



UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Lázaro Ribeiro Monteiro Júnior

Sistema Embarcado para Identificação de Pistas de Rolamento

Mossoró, RN

2017

Lázaro Ribeiro Monteiro Júnior

Sistema Embarcado para Identificação de Pistas de Rolamento

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Leonardo Augusto Casillo, Prof. Dr.

Mossoró, RN

2017

© Todos os direitos estão reservados a Universidade Federal Rural do Semi-Árido. O conteúdo desta obra é de inteira responsabilidade do (a) autor (a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei nº 9.279/1996 e Direitos Autorais: Lei nº 9.610/1998. O conteúdo desta obra tomar-se-á de domínio público após a data de defesa e homologação da sua respectiva ata. A mesma poderá servir de base literária para novas pesquisas, desde que a obra e seu (a) respectivo (a) autor (a) sejam devidamente citados e mencionados os seus créditos bibliográficos.

M772s Monteiro Júnior, Lázaro Ribeiro.
Sistema Embarcado para Identificação de Pistas
de Rolamento / Lázaro Ribeiro Monteiro Júnior. -
2017.
49 f. : il.

Orientador: Leonardo Augusto Casillo.
Monografia (graduação) - Universidade Federal
Rural do Semi-árido, Curso de Ciência da
Computação, 2017.

1. Sistema Embarcado. 2. Arduino. 3. Rede
Neural. 4. Multilayer Perceptron. 5.
Reconhecimento de padrões. I. Casillo, Leonardo
Augusto, orient. II. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC's) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

Lázaro Ribeiro Monteiro Júnior

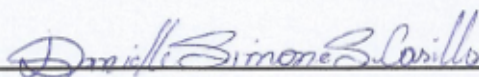
Sistema Embarcado para Identificação de Pistas de Rolamento

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência da Computação.

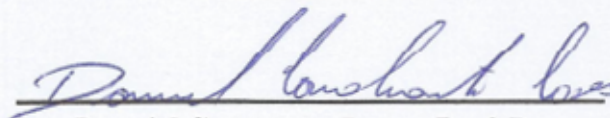
Trabalho aprovado. Mossoró, 19 de outubro de 2017:



Leonardo Augusto Casillo, Prof. Dr.
Orientador e Presidente



Danielle Simone da Silva Casillo, Prof. Dr.ª.
Primeiro membro



Danniell Cavalcante Lopes, Prof. Dr.
Segundo membro

Mossoró, RN
2017

*“A persistência é amiga da conquista.
Se você quer chegar onde a maioria
não chega, faça o que a maioria não faz.”
(Bill Gates)*

Resumo

O solo sobre o qual um veículo automotor trafega é um dos principais fatores que afetam a sua dirigibilidade e, conseqüentemente, sua segurança. Em busca de contribuir com a evolução dos automotores, este trabalho apresenta o desenvolvimento de um sistema embarcado que utiliza uma rede neural *multilayer perceptron* para identificar pistas de rolamento, classificando-as em asfalto, calçamento com pedras ou areia. As redes neurais são amplamente utilizadas para reconhecimento de padrões, pois de forma semelhante à inteligência natural, elas são capazes de aprender a partir de exemplos e se tornam aptas a distinguir dados de diferentes classes. Contudo, desenvolver tal aplicação em um sistema embarcado se torna um desafio devido aos recursos limitados. Utilizando um Arduino e um sensor RGB, o hardware concebido coleta informações da cor da superfície sobre a qual o sensor se desloca e identifica o tipo de pista de rolamento. Desta forma, o dispositivo pode fornecer informações importantes à sistemas já existentes, permitindo que eles tomem as decisões mais adequadas para se adaptar à situação em que se encontram.

Palavras-chave: Sistema Embarcado. Arduino. Rede Neural. *Multilayer Perceptron*. Reconhecimento de padrões.

Abstract

The ground on which a motor vehicle is traveling is one of the main factors that affect its handling and, consequently, its safety. In order to contribute to the evolution of the automotive industry, this work presents the development of an embedded system that uses a multilayer perceptron neural network to identify driving lanes, classifying them in asphalt, stone paving or sand. Neural networks are widely used for pattern recognition, because similarly to natural intelligence, they are able to learn from examples and become able to distinguish data from different classes. However, developing such an application in an embedded system becomes a challenge due to the limited resources. Using an Arduino and an RGB sensor, the designed hardware collects informations about the color of the surface on which the sensor moves and identifies the type of driving lane. Therefore, the device can provide important information to existing systems, allowing them to make the most appropriate decisions to adapt to the situation they are in.

Keywords: Embedded System. Arduino. Neural Network. Multilayer Perceptron. Pattern Recognition.

Lista de ilustrações

2.1.1	Diagrama do corte transversal do olho humano.	14
2.1.2	Espectro eletromagnético.	15
2.1.3	Cubo de cores RGB (a) e seu respectivo diagrama de cores (b).	16
2.2.1	Estrutura do neurônio biológico.	18
2.2.2	Neurônio artificial.	18
2.2.3	Exemplo de rede <i>Perceptron</i> com apenas um neurônio.	19
2.2.4	Exemplo de rede <i>Multilayer Perceptron</i> com uma camada oculta.	21
2.3.1	Arduino Micro.	24
3.1.1	Organização do sistema	29
3.1.2	Diagrama de blocos do <i>hardware</i>	30
3.1.3	Organização da placa de circuitos	31
3.1.4	Arduino Pro Micro Compatível	32
3.1.5	Sensor de cores TCS230	32
3.1.6	Módulo leitor de cartão Micro SD	32
3.1.7	Esquema elétrico	33
3.1.8	Placa de circuitos	34
3.1.9	Diagrama de blocos do <i>software</i> coletor de amostras	35
3.1.10	Diagrama de classes do <i>software</i> coletor de amostras	35
3.1.11	Diagrama de sequência do <i>software</i> coletor de amostras	36
3.1.12	Máquina de estados do <i>software</i> coletor de amostras	36
3.1.13	RNA <i>multilayer perceptron</i>	37
3.1.14	Diagrama de blocos do <i>software</i> MLP	38
3.1.15	Diagrama de classes do <i>software</i> MLP	39
3.1.16	Diagrama de blocos do <i>software</i> classificador	40
3.1.17	Diagrama de classes do <i>software</i> classificador	41
3.1.18	Diagrama de sequência do <i>software</i> classificador	42
3.1.19	Máquina de estados do <i>software</i> classificador	42
4.0.1	Gráfico da curva de aprendizagem	46

Lista de tabelas

3.1.1 Formulário de requisitos	27
4.0.1 Quantidade de neurônios da camada oculta	45

Lista de abreviaturas e siglas

ABS	<i>Anti-lock Braking System</i>
CPU	<i>Central Processing Unit</i>
DC	<i>Direct Current</i>
GPL	<i>General Public License</i>
MLP	<i>Multilayer Perceptron</i>
PC	<i>Personal Computer</i>
PTH	<i>Pin Throught Hole</i>
RGB	<i>Red Green Blue</i>
RNA	Rede Neural Artificial
SMD	<i>Surface-Mount Device</i>
SoC	<i>System on a Chip</i>
TCS	<i>Traction Control System</i>
USB	<i>Universal Serial Bus</i>
VLSI	<i>Very Large Scale Integration</i>

Lista de símbolos

δ	Letra grega minúscula Delta
θ	Letra grega minúscula Theta
$\in$	Pertence
Σ	Letra grega maiúscula Sigma

Sumário

1	INTRODUÇÃO	11
1.1	Problema	11
1.2	Objetivos	12
1.3	Justificativa	12
1.4	Organização	12
2	REFERENCIAL TEÓRICO	14
2.1	A Visão Humana e Sistema de Cores RGB	14
2.1.1	Sistema de Cores RGB	16
2.2	Redes Neurais	17
2.2.1	Neurônios Biológicos e o Modelo Matemático	17
2.2.2	Estrutura (Classificação)	19
2.2.3	Perceptron	19
2.2.4	Multilayer Perceptron (MLP)	20
2.2.5	Algoritmo de Treinamento Backpropagation	21
2.3	Sistemas Embarcados	23
2.3.1	Plataforma Arduino	23
3	MATERIAIS E MÉTODOS	26
3.1	Projeto do Sistema	26
3.1.1	Requisitos	26
3.1.2	Especificação	28
3.1.3	Arquitetura e Organização	29
3.1.3.1	<i>Hardware</i>	30
3.1.3.2	Coletor de Amostras	34
3.1.3.3	Implementação da Rede Neural <i>Multilayer Perceptron</i>	36
3.1.3.4	Software Classificador de Pistas de Rolamento	40
3.2	Método de Amostragem	43
3.3	Processo de Aprendizagem	43
4	RESULTADOS E TESTES	45
5	CONSIDERAÇÕES FINAIS	47
5.1	Trabalhos Futuros	48
	REFERÊNCIAS	49

1 Introdução

Com o avanço da tecnologia, já se considera possível projetar um veículo automotor de passeio (automóvel ou motocicleta) que possa se adaptar ao terreno em que trafega. No asfalto ele ficaria mais baixo, com pneus mais rígidos, frenagem rápida e perfeita estabilidade. Ao passar para uma estrada de terra, este mesmo veículo poderia se tornar mais alto, reduzir a pressão interna dos pneus para melhor absorver as trepidações, frear eficientemente sem travamentos, proporcionando mais conforto e estabilidade, mesmo em uma via de má qualidade.

Com base nestas informações, outros sistemas já existentes, como ABS (*Anti-lock Braking System*), TCS (*Traction Control System*) e suspensões eletromagnéticas, seriam capazes de realizar algumas dessas adaptações. Neste trabalho será desenvolvido o protótipo de um sensor simples, de baixo custo, que deve classificar a pista de rolamento como asfalto, calçamento ou areia. O veículo que utilizar este sensor proposto junto aos atuadores já incluídos poderá ser capaz de se adaptar melhor ao cotidiano brasileiro e seus milhares de quilômetros de rodovias nas mais diversas condições de uso.

1.1 Problema

Um dos fatores que mais influi na dirigibilidade de qualquer veículo automotor é o terreno sobre o qual este veículo trafega, pois a força de atrito entre os pneus e o solo é a única força atuante que impulsiona o veículo à frente. O tipo de pavimento utilizado na pista de rolamento, ou a falta deste, implica em diferentes coeficientes de atrito, alteração da superfície de contato entre o pneu e o solo, quantidade de energia mecânica dissipada na pista e na temperatura interna dos pneus (SILVEIRA, 2011).

O ABS foi criado na década de 1970 e atualmente é item de segurança obrigatório nos automóveis e motocicletas acima de 300cc (Resoluções 380/2011 e 509/2014 do Contran). Este sistema antitravamento de freios monitora a velocidade de cada pneu individualmente e atua reduzindo a força do freio conforme a necessidade, de modo a evitar o travamento das rodas (BURTON et al., 2004). Contudo, o ABS, assim como outros assistentes de direção, precisa ser calibrado para atuar com eficiência, pois dependendo do tempo de resposta e força reduzida nos freios, o assistente pode permitir travamento das rodas ou impedir o motorista de realizar uma frenagem eficiente.

Os automóveis atuais já possuem alguns assistentes de direção que proporcionam melhor dirigibilidade, mais segurança e mais estabilidade ao condutor. Porém, no cotidiano brasileiro existem diversos tipos de pistas de rolamento, com diferentes pavimentações, ou até desprovidas desta. Devido aos fatos apresentados, muitas vezes carros desenvolvidos para uso em pistas com pavimentação asfáltica apresentam baixa eficiência em vias calçadas com pedras ou em estradas

de areia, produzindo muitos ruídos, trepidações, travamentos de rodas durante frenagens, etc. Tais problemas causam desconforto aos motorista e passageiros, além de dificultar a condução e reduzir a segurança do veículo.

1.2 Objetivos

Este trabalho acadêmico tem como objetivo desenvolver um sistema embarcado, de baixo custo, para a identificação de pistas de rolamento. Funcionando como um sensor, o sistema proposto permite identificar se o veículo se encontra sobre uma pista com pavimentação asfáltica, calçada com pedras ou sobre areia.

Os principais objetivos deste trabalho são:

- Desenvolver o protótipo de um sistema embarcado que seja capaz identificar a superfície da pista de rolamento em que se trafega;
- Coletar dados utilizando um sensor de cores;
- Utilizar técnicas de classificação de dados para classificar os dados coletados na entrada;
- Prover uma saída que possa ser utilizada por outros sistemas.

1.3 Justificativa

Os veículos automotores vem revolucionando o mundo no decorrer deste último século. Cada vez mais populares e presentes no cotidiano, se tornaram itens essenciais para os moradores dos grandes centros urbanos. No decorrer do tempo, os automotores vem evoluindo progressivamente, melhorando o *design*, o conforto interno, a eficiência dos motores e, principalmente, a segurança.

Os automóveis modernos contam com centenas de sistemas embarcados, dos quais grande parte são destinados a garantir a segurança dos tripulantes do veículo, como ABS, TCS, *Airbags*, freios autônomos, dentre outros. O sistema proposto busca se integrar a este conjunto e pode auxiliar os sistemas existentes a tomarem decisões mais adequadas e precisas, contribuindo para uma condução mais agradável e segura dos veículos automotores.

1.4 Organização

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta os aspectos mais relevantes da teoria utilizada na elaboração deste trabalho. Nele, são abordados os seguintes assuntos: a visão humana, o sistema de cores RGB, redes neurais, sistemas embarcados e a plataforma Arduino; o capítulo 3 aborda o projeto e o desenvolvimento do sistema embarcado,

explica seu funcionamento e expõe detalhes de seus *hardware* e *softwares*; no capítulo 4 são apresentados os testes realizados e seus resultados, medindo o desempenho e a eficácia do dispositivo em diferentes situações; o capítulo 5 apresenta as conclusões para o sistema embarcado realizado e faz um breve balanço sobre os resultados obtidos. Também são definidos direcionamentos para trabalhos futuros, que devem evoluir e aprimorar o produto.

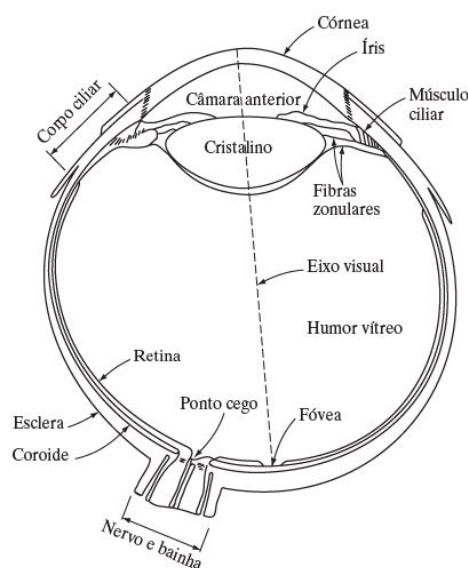
2 Referencial Teórico

Para cumprir os objetivos definidos no capítulo anterior, é necessário ter conhecimento prévio de algumas áreas da ciência. Logo, será feita uma sucinta revisão bibliográfica dos assuntos essenciais à compreensão deste trabalho acadêmico. Neste capítulo, são apresentados os aspectos básicos da visão humana, a representação das cores no sistema RGB, o funcionamento das redes neurais, os conceitos de sistemas embarcados e a plataforma de prototipagem Arduino.

2.1 A Visão Humana e Sistema de Cores RGB

Os olhos são os órgãos responsáveis pela visão, um importante sentido que aprimora a percepção do mundo, possibilitando o reconhecimento de obstáculos, objetos, outros indivíduos, possíveis ameaças, etc. No ser humano, o olho possui formato esférico e diâmetro médio de aproximadamente 20 mm. Na retina, ocorre a percepção das imagens que se dá através de duas classes de receptores distribuídos ao longo de sua superfície, os *cones* e os *bastonetes*. Os cones são os receptores responsáveis pela visão *fotópica* e são encarregados de distinguir pequenos detalhes e da percepção das cores. Já os bastonetes são responsáveis pela visão *escotópica* e proporcionam uma imagem geral do campo de visão, pois não são capazes de discernir detalhes, nem são sensíveis a cores. Contudo, os bastonetes estão presentes em muito maior quantidade, estão distribuídos em uma área mais ampla da córnea e são sensíveis a baixos níveis de iluminação, logo, eles são responsáveis pela visão em ambientes escuros. Na Figura 2.1.1 é apresentado o diagrama do corte transversal do olho humano (GONZALEZ; WOODS, 2009, p.23-28).

Figura 2.1.1 – Diagrama do corte transversal do olho humano.

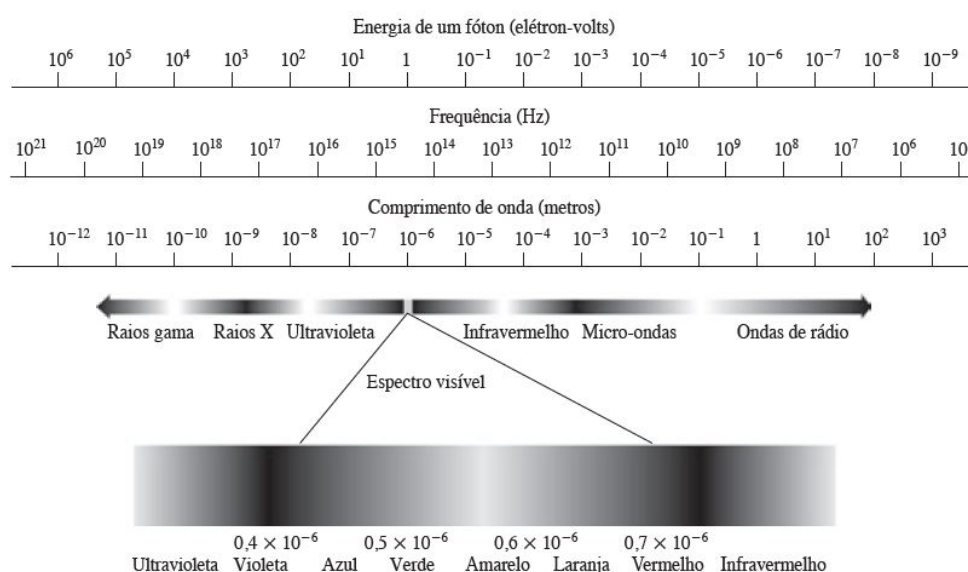


Fonte: (GONZALEZ; WOODS, 2009, p.23)

A luz visível ao ser humano é um tipo específico de radiação eletromagnética que cobre uma pequena gama do espectro de energia na faixa de $0,43$ a $0,79 \mu m$. As cores percebidas pelo olho humano em um objeto é resultante do reflexo da luz emitido pelo mesmo, que retém parte da radiação eletromagnética e reflete o restante. A luz observada atravessa o cristalino e atinge a retina, excitando os receptores (cones e bastonetes) presentes em sua superfície. Esta excitação causada pela luz é então convertida em impulsos nervosos que são transmitidos ao cérebro, onde os sinais são “decodificados” através de um fenômeno fisiopsicológico. A luz colorida é chamada de luz *cromática*, enquanto a luz sem cor é chamada de *monocromática* (GONZALEZ; WOODS, 2009).

A Figura 2.1.2 apresenta o espectro eletromagnético, evidenciando que o espectro visível cobre apenas uma pequena parte deste. Ampliando o espectro visível, nota-se que ele se inicia na cor violeta e termina na cor vermelha, sendo subdividido em seis faixas principais separadas por transições gradativas de cores. Estas faixas são: violeta, azul, verde, amarelo, laranja e vermelho (GONZALEZ; WOODS, 2009, p.23-28).

Figura 2.1.2 – Espectro eletromagnético.



Fonte: (GONZALEZ; WOODS, 2009, p.28)

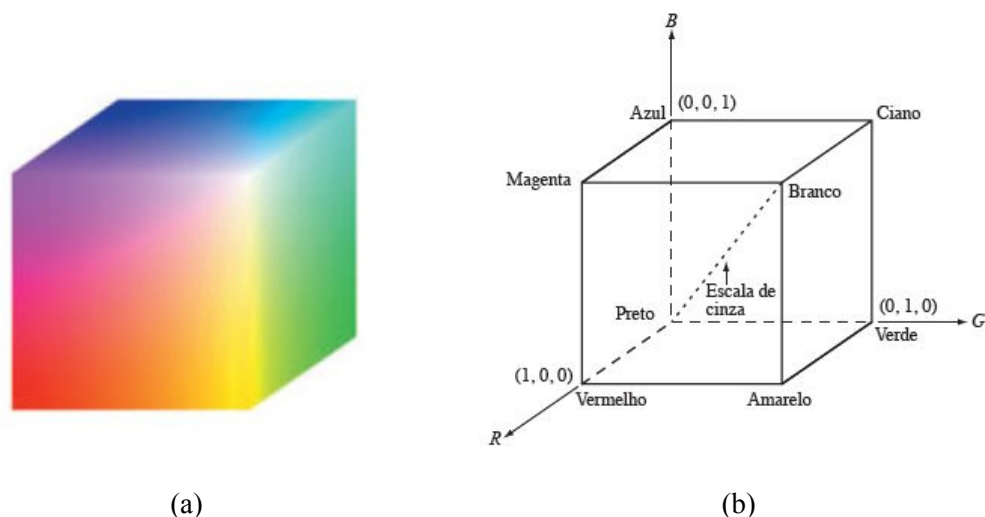
Apesar do olho ser o órgão responsável por capturar a luz em suas diferentes cores, é o cérebro que provê a inteligência e, consequentemente, a faculdade de aprender ou compreender. O cérebro recebe diferentes estímulos que proporcionam inúmeras sensações e experiências. Quando essas informações são relevantes, elas são memorizadas. Logo, o ser humano é capaz de associar novas imagens às guardadas na memória e dessa forma reconhecer ambientes ou objetos a partir de suas formas e/ou cores, o que é uma característica da capacidade de aprendizagem (GONZALEZ; WOODS, 2009, p.24, 269-264) (ARTERO, 2008, p.18-19).

2.1.1 Sistema de Cores RGB

Conforme evidências experimentais, os cones presentes no olho humano podem ser divididos em três principais categorias: cerca de 65% deles são sensíveis à luz vermelha, 33% são sensíveis à luz verde e 2% são sensíveis à luz azul. Devido a essa característica dos humanos, é possível tratar todas as cores do espectro visível como combinações das cores primárias: vermelho, verde e azul (em inglês: red, green e blue, respectivamente). (GONZALEZ; WOODS, 2009, p.259-268).

O modelo de cores RGB especifica um sistema de coordenadas e um subespaço contido nesse sistema, no qual cada cor é representada por um ponto. Cada ponto é composto por seus componentes espectrais primários de vermelho, verde e azul e é posicionado no espaço através de um sistema de coordenadas cartesianas. Este subespaço de cores é representado pelo cubo RGB apresentado na Figura 2.1.3 (a). Conforme o esquema à direita do cubo (Figura 2.1.3 (b)), os valores RGB primários se encontram em três vértices, enquanto as cores secundárias (magenta, ciano e amarelo) estão em outros três vértices. O preto se localiza na origem e o branco no vértice mais distante à origem. A escala de cinza é representada por um segmento de reta que liga o vértice preto ao vértice branco. As demais cores do espectro visível são pontos na superfície do cubo ou em seu interior e são definidas por vetores que se estendem a partir da origem, portanto, cada vetor possui os três componentes R, G e B em diferentes intensidades. O número de bits utilizados para representar cada pixel em espaço RGB é chamado de profundidade de pixel (GONZALEZ; WOODS, 2009, p.259-268).

Figura 2.1.3 – Cubo de cores RGB (a) e seu respectivo diagrama de cores (b).



Fonte: (GONZALEZ; WOODS, 2009, p.265-266)

Dessa forma, o modelo RGB é um dos mais utilizados para representar numericamente as cores. Por se tratar de um modelo aditivo, ele é amplamente empregado em dispositivos geradores de imagens que possuem fundo preto, como por exemplo *displays*, monitores e televisores.

2.2 Redes Neurais

As redes neurais artificiais (RNA's) foram desenvolvidas na década de 1940 pelo neurofisiologista McCulloch e pelo matemático Walter Pitts. Eles pretendiam emular o comportamento do cérebro humano utilizando componentes de hardware para simular o funcionamento de um neurônio e conectá-los de forma a constituir uma rede, de maneira análoga ao sistema nervoso humano. (AGUIAR et al., 2007, p.67-71), (ARTERO, 2008, p.123-126), (LUDWIG JR; MONTGOMERY, 2007, p.9-14).

Embora não tenha atendido às altas expectativas da época, décadas depois as redes neurais passaram a ser utilizadas para resolver problemas de reconhecimento de padrões, pois mesmo que não sejam comparáveis ao cérebro humano em muitos aspectos, elas são capazes de aprender e reconhecer padrões, sendo eficientes para distinguir vozes, imagens e objetos em geral (ARTERO, 2008, p.123-126). De maneira análoga, as redes neurais também são aptas à diferenciar cores.

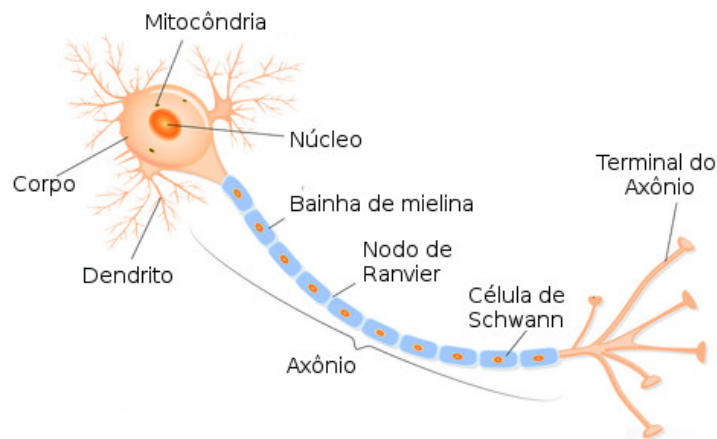
As RNA's são compostas por unidades denominadas neurônios artificiais que são conectadas por ligações direcionadas que propagam os sinais de ativação. Cada ligação tem um peso numérico w_i associado que determina a força e o sinal de conexão. (NORVIG; RUSSELL, 2013, 635-637)

2.2.1 Neurônios Biológicos e o Modelo Matemático

A estrutura dos neurônios biológicos pode ser dividida essencialmente em *dendritos*, *corpo* e *axônio*. Os dendritos são responsáveis de receber os estímulos provenientes de outros neurônios; o corpo contém o núcleo da célula, coleta e combina as informações recebidas pelos dendritos; caso os estímulos recebidos atinjam um determinado limiar de ação, um novo estímulo é transmitido através do axônio para as demais células. Os neurônios se conectam através de sinapses, sinais elétricos com cerca de 70 milivolts. O sistema nervoso é formado por uma rede extremamente complexa de neurônios que ultrapassa 100 bilhões de conexões.

Cada neurônio recebe sinais através de vários dendritos (entradas), os pondera e envia ao axônio, podendo ou não seguir adiante (saída). Durante sua passagem, o sinal pode ser atenuado ou amplificado, conforme a sinapse envolvida. Após vivenciar alguma nova experiência, o cérebro é capaz de alterar as conexões, ou no caso de experiências repetidas, as conexões são reforçadas, caracterizando o aprendizado. A Figura 2.2.1 representa a estrutura de um neurônio biológico (ARTERO, 2008, p.123-126), (LUDWIG JR; MONTGOMERY, 2007, p.9-14).

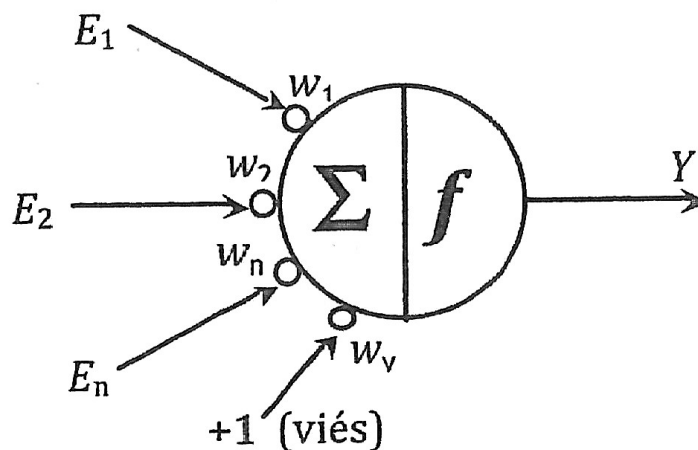
Figura 2.2.1 – Estrutura do neurônio biológico.



Fonte: Infoescola ¹

O modelo matemático proposto para o neurônio artificial (Figura 2.2.2) funciona de maneira análoga ao biológico, possuindo múltiplas entradas E_i e uma saída Y . O corpo pode ser dividido em duas partes: camada de entrada, responsável por somar os produtos das entradas E_i por seus respectivos pesos w_i e a função de ativação f que controla o valor fornecido na saída Y . Além das entradas E_i pode ser utilizada, opcionalmente, uma entrada de viés (ou bias) w_v inicializada com o valor 1 (ARTERO, 2008, p.123-126), (LUDWIG JR; MONTGOMERY, 2007, p.9-14).

Figura 2.2.2 – Neurônio artificial.



Fonte: (ARTERO, 2008, p.125)

Os neurônios artificiais são unidades básicas e também podem ser conectados de forma a constituir uma rede, onde o sinal saída Y de cada neurônio pode ser utilizado como sinal de entrada E_i para um ou mais neurônios da camada posterior ou como sinal de saída da rede. Os

¹ Disponível em <<http://www.infoescola.com/sistema-nervoso/neuronios/>>. Acesso em agosto de 2017.

pesos atribuídos às conexões funcionam como a memória e são ajustados durante a etapa de treinamento, quando ocorre o aprendizado (ARTERO, 2008, p.123-126), (LUDWIG JR; MONTGOMERY, 2007, p.9-14).

2.2.2 Estrutura (Classificação)

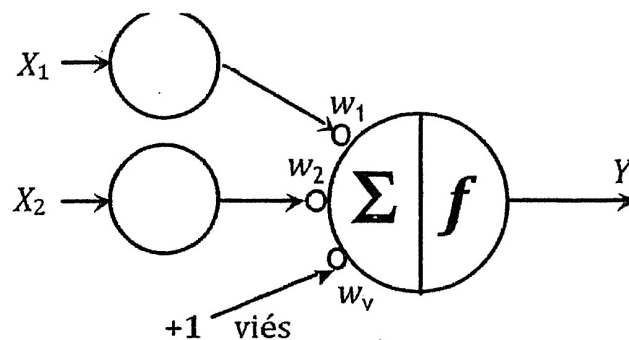
Quanto ao número de camadas, a rede pode possuir duas ou múltiplas camadas, já que a primeira camada apenas efetua a distribuição dos dados e não efetua processamento. A respeito da conectividade da rede, quando os neurônios de uma camada estão conectados a todos os neurônios da camada seguinte, diz-se que a rede está completamente conectada. Caso contrário, a rede está parcialmente conectada. Quando o fluxo de dados se desloca sempre da entrada para a saída, se trata de uma rede de alimentação adiante (*feedforward*). Já quando um neurônio pode receber como entrada saídas de camadas posteriores, se trata de uma realimentação (*feedbackward*). (ARTERO, 2008, p.126-127)

O treinamento pode ser *supervisionado*, quando o ajuste de parâmetros é feito com base na apresentação de um conjunto de pares de entradas e saídas correspondentes, ou *não-supervisionado*, quando o conjunto de treinamento possui somente entradas e utiliza a comparação entre sinais para a construção de grupos semelhantes. A conectividade da rede pode apresentar diversas configurações como *feedforward* ou recorrente, quando um neurônio pode receber entradas de neurônios de qualquer outra camada. (AGUIAR et al., 2007)

2.2.3 Perceptron

A rede perceptron foi criada em 1957 por Frank Rosenblat e reacendeu o interesse pelas RNAs. Este tipo de RNA utiliza treinamento supervisionado e possui uma camada de entrada, não processadora, que realiza a distribuição dos sinais de entrada para as unidades na segunda camada, que é composta por neurônios processadores, ou seja, há apenas uma camada processadora na rede que se conecta diretamente às saídas. (ARTERO, 2008, p.130-133), (NORVIG; RUSSELL, 2013, p.636-637).

Figura 2.2.3 – Exemplo de rede *Perceptron* com apenas um neurônio.



Fonte: (ARTERO, 2008, p.131)

O cálculo da variável *net* representa o somatório dos produtos das entradas E_i por seus respectivos pesos w_i e pode ser obtido com a Equação 2.1. Opcionalmente, pode-se utilizar uma entrada de viés E_v com seu respectivo peso w_v . O erro δ da rede é dado pela diferença do valor desejado na saída pelo valor fornecido pela rede (Equação 2.2). O aprendizado ocorre alterando-se os valores dos pesos a partir dos erros δ , conforme a Equação 2.3, sendo n uma constante referente à taxa de aprendizado (ARTERO, 2008, p.130-133).

$$net = \sum_{i=1}^n E_i \cdot w_i + E_v \cdot w_v \quad (2.1)$$

$$\delta = d - Y \quad (2.2)$$

$$w_{i+1} = w_i + n \cdot \delta \cdot E_i \quad (2.3)$$

Dado ao fatos das redes Perceptron utilizarem funções lineares como funções de ativação, há uma limitação na identificação dos objetos, sendo possível distingui-los apenas quando eles pertencem a classes linearmente separáveis, já que a separação entre as classes é uma função linear (ARTERO, 2008, p.130-133).

2.2.4 Multilayer Perceptron (MLP)

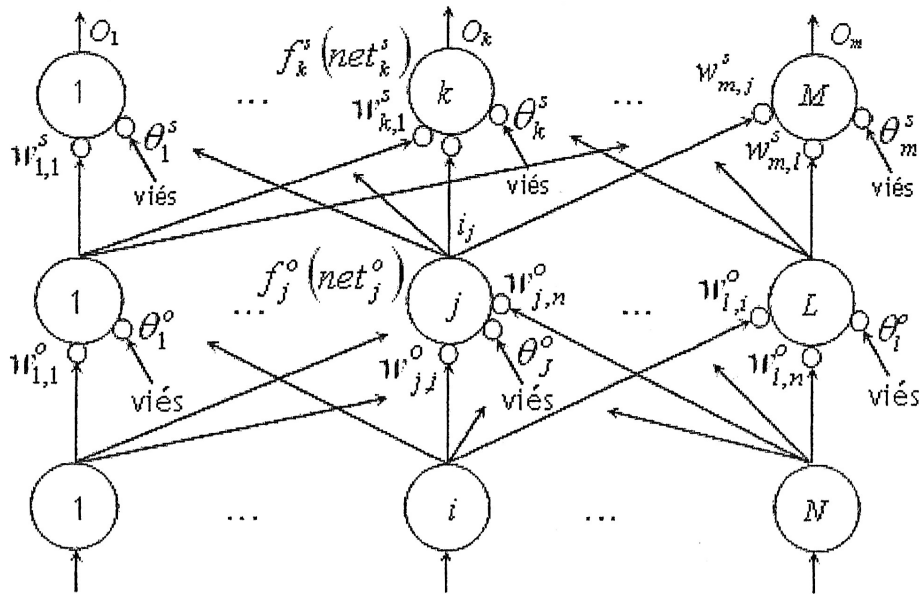
A rede MLP (*Multilayer Perceptron*) é uma generalização da rede Perceptron que surgiu com a necessidade de imitar o funcionamento do cérebro humano. Seu conceito consiste em introduzir uma ou mais camadas de neurônios internas entre as camadas de entrada e saída da rede, funcionando como uma sequência de *Perceptrons* simples interconectados, de forma a gerar a propagação dos sinais de entrada atravessando todas as camadas até a saída da rede. Isto permitiu que as redes passassem a resolver problemas de classificação não lineares (ARTERO, 2008, p.134-139), (LUDWIG JR; MONTGOMERY, 2007, p.45-58).

As redes MLP possuem uma camada de entrada, uma ou mais camadas ocultas e uma camada saída. A camada de entrada possui um nó para cada sinal de entrada, enquanto a camada de saída possui um nó para cada sinal de saída. As quantidades de camadas internas e de seus respectivos nós devem ajustadas arbitrariamente de modo a se obter um resultado satisfatório. O processamento das unidades neuronais de uma mesma camada é paralelo, enquanto o processamento de camada para camada é sequencial (ARTERO, 2008, p.134-139), (LUDWIG JR; MONTGOMERY, 2007, p.45-58).

As funções de transferência mais utilizadas são as linear, logística e tangente hiperbólica, pois são diferenciáveis em qualquer valor do domínio. O treinamento, assim como nas redes Perceptron, é supervisionado e busca minimizar o erro da saída (ARTERO, 2008, p.134-139), (LUDWIG JR; MONTGOMERY, 2007, p.45-58).

A Figura 2.2.4 apresenta uma rede *Multilayer Perceptron* com N neurônios na camada de entrada, L neurônios na camada oculta e M neurônios na camada de saída. $f_j^o(net_j^o)$ e $f_k^s(net_k^s)$ representam as funções de transferência das camadas oculta e de saída respectivamente.

Figura 2.2.4 – Exemplo de rede *Multilayer Perceptron* com uma camada oculta.



Fonte: (ARTERO, 2008, p.135)

2.2.5 Algoritmo de Treinamento Backpropagation

O *Backpropagation*, também conhecido como *regra delta generalizada*, é o algoritmo de treinamento mais utilizado em redes MLP com alimentação adiante. O erro obtido na camada de saída é retro propagado pela rede em direção à camada de entrada para ajustar os pesos nos neurônios das camadas ocultas. Cada nó oculto é responsável por uma fração do erro em cada um dos nós de saída aos quais ele se conecta. Para retro propagar o erro é necessário calcular a derivada do gradiente do erro geral (AGUIAR et al., 2007, p.90-91), (ARTERO, 2008, p.134-139), (NORVIG; RUSSELL, 2013, p.639-642).

A função de transferência da camada oculta é indicada por $f_j^o(net_j^o)$ e a da camada de saída é indicada por $f_k^s(net_k^s)$. Os pesos das camadas oculta, camada de saída e de viés são dados, respectivamente, por w_{ji}^o , w_{kj}^s e w_v . As entradas de viés são representadas por θ_j^o e θ_k^s , a taxa de aprendizado da rede é dada pela constante n e o erro é indicado por δ (ARTERO, 2008, p.134-139).

O processo de treinamento, utilizando o *Backpropagation*, é realizado através dos passos descritos a seguir:

1. Inicializa os pesos com valores aleatórios;

2. Aplica o vetor de entradas $X_1, X_2, \dots, X_n$;

3. Calcula-se os *nets* dos neurônios da camada oculta, para cada $j \in (1, L)$:

$$net_j^o = \sum_{i=1}^N w_{j,i}^o \cdot x_i + \theta_j^o \cdot w_v^o \quad (2.4)$$

4. Aplica a função de transferência para obter as saídas i_j da camada oculta:

$$i_j = f_j^o(net_j^o) \quad (2.5)$$

5. Calcula os *nets* dos neurônios da camada oculta, para cada $k \in (1, M)$:

$$net_k^s = \sum_{j=1}^L w_{kj}^s \cdot i_j + \theta_k^s \cdot w_v^s \quad (2.6)$$

6. Calcula as saídas O_k dos neurônios da camada de saída:

$$O_k = f_k^s(net_k^s) \quad (2.7)$$

7. Calcula os erros para os neurônios da camada de saída:

$$\delta_k^s = (d_k - O_k) \left(f_k^{s'}(net_k^s) \right) \quad (2.8)$$

onde d_k é valor desejado na saída e $f_k^{s'}(net_k^s)$ é a derivada da função de transferência do neurônio k da camada de saída;

8. Calcula-se os erros nos neurônios da camada oculta, para cada $j \in (1, L)$:

$$\delta_j^o = f_j^{o'}(net_j^o) \sum_{k=1}^M \delta_k^s \cdot w_{k,j} \quad (2.9)$$

9. Atualiza os pesos da camada de saída:

$$\begin{aligned} w_{kj}^s &= w_{kj}^s + n \cdot \delta_k^s \cdot i_j \\ \theta_k^s &= \theta_k^s + n \cdot \delta_k^s \end{aligned} \quad (2.10)$$

10. Atualiza os pesos da camada oculta:

$$\begin{aligned} w_{ij}^o &= w_{ij}^o + n \cdot \delta_j^o \cdot x_i \\ \theta_j^o &= \theta_j^o + n \cdot \delta_j^o \end{aligned} \quad (2.11)$$

11. Calcula o erro da rede fazendo:

$$E_r = \frac{1}{2} \sum_{k=1}^M (\delta_k^s)^2 \quad (2.12)$$

Cada amostra de treinamento deve passar por todos os passos descritos até que se obtenha um erro de rede abaixo do limiar ou se atinja um número predefinido de iterações (ARTERO, 2008, p.135-136).

2.3 Sistemas Embarcados

Sistema embarcado é qualquer dispositivo que inclui um computador programável, mas não tem o propósito de ser um computador de uso geral. Eles são utilizados em diversos tipos de produtos, como automóveis, telefones celulares, utensílios domésticos, dentre outros (WOLF, 2008, p.1-10).

Desde a década de 1940 computadores vem sendo embarcados em aplicações específicas, mesmo que nesse período eles ainda ocupassem um grande espaço físico e consumissem muita energia. Na década de 1970, com o surgimento da tecnologia VLSI (Very Large Scale Integration), tornou-se possível colocar uma CPU (*Central Processing Unit*) inteira em um único chip, que passou a ser conhecido como “microprocessador”. Após estes, outro grande marco foi a criação de CPU’s de uso geral, permitindo que se utilize um mesmo *hardware* para diversas aplicações, sendo necessário alterar apenas o *software* (WOLF, 2008, p.1-10).

Microprocessadores são uma maneira muito eficiente para implementar sistemas digitais, pois permitem uma implementação mais rápida da aplicação, oferecem boa performance, baixo consumo energético e baixo custo. Eles executam programas com boa eficiência e podem ser utilizados com diferentes tipos de algoritmos. Sem dúvidas, a programabilidade dos microprocessadores é uma característica fundamental (WOLF, 2008, p.1-10).

O projeto de uma aplicação embarcada, geralmente, é mais complexo que de uma aplicação para PC (*Personal Computer*), pois é necessário lidar com diversas limitações. O *hardware* utilizado deve ser o mínimo possível, o *software* precisa ser otimizado para obter eficiência e o consumo de energia deve ser baixo, principalmente quando alimentado por baterias. Apesar de todas as restrições, o sistema embarcado precisa ser confiável e executar suas tarefas com bom desempenho, muitas vezes em tempo real e/ou com interface gráfica para usuários (WOLF, 2008, p.1-10).

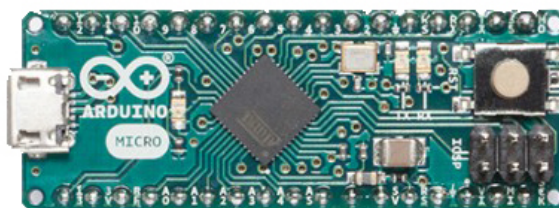
2.3.1 Plataforma Arduino

O Arduino é uma plataforma de prototipagem eletrônica *open source* que foi desenvolvida em 2005, no *Ivrea Interaction Design Institute*, no norte da Itália. Além de seguir os prin-

cípios de código aberto, também é baseada no conceito *easy-to-use*, que garante flexibilidade e boa usabilidade de *hardware* e *software*. Seu principal objetivo é permitir que usuários, principalmente estudantes, sem base de conhecimento em eletrônica e programação criem hardwares interativos com baixo custo (BARROS, 2010, p.25-44), (ARDUINO, 2017). Devido as suas características, os Arduinos podem ser facilmente utilizados em sistemas embarcados.

O hardware é disposto em uma pequena placa de circuitos, cujo tamanho varia conforme o modelo, que possui basicamente um microcontrolador programável *Atmel* e uma interface simples para conexão de sensores e atuadores. A gravação do microcontrolador não necessita de hardware adicional, sendo realizada através da porta USB (*Universal Serial Bus*). Além disso, a placa Arduino pode funcionar de maneira independente ou em conjunto com um computador. Na Figura 2.3.1 é apresentado o modelo Micro, que devido ao seu tamanho reduzido pode ser facilmente acoplado a placas de circuitos (BARROS, 2010, p.25-44), (ARDUINO, 2017).

Figura 2.3.1 – Arduino Micro.



Fonte: Arduino Products ²

Em geral, microcontroladores são SoC's (*System on a Chip*) que possuem um microprocessador de pequeno porte, capaz de executar um conjunto de instruções simples e rápidas. Possuem memória primária, periféricos e interface para se comunicar com outros periféricos (BARROS, 2010, p.25-44).

O desenvolvimento de programas para o Arduino é realizado através do uso de uma linguagem de programação própria baseada em *Wiring* ³ e que se assemelha a C/C++. O ambiente de desenvolvimento (IDE) do arduino é baseado no *Processing* ⁴, é de fácil utilização e possui interface limpa e simples. O *software* pode ser obtido gratuitamente no *website* oficial ⁵ e possui versões para *Windows*, *Mac*, *Linux* e inclusive uma versão *web* que roda diretamente no navegador. Os programas escritos na IDE Arduino são chamados de *Sketches*. (BARROS, 2010, p.25-44), (ARDUINO, 2017).

Todas as placas Arduino são completamente *open source* (*hardware* e *software*), o que permite aos usuários modificá-las conforme suas necessidades pessoais, criar novos projetos a

² Disponível em <<https://store.arduino.cc/usa/arduino-micro>>. Acesso em agosto de 2017.

³ Disponível em <<http://wiring.org.co/>>. Acesso em agosto de 2017

⁴ Disponível em <<https://processing.org/>>. Acesso em agosto de 2017

⁵ Disponível em <<https://www.arduino.cc/>>. Acesso em agosto de 2017

partir do Arduino e contribuir para sua evolução e aprimoramento. No entanto, a marca “Arduino” é registrada sob licença GPL (*General Public License*) e não é permitido utilizá-la sem as devidas permissões (BARROS, 2010, p.25-44), (ARDUINO, 2017).

Desta forma, placas arduino são capazes de ler entradas a partir de sensores e transformá-las em saídas a partir da operação de atuadores, se apresentando como uma ferramenta simples para o desenvolvimento rápido e barato de protótipos. Tanto o *hardware*, quanto o *software* são muito práticos e possuem boa usabilidade. Sua IDE é multiplataforma, o que facilita o desenvolvimento dos programas (ARDUINO, 2017).

3 Materiais e Métodos

Neste capítulo é apresentado o planejamento detalhado das atividades realizadas no decorrer deste trabalho acadêmico. Serão abordados os projetos de todos os componentes do sistema, o método utilizado para coletar as amostras empregadas nos testes e o processo de treinamento da rede neural MLP.

3.1 Projeto do Sistema

Antes de projetar o sistema, é necessário saber o que deve ser feito. As primeiras etapas do processo envolvem a obtenção de informações para criar a arquitetura e os componentes. Este processo, normalmente, se procede em duas fases: primeiro, é definida uma descrição informal do sistema, conhecida como requisitos; em seguida, estes requisitos são refinados para a especificação que contém informações suficientes para planejar a arquitetura do sistema.

Nesta seção serão definidos os requisitos do sistema, especificados suas funcionalidades e modo de operação. Serão elaboradas, separadamente, as arquitetura e organização de cada componente.

3.1.1 Requisitos

A definição dos requisitos é uma importante etapa de planejamento, pois a partir deles serão estabelecidas as funcionalidades e comportamentos que o sistema deve ter. Os requisitos podem ser divididos em funcionais e não funcionais. Os requisitos funcionais envolvem as funções básicas do sistema, descrevendo o que deve ser feito. Já os requisitos não funcionais contêm informações mais específicas sobre como executar tais funções, sendo de fundamental importância para o projeto do sistema (WOLF, 2008, p.12-17).

Requisitos funcionais:

- O sistema recebe uma cor (RGB) de entrada;
- A saída deve indicar o tipo de pista de rolamento;
- Utilizar uma técnica de classificação de dados para classificar a entrada/
- O sistema deve alimentado por bateria.

Requisitos Não Funcionais

- O sistema deve ser implementado na plataforma Arduino;

- O programa deve rodar no microcontrolador ATmega32u4 (Arduino Pro Micro) sem que seu desempenho seja afetado;
- Utilizar um sensor TCS230 para a obtenção dos dados de entrada;
- A saída deve ocorrer através de um barramento de 4 bits, possibilitando valores de 0 a 15;
- Utilizar Rede Neural MLP como técnica de classificação;
- Classificar pistas pavimentadas como asfalto, calçamento de pedras ou areia;
- A bateria utilizada deve ser de 6 a 35v DC;

A Tabela 3.1.1 apresenta uma visão geral sobre o projeto do sistema, expondo seu propósito principal, entradas e saídas, as funções que deve desempenhar e uma estimativa de orçamento. Também são definidas restrições de desempenho, consumo de energia, tamanho e peso.

Tabela 3.1.1 – Formulário de requisitos

Nome:	Sistema Embarcado para Identificação de Pistas de Rolamento
Propósito:	Um sensor para classificar, em tempo real, a pista de rolamento sobre a qual se encontra.
Entrada:	Dados sobre as intensidades das cores vermelho, verde e azul (RGB) de uma região estrita da superfície.
Saídas:	Um valor binário de 4 bits e um arquivo de log.
Funções:	Dados sobre a cor da superfície são obtidos através do sensor de cores TCS230 e a rede neural <i>multilayer perceptron</i> é utilizada para classificá-los. O valor da saída indica o resultado da classificação e as informações relevantes são armazenadas em um arquivo de log através do módulo micro SD.
Desempenho:	O sistema deve executar em tempo real, atualizando a saída a cada segundo.

	Arduino Pro Micro Compatível	R\$ 49,90
	Sensor TCS230	R\$ 39,90
	Módulo Micro SD	R\$ 10,90
	Cartão Micro SD 4gb	R\$ 19,90
	4 Led's Vermelhos	R\$ 0,68
	2 Chaves Tácteis	R\$ 0,50
Custo de desenvolvimento:	Regulador de Tensão L7805	R\$ 1,80
	Capacitores	R\$ 1,60
	Resistores	R\$ 0,80
	Jack P4 Fêmea	R\$ 0,80
	Placa Perfurada tipo Ilha	R\$ 9,92
	Total	R\$ 136,70
Consumo de energia:	Não especificado, embora não ultrapasse 5 w.	
Características físicas:	Tamanho: $6,5 \times 8,9 \times 1,5$ cm (L×C×A). Peso: 50 g.	

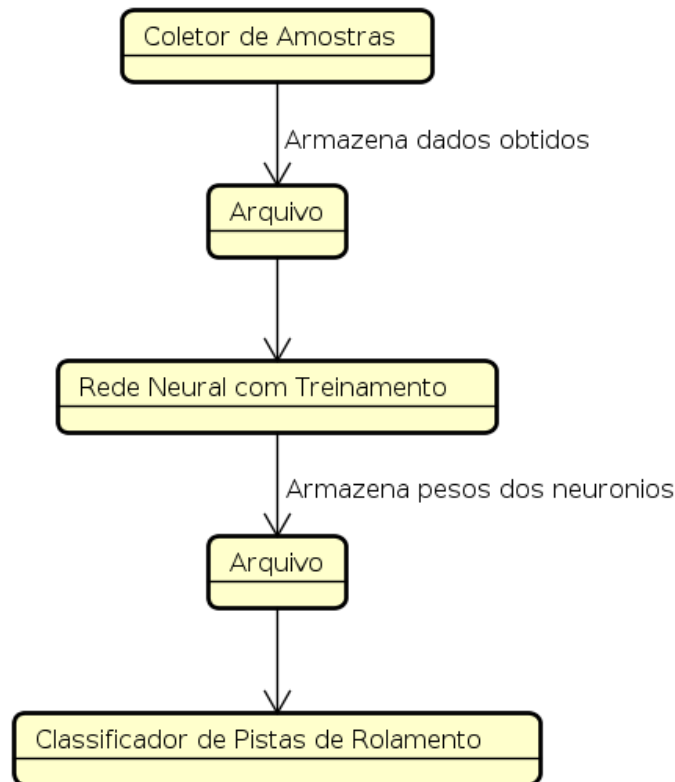
Fonte: Autoria própria

3.1.2 Especificação

O sistema embarcado é composto, por um *hardware* e dois *softwares* distintos. O *hardware* consiste, principalmente, em um Arduino Pro Micro, um sensor de cores TCS230 e um módulo Micro SD. Os *softwares* embarcados executam no Arduino, sendo que o primeiro coleta amostras de cores, enquanto o segundo implementa uma rede neural artificial *multilayer perceptron* treinada *offline*, encarregada de classificar os dados da entrada em tempo real. Além dos *softwares* embarcados, uma aplicação que executa em PC é utilizada para treinar a rede MLP.

A organização do sistema e a interação entre seus *softwares* são apresentadas na Figura 3.1.1. Como é possível observar no diagrama, o funcionamento consiste na execução sequencial e independente das três aplicações, para que, por fim, a terceira execute de forma definitiva atendendo aos requisitos especificados.

Figura 3.1.1 – Organização do sistema



powered by Astah

Fonte: Autoria própria

Primeiramente, amostras de cores são coletadas e armazenadas em um arquivo de texto. O segundo programa, que executa em um PC, recebe estas amostras como entrada e realiza o treinamento da rede neural MLP com o algoritmo *backpropagation*. Os pesos dos neurônios obtidos no treinamento são armazenados em um segundo arquivo de texto que é utilizado para configurar o *software* classificador, que executa no Arduino. Dado ao fato do ambiente embarcado possuir recursos limitados, o treinamento da rede é realizado no PC (fora do ambiente embarcado), o que reduz a complexidade da execução do *software* classificador.

3.1.3 Arquitetura e Organização

A cuidadosa elaboração da arquitetura de cada componente e a organização, são de fundamental importância para o devido funcionamento do sistema. É necessário que ao final do projeto todos os componentes atuem conforme o esperado e se integrem com facilidade.

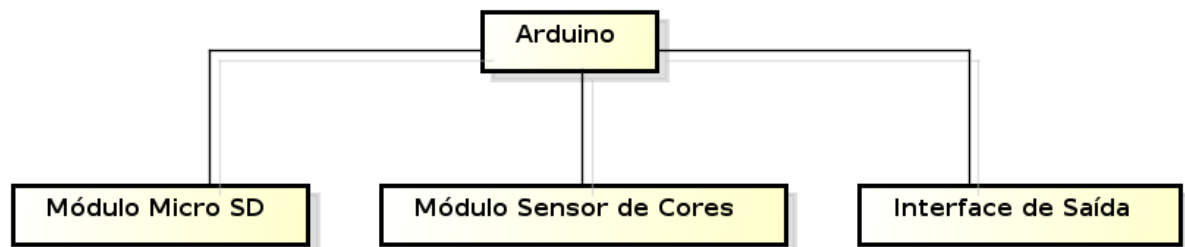
A seguir, serão abordadas as arquiteturas e organizações dos componentes que integram o sistema embarcado.

3.1.3.1 Hardware

O *Hardware* do sistema embarcado é composto por um Arduino Pro Micro, um sensor de cores TCS230 e um módulo microSD, onde o Arduino é o componente central e os demais são periféricos conectados a ele. Além desses componentes, também são utilizados um regulador de tensão, quatro leds, dois botões, resistores e capacitores que são necessários para o devido funcionamento do conjunto.

Na Figura 3.1.2 um diagrama de blocos apresenta uma visão geral sobre os componentes e sua organização.

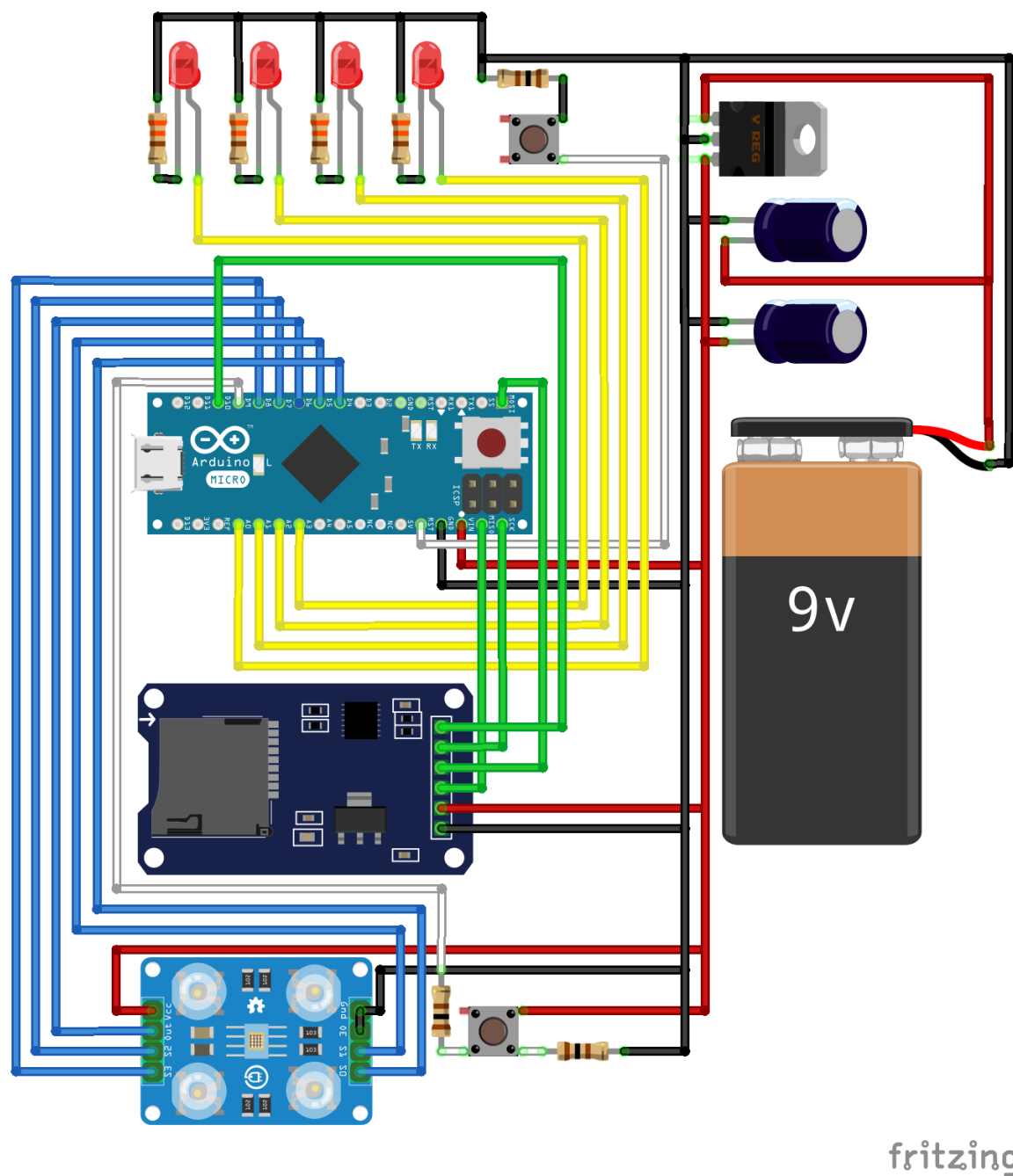
Figura 3.1.2 – Diagrama de blocos do *hardware*



Fonte: Autoria própria

A Figura 3.1.3 ilustra as ligações e a disposição das peças na placa de circuitos.

Figura 3.1.3 – Organização da placa de circuitos

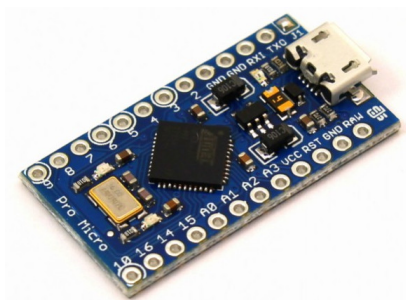


Fonte: Autoria própria

A seguir, serão apresentados os principais componentes utilizados no sistema e suas características mais relevantes:

- **Arduino Pro Micro Compatível:** É o principal componente do sistema. Possui o microcontrolador “ATMega32u4” que executa o programa carregado e faz uso dos componentes periféricos a ele ligados (Figura 3.1.4) (SPARKFUN, 2015).

Figura 3.1.4 – Arduino Pro Micro Compatível



Fonte: Autocore Robótica ¹

- **Sensor de Cores TCS230:** É um detector de cores completo. Possui um chip sensor RGB TCS230, que é um oscilador que produz uma saída proporcional à intensidade da cor recebida. Desta forma, ele é capaz de medir uma vasta gama de cores visíveis (Figura 3.1.5) (TAOS, 2009).

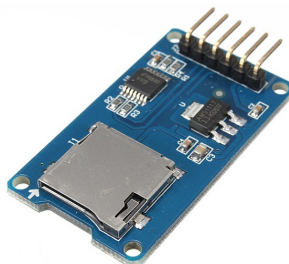
Figura 3.1.5 – Sensor de cores TCS230



Fonte: Autocore Robótica ²

- **Módulo Leitor de Cartão Micro SD:** É utilizado para ler e escrever arquivos em cartões micro SD e Micro SDHC. Permite guardar informações importantes (Figura 3.1.6).

Figura 3.1.6 – Módulo leitor de cartão Micro SD



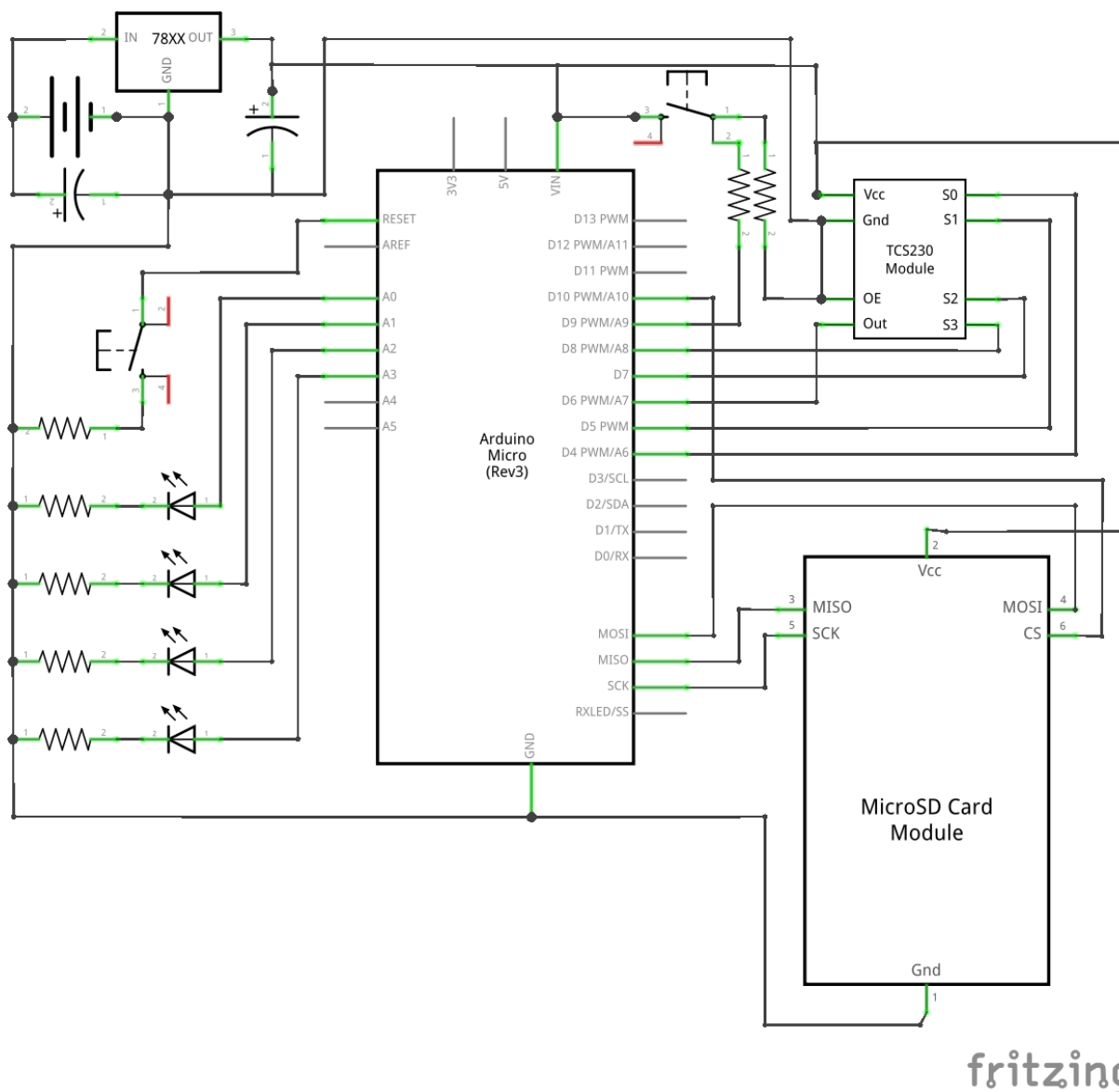
Fonte: Autocore Robótica ³

¹ Disponível em <<https://www.autocorerobotica.com.br/arduino-pro-micro-compativel-atmega32u4-16mhz>>. Acesso em setembro de 2017

² Disponível em <<https://www.autocorerobotica.com.br/modulo-sensor-de-reconhecimento-de-cor-tcs230>>. Acesso em setembro de 2017

Utilizando os itens descritos anteriormente e a organização definida na Figura 3.1.3, foi desenvolvido o esquema elétrico do *hardware* (Figura 3.1.7). O esquemático é uma importante ferramenta para a execução da placa de circuitos, pois representa graficamente todos os componentes que são utilizados e suas ligações através da simbologia apropriada.

Figura 3.1.7 – Esquema elétrico



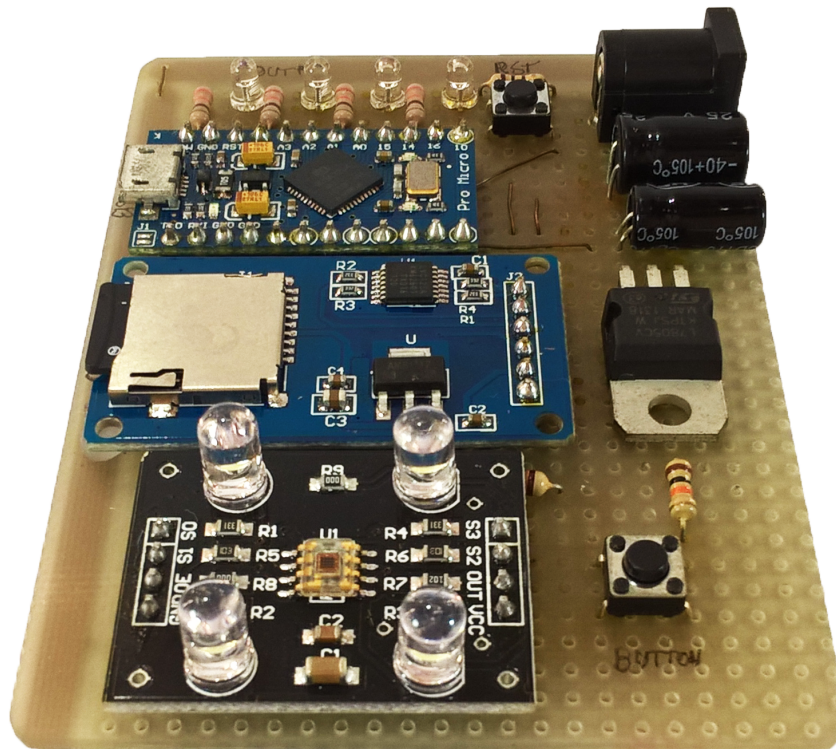
Fonte: Autoria própria

A partir do esquemático, foi montado o protótipo da placa de circuitos, onde todos os componentes foram fixados e conectados em uma placa perfurada do tipo ilha. A escolha deste tipo de placa se dá devido a praticidade de soldar os componentes na furação padronizada e permite criar as trilhas com fios e solda, dispensando a necessidade de utilizar equipamentos

³ Disponível em <<https://www.autocorerobotica.com.br/modulo-leitor-de-cartao-microsd>>. Acesso em setembro de 2017

específicos para impressão, corrosão e furação da placa. É possível observar o resultado final na Figura 3.1.8.

Figura 3.1.8 – Placa de circuitos



Fonte: Autoria própria

O circuito deve ser alimentado com tensão de 6v até 35v corrente contínua (DC). O regulador de tensão L7805 se encarrega de reduzir a tensão para 5v e alimentar devidamente todo o sistema (MICROELECTRONICS, 2004). Os capacitores são posicionados na entrada da alimentação para reduzir os ruídos provenientes da fonte, caso haja. Os quatro led's na parte superior ilustram a interface de saída, representando os 4 bits de um valor binário. O botão superior é utilizado para reiniciar o arduino e o inferior é reservado para uso futuro, não possuindo uma função atribuída até o presente momento. Resistores de valores diversos são utilizados para reduzir a diferença de potencial em pontos específicos.

3.1.3.2 Coletor de Amostras

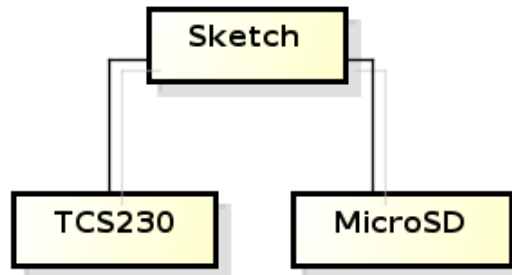
Este primeiro *software* tem o objetivo de coletar amostras de cores para serem utilizadas no treinamento da rede neural MLP implementada no segundo programa. Foi desenvolvido para a plataforma Arduino e utiliza o hardware descrito na seção anterior.

Seu funcionamento consiste em coletar os dados referentes à cor da superfície a partir do sensor TCS230 e armazená-los em um arquivo de texto presente no cartão de memória através

do módulo Micro SD.

O diagrama de blocos apresentado na Figura 3.1.9 representa a estrutura do programa.

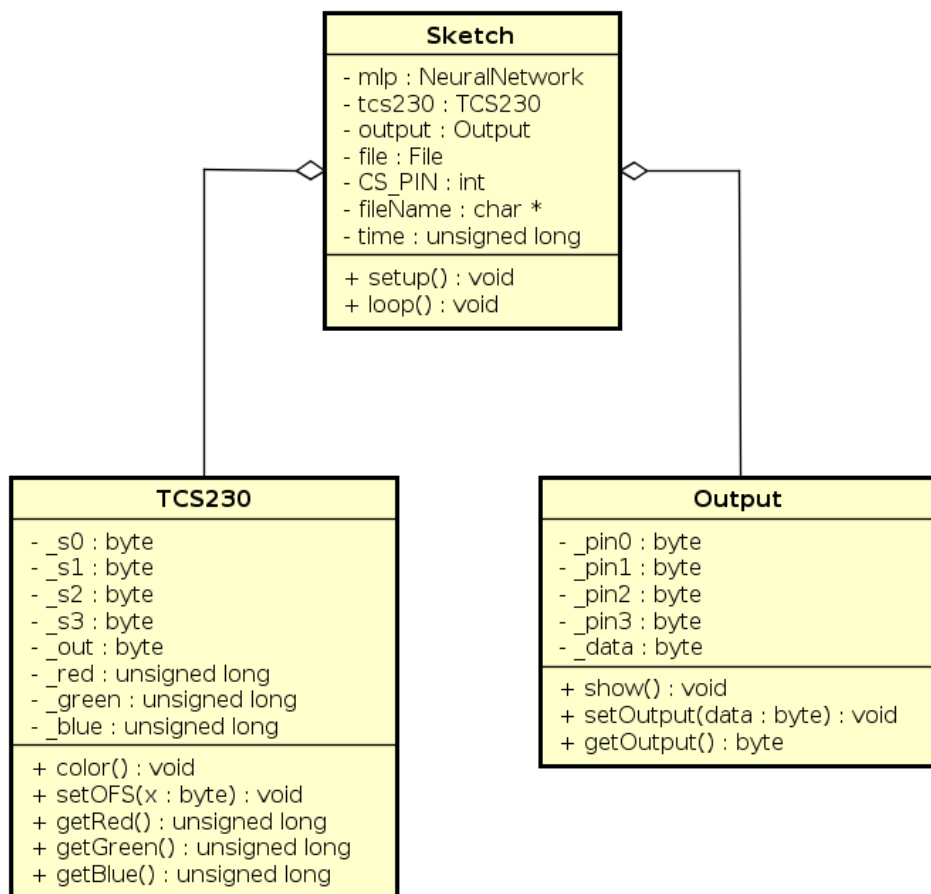
Figura 3.1.9 – Diagrama de blocos do *software* coletor de amostras



Fonte: Autoria própria

Detalhes da implementação podem ser observados no diagrama de classes exibido na Figura 3.1.10. A classe “TCS230” possui métodos para a utilização do sensor de cores, enquanto a classe “Output” dispõe de métodos para alterar facilmente a saída do sistema.

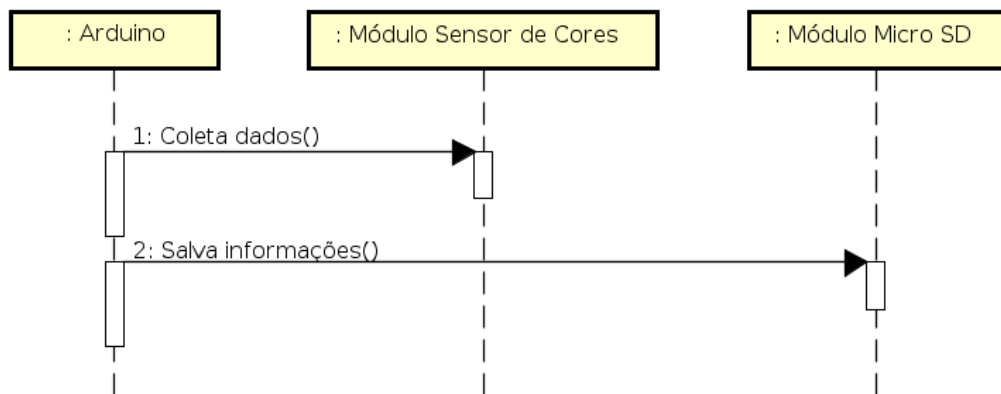
Figura 3.1.10 – Diagrama de classes do *software* coletor de amostras



Fonte: Autoria própria

A Figura 3.1.11 mostra o diagrama de sequência, que ilustra a comunicação entre os módulos do sistema.

Figura 3.1.11 – Diagrama de sequência do *software* coletor de amostras

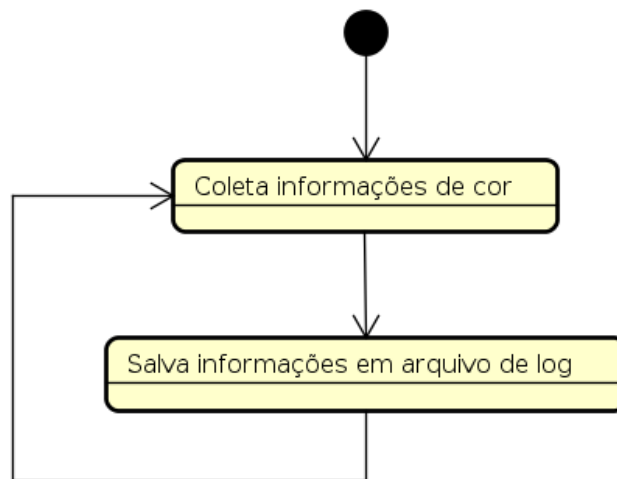


powered by Astah

Fonte: Autoria própria

A máquina de estados apresentada na Figura 3.1.12 representa o funcionamento do sistema.

Figura 3.1.12 – Máquina de estados do *software* coletor de amostras



powered by Astah

Fonte: Autoria própria

3.1.3.3 Implementação da Rede Neural *Multilayer Perceptron*

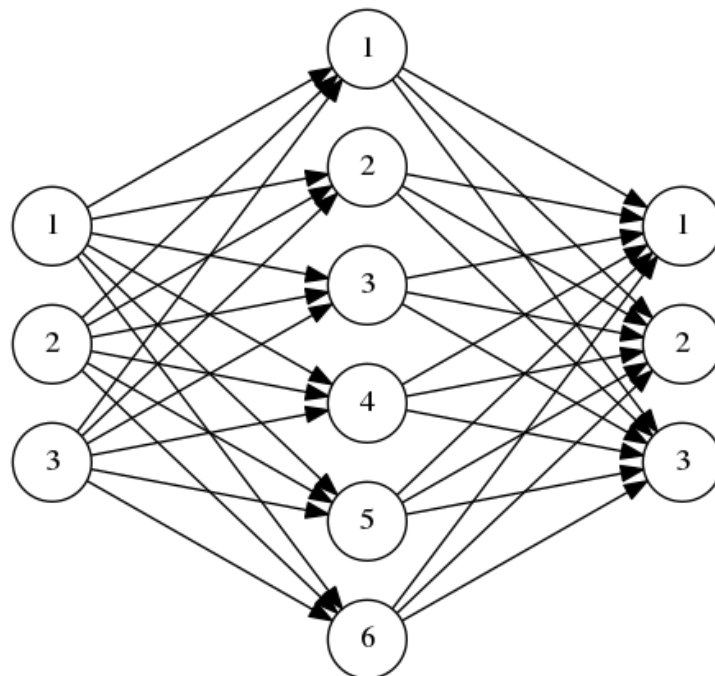
Este segundo *software* executa em PC e foi escrito em Java, devido a praticidade e familiaridade com a linguagem. Apesar de não se caracterizar um sistema embarcado, é de fundamental importância para seu funcionamento. Nesta etapa, foi desenvolvida uma rede neural

multilayer perceptron com 3 camadas, sendo uma de entrada (não processadora), uma oculta e uma de saída. A rede é completamente conectada, com alimentação adiante (*feed-foward*) e utiliza o algoritmo de treinamento supervisionado “*backpropagation*” (abordado na Seção 2.2.5).

A quantidade de neurônios utilizados na rede é um importante fator para seu sucesso, pois caso o número de neurônios da camada oculta seja pequeno, a rede não é capaz de criar limites de decisão complexos. Por outro lado, se o número for excessivo, a rede passa a decorar os dados e perde sua capacidade de generalização (ARTERO, 2008, p.140). Para facilitar esta decisão que deve ser tomada a partir da experimentação, a rede MLP foi implementada de forma genérica, permitindo alterar facilmente a quantidade de neurônios em cada camada através dos parâmetros do construtor da classe “*NeuralNetwork*”.

A quantidade de neurônios das camadas de entrada e saída devem ser, respectivamente, iguais as quantidades de entradas da rede e de classes que se deseja classificar. A quantidade de neurônios da camada oculta é definido de forma empírica, de maneira a se obter o melhor resultado possível. No caso desta rede proposta, são utilizados 3 neurônios na camada de entrada, um para cada componente RGB; 6 na camada oculta, cujo valor foi definido a partir dos testes discutidos no Capítulo 4; e 3 na camada de saída, um para cada classe (asfalto, calçamento com pedras e areia). A Figura 3.1.13 exemplifica a rede MLP.

Figura 3.1.13 – RNA *multilayer perceptron*



Fonte: Autoria própria

Na camada oculta foi utilizada a função tangente hiperbólica (Equação 3.1) e na de saída, a função degrau (Equação 3.3). O algoritmo de retro propagação utiliza as derivadas de ambas

as funções (Equações 3.2 e 3.4, respectivamente).

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (3.1)$$

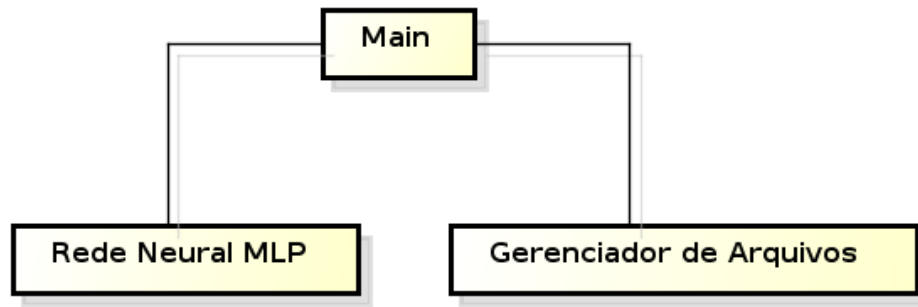
$$f'(x) = 1 - (f(x))^2 \quad (3.2)$$

$$f(x) = \begin{cases} 0 & \text{se } x \leq 0 \\ 1 & \text{se } x > 0 \end{cases} \quad (3.3)$$

$$f'(x) = 1 \quad (3.4)$$

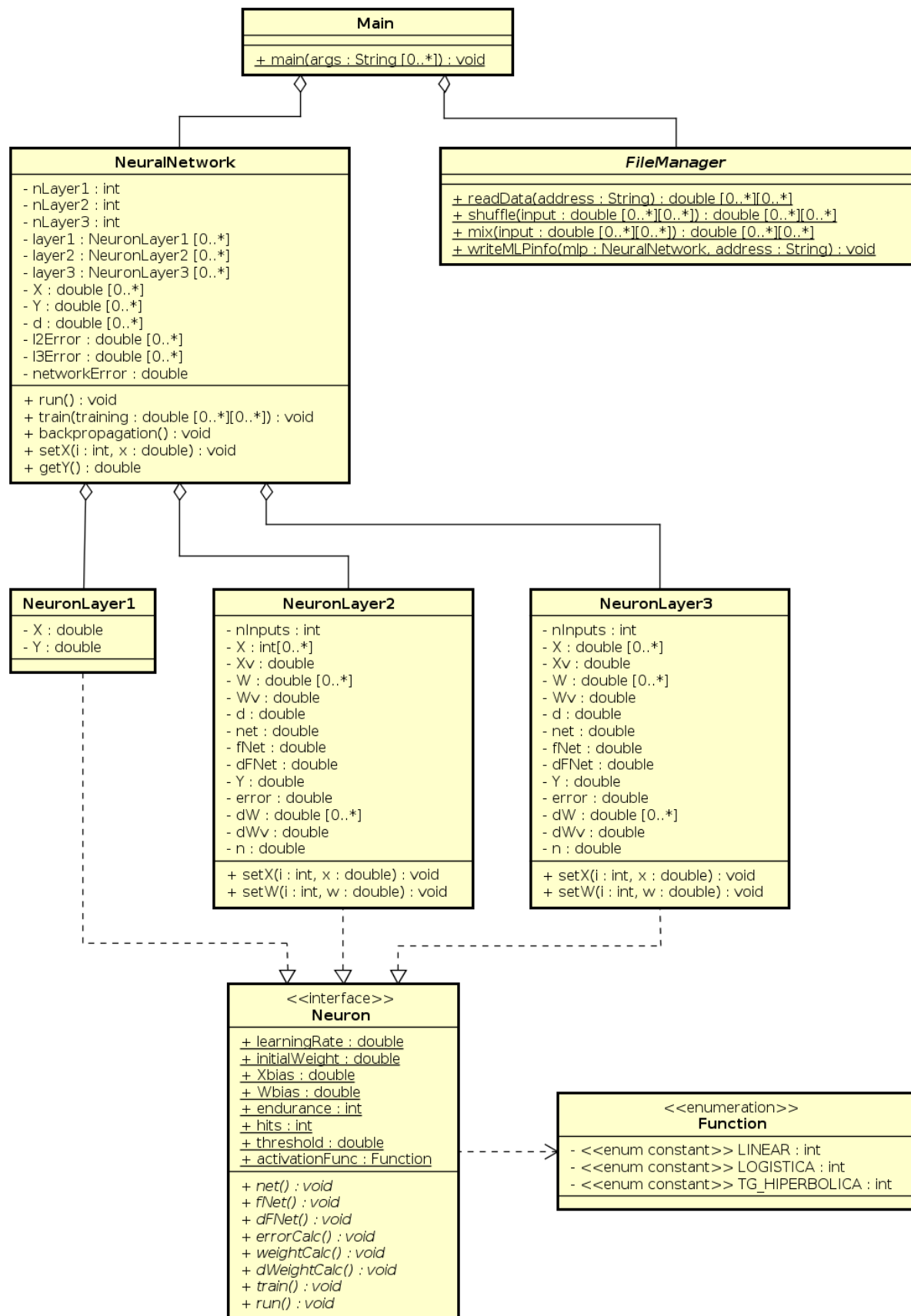
No diagrama apresentado na Figura 3.1.14 é possível observar a estrutura da aplicação, que consiste, basicamente, em uma rede neural *multilayer perceptron* com treinamento supervisionado e uma classe com métodos para manipular os arquivos de entrada e saída. O funcionamento do programa consiste em ler os dados do arquivo de texto que contem as amostras, treinar a rede neural até obter uma taxa de erro aceitável e por fim, gerar um segundo arquivo de texto contendo os dados dos pesos da rede.

Figura 3.1.14 – Diagrama de blocos do *software* MLP



Fonte: Autoria própria

A interface “*Neuron*”, como pode ser visto na Figura 3.1.15, é o esqueleto dos neurônios, as unidades mais simples da rede. A partir dela, são definidos três tipos distintos de neurônios, um para cada camada. A classe “*NeuralNetwork*” instancia a quantidade determinada de neurônios para cada camada e conecta-os. Também nesta classe é implementado o algoritmo de treinamento *backpropagation* que realiza o ajuste dos pesos da rede. A classe “*FileManager*” possui métodos para ler o arquivo produzido pelo programa coletor de amostras, organizar os dados para inseri-los na rede neural e gravar um arquivo com as informações que devem ser utilizadas pelo *software* classificador. Mais detalhes sobre a implementação podem ser observados no diagrama de classes exposto na Figura 3.1.15.

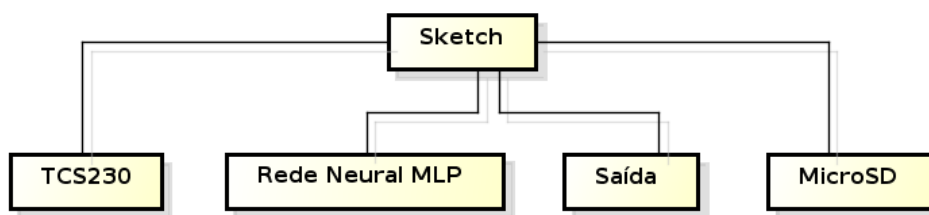
Figura 3.1.15 – Diagrama de classes do *software* MLP

3.1.3.4 Software Classificador de Pistas de Rolamento

Este último *software* é o que deve executar definitivamente no sistema embarcado. Assim como o primeiro, também foi desenvolvido na plataforma Arduino e utiliza o mesmo *hardware*, apresentado na Seção 3.1.3.1. Conforme definido nos requisitos (Seção 3.1.1), esta aplicação recebe informações RGB que descrevem uma cor, utiliza uma rede neural artificial para classificá-la e atualiza a saída conforme o resultado. A cada rodada, o arquivo de log no cartão de memória é atualizado.

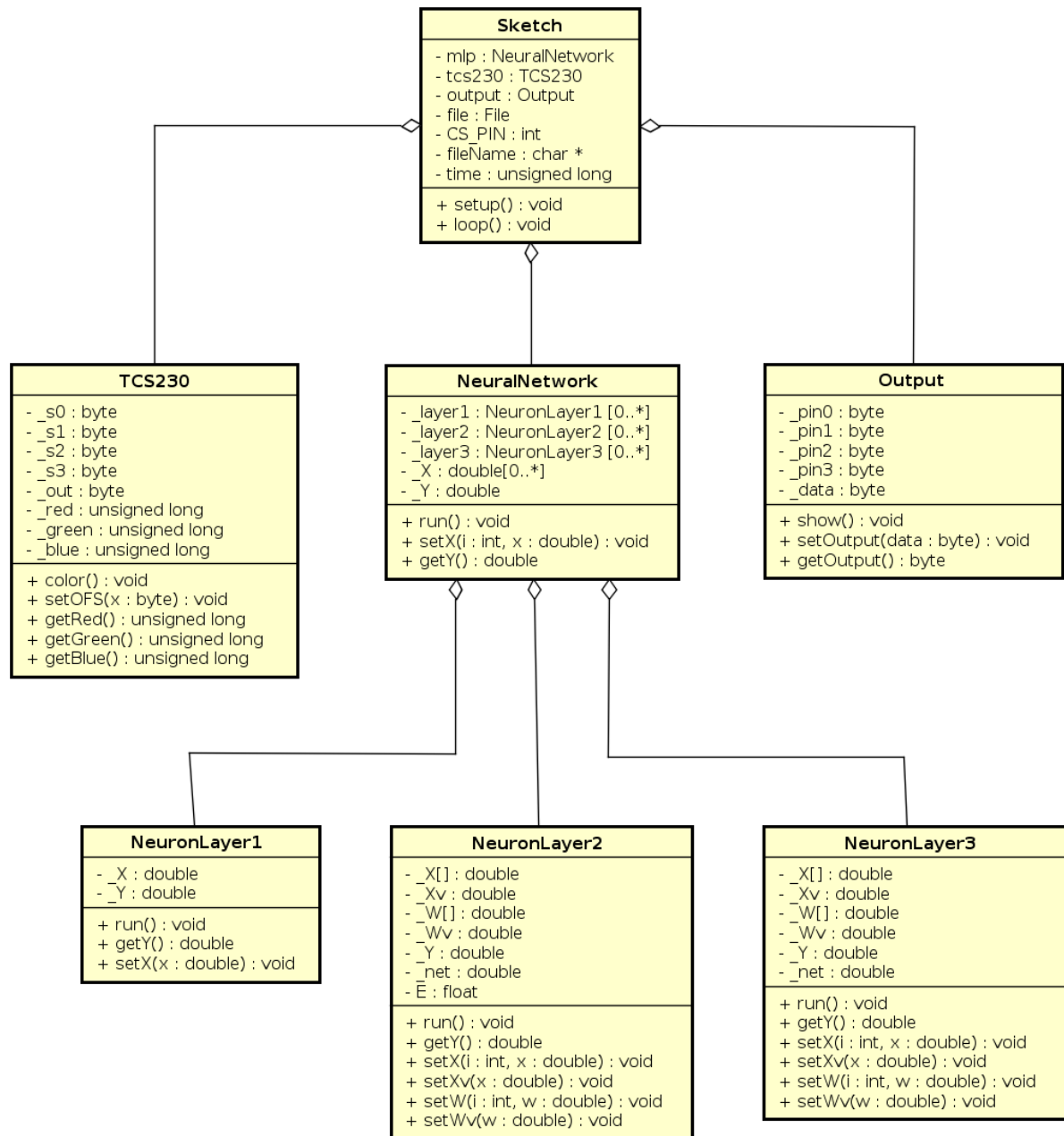
O diagrama de blocos apresentado na Figura 3.1.16 representa a estrutura do programa.

Figura 3.1.16 – Diagrama de blocos do *software* classificador



Fonte: Autoria própria

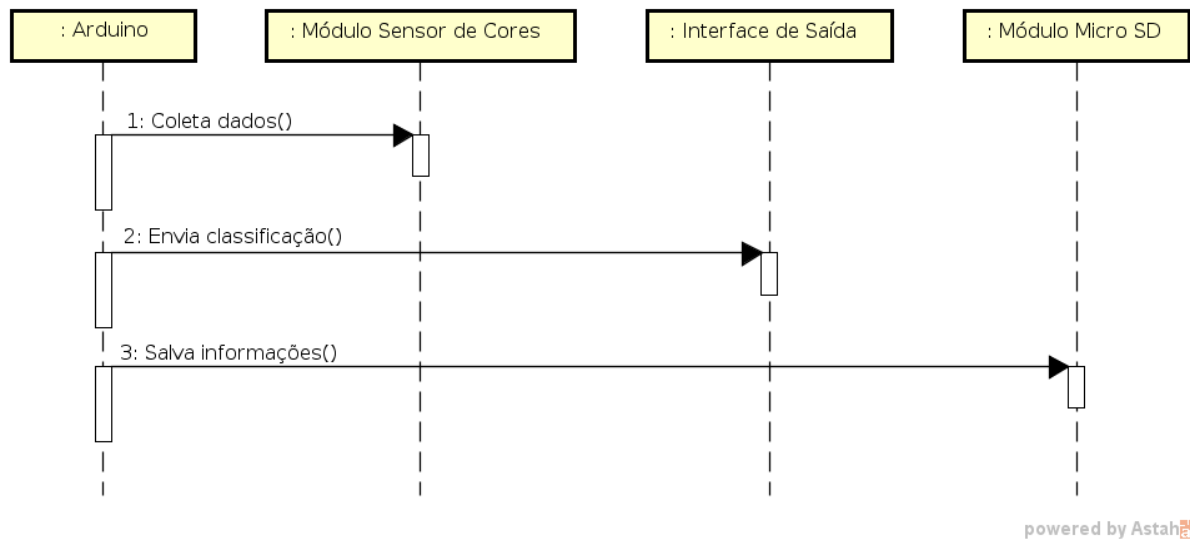
A rede neural implementada nesta aplicação é análoga à realizada na seção anterior (Figura 3.1.13). O projeto adotado também é muito semelhante, com exceção de algumas particularidades. São alocadas o menor número de variáveis possível, utilizando apenas o necessário, são realizados somente os métodos essenciais ao funcionamento da rede neural e não há processo de aprendizagem, pois os pesos obtidos no treinamento da aplicação Java são definidos em constantes. Deste modo, o esforço realizado pelo Arduino é minimizado na medida do possível, reduzindo a memória utilizada e otimizando o desempenho, uma vez que se trata uma aplicação em tempo real executando em um ambiente com recursos limitados. Mais particularidades da implementação podem ser observados no diagrama de classes (Figura 3.1.17).

Figura 3.1.17 – Diagrama de classes do *software* classificador

powered by Astah

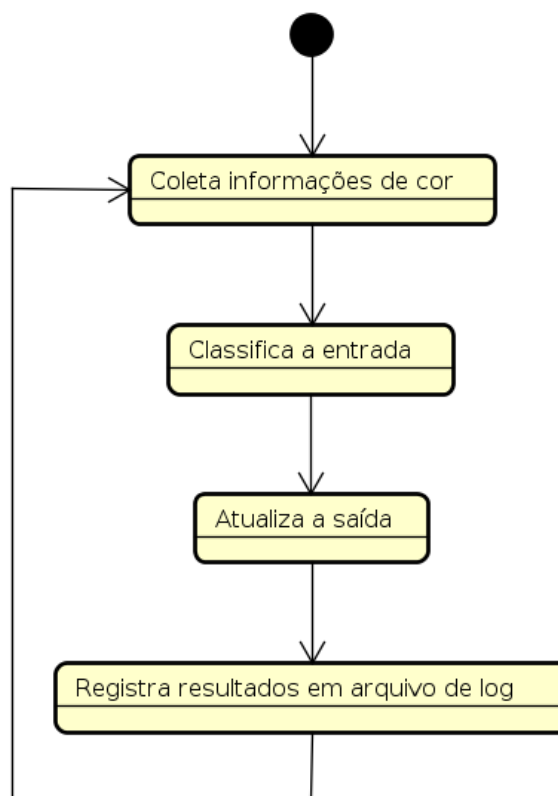
Fonte: Autoria própria

A Figura 3.1.18 mostra o diagrama de sequência, que ilustra a comunicação entre os módulos do sistema.

Figura 3.1.18 – Diagrama de sequência do *software* classificador

Fonte: Autoria própria

A máquina de estados apresentada na Figura 3.1.19 representa o funcionamento do sistema.

Figura 3.1.19 – Máquina de estados do *software* classificador

powered by Astah

Fonte: Autoria própria

3.2 Método de Amostragem

Para realizar a coleta das amostras o sistema foi fixado no interior de uma caixa de madeira, que por sua vez, foi fixada na parte inferior de uma motocicleta com visão livre ao solo. A furação da caixa utilizada permite que a luz atinja devidamente o sensor, que ficou posicionado a uma distância de aproximadamente 15 cm do solo.

Foi percorrido em perímetro urbano um trajeto com cerca de 10 quilômetros de extensão trafegando sobre os variados tipos de terrenos que se deseja classificar (asfalto, calçamento e areia). A velocidade média da motocicleta foi mantida entre 20 e 40 Km/h, conforme as condições da via, com eventuais paradas em cruzamentos e semáforos.

As amostras foram coletadas em diferentes horários do período vespertino para aferir a influência da luz ambiente nos resultados. O intervalo entre as medições também foi diversificado para verificar a periodicidade de dados incoerentes ou ruídos, caso se apresentem. As amostras foram coletadas com pausas de 1, 3 e 5 segundos.

O sensor TCS230 permite configurar a sua frequência de operação, possuindo 4 opções: 100%, 20%, 2% e 0% (desligado). Foi realizada amostragem com diferentes configurações para avaliar qual delas proporciona maior acurácia em conjunto com o Arduino. Quanto mais baixa for a frequência utilizada, mais ampla é a faixa de valores obtidos.

3.3 Processo de Aprendizagem

O processo de aprendizado da rede neural ocorre pela execução do algoritmo de retro propagação de erro *backpropagation*. O procedimento consiste em retro propagar o erro da camada de saída para as camadas internas ajustando os pesos da rede.

Para o treinamento, são utilizados os dados coletados no processo de amostragem descrito na seção anterior. Os dados contidos no arquivo de texto são lidos e armazenados em uma matriz. Como estão ordenados, os dados são previamente classificados, indicando a classe a que pertencem e, conseqüentemente, o valor que a rede neural deve retornar ao recebê-lo. Como os valores se encontram em uma faixa muito ampla, eles são normalizados para uma faixa entre 0 e 1 utilizando a Equação 3.5.

$$f(x) = x \times \frac{1}{\max - \min} + \min; \quad (3.5)$$

O conjunto de treinamento, seu tamanho mínimo e a ordem em que é inserido na rede neural são questões de grande importância. O conjunto deve incluir os valores de fronteira entre as classes, sua dimensão mínima deve ser igual ao quádruplo da quantidade de pesos na rede e deve ser apresentado alternando as classes, ou seja, apresenta-se um registro de cada classe por vez. (ARTERO, 2008, p.140)

Isto posto, após a normalização da entrada, a mesma é reorganizada intercalando as classes para então ser introduzida na rede MLP. Após a rede neural convergir, ou seja, atingir uma taxa de erro aceitável, os pesos de seus neurônios são armazenados em um arquivo de texto e devem ser utilizados pelo *software* classificador.

4 Resultados e Testes

As amostras foram coletadas em diferentes intervalos, a cada 1, 3 e 5 segundos. Contudo, não foram observadas diferenças relevantes entre seus valores e optou-se por utilizar as amostras colhidas com intervalo de 1 segundo. Os dados também foram obtidos com diferentes configurações de frequência do sensor de cores. Os melhores resultados foram obtidos utilizando a opção baixa, referente a 2% da frequência máxima, pois os valores se apresentam em uma faixa mais ampla e com menos ruídos. Quando normalizados para o intervalo entre 0 e 1, se mostraram mais acurados.

A segunda aplicação implementa a rede neural *multilayer perceptron* e realiza seu treinamento com o algoritmo *backpropagation*. A rede utiliza a função tangente hiperbólica como função de transição e foi treinada com 300 dados de cores RGB, sendo 100 pertencentes a cada classe. As amostras utilizadas foram coletadas no período que antecede o pôr do sol, quando a iluminação do solo se apresenta de maneira mais uniforme.

A melhor configuração da rede foi obtida com 6 neurônios na camada oculta, apresentando taxa de acerto de 69% após 2.640 iterações do algoritmo de treinamento. A tabela 4.0.1 mostra os resultados obtidos para diferentes tamanhos da camada oculta. A opção escolhida é a com maior a taxa de acerto e menor número de iterações.

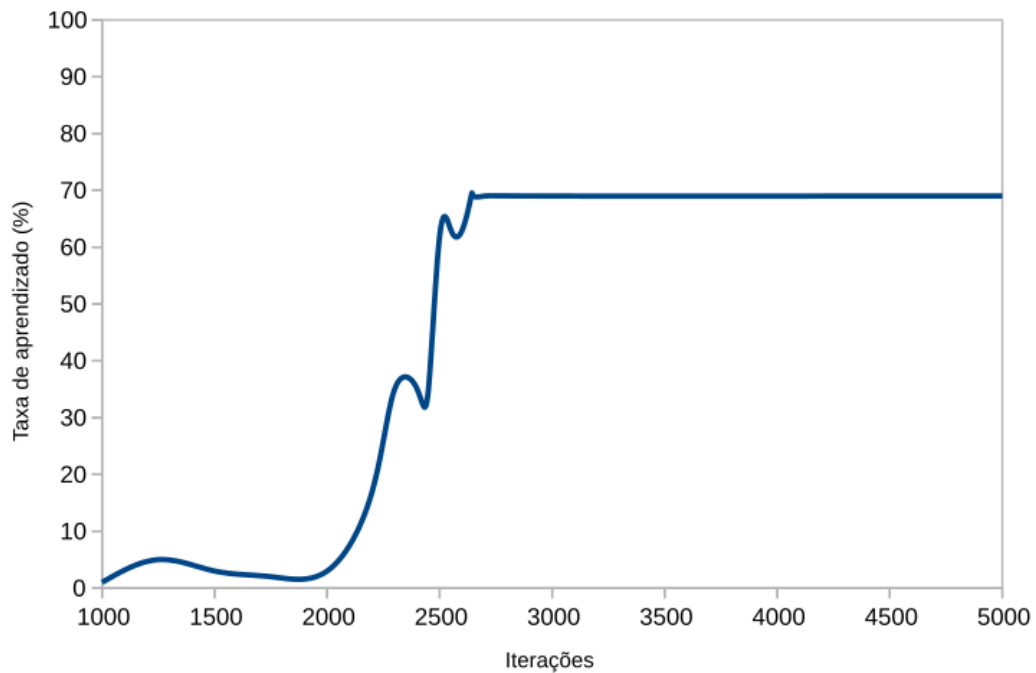
Tabela 4.0.1 – Quantidade de neurônios da camada oculta

Nº de neurônios	Iterações	Taxa de acerto
4	62.866	66 %
5	46.761	64 %
6	2.640	69%
7	não converge	-
8	não converge	-

Fonte: Autoria própria

A curva de aprendizagem da rede é mostrada no Gráfico 4.0.1. É notável, que no progresso das iterações a taxa de acerto também progride até atingir seu ponto máximo. Mesmo ao continuar iterando, a rede mantém a mesma taxa de acerto, não apresentando problema de *overtraining*.

Figura 4.0.1 – Gráfico da curva de aprendizagem



Fonte: Autoria própria

É importante ressaltar que esta taxa de 69% foi a melhor taxa atingida. Em uma segunda amostragem realizada em condições similares, a taxa obtida foi de 62%. Para amostras capturadas em outros horários, com diferentes condições de iluminação ambiente, os resultados variaram entre 29% e 47%. O programa ocupou 22.508 bytes dos 28.672 disponíveis na memória primária do Arduino e apresentou bom desempenho com tempo médio de execução em torno de 500 milissegundos para cada entrada.

5 Considerações Finais

Motivado pelo fato da superfície do terreno ser um dos principais fatores que afetam a dirigibilidade de um veículo automotor, este trabalho propôs o desenvolvimento de um sistema embarcado que seja capaz de classificar a pista de rolamento em que se trafega, diferenciando-a entre asfalto, calçamento com pedras ou areia. Desta forma, o sensor proposto busca auxiliar os módulos já existentes e contribuir para evolução dos veículos de passeio proporcionando mais segurança e conforto.

O *hardware* construído funcionou conforme o esperado. A portabilidade do protótipo é de suma importância e suas características físicas (peso e tamanho) se mostraram muito cômodas para a realização dos testes. O consumo, apesar de não dimensionado, é baixo, pois o sistema foi alimentado sem problemas com uma bateria de 9V que não precisou ser substituída no decorrer das horas de experimentos realizados. A placa de circuitos se mostrou estável e resistente, já que mesmo operando na presença de trepidação intensa e poeira, não necessitou de qualquer alteração no projeto inicial, nem reparos.

O *software* coletor de amostras tem funcionamento simples e funcionou de acordo com o previsto. Apresentou bom desempenho, operando tranquilamente em tempo real e obtendo as amostras em tempo bem inferior aos intervalos definidos, que foram preenchidos com *delays* ao fim de cada execução.

A rede neural utilizada para classificar as entradas foi treinada com um conjunto de 300 dados de cor RGB coletados em intervalos de 1s durante um percurso realizado por diferentes tipos de pistas de rolamento. A rede apresentou melhor desempenho ao utilizar 3 neurônios na camada de entrada, 6 na camada oculta e 3 na camada de saída. A função tangente hiperbólica foi utilizada na camada oculta. Foram obtidas taxas de acerto entre 62% a 69% para amostras coletadas na mesma faixa horária e taxas entre 29% e 47% para faixas diferentes.

O módulo TCS230 se mostrou muito sensível à alteração da luz ambiente, pois seu sensor fica completamente exposto à luz sem qualquer lente ou ajustes de abertura e/ou exposição. Este fato limita a operação do dispositivo para a mesma faixa horária ao qual foi treinado. Ao realizar o percurso cedo, a iluminação do solo é irregular, com certas regiões iluminadas diretamente pela luz solar e outras sob as sombras das edificações. Ao atingir as regiões ensolaradas, são produzidos valores muito baixos, fornecendo valores semelhantes para diferentes classes, o que confunde a rede e impede seu aprendizado. De maneira análoga, ao realizá-lo um pouco mais tarde, já durante o crepúsculo, a luminosidade é baixa, o que causa valores elevados na amostragem, tornando mais sutil as diferenças entre as classes, dificultando a classificação. O período que antecede o pôr do sol se mostrou ser o horário mais propício para o treinamento da rede, pois a luz solar não incide diretamente sobre o solo e ao mesmo tempo o ambiente ainda

possui boa iluminação natural.

Apesar de seus recursos bastante limitados, o Arduino Micro (com microcontrolador AT-Mega23u4) se mostrou perfeitamente capaz de executar a rede neural implementada em tempo real, com tempo médio de execução em torno de 500 milissegundos.

Desta forma, o sistema proposto se mostrou eficaz ao apresentar boa taxa de acerto e um aceitável tempo de execução. Apesar de alguns problemas, algo normal para um protótipo, o produto possui potencial para ser aplicado na indústria automotiva, podendo operar junto aos sistemas embarcados já existentes, aprimorando a dirigibilidade, conforto e segurança dos veículos automotores.

5.1 Trabalhos Futuros

Para trabalhos futuros, deve-se buscar reduzir o impacto da luz ambiente na medições aferidas, permitindo que o dispositivo opere em uma faixa horária mais ampla. Isso pode ser feito através do ajuste de exposição e/ou balanço de brancos nas entradas recebidas. Também é possível utilizar o sensor em conjunto com alguma lente, para ajustar o foco da luz que o atinge.

Um método deve ser desenvolvido para aumentar a confiabilidade do produto. Mesmo com a rede neural não atingindo taxas de acerto próximas a 100%, deve-se garantir que resultados duvidosos não sejam repassados à saída.

A placa de circuito pode ser reduzida através da substituição dos módulos e componentes PTH (*pin through hole*) por componentes SMD (*surface-mount device*). Assim, devem ser reduzidos o tamanho, o consumo energético e o custo de fabricação do sistema.

Pesquisas e testes devem ser realizados para indicar se as melhorias propostas serão eficazes e quais outras medidas podem ser tomadas para aprimorar o produto.

Referências

- AGUIAR, H. et al. *Inteligência Computacional*. 1. ed. [S.l.]: Thomson, 2007. Citado 3 vezes nas páginas 17, 19 e 21.
- ARDUINO. *What is Arduino?* 2017. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Acesso em: agosto de 2017. Citado 2 vezes nas páginas 24 e 25.
- ARTERO, A. *Inteligência Artificial: Teórica e Prática*. 1. ed. São Paulo, Brasil: Livraria da Física, 2008. Citado 9 vezes nas páginas 15, 17, 18, 19, 20, 21, 23, 37 e 43.
- BARROS, T. *Introdução ao Arduino*. 2010. Apresentação de slides. Citado 2 vezes nas páginas 24 e 25.
- BURTON, D. et al. *Evaluation of Anti-lock Braking Systems Effectiveness*. [S.l.], 2004. Citado na página 11.
- GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. 3. ed. São Paulo, Brasil: Pearson, 2009. Citado 3 vezes nas páginas 14, 15 e 16.
- LUDWIG JR, O.; MONTGOMERY, E. *Redes Neurais: Fundamentos e Aplicações com Programas em C*. 1. ed. [S.l.]: Editora Ciência Moderna, 2007. Citado 4 vezes nas páginas 17, 18, 19 e 20.
- MICROELECTRONICS, S. *L7800 Series: Positive Voltage Regulators*. 1. ed. USA, 2004. Citado na página 34.
- NORVIG, P.; RUSSELL, S. *Inteligência Artificial*. 3. ed. [S.l.]: Elsevier, 2013. Citado 3 vezes nas páginas 17, 19 e 21.
- SILVEIRA, F. L. da. Potência de tração de um veículo automotor que se movimenta com velocidade constante. *Revista Brasileira de Ensino de Física*, v. 33, n. 1, p. 1–7, 2011. Citado na página 11.
- SPARKFUN. *Pro Micro & Fio v3 Hookup Guide*. 1. ed. USA, 2015. Citado na página 31.
- TAOS. *TCS3200, TCS3210 Programmable Color Light-To-Frequency Converter Datasheet*. 1. ed. Texas, USA, 2009. Citado na página 32.
- WOLF, W. *Computer as Components: Principles of Embedded Computing System Design*. 2. ed. Massachussetts, USA: MorganKaufmann, 2008. Citado 2 vezes nas páginas 23 e 26.