PROTÓTIPO DE VEÍCULO GUIADO AUTOMATICAMENTE DE CARGA E DESCARGA

PROTOTYPE OF AUTOMATIC GUIDED VEHICLE OF LOADING AND UNLOADING

Jorge Luiz Santos Feitosa[1], Ricardo Rall[2]
[1]Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Botucatu, Jorge.feitosa@fatec.sp.gov.br.
[2]Prof. da Faculdade de Tecnologia de Botucatu, rrall@fatecbt.edu.br.

RESUMO

Devido as mudanças do cenário econômico das industrias brasileiras, existe a necessidade de encontrar meios para reduzir o custo de produção. Atualmente a atividade de carga e descarga nas indústrias é geralmente executada manualmente, por meio da força humana ou por veículos guiados por motoristas, além de ser um trabalho cansativo e repetitivo, colocando o trabalhador em situações de stress, que podem causar falhas de operação e até mesmo acidentes como perda da carga transportada, veículos danificados por choques mecânicos, atropelamentos, etc. Uma alternativa interessante para essa situação é a utilização de veículos guiados automaticamente (do inglês *Automatic Guided Vehicle* – AGV), que não necessitam de operadores, pois seguem percursos pré-definidos, por meio de marcações no solo e detectam obstáculos à sua frente, impedindo com isso choques mecânicos e atropelamentos, além de serem máquinas que trabalham de modo incesante, desde que sejam alimentadas com energia e mantidas periodicamente. A partir disso, o presente trabalho propôs a construção de um protótipo de um veículo guiado automaticamente por linhas, traçadas no solo, construído a partir de um chassi e controlado por um Arduino, sensores e *software* basicamente, os materiais empregados neste trabalho foram motores de corrente contínua alimentados com uma bateria de 9 volts, além de caixas de redução para aumentar o torque dos motores e circuitos que permitem variar a velocidade de rotação dos motores. Para detectar a presença da linha preta o robô foi equipado com emissores e receptores de luz na faixa infravermelho, baseados na reflexão da luz, também foram utilizados sensores ultrassônicos para detecção de obstáculo na frente do veículo e um sistema basculante para carregar e descarregar uma bola de tênis de mesa, em um local pré-definido, foi discutido também detalhes sobre a construção de uma base com sensores de RFID que irá identificar o veículo e carregar por gravidade de forma mecânica e descarregá-lo de forma automática. A programação foi feita utilizando a linguagem C com o ambiente de desenvolvimento Sketch, padrão do Arduino, e os esquemas elétricos da construção do projeto foram realizadas com o *software* Fritzing. Durante os testes o protótipo do veículo executou as tarefas programadas de carga e descarga e demonstrou que mesmo com material de baixo custo e uma escala menor desempenhou as funções que uma indústria necessita destas funções.

Palavras-chave: Arduino. Veículos guiados automaticamente. AGV. Protótipo seguidor de linha.

ABSTRACT

Due to the changes in the economic scenario of the Brazilian industries, there is a need to find ways to reduce the cost of production. Currently the loading and unloading activity in the industries is usually executed manually, through human force or vehicles guided by drivers, requiring a lot of labor, besides being a tiring and repetitive work, placing the worker in situations of stress, which can cause failures in operation and even accidents such as loss of transported cargo, vehicles damaged by mechanical shocks, run over, etc. An interesting alternative to this situation is the use of Automatic Guided Vehicles (AGV), which do not require operators, since they follow pre-defined routes, by means of markings on the ground and detect obstacles in front of them, thus preventing mechanical shocks and trampling, as well as being machines that work tirelessly, as long as they are supplied with energy and maintained periodically. From this, the present work proposed the construction of a prototype of a vehicle guided automatically by lines, traced in the ground, constructed from a chassis and controlled by an Arduino, sensors and software basically, the materials used in this work were engines of direct current powered by a 9-volt battery, plus gearboxes to increase motor torque and circuits that allow you to vary the speed of rotation of motors. To detect the presence of the black line the robot was equipped with emitters and receivers of light in the infrared range, based on light reflection, also used ultrasonic sensors for detection of obstacle in front of the vehicle and a tilting system to load and unload a ball of table tennis, at a pre-defined location, we also discussed details on building a base with RFID sensors that will identify the vehicle and load it mechanically and unload it automatically. The programming was done using the C language with the Sketch development environment, Arduino standard, and the electrical schematics of the project construction was carried out with the help of Fritzing software. During the tests the vehicle prototype performed the scheduled loading and unloading tasks and demonstrated that even with low-cost materials and a smaller scale it performed the functions that an industry requires of these functions.

Key words: Arduino. *Automatic Guided Vehicle. AGV. Line Follower Prototype.*

1 INTRODUÇÃO

No mundo globalizado, onde as mudanças e às adaptações as novas realidades são de vital importância para a sobrevivência das empresas e com a crescente concorrência no mercado de trabalho manifesta-se a necessidade de novas maneiras de se realizar trabalhos rotineiros, de forma a reduzir os custos de produção.

Entendendo essa necessidade e com o aumento crescente do custo logístico para realizar movimentações internas, existe uma demanda para automatização dos processos internos inerentes à linha de produção.

Para solucionar esse problema, com uma maior eficiência nas operações de movimentação, surgiu, o *Automatic Guided Vehicle* (AGV) ou veículos guiados automaticamente, sendo um dispositivo robótico que apresenta uma gama muito grande de oportunidades de aplicações que podem realizar a automatização de tarefas que envolvem a movimentação de cargas em geral, que atualmente é executada manualmente, por meio da força humana ou por veículos guiados por motoristas, exigindo muita mão-de-obra, além de ser um trabalho cansativo e repetitivo, colocando o trabalhador em situações de stress, que podem causar falhas de operação e até mesmo acidentes como perda da carga transportada, veículos danificados por choques mecânicos, atropelamentos, etc. (MCROBERTS, 2011).

O Arduino é uma plataforma de prototipagem eletrônica *open-souce*, baseados em *software* e *hardware* de licença livre, sendo flexíveis e fáceis de usar. O micro controlador é programado com uma linguagem de programação Arduino, baseado na linguagem *Wiring* e a interface de desenvolvimento do Arduino, é baseado no ambiente *Processing*. Os projetos desenvolvidos com Arduino podem ser autônomos ou podem se comunicar com outro computador para realizar outras tarefas, com o uso de *software* específicos (ex*: flash, Processing e MaxMSP*) (ARDUINO, 2015).

O objetivo deste trabalho foi a criação de um protótipo, em escala reduzida, de um veículo guiado automaticamente para carga e descarga de forma autônoma, que possa seguir marcações ou linhas no solo, utilizando um *hardware* e um *software* específico.

2 MATERIAL E MÉTODOS

Para a realização deste trabalho foram empregados dois motores de corrente contínua, alimentados por uma Bateria de 9 volts (v), um Chassi robótico, além de caixas de redução para aumentar o torque dos motores e circuitos que permitem variar a velocidade de rotação dos motores. Para detectar a presença da linha preta o veículo foi equipado com emissores e receptores de luz na faixa infravermelho baseados na reflexão da luz, também foi utilizado um

sensor ultrassônico para detecção de obstáculo na frente do veículo e um sistema de carga e descarga de uma bola de tênis de mesa, Para controlar tudo foi utilizado um micro controlador (controlador lógico programável – CLP) Arduino.

Para o módulo de carga do veículo seguidor de linha, foi criado uma estação para receber uma bola de tênis de mesa (simulando uma carga) que foi descarregado em cima do veículo na primeira estação e descarregada na segunda estação, isso se o mesmo tiver autorização para receber a carga por meio da leitura de uma *tag*, com tecnologia de identificação por rádio frequência (do inglês *Radio Frequency Identification* - RFID) fixada na lateral do veículo.

2.1 Arduino

O Arduino foi criado em 2005 com o intuito de ser um dispositivo de baixo custo, funcional com programação simples, com o principal objetivo facilitar o aprendizado de jovens estudantes e futuros projetistas. Segue a filosofia de *hardware* e *software* livre.

Esta placa foi criada com um microcontrolador Atmega328, com os circuitos de entrada e saída e de fácil conexão com uma IDE *(Integrated Development Environment)* por meio de um cabo USB, sua programação foi desenvolvida a partir do C e C++ (THOMSEN, 2014).

2.2 Módulo Ponte H

Segundo Fonseca (2016), este módulo permite controlar o sentido da rotação e velocidade *Pulse-Width Modulation* (PWN) de até dois motores em conjunto de corrente continua (DC) ou um motor de passo de 4 fases, podendo suportar um motor de até 12v.

2.3 Sensor Ultrassônico

O telêmetro ultrassônico, dispositivo que utiliza ultrassom para medir distâncias (som de frequência muito elevada, acima do limite da audição humana que é cerca de 20 kHz). O sensor ultrassônico está bem acima desse valor. Um pulso de som ultrassônico foi emitido pelo dispositivo a partir de um transdutor e recebido de volta, depois de refletir em um objeto pode-se calcular o tempo que leva para que o pulso retorne (MCROBERTS, 2011, p.325).

O sensor escolhido foi o HC- SR04. Ele possui baixo custo e vale ressaltar algumas de suas especificações retiradas de seu *datasheet* (ELECFREAKS ,2017), possui tensão de operação de 5 V, consumo típico de corrente de 2 *Milliamperes* (mA), Ângulo de atuação de 15 graus (°), sua faixa de atuação e de 2 cm – 500 cm e possui uma precisão de 0,3 cm.

2.4 Sensor Óptico Reflexivo Fototransistor

O Sensor TRTC5000 é um sensor óptico que segundo Branco (2011) é um dispositivo que converte os raios de luz para sinais eletrônicos. Semelhante a um foto resistor, que quantifica a luz de um ambiente e o transforma em dados.

2.5 Servo motor

O Servo motor é um atuador eletromecânico muito utilizado para posicionar e manter um objeto em uma determinada posição. Para realizar essa operação, ele conta com um circuito que compara o sinal de entrada e verifica com a posição atual do eixo, sua movimentação pode varia em um ângulo de 0° a 180° graus Ferraz (2008), no projeto foi utilizado um Servo Motor Tower Pro Sg90.

2.6 RFID

A tecnologia RFID é responsável por utilizar uma comunicação por radiofrequência, para transmitir dados de um dispositivo móvel, como uma simples *Tag* (que podem ser chaveiros ou cartões), para um leitor. As etiquetas RFID são *hardwares* que possuem uma antena e um chip envoltos por algum material, como vidro ou plástico, os quais respondem a sinais remotos de um leitor geralmente conectado a um computador (SANTINI, 2008).

2.7 Sensor LDR (Light Dependent Resistor)

A estrutura do sensor LDRs é formada por fótons que interagem com os elétrons dentro do material. Na absorção de fóton o mesmo liberto elétrons, ou seja, a absorção de fótons aumenta o número de portadores de carga do material ou muda a sua mobilidade. Como os portadores de carga ficam soltos, a resistência do material diminui impedindo a passagem da corrente (THOMAZINI, 2005).

2.8 Montagem do *Hardware* do veículo

Primeiramente, foi necessário utilizar uma base de robô pequena e leve. Um kit chassi robótico contendo um chassi em acrílico com dimensões de 22 x 14,7cm, dois motores DC (3~6v) com redução de 1:48, duas rodas de borracha 7 x 7 x 2,6 cm, uma roda boba (universal), jogo de parafusos e um suporte para pilhas AA.

Na segunda fase do projeto, também foram utilizados dois microcontroladores Arduino Uno R3 Rev3 Atmega328, uma *Shield* de motor driver L9110S, uma mini *protoboard*, vinte *jumpers* machos, vinte *jumpers* fêmeas, dois módulos seguidor de linha TRCT5000.

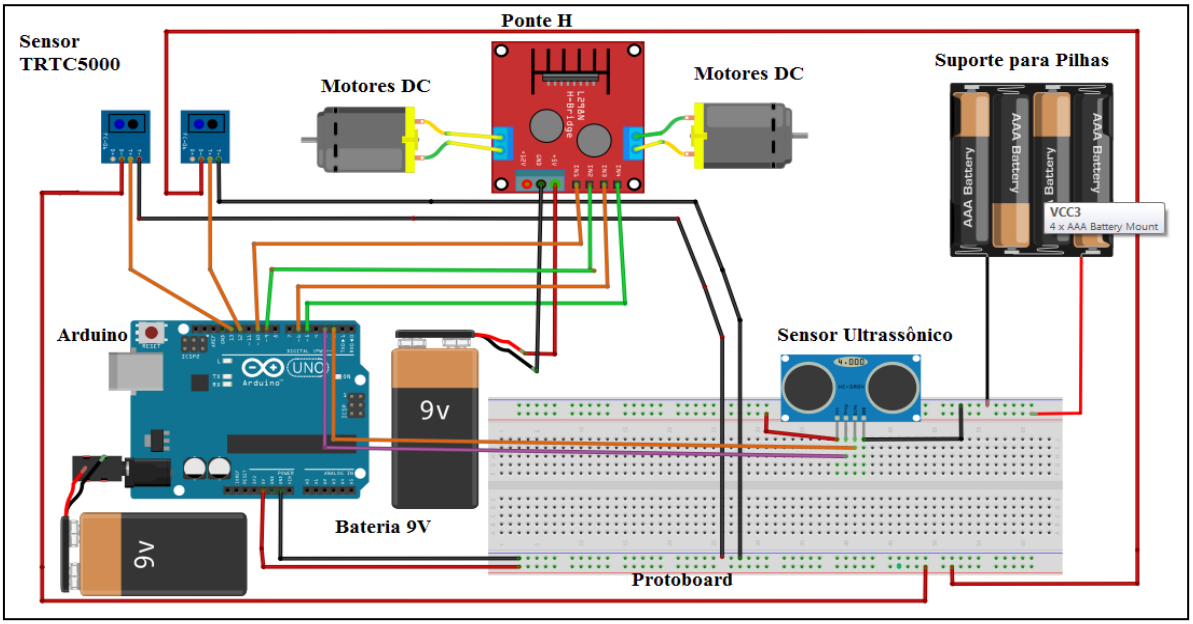
Na terceira fase, foi montado o chassi robótico, conforme instruções do fabricante, unindo-se a roda boba à estrutura de acrílico e os motores com a redução. Em seguida, instalou-se a Shields L9110S, com um parafuso, porca sobressalentes ou braçadeiras e fixado na estrutura do chassi.

Por meio de parafusos com porcas, os sensores TRCT5000 foram fixados na parte dianteira do chassi robótico. Na fixação do Arduino, utilizou-se parafusos ou abraçadeiras no chassi de acrílico.

Os cabos dos motores da Ponte H foram conectados com cada par de cabos em um dos Bornes do tipo Kre.

Os jumpers foram conectados às portas GND e 3.3V do primeiro Arduino na *protoboard*, a alimentação de 5V foi realizado de forma externa com o auxílio de quatro pilhas AA ligada nos dois trilhos de alimentação da *protoboard*. Os módulos TCT5000, foi feito nos trilhos GND e 3.3V da *protoboard*, respectivamente como mostrado na Figura 1. O esquema foi desenvolvido utilizando o *software* gratuito Fritzing.

Figura 1. Esquema elétrico do veículo



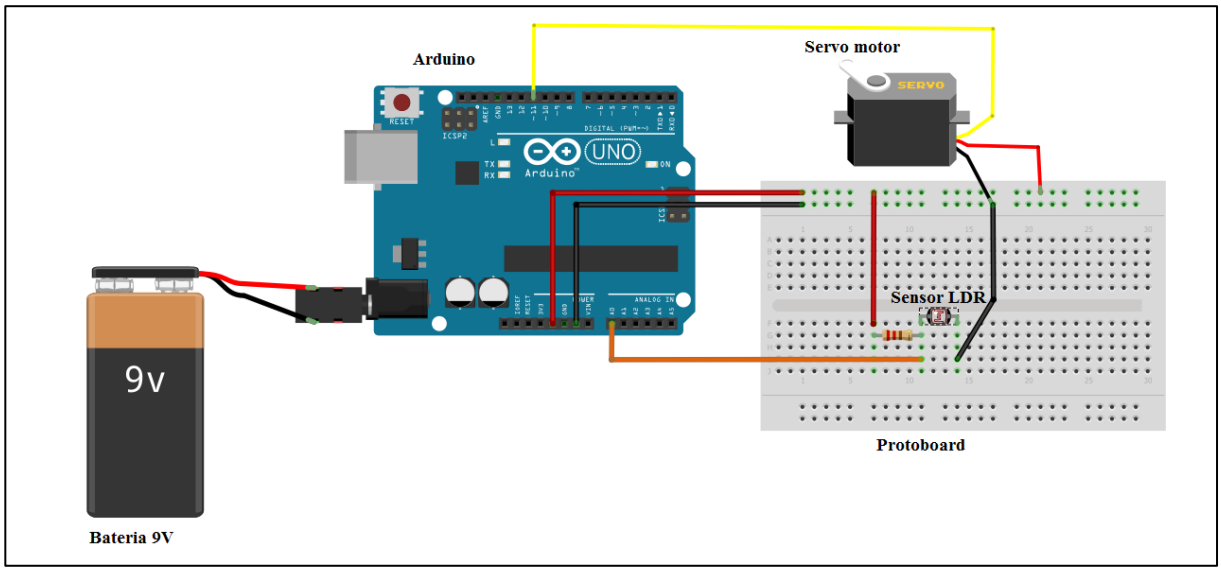
A conexão dos pinos “DO” dos módulos TRCT5000 foi realizada nas portas 12 e 13 do Arduino 1 (presente no veículo), sendo que cada um dos módulos conectados em cada uma das portas, se os cabos não chegarem próximo do Arduino 1, deverá ser utilizada a *protoboard* como extensor.

A conexão dos pinos VCC e GND da Ponte H (L9110) foi feita no pino positivo e negativo da bateria de 9V, na sequência indicada, a conexão dos demais pinos (da esquerda para a direita) foi realizada nas portas 5, 6, 9 e 10 do Arduino 1, ilustrada na Figura 1.

A fixação do sensor ultrassônico modelo HC-SR04 foi realizada na própria *protoboard* e alimentado com uma voltagem de 5V na trilha do pino VCC, o GND do sensor conectado na trilha GND da *protoboard*, o pino ECHO da porta 2 e o pino TRIG na porta 3 do Arduino 1.

No segundo Arduino (chamado de Arduino 2), foi realizada a montagem do sensor LDR, utilizando a entrada analógica do Arduino na porta A0 e alimentada com uma voltagem de 5V no trilho da *protoboard*, também se tornou necessária a utilização de um resistor 1.000 ohms (Ω) para diminuir a voltagem no sensor LDR. O Servo Motor Tower Pro Sg90 foi alimentado com uma tensão de 5V extraída da trilha da *protoboard*, assim com o seu aterramento e conectado na porta 11 do Arduino, como mostra a Figura 2.

Figura 2. Esquema elétrico Arduino 2

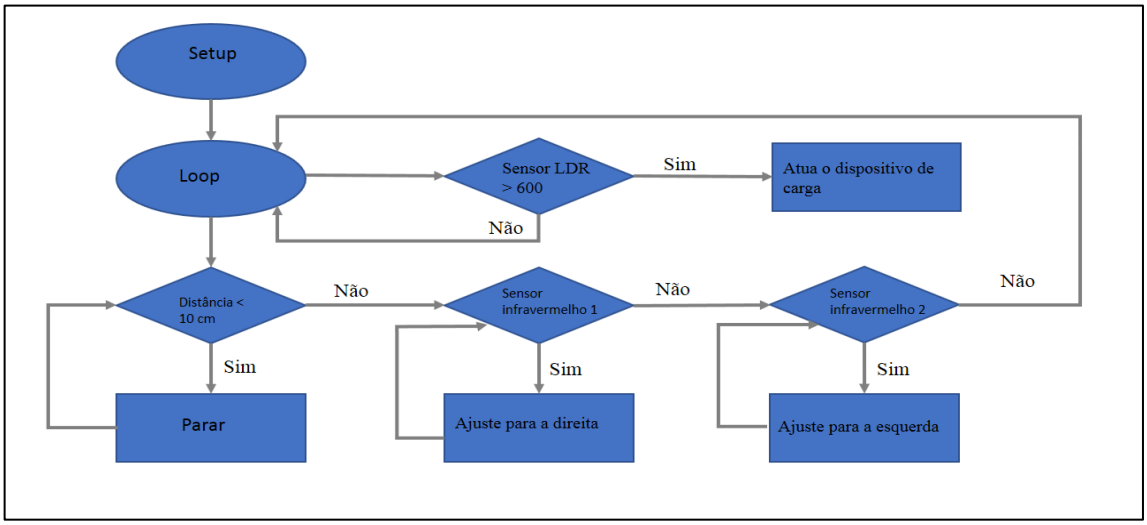


2.9 Desenvolvimento do veículo

A programação desenvolvida utilizou o ambiente de desenvolvimento do Arduino, gerando um código conhecido como *Sketch*. A Figura 3 representa o fluxograma da lógica de programação do Veículo Seguidor de linha.

Realizando uma análise do código, observou-se, basicamente, que existe uma verificação se o sensor ultrassônico está detectando algum obstáculo em uma distância de dez centímetros (10 cm), se essa condição for verdadeira, ela para a movimentação dos motores ou continua com a execução do código.

Figura 3. Fluxograma da lógica



Logo após a verificação da condicional do sensor ultrassônico, foram testadas as condições dos sensores infravermelho, onde foi verificada a presença de luz que é refletida da pista. A presença de deflexão de luz entende que os sensores estão em cima da parte branca da pista, que faz com que os sensores retornem “verdadeiro” (*true*). Se os sensores detectarem a ausência de luz da pista, significa que foi detectado a presença da faixa preta, então foi retornado o valor “falso” (*false*) e executada a correção do curso do veículo. Esse ciclo se repete uma vez em cada sensor e, após o término, o *loop* se repete novamente.

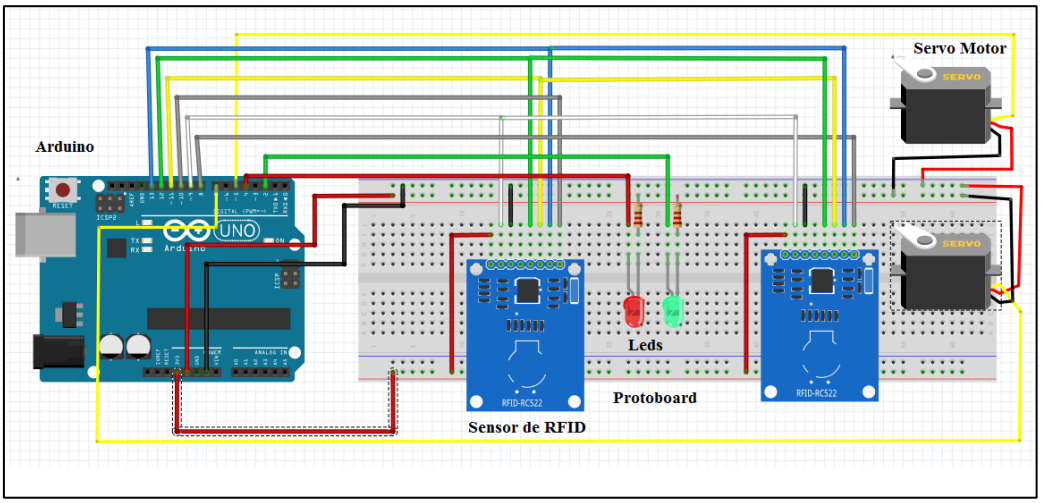
Existe uma verificação se o sensor LDR fez leitura de valores maiores que 600 e dependendo do valor foi acionado o servo que faz com que a bola de tênis de mesa seja descarregada do veículo, o start para a luz da *Station* se deu a partir da leitura da *Tag* RFID acoplada da lateral do veículo.

2.10 Desenvolvimento da *Station*

O desenvolvimento da Station se resume em duas parte, a primeira parte foi responsável por fazer a leitura da Tag RFID e executar uma verificação em um Array para saber se a Tag está cadastrada, se a Tag não estiver no Array a cancela não irá abrir e o veículo fará a leitura de um obstáculo na sua frente e não continua a trajetória, porem se a leitura da Tag RFID estiver cadastrada no Array a cancela irá abrir, deixando o veículo passar e continuar a sua trajetória, um pouco mais a frente, por meio de uma estrutura mecânica, a bola de tênis de mesa será descarregada em cima de uma dispositivo em forma de cone que conduzirá a bola de tênis de mesa até uma mini estrutura basculante.

Na segunda parte da Station há um outro sensor de RFID que irá realizar a mesma verificação no acesso da primeira Station, se o acesso estiver liberado a cancela irá se abrir e disparar uma fonte de luz, que será detectada pelo sensor LDR do veículo seguidor de linha e acionará o dispositivo basculante, que por sua vez descarregará da bola de tênis de mesa do veículo, conforme mostra a Figura 4.

Figura 4. Esquema elétrico *Station*



3. RESULTADOS E DISCUSSÃO

3. 1 Programação do veículo

A programação foi baseada em duas funções principais, a função com o nome de setup() e outra função com o nome de loop(). Existem outras funções, porém essas foram as que mereceram maior atenção, a linguagem também pode compor outras funções criadas pelo programador. A função setup() foi a primeira função a ser inicializada, e foi executada apenas uma vez durante a inicialização. Foi nessa função que foram configurados os pinos. A função loop() foi a segunda função a ser executada, nela as instruções são executadas repetidamente. Foi no loop que foram instanciadas as funções de leitura de pinos, atuação dos pinos e porta serial, também foi a parte do código onde foi programada toda a lógica do processo.

Para a programação do Arduino 1, primeiramente deve-se baixar a biblioteca “NewPing.h” e a biblioteca “Servo.h”, A biblioteca “NewPing.h” foi utilizada pelo sensor ultrassônico para leitura da distância de objetos e deve-se adicioná-la ao código, a biblioteca “Servo.h” foi aplicada para facilitar a utilização de servomotores.

Na porta 3 foi declarada uma variável TRIGGER_PIN que utilizou-se como o valor de gatilho de disparo do sinal, e foi executado em um determinado período de tempo. Na porta 2

foi declarada a variável que utilizou-se como ouvinte do sinal disparado pela trigger que foi chamada de ECHO_PIN e uma constante MAX_DISTANCE que recebeu o valor máximo em cm que foi capturada pelo sensor ultrassônico, tudo que for detectado acima desse valor foi desconsiderado, conforme mostra a Figura 5.

Figura 5. Trecho do código do Arduino 1

```
void setup() {
   Serial.begin(115200); // Abra o monitor serial em 115200 baud para ver os resultados do ping.
   pingTimer = millis();

   pinMode(AIA, OUTPUT); // Colocando os pinos como saída
   pinMode(AIB, OUTPUT);
   pinMode(BIA, OUTPUT);
   pinMode(BIB, OUTPUT);
}
void loop() {
   ultrassonico();
   seguidor();
}
void ultrassonico(){
  //Observe como não há atrasos neste esboço para permitir que você faça outro processamento enquanto faz pings de distância.
  if (millis() >= pingTimer) {     //pingSpeed milissegundos desde o último ping
    pingTimer += pingSpeed;        //Defina o próximo espaço de tempo
    sonar.ping_timer(echoCheck);   //Envie o ping, chama a função "echoCheck" cada 24uS onde você pode verificar o status do ping
  }
}
void echoCheck() { // A interrupção Timer2 chama essa função a cada 24uS, onde você pode verificar o status do ping.
  if (sonar.check_timer()) { // É assim que você verifica se o ping foi recebido.
    Serial.print("Ping: ");
    int valor = sonar.ping_result / US_ROUNDTRIP_CM;//converte em CM com US_ROUNDTRIP_CM.
    Serial.print(valor);
    Serial.println("cm");
  }
}
```

Precisou-se também declarar os portas da ponte h, nas portas PWM 10 com o nome de AIA, na porta 9 com o nome de AIB, porta 6 com o nome BIA e porta 5 com o nome de BIB, todas foram declaradas como variáveis do tipo inteiro.

Para alterar a velocidade dos motores foram declaradas constantes que podem ser modificadas, para ajustes da saída de tensão que forem enviada para Ponte h, esses valores podem ser modificados entre 0-255, porem o motor de corrente contínua (CC) só partirá da inercia a partir de valores acima de 120. Os sensores infravermelhos foram declarados nss portas 12 para o sensor 1 e porta 13 para o sensor 2.

Foi declarada a variável unsigned int pingSpeed = 50; que serviu como um incremento da variável pingTimer. A variável unsigned long pingTimer foi usada para executar uma verificação com o método *millis()* do Arduino, Realizando uma analogia da variável *long*, percebeu-se que ela suporta 4.294.967.295 milissegundos o que seria igual a 4.294.967 segundos, transformando em minutos teríamos 71.582 minutos, 1.193 horas e 49 dias, ou seja ela levaria 49 dias para estourar sem ser reiniciado MCROBERTS (2011).

Para que se tenha a falsa impressão de se realizar várias tarefas com o Arduino (lembrando que o Arduino trabalha de forma procedural), será necessário criar várias funções e fazer a instância das mesmas dentro da função Loop como, é importante ressaltar que não se deve utilizar a função *delay()*, pois a estrutura descrita trabalha em conjunto com a função *millis()*, que não para a execução do código por meio de uma estrutura de condição de paradas do sistema.

A variável pingTimer recebeu então a função *millis()* ilustrada na Figura 5, que foi uma função integrada à linguagem do Arduino, que retorna o número de milissegundos desde que o Arduino iniciou a execução do programa atual, existe uma estrutura condicional onde foi verificada se a função *millis()* foi maior ou diferente da variável pingTimer, essa função foi onde ocorre a verificação se o tempo que está dentro da função *millis()* foi maior que o tempo de pingTimer, se a condição for verdadeira entraremos dentro da estrutura condicional e a primeira instrução foi para acrescentar a variável pingTimer mais 50 milissegundos, a segunda instrução foi para chamar a função sonar.ping_timer(echoCheck) que foi executada a cada 50 milissegundos, essa função chama outra que está sendo passado por parâmetro, no caso a echoCheck que retornara a distância detectada pelos sensores, se essa condição for falsa ela não executa o que está dentro da sua estrutura e continua com o código, evitando dessa forma atrasos, dando a falsa impressão que as funções está ocorrendo de forma simultânea, uma vez que não necessitamos da função *delay()* para esperar o retorno da resposta dos sensores.

Logo após foi executado uma outra verificação condicional, que compara se a distância detectada pelo sensor ultrassónico foi maior que 10 cm, essa condição sendo verdadeira, foi feita a verificação de mais duas estruturas dentro dela. A primeira faz uma verificação do valor detectado pelo sensor infravermelho com a função !digitalRead(sensor2) e passando por parâmetros os valores das portas declarados no sensor2, se essa condição for falso significa que o veículo necessita de correção e a variável speed2 recebe o valor declarado na variável frente que no caso foi 170, porem a variável speedv2 recebe o valor 0, fazendo com a alimentação no motor 1seja suspensa e o veículo sofra uma correção. A segunda verificação executara a mesma lógica, porem a função digitalRead() recebeu a variável sensor1, também existe uma inversão nos valores das variáveis speed2 que recebeu 0 e speedv2 que recebeu o valor 170, como ilustra a Figura 6.

Figura 6. Código Seguidor de linha

```
//--------------------------------------------------------------------------
//                Função Seguidor de linhas
//--------------------------------------------------------------------------
void seguidor(){

  int valor = sonar.ping_result / US_ROUNDTRIP_CM;// Ping retornou, você resulta em ping_result, converte em cm com US_ROUNDTRIP_CM.
  if (valor > 10) {

    if (!digitalRead(sensor2)) {
      speed2 = frente;
      speedv2 = 0;
    } else {
      speed2 = 0;
      speedv2 = voltar;
    }
    if (!digitalRead(sensor1)) {
      speed1 = frente;
      speedv1 = 0;
    } else {
      speed1 = 0;
      speedv1 = voltar;
    }
  } else {
    parada();

  }
  direcaoroda1();
  direcaoroda2();
}
```

Caso a condicional da variável “valor” seja menor que dez centímetros, os valores das variáveis speed1, speedv1, speed2 e speedv2 receberam o valor zero, ao final da condicional foi chamada a função direcaoroda1() e direcaoroda2() para continuar com a alimentação nas duas rodas.

3.2 Programação da *Station*

A programação da *Station* foi construída com base na biblioteca “MFRC522.h” onde trabalha com dois leitores RFID’s ligados em série, entretanto apenas as portas SDA do leitor RFID foram ligados separados, a Figura 4 apresenta o esquema elétrico do projeto da *Station*, também foi utilizada a biblioteca “Servo.h” para facilitar o desenvolvimento da programação dos servos.

A programação padrão da biblioteca “MFRC522.h” necessitaou que fosse realizada duas instâncias para o funcionamento de cada sensor RFID, que se chamou de “mfrc1 (PIN1, RST)” e “mfrc2 (PIN2, RST)” e assim passou-se como parâmetros as portas SDA de cada leitor RFID e o RTS ligado em série entre cada sensor RFID, também se tornou necessário criar um *Array* com a todas as *Tag* que deseja-se ter o acesso liberado no sistema. Dentro da função Loop existe uma verificação para com as funções “mfrc1.PICC_IsNewCardPresent()” que verifica a existência de uma *Tag* no leitor e “mfrc1.PICC_ReadCardSerial()” que executa a leitura da *Tag*.

Após realizar a verificação existem funções que fizeram a construção do conteúdo de cada cartão ou Tag RFID e a transformou de bits para uma *String*, representando o valor de cada *Tag*, o código que ditou as regras de negócio é mostrado nas Figura 7.

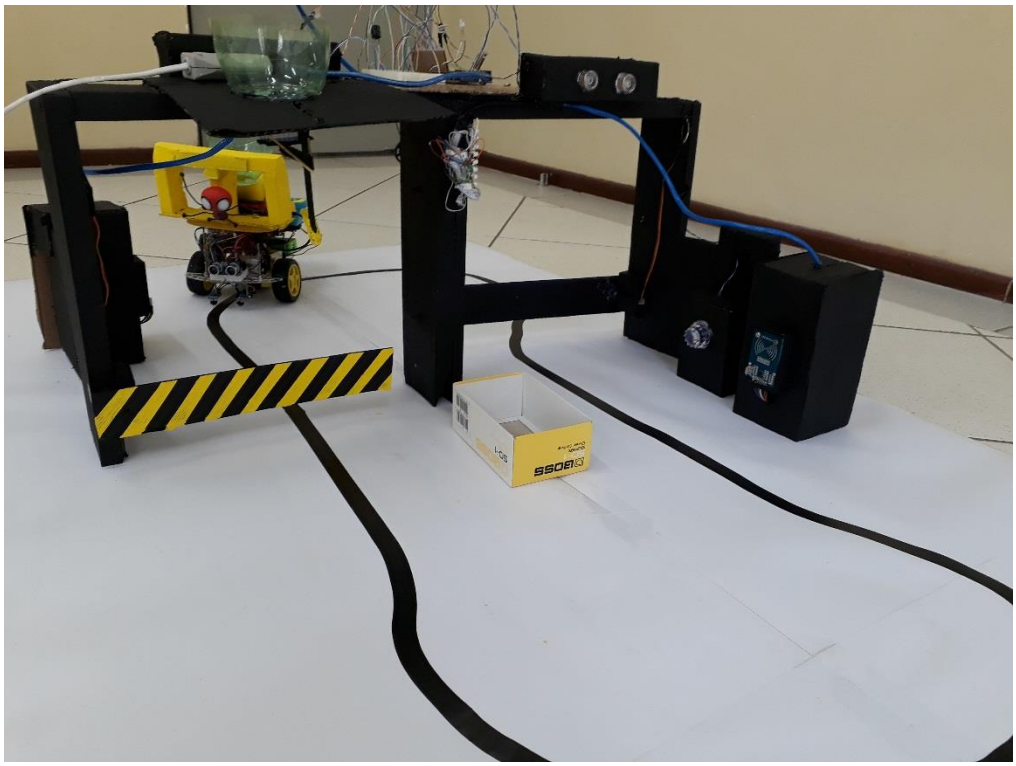
Figura 7. Regra de negócio *Station*

```
  for (int i = 0; i < 4 ; i++){
    if (conteudo.substring(1) == itens1[i]){
    //codigo entra aqui true 2
    Serial.println("Leitura Station 2, Acesso Liberado! ");
    digitalWrite(led_liberado, HIGH);

    microservo6.write(180);

    for (int i= 1; i<5 ; i++) {
    //Ligando o buzzer com uma frequencia de 1500 hz, avisando que a cancela vai abaixar
    tone(buzzer,1500);
    delay(1000);
    //Desligando o buzzer.
    noTone(buzzer);
    delay(1000);
    }

    digitalWrite(led_liberado, LOW);
    microservo6.write(90);

    break;
  }else if(i == 3){
   //codigo entra aqui false 1;
    Serial.println("Leitura Station 2, Acesso Negado!");
    digitalWrite(led_negado, HIGH);
    tone(buzzer,800);
    delay(500);
    digitalWrite(led_negado, LOW);
    noTone(buzzer);
  }
 }
}
```

A Figura 8 representa a performance do veículo com todos os componentes descritos no trabalho funcionando em conjunto, de forma que realize o que foi proposto no início do trabalho, onde o veículo precisa seguir uma linha fixada no solo e seguir o trajeto até a primeira estação, onde irá verificar se o valor da *Tag* RFID fixada no veículo possui acesso para continuar com a trajetória, com acesso liberado o veículo continuará sua trajetória até a segunda estação que irá verificar se possui acesso para descarregar a carga de forma automática.

Lembrando que o veículo está equipado com sensor ultrassônico que verifica se existe um obstáculo na pista, se a condição for verdadeira ele interrompe sua trajetória, simulando um AVG real na vida real.

Figura 8. Performance do veículo



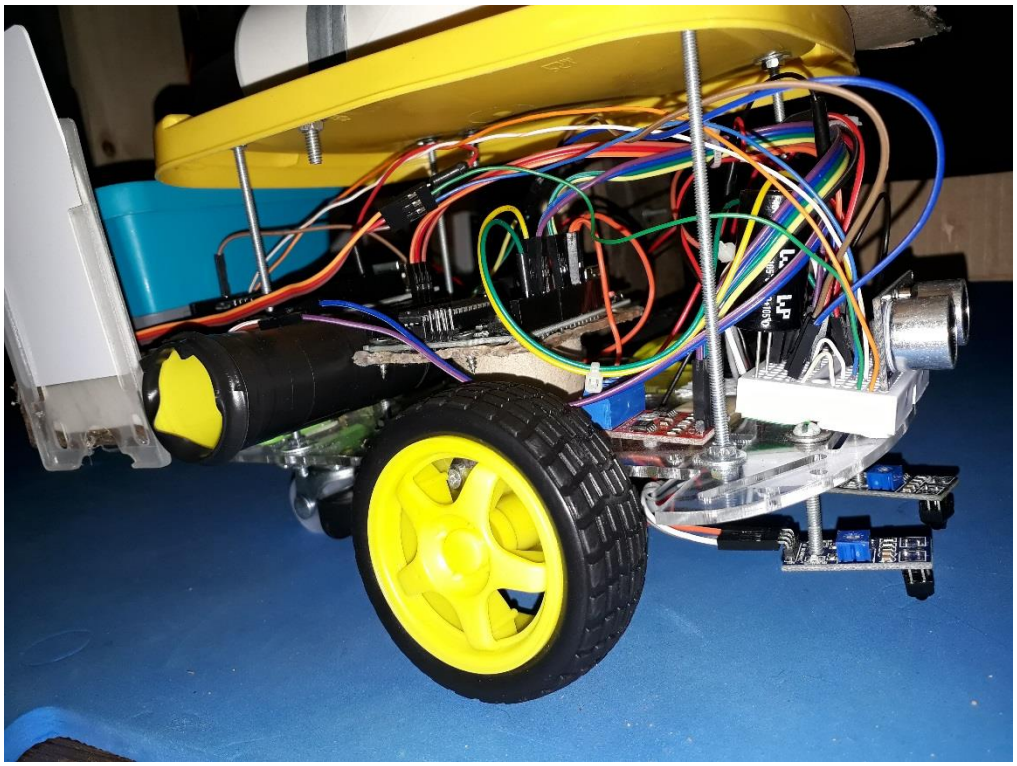
Durante a realização dos testes, é preciso regular os dois sensores infravermelho (TRTC5000) de forma independente de acordo com a distância que os mesmos se encontram do solo, uma distância ideal é de 20 milímetros do chão, também o sensor LDR pode variar o retorno das informações de acordo com o fabricante.

Na construção da base dos sensores RFIDs é preciso alinhar a distância que eles se encontram em relação a linha demarcada no solo, esses valores podem variar de acordo com a distância que a *Tag* se encontra no veículo.

Durante a construção do projeto é importante distribuir o peso entre as duas rodas e não deixar que o centro de gravidade do veículo (CG) fique na traseira, isso pode deixar o veículo sem tração das rodas dianteira e fazer com que o veículo levante a dianteira nas curvas, uma vez que é apoiado sobre três base (duas rodas dianteira e uma roda boba na traseira).

A Figura 9 ilustra a regulagem de alguns dos sensores e alinhamento do sensores Infravermelhos em relação ao solo.

Figura 9. Realização dos testes do veículo



4 CONCLUSÕES

O Arduino Uno se mostrou muito capaz e eficiente para a realização deste projeto, com agilidade, flexibilidade e alto desempenho, o comportamento do veículo guiado se revelou estável e não desviou da linha demarcatória no chão, com algumas melhorias podemos aplicar o conceito em ambiente fabril.

5 REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINOECIA, **Sensores de temperatura e Umidade** DHT22. Disponível em <http://www.arduinoecia.com.br/2015/02/sensor-de-temperatura-e-umidade-dht22.html>. Acesso em: 02 set. 2017.

ARDUINO. Disponível em:< https://www.arduino.cc/>. Acesso em: 02 Set. 2017.

BRANCO, R. **Sensor ótico**: como funciona**.** Disponível em: <http://www.manutencaoesuprimentos.com.br/conteudo/3611-sensor-otico-como-funciona/>. Acesso em: 15 set. 2017.

ELECFREAKS. **Ultrasonic Ranging Module HC-SR04**. Datasheet. Disponível em: < http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf >. Acessado em 16 set. 2017.

FERRAZ, Mariano. **Servo Acionamento**. São Paulo – SP, SENAI, 2008.

FONSECA, U. **Ponte H**. Disponível em: <Http://www.sistemaembutido.com.br/article.php?id=124>. Acesso em: 02 set. 2017.

MCROBERTS, Michael. Motores de passo e robôs. In: ____. **Arduino básico**. São Paulo: Novatec, 2011. Cap. 10.p. 232 – 256, Cap. 14.p. 321 – 343.

SANTINI, Arthur Gambin. RFID: conceitos, aplicabilidades e impactos. **Rio de Janeiro: Ciência**, 2008.

THOMAZINI, Daniel; Pedro Urbano Braga de Albuquerque; **Sensores Industriais – Fundamentos e Aplicações.** 1ª Ed. São Paulo: Editora Érica Ltda., 2005.

THOMSEN, A. **Arduino**. Disponível em <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 02 set. 2017.

AGRADECIMENTOS

Os autores agradecem ao Centro Paula Souza e à FATEC Botucatu pelo apoio e incentivo ao projeto.