FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Centro de Ciências Tecnológicas – CCT
Engenharia de Computação

Relatório Controle Digital

Projeto Robô Seguidor de Parede

Equipe:

Edson Junior 1310780/7
Lucas Abrantes 1320653/8
Rodrigo Costa 1320661/9
Thiago Sales 1410702/9

Fortaleza/2017

1. Introdução

Este projeto tem como objetivo geral a montagem e a configuração do hardware, bem como a implementação de funções de um robô móvel que seja hábil a locomover-se de maneira autônoma com base na percepção de uma parede. Para a realização deste objetivo foram usados servos motores, sensores tipo ultrassónico, arduino como controlador e para ajustar a malha PID foi usada a técnica “tentativa e erro”.

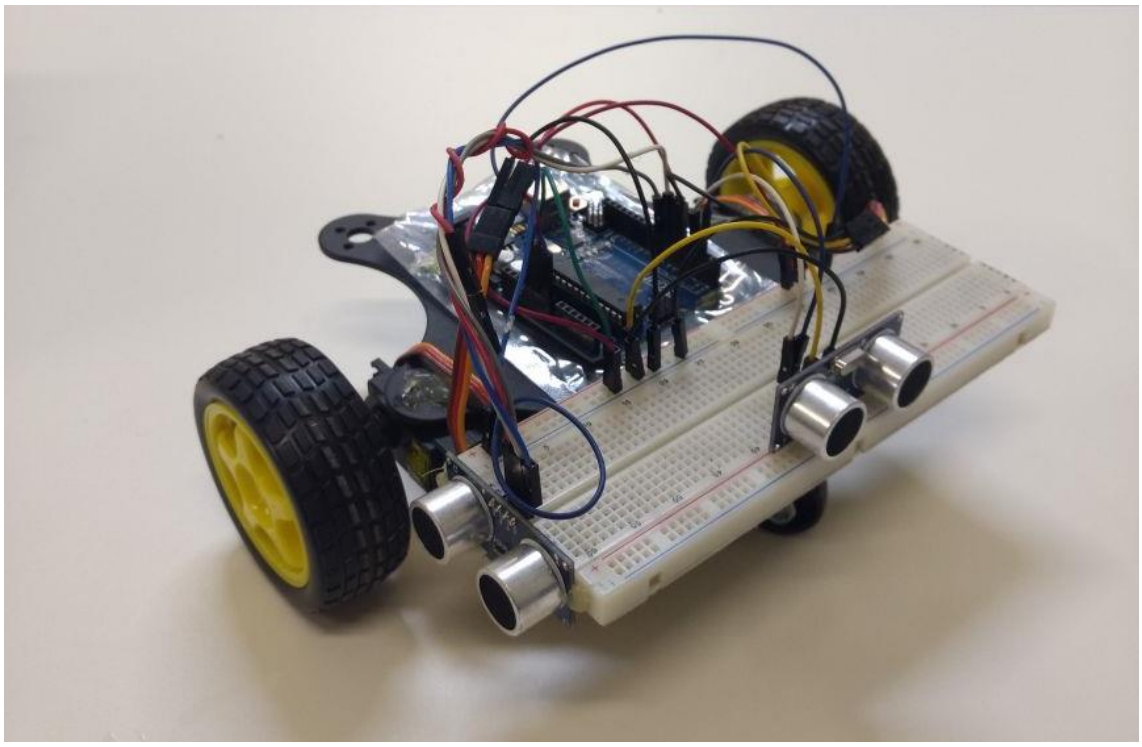

Figura 1 – Robô seguidor de parede.

1.1 Atuadores

- 2 servos motores.

Um servo motor é um atuador rotativo ou linear que garante o controle, velocidade e precisão em aplicações de controle de posição em malha fechada.

1.2 Sensores

- 2 sensores ultrassónicos

Os sensores ultrassónicos são amplamente utilizados para a percepção do ambiente e detecção de obstáculos em muitas aplicações robóticas. Estes sensores são muito bem sucedidos em termos de eficiência de custos, tempo de processamento e precisão.

1.3 Controlador

- Arduino UNO.

O arduino é um microcontrolador voltado para prototipação de projetos interativos, sendo o mesmo open (hardware e software) de fácil manuseio, baixo custo e flexível.

1.4 Lista de Materiais

- Corpo
    - Chapa metálica.
    - Rodas para servo motor.
- Protoboard e fiação.
- 2 Servos Motores(MG-995).
- 2 Sensores Ultrassónicos(HC-SR04).
- Arduino Uno.

2. Embasamento teórico/pratico.

2.1 Fluxograma

O fluxograma mostrado na figura 2 ilustra a cadeia de eventos que ocorre no programa desenvolvido. O cálculo do tempo de execução do código é utilizado para encontrar "dt" para que possamos encontrar "kd" e "ki", posteriormente é disparado um delay de 5 milissegundos para iniciar a leitura do sensor ultrassônico (foi necessário esse delay, pois foi observado que inicialmente estavam sendo feitas várias leituras de valores zeros). Depois da leitura o programa define uma rota com base na distância calculada pelo sensor frontal. Nesse caso, se for detectado um obstáculo a 12 centímetros pelo sensor frontal então o carro vira pra esquerda e volta ao primeiro evento do programa, caso contrário ele continua o seu trajeto normal realizando as correções necessárias com base nos valores detectados pelo sensor lateral, para mantê-lo a 20 centímetros da parede. Mais detalhes do funcionamento do sistema são mostrados no tópico 3(Implementação).

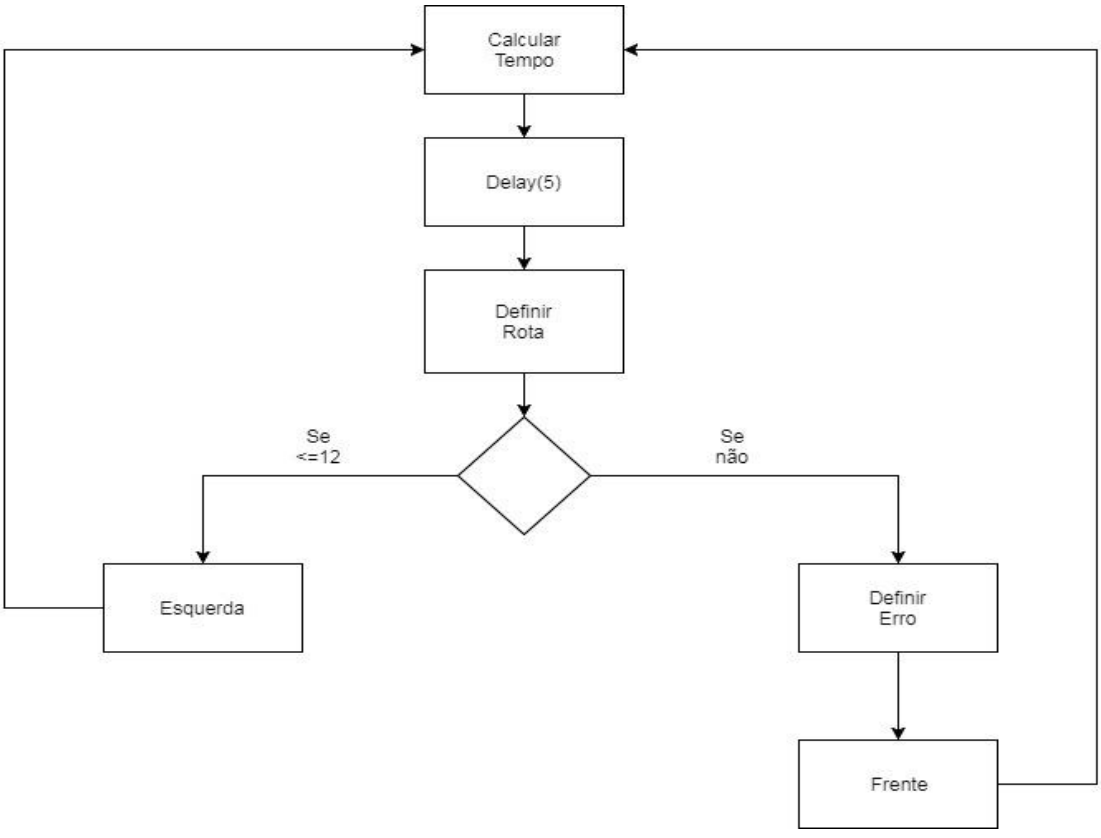

Figura 2.

2.2 Esquemático do projeto

Como é visto na figura 3, foram usadas somente as portas digitais do arduino para os atuadores e sensores, porem para os atuadores as portas estão em modo pwm. Na parte da alimentação foi usada a porta de 5v para os sensores e a porta Vin para os atuadores e um Gnd comum para todos.

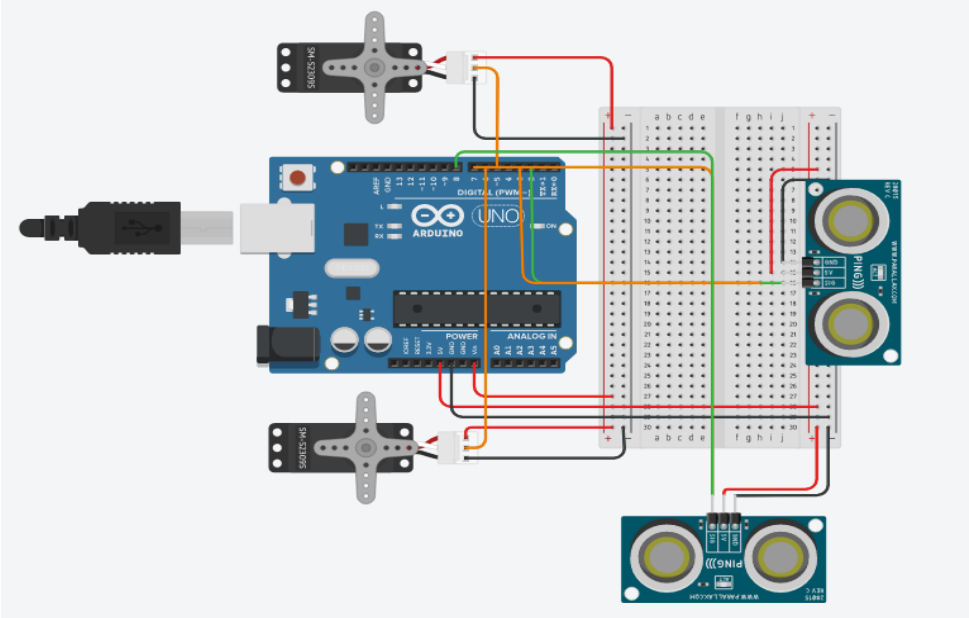

Figura 3.

2.3 Diagrama de Blocos

Na figura 4 o controlador, ou seja o arduino utilizado no projeto manda um sinal para o atuador (servos motores), para que esse componente possa atuar no sistema. De forma paralela o sensor retroalimenta o sistema de controle, mantendo o sistema sempre alimentado com novos dados(distâncias), que são por sua vez processados pelo programa contido no controlador que envia um sinal de resposta para o atuador, e assim por diante.

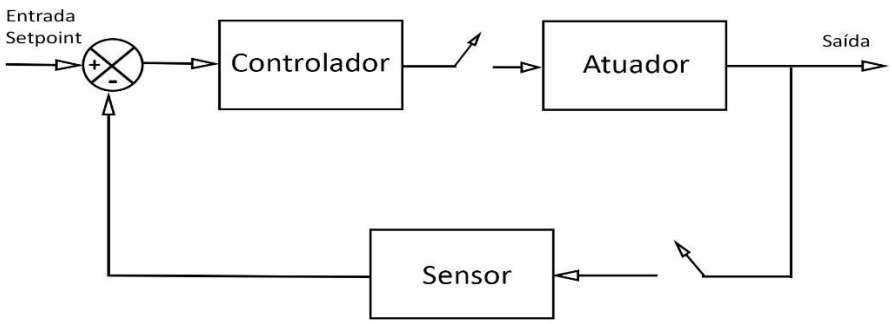

Figura 4.

3. Implementação

3.1 Funcionamento

```
#include <Servo.h>
```

Definindo as variáveis de controle do projeto, para os dois servos motores e os dois sensores ultrassônicos.

```
Servo rodaDireita;
Servo rodaEsquerda;

int echoFrontal = 2;
int trigFrontal = 3;
int echoLateral = 8;
int trigLateral = 7;
```

Na função *setup( )* são configurados os pinos utilizados na placa de prototipagem Arduino, essa configuração serve para determinar o funcionamento de cada um deles, como por exemplo os pinos de *trigFrontal* e *trigLateral,* estão configurados para a função saída de sinal, enquanto que, *echoFrontal e echoLateral* estão funcionando como entrada. Logo em seguida são determinados os pinos que irão controlar a velocidade e o sentido da rotação de cada um dos motores, através das funções *rodaEsquerda.attach(5) e rodaDireita.attach(6).*

```
void setup( ) {
  pinMode(trigFrontal,OUTPUT);
  pinMode(echoFrontal,INPUT);
  pinMode(trigLateral,OUTPUT);
  pinMode(echoLateral,INPUT);
  digitalWrite(trigFrontal,LOW);
  digitalWrite(trigLateral,LOW);
  rodaEsquerda.attach(5);
  delay(1);
  rodaDireita.attach(6);
  rodaEsquerda.write(90);
  rodaDireita.write(90);
}
```

Como o próprio nome já diz a função *dispararPulso (int pinEcho, int pinTrig)* é responsável por disparar um pulso de sinal digital por 10µs, através dos comandos *digitalWrite(pinTrig, HIGH), delayMicroseconds(10) e digitalWrite(pinTrig, LOW)* no pino de *trig,* essa sequência de comandos dá início a leitura do sensor ultrassônico, que funciona da seguinte forma, o emissor dispara uma onda sonora de alta frequência que esbarra em algum obstáculo e reporta para o receptor, enquanto isso é medido o tempo total desse percurso e esse intervalo é estimado em µs, esse valor é retornado através do pino de *echo* do sensor e é armazenado na variável tempo.

```
int dispararPulso (int pinEcho, int pinTrig){
  float tempo;
  digitalWrite(pinTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrig, LOW);
  tempo = pulseIn(pinEcho, HIGH,4500);
  return tempo;
}
```

Para calcular a distância entre o sensor e o objeto a sua frente precisamos calcular o tempo em µs que o som leva para percorrer um 1 cm sabendo que a velocidade do som é de 34000 cm/s:

1000000 µs ⟹ 34000 cm
X µs ⟹ 1 cm

Resolvendo essa regra de três temos que o som percorre 29,4 cm/µs, mas o valor retornado pela função *dispararPulso( )* é o tempo de ida e volta do som, então devemos dividir tudo isso ainda por dois para obtermos a distância correta.

```
int calcularDistancia (int pinEcho, int pinTrig){
  return dispararPulso(pinEcho,pinTrig)/29.4/2;
}
```

Essa função é utilizada para rotacionar o carro em seu próprio eixo para o lado esquerdo a uma velocidade predeterminada pelos parâmetros recebidos pela função.

```
void esquerda(int velocidadeD,int velocidadeE){
  rodaEsquerda.write((90-velocidadeD));
  rodaDireita.write((90-velocidadeE));
}
```

Enquanto essa outra função realiza o movimento do carro para o sentido frontal, para que isso seja possível de forma correta foram estipulados valores limites para a velocidade das rodas. Mas caso os motores apresentem o mesmo funcionamento alguns desses valores poderão ser descartados.

```
void frente(int velocidadeD, int velocidadeE){
  velocidadeE = (velocidadeE / 1.5);
  rodaEsquerda.write((90+velocidadeE));
  rodaDireita.write((90-velocidadeD));
}
```

Aqui estão declaradas as variáveis utilizadas no cálculo do controlador PID e as que armazenam os valores durante a execução das medições e as constantes iniciais.

```
int distancia;
int ultimaDistancia;
int erro,x,y,dt;
int kd=10,kp=3;
float i,ki = 0.001;
int cons = 15;
```

A função *definirErro(int setPoint)* calcula o erro através da diferença entre o *setPoint* e a distância do carro à parede, caso o sensor lateral encontre uma barreira, o controlador irá calcular esse erro e estabelecer um valor para *x* e *y* para que o carro permaneça sempre a uma distância da parede igual ao valor de *setPoint*, caso contrário será estabelecido valores fixos para *x* e *y* para que o carro mantenha um percurso retilíneo.

```
void definirErro(int setPoint){
  ultimaDistancia = distancia;
  distancia = calcularDistancia(echoLateral,trigLateral);
  if(distancia > 0){
    erro =  setPoint - distancia;
    i += (erro*dt*ki);
    x = cons + ((erro*kp) + (kd*(distancia - ultimaDistancia)/dt) + i);
    y = cons - ((erro*kp) + (kd*(distancia - ultimaDistancia)/dt) + i);
   }
  if(distancia == 0){
    x = 10;
    y = 15;
   }
}
```

Para determinar os valores do integrativo e do derivativo da função PID precisamos calcular o tempo em que o programa leva para executar um loop completo, esse valor será o *dt* da função.

```
int temp ;
int tempFinal = millis();
void calcularTempo(){
  temp = tempFinal;
  tempFinal = millis();
  dt = (tempFinal - temp);
}
```

Como atividade extra implementamos também a detecção de barreiras a frente, para isso utilizamos a função *frontal( )*, que muda o sentido do carro para o lado esquerdo caso o sensor frontal esteja a menos de 12 cm de uma barreira, caso contrário ela chama a função calcular erro para verificar se tem barreira ao seu lado, para que ele possa continuar seguindo-a.

```
int distanciaFrontal;
void frontal(){
  distanciaFrontal = calcularDistancia(echoFrontal,trigFrontal);
  if(distanciaFrontal > 0 && distanciaFrontal <= 12){
    esquerda(20,20);
    delay(1000);
    i=0;
    distancia = 0;
    distanciaFrontal = 0;
  }else{
    definirErro(20);
    frente(x,y);
  }
}
```

Esse é o laço principal do programa onde chamamos as funções responsáveis pelo funcionamento do carro.

```
void loop() {
  calcularTempo();
  delay(5);
 frontal();
}
```

4. Conclusão

Aplicações desse tipo de projeto são bem difíceis de serem realizadas, já que a funcionalidade do robô seguidor de parede é apenas manter uma distância fixa de uma parede enquanto se locomove para frente. Nas pesquisas realizadas não foram achadas aplicações desse projeto, porém é notável que esse tipo de aplicação tem um potencial bastante elevado, e portando pode ser evoluído para a construção de um produto. Tal produto poderia ser um robô capaz de mapear a planta interna de uma residência e calcular suas dimensões, afim de que não necessite mais de um trabalho humano braçal para realizar tal atividade. Outra aplicação possível seria a utilização desse sistema de controle em automóveis, para que os mesmos mantenham uma distância segura das barreiras de contenção em pontes, podendo evitar portando graves acidentes automobilísticos. Tal controle poderia ser utilizado também para manter uma distância segura do veículo que está na frente e do que está atrás, e diminuir bastante o número de acidentes ocasionados por freadas bruscas. Esse tipo de sistema já pode ser encontrado nos carros autônomos produzidos pela Tesla Motors, porém com um custo bastante elevado, já que os automóveis produzidos são quase ou completamente autônomos. Dessa maneira é possível notar uma grande quantidade de aplicações possíveis que podem ser desenvolvidas apenas com a princípio básico do controle de distância utilizando sensores ultrassônicos. Portanto criar um produto que seja comercializável e que alcance uma grande massa de consumidores, se torna a tarefa mais desafiadora.