INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

GUILHERME EGLÉ PEGADO LIMA SILVA

Desenvolvimento de software para robô de resgate de vítimas em ambiente de desastre simulado

CEARÁ-MIRIM-RN
2018

GUILHERME EGLÉ PEGADO LIMA SILVA

Desenvolvimento de software para robô de resgate de vítimas em ambiente de desastre simulado

Relatório de Prática Profissional apresentado ao Curso Técnico Integrado em Programação de Jogos Digitais do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial para a obtenção do título de Técnico em Programação de Jogos Digitais.

Orientador: Diego Alves Formiga
Co-orientador: Pedro Iuri Soares de Souza

Ceará-Mirim-RN
2018

GUILHERME EGLÉ PEGADO LIMA SILVA

Desenvolvimento de software para robô de resgate de vítimas em ambiente de desastre simulado

Relatório de Prática Profissional apresentado ao Curso Técnico Integrado em Programação de Jogos Digitais do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial para a obtenção do título de Técnico em Programação de Jogos Digitais.

Aprovado em: ____/____/_____

Nota Final: _____

_____
Prof. Me. Diego Alves Formiga
Matrícula: 2052309

_____
Prof. Pedro Iuri Soares de Souza
Matrícula: 3273771

_____
Prof. Carlos Alberto de Albuquerque Silva
Matrícula: 2378789

_____
Prof. Dr. Gustavo Viella Whately
Coordenador do Curso Técnico Integrado de Programação de Jogos Digitais
Matrícula: 2209199

RESUMO

Esse relatório relata o desenvolvimento de um software/algoritmo para um robô que deve realizar resgates de vítimas em ambientes previamente conhecidos, mas, cujas condições sejam de difícil acesso e/ou ofereçam risco a vida da equipe de resgate, dessa forma, sendo necessária a utilização de um robô autônomo, em um ambiente com os mais variados tipos de problemas e obstáculos que possam se apresentar ao longo do salvamento. Todo o ambiente de resgate será simulado, de acordo com o previsto na Olimpíada Brasileira de Robótica (OBR). Exemplos dessas complicações são: obstruções de caminho, terrenos elevados, sobreposição de trajetos, desvios, entre outras problemáticas alvo de análise. O trajeto deve ser percorrido de forma a encontrar a região onde a vítima se encontra. A vítima, por sua vez, será representada por um material característico, como uma esfera metálica, que deverá ser transportado até um local reservado, fazendo analogia com uma região onde seriam feitos os primeiros procedimentos de socorro à vítima.

Palavras-Chave: Resgate; Robô; Autônomo.

SUMÁRIO

1 INTRODUÇÃO 6

1.1 JUSTIFICATIVA 6
1.2 OBJETIVOS 7

2 DADOS GERAIS DA PESQUISA 8

2.1 SÍNTESE DE CARGA HORÁRIA E ATIVIDADES 8

3 METODOLOGIA 9

4 FUNDAMENTAÇÃO TEÓRICA 10

5 DESCRIÇÃO DA PESQUISA 12

5.1 ESTUDO DA REGRA DAS COMPETIÇÕES DE ROBÓTICA E DESAFIOS A SEREM SUPERADOS PELO ROBÔ **ERRO! INDICADOR NÃO DEFINIDO.**12
5.2 CLASSIFICAÇÃO E ESTUDO DOS COMPONENTES 12
5.3 ANÁLISE ESTRATÉGICA DA POSIÇÃO DOS COMPONENTES NO PROTÓTIPO 12
5.4 DESENVOLVIMENTO DE ALGORITMOS DE MOVIMENTO UTILIZANDO 4 MOTORES DC E 2 PONTES H 12
5.5 DESENVOLVIMENTO DE UM ALGORITMO SEGUIDOR DE LINHA UTILIZANDO SENSORES DE LINHA/INFRAVERMELHO 12
5.6 DESENVOLVIMENTO DE UM ALGORITMO PARA DETECÇÃO E DESVIO DE OBSTÁCULOS UTILIZANDO SENSORES ULTRASSÔNICOS 12

6 CONSIDERAÇÕES FINAIS 29

6.1 TRABALHOS FUTUROS 29
7 ANEXOS 14
7.1 ANEXO I 15
7.2 ANEXO II 16
7.3 ANEXO III 17

REFERÊNCIAS 33

1 INTRODUÇÃO

As tecnologias têm, ao longo dos anos, proporcionado formas diversas de lazer e nos trazem comodidade, alterando tarefas que outrora eram realizadas de forma manual. Porém, estas tecnologias, se corretamente implementadas, também podem representar a nossa sobrevivência e bem-estar. Em um ambiente de desastre, seja ele ambiental ou provocado pelo próprio ser humano, onde a situação em que uma vítima que precisa ser resgatada é perigosa demais até para a própria equipe de salvamento, a tecnologia passa a ser uma importante aliada para a manutenção da vida. A partir dessa premissa, a utilização de robôs de resgate torna-se fundamental para o mundo moderno. O protótipo desenvolvido para este projeto implementa tal tecnologia aliando-se ao baixo custo e versatilidade das plataformas de prototipagem baseadas em microcontroladores.

1.1 JUSTIFICATIVA

A humanidade, ao longo de toda sua existência, sempre esteve sujeita a desastres como tempestades, terremotos e tsunamis. O avanço tecnológico e a exploração de novos territórios acabaram trazendo outras preocupações como desabamentos de construções, vazamentos de material radioativo em usinas nucleares, entre outras situações de sinistros que comumente colocam vidas em situações extremamente perigosas. É neste contexto que o auxílio de robôs autônomos, previamente programados, faz-se necessário e podem vir a determinar o sucesso de uma operação de resgate. Robôs autônomos tem por objetivo o desenvolvimento de tarefas, muitas delas perigosas ou impróprias para o homem, sem nenhuma interação externa. O propósito deste projeto é direcionar esta característica para o resgate de vítimas, ou seja, o desenvolvimento de um software para um protótipo de um robô seguidor de linha autônomo para realização de resgastes em ambientes simulados, sendo testado sua funcionalidade em competições de robótica. O desenvolvimento de um protótipo tendo por base uma situação prática e de extrema relevância.

1.2 OBJETIVOS

O objetivo fundamental deste projeto é o desenvolvimento de um software utilizando a plataforma do arduino que possibilite a autonomia de um robô para que ele seja capaz de realizar a tarefa de resgate de uma vítima percorrendo um caminho previamente conhecido e sendo capaz de superar terrenos irregulares, transpor caminhos onde o trajeto inicial não possa ser reconhecido, desviar e subir terrenos elevados, resgatar a vítima e levá-la até um local seguro. Todos estes obstáculos serão representados por meio de objetos que fazem analogia a um incidente real.

Objetivos Específicos

- Estudo das regras das competições de robótica e desafios a serem superados pelo robô**Erro! Indicador não definido.**;
- Classificação e estudo dos componentes;
- Análise estratégica da posição dos componentes no protótipo;
- Desenvolvimento de algoritmos para movimentação;
- Desenvolvimento de um algoritmo seguidor de linha utilizando sensores de linha/infravermelho;
- Desenvolvimento de um algoritmo para detecção e desvio de obstáculos utilizando sensores ultrassônicos;
- Desenvolvimento de um algoritmo de identificação de direções por fitas de cor verde utilizando sensores RGB;
- Implementação de algoritmo para o resgate de vítimas utilizando uma garra e servo motores. (Não executado)

2 DADOS GERAIS DA PESQUISA

TÍTULO DO PROJETO: Desenvolvimento de software para robô de resgate de vítimas em ambiente de desastre simulado

PERÍODO DE REALIZAÇÃO: 18/04/2018 a 18/11/2018

TOTAL DE HORAS: 400 horas.

CO-ORIENTAÇÃO:

Nome do co-orientador: Pedro Iuri Soares de Souza

Função: Professor EBTT

Formação profissional: Graduação em Engenharia elétrica

2.1 SÍNTESE DE CARGA HORÁRIA E ATIVIDADES

Quadro 1 – Síntese de Carga horária e Atividades.

CARGA HORÁRIA	ATIVIDADES DESENVOLVIDAS
30 horas	Estudo das regras das competições de robótica e desafios a serem superados pelo robô**Erro! Indicador não definido.**
30 horas	Classificação e estudo dos componentes
40 horas	Análise estratégica da posição dos componentes no protótipo
40 horas	Desenvolvimento de algoritmos para movimentação
80 horas	Desenvolvimento de um algoritmo seguidor de linha utilizando sensores de linha/infravermelho
80 horas	Desenvolvimento de um algoritmo para detecção e desvio de obstáculos utilizando sensores ultrassônicos
50 horas	Desenvolvimento de um algoritmo de identificação de direções por fitas de cor verde utilizando sensores RGB
50 horas	Escrita do relatório

3 METODOLOGIA

Iniciou-se pelo estudo das regras de competições de robótica, como a Olimpíada Brasileira de Robótica (OBR) e a SECITEX, com o objetivo de compreender como o robô protótipo deveria se comportar e quais seriam desafios que representariam um ambiente de desastre simulado, deveriam ser superados.

A primeira parte para a criação do robô é a movimentação, essa é implementada logo quando o primeiro modelo de chassi é montado, para ela é necessário aperas o chassi, um arduino, pontes H e os motores de corrente continua. Com isso, desenvolvemos um algoritmo capaz de realizar os seguintes movimentos: andar para frente, andar para trás, parar de andar, fazer curva para direita, fazer curva para esquerda, girar para direita e girar para esquerda.

Posteriormente, quando os sensores infravermelhos foram instalados, começamos o desenvolvimento do algoritmo seguidor de linha, que fosse capaz de interpretar o posicionamento do robô sobre a linha preta e utilizando as funções de movimento conseguisse fazer o percurso.

Baseando no desempenho de competições anteriores, foi colocado como uma necessidade a criação de um algoritmo de caibramento de sensores de linha visando possíveis variações de luminosidade dependendo do ambiente, desse modo, aumentando a eficácia do algoritmo seguidor de linha.

Após isso, foram feitos estudos e testes de controle a fim de definir uma melhor distribuição para os sensores de linha, sua disposição no chassi. Com o robô finalmente seguindo linha passamos para o estudo dos possíveis desafios que poderiam aparecer durante o percurso, como: gap (ausência da linha), encruzilhadas, obstáculos, redutores de velocidade (lombadas) e terrenos mais elevados, afim de alcançar soluções viáveis apara cada desafio.

Para solucionar os gaps e as encruzilhadas, algoritmos de identificação foram incrementados ao código do algoritmo seguidor de linha, também foi feita uma modificação na disposição dos sensores de linha afim do robô ser mais preciso em relação ao processo de seguir linha. Nas encruzilhadas é preciso tomar uma decisão sobre que direção o robô deve seguir, essa direção é identificada por uma marcação

verde. Para identificar essas situações, é utilizado o sensor RGB, sensores que captam a variação da frequência da luz foram posicionados atrás dos sensores infravermelho (de linha), com um algoritmo adequado é possível identificar a presença dessa fita e mudar a direção do robô.

Posteriormente, adicionamos a funcionalidade de identificação e contorno de obstáculos, que são objetos paralelepípedos que são colocados no percurso para dificultar o trajeto do robô, o obrigando a desviar, utilizamos 3 sensores ultrassônicos (sensores medidores de distância), de modo que o sensor frontal realizava leituras constantemente e caso o robô encontra-se um obstáculo durante o caminho, o sensor frontal permite a identificação desse obstáculo pela variação da distância, após a identificação do obstáculo o robô iria contornar e o sensor ultrassônico da lateral a qual o contorno está sendo feito ira regular o contorno com base na distância.

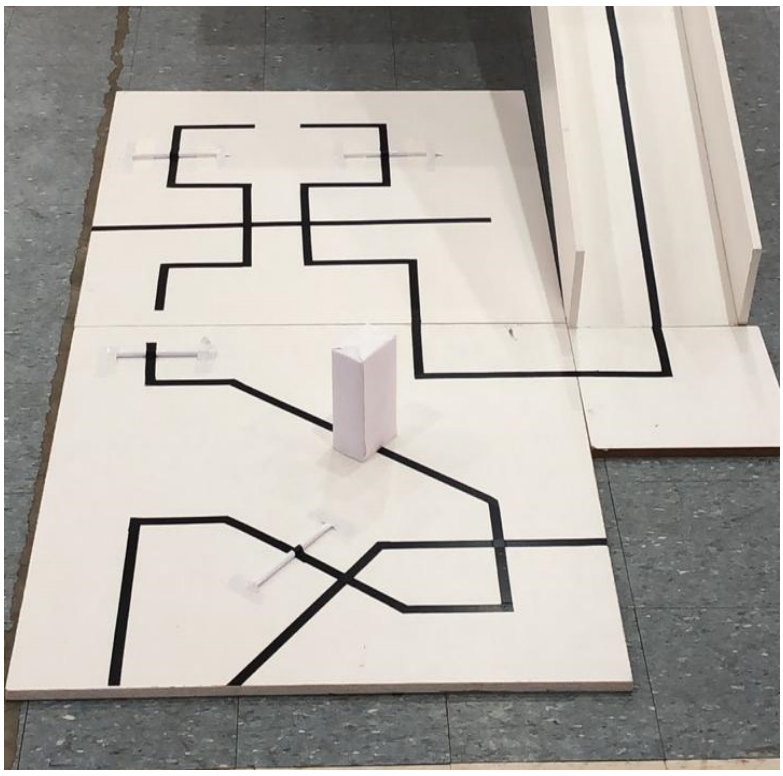

Figura 1: Exemplo de pista com obstáculo.

4 FUNDAMENTAÇÃO TEÓRICA

Microcontroladores (Placas de prototipagem)

Os microcontroladores são computadores de dimensões reduzidas capazes controlar outras maquinas, sistemas e equipamentos eletroeletrônicos através de programas. Esses dispositivos possuem um circuito único integrado. Desse modo, um microcontrolador é um microcomputador integrado em um único chip. Por se tratar de um componente programável, é bem versátil, podendo ser utilizado em diversos tipos de aplicações.

Microcontroladores vem sendo cada vez mais utilizados nos equipamentos eletrônicos modernos, atualmente os microcontroladores estão inseridos em: celulares, televisões, fornos de micro-ondas e etc. O uso dos microcontroladores em grande escala só é possível devido à sua grande versatilidade, já que, em um dispositivo tão pequeno dispõe-se de todos os elementos necessários para controlar as funções desses equipamentos.

Arduino (Plataforma)

A Plataforma Arduino teve sua origem em Ivrea, na Itália, em 2005. O seu surgimento ocorreu quando um grupo de acadêmicos da área de computação, formado por Massimo Banzi, Tom Igoe, Gianluca Martino e David Mellis, insatisfeitos com a complexidade com que os pacotes de computação física, oferecidos no mercado, se apresentavam, desenvolveram um novo projeto, com plataforma de software e hardware mais simples e com um nível menor de exigência de conhecimentos de engenharia da computação. Este projeto obteve sucesso no âmbito acadêmico, atraindo a atenção do engenheiro espanhol, David Cuartielles detentor de grande experiência em microcontroladores, que se interessou pelo desenvolvimento da plataforma em caráter comercial (ARDUINO, 2010).

O propósito desta equipe foi o de criar um ambiente de programação e uma placa padrão que fosse capaz de suportar um microcontrolador, com acesso ilimitado e livre, possibilitando o uso e manuseio tanto do hardware como do software. A plataforma criada foi validada na própria universidade, sendo prontamente aprovada

pelos estudantes que, mesmo não possuindo conhecimentos profundos em computação física e robótica, puderam executar projetos mais complexos (ARDUINO, 2010).

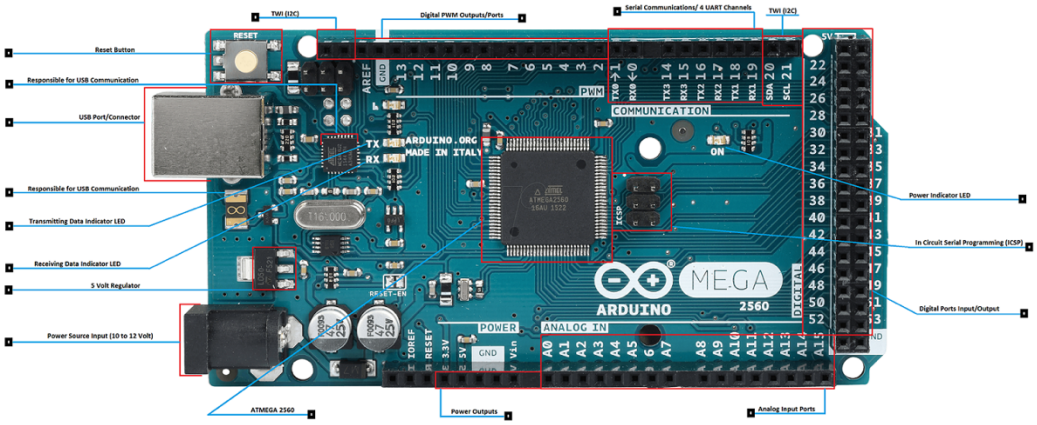

Figura 2: Arduino MEGA 2560.

Ambiente integrado de desenvolvimento (IDE)

O ambiente e a linguagem de programação também merecem destaque. A linguagem facilita a interação com os periféricos, tais como: sensores de luz, de som, ultrassom, infravermelho e diversos tipos de motores. E o ambiente faz menção ao seu IDE (Integrated Development Enviroment ou Ambiente integrado de Desenvolvimento), o software de desenvolvimento que auxilia o programador a trabalhar com as mais variadas ferramentas o que torna a criação do código e do algoritmo mais eficiente.

Software

Os computadores são muito bons em armazenar informações e fazer cálculos, mas não são capazes de tomar decisões sozinhos. Sempre existe um ser humano orientando o computador e dizendo a ele o que fazer a cada passo. Seja você mesmo, teclando e usando o mouse, ou, num nível mais baixo, o programador que escreveu os programas que você está usando. Chegamos então aos softwares, gigantescas cadeias de instruções que permitem que os computadores façam coisas úteis. É aí que entra o sistema operacional e, depois dele, os programas que usamos no dia-a-dia (MORIMOTO, 2007).

Sensor infravermelho/linha/óptico reflexivo

O sensor óptico reflexivo TCRT5000 e composto por dois componentes no mesmo suporte: um LED infravermelho (cor azul) e um transistor IR (fototransistor - cor preta), separados por uma pequena barreira. Quando algum objeto se aproxima do sensor, a luz infravermelha emitida pelo LED e refletida no objeto e ativará o transistor fazendo-o conduzir (fechar).

O sensor de linha TCRT5000 geralmente vem em um modulo acompanhando por outros componentes eletrônicos. Este sensor possui uma saída analógica, mas com a ajuda de um CI comparador e obtida uma saída digital e com o uso de um potenciômetro e admissível um ajuste na sensibilidade do sensor, em que a sensibilidade máxima deste sensor e de 25 mm (FORMIGA, 2018).

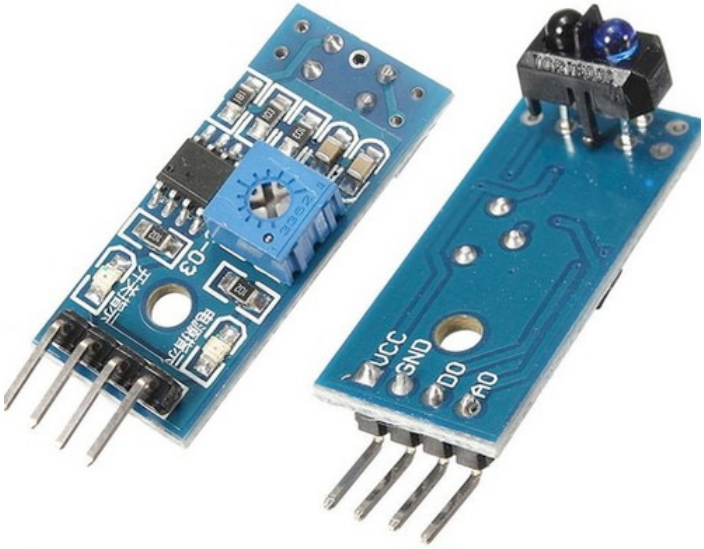

Figura 3: Sensor de linha TCRT5000.

Sensor Ultrassônico

O Sensor Ultrassônico HC-SR04 permite a leitura de distancias entre 2 cm e 4 metros, com precisão de 3 mm. Pode ser utilizado simplesmente para medir a distância entre o sensor e um objeto, como para acionar portas do microcontrolador, desviar um robô de obstáculos, acionar alarmes, etc. O funcionamento do HC-SR04 se baseia no envio de sinais ultrassônicos pelo sensor, que aguarda o retorno (*echo*) do sinal. Como a velocidade de propagação da onda e conhecida e com base no tempo entre envio e retorno, calcula-se a distância entre o sensor e o objeto detectado (FORMIGA, 2018).

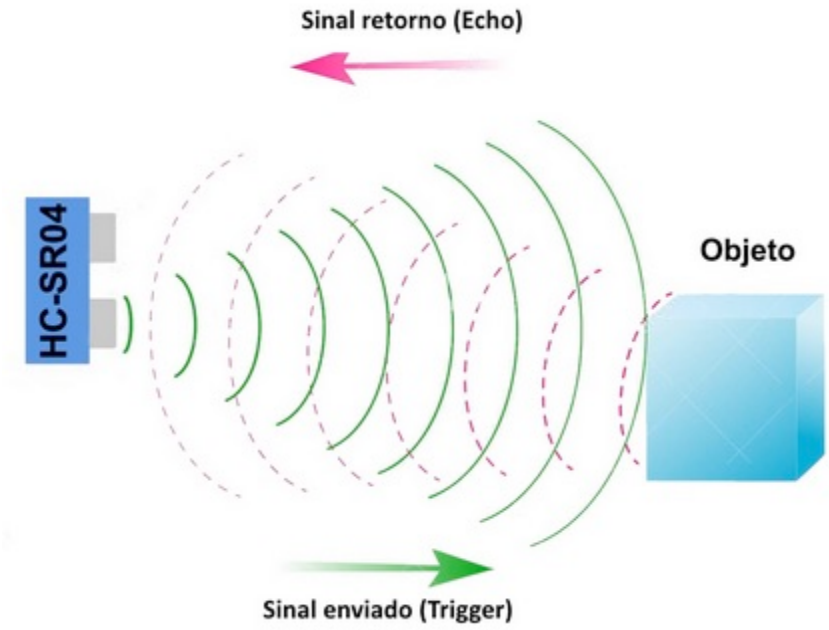

Figura 4: Esquema de funcionamento do sensor HCSR04.

Motor de corrente contínua

Motor elétrico é uma máquina destinada a transformar energia elétrica em mecânica. É o mais usado entre todos os tipos de motores, pois combinam as vantagens da energia elétrica baixo custo, facilidade de transporte, limpeza e simplicidade de comando com sua construção simples, custo reduzido, grande versatilidade de adaptação as cargas dos mais diversos tipos e melhores rendimentos.

Os motores de corrente contínua possuem grande versatilidade em seu controle da velocidade, que pode ser implementado de forma bastante simples ao se atuar no nível de tensão aplicada. Isto resultou, durante muito tempo, no uso preferencial destes motores para os processos de automação. Uma importante classe de motores de corrente contínua, os de ímãs permanentes, é amplamente utilizada em servomecanismos.

Pulse Width Modulation (PWM)

Refere-se ao conceito de pulsar rapidamente um sinal digital em um condutor. Entre inúmeras aplicações, esta técnica de modulação pode ser utilizada para simular uma tensão estática variável e é comumente aplicada no controle de motores elétricos, aquecedores, LEDs ou luzes em diferentes intensidades ou frequências.

Um dispositivo digital como um microcontrolador pode trabalhar com entradas e saídas que possuem apenas dois estados: ligado ou desligado. Assim, você pode facilmente usá-lo para controlar o estado de um LED por exemplo ligando ou desligando o mesmo. Da mesma forma que você pode usá-lo para controlar qualquer dispositivo elétrico usando dispositivos adequados (transistor, triac, relés etc).

No entanto, às vezes é necessário mais do que apenas “ligar” e “desligar” no controle de dispositivos. Caso você deseje controlar o brilho de um LED (ou qualquer lâmpada) ou a velocidade de um motor elétrico CA, simplesmente não será possível aplicando somente o controle (ligar/desligar). Para contornar esta situação, habilmente foi desenvolvida a técnica chamada PWM ou *Pulse Width Modulation*.

PWM é a técnica usada para gerar sinais analógicos de um dispositivo digital como um Microcontrolador e ela é tão eficiente que hoje em dia quase todos os Microcontroladores modernos possuem hardware dedicado para a geração de sinais PWM (BERTULUCCI, 2016).

Ponte H

Na maioria das abordagens em robótica é necessário a utilização de motores DC em diversos tipos de locomoção de robots, movimentação de braços mecânicos, etc. Os motores DC (Direct current ou corrente continua) são cargas indutivas que, em geral, demandam uma quantidade de corrente superior à que as portas do Arduino conseguem fornecer.

Sendo assim, não devemos ligar estes motores diretamente nas portas do Arduino pois se o motor demandar uma corrente acima de 40mA nas portas digitais (máxima fornecida pelo Arduino) pode queimar a porta e danificar a placa.

Para solucionar a questão da alta corrente poderíamos usar transístores, porém é importante que seja possível usando apenas um transístor já que para inverter o sentido de giro devemos inverter a polaridade da alimentação do motor (onde era positivo se pões negativo e vice versa). Um transístor só seria suficiente para ligar e desligar o motor.

Para resolver o problema utilizamos o famoso circuito conhecido como **Ponte H** que não é mais do que 4 transístores. Este circuito é uma elegante solução por ser

capaz de acionar simultaneamente dois motores controlado não apenas seus sentidos, como também as suas velocidades. Além do seu uso ser simples no Arduino (GUEDES, 2017).

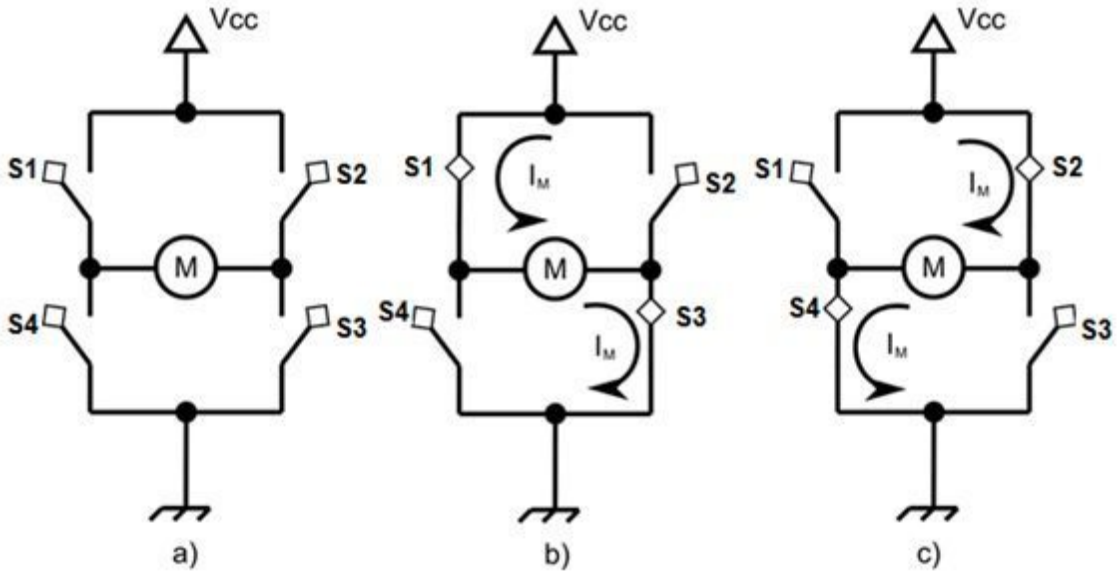

Figura 5: Circuito Ponte H.

5 DESCRIÇÃO DA PESQUISA

5.1 ESTUDO DAS REGRAS DAS COMPETIÇÕES DE ROBÓTICA E DESAFIOS QUE DEVERIAM SER SUPERADOS PELO PROTÓTIPO

Com base no manual é possível deduzir tudo que o robô protótipo precisa ter para se assemelhar com um robô de resgate real.

O ambiente deve ser simulado em um plano de cor branca cujo o caminho a ser percorrido seja uma linha preta de 1,5 cm de espessura. Está linha pode apresentar desvios (curvas) de até 90 graus, cujo o robô deve ser programado para identificar e executar a curva afim de permanecer sob a linha. Durante o trajeto pode acontecer a chamada “encruzilhada” ou “meia-encruzilhada”, existem várias possibilidades em que uma encruzilhada pode aparecer, mas no geral, todas as encruzilhadas ou meia-encruzilhadas que não apresentarem uma fita verde deve ser ignorada, ou seja, o robô deve seguir em frente, já as que apresentarem, o robô deve virar para o lado que é indicado pela fita.

O ambiente simulado também deve conter desafios “físicos”, problemas que não estejam contidos no plano branco e representados com fita, um dele é o redutor de velocidade, esse é representado por um cilindro de 1 cm de diâmetro que pode aparecer em qualquer linha reta do trajeto, ele serve para atrapalhar o robô. Há também obstáculos retangulares de dimensões específicas, ao encontrar um desses obstáculos, o robô deve contorna-lo saindo da linha do trajeto e após o contorno deve retornar a linha o mais breve possível. A área de resgate fica em um plano mais elevado, e para acessa-la é necessário subir um plano inclinado (rampa).

5.2 CLASSIFICAÇÃO E ESTUDO DOS COMPONENTES

Embora meu foco no projeto tenha sido a programação, é necessário entender os componentes afim de conhecer o funcionamento de cada um deles para poder produzir um código que atenda às necessidades do robô, os componentes selecionados para a construção do robô foram: Computador, Arduino/Genuino Mega 2560, Motor DC com caixa de redução 12V/44rpm, Sensor de linha TCRT5000, Sensor de cor TCS34725, Sensor Ultrassônico HC-SR04, Ponte H L298N, um chassi de madeira confeccionado pela própria equipe e rodas de 6 cm de diâmetro.

5.3 ANÁLISE ESTRATÉGICA DA POSIÇÃO DOS COMPONENTES NO PROTÓTIPO

Foram utilizados 5 sensores de linha, 4 localizados alinhados na parte frontal do robô, o que sobrou fica mais a frente entre os dois sensores centrais. Desse modo, os sensores da linha de 4 servirão para identificar os tipos de desvio/curva, encruzilhada, sendo os 2 sensores mais próximas do centro responsáveis principalmente pelo alinhamento, já o sensor da frente serve para ajudar a identificar as encruzilhadas e auxiliar o alinhamento;

Utilizamos também 3 sensores ultrassônicos onde 1 foi posicionado na frente e os outros dois nas laterais, de modo a ficar um de cada lado, o sensor ultrassônico frontal serve para identificar o obstáculo, já os laterais são responsáveis para orientar o contorno;

Para identificar as fitas de cor verde utilizamos 2 sensores de cor, cada um localizado atrás da linha de sensores de linha, de modo que fique um de cada lado.

5.4 DESENVOLVIMENTO DE ALGORITMOS PARA MOVIMENTAÇÃO

Para que o robô possa executar todas as ações com devida eficiência, é de extrema importância que ele tenha uma boa movimentação, essa movimentação pode ser dividida em 7 funções: andar para frente, andar para trás, parar de andar, virar para direita, virar para esquerda, girar para direita, girar para esquerda.

Com cada motor ligado a uma ponte H é possível saber em qual sentido e em qual velocidade o motor deve girar, e cada ponte H L298N pode se ligar a 2 motores, logo, para um robô de 4 rodas, é necessário utilizar duas pontes H L298N.

Desse modo, como cada pino digital do arduino tem capacidade de mandar uma tensão de 0V e 5V, é possível dizer qual o sentido o motor deve girar utilizando as funções digitalWrite e analogWrite. Desse modo em tenho cada função programada da seguinte forma:

```
digitalWrite (Pino do sentido contrário ao movimento, LOW);
analogWrite (Pino do sentido do movimento, valor PWM que representa a velocidade);
```

Dado que, cada motor vai estar indiretamente ligado a dois pinos, sendo cada um deles ligado a um dos polos do motor por intermédio da ponte H, temos:

```
Função andar para frente (velocidade) {
        digitalWrite(pino horário DIREITA FRENTE, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA FRENTE, LOW);
        analogWrite(pino horário ESQUERDA FRENTE, velocidade);
        digitalWrite(pino horário DIREITA TRÁS, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA TRÁS, LOW);
        analogWrite(pino horário ESQUERDA TRÁS, velocidade);
}
```

A função digitalWrite com os parâmetros pino anti-horário DIREITA FRENTE e LOW coloca o pino com tensão de 0 volts e a função analogWrite com parâmetros pino horário DIREITA FRENTE e velocidade, coloca o pino com tensão equivalente a o valor PWM da velocidade, se a tensão dada pelo PWM for maior que 0, uma corrente vai passar de um pino para o outro passando pelo motor e o fazendo girar. Na função acima temos os quatro motores girando de modo que o robô ande para frente. Seguindo esse padrão temos:

```
Função andar para trás (velocidade) {
        digitalWrite(pino anti-horário DIREITA FRENTE, LOW);
        analogWrite(pino horário DIREITA FRENTE, velocidade);
        digitalWrite(pino horário ESQUERDA FRENTE, LOW);
        analogWrite(pino anti-horário ESQUERDA FRENTE, velocidade);
        digitalWrite(pino anti-horário DIREITA TRÁS, LOW);
        analogWrite(pino horário DIREITA FRENTE, velocidade);
```

```
        digitalWrite(pino horário ESQUERDA TRÁS, LOW);
        analogWrite(pino anti-horário ESQUERDA TRÁS, velocidade);
}
Função parar (velocidade) {
        digitalWrite(pino horário DIREITA FRENTE, LOW);
        digitalWrite(pino anti-horário DIREITA FRENTE, LOW);
        digitalWrite(pino anti-horário ESQUERDA FRENTE, LOW);
        digitalWrite(pino horário ESQUERDA FRENTE, LOW);
        digitalWrite(pino horário DIREITA TRÁS, LOW);
        digitalWrite(pino anti-horário DIREITA TRÁS, LOW);
        digitalWrite(pino anti-horário ESQUERDA TRÁS, LOW);
        digitalWrite(pino horário ESQUERDA TRÁS, LOW);
}
```

No caso acima todos os pinos estão com tensão de 0 volts, ou seja, não há corrente, logo nenhum motor girará.

```
Função virar para direita (velocidade) {
        digitalWrite(pino horário DIREITA FRENTE, LOW);
        digitalWrite(pino anti-horário DIREITA FRENTE, LOW);
        digitalWrite(pino anti-horário ESQUERDA FRENTE, LOW);
        analogWrite(pino horário ESQUERDA FRENTE, velocidade);
        digitalWrite(pino horário DIREITA TRÁS, LOW);
        digitalWrite(pino anti-horário DIREITA TRÁS, LOW);
        digitalWrite(pino anti-horário ESQUERDA TRÁS, LOW);
        analogWrite(pino horário ESQUERDA TRÁS, velocidade);
}
```

Na função acima os motores do lado direito permanecem parados enquanto os do lado esquerdo giram de modo a irem para frente, resultando em uma virada para direita.

```
Função virar para esquerda (velocidade) {
        digitalWrite(pino horário DIREITA FRENTE, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA FRENTE, LOW);
        digitalWrite(pino horário ESQUERDA FRENTE, LOW);
```

```
        digitalWrite(pino horário DIREITA TRÁS, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA TRÁS, LOW);
        digitalWrite(pino horário ESQUERDA TRÁS, LOW);
}
```

Na função acima os motores do lado esquerdo permanecem parados enquanto os do lado direito giram de modo a irem para frente, resultando em uma virada para esquerda.

```
Função girar para direita (velocidade) {
        digitalWrite(pino anti-horário DIREITA FRENTE, LOW);
        analogWrite(pino horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA FRENTE, LOW);
        analogWrite(pino horário ESQUERDA FRENTE, velocidade);
        digitalWrite(pino anti-horário DIREITA TRÁS, LOW);
        analogWrite(pino horário DIREITA FRENTE, velocidade);
        digitalWrite(pino anti-horário ESQUERDA TRÁS, LOW);
        analogWrite(pino horário ESQUERDA TRÁS, velocidade);
}
```

Na função acima os motores do lado esquerdo giram para frente enquanto os do lado direito giram para trás, logo o robô vai girar para direita sobre seu próprio eixo.

```
Função girar para esquerda (velocidade) {
        digitalWrite(pino horário DIREITA FRENTE, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino horário ESQUERDA FRENTE, LOW);
        analogWrite(pino anti-horário ESQUERDA FRENTE, velocidade);
        digitalWrite(pino horário DIREITA TRÁS, LOW);
        analogWrite(pino anti-horário DIREITA FRENTE, velocidade);
        digitalWrite(pino horário ESQUERDA TRÁS, LOW);
        analogWrite(pino anti-horário ESQUERDA TRÁS, velocidade);
}
```

Na função acima os motores do lado esquerdo giram para trás enquanto os do lado direito giram para frente, logo o robô vai girar para esquerda sobre seu próprio eixo.

Para melhor controle do robô foi criada uma biblioteca PH.h (ANEXO II e III) e a programação das duas pontes H é feita separadamente:

```
void andarFrente() {
  ph1.goAway(vel);
  ph2.goAway(vel);
}
```

Ao invés de usar uma única função para descrever o movimento, usamos 2 funções separadas para descrever o comportamento de cada ponte H, mas de modo que ambas executem a mesma ação, no caso acima, “goAway” é o nome real da função andar para frente, ph1 e ph2 são as pontes H e “vel” é a velocidade descrita em uma variável global descrita no início do código. Já a programação dentro da biblioteca fica da seguinte maneira:

```
void PH::goAway(int speed) {
   digitalWrite(DA, LOW);
   analogWrite(DH, speed);
   digitalWrite(EA, LOW);
   analogWrite(EH, speed);
}
```

A função acima representa a programação feia na biblioteca PH.h, de modo que essa função aciona o movimento de dois motores para frente. No código principal ela é declarada da seguinte maneira:

```
//Pontes H para 4 motores
PH ph1(7, 4, 8, 9); //ponte H 1
PH ph2(6, 5, 10, 11); //ponte H 2
```

Os números passados como parâmetro representam respectivamente: pino do motor do lado direito anti-horário, pino do motor do lado direito horário, pino do motor do lado esquerdo horário e pino do motor do lado esquerdo anti-horário. O código principal e a biblioteca contendo todas as funções seguem no ANEXO I.

5.5 DESENVOLVIMENTO DE UM ALGORITMO SEGUIDOR DE LINHA UTILIZANDO SENSORES DE LINHA/INFRAVERMELHO

Para entender melhor o funcionamento do algoritmo seguidor de linha é importante observar a disposição dos sensores infravermelhos no chassi.

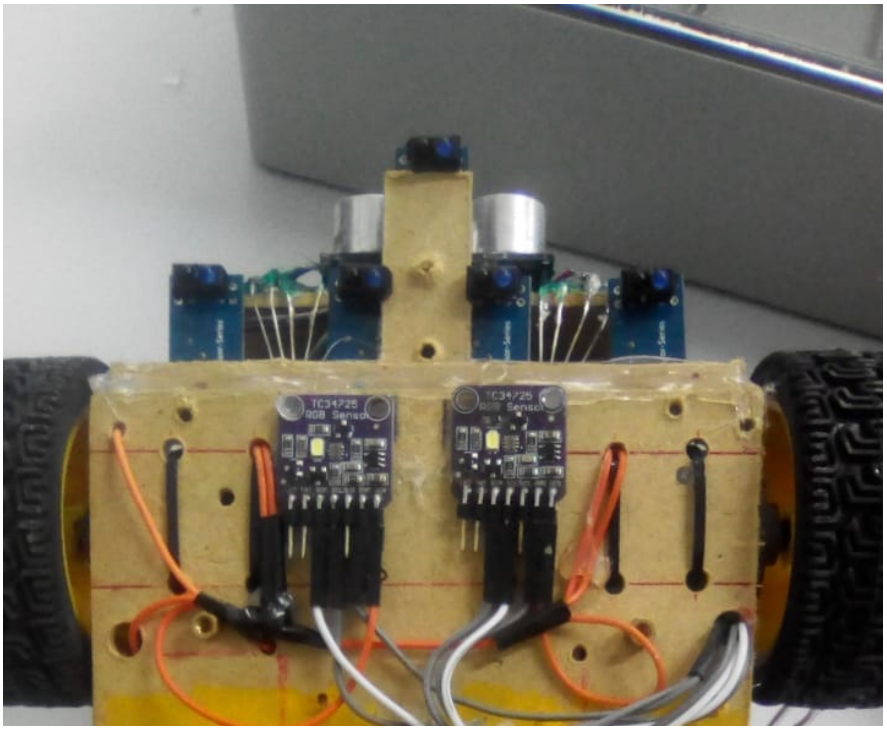

Figura 6: Disposição dos sensores de linha no protótipo.

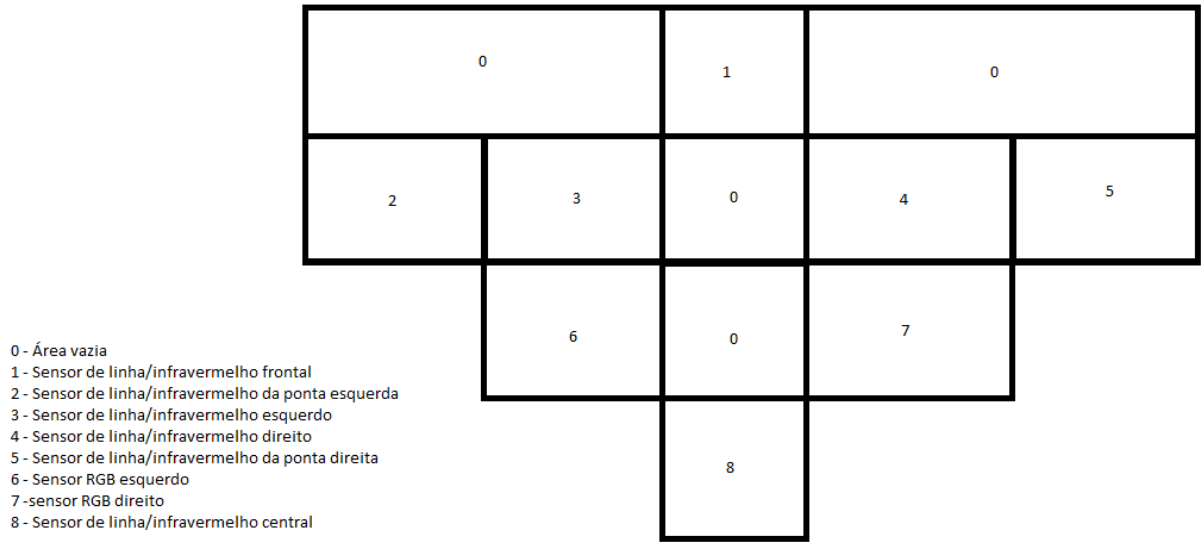

Figura 7: Esquema de posicionamento de sensores de linha e RGB.

O algoritmo seguidor de linha deve receber os valores dados pelas leituras dos sensores e interpretar a posição do robô em relação a linha preta, nesse caso é necessário conhecer todas as possibilidades de leitura e o que elas significam. Nas situações de encruzilhada, o algoritmo de linha dá o comando inicial de andar para frente, mas executa a verificação de cor (para saber se há fita verde), e a verificação de cor diz se haverá ou não mudança no trajeto.

Possibilidades de leitura:

PRETO	BRANCO	Situação	Resposta
1.	2, 3, 4 e 5.	Robô reto sobre linha reta.	Andar para frente.
3.	1, 2, 4, 5.	Leve curva para esquerda ou robô mais à direita da linha reta.	Girar levemente para esquerda até 1 ler preto.
2 e 3.	1, 4 e 5.	Curva de 90 graus a esquerda.	Girar para esquerda até 1 ler preto.
4.	1, 2, 3 e 5.	Leve curva para direita ou robô mais à esquerda da linha reta.	Girar levemente para direita até 1 ler preto.
4 e 5.	1, 2 e 3.	Curva de 90 graus a direita.	Girar para direita até 1 ler preto.
2.	1, 3, 4 e 5.	Robô muito a direita da linha.	Virar para esquerda até 1 ler preto.
5.	1, 2, 3 e 4.	Robô muito a esquerda da linha.	Virar para direita até 1 ler preto.
Nenhum.	1, 2, 3, 4 e 5.	Gap (ausência de linha).	Repetir última resposta.
1, 2, 3, 4 e 5.	Nenhum.	Encruzilhada.	Andar para frente e verificar cor.
1, 4 e 5.	2 e 3.	Meia encruzilhada para direita.	Andar para frente e verificar cor.

1, 2 e 3.	**4 e 5.**	**Meia encruzilhada para esquerda.**	**Andar para frente e verificar cor.**
2, 3, 4 e 5.	**1.**	**Meia encruzilhada para trás**	**Verificar cor.**

Os códigos para cada uma dessas possibilidades se encontram no ANEXO I.

5.6 DESENVOLVIMENTO DE UM ALGORITMO PARA DETECÇÃO E DESVIO DE OBSTÁCULOS UTILIZANDO SENSORES ULTRASSÔNICOS

Para desenvolver o algoritmo de desvio de obstáculos foi utilizada a biblioteca “ultrassonic.h”, e com ela é possível, controlar os sensores ultrassônicos através da função distanceRead(CM), e assim realizar leituras de distância em centímetros entre o sensor e a superfície mais próxima do sensor e que esteja na direção a qual ele esteja direcionado. Ou seja, se aparecer algum obstáculo na frente do robô, o sensor frontal vai captar as variações da distância devido a sua presença.

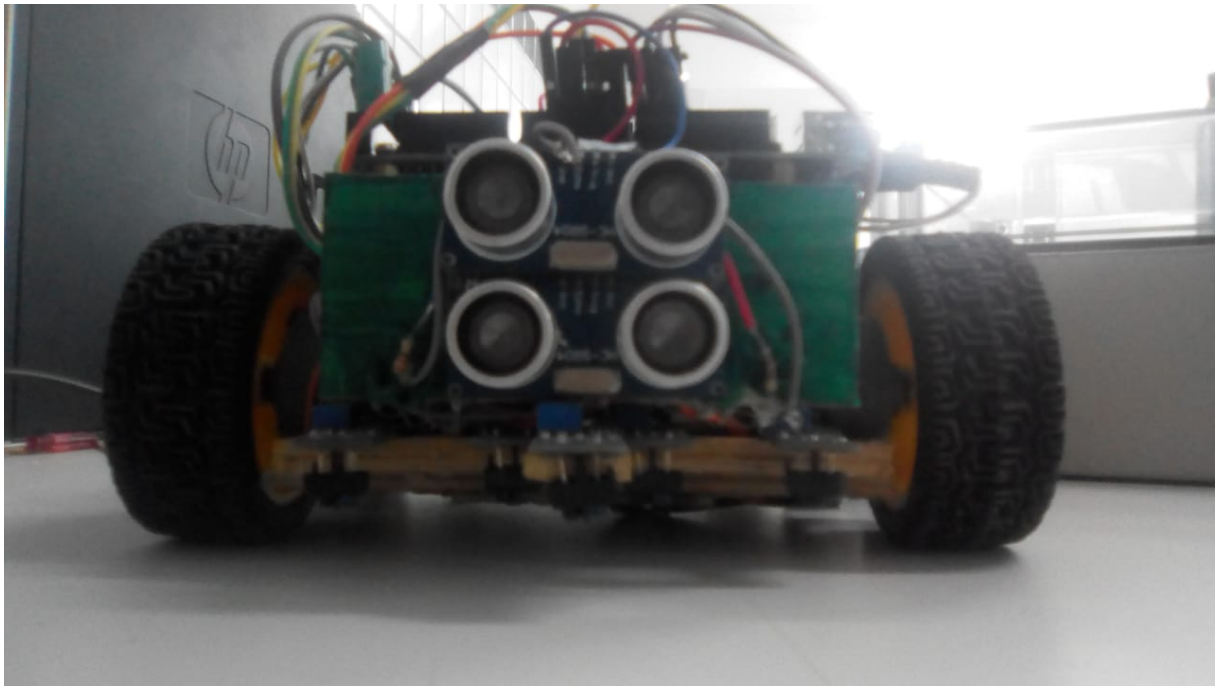

Figura 8: Disposição dos sensores ultrassônicos frontais no protótipo.

Como um obstáculo pode aparecer a qualquer momento, é necessário estar realizando leituras constantemente, no loop principal, antes de verificar o estado da linha e da cor, colocou-se um laço condicional para verificar se o sensor de distância frontal captou alguma variação que descreva um objeto se aproximando, para isso, é necessário estabelecer uma distância mínima, de modo que enquanto aquela distancia não for atingida, significa que não há obstáculo, ou que ele não está perto o suficiente para justificar uma resposta. Enquanto a distância mínima não é alcançada, o robô realiza suas outras funções normalmente, a partir do momento que o que a leitura retorna uma distância menor ou igual a estabelecida, o robô inicia a rotina de contorno.

Já a rotina de contorno inicia-se com a movimentação de giro do robô, fazendo

com que ele fique de lado a o obstáculo, a partir daí o sensor ultrassônico lateral vai coordenar o contorno até os sensores infravermelho detectarem linha novamente.

Essa rotina consiste em realizar o contorno em forma de arco, para isso utilizamos o seguinte algoritmo:

```
while (analogRead(sensorDireito) <= intermed) {
     if(sensorDistanciaEsquerdo.distanceRead(CM)>raioF*2||sensorDist
anciaEsquerdo.distanceRead(CM) == 0) {
          if (ultimaDirecao == 'F') {
               curvaEsquerda();
               ultimaDirecao = 'E';
          }
          else {
               andarFrente();
               ultimaDirecao = 'F';
          }
     }

}
```

Desse modo, enquanto o sensor de distância lateral estiver detectando o obstáculo, o robô vai andar para frente, quando não, o robô irá realizar um giro para o lado do obstáculo, e continuará o processo até achar a linha. O algoritmo completo está disponível no ANEXO I.

5.7 DESENVOLVIMENTO DE UM ALGORITMO DE IDENTIFICAÇÃO DE DIREÇÕES POR FITAS DE COR VERDE UTILIZANDO SENSORES RGB

Segundo as regras da Olimpíada Brasileira de Robótica, todas as encruzilhadas podem apresentar uma fita verde a qual indica uma necessária mudança de direção, conforme mostra a figura a seguir:

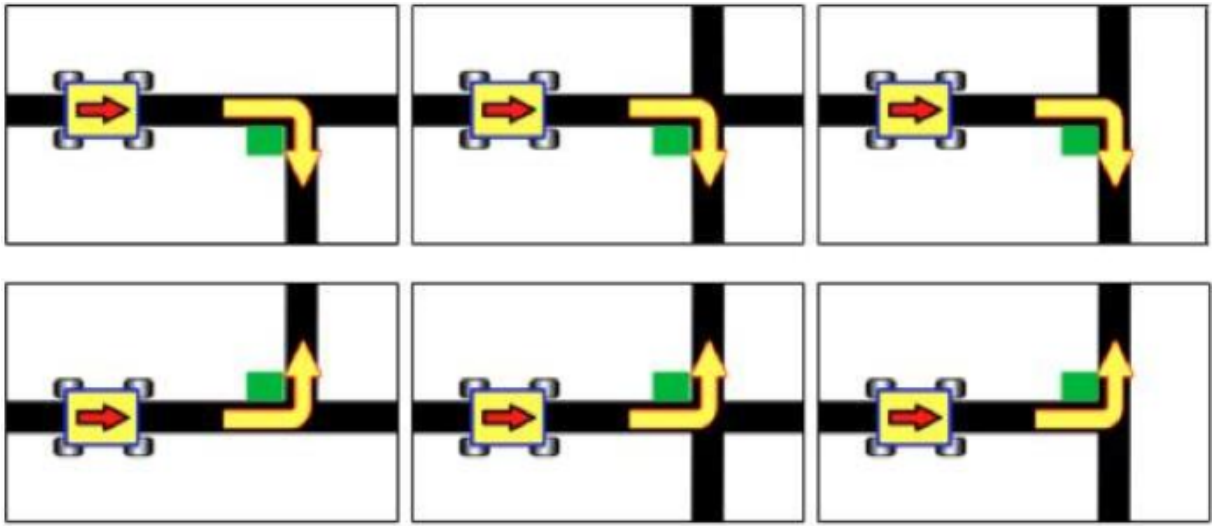

Figura 9: Encruzilhadas com fita verde. Fonte: Manual de regras e instruções OBR

Para identificar a presença dessas fitas verdes utilizamos os sensores RGB já mostrados no esquema da figura 1, pelos números 6 e 7. Para a utilização desse sensor é necessário o uso biblioteca fornecida pelo fabricante “Adafruit_TCS34725softi2c.h”, a partir dessa biblioteca, podemos usar a função “getRawData(&valorVermelho, &valorVerde, &valorAzul, &itensidadeLuminosa)” para realizar a leitura dos valores RGB. Com os valores salvos, basta fazer a comparação e verificar qual deles é o maior, ou seja, mais predominante no ambiente.

No entanto, seguindo essa maneira só é possível comparar as cores, verde, vermelho e azul, mas a pista das competições é composta apenas de preto, branco e verde. E para dificultar ainda mais, caso a leitura seja feita em um plano que não tenha nenhuma das 3 cores, ainda sim, o verde predomina.

Com isso, é necessário buscar outra forma de comparação, verificamos que a mais adequada é:

6 CONSIDERAÇÕES FINAIS

Atualmente, o protótipo realiza quase todas as funções desejadas, ele ainda apresenta algumas dificuldades na identificação das fitas verdes pelos sensores RGB e devido a diversos problemas não foi possível realizar a meta do algoritmo de resgate.

O Robô atualmente realiza funções de seguir linha e obstáculo e consegue transpor encruzilhadas e redutores de velocidade sem muita dificuldade.

O objetivo fundamental deste projeto foi desenvolver um protótipo capaz de realizar a tarefa de resgate de uma vítima percorrendo um caminho e sendo capaz de superar terrenos irregulares, transpor caminhos onde o trajeto inicial não possa ser reconhecido, desviar e subir terrenos elevados, resgatar a vítima e levá-la até um local seguro. Todos estes obstáculos serão representados por meio de objetos que fazem analogia a um incidente real.

Após uma gama de novos modelos e vários testes, diversas análises e adaptações, o desempenho do protótipo foi aprimorado em muito, isso é algo a se considerar por elevar os conhecimentos de mecanismos, mecânica, eletricidade, física, abre um leque de possibilidades que podem vir a mudar a história de diversas pessoas.

A pesquisa foi proveitosa e possibilitou uma enorme evolução do conhecimento de métodos e tecnologias.

6.1 TRABALHOS FUTUROS

Futuramente, pretende-se implementar melhorias na leitura das fitas verdes e implementar também o algoritmo de resgate. A partir deste trabalho podem ser desenvolvidos demasiados outros com outras variadas temáticas, isso é possível graças a versatilidade do arduino que permite que os conhecimentos tecnológicos utilizados nesse projeto possam ser utilizados em vários outros.

7 ANEXOS

7.1 ANEXO I

```
#include <Adafruit_TCS34725softi2c.h>
#include <Ultrasonic.h>
#include <PH.h>
#include <stdio.h>
//pinos
#define SDAD 29
#define SCLD 28
#define SDAE 30
#define SCLE 31
#define sensorFrente A2
#define sensorEsquerdo A1
#define sensorDireito A3
#define sensorEsquerdo2 A0
#define sensorDireito2 A4
#define sensorMeio A5
#define trig_F 42
#define echo_F 43
#define trig_D 49
#define echo_D 48
#define trig_E 47
#define echo_E 46
#define trig_FS 44
#define echo_FS 45
#define ledD 2
#define ledE 3
//variaveis globais
const int raioF = 6;
const int vel = 155;
const int cf = 10;
double timing = 0;
int PRETO;
```

7.2 ANEXO II

```
/*
 Name:    PH.h
 Created: 04/10/2018 15:40:05
 Author:  Gui Egle
 Editor:  http://www.visualmicro.com
*/

#ifndef _PH_h
#define _PH_h

#if defined(ARDUINO) && ARDUINO >= 100
  #include "arduino.h"
#else
  #include "WProgram.h"
#endif


#endif

class PH {
  public:
    int DH, DA, EH, EA;
    PH(int motorDH, int motorDA, int motorEH, int motorEA);
    void goAway(int speed);
    void comeBack(int speed);
    void turnRight(int speed);
    void curveRight(int speed);
    void turnLeft(int speed);
    void curveLeft(int speed);
    void stay();

};
```

7.3 ANEXO III

```
/*
 Name:    PH.cpp
 Created: 04/10/2018 15:40:05
 Author:  Gui Egle
 Editor:  http://www.visualmicro.com
*/

#include "PH.h"


PH::PH(int motorDH, int motorDA, int motorEH, int motorEA) {
  DH = motorDH;
  pinMode(DH, OUTPUT);
  DA = motorDA;
  pinMode(DA, OUTPUT);
  EH = motorEH;
  pinMode(EH, OUTPUT);
  EA = motorEA;
  pinMode(EA, OUTPUT);
}

void PH::goAway(int speed) {
  digitalWrite(DA, LOW);
  analogWrite(DH, speed);
  digitalWrite(EA, LOW);
  analogWrite(EH, speed);
}
void PH::comeBack(int speed) {
  digitalWrite(DH, LOW);
  analogWrite(DA, speed);
  digitalWrite(EH, LOW);
  analogWrite(EA, speed);
}
void PH::turnRight(int speed) {
  digitalWrite(DH, LOW);
  analogWrite(DA, speed);
  digitalWrite(EA, LOW);
  analogWrite(EH, speed);
}
void PH::turnLeft(int speed) {
  digitalWrite(DA, LOW);
  analogWrite(DH, speed);
  digitalWrite(EH, LOW);
  analogWrite(EA, speed);
}
void PH::curveRight(int speed) {
  digitalWrite(DH, LOW);
  digitalWrite(DA, LOW);
  digitalWrite(EA, LOW);
  analogWrite(EH, speed);
}
void PH::curveLeft(int speed) {
  digitalWrite(DA, LOW);
  analogWrite(DH, speed);
  digitalWrite(EH, LOW);
  digitalWrite(EA, LOW);
}
void PH::stay(){
```

REFERÊNCIAS

Alvarez, Luciana. Revista educação, ensino de programação é aposta de colégios em todo o mundo. Novembro de 2014. Disponível em <http://revistaeducacao.uol.com.br/textos/211/aposta-no-futuroo-ensino-de-programacao-tem-se-espalhado-como-330266-1.asp>. Acesso em 29 de junho de 2016.

MONK, S. 30 projetos com arduino. 2. ed. Porto Alegre: Bookman, 2014.

Olimpíada Brasileira de Robótica, Regras (Regionais e estaduais) – Versão 1. fevereiro de 2017.

Olimpíada Brasileira de Robótica. Manual de regras e instruções - Etapa regional/estadual. Versão 1. 0. março de 2018.

ARDUINO, 2010. Disponível em <http://www.arduino.cc>, Acesso em 12.12.2018

MORIMOTO, 2007. Disponível em <https://www.hardware.com.br/termos/software>, Acesso em 12.12.2018

FORMIGA, Diego Alves. Minicurso (componentes para construção de um robô seguidor de linha). Ceará-Mirim. Instituto Federal de ciência e tecnologia do Rio grande do Norte, 2018.

BERTULUCCI, Cristiano. “O que é PWM e Para que Serve?”. 2016. Disponível em: <https://www.citisystems.com.br/pwm/>. Acesso em: 12 dez. 2018.

GUEDES, Mariana. **Modulo Ponte H**. 2017. Disponível em: <https://www.arduinoportugal.pt/modulo-ponte-h-l298n-primeiro-passo-montar-robo-arduino/>. Acesso em: 12 dez. 2018.