UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEBORA FERNANDES DA SILVA

MARCIO LUIS BORGES DOS SANTOS

AVALIAÇÃO DE DESEMPENHO DE ASSISTENTE DE VELOCIDADE APLICADO EM AGV UTILIZANDO CONTROLE FUZZY

CURITIBA

2022

DEBORA FERNANDES DA SILVA

MARCIO LUIS BORGES DOS SANTOS

AVALIAÇÃO DE DESEMPENHO DE ASSISTENTE DE VELOCIDADE APLICADO EM AGV UTILIZANDO CONTROLE FUZZY

PERFORMANCE ASSESSMENT OF SPEED ASSISTANT APPLIED IN AGV USING FUZZY CONTROL

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Miguel Antonio Sovierzoski

CURITIBA

2022

DEBORA FERNANDES DA SILVA
MARCIO LUIS BORGES DOS SANTOS

AVALIAÇÃO DE DESEMPENHO DE ASSISTENTE DE VELOCIDADE APLICADO EM AGV UTILIZANDO CONTROLE FUZZY

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 15/junho/2022

_____

Miguel Antonio Sovierzoski
Doutor
Universidade Tecnológica Federal do Paraná

_____

Mariana Antônia Aguiar Furucho
Doutora
Universidade Tecnológica Federal do Paraná

_____

Ednilson Soares Maciel
Mestre
Universidade Tecnológica Federal do Paraná

CURITIBA

2022

RESUMO

O trabalho proposto busca avaliar o desempenho de uma assistente de velocidade desenvolvido em lógica fuzzy e aplicado a robôs seguidores de linha autônomos. O objetivo principal é evitar que haja uma colisão entre os robôs, o assistente de velocidade deve ser capaz de controlar a velocidade diminuindo ou aumentando a mesma de acordo com a distância para o robô a sua frente. O projeto foi pensado inicialmente para que tanto o robô da dianteira quanto o robô que fica atrás teriam o mesmo código e ambos com o assistente de velocidade para que caso eles trocassem de lugar o comportamento seria o mesmo. No entanto, durante o desenvolvimento do código fuzzy exigiu um maior processamento, sendo assim foi necessário a troca do Arduino Uno pelo Arduino Mega, como o custo do Arduino Mega é bem superior ao do Arduino Uno decidiu-se implementar o assistente de velocidade somente em um dos robôs. Desta forma o robô seguidor de linha que permanece na dianteira permaneceu com o Arduino Uno e o controle é feito sempre de forma manual por um aplicativo desenvolvido no MIT Inventor, no entanto o robô que fica atrás recebeu o Arduino Mega e foi ajustado para funcionar com todos os componentes do hardware e sua velocidade é controlado pelo assistente de velocidade ou de forma manual por outro aplicativo desenvolvido no MIT Inventor. Há ainda um terceiro aplicativo que foi utilizado para coleta de dados do robô com assistente de velocidade que armazena os dados da distância e velocidade em uma planilha do google na nuvem. O trabalho no geral foi bem-sucedido, no entanto, durante o desenvolvimento foram encontrados vários problemas que tiveram que ser contornados durante o projeto, desde adaptações do hardware que já havia sido projetado, mas que não havia sido testado ainda até a troca de Arduino pela falta de capacidade de processamento do mesmo. Concluímos então que o fuzzy atendeu as premissas como assistente de velocidade, mas que talvez usando uma outra inteligência o desempenho seja melhor e que há a necessidade de melhorar o hardware, inserindo novos sensores que facilitem o controle de velocidade ou melhorando a disposição dos sensores existentes. Uma outra necessidade é rever a questão da bateria que no início do projeto tentou-se com uma bateria de 9 V e depois foi trocado para 3 baterias recarregáveis 18650 3800 mAh.

Palavras-chave: Assistente; Robôs; Controle; Fuzzy.

ABSTRACT

The proposed work seeks to evaluate the performance of a speed assistant developed in fuzzy logic and applied to autonomous line follower robots. The main objective is to avoid a collision between the robots, the speed assistant must be able to control the speed decreasing or increasing it according to the distance to the robot in front of it. The project was initially thought so that both the robot in front and the robot behind would have the same code and both with the speed assistant so that if they changed places the behavior would be the same. However, during the development of the fuzzy code, it required more processing, so it was necessary to exchange the Arduino Uno for the Arduino Mega, as the cost of the Arduino Mega is much higher than the Arduino Uno decided to implement the speed assistant in only one of the robots. In this way, the line follower robot that remains in the front remained with the Arduino Uno and the control is always done manually by an application developed in MIT Inventor, however the robot that is behind received the Arduino Mega and was adjusted to work with all hardware components and their speed is controlled by the speed assistant or manually by another application developed in MIT Inventor. There is also a third application that was used to collect data from the robot with a speed assistant that stores the distance and speed data in a google spreadsheet in the cloud. The work in general was successful, however, during the development several problems were found that had to be worked around during the project, from adaptations of the hardware that had already been designed, it hasn't been tested until the exchange of Arduino for the lack of processing power. We then concluded that fuzzy met the premises as a speed assistant, but that perhaps using another intelligence the performance is better and that there is a need to improve the hardware, inserting new sensors that facilitate speed control or improving the arrangement of sensors existing. Another need is to review the issue of the battery, which at the beginning of the project was tried with a 9 V battery and then was changed to 3 rechargeable batteries 18650 3800 mAh.

Keywords: Assistant; Robot; Control; Fuzzy.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A robótica teve um avanço estarrecedor nas últimas décadas. Contudo, engana-se quem pensa que a busca pela idealização de autômatos é exclusiva da atualidade. No Egito antigo, 2000 A.C, foi encontrado um cachorro mecânico. Em outro período da história da humanidade, 1495 D.C, o italiano Leonardo da Vinci projetou um robô com traços humanos (THOMAZINE, 2007). Entretanto, a palavra robô foi utilizada pela primeira vez, em 1921, na peça de teatro do Kapel Kapec, dramaturgo tcheco, nomeado de “R.U.R - *Rossum Universal Robot*” (Klaus Raiser, 2009).

Willian Gray Walter, na década de 1940, desenvolveu o primeiro dispositivo mecânico nomeado de robô. Segundo Raiser, o funcionamento desse robô dava-se através de dois sensores e um motor: um sensor de luz que desempenhava a função de localizar uma fonte de luz e o outro era um sensor de contato que tinha a função de desviar de obstáculos.

Partindo para uma área mais específica da robótica, em 1984, o italiano ciberneticista Valentino Braitenberg descreveu em seu livro intitulado *Vehicles: Experiments in Synthetic Psychology* um modelo de veículos robóticos simples com sensores e motores, que simulavam características psicológicas dos humanos através de mecanismos simples. A utilização deste conceito simplifica comportamentos complexos, os quais se devem, em sua maioria, a respostas simples a estímulos do ambiente.

Atualmente temos uma subárea da robótica denominada robótica móvel. O termo robótica móvel é comumente associado ao termo navegação autônoma. Um robô móvel de navegação autônoma opera sem nenhuma interferência humana, utilizando sensores em sistema de malha fechada para adquirir informações sobre seu ambiente atual e sendo capaz de agir de acordo com as informações obtidas (SECCHI, 2012).

Seguindo no estudo e desenvolvimento de tecnologias associadas a concepção e construção de robôs, este trabalho tem o objetivo de explorar dispositivos e soluções tecnológicas na subárea de robótica móvel. Dessa forma, será implementado um assistente de velocidade em robôs autônomos que controlam o sistema de aceleração e frenagem veicular em situações de risco. O sistema será testado em robôs seguidores de linha, construídos com arduino e sensores

ultrassônicos para detecção de objetos no sentido de deslocamento e controlando uma distância segura. Será utilizado um caminho fechado para os robôs seguirem. Dois robôs ou mais podem estar no mesmo sentido do caminho, sendo que os robôs mais rápidos ao chegarem na distância segura do robô da frente, diminuem a velocidade e acompanham a velocidade do robô mais lento, mantendo-se a distância segura entre os robôs.

1.1 Delimitação do tema

Este trabalho visa desenvolver um sistema inteligente de controle utilizando os métodos da lógica Fuzzy, que controlará a velocidade de veículos, mantendo uma distância segura para obtenção de um dispositivo anticolisão. Através da automatização do processo de detecção de proximidade, pretende-se o sistema de frenagem veicular diminuindo o risco de colisões, ou aumentando a velocidade até que o veículo alcance a velocidade de cruzeiro, caso não haja obstáculos ou veículos em velocidades menores à frente.

1.2 Problemas e premissas

O número de acidentes de trânsito cresce juntamente com o aumento de veículos circulando nas rodovias e ruas do país. Grande parte desses acidentes de trânsito são causados por falha humana, segundo a Organização Mundial da Saúde (OMS).

Há uma grande gama de projetos em teste para veículos autônomos e uma preocupação constante desses veículos é sempre a segurança. O assistente de velocidade para veículos com controle difuso propõe uma alternativa para evitar colisões de veículos evitando acidentes que possam trazer prejuízos à vida e/ou financeiros. Além disso, assistentes inteligentes propõem melhorar o desempenho de tais veículos, principalmente de veículos maiores que precisam de um espaço maior para parar e propõem garantir que um veículo consiga identificar que precise diminuir a velocidade por causa de um obstáculo e/ou outro veículo em velocidade menor é um avanço na segurança e na qualidade de trabalho dos condutores.

O professor orientador cedeu a plataforma de hardware para o desenvolvimento do trabalho. Desta forma, outros trabalhos desenvolvidos que utilizem esta plataforma podem comparar resultados e evoluir nos estudos.

1.3 Justificativa

Dispositivos de eletrônica embarcada são realidade em veículos automotivos, de passeio, de carga e de transporte. Tais dispositivos eletrônicos substituem dispositivos mecânicos, aumentando o conforto, diminuindo peso, entre outras características. Exemplos são vários, desde vidro com acionamento elétrico, que permite acionamento individual e fechamento automático de todos os vidros ao acionar a trava elétrica das portas, a direção *by-wire* que permite a diminuição das peças mecânicas da coluna de direção, e que o sistema de direção seja comandado por um sistema eletrônico assistente de pista, o acelerador *by-wire* que trabalha de forma mais coordenada com a ignição eletrônica, diminuindo o número de peças mecânicas e aumentando a eletrônica embarcada, bem como o piloto automático de velocidade de cruzeiro, liberando o pé do condutor do pedal do acelerador, o assistente de estacionamento, no qual o assistente controla a direção e a aceleração para estacionamento em vaga com uso de marcha ré, entre muitos outros sistemas que as montadoras automotivas têm desenvolvido e aplicado em seus produtos.

E sistemas assistentes para auxílio do motorista passaram de condição de protótipos de ideia, de protótipos acadêmicos e integram os veículos automotores de fabricantes de vanguarda, como assistente de estacionamento, indicador de obstáculo em ponto cego de espelhos, assistente de direção e assistente de velocidade, entre outros.

Esses avanços tecnológicos são inevitáveis, bem como a inserção em produtos automotivos de linha, e abrem espaço para que sejam desenvolvidos estudos acadêmicos e projetos para avaliação de ideias, como a proposta deste projeto de TCC, que apesar de utilizar protótipos de veículos autônomos seguidores de linha, protótipos em escala reduzida, servem para avaliar a ideia e desenvolver soluções que visem à produção e capacitação acadêmica com geração de massa crítica e produção de trabalhos de engenharia.

1.4 Objetivos

1.4.1 Objetivo geral

Realizar estudo, desenvolvimento e avaliação de desempenho de um assistente de velocidade que controla o sistema de aceleração, procurando manter em velocidade de cruzeiro e frenagem veicular em situações de risco (distância mínima).

1.4.2 Objetivos específicos

Para atingir o objetivo principal proposto, necessita-se cumprir os seguintes objetivos específicos:

- Estudar a teoria e aplicações da lógica *Fuzzy* para implementação de um modelo de raciocínio lógico;
- Definir e avaliar os sensores ultrassônicos a serem utilizados para detecção;
- Implementar a programação dos robôs para obter o funcionamento conforme as necessidades do projeto;
- Elaborar a lógica *Fuzzy* para o assistente de velocidade;
- Realizar a comunicação via Bluetooth do Arduino com os seguidores de linha utilizando aplicativo desenvolvido pela equipe para controlar a velocidade de do mesmo, este aplicativo auxiliará na demonstração de funcionamento do veículo portador do assistente de velocidade e possibilitará realizar um monitoramento da velocidade;
- Analisar e demonstrar o correto funcionamento de todo trabalho realizado.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada uma base teórica sobre veículos autônomos, englobando suas principais características e princípios de funcionamento. Serão abordadas definições da lógica fuzzy e conceituação de sensoriamento de distância e velocidade por ultrassom. Assim como, caracterização do Arduino e o modo de funcionamento dos robôs seguidores de linhas.

2.1 Veículos autônomos

Os veículos autônomos são resultados de um aprimoramento do estudo, desenvolvimento e aplicação do que se diz respeito a crescente evolução da robótica. O processo de automação veicular, busca proporcionar segurança, conforto e auxílio na condução do veículo, um exemplo é a manutenção em uma faixa correta para propiciar uma distância segura para outros veículos, conforme as condições encontradas no caminho, contribuindo para uma melhor tomada de decisão do condutor em um momento de situação adversa (BROGGI, 1998).

Um veículo autônomo é definido, como qualquer veículo terrestre que possua a capacidade de transporte, seja de pessoas ou materiais, dos quais não exigem a interferência humana para que sejam realizadas etapas de locomoção. Seu funcionamento é permitido por meio de uma associação composta por sensores e sistemas de controle, que possibilita o veículo perceber as condições do ambiente, e a partir da coleta de dados relevantes, serem realizadas tomadas de decisão quanto a velocidade, direção e demais ações que visam um resultado melhor e mais seguro do que poderia ser proposto por um ser humano (PISSARDINI; WEI; JÚNIOR, 2013).

Veículos autônomos são caracterizados por sua capacidade de identificar o ambiente e delinear de forma inteligente ações a serem tomadas diante de diferentes situações. Tais decisões são determinadas por algoritmos baseados em modelos treinados com amostras obtidas no mundo real, em um processo que procura explorar a maior quantidade possível de exemplos. Dessa forma, “o conjunto de sensores, processadores, rede de comunicação e software permitem o aprendizado e a tomada de decisão na condução do veículo” (BRAYER, 2019).

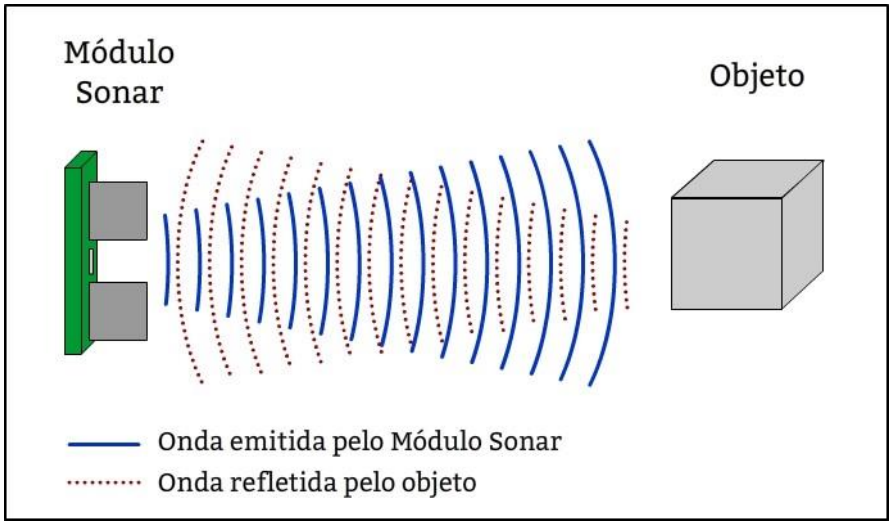
O processo de tomada de decisões mencionado anteriormente só é possível mediante a utilização de sistemas de controle, que são caracterizados como uma interconexão de componentes formando uma configuração de sistema que produzirá uma resposta desejada (DORF; BISHOP, 2001). Esses sistemas são modelados através da criação de algoritmos, que representam de forma prática os conceitos determinados na concepção do sistema.

Neste trabalho, o algoritmo utilizado para a interpretação do sinal dos sensores e a tomada de decisões diz respeito a um algoritmo de controle Fuzzy, em malha fechada.

2.2 Sensoriamento por ultrassom

Os transdutores ultrassônicos são compostos por cristais piezelétricos e conseguem fazer a detecção de objetos utilizando a emissão e recepção de ondas ultrassônicas. Os cristais que compõem os transdutores têm a capacidade de converter onda mecânica (som) em energia elétrica e o contrário também é realizado. O sensoriamento por ultrassom só é possível devido à capacidade dos transdutores de emitir pulsos ultrassônicos e após esses pulsos serem refletido por um obstáculo e receptado pelos cristais piezelétricos novamente. A Figura 4 mostra o funcionamento dos transdutores quando transmitindo ou refletindo um pulso ultrassônico.

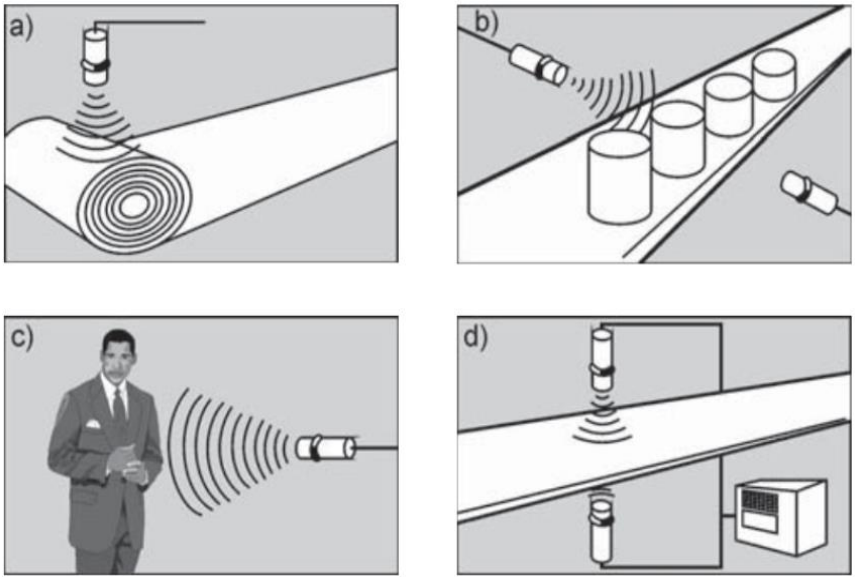
Figura 1 - Funcionamento de um transmissor/receptor ultrassônico



Fonte: TEIXEIRA (2020)

Para que ocorra o funcionamento coerente dos sensores ultrassônicos uma distância mínima deve ser respeitada entre sensor e objeto de estudo, pois, essa necessidade existe para que ocorra uma interpretação plausível dos pulsos enviados e recebidos pelos cristais piezoelétricos. Sensores de ultrassom possuem algumas vantagens em relação aos demais sensores, principalmente sistemas fotoelétricos tanto em detecção de objetos quanto a ser imperceptível no ambiente que está inserido. Algumas aplicações desse tipo de instrumento são apresentadas na Figura 5.

Figura 2 - Exemplo de utilização de sensores ultrassônicos



Fonte: Fundamentos e Aplicações (2011)

2.2.1 Sensoriamento de distância por ultrassom

O sensoriamento de distância pode ser feito através de radar, ultrassom, laser ou infravermelho. O funcionamento dos sensores de ultrassom ocorre com o auxílio de ondas sonoras com velocidade aproximada de 343 m/s, equivalente a 1200 km/h no ar. Essas ondas ou pulsos são emitidos pelos sensores e calcula-se a distância do objeto por intermédio do tempo que o pulso demora para chegar no mesmo e retornar para os transdutores de cristais piezelétricos (Luís Fernando Patsko, 2006, pg. 39).

A velocidade de uma onda mecânica, seja ela transversal ou longitudinal, é intrinsecamente ligada com as características inerciais do meio (para armazenar energia cinética) assim como das propriedades elásticas (para armazenar energia potencial) (HALLIDAY David, 2003). A Tabela 1 fornece alguns valores mais recorrentes na literatura para velocidade do som em diversos materiais distintos.

Tabela 1 - Velocidade do som em diferentes meios

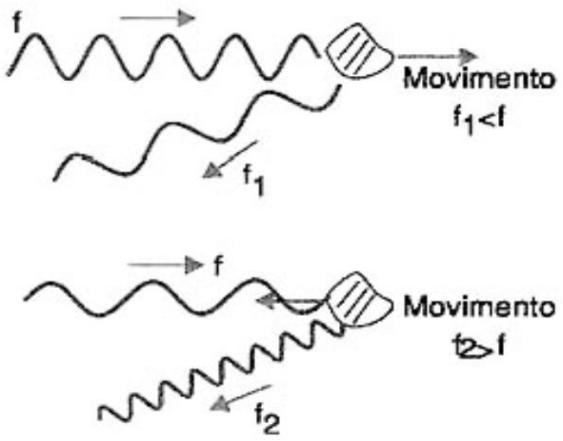
Meio	***Velocidade (m/s)***
Gases	
Ar (0ºC)	331
Ar (20ºC)	343
Hélio	965
Hidrogênio	1284
Líquidos	
Água (0ºC)	1402
Água (20ºC)	1482
Água do mar	1522
Sólidos	
Alumínio	6420
Aço	5941
Granito	6000

Fonte: HALLIDAY (2003)

2.2.2 Sensoriamento de velocidade por ultrassom

Determinar a velocidade de um objeto em movimento através de sensores ultrassônicos só é possível devido ao efeito Doppler. Quando uma fonte dispara pulsos na direção de um objeto que está movendo-se no sentido de propagação da onda, a onda refletida pelo objeto que retorna para fonte volta com um comprimento de onda menor e por consequência uma frequência também menor. A Figura 3 ilustra o comprimento de onda e frequência dependendo do direcionamento de propagação da onda e direção do objeto.

Figura 3 - Comprimento de onda de um sinal refletido



Fonte: BRAGA (2012)

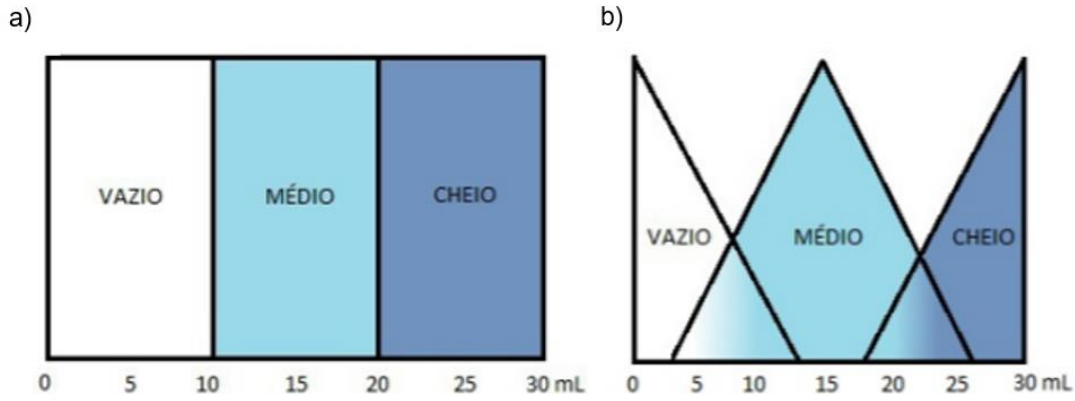
Entretanto, o oposto acontece quando a propagação do pulso a direção do objeto de movimentação são contrárias, pois, a onda refletida que retorna para fonte trás em suas propriedades um comprimento de onda menor e uma frequência maior (Newton C. Braga, 2012). E assim, através da alteração da frequência de onda e considerando a velocidade do som, consegue-se determinar a velocidade que o objeto observado movimenta-se utilizando a fonte como referência no espaço.

2.3 Lógica fuzzy

Os sistemas inteligentes são construídos a partir das percepções humanas buscando analogias ao sistema biológico dos seres humanos, estabelecendo padrões ou tomadas de decisões. No entanto, essas percepções geralmente fogem da lógica clássica em que um evento é verdadeiro ou falso, há várias nuances que devem ser consideradas (SIMÕES, 2007).

Em 1965 o professor da Universidade da Califórnia, Dr. Lofti Zadeh (1921-2017) apresentou a lógica fuzzy como um conceito derivado da lógica booleana. O desenvolvimento da lógica fuzzy surgiu como um estado que pode ir além do conceito binário de zero ou um, esse estado assume valores intermediários, não apenas absolutamente verdadeiro ou absolutamente falso, mas sim que possa existir como parcialmente verdadeiro ou parcialmente falso.

Figura 4 - Representação do copo de água na forma de conjuntos, na esquerda lógica clássica e na direita lógica nebulosa



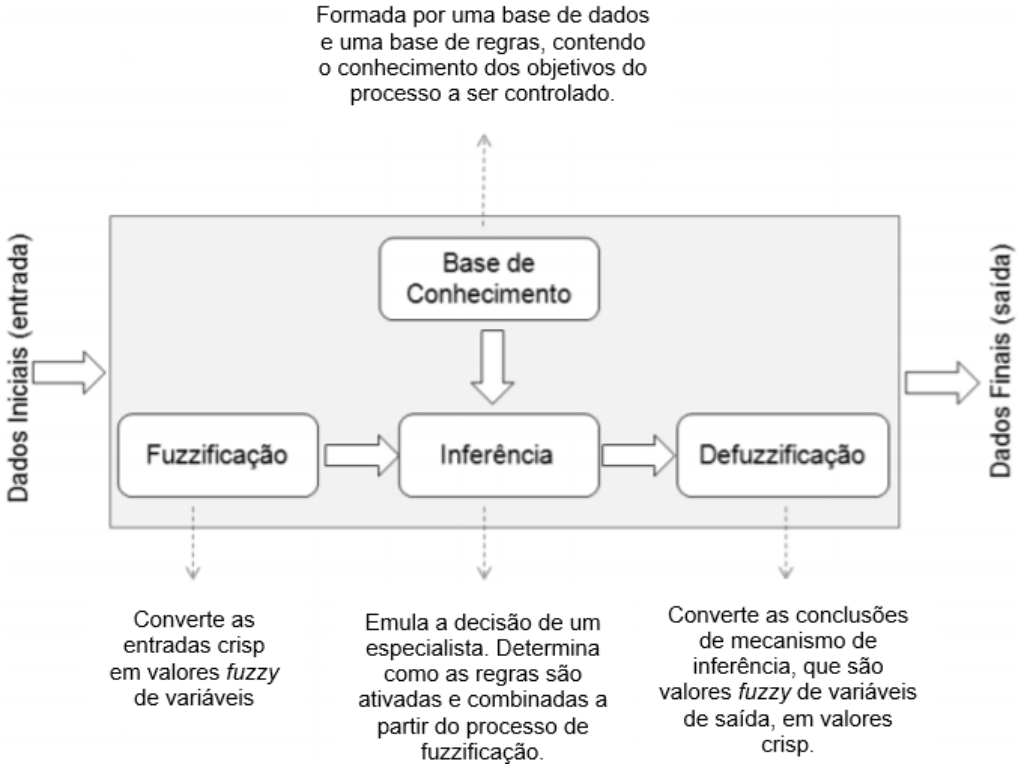
Fonte: Adaptado de MATTIELLO (2014)

A Figura 4 exibe o exemplo dos possíveis estados que um copo de água pode ser definido. O copo representado pelo gráfico “a” mostra um copo de água definido em três estados, já o gráfico “b” apresenta diversão possibilidades que um copo de água pode ser definido, através da lógica fuzzy, dependendo da quantidade de água que existe no recipiente. Além disso, a utilização da lógica fuzzy pode ser encontrada em todas as áreas, pois sua utilização é vista na matemática, engenharia, medicina, entre outras áreas.

A forma de raciocínio imprecisa dos seres humanos é chamada em inglês de fuzzy. A lógica fuzzy é uma técnica que incorpora o modo de pensar humano a um sistema de controle. Um controlador fuzzy típico pode ser implementado para agir conforme o raciocínio dedutivo, onde são utilizadas informações conhecidas para a tomada de decisão, como por exemplo, um processo industrial não linear onde o operador toma decisões baseado em suas experiências e inferências de relações entre variáveis do processo. A lógica fuzzy captura esse conhecimento adquirido pelo operador e permite implementar este conhecimento em um controlador fuzzy para que através do controlador seja desempenhada a mesma tarefa que o operador exerce (SIMÕES, 2007).

Para Simões (2007), a lógica fuzzy possui três tipos de operações estabelecidas em seus conceitos, sendo elas: fuzzificação, inferência e defuzzificação. A Figura 5 mostra a representação destas operações.

Figura 5 - Etapas do sistema fuzzy



Fonte: Adaptada de SIMÕES (2007)

2.3.1 Configuração básica de um controlador fuzzy

A fuzzificação trata-se de um mapeamento do domínio de números reais para o domínio fuzzy. Além disso, a fuzzificação também representa que há atribuição de valores linguísticos, descrições vagas ou qualitativas, definidas por funções de pertinência às variáveis de entrada. A fuzzificação é uma espécie de pré-processamento de categorias ou classes dos sinais de entrada, reduzindo grandemente o número de valores a serem processados. Uma menor quantidade de valores processados significa que há uma computação mais veloz (SIMÕES, 2007).

Em (SIMÕES, 2007), um controlador fuzzy é composto por quatro blocos funcionais, considerados básicos, sendo estes: interface de fuzzificação, base de conhecimento, lógica de tomada de decisões e interface de defuzzificação.

2.3.2 Fuzzificação e função de pertinência

A classificação de um dado, como distância, temperatura, umidade, velocidade, entre outras, de forma acentuada e menos repentina através do grau de

pertinência correlacionado a cada um dos conjuntos fuzzy foi proposto por Zadeh em 1965.

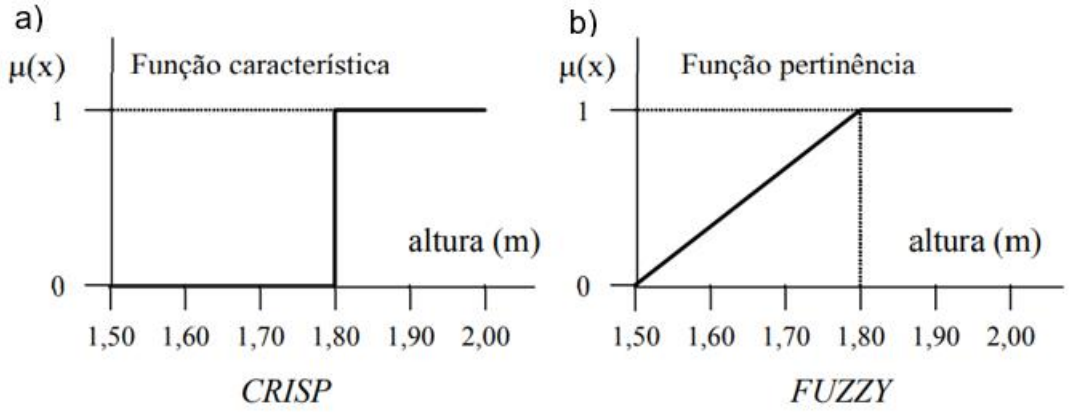
Este grau de pertinência relacionado a um determinado item possui uma função que estabelece um número real entre zero e um. As equações (1) e (2) consideram que a tomada de decisão estabelecida na teoria clássica pode ser expressa pela relação do conjunto A e um elemento α relacionado a esse conjunto, a segunda equação expressa à lógica fuzzy (ZADEH, 1965).

$$f(x) = \begin{cases} 1, se\ , e\ somente\ se, x \in A \\ 0, se, e\ somente\ se, x\ \notin A \end{cases} \quad (1)$$

$$f(x) = \begin{cases} 1,\ se\ , e\ somente\ se, x \in A \\ 0,\ se, e\ somente\ se, x\ \notin A \\ 0 \leq \mu(x) \leq 1,\ \ se\ x\ pertence\ parcialmente\ A \end{cases} \quad (2)$$

Para Zadeh os conjuntos fuzzy possuem um grau contínuo de pertinência, pois, dado um elemento α do espaço A, pertencente a um conjunto fuzzy X, é notável a relação existente com a teoria clássica dos conjuntos. Visto que, a existência de uma função de pertinência dada por μx(a), assume um valor real no intervalo de [0,1].

Figura 6 - Gráfico de representação do sistema CRISP e Fuzzy



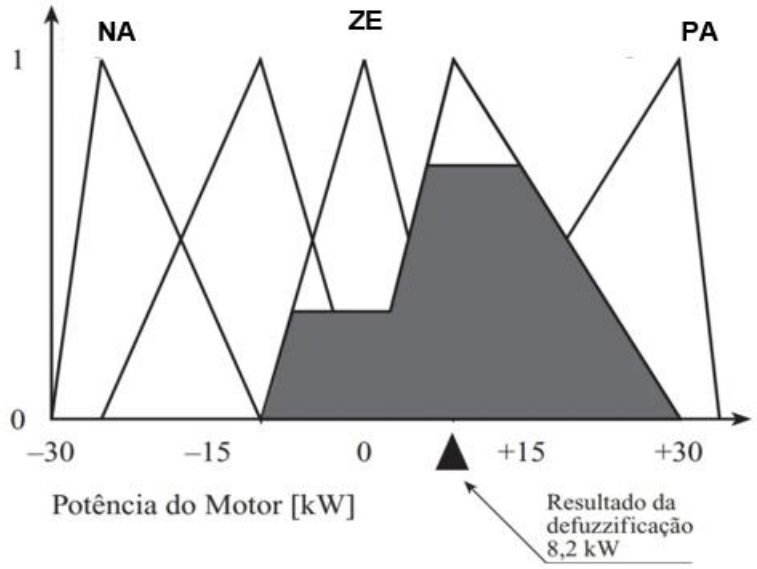
Fonte: Adaptada de IVANQUI (2005)

O exemplo da Figura 6 demonstra o conceito “pessoa alta” representado pelo sistema (crisp) e do lado direito o mesmo conceito, porém, representado pelo sistema fuzzy, no exemplo é considerado como uma pessoa alta se sua altura for maior que 1,80m (IVANQUI, 2005).

2.3.3 Defuzzificação

Para Simões, na defuzzificação, o valor da variável linguística de saída inferida pelas regras fuzzy será traduzido num valor discreto. O objetivo é obter-se um único valor numérico discreto que melhor represente os valores fuzzy inferidos da variável linguística de saída, ou seja, a distribuição de possibilidades. Assim, a defuzzificação é uma transformação inversa que traduz a saída do domínio fuzzy para o domínio discreto. Para selecionar o método apropriado de defuzzificação, pode-se utilizar um enfoque baseado no centroide ou nos valores máximos que ocorrem da função de pertinência resultante. Os principais métodos de defuzzificação utilizados em sistemas de controle são: centro do máximo (C-o-M), média do máximo (M-o-M) e centro da área (C-o-A).

Figura 7 - Método de defuzzificação pelo Centro-da-Área



Fonte: Adaptada de SIMÕES, 2007.

A Figura 7 apresenta como as funções de pertinência de uma variável linguística de saída nomeada como Potência-do-Motor, nesta o Centro-da-Área também conhecido como método do Centro-de-Gravidade, em razão de calcular o centroide da área que compõem o termo de saída fuzzy (μ_{out}), esta saída é composta por todas as uniões que contribuem para as regras. O centroide trata-se de um ponto que divide a saída μ_{out} em duas partes idênticas.

Considerando a existência de cinco funções de pertinência e a saída fuzzy composta por uma ação ou consequência, sendo resultantes das regras de inferência fuzzy.

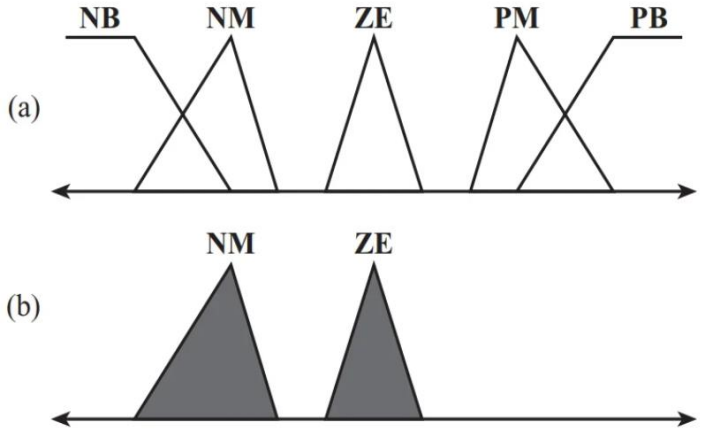
NB = 0,0; *NB* = 0,0; *ZE*= 0,2; *NB* = 0,8; *PB* = 0,0;

A utilização de uma saída diretamente das possibilidades acima pode-se considerar ambígua, pois existem duas ações diferentes no vetor, ou seja, graus de pertinência maior que zero. Segundo o Godoy (2007), a combinação dos conjuntos fuzzy é realizada de acordo com a teoria dos conjuntos. Para calcular o centroide desta área que relaciona o método de defuzzificação utiliza-se a equação (3).

$$u^* = \frac{\sum_{j=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{j=1}^{N} \mu_{OUT}(u_i)} \quad (3)$$

onde μ_{out} (μ_i) representa a área de uma função de pertinência (assim como ZE ou PM) quando modificado o resultado da inferência fuzzy (tais como 0,2 ou 0,8 respectivamente), e μ é a posição da função de pertinência individual - ZE ou PM respectivamente, dessa forma, através de tal equação é possível calcular o centroide composto, onde este contribui para as duas funções de pertinência indicadas. A Figura 8 apresenta as áreas ZE e PM, estas fazem com que o contorno da saída fuzzy seja composta para variável Potência-do-Motor.

Figura 8 - Ocorrência de erros devido à defuzzificação pelo Centro-da-Área



Fonte: Adaptada de SIMÕES, 2007.

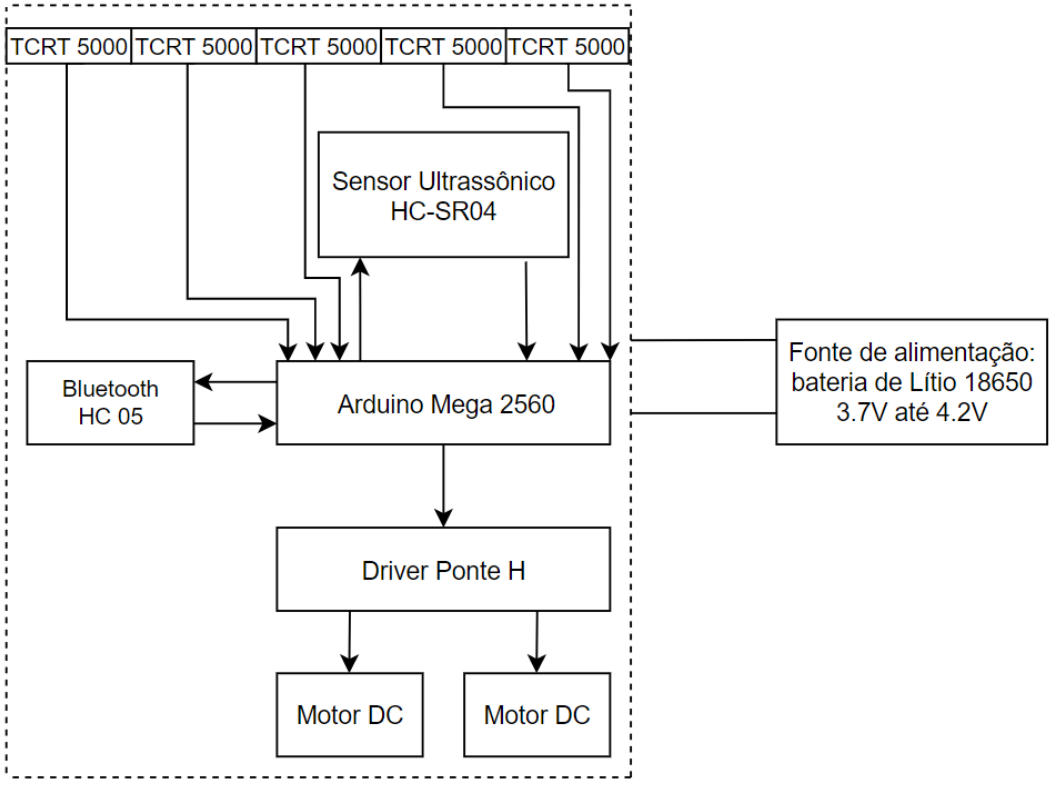
3 MATERIAIS E MÉTODOS

O presente item apresenta uma descrição dos componentes que serão utilizados para montagem e programação do robô seguidor de linha com **assistente de velocidade** elaborado no trabalho, além disso, serão abordados também os métodos de utilização destes componentes. Para a realização dos testes de funcionamento dos dispositivos foram utilizadas plataformas abertas relacionadas à robótica e de simulação que permite a implementação do código para Arduino.

3.1 Materiais

De forma simplificada será apresentado os principais componentes que compõem os robôs seguidores de linha utilizados no projeto. A Figura 9 apresenta um diagrama de blocos que contém parte dos componentes eletrônicos do seguidor de linha e a troca de informação que ocorre entre eles, sendo possível observar que algumas comunicações ocorrem de forma bidirecional e outras unidirecional.

Figura 9 - Digrama do Robô Seguidor de Linha



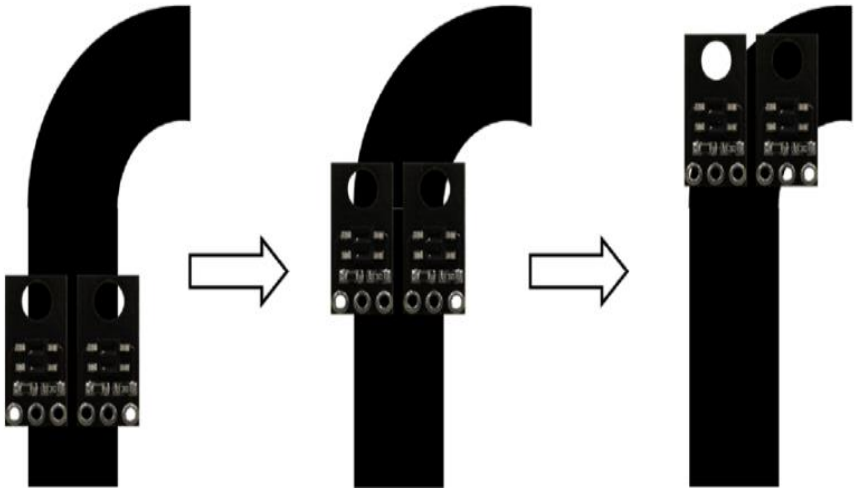
Fonte: Autoria própria.

Outro componente importante é o sensor infravermelho óptico reflexivo TCRT5000, este sensor é responsável pela detecção da faixa. Ainda, no presente item, será apresentado de forma mais claro alguns componentes eletrônicos do robô seguidor de linha e o site App Inventor.

3.1.1 Robô seguidor de linha ou robô segue-faixa

Há diferentes formas de montar um robô seguidor de linha, pode-se utilizar kits prontos como, por exemplo, o Kit Falcon com uma placa Julieta V1. 0 facilitando sua montagem e programação ou comprar os componentes necessários e montar sobre uma placa própria. A segunda opção tem a vantagem de ter um robô personalizado e sem as limitações dos kits prontos, no entanto, é um pouco mais trabalhoso. A Figura 10 ilustra a posição dos sensores em uma linha reta e depois em uma curva para a direita.

Figura 10 - Posicionamento dos sensores do robô seguidor de linha



Fonte: ROBÔ SEGUIDOR DE LINHA OU ROBÔ SEGUE-FAIXA, 2017

O robô seguidor de linha é capaz de seguir uma linha pré-determinada por causa dos sensores que devem estar próximos um do outro e posicionados sobre a linha. A programação é feita com o intuito de deixar os dois sensores sobre a linha. Caso os dois sensores estejam sobre a linha o robô deve andar para frente e caso um deles esteja fora da linha o robô deve fazer a curva.

Partindo desse raciocínio terem-se quatro possibilidades: os dois sensores sobre a linha, o sensor da direita sobre a linha, o sensor da esquerda sobre a linha ou

nenhum sensor sobre a linha. Caso os dois sensores estejam sobre a linha o robô irá andar para frente, para isso os motores devem estar com a mesma velocidade. Para que o robô permaneça no percurso quando o sensor da esquerda, por exemplo, estiver fora da linha é necessário mudar a velocidade de cada motor para que o robô faça a curva. Como neste caso o robô deve fazer a curva para a direita, o motor da direita deve ter velocidade menor do que o motor da esquerda.

O conceito utilizado para a curva é parecido com o conceito diferencial presente nos automóveis.

> "[...]Uma das principais atuações do diferencial é no momento da curva, onde uma roda precisa girar mais do que a outra. O diferencial mantém o torque igual entre elas.

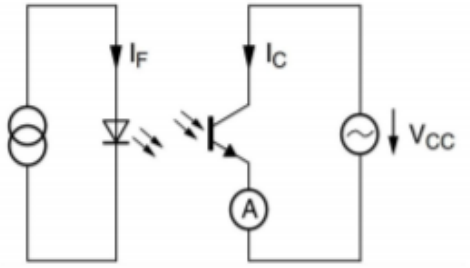
Em um carro no momento da curva caso a velocidade das duas rodas fossem a mesma o raio da curva deveria ser muito grande para que o carro fosse capaz de completar a curva. No entanto, em automóveis nas curvas é considerado também o grau que cada roda fica em relação ao solo. No robô seguidor de linha as rodas são fixas é necessário que para fazer à curva as rodas tenham velocidades diferentes.

Para que o robô ande em linha reta, seja para frente ou para trás é necessário girar as rodas em sentidos opostos, caso elas girem para o mesmo lado o robô irá girar em torno do seu próprio eixo.

3.1.2 Sensor Óptico TCRT 5000

É um sensor reflexivo com transmissor infravermelho e fototransistor integrados. O transmissor é um LED infravermelho que pode transmitir sinais nesta banda de frequência. O fototransistor é um receptor que lê o sinal refletido. Ou seja, o LED emite um feixe infravermelho, que pode ou não ser refletido pelo objeto. Se o feixe for refletido, o fototransistor reconhece o sinal refletido e gera um pulso em sua saída (ELETROGATE, 2017). A Figura 11 mostra o esquema de funcionamento do fototransistor e seu modo de condução.

Figura 11 - Esquema de funcionamento do Fototransístor



Fonte: VISHAY (2009)

A empregabilidade do TCRT5000 neste projeto é devido ao seu baixo custo e por apresentar as características necessárias para realização da função que deverá desempenhar no RSL, que se trata da identificação da linha, diferenciando o preto do branco apenas definindo uma limitação no valor da leitura para processar essa separação. O componente TCRT5000 é apresentado na Figura 12.

Figura 12 - Sensor infravermelho óptico reflexivo TCRT5000

Fonte: ELETRONICS PRO (2021)

3.1.3 Modulo Bluetooth HC05

A tecnologia Bluetooth é um protocolo de comunicação de rádio para uso pessoal, ou seja, uma especificação de rede sem fio classificada como PAN (*Personal Area Network*). Ele foi originalmente desenvolvido pela Ericsson em 1994 como uma alternativa sem fio ao protocolo RS-232. A faixa de frequência usada pelo Bluetooth é de 2,4 a 2,485 GHz. Os módulos HC05 e HC06 são os principais módulos Bluetooth que fazem interface com o Arduino. Para o desenvolvimento do projeto será usado o HC05, porque pode ser utilizado em modo mestre (faz e aceita pareamento com outros dispositivos) ou modo escravo (apenas aceita pareamento). A Figura 13 mostra a foto do módulo Bluetooth HC05 frente e verso.

Figura 13 - Módulo Bluetooth HC05 frente e verso

Fonte: THOMSEN (2015)

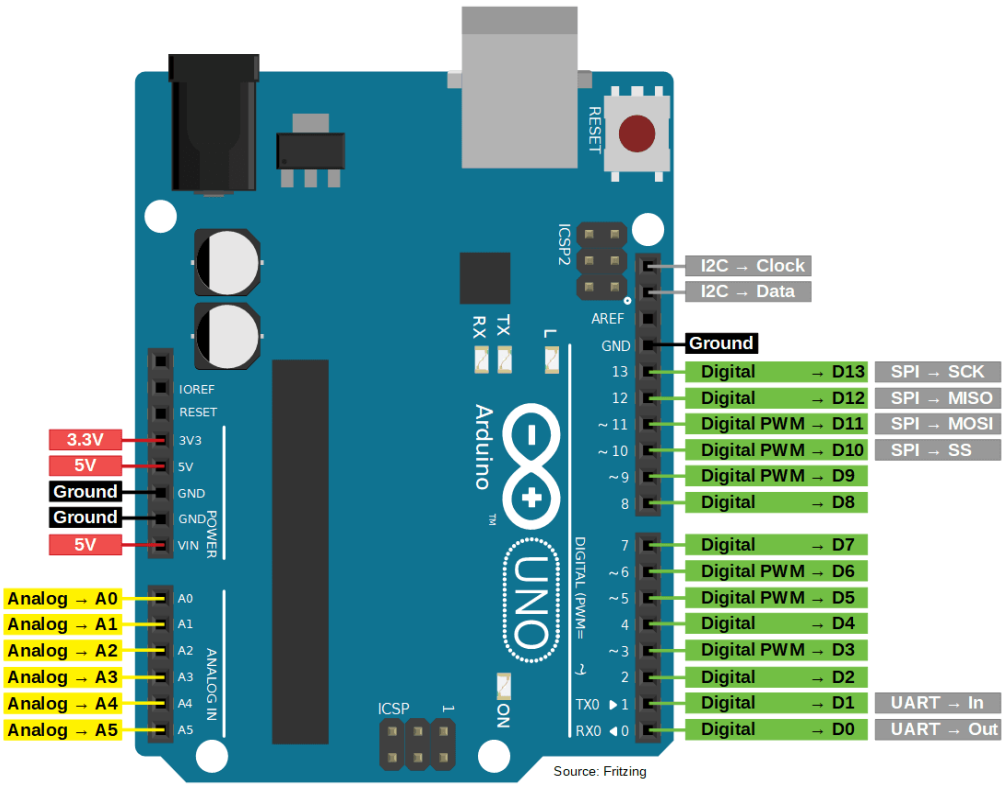
3.1.4 Plataforma de prototipagem Arduino

O Arduino foi criado no *Ivrea Interaction Design Institute* para ser uma ferramenta de prototipação fácil com público alvo sendo estudantes sem contato com eletrônica ou programação.

Segundo o site Arduino.cc "**Arduino Uno** é uma placa microcontrolada baseada no ATmega328P. Possui 14 pinos de entrada / saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um ressonador de cerâmica de 16 MHz (CSTCE16M0V53-R0), uma conexão USB, um conector de alimentação, um conector ICSP e um botão de reinicialização." A placa pode ser conectada a um computador via conexão USB ou ligada com um adaptador com um adaptador AC-DC ou ainda com uma bateria.

O módulo Arduino nas versões básicas utilizam o microcontrolador ATmega328P da ATmel. Um controlador de alto desempenho e com baixo consumo de energia e seu chip não possui soldagem facilitando a sua substituição, possui uma EPROM de 1 Kb (memória não volátil, que não é apagada com a falta de energia) e o seu plugue em formato cilíndrico funciona com uma bateria de 9V. A Figura 14 ilustra um desenho indicando toda a pinagem do Arduino UNO R3.

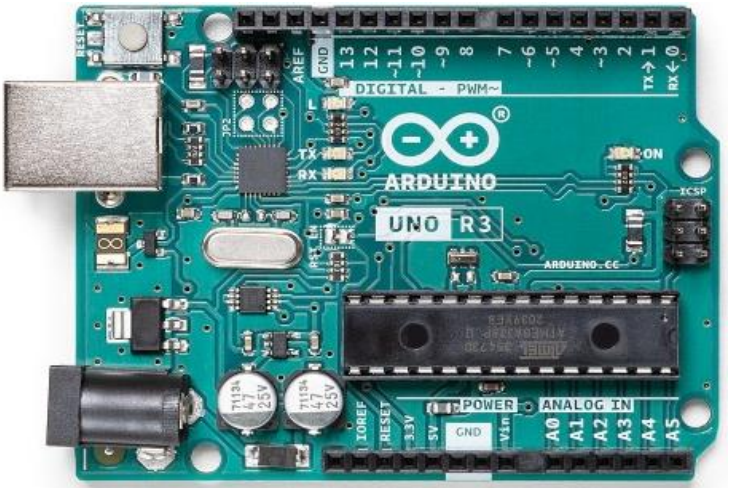
Figura 14 - Pinagem do Arduino UNO R3



Fonte: DIYI0T (2020)

Figura 15 apresenta uma imagem real do Arduino UNO R3, sua construção e toda a pinagem demarca, diretamente na placa.

Figura 15 - Foto do Arduino UNO R3

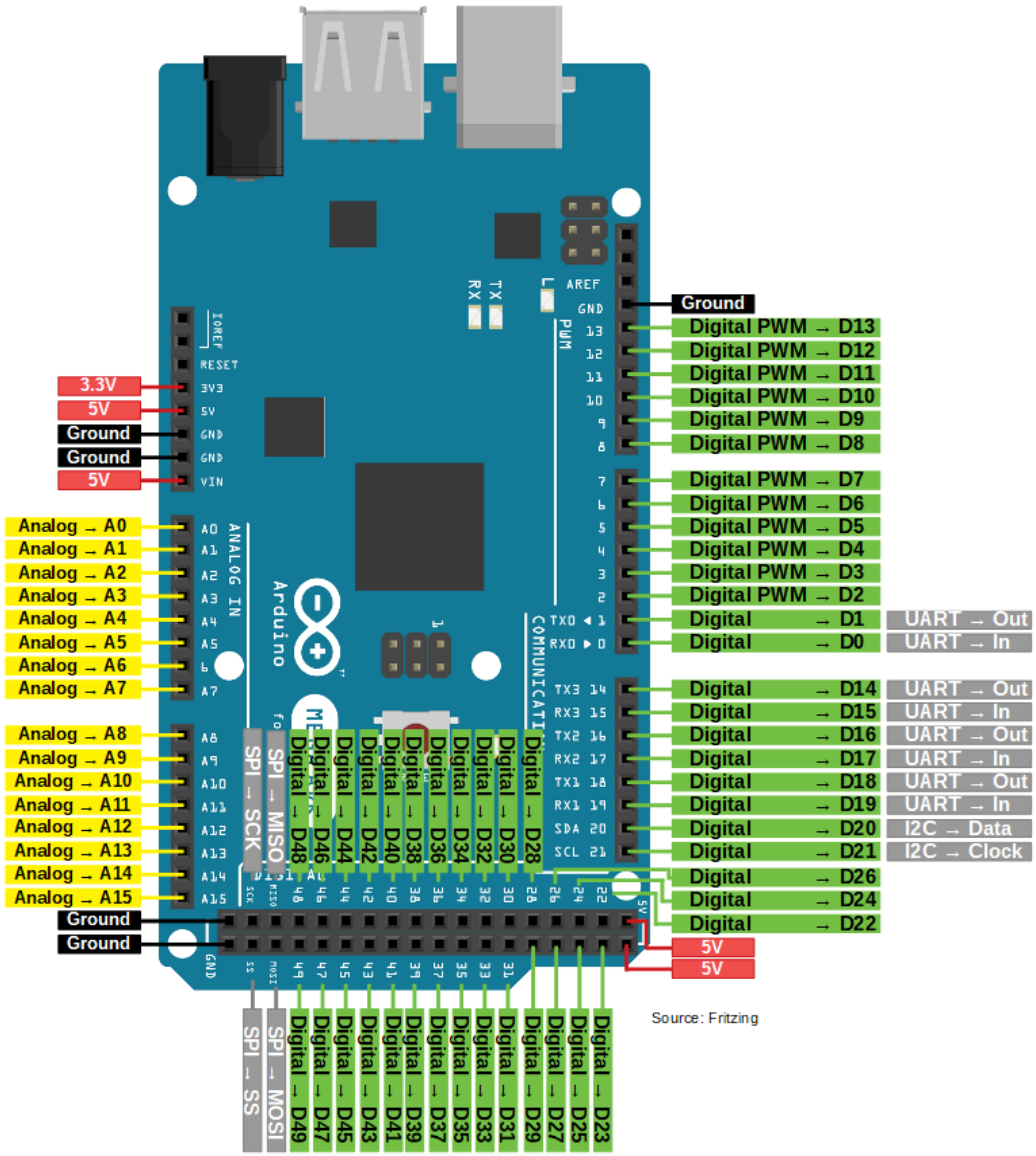


Fonte: UNO R3 (2019)

"[...]O **Arduino Mega 2560** é uma placa microcontrolada baseada no ATmega2560. Possui 54 pinos de entrada/saída digital (dos quais 15 podem

ser usados como saídas PWM), 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um oscilador de cristal de 16 MHz, uma conexão USB, um conector de alimentação, um conector ICSP, e um botão de reset. Ele contém tudo o que é necessário para dar suporte ao microcontrolador; basta conectá-lo a um computador com um cabo USB ou alimentá-lo com um adaptador AC-to-DC ou bateria para começar. A placa Mega 2560 é compatível com a maioria dos shields projetados para o Uno e as antigas placas Duemilanove ou Diecimila (SOUZA, 2020).

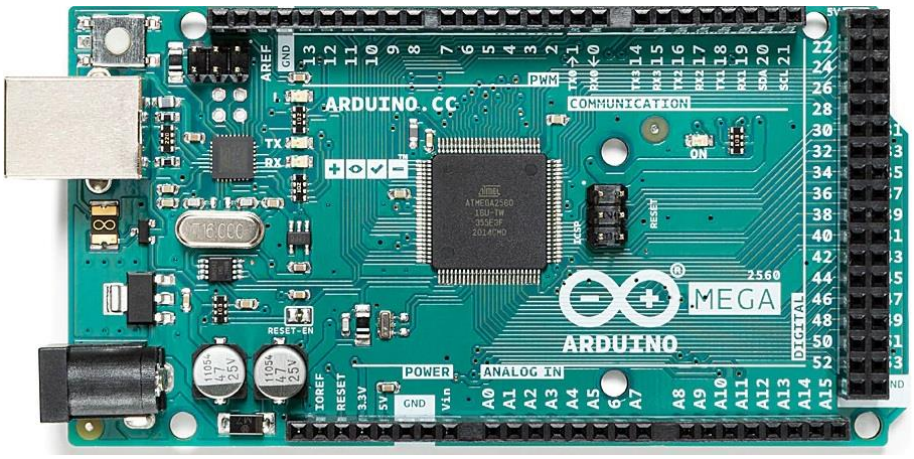
Figura 16 - Pinagem do Arduino Mega Rev3



Fonte: DIYI0T (2020)

A Figura 16 apresenta o Arduino ATmega2560 e toda sua pinagem disponível para utilização no desenvolvimento de aplicações.

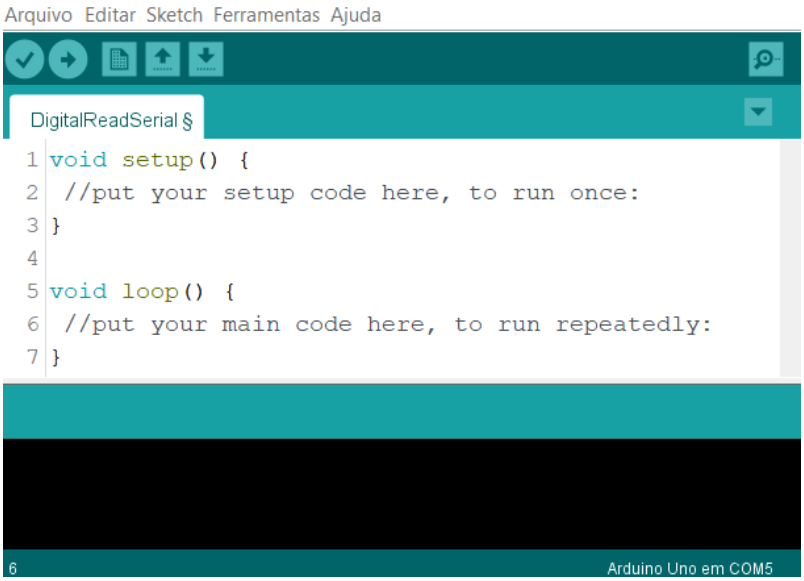
Figura 17 - Arduino Mega 2560 Rev3



Fonte: SOUZA (2020)

O Arduino ambiente de desenvolvimento (IDE - *Integrated Development Environment*) possui uma interface de edição de textos onde os códigos são escritos e os mesmos são salvos com a extensão.ino. Os programas escritos nessa IDE são chamados de *sketches*. A Figura 18 mostra uma imagem da tela da IDE usada para programar o Arduino.

Figura 18 - Print da tela da IDE utilizada para programar o Arduino UNO R3



Fonte: Autoria própria.

Além disso, a IDE possui uma área de mensagem onde são exibidas as informações de compilação do código e os erros encontrados no mesmo. Possui também console de texto que retorna informações ao salvar, exportar e erros e ainda possui uma barra de ferramentas com vários botões que permite fazer a verificação

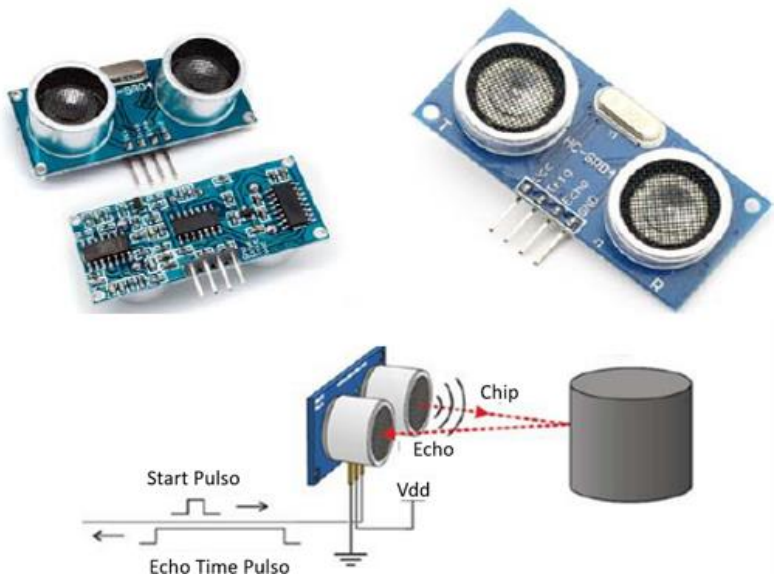
dos programas implementados assim como carregá-los, além de criar, abrir, salvar e monitorar a porta serial. O software se comunica com a placa do Arduino para fazer upload de programas assim como a comunicação com eles.

3.1.5 Transdutor de ultrassom HC-SR04

O sensor ultrassônico HC-SR04 é um sensor de proximidade muito utilizado em diferentes tipos de projetos, desde projetos robóticos para evitar colisões até projetos de detecção de nível e sensores de estacionamento. Este sensor tem um baixo custo e seu método de medição é muito fácil. O sensor HC-SR04 segunda geração ao contrário do seu antecessor pode operar entre 4,8V e 5V DC, tendo uma faixa de tensão de entrada maior, permitindo que o mesmo funcione com o controlador em 3,3V. A medição da distância é feita através de um sonar, um pulso ultrassônico de aproximadamente 40 kHz, acima da faixa audível pelo ser humano, é transmitido da unidade e para a medição da distância até um objeto é feito através da medição do tempo de retorno do eco (VIDAL, 2022).

O sensor possui uma ótima precisão de alcance e estabilidade e seu pacote é fácil de usar. Além disso, seus pinos integrados de 2,54 mm permitem que o sensor seja conectado facilmente em uma placa de ensaio sem a necessidade de solda, facilitando a montagem de protótipos.

Figura 19 - Sensor HC-SR04 e um exemplo do modo de funcionamento



Fonte: VIDAL (2022)

A Figura 19 apresenta o modo de funcionamento do HC-SR04 enviando um pulso e depois desse pulso ser refletido pelo objeto estudado, o sensor capta o eco que retorna em sua direção. A Tabela 2 apresenta os principais valores que o sensor possui em seu hardware. Esses valores são encontrados no datasheet do componente, documento desenvolvido a partir de testes realizados em laboratórios para quantificar a eficácia do dispositivo desenvolvido.

Tabela 2 - HC-SR04 *Ultrasonic Sensor Module Guide*

Parâmetros Elétricos	Valores
Tensão de Operação	3,3 Vdc ~ 5Vdc
Corrente Quiescent	<2 mA
Corrente de Operação	15 mA
Frequência de Operação	40 KHz
Faixa Operação e Precisão	2 cm ~ 400 cm (1in ~13ft) ± 3mm
Sensibilidade	-65 dB min
Pressão Sonora	112 dB
Ângulo Efetivo	15°
Conector	4 pinos *header* com 2,54 mm *pitch*
Dimensão	45 mm x 20 mm x 15 mm
Peso	9 g

Fonte: VIDAL (2022)

3.1.6 App Inventor

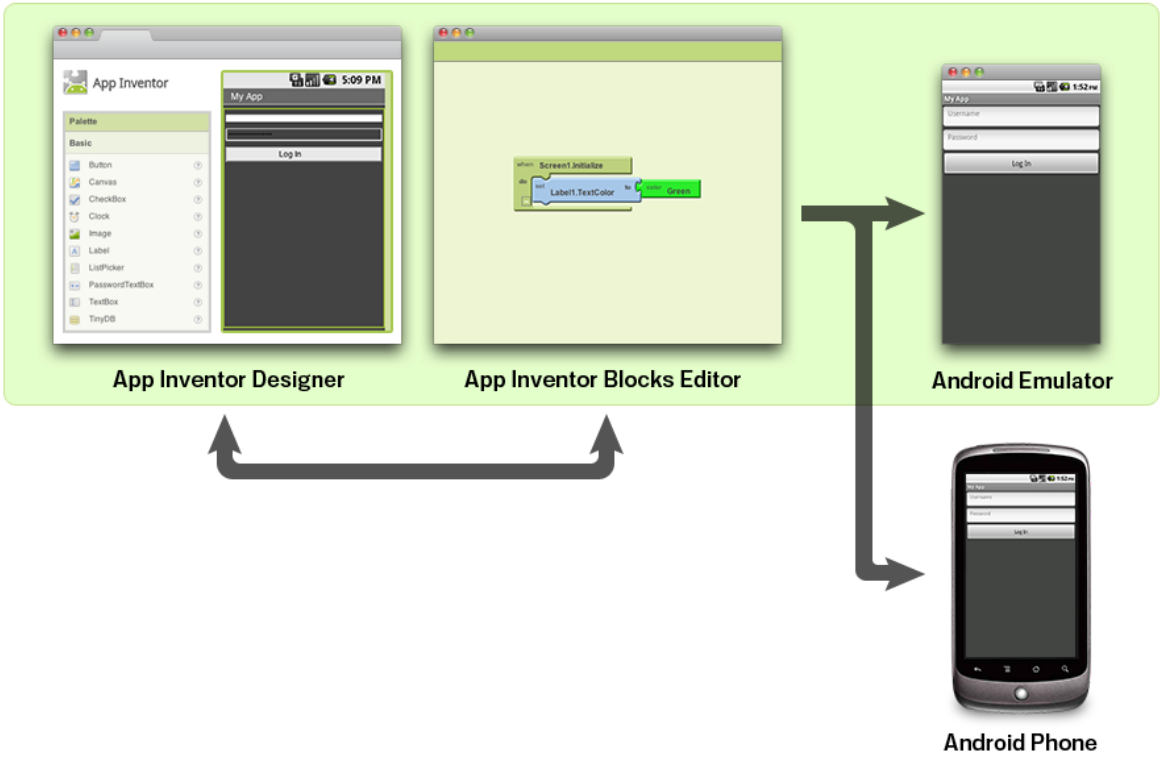
Desenvolvido pela empresa Google, o App Inventor trata-se de uma ferramenta elaborada para auxiliar na criação de aplicativos que utilizam o sistema operacional Android sem a necessidade de um conhecimento aprofundado em qualquer linguagem de programação, pois é um ambiente de programação visual e intuitivo, permitindo que a programação dentro do site possa ser desenvolvida por pessoas de qualquer idade. Atualmente essa ferramenta é suportada pelo Instituto de Tecnologia de Massachusetts (MIT). A figura 20 apresenta a logo do App Inventor.

Figura 20 - Logo da ferramenta App Inventor

Fonte: Adaptado O MIT App inventor (2022)

Uma equipe composta por funcionários e alunos do MIT compõem o núcleo de um grupo internacional de programadores. Para utilizar o App Inventor não existe a necessidade de instalá-lo por trata-se de um software web, ou seja, pode ser utilizado totalmente online para o desenvolvimento de aplicativos suportados pelo sistema Android. Para criar um aplicativo basta selecionar os componentes conforme as necessidades que o App será aplicado. A Figura 21 apresenta as telas de Designer, *Blocks Editor e Android Emulator*.

Figura 21 - Fluxo de funcionamento do App Inventor



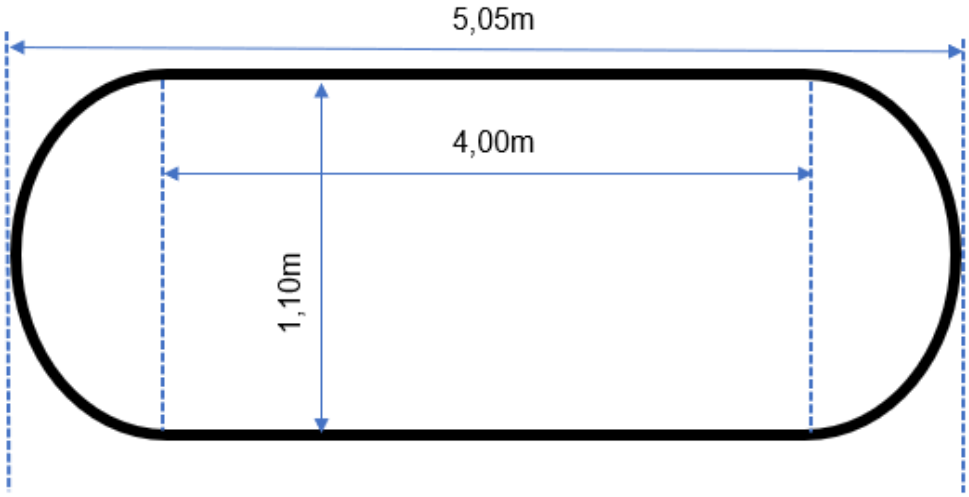
Fonte: Adaptado MIT App inventor (2022)

Os aplicativos desenvolvidos na plataforma App Inventor podem ser testados pelo desenvolvedor de duas formar diferentes. A primeira seria empacotando todo o conteúdo produzido em um aplicativo executável para instalar diretamente no celular, onde o site gera um *QRcode* que permite fazer o download do app diretamente no celular. A segunda opção para realização dos testes do app é utilizando um emulador disponível pela plataforma, onde a ideia é possibilitar o usuário desenvolver e testar suas ideias mesmo sem ter um telefone com sistema Android.

3.1.7 Construção da pista

A pista foi desenvolvida utilizando medidas que respeitam a limitação de detecção do sensor ultrassônico, dessa forma utilizou-se um TNT na cor branca com as medidas de cinco metros de comprimento por dois de largura. Como o seguidor de linha detecta a faixa a partir da diferença de cores entre a faixa e a superfície de fixação, fez-se o uso de uma fita isolante com largura de 2 cm colada no TNT. A Figura 22 apresenta um esboço da pista com suas dimensões.

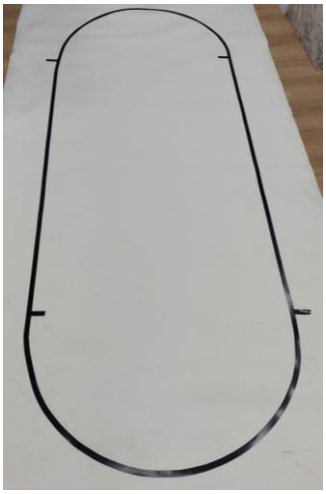
Figura 22 – Medidas pista projetada



Fonte: Autoria própria (2022)

Para o robô identificar que entrou na curva a pista terá 4 pontos fixos, feitos com fita isolante, no encontro da reta com a parte curva para que os sensores de infravermelho das pontas identifiquem a entrada e saída da curvo realizada pelo robô. A Figura 23 apresenta a pista construída em TNT e fita isolante para realização dos testes do assistente de velocidade testado nos robôs seguidores de linha.

Figura 23 - Pista construída em TNT e fita isolante



Fonte: Autoria própria (2022)

3.2 Métodos

O hardware utilizado na execução desse trabalho foi desenvolvido e construído especificamente para aplicação do assistente controlador de velocidade, dessa forma o sensor ultrassônico foi colocado na parte frontal do robô para fazer a detecção de obstáculos, moveis ou imóveis, na frente do seguidor de linha. A Figura 24 apresenta o protótipo do robô utilizado no trabalho e destaca o sensor ultrassônico na parte frontal do seguidor de linha.

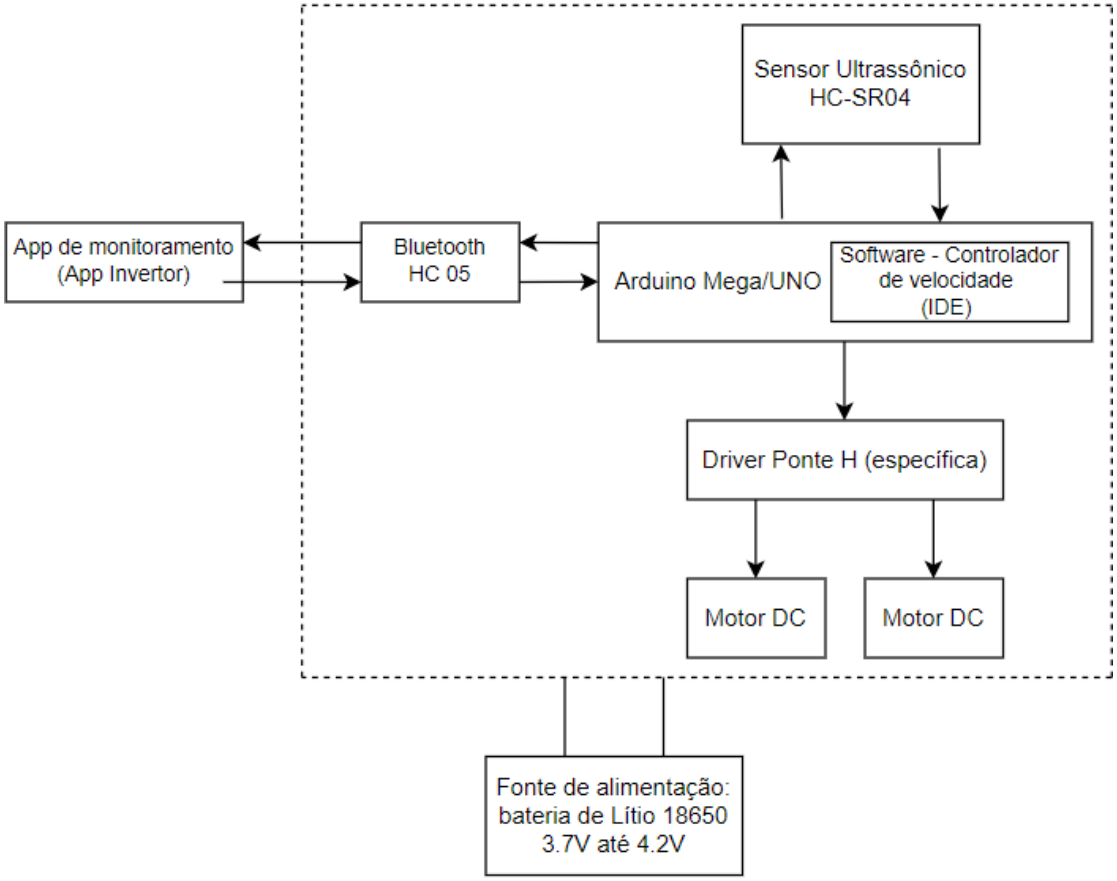
Figura 24 - Foto do robô seguidor de linha

Fonte: Autoria própria (2022)

Inicialmente foi realizado um estudo do robô e o diagrama elétrico disponibilizado para entender como o circuito foi construído e quais entradas e saídas do Arduino estavam sendo utilizadas. A partir disso iniciou-se o desenvolvimento da

programação para que a comunicação do hardware e software fosse possível. Primeiramente foi implementada a lógica para o funcionamento do seguidor de linha, para isso foram utilizados os 3 sensores centrais localizados na parte da frente do robô na parte inferior da placa para identificar a linha do circuito, os dois sensores laterais foram utilizados para a identificação da entrada e saída da curva. O código referente ao seguidor de linha é independente do controlador de velocidade, mas recebe os valores do mesmo ou do aplicativo de controle.

Figura 25 - Diagrama de funcionamento do sistema completo



Fonte: Autoria própria (2022)

A Figura 25 apresenta como ocorre a comunicação do software e hardware, onde o hardware englobado pela linha tracejada apresenta os componentes soldados na placa de circuito. O software contido no Arduino, envia um sinal para o sensor ultrassônico, esse sensor retorna valores para o Arduino e a partir desses valores quantifica as entradas que são usadas no fuzzy. A lógica implementada no Arduino auxilia no controle de velocidade do seguidor de linha a partir de parâmetros, como distância detectada pelo HC-RS04 e variação de velocidade. Esta distância é utilizada

no código para determinação das duas entradas utilizadas na lógica fuzzy. Após o recebimento dessas entradas o fuzzy retorna como saída uma velocidade que é transmitida para a ponte H e posteriormente aos motores DC.

3.2.1 Controle e autonomia dos robôs seguidores de linha

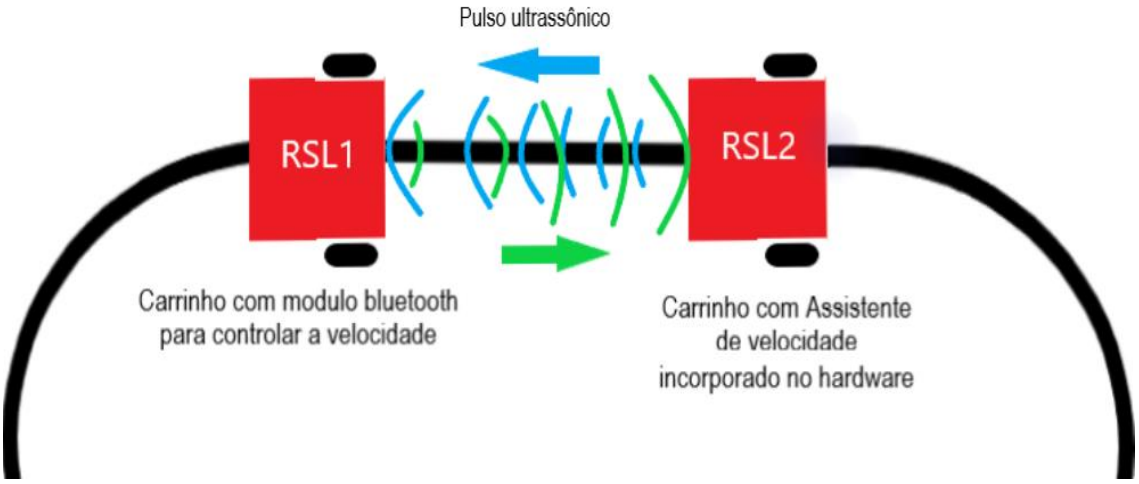
Os robôs seguidores de linha são direcionados através de uma faixa pré-determinada que é detectada através dos sensores TCR5000 para garantir a autonomia do sistema. Entretanto, para que essa autonomia seja eficiente é necessário que a montagem do sistema seja elaborada corretamente para que o hardware trabalhe em conjunto com o software. E para que isso ocorra é essencial que tenha o conhecimento do funcionamento de todos os componentes utilizados.

Para o controle de velocidade dos robôs será utilizado dois métodos, o principal de forma autônoma através da utilização de logica fuzzy, sendo esse o foco do presente trabalho. A aplicação do fuzzy será implementada para que o carro tenha autonomia em identificar quando o controle de velocidade deve atuar e para isso o processo usa como entradas a distância detectada pelo SR-HC04 e a variação de velocidade do robô seguidor de linha 2 (RSL2). E a partir dessas entradas o fuzzy retorna uma velocidade mais adequada conforme as regras que serão pré-determinadas.

O segundo método de controle de velocidade dos robôs acontece a partir do uso do protocolo de comunicação Bluetooth. Optou-se pela implementação dessa comunicação para realizar o monitoramento da velocidade dos seguidores de linha e para que possa ser enviado uma velocidade aos robôs de forma que possibilite a realização de testes que force o robô RSL2 responder a diferentes situações.

A utilização de dois robôs seguidores de linha se faz necessário neste trabalho, pois ambos se deslocam no mesmo circuito. Entretanto o RSL1 tem a função de auxiliar no teste do robô seguidor de linha RLS2, portador do assistente de velocidade.

Figura 26 - Ilustração do modo de funcionamento do projeto



Fonte: Autoria própria (2022)

A Figura 26 apresenta o modo de funcionamento dos robôs seguidores de linha, onde o portador do módulo Bluetooth, que terá a velocidade controlada através de um celular percorre a pista na dianteira, o robô com o assistente de velocidade está atrás e será controlado através da implementação da lógica fuzzy.

3.2.2 Detecção de objetos com sensor ultrassônico HC-SR04

O Arduino que funciona como um circuito externo, envia um pulso de trigger com duração de aproximadamente 10μs em nível logico alto, ao receber o sinal de trigger o sensor projeta 8 pulsos. Caso o receptor do HC-SR04 detecte o retorno desses pulsos o sensor envia um sinal de nível alto, com tempo de duração igual ao tempo calculado entre o envio e retorno da onda ultrassônica, através de seu pino de saída para o Arduino.

Através da diferença de tempo entre a emissão e recepção do sinal, calcula-se a distância que existe entre o sensor e um obstáculo, a partir da equação 4.

$$Distância = \frac{(\text{Diferença de tempo entre a emissão e recepção }) * (\text{Velocidade do som})}{2} \quad (4)$$

Utilizando a função pulseIn(echoPin, HIGH) do Arduino para detecção dos objetos, o sensor recebe um pulso e retorna o valor do tempo da duração deste pulso, em microssegundos. Com esse tempo é possível calcular a distância que existe entre

o obstáculo e o sensor, esse cálculo é feito através da Equação (4) e considerando a velocidade do som igual a 340m/s. A Figura 27 apresenta a classe implementada no código, utilizado no assistente controlador de velocidade, que realiza a detecção dos obstáculos.

Figura 27 - Lógica de detecção da distância dos obstáculos

```
75  class SensorDist {
76      int echoPin, trigPin, distance;
77      long tempo;
78
79    public:
80
81      void Pinout(int pin1, int pin2) {
82        echoPin = pin1;
83        trigPin = pin2;
84        pinMode(trigPin, OUTPUT); // Seta o trigPin como OUTPUT
85        pinMode(echoPin, INPUT); // Seta o echoPin como INPUT
86      }
87      long Distancia() { //Ler distância
88        // Clears the trigPin condition
89        digitalWrite(trigPin, LOW);
90        delayMicroseconds(10);
91        // Seta o trigPin HIGH por 10 microsegundos
92        digitalWrite(trigPin, HIGH);
93        delayMicroseconds(10);
94        digitalWrite(trigPin, LOW);
95        // Lê o echoPin, retorno da tempo da onda sonora em microsegundos
96        tempo = pulseIn(echoPin, HIGH);
97        // Calculating the distance
98        distance = tempo * 0.034 / 2; // Velocidade da onda sonora divida por 2, ida e volta
99
100       return distance;
101     }
102
103 };
```

Fonte: Autoria própria.

Nas linhas de código desenvolvido para a classe SensorDist utilizou-se funções pré-existentes no Arduino, como declarações de variáveis e leituras e escritas de uma determinada porta. A equação apresentada na linha 98 da figura 27, calcula-se a distância e retorna o valor que será utilizado na implementação do fuzzy. Note que está equação trata-se da equação 4, mas de forma aplicada no trabalho.

3.2.3 Aplicativos

O robô RSL2 controla a velocidade com o auxílio do sensor de ultrassom que faz a leitura de ambiente para saber se existe algum obstáculo à frente. Entretanto,

com o app também será possível controlar a velocidade do robô 2 alterando a autônima do controle de velocidade fornecida pela lógica fuzzy através do botão manual/automático.

Figura 28 - Print da interface de funcionamento do App *Controller*

Fonte: Autoria própria (2022)

O aplicativo apresentado na Figura 28 é utilizado para controlar o RSL2 que possui o assistente de velocidade. Esse aplicativo possibilita controlar a velocidade RSL2 na forma manual onde é possível setar um percentual da velocidade padrão ou aumentar e diminuir a velocidade através dos botões mais (+) para aumentar a velocidade e menos (-) para diminuir a velocidade. No modo automático o assistente de velocidade atuará no controle de velocidade do robô. Além disso, é possível acompanhar a velocidade atual pelo display que constantemente mostra a velocidade atual recebida via *bluetooth*. O robô seguidor de linha 1, que estará sempre na dianteira, será controlado através do módulo Bluetooth HC05 com auxílio do aplicativo desenvolvido na plataforma App Invertor.

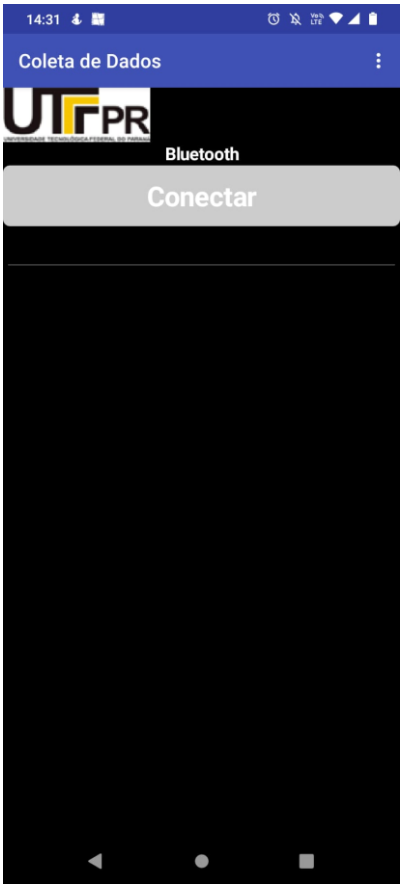
Para o RSL1 foi desenvolvido um aplicativo similar ao mostrado na Figura 28, a diferença está no botão automático/manual que não está presente no aplicativo do

RSL1, pois o controle do mesmo ocorre sempre em modo manual, pois não possui o assistente de velocidade. O layout e as funções dos demais componentes são os mesmos do aplicativo do RSL2. Como existe a limitação de conectar apenas um modulo bluetooth por celular, utilizaremos dois celulares, um para cada robô e seus respectivos aplicativos de controle. Os aplicativos foram desenvolvidos para sistema Android.

Além dos aplicativos de controle dos seguidores de linha foi desenvolvido um aplicativo para coleta de dados recebidos via bluetooth durante os testes. Os valores recebidos são a distância e a velocidade atual do robô seguidor de linha com o assistente de velocidade e os dados são armazenados em uma planilha do google através da internet.

Para o armazenamento dos dados em uma planilha do google foi utilizado um script no Apps Script do google que é uma plataforma Javascript em nuvem que permite integrar e automatizar tarefas nos produtos do Google.

Figura 29 - Print da interface APP Coleta de Dados



Fonte: Autoria própria (2022)

A Figura 28 apresentado o aplicativo desenvolvido para realizar o monitoramento e captura de dados.

4 RESULTADOS E DISCUSSÕES

A lógica implementada no Arduino auxilia no controle de velocidade do seguidor de linha a partir de parâmetros, como distância detectada pelo HC-RS04 e variação de velocidade. Esta distância é utilizada no código para determinação das duas entradas utilizadas na lógica fuzzy.

A comunicação do software e hardware, onde o software contido no Arduino, envia um sinal para o sensor ultrassônico, esse sensor retorna valores para o Arduino e a partir desses valores quantifica as entradas que são usadas no fuzzy. Após o recebimento dessas entradas o fuzzy retorna como saída uma velocidade que é transmitida para a ponte H e posteriormente aos motores DC.

O problema abordado nesse trabalho, quando representado no sistema fuzzy, foi constituído de variáveis de entrada/saída, regras fuzzy e método de inferência fuzzy. Como neste sistema um resultado surge como consequência de fatos e regras fuzzy, utilizou-se variáveis linguísticas para modelar o problema e elaborar uma base de regras. Esta base foi composta por dois universos de discurso, distância e velocidade, e cada um contém seus valores linguísticos que foram listados e apresentados no Quadro 2.

Quadro 1 - Variáveis linguísticas

Variáveis Linguísticas	Distância	Velocidade
Termos primários	Muito Perto	Muito Baixa
	Perto	Baixa
	Intermediária	Média
	Longe	Alta
	Muito Longe	Muito Alta
	Distante	--

Fonte: Autoria própria (2022)

A partir dos universos apresentados no Quadro 2 foi criado um conjunto de variáveis seguindo padrões de linguagem de programação utilizadas por desenvolvedores e boas práticas. Transcrevendo as variáveis linguísticas da língua portuguesa para o que foi utilizado dentro dos programas desenvolvidos para o

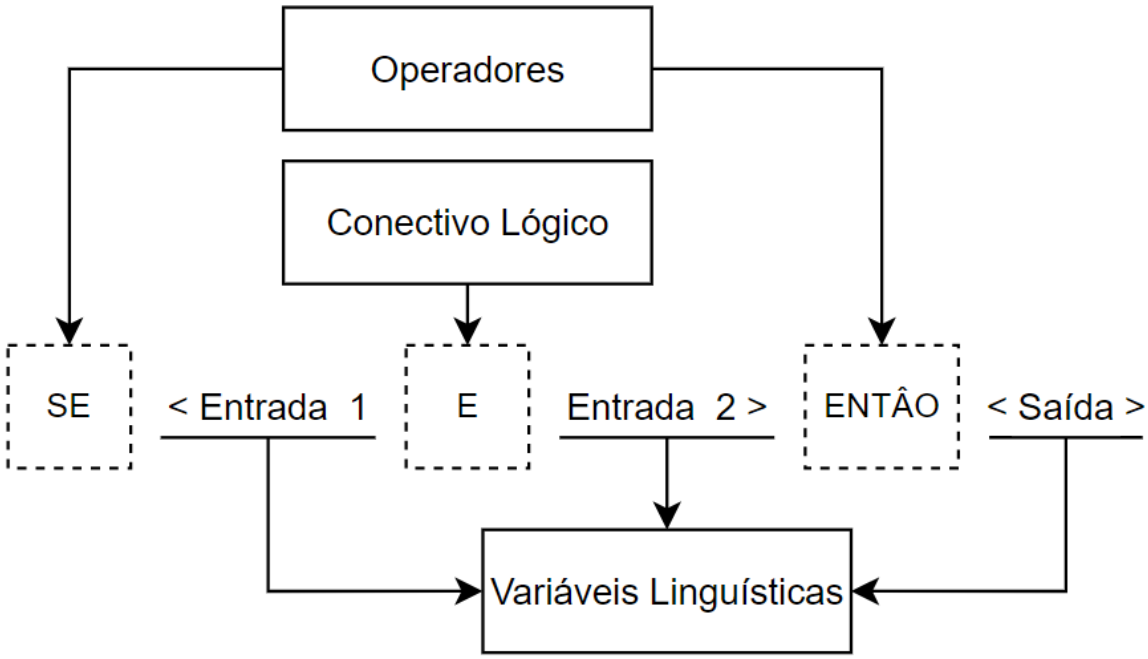
assistente de velocidade, listou-se as variáveis criadas juntamente com o seu significado em cada universo utilizado no sistema fuzzy. Para o universo da velocidade foi necessário utilizar um prefixo nas variáveis linguísticas para distinguir as variáveis de entradas e saída, dessa foi utilizado o prefixo VIn nas entradas e VelCar na saída.

- Variáveis relacionadas aos termos primários:
    - MuitoPerto = Muito Perto
    - Perto = Perto
    - Intermediária = Intermediária
    - MuitoLonge = Muito Longe
    - Distante = Distante

- Variáveis relacionadas ao universo da velocidade:
    - VInMuitoBaixa = Velocidade de entrada muito baixa
    - VInBaixa = Velocidade de entrada baixa
    - VInMedia = Velocidade de entrada média
    - VInAlta = Velocidade de entrada alta
    - VInMuitoAlta = Velocidade de entrada muito alta
    - VelCarMuitoBaixa = Velocidade de saída muito baixa
    - VelCarBaixa = Velocidade de saída baixa
    - VelCarMedia = Velocidade de saída média
    - VelCarAlta = Velocidade de saída alta
    - VelCarMuitoAlta = Velocidade de saída muito alta.

Criando as regras fuzzy a partir das variáveis linguísticas apresentadas no Quadro 2, cada regra elaborada foi composta de proposições combinadas através de conectivos lógicos e operadores. Como a entrada formou-se da composição de duas variáveis foi usado o conector e, pois determinada resposta só pode ser retornada caso as variáveis de entrada tivessem ocorrido conjuntamente. E para completar a lógica fez se necessário o uso do operador se...então para dar clareza aos termos de relações fuzzy. A Figura 30 apresenta como constituiu-se cada umas das regras implementadas na lógica fuzzy.

Figura 30 - Diagrama do esquema da construção das proposições fuzzy



Fonte: Autoria própria (2022)

Onde as estradas são antecedentes, situações contidas no universo da distância e velocidade que iram gerar a saída consequente, que representando a resposta ao antecedente, contida no universo da velocidade e partir disso gerou-se os conjuntos fuzzy. A Tabela 3 apresenta as 30 regras que foram elaboradas para constituir a base do conhecimento, note que para cada fato antecedente será retornado uma resposta como consequência correspondendo a uma das regras fuzzy.

Tabela 3 – Base de conhecimento

REGRA	ENTRADAS (Antecedentes)				SAÍDAS (Consequentes)	
		DISTÂNCIA		VARIAÇÃO DE VELOCIDADE		VELOCIDADE DE SAIDA
1	**SE**	MuitoPerto	**E**	VinMuitoBaixa	**ENTÃO**	VelCarMuitoBaixa
2	**SE**	Perto	**E**	VinMuitoBaixa	**ENTÃO**	VelCarMuitoBaixa
3	**SE**	Intermediaria	**E**	VinMuitoBaixa	**ENTÃO**	VelCarBaixa
4	**SE**	Longe	**E**	VinMuitoBaixa	**ENTÃO**	VelCarMedia
5	**SE**	MuitoLonge	**E**	VinMuitoBaixa	**ENTÃO**	VelCarAlta
6	**SE**	Distante	**E**	VinMuitoBaixa	**ENTÃO**	VelCarMuitoAlta
7	**SE**	MuitoPerto	**E**	VinBaixa	**ENTÃO**	VelCarBaixa
8	**SE**	Perto	**E**	VinBaixa	**ENTÃO**	VelCarMedia
9	**SE**	Intermediaria	**E**	VinBaixa	**ENTÃO**	VelCarMedia
10	**SE**	Longe	**E**	VinBaixa	**ENTÃO**	VelCarAlta

11	**SE**	MuitoLonge	**E**	VinBaixa	**ENTÃO**	VelCarAlta
12	**SE**	Distante	**E**	VinBaixa	**ENTÃO**	VelCarMuitoAlta
13	**SE**	MuitoPerto	**E**	VinMedia	**ENTÃO**	VelCarMedia
14	**SE**	Perto	**E**	VinMedia	**ENTÃO**	VelCarMedia
15	**SE**	Intermediaria	**E**	VinMedia	**ENTÃO**	VelCarMedia
16	**SE**	Longe	**E**	VinMedia	**ENTÃO**	VelCarAlta
17	**SE**	MuitoLonge	**E**	VinMedia	**ENTÃO**	VelCarAlta
18	**SE**	Distante	**E**	VinMedia	**ENTÃO**	VelCarMuitoAlta
19	**SE**	MuitoPerto	**E**	VinAlta	**ENTÃO**	VelCarBaixa
20	**SE**	Perto	**E**	VinAlta	**ENTÃO**	VelCarBaixa
21	**SE**	Intermediaria	**E**	VinAlta	**ENTÃO**	VelCarMedia
22	**SE**	Longe	**E**	VinAlta	**ENTÃO**	VelCarMedia
23	**SE**	MuitoLonge	**E**	VinAlta	**ENTÃO**	VelCarAlta
24	**SE**	Distante	**E**	VinAlta	**ENTÃO**	VelCarAlta
25	**SE**	MuitoPerto	**E**	VinMuitoAlta	**ENTÃO**	VelCarBaixa
26	**SE**	Perto	**E**	VinMuitoAlta	**ENTÃO**	VelCarMedia
27	**SE**	Intermediaria	**E**	VinMuitoAlta	**ENTÃO**	VelCarMedia
28	**SE**	Longe	**E**	VinMuitoAlta	**ENTÃO**	VelCarAlta
29	**SE**	MuitoLonge	**E**	VinMuitoAlta	**ENTÃO**	VelCarAlta
30	**SE**	Distante	**E**	VinMuitoAlta	**ENTÃO**	VelCarMuitoAlta

Fonte: Autoria própria (2022)

Para cada variável linguística foi atribuído um intervalo de valor onde a distância é representada pela unidade de medida centímetros e a velocidade possui uma faixa de valor analógico de 0 até 255. Esses intervalos foram determinados a partir dos testes realizados no seguidor de linha quando o mesmo estava apenas com o código desenvolvido para deslocar-se apenas na faixa predetermina.

Tabela 4 - Parametrização das variáveis de entradas relacionadas a distância

DISTÂNCIA (10cm até 400,1cm)		
SIGLA	**INTERVALO**	
MuitoPerto	0	10
Perto	9	30
Intermediario	29	40
Longe	39	70
MuitoLonge	99	200
Distante	199	400,1

Fonte: Autoria própria (2022)

Tabela 5 - Parametrização das variáveis de entradas relacionadas a variação de velocidade

VARIAÇÃO DA VELOCIDADE (0 até 255)		
SIGLA	**INTERVALO**	
VinMuitoBaixa	0	30
VinBaixa	29	50
VinMedia	49	70
VinAlta	69	110
VinMuitoAlta	109	160,1

Fonte: Autoria própria (2022)

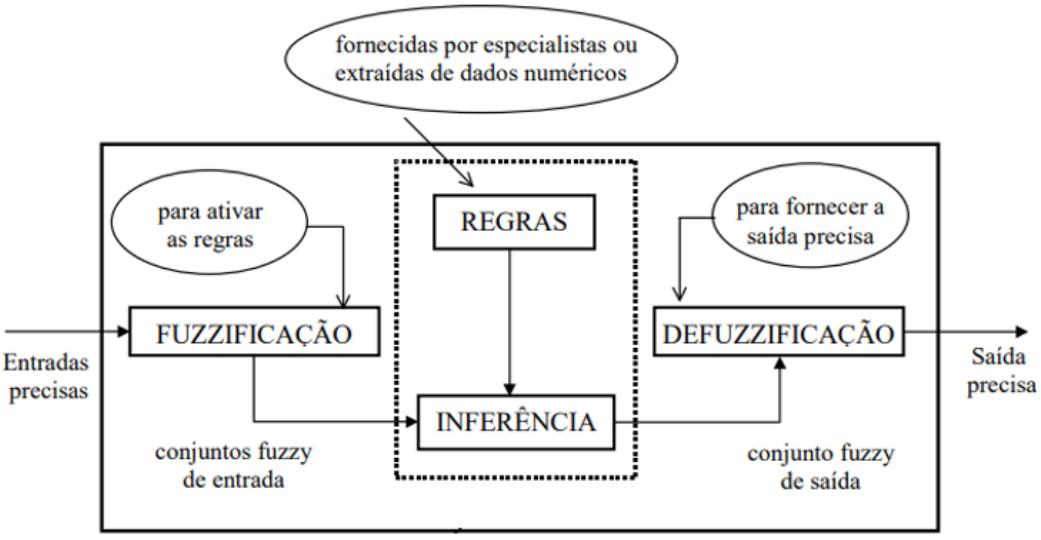
Tabela 6 - Parametrização das variáveis de entradas relacionadas a velocidade de saída

VELOCIDADE DE SAÍDA (0 até 255)		
SIGLA	**INTERVALO**	
VelCarMuitoBaixa	0	50
VelCarBaixa	49	90
VelCarMedia	89	110
VelCarAlta	109	130
VelCarMuitoAlta	129	160,1

Fonte: Autoria própria (2022)

As tabelas 4, 5 e 6 serviram para implementação da lógica fuzzy usada no controlador de velocidade. A Tabela 4 e 5 contém os parâmetros das variáveis de entrada usadas para ativar as regras na fuzzificação, a Tabela 6 possui os valores das velocidades de saída que serão retornadas para o robô após o deffuzificação da inferência. A Figura 31 apresenta um esquema de como ocorre a inferência fuzzy usada no trabalho.

Figura 31 - Diagrama do sistema de inferência fuzzy.



Fonte: TANSCHEIT (2020)

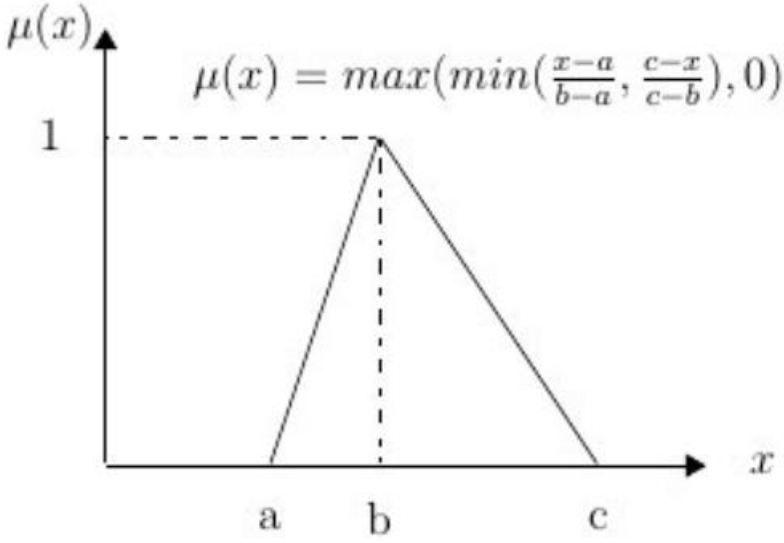
Esse sistema de inferência fuzzy foi utilizado para utilizar as entradas não fuzzy, estabelecidas nas regras a partir dos testes realizados e apresentados nas Tabelas 4, 5 e 6. Para que a inferência possa ocorrer foi executado a estruturação dos dados na fase de fuzzificação. Como este fato gera o conjunto fuzzy através da inferência ele possibilita a defuzzificação que interpreta as informações.

Matematicamente os cálculos acontecem na mesma sequência que a Figura 31 apresentou. Iniciando pela fuzzificação do sistema, quando o código recebe os valores detectados pelo hardware ele determina a regra que será ativada e realiza a fuzzificação através da Equação 4 de máximos e mínimos.

$$\boldsymbol{\mu}(x, a, b, c) = \max\left(\min((x - a/b - a)\,;(c - x\ /b - c)),0\right) \qquad (3)$$

A Equação 3 define uma função de pertinência triangular, como apresentado na Figura 32.

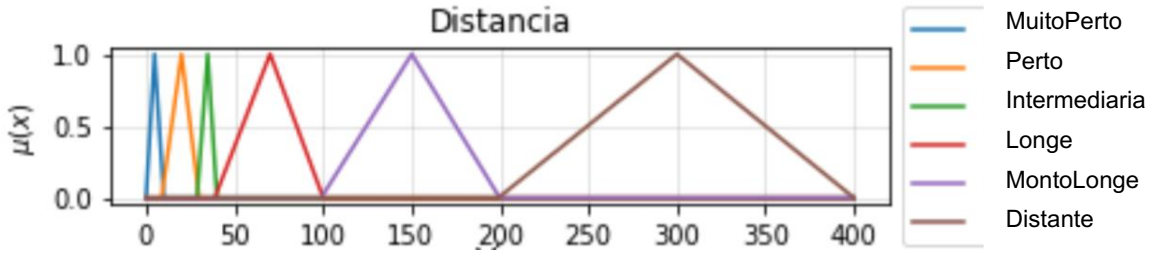
Figura 32 - Função de pertinência triangular dos conceitos fuzzy



Fonte: GAFA (2020)

Com o auxílio do Colab, ferramenta desenvolvida pela empresa google, foi possível apresentar as Tabelas 4 e 5 para facilitar na visualização dos cálculos. A Figura 33 apresenta graficamente os valore das entradas referentes a distância visualização da utilização da Equação 3.

Figura 33 - Gráfico da parametrização do Universo da distância



Fonte: Autoria própria (2022)

Quando detectado uma distância de 120cm, a lógica fuzzy entende que esse valor está contido no termo primário MuitoLonge, como apresentada pela linha da cor roxa no gráfico, então extraindo os valores dessa variável e substituindo na Equação (3), obteve-se os resultados apresentados abaixo.

$$\mathbf{\mu MuitoLonge}(Distancia) =$$

$$\max\left(\min((120 - 100/150 - 100)\,;(200 - 120/200 - 150));0\right) \quad (5)$$

$$\mathbf{\mu MuitoLonge}(Distancia) = \max\left(\min((20/50))\,;(80/50));0\right) \quad (6)$$

$$\mathbf{\mu MuitoLonge}(Distancia) = \max\left(\min\left(0{,}4\,;1{,}6\right));0\right) \quad (7)$$

$$\mathbf{\mu MuitoLonge}(Distancia) = \max\left(\min\left(0{,}4\right);0\right) \quad (8)$$

$$\mathbf{\mu MuitoLonge}(Distancia) = \max\left(0{,}4;0\right) \quad (9)$$

$$\boldsymbol{\mu MuitoLonge}(Distancia) = 0{,}4 \quad (10)$$

Figura 34 - Gráfico da parametrização do Universo da velocidade

Fonte: Autoria própria (2022)

A variação de velocidade recebida como velocidade de entrada (Vin = 80) está contida na faixa representada pela variável linguística *Alta,* e é representada pela linha de cor vermelha no gráfico da Figura 36.

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = \max\left(\min((80 - 69/110 - 90)\,;(110 - 80/110 - 90));0\right) \quad (11)$$

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = \max\left(\min((11/20))\,;(30/20));0\right) \quad (12)$$

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = \max\left(\min\left(0{,}55\,;1{,}5\right));0\right) \quad (13)$$

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = \max\left(\min\left(0{,}55\right);0\right) \quad (14)$$

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = \max\left(0{,}55;0\right) \quad (15)$$

$$\boldsymbol{\mu}\mathbf{Alta}(Vin) = 0{,}55 \quad (16)$$

É necessário ressaltar que ao realizar os mesmos cálculos da fuzzificação para as demais variáveis contidas nos dois universos, tem-se que o resultado dará zero.

Tabela 7 - Resultado da fuzzificação do universo da distância e variação ode velocidade

Termos primários	Resultado
MuitoPerto	0
Perto	0
Intermediario	0
Longe	0
MuitoLonge	0,44
Distante	0
VinMuitoBaixa	0
VinBaixa	0
VinMedia	0
VinAlta	0,55
VinMuitoAlta	0

Fonte: Autoria própria (2022)

Para cada consequência tem-se um conjunto de pertinência para inferência. Observando a Tabela 3 nota-se que a regra 35 será a única ativada pela fuzzificação das variáveis. Dessa forma, o assistente de velocidade deve enviar uma velocidade

alta para o robô autônomo, sendo está a conclusão alcançada pela lógica fuzzy para enfrentar a situação observada pelo seguidor de linha. Se observar a Figura 36 nota-se que a velocidade Alta se encontra aproximadamente entre 110 e 130, representando pela cor vermelha no gráfico.

Figura 35 - Gráfico utilizado para defuzzificação da velocidade

Fonte: Autoria própria (2022)

Executando matematicamente a inferência sobre a regra ativada pelo resultado da fuzzificação utiliza-se a função de pertinência do conjunto da velocidade de saída Velcar ativado pelo menor resultado apresentado pela fuzzificação, no caso apresentado será 0,44. A escolha do menor valor é devido a lógica e usada nas funções para garantir a abrangência de todas as condições. Aplicando a equação 3 para obter o resultado, partir do universo de pontos escolhidos, tem se:

U = [112.5; 115; 117.5; 120; 122.5; 125; 127.5;130]

$$\mu(0{,}44) = \frac{112.5 * 0.44 + 115 * 0.44 + 117.5 * 0.44 + 120 * 0.44 + 122.5 * 0.44 + \ 125 * 0.44 + 127.5 * 0.44 + 130 * 0.44}{0{,}44 + 0{,}44 + 0{,}44 + 0{,}44 + 0{,}44 + 0{,}44 + 0{,}44 + 0{,}44} \quad (17)$$

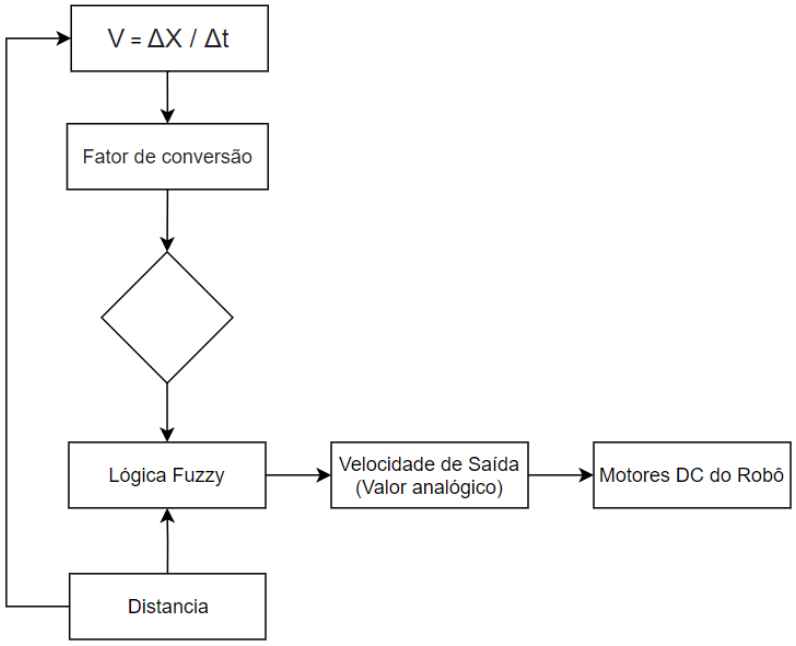
$$\mu = \frac{426{,}80}{3{,}52} \quad (18)$$

$$\mu = \ 121{,}25 \quad (19)$$

Como o resultado obtido nos retorna valores analógicos na faixa de 0 até 255 foi necessário converter esse resultado para uma unidade que represente a velocidade, no caso do presente trabalho foi usado centímetros/segundo. Isso se fez necessário devida a forma que ocorre a lógica dentro do controlador de velocidade, pois ele poderá utilizar o valor analógico de saída caso não tenha obstáculos em sua

frente ou velocidade que deverá ser convertida. A Figura 37 apresenta um diagrama do modo de funcionamento do programa desenvolvido para o assistente.

Figura 36 -Diagrama da lógica do assistente controlador de velocidade



Fonte: Autoria própria (2022)

Para realizar essa conversão foi necessário efetuar a captura de dados, para isso foi marcado a distância de um metro na pista e através de filmagens coletou-se o tempo que o robô levava para percorrer a distância demarcada na pista.

Figura 37 - Pista com a distância de um metro demarcado

Fonte: Autoria própria (2022)

Com auxílio do Excel, utilizou-se uma planilha para determinar o fator de conversão dos valores analógicos para velocidade (centímetros / segundo). Para realizar a coleta de dados apresentado na Tabela 8 foi fixada um valor analógico de

119,63 que foi enviado para os motores do robô. A partir dos valores foi calculado a velocidade apresentada para cada medicação realizada, depois foi determinado o fator de conversão a partir da equação 20.

$$Fc = \frac{Va}{Mv} \tag{20}$$

Onde Fc representa o fator de conversão, Mv a média das velocidades obtidas a partir dos dados compilados e Va o valor analógico.

Tabela 8 - Determinação do fator de conversão Analógica para centímetros / segundos

Nº das voltas	Valor Analógica (Va)	Distância (cm)	Tempo (s)	Velocidade (cm/s)	Média. Vel. (Mv)	Fator Conversão (Fc)
1	119.63	100	3	33.33	33.33	3.58890
2	119.63	100	2.81	35.59		
3	119.63	100	3.01	33.22		
4	119.63	100	2.8	35.71		
5	119.63	100	3.21	31.15		
6	119.63	100	3.03	33.00		
7	119.63	100	3.07	32.57		
8	119.63	100	2.77	36.10		
9	119.63	100	2.87	34.84		
10	119.63	100	2.96	33.78		
11	119.63	100	2.86	34.97		
12	119.63	100	3.27	30.58		
13	119.63	100	3.03	33.00		
14	119.63	100	3.2	31.25		
15	119.63	100	3	33.33		
16	119.63	100	3	33.33		
17	119.63	100	2.93	34.13		
18	119.63	100	2.98	33.56		
19	119.63	100	3.01	33.22		
20	119.63	100	2.99	33.44		

Fonte: Autoria própria (2022)

E após a determinação do fator de conversão foi utilizado esse valor na construção da lógica do controlador de velocidade. Para o funcionamento da lógica fuzzy foi utilizado a biblioteca eFLL (*Embedded Fuzzy Logic Library*) desenvolvida na Universidade Estadual do Piauí (UESPI-Teresina) pelo *Robotic Research Group* (RRP), essa biblioteca é usada em aplicações com microprocessadores como Arduino. A Figura 39 apresenta a forma de incluir a biblioteca fuzzy no programa desenvolvido. Antes de incluir a biblioteca no código do Arduino é necessário instalar

a biblioteca e isso pode ser feito através de uma pesquisa no gerenciador de bibliotecas presente na IDE dor Arduino.

Figura 38- Inclusão da biblioteca fuzzy

```
1 #include <Fuzzy.h>
2 //Include the SoftwareSerial library
3 #include "SoftwareSerial.h"
```

Fonte: Autoria própria (2022)

Utilizando o conceito de programação orientada a objetos, inicia-se o desenvolvimento com a declaração do objeto fuzzy, este objeto contém o Sistema Fuzzy completo. Dentro do ambiente de desenvolvimento (IDE) do Arduino, declara-se o objeto conforme apresentado na Figura 39, linha 107.

Figura 39 - Declaração do Objeto Fuzzy dentro do programa

```
106 // Instantiating a Fuzzy object
107 Fuzzy *fuzzy = new Fuzzy();
```

Fonte: Autoria própria (2022)

O objeto FuzzyInput contém o universo dos conjuntos de entradas. A Figura 40 retrata a implementação da entrada pertencente ao domínio da Distância, note que na linha 132 está sendo declarada o universo referente ao domínio. Outro objeto de extrema importância que será utilizado para o funcionamento do controlador de velocidade, trata-se do FuzzySet. Este objeto conterá as funções de pertinências triangulares.

Figura 40 - Declaração do Objeto FuzzyInput relacionada com a distância

```
131 // FuzzyInput
132 FuzzyInput *Distancia = new FuzzyInput(1);
133
134 FuzzySet *MuitoPerto = new FuzzySet(0, 10, 10, 15);
135 Distancia->addFuzzySet(MuitoPerto);
```

Fonte: Autoria própria (2022)

De forma semelhante à construção do FuzzyInput, relacionado a distância, também foi realizado a implementação da variação de velocidade e para o FuzzyOutput referente a velocidade de saída. A Figura 40 apresenta o objeto

FuzzyInput referente a variação da velocidade, a Figura 41 apresenta a declaração do objeto FuzzyOutput.

Figura 41 - Declaração do Objeto FuzzyInput relacionada com a variação da velocidade

```
150  // FuzzyInput
151  FuzzyInput *VinInput = new FuzzyInput(2);
152
153  FuzzySet *VinMuitoBaixa = new FuzzySet(10, 15, 15, 20);
154  VinInput->addFuzzySet(VinMuitoBaixa);
```

Fonte: Autoria própria.

Figura 42 - Declaração do Objeto FuzzyOutput relacionada com a velocidade de saída

```
166  // FuzzyOutput
167  FuzzyOutput *VelCar = new FuzzyOutput(1);
168
169  FuzzySet *VelCarMuitoBaixa = new FuzzySet(0, 30, 30, 50);
170  VelCar->addFuzzySet(VelCarMuitoBaixa);
```

Fonte: Autoria própria.

Os cálculos que a biblioteca fuzzy realizar para determinar o valor analógico retornado para o robô, utiliza os conceitos iniciados na equação de 3 até a 19. O código completo desenvolvido para o controlador de velocidade encontre em anexo no Apêndice A.

4.1 Análise dos resultados

Para o desenvolvimento do assistente de velocidade foram utilizados dois robôs seguidores de linha sendo que o projeto do mesmo já existia e a placa de circuito impresso já havia sido confeccionada. Com o diagrama elétrico do robô seguidor de linha em mãos e com dois robôs montados com todos os sensores citados em materiais e métodos neste trabalho, foram desenvolvidas as funções para que o robô fosse capaz de seguir a linha e utilizada a lógica fuzzy para controlar a velocidade do robô autônomo. Além disso foram desenvolvidos aplicativos para auxiliar no controle dos mesmos e um aplicativo específico para coleta de dados.

4.1.1 Faixa de velocidade de funcionamento do seguidor de linha

Foi possível observar que os robôs seguidores de linha possuíam uma limitação de velocidade devido o funcionamento do conjunto, pois ao consultar o datasheet dos componentes, integrados no hardware do seguidor de linha, encontrava-se valores acima dos capturados nos testes realizados. A Tabela 9 apresenta os valores teóricos e os valores observados através dos testes realizados.

Tabela 9 - Confronto dos valores teóricos e observados

Velocidade (0 até 255)	Mínimo	Máximo
Datasheet / Teórico	0	255
Observado	60	160

Fonte: Autoria própria.

Os valores teóricos encontrados na Tabela 9 referem-se aos limites apresentados no datasheet dos motores utilizados nos robôs. Entretanto, para o funcionamento do robô seguidor de linha foi necessário estabelecer uma faixa de valor analógico, pois valores a baixo de 60 não aplicava uma tensão suficiente nos motores para tirar o robô da inercia e movimenta-lo, já valores acima de 160 fazia com que o e seguidor de linha se perde a roda facilmente, pois o sistema não apresenta uma velocidade de resposta rápida para mante o robô no circuito.

A implicação que essa limitação de velocidade apresentou no desenvolvimento do assistente de velocidade era notada na dificuldade de observar com clareza a variação de velocidade sem o auxilio do aplicativo de monitoramento para observar os valores resultante do funcionamento do sistema.

4.1.2 Coleta de dados

Os dados coletados através do aplicativo permitiram observar a variação de velocidade que o robô apresentou nos testes conforme ocorria a leitura das variáveis de entrada, distância e variação de velocidade. Apesar da coleta de dados não apresentar quais regras estão sendo usadas em cada situação que o seguidor de linha detecta através da lógica fuzzy é possível observar que conforme ocorre uma variação na distância também se observa uma variação na velocidade que é retornada para o robô. A Tabela 10 apresenta os dados coletados via aplicativo da velocidade setada

no robô e distância, conforme o aplicativo recebia as informações do seguidor de linha esses dados eram armazenados, em tempo real, na nuvem via google.

Tabela 10 - Dados coletados via bluetooth para monitoramento do assistente

Velocidade (0 a 255)	Distância (cm)	Velocidade (0 a 255)	Distância (cm)	Velocidade (0 a 255)	Distância (cm)	Velocidade (0 a 255)	Distância (cm)
90	13	74	17	90	45	90	48
74	13	74	18	90	45	90	48
90	13	91	18	89	48	89	50
90	13	91	17	118	48	118	47
90	13	89	62	118	50	90	48
74	13	118	60	90	48	118	48
74	13	90	57	89	62	90	50
90	16	89	56	89	48	118	52
74	18	118	54	89	47	89	48
74	17	118	54	89	47	90	48
90	16	89	50	118	48	90	50
90	16	89	47	90	61	89	48
74	18	89	48	118	48	89	41
90	17	89	46	90	52	119	41
74	17	89	47	90	52	119	41
74	17	90	53	118	47	89	41
74	18	90	48	118	48	89	40
90	17	90	47	90	47	90	41
74	17	89	47	89	48	119	40

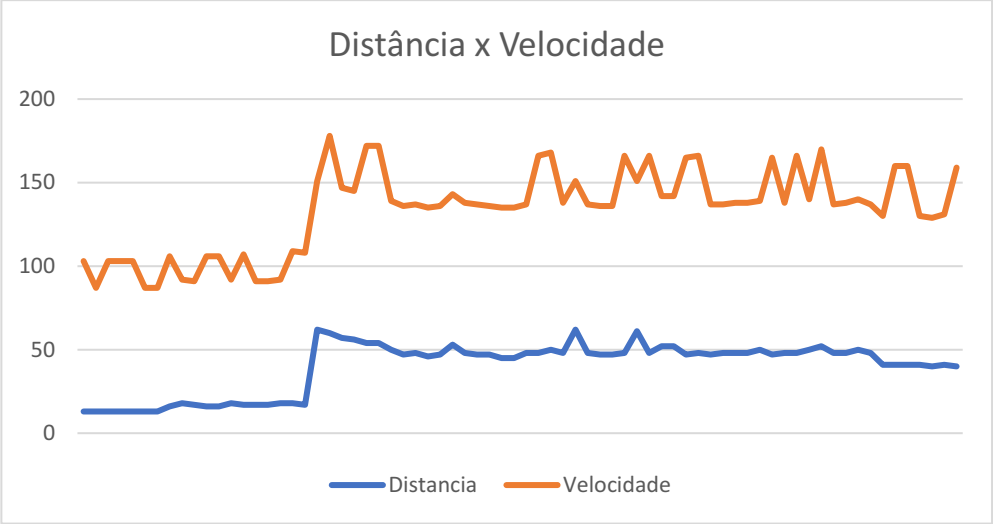
Fonte: Autoria própria (2022)

Esses dados foram coletados durante um teste, para manter apesar valores que expressão o trabalho do assistente de velocidade foi descartado os valores da velocidade quando o robô se encontrava na curva. Além disso, todos os valores armazenados no banco de dados que não estavam contidos na faixa de distância usada no fuzzy foram descartados, pois essa condição fazia a programar retornar uma velocidade fixa para robô.

4.1.3 Análise dos dados coletados

Analisando o gráfico 1 distância versus velocidade é possível notar que em alguns pontos quando há o aumento de distância há também um aumento da velocidade, no entanto em outros pontos quando há uma diminuição da distância há uma diminuição da velocidade mais discreta. Sendo assim, o assistente de velocidade é funcional, mas pode ser melhorado.

Gráfico 1 – Valores distância x velocidade coletados pelo aplicativo



Fonte: Autoria própria.

O robô seguidor de linha autônomo utiliza o assistente de velocidade somente nas retas e nas curvas sua velocidade é sempre constante. Durante a coleta de dados foram armazenados tanto os dados da velocidade e distância na reta quanto na curva. Desprezando os dados coletados durante as curvas foi gerado o gráfico 1 que apresenta a relação da velocidade pela distância.

Pelo gráfico é possível observar que quando há um aumento da distância há também um aumento da velocidade, em alguns momentos o aumento da velocidade não é tão repentino, no entanto, em outros é visível alguns picos de velocidade com uma variação pequena na distância. O ideal seriam curvas mais suaves.

4.1.4 Dificuldades enfrentadas

Apesar dos resultados alcançados, alguns pontos que dificultaram a execução do projeto serão listados para que fique claro as limitações que foram encontradas conforme ocorriam o desenvolvimento do trabalho.

- Problemas de montagem de alguns componentes, tais como bluetooth e motores;
- Capacidade de processamento insuficiente do Arduino Uno para o projeto proposto;

- Algumas falhas de projeto, todos os pinos disponíveis no Arduino Uno estavam sendo utilizados e adicionar mais componentes seria inviável;
- Placa impressa não tinha pinagem prevista para adicionar o Arduino Mega;

5 CONCLUSÃO

O objetivo proposto pelo assistente de velocidade em fuzzy foi alcançado, pois o mesmo é capaz de controlar a velocidade do seguidor de linha autônomo de acordo com a distância do robô seguidor de linha a sua frente e evitar colisões. Com o desenvolvimento do projeto conseguimos ao final dele obter um hardware funcional, mesmo que precisando de alguns ajustes, um software capaz de controlar a velocidade do robô e dois aplicativos de controle e um de coleta. Dentre os aplicativos vale destacar a coleta de dados, pois a coleta foi muito importante para se ter uma análise mais precisa do que estava acontecendo com o robô enquanto estava em funcionamento. Sem essa coleta de dados não conseguiríamos saber se o assistente estava funcionando de acordo com o que previámos nos testes iniciais.

Apesar do assistente de velocidade não ser uma tecnologia inédita, pois existem projetos de controle de velocidade em caminhões, pilotos automáticos em carros de passeio, tecnologias para evitar colisões, entre outras, o estudo de um assistente de velocidade desenvolvido em fuzzy é de extrema importância. Tal estudo abre uma gama de possibilidades para entender melhor como tais tecnologias funcionam e propõem novos estudos acadêmicos sobre o tema. Além disso, se pensarmos que o custo de veículos que possuem tecnologias inteligentes é geralmente mais caro, poderíamos pensar em formas mais baratas que sejam tão eficientes quanto as já comercializadas.

5.1 Trabalhos futuros

- Prever na placa de circuito impressa todos os pinos necessários para o funcionamento do Arduino Mega;
- Ajustar os valores dos intervalos das entradas de velocidade e distância, adicionando mais sobreposições para melhorar a resposta do assistente de velocidade e para que a aceleração ou desaceleração do robô autônomo ocorra de forma mais gradativa;
- Melhorar o hardware adicionando um sensor de velocidade para substituir o cálculo da velocidade média;

- Tentar outras formas de inteligência para compara-la com a lógica fuzzy, uma opção seria implementar uma rede neural por exemplo.

REFERÊNCIAS BIBLIOGRÁFICAS

ABAR, Celina. "**O Conceito Fuzzy**". Pontifícia Universidade Católica de São Paulo, 2004. Disponível em: https://www.pucsp.br/~logica/Fuzzy.htm. Acesso em 02 de abril de 2021.

About US. **MIT APP INVENTOR**, 2018. Disponível em: https://appinventor.mit.edu/about-us. Acesso em: 18 de novembro de 2021.

ARJAN**. Tutorial Arduino uno (Pinagem).** Diyi0t, 2020. Disponível em: https://diyi0t.com/arduino-uno-tutorial/. Acesso em: 23 de agosto de 2021.

AUTOESPORTE, Redação. **"Volvo emociona em novo comercial do XC60**", 2017. Disponível em: https://autoesporte.globo.com/carros/noticia/2017/07/volvo-emociona-em-novo-comercial-do-xc60.ghtml. Acesso em: 02 de abril de 2021.

BRAGA, Newton C. "**Como funcionam os sensores Ultrassônicos (ART691)**", 2012 "disponível em: https://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691. Acesso em: 15 de agosto de 2021.

BRAGA, N. C. ***Projetos com Caixas de Redução (MEC162)***, 2016. Disponível em: https://www.newtoncbraga.com.br/index.php/robotica/9505-projetos-com-caixas-de-reducao-mec162. Acesso em: 12 de agosto de 2021.

BRAYER, Jorge. **Veículos autônomos: estudo do desenvolvimento com viabilidade econômica e inteligência artificial**. Brasil: Ed. Autor, 2019.

BROGGI, A. Vision-based driving assistance. IEEE Intelligent Systems, v. 13, n. 6, p. 22–23, 1998.

CHAVES, Alaor. **Física Básica – Mecânica "**disponível em: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1932-1. Acesso em: 06 de junho de 2022.

DORF, Richard C.; BISHOP, Robert H. **Sistemas de controle modernos**. Rio de Janeiro: LTC –Livros Técnicos e Científicos Editora, 2001.

DIYI0T. **Tutorial Arduino mega (Pinagem**), (2020), "disponível em: https://diyi0t.com/arduino-mega-tutorial/, Acesso em: 04 de junho, 2022".

GAFA, Carmel***. Fuzzy Inference System implementation in Python,*** 2020. Disponível em: https://towardsdatascience.com/fuzzy-inference-system-implementation-in-python-8af88d1f0a6e. Acesso em: 19 de agosto de 2021.

HALLIDAY, David; RESNICK, Robert; WALKER, Jearl. Fundamentos de física. vol 2, 10. ed. Rio de Janeiro, RJ: LTC, 2003.

IVANQUI. JOSMAR. "**Esteira Eletrônica com Velocidade Controlada por Lógica *Fuzzy*",** Dissertação de mestre em ciências, 2005.

KLIR, G.J. E YUAN, B. (1995). *Fuzzy Sets and Fuzzy Logic Theory and Applications, Prentice-Hall*.

O que é Arduino**. Arduino.cc,** 2018. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 18 de agosto de 2021.

PATSKO. Luís Fernando. “Tutorial Aplicações, Funcionamento e Utilização de Sensores”, 2006.

PISSARDINI, R. de S.; WEI, D. C. M.; JÚNIOR, E. S. da F. **Veículos autônomos: Conceitos**, histórico e estado-da-arte. 2013.

RAIZER, Klaus**. “*Braitenberg Vehicles*: Revisão e Aplicações”. Inteligência Artificial,** 2009.

ROBÔ SEGUIDOR DE LINHA OU ROBÔ SEGUE-FAIXA. **Robocore, 2017.**Disponível em: https://www.robocore.net/tutoriais/robo-seguidor-de-linha. Acesso em: 18 de agosto de 2021.

SANTOS, W.E. D. Jr., J.H.C. G. Robótica Industrial - Fundamentos, Tecnologias, Programação e Simulação. [Digite o Local da Editora]: Editora Saraiva, 2015. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536520254/. Acesso em: 04 de agosto de 2021 as 20:10.

SECCHI, H. A. **Uma Introdução aos Robôs Móveis. [S.I.**]: IFES – Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, 2012. 19, 25.

SIMÕES, Marcelo Godoy; Shaw, Ian S. **Controle e Modelagem Fuzzy**, Editora Blucher, 2007.

SOFTWARE ARDUINO (IDE). **Arduino.cc, 2017.** Disponível em: https://docs.arduino.cc/foundations/basics/softwares. Acesso em: 19 de agosto de 2021.

SOUZA, Fábio. **Introdução à Arduino MEGA 2560**, 2014. Disponível em: < https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 15 de abril de 2022.

SOUZA, Lucas Danrley Cajé. **Estruturas Robóticas Simples: Aplicações e Comportamentos com Base nos Veículos de Braitenberg.** Disponível em: https://sites.google.com/a/ee.ufcg.edu.br/jornalpet/materias/ed84art3?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1. Acesso em: 15 de maio de 2021.

TANSCHEIT, Ricardo. **SYSTEMAS FUZZY**, 2020. Disponível em:http://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/LN-Sistemas%20Fuzzy.pdf. Acesso em: 18 de fevereiro de 2022.

TEIXEIRA, Ricardo. **Visão Geral sobre o Sonar HC-SR04**, 2020. Disponível em: https://www.ricardoteix.com/visao-geral-sobre-o-sonar-hc-sr04/. Acesso em: 13 de agosto de 2021.

THOMAZINI, Daniel; ALBUQUERQUE, P.U.B.; Sensores **Industriais – Fundamentos e Aplicações**. Editora Érica, 2009**.**

THOMAZINI, D.; Albuquerque, P.U.B. D. Sensores Industriais - Fundamentos e Aplicações. [Digite o Local da Editora]: Editora Saraiva, 2011. 9788536520261 “disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536520261/, acesso em: 16 de agosto de 2021.

THOMSEN, Adilson. Tutorial Módulo Bluetooth com Arduino, 2015. Disponível em: Tutorial Módulo Bluetooth com Arduino - FilipeFlop. Acesso em: 16 de agosto de 2021.

UNO R3 – Pinagem**. Arduino.cc, 2019.** Disponível em: https://docs.arduino.cc/hardware/uno-rev3 acesso em: 18 de agosto de 2021.

VIDAL, Vitor. **Sensor Ultrassônico HC-SR04 com Arduino**, Blog Eletrogate, 2022, “disponível em: https://blog.eletrogate.com/sensor-ultrassonico-hc-sr04-com-arduino/” “Acesso em: 30 de janeiro de 2022 as 19:40”.

ZADEH. L. A. *Fuzzy sets. Fuzzy Sets, Information and Control*, 8:338 – 353 1965.

WOLLINGER. Leonardo. Baterias de Lítio-Íon: Um guia complete. “disponível em:https://www.embarcados.com.br/baterias-de-litio-ion-um-guia-completo/, acesso: 22 março de 2022 as 00:12.

APÊNDICE A - Código completo do Assistente de Velocidade

Fuzzy30Regras_09_06 | Arduino 1.8.14

Arquivo Editar Sketch Ferramentas Ajuda

Fuzzy30Regras_09_06

```
#include <Fuzzy.h>
//Include the SoftwareSerial library
#include "SoftwareSerial.h"


//Cria um novo software  serial
SoftwareSerial bluetooth(50, 52); //TX, RX (Bluetooth)


int velFuzzy = 0;
int velocidade = 0;
int velBlue = 0;
int velPadrao = 110;
int velCurva = 80;
int estado = 0; // 1 = bluetooth, 2 - fuzzy, 3 - curva
boolean aux = false;
boolean auxLed = false;
int manual = 0;
int auxManual = 0;
String texto = "";
int ledPin = 46;
double distInicial = 0;
double distFinal = 0;
unsigned long tempoInicial = 0;
unsigned long tempoFinal = 0;
unsigned long tempo = 0;
double velcar = 0;
int inicio = 0;
int inputl = 0;
```

```
30
31 class Motor {
32     int vel = 255, Ml, DIRl;
33
34   public:
35
36     void Pinout(int inl, int in2) { // Pinos para controle do motor
37       Ml = inl;
38       DIRl = in2;
39       pinMode(Ml, OUTPUT);
40       pinMode(DIRl, OUTPUT);
41     }
42     void Velocidade(int vell) { // Método Velocidade, salva a velocidade do motor
43       vel = vell;
44     }
45     void Andar() { // Método Andar para fazer o motor girar para frente
46       analogWrite(Ml, vel);
47       digitalWrite(DIRl, LOW);
48     }
49     void Parar() { // Método Parar para fazer o motor ficar parado.
50       digitalWrite(Ml, LOW);
51       digitalWrite(Ml, LOW);
52     }
53 };
54
55 Motor Motorl, Motor2; // Objetos para os dois motores utilizados
56
56
57 //Sensores de linha
58 class Sensor {
59     int pin_Sl, pinl;
60   public:
61
62     void Pinout(int pinl) {
63       pin_Sl = pinl;
64       pinMode(pin_Sl, INPUT);
65     }
66     int Ler() { // Ler sensor
67       int sensor = digitalRead(pin_Sl);
68       return sensor;
69     }
70
71 };
72
```

```
75 class SensorDist {
76     int echoPin, trigPin, distance;
77     long tempo;
78 
79   public:
80 
81     void Pinout(int pin1, int pin2) {
82       echoPin = pin1;
83       trigPin = pin2;
84       pinMode(trigPin, OUTPUT); // Seta o trigPin como OUTPUT
85       pinMode(echoPin, INPUT); // Seta o echoPin como INPUT
86     }
87     long Distancia() { //Ler distância
88       // Limpa o trigPin
89       digitalWrite(trigPin, LOW);
90       delayMicroseconds(10);
91       // Seta o trigPin HIGH por 10 microsegundos
92       digitalWrite(trigPin, HIGH);
93       delayMicroseconds(10);
94       digitalWrite(trigPin, LOW);
95       // Lê o echoPin, retorno do tempo da onda sonora em microsegundos
96       tempo = pulseIn(echoPin, HIGH);
97       // Calcula a distância
98       distance = tempo * 0.034 / 2; // Velocidade da onda sonora divida por 2, ida e volta
99 
100      return distance;
101    }
102 
103 };
104 
105 SensorDist SensorDist1, SensorDist2, SensorDist3; //Objetos sensor de distância
106 
```

```
108
109 // Instantiating a Fuzzy object
110 Fuzzy *fuzzy = new Fuzzy();
111
112
113 void setup()
114 {
115
116   //Configurando os pinos dos motores e sensores
117   Motor1.Pinout(5, 4); // Pinos motor 1
118   Motor2.Pinout(6, 7); // Pinos motor 2
119   Sensor1.Pinout(A0); // Pino sensor 1
120   Sensor2.Pinout(A1); // Pino sensor 2
121   Sensor3.Pinout(A2); // Pino sensor 3
122   Sensor4.Pinout(A3); // Pino sensor 4
123   Sensor5.Pinout(A4); // Pino sensor 5
124
125   SensorDist1.Pinout(9, 12); // Pino sensor de distância central
126
127   pinMode(ledPin, OUTPUT);
128
129
130   // FuzzyInput
131   FuzzyInput *Distancia = new FuzzyInput(1);
132
133   FuzzySet *MuitoPerto = new FuzzySet(0, 5, 5, 10);
134   Distancia->addFuzzySet(MuitoPerto);
135   FuzzySet *Perto = new FuzzySet(9, 20, 20, 30);
136   Distancia->addFuzzySet(Perto);
137   FuzzySet *Intermediario = new FuzzySet(29, 35, 35, 40);
138   Distancia->addFuzzySet(Intermediario);
139   FuzzySet *Longe = new FuzzySet(39, 70, 70, 100);
140   Distancia->addFuzzySet(Longe);
141   FuzzySet *MuitoLonge = new FuzzySet(99, 150, 150, 200);
142   Distancia->addFuzzySet(MuitoLonge);
143   FuzzySet *Distante = new FuzzySet(199, 300, 300, 400.1);
144   Distancia->addFuzzySet(Distante);
```

```
130  // FuzzyInput
131  FuzzyInput *Distancia = new FuzzyInput(1);
132
133  FuzzySet *MuitoPerto = new FuzzySet(0, 5, 5, 10);
134  Distancia->addFuzzySet(MuitoPerto);
135  FuzzySet *Perto = new FuzzySet(9, 20, 20, 30);
136  Distancia->addFuzzySet(Perto);
137  FuzzySet *Intermediario = new FuzzySet(29, 35, 35, 40);
138  Distancia->addFuzzySet(Intermediario);
139  FuzzySet *Longe = new FuzzySet(39, 70, 70, 100);
140  Distancia->addFuzzySet(Longe);
141  FuzzySet *MuitoLonge = new FuzzySet(99, 150, 150, 200);
142  Distancia->addFuzzySet(MuitoLonge);
143  FuzzySet *Distante = new FuzzySet(199, 300, 300, 400.1);
144  Distancia->addFuzzySet(Distante);
145
146  fuzzy->addFuzzyInput(Distancia);
147
148  // FuzzyInput
149  FuzzyInput *VInInput = new FuzzyInput(2);
150
151  FuzzySet *VInMuitoBaixa = new FuzzySet(60, 65, 65, 70);
152  VInInput->addFuzzySet(VInMuitoBaixa);
153  FuzzySet *VInBaixa = new FuzzySet(69, 75, 75, 80);
154  VInInput->addFuzzySet(VInBaixa);
155  FuzzySet *VInMedia = new FuzzySet(79, 85, 85, 90);
156  VInInput->addFuzzySet(VInMedia);
157  FuzzySet *VInAlta = new FuzzySet(89, 95, 95, 100);
158  VInInput->addFuzzySet(VInAlta);
159  FuzzySet *VInMuitoAlta = new FuzzySet(99, 130, 130, 160.1);
160  VInInput->addFuzzySet(VInMuitoAlta);
161
162
163  fuzzy->addFuzzyInput(VInInput);
164
```

```
163  fuzzy->addFuzzyInput(VInInput);
164
165  // FuzzyOutput
166  FuzzyOutput *VelCar01 = new FuzzyOutput(1);
167
168  FuzzySet *VelCar01MuitoBaixa = new FuzzySet(60, 65, 65, 70);
169  VelCar01->addFuzzySet(VelCar01MuitoBaixa);
170  FuzzySet *VelCar01Baixa = new FuzzySet(69, 75, 75, 80);
171  VelCar01->addFuzzySet(VelCar01Baixa);
172  FuzzySet *VelCar01Media = new FuzzySet(79, 95, 95, 100);
173  VelCar01->addFuzzySet(VelCar01Media);
174  FuzzySet *VelCar01Alta = new FuzzySet(99, 110, 110, 160.1);
175  VelCar01->addFuzzySet(VelCar01Alta);
176
177  fuzzy->addFuzzyOutput(VelCar01);
178
179  // Construindo regra 01
180  FuzzyRuleAntecedent *ifMuitoPertoAndVInMuitoBaixa = new FuzzyRuleAntecedent();
181  ifMuitoPertoAndVInMuitoBaixa->joinWithAND(MuitoPerto, VInMuitoBaixa);
182  FuzzyRuleConsequent *thenVelCar01MuitoBaixa1 = new FuzzyRuleConsequent();
183  thenVelCar01MuitoBaixa1->addOutput(VelCar01MuitoBaixa);
184  FuzzyRule *fuzzyRule1 = new FuzzyRule(1, ifMuitoPertoAndVInMuitoBaixa, thenVelCar01MuitoBaixa1);
185  fuzzy->addFuzzyRule(fuzzyRule1);
186
187  // Construindo regra 02
188  FuzzyRuleAntecedent *ifPertoAndVInMuitoBaixa = new FuzzyRuleAntecedent();
189  ifPertoAndVInMuitoBaixa->joinWithAND(Perto, VInMuitoBaixa);
190  FuzzyRuleConsequent *thenVelCar01MuitoBaixa2 = new FuzzyRuleConsequent();
191  thenVelCar01MuitoBaixa2->addOutput(VelCar01MuitoBaixa);
192  FuzzyRule *fuzzyRule2 = new FuzzyRule(2, ifPertoAndVInMuitoBaixa, thenVelCar01MuitoBaixa2);
193  fuzzy->addFuzzyRule(fuzzyRule2);
194
195  // Construindo regra 03
196  FuzzyRuleAntecedent *ifIntermediarioAndVInMuitoBaixa = new FuzzyRuleAntecedent();
197  ifIntermediarioAndVInMuitoBaixa->joinWithAND(Intermediario, VInMuitoBaixa);
198  FuzzyRuleConsequent *thenVelCar01Baixa3 = new FuzzyRuleConsequent();
199  thenVelCar01Baixa3->addOutput(VelCar01Baixa);
200  FuzzyRule *fuzzyRule3 = new FuzzyRule(3, ifIntermediarioAndVInMuitoBaixa, thenVelCar01Baixa3);
201  fuzzy->addFuzzyRule(fuzzyRule3);
```

```
203  // Construindo regra 04
204  FuzzyRuleAntecedent *ifLongeAndVInMedia = new FuzzyRuleAntecedent();
205  ifLongeAndVInMedia->joinWithAND(Longe, VInMedia);
206  FuzzyRuleConsequent *thenVelCar01Media4 = new FuzzyRuleConsequent();
207  thenVelCar01Media4->addOutput(VelCar01Media);
208  FuzzyRule *fuzzyRule4 = new FuzzyRule(4, ifLongeAndVInMedia, thenVelCar01Media4);
209  fuzzy->addFuzzyRule(fuzzyRule4);
210
211  // Construindo regra 05
212  FuzzyRuleAntecedent *ifMuitoLongeAndVInMuitoBaixa = new FuzzyRuleAntecedent();
213  ifMuitoLongeAndVInMuitoBaixa->joinWithAND(MuitoLonge, VInMuitoBaixa);
214  FuzzyRuleConsequent *thenVelCar01MuitoAlta5 = new FuzzyRuleConsequent();
215  thenVelCar01MuitoAlta5->addOutput(VelCar01Alta);
216  FuzzyRule *fuzzyRule5 = new FuzzyRule(5, ifMuitoLongeAndVInMuitoBaixa, thenVelCar01MuitoAlta5);
217  fuzzy->addFuzzyRule(fuzzyRule5);
218  //
219  // Construindo regra 06
220  FuzzyRuleAntecedent *ifDistanteAndVInMuitoBaixa = new FuzzyRuleAntecedent();
221  ifDistanteAndVInMuitoBaixa->joinWithAND(Distante, VInMuitoBaixa);
222  FuzzyRuleConsequent *thenVelCar01Alta6 = new FuzzyRuleConsequent();
223  thenVelCar01Alta6->addOutput(VelCar01Alta);
224  FuzzyRule *fuzzyRule6 = new FuzzyRule(6, ifDistanteAndVInMuitoBaixa, thenVelCar01Alta6);
225  fuzzy->addFuzzyRule(fuzzyRule6);
226
227  // Construindo regra 07
228  FuzzyRuleAntecedent *ifMuitoPertoAndVInBaixa = new FuzzyRuleAntecedent();
229  ifMuitoPertoAndVInBaixa->joinWithAND(MuitoPerto, VInBaixa);
230  FuzzyRuleConsequent *thenVelCar01Baixa7 = new FuzzyRuleConsequent();
231  thenVelCar01Baixa7->addOutput(VelCar01Baixa);
232  FuzzyRule *fuzzyRule7 = new FuzzyRule(7, ifMuitoPertoAndVInBaixa, thenVelCar01Baixa7);
233  fuzzy->addFuzzyRule(fuzzyRule7);
234
```

```
235  // Construindo regra 08
236  FuzzyRuleAntecedent *ifPertoAndVInBaixa = new FuzzyRuleAntecedent();
237  ifPertoAndVInBaixa->joinWithAND(Perto, VInBaixa);
238  FuzzyRuleConsequent *thenVelCar01Media8 = new FuzzyRuleConsequent();
239  thenVelCar01Media8->addOutput(VelCar01Media);
240  FuzzyRule *fuzzyRule8 = new FuzzyRule(8, ifPertoAndVInBaixa, thenVelCar01Media8);
241  fuzzy->addFuzzyRule(fuzzyRule8);
242
243  // Construindo regra 09
244  FuzzyRuleAntecedent *ifIntermediarioAndVInBaixa = new FuzzyRuleAntecedent();
245  ifIntermediarioAndVInBaixa->joinWithAND(Intermediario, VInBaixa);
246  FuzzyRuleConsequent *thenVelCar01Media9 = new FuzzyRuleConsequent();
247  thenVelCar01Media9->addOutput(VelCar01Media);
248  FuzzyRule *fuzzyRule9 = new FuzzyRule(9, ifIntermediarioAndVInBaixa, thenVelCar01Media9);
249  fuzzy->addFuzzyRule(fuzzyRule9);
250
251  // Construindo regra 10
252  FuzzyRuleAntecedent *ifLongeAndVInBaixa = new FuzzyRuleAntecedent();
253  ifLongeAndVInBaixa->joinWithAND(Longe, VInBaixa);
254  FuzzyRuleConsequent *thenVelCar01Media10 = new FuzzyRuleConsequent();
255  thenVelCar01Media10->addOutput(VelCar01Media);
256  FuzzyRule *fuzzyRule10 = new FuzzyRule(10, ifLongeAndVInBaixa, thenVelCar01Media10);
257  fuzzy->addFuzzyRule(fuzzyRule10);
258
259  // Construindo regra 11
260  FuzzyRuleAntecedent *ifMuitoLongeAndVInBaixa = new FuzzyRuleAntecedent();
261  ifMuitoLongeAndVInBaixa->joinWithAND(MuitoLonge, VInBaixa);
262  FuzzyRuleConsequent *thenVelCar01Alta11 = new FuzzyRuleConsequent();
263  thenVelCar01Alta11->addOutput(VelCar01Alta);
264  FuzzyRule *fuzzyRule11 = new FuzzyRule(11, ifMuitoLongeAndVInBaixa, thenVelCar01Alta11);
265  fuzzy->addFuzzyRule(fuzzyRule11);
266
267  // Construindo regra 12
268  FuzzyRuleAntecedent *ifDistanteAndVInBaixa = new FuzzyRuleAntecedent();
269  ifDistanteAndVInBaixa->joinWithAND(Distante, VInBaixa);
270  FuzzyRuleConsequent *thenVelCar01Alta12 = new FuzzyRuleConsequent();
271  thenVelCar01Alta12->addOutput(VelCar01Alta);
272  FuzzyRule *fuzzyRule12 = new FuzzyRule(12, ifDistanteAndVInBaixa, thenVelCar01Alta12);
273  fuzzy->addFuzzyRule(fuzzyRule12);
```

```
275  // Construindo regra 13
276  FuzzyRuleAntecedent *ifMuitoPertoAndVInVInMedia = new FuzzyRuleAntecedent();
277  ifMuitoPertoAndVInVInMedia->joinWithAND(MuitoPerto, VInMedia);
278  FuzzyRuleConsequent *thenVelCar01Media13 = new FuzzyRuleConsequent();
279  thenVelCar01Media13->addOutput(VelCar01Media);
280  FuzzyRule *fuzzyRule13 = new FuzzyRule(13, ifMuitoPertoAndVInVInMedia, thenVelCar01Media13);
281  fuzzy->addFuzzyRule(fuzzyRule13);
282
283  // Construindo regra 14
284  FuzzyRuleAntecedent *ifPertoAndVInMedia = new FuzzyRuleAntecedent();
285  ifPertoAndVInMedia->joinWithAND(Perto, VInMedia);
286  FuzzyRuleConsequent *thenVelCar01Media14 = new FuzzyRuleConsequent();
287  thenVelCar01Media14->addOutput(VelCar01Media);
288  FuzzyRule *fuzzyRule14 = new FuzzyRule(14, ifPertoAndVInMedia, thenVelCar01Media14);
289  fuzzy->addFuzzyRule(fuzzyRule14);
290
291  // Construindo regra 15
292  FuzzyRuleAntecedent *ifIntermediarioAndVInMedia = new FuzzyRuleAntecedent();
293  ifIntermediarioAndVInMedia->joinWithAND(Intermediario, VInMedia);
294  FuzzyRuleConsequent *thenVelCar01Media15 = new FuzzyRuleConsequent();
295  thenVelCar01Media15->addOutput(VelCar01Media);
296  FuzzyRule *fuzzyRule15 = new FuzzyRule(15, ifIntermediarioAndVInMedia, thenVelCar01Media15);
297  fuzzy->addFuzzyRule(fuzzyRule15);
298
299  // Construindo regra 16
300  FuzzyRuleAntecedent *ifLongeHighAndVInMedia = new FuzzyRuleAntecedent();
301  ifLongeHighAndVInMedia->joinWithAND(Longe, VInMedia);
302  FuzzyRuleConsequent *thenVelCar01Alta16 = new FuzzyRuleConsequent();
303  thenVelCar01Alta16->addOutput(VelCar01Alta);
304  FuzzyRule *fuzzyRule16 = new FuzzyRule(16, ifLongeHighAndVInMedia, thenVelCar01Alta16);
305  fuzzy->addFuzzyRule(fuzzyRule16);
306
307  // Construindo regra 17
308  FuzzyRuleAntecedent *ifMuitoLongeAndVInMedia = new FuzzyRuleAntecedent();
309  ifMuitoLongeAndVInMedia->joinWithAND(MuitoLonge, VInMedia);
310  FuzzyRuleConsequent *thenVelCar01Alta17 = new FuzzyRuleConsequent();
311  thenVelCar01Alta17->addOutput(VelCar01Alta);
312  FuzzyRule *fuzzyRule17 = new FuzzyRule(17, ifMuitoLongeAndVInMedia, thenVelCar01Alta17);
313  fuzzy->addFuzzyRule(fuzzyRule17);
```

```
315  // Construindo regra 18
316  FuzzyRuleAntecedent *ifDistanteAndVInMedia = new FuzzyRuleAntecedent();
317  ifDistanteAndVInMedia->joinWithAND(Distante, VInMedia);
318  FuzzyRuleConsequent *VelCar01Alta18 = new FuzzyRuleConsequent();
319  VelCar01Alta18->addOutput(VelCar01Alta);
320  FuzzyRule *fuzzyRule18 = new FuzzyRule(18, ifDistanteAndVInMedia, VelCar01Alta18);
321  fuzzy->addFuzzyRule(fuzzyRule18);
322
323  // Construindo regra 19
324  FuzzyRuleAntecedent *ifMuitoPertoAndVInAlta = new FuzzyRuleAntecedent();
325  ifMuitoPertoAndVInAlta->joinWithAND(MuitoPerto, VInAlta);
326  FuzzyRuleConsequent *thenVelCar01Baixa19 = new FuzzyRuleConsequent();
327  thenVelCar01Baixa19->addOutput(VelCar01Baixa);
328  FuzzyRule *fuzzyRule19 = new FuzzyRule(19, ifMuitoPertoAndVInAlta, thenVelCar01Baixa19);
329  fuzzy->addFuzzyRule(fuzzyRule19);
330
331  // Construindo regra 20
332  FuzzyRuleAntecedent *ifPertoAndVInAlta = new FuzzyRuleAntecedent();
333  ifPertoAndVInAlta->joinWithAND(Perto, VInAlta);
334  FuzzyRuleConsequent *thenVelCar01Baixa20 = new FuzzyRuleConsequent();
335  thenVelCar01Baixa20->addOutput(VelCar01Baixa);
336  FuzzyRule *fuzzyRule20 = new FuzzyRule(20, ifPertoAndVInAlta, thenVelCar01Baixa20);
337  fuzzy->addFuzzyRule(fuzzyRule20);
338
339  // Construindo regra 21
340  FuzzyRuleAntecedent *ifIntermediarioAndVInAlta = new FuzzyRuleAntecedent();
341  ifIntermediarioAndVInAlta->joinWithAND(Intermediario, VInAlta);
342  FuzzyRuleConsequent *thenVelCar01Media21 = new FuzzyRuleConsequent();
343  thenVelCar01Media21->addOutput(VelCar01Media);
344  FuzzyRule *fuzzyRule21 = new FuzzyRule(21, ifIntermediarioAndVInAlta, thenVelCar01Media21);
345  fuzzy->addFuzzyRule(fuzzyRule21);
346
347  // Construindo regra 22
348  FuzzyRuleAntecedent *ifLongeAndVInAlta = new FuzzyRuleAntecedent();
349  ifLongeAndVInAlta->joinWithAND(Longe, VInAlta);
350  FuzzyRuleConsequent *thenVelCar01Media22 = new FuzzyRuleConsequent();
351  thenVelCar01Media22->addOutput(VelCar01Media);
352  FuzzyRule *fuzzyRule22 = new FuzzyRule(22, ifLongeAndVInAlta, thenVelCar01Media22);
353  fuzzy->addFuzzyRule(fuzzyRule22);
```

```
// Construindo regra 23
FuzzyRuleAntecedent *ifMuitoLongeAndVInAlta = new FuzzyRuleAntecedent();
ifMuitoLongeAndVInAlta->joinWithAND(MuitoLonge, VInAlta);
FuzzyRuleConsequent *thenVelCar01Alta23 = new FuzzyRuleConsequent();
thenVelCar01Alta23->addOutput(VelCar01Alta);
FuzzyRule *fuzzyRule23 = new FuzzyRule(23, ifMuitoLongeAndVInAlta, thenVelCar01Alta23);
fuzzy->addFuzzyRule(fuzzyRule23);

// Construindo regra 24
FuzzyRuleAntecedent *ifDistanteAndVInAlta = new FuzzyRuleAntecedent();
ifDistanteAndVInAlta->joinWithAND(Distante, VInAlta);
FuzzyRuleConsequent *thenVelCar01Alta24 = new FuzzyRuleConsequent();
thenVelCar01Alta24->addOutput(VelCar01Alta);
FuzzyRule *fuzzyRule24 = new FuzzyRule(24, ifDistanteAndVInAlta, thenVelCar01Alta24);
fuzzy->addFuzzyRule(fuzzyRule24);

// Construindo regra 25
FuzzyRuleAntecedent *ifMuitoPertoAndVInMuitoAlta = new FuzzyRuleAntecedent();
ifMuitoPertoAndVInMuitoAlta->joinWithAND(MuitoPerto, VInMuitoAlta);
FuzzyRuleConsequent *thenVelCar01Baixa25 = new FuzzyRuleConsequent();
thenVelCar01Baixa25->addOutput(VelCar01Baixa);
FuzzyRule *fuzzyRule25 = new FuzzyRule(25, ifMuitoPertoAndVInMuitoAlta, thenVelCar01Baixa25);
fuzzy->addFuzzyRule(fuzzyRule25);

// Construindo regra 26
FuzzyRuleAntecedent *ifPertoAndVInMuitoAlta = new FuzzyRuleAntecedent();
ifPertoAndVInMuitoAlta->joinWithAND(Perto, VInMuitoAlta);
FuzzyRuleConsequent *thenVelCar01Baixa26 = new FuzzyRuleConsequent();
thenVelCar01Baixa26->addOutput(VelCar01Baixa);
FuzzyRule *fuzzyRule26 = new FuzzyRule(26, ifPertoAndVInMuitoAlta, thenVelCar01Baixa26);
fuzzy->addFuzzyRule(fuzzyRule26);

// Construindo regra 27
FuzzyRuleAntecedent *ifIntermediarioAndVInMuitoAlta = new FuzzyRuleAntecedent();
ifIntermediarioAndVInMuitoAlta->joinWithAND(Intermediario, VInMuitoAlta);
FuzzyRuleConsequent *thenVelCar01Media27 = new FuzzyRuleConsequent();
thenVelCar01Media27->addOutput(VelCar01Media);
FuzzyRule *fuzzyRule27 = new FuzzyRule(27, ifIntermediarioAndVInMuitoAlta, thenVelCar01Media27);
fuzzy->addFuzzyRule(fuzzyRule27);
```

```
  // Construindo regra 28
  FuzzyRuleAntecedent *ifLongeAndVInMuitoAlta = new FuzzyRuleAntecedent();
  ifLongeAndVInMuitoAlta->joinWithAND(Longe, VInMuitoAlta);
  FuzzyRuleConsequent *thenVelCar01Media28 = new FuzzyRuleConsequent();
  thenVelCar01Media28->addOutput(VelCar01Media);
  FuzzyRule *fuzzyRule28 = new FuzzyRule(28, ifLongeAndVInMuitoAlta, thenVelCar01Media28);
  fuzzy->addFuzzyRule(fuzzyRule28);

  // Construindo regra 29
  FuzzyRuleAntecedent *ifMuitoLongeAndVInMuitoAlta = new FuzzyRuleAntecedent();
  ifMuitoLongeAndVInMuitoAlta->joinWithAND(MuitoLonge, VInMuitoAlta);
  FuzzyRuleConsequent *thenVelCar01Alta29 = new FuzzyRuleConsequent();
  thenVelCar01Alta29->addOutput(VelCar01Alta);
  FuzzyRule *fuzzyRule29 = new FuzzyRule(29, ifMuitoLongeAndVInMuitoAlta, thenVelCar01Alta29);
  fuzzy->addFuzzyRule(fuzzyRule29);

  // Construindo regra 30
  FuzzyRuleAntecedent *ifDistanteAndVInMuitoAlta = new FuzzyRuleAntecedent();
  ifDistanteAndVInMuitoAlta->joinWithAND(Distante, VInMuitoAlta);
  FuzzyRuleConsequent *thenVelCar01Alta30 = new FuzzyRuleConsequent();
  thenVelCar01Alta30->addOutput(VelCar01Alta);
  FuzzyRule *fuzzyRule30 = new FuzzyRule(30, ifDistanteAndVInMuitoAlta, thenVelCar01Alta30);
  fuzzy->addFuzzyRule(fuzzyRule30);

  //Iniciando o software serial
  Serial.begin(38400);

  bluetooth.begin(38400);

  //Iniciando variáveis com a velocidade padrão
  velFuzzy = velPadrao;
  velocidade = velPadrao;


}
```

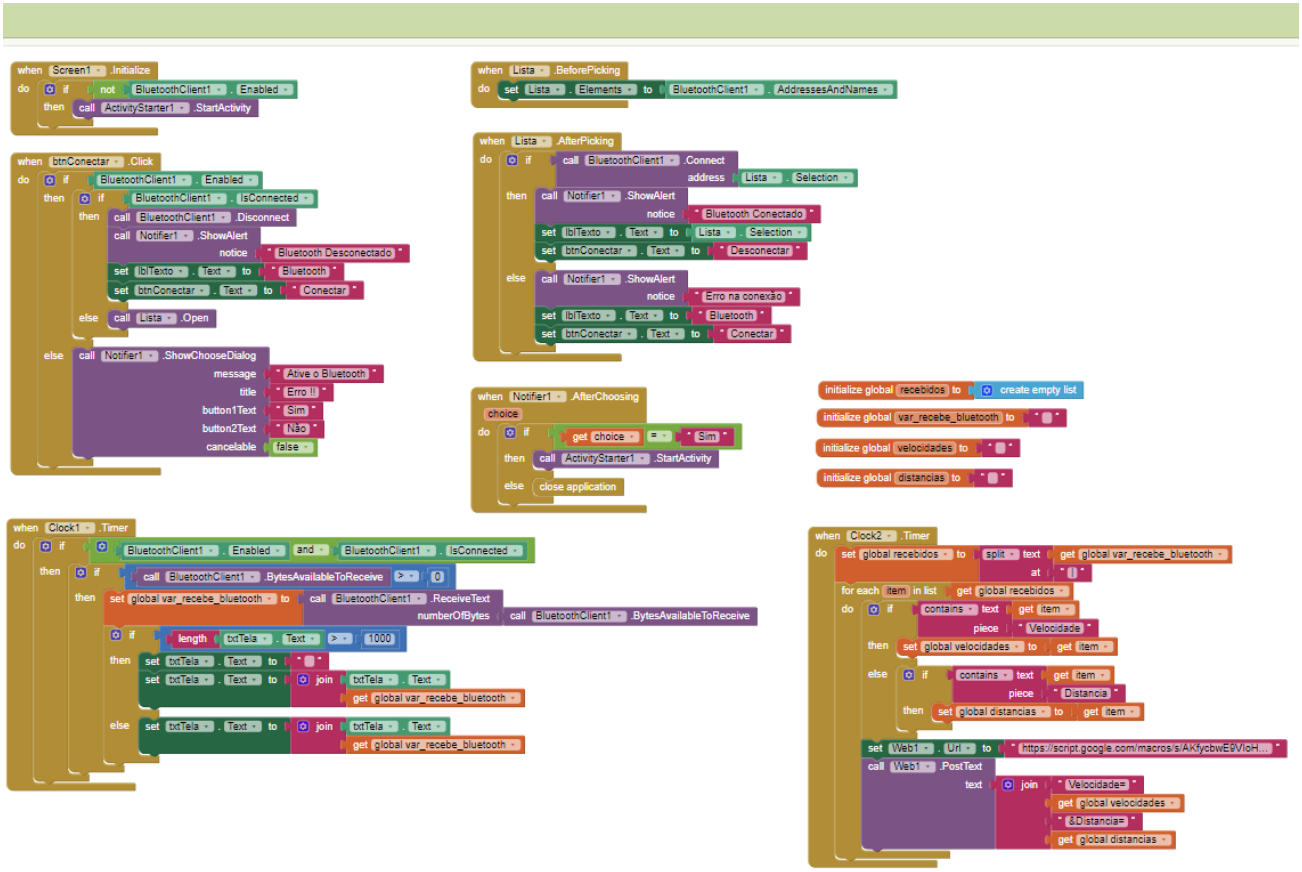
```
431 void loop()
432 {
433
434   //Leitura dos sensores
435   int dist = SensorDist1.Distancia();
436   int sns_1 = Sensor1.Ler();
437   int sns_2 = Sensor2.Ler();
438   int sns_3 = Sensor3.Ler();
439   int sns_4 = Sensor4.Ler();
440   int sns_5 = Sensor5.Ler();
441
442
443
444   //Cálculo da velocidade em cm/s
445   tempo = millis();
446
447   if (inicio == 0) {
448     distInicial = dist;
449     distFinal = dist;
450     tempoInicial = tempo;
451     tempoFinal = tempo;
452     tempoFinal = millis();
453     inicio = 1;
454   }
455
456   else {
457     tempoFinal = tempo;
458     if ((tempoFinal - tempoInicial) >= 3000) {
459       distFinal = dist;
460       double difTempo = ((tempoFinal - tempoInicial) / 3000);
461
462       velcar = abs((distFinal - distInicial) / difTempo);
463       //Conversão para valores entre 0 e 255
464       velcar = velcar * 3.5894;
465       inicio = 0;
466     }
467   }
```

```
//Recebendo valores via bluetooth
if (bluetooth.available() > 0) {
  char txt = bluetooth.read();
  //bluetooth.print(txt);
  if (txt == 'I') {
    if (velocidade < 140) {
      velocidade = velocidade + 1;
    }
  } else if (txt == 'D') {
    if (velocidade > 0) {
      velocidade = velocidade - 1;
    }
  } else if (txt == 'M') {
    manual = 1;
  } else if (txt == 'A') {
    manual = 0;
  } else {
    //Serial.print(txt);
    texto += txt; //Variável para armazenar todos os dados
    if (txt == '\n') { //Se o dado atual for um pulador de linha (\n)
      texto.trim(); //Remove o \n para comparar o texto
      int vel = texto.toInt();
      Serial.println(vel);
      if ((vel >= 10) && (vel <= 100)) {
        velocidade = (vel / 100.0) * velPadrao;
      }

      texto = ""; //Limpa o comando para futuras leituras

    }
  }
}

//Auxiliar automático/manual, comando recebido pelo aplicativo
if ((manual == 1) && (auxManual == 0)) {
  auxManual = 1;
} else if ((manual == 0) && (auxManual == 1))  {
  auxManual = 0;

```

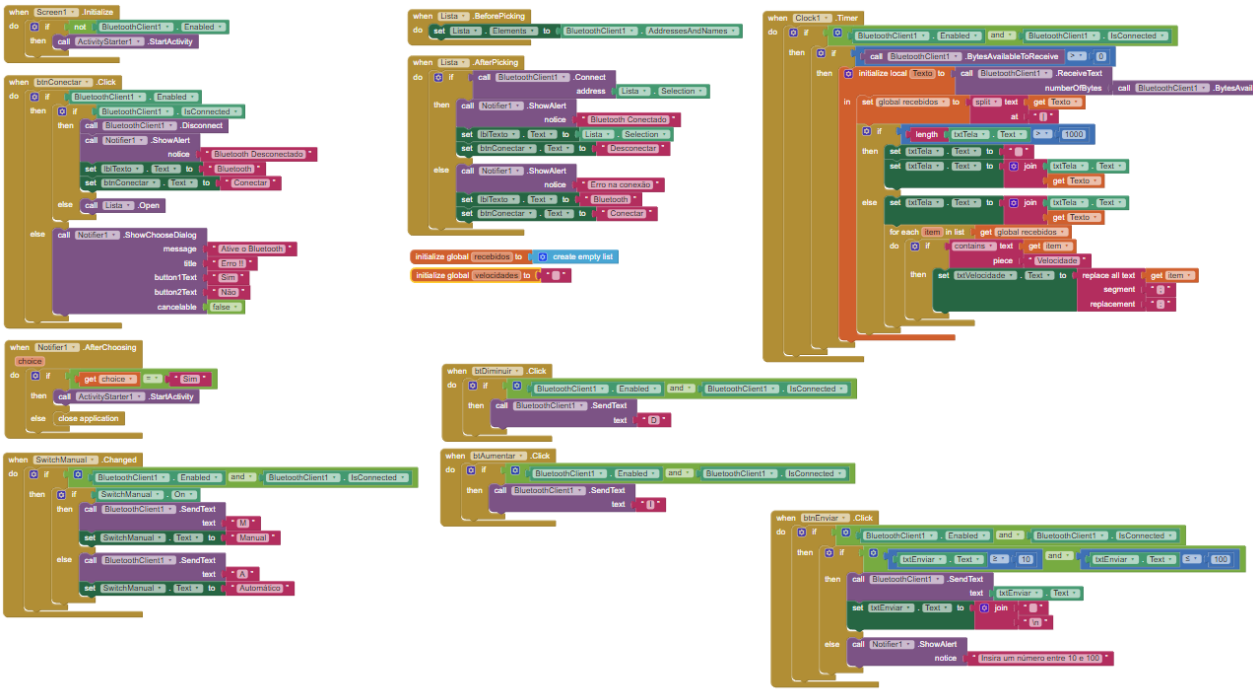
```
508   }
509
510   //Se a velocidade cálculada convertida em valores entre 60 e 160 envia como entrada para o fuzzy, caso contrário realimenta.
511   if ((velcar >= 60) && (velcar <= 160)) {
512     inputl = velcar;
513   } else {
514     inputl = velFuzzy;
515   }
516
517   //Se não estiver em manual, envia para o Fuzzy
518   if (auxManual == 0) {
519     int input = dist;
520
521     if (input < 100) {
522       fuzzy->setInput(1, input);
523       fuzzy->setInput(2, inputl);
524       //Running the Fuzzification
525       fuzzy->fuzzify();
526       //Running the Defuzzification
527       float output = fuzzy->defuzzify(1);
528       velFuzzy = output;
529       estado = 2;
530       auxLed = true;
531     }
532
533     //Auxiliar para entrada e saída da curva
534     if ((sns_1 == 1) && (sns_2 == 1) && (sns_3 == 1) && (sns_5 == 0)) {
535       aux = true;
536     } else if ((sns_1 == 0) && (sns_3 == 1) && (sns_4 == 1) && (sns_5 == 1)) {
537       aux = false;
538       digitalWrite(ledPin, HIGH);
539     }
540
541     if (aux == true) {
542       estado = 3;
543       // auxLed = false;
544       digitalWrite(ledPin, LOW);
545     }
548     //Dependendo do estado seta uma velocidade diferente
549     switch (estado) {
550       case 0:
551         velocidade = velPadrao;
552         break;
553       case 1:
554         velocidade = velBlue;
555         break;
556       case 2:
557         velocidade = velFuzzy;
558         break;
559       case 3:
560         velocidade = velCurva;
561         break;
562       default:
563         velocidade = velPadrao;
564         break;
565     }
566   }
567
568   //Envia a velocidade e distância via bluetooth, quando conectado salva na planilha do google
569   bluetooth.print("|Velocidade; ");
570   bluetooth.print(velocidade);
571   bluetooth.print("|\n");
572   bluetooth.print("|Distancia;");
573   bluetooth.print(dist);
574   bluetooth.print("|\n");
575
576   Motor1.Velocidade(velocidade); // A velocidade do motor pode variar de 0 a 255, onde 255 é a velocidade máxima.
577   Motor2.Velocidade(velocidade);
578
579   Motor1.Andar();
580   Motor2.Andar();
581
```

```
582   //Lógica para seguir linha
583   if ((dist > 15)) {
584     if ((sns_2 == 0) && (sns_4 == 0)) { // Se detectar na extremidade das faixas duas cores brancas
585       Motor1.Andar();
586       Motor2.Andar();
587     }
588     if ((sns_2 == 1) && (sns_4 == 0)) { // Se detectar um lado preto e o outro branco
589       Motor1.Parar();
590       Motor2.Andar();
591     }
592     if ((sns_2 == 0) && (sns_4 == 1)) { // Se detectar um lado branco e o outro preto
593       Motor1.Andar();
594       Motor2.Parar();
595     }
596   } else {
597     Motor1.Parar();
598     Motor2.Parar();
599   }
600 }
```

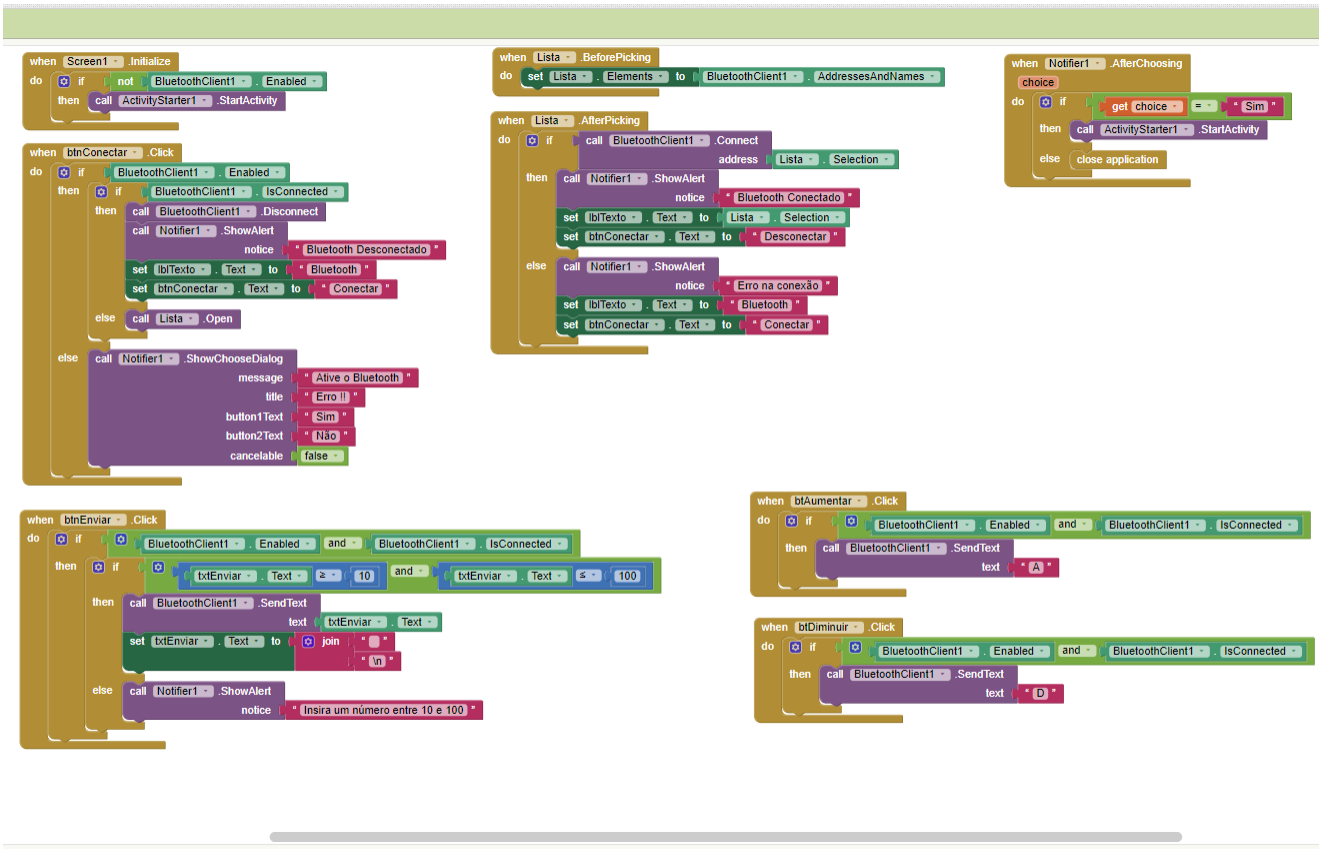
APÊNDICE B - Blocos MIT Inventor aplicativo de coleta



APÊNDICE C - Blocos MIT Inventor aplicativo controle RSL1



APÊNDICE D - Blocos MIT Inventor aplicativo controle RSL2



ANEXO A - Direitos autorais - Lei n. 9.610, de 19 de fevereiro de 1998

Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998[1].

Mensagem de veto

Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I - Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

Art. 5º Para os efeitos desta Lei, considera-se:

I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

VII - contrafação - a reprodução não autorizada;

VIII - obra:

a) em co-autoria - quando é criada em comum, por dois ou mais autores;

b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

c) pseudônima - quando o autor se oculta sob nome suposto;

d) inédita - a que não haja sido objeto de publicação;

e) póstuma - a que se publique após a morte do autor;

f) originária - a criação primígena;

g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado;

XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.

---

[1] Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l9610.htm.