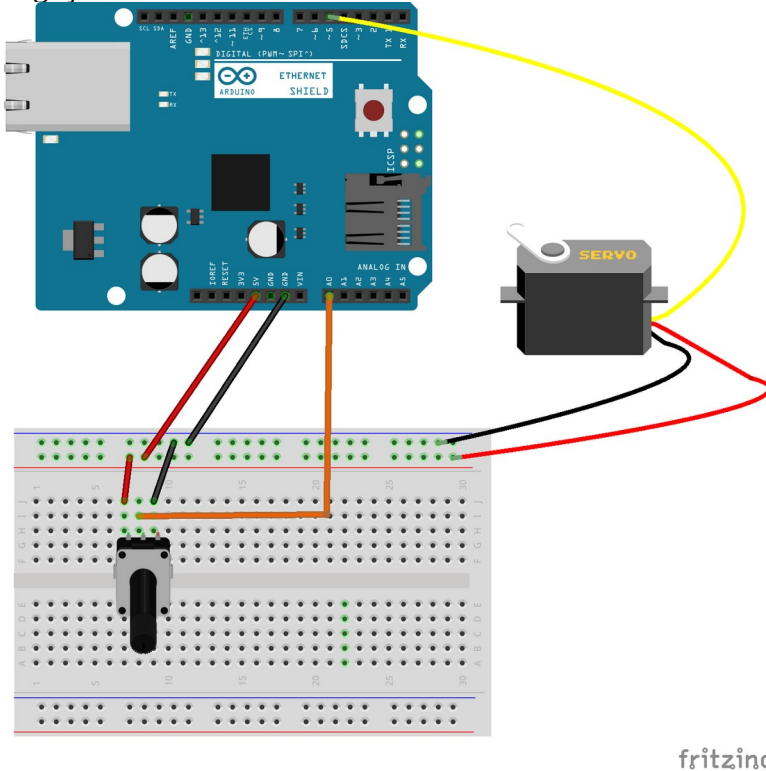


Servo motor

Aspecto - Fio laranja é o controle, preto é o GND e vermelho o +5V



Ligação do hardware:



Foram escolhidas as portas A0 para o potenciômetro e digital 5 para o servo.

Como funciona o servo?

Trata-se de um pequeno motor, o qual pode ou não completar um giro completo (360°). Os servos mais fornecidos em 'kits' de Arduino costumam ter um limite de giro de 180°.

O pino (em geral com fio laranja) por meio do qual são fornecidos os sinais de controle objetiva informar ao circuito de controle do servo motor a qual ângulo desejamos que ele se mova. Em muitos códigos de Arduino simplesmente utilizaremos uma biblioteca que nos permitirá dizer exatamente qual ângulo desejamos.

Em nosso experimento vamos usar o potenciômetro e a função 'map', já vistos, para gerar um número entre 0 e 180, o qual será enviado ao servo motor informando-o de que ele deverá, em correspondência, deslocar-se para uma posição entre 0 e 180°.

Código:

```
1 #include <Servo.h>
2 #define potenciometro A0
3
4 Servo meuServo;
5
6 int valor;
7
8 void setup() {
9     pinMode(potenciometro, INPUT);
10    meuServo.attach(5);
11 }
12
13 void loop() {
14     valor = analogRead(potenciometro);
15     valor = map(valor, 0, 1023, 0, 180);
16     meuServo.write(valor);
17     delay(20);
18 }
```

Na linha 1 foi incluída a biblioteca para trabalho com servo motores.

Na linha 2 foi definida uma variável ‘potenciometro’ e associada ao valor ‘A0’, que é uma porta de comunicação analógica do Arduino.

Na linha 4 foi definido um objeto do tipo ‘Servo’ com o nome de ‘meuServo’.

Na linha 6 foi definida uma variável do tipo int (para números inteiros), com o nome de ‘valor’.

Na linha 9 foi definido que a porta identificada como ‘potenciometro’ será do tipo ‘entrada’ (INPUT).

Na linha 10 o objeto ‘meuServo’ foi conectado à porta digital, do tipo PWM, de número 5 (o que é realizado por meio do método ‘attach’, que podemos traduzir como anexe, ou conecte).

Na linha 14 temos a leitura do valor do potenciômetro, o qual está eletricamente conectado entre as linhas de alimentação ‘GND’ e ‘+5V’; em função de seu giro, usando o princípio de divisão de tensão, teremos um valor de tensão disponível na porta analógica ‘A0’. Este valor será armazenado na variável ‘valor’.

Na linha 15, por meio da função ‘map’, transformaremos o conteúdo da variável ‘valor’, que tem neste momento um número entre 0 e 1023, para um número entre 0 e 180.

Na linha 16, o conteúdo da variável ‘valor’ será enviado ao servo, por meio do método ‘write’. Assim, por exemplo, se o número armazenado na variável ‘valor’ for ‘90’, o servo será instruído para deslocar-se à posição de 90°.

Na linha 17 foi inserida uma pequena espera, de 20 milissegundos, para evitar instabilidade no servo motor, pois se muitos valores forem enviados na sequência ele poderia receber um novo valor de posição enquanto ainda estive se deslocando para uma posição anterior.

Podemos fazer o servo movimentar-se sem o uso de um potenciômetro:

```

1 #include <Servo.h>
2 // #define potenciometro A0
3
4 Servo meuServo;
5
6 int valor;
7
8 void setup() {
9 // pinMode(potenciometro, INPUT);
10 meuServo.attach(5);
11 }
12
13 void loop() {
14 // valor = analogRead(potenciometro);
15 // valor = map(valor, 0, 1023, 0, 180);
16 // meuServo.write(valor);
17 // delay(20);
18 for (int posicao = 0; posicao < 181; posicao++){
19 meuServo.write(posicao);
20 delay(20);
21 }
22 }

```

As linhas 2, 9, 14, 15, 16 e 17 foram comentadas. Portanto, não serão consideradas no momento da compilação (que é o ato de transformar nosso código, chamado de ‘fonte’, em um código compreendido pelo Arduino).

Na linha 18 usamos a função ‘for’ para criar um ‘loop’, um bloco de comandos: foi definida uma variável do tipo inteira (‘int’) que recebeu inicialmente o valor ‘0’; foi informado que o valor limite para esta variável ‘180’ (pois foi colocado no código a instrução ‘posicao < 181’, ou seja, menor do que 181 – e como a variável é inteira, menor do que 181 significa que o maior valor será 180, um inteiro antes); e, foi estabelecido que a variável ‘posicao’ será incrementada de um em um (‘posicao++’, que significa ‘some 1 ao valor atual’) a cada vez que o programa passar pelo ‘loop’. Ao final da linha 18, o ‘{’ abre um bloco de comandos (que será fechado, ‘}’, na linha 21).

Na linha 19 enviamos ao servo (‘meuServo.write(posicao)’) o valor atual da variável ‘posicao’, definida dentro do comando ‘for’ da linha 18.

Na linha 20 inserimos uma espera de 20 milissegundos para evitar instabilidade do servo motor. Você pode testar outros valores, como 15 milissegundos ou 30 milissegundos, verificando o que fica melhor para o seu servo motor.

DESAFIO: quando o *loop* termina o servo motor volta à posição 0 (zero) e reinicia o ciclo. Seu desafio é fazer com que o servo motor volte à posição inicial retrocedendo de 1 em 1 grau, assim como foi feito para avançar.

Dica: ‘posicao--’, começando com o maior valor e terminado com o menor, ou seja, agir na estrutura do comando ‘for’ fazendo-o funcionar o servo motor ‘ao contrário’.