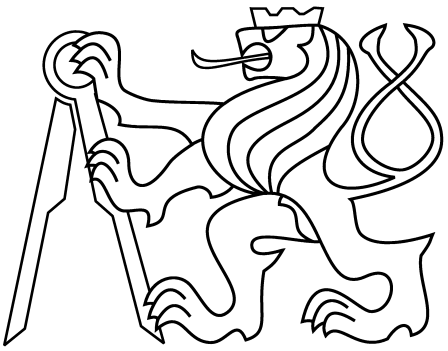
Diplomová práce

26. května 2023

Laboratorní měřicí přístroj s mikrořadičem STM32G031 pro výukové účely

Jan Bittman



Vedoucí práce: doc. Ing. Jan Fischer, CSc.

České vysoké učení technické v Praze Fakulta elektrotechnická,
Katedra mikroelektroniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bittman** Jméno: **Jan** Osobní číslo: **466243**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Laboratorní měřicí přístroj s mikrořadičem STM32G031 pro výukové účely

Název diplomové práce anglicky:

Laboratory Measuring Instrument with STM32G031 Microcontroller for Educational Purposes

Pokyny pro vypracování:

Navrhněte a s využitím mikrořadiče STM32G031 realizujte univerzální laboratorní měřicí přístroj pro výuku základů elektroniky. Funkce přístroje budou: tříkanálový osciloskop s možností vzorkování v reálném i ekvivalentím čase, tříkanálový voltmetr, dvoukanálový impulsní generátor PWM a čítač pro měření frekvence, délky a střídy impulsů. Pro vzorkování v ekvivalentním čase s využitím interního impulsního generátoru zajistěte možnost jemného nastavování frekvence vzorkování i frekvence generovaného signálu. Napájení přístroje a jeho ovládání bude z PC prostřednictvím rozhraní USB a převodníku USB/ UART. Jako ovládací PC aplikaci využijte program Dataplotter [3]. Zařízení otestujte, proveďte ověřovací měření a zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

[1] Yiu, J.: The Definitive Guide to ARM Cortex -M0 and Cortex-M0+ processors; 2015
[2] STMicroelectronics: STM32G0x1 RM0444 Reference manual; 2020
[3] Maier, J.: Univerzální GUI pro osciloskopické PC aplikace; ČVUT- FEL, 2021

Jméno a pracoviště vedoucí(ho) diplomové práce:

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

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

Datum zadání diplomové práce: **14.02.2023** Termín odevzdání diplomové práce: **26.05.2023**

Platnost zadání diplomové práce: **22.09.2024**

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

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání Podpis studenta

Poděkování

Tímto děkuji vedoucímu práce doc. Ing. Janu Fischerovi, CSc. za ochotu a čas vložený do realizace této diplomové práce, především za vřelé a ochotné rady předané během konzultací.

Prohlášení

Tato práce vznikla v laboratoři videometrie, katedry měření ČVUT - FEL v Praze pod vedením doc. Ing. Jana Fischera, CSc. 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

Náplní této diplomové práce je navrhnout, a s použitím mikrokontroléru STM32G031 realizovat, univerzální laboratorní měřící přístroj pro výuku základů elektroniky. Přístroj obsahuje funkce tříkanálového osciloskopu, voltmetru , dvoukanálového impulsního generátoru PWM a čítače pro měření frekvence a střídy signálu. Pro vzorkování v ekvivalentním čase s využitím interního impulzního generátoru je zajištěno jemné nastavovaní frekvence vzorkování i generovaného signálu. Jako ovládací aplikace slouží univerzální PC program Data Plotter.

Klíčová slova

Softwarově definované měřící přístroje, osciloskop, pulsní generátor, čítač frekvence, voltmetr, mikrokontrolér, STM32, výuka.

Abstract

The aim of this master's thesis is to design and implement a universal laboratory measuring instrument for teaching the basics of electronics using the STM32G031 microcontroller. The instrument includes the functions of a three-channel oscilloscope, a voltmeter, a two-channel PWM pulse generator, and frequency and duty cycle counters for signal measurement. Fine adjustment of the sampling frequency and generated signal is ensured for sampling in equivalent time using the internal pulse generator. The universal PC program Data Plotter serves as the control application.

Key words

Software-defined measuring instruments, oscilloscope, pulse generator, frequency counter, voltmeter, microcontroller, STM32, education.

Obsah

Seznam obrázků

1 Úvod

Výkon dostupných mikrokontrolérů každým rokem roste a jejich vybavenost periferiemi, jako jsou např. analogově digitální převodníky ADC, čítače a komunikační rozhraní. To umožňuje realizaci softwarově definovaných přístrojů. Ty mohou vykonávat například funkce běžného voltmetru, generátoru pulzních signálů nebo osciloskopu. Všechny tyto funkce jsou tak realizovány pouze s využitím vnitřních periférií mikrokontroléru s příslušným specializovaným firmware.Pro toto řešení se používá označení SDI - Software defined instrument.

Úkolem této práce je proto vytvořit softwarově definovaný přístroj (SDI) na platformě mikrokontrolérů STM32. Konkrétně pak na řadě STM32G031. Ač se jedná o velice malý mikrokontrolér, disponuje dostatečným výpočetním výkonem a velikostí paměti, spolu s několika časovači, ADC převodníkem i sériovým rozhraním USART. Proto bude pro úspěšnou realizaci postačující.

Tato práce navazuje na řadu podobných projektů, zabývajících se tvorbou SDI, vytvořených na katedře měření v laboratoři videometrie.

2 Rozbor práce

Úkolem této diplomové práce je navrhnout a vytvořit univerzální laboratorní přístroj s využitím mikrokontroléru STM32G031 pro využití ve výuce základů elektroniky. Přístroj má být schopen zastávat funkci tříkanálového osciloskopu s možností vzorkování v reálném i ekvivalentním čase, voltmetru s možností průměrování, dvoukanálového PWM generátoru a čítače pro měření střídy a napětí.

Konkrétně pak funkce osciloskopu a generátoru PWM signálu má být schopna jemného nastavování vzorkovací frekvence i frekvence generovaného signálu. Jakožto komunikační rozhraní má být využita sériová linka USART s pomocí UART-USB převodníku. Jako ovládací PC aplikace mám za úkol využít program Data Plotter[1]. Zařízení mám otestovat a provést zkušební měření.

2.1 Univerzální zobrazovací aplikace Data Plotter

Data Plotter je univerzální PC aplikace, která je schopna vizualizace příchozích dat skrze sériový port. Tato aplikace je primárně určena na tvorbu softwarové definovaných měřících zařízení pomocí mikrokontrolérů. Aplikace sama o sobě umožňuje různou manipulaci s přijatými daty, jako je přiblížení, oddálení, nebo funkce jako je FFT, či průměrování. Na tvůrci softwaru pro mikrokontrolér je pak vytvoření uživatelského rozhraní pro ovládání mikrokontroléru. To lze vytvořit jednak terminálově (Data Plotter umožňuje zobrazení do terminálového okna), nebo za pomoci jazyka QML. Soubor ‚popisující uživatelské rozhraní v jazyce QML, je do Data Plotteru nahrán po připojení z mikrokontroléru. Tento soubor následně aplikace zobrazí v separátním okně, skrze nějž může uživatel komunikovat s mikrokontrolérem, např ovládat různé funkce ap.

2.2 Zvolený mikrokontrolér

Pro realizaci byl využit mikrokontrolér STM32G031J6. Jedná se o 32bitový mikrokontrolér od firmy STMicroelectronics, konkrétně o verzi v pouzdře SO8N, tedy o velmi malou součástku. V takto malém pouzdře je uschováno 32kB paměti FLASH, jádro ARM-Cortex M0+ o frekvenci až 64MHz, paměť SRAM o velikosti 8kB a mnoho periferií, jako jsou čítače, sériové rozhraní USART, I2C, SPI, nebo analogově digitální převodník ADC. Takto malý integrovaný obvod je schopen číst analogové napětí na několika multiplexovaných kanálech a následně ho pomocí rozhraní USART posílat do PC, kde jsou následně data zobrazena pomocí aplikace Data Plotter.

2.3 Vývoj softwaru pro mikrokontrolér

Pro vývoj firmwaru je využito integrované vývojové prostředí Keil MDK s využitím kompilovaného jazyka C. Celý firmware je napsán bez pomoci abstraktních knihoven, pouze přímým ovládáním registrů mikrokontroléru. Vzhledem k velmi malému pouzdru jsem pro vývoj používal mikrokontrolér STM32G031F6. Jedná se o identický mikrokontrolér, pouze ve větším pouzdře. Tím jsem byl schopen implementovat veškeré funkcionality a zároveň mít mikrokontrolér připojený k ladícímu rozhraní SWD (Serial wire debug). Díky tomuto rozhraní lze jednoduše krokovat běh programu, samotný program nahrávat a sledovat jeho chod za běhu. Na tomto pouzdře jsem tedy vše otestoval a odladil. Následně jsem hotový firmware nahrál do menšího pouzdra. Výsledný software je tedy možné používat jak na 20pinovém pouzdře, tak 8mi pinovém. Na 8mi pinovém pouzdře je pouze potřeba nastavit při prvním nahrání tzv. Optien bytes (OB) a vypnout tak funkci reset pinu.

3 Data Plotter

Data Plotter je dílem bakalářské práce pana Bc. Jiřího Majera zhotovená na FEL ČVUT pod katedrou měření. Jedná se o víceúčelovou aplikaci pro tvorbu softwarově definovaných měřících zařízení. Tedy zařízení, které v sobě implementuje, jak samotný firmware pro odebírání vzorků, tak i uživatelské rozhraní které se zobrazuje v aplikaci Data Plotter.

3.1 Dostupné druhy uživatelského rozhraní

V aplikaci je možné zobrazovat uživatelské rozhraní dvěma různými způsoby. Buďto klasickým "terminálovým"způsobem, která Data Plotter umožňuje zobrazovat v jednom ze svých oken, nebo za pomoci QML souborů (vice viz 4). Oba druhy uživatelského rozhraní mají jisté výhody a nevýhody.

3.2 Terminálové uživatelské rozhraní

Za pomoci terminálového uživatelského rozhraní lze vytvořit obdobu ovládání, jako přes program "Putty". Tedy jednoduché znakové rozhraní, které podporuje zobrazování ANSI escape sekvencí. Tedy je možné měnit barvy pozadí, přepisovat jednotlivé znaky ap.. Zároveň v aplikaci Data Plotter lze na jednotlivé znaky v terminálovém okně kliknout, načež je daný znak odeslán do mikrokontroléru. Tímto je zajiště zpětná komunikace s mikrokontrolérm potřebná pro změny jeho nastavení.

Klady

- Jednoduchý způsob zobrazení
- Možná kompatibilita s terminálem Putty
- Bez nutnosti použití jazyka QML

Zápory

- Složité zobrazení zpětné vazby po stisknutí
- Nemožnost využití složitějších ovládacích prvků (slider, textové pole ap.)
- Vyšší náročnost na výpočetní výkon mikrokontroléru

3.3 Uživatelské rozhraní pomocí jazyka QML

V tomto případě je pro definování uživatelského rozhraní použit jazyk QML, konkrétně je daný .qml soubor převeden do binární podoby a odeslán do mikrokontroléru. Tím lze vytvořit uživatelské rozhraní ve stejném stylu jako zbytek aplikace, tedy výsledek vypadá "jako by tam patřil". Díky využití tohoto uživatelského rozhraní lze do ovládání implementovat textové zadávání hodnoty, přepínače, slidery ap. (více v sekci 4). Na začátku běhu programu je pouze potřeba celý soubor do aplikace Data Plotter nahrát skrze rozhraní USART. Následné uživatelské rozhraní je již kompletně zobrazováno přímo PC aplikací, tedy není potřeba se o jeho spolehlivý chod nijak starat.

Klady

- Hezké uživatelské prostředí zapadající vizuálně do zbytku aplikace
- Menší náročnost na výpočetní výkon mikrokontroléru
- Jednodušší implementace vnořených ovládacích oken

Zápory

- Složitější implementace
- Nemožná kompatibilita s jiným terminálovým zobrazovačem
- Nutné odeslání souboru do aplikace po spuštění

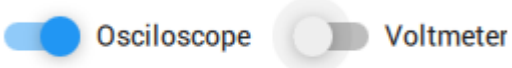
4 Jazyk QML

Jazyk QML, neboli Qt Modelling Language, je proprietární jazyk využívaný k popisu uživatelských rozhraní v Qt frameworku. Jazyk v sobě implementuje jak popis samotných elementů uživatelského rozhraní, jako jsou tlačítka, textová pole ap, tak funkcionality jazyka JavaScript, díky němuž lze přímo elementy v okně ovládat.

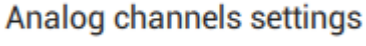
Jelikož se jedná o skriptovací jazyk využívaný pro tvorbu uživatelských rozhraní v Qt frameworku, tak není tolik známý mezi studenty FEL. Proto byl zároveň požadavek vedoucího práce, aby tato práce mohla sloužit jako návod, jak začít jazyk používat, pro další studenty.

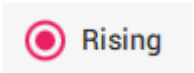
Jazyk QML je velice jednoduchý a intuitivní. Základní část zdrojového souboru se skládá z rozložení jednotlivých ovládacích prvků. Z ovládacích prvků máme na výběr mnoho variant, počínaje tlačítky, přepínači, textovými poli, rádiovými tlačítky, "slidery", nebo nápisy až k složitějším prvkům, sdružující více jednoduchých, jako jsou panely, nebo menší vyskakovací podokénka .

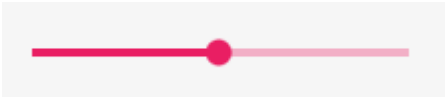
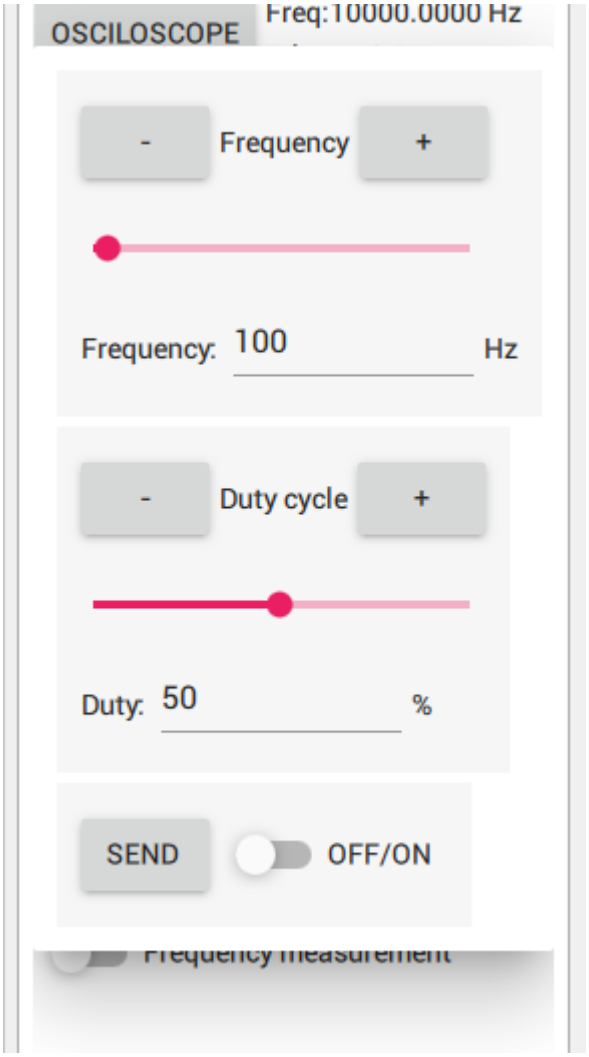
- Tlačítko 

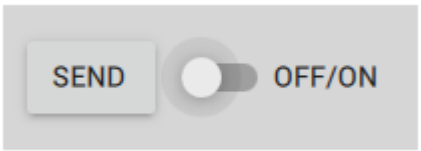
- Přepínač 

- Textové pole 

- Nápisy 

- Rádiové tlačítko 

- Slider 
- Vyskakovací okno 

- Panel 

Ve výsledném firmwaru je využito rozložení skládající se z vertikálně uspořádaných tlačítek vztahujících se k jednotlivým funkcionalitám, jako je osciloskop, voltmetr nebo PWM generátor. Po následném zakliknutí daného tlačítka se zobrazí vyskakovací okno, kde je sdružené veškeré ovládání daného segmentu. V zmíněném vyskakovacím okně je vždy na spodní části tlačítko, které potvrdí výběr nastavení a následně odešle konfigurační řetězec do mikrokontroléru, který na jeho základně přenastaví danou funkcionalitu. Jedná se například o střídu či frekvenci PWM signálu, nebo vzorkovací frekvence osciloskopu, či výběr jednotlivých kanálu používaných v osciloskopu.

Jednotlivé ovládací prvky lze skládat buďto do řádků, či sloupců. Zároveň lze nastavit jejich odsazení, velikost či barvu. To usnadňuje následnou orientaci v uživatelském rozhraní a zjednodušuje celkově ovládání zařízení.

4.1 Vytvoření ovládacího prvku

V základu vzorového projektu pro Data Plotter se nachází jedno okno, do nějž jsem následně vkládal jednotlivé prvky. Ovládací prvek je definován pomocí klíčového slova prvku viz 1.

```
RowLayout
    {
        Switch {
            id: osciloscope_switch
            text: "Osciloscope"
            checked: true
            onClicked: if (osciloscope_switch.checked) {
                voltmeter_switch.checked = false
                osciloscope_button.enabled = true
                voltmeter_button.enabled = false
                send("o_r;")
                }
            }
        Switch {
            id: voltmeter_switch
            text: "Voltmeter"
            onClicked: if (voltmeter_switch.checked) {
                osciloscope_switch.checked = false
                osciloscope_button.enabled = false
                voltmeter_button.enabled = true
                send("v_r;")
                }
            }
    }
```

Zdrojový kód 1: Vytvoření QML prvků seřazených horizontálně

U daného prvku lze zprvu nadefinovat výchozí nastavení, jako je zobrazovaný text spolu s jeho velikostí a barvou, stav daného prvku (zda je aktivní, či ne) a zároveň mu přiřadit funkci volanou ,například při stisku tlačítka. Tato funkce je v podstatě čistý JavaScript volaný při dané události.

Jak je vidět v ukázce č.4.1, jednotlivé prvky jsou zde seřazeny pomocí klíčové slova "RowLayout"horizontálně vedle sebe. Naopak vertikálně je lze řadit pomocí "ColumnLayout"

Obr. 4.1: Vytvořené přepínače

V následující ukázce kódu (viz č. 2) je vidět definice vyskakovacího okna, na němž je umístěn panel s tlačítkem. Panelu je zároveň přiřazena funkcionalita, kdy pozadí lehce změní barvu, pokud nad ním uživatel projede myší. Toto je velmi užitečné k jednodušší orientaci pro uživatele, neboť je léúe vidět s jakým prvkem interaguje.

```
Popup {
        id: osciloscope_popup
        anchors.centerIn: parent
        ColumnLayout {
            Pane {
                background: Rectangle {
                    color: parent.down ? "#bbbbbb" :
                    (parent.hovered ? "#d6d6d6" : "#f6f6f6")
                }
                ColumnLayout {
                    Button{
                    id:poupu_button
                    text: "TEST"
                    }
                    }
                }
        }
    }
```

Zdrojový kód 2: Vytvoření vyskakovacího okna v jazyce QML

Vyskakovací okno lze zobrazit metodou "open()"zavolanou na daný objekt vyskakovacího okna, viz ukázka č.3.

```
Button {
                text: "Osciloscope"
                id: osciloscope_button
                onClicked: osciloscope_popup.open()
            }
```

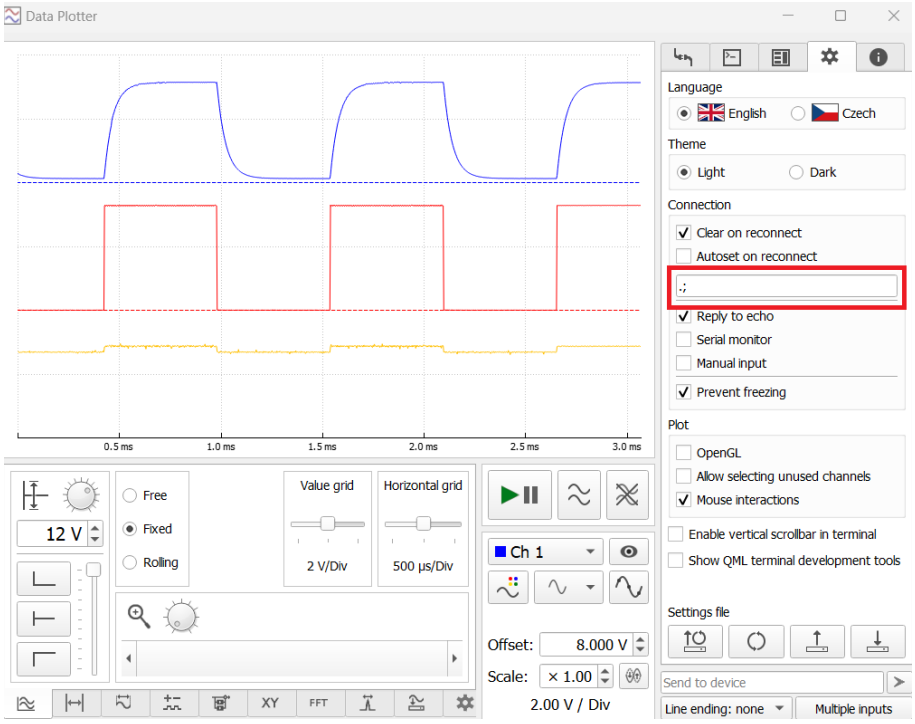
Zdrojový kód 3: Vytvoření vyskakovacího okna v jazyce QML

Ve vzorovém projektu QML souboru je předpřipraveno "API"pro komunikaci s mikrokontrolérem. Pomocí příkazu "$$V(jméno):(hodnota)"lze zapisovat hodnoty do proměnných deklarovaných v QML souboru. Tyto hodnoty lze použít pro zpětnou komunikaci z mikrokontroléru do aplikace Data Plotter, konkrétně s danými prvky v uživatelském rozhraní. V mé práci byla tato komunikace využita například k zobrazování změřených frekvencí, či nastavených hodnot reálné vzorkovací frekvence osciloskopu.

4.2 Nahrání QML souboru do aplikace Data Plotter

Pro úspěsné nahrání QML souboru do aplikace Data Plotter je potřeba nejprve QML soubor z mikrokontroléru odeslat. Pro nahrávání souboru je využitý začáteční přepínač "$$Q(data)", kde data jsou samotný QML soubor v binárním formátu. Překlad souboru QML do binární podoby umožňuje přímo aplikace Data Plotter po přepnutí do "vývojového režimu, kdy můžeme manuálně, nahrát .qml soubor a testovat ho, aniž bychom jej nahrávali přímo z mikrokontroléru".

Samotné nahrání lze realizovat například při spuštění mikrokontroléru, ale to zapříčiní že nemusí být Data Plotter v režimu "připojen", tedy nemusí být QML soubor úspěšně nahrán. Lepší je tedy zvolit "startovací"příkaz, který lze nastavit v panelu "nastavení"(viz 4.2) Data Plotteru. Tento příkaz je po sériové lince odeslán do mikrokontroléru, vždy když je v aplikaci stisknuto tlačítko "připojit". V mém konkrétním případě jsem pro odeslání QML souboru zvolil sekvenci ".;", po jejímž odeslání se spustí chod firmwaru v mikrokontroléru a nahraje se QML soubor.



Obr. 4.2: Příkaz pro odeslání QML souboru

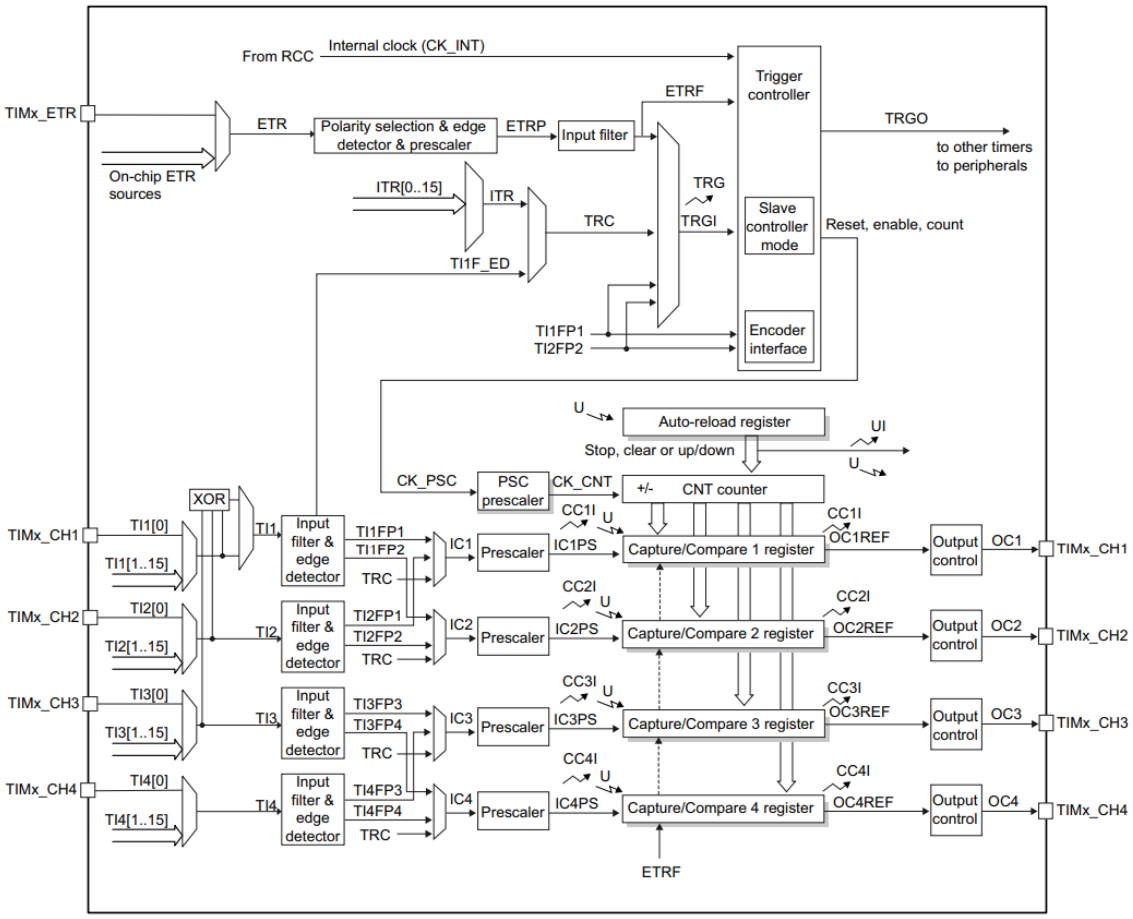
5 Popis mikrokontroléru a využitých periférií

V následující kapitole krátce rozebereme problematiku používaných periférií mikrokontroléru nutných pro správný chod firmwaru. Konkrétně pak popíšeme využité funkcionality a použitá nastavení časovačů, řadiče DMA, analogově digitálního převodníku ADC a sériového komunikačního rozhraní USART.

5.1 Časovače

Mikrokontrolér STM32G031J6 nabízí k dispozici několik různých časovačů, konkrétně pak jeden 32bitový a zbytek 16ti bitové. Nejprve se zaměříme na časovače použité pro generování signálu PWM a časovač použitý pro měření frekvence a délky pulzu signálu. Pro účely vytvoření čítače frekvence a délky pulzu jsem využil časovač TIM2. Jedná se o jediný 32bitový časovač v mikrokontroléru a pro správné měření i malých frekvencí při velkém rozlišení je potřeba jej využít.

Časovač je blok mikrokontroléru, který čítá náběžné hrany řídícího hodinového signálu a tak umožňuje vytvářet časovou referenci, generovat PWM signál, nebo měřit přesné časové okamžiky dějů, jako jsou náběžné/sestupné hrany vnějšího signálu. Blokové schéma vnitřního zapojení časovače můžeme vidět na obr. 5.1. Na vstupní části časovače se nachází nastavitelná předdělička (PSC registr), díky které lze zmenšit frekvenci řídicího signálu. Dále je zde auto-reload registr, který určuje hodnotu, do jaké má čítač počítat, než dojde k jeho vynulovaní. Takto načtený počet řídících pulzů je ukládán v CNT registru. Na něj jsou paralelně zapojeny jednotlivé capture-compare jednotky pro využití časovače např jako generátoru PWM signálu, nebo na čítač frekvence.



Obr. 5.1: Blokové schéma čítače

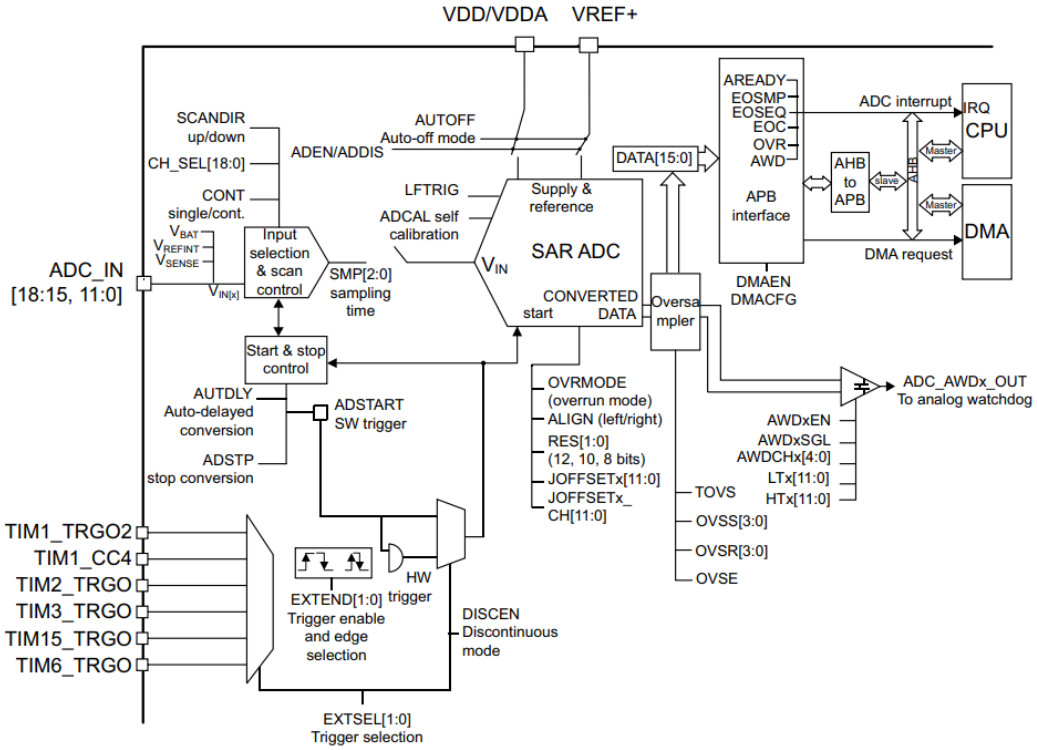
Pro použití časovače na měření frekvence a pro generování PWM signálu jsem využil jeho "capture-compare"jednotek. Tato jednotka umožňuje po správné konfiguraci na režim "capture"zachycení přesného časového okamžiku kdy byla zaznamenána spádová/náběžná hrana vnějšího signálu (v případě použití na měření frekvence).Naopak v režimu "compare", kdy porovnává aktuální vnitřní stav časovače oproti zadané hodnotě, pro změnu výstupní hodnoty (z log.0 na log.1, nebo naopak) v případě potřeby generování PWM signálu.

Časovač je také napojen na interní systém přerušení, neboli NVIC (Nested vector interrupt controller). Ten umožňuje vyvolat přerušení, např při detekci již zmíněné, náběžné či sestupné hrany, nebo naopak při přetečení auto-reload registru. V případě přetečení auto-reload registru lze tedy časovač použít k vytvoření časově pevně daného "callbacku"ve kterém můžeme periodicky vykonávat nějakou rutinu.

5.2 Analogově digitální převodník ADC

Analogově digitální převodník ADC umožňuje čtení analogového napětí na pinech mikrokontroléru. V mikrokontroléru STM32G031J6 je konkrétně použito jeden analogově digitální převodní s postupnou aproximací s rozlišením 12 bitů a možností multiplexování kanálu na jednotlivé piny. Zároveň umožňuje využiti funkce "analog watchdog", díky které lze hardwarově sledovat napětí na zvoleném kanálu. Tato funkcionality je využita

pro funkci "trigger"u osciloskopu. Není tedy potřeba softwarovým způsobem hledat spouštěcí úroveň signálu z již naměřených vzorků, ale periférie je schopna při překročení určité úrovně automaticky vyvolat přerušení a tím dá vědět, že nastala uživatelem zadaná spouštěcí podmínka.



Obr. 5.2: Blokové schéma ADC

Jak je vidět v obr.5.2, tak periférie nabízí i několik interních kanálů k měření např. teploty nebo přesné napěťové reference. Tato přesná napěťová reference díky kalibraci od výrobce disponuje přesným napětím (cca 1.2V), díky jehož změření lze dopočítat velikost napájecího napětí mikrokontroléru. Bez této napěťové reference bychom nebyli schopni napájecí napětí zjistit, neboť maximální změřená hodnota takovýmto převodním vždy bude určena napájecím napětím mikrokontroléru. To však nemusí být přesně 3.3V a tedy skrze tuto referenci spolu se zpětným dopočítáním změny o 1 LSB zjistím velikost napájecího napětí.

5.3 Řadič DMA

Řadič DMA, neboli "direct memmory acess"umožňuje přímý přístup do paměti mikrokontroléru, aniž by bylo potřeba jej řešit pomocí softwaru. Tato periférie dokáže například periodicky vyčítat data z ADC převodníku, nebo zapisovat data do periférie pro sériovou komunikaci USART, nebo i kopírovat velké množství dat z jedné části paměti do druhé. Tímto lze ušetřit velká část výpočetního výkonu, kterou by jinak mikrokontrolér strávil např. vyčítáním hodnot datové registru u ADC, nebo v případě rozhraní USART, odesíláním jednotlivých bajtů do druhého zařízení.

Mikrokontrolér STM32G031J6 disponuje jedním jedním řadičem DMA který nabízí až 5 multiplexovaných kanálů. Jednotlivé kanály lze nastavit na přenos dat třemi způsoby.

- Z periférie do paměti
- Z Paměti do periférie
- Z paměti do paměti

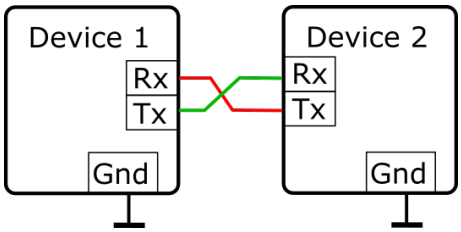
Řadič DMA umožňuje nastavit počet přenesených bloků dat(8/16/32 bitů), díky čemuž lze po překročení tohoto počtu například vyvolat přerušení a konkrétní data nějakým způsobem zpracovat. Mezitím co se kopírují může však jádro dělat jiné úkony a nezatěžovat tak svůj výpočetní výkon obyčejným vyčítáním hodnot z některé periférie. Jedná se tdy o velice účinný nástroj pro záznam/odesílání velkého množství dat najednou.

5.4 Rozhraní USART

Zkratka USART (universal synchronus/asynchronus recivertr-transmitter) značí zařízení, které je dedikované pro vzájemnou sériovou komunikaci dvou zařízení po 2 (popř. 3) vodičích. Tento blok lze najít téměř v každém mikrokontroléru. Není tomu jinak ani u mikrokontrolérů řady STM32G031. Pokud chce uživatel mít možnost k jednoduchému způsobu komunikace s jiným zařízením, je toto nejjednodušší způsob.

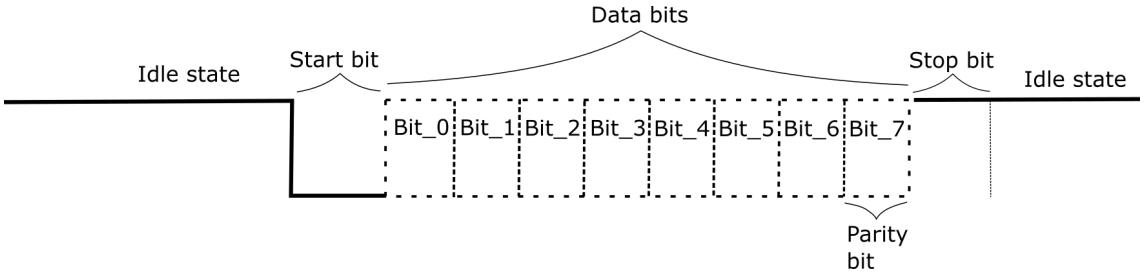
Sériové komunikační rozhraní USART je hojně používaným komunikačním rozhraním k přenosu dat mezi mikrokontrolérem a PC aplikacemi. V mikrokontroléru STM32G031J6 jsou k dispozici dva bloky USART. Skrze něj jsou odesílaná naměřená data do aplikace Data Plotter a naopak přijímána konfigurační data z aplikace pro nastavení parametrů programu.

V asynchronním módu je jeden (Tx - výstup) určen pro vysílání a druhý (Rx - vstup) pro přijímání. Tudíž Tx výstup USARTu jednoho zařízení musí být připojen na Rx vstup druhého zařízení a Tx výstup druhého na Rx vstup prvního (viz obr. 5.3).



Obr. 5.3: USART schéma zapojení - asynchronní mód

Jelikož v tomto režimu nemáme vedený hodinový signál spolu s datovým, je třeba nastavit tzv. „Baud rate“. Baud rate je označení pro rychlost komunikace a je udávaný v bitech/s. Tímto lze v asynchronním režimu docílit správné synchronizace bitů na straně vysílače a přijímače. Celý komunikační protokol má strukturu viz obr. 5.4.



Obr. 5.4: USART protokol - asynchronní mód

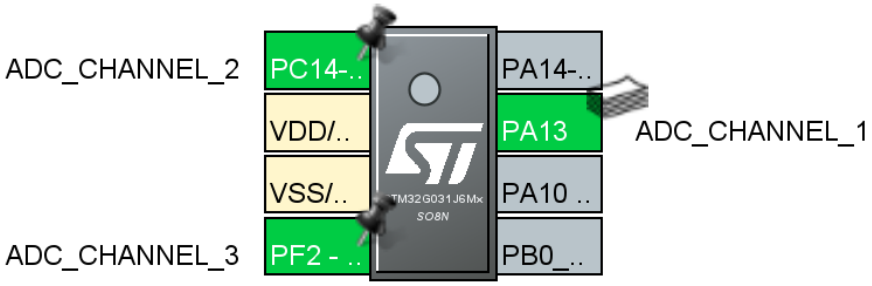
6 Realizace funkce Osciloskop

V této kapitole se budeme zabývat detailním popisem realizace funkce osciloskop. Rozebereme využití veškerých použitých periférií, jejich vzájemné propojení a algoritmu použitého pro sběr vzorků a komunikaci s PC aplikaci Data Plotter.

Mým zadáním je realizovat funkci osciloskop, která bude schopna zaznamenávat vzorky z až 3 různých kanálů, a následně je posílat do aplikace Data Plotter, kde se budou zobrazovat. Zároveň dle požadavku vedoucího práce má mít následující funkcionality.

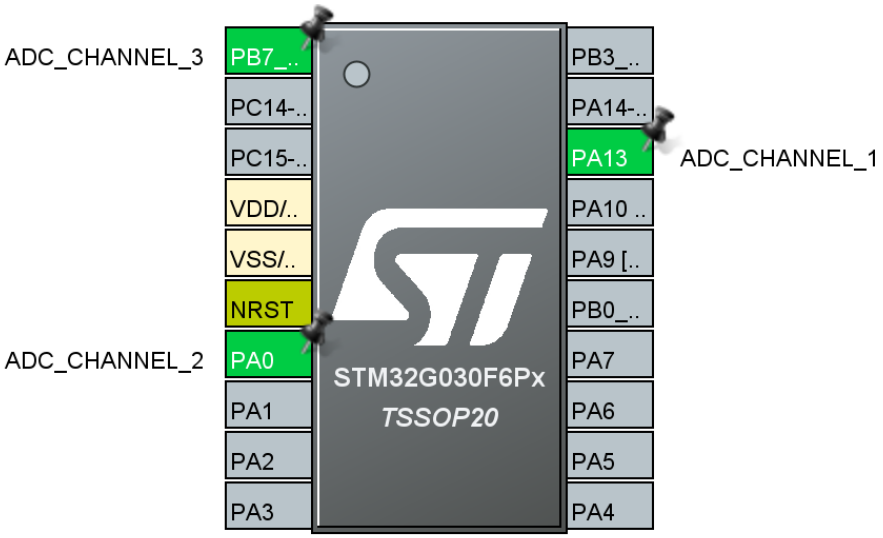
1. Trigger
   (a) Náběžná hrana
   (b) Spádová hrana
   (c) Force trigger
2. Pretrigger
   (a) 15%
   (b) 50%
   (c) 85%
3. Single trigger
4. Uživatelské rozhraní v jazyce QML

Na základě těchto požadavků jsem zvolil rozložení kanálů na mikrokontroléru viz obr. 6.1. Vzhledem k velmi omezeným možnostem spojenými s velmi malým rozměrem pouzdra je potřeba před prvním spuštěním přenastavit "option bajty"a vypnout funkci reset, která se ve výchozím stavu nachází na pinu č.4. Po vypnutí funkce reset je tedy možné pin používat klasickým způsobem, jako každý jiný GPIO pin na pouzdře mikrokontroléru.

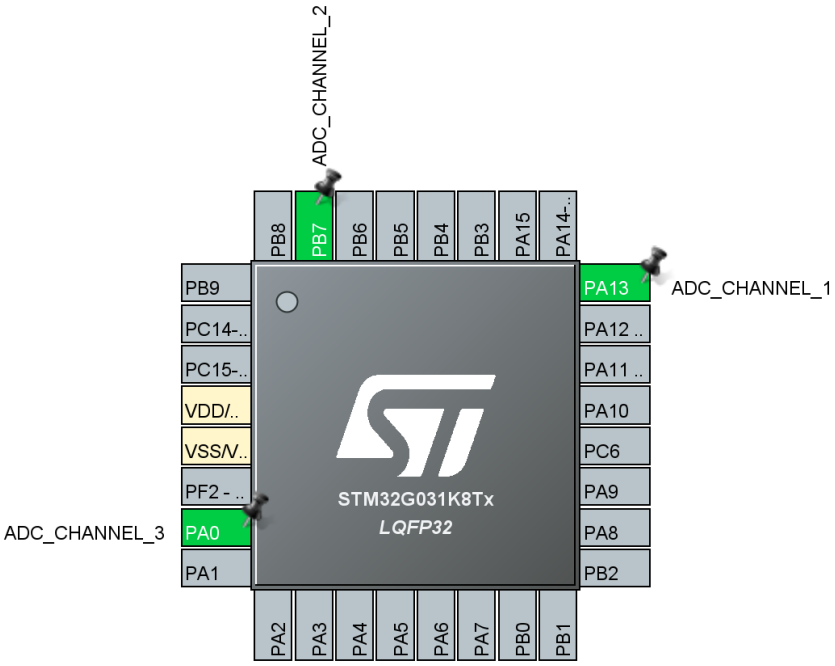


Obr. 6.1: STM32G031J6 - pinout osciloskopu

Díky kompatibilitě i s většími pouzdry lze bez problému software nahrát i do varianty v pouzdře TSSOP20 nebo v pouzdře LQFP32. Obě varianty byly vyzkoušeny. Větší pouzdra mikrokontroléru jsem netestoval, ale dle datasheetu, by měly také fungovat. Na obrázcích č. 6.2 a 6.3 můžeme vidět použité piny na větších pouzdrech.



Obr. 6.2: STM32G031F6 - pinout osciloskopu



Obr. 6.3: STM32G031K6 - pinout osciloskopu

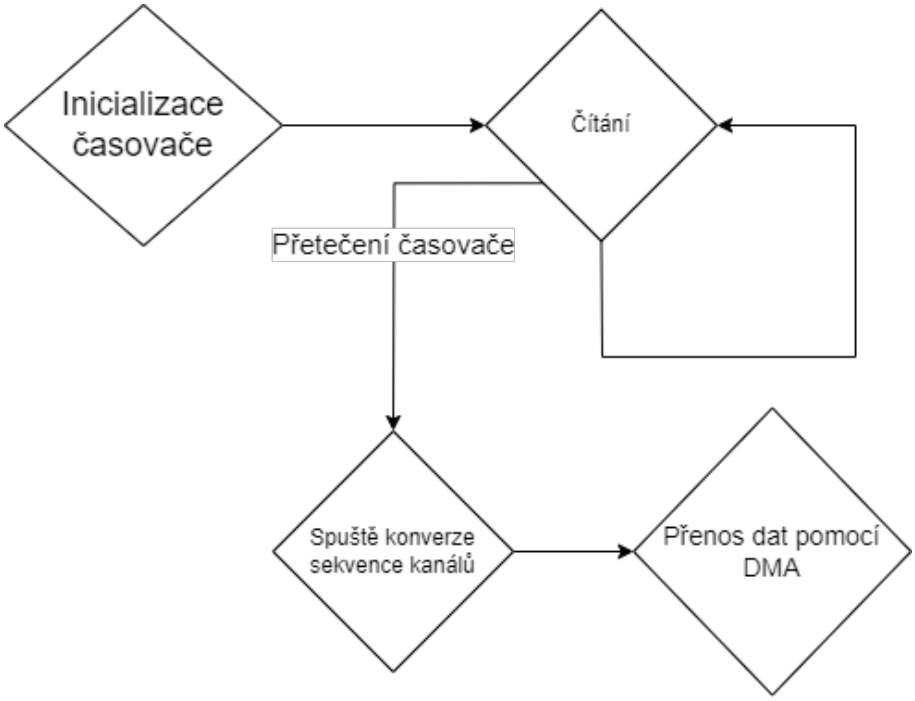
6.1 Paměťová náročnost

Od vedoucího práce jsem měl zadán požadavek, aby byl schopen osciloskop zaznamenat co největší možný počet vzorků. Vzhledem k celkové kapacitě paměti 8kb paměti SRAM v mikrokontroléru jsem tedy zvolil hodnotu 3072 vzorků. Kapacitu by bylo možné rozšířit na dvojnásobek, ale pouze za ústupek v rozlišení převodníku, který umožňuje odebírat vzorky 8/10/12 bitovém rozlišení. Pro účely osciloskopu je potřeba velké rozlišení, tak bylo vybráno rozlišení 12ti bitů. Proto je na každý vzorek potřeba 16ti bitový kus paměti, tedy celková paměťová náročnost na buffer vzorku z ADC je 6kb. Na zbytek aplikace nám tedy zbyly pouze 2kb, což akorát stačí na zpracovávání dat z PC a řídcí struktury pro celý algoritmus.

Paměť FLASH o velikosti 32kb na celý projekt stačila bez problému. Do paměti se vešel celý řídící algoritmus včetně QML souboru, který se po připojení odešle do PC aplikace.

6.2 Princip odběru vzorků

Osciloskop vyžaduje odběr vzorků synchronně se zvolenou časovou základnou v definovaných intervalech. Pro přesné časování odběru vzorků je použit časovač TIM1. Ten je nastaven v režimu, kdy při jeho přetečení (blíže k funkci časovače v sekci 5.1) umožňuje být použit jako spouštěcí signál konverze pro ADC převodník. Ten je tedy nastaven tak, aby periodicky spouštěl konverzi sekvence měřených kanálů na základě přetečení ARR(auto reload register) registru časovače.



Obr. 6.4: Grafické zobrazení odběru vzorků

Celkový odběr vzorků a jejich vyčítání je lze realizovat dvěma způsoby. Buďto softwarovým způsobem, kdy můžeme vždy po dokončení jedné konverze vzorek vyčíst, nebo pomocí řadiče DMA, kdy se odběr vzorků zautomatizuje a není tak vytěžováno jádro mikrokontroléru zbytečnými operacemi kopírování dat do bufferu. Softwarový odběr vzorků může, i za použití přerušení, způsobit chybějící vzorky v záznamu a tím pádem "neúplný"signál. Proto jsem v mém řešení zvolil odběr pomocí DMA, který zajistí celistvost měřených dat a celkově usnadní celý algoritmus Celý algoritmus je graficky naznače v obr. 6.4.

6.2.1 Nastavení vzorkovací frekvence

Vzorkovací frekvence udává počet odebraných sekvencí kanálů za 1s. Analogově digitální převodník v mikrokontrolérech řady STM32G031 umožňuje odběr maximálně 2.5MS/s, tedy v případě tří kanálů a vzorkovací frekvence 700kHz jsme lehce pod hranici možností ADC převodníku, konkrétně na 2.1MS/s. Vzhledem ke krátkému zpoždění při překonfigurování analog-watchdogu je potřeba nevolit nejvyšší počet vzorků za vteřinu. Při odběru méně kanálů (např. 1 kanál) by šla vzorkovací frekvence zvýšit až na úroveň 2-2.5MHz neboť se při jednom cyklu časovače spouští pouze jediná konverze kanálu. Tato funkcionalita však není ve výsledném firmwaru implementována aby měl uživatel zaručenou stejnou vzorkovací frekvenci při jakémkoliv nastavení.

Uživatel osciloskopu chce být schopen nastavovat vzorkovací frekvenci v jednotkách Hertz, avšak takovou hodnotu nelze napřímo zapsat do registru časovač. Proto je potřeba z uživatelem zadané vzorkovací frekvence nastavit předděličku a auto-reload registr časovače. Vzhledem k široké škále vzorkovací frekvence (1Hz-700kHz) a skutečnosti, že časovač TIM1 je pouze 16ti bitový, tak je potřeba předděličku měnit dynamicky podle zadané frekvence. Po nalezení vhodné předděličky už následně pouze podle vzorce 6.1 nastavíme vhodnou velikost auto-reload registru.

$$ARR_{value} = \frac{(\frac{F_{clk}}{PSC_{value}})}{F_{sampling}} \tag{6.1}$$

Jak lze vidět v ukázce 4, tak jednoduše zkoušíme postupně inkrementovat předděličku časovače, dokud nám nevyjde výsledek výpočtu hodnoty pro auto-reload registr menší než 65535, což je největší možný.

```
uint32_t _tmp = 0;
for(uint32_t i = 0; i<UINT16_MAX; i++)
{
    _tmp = CORE_FREQUENCY/(i+1))/_freq;
    if(_tmp < UINT16_MAX)
    {
        TIM1->PSC = (uint16_t)i;
        break;
    }
}
TIM1->ARR = _tmp;
```

Zdrojový kód 4: Algoritmus pro dynamické nastavení předděličky časovače

Tímto nastavováním frekvence však dochází k odchylce od žádané frekvence. Ta bohužel bez zvýšení frekvence hodinového signálu mikrokontroléru nelze snížit, tak jsem do uživatelského rozhraní implementoval textové pole s "reálnou"nastavenou frekvencí v mikrokontroléru. Ta je potřeba pro použití při stroboskopickém vzorkování.

!!!!!!! DOPlnit referenci!!!!!!.

6.3 Trigger

Hlavním účel triggeru (spouštěcího mechanizmu) spočívá v detekci okamžiku, kdy na měřeném signálu nastala náběžná, či spádová hrana. Toho je docíleno zjištěním hodnoty napětí, která klesla /vzrostla nad uživatelem nastavenou hodnotu. Konkrétně tak hledáme rostoucí/klesající trend v průběhu odebírání vzorků. Trigger osciloskopu může fungovat v několika režimech.

- Normal trigger
- Single trigger
- Auto trigger
- Force triger

6.3.1 Druhy triggeru

V režimu **normal trigger** dochází k hledání spouštěcí úrovně kontinuálně. Tedy vždy když je nalezen správný okamžik triggeru, tak je odebrána celá následující sada vzorků a ta je zobrazena, přičemž po zobrazení dojde ihned k opakovanému spuštění. Tím pádem se uživateli měřený signál stále periodicky obnovuje, kdykoliv kdy je nalezna spouštěcí podmínka.

Naopak v režimu **single trigger** dojde pouze k jednomu odebrání sekvence kanálů, a následně se odběr znovu nespustí. Toto je vhodné zejména pro děje, kde užiatel nechce aby se mu odebrané vzorky během okamžiku přepsaly a aby mohl sledovaný signál analyzovat a prohlédnout si jej.

Následně nám zbývají režimy **auto trigger** a **force trigger**. Režim force trigger zcela ignoruje čekání na spouštěcí mechanizmus. Je pouze odebrána sekvence vzorků, která je odeslána do aplikace k zobrazení. tento režim může být vhodný, pokud uživatel neví jak měřený signál vypadá a chce se nejprve podívat, jak by bylo vhodné nastavit např úroveň triggeru nebo vzorkovací frekvenci. Režim auto trigger pak funguje způsobem, že určitou chvíli čeká na hledání triggerovací úrovně, ale pokud není během dané chvíle nalazena spustí odběr vzorků stejně jako force trigger. V mé implementaci funkce osciloskop jsem implementoval pouze funkcionality normal,single a force trigger.

Trigger je možné na mikrokontroléru STM32G031J6 realizovat dvěma způsoby. Buďto softwarově, nebo hardwarově pomocí "analog-watchdog"funkcionality ADC převodníku. Já jsem zvolil řešení triggeru za pomoci právě analog-watchdog periférie.

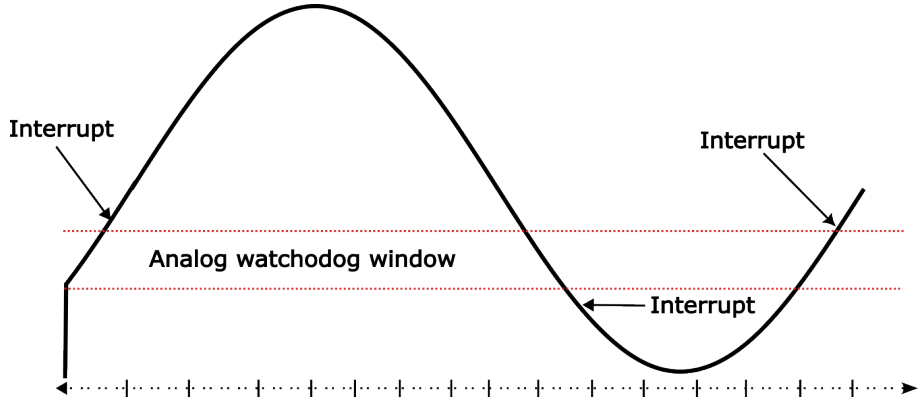
6.3.2 Softwarové řešení triggeru

Softwarové řešení triggeru spočívá v kontrolování každého odebraného vzorku a hledání specifického "spouštěcího"příznaku, tedy úrovně . kterou signál překročí buďto při náběžné, či sestupné hraně. Tento přístup však není zcela vhodný, neboť nám při něm může dojít k vynechání některého vzorku (zejména pak při vyšších vzorkovacích frekvencích) a zároveň je velmi výpočetně náročný. Tento problém lze ošetřit například kontrolováním každého X-tého vzorku ze sekvence, ale tím přijdeme o žádané přesné rozlišení úrovně spouštění a především můžeme přijít o rychlé dynamické děje, které tímto přístupem "přeskočíme".

6.3.3 Hardwarové řešení triggeru

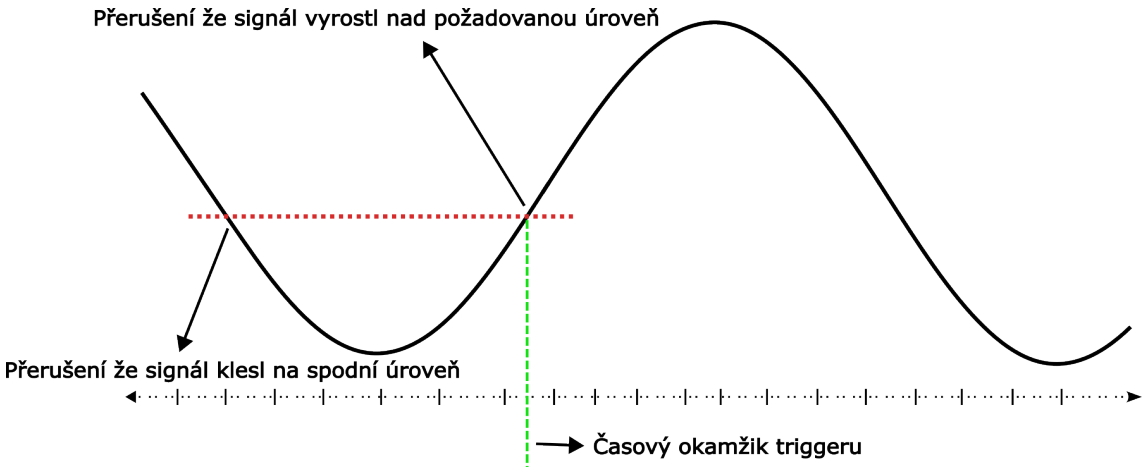
Dalším, v tomto případě lepším, řešením je využít Analog-watchdogu v ADC převodníku. Ten je schopen monitorovat změřené napětí na zvoleném kanálu a při překročení dané úrovně vyvolat přerušení (více v sekci 5.2). Avšak i tento způsob hledání spouštěcí úrovně je potřeba softwarově ošetřit. Převodník ADC, konkrétně jeho část analog-watchdog umožňuje nastavit rozsah monitorovaných hodnot. Jedná se tedy o tzv. window analog-watchdog.

Analog-watchdog je součástí periférie ADC, tedy nefunguje na ní nezávisle. Pouze kontroluje naměřená data. Tím je dáno časové rozlišení okamžiku triggeru, tedy analog-watchdog nemůže fungovat rychleji, než je nastavená vzorkovací frekvence osciloskopu.



Obr. 6.5: Analog-watchdog

Jak je vidět na obr. 6.5, tak pro správnou realizaci triggeru, tedy rozlišení náběžné/-sestupné hrany tak je potřeba použít analog watchdog dvakrát. Například když hledáme náběžnou hranu signálu při úrovni 50% (v našem případě cca 1.65V), tak je potřeba nejprve detekovat kdy je signál mezi úrovní 0-1.65V a následně analog watchdog překonfigurovat na monitorování druhé poloviny signálu, tedy 1.65V-3.3V. Celý proces zjištění přesného okamžiku triggeru na náběžnou hranu je naznačen na obr. 6.6.



Obr. 6.6: Nalezení triggeru náběžné hrany

Po správném nalezení triggeru už pouze stačí odebrat daný počet vzorků a máme kompletní signál, který má na svém počátku požadovanou spouštěcí podmínku. Pro účely funkce osciloskopu může uživatele zajímat i průběh signálu před spouštěcí podmínkou, tedy je potřeba implementovat funkci tzv. pre-triggeru.

6.4 Pre-trigger

Funkce pre-triggeru spočívá v zajištění zaznamenání určitého počtu vzorků před nalezením spouštěcí podmínky (triggeru). Proto je potřeba při naplňování nového datového bufferu spustit samotný odběr vzorků dříve, než se spustí mechanizmus na detekci náběžné/sestupné hrany.

V mém případě jsem k řešení problému přistoupil s použitím kruhového bufferu pro odběr vzorků. Odběr vzorků pro daný kanál běží tedy v periodickém režimu, kdy analog-watchdog spustím až po úspěšném načtení N vzorků, které zajistí, že část signálu je již odebrána a až následně se spustí detekce spouštěcí podmínky.

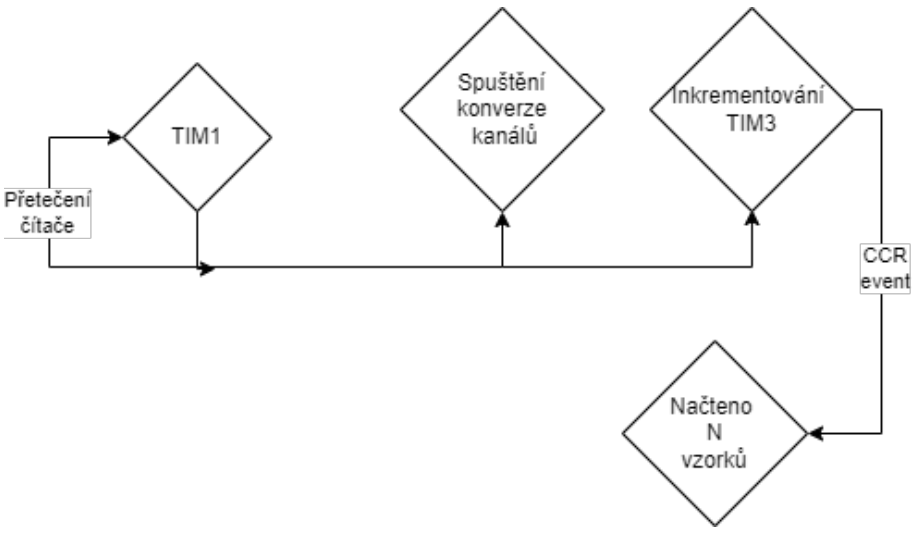
Čítání odebraných vzorků lze řešit dvěma způsoby, buďto softwarovým způsobem, kdy se periodicky dotazujeme kolikátý vzorek byl odebrán, nebo hardwarově, za použití druhého časovače. V mé implementaci funkce osciloskopu jsem využil hardwarovou variantu, neboť tímto způsobem můžeme snížit nároky na výpočetní výkon a mikrokontrolér tak může v průběhu odebírání vzorků vykonávat jinou činnost, například obsluhovat komunikaci s PC aplikací.

6.4.1 Softwarový čítač odebraných vzorků

V této variantě řešení je potřeba se periodicky (korespondujíc s vzorkovací frekvencí) dotazovat na počet odebraných vzorků. Lze se dotazovat například při přerušení z časovače, který spouští odběr vzorků, nebo v přerušení při dokončení konverze vyvolaného z ADC převodníku. Avšak tímto způsobem, za předpokladu že je ADC převodník spouštěn hardwarovým způsobem, můžeme v průběhu dotazu přijít o několik vzorků signálu a tím způsobit jeho necelistvost při výsledném zobrazení.

6.4.2 Hardwarový čítač odebraných vzorků

Ve své práci jsem využil variantu čítání vzorků hardwarovým způsobem za pomoci druhého časovače. Princip činnosti spočívá v inkrementování CNT registru "kontrolního"čítače na základě přetečení čítače, který spouští konverzi kanálů. Takovéto zapojení lze realizovat s časovači, které umožňují tzv. master/slave zapojení. Tedy kdy jeden časovač řídí ten druhý. V našem případě update event časovače TIM1 řídí časovač TIM3. V časovači TIM3 jsem využil jedné z jeho capture-compare jednotek, kterou vždy, když potřebuji odebrat přesný počet vzorků nakonfiguruji tak, aby po N odebraných sekvencí kanálů vyvolala přerušení. Tím jsem schopen včas a efektivně zastavit odběr dalších vzorků. Principiální zapojení obou čítačů lze vidět na obr. 6.7.



Obr. 6.7: Zapojení časovačů v režimu master/slave

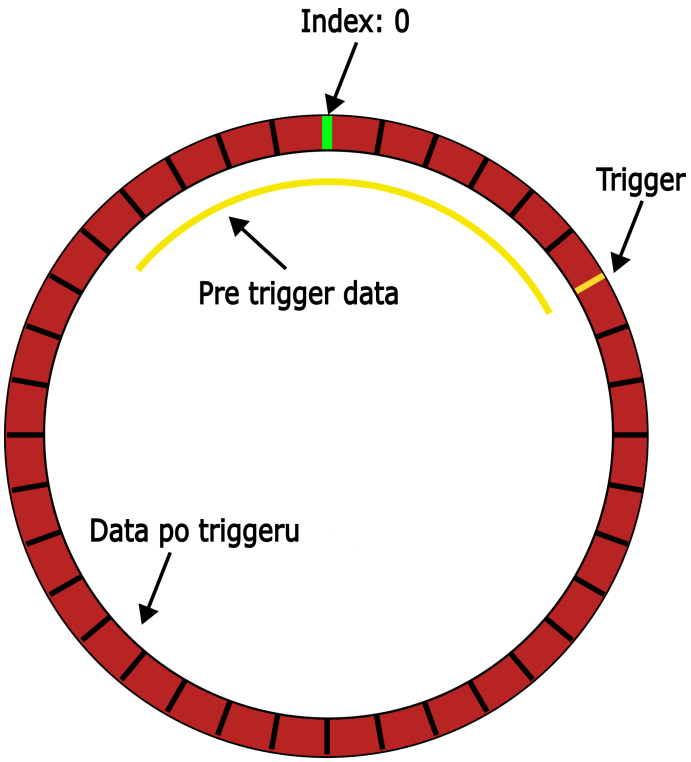
6.5 Sekvence odběru kanálu

V této sekci si detailně popíšeme princip odběru vzorků v jednotlivých režimech, které jsem implementoval, tedy normal,single a force trigger.

6.5.1 Režim single trigger

Princip funkce spočívá ve využití kruhového bufferu, kdy se vzorky periodicky odebírají, dokud není nalezena spouštěcí úroveň. Následně je již odebráno pouze N vzorků (dle volby pretriggeru) a signál je kompletní.

Nejprve je po zadání startovací sekvence uživatelem (skrze aplikaci v PC) spuštěn časovač , který spouští periodický oděr vzorků z ADC převodníku a časovač, který čítá počet odebraných vzorků. Následně je odebrán počet vzorků, podle úrovně pretriggeru. Tedy např v režimu 15% pretriggeru se jedná o 460 vzorků z celkových 3072. Tím je zajištěno, že minimální počet vzorků před signálem je načteno. Následně dojde ke spuštění periférie analog-watchdog, která monitoruje vzorky z daného kanálu. Mezitím se stále odebírají další vzorky a přidávají do kruhového bufferu. Pokud periférie analog-watchodg najde správnou spouštěcí úroveň, je z časovač pro čítání vzorků zjištěno o jaký šlo vzorek a jeho index je uložen. Následně je nakonfigurován časovač pro odběr vzorků tak, aby po zbylých 85% vzorcích (2612) vyvolal přerušení. Po vyvolání tohoto přerušení nový odběr vzorků zastavíme a v kruhovém bufferu máme náš signál uložený, včetně 460 vzorků, které nastaly těsně před nalezením spouštěcí podmínky.



Obr. 6.8: Kruhový buffer vzorků

Takto načtené vzorky v kruhovém bufferu (viz obr. 6.8) však nejsou správě uspořádány, takže je nemůžeme jednoduše odeslat od nultého vzorku po poslední. Proto je nutné jejich přeskládání. Vzhledem k tomu, že v průběhu odběru vzorků jsme si uložili index vzorku,

kdy došlo k nalezení spouštěcí podmínky a faktu, že známe délku záznamu před a po spouštěcí podmínce, tak můžeme signál znovu poskládat do správné podoby a odeslat do PC aplikace.

```
void SendNormalTriggerData(void)
{
    int _num = (int)PRETIGGER_SAMPLES_CNT-(int)pretrigger_endIndex;

    huart2._status = USART_BUSY;
    //overflow of circular buffer
    if(_num >0)
    {
        SendSyncPacket();
        SendDataToPC((_pretriggerStartIndex*2),(ADC_BUFFER_LEN*2));
        SendDataToPC(0,(pretrigger_endIndex*2));
        SendDataToPC((pretrigger_endIndex*2),((_pretriggerStartIndex)*2));
    }
    else
    {
        SendSyncPacket();
        SendDataToPC((_pretriggerStartIndex*2),(pretrigger_endIndex*2));
        SendDataToPC((pretrigger_endIndex*2),((ADC_BUFFER_LEN)*2));
        SendDataToPC(0,((_pretriggerStartIndex)*2));
    }
}
```

Zdrojový kód 5: Algoritmus pro odeslání dat do PC ve správném pořadí

Po úspěšném odeslání kanálu do PC aplikace není v tomto režimu spuštěna nová konverze až do doby, kdy uživatel spustí novou konverzi skrze PC aplikaci.

6.5.2 Režim normal trigger

Tento režim je identický s režimem single, pouze dochází k opětovnému spuštění celé sekvence znovu. Tedy opět dojde k načtení pretrigger dat, následnému vyčkání na spouštěcí podmínku a pak odeslání dat ve správném pořadí.

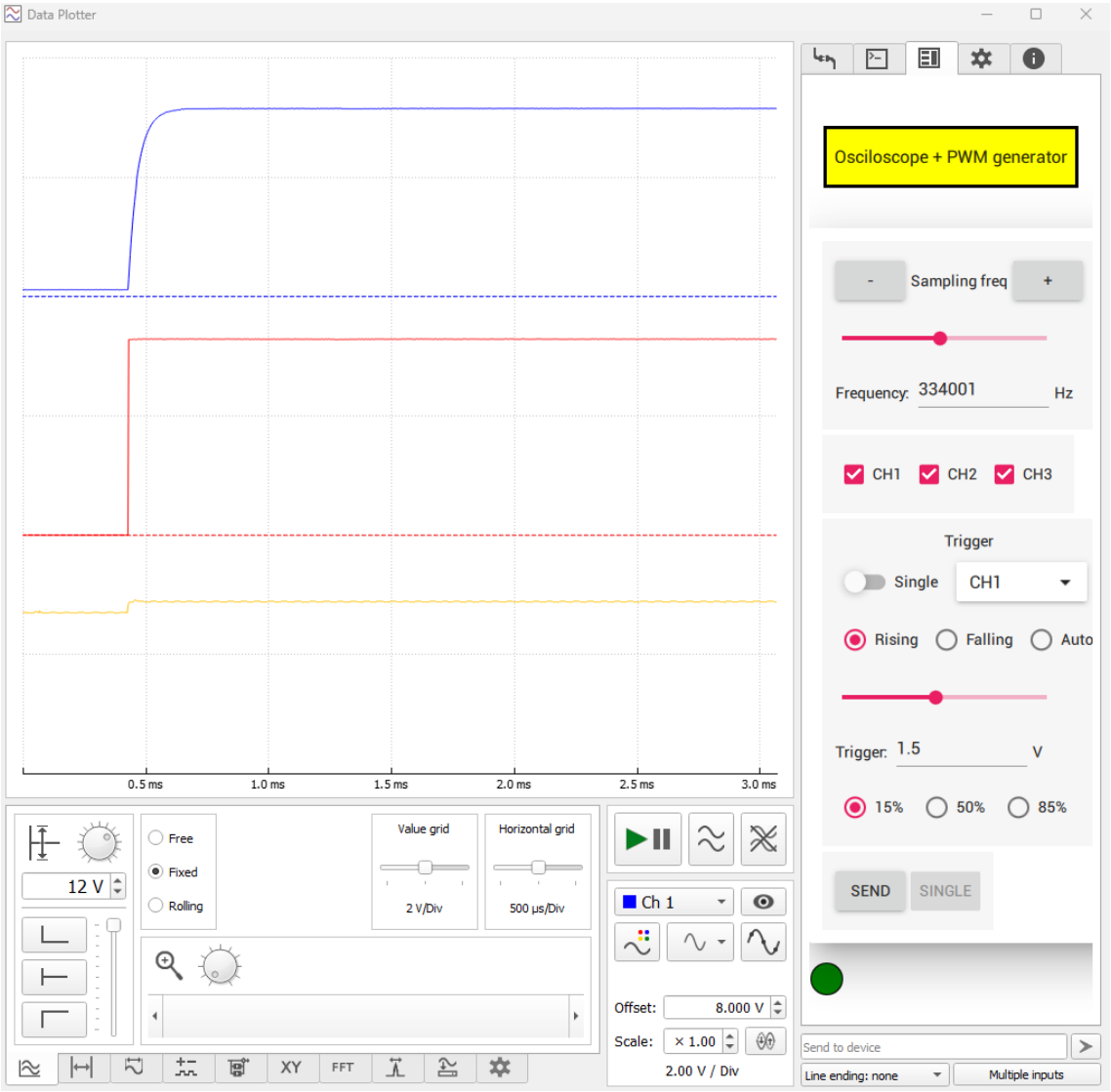
6.5.3 Režim force trigger

V tomto režimu je celý proces velice jednoduchý.Není spuštěn žádný spouštěcí mechanizmus(trigger) a je pouze je spuštěný odběr vzorků a kontrolní čítač nastaven tak, aby vyvolal přerušení po odběru 3072 vzorků. Tato data jsou následně v nepřeskládané podobě odeslána do PC aplikace, kde jsou zobrazena. Tento druh čtení signálu je možný využívat ,obdobně jako normal trigger, v režimu kdy se nová konverze spouští automaticky (normal trigger) nebo v režimu single, kdy je odebrána pouze jedna sekvence a teprve uživatel volí spuštění nové skrze PC aplikaci.

6.5.4 Shrnutí procesu odběru vzorků

Na základě uživatelského nastavení v aplikaci je zvolen žádaný druh odběru vzorků. Odběr vzorků je možný až ze 3 multiplexovaných kanálů s možností volby, na kterém kanálu se bude hledat spouštěcí podmínka. Veškeré algoritmy počítají s paměťovým rozložením kanálů, kdy jednotlivé vzorky se do kruhového bufferu ukládají od kanálu s nejvyšším číslem po ten s nejnižším. Tedy v případě použití 3 kanálů se délka záznamu pro jeden kanál

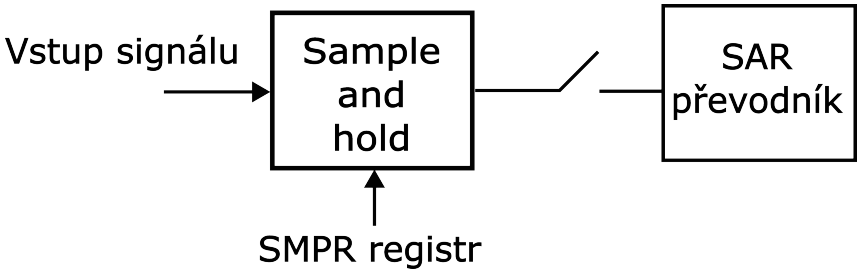
zkrátí z celkových 3072 vzorků na 1024. Na obr. 6.9 můžeme vidět náhled do uživatelského rozhraní nastavení osciloskopu přímo z aplikace Data Plotter s využitím jazyka QML.



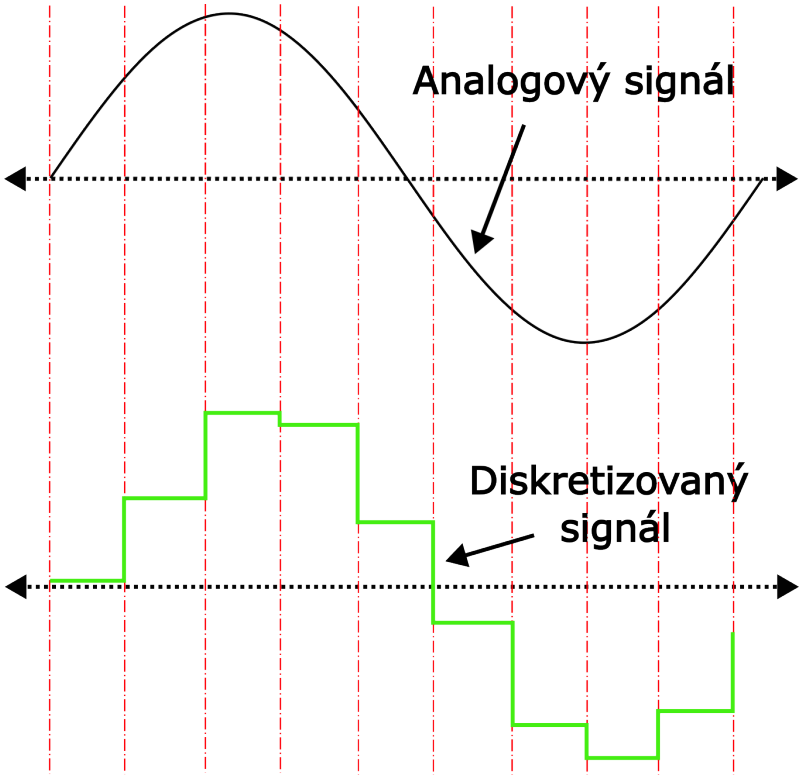
Obr. 6.9: Uživatelské rozhraní nastavení triggeru

6.5.5 Sampling time

Analogově digitální převodník ADC má na svém vstupu integrovaný "sample & hold"obvod, který zajistí stabilitu měřeného napětí v průběhu samotné konverze SAR převodníkem. Tedy jinými slovy diskretizuje vstupní spojitý signál do diskrétních hodnot. Blokové schéma zapojení můžeme vidět na obr. 6.10 a signál před a po diskretizaci na obr. 6.11.



Obr. 6.10: Blokové schéma vstupu ADC převodníku



Obr. 6.11: Spojitý a diskretizovaný signál

Vnitřní zapojení "sample & hold"obvodu je v podstatě kondenzátor, který se nějakou dobu nabíjí a následně je odpojen od vstupu. Tímto vznikne "paměťová"buňka pro již diskrétní signál. Doba, po jakou je vzorkovací kondenzátor otevřen můžeme řídit hodnotou zapsanou v SMPR registru ADC převodníku na hodnoty naznačené v tabulce 1.

Počet hodinových cyklů samplování	Čas samplování při 64MHz
1.5	28ns
3.5	67ns
7.5	143ns
12.5	240ns
19.5	374ns
39.5	758ns
79.5	1525ns
160.5	3081ns

Tabulka 1: Tabluka doby vzorkování

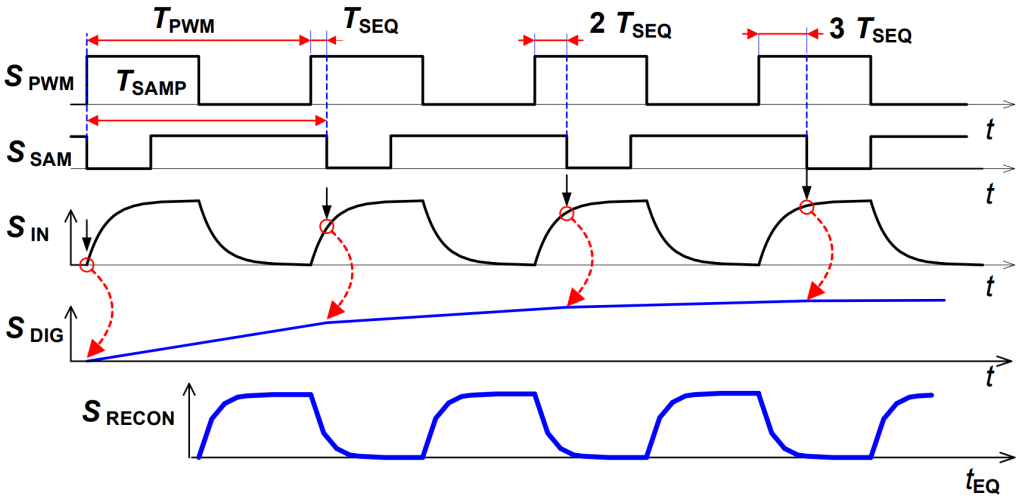
Se snižující se hodnotou SMPR registru roste čas, po který je vzorkovací kondenzátor připojen k vstupnímu signálu. Za předpokladu že je měřený signál připojený přes sériový odpor, tak je potřeba mít dlouhý vzorkovací čas, aby došlo k úspěšnému nabití vzorkovacího kondenzátoru na úroveň odpovídající vstupnímu signálu, Tedy čím větší je doba vzorkování, tím větší může být vnitřní odpor zdroje měřeného signálu. Naopak při velmi malých časech vzorkování se kondenzátor zcela nenabije a výsledek měření je touto chybou ovlivněn. Tato hodnot však nemůže být konstantní napříč celým rozpětím vzorkovacích frekvencí, neboť při vyšších vzorkovacích frekvencích by byla doba "samplování"moc dlouhá. Proto jsem v algoritmu implementoval dynamické nastavení vhodné délky "samplování". tedy při nízkých vzorkovacích frekvencích je doba samplování nejvyšší, naopak při vysokých frekvencích je nízká.

6.5.6 Signalizace nalezení spouštěcí podmínky

Ve spodní části uživatelského rozhraní na obr. 6.9 lze vidět malé zelené kolečko. Jeho funkce spočívá v signalizaci příchodu nových dat z důvodu, kdy periodický signál s triggerem ve stejném čase na obrazovce vypadá identicky s předešlým snímkem. Tedy aby uživatel opticky viděl, zda osciloskop odesílá nové záznamy, nebo zda nic neposílá.

6.5.7 Stroboskopické vzorkování

Princip stroboskopického vzorkování, neboli vzorkování v ekvivalentním čase, je jev, kdy se vzorkovací frekvence měřeného signálu a osciloskopu od sebe liší pouze o velmi malou hodnotu. Tedy každý vzorek je odebrán z jiné periody měřeného signálu, pouze se vzorek vždy posouvá o časový krok. Tímto lze docílit mnohem vyšší "ekvivalentní"vzorkovací frekvence. Podmínkou však je, že měřený signál musí být periodicky se opakující.



Obr. 6.12: Princip vzorkování v ekvivalentním čase [3]

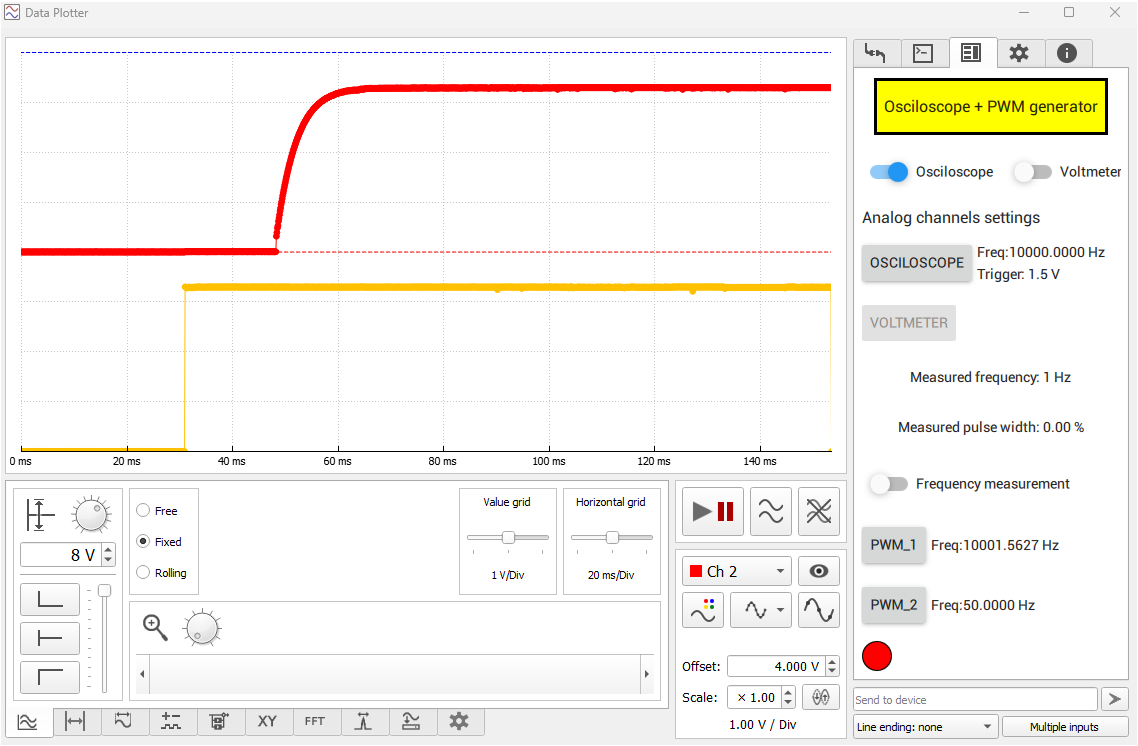
Na obr.6.12 můžeme vidět graficky znázorněný princip vzorkování v ekvivalentním čase. Osa S_{IN} znázorňuje měřený signál a S_{SAM} znázorňuje periodu vzorkování, kdy při každé spádové hraně je odebrán vzorek. Výsledný zrekonstruovaný signál je vidět na průběhu S_{DIG}, tedy ekvivalentní vzorkovací frekvence je násobně vyšší.

Vzorec pro výpočet ekvivalentní vzorkovací frekvence můžeme vidět v rovnici 6.2 [3], kde Δf se vypočítá dle rovnice 6.3 [3].

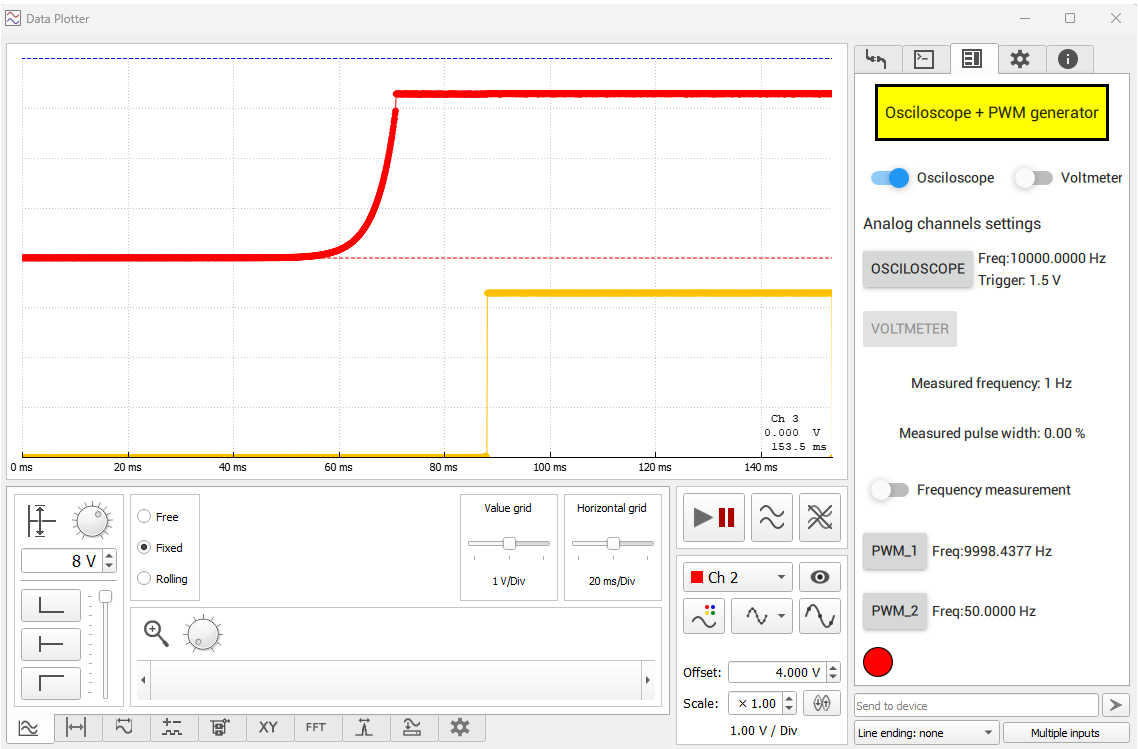
$$f_{SEQ} = \frac{f_{PWM}}{\Delta f} \cdot f_{SAMP} \tag{6.2}$$

$$\Delta f = f_{PWM} - f_{SAMP} \tag{6.3}$$

Z rovnic vyplývá, že pokud je Δf kladná, tak je i výsledná ekvivalentní vzorkovací frekvence kladná. To zapříčiní, že výsledný signál je posunut doprava, tedy časová osa souhlasí s původním signálem. Naopak za předpokladu, kdy Δf by je záporná, tedy frekvence měřeného signálu je menší, než vzorkovací frekvence, tak nám po rekonstrukci vznikne signál, který je zrcadlený.



Obr. 6.13: Využití stroboskopického vzorkování pro měření kapacity vzorkovacího kondenzátoru - kladná Δf



Obr. 6.14: Využití stroboskopického vzorkování pro měření kapacity vzorkovacího kondenzátoru - záporná Δf

Obě varianty ekvivaletního vzorkování můžeme vidět na obr. 6.13 a obr. 6.14, kdy v

prvním případě je rozdíl frekvencí kladný a v druhém záporný.

Vzorkování v ekvivalentním čase je primárně určeno pro jeden ADC převodník, z toho můžeme pozorovat aditivní zpoždění způsobené multiplexováním jednotlivých kanálů.

Zároveň můžeme na záznamu pozorovat časové posunutí jednotlivých kanálů. To je způsobeno multiplexováním jednotlivých kanálu na úrovni ADC převodníku.

7 Realizace funkce voltmetr

V této kapitole se budeme zabývat realizaci funkce voltmetr. Ta vznikla za účely možnosti dlouhého záznamu pomalého signálu a zároveň jakožto náhrada klasického multimetru. Voltmetr disponuje velmi pomalou vzorkovací frekvencí (100Hz), v jejímž rytmu se odebírají vzorky z jednotlivých kanálů . Ty jsou následně průměrovány a výsledek je odeslán do pc aplikace. Konkrétně je daný vzorek přidán k danému kanálu. Tedy na rozdíl od funkce osciloskop se v tomto režimu grafy kanálu nepřepisují, ale naopak se stále zvětšují.

7.1 Požadavky vedoucího práce

- Vytvořit voltmetr se třemi kanály
- Zpracovat na úrovni firmwaru průměrování
- Být schopen zaznamenat velmi dlouhý záznam
- Zobrazovat v uživatelském rozhraní hodnoty rozdílu potenciálů na jendotlivých kanálech

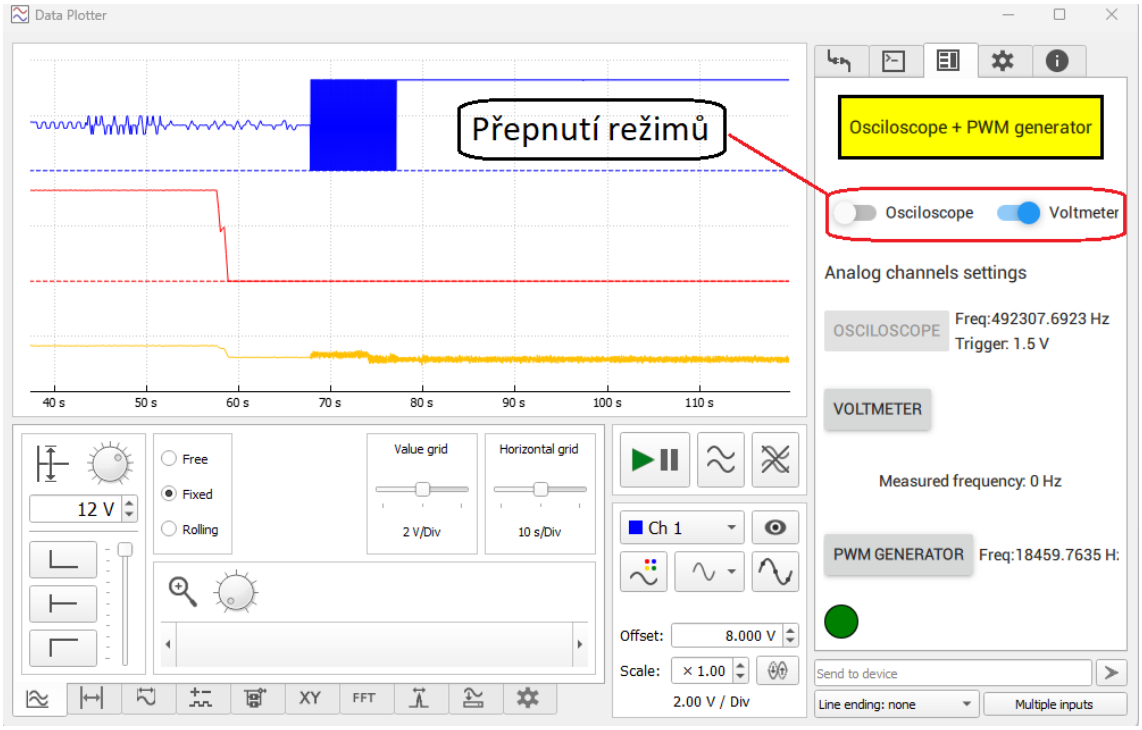
Princip funkce voltmetru funguje obdobně jako běžné multimetry, ale navíc s možností zobrazování záznamu. Tedy firmware měří jednotlivé kanály s velmi malou vzorkovací frekvencí (100Hz), ty následně zpracuje a odešle do aplikace Data Plotter. Zároveň s tím, při odeslání daného bodu do grafu, odešle i hodnoty rozdílu potenciálů na jednotlivých kanálech, tedy hodnoty:

- V1-V2
- V2-V3
- V1-V3

Pinové rozložení jednotlivých kanálu je identické s kanály osciloskopu, tedy jedná se o piny viz obr.6.1, pro pouzdro SO8N, obr.6.2 pro pouzdro TSSOP20 a obr. 6.3 pro pouzdro LQFP32.

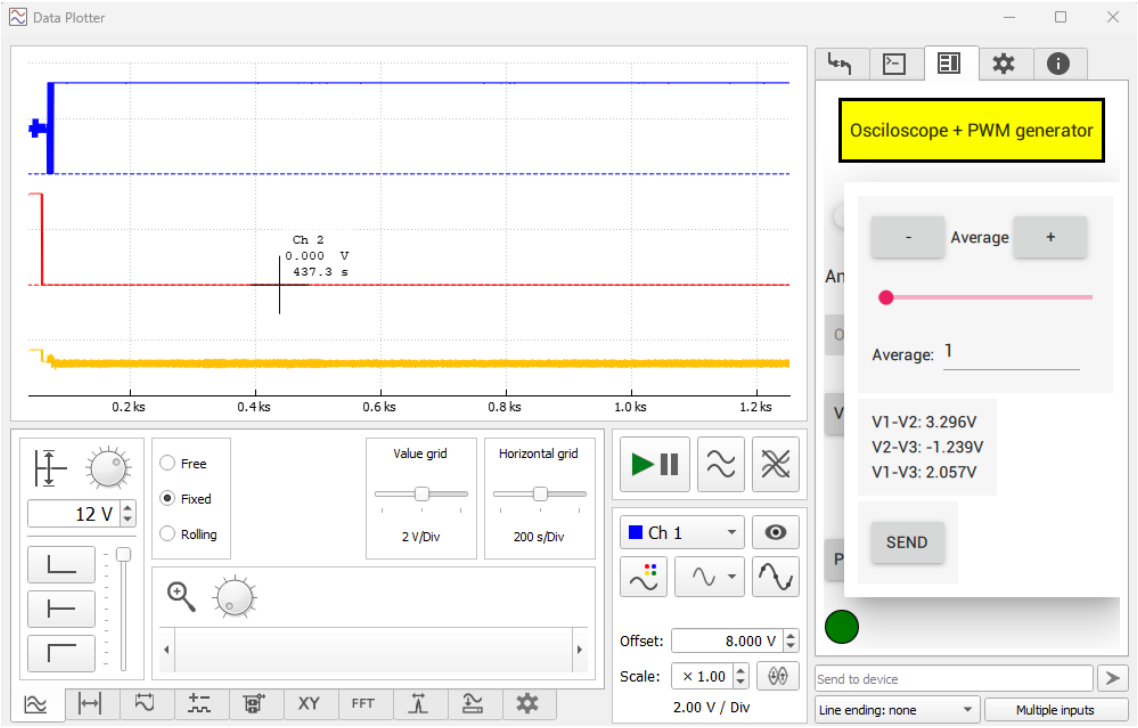
7.2 Způsob použití voltmetru

Nejprve je potřeba v přepnutí mezi voltmetrem a osciloskopem (viz obr. 7.1), v hlavním menu uživatelského rozhraní, přepnout přepínač "Voltmetr"do pozice zapnuto. Tím se automaticky ukončí funkce osciloskopu a přepne se do režimu voltmetru. Obě funkce (myšleno osciloskop a voltmetr) nejde nijak dohromady kombinovat, tedy buďto běží jedna nebo druhá.



Obr. 7.1: Přepnutí do režimu voltmetru

Následně se začnou do jednotlivých grafů zaznamenávat průměrovaná data. Ta jsou přidávána stále dál, tedy celkový graf tvoří dojem, že se zvětšuje. Tím je dosaženo žádané dlouhé délky záznamu a možnosti zpětného analyzování měřeného napětí. na rozdíl od osciloskopu, i dlouhou dobu zpátky.



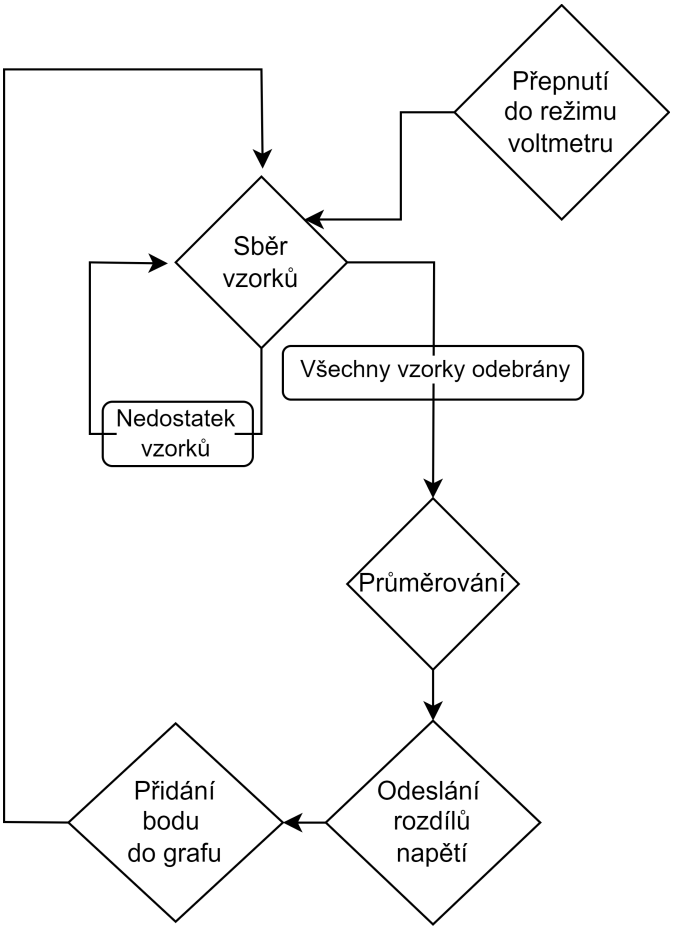
Obr. 7.2: Nastavení voltmetru a zobrazení rozdílů napětí

Po kliknutí na tlačítko "voltmeter", které se zpřístupní po přepnutí funkce, může uživatel do okna (viz obr. 7.2) zadávat požadovanou hodnotu, z kolika vzorků se se určuje průměr. Číslo zobrazené v uživatelském rozhraní určuje počet vzorků pro jeden kanál, tedy číslo 100 odebere 100 vzorků v každém kanálu, tedy celkem 300 vzorků. Zároveň jsou v tomto okně vidět rozdíly napětí mezi jednotlivými kanály. Tato funkcionality je velmi užitečná, například k měření úbytku na odporech, diodách, či jiných součástkách.

7.3 Princip odesílání dat

V tomto režimu, na rozdíl od osciloskopu, je vždy použito přesně N vzorků, tedy nečekáme na žádný okamžik triggeru ap.. V podstatě se jedná o obdobu force triggeru (tedy prostého záznamu kanálů bez spouštěcí podmínky). Proces je takový že je nejprve nakonfigurováno DMA v režimu kdy převede zvolených N vzorků. Zároveň je zapnuto "Transfer-Complete"přerušení periférie DMA. Tím je automaticky vyvoláno přerušení pokud dojde k odběru N vzorků z kanálů.

Po příchodu přerušení od periférie DMA, kdy je odebrán daný počet vzorků, je potřeba vypočítat aritmetický průměr. Na základě průměru můžeme vypočítat žádané rozdíly napětí a ty odeslat do uživatelského rozhraní a ihned poté odešleme bod do grafu, který se k němu pouze přidá, nikoliv ho přepíše. Princip algoritmu pro odběr a odesílání vzorků můžeme vidět na obr. 7.3.



Obr. 7.3: Odběr a odesílaní vzorků

8 Realizace funkce PWM generátor

Jedním z požadavků mé diplomové práce je realizovat dvoukanálový PWM generátor. Ten je potřeba pro řadu experimentů, kdy student například měří kapacitu kondenzátoru nebo zkouší měnit úroveň svitu LED diody. Bez funkce PWM generátoru by tedy student v řadě úloh musel řešit zdroj signálu nějakým externím způsobem, např známým obvodem NE555.

8.1 Požadavky vedoucího

Vedoucí práce měl několik požadavků na funkce generátoru:

- Celkem 2 nezávislé kanály
- Možnost jemného nastavení frekvence
- Zobrazení reálné i požadované frekvence
- Možnost vypnutí a zapnutí funkce PWM generátoru

8.2 Pulsně šířková modulace - PWM

Zkratka PWM (pulse-width modulation), neboli pulsně šířková modulace značí pojem, kdy signál mění svoji amplitudu mezi dvěma hodnotami (typicky u mikrokontrolérů log.0 a log.1) s určitou frekvencí. Duty-cycle, neboli střída, signálu pak označuje poměr mezi dobou, kdy je signál ve vysoké a nízké úrovni, tedy čas po jakou dobu je v log.1 a v log.0. Duty-cycle se tedy často značí v "%".

8.3 Možné realizace

Signál PWM lze generovat v mikrokontroléru dvěma způsoby. Buďto softwarovým způsobem, kdy bychom přímo ovládali výstupní signál na dané GPIO bráně, nebo hardwarovým způsobem, za využití vnitřních časovačů a jejich capture/compare jednotky.

8.3.1 Softwarové generování PWM signálu

Signál PWM lze generovat softwarovým způsobem. Jednoduše budeme měnit výstupní signál na dané GPIO bráně na základě časování vnitřních hodin (viz příklad 6). Tento způsob je však velice neefektivní, neboť jádro stále vykonává výpočty a nelze při takovém generování dělat jakoukoliv jinou činnost.

```
1 while(1)
2 {
3 GPIO_SetPinHigh(PWM_PIN);
4 Delay(PWM_TIME_ON);
5 GPIO_SetPinLow(PWM_PIN);
6 Delay(PWM_TIME_OFF);
7 }
```

Zdrojový kód 6: Softwarové generování PWM signálu pomocí zpoždění

Dalším možným způsobem softwarové realizace PWM signálu je využití přerušení. Zvolíme si pevný časový callback, v němž měníme stav výstupu na základě vnitřní časové proměnné (viz př. 7). Oba zmíněné přístupy však nejsou vhodným příkladem jak vytvořit PWM signál, jsou použitelné pouze v případě, kdy mikrokontrolér nedisponuje žádným časovačem s capture/compare jednotkou.

```
/*
 * 1ms callback
 */
uint32_t _cnt = 0;

void Systick_IRQHandler(void)
{
    if(_cnt>30)
    {
      GPIO_SetPinLow(PWM_PIN);
    }
    else
    {
      GPIO_SetPinHigh(PWM_PIN);
    }
    _cnt++;
    if(_cnt>100)
    {
        _cnt=0;
    }
}
```

Zdrojový kód 7: Softwarové generování PWM signálu pomocí přerušení

8.3.2 Hardwarové generování PWM signálu

Nejlepším způsobem, pokud to mikrokontrolér umožňuje, je využiti vnitřního časovače s capture/compare jednotkou. Časovač je schopen v tzv "compare"režimu, kdy kontroluje vnitřní stav čítače oproti zadané hodnotě, změnit log. stav na výstupu mikrokontroléru. Tedy je schopen vytvářet PWM signál. Frekvence signálu, v případě generování za pomoci časovače, je dána frekvencí řídících hodin a hodnotou "reload"registru, tedy hodnotou při které časovač přeteče. Naopak jeho střída je dána "compare"hodnotou zapsanou v capture/compare registru.

Praktický příklad nastavení časovače můžeme vidět v příkladu 8, kde hodinový signál pro časovač je 64MHz, ARR (reload) registr je nastaven na hodnotu 64000 a CCR registr kanálu 1 je nastaven na hodnotu 32000. Tedy časovač přeteče vždy s frekvencí 1KHz a právě v polovině čítání změní svoji hodnotu z log.0 na log.1.

```

#define SYSTEM_CORE_CLOCK 64000000
void SetPWM(void)
{
    TIM3->ARR = 64000;
    TIM3->CCR1 = 32000;
}
```

Zdrojový kód 8: Softwarové generování PWM signálu pomocí přerušení

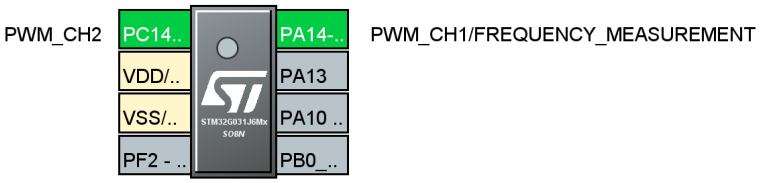
Pro zachování kompatibility s předešlým obdobným řešeni, je využito pinů č.8 pro kanál PWM_1 a pinu č.1 pro kanál PWM_2 na pouzdře SO8N.

8.4 Realizace PWM generátoru

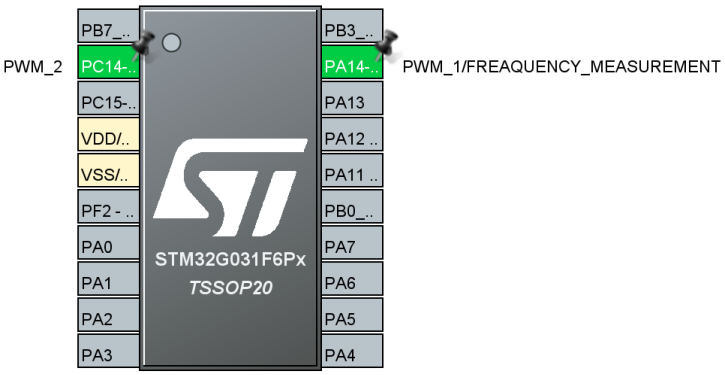
Ve mém řešení je využita hardwarová. Konkrétně je využito časovačů TIM17 a TIM2, kdy časovač TIM2 mění svoji funkcionalitu mezi měřením frekvence a generováním PWM signálu na základě volby uživatele. Nastavení frekvence a střídy signálu je řešeno obdobným

způsobem, jako u osciloskopu, tedy odesláním konfiguračního řetězce pro nastavení žádané frekvence a střídy. Konfigurační řetězec začíná identifikátorem "p", za nímž následují parametry nastavení, jako je daný kanál, požadovaná frekvence a střída.

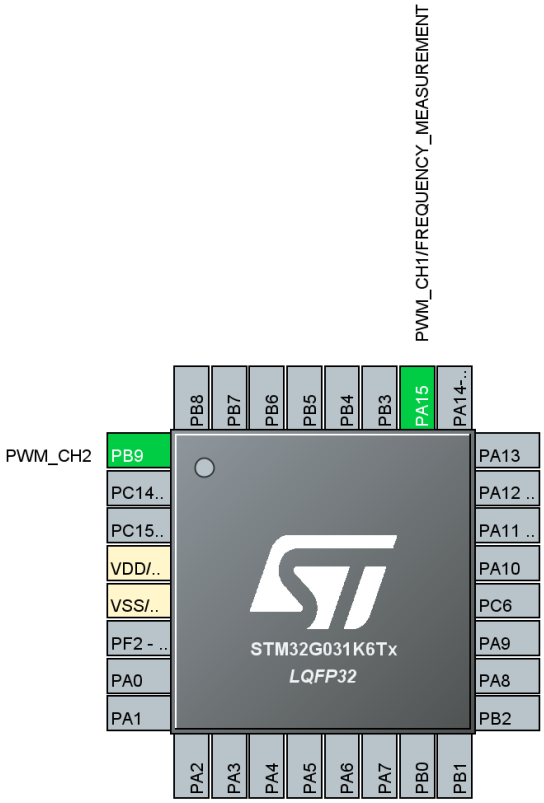
Vzhledem k charakteristice hardwarového čítače a jeho předěličky a auto-reload registru, tak je zároveň v uživatelském rozhraní zobrazena reálná nastavená frekvence. Ta se může totiž lišt (vzhledem k nutnosti celočíselného dělení řídící frekvence) od požadované lišit. U vyšších frekvencí se totiž může jednat o markantní rozdíl mezi žádanou a reálnou frekvencí. Tento problém však nelze vyřešit jiným způsobem, neboť limitující faktor pro rozlišení je řídící frekvence časovače, jenže je u mikrokontroléru STM32G031 maximálně 64MHz.



Obr. 8.1: Rozložení pinů STM32G031J6 -SO8N



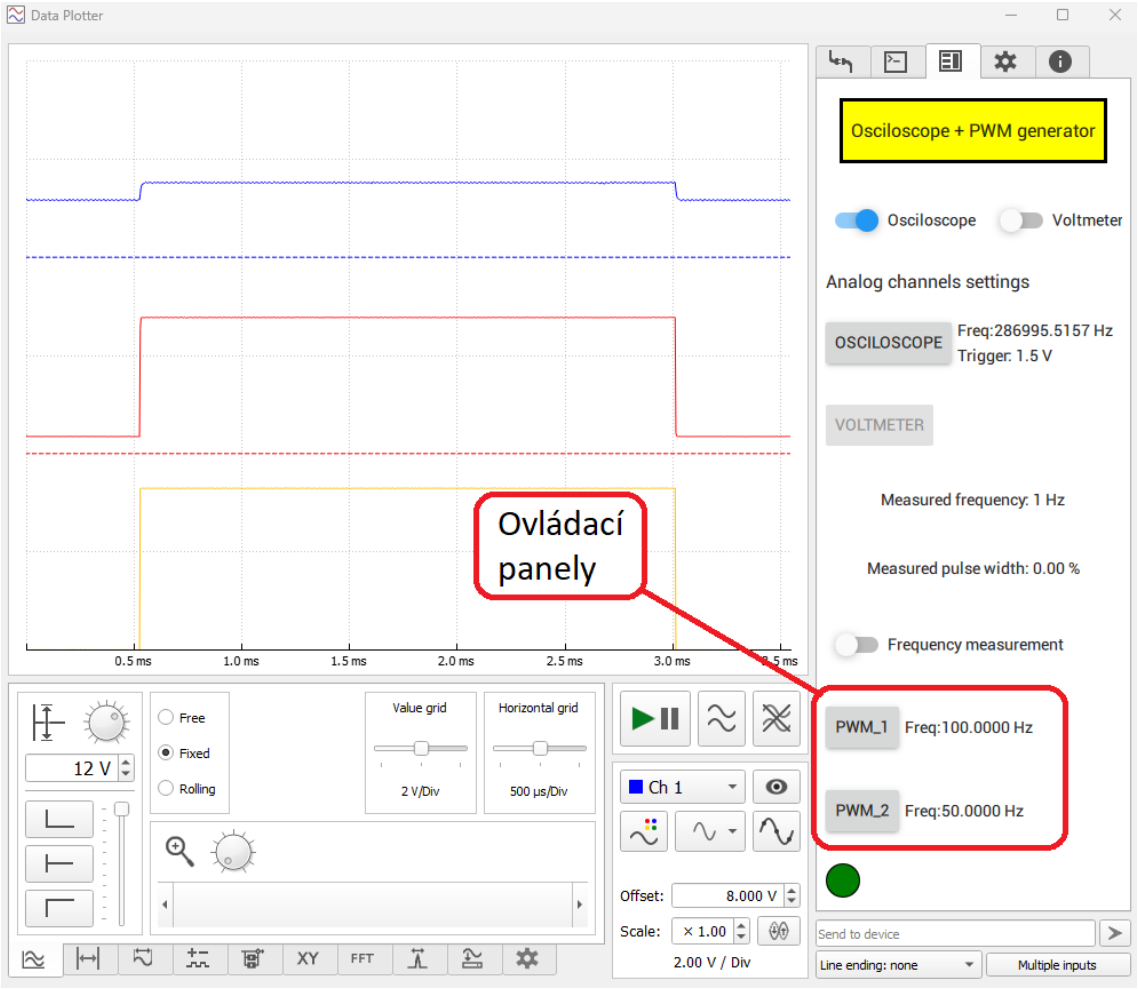
Obr. 8.2: Rozložení pinů STM32G031F6 -TSSOP20



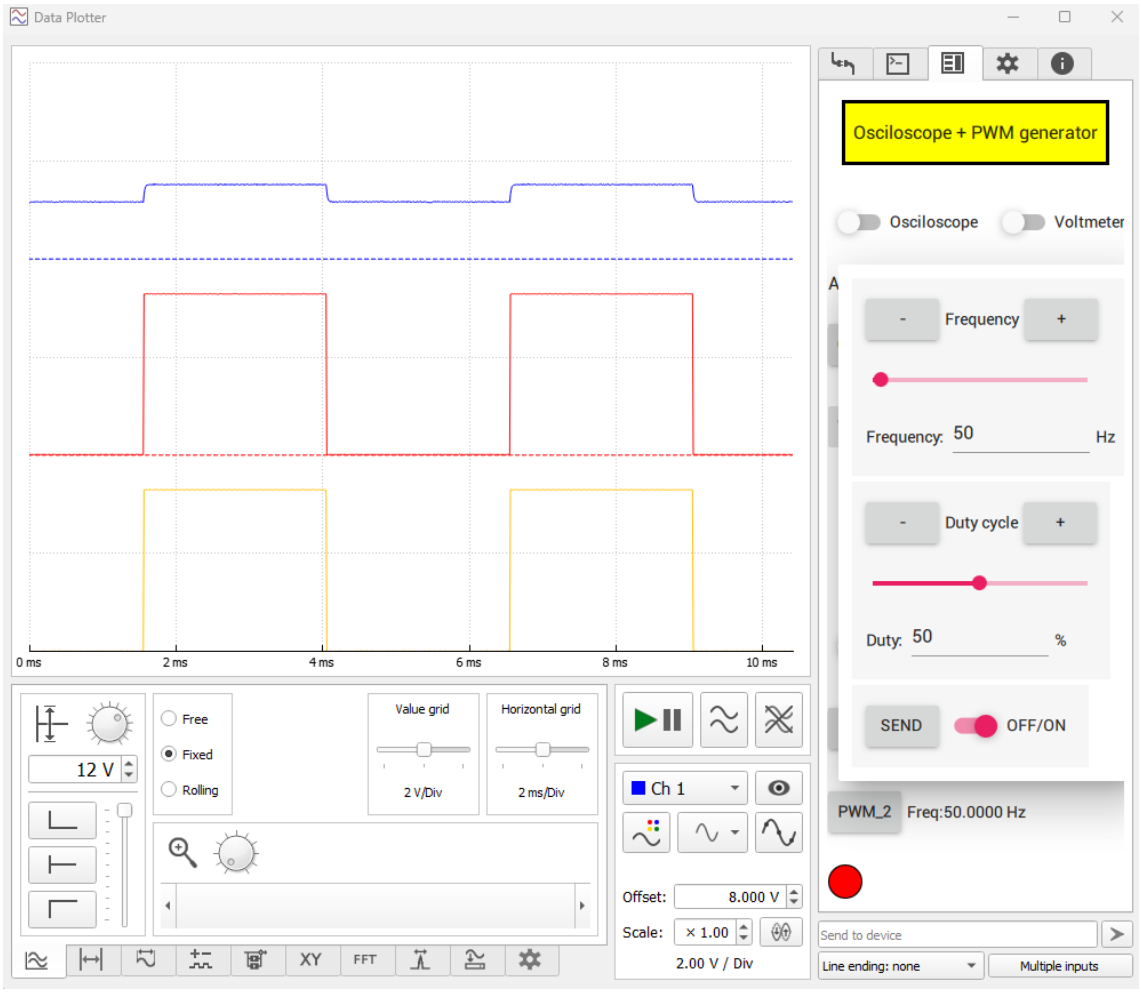
Obr. 8.3: Rozložení pinů STM32G031K6 -LQFP32

8.5 Uživatelské rozhraní

Uživatelské rozhraní obsahuje dvě tlačítka, každé pro jeden kanál. Obě obsahují identická nastavení, tedy uživatelem požadovanou frekvenci a střídu PWM signálu pro daný kanál. Zároveň lze jednotlivé kanály zapínat/vypínat dle potřeby. V hlavním menu je pak vidět reálná nastavená frekvence, vzhledem k použité předděličce a hodnotě v reload registru. Tato frekvence je potřebná pro správné nastavení stroboskopického vzorkování, kdy musí být vzorkovací frekvence a frekvence PWM signálu posunuty o co nejmenší časový krok.



Obr. 8.4: Tlačítka pro nastavování jednotlivých kanálů



Obr. 8.5: Konkrétní nastavení kanálu

9 Realizace funkce čítače

V této sekci je podrobně popsáno vytvoření funkce čítač frekvence, její implementace v softwaru a princip fungování.

9.0.1 Požadavky

Požadavky na funkci čítače zadané vedoucím mé diplomové práce jsou následující:

- Měření frekvence signálu
- Měření střídy signálu

Vzhledem k omezenému počtu pinů na pouzdře SO8N není potřeba aby byly všechny funkcionality dostupné zároveň, tedy je možné vytvořit v uživatelském rozhraní "přepínání"mezi jednotlivými funkcemi. Funkce čítače frekvence není nijak zásadní pro měření experimentů, proto byla zvolena jako sekundární funkce jednoho z generátorů PWM signálu (konkrétně PWM1 výstupu).

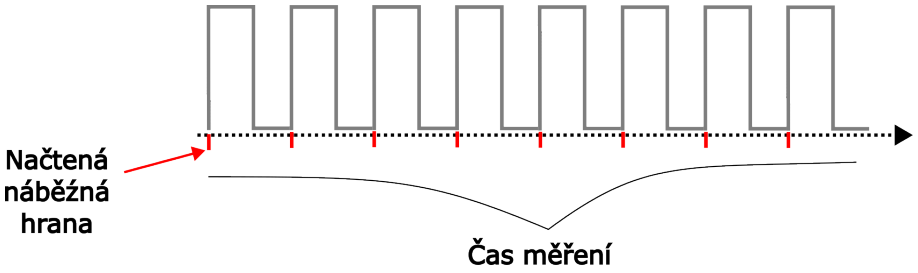
9.1 Možné způsoby řešení

čítání frekvence lze realizovat dvěma základními způsoby:

- Přímé měření frekvence
- Reciproční měření frekvence

9.1.1 Přímé měření frekvence

Přímé měření frekvence spočívá v v čítání počtu period měřeného signálu. Ty jsou čítány v přesných časových intervalech, tedy tímto způsobem můžeme dopočíst výslednou frekvenci měřeného signálu. Díky přímému měření jsme kvůli jednoduchosti schopni měřit i velmi vysoké frekvence. Tento přístup však neumožňuje současné měření velikosti střídy signálu. Z toho důvodu nebyl v práci použit.



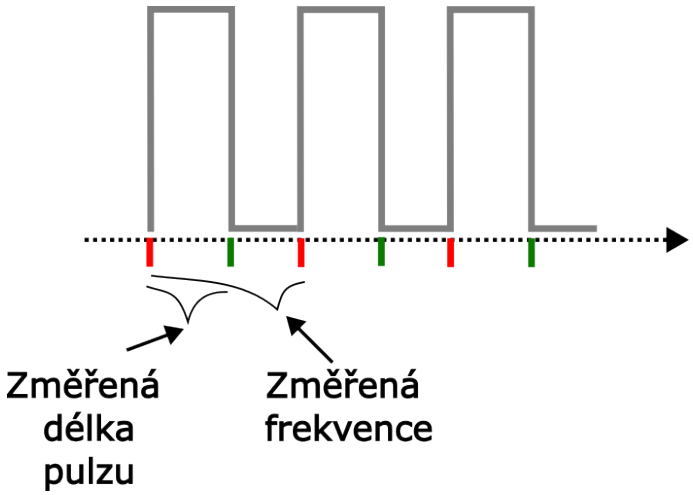
Obr. 9.1: Přímé měření frekvence

Na obr. 9.1 můžeme vidět názorný příklad přímého měření frekvence. časovač bude reagovat na každou náběžnou /sestupnou hranu signálu a následně na základě jejich počtu a separátního časového okna, za které byly načteny můžeme dopočítat výslednou frekvenci měřeného signálu. Zde můžeme vidět že s narůstajícím časem měření se snižuje chyba měření. Tedy pro velmi pomalé signály bychom potřebovali velmi dlouhé časové okno.

9.1.2 Reciproční měření frekvence

Na rozdíl od přímého měření frekvence, využívá reciproční měření změřený časový rozdíl mezi dvěma náběžnými hranami signálu. Tedy je vyčten počet hodinových pulzů řídícího signálu časovače načtených za 1 periodu měřeného signálu. Tímto způsobem můžeme měřit i velmi nízké frekvence bez nutnosti dlouhého časového kroku.

Vzhledem k recipročnímu principu měření lze zároveň měřit časové intervaly mezi náběžnou a spádovou hranou, čímž získáme informaci o velikosti střídy signálu, kterou mám dle požadavku vedoucího práce také měřit.



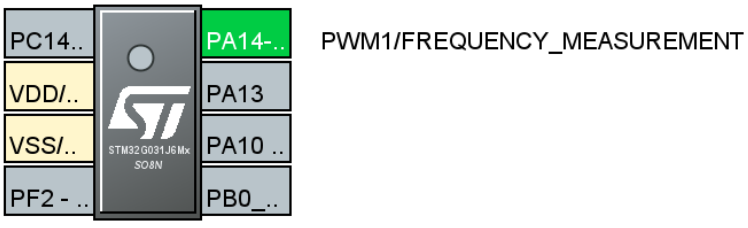
Obr. 9.2: Recipročí měření frekvence

Chyba u recipročního měření frekvence vzniká v omezené frekvenci řídícího signálu časovače. Ta určuje časové rozlišení, s jakým bude náběžná/sestupná hrana detekována a zaznamenán její čas.

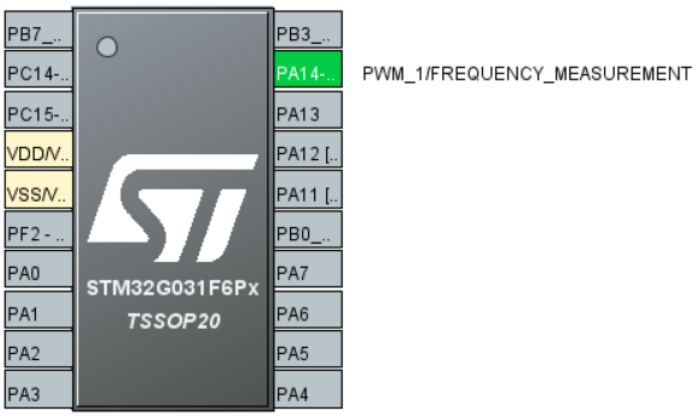
9.2 Princip realizace čítače

K řešení čítání frekvence a délky pulzu je na mikrokontroléru STM32G031J6 využit pin č.8 (konkrétně PA15), neboť disponuje výstupem capture/compare jednotky čítače TIM2. Čítač TIM2 je 32 bitový čítač, tedy dokáže načítat až 2^{32} řídících hodinových pulzů. Díky tomu jsme schopni měřit i velmi malé frekvence, bez jakékoliv implementace softwarového čítání přetečení jednotky čítače.

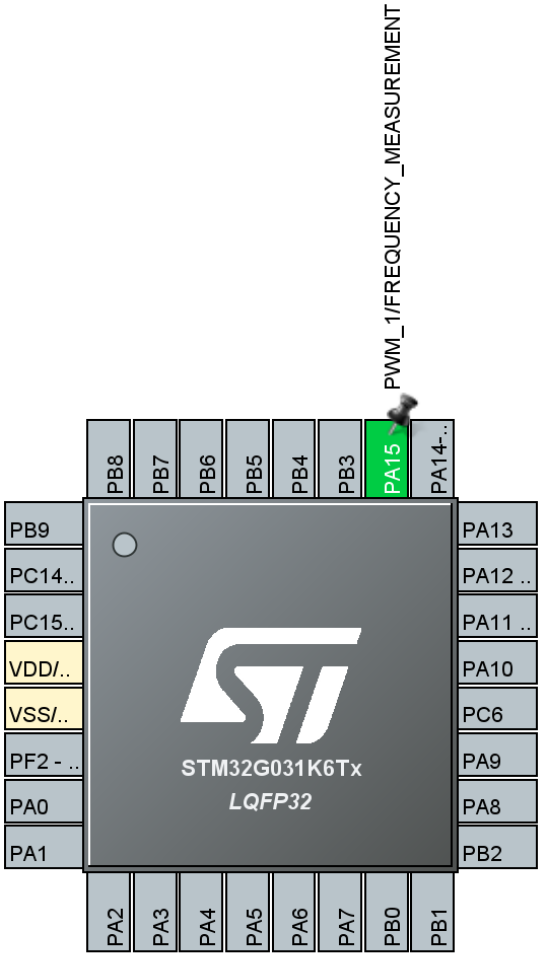
Jednotka čítače TIM2 je potřeba překonfigurovat z režimu, kdy generuje PWM signál do režimu "input capture",tedy vstupního režimu, v němž při náběžné/sestupné hraně signálu bude generovat přerušení. Konkrétně je potřeba zapnout v prvním cyklu reakci na náběžnou hranu. Typ detekované hrany signálu je periodicky "překlápěn"v přerušení tak, aby se střídavě měřila náběžná a následně sestupná hrana.



Obr. 9.3: Rozložení pinů STM32G031J6 -SO8N



Obr. 9.4: Rozložení pinů STM32G031F6 -TSSOP20



Obr. 9.5: Rozložení pinů STM32G031K6 -LQFP32

Na základě tohoto přerušení a hodnot z minulého přerušení (vždy je potřeba načíst alespoň dvě náběžné hrany a jednu sestupnou) jsme schopni dopočítat žádanou frekvenci

a střídu signálu.

Vzhledem k nutnosti a dopočítávání požadovaných hodnot a "překlápění"detekční hrany ze spádové na náběžnou a naopak se maximální rozsah možné měřené frekvence omezí malým výpočetním výkonem mikrokontroléru.

```
/*
 *  Frequency measurement
 */
void TIM2_IRQHandler(void) {

    TIM2 -> SR &= ~TIM_SR_CC1IF;
    if (myFreq.edgeIndex == 1) { //rising edge
        myFreq.timeStampLast = myFreq.timeStamp;
        myFreq.timeStamp = TIM2 -> CCR1;
        TIM2 -> CCER |= TIM_CCER_CC1P;
        if (myFreq.timeStamp < myFreq.timeStampLast) {
            myFreq.timeStampLast = UINT32_MAX - myFreq.timeStampLast;
            myFreq.freq = CORE_FREQUENCY / (myFreq.timeStamp + myFreq.
    timeStampLast);
        } else {
            myFreq.freq = CORE_FREQUENCY / (myFreq.timeStamp - myFreq.
    timeStampLast);
        }
        myFreq.edgeIndex = 0;

        myFreq.timeStampPulseWidthLast = TIM2 -> CCR1;
    } else { //falling edge
        TIM2 -> CCER &= ~TIM_CCER_CC1P;
        myFreq.timeStampPulseWidth = TIM2 -> CCR1;
        if (myFreq.timeStampPulseWidth < myFreq.timeStampPulseWidthLast) {
            myFreq.timeStampPulseWidthLast = UINT32_MAX - myFreq.
    timeStampPulseWidthLast;
            myFreq.pulsewidth = myFreq.timeStampPulseWidth - myFreq.
    timeStampPulseWidthLast;
            myFreq.edgeIndex++;
        } else {
            myFreq.pulsewidth = myFreq.timeStampPulseWidth - myFreq.
    timeStampPulseWidthLast;
            myFreq.edgeIndex++;
        }
    }
}
```

Zdrojový kód 9: Zprácování dat k měření frekvence a střídy

9.3 Zapnutí funkce

Funkce se zapíná v hlavním menu uživatelského rozhraní tlačítkem na obr. 9.6. Po zapnutí funkce se automaticky vypne generování PWM signálu a pin se přepne do režimu čtení frekvence.

Obr. 9.6: Zapnutí funkce měření střídy a frekvence

9.4 Odesílání dat do PC aplikace

Data obsahující vypočítanou frekvenci a střídu se periodicky odesílají do uživatelského rozhraní, kde jsou následně zobrazena ve formě dvou textových polí.

10 Komunikační protokol s PC aplikací

Aplikace Data Plotter nabízí velmi široký a robustní komunikační protokol skrze sériové rozhraní USART. Komunikaci lze provozovat na téměř jakékoliv rychlosti (baud rate). Umožňuje tvorbu uživatelského rozhraní, přijímaní vzorků signálu i změnu vlastních nastavení, jako je měřítko na časové ose, mazání kanálů, či přijímání univerzálních dat pro použití v uživatelském rozhraní.

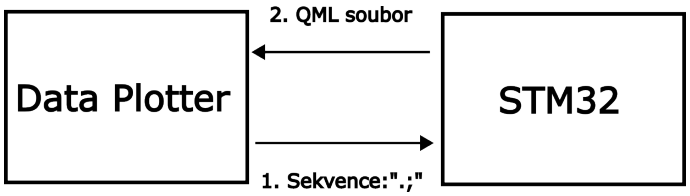
Každý komunikační balík ("packet") dat je zahájen sekvencí **"$$X"**, kde **X** značí druh přenosu. Různé druhy přenosu můžeme vidět v tabulce 2. Nakonec je každý balík zakončen znakem **';'**, který slouží jako korektní ukončení přenosu odesílaného balíku.

Typ zprávy	Funkce
P	přidání bodu do grafu
C	přidání kanálu do grafu
L	přidání logického kanálu do grafu
B	přidání logického bodu do grafu
T	výpsání do terminálového okna
S	nastavení aplikace
I	informační zpráva
W	varovná zpráva
X	chyba zařízení - dojde k odpojení zařízení
E	echo
Q	Odeslání QML souboru
U	neznámý řetězec, data jsou zahozena

Tabulka 2: Popis komunikačních identifikátorů [1]

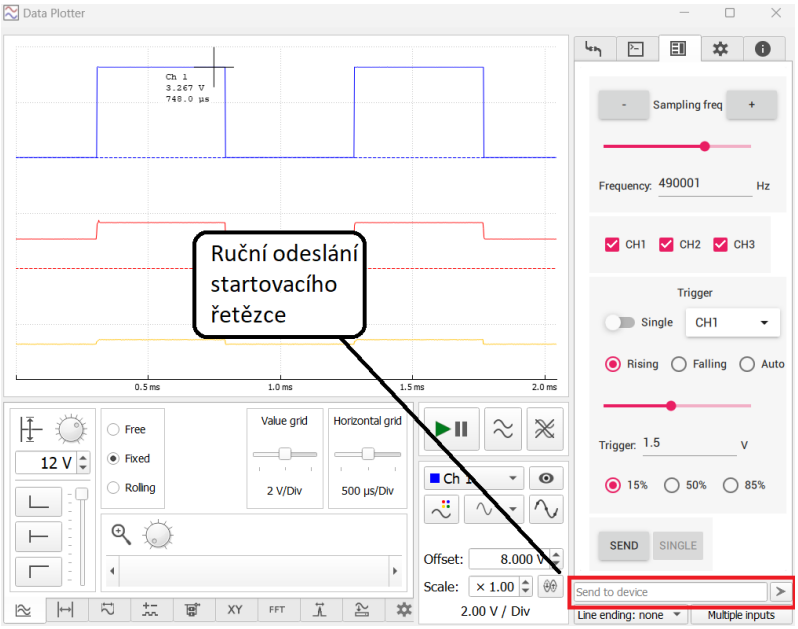
10.1 Prvotní připojení a nahrání QML souboru

Po připojení mikrokontroléru k PC za pomoci UART-USB převodníku je nejprve nutné nahrát do aplikace Data Plotter uživatelské rozhraní. To je do zařízení odesláno až ve chvíli kdy je do mikrokontroléru z aplikace zaslán řetězec **".;"**. Po odeslání QML souboru je ihned spuštěn jeden kanál osciloskopu v režimu force trigger. Výstup PWM generátoru je vypnutý.



Obr. 10.1: Nahrání QML souboru

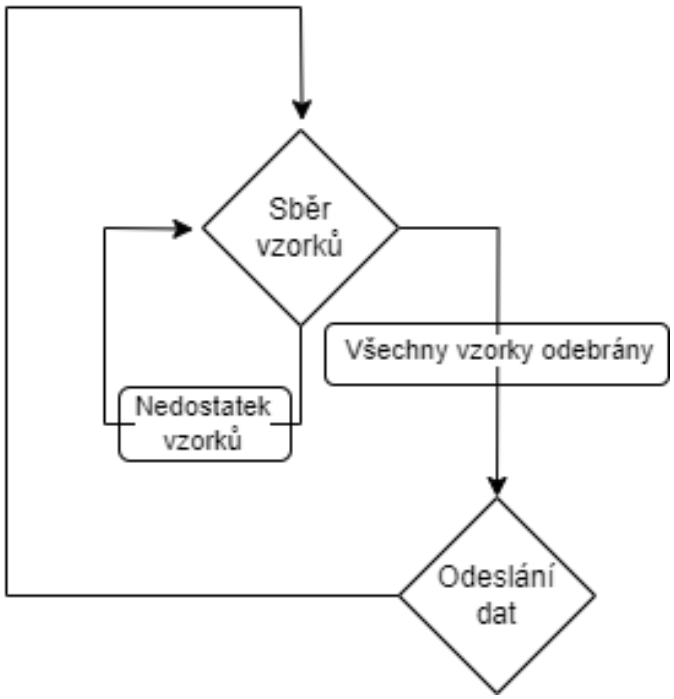
Aplikace Data Plotter umožňuje nakonfigurovat "příkaz po spuštění", tedy do aplikace lze zadat řetězec, který je vždy po stisknutí tlačítka "Connect"ihned odeslán do mikrokontroléru. Toto nastavení je možné i uložit, takže při dalším otevření aplikace se řetězec odesílaný po připojení vždy předvyplní, takže není nutné jej nastavovat znovu. Startovací sekvence lze zadat i ručně, skrze textové pole v dolním pravém rohu aplikace10.2, v případě že se uživateli nechce zapisovat řetězec v nastavení.



Obr. 10.2: Ruční odeslání startovací sekvence

10.2 Odesílání vzorků

K odesílání vzorků lze využít několik možností. Aplikace umožňuje buď zápis celého kanálu/kanálů naráz, nebo přidávání bodů jeden po druhém. V režimu osciloskop je využito vždy zápisu celého kanálu zároveň, neboť je vždy potřeba zobrazit celý jeden záznam dohromady. Naopak při využití voltmetru jsou vzorky zasílány jeden po druhém, což lze vzhledem k nízké vzorkovací frekvenci (100Hz) stíhat.



Obr. 10.3: Princip odesílání vzorků

K odesílání vzorků dochází vždy po odběru daného počtu vzorků, v případě osciloskopu se jedná o naplnění kruhového bufferu a v případě voltmetru o počet vzorků, z kolika je realizováno průměrování. Vždy společně s daty pro jednotlivé kanály, ať už pro osciloskop, nebo pro voltmetr, jsou zároveň odeslána periodická data obsahující jednotlivá napětí, měřenou frekvenci z časovače nebo heartbeat signál pro signalizaci triggeru.

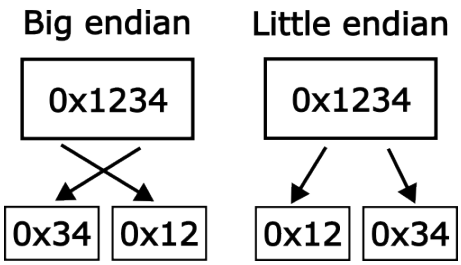
10.2.1 Odeslání kanálu/skupiny kanálů

Za použití komunikačního identifikátoru **"$$C"** lze odeslat skupinu kanálu do aplikace Data Plotter. Z komunikačním kanálem následuje několik parametrů, určujících o jaké kanály se jedná, jaký je časový krok a jakého typu bude aplikace data očekávat. Dále lze naspecifikovat v hlavičce řetězce minimální a maximální hodnotu, na kterou se mají data "rozložit". Tato možnost je využita v aplikaci osciloskop, kdy změřená data v 12ti bitové reprezentaci jsou automaticky reprentována jako hodnota mezi 0-3.3V.

```
1
2     USART_SendString("$$C1,0.001,20,12,3.3;u2
      ????????????????????????????????????????;");
```

Zdrojový kód 10: Příklad odeslání dat do jednoho kanálu

V ukázce 10 můžeme vidět příklad odeslání dat pro kanál č.1, s intervalem mezi vzorky 1ms, 20 vzorky celkem, 2^{12} maximální hodnotou a a přemapováním na 0-3.3V. Přepínač **u2** značí, že se jedná o data v unsigned short podobě, tedy 16ti bitová data. Zároveň syntaxe je case sensitive, a zda je písmeno velké či malé značí jestli jsou data očekávaná v little endian, nebo big endian notaci. Tedy syntaxe "U2"značí big endian, naopak "u2"little endian. Mikrokontroléry řady STM32G0 využívají little endian notaci. To znamená, že data jsou v paměti uspořádána v pořadí od LSB bajtu po MSB bajt viz obr.10.4.



Obr. 10.4: Rozdíl mezi little endian a big endian architekturou

10.2.2 Odeslání jednoho bodu

V režimu osciloskopu jdou data do grafu odesílána "bod po bodu", tedy ne celý záznam najednou. Toho lze docílit použitím identifikátoru **"$$P"**.

```
1     USART_SendString("$$P-auto,xxxx,yyyy,zzzz;");
```

Zdrojový kód 11: Příklad přidání jednoho bodu do grafu

V této konkretní realizaci využívám identifikátor **"$$P"** společně s přepínačem "-auto", který vytváří automatickou časovou osu, tedy program Data Plotter si sám měří časový krok mezi jednotlivými vzorky. Následují parametry pro data 3 kanálů a celý řetězec je opět ukončen středníkem.

10.3 Konfigurace mikrokontroléru

Pro přenastavení funkcí osciloskopu, PWM generátoru, voltmetru a čítače frekvence je zhotoven jednoduchý komunikační protokol, který je odeslán vždy po stisknutí tlačítka "send"v uživatelském rozhraní.

Tento komunikační protokol má jako první znak vždy písmeno, označující danou funkcionalitu, ke které se vztahuje. Po tomto znaku následuje '_' za nímž je druhý identifikátor, který označuje co se s danou funkcí bude dělat, tedy například zda se bude PWM generátor zapínat/vypínat, nebo měnit střída či frekvence. Celý řetězec je vždy výlučně ukončen znakem ";", který označuje poslední znak ze sekvence přenosu a tedy celý řetězec může být vyhodnocen a zpracován.

V ukázce zdrojového kódu 12 můžeme vidět, že pro zpracování dat je využité přerušení z periférie USART volané při každén novém příchozím znaku. V případě že příkozí znak je ";", je příjem ukončen a celý řetězec je vyhodnocen.

```
1
2  /*
3  USART1 - rx handler
4  */
5  void USART1_IRQHandler(void)
6  {
7      USART1->ICR |= USART_ICR_NECF;
8      uint8_t _rxChar = USART1->RDR;
9      if(_rxChar==';')
10     {
11         //comand not valid
12         if(uartRxChar_cnt>UART_RX_CMD_LEN)
13         {
14             //comand not valid
15             uartRxChar_cnt = 0;
16             for(uint32_t i = 0; i< UART_RX_CMD_LEN;i++)
17             {
18                 UartRxBuffer[i] = 0;
19             }
20             return;
21         }
22         //handle comand
23         else
24         {
25             switch(UartRxBuffer[0])
26             {
27             //......code to process data
28             }
29         }
30     }
31         else
32         {
33             //save current char
34             UartRxBuffer[uartRxChar_cnt] = _rxChar;
35         }
36 }
```

Zdrojový kód 12: Princip zpracování příchozích dat

11 Zhodnocení dosažených výsledků

Realizovaný přístroj využívá mikrokontroléru STM32G031. Zařízení nahrazuje funkce tří kanálového osciloskopu / voltmetru, dvoukanálového PWM generátoru a čítače pro měření frekvence a střídy signálu. Takto navržený firmware funguje na všech následujících mikrokontrolérech:

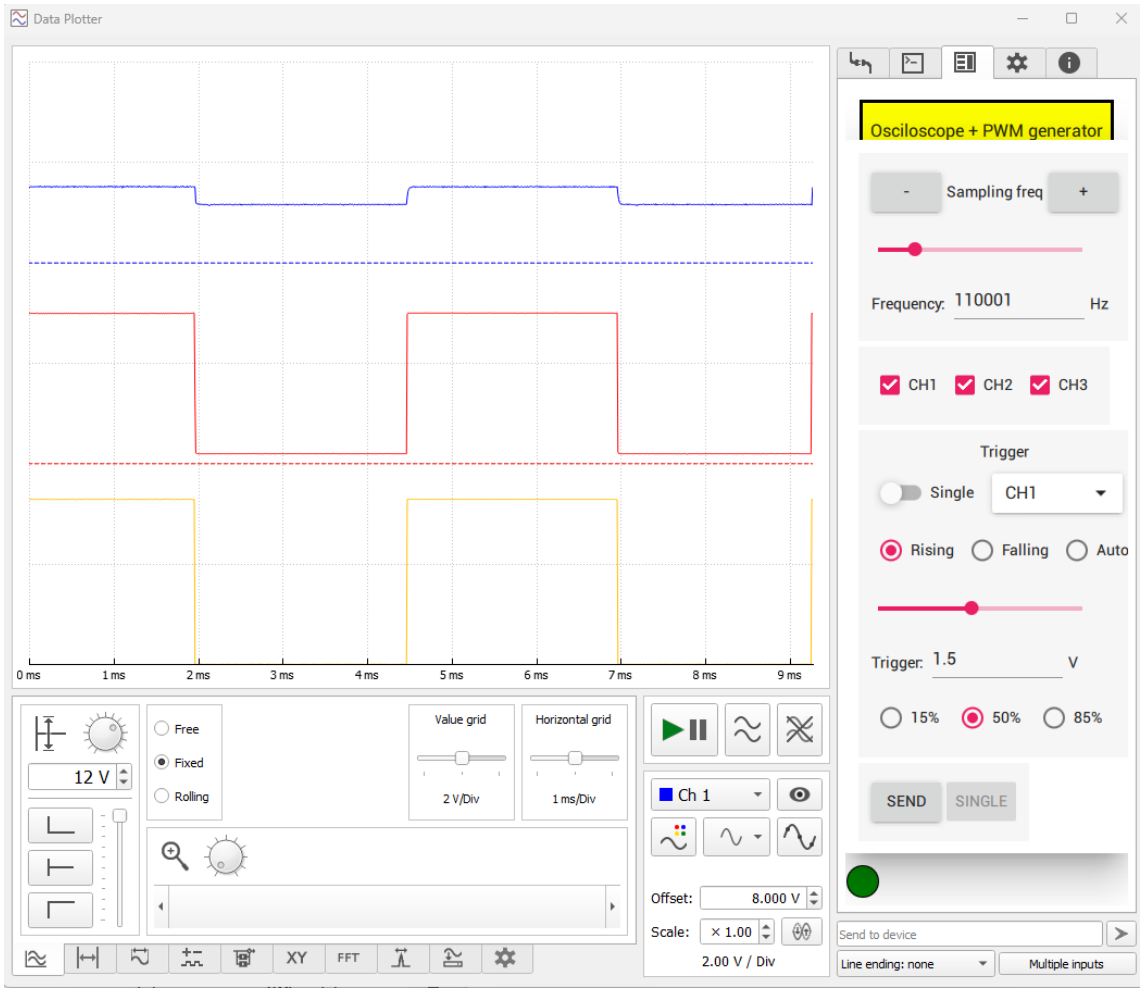
- STM32G031J6 - pouzdro SO8N
- STM32G031F6 - pouzdro TSSOP20
- STM32G031K6 - pouzdro LQFP32

11.1 Osciloskop

Funkce osciloskopu je schopna odebírat vzorky s rychlostí až 2MSa/s(na 3 ranály) a délkou záznamu 3072 vzorků. Dále je schopna detekce spouštěcí podmínky (triggeru) na náběžnou, či sestupnou hranu. V případě kdy si uživatel není jistý, jaký trigger zvolit, existuje zde varianta force trigger, která udělá záznam bez hledání spouštěcí podmínky. Vzorkovací frekvence vztažená na jeden kanál lze měnit po malých krocích kvůli možnosti využiti pro vzorkování v ekvivalentním čase.

Dosažené parametry:

- Rychlost vzorkování 1Sa/s - 2MSa/s
- Režim spouštění normal/single
- Způsoby spouštění Force trigger/rising edge/falling edge
- Délka záznamu 3072 vzorků



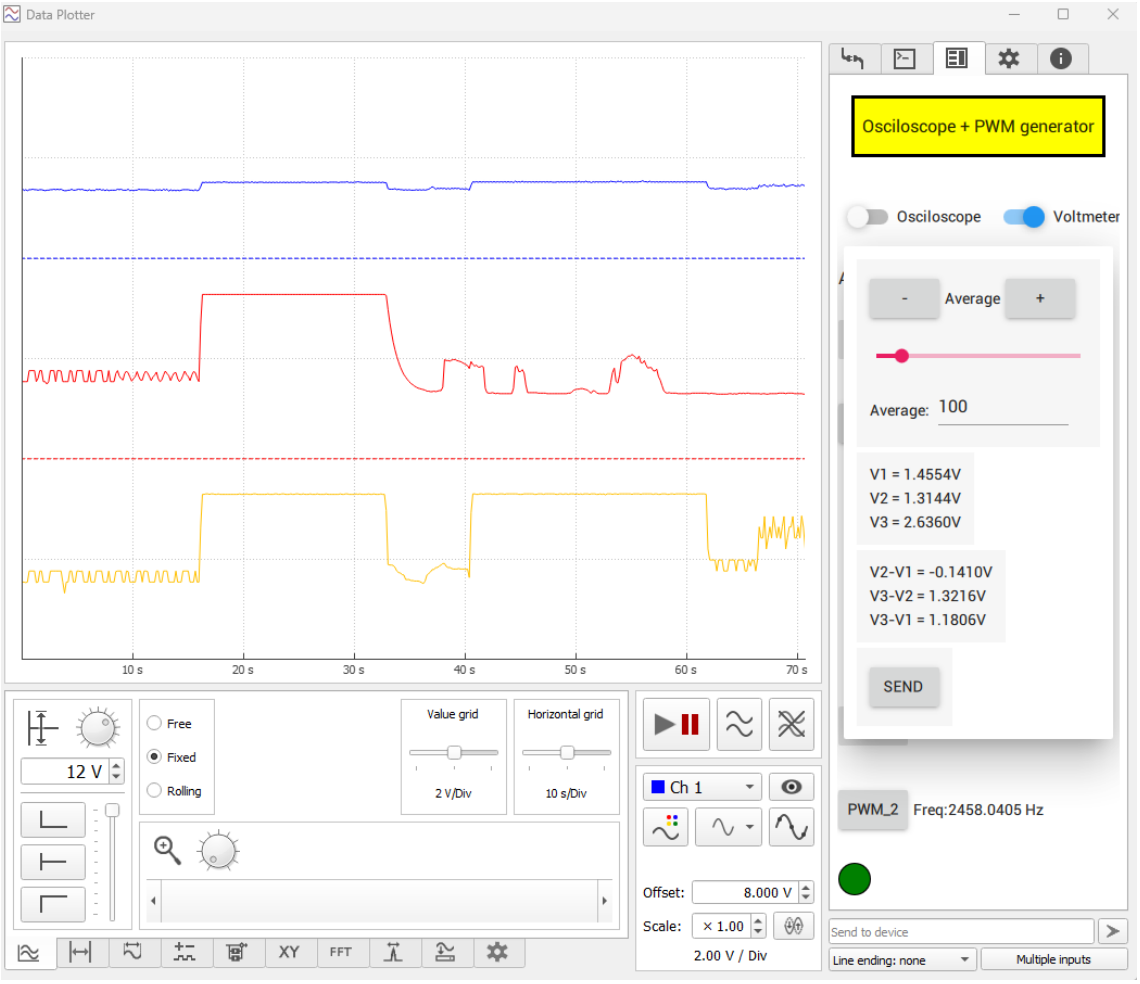
Obr. 11.1: Uživatelské rozhraní osciloskopu

11.2 Voltmetr

Funkce voltmetru disponuje 3mi kanály schopnými kontinuálního záznamu monitorovaného napětí, zároveň je schopen průměrovat naměřené vzorky a počítat rozdíly napětí mezi jednotlivými kanály.

Parametry osciloskopu:

- Vzorkovací frekvence 100 Hz
- Průměrování z 1-1024 vzorků
- Kontinuální záznam měřených napětí
- Zobrazení rozdílů napětí



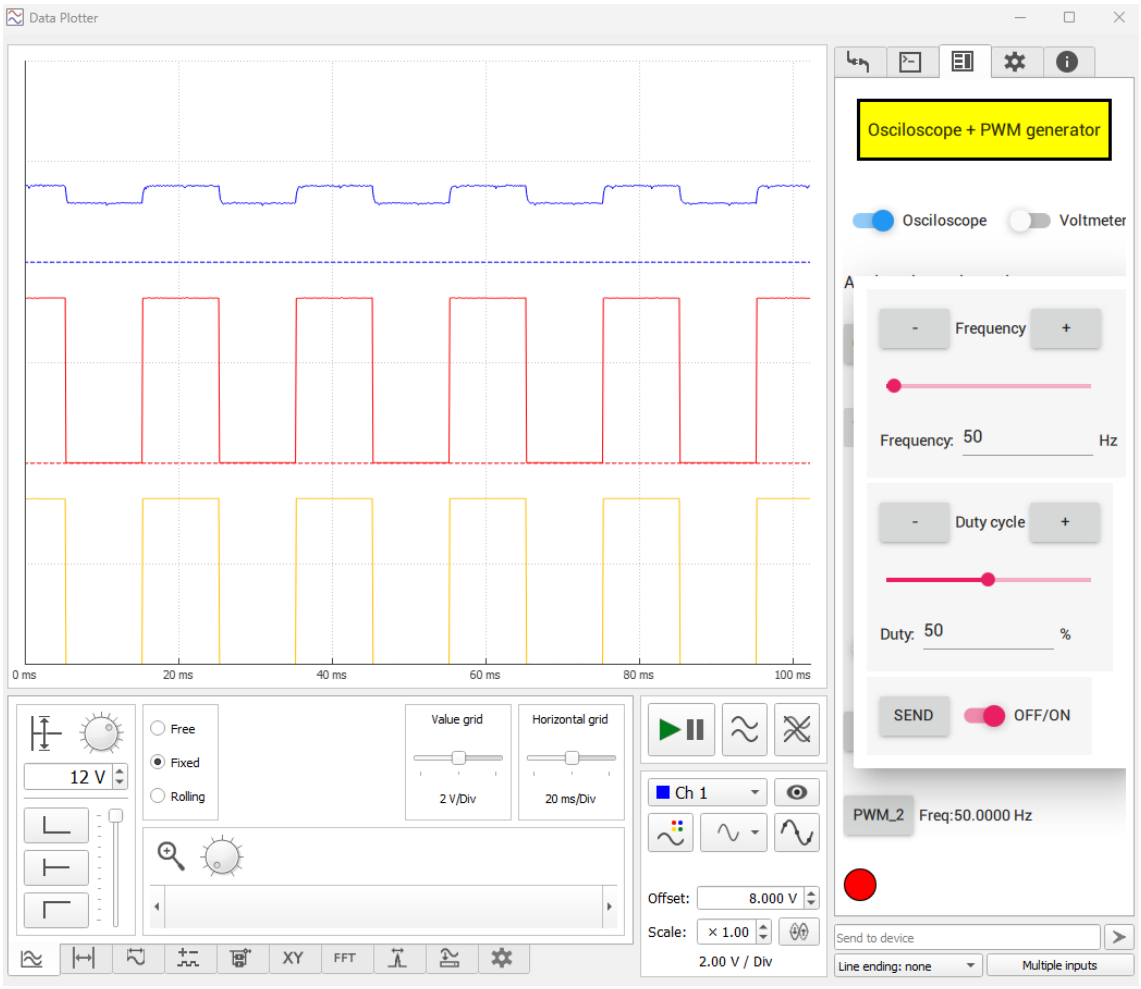
Obr. 11.2: Uživatelské rozhraní voltmetru

11.3 Generátor PWM signálu

Režim PWM generátoru nabízí celkem až 2 oddělené kanály schopné nezávislého nastavování střídy i frekvence. Kanál č.1 sdílí výstupní pin s funkcí čítače, tedy nemohou být zapnuté obě funkce najednou.

Parametry generátoru:

- Maximální frekvence 8 MHz
- Jemné nastavení frekvence pro vzorkování v ekvivalentním čase
- Dva nezávislé kanály
- Nastavení střídy



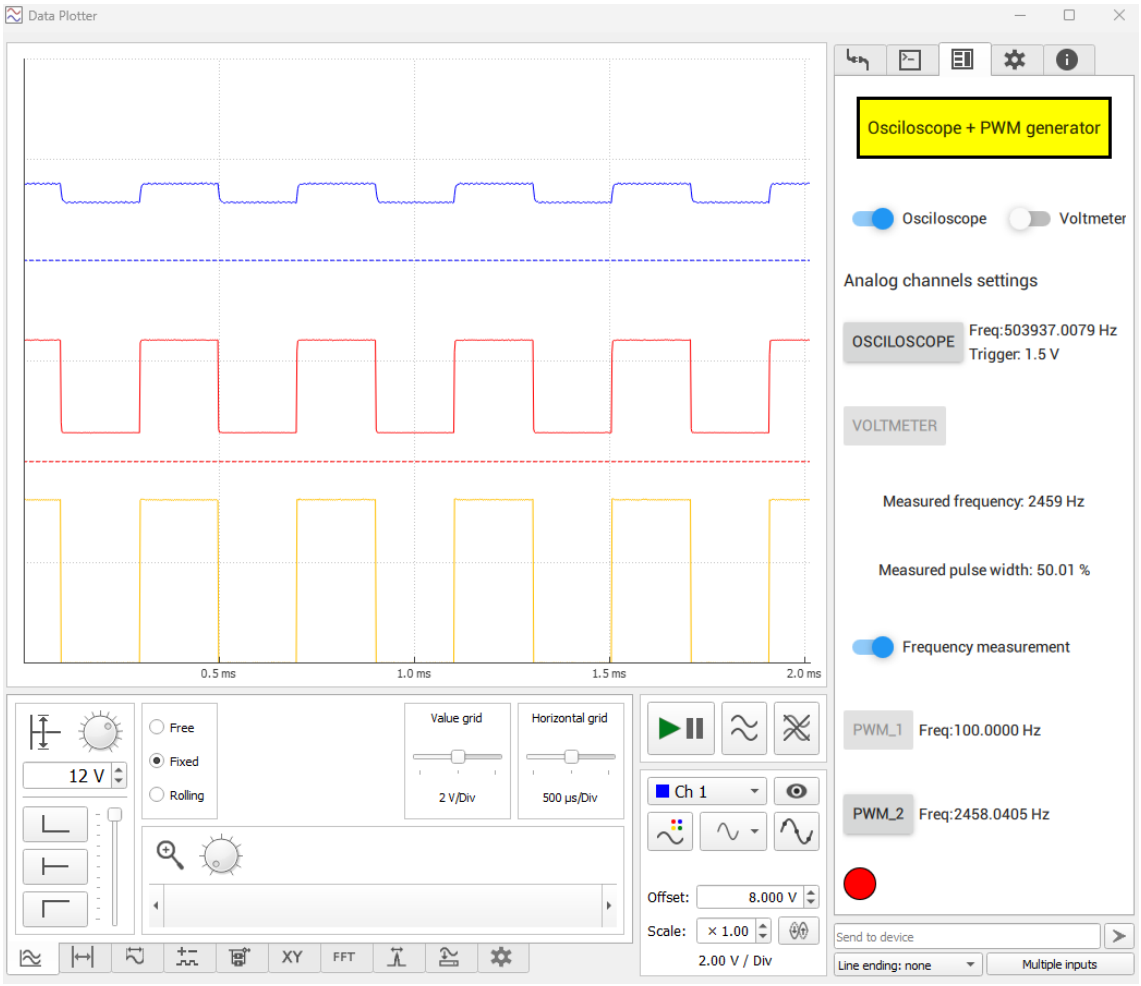
Obr. 11.3: Uživatelské rozhraní PWM generátoru

11.4 Čítač pro měření frekvence a střídy signálu

V zařízení je navržen čítač pro měření frekvence a střídy pulsních signálů dostupný jako sekundární funkce kanálu č.1 PWM generátoru.

Parametry čítače:

- Maximální měřitelná frekvence 100 kHz
- Přepínatelná funkcionalita namísto PWM kanálu č. 1
- Měření velikosti střídy



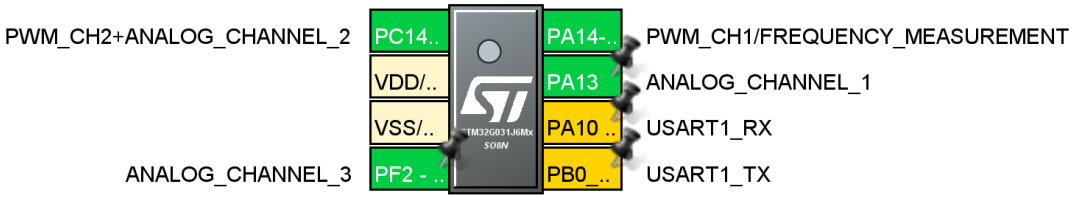
Obr. 11.4: Uživatelské rozhraní se změřenou frekvencí a střídou signálu

11.5 Uživatelské rozhraní

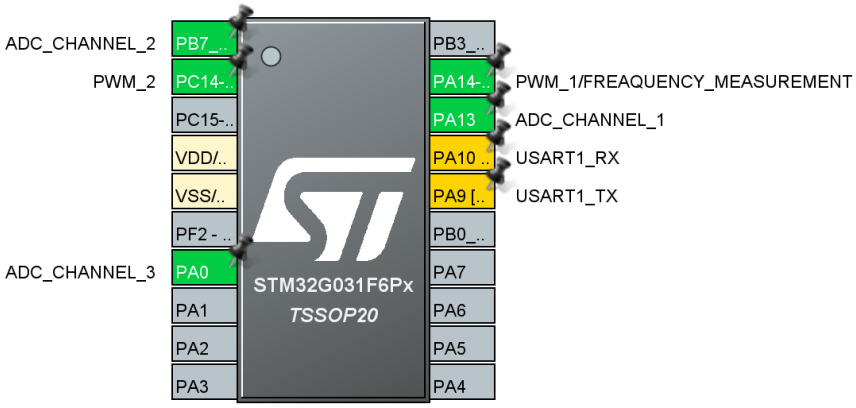
Na rozdíl od předešlých verzí obdobného zařízení, zde nebyla potřeba tvorby vlastní PC aplikace, ale pro zobrazení naměřených dat a ovládání byla využita univerzální PC aplikace Data Plotter [1], v níž je navrženo unikátní uživatelské rozhraní pro ovládání a nastavování jednotlivých funkcí zařízení.

Toto uživatelské rozhraní bylo vytvořeno za pomoci jazyka QML, proprietárního jazyka Qt frameworku používaného pro tvorbu grafických aplikací. Takto vytvořené rozhraní je po připojení mikrokontroléru do aplikace nahráno a skrze něj probíhá komunikace mezi PC aplikací a mikrokontrolérem.

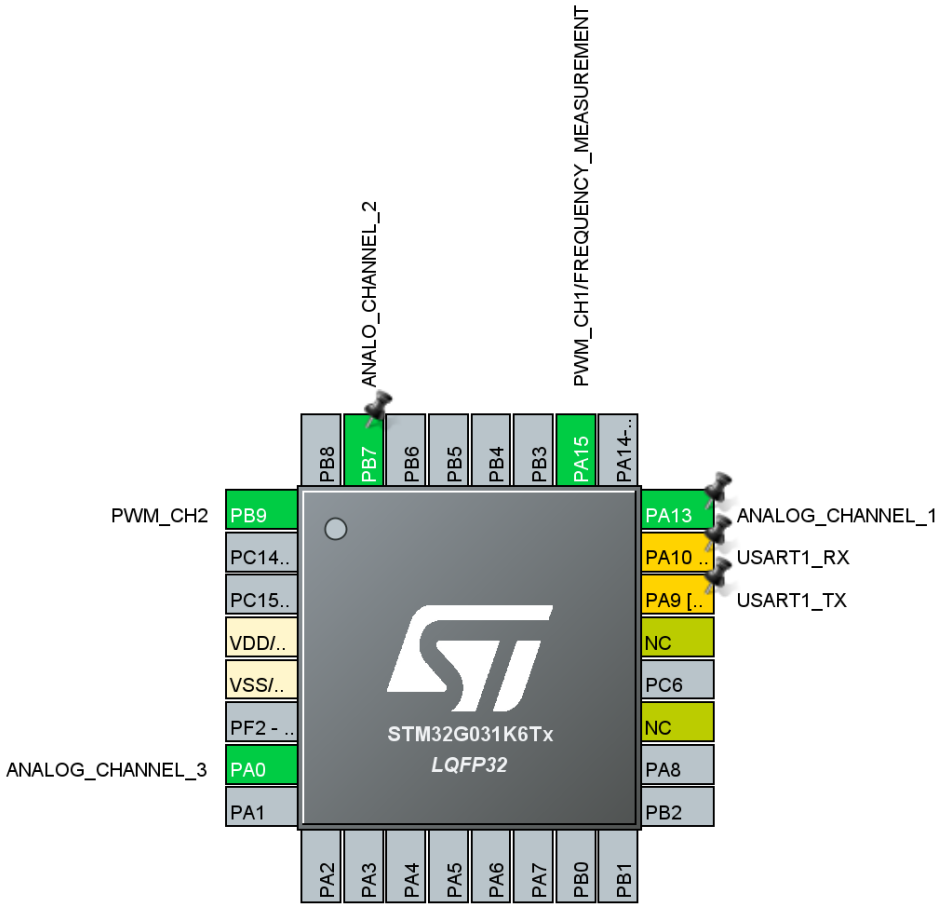
V diplomové práci jsem se potýkal s mnoha problémy spojenými s omezeným výpočetním výkonem mikrokontroléru, velikosti paměti RAM a velmi malého počtu výstupních pinů (v případě verze pro pouzdro SO8N). Dále jsem detailně řešil problematiku korektní hardwarové implementace "triggeru"u funkce osciloskop a implementace recipročního měření frekvence signálu.



Obr. 11.5: Výsledný pinout STM32G031J6 -SO8N



Obr. 11.6: Výsledný pinout STM32G031F6 -TSSOP20



Obr. 11.7: Výsledný pinout STM32G031K6 -LQFP32

12 Závěr

V diplomové práci byl realizován softwarově definovaný přístroj pro mikrokontrolér STM32G031 obsahující funkce tříkanálového osciloskopu schopného vzorkování v reálném i ekvivalentním čase, tříkanálový voltmetr, dvoukanálový PWM generátor a čítač pro měření frekvence a délky impulsů. Zařízení jsem otestoval na pouzdře mikrokontrolérech STM32G031J6 - pouzdro SO8N, STM32G031F6 - pouzdro TSSOP20 a STM32G031K6 - pouzdro LQFP32. Detailní parametry jsou uvedeny v sekci 11.

V práci jsem se detailně seznámil s obsluhou jednotlivých periférií mikrokontroléru, konkrétně pak analogově digitálního převodníku ADC, časovačů, sériového rozhraní USART nebo řadiče pro přímý přístup do paměti DMA. Uživatelské rozhraní jsem navrhl v jazyce QML, které je po připojení nahráno do PC aplikace Data Plotter [1], v níž se zobrazuje.

Zhotovený přístroj se bude využívat v rámci laboratorní výuky různých předmětů v oblasti elektroniky a senzorové techniky na katedře měření. Mimo to nalezne využití též v akcích organizovaných ČVUT - FEL pro středoškolské studenty, jako např. ETC22. Vzhledem k jednoduchosti konstrukce a nízké ceně má předpoklady k využití též v hromadné výuce na středních školách i v různých zájmových kroužcích mládeže.

Zařízení jsem otestoval na pouzdře mikrokontrolérech STM32G031J6 - pouzdro SO8N, STM32G031F6 - pouzdro TSSOP20 a STM32G031K6 - pouzdro LQFP32.

Vzhledem na dosažené parametry uvedené v kapitole 11 se domnívám, že jsem zadání splnil.

Reference

[1] Maier, J.: Univerzální GUI pro osciloskopické PC aplikace; ČVUT- FEL, 2021

[2] Yiu, J.: The Definitive Guide to ARM Cortex -M0 and Cortex-M0+ processors; 2015

[3] 25th IMEKO TC-4 INTERNATIONAL SYMPOSIUM ON MEASUREMENT OF ELECTRICAL QUANTITIES; 2022, Bresica, Italy; ISBN: 978-92-990090-1-7

[4] Berlinger, A.: Implementace přístrojových funkcí s využitím mikrořadičů STM32, ČVUT-FEL, 2016

[5] STMicroelectronics. DS12992. Arm® Cortex®-M0+ 32-bit MCU, up to 64 KB Flash, 8 KB RAM, 2x USART, timers, ADC, comm. I/Fs, 1.7-3.6V. URL: https://www.st.com/resource/en/datasheet/stm32g031j6.pdf

[6] STMicroelectronics. RM0444. STM32G0x1 advanced Arm®-based 32-bit MCUs. URL: https://www.st.com/resource/en/reference_manual/dm00371828-stm32g0x1-advanced-armbased-32bit-mcus-stmicroelectronics.pdf