UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CAETANO MAZZONI RANIERI

SISTEMA PARA NAVEGAÇÃO DE ROBÔS MÓVEIS E INTERAÇÃO HUMANO-ROBÔ BASEADA EM COMANDOS DE VOZ

BAURU
2013

CAETANO MAZZONI RANIERI

SISTEMA PARA NAVEGAÇÃO DE ROBÔS MÓVEIS E INTERAÇÃO HUMANO-ROBÔ BASEADA EM COMANDOS DE VOZ

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho", para obtenção de título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Humberto Ferasoli Filho

BAURU

2013

RESUMO

O presente projeto consiste no desenvolvimento de um ambiente interativo de robótica móvel, voltado para interação humano-robô. O sistema foi desenvolvido para funcionar em um *smartphone*, com sistema operacional Android, embarcado em um robô móvel de pequeno porte. Informações provenientes da câmera e do microfone do *smartphone*, bem como de alguns sensores de proximidade embarcados no robô, são usadas como entradas em uma arquitetura de controle, implementada em *software*. Trata-se de uma arquitetura de controle baseada em comportamentos e receptiva a comandos humanos para assistir a navegação do robô, controlado por comportamentos próprios ou por comandos humanos.

Palavras-chave: *robótica móvel*, *robótica social*, *interface humano-robô*, *arquitetura de controle.*

ABSTRACT

This project is comprised by an interactive mobile robotics' environment, focused in human-robot interaction. The system was developed to work in a smartphone, with Android operating system, embedded in a small size mobile robot. Information provided by the smartphone's camera and microphone, as well as by proximity sensors embedded in the robot, is used as inputs of a control architecture, implemented in software. It is a behavior-based and receptive to human commands control architecture, to assist the robot's navigation. The robot is controlled by its own behaviors or by commands emitted by humans.

Keywords: *mobile robotics, social robots, human-robot interface, control architecture.*

SUMÁRIO

1. INTRODUÇÃO

O interesse em projetos de robôs sociais encontra diferentes motivações. Para crianças, esses robôs podem servir como brinquedos ou educadores. Para adultos, como assistentes em ambientes de trabalho ou em casa. Para idosos, como companhia ou ferramenta para suprir limitações físicas (MALFAZ e SALICHS, 2004). O desenvolvimento de interfaces humano-robô tem sido tema bastante explorado em pesquisas envolvendo robótica social ou assistiva. Em alguns casos, faz-se uso de arquiteturas baseadas em emoções para facilitar e tornar crível essa interação, bem como para favorecer estudos relativos à neurobiologia de seres vivos e criar um campo de testes para teorias acerca de emoções em seres vivos (ARBIB e FELLOUS, 2004).

Ainda que não exista, na atualidade, um robô que possa ser entendido como ser vivo, trabalhos sugerem que crianças podem perceber robôs como formas de vida (TURKLE, 2007). Entende-se que crianças podem conceber robôs com características humanas e inventar histórias que retratam robôs como seres sociais, fato possivelmente inspirado pela popularização de heróis robóticos em livros infantis, revistas em quadrinhos, videogames e desenhos animados, entre outros (BUMBY e DAUTENHAHN, 1999). Portanto, a criação de robôs dotados de comportamentos com os quais a criança possa interagir pode propiciar um cenário promissor para o desenvolvimento de atividades lúdicas ou pedagógicas.

O Grupo de Integração de Sistemas e Dispositivos Inteligentes (GISDI), da Universidade Estadual Paulista (UNESP), vem desenvolvendo, em parceria com pesquisadores de outras universidades, projetos visando o uso da robótica móvel em atividades com crianças portadoras de deficiência motora (VALADÃO, *et al.*, 2011). Em trabalhos mais recentes, tem-se feito o uso de *smartphones* para controle de características funcionais dos robôs e para a interação com o ser humano, fazendo uso da grande disponibilidade de sensores presente nesses dispositivos (ALVES, *et al.*, 2013).

O propósito deste projeto é desenvolver uma arquitetura de controle baseada em comportamentos para robôs móveis de pequeno porte. A ideia é que o usuário possa interagir com o robô de forma semelhante à interação com um animal de estimação. O sistema foi desenvolvido para funcionar em *smartphones*, com sistema operacional Android, embarcados em um robô móvel de pequeno porte. Espera-se que esse sistema possa ser usado no desenvolvimento de atividades com crianças, além de servir como plataforma para desenvolvimento de novos projetos de interface humano-robô.

2. TRABALHOS CORRELATOS

O uso de interação social entre humano e robô tem sido bastante explorado, com diferentes motivações. O sistema GIR-T (*Gestural Interface and Robotic Technology*) (LATHAN e MALLEY, 2001) foi construído com propósitos terapêuticos e educacionais, voltado a atividades com crianças. O robô, concebido no formato de um brinquedo, pode movimentar a cabeça, a boca e o corpo, levantar os braços e deslizar sobre uma superfície. Esse sistema possibilita, entre outras aplicações, uma função em que o robô conta uma história e faz gestos para enfatizar emoções. A criança pode interagir com o robô, que responde à sua voz. Pode-se realizar *download* ou *upload* de histórias por meio da Internet.

Para crianças com algum tipo de deficiência, diferentes interfaces podem adequar-se melhor ao tipo de deficiência em questão. Em Minguez et al. (2007), três diferentes interfaces para controle de uma cadeira de rodas elétrica são apresentadas. A primeira interface descrita é um sistema de reconhecimento de fala, indicado para crianças com pouco controle de suas habilidades físicas, mas que tenham capacidade de falar. A segunda interface é um botão, que gera comandos de acordo com o número de vezes que é pressionado. A terceira consiste em uma tela sensível ao toque oferece um mapa em um monitor, em que a localidade de destino é selecionada.

A interação social entre humanos e robôs requer que o humano possa atribuir intencionalidade ao robô, isto é, que seja capaz de atribuir-lhe crenças, desejos e intenções. Para isso, o robô deve dar pistas às quais humanos têm tendência natural a responder, como direção do olhar, postura, gestos, entonação da voz, expressão facial. Reconhecer as pistas de intencionalidade permite predizer comportamentos, responder corretamente a interações sociais e engajar-se em atos comunicativos (BREAZEAL e SCASSELLATI, 1999). Uma forma de comunicar essas pistas é por meio de expressões faciais, comumente utilizadas em projetos de robótica voltados a interação humano-robô.

O trabalho apresentado em Aoyama e Shimomura (2005) trata de interação humano-robô baseada em reconhecimento e síntese de fala. Dois tipos de comportamento verbal são apresentados: comportamentos de interação por fala em nível de conteúdo (CSB – *Content level Speech interaction Behavior*) e comportamentos de suporte à naturalidade (NSB – *Naturalness Support Behavior*). Os CSB baseiam-se na compreensão e na geração de resposta para o usuário, podendo demandar processos computacionalmente caros que inviabilizam resposta

rápida. Já os NSB incorporam uma coleção de comportamentos que visa dar naturalidade à interação com o sistema artificial. Por exemplo, assentir enquanto o interlocutor fala (*nodding*), preencher o tempo com sons, palavras ou frases do tipo "deixe-me pensar..." (*filler insertion*), olhar para o interlocutor durante a interação (*face tracking*) e reagir a estímulos do ambiente, como um barulho alto (*reaction to environmental stimuli*).

É apresentado um *framework* denominado RWSI (*Real World Speech Interaction*), dividido em duas camadas. A camada superior é a *CSB layer*, e a inferior é a *NSB layer*. A interação entre essas duas camadas é importante para uma interação natural. A estrutura RWSI foi implementada com uso da arquitetura EGO (*Emotionally GrOunded Architecture*), arquitetura de controle baseada em comportamentos para o robô autônomo QRIO SDR-4XII, implementada pela equipe dos autores do artigo.

Uma abordagem para a construção de robôs sociais consiste explorar emoções. Sistemas com essa característica são especialmente aplicáveis em robôs que servem como companhia, assistentes domésticos, brinquedos ou educadores. Como exemplo, podem-se citar robôs projetados para comportarem-se como animais domésticos. Um robô que aparenta sentir e ter personalidade tende a despertar especial interesse, porque apresenta um maior nível de similaridade com o ser humano do que os robôs limitados às atividades essenciais para a concretização de seus objetivos. Emoções podem dar a sensação de que o robô é um ser vivo, dando a impressão de que se trata de uma companhia real (MALFAZ e SALICHS, 2004).

Um modelo replicável para o uso de emoções em robótica é a escala PAD (*Pleasure, Arousal, Dominance*), proposta por Mehrabian (1980), para classificar as emoções. Essa escala é caracterizada por usar um espaço tridimensional, com eixos correspondentes a *prazer*, *excitação* e *dominância*. *Prazer* refere-se ao quão agradável é a emoção, *excitação* é a medida da sua intensidade e *dominância* representa o quanto a emoção é capaz de controlar o comportamento. A variedade de emoções que pode ser gerada por esse espaço depende da quantidade de valores discretos disponibilizados para cada eixo.

Essa escala é utilizada na implementação realizada em Hollinger et al (2006), com os valores de cada eixo normalizados. As emoções geradas são *irritado*, *entediado*, *curioso*, *honrado*, *eufórico*, *faminto*, *tímido*, *amado*, *intrigado*, *sonolento*, *despreocupado* e *violento*. O sistema funciona com robô perambulando pelo ambiente, usando sonares e sensores refletivos de luz infravermelha. Uma câmera embarcada é usada para detectar faces. Ao encontrar uma

face, o robô para e interage por meio de síntese de voz e através de movimentos, baseando-se no seu estado emocional corrente e na cor da camiseta do observador, a qual pode influenciar o seu estado emocional. Validou-se o sistema com o robô atuando nos corredores de uma conferência.

Como se pode ver, a robótica social apresenta diferentes aplicações, constituindo um campo de pesquisa bastante amplo.

3. FUNDAMENTAÇÃO TEÓRICA

3.1. ROBÓTICA MÓVEL: CONCEITOS INTRODUTÓRIOS

Segundo Mataric (2007), pela noção atual, pode-se definir um robô como "um sistema autônomo, que existe no mundo físico, pode sentir o ambiente, e pode agir sobre ele para atingir alguns objetivos". Um robô é constituído por *corpo físico*, *sensores*, *atuadores* e *controladores*.

O *corpo físico* é a constituição física do robô, que existe em um ambiente real e é capaz de atuar sobre ele. Se o sistema existir somente em ambiente virtual, não apresenta a habilidade de modificar o ambiente. Nesse caso, trata-se de uma simulação, não de um robô real.

Sensores são os dispositivos que permitem ao robô obter informações acerca do ambiente físico em que se encontram. Equivalem aos órgãos dos sentidos definidos na biologia, como olhos, ouvidos ou nariz. O conjunto de características de um robô em um determinado tempo e espaço é denominado *estado*, podendo ser *interno* (relativo ao robô) ou *externo* (relativo ao mundo físico percebido pelo robô). O estado interno também pode referir-se a informações sobre o ambiente externo armazenadas sob alguma forma de *representação*.

Atuadores são os dispositivos usados pelos robôs a fim de agir sobre o mundo físico, possibilitando a locomoção do robô pelo espaço ou a manipulação de objetos. Exemplos de atuadores são motores, *displays* ou alto-falantes. De acordo com a natureza e com os objetivos de seus atuadores, os robôs podem ser classificados em *robôs móveis*, cujo objetivo principal é locomover-se de forma autônoma, ou *robôs manipuladores*, representados por braços robóticos que se movem em uma ou mais dimensões, denominadas *graus de liberdade* (DOF – *degrees of freedom*). Essa separação tem se tornado cada vez menos identificável, na medida em que a complexidade das novas aplicações de robótica têm combinado, cada vez mais, essas duas características.

Finalmente, os *controladores* consistem em *hardware* e *software* usados para dar autonomia ao robô, processando as informações sensoriais, analisando o estado interno no robô e realizando deliberações a fim de determinar as instruções a serem executadas pelos atuadores. A autonomia conferida ao robô pelos controladores pode ser total, quando o robô age levando em conta somente as suas próprias deliberações, ou parcial, quando parte do controle é realizado remotamente por um ser humano, o que se denomina *teleoperação*. Um sistema completamente teleoperado foge à definição moderna de robótica.

Embora a implementação intuitiva de conjuntos de regras de controle possa adequar-se a alguns controladores robóticos relativamente simples, dificilmente atenderá aos requisitos de sistemas mais complexos. Por esse motivo, um controlador pode ser organizado segundo algumas políticas, estabelecendo diretrizes e restrições. A essa organização, dá-se o nome de *arquitetura de controle*.

3.2. SENSORES E ATUADORES USADOS NO PROJETO

Apresentados os conceitos essenciais para o desenvolvimento da robótica, faz-se necessário, antes de abordar o controlador, apresentar os sensores e atuadores de interesse na elaboração deste projeto. Trata-se de sensores refletivos de luz infravermelha, usados para detecção de obstáculos, e de servo-motores, usados para locomoção do robô e posicionamento de algumas de suas partes mecânicas. Somente esses sensores e atuadores serão apresentados, embora uma grande variedade de sensores e atuadores possa ser encontrada na indústria e no mercado, com aplicações diversificadas dentro da academia. A forma como esses sensores e atuadores são arranjados no robô utilizado é descrita na seção 4.2.

3.2.1. SENSORES REFLETIVOS DE LUZ INFRAVERMELHA

Esses sensores são baseados na emissão de luz infravermelha, por meio de fotodiodos, e na detecção da quantidade de luz refletida, por meio de fototransistores. De acordo com a cor superficial do objeto, a parcela da luz emitida que será absorvida pelo objeto poderá variar. No entanto, em alguma medida, pode-se assumir que todos os objetos detectáveis refletirão uma quantidade de luz suficiente para que sejam percebidos. O uso de sensores trabalhando com diferentes cores, não somente infravermelho, possibilitaria o reconhecimento da cor do objeto (SOCIETY OF ROBOTS, 2013), o que não é necessário para detecção de obstáculos.

3.2.2. SERVO-MOTORES

Um servo-motor pode ser definido como um sistema mecatrônico que realiza movimentos proporcionais a sinais elétricos de comando. A ideia é que um sinal elétrico de estímulo desequilibra o sistema, que, por sua vez, reage na busca de um novo equilíbrio. Essa reação se dá na forma de um deslocamento mecânico angular proporcional ao sinal de estímulo (SOCIETY OF ROBOTS, 2013).

O motor de corrente contínua usado no servo-motor tem um sistema de engrenagens de redução acoplado a seu eixo. No eixo de saída, é fixado um potenciômetro. A variação da

resistência do potenciômetro provocada pelo giro deste eixo é usada, em conjunto com um capacitor, em um circuito de tempo. O pulso gerado por esse circuito é comparado ao pulso de entrada, que deve variar de 0,5 ms a 2,5 ms. O circuito de controle busca anular a diferença entre estes pulsos, promovendo um movimento de rotação no motor. Os valores extremos, 0,5 ms e 2,5 ms, levam o eixo de saída a girar de um extremo a outro do potenciômetro (em torno de 210°). O valor médio, 1,5 ms, determina o giro à posição intermediária do potenciômetro. A relação entre a largura do pulso de entrada L e o sentido do deslocamento angular resultante é ilustrada na Figura 1 A recorrência das rampas de subida dos pulsos de estímulo, a cada 20 ms, independe do valor de L.

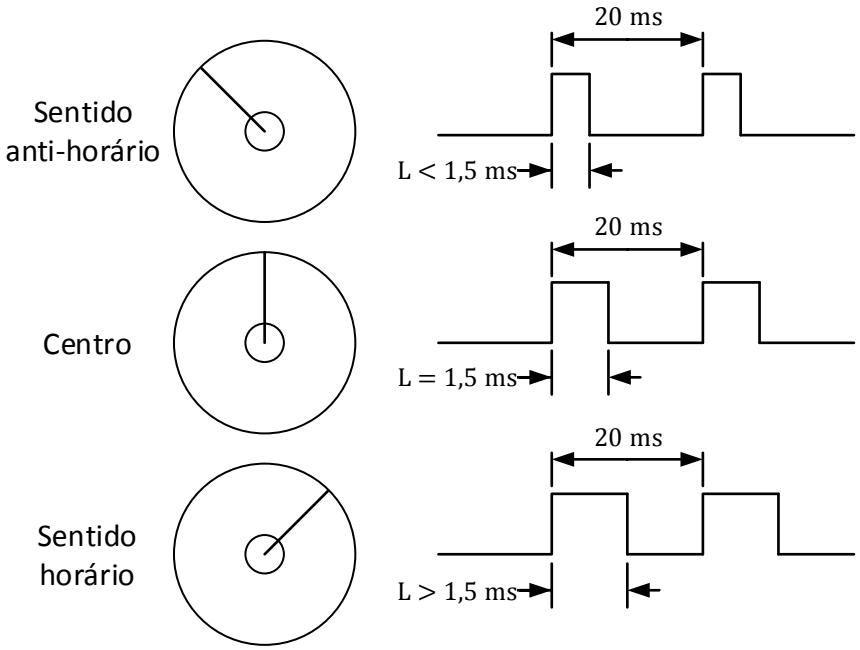

Figura 1 - Direção de deslocamento do eixo de um servo motor, de acordo com a largura do pulso recebido a cada 20 ms.

É possível, ainda, que servo-motores atuem como motores de rotação contínua. Um servo-motor dessa natureza é desprovido de realimentação pelo movimento do eixo do potenciômetro e de travas mecânicas que o limitem o movimento. O potenciômetro é ajustado para pulsos com $L = 1{,}5$ ms, de modo que pulsos de duração superior ou inferior a esse valor resultem em movimento de rotação contínua em sentido horário ou anti-horário, respectivamente. Quanto maior a discrepância entre o L e a referência de 1,5 ms, respeitando-se o limite de 0,5 ms a 2,5 ms, maior a frequência, em rotações por minuto, do movimento de rotação. Como consequência, se a largura do pulso for $L = 1{,}5$ ms, o motor permanece parado.

O motor gira em um sentido horário, para $L > 1{,}5$ ms, ou no sentido anti-horário, para $L < 1{,}5$ ms.

3.3. ARQUITETURAS DE CONTROLE

A seção 3.2, limitou-se a apresentar os sensores e atuadores de interesse para este projeto. No entanto, como o enfoque deste projeto é o controlador, especificamente o desenvolvimento de uma arquitetura de controle, cabe uma descrição mais detalhada dos diferentes tipos de arquiteturas de controle que compõem a literatura. Nesta seção, serão apresentados os conceitos mais importantes acerca do projeto de arquiteturas de controle, com foco na arquitetura de subsunção (BROOKS, 1986), tida como diretriz para a elaboração deste trabalho.

A fim de favorecer o desenvolvimento de sistemas de controle, uma arquitetura de controle é organizada em *módulos*. Um módulo pode ser classificado conforme a sua finalidade dentro do sistema, podendo agregar diversas unidades que levem-no a atingir seus objetivos. As diferentes formas segundo as quais esses módulos atuam sobre os dados, provenientes de sensores ou de outros módulos, podem ser usadas para caracterizar as arquiteturas de controle existentes, agrupando-as segundo determinados critérios.

Conforme postulado por Medeiros (1998), podem-se classificar as arquiteturas de controle presentes na literatura segundo três aspectos, referentes às diretrizes para projeto dos módulos. São eles a forma de interconexão entre os módulos, a natureza dos módulos e a forma de comunicação entre os módulos e o ambiente.

- **Forma de interconexão entre os módulos:** pode ser *hierárquica* ou *centralizada*. Em arquiteturas hierárquicas, cada módulo comunica-se com os módulos "vizinhos" que a ele estiverem conectados. Já em arquiteturas centralizadas, todos os módulos se comunicam com um módulo central, encarregado de interconectar todos os demais.
- **Natureza dos módulos:** módulos podem ser *funcionais* ou *comportamentais*. Em sistemas funcionais, os módulos são organizados segundo os serviços que provêm ao sistema, como aquisição de sinais e cálculo de rotas. Distintamente, os módulos de sistemas comportamentais são encarregados de gerar comportamentos observáveis, como seguir uma parede.
- **Relações entre os módulos e o ambiente:** é a característica usada como critério principal para subdividir as arquiteturas de controle existentes. Segundo esse critério,

sistemas robóticos podem ser baseados em três tipos de paradigma: *deliberativo*, *reativo* ou *híbrido*. Em sistemas deliberativos, as ações do robô são provenientes de uma deliberação prévia, visando resultados de longo prazo. Sistemas reativos respondem aos estímulos do ambiente de forma imediata e previsível. Arquiteturas híbridas contam com módulos reativos e deliberativos, intermediados por uma camada de arbitração.

3.3.1. TIPOS DE ARQUITETURAS DE CONTROLE

Segundo Murphy (2002), podem-se definir três paradigmas da robótica, ilustrados na Figura 2. Esses paradigmas estabelecem relações entre as primitivas *sentir*, *planejar* e *agir*. A primitiva *sentir* refere-se à forma como as informações sensoriais são coletadas por meio dos sensores. A primitiva *planejar* consiste nos algoritmos que determinam as ações que serão tomadas, a fim de se atingirem os objetivos do sistema. A primitiva *agir* consiste na ativação dos atuadores.

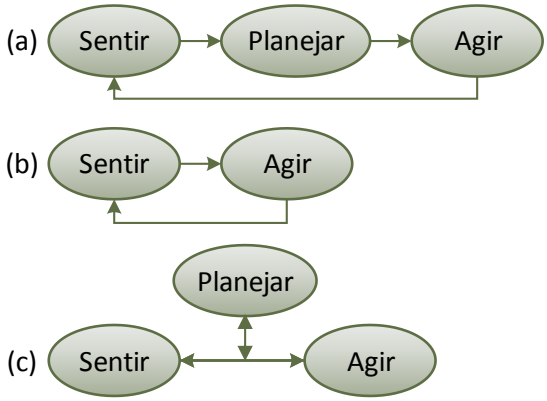

Figura 2 - Paradigmas da robótica, sendo eles: (a) paradigma deliberativo, conhecido como SPA (*sense, plan, act*); (b) paradigma reativo; (c) paradigma híbrido. (MURPHY, 2002)

Três paradigmas são apresentados: o *deliberativo*, em que as primitivas são executadas sequencialmente (SPA – *sense, plan, act*), o *reativo*, em que os estímulos são conectados a uma reação imediata, sem a etapa de planejamento, e o *híbrido*, em que as três primitivas podem ocorrer de forma dinâmica e em diferentes sequências. Em Mataric (2007), são introduzidas, além das arquiteturas deliberativas, reativas e híbridas, as *arquiteturas baseadas em comportamentos*. Essas arquiteturas, embora bastante semelhantes às arquiteturas reativas, apresentam algumas particularidades que as fazem merecer uma classificação específica. Os quatro tipos citados de arquiteturas de controle serão apresentados nos parágrafos que seguem.

Arquiteturas *deliberativas* são caracterizadas pelo uso de algoritmos de planejamento, que precedem todas as tomadas de decisão do robô. Esses algoritmos consistem em, dado um

conjunto de informações sensoriais e um modelo de mundo, buscar a sequência de ações que deverá levar ao objetivo do sistema. Em outras palavras, são previamente determinadas todas as ações do robô entre o estado atual, mensurado pelos sensores ou por estados internos, e o objetivo. Quanto maior o espaço de estados a serem considerados, em que um estado é definido como uma combinação específica de leituras de todos os sensores e um conjunto de informações na representação interna do sistema, maior o tempo demandado para a execução dos algoritmos.

Arquiteturas *reativas* não planejam, explicitamente, ações de longo prazo. A característica marcante desse tipo de arquitetura é a associação direta entre os sinais de entrada nos sensores (condições ou estímulos) e a saída nos atuadores (ações ou respostas). Dessa maneira, respostas rápidas a estímulos são utilizadas, em vez de planejamento de longo prazo.

Um projeto simples de sistema reativo consiste em conectar cada estímulo, individualmente, a uma resposta correspondente, conservando a exclusão mútua das condições. Como múltiplos sensores podem ser embarcados em um robô, seria preciso considerar separadamente cada combinação possível de entradas por parte dos sensores, o que, em grande parte das situações, é intratável. Uma alternativa é, durante o projeto do sistema, gerar ações específicas somente para as condições mais relevantes, introduzindo uma resposta padrão para as demais.

Quando as condições não são mutuamente exclusivas, o sistema fica sujeito a *colisões*, isto é, dois ou mais estímulos, ligados a respostas distintas para um mesmo atuador, são recebidos. Nesse caso, é necessário decidir entre ações distintas associadas a diferentes entradas simultâneas. Esse tratamento pode ocorrer por meio de *arbitração*, quando uma ação é selecionada dentre as demais, ou por meio de *fusão*, quando as diferentes ações candidatas são combinadas. O processo de arbitração pode ser dito competitivo, enquanto o processo de fusão pode ser considerado cooperativo. Outra característica importante relacionada a sistemas reativos é que, para assegurar que eventos não sejam perdidos e que a resposta seja suficientemente rápida, os sensores não devem ser monitorados de forma sequencial, mas sim paralela ou concorrente.

Arquiteturas *híbridas* são divididas em três camadas: uma camada deliberativa, uma camada reativa e uma camada intermediária responsável por unir e intermediar as relações entre as outras duas, denominada *camada de arbitração*. Esse modelo arquitetural permite tanto o planejamento de longo prazo, quanto reações rápidas a estímulos do ambiente, quando

necessário. Em algumas arquiteturas, a camada deliberativa mantém o controle durante a maior parte do tempo, recebendo notificações ou invocando a camada reativa convenientemente. Em outras, o contrário ocorre, isto é, a camada reativa prevalece.

Arquiteturas *baseadas em comportamentos* são caracterizadas por serem construídas como uma coleção de módulos comportamentais. A diferença entre arquiteturas reativas e arquiteturas baseadas em comportamentos reside no fato de comportamentos poderem constituir níveis de abstração mais altos do que as ações de módulos reativos, isto é, comportamentos são mais complexos que ações simples. Além disso, a entrada de um comportamento pode ser tanto a leitura de um sensor ou conjunto de sensores como a saída de outro comportamento. Arbitração e fusão também podem ocorrer em arquiteturas baseadas em comportamentos.

Comportamentos podem ser *observáveis* ou *não observáveis*. Comportamentos observáveis são responsáveis diretos pelos comandos executados pelos atuadores do robô. Comportamentos não observáveis não exercem controle diretamente sobre os atuadores, e suas saídas são usadas, em geral, como entradas para outros comportamentos. A representação interna pode ser realizada de forma distribuída, por redes de comportamentos não observáveis.

Vale ressaltar que os comportamentos constituintes de arquiteturas baseadas em comportamentos não devem ser confundidos com aqueles definidos pela psicologia comportamental, ou behaviorismo. Segundo essa vertente, todos os organismos podem ser definidos em função de seus comportamentos observáveis, dispensando o modelo de uma entidade responsável por realizar deliberações ou representações abstratas de conhecimento (BAUM, 2008). O fato de representações internas serem permitidas em arquiteturas baseadas em comportamento, portanto, torna-as incompatíveis com o behaviorismo.

3.3.2. ARQUITETURA DE SUBSUNÇÃO

Tradicionalmente, sistemas de controle para robôs móveis são decompostos em módulos funcionais, sendo cada unidade responsável por parte do processamento dentro de um fluxo de dados que parte da aquisição de dados pelos sensores até o módulo responsável pelo acionamento dos atuadores. A Figura 3 ilustra essa decomposição tradicional, denominada *decomposição vertical*, em um sistema hipotético.

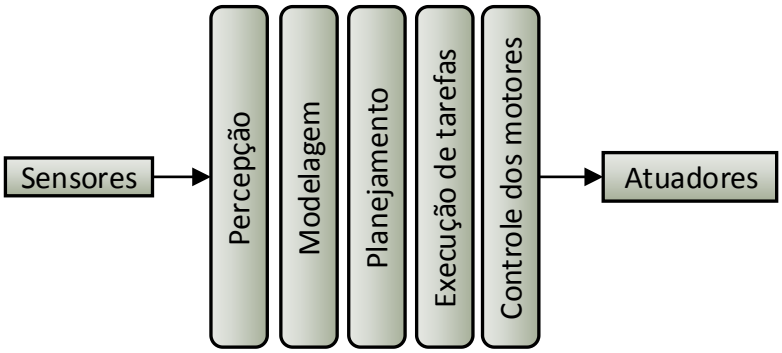

Figura 3 - Decomposição de um sistema de robótica móvel em módulos *funcionais* (BROOKS, 1986).

Como alternativa, foi criada uma arquitetura de controle baseada em comportamentos denominada *arquitetura de subsunção* (tradução livre para *subsumption architecture*), proposta por Rodney Brooks (BROOKS, 1986). Subsumir é considerar algo como parte de um todo maior. Por exemplo, na zoologia, situar uma classe dentro de um filo, e um filo dentro de um reino. No caso da arquitetura de subsunção, o sistema é organizado em módulos comportamentais dispostos hierarquicamente, denominados *níveis de competência*, cada qual compreendendo um subsistema de controle. Essa organização em níveis de competência também é conhecida como *decomposição horizontal*.

O projeto de um nível de competência deve ser norteado pelo padrão comportamental pretendido para o subsistema definido por ele, mais específico quanto mais elevado for o nível de competência em questão. Em outras palavras, um nível caracteriza-se por subsumir todos os níveis que estiverem acima dele. Como particularidade da arquitetura proposta por Brooks, um nível de competência pode suprimir as entradas ou inibir as saídas de comportamentos dos níveis inferiores. A Figura 4 consiste em um modelo da arquitetura de subsunção.

Usando essa arquitetura, o sistema pode ser desenvolvido de forma incremental, com os níveis mais altos sendo adicionados conforme os demais passem a funcionar corretamente. A comunicação entre os níveis deve ocorrer por um mecanismo de troca de mensagens. Em outras palavras, comportamentos de um nível não compartilham com outros níveis informações relativas ao seu estado interno.

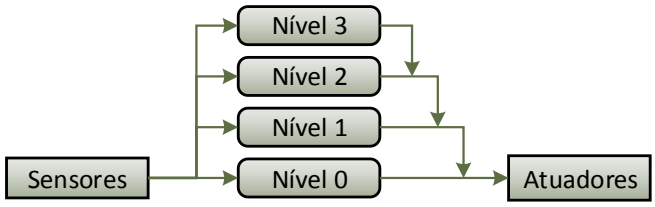

Figura 4 - Modelo da arquitetura de subsunção, com quatro níveis de competência (BROOKS, 1986).

Alguns trabalhos sugerem modificações na arquitetura proposta originalmente por Brooks. Em Rosenblatt e Payton (1989), por exemplo, são apontadas limitações nessa arquitetura. Na proposta de Brooks, impede-se que um "acordo" seja adotado entre diferentes níveis para que uma ação seja tomada pelo sistema, porque cada nível pode expressar somente a sua primeira opção. Em outras palavras, diminuem os recursos para que fusão ou arbitração ocorram da melhor forma possível. O resultado é que comportamentos são inibidos quando seria possível conciliar os interesses de comportamentos diferentes, contanto que alguns comportamentos abram mão da melhor saída possível em favor de alguma outra alternativa aceitável, ainda que não ótima. Como alternativa, é proposto um sistema granular, em que os comportamentos são compreendidos por pequenas unidades de tomada de decisão, as quais recebem múltiplas entradas e computam uma única saída, que pode ser usada como entrada para outra unidade de tomada de decisão ou como saída para um atuador.

O sistema proposto é voltado para o estudo da interação do robô com o ser humano, em diferentes contextos. Nesse sentido, pode ser interessante que o sistema seja construído com base em uma coleção de comportamentos observáveis, desenvolvidos sequencialmente em ordem crescente de especificidade, como propõe a arquitetura de Brooks.

3.4. MODELO CINEMÁTICO PARA ROBÔS MÓVEIS COM RODAS

A arquitetura de controle é encarregada de modularizar o desenvolvimento do controlador, segundo diretrizes e restrições, a fim de que o comportamento gerado leve aos objetivos pretendidos. Para isso, é importante que se tenha conhecimento dos efeitos que o acionamento de um ou mais atuadores tende a ocasionar no espaço ao redor do robô. É oportuno, portanto, o uso de modelos que descrevam matematicamente, de acordo com as leis da física, os movimentos do robô e a interação entre ele e os objetos a seu redor. O modelo adequado a uma aplicação é mais complexo quanto maiores forem as irregularidades e incertezas acerca do ambiente físico em que o robô será inserido, bem como quanto maior for o nível de precisão requerido.

Considerando esse aspecto, os controladores encontrados na literatura dividem-se entre três abordagens: considerar somente o modelo *cinemático*, considerar somente o modelo *dinâmico* ou considerar, conjuntamente, os modelos cinemático e dinâmico (JÚNIOR e HEMERLY, 2003). O modelo cinemático trata o movimento o robô como função da sua

velocidade, enquanto o modelo dinâmico leva em conta o torque dos motores e as demais forças envolvidas no sistema (DEVON e BRETL, 2007). Neste trabalho, somente o modelo cinemático será considerado.

Os atuadores responsáveis pela locomoção do robô podem apresentar limitações em seus movimentos, responsáveis por impedir que o robô, dependendo da sua orientação no espaço, desloque-se em determinadas direções. Esse tipo de restrição classifica o sistema como *não-holonômico*. Não obstante, sistemas não-holonômicos podem atingir qualquer configuração dentro do espaço em que estão definidos, contanto que sejam *globalmente controláveis e alcançáveis* (GOUVÊA, 2011). O modelo de robô móvel ilustrado na Figura 5 é um sistema não-holonômico (MARTINS, 2010). Trata-se de um sistema de locomoção com tração diferencial composto por duas rodas. Conforme será apresentado adiante, o robô usado para a realização deste trabalho é composto por um sistema dessa natureza, embora existam diversas outros arranjos.

Na Figura 5, são definidos o sistema de coordenadas cartesianas, o ponto P de intersecção entre o eixo de simetria e o eixo que une as duas rodas, o centro de massa C ou *ponto de guiamento* do robô, o raio r das rodas, a distância R entre as rodas e o eixo de simetria e a distância d entre C e P. A roda passiva tem finalidade de proporcionar suporte mecânico para o robô, com efeitos desprezíveis na dinâmica do robô. Pode ser substituída por um apoio fixo, contanto que o atrito entre o apoio e a superfície seja desprezível (RANIERI, *et al.*, 2012). O *vetor de postura* é definido pela tripla (x_c, y_c, θ), onde x_c e y_c correspondem às coordenadas do ponto C, e θ é o ângulo entre o eixo de simetria e o eixo X do sistema de coordenadas.

O modelo cinemático é importante na medida em que possibilita definir os movimentos do robô móvel através de suas velocidades linear v e angular ω, referentes ao deslocamento de C. Embora o modelo para determinação da velocidade do robô seja relevante para o presente trabalho, não existe necessidade de trabalhar com um modelo para configuração e postura do robô, uma vez que a navegação é assistida por humanos. Portanto, não serão apresentados os modelos que consideram a terna (x_c, y_c, θ).

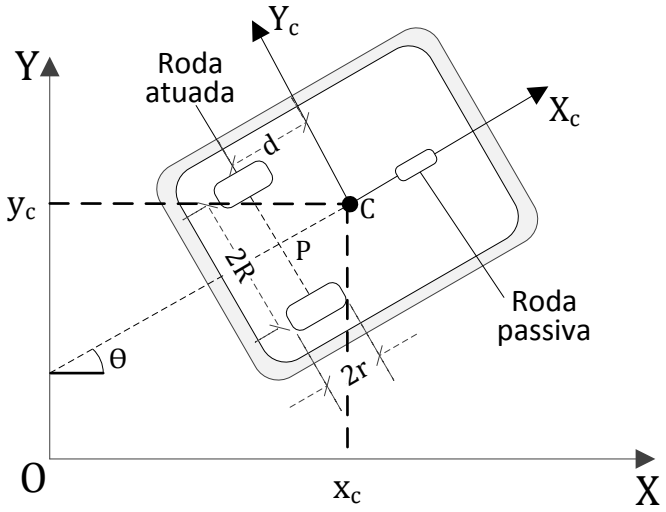

Figura 5 - Robô móvel com rodas e sistema de coordenadas cartesianas (MARTINS, 2010).

Considerando $d = 0$, as restrições cinemáticas impostas ao sistema são de que o robô não pode mover-se lateralmente e de que as rodas atuadas não podem girar em falso (MARTINS, 2010). Desenvolvendo as equações relativas a essas restrições, podem-se relacionar v e ω às velocidades angulares φ_d e φ_e das rodas direita e esquerda, respectivamente, conforme descrito pela Equação 1.

$$\begin{pmatrix} \varphi_d \\ \varphi_e \end{pmatrix} = \begin{pmatrix} \frac{1}{r} & \frac{R}{r} \\ \frac{1}{r} & -\frac{R}{r} \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \quad (1)$$

O uso de um sistema diferencial se estabelece pelo ambiente da aplicação, que é plano, normalmente sobre uma mesa ou no piso ao redor do usuário. Tanto o sistema sensorial como os atuadores (seção 4.2) são determinados, também, pela aplicação e pelo seu ambiente.

3.5. CONSIDERAÇÕES SOBRE O SISTEMA DE LOCOMOÇÃO

Nesta seção, será considerado o uso de servo-motores de rotação contínua em robôs móveis com tração diferencial compostos por duas rodas, conforme a Figura 5. A largura do pulso enviado como sinal de comando aos servo-motores, em milissegundos, é denominada *variável de controle*, podendo ser denotada por λ. Para $\lambda = 1{,}5$, em um servo-motor de rotação contínua, o eixo encontra-se em repouso, enquanto, em um servo-motor padrão, o eixo encontra-se na posição intermediária.

Para se controlar a velocidade do robô, faz-se necessário determinar como o comportamento do robô se modifica diante de alterações no valor escolhido para a variável de controle. No caso das rodas do robô, acopladas a servo-motores de rotação contínua, a velocidade que cada uma delas adquire, para diferentes valores da variável de controle, pode ser determinada por meio de experimentos.

Seja uma variável de controle $\lambda = 1{,}5 + x$, com $x \in [-1{,}0\ ,\ 1{,}0]$. Uma roda deve girar em uma direção, para $x > 0$, na direção contrária, para $x < 0$, ou permanecer em repouso, para $x = 0$. Supondo que os motores estão posicionados em direções opostas, para que as duas rodas girem com a mesma velocidade linear e em sentidos opostos, gerando um movimento em linha reta, seria desejável que, para qualquer valor de x atribuído a um dos dois motores, deva-se atribuir $-x$ ao outro motor. Como consequência, as duas rodas do robô apresentariam comportamento semelhante, e teriam suas velocidades angulares associadas a um mesmo λ. Com isso, impondo velocidades distintas para cada roda, seria possível obter diferentes combinações de velocidades lineares e angulares do robô, conforme a Equação 1 (seção 3.4).

Entretanto, devido a aspectos concernentes à mecânica ou à eletrônica do robô, a atribuição de $\lambda + x$ para um motor e de $\lambda - x$ para o outro não gera um movimento em linha reta, mas sim um movimento circular, situação ilustrada pela Figura 6 (a). Para lidar com essa questão, pode-se determinar, por meio de experimentos, uma constante k. Trata-se de uma aproximação para compensar essa diferença. Sempre que se pretender modificar a velocidade do robô, em um dos motores, $1{,}5 \pm x$ deve ser multiplicado por k, gerando a abstração de que um mesmo valor de λ gera uma mesma velocidade angular, para as duas rodas. Aplicando a constante k, portanto, caso se pretenda realizar um deslocamento em linha reta, o sistema pode comportar-se, aproximadamente, conforme o esperado, o que é ilustrado na Figura 6 (b).

Na seção 7.1, são apresentados os experimentos e os resultados que determinam k, a função que associa variáveis de controle às velocidades de cada roda e a função que associa variáveis de controle ao ângulo α de inclinação do *smartphone*. Esses resultados foram usados para o desenvolvimento da API para controle do robô (seção 6.1), a qual oferece, entre outros, funções para controle da velocidade do robô.

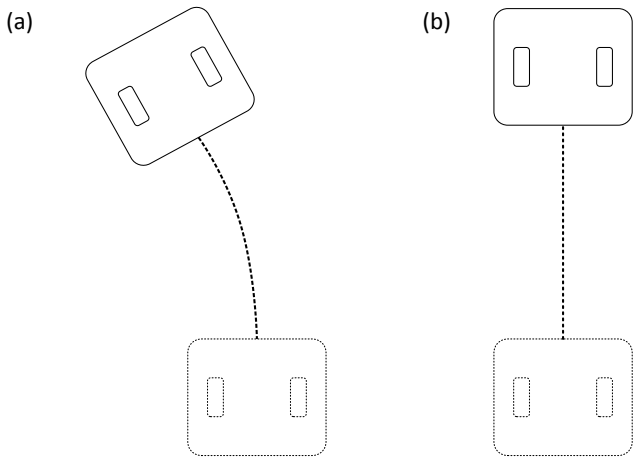

Figura 6 - Trajetória do robô: (a) com $\lambda = 1{,}5 + x$ no motor direito e $\lambda = 1{,}5 - x$ no motor esquerdo; (b) com $\lambda = 1{,}5 + x$ no motor direito e $\lambda = (1{,}5 - x) \times k$ no motor esquerdo.

3.6. O SISTEMA OPERACIONAL ANDROID

Além de considerar a arquitetura de controle e os modelos matemáticos que descrevem o comportamento do robô dentro do ambiente físico, é necessário levar em consideração a forma como o sistema será implementado. No caso deste trabalho, a arquitetura de controle é implementada em *software*, valendo-se de um *smartphone* para realizar o controle.

O interesse no uso de um *smartphone* neste projeto vem da possibilidade de explorar as diversas interfaces e a capacidade de processamento oferecidos por ele. Além disso, *smartphones* são dispositivos amplamente disponíveis comercialmente e de dimensões relativamente pequenas, permitindo serem embarcados até mesmo em robôs de pequeno porte. Incorporar a um robô uma coleção dispositivos que, individualmente, ofereçam cada uma das interfaces oferecidas por um *smartphone* pode ser trabalhoso e relativamente caro, além de demandar tempo de projeto e desenvolvimento.

Dentre as principais famílias de sistemas operacionais para *smartphones*, podem-se destacar o Android, do Google, o iOS, da Apple, o Windows Phone, da Microsoft, e o BlackBerry (LLAMAS, REITH e SHIRER, 2013). O sistema operacional Android foi desenvolvido pela Android Inc., empresa adquirida pelo Google em 2005. Os primeiros *smartphones* com Android foram lançados em outubro de 2008 (DEITEL, *et al.*, 2013).

Trata-se de uma plataforma aberta e gratuita. O SDK (*Software Development Kit*) oficial é direcionado para o desenvolvimento na linguagem de programação Java. Alguns fatores encorajam o desenvolvimento de aplicativos para plataforma Android. O fator mais evidente é a predominância desse sistema operacional nos dispositivos móveis. O Android foi responsável

por 79,2% do mercado de sistemas operacionais para *smartphones* no segundo trimestre de 2013, seguido pelo iOS, com fatia correspondente a 13,2% (LLAMAS, REITH e SHIRER, 2013).

Em termos de programação, um aplicativo para Android é dividido em unidades denominadas *activities*. Cada *activity* caracteriza-se por uma tela do aplicativo, tendo um *contexto* a ela atribuído. Uma *activity* pode invocar métodos que iniciam outra *activity* ou algum serviço disponibilizado no *smartphone*, retornando ou não um resultado quando estes terminam de executar. No que concerne à implementação, as telas de um aplicativo são compreendidas por uma classe descendente de *Activity*, responsável por responder a eventos, e por um documento XML (*eXtended Markup Language*) a ela associado, responsável por indicar os componentes, o *layout* e as referências para alguns eventos da *activity*.

O aplicativo também deve levar em consideração os eventos que fazem parte do *ciclo de vida da activity*, ilustrado na máquina de estados da Figura 7. Os eventos associados a cada método representado pelas transições dessa máquina de estados são descritos em seguida.

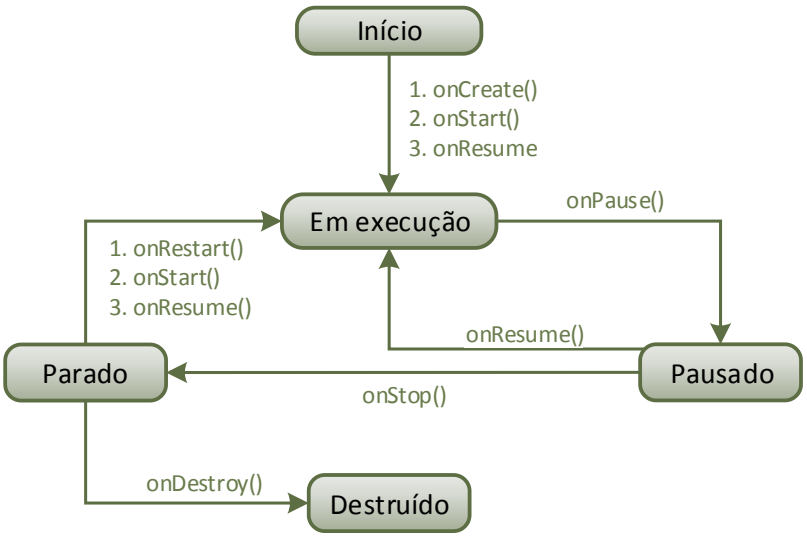

Figura 7 - Ciclo de vida de uma *activity*.

- ***onCreate():*** invocado quando a *activity* é criada pela primeira vez.
- ***onRestart():*** invocado após a *activity* ter parado, antes de ser iniciada novamente.
- ***onStart():*** invocado quando a *activity* está se tornando visível para o usuário.
- ***onResume():*** invocado quando a *activity* vai começar a interagir com o usuário.
- ***onPause():*** invocado quando o sistema vai continuar a execução uma *activity* anterior.

- ***onStop():*** invocado quando a *activity* não está mais visível para o usuário, porque outra *activity* continuou a executar e encobriu o seu lugar.
- ***onDestroy():*** a última chamada antes de a *activity* ser destruída.

3.6.1. ALGUNS SERVIÇOS E BIBLIOTECAS DE *SOFTWARE* PARA ANDROID

A fim de se desenvolver este trabalho, foram incluídos alguns serviços e bibliotecas para uso, em alto nível de abstração, das interfaces com o ser humano disponibilizadas pelo *smartphone*. Esses serviços e essas bibliotecas, cujo uso no projeto será detalhado na seção 6.2, foram incorporadas à *activity* do aplicativo responsável por iniciar e fornecer recursos para a arquitetura de controle. A forma como essa *activity* está inserida dentro do aplicativo é apresentada na seção 6.4.

Para visão computacional, foi incorporado o *OpenCv4Android*. Essa biblioteca, desenvolvida em linguagem C++, oferece uma grande variedade de métodos para processamento e análise de imagens, tendo sido exaustivamente utilizada em diferentes projetos que demandam recursos de visão computacional. Por esse motivo, também é oferecida uma interface para Android, que faz uso de chamadas em função em linguagem Java e tratando as particularidades do ambiente Android. Para usar a biblioteca *OpenCv4Android*, é possível realizar a ligação dinâmica dos métodos, instalando o aplicativo *OpenCv Manager*, disponível na loja de aplicativos oficial do Google, ou ligar a biblioteca estaticamente, no momento da compilação do aplicativo que faz uso dela.

O *OpenCv4Android* captura quadros usando as câmeras traseira ou frontal (se disponível) do *smartphone*, exibe esses quadros no *display* e executa um método, definido pelo programador, toda vez que um quadro é capturado. Para definir esse método, o programador deve implementar a interface *CvCameraViewListener2*.

Além de uma biblioteca para visão computacional, serviços para síntese e reconhecimento de fala são de interesse para este trabalho. Nesse sentido, dois serviços serão apresentados a seguir: o *eSpeakTTS* e o serviço para reconhecimento de fala oferecido pelo Google.

Embora existam classes com métodos para realização de síntese de fala na programação para Android, demanda-se que um serviço para essa finalidade, a ser chamado quando se invocar tal método, esteja instalado no *smartphone*. Neste projeto, foi usado o *eSpeakTTS*,

serviço para síntese de fala gratuito em português, disponibilizado na loja de aplicativos oficial do Google.

Reconhecimento de fala pode ser Trata-se de uma aproximação para compensar essa diferença.realizado pelo sistema do Google, incorporado nativamente à API de desenvolvimento para Android. O áudio é capturado e enviado para o serviço de reconhecimento. Para *smartphones* com versão do Android inferior à versão 4.1, conhecida como *Jelly Bean*, o bom funcionamento desse serviço demanda que esteja estabelecida uma conexão estável com a Internet. Isso ocorre porque serviço para reconhecimento de fala disponibilizado pelo Google, para essas versões, consiste em processamento remoto, na nuvem. Em *smartphones* com Android 4.1 ou superior, é possível que o reconhecimento de fala seja realizado *off-line*.

4. MATERIAIS

4.1. O SMARTPHONE

Neste projeto, para realização de testes de implementação, foi usado um *smartphone* Samsung Galaxy Y, com Android 2.3. Esse *smartphone* é equipado com todos os recursos imprescindíveis para que o projeto pudesse ser concretizado: tela de 3 polegadas sensível ao toque, câmera traseira com resolução de até 2MP para fotografias, ou 640x480 pixels para captura de vídeo, conexões Bluetooth e Wi-Fi 802.11 b/g/n, microfone e alto-falantes. É usado um processador *single-core* ARMv6, de 832 MHz. A implementação foi projetada para funcionar em qualquer *smartphone* com sistema operacional Android 2.2 ou superior, contendo esse conjunto mínimo de recursos.

4.2. O ROBÔ ROBURGUER

Uma fotografia do robô Roburguer, construído pelo GISDI, é mostrada na Figura 8. O *design* e a arquitetura de *hardware* do robô foram desenvolvidos como parte de projetos anteriores, voltados a assistir crianças portadoras de deficiências motoras severas (FERASOLI-FILHO, *et al.*, 2012). As diretrizes para a concepção do Roburguer vêm da necessidade de o robô apresentar-se de forma amigável, ao mesmo tempo que se desperta curiosidade acerca do seu funcionamento. Essa finalidade justifica a construção do robô baseada em placas de acrílico, que conservam a visão de suas partes internas. Essa construção apresenta custo de fabricação total aproximado de R$ 350,00. O tamanho e a mobilidade proporcionados pelo sistema mecânico, apresentado a seguir, permitem que o trabalho de desenvolvimento possa ocorrer sobre uma mesa de trabalho. Uma vez recarregado, o robô pode funcionar entre 30 minutos e uma hora, dependendo da atividade realizada.

O robô é dotado de um sistema de locomoção com tração diferencial, composto por dois motores de rotação contínua. É importante levar em conta que esses motores encontram-se posicionados em direções opostas. Com finalidade de prover estabilidade ao deslocamento do robô, há um apoio na região traseira, em um ponto equidistante dos dois pontos de contato entre cada uma das rodas e a superfície de apoio do robô, porém afastado do eixo que os une. No exemplar utilizado, os valores atribuídos às constantes r e R, apresentadas na Figura 5 (seção 3.4), são $R = 10{,}5$ cm e $r = 3{,}5$ cm.

Figura 8 - Robô Roburguer, com *smartphone* embarcado.

Sobre o robô, é colocado um *smartphone*, cuja orientação (*tilt*) pode ser ajustada por um servo-motor. Essa mudança de orientação, que somente pode ocorrer em uma única dimensão, permite a mudança de visão de objetos, pessoas ou caminhos, o que pode ser usado para elaboração de interfaces humano-robô ou navegação baseada em visão computacional. O eixo em torno do qual ocorre a rotação que resulta nessa mudança de orientação é paralelo ao eixo que une as duas rodas, conforme pode ser visualizado na Figura 8. O ângulo de inclinação resultante é denominado α. Definiu-se que $\alpha = 0°$ se o plano paralelo ao plano da tela do *smartphone*, contido no suporte, é perpendicular à superfície sobre a qual o *smartphone* está apoiado. A Figura 9 ilustra essa inclinação, mostrando o ângulo α.

Três sensores refletivos de luz infravermelha são colocados na dianteira do robô, alinhados entre si e paralelos ao plano sobre o qual o robô está apoiado, podendo ser utilizados para detecção de obstáculos.

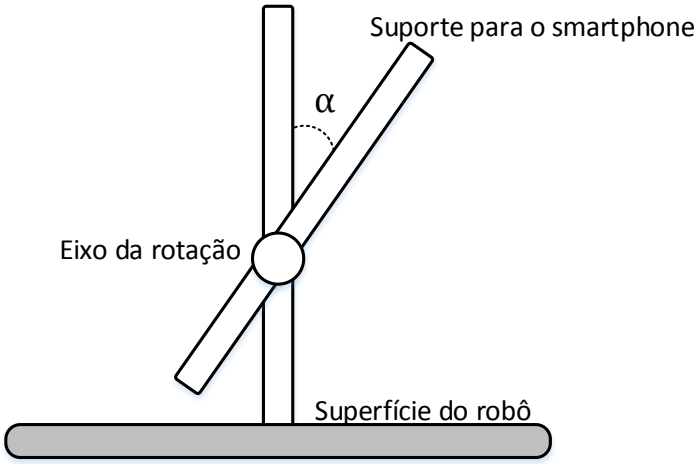

Figura 9 - Inclinação do *smartphone*, em função do ângulo α.

A conexão com o *smartphone* é realizada por meio de comunicação sem fio Bluetooth. Portanto, para que o sistema esteja funcional, basta que o *smartphone* seja posicionado no suporte do robô e que o aplicativo responsável pelo sistema seja iniciado.

4.2.1. ARQUITETURA DE *HARDWARE*

A Figura 10 ilustra a arquitetura de *hardware* do robô Roburguer. Um módulo de comunicação Bluetooth está ligado ao microcontrolador através do porto serial, ou RS-232. O *hardware* do robô é administrado por um sistema supervisor embarcado em um microcontrolador PIC-18F4550. O supervisor limita-se a receber comandos, por meio do porto serial, podendo responder enviando dados de leituras provenientes de sensores ou acionando atuadores. Como o robô não tem nenhuma funcionalidade autônoma embarcada no *firmware*, todo o controle deve ser realizado por um dispositivo externo, que pode ser um computador ou um *smartphone*, por exemplo. Portanto, uma vez conectado, o robô troca mensagens com o dispositivo externo, sendo controlado remotamente.

Os três sensores refletivos de luz infravermelha são conectados às entradas analógicas do microcontrolador. O valor lido é proporcional à intensidade da luz refletida que foi absorvida pelo fototransistor. As duas rodas do robô são atuadas por servo-motores de rotação contínua, enquanto o mecanismo que altera o ângulo α está conectado a um servo-motor padrão. Deve-se atentar ao fato de que os dois motores de rotação contínua que atuam sobre as rodas do robô Roburguer estão posicionados em sentidos opostos.

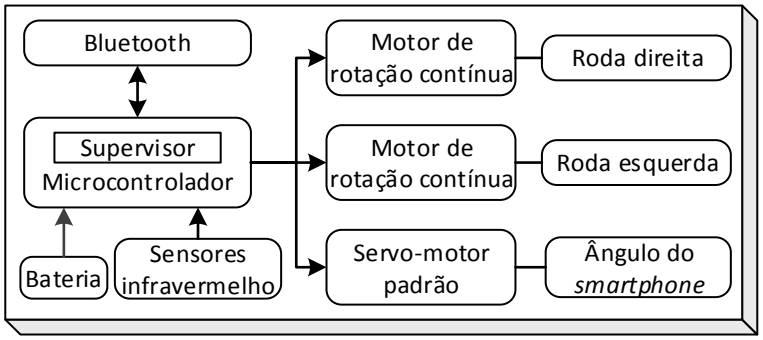

Figura 10 - Arquitetura de *hardware* do robô Roburguer.

5. VISÃO GLOBAL DO SISTEMA

Neste capítulo, será apresentado o ambiente desenvolvido, tanto no contexto da arquitetura de software, como da forma como pode ser visto pelo usuário, isto é, sem considerar a sua implementação. O ambiente é compreendido por um aplicativo executado em um *smartphone* embarcado no robô móvel Roburguer, gerando um ambiente interativo de robótica móvel com humanos.

Um ou mais usuários podem se dispor próximos ao robô, que pode funcionar sobre uma mesa de trabalho. Um usuário deve dar início ao sistema, configurando alguns parâmetros na tela inicial no aplicativo instalado no *smartphone*, responsável tanto pela interação humano-robô quanto pelas funções básicas de controle do robô móvel. Entre os parâmetros configuráveis, estão o endereço MAC do módulo Bluetooth embarcado no robô e o valor de corte para que os sensores de proximidade infravermelho determinem que um obstáculo foi encontrado. Uma vez configurados esses parâmetros, um usuário pode pressionar um botão para iniciar o sistema, conforme detalhado na seção 6.4.

Feito isso, uma contagem regressiva, iniciando com o número 4, precede o início do sistema. Essa contagem regressiva foi implementada para que, nesse período, um usuário coloque o *smartphone* no suporte presente no robô Roburguer. O *smartphone* deve ser posicionado com o *display* voltado de encontro ao suporte, de modo que a câmera traseira fique virada para frente. Isso deve ser feito porque o Samsung Galaxy Y não dispõe de câmera frontal. Havendo câmera frontal, o que pode ser obtido com o substituição do *smartphone*, seria possível posicionar o *smartphone* com o *display* voltado para frente, possibilitando a comunicação visual do sistema com o ser humano por meio de imagens exibidas no *display*.

5.1. ARQUITETURA DE *SOFTWARE*

A Figura 11 mostra a arquitetura de *software* do sistema. A programação foi realizada em linguagem de programação Java, com uso SDK (*Software Development Tools*) oficial, representado pelo pacote *Android Development Tools* (ADT), incorporado ao ambiente de desenvolvimento Eclipse.

Os recursos oferecidos pelo *smartphone*, em nível de *hardware*, são apresentados na camada mais inferior da arquitetura. São eles a interface Bluetooth, responsável pela comunicação com o robô, a câmera, usada para detecção de face, o *display*, que exibe as imagens capturadas, o alto-falante, que emite a fala artificial gerada, o microfone, que captura o som ambiente para possibilitar o reconhecimento de fala, e o adaptador de rede Wi-Fi, que possibilita o reconhecimento de fala na nuvem, se necessário.

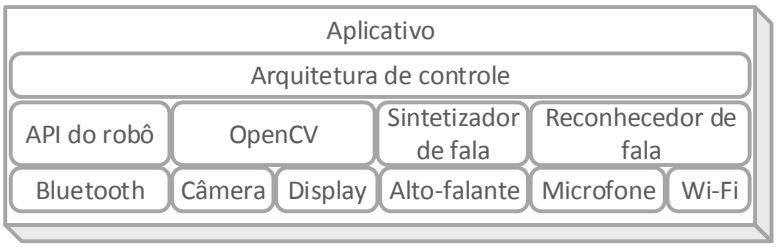

Figura 11 - Arquitetura de *software* do ambiente desenvolvido.

Na camada imediatamente superior, situam-se as bibliotecas que possibilitam usar esses recursos em nível mais alto de abstração. São elas a API para controle do robô, descrita na seção 6.1, e bibliotecas e serviços para interação com o ser humano, apresentados na seção 3.6.1. A forma como essas bibliotecas e esses serviços são disponibilizados para a arquitetura de controle é discutida na seção 6.2.

A terceira camada é a arquitetura de controle, cujos projeto e desenvolvimento são relatados com detalhe na seção 6.3. Essas três camadas são agregadas por um aplicativo, responsável por iniciar todo o ambiente.

Com exceção do reconhecimento de fala, que pode ser realizado na nuvem, todo o processamento ocorre dentro do *smartphone*. Esse arranjo proporciona mobilidade para o sistema robótico, que não depende de uma base computacional fixa para funcionar. Uma alternativa, que geraria um ambiente mais acoplado e com menos mobilidade, seria incluir uma base computacional remota, a qual se poderia conectar ao *smartphone* por meio de uma rede local, usando a interface Wi-Fi. Nesse caso, seria possível trabalhar com bases de dados

maiores, além de executar algoritmos que demandem maior capacidade de processamento. Isso seria realizado dentro da base computacional. Essa alternativa foi descartada, visto que, para o presente projeto, o poder computacional do *smartphone* é suficiente.

5.2. COMPORTAMENTO DO SISTEMA

O sistema iniciado apresenta o padrão de comportamento observável apresentado na máquina de estados da Figura 12. Durante toda a execução, a imagem capturada pela câmera é exibida no *display* do *smartphone*, podendo conter retângulos, de cor verde, delimitando as regiões em que faces são detectadas. Por se tratar de um aplicativo para Android, o sistema é interrompido se o usuário aciona, no *smartphone*, comandos que pausem a aplicação, como o botão *voltar*, a qualquer momento.

O sistema se inicia no estado *Explorar*, em que o robô se locomove pelo ambiente de forma aleatória, enquanto alterna periodicamente o ângulo de inclinação do *smartphone* (o ângulo α) em relação à superfície sobre a qual o robô está apoiado. Isso é feito a fim de aumentar a probabilidade de se detectar uma face humana por meio da câmera. Encontrado um obstáculo, ocorre transição para o estado *Retroceder movimento*, responsável por realizar, de acordo com quais dos sensores de proximidade detectaram obstáculos, um movimento de locomoção na direção contrária à localização frontal dos sensores de proximidade. Terminado esse retrocesso, é realizada a transição de volta para o estado *Explorar*.

Encontrada uma face, é realizada a transição para o estado *Requisitar comando*, em que o robô para de explorar e a interação através de comandos de voz se inicia. Por meio dessa interação, o ser humano pode controlar o robô, que obedece aos comandos, exceto quando detectar obstáculo. Para tornar a interação mais natural, o robô requisita comandos por meio de síntese de fala.

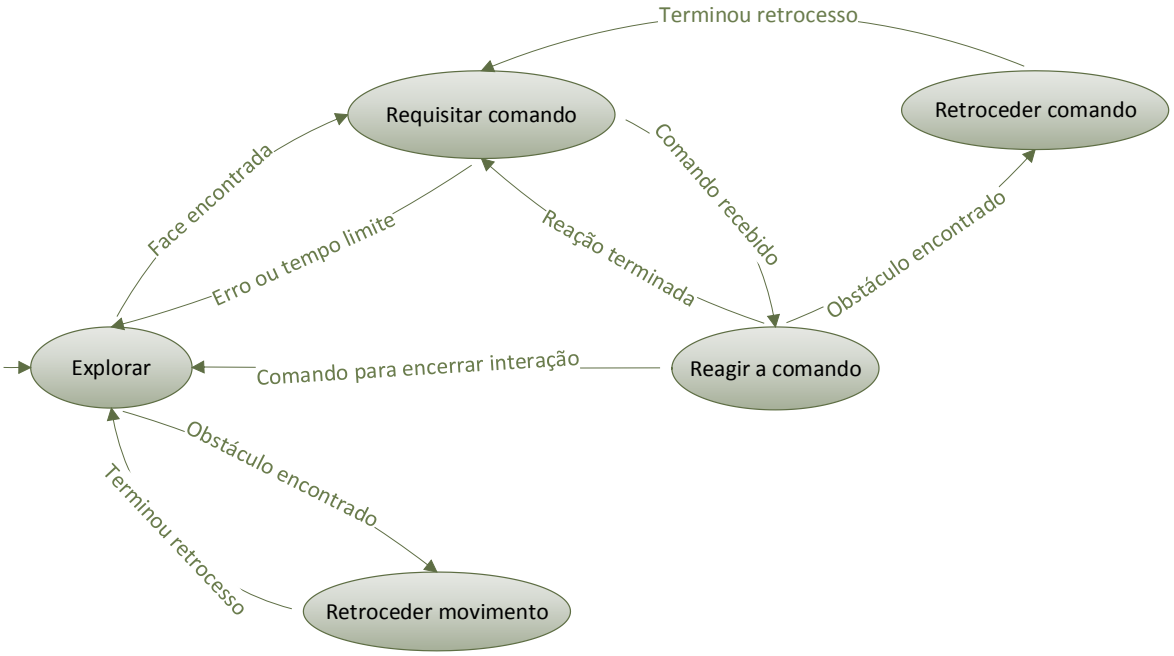

Figura 12 - Máquina de estados, iniciada no estado *Explorar*, representando o padrão comportamental observável do sistema.

Qualquer estímulo sonoro detectável pelo serviço de reconhecimento de fala pode ser considerado um comando, podendo ser *válido*, quando é associado a uma expressão em língua portuguesa identificada pelo serviço de reconhecimento e contida no conjunto de comandos previamente determinado para o robô, ou *inválido*, caso contrário. A Tabela 1 mostra os comandos disponíveis no conjunto definido, bem como os comportamentos a eles associados.

Tabela 1 - Comandos disponíveis na interação por voz e comportamentos a eles associados.

Comando	**Comportamento**
Pode ir	Encerrar interação por voz.
Círculo	Locomover-se em uma trajetória no formato de uma circunferência, de raio 10 cm, em torno do centro de massa do robô.
Virar para trás	Girar 180º em sentido horário.
Frente	Deslocar-se para frente, a 6,0 cm/s, durante 1,5 s.
Horário	Girar 30º em sentido horário.
Anti-horário	Girar 30º em sentido anti-horário.

Se o robô permanecer um período prolongado sem receber um comando, ou um erro ocorrer no serviço para reconhecimento de fala, o sistema volta para o estado *Explorar*. Ao

receber um comando inválido, o sistema usa o sintetizador de fala para dizer “não entendi”, caso o serviço de reconhecimento não associe o áudio capturado a uma expressão da língua portuguesa, ou “o que você quis dizer?”, caso essa associação ocorra, mas nenhuma expressão retornada pelo serviço de reconhecimento esteja contida no conjunto de comandos definido para o sistema. Nos dois casos, sugere-se que o usuário repita o comando ou requisite outro.

Assim que um comando válido é reconhecido, o robô repete o nome do comando, por meio do sintetizador de fala. Em seguida, ocorre transição para o estado *Reagir a comando*. Um comando específico é definido para encerrar a interação por fala, retornando o sistema para o estado *Explorar*. Não sendo esse o comando reconhecido, o comportamento associado a ele é executado. Caso esse comportamento possa ser executado até o fim, é realizada a transição de volta para o estado *Requisitar comando*, prosseguindo com a interação por voz. Detectando-se um obstáculo durante a execução do comportamento, ocorre transição para o estado *Retroceder comando*, que retrocede de forma semelhante ao estado *Retroceder movimento*, pedindo “cuidado”, por meio do sintetizador de fala, e voltando para o estado *Requisitar comando*.

6. DESENVOLVIMENTO DO PROJETO

A finalidade deste capítulo é apresentar o trabalho desenvolvido, em nível de projeto e implementação.

6.1. DESENVOLVIMENTO DA API PARA CONTROLE DO ROBÔ

A API para controle do robô tem por finalidade oferecer uma estrutura orientada a objetos que possibilite a programação em alto nível do robô Roburguer. É compreendida por três classes: *TBlue*, *Controller* e *Robot*, tendo sido as duas últimas implementadas como parte deste projeto. As três classes recebem como parâmetro, em seu construtor, o endereço físico do *chip* Bluetooth a que se pretende conectar. As relações entre essas três classes são ilustradas no diagrama de classes conceitual da Figura 13.

Figura 13 - Diagrama de classes conceitual da API para controle do robô.

A biblioteca aberta *tBlue*, apresentada por Karvinen (2001), trata algumas particularidades encontradas quando se pretende usar a tecnologia Bluetooth para realizar a comunicação entre um *smartphone* e um microcontrolador. Essa biblioteca consiste unicamente em uma classe, denominada *TBlue*, que oferece métodos para conexão, através do endereço físico do *chip* Bluetooth com que se pretende comunicar, e troca de mensagens. O uso dessa classe possibilita a comunicação em alto nível de abstração, o que a torna particularmente útil.

A classe *Controller* consiste em uma adaptação da classe *Controlador*, desenvolvida em projetos anteriores em linguagem C++ para ambiente Windows (ALVES, *et al.*, 2010). Na versão adaptada para linguagem Java e ambiente Android, implementada como parte deste trabalho, essa classe é responsável por usar uma instância de *TBlue* para controlar o robô móvel e obter dados deste. Em outras palavras, a classe *Controller* agrega a classe *TBlue*.

Essa classe dispõe de métodos para trocar mensagens com o robô, trazendo para o *smartphone* funções disponibilizadas pelo *firmware* embarcado no microcontrolador. Dentre essas funções, podem-se citar a obtenção das leitura de sensores refletivos de luz ou a modificação do valor para a variável de controle de um servo-motor. Para evitar que condições

de corrida comprometam o funcionamento correto do sistema, todos os métodos da classe *Controller* foram definidos como métodos do tipo *synchronized*.

Nenhum fluxo de controle faz parte dessa classe. Portanto, para que algum método seja invocado, é necessária a ação de outra classe. Embora este trabalho faça uso do robô Roburguer, a classe *Controller* pode ser utilizada para controlar diversos robôs desenvolvidos pelo GISDI, os quais executam uma versão semelhante do *firmware*. Isso se faz possível porque a classe *Controller* não faz nenhuma suposição acerca de qual a finalidade do servo-motor conectado a cada uma de suas saídas, nem de quais sensores estão conectados a suas entradas digitais ou analógicas.

A classe *Robot* descende de *Controller* e provê métodos mais específicos, designados para a versão utilizada do robô Roburguer. Nessa classe, são aplicados o coeficiente k, o modelo cinemático e a transformação de ângulos de inclinação α escolhidos para o *smartphone* em variáveis de controle para o motor responsável por essa inclinação. Todos os resultados provenientes dos experimentos relatados na seção 7.1 são introduzidos na classe *Robot*. Essa classe contém, ainda, um fluxo de controle responsável por invocar o método de *Controller* responsável por atualizar as leituras dos sensores refletivos de luz, armazenadas em um atributo protegido e disponibilizadas por meio da invocação de um método público da classe *Controller* e, consequentemente, da classe *Robot*.

6.2. PROJETO DE ROTINAS BÁSICAS PARA INTERAÇÃO HUMANO-ROBÔ

Nesta seção, serão apresentados os serviços e as bibliotecas usadas para aplicação em alto nível dos recursos disponibilizados pelo *smartphone*, mostrando a comunicação entre elas e a arquitetura de controle do sistema. Esses recursos foram disponibilizados dentro desta etapa do desenvolvimento, na *activity* responsável pela arquitetura de controle. Essa *activity*, denominada *HriActivity*, é apresentada detalhadamente na seção 6.4.

Na *HriActivity*, foi disponibilizado um método para verificar se uma face foi detectada dentro de até cinco segundos antes da sua invocação. Para isso, biblioteca *OpenCv4Android* é ligada dinamicamente ao aplicativo, com uso do *OpenCv Manager*. Usam-se quadros provenientes da câmera traseira do *smartphone*. Foi usado o classificador em cascata apresentado por Viola e Jones (2001) para realizar detecção de face. Aplica-se esse classificador a cada quadro, buscando por faces que correspondam a, no mínimo, 20% do tamanho total da

imagem. O quadro capturado é exibido no *display*, com retângulos verdes ao redor de cada face detectada com uso do classificador.

Métodos para síntese e reconhecimento de fala também foram incorporados à *HriActivity*. Para síntese de fala pelo alto-falante do *smartphone*, o *eSpeakTTS* foi instalado no *smartphone*. O método disponibilizado é bloqueante, somente continuando a execução do fluxo de controle que o invocou depois de a fala sintetizada terminar de ser emitida.

Já o reconhecimento de fala, com uso do microfone disponível no *smartphone*, foi realizado com uso do serviço disponibilizado pelo Google. No ambiente implementado, com uso do Android 2.3 (*Gingerbread*), o reconhecimento ocorre na nuvem. Dessa forma, a conexão com a Internet foi estabelecida por meio da interface de rede Wi-Fi.

A arquitetura de controle, implementada em um momento posterior e apresentada na seção 6.3, faz uso dessas interfaces, incorporadas à *HriActivity* junto a uma instância da classe *Robot*, apresentada na seção 6.1.

6.3. PROJETO E IMPLEMENTAÇÃO DA ARQUITETURA DE CONTROLE

Esta seção tem por objetivo mostrar a arquitetura de controle, tanto no que se refere aos módulos projetados para o seu funcionamento, quanto no que se refere a detalhes de sua implementação. O desenvolvimento de uma biblioteca para implementação de arquiteturas de controle nos moldes da arquitetura de subsunção precedeu o projeto e a implementação da arquitetura usada neste trabalho. A seção 6.3.1 é dedicada a apresentar essa biblioteca. Já a seção 6.3.2 tem por objetivo apresentar a arquitetura investigada e desenvolvida como parte deste projeto específico.

6.3.1. BIBLIOTECA PARA IMPLEMENTAÇÃO DA ARQUITETURA DE CONTROLE

Como dito anteriormente, a arquitetura de controle implementada segue os princípios da arquitetura de subsunção, proposta por Brooks (1986). Para tornar mais prática a implementação e para assegurar a natureza distribuída dos níveis de competência da arquitetura, foi desenvolvida uma biblioteca para desenvolvimento da arquitetura de controle, como parte inicial de seu projeto para implementação. Essa biblioteca consiste em uma classe base, denominada *CompetenceLevel*, a qual deve ser herdada para se implementar um nível de

competência, e em uma série de mecanismos a serem usados para relacionar os níveis de competência entre si e com as outras partes do sistema.

O construtor da classe *CompetenceLevel* recebe três parâmetros: uma instância de *Robot*, um número inteiro identificador do nível de competência e uma instância de *CompetenceLevel* referente ao nível de competência imediatamente inferior. Caso se trate do nível 0 da arquitetura, o terceiro parâmetro pode ser nulo. Para implementar uma arquitetura de controle, devem ser criadas instâncias das classes descendentes de *CompetenceLevel* associadas a cada nível de competência, do nível mais baixo para o mais alto, passando como parâmetro uma instância da classe relativa ao nível imediatamente inferior. Dessa forma, os níveis de competência ficam, em certa medida, isolados uns nos outros, assegurando a natureza distribuída da arquitetura.

O diagrama de classes conceitual da Figura 14 ilustra as heranças e associações entre as diferentes classes necessárias para a implementação de um sistema hipotético. Todos os níveis de competência devem ser agregados a uma classe responsável por iniciar o funcionamento da arquitetura de controle. Na Figura 14, trata-se da *MainClass*. Essa classe também agrega a classe *Robot*, sendo responsável por criar a instância de *Robot* e, portanto, realizar a conexão Bluetooth.

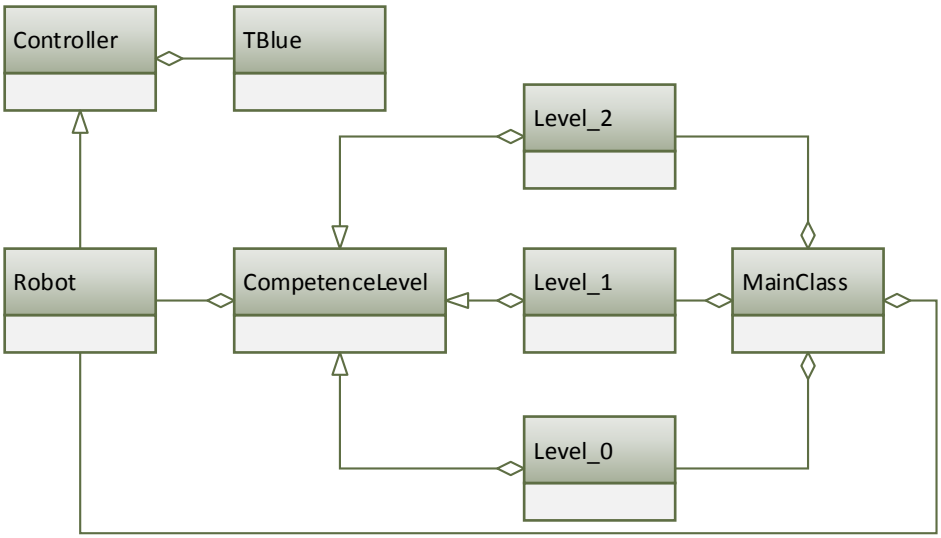

Figura 14 - Diagrama de classes conceitual da biblioteca para implementação da arquitetura de controle, para um sistema hipotético com três níveis de competência: *Level_0*, *Level_1* e *Level_2*. A classe *MainClass* é responsável por criar instâncias dos níveis de competência e iniciar o sistema.

Duas interfaces são disponibilizadas para comunicação entre os níveis de competência. A primeira delas é o método *inhibitCompetenceLevel*, que recebe, como parâmetro, o identificador do nível de competência que terá inibidas as saídas de todos os seus comportamentos. A segunda é o método *releaseCompetenceLevel*, que reverte o efeito do método *inhibitCompetenceLevel*. Um nível de competência somente pode ter acesso aos métodos *inhibitCompetenceLevel* e *releaseCompetenceLevel* de níveis inferiores, o que assegura que a hierarquia da arquitetura será respeitada.

A classe *CompetenceLevel* contém um fluxo de controle, responsável por enviar comandos para o robô, caso existam. Esse fluxo consiste em um laço de repetição, que executa a cada 100 milissegundos. Para verificar se uma saída foi gerada pelo nível em questão e emitir o comando para o robô, esse laço de repetição faz uso de uma variável inteira, denominada *inhibited*, que representa o número de inibições de que o nível em questão foi alvo. A inibição das saídas de um nível de competência ocorre, portanto, quando a variável *inhibited* passa a ter um valor diferente de zero. Nesse caso, o laço de repetição passa a atuar como uma espera ocupada, que verifica a variável de trava *inhibited* a cada 100 milissegundos.

Os algoritmos para implementação dos métodos *inhibitCompetenceLevel* e *releaseCompetenceLevel* são apresentados na Figura 15 e na Figura 16. Em ambas as representações, foram omitidas as invocações de métodos para tratar condições de corrida, limitando-se a mostrar a ideia por trás do algoritmo. O atributo *nível_referência* é o nível de competência cujas saídas se pretende inibir, *nível_corrente* é o nível de referência que está invocando o método e *nível_inferior* é o nível imediatamente inferior. O parâmetro *laço_saídas*, usado na espera ocupada, refere-se ao laço de repetição responsável pelas saídas do nível de competência.

```
Se (nível_referência == nível_corrente)
        inhibited = inhibited + 1
        espera_ocupada(laço_saídas)
Senão
        Se (nível_inferior == NULL)
                retornar
        Senão
                inhibitCompetenceLevel(nível_inferior)
```

Figura 15 - Algoritmo para inibição de um nível de competência.

```
Se ( (nível_referência == nível_corrente) E (inhibited > 0) )
        inhibited = inhibited - 1
Senão
        Se (nível_inferior == NULL)
                retornar
        Senão
                releaseCompetenceLevel(nível_inferior)
```

Figura 16 - Algoritmo para liberação de um nível de competência.

6.3.2. A ARQUITETURA DE CONTROLE DESENVOLVIDA

A biblioteca desenvolvida e apresentada na seção anterior permite a implementação em *software* de diferentes arquiteturas baseadas no modelo proposto por Brooks, independentemente da aplicação. A arquitetura projetada para este trabalho e sua implementação serão apresentados nesta seção.

O sistema desenvolvido baseia-se em quatro níveis de competência. Do mais baixo para o mais alto, são eles: *evitar obstáculos*, *explorar*, *buscar faces* e *interação por voz*. A Figura 17 apresenta os níveis de competência segundo a organização da arquitetura de subsunção.

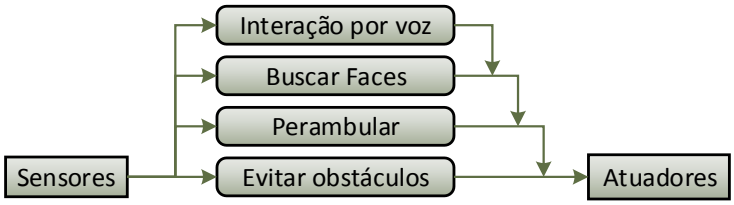

Figura 17 - Modelo da arquitetura de controle projetada.

No nível *evitar obstáculos*, os sensores de proximidade são verificados continuamente. Considera-se que um obstáculo foi detectado quando um ou mais sensores retornam um valor inferior ao valor de corte (*threshold*) escolhido. Enquanto se estiver detectando obstáculo, o robô retrocede. Embora os três sensores apontem para frente, é possível verificar, dependendo de quais sensores estejam detectando obstáculo, qual a melhor estratégia para retroceder ao mesmo tempo que se ajusta a postura do robô. Caso o sensor à direita esteja detectando um obstáculo e o sensor à esquerda não, por exemplo, pode-se dizer que o obstáculo em questão situa-se à direita do robô. Nesse caso, o robô retrocede alterando a sua postura no sentido anti-horário. O inverso ocorre quando um obstáculo é detectado pelo sensor à esquerda esteja detectando um obstáculo e o sensor à direita não. Nas demais situações, o robô retrocede em uma trajetória linear.

No nível *perambular*, o robô se desloca pelo ambiente com base em movimentos aleatórios. São realizados um deslocamento linear, na direção em que está posicionado, e uma rotação do robô em torno do seu próprio eixo, alterando a sua postura. Tanto a velocidade quanto a duração de ambos os movimentos são determinados de forma aleatória. No movimento de giro, determinou-se que a probabilidade de o robô girar no sentido horário é de 70%. Essa deliberação ocorreu tendo em vista que, se as probabilidades de o robô girar nos sentidos horário e anti-horário fossem iguais, o robô se deslocaria, em média, para frente, afastando-se indefinidamente do ponto de partida.

No nível *buscar faces*, o ângulo α é modificado periodicamente, por meio do acionamento do servo-motor padrão. O objetivo é aumentar a probabilidade de se encontrar uma face.

Finalmente, no nível *interação por voz*, o robô inicia, ao encontrar uma face, uma interação por voz, por meio da qual o usuário pode controlar o robô através de comandos. A interface humano-robô, baseada em comandos de voz, está incorporada a esse nível. Ao iniciar a interação, o nível *interação por voz* inibe as saídas dos dois níveis imediatamente inferiores (*perambular* e *buscar faces*), preservando o nível *evitar obstáculos*.

Todos os conflitos entre saídas de comportamentos do robô, inclusive respostas a comandos humanos, são resolvidos por meio de arbitração. Nesse contexto, cabe ao nível de competência mais elevado envolvido no impasse decidir se permite que os níveis inferiores resolvam a situação ou se inibe as saídas destes.

Na arquitetura desenvolvida, ocorre arbitração em três situações. A primeira situação é quando o robô, enquanto explora o ambiente, encontra um obstáculo. Nesse caso, o nível *perambular* abre mão, temporariamente, do controle dos motores de rotação contínua, permitindo que o nível *evitar obstáculos* atue. A segunda situação é quando o comportamento de interação por voz é iniciado. Nesse caso, o nível *interação por voz* impõe o estado de interação do robô com o ser humano, inibindo as saídas dos níveis *buscar faces* e *perambular*. A terceira situação é quando, ao executar um comando emitido por um usuário, o robô se depara com um obstáculo. Nesse caso, em que o robô opta por um comportamento próprio em detrimento de um comando recebido, a arquitetura opta por permitir que a camada *evitar obstáculos* atue, requisitando um novo comando em seguida.

6.4. ELABORAÇÃO DO APLICATIVO

O aplicativo responsável por configurar e iniciar o sistema recebeu o nome de *Robot Control*, tendo sido composto por duas *activities*: *MainActivity* e *HriActivity*.

A *MainActivity*, cujo *layout* é apresentado na Figura 18 (a), consiste na tela inicial do aplicativo, oferecendo a possibilidade de configurar dois parâmetros:

- **Valor de corte para proximidade:** é o *threshold* para determinar se os sensores refletivos de luz detectaram um obstáculo, medido como a variação de potencial entre os terminais do sensor, em volts. Como esse valor pode variar de acordo com a iluminação ambiente, é importante possibilitar que esse parâmetro possa ser modificado de forma prática.
- **Robô ID:** o sistema permite armazenar pares chave-valor de diferentes robôs, associando um nome identificador do robô e o endereço físico do *chip* Bluetooth nele embarcado. Para cadastrar um novo endereço físico, é necessário tocar o botão *Inserir endereço*, o qual abre a caixa de diálogo da Figura 18 (b). O botão *Remover endereço* elimina do registro o par chave-valor relativo ao robô selecionado. O botão *Iniciar sistema* invoca a *activity* responsável por iniciar a *HriActivity*, conforme a máquina de estados da Figura 12 (capítulo 5).

A *HriActivity* mostra, na tela, a imagem capturada pela câmera do *smartphone*. Se uma face estiver sendo detectada, é mostrado, também, um retângulo verde a seu redor. Essa *activity* incorpora todas as bibliotecas de terceiros usadas para o desenvolvimento. Por esse motivo, a classe *HriActivity* implementa as interfaces *CvCameraViewListener2*, referente ao *OpenCv4Android*, e *OnInitListener*, usada para criar uma instância da classe *TextToSpeech*, responsável pela chamada ao serviço disponível no *smartphone* para síntese de voz. A classe interna *HriRecognitionListener* implementa a interface *RecognitionListener*, importante para a chamada ao serviço para reconhecimento de fala.

Os níveis de competência que usam entradas e saídas dos métodos implementados para acesso a esses serviços recebem, como parâmetro de seu construtor, uma referência para a instância de *HriActivity*. Portanto as chamadas a funcionalidades disponibilizadas pelas bibliotecas mostradas na Figura 11 (seção 6.2) são realizadas de forma centralizada, com mecanismos que tratam a exclusão mútua no acesso aos recursos do *smartphone*.

- Figura 18 - *Layout* da *MainActivity*. (a) tela inicial, responsável pela configuração dos parâmetros; (b) caixa de diálogo, para cadastro de um novo par chave-valor que associe um robô a um endereço físico (MAC – *Media Address Control*) para comunicação Bluetooth.

É importante discutir, também, as implementações dos métodos invocados durante o ciclo de vida da *HriActivity*. No método *onStart()*, são instanciadas as classes importantes para funcionamento da arquitetura de controle e do aplicativo, bem como é realizada uma contagem de quatro segundos para que o sistema se inicie, dando tempo para o usuário posicionar o *smartphone* no suporte provido pelo robô. A conexão Bluetooth e o início do funcionamento da arquitetura de controle são realizados no método *onStart()*. As finalizações da conexão e do funcionamento da arquitetura de controle são realizadas no método *onStop()*. Os métodos *onPause* e *onResume* são usados para parar a atualização automática da imagem da câmera do *smartphone* enquanto uma caixa de diálogo ou o menu da aplicação executarem. Finalmente, o método *onDestroy* elimina a instância de *TextToSpeech*, além de executar sua função de eliminar todas as variáveis do objeto.

7. EXPERIMENTOS E RESULTADOS

Neste capítulo, os experimentos realizados como parte do projeto serão divididos em duas categorias: *testes preliminares*, realizados anteriormente ao desenvolvimento do trabalho (seção 6), e *testes para validação*, realizados após o desenvolvimento do projeto, a fim de apurar os resultados obtidos. Nas seções que seguem, todos esses experimentos serão apresentados, em conjunto com os seus resultados.

7.1. TESTES PRELIMINARES

Nesta seção, serão apresentados os testes anteriores à elaboração da API para controle do robô. Tanto λ quanto k, determinados em testes apresentados nesta seção, são definidos na seção 3.5. Três conjuntos de testes foram realizados nessa etapa: testes para determinar a constante k, testes para associar velocidades v_r das rodas do robô à variável λ e testes para associar inclinações α do suporte para *smartphone* à variável λ.

7.1.1. DETERMINANDO A CONSTANTE k

Sejam λ_e e λ_d as variáveis de controle para os motores das rodas esquerda e direita, respectivamente. Verificou-se que, para $\lambda_e = 1{,}5 + x$ e $\lambda_d = 1{,}5 - x$, para todo $x \neq 0$, tendo sido x definido na seção 3.5, o robô descreve uma trajetória circular em sentido anti-horário, com precisão de casa decimal dentro do intervalo $[-0{,}5\ , 0{,}5]$. Definiu-se, portanto, que seria oportuno o uso de uma constante k, a ser multiplicada por λ_e, contanto que $\lambda_e \neq 1{,}5$.

Realizaram-se dois conjuntos distintos de testes, a fim de determinar as constantes k_f, para $\lambda > 1{,}5$, e k_b, para $\lambda < 1{,}5$. Fazendo o robô andar durante dois segundos sobre uma cartolina dotada de um traço em linha reta, usado como referência, foram realizados testes com $x = \pm 0{,}1$, $x = \pm 0{,}2$ e $x = \pm 0{,}3$.

Em ambos os conjuntos de testes, o primeiro valor atribuído foi $k = 0{,}95$, o qual gerou um movimento no sentido horário. Esse valor, então, era acrescido de $0{,}01$ a cada iteração, com os três valores de x. Assim que, para um dado k_i (em que i é o número da iteração), a trajetória do robô ocorresse no sentido anti-horário para os três valores de x testados, seria selecionado o valor usado no teste imediatamente anterior, denotado por k_{i-1}.

Em seguida, realizou-se um refinamento, em que k_{i-1} era acrescido de $0{,}001$ a cada iteração, até que o movimento passasse ocorrer no sentido anti-horário para os três valores de

x, na iteração k_n. O valor k_{n-1} foi, finalmente, aplicado, resultando em k_f e k_b escolhidos. Como resultados dos testes realizados, obteve-se $\boldsymbol{k_f = 0,978}$ e $\boldsymbol{k_b = 0,988}$.

7.1.2. ASSOCIANDO v_r A λ

Introduzidas as constantes k_f e k_b, foram realizados testes para associar λ à velocidade v_r das rodas do robô, possibilitando a aplicação do modelo cinemático descrito na seção 3.4. O objetivo deste conjunto de testes foi conhecer o comportamento da velocidade adquirida pelas rodas do robô Roburguer, de acordo com a variável de controle.

Levando em conta resultados obtidos em projetos anteriores (RANIERI, 2012), com um robô dotado de arquitetura de *hardware* semelhante à do robô Roburguer, o robô 14-Bis, determinou-se que o maior valor a ser aplicado para a variável de controle dos motores de rotação contínua seria $\lambda = 1{,}75$. Esse valor foi escolhido porque, com $\lambda \in [1{,}25\ ,1{,}75]$, a velocidade adquirida pelo robô varia de forma diretamente proporcional com a variável de controle. Para além desses extremos, essa relação torna-se não linear.

Foram aplicados todos os valores $\lambda_e \in \{1{,}55\ ,1{,}60\ ,1{,}65\ ,1{,}70\ ,1{,}75\}$, com $\lambda_d = (3 - \lambda_e) \times k_f$. Com essa relação, as duas rodas giram no mesmo sentido e com velocidade semelhante, de modo que o robô se desloca para frente e em linha reta. Esse movimento era realizado sobre cartolina, colocada sobre uma mesa de trabalho plana. Mantinha-se o movimento por um período de três segundos. Esse período de três segundos foi determinado levando em conta as dimensões da mesa de trabalho, a fim de que o robô se mantivesse dentro de um espaço em que seu deslocamento pudesse ser mensurado.

Para medir o deslocamento, marcava-se, com uso de uma caneta, as posições inicial e final do apoio do robô, apresentado na seção 4.2. Repetiu-se esse experimento três vezes para cada valor escolhido para a variável de controle, e a média aritmética entre os três deslocamentos foi considerada. O gráfico da Figura 19 mostra os resultados obtidos, sugerindo que, dentro do domínio analisado, a relação entre a variável de controle e a velocidade adquirida é diretamente proporcional.

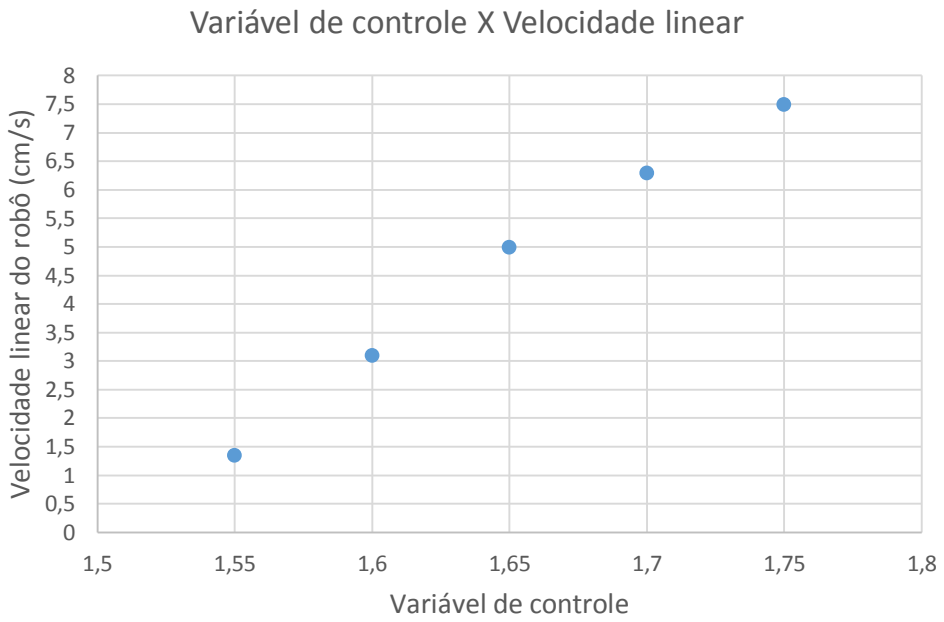

Figura 19 - Gráfico mostrando a velocidade linear adquirida pelo robô em função da variável de controle, conforme os experimentos realizados.

Esses resultados foram usados para determinar uma função que apresente a variável de controle λ como função da velocidade linear v_r de uma roda, isto é, dado um v_r desejado, obtenha-se o λ correspondente. Considerando que, como aproximação, trata-se trata de uma reta, basta usar dois pares (v_r, λ) para definir a equação da reta correspondente a essa função. Usando os pares $(1{,}35\ , 1{,}55)$ e $(6{,}3\ , 1{,}7)$, chega-se à Equação 2.

$$\lambda = 1{,}55 + 0{,}3 \times (v_r - 1{,}35) \qquad (2)$$

A partir da velocidade linear v_r de uma roda, pode-se obter a sua velocidade angular φ por meio da relação $\varphi = \frac{v_r}{r}$, em que r é o raio da roda. Com isso, é possível, por meio das velocidades linear v e angular ω desejadas para o centro de massa C do robô, supondo que C coincide com o ponto médio entre as duas rodas, usar a Equação 1 (seção 3.4) para determinar as velocidades lineares v_e e v_d das rodas esquerda e direita, respectivamente, que deverão gerar esse par (v, ω). A Equação 2 permite, então, associar v_e e v_d às variáveis de controle λ_e e λ_d a serem usadas como largura do pulso enviado ao servo-motor de rotação contínua.

7.1.3. ASSOCIANDO α A λ

Para determinar λ em função de α, possibilitando controlar o ângulo de inclinação do *smartphone* a partir de um valor desejável, em graus, determinou-se a equação de uma reta.

Para isso, determinaram-se três valores identificáveis para α, sendo eles $\alpha = 25°$, $\alpha = 90°$ e $\alpha = 180°$. Escolhendo valores sucessivos para λ, até que α resultasse em cada um dos valores determinados, três pares (α, λ) foram identificados. Dois desses valores foram usados para determinar a equação, enquanto o outro foi usado para verificação.

A Figura 20 mostra os três pontos identificados. O menor valor possível, devido a limitações mecânicas do robô, é $\alpha = 25°$, conforme a Figura 20 (a), o que faz com que esse ângulo seja facilmente identificável. A Figura 20 (b) e a Figura 20 (c) tratam de $\alpha = 90°$ e $\alpha = 180°$ respectivamente. Os pontos identificados foram $P = (90°, 1{,}4)$, $Q = (180°, 2{,}35)$ e $R = (25°, 0{,}7)$.

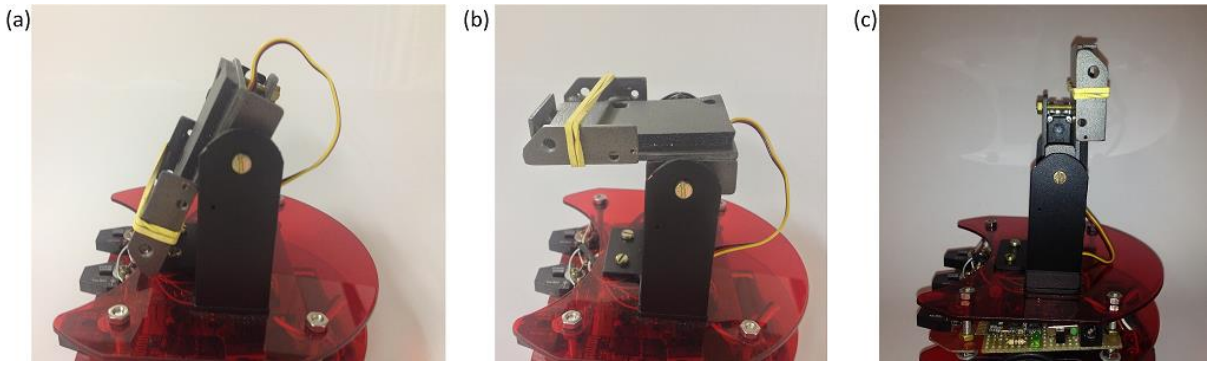

Figura 20 - Diferentes orientações do *smartphone*. (a) $\alpha = 25°$; (b) $\alpha = 90°$; (c) $\alpha = 180°$.

Usando-se os pontos P e Q, chegou-se à Equação 3. Atribuindo $\lambda = 0{,}7$ e $\alpha = 25°$, verifica-se que essa equação é verdadeira para R, com precisão de uma casa decimal.

$$\lambda = 0{,}45 + \frac{0{,}95}{90} \times \alpha \qquad (3)$$

7.2. TESTES PARA VALIDAÇÃO

A fim de validar o sistema desenvolvido, alguns ensaios foram realizados. Esses ensaios ocorreram dentro da UNESP, no Laboratório de Integração de Sistemas e Dispositivos Inteligentes (LISDI). Consistiram em experimentos individuais dos quatro subsistemas que compreendem a arquitetura de controle, a fim de verificar a funcionalidade de cada um.

Todos os testes foram realizados sobre uma plataforma plana de 60 cm x 80 cm, ao redor da qual existe uma borda plana de metal com altura de 5 cm, perpendicular à mesa. Essa borda pôde ser usada para impedir que o robô caísse da plataforma, ao mesmo tempo que serviu como obstáculo detectável pelos sensores de proximidade. A Figura 21 mostra a plataforma descrita.

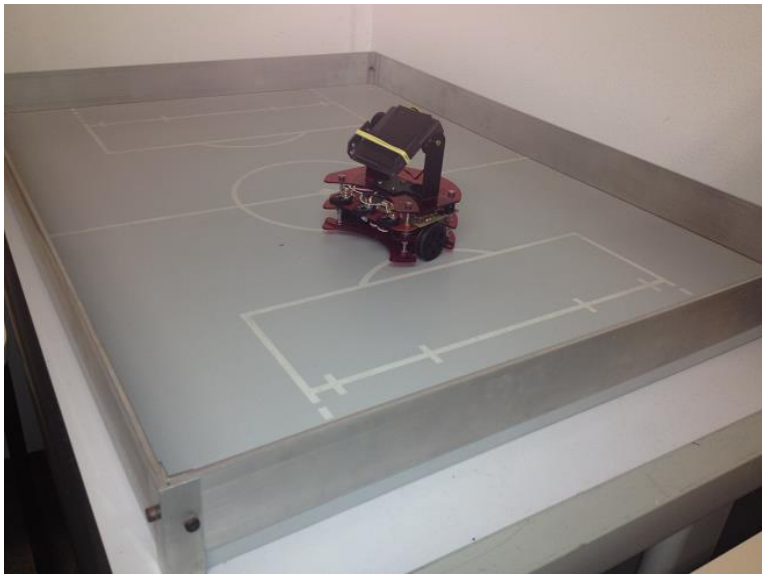

Figura 21 - Plataforma para experimentos com o robô.

Nas subseções seguintes, serão apresentados os experimentos realizados para validação de cada um dos quatro subsistemas de *software* que compõem o ambiente desenvolvido. Os resultados são apresentados por meio de comentários e de endereços de vídeos disponibilizados através da Internet, por meio do YouTube.

7.2.1. EVITAR OBSTÁCULOS

Este experimento pode ser visualizado em http://youtu.be/nMvTQEk-xOE. O robô foi posicionado sobre a plataforma, onde permaneceu imóvel até que um obstáculo fosse colocado próximo a um dos sensores de proximidade. Dessa forma, foi possível conduzir o robô a diferentes posições alcançáveis, bastando posicionar convenientemente o obstáculo próximo aos sensores de proximidade.

7.2.2. PERAMBULAR

No experimento apresentado em http://youtu.be/e4ZyLht-108, um obstáculo foi posicionado em meio à plataforma. O robô locomoveu-se em diferentes direções, alterando posição e postura constantemente. Nota-se que o robô consegue evitar as bordas da plataforma e voltar a posições mais centrais da mesa, fazendo uso tanto do retrocesso realizado pelo nível *evitar obstáculos* quando dos movimentos de rotação do nível *Perambular*. Quando o obstáculo se encontra em uma posição detectável pelos sensores de proximidade, o sistema reage rapidamente, retrocedendo antes de colidir, embora nem sempre o obstáculo na região frontal seja detectável por um dos sensores de proximidade antes que a colisão ocorra.

7.2.3. BUSCAR FACES

O experimento relativo ao nível *Buscar faces* pode ser visualizado no endereço http://youtu.be/zWgiM7XJOvc. O resultado é bastante próximo do que se obteve com subsistema definido pelo nível *Perambular*. Esse nível adiciona ao sistema alterações periódicas no ângulo α. No teste apresentado, verifica-se que esse comportamento não interfere naquele determinado pelos níveis inferiores.

7.2.4. INTERAÇÃO POR VOZ

Trata-se de um teste correspondente ao funcionamento do sistema completo. Um vídeo registrando esse experimento encontra-se disponível em http://youtu.be/dBLXeMwXi9M. Esse teste foi realizado em condições em que havia uma conexão estável com a Internet. Enquanto não há face para ser detectada, o sistema funciona somente com comportamentos das três camadas inferiores, conforme discutido na seção 6.3.2. Usando os sensores de proximidade, para o teste realizado, foi possível levar o robô a uma posição e a uma orientação em que se detectou a face do usuário. Quando isso ocorreu, todos os movimentos do robô pararam de ser realizados, e iniciou-se a interação por voz. Todos os comandos implementados foram verificados, tendo funcionado conforme o esperado. Durante a execução do comando “frente”, o usuário colocou a mão em frente aos sensores de proximidade, que a detectam como obstáculo. Quando isso ocorreu, o robô retrocedeu e, por meio da síntese de fala, requisitou um novo comando.

Nota-se que existe uma latência entre o término do comando pelo usuário e o início da emissão da resposta pelo *smartphone*. Isso é causado tanto pela política da biblioteca de reconhecimento de fala, para determinar que a sentença terminou de ser dita; quanto pelo tempo exigido para envio para o servidor remoto de reconhecimento de fala, pela Internet, do áudio capturado pelo microfone do *smartphone*, para a execução remota do algoritmo, no servidor remoto, e para o envio dos resultados do servidor remoto para o *smartphone*, também pela Internet.

Também existe atraso entre o posicionamento da face do usuário em frente à câmera do *smartphone* e o reconhecimento dessa face pelo sistema. Esse atraso decorre de limitações na capacidade de processamento do *smartphone*, que, com o sistema completo em execução, consegue processar somente cerca de um quadro por segundo. Os atrasos citados não comprometem, todavia, o correto funcionamento do sistema.

8. CONSIDERAÇÕES FINAIS

Este trabalho visa criar novas possibilidades de interação, devido ao uso de arquiteturas de controle baseadas em comportamentos receptivas a comandos humanos e suportadas pelo uso de *smartphones*. Espera-se que a plataforma desenvolvida possa ser adaptada e expandida, de modo a atender demandas reais, seja dentro da educação de crianças, seja como plataforma experimental para o estudo de interfaces humano-robô.

No que se refere a crianças, público-alvo dos projetos mais recentes desenvolvidos pelo GISDI, espera-se as que ações comportamentais dos robôs prendam a atenção das crianças, gerando a ideia de um pequeno animal de estimação, e, com aceitação comandos humanos, dando-se a ideia de um animal adestrado.

Neste trabalho, foi usado um *smartphone* Samsung Galaxy Y, o qual não dispõe de câmera frontal. Em trabalhos futuros, com uso de um *smartphone* com câmera frontal, seria possível introduzir diferentes expressões faciais, o que torna o ambiente ideal para experimentos envolvendo emoções. Diferentes sistemas podem ser avaliados para implementação dentro do ambiente desenvolvido.

A interação por voz, durante o atual estágio do desenvolvimento, limita-se a interpretar e executar comandos, fornecendo resposta ao usuário. Pretende-se, em trabalhos futuros, introduzir uma interação verbal mais rica entre o usuário e o robô, aproximando-se da ideia de um ser vivo artificial. Ainda nesse contexto, um uso mais incisivo de visão computacional também pode aumentar as possibilidades de interação do robô com humanos e com o ambiente externo, o que fomenta o interesse em aumentar o uso dessa tecnologia.

Com o rápido avanço na capacidade de processamento de *smartphones*, é possível prever que, em um futuro próximo, a capacidade de processamento desses dispositivos será compatível com a execução de algoritmos sofisticados. No caso de ser inviável que algum algoritmo seja executado pelo *smartphone*, a proposta é que seja usada uma base computacional remota, que se comunicaria com o *smartphone* através da rede sem fio Wi-Fi.

BIBLIOGRAFIA

ALVES, S. F. R.; RANIERI, C. M.; FERASOLI-FILHO, H.; CALDEIRA, M. A. C.; PEGORARO, R.; SILVA, I. N. **A Friendly Mobile Entertainment Robot for Disabled Children**. ISSNIP Biosignals and Biorobotics Conference (BRC). Rio de Janeiro: [s.n.]. 2013.

ALVES, S.; ROSÁRIO, J. M.; FILHO, H. F.; PEGORARO, R. **Environment for Teaching and Development of Mobile Robot Systems**. Electronics, Robotics and Automotive Mechanics Conference (CERMA). Morelos: [s.n.]. 2010. p. 302 - 307.

AOYAMA, K.; SHIMOMURA, H. **Real world speech interaction with a humanoid robot on a layered robot behavior control architecture**. Proceedings of the 2005 IEEE International Conference on Robotics and Automation. Barcelona: IEEE. 2005. p. 3814-3819.

ARBIB, M. A.; FELLOUS, J.-M. Emotions: from brain to robot. **TRENDS in Cognitive Sciences**, v. 8, n. 12, p. 554-561, Dezembro 2004.

BAUM, W. M. **Compreender o Behaviorismo**. [S.l.]: Artmed, 2008.

BREAZEAL, C.; SCASSELLATI, B. **How to build robots that make friends and influence people**. 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Kyongju: IEEE. 1999. p. 858-863.

BROOKS, R. A. A Robust Layered Control System for a Mobile Robot. **IEEE Journal of Robotics and Automation**, Março 1986. 14-23.

BUMBY, K. E.; DAUTENHAHN, K. **Investigating children's attitudes towards robots:** a case study. Proceedings of the Third Congtive Technology Conference. San Francisco: [s.n.]. 1999. p. 391-419.

DEITEL, P.; MORGANO, M.; DEITEL, H.; DEITEL, A. **Android para programadores:** uma abordagem baseada em aplicativos. Porto Alegre: Bookman, 2013.

DEVON, D.; BRETL, T. **Kinematic and Dynamic Control of a Wheeled Mobile Robot**. Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. San Diego: [s.n.]. 2007.

FERASOLI-FILHO, H.; CALDEIRA, M. A. C.; PEGORARO, R.; ALVES, S. F. D. R.; VALADÃO, C.; BASTOS-FILHO, T. F. **Use of Myoelectric Signals to Command Mobile**

Entertainment Robot by Disabled Children: Design and Control Architecture. Proceedings of the 3rd IEEE Biosignals and Biorobotics conference (ISSNIP). Manaus: [s.n.]. 2012.

GOUVÊA, J. A. **Controle de formação de robôs não-holonômicos com restrição de curvatura utilizando função potencial**. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2011.

HOLLINGER, G. A.; GEORGIEV, Y.; MANFREDI, A.; MAXWELL, B. A.; PEZZEMENTI, Z. A.; MITCHELL, B. **Design of a social mobile robot using emotion-based decision mechanisms**. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. Pequim, China: IEEE. 2006. p. 3093-3098.

JÚNIOR, C. D. S.; HEMERLY, E. M. Controle de robôs móveis utilizando o modelo cinamático. **Controle & Automação**, v. 14, n. 4, p. 384-392, 2003.

KARVINEN, T.; KARVINEN, K. **Make:** Arduino Bots and Gadgets: Six Embedded Projects with Open Source Hardware and Software. [S.l.]: O'Reilly, 2011.

LATHAN, C. E.; MALLEY, S. **Development of a new robotic interface for telerehabilitation**. Workshop on Universal accessibility of ubiquitous computing: providing for the elderly. New York, USA: [s.n.]. 2001.

LLAMAS, R.; REITH, R.; SHIRER, M. Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains, According to IDC. **IDC - Analyse the Future**, 2013. Disponivel em: <http://www.idc.com/getdoc.jsp?containerId=prUS24257413>. Acesso em: 17 Setembro 2013.

MALFAZ, M.; SALICHS, M. A. **A new architecture for autonomous robots based on emotions**. 5th IFAC Symposium on Intelligent Autonomous Vehicles. Lisboa, Portugal: IFAC. 2004.

MARTINS, N. A. **Controle adaptativo e robusto de robôs móveis com rodas**. Universidade Federal de Santa Catarina. Florianópolis, p. 204. 2010.

MATARIC, M. J. **The Robotics Primer**. [S.l.]: MIT Press, 2007.

MEDEIROS, A. A. D. A survey of control architectures for autonomous mobile robots. **Journal of the Brazilian Computer Society**, 4, n. 3, Abril 2008.

MEHRABIAN, A. **Basic dimensions for a general psychological theory:** Implications for personality, social, environmental, and developmental studies. 1. ed. Boston: Oelgeschlager, 1980.

MINGUEZ, J.; MONTESANO, L.; DÍAZ, M.; CANALIS, C. **Intelligent Robotic Mobility System for Cognitive Disabled Children**. II International Congress on Domotics, Robotics and Remote-Assistance for All. Madrid, Spain: [s.n.]. 2007.

MURPHY, R. **An introduction to AI robotics**. [S.l.]: MIT Press, 2002.

RANIERI, C. M. **Um ambiente para controle de robôs móveis por portadores de deficiências motoras severas**. Universidade Estadual Paulista "Júlio de Mesquita Filho", relatório de iniciação científica pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Bauru, SP. 2012.

RANIERI, C. M.; ALVES, S. F. D. R.; FILHO, H. F.; CALDEIRA, M. A. C.; PEGORARO, R. **An Envoronment Endowed with a Behavior-Based Control Architecture to Allow Physically Disabled Children to Control Mobile Robots**. Robocontrol - 5th Workshop in Applied Robotics and Automation. Bauru: [s.n.]. 2012.

ROSENBLATT, J. K.; PAYTON, D. W. **A fine-grained alternative to the subsumption architecture for mobile robot control**. 1989 IJCNN International Joint Conference on Neural Networks. Washington: IEEE. 1989. p. 317-323.

SOCIETY OF ROBOTS. Actuators - Servos. **Society of Robots**, 2013. Disponivel em: <http://www.societyofrobots.com/actuators_servos.shtml>. Acesso em: 24 Outubro 2013.

SOCIETY OF ROBOTS. Color sensors tutorial. **Society of Robots**, 2013. Disponivel em: <http://www.societyofrobots.com/sensors_color.shtml>. Acesso em: 24 Outubro 2013.

TURKLE, S. Authenticity in the Age of Digital Companions. **Interaction Studies**, n. 8(3), 2007. 501-517.

VALADÃO, C.; BASTOS, T. F.; BÔRTOLE, M.; PERIM, V.; CELINO, D.; RODOR, F.; GONÇALVES, A.; FERASOLI, H. **Educational Robotics as a Learning Aid for Disabled Children**. Biosignals and Biorobotics Conference (BRC). Vitória: [s.n.]. 2011. p. 1-6.

VIOLA, P.; JONES, M. **Rapid object detection using a boosted cascade of simple features**. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001). Kauai, HI, EUA: [s.n.]. 2001. p. 511-518.