Universidade Paulista

Curso Bacharel de Ciência da Computação

André Bigatto de Menezes Bittencourt

Davi Vincenzo Merotto

Evelyn Mariani Dos Santos Silva

Nicolas Tadashi Suzuki

Sistema de monitoramento e auxílio para pessoas com mobilidade reduzida

São Paulo

2020

André Bigatto de Menezes Bittencourt

Davi Vincenzo Merotto

Evelyn Mariani Dos Santos Silva

Nicolas Tadashi Suzuki

Sistema de monitoramento e auxílio para pessoas com mobilidade reduzida

Trabalho de conclusão de curso para obtenção do título de graduação em Ciência da computação apresentado a Universidade Paulista - UNIP.

Orientador: Prof. Me. Gustavo Molina Figueiredo

São Paulo

2020

RESUMO

Um robô autônomo tem o intuito de realizar diversos objetivos e tarefas desejados em diversos ambientes sem necessitar da ajuda humana. Eles costumam ser utilizados em diversas áreas diferentes como a da logística, linhas de produção ou na área da saúde por exemplo.

Se tratando da área da saúde, muitas vezes o autônomo é utilizado com intuito de monitorar pacientes, principalmente se os mesmos possuem algum tipo de deficiência ou idade avançada. Com a intenção de garantir segurança dos pacientes, torna-se fundamental o desenvolvimento de um sistema de monitoramento para o grupo em questão.

Este projeto apresenta o desenvolvimento de um sistema autônomo, que faz uso de um controlador Arduino e um *smartphone* com o propósito de monitorar idosos e/ou pessoas com deficiência com o intuito de detectar possíveis quedas, além de agilizar o atendimento que as mesmas possam vir a necessitar através de notificações a contatos de emergência.

Os testes com o sistema foram realizados tanto em ambientes planos, tanto fechados quanto abertos, e obteve um bom retorno considerando tanto a locomoção do autônomo, quanto a detecção de quedas.

Palavras Chaves: Arduino, Robô**,** Monitoramento de idosos**,** Monitoramento de queda**,** Android.

ABSTRACT

An autonomous robot has the goal to perform many objectives and desired tasks in different environments without human intervention. They use to be used in many different areas such as logistics, production lines, or in the health field, for example.

When it comes to the health field, the autonomous robot can be used with the goal of monitoring patients, mainly if they are elderly or have any kind of disability. To ensure the patients' safety, it is essential for the development of a monitoring system for this group in particular.

This project presents the development of an autonomous system, making use of an Arduino controller and a smartphone to monitor the elderly and/or people with disabilities to detect probable falls streamlining the service that those people might need through emergency contact notifications.

Systems' tests were made on flat environments, both wide opened and closed, and got a good result considering both autonomous mobility, and fall detection.

Keywords: Arduino, Robot, Elderly monitoring, Fall monitoring, Android.

LISTAS DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CFG	*Configuration* (Configurações)
CPU	*Central Process Unit* (Unidade Central de Processamento)
DC	*Direct Current* (corrente contínua)
FPDS	*Fallen Person Dataset* (Conjunto de dados de pessoas caídas)
IBGE	Instituto Brasileiro de Geografia de Estatística
IBM	*International Business Machines Corporation* (Corporação Internacional De Máquinas De Negócios)
IDE	*Integrated Development Environment* (Ambiente De Desenvolvimento Integrado)
LCD	*Liquid Crystal Display* (Tela De Cristal Líquido)
LIDAR	*Light Detection and Ranging* (Detecção e Alcance Da Luz)
OMS	Organização Mundial de Saúde
OpenCV	*Open Source Computer Vision Library* (Biblioteca de visão computacional de código aberto)
PcD	Pessoa Com Deficiência
PNS	Pesquisa Nacional de Saúde
RADQ	Robô autônomo detector de quedas
RGB	*Red, Green and Blue* (Vermelho, verde e azul)
ROMs	*Read-Only Memory* (Memória exclusiva para leitura)
UFERSA	Universidade Federal Rural do Semi-Árido
FPS	*Frames Per Second* (Frames Por Segundo)
GPU	*Graphic Processing Unit* (Unidade de Processamento Gráfico)
YOLO	*You Only Look Once* (Você Apenas Olha uma Vez)
MB	*Megabyte*

SUMÁRIO

1. INTRODUÇÃO

Com o passar dos anos, a média da idade humana cresceu exponencialmente. Tal média era de 45,5 anos no ano de 1940, hoje em dia chega a ser de 76,2 anos de acordo com o IBGE. Apresentados estes dados, estima-se que cerca de 39% da população seja composta por pessoas acima dos 40 anos. Quando se trata de deficiência física, estima-se que existem 45 milhões de Brasileiros que fazem parte deste grupo.

Com isso, nota-se que no Brasil, boa parte da população é composta por pessoas que se encontram em uma situação onde sejam dependentes ou que necessitem de algum tipo de observação, seja pela idade avançada, doenças, ou por uma deficiência. Porém, muitas dessas pessoas vivem sozinhas, correndo o risco de sofrer algum acidente doméstico, que pode não ser descoberto a tempo.

O projeto tem como objetivo o desenvolvimento de um *software* e a construção de um robô autônomo, tal que, auxiliará no dia a dia de pessoas com mobilidade reduzida ou idade avançada. O robô realizará a tarefa de detecção de uma queda e o envio de alertas para números de emergência.

Para isso serão utilizadas informações obtidas através de pesquisas, estudos acadêmicos, artigos e matérias online, envolvendo robôs autônomos baseados ou não em Arduino, além de inteligências artificiais, que foram criados justamente com o intuito de ajudar e monitorar pessoas com necessidades especiais que sozinhas ou que precisam de um auxílio a mais.

Há uma grande variedade de projetos pelo mundo envolvidos neste assunto. Porém, visando que no Brasil há poucas opções no mercado para quem necessita, um aprofundamento nesse assunto e o desenvolvimento de um robô autônomo é de extrema importância para garantir uma boa qualidade de vida para pessoas com tais necessidades.

1.1. OBJETIVO

Desenvolver um *software* e um robô autônomo em Arduino, que auxiliará no monitoramento de pessoas de idade avançada e/ou mobilidade reduzida a fim de auxiliar na ocorrência de quedas, e evitar que as consequências da mesma se agravem.

1.2. JUSTIFICATIVA

Conforme o passar do tempo, a taxa de longevidade tende a crescer conforme a população vai envelhecendo e, como consequência, grande parte da população idosa acaba por ir morar em asilos, ou muitas vezes necessitam estar sob constante cuidado da família ou de cuidadores próprios. Muitas vezes o mesmo ocorre com pessoas que possuem alguma deficiência que tem como consequência a redução da mobilidade, tornando mais complicado para alguém com tais dificuldades em viver sozinho. Qual seria o meio mais adequado para oferecer ajuda a tais pessoas? Ter um “ajudante” não humano, seria uma boa ideia.

A criação de um robô autônomo seria de grande ajuda a esse público em questão pela sua principal função, um sensor de quedas que iria ajudá-los em emergências caso aconteça de o usuário cair e se machucar, por exemplo. Além de que somente o fato de o robô estar presente, serviria de grande auxílio para a saúde mental do usuário, que tendem por natureza se sentirem solitários ou abandonados, promovendo a sensação de conforto em saber que, em caso de necessidade, o robô prestará auxílio.

Uma pesquisa a cerca nesse tema fazendo o uso de robôs autônomos em Arduino auxiliará não somente pessoas idosas, mas também qualquer tipo de pessoa que necessitar de tal ajuda, acarretando que uma sensação de conforto seja gerada para famílias que queiram ter um cuidado especial para seus membros mais necessitados.

2. FUNDAMENTAÇÃO TEÓRICA

Em 2050, segundo a Organização Mundial de Saúde (OMS), a população mundial com pessoas acima de 60 anos passará de 541 milhões para 2 bilhões. No Brasil, segundo o Instituto Brasileiro de Geografia de Estatística (IBGE), a população de idosos em 2055 chegaria ao número de 70,3 milhões, enquanto a população jovem (de 0 a 14 anos) chegara a 34,8 milhões de pessoas.

Ainda segundo o IBGE, em 2013 na Pesquisa Nacional de Saúde (PNS), a porcentagem de brasileiros com pelo menos um tipo de deficiência era de 6,2% o equivalente a aproximadamente 13,8 milhões de pessoas. Sendo 58% deficientes

visuais, 21% deficientes físicos, 17,74% deficientes auditivos e 13% deficientes intelectuais.

Baseado nesses dados pode-se notar que ambas comunidades, tanto a de idosos quanto a de Pessoas com Deficiência (PcDs), são uma quantidade considerável de pessoas e necessitam de atenção especial.

Este capítulo aborda conceitos de tecnologias que serão utilizadas neste trabalho. Com o objetivo de auxiliar a população impactada, no seu monitoramento dos mesmos em suas residências. Dois destes conceitos são a base do trabalho: o Arduino e o Android.

Sua aplicação consta com o uso de sensor e uma câmera disponível em um celular Android, que são fundamentais para o correto monitoramento do indivíduo com o máximo de eficiência e baixos falso-positivos.

2.1. ARDUINO

Segundo Arduino Playground (2018), Arduino consiste em uma plataforma de prototipagem eletrônica baseada em hardware e software altamente flexíveis e simples de serem utilizados.

Foi inicialmente desenvolvido por cinco pesquisadores: David Cuartielles, David Mellis, Gianluca Martino, Massimo Banzi e Tom Igoe em 2005 com o intuito de ser um dispositivo de fácil programação e de baixo custo.

A placa permite que diversos circuitos elétricos sejam conectados em seus terminais de forma que possa se ter o controle de dispositivos externos, como motores e sensores de luminosidade.

Existem diversos tipos de placas com finalidades diferentes entre si, que fazem parte da família Arduino, sendo elas: Nano, Lilypad, Mega, Uno e Bluetooth (Arduino, 2014). As placas podem ter números diferentes de portas disponíveis para uso, por exemplo no Arduino Uno tem-se 14 portas digitais e 6 portas analógicas conforme a Figura 1.

Figura 1 - Arduino Uno.

Fonte: Site Embarcados, 2013.

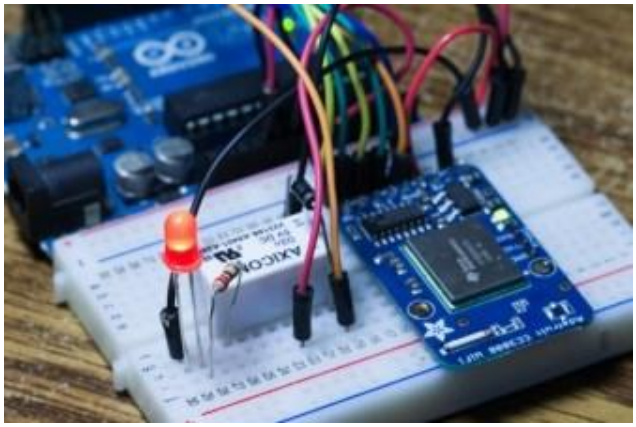
2.1.1. Sensores

De acordo com uma pesquisa encontrada no site da *Blucher Proceedings* (2018), sensores, também denominados como *Shields*, são dispositivos que fornecem uma saída adequada em resposta a uma medida específica, como por exemplo, utilizar um sensor de movimento para acender uma lâmpada no Arduino toda vez que detectasse um movimento.

Com isso em mente a função dos sensores seria a de estender a capacidade do Arduino, captando um parâmetro físico e o convertendo em um sinal adequado pronto para o processamento. Esses sensores são enquadrados em diversos tipos de atividades como, por exemplo, medidas de gases, luminosidade, infravermelho, ultrassom, tensão, temperatura, umidade, aceleração, corrente, movimento e posicionamento em um ambiente, como o giroscópio.

Vários projetos podem ser realizados utilizando a combinação de Arduino e sensores, como por exemplo: Um painel solar que se move de acordo com a incidência de radiação proveniente do sol; Uma planta que manda mensagens no Twitter quando precisa ser regada; Uma caixa de brinquedos aberta por leitura biométrica; Um robô espião sem fio; Lâmpadas com sensores de movimento, entre muitos outros. A figura 2 demonstra o processo do Arduino sendo estendido por um sensor.

Figura 2 - Sensor instalado no Arduino.



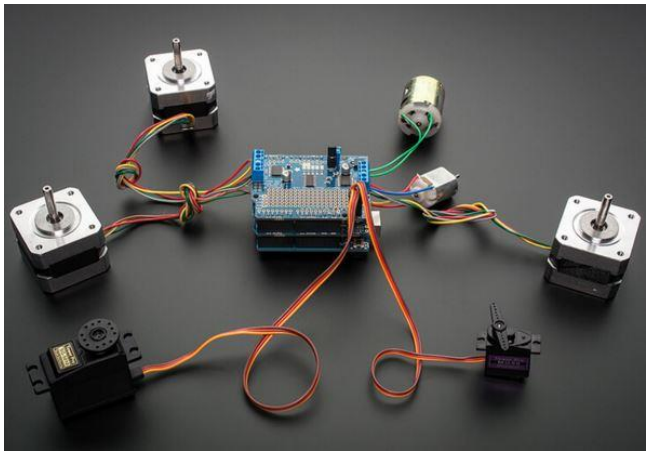
Fonte: Portal vida de silício, 2017.

2.1.2. Shields para Arduino

Se baseando no texto de Reis (2015), os *Shields* são extensões de funções do Arduino, lhe possibilitando por exemplo a reconhecer e se comunicar através de redes *WI-FI*, ou *Bluetooth*, mas também pode controlar motores, sensores de luminosidade, som, entre outras diversas funções.

Dependendo do Arduino utilizado é possível conectar diversos *Shields* diferentes, proporcionando a criação de projetos grandes e complexos, como a criação de um pequeno veículo autônomo, com diversos sensores e motores. Na Figura 3, mostrada abaixo, pode-se ver um *Shield* controlando diversos motores de tipos diferentes.

Figura 3 - *Shields* e motores



Fonte: DX.com, 2020.

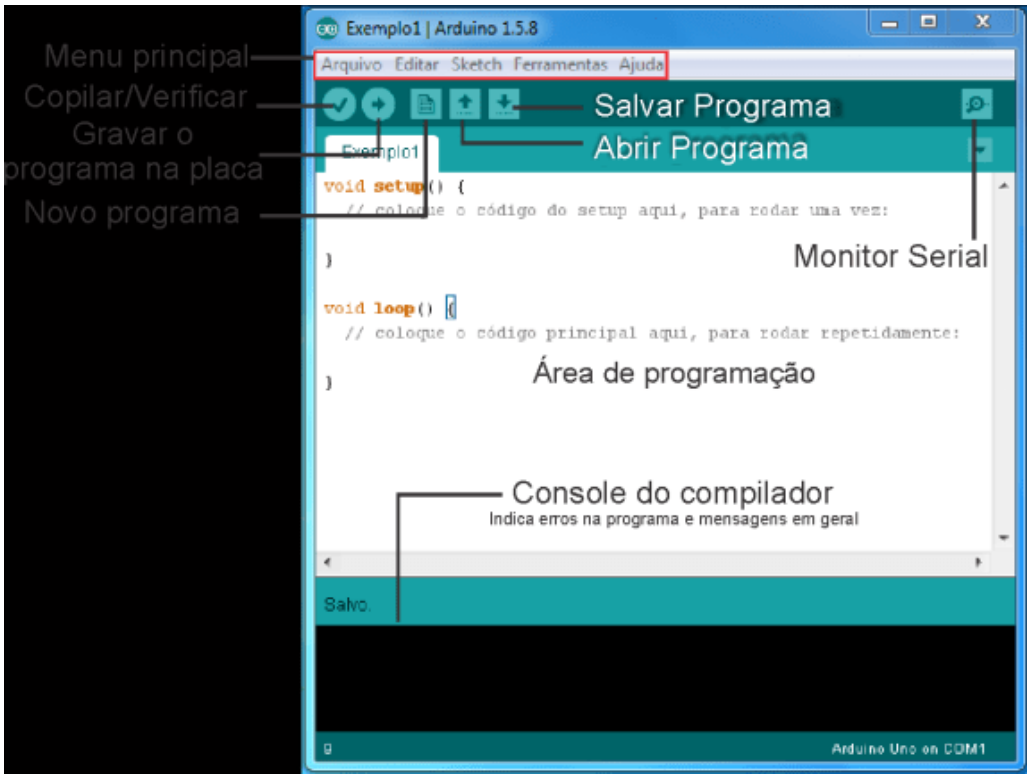
2.1.3. Arduino IDE

Um IDE (*Integrated Development Enviroment*) também conhecido como Ambiente de desenvolvimento integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. Essencialmente é um espaço criado para facilitar o trabalho do programador ao desenvolver seus programas.

IDE do Arduino (figura 4) é muito vantajoso por usar uma linguagem baseada no C/C++, que é linguagem bem difundida que faz uso de uma estrutura simples, o que ajuda muito pessoas com pouco ou sem conhecimento sobre a programar e elaborar programas rapidamente com apenas um pouco de estudo, por conta de ser um programar simples utilização, com uma grande gama de bibliotecas disponíveis na internet.

As funções da IDE do Arduino consistem em: Permitir o desenvolvimento do *Software*; Enviá-lo a placa-mãe para que possa ser executado; interagir com a placa Arduino.

Figura 4- Arduino IDE



Fonte: Vida de Silício, 2017.

2.2. ANDROID

O Android se trata do sistema operacional móvel da Google e teve sua origem em Palo Alto na Califórnia em 2003. O sistema foi desenvolvido pelos fundadores da Android Inc. Andy Rubin, Rich Miner, Nick Sears e Chirs White. Porém a aparição comercial do Android aconteceu somente em 2008.

O sistema tem sua presença em diversos aparelhos de vários fabricantes como por exemplo, Motorola, Samsung e Sony. Também é conhecido por ter seu núcleo baseado em Linux e pelo fato de possuir código aberto, além de uma série de possibilidades de personalização. Por conta disso o sistema tornou-se um dos mais populares entre as plataformas móveis do mundo.

O aspecto mais interessante do Android está presente em sua flexibilidade quando se trata de personalização. Diferente do iOS e do Windows *Phone*, o sistema Android permite que cada fabricante consiga adicionar suas próprias preferências, fazendo com que cada sistema de cada fabricante seja diferente. Isso é possível por meio das ROMs (*Read-Only Memory*, memória exclusiva para leitura) que muitas vezes são customizadas pelos desenvolvedores para a criação de aplicativos voltados para a personalização visual do sistema.

2.2.1. Java

Java é uma linguagem de programação multiplataforma desenvolvida por uma equipe de programadores da Sun Microsystems na década de 90 e é conhecida por ser uma das principais linguagens quando se trata de orientação a objeto.

O Java é baseado na linguagem C/C++, apesar disso, sua funcionalidade acaba sendo bem potente mesmo não tendo os aspectos menos utilizados e mais complexos presentes na C/C++. Por ser uma linguagem orientada a objetos, o Java é capaz de desenhar o *software* com o intuito de vincular suas operações com vários tipos de dados utilizados.

A linguagem oferece uma enorme biblioteca padrão e possui ferramentas que auxiliam para que os programas possam ser categorizados como distribuídos, tal característica está vinculada com a intenção da linguagem de fazer com que os desenvolvedores necessitem escrever o programa apenas uma vez e consigam

executá-lo em qualquer dispositivo. Outra característica trata-se do fato de ser uma linguagem *multi-thread*, ou seja, realiza múltiplas tarefas ao mesmo tempo dentro de um mesmo programa, melhorando assim o desempenho e a velocidade de execução.

A plataforma e confiante no quesito de segurança para desenvolver e executar aplicativos capazes de gerenciar a memória automaticamente, ou seja, fornece canais de comunicação seguros, a fim de proteger a privacidade dos dados e evitar a quebra de código.

2.2.2. Android Studio

O Android *Studio* trata-se de ambiente criado especificamente para desenvolver aplicativos para o sistema Android. Tem como base a IDE (*Integrated Development Environment*) IntelliJ *Community* e foi lançada oficialmente no evento Google I/O em 2013.

Por ser um ambiente relativamente mais recente, o Android Studio é comparado com outras IDEs, como por exemplo o Eclipse, porém demonstra algumas vantagens em relação a outras plataformas, sendo elas: Interface atraente; recurso de autocompletar palavras, métodos e funções; fácil integração e maior variedade de customização.

Porém existem algumas desvantagens, quando comparado ao Eclipse, o Android *Studio* tem um menor desempenho. No quesito administração de projetos, não é permitido que vários projetos estejam presentes em uma única tela, além disso, não é autorizada a reutilização de códigos. Quando comparado ao ambiente *PhoneGap*, o Android *Studio* não tem suporte para padrões *web*, como HTML e Javascript.

2.3. DETECÇÃO DE QUEDAS

Existem diversos tipos de detectores de quedas. Alguns utilizam dispositivos vestíveis como *smartwatches* ou se baseiam na tecnologia LIDAR e outros fazem uso de detecção de objetos através de câmeras e/ou sensores. Cada um destes métodos tem seus pontos positivos e negativos. Enquanto os *smartwatches* são intrusivos, que requerem uma pulseira e carregamento de bateria, os outros dois são o oposto e

podem trabalhar de forma autônoma. Porém, a tecnologia vestível é mais barata e requer uma menor configuração e manutenção. Para este projeto, foi escolhido o método por detecção de objetos.

Muitos modelos de detecção de objeto podem ser encontrados no meio científico, incluindo *RetinaNet*, YOLO e *TensorFlow*. Por conhecimentos prévios no assunto, foi escolhido o modelo YOLO, que será esclarecido no tópico a seguir em conjunto de ferramentas cruciais para o treinamento do modelo e seu uso posterior.

2.3.1. YOLO

O YOLO (*You Only Look Once* - em português, Você Olha Apenas Uma Vez) é um modelo detector de objetos que é conhecido como um dos mais rápidos, chegando a ter uma variação de FPS maior do que detectores como o *Faster R-CNN* e *RetinaNet*. No entanto, mesmo tendo uma velocidade alta, o *YOLO* não é rápido o suficiente para rodar em dispositivos embutidos como o *Raspberry PI.*

Com isso em mente os criadores do YOLO desenvolveram uma variação de sua estrutura conhecida como YOLO *Tiny*, que é aproximadamente 442% mais rápida e consegue a alcançar 244 *FPS* em uma única *GPU* (*Graphics Processing Unit* - em português, Unidade de Processamento Gráfico).

Sendo um modelo bem pequeno, menor que 50MB, e com velocidade de inferência rápida faz com que o YOLO *Tiny* seja um detector de objeto capaz de funcionar naturalmente em dispositivos embutidos com aprendizado profundo como o *Raspberry Pi*, Google *Coral* e *NVIDIA Jetson* Nano.

Apesar da sua capacidade de funcionar bem em dispositivos com baixa capacidade de processamento, o modelo pode não ser a melhor opção dependendo dos tipos de objetos que serão detectados e o nível de precisão necessário.

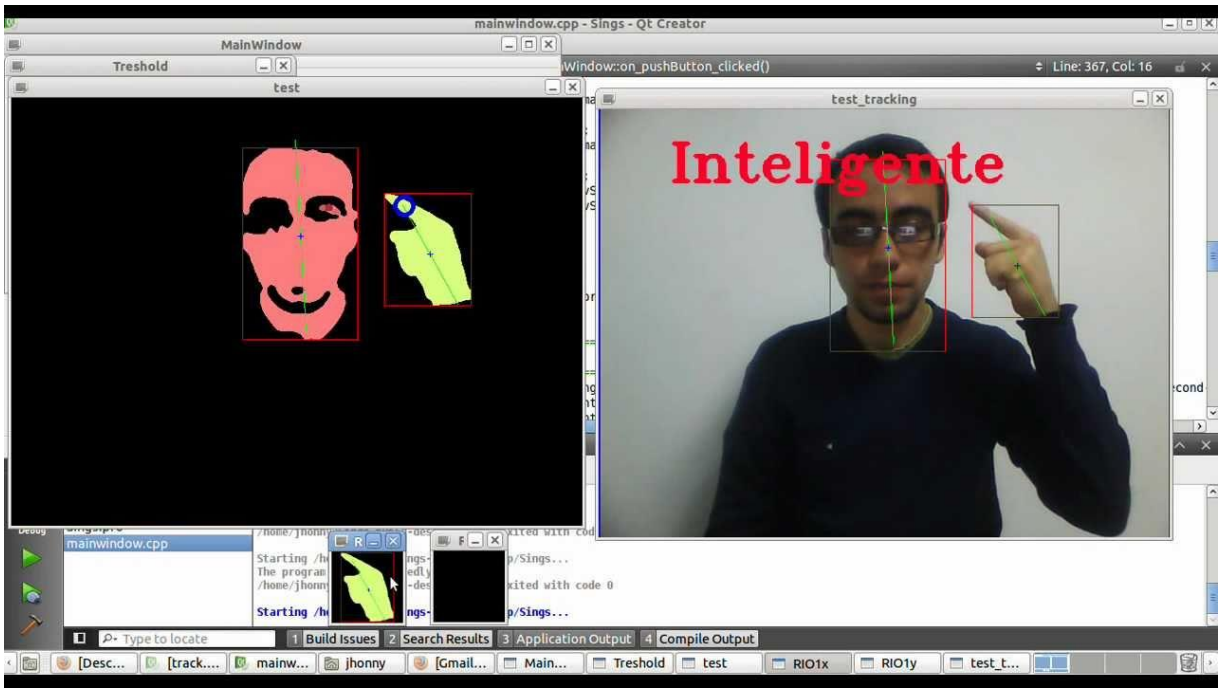
2.3.2. OpenCV

O *OpenCV* (*Open Source Computer Vision Library* - em português, Biblioteca de Visão Computacional de Código Aberto) é, como o próprio nome diz, uma biblioteca multiplataforma para o desenvolvimento de aplicações na área de visão computacional.

Essa biblioteca foi desenvolvida pela Intel em 2000, e possui módulos de processamento de imagens e mais de 350 algoritmos de visão computacional como filtros de imagens e reconhecimento de objetos. Ela foi construída utilizando-se a linguagem de programação C/C++, porém conta com adaptadores para compatibilidade com outras linguagens, como o Java.

Segundo a *Nvidia*, essa biblioteca foi adotada mundialmente por mais de 47 mil pessoas e seu uso varia entre artes interativas à inspeção de minas, além de monitoramento autônomo, navegação de carros, robôs autônomos e análise de imagens médicas. Na Figura 5 pode-se ver um exemplo do uso da biblioteca para detecção de sinais de uma linguagem de sinais.

Figura 5 - Reconhecimento de sinais utilizando *OpenCV*



Fonte: Youtube, 2011.

2.3.3. Google Colaboratory

O Google *Colaboratory*, ou apenas Google *Colab*, é um serviço de nuvem gratuito hospedado pela Google usado com o propósito de incentivar o desenvolvimento de pesquisas em Aprendizado de Máquina e Inteligência Artificial. O que facilita o desenvolvimento de projetos já que muitas vezes são necessários um

enorme poder computacional que não são de fácil acesso para alguns desenvolvedores.

Basicamente, o serviço é construído com base em *Jupyter Notebook*, que se trata de uma aplicação *web* de código aberto destinada ao desenvolvimento de arquivos que contém códigos e equações muito utilizada na área de aprendizado de máquina.

Conta com uma configuração flexível a fim de auxiliar na parte mais pesada de um projeto, além de contar com as seguintes características: suporte para Python 2.7 e Python 3.6; capacidade de suportar comandos *bash*; aceleração de *GPU* gratuita; bibliotecas pré-instaladas, incluindo o *TensorFlow*.

Outra característica importante trata-se do recurso de colaboração, a ferramenta permite que os desenvolvedores usem e compartilhem os projetos entre si, sem a necessidade de *download*, instalação ou execução de qualquer *software*, fazendo uso apenas do navegador. Todos os arquivos podem ser armazenados no Google *Drive*, o qual o *Colaboratory* tem total integração.

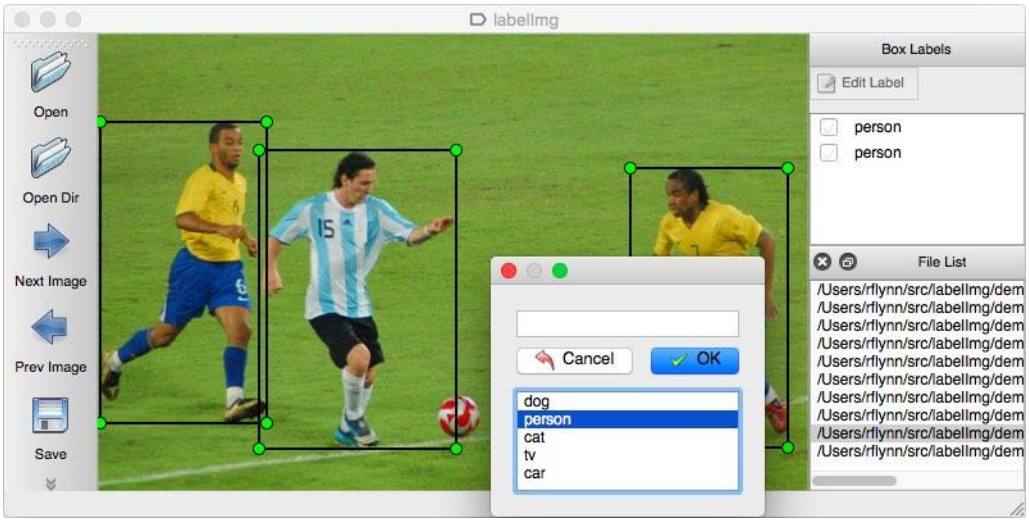
2.3.4. LabelImg

O *LabelImg* é uma ferramenta de código aberto e gratuita de anotação de imagem gráfica. Sua função consiste em auxiliar o desenvolvedor a colocar *tags*, ou etiquetas, nas imagens que serão utilizadas durante o treinamento de um modelo de detecção de objetos.

O desenvolvedor poderá então etiquetar objetos em cada imagem a partir de um conjunto, muitas vezes referido apenas a *dataset*. O processo consiste em criar um retângulo em volta de cada objeto em uma imagem usando a ferramenta, adicionando suas respectivas etiquetas, como pode observado nas Figuras 6 e 7. Esse processo gera um arquivo de texto, composto pelas coordenadas dos quatro pontos do retângulo e um identificador da etiqueta.

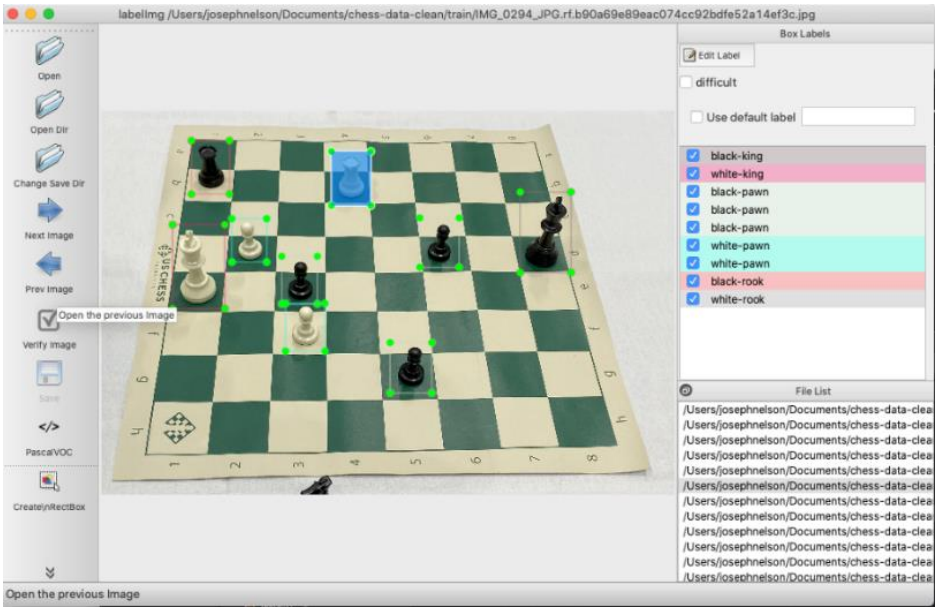
Esse processo é fundamental para o treinamento do modelo, já que o algoritmo se baseará e focar seu estudo no conteúdo dos retângulos criados.

Figura 6- *Labellmg* em jogadores de futebol



Fonte: GitHub, 2015.

Figura 7 - *Labellmg* em tabuleiro de xadrez



Fonte: *Roboflow*, 2020.

2.3.5. RoboFlow

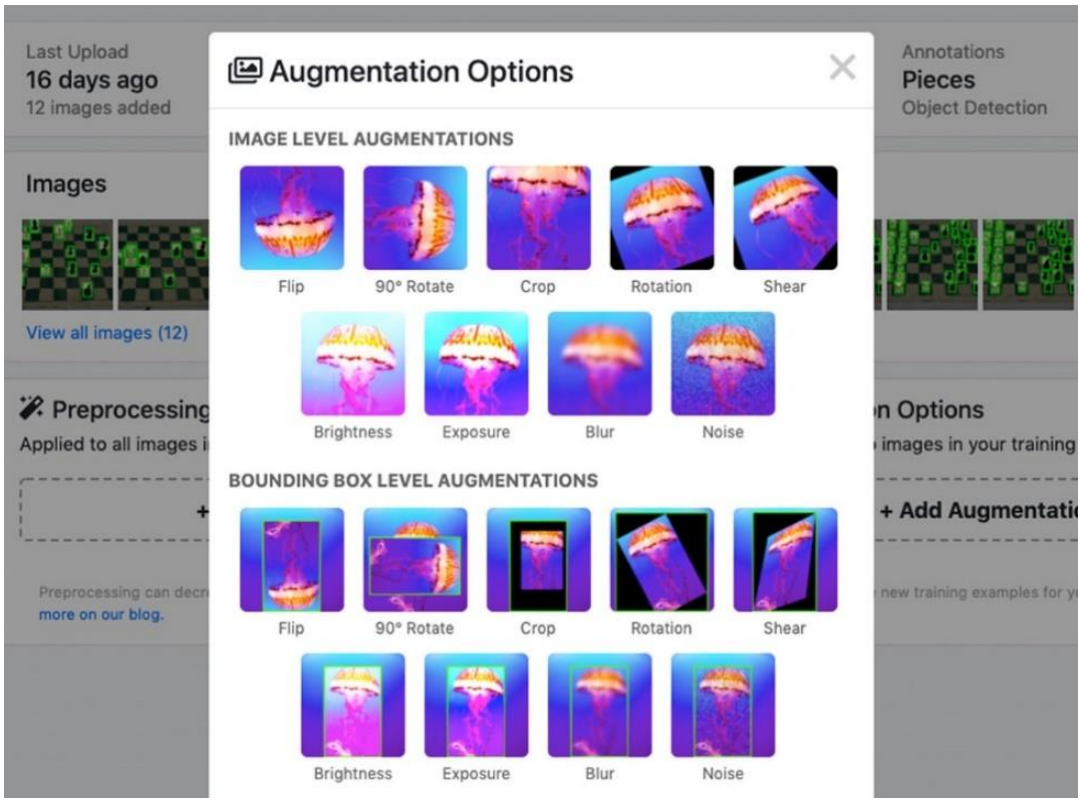
RoboFlow é uma empresa digital que tem como objetivo ajudar desenvolvedores a melhorar suas aplicações de computação visual. Fundada em 2020, ela preencheu uma lacuna na área de pré-processamento de imagens para treinamento de modelos de identificação visual.

Sua função é ser responsável por normalizar imagens de um *dataset*, mantendo as imagens em um padrão de resolução, o que é muito importante ao treinar um modelo. Resoluções de imagens diferentes, podem confundir e reduzir a acurácia do mesmo.

Além da resolução, a *Roboflow* também oferece outros tipos de normalização, como transformar todas as imagens do *dataset* em escala de cinza, para treinar modelos que não necessitam de detalhes nas cores. O ponto positivo em treinar um modelo apenas com escalas de cinza é a rapidez, uma vez que não será treinado utilizando o RGB.

Mas talvez a ferramenta mais importante seja a de criação de *augmented images*, ou imagens aumentadas. Baseando-se em um *dataset* e de acordo com as preferências do desenvolvedor (Figura 8), são gerados arquivos de imagens extras. Esses arquivos têm a função de expandir, melhorar e aumentar a precisão do modelo, lhe ensinando que pode haver diferenças de brilho, posição ou torção nas imagens captadas.

Figura 8 - Opções para “aumentar” imagens



Fonte: *Roboflow*, 2020.

2.4. FIREBASE

Trata-se de uma plataforma digital desenvolvida pela Google destinada a criação do *back-end* de aplicativos de forma simples. Atualmente muitas plataformas suportam a ferramenta, seja Android, iOS ou *Web*. O uso da mesma é feito através de um console *web*. O desenvolvedor pode adicionar um projeto e incluir os serviços necessários, infelizmente, não é totalmente gratuito.

Um dos seus principais serviços é o de *Authentication*, nele é possível adicionar a função destinada a realizar o login no aplicativo através de um sistema de contas próprio, ou de contas já existentes, como por exemplo, Google, Facebook, Twitter e GitHub. O serviço de *Realtime Database* trata-se de um banco de dados que sincroniza os dados em tempo real, sendo possível também definir quem tem acesso a quais dados.

Há também o serviço de *Cloud Messaging*, o qual permite o envio de mensagens entre usuários através do aplicativo. Já o serviço de *Storage* armazena dados por uma conexão segura e permite o compartilhamento deles. É possível testar o aplicativo em diversas plataformas através do serviço de *Test Lab*, tanto de forma manual, quanto de forma automática com o *Robo Test*, mas também é possível que o desenvolvedor crie seus próprios scripts de teste.

No caso de ocorrência de erros, eles podem ser coletados pelo serviço de *Crash Reporting*. O desenvolvedor pode optar pelo envio de Notificações personalizadas aos usuários através do serviço de *Notifications*.

Com as grandes quantidades de diferentes dispositivos disponíveis no mercado, é possível que os desenvolvedores publiquem diferentes versões do aplicativo através do serviço de *Remote Config*, normalmente, tal serviço é usado para testar mudanças com um pequeno grupo de usuários antes de aplicá-las aos demais.

É possível monetizar o aplicativo de forma fácil com as ferramentas *AdWords* e *AdMob*. O serviço de *AdWords* publicará anúncios referentes ao aplicativo no Google, Youtube e *Play Store*, já o *AdMob* inclui anúncios dentro do aplicativo, priorizando fontes que retornam um maior lucro.

Em resumo, o *Firebase* é uma ferramenta completa e essencial no desenvolvimento de aplicativos, sendo uma das melhores opções atualmente. Na

programação do aplicativo foram utilizados os serviços *Authentication*, *Realtime Database* e *Crash Reporting*.

2.5. GITHUB

O GitHub é uma plataforma de hospedagem de código-fonte e de versionamento de arquivos, baseada em *Git*. É muito utilizado por desenvolvedores a fim de divulgação de seus projetos e compartilhamento com o público de seus respectivos códigos-fonte. Esses projetos são comumente chamados de “repositórios”.

O versionamento de arquivos nada mais é que uma lista de versões que um determinado arquivo tem. É comum que um arquivo Java, por exemplo, tenha mais de uma versão durante todo o desenvolvimento de um projeto. Isso pode ser motivado por adição de funcionalidades, remoção de erros ou remoção do arquivo inteiro por desuso. O versionamento é importante já que a qualquer momento o desenvolvedor pode retornar a uma versão anterior, caso alguma modificação tenha tido um efeito indesejado.

O GitHub também pode manter uma base de defeitos abertos por usuários, ou mesmo os desenvolvedores, daquele projeto específico. Essa funcionalidade é muito utilizada para manter registrado as melhorias requisitadas e consertos de funcionalidades.

2.6. USBSERIAL

A *UsbSerial* é uma biblioteca de código aberto e gratuita, desenvolvida em Java com a finalidade de conectar um aparelho Android com uma placa Arduino através de um cabo serial ligando os dois dispositivos.

Cerca de 36 projetos utilizam essa biblioteca para as mais diversas aplicações, como por exemplo a *Mouse4all* que é um aplicativo que conecta o dispositivo Android a um aparelho, semelhante a um mouse, especialmente desenvolvido para um público com dificuldades de se utilizar uma tela *touchscreen*, como pessoas com *Parkinson*, paralisia cerebral e esclerose múltipla.

3. TRABALHOS RELACIONADOS

Na Universidade de Missouri (2017), nos Estados Unidos da América, foram desenvolvidos diversos sistemas para monitoramento de idosos. Um deles (*Intelligent Sensor System for Early Illness Alerts in Senior Housing*) utiliza sensores não-vestíveis como, por exemplo, sensores de movimento, sensores instalados em uma cama e sensores que analisam padrões ao andar. Com todos esses sensores o sistema é capaz de detectar sinais de mudança de comportamento de uma pessoa que poderia indicar problemas de saúde, e de forma automática, alertar funcionários para que eles possam prover cuidados de forma proativa.

A empresa belga *Zora Bots*, desenvolveu um robô para auxiliar os idosos que se encontram em situações de solidão, o robô(Figura 9) trata-se de uma combinação entre um brinquedo e tecnologia, controlado por um ser humano para interagir com o paciente, podendo se mover e se comunicar através de comandos. O robô não tem o intuito de cuidar do paciente, mas seu foco é ajudar com a necessidade de interação física do paciente. Diversos testes foram realizados em hospitais e lares para idosos, e na maioria dos casos o robô traz melhoras significativas no estado psicológico dos pacientes. O robô também pode ser utilizado em outras situações, por exemplo, interagir com crianças deficientes.

Figura 9 - O companheiro

Fonte: *NBC News*, 2017.

A empresa *Apple* desenvolvedora do *Apple Watch*, inseriu em suas versões mais recentes a funcionalidade de detecção de quedas graves. Enquanto o usuário está com seu *Apple Watch* em seu braço, seus movimentos estão sendo monitorados caso ocorra uma queda o aparelho emitirá um alarme e em sua tela será exibida uma mensagem perguntando se o usuário está bem (Figura 10). Caso este não interaja com o aparelho ele envia uma mensagem de emergência com sua localização e liga para os serviços de emergência, reproduzindo uma mensagem automática informando os dados do usuário e sua localização.

Figura 10 - *Apple Watch* pergunta ao usuário se ele está bem

Fonte: Apple, 2019.

A tecnologia LIDAR (*Light Detection And Ranging*) está cada vez mais presente no mundo automobilístico. Trata-se tecnologia óptica que normalmente utiliza o emprego de lasers pulsados, tais esse que possuem o objetivo medir a distância entre eles e um objeto. Para que isso, um pulso é emitido, logo após é calculado o tempo até que o mesmo pulso que foi refletido, seja detectado. Essa tecnologia tem sido usada frequentemente em carros autônomos para gerar modelos digitais das ruas de tal forma que as Inteligências Artificiais sejam capazes de se guiarem e evitarem possíveis colisões. Esta tecnologia agora está sendo estudada pela IBM *Research* e a startup *Cera Care* (2019), no Reino Unido, para ser utilizada em ambientes fechados. Utilizando a Inteligência Aumentada da IBM é possível detectar anomalias no ambiente que possa significar uma possível queda. Pode-se observar na Figura 11 que ao detectar uma anomalia, que no caso, trata-se de uma pessoa caída, um sinal

é emitido pelo sistema e a partir daí pode ser tomada uma série de medidas para que essa pessoa seja socorrida.

Figura 11- Anomalia detectada pelo sistema LIDAR

Fonte: YouTube, 2019.

De acordo com o site Inclusão360 (2019) um robô foi desenvolvido por uma equipe em Pernambuco, logo após a mesma ser escolhida como a representante do Brasil para uma competição chamada *Robocup* 2015, que se trata de uma competição mundial de robótica, que no ano em questão, ocorreu na China. O robô desenvolvido, chamado de *i-Zaq*, foi criado justamente para acompanhar os idosos que moram sozinhos e oferece vários auxílios, como por exemplo, lembrar o horário dos medicamentos, informar o valor nutritivo dos alimentos além de ser útil em situações emergenciais ao acionar o socorro e telefonar para os familiares do paciente.

Diogo Lacerda, o técnico responsável pela mecatrônica, mencionou que o robô *i-Zaq* é um robô doméstico com a função de interagir com pessoas dentro de seus lares, incluindo pessoas doentes e/ou idosas que permanecem sozinhas em casa no decorrer do dia. A interação mencionada é complementada pela presença de um

coração, que dependendo da quantidade de energia presente na bateria, pisca de diferentes maneiras. Caso a bateria estiver cheia o coração pisca com muito mais intensidade. O mesmo também possui um rosto que demonstra expressões virtualmente. A figura 12 mostra uma paciente que faz uso do *i-Zaq* no seu dia a dia.

Figura 12 - Interação com humanos.

Fonte: TV Globo, 2015.

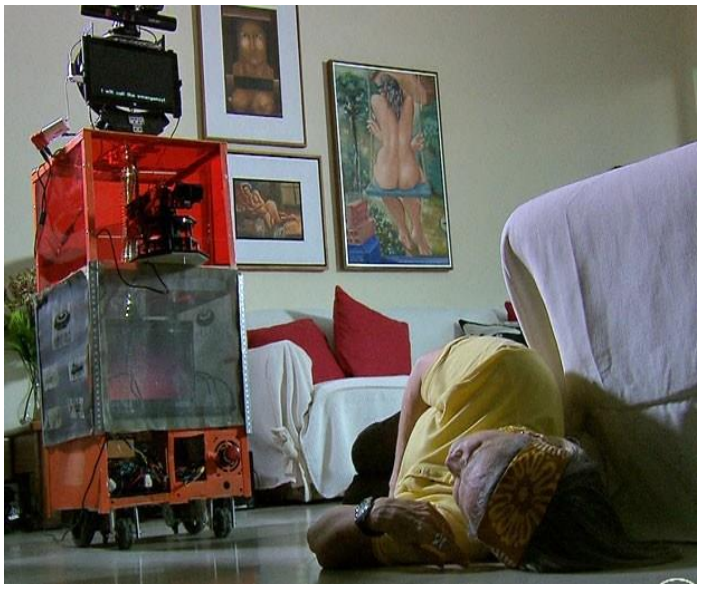
A paciente em questão, que se chama Maria Auxiliadora, realizou vários testes para averiguar as reações do robô. Ao mostrar alguns alimentos industrializados ao *i-Zaq*, o mesmo mostrou a elas valores nutritivos do alimento em questão. Também foi apresentado ao *i-Zaq* suas caixas de remédio, ele retornou a ela os dias em que tomou seus medicamentos e o horário em que deve tomá-los novamente.

Outra função do mesmo é reconhecer os movimentos das pessoas, e as seguir, evitando colisão com paredes ou moveis, além de reconhecer gestos.

O *i-Zaq* possuía função de medição de batimentos cardíacos que é ativada quando um idoso mantém seus olhos fixos no ‘rosto” do robô. A visão computacional o *i-Zaq* consegue deduzir o batimento cardíaco e a irrigação dos vasos sanguíneos ao analisar a variação da tonalidade da pele de uma pessoa.

Quando ocorre uma queda (Figura 13), o *I-Zaq* pergunta para o usuário se está tudo bem, dependendo da resposta, ele realiza um telefonema para o pronto socorro e para os familiares de seu dono.

Figura 13 - Detecção de quedas.



Fonte: TV Globo, 2015.

De acordo com um artigo no site *CanalTech* (2019) a primeira versão de robôs pets (animais) foi criada em 2015 pela empresa *Hasbro*, da série *Joy for All Companion Pets*. Tais robôs foram co-desenvolvidos devido a demandas e solicitações de familiares e cuidadores que buscavam oferecer conforto e relações mais afetuosas com parentes que estivessem envelhecendo.

Esses robôs (Figura 14) animalescos funcionam a partir de vários sensores espalhados pelo corpo, dependendo da raça ou do animal que o robô está baseado seus sensores serão diferentes, e assim trabalham suas interações via comandos pré-estabelecidos em seus *softwares*. Por exemplo, quando é feito um carinho no peito de um robô gato, o robô lambe própria “pata”, e dependendo de outros estímulos feitos neles, os robôs podem mexer as orelhas ou mover a cabeça por exemplo. Já um cachorro robô realiza outros movimentos como, ao receber um estímulo, ele abana o rabo para demonstrar felicidade, e chacoalhar a cabeça para os dois lados. Quando há uma ausência de estímulos, os robôs chamam os pacientes por meio sonoro, imitando um animal real.

Tais robôs costumam ser utilizados no tratamento de pessoas com algum tipo de deficiência mental, como demência ou autismo. A Dra. Marcella Bianca Neves explica que tal interação entre os pacientes e robôs, auxilia na diminuição da agitação e ansiedade nos pacientes, além de também trabalhar questões de interação social e emocionais.

Figura 14 - Pets interativos.

Fonte: *CanalTech*, 2019.

Um artigo encontrado no site da Petrobrás (2019) explica os detalhes de um projeto chamado "*Robot* em ação" possui como objetivo de a melhora do aprendizado entre jovens, através da inclusão digital, mais especificamente jovens de escolas públicas na cidade de Mossoró no Rio Grande do Norte. A motivação de tal projeto foi realizar os sonhos de muitas pessoas de trabalhar na criação robôs que sejam mais próximos o possível dos robôs que foram demonstrados em vários tipos de mídia, como por exemplo, a robô Rosie do desenho "Os Jetsons" ou o robô WALL-E criado pela Disney, entre muitos outros.

O projeto foi patrocinado pela Petrobras e executado pela Universidade Federal Rural do Semiárido (UFERSA) e tem como iniciativa oferecer aulas de robótica educacional fazendo uso de peças de encaixe Lego, a fim mais atraente para os jovens o mundo da computação e robótica. Juntamente as peças de encaixe é implementada uma CPU, que possui entradas para realizar a conexão de motores e sensores juntamente a uma pequena tela de LCD e botões para realizar as configurações nos programas armazenados no controlador. A construção de robôs auxilia os jovens envolvidos nesse projeto a trabalhar suas habilidades de observação e prática na experimentação. Na figura 15 é possível ver o projeto em questão.

Figura 15 - Robô modular.

Fonte:Petrobrás,2019.

Por fim, outro projeto desenvolvido na Espanha pela Universidade de Calcutá, chamado *Fallen People Detection Capabilities Using Assistive Robot* (Capacidades de Detecção de Pessoas Caídas Utilizando um Robô Assistente) tem como objetivo a criação de um robô para monitoramento e auxílio de pessoas idosas ou com mobilidade reduzida e que vivem sozinhas. Além disso, segundo o artigo, o robô poderia trabalhar como um andador, devido ao seu tamanho, como se pode ver na Figura 16.

Figura 16 - Lola, Robô Assistente

Fonte: MDPI, 2019.

O robô é composto por um Arduino Mega, vários sensores, uma câmera e um *Raspberry* e todo o processamento de imagem é feito a partir de um servidor, responsável pela detecção de quedas. O treinamento deste robô foi feito com uso do modelo YOLOv3 utilizando um *dataset* de imagens própria, devido à falta de um banco de imagens de pessoas caídas. O robô é capaz ainda de diferenciar pessoas deitadas em camas ou sofás e pessoas caídas no chão, como mostrado na Figura 17.

Figura 17 - Diferença entre usuário caído e deitado em uma cama.

Fonte: MDPI, 2019.

4. DESENVOLVIMENTO

O sistema se dá por um robô, o qual apelidou-se de RADQ (Robô autônomo detector de quedas) autônomo desenvolvido em Arduino conectado a um *smartphone* e faz uso da câmera do mesmo, com o propósito de identificar possíveis quedas.

O autônomo é montado em cima de um chassi 2wd, que possui dois motores DC, que realizam a ação de acelerar e fazem com que o autônomo se locomova. No chassi foi implantado um Arduino Uno, onde fica localizado o programa feito em linguagem C que comanda todo o restante do robô, realizando comandos, como por exemplo, fazer com que os motores rodarem para que o autônomo ande. O principal comando realizado pelo Arduino está diretamente ligado a um micro servo e um sensor de distância, que tem como objetivo evitar que o robô se choque contra algo ou alguém.

O Arduino tem acesso à câmera do *smartphone* conectado a ele através de um aplicativo desenvolvido com essas e outras finalidades. Inicialmente o protótipo do

mesmo foi criado no programa Adobe XD, logo após foi desenvolvida a parte visual em HTML e CSS. A programação é feita com uma junção de Java e C++.

4.1. FUNCIONAMENTO DO SISTEMA

O sistema é pensado para ser intuitivo e de fácil uso, para que o usuário final não venha a ter dificuldades. Após configurado, não há necessidade de nenhuma outra ação a ser realizada pelo usuário, a não ser a troca de bateria quando necessário. As configurações necessárias tratam-se de abrir o aplicativo no celular, definir um contato de emergência, e conectá-lo ao autônomo.

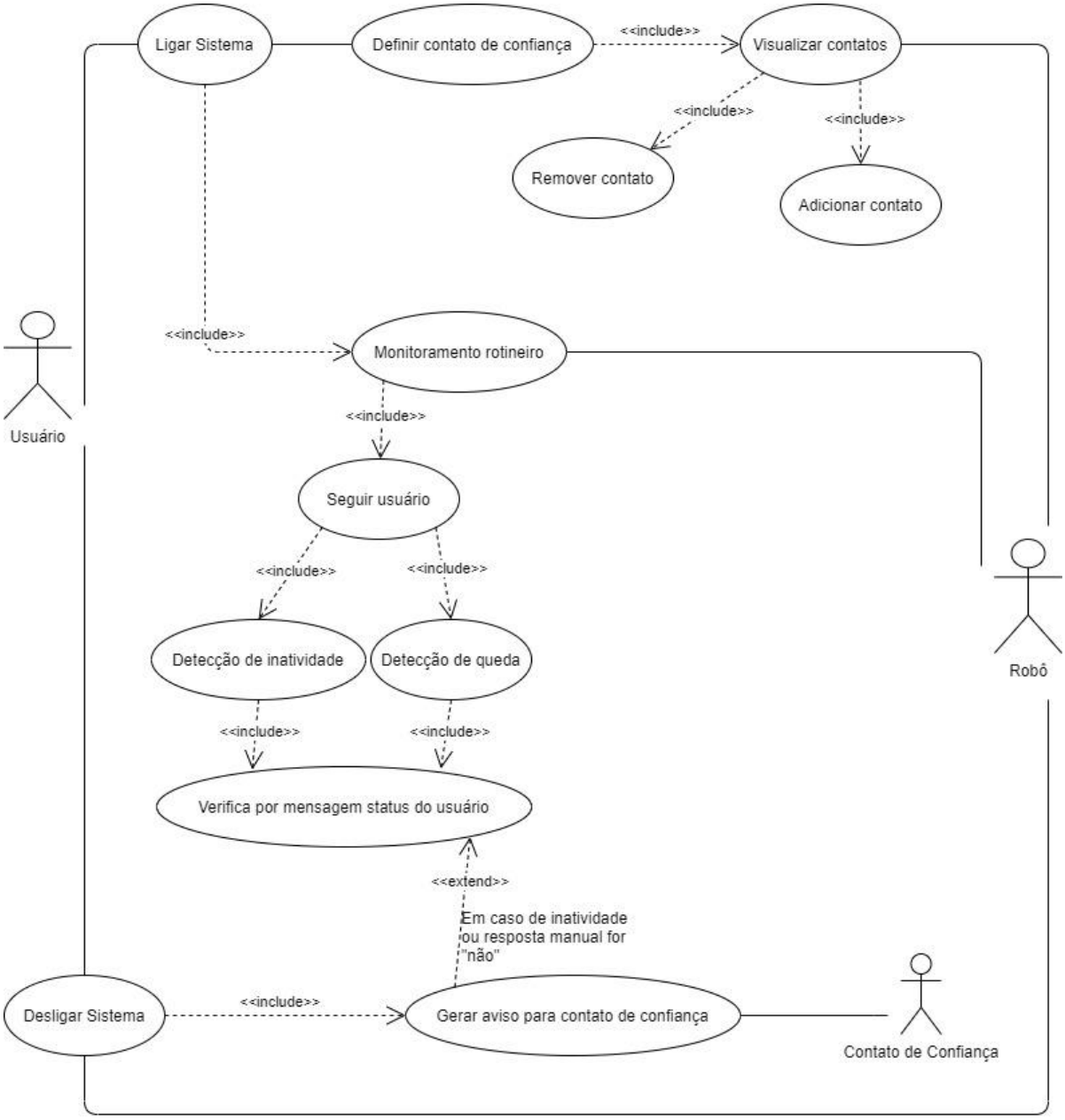
Ao estabelecer a conexão do autônomo com o celular através de um cabo USB e necessário que o usuário abra a opção câmera contida no aplicativo e o posicione acima do robô para que a câmera tenha um bom ângulo de visão, e então o robô realizará a função de seguir o usuário de forma autônoma enquanto faz um reconhecimento de imagem.

Em caso de queda o aplicativo pergunta para o usuário se está tudo bem, caso não haja uma resposta em 10 segundos, é enviado uma mensagem de alerta para o contato de emergência pré-definido. Caso o usuário confirme que está tudo bem, o aplicativo não envia a mensagem, apenas uma notificação informando a queda.

Ambos os alertas de emergência são gerados pelo aplicativo e enviados pela rede *Wi-Fi*. Todo o processamento do sistema é feito pelo Aplicativo, enquanto as funções de movimentação e detecção de colisão é processado pelo Arduino.

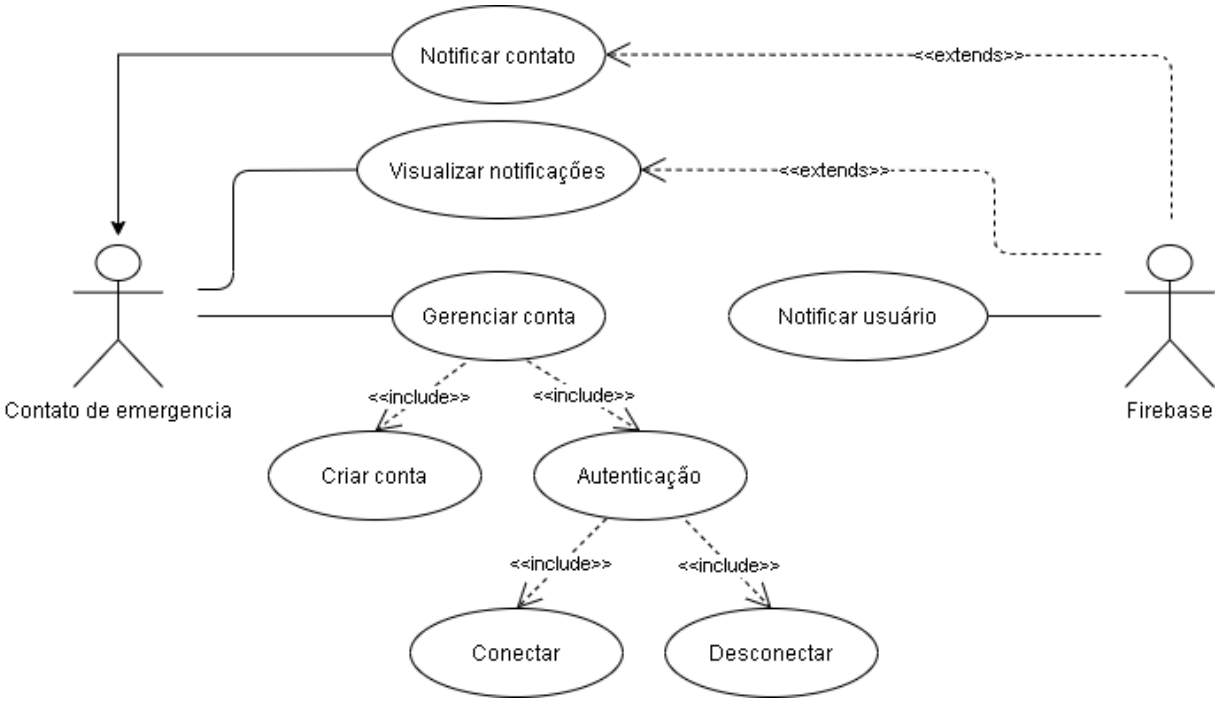
4.2. DIAGRAMAS DE CASOS DE USO

Figura 18 - Diagramas de casos de uso do aplicativo principal



Fonte: Autores, 2020.

Figura 19 - Diagramas de casos de uso do aplicativo secundário.



Fonte: Autores, 2020

4.3. DOCUMENTAÇÃO DOS CASOS DE USO

Tabela 1 - Documentação caso de uso

Nome do caso de uso	Geral
Ator principal	Sistema
Ator secundário.	Proprietário.
Resumo.	Este caso de uso descreve o funcionamento geral do sistema.
pré-condições.	
pós-condições.	
Fluxo principal	
Ações do Ator	Ações do Sistema
	1- Monitoramento rotineiro.
	2- Seguir usuário.
Fluxo Alternativo I	
Ações do Ator	Ações do Sistema

1- Caso usuário deseje informar contato de emergência.	
	2- Registrar contato.
Fluxo Alternativo II	
Ações do Ator	Ações do Sistema
	1- Caso seja detectado uma queda do usuário. Perguntá-lo se está bem.
	2- Caso a resposta seja sim, voltar ao monitoramento e notificar contato de emergência.
	3- Caso a resposta seja não, gerar notificação e enviar para o contato de emergência.
	4- Caso não responda dentro de 30 segundos, gerar alerta notificação e enviar para o contato de emergência.

Tabela 2 - Requisitos Funcionais

Requisito Nº	Nome	Descrição
#RF1	Gerenciamento de contatos	O sistema deve gerenciar os contatos adicionados pelo usuário.
#RF2	Acompanhar usuário	O sistema físico (robô) deve acompanhar o usuário continuamente seguindo-o automaticamente através das imagens capturadas.
#RF3	Detector de queda	O sistema deve, a partir das imagens capturadas pelo aplicativo, detectar uma possível queda do usuário. Caso haja uma queda, o sistema deve emitir um aviso.
#RF4	Verificação por mensagem	O sistema deve solicitar ao usuário que ele responda, através da tela do dispositivo móvel,

		se ele está consciente e não necessitando de ajuda. A verificação só é ativada caso seja detectada uma queda (RF3).
#RF5	Avisos para contato de emergência	O sistema deve, através da internet, enviar um aviso ao contato de emergência caso a verificação por mensagem (RF4) indique que o usuário não respondeu à verificação ou que respondeu necessitar de ajuda manualmente.

Tabela 3 - Requisitos Não Funcionais

Requisito Nº	Nome	Descrição
#RNF1	Configuração	O sistema deve ser feito de forma que as configurações sejam simples e que não haja configurações que impactem a funcionalidade principal de monitoramento do usuário.
#RNF2	Bateria	A bateria do sistema deve ser o suficiente para pelo menos um dia de autonomia.
#RNF3	Funcionamento autônomo	O sistema após a configuração inicial deve ser autônomo. Nenhuma outra configuração será necessária para que ele funcione. O robô deve solucionar problemas relacionados a locomoção de forma autônoma.
#RNF4	Conectividade	O sistema necessita de conexão à internet, para o envio de notificações.
#RNF5	Armazenamento de dados	O sistema deve garantir o armazenamento dos contatos de emergência.
#RNF6	Performance	O sistema necessita que todas as suas decisões sejam rápidas. Um usuário não pode esperar por mais de 1 minuto que uma mensagem de alerta seja enviada ao contato de emergência.
#RNF7	Confiabilidade	O sistema deve garantir que as mensagens sejam enviadas ao contato de emergência.

		O sistema deve reduzir o número de falsos positivos e falsos negativos. O sistema deve garantir que a conexão à internet esteja sempre funcionando enquanto este estiver ligado. Em caso de falha em qualquer parte do sistema, ele deve garantir que a falha seja corrigida.

Tabela 4 - Documentação caso de uso aplicativo secundário

Nome do caso de uso	Geral
Ator principal	Sistema
Ator secundário.	Proprietário.
Resumo.	Este caso de uso descreve o funcionamento geral do sistema.
pré-condições.	
pós-condições.	
Fluxo principal	
Ações do Ator	Ações do Sistema
	1- Recebimento de notificações.
Fluxo Alternativo I	
Ações do Ator	Ações do Sistema
1- Caso usuário deseje criar uma conta.	
	2- Registrar conta.
Fluxo Alternativo II	
Ações do Ator	Ações do Sistema
1- Caso usuário deseje entrar com uma conta existente.	
	2- Solicitar dados de login.
Fluxo Alternativo III	

Ações do Ator	Ações do Sistema
	1- Caso aplicativo principal envie uma notificação.
	2- Apresentar a notificação ao usuário.

Tabela 5 - Requisitos Funcionais

Requisito Nº	Nome	Descrição
#RF1	Gerenciamento de conta	O sistema deve gerenciar a conta dos usuários.
#RF2	Receber avisos do aplicativo principal	O sistema deve, através da internet, receber um aviso do aplicativo principal em todo tipo de ocorrência registrada pelo aplicativo principal.

Tabela 6 - Requisitos Não Funcionais

Requisito Nº	Nome	Descrição
#RNF1	Configuração	O sistema deve ser feito de forma que as configurações sejam simples e que não haja configurações que impactem a funcionalidade principal de recebimento de notificações.
#RNF2	Conectividade	O sistema necessita de conexão à internet, para o recebimento de notificações.

4.4. PROTÓTIPO

4.4.1. Hardware utilizado

O Arduino Uno (Figura 20) é responsável pelo processamento de informações enviadas pelo aplicativo e pelo movimento do robô no quesito andar, virar ou parar quando necessário.

Figura 20 - Arduino Uno

Fonte: FilipeFlop, 2020.

Os motores DC - *Direct Current* (corrente contínua) (Figura 21) são muito importantes para a movimentação do carrinho, o projeto inicial prevê o uso de dois destes motores funcionando de forma independente para possibilitar a troca de direção.

Figura 21 - Motor DC

Fonte: FilipeFlop, 2020.

O sensor ultrassônico HC-SR04 Sensor (Figura 22) será utilizado para evitar colisões com obstáculos, ele mede a distância através de pulsos sonoros e calcula a distância medindo o tempo de retorno do som.

Figura 22 - Sensor ultrassônico

Fonte: FilipeFlop, 2020.

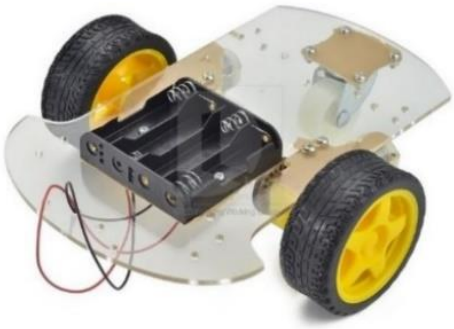
Um Micro servo motor 9g SG90 *TowerPro* (Figura 23) será utilizado para rotacionar o sensor ultrassônico, este motor tem uma rotação de apenas 180 graus. Isso possibilita a precisão da direção à qual o sensor observará.

Figura 23 - Micro servo motor

Fonte: ElectronicsComp, 2020.

Foi utilizado também um chassi 2WD (Figura 24) que será o corpo do nosso protótipo. O mesmo já possui dois motores DC, duas rodas e um conector de pilhas que será substituído por uma bateria de 9 V.

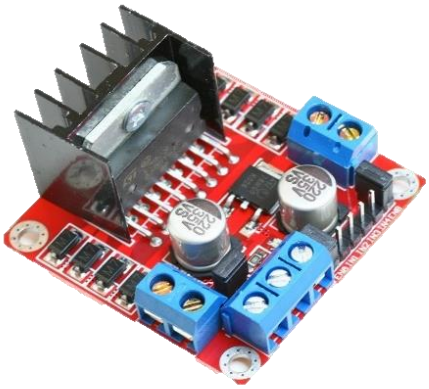
Figura 24 - Chassi 2WD



Fonte: FilipeFlop, 2020.

Para controlar os dois motores DC que farão toda a movimentação do robô, será necessário uma ponte H, um *Shield* para motores com o chip de controle L298N, essa placa possibilita o fornecimento de energia suficiente para os dois motores além de facilitar o controle dos mesmos, a placa específica que será utilizada está representada na Figura 25.

Figura 25 - Ponte H L298N



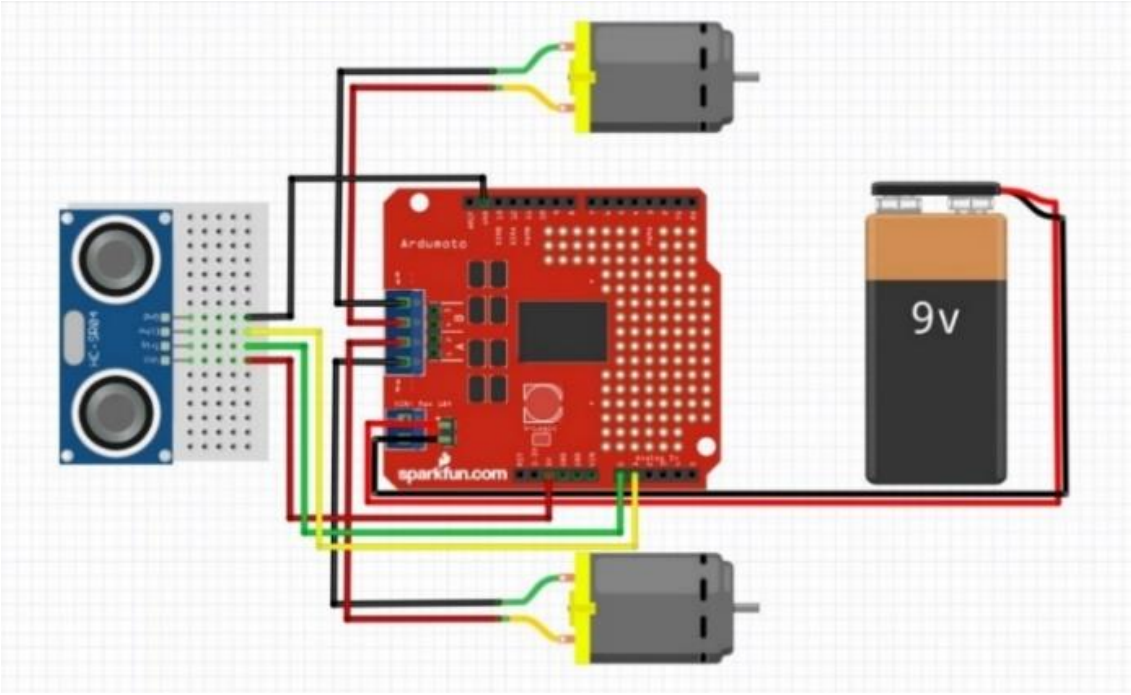
Fonte: Eletrogate, 2020.

4.4.2. Movimentação

A ideia inicial do projeto é que o robô seja montado sobre um carrinho autônomo, composto por dois motores DC - *Direct Current* (corrente contínua) que serão controlados por um *Shield* para motores, que deve controlar a velocidade do motor. Cada motor será independente portanto, para alterar sua direção o Arduino deve apenas desacelerar um dos dois motores para rotacionar o carrinho possibilitando curvas mais acentuadas e menor consumo de energia por não necessitar de 4 motores.

Este *Shield* também estará a cargo de receber informações de um sensor ultrassônico, que servirá como visão para o mesmo, lhe guiando por onde não existem obstáculos próximos. Este dispositivo por sua vez é ligado ao Arduino Uno que fará o controle de todas estas funções, e ditará o funcionamento dele. O *Shield* utilizado na Figura 12 não será o utilizado no projeto final, está sendo utilizado para demonstrar como seria este sistema e suas ligações. Todo este conjunto é alimentado por uma bateria, que na Figura 26 é uma bateria de 9 V apenas para demonstrar a ideia do projeto.

Figura 26 - Esquema 1, movimentação

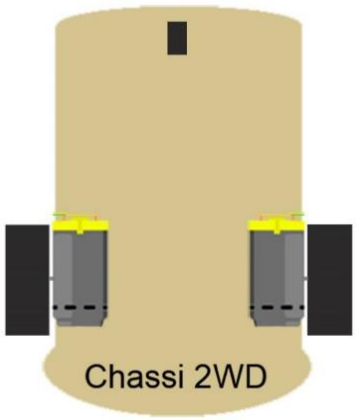


Fonte: Autores, 2020.

4.4.3. Protótipo do veículo

O protótipo do veículo que comportará todo o hardware que foi feito sobre um chassi 2WD. Baseando-se em protótipos já desenvolvidos de veículos autônomos, ou seguidores de linha, projetou-se como será a ideia inicial do nosso projeto. O veículo terá apenas duas rodas que estão ligadas em seus respectivos motores e uma roda auxiliar que não possui motor, a Figura 27 mostra a parte inferior do chassi com as duas rodas ligadas aos motores e a roda auxiliar.

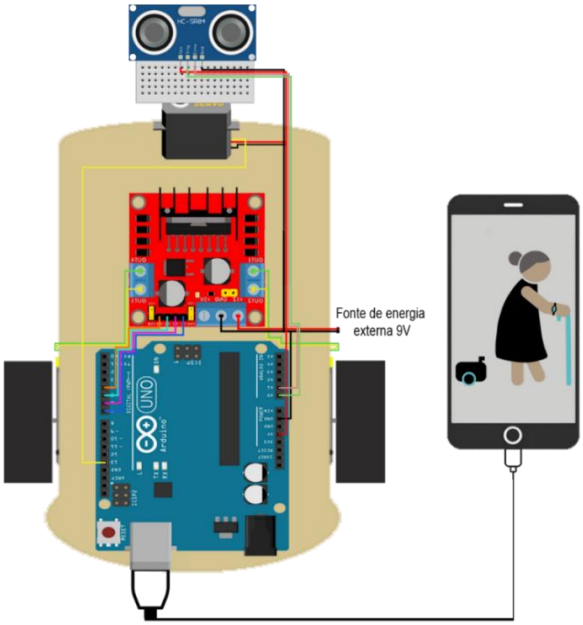
Figura 27 - Parte inferior do chassi.



Fonte: Autores, 2020.

Na parte superior está presente todo o hardware responsável pelo controle do autônomo, na figura 28 é possível ver todos os componentes apresentados anteriormente junto a um *smartphone* já posicionados e conectados. As cores dos fios seguem uma lógica, os fios pretos são conexões de 5v fornecidas pelo Arduino para outros componentes, sendo eles, o servo motor e o sensor ultrassônico. Uma conexão de 9 V que alimenta o *Shield* da ponte H, e conectados a ela tem-se fios verdes e amarelos que são responsáveis por alimentar os motores DC. Os fios restantes são conexões que fornecem informações para a placa Arduino diretamente do sensor e do servo motor. Por fim, um cabo USB cria a conexão do Arduino até o *Smartphone*, onde se encontra o aplicativo que tem como função coordenar a movimentação e realizar as detecções de queda.

Figura 28 - Parte superior do Chassi



Fonte: Autores, 2020.

Além dos componentes eletrônicos, o restante da carcaça do robô foi feita de material reciclável, mais precisamente, papelão como se pode ver na figura 29. Ao redor dos componentes, foi adicionada uma espécie de caixa com uma abertura frontal para a passagem dos cabos, logo acima, foi adicionado um apoio para o *Smartphone*.

Figura 29 - Robô

Fonte: Autores, 2020.

4.4.4. Funcionamento do robô

O robô faz uso das informações geradas pelo aplicativo de *smartphone*, se comunicando e recebendo comandos através de uma porta USB *serial*.

O código implantado no Arduino faz com que o sistema entre em um *loop*, com o intuito de se manter checando a entrada de comandos na porta USB e a proximidade de obstáculos na parte frontal do robô através do sensor ultrassônico.

Quando uma queda é detectada o comando e lido pelo Arduino é tratado, em seguida, é checado qual comando foi recebido. Caso o comando seja o de “emergência” o robô se locomoverá para frente, até que o sensor acuse um obstáculo a menos de 25 centímetros de distância, que no caso, será o usuário caído logo a frente, para que facilite caso o usuário necessite usar a tela para responder se está bem ou não.

4.5. TREINAMENTO DO MODELO DE DETECÇÃO DE OBJETOS

Para o treinamento do modelo de detecção de objetos, utilizou-se o repositório *Darknet*, gratuito e de código aberto. Ele é o algoritmo responsável por treinar modelos, como do tipo YOLO. Por se tratar de uma tarefa computacionalmente complexa, utilizou-se o Google *Colaboratory* que é o responsável pelo processamento durante o treinamento.

O processo completo do treinamento foi feito da seguinte forma:

Coleta de imagens e descarte manual daquelas que não serão usadas ou de baixa qualidade. Para isso, utilizou-se o *dataset* de imagens FPDS (*Fallen Person Dataset*), criado pela Universidade de Calcutá e disponibilizado no artigo *Fallen People Detection Capabilities Using Assistive Robot*. Como o número de pessoas em pé foi insuficiente, utilizou-se o *dataset* de imagens INRIA, disponibilizado pela *Computer Vision Papers*.

1. Etiquetamento de imagens. Utilizou-se o *LabelImg* para cada imagem coletada, etiquetando cada uma delas manualmente a presença de uma pessoa em pé ou deitada ou ambas. Cada imagem gera um arquivo de texto

com as coordenadas das etiquetas. No total, foram etiquetadas 1070 imagens.

2. Envio das imagens etiquetadas para a *Roboflow* para realização do pré-processamento e “aumento” de imagens, gerando por volta de 4000 imagens.
3. *Download* do repositório *Darknet* e envio para o Google *Drive*, junto às imagens coletadas e seus respectivos arquivos de texto.
4. Ajuste de arquivo de configuração do modelo. Um arquivo yolov3-tiny.cfg é necessário para o treinamento. Para este trabalho, utilizou-se o YOLO *Tiny* em vez do YOLO convencional.
5. Configuração do arquivo do *Colaboratory* para utilizar-se de *GPU* e efetuar o treinamento das imagens anexadas no Google *Drive*, utilizando-se do algoritmo *Darknet*.

A cada 1000 iterações de treino, um arquivo *WEIGHTS* é gerado contendo os dados gerados durante o treinamento. Esses arquivos então são comparados de forma automática com imagens de validação, que nos mostram se o treinamento está sendo feito de forma correta, ou não. Por fim, armazenou-se o arquivo *WEIGHTS* em uma pasta no projeto do aplicativo.

4.6. DESENVOLVIMENTO DO APLICATIVO

Como anteriormente já dito, o aplicativo foi escrito em linguagem Java, fazendo o uso de YOLO, Google *Collab* e *Labellmg*. O apelidou-se de RADQ, que significa Robô autônomo detector de quedas.

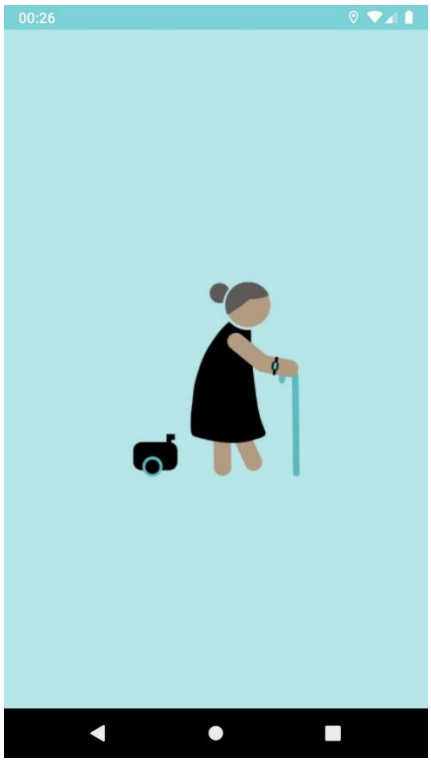
4.6.1. Aplicativo Principal

O aplicativo em questão trata-se do que realizará o monitoramento do usuário juntamente ao robô descrito no capítulo 4.4.

4.6.1.1. Tela de Inicialização

Assim que aberto, o aplicativo exibe uma tela de inicialização contida na *view SplashScreen*, como é mostrado na figura 30. Apesar de não mostrar nada mais, essa *view* também trabalha extraindo dois arquivos essenciais para o funcionamento da detecção de pessoas. Sendo eles o arquivo *CFG* e o *WEIGHTS*, o primeiro trata-se de um arquivo com as configurações necessárias e específicas para este tipo de detecção, já o segundo, contém todos os dados criados, resultado dos treinamentos do modelo. Essa extração dos arquivos é feita pelo método *workOnAdditionalFiles()*.

Figura 30 - Tela de inicialização



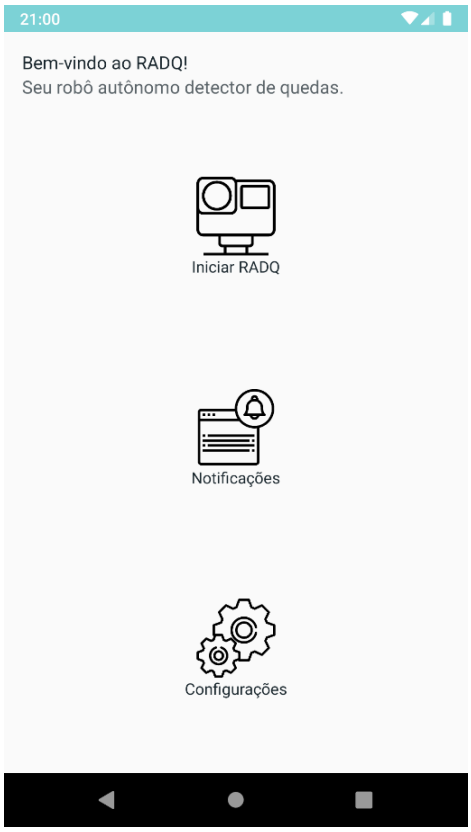
Fonte: Autores, 2020

4.6.1.2. Tela de Menu

Em seguida o usuário é redirecionado para a *view MainActivity*, que se trata da tela principal do aplicativo, como se vê na figura 31. Assim que essa *view* é iniciada uma verificação em segundo plano é feita a fim de averiguar se o aplicativo tem as permissões necessárias para ser executado com sucesso e é feito através do método *CheckPermissions()*. Essas permissões são obrigatórias para toda aplicação e seguem as normas do Sistema Operacional Android.

As permissões necessárias para o funcionamento da aplicação são: *READ_EXTERNAL_STORAGE*; e *CAMERA*. A primeira é principalmente necessária para que a operação de extração dos arquivos na *view SplashScreen* seja feita com sucesso. A segunda é de suma importância, já que o aplicativo gira em torno de uma detecção visual.

Figura 31 - Tela Principal



Fonte: Autores, 2020.

Na tela inicial é possível visualizar botões e seus ícones, sendo eles: Iniciar RADQ (figura 32), Notificações (Figura 33) e Configurações (Figura 34).

Figura 32 - Botão Iniciar RADQ

Fonte: itim2101, Flaticon.

Figura 33 - Botão de Notificações

Fonte: Surang, Flaticon.

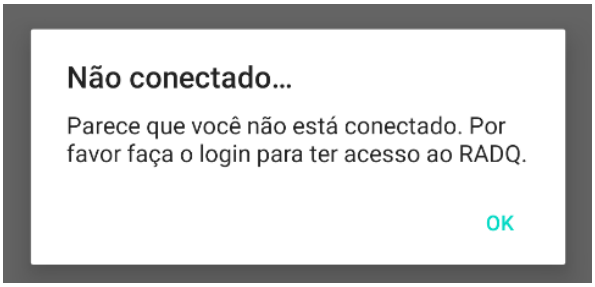
Figura 34 - Botão de Configurações.

Fonte: Freepik, Flaticon.

Ao pressionar o botão Iniciar RADQ alguns procedimentos em segundo plano são feitos para garantir o correto funcionamento do aplicativo. Os principais são: confirmação de login e confirmação de contato na lista de contatos. A primeira basicamente verifica se existe um usuário conectado no aplicativo, enquanto a segunda verifica se este usuário tem algum contato adicionado em sua lista de contatos.

Caso alguma das verificações retorne um problema, como o caso de não existir um contato conectado, uma janela é aberta informando o usuário qual foi o problema encontrado. O método responsável pela abertura da janela é o *alertDialogBox()*, demonstrado na Figura 35.

Figura 35 - Caixa de alerta.



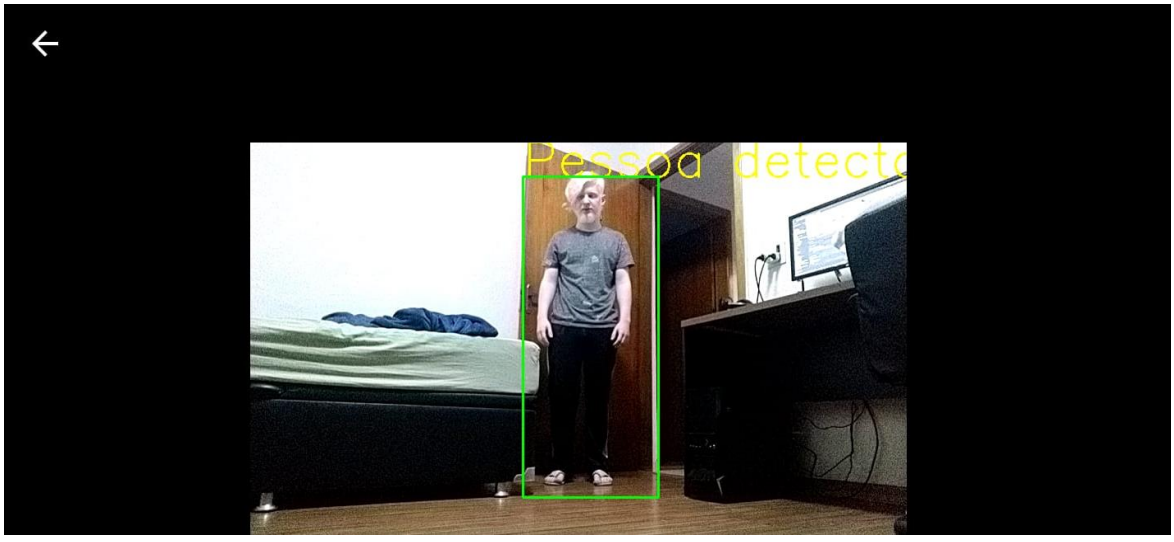
Fonte: Autores, 2020.

4.6.1.3. Tela Iniciar RADQ

Assim que as verificações são feitas e nenhum erro é encontrado, a *view StartRADQ* é aberta e a câmera é inicializada. Essa classe contém os métodos *ServiceConnection*(), *BroadcastReveiver()*, *startService()*, *MyHandler()* e *setFilters()*, que são responsáveis pela conexão com o Arduino que permitirá ao aplicativo controlá-lo. Todos esses métodos são baseados na biblioteca *UsbSerial* explanado no capítulo 2.6 deste trabalho. Eles também se comunicarão com a classe *UsbService* que é a ponte de comunicação entre o Android e o Arduino.

Um dos primeiros métodos a serem chamados é o *onCreate()*, padrão do Android, que é responsável por inicializar todos os serviços, classes e as suas *views* internas, como botões, janelas, textos e a própria câmera, como é mostrado na Figura 36. Possui também a verificação se o sistema fará uso da câmera frontal ou traseira, no caso, deu-se preferência pela câmera frontal, com o intuito de que o usuário possa fazer o uso da tela do *smartphone* caso haja alguma queda. Este método é responsável também pela inicialização do algoritmo de detecção de objetos através do método *initializeDetection()*, que recebe os arquivos *WEIGHTS* e *CFG*.

Figura 36 - *Views*



Fonte: Autores, 2020.

Assim que a câmera é inicializada, as imagens são capturadas e processadas pelo método *onCameraFrame()*. Nem todas as imagens são processadas por conta

da limitação do poder de processamento do *smartphone*, porém a quantidade processada é o suficiente para o funcionamento da aplicação.

O principal trabalho da detecção é feito nesse método. A imagem recebida é transformada em uma matriz e os valores do modelo são carregados, neste caso para pessoas deitadas e pessoas de pé, que são os objetos em questão.

É realizada uma varredura pela matriz a fim de detectar uma das duas opções de valores. Quando um dos objetos é encontrado, um retângulo é criado em volta do objeto, com os pontos identificados, de forma semelhante ao que foi mostrado no *LabelImg*, porém agora de forma matricial e automática.

Caso o objeto encontrado seja de uma pessoa em pé, entra-se em um bloco de código que guardará um número α de alturas. Essas alturas são coletadas a partir do tamanho do retângulo criado. Caso o número de alturas seja igual a α, é chamado o método *calculate*() da classe *CoefficientOfVariationCalculator().* Esse método recebe apenas a lista de alturas.

O cálculo é feito da seguinte forma:

$$Média\ de\ alturas\ = \frac{\sum listaDeAlturas}{\alpha}$$

$$Desvio\ Padrão = \sqrt{\frac{\sum_{\beta}^{\alpha}\sum(altura[\beta] - Média\ de\ alturas)}{\alpha - 1}}$$

$$Coeficiente\ de\ Variação = \frac{Desvio\ Padrão}{Média\ de\ alturas} \times 100$$

Um coeficiente de variação perfeito tem valor zero. Isso quer dizer que: de uma amostra de X valores, todos eles são iguais. Quanto maior o coeficiente de variação, mais os números são variados, portanto para este trabalho, menos confiáveis são esses valores. Estipulou-se que um coeficiente de valor 15 seria o suficiente.

Enquanto os valores do coeficiente forem maiores que 15, nenhum comando é enviado ao Arduino, salvando sua bateria além de reduzir o número de comandos errôneos enviados.

Assim que um valor baixo do coeficiente é registrado, a última altura é recolhida e comparada em três blocos:

- Se a altura for maior que α, é enviado um comando de “ir para trás”, para o Arduino.

- Se a altura for menor que β, é enviado um comando de “ir para frente”, para o Arduino.
- E se a altura estiver entre α e β, é enviado um comando de “parar”, para o Arduino.

Caso o objeto encontrado seja uma pessoa deitada, o processamento entra em outro bloco de código que é responsável por garantir que a queda é real, reduzindo qualquer falso positivo. Para isso é primeiro feita uma verificação se aquela é a primeira vez que foi detectada uma queda, durante aquela execução. Caso verdadeiro, é armazenada a hora em que aquela primeira detecção foi feita.

Se o próximo objeto encontrado for novamente de uma pessoa deitada (Figura 37), é acrescido um valor em uma variável de controle. Se isso se repetir por 5 vezes é verificado então se o tempo entre a primeira detecção e a última, são menores que 10 segundos. Caso maiores, a detecção é descartada como alarme falso.

Caso menores, é enviado um último comando para o Arduino, “emergência”, que fará com que o robô se desloque para frente até encontrar a pessoa caída utilizando o sensor ultrassônico, para facilitar o pressionamento de botões na tela, que serão mostrados pela *view EmergencyActivity*, que será iniciada pelo método *openEmergencyActivity*().

Figura 37 - Pessoa caída.

Fonte: Autores, 2020.

A *view EmergencyActivity* é responsável pela abertura da tela onde encontra-se dois botões, um para o usuário confirmar que está tudo bem e outro para pedir

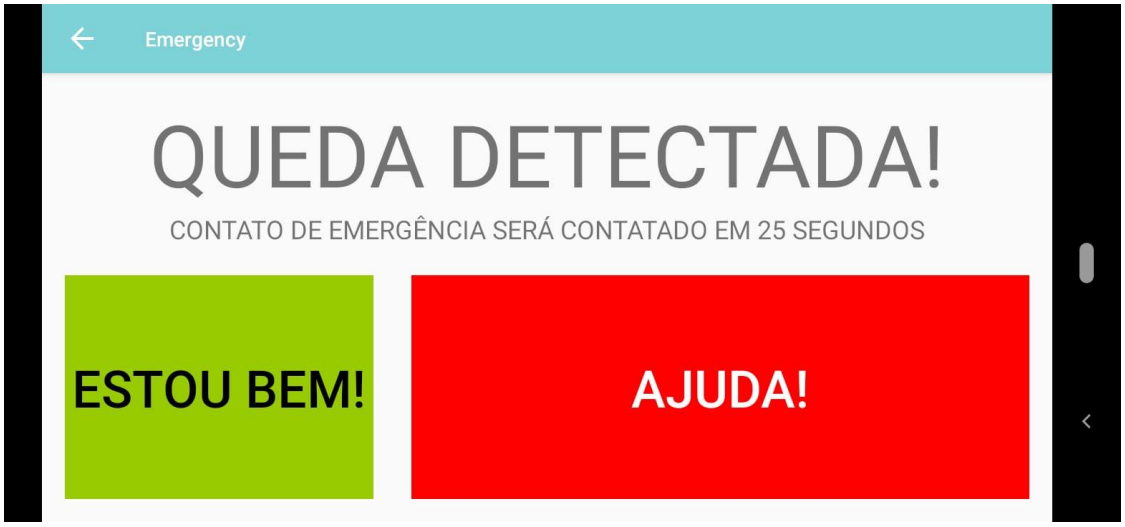
ajuda, demonstrada na figura 38. Neste momento também é acionado um alarme alto, para chamar atenção do usuário ou de qualquer outra pessoa próxima.

Todas as interações do usuário com esta tela são enviadas para o contato de emergência, que são:

- Botão “Estou Bem!” pressionado: um aviso é emitido ao contato de emergência informando que uma queda foi detectada, porém o usuário informou estar bem. Envio imediato da mensagem, alarme desligado e reinicialização da detecção, voltando à tela anterior.
- Botão “Ajuda!” pressionado: um aviso é emitido ao contato de emergência informando que uma queda foi detectada e que o usuário requer ajuda. Envio imediato da mensagem, alarme desligado e um aviso de que o contato de emergência foi contatado.
- Nenhuma ação durante 30 segundos: um aviso é emitido ao contato de emergência informando que uma queda foi detectada e que o usuário não pressionou nenhum dos botões. Envio imediato da mensagem, alarme desligado e um aviso de que o contato de emergência foi contatado.

Por fim, caso a *view StartRADQ* seja fechada, por qualquer motivo que seja: desligamento do celular; fechamento do aplicativo; ou ao voltar ao menu do aplicativo, o que desativa o monitoramento - os contatos de emergência serão avisados. Assim como eles serão avisados caso a *view* seja aberta e a detecção iniciada.

Figura 38 - Queda detectada



Fonte: Autores, 2020.

4.6.1.4. Tela de Notificações

O aplicativo possui uma lista de notificações que são exibidas pela *view NotificationActivity()*. As notificações são coletadas da *view EmergencyActivity* e pela *StartActivity,* ou seja, listará todas as quedas detectadas, tanto as confirmadas, as quedas cujo tempo foi expirado ou o botão de ajuda foi pressionado e todas as vezes que a detecção foi iniciada ou desativada, como mostra a Figura 39.

Figura 39 - Notificações

Fonte: Autores, 2020.

4.6.1.5. Tela de Configurações

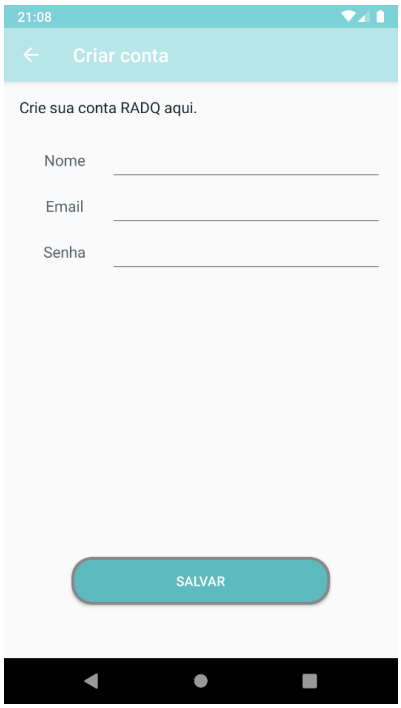
O botão de configurações na tela inicial chama a *view SettingsActivity()*, que dá acesso à tela de Configurações como é mostrado na Figura 40. Nesta tela será possível alternar entre fazer o uso da câmera frontal ou traseira, além de habilitar a visualização das instruções enviadas ao robô.

Figura 40 - Configurações

Fonte: Autores, 2020.

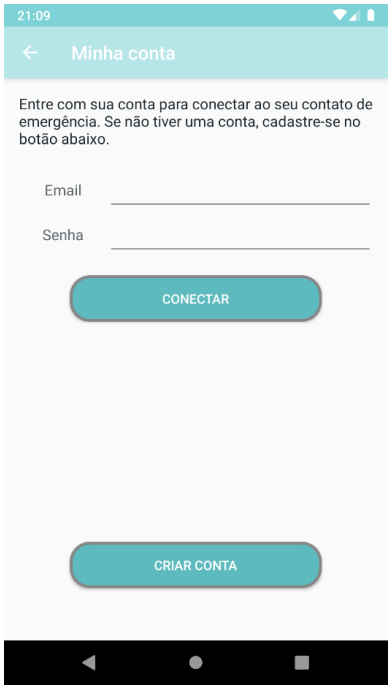
O botão “Minha conta” dá acesso tela de cadastro do usuário como mostra a figura 41, tal cadastro tem como função armazenar os contatos de emergência adicionados. Após o cadastro é possível realizar o login fazendo uso de um e-mail e uma senha, como na figura 42.

Figura 41 - Cadastro do aplicativo



Fonte: Autores, 2020.

Figura 42 -Login do aplicativo

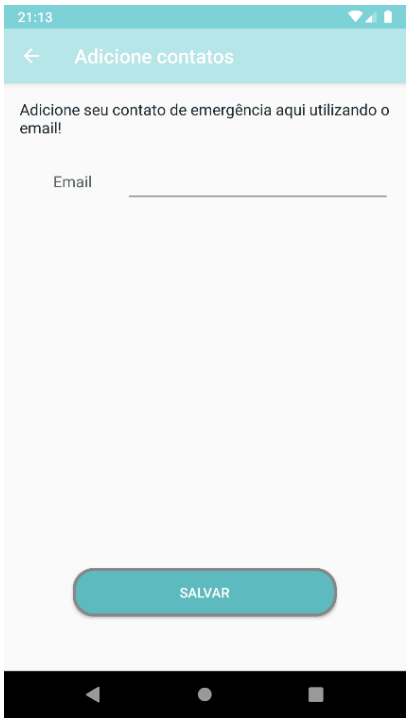


Fonte: Autores, 2020

Ao clicar na opção “Meus contatos” o usuário é redirecionado para a tela onde é possível ver todos os contatos de emergência cadastrados na conta, como mostrado na Figura 43. Se o usuário desejar adicionar mais contatos é possível fazê-lo ao

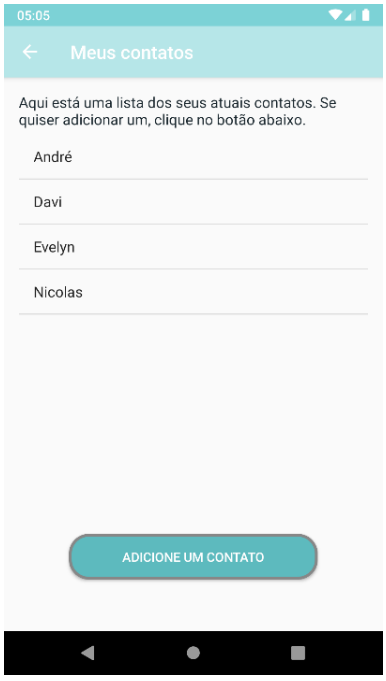
pressionar o botão “Adicionar Contato”, que abrirá a *view AddContactActivity*, como mostrado na Figura 44.

Figura 43- Cadastro contato de emergência



Fonte: Autores, 2020.

Figura 44 - Lista de contatos de emergência



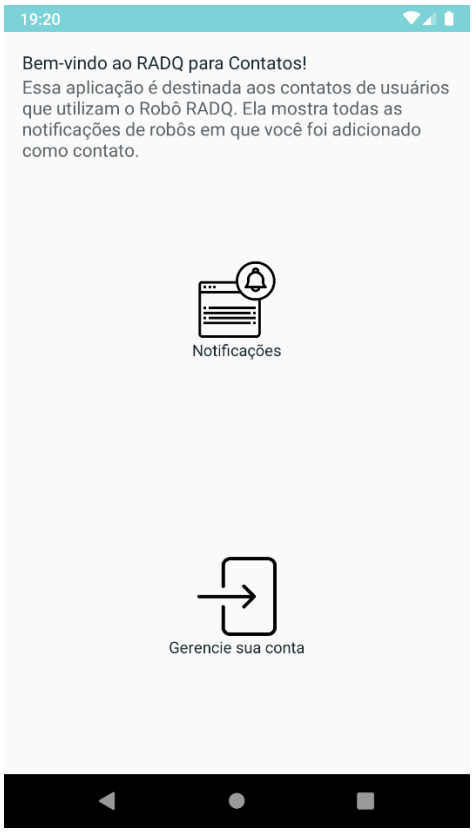
Fonte: Autores, 2020.

4.6.2. Aplicativo do contato de emergência

O aplicativo em questão tem o propósito de receber notificações geradas pelo aplicativo principal onde o contato de emergência foi cadastrado. As pessoas que foram cadastradas como contato devem ter este aplicativo instalado a fim de monitorar o usuário principal.

Assim como o aplicativo principal, este possui a *view SplashScreen* que consiste na tela de inicialização, que é automaticamente redirecionada para a tela principal que possui a *view MainActivity*. Na tela principal são exibidos os ícones “Notificações” e “Gerencie sua conta” (figura 45).

Figura 45 - Tela principal do aplicativo secundário



Fonte: Autores, 2020.

Ao clicar no botão Notificações, o aplicativo chama a *view NotificationsActivity()* e é redirecionado para a tela de Notificações que contém uma lista de notificações geradas pelo RADQ, que consistem em: Quedas que foram detectadas e nenhum botão foi selecionado; Quedas onde o botão “Ajuda!” foi pressionado; Quedas onde o

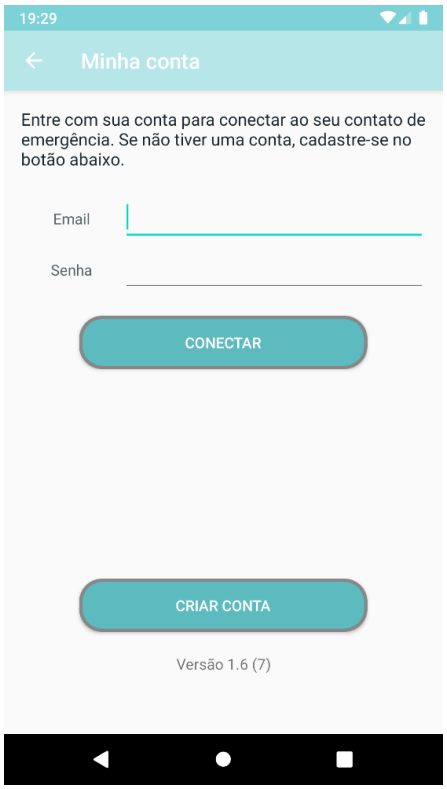
botão “Estou bem!” foi pressionado; desligamento e religamento da detecção de quedas - como mostra a figura 46.

Figura 46 - Tela de notificações do aplicativo secundário

Fonte: Autores, 2020.

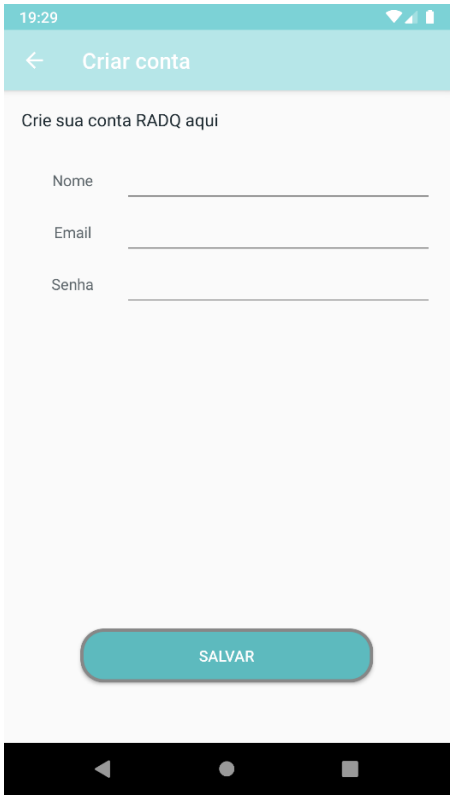
O botão Login chama a *view LoginActivity()* que redireciona o usuário até a tela de *login*. Nesta tela o usuário pode realizar o login através de um *e-mail* e senha (figura 47) ou se cadastrar com nome, *e-mail* e senha (figura 48). Seu objetivo é se vincular ao aplicativo principal para que seja possível receber as notificações. Caso já esteja logado, é exibido o e-mail usado como na figura 49.

Figura 47 - Tela de *Login* do aplicativo secundário.



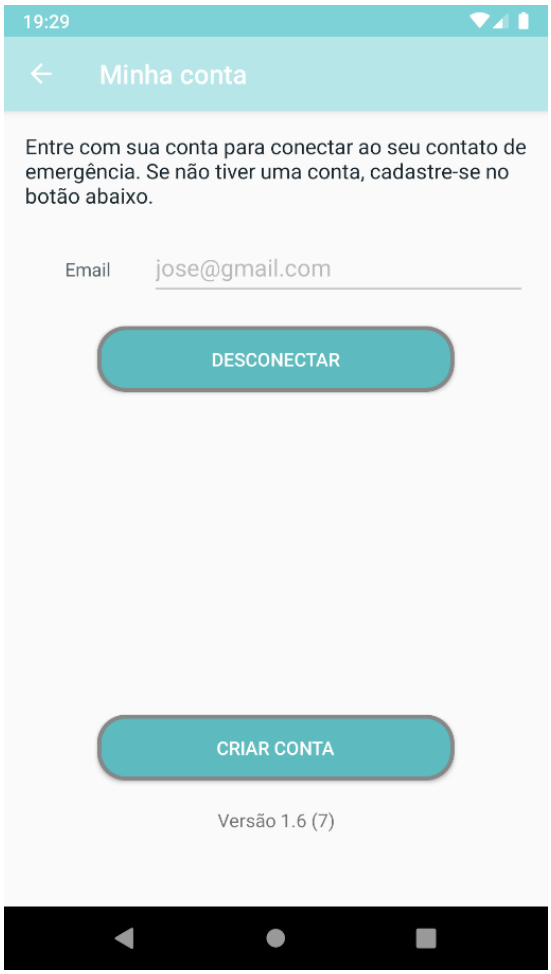
Fonte: Autores, 2020.

Figura 48 - Tela de cadastro do aplicativo secundário.



Fonte: Autores, 2020.

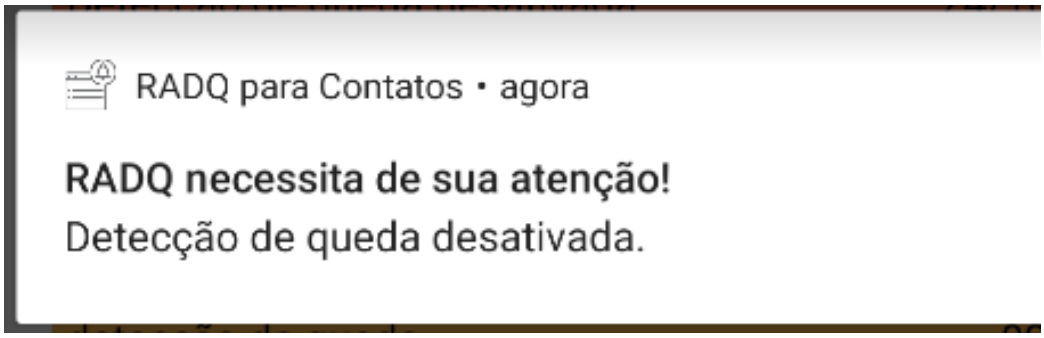
Figura 49 - Tela com o aplicativo logado



Fonte: Autores, 2020.

Por fim, cada nova mudança emitida pelo aplicativo principal, como uma queda, será emitida uma notificação para o contato de emergência mesmo se o aplicativo estiver fechado, como mostra a Figura 50.

Figura 50 - Notificação do RADQ com aplicação fechada



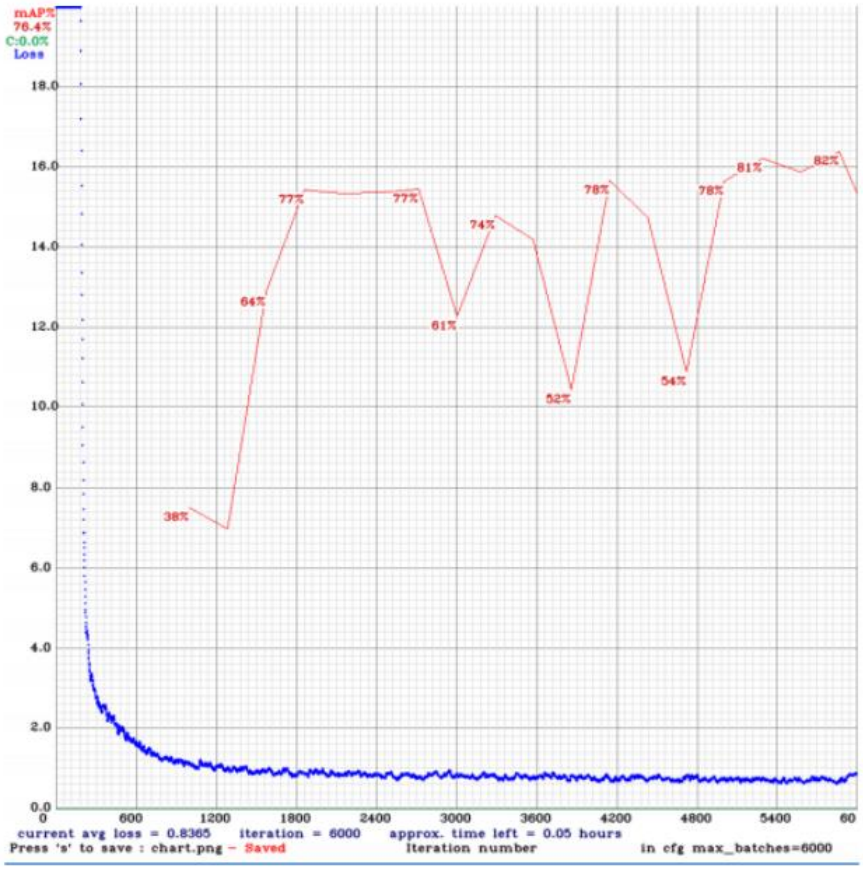
Fonte: Autores, 2020.

5. TESTES REALIZADOS

5.1. DETECÇÃO DE PESSOAS EM FOTOS

Um dos primeiros testes feitos, foi o da detecção de pessoas através da câmera, o modelo usado foi treinado inicialmente com exatas 1070 imagens, e foram validadas 304. A fim de melhorar o treinamento, foi utilizada uma plataforma chamada *Roboflow*, que tem como função realizar alterações nas imagens que já foram testadas através de um pré-processamento de aumento de imagens, como explanado anteriormente. Tal processo cria cópias das imagens a cada alteração, sendo elas: exposição; brilho; *hue*; torção; rotação; e *Zoom*. Ao todo foram verificadas 4312 imagens, e 308 foram validadas. A cada 1000 interações, a plataforma verifica se o treinamento está seguindo de maneira correta, como mostra o gráfico (Figura 51).

Figura 51 - Gráfico de treinamento



Fonte: Autores, 2020.

5.2. DETECÇÃO EM PESSOAS REAIS

Os testes a seguir foram realizados com o aplicativo instalado em um *smartphone* Android. Os primeiros testes foram realizados em um local aberto e bem iluminado, exatas 30 quedas nesse ambiente foram testadas, porém, apenas 24 foram detectadas.

Com o corpo do usuário totalmente aparente, foram detectadas 12 quedas de 16 tentativas, em média, as quedas foram detectadas após 4,16 segundos. Já com o usuário com o corpo apoiado em objetos ou paredes, foram detectadas todas as 4 quedas detectadas, em média o tempo de detecção foi de 5,25 segundos. Com apenas a parte superior do corpo do usuário visível foram detectadas 4 quedas de 5 tentativas, levando em média 3 segundos para a detecção, e com apenas a parte inferior do usuário a mostra, foram detectadas 2 quedas de 5 tentativas, levando em média 4 segundos para serem detectadas.

Em ambiente fechado com pouca iluminação foram testadas 30 quedas, sendo apenas 24 detectadas de fato. Com o corpo do usuário totalmente aparente, foram detectadas 14 quedas de 16 tentativas, demorando em média 3,16 segundos para a detecção. Já com o corpo do usuário apoiado em objetos ou paredes, foram detectadas 3 quedas de 4 tentativas, demorando em média 3,3 segundos para a detecção. Com apenas a parte superior do corpo do usuário aparente, foram detectadas 4 quedas de 5 tentativas, levando em média 2,6 segundos até a detecção. Já com a parte inferior do usuário aparente, foram detectadas 3 quedas de 5 tentativas, levando em média 4,5 segundos para serem detectadas.

O autônomo localiza o usuário fazendo uso do sensor ultrassônico, quando ocorre uma queda, o robô deve se aproximar do usuário, neste quesito foram realizados 10 testes, sendo que, nos 3 primeiros o robô se colidiu contra o usuário, já nos demais 7 testes, o robô localizou o usuário, se dirigiu até ele, e parou antes de se chocar contra o mesmo

6. CONCLUSÃO

Com o intuito de gerar uma maior qualidade de vida para pessoas com mobilidade reduzida ou/e idosos, o desenvolvimento do sistema que foi construído

baseado em Arduino e Android possibilitou uma boa detecção de quedas com o intuito de evitar acidente entre idosos e/ou pessoas com mobilidade reduzida.

O sistema como um todo funciona de forma autônoma. Trata-se de um carrinho robô baseado em Arduino conectado aplicativo disponível em um celular Android, que comanda toda a questão de detecção de queda e movimentação do mesmo. Ao detectar a ocorrência de uma queda, a fim de que não haja maiores consequências por conta da mesma, um segundo usuário é notificado através de um segundo aplicativo.

O aplicativo desenvolvido para Android apesar de se tratar de um sistema distribuído, funciona de forma efetiva e simples, passando em todos os testes realizados com o mesmo.

Ao se tratar da câmera do aplicativo, foram realizados testes tanto em ambiente fechado, quanto em ambiente aberto e bem iluminado, ambos os ambientes apresentaram resultados parecidos, porém, em ambiente fechado com uma menor iluminação a detecção de queda acontece de forma mais rápida, com uma diferença de 71 milissegundos.

Como esperado, a movimentação e o protótipo do autônomo, que tem como objetivo, seguir o usuário e processar informações enviadas através do aplicativo, funciona de forma efetiva e dentro das expectativas. Em 70% dos casos o robô se dirigiu corretamente em direção ao usuário.

Por diversos fatores, como a falta de verba, algumas etapas que foram pensadas inicialmente para o funcionamento do autônomo tiveram de ser alteradas e adaptadas com o propósito de torná-lo funcional mesmo que de forma acessível. Por exemplo, inicialmente foi pensado em usar apenas sensores para a detecção de queda, porém, foi optado pelo uso de uma câmera presente em um *smartphone*.

Ao longo da pesquisa, foram surgindo novas possibilidades que também por falta de verba e por falta de tempo, foram descartadas, porém, por serem usuais, serão apresentadas no tópico a seguir.

6.1. TRABALHOS FUTUROS

Inicialmente foi pensado em usar apenas sensores para a detecção de queda, porém a ideia foi substituída por um *Smartphone*. Recomenda-se que em trabalhos

futuros seja implementada uma câmera avulsa, acompanhado de um módulo *WI-FI* com o intuito de detectar as possíveis quedas e avisá-las ao aplicativo que deverá estar presente no celular do contato de emergência.

Com a troca do *smartphone* por uma câmera, será necessário implantar um painel *touch screen* para que o usuário possa responder em caso de queda, além de definir um contato de emergência e/ou configurar seus alarmes.

É sugerido que haja um melhor treinamento do modelo presente na câmera do aplicativo através da ferramenta YOLO com a finalidade de aperfeiçoar a precisão no quesito detecção de pessoas.

Em caso o autônomo seja usado em uma clínica, seria interessante a existência de um *website*, para que seja possível o monitoramento de vários robôs de forma simultânea, sendo assim, cada paciente teria a possibilidade de ter o seu próprio autônomo.

Por fim, sugere-se a troca da estrutura do autônomo, pois o mesmo só funciona em ambiente plano. Seria interessante se houvesse a implantação de um chassi que fosse capaz de subir escadas, para que não haja a necessidade de o usuário ter de pegá-lo em mãos caso precise se locomover para outro andar, além de que, quedas em escadas costumam ser frequentes.

7. REFERÊNCIAS

Adrian Rosebrok, **YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS**, PyimageSearch, 2020. Disponível em: **<**https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/**>.** Acesso em 02 de outubro de 2020.

Arduino Playground, 2018. Disponível em: <https://playground.arduino.cc/Portugues/HomePage/>. Acesso em: 21 de março de 2020.

Apple Support, **Use fall detection with Apple Watch**, 2019. Disponível em<https://support.apple.com/en-gb/HT208944>.Acesso em: 8 de março de 2020.

Aquarela, **Datasets, o que são e como utiliza-los?**, 2018. Disponível em: <https://www.aquare.la/datasets-o-que-sao-e-como-utiliza-los/>. Acesso em: 03 de outubro de 2020.

Allan Mota, **O que é Arduino e como funciona?** 2017. Disponível em: <https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/>. Acesso em: 21 de março de 2020.

Canal IBM research, **IBM Research and Cera Care Test AI and Lidar for Elder Care**, Youtube, 2019. Disponível em:<https://www.youtube.com/watch?v=fgCDFYOjX-E> Acesso em: 8 de março de 2020.

Carlos Alberto Gasperin, **Firebase: O que é e como funciona**, Micreiros, 2020. Disponível em: <https://micreiros.com/firebase-o-que-e-e-como-funciona/>. Acesso em: 13 de outubro de 2020.

Gilles Mazars, **Datasets,** Computer Vision Papers, 2007-2020. Disponível em: <http://www.cvpapers.com/datasets.html>. Acesso em: 04 de outubro de 2020.

Felipe Herranz, **UsbSerial**, GitHub, 2014. Disponível em: <https://github.com/felHR85/UsbSerial>. Acesso em: 26 de setembro de 2020.

Fidel Forato, **Pets robôs são aliados no tratamento de demência e autismo no Brasil**, Canaltech, 2019. Disponível em <https://canaltech.com.br/saude/pets-robos-sao-aliados-no-tratamento-de-demencia-e-autismo-no-brasil-152599/>. Acesso em: 08 de março de 2020.

Fabio Souza, **Arduino Uno,** Embarcados, 2013. Disponível em <https://www.embarcados.com.br/arduino-uno/> Acesso em: 21 de março de 2020.

IBGE**, Pesquisa nacional de saúde**, 2013. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv94522.pdf>. Acesso em: 21 de março de 2020.

Inclusão360org, **Robô cuidadora ajuda no tratamento de idosos**, 2019. Disponível em<https://www.inclusao360.org/melhor_idade/robo-cuidadora-ajuda-no-tratamento-de-idosos/>. Acesso em: 08 de março de 2020

Jhon Gonzalez, **Sign Language Recognition Software using C# OpenCV (CvBlob-ConvexHull - SVM)**, Youtube, 2011. Disponível em: <https://www.youtube.com/watch?v=cxHMgl2_5zg>. Acesso em: 24 de outubro de 2020.

Joseph Redmon, **YOLO: Real-Time Object Detection**, PJREDDIE, 2016. Disponível em <https://pjreddie.com/darknet/yolo/>. Acesso em: 10 de outubro de 2020.

Joseph Nelson, **Getting Started with LabelImg for Labelimg Object Detection Data,** Roboflow, 2020. Disponível em: <https://blog.roboflow.com/labelimg/>. Acesso em 04 de outubro de 2020.

Juan Arias, **Os 45 milhões de brasileiros com deficiência física são os novos párias,** El País, 2019. Disponível

em:<https://brasil.elpais.com/brasil/2019/05/08/opinion/1557340319_165119.html>. Acesso em: 27 de março de 2020.

Juri Strumpflohner, **UML Use Case “extends” and “include relationships**, 2009. Disponível em:<https://juristr.com/blog/2009/03/uml-use-case-extend-and-include/>. Acesso em: 19 de abril de 2020.

Jupyter Project, **Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages**, 2014. Disponível em : <https://jupyter.org/>. Acesso em 24 de outubro de 2020.

Levindo Neto, **Include e Extend em Diagramas de Casos de uso | Engenharia de Software**, 2016. Disponível em: <https://www.youtube.com/watch?v=LGkzco2pfyc>. Acesso em: 19 de abril de 2020.

Marilyn Rantz, **Intelligent Sensor System for Early Illness Alerts in Senior Housing**, University of Missouri, 2013-2017. Disponível em:< https://www.eldertech.missouri.edu/projects/intelligent-sensor-system-for-early-illness-alerts-in-senior-housing/>. Acesso em: 8 de março de 2020.

Mouse 4all, **Accessibility for Android**, 2018-2020. Disponível em: <https://mouse4all.com/en/>. Acesso em: 25 de outubro de 2020.

Nvidia Developer, **OpenCV,** 2020. Disponível em: <https://developer.nvidia.com/opencv>. Acesso em: 25 de outubro de 2020.

Nações Unidas Brasil, **Mundo terá 2 bilhões de idosos em 2050; OMS diz que ‘envelhecer bem deve ser prioridade global’**, 2014. Disponível em: <https://nacoesunidas.org/mundo-tera-2-bilhoes-de-idosos-em-2050-oms-diz-que-envelhecer-bem-deve-ser-prioridade-global/>. Acesso em: 21 de março de 2020.

Petrobras, **Projeto usa robótica para despertar interesse de crianças pela ciência**, 2019. Disponível em: <https://medium.com/petrobras/projeto-usa-robótica-para-despertar-interesse-de-crianças-pela-ciência-6211a57b0d56>. Acesso em: 08 de março de 2020.

Pedroti M., C.M. Carvalho, G.A.O. Brito, D.C.A. Melo, A.L. Gamas, C.J.G. Pinheiro, **O uso de sensores e Arduino para otimização da pesquisa laboratorial,** 2018. Disponível em: <https://www.proceedings.blucher.com.br/article-details/o-uso-de-sensores-e-arduino-para-otimizao-da-pesquisa-laboratorial-28302>. Acesso em: 21 de março de 2020.

PopulationPyramid, **Pirâmides Populacionais do Mundo desde 1950 até 2100**, 2020. Disponível em: <https://www.populationpyramid.net/pt/brasil/2020/>. Acesso em: 27 de março de 2020.

Reis, Fábio, **Arduino - Conhecendo os Shields,** 2015. Disponível em: <http://www.bosontreinamentos.com.br/eletronica/arduino/arduino-conhecendo-os-shields/>. Acesso em: 28 de março de 2020.

Rockcontent, **Conheça Firebase: A ferramenta de desenvolvimento e análise de aplicativos mobile**, 2019. Disponível em:< https://rockcontent.com/br/blog/firebase/>. Acesso em: 13 de outubro de 2020.

Saturnino Maldonado Bascón, Cristian Iglesias, Pilar Martín, Sergio Lafuente Arroyo, **Fallen People Detection Capabilities Using Assistive Robot**, MDPI, 2019. Disponível em: <https://www.mdpi.com/2079-9292/8/9/915>. Acesso em: 19 de setembro de 2020.

Todos pela educação, **com nova margem de corte, IBGE constata 6,7% de pessoas com deficiência no Brasil,** Estadão, 2018. Disponível em: < https://educacao.estadao.com.br/blogs/educacao-e-etc/com-nova-margem-de-corte-ibge-constata-67-de-pessoas-com-deficiencia-no-brasil/>. Acesso em: 27 de março de 2020.

Tzutalin, **labelImg**, GitHub, 2016. Disponível em: <https://github.com/tzutalin/labelImg>. Acesso em: 03 de outubro de 2020.

Unisinos, **O envelhecimento populacional segundos as novas projeções do IBGE,** 2018. Disponível em: <http://www.ihu.unisinos.br/78-noticias/582356-o-envelhecimento-populacional-segundo-as-novas-projecoes-do-ibge>. Acesso em: 21 de março de 2020.

Vishakha Lal, **Google Colab — The Beginner's Guide,** Medium, 2018. Disponível em: <https://medium.com/lean-in-women-in-tech-india/google-colab-the-beginners-guide-5ad3b417dfa>. Acesso em 02 de outubro de 2020.

Zora Robotics, **Who i'm** 2018. Disponível em: <http://www.zorarobotics.be/index.php/en/who-am-i > Acesso em: 08 março de 2020.