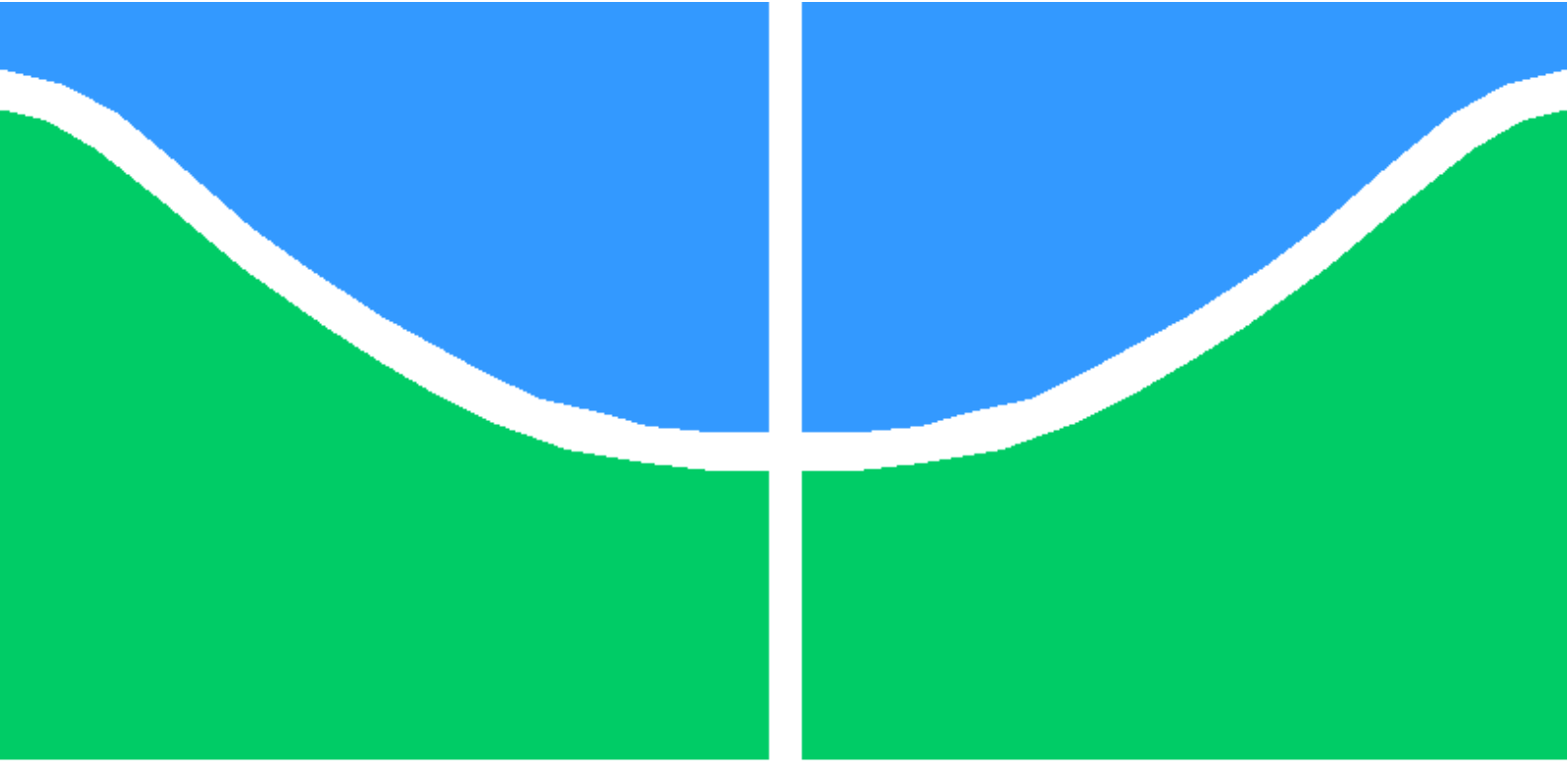


TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE UM ROBÔ-CACHORRO COMPORTAMENTAL: PERCEPÇÃO E MODELAGEM COMPORTAMENTAL

André du Pin Calmon

Nathalie Carvalho Pinheiro

Renan Utida Ferreira

Brasília, dezembro de 2006

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE UM ROBÔ-CACHORRO COMPORTAMENTAL: PERCEPÇÃO E MODELAGEM COMPORTAMENTAL

André du Pin Calmon

Nathalie Carvalho Pinheiro

Renan Utida Ferreira

Relatório submetido ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Geovany Araújo Borges - Docteur, UnB/ Dep (Orientador) _____

Prof. Alexandre Ricardo Soares Romariz - Ph.D., UnB/ Dep (Co-orientador) _____

Prof. Adolfo Bauchspiess - Dr, UnB/ Dep _____

Prof. Ricardo Zelenovsky - Doutor, UnB/ Dep _____

FICHA CATALOGRÁFICA

CALMON, ANDRÉ DU PIN
FERREIRA, RENAN UTIDA
PINHEIRO, NATHALIE CARVALHO
Desenvolvimento de Robô-Cachorro Comportamental: Percepção e Modelagem Comportamental. [Distrito Federal] 2006.
vi, 61p. (ENE/FT/UnB, Engenheiro Eletricista, 2006)
Monografia de Graduação - Universidade de Brasília.
Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1. Robótica comportamental	2. Sistemas embarcados
3. Protocolos de comunicação	4. Sensores para robótica
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CALMON, ANDRÉ DU PIN; FERREIRA, RENAN UTIDA e PINHEIRO, NATHALIE CARVALHO (2006). Desenvolvimento de Robô-Cachorro Comportamental: Percepção e Modelagem Comportamental. Monografia de Graduação, Publicação ENE 02/2006, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 61p.

CESSÃO DE DIREITOS

NOMES DOS AUTORES: André du Pin Calmon, Nathalie Carvalho Pinheiro e Renan Utida Ferreira.

TÍTULO: Desenvolvimento de Robô-Cachorro Comportamental: Percepção e Modelagem Comportamental.

GRAU / ANO: Engenheiro Eletricista / 2006.

André du Pin Calmon
SHCGN 713, Bloco P, Apt. 201 - Asa Norte
CEP 70760-741 - Brasília - DF - Brasil.

Nathalie Carvalho Pinheiro
SQN 112, Bloco K, Apt. 206 - Asa Norte
CEP 70762-110 - Brasília - DF - Brasil.

Renan Utida Ferreira
SQS 105, Bloco C, Apt. 201 - Asa Sul
CEP 70344-030 - Brasília - DF - Brasil.

Dedicatórias

Para os meus avôs.

André du Pin Calmon

Dedico aos meus pais, Nilton e Neuman, e ao meu irmão, Nathan, por sempre me darem apoio.

Nathalie Carvalho Pinheiro

Aos meus pais, irmãos e sobrinhos.

Renan Utida Ferreira

Agradecimentos

A Deus por ter, ao longo desta jornada, iluminado o caminho nos momentos de escuridão, me carregado nos momentos de dificuldade e por estar sempre ao meu lado.
Aos meus orientadores, Prof. Geovany Araújo Borges e Prof. Alexandre Romariz, pelo apoio, dedicação, paciência e amizade ao longo deste trabalho. Foi uma honra poder ser orientando de vocês.
Aos meus colegas e amigos, Nathalie Pinheiro e Renan Utida, pela coragem e dedicação. Duvido encontrar novamente uma equipe tão esforçada e competente quanto esta.
Aos professores que me acompanharam ao longo da minha graduação. Em especial, gostaria de agradecer ao Prof. José Camargo da Costa e ao Prof. Alexandre Romariz, junto aos quais dei os primeiros passos na vida acadêmica.
Aos meus avôs, que não puderam ver este trabalho concluído. Obrigado pelo exemplo de vida e pelo legado.
À minha família, por tudo e por serem tudo para mim.
Ao meu irmão, por ser o meu melhor amigo.
À minha namorada, Fernanda, pelo carinho, amor, amizade e dedicação.
Aos meus colegas que, ao longo destes últimos anos, tornaram-se irmãos.

André du Pin Calmon

Agradecimentos

À "Família ENE02", pela união destes quase 5 anos, tanto nas horas de estudo, em que sempre reinou o compartilhamento de conhecimentos, como nos momentos festivos; aos colegas do LARA, pela boa vontade em transmitir suas experiências e pelo auxílio ao longo do projeto; aos meus colegas de projeto, em especial ao André e ao Renan, de quem a união e dedicação foi essencial para os resultados alcançados.

A todos os professores que se empenham em garantir uma formação de qualidade aos alunos que concluem o curso de Engenharia Elétrica, especialmente aos professores com quem tive contato mais pessoal: o Prof. Geovany Borges, meu orientador neste projeto, e o Prof. Leonardo Menezes, com quem fiz projeto de Iniciação Científica.

Por fim, à minha família e a todos que convivem comigo, pois sempre me apoiaram em minha trajetória e compreenderam quando tive que limitar, por conta de minhas obrigações acadêmicas, as atenções a eles dispensadas.

Nathalie Carvalho Pinheiro

Agradecimentos

Aos meus pais, por terem batalhado tanto para que eu pudesse chegar até aqui.
Aos meus orientadores, professores Adson da Rocha e Geovany Borges, por terem aberto para mim as portas da vida acadêmica e terem me ensinado a gostar mais ainda da ciência e da busca pelo conhecimento.
A todos os meus colegas de curso que se tornaram uma segunda família. Em especial, agradeço ao André e à Nathalie, pelo esforço, empenho e garra e por me levarem a sempre buscar algo mais e ao grande amigo Igor Cardoso por compartilhar tantos momentos importantes da minha vida.
À minha namorada Vívian, pelo amor e companheirismo.
A Deus, por tudo.

Renan Utida Ferreira

RESUMO

O objetivo deste trabalho é propor uma metodologia para criação de um *pet-robot* comportamental. Partindo de uma visão "*top-down*", diversas considerações em alto-nível foram realizadas para nortear o desenvolvimento da arquitetura do robô. Um modelo comportamental foi proposto, permitindo ao robô aprender sobre o mundo à sua volta e adaptar-se ao ambiente no qual está inserido. Para interagir com esse ambiente, diversos sensores foram criados e implementados, de modo que o *pet-robot* pudesse receber os mesmos tipos de estímulos aos quais um animal de estimação real está sujeito.

ABSTRACT

In this work, a framework for developing a pet-robot is presented. Based on a few high level considerations and using a Top-Down approach, the robot's architecture will be defined. A behavior model will be proposed, allowing the robot to learn about the world around him and adapt to the environment in which he is inserted. To interact with this environment, many sensor circuits were designed and implemented, permitting the pet-robot to receive the same types of stimulus a real pet receives.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A revolução tecnológica, iniciada com o surgimento do computador pessoal, vem realizando profundas transformações na sociedade moderna. As pessoas dependem e interagem, cada vez mais, com computadores e diversos outros equipamentos eletrônicos. Desta forma, surge a necessidade de se criar sistemas que atinjam o objetivo para o qual foram projetados com o mínimo de intervenção humana e sejam amigáveis com o usuário, tornando-os fáceis e intuitivos de usar.

Nos últimos 20 anos, surgiram diversos robôs autônomos, capazes de realizar tarefas sem auxílio humano. Para o uso doméstico, já existem vários exemplos comercialmente disponíveis, como o robô-cachorro Aibo, da empresa japonesa Sony, e o aspirador de pó autônomo Trilobite, da Electrolux. Eles apresentam um grau relativamente alto de inteligência, no sentido de conseguirem tomar decisões autonomamente dentro do escopo da sua aplicação. Sendo assim, muitas empresas e pesquisadores acreditam que o século 21 será o século dos robôs autônomos, que irão ajudar as pessoas em diversas atividades cotidianas, não estando restritos apenas a aplicações industriais. Surge, assim, um campo de pesquisa muito promissor que motiva o desenvolvimento deste projeto.

O estudo de robôs sociáveis é interessante para a criação de robôs autônomos que possam interagir de forma eficiente e agradável com as pessoas. Além disso, conforme colocado por Breazeal [7], a construção e o desenvolvimento de robôs socialmente inteligentes permitirão um melhor entendimento do ser humano e das relações sociais existentes na humanidade.

O desenvolvimento de robôs que possam interagir com naturalidade com os humanos é um desafio que vem sendo pesquisado há pelo menos 20 anos. Alguns projetos como, por exemplo, o Kismet [7], criado no MIT, já possuem um comportamento bastante desenvolvido, apresentando respostas típicas de seres humanos. Além disso, já existem, inclusive, robôs disponíveis comercialmente, como é o caso dos robôs-cachorros AIBO (1.1), desenvolvidos pela Sony e o i-Cybie (1.2), desenvolvido pela Silverlit Electronics. A seguir, serão apresentadas algumas características destes robôs, que inspiraram a arquitetura da plataforma de base de *pet-robot* desenvolvida neste trabalho.

1.1.1 AIBO

Provavelmente, o AIBO é o robô-cachorro mais conhecido do mundo. Baseado no conceito de "*robot entertainment*", o AIBO busca possuir aparência e comportamento semelhantes a animais reais, conforme colocado por Fujita, et al. [1]. Para alcançar este objetivo, tentou-se maximizar a complexidade dos movimentos e respostas do robô. Algumas características do robô são:

- Possui uma configuração com quatro pernas, cada uma com 3 graus de liberdade, um pescoço com um grau de liberdade e o rabo com um grau de liberdade também. Conseqüentemente, existe um total de 16 graus de liberdade no cachorro. Para coordenar estes movimentos, o robô possui 16 servomotores DC;
- Uma câmera, um microfone estéreo e alto-falantes para a interação com os humanos;
- A geração de comportamentos é baseada em motivação, de forma semelhante ao comportamento animal real. Desta forma, é possível realizar a combinação de diversos comportamentos, aumentando a versatilidade do animal;

- A arquitetura comportamental é baseada em agentes, modularizando o robô e permitindo que cada parte possua estímulos de entrada e comportamentos diferentes. A partir disto, pode-se aumentar a gama de comportamentos que o AIBO pode ter;
- O *status* interno do robô (que representa os instintos e emoções) muda o comportamento do cachorro em relação a estímulos externos. Sendo assim, é aumentada a complexidade do comportamento do animal;
- Existe a adaptação por meio de aprendizado, de forma que a interação com os usuários permita ao robô desenvolver comportamentos mais complexos.

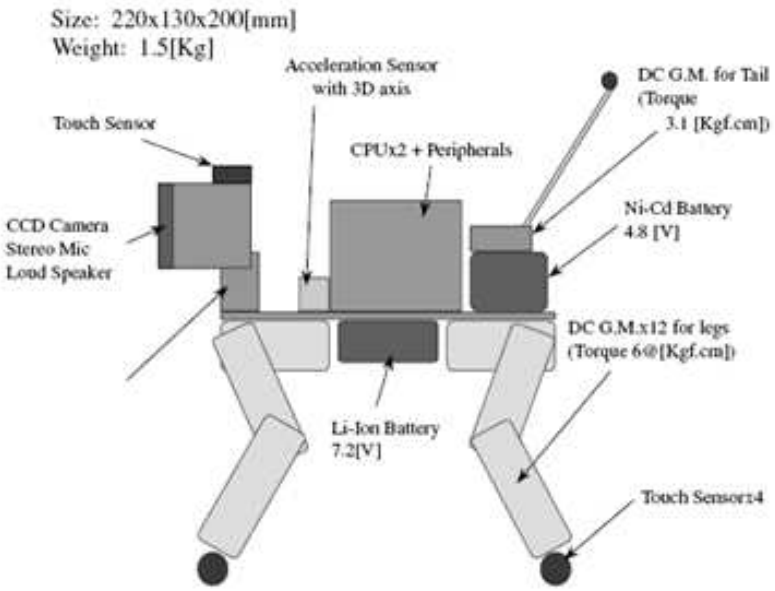


Figura 1.1: Detalhes da configuração mecânica e sensores do AIBO (Fonte: [1]).

1.1.2 i-Cybie

O i-Cybie é um robô-cachorro mais simples do que o AIBO, apresentando apenas quatro comportamentos: feliz, agitado, sonolento e triste. Além disso, ele possui diversos botões espalhados pelo corpo, permitindo que o usuário possa dar ordens ao sistema apenas apertando estes botões. Por exemplo, o robô possui um botão nas costas que, caso apertado, indica que o cachorro deverá sentar e um botão no nariz utilizado para repreender o animal.

Além disso, este robô-cachorro possui diversos sensores, aumentando-lhe a versatilidade comportamental. Dos sensores presentes no i-Cybie, destacam-se:

- Sensor de movimento: ativa o sistema quando detecta algum tipo de movimentação;
- Sensor de obstáculos: provavelmente baseado em ultra-som, porém o tipo exato deste sensor não foi explicitado pelo fabricante;
- Sensor de toque: onde o usuário pode fazer "afagos" no cachorro. Caso seja ativado, o cachorro fica "feliz";
- Sensor de som: utilizado para a entrada de comandos de voz ou outros tipos de comandos sonoros, como palmas;

- Sensor de orientação e equilíbrio: é a central inercial do cachorro, indicando se ele foi derrubado ou não;
- Sensor de infravermelho: o i-Cybie também pode ser comandado por controle-remoto.

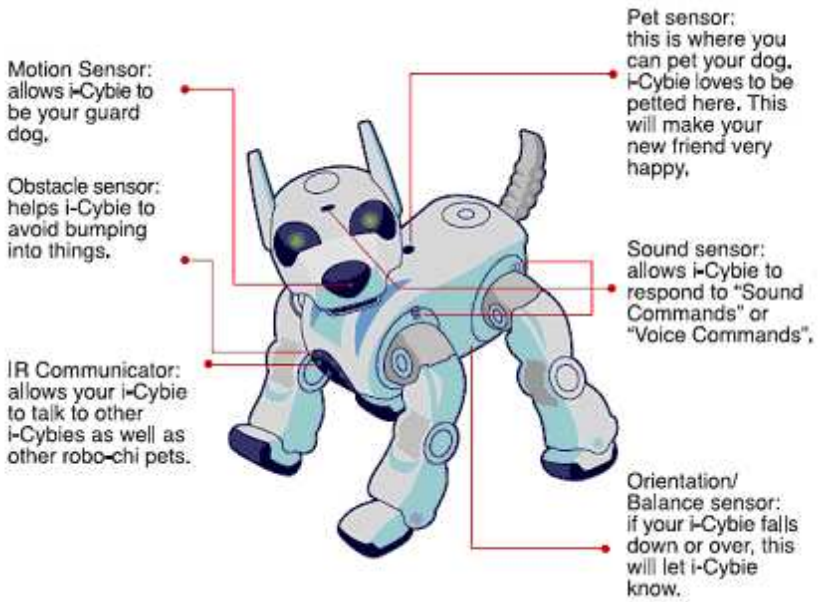


Figura 1.2: Sensores do i-Cybie (Fonte: [2]).

1.2 OBJETIVOS DO PROJETO

Neste trabalho, será proposto e desenvolvido uma plataforma de base de *pet-robot* sociável. Ele deverá interagir com as pessoas de forma semelhante a um animal normal, apresentando um comportamento bem desenvolvido. Com este objetivo, será mostrada a parte eletrônica desenvolvida para o *pet-robot* e os sensores e interfaces com o usuário que ele terá. Toda a parte eletrônica será criada enfatizando a modularidade, permitindo a inserção de mais funções no robô futuramente. Além disso, serão expostas as perspectivas e possibilidades para o desenvolvimento da parte comportamental do *pet-robot*.

1.3 APRESENTAÇÃO DO MANUSCRITO

O Capítulo 2 apresenta de maneira superficial mas abrangente a estrutura geral do projeto com uma breve descrição de cada etapa. Em seguida, o Capítulo 3 descreve toda a parte de compreensão do robô com relação ao mundo externo por meio de sensores, explicando em detalhes cada tipo de sensor utilizado e suas funções. Além dos sensores, é descrita a reprodução de áudio e o protocolo de comunicação de cada etapa de *hardware* com o sistema central. O Capítulo 4 traz todo o embasamento teórico e o funcionamento da inteligência com relação às respostas aos estímulos externos. Ele apresenta também o Módulo de Percepção, que é responsável pela "tradução" do sensoriamento para estímulos comportamentais. Por fim, temos o capítulo das conclusões, seguido dos anexos, que trazem os programas desenvolvidos tanto para os microcontroladores quanto para o PC. A parte de locomoção não foi detalhada em nenhum capítulo por ser assunto de trabalho de outro grupo, enquanto o presente grupo ficou responsável pelos sensores, atuadores e comportamento.

Como o objetivo inicial era a construção de um robô-cachorro, em muitas partes do texto se faz referência ao robô como sendo um cachorro. Toda a teoria e os módulos desenvolvidos, no entanto, foram criados de forma genérica, permitindo que fossem usados na criação de qualquer tipo de *pet-robot*. A última versão do projeto acabou, então, por transformar-se em um robô-tamanduá. Entre outros motivos, a equipe envolvida no desenvolvimento do robô optou por esta mudança por ser o tamanduá-bandeira um símbolo da fauna brasileira.

2 ARQUITETURA PROPOSTA

2.1 ARQUITETURA GERAL

2.1.1 Introdução

Para um robô-cachorro ser similar a um cachorro real, é fundamental que ele apresente comportamentos próximos aos do animal real. Observa-se que, para tanto, não basta que o robô apresente atitudes pré-programadas. É necessário que ele possa interagir de forma ativa com o meio no qual ele está inserido e atender a alguns objetivos que norteiam a sua "existência". Para tanto, o robô-cachorro deverá conseguir interpretar o que está ocorrendo em sua volta e, a partir destes dados, realizar decisões que atendam a objetivos pré-estabelecidos. Além disso, o cachorro deverá aprender a partir da sua experiência, de forma que ele possa fazer as melhores escolhas possíveis, e cumprir os seus objetivos da forma mais eficiente possível.

Logo, tentaremos propor um modelo que se aproxime do que ocorre dentro de animais reais. Para tanto, será necessário realizar algumas considerações sobre o comportamento do cachorro e adaptá-los para uma implementação computacional. Todo o modelo será realizado seguindo uma metodologia *top-down*, partindo de considerações genéricas e seguindo até a implementação em hardware. No robô, o comportamento e o aprendizado serão gerenciados por um módulo comportamental, que será explicado adiante.

2.1.2 Estrutura do robô-cachorro

Conforme indicado na Figura 2.1 abaixo, o robô-cachorro será organizado em cinco módulos: sensores, Módulo de Percepção, Módulo Comportamental, Módulo de Linguagem e Locomoção e os Atuadores. A seguir, cada módulo será explicado sucintamente.

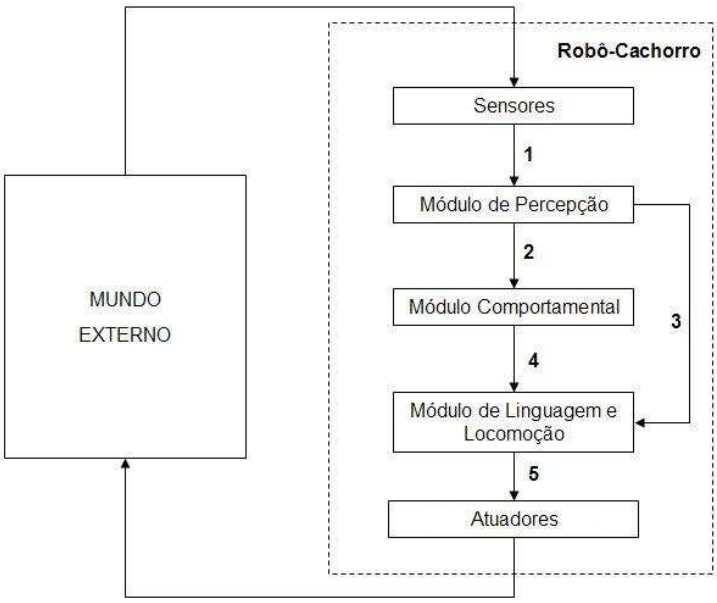


Figura 2.1: Organização dos módulos do robô-cachorro. As interfaces entre os módulos estão representadas pelos números de 1 a 5.

2.1.2.1 Sensores

O sistema de sensoreamento do robô-cachorro se dá por meio de diferentes blocos de sensores, cada um responsável por determinado tipo de aquisição. Essas aquisições se referem a aproximações dos sentidos de um animal implementadas em *hardware* (exceto pela visão, que é implementada diretamente no PC) de forma a tornar o robô o mais verossímil possível. O sensor de toque, referente ao tato, deve determinar quando o cachorro é tocado e por quanto tempo se dá esse toque. A identificação de intensidade sonora seria uma aproximação da audição e é responsável pela detecção de direção de um som a partir da comparação de som capturado por diferentes microfones. Pensa-se em, futuramente, aprimorar o sistema de forma que não se faça apenas comparação de intensidade de som, mas também a real interpretação do áudio por meio de ferramentas de decomposição espectral. A visão é feita por uma câmera e tem por finalidade a detecção de faces e percepção de intensidade luminosa. O sensor de ultra-som deve determinar a distância de um obstáculo posicionado à frente do robô, bem como a sua aproximação com relação ao obstáculo. Em um cachorro real, esta função é realizada pela visão. No entanto, em termos de implementação, o ultra-som se aplica muito para atingir o objetivo desejado. A figura 2.2 mostra um bloco de comunicação intermediando cada sensor com o sistema central no PC.

O bloco de comunicação interage com os sensores pelo padrão serial RS485 (diferencial) e com o PC pelo padrão serial RS232 (não-diferencial). A implementação em *hardware* dos diversos sensores utiliza microcontrolador ATmega8. Os microcontroladores são responsáveis pela interpretação dos dados adquiridos e envio para o PC de informação caracterizada. Por exemplo, a detecção de intensidade sonora processa o som adquirido pelos microfones e retorna para o PC apenas se o som vem da direita, da esquerda ou de uma região central (a frente, acima ou de trás) do cachorro.

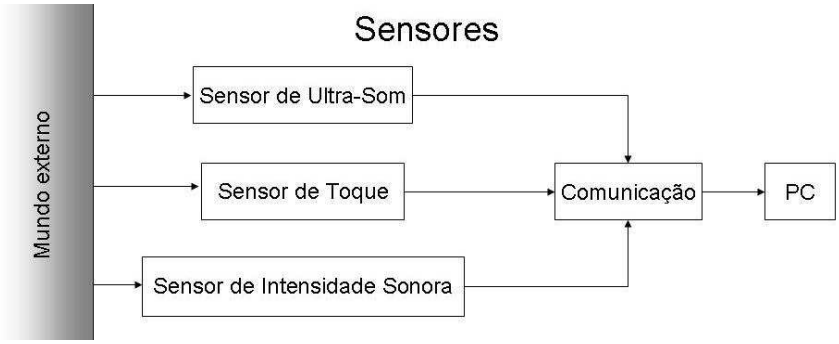


Figura 2.2: Camada de Sensores.

2.1.2.2 Módulo de Percepção

Este módulo é responsável por interpretar os dados recebidos dos microcontroladores e transformá-los em informações relevantes para a atitude a ser tomada pelo robô-cachorro. Dois tipos possíveis de resposta podem aparecer. O primeiro tipo é chamado de Saída do Módulo de Percepção e designa informações a serem transmitidas ao Módulo Comportamental, que podem ser de cinco tipos:

- Nível de atenção;
- Carinho/Agressão;
- Claro/Escuro;
- Som alto/Som baixo;
- Presença-Ausência/Ausência-Presença.

O segundo tipo possível de resposta seria uma resposta reflexiva. Esta resposta envia informação diretamente ao Módulo de Linguagem e Locomoção e determina que o robô tome uma atitude locomotiva, seja parar para não se chocar contra um obstáculo ou virar sua cabeça para a direção de onde ele capta som com maior intensidade.

2.1.2.3 Módulo Comportamental

O módulo comportamental é responsável por coordenar as atitudes e reações do robô-cachorro. Ele analisa os estímulos aos quais o robô está sujeito e tenta adaptar o seu comportamento de forma a atendê-los. A resposta a estes estímulos toma a forma de um estado emocional que o cachorro pode assumir. Em um primeiro momento, estes estados emocionais serão: alegria, tristeza, raiva, neutro, medo e dormir (sonolência). Neste contexto, o objetivo principal do cachorro será maximizar o nível de atenção do usuário. Desta forma, a partir de um modelo de aprendizagem por reforço, o robô aprende a escolher emoções que levem a uma resposta positiva do usuário. Por exemplo, o cachorro-robô, ao perceber que o usuário sempre o acaricia quando ele fica triste, tenderá a passar mais tempo neste estado emocional.

2.1.2.4 Módulo de Linguagem e Locomoção

Este módulo tem por objetivo o envio de diretivas para os LEDs (representando olhos) e para os atuadores a partir de informações recebidas tanto do módulo comportamental quanto do módulo de percepção. No caso de recebimento do módulo comportamental, as respostas determinam a movimentação das patas para realizar determinadas atividades, como sentar ou deitar, por exemplo. Existem também as respostas relativas ao estado emocional do robô, como o abanar da cauda e a cor dos LEDs. As informações provenientes do módulo de percepção, conforme apresentado anteriormente, determinam ações reflexivas.

2.1.2.5 Atuadores

Esta parte tem por finalidade expressar o estado emocional do robô-cachorro e realizar a locomoção. O estado emocional é apresentado por meio de reprodução sonora e LEDs. O tipo de som e a cor do LED exprimem emoções distintas. A reprodução do som deveria partir de um bloco de *hardware* também com um microcontrolador e o som seria lido de uma memória *flash*, entretanto dificuldades foram encontradas, de forma que não se conseguiu atingir este objetivo e a reprodução passou a ser feita diretamente do PC. Já os LEDs são controlados por um microcontrolador, até por ser sua implementação mais simples que a reprodução do som.

As funções de locomoção, tanto das patas quanto da cabeça e do rabo, são parte do trabalho desenvolvido por um projeto paralelo a este. Gustavo Cotta e Laurindo Neto são os responsáveis por toda a parte de controle e atuação de servo-mecanismos destinados a reproduzir os movimentos, bem como pela construção da estrutura mecânica do robô-cachorro.

2.1.3 Interfaces

Entre os módulos do cachorro existirão 5 interfaces, conforme indicado na Figura 2.1. Essas interfaces representam os tipos de informações a serem transmitidas de módulo para módulo. As interfaces 1 e 5 são entre microcontroladores e o PC e as interfaces 2, 3 e 4 ocorrem todas no PC.

- Interface 1: contém dados dos pré-processados pelos microcontroladores. Esses dados serão analisados no Módulo de Percepção.
- Interface 2: contém informações de estímulo e se dá entre o Módulo de Percepção e o Módulo

Comportamental. As informações presentes serão:

1. Nível de atenção (utilizado no modelo de aprendizagem);
2. Carinho/Agressão;
3. Claro/Escuro;
4. Som Alto/Baixo;
5. Presença/Ausência;

- Interface 3: estarão presentes informações referentes ao comportamento reflexivo do robô. Mais especificamente, as informações são de cessar movimento pela aproximação de um obstáculo e de girar a cabeça para a direção de origem de um som lateral.
- Interface 4: nela estarão caracterizadas as informações pertinentes ao estado emocional do cachorro, gerando os subsídios necessários para a realização de uma atualização da sua postura e expressão pelo estado emocional.
- Interface 5: entre o PC e os AVRs. São enviadas informações específicas para a atualização dos motores e dos LEDs da face.

3 SISTEMAS DE SENSORIAMENTO E ATUAÇÃO

3.1 ULTRA-SOM

Para que um robô consiga se deslocar, é necessário que ele possua sensores que lhe informem se há algum obstáculo à sua frente e a que distância ele se encontra, de modo que o robô desvie sua trajetória quando um objeto estiver atrapalhando sua passagem. Um sensor muito usado com esta finalidade é o de ultra-som. Neste projeto, a medição de distâncias é feitas por meio deste sensor.

O módulo do ultra-som consiste em um circuito envolvendo um microcontrolador ATmega8, responsável por gerar o trem de pulsos a ser emitido e analisar o sinal recebido do receptor. Este microcontrolador é, ainda, responsável pela comunicação serial com o sistema central, que está sendo testado em um computador, porém em uma etapa futura do projeto também será embarcado. Os transdutores de ultra-som localizam-se na parte frontal do robô e estão ligados à placa de ultra-som, que fica no tronco do animal. Nesta, encontra-se o microcontrolador e um circuito de recepção, o qual amplifica a resposta do receptor e a transforma em um sinal digital da forma desejada (o sinal é nulo enquanto não houver obstáculos e passa para 1 quando algo for detectado). O projeto deste circuito teve como base o modelo do transdutor comercial MaxSonar-EZ1 [8], que já vem com um circuito de recepção integrado. Os filtros foram, no entanto, reprojetados, de forma a otimizar o circuito para a freqüência de operação desejada e algumas outras modificações foram realizadas para ajustar melhor o circuito à nossa aplicação.

O programa embarcado no microcontrolador da placa de ultra-som gera periodicamente uma onda de 15 pulsos a 40kHz em dois pinos. O valor de um pino é o inverso do outro, de modo a gerar uma entrada diferencial para o transdutor emissor de ultra-som. O uso da tensão diferencial foi adotado para aumentar a potência de emissão, garantindo uma região de detecção maior. Ao iniciar a geração dos pulsos, o microcontrolador dispara um cronômetro, que é usado tanto para o cálculo da distância, como para delimitar o intervalo de tempo entre uma emissão e outra (16ms). Este intervalo foi escolhido de forma que fosse possível realizar cinco medições e calcular a média entre elas em menos de 100ms, pois este é o tempo entre um pedido de dados do sistema central e outro.

3.1.1 Medição de Distância com Ultra-som

O termo ultra-som refere-se às ondas com freqüências mais altas que a audibilidade humana, cujo valor máximo é em torno de 18kHz. As ondas ultra-sônicas obedecem às leis básicas da acústica, porém possuem comprimento de onda curto, o que reduz a difração e a dispersão pelos obstáculos e facilita o direcionamento da emissão.

O dispositivo mais usado para transmissão de ultra-som é o cristal piezo-elétrico, que converte energia mecânica em elétrica e vice-versa. Dessa forma, a aplicação de uma tensão senoidal no transmissor produz deformação senoidal correspondente. A vibração do cristal piezo-elétrico passa para o meio de transmissão e neste se propaga. Para aplicação em medição de distâncias, são usadas ondas de pequena potência, que não deformem permanentemente o meio. No receptor, ocorre o processo inverso, de modo que a vibração provocada por uma onda ultra-sônica é transformada em tensão.

A partir da equação de propagação de ondas e da lei geral dos gases, a velocidade do som nos gases pode ser deduzida (Ref. [9]):

$$C_L = \sqrt{\kappa \cdot R \cdot T}, \tag{3.1}$$

em que κ é o índice adiabático do gás (relação entre calor específico a pressão constante e calor específico a volume constante), R é constante universal dos gases dividida pela massa molar do gás e T é a temperatura absoluta em Kelvin.

Para o ar seco, encontra-se na literatura $R = 287,05 J/(kg \cdot K)$ e $\kappa \approx 1,4$. Assim, a equação reduz-se a

$$C_L \cong 20,04 \cdot \sqrt{T} m/s. \tag{3.2}$$

O uso de ultra-som para a modelagem do ambiente é baseado, portanto, nessa fórmula de velocidade, pois a idéia é colocar um transmissor e um receptor próximos e emitir um trem de pulsos. Como a impedância do ar é da ordem de 1000 a 10000 vezes maior que a maioria dos líquidos e sólidos, o coeficiente de reflexão é praticamente igual a 1 para a maioria das fronteiras ar/líquido e ar/sólido. Dessa forma, um trem de pulsos emitido, ao encontrar um obstáculo, é refletido e captado pelo emissor. O sinal elétrico resultante no emissor é, então, processado e, a partir do tempo medido entre emissão e recepção e da fórmula da velocidade exposta acima, a distância é calculada.

Neste projeto, foi inicialmente usado um valor fixo para a temperatura. Para a primeira versão do robô-cachorro, o erro resultante na distância não representa um problema, pois esta não é uma aplicação que exige grande precisão nas medidas. Além disto, o robô não estará sujeito a temperaturas muito diferentes, uma vez que deve funcionar apenas em ambientes fechados onde não há, por exemplo, exposição ao sol. O aprimoramento das medições de distância, entretanto, pode tornar necessário numa fase futura do robô-cachorro ou mesmo em outras aplicações. A correção desta aproximação pode, então, ser feita com a utilização de um sensor de temperatura.

Na medição de distâncias com o ultra-som, deve-se tomar o cuidado de, entre uma medição e outra, esperar um intervalo de tempo suficiente para que todos os ecos da primeira medição se extinguam, uma vez que o pulso pode ser refletido no receptor e novamente refletido no obstáculo até que a atenuação do meio o elimine. Além disso, é importante observar que há um valor limite para alcance das medições, uma vez que a onda é atenuada pelo meio em que se propaga. A distância máxima de detecção é determinada pela energia contida no trem de pulsos. No módulo desenvolvido para aplicação no robô-cachorro, as medições realizadas para distâncias por volta de 30cm ainda são confiáveis. Distâncias maiores que esta, no entanto, diminuem a confiabilidade gradualmente.

É importante ressaltar que distâncias muito pequenas, de aproximadamente 5cm ou menos, não resultam em boas medições, visto que a grande proximidade do obstáculo dificulta o retorno direto da onda transmitida para o receptor, pelo fato de os dois transdutores estarem envoltos em pequenos tubos para guiar a onda. Estes guias foram usados por evitarem que os pulsos transmitidos atingissem o receptor mesmo sem reflexão (caminho direto), porém, na saída dos tubos surge um cone de abertura que, para obstáculos muito próximos, pode tornar insuficiente uma única reflexão para os pulsos alcançarem o receptor.

3.1.2 Arquitetura Eletrônica

Para transmissão, o circuito é muito simples, pois deve-se apenas ligar o transdutor de emissão aos pinos do microcontrolador nos quais os trens de pulsos estão sendo gerados, conforme mostrado na Fig. 3.1. Já o circuito de recepção do ultra-som é constituído de três estágios com amplificadores MCP604. Os dois primeiros estão na configuração de filtros passa-altas, projetados de forma a fornecer um ganho alto, mantendo as freqüências de corte suficientemente baixas para maximizar o ganho nas freqüências de ultra-som (mais especificamente, em $40kHz$, pois é a freqüência usada nesta aplicação). A Fig. 3.2 mostra como são calculados o ganho em altas freqüências e as freqüências de corte para esses filtros. O ganho do primeiro estágio é de $163,5$, enquanto o do segundo é 11, totalizando um ganho de $1798,5$. As freqüências de corte mais baixas são $138,27Hz$ para o primeiro e $2,055kHz$, enquanto as mais altas são iguais para os dois estágios e valem $22,607kHz$.

O diodo entre o segundo e o terceiro estágios, juntamente com o capacitor de 10nF, constitui um detector de envoltória. Como o diodo está reversamente polarizado, a parte positiva do sinal é eliminada e a tensão sobre o capacitor é a envoltória negativa. Como o terceiro estágio é um amplificador na configuração inversora que compara esta entrada com a tensão de referência (pelo divisor de tensão, 2,13V), o sinal na

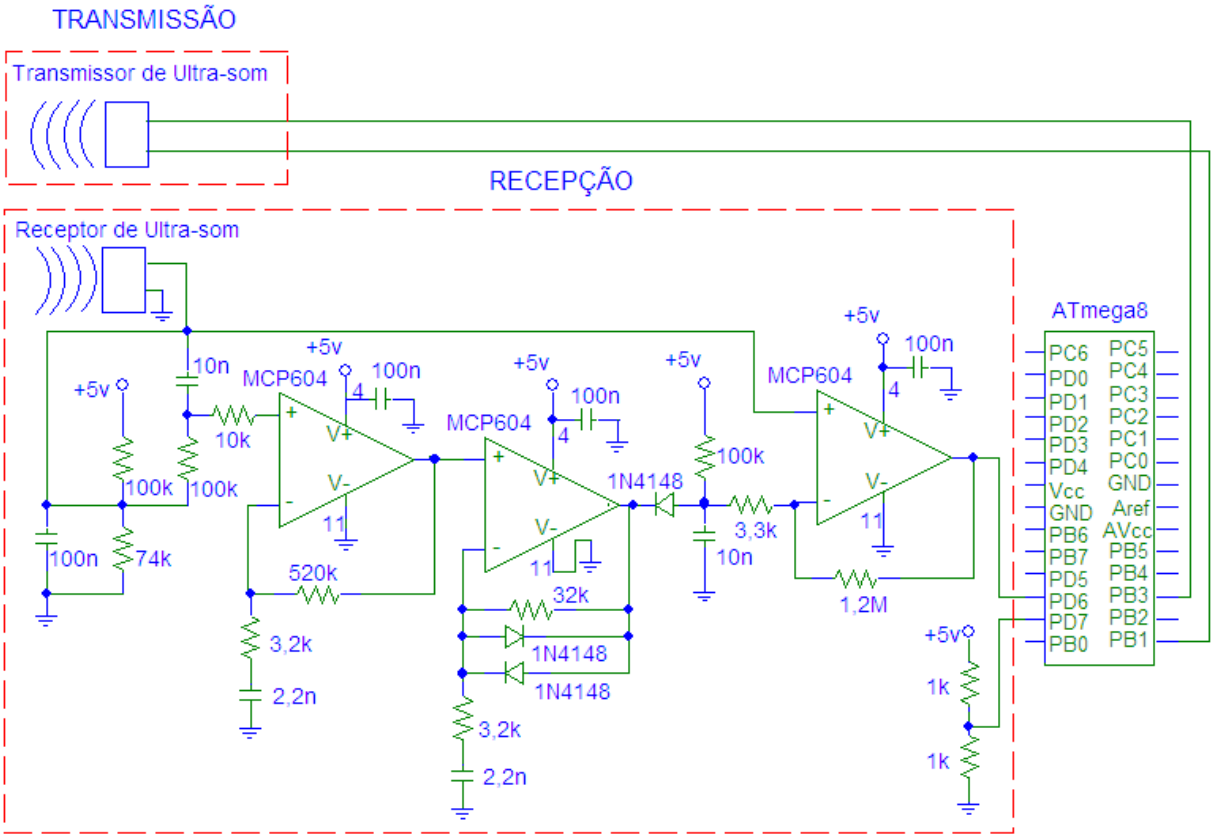


Figura 3.1: Circuito de transmissão e de recepção do módulo do ultra-som.

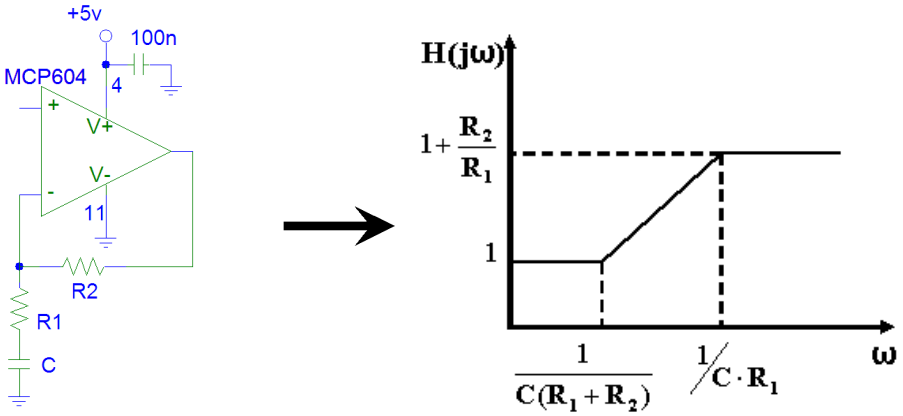


Figura 3.2: Configuração dos dois primeiros amplificadores e resposta em freqüência destes subcircuitos.

saída é uma onda com apenas a parte acima da tensão de operação. O ganho neste estágio, no entanto, é muito alto $-R_2/R_1 = -363,636$, onde R2 é a resistência de realimentação. Isso faz com que a saída sature e, portanto, fique em 1 quando o receptor captar algum sinal e em 0 quando a perturbação for nula (Fig. 3.3).

A saída do último estágio é conectada no terminal positivo do comparador analógico do microcontrolador, enquanto a entrada negativa do comparador está ligada a uma tensão de 2,5V. Dessa forma, quando um obstáculo é detectado e a saída do circuito de recepção é levada para nível lógico 1, o microcontrolador detecta uma mudança no resultado de seu comparador analógico e uma interrupção pode ser gerada.

O transdutor receptor é colocado próximo ao transmissor, de forma a tornar possível a aproximação

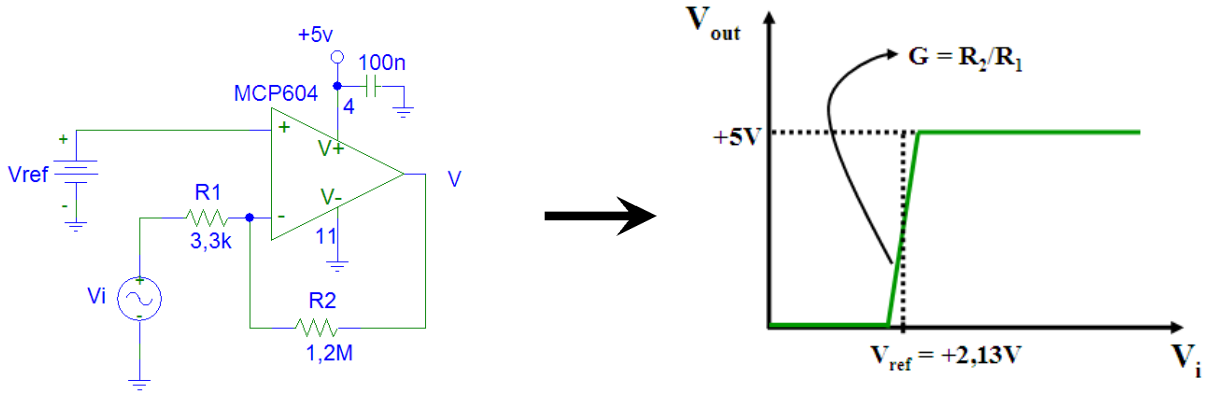


Figura 3.3: Configuração inversora do amplificador do terceiro estágio e representação da saída saturada.

da distância medida entre o obstáculo e o robô como sendo metade da distância calculada a partir do tempo entre transmissão e recepção dos pulsos gerados. Esta configuração espacial, no entanto, provoca a captação do sinal transmitido diretamente pelo receptor, independente da reflexão em objetos. Para evitar a geração de um pulso na recepção e a conseqüente falsa detecção de obstáculos, os transdutores foram envolvidos em tubos de papel, os quais atuam como guias de onda, eliminando o pulso indesejado.

3.1.3 Geração e processamento digital dos sinais do ultra-som

Para a emissão de pulsos a 40kHz, é usado o temporizador 0 do microcontrolador ATmega8, que realiza contagens em 8 bits. O temporizador 1, cuja contagem é em 16 bits, foi usado como cronômetro, realizando a medição do tempo total entre duas emissões e do tempo até encontrar um obstáculo (caso isto ocorra).

O comparador analógico deste microcontrolador também é usado, com uma tensão de referência aplicada na entrada negativa (2,5V) e o sinal do receptor na entrada positiva. Este sinal fica normalmente em nível baixo, porém quando um obstáculo é detectado seu sinal é alterado para nível alto, provocando uma interrupção por flanco de subida do comparador.

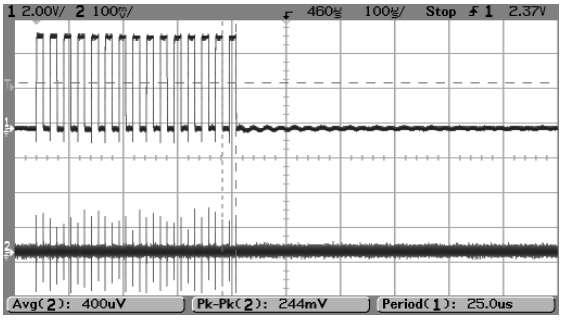
A dinâmica de operação do microcontrolador é toda baseada em interrupções. Desta forma, a função principal apenas faz as configurações iniciais e em seguida aguarda cada chamada de interrupção. Assim que o módulo de ultra-som é alimentado e toda a inicialização é executada, o temporizador 0 é ligado. Com isso, o microcontrolador espera por sua interrupção de *overflow*. Ao emitir o primeiro pulso, o temporizador 1 é ligado e a interrupção do comparador analógico é habilitada. Após gerar todos os pulsos, a interrupção de *overflow* do temporizador 0 ainda foi configurada para ser chamada algumas vezes e, em seguida, desabilitar sua própria ocorrência e habilitar a interrupção de *overflow* do temporizador 1.

Inicialmente, a idéia foi de criar um tempo mínimo para que o cálculo da distância fosse realizado, um pouco maior que o tempo de emissão do trem de pulsos, criando uma "zona morta", em que seria retornada distância nula para evitar problemas de acoplamento entre emissor e receptor. Esta é uma solução de projeto que pode ser a melhor alternativa para algumas aplicações. No ultra-som do robô-cachorro, no entanto, optou-se por calcular a distância mesmo quando o obstáculo está bem próximo. Isto não causa problemas de múltiplas medições para o mesmo trem de pulsos emitidos, pois a interrupção de comparação analógica, responsável pelo cálculo da distância, é desabilitada quando o correspondente código é executado e só é reabilitada no início da próxima emissão de pulsos.

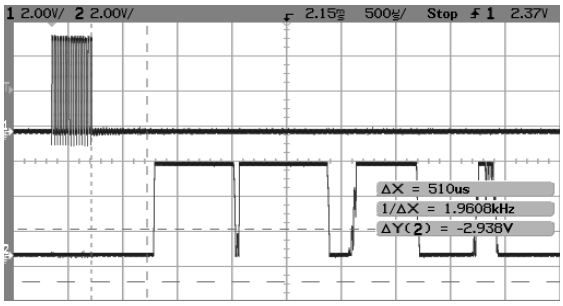
Quando o cronômetro indica que já passaram 16ms (interrupção de *overflow* do temporizador 1), uma nova emissão de pulsos é ativada (habilitação do temporizador 0) e, se cinco medidas já tiverem sido realizadas, a média delas é calculada e escrita na fila de saída do protocolo de comunicação, de forma a ser enviada ao computador assim que este requisitar os dados deste sensor.

3.1.4 Resultados Experimentais

O trem de pulsos gerado pelo microcontrolador ATmega8 para entrar no transdutor emissor de ultra-som é apresentado na Fig. 3.4 (a), capturada com o osciloscópio. Nesta figura, é possível perceber que a temporização do programa funciona adequadamente. Na Fig. 3.4 (b), observa-se como é o sinal enviado do circuito analógico de recepção para o digital ao encontrar-se um obstáculo. A alteração do nível lógico do sinal provoca a interrupção de cálculo da distância.



(a) Trem de pulsos que entra no transmissor e sinal de saída do circuito de recepção quando não há obstáculo (escala 20 vezes maior para mostrar ruído pequeno)



(b) Trem de pulsos no transmissor e sinal de saída do circuito de recepção quando há obstáculo a aproximadamente 15 cm.

Figura 3.4: Sinais nos transdutores do ultra-som.

Para verificação das medidas de distância, um obstáculo foi colocado em diferentes posições e, para cada uma, foram realizadas 300 medições. O valor da distância medida pelo sensor era enviado para o computador e armazenado para possibilitar a análise estatística. Como resultado destas medições, obtiveram-se os gráficos da Fig. 3.5. Os dados também são mostrados na Tabela 3.1. O microcontrolador envia para o computador as distâncias em milímetros. Para tornar as distâncias mais intuitivas, entretanto, o valor teórico de cada medição foi mantido em centímetros.

Tabela 3.1: Resultados dos testes do módulo de ultra-som.

Valor Teórico (cm)	**Média (mm)**	**Mediana (mm)**	**Variância (mm)**	**Desvio-padrão (mm)**	**Mínimo (mm)**	**Máximo (mm)**
5	61,94	62,00	0,66	0,65	60,33	62,67
7	74,97	74,33	2,23	1,41	73,67	78,67
10	105,17	105,00	6,53	2,29	102,00	110,33
12	125,86	125,67	1,82	1,20	124,00	129,00
15	158,01	158,00	3,24	1,79	154,33	161,33
17	179,09	179,33	2,97	1,46	177,00	181,33
20	208,01	208,13	3,72	1,86	204,75	212,25
22	230,65	230,67	1,56	1,23	227,33	233,33
25	257,99	258,00	2,97	1,58	255,33	263,00
27	283,60	283,67	2,47	1,51	281,33	286,00
30	309,75	310,00	2,70	1,63	305,67	312,67

Da análise do primeiro gráfico (ou da tabela), é possível perceber que o resultado das medidas ficou bem próximo do valor real da distância, porém sempre um pouco maior. Para corrigir a medida dos sensores, pode ser usado o fato de, na faixa de distâncias que será usada, o erro percentual situar-se sempre por volta

de 5% do valor medido, conforme mostrado na Fig. 3.5 (b). A correção pode, então, ser feita por *software*, apenas multiplicando-se a distância obtida por 1,05.

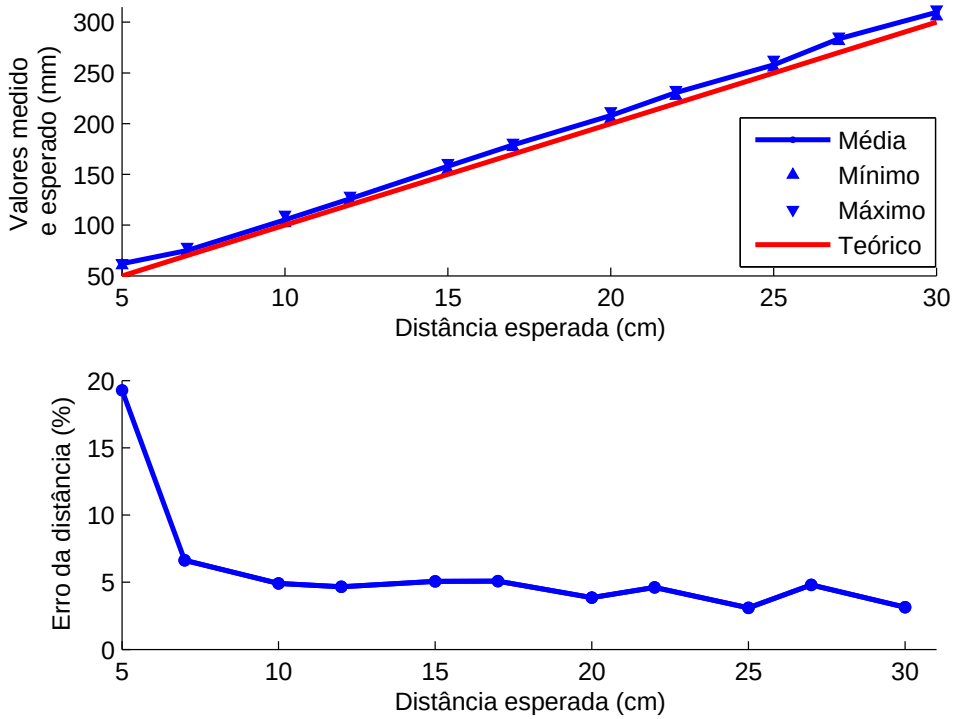


Figura 3.5: Análise das medições de distância do ultra-som: (a) Valor medido médio, mínimo e máximo, comparados com o valor teórico de cada medição; (b) erro percentual das medições.

É interessante ressaltar, ainda, que as medidas para obstáculos muito próximos dos transdutores apresentam um erro mais elevado, conforme já mencionado. Assim, para objetos a 5cm, os resultados retornados pelo ultra-som foram mais elevados (pelo menos 6cm). As medidas para 7cm, no entanto, já apresentaram erros percentuais bem mais baixos, conforme Fig. 3.5 (b).

3.2 TOQUE

Uma das possíveis formas de interagir com o cachorro é por meio do toque. O cachorro percebe este tipo de estímulo e, por meio da duração do toque, classifica como uma interação positiva ou negativa. Deste modo, um toque rápido pode ser interpretado como um tapa, enquanto um toque mais lento é classificado como carinho.

Além desta classificação do estímulo tátil, o robô-cachorro também é capaz de identificar qual dos vários sensores espalhados em sua superfície está sendo tocado. Isto é útil para criar um comportamento reflexivo, pois o cachorro pode, por exemplo, perceber que foi tocado em uma perna traseira e andar para a frente ou mexer a perna. Este comportamento pode, também, ser resultado do estado emocional do cachorro. Assim, se o cachorro estiver com medo, ele pode se tornar arredio e tender a se afastar quando for tocado.

Para criar o sensor de toque, inicialmente foi desenvolvido um circuito, que funcionou muito bem nos primeiros testes, porém, com a inclusão do microcontrolador, percebeu-se que havia uma grande sensibilidade a interferências eletromagnéticas. O sensor apresentava algumas flutuações e chegava até a não

funcionar, dependendo dos componentes que estavam ligados ou mesmo próximos. Algumas tentativas de aumentar sua proteção eletromagnética foram realizadas, como a inclusão de um anel de terra ao redor do circuito sensor, porém não foi encontrada uma solução eficiente para resolução do problema. Outras alternativas foram procuradas e optou-se, então, pela compra dos CI's QT110-DG, do *Quantum Research Group*.

3.2.1 Arquitetura Eletrônica do Sensor Desenvolvido

O princípio de funcionamento do sensor de toque desenvolvido está baseado no fato de os seres humanos funcionarem como antenas para baixas freqüências. Como a transmissão de energia elétrica no Brasil é feita em 60Hz, este é o valor da freqüência predominante do ruído do ambiente e, por ser esta uma freqüência baixa, os homens absorvem e propagam com boa qualidade. Desta forma, ao tocar um objeto não condutor em um dos sensores, o cachorro não responde. Se uma pessoa quiser interagir com ele e tocá-lo, todavia, o sensor perceberá um sinal de 60Hz na entrada e responderá a este, alterando o nível lógico da saída de 1 para 0 e permanecendo neste até que o usuário solte o terminal do sensor.

O diagrama do circuito implementado é mostrado na Figura 3.6. Para facilitar a análise, o circuito foi dividido em duas partes, conforme mostrado na figura. A primeira parte do circuito é constituída pelo eletrodo de entrada, os dois diodos, um resistor *pull-up* de 1MΩ e a primeira porta NAND, que possui uma entrada fixa em +5V. A segunda parte do circuito é constituída por um diodo, pelo resistor de 100kΩ que está em paralelo com o capacitor de 1μF e a segunda porta NAND.

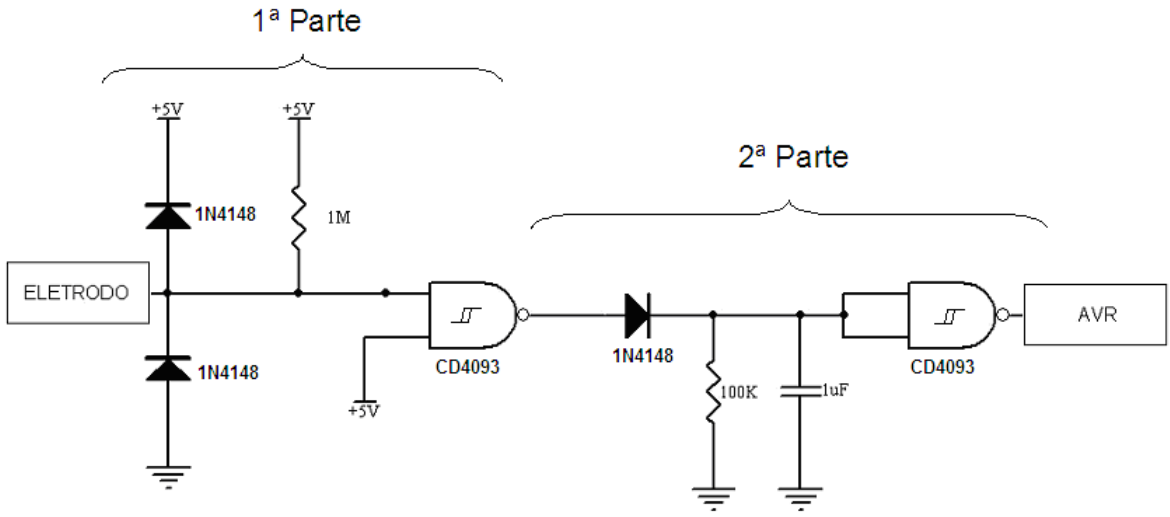


Figura 3.6: Circuito inicialmente implementado para detecção de toque.

Quando o eletrodo não estiver em contato com nada, o resistor de 1MΩ garante que a entrada da porta NAND a qual está conectada seja +5V. Desta forma, a saída da porta será 0V, correspondente ao nível lógico 0. Nesta situação, a entrada da segunda parte do circuito será uma tensão de 0V. Logo, o diodo não irá conduzir e o resistor de 100kΩ atuará como um *pull-down*, levando as duas entradas da segunda porta NAND para o nível lógico 0. Conseqüentemente, a saída do circuito, que está conectada ao AVR, será +5V.

No momento que alguém tocar no eletrodo, surgirá uma tensão AC de pequena amplitude e freqüência próxima de 60Hz na entrada do circuito. O diodo ligado ao terra irá conduzir, de forma que a entrada da porta NAND à qual está conectado esteja no nível lógico 0. Sendo assim, a saída da porta NAND será +5V. Neste caso, a entrada da segunda parte do circuito será, momentaneamente, +5V, fazendo com que o diodo conduza e o capacitor de 1μF seja carregado rapidamente. Como as entradas da segunda porta

NAND estão em curto-circuito, a saída do circuito será +5V.

Quando a tensão AC da entrada for positiva, a saída da primeira parte do circuito voltará a ser 0V. Desta forma, o diodo da segunda parte não irá conduzir e o capacitor irá descarregar-se com uma constante de tempo de 1/RC, ou 10 segundos. Como a freqüência do sinal de entrada é de aproximadamente 60Hz, o capacitor, durante o ciclo positivo do sinal de entrada, irá descarregar-se muito pouco, de forma que a tensão sobre ele continuará próxima de +5V, correspondente ao nível lógico 1,e a saída do circuito, após passar pela porta NAND, que atua como inversora, será nível lógico 0. Logo, enquanto alguém estiver tocando no eletrodo, a saída do circuito será o nível lógico 0.

3.2.2 O Processamento Digital do Sensor de Toque

Conforme explicado na seção anterior, o circuito sensor de toque gera como saída um sinal que fica em 1 quando não há ninguém tocando em seu terminal e em 0 enquanto está sendo tocado. Este sinal serve como entrada em um microcontrolador ATmega8 responsável pelo processamento de todos os sensores de toque. Este microcontrolador lê periodicamente a entrada de cada sensor e, se detectar que houve toque, liga um cronômetro.

O ATmega8 guarda em um *byte* a informação dos sensores que foi lida. Para isto, foi criada uma correspondência de um bit para cada sensor, de modo que, se houver vários terminais sendo tocados, o sistema central possa identificar todos e gerar a resposta necessária. Desta forma, para esta versão do cachorro, existe a possibilidade de se usar oito sensores. Este número é suficiente para se implementar o estudo comportamental realizado. A idéia é distribuir os sensores de forma que o robô-cachorro tenha sensores na cabeça e no rabo e pelo menos um sensor para a região correspondente a cada pata.

Assim, o microcontrolador envia para o sistema central a informação de quais sensores estão sendo tocados, juntamente com o intervalo de tempo em que isto está ocorrendo. O sistema central não recebe a medida do tempo exato, mas sim um dado já pré-processado, com uma classificação do tempo dentro de intervalos estabelecidos, para apenas julgar seu significado de acordo com os modelos comportamentais. Para pequenos toques, os dados só são enviados para o sistema central quando o contato com o terminal cessa. Se a duração do toque for longa, no entanto, a informação de que está sendo tocado deve ser mandada antes do final do contato, para que o cachorro já reaja ao estímulo. Assim, a cada patamar de tempo atingido, o sistema central recebe uma informação de que o sensor continua sendo ativado.

3.3 SENSOR DE INTENSIDADE SONORA

A busca por um grau maior de interação entre o cachorro robô e o usuário leva ao desenvolvimento de diversas formas de sensoriamento. A audição é um sentido que permite um reconhecimento do ambiente bastante factível. Para certos animais, esse não é um sentido tão aguçado e desenvolvido quanto o faro, porém ele pode ser realizado, pelo menos em aproximação, por sistemas eletrônicos por meio de microfones. O grau de complexidade deste tipo de sistema depende daquilo que se objetiva, o que pode ir desde a simples detecção de níveis sonoros até mesmo ao reconhecimento de interlocutores.

Para este projeto, o objetivo deste módulo é fazer a captura de intensidades de áudio por meio de microfones instalados ao lado da cabeça do cachorro (como se fossem ouvidos) e, por meio das intensidades de som capturadas por cada microfone, determinar a direção do som e permitir que o robô responda a esse estímulo provocando um movimento da cabeça para a direção do som.

O módulo de audição é composto por um circuito de aquisição de áudio conjuntamente com um microcontrolador. O circuito eletrônico tem por função o tratamento do som capturado por microfones para que se tenha condições de interpretá-lo. A parte de programação do microcontrolador consiste na interpretação dos níveis sonoros adquiridos.

3.3.1 Arquitetura Eletrônica

Conforme dito anteriormente, o circuito eletrônico faz um tratamento do sinal de áudio. Este tratamento é constituído de filtragem, amplificação e detecção de envoltória. O circuito é apresentado na figura 3.7. A filtragem ocorre por um filtro passa-faixa de segunda ordem, com resposta em freqüência mostrado na figura 3.8, em que as curvas apresentadas se referem aos ganhos mínimo e máximo. As curvas são alteradas pela variação do valor do potenciômetro.

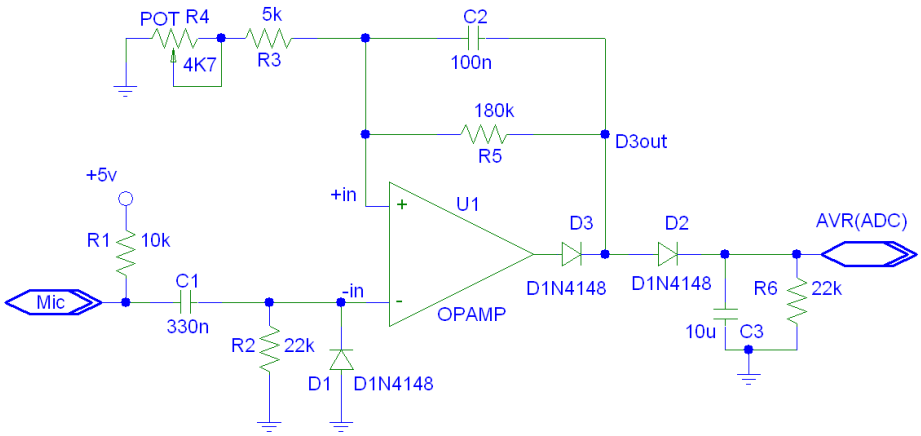


Figura 3.7: Circuito da captura de som.

O circuito inicialmente proposto tinha as mesmas características de filtragem, porém a constante de tempo do seu detector de envoltórias era bastante rápida, ou seja, para sinais de freqüência da ordem de centenas de Hz, ele funcionava apenas como um retificador de meia onda. Assim, fez-se necessário aumentar essa constante de tempo para uma faixa aproximada entre 200ms e 300ms. A determinação do valor para a nova constante de tempo se deu de maneira empírica, de forma que uma resposta lenta era desejada para que o sistema pudesse registrar a detecção do som. Entretanto, uma resposta lenta demais poderia causar a detecção repetida de um único evento sonoro.

Conforme já foi comentado, são usados dois microfones para a aquisição de áudio. Para tal, fez-se necessária a confecção de dois circuitos independentes. Previu-se também a possibilidade de um terceiro microfone instalado na parte anterior do corpo do cachorro. Este terceiro microfone possibilitaria que o reconhecimento de direção não se desse somente entre esquerda, direita e meio, mas também seria adicionada uma possibilidade de o som vir de trás do cachorro, que aumentaria a complexidade na reação do robô, que não teria mais somente que virar a cabeça, mas sim o corpo inteiro. Outra consideração feita com o objetivo de aumentar a robustez da detecção seria a instalação de mais um microfone. Claramente, quanto mais sensores, maior a robustez, mas também há um aumento considerável na complexidade da programação. Essa outra opção seria posicionar um microfone no topo da cabeça do robô para determinar se o som vem de cima dele, uma vez que a idéia é que o cachorro fique no chão. Por outro lado, essa seria uma redundância, visto que praticamente todo som provocado com a intenção de interagir com o cachorro viria de uma região mais alta que ele próprio.

Por fim, os ganhos dos dois circuitos são ajustados para que um mesmo som adquirido por ambos os microfones produzam uma resposta de mesma intensidade. É aceitável também que as respostas variem um pouco de um para outro circuito, devido à margem de variação nos valores dos componentes. Ajustes mais delicados podem ser feitos no programa que faz a interpretação dos dados no microcontrolador.

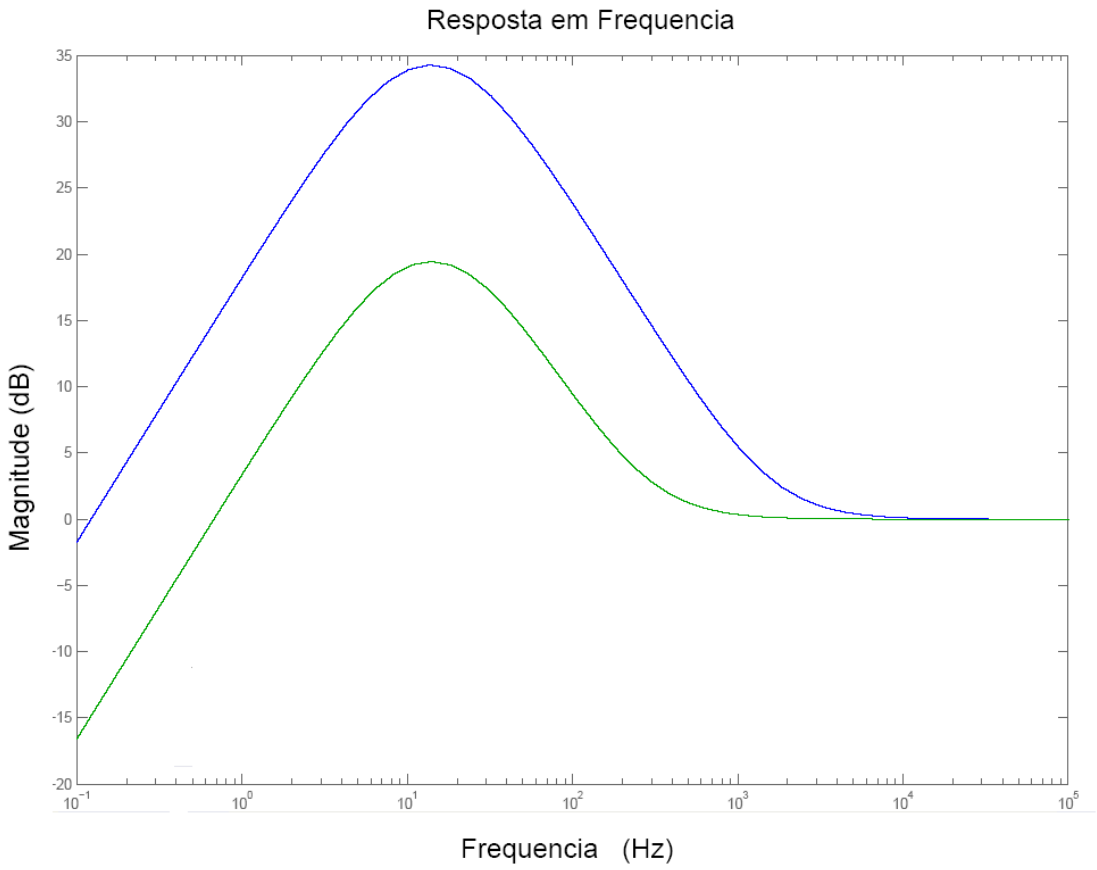


Figura 3.8: Diagrama de Bode do Ganho do Filtro.

3.3.2 O processamento digital no sensor de intensidade sonora

A idéia por trás do programa parcialmente desenvolvido para interpretar o som pré-processado tem uma estrutura relativamente simples. Os sinais de saída dos circuitos analógicos são capturados periodicamente pelo conversor A/D do microcontrolador em canais distintos. Os valores convertidos são então comparados. Se a diferença entre eles for menor que um determinado valor, é considerado que o som está vindo da frente, de cima ou de trás do robô. Se a diferença for maior que esse limiar fixo, o maior valor determinará a direção do som.

Apesar de a idéia ser simples, a sua realização requer um pouco de cuidado. O primeiro aspecto que deve ser verificado é se o ganho dos circuitos é realmente o mesmo e se os conversores estão funcionando de maneira correta de forma a registrar o mesmo valor digital para os dois canais. Para se fazer esta verificação, foi usado um mostrador de cristal líquido. Infelizmente os valores mostrados não eram exatamente verossímeis, então o ajuste dos limiares que interpretam a direção do som tiveram que ser feitos heuristicamente. Conseguiu-se até agora um resultado relativamente satisfatório, mas que ainda pode ser melhorado.

Uma dificuldade observada no processo de detecção da direção do som foi o problema de eco. Se o robô estiver a uma distância de uma superfície, como uma parede por exemplo, que reflita bem o som, esta reflexão fará com que um mesmo som provocado de um lado do cachorro seja interpretado com um certo atraso como proveniente do lado oposto. Para evitar esse efeito, é necessário que os intervalos de leitura do conversor do microcontrolador não sejam muito curtos. Mas isso não é um grande problema, pois o tempo de resposta do robô, para esse estímulo, não deve ser muito rápido.

Anteriormente a se resolver pela alteração do detector de envoltórias dos circuitos eletrônicos, pensou-

se na possibilidade de fazer filtragem digital dos dados, mas de forma que a aquisição seria feita com uma taxa de amostragem bem mais elevada. A filtragem teria característica de passa-baixas com o objetivo de se determinar o valor médio do sinal amostrado. Neste caso, a comparação entre os sinais de áudio adquiridos seria pelos seus valores médios, e não pelos valores de pico. O programa implementado não estava funcionando conforme o esperado, mas felizmente verificou-se que pelo ajuste da constante de tempo do detector de envoltória ter-se-ia um resultado mais interessante. Existe uma diferença considerável quando se leva em conta para a comparação os valores de pico ou os valores médios, isto porque uma diferença entre os valores de pico que pudesse vir a caracterizar a detecção lateral, poderia não provocar uma diferença considerável entre os valores médios. De qualquer forma, o uso de filtragem digital foi descartado.

3.4 REPRODUÇÃO DE ÁUDIO

Os diferentes tipos de sons emitidos pelo animal são uma das características observadas para o reconhecimento do estado emocional em que ele se encontra. Ao emitir um som, o animal fornece à pessoa que interage com ele uma resposta a respeito do efeito de suas ações sobre o animal. Baseada nesta informação, a pessoa decide se deve continuar com aquele estímulo ou interagir de outra forma para obter o resultado esperado. Desta forma, portanto, a emissão de sons é uma importante ferramenta de interação entre robô e treinador em robótica comportamental. Foi então previsto que o robô-cachorro tivesse um módulo para emissão de sons gravados, de acordo com os diferentes estados emocionais.

O módulo de reprodução de sons consiste em um microcontrolador conectado a um circuito de saída de áudio e um cartão de memória SD (Secure Digital) Card ou MMC (Multi Media Card) onde os sons estão gravados. O sistema central envia para este módulo uma mensagem indicando o estado emocional do robô e requisitando a emissão de som. O microcontrolador lê a região da memória em que um som referente ao estado indicado está armazenado e retransmite a informação recebida da memória para um conversor digital/analógico. A saída deste conversor D/A está conectada no circuito amplificador de áudio responsável pela geração do som.

Para gravar a memória sem nenhum tipo de formatação, foi criado um programa que lê um arquivo *wave* e retira dele apenas os *bytes* de dados, gravando-os diretamente na memória, por meio de uma conexão com um leitor de cartões conectado à porta USB. Decidiu-se por usar o cartão de memória MMC por ter maior facilidade de acesso que a memória SD.

A saída de áudio foi testada com a emissão de uma onda senoidal gerada no microcontrolador e está funcionando. Observando os canais de comunicação da memória (modo SPI) com o microcontrolador, também é possível perceber que o comando de leitura da memória é reconhecido pelo cartão e este retorna uma seqüência de bits do tamanho desejado como resposta. O sistema total, no entanto, não funcionou, possivelmente por problemas de temporização ou por algum erro na conversão de um modelo de transmissão para o outro, visto que na comunicação SPI com a memória os dados chegam em *bytes* e o envio para o conversor D/A deve ser feito em blocos de 4 bits de controle e 12 de dados.

3.4.1 A gravação da memória

A realização proposta foi de dividir a área de memória do cartão em blocos de 128kB em que estariam contidas informações relevantes de um determinado arquivo de áudio. As informações seriam subdivididas em dois setores, sendo um o cabeçalho e outro a parte contendo dados de áudio. O cabeçalho contém quatro linhas iniciais de texto, em que são informados o índice do bloco, o seu título, o nome do arquivo e a determinação do início dos dados. Essas informações seriam passadas ao cartão de memória por meio da escrita de um arquivo de listagem, com extensão .img. Na parte dos dados, tem-se que os primeiros quatro

bytes determinam o tamanho do trecho que contém unicamente dados de áudio. Desta forma, quando fosse necessária a leitura de um som específico, o microcontrolador faria a leitura diretamente do bloco desejado a partir da identificação do cabeçalho e faria a leitura dos dados de áudio, uma vez que é sabido o seu comprimento, em *bytes*. Abaixo vê-se o cabeçalho e o início do trecho de dados de um bloco de memória, em hexadecimal e sua interpretação lida por um editor de texto:

49 4E 44 49 43 45 20 3D 20 31 0D 0A 54 49 54 55 4C 4F 20 3D 20 4C 61 74 69 64 6F 20 31 0D 0A 4E 4F 4D 45 20 44 4F 20 41 52 51 55 49 56 4F 20 3D 20 6C 61 74 65 31 2E 77 61 76 20 0D 0A 44 41 44 4F 53 20 3D 20 0D 0A DB AC 00 00 7E 7F 80 80 81 80 7F 80 81 81 80 7E 7F 7F 7F 7E 7E 7F 80 80 81

Tabela 3.2: Intepretação do arquivo de listagem

49 4E 44 49 43 45 20 3D 20 31 0D 0A	INDICE = 1
54 49 54 55 4C 4F 20 3D 20 4C 61 74 69 64 6F 20 31 0D 0A	TITULO = Latido 1
4E 4F 4D 45 20 44 4F 20 41 52 51 55 49 56 4F 20 3D 20 6C 61 74 65 31 2E 77 61 76 20 0D 0A	NOME DO ARQUIVO = late1.wav
44 41 44 4F 53 20 3D 20 0D 0A	DADOS =
DB AC 00 00	44251 B

É importante ressaltar essa representação por editor de texto, pois a parte de dados apresenta uma série de caracteres incoerentes, mas essa é simplesmente a representação em código ASCII dos dados de áudio, bem como os primeiros quatro caracteres que determinam o tamanho do trecho de áudio.

O formato original dos arquivos de áudio eram do tipo *wave*, com extensão .wav, como pode ser visto no nome do arquivo apresentado no cabeçalho do bloco. Este tipo de arquivo foi escolhido por permitir ser manipulado com maior facilidade principalmente pelo fato de não ser compactado.

3.4.1.1 Estrutura do arquivo .wav

O arquivo *wave*, conforme previamente comentado, armazena informação "crua", ou seja, não há necessidade de qualquer tipo de pré-processamento dos dados armazenados no arquivo para que se possa extrair o áudio. Além disso, o arquivo *wave* suporta diferentes taxas de amostragem, resoluções de bits e canais de áudio. É necessário apenas que se conheça a estrutura e a organização do arquivo para se fazer a leitura corretamente.

O arquivo *wave* contém, no mínimo, três "pedaços" (do inglês *chunk*), chamados *riff*, *format* e *data*, e armazena os *bytes* no formato *little endian*. Cada pedaço tem uma função, armazena informações específicas e tem um estrutura padrão. Os primeiros quatro *bytes* do pedaço são o identificador, em código ASCII, os quatro *bytes* seguintes determinam o seu tamanho e em seguida vêm os dados específicos do pedaço.

O primeiro pedaço, *riff*, que é considerado por alguns como um cabeçalho do arquivo e não um pedaço propriamente dito, tem comprimento de 12 *bytes*, com o identificador "RIFF", seguido do tamanho em *bytes* do arquivo inteiro (eis o motivo de o *riff* não ser considerado pedaço ou então ser considerado pedaço e os demais trechos serem considerados sub-pedaços), e por fim o formato do arquivo apresentado pelo texto "WAVE", também em código ASCII, que indica *wave* como sendo o tipo de arquivo.

O segundo pedaço, *format*, tem o identificador "fmt", seguido do seu tamanho, do formato de áudio, número de canais, taxa de amostragem, taxa de bits, alinhamento de bloco e o número de bits por amostra. Mas o pedaço mais importante para este trabalho é o que contém os dados propriamente ditos, ou seja, o pedaço *data*. Este pedaço apresenta o identificador "data", seguido do seu tamanho e de todo o dado de áudio. Abaixo são mostrados os pedaços *riff* e *format* e o início do pedaço *data* de um arquivo de áudio, a título de exemplificação:

52 49 46 46 62 97 06 00 57 41 56 45 66 6D 74 20 12 00 00 00 01 00 02 00 80 BB 00 00 00 EE 02 00 04 00 10 00 64 61 74 71 30 97 06 00

Tabela 3.3: Intepretação do arquivo .wav

52 49 46 46	RIFF
62 97 06 00	431970 B
57 41 56 45	WAVE
66 6d 74 20	fmt
12 00 00 00	18 B
01 00	PCM
02 00	stereo
80 BB 00 00	48 kHz
00 EE 02 00	192 kB/s
04 00	
10 00	16 bits
64 61 74 71	data
30 97 06 00	431970 B

Foi definido que todos os arquivos de áudio a serem usados teriam as mesmas características padronizadas com taxa de amostragem de 22kHz e um canal mono de 8 bits, uma vez que não se faz necessário ter uma grande fidelidade de som. Com isso, foi possível se desconsiderar os dois pedaços iniciais do arquivo *wave* e tomar apenas a informação referente ao próprio som digitalizado.

3.4.1.2 Criação e transferência do arquivo de listagem

A partir dos arquivos *wave* e conhecendo sua estrutura, foi possível criar um programa para juntar em um único arquivo de listagem apenas os dados de áudio com os cabeçalhos particulares apresentados anteriormente, de forma que cada conjunto de dados com seu cabeçalho ocupasse um bloco distinto de $128kB$. A montagem do arquivo de listagem em si é bastante simples. Ele basicamente, para cada bloco de $128kB$ de memória, escreve o cabeçalho, em seguida verifica o tamanho do pedaço de dados do arquivo *wave* e copia esses dados para o bloco. Caso o cabeçalho juntamente com os dados ultrapassem $128kB$ o corre o truncamento do restante dos dados. Em seguida, passa para o próximo bloco.

De posse do arquivo de listagem, foi possível transferi-lo para o cartão de memória. Como o arquivo tem uma estrutura sem qualquer tipo de formatação, a escrita deveria ser feita em baixo nível, *byte* a *byte*. Para tal, foi usado o programa "*dd for Windows*" (doravante referido por DDfW), desenvolvido por John Newbigin, da *Swinburne University of Technology*, em Melbourne, Austrália, e disponibilizado livremente. Este programa traz as funcionalidades básicas do programa dd (*data definition*), desenvolvido para o sistema operacional Unix para a transferência de *bytes* ou blocos de *bytes* de informação não formatada de partições ou discos inteiros para trechos de memórias, sejam do próprio disco rígido, disco removível ou memória não volátil. O programa de Newbigin permite este tipo de transferência em Windows, uma vez que o próprio sistema operacional não provê maneiras de fazê-lo.

Durante a fase de leitura dos blocos escritos no cartão de memória, foram observados comportamentos inesperados. O programa DDfW foi usado tanto para a cópia para a memória quanto para a leitura dela. No processo de escrita, aparentemente ocorria tudo conforme esperado e o arquivo de listagem era passado na íntegra para ela. Entretanto, para a leitura, observaram-se anomalias. Percebeu-se que, após a leitura da memória, os blocos de 128kB eram alongados com *bytes* extras sem qualquer informação. Cada bloco acrescentava um número fixo de *bytes*, independentemente da sua posição no arquivo de listagem. Isso provavelmente ocorria em algum momento da execução do DDfW. Para se executar o programa DDfW, era necessário usar o *prompt* de comando ou a execução de um arquivo de extensão .bat. Observou-se que,

na execução pelo *prompt*, a extensão dos blocos ocorria apenas após o fechamento da janela do *prompt*. Infelizmente este problema não pôde ser investigado mais a fundo devido à decisão tomada de que a execução do áudio, pelo menos por ora, não mais se daria pela leitura do cartão de memória.

3.4.2 O cartão de memória MMC

Os cartões de memória não volátil são muito interessantes para o armazenamento de dados quando se necessita de dispositivo de tamanho reduzido, baixo consumo de energia e baixo custo. Além disso, os cartões MMC também operam bem sob larga variação de temperatura e vibração. Como o seu uso é previsto em permanência no cachorro, ele deve ter um bom funcionamento enquanto o cachorro se locomove.

As memórias MMC têm dois protocolos de comunicação: o modo MMC e o modo SPI (*Serial Peripheral Interface*). O uso do modo SPI é bastante vantajoso para este projeto por ser um protocolo suportado pelo microcontrolador usado, além de ser mais simples de ser usado e controlado que o modo MMC. Mesmo embora o modo MMC seja mais completo e robusto, o modo SPI supre todas as nossas necessidades e por isso foi adotado como protocolo de comunicação entre o cartão de memória e o microcontrolador [10].

O programa para a comunicação entre a memória e o microcontrolador foi todo desenvolvido baseado na estrutura do protocolo SPI. Assim como muitos outros projetos de engenharia, este programa usou bibliotecas previamente desenvolvidas e disponibilizadas gratuitamente. Obviamente essas bibliotecas tiveram de ser estudadas e conferidas para garantir um funcionamento correto de acordo com nossas necessidades e especificações. Essas bibliotecas definem funções básicas para operar a memória a partir do microcontrolador, como inicialização, escrita e leitura.

3.4.2.1 Comunicação pelo Protocolo SPI

A determinação do modo SPI é feita logo que o cartão é energizado e não pode ser alterado enquanto ele assim for mantido. Assim sendo, a comunicação entre o cartão e o microcontrolador será toda em *bytes*, ou seja, todas as palavras terão tamanho com múltiplos de 8 bits. Quatro são os sinais utilizados no canal de comunicação, sendo três deles do microcontrolador para o cartão e apenas um no sentido contrário. O microcontrolador enviará ao cartão de memória os sinais CS (*chip select*), CLK (*clock*) e DataIn (DI) e receberá o sinal DataOut (DO), sendo os dois últimos sinais de transferência de dados com sentidos referenciados à memória.

O modo SPI permite a escrita e a leitura tanto de um único bloco de 512 *bytes* (tamanho esse que pode ser alterado) ou bloco múltiplo. Como o nosso interesse apenas se refere à leitura, não será tratada aqui a comunicação para escrita. A leitura de bloco múltiplo é aparentemente mais apropriada para se realizar a tarefa desejada, entretanto foi necessário se trabalhar primeiro com a leitura de um bloco apenas e, com a suspensão deste segmento do projeto, não foi possível verificar a leitura por bloco múltiplo.

Para que o cartão de memória produza qualquer resposta, ele deve receber um comando do microcontrolador pelo canal DataIn. Entre esses comandos estão aqueles que determinam a inicialização da memória, a leitura e a escrita, o tamanho do bloco, a quantidade de blocos para operação de bloco múltiplo, entre outros. Cada comando é identificado por um índice de 6 bits e como toda palavra durante a comunicação será de 8 bits, o índice é precedido por um 0 e um 1 lógicos. Alguns comandos exigem um argumento, como por exemplo, no caso de o comando dado ser o de leitura, deve ser informado o endereço da leitura. O argumento do comando tem um total de 8 *bytes*. Em seguida vêm 7 bits de CRC (que pode ou não ser usado) e um bit 1. Essa seqüência de 6 *bytes* é chamada de Janela de Comando e sempre deve preceder qualquer operação com a memória. A partir desse momento o canal DataIn é mantido em nível lógico 1 e o microcontrolador deve aguardar uma resposta da memória.

Durante toda a Janela de Comando (*Command Frame*), o canal DataOut é mantido em nível lógico

1. Ele pode ser mantido neste estado por até mais 8 *bytes*, que geram o chamado Ncr, ou tempo de resposta ao comando, em que N seria a quantidade de *bytes* e cr vem do inglês *command response*. Ao fim deste período, a memória escreve no canal DataOut uma resposta R1, R1b ou R2. Essas respostas trazem determinadas informações antes que os dados sejam transmitidos. A resposta R1 tem um *byte* de comprimento, sendo o bit mais significativo igual a zero. Esta resposta traz *flags* contendo informações de erros, comando ilegal, *erase*, *reset* e estado de inicialização. A resposta R1b tem a mesma estrutura principal de R1 acrescida de um sinal opcional de ocupado. Já a resposta R2 é dada apenas quando é pedido o *status* do registrador e tem o mesmo *byte* de R1 acrescido de um outro *byte* com informações específicas do registrador. Para que a comunicação ocorra corretamente, é necessário que as respostas tenham valor zero. Finalmente, após o(s) byte(s) de resposta, a memória escreve no canal DataOut os dados seguidos de dois *bytes* de CRC. A figura abaixo mostra graficamente a parte inicial de cada trecho de comunicação desde o recebimento do comando até o envio da resposta.

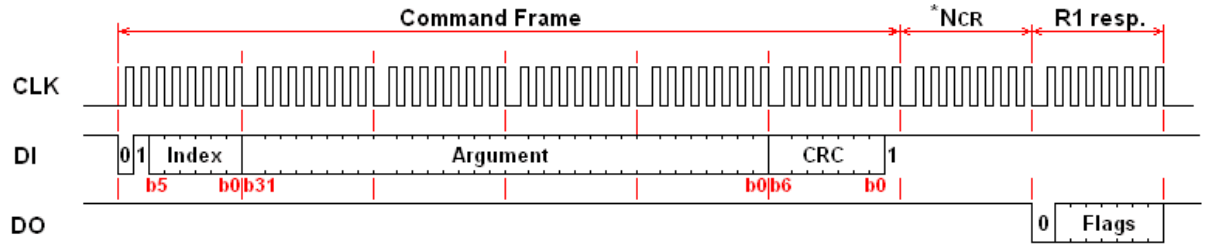


Figura 3.9: Temporização da comunicação SPI (Fonte: [3]).

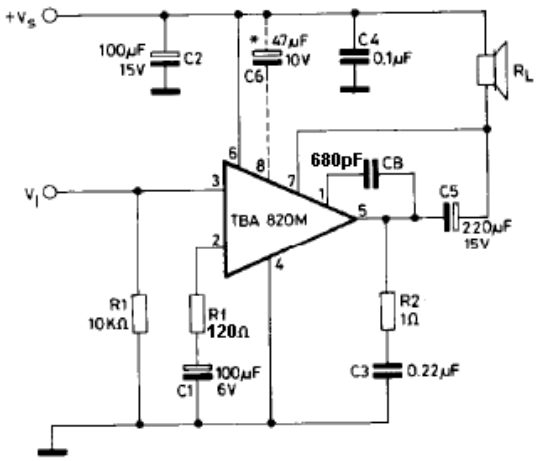
3.4.3 Arquitetura Eletrônica

O módulo do áudio pode ser dividido em duas partes: a ligação do cartão de memória ao microcontrolador e a saída de áudio, que consiste nas conexões do microcontrolador com o conversor digital/analógico e deste com o circuito analógico de amplificação do sinal e caixa de saída do som. O esquemático desses circuitos é apresentado na Fig. 3.10. Como este módulo não foi concluído, todos os testes foram realizados apenas em *protoboard*.

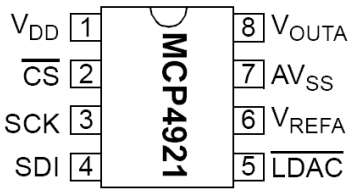
O conversor D/A utilizado foi o MCP4921 [4]. Este CI foi projetado para receber os dados a serem convertidos por comunicação SPI, unidirecionalmente, em comandos de escrita de 16 bits. O microcontrolador ATmega8 também possui interface para comunicação SPI, porém esta é feita *byte* a *byte*, o que tornou necessário fazer a comunicação entre os dois com a geração de todos os sinais por software em outras portas do microcontrolador. Isso deixou as portas da comunicação SPI do ATmega8 disponíveis para a comunicação com a memória MMC.

A tensão de referência do MCP4921 foi colocada em +5V, assim como a tensão V_{DD}, de modo que os sinais analógicos resultantes da conversão pudessem excursionar até este valor. A saída deste DAC foi ligada à entrada do amplificador de áudio TBA820M. As conexões deste amplificador foram feitas segundo o circuito sugerido pelo fabricante SGS-Thomson Microeletronics, conforme ilustrado na Fig. 3.10(a). No circuito, o resistor R_f e o capacitor C_B estavam em aberto, para serem escolhidos conforme as características desejadas. De acordo com os gráficos de resposta em freqüência fornecidos no *datasheet*, os valores $R_f = 120\Omega$ e $C_B = 680pF$ foram considerados os mais adequados, por resultarem em uma freqüência de corte de aproximadamente 5kHz. Este valor já é suficiente para uma boa comunicação, pois, apesar de o ser humano conseguir ouvir até freqüências próximas de 20kHz, o ouvido humano é mais sensível a freqüências de 1 a 3,5kHz. Os canais telefônicos, por exemplo, mantêm uma banda de até 4kHz para a comunicação.

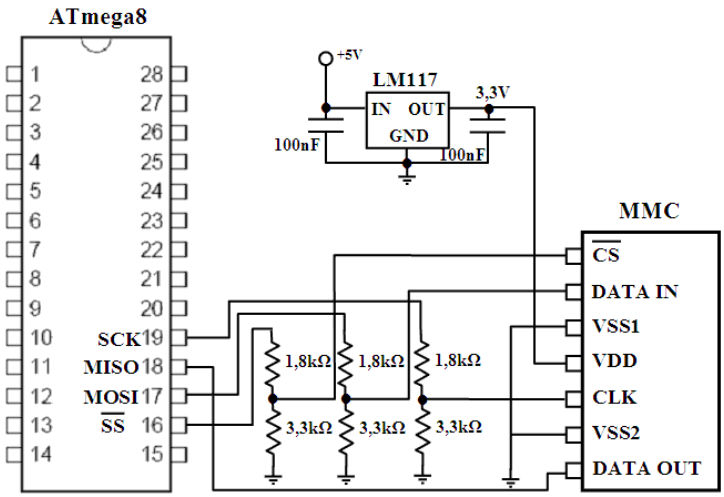
Quanto à conexão do cartão de memória com o microcontrolador, é necessário apenas ligar os pinos relativos à comunicação SPI do ATmega8 com os pinos correspondentes da memória por meio de um



(a) Circuito do amplificador de áudio (Fonte: [11]).



(b) Ligação do conversor D/A MCP4921 (Fonte: [4]).



(c) Ligação do cartão de memória ao microcontrolador.

Figura 3.10: Circuitos de saída de áudio.

divisor de tensão, uma vez que a memória é alimentada em 3,3V.

3.4.4 O processamento digital no módulo da fala

Assim como na parte eletrônica deste módulo, o processamento digital também pode ser dividido em duas partes: a leitura da memória e a transmissão dos sinais de áudio. É no microcontrolador, no entanto, que estas duas partes se comunicam e as informações da memória são passadas para a função de envio de dados ao conversor D/A.

Quanto à comunicação com o circuito de saída de áudio, o microcontrolador ATmega8 tem que gerar quatro sinais para a comunicação com o D/A MCP4921. Este recebe os dados pelo pino SDI, amostrando-os no flanco de subida do sinal de *clock* (CLK), também enviado pelo microcontrolador. Além destes sinais, é necessário gerar um sinal para habilitar o MCP4921 enquanto o comando de escrita estiver sendo enviado ($\overline{CS}$ - *Chip-select*). Há, por fim, um sinal para indicar final da transmissão dos dados e habilitar a saída do valor analógico correspondente ($\overline{LDAC}$ - *Latch DAC input*). A Fig. 3.11 mostra o diagrama temporal destes sinais durante o comando de escrita.

Em cada comando, a entrada SDI recebe 16 bits, porém apenas os 12 últimos são de informação, os

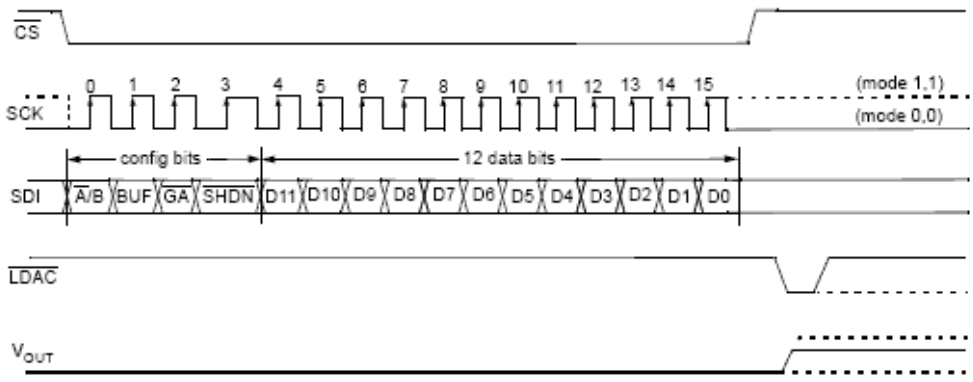


Figura 3.11: Diagrama temporal de um comando de escrita no DAC (Fonte: [4]).

outros são bits de configuração. O primeiro destes, $\overline{A}/B$, indica qual dos conversores está sendo usado em outro CI da mesma família, o MCP4922, que possui dois DACs. O segundo bit, BUF, é responsável pelo controle do *buffer* de entrada da tensão de referência. O valor padrão é *buffer* desligado, o qual permite uma maior faixa de tensões de entrada, porém se este bit for colocado em 1 a tensão passa a ser buferizada, o que torna a resistência de entrada muito mais alta. O próximo bit, $\overline{GA}$, seleciona se há ganho de tensão (fator multiplicativo para dobrá-la) ou não, ou seja, se $\overline{GA} = 0$, então $V_{OUT} = 2 * V_{REF} * dado/4096$. Por fim, o quarto e último bit de configuração, $\overline{SHDN}$, seleciona, em nível baixo, o modo de baixo consumo de potência. No programa desenvolvido para enviar os dados para este DAC, adotamos a configuração:

$\overline{A}/B = 0$ $\longrightarrow DAC_A$,
$BUF = 0$ $\longrightarrow$ sem *buffer* na tensão de referência,
$\overline{GA} = 1$ $\longrightarrow$ sem ganho e
$\overline{SHDN} = 1$ $\longrightarrow$ sem modo de baixo consumo de potência.

Com relação a esta parte do circuito de áudio, o programa do microcontrolador consiste essencialmente em uma função de inicialização, que habilita as portas como saída e define seus valores iniciais, e outra função para escrever no DAC, que recebe um *byte* com os dados, transforma-o em um dado de 12 bits, coloca os bits de controle e faz a transmissão, controlando as quatro linhas de comunicação de acordo com o diagrama apresentado na Fig. 3.11.

Para a comunicação com o cartão de memória, dois arquivos contendo funções já desenvolvidas foram utilizados [12]. O mais simples deles é responsável apenas pela inicialização da comunicação SPI e do envio e recebimento de informação por este protocolo. O outro trata da comunicação com a memória MMC. Ele traz funções de inicialização da comunicação do microcontrolador com a memória, escrita de dados na memória e leitura de dados em blocos simples ou múltiplos. O novo programa desenvolvido tem por objetivo fazer a leitura da memória usando as funções pré-existentes e fornecer esses dados ao sistema de reprodução de áudio.

3.5 PROTOCOLO DE COMUNICAÇÃO

Conforme colocado anteriormente, a parte eletrônica do robô é modularizada. Cada módulo é responsável por uma função específica e há um módulo central, que nesta primeira versão do robô-cachorro é um PC, onde é coordenada a operação de todos os outros, isto é, a comunicação se dá em uma configuração mestre/escravo.

Dentro deste contexto, o padrão RS485, por permitir grandes velocidades de transmissão com baixa interferência, demonstrou ser a melhor opção para a comunicação entre o módulo central e os diversos módulos periféricos. Aproveitando o *buffer* de três estados dos transceptores DS485 ou ST485, será usado um esquema de comunicação *half-duplex*, onde todos os módulos serão conectados no mesmo cabo de par trançado. A Figura 3.12 mostra um esquemático com as conexões do sistema. Futuramente o sistema central deve ser implementado em um processador Geode de 500MHz, componente principal de um PC embarcado, porém atualmente todo o sistema está sendo desenvolvido em um PC e a comunicação com este é feita pela porta serial, via padrão RS232. Para isto, será construído um conversor RS232/RS485, conforme proposto por Borges, *et al* [13].

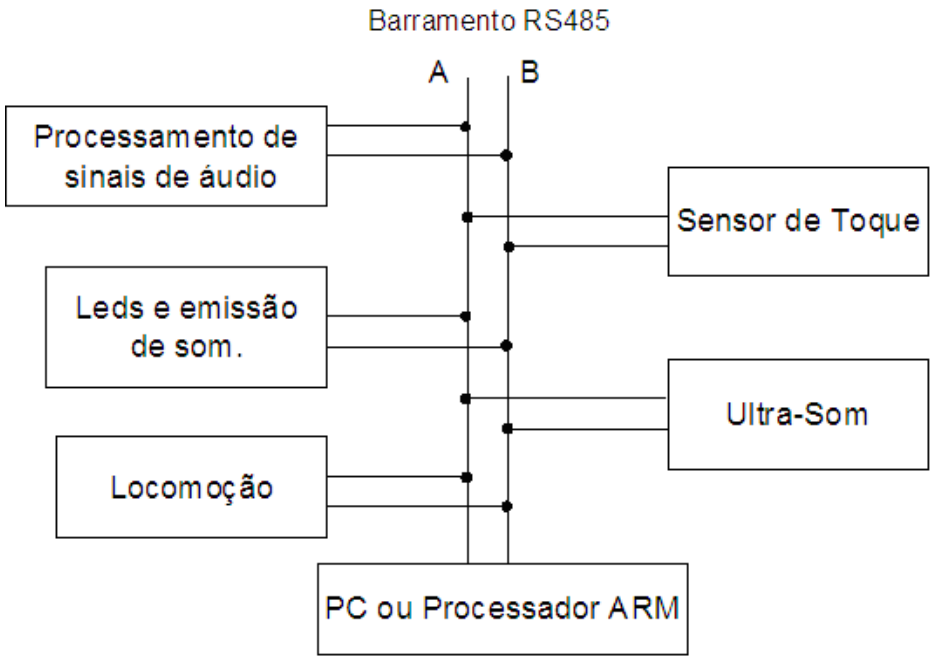


Figura 3.12: Conexão dos diversos módulos do robô-cachorro.

Para os módulos periféricos, a parte de comunicação é centrada na USART dos microcontroladores AVR. Foi criada uma biblioteca de funções que é responsável pela coordenação da comunicação. A estrutura de controle de entrada e saída é mostrada na Figura 3.13.

Buscando obter uma estrutura relativamente simples e que seja amigável ao usuário, a parte de comunicação foi dividida em dois blocos: Protocolo e Transmissão/Recepção. O bloco Protocolo é responsável por colocar os dados a serem transmitidos em um formato que seja facilmente interpretado pelo módulo mestre, e o módulo Transmissão/Recepção é responsável pela coordenação das filas de entrada e de saída e pelo controle da USART do AVR.

Todo o sistema de comunicação foi organizado baseado nas interrupções da USART. No momento em que for recebido algum dado, será ativada a interrupção de recepção completa da USART. Os dados recebidos serão armazenados na fila de entrada do sistema e o primeiro *byte* da mensagem será analisado pelo bloco Protocolo, identificando o tamanho total da mensagem. Enquanto todos os *bytes* de uma determinada mensagem não forem completamente processados pelo AVR, novos dados não poderão ser analisados pelo bloco de protocolo. Conseqüentemente, o microcontrolador irá executar as instruções do mestre de forma rápida e eficiente garantindo, porém, a atomicidade da operação.

A seguir, os fundamentos dos padrões de comunicação utilizados serão expostos e o protocolo de comunicação criado será detalhado. Posteriormente, o funcionamento das filas de entrada e de saída será explicado e a dinâmica das interrupções será mostrada com mais detalhes, especialmente a função de recepção.

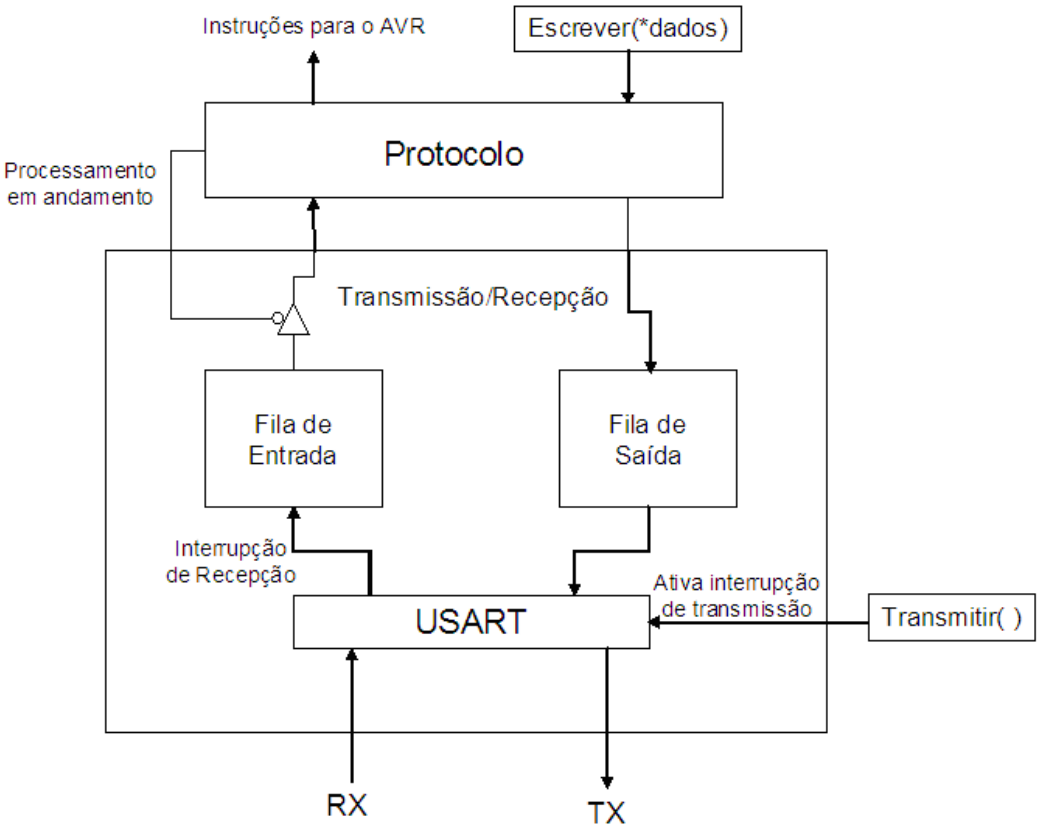


Figura 3.13: Sistema de comunicação.

3.5.1 Comunicação RS232 e RS485

Na comunicação serial, a transmissão de informação pode ser realizada de uma maneira síncrona ou assíncrona. Na comunicação serial síncrona, existe um sinal de relógio (*clock*) que marca o instante em que cada bit é disponibilizado no canal serial. Logo, na comunicação serial síncrona *full-duplex* entre dois dispositivos, serão necessários quatro fios: dois para o canal de transmissão, um para o relógio e um quarto para a referência das tensões. Já na comunicação serial assíncrona, não é utilizado um relógio para validar os bits de dados. Neste caso, para comunicação *full-duplex*, são necessários apenas três fios: dois para o canal de comunicação e um para a referência de tensões.

Normalmente, o sub-sistema responsável pela comunicação serial em computadores recebe o nome de UART (*Universal Asynchronous Receiver/Transmitter*) ou USART (*Universal Synchronous-Asynchronous Receiver/Transmitter*). Estes dispositivos são capazes de transmitir *bytes* de dados na forma de uma seqüência de bits. Como o próprio nome indica, dispositivos do tipo USART podem se comunicar de forma síncrona ou assíncrona, enquanto dispositivos do tipo UART comunicam-se apenas de forma assíncrona. O formato da seqüência de bits, também chamado de quadro, transmitido por uma UART ou USART pode seguir o esquema mostrado na Figura 3.14. Esta figura mostra o formato do quadro de dados da USART do ATmega8 [14], microcontrolador utilizado neste projeto.

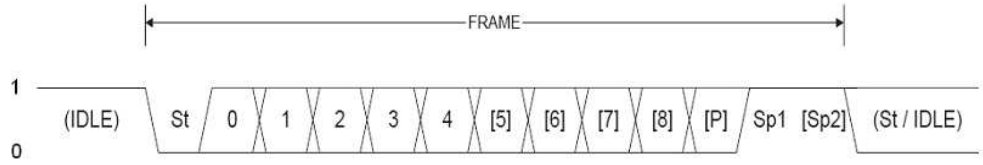


Figura 3.14: Formato do quadro de dados utilizado na USART do ATmega8 (Fonte: [5]).

Analisando a Figura 3.14, observa-se que enquanto não estiver ocorrendo comunicação o sistema estará em espera, estando, continuamente, no nível lógico "1". No momento em que a comunicação for iniciada, o canal de transmissão desce para o nível lógico "0". Este bit que marca o início da comunicação é chamado de "*start*". Após este bit, seguem de 4 a 8 bits de dados, dependendo da configuração da UART. Na seqüência de bits pode ser incluído, também, um bit de paridade para identificação de possíveis erros na transmissão. Finalmente, são utilizados um ou dois bits de parada ("*stop*"), no nível lógico "1", indicando o fim da transmissão do quadro.

Por estar compreendido entre 0V e 5V, utilizar níveis elétricos TTL na comunicação serial acabaria limitando a comunicação devido à interferência eletromagnética. Conseqüentemente, diversos padrões elétricos alternativos foram propostos. Dentro do escopo deste trabalho, dois padrões foram analisados: o RS232 e o RS485.

3.5.1.1 O padrão RS232

Largamente utilizado em PCs, o padrão RS232 trabalha com tensões mais elevadas do que as do padrão TTL. No caso, o nível alto (5V) TTL é convertido em -12V e o nível baixo TTL (0V) em +12V. Na recepção, qualquer sinal entre +3V e +12V é considerado como o nível lógico baixo, enquanto sinais entre -3V e -12V são considerados como nível lógico alto. Caso o sinal recebido esteja entre -3V e +3V, ele será considerado como indefinido. A Figura 3.15 criada por Martins, *et al*, [9] mostra graficamente os níveis elétricos do RS 232.

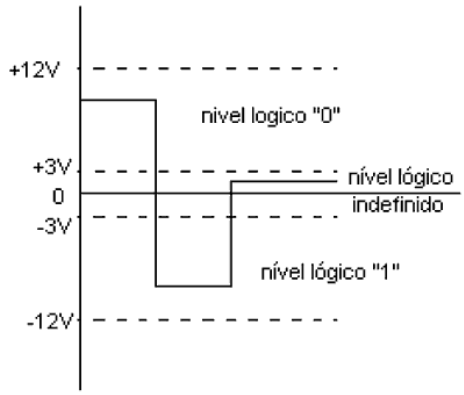


Figura 3.15: Tensões do RS232 (Fonte: [5]).

Taxas de transmissão maiores do que 20kbps são difíceis de alcançar usando o padrão RS232. Isto ocorre porque o sistema é altamente susceptível à interferência eletromagnética, prejudicando transmissões a distâncias médias e grandes. Além disso, por trabalhar com uma faixa relativamente alta de tensões (-12V a +12V), para alcançar taxas de transmissão altas, é necessária a utilização de drivers com alto *slewrate*.

3.5.1.2 O padrão RS485

O padrão RS485 adota uma estratégia diferente do padrão RS232 para minimizar os efeitos de interferências externas. Ao invés de utilizar um único sinal de transmissão de alta amplitude, o RS485 faz uso de um esquema de transmissão diferencial. Desta forma, dois sinais, A e B, são transmitidos e a informação é codificada na diferença entre estes sinais [A-B], conforme colocado por Borges, et al, [13].

Transceptores RS485 como o DS485 e o ST485, transformam um sinal TTL de entrada em dois sinais diferentes. O sinal A possui o mesmo nível lógico do sinal TTL da entrada, enquanto o sinal B é o seu complementar. Tanto o sinal A quanto o sinal B possuem amplitude entre 0V e 5V. Na recepção, é analisada

a diferença, A-B, entre estes dois sinais. Se esta diferença for maior do que 200mV, é considerado que o sinal recebido possui nível lógico "1". Por outro lado, se esta diferença for menor do que -200mV, é considerado que o sinal possui nível lógico "0". Se o sinal recebido estiver entre -200mV e 200mV, o nível lógico é indefinido. A Figura 3.16, originalmente usada por Randazzo [5], mostra os sinais diferenciais e os diagramas dos transceptores RS485 e DS232.

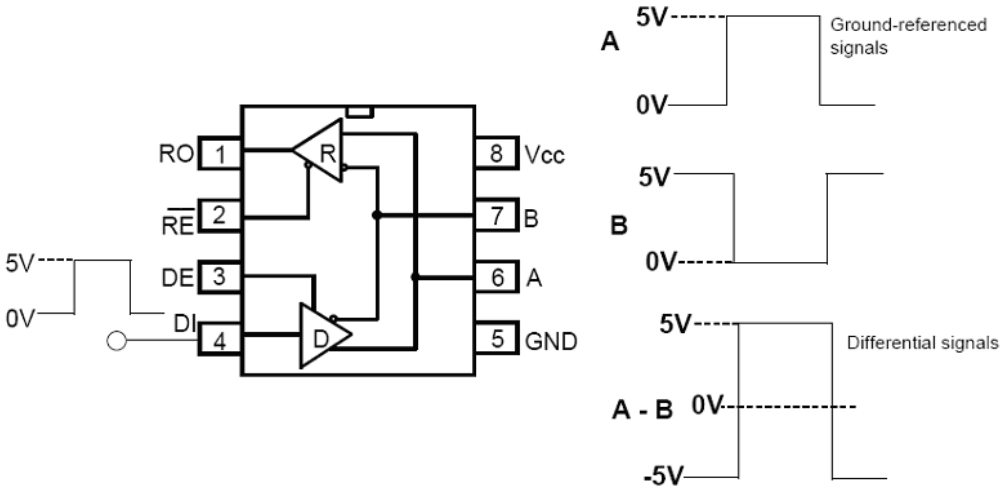


Figura 3.16: Esquemáticos dos transceptores DS485 e ST485, e sinais diferenciais (Fonte: [5]).

Desta forma, é possível alcançar taxas de transmissão de até 10Mbps a distâncias de até 1km utilizando este esquema de transmissão. Outra grande vantagem do padrão RS485 é a possibilidade de ligar até 32 dispositivos no mesmo cabo de par trançado, tornando este padrão adequado para sistemas de comunicação onde há um mestre e muitos escravos. Para a comunicação *full-duplex* devem ser utilizados dois cabos de par trançado e para comunicação *half-duplex* basta um cabo. Neste caso, devem ser utilizados transceptores com saídas *tri-state* para evitar que vários dispositivos tenham seu *buffer* de transmissão habilitado simultaneamente.

3.5.2 Formato da Mensagem

O protocolo utilizado tomou como ponto de partida o protocolo proposto por Martins, *et al*, [9]. São utilizados dois bytes de cabeçalho, seguidos pelos *bytes* de dados, cuja quantidade é determinada no próprio cabeçalho. Uma exceção, no entanto, é o caso em que o sistema central envia um pedido de dados para os microcontroladores. Como esta é uma função muito usada, uma vez que todos os módulos sensores têm que responder a este tipo de solicitação, foi criada uma forma de otimizar esta operação. O formato do protocolo utilizado é detalhado a seguir.

3.5.2.1 Formato do Cabeçalho

O cabeçalho das mensagens é formado por dois *bytes* que possuem a estrutura mostrada na Figura 3.17. O primeiro *byte* do cabeçalho contém informações sobre o endereço do módulo que vai receber a mensagem e três dos cinco bits que indicam a função a ser transmitida. Os outros dois bits da função encontram-se no segundo *byte* de cabeçalho, juntamente com o tamanho da mensagem.

O bit mais significativo de cada *byte* da mensagem indica se este é o início do cabeçalho ou não. Se o primeiro bit estiver em 1, então trata-se do primeiro *byte* do cabeçalho de uma nova mensagem. Isto foi criado pois, como o barramento é comum a todos os módulos, a cada *byte* enviado, todos os microcontroladores entram na interrupção de recepção e checam se é o início de um cabeçalho. Se for, os

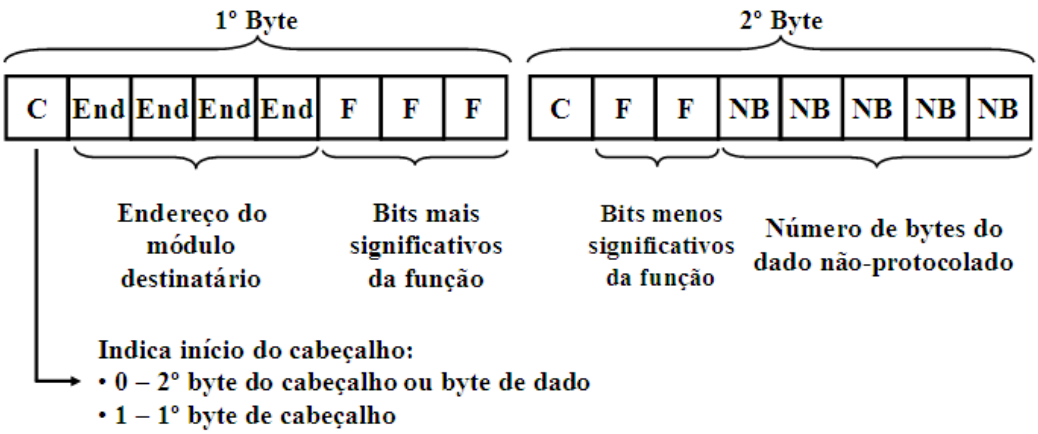


Figura 3.17: Significado de cada bit do cabeçalho.

módulos checam se a mensagem é para eles ou não e, no caso de ser, armazenam o resto dos *bytes* desta mensagem para tratar os dados ou executar a função que esteja sendo solicitada. Para que a identificação dos módulos seja possível, cada microcontrolador recebe um endereço em sua inicialização, que deve estar em conformidade com o valor presente na tabela de endereços do sistema central.

Um caso especial acontece quando o PC manda uma solicitação de envio de dados para os módulos sensores. Para otimizar este procedimento, criou-se uma maneira de, com apenas um *byte* recebido, os dados já serem retornados. Da forma como o protocolo foi criado, cada módulo pode ter até $2^5 = 32$ funções. Ocorre, no entanto, que nenhuma unidade vai usar tal quantidade de funções. Foi criada uma redundância e as funções do tipo $FFFFF = 111XX$, em que X representa qualquer valor lógico, foram escolhidas para significar envio de informações para o sistema central. Como os dois últimos bits da função não importam, é possível reconhecer esta seqüência apenas com o primeiro *byte* do cabeçalho e a resposta pode ser enviada mais rapidamente.

3.5.2.2 Formato dos Dados

O bit mais significativo de cada *byte* de dado deve ser 0, para não ser confundido com o início de um cabeçalho, então sobram apenas 7 bits de dados por transmissão. Como os dados são guardados em *bytes*, tanto no microcontrolador quanto no PC, antes de se enviar uma mensagem, os dados devem ser deslocados e escritos de 7 em 7 nas variáveis a serem passadas para o *buffer* de transmissão. Isto provoca um aumento do número de *bytes* a serem enviados em relação ao número de *bytes* do dado original. O número de *bytes* informado no cabeçalho, entretanto, informa o espaço que os dados ocupavam na memória antes de serem colocados no formato do protocolo.

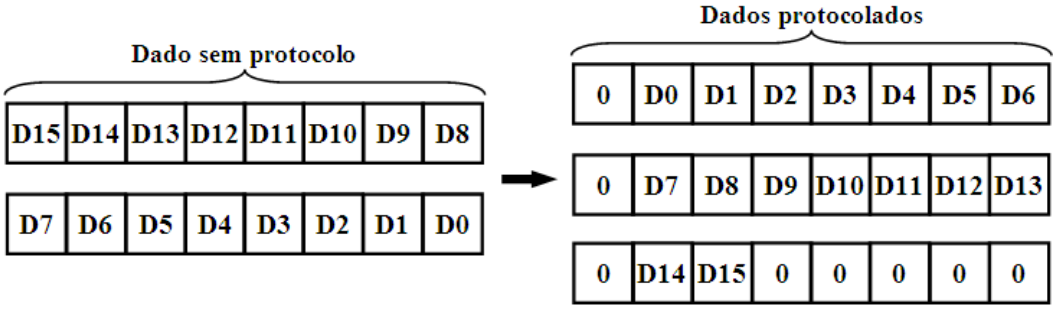


Figura 3.18: Exemplo de dado protocolado.

3.5.3 Transmissão e Recepção nos Microcontroladores

3.5.3.1 Filas de Entrada e Saída

As filas de entrada e de saída possuem 20 *bytes* de extensão, pois este é um valor adequado para o tamanho das mensagens que são utilizadas na comunicação entre os módulos. De forma a minimizar a ocupação de memória que cada fila possui, foram utilizadas filas circulares. Os formatos das filas são mostrados na Figura 3.19. A fila está cheia quando o ponteiro de escrita estiver na posição anterior ao ponteiro de leitura. Neste caso, o ponteiro de escrita será impedido de avançar, de forma que ele não possa sobrescrever o primeiro *byte* da mensagem, corrompendo, completamente, a informação recebida.

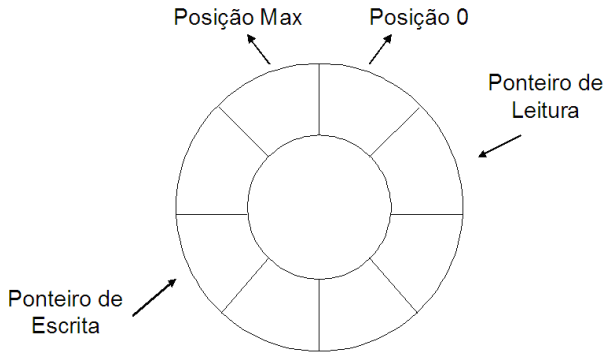


Figura 3.19: Fila Circular. No sistema de comunicação utilizado, Max = 20.

3.5.3.2 O processo de comunicação

Para realizar uma transmissão, existem duas etapas: primeiro os dados são colocados no formato do protocolo de comunicação e escritos na fila de saída, por meio da função *Write_TX()*. Os microcontroladores só enviam informação para o sistema central quando este requisita. Assim, cada *byte* que é recebido pelo microcontrolador provoca uma interrupção de recepção da USART e a função *Read_RX()* é chamada. Se o dado recebido corresponder à instrução de solicitação de dados, a função *Start_TX()* é executada, iniciando o processo de envio. Caso a informação recebida não seja uma instrução requisitando transmissão, ela é desprotocolada por meio de sucessivas chamadas à função *Read_RX()*, até todos os *bytes* daquela mensagem serem recebidos.

O envio de uma mensagem pelo microcontrolador é, então, iniciado com a chamada à *Start_TX()* dentro da interrupção de recepção. Esta função apenas faz o controle do barramento RS485, preparando-o para transmissão e, em seguida, habilita as interrupções de *buffer* vazio e de transmissão completa da USART. Com isso, toda vez que o *buffer* de saída esvaziar, o próximo *byte* é retirado da fila de saída e transmitido. Quando não houver mais nada a ser transmitido na fila, a interrupção de *buffer* vazio é desabilitada e ocorre a interrupção de transmissão completa para ajustar o pino de controle do barramento e o microcontrolador volta a ficar em modo de recepção.

4 COMPORTAMENTO

4.1 FUNDAMENTOS TEÓRICOS

O propósito deste capítulo é apresentar os conceitos fundamentais utilizados na elaboração do modelo comportamental do robô-cachorro. De acordo com o escopo deste trabalho, será dada maior ênfase à interpretação destes conceitos ao invés de tentar demonstrá-los rigorosamente. Sendo assim, será possível cobrir, com clareza, as ferramentas matemáticas e os métodos necessários para o desenvolvimento do modelo de aprendizagem.

A primeira parte deste capítulo irá definir o que são Cadeias de Markov, e explicar algumas de suas propriedades. Já a segunda parte irá abordar o problema da aprendizagem por reforço. Será exposto o que é aprendizagem por reforço, sua formulação matemática e será explicado o funcionamento do modelo de aprendizagem utilizado no robô-cachorro.

4.1.1 Cadeias de Markov

Antes de prosseguir, é conveniente definir o que é um processo estocástico. Conforme colocado por Ross [15], um processo estocástico $S(t), t \in T$ é uma coleção de variáveis aleatórias onde, para cada $t \in T$, $S(t)$ é uma variável aleatória. O termo T é denominado índice do processo e, caso pertença a um conjunto contável, o processo estocástico é dito de tempo discreto. Ao longo do texto, um processo estocástico discreto será representado como $\{S_k, k = 0, 1, 2...\}$, de modo que se $S_k = i$, o processo é dito estar no estado i no instante k.

Uma Cadeia de Markov é um tipo especial de processo estocástico discreto. Quando, no instante k, este tipo de processo estiver em um estado i, existe uma probabilidade P_{ij} de que o próximo estado será j. No caso de P_{ij} ser independente de k (tempo), então a cadeia é dita homogênea. Desta forma, por ser um processo Markoviano, temos:

$$Pr\{S_{k+1} = j | S_k = i, S_{k-1} = i_{k-1}, \ldots, S_0 = i_0\} = Pr\{S_{k+1} = j | S_k = i\} \tag{4.1}$$

Logo, a distribuição condicional de um dado estado futuro depende apenas do estado atual do processo. É importante ressaltar que todas as probabilidades serão sempre não negativas e que:

$$\sum_{j=0}^{\infty} P_{ij} = 1, i = 0, 1, \ldots \tag{4.2}$$

As probabilidades de transição entre um estado e outro para um único passo de tempo podem ser representadas por uma matriz de transição **P**, de forma que:

$$\mathbf{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \ldots \\ P_{10} & P_{11} & P_{22} & \ldots \\ \vdots & & & \\ P_{i0} & P_{i1} & P_{i2} & \ldots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{4.3}$$

A partir desta notação, definimos a probabilidade de um processo que está no estado i ir para o estado j após k transições como sendo P_{ij}^k. Seu valor pode ser obtido através da Equação de Chapman-Kolmogorov,

[2], que é:

$$P_{ij}^{k+m} = \sum_{a=0}^{\infty} P_{ia}^{k} P_{aj}^{m} \tag{4.4}$$

Conseqüentemente, a matriz de probabilidade de transição para k passos, representada por $\mathbf{P}^{(k)}$, será simplesmente o produto da matriz de transição $\mathbf{P}$ multiplicada por si mesma k vezes. Logo,

$$\mathbf{P}^{(k)} = \mathbf{P}^{k} \tag{4.5}$$

A partir destes resultados, já podemos observar como pode ser realizada a simulação da transição de estados de uma Cadeia de Markov. Primeiramente, considera-se um vetor linha $\boldsymbol{\pi}(k)^T$, com comprimento igual ao número de estados possíveis do processo, de modo que, um elemento π_i de $\boldsymbol{\pi}(k)^T$ representa a probabilidade de o estado no instante de tempo k ser igual a S_i. Se o processo for iniciado no estado S_j, o vetor $\boldsymbol{\pi}(0)^T$ será um vetor unitário com todas as posições nulas, exceto a posição j que será igual a 1. Desta forma, os vetores de probabilidades seguintes serão gerados recursivamente através da expressão:

$$\boldsymbol{\pi}(k+1)^T = \boldsymbol{\pi}(k)^T \mathbf{P} \tag{4.6}$$

É importante observar que $\boldsymbol{\pi}(k)^T$ não representa o estado do Processo Markoviano. A cada passo de simulação, o estado será um de i possíveis estados, de forma que $\boldsymbol{\pi}(k)^T$ representa as probabilidades de o processo estar em algum estado. Uma discussão mais aprofundada deste fato, além da comparação de uma Cadeia de Markov com um sistema dinâmico pode ser encontrada em [16]

Um último conceito importante dentro da teoria de Cadeias de Markov é a definição de Cadeias Ergódicas. Uma Cadeia de Markov será considerada *ergódica* se é possível, a partir de um estado qualquer ir para qualquer outro estado. Observe que esta transição pode ocorrer em diversos passos de simulação.

A seguir será mostrado um exemplo para ilustrar a teoria apresentada.

4.1.1.1 Exemplo

Um estudante de engenharia, em um dia qualquer, pode estar feliz (F), estressado (E) ou triste (T). Se hoje ele estiver feliz, as probabilidades de ele estar F, E ou T amanhã serão 0, 0,4 e 0,6, respectivamente. Já se ele estiver estressado, ele poderá ficar F, E ou T amanhã com probabilidades 0,2, 0,5 e 0,3. Por fim, se o estudante estiver triste hoje, as probabilidades de ele ficar F, E ou T amanhã serão 0,1, 0,4 e 0,5. Se hoje o estudante está feliz, qual é a probabilidade de ele estar feliz daqui a 3 dias?

Para resolver este problema, primeiramente determinamos a matriz de probabilidades de transição $\mathbf{P}$, que será dada por:

$$\mathbf{P} = \begin{array}{c} \\ F \\ E \\ T \end{array} \begin{array}{c} \begin{array}{ccc} F & E & T \end{array} \\ \begin{bmatrix} 0,0 & 0,4 & 0,6 \\ 0,2 & 0,5 & 0,3 \\ 0,1 & 0,4 & 0,5 \end{bmatrix} \end{array} \tag{4.7}$$

O diagrama de estados correspondente a esta cadeia é mostrado na Figura 4.1. Analisando a figura observa-se, claramente, que a cadeia é ergódica por ser possível transitar de um estado para qualquer outro em um número finito de passos.

Como o estudante está inicialmente feliz, o vetor de probabilidades $\boldsymbol{\pi}^T(k)$ no instante $k = 0$ será:

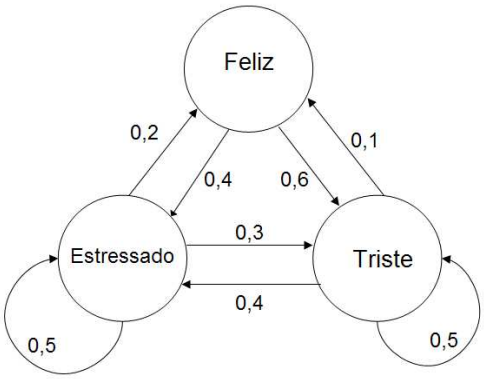


Figura 4.1: Cadeia de Markov do exemplo.

$$\boldsymbol{\pi}(0)^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{4.8}$$

O vetor de probabilidades para $k = 3$ poderá ser determinado utilizando a iteração mostrada na Equação 4.6.

$$\boldsymbol{\pi}(1)^T = \boldsymbol{\pi}(0)^T \begin{bmatrix} 0,0 & 0,4 & 0,6 \\ 0,2 & 0,5 & 0,3 \\ 0,1 & 0,4 & 0,5 \end{bmatrix}$$
$$\therefore \boldsymbol{\pi}(1)^T = \begin{bmatrix} 0 & 0,4 & 0,6 \end{bmatrix} \tag{4.9a}$$

$$\boldsymbol{\pi}(2)^T = \boldsymbol{\pi}(1)^T \begin{bmatrix} 0,0 & 0,4 & 0,6 \\ 0,2 & 0,5 & 0,3 \\ 0,1 & 0,4 & 0,5 \end{bmatrix}$$
$$\therefore \boldsymbol{\pi}(2)^T = \begin{bmatrix} 0,14 & 0,44 & 0,42 \end{bmatrix} \tag{4.9b}$$

$$\boldsymbol{\pi}(3)^T = \boldsymbol{\pi}(2)^T \begin{bmatrix} 0,0 & 0,4 & 0,6 \\ 0,2 & 0,5 & 0,3 \\ 0,1 & 0,4 & 0,5 \end{bmatrix}$$
$$\therefore \boldsymbol{\pi}(3)^T = \begin{bmatrix} 0,13 & 0,444 & 0,426 \end{bmatrix} \tag{4.9c}$$

Desta forma, a probabilidade de ele estar feliz após três dias será de 0,13. Observe que, baseado na equação de Chapman-Kolmogorov, o procedimento anterior é equivalente a fazer:

$$\boldsymbol{\pi}(3)^T = \boldsymbol{\pi}(0)^T \mathbf{P}^3$$
$$\boldsymbol{\pi}(3)^T = \boldsymbol{\pi}(0)^T \begin{bmatrix} 0,13 & 0,444 & 0,426 \\ 0,132 & 0,445 & 0,423 \\ 0,132 & 0,444 & 0,425 \end{bmatrix}$$
$$\boldsymbol{\pi}(3)^T = \begin{bmatrix} 0,13 & 0,444 & 0,426 \end{bmatrix} \tag{4.10}$$

Observa-se que o resultado é o mesmo.

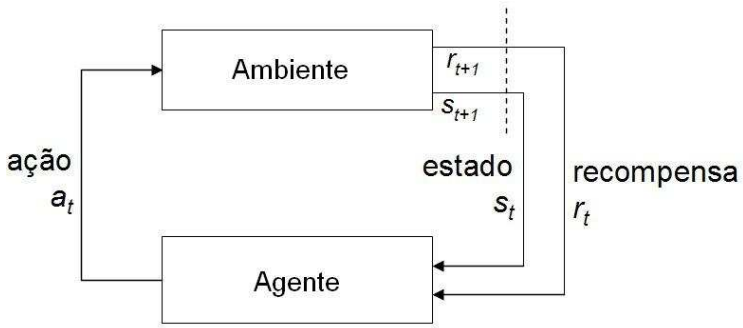


Figura 4.2: Interação de um agente com seu ambiente na aprendizagem por reforço.

4.1.2 Aprendizagem por Reforço

4.1.2.1 Agente Autônomo: Uma Definição

Uma definição fundamental encontrada em qualquer ramo da inteligência artificial e aprendizagem de máquina é o conceito de agente autônomo. Conforme colocado em [17], a definição do que é um agente não é fechada, de forma que o mundo não pode ser simplesmente dividido em agentes ou não agentes. Um agente é muito mais uma ferramenta para análise e descrição de sistemas do que propriamente um conceito. Dentro do escopo deste trabalho, o conceito de agente a ser utilizado será aquele proposto por Franklin, *et al*, [18] ao tentar fazer uma classificação dos diversos tipos de agentes existentes na literatura. Sendo assim, *um agente autônomo será considerado como um sistema situado dentro de um ambiente, que percebe este ambiente e que, ao longo do tempo, age sobre ele em busca de seus objetivos e para modificar o que ele percebe no futuro.*

4.1.2.2 Conceitos e Elementos Básicos Adicionais

Aprendizagem por reforço é um tipo de aprendizagem de máquina não supervisionado, onde o mapeamento entrada/saída é realizado através da interação contínua com o ambiente, visando maximizar (ou minimizar) um determinado índice de desempenho, normalmente chamado de *recompensa* [19]. Desta forma, não há nenhum "supervisor" externo que possa fornecer exemplos para o sistema e o agente deve aprender a partir da sua própria experiência. Este tipo de aprendizagem é muito útil para interações em tempo real de um agente com um ambiente desconhecido.

No modelo padrão de aprendizagem por reforço, [20], um agente está conectado ao seu ambiente através da percepção e ação. Em cada passo da interação, o agente recebe como entrada uma indicação do estado atual do ambiente. A partir desta indicação, ele escolhe uma ação de forma a gerar uma saída. Esta escolha, por sua vez, irá alterar o estado do ambiente, e o valor desta transição é comunicado para o agente na forma de uma recompensa. Dentro deste contexto, o agente deve escolher ações que tendam a maximizar, a longo prazo, a recompensa média recebida.

A Figura 4.2 descreve como funciona a interação entre um agente e o seu ambiente. Observe que esta interação ocorre em instantes discretos do tempo. Em cada instante de tempo, o agente recebe uma representação do estado do ambiente, $s_t \in S$, onde S é o conjunto de todos os estados possíveis, e opta por uma ação $a_t \in A(s_t)$, onde $A(s_t)$ corresponde ao conjunto de todas as ações possíveis em um estado $s_t \in S$. Feito isso, um passo de tempo depois, ele recebe uma recompensa $r_{t+1} \in R$ e encontra o ambiente em um novo estado s_{t+1}.

Os principais elementos da aprendizagem por reforço, além do agente e do seu ambiente, são: a política, a função de recompensa e a função de valor [21]:

- **Política**: define o comportamento do agente em um dado instante. Ela pode ser interpretada como o mapeamento dos estados percebidos com as ações que devem ser tomadas naqueles estados. A política determina o comportamento do agente e é, normalmente, estocástica. Deste modo, a política será denotada por $\pi_{t,s}$, onde $\pi_{t,s}(a)$ é a probabilidade de a ação escolhida no estado s, em um dado instante de tempo t, ser a.

- **Função de recompensa**: define o objetivo do problema de aprendizagem por reforço. Ela mapeia cada estado percebido do ambiente para uma única recompensa, indicando se aquele é um estado desejado ou não. Conforme colocado anteriormente, o único objetivo da aprendizagem por reforço é maximizar a recompensa recebida a longo prazo. Logo, a função de recompensa ajuda a identificar quais foram as melhores decisões tomadas pelo agente, podendo ser utilizada para atualizar a sua política.

- **Função de valor de um estado**: corresponde à recompensa total esperada que um agente pode acumular no futuro, caso comece em um determinado estado. Ao contrário da função de recompensa, que indica de forma imediata se um dado estado é desejado ou não, a função de valor indica, a longo prazo, se um estado é desejado ou não. Por exemplo, um estado pode sempre retornar uma recompensa pequena a curto prazo, mas possuir, mesmo assim, uma função de valor alta por ser seguida de outros estados que retornam recompensas altas.

Um conflito básico que surge na aprendizagem por reforço é o *trade-off* que surge entre a *exploração* e a *descoberta*. Apesar de estas palavras parecerem sinônimas, no contexto da aprendizagem por reforço elas surgem como operações contrárias. Para maximizar a quantidade de recompensa recebida, o agente deve optar por ações que, no passado, levaram a uma maior recompensa, *explorando-as*. Entretanto, para determinar quais são as melhores ações, i.e., aquelas que geram a maior recompensa, ele deve tentar ações que nunca utilizou antes, *descobrindo* novas ações que podem levar, ou não, a um maior retorno. Logo, na aprendizagem por reforço, deve ser buscada uma relação de compromisso onde não ocorram apenas operações de descoberta ou de exploração.

4.1.2.3 Retorno Esperado e Fator de Desconto

O próximo passo na discussão sobre aprendizagem por reforço é formalizar o conceito de retorno esperado a longo prazo. Denotando a seqüência definida à direita de recompensas recebidas T passos após um determinado instante de tempo t_0 por $r[t]$, onde $t_0 < t < T$, o retorno esperado será alguma função que atua sobre esta seqüência. Supondo um agente que interaja continuamente com o usuário, T irá para infinito, dificultando o cálculo do retorno esperado. Por exemplo, se o retorno esperado for simplesmente o somatório das recompensas recebidas, em um cenário de interação contínua, esta seqüência divergirá.

Sendo assim, é interessante inserir um termo adicional no cálculo de retorno esperado, limitando o seu valor. Surge assim o conceito de desconto, [21], onde o agente busca selecionar ações de forma que a soma das recompensas descontadas que ele receberá no futuro será máxima. Mais especificamente, sendo γ o fator de desconto e R_t a função de recompensa, o agente deverá selecionar uma ação a_t de forma a maximizar a expressão:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4.11}$$

Considerando que $0 < \gamma < 1$ e supondo que a recompensa recebida é limitada, fica claro que esta expressão converge. Um fator de desconto próximo de um significa que o agente irá maximizar recompensas imediatas, pouco se importando com as recompensas a longo prazo. Já um fator de desconto próximo de zero faz justamente o contrário e as recompensas a longo prazo são maximizadas.

4.1.2.4 O Processo de Decisão Markoviano

Neste trabalho será considerado que o estado do ambiente recebido pelo agente e o conjunto de ações que este pode tomar possuem toda a informação necessária para prever qual o estado e a recompensa recebidos no próximo instante de tempo. Este fato pode ser descrito pela expressão:

$$Pr\left\{s_{t+1} = j, r_{t+1} = r | s_t = i, a_t = a\right\} = P_{ij}(a) \tag{4.12}$$

Fica clara a relação entre a expressão anterior e a Equação 4.1. Sendo assim, observa-se que o problema da aprendizagem por reforço possui a propriedade de Markov, descrita na Seção 4.1.1. Além disso, ele possui reforço atrasado, de modo que a recompensa é obtida apenas no instante de tempo seguinte à decisão. Sistemas com estas características são chamados de Processos de Decisão Markovianos. Uma análise aprofundada destes tipos de processo pode ser encontrada em [22]. Neste modelo, a probabilidade de transição de um estado i para um estado j, dada uma determinada ação a, é denotado por:

$$P_{ij}^{a} = Pr\left\{s_{t+1} = j | s_t = i, a_t = a\right\} \tag{4.13}$$

De forma semelhante, dado um estado atual i, um estado seguinte j e uma determinada ação a, o valor esperado da próxima recompensa será:

$$R_{ij}^{a} = E\left\{r_{t+1} | s_t = i, a_t = a, s_{t+1} = j\right\} \tag{4.14}$$

4.1.2.5 Funções de Valor

As Funções de Valor são utilizadas em algoritmos de aprendizagem por reforço para determinar quão desejado é que o agente esteja em um determinado estado. Em outras palavras, a Função de Valor determina qual o retorno esperado que o agente receberá por estar em um estado. Logo, considerando uma política π e que o retorno esperado é calculado utilizando um fator de desconto, temos:

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} \tag{4.15}$$

Uma política é considerada melhor que outra se o seu retorno médio, ou seja, sua Função de Valor, for maior. Logo, a função de valor da política ótima, denotada por V^*, será dada por:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{4.16}$$

Desenvolvendo essas equações, obtemos:

$$V^*(i) = \max_{a \in A(s)} \sum_{j} P_{ij}^{a}\left[P_{ij}^{a} + \gamma V^*(j)\right] \tag{4.17}$$

Esta equação é conhecida como a Equação de Otimalidade de Bellman [23]. Observa-se que a política ótima deve maximizar tanto o retorno recebido a longo prazo quanto as funções de valor dos próximos estados a serem alcançados, considerando um termo de desconto.

4.1.2.6 Aprendizagem Temporal-Diferencial

Neste trabalho o agente irá aprender mediante uma arquitetura Temporal-Diferencial (TD), mais especificamente, através de um método conhecido como "Ator-Crítico". Este é um dos algoritmos mais utilizados em robótica comportamental e para a simulação de agentes com emoções, mas possui aplicações em diversas áreas como, por exemplo, finanças, [24], e medicina, [25].

Na aprendizagem TD, o conhecimento prévio, i.e., a experiência do agente, é utilizado para determinar qual decisão ele deve tomar. Neste sentido, após visitar um estado $s_t = i$ em um dado instante t, a sua estimativa $V(s_t)$ é atualizada com base na recompensa recebida. A atualização da função de valor pode ocorrer a cada instante de tempo, não necessitando de uma estimativa confiável da função de retorno. O algoritmo utilizado para atualizar $V(s)$ é:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right], \tag{4.18}$$

onde γ é o fator de desconto e α é uma constante corresponde ao tamanho do passo. Para mostrar que esta expressão é uma estimativa de $V^{\pi}(s)$, expandimos a Equação 4.15:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\} \\ &= E_{\pi} \left\{ r_{t+1} + \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\} \\ &= E_{\pi} \left\{ r_{t+1} + \gamma V^{pi}(s_{t+1}) \middle| s_t = s \right\} \end{aligned} \tag{4.19}$$

Logo, ao atualizar a função de valor, é feita uma aproximação da equação acima, por utilizar V_t e não V^{π} real. Esta aproximação é utilizada como alvo na aprendizagem TD, de forma que V_t é atualizado a partir da sua diferença com a aproximação de V^{π}. É muito interessante observar que os métodos TD aprendem suas estimativas a partir de suas próximas estimativas, ou seja, eles aprendem um "chute" baseado no "chute" anterior. Entretanto, se o sistema mantiver uma política constante π, este algoritmo irá convergir para a V^{π} com probabilidade 1, caso o passo de tempo seja pequeno o suficiente, conforme mostrado em [26]. A aprendizagem por reforço que utiliza este tipo de atualização da função de valor chama-se TD(0) e todos os outros métodos de aprendizagem TD são baseados nele.

Uma das grades vantagens deste tipo de aprendizagem é que ela não necessita de nenhum modelo do ambiente. Além disso, ela é implementada de modo puramente incremental, necessitando de pouca memória e convergindo rapidamente. Desta forma, este método mostra ser muito superior a outros, como programação dinâmica e simulação de Monte-Carlo.

4.1.2.7 Métodos Ator-Crítico

Os Métodos Ator-Crítico são uma forma de aprendizagem TD que possuei uma estrutura de memória separada para representar a política de forma independente da função de valor. Esta estrutura possuidora da política e, portanto, responsável por tomar as decisões é chamada de Ator. Já a estrutura responsável por atualizar e estimar a função de valor é chamada de Crítico, por analisar a qualidade das decisões tomadas pelo ator. O aprendizado, neste caso, altera a política adotada, de forma que o crítico "aprende" e "crítica" a política utilizada pelo ator, alterando-a [21]. Esta crítica assume a forma de um erro TD, de forma semelhante ao TD(0), sendo calculado como na Equação 4.18. Conseqüentemente, o erro TD será:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4.20}$$

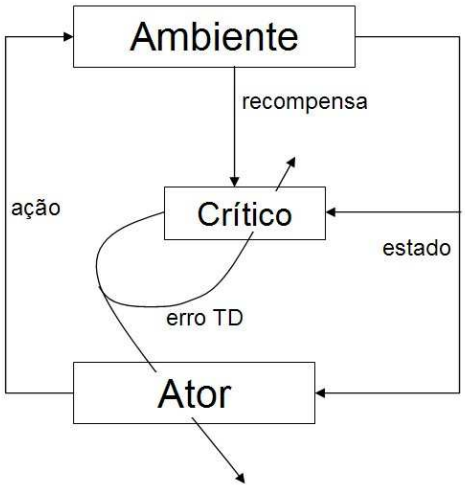


Figura 4.3: Método Ator-Crítico para aprendizagem por reforço.

Neste caso, $V(s_t)$ representa a Função de Valor atual considerada pelo Crítico. Após cada decisão tomada pelo Ator, o Crítico avalia o novo estado do ambiente, de forma a determinar se a decisão tomada foi adequada ou não. Por exemplo, se o erro TD para um dado estado s_t e ação a_t for positivo, a tendência do agente por tomar esta ação deve ser aumentada, de forma que ele escolha a_t mais vezes no futuro.

Para cada política, o Crítico manterá funções de valor diferentes, de forma a manter coerência e permitir que políticas diferentes possam ser utilizadas para diferentes estados do ambiente. Além disso, o crítico atualiza, após cada iteração, a função de valor daquele estado de acordo com a equação 4.18. Esta forma de aprendizagem é representada esquematicamente na Figura 4.3.

Para mostrar uma possível aplicação deste método, vamos supor que as ações tomadas sejam determinadas por uma Cadeia de Markov. Neste caso, a política $\pi_t(s,a)$ será a distribuição determinada por uma Cadeia de Markov cuja matriz de transição é dada por **P**, de forma que:

$$\boldsymbol{\pi}_t(s,a)^T = \boldsymbol{\pi}_t(s,a)^T\mathbf{P} \tag{4.21}$$

Os termos P_{ia} desta cadeia irão representar a probabilidade de este sistema optar por uma ação a ao estar no estado i. Conseqüentemente, para aumentar ou diminuir a tendência do Ator tomar uma decisão a, basta alterar P_{ia} fazendo, por exemplo:

$$P_{sa} \leftarrow P_{sa} + \beta\delta_t \tag{4.22}$$

onde β é um termo heurístico indicando o tamanho do passo. De forma análoga, a Função de Valor do estado será atualizada utilizando a expressão:

$$V(s) \leftarrow V(s) + \alpha\delta_t \tag{4.23}$$

onde α é um fator que indica o tamanho do passo e é determinado heuristicamente.

Ao analisar esta metodologia de atualização surgem, imediatamente, questões relacionadas à convergência deste método. Entretanto, os Modelos Ator-Crítico foi discutida por Konda, *et al*, [24] que demonstrou a sua convergência. Além disso, os efeitos da alteração da matriz de transição de uma Cadeia de Markov sobre a sua distribuição foram analisados por Meyer, *et al*, [27], indicando que, caso a variação da matriz seja relativamente pequena, a distribuição resultante se adaptará rapidamente. Sendo assim,

ao longo deste trabalho, será utilizado um fator β relativamente pequeno ($< 0,2$), de forma a garantir convergência.

4.2 MÓDULO DE PERCEPÇÃO

A função do módulo de percepção será transformar os dados medidos pelos diversos sensores do cachorro em informação relevante para o seu comportamento. Neste módulo, será determinado que tipo de sinal o usuário está enviando para o robô, buscando determinar o nível de atenção do usuário e o estímulo que está sendo recebido. É importante observar que, ao tentar interpretar as informações advindas dos sensores, o robô está tentado definir a intenção e o estado do usuário, o que pode ser relacionado com uma tentativa de tentar construir um modelo do mundo a sua volta. A Tabela 4.1 mostra qual função do módulo de percepção é influenciada pelas informações de cada sensor.

Tabela 4.1: Dados dos sensores para funções do módulo de percepção

		Ultra-som	**Toque**	**Áudio**	**Vídeo**
Funções dos Sensores		-Determinar distância -Calcular a média de 5 medições	-Medir duração do toque -Quais sensores são tocados	-Intensidade do som de cada microfone -Direção do som -Intensidade média	-Localizar faces -Nível de luminosidade
Dados enviados do sensor para o PC		-Distância(mm)	-Faixa de tempo -Qual sensor	-Direção do som -Faixa de intensidade média (baixo, médio, alto)	-Há face ou não -Claro ou escuro
Saídas do módulo de percepção	**Nível de atenção**		X	X	X
	Carinho/Agressão		X		
	Claro/Escuro				X
	Som alto/Som baixo			X	
	Presença/Ausência		X		X
Reflexos	**Movimento**	X		X	

4.2.1 Entradas do Módulo

As entradas do módulo de percepção são aquelas informações enviadas pelos sensores. Essas informações são simples o suficiente de forma que sua interpretação seja a mais direta possível. Isso se daria por uma transição direta de interpretação entre informação de entrada e saída do módulo e é isso que a Tabela 4.1 determina. O sensor de toque deve retornar para o módulo qual sensor foi tocado e durante quanto tempo, conforme já foi mencionado em 3.2.2. A faixa de tempo indicada na tabela indica justamente a duração do toque. Por simplicidade e não se haver a necessidade de medir a duração do toque com uma resolução muito alta, decidiu-se por determinar apenas se o toque é continuado durante determinados limiares temporais. O sensor de intensidade de som deve enviar ao módulo a informação de se o som é proveniente do lado direito, do meio ou do lado esquerdo do robô. Se vindo do meio, o som pode ser frontal ou traseiro. A intensidade desse som também é passada ao módulo, sendo dividia em três níveis. O vídeo é responsável pela determinação de presença de pessoa(s) pela detecção de face, bem como determinação da luminosidade ambiente, dividia em dois níveis (claro e escuro). Por fim, o sensor de ultra-som envia a distância do robô a um obstáculo, de forma que, por meio de medições consecutivas, é possível determinar a aproximação ou o distanciamento entre os dois.

4.2.2 Carinho-Agressão

Conforme pode ser observado na tabela 4.1, o módulo de percepção determina se o cachorro está recebendo um estímulo de carinho ou agressão a partir da informação enviada pelo sensor de toque. A idéia da temporização do toque serve justamente para diferenciar o tipo de estímulo. O carinho é um estímulo caracterizado pela suavidade enquanto que a agressão é observada por atitudes súbitas. Outra forma possivelmente mais apropriada de se determinar essa distinção seria pela implementação de sensores de pressão, pois atos agressivos têm a tendência de ser mais fortes e imprimir maior força no objeto da ação equanto atos de carinho são mais fracos. Como não se tinha esse tipo de sensor, o tempo de ativação do sensor de toque pode substituí-lo com certa verossimilhança. Assim, se o sensor for percebido ativo por um curto espaço de tempo, o módulo responderá com um estímulo de agressão, enquanto que um tempo mais demorado determinará um carinho.

Claramente este tipo de reconhecimento exige uma certa flexibilidade e adaptação do usuário no que diz respeito a seus atos. Quando as pessoas interagem com cachorros reais, gestos agressivos podem até realizar um contato físico mais longo, como dar um tapa e permanecer a mão sobre o animal ou até mesmo dar um beliscão. Porém, esse tipo de coisa não é razoável de ser feita com um robô, primeiramente porque dar um tapa provavelmente machucará a mão do usuário e, em segundo lugar, não é possível dar beliscões em chapas de metal, pedaços de plástico rígido ou acrílico. Assim, a ação do usuário deve ser regida pela maneira como o robô a interpreta e não o contrário.

4.2.3 Som Alto - Som Baixo

Claramente, a intensidade sonora só poderá ser determinada pela captura do sensor que tem justamente essa finalidade. Independentemente da direção de onde vem o som, a sua intensidade será registrada. Não se fez qualquer julgamento sobre a duração do som, pois essa ainda não é a intenção, visto que as ferramentas usadas não permitem tal distinção. Ou seja, sons sobrepostos poderiam de certa forma "mascarar" esse tipo de detecção e levar a interpretações errôneas. A intenção é somente determinar o volume do som e permitir que o cachorro reaja a esse estímulo. Independentemente de o som ser gerado por um usúario interessado em interagir com o robô, ele será detectado pelos sensores e isso provocará uma reação.

4.2.4 Presença-Ausência e Ausência-Presença

O par presença-ausência e ausência-presença consiste na alteração de estado de solidão do robô. É importante ressaltar que não se está verificando se existe alguém em interação com o cachorro ou não, mas sim se o usuário passou ou deixou de interagir, isto porque as outras funções do módulo já determinam a continuidade de interações, enquanto que esta determina o início ou término desses contatos. Resolveu-se por fazer essa identificação apenas por toque e visão porque, no caso de intensidade sonora, a captura pode estar ocorrendo de algum ruído que não proveniente de usuário, enquanto que os outros dois considerados podem apenas ser gerados por usuários que tenham realmente interesse em interagir com o cachorro. Como cada função do módulo de percepção tem um objetivo próprio, esta função deve verificar a intenção do usuário em interagir ou não com o robô. A maneira como ele interage ou como o robô percebe o ambiente a sua volta, é determinado pelas demais funções.

4.2.5 Luminosidade e reflexos

Todo animal reage de uma determinada maneira aos níveis de intensidade luminosa do ambiente. Com o robô-cachorro não deve ser diferente e ele deve ser capaz de reconhecer diferentes níveis. No entanto, apenas dois níveis, claro e escuro, são necessários para as reações previstas. Claramente, este tipo de

percepção só pode ser averiguado por meio da câmera de vídeo.

O nível de proximidade pode apenas ser detectado pelo sensor de ultra-som. Como a tabela 4.1 mostra, esta informação não é uma saída do módulo de percepção para determinar um comportamento do robô baseado em seu estado emocional, mas sim uma reação puramente mecânica (reflexo) de forma a evitar choque. Semelhantemente, o robô responde à detecção de som. Enquanto que a intensidade sonora determina um nível de interação, a direção do som deve apenas determinar um reflexo do cachorro para que ele tente se voltar para o seu "interlocutor". Se for arbitrada capacidade auditiva bilateral equilibrada, a cabeça do cachorro deverá se virar de forma a tentar posicionar a fonte do som à sua frente. Esta é a importância de se ter mais um microfone na parte traseira do robô, que mesmo não correspondendo à realidade em termos de audição animal, seria uma boa aproximação para a tarefa realizada pelas orelhas, que se movem de forma a tentar determinar mais precisamente a direção da fonte sonora. Desta forma, seria possível verificar, caso o som viesse de uma região considerada de meio, se ele viria da frente ou de trás e assim promover uma possível reação reflexiva de virar não somente a cabeça, mas o corpo inteiro.

4.2.6 Nível de Atenção do Usuário

Para tentar determinar o nível de atenção do usuário é necessário, primeiramente, interpretar os dados recebidos medidos nos sensores. Por exemplo, caso seja detectado que há algum tipo de toque contínuo e leve nas costas do cachorro e, além disso, seja observado que há um rosto no campo de visão do animal, ele deverá entender que há um usuário dando-lhe atenção. Entretanto, caso ele sinta toque fortes nas suas costas e barulhos muito altos, ele deverá interpretar estes dados como alguma forma de agressão. Logo, o nível de atenção do usuário será normalizado de forma que ela possa ser quantificada entre os valors -1 e 1. Um nível de atenção próxima de 1 indica que o agente está recebendo atenção positiva, no caso, carinho ou algum outro estímulo não agressivo. Um nível de atenção próximo de zero mostra que o robô não está recebendo nenhum tipo de estímulo, enquanto um nível de atenção negativo indica que o estímulo recebido é agressivo como, por exemplo, um grito ou um empurrão.

O nível de atenção do usuário dependerá dos dados recebidos dos sensores de toque, do sensor de áudio e das informações da câmara de vídeo presentes no robô. Mais especificamente, o nível de atenção será determinado a partir dos sinais:

- Carinho/Agressão (CA): será 1 se o usuário estiver acariciando o cachorro ou realizando toques leves, 0 se não houver nenhum tipo de toque e -1 se houver alguma forma de agressão;
- Nível sonoro (NS): se o estímulo sonoro for muito alto, será -1. Caso o estímulo seja de intensidade média ou alta, será 1. Se a intensidade sonora recebida for baixa, será 0;
- Presença/Ausência (PA): será 1 se houver algum rosto no campo visual do robô-cachorro e não estiver ocorrendo nenhum tipo de estímulo agressivo. No caso de haver um rosto no campo visual e ocorrer um estímulo agressivo ou se o usuário for embora (presença $\rightarrow$ ausência), valerá -1.

Como o carinho é o estímulo mais importante para determinar o nível de atenção do usuário, ele também possuirá o maior peso α. Logo, em um determinado instante k, o nível de atenção NA_k será dado por:

$$NA_k = sig\left(NA_{k-1} + \frac{4}{7} \cdot \beta\left(CA_k + \frac{PA_k}{2} + \frac{NS_k}{4}\right)\right), \tag{4.24}$$

onde $sig()$ representa a função sigmóide e β, onde $0 < \beta < 1$, é o nível de influência da nova entrada sobre o estado atual.

4.3 MODELO COMPORTAMENTAL

Esta seção aborda um dos tópicos centrais que inspiraram a realização deste trabalho: como fazer o robô aprender a partir da interação com o usuário e com o ambiente ao seu redor? Conforme colocado anteriormente, objetiva-se construir um robô que esteja sujeito a estímulos internos e externos e que reaja a eles, de forma semelhante a um animal real. Neste sentido, o robô-cachorro deverá também sempre buscar atender às suas motivações que correspondem aos seus estímulos internos. Sua principal motivação será manter o nível de atenção do usuário alto.

Todo o modelo comportamental do robô será aqui explicitado mais detalhadamente. O processo decisório é analisado de forma a propor um modelo de aprendizagem. Além disso, são apresentadas diversas simulações do modelo comportamental proposto e os seus resultados serão analisados.

4.3.1 O Cachorro como Agente

O módulo comportamental é responsável por tomar as decisões referentes ao comportamento do robô, determinando qual será a resposta que será apresentada ao usuário. Além disso, ele analisa se as decisões foram, ou não, corretamente tomadas, alterando seu comportamento de forma a maximizar uma recompensa que, no caso, será o nível de atenção do usuário. Sendo assim, este módulo encaixa-se perfeitamente na definição de agente autônomo apresentado anteriormente na seção 4.1.2.1.

Devido ao fato de o módulo comportamental atuar como um agente, é necessário determinar a fronteira entre este módulo e o mundo externo a ele. Esta delimitação é essencial para o desenvolvimento do modelo comportamental. O primeiro passo para definir este limite é a análise dos estímulos aos quais o módulo estará sujeito.

O cachorro estará sujeito a dois tipos de estímulos: internos ou externos. Os estímulos internos consistem nas suas motivações internas, às quais ele deverá atender. Como este trabalho está focado em estudar a interação de um *pet-robot* com um usuário, a principal motivação interna do robô será buscar manter um certo nível de atenção das pessoas ao seu redor. Desta forma, ele será sempre motivado a procurar despertar o interesse de um usuário. É importante lembrar que podem (e devem) existir outras motivações internas como, por exemplo, o nível de cansaço, de forma que o robô-cachorro deverá sempre tentar minimizar o nível de cansaço, como um animal real. No entanto, cabe ressaltar que todo o desenvolvimento do modelo comportamental descrito neste capítulo foi realizado considerando apenas o nível de atenção. Por outro lado, a última seção mostrará como generalizar este resultado para outros tipos de estímulo.

Os estímulos externos consistem em informações coletadas pelos sensores colocados no corpo do cachorro. Nesta primeira versão, o robô possuirá sensores de toque, ultra-som, dois microfones que permitem perceber a direção de estímulo sonoro e uma câmera de vídeo, que permitirá ao robô localizar os rostos dos usuários por programa desenvolvido pelo Prof. Geovany Araújo Borges. Conforme colocado na Seção 4.2, o principal desafio para o processamento de estímulos externos é converter os dados medidos pelos sensores em informações relevantes para o comportamento do cachorro. Esta interpretação é realizada pelo módulo de percepção, foi explicado na Seção 4.2.

Conseqüentemente, o ambiente no qual o agente está inserido é definido como tudo aquilo que não pertence ao módulo comportamental. O estado do ambiente será determinado pelos estímulos que o usuário estará enviando para o cachorro e pelo estado atual do robô-cachorro. Já os estímulos internos corresponderão a uma espécie de recompensa às decisões do cachorro, de forma que um nível de atenção alto representará uma recompensa alta. A partir do desenvolvimento realizado na Seção 4.2, a Tabela 4.2 mostra a divisão dos estímulos externos e internos do robô.

Definimos, assim, as entradas do nosso modelo comportamental. O cachorro deverá "gerenciar" seu estado emocional de forma a gerar estímulos externos que atendam a seus estímulos internos. Logo, observe que temos uma situação onde é necessário realizar uma otimização multi-critério. Inicialmente, o

Estímulos Externos (Estado do Usuário)	Estímulos Internos (Recompensas)
Carinho / Agressão	Nível de atenção do usuário
Som Alto / Som Baixo	Nível de Cansaço
Ausência → Presença	
Presença → Ausência	
Claro → Escuro	
Escuro → Claro	

Tabela 4.2: Estímulos do robô-cachorro

único estímulo interno que iremos considerar será o nível de atenção do usuário, já que todas as etapas envolvidas no tratamento deste estímulo podem ser facilmente generalizadas para lidar com qualquer outro estímulo interno.

Ainda analisando o módulo comportamental como um agente, é importante ressaltar três pontos:

- Neste modelo, o estado do ambiente, que será utilizado no processo decisório, será determinado por duas variáveis: o estímulo externo gerado pelo usuário e o estado emocional atual do robô-cachorro.
- A ação que o agente irá tomar, de forma a tentar alterar o ambiente a sua volta, toma a forma de um estado emocional, que será escolhido dependendo da entrada do usuário e do estado emocional no qual o agente se encontra no momento da decisão.
- O agente avaliará se as suas decisões estão sendo adequadas ou não analisando o nível de atenção do usuário, que será determinado no módulo de percepção.

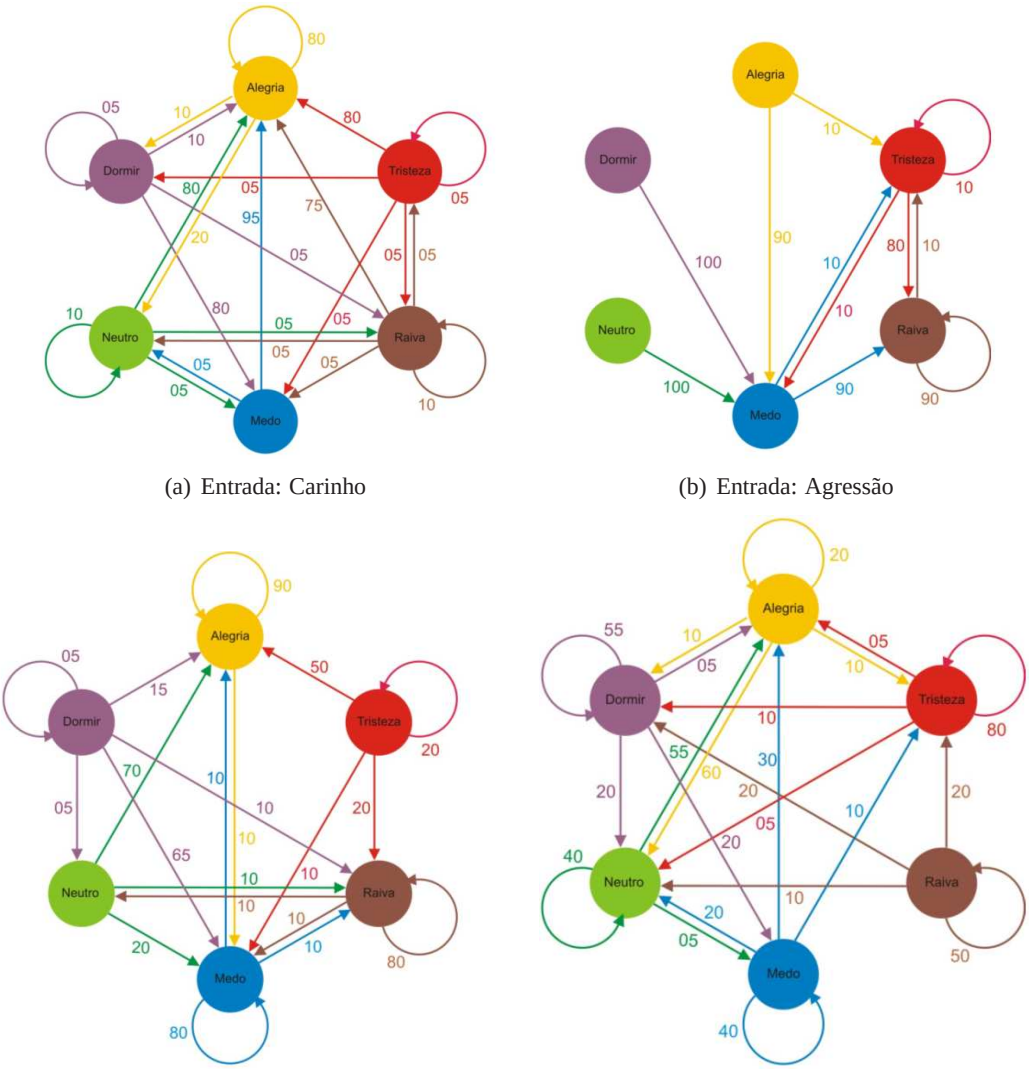
4.3.2 Organização Emocional e o Processo Decisório

O estado emocional do robô-cachorro definirá a resposta que será dada ao usuário. Desta forma, a decisão de qual estado emocional o robô apresentará em um dado instante de tempo corresponde à ação que o agente tomará para modificar o estado do ambiente a sua volta. A emoção escolhida pelo robô-cachorro irá refletir na sua expressão corporal, nos sons que ele estará emitindo e na cor dos seus LEDs. Por exemplo, se o cachorro ficar alegre, ele irá balançar o rabo e ficar com a cabeça erguida. Já se o robô estiver com raiva, seus olhos ficarão vermelhos e ele irá se posicionar de forma a confrontar o usuário, emitindo sons agressivos. Um estudo detalhado sobre como o cachorro adaptará seus atuadores para responder ao usuário pode ser encontrado em [6] e [28], que foram trabalhos desenvolvidos em paralelo a este, pelo mesmo grupo de pesquisa dos autores, com a cooperação do professor responsável e de colegas do Laboratório de Automação e Robótica Autônoma da Universidade de Brasília.

Neste momento, é interessante fazer algumas considerações sobre como são organizadas as emoções do robô. O cachorro poderá estar em 6 estados emocionais: **alegria, medo, tristeza, raiva, neutro ou dormir (cansaço)**. Estas são algumas das emoções básicas a todos os mamíferos e, através delas, será possível permitir que o robô-cachorro apresente atitudes próximas àquelas de um animal real, o que levará a uma interação mais natural com o usuário.

4.3.2.1 Escolha de Decisões como um Processo Markoviano

Antes de iniciar o desenvolvimento matemático de como irão ocorrer as transições entre os estados, é fundamental entender como será o processo decisório desenvolvido pelo agente. Conforme colocado na seção anterior, a ação escolhida pelo agente corresponde à emoção que será mostrada para o usuário, e



(a) Entrada: Carinho

(b) Entrada: Agressão

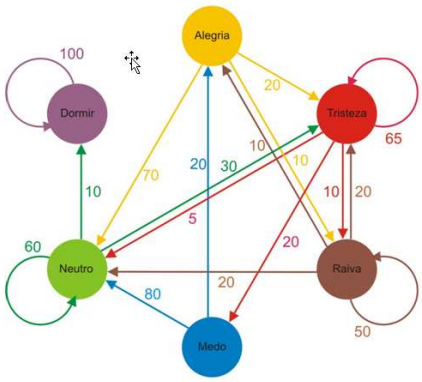
(c) Entrada: Som Alto

(d) Entrada: Som Baixo

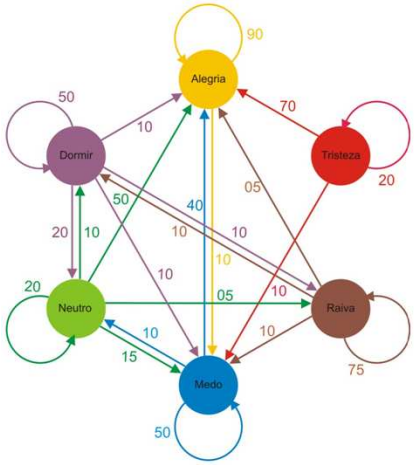
Figura 4.4: Cadeias de Markov iniciais para os estímulos Carinho (a), Agressão (b), Som Alto (c) e Som Baixo (d). Os diagramas foram desenvolvido por Nóbrega, [6].

o estado atual do ambiente será determinado pelos estímulos externos recebidos pelo cachorro e pelo seu estado emocional atual. Desta forma, dependendo do seu estado atual, o cachorro irá escolher uma ação que será apresentada para o usuário. Por exemplo, se o robô interpretar um estímulo externo como uma agressão e ele estiver alegre, ele poderá ir ou para o estado de tristeza ou para o estado de raiva. Considerando que estas decisões constituem um processo estocástico, de forma que a decisão por um estado emocional ou outro é probabilística, observa-se que a escolha do próximo estado emocional depende apenas do estado atual do ambiente. Desta forma, fica claro que o processo decisório constitui um Processo Markoviano.

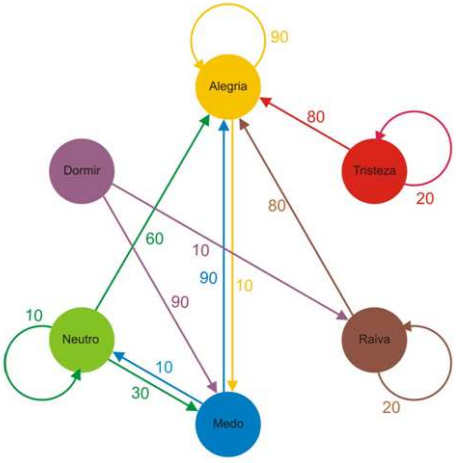
Sendo assim, o estado atual do ambiente no qual o agente está inserido será mapeado com as saídas através de uma política $\pi_{s_t}(a)$, que corresponde à probabilidade, em um dado instante de tempo t, de o agente tomar uma determinada decisão a quando o ambiente encontra-se no estado s. A distribuição $\pi_{s_t}(a)$ será determinada por uma cadeia de Markov cuja matriz de transição de probabilidades será dada por $\mathbf{P}_{E_{ext}}$, onde E_{ext} é o estímulo externo recebido pelo agente. Sendo $\boldsymbol{\pi}_{s_t}$ um vetor com comprimento igual ao número de ações que o agente pode tomar, a evolução da política pode ser simulada utilizando a Equação 4.6, de forma que, para um dado instante t:



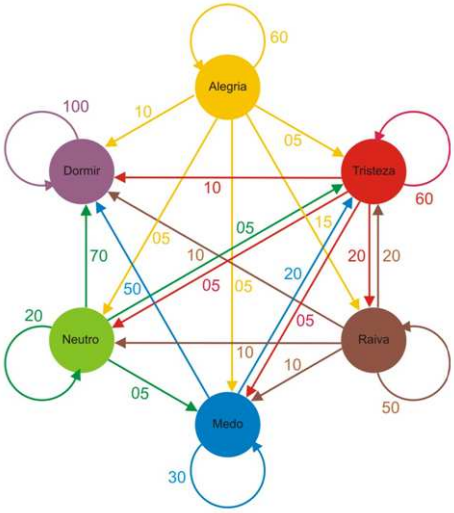
(a) Entrada: Presença→Ausência



(b) Entrada: Ausência→Presença



(c) Entrada: Escuro→Claro



(d) Entrada:Claro→Escuro

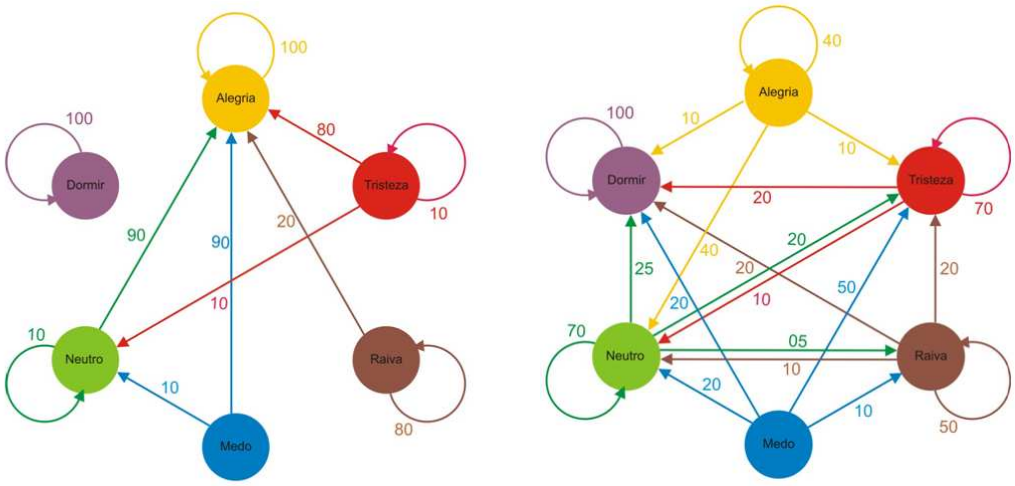
Figura 4.5: Cadeias de Markov iniciais para os estímulos Presença→Ausência (a), Ausência→Presença (b), Escuro→Claro (c) e Claro→Escuro (d). Os diagramas foram desenvolvido por Nóbrega, [6].

$$\boldsymbol{\pi}_{s_t}^T = \boldsymbol{\pi}_{s_{t-1}}^T \mathbf{P}_{E_{ext}} \tag{4.25}$$

Além disso, considerando a definição colocada na Seção 4.1.1, este processo é ergódico. As probabilidades de transição iniciais utilizadas nestas simulações foram determinadas por Nóbrega, [6], e são mostradas nas Figuras 4.4 a 4.6 como um diagrama de estados de uma cadeia de Markov.

4.3.2.2 Interação com o usuário como um processo de decisão markoviano

A avaliação dos resultados de uma determinada ação será realizada a partir da variação dos estímulos internos do robô-cachorro. Por exemplo, como o principal objetivo do agente é prender a atenção do usuário, uma determinada decisão (que se reflete em uma ação), tomada pelo módulo comportamental será considerada adequada se conseguir aumentar o nível de atenção do usuário ou mantê-lo em um nível alto. A questão de como o nível de atenção do usuário é calculado foi abordada em detalhe na Seção 4.2.6. Ela tenderá para 1, caso esteja ocorrendo uma interação positiva entre o usuário e o cachorro robô, tenderá para -1 caso a interação seja agressiva e, finalmente, tenderá para 0 casa não haja interação. Idealmente,



(a) Entrada: Cor Quente

(b) Entrada: Cor Fria

Figura 4.6: Cadeias de Markov iniciais para os estímulos Cor Quente (a) e Cor Fria (b). Os diagramas foram desenvolvido por Nóbrega, [6].

o robô-cachorro deveria escolher apenas aquelas decisões que mantivessem o nível de atenção do usuário alto. Conseqüentemente, há uma forte relação entre a idéia do nível de atenção do usuário e a recompensa que existe no Processo de Decisão Markoviano (MDP).

Na realidade, a interação do agente com o usuário e o ambiente a sua volta pode ser caracterizada como um MDP. Isto ocorre pois o processo decisório é Markoviano e porque, utilizando apenas o estado atual do ambiente e as possíveis decisões que o cachorro pode tomar, podemos fazer uma estimativa da próxima recompensa e do próximo estado do ambiente. Desta forma, definimos o modelo de tomada de decisões do robô.

4.3.3 Aprendizado

Diversos trabalhos abordam a questão de aprendizagem e tomada de decisões de *pet-robots* e outros robôs cujo objetivo é interagir com o usuário. Um trabalho interessante sobre o papel das emoções na aprendizagem de máquina e na interação robô-homem é conduzido pelo grupo de robótica afetiva do MIT Media Lab e os seus resultados tiveram grande influência sobre este trabalho. Os artigos [29], [30], [31] e [32] abordam extensamente esta questão da "robótica afetiva" e robótica comportamental.

É importante lembrar que, dentro do escopo deste trabalho, o principal objetivo do robô-cachorro é prender a atenção do usuário. Desta forma, independentemente do método de aprendizagem utilizado, a sua meta será influenciar as decisões do agente de forma que ele escolha ações que mantenham o nível de atenção alto. Além disso, é importante levar em consideração que, como mostrado na seção anterior, o processo decisório do agente constitui um Processo de Decisão Markoviano.

A escolha do método de aprendizagem deve levar em conta o fato de não ser possível formular um modelo preciso do ambiente no qual o agente está inserido. Como o robô foi projetado para interagir com o usuário e prender sua atenção, para modelar o "mundo" do agente, seria necessário montar um modelo do comportamento humano e da sua interação com o robô. A questão da interação homem/*pet-robot* foi estudado por Kerepesi, et al, [33] e mostra que é muito difícil prever como um ser humano, adulto ou criança, irá interagir com o robô-cachorro. Devido a estas questões, a aprendizagem não deve necessitar de um modelo do ambiente no qual o robô está inserido.

A aprendizagem do agente também deve ser "*on-line*", ou seja, em tempo-real. Como o robô estará

inserido em um ambiente dinâmico, onde haverá interação constante com o usuário e o mundo a sua volta, seu modelo comportamental também deverá evoluir dinamicamente. Esta mesma questão impossibilita a utilização de qualquer forma de treinamento supervisionado, pois não há como avaliar, de forma direta, se as respostas do cachorro estão sendo adequadas ou não.

Deverá ser considerada, também, a questão da eficiência computacional da forma de aprendizagem a ser utilizada. Como, futuramente, todo comportamento do cachorro será colocado em um processador embarcado, questões como quantidade de memória exigida pelo modelo e tempo de processamento tornam-se essenciais. Por exemplo, alguns Métodos de Monte-Carlo para aprendizagem, assim como abordagens construídas a partir do uso de redes neurais, exigem considerável poder de processamento, limitando um modelo de percepção e aprendizagem em larga escala.

Conseqüentemente, para a evolução do modelo comportamental do robô-cachorro propõe-se que seja utilizado um modelo que permita aprendizagem *on-line* de forma não-supervisionada e que seja compatível com o método de tomada de decisão do agente, que pode ser modelado como um Processo de Decisão Markoviano. Dentro deste escopo, a utilização da aprendizagem por reforço apresenta-se como a melhor escolha.

Dentro da aprendizagem por reforço, o método temporal-diferencial (TD) é um método computacionalmente eficiente e que não necessita de um conhecimento preciso do ambiente no qual o agente está inserido. Este método é relativamente simples de implementar e permite uma evolução rápida e eficiente do modelo comportamental, exigindo uma memória mínima e poucas operações do sistema responsável pelo comportamento. Além disso, não é desejado que o modelo comportamental seja atualizado a cada ciclo de interação com o usuário. Isto poderia levar facilmente a um comportamento aparentemente "caótico" do robô e uma evolução falha da política do agente. Logo, é interessante que o módulo tomador de decisões atue de forma independente do módulo responsável por analisar as conseqüências das ações do agente. Um modelo de aprendizagem temporal-diferencial que atende a todos estes quesitos é o método Ator-Crítico, exposto na Seção 4.1.2.7. O restante desta seção se dedicará a explicar como este método foi utilizado para permitir o aprendizado do robô-cachorro.

4.3.4 Modelo Ator Crítico Aplicado ao Robô-Cacorro

Como mencionado na Seção 4.1, o método de aprendizagem ator-crítico, o ator é responsável pela política de escolhas de decisões e por efetivamente escolher as ações que o agente irá realizar sobre o ambiente a sua volta, enquanto o crítico avalia as decisões tomadas pelo ator e atualiza a sua política, aumentando ou diminuindo a tendência de optar por uma determinada decisão. Conseqüentemente, o ator e o crítico são implementados em estruturas separadas, de forma que atuem independentemente um do outro. Para o robô-cachorro desenvolvido neste trabalho, o módulo comportamental foi estruturado de acordo com o diagrama de blocos mostrado na Figura 4.7. A seguir, o funcionamento do ator e do crítico serão explicados.

4.3.4.1 O Ator

O ator foi dividido em duas partes: o seletor de políticas e o seletor de estado emocional. Esta divisão foi feita de forma a simplificar e organizar a implementação computacional do ator, além de ajudar no entendimento do sistema. O seletor de políticas irá receber os estímulos externos (Tabela 4.2) já processados pelo módulo de percepção e selecionar a matriz de transição $P_{E_{ext}}$ correspondente. O seletor de estado emocional irá receber do seletor de políticas uma matriz de transição e, baseado nesta matriz e no estado emocional atual, determinar qual estado o agente irá escolher, enviando-o para o módulo de linguagem e comunicação.

O seletor de políticas pode receber várias entradas simultaneamente. Por exemplo, o cachorro pode

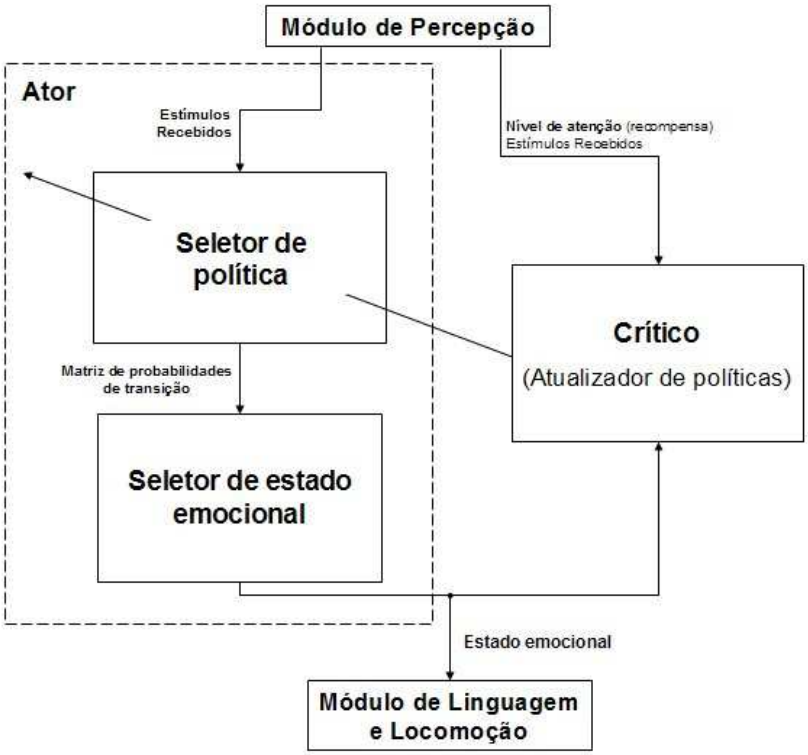


Figura 4.7: Modelo ator-crítico para aprendizado por reforço.

perceber, no mesmo instante de amostragem, um carinho e um som baixo, ou um som alto e o sinal de ausência → presença. Quando múltiplas entradas são detectadas, a matriz de transição que será enviada para o seletor de estado emocional será uma média ponderada das matrizes de transição de cada entrada. Desta forma, é possível obter uma distribuição que representa as características da política de cada entrada. Uma desvantagem desta forma de processar os sinais de entrada é que todas as entradas possuirão o mesmo peso, não existindo uma hierarquia entre elas. No cachorro real, isto não acontece. Por exemplo, uma agressão certamente irá influenciar mais o próximo estado emocional do cachorro do que um som baixo. Este problema foi compensado em parte no cálculo do nível de atenção, mostrado na seção 4.2.6, onde o nível de atenção é atualizado a partir de uma soma ponderada das entradas percebidas pelo robô-cachorro em um dado instante. Logo, a matriz de transição $\mathbf{P}_{trans}$ que será enviada para o módulo tomador de decisões será:

$$\mathbf{P}_{trans} = \frac{1}{ent} \sum_{ent} \mathbf{P}_{E_{ext}} \tag{4.26}$$

Na equação acima, ent corresponde ao número estímulos recebidos do módulo de percepção. É importante ressaltar que as matrizes de transição sempre serão normalizadas de forma que a soma dos elementos em cada linha seja igual a 1.

O seletor de estados emocionais irá atualizar a política π, a partir da matriz de transição recebida do seletor de políticas, utilizando a expressão 4.6. A partir disso, ele gera uma variável aleatória de distribuição uniforme e, baseado nela, atualiza o estado emocional. O novo estado emocional será enviado para o Módulo de Linguagem e Locomoção, conforme descrito na Figura 4.7 e, para o Crítico, de forma a avaliar a recompensa para a decisão tomada.

4.3.4.2 O Crítico

O crítico é a estrutura responsável por analisar o efeito das decisões tomadas pelo ator e alterá-lo de forma que ele sempre tome decisões que, idealmente, sempre induziriam o usuário a ter uma interação positiva com o robô, mantendo o nível de atenção alto. Não há uma política "ideal" para este problema da interação usuário-agente, pois os padrões de comportamento que terá o usuário irão variar de pessoa para pessoa. Conseqüentemente, é desejado que as ações tomadas pelo agente possam se adaptar para o usuário que está interagindo com ele naquele momento. Logo, o modelo comportamental deverá conseguir se adaptar de forma relativamente rápida, buscando maximizar, a curto prazo, o nível de atenção do usuário (NA), que será a recompensa deste esquema de aprendizagem por reforço.

O crítico manterá uma função de valor $V(s_t)$, para cada estado do ambiente no qual o agente está inserido. Como o estado atual do ambiente é determinado pelo estímulo externo que ele está recebendo naquele instante e pelo seu estado emocional atual, será necessários armazenar N funções de valor, que será dado por:

$$N = N_{E_{ext}} N_{Eemoc} \tag{4.27}$$

Na equação anterior, $N_{E_{ext}}$ é o número de estímulos externos diferentes que o cachorro poderá receber e N_{Eemoc} corresponde ao número de estados emocionais diferentes que o agente poderá assumir. Se o robô receber mais de um tipo de estímulo externo em um determinado instante, a função de valor que será considerada para aquele instante será uma média ponderada das funções de valor dos estados correspondentes aos estímulos externos recebidos. Desta forma, o erro TD (δ_t), que será utilizado para atualizar a política e as funções de valor, será calculado como:

$$\delta_t = NA_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4.28}$$

Observe que a recompensa para ações tomadas no instante t serão recebidas no instante $t+1$. Deste modo, a função de valor $V(s_t)$ será atualizada apenas no instante $t+1$. O termo de desconto γ será sempre pequeno ($\gamma < 0,3$), de forma a permitir uma adaptação rápida do modelo. A função de valor será atualizada utilizando a regra de atualização:

$$V(s) \leftarrow V(s) + \alpha\delta_t, \tag{4.29}$$

onde α é um fator que indica o tamanho do passo e é determinado heuristicamente. Analogamente, para um dado instante t, a probabilidade $P_{s_t a_t}$ de o agente tomar a decisão a, quando o ambiente estiver no estado s será atualizada seguindo a relação:

$$P_{sa} \leftarrow P_{sa} + \beta\delta_t \tag{4.30}$$

Após esta atualização, sua matriz de transição é normalizada, de forma que a soma dos elementos de cada linha sempre seja igual a 1. Sendo assim, o crítico atualiza a sua função de valor e as matrizes de transição, de forma que o agente tenha uma maior tendência a escolher decisões que aumentem o nível de atenção do usuário.

4.3.5 Decisões Encadeadas

Uma característica importante do modelo comportamental proposto neste trabalho é a facilidade de ser expandido, permitindo que ocorram diversas decisões em cascata ou em paralelo, onde cada processo decisório constitui um Processo Decisão Markoviano independente. Neste cenário, o agente seria dividido em vários subagentes, cada um sendo constituído por um Ator e um Crítico. Logo, os módulos Ator/Crítico

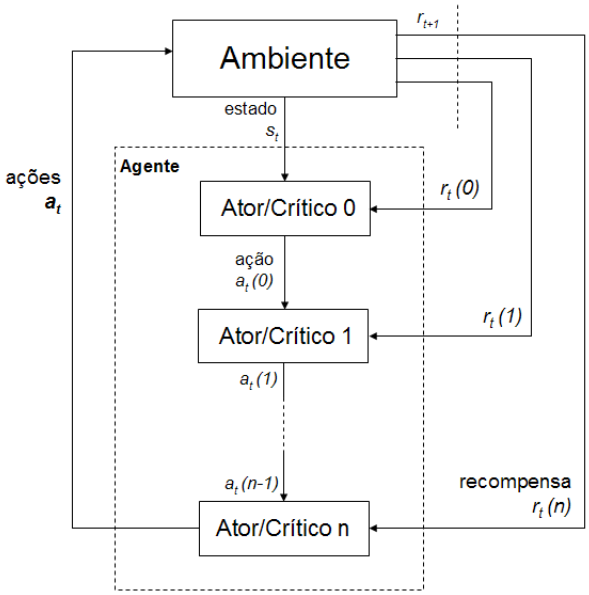


Figura 4.8: Modelo ator-crítico para decisões encadeadas.

em cascata corresponderiam a uma série de decisões encadeadas, sendo que a entrada do subagente mais externo seria o estado do ambiente, denotado por s_t, e a entrada dos subagentes seguintes corresponderiam a ação tomada pelo subagente diretamente acima dele. Desta forma, para um subagente n, onde $n > 0$, sua entrada seria a ação tomada pelo subagente $n - 1$. Um modelo de decisões encadeadas é apresentado esquematicamente na Figura 4.8, onde ocorrem $n + 1$ decisões encadeadas. O resultado deste processo decisório será uma ação com alto grau de complexidade, realizada pelo agente sobre o ambiente.

Cada subagente poderá ter um conjunto único de decisões, de políticas e de recompensas. Sendo assim, a evolução de cada subagente ocorre de forma independente dos demais subagentes, o que facilita a sua implementação em hardware e torna o sistema mais robusto. Além disso, é possível ter diversas cadeias de decisões em paralelo, cada uma correspondendo a uma ação sobre um aspecto do ambiente. Alguns estudos relacionam este tipo de arquitetura de aprendizado com a organização do cérebro humano, conforme colocado por Daphna, et al, [25].

Para exemplificar uma possível expansão do modelo comportamental utilizado, considere que o agente deverá escolher, além de qual emoção manifestar para o usuário, a intensidade desta emoção. Para um dado estado emocional, poderão ser escolhidos três níveis de intensidade diferentes: alto, médio ou baixo. A intensidade emocional irá depender apenas do estado emocional do agente, sendo independente do estímulo externo recebido, de forma que o cachorro poderá ser muito ou pouco "emotivo". Este tipo de decisão é conveniente, pois se for associado a cada nível de intensidade emocional uma configuração dos LEDs ou posição dos servo-motores, será possível que o robô-cachorro configure sozinho os seus atuadores, de forma a gerar reações que estimulem a interação positiva com o usuário, sem a necessidade de um ajuste prévio destes parâmetros. A recompensa, neste caso, estaria associada ao nível de atenção, de forma semelhante ao processo de escolha do estado emocional. A política adotada poderia estar associada, por exemplo, a uma distribuição de Gibbs [21].

Além disso, suponha que o agente esteja sujeito a um novo estímulo interno, o nível de cansaço, que estará associado à carga de suas baterias. Este estímulo será utilizado para determinar a configuração dos seus atuadores, de forma que o robô receberá uma recompensa pequena se a bateria estiver com carga baixa e ele optar por “andar rápido” ou tomar outras atitudes que consumam energia. Desta forma, o agente aprenderá a “gerenciar” o seu consumo de energia.

A organização deste sistema é mostrado na Figura 4.9 e poderá servir de base para uma futura expansão do módulo de comportamento do robô-cachorro. Nesta figura, o subagente Ator/Crítico 0 é responsável

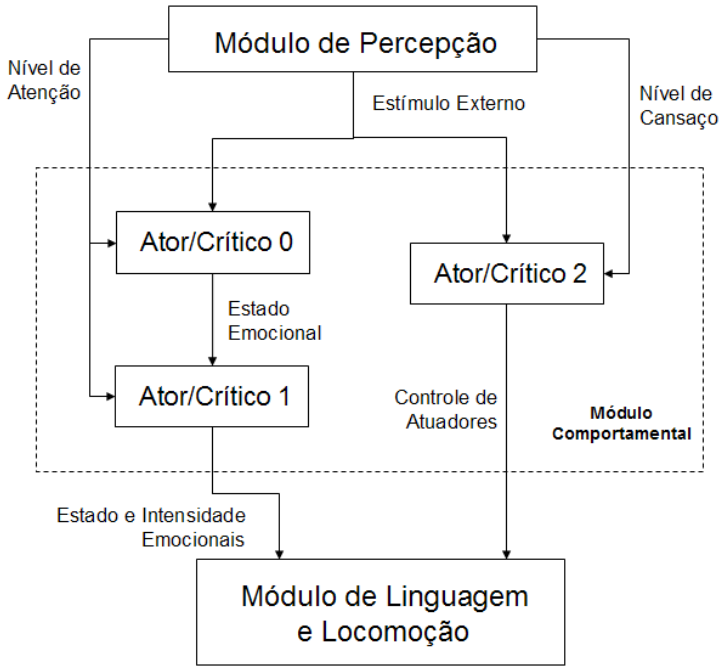


Figura 4.9: Modelo Comportamental considerando dois novos processos decisórios: a escolha da intensidade da emoção e a gerência do nível de cansaço.

por escolher o estado emocional, o Ator/Crítico 1 é responsável pela escolha da intensidade da emoção e o Ator/Crítico 2 coordena os atuadores de forma a minimizar o cansaço.

4.3.6 Metodologia e Resultados experimentais

Conforme colocado anteriormente, o módulo comportamental foi desenvolvido para o LINUX, utilizando a linguagem C, de forma que ele pudesse ser futuramente implementado em um sistema embarcado que atuaria como o "cérebro" do robô-cachorro. Para validar o modelo proposto nas seções anteriores, foram realizadas diversas simulações computacionais de forma a analisar a evolução da política utilizada pelo ator e a coerência do processo de tomada de decisão. A seguir, a metodologia utilizada e alguns resultados obtidos serão expostos.

4.3.6.1 Metodologia de Testes

Diversas dificuldades surgem ao tentar realizar uma simulação do módulo comportamental. A dificuldade inicial decorre do fato de não ser possível criar um modelo "preciso" do ambiente no qual o agente está inserido. Isto ocorre pois é impossível modelar um usuário real. Cada pessoa possuirá uma reação diferente em relação ao cachorro, não havendo uma política ideal que possa ser utilizada pelo ator.

O segundo problema consiste no desafio de como realizar uma simulação de interação com o usuário sem possuir um robô construído. Para possuir um robô-cachorro próximo do animal real, é necessário que a parte comportamental conduza o desenvolvimento da parte física e não o contrário. Sendo assim, é necessário que exista um modelo comportamental válido para ajudar na escolha dos sensores, atuadores e do formato do corpo do robô. O desenvolvimento de um modelo virtual do cachorro seria uma alternativa, possível que permanecesse como área promissora para pesquisas futuras.

Um último desafio que precisou ser enfrentado para a definição da metodologia de testes do módulo comportamental foi o estudo dos limites do modelo de aprendizado. A análise do comportamento do

Crítico e da sua influência sobre o Ator em um ambiente de interação simulada com o usuário é essencial para analisar como será o comportamento e a evolução do agente no ambiente "real". Além disso, é necessário determinar as diversas constantes que são utilizadas no modelo de aprendizagem por reforço. Estes termos são definidos heuristicamente, mas podem ser estimados realizando algumas considerações sobre o significado de cada um deles e sobre o comportamento que se espera do agente.

Desta forma, as simulações e resultados apresentados nesta seção não objetivam prover uma descrição exata de como o agente atuará em um cenário real, onde o usuário interage com o robô já montado, mas sim uma análise dos limites e de algumas possibilidades do modelo de aprendizado, de forma que se possa validar o modelo. Deseja-se observar como o Crítico altera a sua política dependendo da "atitude" do usuário. É importante ressaltar que a política do Ator é atualizada por meio da alteração das probabilidades de transição descritas na Seção 4.3.4.

Como não existe uma política ótima, de forma que é muito difícil avaliar a "qualidade" de uma determinada política, as simulações foram realizadas de forma a se analisar a evolução da política do agente ao estar perante um usuário com atitude "enviesada". Considerando os estímulos externos aos quais o robô-cachorro está sujeito, um *usuário enviesado* será aquele que, para um determinado estímulo de entrada, só interagirá positivamente a um tipo de resposta do robô a este estímulo. Por exemplo, o usuário apenas fará carinho no cachorro se ele ficar com medo ao ouvir um som alto. Qualquer outro estado emocional escolhido pelo agente corresponderá a uma reação agressiva por parte do usuário. Conseqüentemente, será analisado se o agente irá alterar a sua política de forma a sempre escolher a atitude que leve a uma interação positiva com o usuário enviesado. No exemplo colocado, espera-se que o Crítico aumente a tendência do Ator escolher o estado "medo", quando receber o estímulo "som-alto", independentemente do estado emocional no qual o agente se encontra no momento em que o estímulo é recebido.

Antes de iniciar as simulações, foram realizadas algumas considerações sobre as constantes utilizadas no aprendizado e que precisam ser determinadas heuristicamente. A mais importante destas constantes é o fator de desconto, γ. Caso esta constante seja próxima de 1, o agente irá buscar otimizar a recompensa a longo prazo, ou seja, as decisões serão tomadas de forma que o agente sempre vá para aqueles estados que possuem a maior Função de Valor. Já se ela for próxima de zero, as decisões serão tomadas de forma a maximizar as recompensas de curto prazo. Logo, para um γ pequeno, o agente se adaptará rapidamente às variações de padrão de atitude do usuário. Como esta é uma característica desejada do robô-cachorro, foi escolhido $\gamma = 0,3$.

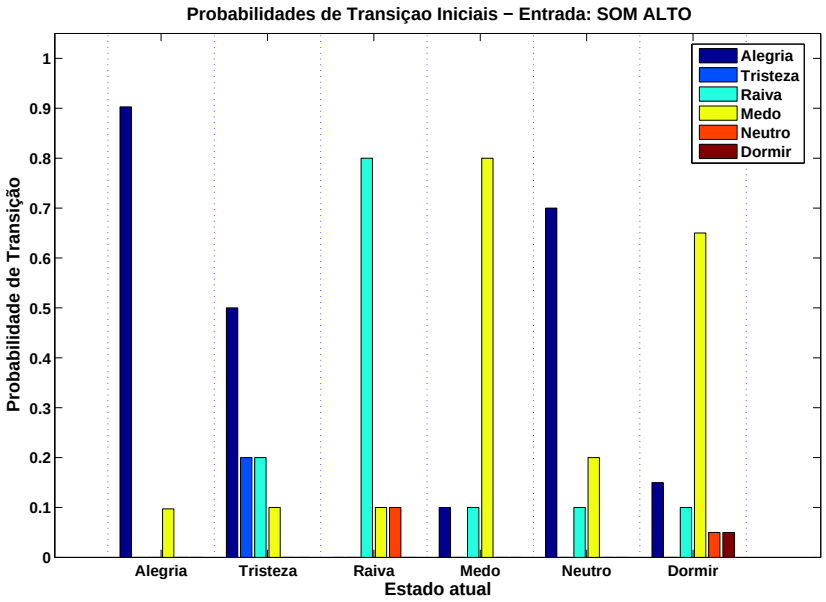
As outras duas constantes a serem determinadas correspondem ao tamanho do passo que será utilizado para atualizar a Função de Valor, denotado por α, e ao tamanho do passo que será utilizado para atualizar as probabilidades de transição, denotado por β. Como o erro TD já será bastante pequeno, foi considerado $\alpha = \beta = 0,4$. Observe que estes fatores não são determinantes para a convergência da simulação, pois alteram apenas a sua velocidade.

Para cada teste realizado, foram simuladas 200 interações entre o usuário e o agente. As matrizes de transição resultantes foram armazenadas em arquivos de dados no formato .dat e traçadas como um gráfico de barras utilizando o *software* de simulação matemática MATLAB. Baseado nestes gráficos, será realizada a análise dos resultados obtidos.

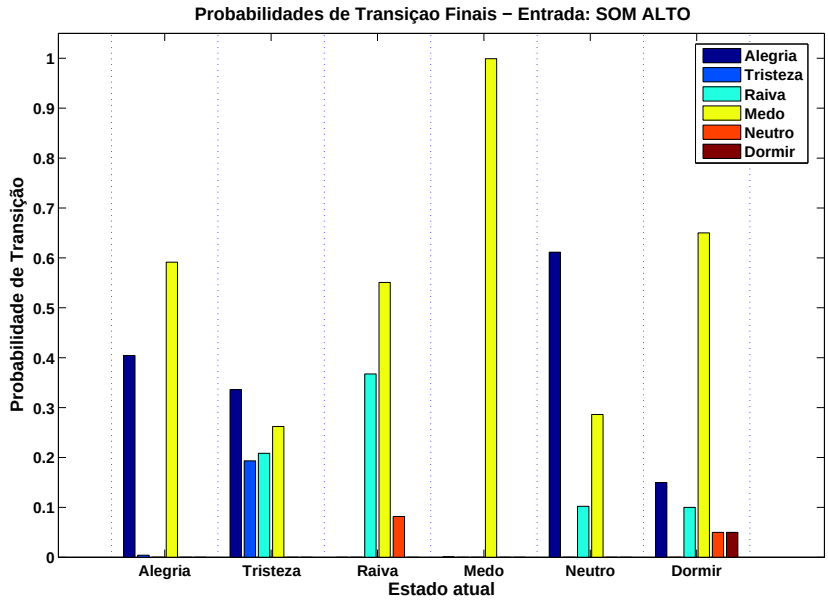
Conseqüentemente, o objetivo das simulações não é observar se o robô-cachorro apresenta um comportamento próximo ao de um cachorro real, mas sim se ele consegue atender à sua motivação interna, adaptando seu comportamento de forma a tentar prender a atenção do usuário, mantendo o seu nível de atenção alto. Dentro deste contexto, os resultados de duas simulações serão apresentados a seguir.

4.3.6.2 Simulação 1 - O cachorro medroso

A primeira simulação realizada corresponde ao exemplo dado no início da seção anterior. Foi simulado um usuário enviesado que apenas respondia positivamente ao cachorro caso ele escolha o estado emocional



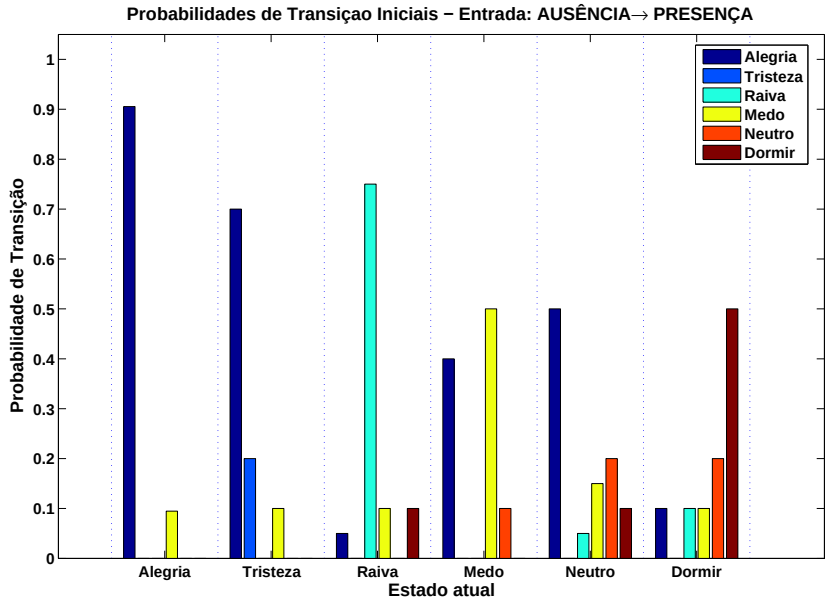
(a) Probabilidades Iniciais



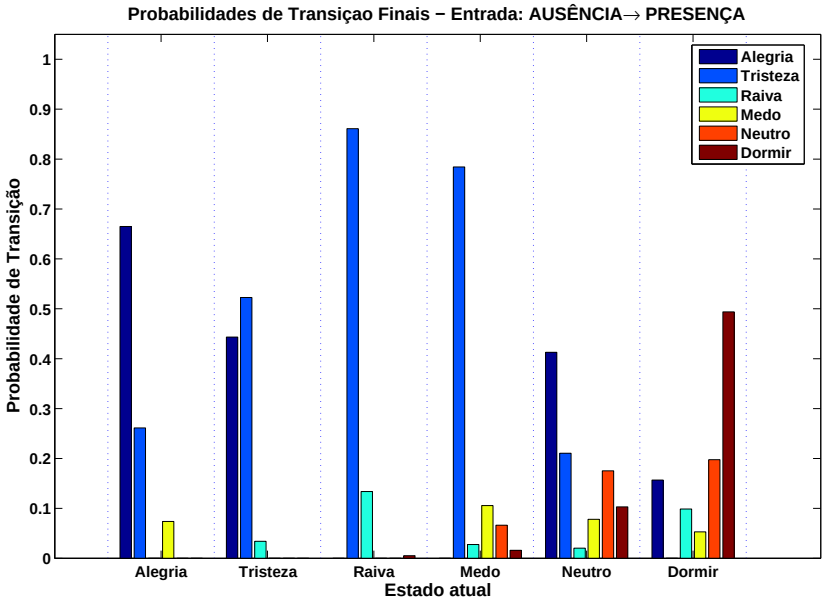
(b) Probabilidades Finais

Figura 4.10: Simulação do cachorro medroso. A figura (a) corresponde às probabilidades de transição iniciais, enquando a figura (b) corresponde às probabilidades de transição após 200 passos de simulação

"medo" ao receber um estímulo correspondente a um barulho alto. Desta forma, o usuário emite um barulho alto e se o cachorro ficar com medo, o próximo estímulo que o usuário irá realizar sobre o cachorro será um carinho. Caso o cachorro assuma um estado emocional que não seja o medo, ele receberá uma agressão. Como, inicialmente, o cachorro tenderá a ficar alegre ao ouvir um som alto, ele deverá se adaptar para tomar uma ação contrária a sua tendência inicial. Os resultados desta simulação estão expostos na Figura 4.10.



(a) Probabilidades Iniciais



(b) Probabilidades Finais

Figura 4.11: Simulação do cachorro tristonho. A figura (a) corresponde às probabilidades de transição iniciais, enquanto a figura (b) corresponde às probabilidades de transição após 200 passos de simulação

4.3.6.3 Simulação 2 - O cachorro tristonho

A segunda simulação é semelhante a primeira, mas o usuário responderá positivamente apenas se o agente ficar "triste" ao perceber sua presença. Desta forma, para receber carinho, o robô deverá ficar triste ao receber um sinal de ausência→presença. Qualquer outro estado levará a uma resposta agressiva do usuário. Mais uma vez o robô deverá tomar uma atitude contrária a sua tendência inicial. Os gráficos correspondentes a esta simulação são mostrados na Figura 4.11.

4.3.7 Análise dos Resultados e Discussão

Em ambas as simulações realizadas podemos observar, claramente, que o agente comporta-se de acordo com o previsto. Nos dois casos, ele se adaptou de forma a aumentar a tendência do ator escolher ações que levem a um aumento do nível de atenção do usuário. Isto ocorreu através da variação das matrizes de transição, o que afeta diretamente a política utilizada pelo ator para tomada de decisões. Nas 200 interações simuladas, a entrada principal do usuário enviesado foi aplicada 100 vezes, enquanto as interações restantes correspondiam a interações de "avaliação", onde o agente recebia uma recompensa de acordo com a sua resposta.

No caso do cachorro medroso, o agente só receberia atenção e, conseqüentemente, uma recompensa positiva, caso tomasse uma atitude "medrosa". Observamos, na Figura 4.10 que, de fato, isto ocorreu. Para todos os estados emocionais, exceto o estado "dormir", que nunca é assumido nesta simulação, a probabilidade de transição para o estado "medo" aumentou quando o estímulo externo correspondia a um barulho alto. Fica clara a adaptação do agente de forma a atender os seus estímulos internos. Entretanto, observamos que, apesar do aumento significativo da probabilidade do agente se comportar como um cachorro medroso, esta não será, necessariamente, sempre a sua atitude preferência. Isto pode ser visto, claramente, nas probabilidades de transição quando o estado atual é o "neutro". Isto indica que as 200 interações simuladas não foram suficientes para levar a uma convergência da política utilizada pelo ator para uma política ótima. Além disso, os resultados refletem as limitações do método de simulação utilizado, onde o usuário enviesado possui um comportamento determinístico.

A mesma conclusão pode ser obtida ao analisar os resultados da simulação do cachorro tristonho, apresentado na Figura 4.11. A probabilidade de transição para a emoção "triste" ao perceber a presença do usuário aumenta para todos os estados, exceto para o estado "dormir", que não é acessado nesta simulação. Novamente, pode se observar que o agente foi capaz de "aprender" e se adaptar ao comportamento do usuário. Além disso, como ocorreram apenas 200 interações, não foi possível que o agente atingisse a política "ideal" que, neste caso, seria sempre ir para o estado triste ao perceber a presença do usuário, independente do estado emocional atual.

Analisando estas simulações sobre outro aspecto, observa-se que ocorreu uma espécie de "adestramento emocional" do agente. De forma semelhante ao cachorro de Pavlov, o agente foi induzido a associar a escolha do estado emocional a uma recompensa que poderia ser recebida no futuro. A utilização de uma técnica de "adestramento simulado" para o treinamento de sistemas baseados em aprendizado por reforço foi utilizado por Kaplan, *et al*, [33], para o adestramento de um cachorro virtual. Sendo assim, o método de aprendizado Ator-Crítico pode ser utilizado não apenas para a coordenação de um "modelo emocional" de um agente, mas para qualquer outro modelo que envolva a interação com um usuário.

Conseqüentemente, o modelo comportamental proposto para o robô-cachorro desenvolvido neste trabalho demonstrou ser capaz de aprender e se adequar ao padrão de comportamento do usuário, mesmo que isto signifique uma alteração significativa da sua política inicial. Além disso, o sistema mostrou ser uma forma eficiente de aprendizado *on-line*, exigindo baixo esforço computacional e adaptando-se rapidamente a variações do padrão de comportamento do usuário. É importante ressaltar que para alterar a velocidade de adaptação é necessário apenas modificar as constantes γ, β e α do processo de aprendizagem. Logo, o modelo comportamental proposto neste capítulo atende aos objetivos colocados inicialmente para o comportamento do robô-cachorro.

5 CONCLUSÕES

Neste trabalho foi concebido e desenvolvido uma plataforma de base de *pet-robot* que será utilizado como uma ferramenta para estudos sobre aprendizagem de máquina e da interação homem/máquina. A metodologia utilizada, baseada em uma visão "*top-down*" do problema, foi apresentada e discutida. Dentro deste contexto, o projeto foi dividido em duas partes: a primeira, de baixo nível, refere-se aos sensores e atuadores, e a segunda, de mais alto nível, corresponde ao modelo comportamental e de aprendizado do robô.

A parte sensorial corresponde à elaboração de blocos para aquisição de dados do ambiente. Já a parte de atuação diz respeito às respostas do sistema ao mundo externo. Conforme foi comentado, parte dessa etapa de atuação, aquela responsável pelos movimentos, foi um trabalho paralelo a este, desenvolvido por dois alunos de Engenharia Mecatrônica. Esta realização paralela se deu devido a seu nível de complexidade elevado e exigência de conhecimentos distintos daqueles aqui utilizados. O sensor de ultra-som foi desenvolvido integralmente e diversos testes foram realizados, de modo a garantir a confiabilidade deste módulo para o robô-cachorro e a fornecer as informações necessárias para seu uso em outras aplicações. Os sensores de toque e de intensidade sonora, bem como a reprodução de áudio (atuação), foram desenvolvidos parcialmente, com a preocupação de deixar as diretrizes para os alunos que derem continuidade ao trabalho. Para o caso do sensor de toque, optou-se por substituí-lo por um sensor encapsulado em um circuito integrado, porém o sistema não chegou a ser testado com os novos sensores. Para a detecção de nível sonoro, os microfones utilizados apresentaram respostas distintas entre si, de forma que as saídas dos dois sistemas não foram suficientes para atingir um nível de detecção desejável. Acredita-se que o problema seja com os microfones e não exatamente com o circuito, uma vez que um deles se comportou como esperado em ambos os circuitos e o outro precisou que o ganho do circuito fosse máximo para apresentar uma resposta pífia. Quanto à reprodução de áudio, optou-se por temporariamente substituí-la pela reprodução direta do PC. Sugere-se que a leitura da memória não seja mais feita *byte* a *byte*, mas sim pela formatação FAT32, suportada tanto pela memória quanto pelo microcontrolador, pois ela permite maior flexibilidade de operação dos dados. Desenvolveu-se, ainda, um protocolo de comunicação, que realiza a interligação de cada bloco com o sistema central. Por ter sido desenvolvido em baixo nível, seu funcionamento foi descrito dentro da primeira parte.

A parte comportamental, que norteou o desenvolvimento da parte de baixo nível, corresponde a um agente tomador de decisões responsável por definir quais respostas o robô envia ao mundo externo. Baseado em estudos sobre o comportamento animal, o robô deverá escolher entre seis estados emocionais que ele pode assumir. Estes estados correspondem às emoções: feliz, triste, medo, raiva, neutro e dormir (sonolência). Um modelo para o processo de tomada de decisão foi criado utilizando o Método Ator-Crítico de aprendizagem por reforço, permitindo que o robô aprendesse através da interação com o ambiente a sua volta, adaptando o seu comportamento de forma a sempre tentar maximizar o nível de atenção do usuário. O modelo foi implementado e simulado, de forma a testar a sua validade e eficiência. Os resultados, expostos no Capítulo 4, evidenciaram que o modelo proposto consegue adaptar-se a diferentes padrões de comportamento do usuário, "aprendendo" a gerenciar suas emoções de forma ótima. Além disso, o modelo desenvolvido pode ser facilmente expandido, permitindo que o agente possa tomar diversas decisões encadeadas, o que leva a uma ação mais elaborada.

O trabalho indica haver várias áreas e projetos a serem desenvolvidos no futuro. Por exemplo, os módulos sensoriais podem ser aperfeiçoados, permitindo a obtenção de informações mais precisas sobre o usuário e o ambiente no qual o *pet-robot* está inserido. Além disso, pode e deve ser finalizada a integração entre estes módulos e o módulo comportamental para que possa ser observado como o *pet-robot* irá agir e aprender através da interação com um usuário real. Uma vez realizada esta integração, o módulo comportamental deverá ser implementado em um sistema embarcado, que atuará como "cérebro" do *pet-robot*.

Um último desafio que deve ser resolvido futuramente é a expansão do modelo de tomada de decisões e aprendizado do robô, de forma que ele possa apresentar comportamentos mais elaborados para o usuário.

Por fim, conclui-se que o trabalho desenvolvido atingiu grande parte dos objetivos colocados no início do projeto. Do ponto de vista acadêmico, o trabalho permeou diversas áreas da Engenharia Elétrica e Computação, permitindo um processo de aprendizado e amadurecimento importante para os alunos. Além disso, a metodologia utilizada e os resultados obtidos podem servir como referência para o desenvolvimento de outras plataformas inteligentes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] FUJITA, M. Aibo: Toward the era of digital creatures. *The International Journal of Robotics Reaserch*, Sage Publications, v. 20, n. 10, p. 781–794, Outubro 2001.

[2] OFFICIAL i-Cybie Web Site. 2006. Disponível em: <http://www.i-cybie.com/English/EServiceHF.html>. Acesso em 07.dez.2006.

[3] HOW to Use an MMC. Apr 2006. Disponível em: <http://elm-chan.org/docs/mmc/mmc_e.html>. Acesso em 10.dez.2006.

[4] MCP 4921/4922 datasheet: 12-Bit DAC with SPIŹ Interface. 2004. Disponível em: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1434>. Acesso em 07.dez.2006.

[5] RANDAZZO, A. *An RS-485 Based Interface with Lower Data Bit Errors*. [S.l.], Jun 2004. Disponível em: <http://www.st.com/stonline/products/literature/an/7628.pdf>. Acesso em 07.dez.2006.

[6] NOBREGA, C. M. da. Dissertação de Mestrado, *Arte robótica: Vida artificial para uma sociedade pós-biológica*. Brasília, Brasil: [s.n.], Novembro 2006.

[7] BREAZEAL, C. *Designing Sociable Robots*. Massachusetts, EUA: The MIT Press, 2004.

[8] MAXSONAR-EZ1 Datasheet: High Performance Sonar Range Finder. Jan 2006. Disponível em: <http://www.sandisk.com/Assets/File/OEM/Manuals/ProdManRS-MMCv1.3.pdf>. Acesso em 07.dez.2006.

[9] MARTINS, A. S.; ANDRADE, D. *Cinturão de Ultra-som para o robô Omni*. Dissertação (Projeto Final de Graduação em Engenharia Mecatrônica) — Universidade de Brasília, Brasília, Brasil, Fev 2005.

[10] SANDISK MultiMediaCard and Reduced-Size MultiMediaCard. [S.l.], Apr 2005. Acesso em 07.dez.2006.

[11] TBA820M datasheet: 1.2W audio amplifier. 1988. Disponível em: <http://www.datasheetcatalog.com/datasheets_pdf/T/B/A/8/TBA820M.shtml>. Acesso em 07.dez.2006.

[12] ATMEL ATmega (ATmega16) - MMC (Multi Media Card) Flash Memory Extension. Dec 2005. Disponível em: <http://www.captain.at/electronic-atmega-mmc.php>. Acesso em 07.dez.2006.

[13] BORGES, G. A.; MARTINS, A. S. *Introdução ao padrão físico RS-485 para comunicação serial*. Brasília, Brasil, Jun 2006. Disponível em: <http://www.ene.unb.br/ğaborges/recursos/notas/index.htm>. Acesso em 07.dez.2006.

[14] ATMEGA8 datasheet: 8-bit AVR with 8K Bytes In-System Programmable Flash. 2006. Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf>. Acesso em 07.dez.2006.

[15] ROSS, S. M. *Introduction to Probability Models*. Eighth. San Diego, EUA: Academic Press, 2003.

[16] LUENBERGER, D. G. *Introduction to Dynamic Systems: Theory, Models, and Applications*. Primeira edição. [S.l.]: Wiley, 1979.

[17] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Segunda edição. [S.l.]: Prentice-Hall, 2003.

[18] FRANKLIN, S.; GRAESSER, A. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: *Third International Workshop on Agent Theories, Architectures, and Languages*. [S.l.]: Springer-Verlag, 1997. p. 21–35.

[19] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Segunda edição. [S.l.]: Prentice-Hall, 1998.

[20] KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, 1996.

[21] SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Primeira edição. [S.l.]: The MIT Press, 1998.

[22] BERTSEKAS, D. *Dynamic Programming: Deterministic and Stochastic Models*. Primeira edição. [S.l.]: Prentice-Hall, 1987.

[23] BELLMAN, R. *Dynamic Programming*. [S.l.]: Princeton University Press, 1957.

[24] KONDA, V. R.; TSITSIKLIS, J. N. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, v. 42, n. 4, p. 1143–1166, 2003.

[25] JOEL, D.; NIV, Y.; RUPPIN, E. Actor-critic models of the basal ganglia: new anatomical and computational perspectives. *Neural Netw.*, Elsevier Science Ltd., Oxford, UK, UK, v. 15, n. 4, p. 535–547, June 2002. ISSN 0893-6080.

[26] SUTTON, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, v. 3, p. 9–44, 1988.

[27] LANGVILLE, A. N.; MEYER, C. D. Updating methods for finite markov chains. *J. Stat. Comp. and Simulation*, v. 11, p. 163–181, 1980.

[28] COTTA, G.; NETO, L. Projeto Final de Graduação em Engenharia Mecatrônica, *Concepção mecânica, acionamento e modelagem cinemática de um robô cachorro*. Brasília, Brasil: [s.n.], Dezembro 2006.

[29] PICARD, R. et al. Affective learning - a manifesto. *BT Technical Journal*, v. 22, n. 4, p. 253–269, 2004.

[30] AHN, H.; PICARD, R. Affective cognitive learning and decision making: The role of emotions. In: *Third International Workshop on Agent Theories, Architectures, and Languages*. Vienna, Austria: [s.n.].

[31] BLUMBERG, B. M. *Old tricks, new dogs: ethology and interactive creatures*. Tese (Doutorado) — Massachusetts Institute of Technology, MIT Media Lab, 1997. Supervisor-Pattie Maes.

[32] AHN, H.; PICARD, R. Affective cognitive learning and decision making: A motivational reward framework for affective agents. In: *The 1st International Conference on Affective Computing and Intelligent Interaction*. Beijing, China: [s.n.], 2005.

[33] KAPLAN, F. et al. Robotic clicker training. *Robotics and Autonomous Systems*, v. 38, n. 3-4, p. 197–206, 2002.

ANEXOS

I. CIRCUITOS IMPLEMENTADOS

Este anexo apresenta os circuitos implementados nos blocos de ultra-som I.1, de detecção de intensidade de som I.2 e reprodução de áudio I.3. Para cada um dos blocos, é mostrado os circuito analógico conectado ao AVR e o circuito referente à parte da comunicação serial. O circuito da comunicação serial segue o modelo apresentado por Borges *et al.* [13]. Deve-se notar a presença de *jumpers* no circuito. Eles são usados pelo fato de todos os blocos, incluindo aqueles referentes à parte de atuação, terem as resistências ligadas no barramento de comunicação. Desta forma, apenas o último dos blocos deve ter essas resistências ativas, pois caso todos tivessem essas resitências, o valor equivalente final paralelo das resistências de cada bloco, o que levaria a um curto circuito entre A, B, +5V e 0V.

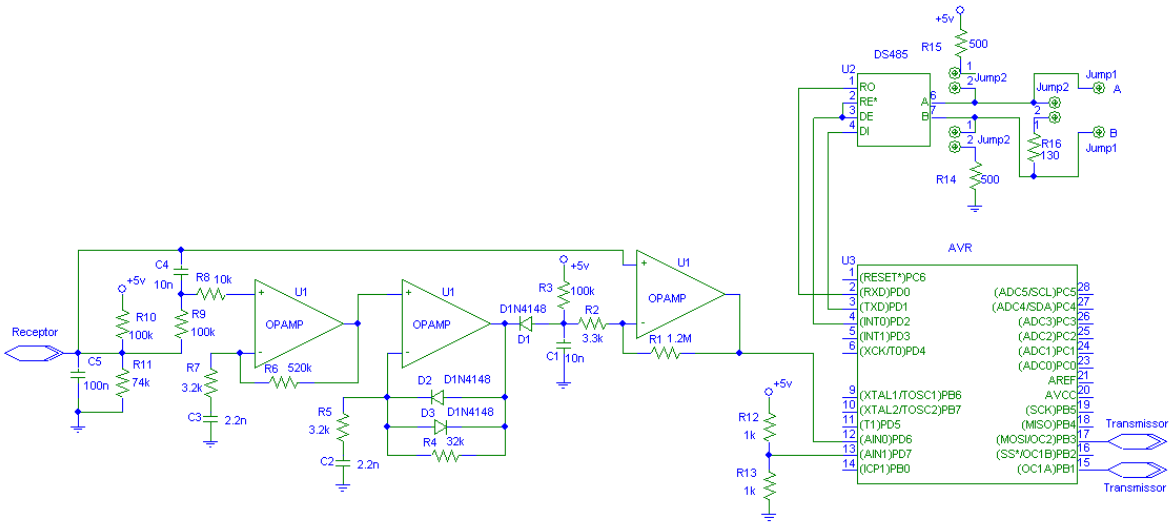


Figura I.1: Circuito completo do ultra-som

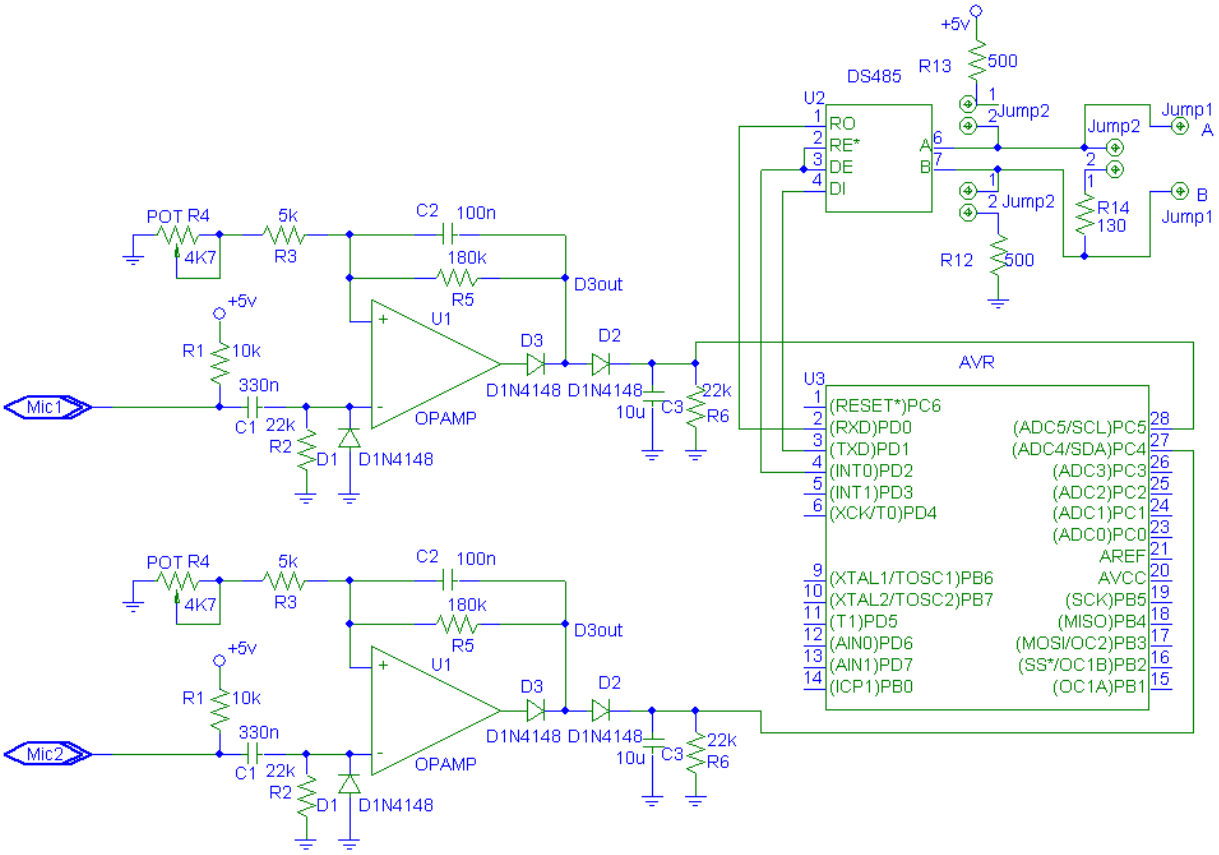


Figura I.2: Circuito completo do detector de intensidade sonora

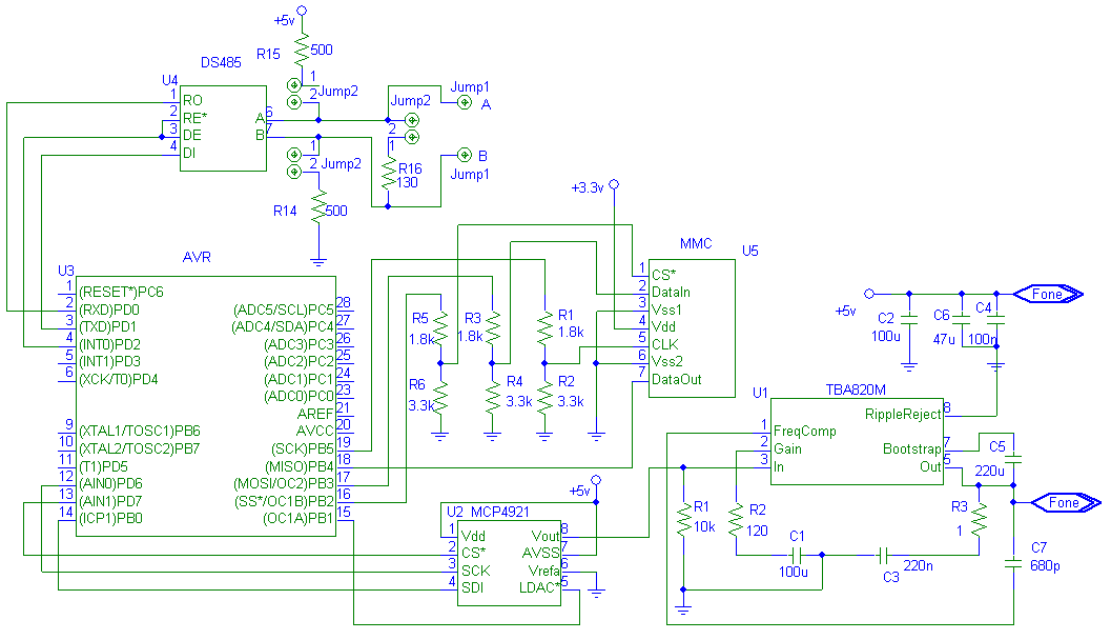


Figura I.3: Circuito completo da reprodução de áudio

II. DESCRIÇÃO DO CONTEÚDO DO CD

Em anexo, este relatório contém um CD com os códigos dos programas desenvolvidos. Os arquivos foram organizados em duas pastas, **AVRs** e **PC**. A primeira apresenta os códigos dos programas desenvolvidos para os microcontroladores de cada módulo, enquanto a outra apresenta os programas do sistema central, as simulações comportamentais e alguns programas de testes.

A pasta **AVRs** está dividida em subpastas, correspondentes a cada sensor/atuador desenvolvido por este grupo. Cada pasta contém um projeto completo feito no *software Programmer's Notepad 2*, compilador da ATMEL desenvolvido para programação de seus microcontroladores. Para programação, foi usada a linguagem C e houve a preocupação de, sempre que possível, utilizar interrupções. Em todos os módulos, foram usadas duas bibliotecas desenvolvidas pelo grupo, a **fifo_RX_TX.h** e a **Protocolo.h**, correspondentes, respectivamente, à criação das filas de entrada e saída e ao protocolo de comunicação. Cada módulo, à exceção do sensor de intensidade sonora, está organizado em:

- um arquivo principal com o nome do sensor/atuador ao qual se destina, onde encontra-se a função *main()*;
- "includes.h", com as definições e as bibliotecas a serem incluídas;
- "Init.c" e "Init.h", que contêm as funções para configuração de todas as utilidades do microcontrolador que serão usadas;
- "Interrupções.h", em que são declaradas todas as interrupções do correspondente módulo.

O sensor de intensidade sonora possui a biblioteca **STDAVR.h**, desenvolvida por Carlos Alberto Cascão Junior, que contém as configurações do microcontrolador, semelhantemente a "Init.c" e "Init.h".

A pasta **PC** está dividida nas subpastas **Sistema Central**, **Comportamento**, **Testes-Matlab**, **Testes-Dev** e **Memória**:

- **Sistema Central** - Programa desenvolvido em C para rodar em Linux e atuar como o módulo central, que recebe as informações dos sensores e indica o que os atuadores devem fazer, baseado no modelo comportamental, ou seja, realiza o processamento de alto-nível do robô.
- **Comportamento** - Simulação desenvolvida em C para testar as funções do módulo comportamental.
- **Testes-Matlab** - Funções desenvolvidas em Matlab para testar a comunicação com os módulos antes de o sistema central ficar pronto. Faz uso do padrão RS232.
- **Testes-Dev** - Funções desenvolvidas em C usando o compilador Dev-C++ para testar as funções do protocolo de comunicação. É uma forma fácil de obter o formato do dado protocolado e verificar se as mensagens que estão sendo recebidas/enviadas correspondem ao esperado.
- **Memória** - Programas para a escrita do arquivo de listagem com extensão .img para a memória MMC.