ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **David** Jméno: **Petr** Osobní číslo: **420110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Řídicí jednotka s USB rozhraním pro optoelektronické inkrementální snímače

Název bakalářské práce anglicky:

USB Control Unit for Optoelectronic Incremental Encoder

Pokyny pro vypracování:

Navrhněte a s využitím mikrořadiče řady STM32 (STM32F042, F103, F303) realizujte řídicí jednotku pro vyhodnocení signálu optoelektronických inkrementálních snímačů s nulovým impulsem. Jednotka umožní přenos hodnot do PC s terminálovým programem prostřednictvím rozhraní USB i pomocí kanálu UART. Lokální zobrazení hodnot bude pomocí LCD. Navrhněte a realizujte funkce jednotky pro generaci dráhových impulsů i funkce pro měření rychlosti a dráhy. Mimo základních programů vytvořte též jejich rozšířené verze pro regulaci polohy nebo rychlosti akčních členů s inkrementálním snímačem. Vypracujte metodiku umožňující využít výsledky práce při realizaci řídicích jednotek s různými variantami mikrořadičů STM32.

Seznam doporučené literatury:

[1] Yiu, J.: The Definitive Guide to ARM Cortex -M0 and Cortex-M0+ processors
[2] STMicroelectronics:RM0091: STM32F0x2 Reference manual
[3] STMicroelectronics: DS10147 STM32F042 Data

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jan Fischer, CSc., katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **20.09.2019** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce zimního semestru 2020/2021

doc. Ing. Jan Fischer, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

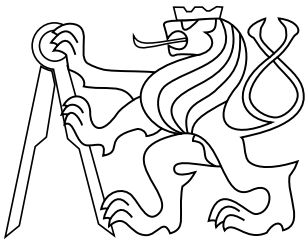
. _____
Datum převzetí zadání Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

bakalářská práce

Řídicí jednotka s USB rozhraním pro optoelektronické inkrementální snímače

Petr David



Prosinec 2019

doc. Ing. Jan Fischer, CSc.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra měření

Poděkování

Chtěl bych poděkovat svému vedoucímu práce doc. Ing. Janu Fischerovi, CSc. za vytrvalé a trpělivé vedení v průběhu dlouhého období psaní mé práce. Dále bych chtěl poděkovat svým rodičům za dlouhodobou podporu a v neposlední řadě bych chtěl poděkovat své přítelkyni za neutuchající péči.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Abstrakt

Práce se zabývá vývojem řídící jednotky pro vyhodnocení signálu optoelektronických inkrementálních snímačů s nulovým pulzem. Při návrhu je využito mikrořadičů řady STM32. Konkrétně jsou v této práci využity čipy STM32F042, STM32F103,STM32F303 a STM32L152. Řídící jednotka informuje o stavu měření prostřednictvím standardní aplikace v PC emulující sériový terminál. Komunikace v tomto případě probíhá pomocí protokolu UART nebo pomocí USB. Organizaci výstupu v textové podobě obstarává použitý mikrořadič. V této práci je ukázán ještě alternativní způsob lokálního zobrazování naměřených dat pomocí LCD displeje. Řídící jednotka částečně nahrazuje využití na trhu dostupných indikací a rozšířená verze ukazuje možnosti využití inkrementálních snímačů při regulaci akčních členů pomocí PID regulace.

Klíčová slova

Inkrementální snímače, STM32, čítače, LCD, indikace

Abstrakt

This work shows the development of the control unit for measuring the signal of optoelectronic incremental encoders with reference pulse. Variety of MCUs of STM32 family was used for designing this control. Specifically, units STM32F042, STM32F103, STM32F303 and STM32L152 were used. Control unit informs user about the state of measurement through standard serial terminal application in PC. For communication with serial terminal in PC USB and UART interfaces are used. Organization of the text output is controlled by used MCU. In this work, there is alternative way shown for local view of measurement values with LCD display. This developed control unit partially replaces the use of indication units available on the market. Extended variety of this controller unit shows possibilities of using incremental encoders in application automatic motor control.

Keywords

Incremental encoders, STM32, counters, LCD, indication

Obsah

1 Úvod

Náplní této práce je zpracování a realizace návrhu řídicí jednotky pro vyhodnocení kvadraturního signálu využívaného jako výstup optoelektronických inkrementálních snímačů. Tyto snímače se běžně používají v průmyslové automatizaci a najdou uplatnění kdekoli, kde je potřeba přesného měření polohy, úhlu natočení či rychlosti otáček.

Na trhu existuje množství dostupných indikačních jednotek, které jsou ale zaměřeny na profesionální využití v průmyslu a tomu odpovídá i jejich pořizovací cena. Pro rozšíření využití optoelektronických snímačů v rámci výuky na FEL, vznikla na katedře měření myšlenka vytvoření indikační jednotky pro tyto snímače s využitím mikrokontrolérů.

Motivací toho bylo, že na světě existují výrobci, kteří sice vyrábějí dostupné optoelektronické inkrementální snímače, avšak nenabízejí k nim jednoduché a levné indikace.Takovým případem je i firma LARM, která sice vyrábí řadu lineárních i rotačních inkrementálních snímačů polohy, nicméně vlastní indikační jednotku tato firma nenabízí. Výsledek této práce by pak měl být nabídnut firmě LARM, aby mohla tato firma při prezentaci svých produktů ukázat jednoduchost využití jejich snímačů.

Orientace této práce bude na mikrokontroléry of firmy STMicroelectronics jejichž produkty řady STM32 nabízejí nativní podporu pro čtení kvadraturního signálu prostřednictvím periferie čítačů. Dále práce popíše princip fungování čítačů v tomto módu pro čtení signálu z enkodéru, ale zároveň nastíní další způsoby využití této periferie.

Výsledná indikační jednotka bude komunikovat s uživatelem pomocí sériové komunikace zobrazované prostřednictvím standardní aplikace v počítači pro emulaci sériového terminálu. Zároveň ale bude možné lokální zobrazování stavu měření pomocí LCD displeje. Indikační jednotka by měla být schopná pracovat s více typy snímačů. Podstatnou součástí práce bude pak vytvoření metodiky pro přenositelnost vytvořeného programu mezi různými procesory řady STM32, aby výsledky práce mohly být využity i v nově vznikajících projektech.

2 Rozbor

Myšlenka vytvořit indikační jednotku pro práci se optoelektronickými inkrementálními snímači vznikla z touhy najít levnou a dobře přenositelnou alternativu ke komerčním řešením těchto indikací. Tyto indikace, stejně jako snímače pro které jsou určeny, jsou hojně využívané v průmyslové automatizaci pro práci s elektrickými pohony, obráběcím průmyslu či dalších oborech se zaměřením na měřící techniku. Indikační jednotky tak bývají velmi specializovaná a drahá zařízení. Nicméně s nízkou cenou, za kterou se dají pořídit neprofesionální snímače, se pojí rozšíření využití v amatérských projektech, kde chceme nějakým způsobem sledovat polohu natočení či posuvu nebo pracovat s měřením rychlosti nějakého pohyblivého prvku. Z tohoto důvodu se zdá příhodné mít nějakou méně nákladnou variantu ke zpracování signálu z těchto snímačů, než jsou číslicové indikace určené pro profesionální využití.

V rámci této práce je cílem vytvořit indikační jednotku, která alespoň částečně nahrazuje využití profesionálních měřících přístrojů. Účelem této indikační jednoty by mělo být jednoduché zobrazení měřených veličin posuvu nebo natočení pro demonstraci práce s inkrementálními snímači. Kromě indikace polohy by indikační jednotka měla uvádět i informaci o rychlosti pro ukázku práce s motorem poháněnými pohyblivými prvky. V rozšířené variantě patří mezi další cíle ukázat možnosti využití dat získaných ze snímače pro řízení daného akčního členu a s tím naznačit základní využití těchto snímačů v robotice a průmyslové automatizaci.

Lineární a rotační optoelektronické inkrementální snímače používají stejnou formu výstupního signálu. Pro správné fungování indikační jednotek je podstatné, aby uživatel specifikoval vlastnosti použitého snímače jako je typ a rozlišení senzoru. Některé senzory disponují indikací polohy pomocí tzv. 'nulového pulzu'. Takto vybavené snímače umožňují určení absolutní polohy bez ohledu na počáteční polohu, což typicky u inkrementálních snímačů není možné. Určení absolutní polohy má mnoho aplikací a proto jedním z požadavků na vytvářenou indikační jednotku je schopnost náležitě reagovat na impulz signalizující nulovou polohu.

S rozvojem mikrokontrolérů přibývá integrovaných periferií rozšiřujících schopnosti těchto už tak versatilních zařízení. Pro návrh mé vlastní indikační jednotky je klíčové rozšíření čítačových periferií některých mikrokontrolérů o podporu zpracování signálu z kvadraturních enkodérů. Potřebné HW periferie pro práci s enkodéry mají vestavěné některé z MCU řady STM32 od firmy STMicroelectronics a to je jedním z hlavních důvodů proč je tato práce orientovaná na jejich mikrořadiče. Dalším z důvodů je jejich široká nabídka vývojových desek obsahující ST-Link debugger a programátor usnadňující vývoj a testování programu.

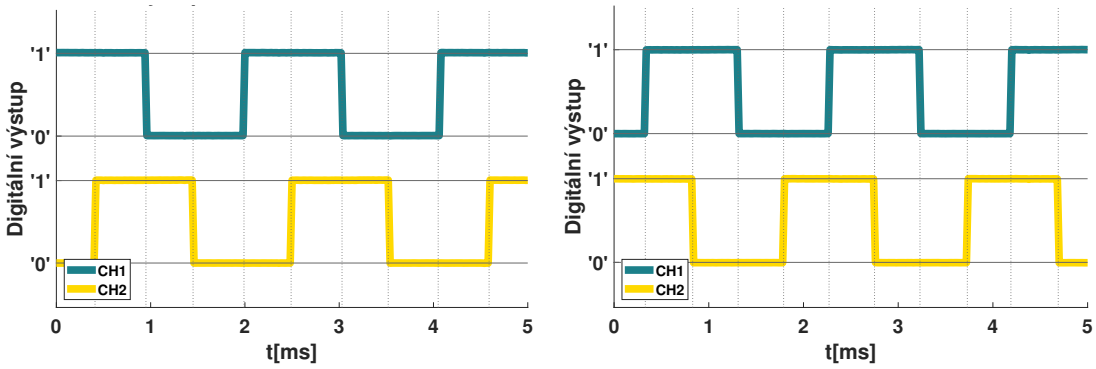
Samotné zpracování kvadraturního signálu pomocí mikrokontroléru však k vytvoření indikační jednotky nestačí. Jako další je potřeba vyřešit, jakým způsobem bude uživatel informován o naměřené hodnotě. Zdá se výhodné využít širokou nabídku rozhraní pro sériovou komunikaci, která je implementována do MCU řady STM32 a navrhnout způ-

sob jakým budou data posílána do počítače. Ač toto řešení by se mohlo zdát dostatečné, tak pro určité aplikace bychom mohli preferovat samostatně fungující měřící zařízení podobné právě komerční indikační jednotky. Pro lokální zobrazování hodnot bez využití počítače by bylo vhodné využít segmentového LCD displeje, jaké jsou obvyklé u další měřící techniky.

2.1 Popis funkce inkrementálního enkodéru

Enkodér je snímací zařízení, které umožňuje vyhodnocovat polohu, případně měřit rychlost pohybu. Existují dvě varianty těchto snímačů a to buď inkrementální (přírůstkové) enkodéry nebo absolutní enkodéry. Konstrukčně složitější absolutní enkodéry informují o poloze kódovaným signálem s distinktivním kódem pro každý jednotlivý měřící krok. Informace o přesné poloze snímače je tedy k dispozici ihned. Inkrementální enkodéry používají kvadraturní signál pro interpretaci změny polohy. Poskytují tedy jen relativní informaci o poloze vztaženou k výchozí pozici v momentu zapnutí snímače. Informují o přírůstku změny polohy a v literatuře můžeme tyto zařízení také nalézt pod označením přírůstkové snímače. V této práci jsem se dále zabýval pouze inkrementální variantou těchto snímačů.

Inkrementální nebo také někdy označované kvadraturní snímače informují o změně polohy pomocí dvou fázově posunutých signálů. Každá jednotlivá změna výstupního signálu pak informuje o změně polohy. Při konstantní rychlosti na výstupech pak můžeme pozorovat dva obdélníkové signály jejichž fázový posuv odpovídá čtvrtině periody a zmíněná dvojice signálů pak dohromady tvoří onen kvadraturní signál. Tento fázový posuv znamená, že jeden výstupní signál je vždy ve zpoždění za druhým signálem o čtvrtinu kompletního cyklu. Díky tomuto fázovému posunu, lze pak rozlišit směr změny polohy.[1]. Na obrázku 1 je zobrazen různý fázový posuv výstupních signálů v závislosti na směru pohybu změřeném inkrementálním snímači.

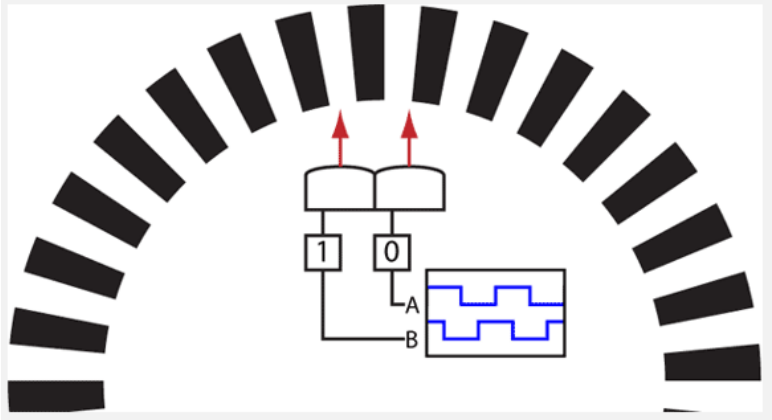


a) Pohyb ve směru hodinových ručiček **b)** Pohyb proti směru hodinových ručiček

Obr. 1 Výstup rotačního kvadraturního enkodéru pro dva různé směry pohybu

2.1.1 Princip vzniku signálu inkrementálních snímačů

Inkrementální snímače převádí rotační nebo lineární pohyb na elektrické impulzy pomocí fotoelektrického snímání rastrů dvou skleněných prvků nebo využívají Hallova jevu u snímačů magnetických [2] Viz obrázek 2. Rozlišení enkodéru je určeno počtem generovaných pulzů za jedno kompletní otočení rotačního snímače nebo počtem pulzů za posunutí o danou vzdálenost u snímače lineárního.



Obr. 2 Princip vzniku kvadraturního signálu převzato z[1]

2.1.2 Popis vlastností kvadraturního signálů

Na každém z dvou výstupů se nachází jedna ze dvou napěťových úrovní obdélníkového signálů, které se dají představit jako logické úrovně '1' a '0'. Výstupy kvadraturního enkodéru se tedy mohou nacházet ve čtyřech různých kombinacích (stavech). Viz tabulka 1.

Číslo stavu	1	2	3	4
Výstup A	0	1	1	0
Výstup B	0	0	1	1

Tab. 1 Různé výstupní kombinace kvadraturního enkodéru

Při každé změně výstupů - tedy při náběžné nebo sestupné hraně jednoho z výstupních kanálů je možno vyhodnotit směr pohybu díky znalosti napěťové úrovně druhého výstupního signálu.[1] Tedy například:

- Kanál A se mění z napěťové úrovně logické '0' do úrovně logické '1' (tedy náběžná hrana na signálu kanálu A)
- Kanál B se nachází v napěťové úrovni logické '0'
- Pohybuji se ve směru hodinových ručiček podle obrázku 1a

Definování směru pohybu si rovněž můžeme představit pomocí porovnání původního a nově vzniklého stavu podle tabulky 1 a směr pohybu určíme tak, že:

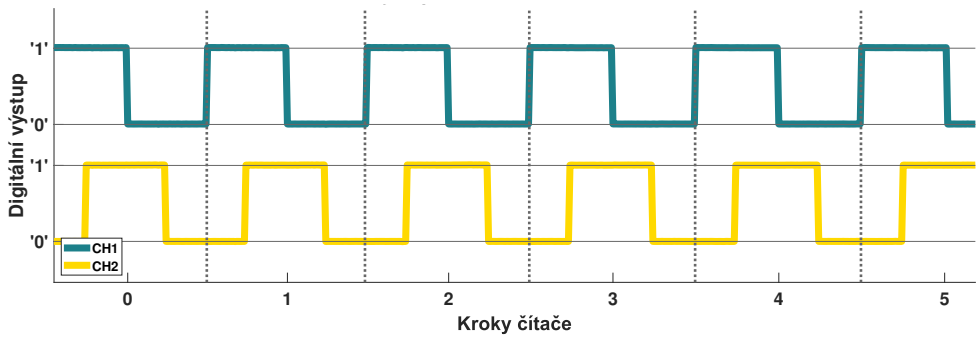
- Pro jeden směr platí tyto změny stavů 1→2→3→4→1→...
- Pro druhý směr platí tyto změny stavů 4→3→2→1→4→...

2.1.3 Různé typy kvadraturního kódování

Kvadraturní signál se dá vyhodnocovat různými způsoby odlišujícími se tím, které náběžné a sestupné hrany kvadraturního signálů uvažujeme. K zaznamenání se vždy používá čítač schopný měnit svoji hodnotu oběma směry(up/down counter). Zvolený typ kódovaní ovlivňuje rozlišovací schopnost indikace, tedy velikost jednoho měřícího kroku. V praxi se díky nejvyššímu rozlišení nejvíce využívá X4 kódování, kde uvažujeme každou hranu kvadraturního signálu. Nicméně v případě nižších nároků na přesnost měření může být výhodou zvolení méně přesného kódování a tím naopak zvětšit měřitelný rozsah. HW čítače mívají totiž 16 nebo 32 bitové registry pro uchování počtu měřících kroků a v případě použití snímačů s vysokým rozlišením může být pro nás měřitelná vzdálenost příliš limitující.

X1- vyhodnocování kvadraturního signálu

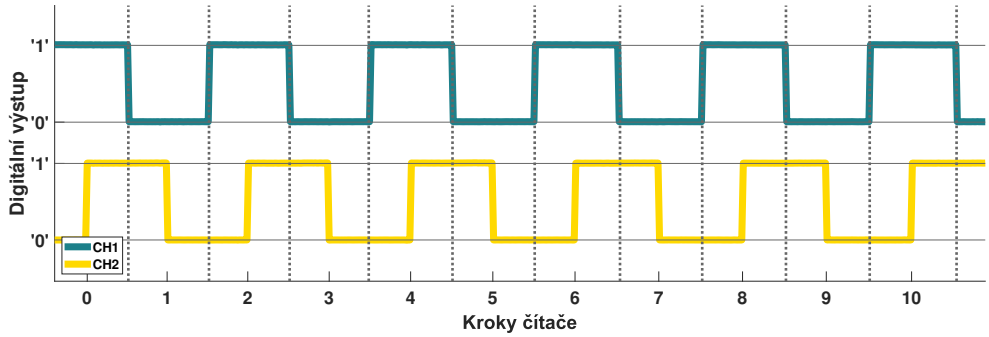
U X1 vyhodnocování se mění hodnota čítače pouze na jedné buď sestupné nebo náběžné hraně jednoho ze signálů. Měřící krok v tomto případě odpovídá 1 celé periodě vstupního signálu. Přestože nepočítáme pulzy na druhém kanálu, potřebujeme stále oba vstupní signály pro správné vyhodnocení směru čítání. Viz obrázek 3.



Obr. 3 X1 čítání pulzů kvadraturního enkodéru

X2- vyhodnocování kvadraturního signálu

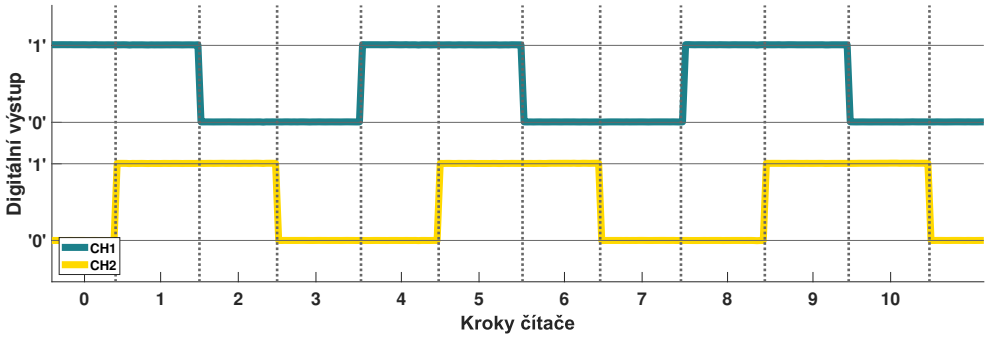
Podobně jako v případě X1 i u X2 vyhodnocování probíhá k čítání v závislosti na změnách signálu na jednom ze vstupních kanálů. Na rozdíl od X1 dochází k čítání při náběžných i sestupných hranách. Tímto se docílí dvojnásobného rozlišení oproti metodě vyhodnocování X1. Měřící krok odpovídá polovině periody vstupního signálu. Průběh čítání je zaznamenán na obrázku 4.



Obr. 4 X2 čítání pulzů kvadraturního enkodéru

X4- vyhodnocování kvadraturního signálu

V tomto případě se druhý kanál používá kromě určení směru také přímo ke zvýšení rozlišení. Při této metodě počítáme náběžné i sestupné hrany obou vstupních signálů. Díky tomu na jednotku délky nebo na jedno otočení spočítáme čtyřikrát tolik změn oproti počítání pulzů na jednom ze signálů. Měřící krok odpovídá čtvrtině periody vstupního signálu.



Obr. 5 X4 čítání pulzů kvadraturního enkodéru

2.1.4 Výpočetní vztahy pro vyhodnocení měření a určení vlastností snímačů

Pro převedení vyhodnoceného počtu kroků dle zvoleného kvadraturního kódování na údaj o změřené vzdálenosti nebo úhlu natočení musíme znát rozlišení přístroje. Rozlišovací schopnost rotačních enkodérů bývá nejčastěji vyjádřena v počtech pulzů na jednu celou otáčku. Pro upřesnění uvádím, že se vždy jedná o počet pulzů pouze na jednom z výstupů. Číslo se dá také chápat jako počet rysek na kotouči skleněného rastru optoelektronického snímače.

Podobná situace je u lineárních snímačů, kde je uváděn počet pulzů na jednotku vzdálenosti (například počet pulzů na milimetr). Alternativně potom bývá uváděn nejmenší možný měřící krok. Tím je myšlena vzdálenost odpovídající jednomu kroku při X4 kódování, tedy vzdálenost mezi 2 změnami jednoho ze signálu. Pro sjednocení přístupu k oběma typům snímačům jsou definovány následujícím způsobem výpočetní vztahy pro přepočtové konstanty, které charakterizují rozlišení jednotlivých typů snímačů. Čísla N_R a N_L určují počet metrických jednotek ke kolika je příslušná konstanta vztažena. Zavádím tedy:

- K_R pro rotační snímače s $N_R = 360\ ^\circ$ - počet stupňů v plném úhlu
- K_L pro lineární snímače s $N_L = 1$ mm -konstanta je vztažena na 1 mm

Použité kódování dle kapitoly 2.1.3 popisuje počet měřících kroků na jednu periodu vstupního signálu. Ve vzorcích je toto kódování zohledněno pomocí konstant E_X $[-]$ za které dosazujeme hodnoty podle tabulky 2. Díky těmto definicím lze základní převodní vztahy mezi přepočtovými konstantami a k nim náležejícími nejmenšími měřícími kroky d popsat pomocí obecných vztahů (1 a 2).

Použité vyhodnocování	E_X
X1	$E_{X1} = 1$
X2	$E_{X2} = 2$
X4	$E_{X4} = 4$

Tab. 2 Počet měřících kroků na 1 periodu měřícího signálu E_X

$$K = \frac{N}{d \cdot E_X} \tag{1}$$

$$d = \frac{N}{K \cdot E_X} \tag{2}$$

Pro rotační snímače přepočtová konstanta K_R odpovídá uváděnému počtu pulzů na otáčku. Nejmenší měřitelný krok rotačního snímače d_R [°] pro snímač s 600 pulzy na otáčku při měření s největším možným rozlišením(X4) s odkazem na vztah (2) dopočítáme následujícím způsobem:

$$d_R = \frac{N_R}{K_R \cdot E_X 4} = \frac{360^\circ}{600 \cdot 4} = 0.15^\circ \tag{3}$$

Pro lineární snímače je situace opačná. Výrobci nejčastěji uvádí nejmenší měřící krok a z něj pak dopočítáváme přepočtovou konstantu K_L. Nejmenším měřícím krokem lineární snímače d_L [mm] se opět myslí uražená vzdálenost mezi dvěma hranami kvadraturního signálu, tedy měřící krok odpovídající X4 kódovaní. Přepočtová konstanta zde vyjadřuje počet pulzů na jeden mm. Pro lineární snímač s rozlišením 5 μ dopočítáme K_L podle vzorce (1):

$$K_L = \frac{N_L}{d_L \cdot E_{X4}} = \frac{1}{0.005 \cdot 4} = 50 \tag{4}$$

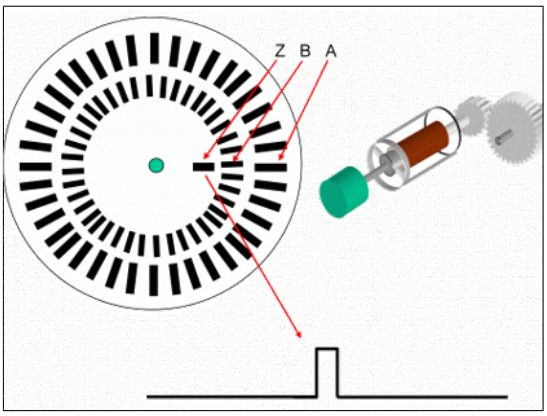
Čítačem zpracováváme kvadraturní signál pomocí zvoleného druhu kódování. Výsledkem toho je počet měřících kroků n od výchozí polohy. Tento počet měřících kroků poté převedeme na vzdálenost či úhel natočení uražené od výchozí polohy pomocí výše zmiňovaných definic. V níže uvedených vzorcích (5) a (6) označujeme počet změřených kroků(stav čítače) n. Výsledný úhel natočení α [°] a změřenou vzdálenost l [mm].

$$\alpha = \frac{n \cdot N_R}{K_R \cdot E_X} = \frac{n \cdot 360^\circ}{K_R \cdot E_X}\ [^\circ] \qquad (5) \qquad l = \frac{n \cdot N_L}{K_L \cdot E_X} = \frac{n \cdot 1\ \text{mm}}{K_L \cdot E_X}\ [\text{mm}] \qquad (6)$$

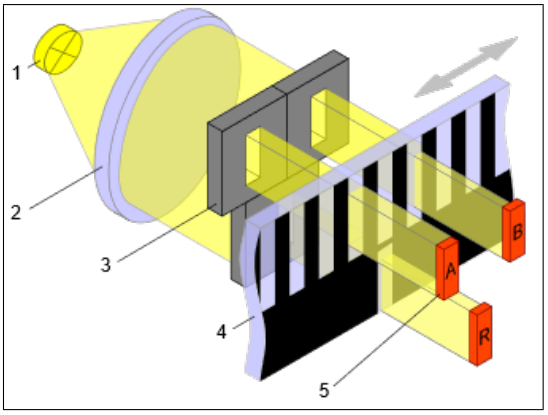
2.2 Určení absolutní polohy snímače pomocí nulového pulzu

Jak bylo zmíněno v kapitole 2.1 hlavní nevýhodou inkrementálních enkodérů oproti absolutním je, že z jejich výstupních signálů nedokážeme určit informaci o absolutní poloze. Z tohoto důvodu některé snímače, kromě dvou výstupních signálů poskytujících výše zmíněný kvadraturní signál, nabízejí navíc ještě třetí výstup řešící tento nedostatek. Tento výstup se nazývá referencí, signálem nulového pulzu či se používá označení "index pulse". Výstup ze snímače se většinou označuje písmeny 'R','Z' nebo také 'N'.

Rastr pro optoelektronické nebo magnetické snímání v tomto případě obsahuje referenční značku pro určení nulové pozice snímače. Viz obrázky a 6 a 7 . V této poloze elektronika snímače vygeneruje na výstup obdélníkový pulz. Při zapnutí snímače stačí přejet jednou přes tuto referenční značku, v ní vynulovat čítač a dále již máme informaci o absolutní poloze v podobě počtu pulzů od výchozí pozice, která se nemění. Díky tomu je poté možno nastavit požadovanou polohu bez znalosti startovní pozice.

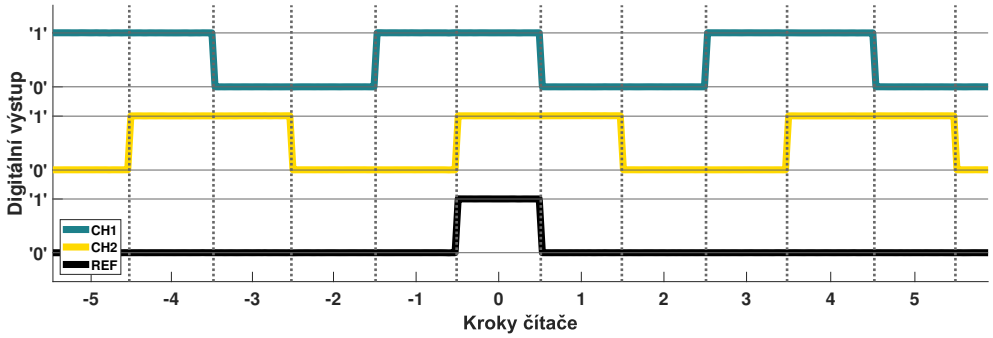


Obr. 6 Nulový pulz u rotačního snímače, převzato z [3]



Obr. 7 Nulový pulz u lineárního snímače, převzato z [4]

Digitální výstup z takovýchto snímačů je zaznamenán na obrázku 8. Pro určení absolutní polohy tedy potřebuje indikační jednotka zpracovávat signál ze všech tří výstupů snímače.



Obr. 8 Vyhodnocení výstupu kvadraturního enkodéru s nulovým pulzem

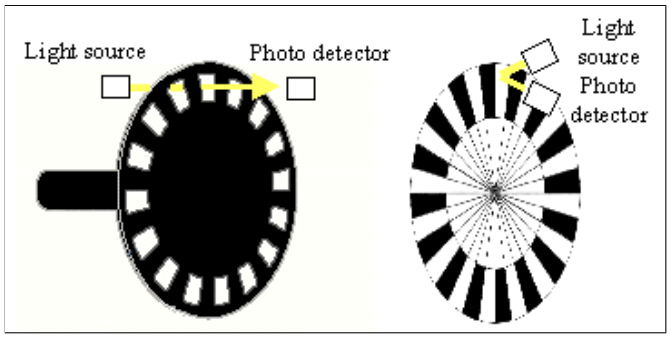
2.3 Optoelektronické inkrementální snímače

Optoelektronické inkrementální snímače pro zpracování rotačního nebo lineárního pohybu využívají fotoelektrického snímání rastru k vytvoření kvadraturního signálu. Využívá se přitom transparentní systém nebo systém reflexní. Tyto dva systémy můžeme od sebe rozlišit podle toho, jestli vyhodnocovací obvod zpracovává světlo odražené od reflexního rastru nebo vyhodnocuje světlo, které rastrem prochází. V obou případech je pak intenzita světla dopadajícího na snímací obvod dále zpracována pomocí elektroniky pro vytvoření obdélníkových signálů vhodných pro digitální zpracování pomocí čítačů. [5]

2.3.1 Rotační a úhlové optoelektronické snímače

Tyto snímače slouží pro měření úhlu natočení a počtu otáček. Dělají se v různých provedeních odlišující se způsobem montáže, výstupním napětím, přítomností nulového pulzu a hlavně počtem pulzů na jednu otáčku určující rozlišení snímače.

Úhlové snímače se vyznačují vysokou rozlišovací schopností - až do oblasti zlomků úhlových vteřin. Způsoby používání těchto zařízení jsou např. otočné stoly a naklápěcí hlavy u obráběcích strojů, děličky, vysoce přesné úhlové měřicí stoly, přesné systémy



Obr. 9 Rozdíl mezi reflektivním a transparentním způsobem snímání převzato z [5]

měření úhlu, antény a teleskopy. Typický počet pulzů na otáčku je 9000-180000.

Rotační snímače slouží jako snímače hodnoty rotačního pohybu, úhlových rychlostí a ve spojení s mechanickým měřítkem (jako je např. závitové vřeteno) také ke zjišťování lineárních pohybů. Oblasti použití jsou mimo jiné elektrické pohony, obráběcí stroje, tiskárenské stroje, roboty, zdviže a manipulátory, měřicí a zkušební přístroje nejrůznějšího druhu. Počet pulzů na otáčku bývá oproti úhlovým snímačům menší a to typicky 16 až 5000[6, str. 18]

Optoelektronické rotační snímače se typicky skládají ze základního tělesa v němž je uložen systém odečítání, skleněný rastrový pár, (stator a rotor), osvětlovací systém a vyhodnocovací elektronika. [7]

Pro svá měření jsem měl k dispozici rotační enkodér s označením LPD3806 s 600 pulzy na jedno otočení od značky GTEACH vyrobený v Číně zobrazený na fotografii 10. Tento typ snímače má vyvedeny 4 piny. 2 pro napájení a 2 pro výstupní kvadraturní signál. Vývody tohoto snímače jsou typu otevřený kolektor. Dále jsem měl k dispozici rotační snímač s diferenciálními výstupy včetně výstupů pro signál reference IRC125 od firmy LARM viz obrázek 11. Tento snímač poskytoval 500 pulzů na otáčku a výstupní signál typu linkový budič RS422, který je kompaktibilní s logikou TTL.

Obr. 10 Fotografie použitého rotačního enkodéru LPD3806[8]

Obr. 11 Fotografie použitého rotačního enkodéru IRC125[8]

2.3.2 Lineární optoelektronické snímače a měřící sondy

Obě zařízení slouží k přesnému odměřování lineárních vzdáleností. Od sebe navzájem se liší především měřícím rozsahem. Měřící sondy mají typický rozsah 10-100mm a lineární snímače od 50 mm do konstrukčních řešení v řádech metrů.

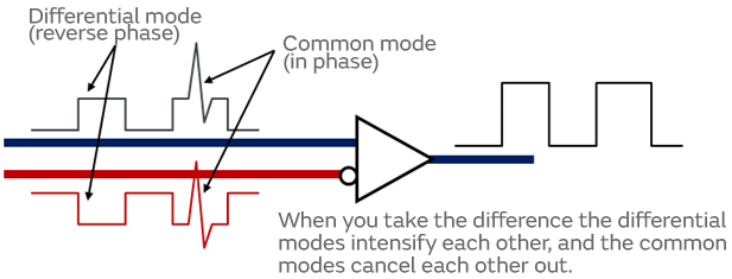
Měřící sondy tvoří základní těleso s lineárním kuličkovým ložiskem, vratnou pružinou, skleněný rastrový pár, osvětlovací systém a vyhodnocovací elektronika. Využití měřicích sond je velmi všestranné. Používají se v robotice, laboratorní technice, lékařství, přesném strojírenství, pro průběžné měření tloušťky materiálu i jako přesný koncový spínač.[9]

Pro svá měření jsem měl k dispozici měřící sondu od firmy LARM typ MSL50.162PA. Vyobrazenou na obrázku 12 Tato měřící sonda má měřící krok 5 μ, napájecí napětí 5V a výstupní signál úrovně TTL.

Obr. 12 Fotografie použité měřící sondy MSL50 [8]

2.3.3 Popis typů výstupů u inkrementálních snímačů

Optoelektronické i magnetické inkrementální snímače se vyrábějí s různým provedením výstupů. Liší se od sebe napěťovými úrovněmi, tvarem výstupního signálu a elektronikou použitou pro generování výstupního signálu. Provedení výstupů je dobré nejdříve rozdělit na výstupy s kvadraturním obdélníkovým signálem, který je výsledkem vyhodnocovacího obvodu vevnitř snímače a výstupy tzv. SIN/COS enkodérů, které mají na výstupu signál podobný harmonické funkci. Dále se provedení výstupů dělí na jednoduché a diferenciální. U diferenciálních výstupů se na dvou vodičích přenáší stejný signál, ale s opačnou polaritou. Informaci pak přenáší rozdíl(diference) mezi těmito dvěma signály. Výhodou tohoto řešení je, že případné rušení vstupuje do vodičů(vedených vedle sebe) ve stejné fázi a na výstupu se tedy potom neprojeví. Vysvětlující diagram je na obrázku 13. Toto provedení výstupů se využívá právě díky své odolnosti proti rušení.



Obr. 13 Zpracování rušení u diferenciálního výstupu. Převzato z [10]

Výstupy s obdélníkovým tvarem signálu

Otevřený kolektor (Open collector)

Elektronika výstupu je tvořena pouze jedním tranzistorem a v zapojení jako výstup schopný odebírat proud. Ve vypnutém stavu je na výstupu plovoucí napětí, protože kolektor není spojen s žádným napětím. V zapnutém stavu je pak kolektor spojený se zemí.

U enkodérů s tímto druhem výstupu musíme pro připojení na vstup připojit tzv. pull-up rezistor. Pull-up rezistor pak na vstupu zajišťuje napětí pro logickou úroveň '1', když je tranzistor ve vypnutém stavu

Výhodou tohoto druhu výstupu je možnost jej přímo zapojit na vstup MCU, bez obavy přetížení vstupního pinu.

Dvoujčinný výstup -Push-Pull, HTL, TTL

Dva tranzistory jsou v zapojení tak, že ve vypnutém stavu je tento výstup propojený se zemí a v aktivním stavu je výstup propojen s kladným napájecím napětím.

- TTL(Transistor-transistor logic)
  TTL je velmi rozšířeným standardem pro implementaci digitálních integrovaných obvodů.

  Pro výstupní napěťové úrovně platí:$U_{výstup} \leq 0.4$ V pro logický stav '0'
  a 5 V$\geq U_{výstup} \geq 2.7$ V pro logický stav '1'

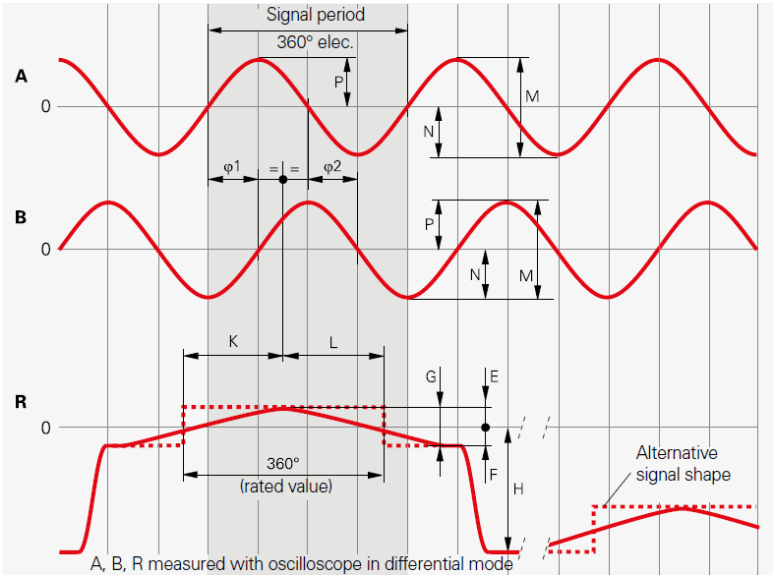
- HTL(High treshold logic)
  U logiky HTL se napájecí napětí pohybuje typicky v rozmezí 8-30V. Vysoký rozkmit napětí úrovní se využívá též jako ochrana proti rušení.

  Pro výstupní napěťové úrovně platí:$U_{výstup} \leq 1$ V pro logický stav '0'
  a $U_{výstup} \geq 3$ V pro logický stav '1'.

Sin/Cos výstupy enkoderů

Snímače s kvadraturním obdélníkovým signálem zpracovávají signál ze snímacího obvodu, jehož signál je podobný harmonické funkci. Pro zvýšení rozlišení snímačů se využívá výkonných interpolátorů. Ty mohou být součástí snímače jako takového nebo mohou být v rámci externího vyhodnocovacího obvodu. Sin/Cos typy enkodéry mají

signálové výstupy přizpůsobeny pro externí interpolaci signálu. Výstup takovýchto enkodérů je ukázán na obrázku 14. Zpracovávají signál ze snímacího obvodu pro dosažení definovaného napěťového nebo proudového rozsahu na výstupech. Firma Heidenhain vyrábí Sin/Cos enkodéry ve variantě poskytující napěťový signál $1V_{PP}$ tedy s rozsahem 1V nebo ve variantě poskytují proudový signál $11\mu A_{PP}$ s výstupním rozsahem $11\mu A$. [11] Existují různá provedení včetně Sin/Cos enkodérů s diferenciálními výstupy nebo se signálem určující nulovou polohu. Díky možnosti externí interpolace můžeme přímo ovlivnit rozlišení měření, a přizpůsobit ho našim nárokům.



Obr. 14 Ukázka výstupního signálů ze' sin/cos' snímačů polohy. Převzato z [11]

2.4 Indikační jednotky pro optoelektronické snímače

Jedná se přístroje určené ke zpracování signálu z inkrementálních snímačů. Zpravidla slouží k vizualizaci naměřených hodnot z jednoho nebo i více snímačů. Používají se pro monitorování měřících přístrojů, práci s manuálně ovládanými výrobními stroji či kalibraci automaticky pracujících strojů. Základem bývá čitelný alfanumerický displej a klávesnice ovládající rozšiřující funkce indikační jednotky.

2.4.1 Popis číslicové indikace Essa ADP1

Tento přístroj od firmy ESSA je určený pro zpracování dat z inkrementálních snímačů s výstupním signálem RS 422 nebo TTL a jejich zobrazování na číslicovém displeji LED. Umožňuje připojení snímačů s diferenciálním výstupem i snímačů s nulovým pulzem. Dále mezi jeho funkce patří vysílání dat po sériové lince RS232. K dispozici na přístroji je šest ovládacích tlačítek, pomocí kterých ovládáme funkce jako: nastavení referenční značky, změna směru čítání, hlídání vstupní frekvence snímačů, vypínání s pamětí polohy, přepínání mezi absolutním a přírůstkovým zobrazováním, nulování, zmrazení údaje na displeji, zobrazování úhlů a další.[12] Vzhled přístroje je vyobrazen na obrázku16.

Obr. 15 Číslicová indikace ADP1 převzato z[12]

2.4.2 Popis číslicové indikace Heidenhain ND5023

Tato indikační jednotka je zástupcem pokročilejších indikací vyrobena pro použití na ručně ovládaných obráběcích strojích. V kombinaci s lineárními a rotačními snímači úhlů zobrazí umístění nástroje ve více než jedné osy a poskytuje další funkce pro práci obráběcích strojů. Je určena pro enkodéry s výstupním signálem typu TTL s maximální vstupní frekvencí 500 KHz. Kromě počtu snímačů, které mohou být najednou připojeny k zařízení, se tato indikační jednotka od ADP1 také liší v tom, že ji lze ovládat přes datové rozhraní USB z hostitelské aplikace. [13] Tato jednotka má grafický displej umožňující sledování měření až na 3 snímačích zároveň.

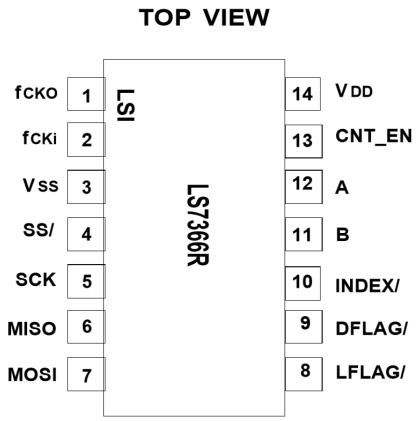
Obr. 16 Číslicová indikace ND5023 převzato z [13]

2.5 Integrované obvody pro dekódování kvadraturního signálu

Tyto specializované obvody mají za funkci zpracovávat kvadraturní signál, ukládat měření do vlastního registru a předávat tuto informaci do mikrokontroléru. Ke komunikaci s MCU tyto obvody většinou využívají sériové komunikace. Využijeme je ale především, pokud námi používaný mikrokontrolér čítání kvadratuního signálu nepodporuje nebo pokud potřebujeme zpracovávat signál z více snímačů, než nám umožní naše zařízení. Mezi firmy zabývající se mimo jiné i těmito specializovanými obvody patří Agilent Technologies nebo LSI Computer Systems.

2.5.1 Popis vlastností a využití obvodu LS7366

Integrovaný obvod LS7366R od firmy LSI Computer systems nabízí 32-bitový registr pro ukládání měření a komunikační rozhraní pomocí SPI. Dále obvod umožňuje používat různé způsoby čtení kvadraturního signálu zmíněné v kapitole 2.1.3 a podporuje několik režimu měření využívající nulový pulz. Při napájení 5V umožňuje číst kvadraturní signál do maximální frekvence 9.6 Mhz. Obsahuje též číslicový filtr vstupních signálů pro potlačení rušení. [14]



Obr. 17 Vyvedení pinů integrovaného obvodu LS7366R [14]

3 Periferie vhodné pro realizaci řídicí jednotky

Tato práce má za cíl vytvoření zařízení, které alespoň částečně nahrazuje funkce komerčních číslicových indikací podobných uvedeným jednotkám v kapitole 2.4. V této kapitole jsou jednotlivě popsány dílčí komponenty vhodné pro tento účel. USB a UART periferie slouží pro komunikaci s PC. Prostřednictvím těchto periferií bude uživatel moci číst naměřené hodnoty a ovládat nastavení indikační jednotky tak, aby indikační jednotka mohla pracovat s různými snímači. Dále je popsáno využití displejů, pro prosté zobrazování naměřených hodnot bez potřeby počítače. Základem indikační jednotky jsou pak samotné čítací periferie pro zpracování kvadraturního signálu enkodérů.

Pro čítací periferie se tato práce orientovala na čítače v mikrokontrolérech od firmy STMicroelectronics konkrétně řady STM32. Některé z čítačů obsažených v těchto zařízeních podporují čítání kvadraturního signálu a mohou tedy skvěle posloužit jako základní stavební kámen pro vlastní indikační jednotku. Dále mohou být tyto čítače využity pro realizaci dalších funkcí jako pro řízení aktivního členu pomocí PWM signálu nebo pro periodické generování pulzů v závislosti na uražené vzdálenosti.

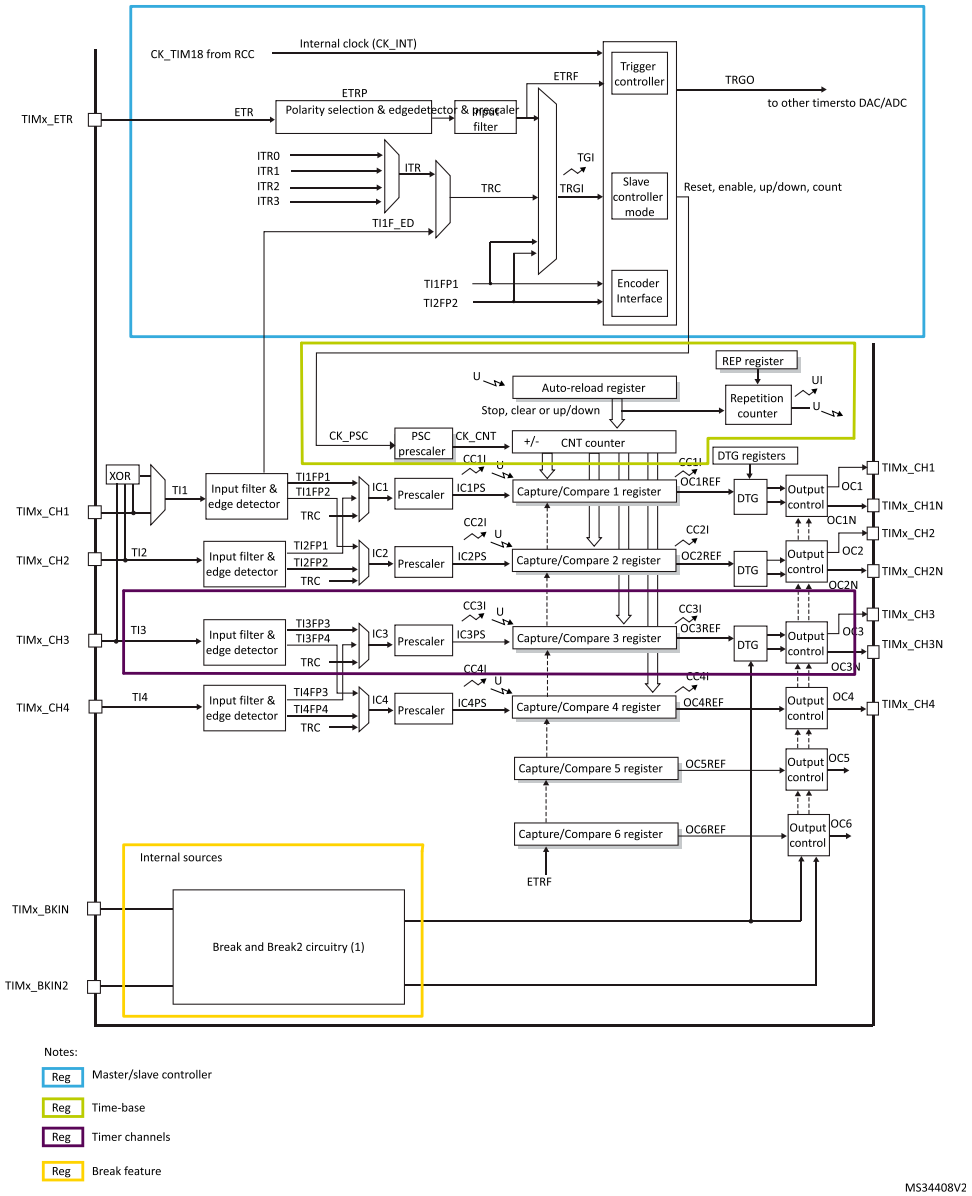
3.1 Časovače/Čítače (Timers) v STM32 mikrokontrolérech

Tyto obvody jsou součástí základní výbavy všech STM32 mikrokontrolérů. Jedná se HW obvody (periferie), u kterých rozlišujeme jestli se jedná o čítač nebo časovač v závislosti na tom, jakým způsobem a za jakých okolností mění periférie hodnotu ve svém přiřazeném čítacím registru označovaném CNT (counter register). V případě časovačů se pak jedná o čítání periodických impulsů daných frekvencí řídícího oscilátoru a čítače mění svojí hodnotu v závislosti na vnějších změnách vstupního signálu. V dokumentaci k mikrokontrolerům značky STM32 najdeme tuto periferii pod anglickým označením 'timer' (TIM)
. Dle [15, str. 5] rozlišujeme několik tříd časovačů, podle jejich složitosti a možnostmi využití:

- Časovače s pokročilými možnostmi nastavení -např. TIM1,TIM8
- Víceúčelové časovače - např. TIM2, TIM3
- Časovače s odlehčenými možnostmi nastavení např. TIM9, TIM16
- Časovače s základními možnostmi nastavení -např. TIM6 a TIM7

3.1.1 Popis funkce TIM periferie v procesorech STM32

Na obrázku 1.1 je možné sledovat blokové schéma popisující funkci obvodu časovače TIM 1. Hlavní jsou čtyři bloky. Řídicí jednotka Master/Slave(Master/slave controller), blok časové základny(Time-base) s registrem CNT, jehož hodnota inkrementuje nebo dekrementuje v závislosti na vstupních signálech a nastavení. Dále sledujeme bloky pro jednotlivé kanály časovače a nakonec blok přerušení(Break feature).



Obr. 18 Blokové schéma časovače s pokročilými možnostmi nastavení TIM1, převzato z [15]

3.1.2 Popis registrů řídící chování TIM periferie

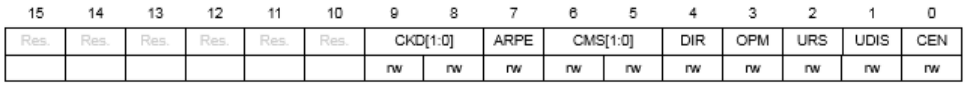
V této kapitole jsou popsány registry čítačů v STM32 mikrokontrolérech, které slouží pro definování chování této periferie a sledování jejího stavu. Je zde uvedeno, jak tyto registry nastavit pro využití dané periferie pro čítaní signálů z kvadraturních enkódérů. Samotné nastavení registru lze pak provést přímo definováním jejich hodnot v našem programu nebo je možné využít automatické inicializace s využitím STM32CubeMX, která vytvoří inicializační část programu za nás.

Malé x v názvech registrů zastupuje číslo jednotlivých časovačů tedy TIMx může představovat TIM1 TIM2 atd. Jednotlivé periferie TIMx mají obdobnou strukturu jejich ovládacích registrů. Nicméně se význam jednotlivých registrů mění se zvyšující se složitostí dané periferie, a je tedy vždy nutné vlastnosti a určení registrů ověřit s referenčním manuálem pro konkrétní MCU a konkrétní TIMx. Význam registrů zde popsa-

ných je založen na informacích [16, str. 435-455] pro MCU STM32F042 a TIM2. Vždy je potřeba si ověřit specifika jednotlivých registrů s příslušným referenčním manuálem k danému MCU.

TIMx control register 1 (TIMx_CR1)

Offset adresy: 0x00
Hodnota po resetu: 0x0000



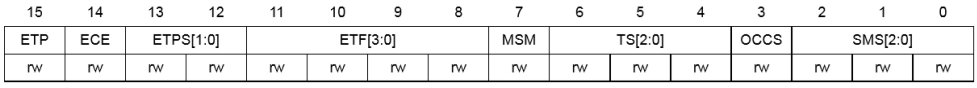
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Obr. 19 Význam bitů v TIMx_CR1 control register 1 [16]

- CKD[8:2]: Clock division
  Nastavuje děličku vstupního hodinového signálu s možnými hodnotami:
  B00-1,B01-2,B10-4, B11 je rezervovaná hodnota.
- ARPE[7:1]: Auto-reload preload enable
  určuje jestli je nebo není nastavena hodnota pro automatické přetečení v registru TIM1 auto-reload register (TIMx_ARR).
- DIR[4:1]: Direction
  Tento bit určuje směr čítání. V enkodérovém modu je tento bit dostupný pouze pro čtení.
- OPM[3:1]:One-pulse mode Určuje jestli se při přetečení čítačem časovač zastaví nebo ne.
- CEN[0:1]: Counter enable.
  Tato hodnota určuje zda časovač je v provozu.

TIMx slave mode control register (TIMx_SMCR)

Offset adresy: 0x08
Hodnota po resetu: 0x0000



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Obr. 20 Význam bitů v TIMx_SMCR registru [16]

Společně s řídícími registry TIMx_CR1 a TIMx_CR2 určuje tento registr chování bloku časovače Master/slave controller znázorněném na obrázku 18. Pro mód enkodéru jsou v tomto registru podstatné bity SMS.

- SMS[0:3] Slave mode selection
  : B000:Slave mód neaktivní
  . B001:Encoder mode 1 čítání na hranách signálu na TI2FP2. Směr čítání záleží na úrovni signálů TI1FP1. Jestli dochází k čítání na nábežných nebo sestupných hranách záleží na nastavení registru TIMx_CCER
  . B010:Encoder mode 2 Podobný jako encoder mód 1, ale k čítání dochází na hranách signálu TI1FP1 v závislosti na stavu TI2FP2
  . B011:Encoder mode 3 k čítání časovače dochází na hranách TI2FP2 i TI1FP1. Směr čítání znovu ovlivňuje stav druhého ze signálů, než na kterém došlo ke změně.

TIMx Capture/compare mode register 1 (TIMx_CCMR1)

Offset adresy: 0x18
Hodnota po resetu: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Obr. 21 Význam bitů v TIMx_CCMR1 registru [16]

Kanály časovače mohou sloužit jako vstup (capture mode) nebo jako výstup(compare mode). Použití jednotlivých vstupních kanálů určují bity CCxS. Zbylé bity mají různý význam podle zvoleného druhu použití. Jednotlivé bity OCxx popisují vlastnosti příslušného kanálu použitého jako výstup a ICxx popisují vlastnosti kanálů v režimu vstupu [16].

- CC2S[8:2]: Capture/Compare 2 selection
  B00: TIMxCH2 slouží jako výstup
  B01: TIMxCH2 slouží jako vstup, TI2FP2 a IC2 je namapován na vstup TI2
  B10: TIMxCH2 slouží jako vstup, IC2 je namapován na vstup TI1
  B11: IC2 je namapován na signál TRC.
- CC1S[0:2]: Capture/Compare 1 selection
  B00: TIMxCH1 slouží jako výstup
  B01: TIMxCH1 slouží jako vstup, TI1FP1 a IC1 je namapován na vstup TI1
  B10: TIMxCH2 slouží jako vstup, IC1 je namapován na vstup TI2
  B11: IC1 je namapován na signál TRC.
- ICxF Input capture filter
  číslicová filtrace vstupního kanálu x
- ICxPSC Input capture prescaler

TIMx capture/compare enable register (TIMx_CCER)

Offset adresy: 0x20
Hodnota po resetu: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Obr. 22 Význam bitů v TIMx_CCER registru [16]

- CC1NP/CC1P Capture/Compare 1 output Polarity
  Tyto bity dohromady určují jestli je časovač citlivý na náběžnou hranu, sestupnou hranu nebo na obě hrany signálu IC1 respektive TIxFP1.
  B00: obvod je citlivý na náběžnou hranu
  B01: obvod je citlivý na sestupnou hranu
  B10: je rezervovaná hodnota - nepoužívat
  B11: obvod je citlivý na obě hrany, nesmí se používat při módu enkodéru.
- CC2NP/CC2P Capture/Compare 2 output Polarity
  Tyto bity dohromady určují jestli je časovač citlivý na náběžnou hranu, sestupnou hranu nebo na obě hrany signálu IC2 respektive TIxFP2.
  Definice bitů je stejná jako v případě CC1NP/CC1P jen pro I2C respektive TIxFP2.

Hodnotové registry

Tyto registry obsahují 16 nebo 32 bitové hodnoty podle daného časovače.

- TIMx_CNT: Counter value
  Offset adresy: 0x24, hodnota po resetu: 0x0000
  Aktuální stav čítače/časovače
- TIMx_PSC: Prescaler
  Offset adresy: 0x28, hodnota po resetu: 0x0000
  16-bitová hodnota určující dělení vstupní frekvence. Použití této hodnoty dává smysl pouze v módu časovače.
- TIMx_ARR: Auto-reload register
  Offset adresy: 0x2C, hodnota po resetu: 0xFFFFFFFF
  maximální hodnota pro CNT registr čítače. Při překročení této hodnoty dojde k vynulování CNT registru do nuly a začne pracovat od počátku. Při záporném směru čítání dojde po překročení 0 k nahrání hodnoty ARR do CNT registru.
- (TIMx_DIER) DMA/Interrupt enable register
  Offset adresy: 0x0C, hodnota po resetu: 0x0000
  Určuje, které části časovače můžou vyvolat přerušení a DMA transfer.

3.1.3 Popis módu enkodéru(Encoder mode) periferií TIM

Tento mód čítače podporují pouze některé obvody Timerů v rodině STM. Jmenovitě to jsou víceúčelové časovače a časovače s pokročilými možnostmi nastavení. V tabulce 3 je uveden seznam podporovaných časovačů v této práci používaných MCU.

Velikost CNT registru	16 bitů	32 bitů
STM32F042	TIM1, TIM3	TIM2
STM32F303	TIM1,TIM3,TIM4,TIM8	TIM2
STM32F103	TIM1,TIM2,TIM3,TIM4	-

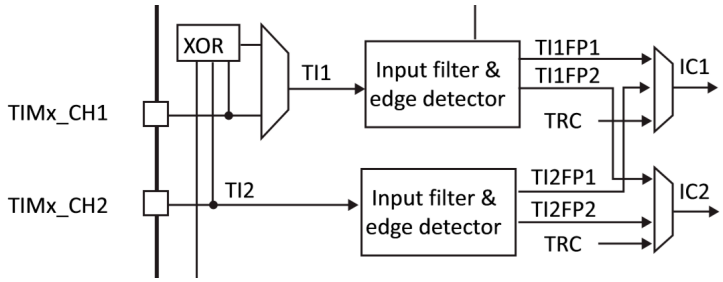
Tab. 3 Časovače s podporou modu Encoder v použitých MCU

Princip činnosti časovače TIMx v módu enkodéru a nastavení příslušných registrů

Čítač sleduje vstupní signál na 2 vstupních kanálech označených jako TI1 a TI2, které jsou propojeny s příslušnými vstupními piny čítače TIMxCH1 a TIMxCH2 viz obrázek 18. Tyto piny je třeba nejdříve nastavit. Registry pro nastavování pinů se mezi rodinami STM32 liší, ale je potřeba se ujistit, že máme nastaveny pro příslušné piny časovače tyto parametry:

- Zapnutý hodinový signál pro příslušnou vstupně výstupní bránu(Port) v příslušném registru RCC(Reset and clock control).
- Nastavení správné alternativní funkcie TIMx_CH1 a TIMx_CH2 pro příslušné piny.
- Pokud je snímač s výstupem typu otevřený kolektor viz 2.3.3, je třeba zapnout vnitřní pull-up rezistory pro příslušné piny.

Po nastavení pinů přichází na řadu nastavení registrů zvoleného časovače. Nastavení musí být započato přivedením hodinového signálu znovu v příslušném registru RCC. Časovač mění svojí hodnotu na základě platných změn signálů TI1FP1 a TI2FP2[16, str. 358], které získá zpracováním vstupních kanálů podle obrázku 23

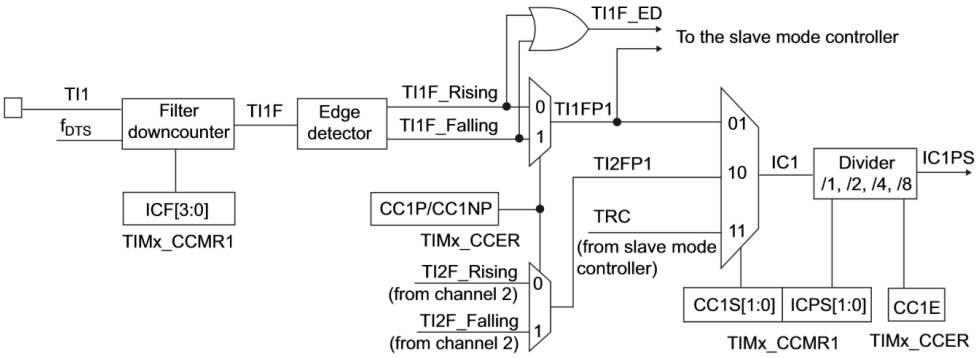


Obr. 23 Vznik TI1FP1 a TI2FP2 řídících signálů ze vstupních pinů TIMx převzato z [15]

Nejdříve je zapotřebí namapovat signál TI1FP1 na vstup TI1 a TI2FP2 na TI2 nastavením CC1S=B01 a CC2S=B01 v registru TIMx_CCMR1 popsaném v 3.1.2. V tomto stejném registru je ještě možné nastavit filtraci vstupního signálu v bitech ozačených ICxF. Nastavováním se ovlivňuje vzorkovací frekvence a počet cyklů, po který musí být vstupní hodnota stejná, aby změna signálu byla považována za platnou. Pokud se není třeba obávat rušení vstupního signálu, je možné filtraci ponechat vypnutou, tedy ve stavu po resetu. Příslušné definice pro nastavení filtrace můžou být dohledány v referenčních manuálech.

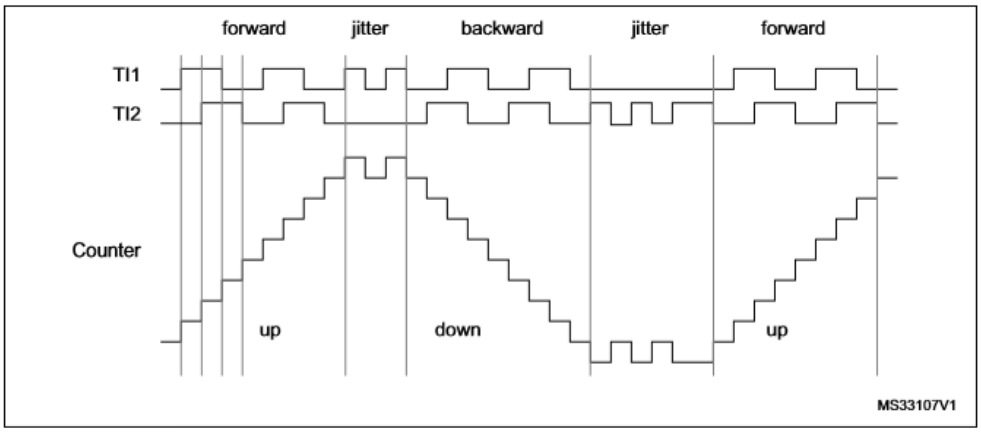
Další na řadě je nastavení polarity detektoru hran v registru TIMx_CCER viz 3.1.2. V tomto registru můžeme různou kombinací nastavení bitů CC1P a CC2P docílit různého směru čítání. Tyto bity určují citlivost na náběžné nebo sestupné hrany na vstupních pinech. Jeden směr čítání je pokud jsou bity polarit CC1P a CC2P stejné a druhý směr čítání je pokud se liší. Dále ještě třeba vstupní piny aktivovat nastavením bitů CC1E a CC2E do hodnoty ('1'). Pokud nevyužíváme zbylé piny periferie k dalším účelům, zbytek bitů registru by měl zůstat ve stavu po resetu ('0').

V tomto bodě je nastavená vstupní fáze zpracování signálů z TI1 a TI2, jež ovlivňují výslednou podobu signálů TI1FP1 a TI2FP2, které pak pokračují do řídící jednotky slave, respektive rozhraní enkodéru. Celá tato fáze je znázorněná na obrázku 24.



Obr. 24 Detailnější pohled na vstupní fázi kanálu Capture/compare [16]

Řídící jednotku periferie nastavíme pomocí registru TIMx_SMCR popsaném v 3.1.2. Změnou bitů SMS(Slave mode selection) nastavíme jeden z módů enkóderů. Pro co největší rozlišení snímání zvolíme mód 3 SMS=B011. V tomto módu rozhraní enkodéru upravuje hodnotu TIMx_CNT při každé změně vstupních kanálů TI1FP1 i TI2FP. Na obrázku 25 je příklad činnosti čítače s neivertovanými vstupy a enkodérovém módu 3.



Obr. 25 Chování čítače enkóderovém módu převzato z [16]

Pro správné fungování je ještě třeba nastavit hodnotu registru TIMx_ARR odpovědného za automatické nulování čítacího registru TIMx_CNT. Tento registr chceme nastavit na nejvyšší možnou hodnotu k dosažení maximálního rozsahu měření. K získání informace, že došlo k přetečení čítacího registru, nastavíme v registru TIMx_DIER generování přerušení při přetečení nastavením UIE bitu - Update interrupt enable. Na závěr aktivujeme čítač nastavením Counter enable bitu(CEN) v registru TIMx_CR1 viz 3.1.2. Na hodnotu '1'.

3.1.4 Shrnutí nastavení registrů časovače TIM2 v STM32F042 pro čtení kvadraturního signálu z enkodéru

Po nastavení přivedení hodinové signálu na periferie čítače a nastavení pinů náležejících čítači v registrech příslušné GPIO brány. Musíme registry pro čítání kvadraturních signálů nastavit následujícím způsobem:

Shrnutí nastavení registrů čítače TIM2 pro enkodér mód

- SMCR: Slave mode control register
  Nastavení požadovaného módu vyhodnocování vstupního signálu pomocí bitů SMS.
  Pro kódování X4 platí SMCR=0x3
- CCMR1: Capture/compare mode register
  Nastavení pinů čítače jako vstupní piny a jejich namapování na řídící rozhraní enkodéru.
  Pro vstupní piny TIM2_CH1 a TIM2_CH2: CCMR1=0x101
- CCER: Capture/compare enable register
  Zapnutí vstupních pinů a možné invertování směru čítání.
  Bez invertování vstupů: CCER=0x11
- ARR: Auto-reload register
  Maximální dosažitelná hodnota pro CNT registr.
  Pro 32-bitový register TIM2: ARR=0xFFFF FFFF

- DMA/Interrupt enable register
  Generování přerušení a DMA transferů.
  Pro přerušení při přetečení CNT registru: DIER=0x1
- CR1: Control register 1
  Spínání čítače: CR1=0x1

3.2 USART periferie v STM32 MCU

USART(universal synchronous asynchronous receiver transmitter) rozhraní slouží pro synchronní nebo asynchronní sériovou komunikaci mezi dvěma zařízeními. Pro účely komunikace mezi MCU a PC se typicky používá asynchronního způsobu přenosu, proto dále bude používáno označení UART. Obecnější označení UART je pak spojováno s přesněji definovanými standardy RS232 a RS485, které popisují protokol a elektrické rozhraní pro komunikaci mezi dvěma zařízeními. Kromě označení pro periferii, která tuto sériovou komunikaci zprostředkovává se UART rovněž používá jako označení pro nízkoúrovňový protokol, kterým se komunikace řídí.

Pro přenos dat se typicky využívá jednoho vodiče pro jeden směr komunikace. S využitím dvou vodičů je možné docílit souběžné obousměrné komunikace(tzv. full duplex). RS232 pak definuje ještě další řídící signály z nichž periferie USART u STM32 podporuje RTS(Request to Send) a CTS (Clear to Send). Pro komunikaci s PC pomocí terminálové aplikace bylo využito pouze signálu Rx a Tx pro datový přenos.

3.2.1 Popis komunikačního protokolu UART

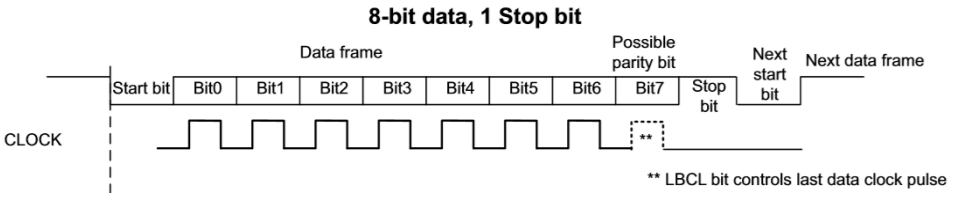
Jak již bylo zmíněno, komunikace probíhá sériově, kdy dochází k přenosu jednoho znaku (paketu) z vysílacího výstupu Tx z jednoho UART rozhraní na vstup (Rx) druhého UART rozhraní. Tato komunikace definuje několik parametrů, kdy pro správné fungování musí být obě komunikační rozhraní nastavena shodně. Prvním z parametrů je přenosová rychlost (baud rate), která odpovídá počtu symbolů(pulzů) přeneseným za jednu vteřinu. U STM32 zařízení rychlost baud rate stanovují dva parametry, a to frekvence hodinového signálu vstupujícího do periferie f_{CLK} a hodnota USARTDIV uložena v Baud rate register (USART_BRR). Výpočet přenosové rychlosti je závislý na zvoleném převzorkování(oversampling) a i na konkrétním zařízení. Tím pádem je třeba příslušný vzorec dohledat v příslušném referenčním manuálu. Pro STM32F042 s 16ti násobným převzorkováním platí:

$$\text{Baud rate} = \frac{f_{CLK}}{\text{USARTDIV}} \tag{7}$$

USARTDIV je v tomto případě 16 bitové bitové číslo s pevnou řádovou čárkou(fixed point).[16] Je patrné, že může nastat situace, kdy pro určité frekvence $_{CLK}$ nebude možné nastavit některé typické hodnoty přenosových rychlostí úplně přesně. Nicméně protokol UART má při použití převzorkování určitou toleranci v řádech procent pro rozdílné frekvence mezi vysílačem a přijímačem.

Další parametry přenosu se zabývají tvarem samotné posílané zprávy(packetu). Těmi parametry jsou počet datových bitů v jednom přenosu, parita a počet stop bitů. Parita může pomoci odhalit chybu v přenosu. K posílaným datům se přidá ještě takzvaný paritní bit, který nastaví hodnotu, aby celkový počet posílaných bitů v logické úrovni '1' byl buď lichý (odd) nebo sudý(even). V klidovém stavu je sběrnice ve stavu logické

'1', poté následuje start-bit logická nula a poté probíhá další vysílání podle obecného schéma na obrázku 26.

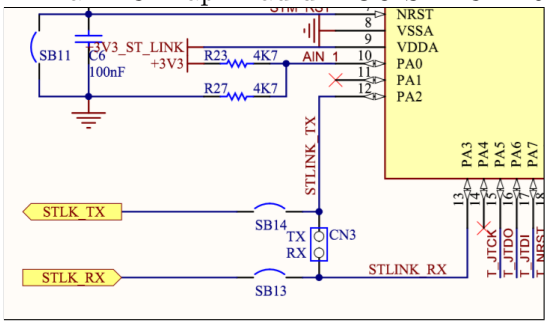


Obr. 26 Diagram poslání jedné zprávy pomocí protokolu UART [16]

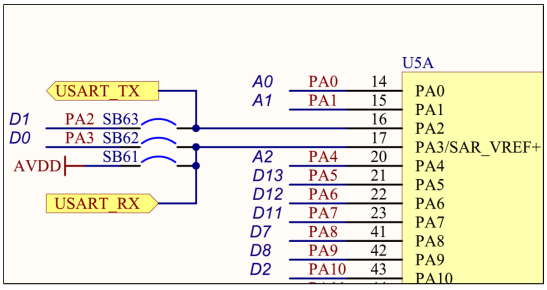
3.2.2 UART komunikace v STM32 mikrokontrolérech

S použitím grafického rozhraní software pro inicializaci periferií STM32CubeMX a použití HAL knihoven je velmi jednoduché rychle zprovoznit komunikaci pomocí této periferie i bez znalosti příslušných řídících registrů. Navíc můžeme snadno nastavit pokročilé funkce této periferie jako využití DMA(Direct Memory Access) transferu. Při využití DMA u UART komunikace, přiřazujeme UART periferii adresový prostor, kde se nachází data pro odeslání nebo kam má naopak přijatá data ukládat. Samotné přijímání znaků a jejich odesílání potom probíhá, bez závislosti na MCU, které se může věnovat běhu zbytku programu. Po odeslání všech dat ze zvoleného adresového prostoru respektive po přijetí definovaného počtu znaků, nás potom periferie informuje o dokončení činnosti pomocí přerušení.

Používáme-li vývojové desky Nucleo od STMicroelectronics, můžeme v některých případech využít vestavěného USB-UART převodníku, který je součástí ST-Linku. V usermanualu k dané vývojové desce můžeme zjistit, jestli tuto možnost deska podporuje, a která z UART periferií daného čipu je s ST-Linkem spojená. Například v manuálu pro vývojovou desku Nucleo-64 se dozvíme, že piny označené USART_TX a USART_RX jsou propojeny s piny ST-Linku STLK_RX a STLK_TX. Na obrázcích 27 a 26 jsou části elektrického schématu, ze kterého můžeme vyčíst, že piny námi zvolené UART periferie buď můžeme externě připojit na konektor CN3 nebo v případě přítomnosti spojek SB13 a SB14 můžeme komunikovat s USART periferií, která má přiřazené piny PA2 a PA3. Například u MCU STM32F103 je k těmto pinům připojen USART2.



Obr. 27 Piny ST linku pro UART-USB převod z [17]



Obr. 28 Propojení pinů PA2 a PA3 s ST-Linkem [17]

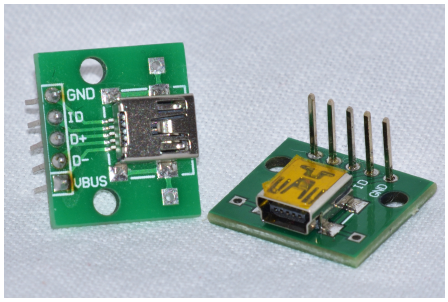
3.3 Full speed USB komunikace s STM32 MCU

Podpora USB je velmi rozšířená mezi rodinami MCU řady STM32. Jedním ze způsobů jak tuto periferii využít je komunikace mezi MCU a počítačem. USB zařízení jsou rozdělena do různých tříd. Pro komunikaci je určena třída USB CDC (Communication device class. Tato třída zařízení se při použití dostupných ovladačů pro VCP(Virtual COM port) jeví v PC jako připojené zařízení k sériovému portu a můžeme tedy navázat sériovou komunikaci pomocí dostupných aplikací emulující sériový terminál. Tyto aplikace jako TeraTerm, Putty, xterm jsou dostupné pro většinu operačních softwarů.

Komunikační protokol USB je velmi komplexní. Naštěstí si můžeme znovu usnadnit práci díky možnosti využití automatického generování kódu s využitím nástroje STM32CubeMX a poskytovanými knihovnami pro USB od STMicroelectronics. Pro základní zprovoznění komunikace mezi zařízením a počítačem se poté tímto komunikačním protokolem není třeba zabývat. V dalších odstavcích budou shrnuty minimální znalosti potřebné pro úspěšné zavedení komunikace mezi PC MCU z rodiny STM32 pomocí USB. Pro vytvoření následujícího textu bylo vycházeno z videoprezentace na toto téma [18].

3.3.1 HW nároky pro navázání komunikace mezi PC a MCU s USB FS

Zařízení a PC se propojují pomocí USB kabelů s vybranými USB konektory. Tyto konektory pro verze USB 1.1-2.0 mají typicky 4-5 pinů a některé vývojové desky již mají USB konektor připojený k určeným pinům a připravený rovnou k použití. Pokud tomu tak není, je možné použít libovolný tvar USB konektoru, který je k dispozici. Typicky rozlišujeme konektory pro hostitelská zařízení- typ A a konektory pro koncová zařízení typy B. Na fotografii 29 jsou v této práci používané konektory typu USB 2.0 Mini B.



Obr. 29 Konektor USB 2.0 Mini B používaný pro koncová zařízení. [8]

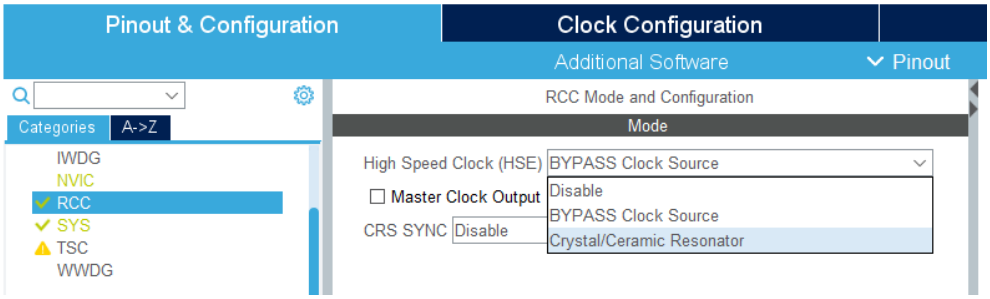
Datový přenos probíhá pomocí diferenciálního signálu po dvou vodičích jejichž piny se označují jako `D+` a `D-`. Dále je možné využít 5V napětí na pinu `VBUS` a je třeba propojit zemnící pin `GND` s příslušným pinem na našem MCU nebo vývojové desce. Dle specifikace pro full speed USB je potřeba na straně zařízení připojit k pinu `D+` pull-up rezistor o hodnotě 1.5 KΩ. Některá zařízení z rodiny STM32 mají tento pull-up rezistor již vestavený a stačí ho aktivovat nastavením příslušného registru, ale jiná ne. V tabulce je uveden přehled v této práci používaných zařízení a přítomností pull-up rezistorů. Podrobnější přehled lze nalézt v [19].

MCU device	Vestavěný pull-up rezistor na D+ pinu	Možnost použití bez krystalového oscilátoru
STM32F042	Ano	Ano
STM32F103	Ne	Ne
STM32F303	Ne	Ne
STM32F411	Ano	Ne
STM32L152	Ano	Ne

Tab. 4 Přítomnost vestavěného pull-up rezistoru pro D+ v použitých MCU

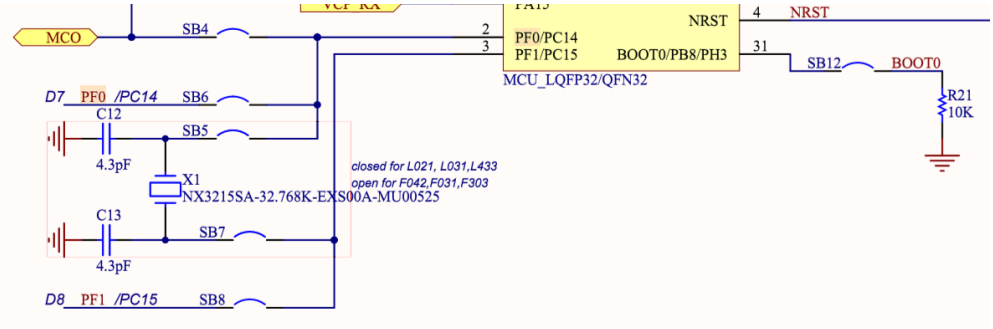
3.3.2 Nastavení hodinového signálu pro USB periferii

Pro správné fungování USB periferie je potřeba mít dostatečně přesný 48MHz zdroj hodinového signálu. Na výběr je několik možností závisejících na dané použité vývojové desce a použitém MCU. Typicky se využívá externího vysokorychlostního zdroje hodinového signálu HSE(High-Speed External) poskytovaného buď krystalovým oscilátorem zapojeným mezi 2 vstupními piny MCU nebo jiného přesného zdroje hodinového signálu přivedeného na jeden definovaný pin (bypass). Bypass se například využívá u vývojových kitů Nucleo, kde se sice nachází externí krystalový oscilátor, ale není připojen k programovanému MCU. Místo toho je oscilátor připojen k integrovanému ST-Linku. Jako přesný zdroj hodinového signálů pro HSE se pak používá výstup MCO z ST linku.



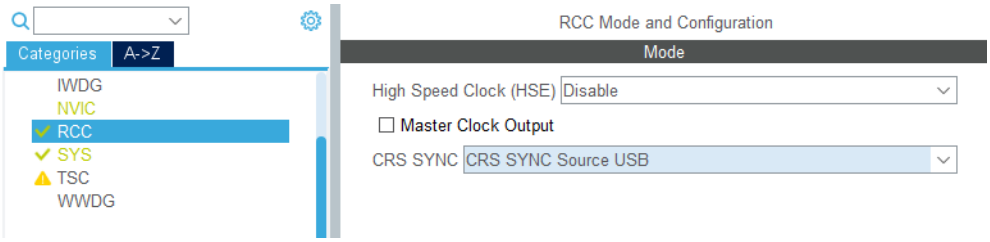
Obr. 30 Výběr zdroje signálu HSE v aplikaci STM32CubeMX

Z HSE signálu je poté pomocí obvodu PLL vytvořen zdrojový hodinový signál náležité frekvence rozváděný do dalších částí MCU. Pro použití externích zdrojů hodinového signálu je zapotřebí se obeznámit se schématem zapojení používané vývojové desky. Piny pro zdroj externího hodinového signálu mívají různé konfigurace měněné pomocí propojek (Solder bridge SB). Je potřeba se ujistit, že na vstupních pinech pro HSE je připojené to, co očekáváme. Na obrázku 31 je ukázka z elektrického schématu pro vývojový kit Nucleo 32. Přítomnost či absence propojek SB rozhoduje o HW konfiguraci pinů PF0 a PF1 využívané pro u STM32F042 pro přivedení HSE signálu.



Obr. 31 Možnosti HW konfigurace pinů PF0 a PF1 na Nucleo32 MB1180 [20]

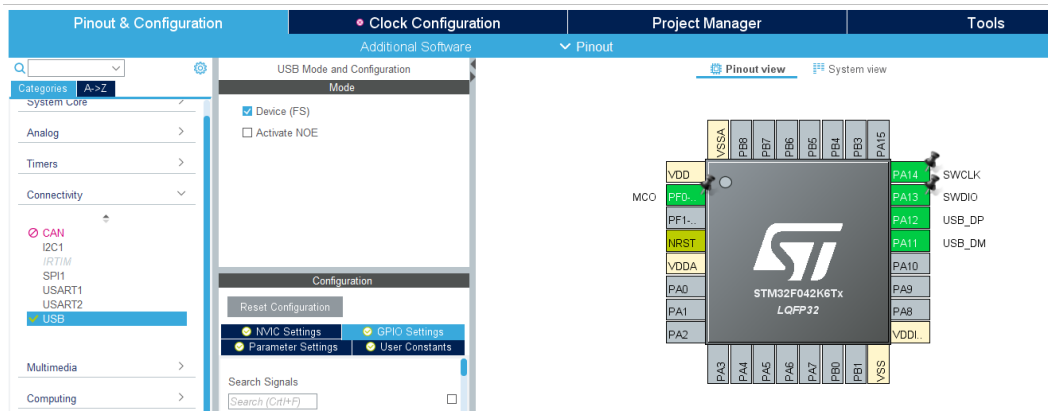
Některá zařízení jako například STM32F042 mají vestavěný speciální RC oscilátor HSI48 který poskytuje 48MHz hodinový signál. Jeho přesnost bohužel není dostatečná a signál se musí ještě dodatečně synchronizovat. Pro tento účel může být použito samotného USB datového toku, konkrétně SOF (Start of frame) paketu vysílané hostitelským zařízením(typicky PC). Druhou možností je synchronizace HSI48 pomocí externího LSE krytalového oscilátoru. Synchronizaci obstarává systém CRS(Clock recovery system) a nastavuje v STM32CubeMX pod záložkou RCC(Reset and Clock Control).



Obr. 32 Nastavení CRS synchronizace HSI48 pomocí signálů z USB kanálu

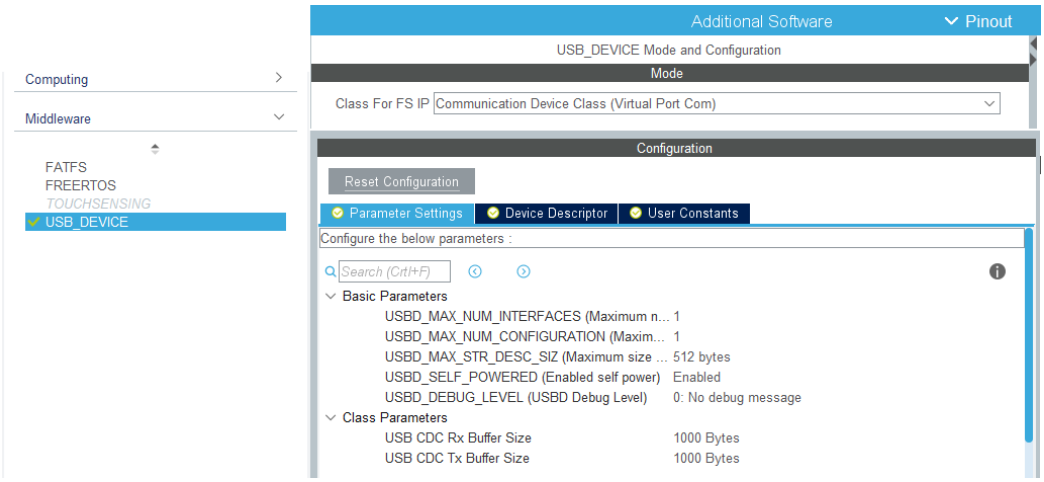
3.3.3 Nastavení USB periferie pomocí STM32CubeMX

Zprovoznění komunikace mezi počítačem a MCU prostřednictvím USB je při použití softwaru STM32CubeMX velmi jednoduché. Zprovoznění je předvedeno na MCU STM32F042, ale postup je u ostatních MCU řady STM32 prakticky stejný. Informace jak založit nový projekt se nachází v dokumentu user manuál UM1718 od STMicroelectronics[21]. Po založení projektu zvolíme v seznamu dostupných periferií kategorii `Connectivity` položku USB a zaškrtneme volbu `Device(FS)`. Tímto se aktivují piny přiřazené USB, v tomto případě `PA12` a `PA11` označené jako `USB_DP` a `USB_DM`. Tato část konfigurace je znázorněn na obrázku 33.



Obr. 33 Konfigurace USB v STM32CubemMX krok 1

Dále v kategorii `Middleware` zvolíme možnost `USB_DEVICE` a vybereme pro zařízení již zmiňovanou třídu `Communication Device Class`. Zbytek nastavení můžeme ponechat v základní podobě nabízené STM32CubeMX. Na obrázku 34 je toto nastavení ukázáno. Nakonec je doporučeno ještě nastavit větší množství pro heap část paměti například 0x600 Bytů.



Obr. 34 Konfigurace USB v STM32CubemMX krok 1

3.3.4 Vytvoření SW pro komunikaci s PC prostřednictvím USB

Po vygenerování kódu při splnění postupu z předešlé kapitoly se do projektu importuje USB knihovna a několik zdrojových souborů. Inicializací USB periferie se zabývá `usbd_conf.c` zatímco inicializace knihoven pro práci s USB je popsána v souboru `usb_device.c`. USB deskriptory popisující vlastnosti USB zařízení přednastaveny v STM32CubeMX jsou dostupné v `usbd_desc.c`. Při používání tohoto automaticky generovaného software pro USB zařízení je uživatel zodpovědný za implementaci funkcí v souboru `usbd_cdc_if.c` definující fungování rozhraní zvolené třídy. [21, str. 318]

Všechny změny v automaticky generovaném kódu zmiňované v této kapitole se provádí ve zdrojovém souboru `usbd_cdc_if.c`. Pro správné fungování komunikace je po-

```
33  /* USER CODE BEGIN PV */
34  /* Private variables ---------------------------------------------------------*/
35  USBD_CDC_LineCodingTypeDef LineCoding = { .bitrate = 115200,
36                                            .format = 0,      // 1 stop bit
37                                            .paritytype = 0, // No parity used
38                                            .datatype = 8 }; // 8 data bits
39  /* USER CODE END PV */
```

Obr. 35 Inicializace datové struktury pro uchovávání parametrů přenosu

třeba, aby si zařízení a terminál byly navzájem schopny předávat parametry sériové komunikace nazývané také 'Line coding'. K nastavení těchto údajů, mezi kreré patří například přenosová rychlost či parita, slouží řídící povely CDC třídy `CDC_SET_LINE_CODING` a `CDC_GET_LINE_CODING` zpracovávané ve funkci `int8_t CDC_Control_FS()`. Parametry komunikace ukládáme do knihovnou definované struktury typu `USBD_CDC_LineCodingTypeDef`, kterou si můžeme libovolně pojmenovat a je definovaná dle tabulky 5.

Offset [B]	Datový typ	Název	Význam	Hodnoty
0	uint32_t	bitrate	Přenosová rychlost	
4	uint8_t	format	Počet stop bitů	0- 1 stop bit 1-1.5 stop bitů 2-2 stop bity
5	uint8_t	paritytype	Parita	0 - bez paritního bitu 1- lichá parita 2-sudá parita
6	uint8_t	datatype	Počet datových bitů	5,6,7,8,16

Tab. 5 Definice struktury pro uchovávání parametrů přenosu.

Strukturu je potřebné v `usbd_cdc_if.c` inicializovat, proto je potřeba přidat řádky dle obrázku 35. Dále je zapotřebí dodefinovat zmiňované povely `CDC_SET_LINE_CODING` a `CDC_GET_LINE_CODING`, které kontrolují kopírování parametrů přenosu mezi počítačem a datovou strukturou našeho zařízení viz obrázek 36. Těmito úpravy je konfigurace USB pro komunikaci s PC hotová.

Funkce `int8_t CDC_Receive_FS()` je vždy automaticky zavolána, když dojde přijetí nějakých dat a funkce `uint8_t CDC_Transmit_FS()` slouží pro posílání dat. Vstupními parametry obou funkcí je ukazatel na počátek přenášených dat a počet přenášených znaků. V nejjednodušší podobě využití, kdy zařízení pouze posílá přijaté znaky zpátky poté vypadají tyto funkce jako na obrázku 37.

```
219
220   case CDC_SET_LINE_CODING:
221     LineCoding.bitrate = (uint32_t) (pbuf[0] | (pbuf[1] << 8)
222                                      | (pbuf[2] << 16) | (pbuf[3] << 24));
223     LineCoding.format = pbuf[4];
224     LineCoding.paritytype = pbuf[5];
225     LineCoding.datatype = pbuf[6];
226     break;
227   case CDC_GET_LINE_CODING:
228     pbuf[0] = (uint8_t) (LineCoding.bitrate);
229     pbuf[1] = (uint8_t) (LineCoding.bitrate >> 8);
230     pbuf[2] = (uint8_t) (LineCoding.bitrate >> 16);
231     pbuf[3] = (uint8_t) (LineCoding.bitrate >> 24);
232     pbuf[4] = LineCoding.format;
233     pbuf[5] = LineCoding.paritytype;
234     pbuf[6] = LineCoding.datatype;
235     break;
```

Obr. 36 Dodefinování povelů pro předávání parametrů sériové komunikace

```
267 static int8_t CDC_Receive_FS(uint8_t *Buf, uint32_t *Len) {
268   /* USER CODE BEGIN 6 */
269   CDC_Transmit_FS(Buf,*Len);
270   USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
271   USBD_CDC_ReceivePacket(&hUsbDeviceFS);
272   return (USBD_OK);
273   /* USER CODE END 6 */
274 }
275
276 /**
287 uint8_t CDC_Transmit_FS(uint8_t *Buf, uint16_t Len) {
288   uint8_t result = USBD_OK;
289   /* USER CODE BEGIN 7 */
290   USBD_CDC_HandleTypeDef *hcdc =
291       (USBD_CDC_HandleTypeDef*) hUsbDeviceFS.pClassData;
292   if (hcdc->TxState != 0) {
293     return USBD_BUSY;
294   }
295   USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
296   result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
297   /* USER CODE END 7 */
298   return result;
299 }
```

Obr. 37 Funkce pro posílání a odesílání dat přes USB

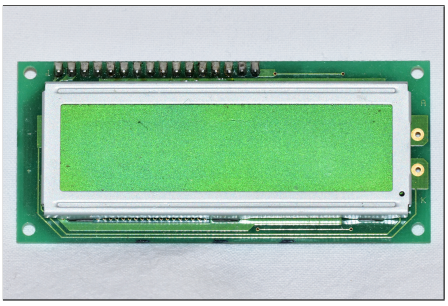
3.4 Zobrazování pomocí segmentových LCD displejů

Součástí indikační jednotky ADP 1 popsané v kapitole 2.4.1 je segmentový displej informující uživatele o průběhu měření. S těmito displeji se můžeme běžně setkat u laboratorní techniky a zařízeních, kde není potřeba složitějšího grafického výstupu. Primárním způsobem pro zobrazování dat o měření, u mnou navrhované indikační jednotky, bude prostřednictvím sériové komunikace zobrazované v terminálové aplikaci počítače. Pro některé případy je vhodné mít ale i možnost lokálního způsobu zobrazování měření, bez nutnosti využití dalšího zařízení, jako je počítač. Z tohoto důvodu je do mých návrhů zahrnuta i varianta indikační jednotky s LCD displejem.

Použití těchto typů displejů s mikrokontroléry usnadňují různé integrované obvodů.

Jedním z takovýchto integrovaných obvodů je řadič HD44780 ulehčující práci alfanumerickými displeji typu LCD, podobným mnou použitému 16x2 pozicovému displeji od firmy Ampire na fotografii 38. Jednotlivé znaky se skládají z 8x5 pixelové mřížky a jejich definice jsou uloženy v paměti řadiče. LCD displej může být napájen 5V nebo 3.3V. Komunikace probíhá typicky pomocí 4 nebo 8 bitové datové sběrnice a 3 řídící signálů. Budeme-li do paměti řadiče pouze zapisovat, můžeme řídicí pin R/W nepřipojovat k MCU, ale natrvalo ho uzemnit. K propojení námi použitého čipu a řadiče displeje poté stačí pouze 6 výstupních pinů. Přehled pinů mnou použitého displeje s řadičem je v tabulce 6. [22]

.



Obr. 38 LCD Modul AC162B od Ampire Co. [8]

PIN	Význam	
1	GND:	Zemnící pin LCD
2	V_{IN}:	Napájecí pin LCD
3	V_O:	Pin pro nastavení kontrastu
4	RS:	Řídí zadávání instrukcí / dat
5	R/W:	Řídí rozlišení mezi čtením/zápisem
5	E:	(Enable) Potvrzení platnosti dat
7-14	DB0-DB7:	Piny datové sběrnice
15	LED_A:	Napájecí pin pro podsvětlení
16	LED_K:	Zemnící pin pro podsvětlení

Tab. 6 Piny LCD s řadičem HD44780

Práce s řadičem poté probíhá v sekvencích, kdy pomocí pinů RS a R/W rozhodujeme, jestli zadáváme do řídící jednotky instrukce nebo probíhá zápis či čtení dat. Platnost dat na pinech se potvrzuje pinem E. Při komunikaci s displejem je zapotřebí dodržet časovací specifikace protokolu komunikace, které mohou být dohledány v příslušném manuálu. Na internetu je k dispozici celá řada knihoven pro usnadnění komunikace s displeji, které mají řadič HD44780. Ve své práci jsem pro použití displeje využil knihovny od Tilen Majerle [23].

4 Popis SW implementace dílčích částí terminálové aplikace

4.1 Seznam použitých vývojových nástroje

4.1.1 STM32CubeMX

Jedná se o grafický nástroj pro STM32 mikrokontroléry umožňující vytváření inicializačních kódů v programovacím jazyce C s využitím Low Layer API(LL), hardwarovou abstraktní vrstvu HAL a dalšími middleware komponenty jako USB, RTOS a další.[21]. Práce s STM32CubeMX nabízí zjednodušení práce při inicializování periférií MCU, názorné grafické rozhraní pro nastavení distribuce hodinového signálů. Díky schopnostem STM32CubeMX dochází k usnadnění přenášení programů mezi různými MCU.

4.1.2 CMSIS

Jedná se abstraktní vrstvu obsluhující hardware pro mikrokontrolery založeny na Arm Cortex procesorech. Její součástí jsou ovladače pro rozhraní periferií nebo například DSP knihovna s různými výpočetními funkcemi, mimo jiné i pro PID regulátory, které bylo využito v rozšířené verzi terminálové aplikace, která obsahovala řízení motoru.

4.2 Využití časovačů k periodickému volání funkcí

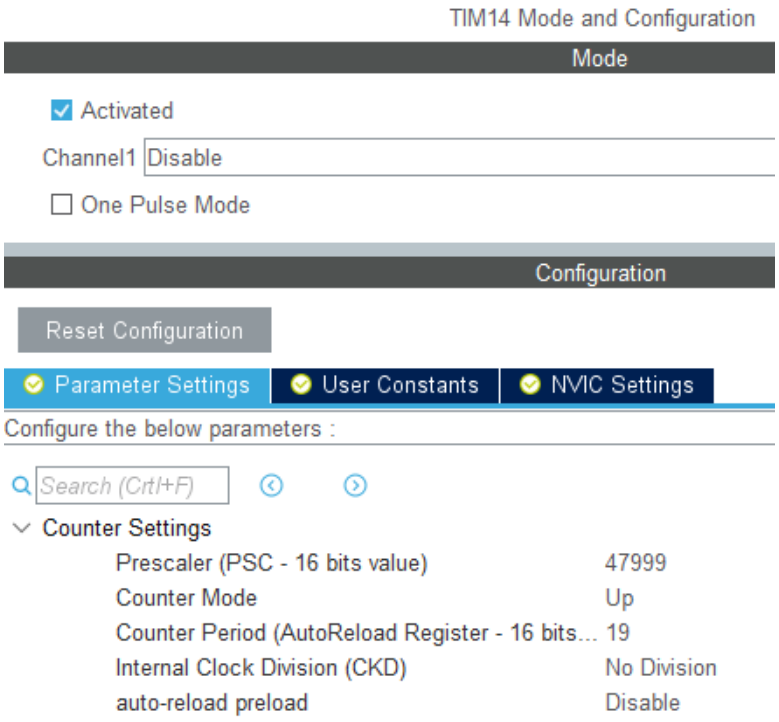
K tomuto účelů nám postačí jakýkoli dostupný časovač. Na obrázku 39 je vidět nastavení časovače TIM16 pro vyvolání události automatické přetečení a s tím spojenou obsluhu přerušení každých 500ms při systémové frekvenci 48Mhz. Po zaškrtnutí aktivace periferie TIM14 (Activated) je třeba nastavit hodnotu PSC děličky vstupního signálu systémových hodin. Požadovaná hodnota se nastaví tak, že vezmeme požadovanou hodnotu, kterou chceme vstupní frekvenci časovače dělit a do registru PSC zapíšeme hodnotu o jedna menší:

$$\text{PSC} = \text{Požadovaná hodnota} - 1 \tag{8}$$

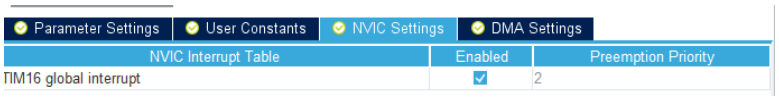
Při nastavené systémové frekvenci 48MHz a nastaveném PSC 9999 dostaneme na vstup časovače hodinový signál s frekvencí 4800Hz. Dále nastavíme periodu čítače v registru ARR, kam zadáváme maximální hodnotu čítače před tím, než čítač vyvolá přerušení, po kterém se čítač vynuluje. Výslednou dobu jednoho cyklu lze poté spočítáme podle vzorce:

$$T_{cyklu} = \frac{(\text{ARR} + 1) \cdot (PSC + 1)}{f_{\text{sys}}} s = \frac{(2399 + 1) \cdot (999 + 1)}{48 \cdot 10^6} s = 500\text{ms} \tag{9}$$

Nakonec je v CubeMX nutno povolit periferii vyvolávání přerušení v záložce NVIC Setting zaškrtnutím možnosti global interrupt enabled viz obrázek40.



Obr. 39 Nastavení periferie TIM v STM32cubeMX



Obr. 40 Povolení vyvolání přerušení pro danou periferi v CubeMX

Ve vygenerovaném kódu, pak nalezneme vytvořenou funkci `MX_TIM16_Init()`, která slouží k nastavení registrů periferie. Je nicméně třeba brát na vědomí, že po zavolání této funkce je časovač pouze inicializován, ale ještě neběží. Po inicializaci ještě musí následovat spuštění, čehož docílíme zavoláním funkce `HAL_TIM_Base_Start_IT( &htim16);` nebo `HAL_TIM_Base_Start( &htim16);`. Na závěr ještě musíme implementovat funkci `void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)`, která je při použití HAL ovladače pro časovače vyvolána pokaždé, kdy některý z časovačů vyvolá obsluhu přerušení, vyvolaném dosažením hodnoty v ARR registru. Jelikož je funkce volaná všemi periferiemi časovačů stejná, tak v případě použití více obvodů časovačů musíme rozlišit, která z periferií přerušení vyvolala. Toho nejjednodušeji dosáhneme porovnáním argumentu funkce htim s handle chtěného časovače. Viz obrázek 41:

```
122 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) {
123   if(htim==&htim16){
124   /**User code**/
125   }
126 }
```

Obr. 41 Naznačení implementace funkce `void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)`

4.3 Měření rychlosti posunu či rotace

Měření rychlosti se dá implementovat dvěma způsoby. Buď měřením času za určitý počet pulzů nebo měřením počtu pulzů za zvolený časový interval. V případě první možnosti narazíme na problém v případě měření rychlosti v obou směrech. Z důvodů vysoké rychlosti enkodérového signálu nelze neustále přesně měřit čas, za který se hodnota counteru změní o jedna. V případě měření doby, za kterou se enkodér posune o více pulzů zase narazíme na problém možnosti změny směru pohybu. Jako lepší řešení se jeví možnost druhá. Tedy měřit, jak se hodnota CNT změní za určitý čas

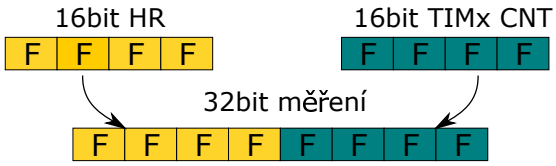
4.3.1 Implementace měření rychlosti pomocí periodického přerušení

S využitím poznatků v4.2 vytvoříme funkci, která každých 500ms načte novou hodnotu čítače enkodéru a uloží starou. Po proběhnutí 2 cyklů už vždy budeme mít uložené 2 po sobě jdoucí stavy registru CNT. Při komplikovaných programech, které pracují s více možnostmi přerušení může nastat konkrétní situace. Místo zavolání měření počtu pulzů v přesně daný čas, dojde ke změření později, z důvodu potřeby vykonání přerušení s vyšší prioritou než je naše měření. Tato situace se dá vyřešit pomocí funkce `HAL_GetTick();`, která vrací čas běhu programu v ms. Tohoto využijeme tak, že pro každou změřenou hodnotu zaznamenáme, kdy byla změřena. V tuto chvíli již máme všechny potřebné informace. Výpis změřené rychlosti je již jen záležitostí aritmetiky, k dostání výsledku v námi preferovaném tvaru.

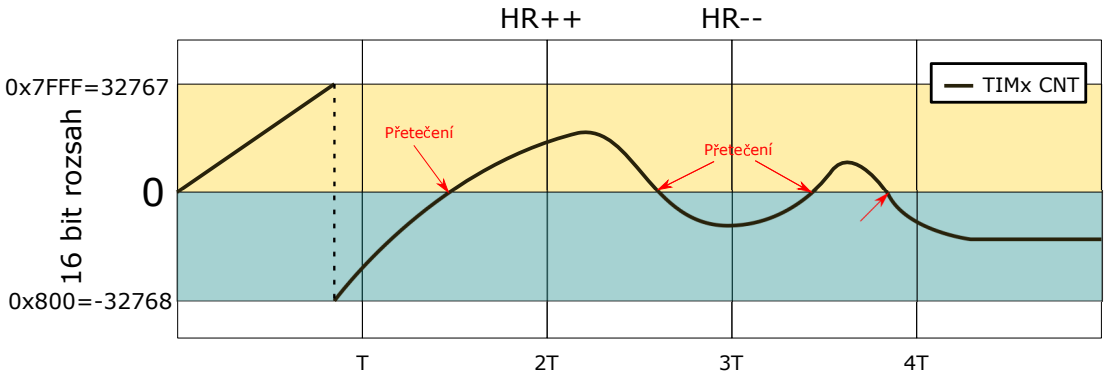
4.4 SW rozšíření měřitelného rozsahu při použití 16bitového HW čítače

Většina čítačů v STM32 mikrokontrolérech je šestnáctibitových. S vysokým rozlišením snímačů jsme potom omezení na měřitelný rozsah, který čítačem můžeme měřit. V případě rozlišení 1 μm na pulz to znamená, že jsme schopní změřit vzdálenost pouze přibližně 6,5 cm než čítač přeteče. Tento problém řeší použití 32-bitových čítačů, nicméně jejich zastoupení je mezi ostatními časovači minimální. Na STM32F042 a STM32F303 je bohužel k dispozici pouze jeden 32 bitový čítač - TIM2. Na jiných zařízeních nemusíte najít žádný.

Z důvodu omezení počtu dostupných čítačů s 32-bitovým registrem, vznikla myšlenka pokusit se nějakým způsobem toto omezení mikrořadičů obejít pomocí vhodného software. Základní princip spočívá ve vytvoření doplňkové 32-bitové proměnné, jejíž hodnota se skládá ze dvou 16-bitových částí. Horních 16 bitů spravuje program pomocí inkrementace a dekrementace proměnné HR a spodních 16 bitů má význam aktuálního stavu 16- bitového registru čítače viz obrázek 42.



Obr. 42 Kombinace dvou 16-bitových proměnných pro vytvoření jedné 32-bitové



Obr. 43 Periodické kontrolování stavu TIMx CNT

Původně se zdála vhodné využít přerušení při přetečení CNT registru čítače pro vyvolání aktualizace proměnné HR. Pro správnou aktualizaci HR musí program ale kromě přetečení TIMx CNT vyhodnotit i směr, kterým se snímač zrovna pohyboval a podle toho rozhodnout, jestli má HR inkrementovat nebo dekrementovat. V případě rychlých změn kolem na hraniční hodnoty čítače, může dojít k nestihnutí dokončení úprav HR před tím, než dojde k přetečení znovu z důvodu změny směru. V tomto případě pak dojde ke znehodnocení měření, protože indikace pak již bude od té chvíle ukazovat nesprávný údaj kvůli špatné úpravě HR. Jako lepší řešení se ukázalo zkombinovat informaci o přetečení registru společně s periodickou kontrolou hodnoty čítače.

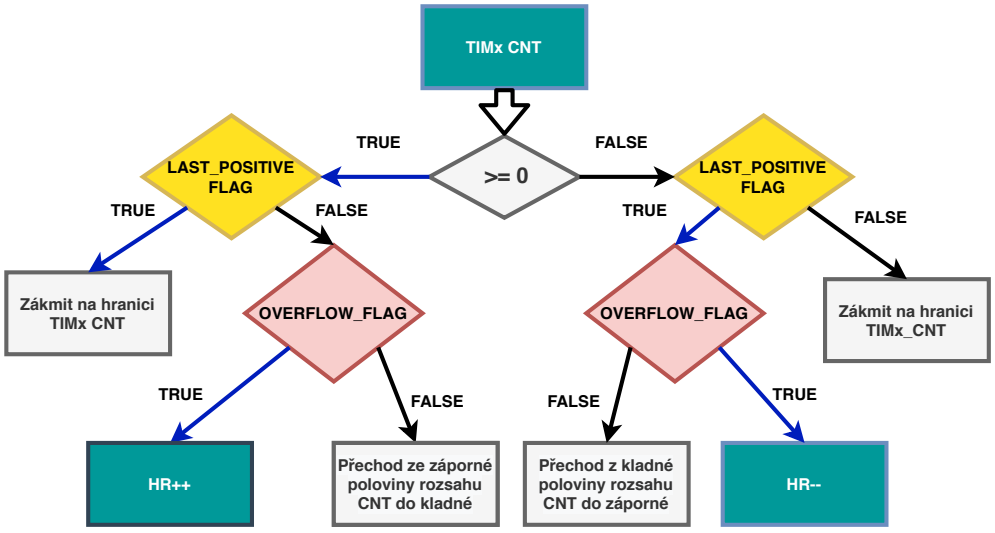
4.4.1 Využití periodického přerušení k rozšíření měřitelného rozsahu

K časovači s 16-bitovým čítačem nastaveném v modu enkódéru podle 3.1.3 je přidán ještě druhý časovač nastavený pro periodické volání funkce dle 4.2. Délka periody poté zásadně ovlivňuje za jakých podmínek toto řešení bude fungovat. Zmíněná periodicky volaná funkce spravuje 16-bitovou proměnnou HR a pomocnou proměnnou LAST_POSITIVE_FLAG, která nám udržuje informaci o to tom jestli jsme se nacházeli v pozitivní nebo negativní polovině 16-bitového rozsahu. Dále ještě je použitá proměnná OVERFLOW_FLAG, která je nastavena při přetečení registru TIMx CNT.

Aby toto implentace fungovala správně, je zásadní správné nastavení času periodického volání funkce. Abychom mohli správně vyhodnotit podtečení/přetečení, musí být perioda volání menší než polovina nejkratšího předpokládaného času, za který je čítač schopný naplnit celý svůj registr. Hledání tohoto času je tedy silně závislé na znalosti toho, v jakém rozlišení budeme měřit. Respektive jak rychle se budou měnit vstupní hodnoty.

Princip metody spočívá v rozdělení 16bitového rozsahu na 2 poloviny. Bylo zvoleno rozdělení 16 bitového rozsahu na kladnou a zápornou polovinu, považujeme-li těchto 16 bitů za celočíselnou proměnnou. Dále pak je jen už využito výše zmíněno předpokladu, že nemůže nastat situace taková, že mezi jednotlivými kontrolami dojde k přetečení registru TIMx CNT a zároveň se posunout o více než polovinu rozsahu. Na obrázku je naznačen průběh měření pomocí 16-bitové čítače v módu enkodéru. Kontroly probíhají s periodou T. K přetečení registru TIMx CNT dochází při překročení nulové hranice. Ke inkrementaci HR za podmínek, předchozí kotrolu se TIMx CNT nacházel v záporné polovině, momentálně se nachází v kladné polovině a zároveň od poslední kontroly došlo přetečení registru TIMx CNT. Na obrázku 43 tato situace nastane v čase 2T. K dekrementaci HR dojde v případě přetečení registru a od poslední kontroly k přesunu

kladné do záporné poloviny. Diagram jakým způsobem je vyhodnocována změna na HR je zobrazen na diagramu 44:



Obr. 44 Funkční diagram pro vyhodnocení změny CNT_H

4.5 Software pro práci s inkrementálními snímači

Pro usnadnění práce s různými druhy snímačů v rámci této práce vznikla knihovna s implementovanými funkcemi jen pro tento účel. Funkce pracují s definovanou strukturou `ENCODER`, která obsahuje řídící, hodnotové a vlastnosti popisující proměnné a ukazatel na adresu registru přidělené HW periferie. Vytvoření takovéto knihovny se zdálo vhodné pro zvýšení přehlednosti kódu a tím jednodušší zužitkování výsledků práce v dalších projektech.

4.5.1 Popis řídící struktury `ENCODER`

Proměnné spojené s měřením

- `uint32_t value`- aktuální změřená hodnota
- `uint32_t prev_value`- předchozí změřená hodnota
- `uint32_t zero_value`- hodnota považovaná nulovou sloužící pro inkrementální způsob měření

  Formátování výstupu
- `uint32_t decimals`- určuje počet desetinných míst na výstupu
- `UNITS encoder_units`- volí v jakých jednotkách bude výstup

  Řídící proměnné
- `bool HOLD_ON`- pozastavuje aktualizaci měření
- `bool ABSOLUTE_MODE_ON`-rozhoduje mezi inkrementálním a absolutním způsobem měření
- `bool REF_SEEK_ON`- zapíná hledání nulového pulzu pro určení absolutní polohy
- `bool INVERTED`- změnou této proměnné můžeme upravit směr čítání
- `bool OPTIONS_CHANGED`- upozorňuje, že došlo ke změně nastavení enkodéru

  Proměnné popisující vlastnosti snímače a použitého čítače
- `CNT_TYPE counter_type` druh použitého čítače - `CNT_TYPE_HALF` pro 16bitový čítač, `CNT_TYPE_FULL` pro 32 bitový čítač.

- `ENC_TYPE encoder_type;`- možnostmi jsou `LINEAR` a `ROTARY`
- `uint32_t constant`- měřící konstanta použitého snímače popisující počet pulzů na otáčku nebo na jednotku vzdálenosti
- `ENC_MODE encoder_mode` určuje druh kódování viz 2.1.3

Extra proměnné pro měření s 16bitovým čítači

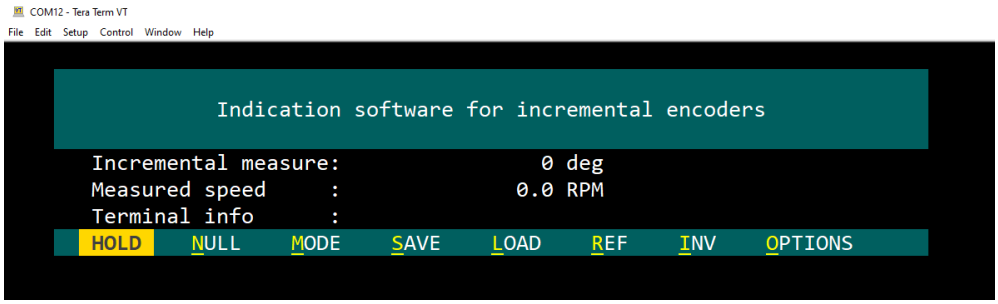
- `uint16_t CNT_HR`- hodnota horních 16 bitů. Používaná k rozšíření měřitelného intervalu v případě použití 16-bitového čítače.
- `bool LAST_POSITIVE_FLAG` viz 4.4
- `bool OVERFLOW_FLAG` viz 4.4

Další

- `TIM_TypeDef *timer_handle` ukazatel na adresu registru použitého čítače
- `double measurement` poslední vypočtená hodnota měření pomocí funkce `encoder_output(ENCODER *encoder_ptr)`

4.5.2 Popis implementovaných funkcí

- `encoder_init(TIM_HandleTypeDef *tim, CNT_TYPE type)`
  Tato funkce přijímá za parametr ukazatel na handler pracující s periferií a údaj, zda se jedná o 16-bitový nebo 32-čítač. Funkce zapíná měření čítače a vrací strukturu pro práci s enkodérem.
- `void encoder_update(ENCODER *encoder_ptr)`
  Kopíruje hodnotu CNT registru do řídící struktury `ENCODER`. V případě použití pouze 16-bitového čítače hodnotu CNT registru kombinuje s hodnotou pro horních 16 bitů `CNT_HR` uloženou v `ENCODER` a spravovanou funkcí `encoder_overflow_control()`.
- `void encoder_HW_update(ENCODER *encoder_ptr, HW_UPDATE update)`
  Slouží k úpravám registru čítače. Přijímá příkazy definované jako výčtový typ `HW_UPDATE`. Pomocí těchto příkazů můžeme měnit směr čítání, nulovat nebo nahrát novou hodnotu do CNT registru čítače a nastavit druh kódování.
- `void encoder_REF_interrupt(ENCODER *encoder_ptr)`
  Určuje chování při zaznamenání nulového pulzu. Tuto funkci by mělo volat externí přerušení spojené s pinem, na který je přiveden signál z enkodérů generující nulový pulz
- `void encoder_overflow_control(ENCODER *encoder_ptr)`
  Řídí hodnotu `CNT_HR`. Tato funkce by měla být periodicky volaná v případě použití 16-bitového čítače. Implementuje rozšíření měřitelného rozsahu popsaného v 4.4.
- `void encoder_overflow_update(ENCODER *encoder_ptr)`
  Zaznamenání do struktury `ENCODER`, že došlo k přetečení registru čítače.
- `double encoder_output(ENCODER *encoder_ptr)`
  Vyhodnocuje měření polohy v závislosti na nastavených parametrech pro použitý snímač. Výsledek ovlivňuje jestli se jedná o lineární nebo rotační snímač, nastavená měřící konstanta, druh kódování a jestli je nastaven výstup v pulzech nebo metrických jednotkách.
- `double dist_counts2units(ENCODER *enc_ptr,uint32_t counts)`
  Funkce funguje podobně jako encoder_output, nicméně pro výstup nepoužívá informace o stavu snímače, ale vypočítává polohu odpovídající zadanému argumentu counts určující počet měřících kroků.
- `uint32_t dist_units2counts(ENCODER *enc_ptr,double distance)` Tato funkce je opačná k funkci dist_count2units. Této funkci předáváme parametr vzdálenosti společně se strukturou enkodéru. Taková hodnota odpovídá 32 bitové hodnotě



Obr. 45 Podoba základní aplikace pro práci s inkrementálními snímači

počtu měřících kroků nebjlíže odpovídající zadané vzdálenosti v metrických jednotkách.

4.6 Terminálová aplikace pro indikační jednotku

Místo specializované PC aplikace pro zpracovávání dat mikrořadiče, jsem využil standardní terminálové aplikace pro sériovou komunikaci, která slouží pouze pro zobrazení znaků posílaných do ní a odesílání stisku klávesnice na zpět. Organizaci programu a zobrazení, pak ale řeší samotný program v mikrořadiči. PC s terminálovou aplikací jako je například TeraTerm nebo RealTerm svou funkcí i ovládáním tak nahradí velkoplošný znakový zobrazovač LCD. Dále v práci pod pojmem "terminálová aplikace"označuji část programu mikrořadiče obstarávající vzhled, organizaci a fungování uživatelského prostředí pro měření, nastavování parametrů snímače a dalších rozšířených funkcí.

Pro komunikaci s indikační jednotkou jsem vytvořil terminálovou aplikaci na způsob textového uživatelského rozhraní TUI (Text user Interface) pomocí ANSI sekvencí. Oproti rozšířenějšímu grafickému uživatelskému rozhraní je výhodou tohoto přístupu bezesporu multiplatformnost. Terminálové emulátory podporující ANSI sekvence existují pro většinu operačních systémů a bývají na nich i často předinstalovány. Díky tomu je velmi snadná přenosnost hotové aplikace mezi systémy a terminálovou aplikaci pro řízení indikační jednotky lze používat na jakémkoli zařízení bez ohledu na operační systém.

4.6.1 Základní podoba terminálové aplikace k indikaci s různými snímači

Nároky na aplikaci byly takové, aby svými možnostmi co nejvíce nahrazovala použití komerčních indikačních jednotek zmiňovanými v kapitole 2.4. Aplikace musela zvládat základní ovládání při měření jako je nulování, pozastavení indikace nebo započatí hledání referenční značky. Také bylo potřeba umožnit práci s různými snímači. Tedy v aplikaci muselo jít nastavit vlastnosti použitého senzoru jako typ snímače a jeho přepočtová konstanta dle kapitoly 2.1.4. Dále bylo potřeba zařídit jednoduché ovládání pomocí klávesnice, tak aby aplikace byla uživatelsky co nejvíce příjemná. Výsledná podoba je vyobrazena na obrázku 45

4.6.2 Ovládání terminálové aplikace

Na obrázku 45 je lišta s ovládacími prvky. K pohybu mezi jednotlivými prvky se používá šipek klávesnice nebo alternativně kláves `J` a `K`. Těchto alternativních kláves je nutno

využít při komunikaci pomocí UART, jelikož aplikace v tomto případě špatně vyhodnocuje přijaté šipky jako klávesu ESC. Důvodem pro toto je, že v případě použití UART aplikace vyhodnocuje komunikaci znak po znaku, ale klávesy šipek generují kódová slova o délce tří znaků. Pro potvrzení zvolení ovládacího prvku pak slouží klávesa ENTER. Pro zjednodušení ovládání můžou tyto hlavní ovládací prvky být také zvoleny přímo a to klávesou náležící prvnímu písmenu chtěné možnosti. Například pro možnost NULL stačí stisknout klávesu N. Tento způsob ovládání je i naznačen zvýrazněním písmen těchto ovládacích prvků. Níže uvádím popis jednotlivých řídicích prvků. Podrobnější informace o jejich fungování jsou uvedeny v 4.6.3.

- HOLD: Pozastavení zobrazované hodnoty, samotné měření ale pokračuje. Po opětovné volbě se výstup znovu začne aktualizovat.
- NULL: Tato volba přepne do inkrementálního způsobu měření a uloží současnou hodnotu čítače jako nulovou pozici.
- MODE: Přepíná mezi absolutním a inkrementálním způsobem měření.
- SAVE:Uloží údaje o měření a nastavení snímače do FLASH paměti.
- LOAD:Obnoví údaje o měření a nastavení snímače, které byli jako poslední uloženy.
- REF: Zahájí čekání na referenční značku, během čekání měřená hodnota bliká. Při opětovné volbě čekání přestane.
- INV: Změna směru čítání.
- OPTIONS: Otevírá rozšířené možnosti nastavení týkající se použitého snímače.

4.6.3 Podrobný popis funkcí terminálové aplikace

Při startu dojde ke kontrole, zda je na vyhrazeném místě v paměti FLASH uložená konfigurace nastavení enkodéru ve tvaru řídící struktury ENCODER popsané v kapitole 4.6.3. Pokud je k dispozici, načte uloženou strukturu a uplatní základní nastavení: absolutní způsob měření, vypnutí pozastavení zobrazovaní(HOLD) a vynulování měření. Pokud chceme načíst i poslední uloženou polohu, můžeme použít ovládací prvek LOAD, který načte uloženou konfiguraci kompletně.

Zobrazovací část aplikace informuje o průběhu měření, rychlosti pohybu snímače a uvádí zpětnou vazbu o řízení programu. Podoba této části aplikace je zobrazena na obrázku 46. Poslední řádek Terminal info poskytuje zpětnou vazbu o běhu aplikace. Objevují se zde informativní sdělení o změně směru čítání, chybové hlášení nebo potvrzovací zprávy o úspěšném uložení parametrů. Na druhém řádku Measured speed je informace o rychlosti pohybu snímače. První řádek je pak věnován samotnému měření polohy.

```
Incremental measure:          116.85 deg
Measured speed     :            24.8 RPM
Terminal info      :  Successfuly saved encoder data.
```

Obr. 46 Zobrazovací část terminálové aplikace

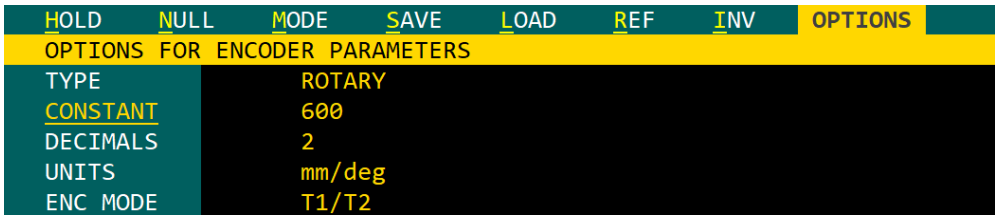
Měření může probíhat ve dvou režimech a to buď v absolutním nebo přírůstkovém(inkrementálním). V případě absolutního způsobu měření se pro určení polohy používá přímo hodnota čítače. V případě inkrementálního měření polohu určuje rozdíl mezi hodnotou čítače a uloženou nulovou polohou. NULL uloží současnou polohu jako nulovou a přepne automaticky do inkrementálního způsobu měření. Zobrazovaná hodnota je tedy vynulovaná, ale v registru čítače se neztrácí informace o absolutní poloze. Mezi

režimy pak můžeme přepínat pomocí MODE.

Absolutní poloha je nejdříve určena polohou při startu jako nulová hodnota. V případě přítomnosti signálu s nulovým pulzem můžeme absolutní polohu vztáhnout k této referenční značce. REF zahájí hledání referenční značky a signalizuje tak blikáním měřené hodnoty po přejetí značky dojde k vynulování měření registru čítače a přepnutí do absolutního způsobu měření. Směr změny polohy můžeme otočit pomocí volby INV. Díky tomu nemusíme upravovat fyzické zapojení snímače v případě, že za kladný směr čítání chceme považovat ten opačný směr než zrovna při daném zapojení máme. Zobrazovanou hodnotu lze zmrazit a znovu obnovit změny pomocí HOLD.

Důležitou součástí aplikace je možnost formátování výstupu s ohledem na použitý snímač. K tomuto slouží rozšířené nastavení aplikace ukryté pod možností OPTIONS. Po zvolení této možnosti se nám zobrazí menu zobrazující momentální nastavení - viz obrázek 47. V levé části panelu se nachází názvy nastavitelných parametrů a v pravé části momentálně nastavené hodnoty. Šipkami či klávesami J a K se pohybujeme kurzorem ukazující na jeden z parametrů. Poloha kurzoru je vyjádřena zvýrazněním názvu parametru. Klávesa ESC toto rozšířené menu schová a vrátí nás zpět na lištu s hlavními ovládacími prvky. Klávesu ENTER naopak použijeme, pokud si přejeme určitý z parametrů upravit.

Nastavené parametry snímače můžeme uložit do FLASH paměti MCU pomocí možnosti SAVE. Do paměti se ukládá celá řídící struktura pro práci s enkodéry popsaná v kapitole . Úspěšné uložení struktury je spojeno s vygenerováním CRC32 kódového slova z uložených dat a jeho následné uložení na definované místo ve FLASH paměti. V případě nahrávání uložené struktury pomocí možnosti LOAD jsou potom data ve FLASH paměti kontrolována, že jsou uložená data přítomná a nedošlo k jejich poškození právě porovnáním CRC32 kódového slova dat v paměti s kódovým slovem již uloženým. V případě nepřítomnosti dat nebo pokud nastane chyba při ukládání tak nedojde k nahrání neplatných dat.



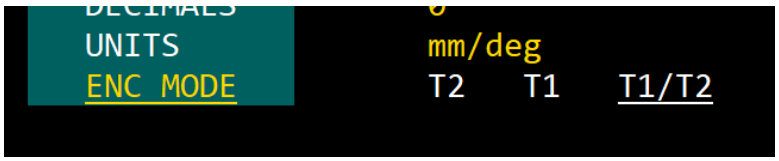
Obr. 47 Ovládací panel pro úpravu vlastností použitého snímače a formátování výstupu

Úprava parametrů poté probíhá ve dvou režimech. V případě, že pro daný parametr je na výběr jen několik možností, tak se nám po stisku ENTER vypíšou jednotlivé možnosti. Polohu kurzoru zde vyjadřuje podtržení. Kurzorem pohybujeme běžným způsobem a volbu opět potvrdíme klávesou ENTER. Druhý režim je používaný pro zadávání číselných hodnot. V tomto případě se vymaže příslušný řádek a zobrazí se psací kurzor. Během zadávání číselných hodnot lze sledovat zadávanou hodnotu na terminál. Opravovat chyby lze pomocí klávesy BACKSPACE. Rozmyslíme-li si zadávání hodnoty, můžeme režim zadávání opustit pomocí klávesy ESC. Pokud jsme spokojeni s napsanou numerickou hodnotou, zadání potvrdíme klávesou ENTER.

Většiny numerických konstant lze zadat jen v určitém rozsahu a formátu. Pokud není dodržen rozsah aplikace, novou hodnotu nepoužije a informuje o tom chybovou hláškou. V případě nedodržení formátu se může stát, že hodnota nebude změněna nebo bude nastavena podle toho jak velkou část zadané hodnoty rozpozná. V jakémkoli případě se ale aktuálně nastavená hodnota zobrazí vedle názvu parametru.

Výpis nastavitelných parametrů a jejich možností nastavení.

- `TYPE`: Zde dochází k výběru jestli používáme rotační snímač `ROTARY` nebo snímač lineární `LINEAR`. Používáme-li výstup v metrických jednotkách, tak tato volba ovlivní jestli výstup bude ve stupních (`deg`) nebo millimetrech(`mm`).
- `CONSTANT`: Přepočtová konstanta je charakteristické číslo snímače vyjadřující jeho rozlišení. Zadáváme ho ve tvaru počtu pulzů jednoho kanálu na jednu celou otáčku v případě rotačního snímače a počtu pulzů jednoho kanálu na 1mm v případě lineárního snímače. Popis přepočtu je vysvětlen v kapitole 2.1.4.
- `DECIMALS`: Tento parametr určuje počet desetinných míst u výpisu naměřené hodnoty. Tento parametr můžeme nastavit v celočíselném rozsahu 0-9.
- `UNITS`: Zde dochází k výběru, jestli výstup bude v metrických jednotkách podle typu snímače- volba `mm/deg` nebo bude výstup pouze v počtu změn
- `ENC MODE`: Režim kvadraturního kódování určuje dle 2.1.3, jaké události budou podmiňovat inkrementaci čítače. Možnostmi jsou změna signálu(náběžná nebo sestupná hrana) na jednom ze vstupních kanálů. `T1` pro změny na vstupním kanálu 1 a `T2` na kanálu 2. Pro maximální rozlišení bychom měli zvolit volbu `T1/T2`, při které dochází k čítání na každé hraně- kódování X4. Viz obrázek 48



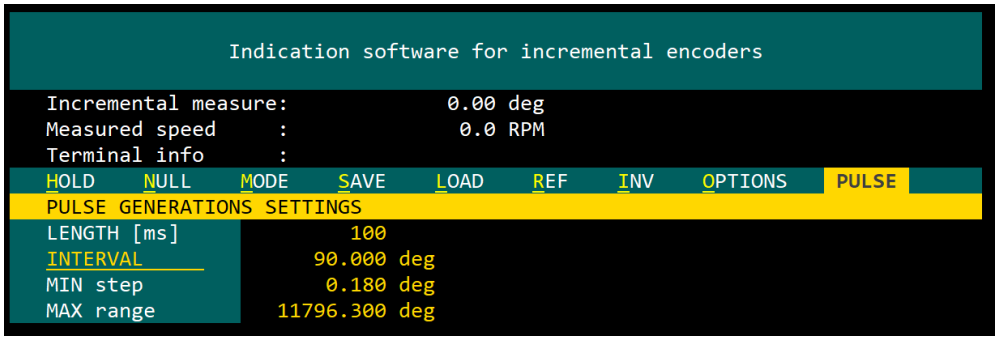
Obr. 48 Volba režimu kvadraturního kódování v řídící aplikaci

4.7 Rozšíření terminálové aplikace pro demonstraci využití snímačů

4.7.1 Generování dráhových pulzů

Samotná číslicová indikace informuje uživatele o stavu měření, nicméně získávaná data mohou být využita i jiným způsobem. V průmyslové automatizaci se často využívá dat z inkrementálních snímačů pro automatické provádění nějakých úkonů v závislosti na poloze snímače. Může tak být například obstarávána nějaká periodicky opakovaná činnost, jako je dávkování materiálu na pohyblivém páse nebo automatické zastavení po dojetí pohyblivého prvku do určité polohy. K těmto a dalším podobným úkonům by se dalo využít dráhových pulzů. Ty spočívají v generování impulsů ve specifikovaných intervalech měřené polohy.

Pro tento způsob využití jsem vytvořil rozšíření terminálové aplikace. HW základem je druhý čítač používaný také pro čtení signálu z používaného enkodéru a časovač nastavený v jedno cyklovém režimu PWM pro generování žádaného pulzu. Na rozdíl od čítače používaného pro měření polohy je registr Timx_ARR pro automatické přetečení



Obr. 49 Ovládací panel pro nastavení generátoru dráhových pulzů

čítacího registru nastaven na hodnotu odpovídající počtu kroků čítače mezi jednotlivými dráhovými pulzy.

Pohyb v menu pro správu generátoru dráhových pulsů je obdobný jako v celé aplikaci. Vzhled menu je ukázán na obrázku 49. Nastavitelnými parametry jsou doba trvání pulzu a velikost dráhového intervalu. Poslední dva řádky jsou pak pouze informativního charakteru, odpovídající vlastnostem použitého snímače a zvolenému formátování výstupu nastavených v záložce `OPTIONS`.

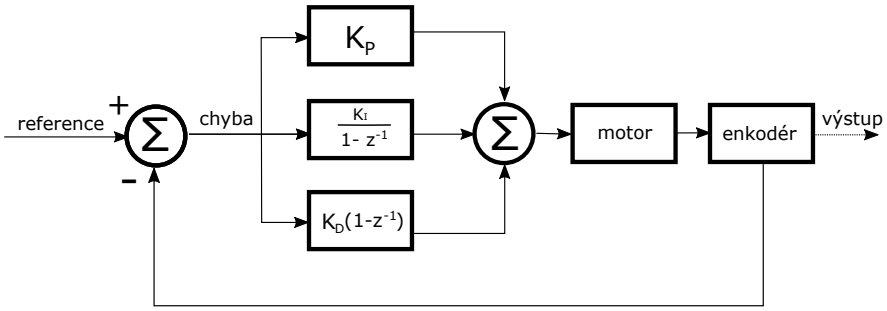
Popis význam jednotlivých řádků a možností nastavení.

- `LENGTH [ms]`: Definuje délku generovaného pulzu v ms. Tato hodnota může být nastavena v celočíselném rozmezí 1-1000 ms.
- `INTERVAL`: Dráhový interval mezi jednotlivými pulzy. Hodnotu nastavujeme ve stejném formátu, který je zrovna nastaven pro výstup měření. Zadána hodnota by měla být v intervalu od hodnoty `MIN step` po hodnotu `MAX range`. Po potvrzení se hodnota automaticky přepočítá, aby odpovídala celočíselné násobnosti minimálního kroku `MIN step`.
- MIN step: V případě používání metrických jednotek tento údaj odpovídá velikosti nejmenšího měřícího kroku, vzhledem k použitému typu vyhodnocování kvadraturního signálu, zadané přepočtové konstantě a typu snímače.
- MAX range: Tento údaj je závislý na stejných parametrech jako `MIN step`. Odpovídá maximálnímu měřitelnému rozsahu s použitím 16-bitového čítače. Zároveň se také jedná nejvyšší možnou hodnotu, kterou můžeme zadat pro délku dráhy mezi dvěma pulzy.

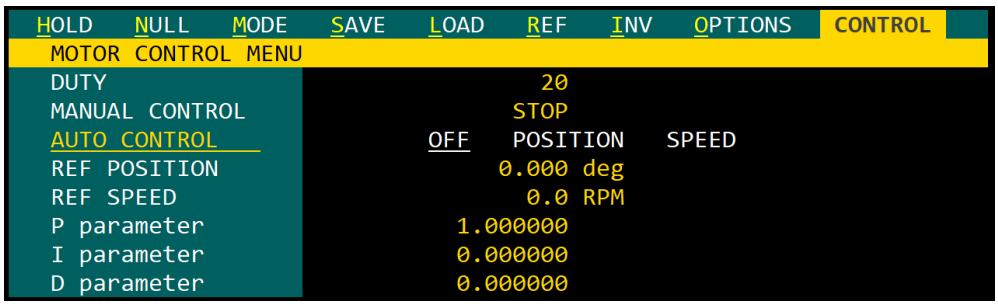
4.7.2 Využití dat z inkrementálního snímače pro regulaci polohy a rychlosti

V robotice a dalších technických disciplínách se využívá inkrementálních snímačů k získání dat pro řízení motorů pomocí zpětné vazby. Aplikací, při kterých využijeme možnost regulovat rychlost nebo polohu nějakého akčního členu, určitě najdeme dostatek. Jedním ze základních způsobů regulace, která je široce uplatnitelná pro různé typy aplikací, je PID regulace. Pro implementaci PID regulátoru jsem využil dostupné knihovny CMSIS DSP, která podporuje funkce pro implementaci PID regulátorů.

PID regulátor přijímá chybovou hodnotu, kterou tvoří momentální výstupní hodnota získaná pomocí enkodéru odečtená od referenční hodnoty. Dále se regulátor pokouší minimalizovat tuto hodnotu nastavením nové řídící hodnoty regulace pro motor. Funkci regulátoru ovlivňují 3 parametry: proporcionální parametr P určuje reakci na aktuální chybu, integrální parametr I reaguje na součet posledních chyb a D parametr reaguje



Obr. 50 Schémea PID regulace motoru s inkrementálním snímačem



Obr. 51 Kontrolní menu pro řízení motoru prostřednictvím PID regulátoru.

na rychlost změny chybového vstupu do regulátoru. [24]. Viz schéma na obrázku 50.

Ovládání motoru je řešeno pomocí H můstku a 2 řídících PWM signálů z časovače periferie TIM. Je možné měnit směr pohybu motoru podle toho, který z kanálů PWM je aktivní. Nastavováním střídy PWM signálu se upravuje rychlost, kterou se motor pohybuje. Pro účely práce s pohonem jsem vytvořil pro terminálovou aplikaci další rozšířené menu. Tato část aplikace byla navrhována pro umožnění co nejširší podpory práce s motory doplněnými inkrementálními enkodéry.

Prostřednictvím tohoto rozšíření můžeme manipulovat s motorem v manuálním režimu, kdy nastavíme střídu a směr otáčení. Dále je možné řídit motor tak, aby se automaticky dostal do referenční pozice a v té pak zůstal nebo můžeme využít PID regulace pro udržování konkrétní nastavené rychlosti. Správné parametry PID regulace si musí uživatel zjistit pro svůj motor sám.A je potřeba mít na paměti, že pro regulaci polohy budou potřeba jiné PID parametry než pro řízení rychlosti.

Popis význam jednotlivých řádků a možností nastavení CONTROL MENU.

- `DUTY [%]`:
  Definuje střídu PWM signálu pro manuální ovládání rychlosti.
- `MANUAL CONTROL`:
  Prostřednictvím vybrání jedné z možností tohoto menu manuálně ovládáme směr pohybu motoru. Na výběr jsou možnosti:`OFF`-motor je vypnutý `POSITIVE`-motor se pohybuje v kládném směru otáčení NEGATIVE-motor se pohybuje záporném směru otáčení.
- `AUTO CONTROL`:
  Zde se zapíná režim regulace na výběr jsou možnosti: `OFF`- automatická regulace vypnutá, `POSITION`-regulace polohy, `SPEED`- regulace rychlosti.

- `REF POSITION` / `REF SPEED` referenční hodnoty pro regulátor.

- `P,I,D parametres` paramatery definující vlastnosti PID regulátor.

Automatická regulace byla měla být zahájena zadáním PID parametrů a zvoleného referenčního parametru. V případě pokusu o upravení PID parametrů, totiž dojde k vypnutí automatické regulace včetně motoru. Referenční pozice je v paměti uložená jako počet měřících kroků. Při zadávání v metrických jednotkách dojde k přepočtu zadané hodnoty na nejbližší možnou hodnotu odpovídající počtu měřících kroků za použití definovaného snímače. Při ukládání parametrů snímače do paměti FLASH dojde zároveň k uložení PID parametrů - bez referenční pozice a rychlosti.

4.8 Metodika pro využití ukázkových aplikací v různých mikrořadičích STM32

Pro lepší využitelnost výsledků této práce tato kapitola ukáže pracovní postup pro realizaci indikačních jednotek na různých čipech řady STM32. Pro usnadnění bude využito automatického generování kódu pro inicializaci periferií prostřednictvím již zmiňovaného STM32CubeMX a bude plně využito HAL knihoven pro co největší možnou univerzálnost tohoto návodu. Díky tomu by mělo být pro případné zájemce snadné vytvořit funkční základní variantu indikace pro jakýkoli čip z řady STM32. Tato základní varianta by pak mohla posloužit jako základ pro další rozšiřování aplikace vyplývajících z požadavků nově vznikajících projektů.

4.8.1 Inicializace periferií STM32F303RE

Po zvolení zařízení a vytvoření projektu v STM32CubeMX přichází na řadu základní nastavení vestavěných periferií a definování hodinových signálů. Celý pracovní postup budu ukazovat na zařízení STM32F303RE, v mém případě integrovaném na vývojové desce NUCLEO-64. Jednou z výhod těchto desek je přítomnost USB-UART převodníku v modulu ST-LINKU. Díky tomu můžeme využít rozhraní UART pro komunikaci s PC. K základní funkci indikační jednotky potřebujeme pouze tyto periferie:

- Komunikační rozhraní: typicky UART nebo USB. Případně můžeme použít i další způsoby sériové komunikace pokud to situace vyžaduje.
- 2 obvody TIM: Jeden podporující práci s enkodéry a druhý pro periodicky volané funkce programu
- 1 pin externího přerušení pro čtení nulového pulzu.
- Výpočetní jednotku CRC - přítomnou v STM32 MCU.

K této práci jsou přiložené vzorové projekty pro ukázání správné konfigurace periferií. Nastavení časovače pro periodické volání funkcí je detailněji věnovaná kapitola 4.2. Libovolný časovač nastavíme pro opakování s periodou 20ms. Seznam periferií TIM s podporou zpracování signálu z enkóderů je pro některé mnou používané MCU uveden v kapitole 3.1.3. V této kapitole je také podrobněji popsáno chování TIM v tomto módu a jakým způsobem periferii nastavit, aby fungovala požadovaným způsobem. Pro zpracování signálu nulového pulzu nastavíme pin externího přerušení tak, aby byl citlivý na náběžnou hranu vstupního signálu. V STM32CubeMX nastavíme tomuto pinu označení REF. A CRC periferie musí být nastavena pro přijímání dat délky 32bitů.

Komunikaci s PC bude v tomto případě zajišťovat periferie USART2 v asynchronním módu. V STM32CubeMX zvolíme pro tuto periferii generování přerušení, protože

```
30 /* Private includes ----------------------
31 /* USER CODE BEGIN Includes */
32 #include "STM32_encoder.h"
33 #include "terminal_functions.h"
34 #include "user_functions.h"
35 /* USER CODE END Includes */
```

Obr. 52 Ukázka ohraničení uživatelských částí kódu komentáři

příkazy z klávesnice budou přicházet nepravidelně a kdybychom používali tzv 'Polling mode' způsob příjmu dat, tak bychom zbytečně zaměstnávali procesor čekáním. Parametry přenosu je možné si zvolit dle preferencí, ale pro vyšší přenosové rychlosti jako 115200 Bits/s. Terminálová aplikace je napsána tak, aby pracovala s UART periferií pomocí DMA a je třeba ji příslušným způsobem pro toto nakonfigurovat i v STM32CubeMX. Pro generování časové základny je vhodné využít externí oscilátor, stejně jako je to nutné v případě využití USB. Zdrojům hodinového signálu se více věnuji v kapitole 3.3.2.

4.8.2 Popis importu knihoven a inicializace potřebných proměnných

Do vytvořeného projektu importujeme mnou vytvořené zdrojové soubory a jim odpovídající hlavičkové soubory. Jedná se o tyto soubory, které mohou být zkopírovány ze vzorového projektu přiloženého k této práci:

- `terminal_functions.c`
  Obsahuje definici řídící struktury terminálové aplikace a popisuje funkce s ní pracující. Dále jsou zde popsané funkce pro komunikaci s uživatelem.
- `print_functions.c`
  Obsahuje definice funkcí pro textový výstup do terminálové funkce.
- `user_functions.c`
  Obsahuje funkce pro ukládání do FLASH paměti
- `STM32_encoder.c`
  obsahuje definice řídící struktury enkodéru a funkcí s ní pracující.

Po přidání potřebných souborů do projektu pokračujeme úpravou nejdříve hlavního zdrojového souboru `main.c`. Pro vzor úprav použijeme přiložený vzorový projekt Basic_Indication vytvořený pro procesor STM32F042. Začneme zkopírováním částí kódu ohraničenými komentáři pro uživatelský kód, vytvořenými automaticky pomocí STM32CubeMX. Viz obrázek:

Další na řadu přichází přiřazení použitých periferií. Zde je znovu využito vlastností STM32CubeMX a HAL, které pro definované periferie automaticky vytvoří řídící struktury. V programu je toho pak využito vytvořením ukazatelů na tyto řídící struktury, se kterými potom pracuje zbytek programu a při použití periferie stačí přepsat pouze definování tohoto ukazatele. Ukazatelům přiřadíme adresy řídících struktur periferií, které si můžeme libovolně zvolit a které byly inicializovány pomocí STM32CubeMX viz obrázek 53. V tomto případě bylo využito 32-bitové TIM2 pro práci s enkodérem, TIM6 pro periodická přerušení a USART2 periferie pro komunikaci.

```
53 /* USER CODE BEGIN PV */
54 TIM_HandleTypeDef *encoder_timer_pointer = &htim2;
55 TIM_HandleTypeDef *interrupt_timer_pointer = &htim6;
56 CRC_HandleTypeDef *CRC_pointer = &hcrc;
57
58 #ifdef terminal_UART
59 extern char rec_buf[RECEIVE_LEN + 1];
60 UART_HandleTypeDef *terminal_uart_pointer = &huart2;
61 uint8_t uart_char; // variable for storage of received character through UART.
62
63 #endif
```

Obr. 53 Přiřazení adres řídících struktur periferií ukazatelům.

Pro správnou funkci programu, musíme inicializační funkci struktury práce s enkodérem sdělit, jestli je zvolen 16-ti nebo 32-bitový čítač. Toho se docílí upravením druhého argumentu funkce `encoder_init_HAL`. Pro 32-bitový čítač TIM zadáme argument CNT_TYPE_FULL. Terminálová aplikace komunikuje s uživatelem pomocí UART nebo USB, na základě definice v hlavičkovém souboru `main.h`. Pro komunikaci pomocí UART musíme do tohoto souboru přidat definici `TERMINAL_UART` pro komunikaci pomocí USB pak `TERMINAL_USB`. Viz obrázek 54.

```
48 /* Exported macro ---------------
49 /* USER CODE BEGIN EM */
50 //#define terminal_USB
51 #define terminal_UART
52 /* USER CODE END EM */
```

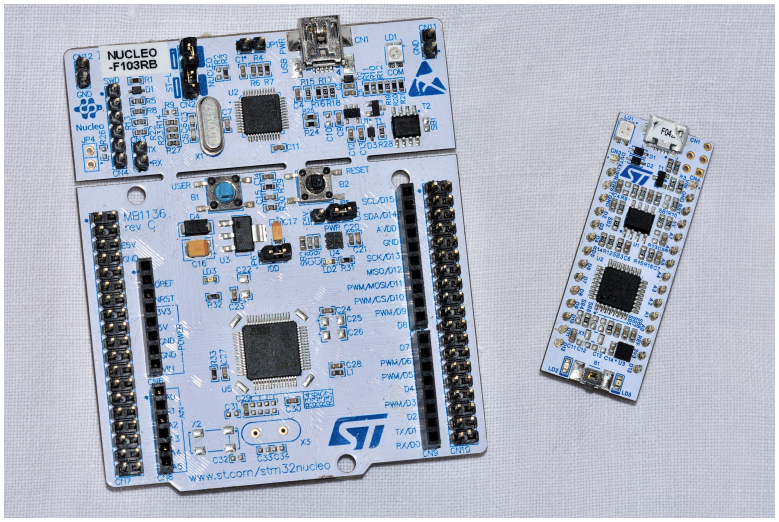
Obr. 54 Pomocná definice pro určení způsobu sériové komunikace MCU

Poslední úpravy projektu se týkají ukládání parametrů použitého snímače do FLASH paměti. Je zapotřebí definovat adresu, kam se budou moct parametry bezpečně uložit. Je potřeba mít na paměti, že paměť FLASH se dělí na jednotlivé celky nazývané stránky(FLASH_PAGE). Pro naprogramování je pak nejdříve zapotřebí vymazat celý tento celek, kde se nachází adresa na kterou chceme zapisovat. Zvolený celek pro ukládání vlastností snímače pak nelze využít pro vlastní program, protože při každém ukládání tato část paměti bude nejdříve vymazána. Definice adres a jejich velikosti jednotlivých celků mohou být dohledány v příslušném referenčním manuálu. Doporučený postup je využít poslední dostupnou stránku FLASH paměti a hlídat, že velikost programu nepřekročí příslušnou mez. Definice adresy paměti, kam bude MCU ukládat, se upravuje v souboru `user_functions.h` definováním makra USER_FLASH_PAGE. STM32F303RE má velikost jedné stránky definovanou 2KB. Poslední dostupná stránka je 255, která začíná na adrese 0x0807F800.

5 Realizované ukázky na různém hardware

5.1 Využití kitů Nucleo 32 a Nucleo 64 pro indikaci

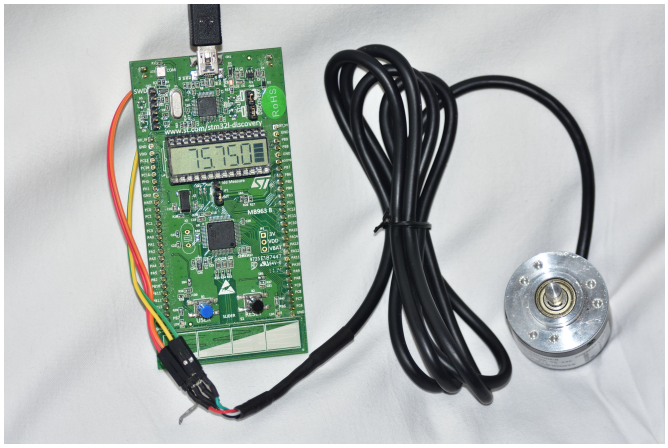
Tyto desky jsou vhodné pro využití jako indikační jednoty prostřednictvím mnou vytvořeného softwaru díky připravenosti jejich HW. Nemusíme řešit napájení MCU a jejich programování je snadné díky přítomnému ST-Linku. Navíc jejich velká většina je vybavena externím oscilátorem HSE 8MHz, který je vhodný pro zprovoznění komunikace pomocí USB. V rámci této práce vznikly vzorové projekty pro indikační jednotky vhodné pro kity STM32F042K6 a STM32F303RE.



Obr. 55 Vývojové desky Nucleo 32 a Nucleo 64 [8]

5.1.1 Práce se vzorovými projekty pro indikační jednotku na kitech Nucleo

Tyto vzorové projekty pro STM32F042 a STM32F303RE jsou připraveny pro komunikaci pomocí UART, využívajícího vestavěného USB/UART převodníku v ST-Linku těchto zařízení. Po připojení těchto kitů a nahrání SW stačí připojit snímač k pinům PA0 A PA1 patřícího čítači TIM2. Dále v PC aplikaci emulující sériový terminál vybrat port patřící ST-Linku. Rychlost přenosu je nastavena na 115200 B/s. Zbytek ovládání je již popsán v kapitole 4.6 věnující se mé terminálové aplikaci.



Obr. 56 Měřící přípravek s STM32L-DISCOVERY kitem a rotačním enkóderem [8]

5.2 Měřící přípravek s LCD displejem a STM32L152

K vytvoření tohoto přípravku vedla myšlenka nalezení jednoduché náhrady za komerční indikační jednotky, která by vyžadovala minimum příprav. STM32L-DISCOVERY kit od firmy STMicroelectronics se k tomuto účelu skvěle hodil, protože nabízí LCD displej a USB-UART převodník vestavěný v ST-LINKu. Díky tomu se po nahrání kódu z tohoto kitu stává přenosný a účelný měřící přípravek, ke kterému stačí již jen připojit snímač dle preferencí jako na obrázku 56.

Vybrané vlastnosti STM32L-DISCOVERY kitu dle [25]

- STM32L152RBT6 (128 KB Flash memory, 16 KB RAM, 4 KB EEPROM)
- LCD displej s možností zobrazit 6 znaků i desetinnou čárku.
- USB mini konektor pro práci s ST-Link
- maximální frekvence jádra 32MHz

5.2.1 Popis měření s přípravkem STM32L-DISCOVERY

Ke kitu připojíme napájecí a zemnící vodiče lineárního nebo rotačního snímače. Signálové vodiče s kvadraturním signálem přivedeme na piny PB6 a PB7, které jsou spojeny s 16-bitovou periferií čítače TIM4. Tyto piny tolerují napětí do 5V. Můžeme na ně přivést signál přímo z většiny snímačů. Pokud je na snímači přítomen signál s referenční značkou, připojíme ho na pin PA4. Dále jen přivedeme napájení pomocí USB mini konektoru na kitu. V tuto chvíli již můžeme začít měřit. K výpisu měřených dat nepotřebujeme počítač díky přítomnosti LCD displeje.

Kit je nastaven tak, aby se po zapnutí nejdříve čekalo na nulový pulz. Čekání signalizuje blikáním displeje. Po přejetí přes referenční značku, se měřený údaj vynuluje a zařízení přestane blikat. Pokud použitý snímač nebo signál s nulovým pulzem není k dispozici, můžeme přerušit hledání stiskem modrého tlačítka, které zároveň slouží pro nulování.

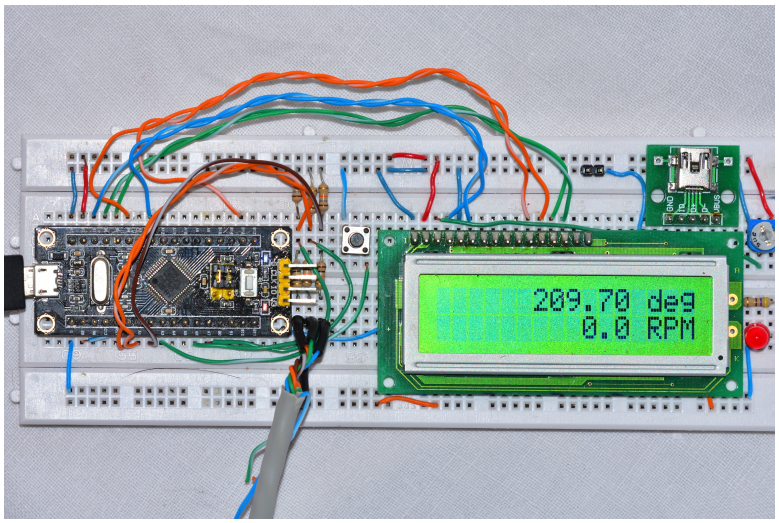
Pro upravení parametrů použitého snímače či přepnutí způsobu výpisu mezi metrickými jednotkami a počtem měřících kroků je zapotřebí spojit kit s počítačem. K tomu je připraveno rozhraní USB na pinech PA12(D+) a PA11(D-), ke kterým musíme nejdříve připojit externí konektor. Nastavené parametry mohou být uložené do paměti zařízení, takže při opětovném použití není třeba fázi nastavování podstupovat znovu.

5.3 Black pill s LCD displejem generování dráhových pulzů

Displej discovery kitu s STM32L152 má omezený počet zobrazovaných pozic a tedy se na něj nevejdou doplňkové údaje, jako jsou údaje o rychlosti pohybu, či informace o jednotkách, ve kterých je zrovna hodnota měření vypisována. Proto jsem se rozhodl u dalšího měřícího přípravku využít displeje s 16x2 vypisovatelnými pozicemi, popsanému v kapitole 3.4. Tento přípravek také slouží pro demonstraci rozšíření terminálové aplikace pro generování dráhových pulzů zmiňovaném v kapitole 4.7.1. Pro tento přípravek bylo využito vývojové desky Black pill s MCU od firmy STMicroelectronics STM32F103C8.

Vybrané vlastnosti přípravku s Black pill (STM32F103C8)

- STM32L152RBT6 (64 KB Flash memory, 20KB RAM)
- LCD displej 16x2 s řadičem typu HD47780 od firmy Ampire Co.
- Externí oscilátor HSE 8 MHz
- maximální frekvence jádra 72MHz



Obr. 57 Přípravek s STM32F103 a displejem pro generování dráhových pulzů [8]

5.4 Popis měření s přípravkem- Black pill

Manipulace s tímto kitem je obdobná práci s přípravkem využívající STM32L-DISCOVERY kit zmiňovaném v kapitole 5.2.1. Na rozdíl od něj je zde přítomný napěťový dělič pro práci se snímači, které mají výstup napěťové úrovně TTL. Pro čítání kvadraturního signálu je využita 16-bitovou periferie TIM2 na pinech PA0 a PA1. Referenční pulz je přes napěťový dělič spojen s pinem PA0. Tlačítko pro nulování měření připojeno pinu PA3. Na nepájivém poli dále najdeme externí konektor USB pro poskytnutí 5 V napájecí napětí pro snímače a trimr pro nastavování kontrastu obrazovky.

Deska Black pill je napájena prostřednictvím micro-usb konektoru, který zároveň slouží komunikaci s PC prostřednictvím terminálové aplikace. Deska Black pill je připravena pro komunikaci pomocí USB díky přítomnosti pull-up rezistoru na (D+) a přítomností 8 MHz externího krystalového oscilátoru. K samotnému měření ale spojení s počítačem není potřeba díky přítomnosti LCD dipleje.

Pro demonstraci rozšíření základní varianty terminálové aplikace o generování dráhových pulzů je k desce připojena LED dioda, která bude signalizovat průběh dráhových pulzů. Dále je využito druhé periferie čítače TIM3 na pinech PA7 a PA8 pro měření dráhy mezi jednotlivými pulzy. Ukládací proces terminálové aplikace, kromě nastavení týkajícího se snímače, také ukládá nastavené parametry pro generování pulzů, aby nastavení bylo zachováno i po vypnutí MCU.

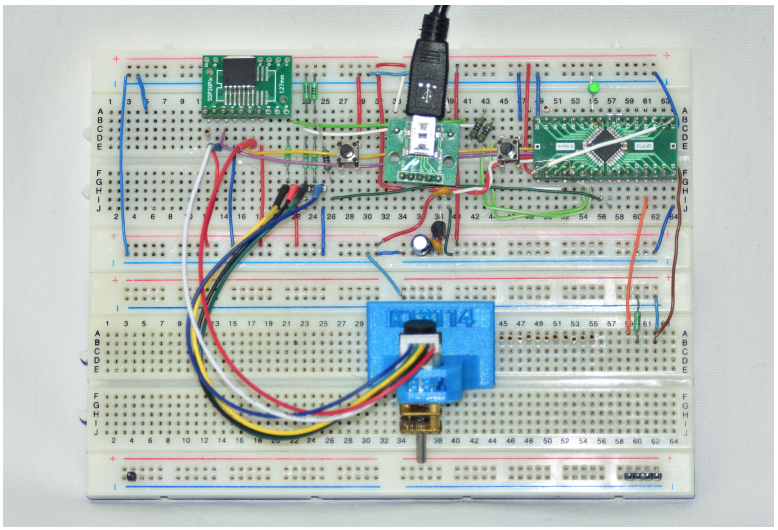
5.5 Přípravek s STM32F042 a akčním členem

Na rozdíl od ostatních přípravků prezentovaných v této práci, tento přípravek se neskládá z žádné předem hotové vývojové desky. Základem je procesor STM32F042K6, jehož napájení je řešeno pomocí konektoru USB a napěťového regulátoru LE33. Dále na nepájivém poli nalezneme H můstek pro ovládání DC motorku, tlačítka a pár dalších pasivních prvků. Podoba zapojení na nepájivém poli je na obrázku 58.

Tento přípravek slouží pro demonstraci využití dat získaných z inkrementálních snímačů pro účely regulace polohy a rychlosti akčních členů. K regulaci simuluje PID regulátor pomocí vestavěné knihovny CMSIS DSP od firmy ARM. Parametry regulace jsou nastavitelné pomocí terminálové aplikace a tedy kromě demonstrace využití dat z inkrementálních enkodérů, může tento přípravek sloužit zároveň pro demonstraci principu PID regulace.

Vybrané vlastnosti STM32F042

- STM32L152RBT6 (32 KB Flash memory, 6KB RAM)
- Interní oscilátor HSI48 vhodný pro použití s USB
- 9 TIM periferií



Obr. 58 Přípravek pro řízení akčního členu s STM32F042 [8]

6 Závěr

Tato práce vysvětlila základní principy měření s inkrementálními enkodéry poskytujícími informaci o relativní změně polohy. Vysvětluje jakým způsobem se dá využít čítačů v mikrokontrolérech řady STM32, které podporují čtení kvadraturního signálu. Dále ukazuje rozšířené možnosti využití těchto mikrokontrolérů vhodných k vytvoření řídící jednotky pro vyhodnocování signálu z inkrementálních snímačů s nulovým pulzem.

V úvodní kapitole jsou popsány vlastnosti kvadraturního signálu a naznačeny způsoby jakými tento signál v inkrementálních snímačích vzniká. Dále jsou popsány konkrétní typy snímačů a existujících indikačních jednotek pro práci s těmito snímači. Ty sloužily jako předloha pro funkce, které by mnou vytvářená řídící jednotka měla nabízet.

Následující kapitola se zabývá jednotlivými periferiemi vhodnými pro vytvoření samostatné indikační jednotky. Vysvětluje jaké výhody má využití jednotlivých periferií a základy práce s nimi. Tato část práce by mohla být využita jako výukový materiál v rámci fakulty FEL. V části věnující se rozhraní USB detailněji popisuje základní nastavení této periferie s využitím STM32CubeMX a HAL knihoven. Tato část by mohla najít vhodné využití v dalších studentských projektech. Do hloubky je také popsáno fungování periferie TIM v módu enkodéru.

Čtvrtá a pátá kapitola se zabývá samotným vývojem řídící jednotky. Jsou zde popsány postupy při implementaci jednotlivých funkcí výsledné indikační jednotky a ukázky výsledného softwaru. Dále v rámci těchto kapitol vzniklo několik funkčních variací s různými mikrokontroléry STM32 a byla popsána metodika přenosu vzorového projektu do STM32 mikrokontrolérů obecně, včetně možnosti změn použitých periferií.

Zadání bylo splněno v plném rozsahu. Výsledná práce by mohla posloužit jako výuková pomůcka, ať v už předmětech zabývajícími se mikrokontroléry nebo pro demonstraci vlastností PID regulátorů a inkrementálních snímačů. Indikační jednotka by také mohla posloužit jako jednoduchá náhrada složitějších indikačních jednotek jako je ADP1 od firmy ESSA pro měření v laboratorních cvičeních.

Literatura

[1] Creative Robotics Ltd. *What are Quadrature Encoders*. [online]. [cit. 2020-03-05]. Čvc 2017. Dostupné z http://www.creative-robotics.com/quadrature-intro.

[2] LARM a.s. *Snímače, ventily, sondy*. [online]. [cit. 2020-03-05]. Dostupné z http://www.larm.cz/snimace-ventily-sondy/bxc.

[3] A. Vojáček. *Princip optických enkodérů polohy pro řízení motorů*. [online]. [cit. 2020-04-7]. Ún. 2006. Dostupné z https://automatizace.hw.cz/clanek/2006022801.

[4] Wikipedie: otevřená encyklopedie. *Incremental encoder*. [online]. [cit. 2020-04-12]. Dub. 2020. Dostupné z https://en.wikipedia.org/wiki/Incremental_encoder.

[5] Sinotech. *Optical Encoders Overview*. [online]. [cit. 2020-05-2]. Květ. 2019. Dostupné z https://sinotech.com/products/motors/optical-encoders-overview/.

[6] HEIDENHAIN. *Výrobní Program*. [online]. [cit. 2020-05-9]. Zář. 2018. Dostupné z https://www.heidenhain.cz/fileadmin/pdb/media/img/350457-C4_V%3Frobn%3F_program_cs.pdf.

[7] Essa Praha. *Rotační snímače SR xxxx*. [online]. [cit. 2020-03-05]. 2019. Dostupné z https://www.essapraha.cz/rotacni-snimace-technicky-popis.

[8] Petr David. *Vlastní fotografie vzniklé pro účely BP*. 2020.

[9] Essa Praha. *Měřicí sondy SM xx / SMK xx*. [online]. [cit. 2020-02-25]. 2019. Dostupné z https://www.essapraha.cz/merici-sondy-technicky-popis.

[10] Murata Manufacturing. *Characteristics of common mode choke coils for signal lines and how to choose one*. [online]. [cit. 2020-04-25]. Science Direct, pros. 2016. Dostupné z https://www.murata.com/en-global/products/emiconfun/emc/2016/11/15/20161115-p1.

[11] HEIDENHAIN. *Interfaces of HEIDENHAIN Encoders*. [online]. [cit. 2020-05-12]. Zář. 2019. Dostupné z https://www.heidenhain.cz/cs_CZ/dokumentace/zaklady/interfaces/.

[12] Essa Praha. *Číslicová indikace ADP 1*. [online]. [cit. 2020-04-02]. 2019. Dostupné z https://www.essapraha.cz/cislicova-identifikace-adp-1.

[13] HEIDENHAIN. *Návod k obsluze ND5023*. [online]. [cit. 2020-05-12]. Květ. 2019. Dostupné z https://www.heidenhain.cz/fileadmin/pdb/media/MA/ND5000/1221049-C1_BA_ND5023_cs.pdf.

[14] Inc. LSI Computer Systems. *datasheet LS7366R 32-BIT QUADRATURE COUNTER WITH SERIAL INTERFACE*. [online]. [cit. 2020-04-29]. Červ. 2014. Dostupné z https://lsicsi.com/datasheets/LS7366R.pdf.

[15] STMicroelectronics. *Application note AN4776 General-purpose timer cookbook for STM32 microcontrollers*. [online]. [cit. 2020-02-05]. Rev 3, čvc 2019. Dostupné z https://www.st.com/resource/en/application_note/dm00236305.pdf.

[16] STMicroelectronics. *Reference manual RM0091 STM32F0x1/STM32F0x2/STM32F0x8 advanced ARM®-based 32-bit MCUs*. [online]. [cit. 2020-04-30]. Rev 9, led. 2017. Dostupné z https://www.st.com/resource/en/reference_manual/dm00031936.pdf.

[17] STMicroelectronics. *User manual UM1724 STM32 Nucleo-64 boards (UM1724)*. [cit. 2020-05-18]. Rev 13. Dub. 2019. Dostupné https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf.

[18] STMicroelectronics. *STM32 USB training - 09.1 USB CDC device basic labs*. [online]. [cit. 2020-03-10]. Dub. 2019. Dostupné z https://www.youtube.com/watch?v=h9T0RTu9Muc.

[19] STMicroelectronics. *USB hardware and PCB guidelines using STM32 MCUs AN4879*. [online]. [cit. 2020-04-13]. Rev 4, pros. 2018. Dostupné z https://www.st.com/resource/en/application_note/dm00296349-usb-hardware-and-pcb-guidelines-using-stm32-mcus-stmicroelectronics.pdf.

[20] STMicroelectronics. *User manual UM1956 STM32 Nucleo-32 boards (MB1180)*. [online]. [2020-02-03]. Rev 5, lis. 2018. Dostupné https://www.st.com/resource/en/user_manual/dm00231744-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf.

[21] STMicroelectronics. *User manual UM1718 STM32CubeMX for STM32 configuration and initialization C code generation*. [online]. [cit. 2020-03-10]. Rev 30, říj. 2019. Dostupné z https://www.st.com/resource/en/user_manual/dm00104712.pdf.

[22] AMPIRE C. LTD. *Specifications for LCD module AC162B*. [online]. [cit. 2020-04-22]. Rev 1, říj. 1999. Dostupné z https://www.olimex.com/Products/Components/LCD/LCD16x2-BL/resources/LCD16x2-BL.pdf.

[23] Tilen Majerle. *HAL library 15- HD44780 for STM32Fxxx*. [online]. [cit. 2020-05-20]. Motion Control tips, čvc 2015. Dostupné z https://stm32f4-discovery.net/2015/07/hal-library-15-hd44780-for-stm32fxxx/.

[24] Arm Limited. *PID Motor Control*. [online]. [cit. 2020-05-06]. Dub. 2020. Dostupné z http://www.keil.com/pack/doc/CMSIS/DSP/html/group__PID.html.

[25] STMicroelectronics. *Discovery kits for STM32L151/152 line*. [cit. 2020-02-05]. Rev 4. Zář. 2014. Dostupné z https://www.st.com/resource/en/data_brief/32l152cdiscovery.pdf.