FACULDADE DE TECNOLOGIA DE AMERICANA

Análise e Desenvolvimento de Sistemas

Elton Lima da Silva

DESENVOLVIMENTO DE UM DISPOSITIVO DE HARDWARE E SOFTWARE COM ARDUINO PARA AUXILIAR NA ACESSIBILIDADE VISUAL.

Americana, S. P.

2017

FACULDADE DE TECNOLOGIA DE AMERICANA

Análise e Desenvolvimento de Sistemas

Elton Lima da Silva

DESENVOLVIMENTO DE UM DISPOSITIVO DE HARDWARE E SOFTWARE COM ARDUINO PARA AUXILIAR NA ACESSIBILIDADE VISUAL.

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso de Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Kleber de Oliveira Andrade.

Área de concentração: Sistemas Embarcados.

Americana, S.P.

2017

S579d SILVA, Elton Lima da

Desenvolvimento de um dispositivo de hardware e software com Arduino para auxiliar na acessibilidade visual ./ Elton Lima da Silva. – Americana: 2017.

87f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Kleber de Oliveira Andrade

1. Sistemas embarcados 2. Dispositivos móveis - aplicativos I. ANDRADE, Kleber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518
681.519

Elton Lima da Silva

DESENVOLVIMENTO DE UM DISPOSITIVO DE HARDWARE E SOFTWARE COM ARDUINO PARA AUXILIAR NA ACESSIBILIDADE VISUAL

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Sistemas Embarcados.

Americana, 28 de junho de 2017.

Banca Examinadora:

Kléber de Oliveira Andrade (Presidente)
Doutor
Fatec Americana

Paula da Fonte Sanches (Presidente)
Mestre
Fatec Americana

Diógenes de Oliveira (Membro)
Mestre
Fatec Americana

AGRADECIMENTOS

A todos os professores que contribuíram para o meu enriquecimento cultural ao longo desses três anos de graduação. Em especial ao meu orientador Prof. Kleber Andrade, pelo apoio, conversas e discussões no processo de elaboração desta monografia, que compartilhou parte da sua sabedoria, conduzindo o trabalho de maneira firme, porém amiga, deixando uma contribuição extremamente importante e positiva nesta fase da minha vida acadêmica. Ao governo do Estado de São Paulo, por proporcionar que, Brasileiros como eu possam realizar seu sonho de graduar-se em um curso superior público e de qualidade. A minha esposa e a todos os meus familiares que, direta ou indiretamente, colaboraram para a conclusão deste trabalho me incentivando e me dando forças para não desistir perante as dificuldades encontradas ao longo desta jornada que se encerra.

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus, a minha esposa pelo apoio e paciência, aos professores e finalmente aos amigos que aqui fiz e que eternamente estarão em minhas lembranças.

RESUMO

Este projeto acadêmico tem como objetivo, proporcionar para as pessoas com limitações visuais uma nova forma de auxílio para sua locomoção, sendo uma ferramenta de apoio para detecção de obstáculos localizados na altura do tórax do indivíduo, evitando assim a colisão frontal com estes objetos, este dispositivo não descarta o uso da tradicional bengala que tem a função de detectar obstáculos no solo, mas é uma alternativa que vem substituir a aquisição de um cão guia que possui um custo elevado. O protótipo basicamente consiste em uma interface de hardware que possui uma placa eletrônica de prototipagem chamada Arduino Uno, um sensor ultrassônico, um módulo de Bluetooth, um micromotor de vibração, uma sirene do tipo *Buzzer*, um *display LCD* e uma bateria de 9 Volts, a lógica consiste em enviar pulsos ultrassônicos que irão colidir com obstáculos à frente, estes pulsos retornarão ao sensor que medirá a distância do objeto, ao detectar este objeto, o motor de vibração e o *Buzzer* serão acionados em intervalos que variam de acordo com a aproximação em relação ao obstáculo, quanto menor a distância maior a taxa de vibração e do alerta sonoro. Os dados referentes à distância serão enviados via Bluetooth para um smartphone que sintetizará em forma de voz a distância do objeto à frente. O usuário também poderá por meio de comando de voz falar um destino do qual gostaria de ir e o aplicativo no smartphone o guiará por meio do *GPS* do aparelho ao local informado.

Palavras Chave: Deficiência visual, Arduino, Ultrassom, Comando de voz

ABSTRACT

This academic project aims to provide for people with visual limitations a new form of aid to locomotion, with a support tool for detecting obstacles located in the individual's chest height, avoiding head-on collision with these objects, this device does not rule out the use of traditional cane that has the function to detect obstacles on the ground, but it is an alternative that replaces the purchase of a guide dog that has a high cost. The prototype basically consists of a hardware interface that has an electronic board prototyping called Arduino Uno, an ultrasonic sensor, a Bluetooth module, a vibration micromotor, a siren Buzzer type, an *LCD* display and a battery 9 Volts, the logic is to send ultrasonic pulses that will collide with the obstacles ahead, these pulses return to the sensor that will measure the distance of the object, to detect an object, vibration engine and the Buzzer will be triggered at intervals that vary according to the approach in the obstacle, the smaller the greater distance the rate of vibration and audible alert. Data on distance will be sent via Bluetooth to a smartphone that synthesize in the form of voice away from the front object. The user can also speak through a voice command a destination that would like to go and the application on the smartphone will guide you through the GPS of the device to the location informed.

Keywords: Visual limitations, Arduino, Ultrasonic sensor, voice command.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

A humanidade tem evoluído exponencialmente em suas conquistas tecnológicas, como a ida do homem à Lua, o envio de robôs ao planeta Marte, telescópios superpotentes como o *Hubble*, a Internet das coisas, o projeto Genoma dentre outras maravilhas. Mesmo assim com toda essa tecnologia, existem pessoas portadoras de necessidades especiais com enormes limitações de acessibilidade. Este trabalho de graduação abordará tecnologias estudadas no decorrer do curso de Análise e desenvolvimento de Sistemas, e que associadas a outras tecnologias de hardware e eletrônica como a placa de prototipagem chamada Arduino, venha a desenvolver um sistema que facilite a vida das pessoas que possuem algum tipo de limitação visual. Proporcionando autonomia e trabalhando em conjunto com as habilidades já adquiridas pelo indivíduo, irá substituir por completo um cão-guia que é extremamente caro e passíveis de falha, pois o animal pode ser influenciado por agentes externos como o clima, doenças, ações de outras pessoas, enfim, o animal além de ser de altíssimo custo e sujeito à agentes externos, também não é totalmente confiável pois possui vontade própria e instinto de autopreservação.

Desde o início da história da humanidade, é sabido que pessoas que nascem ou adquirem no decorrer de suas vidas algum tipo de limitação física, não podem exercer em sua plenitude seus direitos como cidadão, direitos estes aparentemente simples como o de utilizar um transporte público, votar, frequentar uma escola ou simplesmente ter um emprego. Essas tarefas tornam-se praticamente impossíveis no quesito autonomia sem a ajuda de terceiros, um cadeirante ou um indivíduo que possui estas limitações, não consegue ser totalmente independente levando-o a ter uma baixa estima e consequentemente ser excluído da sociedade.

O objetivo geral deste trabalho é desenvolver um dispositivo capaz de auxiliar pessoas com limitações visuais a se locomoverem sem auxílio de um cão guia ou da ajuda de outra pessoa, para isso deve-se incluir como tópicos de pesquisa:

i. Estudar a teoria e prática do funcionamento de uma placa de Arduino, suas versões e micro controladores existentes;

ii. Estudar o funcionamento dos módulos de *Bluetooth* e sensores ultrassônicos;

iii. Demonstrar as características técnicas dos componentes envolvidos com seus *Datasheets*[1] utilizando também de tabelas ilustrativas, figuras e montagem física dos componentes;

iv. Analisar os principais problemas dos portadores de limitações visuais e as alternativas existentes hoje para ajudar sua acessibilidade além das vantagens oferecidas pelo projeto a ser desenvolvido;

v. Estudar como o Arduino é programado, qual linguagem de programação é utilizada em seu micro controlador, a *IDE*[2] oficial disponível no mercado;

vi. Mostrar o que é o Sistema operacional Android, o desenvolvimento de uma aplicação que receberá informações vindas da placa Arduino, como sintetizar em forma de voz através de um dispositivo móvel com Android as medidas de distância do obstáculo à frente e como utilizar a *API*[3] de geolocalização do Google por meio de comando de voz.

Diante destas questões analisadas no estudo de caso, é necessária a construção de um dispositivo que utilize da tecnologia Arduino aliada a sensores de ultrassom. Este dispositivo terá também um módulo *Bluetooth* que estará pareado com um *Smartphone* com sistema operacional *Android*. Os dados relativos ao obstáculo como a distância por exemplo, serão enviados via Bluetooth da placa Arduino para este *Smartphone* que por sua vez irá reproduzir um aviso em forma de voz, informando a presença de um obstáculo, se por algum motivo ocorrer uma falha no pareamento entre Arduino e o dispositivo móvel com Android, existirá uma opção

[1] *Datasheet*, “folha de dados”, documento que identifica detalhes de um determinado produto
[2] *IDE* (*Integrated Development Environment*), “Ambiente Integrado de Desenvolvimento”
[3] *Application Programming Interface*, ou em Português, Interface de programação de aplicativos

de alerta em forma de bipe que irá aumentar de intensidade conforme o obstáculo se aproxima do sensor de ultrassom. Caso a deficiência além de visual também for auditiva, o dispositivo também terá a opção de vibrar indicando a presença de um obstáculo a frente, a intensidade da vibração também aumentará ao passo em que a distância ficar cada vez menor, o dispositivo será fixado na altura do tórax e apontado para a frente, os obstáculos suspensos que não são detectáveis por uma bengala que tateia o chão, serão reconhecidos por este equipamento impedindo a colisão frontal.

A importância deste trabalho além de ajudar as pessoas com necessidades especiais, é deixar como legado aos discentes uma visão diferenciada na aplicação de seus conceitos em matéria de desenvolvimento de aplicações, não se restringindo somente ao desenvolvimento de software mas também no desenvolvimento de hardware, e que ambos trabalhem em sintonia, proporcionando valores mais amplos e reais para a sociedade, fazendo com que os alunos tenham o poder de mudar como nós seres humanos interagimos uns com os outros e com o planeta em que vivemos.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os fundamentos teóricos envolvendo os portadores de deficiência visual, algumas soluções utilizadas até o momento para atenuar suas dificuldades de locomoção, bem como a teoria do funcionamento de todos os componentes eletrônicos envolvidos no projeto.

2.1. Deficiência visual

Segundo a Organização Mundial de Saúde (OMS), que publicou em 1989 um documento que classifica internacionalmente as deficiências, uma deficiência é caracterizada pela perda ou anormalidade de uma função psicológica, fisiológica ou anatômica que torna o indivíduo incapaz de realizar uma atividade na forma em que se considera normal para um ser humano, limitando assim o seu papel de cidadão na sociedade (BREDARIOL, 2010).

A deficiência visual é classificada em diversas áreas conforme a finalidade, dentre elas, as **legais**, onde existe para efeito do ingresso em programas assistenciais e para obtenção de recursos através da previdência social; as **clínicas** que tem por finalidade o diagnóstico, tratamento e acompanhamento médico especializado; as **educacionais** para o processo de ensino-aprendizagem e as **esportivas** que estabelecem critérios de divisão em diferentes categorias para efeito de competições e eventos esportivos (BREDARIOL, 2010).

Uma pessoa com deficiência visual parcial ou total, tem limitações em sua coordenação motora, restringindo assim sua velocidade de trabalho, orientação e mobilidade bem como a capacidade de realizar tarefas rotineiras. Estas dificuldades são mais acentuadas quando a deficiência é adquirida na infância, onde a criança está em desenvolvimento psicomotor, não sendo possível receber estímulos de brinquedos coloridos que causam reações gerando pequenos movimentos com braços e pernas na criança o que não ocorre em bebês videntes, estes problemas em seu desenvolvimento motor gerados por uma falta de estimulação visual, podem causar na criança um atraso em começar a engatinhar e alcançar objetos, posteriormente evoluindo para uma demora em iniciar-se a andar, trazendo

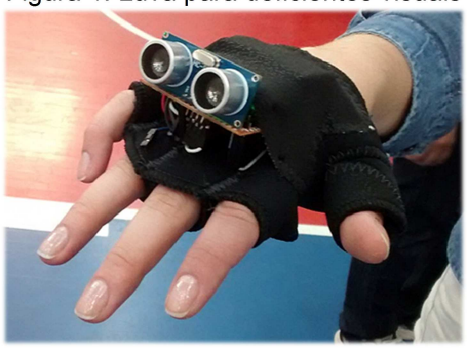
prejuízos no controle de seus músculos, as vezes isto pode ser confundido com distúrbios mentais (BREDARIOL, 2010).

Diante destes impedimentos apresentados, existem algumas soluções alternativas já desenvolvidas seja por meio eletrônico utilizando de projetos de *hardware* e *software* ou através do treinamento de animais conforme será apresentado.

2.2. Luva Eletrônica

É um projeto que é composto basicamente por um sensor de ultrassom, um microcontrolador da família PIC18 do fabricante *Microship* e um motor *vibracall*, sendo alimentado por bateria. A luva deverá ser capaz de determinar a existência de um obstáculo à sua frente bem como a qual distância o objeto se encontra do dispositivo, traduzindo essa distância em estímulos vibratórios de níveis e intensidades pré-determinadas (SCHIRMER, et al., 2015).

Figura 1: Luva para deficientes visuais



Fonte: (SCHIRMER, et al., 2015)

2.3. Bengala Eletrônica

O projeto da bengala tem como objetivo gerar alertas vibratórios e audíveis quando o deficiente visual encontrar um obstáculo em seu caminho. Esta detecção deverá ocorrer para objetos abaixo e acima da linha da cintura, fazendo assim com que o deficiente tenha tempo de desviar do obstáculo antes de colidir com o mesmo (CARDOZO BUENO, 2010).

Figura 2**:** Bengala Eletrônica para deficientes visuais

Fonte**:** (BUENO, 2010, p. 25.)

2.4. Cão guia

Amplamente divulgados em filmes e seriados norte-americanos, os cães-guia aparecem acompanhando deficientes visuais, mostrando-se companhias essenciais para o dia-a-dia, mas no Brasil a realidade é bem diferente, em grandes capitais como São Paulo, existem até lugares nos ônibus destinados a acomodá-los ao lado de seus donos, mas dificilmente as pessoas podem encontrar estes animais nas ruas brasileiras, e isto tem um motivo, o alto custo em adquirir estes animais. Estima-se que milhares de pessoas com deficiência visual no Brasil estão inscritas em programas de entrega destes animais, mas todos encontram a mesma barreira, ausência de dinheiro necessário para a manutenção e treinamento de cães-guias. Dentre os gastos necessários para a certificação destes animais estão, locais de treinamento, treinadores, veterinários e alimentação, fora o fato de que precisam ser substituídos a cada dez anos, além disso ao serem treinados, só podem exercer sua atividade no auge de sua idade onde estão mais maduros emocionalmente, diante deste fato, existem ONGs onde por meio de doações, adestram estes animais e doam a pessoas sem condições financeiras mas a fila de espera é grande e muitas pessoas dizem que é mais fácil ganhar um prêmio na loteria do que um cão guia (DAVID SHALOM, 2014).

Apesar de todo o investimento, treinamento e dedicação para formar um cão-guia, isto não quer dizer que todos os animais serão aptos a exercer sua função, o índice de reprovação é alto, geralmente eles são reprovados por problemas de saúde como problemas de visão, no estômago e no quadril ou também por ter um temperamento que não se encaixa no perfil, o animal não pode ser calmo demais ou muito agitado (SULLIVAN, 2013).

Figura 3:Cão Guia

Fonte: (guichevirtual.com.br[4])

2.5. Arduino

O Arduino é um projeto para prototipagem *open source*[5] de circuitos eletrônicos que surgiu em meio acadêmico, mais especificamente no *Interaction Design Institute* na cidade de *Ivrea* na Itália. O intuito inicial deste projeto era fornecer um meio barato para que os alunos do curso de Arte e *Design* pudessem trabalhar com tecnologia. As tecnologias de prototipagem existentes na época eram caras e de difícil acesso, então os professores se uniram a alguns pesquisadores e alunos e conseguiram utilizar de um micro controlador barato para confeccionar a primeira placa, nascendo assim o Arduino. Hoje em dia esta placa é utilizada nas mais diversas áreas de tecnologia, como por exemplo em Robótica com robôs seguidores de linha (*line-following*), onde é possível fazer o robô seguir uma faixa no

[4] https://www.guichevirtual.com.br/blog/deficiente-visual-embarcar-cao-guia/
[5] Termo em Inglês que se refere a código fonte aberto

chão para a entrega de componentes em um determinado setor de uma fábrica com trajetos pré-determinados (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 4:Arduino UNO

Fonte: (www.arduino.cc[6])

2.5.1. *Hardware* do Arduino

Existem diversas versões do Arduino, todas baseadas em um microprocessador *Atmel AVR* de 8 bits baseadas em arquitetura *RISC*[7]. As primeiras placas eram fabricadas com o microprocessador ATmega8, possuíam uma velocidade de *Clock* de 16 MHz e capacidade de memória *flash* de apenas 8 KB. Depois evoluíram para processadores mais velozes como o Atmega168 com memória de 16 KB com os nomes de Arduino NG Plus e a *Diecimila* (10.000 em Italiano), vieram em seguida as placas com os codinomes *Duemilanove* (2009 em italiano) e *Uno* (1 em Italiano) com processador Atmega328 com memória *flash* de 32 KB, logo em seguida vieram as versões com 128 KB respectivamente estas para projetos mais robustos as conhecidas como Arduino Mega 1280 e uma mais recente chamada de Arduino Mega 2560 com memória de 256 KB. As placas possuem 14 pinos digitais que tanto servem como pinos de entrada como pinos de saída e seis entradas analógicas, além disso, seis pinos podem ser programados para

[6] https://store.arduino.cc/product/A000066
[7] *RISC* (Reduced Instruction Set Computer; "Computador com um conjunto reduzido de instruções")

trabalharem com *PWM*[8], ou seja, fornecendo uma saída de modulação por largura de pulso (EVANS; NOBLE; HOCHENBAUM, 2013).

Tabela 1:Especificações Técnicas Arduino Uno R3

Arduino UNO R3	
Microcontrolador	ATMega328
Tensão de Entrada	7-12V
Tensão de Operação	5V
Portas Digitais	14 (6 podem ser usadas como PWM)
Portas Analógicas	6
Corrente Pinos I/O	20mA
Corrente Pinos 3,3V	50mA
Memória Flash	32KB (0,5KB usado no bootloader)
SRAM	2KB
EEPROM	1KB
Velocidade do Clock	16MHz

Fonte: (www.arduino.cc[9])

2.6. Componentes eletrônicos

Os componentes eletrônicos fazem parte da estrutura do projeto de hardware, são todos os elementos físicos que possuem características elétricas de atuação distintas como resistores, potenciômetros, microcontroladores, sensores ultrassônicos, displays de *LCD*, *Buzzer* e módulo *Bluetooth*, cada um destes componentes terão seus comportamentos e funcionalidades elencadas no decorrer do desenvolvimento do projeto.

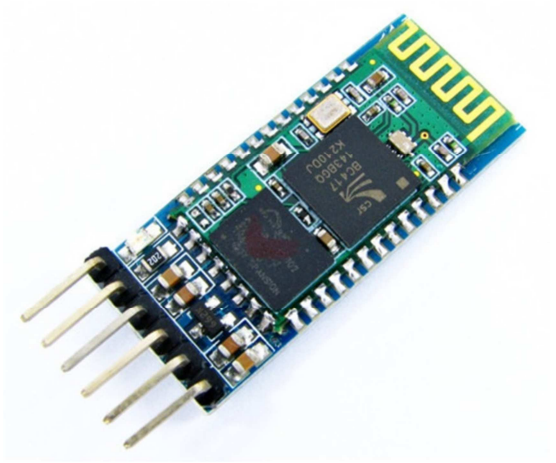
2.6.1. Bluetooth

Foi uma tecnologia sem fio desenvolvida a princípio para substituir o método de comunicação serial com fio existente na época, este recurso inicialmente foi utilizado em fones de ouvido de celulares e em periféricos como mouse e teclados. Este tipo de comunicação é apropriado para curtas distâncias entre dois dispositivos pois seu alcance é limitado embora seja veloz o suficiente para muitos casos. A maneira de transmitir os dados por meio de radiofrequência e semelhante ao método serial padrão, tanto que ao programar o Arduino para utilizar este recurso é como se estivéssemos programando uma porta serial física, a sintaxe é praticamente a

[8] *PWM* (Pulse-Width Modulation), “modulação por largura de pulso”
[9] https://www.arduino.cc/en/Main/ArduinoBoardUno

mesma mudando apenas o meio de transmissão (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 5:Módulo Buetooth HC-05



Fonte: (filipeflop.com[10])

Tabela 2:Especificações Técnicas Módulo *Bluetooth*

Módulo *Bluetooth*	
Protocolo *Bluetooth*	V2.0+EDR
Frequência	2.4Ghz Banda ISM
Tensão	3.3v (2.7-4.2v)
Corrente	Pareado 35mA; conectado 8mA
Temperatura	-40 - +105 °C
Alcance	10m
Baud rate	4800-1382400

Fonte:(www.filipeflop.com[11])

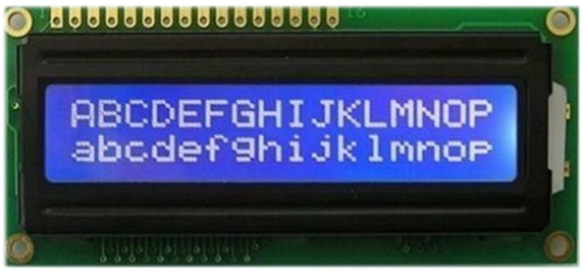
2.6.2. Display *LCD*

Os visores de *LCD*[12] estão cada vez mais presentes em nossas vidas, do momento em que você acorda até o momento em que você vai para a sua cama, ou seja, durante o seu dia-a-dia em algum momento nos deparamos com um visor de *LCD*, no relógio despertador, nos tocadores de MP3, no aparelho de Ar-

[10] http://blog.filipeflop.com/wireless/tutorial-modulo-bluetooth-com-arduino.html
[11] http://www.filipeflop.com/pd-b4742-modulo-bluetooth-rs232-hc-05.html?ct=41d98&p=1&s=1
[12] Liquid Crystal Display ou tela de cristal líquido

condicionado, no relógio de pulso, no carro, no telefone, e até mesmo em telas de *notebooks* e *desktops* o *LCD* está presente, enfim, o display de *LCD* é uma das formas mais primárias de experimentarmos os dados eletrônicos. Existem basicamente dois tipos de displays *LCD*, os que mostram somente caracteres ao usuário que são do tipo Serial e Paralelo, geralmente possuem uma matriz de pixels de 5 x 8 para cada caractere e os *LCDs* gráficos utilizados para imagens e desenhar gráficos, nesta categoria de *LCDs* existem as mais variadas resoluções, podendo ser desde uma matriz simples de 128 x 64 pixels até podendo chegar a uma resolução de um Monitor de computador de 1600 x 1200 pixels (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 6: *LCD* para caracteres – Resolução de 5 x 8 pixels por caractere



Fonte: (www.eletrodex.com[13])

Figura 7: *LCD* gráfico - Resolução de 128 x 64 pixels

Fonte: (www.ebay.com[14])

[13] http://www.eletrodex.com.br/display-lcd-16x2-c-back-verde-pinagem-superior.html
[14] http://www.ebay.com/itm/12864-128X64-Graphic-LCD-Module-Display-Screen-LCM-build-in-KS0108-Controller-/190343763345

Tabela 3:Funcionalidades dos terminais LCD

***LCD* - Conexões dos Pinos**		
Pino	Terminal	Função
1	VSS	GND
2	VDD	+5V
3	V0	Ajuste do Contraste
4	RS	Registrar sinal de seleção
5	R/W	Sinal de Leitura/Escrita
6	E	Habilitar sinal
7	DB0	Barramento de dados
8	DB1	Barramento de dados
9	DB2	Barramento de dados
10	DB3	Barramento de dados
11	DB4	Barramento de dados
12	DB5	Barramento de dados
13	DB6	Barramento de dados
14	DB7	Barramento de dados
15	A	+4.2V para luz de fundo
16	K	GND para luz de fundo

Fonte: (www.xmocular.com[15])

2.6.3. *Buzzer* eletromagnético

O *Buzzer* ou Sirene é um componente que utiliza de um diafragma de metal, um ímã e uma bobina para emitir sons. Quando aplicada uma tensão elétrica nos terminais do componente, é gerado um campo magnético fazendo com que o ímã venha a se mover vibrando assim o diafragma, esta vibração no ar resultará em energia mecânica fazendo com que o ouvido humano reconheça essa energia dissipada em forma de sons. Este tipo de Sirene produz um som suave e agradável, sendo seu uso mais indicado para eletrodomésticos, equipamentos médicos e

[15] http://www.xmocular.com/Upload/CMpdf/GDM1602B-01010892985.pdf

computadores pessoais (Are Electromagnetic Or Piezoelectric Buzzer For You?, 2014, tradução nossa).

Figura 8: *Buzzer* com oscilador interno

Fonte: (www.webtronico.com[16])

Tabela 4: Especificações técnicas *Buzzer*

Buzzer	
Tensão Nominal	5 VDC
Tensão de Operação	4-7 VDC
Máxima Corrente	4mA a 12 VDC
Pressão Sonora	80 dB a 12 VDC
Temperatura de Operação	-20~+70ºC

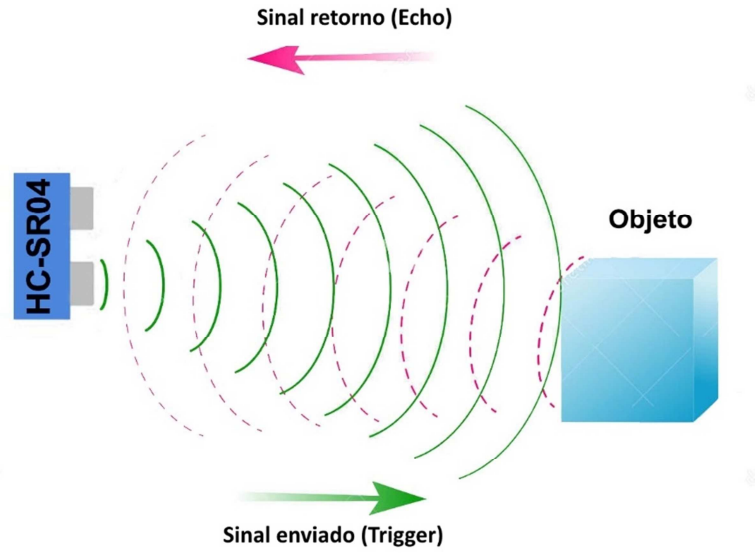
Fonte**:** (www.robocore.net[17])

2.6.4. Sensor de Ultrassom

O ultrassom é um ótimo meio de detectar objetos, principalmente se quisermos saber com precisão sua localização da fonte emissora. Seu princípio de funcionamento é simples, primeiramente emite-se um pulso sonoro através de um módulo transmissor, esta onda sonora vai de encontro com o objeto, bate no mesmo e volta, então aguardamos o eco que será captado pelo módulo receptor, sabendo-se o tempo de ida e volta e a velocidade de propagação do som no ar, podemos calcular a distância deste objeto com certa precisão respeitando as especificações do sensor (EVANS; NOBLE; HOCHENBAUM, 2013).

[16]http://www.webtronico.com/buzzer-dc-5v.html
[17] https://www.robocore.net/loja/produtos/buzzer-5v-ativo.html

Figura 9: Funcionamento sensor ultrassônico



Fonte: (www.sistemaembutido.com[18])

Tecnicamente, o sensor ultrassônico consiste de três componentes, um que envia o sinal sonoro (transmissor), outro que recebe de volta o eco (receptor) e um terceiro que é um micro controlador responsável em determinar o tempo entre o envio do sinal e o retorno. Este valor é codificado em forma de tensão elétrica, quanto maior o atraso, maior será o valor da tensão elétrica que trabalha na faixa entre 0 e 5 volts que é a tensão de trabalho do sensor. Neste projeto, será utilizado um sensor modelo HC-SR04.Isto é conhecido como eco localização, técnica utilizada por animais como o morcego e os golfinhos para localizarem alimento, predadores e para navegação em ambientes escuros ou com pouca luminosidade (EVANS; NOBLE; HOCHENBAUM, 2013).

[18] http://www.sistemaembutido.com.br/article.php?id=118

Figura 10**:** Sensor ultrassônico HC-SR04

Fonte: (www.pandoralab.com[19])

Tabela 5:Especificações Técnicas Sensor de Ultrassom

Sensor Ultrassônico HC-SR04	
Alimentação	5V VDC
Corrente de Operação	2mA
Ângulo de efeito	15 Graus
Alcance	2cm – 4m
Precisão	3mm

Fonte: (www.micropik.com[20])

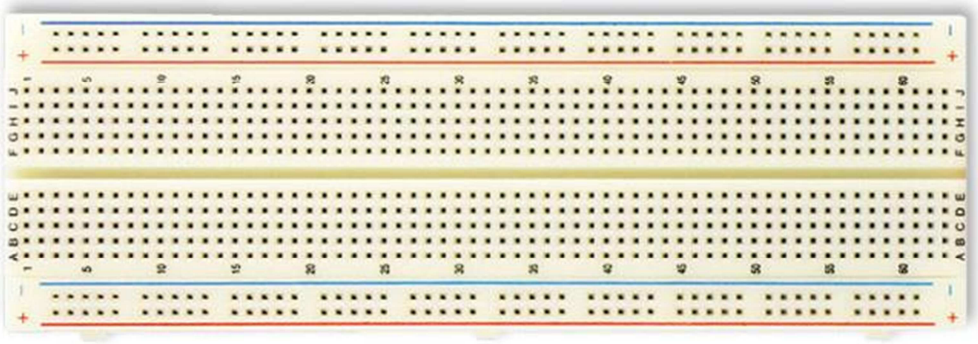
2.6.5. Protoboard

A *Protoboard* ou placa de matriz de contatos, tem como função principal oferecer um meio onde é possível montar os componentes fisicamente de modo provisório para que assim possa-se realizar testes e experimentos antes de ser confeccionada uma placa definitiva para o produto final. Basicamente é composta de uma base de material plástico contendo orifícios onde pode-se encaixar os terminais dos componentes eletrônicos. Internamente possui interligações com material condutivo para que a corrente elétrica venha a fluir entre os componentes do circuito, podendo estes serem retirados facilmente sem a utilização de solda. Estes orifícios são distribuídos em linhas e colunas. As linhas ou barramentos, estão localizados na parte superior e inferior da protoboard, seus furos são interconectados no sentido horizontal. As colunas estão localizadas no centro entre os barramentos e são interligadas no sentido vertical. As colunas são formadas exatamente por cinco furos cada uma. Existem vários grupos de colunas, isto

[19] https://pandoralab.com.br/tutorial/tutorial-regua-eletronica-com-sensor-ultrassonico-hc-sr04/
[20] http://www.micropik.com/PDF/HCSR04.pdf

dependerá do modelo e tamanho da *protoboard*, cada grupo de colunas está separada por uma cavidade central localizada horizontalmente. Essa cavidade divide a *protoboard* em partes iguais podendo variar entre duas ou mais partes (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 11:Placa de prototipagem

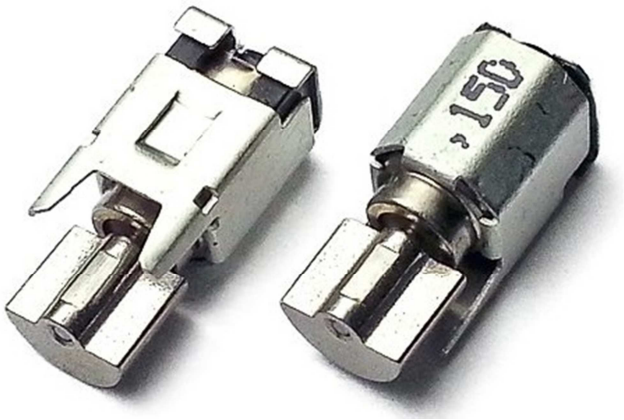


Fonte: (www.eletronicadidatica.com.br[21])

2.6.6. Motor de vibração

É um pequeno motor elétrico que possui uma massa deslocada ligada em seu eixo. A medida que esta massa ou peso é rotacionado pelo motor, este gera uma força centrípeta resultando em um deslocamento do motor. O Motor é constantemente deslocado e movido por esta força repetidas vezes gerando um efeito de vibração através deste desequilíbrio rotativo (UNDERSTANDING ERM VIBRATION MOTOR CHARACTERISTICS, 2017, tradução nossa).

Figura 12: Micromotor de vibração



Fonte: (www.tindie.com[22])

[21] http://www.eletronicadidatica.com.br/protoboard.html
[22] https://www.tindie.com/products/BBTech/micro-vibration-motor-bb-vm02/

2.7. *IDE* Arduino

Para o desenvolvimento da aplicação que irá controlar o microcontrolador, será utilizado o *IDE* oficial do Arduino que pode ser baixado gratuitamente em *www.arduino.cc/en/Main/Software*, tenha certeza de baixar a versão correta para o seu sistema. Disponível para Windows, Mac OS X e Linux, a linguagem utilizada é baseada no C/C++ com diversas bibliotecas para controlar os inúmeros componentes eletrônicos como visor *LCD*, *Bluetooth*, *GPS*, ultrassom, *Wi-fi* dentre outros. É aconselhável optar-se pelo uso dessas bibliotecas para facilitar o desenvolvimento, pois existem diversas rotinas prontas que simplificam e aumentam a produtividade ao desenvolvedor, não sendo necessário um conhecimento profundo do funcionamento interno de cada componente (EVANS; NOBLE; HOCHENBAUM, 2013).

Figura 13:IDE para programação em Arduino

Fonte: (arduino.cc)[23]

[23] https://www.arduino.cc/en/Main/Software

2.8. Android *OS*

O Android é um sistema operacional *Open source* baseado no *kernel* linux e voltado para dispositivos móveis como *smartphones*, *tablets*, relógios, TVs e centrais de multimídia para automóveis conhecido como Android Auto, o kernel Linux tem a função gerenciar processos, drivers, memória e energia do aparelho. Seu código-fonte é distribuído sob licença da Apache, portanto cada fabricante pode modificar sua versão do Android e distribuí-la junto com seus produtos, criando uma versão com a cara da empresa, mas para que o produto saia de fábrica com os aplicativos oficiais da Google como o Gmail, *Google Maps*, *Hangout*, etc., é preciso passar por uma homologação seguindo uma bateria de testes para que as modificações feitas pelo fabricante não venham a comprometer as *APIs* que os desenvolvedores utilizam no desenvolvimento dos aplicativos (GLAUBER, 2015).

2.9. MIT App Inventor

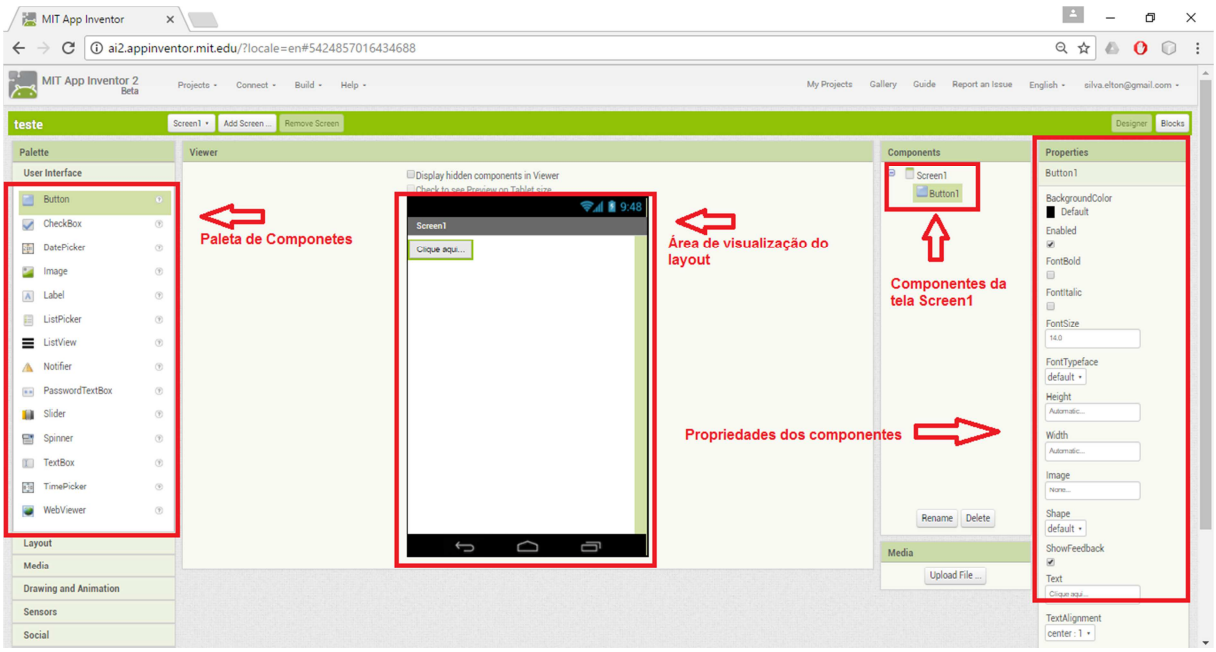
O *MIT App Inventor* é uma tecnologia de desenvolvimento voltado para a construção de aplicativos para dispositivos móveis. Esta iniciativa do *MIT* (*Massachusetts Institute of Technology*) tem como propósito tornar o ato de programar uma tarefa simples, quebrando o paradigma de que para programar é necessário dominar linguagens textuais complexas e que exigem muito tempo de estudo, para uma linguagem totalmente visual onde uma pessoa novata e sem experiência em programação em poucas horas consiga desenvolver aplicativos totalmente funcionais. Esta quebra de conceito, tende a democratizar o processo de desenvolvimento de software, trazendo praticamente todos as pessoas, especialmente os mais jovens a deixarem de serem meros consumidores de tecnologia para se tornarem desenvolvedores das tecnologias que eles mesmos consomem (*MIT App Inventor, 2017).*

O ambiente de desenvolvimento é no próprio *browser*, sendo necessário possuir uma conte de e-mail do Google para fazer a autenticação e assim começar a usar o ambiente. Para executar o aplicativo desenvolvido no App Inventor, existem basicamente três opções, a primeira é rodar o seu aplicativo direto no dispositivo móvel através da rede *Wi-Fi*, a segunda opção é rodar o seu aplicativo em um emulador que deve ser baixado e instalado em seu computador e a terceira opção é executar no dispositivo móvel através de um cabo *USB*. Para este projeto, foi

utilizada a terceira opção por ter um tempo de resposta mais rápido tornando o desenvolvimento mais eficientes não necessitando de rede Wi-Fi e com maior fidelidade nos resultados, pois o aplicativo será executado no ambiente real, ou seja, em um *smartphone* ou *tablet*.

O *IDE* é composto por duas partes básicas, uma responsável pelo *design* do projeto, onde nela é possível clicar e arrastar componentes para dentro da área de visualização, bem parecido com as principais ides do mercado como Android Studio e *Visual Studio* da *Microsoft*. Nesta área de design no lado esquerdo é possível identificar os componentes divididos por categoria, nesta área temos os componentes de interface com o usuário onde existem botões, caixas de texto, *Checkbox*, *Listviews*, componentes de *Layout*, onde é possível escolher qual tipo de *layout* utilizar na aplicação, componentes de Media que fornece acesso a câmera do aparelho, reconhecimento de voz, sintetizador de voz, armazenamento em cartão de memória, gravação de vídeos dentre outros, enfim, diversos componentes já bem conhecidos no mundo do desenvolvimento de aplicativos móveis. Ao centro, nota-se que existe a área de visualização do *Layout* da aplicação, é lá onde serão colocados os componentes visuais da aplicação. A direita, ficam as propriedades dos componentes, podendo ser possível editá-las alterando suas características visuais e comportamentais na aplicação.

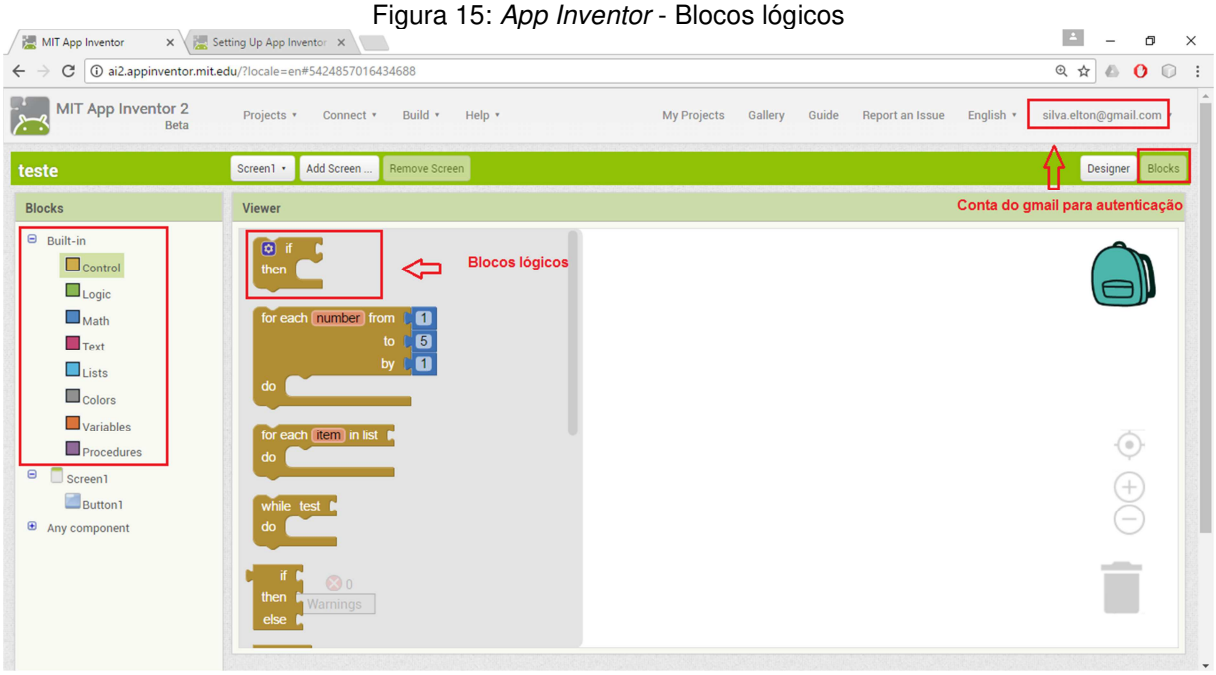
Figura 14:App Inventor - Área de design do projeto



Fonte: Próprio autor

Já na área de Blocos é onde toda a lógica do aplicativo é desenvolvida, nesta área é possível selecionar como os componentes que estão na área de visualização irão se comportar e interagir uns com os outros, nela é possível criar variáveis, instruções de controle com "*if else*", loopings "*while*", criar procedures, definir valores lógicos "*true*" ou "*false*", criar listas, operações matemáticas e manipulação de *strings*, toda a lógica que é possível com linguagem textual é possível com a lógica por blocos que o *App Inventor* oferece.

Figura 15: *App Inventor* - Blocos lógicos



Fonte: Próprio autor

3. PROJETO ELETRÔNICO

Após o levantamento de todos os componentes eletrônicos que serão utilizados no projeto e concluído o estudo detalhado de suas características físicas e lógicas de trabalho, a próxima etapa será a montagem física do protótipo. Para este início de testes tanto de *hardware* quanto de *software* será necessário a utilização da placa *protoboard*, pois facilita a interligação dos componentes antes de futuramente confeccionarmos uma placa eletrônica definitiva. Para alimentar o circuito, a princípio será utilizada energia elétrica proveniente da porta *USB*[24] onde a mesma também será destinada para transferência do código compilado desenvolvido em linguagem C para o microcontrolador ATMEGA328, ao longo do projeto, quando o circuito estiver montado e funcionando adequadamente, o mesmo será alimentado por uma bateria de 9 Volts deixando o protótipo independente do computador para testes em campo. O programa será responsável pelo gerenciamento da captação dos dados referentes à distância dos obstáculos à frente, estes dados serão provindos do sensor de ultrassom e convertidos em centímetros, também acionará o motor de vibração alertando o usuário através de estímulos vibratórios, quanto mais próximo do objeto a frente, maior a intensidade da vibração, também será responsável pelo acionamento do *Buzzer* ou Sirene gerando alertas sonoros de proximidade, o usuário poderá optar entre estas duas modalidades de alertas, o programa ainda será responsável pelo envio das medições para o dispositivo móvel com Android (*smartphone* ou *tablet*) por meio do módulo *Bluetooth* via comunicação serial.

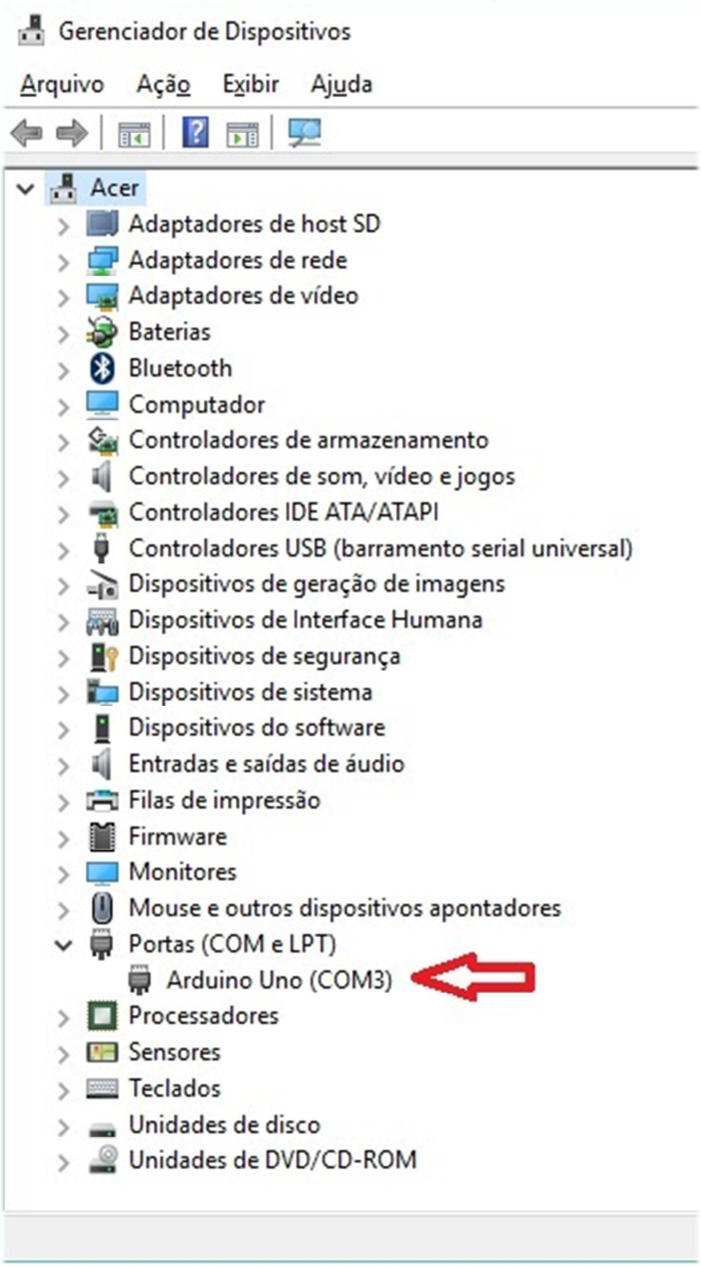
3.1. Preparação do Ambiente

Para dar início ao desenvolvimento deste projeto com Arduino é necessário primeiramente a instalação do JRE ("*Java Runtime Environment*") pois a *IDE* que será utilizada foi desenvolvida com esta tecnologia e não funcionará sem a máquina virtual Java, o instalador pode ser encontrado no link https://www.java.com/pt_BR/download/. Feita a instalação da JRE, o próximo passo é instalar a *IDE* oficial do Arduino que pode ser baixada pelo *link*

[24] USB – "Universal Serial Bus" ou Porta Universal em Português

https://www.arduino.cc/en/main/software. Após instalação da *IDE*, o próximo passo é plugar a placa no computador por meio de um cabo *USB* e aguardar até que o Windows de início ao processo de instalação do driver através do Windows *Update*. Feito isso, checar se a placa Arduino foi reconhecida pelo Sistema Operacional, para tanto, é necessário chamar o gerenciador de dispositivos (“devmgmt.msc”) e localizar em “Portas (COM e LPT) ” se aparece a placa “Arduino Uno”.

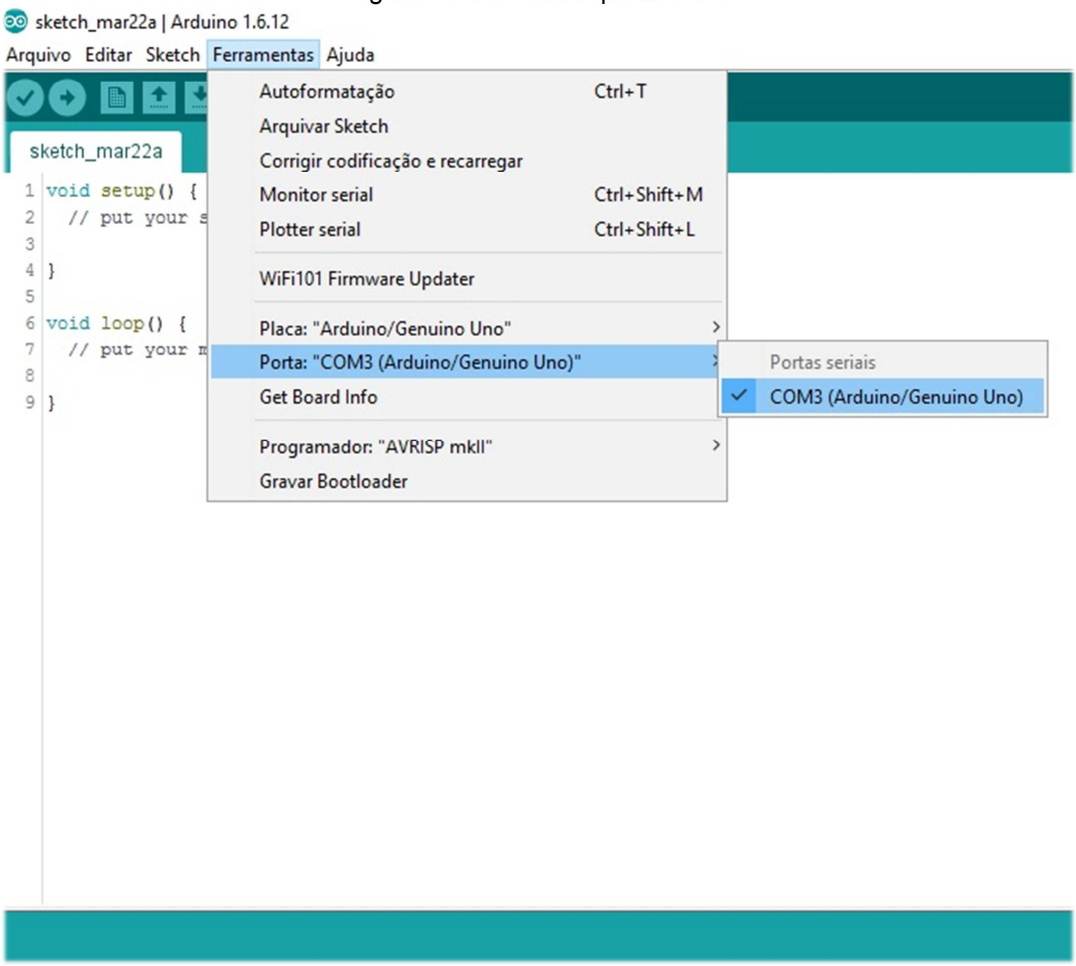
Figura 16:Placa Arduino reconhecida pelo Windows



Fonte: próprio autor

Após o Windows reconhecer a placa, abrir o *IDE* do Arduino, no menu “Ferramentas/Porta” selecione a respectiva porta COM sendo a mesma mostrada no gerenciador de dispositivos.

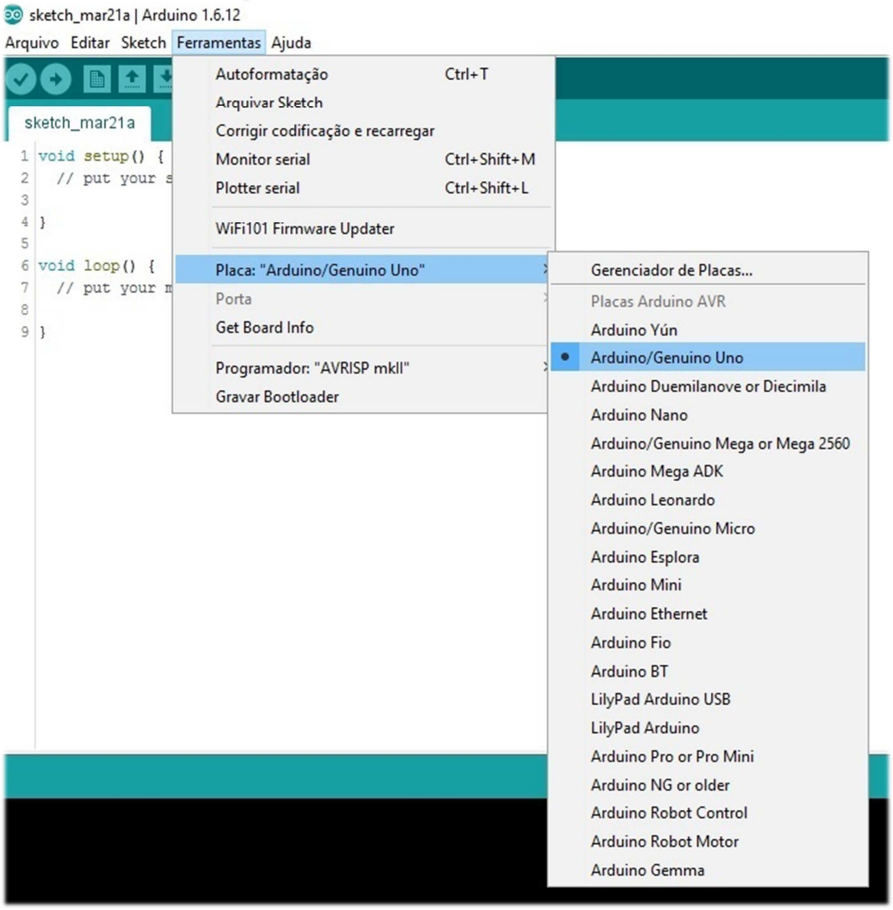
Figura 17:Selecionar porta COM



Fonte: próprio autor

Para selecionar o modelo da placa, acesse o menu “Ferramentas/Placa”, selecione o modelo do Arduino em uso, no caso deste projeto seleciona-se o “Arduino Uno”.

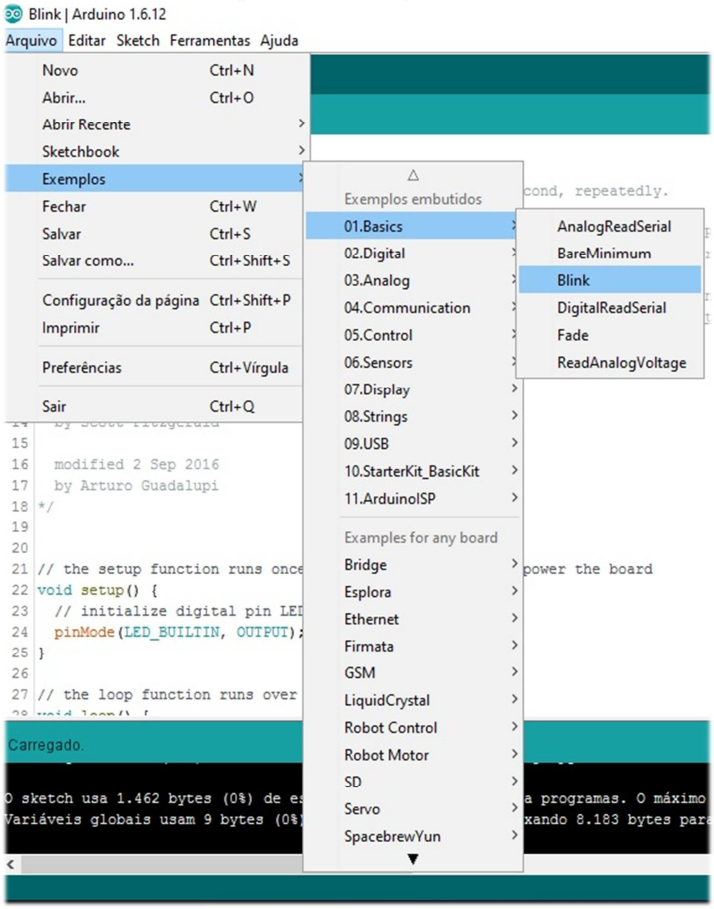
Figura 18: Selecionar modelo da Placa



Fonte**:** próprio autor

Dar início a um teste preliminar de funcionamento da placa, para isto a *IDE* já vem com diversos exemplos de código já prontos, será utilizado para o teste código "*Blink*" que faz com que o *LED* da placa pisque em intervalos de 1000 milissegundos.

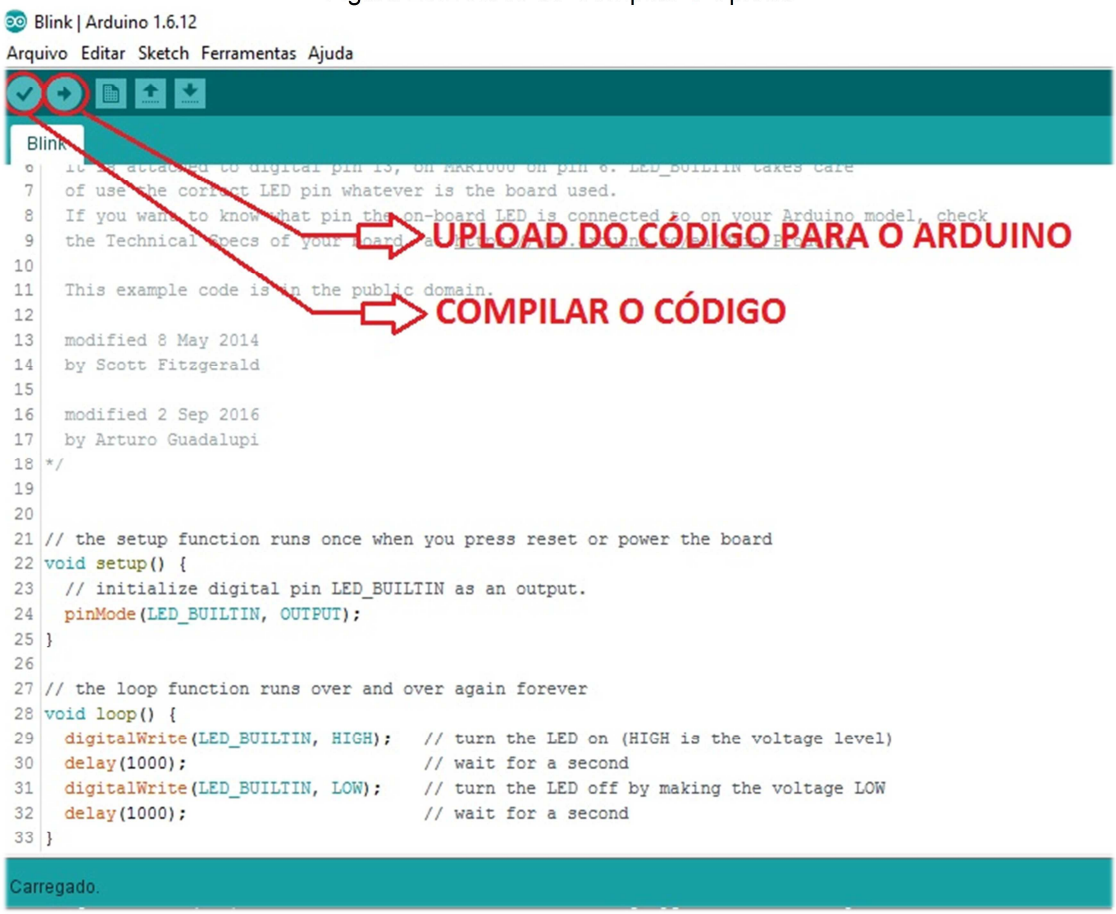
Figura 19:Exemplo Blink



Fonte: próprio autor

Após o carregamento do código é preciso compilar e fazer o *upload* do mesmo para a placa, para isso, o cabo *USB* deve estar conectado no *PC* e na placa do Arduino.

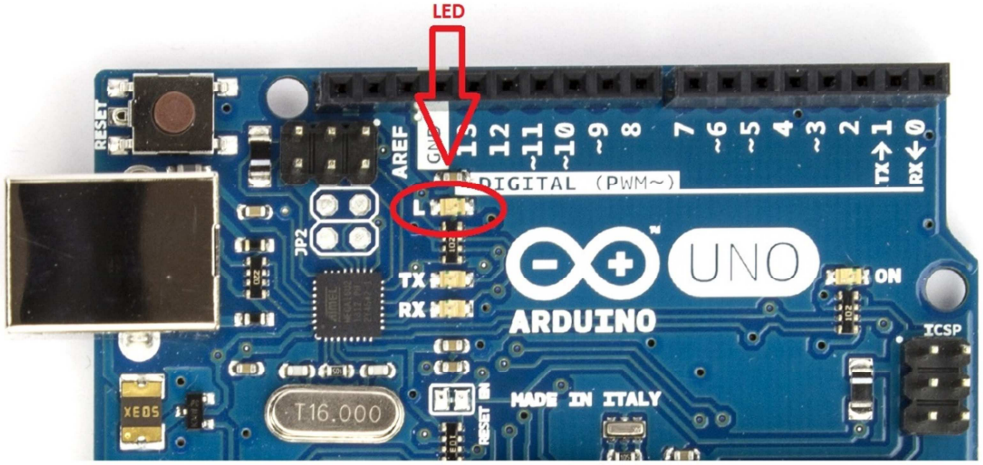
Figura 20:Botões de Compilar e Upload



Fonte: próprio autor

Após o upload da aplicação já é possível ver o *LED* da placa piscando, comprovando que todos os passos da preparação do ambiente já foram concluídos, mediante isso dá-se prosseguimento no projeto montando gradativamente os componentes eletrônicos, codificando e realizando os devidos testes de *hardware* e *software* necessários.

Figura 21:LED da placa Arduino piscando

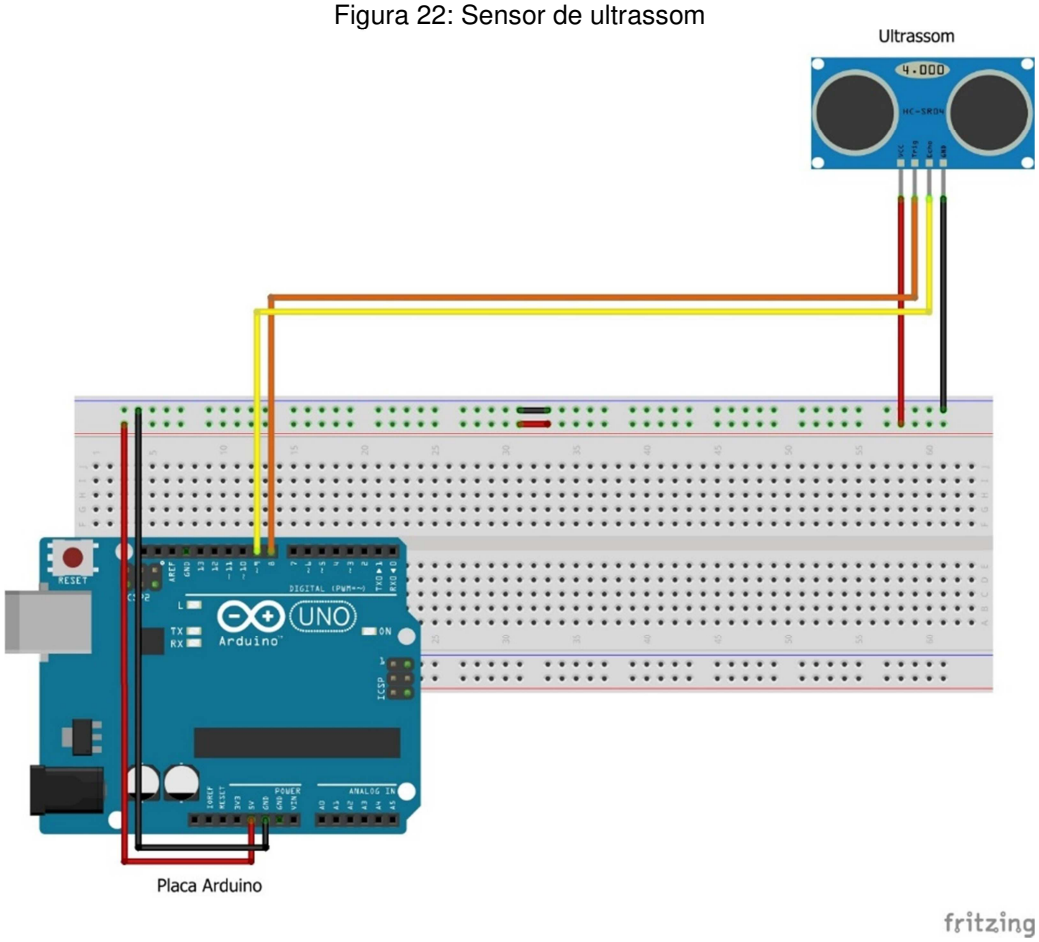


Fonte: próprio autor

3.2. Montagem do ultrassom

A montagem do sensor de ultrassom na *protoboard* inicia-se com a alimentação da placa Arduino, conecta-se a porta do Arduino identificada como “5V” na segunda linha perfurada horizontal no barramento superior da *protoboard* (fio vermelho), a porta do Arduino identificada como ”*GND*” (fio preto) será ligada na primeira linha perfurada na horizontal do barramento da *protoboard*. O sensor de ultrassom possui 4 terminais, *VCC*, *Trigger*, *Echo* e *GND*, todos serão conectados na *protoboard*, o terminal *VCC* será conectado no positivo do barramento da *protoboard*, o *Trigger* na porta digital número 8 do Arduino, o Echo na porta digital número 9 e o *GND* no negativo do barramento da *protoboard*.

Figura 22: Sensor de ultrassom



Fonte: próprio autor

Após a montagem física do componente é necessário o desenvolvimento do código fonte para o microcontrolador do Arduino que irá controlar o sensor, primeiramente, define-se as constantes que por sua vez irão dar nomes as portas digitais ficando assim mais fácil sua utilização no código, bastando apenas referenciá-las pelos seus respectivos nomes, a porta número 8 será chamada de "*TRIGGER*" ficará incumbida de disparar o feixe de ultrassom, a porta número 9 será chamada de "*ECHO*" será colocada em nível alto para aguardar o retorno do som.

```
17 // Pinos para o trigger e echo do ultrassom
18 #define TRIGGER 8
19 #define ECHO    9
```

Nesta área serão definidas as variáveis globais do sistema, estas irão receber informações da duração do tempo de retorno som, o cálculo referente a distância do objeto encontrado e as definições de alcance mínimo e máximo das ondas sonoras,

estas irão ajudar na definição de um filtro para uma faixa de operação que será entre 3 cm até 400 centímetros que é justamente a faixa de operação do sensor segundo dados do fabricante.

```
40 /****************************************
41 *           Variáveis Globais           *
42 ****************************************/
43 // Medidas em cm
44 int alcanceMaximo = 400;
45 int alcanceMinimo = 3;
46
47 float duracao = 0.0;
48 float distancia = 0.0;
```

Esta é a rotina de configuração ou parametrização da placa, nela é possível definir a forma como as portas digitais irão trabalhar, a porta *TRIGGER* que é a de número 8 irá trabalhar em modo *OUTPUT* ou seja, irá acionar algo do mundo externo enviando 5 volts em sua saída (nível alto) e irá disparar um feixe de 8 pulsos sonoros da ordem de 40 KHz cada, a porta *ECHO*, ao contrário, irá trabalhar como *INPUT*, ou seja, irá receber informações vindas do sensor referentes ao tempo de retorno das ondas sonoras ao encontrar um obstáculo.

```
141
142 /****************************************
143 *     Rotina de configuraço do Arduino    *
144 ****************************************/
145 void setup() {
146   pinMode(TRIGGER, OUTPUT);
147   pinMode(ECHO, INPUT);
```

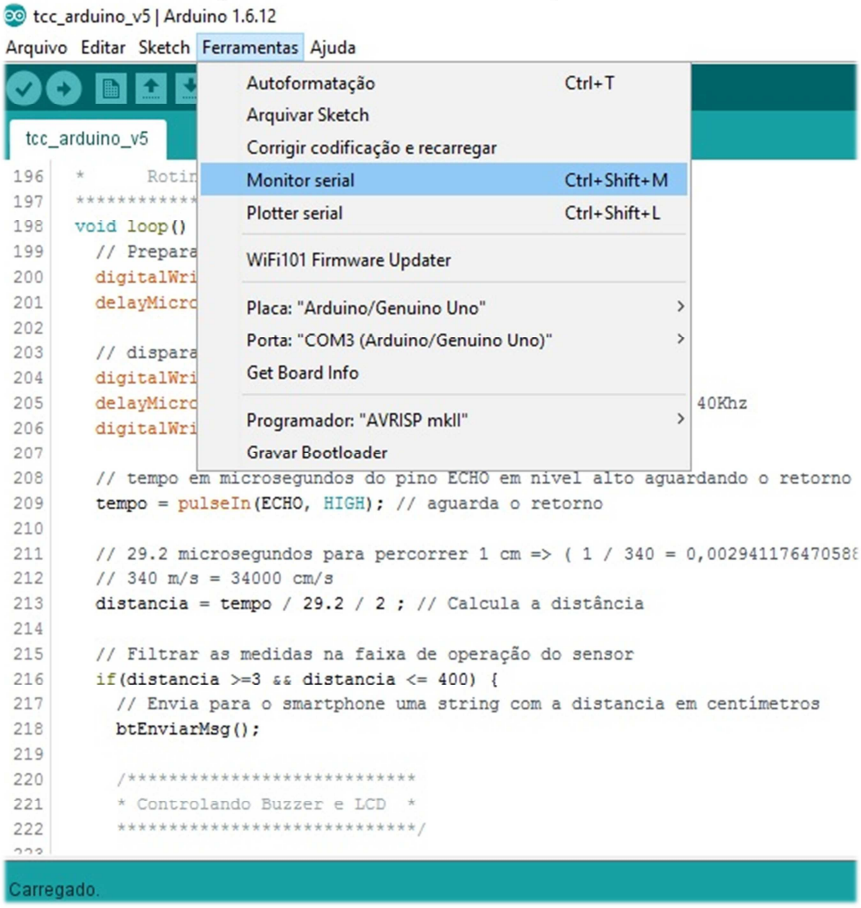
Na rotina “loop()” é onde toda a lógica do programa acontece, nela é possível mostrar o funcionamento do sensor ultrassónico, na linha número 193 o *TRIGGER* ou gatilho é preparado para ficar em nível baixo, ou seja, 0 volts, logo em seguida é feita uma pausa de 2 microssegundos. Em seguida na linha 197 o gatilho é disparado sendo colocado em nível alto, agora é enviado um conjunto de 8 pulsos de 40 KHz de frequência cada durante 10 microssegundos, após isso o gatilho é colocado em nível baixo. O *ECHO* é colocado em nível alto, isto irá permanecer até que o som retorne ao receptor do sensor. Na linha número 202 é chamada a rotina “*pulseIn*” que retornará para a variável “duracao” o tempo de ida e volta dos pulsos sonoros, este tempo é calculado por um microcontrolador embutido no sensor

ultrassônico. A partir deste tempo de retorno do som recebido pela variável “duração” é que será possível fazer o cálculo da distância. Para realizar o cálculo, deve-se dividir por 2 o valor da variável “distancia”, pois, o som vai em direção ao objeto colidindo com o mesmo, depois volta sendo captado pelo receptor, então divide-se por 29.2 que é o tempo em microssegundos que o som leva para percorrer 1 cm, fazendo isso obtém-se a distância em centímetros.

```
188 /***************************
189 *      Rotina principal     *
190 ***************************/
191 void loop() {
192   // Prepara o "gatilho" de disparo do ultrassom
193   digitalWrite(TRIGGER, LOW);
194   delayMicroseconds(2);
195 
196   // dispara o feixe sonoro
197   digitalWrite(TRIGGER, HIGH);
198   delayMicroseconds(10); // tempo para disparar 8 feixes de 40Khz
199   digitalWrite(TRIGGER, LOW);
200 
201   // tempo em microsegundos do pino ECHO em nivel alto aguardando o retorno do som
202   duracao = pulseIn(ECHO, HIGH); // aguarda o retorno
203 
204   // 29.2 microsegundos para percorrer 1 cm => ( 1 / 340 = 0,0029411764705882 )
205   distancia = (duracao/2) / 29.2 ; // Calcula a distância
206 
```

Assim que efetuado o upload para o microcontrolador, o protótipo começa a funcionar. Pode-se monitorar pela *IDE* por meio da porta *USB* os dados referentes à distância dos objetos detectados pelo sensor, para isso acessar o menu “Ferramentas” no item “Monitor Serial”.

Figura 23:Monitorar comunicação Serial

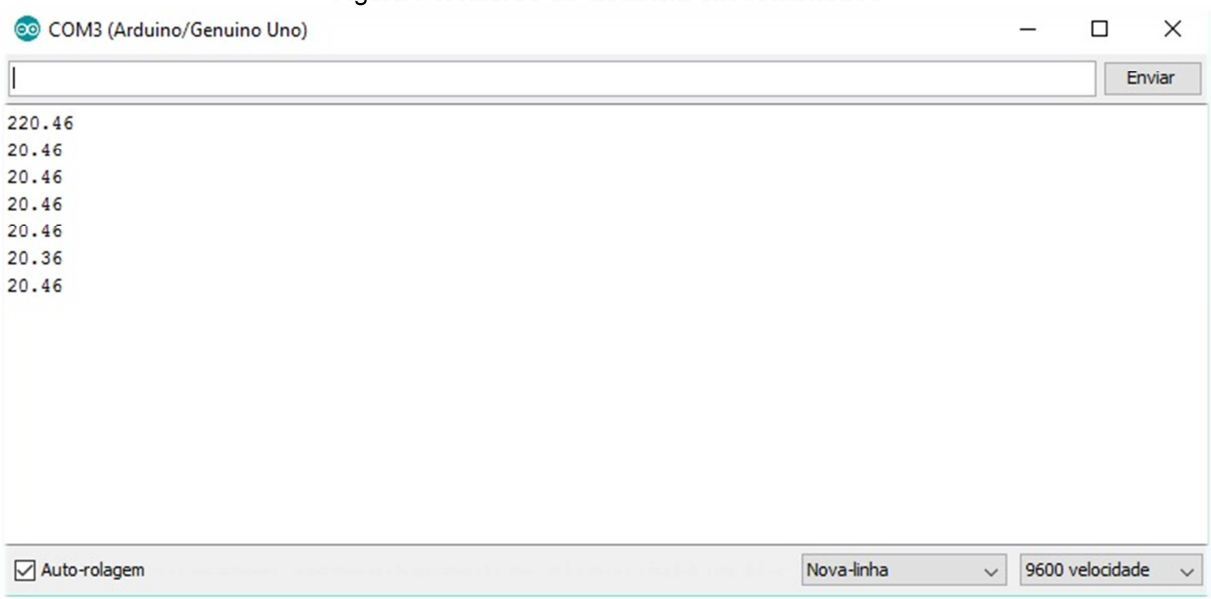


Fonte: próprio autor

Os dados captados pelo módulo de ultrassom agora podem ser visualizados em tempo real sendo uma alternativa de depuração do projeto. É possível também configurar a inserção de uma nova linha após a leitura de cada valor recebido como também a velocidade de transmissão em *baudrate*[25] e também a auto rolagem da tela.

[25] Medida de velocidade para comunicação indicando o número de bits transmitidos por segundo

Figura 24:Valores da distância em centímetros

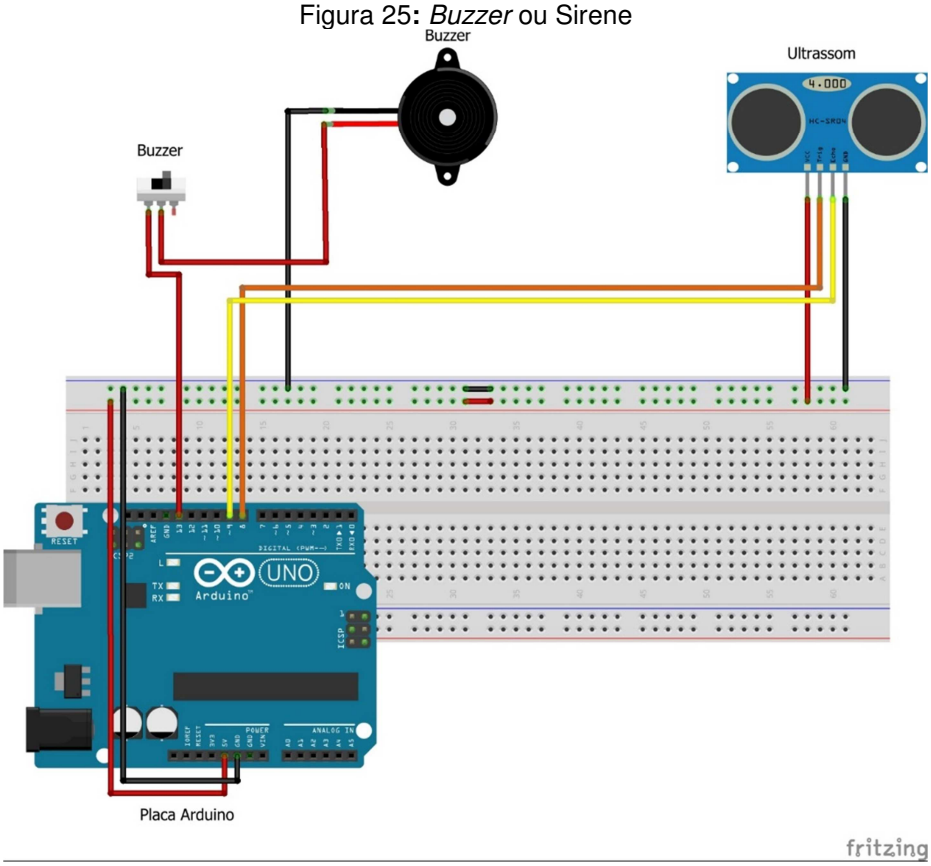


Fonte: próprio autor

3.3. Montagem do *Buzzer*

O *Buzzer* ou Sirene possui dois terminais, um positivo que será conectado a uma mini chave Liga/Desliga, esta será conectada na porta digital 13 da placa do Arduino, possui também um terminal negativo que será conectado ao barramento negativo da *protoboard*. A mini chave Liga/Desliga permite ao usuário optar ou não em ouvir o som emitido pelo *Buzzer*, podendo se preferir ficar somente com o alerta sonoro do *Buzzer* ou com o alerta vibratório gerado pelo motor vibratório que será mostrado no decorrer do projeto.

Figura 25: *Buzzer* ou Sirene



Fonte: próprio autor

Para a codificação do *Buzzer*, também é prudente a criação de uma constante para nomear a porta número 13 conforme exemplificado na linha 20 do código fonte.

```
17 // Pinos para o trigger e echo do ultrassom
18 #define TRIGGER 8
19 #define ECHO    9
20 #define BUZZER 13
```

Após definida a constante que identifica a porta relativa ao acionamento do *Buzzer*, foram desenvolvidas algumas rotinas utilitárias para emitir *Bips*, estas rotinas tem a função de alertar o usuário da distância que este se encontra dos obstáculos, quanto mais próximo, maior o número de bips e inversamente, quanto mais longe, menor a quantidade seguindo a faixa de atuação do sensor ultrassônico que é de 3cm até 4metros. A função "*tone*" recebe como parâmetros primeiramente a porta que será colocada em nível alto (*HIGH*) e que está conectado o dispositivo emissor de som e uma frequência em Hertz, esta frequência pode variar de 31 até 65535 Hertz. A função "*noTone*" como é de se esperar, colocar em nível baixo a porta especificada como parâmetro desligando o dispositivo gerador do som.

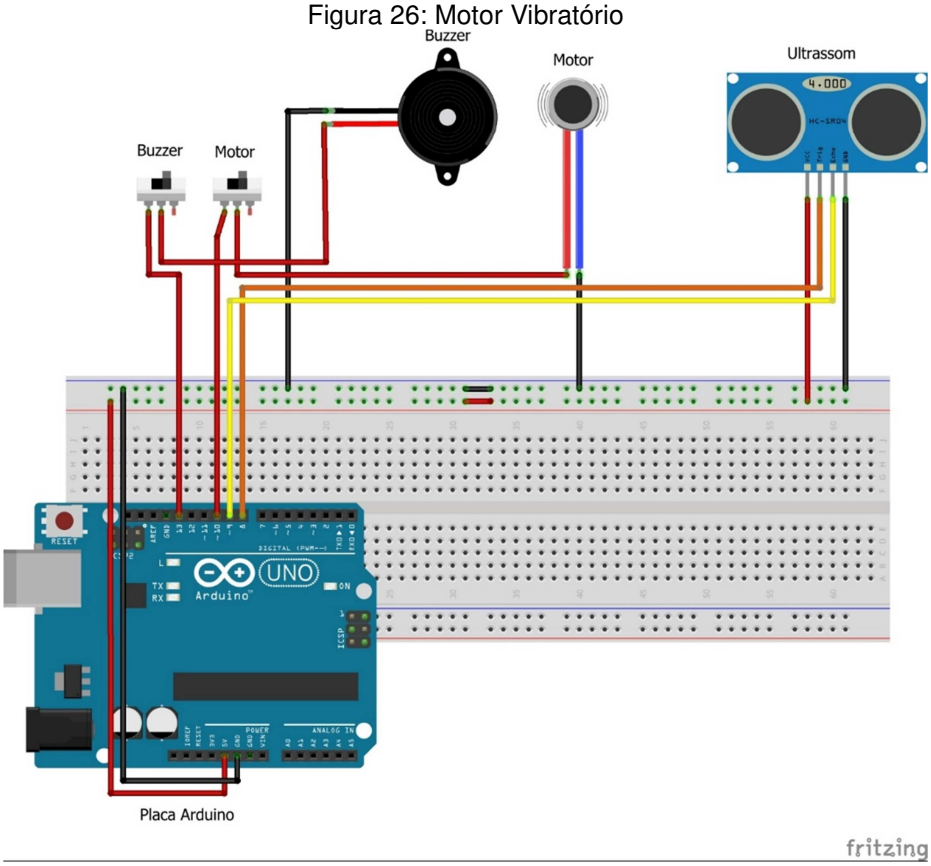
```
71 // Emite 4 bips para objeto curtíssima distância
72 // inferior a 50 centímetos
73 void BeepMuitoPerto() {
74   tone(BUZZER, 440);
75   delay(50);
76   noTone(BUZZER);
77   delay(100);
78   tone(BUZZER, 200);
79   delay(50);
80   noTone(BUZZER);
81   delay(100);
82   tone(BUZZER, 100);
83   delay(50);
84   noTone(BUZZER);
85   delay(100);
86   tone(BUZZER, 150);
87   delay(50);
88   noTone(BUZZER);
89   delay(100);
90 }
```

```
92 // Emite 3 bips para objeto muito proximo
93 // maior que 50 centimetros até 1 metro
94 void BeepPerto() {
95   tone(BUZZER, 440);
96   delay(50);
97   noTone(BUZZER);
98   delay(100);
99   tone(BUZZER, 200);
100  delay(50);
101  noTone(BUZZER);
102  delay(100);
103  tone(BUZZER, 100);
104  delay(50);
105  noTone(BUZZER);
106  delay(100);
107 }
...
109 // Emite 2 bips para objeto a media distancia
110 // entre 1 e 2 metros
111 void BeepMedio() {
112   tone(BUZZER, 440);
113   delay(50);
114   noTone(BUZZER);
115   delay(100);
116   tone(BUZZER, 200);
117   delay(50);
118   noTone(BUZZER);
119   delay(100);
120 }
121
122 // Emite 1 bip para objeto longe
123 // entre 2 e 3 metros
124 void BeepLonge() {
125   tone(BUZZER, 440);
126   delay(50);
127   noTone(BUZZER);
128   delay(100);
129 }
```

3.4. Montagem do Motor Vibratório

A montagem do motor de vibração é simples, ele possui dois terminais, um positivo (vermelho) e outro negativo (preto), o positivo será conectado em uma Mini chave Liga/Desliga que por sua vez será conectada na porta digital número 10 da placa Arduino, o terminal negativo será conectado no barramento negativo da *protoboard*.

Figura 26: Motor Vibratório



Fonte: próprio autor

Definir em seguida as constantes utilizadas para referenciar o pino da saída digital para o acionamento do motor na porta 10, para esta porta atribui-se a constante “MOTOR” para fazer referência ao motor de vibração.

```
17 // Pinos para o trigger e echo do ultrassom
18 #define TRIGGER 8
19 #define ECHO    9
20 #define BUZZER 13
21 #define MOTOR  10
```

Foi desenvolvida uma rotina utilitária chamada “AcionaMotor()” que tem o propósito de acionar o motor vibratório por um determinado intervalo de tempo e no número de vezes desejado. Esta rotina é composta por um laço principal que é incrementado através da quantidade de acionamentos desejados passados como parâmetro para a função, dentro do laço pode-se observar que a porta relativa ao motor é colocada em nível alto(*HIGH* = 5 volts), acionando assim o motor, em seguida é chamada a função “*delay*” que tem o propósito de estabelecer uma pausa

delimitando assim o tempo em que fica acionado, depois a porta é colocada em nível baixo(*LOW* = 0 volts) desligando o motor e novamente aguardando por uma nova chamada de "*delay*" agora marcando o tempo de parada. Este intervalo de pausa acionado e pausa desligado dá a sensação de pulsação intermitente, chamando a atenção do usuário para o obstáculo que está a sua frente, se está muito perto, o motor irá vibrar mais vezes e em intervalos menores se está longe vibrará em menor quantidade e em intervalos mais espaçados.

```
61 // Aciona o vibracall em milisegundos
62 void AcionaMotor(int tempoAcionado, int quantidade, int tempoParado) {
63   for (int i = 1; i <= quantidade; i++) {
64     digitalWrite(MOTOR, HIGH);
65     delay(tempoAcionado);
66     digitalWrite(MOTOR, LOW);
67     delay(tempoParado);
68   }
69 }
```

Na área de configuração do Arduino, "*setup*()" foi especificado o modo de operação da porta MOTOR, que irá trabalhar em modo *OUTPUT*, ou seja, para acionamento do dispositivo conforme linha 149.

```
142 /*****************************************
143 *     Rotina de configuraço do Arduino    *
144 *****************************************/
145 void setup() {
146   pinMode(TRIGGER, OUTPUT);
147   pinMode(ECHO, INPUT);
148   pinMode(BUZZER, OUTPUT);
149   pinMode(MOTOR, OUTPUT);
```

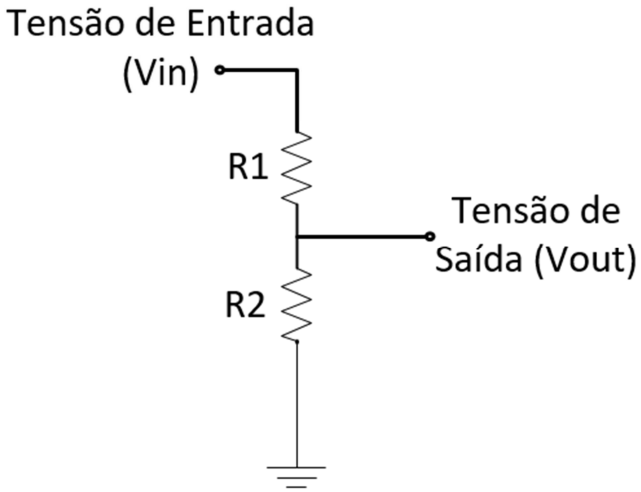
Na área de configuração da placa Arduino, foi realizada uma sequência de instruções que fazem um auto teste do motor acionando-o por 1 segundo ou 1000 milissegundos.

3.5. Montagem do *Bluetooth*

Este componente possui 6 terminais, dentre eles, serão utilizados apenas 4 respectivamente, RXD, TXD, GND e *VCC*. A comunicação entre o *Bluetooth* e a placa Arduino é feita de forma serial, o *Bluetooth* envia dados serialmente para a placa Arduino, portanto o terminal RXD (*Receiver* ou Receptor) do Bluetooth deverá

ser conectado na porta TXD (*Transciever* ou Transmissor) do Arduino, e o terminal TXD do *Bluetooth* na porta RXD do Arduino. Para a montagem do módulo de *Bluetooth*, deve-se prestar uma devida atenção no nível de tensão nominal do dispositivo, uma vez que existem no mercado dispositivos que trabalham com 5 volts e outros modelos na faixa de 3.3 volts, no caso deste projeto, será utilizado um componente modelo HC-05 que trabalha com uma tensão nominal de 3.3 volts (2.7~4.2V), portanto é necessária a utilização de divisores de tensão para que se obtenha uma queda na tensão fornecida pela placa Arduino (5 volts) para não danificar o componente, esta queda será obtida por meio da associação de componentes elétricos chamados resistores que, quando ligados em série, proporcionam uma redução na tensão elétrica próxima ao valor esperado (3,3 Volts).

Figura 27:Divisor de tensão com resistores



Fonte: (www.arduinoecia.com.br[26])

Para obter-se a correta queda de tensão necessária para o funcionamento do Bluetooth será necessário aplicar a fórmula de divisor de tensão com resistores. Basicamente a tensão de saída necessária será o resultado da divisão de R2, ou seja, resistor 2 (2.200 Ω) pela soma dos dois resistores R1+R2 multiplicado pela tensão de entrada fornecida pela placa Arduino (5 volts).

$$\boldsymbol{Vout = \frac{R2}{R1+R2} * Vin}$$

Demonstrando as variáveis mencionadas na fórmula:

R1 = 1500 Ω

R2 = 2200 Ω

Vin = 5 volts (tensão de entrada, fornecida pela placa Arduino)

Vout = Tensão de saída, espera-se uma tensão aproximada de 3.3 volts

Executando os cálculos deduz-se que:

Vout = 2200 / (1500 + 2200) * 5

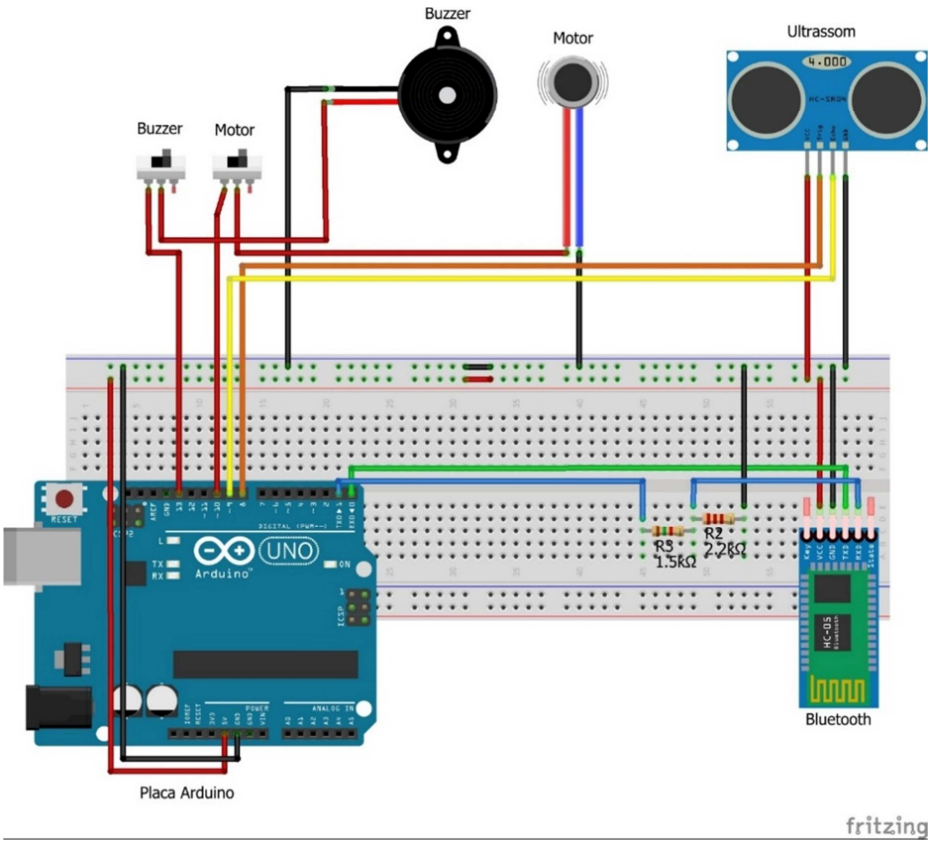
Vout = 2200 / 3700 * 5

Vout = 0,5945 * 5

Vout = 2,9729 ~ **3 volts**

Após efetuado o cálculo de queda de tensão, o terminal *GND* do *Bluetooth* deverá ser conectado no barramento negativo da protoboard e o *VCC* no barramento positivo.

Figura 28:Módulo *Bluetooth*



Fonte: próprio autor

[26] http://www.arduinoecia.com.br/p/calculador-divisor-de-tensao-function.html

Carregar a biblioteca para comunicação serial entre o módulo *Bluetooth*, a placa Arduino e a aplicação do Android do *Smartphone*.

```
10 #include <SoftwareSerial.h> // Carrega a biblioteca para comunic. Serial
```

Nesta área é definido os nomes das portas 0 e 1 do Arduino responsáveis pela comunicação da placa Arduino com o módulo *Bluetooth*.

```
23 // Pinagem de comunicacao RX/TX Bluetooth
24 #define RX_BT 0
25 #define TX_BT 1
```

Inicializa o objeto “bt” passando como parâmetros as portas de comunicação 0 e 1 nomeadas de RX_BT e TX_BT.

```
27 // Inicializa bluetooth nos pinos 0 e 1
28 SoftwareSerial bt(RX_BT, TX_BT);
```

Esta rotina tem a função de enviar os valores obtidos pelo ultrassom na forma de uma cadeira de caracteres para uma aplicação Android em um dispositivo móvel desde que esteja pareado com o módulo de *Bluetooth*.

```
55 // Envia para o smartphone uma string com a distancia
56 void btEnviarMsg() {
57   bt.println(distancia);
58   delay(2000);
59 }
```

A partir de agora com a utilização do Bluetooth, motor de vibração e *Buzzer* é possível enviar as informações coletadas pelo ultrassom para um dispositivo móvel e também trabalhar com medições e filtros atuando o *Buzzer* e o motor de vibração conforme variam a distância do obstáculo a frente, após obtida a distância, será possível o envio através da rotina “btEnviar” conforme a linha 208.

```
191 void loop() {
192   // Prepara o "gatilho" de disparo do ultrassom
193   digitalWrite(TRIGGER, LOW);
194   delayMicroseconds(2);
195 
196   // dispara o feixe sonoro
197   digitalWrite(TRIGGER, HIGH);
198   delayMicroseconds(10); // tempo para disparar 8 feixes de 40Khz
199   digitalWrite(TRIGGER, LOW);
200 
201   // tempo em microsegundos do pino ECHO em nivel alto aguardando o retorno do som
202   duracao = pulseIn(ECHO, HIGH); // aguarda o retorno
203 
204   // 29.2 microsegundos para percorrer 1 cm => ( 1 / 340 = 0,0029411764705882 )
205   distancia = (duracao/2) / 29.2 ; // Calcula a distância
206 
207   // Envia para o smartphone uma string com a distancia em centímetros
208   btEnviarMsg();
```

Feito o envio, agora pode-se trabalhar com a faixa de atuação do sensor ultrassônico acionando o *Buzzer* e/ou Motor vibratório quando necessário.

```
166   /**********************************
167   * Controlando Buzzer, LCD e Motor  *
168   **********************************/
169 
170   // Distância em Centímetos
171   if (distancia < 100) {
172     lcd.clear();
173     lcd.setCursor(0, 0);
174     lcd.print("Centimetros: ");
175     lcd.setCursor(0, 1);
176     lcd.print(round(distancia));
177 
178     if (distancia < 50) {
179       AcionaMotor(1000, 1, 0);
180       BeepMuitoPerto();
181     }
182     else {
183       AcionaMotor(100, 3, 100);
184       BeepPerto();
185     }
186   }
```

```
188   // Distância em metros
189   else {
190     lcd.clear();
191     lcd.setCursor(0, 0);
192     lcd.print("Metros: ");
193     lcd.setCursor(0, 1);
194     lcd.print(distancia/100);
195
196     if (distancia >= 100 && distancia < 200) {
197       AcionaMotor(100, 2, 100);
198       BeepMedio();
199     }
200
201     else if (distancia >= 200 && distancia < 400) {
202       AcionaMotor(100, 1, 100);
203       BeepLonge();
204     }
205
206     // Fora de alcance
207     else {
208       lcd.clear();
209       lcd.setCursor(0, 0);
210       lcd.print("Fora de alcance");
211     }
212   }
213 } // FIM
```

É importante frisar que antes de fazer o upload do código compilado para o Arduino é preciso cortar a alimentação do módulo *Bluetooth*, pois o *upload* será realizado entre o computador e a placa do Arduino de forma serial através do cabo *USB* e o módulo *Bluetooth* e a placa do Arduino também se comunicam de forma serial, sendo assim haverá um conflito na comunicação e fatalmente um erro na transferência do código para o microcontrolador. Após o término do upload reconecta-se a alimentação para o módulo *Bluetooth*.

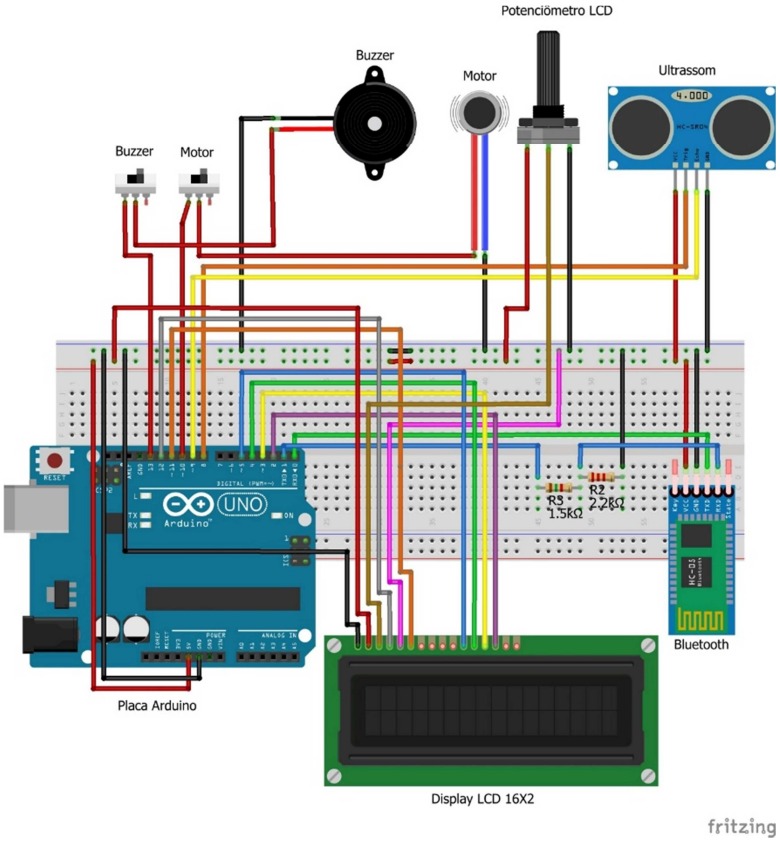
3.6. Montagem do *LCD*

O Display de *LCD* será utilizado apenas para demonstração em campo e aferição comparando os dados recebidos pelo hardware com os dados recebidos pelo software do dispositivo móvel rodando um aplicativo em *Android*, sua montagem é um pouco mais complexa comparada aos demais componentes envolvidos até agora pois possui mais conectores, também para ajuste da

luminosidade do *display*, será utilizado um componente chamado potenciômetro ou resistor variável.

Seguindo as especificações do fabricante, primeiramente liga-se o pino 1 (VSS) no barramento negativo do *protoboard*, o pino 2 (VDD) no barramento positivo, pino 3 (VO) vai no terminal "comum" do potenciômetro que é o terminal central do componente, os outros dois terminais do potenciômetro serão ligados um no barramento positivo e o outro no negativo, pino 4 (RS) vai na porta digital 12 do Arduino, pino 5 (*R/W*) vai no barramento negativo da protoboard, pino 6 na porta 11 do Arduino, pinos 7, 8, 9, 10 não serão utilizados, pino 11 (DB4) ligado na porta 5, pino 12 (DB5) na porta 4, pino 13 (DB6) na porta 3 e finalmente o pino 14 (DB7) será conectado na porta 2, os pinos 15 e 16 não serão utilizados.

Figura 29:Montagem do LCD



Fonte: próprio autor

Chamada à biblioteca "*LiquidCrystal*" que tem a função de oferecer rotinas para o controle do display de LCD.

```
11 #include <LiquidCrystal.h> // Carrega Biblioteca de controle do LCD
```

Inicializa o Display especificando as portas do Arduino utilizadas para envio dos dados.

```
30 // Inicializa lcd na pinagem abaixo
31 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
32   // RS  - Pino 12
33   // R/W - Pino GND
34   // E   - Pino 11
35   // DB4 - Pino 5
36   // DB5 - Pino 4
37   // DB6 - Pino 3
38   // DB7 - Pino 2
```

Configura o *LCD* na linha 152 para atuar com uma resolução de 16 colunas por 2 linhas, visto que este tipo de *display* é somente para mostrar caracteres, portanto não temos um controle sobre cada pixel individualmente, somente sobre caracteres ASCII.

```
142 /*****************************************
143 *     Rotina de configuraço do Arduino    *
144 *****************************************/
145 void setup() {
146   pinMode(TRIGGER, OUTPUT);
147   pinMode(ECHO, INPUT);
148   pinMode(BUZZER, OUTPUT);
149   pinMode(MOTOR, OUTPUT);
150
151   // Inicializa LCD com 16 colunas x 2 linhas
152   lcd.begin(16, 2);
```

4. PROJETO DO APLICATIVO

Este capítulo tem por objetivo mostrar o desenvolvimento do aplicativo para Android utilizando a *IDE* online do *MIT App Inventor*. Este app irá fazer a comunicação com o Arduino via *Bluetooth*, recebendo informações da distância dos obstáculos em centímetros e falando para o usuário com limitação visual a presença dos mesmos quando estiverem a uma distância de iminente colisão. Serão utilizadas técnicas de engenharia de software como, levantamento de requisitos, diagrama de casos de uso e *Mock-Up*[27] da aplicação.

4.1. Levantamento dos Requisitos

De uma forma geral, o levantamento de requisitos tem por finalidade entender as necessidades do cliente, as regras do negócio, o impacto do software sobre o negócio, o que o cliente quer e principalmente como será a interação dos usuários com este software. Estes requisitos podem ser classificados como funcionais onde o intuito é descrever as necessidades que o sistema pretende atender e os requisitos não funcionais, onde serão descritas as restrições e propriedades que o sistema deverá possuir (PRESSMAN, 2011).

[27] É um modelo em escala ou de tamanho real de um projeto ou dispositivo, usado para ensino, demonstração, avaliação de design.

4.1.1. Requisitos Funcionais

Será elencado a seguir as funcionalidades do aplicativo para Android e também do aplicativo para Arduino tanto em nível de *Software* como também de *hardware*, tarefas estas que são necessárias para o correto funcionamento e interação entre os dispositivos e usuários utilizadores do sistema.

Tabela 6: Requisitos Funcionais

Identificador	Nome
RF001	Android: Parear com *Bluetooth*
RF002	Android: Receber Dados
RF003	Android: Geolocalização
RF004	Android: Traçar rota
RF005	Android: Log de eventos
RF006	Android: Interagir com Usuário através da voz
RF007	Arduino: Vibrar
RF008	Arduino: Emitir Sons
RF009	Arduino: Transmitir dados
RF010	Arduino: Parear com *Bluetooth*
RF011	Arduino: Notas musicais ao ligar equipamento

Fonte: próprio autor

4.1.2. Requisitos Não Funcionais

Os requisitos não funcionais mostram as necessidades e restrições que fogem do escopo do *software* para que todo o sistema tenha condições de trabalhar adequadamente.

Tabela 7:Requisitos Não Funcionais

Identificador	Nome
RNF001	Android: Deve ser compatível com *Android* a partir da versão 4.4
RNF002	Android: Necessita estar pareado para receber dados do Arduino
RFN003	Android: Deve ser capaz de navegar por *GPS* sem estar pareado

RNF004	Android: Deve ter um design minimalista
RNF005	Arduino: Deve possuir carga suficiente na bateria
RNF006	Arduino: Sensores funcionando adequadamente

Fonte: próprio autor

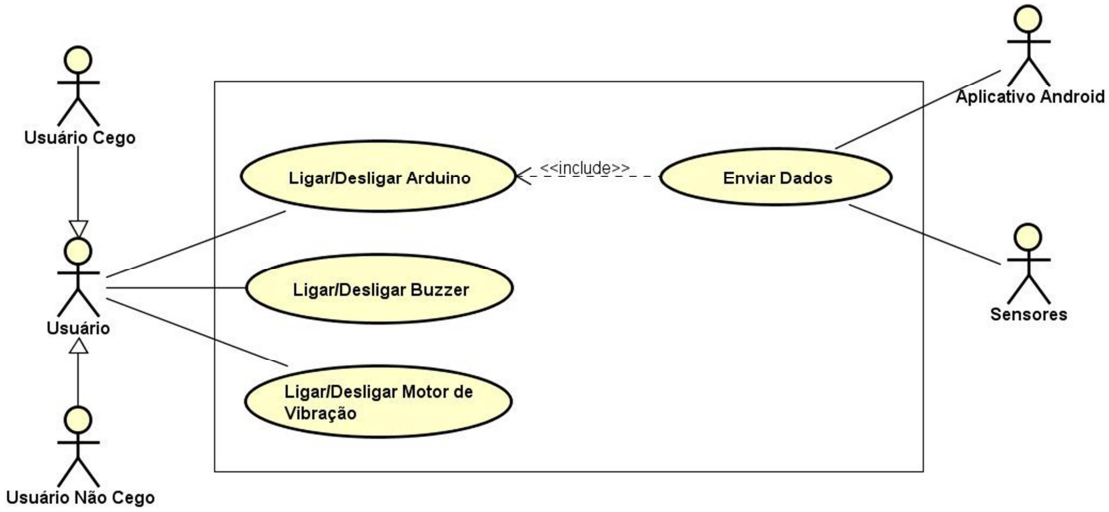
4.2. Diagrama de Caso de uso

O diagrama de caso de uso tem como objetivo, descrever o comportamento da aplicação de acordo as necessidades dos atores envolvidos no sistema. Isto descreve como o usuário final exercendo diversos papéis interage com o sistema sob várias circunstâncias específicas (PRESSMAN, 2011).

4.2.1. Diagrama de Caso de Uso - Arduino

Este diagrama de caso de uso é específico para o Arduino, ele mostra a interação entre os usuários utilizadores do sistema, entre os sensores responsáveis pela coleta e envio dos dados para o aplicativo Android e o acionamento de atuadores como o *Buzzer* e o motor de vibração que são meios secundários de alerta ao usuário evidenciando a presença de obstáculos em seu trajeto.

Figura 30:Caso de Uso - Arduino



Fonte: próprio autor

A seguir serão descritos detalhadamente em forma de tabelas cada item do diagrama de Caso de uso, envolvendo os atores principais e secundários, um resumo detalhado de cada item em destaque, as ações de cada ator e as ações do Sistema bem como as restrições necessárias para que estas ações possam ocorrer.

4.2.2. Documentação do Caso de Uso – Arduino

Esta tabela tem como objetivo, elucidar passo a passo o desenvolvimento da lógica de como é ligar o hardware composto pelo Arduino e seus sensores, o Arduino é a base para que todos os sensores e atuadores possam trabalhar em conjunto sob a lógica programada no microcontrolador ATMEGA.

Tabela 8: Caso de uso – "Ligar Arduino"

Nome do Caso de Uso	Ligar Arduino
Ator Principal	Usuário
Atores Secundários	Sensores
Resumo	Este caso de uso descreve como o usuário poderá **ligar o dispositivo Arduino** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga do Arduino na posição "Ligado"	
	2. Energizar o Arduino
	3. Energizar Componentes Eletrônicos
	4 Emitir Notas musicais de Aviso
Restrições/Validações	Deve possuir carga suficiente na Bateria

Fonte**:** próprio autor

A tabela a seguir tem como objetivo, elucidar passo a passo o desenvolvimento da lógica de como é o desligamento do hardware composto pelo Arduino e seus sensores, este desligamento é feito através do acionamento de uma chave liga/desliga.

Tabela 9: Caso de Uso - "Desligar Arduino"

Nome do Caso de Uso	Desligar Arduino
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá **desligar o dispositivo Arduino** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga na posição “Desligado”	
	2. Corta totalmente o suprimento de energia para todo o sistema, desligando completamente o Arduino e os sensores.
Restrições/Validações	1. O Arduino tem que estar ligado
	2. Deve possuir carga o suficiente na Bateria

Fonte: próprio autor

Esta tabela tem como objetivo, elucidar passo a passo o desenvolvimento da lógica de como são ligados o *Buzzer* e as restrições necessárias para que ele desempenhe a sua função que é emitir sons para alertar o usuário da presença de obstáculos.

Tabela 10**:** Caso de Uso - "Ligar *Buzzer*"

Nome do Caso de Uso	Ligar *Buzzer*
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá **ligar o *Buzzer*** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga do *Buzzer* na posição “Ligado”	
	2. Habilita porta digital número 13 do Arduino a receber 5 Volts em nível Alto (*HIGH*)

Restrições/Validações	1. O Arduino tem que estar ligado
	2. Deve possuir carga o suficiente na Bateria

Fonte: próprio autor

A tabela 11 mostra, a sequência de eventos para que o usuário consiga desligar o *Buzzer* acionando uma chave liga/desliga no painel do dispositivo Arduino e as condições mínimas para que isso seja possível.

Tabela 11**:**Caso - "Desligar o *Buzzer*"

Nome do Caso de Uso	Desligar *Buzzer*
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá **desligar o *Buzzer*** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga do *Buzzer* na posição "Desligado"	
	2. Corta a energia na porta digital número 13 do Arduino impossibilitando o acionamento do *Buzzer*
Restrições/Validações	1. O Arduino tem que estar ligado
	2. Deve possuir carga o suficiente na Bateria

Fonte: próprio autor

A seguir é mostrado em forma de tabela o processo capaz de permitir com que o motor de vibração opere e venha a auxiliar o deficiente visual em seu trajeto informando por meio de vibrações os obstáculos físicos à frente, conforme diminui a distância entre o dispositivo Arduino e o obstáculo, o motor vibrará com maior intensidade.

Tabela 12: Caso de Uso - "Ligar motor de Vibração

Nome do Caso de Uso	Ligar Motor de Vibração
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá **Ligar o Motor de Vibração** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga do Motor na posição “Ligado”	
	2. Habilita porta digital número 10 do Arduino a receber 5 Volts em nível Alto (*HIGH*)
Restrições/Validações	1. O Arduino tem que estar ligado
	2. Deve possuir carga o suficiente na Bateria

Fonte: próprio autor

A tabela mostra o processo de desligamento do motor de vibração, podendo utilizar como alternativa o B*uzzer* e a interação por voz do aplicativo para Android.

Tabela 13:Caso de uso "Desligar Motor de Vibração"

Nome do Caso de Uso	Ligar/Desligar Motor de Vibração
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá **Desligar o Motor de Vibração** através de uma chave liga/desliga
Ações do Ator	**Ações do Sistema**
1. O usuário aciona manualmente a chave Liga/Desliga do Motor na posição “Desligado”	
	2. . Corta a energia na porta digital número 10 do Arduino impossibilitando o acionamento

	do Motor
Restrições/Validações	1. O Arduino tem que estar ligado
	2. Deve possuir carga o suficiente na Bateria

Fonte: próprio autor

A tabela a seguir mostra as etapas que envolvem a operação de “Enviar Dados” através do módulo HC-05 *Bluetooth*, estes dados são referentes as medidas da distância dos obstáculos obtidas pelo sensor de ultrassom.

Tabela 14:Caso de Uso - "Enviar Dados"

Nome do Caso de Uso	Enviar Dados
Ator Principal	Módulo *Bluetooth* HC-05
Atores Secundários	1. Sensor ultrassônico
	2. Dispositivo *Android*
Resumo	Este caso de uso descreve como o módulo de Bluetooth irá enviar dados para o aplicativo Android
Ações do Ator	**Ações do Sistema**
	1. O módulo emissor do sensor ultrassônico emite 8 pulsos de 40 KHz
	2. O sensor ultrassônico coloca o pino *ECHO* em nível alto aguardando o retorno do som iniciando assim a contagem de tempo do retorno
	3. Quando o som retornar para o módulo receptor do sensor ultrassônico, o pino *ECHO* é colocado em nível baixo e finaliza-se a contagem do tempo decorrido
	4. Efetua-se o cálculo da distância do obstáculo em centímetros com base no tempo de ida e volta do som e da velocidade do som em nossa atmosfera: **Velocidade do som** = 340m/s **Distância** = [Tempo *ECHO* em nível alto * Velocidade do Som] / 2 * 100
	5. O Arduino irá checar a disponibilidade do *Bluetooth*
6. Se disponível, o módulo Bluetooth	

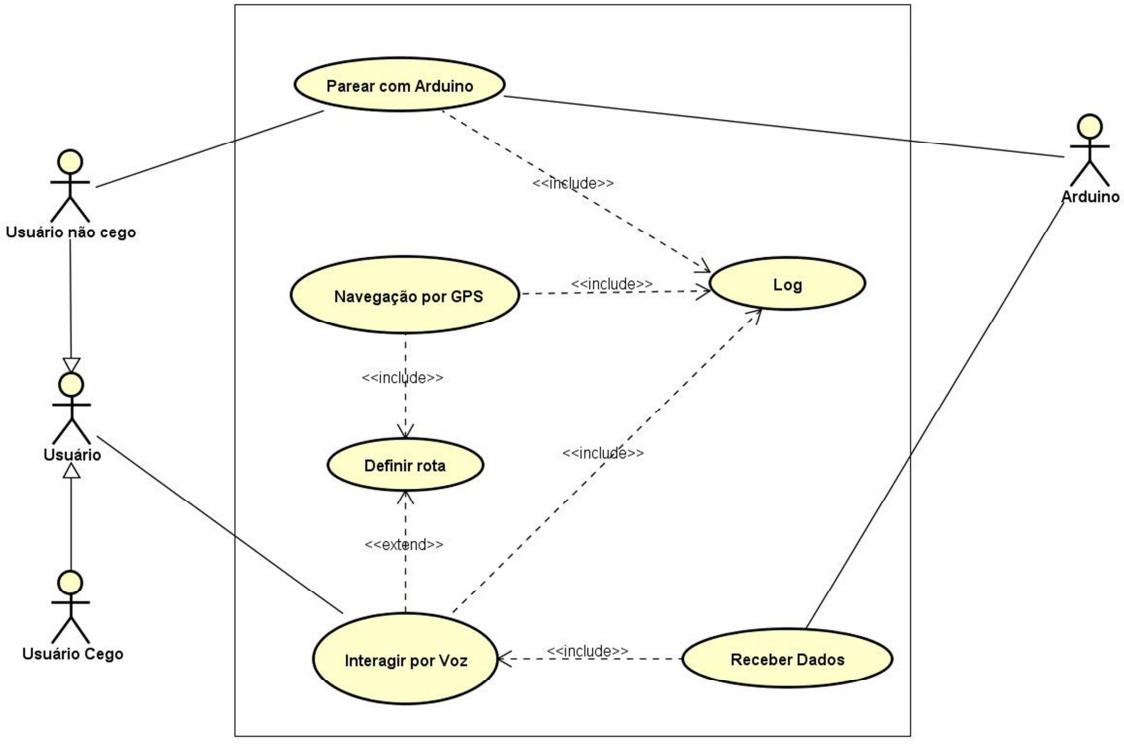
repassa estes dados para o dispositivo Android previamente pareado	
Restrições/Validações	1. O Arduino tem que estar ligado
	2. O Arduino tem que estar pareado com o dispositivo Android
	3. Deve ter carga suficiente na bateria

Fonte: próprio autor

4.2.3. Diagrama de Caso de Uso - Android

Este diagrama tem por objetivo descrever graficamente as diversas interações entre os atores do cenário que é o aplicativo para *smartphone Android*, as rotinas internas inerentes a lógica do funcionamento do aplicativo e o papel de cada ator no contexto da aplicação.

Figura 31:Caso de Uso - Android



Fonte: próprio autor

4.2.4. Documentação do Caso de Uso – Android

Esta tabela 15 exemplifica a tarefa de pareamento com o dispositivo Arduino, esta tarefa poderá ser realizada por intermédio do usuário não cego na primeira execução da aplicação, ficando em seguida este pareamento realizado de forma automática no carregamento do aplicativo.

Tabela 15:Caso de Uso - "Parear com Arduino"

Nome do Caso de Uso	Parear com Arduino
Ator Principal	Usuário Não Cego
Atores Secundários	Arduino
Resumo	Este caso de uso descreve como o usuário não cego poderá auxiliar na tarefa do primeiro pareamento do Android com o Arduino
Ações do Ator	**Ações do Sistema**
1. O usuário não cego irá ativar o Bluetooth	
2. Usuário não cego acessa menu de "Conexões sem fio e rede" do smartphone com Android	
1. Seleciona "Configurações de *Bluetooth*"	
2. Pesquisa dispositivos para encontrar o Arduino "HC-05"	
3. Seleciona o dispositivo "HC-05"	
4. Introduz a senha padrão "1234"	
	5. Gerar Log
	6. O Aplicativo Android começa a receber a distância dos obstáculos
Restrições/Validações	1. O Arduino tem que estar ligado
	2. O Arduino tem que ter carga disponível na Bateria

Fonte: próprio autor

Esta tabela tem por finalidade mostrar a sequência lógica do processo da interação por voz com o aplicativo *Android*, esta interação poderá ser realizada por

qualquer usuário do sistema e será uma comunicação de duas vias, ora dando instruções para a aplicação, ora o aplicativo irá falar com o usuário.

Tabela 16:Caso de Uso - "Interagir por Voz"

Nome do Caso de Uso	Interagir por Voz
Ator Principal	Usuário
Atores Secundários	Arduino
Resumo	Este caso de uso descreve como o usuário poderá interagir por **comando de voz** com o dispositivo Android
Ações do Ator	**Ações do Sistema**
	1. A aplicação aguarda por instruções
2. O usuário fala uma instrução	
	3. A instrução será submetida a validação
	4. Se é uma instrução válida:
	5. Carregar caso de uso “Definir rota”
	6. Se não for uma instrução válida:
	7. Alertar usuário por voz
Restrições/Validações	1. Deve ter carga suficiente na bateria
	2. O *GPS* deve estar habilitado

Fonte: próprio autor

Esta tabela 17 tem por finalidade mostrar a sequência lógica do processo da interação por voz com o aplicativo Android que irá falar para o usuário a distância dos obstáculos detectados pelo sensor ultrassônico do Arduino.

Tabela 17:Caso de Uso "Interagir por Voz"

Nome do Caso de Uso	Interagir por Voz
Ator Principal	Usuário
Atores Secundários	Arduino

	Aplicativo Android
Resumo	Este caso de uso descreve como o usuário poderá **receber alertas de voz** dos obstáculos detectados à frente.
Ações do Ator	**Ações do Sistema**
	1. A aplicação Android recebe do Arduino a distância dos obstáculos em centímetros.
	2. Distâncias menores que 1 metro serão mantidas em centímetros, maiores ou iguais a 1 metro serão convertidas para metro.
	3. A aplicação Android irá sintetizar em forma de voz a distância dos obstáculos.
	4. Gerar Log de eventos
Restrições/Validações	1. Deve ter carga suficiente na bateria
	2. Arduino e Android devem estar pareados

Fonte: próprio autor

A tabela 18 mostra a sequência lógica de como o aplicativo Android irá definir a rota de navegação através do destino falado pelo usuário.

Tabela 18:Caso de Uso - "Definir Rota"

Nome do Caso de Uso	Definir Rota
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá definir as rotas para navegação
Ações do Ator	**Ações do Sistema**
	1. Verificar se o *GPS* está habilitado
	2. Se não habilitado:
	3. Alertar usuário por voz
	4. Habilitar *GPS*
	5. Se habilitado:

	6. Obter através do *GPS* as coordenadas latitude e longitude atuais do usuário
	7. Converter de voz para texto o destino informado pelo usuário
	8. Verificar se o destino existe
	9. Se o destino não existir:
	10. Alertar usuário por voz
	11. Se o destino existir:
	12. Obter através do *GPS* as coordenadas de latitude e longitude do destino
	13. Traçar a Rota de destino
	14. Gerar Log
Restrições/Validações	1. O Smartphone deve ter carga suficiente na bateria
	2. O aplicativo deve ter permissão para acessar o *GPS*

Fonte: próprio autor

Esta tabela 19 descreve a lógica de funcionamento da rotina de navegação por *GPS*, as restrições para que a operação ocorra e os mecanismos internos do Android para habilitar a navegação.

Tabela 19:Caso de Uso - "Navegação por GPS"

Nome do Caso de Uso	Navegação por *GPS*
Ator Principal	Usuário
Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como o usuário poderá iniciar a navegação por *GPS*
Ações do Ator	**Ações do Sistema**
	1. Obter dados da rota do caso de uso “Definir Rota”
	2. Obter através do *GPS* as coordenadas latitude e longitude atuais do usuário
	3.
	4.
	5.
	6. Carregar caso de uso LOG
Restrições/Validações	1. O Aplicativo deve ter permissão para usar o *GPS*
	2. O *GPS* do Smartphone deve estar habilitado
	3. O Smartphone deve ter carga suficiente na bateria

Fonte: próprio autor

A tabela 20 a seguir mostra o processo de geração de logs do sistema registrando as interações do usuário e eventos que poderão acontecer no decorrer do uso do aplicativo.

Tabela 20:Caso de Uso - "Log"

Nome do Caso de Uso	Log
Ator Principal	1. Usuário

Atores Secundários	Nenhum
Resumo	Este caso de uso descreve como sistema gera os logs das atividades do aplicativo
Ações do Ator	**Ações do Sistema**
	1. Algum caso de uso chama a rotina de LOG
	2. Criar arquivo no formato JSON
	3. Gravar dados no arquivo
	4. Mostrar na tela o progresso do Log
Restrições/Validações	O Smartphone deve ter carga suficiente na bateria

Fonte: próprio autor

A tabela 21 mostra os passos da rotina de recebimento de dados vindos do dispositivo Arduino através da comunicação serial via *Bluetooth*.

Tabela 21:Caso de Uso - "Receber Dados"

Nome do Caso de Uso	Receber Dados
Ator Principal	Arduino
Atores Secundários	Usuário
Resumo	Este caso de uso descreve como é a interação entre Arduino e usuário no recebimento dos dados referente aos obstáculos detectados
Ações do Ator	**Ações do Sistema**
1. Checar disponibilidade do *Bluetooth*	
2. Enviar dados da distância para o Android	
	3. Recebe os dados
	4. Carrega caso de uso “Interagir por voz”
Restrições/Validações	1. O Arduino tem que estar ligado
	2. O Arduino tem que estar pareado
	3. O Arduino deve ter carga suficiente na bateria
	4. O Smartphone deve estar ligado

	5. O Smartphone deve ter carga suficiente na bateria

Fonte: próprio autor

4.3. *Mock-Up* do Aplicativo

Mock-up é uma forma de representar graficamente o design de um produto ou aplicativo, mostrando suas principais funcionalidades de uma forma estática, podendo ser considerado como um rascunho do projeto final. É bastante utilizado como forma de vender a ideia do produto antes dele mesmo estar pronto para o seu público estratégico (RUTE, 2014).

O *Mock-up* a seguir mostra detalhes da tela principal do aplicativo, na parte superior destacam-se dois ícones que mostram o *status* do *Bluetooth* e *GPs* do smartphone, quando estes recursos estão ativados ficam na coloração verde, quando estão em amarelo ocorre a tentativa de ativar estes recursos, quando estão em vermelho os recursos não estão disponíveis.

A região inferior da tela é destinada a visualização dos eventos decorrentes da interação do usuário através da interação por voz, detecção de obstáculos, navegação por *GPS* e pareamento com o *Bluetooth*.

Na região central, pode-se visualizar a distância dos obstáculos detectados à frente em metros ou centímetros dependendo da proximidade do obstáculo, também pode-se observar o status do que o aplicativo está fazendo no momento, seja aguardando instruções de voz, pareando com *bluetooth*, ativando *GPS*, dentre outros.

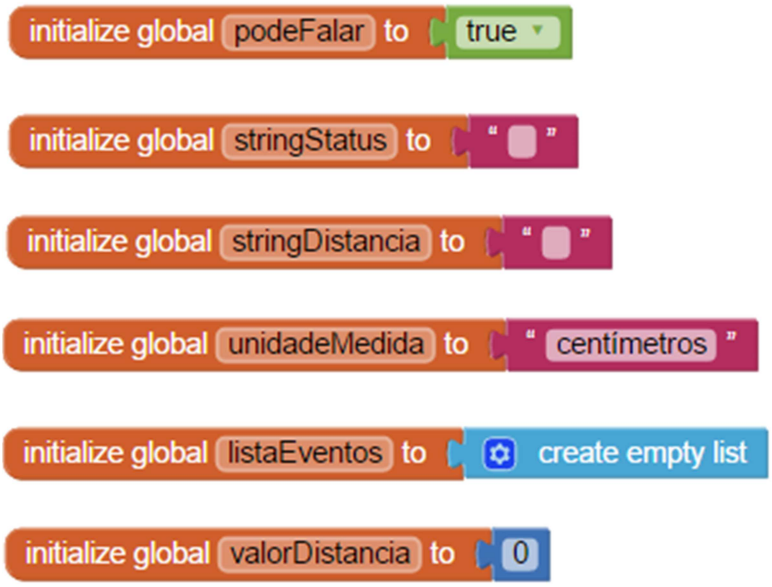
Também na região central da tela, está localizado um ícone representando um microfone que mostra atividade da interação com o usuário quando o aplicativo recebe uma instrução por voz.

Figura 32:Tela Principal

Fonte: próprio autor

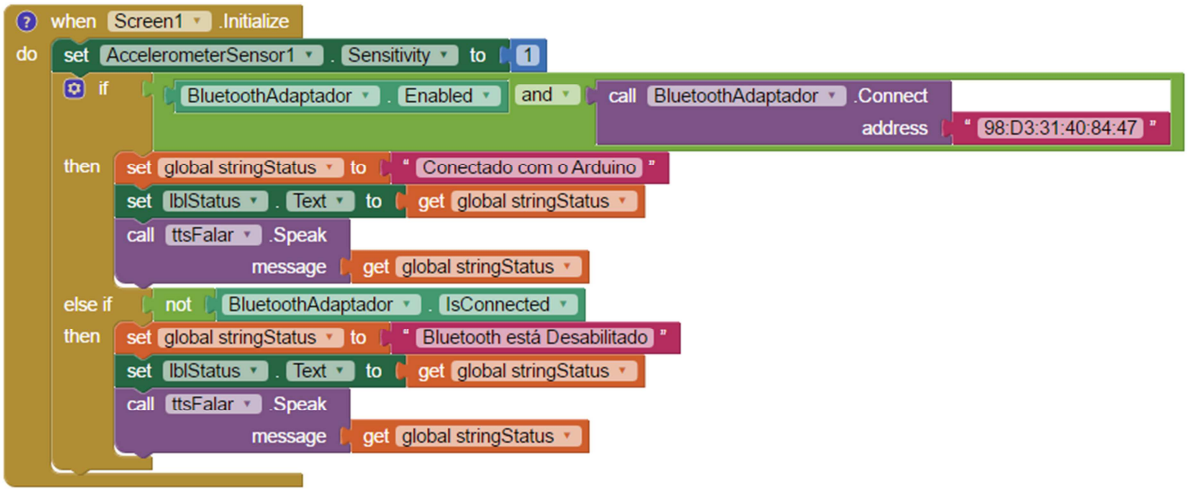
4.4. Implementação no App Inventor

Estes blocos são responsáveis pela inicialização das variáveis globais do sistema, onde temos a variável “podeFalar” responsável por habilitar ou desabilitar a fala do sintetizador de voz quando sacudirmos o aparelho, a variável “stringStatus” que terá a situação atual do comportamento do aparelho, como a perda de conexão com o *bluetooth*, a variável “stringDistancia” que irá concatenar as medidas da distância vindas do Arduino (‘valorDistancia”) com a palavra “centímetros” ou “metro” (“unidadeMedida”) de acordo com a distância do obstáculo e depois será armazenada na lista de log de eventos do sistema (“listaEventos”).

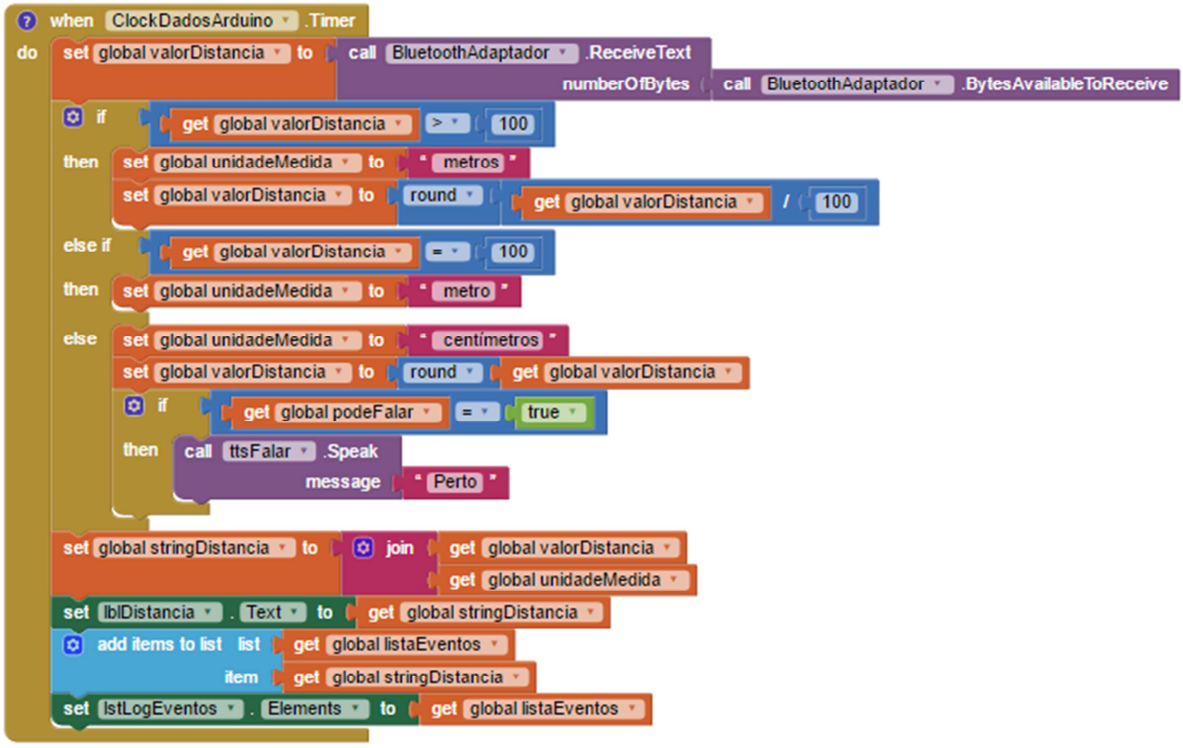


O bloco lógico a seguir é responsável pelo comportamento do aplicativo em seu carregamento inicial, nele é possível observar a propriedade “*Initialize*” que trata o evento da inicialização da tela “*Screen1*” que é a tela principal da aplicação. Ao iniciar a tela “*Screen1*”, a sensibilidade do acelerômetro é configurada em “1” que é o valor menos sensível possível em uma escala que vai de 1 a 10, onde 1 é o menos sensível e 10 uma sensibilidade máxima, foi determinada uma sensibilidade baixa do acelerômetro pois movimentos involuntários realizados pelo usuário poderiam disparar o evento do reconhecimento de voz. O acelerômetro será utilizado para quando o usuário “sacudir” o aparelho este chame o recurso de reconhecimento de voz do Android para passar comandos para a aplicação. Estes comandos serão para traçar uma rota de destino através do *GPS* do aparelho, baseando-se na posição de latitude e longitude atuais do usuário e baseados no destino que o mesmo irá falar

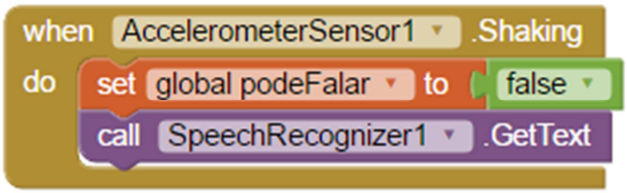
para a aplicação, é possível traçar esta rota e auxiliar o deficiente visual em sua caminhada rumo ao destino informado. Em seguida temos um bloco "*If else*" que verifica se o *bluetooth* do aparelho está disponível e tenta conectar no dispositivo que possui o Mac *Address* informado com parâmetro da procedure "*call BluetoothAdaptador.ConnectAddress*", o endereço MAC informado é o do dispositivo Arduino.



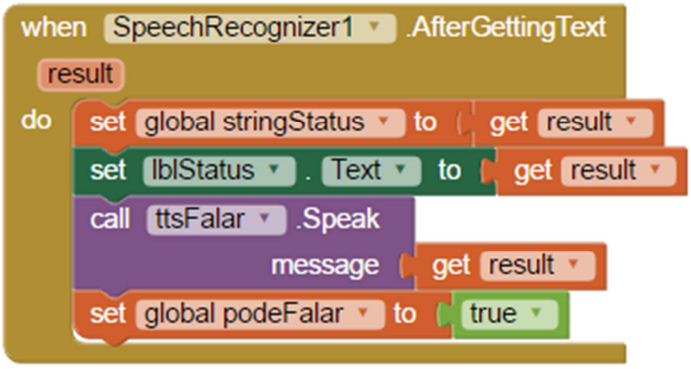
O próximo bloco controla um "*timer*" que é invocado em um intervalo pré-determinado, a cada execução, este timer irá checar as medidas vindas do ultrassom do Arduino e falar se o obstáculo está perto do usuário, montar a *string* contendo as medidas e alimentar uma lista que contém o valor destas medidas em centímetros ou metros dependendo da distância.



Em seguida, este bloco chama o "*SpeechRecognizer*" que é o comando de voz do Android para que o usuário possa passar instruções ao aplicativo.



Este próximo bloco que é responsável por pegar o texto informado por comando de voz e atribuir para a variável de status, evidenciando o comando que foi passado pelo usuário, através deste texto é possível atribuir o destino para rotina de criação das rotas de destino do *GPS*.



4.5. Experimentos e Resultados

Os experimentos começaram com testes de execução de cada componente eletrônico conectado a placa Arduino, um por um, foram sendo montados na protoboard e programados e os resultados obtidos de maneira satisfatória seguindo o manual de montagem de cada fabricante dos componentes. No caso do sensor de ultrassom por exemplo, só foi possível visualizar seus dados obtidos por meio de logs gerados pela *IDE* do Arduino, uma vez que ainda não tinha montado o display de *LCD* nem desenvolvido o aplicativo para Android para visualizar estas informações. O componente de maior complexidade na montagem foi o display de *LCD*, pelo seu grande número de terminais e pelo uso de um potenciômetro que controla o contraste do visor. A implementação do Arduino ia crescendo à medida que os outros componentes foram agregados ao projeto.

Quanto ao desenvolvimento do aplicativo mobile, os testes foram executados primeiramente fazendo a checagem da disponibilidade do *bluetooth* no aparelho e em seguida realizando o pareamento e obtenção dos dados vindos do Arduino, posteriormente já com estes dados, o passo seguinte era mostrar estas informações na tela para checar se o que sai do Arduino e o que chega na aplicação mobile são coerentes. Foi implementado um recurso de “Sacudir” o aparelho para acionar o reconhecimento de voz que transforma em texto o que o usuário falou, isto pode ser usado futuramente para a definição de trajetos trabalhando em conjunto com o *GPS* do aparelho.

5. CONSIDERAÇÕES FINAIS

Este trabalho teve como principal objetivo, desenvolver uma aplicação de baixo custo que fosse útil para os portadores de algum tipo de deficiência visual, auxiliando-os em seu trajeto diário gerando alertas sonoros e vibratórios sobre obstáculos físicos que estivessem em seu caminho. Esta aplicação basicamente, consiste em dois módulos, um primeiro módulo, que foi construído com componentes eletrônicos agregados em uma placa de prototipagem chamada Arduino e um segundo módulo que foi o desenvolvimento de um aplicativo mobile que trabalhasse em conjunto com o primeiro módulo eletrônico.

Para que isso fosse possível, o trabalho inicialmente explorou os conceitos relacionados aos tipos de deficiências existentes, dentre elas a deficiência visual, que é a principal em questão neste trabalho, empreendeu-se também todos os conceitos teóricos envolvidos no funcionamento de cada componente eletrônico que foi utilizado no projeto, demonstrou-se protótipos com soluções realizadas por iniciativa de alunos de outras instituições de ensino como a luva eletrônica e a bengala eletrônica que também propõem o bem estar e diminuir as dificuldades diárias dos portadores de deficiência visual.

Na sequência, foram apresentadas as etapas de desenvolvimento do projeto físico, inicialmente focando na montagem dos componentes na placa de prototipagem, na programação de cada um dos componentes e o teste de cada componente obtendo assim o comportamento e o resultado esperado, nesta fase a maior dificuldade foi em entender o processo de montagem dos componentes, ligações em série e paralelo, divisores de tensão, o funcionamento de um potenciômetro, as ligações dos pinos do display de LCD, são detalhes mais técnicos que ao menor erro de montagem podem gerar uma dificuldade em saber onde está o defeito, gerando assim tempo em revisar tudo o que foi feito na protoboard para encontrar onde foi feita a ligação errada na placa. A etapa seguinte, foi o desenvolvimento do aplicativo mobile, esta foi a etapa que mais apresentou dificuldades no projeto, pois esta iria fazer a integração das duas partes, *hardware* e *software*, portanto encontrar uma tecnologia simples e de fácil entendimento e que focasse diretamente no problema proposto consumiu muito tempo em testes com diversas linguagens.

O desafio de desenvolver um aplicativo que possibilite a comunicação via *Bluetooth* com a plataforma Arduino foi superado com este trabalho. Seu desenvolvimento mostrou-se alinhado com os propósitos do projeto e com grande possibilidade de expansão futura.

Acredito que o projeto obteve êxito satisfatório desde o levantamento dos requisitos e estudo de caso até a montagem final dos componentes eletrônicos e no desenvolvimento do aplicativo mobile, sendo uma ótima fonte de estudos para trabalhos futuros de outros discentes da instituição.

5.1. Trabalhos Futuros

Uma proposta de melhoria futura é fazer com que o aplicativo trabalhe em conjunto com o Google *Maps*, pegando o destino informado pelo usuário através do comando de voz e passando como parâmetro para a *API* do Google responsável pela geolocalização do aparelho. Este recurso poderá ser implementado criando uma interface de orientação para o usuário no decorrer de todo o trajeto que ele informou para o aplicativo. Uma outra melhoria seria na parte de hardware, acondicionando todo o circuito eletrônico em um invólucro plástico dando uma cara mais profissional ao projeto, existem no mercado caixas plásticas que podem sem utilizadas para esta finalidade. Com relação ao design do aplicativo, minha falta de experiência e domínio em harmonia de cores e heurística de interface com o usuário tornam a aplicação pouco agradável visualmente falando, uma proposta de melhoria futura é trabalhar no enriquecimento visual do aplicativo estudando técnicas de interface com o usuário mesmo sendo um aplicativo para deficientes visuais.

6. REFERÊNCIAS

BRUNA BREDARIOL. Acessibilidade de deficientes visuais à prática da natação: uma revisão de literatura. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=000806319&opt=4,2010. > Acesso em: 7 de setembro de 2016.

CARDOZO BUENO, Alessandro. Bengala Eletrônica para Deficientes Visuais. 2010. (Graduação em Engenharia da Computação) – Universidade Positivo Núcleo de Ciências Exatas e Tecnológicas, Curitiba, 2010. Disponível em: <http://www.up.edu.br/blogs/engenharia-da-computacao/wp-content/uploads/sites/6/2015/06/2010.1.pdf> Acesso em: 15 de março de 2017.

DAVID SHALOM. Mais de 2,2 mil aguardam cão-guia no País, que tem 100 animais em atividade - Brasil - iG. Último Segundo. Notícias. Disponível em: <http://ultimosegundo.ig.com.br/brasil/2014-10-09/mais-de-22-mil-aguardam-cao-guia-no-pais-que-tem-100-animais-em-atividade.html, 2014, outubro 9> Acesso em: 7 de setembro de 2016.

Eletrônica Didática. Disponível em: <http://www.eletronicadidatica.com.br/protoboard.html> Acesso em: 19 de novembro de 2015.

EMERSON ALECRIM. Tecnologia Bluetooth: o que é e como funciona? Tecnologia Bluetooth: o que é e como funciona? Disponível em: <http://www.infowester.com/bluetooth.php, 2013, março 09> Acesso em: 20 de setembro de 2016.

EVANS, Martin; NOBLE Joshua; HOCHENBAUM, Jordan. Arduino em Ação. Trad. Camila Pauan. São Paulo: Novatec, 2013. 424 p.

ISL INFORMATION BLOG. Are Electromagnetic Or Piezoelectric Buzzer For You?. Disponível em: <http://www.islproducts.com/isl-blog/entry/are-electromagnetic-or-piezoelectric-buzzers-for-you.html> Acesso em: 21 de março de 2017.

MIT App Inventor, Anyone Can Build Apps That Impact the World. Disponível em: <http://appinventor.mit.edu/explore/about-us.html/> Acesso em: 01 de junho de 2017.

PAUL SULLIVAN. Com custo de até R$ 138 mil, treinar cães-guia é desafio para ONGs nos EUA - New York Times - iG. Com custo de até R$ 138 mil, treinar cães-guia é desafio para ONGs nos EUA. Notícias. Disponível em: <http://ultimosegundo.ig.com.br/mundo/nyt/2013-11-10/com-custo-de-ate-r-138-mil-treinar-caes-guia-e-desafio-para-ongs-nos-eua.html, 2013, novembro 10> Acesso em: 7 de setembro de 2016.

PRECISION MICRODRIVES. AB-004: UNDERSTANDING ERM VIBRATION MOTOR CHARACTERISTICS. Disponível em: <https://www.precisionmicrodrives.com/application-notes/ab-004-understanding-erm-vibration-motor-characteristics> Acesso em: 15 de março de 2017.

PRESSMAN, Roger S. Engenharia de Software: Uma Abordagem Profissional, Sétima Edição. Editora MCGrawHill: Porto Alegre, 2011.

Projeto Cão Guia, Disponível em:<http://pessoas.hsw.uol.com.br/caes-guia1.htm> Acesso em: 23 de setembro de 2016.

RUTE, Ana. Wireframe, protótipo e mockup – Qual a diferença? 18 de abril de 2014. Disponível em: <https://anarute.com/wireframe-prototipo-e-mockup-qual-a-diferenca/> Acesso em: 25 de abril de 2017.

SCHIRMER, Camila Lopes. et al.: Luva guia para deficientes visuais. Scientia Prima, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, v. 3, n. 3, dez. 2015. Disponível em: <http://scientiaprima.incentivoaciencia.com.br/2016/02/29/leia-o-3o-volume-do-scientia-prima/> Acesso em: 15 de março de 2017.

APÊNDICE A – CÓDIGO FONTE ARDUINO

```
/*******************************************************
    Projeto TCC - Acessibilidade visual com Arduino
    ----------------------------------
    $Author: Elton Lima da Siva
    $Date: 2017-05-30 16:40
    $Revision: 5.1
    ----------------------------------
*******************************************************/

#include <SoftwareSerial.h> // Carrega a biblioteca para comunic. Serial
#include <LiquidCrystal.h> // Carrega Biblioteca de controle do LCD

/*****************************************
   Definicoes das Constantes do sistema
*****************************************/

// Pinos para o trigger e echo do ultrassom
#define TRIGGER 8
#define ECHO    9
#define BUZZER 13
#define MOTOR  10

// Pinagem de comunicacao RX/TX Bluetooth
#define RX_BT 0
#define TX_BT 1

// Inicializa bluetooth nos pinos 0 e 1
SoftwareSerial bt(RX_BT, TX_BT);

// Inicializa lcd na pinagem abaixo
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
    // RS  - Pino 12
    // R/W - Pino GND
    // E   - Pino 11
    // DB4 - Pino 5
    // DB5 - Pino 4
    // DB6 - Pino 3
    // DB7 - Pino 2

/*****************************************
*           Variáveis Globais            *
*****************************************/
// Medidas em cm
int alcanceMaximo = 400;
int alcanceMinimo = 3;

float duracao = 0.0;
float distancia = 0.0;
char msgDistancia[5];

//melodia do MARIO THEME
// http://www.nubiasouza.com.br/musica-tema-mario-arduino-speaker/
int melodia[] = {660,660,660,510,660,770,380,510,380,320,440,480,450,430,380,660,760,860,
                 700,760,660,520,580,480,510,380,320,440,480,450,430,380,660,760,860,700,
                 760,660,520,580,480,500,760,720,680,620,650,380,430,500,430,500,570,500,
                 760,720,680,620,650,1020,1020,1020,380,500,760,720,680,620,650,380,430,
                 500,430,500,570,585,550,500,380,500,500,500,500,760,720,680,620,650,380,
                 430,500,430,500,570,500,760,720,680,620,650,1020,1020,1020,380,500,760,
                 720,680,620,650,380,430,500,430,500,570,585,550,500,380,500,500,500,500,
                 500,500,500,580,660,500,430,380,500,500,500,500,580,660,870,760,500,500,
                 500,500,580,660,500,430,380,660,660,660,510,660,770,380};

```

```
63   //duraçao de cada nota
64   int duracaodasnotas[] = {100,100,100,100,100,100,100,100,100,100,100,80,100,100,100,80,50,100,
65                            80,50,80,80,80,80,100,100,100,100,80,100,100,100,80,50,100,80,50,80,
66                            80,80,80,100,100,100,100,150,150,100,100,100,100,100,100,100,100,100,
67                            100,150,200,80,80,80,100,100,100,100,100,150,150,100,100,100,100,100,
68                            100,100,100,100,100,100,100,100,100,100,100,100,150,150,100,100,100,
69                            100,100,100,100,100,100,100,150,200,80,80,80,100,100,100,100,100,150,
70                            150,100,100,100,100,100,100,100,100,100,100,100,100,100,100,60,80,60,80,
71                            80,80,80,80,80,60,80,60,80,80,80,80,80,60,80,60,80,80,80,80,80,80,100,
72                            100,100,100,100,100,100};
73
74   // pausa depois das notas
75   int pausadepoisdasnotas[] = {150,300,300,100,300,550,575,450,400,500,300,330,150,300,200,200,150,
76                                300,150,350,300,150,150,500,450,400,500,300,330,150,300,200,200,150,
77                                300,150,350,300,150,150,500,300,100,150,150,300,300,150,150,300,150,
78                                100,220,300,100,150,150,300,300,300,150,300,300,300,100,150,150,300,
79                                300,150,150,300,150,100,420,450,420,360,300,300,150,300,300,100,150,
80                                150,300,300,150,150,300,150,100,220,300,100,150,150,300,300,300,150,
81                                300,300,300,100,150,150,300,300,150,150,300,150,100,420,450,420,360,
82                                300,300,150,300,150,300,350,150,350,150,300,150,600,150,300,350,150,
83                                150,550,325,600,150,300,350,150,350,150,300,150,600,150,300,300,100,
84                                300,550,575};
```

```
85
86   /*****************************************
87   *            Rotinas Utilitarias           *
88   *****************************************/
89
90   // Envia para o smartphone uma string com a distancia
91   void btEnviarMsg() {
92       bt.println(distancia);
93       delay(2000);
94   }
95
96   // Aciona o vibracall em milisegundos
97   void AcionaMotor(int tempoAcionado, int quantidade, int tempoParado) {
98       for (int i = 1; i <= quantidade; i++) {
99           digitalWrite(MOTOR, HIGH);
100          delay(tempoAcionado);
101          digitalWrite(MOTOR, LOW);
102          delay(tempoParado);
103      }
104  }
105
```

```
106   // Emite 4 bips para objeto curtíssima distância
107   // inferior a 50 centímetos
108   void BeepMuitoPerto() {
109       tone(BUZZER, 440);
110       delay(50);
111       noTone(BUZZER);
112       delay(100);
113       tone(BUZZER, 200);
114       delay(50);
115       noTone(BUZZER);
116       delay(100);
117       tone(BUZZER, 100);
118       delay(50);
119       noTone(BUZZER);
120       delay(100);
121       tone(BUZZER, 150);
122       delay(50);
123       noTone(BUZZER);
124       delay(100);
125   }
126
```

```
127   // Emite 3 bips para objeto muito proximo
128   // maior que 50 centimetros até 1 metro
129   void BeepPerto() {
130       tone(BUZZER, 440);
131       delay(50);
132       noTone(BUZZER);
133       delay(100);
134       tone(BUZZER, 200);
135       delay(50);
136       noTone(BUZZER);
137       delay(100);
138       tone(BUZZER, 100);
139       delay(50);
140       noTone(BUZZER);
141       delay(100);
142   }
143
144   // Emite 2 bips para objeto a media distancia
145   // entre 1 e 2 metros
146   void BeepMedio() {
147       tone(BUZZER, 440);
148       delay(50);
149       noTone(BUZZER);
150       delay(100);
151       tone(BUZZER, 200);
152       delay(50);
153       noTone(BUZZER);
154       delay(100);
155   }
```

```
156
157     // Emite 1 bip para objeto longe
158     // entre 2 e 3 metros
159     void BeepLonge() {
160         tone(BUZZER, 440);
161         delay(50);
162         noTone(BUZZER);
163         delay(100);
164     }
165
166
167     /*****************************************
168     *      Rotina de configuraço do Arduino    *
169     *****************************************/
170     void setup() {
171         pinMode(BUZZER, OUTPUT);
172         pinMode(MOTOR, OUTPUT);
173         pinMode(TRIGGER, OUTPUT);
174         pinMode(ECHO, INPUT);
175
176         // Inicializa Bluetooth com 9600 bits por segundo
177         bt.begin(9600);
178
179         // Inicializa LCD com 16 colunas x 2 linhas
180         lcd.begin(16, 2);
181
182         // Acionamento inicial do Buzzer
183         lcd.clear();
184         lcd.setCursor(0, 0);
185         lcd.print("Buzzer...");
```

```
186
187         // Tocar Tema do Super Mario Bros avisando que o dispositivo foi ligado
188         //for para tocar as 7 primeiras notas começando no 0 ate 7
189         for (int nota = 0; nota < 7; nota++) {
190             int duracaodanota = duracaodasnotas[nota];
191             tone(BUZZER, melodia[nota], duracaodanota);
192
193             //pausa depois das notas
194             delay(pausadepoisdasnotas[nota]);
195         }
196         noTone(BUZZER);
197         lcd.print(" - Ok");
198         delay(1000);
199
200         // Acionamento inicial do Motor de vibração
201         lcd.clear();
202         lcd.setCursor(0, 0);
203         lcd.print("Motor...");
204         AcionaMotor(1000, 1, 0);
205         lcd.print(" - Ok");
206         delay(1000);
207     }
208
```

```
208
209 /*****************************
210  *      Rotina principal     *
211  *****************************/
212 void loop() {
213     // Prepara o "gatilho" de disparo do ultrassom
214     digitalWrite(TRIGGER, LOW);
215     delayMicroseconds(2);
216
217     // dispara o feixe sonoro
218     digitalWrite(TRIGGER, HIGH);
219     delayMicroseconds(10); // tempo para disparar 8 feixes de 40Khz
220     digitalWrite(TRIGGER, LOW);
221
222     // tempo em microsegundos do pino ECHO em nivel alto aguardando o retorno do som
223     duracao = pulseIn(ECHO, HIGH); // aguarda o retorno
224
225     // 29.2 microsegundos para percorrer 1 cm => ( 1 / 340 = 0,0029411764705882 )
226     distancia = (duracao/2) / 29.2 ; // Calcula a distância
227
228     // Envia para o smartphone uma string com a distancia em centímetros
229     btEnviarMsg();
230
```

```
231     /***********************************
232     * Controlando Buzzer, LCD e Motor  *
233     ***********************************/
234
235     // Distância em Centímetos
236     if (distancia < 100) {
237         lcd.clear();
238         lcd.setCursor(0, 0);
239         lcd.print("Centimetros: ");
240         lcd.setCursor(0, 1);
241         lcd.print(round(distancia));
242
243         if (distancia < 50) {
244             AcionaMotor(1000, 1, 0);
245             BeepMuitoPerto();
246             delay(250);
247         }
248         else {
249             AcionaMotor(100, 3, 100);
250             BeepPerto();
251             delay(250);
252         }
253     }
```

```
254
255         // Distância em metros
256         else {
257             lcd.clear();
258             lcd.setCursor(0, 0);
259             lcd.print("Metros: ");
260             lcd.setCursor(0, 1);
261             lcd.print(distancia/100);
262
263             if (distancia >= 100 && distancia < 200) {
264                 AcionaMotor(100, 2, 100);
265                 BeepMedio();
266                 delay(250);
267             }
268
269             else if (distancia >= 200 && distancia < 400) {
270                 AcionaMotor(100, 1, 100);
271                 BeepLonge();
272                 delay(250);
273             }
274
275             // Fora de alcance
276             else {
277                 lcd.clear();
278                 lcd.setCursor(0, 0);
279                 lcd.print("Fora de alcance");
280             }
281         }
282     } // FIM
283
```