

Componentes Eletronicos

Guia do Motor de Passo 28BYJ-48 + Driver ULN2003

Eletrogate 23 de julho de 2018

Atualizado em: 09 set 2022

9 min

Nesse Guia Completo do Motor de Passo 28BYJ-48 + Driver ULN2003, Você Verá

Introdução

Essa montagem, eu diria que é a mais simples para controle de um [motor](#) de passo (stepper motor). Esse motorzinho tem uma redução interna que aumenta bem o torque, apesar do pequeno tamanho. O módulo de controle com o chip ULN2003 é bem pequeno também, o que pode facilitar a montagem do circuito.

As vantagens são a simplicidade na montagem , Sketch pode ser bem enxuto, o uso de tensão de 5V para alimentar o motor, etc.

A desvantagens são a rotação baixa do motor (devido a redução na caixa de engrenagens) , o motor unipolar é menos eficiente do que o bipolar e o uso de quatro portas do [Arduino](#). Alguns módulos de controle de motor de passo usam interfaces com duas portas apenas (por exemplo, com interface I2C).

Se você quiser saber mais sobre motores de passo, recomendo a leitura desse outro tutorial :

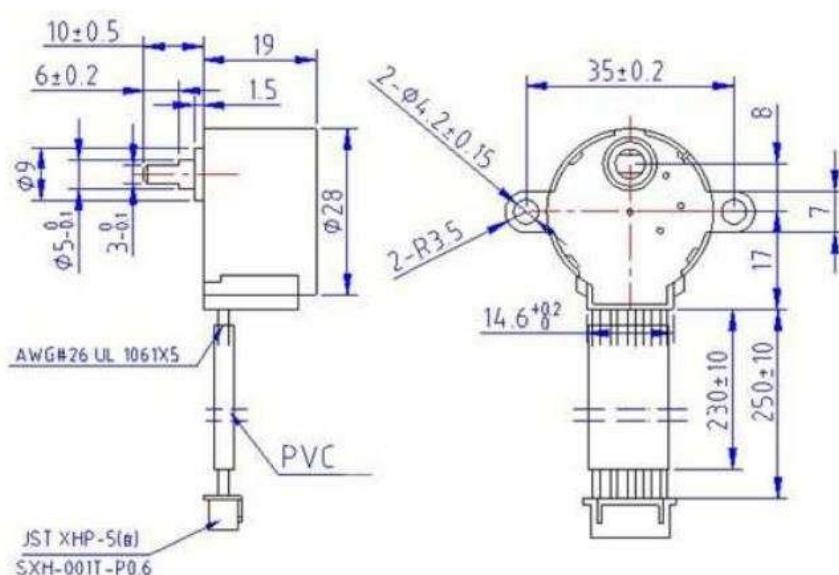
[Tudo sobre DRIVER A4988 e Motor de Passo > Usando o Arduino](#)

Informações Sobre o Motor de Passo 28BYJ-48

Esse é mais um motorzinho fabricado na China. Ele deve ser produzido em larga escala e por isso ele tem um preço bem acessível. Interessante , que ele pode ser alimentado com 5V e consome baixa corrente, o que facilita a montagem. Esse motor é Unipolar pois possui 4 enrolamentos que chamamos de fases. Em uma das pontas das fases, todas estão conectadas juntas. Portanto, esse motor não pode ser usado em Drivers para motores Bipolares (com duas fases somente).

Essas são as especificações do Motor 28BYJ-48:

[Motor28BYJ48Kiatronics](#)



A distância entre os dois suportes de parafusos é de 35 mm. O diâmetro do motor é de 28 mm e a profundidade é de 19 mm. O diâmetro do eixo é 5mm, e ele é chanfrado.

Tensão de operação : 5V CC

Número de fases : 4

Razão da variação de velocidade : 1/64 (mecanismo de redução)

Angulo do passo : 5,625 graus => 64 passos/volta ($360/64=5,625$)

Resistência CC : 50 ohms

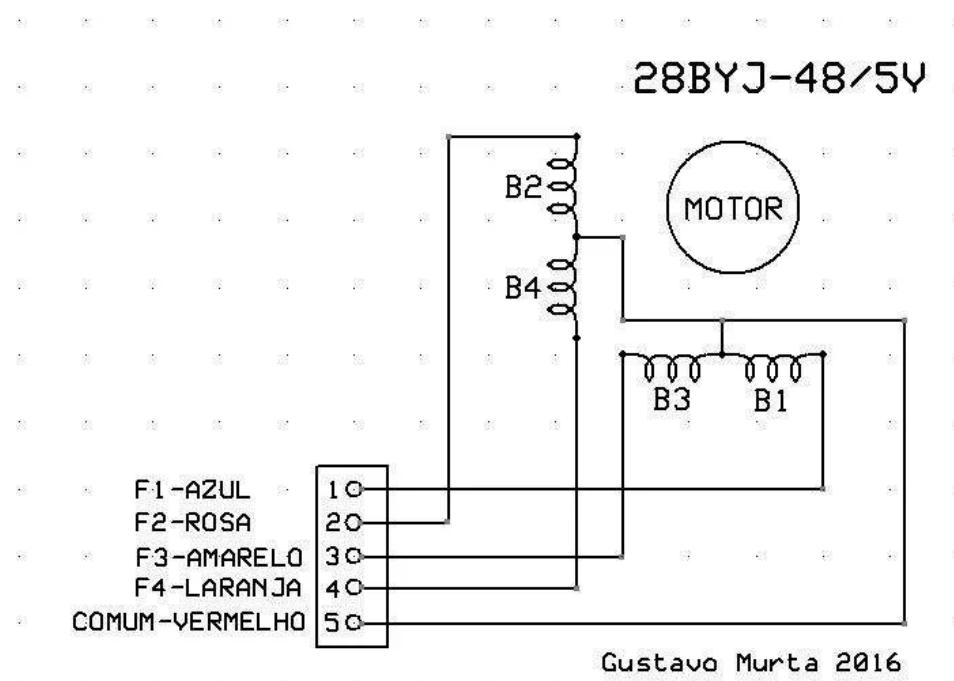
Frequência : 100 Hz

Torque de tração: > 34,3 mN.m

OBS: a resistência medida nas bobinas do motor foi de 23 ohms. Assim podemos deduzir o consumo estático de corrente em cada bobina :

$$I = V / R = 5V / 23 = 217 \text{ mA} \text{ aproximadamente (a corrente medida foi um valor aproximado disso)}$$

Esse é o diagrama do motor (desenhado por mim) :



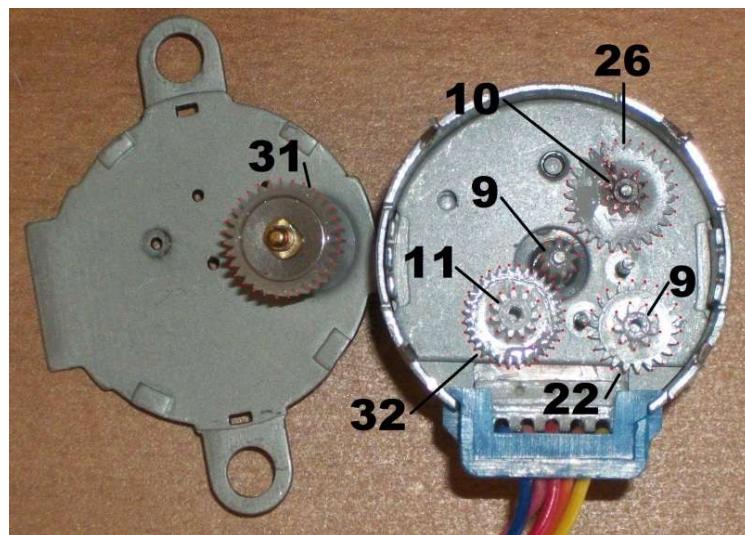
Para entender o mecanismo de redução do motor, veja esse excelente video :

THE 28BYJ-48 MOTOR AND ULN2003 DRIVER



No Forum do Arduino encontrei um tópico onde desmontaram o motor, para entender como funciona a caixa de redução de engrenagens. Muito curioso !

Geared Stepper Motor



Como podem ver, a caixa tem várias engrenagens (31 dentes, 32 dentes, 26 dentes, 22 dentes, 11 dentes, 10 dentes, e mais duas com 9 dentes).

O pessoal fez alguns cálculos para determinar com maior precisão qual era a redução.

$$(31 \times 32 \times 26 \times 22) / (11 \times 10 \times 9 \times 9) = 283712 / 4455 = 25792 / 405 = 63,68395...$$

Na especificação do fabricante, ele aproximou o valor para 64.

Para calcular o número de passos do motor interno para girar uma volta do eixo externo (com redução) :

$$(64 \times 25792) / 405 = 4075,7728395.$$

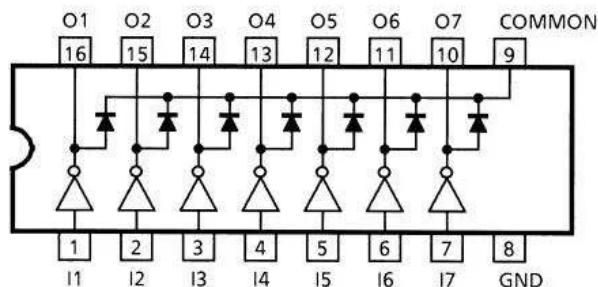
Isto é , são necessários aproximadamente 4075 passos no motor de passo interno, para uma volta no eixo externo. Esse valor pode ter uma pequena variação devido às folgas nas engrenagens.

Informações Sobre o Módulo Driver ULN2003

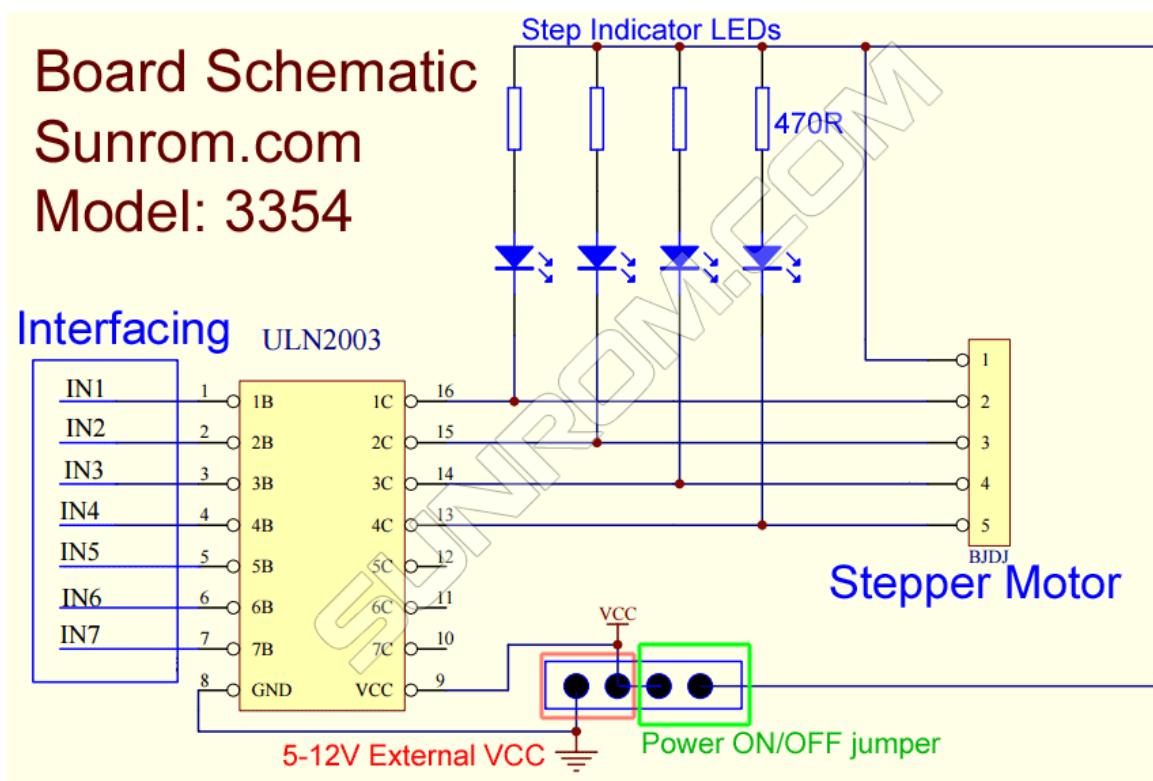
O único chip no módulo é o ULN2003. Esse chip possui um conjunto de sete drivers de transistores Darlington que permitem o acionamento de cargas indutivas. Todas as saídas tem o coletor aberto e **diodos** de supressão (Clamp). Os transistores suportam tensões de até 50V e correntes de até 500 mA. Todas as entradas IN1, IN2, IN3 e IN4 são compatíveis com sinais TTL e CMOS, com limite de 5V. O pino comum tem que ser conectado na tensão de alimentação do motor. Nesse caso é conectado no 5V.

[Datasheet ULN2003APG](#)

Essa é disposição dos circuitos drivers no chip ULN2003APG:



Esse é o diagrama da placa de circuito do Módulo ULN2003APG :



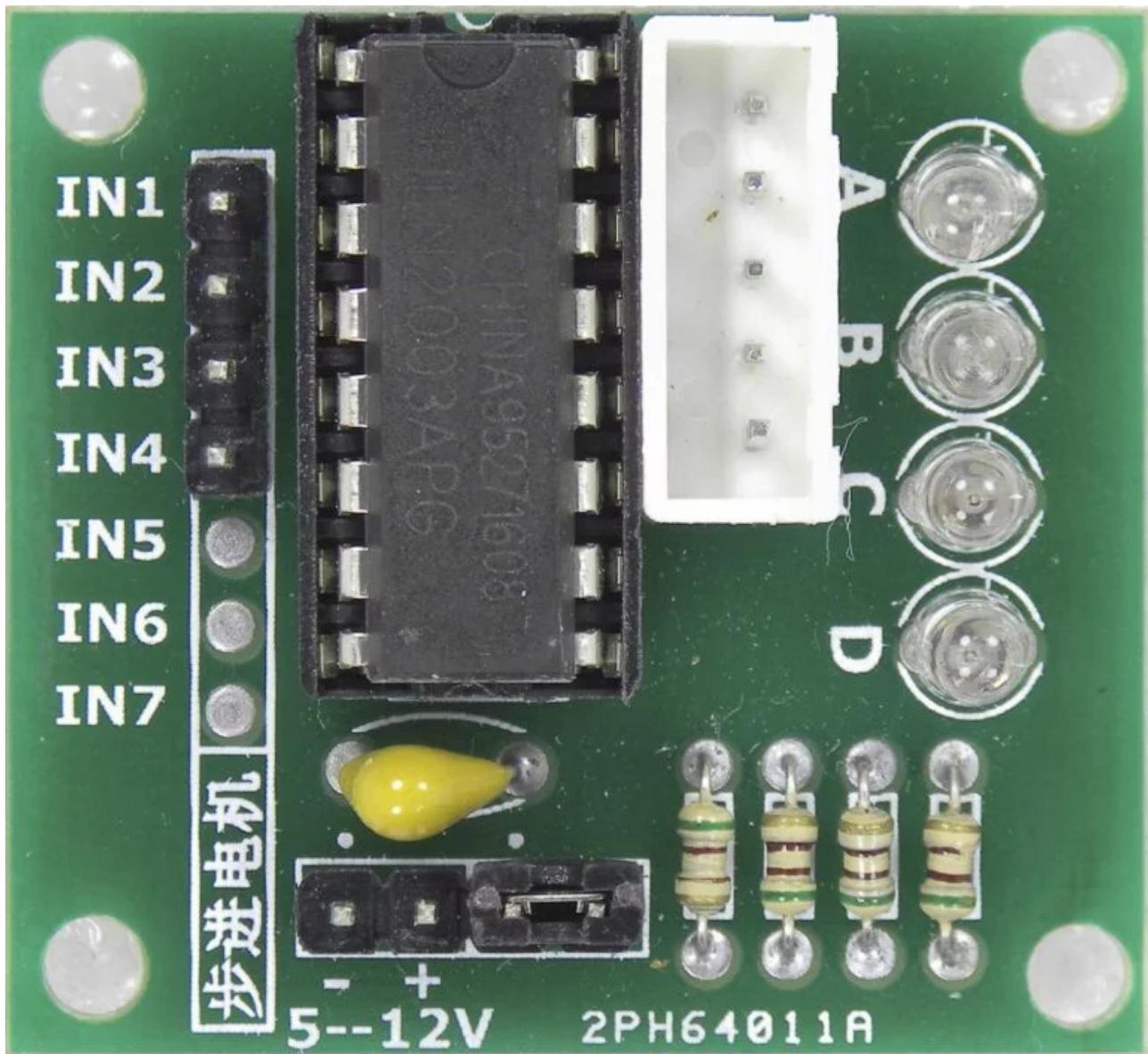
Os quatro leds vermelhos (A,B,C e D) são usados para indicar o acionamento de cada um dos drivers (fases do motor). Mesmo que o diagrama mostre sete entradas, somente quatro podem ser usadas e somente essas tem pinos no conector.

Muita atenção ao conectar a alimentação nos pinos.

O pino mais à esquerda (-) no módulo é o terra (veja foto abaixo). Esse terra tem que ser conectado ao terra da fonte e ao terra do Arduino. O pino (+) no caso do

Motor 28BYJ-48, tem que ser conectado ao positivo de uma fonte de 5V (preferencialmente de 1 Ampére). Não recomendo que conecte no 5V do Arduino, pois poderá sobrecarregar o regulador de tensão do mesmo.

O jumper Power ON/OFF é usado para ligar ou desligar o motor.



Módulo ULN 2003 – foto Gustavo Murta

Motor de Passo - Modos de Operação

Nesse tutorial o Motor de passo é Unipolar, isto é, ele possui quatro enrolamentos que chamamos de Fases.

Cada circuito driver do chip ULN2003A aciona uma das fases. E a ativação de cada driver é realizada pelas portas digitais do Arduino (portas D08, D09, D10 e D11) .

F1 (azul) – D08

F2 (rosa) – D09

F3 (amarelo) – D10

F4 (laranja) – D11

Para um motor de Passo Unipolar, temos alguns modos de operação. O modo Passo completo com alto torque (Full step), o modo Passo completo com baixo torque (Wave Step), o modo Meio Passo (half step) e Micro-passo (Micro stepping).

No caso do Micro passo, a corrente nos enrolamentos deve ser alterada. Como o nosso circuito não permite o controle da corrente, esse modo de operação não se aplica.

Para entender como as Fases são acionadas em cada caso, vejam essas cartas de tempo:

Obs: 0 (zero) = enrolamento desativado e 1 = enrolamento ativo.

Nas linhas estão a sequência dos passos e nas colunas estão as Fases.

Passo completo com alto torque (Full step):

Duas Fases são acionadas ao mesmo tempo.

Step	ϕ_4	ϕ_3	ϕ_2	ϕ_1
0	0	0	1	1
1	0	1	1	0
2	1	1	0	0
3	1	0	0	1

Passo completo com baixo torque (Wave Step):

Somente uma Fase acionada de cada vez.

Step	ϕ_4	ϕ_3	ϕ_2	ϕ_1
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

Meio Passo (half step):

Na sequência de oito passos, em alguns passos temos somente uma fase acionada e em outros passos, temos duas fases acionadas.

Step	ϕ_4	ϕ_3	ϕ_2	ϕ_1
0	0	0	0	1
1	0	0	1	1
2	0	0	1	0
3	0	1	1	0
4	0	1	0	0
5	1	1	0	0
6	1	0	0	0
7	1	0	0	1

Referência : [Stepper_Motors_2011.pdf](#)

Montagem do Motor e do Driver com Arduino

Esse é o diagrama da montagem do circuito com Arduino.

Materiais necessários para a montagem do projeto com Motor de Passo 28BYJ-48 + Driver ULN2003 e Arduino Uno

1x [Uno R3 + Cabo Usb para Arduino](#)

1x [Protoboard 400 pontos](#)

1x [Jumpers – Macho/Femea – 20 Unidades de 20cm](#)

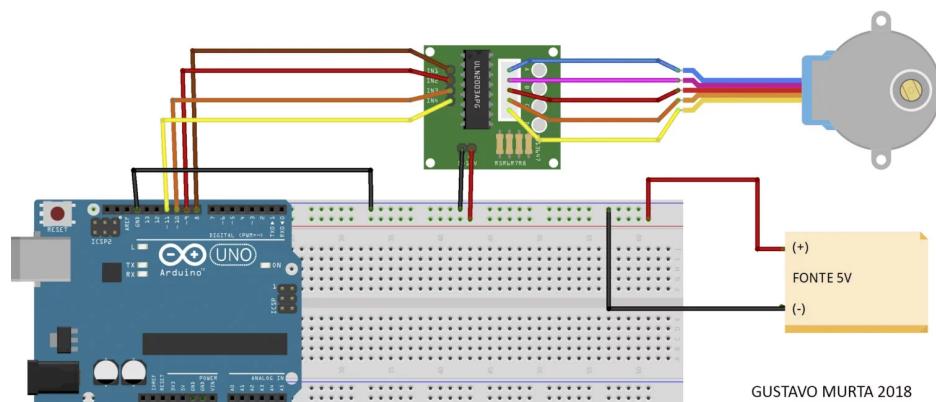
1x [Motor De Passo + Módulo De Controle \(driver ULN2003\)](#)

1x [Fonte 5V 1A Bivolt](#)

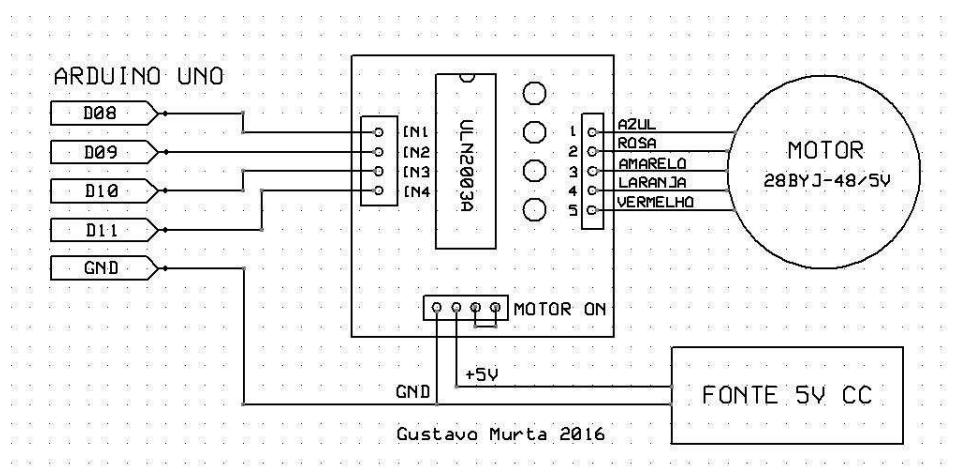
1x [Módulo Conector Jack P4 Femea \(J4 2.1mm\)](#)



Obs: como já informei, recomendo que use uma fonte externa de 5V para alimentar o Driver + Motor de passo.



Esse é o diagrama eletrônico do mesmo circuito:



Modo de Passo Completo com Alto Torque

No Modo de Passo completo com alto torque, para cada passo duas fases são ativadas simultaneamente. Esse modo é o mais usado, pois como o nome já diz,

tem mais torque. O consumo de corrente é mais alto, devido à ativação das duas fases.

Um dos motivos para a escolha desse Módulo driver ULN2003 é que ele possui quatro leds que permitem a indicação da ativação das fases do motor. Para fins didáticos, isso é muito útil.

Essa é a Carta de Tempo do Modo Passo completo com alto torque (FullStep):

Veja que para cada passo, duas Fases (Channel) são ativadas ao mesmo tempo.



Para melhor visualizar o avanço dos Leds piscando de acordo com a ativação das Fases, altere a variável atraso_fase para 500 milisegundos :

```
int atraso_fase = 500 ;
```

Fica visível que dois Leds piscam ao tempo.

Verificando os valores dos Bytes das matrizes do programa : AHO e HOR , percebe-se que dois bits são ativados ao mesmo tempo. E é claro, uma sequência de bytes é inversa da outra.

[Full_step.ino](#)

```
1. // Controle de Motor de Passo - Modo Passo Completo alto torque (Full step)
2. // Blog Eletrogate - https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003
3. // Baseado em http://www.elecrow.com/wiki/index.php?title=ULN2003\_Stepper\_Motor\_Driver
4. // Motor 28BYJ48/5V com Módulo ULN20023 - Arduino Nano / IDE 1.8.5
5. // Uma volta no eixo = 4075 pulsos / 512 x 8 = 4096
6. // Gustavo Murta 23/jul/2018
7.
8. byte HOR[4] = {0x09,0x03,0x06,0x0C};      // Matriz dos bytes das Fases do Motor - sentido Horário Full Step
9. byte AHO[4] = {0x0C,0x06,0x03,0x09};      // Matriz dos bytes das Fases do Motor - sentido Anti-Horário Full Step
10. int atraso_fase = 2 ;                      // Intervalo de tempo entre as fases em milisegundos - min 2 para Full Step
11. int intervalo = 1000 ;                     // Intervalo de tempo entre os movimentos do motor em ms
12.
13. void Motor_AHO()                         // Movimento no sentido anti-horário
14. {
15.     for(int i = 0; i < 512; i++)          // incrementa o contador i de 0 a 511 - uma volta
16.
17.         for(int j = 0; j < 4; j++)        // incrementa o contador j de 0 a 3
18.         {
19.             PORTB = AHO[j];            // Carrega bytes da Matriz AHO na Porta B
20.             delay (atraso_fase);       // Atraso de tempo entre as fases em milisegundos
21.         }
22.     }
23.
24. void Motor_HOR()                         // Movimento no sentido horário
25. {
26.     for(int i = 0; i < 512; i++)          // incrementa o contador i de 0 a 511 - uma volta
27.
28.         for(int j = 0; j < 4; j++)        // incrementa o contador j de 0 a 3
29.         {
30.             PORTB = HOR[j];            // Carrega bytes da Matriz HOR na Porta B
31.             delay (atraso_fase);       // Atraso de tempo entre as fases em milisegundos
32.         }
33.     }
34.
35. void setup()
36. {
37.     DDRB = 0x0F;                          // Configura Portas D08,D09,D10 e D11 como saída
38.     PORTB = 0x00;                         // Reset dos bits da Porta B (D08 a D15)
```

```

39. }
40.
41. void loop()
42. {
43.   Motor_HOR();           // Gira motor no sentido Horário
44.   delay (intervalo);    // Atraso em milisegundos
45.   Motor_AHO();          // Gira motor no sentido Anti-Horário
46.   delay (intervalo);    // Atraso em milisegundos
47. }
```

Referências:

http://www.elecrow.com/wiki/index.php?title=ULN2003_Stepper_Motor_D...

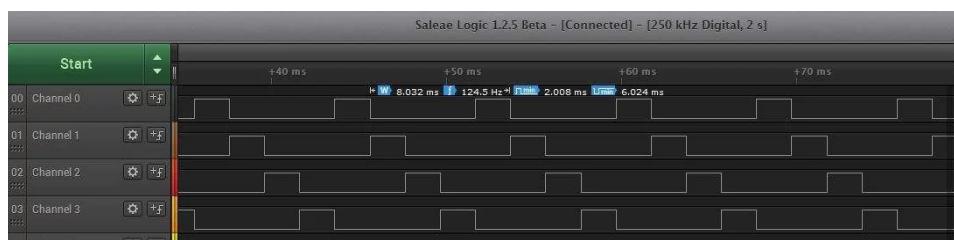
Modo de Passo completo com baixo torque (Wave Step):

Como já foi dito, no modo Passo completo com baixo torque (Wave step) somente uma fase é acionada de cada vez, portanto o torque do motor é menor do que nos outros modos. Esse modo é pouco usado exatamente por isso. O rendimento do motor é menor, mas tem-se como vantagem um menor consumo de corrente, já que somente uma fase é ativada em cada passo.

Para melhor visualizar o avanço dos Leds piscando de acordo com a ativação das Fases, altere a variável atraso-fase para 500 milisegundos :

```
int atraso_fase = 500 ;
```

Desse modo poderá perceber que os Leds piscam um de cada vez , confirmando a sequência da Carta de Tempo:



Importante ressaltar que nas matrizes do programa : AHO e HOR , estão carregados os bytes de ativação das fases, de acordo com a tabela anexada no tópico anterior. Uma sequência de bytes é inversa da outra.

[wave_step.ino](#)

```

1. // Controle de Motor de Passo - Modo Passo Completo baixo torque (Wave step)
2. // Blog Eletrogate - https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003
3. // Baseado em http://www.elecrow.com/wiki/index.php?title=ULN2003\_Stepper\_Motor\_Driver
4. // Motor 28BYJ48/5V com Módulo ULN20023 - Arduino Nano / IDE 1.8.5
5. // Uma volta no eixo = 4075 pulsos / 512 x 8 = 4096
6. // Gustavo Murta 23/julho/2018
7.
8. byte HOR[4] = {0x01,0x02,0x04,0x08};      // Matriz dos bytes das Fases do Motor - sentido Horário Wave Step
9. byte AHO[4] = {0x08,0x04,0x02,0x01};      // Matriz dos bytes das Fases do Motor - sentido Anti-Horário WaveStep
10. int atraso_fase = 2 ;                      // Intervalo de tempo entre as fases em milisegundos - min 2 para WaveStep
11. int intervalo = 1000 ;                     // Intervalo de tempo entre os movimentos do motor em ms
12.
13. void Motor_AHO()                         // Movimento no sentido anti-horário
14. {
15.   for(int i = 0; i < 512; i++)           // incrementa o contador i de 0 a 511 - uma volta
16.
17.     for(int j = 0; j < 4; j++)           // incrementa o contador j de 0 a 3
18.     {
19.       PORTB = AHO[j];                  // Carrega bytes da Matriz AHO na Porta B
20.       delay (atraso_fase);            // Atraso de tempo entre as fases em milisegundos
21.     }
22. }
23.
24. void Motor_HOR()                         // Movimento no sentido horário
25. {
26.   for(int i = 0; i < 512; i++)           // incrementa o contador i de 0 a 511 - uma volta
27.
28.     for(int j = 0; j < 4; j++)           // incrementa o contador j de 0 a 3
29.     {
30.       PORTB = HOR[j];                  // Carrega bytes da Matriz HOR na Porta B
31.       delay (atraso_fase);            // Atraso de tempo entre as fases em milisegundos
32.     }
}
```

```

33. }
34.
35. void setup()
36. {
37.     DDRB = 0x0F;           // Configura Portas D08,D09,D10 e D11 como saída
38.     PORTB = 0x00;          // Reset dos bits da Porta B (D08 a D15)
39. }
40.
41. void loop()
42. {
43.     Motor_HOR();          // Gira motor no sentido Horário
44.     delay (intervalo);    // Atraso em milisegundos
45.     Motor_AHO();          // Gira motor no sentido Anti-Horário
46.     delay (intervalo);    // Atraso em milisegundos
47. }

```

Modo de Meio Passo (Half Step):

O Modo Meio Passo é uma mesclagem dos outros dois modos, Full Step e Wave Step.

Esse modo é usado quando o movimento do motor precisa ser delicado e preciso e o torque não é importante.

O motor interno precisa de 64 passos para uma revolução (volta), no modo Half Step o número de passos dobra para 128 passos !

Por isso é chamado de Meio Passo. O efeito Meio passo é conseguido intercalando um passo (com duas fases do Motor ativadas simultaneamente) com outro passo (ativando somente uma fase), obedecendo uma sequência própria para isso. Devido à essa alternância, o torque e o consumo de corrente variam de acordo com o passo. Com mais fases ativas, terá maior torque e maior consumo de corrente.

O interessante nesse modo é que as matrizes dos Bytes AHO e HOR , tem oito valores e não somente quatro como nos outros modos. Analisando os Bytes, dá para perceber que as matrizes dos outros modos foram agrupadas.

Half_step.ino

```

1. // Controle de Motor de Passo - Modo Meio Passo (Half step)
2. // Blog Eletrogate - https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003
3. // Baseado em http://www.elecrow.com/wiki/index.php?title=ULN2003\_Stepper\_Motor\_Driver
4. // Motor 28BYJ48/5V com Módulo ULN20023 - Arduino Nano / IDE 1.8.5
5. // Uma volta no eixo = 4075 pulsos / 512 x 8 = 4096
6. // Gustavo Murta 23/jul/2018
7.
8. byte HOR[8] = {0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08};      // Matriz dos bytes das Fases do Motor - sentido I
9. byte AHO[8] = {0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};      // Matriz dos bytes das Fases do Motor - sentido ,
10. int atraso_fase = 1 ;                                         // Intervalo de tempo entre as fases em milisegundos
11. int intervalo = 1000 ;                                         // Intervalo de tempo entre os movimentos do motor
12.
13. void Motor_AHO()          // Movimento no sentido anti-horário
14. {
15.     for(int i = 0; i < 512; i++) // incrementa o contador i de 0 a 511 - uma volta
16.
17.         for(int j = 0; j < 8; j++) // incrementa o contador j de 0 a 7
18.         {
19.             PORTB = AHO[j];        // Carrega bytes da Matriz AHO na Porta B
20.             delay (atraso_fase);   // Atraso de tempo entre as fases em milisegundos
21.         }
22. }
23.
24. void Motor_HOR()          // Movimento no sentido horário
25. {
26.     for(int i = 0; i < 512; i++) // incrementa o contador i de 0 a 511 - uma volta
27.
28.         for(int j = 0; j < 8; j++) // incrementa o contador j de 0 a 7
29.         {
30.             PORTB = HOR[j];        // Carrega bytes da Matriz HOR na Porta B
31.             delay (atraso_fase);   // Atraso de tempo entre as fases em milisegundos
32.         }
33. }
34.
35. void setup()
36. {
37.     DDRB = 0x0F;           // Configura Portas D08,D09,D10 e D11 como saída
38.     PORTB = 0x00;          // Reset dos bits da Porta B (D08 a D15)
39. }
40.
41. void loop()

```

```

42.  {
43.   Motor_HOR();           // Gira motor no sentido Horário
44.   delay (intervalo);    // Atraso em milisegundos
45.   Motor_AHO();          // Gira motor no sentido Anti-Horário
46.   delay (intervalo);    // Atraso em milisegundos
47. }

```

Essa é a Carta de Tempo do Modo Meio Passo (Half Step):

Em alguns passos, tem duas fases ativadas e em outros tem somente uma fase ativada .

Bibliotecas para Motor de passo:

Existem várias bibliotecas para o controle de motor de passo com o Arduino. As mais comuns são essas:

[Biblioteca Stepper](#)

[Biblioteca AccelStepper](#)

[Biblioteca CustomStepper](#)

Exemplo de Sketch com a Biblioteca AccelStepper. Gire o potenciômetro, e o motor irá acompanhar o movimento do mesmo (Giro total de 180 graus do eixo do motor). Use um potenciômetro de 10 K ohms.

Diagrama do circuito :

[AccelStepperPOT.ino](#)

```

1. // Controle de Motor de Passo com Potenciometro
2. // Blog Eletrogate - https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003
3. // Baseado em http://www.airspace.com/mikem/arduino/AccelStepper/ProportionalControl\_8pde-example.html
4. // Motor 28BYJ48/5V com Modulo ULN20023 - Arduino UNO / IDE 1.8.5
5. // Uma volta no eixo = 4075 pulsos
6. // Gustavo Murta 02/jago/2018

```

```

7.
8. #include <AccelStepper.h>                                // biblioteca AccelStepper
9.
10. #define ANALOG_PIN A0                                     // pino A0 para leitura da tensão do Potenciômetro
11.
12. AccelStepper motorPasso (AccelStepper::FULL4WIRE, 8, 10, 9, 11); // Passo completo 4 fios
13.
14. void setup()
15. {
16.     motorPasso.setMaxSpeed(500);                           // maxima velocidade = 500 pulsos por segundo
17. }
18.
19. void loop()
20. {
21.     int analog_in = analogRead(ANALOG_PIN);                // lendo a tensão do pino A0 do Arduino
22.     motorPasso.moveTo(analog_in);                          // gira o eixo do motor X pulsos (0 a 1023)
23.     motorPasso.setSpeed(100);                             // velocidade = 100 pulsos por segundo
24.     motorPasso.runSpeedToPosition();                      // gira o eixo para a posição definida
25. }

```

Aplicações Interessantes para o Motor de Passo 28BYJ-48 + Driver ULN2003

Esse motor de passo permite movimentos precisos e com baixa rotação. Essas são algumas aplicações interessantes voltadas para fotografia:

[ARDUINO TIME-LAPSE PANORAMA CONTROLLER](#)