UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Faculdade de Engenharia e Ciências de Guaratinguetá
Departamento de Engenharia Elétrica

Sistemas Microcomputadorizados

Laboratório 7

Interação com Motores de Passo

Sumário

1 Objetivos

- Estudar e implementar o controle de motores de passo.

2 Motores de Passo

Motores de passo são dispositivos eletromecânicos capazes de converter pulsos elétricos em movimentos mecânicos discretos. Diferentemente de motores elétricos convencionais, os quais giram de forma contínua, motores de passo movimentam-se em incrementos fixos, chamados de "passos". Estes movimentos discretos ocorrem quando pulsos elétricos de comando são aplicados ao motor em uma sequência pré-estabelecida, gerando um deslocamento angular preciso.

Motores de passo estão disponíveis em uma ampla faixa de resolução angular. Motores de alta resolução conseguem lidar, geralmente, com 1,8 ou até 0,72 graus por passo [1]. São amplamente utilizados em aplicações que requerem controle preciso de posição e velocidade, sem a necessidade de sensores de realimentação. Podem ser posicionados precisa e repetidamente, já que bons motores de passo têm precisão na faixa de 3% a 5% de um passo e estes erros não são cumulativos de um passo para o seguinte.

Exemplos típicos de uso incluem: impressoras 3D, equipamentos de automação industrial, máquinas CNC, sistemas de posicionamento em robótica, entre outros.

2.1 Princípio de Funcionamento

O funcionamento do motor de passo baseia-se na interação eletromagnética entre estator e rotor. O estator possui enrolamentos que, quando energizados, criam campos magnéticos capazes de atrair pólos magnéticos do rotor. Ao energizar, sequencialmente, os enrolamentos do estator, cria-se um movimento rotacional controlado. Cada sequência de energização corresponde a um passo do motor, sendo que o número de passos por volta depende de sua construção.

2.2 Tipos de motores de passo

Motores de passo apresentam três tipos de construção: motores de magneto permanente, relutância variável e híbridos.

2.2.1 Motor de Ímã Permanente

Neste tipo de motor, o rotor é composto por um ímã permanente que interage diretamente com os campos magnéticos gerados pelo estator. Estes motores possuem uma mecânica mais simples e de baixo custo. Os ímãs permanentes permitem que este tipo de motor produza um torque mais alto. Entretanto, o desempenho em alta velocidade é limitado devido à grande perda durante a rotação. É geralmente utilizado para posicionamento simples, pois sua resolução tende a ser maior do que outros tipos [2].

Quando uma corrente é aplicada aos enrolamentos, os pólos do estator são magnetizados e se alinham com os pólos opostos do rotor de ímã permanente [2]. A Figura 1, mostra um rotor e um estator nos quais uma corrente é aplicada de forma que o pólo A_1 seja magnetizado para o pólo sul e o pólo A_2 para o pólo norte, atraindo assim os pólos opostos dos ímãs permanentes do rotor. Se a corrente fosse comutada da fase A para a fase B para magnetizar B_1 para o pólo sul e B_2 para o pólo norte, o rotor giraria 90 graus no sentido horário.

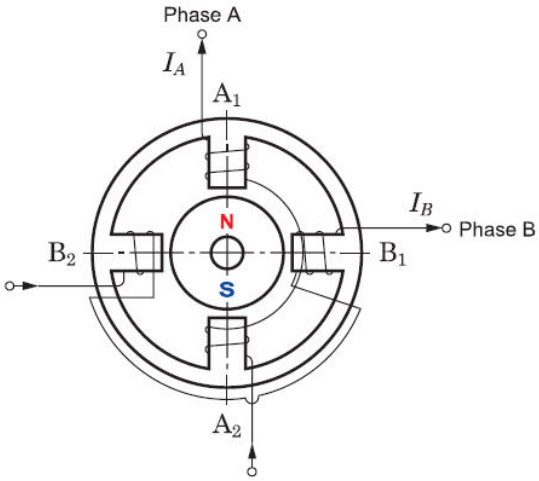


Figura 1: Esquemático exibindo o corte transversal de um motor de ímã permanente [2].

A resolução do motor de passo com ímã permanente pode ser aumentada com o aumento do número de pólos no rotor, ou o aumento do número de fases.

2.2.2 Motor de Relutância Variável

Possui o rotor feito de algum material ferromagnético laminado, sem ímãs permanentes. Possui dentes tanto no rotor quanto no estator, onde as forças magnéticas são concentradas. O rotor tende a alinhar-se com o campo magnético gerado pelas bobinas do estator. Este tipo de motor apresenta um projeto simples e econômico. A Figura 2 exibe o corte transversal de um motor de relutância variável com três fases, 8 pólos e 12 *slots*. Os dentes no rotor são atraídos pelos pólos energizados no estator.

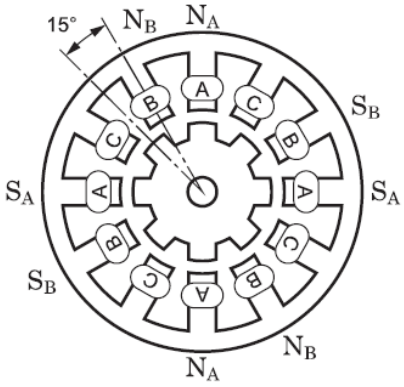


Figura 2: Esquemático exibindo o corte transversal de um motor de relutância variável [2].

2.2.3 Motor híbrido

Combina características dos motores de relutância variável e de ímãs permanentes. O rotor contém ímãs permanentes e dentes que aumentam a resolução e o torque, assim como o estator também possui dentes. São os motores de passo mais utilizados devido à sua precisão e eficiência. A Figura 3 apresenta um esquemático do corte transversal de um motor deste tipo.

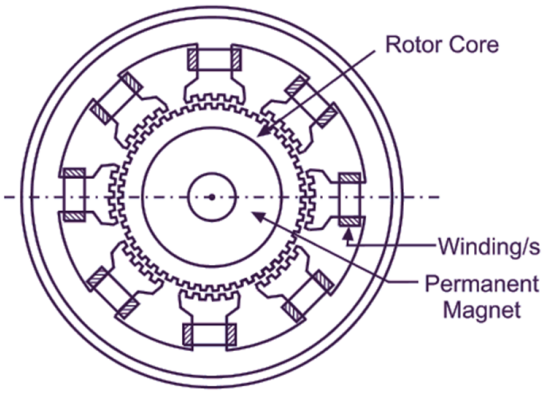
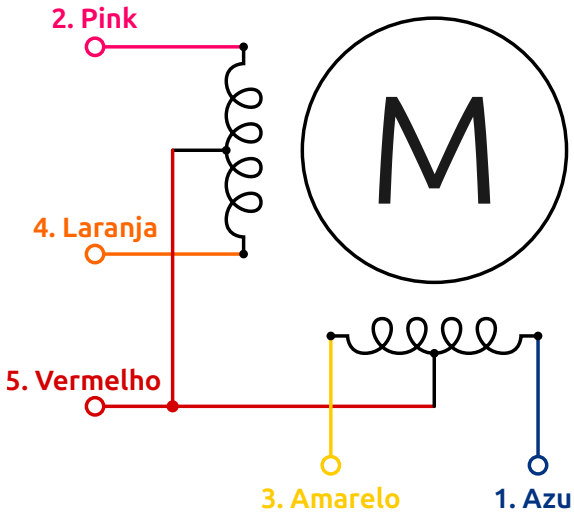


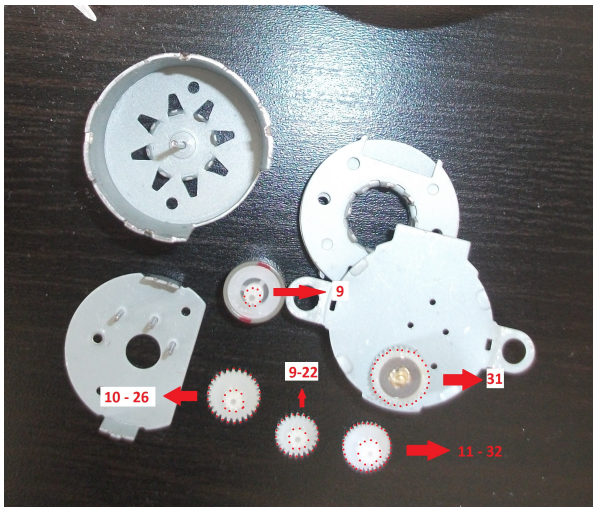
Figura 3: Esquemático exibindo o corte transversal de um motor híbrido [3].

2.3 Motor de passo 28BYJ-48

O motor de passo 28BYJ-48 é um modelo amplamente utilizado em aplicações didáticas e projetos de automação de pequeno porte. É um motor de passo unipolar, com 4 bobinas organizadas em um conjunto de 5 fios (com fio comum). A Figura 4a apresenta o diagrama interno dos enrolamentos do motor `28BYJ-48`, enquanto a Figura 4b foi encontrada em um fórum do próprio site do Arduino [4] e exibe as engrenagens que fazem a redução do passo do motor.



(a) Diagrama dos enrolamentos. Adaptado de [5].



(b) Engrenagens do motor. Obtida em [4].

Figura 4: Características do motor de passo `28BYJ-48`.

Abaixo, são listadas algumas das principais características deste motor[1]:

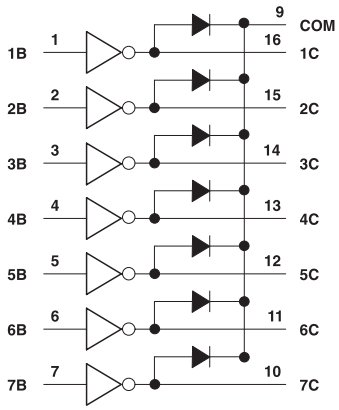
- Tensão nominal: 5 V
- Número de fases: 4
- Relação de variação de velocidade 1/64
- Ângulo de passo: $5,625^o$ (sem considerar caixa de redução)
- Frequência: 100 Hz

[1]Para outras características, consultar o *datasheet* do motor, disponível no Google Classroom da disciplina.

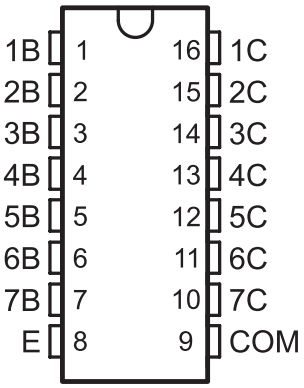
2.4 Acionamento com *driver* ULN2003APG

O *driver* ULN2003APG é um arranjo de transistores do tipo *darlington* que permite acionar cargas indutivas, como motores de passo, relés e solenóides. Cada canal deste *driver* suporta correntes de até 500 mA e tensões de até 50 V. Normalmente, é utilizado para acionar periféricos conectados às portas GPIO (*General Purpose Input Output*) de sistemas digitais, como microcontroladores. Esses periféricos podem exigir altas correntes ou tensões (ou ambos), sendo estas condições de operação que sistemas digitais não conseguem suportar.

A Figura 5a exibe um diagrama de blocos simplificado do CI (circuito integrado) ULN2003APG, enquanto a Figura 5b apresenta a vista superior do encapsulamento do mesmo.



(a) Diagrama de bloco simplificado.



(b) Vista superior com pinagem.

Figura 5: Diagramas do CI ULN2003APG. Obtido em [6].

A Figura 6 apresenta o esquemático do módulo que contém o CI ULN2003APG. Os quatro leds vermelhos (D_1, D_2, D_3 e D_4) são utilizados para indicar o acionamento de cada uma das bobinas através de seus respectivos *drivers*. Mesmo que o diagrama mostre sete entradas neste módulo, apenas quatro podem ser usadas e somente estas possuem pinos disponíveis no conector [7].

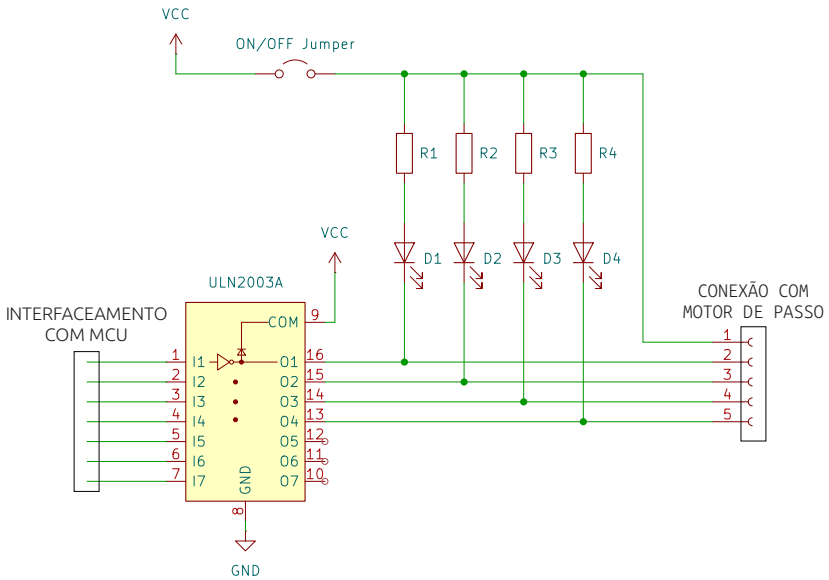


Figura 6: Esquemático do módulo que contém o driver do motor de passo utilizado neste laboratório. Adaptado de [7].

A Figura 7 apresenta uma foto do módulo de acionamento do motor de passo. O pino mais à esquerda ("-") é o pino em que deve-se conectar o terminal do terra (`GND`) da fonte externa. Este terra deve ser conectado, também, ao pino de terra do microcontrolador em uso. O pino ("+"), por sua vez, deve ser conectado ao terminal positivo da fonte de externa de 5 V[2]. **Não utilize o microcontrolador para alimentar o *driver*, pois, devido às correntes envolvidas, serão causados danos ao dispositivo. Utilize apenas a fonte externa para esta finalidade**. O jumper "`ON/OFF`" é usado para ligar ou desligar o motor.

Figura 7: Foto do módulo de acionamento do motor de passo.

2.5 Modos de acionamento

O motor é acionado ao serem enviados sinais para o *driver* ULN2003APG, o qual ativa as bobinas do motor. Para um motor de passo unipolar, existem diferentes modos de operação: passo completo com alto torque (*full step*), passo completo com baixo torque (*wave step*), meio passo (*half step*) e micro passo (*micro stepping*). Para aplicar o controle de micro passo, é necessário controlar a corrente e tensão de acionamento das bobinas, de forma que o passo do motor torne-se menor que o passo de deslocamento padrão. Como o *driver* utilizado não possui esta funcionalidade, este modo de operação não poderá ser utilizado. Para acionar o motor utilizando os demais modos de operação, deve-se atentar à sequência de acionamento de cada bobina.

2.5.1 Passo completo com alto torque (*full step*)

Neste modo de operação, duas bobinas são acionadas ao mesmo tempo. A Tabela 1 apresenta a sequência de acionamento do ponto de vista do controlador. `IN1`, `IN2`, `IN3` e `IN4` representam os pinos de entrada do *driver*.

Tabela 1: Sequência de acionamento para o modo de operação *full step*.

Step	`IN1`	`IN2`	`IN3`	`IN4`
0	1	0	0	1
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0

[2]Deve-se utilizar 5 V para o acionamento do motor 28BYJ-48, o qual opera com tensão nominal de 5 V. Este *driver* pode ser utilizado com outros níveis de tensão para outros motores. Consultar os *datasheets* dos respectivos dispositivos para correta configuração de operação.

2.5.2 Passo completo com baixo torque (*wave step*)

Nesta situação, apenas uma bobina é acionada por vez.

Tabela 2: Sequência de acionamento para o modo de operação *wave step*.

Step	IN1	IN2	IN3	IN4
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

2.5.3 Meio passo (*half step*)

Por fim, no modo de operação de meio passo, o número de bobinas acionadas depende do passo em questão. Utiliza uma sequência de oito passos.

Tabela 3: Sequência de acionamento para o modo de operação *half step*.

Step	IN1	IN2	IN3	IN4
0	1	0	0	1
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1

3 SSH para Conexão Remota

SSH (*Secure Shell*) é um protocolo de rede criptografado utilizado para para prover acesso de forma segura e remota a sistemas, como servidores e dispositivos, através de uma rede não segura, como a internet. Como neste protocolo todas as comunicações entre o cliente e o servidor são protegidas por criptografia, a transmissão de dados sensíveis, como senhas e comandos, não é interceptada por terceiros.

O SSH é amplamente utilizado em administração de servidores remotos, acesso a sistemas de rede e execução de comandos de forma remota, substituindo protocolos antigos, como o Telnet, que não oferecem criptografia e são vulneráveis a ataques. Ele pode ser utilizado, também, para transferir arquivos de maneira segura, através do SFTP (*Secure File Transfer Protocol*) ou SCP (*Secure Copy Protocol*), além de possibilitar o uso de tunelamento seguro para outros protocolos de rede [8].

3.1 Como o protocolo SSH funciona

O protocolo foi implementado baseado no modelo cliente-servidor, o que significa que a conexão é estabelecida pelo cliente ao conectar-se ao servidor SSH. O cliente SSH conduz o processo de

configuração da conexão e usa criptografia de chave pública para verificar a identidade do servidor SSH. Após a fase de configuração, o protocolo SSH utiliza algoritmos de criptografia simétrica e de *hash* para garantir a privacidade e a integridade dos dados trocados entre o cliente e o servidor [9].

A Figura 8, abaixo, apresenta um fluxo simplificado de configuração de uma conexão SSH.

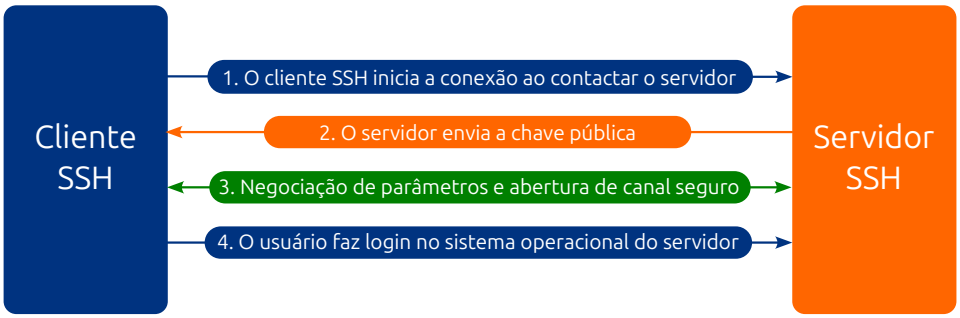


Figura 8: Fluxo simplificado de configuração de uma conexão SSH. Adaptado de [8] e [9].

3.2 Chaves SSH

Uma das principais razões para o uso de SSH é a segurança. Para reforçá-la, recomenda-se fortemente a utilização de autenticação por chaves ao invés de senha, o que torna o acesso muito mais difícil para potenciais invasões. Chaves SSH são credenciais de segurança. Do ponto de vista de funcionalidade, assemelham-se ao uso de senhas. Entretanto, técnicamente, as chaves SSH são chaves criptográficas que utilizam um sistema de criptografia de chave pública.

A autenticação por chave pública/privada é um processo onde o cliente gera um par de chaves: uma chave pública (que é colocada no servidor) e uma chave privada (que permanece no cliente). Quando o cliente tenta se conectar ao servidor, o servidor envia um desafio criptográfico, que só pode ser resolvido com a chave privada correspondente à chave pública que fora préviamente instalada no servidor. Este método elimina a necessidade de enviar senhas pela rede e torna o processo de autenticação mais seguro, já que a chave privada nunca é compartilhada e a comunicação é criptografada de ponta a ponta.

Neste contexto, uma chave pública seria análoga a um cadeado, onde apenas quem possuir a chave correta (chave privada) conseguirá abrí-lo [10].

3.3 Como utilizar o protocolo SSH

Em sistemas Linux, o cliente SSH, geralmente, já vem instalado por padrão. Já em sistemas Windows, a partir da versão 10, o cliente SSH também já está incluso no sistema. Para trabalhar com conexões SSH, basta abrir o respectivo terminal de comandos de cada sistema e digitar os comandos SSH. Para sistemas Windows, existem, também, diversos programas de terceiros para esta mesma tarefa. Um dos mais conhecidos é o programa chamado `PuTTY`, disponível para download em [11]. Para os exemplos a seguir, este tutorial utilizará os terminais de comando, uma vez que os comandos são os mesmos para ambos sistemas.

3.3.1 Gerando as chaves pública e privada

Para gerar um par de chaves SSH (pública e privada), use o seguinte comando: `ssh-keygen`. Este comando cria duas chaves[3]: uma pública (`/.ssh/id_rsa.pub`) e uma privada (`/.ssh/id_rsa`). A

[3]Durante a execução do comando `ssh-keygen`, será solicitado ao usuário para especificar o nome dos arquivos de chave pública e privada. Os nomes aqui utilizados são apenas a sugestão padrão que este comando produz durante o

chave pública pode ser copiada para o servidor remoto utilizando o comando `ssh-copy-id`. Exemplo:

```
$ ssh-copy-id usuario@ip-ou-nome-servidor
```

3.3.2 Conexão SSH

Para estabelecer uma conexão com um servidor remoto, o comando SSH tem a seguinte forma:

```
$ ssh usuario@ip-ou-nome-servidor
```

Onde `usuario` é o nome de usuário do servidor e `ip-ou-nome-servidor` é o endereço IP ou nome de domínio do servidor ao qual deseja-se conectar. Segue um exemplo abaixo:

```
$ ssh admin@192.168.1.10
```

Caso seja a primeira vez que a conexão a um determinado servidor esteja sendo estabelecida, será exibida ao usuário uma mensagem de aviso sobre a autenticidade do servidor. Ao ser autorizada a conexão ao servidor pela primeira vez, a chave pública deste servidor será armazenada localmente, permitindo conexões futuras sem o mesmo aviso.

Se for necessário especificar uma chave privada para autenticação (caso esteja sendo utilizada uma chave ao invés de uma senha), é preciso especificar o caminho, no sistema local, para o arquivo que contém a chave:

```
$ ssh -i /caminho/para/a/chave-privada.pem usuario@ip-ou-nome-servidor
```

Também é possível usar o comando `scp` para copiar arquivos de maneira segura entre máquinas. Por exemplo:

```
$ scp arquivo.txt usuario@ip-ou-nome-servidor:/caminho/do/destino
```

3.3.3 Automatizando envio de arquivos

Os sistemas operacionais fornecem uma forma de automatizar tarefas através de arquivos de *script*. Portanto, podemos utilizar esta funcionalidade para automatizar a transferência de arquivos entre duas máquinas. O Bloco de Código 1, apresenta o conteúdo de um arquivo `.bat`[4] onde é realizada a tarefa de transferência de alguns arquivos através do comando `scp`. Lembre-se de ajustar as variáveis de acordo com o seu caso de uso.

Bloco de código 1: Arquivo `.bat` para transferência de arquivos entre duas máquinas.

```
1  @echo off
2
3  REM Configuracoes de acesso ao servidor
4  set REMOTE_USER="pi"
5  set REMOTE_HOST="200.145.18.187"
6  set REMOTE_PORT="2222"
7  set REMOTE_PATH="/home/pi/Documents/lab7"
8  set LOCAL_FILES="C:\Users\tjmic\Documents\main.c" "C:\Users\tjmic\Documents\
       stepper.c"
9
10 REM Executando o comando scp
11 echo Iniciando copia dos arquivos
12 scp -P %REMOTE_PORT% %LOCAL_FILES% %REMOTE_USER%@%REMOTE_HOST%:%REMOTE_PATH%
```

processo de criação das chaves.

[4]Os arquivos de *script* em Windows possuem o nome de *batch file* e possuem extensão `.bat`.

```
13
14 if %errorlevel% ==0 (
15   echo Copia concluida
16 ) else (
17   echo Erro ao copiar arquivos.
18 )
19
20 pause
```

4 Exemplo de Programação

Neste exemplo, será apresentado o código-fonte de um programa em C, desenvolvido para Raspberry Pi, com o uso da biblioteca `wiringPi`, para realizar o acionamento do motor de passo utilizando o método de passo completo.

Bloco de código 2: Arquivo de cabeçalho `stepper.h`

```
1  #ifndef _STEPPER_H_
2  #define _STEPPER_H_
3
4  #include <stdint.h>
5  #include <stddef.h>
6
7  extern uint8_t full_step_seq[4][4];
8
9  typedef void (*process_item_callback)(uint8_t item);
10
11 void iterate_array(uint8_t *array, size_t count, process_item_callback callback);
12 void configure_stepper_io(uint8_t port_number);
13 void turn_off_coil(uint8_t port_number);
14 void step_motor(size_t step, uint8_t *ports, uint8_t port_count);
15
16 #endif /* _STEPPER_H_ */
```

Bloco de código 3: Arquivo de código-fonte `stepper.c`

```
1  #include <wiringPi.h>
2  #include "stepper.h"
3
4  uint8_t full_step_seq[4][4] = {
5    {1, 0, 0, 1},
6    {0, 0, 1, 1},
7    {0, 1, 1, 0},
8    {1, 1, 0, 0},
9  };
10
11 void iterate_array(uint8_t *array, size_t count,
12   process_item_callback callback) {
13   for (size_t i = 0; i < count; i++) {
14     callback(array[i]);
15   }
16 }
17
18 void configure_stepper_io(uint8_t port_number) {
19   pinMode(port_number, OUTPUT);
20 }
```

```
21
22 void turn_off_coil(uint8_t port_number) {
23   pinMode(port_number, LOW);
24 }
25
26 void step_motor(size_t step, uint8_t *ports, uint8_t port_count) {
27   for (size_t i = 0; i < port_count; i++) {
28     digitalWrite(ports[i], full_step_seq[step][i]);
29   }
30 }
```

Bloco de código 4: Arquivo de código-fonte `main.c`

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <errno.h>
4  #include <stdlib.h>
5  #include <wiringPi.h>
6
7  #include "stepper.h"
8
9  int main(int argc, char *argv[]) {
10   if (wiringPiSetupGpio() < 0) {
11     fprintf(stderr, "Unable to start WiringPi: %s\n", strerror(errno));
12     return EXIT_FAILURE;
13   }
14
15   uint8_t stepper_ports[] = {18, 23, 24, 25};
16   size_t port_count = sizeof(stepper_ports) / sizeof(uint8_t);
17
18   iterate_array(stepper_ports, port_count, configure_stepper_io);
19
20   size_t full_rotation = 4096;    // steps for full rotation
21
22   iterate_array(stepper_ports, port_count, turn_off_coil);
23
24   puts("Starting rotation...");
25
26   // Full step rotation
27   for (size_t i = 0; i < full_rotation; i++) {
28     step_motor((i % 4), stepper_ports, port_count);
29     delay(30);
30   }
31
32   puts("Done");
33
34   return EXIT_SUCCESS;
35 }
```

Bloco de código 5: Arquivo `Makefile`

```
1 TARGET=stepper
2 TARGET_SOURCES=main.c stepper.c
3
4 FLAGS=-O2 -Wall -MMD -std=c99
5 LIBS=-lwiringPi
6
7 INCLUDE=-I.
```

```
8
9  CMP = gcc
10 LDFLAGS=$(LIBS)
11
12 .PHONY=clean
13
14 all: $(TARGET)
15
16 $(TARGET): $(TARGET_SOURCES:.c = .o)
17   $(CMP) $(FLAGS) $(INCLUDE) -o $@ $^ $(LDFLAGS)
18
19 %.o: %.c
20   $(CMP) $(FLAGS) $(INCLUDE) -c -o $@ $<
21
22 -include $(TARGET_SOURCES:.c=.d)
23
24 clean:
25   @rm -rf *.o *.d *~
```

Referências

[1] D. W. Jones, "Control of Stepping Motors – A Tutorial." https://homepage.divms.uiowa.edu/~jones/step. Último acesso em 25/05/2025.

[2] J. Tang, "Stepper Motor Basics: PM vs VR vs Hybrid." https://blog.orientalmotor.com/stepper-motor-basics-pm-vs-vr-vs-hybrid. Último acesso em 25/05/2025.

[3] E. Workbook, "Hybrid Stepper Motor – Working, Circuit Diagram and Construction." https://electricalworkbook.com/hybrid-stepper-motor, 2021. Último acesso em 25/05/2025.

[4] "Geared Stepper Motor – Arduino Forum." https://forum.arduino.cc/t/geared-stepper-motor/71308/33. Último acesso em 25/05/2025.

[5] Kiatronics, "28BYJ-48 - 5V Stepper Motor." Datasheet.

[6] T. Instrument, "ULN200x, ULQ200x High-Voltage, High-Current Darlington Transistor Arrays," 2019. Datasheet.

[7] J. G. A. Murta, "Guia do Motor de Passo 28BYJ-48 + Driver ULN2003." https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003. Último acesso em 25/05/2025.

[8] T. Ylonen, "What is ssh (secure shell)?." https://www.ssh.com/academy/ssh, 2024. Último acesso em 11/05/2025.

[9] T. Ylonen, "What is the secure shell (ssh) protocol?." https://www.ssh.com/academy/ssh/protocol, 2024. Último acesso em 11/05/2025.

[10] T. Ylonen, "What is an ssh key? an overview of ssh keys." https://www.ssh.com/academy/ssh-keys, 2024. Último acesso em 11/05/2025.

[11] "PuTTY - a free SSH and telnet client for Windows." https://www.putty.org. Último acesso em 11/05/2025.