# Relatório de Projeto - Robô Autônomo

Breno Augusto Mariano, Bruno Pena Modesto e Francis Carlos Santos

## I. Resumo

Este trabalho objetiva a concepção, construção e teste de um Robô na forma de um chassi para reconhecimento da localização de blocos e sua posterior movimentação para outra determinada região. O principal desafio deste projeto é implementar um software necessário para que um robô possa ser capaz de receber informações de sua localização e de obstáculos via *bluetooth* e desempenhar algumas tarefas pré-estabelecidas a partir das informações recebidas.

## II. Introdução

Durante a disciplina de Laboratório de Projetos 3 foi proposto a elaboração de um projeto para a criação de um robô que moveria alguns corpos (blocos) para regiões previamente especificadas. Para realizar tal missão, é necessário um software adequado para que o robô possa desempenhar essas atividades. O software desenvolvido possui uma parte destinada as configurações de comunicação fornecidas pelo professor, uma parte para teste da comunicação via *bluetooth* e uma parte destinada ao controle do robô durante seu deslocamento. Na seção Materiais e Métodos será mostrado os materiais que foram utilizados no projeto. Nas seções de Projeto da Parte de Software para Comunicação e Projeto da Parte de Software para Controle serão descritas e detalhadas as decisões de implementação que foram tomadas no projeto havendo uma discussão e justificativa sobre o porquê de cada escolha ter sido realizada. As motivações que iniciaram esse estudo sobre essa tecnologia de veículos autonômos são expostas a seguir. A principal vantagem dos carros autônomos é justamente de não cometerem os mesmos erros de um condutor humano, tais quais: condução egoísta, lentidão de reflexos, violação de normas de segurança, sono e embriaguez. Os carros autônomos podem inclusive superar o reflexo normal humano, prevendo prováveis acidentes em sua trajetória []c2. Plataformas de transporte de pessoas em veículos privados, como a *Uber*, não funcionam com um frota de veículos de forma contínua devido a limitação humana do condutor. Com veículos autônomos, uma menor frota de carros poderia atender de igual forma o mesmo número de usuários do aplicativo, sem que seja necessário ter gastos adicionais por trabalho noturno ou durante finais de semana e feriados. Além disso, outro ponto interessante ao usuário é a questão da privacidade, já que ao se retirar o condutor humano do veículo, o cliente poderá sentir-se mais confortável durante seu transporte. Há também o fator de melhoria de segurança do serviço, pois conforme testes realizados pelo *Google* em seus veículos autônomos []c1, o fator humano pode ser considerado como a principal causa de acidentes envolvendo carros autônomos; por exemplo, houveram acidentes principalmente por culpa de condutores humanos de outros veículos convencionais []c3.

## III. Materiais e Métodos

Todos os materiais utilizados ao longo do projeto foram listados abaixo. Lista de Materiais:

- 1 Protoboard;
- 2 conjuntos (Roda de Tração + Motor);
- 1 Módulo Receptor Bluetooth;
- 1 Arduíno Uno;
- 1 Roda Boba;
- 1 Ponte H;
- Palitos de madeira e cola;
- 1 Bateria 9V;
- 2 Leds vermelho;
- 2 Resistores de 150 ohm, 330 ohm e 660 ohm;
- Software de Programação Arduino;
- Aplicativo Android Bluetooth SPP Pro.

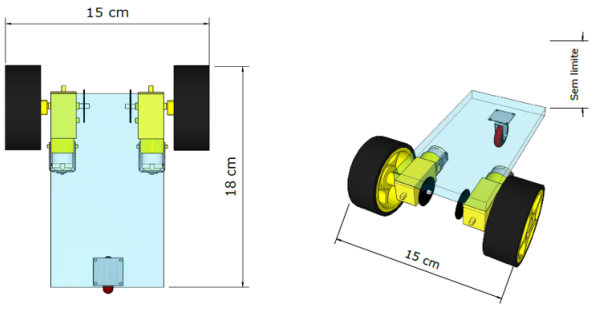


Figura 1: desenho inicial do robô e suas dimensões máximas permitidas

Especificação dos Motores e das Rodas:

- Tensão de Operação: 3-6 V;
- Taxa de Redução 48:1;
- Peso:30 gramas;
- Corrente sem carga: até 200 mA(6V) e até 150 mA(3V);
- Velocidade sem carga: 200 RPM (6V) e 90 RPM (3V);
- Diâmetro: 6.8 cm ;
- Largura: 2.6 cm;
- Furo central: 5.3 x 3.66 mm (semi-círculo);
- Peso:50 gramas.

## IV. Projeto Parte Mecânica e Eletrônica

Na figura Figura 2 há a estrutura principal do robô, feita de palitos de madeira. Há dois conjuntos roda e motor elétrico, além de uma roda boba na parte frontal do chassi dando ao robô estabilidade lateral. Na parte superior estão alocados os demais componentes especificados na lista de materiais. Eles serão responsáveis pelo controle do robô juntamente com a função de comunicação dos dados de operação do robô. Será utilizado um padrão de identificação via *Bluetooth* (*QR code*)

na parte superior dos blocos e do robô. Um sistema exterior e de responsabilidade do Cliente fará a leitura da localização atual do bloco e das coordenadas de sua alocação futura, transmitindo tais informações para o robô. O sistema embarcado do robô interpretará os dados recebidos e controlará o robô para realizar a movimentação. A explicação detalhada da função de cada componente está presente na seção de projeto mecânico e eletrônico desse relatório. A ideia inicial de desenho do robô foi baseada na proposta mostrada pelo Professor-Orientador, a qual encontra-se na figura Figura 1 , juntamente com suas dimensões máximas permitidas.

## V. Projeto da Parte Mecânica

O desenho do robô em uma vista isométrica para uma melhor perspectiva é mostrado na figura Figura 2, todos os desenhos foram feitos na plataforma *SketchUp*, para atender uma restrição de projeto de utilização de ferramentas e plataformas abertas: O objetivo principal que norteou o novo desenho do

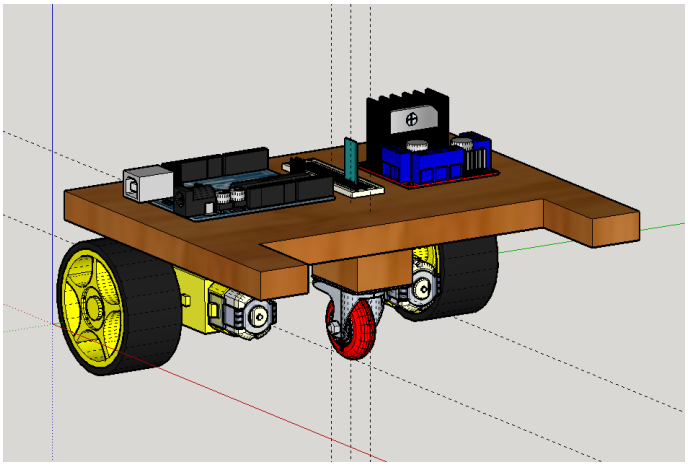

Figura 2: Vista isométrica do robô

chassi proposto foi a estabilização e posterior movimentação do bloco, para isso foi criada duas "garras" na parte frontal do chassi, entre as quais serão alocados os blocos e em seguida será iniciado a sua operação de movimentação. O tamanho das garras foi projetado para se manter o centro de massa, e consequentemente o centro geométrico ( considerando que o bloco é uniforme, esses dois pontos são coincidentes) do bloco estável durante o seu deslocamento. É preciso ressaltar que os fios elétricos irão contornar o chassi e serão fixados com adesivos para não haver qualquer tipo de deslocamento dos mesmos, podendo atrapalhar a dinâmica de movimentação do robô. A solução inicial de perfuração de um orifício no chassi principal do robô foi descatada tendo em vista a manutenção da estrutura inicial do chassi do robô, já que uma perfuração acarretaria no surgimento de pontos concentradores de tensão, os quais poderiam evoluir para trincas ou micro-rupturas, danificando a estrutura do chassi; além do fato de a solução adotada ser de implementação mais simples. Conforme é possível averiguar na Figura 1 que mostra os tamanhos máximos estabelecidos para o robô, somente duas dimensões foram limitadas. Assim, a terceira dimensão (altura do robô) foi utilizada para alocar todos os componentes. A figura Figura 3 com melhor detalhes a disposição das rodas do robô. Baseado na estratégia de utilização da dimensão não-

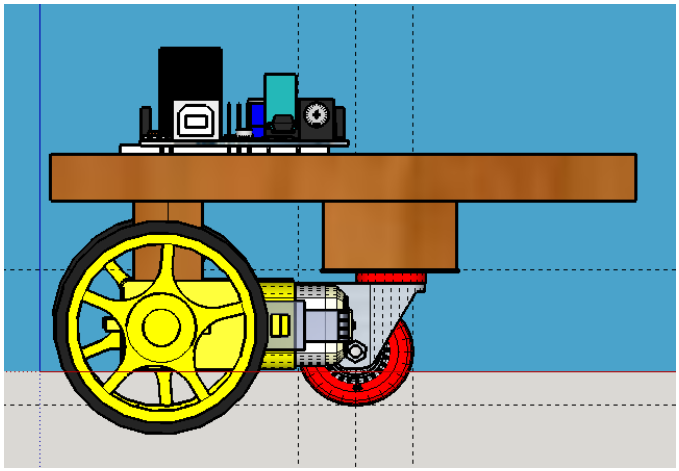

Figura 3: vista lateral do robô com destaque para as rodas

limitada do robô, optou-se pela colocação de suportes fixados nos motores para aumentar a altura do chassi principal em relação às rodas de tração (rodas amarelas), impedindo que haja contato físico entre o chassi e essas rodas, o que poderia danificar o chassi, prejudicar a rotação das rodas e impedir o funcionamento correto do robô. Foi adicionado também um suporte para a roda boba, de forma a nivelar o chassi.

## VI. Parte Eletrônica

Será usada uma bateria de 9V para alimentar o circuito. A bateria foi selecionada por ser mais compacta para a montagem; além de os participantes do projetos já possuírem e terem conhecimento prévio sobre esse componente. Uma ligação em paralelo será feita para utilizar a mesma bateria, alimentando ambos os componentes. Embora o arduíno necessite de 5V de tensão na entrada, o circuito é preparado internamente com um divisor de tensão, ou seja, qualquer tensão além dos 5V será dissipada. A Figura 4 mostra o esquema de ligação dos componentes eletrônicos.

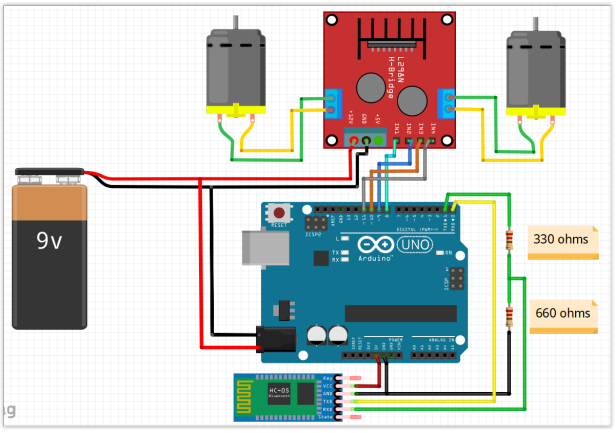


Figura 4: Esquemático de montagem dos componentes eletrônicos básicos do carrinho

A seguir descreve-se a disposição proposta dos componentes e o motivo de sua utilização. A ponte H (Figura 5) é um circuito que permite realizar a inversão da polaridade da corrente que flui através de uma carga, por isso é interessante para controlar os motores DC.

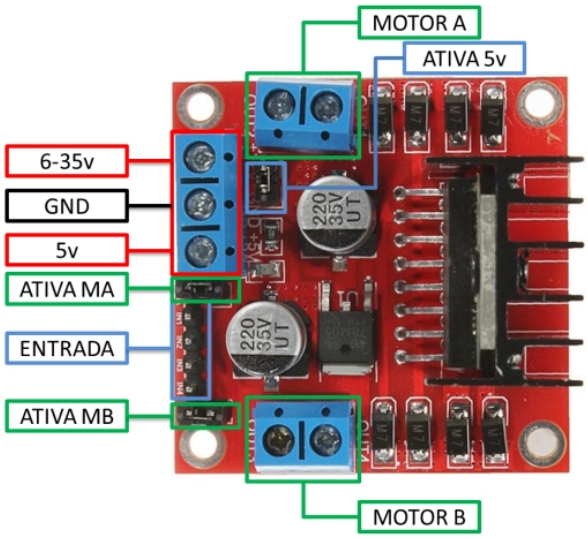


Figura 5: Ponte H com indicação de pinos

A Figura 5, mostra a *ponte H* e suas entradas. A *ponte H* pode ser alimentada de 2 formas, com tensão de 5V, no caso de ligação direta do arduino e *6-35V* para alimentação externa. Para utilizar a alimentação de 5V, o jumper em *Ativa 5V* deve ser removido para configurar uma referência na entrada de *6-35V*. *Ativa MA* corresponde ao *motor A* e *Ativa MB* à ativação do nível lógico para o *motor B*, para nosso projeto nenhum jumper foi removido. O arduíno foi o micro-controlador escolhido, o qual transmitirá para as rodas as coordenadas que serão informadas pelo dispositivo *bluetooth*. O arduíno passará instruções para os motores por meio das entradas *PWM* ( modelagem por largura de pulso) de números 4,5,6 7, 2 e 3. As portas 4,5,6 e 7 serão utilizadas para dar sentido para o movimento e as portas 2 e 3 serão a alimentação *PWM* na entrada de *enable* na *ponte H* dessa forma variando a tensão de ativação das rodas simulando vários *pings* de 0 a 5 volts. A utilização das portas *PWM* possibilita que a velocidade de rotação dos motores seja controlada e não uma constante. O *Arduíno Uno* foi escolhido por sua implementação ser mais simples, por exemplo a possibilidade de se realizar a sua carga diretamente do computador e também pela experiência prévia dos participantes do projeto com tal dispositivo.

O módulo *Bluetooth* escolhido foi o modelo *HC-06* que possui 4 pinos (alimentação, *GND*,*RX* e *TX*). Os dois últimos são utilizados para comunicação com o micro-controlador via serial. O nível lógico dos pinos *RX* e *TX* é 3.3V, logo precisamos de um divisor de tensão no pino *RX* para que esse não seja danificado. Esses pinos serão ligados na entrada serial do *Arduino Uno* que possui um par 0(*RX*) e 1(*TX*). A utilização do módulo *Bluetooth* deve-se ao processo de recebimento das coordenadas de localização do robô através de um marcador *QR-code*, esse sinal será transmintido através de um dispositivo *bluetooth*. O modelo *HC-06* foi escolhido por ter a implementação mais simples, a sua desvantagem é a restrição em somente receber dados. O arduíno foi ligado em paralelo a bateria de 9V. O nível lógico do arduino é de 5V e a sua capacidade de entrada é de 3.3V, para não haver nenhum dano foi montado um divisor de tensão utilizando dois resistores, projetando uma tensão máxima de 3.3V no pino *RX*.

## VII. Projeto da Parte de Software para Comunicação

A parte de software estabelecida para o projeto consiste de 3 códigos que foram utilizados para fazer um teste inicial de comunicação via *bluetooth* com smartphone, outro para testar a recepção dos dados recebidos do sistema de localização e por último um código responsável para receber e tratar os dados recebidos para serem utilizados para o controle do carrinho. Em todos os códigos, dentro do procedimento *setup*, é definido inicialmente uma comunicação serial via *bluetooth* e através de quais pinos do arduíno ela ocorre. Em todos os códigos a comunicação *bluetooth* recebeu o nome de *mySerial*. Para implementação do código foi utilizado o software do arduíno, escrito predominantemente em C++.

Após a configuração do módulo *bluetooth* para comunicação, foi implementado um código para testar a comunicação. Esse código realiza inicialmente um comando $(if(mySerial.available() > 0))$ para verificar se a comunicação *bluetooth* está estabelecida, depois de o ser teste realizado, foi utilizado o aplicativo android *Bluetooth SPP Pro* para enviar as letras A e B para o arduíno, dependendo da letra recebida, um *led* era aceso e em caso de outra letra recebida, outro *led* era aceso e nos dois casos uma mensagem de confirmação era retornada para o smartphone.

O código utilizado para testar o recebimento dos dados enviados pelo sistema e verificar o formato dos dados recebidos, faz um teste inicial da comunicação através dos comandos $if(mySerial.available() > 0)$ e $if(Serial.available() > 0))$, para verificar sua integridade, caso a comunicação esteja estabelecida, as mensagens recebidas do sistema de localização via *bluetooth* são passadas para o arduíno através do comando $(mySerial.write(Serial.read())$ que está dentro do *if* que avalia a comunicação serial) são escritas na saída (monitor serial do arduíno) utilizando o comando $(Serial.write(mySerial.read())$ que está dentro do if que avalia a comunicação *mySerial*), os dados são mostrados conforme recebidos, sem nenhum tratamento.

A parte de recebimento e manipulação da informação, basicamente (em alto nível) armazena a mensagem recebida, carácter por carácter, e depois separa a informação de acordo com o seu destino. Assim, caso a mensagem se refira ao nosso carrinho (letra C como identificador), sua posição nos eixos cartesianos x-y e o ângulo em relação a referência serão salvos em três variáveis. Caso a mensagem se refira a algum outro carrinho ou obstáculo (letras A, B, D ou E), as informações serão armazenadas em outras três variáveis. A tarefa desempenhada por esse código pode ser melhor representada por um fluxograma, conforme Figura 6 abaixo:

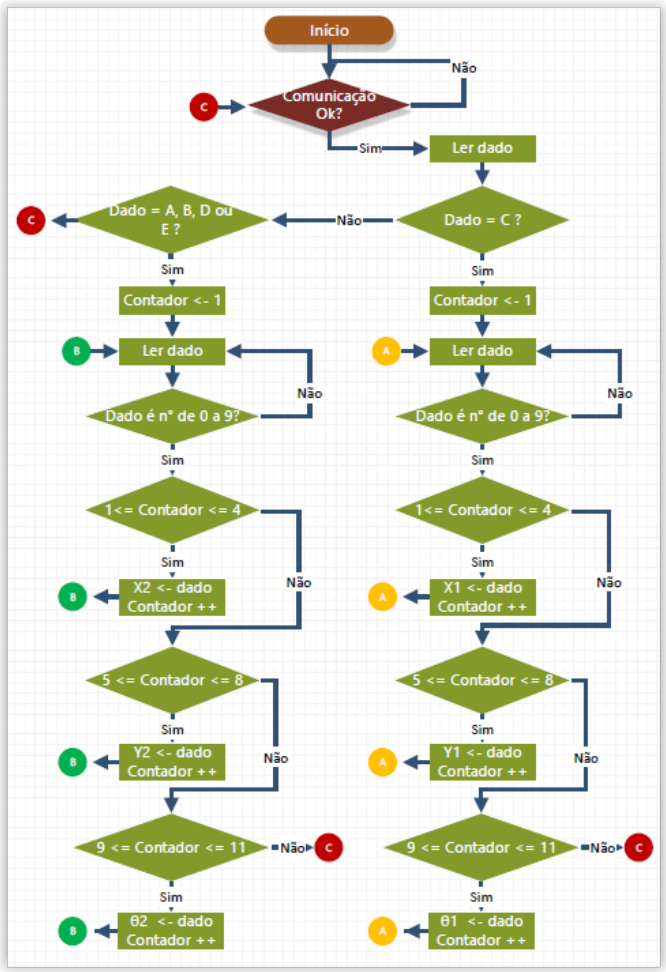


Figura 6: Fluxograma de Recepção dos Dados

Na Figura 6 as variáveis $X_1$, $Y_1$, $X_2$ e $Y_2$ são vetores de tamanho 4 e as variáveis $\theta_1$ e $\theta_2$ são vetores de tamanho 3. Os dados recebidos são do tipo *char* e são manipulados dessa forma, após a separação eles são convertidos para inteiro para que sejam utilizados em cálculos matemáticos. O fluxograma mostra como é feito o recebimento e manipulação para dois marcadores (carrinhos ou obstáculos), caso em uma aplicação exista mais marcadores, o mesmo código implementado baseado no fluxograma pode ser reutilizado com poucas alterações, incluindo novas variáveis X'n°', Y'n°' e $\theta$'n°' para cada marcador adicional.

## VIII. Projeto da Parte de Software para Controle

### A. Príncipios físicos e modelagem

A definição do modelo cinemático é de extrema importância para a definição do sistema de controle. Pois ele determina as equações que conduzem a dinâmica de movimentação do robô. Na figura Figura 7 pode-se observar esse modelo.

Esse modelo pode ser representado, pelas seguintes equações:

$$\frac{dx}{dt} = \frac{V_e + V_d}{2} cos\theta$$

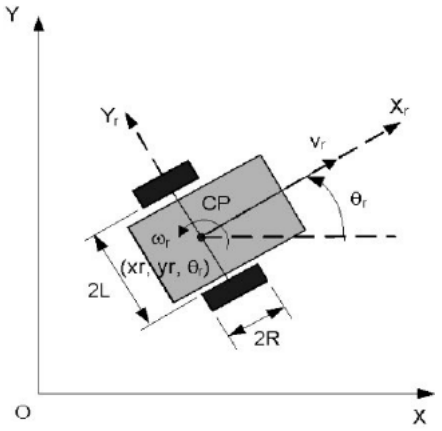


Figura 7: Modelo cinemático do Modelo

$$\frac{dy}{dt} = \frac{V_e + V_d}{2} sin\theta$$

$$\frac{d\theta}{dt} = \frac{V_e - V_d}{2}$$

A distância de referência para partida será calculada através da fórmula de distância Euclidiana mostrada abaixo:

$$d = \sqrt{(x_{k+1} - x)^2 + (y_{k+1} - y)^2}$$

### B. Projeto de Controlador

O controlador foi implementado utilizando a lógica *fuzzy*. O controlador *fuzzy* pode controlar a navegação e evitar, simultaneamente, os obstáculos. A estrutura básica do projeto contém três partes principais: fuzzificação, inferência e desfuzzificação. A primeira etapa é a fuzzificação que transforma os dados de entrada reais em funções de pertinência classificadas por magnitude. A segunda parte é a inferência da saída fazendo-se uma análise das entradas comparando-as com as regras pré-definidas. No caso deste trabalho como teremos mais de uma entrada a regra de decisão será do tipo:

$$IF(CONDICAO1)THEN(CONCLUSAO)$$

Por último será realizada a desfuzzificação que é o processo de se retirar resultados numéricos da função de pertinência de saída baseada nas saída da segunda etapa.

No *Matlab* o processo pode ser criado através do *plugin* mostrado abaixo na Figura Figura 8. Porém, com o intuito de desenvolver o controlador ferramentas *Open Source* foi decidido implementar a lógica *fuzzy* diretamente no *arduino* utilizando a biblioteca ***eFLL*** - *Embebed Fuzzy Logic Library*.

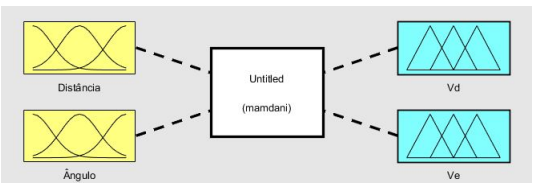


Figura 8: Diagrama Fuzzy

### C. Biblioteca Utilizada

Para implementar a lógica *fuzzy* no arduíno, existe a biblioteca *Embebed Fuzzy Logic Library* sendo toda desenvolvida utilizando apenas a biblioteca *stdlib.h*, logo é uma biblioteca que pode ser usada em qualquer sistema que suporte a linguagem C []c4. Os compomentes principais dessa biblioteca são:

- **Objeto *Fuzzy*:** Objeto que engloba todo o sistema *fuzzy*, com ele podemos manipular todos os conjuntos *fuzzy* e suas variáveis linguísticas.
- **Objeto *FuzzyInput*:** Representação da variável linguística de entrada do sistema. É bastante similar a variável criada para a saída do sistema.
- **Objeto *FuzzySet*:** É utilizado para a modelagem da função de pertinência de uma variável. As funções podem ser triangulares, trapezoiais e *singleton*.
- **Objeto *FuzzyOutput*:** Da mesma forma que o *FuzzyInput*, este objeto agrupa vários *FuzzySet* e formam a função de pertinência de saída.
- **Objeto *FuzzyRule*:** Esse objeto representa uma regra de decisão que será adicionada ao objeto *fuzzy*. Esse objeto é composto por um *FuzzyRuleAntecedent* e um *FuzzyRuleConsequent*. A base de conhecimento *fuzzy* deve, obrigatoriamente, conter um ou mais objetos *FuzzyRule*.
- **Objeto *FuzzyRuleAntecedent*:** Armazena a condição de ativação de uma regra.
- **Objeto *FuzzyRuleConsequent*:** Armazena o resultado se a regra for ativada.

### D. Implementação

Com a utilização da biblioteca *eFLL* a implementação é simplificada. O que torna a manutenção do código simples, se o mesmo for funcional, e facilita a inclusão de novas funcionalidades assim como a adicionar novas entradas e saídas. O primeiro passo da implementação é a criação do objeto *fuzzy* que armazenará todos os componentes do sistema:

$$Fuzzy * Fuzzy = newFuzzy();$$

A instância do objeto *fuzzy*, assim como todo o processo de criação da lógica *fuzzy*, deverá ser implementado dentro da função "*void setup()*"do arduíno, pois precisa ser feito apenas uma vez na inicialização do sistema. Na função *setup* , as variáveis de entrada "Distância"e "Ângulo"são criadas utilizando o objeto *FuzzyInput* e seus respectivos *IDs*, os quais devem ser passados como parâmetros. A modelagem da variável é feita utilizando-se os objetos *FuzzySet*. O código abaixo mostra a implementação da variável de entrada "distancia". Para uma modularização maior, primeiro foi instaciado os *FuzzySet*

$$FuzzySet * dmuitoPequena = new$$

$$FuzzySet(0, 0, 0, 125);$$

$$FuzzySet * distanciaPequena = new$$

$$FuzzySet(0, 125, 125, 250);$$

$$FuzzySet * distanciaMedia = new$$

$$FuzzySet(125, 250, 250, 375);$$

Após a modelagem, ainda na função *setup*, os *fuzzySets* são incorporados à variável *fuzzyInput distancia*, como pode ser visto abaixo.

$$FuzzyInput * dist = newFuzzyInput(2);$$

$$dist- > addFuzzySet(distanciaMuitoPequena);$$

$$dist- > addFuzzySet(distanciaPequena);$$

$$dist- > addFuzzySet(distanciaMedia);$$

Finalmente, o objeto *FuzzyInput* é incorporado no objeto *fuzzy*:

$$fuzzy- > addFuzzyInput(dist);$$

Os passos citados acima devem ser feitos para todas as variáveis de entrada e saída do sistema. A única diferença é que para o sistema de saída o objeto é *FuzzyOutput()*. Uma vez cadastrado todas as entrada e saída do sistema, é preciso incluir a base de regras. Na biblioteca *eFLL* a regra é formada por uma condição antecedente e por um estado que é consequência. Primeiramente é importante definir qual será a regra. Nesse caso, como exemplo temos: **Se** a distância é grande **E** o ângulo é positivo pequeno **ENTÂO** a velocidade de saída na roda direita é baixa e na roda esquerda é média. A Regra é criada da seguinte forma, primeiro é instanciado um objeto *FuzzyRuleAntecedent*

$$FuzzyRuleAntecedent*$$

$$distanciaGrandeEAnguloPPequeno = new$$

$$FuzzyRuleAntecedent();$$

Depois é utilizada a função *joinWithAND* para a operação AND da seguinte forma:

$$distanciaGrandeEAnguloPPequeno$$

$$- > joinWithAND(dmuitoPequena, pPequeno);$$

Em seguida, é instanciado o objeto que representará a consequência:

$$FuzzyRuleConsequent*thenVdisBmAndVeZ = new$$

$$FuzzyRuleConsequent();$$

Feito isso precisamos incorporar essa regra à saída do sistema.

$$thenVdisBmAndVeZ- > addOutput(vBaixa_d);$$

Ao final é criada a própria regra utilizando o objeto *FuzzyRule*. Todos os passos descritos acima devem ser seguidos para criar todas a regras necessárias.
A entrada do sistema, fuzzificação, e saída, desfuzzificação, são implementadas na função *loop()*, nesse projeto os parâmetros de entrada, distância e ângulo seriam obtidos através de funções também codificadas para cálculo do ângulo a ser ajustado para

o movimento e da distância euclidiana. Os valores de distância iriam variar de 0 a 1000mm e deveriam ser inseridos no sistema fuzzy da seguinte forma:

$$fuzzy->setInput(1, angulo);$$

$$fuzzy->setInput(2, distanca);$$

Para iniciar o processo de fuzzificação, é utilizada a função *fuzzify()*:

$$fuzzy->fuzzify();$$

Essa função faz a inferência baseada nos valores de entrada de cada *fuzzyInput*. Para a desfuzzificação é necessário fazer a chamada da função *defuzzify* passando como parâmetro o *ID* do objeto *FuzzyOutput* que corresponde a saída desejada.

$$floatv1 = Fuzzy->defuzzify(1)$$

Por último, o valor de saída, que será entre 0 e 70, foi convertido para o valor utilizado no *PWM* do arduíno, que é entre 0 e 255, dessa forma podemos controlar a velocidade dos motores. Outro passo que deve ser feito na função de *setup* é a configuração da porta que deverá ser utilizada como *PWM*.

$$pinMode(3, OUTPUT);$$

O valor pode ser convertido com a função mostrada abaixo. Essa função basicamente linearia todos os valores com base no máximo estabelecido e então multiplica pelo valor máximo que pode ser aplicado na saída *PWM*.

$$velodidadeD = \frac{v1}{70} \times 255$$

Na saída *PWM* o valor pode ser aplicado da seguinte maneira:

$$analogWrite(3, velocidadeD);$$

As regras que serão utilizadas para implementação do sistema estão exemplificadas na tabela I. Os estados são explicados abaixo:

1) Distância $d$
   - MP : Muito Pequena.
   - P : Pequena.
   - M : Médio.
   - G : Grande.
   - MG : Muito Grande.
2) Ângulo de deslocamento $\theta$
   - Z : Zero.
   - PP : Positivo Pequeno.
   - PM : Positivo Médio.
   - PG : Positivo Grande.
   - NP : Negativo Pequeno.
3) Velocidade
   - Z : Zero.
   - MB : Muito Baixa.
   - B : Baixa.
   - M : Média.

Tabela I: Indicação Variáveis

| Distância | Ângulo | Vd | Ve |
|---|---|---|---|
| MP | Z | MB | MB |
| P | PP | Z | MB |
| M | PM | MB | B |
| G | PG | R | MR |
| MG | NP | B | MB |

   - R : Rápida.
   - MR : Muito Rápido.

Os funções de pertinência de entrada têm características como mostrados na figura Figura 9 e figura Figura 10

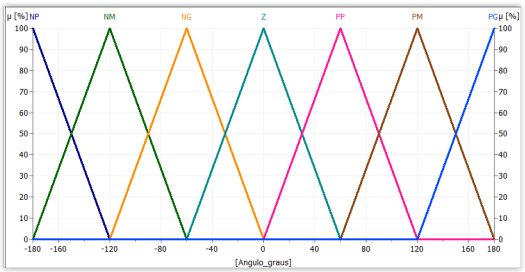


Figura 9: Função de pertinência para o ângulo.

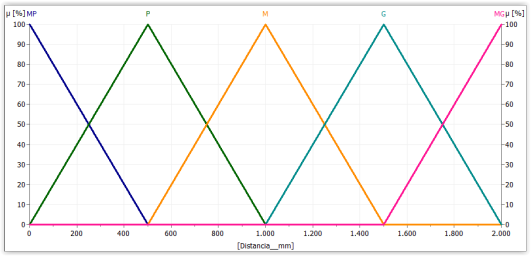


Figura 10: Função de pertinência para a distância.

### *E. Decisões de Implementação*

O software de controle irá basicamente definir qual a velocidade de cada roda, ou seja, a velocidade de rotação de cada motor DC. As velocidades são definidas como $V_d$ e $V_e$ sendo respectivamente as velocidades de rotação das rodas direita e esquerda. O projeto de controle foi dividido em duas fases. Na primeira fase, o deslocamento do destino até a origem. Na segunda fase, o robô realizará o trajeto proposto evitando obstáculos. Em resumo:

- Fase 1: Deslocamento Simples;
- Fase 2: Deslocamento com desvio de obstáculos;

As variáveis utilizadas na solução serão basicamente as mostradas na tabela II

## IX. Fuzzy logic Controller x Controle Clássico

Durante o desenvolvimento do trabalho dificuldades foram encontradas para debugar o código quando os resultados apresentados eram resultados não esperados. O controlador de lógica *fuzzy* foi escolhido para resolver o problema devido a proximidade das regras de decisão com a linguagem

Tabela II: Variáveis de controle

| Parâmetros | |
|---|---|
| d | Distância |
| $\theta$ | Ângulo de deslocamento |
| $V_d, V_e$ | Velocidade |
| X,Y | Coordenadas de Origem |
| x,y | Coordenadas de Destino |
| $x_o, y_o$ | Coordenadas do Obstáculo |

natural. Uma outra solução seria a implementação de um controlador proporcional integrativo derivativo (PID),esses três componentes são variados para se obter uma resposta ideal. O controlador PID é um dos mais usados devido a sua simplicidade. O controlador baseado em lógica *fuzzy* têm diversas vantagens, mas apresenta também muitos pontos negativos que serão citados a seguir. Dentre as vantagens mais importantes da lógica Fuzzy pode-se citar:

- Flexível e a base de conhecimento construída para o controle é bastante intuitiva pois as regras de controle são facilmente entendidas na linguagem normal.
- Conveniente para o usuário final, é simples para o usuário final identificar as regras e suas consequências.
- A lógica fuzzy é simples de ser implementada computacionalmente.
- Fácil de validar e verificar as regras de saúde, uma vez que o sistema seja funcional. Isso facilita nos casos onde a lógica *fuzzy* será utilizada juntamente com algum algoritmo de aprendizado de máquina.
- *fuzzy* é uma forma não ambígua de representar uma informação incerta na forma de linguagem natural.

Apesar de parecer ser aplicável em inúmeros cenários e ser a solução perfeita em vários casos, a lógica fuzzy apresenta também alguns pontos negativos. Abaixo são citados alguns:

- A sintonização da regras de decisão devem ser feitas manualmente, o que aumenta o tempo necessário para o desenvolvimento.
- Não é necessariamente melhor que um controle PID, que é mais simples de ser implementado.
- Em algumas situações, a relação robustez-performance não é considerada na sintonização do controlador *fuzzy*. Geralmente assume-se que a robustez é uma propriedade fundamental do controlador *fuzzy*, o que não é verdade, uma vez que se for mal sintonizado as regras podem não avaliar todo o domínio da planta.
- Em alguns casos, as regras de decisão são equivalentes a pesquisar o resultado em uma matriz de decisão.
- Se a dimensão do problema for muito grande é praticamente impossível de se implementar todas as regras de decisão, o que torna a dimensionalidade uma das maiores restrições do Controlador de Lógica *fuzzy* []c6.

O sistema de controle desejado nesse trabalho era, de certa forma, simples. Tardiamente foi constatado que a lógica *fuzzy* pode ser entendida como uma solução muito complexa para o desafio proposto. A falta de sucesso no projeto, pode ser ressaltada também para evidenciar a dificuldade da identificação de problemas com a lógica *fuzzy*, já que por ser muito próxima da linguagem natural é difícil de entender onde está localizado o erro. Para trabalho, a utilização de uma biblioteca em uma linguagem mais baixo nível, torna o processo mais lento, pois se trabalha com a construção manual das curvas e não com uma interface gráfica, como seria se tivesse sido utilizado o *MATLAB*.

### A. *Resultados Experimentais e Simulações do Controlador*

Após finalizar a implementação do controlador, testes foram feitos para averiguar se o sistema respondia adequadamente para determinadas entradas e saídas. Foi observado então que a velocidade de saída não era suficiente para a alimentação do da porta *PWM*. Para entender melhor o como as regras estavam atuando, as regras foram simuladas utilizando a função *fuzzy* no MATLAB. Na Figura Figura 11 mostramos o diagrama de regras de decisão para a seguinte entrada: distância igual a 364mm e ângulo a ser ajustado igual a 70 graus. Podemos observar que a velocidade de saída é baixa, como esperado, pois o robô estaria fazendo o ajuste do ângulo para o movimento. A Velocidade de saída na roda direita é 17,9mm/s e 33,9mm/s na roda esquerda. O que colocado na fórmula mostrada os dá um 25% de 255 e 48% de 255 na outra (255 é o maxímo valor que pode ser colocado na saída PWM). Como o controlador implementado não é integrador, ou seja não existe uma previsão do caminho incialmente traçado e portanto torna muito difícil simular um caminho qualquer com o robô.

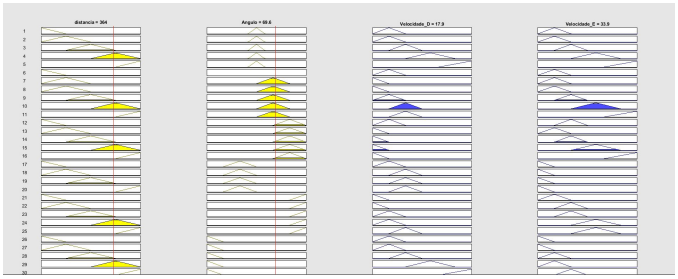

Figura 11: Regras de decisão do controlador plotadas no MATLAB

Na Figura Figura 12 é possível observar a saída comentada acima na saída serial do arduino como mostrada na simulação mostrada no MATLAB.

```
Distancia: 0.00, 0.00, 0.41, 0.59
, 0.00
Angulo: 0.00, 0.00, 0.16, 0.00
Velocidade Roda Direita: 17.90, Velocidade Roda Esquerda: 33.90
Distancia: 0.00, 0.00, 0.41, 0.59
, 0.00
Angulo: 0.00, 0.00, 0.16, 0.00
Velocidade Roda Direita: 17.90, Velocidade Roda Esquerda: 33.90
Distancia: 0.00, 0.00, 0.41, 0.59
, 0.00
Angulo: 0.00, 0.00, 0.16, 0.00
Velocidade Roda Direita: 17.90, Velocidade Roda Esquerda: 33.90
```

Figura 12: Resultados da simulação de controle para entradas: $Distancia = 324$ e $Angulo = 69.3$

## X. Resultados Experimentais e Simulações da Comunicação

Para o teste de comunicação inicial foi implementado um circuito utilizando 2 *leds* vermelhos e 2 resistores de 150 ohms.

Quando a letra ”A”é recebida do smarthphone o *led 1* é aceso e uma mensagem, ”LED 1 ligado !”, é retornada para o telefone, caso a letra ”B”é recebida o *led* 2 é aceso e a mensagem de confirmação é, ”LED 2 ligado !”. O teste foi realizado diversas vezes e apresentou resultado satisfatório em todas elas.

No teste para recepção e verificação do formato das mensagens recebidas, o sistema de localização enviou centenas de mensagens e todas foram recebidas corretamente. A seguir na Figura 13 é mostrado algumas mensagens recebidas:

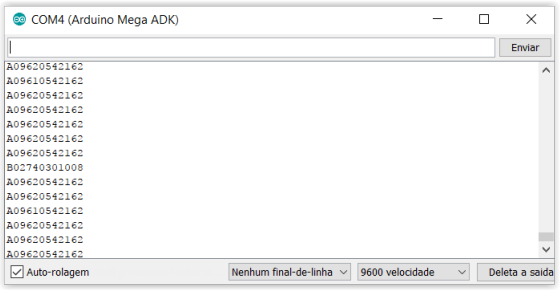


Figura 13: Mensagens recebidas do sistema de localização

Para testar a parte do código referente à recepção e manipulação dos dados, foram utilizadas várias mensagens em sequência com o mesmo formato do protocolo de comunicação (recebidas do sistema em outro momento): 1 letra para identificar o marcador seguida de 11 caracteres, sendo 4 para cada eixo de referência, 3 para o ângulo e 1 carácter de mudança de linha. Essas mensagens foram enviadas do computador para o arduíno nas taxas de 9600, 38400 e 115200 bps. Para simulação foi incluído no código proposto duas variáveis para contabilizar o número de mensagens recebidas e destinadas para um identificador e para o outro identificador e comandos para apresentação do valor dessas variáveis no monitor serial. Na taxa de 38400 bps foram enviadas 200 mensagens, sendo 100 para cada identificador, o procedimento foi repetido 5 vezes, sendo que em média 88,8% das mensagens foram recebidas e processadas pelo arduíno. Para as taxas de 9600 e 115200 bps foi repetido o procedimento, onde 89% e 91,9% em média das mensagens foram recebidas, respectivamente.

## XI. Conclusões

Durante a primeira parte do projeto foi realizada uma análise macro dos objetivos, requisitos e tarefas necessárias para se atingir o produto final desejado. Havendo a criação de diferentes desenhos e soluções para o problema proposto. Três fatores principais nortearam a escolha do desenho final e da disposição dos componentes. Primeiramente o custo dos materiais, optou-se por associar confiabilidade com o preço acessível dos componentes; outro fator importante foi o objetivo do cliente, o qual tinha que estar submetido a algumas restrições de projeto já citadas; e o último fator importante para a parte mecânica foi a estabilidade, agilidade e resistência do robô. Nas etapas de construção e implementação da parte mecânica e eletrônica desse projeto não houveram problemas significativos.

Na fase de implementação de software e testes para comunicação não foi possível verificar se a manipulação dos dados estava sendo realizada de forma correta. Isso ocorreu devido a não realização de simulações com o objetivo de validar essa funcionalidade do código, já que a comunicação estava funcionando corretamente. Assim, não foi constatada a efetividade dessa funcionalidade e alterações foram implementadas no código para que o resultado da manipulação fosse apresentado no monitor serial da IDE do arduíno. Porém isso não realizado em tempo hábil, caso a simulação da comunicação entre computador e arduino fosse realizada previamente, não haveria sido demandado longo tempo com uma medida paliativa, a qual se mostrou ineficiente. Após a apresentação do trabalho foi realizado a simulação apresentada na sessão e foi observado que o código proposto provavelmente seria capaz de realizar a tarefa de manipulação dos dados, ou seja, não seria necessário realizar a validação utilizando o sistema de localização como foi tentado e não deu certo. Apesar dos equívocos nesta parte, pode-se obter o aprendizado de que é preciso definir de forma prévia e clara quais os testes e os resultados necessários para validação parcial de uma etapa do projeto, bem como medidas alternativas em caso da não possibilidade de realização ou resultado insatisfatório dos testes.

A implementação do sistema de controle foi desafiadora e nesse trabalho foi apresentada apenas uma alternativa de implementação. É importante deixar claro que os métodos acima descritos foram verificados separadamente e portanto não foi possível verificar o funcionamento do sistema como um todo. Os próximos passos serão a implementação completa das duas fases e testes. Um dos pontos a ser melhor trabalhado em projetos futuros é o caracterização mais completa do projeto, com a criação e determinação de sub-tarefas e sub-objetivos dentro do escorpo principal do projeto, facilitando a solução dos problemas que surgirem ao longo do projeto e detalhando melhor a abordagem do problema principal.

## Referências

[1] https://tecnoblog.net/178274/carros-autonomos-google-ruas-california. Acesso em 07/12/2017.
[2] https://g1.globo.com/carros/noticia/rival-do-uber-faz-testes-com-clientes-em-carros-autonomos-nos-eua.ghtml. Acesso em 07/12/2017.
[3] http://irisbh.com.br/carros-autonomos-desafios-e-perspectivas . Acesso em 07/12/2017.
[4] eFLL - Embebed Fuzzy Logic Library. Disponível em github.com/zerokol/eFLL. Acessado em 10 de Outubro de 2017.
[5] http://www1.folha.uol.com.br/mercado/2017/11/1936747-uber-planeja-comprar-24-mil-carros-autonomos-da-volvo.shtml. Acesso em 07/12/2017.
[6] Albertos, Pedro, and Antonio Sala. ”Fuzzy logic controllers. Advantages and drawbacks.”VIII International Congress of Automatic Control. Vol. 3. 1998.