Servo

Knihovna Servo je určena pro řízení modelářských servomotorů (viz Příloha 1). K jejímu využití je potřeba nejméně jedno servo a odpovídající zdroj napájení.
Knihovna je jednou ze standardních knihoven Arduino a je instalována společně s prostředím Arduino IDE.

Základní vlastnosti a použití

Při použití třídy Servo můžou běžná Arduina (UNO, mini, micro apod.) řídit až 12 serv, přestane ale pracovat funkce `analogWrite()` (PWM) na pinech 9 a 10 a to bez ohledu na to, zda je k těmto pinům servo připojeno či nikoli.
Arduinem Mega je možno řídit až 48 serv. Prvních 12 serv lze používat bez narušení funkčnosti PWM, použití dalších serv deaktivuje funkci PWM na pinech 11 a 12.

Arduinem Due je možno řídit až 60 serv.

Knihovna používá časovače (timery), proto při současném použití jiných knihoven zkontrolujte, zda nejsou ve vzájemném konfliktu (počet časovačů, konkrétní přiřazení a nastavení záleží na konkrétním typu Arduina a velmi se mezi sebou liší).

Třída Servo

Konstruktor

Konstruktor se automaticky zavolá při deklaraci proměnné; vytvoří jednu instanci třídy Servo, tj. jeden objekt typu Servo.

Vytvoříte-li objektů více, můžete nezávisle řídit více serv najednou.

Příklad:

```
Servo servo1;
```

Členské funkce (metody)

attach()

Vytvoří propojení daného serva s fyzickým pinem a volitelně může nastavit i jeho počáteční a koncovou pozici.

Syntaxe:

```
servo1.attach(pin, [min], [max]);
```

Parametry:

pin (*byte*) – číslo pinu, na nějž bude servo připojeno
min (*double*) – [*nepovinné*] doba trvání pulsu v mikrosekundách, která určuje výchozí pozici serva (výchozí hodnota je 544)
max (*double*) – [*nepovinné*] doba trvání pulsu v mikrosekundách, která určuje koncovou pozici serva (výchozí hodnota je 2400)
Pokud nejsou uvedeny parametry min a max, použijí se výchozí hodnoty 544 a 2400[1].

Návratová hodnota:

Pořadí serva[2], pokud je vše v pořádku, nebo 255, pokud byl překročen maximální možný počet serv.

Příklad:

```
servo1.attach(3, 500, 2500);
```

Důrazně doporučujeme při použití této funkce uvádět parametry min a max. Pro normální serva jsou správné hodnoty 500 a 2500 místo výchozích, tj. inicializuje se například příkazem `servo1.attach(3, 500, 2500);`. Špatné nebo žádné nastavení parametrů `min` a `max` má za následek to, že při polohování serva pomocí funkce `write()` bude výsledný úhel natočení chybný.

Po zavolání této funkce se servo ihned začne natáčet do polohy dané naposledy použitou hodnotou parametru funkcí `write()` nebo `writeMicroseconds()`, nebo (pokud žádná z těchto dvou funkcí dosud volána nebyla) hodnotou 1500 jako parametrem funkce `writeMicroseconds()`.

Standardní serva mají rozsah pohybu ±45° od středové polohy. Řada serv má rozsah větší, často ±90° (nebo i více) od středové polohy[3].

detach()

Zruší propojení daného serva s fyzickým pinem.

Syntaxe:

```
servo1.detach();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

Příklad:

```
servo1.detach();
```

Poznámka:

Pokud jsou odpojeny všechny instance třídy Servo, lze opět pro výstup PWM pomocí funkce `analogWrite()` používat piny 9 a 10 (resp.11 a 12 u Arduino Mega) .

write()

Generuje pulzy pro servo tak, aby poloha hřídele odpovídala zadanému úhlu.

Syntaxe:

```
servo1.write(angle);
```

Parametry:

angle (*int*) – hodnota v rozsahu 0 až 180 určuje úhel natočení hřídele serva. Hodnoty menší než 0 jsou nahrazeny 0 a hodnoty větší než 180 jsou nahrazeny 180.

Návratová hodnota:

Funkce nic nevrací.

Příklad:

```
servo1.write(120);
```

Poznámka:

Parametr **angle** určuje u standardního serva úhel natočení jeho hřídele ve stupních (0 až 180 stupňů)[5]. U serva s nepřetržitým otáčením tento parametr určuje rychlost a směr otáčení: při 0 se servo točí plnou rychlostí v jednom směru, při 180 se točí plnou rychlostí v opačném směru a při 90 se otáčení serva zastaví[6].

Toto nastavení úhlu resp. rychlosti funguje správně pouze při inicializaci rozsahu pohybu serva odpovídajícím způsobem (viz popis funkcí `attach()` a `writeMicroseconds()`). V každém případě bude hodnota parametru přepočítána pro nastavení polohy serva tak, že 0 odpovídá hodnotě `min` a 180 hodnotě `max` použité při volání funkce `attach()`.

writeMicroseconds()

Generuje pulzy o zadané době trvání vysoké úrovně signálu, podle něhož se nastavuje poloha hřídele serva[7].

Syntaxe:

```
servo1.writeMicroseconds(us);
```

Parametry:

us (*int*) – délka pulsu v mikrosekundách

Návratová hodnota:

Funkce nic nevrací.

Příklad:

```
servo1.writeMicroseconds(1234);
```

Poznámka:

Parametr **us** určuje dobu trvání výstupního signálu v mikrosekundách (µs nebo us[8]). Hodnota musí být v rozsahu určeném při připojování serva pomocí funkce `attach()`.

U standardních serv doba trvání výstupního signálu určuje natočení serva. U serv s nepřetržitým otáčením tato doba určuje rychlost i směr obdobně, jak bylo popsáno u metody write.

U standardních serv je povolený rozsah natočení ±45° od střední polohy, čemuž odpovídají hodnoty parametru 45 až 135 pro funkci `write()`[9], resp. 1000 až 2000 pro funkci `writeMicroseconds()`. Řada běžných serv má skutečný možný rozsah ±90°, takže je možné používat hodnoty 0 až 180 pro `write()` resp. 500 až 2500 pro `writeMicroseconds()`. V obou případech je třeba servo připojit pomocí funkce `attach()` tak, že její parametr `min` bude reprezentovat natočení o čtvrt otáčky od středové polohy jedním směrem a `max` natočení o čtvrt otáčky druhým směrem, pro standardní serva tedy např. `servo1.attach(3, 500, 2500);`.

U serva s menším možným úhlem natočení než ±90° musíme parametry funkce `attach()` nastavit, jakoby se servo mohlo natáčet o ±90°, ale nesmíme požadovat natočení mimo jeho skutečný rozsah, ať už při volání funkce `write()` nebo `writeMicroseconds()`.

Pokud bychom chtěli u serva, které se mechanicky může natáčet i ve větším rozsahu než ±90°, využít i tento zvětšený rozsah, pak již nemůžeme použít knihovnu Servo a museli bychom si obsluhu serva naprogramovat sami.

Upozornění:

V mnoha materiálech je uvedena informace, že 1500 µs odpovídá 90°, resp. že zavolání `writeMicroseconds(1500)` nastaví servo do střední polohy stejně jako zavolání `write(90)`. To je pravda pouze pro serva, která jsou inicializována funkcí `attach()` se správnými parametry, nikoli se špatnými nebo vynechanými. Vždy je totiž poloha natočení dána takto: při požadavku na natočení 0° (tj. zavoláním `servo1.write(0);`) se servo natočí do polohy odpovídající hodnotě `min` z funkce `attach()`, pro 90° (`servo1.write(90);`) do poloviny mezi `min` a `max` a pro 180° (`servo1.write(180);`) se servo natočí do polohy dané parametrem `max`. Avšak pouze u serv správně nastavených vhodnými parametry funkce `attach()` bude požadavek na natočení odpovídat úhlu natočení i ve skutečném světě.

S vynechanými parametry při propojování serva pomocí funkce `attach()` bude příkaz `servo1.write(90);` odpovídat příkazu `servo1.writeMicroseconds(1472);` a to proto, že vynecháním parametrů `min` a `max` se použijí předdefinované hodnoty 544 a 2400, přičemž polovina mezi 544 a 2400 není 1500, ale právě 1472. Rozdíl v natočení mezi 1500 a 1472 je cca 2,5°, v případě kontinuálního serva se bude servo mírně otáčet..

read()

Vrací naposledy nastavenou hodnotu polohy serva funkcemi write() nebo writeMicroseconds().

Syntaxe:

```
servo1.read();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Úhel serva od 0 do 180 stupňů (*int*).

Příklad:

```
pozice = servo1.read();
```

Poznámka:

Hodnota, kterou tato funkce vrátí, obvykle znamená aktuální polohu serva. Někdy se ale může vrácená hodnota od skutečné polohy serva lišit. To se stane například tehdy, když od posledního nastavení uběhl teprve příliš krátký čas na to, aby se servo příslušně pootočilo, nebo když na servo působí vnější sily tak velké, že servo nemá potřebný točivý moment k dosažení nebo udržení požadované polohy.

readMicroseconds()

Vrací naposledy nastavenou hodnotu polohy serva funkcemi write() nebo writeMicroseconds().

Syntaxe:

```
servo1.readMicroseconds();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Délka pulsu v mikrosekundách (*int*).

Příklad:

```
delka = servo1.readMicroseconds();
```

Poznámka:

Hodnota, kterou tato funkce vrátí, obvykle znamená aktuální polohu serva. Někdy se ale může vrácená hodnota od skutečné polohy serva lišit. To se stane například tehdy, když od posledního nastavení uběhl teprve příliš krátký čas na to, aby se servo příslušně pootočilo, nebo když na servo působí vnější sily tak velké, že servo nemá potřebný točivý moment k dosažení nebo udržení požadované polohy.

attached()

Vrací informaci, zda je toto servo propojeno s fyzickým pinem.

Syntaxe:

```
servo1.attached();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Stavová informace (*bool*)

TRUE – pokud je určené servo propojeno s pinem[10]

FALSE – v opačném případě, tj. servo není s pinem propojeno

Příklad:

```
if(servo1.attached())
{
  Serial.println("Servo JE připojeno.");
}
else
{
  Serial.println("Servo NENÍ připojeno.");
}
```

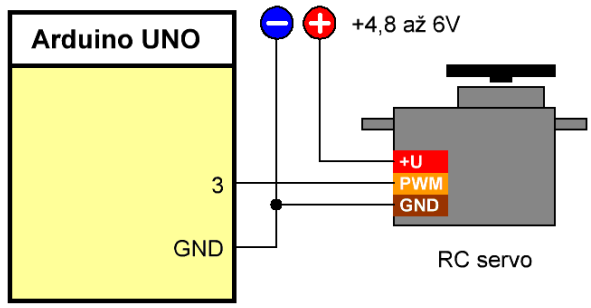
Příklady:

V následujících příkladech používáme Arduino UNO a servo připojujeme k doplňku „Connector Shield“. Napájení serv je z odděleného zdroje (4,8 až 6 V, proud alespoň 1 A na jedno servo). Dejte pozor, abyste zdroj nepřipojili přímo k Arduinu.

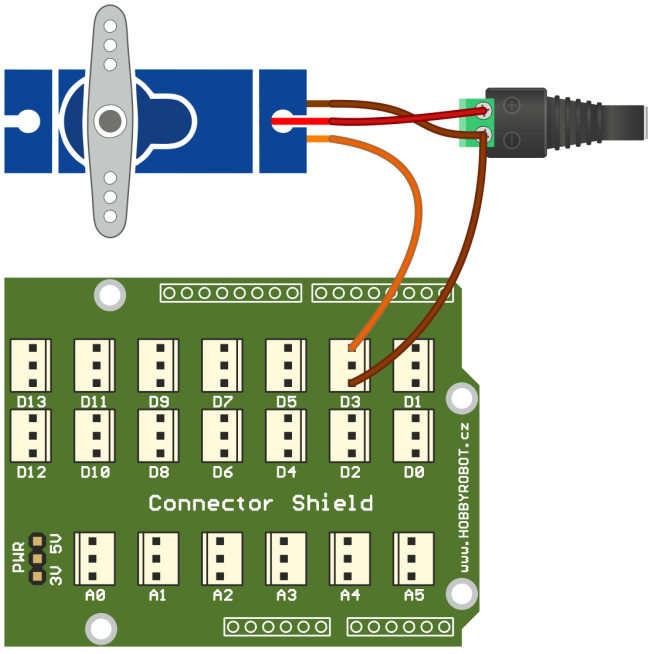
Cik-cak

Otáčí servem od jednoho konce dráhy pohybu k druhému a zpět.

Zapojení:



Rozložení součástek:



Barvy vodičů a zapojení serva viz příloha na konci tohoto dokumentu.

Program:

```
/*
Vzorový program „Cik-cak“ pro příručku knihovny Servo
(c) JeDe Robot s.r.o. 2020
Servo_01.ino
*/

#include <Servo.h>
Servo mojeServo; // vytvoří instanci třídy Servo s názvem mojeServo

// Zvolíme, kam bude připojen řídící pin serva
const int SERVOPIN = 3;

void setup()
{
  // Propojíme servo s pinem SERVOPIN a určíme standardní rozsah pohybu
  servo1.attach(SERVOPIN, 500, 2500);
}

void loop()
{
  // proměnná, která bude představovat aktuální pozici serva
  int pos = 0;

  // budeme otáčet servem z polohy 0 do polohy 180 stupňů v krocích po jednom stupni
  for (pos = 0; pos <= 180; pos++)
  {
    servo1.write(pos);  // natočíme servo do pozice, dané proměnnou ‚pos‘
    delay(15);  // prodleva 15 ms umožní, aby servo dojelo na požadovanou pozici
  }

  // nyní budeme otáčet servem zpět z polohy 180 do polohy 0 stupňů
  // opět po jednom stupni
  for (pos = 180; pos >= 0; pos--)
  {
    servo1.write(pos);  // natočíme servo do pozice, dané proměnnou ‚pos‘
    delay(15);  // prodleva 15 ms umožní, aby servo dojelo na požadovanou pozici
  }
}
```

Metronom

Přehazuje servo z jedné strany na druhou ve vteřinové[11] frekvenci .

Zapojení je stejné, jako v předchozím příkladu.

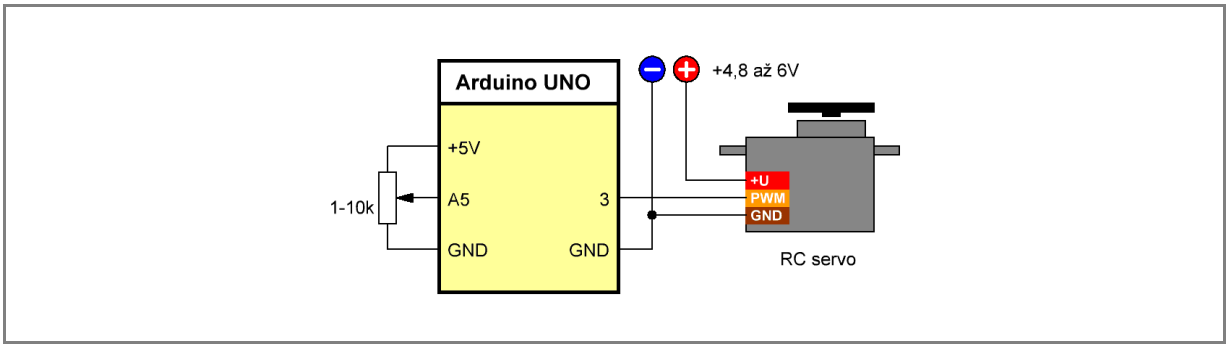
Program:

```
/*
Vzorový program „Metronom“ pro příručku knihovny Servo
(c) JeDe Robot s.r.o. 2020
Servo_02.ino
*/

#include <Servo.h>
Servo mojeServo; // vytvoří instanci třídy Servo s názvem mojeServo

// Zvolíme, kam bude připojen řídící pin serva
const int SERVOPIN = 3;

// Zvolíme časovou konstantu (v milisekundách)
const unsigned long TIME = 1000;

// Zvolíme pozice, mezi kterými bude servo překlápět
const int LEFT = 90-45;  // budeme naklápět o 45 stupňů sem a tam
const int RIGHT = 90+45;

void setup()
{
  // Propojíme servo s pinem SERVOPIN a určíme standardní rozsah pohybu
  servo1.attach(SERVOPIN, 500, 2500);
}

void loop()
{
    servo1.write(LEFT);  // natočíme servo na jednu stranu
    delay(TIME);  // prodleva

    servo1.write(RIGHT);  // natočíme servo na druhou stranu
    delay(TIME);  // prodleva
}
```

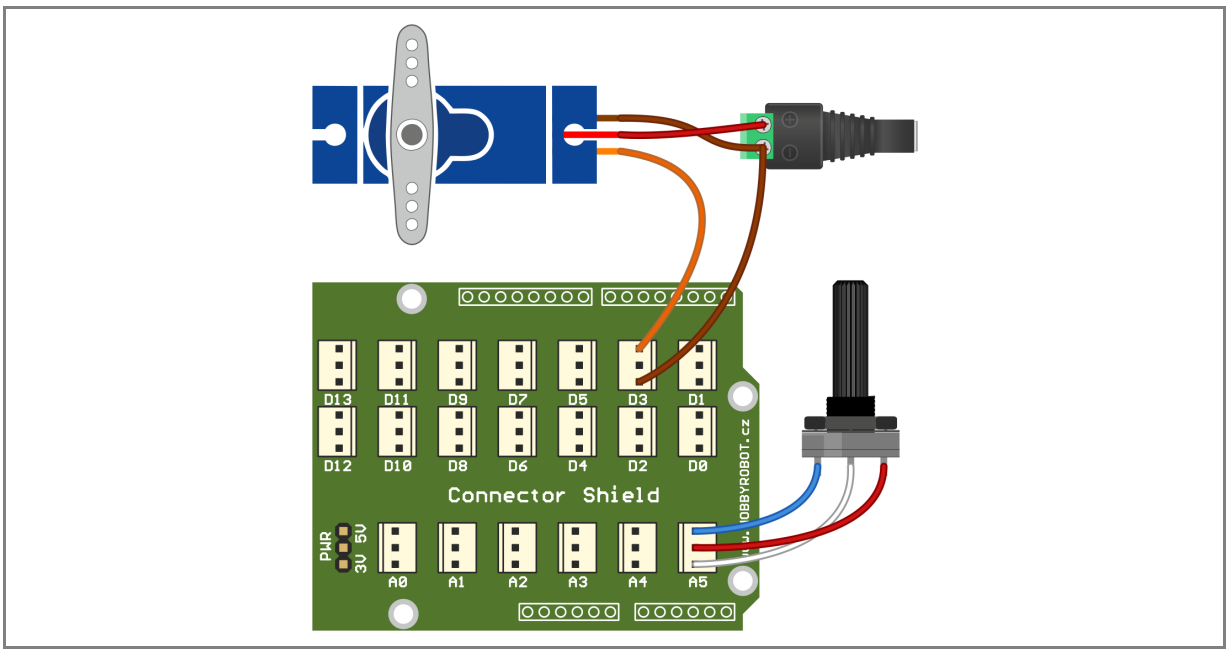
Nastavení polohy tlačítkem

Ovládání serva tlačítkem – na první zmáčknutí přejede na jednu stranu, na další přejede na opačnou stranu atd.

Zapojení:



Rozložení součástek:



Program:

```
/*
Vzorový program „Nastavení polohy tlačítkem“ pro příručku knihovny Servo
(c) JeDe Robot s.r.o. 2020
Servo_03.ino
*/

#include <Servo.h>
Servo mojeServo;  // vytvoří instanci třídy Servo s názvem mojeServo
```

```
// Zvolíme, kam bude připojen řídící pin serva
const int SERVOPIN = 3;

// Zvolíme, kam bude připojeno tlačítko
const int BUTTONPIN = 4;

void setup()
{
  // Propojíme servo s pinem SERVOPIN a určíme standardní rozsah pohybu
  servo1.attach(SERVOPIN, 500, 2500);

  // nastavíme pin pro připojení tlačítka jako vstupní
  // s aktivovaným zvyšujícím rezistorem
  pinMode(BUTTONPIN, INPUT_PULLUP);
}

void loop()
{
  waitForButton();  // počkáme na zmáčknutí a uvolnění tlačítka
  servo1.write(0);  // nastavíme pozici serva na jednu stranu

  waitForButton();  // počkáme na zmáčknutí a uvolnění tlačítka
  servo1.write(180);  // nastavíme pozici serva na druhou stranu
}

// pomocná funkce, která čeká, až zmáčkneme a zase pustíme tlačítko
void waitForButton(void)
{
  // počkáme na zmáčknutí tlačítka - neděláme nic, dokud není zmáčknuté
  do
  {
    // nic
  } while(digitalRead(BUTTONPIN) == HIGH);

  // nyní počkáme na jeho uvolnění - neděláme nic, dokud je zmáčknuté
  do
  {
    // nic
  } while(digitalRead(BUTTONPIN) == LOW);

  // nyní víme, že tlačítko bylo zmáčknuté a pak zase uvolněné
}
```

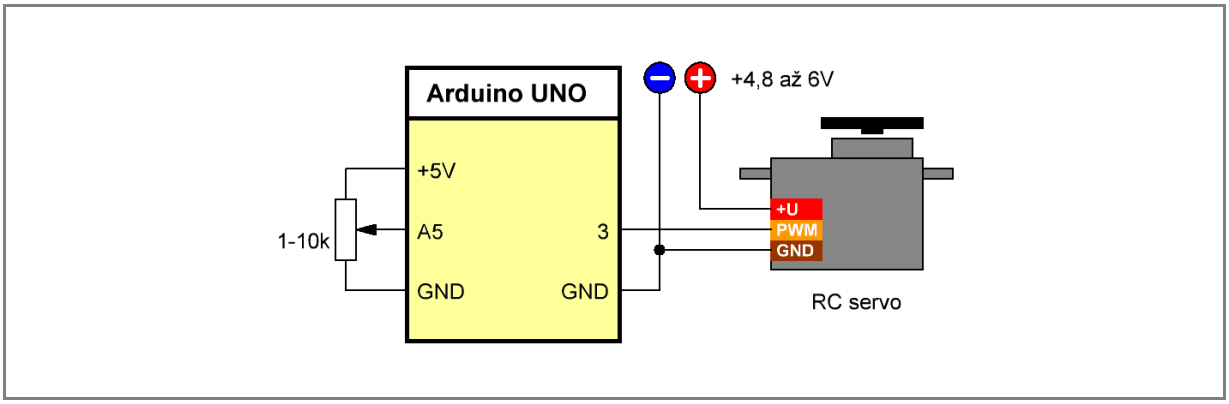
Poznámka:

Tento program nijak neřeší tzv. „zákmity“ tlačítka, které v nepříznivé situaci mohou vyvolat při každém zmáčknutí tlačítka mnohonásobně opakovaný signál. Pro základní vyzkoušení však obvykle toto nepředstavuje velký problém.

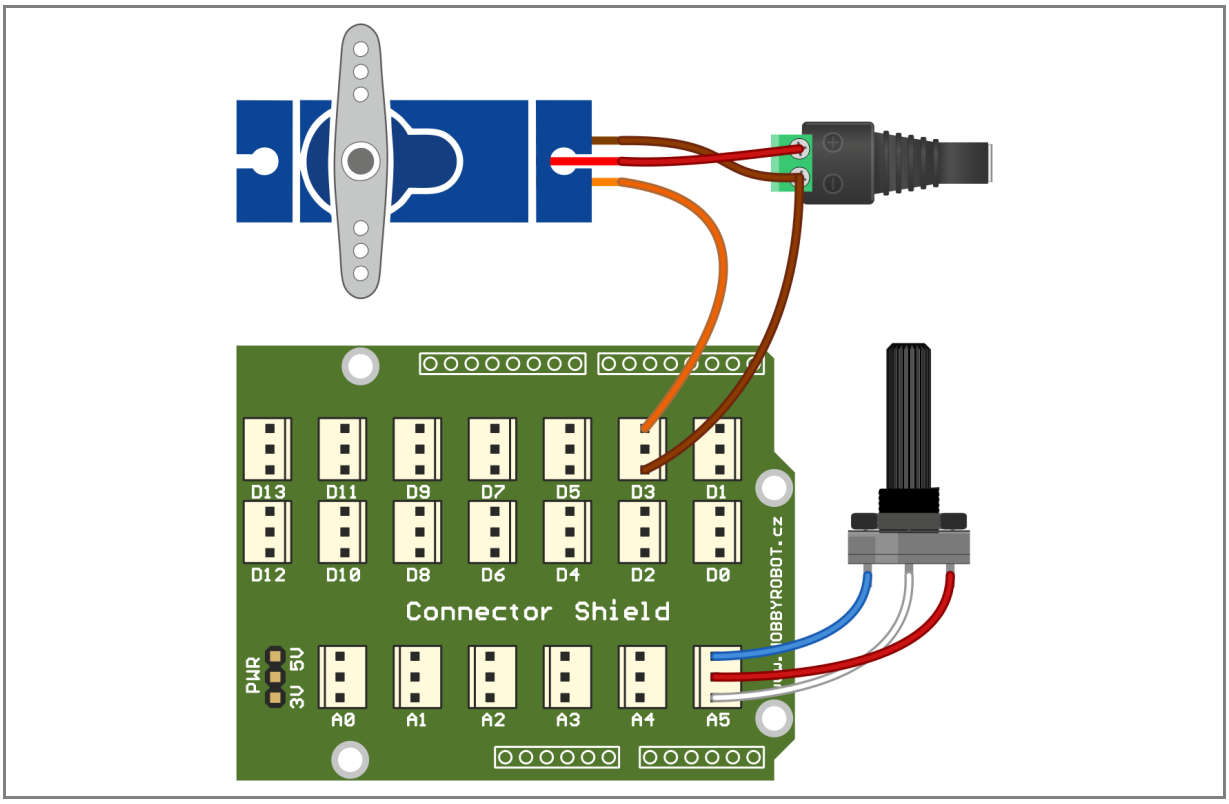
Ruční ovládání

Ovládání serva potenciometrem (servotester).

Zapojení:



Rozložení součástek:



Program:

```
/*
Vzorový program „Ruční ovládání“ pro příručku knihovny Servo
(c) JeDe Robot s.r.o. 2020
Servo_04.ino
*/

#include <Servo.h>
Servo mojeServo;  // vytvoří instanci třídy Servo s názvem mojeServo

// Zvolíme, kam bude připojen řídící pin serva
const int SERVOPIN = 3;

// Zvolíme, kam bude připojen potenciometr
const int POTPIN = A5;

void setup()
{
  // Propojíme servo s pinem SERVOPIN a určíme standardní rozsah pohybu
  servo1.attach(SERVOPIN, 500, 2500);
}

void loop()
{
  // proměnná, která bude představovat aktuální hodnotu napětí
  // z analogového vstupu A5
  int analogValue = 0;

  // proměnná, která bude představovat požadovaný úhel natočení serva
  int position = 0;

  // přečteme aktuální hodnotu napětí, nastaveného
  // potenciometrem (rozsah od 0 do 1023)
  analogValue = analogRead(POTPIN);

  // přepočtení proměnné val z rozsahu 1023 na rozsah 0 až 180
  position = map(analogValue, 0, 1023, 0, 180); 0 až

  servo1.write(position);  // nastavíme pozici serva do požadované polohy

  delay(15);  // prodleva 15 ms umožní, aby servo dojelo na požadovanou pozici
}
```

Poznámka:

Pokud potenciometrem otáčíme rychle sem-tam z jedné strany na druhou, nemusí být prodleva 15 ms ve funkci loop dostatečná pro dosažení požadovaného úhlu natočení serva,

protože servo nemusí být tak rychlé. Necháme-li však potenciometr po natočení v klidu, servo požadovanou polohu nastaví a bude udržovat i přes případný mírný tlak, kterým bychom jej chtěli vychýlit.

Příloha 1

Takzvaná „modelářská“ nebo „RC“ serva byla původně určena pro použití v radiem řízených (RC) modelech. Jsou však také dobře použitelná pro řadu školních experimentů, řízení laboratorních zařízení, automatizaci domácnosti (například polohování žaluzií) a podobné použití[12].

Základem běžného serva je stejnosměrný elektromotorek, vícestupňová převodovka sestavená z ozubených kol, zpětnovazební snímač polohy výstupního hřídele (potenciometr nebo bezkontaktní snímač) a řídící elektronika. Požadovaná pozice výstupní osy serva je přenášena do řídící elektroniky z nadřazeného řídícího systému pomocí pulzně šířkově (PWM) modulovaného řídícího signálu.

Servo upravené pro kontinuální otáčení má obvykle zpětnovazební snímač nahrazen dvojicí pevných rezistorů[13] a mají odstraněny vnitřní mechanické koncové dorazy, jinak je servo a zejména jeho elektronika beze změny. Pevné rezistory řídící elektronika serva vyhodnotí, jako by servo bylo nastaveno stále do střední polohy. Pokud jej budeme funkcemi `write()` nebo `writeMicroseconds()` nastavovat na jinou hodnotu, než jaká odpovídá střední poloze, pak řídící se elektronika pokusí servo natočit na příslušnou polohu a to tím rychleji, čím větší je rozdíl mezi aktuální a požadovanou polohou (tak to dělají serva vždy). Protože je ale zpětnovazební snímač u takto upraveného serva nahrazen rezistory, servo se do požadované polohy nikdy nedostane a proto bude točit motorem neustále. Rychlost otáčení je závislá na rozdílu požadované hodnoty od střední hodnoty (tj. od 90° nebo od poloviny mezi `min` a `max` použitými ve funkci `attach()`) – čím větší rozdíl, tím rychleji. Směr otáčení bude dán tím, je-li požadovaná hodnota menší nebo větší než 90° (obvykle proti směru hodinových ručiček pro <90 a po směru hodinových ručiček pro >90). Pokud servu nastavíme úhel natočení 90°, bude stát (je-li správně nastaveno, viz popis funkcí `attach()` a `writeMicroseconds()`).

Napájení a připojení

Do serva vedou tři vodiče; dva servo napájejí a po třetím jsou přenášeny povely. Jako připojovací konektor se obvykle používá plochý trojpinový typ s roztečí dutinek 2,54 mm. Tvar pouzdra konektoru je závislý na výrobci, ale kterýkoli z nich lze nasunout na pinovou lištu se správnou roztečí kolíků.

Napájecí napětí běžných serv se pohybuje v rozmezí 4,8 až 6 V. Takzvaná mikroserva pracují většinou už od napájecího napětí 3 V, ovšem při tomto napětí jsou už nespolehlivá a točivý moment zcela zanedbatelný.

Moderní serva mohou naopak pracovat i při vyšších napájecích napětích (7,2 V; 8,4 V nebo i 12 V), což umožňuje větší výkon serva. Jejich cena je ale vysoká a pro běžné aplikace jsou zbytečná.

Aby servo mohlo správně pracovat, je třeba, aby napájecí zdroj dodal odpovídající proud; většina serv má špičkový odběr v řádu jednotek ampér. Napájení z Arduina, které je napájeno pouze z USB, proto nemusí stačit už při připojení jediného serva a může dojít k cukání serva, zablokování nebo náhodným restartům Arduina, v extrémních případech ke zničení Arduina nebo dokonce připojeného počítače. I v případě napájení Arduina z externího zdroje doporučujeme oddělit napájení serv, abyste zabránili pronikání nežádoucího šumu do řídící elektroniky a snížili tak nebezpečí zmíněných problémů.

Význam	*Barevná kombinace*		
[-] napájení (GND)	černá	černá	hnědá
[+] napájení	rudá	rudá	rudá
Řídící signál	bílá	žlutá	oranžová
Používá výrobce	*Futaba*	*Hitec*	*Graupner*

Obr. 1: Barevné značení kabelů serv

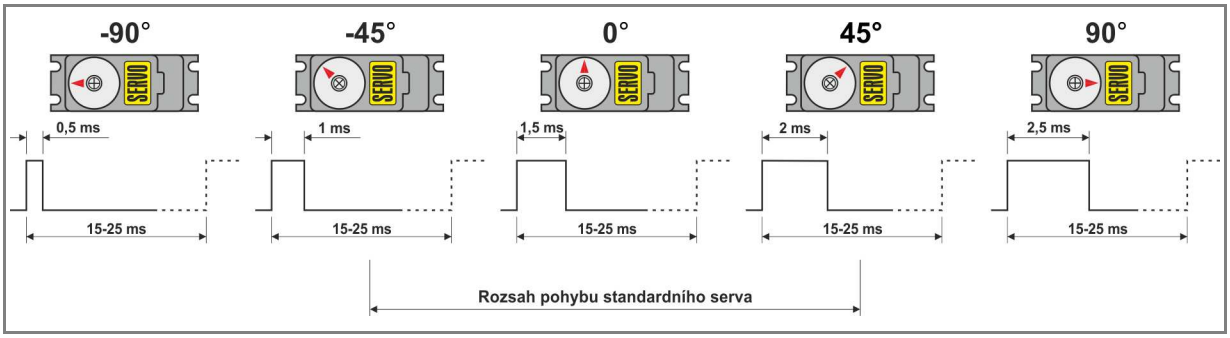
Řídící signál serva

U běžných serv tohoto typu se řízení polohy provádí řídícím signálem, který tvoří opakované impulsy. Poloha výstupního hřídele serva odpovídá proporcionálně šířce řídícího impulzu. Řídící impulz je pozitivní s amplitudou od 3 V do napájecího napětí a aktivní dobou trvání proměnnou od 1 do 2 ms. Době trvání řídícího pulzu 1-2 ms odpovídá rozsah polohy výstupního hřídele serva ±45° od středové polohy. Běžná serva mají obvykle větší mechanický rozsah pohybu výstupní osy, takže zvětšením rozsahu řídících impulzů na šířku od 0,5 do 2,5 ms je možné takové servo nastavovat běžně v rozsahu ±90°. Je však třeba dodat, že řídící impulzy mimo rozsah daný výrobcem serva mohou způsobit najíždění serva na mechanický doraz a tím jeho poškození (mechanické poškození nárazem nebo poškození řídící elektroniky jejím přetížením).

Řídící impulz se obvykle opakuje 50x za sekundu (tj. perioda 20 ms). Tato hodnota však není zcela kritická, závisí na ní především dosažitelný točivý moment a klidový přídržný moment serva (poloha je dána době trvání vysoké úrovně signálu). Výstupní hřídel serva se do střední polohy nastavuje impulzem délky 1,5 ms (takže celkově je signál 1,5 ms ve vysoké úrovni a 18,5 ms v nízké).

Modernější serva mohou být také řízena výše uvedeným signálem, ale interně pracují s digitálním řízením zabudovaného motoru a obvykle s výrazně vyšší frekvencí řízení motorku. Taková serva se na první pohled chovají velmi podobně až stejně, jako zmíněná levná serva, ale obvykle dosahují přesnějšího polohování a hladšího chodu.

Existují také ještě tzv. „digitální serva“ [14], která mají se servy zmíněnými v této publikaci společné snad pouze to, že je možné je nějakým způsobem nastavovat jejich polohu. Jejich řízení je obvykle pomocí nějakého digitálního protokolu a poskytují řadu možností nastavení, řízení, zpětné vazby, programování pohybu atd. a spíš než „motorek co se dá nastavit na polohu“ jsou „malý počítač, co dokáže velmi sofistikovaným způsobem točit a nastavovat polohu“.



Obr. 2: Řídící signál běžného analogového serva

Příloha 2

Autoři knihoven pro Arduino nachystali nepozorným uživatelům nejednu poťouchlost. Tahle se týká ovládání serv.

Knihovna Servo v Arduinu poskytuje rozhraní, které pro řízení nabízí dvě funkce, `write()` a `writeMicroseconds()`. Funkce write umožňuje nastavit servo na požadovaný úhel, takže `write(0)`; nastaví servo na úhel 0°, `write(90)`; na 90°, `write(180)`; na 180° atd. Funkce `writeMicroseconds()` nastavuje délku signálu v mikrosekundách, takže `writeMicroseconds(1500)`; nastaví servo doprostřed, `writeMicroseconds(1000)`; na jednu stranu a writeMicroseconds(2000); na druhou.

Jenže ono to tak úplně není! Parametr předaný funkci `write()` se totiž přepočítává podle rozsahu serva tak, aby 0 odpovídala dolní hodnotě pro řízení serva a 180 horní. Ten rozsah se dá nastavit při zapínání serva pomocí funkce `attach()`, ale většina uživatelů používá tuto funkci zjednodušenou, jen s jedním parametrem určujícím pin, ke kterému je servo připojeno (např. `mojeServo.attach(12)`). Existuje ale ještě druhá varianta funkce `attach()`, kdy parametrem je kromě pinu Arduina ještě minimum a maximum doby signálu v mikrosekundách. Pokud použijete první variantu, tak se ty zbylé dva parametry, minimum a maximum, nastaví na default. No a tady právě vycítil čertík příležitost, a našeptal vývojářům Arduina, aby nastavili ty defaultní hodnoty na 544 (minimum) a 2400 (maximum). Kolik je tedy uprostřed? Není to 1500, ale 1472.

Výsledkem tedy je, že když jednou servo nastavíte do poloviny rozsahu příkazem `writeMicroseconds(1500)` a podruhé příkazem `write(90)`, tak to nebude totéž. Bude to o kousek vedle, tak asi o 2,5 stupně. V případě serva upraveného pro kontinuální otáčení pak nebude oboje znamenat stop, ale jedno volání motor zastaví, zatímco druhé ho nechá pomalu točit.

Funkci `write()` je možno též místo úhlu zadat dobu trvání pulzu v mikrosekundách. Vývojáři nejspíše usoudili, že vyjdou vstříc lenivým uživatelům, kteří si nepamatují správná jména funkcí, a upravili zdrojový kód tak, že když je parametr funkce `write()` menší než minimum, tak to je úhel, a pokud je větší nebo roven, tak to je doba v mikrosekundách. A navíc se to testuje proti minimu defaultnímu, tj. 544, nikoli tomu, které si uživatel zadal pomocí funkce attach. To už je vyloženě nepříjemné!

Shrnutí:

Zásadně servo dekladujte s uvedením minima a maxima, např. `zamerovac.attach( 12, 900, 2100 )` abyste se vyhnuli problémům s polohováním.

Vysvětlivky

[1] Výchozí hodnoty 544 a 2400 jsou smutným historickým dědictvím, když tehdejší autor knihovny podle všeho použil jakési nestandardní servo, které tyto hodnoty vyžadovalo, zatímco pro standardní serva a drtivou většinu normálních běžně dostupných serv i návodů k použití jsou správné hodnoty 500 a 2500. Použití jiných hodnot než těchto může mít za následek špatný pohyb standardního serva na zvolené pozice nebo v případě serv s nepřetržitým otáčením to, že se servo při požadavku na zastavení nezastaví zcela a bude se pomalu otáčet. Autoři systému Arduino však tuto chybu v knihovně odmítli opravit s odkazem na to, že by oprava mohla ovlivnit již dříve existující programy Arduino, jejichž uživatelé by nemuseli vědět, jak svůj program správným hodnotám přizpůsobit…

[2] Toto pořadí serva není prakticky k ničemu jinému, než k případnému ověření, že jste dosud nevyčerpali maximální možný počet serv. Na každém Arduinu jich může být přinejmenším 12, proto většina příkladů s „jen pár servy“ tuto hodnotu vůbec netestuje, stejně jako v příkladech v této publikaci.

[3] Můžete to vyzkoušet na svém servu, ale pečlivě dbejte, aby servo trvale nedojíždělo až na mechanické koncové dorazy, protože tehdy je elektronický ovládací obvod přetížen a může se zničit.

[5] U nestandardních nebo méně kvalitních serv zadaný úhel nemusí přesně odpovídat realitě.

[6] Vztah však není lineární, tj. 45 neznamená poloviční rychlost. Skutečná rychlost záleží na konkrétním servu.

[7] Tato funkce je duální k funkci `write()`, oběma je možné nastavovat polohu.

[8] V bežném textu a počítačových programech se dílčí předpona jednotek SI *mikro* pro vyjádření 10^{-6} často píše jako malé písmeno u latinkou místo správného μ z řecké abecedy (z řeckého μικρός, mikros, česky „malý“).

[9] Funkce write chápe 90 jako uprostřed, 45 je tedy natočení o 45° na jednu stranu (90-45) a 135 je natočení o 45° na druhou stranu (90+45).

[10] Propojení je ve smyslu programové obsluhy (tj. zda bylo nebo nebylo aktivováno funkcí `attach()`). Arduino nemá možnost zjistit, jestli je se servem propojeno fyzicky (tj. „jestli k němu vede drát“).

[11] Není to zcela přesně vteřina; pokud byste měřili přesnými stopkami, zjistíte, že se po čase tento metronom postupně bude od přesného tikání odchylovat. Vysvětlení by ale bylo zbytečně složité nad rámec tohoto dokumentu.

[12] Existují také jak velmi velká, tak i velmi malá serva; největší servomotory pro běžné průmyslové použití údajně vyrábí GE Fanuc Automation: řada DiS-B servomotorů obsahuje serva s hmotností přes 100kg, která disponují točivým momentem až 10 kNm, naopak nejmenší ještě dostupná serva váží kolem gramu a mají točivý moment v řádu jednotek miliNewtonmetrů.

[13] Někdy jsou tyto rezistory doplněny ještě potenciometrem pro doladění nulové polohy.

[14] Po jejich rozšíření se o zde popisovaných „modelářských“ nebo „RC“ servech hovoří také jako o „analogových servech“.