FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Centro de Ciências Tecnológicas – CCT
Engenharia de Computação
Engenharia de Controle e Automação

Relatório Robótica

Projeto Robô Seguidor de Trilha

Equipe:
Thiago Sales 1410702/9

Rodrigo Paulo 1121227 / 1

Fortaleza/2017

1. Introdução

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

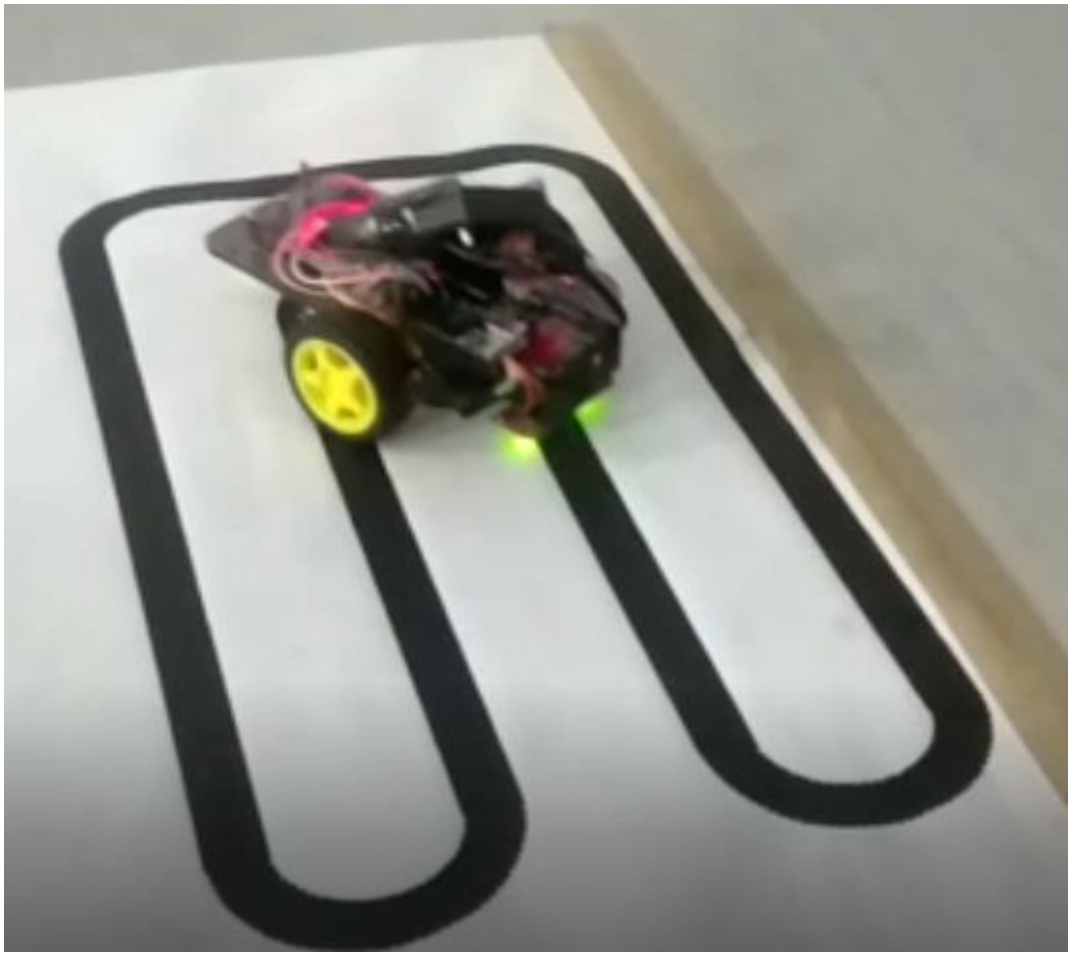

Figura 1 – Robô seguidor de trilha.

1.1 Atuadores

- Motores.

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

- Ponte h

Ponte H é um circuito electrónico que permite variar velocidades de um motor DC, assim como comutar o sentido de rotação dos motores, através de um sinal PWM. Estes circuitos são muito utilizados em robótica e estão disponíveis em circuitos prontos ou podem ser construídos por componentes.

1.2 Sensores

- Sensores infravermelho

Os sensores infravermelhos são amplamente utilizados para a percepção do ambiente e detecção de objetos ou pessoas em muitas aplicações robóticas e residêncial. Estes sensores são muito bem-sucedidos em termos de eficiência de custos, tempo de processamento e precisão.

1.3 Controlador

- Arduino duemilanove.

O arduino é nada mais que um microcontrolador com várias modificações 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 ou plástico.
- Rodas para o motor.
- Protoboard, fiação e bateria 7~9v.
- 2 Motores.
- 3 Sensores infravermelho.
- Ponte H.
- Arduino duemilanove.

2. Embasamento teórico/pratico.

2.1 Fluxograma

O fluxograma mostrado na figura 2 ilustra os eventos que ocorrem no programa desenvolvido. O cálculo do tempo de execução do código é utilizado para posteriormente definir os valores de “kd” e “ki”. Próximo passo é a leitura dos sensores infravermelhos para ser usado no definir erro. Depois da leitura o programa entra no definir erro que é onde calcula o valor do PID e aplica nas constantes resultando um valor para X e Y que são o valor da “velocidade”. Controle define uma rota com base no valor de X, Y, nesse caso se X e Y for maior ou igual que 0 ele vai continuar indo para frente, caso seja X maior que 0 e Y menor que 0, o carro irá com sentido para esquerda até que o trajeto seja normalizado, e o inverso quando o X for menor que 0 e o Y maior 0, o sentindo será o da direita. Lembrando que X e Y são o valor da rotação do motor (velocidade) e no comum esse valor é constante.

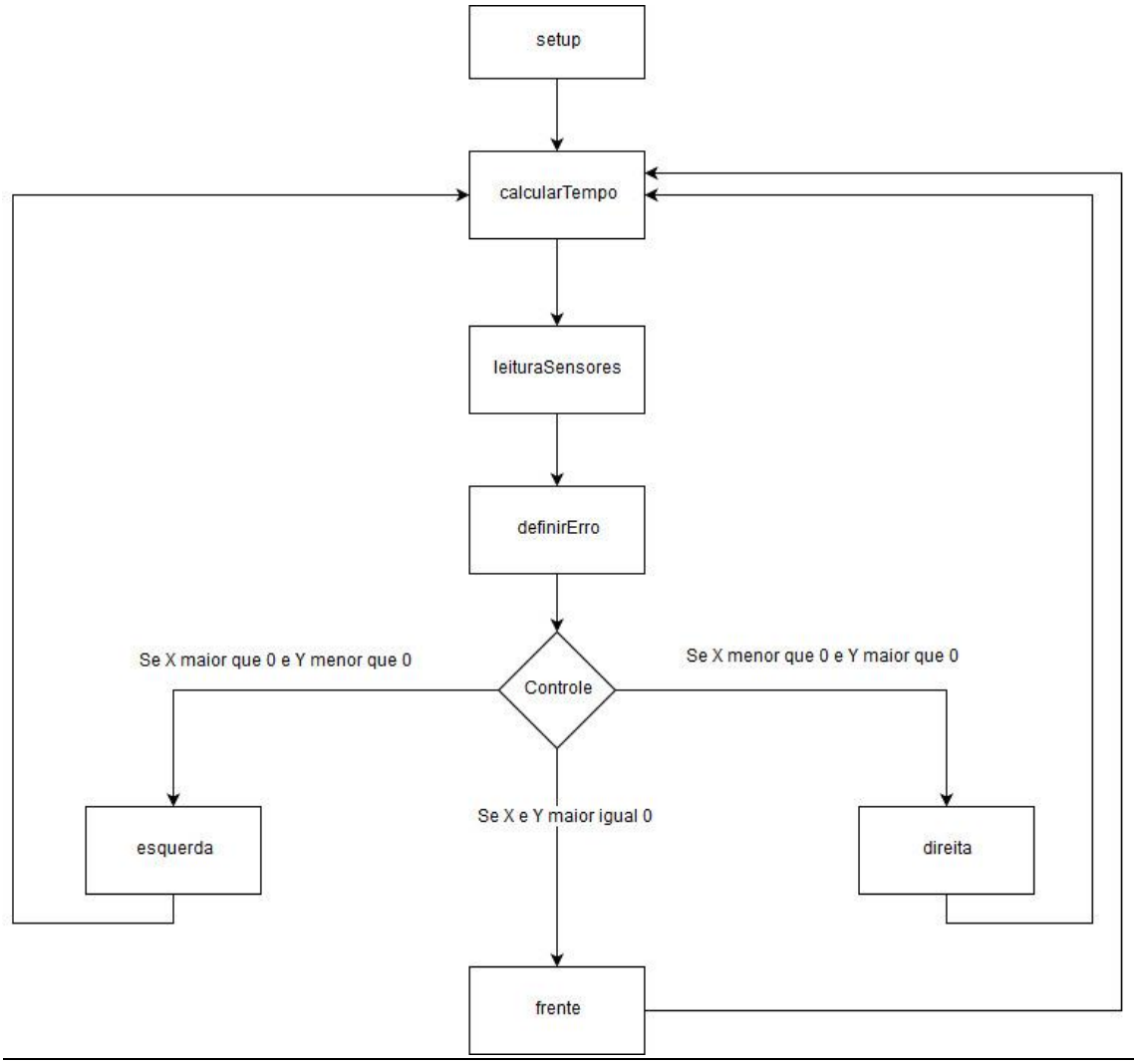

Figura 2.

2.2 Esquemático do projeto

Como é visto na figura 3, foram usadas somente as portas digitais do arduino para os sensores, porem para os atuadores tem portas em modo PWM e digital que define o eixo de rotação. 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.

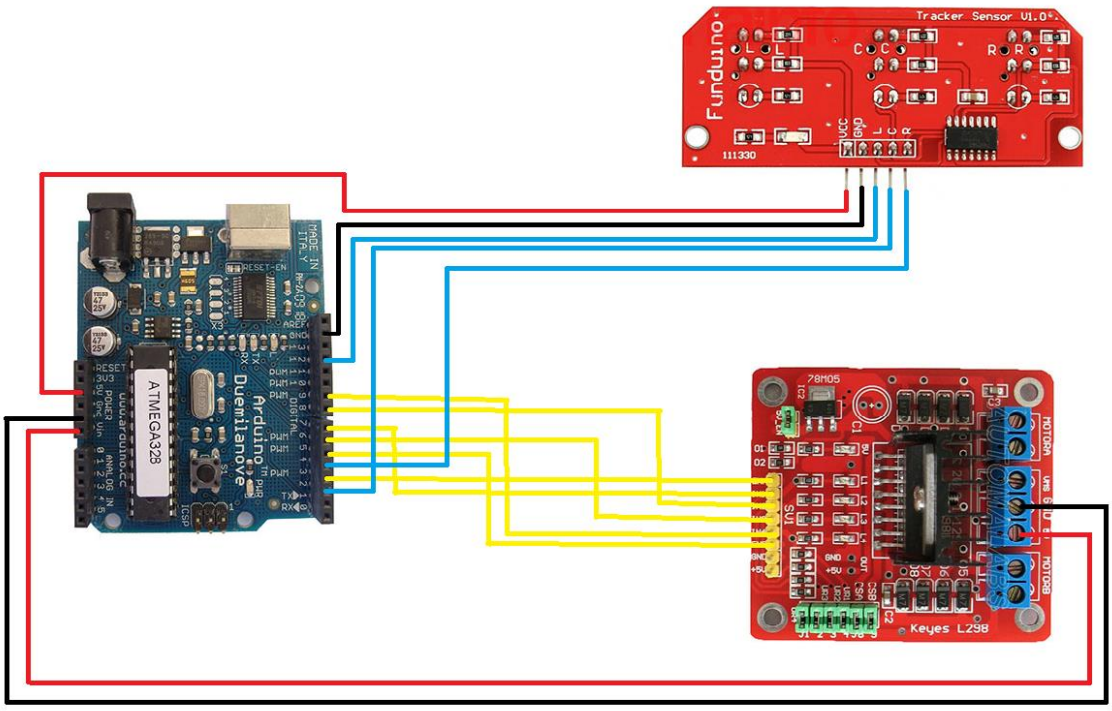

Figura 3.

2.3 Diagrama de Blocos

Na figura 4 o controlador, ou seja, o Arduino duemilanove utilizado no projeto manda um sinal para o atuador que é a ponte H e posteriormente os 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 (trajeto), que são por sua vez processados pelo programa contido no controlador que envia um sinal de resposta para o atuador, e assim por diante.

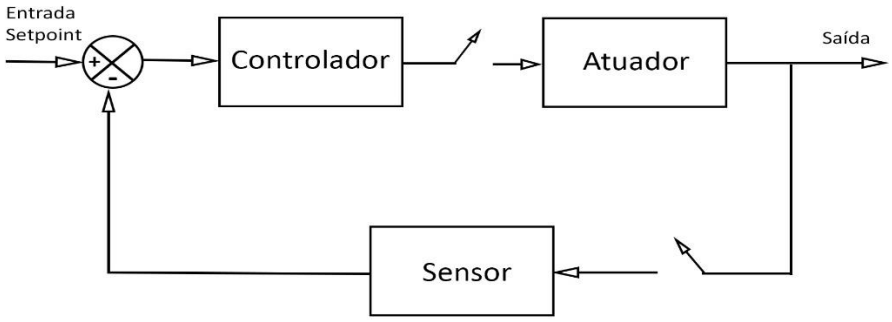

Figura 4.

3. Implementação

3.1 Funcionamento

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

Definindo as variáveis de controle do projeto, para ponte H e os três sensores infravermelho.

```
int ENA = 3;
int ENB = 5;
int IN1 = 7;
int IN2 = 8;
int IN3 = 6;
int IN4 = 9;
int sensorCentro = 2;
int sensorEsquerda = 12;
int sensorDireita = 4;
```

Na função *setup()* são configurados os pinos utilizados na placa Arduino, essa configuração serve para determinar o funcionamento de cada um deles, como por exemplo os pinos da ponte H que estão configurados para saída, enquanto que, *sensores* infravermelho estão funcionando como entrada. Logo em seguida são determinados os valores da saída dos pinos da ponte H que irão controlar a velocidade e o sentido da rotação de cada um dos motores, nesse momento estão tudo em low para evitar qualquer problema futuro.

```
void setup( ) {
        pinMode(ENA,OUTPUT);
        pinMode(ENB,OUTPUT);
        pinMode(IN1,OUTPUT);
        pinMode(IN2,OUTPUT);
        pinMode(IN3,OUTPUT);
        pinMode(IN4,OUTPUT);
        pinMode(sensorCentro,INPUT);
        pinMode(sensorEsquerda,INPUT);
        pinMode(sensorDireita,INPUT);
        digitalWrite(ENA,LOW);
        digitalWrite(ENB,LOW);
        digitalWrite(IN1,LOW);
        digitalWrite(IN2,LOW);
        digitalWrite(IN3,LOW);
        digitalWrite(IN4,LOW);
        tempoFinal = millis();
}
```

Essa função é utilizada para girar o carro em seu próprio eixo para o lado esquerdo ou direita a uma velocidade predeterminada pelos parâmetros recebidos pela função, nesse caso como as curvas das pista eram fechadas, um *offset* era necessário para rodar o

motor no sentido oposto com uma velocidade maior, fazendo que o carro reaja melhor nessas curvas, pois sem esse *offset* o carro na maioria das vezes passava da curva, como cada pista tinha uma curva diferente foi necessário encontrar o *offset* ideal para cada uma, esse caso é o da pista U.

```
void esquerda(int velocidadeMotor1, int velocidadeMotor2){
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENA,velocidadeMotor1);
  analogWrite(ENB,velocidadeMotor2+65);
}
void direita(int velocidadeMotor1, int velocidadeMotor2){
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  analogWrite(ENA,velocidadeMotor1 + 65);
  analogWrite(ENB,velocidadeMotor2);
}
```

Enquanto essa outra função realiza o movimento do carro para o sentido frontal, nesse caso as velocidades são iguais mas caso os motores funcionem com uma rotação distinta apesar de terem os mesmos valores, será necessário um *offset* para que ambos fiquem na mesma rotação. Mas caso os motores apresentem o mesmo funcionamento alguns desses valores poderão ser descartados.

```
void frente(int velocidadeMotor1, int velocidadeMotor2){
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  //velocidadeMotor1 = velocidadeMotor1 * 0.825;
  analogWrite(ENA,velocidadeMotor1);
  analogWrite(ENB,velocidadeMotor2);
}
```

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. Lembrando que para cada pista foi usado um kp, ki e kd diferente, caso precisasse mudar de fonte a constante deveria ser modificado levando em conta a potência da mesma. Esse caso é o da pista U.

```
int v = 0;
int erro = 0;
int x,y;
int constante = 120; //120constante
float kp = 1.80;//1.75
float ki = 0.0007;//0.0006
int kd = 0;//2
int dT = 0;
long tempoFinal=0;
long tempo = 0 ;
int ultimaLeitura;
float integral ; // armazena o valor do integrativo
```

A função leitura apenas pega os valores dos sensores usando as portas selecionadas anteriormente e armazena em três variáveis para posteriormente ser usado no definirErro.

```
void leitura(){
  leituraEsquerda = digitalRead(sensorEsquerda);
  leituraDireita = digitalRead(sensorDireita);
  leituraCentro = digitalRead(sensorCentro);
}
```

A função *definirErro(int setPoint)* calcula o erro através da diferença entre o *setPoint* e o cálculo da função V que utiliza os dados do sensores infravermelho do carro, caso o sensor da esquerda entre na rota, o controlador irá calcular esse erro e estabelecer um valor para *x* e *y* para que o carro permaneça sempre na trilha. Lembrando que o i é acumulativo logo se ficar muito tempo fora da rota esse valor vai ser grande suficiente para causar imprecisão por isso o carro deve ser ligado próximo a trilha.

```
void definirErro(int setPoint){
  ultimaLeitura = v;
  if((leituraEsquerda+leituraCentro+leituraDireita ) !=0 &&
(leituraEsquerda+leituraCentro+leituraDireita ) !=3){
    v = (300*leituraEsquerda + 200*leituraCentro + 100*leituraDireita
)/(leituraEsquerda+leituraCentro+leituraDireita );

  erro =  setPoint - v;
  integral += (erro*dT*ki);
```

```
  x = constante + ((erro*kp) + (kd*(v - ultimaLeitura)/dT) + integral);
  y = constante - ((erro*kp) + (kd*(v - ultimaLeitura)/dT) + integral);
  }
}
```

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.

```
void definirTempo(){
  tempo = tempoFinal;
  tempoFinal = millis();
  dT = (tempoFinal - tempo);
}
```

Como precisamos definir qual função deve ser chamada para continuar na rota a partir do valor de X e Y foi criado a função *controle()* , nela é chamado uma função para definir uma rotação máxima dos motores(velocidade), após isso é criado três condições que vai definir a função de direção , a primeira é caso X e Y sejam iguais ou maiores que zero, será chamado a função de frente , caso o X seja maior que zero e Y menor que zero será chamado a função que movimenta o carro pra esquerda, a última e quando X é menor que zero e Y maior que zero , o carro irá na direção da direita.

```
 void limiteXY(){
  if(x >= 200){
    x = 180;
  }
  if(y >= 200){
    y = 180;
  }

  if(x < -180){
    x = -180;
  }
  if(y < -180){
   y = -180;
  }
}
void control(){
  limiteXY();
  if(x >= 0 && y>= 0){
    frente(x,y);
  }else if (x < 0 && y> 0){
    direita(-x,y);
```

```
  }else if(x > 0 && y< 0){
    esquerda(x,-y);
  }
}
```

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

```
void loop() {
  definirTempo();
  leitura();
  definirErro(200);
  control();


}
```

3.2 Fase de testes.

As dificuldade encontradas no projeto foram diversas, a primeira foi a rotação dos motores que não giravam na mesma rotação fazendo que não fossem em uma reta com a mesma constante , após a correção outro problema apareceu , as baterias quando começavam a descarregar, consequentemente a velocidade e as possíveis correções elaboradas anteriormente não funcionavam corretamente , precisando de novos ajustes, com isso mais tempo era gasto , recomendável é usar uma fonte que possa fornecer uma potência constante com isso os ajustes se manteriam , ondulações e a iluminação (fluorescente) é um problema para o carro continuar seu percurso, um exemplo disso foi quando utilizado uma lona que continha algumas ondulações o carro tinha que fazer correções no trajeto constantemente para se manter, por mais que a ondulação seja pequena. Em relação as pistas e suas calibrações, a pista do formato U e S não foi preciso usar a constante Kd (derivativo), sendo a correção do trajeto feito pelo ki e kp, em relação ao valor atribuído, a pista S teve valor um pouco menor em kp e no *offset* em comparação com U, pois as curvas da pista S era menos fechada, na pista H foi necessário adicionar o Kd, pois as curvas eram mais fechadas que as outras fazendo que o carro saísse da pista.

4. Conclusão

Aplicações desse tipo de projeto são bem difíceis de serem realizadas, já que a funcionalidade do robô seguidor de linha é seguir o trajeto demarcado enquanto se locomove para frente. Nas pesquisas encontradas poderá ser usado para rotinas repetitivas, como por exemplo, ir de um ponto a outro do ambiente transportando materiais, ou ainda, através do código implementado, realizar alguma tarefa durante o trajeto. Na atualidade existem algumas empresas que trabalham com robôs que fazem trajetórias pré-estabelecidas e autônomas. A Amazon usa os robôs para separar os produtos no estoque, a STO Express usa os robôs para separar as correspondências por localização. Dessa maneira é possível notar uma pequena quantidade de empresas que trabalham com essa tecnologia. Resumindo, foi um projeto muito desafiador, tivemos trabalho em achar os valores do controle PID no programa, cada pista teve seu valor escolhido. Erramos muitas vezes, mas conseguimos atingir o nosso objetivo. O nosso robô cumpriu com os objetivos estabelecidos.