VarSpeedServo

Knihovna umožňuje současně nezávisle na sobě pohybovat osmi RC servy. U každého z nich je možno nastavovat rychlost pohybu, blokovat vykonání dalšího příkazu před dokončením pohybu podle toho předchozího a vytvářet sekvence pohybů.
Knihovna VarSpeedServo.h, která vznikla úpravou standardní knihovny Servo.h, používá přerušení.

Konstruktor

VarSpeedServo()

Aby bylo možno řídit více serv naráz, je nutno vytvořit více instancí třídy VarSpeedServo.

Syntaxe:

```
VarSpeedServo servo1;
```

vytvoří objekt s názvem servo1

```
VarSpeedServo servo2;
```

vytvoří objekt s názvem servo2
atd.

Členské funkce (metody)

Výkonné metody

attach()

Určuje pin k připojení serva a volitelně může nastavit i jeho počáteční a koncovou pozici.

Syntaxe:

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

Parametry:

pin (*uint8_t*) – číslo pinu, na který chcete připojit servo
min (*uint16_t*) – [*nepovinné, výchozí hodnota je 544*] počáteční pozice serva. Hodnoty se zadávají v mikrosekundách (µs).
max (*uint16_t*) – [*nepovinné, výchozí hodnota je 2400*] koncová pozice serva. Hodnoty se zadávají v mikrosekundách (µs).

Návratová hodnota:

Funkce nic nevrací.

detach()

Zruší propojení daného serva s fyzickým pinem.

Syntaxe:

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

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

write()

Nastavuje pozici serva v úhlových stupních a volitelně může nastavit rychlost pohybu a blokovat opětovné volání této funkce, pokud ještě není dokončen předchozí pohyb serva.

Syntaxe:

```
sevo1.write(value, [speed], [wait]);
```

Parametry:

value (*uint16_t*) – pozice serva.
Při zadání parametru **value** mimo rozsah 0 až 180 nebude hodnota brána jako úhel, ale jako délka pulzu v mikrosekundách.
speed (*uint8_t*) – [*nepovinné*] rychlost pohybu serva,
0 – pohyb plnou rychlostí
1 až 255 – rychlost pohybu od nejpomalejší k nejrychlejší.
wait (*bool*) – [*nepovinné*]
TRUE blokuje opětovné volání funkce až do doby, než je dokončen předchozí pohyb serva.
FALSE opětovné volání povoluje.

Návratová hodnota:

Funkce nic nevrací.

writeMicroseconds()

Nastavuje délku řídícího pulzu v mikrosekundách

Syntaxe:

```
servo1.writeMicroseconds(value);
```

Parametr:

value (*uint16_t*) – délka řídícího pulzu v mikrosekundách.

Návratová hodnota:

Funkce nic nevrací.

slowmove()

Nastavuje pozici serva ve stupních a rychlost jeho pohybu. Funkce je shodná s funkcí `write()` *a je v knihovně implementována pouze z důvodu zpětné kompatibility.*

Syntaxe:

```
servo1.slowmove(value, speed);
```

Parametry:

value (*uint16_t*) – pozice serva.
Při zadání parametru value mimo rozsah 0 až 180 nebude hodnota brána jako úhel, ale jako délka pulzu v mikrosekundách.
speed (*uint8_t*) – [*nepovinné*] rychlost pohybu serva
0 – pohyb plnou rychlostí
1 až 255 – rychlost pohybu od nejpomalejší k nejrychlejší.

Návratová hodnota:

Funkce nic nevrací.

wait()

Vyčká na dokončení probíhajícího pohybu

Syntaxe:

```
servo1.wait();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

stop()

Okamžitě zastaví pohyb určeného serva.

Syntaxe:

```
servo1.stop();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

sequencePlay()

Přehrává zvolenou sekvenci pohybů s možností volby výchozí pozice a jednoduchého nebo vícenásobného přehrávání sekvence pohybů.

Syntaxe:

```
servo1.sequencePlay(sequence, sequencePositions, [loop],
[startPosition]);
```

Parametry:

sequence (*uint8_t*) – sekvence pohybů
sequencePositions (*uint8_t*) – číslo určuje, od které pozice přehrávání sekvence začne.
loop (*bool*) – [*nepovinné*]
TRUE – sekvence se bude přehrávat stále dokola
FALSE – sekvence se přehraje jen jednou
startPosition (*uint8_t*) – [*nepovinné, výchozí hodnota je 0*]

Návratová hodnota:

Funkce nic nevrací.

sequenceStop()

Okamžitě zastaví sekvenci pohybů.

Syntaxe:

```
servo1.sequenceStop();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Funkce nic nevrací.

Dotazy

isMoving()

Dotaz na pohyb serva.

Syntaxe:

```
servo1.isMoving();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

TRUE – pokud se zvolené servo stále pohybuje
FALSE – v opačném případě

Příklad:

```
// vytvoří proměnnou, do které se uloží návratová hodnota funkce
bool jeServo;

jeServo = servo1.isMoving();
```

readMicroseconds()

Dotaz na poslední použitou hodnota řídícího pulzu.

Syntaxe:

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

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Poslední použitá délka řídícího pulzu jako čas v mikrosekundách.

Příklad:

```
// vytvoří proměnnou, do které se uloží návratová hodnota funkce
uint16_t pozice;
```

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

attached()

Kontrolujte, zda je instance třídy VarSpeedServo propojena s fyzickým pinem.

Syntaxe:

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

Parametr:

pin (*uint8_t*) – číslo pinu, na který je připojeno zvolené servo

Návratová hodnota:

TRUE – pokud je servo k pinu připojeno
FALSE – v opačném případě

Příklad:

```
// vytvoří proměnnou, do které se uloží návratová hodnota funkce
bool jeServo;

jeServo = servo1.attached(pin);
```

read()

Dotaz na poslední použitou hodnotu řídícího pulzu.

Syntaxe:

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

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Poslední použitá hodnotu řídícího pulzu jako číslo v rozsahu 0 až 180 stupňů.

Příklad:

```
// vytvoří proměnnou, do které se uloží návratová hodnota funkce
uint8_t pozice;

pozice = servo1.read();
```

Příklady

Čekání na dokončení pohybu

Ačkoli dva rozdílné polohovací příkazy následují v programu ihned za sebou, servo nejprve dokončí první a teprve potom započne druhý.

```
#include <VarSpeedServo.h>

// vytvoření objektu servo1
VarSpeedServo servo1;

void setup()
{
  // servo1 bude ovládáno signálem na pinu 9
  servo1.attach(9);
}

void loop()
{
  // otoč se na pozici 180 stupňů rychlostí 30 a vyčkej, dokud servo nedokončí pohyb
  servo1.write(180, 30, true);
  // otoč se na pozici 0 stupňů rychlostí 30 a vyčkej, dokud servo nedokončí pohyb
  servo1.write(0, 30, true);
}
```

Různé rychlosti

Obě serva se pohybují společně různou rychlostí a to rychlejší vyčká po dokončení pohybu na dokončení pohybu toho pomalejšího. Teprve pak bude vykonán další příkaz.

```
#include <VarSpeedServo.h>

// vytvoření objektů servo2 a myservo2
VarSpeedServo servo2;
VarSpeedServo myservo2;

void setup()
{
  servo2.attach(9);
  myservo2.attach(8);
}

void loop()
```

```
{
  int LEF = 0;
  int RIG = 180;

  int SPEED1 = 160;
  int SPEED2 = 100;

  servo2.write(LEF, SPEED1);
  myservo2.write(LEF, SPEED2);
  servo2.wait();  // čeká se, dokud první servo nedokončí pohyb
  myservo2.wait();  // čeká se, dokud druhé servo nedokončí pohyb

  servo2.write(RIG, SPEED1);
  servo2.wait();  // čeká se, dokud první servo nedokončí pohyb

  servo2.write(LEF, SPEED1);
  myservo2.write(RIG, SPEED2);
  servo2.wait();  // čeká se, dokud první servo nedokončí pohyb
  myservo2.wait();  // čeká se, dokud druhé servo nedokončí pohyb

  servo2.write(RIG, SPEED1);
  servo2.wait();  // čeká se, dokud první servo nedokončí pohyb

  delay(1000);
}
```

Ruční řízení

Řízení pozice serva potenciometrem (servotester)

```
#include <VarSpeedServo.h>

// vytvoří objekt servo1
VarSpeedServo servo1;

// na tento pin bude připojen běžec potenciometru
const int potPin = 0;

// na tomto pinu bude generován řídící signál pro servo
const int servoPin = 9;

// proměnná, do které se bude ukládat aktuální analogová hodnota na pinu 0
int val;

void setup()
{
  // připojte servo na pin 9
```

```
  servo1.attach(servoPin);
}

void loop()
{
  // přečte aktuální hodnotu na pinu 0 (může být v rozsahu 0 až 1023)
  val = analogRead(potPin);

  // změna hodnoty z 0 až 1023 na 0 až 180)
  val = map(val, 0, 1023, 0, 180);

  // nastaví pozici serva podle momentální hodnoty proměnné val
  servo1.write(val);

  // malá prodleva před dalším změnou pozice
  delay(15);
}
```

Sekvence

Různé sekvence pohybů jsou přehrávány podle okamžité hodnoty na analogovém vstupu.

```
#include <VarSpeedServo.h>

VarSpeedServo servo2;

const int servoPin1 = 9;  // na tento pin je připojeno servo

// sekvence jsou definovány jako pole jednotlivých pozic
// každá položka obsahuje pozici v rozsahu 0 do 180 a rychlost pohybu

servoSequencePoint slow[] = {{100,20},{20,20},{60,50}};
// otoč se na pozici 100 rychostí 20, na pozici 20 rychlostí 20 a na pozici 60 rychlostí
50

servoSequencePoint twitchy[] = {{0,255},{180,40},{90,127},{120,60}};

const int analogPin = A0;

void setup()
{
  servo2.attach(servoPin1);
}

void loop()
{
```

```
  // přečti analogovou hodnotu na pinu 0:
  int sensorValue = analogRead(analogPin);
  if (sensorValue > 200)
  {
     servo2.sequencePlay(slow, 3);
     // přehraj sekvenci "slow" která má 3 položky,
     // a vrať se na první pozici
  }
  else
  {
     servo2.sequencePlay(twitchy, 4, true, 2);
     přehraj sekvenci "twitchy" a vrať se na třetí pozici
  }
  // malá prodleva před dalším změnou pozice
  delay(2);
}
```

Cik-cak

Otáčí servem od jednoho konce dráhy pohybu k druhému a zpět s čekáním na dokončení předchozího pohybu.

```
#include <VarSpeedServo.h>

VarSpeedServo servo1;
const int servoPin = 9;

void setup()
{
  servo1.attach(servoPin);
  servo1.write(0,255,true);
  // nastaví výchozí pozici serva
}

void loop()
{
  servo1.write(180,255,true);
  // otočí servem do pozice 180 maximální rychlostí, čeká na dokončení pohybu

  servo1.write(0,30,true);
  // otočí servem pomalu do pozice 0, čeká na dokončení pohybu
}
```

Cik-cak se dvěma servy

Otáčí oběma servy od jednoho konce dráhy pohybu k druhému a zpět a čeká na dokončení předchozích pohybů.

```
#include <VarSpeedServo.h>

VarSpeedServo servo2;
VarSpeedServo myservo2;

const int servoPin1 = 9;
const int servoPin2 = 10;

void setup()
{
  servo2.attach(servoPin1);
  servo2.write(0,255,false);
  // plnou rychlostí do pozice 0, na dokončení pohybu se nečeká

  myservo2.attach(servoPin2);
  myservo2.write(0,255,true);
  // plnou rychlostí do pozice 0, čeká se na dokončení pohybu
}

void loop()
{
  servo2.write(180,127,false);
  // servo2 do pozice 180 poloviční rychlostí, nečeká se na dokončení pohybu

  myservo2.write(180,127,true);
  // myservo2 do pozice 180 poloviční rychlostí, čeká se na dokončení pohybu

  servo2.write(0,30,false);
  // servo2 do pozice 0 pomalu, nečeká se na dokončení pohybu

  myservo2.write(0,30,true);
  // myservo2 do pozice 0 pomalu, čeká se na dokončení pohybu
}
```