Knihovna Encoder

Knihovna Encoder je určena pro zpracování signálu z dvoukanálových kvadraturních enkodérů, které se používají v automatizační technice ke snímání pozice nebo rychlosti a směru pohybu objektů a ve spotřební elektronice k zadávání údajů do zařízení. Podrobněji je o principu a druzích enkodérů pojednáno v příloze.

Elektrické zapojení

Levné enkodéry ve spotřební elektronice spínají piny proti zemi. Knihovna Encoder proto aktivuje pull-up rezistory v mikrokontroléru, ale pokud je enkodér připojen delšími vodiči, je vhodné připojit ke vstupům mikrokontroléru ještě externí pull-up rezistory o hodnotě 1 až 10 kiloohmů.
Výstupem enkodéru jsou dva fázově posunuté signály (viz Příloha 1), z jejichž frekvence se odvozuje rychlost a z posloupnosti směr pohybu, enkodér je tedy nutno vždy připojit na dva piny mikrokontroléru.

Existují tři možnosti připojení:

- Oba výstupní signály enkodéru jsou připojeny na piny, které generují přerušení. Toto zapojení poskytuje nejlepší výsledky.
- Jeden z výstupních signálů enkodéru je připojen na pin, který generuje přerušení. Toto zapojení poskytuje vyhovující výsledky.
- Oba výstupní signály enkodéru jsou připojeny na běžné digitální vstupy. Takto zapojený enkodér může v případě nevhodně zvoleného algoritmu vracet chybné údaje (viz níže).

Board	Interrupt	LED Pin (ten nepoužívejte)
Arduino Due	Všechny digitální piny	13
Arduino Uno	2, 3	13
Arduino Leonardo	0, 1, 2, 3	13
Arduino Mega	2, 3, 18, 19, 20, 21	13
Teensy 4.0 - 4.1	Všechny digitální piny	13
Teensy 3.0 - 3.6	Všechny digitální piny	13
Teensy LC	2 až 12, 14, 15, 20 až 23	13
Teensy 2.0	5, 6, 7, 8	11
Teensy 1.0	0, 1, 2, 3, 4, 6, 7, 16	
Teensy++ 2.0	0, 1, 2, 3, 18, 19, 36, 37	6
Teensy++ 1.0	0, 1, 2, 3, 18, 19, 36, 37	

Pollovací metoda připojení enkodéru

Pokud enkodér připojíte na piny, které nevyvolávají přerušení, bude jeho stav čten pouze v okamžiku volání funkce `read()`. Pro správný výsledek tedy musíte funkci `read()` volat co nejčastěji.

Tato metoda vyhovuje (s jistými výhradami) pouze pro rotační enkodéry, určené pro ruční zadávání údajů. Pokud již musíte v hlavní smyčce použít funkci `Serial.print()`, pak nastavte co možná nejvyšší přenosovou rychlost, aby data byla odeslána v nejkratším možném čase.

Pro enkodéry pracující vysokou rychlostí, například ty, které snímají otáčky motoru, vždy použijte režim přerušení!

Třída Encoder

Konstruktor

Můžete vytvořit více instancí knihovny, kterým přidělíte vždy dva piny. Nejméně jeden z těchto pinů by měl mít funkci přerušení, ovšem připojení enkodéru na dva piny umožňující přerušení přinese nejlepší výsledky.
Knihovna sice dovolí připojit enkodér na dva běžné digitální piny, ale výsledky, které v takovém případě dostanete, mohou být poněkud neočekávané…

Encoder()

Vytvoří instanci knihovny Encoder a přidělí jí dva piny.

Syntaxe:

```
Encoder mujEnkoder(pin1, pin2);
```

Parametry:

mujEnkoder – jméno vytvořené instance

pin1 (*const int*) – číslo pinu, na který bude připojen výstup enkodéru

pin2 (*const int*) – číslo pinu, na který bude připojen výstup enkodéru

Návratová hodnota:

Funkce nic nevrací.

Členské funkce (metody)

read()

Vrací pozici enkodéru jako kladné nebo záporné číslo.

Syntaxe:

```
mujEnkoder.read();
```

Parametry:

Funkce nemá žádné parametry.

Návratová hodnota:

Současná pozice enkodéru (*long*).

write()

Nastavuje novou hodnotu pozice.

Syntaxe:

```
mujEnkoder.write(newPosition);
```

Parametr:

newPosition (*long*) – nová pozice

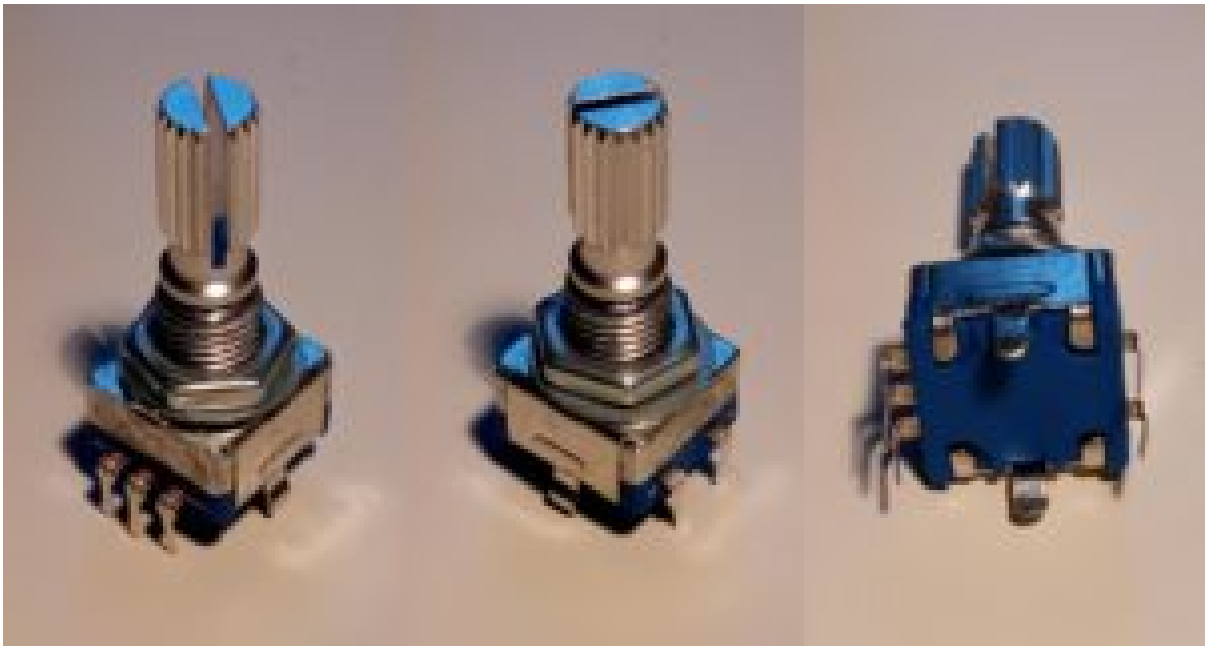
Návratová hodnota:

Funkce nic nevrací.

Příloha: Jak na enkodéry

RNDr. David Obdržálek, PhD.

V řadě přístrojů se k nastavování nějaké hodnoty používá „točítko“, například k regulaci hlasitosti muziky nebo nastavení času ohřevu na mikrovlnce. Dříve to bylo téměř výhradně pomocí potenciometru, dnes se stále častěji používají takzvané „enkodéry“. Jsou snadno dostupné a bývají často ve výukových sadách s Arduinem. Jak fungují a jak je použít ve vlastním projektu, si přečtete v tomto článku.



Uvnitř enkodéru[1] používaného jako výše zmíněné „točítko“ jsou dva spínače, které při otáčení osičky postupně spínají a rozpínají svůj kontakt. Když zatočíme takovýmto enkodérem, můžeme cítit a slyšet cvakání, které je způsobeno mechanickou konstrukcí – mechanické kontakty prostě přeskakují z jedné polohy do druhé[2].

Počítáním jednotlivých sepnutí můžeme určit, jak moc jsme enkodérem zatočili. Na to by stačil jeden kontakt, ale nevěděli bychom, kterým směrem se točilo. V enkodéru jsou ale spínače dva a díky tomu to už možné je. Kontakty uvnitř enkodéru jsou totiž vzájemně o něco posunuty, takže při otáčení nespínají a nerozepínají oba současně a to právě slouží k určení směru.

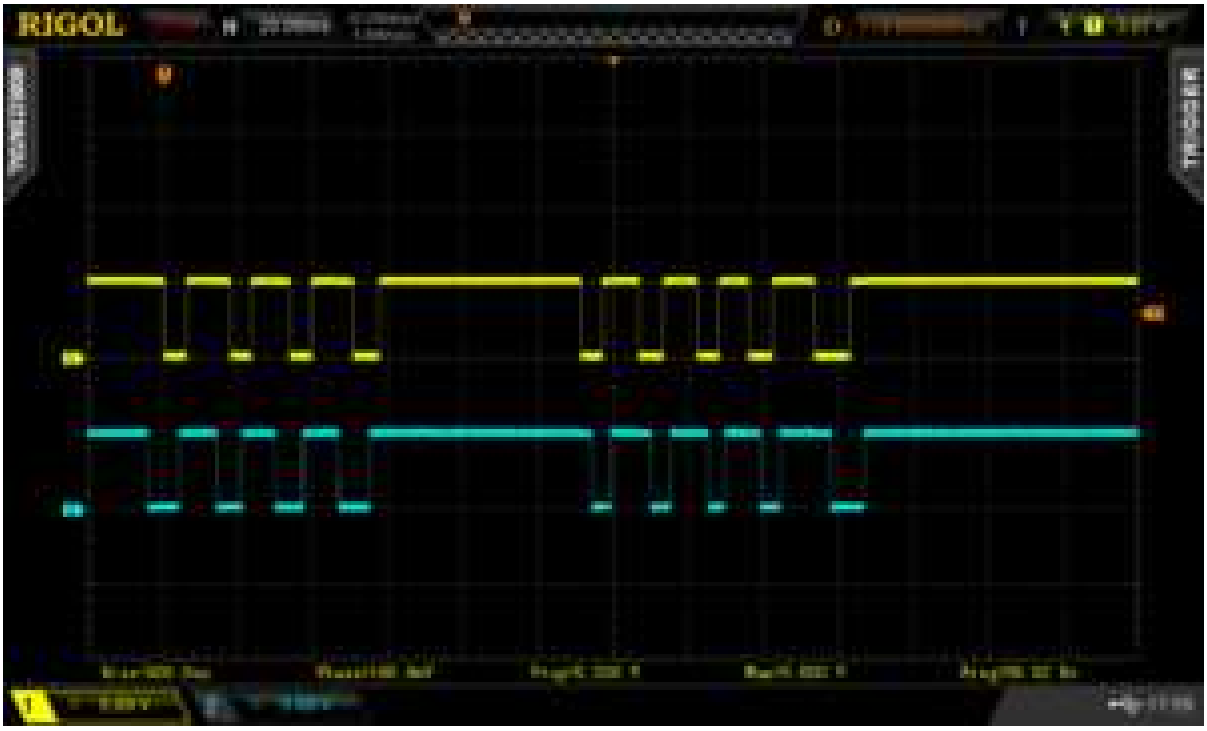
V klidovém stavu jsou oba výstupy řekněme ve stavu HIGH. Při každém cvaknutí jedním nebo druhým směrem přeskočí oba výstupy z HIGH do LOW a zpátky[3], akorát při otočení jedním směrem to dřív udělá kanál A než B a při otočení druhým směrem dřív B než A. Pro náš

[1] *Z anglického „encoder“; nemáme pro něj obrozenecky české slovo, ale potenciometr taky není české slovo a už jsme si zvykli.*

[2] *Celkový počet poloh záleží na konstrukci, například ty, co mám zrovna na stole, mají oba 20 poloh na jednu plnou otočku.*

[3] *Takhle to je u mého enkodéru, u jiného to může být naopak, v klidu LOW a přeskakuje to na HIGH, ale to na principu nic nemění*

enkodér tedy stačí kontrolovat, zda při změně signálu A z LOW na HIGH je signál B LOW nebo HIGH a podle toho se rozhodnout o přičtení nebo odečtení jedničky k čítači[4]. A to je celé.



Na obrázku z osciloskopu vidíme průběh signálů. Zatočil jsem trochu tam a po krátké pauze zase trochu zpátky, takže vidíme chování při točení oběma směry. V levé části začínáme s oběma signály HIGH. Modrý kanál spadne na LOW a za chvilku žlutý. Pak se modrý vrátí na HIGH a za chvilku žlutý a to celé se opakuje čtyřikrát. Pak je chvilka pauza (netočil jsem) a pak jsem začal točit na druhou stranu. V pravé části obrázku tak vidíme, že nejprve spadnul na LOW žlutý signál, pak teprve modrý, pak se žlutý vrátil na HIGH a pak teprve modrý. To celé pětkrát. Různé délky signálu v polohách low a HIGH nehrají roli[5].

Také je dobré si všimnout, že k určení směru otáčení nestačí jeden kanál – pokud bychom jeden signál zakryli, nebyli bychom schopni určit, jestli jsem po pauze točil na stejnou nebo opačnou stranu, než předtím.

Ještě poznámka k otáčení: Sestava má přirozenou tendenci skákat se po jednotlivých krocích, do kterých sama docvakává. Pokud bychom si dali tu práci a při točení zastavili v mezipoloze, tj. neposkočili do další polohy a zůstali někde na půl cesty, pak by jeden nebo oba dva signály byly v úrovni LOW, dokud pohyb nedokončíme. Podle toho, jak daleko mezi dvěma stabilními polohami jsme zastavili, by se buď už změnil čítač (protože už nastala ta hledaná změna sledovaného kanálu), nebo ještě ne, každopádně po dokončení pohybu víme, že sledovaný kanál prošel obě hodnoty a tudíž čítač už určitě bude aktualizovaný.

Pokud bychom při točení nedocvakli do následující polohy, ale z mezipolohy bychom se vrátili zpět, tak výsledná hodnota čítače zůstane nezměněna. Je možné, že se dočasně změní na hodnotu odpovídající dokončenému kroku, ale při návratu zpět by se opět změnila zpět.

---

[4] Můžeme místo toho sledovat změnu z HIGH na LOW anebo prohodit signály A a B. Vyjde to nastejno, jen se tím obrátí směr počítání.

[5] Dalo by se z toho spočítat, jak rychle jsem točil a že jsem v závěru v točení poněkud polevil, ale pokud nás zajímá jen počet tiků, tak to není zajímavé

Dočasná změna hodnoty i její následný návrat je způsoben stejným důvodem, jako v předchozím odstavci podle toho, jestli už se sepnul spínač.

Způsob připojení

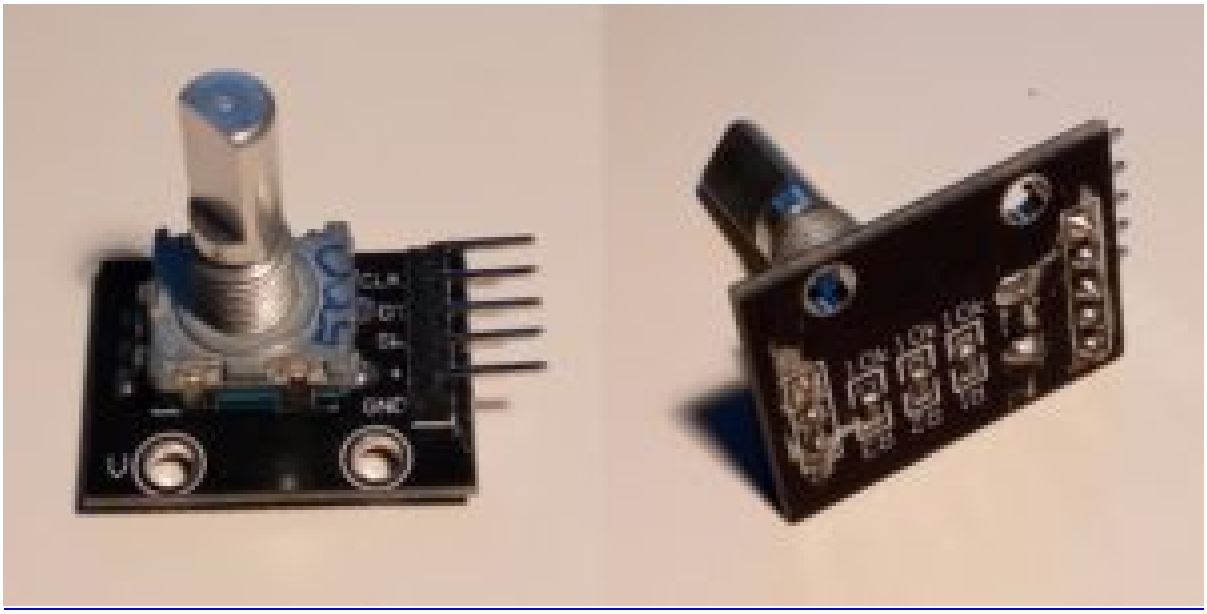
Jak bylo popsáno výše, enkodér je ve své podstatě složený ze dvou obyčejných spínacích nebo rozpínacích kontaktů[6]. Každopádně jsou úplně obyčejné a pro jejich spolehlivé připojení je potřeba zajistit, aby měly vždy správně definovanou úroveň výstupního signálu, HIGH nebo LOW, ať už jsou sepnuté nebo rozepnuté. Připojují se proto za pomoci pull-up nebo pull-down rezistorů, které jsou často osazeny už na modulu s enkodérem. Pokud nejsou, můžeme obvykle využít patřičný rezistor uvnitř mikrokontroleru – například u rodiny mikrokontrolerů AVR, které jsou v běžných malých Arduinech, je při konfiguraci vstupu pomocí funkce pinMode možné zvolit, zda nebude nebo bude aktivován – `pinMode(x,INPUT)`; a `pinMode(x,INPUT_PULLUP)`;.

Enkodér z horního obrázku má pět kontaktů, tři na jedné a dva na druhé straně těla (dva plíšky na zbylých stranách nejsou kontakty, slouží jen k uchycení těla enkodéru). Tři kontakty vlevo jsou připojeny ke dvěma spínačům, o kterých se psalo dosud: jeden kontakt je společný (na mém enkodéru prostřední), zbylé dva jsou druhý kontakt toho kterého spínače. Na druhé straně těla enkodéru jsou dva kontakty spínacího tlačítka, které se sepnou, když stiskneme osičku; s rotačními kontakty toto tlačítko není nijak spojeno.

Na hotových modulech je obvykle pět kontaktů (pinů). Tři jsou kontakty spínačů enkodéru a tlačítko na osičce, které jsou zapojeny svým druhým kontaktem na další pin modulu a jsou doplněny o pull-up rezistory, které jsou společně vyvedeny na poslední pin modulu. Označení kontaktů se dost různí, například mohou být označeny kanály A, B a společný kontakt COM (jako common), nebo jako na mém modulu CLK, DT a GND (clock, data, ground, i když tedy názvy clock a data tu nedávají moc smysl) – anebo jakkoli jinak. Společný kontakt

[6] *Ty enkodéry, co mám já, mají kontakty v klidu rozepnuté, ale mohlo by to být i opačně.*

rezistorů je na mém modulu označen „+“, protože jejich nejobvyklejší použití je jako pull-up rezistory, ale je možné je snadno využít i jako pull-down: připojíte „+“ na zem a GND na napájení[7].



Modul s enkodérem; při pohledu zdola jsou dobře vidět (kromě Číňanem nedbale omyté desky) pull-up rezistory pro enkodér a pro tlačítko

Programová obsluha

Pro Arduino existuje mnoho různých implementací pro připojení enkodérů. Některé projekty to řeší vlastním způsobem, ale pokud nemáte stejný hardware a stejný program jako ten projekt, může být využití obsluhy enkodéru ve vašem vlastním projektu obtížné (protože se to typicky bude s něčím tlouct). Existují také různé knihovny (AceButton, AnalogIOArduino, Encoder, EncoderStepCounter a další), já bych asi doporučil knihovnu „Encoder“ od Paula Stoffregena, která je docela dobře optimalizovaná a je funkční pro řadu různých Arduin, nejen založených na AVR, ale i „větších“ jako je ESPxx, různé ARM apod. Je určena pro obsluhu obecných enkodérů, takže při použití s „cvakacím knoflíkem“ je potřeba hodnotu čítače dělit čtyřmi, protože při každém přecvaknutí do sousední polohy dojde ke čtyřem změnám ve stavu těch dvou spínačů. Anebo si obsluhu můžete napsat a ušít na míru svého projektu sami, akorát bych stejně jako Paul Stoffregen upozornil, že je potřeba vyhodnocovat otáčení dostatečně často, aby vám nějaký krok neutekl. Při nesprávné implementaci, například když zpracování vstupů z enkodéru proložíte posíláním ladících dat po pomalé seriové lince, totiž velmi snadno získáte točítko, které vůbec nebude reagovat nebo se bude chovat naprosto chaoticky.

[7] *Samozřejmě POUZE pokud modul obsahuje jen takovýhle enkodér, kde není žádná další elektronika. Označení pak sice nesedí, ale elektricky to je v tomto případě naprosto v pořádku*

Použití knihovny Encoder

Knihovnu Encoder nainstalujete standardně ze „Správce knihoven“ v Arduino IDE. Každý enkodér bude reprezentován proměnnou typu Encoder, které při vytvoření určíte piny, kam jsou připojeny jeho výstupy[8]. Takových enkodérů můžete připojit tolik, kolik máte k dispozici volných dvojic digitálních pinů.

Jakmile máte proměnnou, máte víceméně hotovo. Kdykoli se jí zeptáte, dozvíte se aktuální počet nasbíraných impulsů (metoda read), můžete také aktuální počet nastavit na svou hodnotu (metoda write) anebo můžete získat počet impulsů od posledního dotazu (metoda readAndReset). Encoder počítá interně hodnotu v rozsahu typu long[9].

Následující prográmek pro Arduino vypíše při každém pootočení po seriové lince aktuální hodnotu čítače[10]:

```
#include <Encoder.h>

// Změňte následující čísla podle toho, kam máte enkodér připojený.
//   Nepřipojujte k pinům, kam je připojena LED !
//   Nejlepší výkon: oba piny je možné obsloužit přerušením
//   Dobrý výkon: přerušením je možné obsloužit pouze první pin
//   Nízký výkon: ani jeden pin není možné obsloužit přerušením
Encoder mujEnkoder(2, 3);

void setup()
{
  Serial.begin(115200);
  Serial.println("Základní test enkodéru:");
}

// trik, aby se už hned na začátku programu něco vypsalo,
// enkodér si při vytvoření totiž myslí 0.
long staraHodnota  = -999;

void loop()
{
  long novaHodnota = mujEnkoder.read() / 4;
  if (novaHodnota != staraHodnota)
  {
    staraHodnota = novaHodnota;
    Serial.println(novaHodnota);
  }
}
```

[8] Určujete tedy digitální piny Arduina pro ty dva spínače, společný pin připojíte na zem a pull-upy k napájení. Pokud chcete využít tlačítko osičky, připojíte ho jako každé jiné tlačítko s pull-upem.

[9] long je celočíselný typ s možnou hodnotou v rozsahu -2 147 483 648 až +2 147 483 647

[10] Program je převzat jako příklad z knihovny, pouze jsem přepsal komentáře a dělím hodnotu čítače 4, abych dostal jednotlivé cvaky

Autor knihovny uvádí, že je vhodné, aby alespoň první pin bylo možné nastavit pro detekci změny stavu pomocí vnějšího přerušení (external interrupt), lépe oba. Ale že to funguje i kdyby oba byly připojeny k pinům, na kterých toto není možné hlídat[11], jen je pak potřeba volat metodu read dostatečně často, aby pohyb zachytila, a mezi jednotlivými voláními nedělat nic časově náročného (autor uvádí, že například funkce `Serial.print()` může způsobovat problémy).

Enkodéry obecněji

Výše popsaný enkodér se používá jako jednoduché vstupní zařízení. Stejný princip se používá také pro snímání otáčení motoru nebo kol, ve starších počítačových myších s kuličkou se používal pro zjištění otáčení kuličky a tak spočítání polohy myši[12], v inkoustových tiskárnách se používá ke snímání pohybu tiskové hlavy a podobně. Pro takováto použití se častěji používají nekontaktní enkodéry, kde místo mechanických kontaktů jsou snímače optické (např. infračervená LED a fototranzistor) nebo magnetické (Hallovy snímače). Při pohybu se tak nic mechanicky nedotýká, pohyb nic nebrzdí a je plynulý (nepřecvakává). Je také možné vyhodnocovat otáčení až se čtyřnásobným rozlišením, než které nám poskytují běžné rotační enkodéry z fotografií v tomto článku[13]. A také se na takových enkodérech dá dosáhnout podstatně většího rozlišení, než 20 impulsů na otáčku (klidně tisíc i víc).

Porovnání obyčejných rotačních enkodérů typu „točítko s tlačítkem“ s obyčejnými potenciometry

Výhody:

- pro připojení stačí standardní digitální vstupy (není potřeba A/D převodník)
- je možno otáčet mnohokrát dokola (běžné potenciometry mají rozsah méně než jednou dokola; existují i víceotáčkové potenciometry, ale nejsou tak obvyklé a jsou drahé)
- mají tlačítko, které se dá použít třeba pro uložení nastavované hodnoty (potenciometry s tlačítkem existují, ale jsou dost drahé)

[11] *Na Arduinech Uno, Nano, Micro a všech dalších, ve kterých je ATmega168 nebo 328, jsou vnější přerušení dostupná pouze na pinech 2 a 3*

[12] *Kulička se opírá o dva navzájem kolmé válečky, které se podle pohybu kuličky otáčejí a toto otáčení se vyhodnocuje optickými enkodéry připojenými k válečkům*

[13] *Vyhodnocují se všechny 4 možné změny na spínačích a podle zapamatovaného předchozího stavu se rozpozná, kterým směrem se otáčí. To se také někdy označuje jako „režim 4x“.*

Nevýhody:

- pro připojení jsou potřeba dva vstupní piny (na rozdíl od potenciometru, pro který stačí jediný analogový vstup)
- mají malé rozlišení, tj. malý počet impulsů na jednu otáčku a není možno snímat jemnější otáčení, než jaké odpovídá jednomu „cvaku“ (u potenciometru je rozlišení dáno rozlišením AD převodníku, které je obvykle aspoň 8 bitů, tj. 256 hodnot přes celý rozsah)
- neposkytuje absolutní informaci, pouze relativní (tj. nevíte jeho natočení a snímáte jen, že se pootočil)
- vzhledem k mechanickým kontaktům je nutné zvážit potřebu ošetření zákmitů spínačů