Colégio Técnico de Campinas

Cotuca

Relatório de Trabalho de Conclusão de Curso

Robô Seguidor de Linha
Robô Sumo

Nome: Jean Carlos
Nome: Danilo Bonini

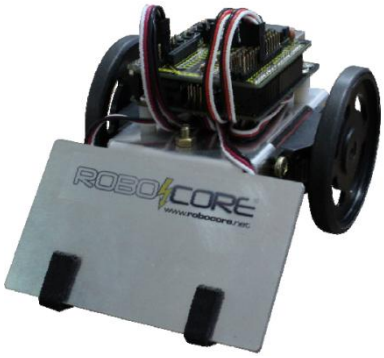
Semestre/Ano: 4ª ano Turma: 2014

Sumário

Introdução Teórica

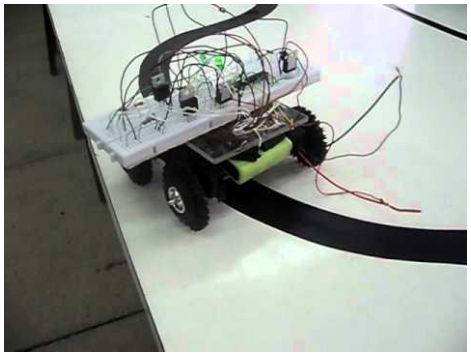
Robô Autônomo

Robôs autônomos são robôs que podem realizar os objetivos desejados em ambientes desestruturados sem a ajuda humana. Muitos tipos de robôs são autônomos em certos níveis. Diferentes tipos de robôs podem ser autônomos de diferentes formas. Um alto nível de autonomia é particularmente desejado em campos como a exploração espacial, onde a comunicação possui atrasos e as interrupções são inevitáveis.



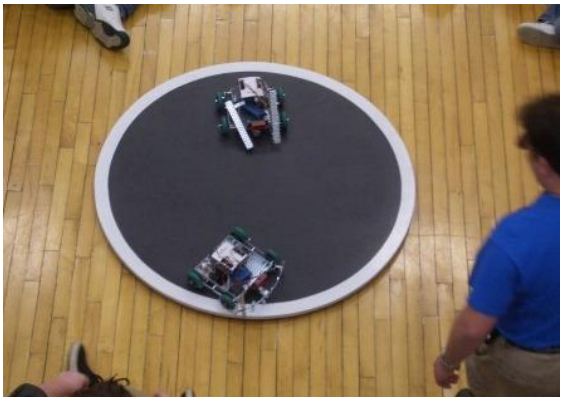
- **Robô Seguidor de linha**

Um robô seguidor de linha se trata de robô autônomo capaz de seguir um trajeto já determinado, para isso o robô deve fazer uso de sensores, servo motores, bateria, processador e uma boa linha código para fazer curvas ou seguir seu trajeto.



- **Robô Sumo**

Um robô sumo trata-se de uma máquina autônoma cujo objetivo se iguala ao do lutador original que é derrubar o inimigo do dojô, para isso o robô deve fazer uso de sensores, servo motores, bateria, processador e uma boa linha código para realizar seu objetivo.



Objetivo

O objetivo deste trabalho foi o desenvolver um projeto no qual reuni-se inúmeros conhecimentos adquiridos ao longo do curso em um trabalho final para reforçar o conhecimento adquirido nas disciplinas através dessa atividade prática.

Materiais utilizados

I. Arduino UNO R3

Arduino é uma plataforma de computação *open-source* baseada em uma simples placa com entradas e saídas tanto digitais como analógicas. Possui um próprio ambiente de desenvolvimento que implementa a Linguagem C.

II. Servo Motor

Este é um servo motor de rotação contínua (ele não é de posição angular, ou seja, quando você jogar um comando nele para ir para ângulos maiores que 90º ele irá girar para um lado sem parar, colocando valores menores que 90º ele irá girar para o outro lado sem parar - teoricamente, colocando em 90º ele deve ficar parado).

III. Sensor de Refletância QRE - Analógico

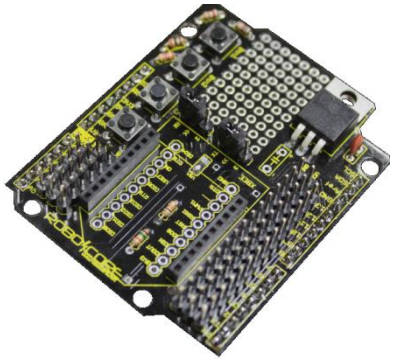
Esta versão da QRE1113 permite de uma maneira fácil com uma saída analógica, fazer a detecção de uma linha, por meio da variação de luz infravermelha refletida pela superfície onde está a linha.

IV. Chassi

E uma estrutura de suporte para outros componentes que pode ser feita de aço, alumínio, ou qualquer outro material rígido.

V. Arduino Shield ProtoFull

Este *shield* é altamente recomendado para o acionamento dos módulos relés, acionamento de servo motores, além da facilidade de conexão com a grande maioria dos sensores existentes e fazer comunicação sem fio utilizando módulos *Xbees* de quaisquer séries.



VI. Bateria (pilha)

Pilha voltaica é um dispositivo que utiliza reações de óxido-redução para converter energia química em energia elétrica. A reação química utilizada será sempre espontânea.

VII. Rodas

Esta roda possui o furo para eixo padrão de servo motores de rotação contínua *Futaba Parallax*. Perfeito para colocar em projetos que precisam de movimento, juntamente a servo motores de rotação contínua.

Funcionamento

Montagem

Primeiro confira as peças:

Chassi (1x)

Arduino Shield – ProtoFull (1x)

Sensor de Linha Analógico (2x)

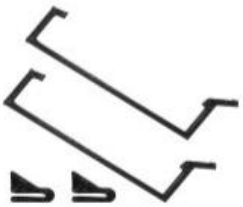
Roda para Servo (2x)

Servo Motor de Rotação Contínua (2x)

Cabo Extensor (2x)

Suporte para 6 Pilhas AA (1x)
(pilhas não inclusas)

Conector DC P4 (1x)



Segura-pilhas (2x)
Segura-sensores (2x)

Parafusos M4x30 (2x)

Parafusos M3x6 (2x)
Porcas M3 (2x)
Espaçador de Nylon M3 (2x)

Parafusos M3x8 (2x)

Parafusos M4x8 (6x)

Porcas M4 (8x)

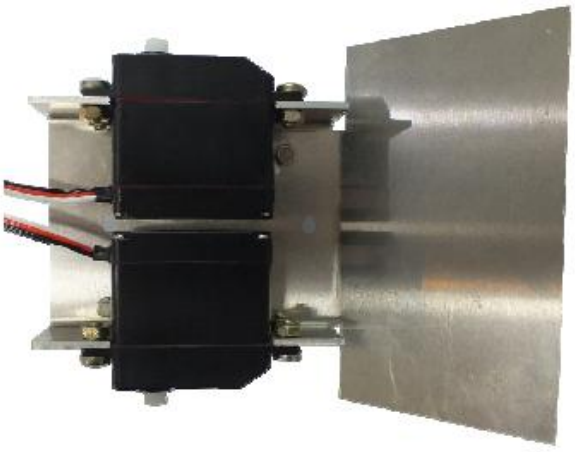
Rolo de Fita Isolante (1x)

Obs.; a placa Arduino UNO R3 também acompanha o conjunto de peças.
O passo a passo para a montagem sugerido é:

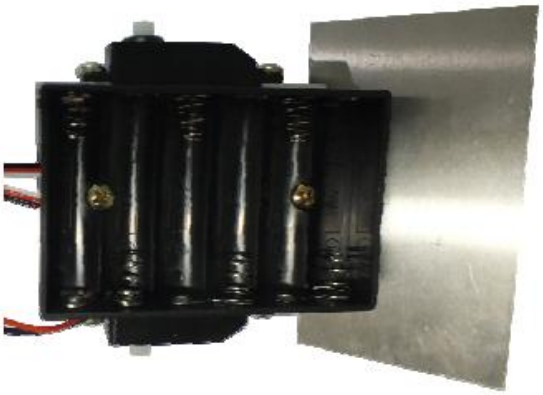
1 – Coloque o Espaçador de Nylon M3 no furo da extremidade inferior esquerda conforme a figura

2 – Prenda o espaçador com a sua devida porca (M3)

3 – Repita o procedimento e prenda o outro Espaçador de Nylon na extremidade superior direita conforme a figura.



4 – Prenda os servos utilizando os parafusos M4x8, conforme a imagem acima.



5 – Prenda o suporte de pilhas utilizando os parafusos M4x30 conforme a imagem acima.

6 – Coloque as tiras de borracha nas rodas conforme a figura acima

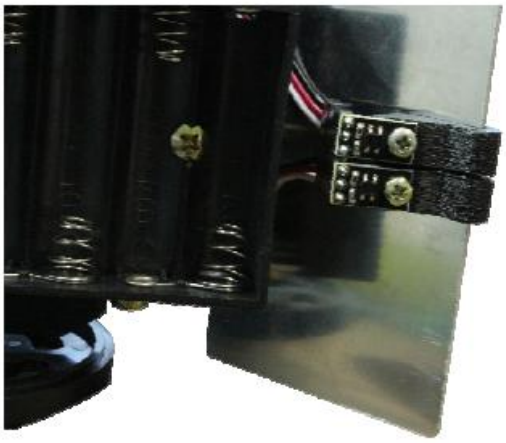
7 – Coloque as rodas nos servos motores e fixe-as com o parafuso soberbo que vem junto com as peças do servo. OBS: As rodas entram muito justas nos servos, é necessário forçar um pouco o encaixe

8 – Coloque o Arduino sobre os Espaçadores de Nylon e prenda com os parafusos M3x6

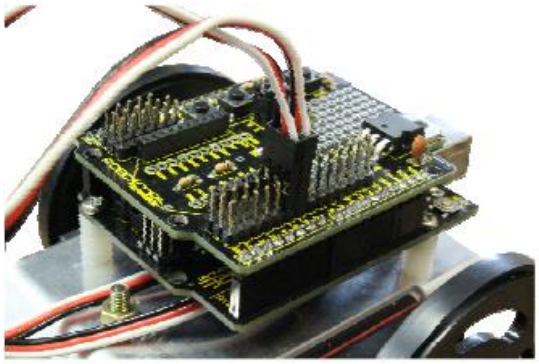
9 – Coloque o Arduino Shield - ProtoFull sobre o Arduino.

10 – Parafuse o sensor de linha da seguinte forma em seu suporte

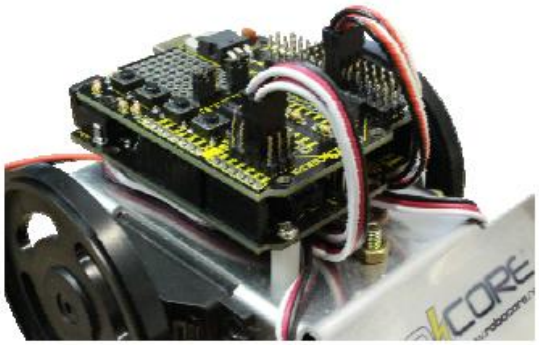
11 – Conecte o cabo extensor no sensor de linha.
ATENÇÃO: certifique-se que o cabo vermelho esteja na posição VCC e o cabo preto na GND.



12 – Prenda os suportes de sensores na rampa frontal conforme a figura.

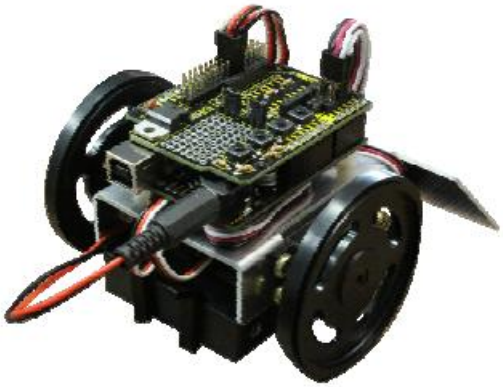


13 – Conecte os cabos provenientes do servo motor nos pinos que estão ligados nas saídas digitais (~5) e (~6) do Arduino.
ATENÇÃO: certifique-se que o cabo preto está conectado no GND e o vermelho no 5V



14 – Conecte os cabos provenientes dos sensores nos pinos que estão ligados nas entradas analógicas (3) e (4).
ATENÇÃO: certifique-se que o cabo preto está conectado no GND e o vermelho no 5V

15 – Coloque as pilhas no suporte e as trave utilizando os Segura-pilha conforme a imagem



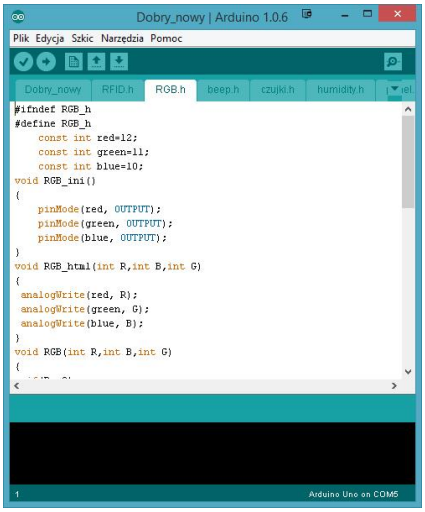
16 – Conecte o Conector DC P4 no Arduino.
ATENÇÃO: Caso haja alguma programação já existente no Arduino, ao energizar a placa o robô poderá se locomover.

ATENÇÃO: Tome MUITO cuidado ao fazer as ligações dos cabos. Caso você ligue qualquer cabo ao contrário poderá danificar permanentemente os sensores, os servos ou a placa Arduino. Tome muito cuidado ao fazer a montagem proposta.

Princípios de Funcionamento – Seguidor de Linha

Uma placa Arduino como Arduino UNO deverá ser utilizada em conjunto com esse kit e servirá como cérebro de nosso robô. Os sensores e os servo motores serão ligados a esta placa, que irá fazer a leitura dos sensores e, a partir desta leitura, tomará as devidas providências sobre o que fazer.

A criação do código para o seguidor de linha foi usado o programa **Arduino IDE – Arduino 1.0.6** como ponte entre o programador e a placa Arduino.



Na montagem deixou-se um espaçamento entre os sensores onde cabe exatamente a largura de uma fita isolante. A razão disto é obvia para seguir uma linha o mais indicado é que nossos dois sensores estejam dentro dela, como nas imagens abaixo:

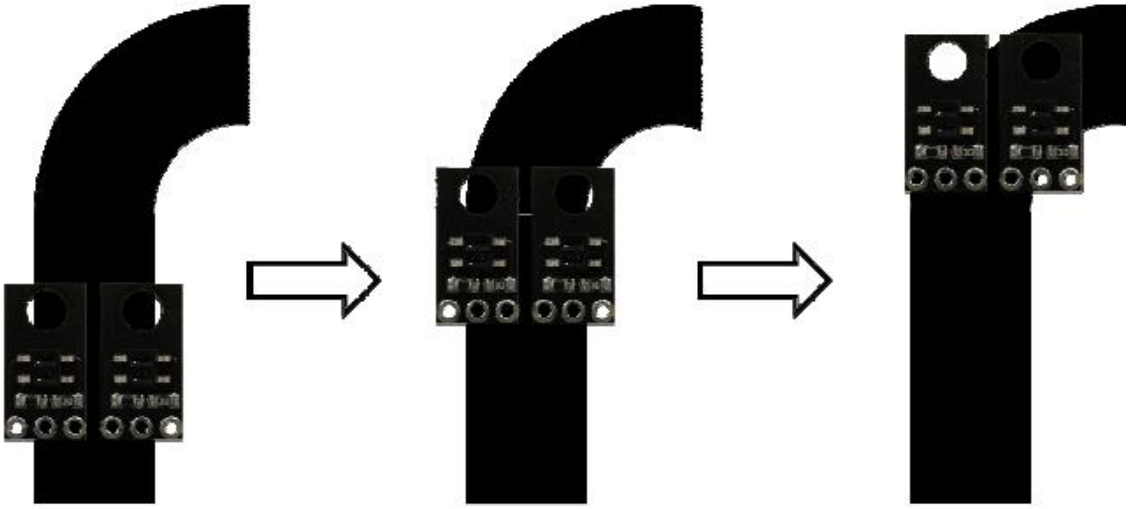
Sensores na linha vistos de cima

Sensores na linha vistos de baixo

Caso o robô esteja com os dois sensores dentro da linha preta, ou seja, os dois sensores dentro da fita isolante, quer dizer que nosso robô está na rota certa e ele deve aumentar sua velocidade. Nesta situação é bem provável que o robô esteja em uma reta. Em uma reta é interessante correr o mais rápido que puder.

Imagine agora que o robô esteja nesta linha reta e chega a uma curva para a direita. Como o robô deverá agir?

Veja abaixo os sensores chegando na curva:



Se nada for feito, o robô simplesmente sairá do percurso, pois está apenas programado para seguir uma linha reta. Portanto, ao chegar a uma curva, deveremos compensar a direção do robô alterando a velocidade de cada motor. Em o robô, possuí-se dois servo-motores. Desta forma, quando chegar a uma curva para a direita, a roda de dentro da curva deverá ir

mais devagar do que a roda de fora da curva, fazendo assim com que o robô consiga virar. Então faça o robô girar em torno do seu próprio eixo.

Para isso entenda o código:

```
#include  <Servo.h> //inclui a biblioteca Servo

Servo ServoD; //falo para o programa que usarei um Servo chamado ServoD
Servo ServoE; //falo para o programa que usarei um Servo chamado ServoE

void setup(){
   ServoD.attach(5); //o servo ServoD esta colocado no pino 5 digital (que tem PwM)
   ServoE.attach(6); //o servo ServoE esta colocado no pino 6 digital (que tem PwM)
}

void loop{
 ServoD.write(180) ; //velocidade máxima do ServoD para um sentido
 ServoE.write(180) ; //velocidade máxima do ServoE para o mesmo sentido
}
```

#include <Servo.h> //inclui a biblioteca Servo

Primeiramente incluí-se a biblioteca Servo.h ao programa. Esta biblioteca é nativa do Arduino, ou seja, ela já vai com a IDE do Arduino ao baixar a mesma da internet. Esta biblioteca serve para controlar servo motores. Ela converte os sinais PWM que podemos obter da placa Arduino para os sinais que os servo-motores entendem. Esta biblioteca possui algumas funções pré-definidas, como toda boa biblioteca, tornando fácil o uso de servos com a placa Arduino. Nesse código, utiliza-se duas destas funções, o SERVO.ATTACH e o SERVO.WRITE.

Veja o código a seguir:

```
Servo ServoD; //falo para o programa que usarei um Servo chamado ServoD
Servo ServoE; //falo para o programa que usarei um Servo chamado ServoE
```

Para esta biblioteca funcionar, é preciso atribuir a cada servo motor que irá se utilizar um nome. No caso, atribuiu se os nomes ServoD e ServoE. Nomes dados, próxima parte do código:

```
void setup(){
ServoD.attach(5); //o servo ServoD está colocado no pino 5 digital (que tem PWM)
ServoE.attach(6); //mesmo para o ServoE
}
```

O SETUP do programa, onde se configurou o que deve ser configurado previamente. Nesta parte se diz ao microcontrolador onde estão ligados fisicamente os servo-motores. No caso, o que chama de ServoD está ligado ao pino digital 5, e o que chama de ServoE está ligado ao pino digital 6 – conforme o hardware. Enfim, vá ao LOOP:

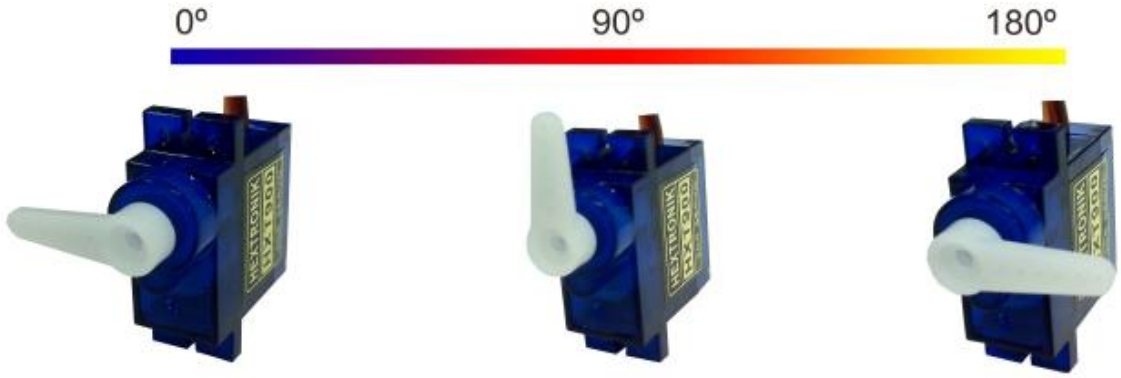
```
void loop(){
ServoD.write(180); //velocidade máxima do ServoD para um sentido
ServoE.write(180); //velocidade máxima do ServoE para o mesmo sentido
```

Aqui se informa a velocidade E o sentido dos motores. No caso, como se disse anteriormente, os dois motores estão com a velocidade máxima para o mesmo lado.

Não entendi velocidade e sentido com apenas uma linha de código?

Existem dois tipos de servo-motores: os de posição angular e os de rotação contínua. Nos servos de posição angular, o eixo está ligado diretamente a um potenciômetro, o qual

serve de guia para o servo. Este potenciômetro possui dois limites, um para um lado e outro limite para o outro lado de seu eixo. Desta forma, o eixo do servo motor de posição angular possui limites. Estes limites estão intimamente ligados aos limites do potenciômetro que existe dentro de sua carcaça. Basicamente, ao controlar um servo motor de posição angular, possuímos uma faixa de controle indo de 0º a 180º.



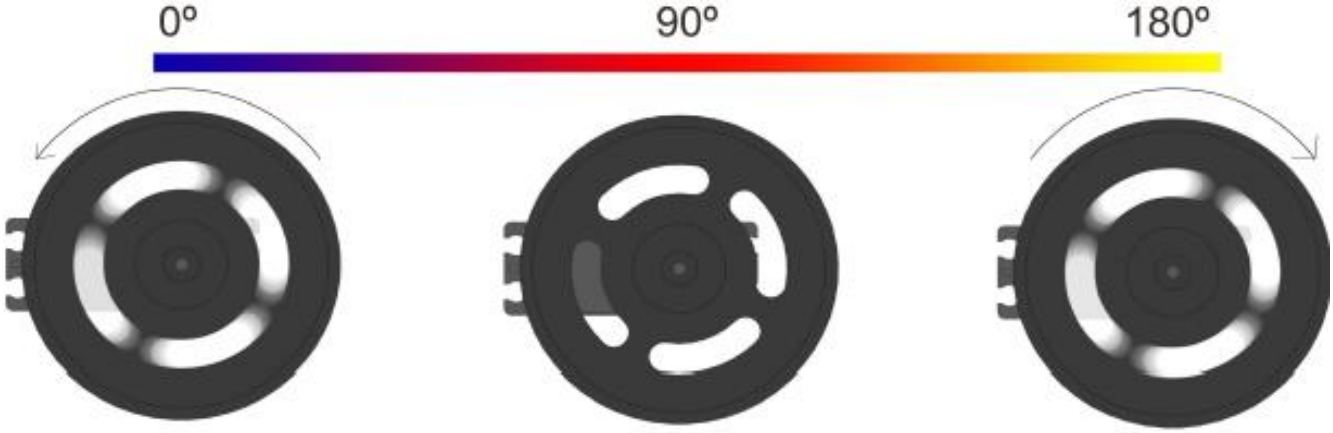
A biblioteca Servo.h também serve para controlar servo motores de posição angular. Desta forma, para colocar o servo motor na posição de 90º, deve-se colocar na programação do Arduino o seguinte:

Servo.write(90);

Sim, apenas isto já coloca o eixo do servo na posição de 90º. Desta forma, possuí-se uma resolução de 1º para esta faixa de valores, ou seja, pode-se mudar com precisão (dependendo da qualidade do servo motor), em incrementos de 1º em 1º, a posição do eixo.

GIRA SEM FIM E NÃO PÁRA NUNCA? COMO VAI DE UM LADO PARA O OUTRO?

Como o centro do potenciômetro agora é o centro de dois resistores, como em um divisor de tensão, quando se fala para o servo ir para a posição de 90º, o servo motor de rotação contínua irá ficar em seu regime parado, pois a resistência entre os dois lados do divisor de tensão é a mesma então não haverá movimento.



Na figura acima é mostrado que, se colocarmos aquele comando Servo.write(90); o servo motor de rotação contínua irá ficar parado. Se quisermos que ele gire sua roda, e haja como um motor DC comum, para um lado em sua velocidade máxima, basta enviarmos o comando Servo.write(0); e, caso queiramos que a roda gire para o outro sentido em sua velocidade máxima basta enviarmos o comando Servo.write(180);

Veja que agora aqueles incrementos de 1º em 1º servirão para controlar a velocidade que nossa roda gira. Se enviarmos o comando Servo.write(100); a roda irá girar para um lado

com uma velocidade reduzida, e se enviarmos o comando Servo.write(80); a roda irá girar para o outro lado, com a mesma velocidade do caso anterior. Então vamos agora fazer o robô se mover para frente, ainda sem seguir nenhum tipo de percurso, e vamos alterando sua velocidade com o passar do tempo.

Copie para a IDE do Arduino o seguinte código:

```
#include <Servo.h>

Servo ServoD;
Servo ServoE;

void setup(){
  ServoD.attach(5);
  ServoE.attach(6);
}

void loop(){
 ServoD.write(90);
 ServoE.write(90);
 delay(1000); //aguarda 1000 milisegundos
 ServoD.write(130);
 ServoE.write(50);
 delay(1000);
 ServoD.write(180);
 ServoE.write(0);
 delay(1000);
 ServoD.write(0);
 ServoE.write(180);
 delay(1000);
 ServoD.write(50);
 ServoE.write(130);
 delay(1000);
}
```

Este código testa se você realmente entendeu como controlar servos de rotação contínua com o Arduino. O começo do código é exatamente idêntico ao anterior que colocamos no Arduino. O que muda é o que existe dentro da rotina LOOP. Vamos dar uma olhada:

```
void loop(){
 ServoD.write(90);
 ServoE.write(90);
 delay(1000); //aguarda 1000 milisegundos
 ServoD.write(130);
 ServoE.write(50);
 delay(1000);
 ServoD.write(180);
 ServoE.write(0);
 delay(1000);
 ServoD.write(0);
 ServoE.write(180);
 delay(1000);
 ServoD.write(50);
 ServoE.write(130);
 delay(1000);
}
```

Bem, as primeiras duas linhas de código dentro do LOOP estão fáceis de entender, pois já falamos delas aqui. Conforme esta escrita na página anterior: “se colocarmos aquele

comando Servo.write(90); o servo motor de rotação contínua irá ficar parado.”. Portanto estas duas linhas dizem que nosso robô deve ficar parado. A próxima linha diz delay(1000); que nada mais é do que deixar o microcontrolador sem fazer nada durante 1000 milisegundos, ou seja, 1 segundo – entenda que este “sem fazer nada” faz com que os comandos enviados anteriormente continuem sendo feitos, ou seja, neste caso o robô irá ficar parado durante 1000 milisegundos.

Já vimos que as duas primeiras linhas fazem o robô ficar parado. A terceira linha do código faz o robô aguardar 1 segundo antes da próxima instrução. Então temos ServoD.write(130); que, como vimos anteriormente, faz com que o servo motor da direita gire para um lado com uma velocidade mediana, que não é a máxima mas também não é a mínima e, logo em seguida temos ServoE.write(50); que faz com que o outro servo, o servo da esquerda, gire para o outro lado, com velocidade proporcional a do servo da direita. Durante este segundo o robô irá se mover lentamente para frente. Então temos ServoD.write(180); e ServoE.write(0); que, como você já deve estar imaginando, move cada um dos servos em sua velocidade máxima, um para cada lado – fazendo com que o robô mova-se para frente em sua velocidade máxima.

Voltando ao código, o restante do mesmo é muito simples de se entender, agora que já vimos o principal. Após chegar a sua velocidade máxima e se manter nela por 1 segundo, o robô começa a andar para trás e então para.

Muito bem, dominamos a arte de controlar servo-motores de rotação contínua com Arduino. Agora vamos ler os sensores, juntar tudo e seguir a linha.

COMO FUNCIONA A COISA TODA?

Os sensores que acompanham este robô são os mesmos sensores infravermelhos analógicos que acompanha o Arduino. Eles possuem três pinos, um ligado à alimentação 5V, o outro ligado ao GND e o outro deve ser ligado a uma porta de entrada analógica no Arduino. Como já soldamos e ligamos todo nosso hardware, sabemos que existe um sensor deste tipo ligado à porta analógica 3 e outro ligado à porta analógica 4. Aqui vale lembrar que as portas analógicas do Arduino estão ligadas a um conversor analógico
digital. Ou seja, uma vez vindo valores analógicos no pino, este conversor faz com que o microcontrolador presente na placa Arduino entenda o que está recebendo e consiga lidar com isso.

A primeira coisa a se fazer é calibrar a leitura dos sensores. Para isto, coloque o seguinte código em sua placa Arduino:

```
int Sensor1 = 3;
int Valor_Sensor1 = 0;
int Sensor2 = 4;
int Valor_Sensor2 = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){

  Valor_Sensor1 = analogRead(Sensor1);
  Valor_Sensor2 = analogRead(Sensor2);
  Serial.print("Sensor 1 = ");
  Serial.print(Valor_Sensor1);
  Serial.print(" | ");
  Serial.print("Sensor 2 = ");
  Serial.println(Valor_Sensor2);
  delay(250);

}
```

Vamos entender este código. Primeiramente mostramos ao Arduino em que portas analógicas estão os sensores que iremos usar: int Sensor1 = 3;e int Sensor2 = 4;

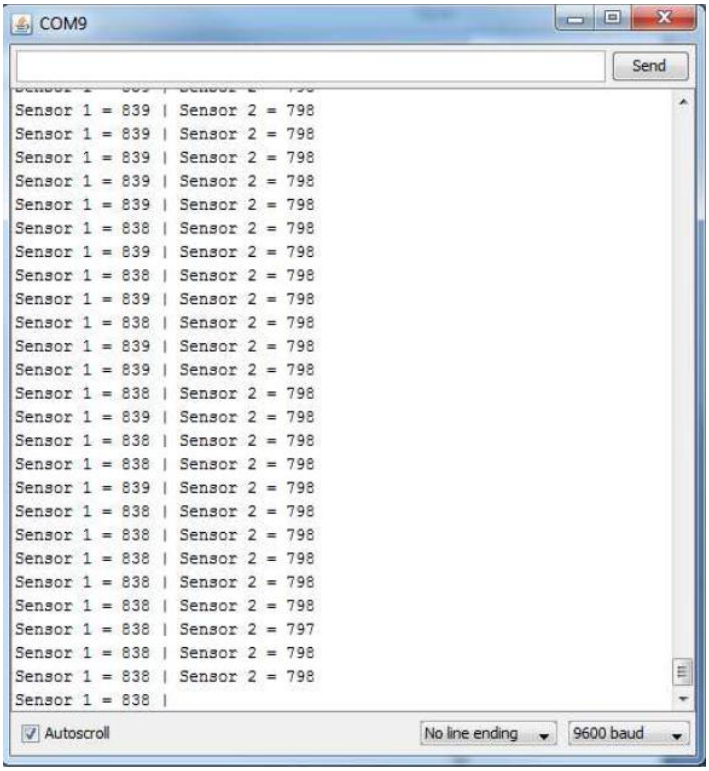
Existem diversos modos de dizer ao Arduino que você usará as portas analógicas. Este é apenas um dos métodos.

Ainda no começo de nosso código, criamos duas variáveis do tipo inteiro que servirão para armazenar os valores lidos pelos sensores. Nós trabalharemos apenas com estas variáveis no futuro. Os comandos para criar estas variáveis foram: int Valor_Sensor1 = 0;e int Valor_Sensor2 = 0;

Visto isto, entramos no SETUP de nosso programa. Neste bloco possuímos apenas um comando:
Serial.begin(9600); o qual simplesmente inicia a comunicação serial entre Arduino e computador, para que possamos ler no Monitor Serial os dados fornecidos pelos sensores.

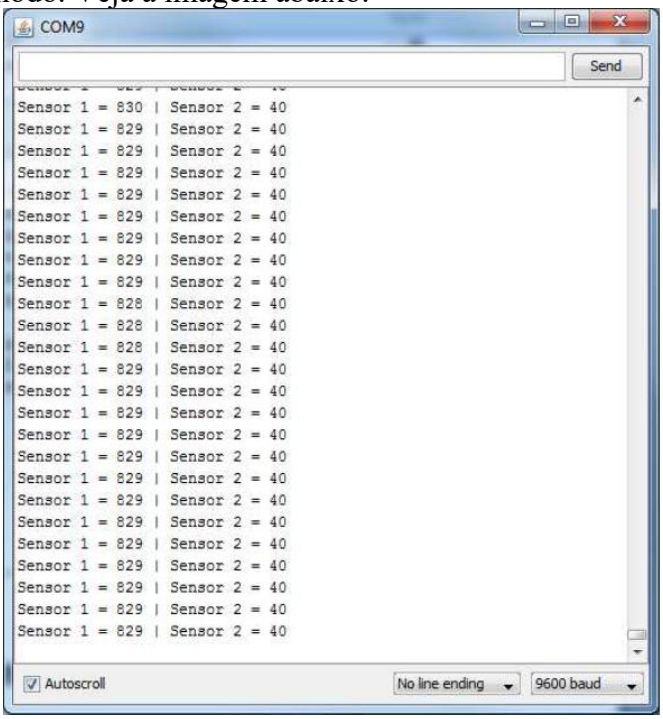
Na rotina dentro do LOOP, simplesmente lemos os sensores e mostramos no monitor serial os valores que eles apresentam. As primeiras duas linhas (Valor_Sensor1 = analogRead(Sensor1); e Valor_Sensor2 = analogRead(Sensor2);) fazem com que o conversor analógico digital seja usado e o valor lido seja salvo nas variáveis informadas. Com os valores lidos dentro da placa Arduino, vamos então mostrar eles no monitor serial para entender o que o Arduino irá receber quando nosso robô estiver com ambos os sensores sobre a linha, quando um sensor estiver sobre a linha e o outro não e quando os dois sensores estiverem fora da linha.

Após passar o código para o Arduino, coloque seu robô sobre a linha, certificando-se que os dois sensores estejam sobre a linha preta. Abra então o Monitor Serial e vamos observar o que temos:



Com os dois sensores sobre a linha, temos valores acima de 700 para ambos. Continue observando por alguns instantes os valores que aparecem e determine um valor que você possa colocar em sua futura programação do robô seguidor de linha que garanta que o robô está sobre a linha preta.

Muito bem, agora desloque seu robô um pouco para os dois lados e veja os valores no Monitor Serial. Se você mover só um pouco seu robô para o lado, deixando um sensor sobre a linha e outro fora dela, verá que os valores de um dos sensores se alterou, porém o outro continua do mesmo modo. Veja a imagem abaixo:



Veja que o Sensor 1 continua na linha, logo continua apresentando valores altos, maiores que 800. Já o Sensor 2, que saiu da linha, passou a apresentar valores menores que 50.

Agora vamos pegar um valor médio, algo em torno de 450. Então vamos elaborar uma rotina que diga: SE os dois valores lidos forem maiores que 450, iremos FAZER: andar o mais rápido possível. Para isto, iremos utilizar uma rotina condicional do tipo IF:

```
if((Valor_Sensor1 > 450) && (Valor_Sensor2 > 450)){
    ServoD.write(0);
    ServoE.write(180);
}
```

Então colocamos nossa rotina condicional IFEstamos verificando se ambos os sensores estão sobre a linha, logo, se ambos estão fornecendo valores maiores que 450. É usado um operador chamado AND, que é simbolizado em linguagem C como &&. Neste caso
ele está "juntando" as variáveis, e está dizendo que só acontecerá o que está dentro desta rotina IF caso os valores dos dois sensores sejam maiores que 450. Então, se os dois valores forem maiores que 450, nossos servos irão rodar em sua velocidade máxima.

Sabemos que, caso um sensor saia da linha, teremos valores menores que 100 em um sensor. Portanto, teremos que fazer outra rotina condicional IF para o caso de nosso robô sair da rota porque chegou a uma curva para a direita e teremos que fazer outra rotina IF caso nosso robô saia da linha por causa de uma curva para a esquerda. As duas rotinas condicionais ficam da seguinte maneira:

```
if((Valor_Sensor1 < 450) && (Valor_Sensor2 > 450)){
    ServoD.write(90);
    ServoE.write(180);
 }
 if((Valor_Sensor1 > 450) && (Valor_Sensor2 < 450)){
     ServoD.write(0);
     ServoE.write(90);
 }
```

Ok, por que mudamos as velocidades dos servos e porque paramos um deles sempre que chega a curva? Quando chegamos a curva iremos fazer com que o servo de dentro da curva pare momentaneamente e o de fora da curva ande em uma velocidade reduzida, afim de consertar a rota do robô.

Veja que nestas duas novas rotinas IF testamos as variáveis para saber se alguma delas é MENOR que 450. Fazemos isto para garantir que o sensor está fora da linha. Poderíamos muito bem testar se os valores das variáveis são menores que 100, porém colocando um valor alto garante que nosso robô comece a consertar sua posição no percurso assim que ele sair da linha por algum motivo.

Então nosso código completo de nosso primeiro seguidor de linha ficou assim:

```
#include <Servo.h>

Servo ServoD;
Servo ServoE;

int Sensor1 = 3;
int Valor_Sensor1 = 0;
int Sensor2 = 4;
int Valor_Sensor2 = 0;

void setup(){

  ServoD.attach(5);
  ServoE.attach(6);
}
```

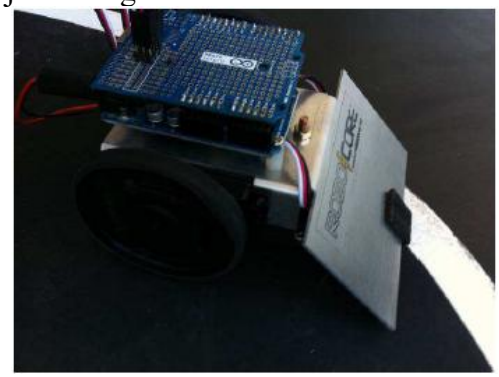
```
void loop(){
  Valor_Sensor1 = analogRead(Sensor1);
  Valor_Sensor2 = analogRead(Sensor2);
  if((Valor_Sensor1 > 450) && (Valor_Sensor2 > 450)){
  ServoD.write(0);
  ServoE.write(180);
  }
  if((Valor_Sensor1 < 450) && (Valor_Sensor2 > 450)){
  ServoD.write(90);
  ServoE.write(180);
  }
  if((Valor_Sensor1 > 450) && (Valor_Sensor2 < 450)){
  ServoD.write(0);
  ServoE.write(90);
  }
  if((Valor_Sensor1 < 450) && (Valor_Sensor2 < 450)){ //caso os dois sensores
  ServoD.write(0); //saiam da linha
  ServoE.write(90);
  }
}
```

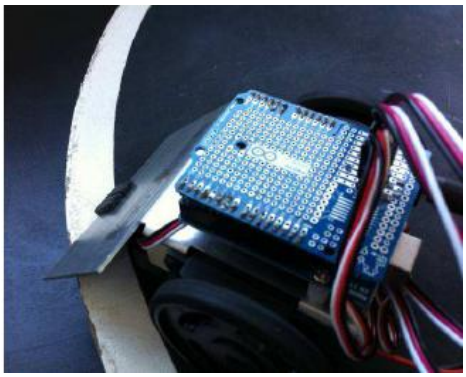
Princípios de Funcionamento – Robô Sumô

Tanto o percurso do seguidor de linha quanto a borda do dojô de sumo possuem uma cor que contrasta de outra. Ou seja, o percurso todo é feito com uma linha preta em uma superfície branca e o dojô é todo feito em uma cor e tem uma borda de outra cor! Logo, podemos usar o mesmo conceito do seguidor de linha para fazer nosso robô saber quando ele chegou à borda do dojô, e assim virar e voltar para não cometer suicídio.

Se apenas colocarmos nosso robô do jeito que está (com o código do seguidor de linha) em nosso dojô improvisado não irá acontecer muita coisa. Precisa-se, obviamente, fazer a programação adequada. Porém não é só isso.

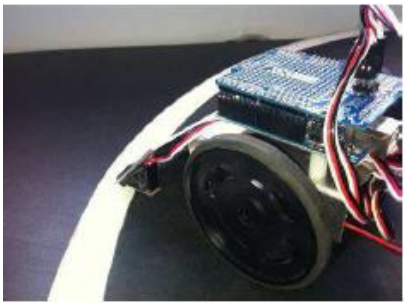
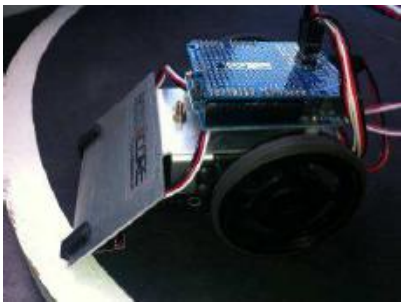
Atualmente os dois sensores de nosso robô estão no centro da rampa. Isto porque queremos sempre que o meio do robô fique alinhado com o percurso do seguidor de linha. Se deixarmos os sensores deste modo, nosso robô irá demorar muito para saber que chegou a borda. Veja as imagens abaixo:



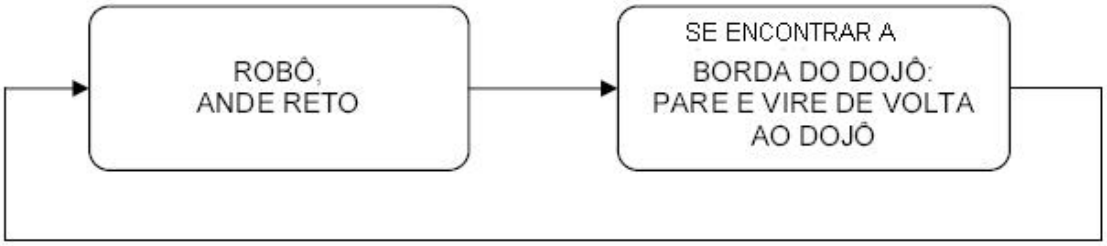


Ou seja, se deixarmos os sensores no centro da rampa como estava no seguidor de linha, as extremidades de nossa rampa chegarão a sair do dojô de sumo antes que nosso robô saiba que ele está saindo. Desta forma, nosso robô já estará quase com metade de seu chassi pra fora quando ele perceber que está saindo e então fazer algo para contornar esta situação.

Vamos contornar a situação trocando o lugar dos sensores. Então, coloque um sensor em uma das bordas da rampa e o outro sensor na outra borda. Desta forma, assim que o robô começar a sair do dojô nosso robô saberá e poderá agir de alguma maneira e sair daquele canto. Veja as imagens abaixo:


Como temos apenas dois sensores para nos basear, nosso fluxograma de programação não poderá ser muito além de:



O robô deverá ir sempre em linha reta. Quando o sensor encontrar a borda do dojô, o robô deverá parar completamente por algum tempo, virar para o dojô novamente e então continuar andando até que encontre a borda novamente, então deve parar, virar, voltar, encontrar a borda, parar, virar voltar, etc.

Certo, sabe-se como falar para nossos servo-motores como andar para frente, como parar e como girar em torno do seu próprio eixo, então a programação não será muito difícil nem muito diferente da do robô seguidor de linha. Em termos de programação, para o robô andar para frente teremos:

```
ServoD.write(0);
ServoE.write(180);
```

Para ele parar, teremos:

```
ServoD.write(90);
ServoE.write(90);
```

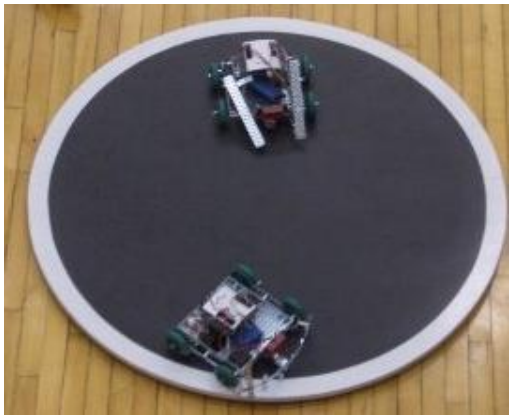
Para ele girar em torno do próprio eixo, teremos:

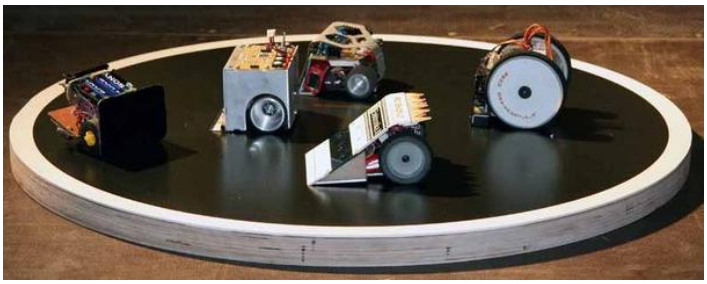
```
ServoD.write(180);
ServoE.write(180);
```

O ângulo que irá girar depende muito do tempo que irá fazer este giro em torno do próprio eixo. Como não usa-se neste robô nenhum tipo de bússola, não sabe-se com exatidão quantos graus ele irá girar quando fizer este giro em torno do próprio eixo.

Considerando que seu dojô será feito na mesma superfície que fez seu percurso para seguidor de linha, e que se use a mesma fita isolante para fazer a borda, já se sabe os valores que seus sensores leem quando chegam a borda do dojô, certo? No caso do robô que foi sendo feito durante a digitação deste relatório, sei que o valor que meus sensores leem quando estão sobre a linha preta é de algo em torno de 450. Então o que vai se fazer com este valor? Vai ir lendo ele e, enquanto nenhum sensor mostra este valor de 450, o robô vai andando para frente.

Outros exemplos de robô sumo:





Ok, então nossa programação fica assim:

```
#include <Servo.h>

Servo ServoD;
Servo ServoE;

int Sensor1 = 3;
int Valor_Sensor1 = 0;
int Sensor2 = 4;
int Valor_Sensor2 = 0;

void setup(){
  ServoD.attach(5);
  ServoE.attach(6);
}

void loop(){

  int Valor_Sensor1 = analogRead(Sensor1);
  int Valor_Sensor2 = analogRead(Sensor2);

  if ((Valor_Sensor1 < 450) && (Valor_Sensor2 < 450)){ //está fora da borda
    ServoD.write(0);
    ServoE.write(180);
   }

  if ((Valor_Sensor1 > 450) || (Valor_Sensor2 > 450)){ //um dos sensores está na borda
    ServoD.write(90); // ufa! paramos!
    ServoE.write(90);
    delay(150);
    ServoD.write(180); // meia volta, volver!
    ServoE.write(180);
```

```
      delay(200);
    }
}
```

No código acima apareceu um operador que ainda não havia sido visto: o operador OR, ou seja, o operador OU que é utilizado em linguagem C como duas barras paralelas: || (estas barras costumam estar sobre a tecla ENTER dos teclados, ou no canto inferior direito, perto da tecla SHIFT). Este operador é parecido com o &&, porém ao invés da condição ser "uma varíavel E outra varíavel" é: "uma variável OU outra variável é maior que 450".

Conclusão

Por meio desse experimento, foi possível exercitar parte dos conhecimentos obtidos em disciplinas anteriores como eletrônica, sendo possível ampliar o conhecimento relacionado à programação (linguagem C) e observar a importância do estudo de um projeto.

A experiência consistiu em fazer um robô seguir com certa perfeição e com velocidade o trajeto. Os problemas enfrentados pela equipe para a conclusão do trabalho serviu como ferramenta de revisão dos conceitos até então trabalhados, além do desenvolvimento de habilidades e competências primordiais para futuro profissional, tais como saber ser, saber aprender e saber discutir.

Assim, o experimento provou-se bastante enriquecedor no sentido de amadurecimento dos conhecimentos adquiridos relacionando a teoria à prática.

Bibliografia

Materiais Utilizados

- Arduino Uno R3
https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=120
- Servo Motor
https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=427
- Sensor de Refletância QRE – Analógico
https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=519
- Chassi
http://pt.wikipedia.org/wiki/Chassi
- Arduino Shield ProtoFull
https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=514
- Bateria
http://pt.wikipedia.org/wiki/Pilha
- Rodas
https://www.robocore.net/modules.php?name=GR_LojaVirtual&prod=364