This document has only been translated. I am not the author and all copyrights belong to the author, which is Doctek. I would like to thank him for his efforts in creating this missing manual.

Original available from: https://hackaday.io/project/183279-accelstepper-the-missing-manual

Translator: TOMZORD

Chybějící manuál k Accel stepper library

MISSING MANUAL

DOCTEK

Chybějící manuál

Přeloženo z Hackaday.io

Autor: DOCKTEK

Překlad: TOMZORD

Software pro překlad: Deepl translator: https://www.deepl.com/translator

Popis:

Když jsem zjistil, že opět potřebuji použít knihovnu AccelStepper, a po několika hledáních jsem opět našel jen málo dokumentace, rozhodl jsem se napsat tento návod jak pro sebe, tak pro ostatní. Tím nechci říct, že dokumentace neexistuje, jen je generována z komentářů ve zdrojovém kódu knihovny v C++ pomocí Doxygenu, linuxového generátoru dokumentace. To sice vytváří užitečnou dokumentaci pro vývojáře, ale není uspořádána tak, aby pomohla uživatelům Arduina. Tato příručka uspořádá informace z Doxygenu použitelnějším způsobem, přidá podrobnosti, které jsem se naučil experimentováním, a zahrne užitečné příklady.

Podrobnosti:

Krokové motory si zaslouží, aby se v našich projektech používaly mnohem více, než se používají. Díky jejich použití v 3D tiskárnách se ceny motorů a ovladačů výrazně snížily. Knihovny pro jejich pohon však zaostávají. Knihovna Stepper, která je součástí Arduina, je velmi omezená, protože blokuje a umožňuje pohyb pouze jednoho motoru najednou. Knihovna AccelStepper, kterou vytvořil Mike McCauley a kterou lze snadno nainstalovat pomocí správce knihoven Arduino, tato omezení překonává, ale je považována za obtížně použitelnou. Účelem této příručky je vysvětlit knihovnu AccelStepper, aby se její použití mohlo stát mnohem rozšířenějším.

Používání knihovny Accel Stepper Library přehled:

Knihovna AccelStepper je knihovna pro Arduino napsaná v jazyce C++. Chcete-li ji používat, zkonstruujte (nebo v řeči C++ "instanciujte") pojmenované objekty typu AccelStepper. Tyto softwarové objekty mají obvykle v názvu slovo "motor" nebo "stepper" a jsou přímo spojeny s fyzickými krokovými motory a jejich rozhraním ("ovladači"). Podporováno je několik různých rozhraní. Při konstrukci objektu je třeba určit vhodné rozhraní pro konkrétní aplikaci. Jakmile je objekt zkonstruován, lze k jeho ovládání použít různé funkce (nazývané "členské funkce"), které jsou v knihovně k dispozici.

Zjistil jsem, že pomáhá rozdělit je do skupin. První skupinou jsou nastavovací funkce, které určují fyzikální schopnosti (například rychlost a zrychlení) motoru a fyzikální hodnoty (například polohu, do které se má pohybovat). Další skupinou jsou funkce, které skutečně pohybují motorem tím, že mu posílají signály, které způsobí kroky. Tyto funkce používají k řízení pohybu hodnoty z funkcí nastavení. Další skupinou jsou funkce, které hlásí informace o stavu pohybu. Pak následují funkce správy pinů, které konfigurují a ovládají piny, které komunikují s ovladačem. Tyto funkce budou popsány níže, ale nejprve je dobré pochopit, jak AccelStepper funguje.

Přehled pohybů:

Krokový motor se pohybuje, když obdrží elektrický signál, který způsobí, že motor udělá krok. Knihovna AccelStepper má pouze jednu volatelnou funkci, která způsobí pohyb: runSpeed(). Ostatní funkce, které vedou k pohybu, volají runSpeed(), aby vytvořily skutečné kroky. Každá taková funkce má ve svém názvu slovo "run". Funkce pro vyvolání pohybu jsou dvojího typu: konstantní rychlost (omezená aktuální hodnotou rychlosti) a proměnná rychlost (omezená nastavením zrychlení a maxSpeed a polohou vzhledem k cíli). Každý z těchto typů obsahuje blokovací a neblokující funkce. Blokovací funkce budou probíhat, dokud nenastane podmínka zastavení, ale dokud nebudou dokončeny, nebude probíhat žádný jiný kód. Neblokovací funkce způsobí jeden krok (pokud je krok nutný - viz níže) a v každém případě se okamžitě vrátí. Protože každé volání neblokující funkce způsobí nejvýše jeden krok, je třeba je volat co nejčastěji; obvykle v hlavní smyčce. Jinak motor nebude krokovat požadovanou rychlostí.

Funkce runSpeed() určuje, kdy má být proveden krok. runSpeed() odečte čas posledního kroku od aktuálního času. Pokud je výsledek větší nebo roven hodnotě stepInterval (říkáme: "Je třeba udělat krok."), runSpeed zvýší (nebo sníží, podle potřeby) currentPosition, zavolá step() a aktualizuje čas, kdy byl proveden poslední krok. (step() je interní funkce - není přímo volatelná. Způsobí, že se na rozhraní pošle správný elektrický signál.) Každé volání funkce runSpeed() tento proces opakuje, takže funkce runSpeed() musí být volána alespoň rychlostí za sekundu. Obvykle to znamená umístit volání runSpeed() do funkce loop(). Volání runSpeed() v době, kdy není třeba provést žádný krok, nic neudělá

Řídicí proměnnou pro runSpeed() je stepInterval, takže musíme pochopit, jak ji AccelStepper počítá. Interní proměnná stepInterval není přímo manipulovatelná uživatelem. V nejjednodušším případě vyvolání funkce setSpeed() způsobí výpočet nové hodnoty rychlosti - a stepIntervalu. Hodnota rychlosti bude omezena na +/- maxSpeed. Tuto hodnotu lze změnit pomocí funkce setMaxSpeed(). Rychlost a směr se sledují odděleně, takže stepInterval je vždy kladný; směr závisí na znaménku rychlosti (v setSpeed()) nebo na směru aktuálníPozice vzhledem k cílovéPozici.

"Dobře, fajn," řeknete. "Dostanu konstantní rychlost, ale jak motor zrychluje nebo běží do polohy?" "Ano," odpovím. Dobrá otázka! K obojímu slouží funkce run(). Funkce run() nejprve zavolá funkci runSpeed(), aby způsobila krok (pokud je to nutné, jak je definováno výše) při aktuální rychlosti a směru. Poté zavolá funkci computeNewSpeed(). Funkce computeNewSpeed() vypočítá novou rychlost (stepInterval) a nastaví ji jako aktuální rychlost.

Podrobněji řečeno, funkce computeNewSpeed() vynutí výpočet nové okamžité rychlosti (stepInterval) a nastaví ji jako aktuální rychlost. Je volána knihovnou:

- po každém kroku, pokud se používá run() nebo runToPosition() nebo runToNewPosition().

- po změně rychlosti maxSpeed pomocí funkce setMaxSpeed().

- po změně zrychlení pomocí setAcceleration().

- po změně cílové polohy (relativní nebo absolutní) - prostřednictvím move() nebo moveTo().

- po volání funkce stop() - prostřednictvím move().

Zvyšuje (nebo snižuje) rychlost na základě polohy (vzdálenost od cíle určuje, zda je čas zpomalit nebo zrychlit), zrychlení (vyšší nebo nižší) a maxSpeed (bylo dosaženo?). Všimněte si, že hodnoty rychlosti nastavené funkcí setSpeed() jsou ignorovány a tato funkce je skutečně přepíše. Tuto funkci není třeba volat přímo a nemělo by se to ani dělat.

Mezi další funkce, které změní krokInterval, patří setCurrentPosition(), která nastaví rychlost a krokInterval na 0. Také kdykoli je zavolána funkce setSpeed(), vypočítá nový krokInterval na základě zadané rychlosti. Hodnota bude omezena hodnotou maxSpeed. Jak bylo uvedeno výše, tato hodnota bude použita pouze v případě použití funkce runSpeed(); v případě použití funkce run() bude ignorována a přepsána. Konečně konstruktor také nastaví hodnotu stepInterval na 0, takže nedojde k žádnému pohybu, dokud nebude zavolána funkce, která ji nastaví. Obvykle se volá funkce setMaxSpeed(), po níž následuje buď funkce setSpeed(), nebo moveTo().

Mezi další funkce, které způsobí pohyb, patří runSpeedToPosition(), která kontroluje, zda bylo dosaženo cílové pozice, a pokud ne, zavolá runSpeed(). Další je runToPosition(), která jednoduše volá run(), dokud není dosaženo cílové pozice - to znamená, že blokuje, dokud není dosaženo pozice. A konečně runToNewPosition() umožňuje zadat novou pozici, pak zavolá moveTo(), aby ji nastavil jako cílovou, a zavolá runToPosition(), aby provedl pohyb.

Shrňme si, jak AccelStepper funguje. Při vytvoření objektu AccelStepper jsou maxSpeed a zrychlení nastaveny na 1,0. CurrentPosition a TargetPosition jsou nastaveny na 0 a rychlost je nastavena na 0,0. V nejjednodušším případě se pro zahájení pohybu musí zavolat funkce setSpeed(), aby se nastavila rychlost pro následná volání funkce runSpeed(). Při použití výchozích hodnot bude rychlost omezena na 1,0 kroku za sekundu. Nyní volejte runSpeed() co nejčastěji - obvykle v hlavní smyčce. Motor bude nepřetržitě běžet konstantní rychlostí. Funkci setSpeed() lze volat se záporným číslem, aby došlo k pohybu v opačném směru (funkci setMaxSpeed() volat nemusíte, ale rychlost bude omezena na 1 krok za sekundu nebo méně).

Alternativně lze zavolat funkci moveTo() pro zadání nové cílové polohy. Volání funkce setSpeed() je zbytečné, protože rychlost bude vypočtena funkcí computeNewSpeed(), která je volána funkcí moveTo(). Volání run() místo runSpeed() způsobí, že motor zrychlí na maxSpeed a rozběhne se směrem k cílové poloze. Jak se motor blíží k cílové poloze, zpomaluje se a po dosažení cílové polohy se zastaví. Další volání funkce run() nezpůsobí žádný pohyb, dokud nebude provedeno nové volání funkce moveTo() nebo move(). Pokud je požadováno rychlejší zrychlování a zpomalování, pak je třeba použít funkci setAcceleration(). Při každém volání run() se hodnota rychlosti vypočítá pomocí computeNewSpeed() (interní rutina, která by neměla být volána přímo). Jakákoli hodnota nastavená pomocí setSpeed bude ignorována. Tyto funkce budou podrobně popsány níže.

S těmito znalostmi se nyní podíváme na detaily funkcí v AccelStepperu.

Tvorba objektu Accelstepper

Můžete mít více krokových ovladačů současně, které se budou pohybovat různými rychlostmi a zrychleními, pokud budete volat jejich funkce run() nebo runSpeed() v dostatečně častých intervalech. O těchto funkcích a jejich fungování bude pojednáno dále v části Pohybové funkce. Chcete-li zkonstruovat objekt AccelStepper, musíte znát typ použitého ovladače a čísla pinů, kterými je ovládán. Zde je prototyp konstruktoru:

AccelStepper(uint8_t interface = AccelStepper::FULL4WIRE, uint8_t pin1 = 2, uint8_t pin2 = 3, uint8_t pin3 = 4, uint8_t pin4 = 5, bool enable = true)

Klíčovou položkou je typ použitého rozhraní (ovladače). To bude záviset na typu krokového motoru, který používáte, a případně na režimu, ve kterém se nachází. Typ by měl být specifikován pomocí výrazů, které jsou definovány ve výčtu v souboru AccelStepper.h. Nejběžnějšími ovladači jsou krokový a směrový ovladač, například ovladač a4988; ovladač s dvěma H-můstky, například ovladač L298 nebo TB6612; nebo ovladač pro unipolární motory, například ULN2003. Ty se specifikují pomocí terminologie AccelStepper takto:

AccelStepper::DRIVER (Step-and-direction Driver, 2 driver pins required. Note that any use of microstepping doesn't matter here. The driver only cares about steps.)

AccelStepper::FULL4WIRE (4 wire full stepper, 4 motor pins required. For a Dual-H-Bridge controller or unipolar driver. This is the default.)

The following interfaces are ones I have no experience with, but are supported by the library.
AccelStepper::FULL2WIRE (2 wire stepper, 2 motor pins required.)

AccelStepper::FULL3WIRE (3 wire stepper, such as HDD spindle, 3 motor pins required.)

AccelStepper::HALF3WIRE (3 wire half stepper, such as HDD spindle, 3 motor pins required.)

AccelStepper::HALF4WIRE (4 wire half stepper, 4 motor pins required.)

The last interface is used primarily by Adafruit for their motor interface board. It is best documented by Adafruit and I will not discuss it further.

AccelStepper::FUNCTION (Use the functional interface, implementing your own driver functions, as Adafruit does.)

Toto používat pro driver DM860H:

//Example using a driver that accepts step/direction input. An enable pin is not used so enable = true (the default) need not be specified.

// Define pin connections

const int dirPin = 4; //směrový pin

const int stepPin = 5;//signálový pin

// Creates an instance

AccelStepper myStepper(AccelStepper::DRIVER, stepPin, dirPin);

Funkce nastavení

Tyto funkce nastavují hodnoty pro pozdější použití pohybovými funkcemi. Měly by být volány nejprve pro nastavení požadovaných podmínek, poté mohou být volány znovu pro jejich změnu. Kladné a záporné hodnoty pro směr jsou libovolné v závislosti na způsobu zapojení ovladače. Klíčové je, že pohybují motorem ve dvou opačných směrech. Na kladných a záporných hodnotách pro rychlost nezáleží, rychlost je uložena jako absolutní hodnota.

moveTo((long) absolute_position)

Argument: absolute_position v krocích. Požadovaná absolutní pozice typu long. Může být kladná nebo záporná.

Nastavení cílové pozice. Funkce run() se pokusí posunout motor (maximálně o jeden krok na jedno volání) z aktuální polohy do cílové polohy nastavené posledním voláním této funkce. Pozor: funkce moveTo() také přepočítá rychlost pro další krok. Pokud se snažíte používat pohyby konstantní rychlostí, měli byste po volání moveTo() zavolat funkci setSpeed(). Pokud je funkce moveTo() zavolána během pohybu motoru,cílová poloha se okamžitě změní a pro výpočet nové rychlosti se použije algoritmus zrychlení. Pokud motor běží vysokou rychlostí v určitém směru a nová cílová poloha je v opačném směru, motor bude pokračovat v běhu stejným směrem, zpomalí až zastaví a pak zrychlí v novém směru, dokud se nepřiblíží k novému cíli a nezpomalí až na doraz.

Příklad pohybu do absolutní polohy 2038. Nová hodnota targetPosition je po zavolání této funkce 2038.

```
myStepper.moveTo(2038);
```

move((long) relative_movement)

Argument: relative_movement v krocích. Požadovaný pohyb vzhledem k aktuální pozici. Typ argumentu je long a může být kladný nebo záporný.

Nastavení cílové polohy vzhledem k aktuální poloze. Upozornění: funkce move() přepočítává také rychlost pro další krok. Pokud se snažíte používat pohyby konstantní rychlostí, měli byste volat setSpeed() až po volání move(). Pokud je funkce move() volána během pohybu motoru, výsledek je stejný jako u funkce moveTo(). Jediný rozdíl je ve způsobu výpočtu nové cílové polohy.

Příklad pro posun o jeden krok v kladném směru vzhledem k aktuální poloze.

```
myStepper.move(1);
```

setMaxSpeed((float) speed)

Argument: rychlost v krocích za sekundu. Požadovaná maximální rychlost jako hodnota v plovoucí desetinné čárce. Lze předat zápornou hodnotu, ale bude uložena jako absolutní hodnota.

Nastavení maximální přípustné rychlosti. Tato funkce se obvykle volá jako první při použití objektu AccelStepper (tj. krokového motoru). Pokud se použije funkce run(), pak se motor zrychlí na tuto rychlost. Jakákoli hodnota rychlosti nastavená funkcí setSpeed() bude ignorována. Pokud je použita funkce runSpeed(), pak musí být funkce setSpeed() zavolána až po funkci setMaxSpeed(). Pozor: Maximální dosažitelná rychlost závisí na procesoru a taktovací frekvenci. Hodnoty až 4000,0 mohou fungovat - ale 1000,0 je sázka na jistotu. Rychlosti, které překračují maximální rychlost podporovanou procesorem, mohou vést k nelineárnímu zrychlování a zpomalování. Výchozí hodnota maxSpeed je 1,0 kroku za sekundu.

Příklad nastavení maximální rychlosti na 1000,0 kroků za sekundu.

```
myStepper.setMaxSpeed(1000.0);
```

setAcceleration((float) acceleration)

Argument: zrychlení v krocích za sekundu za sekundu jako hodnota s pohyblivou desetinnou čárkou. Může být zadáno jako záporné, ale uloží se pouze absolutní hodnota.

Nastavuje míru zrychlení/zpomalení. Zrychlení se používá funkcí run() ke zvýšení (nebo snížení) rychlosti, kterou motor krokuje. Jedná se o nákladné volání, protože vyžaduje výpočet odmocniny. Nevolejte jej častěji, než je nutné. Výchozí hodnota je 1,0.

Příklad nastavení zrychlení na 50,0 kroků za sekundu za sekundu.

```
myStepper.setAcceleration(50.0);
```

Možná se zeptáte: "Nechápu zrychlení! Co přesně znamená krok za sekundu za sekundu?". Vysvětlím vám to.

Začněte s rychlostí maxSpeed nastavenou na 200,0 kroků za sekundu a zrychlením nastaveným na 50,0 kroků za sekundu za sekundu. Nechte náš program opakovaně volat funkci run(). Funkce run() začne pohybovat motorem rychlostí 0,0 kroku za sekundu a každou sekundu běhu motoru bude tuto rychlost zvyšovat o 50 kroků za sekundu. Uvažujme rychlost a polohu na konci jedné sekundy. Motor začne pracovat rychlostí 0,0 kroku za sekundu a za jednu sekundu se zrychlí o 50 kroků za sekundu. Pohyboval se asi o 25 kroků (přibližně průměrná rychlost za první sekundu; ~1/2 * 50). Zrychlení je konstantní, takže zde jsou hodnoty rychlosti a polohy (uražené vzdálenosti) v průběhu času. Toto jsou skutečné výsledky ze souboru UnoAccelStepperRunDemo.ino. Protože každé volání funkce run() způsobí aktualizaci, jsou hodnoty hlášené každou sekundu poněkud jiné, než byste očekávali. S tímto náčrtem můžete experimentovat, abyste se o zrychlení dozvěděli více. Pro zjištění výsledků ani nemusíte připojovat motor nebo ovladač, ale je to větší zábava, když to uděláte!

Second:	1	2	3	4	5	6
Speed:	53	103	153	200	200	200
Distance:	28	106	235	413	613	813

Co když je zrychlení zlomkové? Funguje to dobře, jen zrychlení je pomalé. Byla testována hodnota zrychlení 1e-14. Algoritmus stále funguje, ale po velmi dlouhou dobu nedojde k žádnému pohybu. (Už mě to čekání nebaví!)

setSpeed((float) speed)

Argument: rychlost v krocích za sekundu jako hodnota s plovoucí desetinnou čárkou. Může být kladná nebo záporná.

Nastaví požadovanou konstantní rychlost pro použití s funkcí runSpeed(). Rychlost bude omezena aktuální hodnotou funkce setMaxSpeed() na +/- maxSpeed. Tato rychlost se bude používat tak dlouho, dokud bude volána funkce runSpeed() a výsledkem bude provoz s konstantní rychlostí. Pokud je zavolána funkce run(), bude tato hodnota ignorována (a přepsána). Rychlosti vyšší než 1000 kroků za sekundu mohou být nespolehlivé. Mohou být nastaveny velmi nízké rychlosti (např. 0,00027777 pro jednou za hodinu, přibližně. Přesnost rychlosti závisí na krystalu Arduina. Jitter závisí na tom, jak často voláte funkci runSpeed().

Příklad nastavení rychlosti (pro provoz s konstantní rychlostí) na 200,0 kroků za sekundu.

```
myStepper.setSpeed(200.0);
```

setCurrentPosition((long) position)

Argument: Požadovaná hodnota polohy v krocích od místa, kde se motor právě nachází. Může být kladná nebo záporná a je typu long.

Tato funkce zajistí, aby se aktuální poloha motoru a cílová poloha rovnaly zadané hodnotě. Například pokud jste motor přesunuli do počáteční polohy (řekněme 213) a zavoláte setCurrentPosition(100), aktuální poloha i cílová poloha budou nastaveny na hodnotu 100. K dosažení polohy 200 bude potřeba 100 kladných kroků (pokud zavoláte moveTo(200)), nikoli 13 kroků v záporném směru. Tato funkce také vynuluje hodnotu rychlosti na 0,0. Tato funkce je nejužitečnější pro nastavení nulové polohy na krokovacím zařízení po počátečním hardwarovém polohovacím pohybu.

Příklad nastavení aktuální polohy na 0 kroků.

```
setCurrentPosition(0);
```

Upozornění: Tuto funkci nevolejte, pokud probíhá pohyb! Hodnoty aktuální polohy a cílové polohy se okamžitě změní a rychlost se okamžitě nastaví na 0. To donutí motor k pokusu o okamžité zastavení a s největší pravděpodobností povede k vynechání kroků a možnému poškození systému. Před voláním funkce setCurrentPostion() se ujistěte, že se krokový ovladač zastavil.

stop()

Tato funkce nastaví novou cílovou polohu, která způsobí co nejrychlejší zastavení krokového ovladače s využitím aktuálních parametrů rychlosti a zrychlení. Volání funkce runToPosition() přivede motor k zastavení. Opakovaná volání funkce run() provedou totéž. Motor udělá více kroků ve směru, kterým se pohybuje, než se zastaví. To může být několik kroků, pokud motor běží rychle a zrychlení je malé.

Pohybové funkce

Úplné vysvětlení, jak AccelStepper používá tyto funkce k řízení pohybu, je uvedeno výše v části Přehled pohybu. V každém případě si najděte čas na prostudování této části, pokud potřebujete nahlédnout do fungování těchto pohybových funkcí. Tyto funkce způsobují skutečný chod motorů - konfigurační a nastavovací funkce by měly být volány před voláním pohybových funkcí.

Nejprve zvažte neblokující pohybové funkce.

runSpeed()

Pokud je třeba udělat krok (jak je popsáno výše), posune se o jeden krok a zavede konstantní rychlost nastavenou posledním voláním příkazu setSpeed(). Tuto funkci musíte volat často, nejméně však jednou za interval kroků. Vytváří provoz s konstantní rychlostí. Protože nehledá polohu, lze ji použít k nepřetržitému chodu motoru. Všimněte si, že funkce runSpeed() inkrementuje nebo dekrementuje hodnotu currentPosition, ale cílovou polohu ignoruje.

Funkce vrátí true, pokud byl motor krokovaný, a false, pokud ne. Všimněte si, že to není stejné jako chování funkce run()! Funkce runSpeed() může být volána několikrát, než skutečně způsobí krok. Vrátí false pokaždé, když krok neudělá; true se vrátí pouze tehdy, když krok udělá. Na druhou stranu funkce run() může být také volána mnohokrát, než způsobí krok, ale pokaždé vrátí true, dokud se motor nezastaví.

Pozor: Protože funkce runSpeed() nepoužívá zrychlení, může nastavení vysoké hodnoty rychlosti a následné volání funkce runSpeed() způsobit zastavení motoru. Chcete-li dosáhnout maximální rychlosti motoru, použijte funkci run() a nechte motor zrychlit.

Příklady použití funkce runSpeed(): UnoAccelStepper_ConstantSpeed.ino (nejjednodušší verze), UnoAccelStepperExper_1 (jednoduchá, ale hlásí výsledky každou sekundu) a také UnoAccelStepper_speedControl.ino, který používá potenciometr pro změnu rychlosti a hlásí výsledky každou sekundu. Mezi experimenty, které můžete vyzkoušet, patří nastavení maximálních otáček a rychlosti na velmi vysoké hodnoty a zjištění, zda se motor rozběhne, nebo se zastaví. Poté zkuste totéž s funkcí speedControl, aby se rychlost pomalu zvyšovala, a zjistěte, zda je možné dosáhnout vysokých otáček.

run()

Funkce run() zavolá funkci runSpeed() a poté zavolá funkci computeNewSpeed() (viz popis v části Přehled pohybu výše). Krok se provede, pokud je třeba a aktuálníPoloha se nerovná cílovéPoloze. Tato funkce implementuje zrychlení a zrychlení pro pohyb motoru. Musíte ji volat co nejčastěji, nejméně však jednou za minimální časový interval kroku,nejlépe v hlavní smyčce. Funkce vrátí true, pokud rychlost není nulová nebo nebylo dosaženo konečné polohy. (To znamená, že motor je krokovaný.) Pokud jsou tyto podmínky splněny, bude vrácena hodnota true bez ohledu na to, zda motor krokoval nebo ne. Všimněte si, že každé volání funkce run() provede maximálně jeden krok, a to pouze tehdy, když má dojít ke kroku.

Příklad použití run(): UnoAccelStepperRunSimple.ino- Jednoduchá ukázka funkce run(). Další skica, UnoAccelStepperRunDemo.ino, je také pěkná, čistá ukázka run(), která každou sekundu podává hlášení na sériový monitor - pomáhá vytvořit si představu o tom, jak AccelStepper funguje. Vyzkoušejte různé maximální rychlosti a zrychlení a podívejte se, co hlásí.

runSpeedToPosition()

Provede funkci runSpeed(), pokud není dosaženo cílové polohy. Tuto funkci je třeba volat často stejně jako runSpeed() nebo run(). Spustí motor aktuálně zvolenou rychlostí, pokud není dosaženo cílové polohy. Neimplementuje zrychlení.

Funkce vrátí true, pokud krokování proběhlo, a false v opačném případě. Pokud chcete zjistit, zda bylo dosaženo cílové polohy, zavolejte níže popsanou funkci distanceToGo().

Příklad použití funkce runSpeedToPosition(): UnoAccelStepper_ProportionalControl.ino Všimněte si použití funkce setSpeed() po volání moveTo().

Nyní se podíváme na funkce blokování pohybu. "Blokování" znamená, že dokud se tyto funkce nedokončí, neproběhne žádný další programový kód.

runToPosition()

Tato funkce bude neustále volat funkci run(), čímž zablokuje ostatní příkazy, dokud nevrátí hodnotu false, což znamená, že bylo dosaženo požadované pozice. Pohybuje motorem (se zrychlením/zpomalením) do cílové polohy a blokuje jej, dokud není v poloze.

runToNewPosition((long) absolute position)

Provede moveTo(pozice) a poté runToPosition().

Přesune motor (se zrychlením/zpomalením) do nové cílové polohy a zablokuje jej, dokud není v poloze.

Argument: absolute_position v krocích. Požadovaná absolutní poloha typu long. Může být kladná nebo záporná.

Příklad blokování pohybu na 200.

```
runToNewPosition(200);
```

Shrnutí:

- Pro inicializaci použijte funkci setMaxSpeed(), poté funkci setSpeed() a poté opakovaně volejte funkci runSpeed(), aby se motor pohyboval konstantní rychlostí. Přestaňte volat runSpeed(), abyste motor zastavili.

- Použijte setMaxSpeed(), pak moveTo() nebo move() pro nastavení cíle, pak setSpeed() pro inicializaci, pak opakovaně volejte runSpeedToPosition() pro krokování motoru konstantní rychlostí, dokud není dosaženo cílové polohy.

- Použijte setMaxSpeed(), pak setAcceleration(), pak moveTo() nebo move() pro nastavení cíle, pak opakovaně volejte run() pro zrychlení motoru, pohyb k cíli a zpomalení až do zastavení.

Ve smyčce je třeba zavolat pouze jednu z funkcí run.

- Pokud je požadováno blokovací chování, použijte funkci setMaxSpeed(), pak setAcceleration(), pak moveTo() nebo move() pro nastavení cíle a pak zavolejte runToPosition(). Motor se bude chovat, jako by se opakovaně volala funkce run(), ale dokud nebude dosaženo cílové polohy, nebude možné provádět žádný další kód. Alternativně lze pomocí runToNewPosition() přeskočit volání move() nebo moveTo().

Je špatný nápad vkládat volání delay() do smyčky s run()! To způsobí pomalý běh smyčky a omezí rychlost motorů.

Jen pro úplnost: pokud je použita funkce run(), funkce setSpeed() rychlost nenastavuje! Rychlost nastaví pouze funkce computeNewSpeed(), jak je popsáno výše v části Přehled pohybu.

Funkce stop() motor nezastaví! Nastaví novou rychlost a cíl. Chcete-li zastavit, zavolejte funkci run() nebo runToPosition().

Funkce move() a moveTo() nezpůsobí pohyb. Pro vyvolání pohybu je třeba zavolat funkci run(). (Funkce runToPosition() to udělá také.)

Pokud se používá funkce runSpeed(), záleží pouze na hodnotě rychlosti. Hodnota rychlosti je poslední hodnota nastavená funkcí setSpeed() nebo vypočtená funkcí computeNewSpeed(), pokud byla použita funkce run().

Informativní funkce:

Ty mohou být volány pro načtení hodnot určitých proměnných nebo pro kontrolu stavu pohybu. Všimněte si, že rychlost je vždy v krocích za sekundu a poloha je v krocích od polohy 0. Směr je považován za pravý nebo levý, ale tyto hodnoty jsou libovolné v závislosti na tom, jak je motor zapojen do elektroniky rozhraní. Kladné hodnoty polohy jsou opačné než záporné hodnoty.

maxSpeed()

Maximální rychlost nastavená pomocí setMaxSpeed - pro případ, že jste zapomněli. Vrací hodnotu s pohyblivou řádovou čárkou.

speed()

Vrátí poslední rychlost jako hodnotu s plovoucí desetinnou čárkou v krocích za sekundu. Může to být rychlost nastavená funkcí setSpeed(), pokud byla volána pouze funkce runSpeed(), nebo rychlost vypočtená, pokud byla volána funkce run() nebo pokud byla volána polohovací rutina.

targetPosition()

Cílová pozice nastavená funkcí move() nebo moveTo(). Vrací hodnotu long.

currentPosition()

Kde se motor právě nachází. Vrací hodnotu long.

distanceToGo()

Rovná se targetPosition - currentPosition jako kladné celé číslo. Vrací hodnotu long. Vzdálenost z aktuální polohy do cílové polohy v krocích.

isRunning()

Vrací true, pokud rychlost není nulová a distanceToGo není 0.

Vzorové programy

Pokud jste tak ještě neučinili, je nyní vhodná doba, abyste se podívali na ukázkové programy, které jsem přiložil. Ty byly představeny výše, když se probíraly funkce runSpeed() a run(), ale chci se ujistit, že si jich všimnete. Můžete si je stáhnout jednotlivě a vybrat si jen ty, které vás zajímají, nebo si stáhnout archiv AccelStepperDemos.zip. Podporovány jsou tři typy ovladačů. Stačí odkomentovat ten, který chcete používat, a ostatní zakomentovat. Všechny byly testovány na Uno a Duemilanovi, ale měly by fungovat na jakékoli verzi Arduina podporující AccelStepper. Knihovna ElapsedMillis slouží ke snadnému tisku v sekundových intervalech. Lze ji stáhnout pomocí správce knihoven, nebo lze s trochou snahy použít také knihovnu millis().

UnoAccelStepper_ConstantSpeed - Používá funkci runSpeed() k řízení jednoho krokového zařízení.

UnoAccelStepperExper_1 - Jednoduchý příklad runSpeed() s jednosekundovým hlášením.

UnoAccelStepper_speedControl - Používá potenciometr k řízení rychlosti. Má hlášení. Používá runSpeed() - může provádět spoustu experimentů.

UnoAccelStepperRunSimple - Stepper poskakuje mezi mezními hodnotami. Nejlepší jednoduchá ukázka run().

UnoAccelStepperRunDemo - ukazuje, jak funguje run(). Také demonstruje účinky funkcí move a stop. Obsahuje hlášení každou sekundu. Funguje skvěle pro provádění mnoha experimentů - hlášení je velmi užitečné pro získání přehledu.

UnoAccelStepper_ProportionalControl - Používá vstup potenciometru. Má volitelnou rutinu pro zobrazení portu. Vypisuje informace o poloze. Ukazuje použití funkce runSpeedToPosition() a setSpeed() po funkci moveTo().