* não está sendo apertado (aberto), não há conexão entre as suas duas pernas, então o pino digital é conectado ao GND e lemos uma tensão LOW. Quando o botão é pressionado, o pino digital é conectado ao +5V, e lemos HIGH.

2) *Potenciômetro*
O potenciômetro funciona como controle de volume neste projeto. Para isso, nele é feita uma conversão A/D.
Seus pinos das laterais foram ligados em +5V e GND. O pino central foi ligado no pino analógico A0 do Arduino. No potenciômetro existe um cursor, que vai percorrendo o resistor de uma ponta até a outra, variando a resistência de 0 ao valor máximo (10KΩ), dividindo a tensão, que sairá no pino central e será lida pelo Arduino, que posteriormente converterá este valor em um valor de 0 a 1023.

3) *Módulo Bluetooth*
Precisa-se conectar 4 pinos do módulo Bluetooth. O pino VCC em +5V, o pino GND no terra do Arduino. O pino TXD transfere dados e deve ser ligado no pino RXD do Arduino, que recebe os dados.
O nível lógico dos pinos RX e TX é de 3.3V, o que significa que, para o Arduino Uno, por exemplo, é necessário um divisor de tensão no pino RX para evitar que o módulo seja danificado. Isso é necessário pois o Arduino Uno trabalha com nível de sinal de 5V. Por isso, adicionaram-se dois resistores de 300Ω.

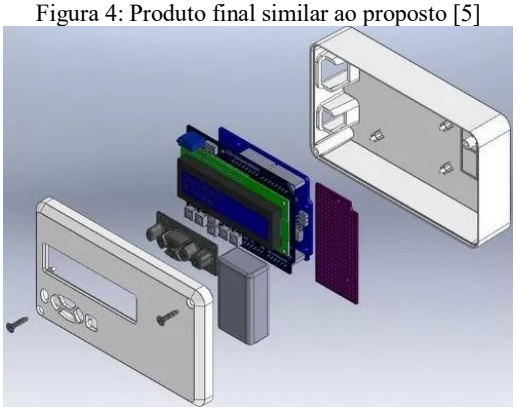
3) LCD

Figura 3: "Pinagem" do LCD



O LCD 16x2 possui 16 pinos. O pino 1 é ligado ao GND. O pino 2, ao +5V. O pino 3 (ajuste de contraste da tela) foi ligado ao GND.
Os pinos 4, 5 e 6 são usados para variáveis de controle do LCD. São elas: RS (register select), RW (read/write) e EN (enable), respectivamente. Escolheu-se ligar em 8, 9 e 10.
Os pinos 7, 8, 9, 10 não são conectados. Os pinos 11, 12, 13, 14 são conectados nas portas 4, 5, 6 e 7 do Arduino. O LCD envia e recebe dados de 8 bits, pois são caracteres. Porém, não restaram 8 portas digitais no Arduino Uno. O LCD possui a possibilidade de configurar se queremos enviar e receber dados de 4 em 4 bits ou mandar 8 bits diretamente. Configuramos para mandar os 8 bits (char) 4 bits por vez, em 2 vezes. Ou seja, é um dado de 8 bits mandado e recebido de forma "parcelada", para poupar portas digitais do Arduino.
Escolheu-se ligar os pinos em sequência no Arduino (4, 5, 6 e 7) pois facilita o código. Ao invés de mandar cada *bit* da variável *char* pino por pino, podemos mandar uma variável *char* diretamente ao registrador, já que os pinos estão em sequência.
No mais, os pinos 15 e 16 são para ajustar a luz de fundo (backlight). Para isto, no pino 15 conectamos um resistor de 300Ω ao +5V. O pino 16 vai direto ao GND.
Posteriormente, para o produto final, seria feito um esquema PCB, uma placa de circuito impresso que integrasse os componentes e seria adicionada uma bateria de alimentação ao Arduino.
O protótipo do produto final seria similar ao seguinte:

Figura 4: Produto final similar ao proposto [5]



E. Programação do software Arduino

Primeiramente, foi feita uma biblioteca para o LCD inspirada na biblioteca para LCD "LiquidCrystal", do Arduino. Isto porque foi utilizada a linguagem ANSI C, somente, sem usar nenhuma função pronta.

No início do código, algumas definições para facilitar o entendimento do mesmo. O registrador PORTD configura os pinos como LOW ou HIGH. As portas D são os pinos 0 a 7 no Arduino. Nas portas 4, 5, 6, 7, há a entrada e saída de dados de 8 bits, enviados em 4 em 4 bits. Por isto, este registrador foi denominado "DataBus".

O registrador PORTB (HIGH/LOW) controla as portas do tipo B (8 a 13). Nas portas 8, 9 e 10 foram configuradas as variáveis de controle: RS, RW e EN. Por isto, esse registrador foi denominado "ControlBus".

O registrador DDRD configura os pinos como input ou output. Ele determina se o Arduino está recebendo ou mandando um caractere. Ele foi denominado "DataDir_DataBus", pois determina a direção dos dados (Arduino->LCD ou LCD->Arduino).

A mesma coisa ocorre com o registrador DDRB para os pinos de controle. Foi denominado "DataDir_ControlBus".

Foi escolhido que o pino RS fosse ligado na porta 8, que, para o registrador PORTB, é a porta 0. O pino RW na porta 9 (1) e o pino EN na porta 10 (2). Tanto o registrador das variáveis de controle quanto os pinos podem ser alterados neste cabeçalho, verificando sempre o que o código possibilita.

O LCD possui várias configurações a serem feitas na inicialização. As configurações são feitas em código hexadecimal de 8 bits. Por isso, foram definidas *flags* para

traduzir os códigos hexadecimais para serem localizados mais facilmente. Os valores hexadecimais foram encontrados no manual do LCD Hitachi hd44780 e no código-fonte da biblioteca "LiquidCrystal" do Arduino.

Agora, começam as funções principais. O LCD possui 3 pinos que mandam e recebem comandos:

- RS é o seletor de registro (Register Select). Caso seja ajustado em 0, significa que será enviado um comando (ajustes, configurações, mudar a posição do cursor, limpar o display...). Caso 1, indica-se que será enviado um caractere (as palavras que serão escritas no display).
- RW é o seletor de *read/write*: 0 = escreve para o LCD; 1 = lê do LCD.
- EN é o sinal *enable*. Quando manda-se um pulso High-Low (mandar 1, dar delay, mandar 0), manda-se o LCD ler o comando ou caractere que foi enviado. De forma simplificada, é como se o LCD estivesse "no escuro" e não conseguisse ler o comando que foi enviado. Ao dar um pulso High-Low ativando o Enable, é como se fosse momentaneamente acesa a luz, para que o comando possa ser lido e executado.

Por isto, foi feita uma função "Pulse_Enable()". Ela é chamada toda vez que se queira que um comando seja executado. Ela coloca EN como 0 (LOW), dá um *delay*, coloca como 1 (HIGH), dá um *delay* e LOW novamente.

A função "Send_A_Command" manda um comando ao LCD. Primeiramente, ela recebe o comando com seus 8 bits originais (comandos definidos nas flags acima) e prepara o ControlBus para receber os dados colocando RS e RW como 0 (enviar comando - escrever para o LCD).

Como as portas 11, 12, 13 e 14 do LCD, que são (DB4, DB5, DB6 e DB7) estão conectadas aos pinos 4, 5, 6 e 7 (PD4, PD5, PD6 e PD7), pode-se simplesmente fazer DataBus = value, já que os números das portas são correspondentes. O LCD guardará os 4 últimos bits do comando.

A seguir, manda-se um pulso ao Enable para ler a primeira metade. Manda-se a segunda metade do comando ao LCD. Desta vez, deseja-se mandar os 4 primeiros bits do comando. Por isso, faze-se DataBus = (value << 4) e pulsa-se o Enable.

A função "Send_A_Character" manda um caractere para o display LCD. Primeiramente, põe-se RW como 0 (escreve para o LCD) e RS como 1 (mandar um caractere). Repetem-se os passos descritos na função anterior.

A função "Send_A_String" recebe uma string e manda um caractere por vez para o LCD.

Não menos importante, foi necessária uma função para inicializar o LCD. Esta função foi chamada de "LCD_Init". Primeiramente, seta-se os pinos de controle como OUTPUT e LOW e os pinos de dados como OUTPUT. Isto é necessário para começar a enviar comandos. De acordo com o manual do fabricante, é necessário mandar o comando de código 0x03 três vezes, com delay, e depois o comando 0x02 para mudar para o modo 4 bits (mandaremos 8 bits de 4 em 4 bits). Agora, pode-se configurar o LCD. Manda-se os comandos de modo 4 bits, escrever em 2 linhas, tamanho da letra (5x8), liga-se o display, desliga-se o cursor e manda-se o cursor parar de piscar. Limpa-se a tela e configura-se o modo de entrada.

Foram também feitas funções idênticas às da "LiquidCrystal". As principais são: "LCD_Clear" limpa a tela, "LCD_ReturnHome()" coloca o cursor na posição zero e "LCD_ScrollDisplayLeft()" roda a frase do *display* para a esquerda, para ler frases longas.

Na função principal, incluiu-se a biblioteca LCD. Foram feitas funções para usar a comunicação via UART, que é o meio pelo qual a comunicação Bluetooth se dá. Nelas, foram configuradas a velocidade de transferência, acionada a transferência de dados e definidas as funções de enviar e receber caracteres.

Após, foi configurado o conversor Analógico/Digital (AD) do potenciômetro, que funciona como controle de volume e determinou-se sua porta como sendo A0.

Posteriormente, foi feita uma função chamada "EscreveLCD", que escreve no *display* o nome da música que está tocando. O aplicativo irá enviar o nome da música caractere por caractere via UART. O Arduino receberá através da função Uart_getchar() e guardará os caracteres em uma matriz. Quando o caractere for igual a '3' isso significa que a mensagem terminou de ser mandada. Esse valor foi definido no código Android como MESSAGE_READ = 3.

A seguir, o LCD é limpo e é escrito a *string* recebida nele. O *display* então dá um scroll para a esquerda para que toda a frase seja lida.

Na função principal, primeiramente, Bluetooth, potenciômetro e LCD são iniciados.

Posteriormente, o valor do AD é lido e transformado em um caractere de 'A' a 'P'. É necessário transformar em um caractere pois precisa enviar este valor para o celular via UART e esta só recebe caracteres. A seguir, este trecho do código:

```
while(1)
{
  u16_ad = ADCL;
  u16_ad = (ADCH << 8) + u16_ad;

  if(u16_ad<=10)
  {
   c_SomNovo = 'A';
  }
  if(u16_ad>=60 && u16_ad<=134)
  {
   c_SomNovo = 'B';
  }
...
  if(u16_ad >= 1001){
   c_SomNovo = 'P';
  }
```

Os celulares geralmente vêm com 15 níveis de volume. Por isso, são 16 caracteres (contando o 0). Porém, esta função também funciona com celulares com qualquer níveis de volume. Veja a função que trata os caracteres recebidos do AD no Android:

```
for(int i = 0; i <= 15; i++){
   if(codigo.contains(Character.toString ((char)
(65+i)))){ //if(codigo.contains("A" -- "P"))
```

```
            volumeSeekBar.setProgress((max/15)*i);

PlayerAudioManager.setStreamVolume(AudioManager.ST
REAM_MUSIC, (max/15)*i, 0);
        }
    }
```

Observe que esta função gerará níveis de volume entre máximo e mínimo para qualquer tipo de celular.

```
if(c_SomNovo != c_SomAntigo)
{
 Uart_putchar(c_SomNovo);
 c_SomAntigo = c_SomNovo;
}
```

Esta condição é para mandar um caractere representando um nível de volume somente se este for alterado (para não ficar continuamente mandando o caractere). Finalmente, o código a seguir resolve o que será feito quando o usuário apertar um botão. O LCD será limpo, o nome do botão apertado aparecerá na tela e será armazenado e a função "EscreveLCD" será chamada, mostrando o nome da música no display.

```
if (PINB & (1 << PINB3))   // pino 11
{
   LCD_Clear();
   _delay_us(2000);
   Send_A_String("Pause");
   c_BotaoNovo = 'X';//Play/Pause
   _delay_ms(500);
   if(c_BotaoNovo != c_BotaoAntigo)
   {
    Uart_putchar(c_BotaoNovo);
   }
   EscreveLCD();
}
```

F. Programação do aplicativo Android

Foi necessário criar um aplicativo Android de gerenciamento de músicas próprio. Isto porque seria necessário editar o código fonte do aplicativo para que ele possa entender comandos do Arduino. Ele foi feito no software Android Studio [7].

O aplicativo gerenciador de músicas pergunta ao usuário em qual pasta estão suas músicas. Inserido o caminho, o aplicativo faz uma lista com o nome de todos os arquivos com a extensão ".mp3" que ele encontrou na pasta especificada. Ele então mostra essa lista na interface.

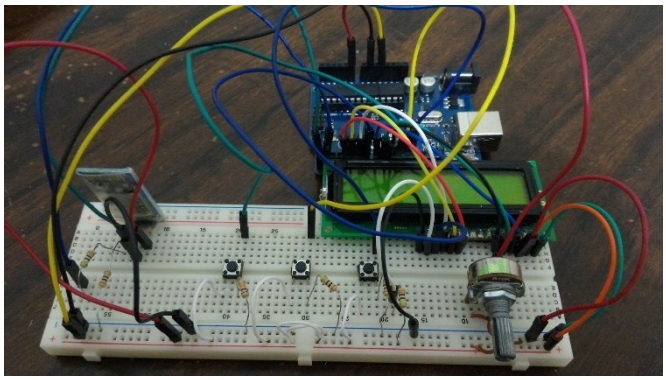
Após isso, o usuário seleciona a música que deseja através de um toque em cima do nome do arquivo, na lista. Ele agora pode interagir com as músicas através de botões na interface com os seguintes comandos: play, pause, próximo e anterior.

III. RESULTADOS E DISCUSSÃO

A. Controle remoto

O controle foi montado conforme Figura 5 e funcionou corretamente. Ele consegue mostrar o nome da música no display sem muitos problemas, somente demora alguns segundos, pois tanto o envio Bluetooth quanto o próprio LCD são relativamente lentos.

Figura 5: Montagem do hardware



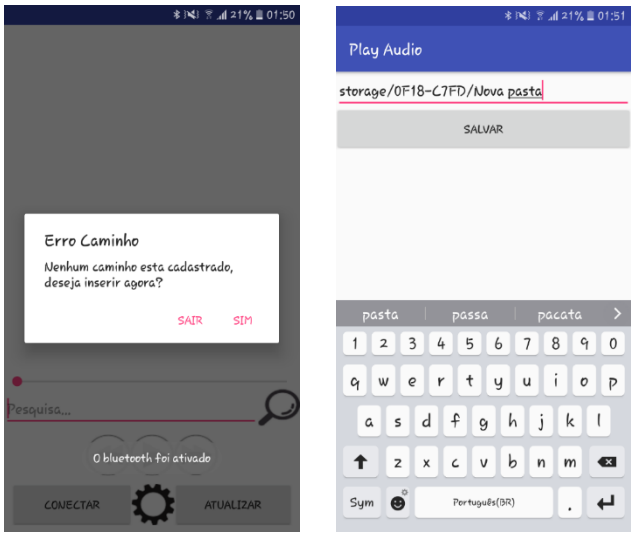
B. Aplicativo de celular

Para instalar o aplicativo, primeiramente, deve-se habilitar a opção "Fontes desconhecidas" em Configurações -> Segurança. Após isto, basta abrir o arquivo ".apk" e apertar "Instalar". Em seguida, "Abrir".

Ao abrir o aplicativo, o mesmo irá solicitar permissão para ativar o bluetooth, caso esteja desativado.

Agora, o aplicativo irá solicitar para o usuário cadastrar um caminho (Figura 6.a). Na nova tela, deve-se inserir o caminho da pasta de músicas (Figura 6.b).

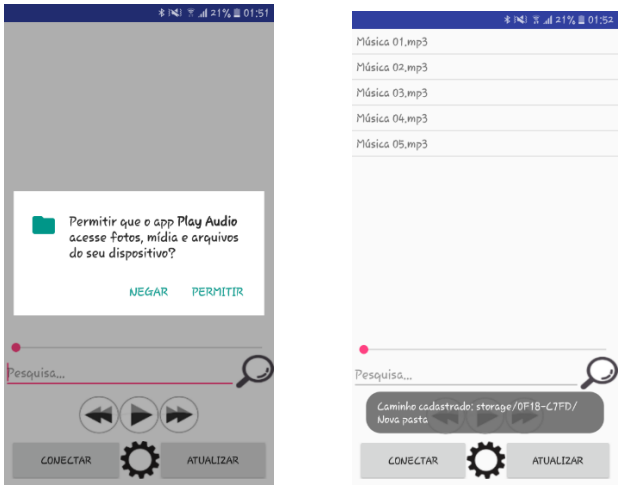
Figuras 6 a) e b): Acessando a pasta de músicas



Foi escolhida esta opção para acessar a pasta porque cada celular possui um nome diferente de cartão SD (onde comumente ficam as músicas) e não é possível acessá-lo através de código.

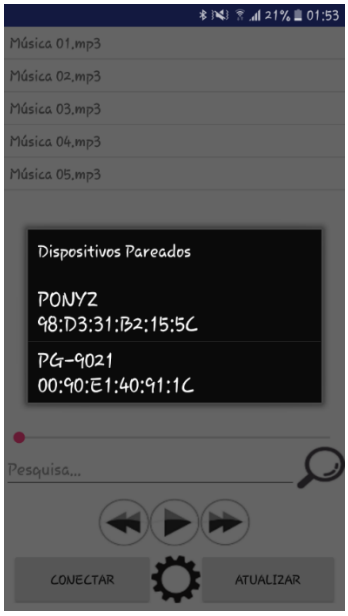
A seguir, deve permitir-se o acesso do aplicativo aos arquivos clicando em "Permitir" (figura 7.a). Então, fechar e abrir novamente o aplicativo. Após esta atualização, as da pasta determinada deverão estar listadas na interface (figura 7.b).

Figura 7 a) e b): Cadastrando o caminho da pasta de músicas



Feito isto, para conectar com o Arduino, deve-se apertar "Conectar", encontrar o nome do módulo Bluetooth e selecioná-lo (figura 8).

Figura 8: Conectando ao Bluetooth do controle remoto



No mais, o botão de configuração acessa novamente a tela de inserir caminho. O botão "atualizar" atualiza a lista de músicas. Na barra de pesquisa é possível pesquisar a música pelo nome.

IV. CONCLUSÕES

Diante do proposto, o controle baseado em Arduino e o aplicativo Android desenvolvidos neste trabalho conseguiram corresponder às expectativas e mostraram ter uma ampla aplicação ao cotidiano. É também um sistema que difere das soluções comumente apresentadas no mercado.

Como continuação deste trabalho, pode-se fazer melhorias estéticas no aplicativo, desenvolver uma placa de circuito impressa e o invólucro do controle, assim como expandir as funcionalidades do aplicativo, adicionando, por exemplo, opção de outros arquivos de mídia, como vídeos.

AGRADECIMENTOS

A autora agradece ao CNPq, pelo incentivo e apoio financeiro, através de bolsa PIBIC, ao colega Vinícius Silva e ao professor Fábio Vincenzi pela colaboração neste trabalho.

REFERÊNCIAS

[1] Nuwer, Rachel. “Why Is Bluetooth Called Bluetooth? (Hint: Vikings!).” *Smithsonian.com*, Smithsonian Institution, 27 Ago. 2012, www.smithsonianmag.com /smart-news/why-is-bluetooth-called-bluetooth-hint-vikings-16270647/.

[2] Hansen, Jarle, and Gheorghita Ghinea. "Multi-Platform Bluetooth Remote Control: Implementation and Results." *Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications*. IGI Global, 2012. 880-898.

[3] Feldbusch, F., Paar, A., Odendahl, M., & Ivanov, I. (2003). The BTRC Bluetooth remote control system. *Personal and Ubiquitous Computing*, 7(2), 102–112. doi:10.1007/s00779-003-0235-x

[4] Guilherme C. A. Tolentino, Douglas B. Tsukamoto, Shigueo Nomura, “Estudo de caso: Utilização do Arduino para um Sistema de Controle remoto de dispositivos via internet”, *XI CEEL*, Nov. 2013.

[5] Anselmo, Luciana. “9 Melhores Apps De Controle Remoto Para Android.” *AppTuts.com.br - Aplicativos Android, IPhone, IPad, Mac OSX e Windows*, Apptuts.com.br, 12 Fev. 2016, https:// www.apptuts.com.br/tutorial/android/melhores-apps-de-controle-remoto-para-android/.

[6] Thingiverse.com. “Mmintbox 1 Enclosure by Vector_Mayhem.” *Thingiverse*, 29 Aug. 2013, https://www.thingiverse.com/thing:142282/collections.

[7] “Download Android Studio and SDK Tools : Android Studio : Android Developers.” *Android Developers*, 17 Ago. 2019, https://developer.android.com/studio.