UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

VITOR CONSTANTINO SCARDUA

INSTRUMENTAÇÃO E DESENVOLVIMENTO DE COMPORTAMENTO PARA UM ROBÔ DE INTERAÇÃO SOCIAL COM CRIANÇAS COM TRANSTORNO DO ESPECTRO AUTISTA

VITÓRIA – ES

JULHO/2018

VITOR CONSTANTINO SCARDUA

INSTRUMENTAÇÃO E DESENVOLVIMENTO DE COMPORTAMENTO PARA UM ROBÔ DE INTERAÇÃO SOCIAL COM CRIANÇAS COM TRANSTORNO DO ESPECTRO AUTISTA

Parte manuscrita da Proposta de Projeto de Graduação do aluno Vitor Constantino Scardua. Apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Teodiano Freire Bastos Filho

Coorientador: Eng. Dr. Carlos Torturella Valadão

VITÓRIA – ES

JULHO/2018

VITOR CONSTANTINO SCARDUA

INSTRUMENTAÇÃO E DESENVOLVIMENTO DE COMPORTAMENTO PARA UM ROBÔ DE INTERAÇÃO SOCIAL COM CRIANÇAS COM TRANSTORNO DO ESPECTRO AUTISTA

Parte manuscrita da Proposta de Projeto de Graduação do aluno Vitor Constantino Scardua. Apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 13, de Julho de 2018.

COMISSÃO EXAMINADORA:

Prof. Dr. Teodiano Freire Bastos Filho
Universidade Federal do Espírito Santo
Orientador

Carlos Torturella Valadão

Eng. Dr. Carlos Torturella Valadão
Universidade Federal do Espírito Santo
Coorientador

Caldeira

Prof. Dra. Eliete Maria de Oliveira Caldeira
Universidade Federal do Espírito Santo
Examinadora

André Ferreira

Prof. Dr. André Ferreira
Universidade Federal do Espírito Santo
Examinador

VITÓRIA – ES

JULHO/2018

RESUMO

Atualmente, pesquisas com robôs para interação com crianças com Transtorno do Espectro Autista (TEA) vêm mostrando crescimento. Os objetivos deste Projeto de Graduação são a instrumentação e desenvolvimento de comportamento "passeio" do robô N-MARIA (*New-Mobile Autonomous Robot for Interaction Autistics*) desenvolvido na Universidade Federal do Espírito Santo (UFES). Inicialmente, foi realizado um estudo para avaliar os tipos de sensores disponíveis no mercado para, assim definir os mais adequados, seguros e eficientes para auxiliar a interação entre o robô e a criança. Por meio de testes, foram definidos os locais mais adequados para a instalação dos sensores no robô. Os experimentos com o robô foram realizados para o comportamento de "passeio", onde o robô realizou passeio com um voluntário pelo ambiente. Possíveis riscos que o robô pudesse oferecer foram avaliados, também foi desenvolvida uma rotina de segurança para parada imediata do robô. Por fim, foram realizados testes de validação do sistema como um todo.

Palavra-chave: Transtorno do espectro autista, Autismo, Robô, Interação, Instrumentação, Sensores.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
TEA	Transtorno do Espectro Autista
UFES	Universidade Federal do Espírito Santo
UART	Universal Asynchronous Receiver Transmitter
NUC	Next Unit Computing
N-MARIA	New-Mobile Autonomous Robot for Interaction Autistics
I2C	Inter-Integrated Circuit
NTA	Núcleo de Tecnologia Assistiva
IRQ	Interrupt ReQuest
PWM	Pulse Width Modulation
SDK	Software Development Kit
FSR	Force-Sensing Resistor
PHP	PHP: Hypertext Protocol

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

1 APRESENTAÇÃO E OBJETIVO DE PESQUISA

Segundo a Organização Mundial de Saúde (OMS), o Transtorno do Espectro Autista (TEA) atinge cerca de 70 milhões de pessoas em todo o mundo. As crianças com tal transtorno possuem algumas desordens para realizar interações sociais, tais como dificuldades de interpretar os comportamentos humanos naturais, como emoções, expressões faciais e afins. Tal falta de interação social pode prejudicar o desenvolvimento dessas crianças como um todo, podendo resultar no desenvolvimento de atitudes, por vezes agressivas ou reclusivas (ONU BRASIL, 2015).

O Instituto de Pesquisa e Tratamento de Distúrbios do Espectro Autista (2013) observou uma melhora nas interações sociais de crianças com TEA quanto ao uso de tecnologias, mostrando uma maior aceitação destas por parte das crianças. Devido a essa melhora nas interações, foram desenvolvidas várias tecnologias, sendo que uma delas são robôs que interagem de diversas formas com essas crianças, podendo ser por meio de toque (para identificação de texturas), até interações mais complexas, onde o robô fica responsável por estimular, através de brincadeiras e movimentos, a fala da criança (AUTISM RESEARCH INSTITUTE, 2013).

Várias universidades pelo mundo realizam pesquisas sobre a interação de crianças com TEA e robôs. Entre os vários robôs existentes o ZENO, desenvolvido pela Universidade do Texas em Arlington (Estados Unidos), que utiliza sensores presentes na roupa da criança e, através deles, realiza interações utilizando fala e gestos. A mesma universidade desenvolveu também o MILO, outro robô de interação, porém, diferentemente do ZENO, o MILO utiliza câmeras, posicionadas na região dos olhos do robô, que gravam a reação das crianças, além de captar, através de sensores conectados à bochecha das mesmas, informações sobre batimento cardíaco e expressões faciais (TRUCKER, 2015).

Robôs de diversos tamanhos e formas foram desenvolvidos para interação com crianças com TEA, tais como o *Kaspar*, criado na Universidade Hertfordshire, na Inglaterra, que já é utilizado em tratamentos efetivos. *Kaspar* possui o tamanho de uma criança de 4 anos, interage através de frases simples e repetitivas, e vem mostrando resultados muito positivos (BBC, 2014).

Este trabalho busca fazer parte do leque de estudos e desenvolvimentos tecnológicos nesta área de pesquisa, tendo como alvo o desenvolvimento de habilidades sociais de crianças com TEA, tais como envolvimento e atenção compartilhada, imitação, interpretação de comandos, redução

à aversão ao toque, além de outras interações sociais através da utilização de um robô em um comportamento de “passeio” que incentive a criança a desenvolver essas habilidades sociais. O robô é equipado com vários sensores como *encoders,* que geram informações de posição e orientação para o controlador do robô, sensor laser RP LIDAR, para a detecção da criança no ambiente e para permitir ao robô uma aproximação ou afastamento em relação à criança, sensores de toque, capazes de identificar quais partes do robô foram tocadas pela criança, e sensores de força, que identificam a força exercida no toque.

1.1 Estrutura do Projeto de Graduação

No primeiro capítulo desse Projeto de Graduação é contextualizada a utilização da robótica para o Transtorno do Espectro Autista (TEA), além dos sensores a serem utilizados neste trabalho. Já no segundo capítulo, são apresentadas as justificativas seguidas dos objetivos no terceiro capítulo. No quarto capítulo é mostrado o embasamento teórico utilizado para este trabalho e posteriormente no quinto e sexto capítulos, são explicadas a instrumentação e a montagem do robô N-MARIA. Por último, no sétimo capítulo tem-se os experimentos e resultados obtidos e no oitavo capítulo são apresentadas as conclusões e trabalhos futuros. Nos apêndices são disponibilizados os códigos implementados neste trabalho.

1.2 Justificativa

Atualmente, a utilização de robôs para a interação com crianças com TEA mostra-se muito eficiente. Tais robôs têm sido desenvolvidos por universidades por todo o mundo, onde esta área da robótica tem aumentado sua expressividade ao longo do tempo. Como exemplo desses robôs, pode-se ressaltar um já citado anteriormente, o *Kaspar,* robô inglês que já se encontra em estado avançado de estudos e já foi utilizado em várias terapias fora do mundo universitário, tendo grande aceitação por parte das crianças e terapeutas (WAINER et al., 2014).

O Núcleo de Tecnologias Assistivas (NTA) da Universidade Federal do Espírito Santo (UFES) possui pesquisas em várias áreas da robótica, sendo que uma delas tem o objetivo de construir

um robô que auxilie o terapeuta na interação com crianças com TEA, assim possibilitando um melhor desenvolvimento social da criança.

O robô N-MARIA (*New-Mobile Autonomous Robot for Interaction with Autistics*) encontra-se em fase de desenvolvimento no NTA/UFES, o qual é uma versão melhorada do robô anterior MARIA, que possui o mesmo acrônimo do N-MARIA, porém sem a palavra "New". Para esta nova versão, estão sendo implementados e incluídos novos componentes, dispositivos e sensores, tais como um rosto dinâmico que, atualmente, é exibido em um *tablet touchscreen* e possui feições que se alternam mediante interações com a criança com TEA e comandos do terapeuta; além de um sistema de câmeras RGB e térmica instalados no robô para capturar imagens da criança; um sensor de distância a laser RP LIDAR; e sensores de força e toque.

Assim, atualmente, o robô N-MARIA possui sensores que permitem quantificar a interação da criança que estiver interagindo com ele. Por exemplo o sensor laser RP LIDAR SLAMTEC A2M8 permite, após o tratamento dos seus dados, identificar a localização da criança em uma faixa de 360º e até 8 metros de distância do sensor, possibilitando assim ao robô efetuar uma aproximação ou afastamento da criança para que haja a interação. Tais movimentos de aproximação e afastamento devem ser auxiliados também pelos sensores de toque e força para melhorar a interação por parte da criança. O foco deste trabalho está nesta parte, pois tendo em vista a possibilidade de uma melhor interação por parte do robô N-MARIA com crianças com TEA, foi realizado o estudo e a implementação de sensores de toque e força, de tal forma que permitam detectar e quantificar a interação com a criança através de um comportamento de "passeio" e, desta maneira, incentivar um desenvolvimento da mesma, desenvolvimento este cognitivo e social (VALADÃO et al., 2016).

1.3 Objetivo Geral:

- Especificar e implementar sensores de toque e força para um robô de interação com crianças com Transtorno do Espectro Autista (TEA), além de um comportamento de "passeio", com a inclusão de rotina de segurança para a interação com o robô.

1.4 Objetivos Específicos:

- Definir um ou mais sensores de toque a serem implementados no robô;
- Definir um ou mais sensores de força a serem implementados no robô;
- Definir tipos de materiais que garantam uma interface segura entre a criança e os sensores;
- Definir os locais mais adequados do robô para a instalação dos sensores;
- Desenvolver formas de capturar e armazenar os dados obtidos pelos sensores;
- Desenvolver um comportamento de “passeio”, para avaliar a interação com a criança;
- Definir possíveis riscos que o robô possa vir a oferecer à criança;
- Projetar medidas emergenciais para minimizar possíveis riscos à criança;
- Realizar testes de validação dos sistemas propostos.

2 EMBASAMENTO TEÓRICO

Este capítulo descreve alguns aspectos de robôs para interação social, a forma de interação entre pessoas e robôs, além de como estes são utilizados para terapias com crianças com TEA. Também discorre sobre possíveis sensores para interação, critérios para desenvolvimento de um comportamento de "passeio", além de uma rotina de segurança e programas computacionais para captação de dados.

2.1 Transtorno do Espectro Autista (TEA)

Crianças normalmente necessitam de atenção especial por se encontrarem em fase de desenvolvimento. O Transtorno do Espectro Autista (TEA), quando diagnosticado, caracteriza-se, em geral, pela criança apresentar sensibilidade a sons, luzes e toques, o que torna, por muitas vezes, um desafio para a criança realizar interações sociais como um todo (MATTOS, 2011). O Transtorno do Espectro Autista (TEA) compõe-se de uma gama de síndromes diversas que possuem perturbações no desenvolvimento neurocomportamental como característica principal. Dentro de tais perturbações no desenvolvimento, podem-se apresentar déficits de diversas habilidades importantes para interação social e desenvolvimento cognitivo, como dificuldade de comunicação (habilidade de falar), dificuldade no domínio da linguagem, não entendimento de figuras de linguagem, e padrão de comportamento repetitivo e restrito, sendo que tais características podem ser apresentadas em conjunto ou isoladamente (KLIN, 2006; VARELLA, 2014).

Eugene Bleuler, no século XX, utilizou o termo "autismo", porém, como apresentava um conjunto de distúrbios de comunicação e perda do contato com a realidade, este foi correlacionado à esquizofrenia. Leo Kanner, em 1943, ao observar onze crianças que tinham em comum a inabilidade nata de estabelecer contato afetivo e interpessoal, também utilizou o termo "autismo" para defini-las. Em 1944, Hans Asperger descreveu o mesmo tipo de perturbação em crianças, porém com características mais brandas. Kanner e Asperger foram pioneiros no reconhecimento do autismo como uma perturbação distinta das outras patologias, marcada por isolamento social inato que persistia por toda adolescência e idade adulta (BRAGA, 2010). Hoje em dia o que se conhece por "autismo" se tornou Transtorno do Espectro

Autista (TEA), recebendo o nome de espectro, pois envolve situações e apresentações muito diferentes umas das outras, numa graduação que vai da mais leve à mais grave. Todas, porém, em menor ou maior grau, estão relacionadas com a dificuldade de comunicação e relacionamento social (VARELLA, 2014).

O TEA, quanto ao seu quadro clínico, pode ser classificado, em termos gerais, de três formas: a primeira delas é o Autismo clássico, onde o grau pode variar bastante, porém, os portadores são introvertidos, não interagem com o ambiente nem mantém contato visual e, embora consigam falar, não utilizam a fala como ferramenta de comunicação. Em casos mais severos, apresentam completa ausência de contato interpessoal e, sendo assim, são crianças isoladas, não aprendem a falar e nem olham para pessoas, além de repetirem movimentos estereotipados e apresentarem déficit cognitivo importante. Outra classificação dentro do TEA é o Autismo de alto desempenho (antes chamado de síndrome de Asperger), onde as pessoas acometidas apresentam a mesma dificuldade de interação das que possuem o Autismo clássico, porém num nível bem mais brando, de forma que conseguem utilizar a fala para comunicação e apresentam extrema inteligência em assuntos específicos de interesse, confundidos por vezes com gênios. Por último, há os distúrbios globais do desenvolvimento sem outras especificações, casos em que os portadores possuem dificuldade de interação social e comunicação, porém, o conjunto de sintomas não são claros o suficiente para incluí-los em nenhuma das categorias do transtorno, o que torna o diagnostico muito mais difícil (VARELLA, 2014).

2.2 Robôs e interação social

Atualmente, robôs são utilizados cada vez mais no dia a dia das pessoas para auxiliá-las em diversas tarefas. Um dos focos mais comuns que vem crescendo é a utilização de robôs para ajudar na interação social, onde são encarregados por motivar, treinar, supervisionar e acompanhar pessoas. A interação desse tipo de robô com as pessoas é cada vez mais satisfatória, de forma a auxiliá-los no desenvolvimento das habilidades ligadas ao comportamentos humano (FONG; NOURBAKHSH; DAUTENHAHN, 2003).

Robôs de assistência social são relativamente recentes, porém, é uma área que vem apresentando um rápido crescimento. Enquanto robôs assistivos geralmente são focados em

reabilitação, precisão de movimento e repetição, características importantes para robôs com engajamento físico com pessoas, robôs voltados à interação social enfatizam sua expressividade emocional, o envolvimento com o usuário, sua aparência física e robustez durante a interação, pois tais robôs devem auxiliar o usuário, além de treinar, motivar e influenciar mudanças em seu comportamento, fazendo com que seja necessário que as interações incluam profissionais das áreas de psicologia, fisiologia e sociologia ao mesmo tempo (SCASSELLATI; HENNY ADMONI; MATARIĆ, 2012).

A utilização de robôs para auxílio de crianças com TEA vem mostrando consideráveis melhoras no nível de atenção, engajamento e interação social por parte das mesmas (SCASSELLATI; HENNY ADMONI; MATARIĆ, 2012). Robôs projetados para interagir com crianças com TEA têm por objetivo o desenvolvimento de suas habilidades cognitivas, comportamentais e sociais, a fim de auxiliar terapeutas e cuidadores nesta tarefa. A aceitação das crianças com TEA para com robôs é positiva, tendo em vista que os robôs são mais simples e previsíveis do que humanos, por apresentarem menos informações sociais, e, assim, são mais fáceis de serem entendidos por crianças com TEA (DUQUETTE; MICHAUD; MERCIER, 2008; ROBINS et al., 2010).

2.3 Sensoriamento em robôs para interação social

Os sensores permitem que robôs possam interagir com o ambiente e com os seus meios. Robôs desenvolvidos para interação social necessitam também de interação com o ambiente ao redor. O uso de sensores permite aos robôs um maior nível de “inteligência” para lidar com o ambiente e pessoas, e isso vem sendo alvo de pesquisas intensas no campo da robótica.

Para que um robô realize tal interação, com a realização de tarefas complexas, são necessários mecanismos de controle mais flexíveis, quando comparados com sistemas de máquinas pré-programadas. Sensores podem também ser mais facilmente adaptáveis a uma maior variedade de tarefas e, desta forma, pode-se atingir um maior grau de universalidade, algo importante e necessário em interações humano-robô (OLIVEIRA et al., 2017).

2.4 Sensores

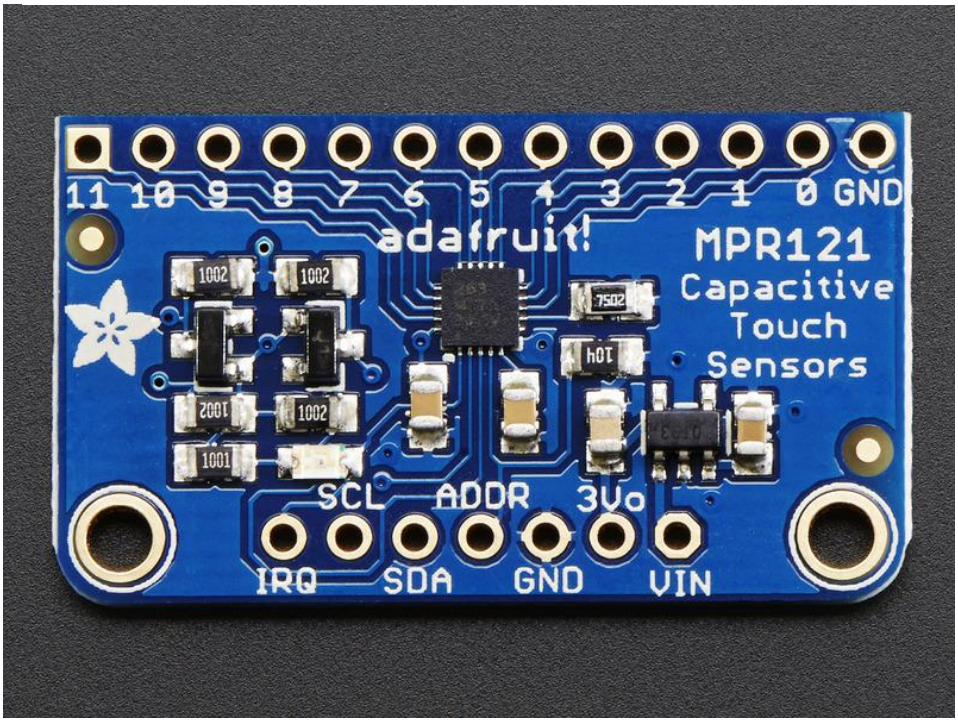
Vários sensores que possibilitem detectar a interação com crianças com TEA foram estudados, desde sensores de pressão e temperatura até sensores de toque. Como o intuito deste trabalho é quantificar a interação, foram escolhidos sensores de toque capacitivos e sensores de força resistivos, os quais se mostraram mais precisos e adequados ao trabalho.

2.4.1 Sensor de toque capacitivo

Este trabalho utiliza sensores para detectar e quantificar os toques, sendo que o sensor escolhido foi o MPR121, mostrado na Figura 1. Este sensor de toque capacitivo, que é composto por 12 terminais de sensores capacitivos que possuem calibração independente, capaz de medir capacitâncias entre 10pF até 2000pF, podendo identificar variações de até 0,01pF. Essa variação é proveniente da capacitância (capacidade de um objeto reter carga) própria de todo objeto e, quando em contato com uma pessoa, essa capacitância tende a sofrer alteração. Para a transmissão desta alteração, o MPR 121 utiliza comunicação I2C (*Inter-Integrated Circuit*) com saída de interrupção IRQ (*Interrupt ReQuest*), o qual gera um alerta ao Arduino, que possui o microcontrolador ATMega 2560, indicando que houve alteração da capacitância de algum dos terminais. O I2C utiliza-se de duas linhas bidirecionais (leitura e escrita) para comunicação, sendo eles o *Serial Clock* (SCL) e o *Serial Data* (SDA)**.** O SCL é a linha de *clock* do sensor que é utilizada para o sincronismo de informação do dispositivo que habilita o momento de atualizar os dados pelo barramento I2C. O SDA é a linha de dados onde são transmitidas as informações de qualquer dispositivo I2C que esteja ligado ao barramento. Uma linha de terra (GND – *ground*) e outra de alimentação (VCC) devem ser incluídas ao MPR 121 para que tenha uma referência, onde terra é identificado como nível lógica 0, e a fonte (alimentação) como nível lógica 1. Quando há uma alteração na capacitância de algum dos sensores, o IRQ passa para lógica 0 (a verificação dessa alteração é feita através do *loop* do programa implementado), que é identificado pelo Arduino através do pino que o IRQ se encontra conectado. Além de ser possível calibrar a variação, é possível escolher uma rotina de verificação de toque que realiza leituras seguidas dos sensores. Para que seja considerado toque, é necessário que o sensor identifique a variação de capacitância por um número predeterminado de vezes. Essa rotina é

importante para determinar um limiar (*threshold*) a partir do qual será considerado toque e, assim, evitar leituras erradas de variações. Desta forma, o MPR 121 pode diferenciar quando um sensor foi tocado ou não (ŠEKORANJA et al., 2014).

Figura 1 – Sensor de toque Capacitivo MPR121.

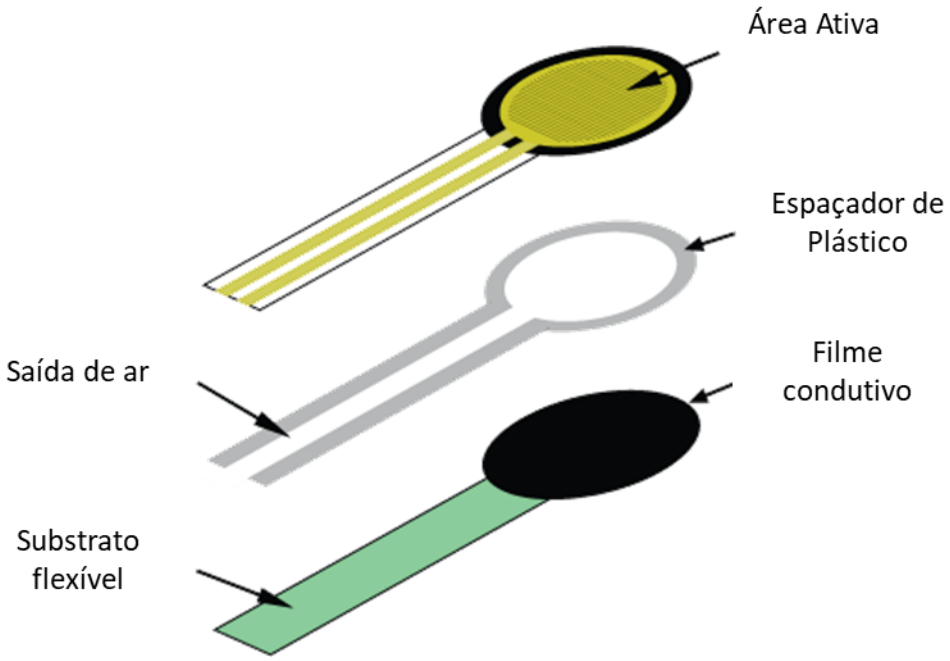


Fonte: adaptado de (ADAFRUIT, 2014).

2.4.2 Sensor de força resistivo

Este trabalho utiliza alguns sensores de força resistivos do tipo FSR (*Force-Sensing Resistor*), o qual possui resistência que varia em função da pressão aplicada sobre o mesmo. O sensor é constituído por quatro camadas, sendo uma camada de plástico eletricamente isolado, uma camada com área ativa, que consiste em um padrão de condutores que são conectados a cabos na parte inferior para que sejam carregados com tensão elétrica, uma camada com espaçador de plástico que inclui uma abertura alinhada à área ativa, bem como uma saída de ar na parte inferior e, por último, uma camada feita de substrato flexível revestido com um filme condutor de polímero espesso alinhado à área ativa, como mostrado na Figura 2.

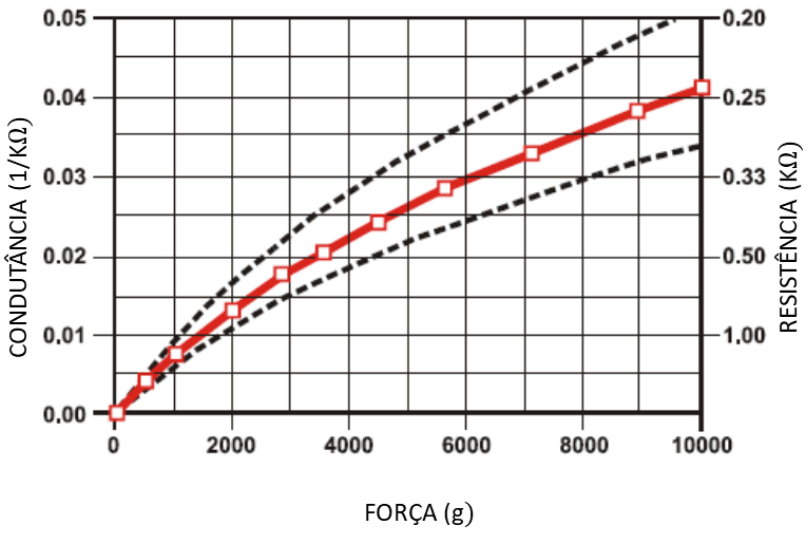
Figura 2 - Diagrama do sensor de força resistivo.



Fonte: adaptado de (ELECTRONICS, 2007).

Quando o sensor sofre uma força externa, o elemento resistivo se deforma contra o substrato, assim, o ar que se encontra entre a área ativa e o filme condutor é expelido através da saída de ar na camada de plástico, fazendo com que o substrato entre em contato com a área ativa permitindo a passagem de energia elétrica pelo sensor. Quanto mais área ativa entrar em contato com o elemento condutivo, menor será a resistência equivalente do sensor, e maior será a corrente que passará pelo sensor. Quando em repouso (sem aplicação de pressão), o sensor possui uma resistência aproximada de 10 MΩ, e quando pressionado, esta reduz, podendo chegar até a 250 Ω. O gráfico da Figura 3 mostra a relação entre força, condutividade e resistência deste sensor (ELECTRONICS, 2007).

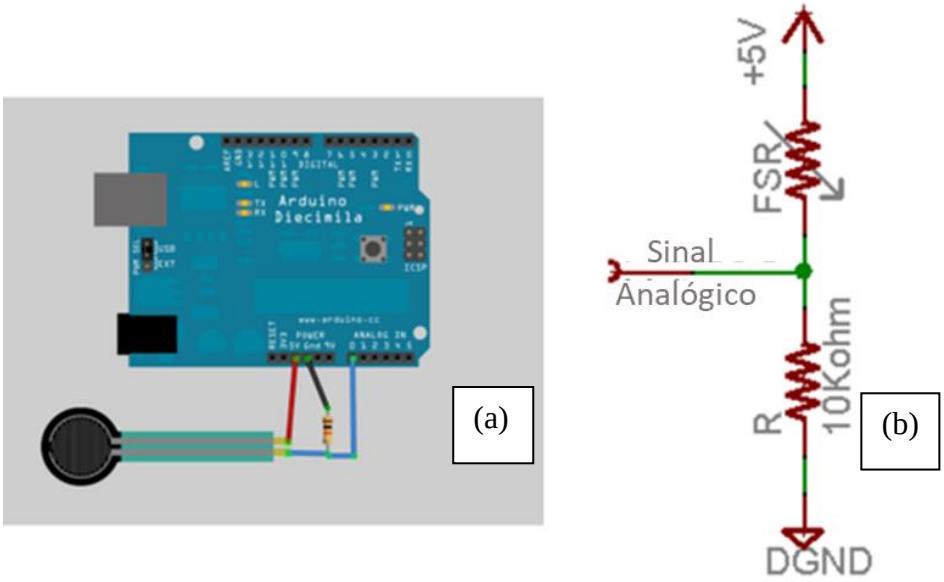
Figura 3 - Gráfico da relação entre força, condutividade e resistência.



Fonte: adaptado de (ELECTRONICS, 2007).

Para a ligação do sensor de força com o Arduino, foi montado um circuito divisor de tensão simples entre o sensor e uma resistência de 100 kΩ, como mostrado na Figura 4. A Figura 4a mostra um exemplo de ligação e a Figura 4b mostra um exemplo de divisor de tensão.

Figura 4 - Esquema de ligação do sensor de força ao Arduino: a) exemplo de ligação b) exemplo de divisor de tensão.



Fonte: adaptado de (ADAWIKI, 2016).

2.5 Sensor Laser RP Lidar

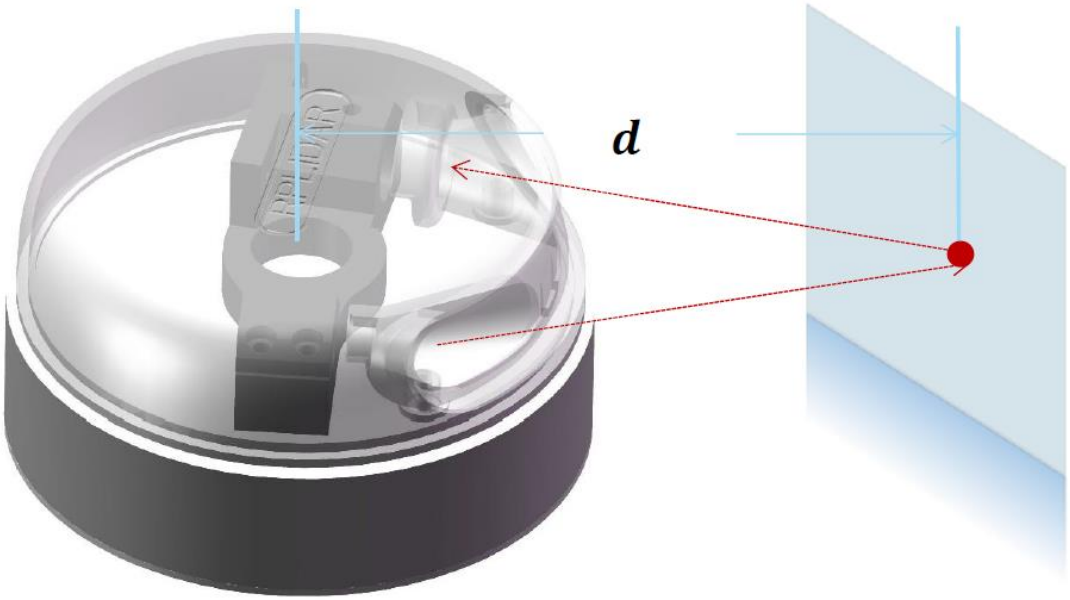
Para a detecção da posição e orientação da criança, foi utilizado um sensor laser RP Lidar A2M8, mostrado na Figura 5, o qual é um sensor de baixo custo para fazer varredura de distâncias, desenvolvido pela empresa SLAMTEC. O sensor possui faixa de medida de 360º e 8 metros de distância, com resolução angular de 0,45º até 1,35º (SLAMTEC, 2016).

Figura 5 - Sensor Laser RP Lidar.

Fonte: adaptado de (SLAMTEC, 2016).

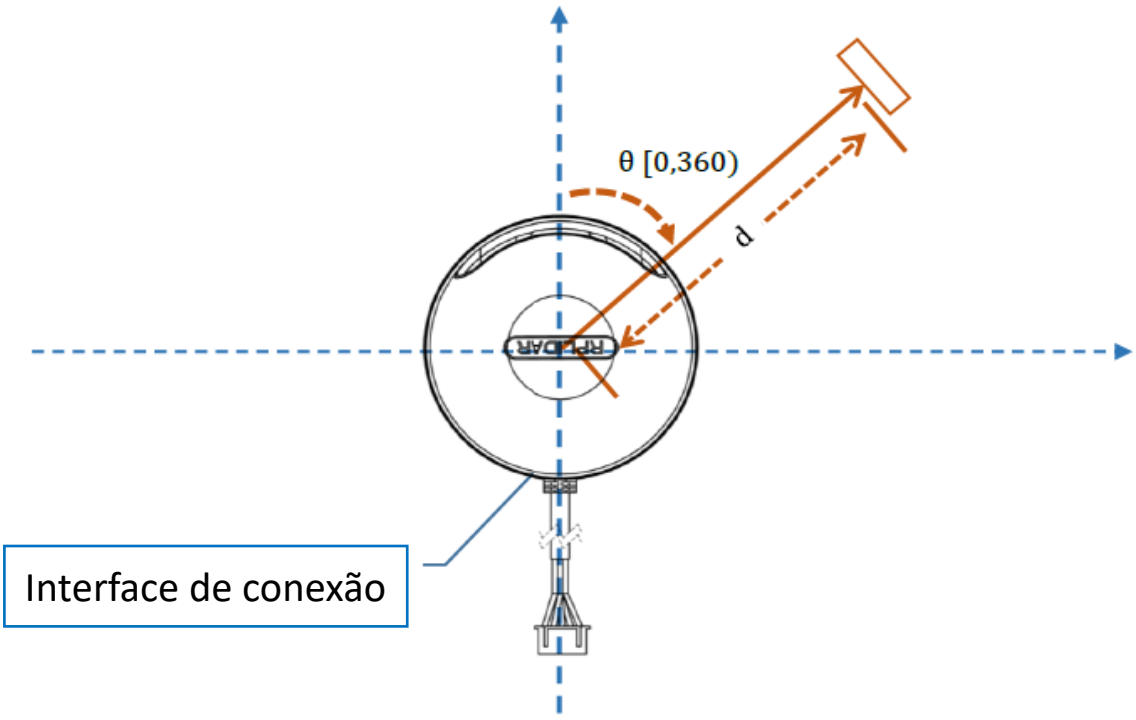
O sensor RP Lidar obtém medidas de distância baseado no método de triangulação por laser, e utilizando aquisição de imagem de alta velocidade. O método consiste na emissão de um sinal laser infravermelho modulado, e sua detecção após refletido por um objeto. Este sinal refletido é captado pelo sistema de aquisição de imagem do RP Lidar, e o microprocessador interno do sensor faz a conversão dos dados recebidos a uma taxa de até 8000 Hz, convertendo-os em valores de distância e ângulo do objeto em relação ao sensor (SLAMTEC, 2016), tal como mostrado na Figura 6. Uma representação da medição é mostrada na Figura 7.

Figura 6 - Método de medição do RP Lidar.



Fonte: (SLAMTEC, 2016).

Figura 7 - Sistema de coordenadas do RP Lidar.



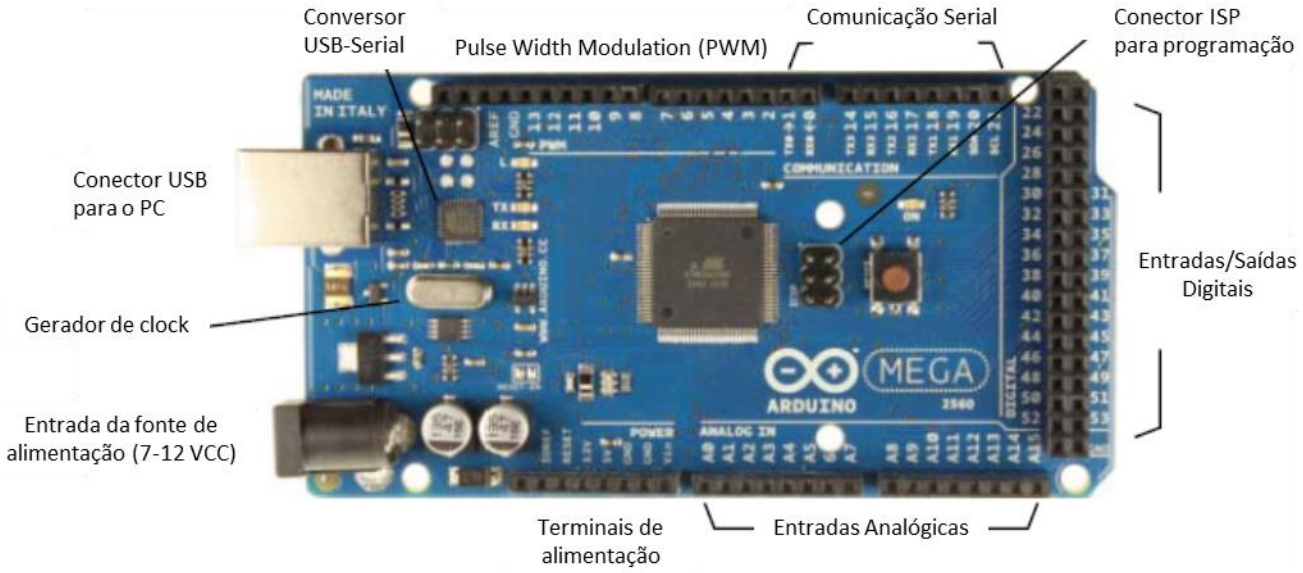
Fonte: adaptado de (SLAMTEC, 2016).

O sensor RP Lidar dispõe de um adaptador USB que permite comunicação com computador e um SDK (*software development kit*), o qual exibe os resultados de varredura, além de salvar os dados de distância e ângulos obtidos (MARKOM; ADORN, 2015).

2.6 Microcontrolador

Neste trabalho, pelo fato de envolver vários sensores, houve a necessidade da utilização de um microcontrolador. Foi escolhido o Arduino Mega 2560, que é uma plataforma de desenvolvimento de *software* e *hardware* compartilhados inteiramente com o usuário (STEVEN, 2012). O Arduino Mega 2560, mostrado na Figura 8, utiliza o microcontrolador ATmega2560 em sua placa, e possui 54 entradas/saídas digitais, onde 14 podem ser usadas como saída PWM (*Pulse Width Modulation*), 16 entradas analógicas, 4 UARTs (*Universal Asynchronous Receiver/Transmitter* – no caso, as portas seriais do *hardware*), um cristal de *clock* de 16 MHz, conexão USB e entrada para fonte de alimentação.

Figura 8 - Diagrama do Arduino Mega 2560.



Fonte: adaptado de (STEVEN, 2012).

A placa do Arduino permite conexão fácil com o computador através de cabo USB-serial que o alimenta, sendo que a mesma também pode ser alimentada por uma bateria ou uma fonte externa AC-CC com saída CC de 7-12V. A placa é equipada com LEDs para TX, RX e um outro indicando quando o algoritmo está rodando (STEVEN, 2012). Os pinos de saída possuem tensão de operação de 5V para o nível lógico alto e 0V para o nível lógico baixo e, nos pinos de entrada, a placa permite tensão na faixa de 6-20V. Além disso, a corrente máxima fornecida pela placa é de 40mA e a mesma possui uma memória SRAM de 8 kB (ROBOTSHOP, 2011). Podem ser adicionados vários *Shields*, caso seja necessária a expansão da placa ou a utilização de funções extras (STEVEN, 2012). O Arduino Mega 2560 é programado no computador através do Arduino IDE, e sua linguagem é a linguagem *Wiring-based*, a qual é bem similar ao C++, porém com algumas modificações e simplificações (ASHIQ et al., 2012).

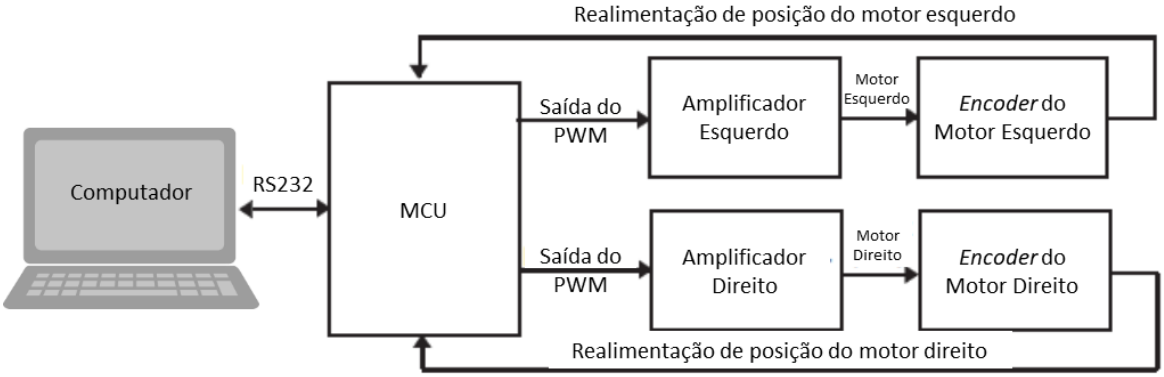
2.7 Robô Pioneer

O robô móvel utilizado neste trabalho é o Pioneer 3DX, o qual é um robô móvel com duas rodas independentes e motorizadas, e uma roda mamona (também conhecida como roda boba, castor ou louca) para estabilidade, como mostrado na Figura 9. Possui um microcontrolador interno que monitora as informações provenientes de dois *encoders*, e gera comandos PWM que são amplificados e enviados, independentemente, para cada roda motorizada, como mostrado na Figura 10. Os dois *loops* de controle independentes de cada roda são controlados por um algoritmo PID (SUSNEA; VASILIU; FILIPESCU, 2008).

Figura 9 - Pioneer 3DX.

Fonte: (ADEPT, 2011).

Figura 10 - Estrutura do Pioneer 3DX.



Fonte: adaptado de (SUSNEA; VASILIU; FILIPESCU, 2008).

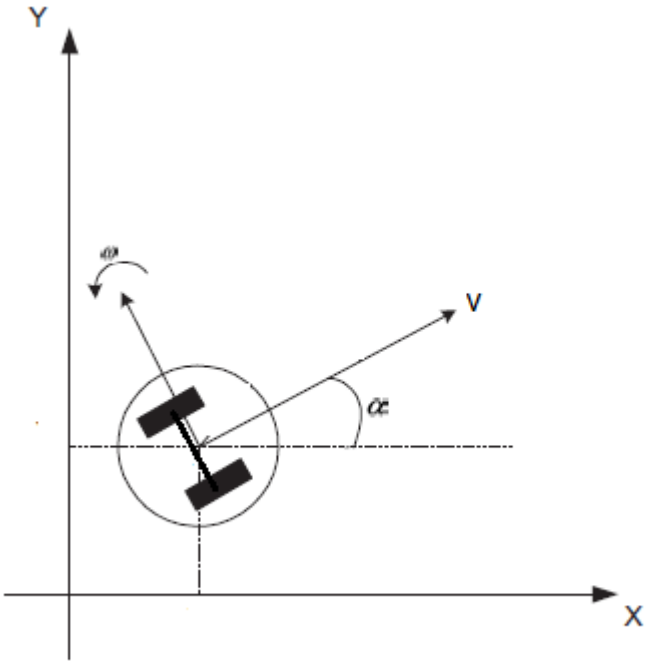
Toda a comunicação com processadores externos é feita através de uma porta serial RS232, de acordo com protocolo próprio (SUSNEA; VASILIU; FILIPESCU, 2008).

2.8 Controle baseado em comportamento

O controle de velocidade do robô para a execução do comportamento “passeio” é feito através do controle das velocidades linear e angular do robô, onde o modelo cinemático (uniciclo com restrições não-holonômicas) é mostrado nas Equações 1 e no diagrama da Figura 11.

$$\begin{aligned} \dot{x} &= v \cos \alpha \\ \dot{y} &= v \sin \alpha \\ \dot{\alpha} &= \omega \end{aligned} \tag{1}$$

Figura 11 - Diagrama de velocidade do Pioneer 3DX.



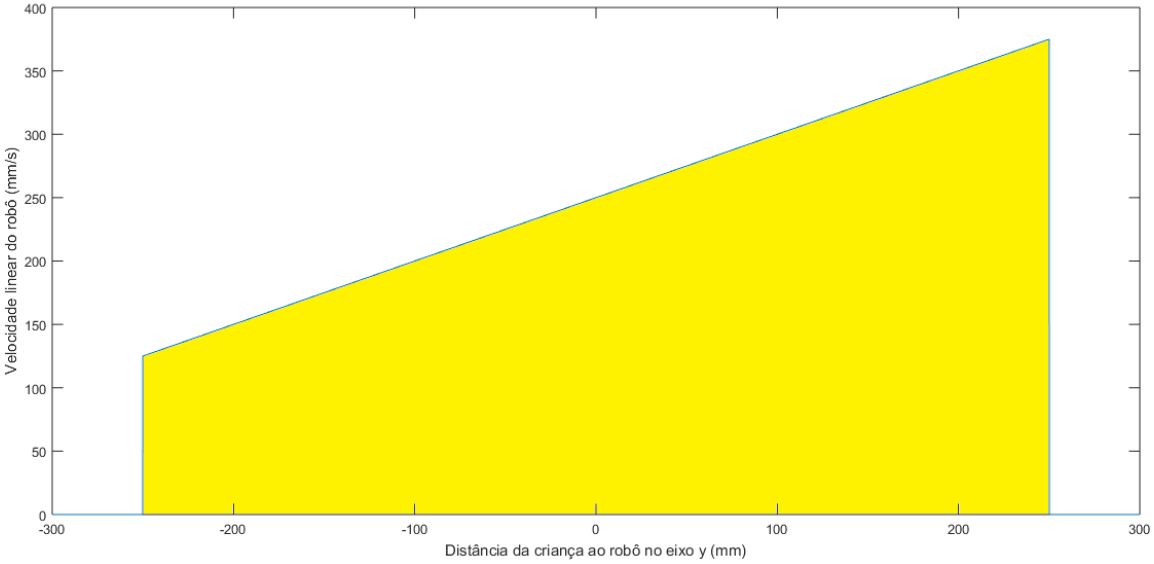
Fonte: adaptado de (FERREIRA et al., 2008).

Na Equação 1, tem-se o espaço de estados com a derivada de x, y e α representando assim a velocidade linear no eixo x e no eixo y, dadas pela velocidade linear ν do robô, e a velocidade angular ω. Quanto a questão do caminho executado pelo robô para o caso do comportamento “passeio”, este consiste em uma circunferência dada por $x^2 + y^2 = r^2$, onde r é o valor da circunferência.

O controle de velocidade foi linearizado, ou seja, para o controle de velocidade foi utilizado um modelo de equação da reta mostrado na equação 2. A equação 2 é representada no Gráfico 1, o qual cada posição da criança corresponde a uma velocidade linear do robô. Fora da região preenchida em amarelo no Gráfico 1 o robô realiza a parada de segurança.

$$v = 0{,}5 \cdot yCriança + 250\ (mm/s) \tag{2}$$

Gráfico 1 - Distância criança-robô no eixo y vs velocidade linear do robô.



2.9 NUC

Para as realizações deste trabalho são necessários dois minicomputadores para fazer a interligação de programas e controladores. Os minicomputadores escolhidos foram do tipo NUC (*Next Unit Computing*), mostrados na Figura 12, os quais possuem dimensões compatíveis, e poder de processamento satisfatório para o trabalho. Os NUCs possuem as especificações mostradas na Tabela 1 (INTEL, 2016).

Tabela 1 - Especificações do NUC.

Nome	NUC – D54250WYKH
Processador	Intel Core I5-4250U (2 núcleos)
Memória	4 GB SO-DIMM (expansível até 16 GB)
Armazenamento	120 GB SSD mSATA
Dimensão	116,6 mm X 112,0mm X 49,5 mm
Alimentação e Potência	Alimentação DC 12 – 19V, 65W
Peso	1,14kg
Conectividade	4 portas USB 3.0 Placa de rede 10/100/1000
Chipset Vídeo	Intel HD Graphics 5000
Saída de Vídeo	Mini HDMI e Port Display

Figura 12 - Minicomputador NUC

Fonte: (INTEL, 2016)

3 DETALHES DA INSTRUMENTAÇÃO DO ROBÔ N-MARIA

O comportamento “passeio” desenvolvido apresenta interações homem-robô e, por isso, necessitou da utilização de alguns sensores. Cada sensor possui sua funcionalidade e importância. Nesta seção, é importante deixar claro que direta ou indiretamente todos os sensores encontram-se conectados a um dos dois minicomputadores NUC.

É importante especificar os dispositivos que definem propriamente o robô N-MARIA. O Pioneer 3DX é o robô que confere a mobilidade do robô N-MARIA e, para isso, possui tanto os atuadores (motores), bem como os sensores internos (*encoders*) que são utilizados para que seja possível a sua movimentação e a realimentação das velocidades linear e angular. O Pioneer 3DX possui comunicação com o NUC através de uma porta USB, entretanto, pelo fato do Pioneer não possuir conexão USB, é necessária a utilização de um conversor USB-RS232 (Figura 13). A linguagem de programação do robô é o C++, porém utiliza a biblioteca ARIA (*Advanced Robot Interface for Applications*), a qual possui funções específicas para o robô, tais como envio de velocidade ou captura de posição do robô, dentre outros dispositivos que podem ser conectados aos robôs da linha Pioneer. Para o comportamento “passeio”, são utilizadas as velocidades angular e linear e os valores da posição e orientação do robô, dados que são extraídos através de processamento das informações geradas pelos *encoders*.

Figura 13 - Conversor USB-RS232.

Fonte: (PIRR@INFORMATICA, 2015).

Importante ressaltar que, apesar de possuir microcontrolador embutido, nenhuma tomada de atitude é realizada diretamente no Pioneer, pois todas as tomadas de decisão são processadas no programa que roda no minicomputador NUC e este, por sua vez, envia as informações ao Pioneer, que aciona seus motores, tendo papel de atuador. A única informação que o Pioneer processa com os seus sensores internos são as informações dos *encoders* de cada roda para obter as informações de posição, orientação e velocidade linear e angular do robô.

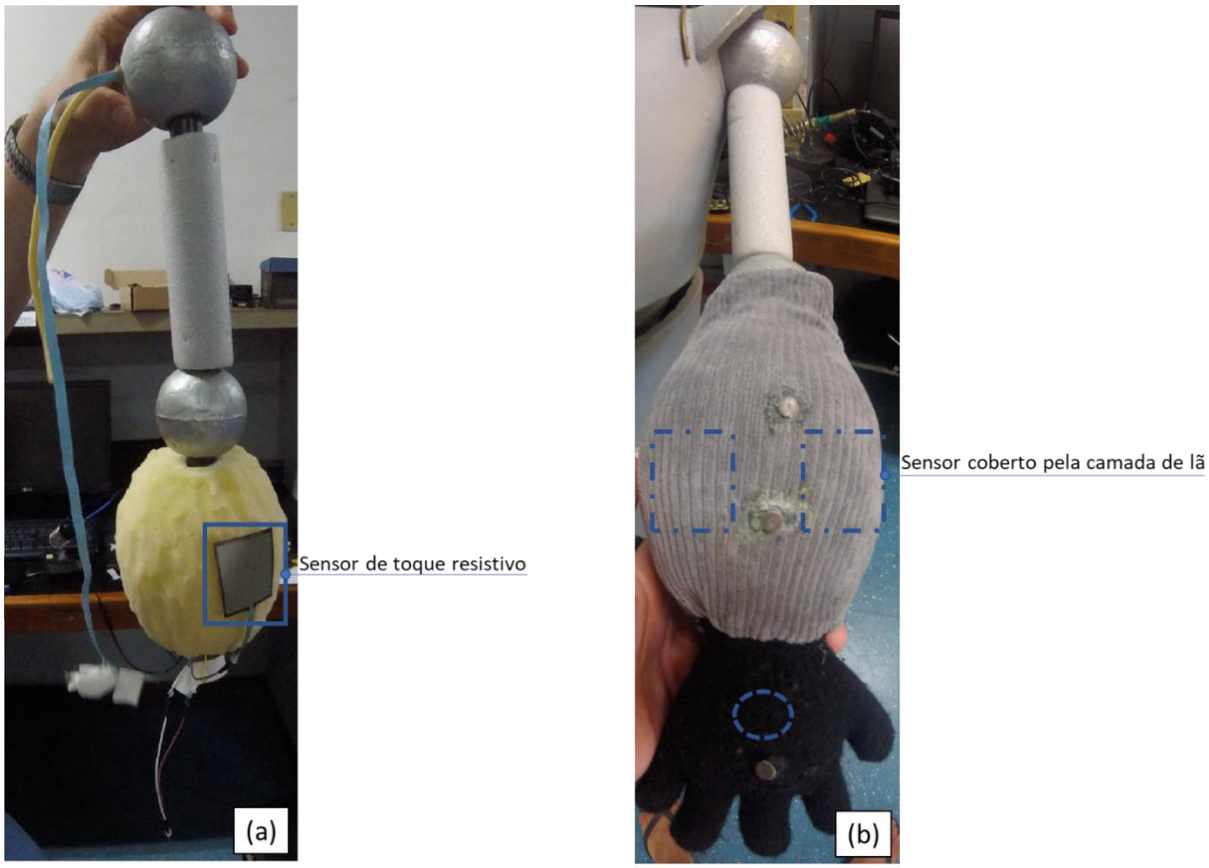
Um dado importante e necessário no comportamento "passeio" é a localização da criança dentro do ambiente e, para a obtenção de tal dado, é utilizado o sensor laser RP Lidar A2M8. O RP Lidar é um sensor que utiliza um laser com varredura de 360º e 8 metros (máximo) para realizar varredura por todo o ambiente. Sua medição é feita através do método de triangulação que já foi explicado no capítulo de embasamento teórico. Através da medição, o RP Lidar é possível conhecer a posição e a orientação da criança em relação ao próprio robô. Para o comportamento desenvolvido, o programa seleciona a faixa na qual deve procurar a criança. Isto foi necessário pois, caso a criança segure a mão esquerda do robô, por exemplo, o sensor laser deve procurar a criança somente no hemisfério esquerdo do robô, para evitar leituras erradas e eventual problema na execução do comportamento, incluindo detecção de objetos não desejados. Além disso, isso permite ao terapeuta ficar do lado em que o robô não esteja considerando as leituras do sensor laser e, desta maneira, se manter relativamente próximo à criança. As zonas da criança e do terapeuta são mostrados na Figura 14. A comunicação do RP Lidar com o robô é feita através de uma porta USB, diretamente com o NUC, que processa a informação dentro do programa, ou seja, o Pioneer não recebe diretamente a posição e orientação da criança, uma vez que tudo é intermediado pelo código que está rodando dentro do NUC.

O robô necessita, para o comportamento de "passeio", identificar quando sua mão está sendo segurada, e, para isso, são utilizados sensores de força resistivos (FSR), os quais permitem diferenciar quando a mão está sendo somente tocada ou se está, de fato, sendo segurada. Os sensores de força resistivos foram instalados nas mãos e antebraços do robô, isto porque o antebraço também é feito de lã e possui área maior, e por isso, por vezes, chama mais atenção e acaba sendo segurado no lugar da mão. Os sensores de força resistivos estão instalados conforme a Figura 15. Na imagem à esquerda (Figura 15a) é mostrado o sensor de força e sua localização dentro do braço, enquanto na imagem à direita (Figura 15b) é mostrado o braço

coberto com a camada de lã, bem como a posição dos outros sensores de força, tanto no braço como na mão.

Figura 14 - Zonas da criança e do terapeuta.

Figura 15 - Sensores de força instalados no braço.



Fonte: Produção do próprio autor.

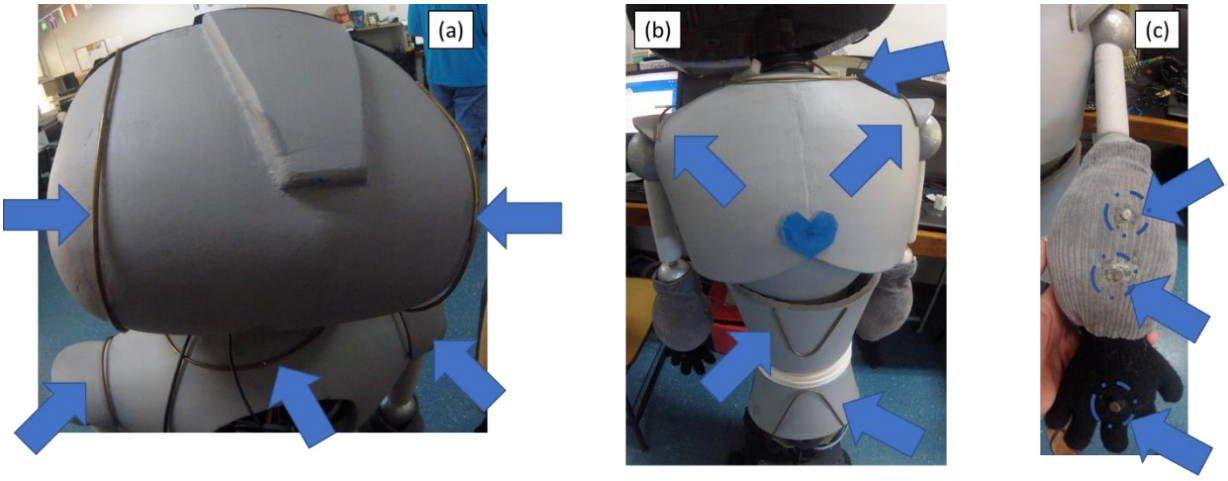
O sensor de força é ligado a um circuito divisor de tensão, como mostrado no capítulo de embasamento teórico, e também ligado ao Arduino Mega. Este, por sua vez, envia os valores relacionados à força, para o NUC através de porta USB. Como o sensor de força resistivo gera um sinal elétrico é necessário quantizar a grandeza em força onde é criada uma relação entre a força e a tensão que chega na entrada analógica do Arduino. Para o trabalho em questão, o circuito do divisor de tensão foi instalado invertido para garantir que o sensor não esteja em contato com energia quando em repouso. Neste caso, quando em repouso, o sensor possui resistência alta e, desta forma, o Arduino recebe tensão de 5V em sua porta analógica (onde o circuito do sensor está conectado) que é convertida em 10 bits num valor entre 0 e 1023. À medida que o sensor sofre uma força, sua resistência decresce e, com isso, a tensão que chega

ao Arduino também reduz. A conversão da grandeza é de 5/1024, ou seja, o Arduino possui uma resolução de aproximadamente 5mV por unidade de grandeza, e consegue perceber qualquer variação maior que 5mV.

Cada sensor possui um conjunto de cabos que fornece certa resistência e foram usados dois modelos do sensor de força: um modelo de placa retangular de área maior e um modelo de placa circular de área menor, sendo que houve a necessidade de calibrar cada sensor separadamente para melhor adequação ao trabalho e para que o circuito não identificasse “toque” como “pressão”.

Além da identificação de força, a identificação de toques realizados se mostra tão importante para o trabalho como um todo (e para interações futuras na N-MARIA) quanto a força. A identificação de locais e materiais que geraram maior interesse e foram tocados com mais frequência também são importantes para estudos e projetos atuais e futuros com o robô. Com isso, houve a necessidade de instalação de sensores de toque por todo o corpo da N-MARIA. O sensor de toque utilizado neste trabalho é o MPR121, sensor de toque capacitivo criado pela empresa Adafruit, que atendeu a todas as necessidades do trabalho, possuindo características como tamanho reduzido, entradas para 12 terminais de sensores de toque, não utilização de energia nos terminais de toque (importante para a segurança da criança), compatibilidade com Arduino Mega e utilização da variação da capacitância dos sensores ligados em seu terminal para detecção de toque. Os sensores de toque foram instalados na cabeça (Figura 16a), tronco (Figura 16b) e braços (Figura 16c) do robô.

Figura 16 - Localização dos sensores de toque no robô.



Fonte: Produção do próprio autor.

O material utilizado no tronco e na cabeça do robô para os toques foram tarugos de latão pois, além de serem condutivos e possuírem todas as características para serem utilizados como terminais do sensor de toque, após instalados, deram uma aparência de adornos no corpo da N-MARIA, o que os transformam em pontos de destaque e gera interesse para com as crianças.

Já o material utilizado nos terminais do sensor de toque localizados nas mãos e antebraços foram eletrodos de ECG de liga de prata/cloreto de prata (Ag/AgCl). O grande problema encontrado com os eletrodos foi que a liga de prata não se fundia com o estanho da solda e, por isso, tiveram de ser soldados aos cabos com auxílio de um pouco de supercola (utilizada com cautela, pois, em excesso, poderia isolar todo o contato do cabo com o eletrodo). A escolha dos eletrodos foi dada pelo fato de serem pequenos e discretos e possuírem coloração acinzentada como as luvas, permitindo que fossem mais discretos, pois não era a intenção que os sensores se sobressaíssem em relação à textura.

Os terminais do sensor de toque foram ligados ao MPR121, que por sua vez, é ligado nos pinos SDA e SCL do Arduino Mega. O código do Arduino realiza leituras de todos os terminais, toda vez que houver um sinal de IRQ do sensor e identifica qual terminal que foi tocado ou não. Essa informação de toque ou liberação do toque do terminal é enviada ao NUC através da saída serial. Esta saída serial, por sua vez, é lida por um código em Python (presente no Apêndice 1) e este código em Python armazena esta informação em um documento de texto dentro do NUC.

Para o caso do comportamento “passeio”, foi necessário que o Arduino, além de enviar quando os terminais das mãos e antebraços foram tocados ou liberados, envie também um sinal para o robô de que a mão está sendo segurada e que ele pode começar o movimento. O código do Arduino faz uma varredura de 80 leituras (para evitar que envie a informação caso haja um erro de leitura) e, após essa varredura, envia o sinal pela saída serial que é, por sua vez, recebido pelo código Python dentro do NUC. O código Python, entretanto, não irá armazenar este sinal em um documento, mas sim reenviá-lo ao servidor, assim, o código do NUC terá acesso a este sinal e enviará ao robô as informações para que comece o movimento de passeio.

4 METODOLOGIA

Neste capítulo será apresentado detalhadamente como o trabalho foi desenvolvido, além de explicar seu funcionamento e a integração de cada componente. O robô utilizado foi o N-MARIA, que é uma versão mais atualizada do robô MARIA, desenvolvido na UFES (Universidade Federal do Espirito Santo) pela equipe do NTA (Núcleo de Tecnologia Assistiva), com o propósito de auxiliar no tratamento de crianças com TEA (Transtorno do Espectro Autista).

4.1 Robô N-MARIA

A estrutura física do robô N-MARIA (acrônimo de *New-Mobile Autonomous Robot for Interaction with Autistics*) é feita de diversos materiais: o tronco e a cabeça são feitos de EVA, as mãos e os braços são de lã, com uma textura mais maleável, contendo espuma industrial em seu interior, para dar um aspecto mais macio e, por último, o antebraço é feito de espuma de polietileno de baixa densidade. A ideia é que cada material chame a atenção da criança de forma diferente e que, no final, ela possa interagir da melhor forma possível com o robô.

A estrutura externa foi customizada de forma a apresentar feição amigável, além de mobilidade e, por isso, sua construção foi feita sobre um robô móvel Pioneer 3DX, o qual lhe confere tal mobilidade. O robô N-MARIA apresenta, também, traços robóticos misturados com traços humanoides, onde seu rosto, formado por um *tablet*, exibe as expressões faciais dinâmicas em sua tela. Junto ao *tablet*, na região da cabeça do robô, estão presentes dois alto-falantes localizados na região análoga aos ouvidos dos humanos, uma câmera RGB, e uma câmera térmica, as quais estão localizadas acima do *tablet*, na região frontal do rosto.

As cores do robô foram pensadas para que o foco da criança fosse no rosto, onde apresentam-se expressões faciais coloridas. Desta forma, o contraste entre as cores do *tablet* e as cores do corpo do robô se devem ao fato das últimas serem cinzas e mais sóbrias (neutras), enquanto as primeiras, relacionadas às expressões faciais no *tablet*, são mais vibrantes (coloridas) e dinâmicas (variam com as expressões e apresentam movimentos característicos a cada expressão). Todas estas características de cores e estruturas da N-MARIA (Figura 17) foram

planejadas para promover conforto durante a interação com as crianças, passando uma imagem amigável e divertida, e com falas artificiais encorajadoras.

Figura 17 - Robô N-MARIA.

Fonte: Produção do próprio autor.

Como mencionado anteriormente, o robô foi personalizado de forma a apresentar uma feição amigável, com traços robóticos para realizar interações com as crianças, estimulando uma gama de habilidades sociais, tais como a imitação, o toque e a atenção compartilhada. Inicialmente, o robô iria realizar funções simples como apresentar-se, aproximar-se da criança e girar, além de expressar algumas emoções através de um *tablet* em seu rosto, porém, como a terapia com crianças com TEA tem por objetivo incentivar a interação social da criança, apresentou-se a necessidade de adicionar comportamentos que possibilitassem uma interação mais aprofundada e dinâmica, somado ao fato de algumas crianças apresentarem dificuldades de interagir e

seguirem um adulto em um passeio rotineiro, mostrou-se necessário desenvolver uma proposta com um enfoque centrado nesta área.

Pensando nisto, foi desenvolvido um comportamento para o robô para simular e estimular esta habilidade de interação na criança, com o objetivo de ajudá-la a sentir-se mais confortável ao encontrar-se em situações similares. Para que o trabalho alcançasse seu êxito, foi necessário determinar que todo o comportamento do robô fosse executado dentro de um ambiente controlado, com menor quantidade de perturbações por estímulo externos, e não apresentasse em momento algum movimentos bruscos ou repentinos, pois tais ações poderiam acarretar numa total desqualificação do processo, podendo causar traumas à criança; o robô deve, assim, executar somente movimentos socialmente aceitáveis (GOULART, 2014).

Importante ressaltar que, apesar de todas as medidas de segurança tomadas na construção do sistema do robô, que por vezes são redundantes, todo o comportamento foi planejado sobre uma condição básica e necessária para que fosse utilizado em ambiente controlado, isto é, em ambiente fechado, e com presença integral de um terapeuta capaz de manusear o robô em caso de alguma emergência ou até mesmo eventual necessidade de troca ou escolha de outro comportamento para comandar as ações do robô. Isso é necessário, pois, apesar de possuir certa independência, o robô necessita de um comando prévio por parte do terapeuta para que seja feita a escolha do comportamento que o robô deve executar e, assim, realizar as tarefas de forma conveniente.

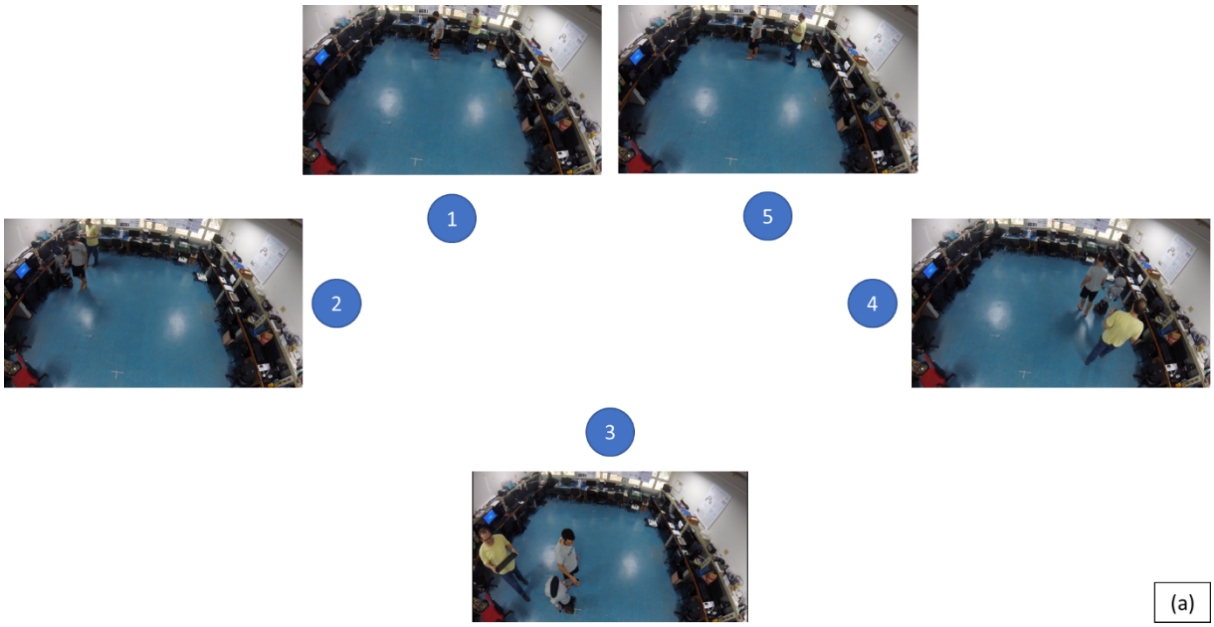
O robô N-MARIA, como a maioria dos robôs desenvolvidos para interação social, apresenta uma gama de sensores e processos que serão abordados mais a fundo nas seções subsequentes deste capítulo. Este sensoriamento permite que a N-MARIA possa interagir com o ambiente e com a criança sob terapia. Tal sensoriamento envolve as câmeras RGB e térmica, sensor laser RP Lidar, além dos sensores de toque, que se encontram espalhados pela cabeça, tronco e braços, e os sensores de força, que se encontram nas mãos e braços do robô. Alguns dos sensores supracitados foram programados em linguagens distintas entre si, porém, por ser um projeto integrado, todos eles necessitam comunicar-se com o computador central para que ações sejam tomadas pelo robô. Desta maneira, foi desenvolvido, pela equipe de projeto do NTA, um servidor *web* hospedado no computador central, o qual é capaz de reunir as informações vindas dos diferentes sensores.

4.2 Comportamento desenvolvido

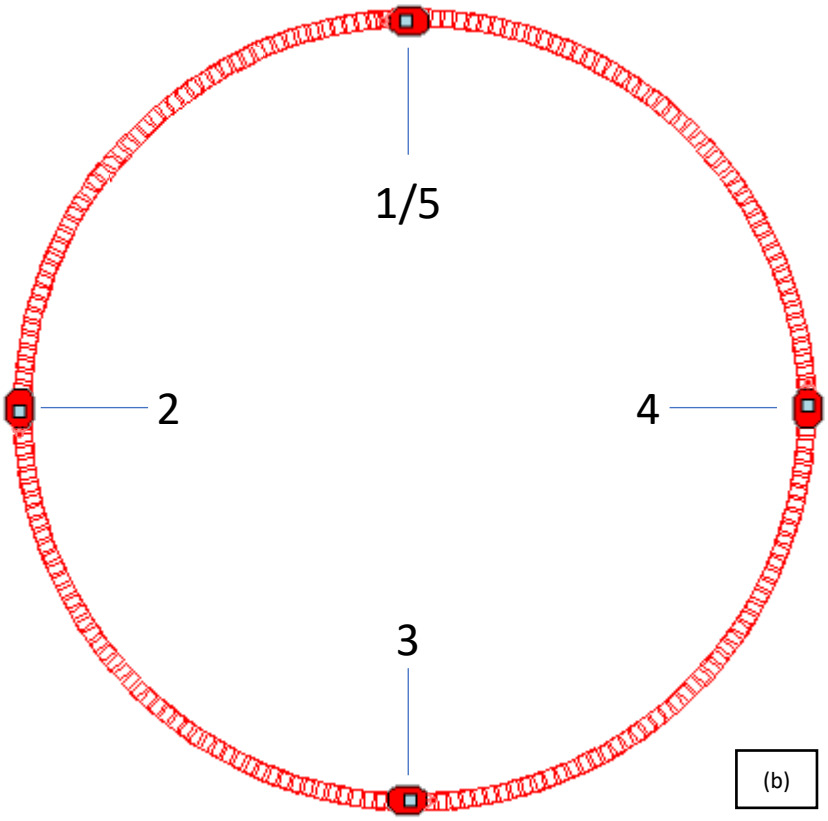
Como citado na seção anterior, este trabalho tem o objetivo de desenvolver um comportamento para o robô que simule um “passeio” e que futuramente possa auxiliar em terapias de crianças com TEA para que desenvolvam habilidades sociais. A ideia de um comportamento de passeio surgiu da observação de que, com a interação social prejudicada, crianças com TEA possuíam problemas para realizar tarefas simples como um passeio com adultos.

A ideia base do comportamento desenvolvido é que a criança que estiver interagindo com o robô se sinta confortável e segura para segurar a mão do robô e realizar um passeio pela sala, onde o passeio se caracterizará por seguir o caminho de uma circunferência com um raio predeterminado (Figura 18). No passeio, a criança será colocada sempre ao lado mais próximo do centro do caminho enquanto o robô se encontrará localizado do lado externo. Foi utilizado um método para que o robô dite o ritmo e controle ao máximo a situação, ajustando a própria velocidade de forma a manter uma faixa de distância entre ele e a criança (medida por meio do sensor laser), porém, jamais perdendo a rota circular predeterminada. A Figura 18a mostra as fotos feitas durante a execução do comportamento “passeio”, e a Figura 18b uma simulação de como é feito o “passeio” no *software* MobileSim da Adept.

Figura 18 - Movimentos feitos pelo robô e pelo humano durante o passeio: a) foto representando o passeio b) Resultado da simulação do comportamento “passeio”.



Fonte: Produção do próprio autor.



Fonte: Produção do próprio autor.

Foi escolhido que o robô se concentrasse exclusivamente na criança, e este é um dos motivos da necessidade de todo processo ocorrer em local controlado, caso contrário, poderiam aparecer obstáculos imprevistos e indesejáveis. O outro motivo da necessidade de um local controlado é o fato de crianças com TEA serem propícias a perderem atenção mediante qualquer estímulo, o que dificultaria manter a atenção da criança em todo o processo.

O comportamento de “passeio” foi programado para que o robô espere que a criança se aproxime, segure sua mão, percorra o percurso e pare ao final, quando todo o caminho em circunferência tiver sido completado ou quando a criança soltar a mão do robô. Entretanto, para que o robô inicie o passeio e o mesmo esteja dentro do contexto da terapia, deve haver uma prévia autorização do terapeuta para que o robô entenda que ele pode começar o comportamento

“passeio”, ou seja, o terapeuta deve enviar um sinal através de um *tablet* de controle, o qual é disponibilizado para que o robô receba comandos de que deve esperar e agir durante a interação (no caso, começar a mover-se ao detectar que a criança segurou a sua mão). Esta autorização prévia pelo terapeuta foi um critério adotado para que houvesse maior segurança em todo o processo, pois, sem tal comando, o robô poderia interpretar em momentos errados o comando para execução do passeio, o que não seria conveniente para o trabalho nem para a criança nem para o terapeuta.

Existem situações em que o robô entra automaticamente em modo de segurança, e para o movimento de imediato, por identificar que são situações que podem colocar a criança em risco. Serão citados a seguir os casos em que a parada imediata de movimentos do robô deverá ocorrer:

- **Caso a criança entre na frente do robô em movimento**: caso o comportamento de “passeio” já tenho iniciado e em algum momento, antes do final do “passeio”, a criança, por algum motivo, cruzar a rota do robô, ele irá parar, entendendo que nestas situações, se não for realizada a parada completa do robô, o mesmo irá colidir com a criança.
- **Caso a criança se aproxime demais da base do robô**: se o comportamento de “passeio” já houver iniciado e, em algum momento, antes do final do “passeio”, a criança se aproxime demais da base do robô, ele irá parar, entendendo que nesta situação pode acontecer de uma de suas rodas colidir com a criança e causar danos a ela.
- **Caso a criança se afaste demais da base do robô**: caso o comportamento de “passeio” já tenha iniciado, e em algum momento, antes do final do “passeio”, a criança se afaste demais da base do robô, ele irá parar, entendendo que neste caso possa causar danos à sua estrutura, tal como um braço ou alguns fios dos sensores se soltarem, prejudicando assim o funcionamento do sistema.
- **Caso a criança vá muito mais rápido que o robô ou fique muito pra trás**: nas situações em que o comportamento de “passeio” já tenha sido iniciado e, por alguma razão, antes do final do “passeio”, a criança ande rápido demais para que o robô a acompanhe (acima da velocidade máxima de segurança estipulada pelo robô), fique parada ou ande muito devagar e, assim, se distanciar demais do robô, ele irá parar, entendendo que a atenção da criança não estará por completo no movimento, além de

haver o risco de ocorrer prejuízos ao robô, já que algum braço pode se soltar ou sensores serem desconectados.

- **Caso a criança solte a mão do robô**: caso o comportamento de "passeio" já tenha iniciado e em algum momento, antes do final do "passeio", a criança solte por completo a mão do robô, ele irá parar, entendendo que a criança não está mais com sua atenção voltada para o movimento, o que tiraria o propósito inicial do movimento.
- **Caso o terapeuta envie um comando diferente para o robô**: caso o comportamento de "passeio" já tenha iniciado e, em algum momento, antes do final do "passeio", o terapeuta envie um comando para o robô diferente do comportamento de "passeio", o robô deve entender e acatar a mudança de comando imediatamente, por identificar que, caso não haja mudança do comando, poderá acarretar em danos à criança.

Todo o comportamento de "passeio" foi planejado de modo a manter o maior controle possível para o robô e, ao mesmo tempo, deixar que o movimento seja fácil para a criança. Quanto à segurança, há um limite máximo de velocidade do robô e a velocidade instantânea é ajustada dinamicamente, onde o robô está sempre verificando a posição da criança em relação ao eixo Y, e recalculando-a para verificar a necessidade de acelerar ou reduzir a velocidade, para tentar manter a criança o mais paralelo ao robô possível, dentro da distância permitida entre eles. O ajuste de velocidade é calculado pelas Equações 2, e atualizado a cada ciclo de *clock* do robô.

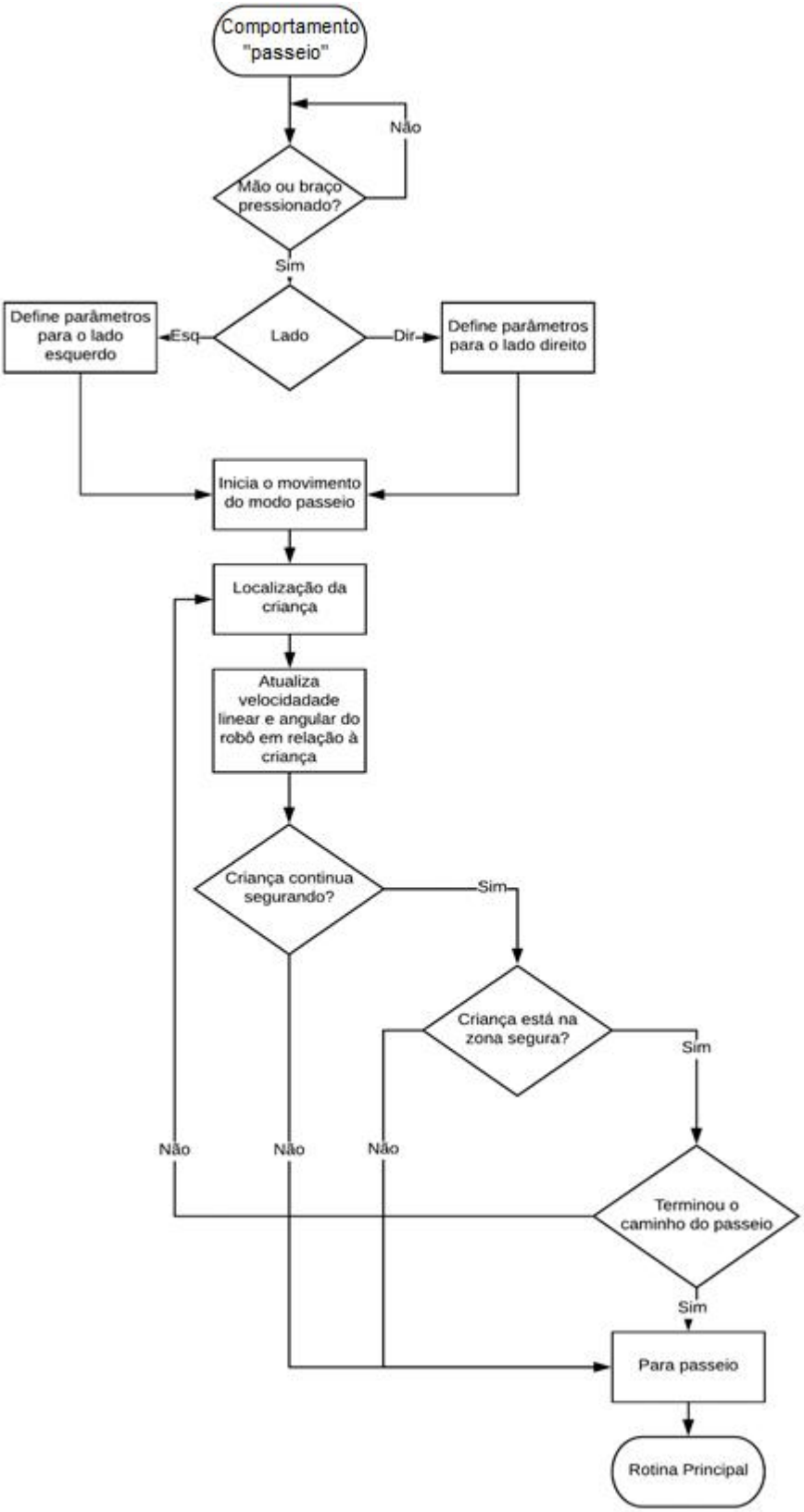
$$vel = 0{,}5 \cdot yCriança + 250$$
$$\omega = \pm 180 \cdot \frac{vel}{\pi * r} \quad , \qquad (2)$$

onde "vel" é a velocidade angular do robô, $yCriança$ é a posição da criança no eixo Y em relação à base do robô, ω é a velocidade angular, e "r" é o raio da circunferência do movimento. Na Equação 2, a velocidade "vel" é dada pela equação da reta (linearizada, por representar o movimento satisfatoriamente), dada em mm/s que é a unidade de velocidade que o robô utiliza. A velocidade angular ω vem da equação $vel = \omega r$, e na Equação 2, ela é representada dividindo pelo raio "r" e multiplicando por ±180, para converter a unidade de rad/s para deg/s, que é a utilizada pelas bibliotecas do robô. O sinal utilizado na Equação 2 depende se o movimento

será para o sentido horário ou anti-horário, que, por sua vez, depende de qual mão do robô estiver sendo segurada pela criança.

O robô N-MARIA possui vários comportamentos em funcionamento, por isso cada comportamento deve ser escolhido pelo terapeuta através do *tablet* disponível. Quando o terapeuta seleciona o comportamento “passeio” no *tablet*, é enviado ao servidor um comando de identificação do comportamento, o qual é recebido pelo robô e identificado como uma permissão para realizar o movimento. A partir do momento em que o comando é lido pelo robô, o mesmo entra no estado de verificação para saber se a criança segurou a mão. No momento em que a criança segura a mão do robô é que então o comportamento de “passeio” realmente começa. Com esta logística, assegura-se que o robô não vá movimentar-se sem que a criança esteja segurando de fato a sua mão. Os passos executados durante o comportamento “passeio” são apresentados no fluxograma da Figura 19.

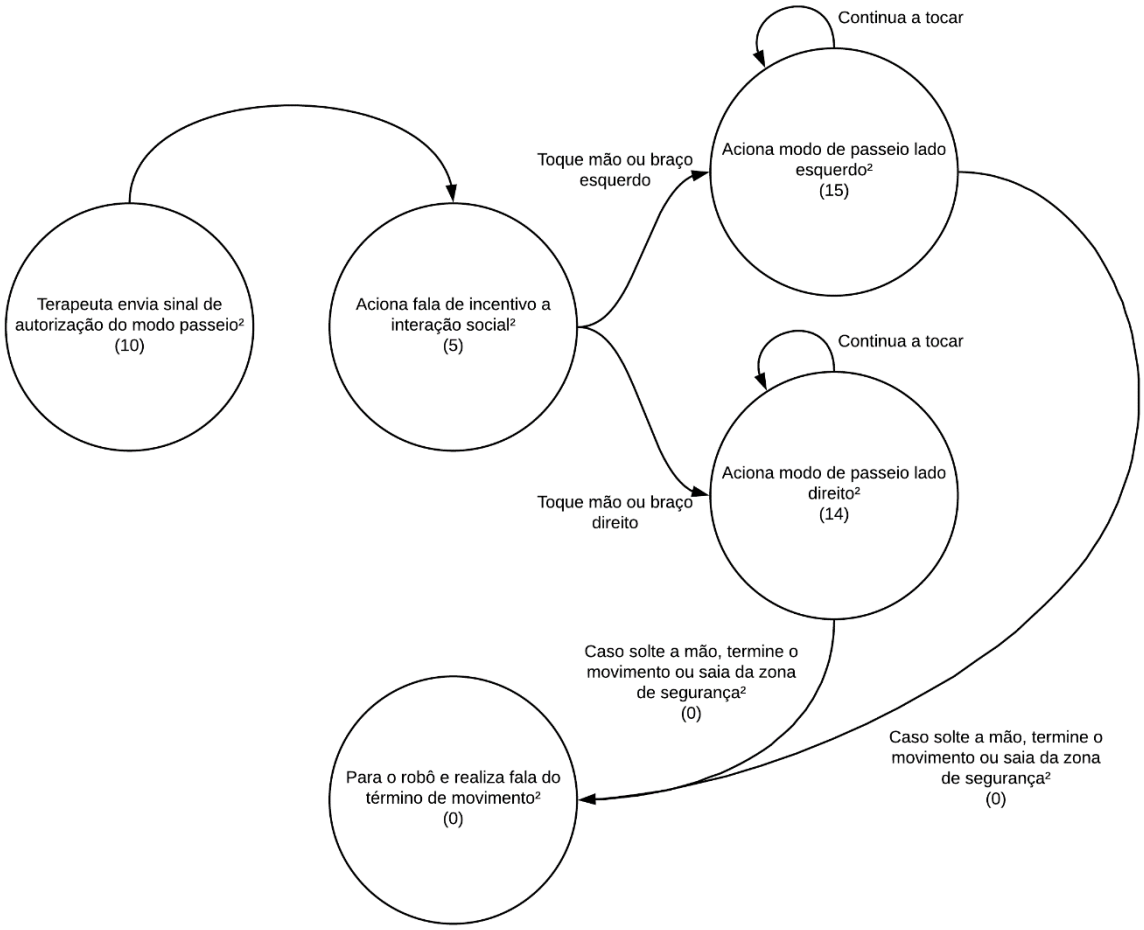
Figura 19 - Fluxograma do comportamento "passeio".



Fonte: Produção do próprio autor.

O comportamento “passeio” é sincronizado com falas artificiais do robô para, primeiramente, incentivar a criança a segurar a sua mão e realizar o movimento e, após o início do movimento, a qualquer momento em que o robô parar, uma fala é emitida para parabenizar a criança por terminar o movimento (mesmo que a criança não consiga terminar o movimento por completo). Isso é feito para estimular a criança a interagir com o robô. O diagrama de estados do comportamento “passeio” é mostrado na Figura 20.

Figura 20 - Diagrama de estados do comportamento de “passeio”.



Fonte: Produção do próprio autor.

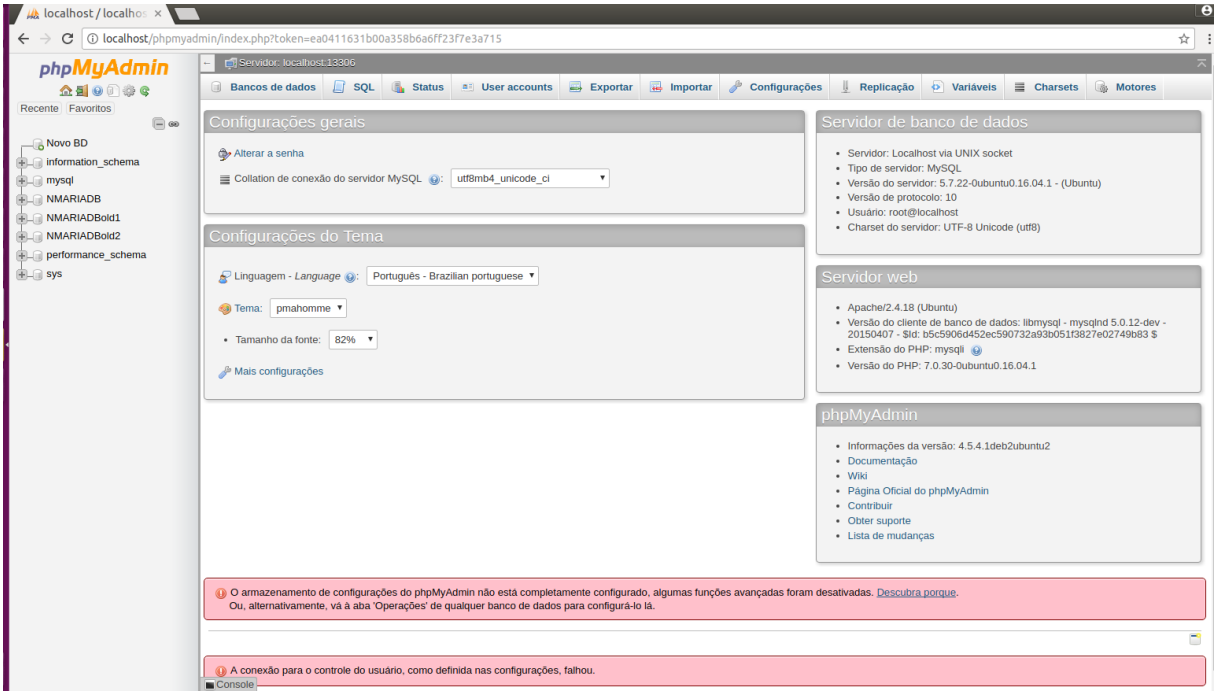
4.3 Servidor Web e comunicação entre NUCs

Neste trabalho são utilizados dois computadores NUC que, para comunicarem entre si, utilizam um servidor *web* escrito em linguagem PHP (sigla atual e recursiva para "PHP: Hypertext Preprocessor", originalmente *Personal Home Page*). O NUC, localizado na parte superior do tronco da N-MARIA, encontra-se conectado diretamente com o Arduino Mega por uma porta USB. Este Arduino, que foi programado em C, recebe informações dos sensores de força e toque e os envia, via porta serial, para o NUC que roda um *script* em Python para ler a informação recebida pela porta serial. Ele utiliza-se deste mesmo código Python para enviar e receber informações do servidor *web* através da biblioteca cURL para Python.

O NUC localizado na base da N-MARIA utiliza um código escrito em linguagem C++ para comunicar-se com o RP Lidar, que possui biblioteca própria para o C++ e, por isso, já envia as informações ao NUC pela entrada USB. Já o Pioneer 3DX utiliza a biblioteca ARIA, também escrita para a linguagem C++ e desenvolvida pela Adept MobileRobots, para se comunicar com o NUC. Através da biblioteca cURL para C++, este código comunica-se com o servidor *web*, permitindo enviar ou obter dados do servidor e, consequentemente, capturar dados de sensores diferentes.

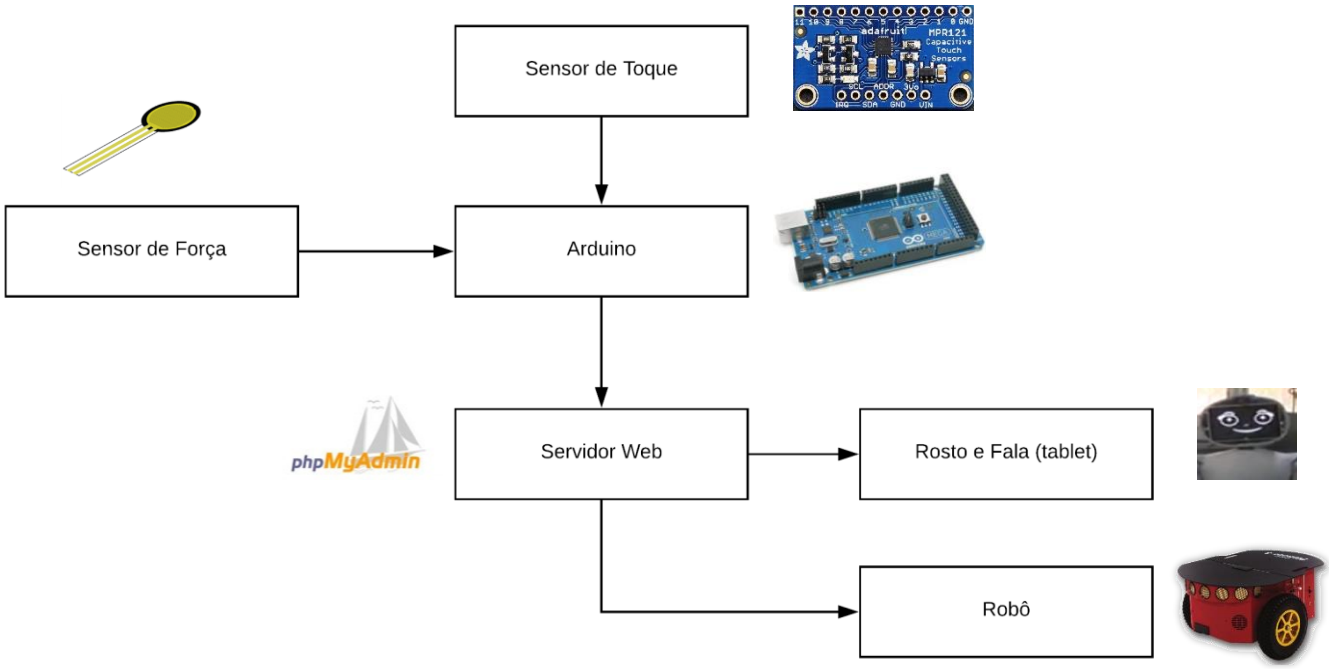
O servidor *web* ,cuja a página inicial está apresentada na Figura 21,utiliza-se de linguagem PHP e é estruturado de forma que existem áreas separadas para dados obtidos, sinais de fala recebidos e enviados, sinais de ações recebidos ou enviados, e informações do processo, tais como posição e orientação da criança e do robô. Cada área é montada de forma a assemelhar-se a uma tabela, onde os dados são alocados em suas posições correspondentes na tabela, sendo que tal organização permite uma padronização e uma facilidade maior quando algum dos NUCs precisar acessar alguma informação do servidor *web*. Para maior compreensão da comunicação entre as partes da N-MARIA, o fluxo de informações entre essas partes está detalhado no diagrama da Figura 22.

Figura 21 - Página inicial do servidor.



Fonte: Produção do próprio autor.

Figura 22 -Diagrama de como é feito a comunicação entre as partes do robô



5 EXPERIMENTOS E RESULTADOS

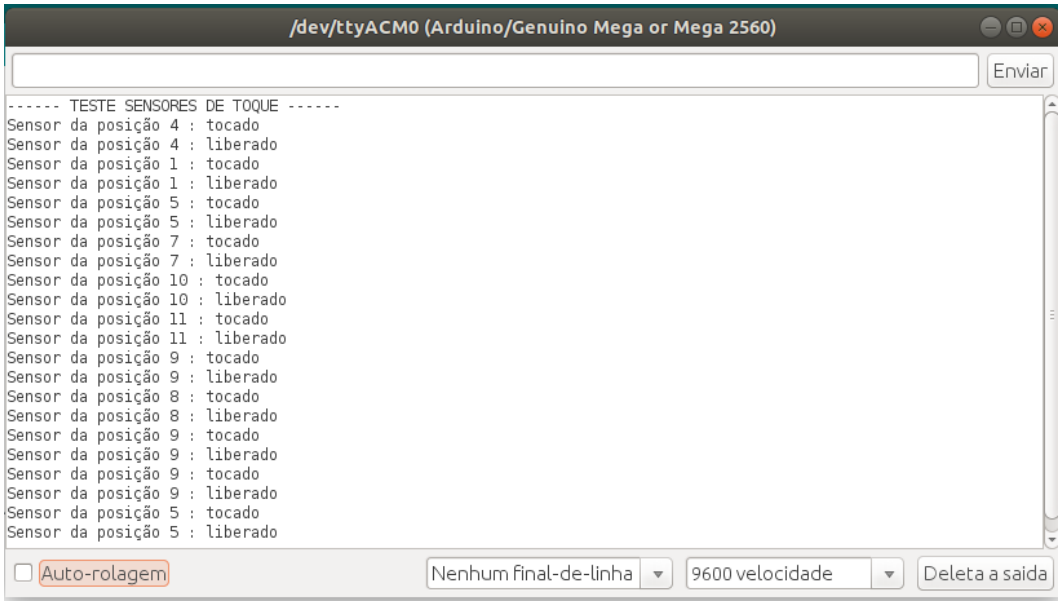
Neste capítulo será discorrido sobre as experiências realizadas para validação do trabalho. Foram realizados testes com os sensores de toque e força, no movimento e nas condições de segurança. Cada experimento foi realizado de forma diferente para acentuar as características desejadas de cada elemento.

5.1 Sensor de toque capacitivo

Para o experimento com o sensor de toque, foi necessário acessar a saída da porta serial do Arduino Mega ligado no NUC. A porta serial deveria enviar qual terminal estava sendo tocado ou liberado em ordem; caso fosse tocado mais de um terminal ao mesmo tempo, ele deveria identificar cada um individualmente.

Na Figura 23, que mostra o teste, é possível observar que os terminais do sensor de toque responderam bem aos toques, mostrando quando eram tocados ou liberados. Os terminais são representados por números que identificam a posição em que eles estão ligados no MPR121, e o número à frente representa se o sensor foi tocado ou liberado. Caso o valor seja 0 significa que o sensor foi tocado; já se o valor for 1 significa que o sensor foi liberado.

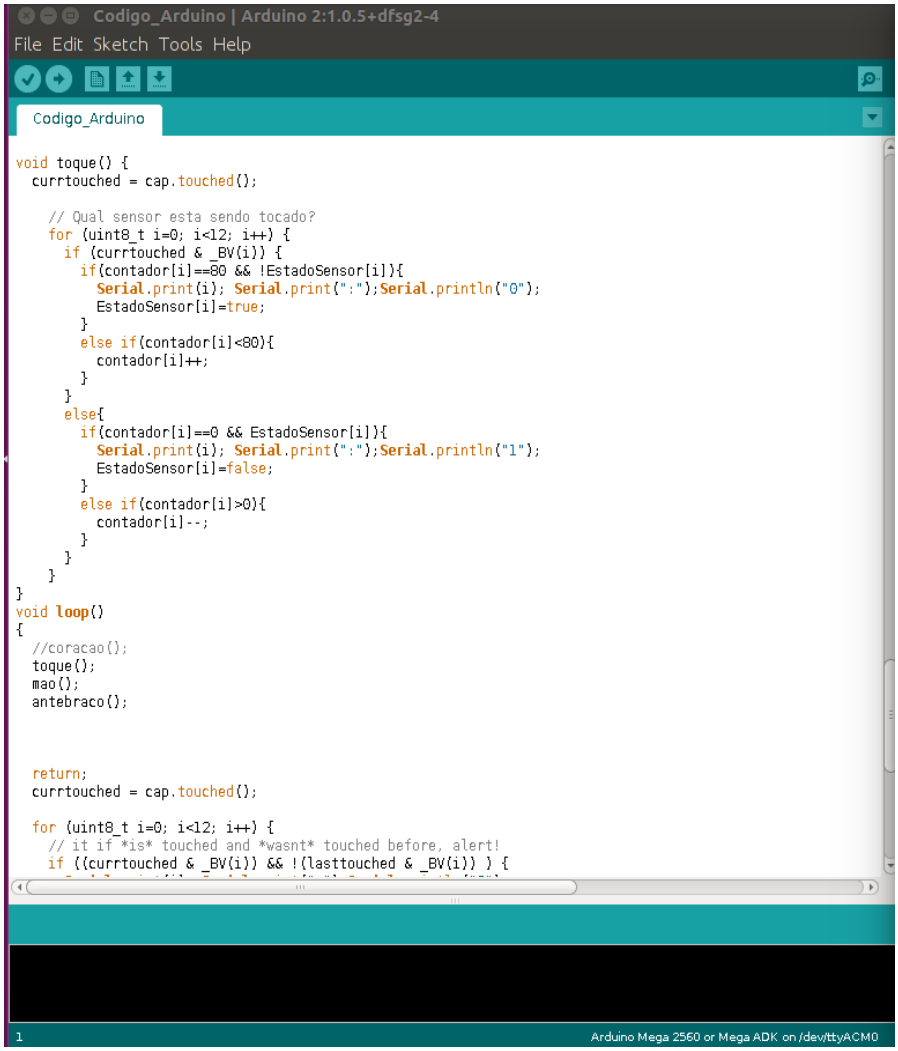
Figura 23 - Experimentos com os 12 sensor de toque.



Fonte: Produção do próprio autor.

Na Figura 24 é mostrado o código para leitura do sensor. A função “toque” é constituída de um *for* em que todos os terminais do sensor de toque são lidos. Quando um terminal é tocado, o programa verifica se já foram feitas 80 leituras seguidas desse terminal e então seu marcador é definido como tocado e, só então, o Arduino envia para a porta serial que o terminal em questão foi tocado. Caso o marcador já se encontre no modo tocado, então o código mantém o estado e não escreve nada na tela. Se o contador do terminal de toque não estiver em 80, seu contador será incrementado em uma unidade a cada leitura. Caso não tenha sido detectado toque em nenhum terminal, o código irá decrementar os contadores de cada terminal que possuam valores em seus contadores diferente de zero. Quando um contador é zerado, o marcador desse terminal é alterado para “liberado” (o sensor deixou de ser tocado) e, então, o Arduino publica na porta serial que o terminal em questão foi liberado.

Figura 24 - Código de leitura do sensor de toque.



5.2 Sensor de Força Resistivo (FSR)

Para o experimento com os sensores de força resistivos houve a necessidade de acessar o Arduino através do NUC, pois os circuitos dos sensores de força são ligados em portas analógicas do Arduino como citado anteriormente neste trabalho. A saída da porta serial do Arduino fornece o valor de força de cada um destes sensores. O código faz a verificação dos sensores de força dos antebraços separados (esquerdo e direito) e, dentro de cada antebraço, separado por hemisfério. Isso foi preciso, pois, devido à circunferência do antebraço, foi necessária a instalação de dois sensores (ou conjunto deles) de força por antebraço. Desta maneira, pode-se também aproveitar a montagem para identificar qual parte do antebraço foi pressionada. As mãos são identificadas por somente um sensor de força presente em cada uma das mãos. Na Figura 25 é apresentado o experimento feito com os sensores de força, onde são exibidos os valores obtidos e na Figura 26, o fluxograma do código do sensores de força.

Figura 25 - Valores encontrados nos experimentos com os sensores de força dos antebraços do robô.

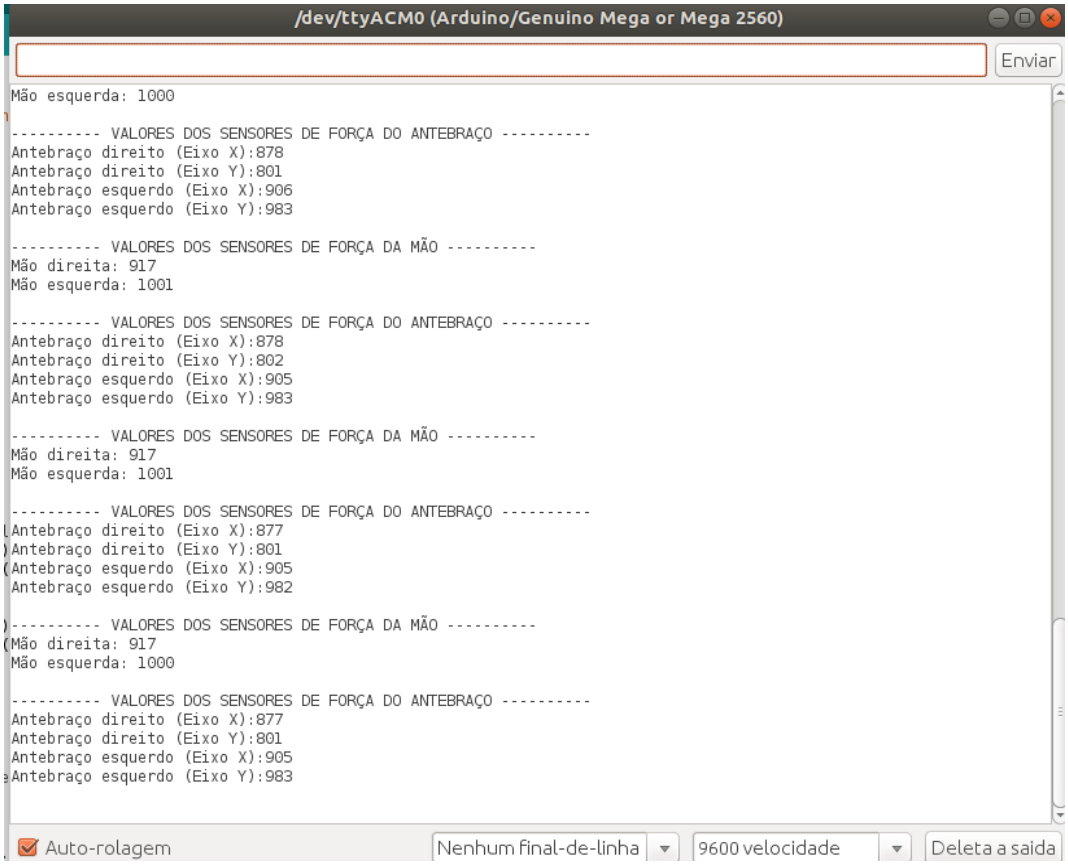
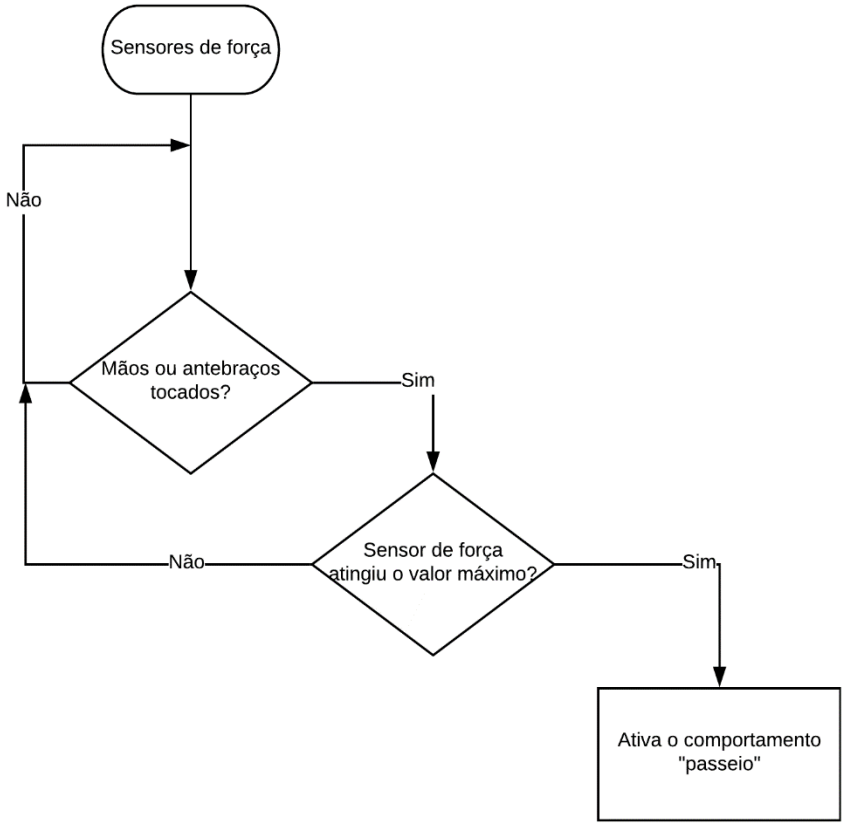


Figura 26 - Fluxograma do código do sensor de força.

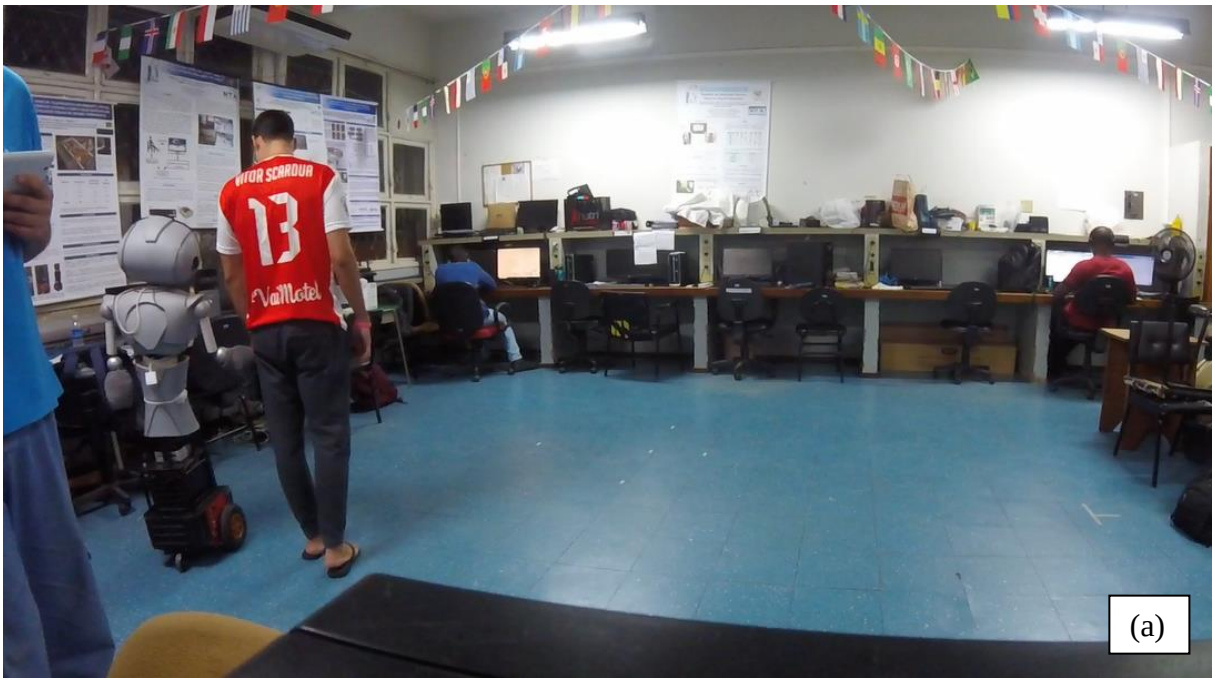


O experimento foi utilizado para calibrar os sensores de tal forma que fosse possível observar quais valores seriam os mais coerentes para cada sensor de força, de tal maneira que fosse possível diferenciar quando era um leve toque (que não deveria representar ativação do sensor) ou uma pressão (que ativaria o sensor). Um exemplo de calibração é mostrado no código da Figura 25, onde o condicional para acionar a rotina do antebraço direito, independentemente do hemisfério x ou y, deve atingir valores abaixo de 660, enquanto para acionar a rotina do antebraço esquerdo o hemisfério x deve atingir valores abaixo de 700, e para o hemisfério y deve atingir valores abaixo de 660. Essa diferença, inclusive entre sensores de um mesmo antebraço, ocorre, pois, as luvas são apertadas e causam pressões que não são distribuídas de forma igualitária em cada sensor.

5.3 Experimento com o "passeio"

O comportamento "passeio" é o experimento mais completo e o principal do trabalho, pois permitiu a validação do sistema como um todo. Para este experimento, foi utilizado o ambiente do laboratório do NTA no prédio do CT2 na UFES, já que este ambiente é isolado de interferências externas e totalmente controlado. Todos os obstáculos foram retirados para que o caminho estivesse livre. O sistema foi testado por completo, ou seja, o comportamento "passeio" junto aos sensores de toque e força, simulando um comportamento real. Na Figura 27 são mostradas sequências de imagens do experimento. A Figura 27a mostra o início do percurso, e a figura 27b mostra o meio do percurso, e a Figura 27c mostra o final do percurso.

Figura 27 - Experimento do comportamento “passeio”.



Fonte: Produção do próprio autor.

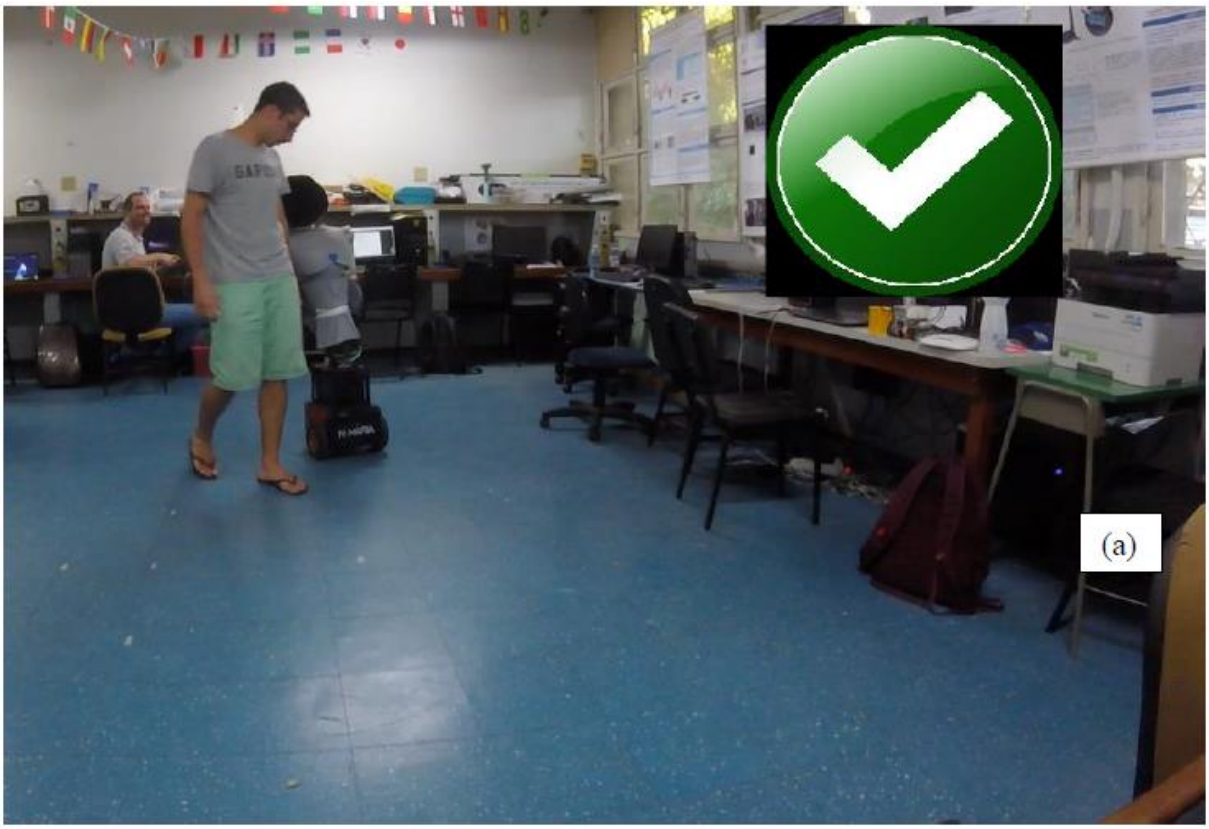
O percurso todo foi feito em aproximadamente 1 minuto, porém, este tempo pode variar dependendo da velocidade em que a criança se locomover, pois o robô ajusta a velocidade de acordo com a velocidade que a criança for se locomovendo. O código do robô N-MARIA encontra-se no Apêndice 2.

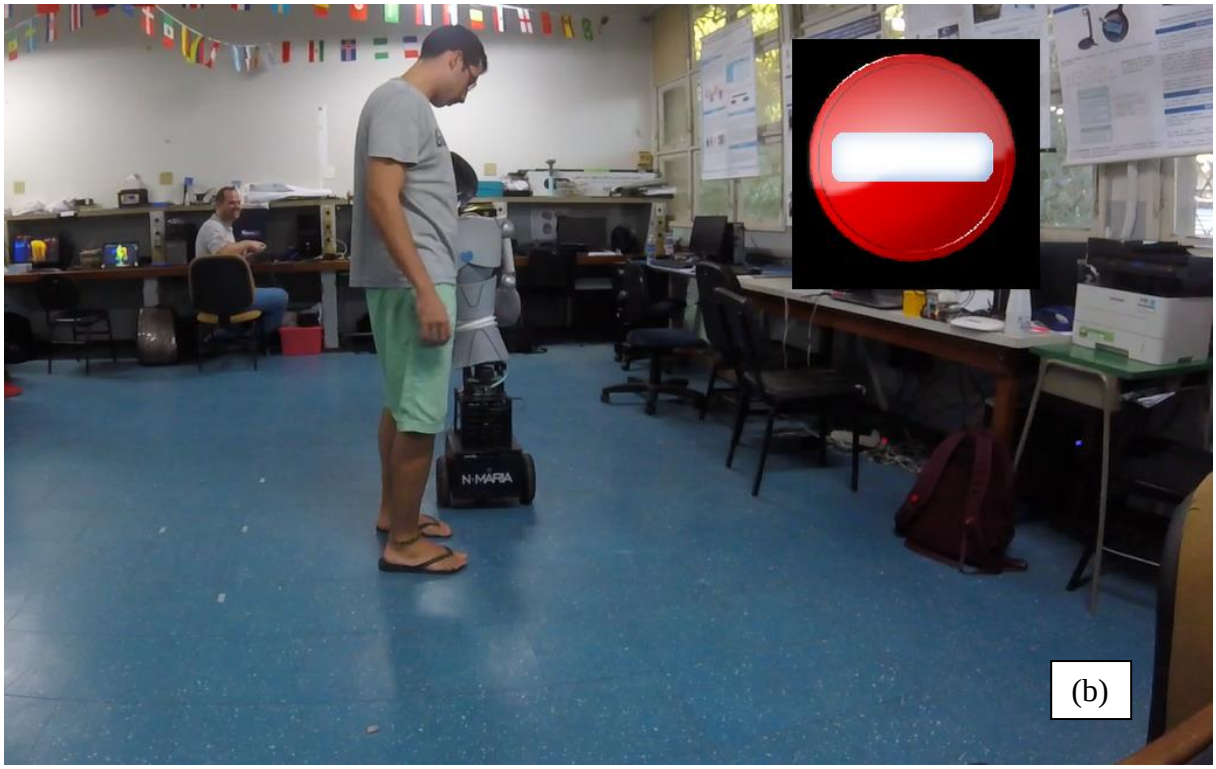
5.4 Experimento do sistema de segurança

Após os experimentos do comportamento "passeio" funcionarem em sua totalidade, foram realizados experimentos para testar o sistema de segurança do robô, ou seja, todas as situações citadas anteriormente em que o robô deve parar. Os experimentos foram feitos abrangendo todas as situações de segurança, porém, as que tiveram maior ênfase foram as que tiveram maior risco para o ser humano. Essas situações são quando a pessoa entra na rota do robô e quando a pessoa solta a mão do robô. Para ambos os casos as respostas ocorreram em tempo satisfatório (não foram instantâneas, pois necessitam de acessos ao servidor *web* que possui latência), ou

seja, o robô realizou parada em distância segura. A Figura 28 mostra o momento em que o robô realizou a parada quando detectou que a pessoa estava no caminho do robô, sendo que na Figura 28a o robô está em movimento e na Figura 28b é o instante em que o robô realiza a parada.

Figura 28 - Experimento do sistema de segurança.





Fonte: Produção do próprio autor.

5.5 Resultados

Com os experimentos realizados, foi possível calibrar e constatar o funcionamento correto dos sensores e atuadores do robô. Os sensores, após serem calibrados, responderam satisfatoriamente, tendo respostas praticamente instantâneas. Tanto os sensores de força resistivos quanto os capacitivos foram possíveis de serem ajustados para a sensibilidade desejada, podendo ser calibrado para casos diferentes.

A integração de sensores e atuadores tiveram resposta satisfatória em conjunto e possibilitaram a realização correta do comportamento “passeio”. Este comportamento mostrou-se bem projetado, tendo em vista que alcançou todas as expectativas desejadas tanto para o movimento e o controle de velocidade, quanto no sistema de segurança, mostrando coerência e segurança nas decisões tomadas.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi desenvolvido um comportamento de “passeio” e a instrumentação de um robô voltado à interação social com crianças com TEA. Foram testados também sensores que podem ser utilizados em projetos futuros.

Os resultados obtidos pela parte de instrumentação do robô demonstram que os sensores e atuadores do robô são capazes de cumprir seus respectivos propósitos. A instrumentação permitiu que o robô N-MARIA interagisse com o ambiente e pessoas em seu entorno.

Quanto aos sensores, o sensor laser RP Lidar possibilitou a detecção da criança, pois permitiu alcance e angulação necessários ao movimento. O fato de possibilitar uma leitura 360º permitiu uma varredura completa do ambiente em busca da pessoa que estiver realizando a interação.

Já o sensor de toque capacitivo MPR121 também se mostrou eficiente para o trabalho. Inicialmente, a escolha do material para o terminal de toque do sensor mostrou-se um problema que, porém, foi resolvido de forma a manter o projeto do robô e toda a logística de interação além, é claro, de possibilitar a captação de toques realizados pela pessoa que estiver interagindo com o robô N-MARIA.

A escolha do sensor de toque MPR121 mostrou-se acertada a partir do momento que possibilitou a instalação de terminais por todo o corpo do robô, permitindo a discriminação dos locais em que houveram contatos. Além disso, permitiu também que fossem detectados toques nos braços e mãos do robô, o que facilitou a lógica do comportamento de “passeio”.

Da mesma forma que os sensores citados anteriormente, o sensor de força resistivo apresentou-se adequado ao trabalho. A ideia de necessitar de um circuito divisor de tensão para a instalação, que primeiramente aparentou ser um problema, mostrou-se indiferente, tendo em vista o espaço interno do robô N-MARIA. Quanto à medição de força, este sensor possibilitou a diferenciação de toques e pressão, com precisão suficiente para o trabalho.

A escolha da utilização de um servidor *web* mostrou-se também uma alternativa acertada por permitir comunicação de dispositivos programados em diferentes linguagens e de diferentes sistemas. A comunicação entre computadores e entre sensores-computador, apesar de em alguns casos mostrar certa lentidão ao acessar o servidor, cumpriu satisfatoriamente o propósito para o qual foi planejado.

O comportamento “passeio”, que foi implementado pensando em uma interação social específica para crianças com TEA, trouxe resultados desejados, tendo em vista que atingiu os objetivos traçados, onde o robô mostrou-se capaz de guiar um movimento de passeio, conseguindo ajustar a velocidade de acordo com a pessoa que o acompanha. Além disso, também demonstrou um sistema de segurança, que para interações futuras com crianças, é de extrema importância, sendo que o mesmo foi colocado a prova em situações adversas e correspondeu de forma exemplar.

6.1 Trabalhos futuros

O trabalho em questão apresentou um comportamento de “passeio” e uma instrumentação de um robô para interação social em que os experimentos serviram como avaliação preliminar para a implantação do trabalho em um ambiente com crianças com TEA. Porém, para a implantação em interações com essas crianças, que necessitam de um cuidado maior, os seguintes pontos devem ser aprimorados:

- Controle de velocidade mais robusto para evitar que o robô sofra com acelerações/desacelerações constantes;
- Realização de experimentos prévios com crianças que não apresentem TEA, para análise e estudo de caso, verificando a forma de interação das crianças;
- Estudo da implementação de novos comportamentos auxiliares, tais como falas mais elaboradas;
- Estudo de situações fora do padrão esperado, para diminuir ainda mais as possibilidades de acidentes;
- Implementar outro conjunto de sensores de toque e permitir ligar terminais em regiões que ainda não foram implantadas e, assim, permitir uma análise mais completa da interação;
- Realizar experimentos com crianças com TEA para análise e estudo, e, desta forma, possibilitar que as informações coletadas sejam usadas para auxiliar na terapia dessas crianças;

REFERÊNCIAS

ADAWIKI. **Sensor FSR**. 2016. Disponível em: <http://www.ladyada.net/wiki/tutorials/learn/sensors/fsr.html>. Acesso em: 05/jul./18.
ADEPT. **Pioneer 3-DX**. *Adept Mobile Robots*, [s.l.], p. 2, 2011.

ASHIQ, M. et al. **Wireless Mobile Robotic Arm**. [s.l.], v. 41, nº Iris, p. 1072–1078, 2012. ISSN: 1877-7058, DOI: 10.1016/j.proeng.2012.07.285.

AUTISM RESEARCH INSTITUTE. **Instituto de Pesquisa Sobre Autismo: 40 Anos de Esperança e Sucesso**. 2013. Disponível em: <https://www.autism.com/TRANS_PORTUGUESE_INSTITUTO>. Acesso em: 27/jun./17.

BBC. **Robô ajuda crianças com autismo a se comunicar**. *British Broadcasting Corporationn (BBC)*. 2014. Disponível em: <https://www.bbc.com/portuguese/videos_e_fotos/2014/03/140311_robo_autismo_fn>. Acesso em: 27/jun./17.

BRAGA, C. **Perturbações do Espectro do Autismo e Inclusão: atitudes e representações dos pais, professores e educadores de infância**. *Universidade do Minho*, [s.l.], p. 130, 2010. ISSN: 1983-3482, DOI: 10.4013/ctc.2012.52.07.

DUQUETTE, A.; MICHAUD, F.; MERCIER, H. **Exploring the use of a mobile robot as an imitation agent with children with low-functioning autism**. *Autonomous Robots*, [s.l.], v. 24, nº 2, p. 147–157, 2008. ISBN: 0929-5593, ISSN: 09295593, DOI: 10.1007/s10514-007-9056-5.

ELECTRONICS, I. **Interlink Electronics Inc., FSR integration Guide and Evaluation parts catalog with suggested Electrical interfaces**. *Version 1.0, 90-45632 Rev. D, 2007*. [s.l.]: [s.n.], 2007. v. 1.0, 1-26 p. DOI: integration Guide and Evaluation.

FERREIRA, A. et al. **An Approach to Avoid Obstacles in Mobile Robot Navigation: The Tangential Escape**. *Sba: Controle & Automação Sociedade Brasileira de Automatica*, [s.l.], v. 19, nº 4, p. 395–405, 2008. ISBN: 1424404975, ISSN: 0103-1759, DOI: 10.1590/S0103-17592008000400003.

FONG, T.; NOURBAKHSH, I.; DAUTENHAHN, K. **A survey of socially interactive robots**. *Robotics and Autonomous Systems*, [s.l.], v. 42, nº 3–4, p. 143–166, 2003. ISBN: 09218890, ISSN: 09218890, DOI: 10.1016/S0921-8890(02)00372-X.

GOULART, C. **Uma Contribuição ao Estudo de Sinais de EEG para Avaliar Estados Emocionais e Mentais de Criança com Autismo na Interação com robô Móvel**. 2014.
INTEL. **Intel® NUC Kit D54250WYKH**. [s.l.]: [s.n.], 2016. 6-9 p.

KLIN, A. **Autismo e síndrome de Asperger: Uma visão geral**. *Revista Brasileira de*

Psiquiatria, [s.l.], v. 28, nº SUPPL. 1, p. 3–11, 2006. ISBN: 1516444620, ISSN: 15164446, DOI: 10.1590/S1516-44462006000500002.

MARKOM, M. A.; ADORN, A. H. **A Mapping Mobile Robot using RP Lidar Scanner**. [s.l.], p. 87–92, 2015. ISBN: 9781467371247.

MATTOS, L. K. De. **Reflexões sobre a inclusão escolar de uma criança com diag- nóstico de autismo na educação infantil Reflections about educational inclusion of a child with a diagnostic of autism in nursery schooling**. *Rev. Educ. Espec. Santa Maria*, [s.l.], v. 24, nº 39, p. 129–141, 2011.

OLIVEIRA, B. Q. De et al. **Tipos e aplicações de sensores na robótica**. *Caderno de Graduação - Ciências Exatas e Tecnológicas - UNIT - ALAGOAS*. [s.l.]: [s.n.], 2017. p. 223–229. ISBN: 2316-3135.

ONU BRASIL. **Especialistas da ONU em direitos humanos pedem fim da discriminação contra pessoas com autismo**. 2015. Disponível em: <https://nacoesunidas.org/especialistas-em-direitos-humanos-da-onu-pedem-fim-da-discriminacao-contra-pessoas-com-autismo/>.

PIRR@INFORMATICA. **Cabo adaptador conversor usb rs232 usb 2.0 x serial**. 2015. Disponível em: <https://www.pirrainformatica.com.br/552/cabo-adaptador-conversor-usb-serial-rs232-usb-20-x-serial?ml>. Acesso em: 05/jun./18.

ROBINS, B. et al. **Tactile interaction with a humanoid robot for children with autism: A case study analysis involving user requirements and results of an initial implementation**. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, [s.l.], p. 704–711, 2010. ISBN: 9781424479917, ISSN: 1944-9445, DOI: 10.1109/ROMAN.2010.5598641.

ROBOTSHOP. **Arduino Mega 2560**. *Retrieved November*. [s.l.]: [s.n.], 2011. 2560 p. ISBN: 9788578110796, ISSN: 1098-6596, DOI: 10.1017/CBO9781107415324.004.

SCASSELLATI, B.; HENNY ADMONI; MATARIĆ, M. **Robots for Use in Autism Research**. *Annual Review of Biomedical Engineering*, [s.l.], v. 14, nº 1, p. 275–294, 2012. ISBN: 978-0-8243-3514-4, ISSN: 1523-9829, DOI: 10.1146/annurev-bioeng-071811-150036.

ŠEKORANJA, B. et al. **Human-Robot Interaction Based on use of Capacitive Sensors**. [s.l.], v. 69, p. 464–468, 2014. ISBN: 0000000000, DOI: 10.1016/j.proeng.2014.03.013.

STEVEN, B. **Arduino Microcontroller: Processing for Everyone! Second Edition**. [s.l.]: [s.n.], 2012. 351 p. ISBN: 9781608454372, DOI: 10.2200 / S00421ED1V01Y201205DCS038.
SUSNEA, I.; VASILIU, G.; FILIPESCU, A. **Real-time , embedded fuzzy control of the Pioneer3-DX robot for path following 2 . THE STRUCTURE OF THE MOBILE ROBOT PIONEER3-DX . 4 . THE KINEMATIC MODEL OF**. [s.l.], nº July, 2008. ISBN: 9789606766831.

TRUCKER, E. **How Robots are Helping Children with Autism**. *The Guardian*. 2015. Disponível em: <https://www.theguardian.com/lifeandstyle/2015/feb/01/how-robots-helping-children-with-autism>. Acesso em: 19/jul./17.

VALADÃO, C. T. et al. **Analysis of the use of a robot to improve social skills in children with autism spectrum disorder**. *Research on Biomedical Engineering*, [s.l.], v. 32, nº 2, p. 161–175, 2016. ISSN: 2446-4740, DOI: 10.1590/2446-4740.01316.

VARELLA, M. H. **O Transtorno do Espectro Autista(TEA)**. 2014.

WAINER, J. et al. **Using the Humanoid Robot KASPAR to Autonomously Play Triadic Games and Facilitate Collaborative Play Among Children With Autism**. [s.l.], v. 6, nº 3, p. 183–199, 2014.

7 APÊNDICE 1 – CÓDIGO PYTHON

C:\Users\Vitor Scardua\Desktop\Eng. Elétrica\PG2\codigoPythonTXT.py sexta-feira, 6 de julho de 2018 13:34

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-
import time import serial import requests import os

'
comport = serial.Serial('/dev/ttyACM0', 9600)
#comport = serial.Serial("/dev/ttyUSB0", 9600, timeout=1) # Setando timeout 1s para a
conexao arquivo = open("log.txt", "w")


PARAM_CARACTER="t"
PARAM_ASCII=str(chr(116))        # Equivalente 116 = t
print("Capturando toque nos sensores...")
# Time entre a conexao serial e o tempo para escrever (enviar algo)
time.sleep(1.8) # Entre 1.5s a 2s

pingCheck = 1;
while pingCheck:
    pingCheck = os.system("ping -c 1 " + "localhost")

while 1:
#comport.write(PARAM_CARACTER)

    comport.write(PARAM_ASCII)

    VALUE_SERIAL=comport.readline()
    try:
        r = requests.get("http://192.168.0.110/action/buscarUltimaAcao.php")
        aux = r.content
        #time.sleep(2);
        r = int(aux.strip(""))

        idComportamento, tipoComportamento = VALUE_SERIAL.split(":")
        idComportamento = int(idComportamento)
        print r
        print idComportamento
        print VALUE_SERIAL
        #Caso a porta serial tiver valor maoir que as posicoes do sensor de toque,
        considerar como uma ação
        if idComportamento>11 and (r==10 or r==14 or r==15):
            if idComportamento == 16:
                r = requests.post("http://192.168.0.110/action/executarFalaRosto.php",
                data={"idFalaRosto": "16"})
                r = requests.post("http://192.168.0.110/action/cadastrarAcao.php",
                data={"idComportamento": "0"})
            else:
                r = requests.post("http://192.168.0.110/action/cadastrarAcao.php",
                data={"idComportamento": idComportamento})



        else:
                #print "\nRetorno da serial: %s" % (VALUE_SERIAL)
                idToque, tipoToque = VALUE_SERIAL.split(":")

                localtime = time.asctime( time.localtime(time.time()) )

                arquivo.write( (localtime) +" "+(idToque) +":"+ (tipoToque))

                r = requests.post("http://192.168.0.110/action/cadastrarToque.php",
                data={"idToque": idToque, "tipoToque": tipoToque})
    except:

        print "Falha no get/postman"


arquivo.close()
```

```
#print '\nRetorno da serial: %s' % (idToque)
#print '\nRetorno da serial: %s' % (tipoToque)

#r = requests.post("http://nmaria.000webhostapp.com/action/cadastrarExperimento.php",
data={'terapeutas_idTerapeuta': 1, 'criancas_idCrianca': 1, 'instituicoes_idInstituicao': 1})
#r = requests.post("http://nmaria.000webhostapp.com/action/finalizarExperimento.php",
data={'idTerapeuta': 1, 'idCrianca': 1, 'idInstituicao': 1})


#print(r.status_code, r.reason)
    #print(r.text[:300]+ '...')


# Fechando conexao serial
comport.close()
```

8 APÊNDICE 2 – CÓDIGO C++


```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <Aria.h>
#include "rplidar.h"
#include <pthread.h>
#include <iostream>
#include <string>
#include <sstream>
#include <curl/curl.h>
#include <time.h>

#ifndef _countof
#define _countof(_Array) (int)(sizeof(_Array) / sizeof(_Array[0]))
#endif

//variaveis globais++
volatile int j=0,a=0;
volatile int modo=0,modoAux=-1;
float vetorLidar[360] = {0};
volatile float distanciaCrianca, anguloCrianca,xCrianca,yCrianca;
volatile double anguloRobo, anguloRobo2, xRobo,yRobo;
volatile double anguloGiro=0;
volatile double erroAngular=0,erroAngular2=0,erroLinear=0,erroLinear2=0;
int distanciaSocial=400;
volatile int flagCrianca=0;
bool flagRoboCostas=0;
void *comunicacaoWebService(), leituraLidar(), posicaoCrianca_menorDistancia(), posicaoRobo();
void modoParado(), modoApresentacao(), modoGiro(), modoAproxima(), modoAfasta(),
modoSeguir(), modoFugir(), modoPasseio(), modoOrigem();
void escreveLog(), EnviaParada();
clock_t inicio, fim=clock();
time_t inicioLog,fimLog = time(NULL);
int modoAntigo=-1;
FILE * fip;
time_t rawtime;
struct tm * timeInfo;

//--

ArRobot robot;
ArRobotConnector* robotConnector;

using namespace rp::standalone::rplidar;
u_result     op_result;
RPlidarDriver * drv = RPlidarDriver::CreateDriver(RPlidarDriver::DRIVER_TYPE_SERIALPORT);

using namespace std;

string data; //will hold the url's contents

bool checkRPLIDARHealth(RPlidarDriver * drv)
{

    rplidar_response_device_health_t healthinfo;

    op_result = drv->getHealth(healthinfo);
    if (IS_OK(op_result))
    { // the macro IS_OK is the preperred way to judge whether the operation is succeed.
        printf("RPLidar health status : %d\n", healthinfo.status);
        if (healthinfo.status == RPLIDAR_STATUS_ERROR) {
            fprintf(stderr, "Error, rplidar internal error detected. Please reboot the
            device to retry.\n");
            // enable the following code if you want rplidar to be reboot by software
            // drv->reset();
            return false;
        }
        else
        {
            return true;
        }
```

```
    }
    else
    {
        fprintf(stderr, "Error, cannot retrieve the lidar health code: %x\n", op_result);
        return false;
    }
}

size_t writeCallback(char* buf, size_t size, size_t nmemb, void* up)
{
    //buf is a pointer to the data that curl has for us
    //size*nmemb is the size of the buffer

    for (int c = 0; c<size*nmemb; c++)
    {
        data.push_back(buf[c]);
    }
    return size*nmemb; //tell curl how many bytes we handled
}

void *comunicacaoWebService(void*)
{
while(1){

        inicio=clock();
        string comportamento;

        if(fabs(inicio-fim)>3000000)
        {
            comunicacao:

            //cout << endl << clock() << endl;
             posicaoCrianca_menorDistancia();
             posicaoRobo();
             char buffer[50]="";
             //double xRobo=a, yRobo=a, xCrianca=a, yCrianca=a;
             sprintf(buffer,"xRobo=%f&yRobo=%f&xCrianca=%f&yCrianca=%f",xRobo, yRobo,
             xCrianca, yCrianca);

             CURL* curl; //our curl object

             curl_global_init(CURL_GLOBAL_ALL); //pretty obvious
             curl = curl_easy_init();

             curl_easy_setopt(curl, CURLOPT_POSTFIELDS, buffer);
             //curl_easy_setopt(curl, CURLOPT_URL,
             "http://nmaria.000webhostapp.com/action/controle.php");
             curl_easy_setopt(curl, CURLOPT_URL, "http://192.168.0.110/action/controle.php");
             curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, &writeCallback);
             //curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L); //tell curl to output its
             progress

             curl_easy_perform(curl);

                 if (data.length() != 0) {
                 comportamento = data.substr((data.find("comportamento", 0)+16), 2);
                 modo=stoi(comportamento);
                 data="";
                 cout << endl << modo << endl;

                 curl_easy_cleanup(curl);
                 curl_global_cleanup();
             }
             else goto comunicacao;

             fim = clock();

        }

    }
```

```
}

void EnviaParada() //Envia modo 0 para servidor
{
    CURL *hnd = curl_easy_init();

    curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "POST");
    curl_easy_setopt(hnd, CURLOPT_URL, "http://192.168.0.110/action/cadastrarAcao.php");

    struct curl_slist *headers = NULL;
    headers = curl_slist_append(headers, "postman-token:
    3ffe7b8a-8b97-ff8b-d242-9ed3e73dd58f");
    headers = curl_slist_append(headers, "cache-control: no-cache");
    headers = curl_slist_append(headers, "content-type: multipart/form-data;
    boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW");
    curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

    curl_easy_setopt(hnd, CURLOPT_POSTFIELDS,
    "------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data;
    name=\"idComportamento\"\r\n\r\n0\r\n------WebKitFormBoundary7MA4YWxkTrZu0gW--");
    //curl_easy_setopt(hnd, CURLOPT_POSTFIELDS,
    "------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data;
    name=\"idComportamento\"\r\n\r\n0\r\n------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Di
    sposition: form-data;
    name=\"idFalaRosto\"\r\n\r\n3\r\n------WebKitFormBoundary7MA4YWxkTrZu0gW--"); com fala
    rosto

    CURLcode ret = curl_easy_perform(hnd);
}

void escreveLog()
{

        inicioLog= time(NULL);
//printf("%ld %ld\n",inicioLog,fimLog);
        if ( (inicioLog-fimLog)>1)
        {
//printf("entrei\n");
            time ( &rawtime );
            timeInfo= localtime ( &rawtime );
            fprintf(fip, "%s    %d  %f  %f  %f  %f  %f  %f  %f  %d  %f  %f  %f  %f
            \n",asctime (timeInfo),modo,anguloRobo,
            xRobo,yRobo,erroAngular,erroAngular2,erroLinear,erroLinear2,flagCrianca,distanciaC
            rianca,anguloCrianca,xCrianca,yCrianca);
            fimLog=time(NULL);
        }

}

void leituraLidar()                         //feito
{
    rplidar_response_measurement_node_t nodes[360*2];
    size_t    count = _countof(nodes);
    op_result = drv->grabScanData(nodes, count);
    if (IS_OK(op_result))
    {
        drv->ascendScanData(nodes, count);
        for (int pos = 0; pos < (int)count ; ++pos)
        {
            float theta = (nodes[pos].angle_q6_checkbit >>
            RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f;
            float dis=nodes[pos].distance_q2/4.0f;
            theta=abs(theta-360);
            int theta2=round(theta);
            vetorLidar[theta2]=dis;
            //printf("%f %f\n",dis,theta);
        }
    }
}
```

```
void posicaoCrianca_menorDistancia()
{
    leituraLidar();
    flagCrianca = false;
    distanciaCrianca=5000;

    for (int ang = 0; ang < 360 ; ++ang) // faz a analise com as medidas de cada angulo
    {
        posicaoRobo(); //obtem a posição do robô
        float anguloRoboAux=anguloRobo;
        if (anguloRoboAux<0) anguloRoboAux=anguloRoboAux+360; //angulo do robô é de 180 a
        -180. Aqui ele fica de 0 a 360.
        float xObj=(cos((ang+anguloRoboAux)*M_PI/180)*vetorLidar[ang])+xRobo; //calcula a
        posição x do objeto
        float yObj=(sin((ang+anguloRoboAux)*M_PI/180)*vetorLidar[ang])+yRobo; //calcula a
        posição y do objeto
        if ((vetorLidar[ang]>300)&&(abs(xObj)<4000)&&(abs(yObj)<4000)) // se o objeto estive
        dentro de um quadrado de 1800mm, entra no "if"
        {
            if (vetorLidar[ang]<distanciaCrianca) // se o objeto estiver mais perto que o
            registro do ultimo objeto mais perto, entra  no "if"
            {
                distanciaCrianca = vetorLidar[ang];
                anguloCrianca=ang;
                xCrianca = (cos((ang+90)*M_PI/180)*vetorLidar[ang]);
                yCrianca = (sin((ang+90)*M_PI/180)*vetorLidar[ang]);
                flagCrianca=true;
            }
        }
        else vetorLidar[ang]=5000;
    }
}

void posicaoRobo()                          //feito
{
    anguloRobo=robot.getTh();
    xRobo=robot.getX();
    yRobo=robot.getY();
    //printf("%f\n",xRobo);

}


void modoParado()                           //feito modo=0
{
    printf("Modo Parado\n");
    while(modo==0)
    {
        robot.setRotVel(0);
        robot.setVel(0);
    }
}

void modoApresentacao ()                    //feito modo=1
{
    printf("Modo Apresentação\n");
    while (modo==1)
    {

    }
}

void modoGiro ()                            //feito modo=2
{
    printf("Modo Giro\n");
    float anguloRoboG=robot.getTh();
    if (anguloRoboG<0) anguloRoboG=-1*anguloRoboG;
    double anguloGirado=0;
    double anguloAnterior=anguloRoboG;
    erroAngular2=0;
    erroAngular=0;
```

```
    erroLinear=0;
    erroLinear2=0;
    anguloGiro=360;
    while(modo==2||modoAux==2)
    {
        escreveLog();

        anguloRoboG=robot.getTh();

        if (anguloRoboG<0) anguloRoboG=-1*anguloRoboG;
        anguloGirado=anguloGirado+fabs(anguloRoboG-anguloAnterior);
        anguloAnterior=anguloRoboG;
        erroAngular=anguloGiro-anguloGirado;
        erroAngular2=1*tanh((M_PI/180)*erroAngular/2);
        robot.setRotVel(1*(erroAngular2*180/M_PI));

        if(fabs(erroAngular)<(1))
        {
            robot.setRotVel(0);
            modoAux=0;
            while(modo==2)
            {
                robot.setRotVel(0);
            }
        }
    }
}

void modoAproxima ()                    //feito modo=3
{
    printf("Modo Aproxima\n");
    posicaoRobo();
    posicaoCrianca_menorDistancia();
    erroAngular=0;
    erroAngular2=0;
    erroLinear=0;
    erroLinear2=0;
    distanciaSocial=400;
    while((modo==3||modoAux==3))
    {
        if (flagCrianca==1)
        {
            escreveLog();

            posicaoRobo();
            posicaoCrianca_menorDistancia();

            if (anguloRobo<0) anguloRobo=360+anguloRobo;
            erroAngular=anguloCrianca;
            if(erroAngular>180)erroAngular=erroAngular-360;
            erroAngular2=2.5*tanh((M_PI/180)*erroAngular);
            robot.setRotVel((erroAngular2*180/M_PI)*0.5);

            if(fabs(erroAngular)<(1)) robot.setRotVel(0);

            erroLinear=distanciaCrianca-distanciaSocial;
            erroLinear2 = ((0.5*tanh(erroLinear/1000))*1000);
            robot.setVel((erroLinear2)*0.8);

            if(fabs(erroLinear2)<100)
            {
                while(modo==3)
                {
                    robot.setVel(0);
                    robot.setRotVel(0);
                }
            }
        }
        else
        {
            robot.setRotVel(0);
```

```
            robot.setVel(0);
        }
    }
}

void modoAfasta ()                          //feito modo=4
{
    printf("Modo Afasta\n");
    posicaoRobo();
    posicaoCrianca_menorDistancia();
    erroAngular=0;
    erroAngular2=0;
    erroLinear=0;
    erroLinear2=0;
    distanciaSocial=1500;
    while((modo==4||modoAux==4))
    {
        if (flagCrianca==1)
        {
            escreveLog();

            posicaoRobo();
            posicaoCrianca_menorDistancia();

            if (anguloRobo<0) anguloRobo=360+anguloRobo;
            erroAngular=anguloCrianca;
            if(erroAngular>180)erroAngular=erroAngular-360;
            erroAngular2=2.5*tanh((M_PI/180)*erroAngular);
            robot.setRotVel((erroAngular2*180/M_PI)*0.5);

            if(fabs(erroAngular)<(1)) robot.setRotVel(0);

            erroLinear=distanciaSocial-distanciaCrianca;
            erroLinear2 = ((0.5*tanh(erroLinear/1000))*1000);
            robot.setVel((-erroLinear2)*0.8);

            if((fabs(erroLinear2)<100)||(fabs(xRobo)>1900)||(fabs(yRobo)>1900))
            {
                while(modo==4)
                {
                    robot.setVel(0);
                    robot.setRotVel(0);
                }
            }
        }
        else
        {
            robot.setRotVel(0);
            robot.setVel(0);
        }
    }
}

void modoSeguir ()                          //feito modo=5
{
    printf("Modo Seguir\n");
    posicaoRobo();
    posicaoCrianca_menorDistancia();
    erroAngular=0;
    erroAngular2=0;
    erroLinear=0;
    erroLinear2=0;
    distanciaSocial=400;
    while(modo==5)
    {
        if (flagCrianca==1)
        {
            escreveLog();

            posicaoRobo();
            posicaoCrianca_menorDistancia();
```

```
            if (anguloRobo<0) anguloRobo=360+anguloRobo;
            erroAngular=anguloCrianca;
            if(erroAngular>180)erroAngular=erroAngular-360;
            erroAngular2=2.5*tanh((M_PI/180)*erroAngular);
            robot.setRotVel((erroAngular2*180/M_PI)*0.5);

            if(fabs(erroAngular)<(1)) robot.setRotVel(0);

            erroLinear=distanciaCrianca-distanciaSocial;
            erroLinear2 = ((0.5*tanh(erroLinear/1000))*1000);
            robot.setVel((erroLinear2)*0.8);

            if(fabs(erroLinear2)<100)
            {
                robot.setVel(0);
            }
        }
        else
        {
            robot.setRotVel(0);
            robot.setVel(0);
        }
    }
}

void modoFugir ()                    //vazio
{
printf("Modo Fugir\n");

}


void modoPasseio()                   //Modo 14 e 15
{

printf("Modo Passeio\n");

//***************************************** Modo Passeio (Direita)
*********************************************
    if(modo == 14) {
                flagCrianca = false;
            // printf("%d",(int)flagCrianca);
            while(flagCrianca == false){ //até localizar crianca
                posicaoCrianca_menorDistancia();//Busca a posição da Criança
            }

            /*ArKeyHandler keyHandler;
            Aria::setKeyHandler(&keyHandler);
            robot.attachKeyHandler(&keyHandler);
            printf("You may press escape to exit\n");*/

            // Start the robot processing cycle running in the background.
            // True parameter means that if the connection is lost, then the
            // run loop ends.

            //robot.runAsync(true);

            // Print out some data from the SIP.

            // We must "lock" the ArRobot object
            // before calling its methods, and "unlock" when done, to prevent conflicts
            // with the background thread started by the call to robot.runAsync() above.
            // See the section on threading in the manual for more about this.
            // Make sure you unlock before any sleep() call or any other code that will
            // take some time; if the robot remains locked during that time, then
            // ArRobot's background thread will be blocked and unable to communicate with
            // the robot, call tasks, etc.
```

```
// Inicializando dos parametros do robo
robot.setTransVelMax(400);
robot.setRotVelMax(14.33);

clock_t t_inicial, t_momento;
double tempo_gasto;
int Cont_erro = 0;
robot.moveTo(ArPose(0,0,0),1); //Reset nas coordenadas do robo
double Vel = 0; // Inicializa velocidade linear
double VelRot = 0; // Inicializa velocidade angular
double Vetor_Vel[3] = {0,0,0};


// Processo acionado caso a criança segure a mão direita do robô N-Maria
t_inicial = clock();

// ArLog::log(ArLog::Normal, "X %.2lf  Y %.2lf  M %d", robot.getX(),
robot.getY(), modo);

while (robot.getX() < -100 || robot.getX() >= -1 || robot.getY() < -200) // Para
proximo do ponto inicial
{
    inicio_dir:
    t_momento = clock();
    tempo_gasto = (t_momento - t_inicial) / (double)CLOCKS_PER_SEC;
    posicaoRobo();      //Busca posição do robô
    posicaoCrianca_menorDistancia();//Busca a posição da Criança
    //printf("Desativado: %f\n",xCrianca);
    if(xCrianca < 0) goto inicio_dir; // ignora o hemisfério esquerdo
    // printf("Ativo: %f\n",xCrianca);
    // robot.lock();
    if(xCrianca > 0){ // só observa criança no hemisfério direito
    //calculo da velocidade em relação a posição da criança
    Vel = 0.5*yCrianca+250; //Equação da reta para velocidade máxima seja 400
    mm/s
    // For para criação de vetor de suavização da velocidade
    for(int i=1; i==2; i++)
    {
        Vetor_Vel[i-1]=Vetor_Vel[i];
    }
    Vetor_Vel[2] = Vel;
    int Kp = 3; // ganho proporcional para aumentar velocidade no meio da
    trajetória
    Vel = Kp*tanh(tempo_gasto)*(Vetor_Vel[0]+Vetor_Vel[1]+Vetor_Vel[2])/3; //
    Velocidade linear filtrada pela média para suavizar a velocidade //
    (Acrescentei a tangente para suavizar o tempo no inicio)
    Vetor_Vel[2] = Vel; // Utiliza o valor da media para a proxima suavização
    VelRot = -180 * Vel / (M_PI * 2000); //Equação velocidade angular deg/s
    (eixos do robo são rotacionados, logo, sentido horário com velocidade
    negativa)
    }

    if (Vel<0) Vel=0;
    if(modo != 14)
    {
        Vel = 0;
        VelRot = 0;
        return;
    }
    robot.setVel(Vel); // Inicializa velocidade linear no robo
    robot.setRotVel(VelRot); // Inicializa velocidade angular no robo negativa

    ArLog::log(ArLog::Normal, "simpleMotionCommands: Pose=(%.2lf,%.2f,%.2f),
    Trans. Vel=%.4lf, Rot. Vel=%.4lf, Battery=%.2fV", robot.getX(),
    robot.getY(), robot.getTh(), robot.getVel(), robot.getRotVel(),
    robot.getBatteryVoltage());

    // robot.unlock();

    if (tempo_gasto > 180) break; // Para o processo caso ocorra algum erro e o
    processo esteja rodando a mais de 3 min
```

```
                /* Este If e else seguintes foram colocados para que o robo não pare em
                casos de eventuais erros de leitura do RPLidar */
                if (xCrianca < 270 || xCrianca > 1000 || yCrianca >500 || yCrianca < -400)
                Cont_erro++; // Caso a criança fique muito para trás ou se adiante muito
                robo, incrementa contador de erro de faixa
                else Cont_erro = 0; //Caso a criança volte a faixa, zera contador
                /* Se fizer 10 leituras fora da faixa, para o robo */
                if(Cont_erro == 5)
                {
                    Vel = 0;
                    VelRot = 0;
                    Cont_erro = 0;
                    goto fim_D;
                }
            }


    fim_D:
    if(modo != 14)
    {
        Vel = 0;
        VelRot = 0;
        return;
    }
    ArLog::log(ArLog::Normal, "N-Maria parando");
    robot.setVel(0);
    robot.setRotVel(0);

    // wait for the thread to stop

    EnviaParada(); //Envia modo 0 para servidor

    // ArLog::log(ArLog::Normal, "simpleMotionCommands: Exiting.");
    //robot.stopRunning();
    //robot.waitForRunExit();

 } //Termino da rotina da mão direita

//****************************** Modo Passeio (Esquerda) **********************************
    if (modo==15){
        flagCrianca = false;
        while(flagCrianca == false){ //até localizar crianca
        posicaoCrianca_menorDistancia();//Busca a posição da Criança
        }

        /*ArKeyHandler keyHandler;
        Aria::setKeyHandler(&keyHandler);
        robot.attachKeyHandler(&keyHandler);
        printf("You may press escape to exit\n");*/

        // Start the robot processing cycle running in the background.
        // True parameter means that if the connection is lost, then the
        // run loop ends.

        //robot.runAsync(true);

        // Print out some data from the SIP.

        // We must "lock" the ArRobot object
        // before calling its methods, and "unlock" when done, to prevent conflicts
        // with the background thread started by the call to robot.runAsync() above.
        // See the section on threading in the manual for more about this.
        // Make sure you unlock before any sleep() call or any other code that will
        // take some time; if the robot remains locked during that time, then
        // ArRobot's background thread will be blocked and unable to communicate with
        // the robot, call tasks, etc.


        // Inicializando dos parametros do robo
        robot.setTransVelMax(400);
```

```
robot.setRotVelMax(14.33);

clock_t t_inicial, t_momento;
double tempo_gasto;
int Cont_erro = 0;
robot.moveTo(ArPose(0,0,0),1); //Reset nas coordenadas do robo
double Vel = 0; // Inicializa velocidade linear
double VelRot = 0; // Inicializa velocidade angular
double Vetor_Vel[3] = {0,0,0};


// Processo acionado caso a criança segure a mão esquerda do robô N-Maria
    t_inicial = clock();
    // ArLog::log(ArLog::Normal, "X %.2lf  Y %.2lf  M %d", robot.getX(),
    robot.getY(), modo);
    while ((robot.getX() < -100 || robot.getX() >= -1 || robot.getY() > 200)) //
    Para proximo do ponto inicial
    {
        inicio_esq:
        t_momento = clock();
        tempo_gasto = (t_momento - t_inicial) / (double)CLOCKS_PER_SEC;
        posicaoRobo();      //Busca posição do robô
        posicaoCrianca_menorDistancia();//Busca a posição da Criança
        if(xCrianca > 0) goto inicio_esq; // ignora o hemisfério direito
        //printf("%lf",xCrianca);
        //robot.lock();
        if(xCrianca < 0){ // realiza leitura somente no hemisferio esquerdo
        //calculo da velocidade em relação a posição da criança
        Vel = 0.5*yCrianca+250; //Equação da reta para velocidade máxima seja 400
        mm/s
        // For para criação de vetor de suavização da velocidade
        for(int i=1; i==2; i++)
        {
            Vetor_Vel[i-1]=Vetor_Vel[i];
        }
        Vetor_Vel[2] = Vel;
        int Kp = 3; // ganho proporcional para aumentar velocidade no meio da
        trajetória
        Vel = Kp*tanh(tempo_gasto)*(Vetor_Vel[0]+Vetor_Vel[1]+Vetor_Vel[2])/3; //
        Velocidade linear filtrada pela média para suavizar a velocidade //
        (Acrescentei a tangente para suavizar o tempo no inicio)
        Vetor_Vel[2] = Vel; // Utiliza o valor da media para a proxima suavização
        VelRot = 180 * Vel / (M_PI * 2000); //Equação velocidade angular deg/s
        }
        if(Vel<0)Vel=0;
        if(modo != 15)
        {
            Vel = 0;
            VelRot = 0;
            return;
        }
        robot.setVel(Vel); // Inicializa velocidade linear no robo
        robot.setRotVel(VelRot); // Inicializa velocidade angular no robo

        ArLog::log(ArLog::Normal, "simpleMotionCommands: Pose=(%.2lf,%.2f,%.2f),
        Trans. Vel=%.4lf, Rot. Vel=%.4lf, Battery=%.2fV", robot.getX(),
        robot.getY(), robot.getTh(), robot.getVel(), robot.getRotVel(),
        robot.getBatteryVoltage());

        //robot.unlock();

        if (tempo_gasto > 180) break; // Para o processo caso ocorra algum erro e o
        processo esteja rodando a mais de 3 min

        /* Este If e else seguintes foram colocados para que o robo não pare em
        casos de eventuais erros de leitura do RPLidar */
        if (xCrianca > -270 || xCrianca < -1000 || yCrianca > 500 || yCrianca <
        -400) Cont_erro++; // Caso a criança fique muito para trás ou se adiante
        muito robo, incrementa contador de erro de faixa
        else Cont_erro = 0; //Caso a criança volte a faixa, zera contador
        /* Se fizer 10 leituras fora da faixa, para o robo */
```

```
                if(Cont_erro == 5)
                {
                    Vel = 0;
                    VelRot = 0;
                    Cont_erro = 0;
                    goto fim_E;
                }
            }
        fim_E:

        if(modo != 15)
        {
            Vel = 0;
            VelRot = 0;
            return;
        }
        ArLog::log(ArLog::Normal, "N-Maria parando");

        // wait for the thread to stop
        robot.setVel(0);
        robot.setRotVel(0);
        EnviaParada(); //Envia modo 0 para o servidor

        // ArLog::log(ArLog::Normal, "simpleMotionCommands: Exiting.");

        //robot.waitForRunExit();
    } //Termino rotina mão esquerda
}
//*****************************************************************************************
*********************************************


void modoOrigem()                        //modo=9
{
    printf("Modo Origem\n");
    posicaoRobo();
    erroAngular=0;
    erroAngular2=0;
    erroLinear=0;
    erroLinear2=0;
    robot.setVel(0);
    robot.setRotVel(0);
    while(modo==9)
    {
        escreveLog();

        posicaoRobo();
        if (anguloRobo<0) anguloRobo=360+anguloRobo;
        erroAngular=180-(atan2(-yRobo,xRobo)*180/M_PI)-anguloRobo;
        if(erroAngular>180)erroAngular=erroAngular-360;
        if(-180>erroAngular)erroAngular=erroAngular+360;

        erroAngular2=2.5*tanh((M_PI/180)*erroAngular);
        robot.setRotVel((erroAngular2*180/M_PI)*0.5);

        if(fabs(erroAngular)<(1))
        {
            robot.setRotVel(0);
            erroLinear=sqrt((xRobo*xRobo)+(yRobo*yRobo));
            erroLinear2 = ((0.5*tanh(erroLinear/1000))*1000);
            robot.setVel((erroLinear2)*0.8);

            if(fabs(erroLinear)<50)
            {
                robot.setVel(0);
                modo=0;
            }
        }
    }
}

int main(int argc, char** argv) //feito
```

```
{
fip = fopen("teste.txt", "w");

//printf("1\n");
fprintf(fip, "hora  modo   anguloRobo  xRobo   yRobo   erroAngular erroAngular2
erroLinear  erroLinear2 flagCrianca distanciaCrianca   anguloCrianca   xCrianca
yCrianca\n");
fimLog = time(NULL);
// RPLIDAR*********************************************************************

    const char * opt_com_path = "/dev/ttyUSB0"; // Modificado para ajustar ao reset da USB
    _u32         opt_com_baudrate = 115200;
    u_result     op_result;

    // create the driver instance
    drv = RPlidarDriver::CreateDriver(RPlidarDriver::DRIVER_TYPE_SERIALPORT);

    if (!drv) {
        fprintf(stderr, "insufficent memory, exit\n");
        //exit(-2);
    }

    // make connection...
    if (IS_FAIL(drv->connect(opt_com_path, opt_com_baudrate))) {
        fprintf(stderr, "Error, cannot bind to the specified serial port %s.\n"
            , opt_com_path);
        //exit(-1);
    }

    // check health...
    if (!checkRPLIDARHealth(drv)) {
        //exit(-1);
    }

    // start scan...
    drv->startMotor();
    drv->startScan();

//printf("2\n");
//ARIA*************************************************************************

Aria::init();
//ArRobot robot;

  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();

  ArLog::log(ArLog::Terse, "Moving the robot...");

  robotConnector = new ArRobotConnector(&parser, &robot);

  if(!robotConnector->connectRobot())
  {
    ArLog::log(ArLog::Terse, "simpleMotionCommands: Could not connect to the robot.");
    if(parser.checkHelpAndWarnUnparsed())
    {
        Aria::logOptions();
        Aria::exit(1);
    }
  }
  if (!Aria::parseArgs())
  {
    Aria::logOptions();
    Aria::shutdown();
    return 1;
  }

  ArLog::log(ArLog::Normal, "simpleMotionCommands: Connected.");

  robot.runAsync(true);
```

```
  // for 1 second.
  ArLog::log(ArLog::Normal, "simpleMotionCommands: Will start driving in 1 second...");
  ArUtil::sleep(1000);

//INICIO*****************************************************************************
//printf("3\n");
    robot.lock();
    robot.enableMotors();
    robot.unlock();
//printf("4\n");
    pthread_t t1;
//printf("5\n");
    pthread_create(&t1, NULL, &comunicacaoWebService, NULL);
//printf("6\n");
    while(1)
    {
    printf("%d %d\n",modo,modoAntigo);

    switch(modo)

                {
                    case 0:
                        modoParado();
                    break;
                    case 1:
                        modoApresentacao();
                    break;

                    case 2:
                        anguloGiro=360;
                        modoGiro();
                    break;
                    case 3:
                        modoAproxima();
                    break;
                    case 4:
                        modoAfasta();
                    break;
                    case 5:
                        modoSeguir();
                    break;
                    case 6:
                        modoFugir();
                    break;
                    case 14:
                        modoPasseio();
                    break;
                    case 15:
                        modoPasseio();
                    break;
                    case 9:
                        modoOrigem();
                    break;
                    case 8:
                        fclose(fip);
                        pthread_exit(NULL);
                        // finaliza o rplidar
                        RPlidarDriver::DisposeDriver(drv);
                        // finaliza o robô
                        robot.waitForRunExit();
                        Aria::exit(0);
                        return 0;
                    break;
                }


 }

}
```