Stepper

K využití knihovny využití je potřeba kromě krokového motoru také odpovídající řadič; nikdy nepřipojujte krokový motor přímo k Arduinu!
Knihovna je jednou ze standardních knihoven Arduino a je instalována společně s prostředím Arduino IDE.

Konstruktor[1]

Konstruktor vytvoří jednu instanci třídy Stepper, tj. jeden objekt typu Stepper. Vytvoříte-li objektů více, můžete nezávisle řídit více krokových motorů různou rychlostí a směrem. Jejich maximální počet v programu ale obecně není možné obecně určit, neboť záleží na velikostí paměti použitého Arduina, jeho rychlostí a velikosti a požadavcích konkrétního programu.
Počet parametrů konstruktoru záleží na tom, jak je řadič motoru připojen, zda dvěma nebo čtyřmi vodiči (viz Příloha 1).

Syntaxe:

```
Stepper motor(steps, pin1, pin2);
Stepper jinyMotor(steps, pin1, pin2, pin3, pin4);
```

Parametry:

steps (*byte*) – počet kroků na jednu otáčku motoru[2]
pin1, pin2 (*byte*) – dvoupinové připojení (*obr. 1 a 3 v příloze 1*)
pin1, pin2, pin3, pin4 (*byte*) – čtyřpinové připojení (*obr. 2 a 4 v příloze 1*)

Příklad:

```
Stepper motor1(32, 8, 9);
Stepper motor2(64, 8, 9, 10, 11);
```

[1] *Konstruktor je speciální funkce (metoda) třídy, která se volá výhradně ve chvíli vytváření instance této třídy.*

[2] *Pokud je velikost kroku motoru udávána jako údaj v úhlových stupních, počet kroků na otáčku vypočtěte jako podíl čísla 360 úhlem kroku (např. pro nejběžnější úhel kroku motoru 1,8° je počet kroků 360 / 1,8 = 200). V případě použití motoru s převodovkou doporučujeme pracovat s hodnotou. odpovídající počtu kroků na otáčku výstupní hřídele převodovky.*

Členské funkce (metody)[3]

setSpeed()

Funkce nastavuje rychlost v otáčkách za minutu (ot/min nebo RPM), kterou se motor bude otáčet při volání funkce step().

Syntaxe:

```
motor1.setSpeeed(rpm);
```

Parametr:

rpm (*unsigned long*) – rychlost otáčení motoru v otáčkách za minutu

Příklad:

```
motor1.setSpeed(150);
```

step()

Funkce otáčí motorem o parametrem určený počet kroků, a to rychlostí, která byla nastavena posledním voláním funkce setSpeed().
Funkce je blokující[4]*.*

Syntaxe:

```
motor1.step(steps);
```

Parametr:

steps (*int*) – počet kroků pro otáčení motoru. Kladné číslo znamená otáčení jedním směrem, záporné otáčení opačným.

Příklad:

```
motor1.step(123);
```

[3] *Členská funkce = metoda (Member function)*

[4] *Program v těle funkce setrvá tak dlouho, dokud není úloha zcela splněna. Například při zadání parametru steps 32 a zároveň nastavení 1 otáčky za minutu pomocí zavolání funkce SetSpeed s parametrem 1 setrvá program při řízení motoru s třiceti dvěma kroky na otáčku v těle funkce jednu minutu a po tu dobu blokuje veškerou ostatní činnost Arduina.*

Příloha 1:

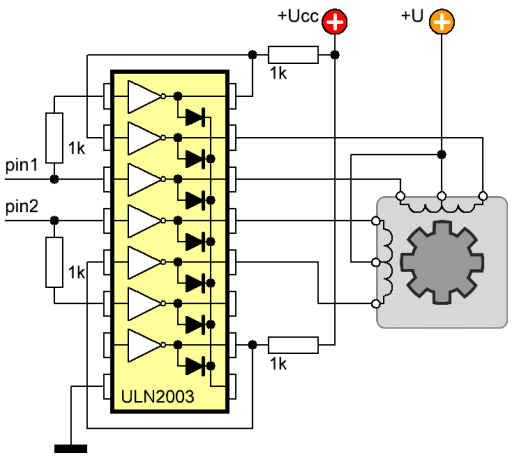
V této příloze uvádíme základní a nejjednodušší zapojení unipolárního a bipolárního motoru. V praxi se setkáme s mnohem pokročilejšími zapojeními, avšak pro pochopení principu a pro testovací připojení malého motoru k Arduinu bez nároku na velké výkony jsou zde uváděná zapojení dostatečná.

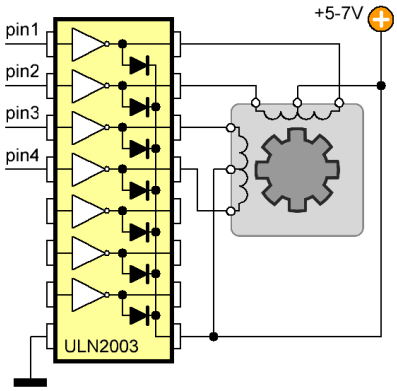
Důležité upozornění!

Ve zde uváděných příkladech zapojení lze použít pouze krokový motor, určený pro napájení napětím (v jeho parametrech je převážně uvedeno pracovní napětí 5, 12, nebo 24 V).

Motory s pracovním napětím v rozsahu do 4 V jsou určeny pro napájení stabilizovaným proudem a na tyto drivery nesmí být připojeny, jinak hrozí zničení driveru i Arduina!

Možná zapojení unipolárního motoru



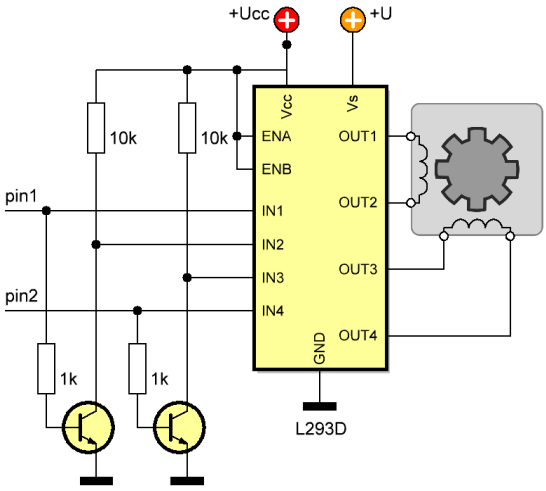


Obr. 1 – Stepper(steps, pin1, pin2)

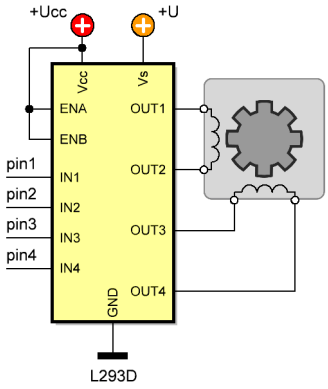
Obr. 2 – Stepper(steps, pin1, pin2, pin3, pin4)

Toto zapojení využívá integrovaný obvod ULN2003 s povoleným odběrem až 500 mA na jeden výstup. Rezistory na obr. 1 jsou ochranné a slouží k základní ochraně vstupů obvodu ULN2003 resp. výstupů připojeného Arduina.

Možná zapojení bipolárního motoru



Obr. 3 – Stepper(steps, pin1, pin2)



Obr. 4 – Stepper(steps, pin1, pin2, pin3, pin4)

Toto zapojení využívá obvod L293D s povoleným odběrem až 600 mA na jeden výstup, který je možné (kromě jiného) použít na řízení bipolárního krokového motoru. Tranzistory uvedené na obr. 3 jsou obyčejné NPN tranzistory, které slouží pouze k invertování signálu, jejich parametry jsou tudíž nepříliš podstatné a vyhoví „libovolný běžný NPN tranzistor, který najdete v šuplíku.“ Resistory s odporem 1k jsou bázové rezistory pro řízení otevření tranzistorů a zároveň chrání výstupy připojeného Arduina, rezistory s hodnotou 10k zajišťují správnou úroveň vstupů IN2 a IN3 při zavřeném tranzistoru.

Příloha 2:

V této příloze uvádíme pět příkladů pro jednoduché použití krokového motoru. Všechny příklady jsou určeny pro jeden z nejběžnějších miniaturních unipolárních krokových motorů, 28BYJ-48, s 32 kroky na otáčku a vestavěnou převodovkou s převodovým poměrem 64:1. Na jednu otočku výstupní hřídele (osy) převodovky se tedy osa motoru otočí 64x a na to je potřeba vykonat 32 x 64 = 2048 kroků. Pro jiný motor nebo jinou převodovku je potřeba tuto hodnotu patřičně upravit.

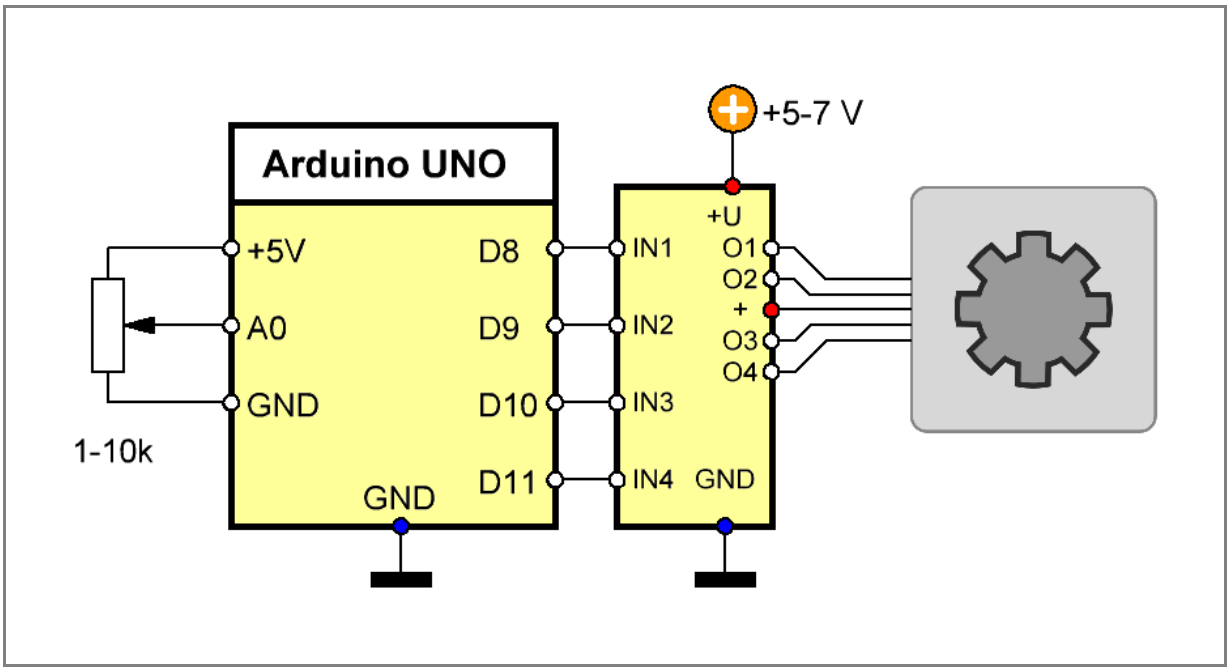
Námi použitý motor je často ve výukových sadách doplněn modulem osazeným řadičem ULN2003 (na fotografii vpravo), který umožňuje jednoduché připojení tohoto krokového motoru k Arduinu a dalším mikrokontrolerům (naopak bez něj by při přímém připojení motoru k Arduinu došlo ke zničení Arduina). Na obrázku jsou také vidět indikační LED, které svítí, když je příslušné vinutí motoru aktivováno.

Ve všech příkladech zapojujeme řadič motoru k pinům Arduina D8, D9, D10, D11, ale je možné je připojit i k jiným pinům[5].

V příkladech, kde Arduino komunikuje s počítačem, používáme přenosovou rychlost 115200 baudů. Tu je potřeba v Seriovém monitoru na počítači nastavit. Seriový monitor se spouští z Arduino IDE z menu Nástroje nebo klávesovou zkratkou Ctr+Shift+M, přenosová rychlost se volí v rozbalovacím seznamu v pravém dolním rohu okna Seriového monitoru.

[5] *Pokud má program komunikovat s počítačem, je třeba motor nepřipojovat k pinům 0 a 1.*

Schéma zapojení:



Vzorové programy

1. Nejjednodušší řízení

V tomto příkladu necháme motor točit „chvíli tam a chvíli zpět“.

```
/*
Vzorový program „Chvíli tam a chvíli zpět“ pro příručku knihovny Stepper
(c) JeDe Robot s.r.o. 2020
Stepper_01.ino
*/

// použijeme knihovnu Stepper
#include <Stepper.h>

// definice počtu kroků motoru na jednu otáčku výstupní hřídele převodovky
const int STEPS = 32*64; // 32 kroků na otáčku motoru, převodovka 64:1

// přiřazení řídících pinů pro řadič motoru
const int IN1 = 8;
const int IN2 = 9;
const int IN3 = 10;
const int IN4 = 11;

// tato deklarace globální proměnné jménem motor1 vytvoří instanci knihovny
// Stepper připojenou ke čtyřem uvedeným pinům
Stepper motor1(STEPS, IN1, IN2, IN3, IN4);

// zvolíme rychlost motoru (v otáčkách za minutu)
const long SPEED = 60;

// zvolíme dobu trvání pauzy při přepínání směru (v milisekundách)
const unsigned long PAUSE = 100;

void setup()
{
  // nastavíme rychlost otáčení motoru
  motor1.setSpeed(SPEED);
}

void loop()
{
  // necháme motor otočit výstupní osu převodovky jednou dokola
  motor1.step(STEPS);  // pro jednu otočku je potřeba STEPS kroků motoru
```

```
  delay(PAUSE);  // po otáčení chvilku počkáme

  // necháme motor otočit výstupní osu převodovky jednou dokola opačným směrem
  motor1.step(-STEPS);  // pro jednu otočku je potřeba -STEPS kroků motoru
  delay(PAUSE);  // po otáčení chvilku počkáme
}
```

2. Řízení polohy potenciometrem

V tomto příkladu se k řízení polohy hřídele motoru používá potenciometr připojený k analogovém vstupu A0. Motorek se pootáčí synchronně s pohybem osičky potenciometru, zároveň předcházíme chvění způsobenému nepřesnostmi zapojení a odečítání polohy.

```
/*
Vzorový program „Řízení polohy potenciometrem“ pro příručku knihovny Stepper
(c) JeDe Robot s.r.o. 2020
Stepper_02.ino
*/

// použijeme knihovnu Stepper
#include <Stepper.h>

// definice počtu kroků motoru na jednu otáčku výstupní hřídele převodovky
const int STEPS = 32*64; // 32 kroků na otáčku motoru, převodovka 64:1

// přiřazení řídících pinů pro řadič motoru
const int IN1 = 8;
const int IN2 = 9;
const int IN3 = 10;
const int IN4 = 11;

// vytvoříme instanci knihovny Stepper, pojmenovanou motor1
Stepper motor1(STEPS, IN1, IN2, IN3, IN4);

// potenciometr (1k až 10k) je připojen na pin A0
const uint8_t POTENTIOMETER = A0;

// zvolená konstantní rychlost otáčení motoru (v ot/min)
const long SPEED = 60;

// zvolené pásmo necitlivosti potenciometru
const byte INSENSIVITY = 2;

// předchozí hodnota na analogovém vstupu (pro potlačení chvění motoru)
int previousValue = 0;
```

```
void setup()
{
  // nastavení rychlosti otáčení motoru
  motor1.setSpeed(SPEED);
}

void loop()
{
  // pracovní proměnná pro uložení hodnoty analogového vstupu (z potenciometru)
  int currentValue;

  // do proměnné ‘currentValue‘ přečteme okamžitou hodnotu analogového vstupu
  currentValue = analogRead(POTENTIOMETER);

  // Potlačení chvění:
  // Liší se nynější a předchozí poloha potenciometru více než je stanovená necitlivost?
  if((abs(currentValue - previousValue)) > INSENSIVITY)
  {
    // Ano, liší, takže provedeme potřebný počet kroků.
    // Směr a počet kroků jsou dány rozdílem proměnných 'currentValue' a 'previousValue'
    motor1.step(currentValue - previousValue);

    // aktuální hodnotu si zapamatujeme pro příště
    previousValue = currentValue;
  }
}
```

3. Jedna otáčka

V tomto příkladu motor otočí výstupní hřídel převodovky o 360 stupňů vždy jedním a pak druhým směrem. Směr otáčení je odesílán sériovým portem; pro zobrazení spusťte sériový monitor.

```
/*
Vzorový program „Jedna otáčka“ pro příručku knihovny Stepper
(c) JeDe Robot s.r.o. 2020
Stepper_03.ino
*/

// použijeme knihovnu Stepper
#include <Stepper.h>

// definice počtu kroků motoru na jednu otáčku výstupní hřídele převodovky
const int STEPS = 32*64; // 32 kroků na otáčku motoru, převodovka 64:1

// přiřazení řídících pinů pro řadič motoru
const int IN1 = 8;
```

```
const int IN2 = 9;
const int IN3 = 10;
const int IN4 = 11;

// vytvoříme instanci knihovny Stepper, pojmenovanou motor1
Stepper motor1(STEPS, IN1, IN2, IN3, IN4);

// zvolíme konstantní rychlost otáčení motoru (v ot/min)
const long SPEED = 10;

// zvolíme dobu trvání pauzy při přepínání směru (v milisekundách)
const unsigned long PAUSE = 100;

void setup()
{
  motor1.setSpeed(SPEED);  // nastavení rychlosti otáčení motoru
  Serial.begin(115200);  // inicializace sériového portu
}

void loop()
{
  // otáčej se vpřed
  Serial.println("Vpřed!");  // informujeme uživatele - odešli řetězec "Vpřed!"
  motor1.step(STEPS);  // otoč motorem o jednu otočku vpřed
  delay(PAUSE);  // čekej zvolenou dobu

  // otáčej se dozadu
  Serial.println("Zpět!");  // informujeme uživatele - odešli řetězec "Zpět!"
  motor1.step(-STEPS);  // otoč motorem o jednu otočku vzad
  delay(PAUSE);  // čekej zvolenou dobu
}
```

4. Pomaloučku

Program otáčí motorem rychlostí přibližně[6] deset kroků za sekundu, takže je možno pouhým pohledem odhalit případné chyby při zapojování motoru. Zároveň odesílá sériovým portem počet již vykonaných kroků; pro zobrazení spusťte sériový monitor.

```
/*
Vzorový program „Pomaloučku“ pro příručku knihovny Stepper
(c) JeDe Robot s.r.o. 2020
Stepper_04.ino
*/

// použijeme knihovnu Stepper
```

[6] *Není to zcela přesně 10, ale vysvětlení, proč to není přesně 10 ale přibližně, přesahuje rámec tohoto návodu.*

```
#include <Stepper.h>

// definice počtu kroků motoru na jednu otáčku výstupní hřídele převodovky
const int STEPS = 32*64; // 32 kroků na otáčku motoru, převodovka 64:1

// přiřazení řídících pinů pro řadič motoru
const int IN1 = 8;
const int IN2 = 9;
const int IN3 = 10;
const int IN4 = 11;

// vytvoříme instanci knihovny Stepper, pojmenovanou motor1
Stepper motor1(STEPS, IN1, IN2, IN3, IN4);

// zvolíme konstantní rychlost otáčení motoru (v ot/min)
const long SPEED = 10;

// počet již vykonaných kroků
int stepCount = 0;

void setup()
{
  motor1.setSpeed(SPEED);  // nastavení rychlosti otáčení motoru
  Serial.begin(115200);  // inicializace sériového portu
}

void loop()
{
  motor1.step(1);  // krokujeme

  Serial.print("Krok: ");  // odešli řetězec "Krok:"
  Serial.println(stepCount);  // doplň aktuální počet vykonaných kroků a odřádkuj

  stepCount++;  // počet kroků zvýšíme o jedničku
}
```

5. Řízení rychlosti otáčení motoru potenciometrem

Program otáčí motorem rychlostí a směrem, určeným polohou osičky potenciometru. Nulová rychlost otáčení je uprostřed dráhy potenciometru, otáčením od středu k jednomu konci dráhy se rychlost zvyšuje jedním směrem, otáčením potenciometru od středu opačným směrem se rychlost zvyšuje opačným směrem.
Program využívá pro zjednodušení a zpřehlednění struktury uživatelsky vytvořené funkce stop() a run().

```
/*
Vzorový program „Řízení rychlosti otáčení motoru“ pro příručku knihovny Stepper
```

```
(c) JeDe Robot s.r.o. 2020
Stepper_05.ino
*/

// použijeme knihovnu Stepper
#include <Stepper.h>

// definice počtu kroků motoru na jednu otáčku výstupní hřídele převodovky
const int STEPS = 32*64; // 32 kroků na otáčku motoru, převodovka 64:1

// přiřazení řídících pinů pro řadič motoru
const int IN1 = 8;
const int IN2 = 9;
const int IN3 = 10;
const int IN4 = 11;

// vytvoříme instanci knihovny Stepper, pojmenovanou motor1
Stepper motor1(STEPS, IN1, IN2, IN3, IN4);

// potenciometr (1k až 10k) je připojen na pin A0
const uint8_t POTENTIOMETER = A0;

// maximální rychlost 1000 kroků / s
const long MAX_SPEED = 1000;

// nulové pásmo („mrtvá zóna“)
const byte DEADZONE = 20;

// výsledná hodnota rychlosti motoru (přepočtená na zvolený rozsah rychlostí -MAX_SPEED
// až MAX_SPEED)
int speed;

void setup()
{
  // prázdno - nepotřebujeme nic inicializovat
}

void loop()
{
  int val;  // okamžitá hodnota z potenciometru

  val = analogRead(POTENTIOMETER);  // přečteme analogovou hodnotu z potenciometru...

  speed = map(val, 0, 1023, -MAX_SPEED, MAX_SPEED);  // ... a převedeme jí na rozsah
                                                     // -MAX_SPEED až MAX_SPEED

  // je-li potenciometr přibližně uprostřed, stojíme, jinak točíme.
```

```
  if(abs(speed) < DEADZONE)
  {
    stop();  // jsme uvnitř mrtvého pásma a proto stojíme
  }
  else
  {
    run();  // jsme mimo mrtvé pásmo a proto točíme
  }
}

// funkce stop zastaví motor
void stop()
{
  // všechny výstupy nastavíme do nízké úrovně, aby nebylo aktivováno zádné vinutí motoru
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

// funkce run otočí motor o jeden krok směrem, který je určen znaménkem rychlosti
// (kladné pro jeden směr, záporné pro opačný)
void run()
{
  motor1.setSpeed(abs(speed));  // nastavíme motoru rychlost (vždy musí být kladná)...

  // ... a rozhodujeme, kterým směrem se má točit
  if(speed >= 0)
  {
    motor1.step(1);  // otoč motorem o jeden krok jedním směrem
  }
  else
  {
    motor1.step(-1);  // otoč motorem o jeden krok opačným směrem
  }
}
```