Universidade do Minho

Escola de Engenharia
Departamento de Engenharia Mecânica

Mestrado em Engenharia Mecatrónica

Motor de Passo - Arduino

Unidade de Crédito Máquinas de Comando Numérico

20 643 José António Barbosa Goncalves
20 242 André Lourenço Caldeira Pinto

Docente: Caetano Monteiro

26 Julho 2012

Índice

Introdução

O trabalho consiste em comandar o eixo de um mecanismo utilizando um motor de passo e uma placa Arduino. Para simulação foi utilizada uma velha impressora em que é comandado o motor que controlava a posição do tinteiro. Neste documento será explicado o funcionamento do motor de passo, e como o controlar, de modo a entender o código de programação da placa Arduino. O controlo de dois eixos resume-se basicamente em duplicar o trabalho aqui realizado.

1. Funcionamento do motor de Passo

Tal como a grande maioria dos motores elétricos, o motor de passo trabalha pela interação entre campos eletromagnéticos. Quando um solenoide é percorrido por uma corrente constante, dá-se como resultado a produção de um campo magnético uniforme no seu interior. Invertendo o sentido da corrente elétrica inverte-se o sentido do campo magnético.

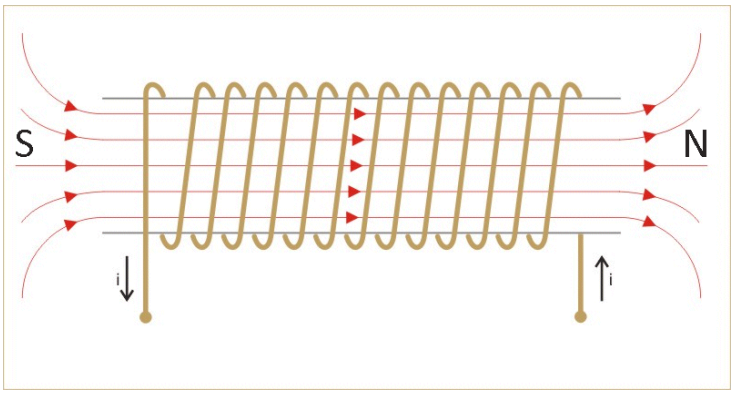


Figura 1

Combinando vários solenóides, e um íman permanente posicionado entre os mesmos, pode-se controlar a posição do íman permanente, como se observa em alguns exemplos na figura 2.

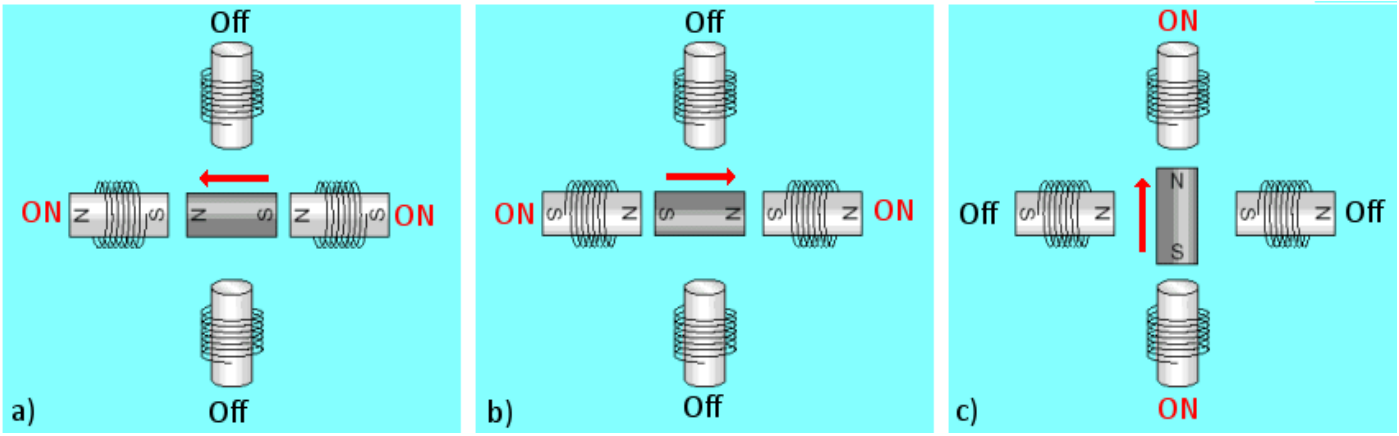


Figura 2

Por consequente com a alimentação sequencial de forma adequada dos solenóides, pode-se obter um movimento de rotação do íman permanente. Este é o princípio de funcionamento do motor de passo de íman permanente que se utilizou no desenvolvimento deste trabalho.

Tipos de motores de passo:

Os motores de passo podem ser divididos em três tipos:
- Relutância variável.
- Íman permanente.
- Híbridos, este combina os princípios operacionais dos dois outros tipos.

Os motores de passo híbridos são de longe o motor de passo mais utilizado em aplicações industriais, de tal modo que é o tipo de motor considerado para o trabalho.

O estator do motor de passo híbrido é constituído por dois solenóides, sendo que em certos motores cada um tem 2 fios de ligação, um para cada extremidade, enquanto outros poderão ter 3 fios de ligação, sendo um deles intermédio. Nas figuras 3a e 3b podemos observar estes dois tipos de ligações.

A ligação dos motores de passo com 4 cabos, chamados de motores de passo bipolar é mostrada na figura 3a. Observa-se dois solenóides, 1 e 2, estes dividem-se em duas partes no esquema, (1a + 1b) e (2a + 2b), mas na prática podemos considerar como sendo somente o solenóide 1 e o solenóide 2, pois 1a liga a 1b, e 2a liga a 2b. É chamado de motor bipolar pois para inverter a polaridade do campo magnético, é invertida a tensão aplicada nos solenóides. A tensão aplicada às bobines tem portanto duas polaridades.

A ligação dos motores de 6 cabos, chamados de motores de passo unipolares é esquematizada na figura 3b. Normalmente o ponto intermédio de cada solenóide é ligado à massa, e é aplicada uma tensão positiva numa das duas extremidade de cada solenóide. A polaridade do campo magnético dependerá de qual as extremidades é alimentada. Cada solenóide é parcialmente alimentado, alternando entre a parte 1a e 1b, e a parte 2a e 2b. O binário é reduzido, mas é possível utilizar em modo bipolar se a ligação intermédia for ignorada. Este tipo de funcionamento é chamado de unipolar pois a polaridade em cada solenoide é fixa, sendo que cada extremidade é ou não alimentada.

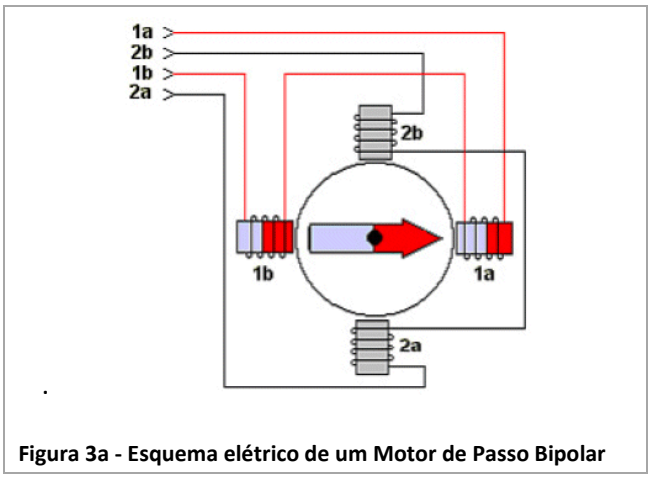


Figura 3a - Esquema elétrico de um Motor de Passo Bipolar

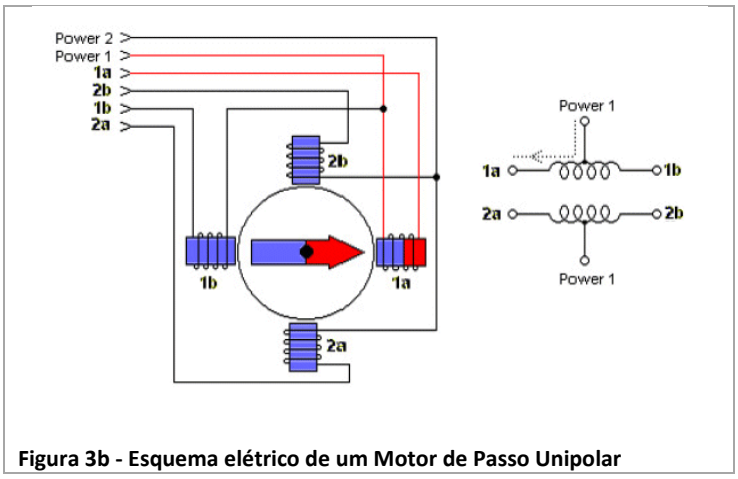


Figura 3b - Esquema elétrico de um Motor de Passo Unipolar

Nas figuras 4,5,6 e 7 são demonstrados os principais tipos básicos de movimentos, que se dividem em funcionamento de passo inteiro e o de meio passo.

Fica a nota de que existe ainda um terceiro modo de funcionamento chamado de o micropasso *(micro stepping)* que permite aumentar significativamente a exatidão do motor de passo ao aumentar o número de passos, mas que não é aqui abordado.

Unipolar, Passo completo (*full-step*)

Neste tipo de atuação os solenoides são parcialmente alimentados, com a sequência: 1a -> 2a -> 1b -> 2b.

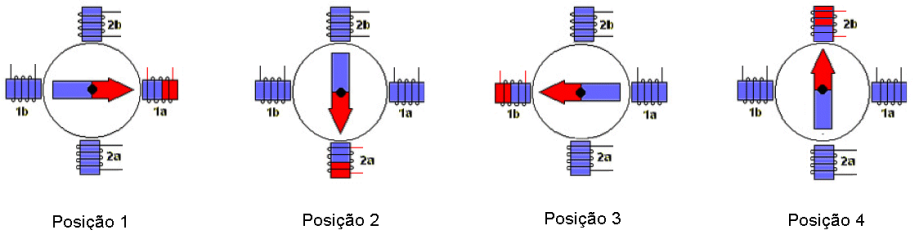


Figura 4

Bipolar passo completo (*full-step*)

Neste tipo de ação são atuados 1a e 1b ou 2a e 2b simultaneamente para cada passo, pelo que o binário produzido é maior, assim como o consumo de energia. É chamado de bipolar porque o sentido da corrente com que são alimentados os solenóides, ao contrário das atuações unipolares, não é sempre a mesma.

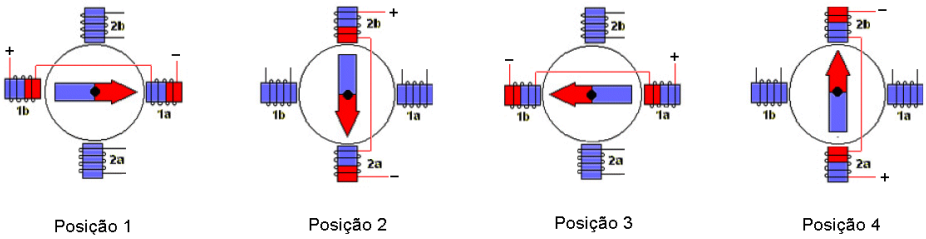


Figura 5

Unipolar, passo completo (*full-step*)

Neste tipo de ação também são atuados 2 enrolamentos em simultâneo para cada passo, mas agora com outra configuração.

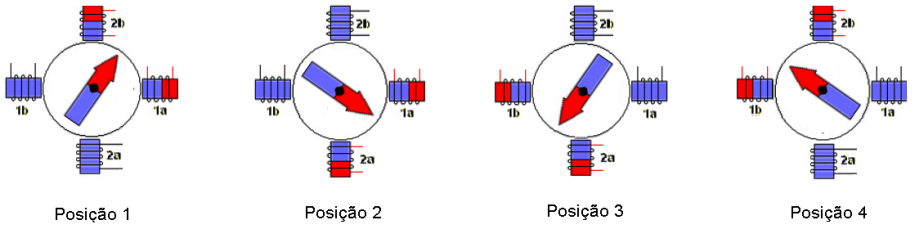


Figura 6

Unipolar, meio-passo (*half-step)*

Outro tipo de atuação possível consiste em alimentar alternadamente, um e dois enrolamentos, permitindo deste modo avançar meio passo de cada vez, duplicando assim o número de passos. Na figura só estão ilustrados 4 passos, apesar de serem necessários 8 passos para concluir uma volta completa.

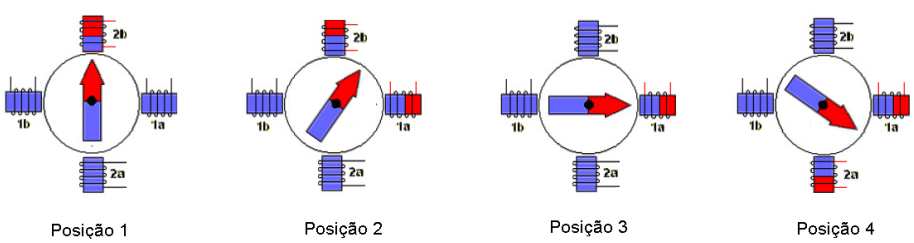


Figura 7

Motores de passo híbridos

Já foi visto basicamente o funcionamento do motor de passo híbrido, no entanto ira-se abordar a sua composição de modo a entender como é aumentado o número de passos por volta, pois 8 passos é muito insuficiente para a maioria das aplicações.

Este tipo de motor é caracterizado pelas seguintes características principais:
- Apresenta rotor e estator multidentados;
- Rotor é constituído por duas partes multidentadas;
- O rotor é de íman permanente, tem vários polos, e é magnetizado axialmente.

A figura 8 permite visualizar de um modo geral a constituição de um motor de passo híbrido.

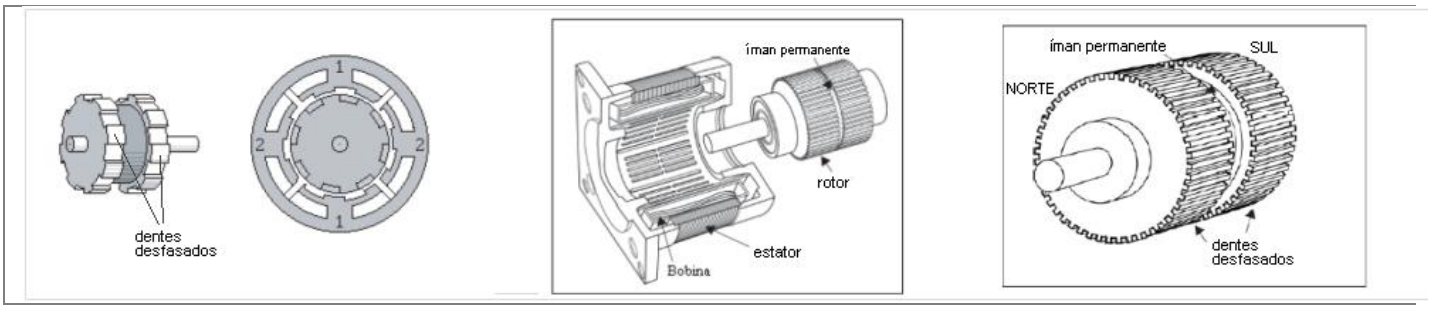


Figura 8

O estator do motor de passo híbrido mais comum é composto por dois solenóides. Na figura 9 podemos identificar o solenóide 1, composto pelas partes 1A e 1B, e o solenóide 2 composto pelas partes 2A e 2B, numa ligação bipolar.
O rotor de íman permanente é constituído por duas partes, neste exemplo com três dentes cada um. Uma parte constitui o polo Norte e outra o polo Sul, e ambas estão desfasadas entre si conforme mostra a figuras 9. O rotor multidentado faz com que este tenha vários polos, permitindo desdobrar o número de passos por rotação. Um rotor com um único polo Norte e outro Sul permitiria um número máximo de 8 passos como vimos na figura 7, o que limita a precisão. É comum um motor de passo com 200 passos, i. e., permite passos de 1.8º (360/200).

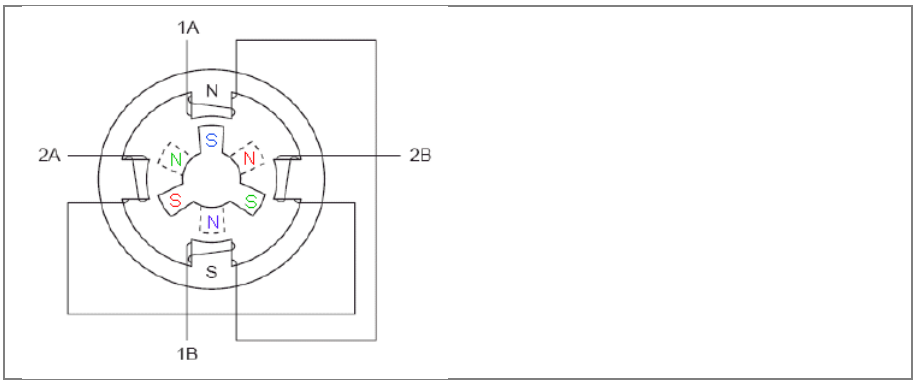


Figura 9 - Exemplo simples de um motor híbrido simples de 12 passos/revolução.

Na figura 10 está ilustrado o princípio de funcionamento de um motor de passo com rotor multidentado, o princípio é idêntico ao exemplificado nas figuras 4,5,6 e 7, com a diferença de que os passos são agora mais pequenos.

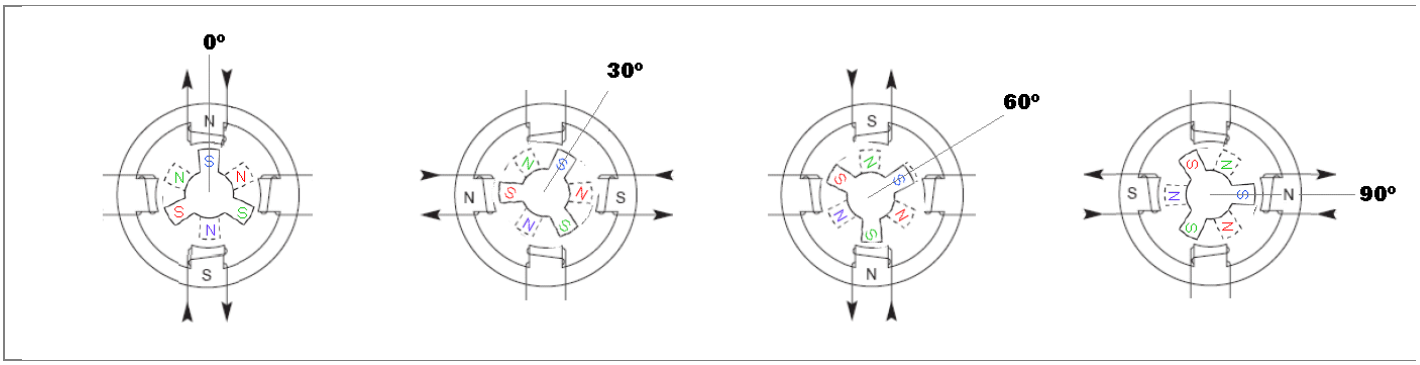


Figura 10

2. Controlo do motor de Passo – circuito de potência

Para que um motor de passo funcione, é necessário que sua alimentação seja feita de forma sequencial e repetida. Para que se obtenha uma rotação constante é necessário que a energização dos solenóides seja periódica. Esta periodicidade é proporcionada por circuitos electrónicos que controlam a velocidade e o sentido de rotação do motor.

Para alimentar os solenóides são normalmente utilizados circuitos elétricos designados de *ponte H*, ilustrado na figura 11a. O funcionamento consiste em fechar simultaneamente os interruptores *a* e *d* para que o solenóide seja alimentado por +V e GND, ou fechar os interruptores *b* e *c*, e neste caso teremos o solenóide também alimentado por +V e GND mas agora com polaridade inversa pelo que o campo gerado será invertido.

De notar que não se deve fechar simultaneamente os interruptores *a* e *b*, ou *c* e *d* simultaneamente, pois nessa condição cria-se um curto-circuito entre +V e GND que muito provavelmente danificará o circuito de comando.

Os interruptores foram utilizados no esquema para exemplificar o funcionamento, na prática são substituídos por transístores, como podemos ver na figura 11b. A utilização dos díodos é importante para proteger o circuito de comando das correntes indutivas geradas pelos solenóides.

Os transístores são controlados por sinais digitais (impulsos), aplicados nas bases dos transístores designadas na figura11b por Ba, Bb, Bc, e Bd. Um transístor é considerado fechado se tiver na base um sinal positivo de valor típico 5V designado por "1" na eletrónica digital, e é considerado aberto se tiver na sua base um sinal de 0V designado por "0" na eletrónica digital.

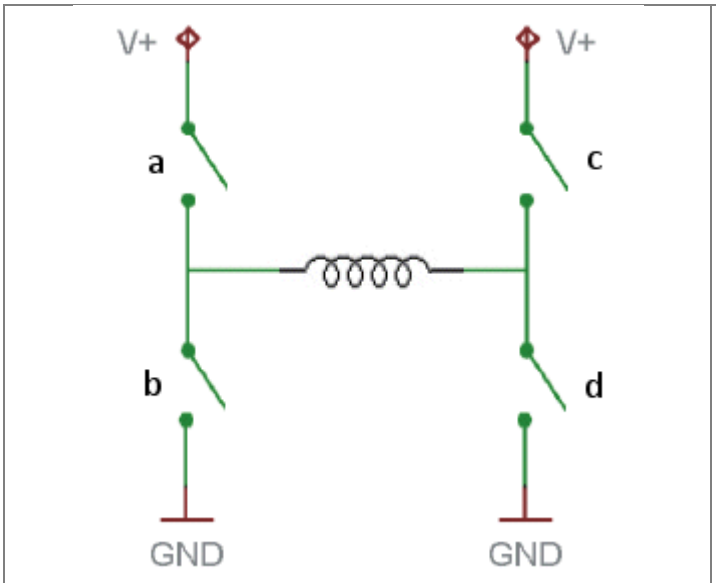


Figura 11a

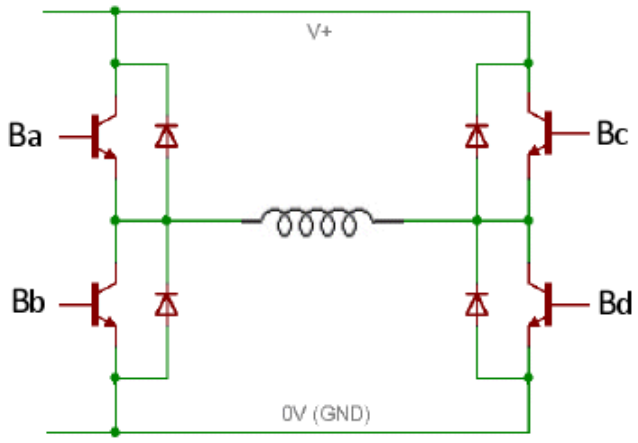


Figura 11b

O circuito da figura 12 representa o esquema elétrico do circuito integrado L298, este é composto por duas pontes H, ou seja permite comandar os dois solenoides de um motor de passo.

Na primeira ponte H o solenoide1 é ligado às saídas OUT1 e OUT2, e é comandado pelas entradas In1, In2 e EnA. Na segunda ponte H o solenóide2 liga às saídas OUT3 e OUT4, e é comandado pelas entradas In3, In4, e EnB.

Como se observa os transístores não são comandos diretamente pelas entradas In1, 2, 3 e 4, existe um circuito intermédio constituído por portas lógicas AND. Estas portas lógicas simplificam o controlo da ponte H. Considerando os pares de transístores (a1,b1), (c1,d1), (a2,b2), (c2,d2), para cada par o circuito lógico só permite que um dos transístores esteja ligado de cada vez, anulando assim o risco de curto-circuito como foi falado na página anterior.

O circuito lógico das portas AND é alimentado pela tensão +Vss que deve ser tipicamente de 5V (o valor lógico "1" equivale ao valor analógico +Vss, e o valor lógico 0 corresponde ao valor 0V), enquanto o circuito dos transístores designado de circuito de potência é alimentado pela tensão +Vs, e esta pode ter valor até 46V. Quanto maior a tensão +Vs, maior será o binário obtido pelo nosso motor, mas deve-se ter em atenção qual a tensão máxima suportada pelo motor utilizado.

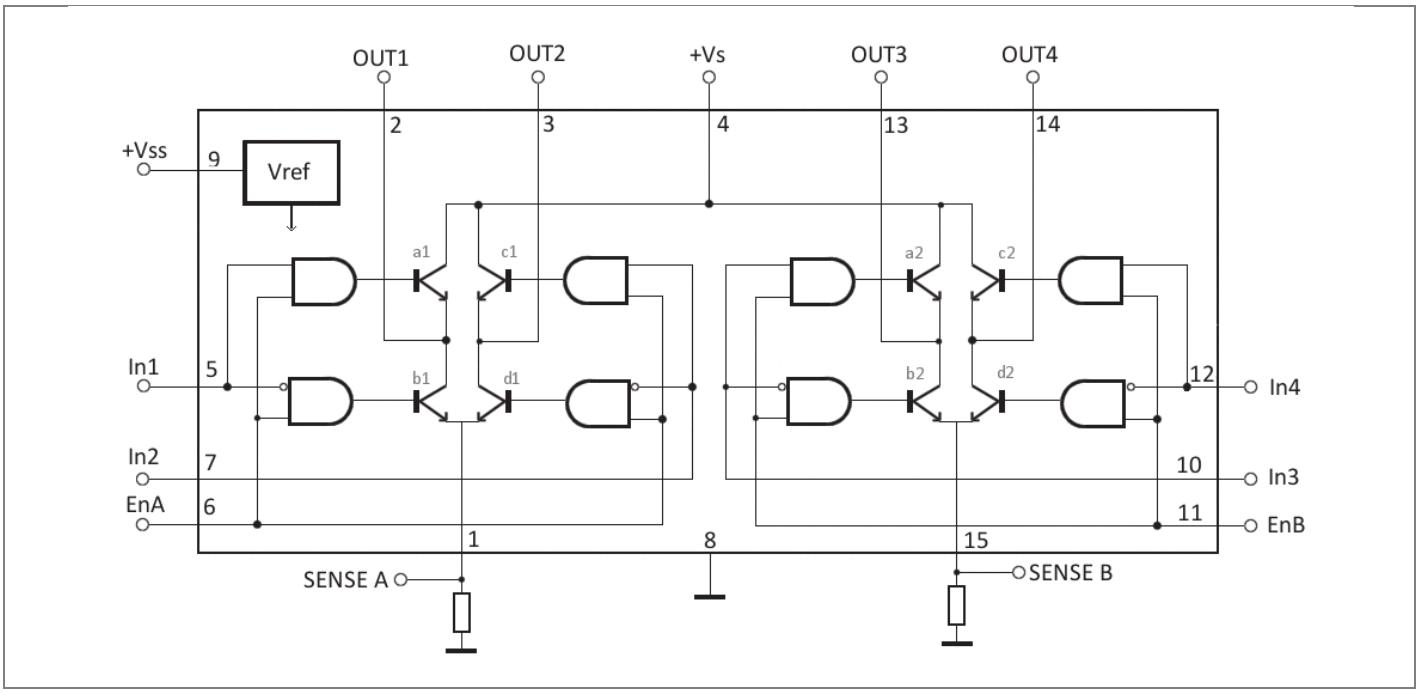


Figura 12 – Pin out de L298 para o package Multiwatt15

Existem dois tipos de encapsulamento ("package") para o circuito integrado L298. O encapsulamento chamado de Multiwatt15 que podemos ver na figura 13, e para o qual corresponde o *pinout* da figura 12. Ao encapsulamento PowerSO20, figura 15, corresponde o *pinou*t do esquema da figura 14.

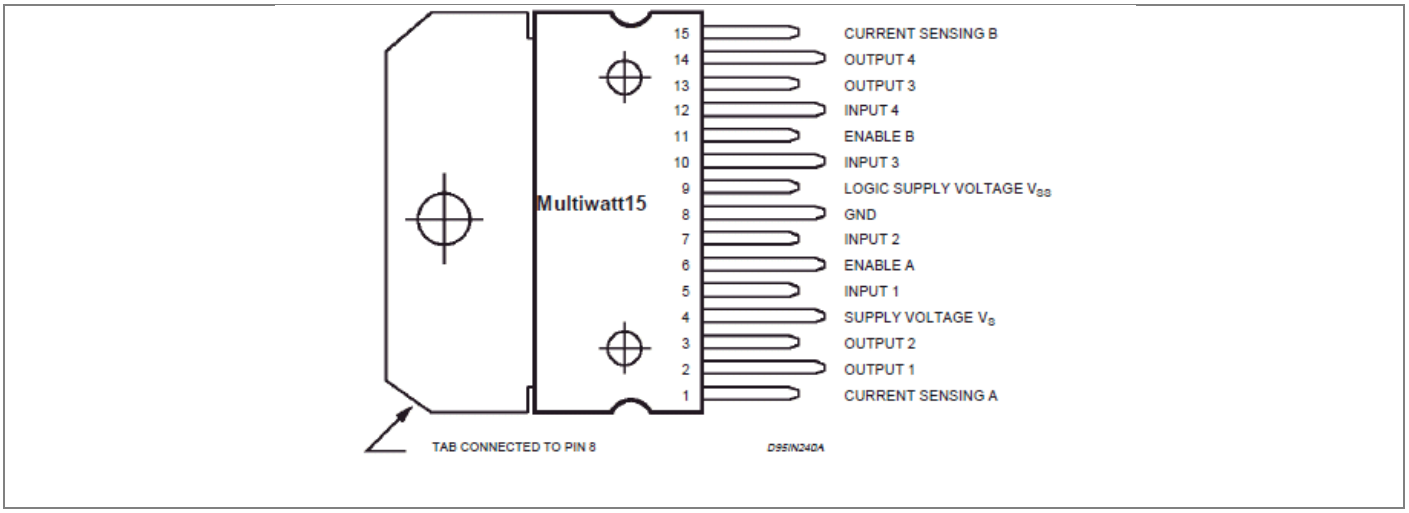


Figura 13 - L298, Package Multiwatt15

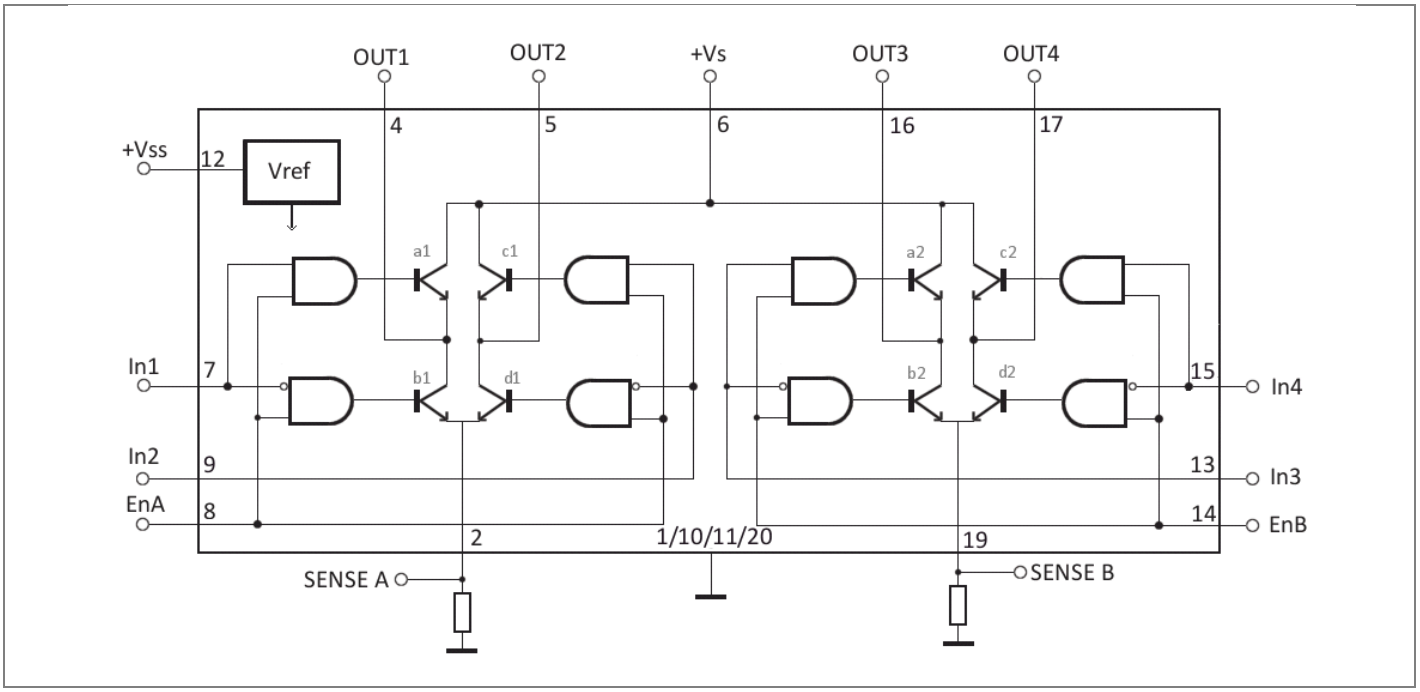


Figura 14 – Pin out de L298 para o package PowerSO20

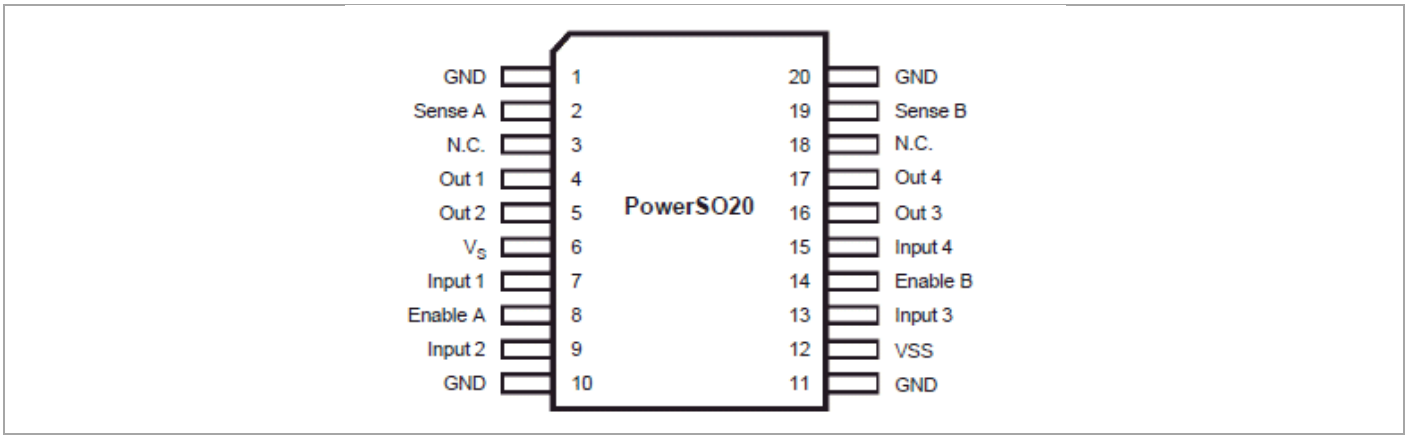


Figura 15 – L298, Package PowerSO20

3. Controlo do motor de Passo pela placa Arduino UNO

O controlo do motor de passo será constituído por uma placa Arduino UNO, figura 16, que tem a funcionalidade de controlo digital, e por uma placa Motor Shield UNO, figura 20, que tem a funcionalidade de controlo de potência pois contém o circuito integrado L298.

O circuito digital (Arduino UNO) permite comandar os transístores da ponte H (Motor Shield Arduino). É Conveniente que o circuito digital seja programável pois reduz consideravelmente a sua complexidade e torna o controlo dos solenoides muito mais flexível, tanto na sequência como na periocidade com que estes são alimentados. A placa Arduino UNO é programada através dum conector USB por intermédio de uma aplicação própria ao Arduino que utiliza uma sintaxe de programação muito simples e semelhante à linguagem C.

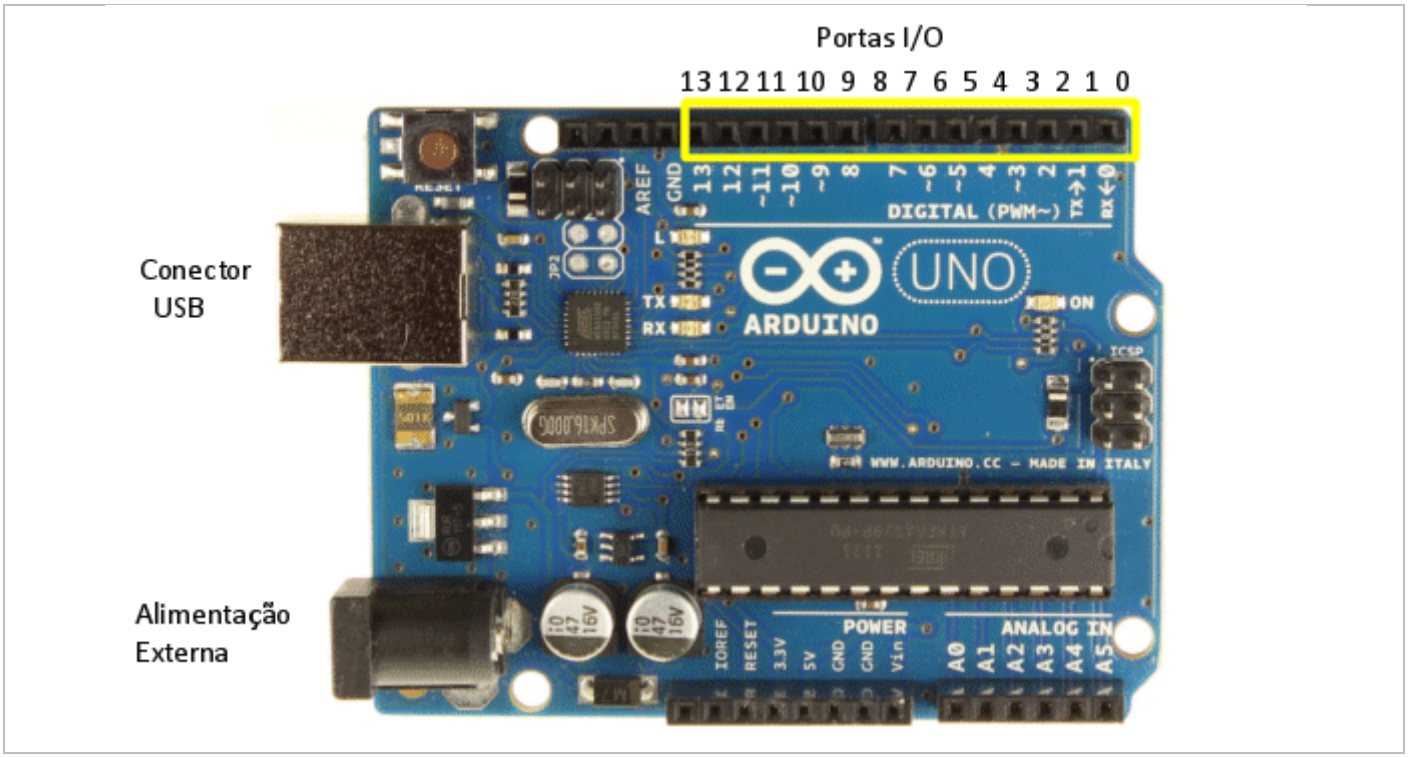


Figura 16 – Placa Arduino UNO

Como se pode ver na figura 16, a placa contém 13 portas I/O (Input/Output) que podem ser definidas como entradas ou saídas, conforme necessidade. No caso especifico deste projeto será necessário definir 6 portas como sendo saídas para comandar as entradas In1, In2, EnA, In3, In4, e EnB do circuito integrado L298.

Segue-se a exemplificação do código que permite definir as portas:

```
pinMode (1,INPUT)      ; Define a porta 1 como sendo uma entrada
pinMode (2,OUTPUT)     ; Define a porta 2 como sendo uma saída

digitalWrite (2, LOW)  ; Coloca o estado digital 0 na porta de saída 2 (por exemplo desligar transístor da ponte H)
digitalWrite (2, HIGH) ; Coloca o estado digital 1 na saída 2 (ligar transístor da ponte H)
```

Resumindo muito brevemente, o código de controlo terá de ser escrito na aplicação da Arduino como se pode ver na figura 17, e descarregado uma vez concluído para a placa Arduino ao selecionar na seta  pelo intermédio dum cabo USB.

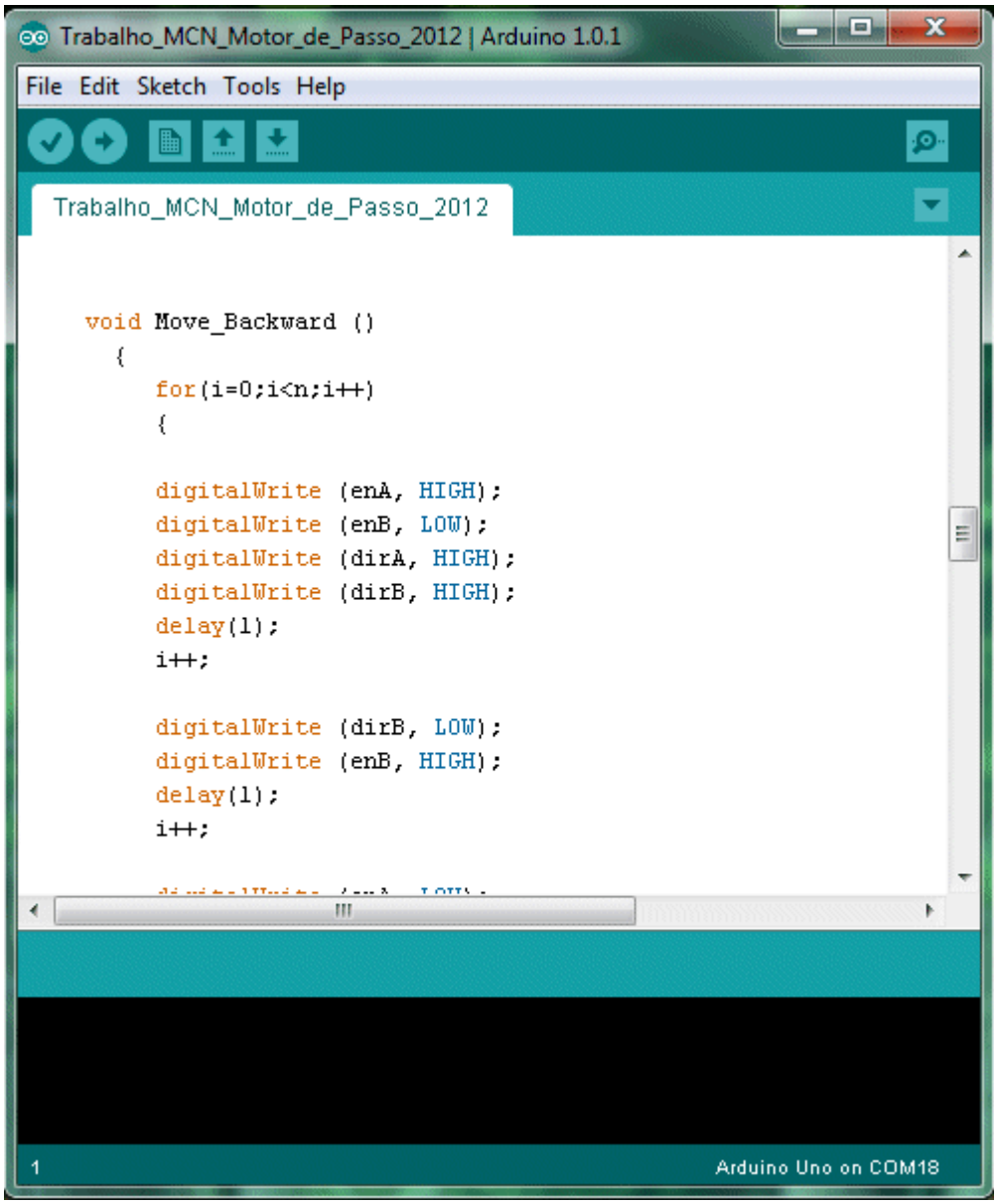


Figura 17 – Aplicação Arduino

A aplicação do Arduino permite enviar comandos à placa através do cabo USB por comunicação dita série, selecionando a opção “Serial Monitor” exemplificado na figura.

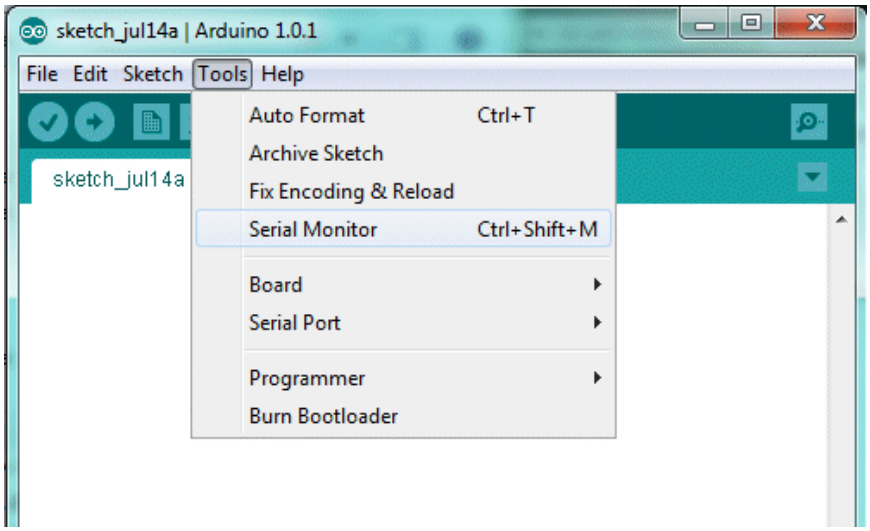


Figura 18 – Selecionando a aplicação “Serial Monitor”

Abrirá uma janela onde poderemos escrever os comandos que desejamos, no exemplo da figura 19 serão enviados os dados “400x” quando clicar no botão “Send”. No código desenvolvido o qual está descrito mais adiante, o comando “400x” corresponde à ordem de rodar 400 passos do motor respetivo ao eixo dos X.

Não esquecer que na comunicação série, tem de ser definido a velocidade de comunicação de dados, designada por “baudrate”, podemos ver no canto inferior direito da figura 19 que está selecionado um baurate de 9600 bits por segundo, que deve corresponder ao mesmo valor definido no código da Arduino com a seguinte linha de comando :

```
Serial.begin(9600) ;
```

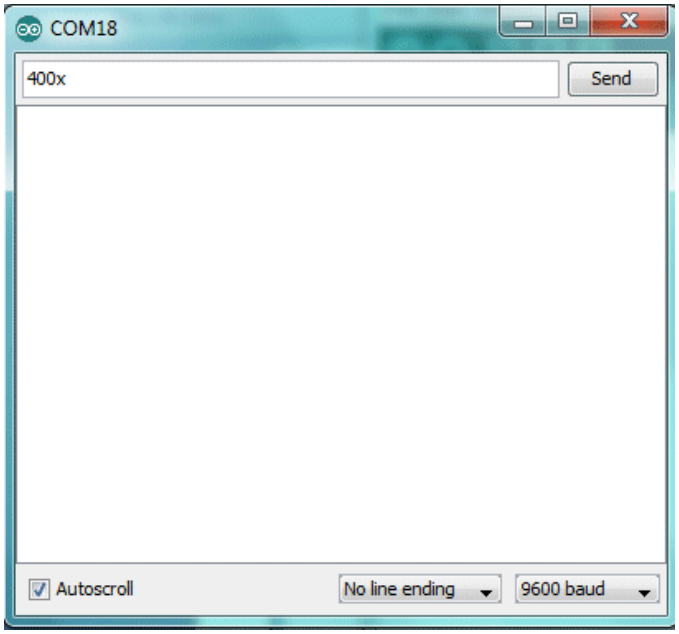


Figura 19 – Janela “Serial Monitor” Arduino

Juntamente com a placa Arduino UNO é portanto utilizada uma placa Arduino Motor Shield, ilustrada na figura 20. Esta é encaixada na placa Arduino UNO na parte superior através dos conectores. Verifica-se a utilização do circuito integrado L298 com package PowerSO20. Esta placa só permite comandar um motor de passo pois só tem um circuito integrado L298, ou seja, só tem uma ponte H.

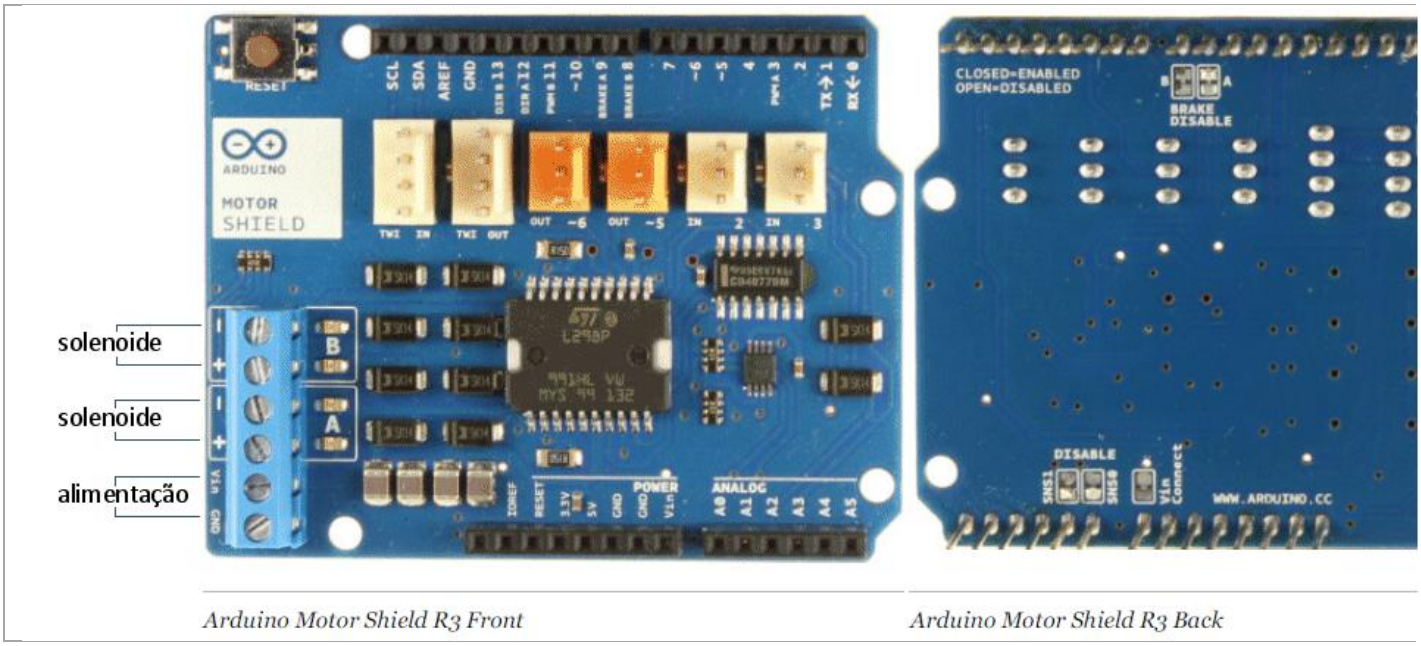
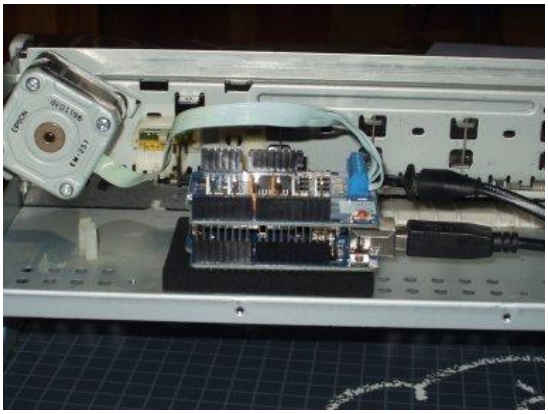
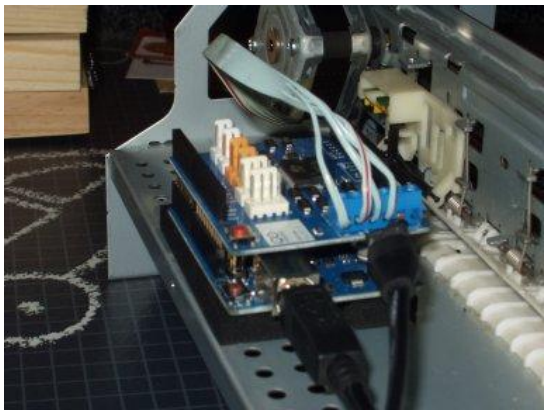


Figura 20 – Foto da placa *Arduino Motor Shield R3*

As seguintes imagens ilustram a montagem do trabalho realizado, onde podemos ver a placa Arduino Motor Shield montada em cima da placa Arduino UNO. O conjunto controla um motor de passo de forma quadrada nas imagens pertencente à estrutura de uma velha impressora. Um cabo USB liga à placa Arduino UNO para permitir mandar comandos ao motor. Um transformador comum de 12V liga à placa Arduino Motor Shield permitindo dar potência ao passo, e este liga os seus 4 fios à placa Motor Shield conforme mostrado na figura 20.

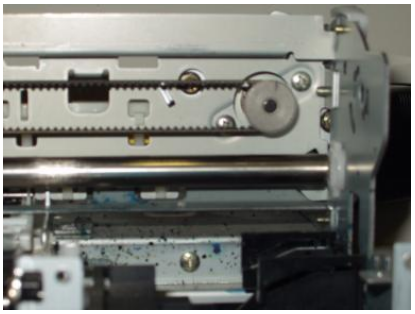


a) Motor de passo e placas Arduino



b) Visualização das conexões

c) Transformador



d) Ligação do motor de passo à correia da impressora.

Figura 21 – fotos do trabalho

O seguinte esquema da figura 22 explica de forma resumida e explicita a função da placa Arduino Motor Shield. Verifica-se que são utilizadas as portas 3 e 12 para comandar um dos solenóides do motor, e as portas 11 e13 para comandar o segundo solenoide. De notar que se PWMA = 0 ou PWMB = 0, o respetivo solenoide fica desconectado.

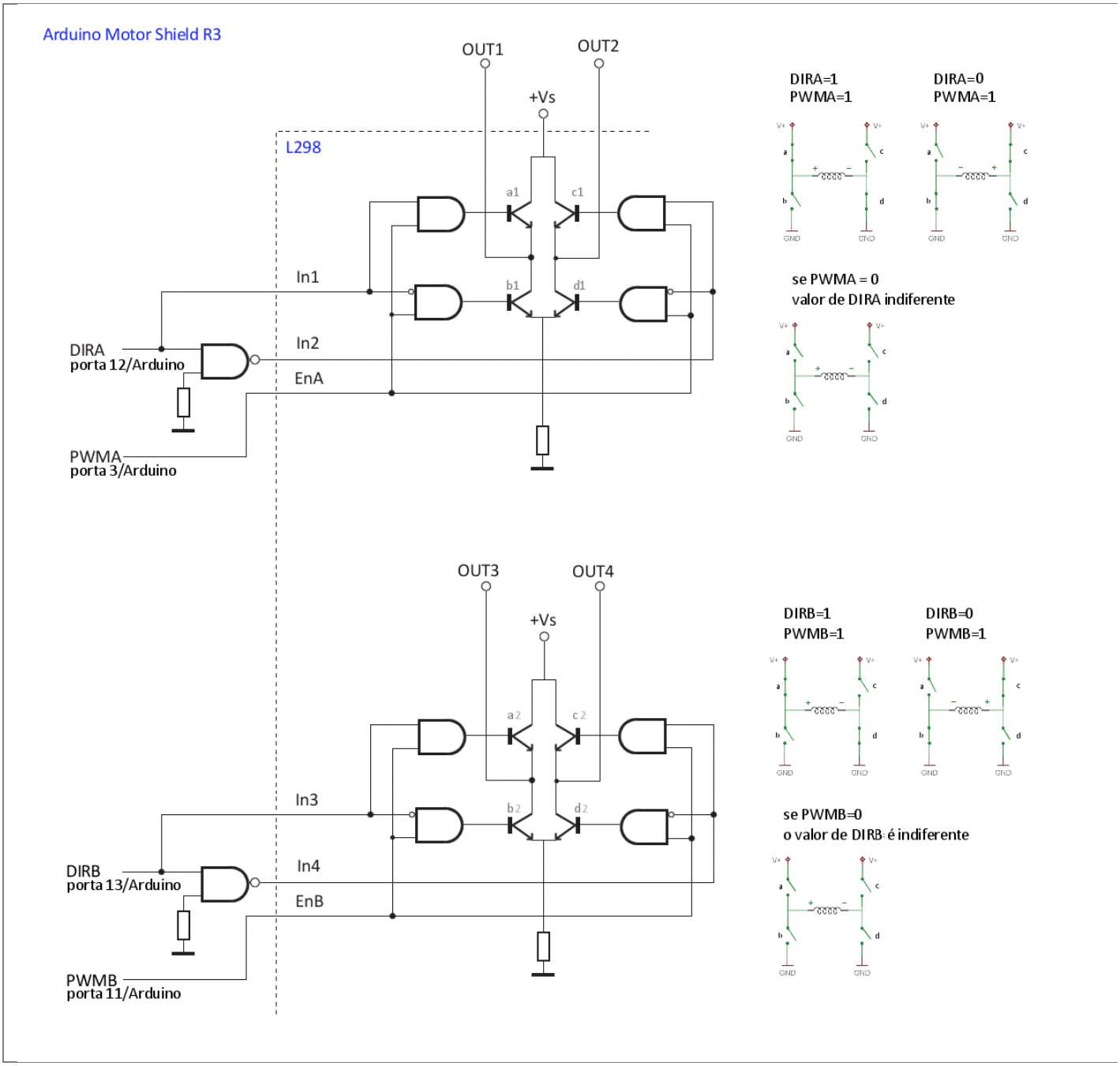


Figura 22 – Esquema simplificado da placa Arduino Motor Shield

Segue-se o código referente à definição das portas que comandam a placa Arduino Motor Shield :

```
int enA  = 3 ;   // entrada enA comandada pela porta 3
int enB  = 11 ; // entrada enB comandada pela porta 11
int dirA = 12;  // entrada dirA comandada pela porta 12
int dirB = 13;  // entrada dirB comandada pela porta 13

pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(dirA, OUTPUT);
pinMode(dirB, OUTPUT);
```

Sabendo como é controlado um motor de passo, falta definir a sequência dos sinais digitais DirA, PWMA, DirB e PWMB. O motor de passo utilizado na impressora anteriormente ilustrada é um motor de 200 passos, e tendo somente 4 fios tem obrigatoriamente de ser controlado em modo bipolar. Será utilizada uma atuação do tipo meio-passo, ou seja, é alimentada alternadamente uma e duas solenoides, conseguindo assim obter uma duplicação do número de passos para 400.

Caso o motor tivesse um rotor com somente dois polos, como ilustrado na figura 22 seriam necessários 8 comandos sequencias para realizar uma volta completa. O motor utilizado no trabalho sendo híbrido tem um rotor multidentado e tem portanto vários polos. Este é comandado da mesma forma, a sequência de comandos é exatamente a mesma, com a diferença de que a cada ciclo o motor não executa uma volta completa de 360º, mas sendo de 400 passos executa um deslocamento angular de (8/400) * 360 = 7.2º. Será necessário efetuar 50 ciclos de 8 passos para efetuar uma volta de 360º.

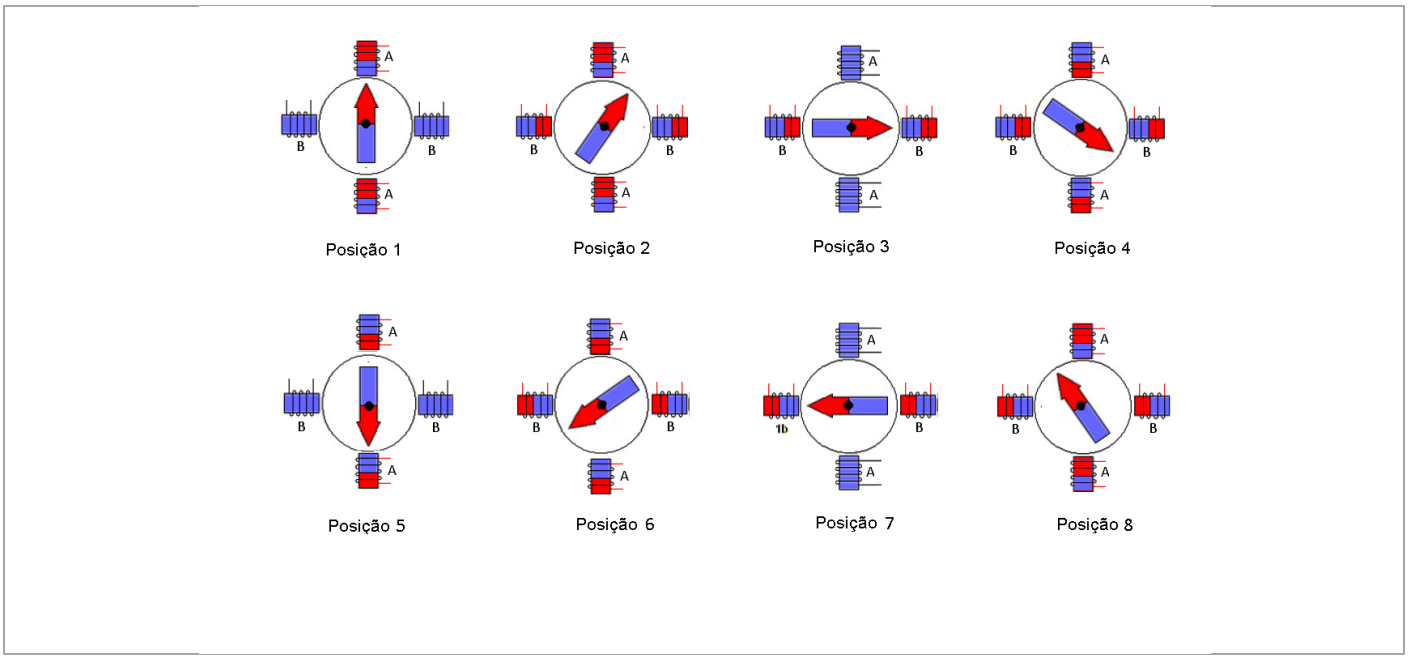


Figura 22 – Motor de passo com rotor de dois polos.

A seguinte tabela ilustra a sequência cíclica dos comandos enviados ao motor de passo. Chegando à posição 8, volta-se a enviar os comandos correspondentes à posição 1 , e assim sucessivamente. O valor "x" significa que é indiferente ter valor "0" ou "1", pois o respetivo PWM tem o valor "0", ou seja o respetivo solenoide está desligado. O valor de DIRA ou DIRB define a polaridade do respetivo solenoide, e o valor de PWMA ou PWMB define se respetivo solenoide é ligado ou não, pois alternadamente só é ligado um solenoide de cada vez.

	DIRA Porta 12	DIRB Porta13	PWMA Porta3	PWMB Porta11
Posição 1	1	x	1	0
Posição 2	1	1	1	1
Posição 3	x	1	0	1
Posição 4	0	1	1	1
Posição 5	0	x	1	0
Posição 6	0	0	1	1
Posição 7	x	0	0	1
Posição 8	1	0	1	1

Na figura 23 é ilustrada a sequência de movimento de um rotor multidentado. Como podemos verificar após 8 sequências voltamos a ter o rotor no mesmo estado, ou seja temos um polo sul a apontar para cima, de modo que a posição 9 corresponde em termos funcionais à posição 1.

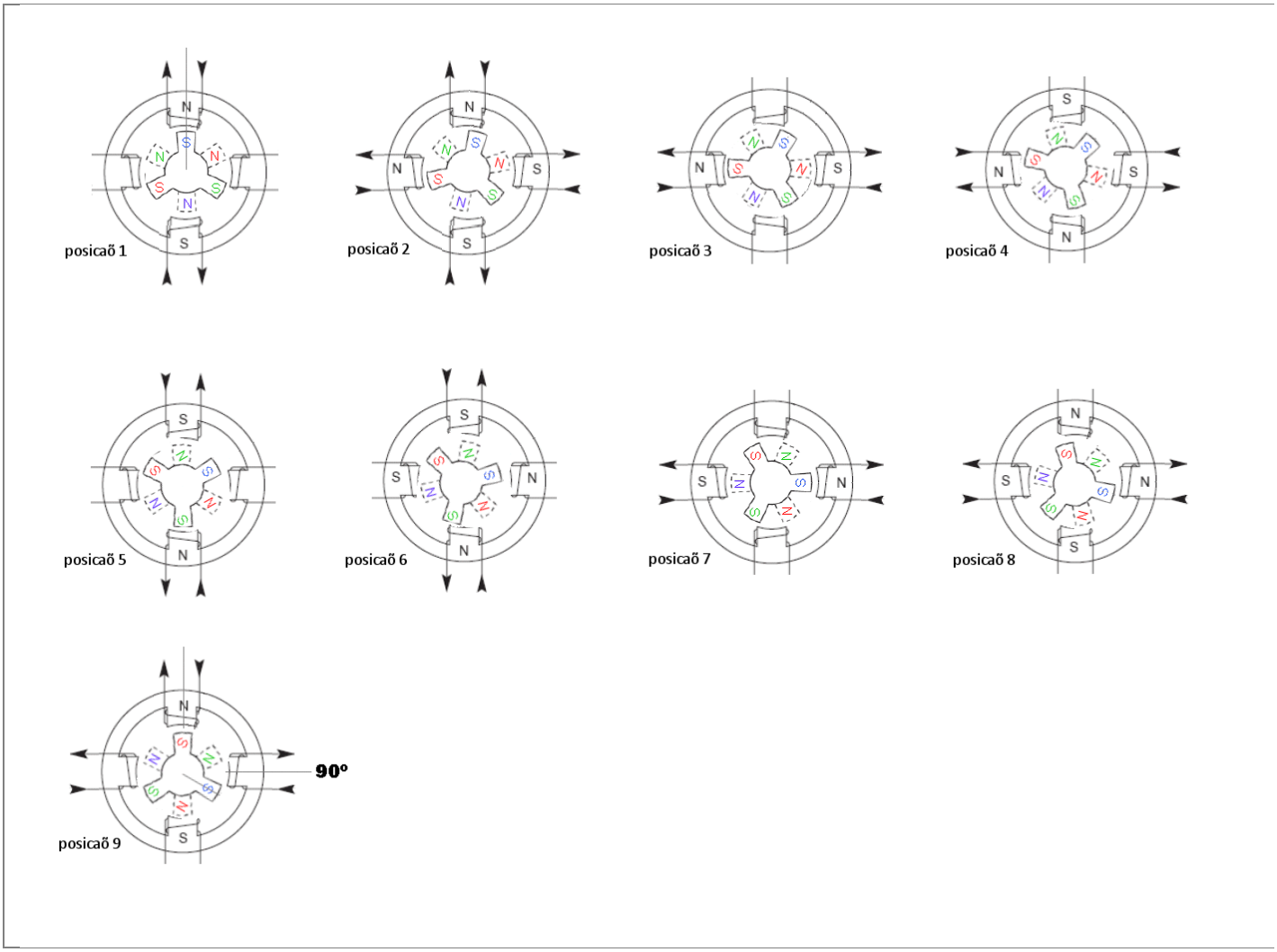


Figura 23 – Motor de passo com rotor multidentado de 6 polos.

A tabela de comando é idêntica à tabela correspondente a um motor com rotor de dois polos :

	DIRA Porta 12	DIRB Porta13	PWMA Porta3	PWMB Porta11
Posição 1	1	x	1	0
Posição 2	1	1	1	1
Posição 3	x	1	0	1
Posição 4	0	1	1	1
Posição 5	0	x	1	0
Posição 6	0	0	1	1
Posição 7	x	0	0	1
Posição 8	1	0	1	1

Passando a tabela para código Arduino, a cada posição não se define novamente o valor de cada variável, foi utilizado o conceito de "função Modal" utilizado em programação código G. Dando um exemplo, para passar da posição 1 para a posição 2, só muda o valor de PWMB que transita do valor 0 para valor 1 (o valor de DIRB é indiferente no caso da posição 1, mas de maneira a simplificar é lhe atribuído o valor 1 pois é o valor que tem também na posição 2)

forward	DIRA Porta 12	DIRB Porta13	PWMA Porta3	PWMB Porta11
Posição 1	1	x	1	0
Posição 2	1	1	1	1
Posição 3	x	1	0	1
Posição 4	0	1	1	1
Posição 5	0	x	1	0
Posição 6	0	0	1	1
Posição 7	x	0	0	1
Posição 8	1	0	1	1

É considerado que esta sequência faz mover o motor no sentido de rotação considerado positivo, e é portanto atribuído o nome da respetiva função Arduino como sendo Move_Forward(). Esta função executa ciclos de 8 passos até atingir o número de passos (variável "n") que lhe foram comandados.

```
void Move_Forward ()
  {
    for(i=0;i<n;i++)
    {

    posicao = 1;
    digitalWrite (dirA, HIGH);
    digitalWrite (dirB, HIGH);
    digitalWrite (enA, HIGH);
    digitalWrite (enB, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 2;
    digitalWrite (enB, HIGH);
    delay(1);
    if(i==n)break;
    i++;
```

```
    posicao = 3;
    digitalWrite (enA, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 4;
    digitalWrite (dirA, LOW);
    digitalWrite (enA, HIGH);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 5;
    digitalWrite (enB, LOW);
    delay(1);
    if(i==n)break;
    i++;
```

```
    posicao = 6;
    digitalWrite (dirB, LOW);
    digitalWrite (enB, HIGH);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 7;
    digitalWrite (enA, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 8;
    digitalWrite (dirA, HIGH);
    digitalWrite (enA, HIGH);
    delay(1);
    if(i==n)break;
    }
}
```

Para movimentar o motor de passo em sentido contrário, basta enviar os mesmos comandos mas agora com sequência inversa. Como podemos ver na tabela depois da posição 1, temos agora a posição 8.

Backward	DIRA Porta 12	DIRB Porta13	PWMA Porta3	PWMB Porta11
Posição 1	1	x	1	0
Posição 8	1	0	1	1
Posição 7	x	0	0	1
Posição 6	0	0	1	1
Posição 5	0	x	1	0
Posição 4	0	1	1	1
Posição 3	x	1	0	1
Posição 2	1	1	1	1

À função que movimenta o motor em sentido considerado negativo é atribuído o nome Move_Backward():

```
void Move_Backward ()
  {
    for(i=1;i<=n;i++)
    {

    posicao = 1;
    digitalWrite (dirA, HIGH);
    digitalWrite (dirB, HIGH);
    digitalWrite (enA, HIGH);
    digitalWrite (enB, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 8;
    digitalWrite (dirB, LOW);
    digitalWrite (enB, HIGH);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 7;
    digitalWrite (enA, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 6;
    digitalWrite (dirA, LOW);
    digitalWrite (enA, HIGH);
    delay(1);
    if(i==n)break;
    i++;

  posicao = 5;
    digitalWrite (enB, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 4;
    digitalWrite (enB, HIGH);
    digitalWrite (dirB, HIGH);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 3;

    digitalWrite (enA, LOW);
    delay(1);
    if(i==n)break;
    i++;

    posicao = 2;
    digitalWrite (dirA, HIGH);
    digitalWrite (enA, HIGH);
    delay(1);
    if(i==n)break;
    }
}
```

O código Arduino para funcionar tem de conter a função void Setup () e a função void loop(). Na função void Setup () como o indica o nome é onde é por exemplo definido a velocidade de comunicação série , as portas utilizadas, etc… e é somente executada ao ligar a placa Arduino. Depois de executar a função void Setup (), é executada continuamente a função void loop () até desligar a placa Arduino.

Segue-se o conteúdo das funções void Setup () e void loop() do código Arduino a que lhe foi atribuído o nome de Trabalho_MCN_Motor_de_Passo_2012.ino

Para além dessas duas funções, o código contém ainda as funções Move_forward () e Move_Backward() anteriormente apresentadas.

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

     pinMode(enA, OUTPUT);
     pinMode(enB, OUTPUT);
     pinMode(dirA, OUTPUT);
     pinMode(dirB, OUTPUT);
  }


void loop()
{

 if (Serial.available() > 0)
         {
           n = Serial.parseInt() ;                  // lê inteiro recebido pela comunicação série
           if (Serial.read() != 'x')   {  n=0 ; }   // só aceita valor se for seguido pelo caractere “x”
         }


    if (n>0)              // se valor de n é positivo, movimento no sentido positivo, Move_Forward ()
     {
     Move_Forward ();
     digitalWrite (enA, LOW);
     digitalWrite (enB, LOW);
     }


     if (n<0)             // se valor de n é positivo, converte-o para positivo, mas executa Move_Backward ()
     {
     n= -n ;
     Move_Backward ();
     digitalWrite (enA, LOW);
     digitalWrite (enB, LOW);
     }

    n=0 ;
}
```

O motor é comandado enviando comandos do tipo “400x”, exemplo ilustrado na figura 24, que corresponde a fazer rodar 400 passos no sentido considerado positivos, ou comandos do tipo “-400x” que corresponde a fazer rodar 400 passos no sentido considerado negativo.

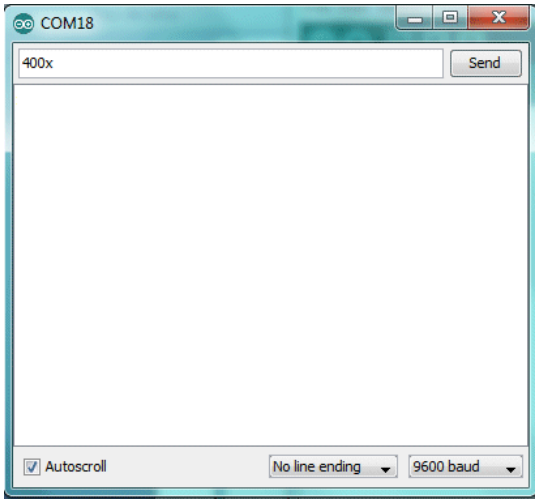


Figura 24 – Janela “Serial Monitor” Arduino

Caso o motor não esteja a rodar no sentido esperado, na figura 25 são mostradas 3 alternativas de como para alterar o sentido de rotação do motor alterando as ligações do motor.

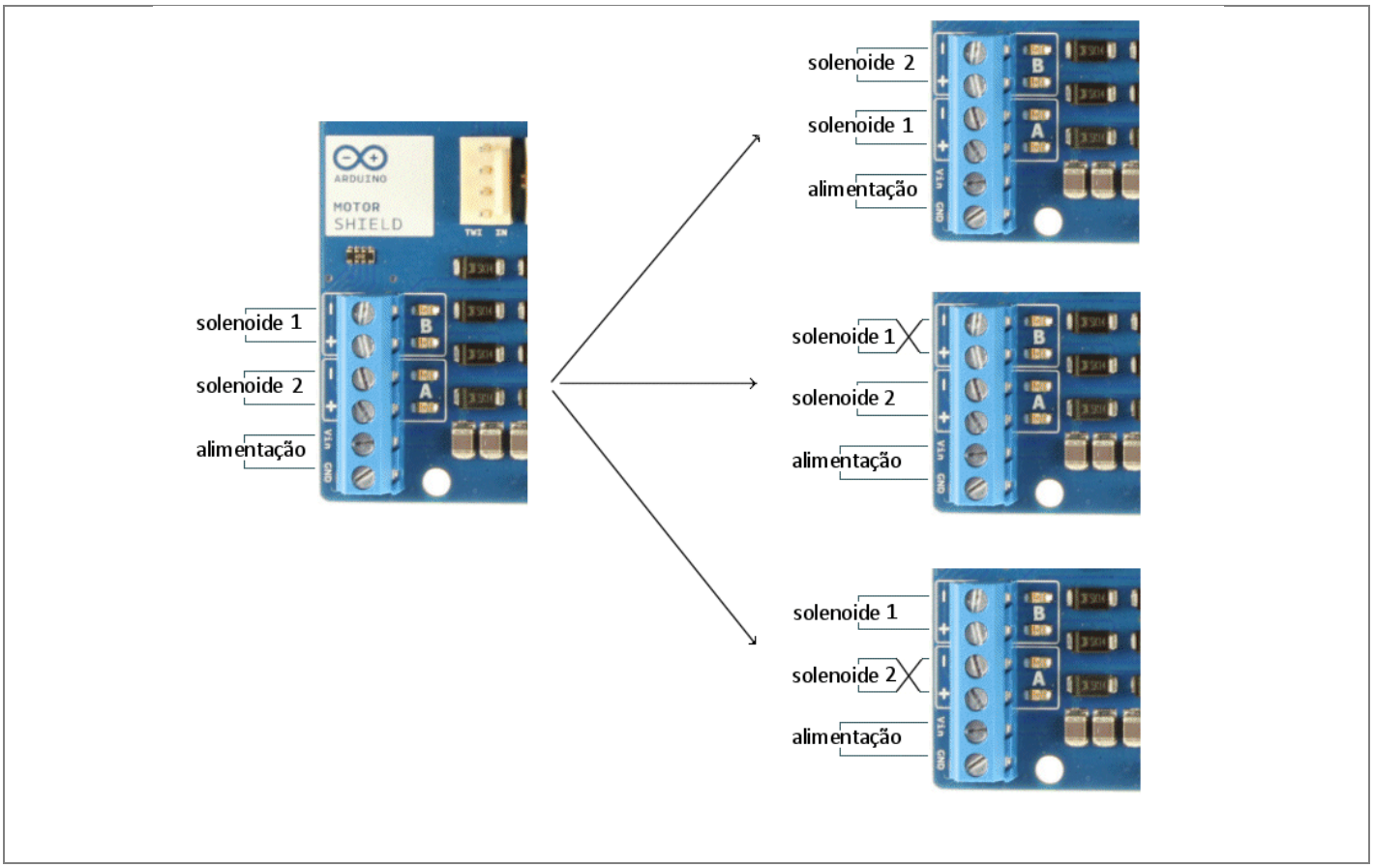


Figura 24 – Inverter sentido de rotação.

4. Conclusões

Não foram aqui discutidos aspetos técnicos relativos ao motor de passo, mas num projeto teriam de ser consideradas características técnicas importantes tais como por exemplo o binário máximo produzido em função da velocidade, pois quanto maior for a velocidade de rotação menor será a duração dos impulsos que alimentam os solenoides e por consequente menor será o binário. Este é um dos aspetos mais importantes a ser considerados em qualquer projeto, pois deve ser sempre garantido que não haja perda de passos, principalmente se trabalhar em modo malha aberta.

Relativamente ao código Arduino, podem ser efetuadas melhorias, tal como estar sempre a para da posição (1 a 8) em que se encontra o rotor. No código existente são sempre efetuados os comandos (Move_forward e Move_Backward) considerando que o rotor está posicionado na posição 1, o que está obviamente errado. Se o rotor se encontrar por exemplo na posição 4, o primeiro comando terá de coloca-lo na posição 5, depois na posição 6, e assim sucessivamente.

Para controlar dois eixos, ou seja dois motores, é aconselhável criar uma placa com dois circuitos integrados L298.
No código Arduino teria –se de duplicar as funções Move_forward () e Move_Backward(), e o segundo motor seria comandado por um comando do tipo “400y” considerando que este controlasse o eixo dos Y.

5. Referencias

1 - Http://www.feiradeciencias.com.br/sala22/motor_teoria1.asp

2 - Http://www.scribd.com/doc/50369954/Trab-Tec-Automacao-pdf

3 - Http://arduino.cc/en/Main/ArduinoBoardUno/

4 - Http://arduino.cc/it/Main/ArduinoMotorShieldR3