# ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTRICKÝCH POHONŮ A TRAKCE


# BĚŽÍCÍ TEXT POMOCÍ ST NUCLEO

BAKALÁŘSKÁ PRÁCE

Studijní program: Elektrotechnika, energetika a management

Studijní obor: Aplikovaná elektrotechnika

Vedoucí práce: Ing. Jan Bauer, Ph.D.


David Jäschke


Praha 2020

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jäschke** Jméno: **David** Osobní číslo: **474741**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrotechnologie**

Studijní program: **Elektrotechnika, energetika a management**

Studijní obor: **Aplikovaná elektrotechnika**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Běžící text pomocí ST NUCLEO**

Název bakalářské práce anglicky:

**Programming of LED running text on ST - NUCLEO board**

Pokyny pro vypracování:

1) Seznamte se s kitem ST NUCLEO
2) Navrhněte algorimus pro obsluhu RGB LED WS2812
3) Naprogramujte funkci "běžícího textu" do kitu NUCLEO

Seznam doporučené literatury:

[1] datasheet ST NUCLEO
[2] Poupa J. Programovací jayzk C

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. Jan Bauer, Ph.D., katedra elektrických pohonů a trakce FEL**

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

Datum zadání bakalářské práce: **12.02.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

| Ing. Jan Bauer, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>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

# Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a že jsem uvedl veškeré použité podklady v seznamu použité literatury na konci práce.

V Praze dne _____ _____

podpis

# Abstrakt

Tato bakalářská práce se zabývá vytvářením programu, který umožňuje tvorbu běžícího textu. Tento běžící text je zobrazován pomocí dvou vyrobených přípravků, které jsou osazeny vždy osmi LED jednou s komunikací diskrétní a podruhé s datovou. Teoretická část práce zahrnuje základní popis zvoleného procesorového kitu a vývojového prostředí. V praktické části je uveden popis vytvořených programů a je zde vysvětlena jejich funkčnost.

# Klíčová slova

Kit NUCLEO-F302R8, běžící text, stejnosměrný motor, diskrétní LED, RGB LED s datovou komunikací

# Abstract

This bachelor thesis deals with a microcontroller program that allows for displaying of a running text. This running text is displayed using two manufactured test kits, which are each equipped with eight LEDs, first kit with LEDs based on discrete communication and the second one based on data communication. The theoretical part of the work includes a basic description of the selected processor kit and the development environment. The practical part describes the created programs and explains their functionality.

# Key words

Kit NUCLEO-F302R8, running text, DC motor, discrete LED, RGB LED with data communication

# Poděkování

Tímto bych chtěl poděkovat panu Ing. Janu Bauerovi, Ph.D. za nezměrnou ochotu a vstřícnost při vedení práce a za všechny poskytnuté rady. Obzvláště jsem ocenil jeho vždy rychlou odezvu na emailovou komunikaci.

Dále bych chtěl také poděkovat své rodině a kamarádům za podporu po celou dobu studia.

# Obsah

# 1 Úvod

Mikroprocesory nacházejí v dnešní době uplatnění v nespočetně mnoho elektrických zařízení. Může se jednat jak o zařízení, se kterými člověk přichází do styku na denní bázi nebo o zařízení pro specifické a nepříliš časté využití. Jako uplatnění, se kterým běžný člověk nepřichází denně do styku, bych uvedl například řízení otáček asynchronního motoru pomocí frekvenčního měniče, který je řízen mikroprocesorovou technikou.

I z těchto důvodů vysoké perspektivy jsem si vybral téma práce zaměřené na použití mikroprocesorové techniky. Cílem této práce je vytvoření programu pro kit NUCLEO-F302R8 za pomocí kterého je možné vytvářet běžící text. K zobrazování textu slouží dva vyrobené přípravky. První je osazen klasickými diskrétními LED (kapitola 5) a druhý využívá RGB LED (kapitola 6). Obě varianty použitých LED vyžadují rozdílné řízení, a proto vznikly i dvě odlišné verze programu.

K účelům programování kitu NUCLEO bylo zvoleno vývojové prostředí STM32CubeIDE vyvíjeného společností STMicroelectronics. Základní popis tohoto prostředí je v kapitole 3.

Aby byl běžící text čitelný, musí se jevit oku pozorovatele jako statický. Neboli psaní textu musí začínat každou otáčku ve stejné poloze. Pro zajištění této podmínky je využito optické závory, která je každou otáčku protínána rotující částí přípravku, což vede k začátku zobrazování běžícího textu.

V kapitole 4 je popsáno provedení pohonu rotující části. Pro možnost ovládání rychlosti otáčení rotující části je použito principu snižování střední hodnoty stejnosměrného napájecího napětí motoru za pomocí stejnosměrného měniče napětí, který je řízen pulzně šířkovou modulací z kitu NUCLEO.

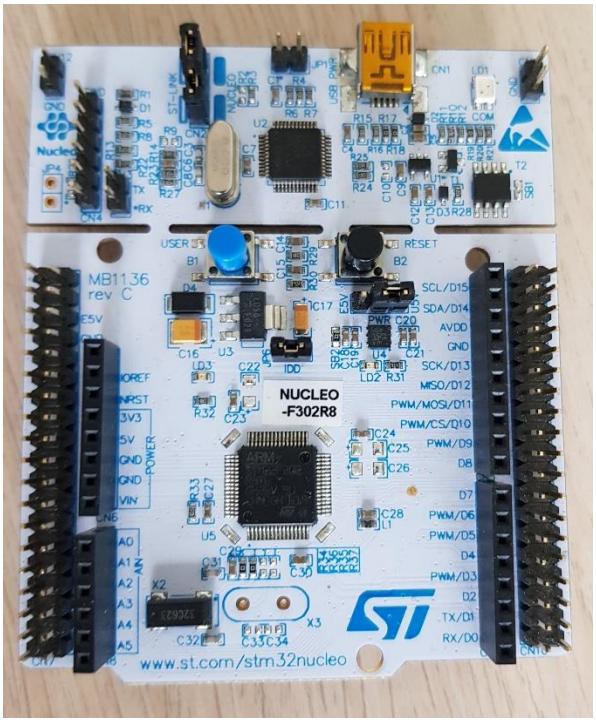
Jako jedno z možných využití přípravků se jeví možnost použití jako demonstrační příklad využití mikroprocesorové techniky například na dnech otevřených dveří katedry elektrických pohonů a trakce nebo na podobných propagačních akcích.

# 2 Kit ST NUCLEO-F302R8

Tento kit byl zvolen z důvodů jeho dobré dostupnosti a nízké ceně. V této části budou popsány především mnou používané periferie pro potřeby přípravků. Kompletní popis jednotlivých periferií lze nalézt v [1].

## 2.1 Základní vlastnosti

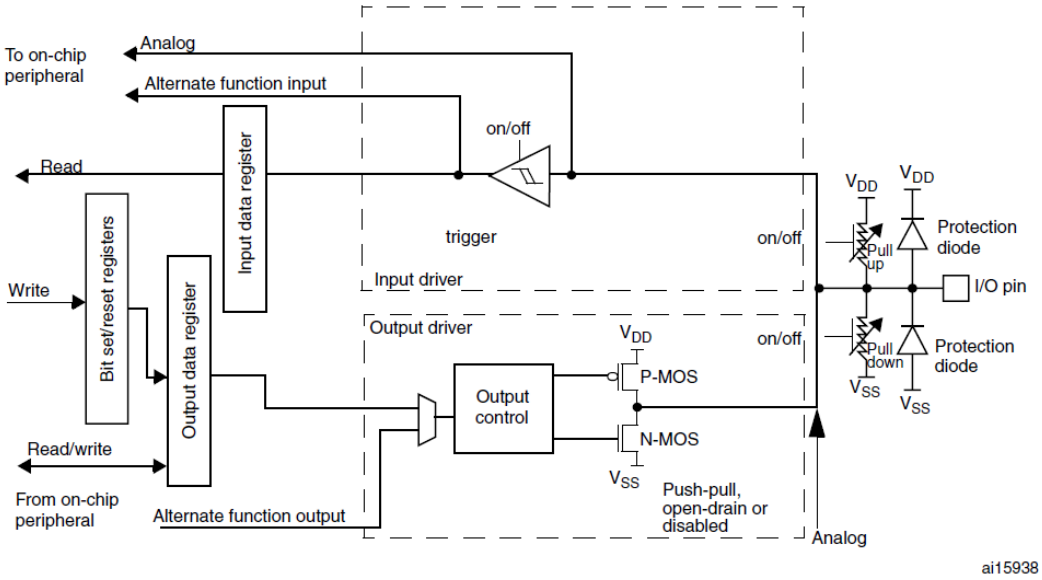
- ARM®32-bit Cortex®-M4 CPU
- 72 MHz maximální CPU frekvence
- VDD od 2 do 3,6 V
- 64 KB flash paměť
- 16 KB SRAM paměť
- 51 GPIO pinů
- 12-bit ADC s 15 kanály
- 12-bit DAC
- RTC
- 6× čítač
- komunikace: 3× I2C, 2× SPI, USB 2.0




Obr. 2.1 – Kit NUCLEO-F302R8

## 2.2 GPIO výstup

Deska F302R8 disponuje 51 GPIO výstupy, které jsou rozděleny mezi několik portů označovanými velkými písmeny (například PA5 je GPIO výstup číslo 5 na portu A). Ke konfiguraci těchto portů slouží čtyři 32-bitové registry (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR). Jako datové registry slouží dva 32-bitové registry (GPIOx_IDR, GPIOx_ODR). Pro set či reset je dostupný 32-bitový registr (GPIOx_BSRR). Pro zamknutí GPIO výstupů slouží 32-bitový registr (GPIOx_LCKR). Pro přiřazení alternativních funkcí GPIO výstupům slouží dva 32-bitové registry (GPIOx_AFRH, GPIOx_AFRL).




Obr. 2.2 – Blokové zapojení GPIO výstupu [1]

## 2.3 Čítač

Čítač je periferie, která mění svůj obsah CNT registru o jedničku s každým příchozím pulzem na svůj vstup hodin. Deska F308R8 má pro uživatele k dispozici celkem 6 čítačů. Pro potřeby generování PWM výstupu na některém z kanálu čítače jsou důležité následující registry. V registru CR1 probíhá základní nastavení včetně s jeho zapínání. Registr CCMR1 slouží ke zvolení modu PWM. Registr PSC obsahuje hodnotu předděličky, která umožňuje zpomalení čítání čítače vůči dostupnému signálu hodin. Hodnota v registru ARR určuje periodu přetečení čítače. Délka pulzu je určená hodnotou uloženou v registru CCR1.

## 2.4 DMA

Tato periferie dovoluje přenos dat mezi pamětí a pamětí nebo mezi pamětí a periferií, o který se nestará procesor, a tudíž není procesor tímto přenosem zahlcený. Jedná se tedy o rychlý způsob přenosu velkého množství dat. V registru CCRx probíhá základní nastavení kanálu x včetně jeho povolení. Do registru CNDTR se zapisuje počet potřebných přenosů. Každým přenosem se tento počet sníží o jedničku a pokud je v tomto registru zapsaná nula tak již žádný přenos neproběhne. Adresy periferie, respektive místa v paměti pro cíl nebo zdroj přenosu se zapisují do registru CPAR, respektive do CMAR.

# 3 Vývojové prostředí STM32CubeIDE

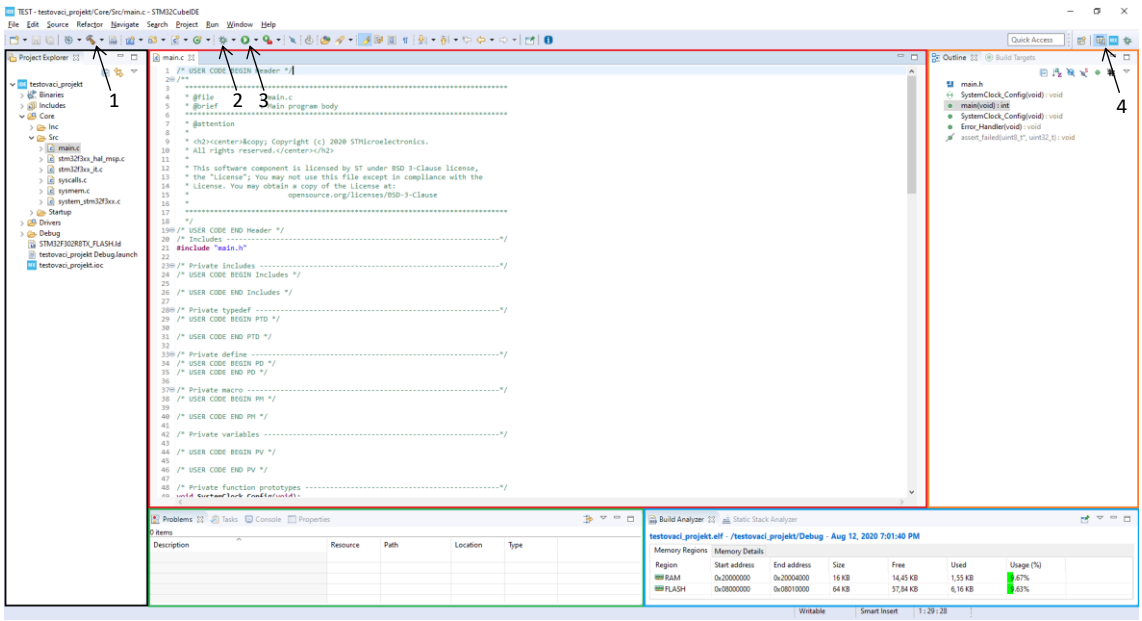
Toto vývojové prostředí bylo zvoleno na základě několika důvodů. Je postaráno o dobrou kompatibilitu mezi kitem NUCLEO a procesorem, a to z důvodu, že oba výrobci jsou firma ST. Dále se skládá ze dvou částí, které výrazně usnadňují práci s procesorem. Tyto části jsou podrobněji popsány v kapitolách 3.1 a 3.2.

## 3.1 TrueSTUDIOforSTM32

Tato komponenta dovoluje psaní samotného zdrojového kódu pro mikrokontrolerové jednotky řady STM32 v jazyce C nebo C++ a dovoluje i následné odladění kódu v debugeru.

Černou barvou je ohraničen průzkumník projektu, ve kterém otvíráním složek lze nalézt všechny vygenerované soubory. Červené okno je okno textového editoru pro úpravu zdrojového kódu. Oranžové okno obsahuje přehled použitých proměnných, připojených souborů a použitých funkcí v souboru, který je otevřen v textovém editoru. V zeleném okně lze nalézt chyby a výstrahy, které jsou objeveny při překladu. Modré okno obsahuje přehled využití příslušných pamětí procesoru.

Ikonou na pozici 1 se překládá program. Ikona 2 slouží ke spuštění ladění programu v reálném čase. Spustit program bez možnosti ladění lze ikonou na pozici 3. Pod číslem 4 se nachází trojice ikon, kterými lze přepínat mezi zobrazovacími mody. Zleva se jedná o mód textového editoru, STM32CubeMX a ladícím.

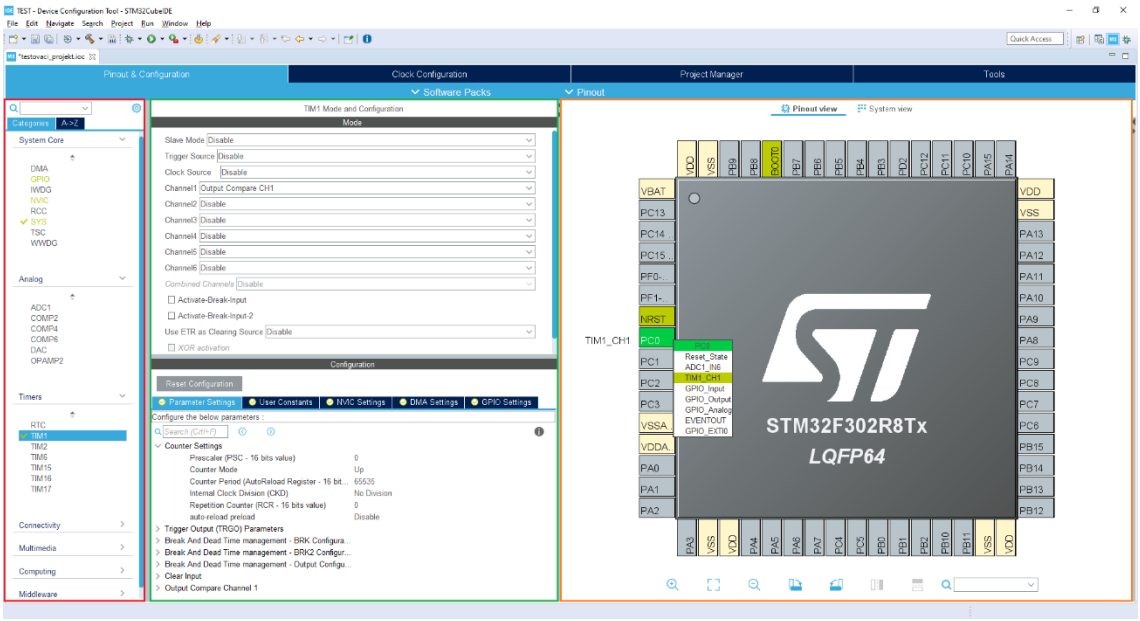



Obr. 3.1 – Vzhled prostředí pro komponentu TrueSTUDIOforSTM32

## 3.2 STM32CubeMX

Tato část slouží pro snadnou inicializaci zvolených periferií procesoru. Lze zde nastavovat také konfiguraci hodin, vytváření automatického kódu dle inicializace a nachází se zde i přehled energetické náročnosti kitu.

V červené části se nachází seznam všech programovatelných periferií, které lze po výběru konfigurovat v zeleném okně. Zde je pro ukázku zvolený TIM1, kterému je nastavený na kanál 1 funkce output compare. Program zvolí, na kterém pinu bude funkce probíhat, ale pokud se funkce nalézá i na jiném pinu tak lze tuto volbu změnit. V oranžové části se nalézá grafická vizualizace s popsanými piny procesoru. Zobrazené piny jsou interaktivní a po jejich rozkliknutí se otevře nabídka všech dostupných funkcí zvoleného pinu.




Obr. 3.2 – Vzhled prostředí STM32CubeMX

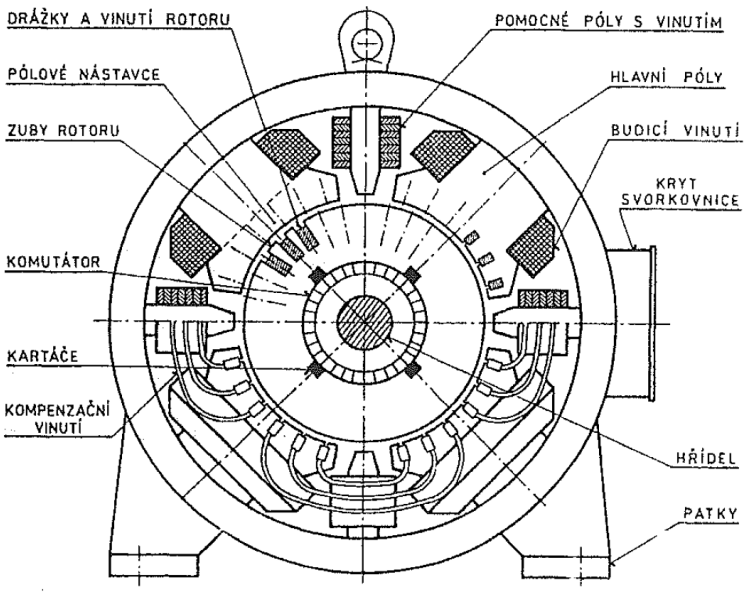
# 4 Pohon rotující části přípravků

Jako pohon obou variant je zvolen stejnosměrný motor s permanentními magnety. Otáčivý pohyb z hřídele motoru je přenášen na rotující část přípravku pomocí třecí vazby viz obrázek 4.1.

Obr. 4.1 – Pohon rotující části

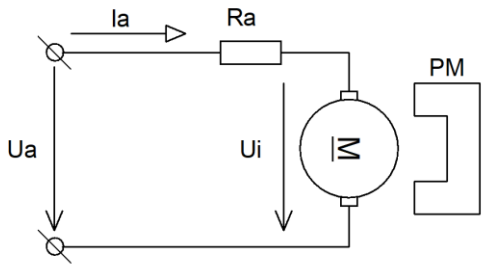
## 4.1 Stejnosměrný motor

Stejnosměrný motor je točivý elektrický stroj, který je napájený stejnosměrným napětím. Na rotoru je uloženo v drážkách magnetického obvodu rotorové vinutí, které je připojeno na komutátor. Na statoru je umístěno budící vinutí, které vytváří budící magnetický tok. V případě stejnosměrného motoru s permanentními magnety není na statoru umístěno budící vinutí, ale budící magnetický tok je vytvářen permanentními magnety.




Obr. 4.2 – Příčný řez stejnosměrným strojem [2]

Základní obvodové schéma stejnosměrného motoru s permanentními magnety se nachází na obrázku 4.3. Stejnosměrný motor popisují tři základní rovnice v ustáleném stavu (4.1) až (4.3).




Obr. 4.3 – Obvodové schéma stejnosměrného motoru s permanentním magnetem

$$U_a = R_a I_a + U_i \tag{4.1}$$

kde $U_a$ je svorkové napětí obvodu kotvy, $R_a$ činný odpor obvodu kotvy, $I_a$ proud kotvou a $U_i$ indukované napětí na kotvě.

$$U_i = C\Phi\Omega \tag{4.2}$$

kde C je konstrukční konstanta stroje, Φ je budící magnetický tok a Ω je úhlová rychlost otáčení.

$$M = C\Phi I_a \tag{4.3}$$

kde M je elektromagnetický moment.

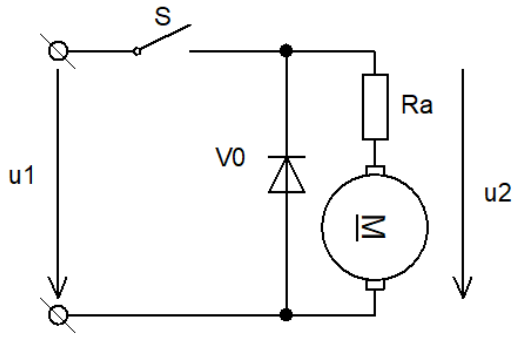
Vyjádřením úhlové rychlosti v rovnici (4.2) a dosazením rovnice (4.1) vzniká rovnice (4.4)

$$\Omega = \frac{U_a - R_a I_a}{C\Phi} \tag{4.4}$$

Z rovnice (4.4) plynou možné způsoby regulace úhlové rychlosti. Jedná se o změnu budícího toku, změnu svorkového napájecího napětí kotvy a změnu činného odporu v obvodu kotvy. V případě použití stejnosměrného motoru s permanentními magnety nelze regulovat budící magnetický tok, takže tato možnost odpadá. Změna činného odporu obvodu kotvy je pro případ této práce příliš mechanicky náročná a špatně realizovatelná. Optimálním způsobem se jeví regulace napájecího napětí obvodu kotvy, která bude realizována pulzně šířkovou modulací stejnosměrným měničem napětí.

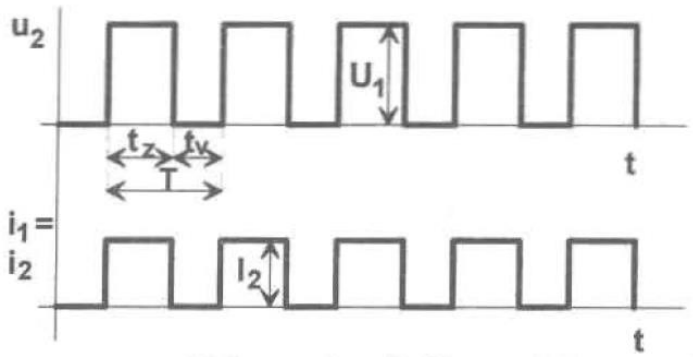
## 4.2 Stejnosměrný měnič napětí

Snižovací stejnosměrný měnič napětí si lze představit jako spínač, který periodicky připíná zátěž ke konstantnímu zdroji napětí, čímž pomocí regulace doby sepnutí za předpokladu konstantní periody reguluje střední hodnotu výstupního napětí od hodnoty konstantního napětí až po nulové napětí. Rychlost spínání musí být alespoň tak vysoká, aby časová konstanta kotvy motoru filtrovala periodické připnutí a odepnutí od zdroje.




Obr. 4.4 – Zjednodušené schéma zapojení měniče

Časový průběh výstupního napětí a proudu z měniče je zobrazen na obrázku 4.5 a je popsán rovnicí (4.5).




Obr. 4.5 – Časový průběh výstupního napětí a proudu měniče při R zátěži [3]

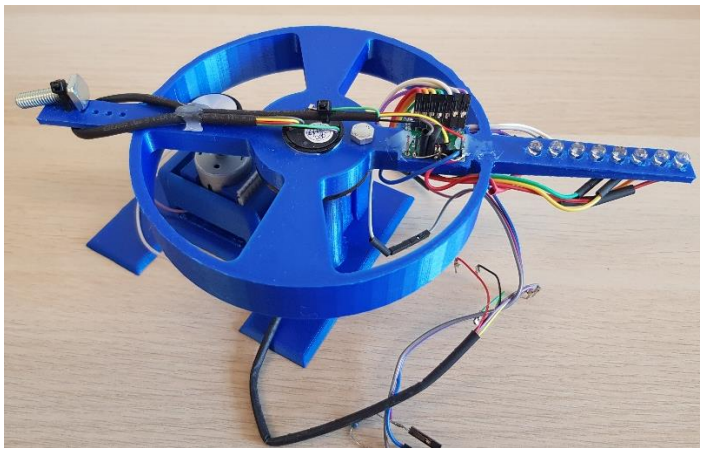
$$u_{2(AV)} = \frac{1}{T}\int_0^{t_z} U_1 \, dt = U_1 \frac{t_z}{T} = U_1 z \tag{4.5}$$

Zde uvedený průběh napětí a proudu na obrázku 4.5 platí pouze pokud je měnič zatížen čistě R zátěží, což není ovšem případ elektromotoru, který představuje RL zátěž s aktivní složkou protinapětí. Obrázek slouží pro demonstraci principu měniče. Popis průběhu proudu při RL zátěži s aktivní složkou je nad rámec této práce a lze ho například nalézt v kapitole číslo 5 zdroje [3].

Obrázek 4.4 je pouze zjednodušený. Pro potřeby práce, s ohledem na technické možnosti kitu NUCLEO, je ve skutečnosti motor regulován tranzistorem MOSFET IRLZ24N, jehož gate je buzen z kitu NUCLEO.

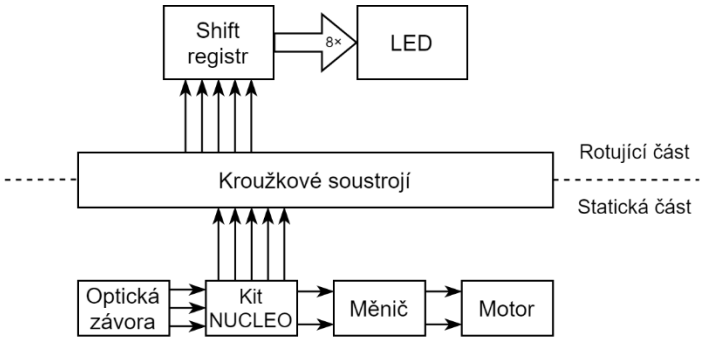
# 5 Varianta přípravku s diskrétními LED

Tento přípravek je osazen osmi klasickými LED, které mají nevýhodu, že mohou svítit pouze jednou barvou, ale jejich výhodou je jednoduchá obsluha. Je ovšem potřeba zajistit, aby se LED rozsvěcely ve stejný časový okamžik. První možností by bylo každou LED ovládat jedním výstupním pinem kitu NUCLEO, jelikož jsou LED umístěny na rotující části, znamenalo by to přenášet informace sběračem s devíti kroužky. Proto se jako efektivnější cesta jevilo využít zjednodušenou datovou komunikaci. K tomu byl zvolen 8-bitový shift registr 74HC594. Tento přípravek byl vyroben na 3D tiskárně.



Obr. 5.1 – Přípravek osazený diskrétními LED

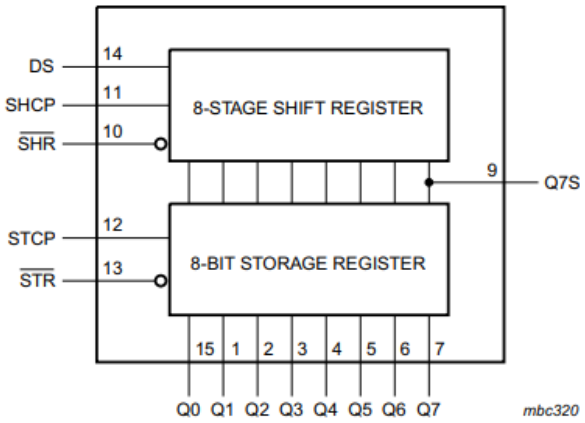
Kvůli snížení setrvačných hmot přípravku, není kit NUCLEO uložen na rotující části, kde jsou uloženy LED a shift registr. Kit se shift registrem jsou propojeny pomocí kroužkového soustrojí uloženého v ose otáčení. Zjednodušené blokové zapojení přípravku se nalézá na obrázku 5.2.




Obr. 5.2 – Blokové zapojení přípravku s diskrétními LED

## 5.1 Posuvný 8-bitový registr 74HC594

Tato součástka obsahuje 8-bitový shift registr se sériovým vstupem dat a sériovým či paralelním výstupem dat. Na paralelní výstup dat je zapojen 8-bitový paměťový registr realizovaný osmi D klopnými obvody. Shift a paměťový registr mají na sobě nezávislé přímé nulování registru a také řídící vstup hodin. Blokové schéma zapojení lze nalézt na obrázku 5.3.




Obr. 5.3 – Blokové schéma zapojení [4]

### 5.1.1 Vysvětlení funkce

V okamžiku přivedení náběžné hrany na vstup SHCP si první D klopný obvod (shift registr) FFSH0 převezme na svůj výstup Q logickou hodnotu, která se nachází na jeho vstupu D, který je přímo propojený se vstupem DS. Analogicky toto provedou zbylé klopné obvody 1 až 7. Přiváděním náběžných hran na vstup SHCP probíhá vsouvání dat z DS do shift registru.

Po přivedení náběžné hrany na vstup STCP si klopné obvody (storage register) FFST0 až FFST7 přesunou logickou hodnotu z jejich vstupů D na jejich výstupy Q, které jsou vyvedeny na výstupy Q0 až Q7.

Pro potřebu nulování shift, respektive paměťového registru stačí přivést na vstup SHR, respektive na vstup STR logickou úroveň LOW. Výstup Q7S je sériový výstup dat, který lze využít v případě potřeby rozšíření z pouze 8-bitového registru na n-bitový.

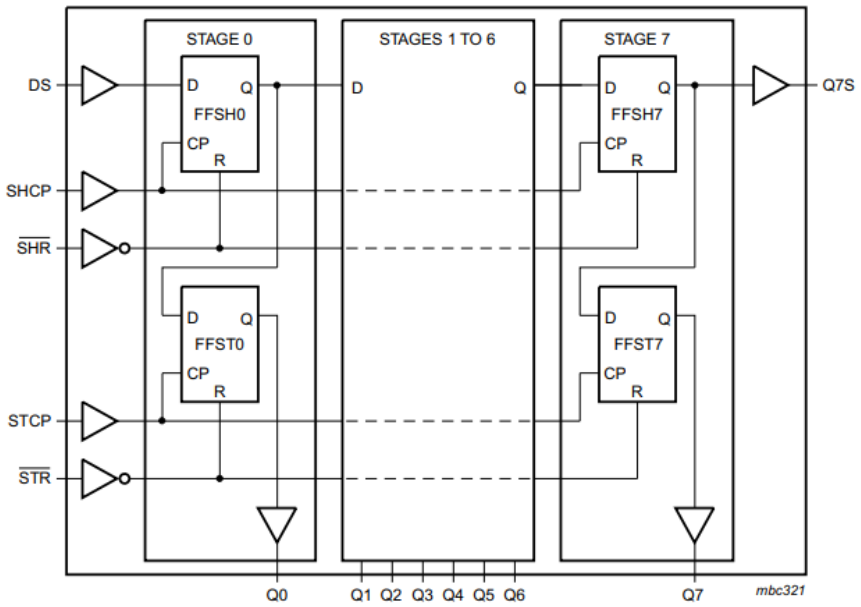
Funkci také popisuje následující tabulka 5.1.

| | $\overline{SHR}$ | $\overline{STR}$ | SHCP | STCP | DS |
|---|---|---|---|---|---|
| Vynulování shift registru | L | X | X | X | X |
| Vynulování paměťového registru | X | L | X | X | X |
| Načtení hodnoty na DS do shift registru | H | X | ↑ | X | H nebo L |
| Převedení hodnot ze shift registru na výstupy Q0 až Q7 | X | H | X | ↑ | X |

Tab. 5.1 – Funkce posuvného registru

H .... logická úroveň HIGH
L ..... logická úroveň LOW
↑ ... náběžná hrana signálu
X .... nezáleží na přivedeném signálu

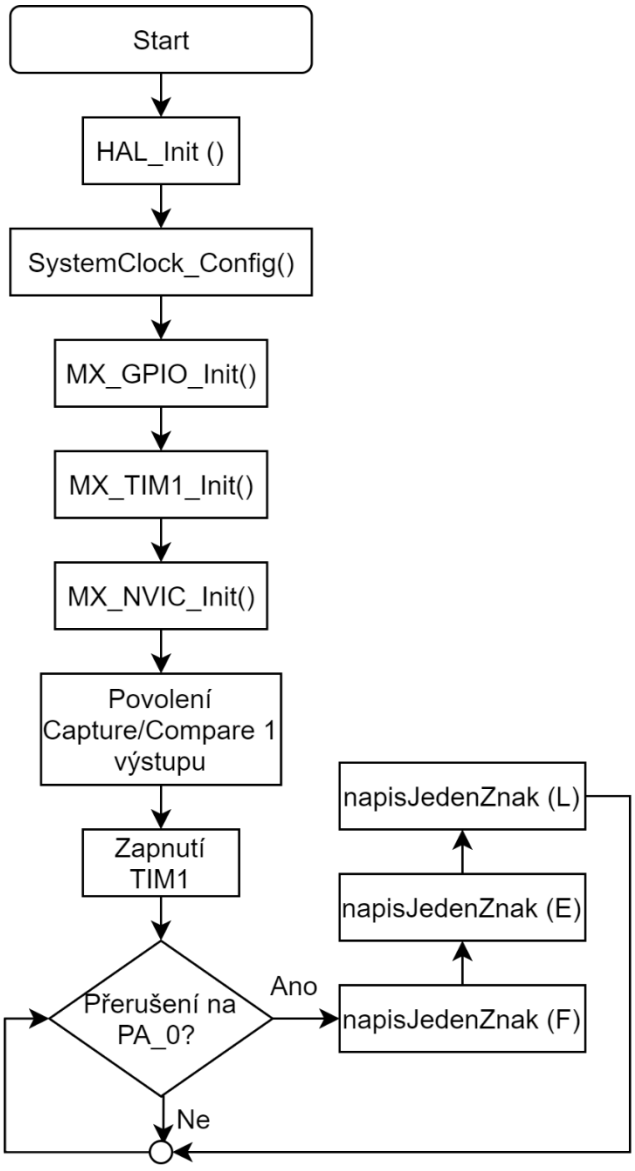
Na obrázku 5.4 lze vidět zapojení jednotlivých D klopných obvodů. Obrázek je rozdělen na osm segmentů, které jsou analogicky zapojené a kde každý segment představuje jeden paralelní výstup dat. V horní části obrázku jsou klopné obvody shift registru, ve spodní části pak klopné obvody paměťového registru.




Obr. 5.4 – Vnitřní zapojení registru [4]

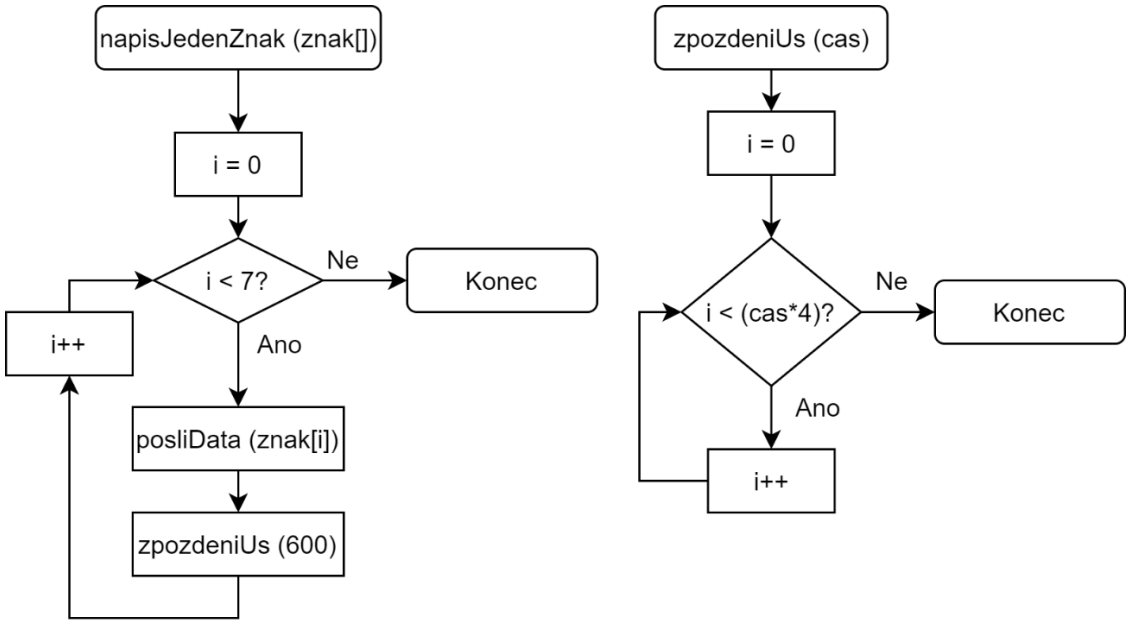
## 5.2 Vývojové diagramy programu

Program začíná inicializací používaných periferií. Poté čeká na průchod rotující části světelnou závorou, což vygeneruje přerušení na pinu PA_0 a začne se zobrazovat požadovaný nápis viz Obr. 5.5.



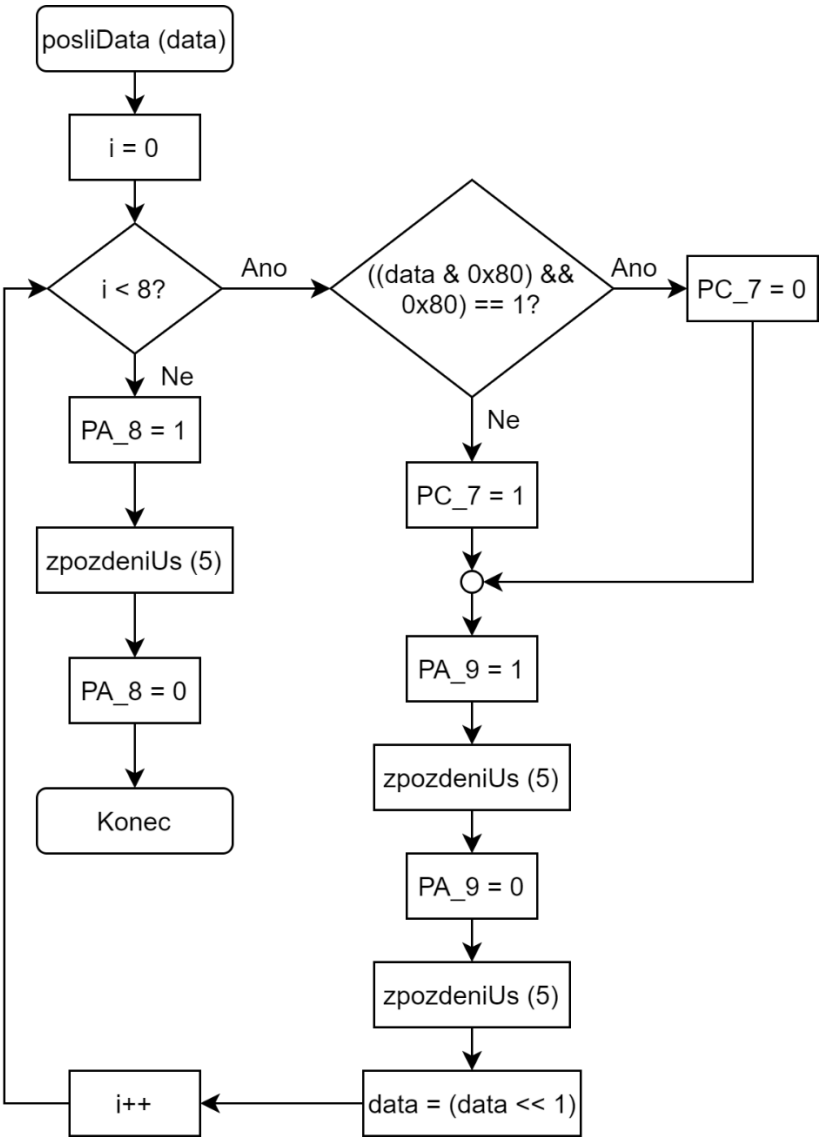

Obr. 5.5 – Vývojový diagram hlavní části programu

Zde je vysvětlení funkce napisJedenZnak použité v předchozím diagramu a funkce zpozdeniUs, která vytváří zpoždění začínající v řádu µs.




Obr. 5.6 – Vývojové diagramy pro funkce napisJedenZnak a zpozdeniUs

Funkce posliData slouží k rozsvěcování a zhasínání LED. K tomu dochází díky posuvnému registru, do kterého je nejdříve potřeba naplnit data podle toho, která LED svítit má a která ne.




Obr. 5.7 – Vývojový diagram funkce posliData

## 5.3 Zdrojový kód

Funkce zpozdeniUs slouží jako programovatelné zpoždění, pro časování shift registru. Experimentálně bylo změřeno že 4 chody cyklu odpovídají přibližně času 1 µs. Vstupním parametrem této funkce je tedy čas zpoždění v mikrosekundách.

```
293 void zpozdeniUs (uint32_t cas) // 4 chody cyklu = 1 us
294 {
295     for (uint32_t i = 0; i < (cas*4); i++) {}
296
297 }
298
```

Obr. 5.8 – Funkce zpozdeniUs

Zápis dat do posuvného registru je prováděn pomocí funkce posliData. Vstupním parametrem je osmibitové číslo. Každý bit představuje jednu LED, kdy nejvyšší bit představuje LED umístěnou nejvzdáleněji od středu otáčení.

Zápis dat do registru probíhá impulzem na pinu PA_9. Zapsaná hodnota je dle stavu pinu PC_7, který se volí podmínkou dle hodnoty proměnné „data“. Pokud má LED svítit tak bude na pinu PC_7 logická úroveň „0“ a obráceně. Podmínka je testována pouze pro nejvyšší bit a aby nedocházelo k opakovanému testování stejného bitu tak je vždy na konci cyklu proveden bitový posun vlevo proměnné „data“ o jednu pozici.

Po provedení zápisu dat pro všechny LED do posuvného registru se přivede impulz na pin PA_8, kterým se provede vysunutí dat na výstupní piny registru Q0 až Q7, které jsou připojeny na samotné LED a tím se rozsvítí požadovaný počet LED.

```
299 void posliData (uint8_t data) //posilam od nejvyssiho bitu dat protoze rozsvicim led od nejvzdalenejsi
300 {
301     for (uint8_t i = 0; i < 8; i++) //jeden cyklus pro kazdou led
302     {
303         if ((data & 0x80) && 0x80)
304             GPIOC->ODR &= ~GPIO_ODR_7;
305         else
306             GPIOC->ODR |= GPIO_ODR_7;
307
308         GPIOA->ODR |= GPIO_ODR_9; //udelam pulz pro zapis jednoho bitu
309         zpozdeniUs(5);
310         GPIOA->ODR &= ~GPIO_ODR_9;
311         zpozdeniUs(5);
312         data = (data<<1);
313     }
314     GPIOA->ODR |= GPIO_ODR_8; //pulz pro vyslani dat na LED
315     zpozdeniUs(5);
316     GPIOA->ODR &= ~GPIO_ODR_8;
317
318 }
```

Obr. 5.9 – Funkce posliData

Funkce napisJedenZnak v cyklu volá funkci posliData a tím postupně zobrazí jeden znak. Každý znak je tvořen celkem sedmi osmibitovými čísly (viz Obr. 5.11), které si postupně volá funkce posliData. Po odeslání jedné sady dat na LED funkcí posliData následuje zpoždění 600 µs, která zajišťuje vizuální mezeru mezi jednotlivými částmi znaků.

```
320 void napisJedenZnak (uint8_t znak[])
321 {
322     for (uint8_t i = 0; i < 7; i++)
323     {
324         posliData(znak[i]);
325         zpozdeniUs(600);
326     }
327
328 }
329
```

Obr. 5.10 – Funkce napisJedenZnak

```
46 /* USER CODE BEGIN PV */
47 uint8_t F [] = {0xFF,0x90,0x90,0x80,0x80,0x00,0x00};
48 uint8_t E [] = {0xFF,0x91,0x91,0x91,0x91,0x00,0x00};
49 uint8_t L [] = {0xFF,0x01,0x01,0x01,0x01,0x00,0x00};
50 /* USER CODE END PV */
```

Obr. 5.11 – Deklarace proměnných pro zobrazení slova „FEL“

Impulzem pro začátek tvoření běžícího textu je průchod světelnou závorou a tím vygenerování přerušení programu, které se obsluhuje ve funkci na obrázku 5.12. Postupně takto dochází k zobrazení jednotlivých písmen běžícího textu.

```
330 void EXTI0_IRQHandler(void)
331 {
332   /* USER CODE BEGIN EXTI0_IRQn 0 */
333     napisJedenZnak(F);
334     napisJedenZnak(E);
335     napisJedenZnak(L);
336   /* USER CODE END EXTI0_IRQn 0 */
337   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
338   /* USER CODE BEGIN EXTI0_IRQn 1 */
339   /* USER CODE END EXTI0_IRQn 1 */
340 }
```

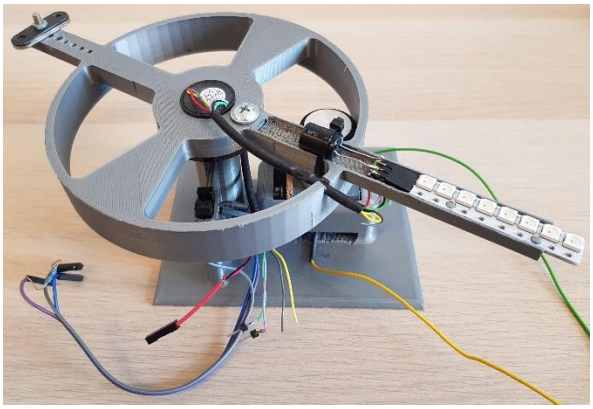
Obr. 5.12 – Funkce obsluhy přerušení

Výslednou podobu běžícího textu s nápisem FEL lze vidět na obrázku 5.13.

Obr. 5.13 – Ukázka běžícího textu

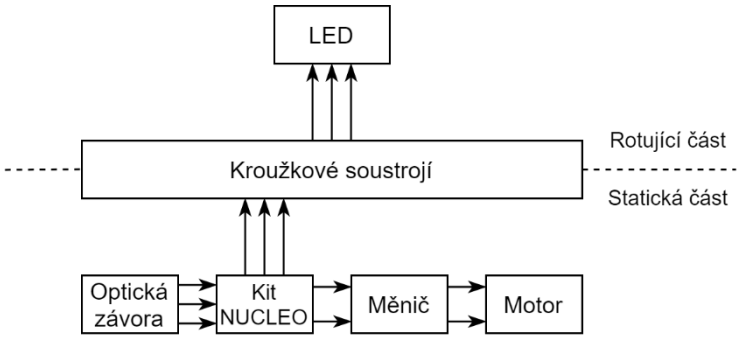
# 6 Varianta přípravku s RGB LED s datovou komunikací

Tato varianta vznikla za účelem zobrazování běžícího textu, který má libovolnou barvu, a proto byla osazena osmi RGB LED. Mechanická konstrukce byla vyhotovena na 3D tiskárně.



Obr. 6.1 – Přípravek osazený RGB LED

Stejně jako u varianty přípravku s diskrétními LED (popsaného v kapitole 5) tak i u této varianty není ze stejného důvodu kit NUCLEO umístěn na rotující části. K propojení kitu s RGB LED postačí pouze tři vodiče oproti pěti vodičům u varianty s diskrétními LED. Tato skutečnost může představovat výhodu v podobě jednoduššího kroužkového ústrojí.




Obr. 6.2 – Blokové zapojení přípravku s RGB LED

## 6.1 Použité RGB LED WS2812B

Tyto diody obsahují vlastní řídicí jednotku, která je řízena datovou komunikací v podobě přesně definovaných pulzů o určité periodě a šířce pulzu. Podle šířky pulzu LED určuje, zda se jedná o logickou jedničku či nulu. Tyto logické hodnoty poskládají binární číslo o velikosti 8 bitů pro každou barevnou LED (červená, zelená, modrá), které definuje jas každé barvy v pouzdru. Pokud toto číslo převedeme z binární do dekadické soustavy tak každá dílčí LED může svítit s intenzitou 0 (led nesvítí) nebo s intenzitou 255 (led svítí s plnou intenzitou). Výsledná barva RGB LED je složena z dílčích barev dle skládání barev podle modelu RGB. Složení datové zprávy pro jednu RGB LED je uvedeno na obrázku 6.3.

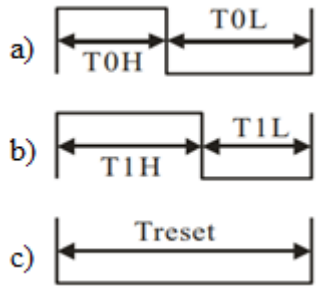
| G7 | G6 | G5 | G4 | G3 | G2 | G1 | G0 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Obr. 6.3 – Složení bitů pro jednu RGB LED

Přenos řídících bitů mezi jednotlivými LED probíhá sériovým způsobem. Pro RGB LED jsou směrodatné pouze bity, po kterých následuje série impulzů s nulovou šířkou pulzu, které trvají minimálně 50 µs. Po takovéto kombinaci dojde k dekódování datové zprávy a rozsvícení RGB LED požadovanou barvou.

Je nutné odesílat bity v pořadí barev GRB a také je nutné začínat vysíláním od nejvyššího bitu a poté postupovat dle obrázku 6.3.

Jednotlivé průběhy impulzů jsou vidět na obrázku 6.4 a v tabulce 6.1 jsou uvedeny doby trvání s jejich tolerancemi.




Obr. 6.4 – Průběhy impulzů [5]

| | | | |
|---|---|---|---|
| a) | Logická hodnota nula | T0H = | (0,4 ± 0,15) µs |
| | | T0L = | (0,85 ± 0,15) µs |
| b) | Logická hodnota jedna | T1H = | (0,8 ± 0,15) µs |
| | | T1L = | (0,45 ± 0,15) µs |
| c) | Sekvence pro rozsvícení LED | Treset = | > 50 µs |

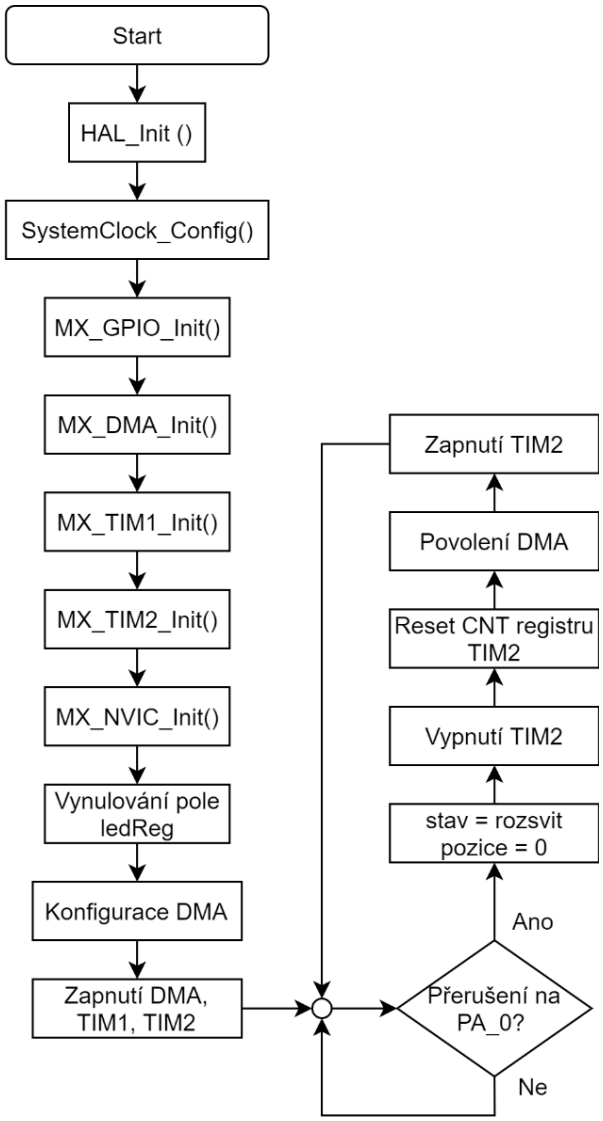
Tab. 6.1 – Doby trvání jednotlivých částí impulzů

## 6.2 Algoritmus pro obsluhu RGB LED

Algoritmus vychází ze základního požadavku na podobu impulzů, který klade zvolený typ LED viz kapitola 6.1. Tyto impulzy jsou tvořeny pomocí periferie TIM2 jakožto PWM výstup. Z důvodu, aby nedocházelo ke zpomalování programu při zápisu nové hodnoty na šířku pulzu pomocí procesoru je využito periferie DMA, která tento zápis provádí automaticky bez potřebné činnosti procesoru. Hodnoty pro šířky pulzů jsou předem vypočteny před zahájením vysílání PWM a uloženy v proměnné „ledReg“. Po povolení DMA a TIM2 se po každém přetečení čítače TIM2 (perioda pulzu) požádá DMA o uložení nové šířky pulzu do compare registru TIM2, dokud se neodvysílají všechna data.

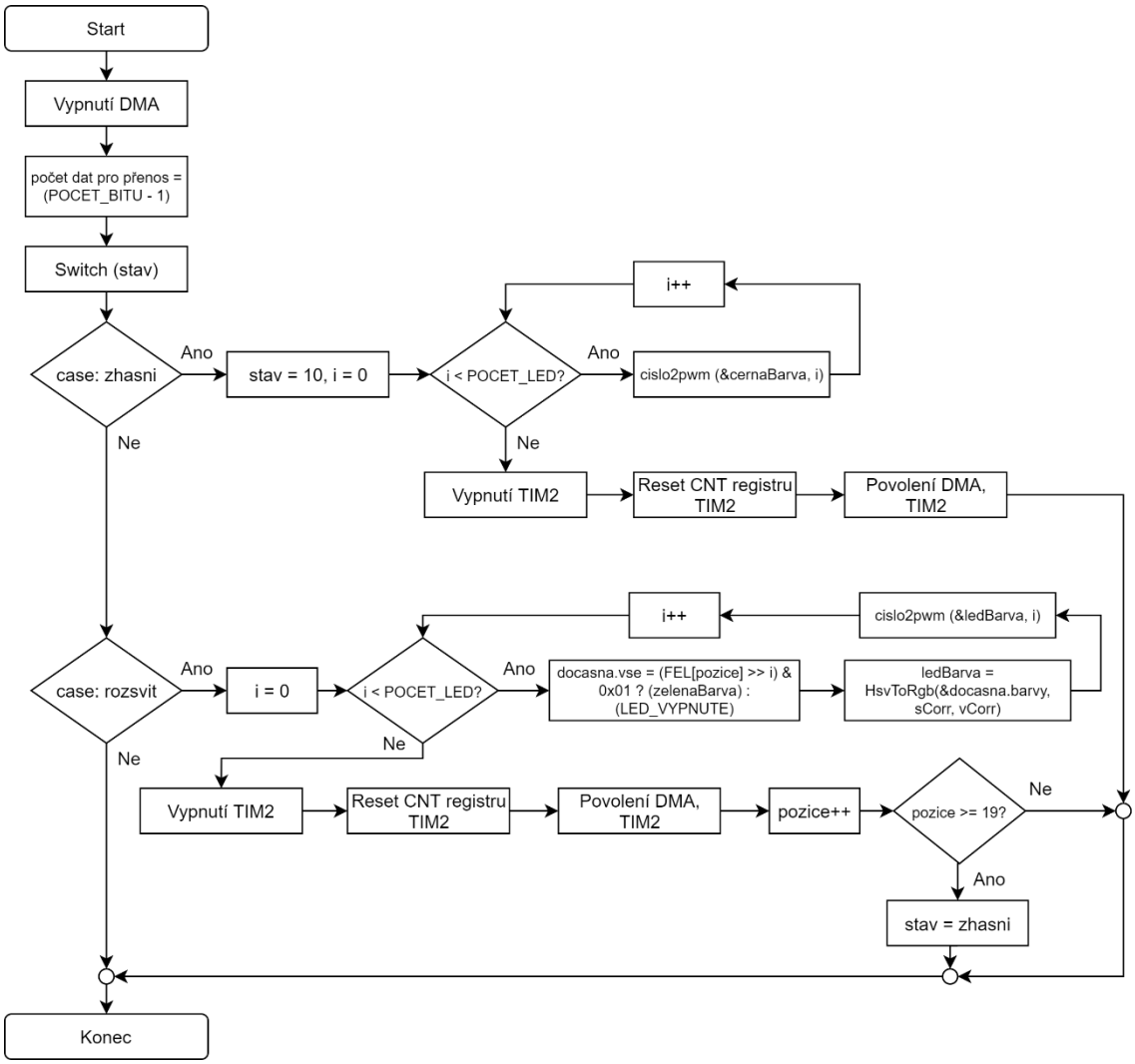
## 6.3 Vývojové diagramy programu

Program začíná inicializací používaných periferií a následně čeká na přerušení na pinu PA_0, které je generované průchodem rotující části skrz světelnou závoru. Následně probíhá obsluha přerušení, ve kterém odvysílá TIM2 svoji první kombinaci na RGB LED.



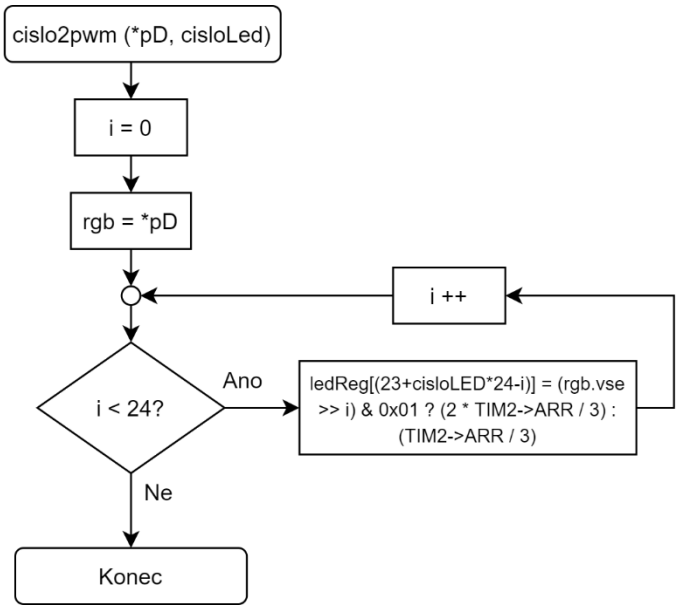

Obr. 6.5 – Vývojový diagram pro hlavní část programu

Ve chvíli, kdy je DMA přenos zcela ukončený se vygeneruje přerušení, jehož popis vývojovým diagramem je na obrázku 6.6. Přerušení od DMA bude detailněji vysvětleno v kapitole 6.4.




Obr. 6.6 – Vývojový diagram přerušení od DMA po ukončení přenosů

Funkcí cislo2pwm je plněna proměnná „ledReg“ hodnotami, které posléze DMA přenáší do TIM2 a tím se mění délka pulzu PWM. Hodnoty se volí na základě rozhodování, zda příslušný impulz má představovat logickou hodnotu nula či jedna. To, o kterou hodnotu se má jednat se docílí výpočtem dle požadavku, zda má daná LED svítit určitou barvou nebo být zhasnutá.




Obr. 6.7 – Vývojový diagram funkce cislo2pwm

## 6.4 Zdrojový kód

V přerušení od světelné závory se nastaví proměnná „stav“ na hodnotu „rozsvit“, která určí chod programu ve funkci case v DMA přerušení. Proměnnou „pozice“ se zde volí začátek pole, ve kterém jsou uloženy hodnoty pro běžící text. Poté dojde k zablokování TIM2, vynulování TIM2, povolení DMA a opětovnému zapnutí TIM2. Tato sekvence zajišťuje, aby TIM2 čítal od nuly a nikoliv od nějaké nespecifikované hodnoty, což by mohlo rozsvítit RGB LED jinou barvou.

```
470 void EXTI0_IRQHandler(void)
471 {
472   /* USER CODE BEGIN EXTI0_IRQn 0 */
473     stav = rozsvit;
474     pozice = 0;
475
476     TIM2->CR1 &= ~TIM_CR1_CEN;
477     TIM2->CNT = 0x00;
478     DMA1_Channel5->CCR |= DMA_CCR_EN;
479     TIM2->CR1 |= TIM_CR1_CEN;
480
481   /* USER CODE END EXTI0_IRQn 0 */
482     HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
483   /* USER CODE BEGIN EXTI0_IRQn 1 */
484
485   /* USER CODE END EXTI0_IRQn 1 */
486 }
```

Obr. 6.8 – Obsluha přerušení vyvolaná světelnou závorou

Po přenosu všech dat určených pro DMA se vygeneruje obsluha přerušení DMA. V tomto přerušení se nejdříve zablokuje přenos DMA a potom se nastaví nový počet dat pro přenos do registru CNDTR. Podle uložené hodnoty v proměnné „stav“ program pokračuje příslušným case.

Case „rozsvit“ slouží k odvysílání běžícího textu. V cyklu for se postupně volá funkce cislo2pwm, která plní proměnnou „ledReg“ hodnotami pro všechny RGB LED. Tyto hodnoty jsou voleny na základě podmínky, zda příslušná RGB LED má svítit určitou barvou nebo má být zhasnutá. Po výpočtu dat pro všechny RGB LED se zablokuje TIM2, vynuluje TIM2, povolí DMA a spustí TIM2, čímž se začnou vysílat data na LED. Po odvysílání všech dat se opět vygeneruje DMA

přerušení. V case „rozsvit“ se program pohybuje, dokud TIM2 neodvysílá data pro všechny znaky běžícího textu, poté se nastaví proměnná „stav“ na hodnotu „zhasni“.

Case „zhasni“ je konečný stav, který zhasne všechny RGB LED a který nastaví proměnnou „stav“ na hodnotu „10“ což zařídí při dalším DMA přerušení, že program půjde do default. Nyní je celý běžící text odvysílán a čeká se na další průchod světelnou závorou.

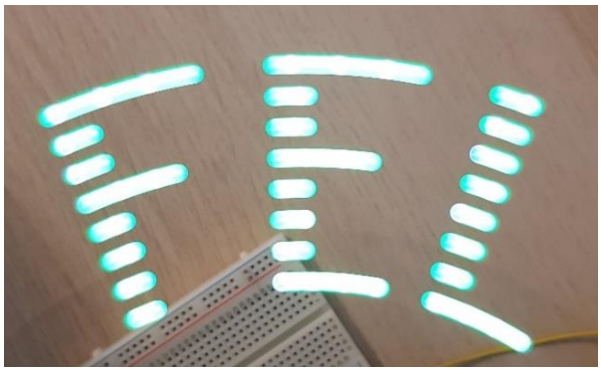
```
411 void DMA1_Channel5_IRQHandler(void)
412 {
413   /* USER CODE BEGIN DMA1_Channel5_IRQn 0 */
414     DMA1_Channel5->CCR &= ~DMA_CCR_EN;
415     DMA1_Channel5->CNDTR = (POCET_BITU-1);
416     switch (stav)
417     {
418         case rozsvit:
419         {
420             for(uint8_t i = 0; i < POCET_LED; i++)
421             {
422                 docasna.vse = (FEL[pozice] >> i) & 0x01 ? (zelenaBarva) : (LED_VYPNUTE);
423                 ledBarva = HsvToRgb(&docasna.barvy, sCorr, vCorr);
424 
425                 cislo2pwm(&ledBarva, i);
426             }
427             TIM2->CR1 &= ~TIM_CR1_CEN;
428             TIM2->CNT = 0x00;
429             DMA1_Channel5->CCR |= DMA_CCR_EN;
430             TIM2->CR1 |= TIM_CR1_CEN;
431             pozice++;
432 
433             if (pozice >= 19)
434                 stav = zhasni;
435             break;
436         }
437         case zhasni:
438         {
439             stav = 10;
440             for (uint8_t i = 0; i < POCET_LED; i++)
441             {
442                 cislo2pwm(&cernaBarva, i);
443             }
444             TIM2->CR1 &= ~TIM_CR1_CEN;
445             TIM2->CNT = 0x00;
446             DMA1_Channel5->CCR |= DMA_CCR_EN;
447             TIM2->CR1 |= TIM_CR1_CEN;
448             break;
449         }
450         default:
451         {
452             break;
453         }
454     }
455   /* USER CODE END DMA1_Channel5_IRQn 0 */
456   /* USER CODE BEGIN DMA1_Channel5_IRQn 1 */
457     DMA1->IFCR = DMA_IFCR_CGIF5;
458     NVIC_ClearPendingIRQ(DMA1_Channel5_IRQn);
459   /* USER CODE END DMA1_Channel5_IRQn 1 */
460 }
```

Obr. 6.9 – Obsluha přerušení od DMA

Funkce cislo2pwm plní proměnnou „ledReg“ hodnotami, které potom DMA přesouvá do CCR1 registru TIM2. Ukládání dat do „ledReg“ probíhá od nejnižšího bitu po nejvyšší pro každou RGB LED.

```
401 void cislo2pwm (T_RgbBarva *pD, uint8_t cisloLED)
402 {
403     T_RgbBarva rgb;
404     rgb = *pD;
405     for (uint8_t i = 0; i < 24; i++)
406     {
407         ledReg[(23+cisloLED*24-i)] = (rgb.vse >> i) & 0x01 ? (2 * TIM2->ARR / 3) : (TIM2->ARR / 3);
408     }
409 }
```

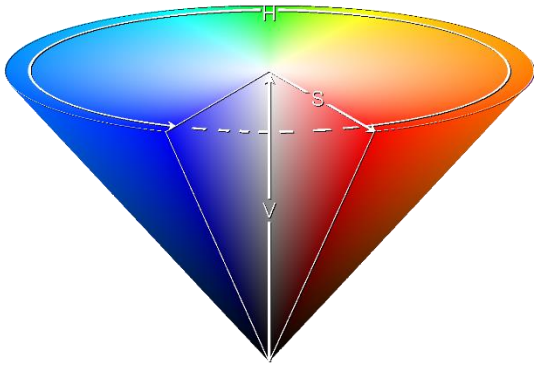
Obr. 6.10 – Funkce cislo2pwm



Obr. 6.11 – Ukázka běžícího textu

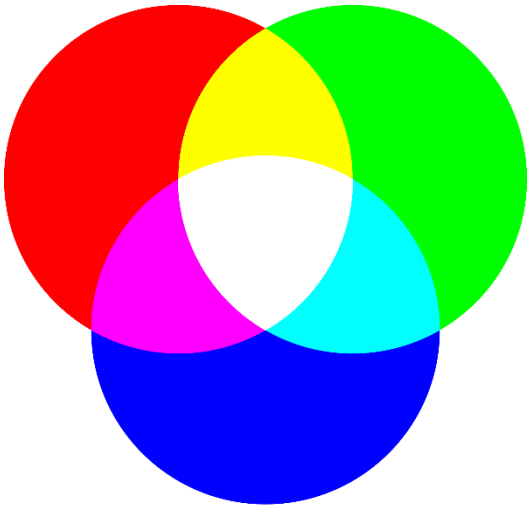
## 6.5 Funkce HsvToRgb a RgbToHsv

Ve stavovém diagramu na obrázku 6.6 se vyskytují navíc dvě funkce HsvToRgb a RgbToHsv. Tyto funkce byly vloženy kvůli univerzálnosti kódu a dalším možným aplikacím s proměnnými barvami. Volba barvy LED v barevném modelu HSV se skládá ze tří hodnot, a to z odstínu (H), sytosti barvy (S) a z hodnoty jasu (V). Výsledná barva je tvořena dle obrázku 6.12.




Obr. 6.12 – Kuželová reprezentace HSV modelu [6]

Výhoda HSV modelu spočívá v tom, že zvolením nulového odstínu lze dosáhnout výpočtem černé barvy (LED přestanou svítit), čehož skládáním barev v modelu RBG nelze dosáhnout viz obrázek 6.13 a do programu by musela být vložena funkce pro zhasnutí LED.



Obr. 6.13 – Skládání barev podle modelu RGB [7]

Dle kapitoly 6.1 zvolené RGB LED ovšem vyžadují barvu v modelu RGB. Pro přepočet barvy z HSV do RGB slouží funkce HsvToRgb. Opačně funguje funkce RgbToHsv, která provádí přepočet z RGB do HSV modelu. Obě tyto funkce byly převzaty z [8] a mírně upraveny dle potřeb programu.

# 7 Závěr

Cílem této bakalářské práce bylo vytvoření programu, který by umožňoval tvorbu běžícího textu. Byly vytvořeny dva přípravky z důvodu použití dvou variant LED kvůli možnosti porovnání kvality zobrazovaného textu.

První variantou použitých LED jsou LED diskrétní. Výhoda těchto LED spočívá v jejich jednoduchém ovládání což se odráží ve složitosti programu. Jejich nevýhodou v porovnání s druhou variantou je omezení výběru barvy textu pouze na jednu barvu a také nutnost přenášet signál od statické části do rotující části pomocí pěti kroužků namísto třech.

Druhou variantou jsou RGB LED s vlastní řídící jednotkou s datovou komunikací. Velkou výhodou je možnost zobrazování textu téměř jakoukoliv barvou, což otevírá nové možnosti vzhledu textu. Výhoda volby barvy je ovšem vykoupena poněkud většími nároky na komplikovanost programu.

Na jednoduchém demonstračním nápisu bylo pozorováno, že vizuální kvalita a čitelnost textu se na obou přípravcích výrazně neliší a je proto téměř stejná.

Jako příklad možného rozšíření této práce bych viděl přidání některých funkcí jako například měření a následné zobrazování otáček rotující části nebo zobrazování textu v reálném čase psaného uživatelem na PC a následně převedeného do kitu NUCLEO některým z dostupných způsobů komunikace kitu a PC.

# 8 Seznam použité literatury

[1] *STM32F302x6/8 Reference manual*. Revision 7. Geneva, Switzerland, 2017. Dostupné také z: https://www.st.com/resource/en/reference_manual/dm00094349-stm32f302xbcde-and-stm32f302x68-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[2] VOŽENÍLEK, Petr, Vladimír NOVOTNÝ a Pavel MINDL. *Elektromechanické měniče*. 2. vyd. Praha: České vysoké učení technické v Praze, 2011. ISBN 978-80-01-04875-7.

[3] PAVELKA, Jiří, Zdeněk ČEŘOVSKÝ a Jiří LETTL. *Výkonová elektronika*. Vyd. 3., přeprac. Praha: Nakladatelství ČVUT, 2007. ISBN 978-80-01-03626-6.

[4] *Product data sheet of 8-bit shift register with output register*. Nijmegen, Netherlands, 2017. Dostupné také z: https://assets.nexperia.com/documents/data-sheet/74HC_HCT594.pdf

[5] *WS2812B Datasheet and Specifications*. DaLingShan Town, China, 2016. Dostupné také z: https://voltiq.ru/datasheets/WS2812B_datasheet_EN.pdf

[6] Kuželová reprezentace HSV modelu. In: *Wikipedie* [online]. San Francisco, USA: Wikimedia Foundation, 2019 [cit. 2020-07-10]. Dostupné z: https://commons.wikimedia.org/wiki/File:HSV_cone.png

[7] RGB color model. In: *Wikipedie* [online]. San Francisco, USA: Wikimedia Foundation, 2019 [cit. 2020-07-22]. Dostupné z: https://commons.wikimedia.org/wiki/File:RGB_color_model.svg

[8] Algorithm to convert RGB to HSV and HSV to RGB in range 0-255 for both. *Stackoverflow* [online]. London, England: Stack Overflow, 2010 [cit. 2020-07-10]. Dostupné z: https://stackoverflow.com/questions/3018313/algorithm-to-convert-rgb-to-hsv-and-hsv-to-rgb-in-range-0-255-for-both

# 9 Seznam použitých zkratek

- CPU – Central processing unit
- VDD – Voltage drain drain
- GPIO – General-purpose input/output
- ADC – Analog-to-Digital converter
- DAC – Digital-to-Analog converter
- RTC – Real-time clock
- DMA – Direct memory access
- TIM – Timer
- PWM – Pulse width modulation

# Přílohy

- Příloha 1 – Programy pro tvorbu běžícího textu spustitelné v programu STM32CubeIDE
  - Program pro diskrétní LED
  - Program pro datové RGB LED
- Příloha 2 – Výňatek z datasheetu shift registru dle [4]



## 4. Ordering information

Table 1. Ordering information

| Type number | Package | | | |
|---|---|---|---|---|
| | Temperature range | Name | Description | Version |
| 74HC594D | −40 °C to +125 °C | SO16 | plastic small outline package; 16 leads; body width 3.9 mm | SOT109-1 |
| 74HCT594D | | | | |
| 74HC594DB | −40 °C to +125 °C | SSOP16 | plastic shrink small outline package; 16 leads; body width 5.3 mm | SOT338-1 |
| 74HCT594DB | | | | |

## 5. Functional diagram

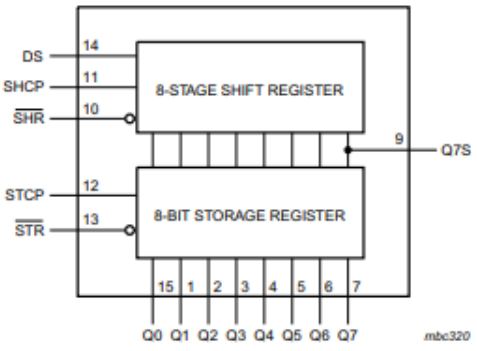



Fig 1. Functional diagram

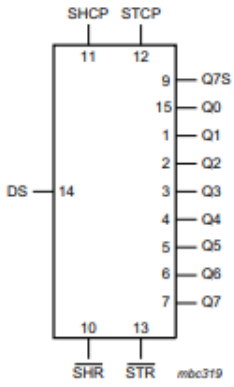



Fig 2. Logic symbol

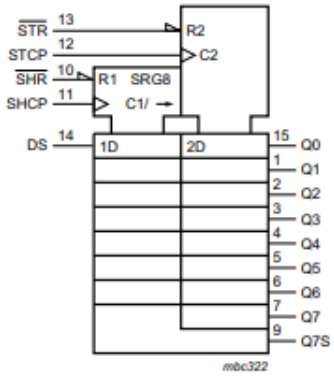



Fig 3. IEC logic symbol

# 6. Pinning information

## 6.1 Pinning

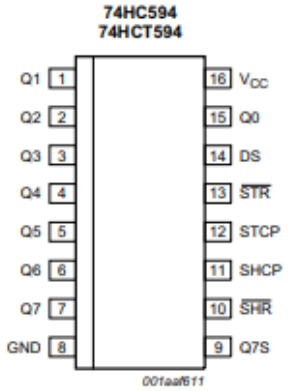



Fig 6. Pin configuration SO16

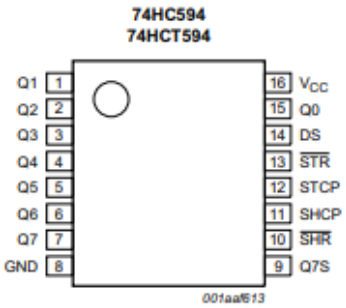



Fig 7. Pin configuration SSOP16

## 6.2 Pin description

Table 2. Pin description

| Symbol | Pin | Description |
|---|---|---|
| Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 | 15, 1, 2, 3, 4, 5, 6, 7 | parallel data output |
| GND | 8 | ground (0 V) |
| Q7S | 9 | serial data output |
| $\overline{SHR}$ | 10 | shift register reset (active LOW) |
| SHCP | 11 | shift register clock input |
| STCP | 12 | storage register clock input |
| $\overline{STR}$ | 13 | storage register reset (active LOW) |
| DS | 14 | serial data input |
| $V_{CC}$ | 16 | supply voltage |


All information provided in this document is subject to legal disclaimers.
© Nexperia B.V. 2017. All rights reserved

## 7. Functional description

Table 3. Function table[1]

| Function | Input | | | | |
|---|---|---|---|---|---|
| | $\overline{SHR}$ | $\overline{STR}$ | SHCP | STCP | DS |
| Clear shift register | L | X | X | X | X |
| Clear storage register | X | L | X | X | X |
| Load DS into shift register stage 0, advance previous stage data to the next stage | H | X | ↑ | X | H or L |
| Transfer shift register data to storage register and outputs Qn | X | H | X | ↑ | X |
| Shift register one count pulse ahead of storage register | H | H | ↑ | ↑ | X |

[1] H = HIGH voltage level;
L = LOW voltage level;
↑ = LOW-to-HIGH transition;
X = don't care.

## 8. Limiting values

Table 4. Limiting values
*In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to GND (ground = 0 V).*

| Symbol | Parameter | Conditions | | Min | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{CC}$ | supply voltage | | | −0.5 | +7.0 | V |
| $I_{IK}$ | input clamping current | $V_I < -0.5$ V or $V_I > V_{CC} + 0.5$ V | [1] | - | ±20 | mA |
| $I_{OK}$ | output clamping current | $V_O < -0.5$ V or $V_O > V_{CC} + 0.5$ V | [1] | - | ±20 | mA |
| $I_O$ | output current | $V_O = -0.5$ V to $V_{CC} + 0.5$ V | | | | |
| | | Serial data output Q7S | | - | ±25 | mA |
| | | Parallel data output | | - | ±35 | mA |
| $I_{CC}$ | supply current | Serial data output Q7S | | - | 50 | mA |
| | | Parallel data output | | - | 70 | mA |
| $I_{GND}$ | ground current | Serial data output Q7S | | - | −50 | mA |
| | | Parallel data output | | - | −70 | mA |
| $T_{stg}$ | storage temperature | | | −65 | +150 | °C |
| $P_{tot}$ | total power dissipation | $T_{amb} = -40$ °C to +125 °C | [2] | - | 500 | mW |

[1] The input and output voltage ratings may be exceeded if the input and output current ratings are observed.

[2] For SO16 packages: above 70 °C the value of $P_{tot}$ derates linearly with 8 mW/K.
For SSOP16 packages: above 60 °C the value of $P_{tot}$ derates linearly with 5.5 mW/K.


All information provided in this document is subject to legal disclaimers.
© Nexperia B.V. 2017. All rights reserved

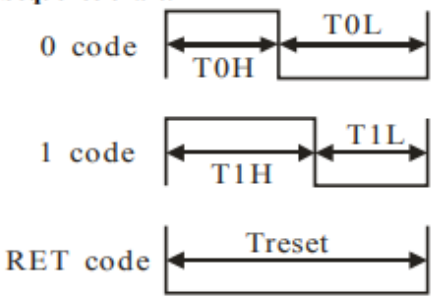
- Příloha 3 – Výňatek z datasheetu RGB LED dle [5]



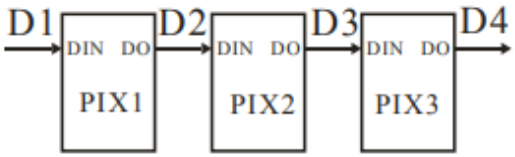
**Data transfer time(** TH+TL=1.25μs±600ns)

| | | | |
|---|---|---|---|
| T0H | 0 code ,high voltage time | 0.4us | ±150ns |
| T1H | 1 code ,high voltage time | 0.85us | ±150ns |
| T0L | 0 code , low voltage time | 0.85us | ±150ns |
| T1L | 1 code ,low voltage time | 0.4us | ±150ns |
| RES | low voltage time | Above 50μs | |

**Sequence chart:**




**Cascade method:**



Note: The data of D1 is send by MCU,and D2, D3, D4 through pixel internal reshaping amplification to transmit.

**Composition of 24bit data:**

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G7 | G6 | G5 | G4 | G3 | G2 | G1 | G0 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |

Note: Follow the order of GRB to sent data and the high bit sent at first.

**Typical application circuit:**