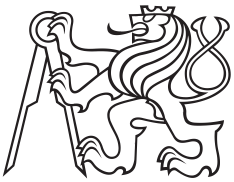
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Arbitrary – funkční generátor s mikrořadičem STM32

Bc. Jindřich Rozkopal

Vedoucí: doc. Ing. Fischer Jan CSc.
Obor: Elektronika a komunikace
Studijní program: Elektronika
Květen 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rozkopal** Jméno: **Jindřich** Osobní číslo: **457138**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Arbitrary – funkční generátor s mikrořadičem STM32

Název diplomové práce anglicky:

Arbitrary Function Generator Based on STM32 Microcontroller

Pokyny pro vypracování:

Navrhněte a s využitím mikrořadičů STM32G431 a STM32F303 realizuje přístroj – dvoukanálový arbitrary funkční generátor orientovaný pro použití v laboratorní výuce na ČVUT- FEL. Přístroj bude mimo základních signálů typu sinus, trojúhelník a dalších umožňovat také generaci uživatelsky definovaných průběhů signálu. Pro využití při proměřování frekvenční charakteristiky obvodů bude možná také generace signálu s proměnnou frekvencí – režim „sweep“. Ovládání přístroje bude z PC prostřednictvím rozhraní USB. Pro ovládání se orientujte na PC Aplikaci DataPlotter. Zařízení otestujte a zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

1. Yiu, J.: The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors,
2. STMicroelectronics: RM0440 Reference manual STM32G4 Series, 2019
3. Maier J.:Univerzální GUI pro osciloskopické PC aplikace, bakalářská práce, 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: **09.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

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

Rád bych poděkoval svému vedoucímu doc. Ing. Janu Fischerovi CSc. za odborné vedení práce, věcné připomínky a vstřícnost při konzultacích. Také srdečně děkuji rodině, kamarádům a blízkým za podporu během nelehkých chvil studia.

Prohlášení

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

V Praze, 20. května 2022

.....
Bc. Jindřich Rozkopal

Abstrakt

Tato práce se zabývá návrhem dvoukanálového funkčního generátoru založeného na mikrokontrolérech STM32. Přístroj je vyvíjen pro potřeby výuky na laboratorních cvičeních ČVUT FEL. Základním módem funkce přístroje je generování dvou signálů o volitelné frekvenci pomocí techniky DDS. Dále přístroj disponuje i módem „frequency sweep“, který po nastavený čas plynule mění generovanou frekvenci. Přístroj dokáže generovat kromě vestavěných základních signálů (sinus, trojúhelník a pila) i uživatelem definované a nahrané signály. Přístroj lze ovládat pomocí PC aplikace DataPlotter, se kterou komunikuje přes sběrnici USB. Přístroj je implementován na mikrokontrolerech STM32G431 a STM32F303 s důrazem na minimální počet součástek a možnost jednoduše zařízení zapojit na nepájivém poli.

Klíčová slova: Funkční generátor, arbitrary function, STM32, SDI, frequency sweep, DataPlotter

Vedoucí: doc. Ing. Fischer Jan CSc.

Abstract

This thesis covers the design of a two-channel arbitrary function generator based on STM32 microcontrollers. The purpose of this device is to be used during lab classes at ČVUT FEL to aid in teaching about frequency characteristics of circuits. The basic mode of operation of this device is based on DDS technique, allowing generation of signals with configurable frequency. In addition to that, the device supports frequency sweep mode, which varies the generated frequency seamlessly during a given time period. The user can choose the shape, frequency, amplitude and voltage offset of the generated signal. Three basic signal shapes are built-in (sine, triangle, sawtooth) but the user may choose to upload an arbitrary function. The device is controlled from PC application DataPlotter via USB. Hardware of this device is based on STM32G431 and STM32F303 microcontrollers with emphasis on circuit simplicity, low part count and breadboard-ability.

Keywords: Arbitrary function generator, STM32, SDI, frequency sweep, DataPlotter

Title translation: Arbitrary Function Generator Based on STM32 Microcontroller

Obsah

Obrázky

Tabulky

Seznam použitých zkratek a symbolů

API Application Programming Interface.

ASCII American Standard Code for Information Interchange.

CCM Core Coupled Memory.

CDC Communications Device Class.

COM Communication Port.

CORDIC Coordinate Rotation Digital Computer.

CSV Comma-Separated Values.

DA Digitálně Analogový (převodník).

DAC Digital Analog Converter.

DDS Direct Digital Synthesis.

DMA Direct Memory Access.

FFT Fast Fourier Transform.

FMAC Filter Math Accelerator.

FPU Floating Point Unit.

GUI Graphical User Interface.

HAL Hardware Abstraction Layer.

IDE Integrated Development Environment.

LED Light Emitting Diode.

LUT Look-Up Table.

PLL Phase Locked Loop.

PWM Pulse Width Modulation.

QML Qt Modeling Language.

RAM Random Access Memory.

RF Radio Frequency.

RMS Root Mean Square.

RTOS Real-Time Operating System.

SDI Software Defined Instrument.

SFDR Spurious Free Dynamic Range.

SMD Surface Mount Device.

SNR Signal to Noise Ratio.

SQR Signal to Quantization noise Ratio.

SWD Serial Wire Debugging.

SWO Single Wire Output.

SWV Swrial Wire Viewer.

TUI Text-based User Interface.

UART Universal Asynchronous Receiver and Transmitter.

USART Universal Synchronous/Asynchronous Receiver and Transmitter.

USB Universal Serial Bus.

VCP Virtual COM Port.

Kapitola 1

Úvod

Signálový generátor je zařízení, které v elektrotechnické laboratoři nachází mnoho využití. Například v oblastech generování nosných signálů, měření zkreslení a měřením frekvenčních charakteristik obvodů nekonče. Ve školních laboratořích je signálový generátor využíván zvláště při výuce filtrů, frekvenčních charakteristik obvodů a frekvenční analýzy.

Signálové generátory běžně využívané ve školních laboratořích jsou svými parametry pro potřeby výuky naprosto dostačující přístroje. Jejich parametry často i přesahují požadavky, které jsou na ně kladeny, například co se týče frekvenčního rozsahu nebo zkreslení výstupního signálu. Jinými vlastnostmi jsou však pro výuku méně než ideální. Většinou jde o relativně drahé přístroje, což vzhledem k tomu, že jich v laboratoři musí být několik, není zanedbatelné. Zároveň to znamená, že s takovýmto přístrojem se student pravděpodobně setká pouze ve školní laboratoři a nebude si moci dovolit vlastní na osobní experimenty mimo výuku. Další neideální vlastností laboratorních signálových generátorů je neintuitivnost ovládání způsobená omezenými ovládacími prvky.

Tyto myšlenky vedly k nápadu navrhnout vlastní jednoduchý signálový generátor uzpůsobený požadavkům výuky, který je levný a zároveň hardwarem i ovládáním přístupný studentům. Základním požadavkem na tento přístroj je nahradit standardní laboratorní signálový generátor ve funkci generování harmonických signálů o frekvencích do desítek kHz. Navíc je oproti běžným generátorům požadováno, aby dokázal generovat signály na dvou nezávislých kanálech. Stejně jako laboratorní signálový generátor by i tento přístroj měl být schopen generovat volitelné tvary periodických signálů včetně možnosti nahrát libovolný „arbitrary“ tvar. Další funkcí, kterou by měl tento nový přístroj od signálového generátoru přebírat, je režim „frequency sweep“, který lze s výhodou využít pro měření frekvenčních charakteristik obvodů.

Jednoduchosti a dostupnosti je dosaženo realizací přístroje jako tzv. Software Defined Instrument. Samotný hardware takového přístroje je univerzální a lze tak pouhou změnou firmwaru měnit realizovaný přístroj. Obvod tohoto přístroje lze jednoduše sestavit na nepájivém poli a hardware nutný pro sestavení se skládá pouze z jednoho mikrokontroléru a nejnutnějších podpůrných komponentů jako napěťový regulátor, kondenzátory a konektor. Alternativně lze tento přístroj realizovat i pomocí samotné vývojové desky s mikrokontrolérem. Vyvinutý firmware definující tento přístroj je dostupný pro mikrokontrolér STM32G431KB a vývojovou desku Nucleo-F303RE s mikrokontrolérem STM32F303RE.

Kapitola 2

Rozbor problematiky

2.1 Signálový generátor

Signálový generátor nebo také funkční generátor je jeden z přístrojů, které lze běžně nalézt v každé vybavené laboratoři pracující s analogovými obvody. Jde o přístroj, jehož účelem je generovat signály o nastavitelné frekvenci, amplitudě a v neposlední řadě též o nastavitelném tvaru.

2.1.1 Parametry

Od signálového generátoru se očekává, že ačkoliv jde ve většině případů o digitální přístroje, budou generovat co nejvěrnější rekonstrukci zadaného signálu. Ať už jde o čistý harmonický signál nebo o arbitrary funkci, parametry přístroje se projeví na kvalitě signálu, jeho spektrálním složení a také na obecné vhodnosti pro použití v dané oblasti elektroniky. Jednotlivé vlastnosti generátorů založených na DDS algoritmu a jejich vliv na kvalitu signálu jsou detailněji rozebrány v následujících kapitolách. Zde je tedy tato problematika popsána pouze zkráceně.

Prvním parametrem, který lze pravděpodobně o signálovém generátoru zjistit, je rozsah frekvencí, které dokáže generovat. Tento parametr předurčuje přístroj k využití v různých odvětvích elektroniky. Například generátor s maximální frekvencí 100 MHz bude stačit pro testování běžnějších analogových filtrů, pro testování RF obvodů však existují specializované generátory s rozsahem i do desítek GHz [4].

Maximální frekvence generovaného signálu je většinou dána pouze maximálním povoleným zkreslením generovaného signálu. To se vlivem způsobu rekonstrukce zvyšuje s frekvencí signálu. Hlavní parametry, které ovlivňují frekvenční rozsah jsou vzorkovací frekvence, rozlišení výstupního DA převodníku a kvalita výstupního filtru. Na rozlišení DA převodníku závisí množství kvantizačního šumu, který vzniká. Běžně se rozlišení generátorů pohybuje mezi 12 a 16 bity. Tato informace však u profesionálních přístrojů nebývá uváděna, výrobci uvádějí pouze výsledné parametry signálu, jako harmonické a neharmonické rušení, fázový šum či výkon signálu.

Zkreslení vzniklé kvantizací pomáhá snižovat tzv. oversampling. Čím vyšší je vzorkovací frekvence, tím je při stejné generované frekvenci vyšší poměr převzorkování, a to snižuje zkreslení způsobené kvantizací. V laboratorních generátorech se lze setkat se vzorkovacími frekvencemi od stovek MHz až do GHz. Kvantizační šum i další artefakty vzniklé rekonstrukcí signálu pomáhá odstranit

výstupní rekonstrukční filtr. Tyto filtry bývají relativně komplexní a vysokých řádů. Díky nim však lze kvalitně rekonstruovat signály i z několika málo vzorků na jednu periodu.

2.1.2 Ovládání

Signálové generátory jsou vyráběny pro použití v laboratořích společně s dalšími laboratorními přístroji. Sdílí s nimi tedy fyzickou konstrukci a základní přístup k ovládání a nastavování parametrů. Většinou jde o typické krabicové stolní přístroje s ovládáním a výstupy na čelní straně. Uživatelské rozhraní se u starších typů skládá z alfanumerického displeje, kolečka a tlačítkového číselníku pro nastavování hodnot a několika dalších tlačítek například pro nastavování jednotek nebo režimu přístroje. Moderní přístroje využívají grafických a někdy i dotykových displejů pro zjednodušení ovládání. Obrázek 2.1 ukazuje běžný moderní laboratorní signálový generátor.

Obr. 2.1: Signálový generátor OWON AG051 (převzato z [5])

2.1.3 Využití signálového generátoru

Signálový generátor lze využít například pro hledání rezonančních frekvencí, měření frekvenčních charakteristik či jako laditelný zdroj signálu pro senzory založené na změně impedance. Dále nalézá uplatnění například v komunikacích, kde ho lze využít jako laditelný zdroj frekvence pro PLL, jako mezifrekvenční oscilátor nebo pro přímé generování modulovaného signálu. [7]

2.2 Sofware Defined Instruments

Všechny laboratorní přístroje jsou postaveny na hardwaru, který je specializovaný na výkon funkce daného přístroje. Jen díky tomu mohou efektivně dosahovat výborných parametrů. Znamená to ovšem také, že pokud nejsou masově vyráběné, budou pravděpodobně relativně drahé a také pouze jednoúčelové. Jsou tedy typicky kvalitní, s dobrými parametry, ale drahé a specializované na jeden

účel. V laboratorní výuce mají signálové generátory jistě své místo a přístroj navržený v této práci je nikdy nemůže zcela nahradit. Nároky na laboratorní přístroje a očekávání od nich jsou však pro laboratorní výuku jiné, než pro profesionální laboratorní měření. Minimálně pro úvodní předměty studia elektrotechniky a pro seznámení studenty se základními principy měření. Pro demonstraci základních obvodů a fenoménů v elektronice, jako například měření proudu diodou, základní zapojení operačních zesilovačů, RC filtrů nebo Fourierova transformace, není potřeba pracovat s nejsofistikovanějšími přístroji.

Namísto toho může být pro výuku užitečnější, kdyby každý student měl k dispozici vlastní přístroj, který by si navíc mohl například odnést domů na vlastní experimenty nebo si dokonce svůj vlastní přístroj postavit. Moderní mikrokontroléry se zlepšují v několika směrech. Daří se v nich integrovat vyšší výpočetní výkony, mohou mít nižší spotřebu a v neposlední řadě se také vyvíjejí jejich periferie. Zvláště díky sofistikovanějším periferiím, možnostem jejich propojení a vzájemné spolupráce se i relativně běžné mikrokontroléry stávají schopnými zařízeními v oblastech zpracování signálu, komunikace či řízení. Již před nějakou dobou tak mikrokontroléry dosáhly dostatečné úrovně, aby zastaly základní funkce některých laboratorních přístrojů. V kombinaci s dostupností a cenou (před obdobím covidových odstávek továren a boomem těžení kryptoměn) se tak mikrokontroléry staly ideální platformou pro implementaci zjednodušených verzí laboratorních přístrojů.

Většinu potřebného hardwaru potřebnou k realizaci některého z přístrojů lze najít již integrovanou v mikrokontroléru, ale kýžený přístroj se z něj stává až nahráním programu, který jednotlivé periferie využije a zprostředkuje měření uživateli. Mimoto se jeden mikrokontrolér může stát jedním z více přístrojů, záleží na nahraném programu. Proto se takovéto přístroje nazývají SDI – Software Defined Instruments.

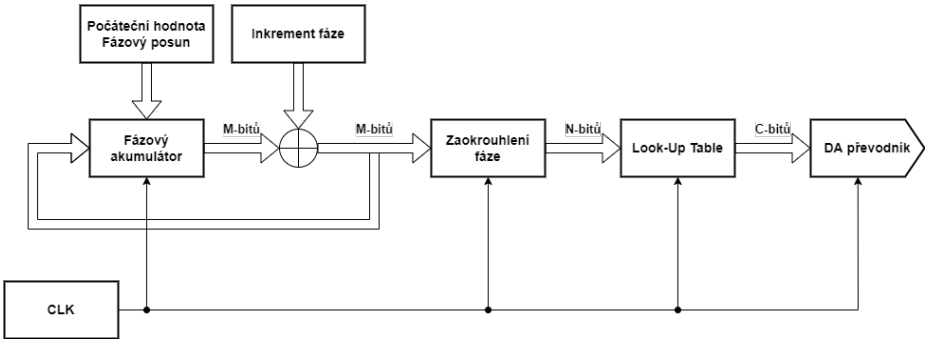
Na Katedře měření Fakulty elektrotechnické na ČVUT bylo takovýchto softwarově definovaných přístrojů vyvinuto již několik. V rámci bakalářských a diplomových prací byly publikovány osciloskopy, voltmetry, čítače nebo například logický analyzátor. Tato práce se k nim přidává, a dokonce staví na jedné z předchozích [1].

Adaptaci tohoto nápadu urychlila distanční výuka v době covidu, kdy možnost vytvořit si doma vlastní přístroj na nepájivém poli umožnila alespoň z části zachovat užitečnost (praktičnost) laboratorních cvičení. I nyní, kdy už probíhá výuka znovu běžným, kontaktním způsobem, nachází SDI využití při laboratorní výuce a při práci studentů na individuálních projektech, kdy mají tyto přístroje dostupné pro práci ve škole i doma.

2.3 DDS algoritmus

Možností, jak generovat periodické signály, je více, ale díky vývoji digitální techniky se algoritmus Direct Digital Synthesis stal jedním z nejpoužívanějších pro signálové generátory. Jeho výhodou je jednoduchost implementace číslicovými obvody, preciznost nastavení fáze i frekvence a možnost generovat jakýkoliv tvar signálu. V této kapitole je vysvětlen princip DDS algoritmu a funkce jeho jednotlivých bloků.

Základní bloky tohoto algoritmu tvoří čítač, fázový akumulátor a tabulka s hodnotami průběhu signálu (Look-Up Table - LUT) [6] [7] [8] [9]. Jejich propojení je znázorněno v blokovém schématu na obrázku 2.2.

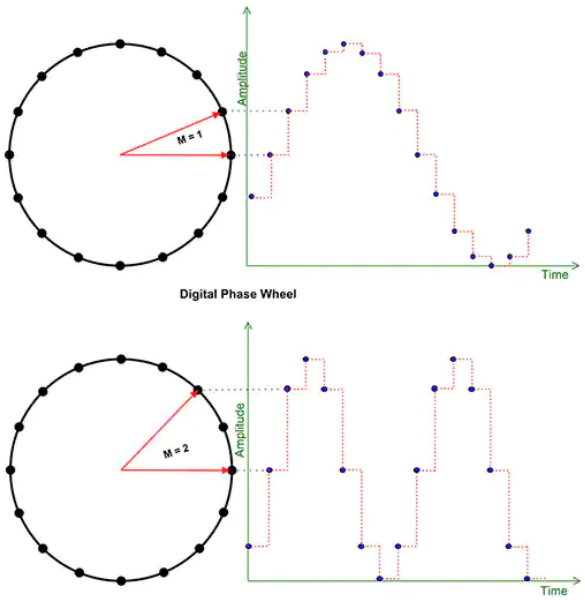


Obr. 2.2: Blokové schéma DDS algoritmu

2.3.1 Fázový akumulátor

V jádru DDS je fázový akumulátor. Jde o registr, ve kterém je uložena hodnota aktuální fáze signálu. S každým hodinovým pulzem je k této hodnotě přičtena hodnota inkrementu fáze a nová hodnota je pak uložena zpět do tohoto registru pro použití v příštím kroku. Výstupem fázového akumulátoru je tak rostoucí řada hodnot, jejíž rychlost růstu je určena velikostí přičítaného inkrementu fáze. Protože kapacita fázového akumulátoru je omezená, udává tato rychlost růstu i dobu, za kterou hodnoty fáze projdou celý rozsah fázového akumulátoru a vrátí se zpět na začátek, stejně jako je tomu u fáze periodického signálu. Inkrement fáze tedy přímo určuje délku periody a jí odpovídající frekvenci generovaného signálu. Tento koncept dobře znázorňuje obrázek 2.3. Frekvenci generovaného signálu lze vypočítat dle rovnice 2.1, kde i značí inkrement fáze, N je kapacita fázového akumulátoru a f_S je hodinová frekvence algoritmu.

$$f = \frac{i}{N} f_S \tag{2.1}$$



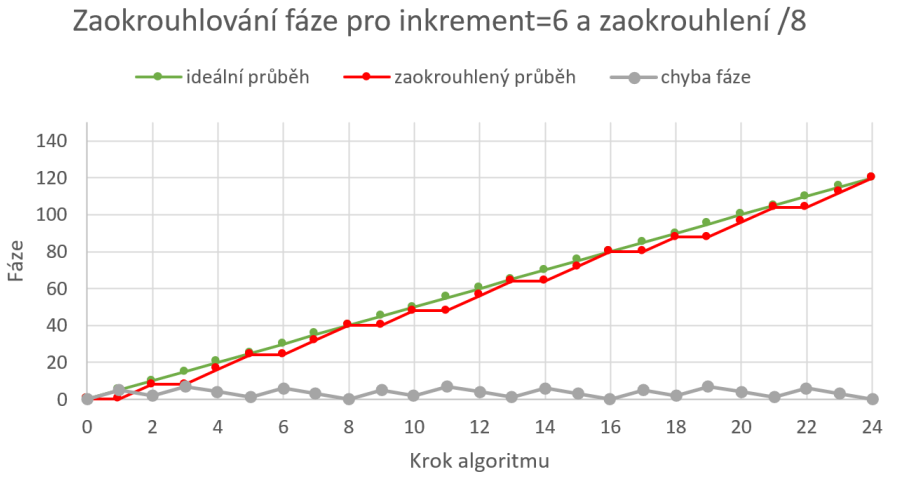
Obr. 2.3: Znázornění fázového akumulátoru jako periody periodického signálu (převzato z [8])

2.3.2 Zaokrouhlování fáze

Registr a sčítačka je hardwarově velmi jednoduchý obvod a není tedy problém pracovat s více než 32bitovým rozlišením. To by znamenalo fázové rozlišení lepší než $1/2^{32} \doteq 0,00000000023283 f_S$. V praxi tomu tak ale není, zvláště proto, že by to znamenalo znát pro každou z 2^{32} hodnot fáze odpovídající hodnotu amplitudy signálu. S tím spojené nároky na paměť jsou ve většině případů nesplnitelné. Proto je do algoritmu zařazeno zaokrouhlování fáze, které zachovává rozlišení fázového akumulátoru, ale pro použití v převodu fáze-amplituda je použita jen zaokrouhlená hodnota (přesněji celá část po vydělení).

2.3.3 Jitter-free frekvence

Nevýhoda této optimizace je, že kvůli zaokrouhlování se výstupní fáze signálu „chvěje“. To znamená, že fáze se nezvyšuje dokonale lineárně, ale rozdíly mezi jednotlivými kroky mohou být různě velké. Toto se anglicky nazývá „phase jitter“. Tento jev má vliv na kvalitu spektra generovaného signálu. Obrázek 2.4 znázorňuje jednoduchý příklad vlivu zaokrouhlování na průběh fáze. Zkreslení způsobené zaokrouhlením fáze je však pro tento přístroj zanedbatelné vůči harmonickému zkreslení kvantizací signálu. Existují ale frekvence, pro které vychází inkrement fáze takový, že se zaokrouhlení neprojeví. Inkrement v takovém případě musí být celočíselným násobkem nejnižší možné hodnoty fáze Tyto frekvence se nazývají „jitter-free“.



Obr. 2.4: Vliv zaokrouhlování na průběh fáze

2.3.4 Převod fáze-amplituda

Pokud si představíme rovnici periodického signálu, například sinusového, vidíme, že fázový akumulátor je v podstatě argument periodické funkce. Pro generování signálu tedy zbývá ještě převést fázi na odpovídající amplitudu.

$$y[k] = \sin(\omega k + \varphi_0) \tag{2.2}$$

Ve výše zmíněné rovnici 2.2 provádí převod z fáze na amplitudu funkce sinus. V DDS algoritmu jsou pro tento převod dvě možnosti. Jednou možností je v každém kroku hodnotu amplitudy vypočítat podle předpisu funkce nebo její aproximace. Tím lze získat v podstatě libovolnou přesnost výsledku, ale za cenu vyššího výpočetní náročnosti. Druhou metodou je použití paměti jako tzv. „Look-Up Table“. V této paměti jsou uloženy hodnoty amplitudy jedné periody požadovaného signálu. Rozsah adres paměti a rozsah fázového akumulátoru po zaokrouhlení jsou si rovny a oba tyto rozsahy symbolizují jednu periodu signálu. Přečtením hodnoty na adrese získané zaokrouhlením fázového akumulátoru tedy získáme amplitudu signálu. Ve většině případů je pro kvalitu signálu důležitější vyšší obnovovací frekvence algoritmu než amplitudové rozlišení (které je navíc omezeno výstupním DA převodníkem), a proto je většinou použita druhá metoda.

2.3.5 Spektrum DDS signálu

Signál generovaný DDS algoritmem je pouze digitální rekonstrukcí požadovaného signálu. Z důvodu odchylek generovaného signálu od požadovaného vznikají chyby, které se projeví jak v časové, tak zejména ve frekvenční oblasti. Hlavními zdroji chyb je aliasing, kvantizace, zaokrouhlování fáze a nelinearity DA převodníku.

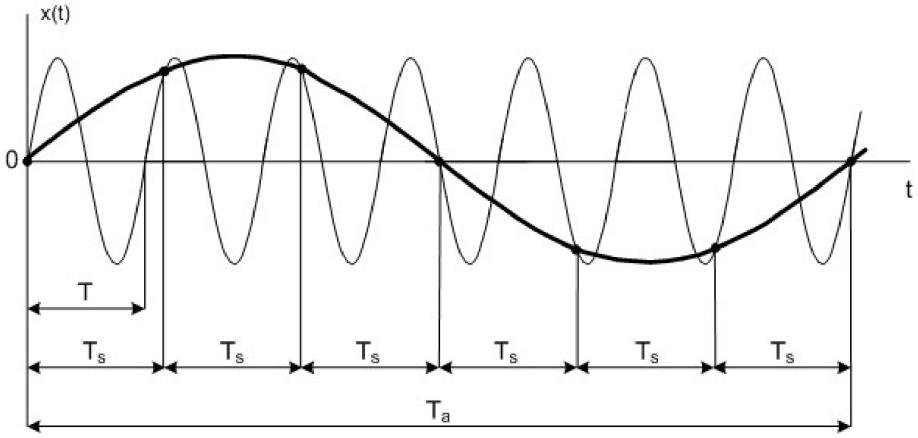
Aliasing vzorkovaného signálu

Aliasing je chyba společná všem systémům pracujícím se vzorkovanými signály. Vzorky signálu jsou odebírány či generovány v pravidelných časových intervalech. Na vznik aliasingu se lze dívat tak, že o průběhu signálu mezi těmito vzorky nemáme žádné informace a stejnými vzorky by mohl například procházet i signál o vyšší frekvenci. Obrázek 2.5 toto dobře ilustruje. Z tohoto důvodu vznikají ve spektru vzorkovaného signálu tzv. obrazy signálu. Tyto obrazy se objevují na frekvencích splňujících následující vztah 2.3.

$$f_i = kf_S \mp f_{sig} \; ; k \in Z \tag{2.3}$$

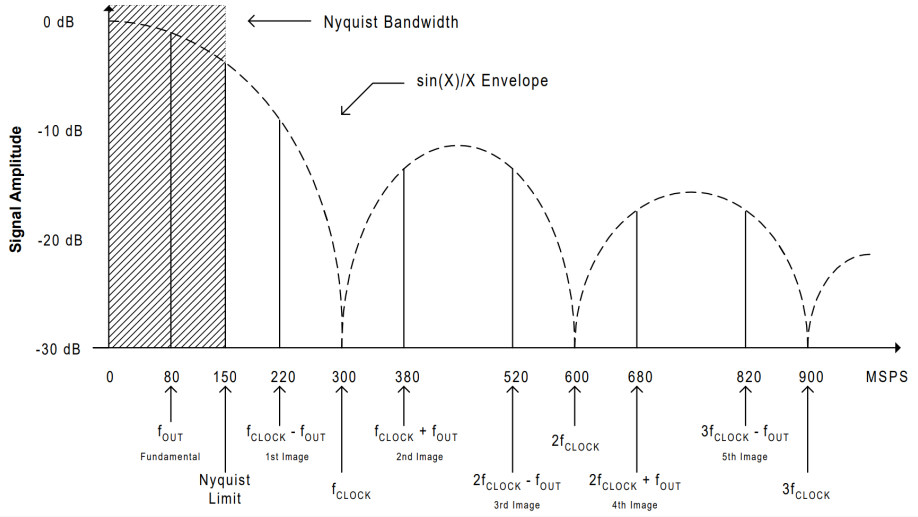
Ve spektru aliasing vypadá, jako by se obrazy signálu zrcadlily okolo vzorkovací frekvence f_S ve vzdálenosti f_{sig}. Dle [9] lze dokázat, že amplituda obrazů signálu sleduje následující funkci 2.4.

$$A = 20 \log_{10} \left(\left| \frac{sin\left(\pi \frac{f}{f_S}\right)}{\pi \frac{f}{f_S}} \right| \right) \tag{2.4}$$



Obr. 2.5: Vznik aliasingu v časové doméně (převzato z [10])

Ze vztahu 2.4 i z obrázku 2.6 vychází, že čím vyšší je poměr f_S/f , tím blíže jsou obrazy k bodům největšího útlumu okolo násobků f_S. Toto potvrzuje intuitivní myšlenku, že čím vyšší vzorkovací frekvenci použijeme, tím přesnější reprodukci signálu získáme.



Obr. 2.6: Aliasing generovaného signálu ve spektru (převzato z [6])

Kvantizační šum

Na výstupu DDS algoritmu je DA převodník, který má určené maximální rozlišení, typicky počtem bitů. Zatímco požadovaný signál může teoreticky nabývat jakýchkoli hodnot, výstup DA převodníku je omezen na 2^n konkrétních hodnot, kde n je počet bitů převodníku. Tím pádem nemůže výstup převodníku růst nebo klesat plynule, ale pouze po skocích. Tyto skoky mezi konkrétními hodnotami mají za následek přítomnost vyšších harmonických složek ve spektru. Tomuto jevu se říká kvantizační šum. Lze prokázat, že celková energie kvantizačního šumu je závislá pouze na rozlišení převodníku a lze vypočítat dle následující rovnice 2.5, kde q je kvantizační krok převodníku a n je počet bitů převodníku [11].

$$E_q = \frac{q}{\sqrt{12}} \; ; \; q = \frac{1}{2^n} \tag{2.5}$$

Pro čistě harmonický signál o amplitudě plně využívající rozsah převodníku lze vyjádřit poměr signálu ke kvantizačnímu šumu následovně.

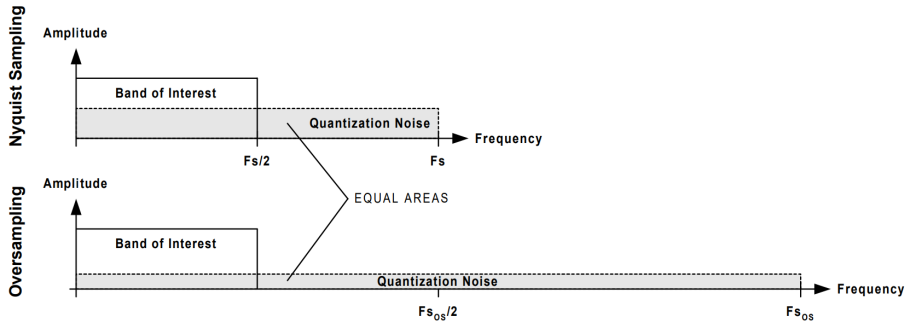
$$SQR = \; 20\log_{10}\left(\frac{E_{sig}}{E_q}\right) = 20\log_{10}\left(\frac{\frac{q2^n}{2\sqrt{2}}}{\frac{q}{\sqrt{12}}}\right) \; [dBc] \tag{2.6}$$

$$SQR = 1,76 + 6,02n \; [dBc] \tag{2.7}$$

Vypočítaná hodnota výkonu kvantizačního šumu je integrována přes celé spektrum od 0 po Nyquistovu frekvenci $f_S/2$. Pokud použijeme vzorkovací frekvenci vyšší, než jakou diktuje Nyquistův teorém, bude splňovat podmínku $f_S > 2f_{sig}$. Ta samá energie kvantizačního šumu tedy bude rozprostřena na širším spektru. Výkon kvantizačního šumu tak ve stejném frekvenčním rozsahu musí být nižší. Rovnice pro odstup signálu od kvantizačního šumu se tak změní na:

$$SQR = 1,76 + 6,02n + 10\log_{10}\left(\frac{f_S}{2f_{sig}}\right) \; [dBc] \tag{2.8}$$

Například zvýšením vzorkovací frekvence 100x tak lze zlepšit SQR o 20 dB. Obrázek 2.7 ilustruje snížení výkonu kvantovacího šumu v daném frekvenčním pásmu díky rozprostření energie kvantovacího šumu na širší spektrum.



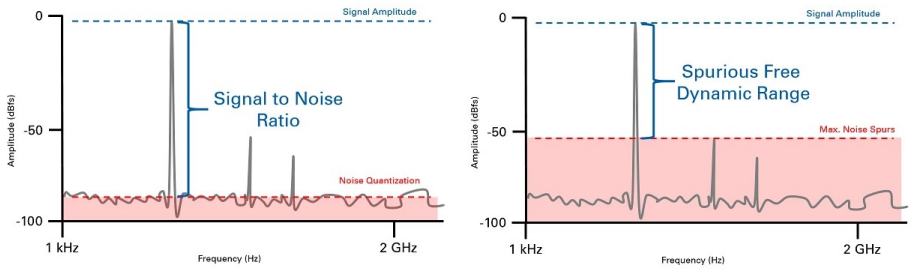
Obr. 2.7: Znázornění snížení výkonu kvantovacího šumu v daném frekvenčním pásmu pomocí oversamplingu (převzato z [6])

Výše psané vztahy platí pouze pro sinový signál o amplitudě odpovídající plnému rozsahu DA převodníku. Pokud generovaný signál nevyužije celý rozsah, snižuje se kvůli nižšímu využitému rozlišení odstup signálu od šumu následovně [6]. $A_{sig\ p-p}$ značí peak-to-peak rozsah signálu a FS značí maximální rozsah převodníku.

$$SQR = 1,76 + 6,02n + 10\log_{10}\left(\frac{f_S}{2f_{sig}}\right) - 20\log_{10}\left(\frac{A_{sig\ p-p}}{FS}\right) \; [dBc] \tag{2.9}$$

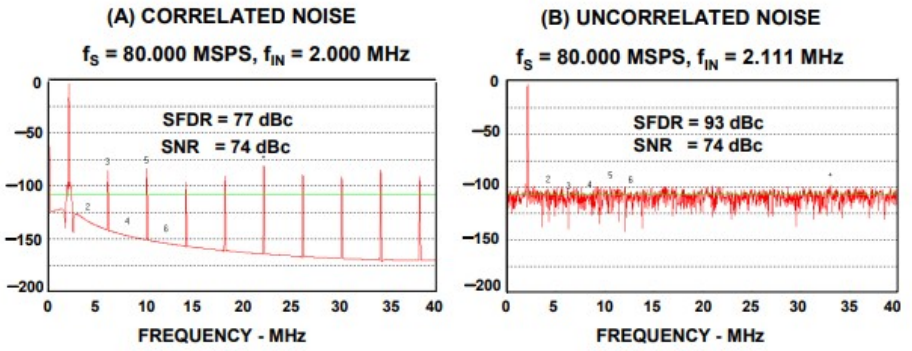
Tyto úvahy jsou založeny na RMS hodnotě výkonu kvantizačního šumu. Pro kompletní přehled o nežádoucích složkách ve spektru však tyto výpočty nemusí stačit. Je potřeba rozlišovat mezi ukazateli SNR a SFDR (Spurious Free Dynamic Range), které se mohou velmi lišit, pokud není rozložení šumu rovnoměrné. Význam SNR a SFDR a jejich rozdíl je znázorněn na následujícím obrázku 2.8.

$$SNR = \frac{P_{sig}}{P_n}\ ; SFDR = \frac{P_{sig}}{P_{spur}} \tag{2.10}$$



Obr. 2.8: Srovnání významu SNR a SFDR (převzato z [13]

Pokud je kvantizační šum nekorelovaný s užitečným signálem, bude mít spektrální rozložení bílého šumu. V takovém případě by SNR stačilo pro zhodnocení kvality signálu. Může se však stát, že kvantizace je korelovaná s generovaným signálem a v takovém případě se většina energie kvantovacího šumu soustředí do špiček na násobcích fundamentální frekvence. Tento fenomén ilustruje obrázek 2.9, kde v prvním případě koreluje vzorkovací frekvence s frekvencí generovaného signálu, zatímco ve druhém případě je generovaná frekvence lehce posunuta, a tudíž je korelace slabá.



Obr. 2.9: Spektrum korelovaného a nekorelovaného kvantizačního šumu (převzato z [11])

Zaokrouhlení fáze

Vliv zaokrouhlení fáze v časové oblasti již byl ukázán na obrázku 2.4, ve spektru se však projeví také. Chyba fáze způsobená zaokrouhlením je prokazatelně periodická a ve frekvenční oblasti se tedy objeví jako úzké spektrální čáry. Jejich amplituda a pozice je závislá na počtu zaokrouhlených bitů a na inkrementu fáze.

Nejvyšší amplitudy dosahuje první vzniklá spektrální čára, pokud je splněna následující podmínka 2.11, kde i je inkrement fáze a m značí počet zaokrouhlených bitů [6].

$$NSD\,(i,\ 2^m) = 2^{m-1} \tag{2.11}$$

Tato podmínka je splněna, pokud hodnota inkrementu je polovina zaokrouhlované hodnoty. Při takovémto nastavení je amplituda spektrální čáry maximální a to o 6,02 m dB menší než generovaný signál, jak je naznačeno rovnicí 2.12 [6].

$$20\log_{10}\left(\frac{P_{sig}}{P_z}\right) = 6,02m \tag{2.12}$$

Naproti tomu existují „jitter-free“ frekvence, které jsou popsány následující podmínkou 2.13 [6]. Při těchto frekvencích (v kombinaci s fázovým akumulátorem a zaokrouhlením) nevzniká žádné zkreslení zaokrouhlením, protože jsou to násobky zaokrouhlované hodnoty a zaokrouhlovaná část je tak vždy nulová.

$$NSD\,(i, 2^m) = 2^m \tag{2.13}$$

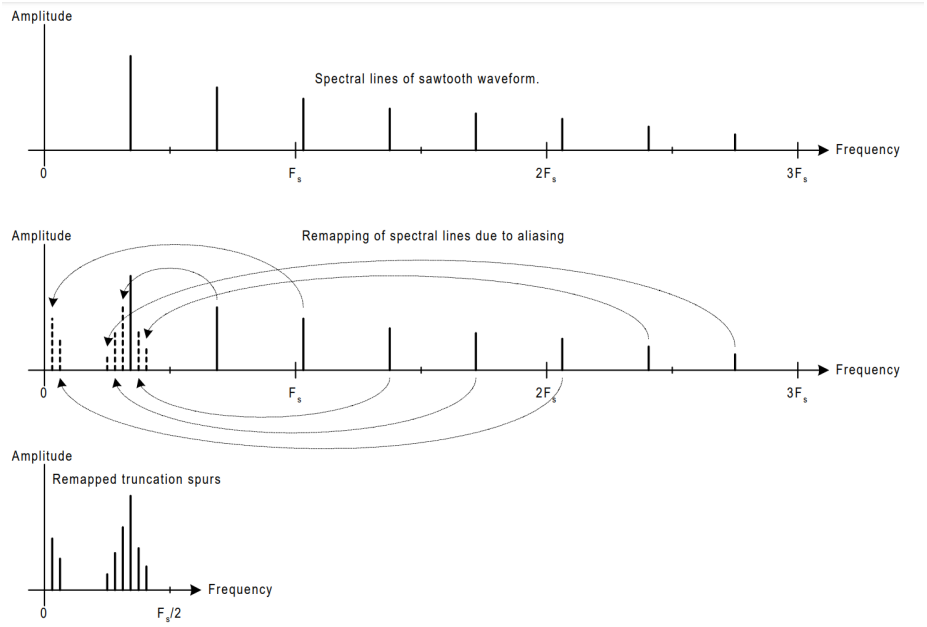
Kromě „jitter-free“ frekvencí popsaných rovnicí 2.13 lze za „jitter-free“ považovat též ty frekvence, pro které sice dochází k zaokrouhlování fáze, avšak počet vzorků na periodu vyjde celočíselně a každou periodu se tedy opakují ty samé vzorky. Dochází sice k zaokrouhlování fáze, ale nedochází k fázovému „chvění“. Definici „jitter-free“ frekvencí pak lze rozšířit na následující rovnici 2.14, kde n je počet bitů fázového akumulátoru.

$$f_{jitter\_free} = \frac{f_S}{2^x}; x \in\ N^+; 1 \leq x \leq n \tag{2.14}$$

Výpočet pozice jednotlivých spektrálních čar způsobených zaokrouhlováním fáze není zcela triviální a pro vyšší rozlišení fázových akumulátorů může vyžadovat speciální přístupy kvůli výpočtům s velmi vysokými čísly. Lze ale s jistotou říct, že rozmístění těchto čar je diktováno rozmístěním harmonických frekvencí trojúhelníkového signálu. První z těchto harmonických se nachází na frekvenci vypočitatelné dle následujícího vztahu 2.15 [6].

$$f_{z1} = f_S\frac{i\ mod\ 2^m}{2^m} \tag{2.15}$$

Další vzniklé spektrální čáry většinou zasahují i daleko za Nyquistův limit i f_S a zrcadlí se tedy ve spektru zpět do základního pásma, jak naznačuje obrázek 2.10.



Obr. 2.10: Znázornění harmonických vzniklých zaokrouhlením fáze a jejich zrcadlení do základního pásma (převzato z [6])

Chyby DA převodníku

Posledním zde diskutovaným zdrojem zkreslení výstupu DDS algoritmu jsou chyby výstupního DA převodníku. Sem lze zařadit nelinearitu převodu napětí, zákmity při rychlých přechodech nebo také rušení hodinovým signálem. I tyto chyby se projeví jako harmonické složky ve spektru.

2.4 Výběr mikrokontroléru

Již ze zadání bylo jisté, že pro realizaci přístroje bude využito mikrokontroléru. Na výběr je však mnoho druhů mikrokontrolérů s různým zaměřením, různým výkonem a vybavených různými periferiemi. Pro správný výběr je třeba znát předem požadavky vyplývající z požadované aplikace.

Prvním diskriminačním faktorem, který pravděpodobně vyřadí z výběru nejvíce mikrokontrolérů, je přítomnost zabudovaného DA převodníku s dostatečnou maximální vzorkovací frekvencí. Pro generování signálů o frekvenci do desítek kHz je potřeba vzorkovací frekvence v řádu alespoň stovek kHz až jednotek MHz.

Druhým faktorem je výpočetní výkon, který musí být dostatečný, aby dokázal při dané vzorkovací frekvenci zásobovat DA převodník novými hodnotami k převodu. S ohledem na očekávanou vzorkovací frekvenci a potřebné výpočty nemá velký smysl hledat jiné mikrokontroléry, než ty s 32bitovým jádrem uzpůsobeným pro zpracování signálu a s frekvencí v řádu vyšších desítek až stovek MHz.

Co se týče paměti pro program a data, jsou dnešní mikrokontroléry, které splňují již zmíněné podmínky, většinou vybaveny dostatečně. Problém by mohl nastat snad jen s velikostí RAM u nižších řad mikrokontrolérů vzhledem k tomu, že DDS algoritmus vyžaduje v tomto případě dvě LUT a buffer hodnot a další buffery pro komunikaci s PC. Tyto buffery se pohybují v řádu jednotek kB, a tak je třeba hledat mikrokontrolér s RAM alespoň v řádu několika desítek kB. Již byla zmíněna komunikace s PC, využitá pro ovládání přístroje. Pro připojení k PC by bylo ideální použít mikrokontrolér s integrovanou podporou USB, ale běžně se k tomuto účelu používá i rozhraní UART spolu s převodníkem USB-UART.

V neposlední řadě je důležité myslet i na pouzdro mikrokontroléru. Téměř všechny dnešní mikrokontroléry, a zvláště ty splňující vznesené podmínky, se vyrábějí výhradně v SMD pouzdrech, která nelze přímo zapojit do nepájivého pole. Proto je třeba počítat s tím, že mikrokontrolér bude někdo muset ručně připájet na tzv. „breakout board“, který vyvede všechny piny na standardní hřebenový konektor. Nepřichází tak v úvahu pouzdra BGA, která se připojují pomocí cínových kuliček zespodu pouzdra, ani pouzdra QFN s pájecími ploškami na hraně pouzdra, ani pouzdra s nožičkami s příliš jemnou roztečí pinů.

Kapitola 3

Rozbor řešení

3.1 Použitý hardware

V době, kdy byla tato práce zadávána, se stala dostupnost mikrokontroléru tak vzácnou vlastností, že dokázala zastínit všechny ostatní parametry a požadavky. Naštěstí mezi těmi, které byly pro tuto práci a následnou výuku v laboratořích dostupné, byly i mikrokontroléry, které splňovaly požadavky na tuto aplikaci a nebylo tak potřeba hledat alternativní způsoby, jak aplikaci implementovat s omezenými prostředky.

Z dostupných mikrokontrolérů byly pro implementaci vybrány dva – STM32G431KBT6 a STM32F303RET6 na desce Nucleo-F303RE. V obou případech jde o mikrokontroléry od firmy ST Microelectronics postavené na jádře Arm® Cortex®-M4, doplněném o analogové, časovací a komunikační periferie. Oba mikrokontroléry jsou sice z jiných řad, i přesto je ale většina periferií a práce s nimi stejná.

3.1.1 Společné znaky použitých mikrokontrolérů

Jádro Arm® Cortex®-M4

Oba mikrokontroléry obsahují 32bitové jádro Arm® Cortex®-M4. Pro tuto aplikaci je výhodné, že toto jádro disponuje přidanými instrukcemi pro digitální zpracování signálu, například operace s plovoucí desetinnou čárkou nebo instrukce MAC (Multiply And Accumulate).

Výhodou by měla být přítomnost CCM (Core Coupled Memory), která slibuje zvýšit rychlost programu tím, že z ní lze vykonávat program maximální rychlostí a bez čekacích stavů paměti Flash. Navíc by měla dovolovat načítat data i instrukce zároveň díky vlastnímu připojení k jádru přes BusMatrix [18]. Výhody této paměti však není jednoduché využít, neboť je k tomu vyžadován zásah do souboru linkeru a do startupu mikrokontroléru. Navíc ani po nastudování této problematiky a provedení požadovaných úprav se použitím CCM RAM nepodařilo program jakkoliv zrychlit.

Vestavěné časovače

Snad všechny rodiny mikrokontrolérů STM32 sdílí stejné základní (basic), víceúčelové (general-purpose) a pokročilé (advanced-control) časovače. Rozdíly jsou jen v jejich počtech a případně v implementaci některé pokročilé funkce. Tyto periferie slouží k odměřování přesných časových intervalů, spouštění pravidelných úkonů nebo ke generování PWM. Většina časovačů je 16bitová, nabízí volitelný předdělič a volitelnou periodu. Víceúčelové a pokročilé časovače navíc nabízí funkce generování PWM, snímání délky pulzů, dekódování enkodérů nebo komplexní řetězení více časovačů. Každý časovač také nabízí možnost generovat přerušení programu v reakci například na přetečení periody nebo shodu hodnoty časovače se zadanou hodnotou. V tomto přístroji jsou využity pouze dva časovače. Základní časovač TIM6 pro své propojení s DAC a víceúčelový časovač TIM3 pro pravidelné odesílání zpráv.

Direct Memory Access

Direct Memory Access (DMA) je periferie, která umožňuje přenos dat mezi jinými periferiemi a pamětí nebo mezi různými adresami v paměti. V mikrokontroléru bez DMA musí každý přenos dat, ať už mezi různými proměnnými nebo mezi pamětí a periferií, projít přes procesor. Vzhledem k omezenému výpočetnímu výkonu by to zvláště v případě této aplikace znamenalo, že procesor by většinu času strávil přenosem dat namísto výpočtů a obsluhy uživatelského rozhraní. Proto lepší mikrokontroléry obsahují DMA, které umožní automatizovaný přenos dat na pozadí, zatímco procesor dále vykonává program. V tomto přístroji je DMA využito pro přenos dat mezi bufferem hodnot v RAM a DA převodníkem a také pro ukládání přijatých dat a odesílání přes UART.

DA převodník

Digitálně-analogový převodník je pro tuto aplikaci naprosto nezbytným modulem mikrokontroléru. DA převodník DAC1 ve vybraných dvou mikrokontrolérech dokáže převádět na dvou nezávislých kanálech digitální hodnoty na analogové napětí v rozmezí 0 až VDDA s rozlišením 12 bitů. Obsahuje též integrovaný výstupní buffer (sledovač) pro snížení výstupní impedance, se kterým dokáže měnit napětí na výstupu se vzorkovací frekvencí 1MHz. DA převodníky obou mikrokontrolérů jsou ve většině aspektů stejné, jen rodina STM32G4 nabízí navíc některé funkce, jako převod dat v signed módu nebo Sample-and-Hold výstupní buffer. Mikrokontrolér STM32G431 navíc obsahuje více DAC modulů, které ale nemají výstupní buffer ani výstup vyvedený na pin.

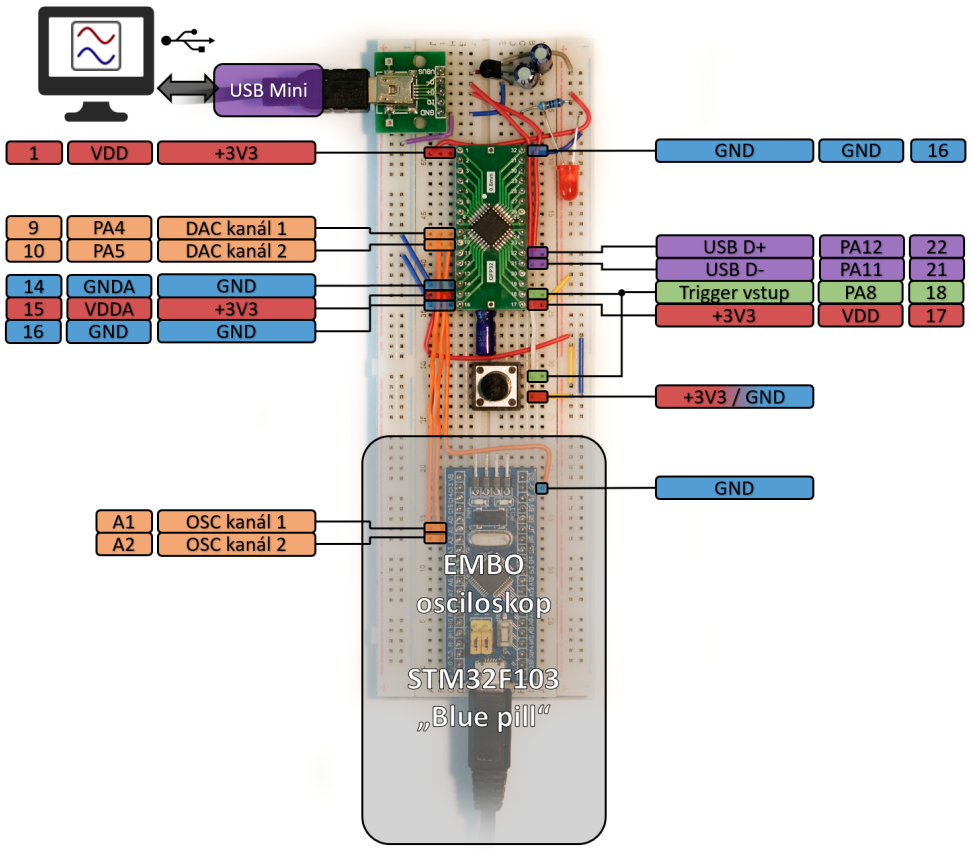
3.1.2 Mikrokontrolér STM32G431KBT6

Mikrokontrolér STM32G431BT6 disponuje 128 kB Flash paměti pro program a konstanty a 32 kB RAM paměti, z čehož 10 kB činí CCM RAM [19] [20]. Maximální frekvence procesoru je 170 MHz a lze ho napájet napětím od 1,71 V do 3,6 V. Tento konkrétní model je vyráběn v SMD pouzdře LQFP32 s 32 piny. Na rozdíl od STM32F303, obsahuje tato novější rodina mikrokontrolérů některé vyspělejší funkce, jako například pokročilejší DMA, instrukční i datovou cache a koprocesory CORDIC a FMAC. Z důvodu kompatibility s STM32F303 však například CORDIC modul nebyl využit, i když by své využití v DDS algoritmu pravděpodobně našel.

Velkou výhodou oproti desce Nucleo s druhým použitým mikrokontrolérem je hardwarová podpora USB ve specifikaci Full Speed. Díky tomu lze připojit mikrokontrolér přímo k PC bez nutnosti USB-UART převodníku.

Naopak nevýhodou použití samotného mikrokontroléru jako základu je nutnost zajistit napájení a konektivitu vlastním zapojením.

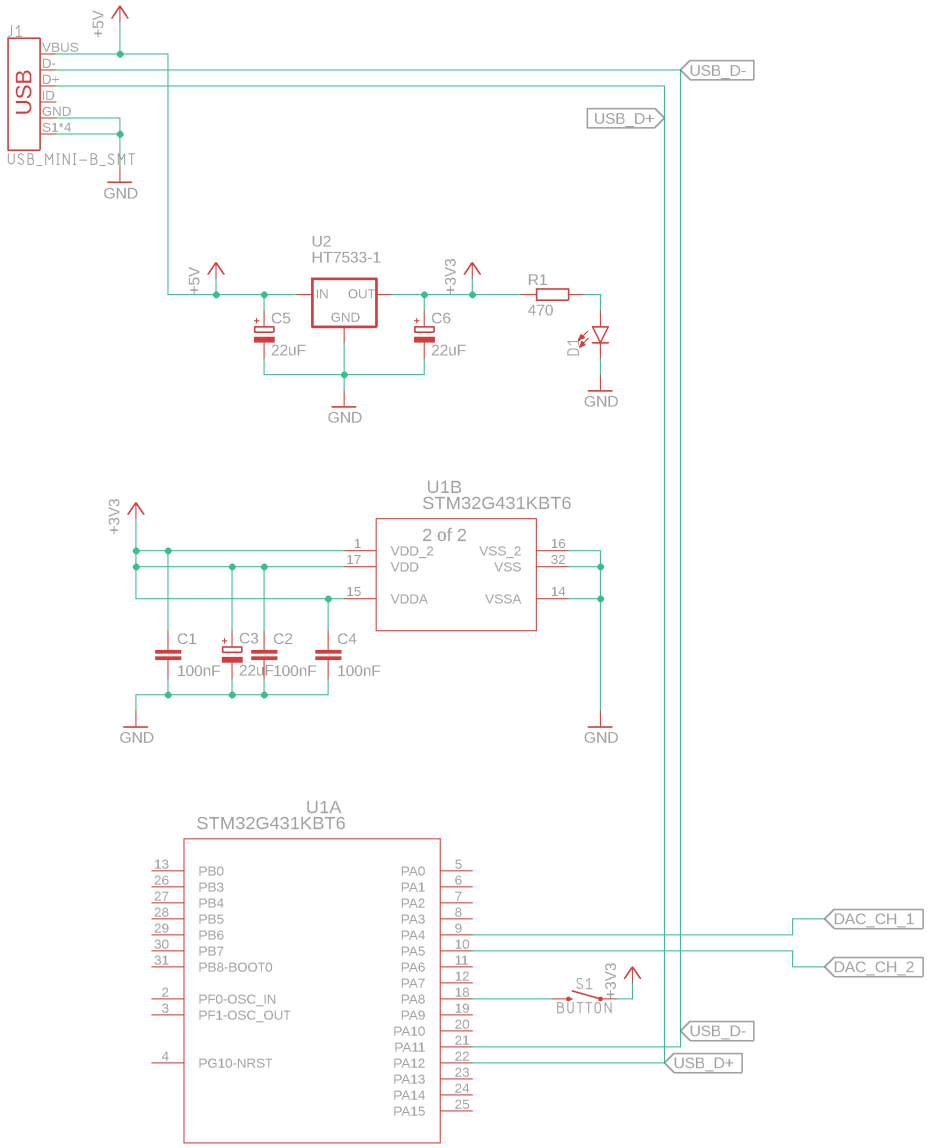
Zapojení s STM32G431KBT6



Obr. 3.1: Zapojení STM23G431 na nepájivém poli a rozmístění použitých pinů

Pro fyzickou realizaci přístroje s mikrokontrolérem STM32G431KBT6 bylo vybráno sestavení obvodu na nepájivém poli. Tento způsob sestavování obvodů je v laboratorních cvičeních často využívaný, a proto je výhodné jej použít i pro tento přístroj. Celý obvod se skládá z 10 komponentů a 14 propojek, což lze sestavit za několik minut. Obrázek 3.1 ukazuje příklad zapojení signálového generátoru s mikrokontrolérem STM32G431KBT6 a význam využitých pinů. Spolu s generátorem je na obrázku 3.1 též SDI osciloskop EMBO [2] založený na vývojové desce „Bluepill“ s mikrokontrolérem STM32F103, který generátor při vývoji dobře doplňoval.

Kromě mikrokontroléru připájeného na „breakout board“ vyžaduje obvod pouze USB konektor, stabilizátor napětí a filtrovací kondenzátory napájení. Pro stabilizaci napětí zde byl použit lineární regulátor HT7533-1, který napájí mikrokontrolér napětím 3,3 V. Volitelně lze přidat tlačítko pro spouštění frekvenčních sweepů. Schéma obvodu přístroje, včetně hodnot komponentů je zobrazeno na 3.2.



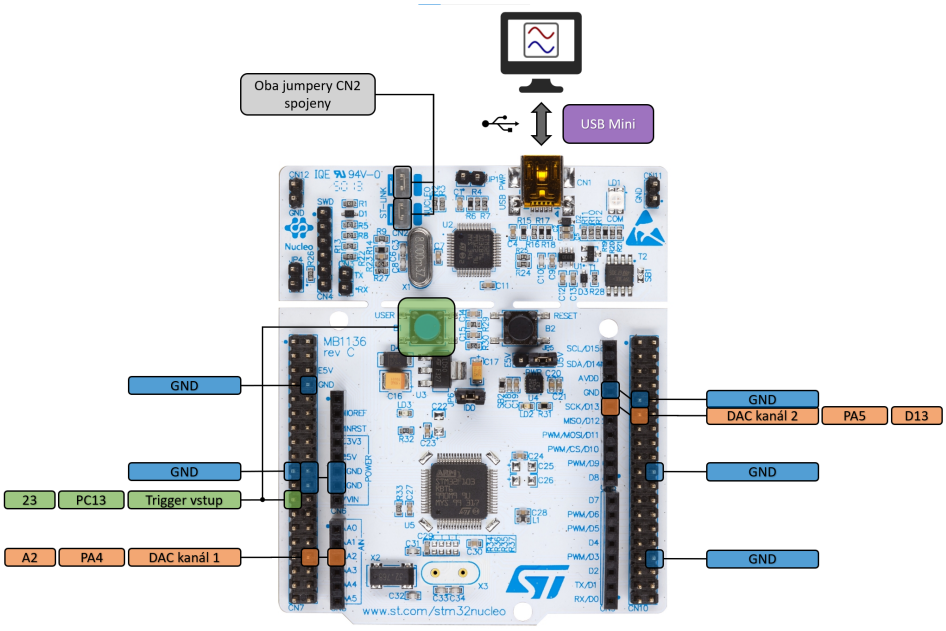
Obr. 3.2: Schéma obvodu přístroje s mikrokontrolérem STM32G431KBT6

3.1.3 Mikrokontrolér STM32F303RET6

Druhý vybraný mikrokontrolér STM32F303RET6 poskytuje více než dostatek paměti - 512 kB Flash paměti a 80 kB RAM, z čehož 16 kB je CCM RAM [21] [22]. Oproti STM32G431 má však méně než poloviční maximální frekvenci jádra. To se zpočátku zdálo jako omezující faktor, ale nakonec dosahuje aplikace požadovaných parametrů na obou mikrokontrolérech. Tento konkrétní model je umístěn v SMD pouzdře LQFP64 s 64 piny a lze jej napájet napětím od 2 V do 3,6 V. Požadavek na vyšší minimální napájecí napětí je pravděpodobně způsoben starší výrobní technologií oproti nové rodině mikrokontrolérů STM32G4.

Zapojení s Nucleo-F303RE

Stejnou funkcionalitu jako s mikrokontrolérem STM32G431 lze jednoduše dosáhnout i na vývojové desce Nucleo-F303RE s mikrokontrolérem STM32F303RET6. Tato deska již obsahuje všechny potřebné podpůrné obvody. Pro použití tedy stačí jen připojit desku přes Mini USB k PC a nahrát firmware. Obrázek 3.3 znázorňuje rozmístění použitých pinů a možnosti připojení společné země. Nutno podotknout, že na této desce je poněkud nešťastně vyveden pin PA5, na kterém je výstup DA převodníku, protože na stejný pin je také připojena LED dioda [23]. Sice tímto uživatel získává zpětnou vazbu o činnosti 2. kanálu, ale teoreticky to může způsobovat zkreslení signálu vlivem vyššího odběru proudu z výstupu DAC.



Obr. 3.3: Rozmístění použitých pinů na desce Nucleo-F303RE

3.2 Implementace algoritmu DDS

Algoritmus DDS neboli Direct Digital Synthesis, který je použit ke generaci signálů v tomto signálovém generátoru je v principu stejný jako ten, který se používá v profesionálních přístrojích. V prodávaných signálových generátorech však tento algoritmus běží na dedikovaném hardwaru, uzpůsobeném přímo pro tuto aplikaci. Tyto generátory tak dokážou pracovat s mnohem vyššími obnovovacími frekvencemi (> stovky MHz) a vyšším počtem i rozlišením vzorků signálu. V neposlední řadě, tyto generátory obsahují výstupní filtry, které filtrují vznikající kvantizační šum, obrazy signálu a vyšší harmonické, a dovolují tak generovat signály vyšší frekvence při stejné obnovovací frekvenci.

V přístroji vytvářeném v rámci této práce však tento algoritmus musí běžet na běžném mikrokontroléru. I když jsou již dnešní mikrokontroléry v této tzv. „mixed signal processing“ oblasti relativně obstojné, stále nemohou dosáhnout výkonu specializovaného hardwaru a je tedy třeba algoritmus přizpůsobit. V této kapitole jsou rozebrány implementace dílčích částí DDS algoritmu a aspekty, které ovlivnily výběr řešení a parametry.

3.2.1 Časování algoritmu

Určení obnovovací frekvence

Jedním z hlavních požadavků na signálový generátor je co nejvyšší obnovovací frekvence výstupního signálu. Maximální frekvence, se kterou dokáže spolehlivě obnovovat svůj výstup použité DAC je 1 MHz. Požadovaná vzorkovací frekvence algoritmu f_S je tedy také 1 MHz. Protože maximální frekvence jádra použitých mikroprocesorů je 170 a 72 MHz, vychází nám, že procesor bude mít průměrně 170 respektive 72 cyklů na výpočet jedné hodnoty pro dva kanály.

Zajištění stability časování

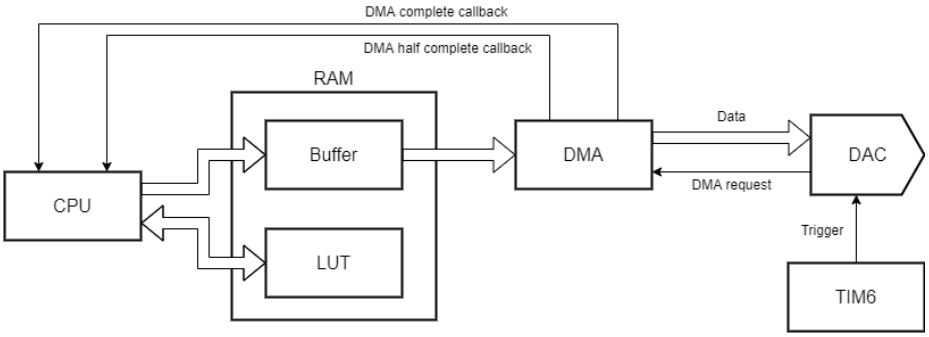
Zároveň s co nejvyšší vzorkovací frekvencí je také třeba dosáhnout co nejvyšší stability této frekvence. Proto je nutné celý algoritmus synchronizovat časovačem odvozeným od stabilního hodinového signálu.

Synchronizace časovačem a přerušením. Jednou z možností, jak celý algoritmus synchronizovat, je časovačem pravidelně spouštět přerušení, ve kterém by se provedl jeden krok algoritmu a nově vypočítané hodnoty by se ihned vystavily na výstup DAC. Tento přístup je však pro požadovanou frekvenci přerušení nevhodný, protože každý vstup do obsluhy přerušení vyžaduje jistou režii. Je třeba zastavit aktuální instrukci a uložit používané registry do zásobníku, což dle [24] trvá 12 instrukčních cyklů. Po přičtení případných čekacích stavů Flash paměti, softwarové režie v obsluze přerušení a návratu z přerušení, dojdeme k závěru, že režie zabírá značnou část času procesoru. Délka těchto prodlev je navíc proměnná například podle toho, zda na právě přerušovaný kód využíval FPU nebo zda je třeba uložit více než 5 registrů. Pro generování časové základny tohoto přístroje se tak nehodí.

Synchronizace časovačem s pomocí DMA. Řešení použité v tomto přístroji zajišťuje stabilitu obnovovací frekvence bez zvýšené režie díky propojení časovače TIM6, modulu DAC1 a DMA.

Časovač TIM6 je nastaven na periodu přetečení odpovídající požadované frekvenci $f_S = 1$ MHz. Při každém přetečení časovač generuje trigger signál, který je přiveden na DAC modul. Ten s každým triggerem ihned spustí konverzi hodnoty, kterou má právě uloženou v registru DAC_DHRx a zároveň s tím vygeneruje žádost o přenesení další hodnoty skrze DMA. DMA pracuje v režimu kruhové inkrementace a přenáší hodnoty z bufferu v RAM do registru DAC_DHRx. Obrázek 3.4 znázorňuje tento přístup na blokovém schématu propojení částí mikrokontroléru.

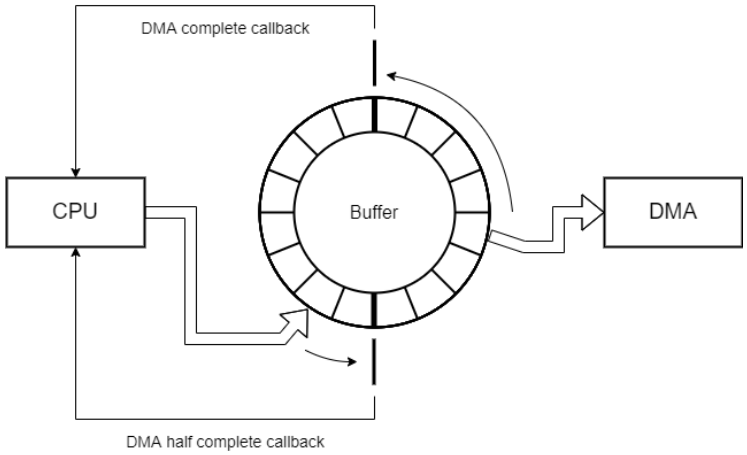
Tento přístup zajišťuje nejlepší možnou pravidelnost obnovování výstupu DAC a zároveň díky použití DMA eliminuje téměř veškerou režii spojenou s obsluhou přerušení a přenosem dat. Přesnost a pravidelnost této frekvence závisí pouze na přesnosti a jitteru použitého oscilátoru.



Obr. 3.4: Blokové schéma propojení částí mikrokontroléru pro generaci signálu

3.2.2 Výpočet hodnot signálů

Díky využití DMA pro přenos dat mezi pamětí a DAC je téměř většina času procesoru k dispozici pro výpočty hodnot signálu. Další optimalizací pro rychlost výpočtu, kterou DMA umožňuje, je počítání hodnot po dávkách a předem, na rozdíl od počítání hodnoty těsně před jejím použitím. Tímto způsobem lze vypočítat dávku hodnot v jednom cyklu, což je efektivnější díky možnosti mít v registrech procesoru uloženo více používaných proměnných bez nutnosti je načítat a ukládat do paměti. Vypočítané hodnoty jsou ukládány do bufferu, ze kterého zároveň DMA vyčítá a přenáší data do DAC. Zatímco DMA vyčítá hodnoty z jedné poloviny bufferu, DDS algoritmus ukládá vypočítané hodnoty do druhé poloviny bufferu a naopak. Synchronizace těchto dvou kroků je zajištěna callbacky, které DMA vyvolává při dosažení poloviny a konce bufferu. Callback nastaví ukazatel pro ukládání hodnot na začátek druhé poloviny bufferu a signalizuje algoritmu, že je možno ukládat další hodnoty. Po dosažení konce bufferu přejde DMA automaticky zpět na začátek díky kruhové inkrementaci zdrojové adresy. Rychlost výpočtu hodnot musí být dostatečně vysoká, aby za dobu, kdy DMA vyčítá data z jedné poloviny bufferu dokázal algoritmus zaplnit druhou polovinu a zbyl dostatečný čas na obsluhu případné příchozí komunikace. Použitý buffer má velikost 512 dvojic 16bitových hodnot, což znamená 2048 B.



Obr. 3.5: Znázornění jednoho kroku plnění a vyčítání bufferu hodnot

V ideálním případě by DMA mělo nastavitelnou hodnotu inkrementace adresy a mohlo by tak přímo z LUT vybírat hodnoty přesně podle aktuální fáze. To ale možné není, a tak je hlavním úkolem algoritmu vybírat z LUT správné hodnoty a po dávkách je ukládat do bufferu, odkud je

pak DMA přenáší do DAC. Druhou možností, jak tento problém obejít, by bylo implementovat variabilní délku LUT, jako to například udělal pan Jiří Hladík ve své práci [3]. Nastavení frekvence v takovém případě probíhá nalezením vhodné kombinace délky LUT a obnovovací frekvence. Díky tomu by nebylo třeba mezikroku kopírování vybraných vzorků z LUT do bufferu, ale DMA by mohlo vyčítat hodnoty přímo z LUT.

Pro svou univerzálnost však pro tento přístroj bylo vybráno řešení pomocí čistého DDS algoritmu. Zmíněnou metodu s variabilní LUT je však možno implementovat v budoucnosti, pokud bude potřeba vyšších obnovovacích kmitočtů algoritmu.

3.2.3 Fázový akumulátor

Pro uchovávání informace o aktuální fázi signálu slouží fázový akumulátor. Zatímco v hardwaru by byl implementován jako samostatný registr a sčítačka, zde jej v programu nahrazuje proměnná, která je v každém kroku cyklu inkrementována. Protože tato proměnná je využívána v časově kritické sekci v cyklu, je třeba, aby operace s touto proměnnou probíhaly co nejrychleji.

Protože jádro použitého mikrokontroléru je 32bitové, není důvod použít pro tuto proměnnou menší datový typ, než 32bitový celočíselný uint32_t. Získáme tím více než dostačující frekvenční rozlišení, zatímco operace s touto proměnnou jsou stále jedno cyklové. Díky využití celého rozsahu proměnné není třeba po jejím inkrementování kontrolovat překročení maximální hodnoty, protože hodnota automaticky přeteče. Při obnovovací frekvenci algoritmu $f_S = 1$ MHz, dané maximální obnovovací frekvencí DA převodníku, vychází frekvenční rozlišení dle rovnice 3.1.

$$\Delta f = \frac{1}{2^{32}} f_S = 0,233 mHz \tag{3.1}$$

Inkrement fáze

Hodnota inkrementu fáze je přímo závislá na požadované frekvenci. Při každé změně nastavené frekvence je hodnota inkrementu vypočítána dle následující rovnice 3.2, kde f je požadovaná frekvence, 2^{32} je maximální hodnota fázového akumulátoru (neboli délka jedné periody) a f_S je obnovovací frekvence DAC.

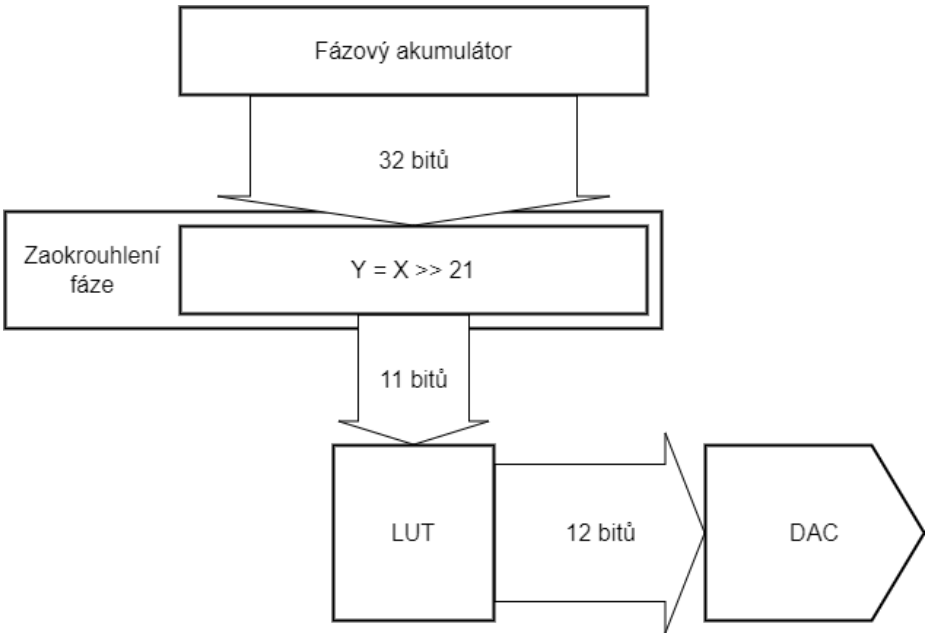
$$\Delta\varphi = f\frac{2^{32}}{f_S} \tag{3.2}$$

3.2.4 Zaokrouhlování fáze

Pro plnou kvalitu generovaného signálu s výše vypočítaným frekvenčním rozlišením by bylo potřeba pro každou z možných hodnot fáze mít odpovídající hodnotu amplitudy v převodníku fáze-amplituda. Taková převodní tabulka by se však nevešla ani do celého adresovatelného prostoru 32bitového mikrokontroléru, natož do jeho RAM.

Proto je použito zaokrouhlování fáze, které za cenu malé nepřesnosti výsledné fáze dovolí násobně zmenšit velikost LUT při zachování frekvenčního rozlišení. V tomto přístroji je využito

toho, že pokud jsou velikosti vstupního i výstupního rozsahu mocniny 2, lze zaokrouhlení provést jednoduše a efektivně aritmetickým posunem doprava. Velikost dostupné RAM paměti na použitém mikrokontroléru STM32G431KB je 32 kB. S ohledem na další buffery (buffer hodnot pro DMA, Rx a Tx buffer USB) a další proměnné byla zvolena velikost LUT 2048 vzorků, tedy 4096 B na každý kanál.



Obr. 3.6: Znázornění zaokrouhlení fáze

Jitter-free frekvence

Tabulka 3.1 nabízí „jitter-free“ frekvence navrhovaného signálového generátoru dle rovnice 2.14 pro $f_S = 1$ MHz a s délkou LUT 2048 vzorků.

frekvence [Hz]	vzorků na periodu
500000	2
250000	4
125000	8
62500	16
31250	32
15625	64
7812,5	128
3906,25	256
1953,125	512
976,5625	1024
488,28125	2048
244,140625	4096
122,0703125	8192
61,03515625	16384
30,51757813	32768
15,25878906	65536
7,629394531	131072
3,814697266	262144
1,907348633	524288
0,9536743164	1048576
...	...
...	...
...	...
0,0002328306437	4294967296

↑
zmenšené rozlišení
plné rozlišení
↓

Tabulka 3.1: Jitter-free frekvence

3.2.5 Převod fáze-amplituda

Pro převod z fáze na odpovídající napětí signálu je třeba znát přesný tvar signálu. Ten ovšem záleží kromě obecného tvaru (např. sinus, pila, trojúhelník, ...) také na nastavené amplitudě a napěťovém offsetu. Tyto parametry jsou nastavované uživatelem a lze tedy předpokládat, že se budou vzhledem k frekvenci výpočtů měnit jen zřídka. Proto lze tyto uživatelské parametry v krátkodobém měřítku považovat za konstanty a není tedy třeba s nimi v každém kroku algoritmu znovu operovat. Namísto toho je vliv těchto nastavení promítnut přímo do hodnot v LUT.

Při každé změně nastavené amplitudy, offsetu nebo tvaru je proveden „předvýpočet“ LUT. Požadovaný obecný tvar signálu je načten z Flash paměti. Jeho hodnoty jsou nejdříve vynásobeny na požadovanou amplitudu, potom posunuty o napěťový offset, a nakonec je jejich hodnota omezena na maximální a minimální hodnotu DAC převodníku. Takto upravený průběh je uložen v RAM a odtud je poté používán DDS algoritmem jako tabulka pro převod z fáze na amplitudu. Hodnoty základního signálu ve Flash mají 16bitové rozlišení, ale po „předvýpočtu“ jsou hodnoty omezeny na 12 bitů, což je maximální rozlišení DAC. Pro omezení hodnot je použita instrukce SSAT [25], která u přesahujících hodnot nezpůsobí přetečení, ale omezí je na maximální nebo minimální hodnotu rozsahu.

Kromě běžně využívaného přístupu s LUT nabízí mikrokontrolér STM32G431 matematický koprocesor CORDIC, který dokáže během několika taktů procesoru vypočítat hodnotu trigono-

metrické funkce i bez tabulky v paměti. Protože však druhý použitý mikrokontrolér STM32F303 tento modul neobsahuje, nebyl pro toto zařízení zatím uvažován.

3.2.6 Úprava DDS algoritmu pro frekvenční sweep

Běžný signálový generátor umožňuje generovat kromě periodických signálů i frekvenční sweep. Od tohoto přístroje se tedy též očekává přítomnost této funkce. Výše popsaný DDS algoritmus lze s menší úpravou použít i pro generování frekvenčního sweepu.

Implementace lineárního sweepu

Lineárním sweepem se rozumí generování signálu, jehož okamžitá frekvence se lineárně mění v čase. Průběh frekvence v čase lze zapsat následujícími rovnicemi 3.3 a 3.4.

$$f\left[x\right] = f_{start} + \frac{x}{délka}(f_{konec} - f_{start}) \tag{3.3}$$

$$f\left[x\right] = f\left[x-1\right] + \frac{f_{konec} - f_{start}}{délka} \tag{3.4}$$

Z druhé rovnice je zřejmé, že frekvence mezi dvěma kroky algoritmu se vždy liší jen o přičtenou konstantu. Vzhledem k tomu, že inkrement fáze je přímo závislý na požadované frekvenci, lze lineární sweep realizovat přidáním jedné operace do algoritmu. V každém kroku je stejně jako v normálním algoritmu inkrementována fáze a proveden převod fáze na amplitudu. Oproti normálnímu algoritmu je zde navíc navýšení (nebo snížení) inkrementu fáze o konstantu předem vypočítanou ze zadané počáteční a koncové frekvence a doby přeběhu.

Protože fázový akumulátor, inkrement fáze i navyšování inkrementu fáze jsou prováděny v celočíselné aritmetice, může dojít opakovaným přičítáním stejné nepřesné hodnoty k akumulaci chyby. Ta se projeví buďto pomalejším nebo rychlejším růstem frekvence, než jaký byl zadán. Proto je implementován mechanismus korekce této chyby, který po vypočítání jedné poloviny bufferu opraví aktuální frekvenci podle počtu již vypočítaných vzorků a požadované délky sweepu – tedy podle rovnice 3.3.

Implementace logaritmického sweepu

Logaritmický sweep mění frekvenci exponenciálně. To znamená, že každou frekvenční dekádu projde za stejný čas. Díky tomu lze říci, že logaritmický sweep vykreslí na každou dekádu stejný počet period, což neplatí o lineárním sweepu. Lineární sweep na nižších frekvencích nemusí vykreslit ani jednu kompletní periodu zatímco na vyšších frekvencích jich vykreslí několik. Logaritmický sweep má pro každou dekádu stejné poměrné rozlišení a pokud vyneseme do grafu průběh frekvence v čase s logaritmickou osou frekvence, získáme přímku. Logaritmický sweep je tedy vhodnější pro větší frekvenční rozsahy sweepů a také například pro určování impulzní odezvy systémů [14]. Pro logaritmický průběh frekvence lze napsat následující rovnice.

$$f\left[x\right] = f_{start} 10^{\frac{x}{délka} log_{10}\left(\frac{f_{konec}}{f_{start}}\right)} \tag{3.5}$$

$$f[x] = f[x-1]\, 10^{\frac{log_{10}\left(\frac{f_{konec}}{f_{start}}\right)}{délka}} \tag{3.6}$$

$$\frac{f[x]}{f[x-1]} = 10^{\frac{log_{10}\left(\frac{f_{konec}}{f_{start}}\right)}{délka}} = \alpha \tag{3.7}$$

Frekvence dvou po sobě jdoucích kroků při logaritmickém sweepu jsou spolu vždy v konstantním poměru. V každém kroku je tedy třeba frekvenci vynásobit konstantou předem vypočítanou ze zadaných parametrů sweepu.

Tento výpočet je zřejmě ještě náchylnější na kumulaci chyby z důvodu násobení proměnných s omezenou přesností. Protože je však tento výpočet prováděn pro každý krok algoritmu, tedy milionkrát za sekundu, není možné pracovat s čísly s plovoucí desetinnou čárkou s plným rozlišením. Namísto toho je operace v během výpočtu dávky hodnot prováděna na proměnných typu float, a stejně jako v případě lineárního sweepu je hodnota frekvence po každé dávce hodnot korigována. V tomto případě absolutní výpočet zahrnuje funkce 10^x a $\log_{10} x$, které jsou výpočetně náročnější. Pro delší doby sweepu, pro které vychází dle rovnice 3.7 malá hodnota poměru α, je však korekce tímto způsobem nezbytná. Výpočet opravy frekvence se provádí dle rovnice 3.5.

Spouštění frekvenčního sweepu

Pro režim frekvenčního sweepu byla implementována možnost začít generování signálu na jeden z několika spouštěčů a také možnost ukončit generování signálu po daném počtu celých sweepů.

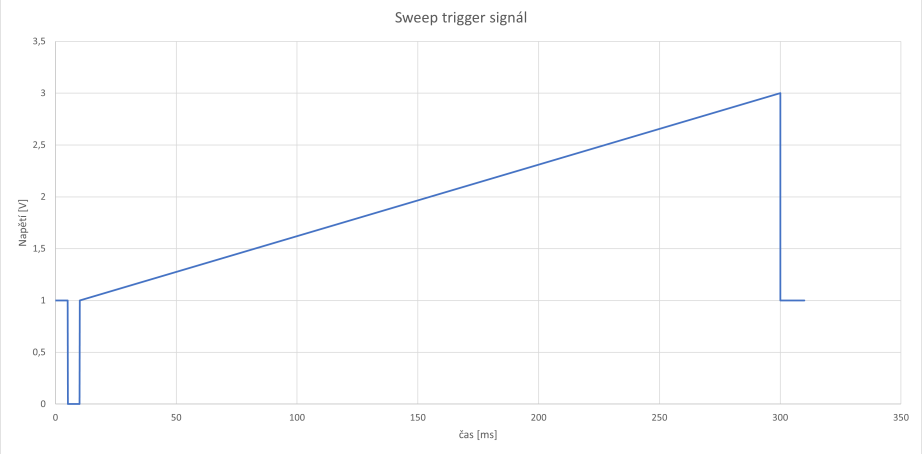
Sweep lze spustit jedním z následujících způsobů:

- Vzestupná hrana na pinu PA8
- Sestupná hrana na pinu PA8
- Vzestupná/sestupná hrana na pinu PA8
- Stisk tlačítka GO v ovládacím panelu přístroje v DataPlotteru
- Automatické nepřetržité spouštění (automatické opakování sweepů donekonečna)

Implementace triggerovacího signálu

Z důvodu omezeného výkonu mikrokontroléru a vyšší výpočetní náročnosti logaritmického sweepu je sweep generován pouze na prvním kanále. Stále však zbyl dostatek výkonu na použití druhého kanálu pro generování signálu pro jednoduché triggerování osciloskopu.

Se začátkem sweepu je na triggerovacím signálu sestupná hrana z 1 V na 0 V. Následujících 5 ms je signál stále na 0 V. Po tomto čase přichází vzestupná hrana na 1 V a ihned po ní signál lineárně roste až do konce sweepu, kdy dosáhne 3 V. Pokud šlo o poslední sweep ze série, zůstane po konci signál na 1 V, pokud ihned pokračuje další sweep, přejde signál rovnou na první popsaný krok. Graf na obrázku 3.7 zobrazuje popsaný časový průběh.



Obr. 3.7: Příklad průběhu trigger signálu v režimu sweep

S nastavením triggeru osciloskopu dle tabulky 3.2 tak lze jednoduše zachytit začátek, konec nebo průběh sweepu.

	Úroveň	Hrana
Začátek sweepu	0 – 1 V	Vzestupná/sestupná
Během sweepu	1 – 3 V	Vzestupná hrana
Konec sweepu	1 – 3 V	Sestupná hrana

Tabulka 3.2: Nastavení triggeru osciloskopu pro zachycení různých částí sweepu

3.2.7 Výpočet parametrů přístroje

Nyní, když známe parametry implementace DDS algoritmu, můžeme vypočítat teoretické odhady hodnot SQR pro různé frekvence. Tabulka 3.3 nabízí přehled parametrů relevantních pro výpočet SQR neboli odstupu signálu od kvantizačního šumu.

Rozlišení DAC [bity]	n	12
Vzorkovací frekvence [Hz]	f_S	1 MHz
Max. rozkmit DAC [V]	FS	3,3 V
Max. rozkmit bez zkreslení [V]	A_{sigp-p}	3 V

Tabulka 3.3: Seznam parametrů pro výpočet SQR

Důvodem pro zvolení nižšího rozkmitu signálu, než je maximum DAC je zkreslení výstupu okolo minimálního a maximálního napětí. DAC modul totiž sice pracuje s referenčním napětím 3,3 V a maximální vstupní digitální hodnota teoreticky odpovídá 3,3 V, ale ve skutečnosti nedokáže DAC generovat napětí menší než cca 60 mV a vyšší než $V_{ref} - 60$ mV. Proto byl zvolen maximální rozkmit 3 V, při kterém se již zkreslení neprojevuje.

Následující rovnice 3.8 je zde uvedena k připomenutí vztahu pro výpočet odstupu signálu od kvantizačního šumu.

$$SQR = 1,76 + 6,02n + 10\log_{10}\left(\frac{f_S}{2f_{sig}}\right) - 20\log_{10}\left(\frac{A_{sig\ p-p}}{FS}\right)\ [dB] \tag{3.8}$$

Podle této rovnice byla vypočítána následující tabulka 3.4 ukazující závislost SQR na generované frekvenci. Je zde vidět, že s rostoucí frekvencí klesá poměr oversamplingu a tedy klesá SQR, protože roste kvantizační šum.

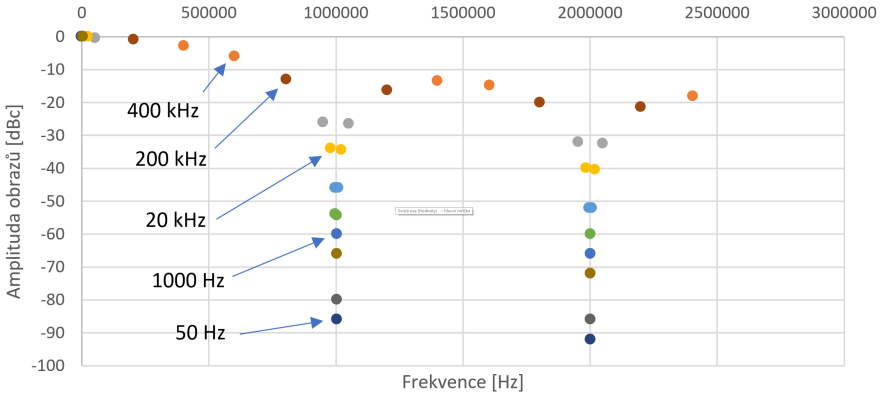
frekvence [Hz]	SQR [dBc]
50	-114,000
100	-110,990
500	-104,004
1000	-100,997
2000	-97,994
5000	-94,036
10000	-91,061
20000	-88,121
50000	-84,345
100000	-81,655

Tabulka 3.4: SQR v závislosti na generované frekvenci

Tyto výsledky však uvažují pouze nekorelovaný kvantizační šum. V praxi to znamená, že úroveň rušení v okolí frekvence signálu pravděpodobně bude vyšší, protože kvantizační šum nebude mít konstantní hodnotu v celém frekvenčním rozsahu.

Kromě úrovně kvantizačního šumu je důležité znát i vliv aliasingu na spektrum generovaného signálu. Jak již bylo zmíněno, kvůli vzorkování signálu se ve spektru objevují obrazy generovaného signálu na frekvencích dle rovnice 2.3 a s amplitudou dle funkce 2.4. V následujícím grafu jsou

vyneseny frekvence a amplitudy signálů a jejich prvních čtyř obrazů. Ze spojnice bodů je vidět, že amplituda sleduje obálku funkce $\frac{\sin(x)}{x}$.



Obr. 3.8: Pozice a amplituda obrazů signálu v závislosti na frekvenci

Jitter-free frekvence již byly vypočteny v kapitole 3.2.4. Na druhou stranu frekvence, pro které vyjde zaokrouhlení nejhůře, budou tvořit ve spektru špičky o maximální amplitudě $-6,02 \cdot m = -6,02 \cdot 21 = -126,4$ dB. Špičky ve spektru vzniklé zaokrouhlováním fáze by tak teoreticky měly být pod úrovní kvantizačního šumu a obrazů signálu.

3.3 Uživatelské rozhraní

3.3.1 Filozofie ovládání

Zadáním této práce bylo navrhnout jednoduchý a přístupný přístroj. Hardwarově by neměl potřebovat o moc více, než jen samotný mikrokontrolér s nejnutnějšími podpůrnými součástkami. Zároveň je velmi pravděpodobné, že mikrokontrolér nebo deska bude relativně často nahráním jiného SDI měnit svoji roli. Nebylo by tedy žádoucí obklopit mikrokontrolér tlačítky a displejem, když by se tento obvod musel vždy se změnou firmwaru měnit. Ovládání tohoto přístroje je tedy realizováno PC aplikací, se kterou přístroj komunikuje pomocí USB nebo UART. Tento přístup byl vybrán i pro následující výhody.

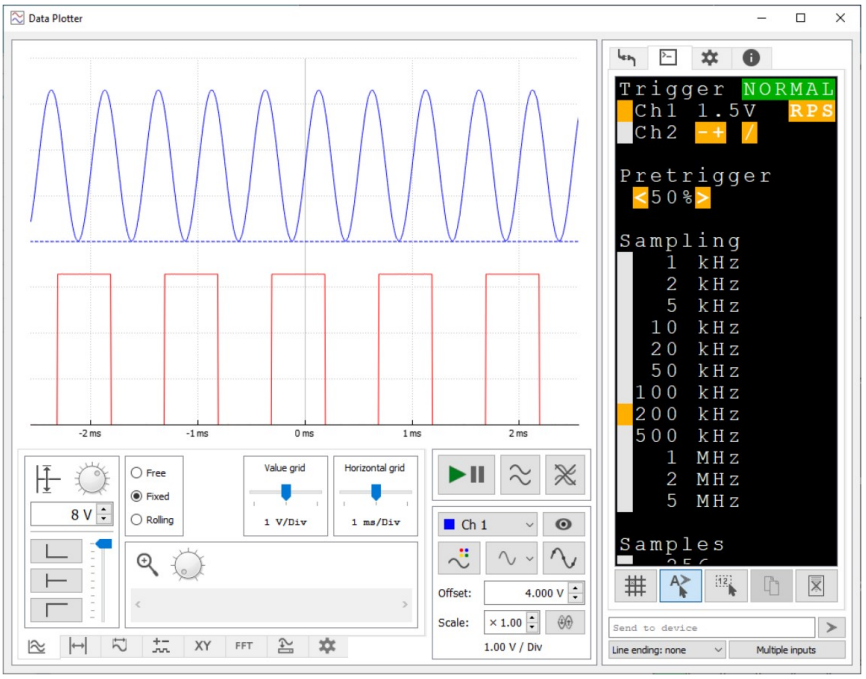
Vzhledem k výpočetnímu výkonu a paměti mikrokontroléru by grafické rozhraní realizované plně na mikrokontroléru bylo buďto velmi omezené nebo zbytečně náročné na implementaci. Díky přesunutí grafické části rozhraní na PC je možno soustředit výpočetní výkon a paměť mikrokontroléru téměř výhradně na funkci generátoru.

Grafické rozhraní v PC aplikaci také otevírá zcela nové možnosti pohledu na ovládání signálového generátoru. Věřím, že každý uživatel laboratorního signálového generátoru ocenil kolečko pro nastavení frekvence, které lze v PC aplikaci těžko napodobit. Nastavení ostatních funkcí laboratorního generátoru ale může být kvůli malému displeji a omezeným tlačítkům značně neintuitivní a nepraktické. V PC aplikaci však má designér volné ruce k implementaci nového způsobu nastavování parametrů přístroje.

V neposlední řadě je argumentem pro tento způsob ovládání i fakt, že tento přístroj bude pravděpodobně používán v laboratorní výuce společně s dalšími SDI, které rovněž používají připojení k PC pro zprostředkování informací uživateli.

3.3.2 Aplikace DataPlotter

Jako aplikace poskytující uživatelské rozhraní na PC byl zvolen výsledek práce pana Jiřího Maiera, program DataPlotter [1]. Tato aplikace byla vyvinuta jako rozhraní osciloskopu navrženého ve stejné práci. Díky tomu, že je navržena velmi univerzálně a nabízí designérovi možnost definice vlastního rozhraní, ji lze s výhodou využít i jako rozhraní pro signálový generátor.



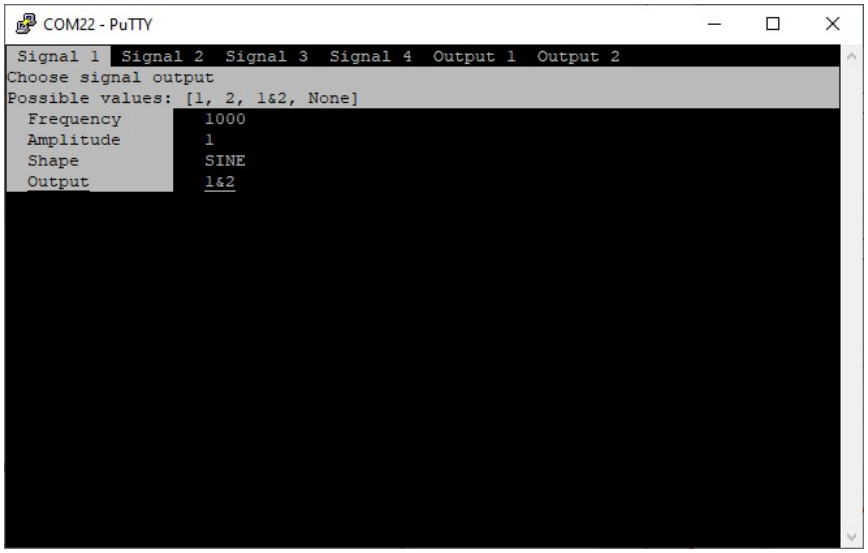
Obr. 3.9: Aplikace DataPlotter jako rozhraní osciloskopu (převzato z [1])

Program DataPlotter nabízí kromě běžného zobrazení naměřeného průběhu signálu i mnoho dalších funkcí, jako například XY režim, FFT, automatické měření veličin signálů nebo režim logického analyzátoru. Jeho nespornou výhodou oproti ostatním podobným zobrazovacím programům je ale plně konfigurovatelné ovládací rozhraní, čehož je v této práci plně využito. Obrázek 3.9 ukazuje aplikaci při připojení k jejímu osciloskopu. V levé horní části jsou zobrazeny průběhy signálů, pod nimi se nachází ovládací prvky zobrazení a pravý sloupec je vyhrazen pro terminál ovládání zařízení.

TUI terminál

Nastavování parametrů přístroje je v aplikaci možno realizovat pomocí tzv. TUI (Text-based User Interface) terminálu. Celé takovéto rozhraní je tvořeno sekvencí ASCII znaků, které tvoří v terminálu například výpisy hodnot nebo tabulky možností. Využívá se též tzv. „escape sekvencí“, což jsou speciální sekvence ASCII znaků. Tyto sekvence tvoří příkazy, kterými lze například upravit chování terminálu, nastavit pozici kurzoru nebo měnit barvy vykreslování. Vypisování tohoto rozhraní i vnitřní logika a uspořádání ovládacích prvků je implementováno plně na přístroji. V PC pouze běží terminál, který zobrazuje přijaté znaky.

S běžným TUI rozhraním lze interagovat buďto pomocí příkazů zadávaných na klávesnici nebo pomocí výběru možností šipkami. Přístroj po připojení odešle sekvenci ASCII a escape znaků, která vykreslí v terminálu požadované informace. Poté přístroj přijímá znaky stisknuté na klávesnici, vyhodnocuje je a v reakci na ně odesílá nové znaky, které upraví nebo přepíšou rozhraní v terminálu. Pro interakci s přístrojem je však toto poněkud nepraktické, a navíc není vůbec využito možností, které nabízí ovládací prvky PC.



Obr. 3.10: Ukázka TUI rozhraní signálového generátoru ze semestrálního projektu předcházejícího této práci

Proto byl terminál DataPlotteru svým autorem doplněn o možnost interakce s vykreslenými ovládacími prvky myší. Znaky, které byly pomocí escape sekvencí vykresleny s upravenou barvou pozadí, se stávají tlačítky, která po kliknutí odešlou do zařízení svoji hodnotu. Jedním kliknutím tak lze například přímo vybrat konkrétní vypsanou hodnotu namísto navigování mezi možnostmi šipkami. DataPlotter dokonce obsahuje režim návrhu takovéhoto rozhraní, který pomáhá například s generováním příkazů pro změnu pozice kurzoru nebo pro změnu stylu písma či pozadí.

Omezením tohoto přístupu je však to, že každé tlačítko musí obsahovat jiný znak, jinak by mezi nimi zařízení nebylo schopné rozlišit. Stejný znak, jako je zobrazen, je totiž i posílán do zařízení jako odezva na jeho stisknutí. I když je toto vylepšení krokem správným směrem, stále nedosahuje uživatelská přívětivost stejné úrovně, jakou dovolují nativní PC aplikace.

QML terminál

Všechna výše zmíněná omezení však odstraňuje nová vlastnost DataPlotteru, kterou jeho autor přidal po publikování své práce. Jde o možnost definovat vlastní ovládací panel zařízení pomocí QML jazyka. Celý front-end aplikace DataPlotter je založen na frameworku Qt, jehož jedním modulem je i QML. Zpřístupnění části okna aplikace pro vývojáře jiných zařízení tedy nabízí možnost navrhnout rozhraní využívající stejné prvky, jako běžné PC aplikace.

3.3.3 Jazyk QML

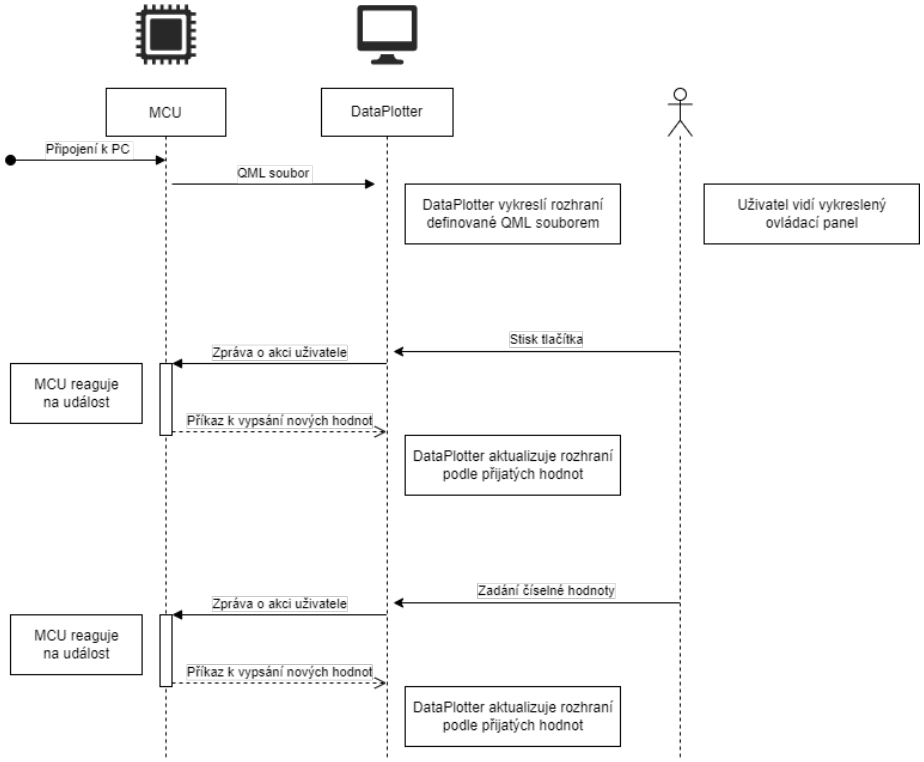
Zkratka tohoto jazyka znamená Qt Modeling Language. Česky se dá popsat jako značkovací (markup) jazyk pro popis uživatelských rozhraní. QML soubor popisuje hierarchickou strukturu objektů, jejich vlastnosti a jejich propojení. Díky možnosti používat JavaScript přímo v QML souboru, lze jednoduše definovat i zpracování událostí. Pro detailní vysvětlení základů i pokročilých struktur jazyka QML lze doporučit oficiální online dokumentaci [15], kde lze nalézt i množství příkladů. Protože použití QML namísto TUI je mezi SDI přístroji novým přístupem k popisu uživatelského rozhraní, neuškodí tento přístup rozebrat detailněji.

3.3.4 Základní princip fungování QML terminálu v DataPlotteru

Stejně jako TUI, i QML terminál je kompletně řízen mikrokontrolérem. Aplikace DataPlotter interpretuje obdržený popis rozhraní a vykreslí podle něj ve vyhrazeném okně grafické a ovládací prvky. Díky zmíněné podpoře JavaScriptu též DataPlotter reaguje na akce uživatele (například kliknutí na tlačítko), typicky odesláním zprávy mikrokontroléru. Zjednodušená sekvence komunikace mezi přístrojem a DataPlotterem je znázorněna diagramem na 3.11.

Prvním krokem v komunikaci mezi mikrokontrolérem a DataPlotterem musí být odeslání QML souboru, který definuje požadované uživatelské rozhraní. Tento soubor může mít v závislosti na složitosti rozhraní i několik kilobajtů, ale je poslán pouze jednou po připojení. Na základě tohoto souboru DataPlotter zobrazí rozhraní a od této chvíle s ním uživatel může interagovat. Díky tomu, že si mikrokontrolér definuje vlastní rozhraní, může DataPlotter sloužit různým přístrojům, které si každé nadefinuje vlastní rozhraní.

I když mikrokontrolér může redefinovat rozhraní posláním nového QML souboru, pro další komunikaci mezi mikrokontrolérem a DataPlotterem většinou stačí pouze krátké zprávy, které nastavují hodnoty proměnných definovaných v QML souboru. Když uživatel v rozhraní například klikne na tlačítko nebo nastaví v textovém poli novou hodnotu, je o tom poslána zpráva mikrokontroléru. Ten na ní zareaguje (například nastavením amplitudy generovaného signálu) a poté může (ale nemusí) poslat DataPlotteru zprávu, která aktualizuje zobrazovaný obsah rozhraní. V této zprávě nejde o novou definici rozhraní, ale jen o nastavení proměnných definovaných v QML souboru. Takovou proměnnou může být například stav zaškrtnutí políčka nebo zobrazený text. Další příklady lze nalézt v následujících ukázkách syntaxe QML a tvorby jednotlivých ovládacích prvků.



Obr. 3.11: Sekvence komunikace mezi QML rozhraním v DataPlotteru a mikrokontrolérem

3.3.5 Nástroje pro vývoj rozhraní v jazyce QML

Vývojáři se pro návrh rozhraní pro DataPlotter nabízí dvě možnosti vývojových prostředí.

Qt Creator

Qt Creator je oficiální integrované vývojové prostředí pro návrh aplikací s uživatelským rozhraním. Nabízí jak nástroje pro vývoj back-endu rozhraní tak i nástroj Qt Designer pro návrh vizuální stránky rozhraní. Pro návrh rozhraní pro DataPlotter může být vhodný právě program Qt Designer, ve kterém lze v grafickém rozhraní navrhnout rozmístění a vlastnosti vyvíjeného uživatelského rozhraní. Qt Creator je zdarma pro open-source projekty nebo lze stáhnout 10-denní zkušební verzi.

Vývoj v textovém editoru

Druhou možností je, jako pravděpodobně u všech programovacích jazyků, psát kód bez jakéhokoliv IDE, jen v textovém editoru a s pomocí externího kompilátoru či interpreteru. Výhodné je použít textový editor s podporou zvýrazňování syntaxe, jako například Notepad++, pro který existuje plugin pro QML soubory. S jednodušším strukturami rozhraní lze v podobném programu bez větších problémů pracovat.

Vývojářské nástroje DataPlotteru

Ať už si vývojář vybere kterýkoliv z přístupů, pro použití výsledného rozhraní v DataPlotteru je třeba ověřit jeho kompatibilitu s DataPlotterem. Pro tento účel nabízí DataPlotter vývojářské nástroje, které lze povolit v nastavení. S pomocí vývojářských nástrojů lze manuálně vybrat a nahrát QML soubor z PC, jakoby přišel z mikrokontroléru. Poté, co se vyvinuté rozhraní osvědčí tímto způsobem, může si vývojář nechat nahraný QML soubor DataPlotterem zkomprimovat a exportovat jako konstantní pole znaků, které lze přímo vložit do zdrojového kódu pro mikrokontrolér. Následující výstřižek z firmwaru mikrokontroléru ukazuje, jak lze toto pole použít.

```
// QML soubor k odeslání DataPlotteru
const unsigned char terminalQml[4263] = {'$','$','Q',1,1,57,103,51,  ...  ,0};
const uint16_t qml_length = sizeof(terminalQml)/sizeof(terminalQml[0]);

/*
 * Pošle QML soubor aplikaci DataPlotter
 * DataPlotter podle něj zobrazí uživatelské rozhraní
 *
 * QML soubor je vygenerován v DataPlotteru
 */
int16_t GUI_SendQML() {
    // odeslání přes USB
    CDC_SendFile(terminalQml, qml_length);
    return 0;
}
```

Pro tuto práci byl použit druhý popsaný přístup k vývoji. Veškerý QML kód byl psán v programu Notepad++ a ověřován byl přímo v DataPlotteru.

3.3.6 Příklady využití QML jazyka v této práci

Objekt Rect v QML

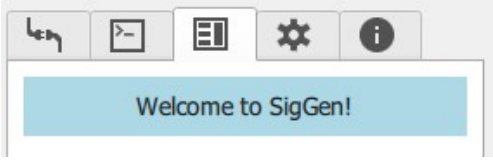
Syntaxi a vlastnosti jazyka QML je nejlepší ukázat na jednoduchém příkladu. Snad nejjednodušším objektem, který lze v QML vytvořit, je obdélník. Lze ho využít buďto přímo jako obdélník vykreslený na obrazovce nebo jako objekt, ke kterému lze vztáhnout pozici jemu podřízených objektů. Zde jsou například využity obě vlastnosti najednou pro vytvoření nadpisu rozhraní na světle modrém podkladu.

```
Rectangle{
    anchors.horizontalCenter: parent.horizontalCenter

    height: title.implicitHeight*2
    width: parent.width

    color: "lightblue"

    Label {
        id: title
        anchors.centerIn: parent
        horizontalAlignment: Text.AlignHCenter
        text: "Welcome to SigGen!"
    }
}
```



Obr. 3.12: Využití objektu Rectangle

V kódu si lze všimnout, že objektům lze přiřadit různé vlastnosti (properties), zde jsou to například vlastnosti `height`, `width`, `color`, `id` nebo `text`. Některé vlastnosti mají objekty defaultně, jiné jim lze explicitně vytvořit.

Pozornost je třeba věnovat přiřazování hodnot proměnným či vlastnostem. V QML totiž existují dva způsoby, jak nastavovat jejich hodnotu: buďto statickým přiřazením nebo pomocí tzv. „property binding“. Statické přiřazení hodnoty funguje stejně jako v běžném imperativním programovacím jazyce, tedy hodnota je přiřazením nastavena a zůstává taková, dokud není dalším přiřazením změněna. Property binding naproti tomu hodnotu cílové proměnné „sváže“ zadaným výrazem s hodnotou zdrojové proměnné a bude se měnit vždy, když se změní zdrojová proměnná. Takto lze například jednoduše implementovat rozhraní, které automaticky reaguje na změnu velikosti okna. V příkladu výše je pomocí property binding svázána hodnota `height` s hodnotou `title.implicitHeight` tak, že `height` bude vždy dvakrát větší, než `title.implicitHeight`.

Další vlastností, kterou lze v příkladu najít, je hierarchická struktura objektů. Objekt Label je zde vytvořen uvnitř bloku Rectangle, a je tak jemu podřízený.

Kromě jednoduchých objektů, jako již zmíněný obdélník, nabízí QML mnoho komplexnějších objektů. V této práci jsou z nich použity například tlačítka, textová pole nebo rozbalovací nabídky.

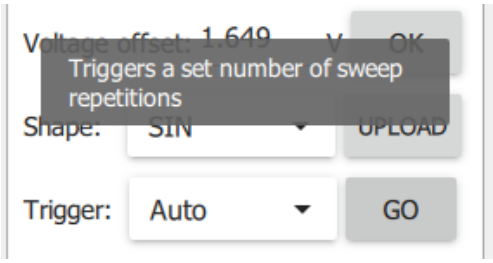
Tlačítko v QML

Tlačítko je základním ovládacím prvkem v aplikacích na PC. Jeho vytvoření v jazyce QML je stejně jednoduché, jako předchozí příklad. Následující blok kódu vytvoří tlačítko s nápisem GO, u kterého se navíc zobrazí nápověda, pokud nad ním uživatel 1 sekundu podrží kurzor.

```
Button {
    id: swtr1Btn
    text: "Upload"
    font.pointSize: 11

    hoverEnabled: true
    ToolTip.visible: hovered
    ToolTip.delay: 1000
    ToolTip.text: qsTr("Triggers a set number of sweep repetitions")

    onClicked: sendVar(1, "go1")
}
```



Obr. 3.13: Ukázka tlačítka s nápovědou

Zpracování událostí

Kromě vytvoření tlačítka tento kód také demonstruje způsob zpracování událostí. Události, jako například kliknutí myší, stisknutí klávesy nebo podržení kurzoru nad objektem, se v QML nazývají Signals. Na tyto signály mohou reagovat funkce zvané Handlers. V tomto případě jde o handler onClicked, který při obdržení signálu kliknutí na tlačítko spustí JavaScriptovou funkci `send()`.

Zde použitá funkce `send()` odešle skrze DataPlotter do zařízení příkaz `##Vgo1:1`. Následující blok kódu ukazuje, jak je tato funkce implementována.

```
function sendVar(value, name) {
    // Pošle zprávu se zkráceným jménem a hodnotou proměnné
    var msg = "##V%1:%2;"
    dataPlotter.transmitToSerial(msg.arg(name).arg(value))
}
```

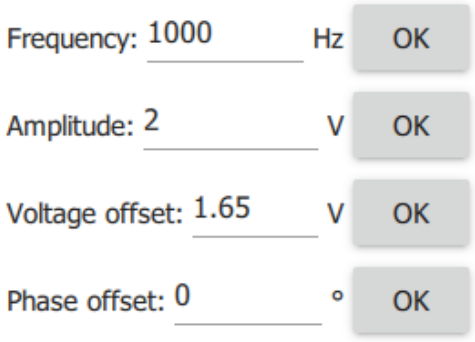
Textové pole

Pro nastavování hodnot parametrů je v tomto zařízení využito textových polí, která umožňují přímé zadávání číselných hodnot na klávesnici. Jejich implementace v QML je taktéž velmi jednoduchá, jak ukazuje následující blok kódu.

```
TextField {
    id: f1Text
    selectByMouse: true
    maximumLength: 9
    implicitWidth: 110
    validator: DoubleValidator{}

    hoverEnabled: true
    ToolTip.visible: hovered
    ToolTip.delay: 1000
    ToolTip.text: qsTr("Frequency to be generated\n[0 Hz - 100 kHz]")

    onAccepted: sendVar(f1Text.text,"f1")
}
```



Obr. 3.14: Ukázka textového pole

Vlastnost `selectByMouse` umožňuje označovat kurzorem text a přepisovat jej, `maximumLength` určuje maximální počet číslic, `implicitWidth` udává šířku textového pole, a nakonec `validator` omezuje zadatelné znaky na číslice a desetinnou čárku. Handler `onAccepted` zajistí odeslání zadané hodnoty při stisknutí klávesy Enter.

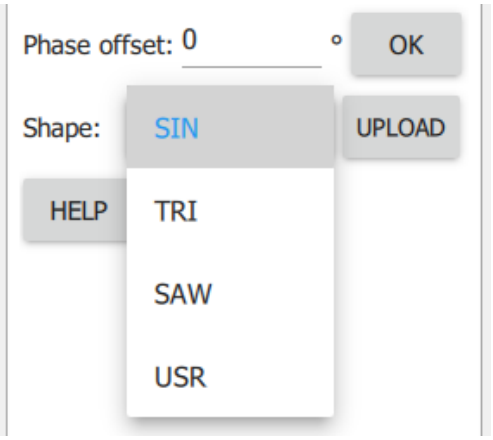
Rozbalovací nabídka

Pro výběr z omezeného množství předem známých možností je ideálním prvkem rozbalovací nabídka, kterou QML nabízí pod názvem ComboBox. Prvním kliknutím na prvek se objeví nabídka možností, ze kterých si uživatel může kliknutím vybrat nebo kliknout mimo pro zavření nabídky. V rozhraní tohoto přístroje je rozbalovací nabídka využita pro výběr tvaru signálu a zdroje triggeru. Kód použitý pro použitý ComboBox vypadá následovně.

```
ComboBox {
    id: s1Combo
    textRole: "key"
    model: ListModel {
        ListElement { key: "SIN"; value: 0 }
        ListElement { key: "TRI"; value: 1 }
        ListElement { key: "SAW"; value: 2 }
        ListElement { key: "USR"; value: 3 }
    }

    onActivated: sendVar(s1Combo.currentIndex, "s1")

    hoverEnabled: true
    ToolTip.visible: hovered
    ToolTip.delay: 1000
    ToolTip.text: qsTr("Choose signal shape here")
}
```



Obr. 3.15: Ukázka rozbalovacího menu

Záložky

Protože tento přístroj má dva nezávislé výstupní kanály, u kterých lze nastavovat stejné parametry, je vhodné tato nastavení seskupit podle kanálů. Toho je dosaženo použitím dvou stránek, mezi kterými se přepíná pomocí záložek. Stejný přístup je použit i pro přepínání mezi běžným režimem generátoru a režimem frekvenčního sweepu. V QML toho lze dosáhnout pomocí objektů TabBar a StackLayout.

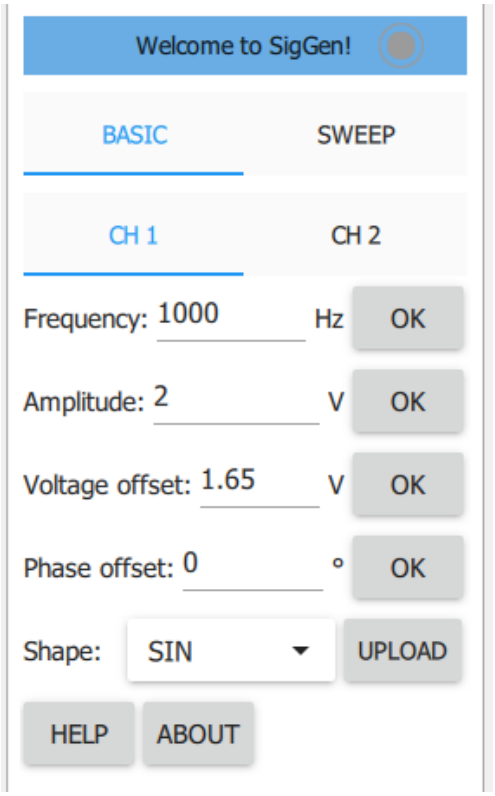
Přepínání mezi stránkami funguje na následujícím principu. Stisknutím některého z tlačítek TabButton dojde kromě spuštění handleru `onClicked` a funkce `sendVar()`, také ke změně vlastnosti `sweep1Bar.currentIndex`, která označuje právě aktivní tlačítko. Na tuto vlastnost je navázána i vlastnost `currentIndex` objektu StackLayout, který tím změní aktuálně zobrazovanou stránku.

```
TabBar {
    id: sweep1Bar
    width: parent.width
    currentIndex: m1

    TabButton {
        text: qsTr("Basic")
        onClicked: sendVar(sweep1Bar.currentIndex, "m1")
    }
    TabButton {
        text: qsTr("Sweep")
        onClicked: sendVar(sweep1Bar.currentIndex, "m1")
    }
}
StackLayout {
    id: sweep1Stack
    width: parent.width
    height: sweep1Bar.height + title.height
    currentIndex: sweep1Bar.currentIndex
    y: sweep1Bar.y + sweep1Bar.height

    Item {
        // Obsah 1. stránky
    }
    Item {
        // Obsah 2. stránky
    }

}
```



Obr. 3.16: Ukázka záložek a stránky s nastavením jednoho kanálu

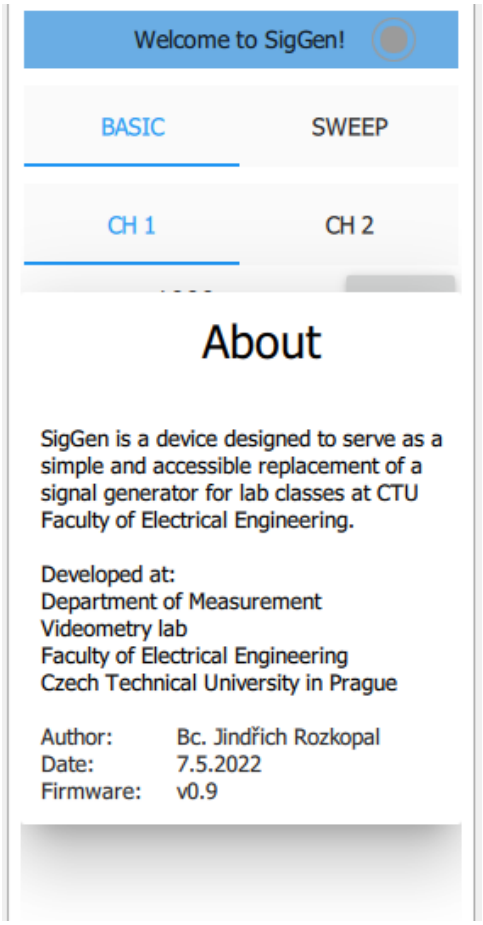
Vyskakovací okno

Posledním důležitým prvkem použitým v rozhraní navrhovaného přístroje je objekt Popup, neboli vyskakovací okno. Zde bylo vhodným kandidátem pro zobrazení nápovědy a informací o programu. Po stisknutí tlačítka Help nebo About je díky handleru `onClicked` zavolána funkce `popup.open()`, která vyvolá okno překrývající zbytek rozhraní.

```
Popup {
    id: aboutPopup
    anchors.centerIn: parent
    Column {

        // Obsah vyskakovacího okna

    }
}
```



Obr. 3.17: Ukázka vyskakovacího okna

3.4 Komunikační protokol mezi DataPlotterem a mikrokontrolérem

Aby dokázal přístroj pracovat s uživatelským rozhraním a měnit zobrazované informace, je třeba oboustranné komunikace mezi programem DataPlotter a mikrokontrolérem. Struktura zpráv, které dokáže DataPlotter zpracovat, je důkladně popsána v dokumentech [16] a [17], které autor vydal společně s programem DataPlotter. Definované zprávy umožňují DataPlotteru přijímat naměřené hodnoty, měnit nastavení programu, vypisovat informace do terminálu a v neposlední řadě také umožňují definovat vlastní ovládací prvky. V této práci použité zprávy jsou popsány v následujících kapitolách.

3.4.1 Zobrazení naměřených dat

DataPlotter nabízí několik způsobů, jak dokáže přijímat data k zobrazení v oblasti grafu. Pro toto zařízení byl jako nejefektivnější způsob vybráno odesílání dat pro celý kanál najednou. Tato zpráva je skládána dle následujícího předpisu.

```
$$C(číslo kanálu),(časový krok),(délka),(bity),(min),(max),(index nuly);u2(data);
```

Každá zpráva pro DataPlotter musí začínat znaky `$$`. Následující znaky `C` a číslo kanálu označují tuto zprávu jako odesílání více hodnot najednou pro označený kanál osciloskopu. Zpráva pokračuje záhlavím, které definuje parametry zprávy.

číslo kanálu Číslo kanálu, jehož data jsou posílány (číslováno od 1)

časový krok Časový rozestup mezi naměřenými vzorky

délka Počet odesílaných vzorků

bity Rozlišení vzorků

min Hodnota, na kterou bude namapováno přijaté číslo 0

max Hodnota, na kterou bude namapováno přijaté číslo 2^{bity}

index nuly Index prvního právě odesílaného vzorku - lze přepisovat starší vzorky

typ vzorku Typ a velikost proměnné, zde u2 - uint16_t (Little-endian pořadí)

data Udaný počet naměřených vzorků v binární formě

Této zprávy je využito pokaždé, když je změněn tvar generovaného signálu. Odesílaný náhled má proměnný počet vzorků v závislosti na frekvenci signálu. Pokud pro generovaný signál vychází více než 2048 vzorků na jednu periodu, je z důvodu šetření kapacity komunikace odesláno právě všech 2048 hodnot, které obsahuje LUT. V případě, že pro signál vychází méně než 2048 vzorků na periodu, jsou odesílané vzorky vybrány stejným algoritmem jako jsou generovány, a náhled tedy korektně reprezentuje generovaný signál. Pokud frekvence obou kanálů nejsou stejné, je signál s Ykratší periodou opakován pro zaplnění stejného časového intervalu na obrazovce. Maximální počet odeslaných hodnot je omezen na 2^{15}, aby odesílání náhledu v případě velmi rozdílných frekvencí netrvalo příliš dlouho.

3.4.2 Výpis do terminálu

Pro ladění programu a pro informování uživatele o přijetí příkazu mikrokontrolérem je využito informačních a varovných zpráv. Informační zprávy začínající znaky `$$I` jsou odesílány mikrokontrolérem pro potvrzení přijetí nastavení a varovné zprávy začínající `$$W` slouží pro výpis případných chybových hlášek. Po úvodních třech znacích rovnou pokračuje text k zobrazení a zpráva nemusí být jakkoli zakončena.

3.4.3 Nahrání QML souboru

Jak již bylo zmíněno, program DataPlotter umožňuje nahrát definici vlastního ovládacího panelu zařízení pomocí jazyka QML. Po ověření funkčnosti QML souboru pomocí vývojářských nástrojů v DataPlotteru je tento soubor zkomprimován a vložen do programu mikrokontroléru jako konstanta. Při kompilaci programu je tak tento soubor uložen do Flash paměti. Po připojení přístroje k DataPlotteru je soubor zprávou `$$Q` odeslán a DataPlotter podle něj vykreslí žádané ovládací prvky.

3.4.4 Úprava proměnných QML rozhraní

Aby mohl mikrokontrolér například měnit zobrazované informace, nabízí DataPlotter následující příkaz.

```
$$V(jméno proměnné):(hodnota);
```

Touto zprávou vydává přístroj příkaz DataPlotteru k úpravě hodnoty jemnované proměnné v QML souboru. Touto zprávou lze měnit hodnotu jakékoliv vlastnosti (property) definované v odeslaném QML souboru. Lze takto například měnit zobrazovaný text, vybranou záložku, barvu některého prvku či vybranou hodnotu v rozbalovacím menu. Každý ovládací prvek (tlačítka, textová pole atd.) má svou definující vlastnost (tlačítko stisknuto/nestisknuto, zobrazený text v textovém poli, atd.) spojenou s jejím aliasem, neboli zástupnou proměnnou s krátkým názvem, aby nebylo třeba pokaždé posílat celé jméno proměnné. Tato zástupná proměnná a její jméno je pak používáno při komunikaci mezi přístrojem a DataPlotterem. Následující výstřižek kódu ukazuje tento koncept.

```
// proměnná f1 a její prvotní nastavení
// frekvence 1. kanálu
f1: {1000.0}

// propojení proměnné f1 s textem zobrazeným v text. poli f1Text
property alias f1: f1Text.text

TextField {
    id: f1Text
    // objekt TextField má vlastnost TEXT definovanou implicitně

    // při potvrzení odeslat hodnotu přístroji
    onAccepted: sendVar(f1Text.text,"f1")
}
```

Například zde zobrazený kód popisuje textové pole pro nastavování frekvence. Pokud DataPlotter obdrží zprávu `$$Vf1:123,5;` změní hodnotu proměnné `f1` na 123,5 a ta se zároveň také zobrazí v textovém poli `f1Text`. A naopak pokud uživatel zadá do textového pole hodnotu 2000 a potvrdí Enterem, odešle funkce `sendVar()` mikrokontroléru zprávu `##Vf1:2000;` .

3.4.5 Přímý vstup dat do QML

```
$$D(data)\0
```

V navrženém uživatelském rozhraní jsou tyto zprávy využity pro signalizaci aktivního připojení. Mikrokontrolér pravidelně posílá zprávu „`$$D!!\0`" , kterou JavaScriptová funkce vyhodnotí a při každém jejím přijetí přepne stav indikátoru připojení. Při aktivním připojení tak indikátor pomalu bliká, při ztrátě spojení se zastaví.

3.4.6 Žádost mikrokontroléru o upload souboru

DataPlotter umožňuje mikrokontroléru zažádat si o soubor, který pak uživatel vybere v prohlížeči souborů a který je poté odeslán mikrokontroléru. Mikrokontroléru je nechána možnost určit velikost paketu, na které bude soubor rozdělen, aby je byl mikrokontrolér schopen pojmout do bufferu a včas zpracovat. Může si též určit znak pro označení konce souboru a také, zda je tímto znakem doplněna zpráva na zadanou velikost, či je tímto znakem poslední zpráva ukončena a její délka může být kratší, než zadaná.

Upload souboru začíná žádostí mikrokontroléru, ve které je specifikována délka paketu a zakončení.

```
$$Rnew,(velikost paketu),(zakončení);
```

Velikost paketu maximální délka jednoho paketu zadána jako číslo ve formě stringu nebo string „all“ pro odeslání celého souboru.

Zakončení jeden z následujících stringů:

„0“ zakončení znakem 0x00

„EOT“ znak End Of Transmission – 0x04

„EOF“ znak End Of File

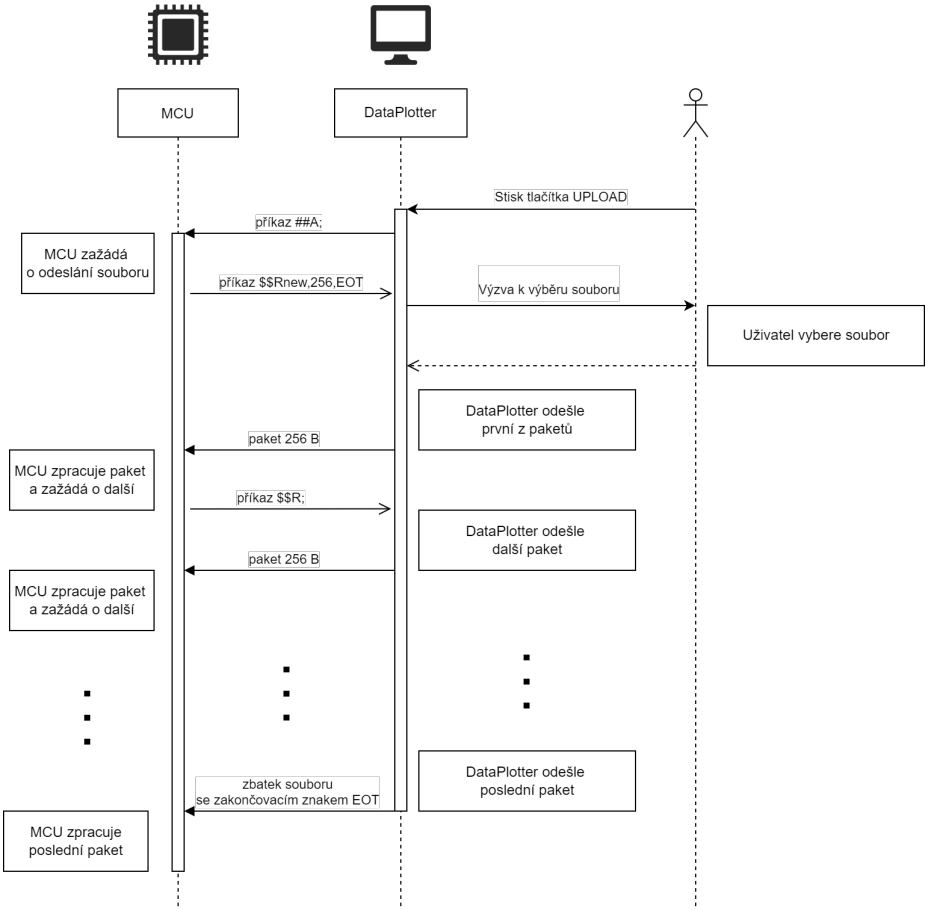
„SEMIC“ znak středníku „;“

„DOLLAR“ znak dolaru „$“

„LF“ znak Line Feed „\n“

„CR“ znak Carriage Return „\r“

Po této zprávě je uživatel DataPlotteru vyzván k výběru souboru k odeslání a následně je odeslán jeho první paket. Pro zaslání dalšího paketu po zpracování předchozího musí mikrokontrolér zažádat příkazem `$$R;` . Celá tato sekvence je znázorněna na následujícím diagramu 3.18.



Obr. 3.18: Sekvence zpráv pro upload LUT

3.4.7 Echo zpráva

Protože ovládací panel tohoto přístroje je dostupný až po nahrání QML souboru, musí mít mikrokontrolér způsob, jak zjistit, že DataPlotter byl připojen, a že je třeba odeslat QML soubor. Stejně tak je třeba detekovat ztrátu spojení. Proto mikrokontrolér pravidelně (s frekvencí 2 Hz danou časovačem TIM3) odesílá tzv. „heartbeat“ zprávu, kterou zjišťuje, zda je připojen DataPlotter. Tato zpráva vypadá následovně:

```
$$E##H
```

Příkaz `$$E` je Echo příkaz, který vrací odesílateli zbytek zprávy jako ozvěna. V příkladu výše DataPlotter odesílá zpět mikrokontroléru příkaz `##H`, díky kterému mikrokontrolér pozná, že DataPlotter je připojen.

3.5 Komunikační protokol mezi mikrokontrolérem a DataPlotterem

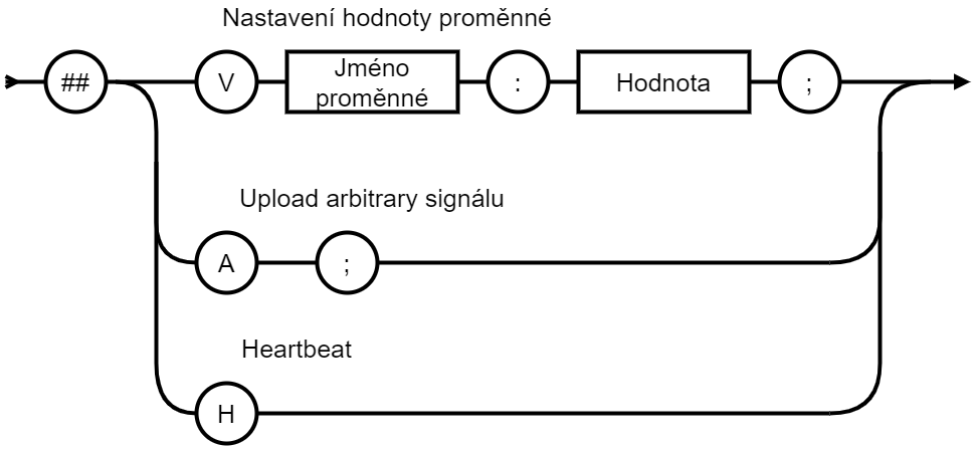
Pro komunikaci opačným směrem, tedy z DataPlotteru na PC do mikrokontroléru bylo třeba navrhnout vlastní protokol. Pro jednoduchost byla zachována základní struktura příkazů, jakou

používá DataPlotter. Znázornění struktury podporovaných zpráv je na obrázku 3.19. Všechny příkazy musí začínat dvojicí znaků `##` a všechny ostatní znaky jsou až do nalezení prvního výskytu této dvojice ignorovány. Následuje jeden znak označující zprávu jako jednu ze tří možných:

##V - změna hodnoty proměnné

##A - žádost o nahrání LUT arbitrary signálu

##H - heartbeat zpráva



Obr. 3.19: Struktura jednotlivých příkazů pro mikrokontrolér

3.5.1 Nastavení hodnoty proměnné

Na tomto příkazu je založeno odesílání jak všech změn nastavených parametrů, tak například kliknutí na tlačítko. Syntaxe tohoto příkazu je stejná, jako pro komunikaci opačným směrem. Jméno proměnné i hodnota jsou přenášeny jako string.

```
\#\#V:(jméno proměnné):(hodnota);
```

3.5.2 Žádost o upload „arbitrary“ signálu

Příkaz `##A;` informuje mikrokontrolér o tom, že uživatel klikl na tlačítko k nahrání uživatelského průběhu signálu. Protože parametry jako velikost paketu a zakončení souboru jsou v režii mikrokontroléru, musí poté mikrokontrolér o odeslání souboru sám zažádat příkazem `$$Rnew;`, který je popsán výše.

Přijatý paket souboru je předán funkci `CSV_Parse`, která se stará o dekódování uživatelského signálu v přijatém .csv souboru. V případě, že paket byl úspěšně dekódován a nebyl zakončen zakončovacím znakem, požádá mikrokontrolér o další paket příkazem `$$R;`.

3.5.3 Heartbeat zpráva

S touto zprávou souvisí mechanismus detekce připojení DataPlotteru a ztráty připojení. Při přijetí zprávy `##H` mikrokontrolér ví, že DataPlotter je připojen, protože tato zpráva je odpovědí na Echo zprávu `$$E##H`, kterou pravidelně odesílá. Pokud jde o první přijetí této zprávy, odešle mikrokontrolér QML soubor, aby DataPlotter mohl zobrazit rozhraní.

Spolu s uvedenou Echo zprávou mikrokontrolér pravidelně posílá zprávu `„$$D!!\0"` , kterou JavaScriptová funkce v QML vyhodnotí a při každém jejím přijetí přepne stav indikátoru připojení v uživatelském rozhraní. Při aktivním připojení tak indikátor pomalu bliká, při ztrátě spojení se zastaví.

Pokaždé, když mikrokontrolér odešle Echo zprávu, zvýší hodnotu počitadla odeslaných zpráv bez odpovědi a pokaždé, když obdrží odpověď, toto počitadlo vynuluje. Pokud počitadlo dosáhne hodnoty 4, a tedy mikrokontrolér neobdržel odpověď na předchozí tři Echo zprávy, považuje se DataPlotter za odpojený. V takovém případě při dalším případném obdržení odpovědi mikrokontrolér znovu pošle celý QML soubor a všechny hodnoty proměnných, jako při prvním připojení.

3.5.4 Zobrazování generovaných signálů

Kromě QML grafického rozhraní pro nastavování parametrů přístroje je pro zlepšení interakce uživatele se zařízením využita i oblast grafu průběhů signálů. DataPlotter byl sice navržen pro zobrazování průběhů signálů snímaných osciloskopem, ale nic nebrání tomu zobrazovat v grafu data vypočítaná signálovým generátorem namísto hodnot naměřených osciloskopem.

Vzor signálu, který mikrokontrolér generuje, je vždy k dispozici v paměti RAM a vzhledem k velikosti LUT (4 kB) není problém tento průběh odeslat do DataPlotteru, který tento průběh zobrazí. Uživatel tak má přímo v rozhraní vizuální zpětnou vazbu o tom, jak se nastavené parametry projevují na generovaném signálu.

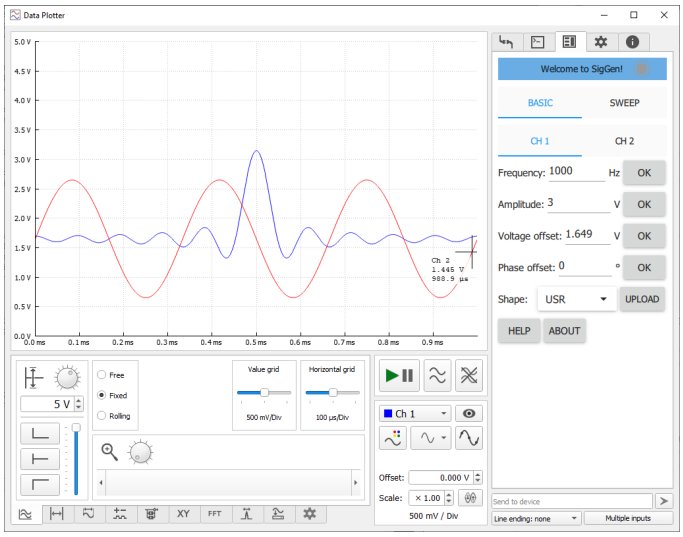
V běžném režimu generování signálů je náhled signálů věrnou reprodukcí signálů, které mikrokontrolér opravdu generuje. Náhled bere v úvahu frekvenci, amplitudu, napěťový i fázový offset a dokonce i počet vzorků na periodu při dané frekvenci (ale maximálně 2048).

V režimu frekvenčního „sweepu“ by věrná reprodukce generovaného signálu vyžadovala příliš mnoho vzorků, protože generovaný signál není periodický. Je tedy zobrazena pouze jedna perioda vybraného tvaru signálu, ale též s přihlédnutím k nastavené amplitudě a napěťovému offsetu. Kromě tvaru generovaného signálu je pro informaci zobrazen i tvar triggerovacího signálu, který je generován na druhém kanálu generátoru.

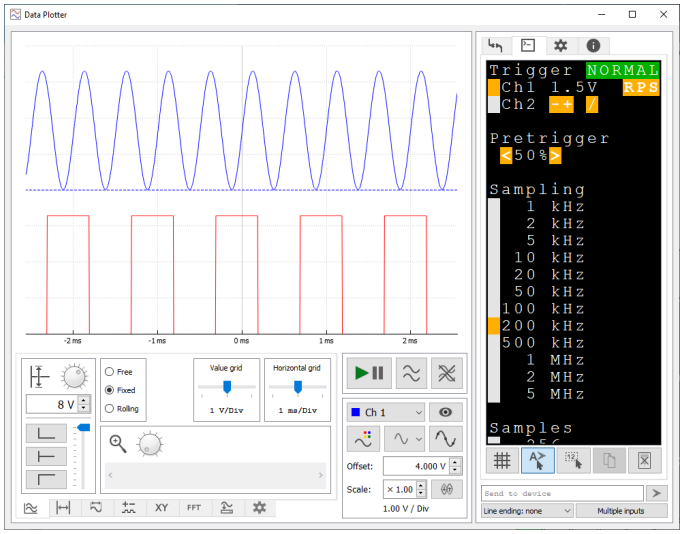
3.5.5 Výsledné rozhraní

Výsledkem kombinace aplikace DataPlotter s implementací vlastního ovládacího panelu pomocí jazyka QML je rozhraní, které by pro uživatele PC mělo být intuitivní, jednoduché, čisté a vizuálně v souladu se zbytkem aplikace. Ve srovnání s předchozími realizacemi terminálového rozhraní je ovládání přímočařejší, zvláště díky možnosti využít myš a plnohodnotné grafické rozhraní. Následující obrázky 3.20 a 3.21 porovnávají výsledek této práce a TUI rozhraní osciloskopu vytvořeného společně s aplikací DataPlotter. Tímto bych chtěl poděkovat panu Jiřímu Maierovi za

jeho práci a za to, že takto zpřístupnil tuto část jeho aplikace a umožnil tak ostatním založit na ní další práce.



Obr. 3.20: Snímek rozhraní navrženého signálového generátoru



Obr. 3.21: Snímek obrazovky s TUI rozhraním pro osciloskop navržený v [1]

3.6 Firmware přístroje

Jak vyplývá ze zkratky SDI, která vystihuje navrhovaný přístroj, je třeba nejvíce úsilí směřovat na vývoj firmwaru, protože jen ten z hardwaru udělá komplexní přístroj. Úkolem firmwaru tohoto zařízení je vhodně nakonfigurovat periferie mikrokontroléru a využít možností jejich propojení, dále provádět výpočty spojené s generováním signálu a v neposlední řadě také vytvářet rozhraní pro interakci s uživatelem. V následujících kapitolách je detailněji popsána funkce hlavních částí programu a jejich realizace.

3.6.1 Výběr IDE

Při výběru vývojového prostředí pro vývoj firmwaru na použité mikrokontroléry z rodiny STM32 od firmy ST Microelectronics padla volba na framework STM32Cube a jeho vývojové prostředí STM32CubeIDE, nabízené přímo firmou ST Microelectronics. Jde o zdarma dostupné vývojové prostředí pro vývoj aplikací pro mikrokontroléry STM32 v jazycích C, C++ a Assembler. Navzdory tomu, že je nabízeno zadarmo, nejsou jeho funkce nijak omezeny, ani maximální velikostí kompilovaného programu, ani znepřístupněním některých funkcí. Nabízí také kompletní podporu pro debugování firmwaru mikrokontroléru a v neposlední řadě také propojení s dalšími programy frameworku STM32Cube, například STM32CubeMX nebo STM32CubeProgrammer. STM32CubeMX je program nabízející přehledné grafické rozhraní pro konfiguraci mikrokontroléru a jeho periferií a následné automatické generování kódu pro jejich nastavení a obsluhu, což i zkušenějšímu programátorovi ušetří čas strávený studováním jmen registrů a psaním vlastních knihoven pro jejich obsluhu. I tak je ale pro správné použití potřeba znát funkci daných periferií a jejich závislosti na ostatních částech mikrokontroléru. Kromě toho, že STM32CubeIDE je kompletní a zadarmo nabízené prostředí, bylo vybráno také pro předchozí zkušenosti z absolvovaných předmětů ve škole. Tato práce však předchozí znalosti prohloubila mnohem více.

Jako jazyk vyvíjeného firmwaru byl vybrán jazyk C s případnou implementací kritických částí kódu v Assembleru. Použití Assembleru se však nakonec neukázalo jako nutné, protože oba mikrokontroléry dokázaly splnit požadavky na rychlost i bez takovéto manuální optimalizace. Kvůli počátečním obavám o nedostatečný výpočetní výkon mikrokontroléru STM32F303 též nebylo ani uvažováno použití RTOS, i když nakonec mají oba mikrokontroléry ještě výkonovou rezervu.

HAL knihovny

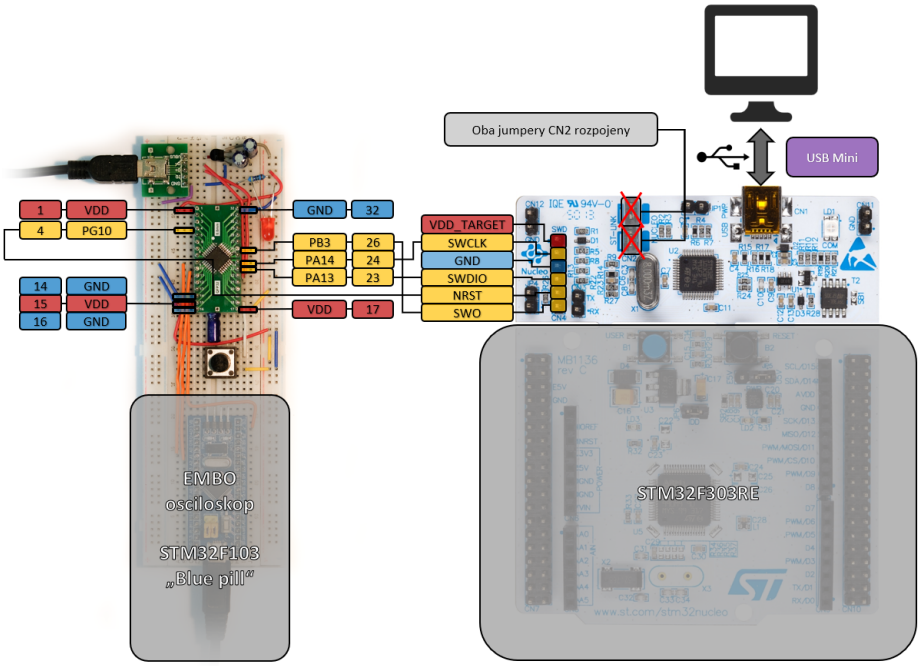
Samotný navrhovaný program je založen na knihovnách HAL (Hardware Abstraction Layer), které generuje program STM32CubeMX, a které umožňují oddělit program aplikace od přímého interagování s registry mikrokontroléru. Interakci s hardwarem, který se může mezi jednotlivými rodinami mikrokontrolérů mírně lišit, zajišťují drivery, které tyto rozdíly berou v potaz a navenek nabízí jednotné API. Toto by mělo usnadnit přenášení aplikace na jiné mikrokontroléry, protože většina samotné aplikace může zůstat nezměněná a například rozdíly v nastavování hodinových signálů pro periferie vyřeší drivery pro daný mikrokontrolér. Protože již od začátku práce na tomto přístroji bylo jasné, že bude potřeba, aby tato aplikace běžela na více rodinách mikrokontrolérů STM32, byly vybrány knihovny HAL namísto přímé práce s registry.

Nevýhodou tohoto přístupu je fakt, že HAL knihovny už ze své podstaty musí být psány naprosto univerzálně, aby bez nutnosti programátorova zásahu do jejich kódu obsáhly všechny možnosti interakce. To se sice neslučuje s požadavkem na co nejvyšší rychlost programu, ale i tak byla nakonec rychlostní omezení způsobená generovanými HAL knihovnami odstraněna.

Programování mikrokontroléru

Jako prostředník mezi PC, na kterém běží framework STM32Cube a mezi mikrokontrolérem byl použit ST-LINK/V2-1, který je integrován na desce Nucleo-F303RE. Programátor ST-LINK zde slouží jako převodník mezi USB a SWD (Serial Wire Debugging). Pomocí této Nucleo desky lze programovat jak mikrokontrolér na desce, tak po přepojení dvou propojek a připojení SWD signálů i jakýkoliv jiný mikrokontrolér STM32. Příkladem může být použitý mikrokontolér

STM32G431KBT6, jehož propojení s programátorem na desce Nucleo je vidět na obrázku 3.22. Nedílnou součástí vývoje programů je hledání chyb a ladění, neboli debugging. Debugování programu na mikrokontroléru umožňuje ST-LINK pomocí vkládání breakpointů, krokování programu, vyčítání hodnot registrů nebo sledování hodnot proměnných. Pokud využijeme i signál SWO, můžeme využít funkce SWV (Serial Wire Viewer), díky které lze například sledovat procenta času strávená procesorem v jednotlivých funkcích programu. Toho bylo využito pro optimalizaci rychlosti programu a nalezení rychlostních omezení.



Obr. 3.22: Využití programátoru ST-LINK na desce Nucleo pro programování mikrokontroléru STM32G431

3.6.2 Hlavní cyklus programu

Soubor `main.c` je vstupním bodem programu. Tento soubor je spolu s potřebnými soubory HAL knihoven generován programem STM32CubeMX podle nastavených parametrů (např. frekvence jednotlivých hodinových domén a konfigurace periferií a jejich propojení). Vygenerovaný soubor nabízí prázdnou kostru programu s připravenou inicializací periferií. Kód vyvíjené aplikace je umisťován do vyznačených sekcí mezi komentáře `/* User code begin */` a `/* User code end */`. Upravovat lze samozřejmě celý soubor i vygenerované knihovny, ale pokud programátor během vývoje změní nastavení mikrokontroléru v CubeMX a nechá vygenerovat základnu kódu znovu, jsou kromě vyznačených uživatelských částí všechny změny v kódu ztraceny a nahrazeny nově vygenerovaným kódem.

Po zapnutí nebo resetu mikrokontroléru je ještě před vstupem do hlavního programu spuštěn program `startup.s`. Ten nejdříve inicializuje hodnotu Stack pointer a Program counter a tabulku vektorů obsluh přerušení, poté provede základní nastavení hodin a načte potřebné hodnoty proměnných z Flash do RAM a nakonec zavolá funkci `main()`. Funkce `main()` nejdříve vykoná vygenerované inicializace hodin a periferií a tím končí inicializační část připravená frameworkem STM32CubeMX.

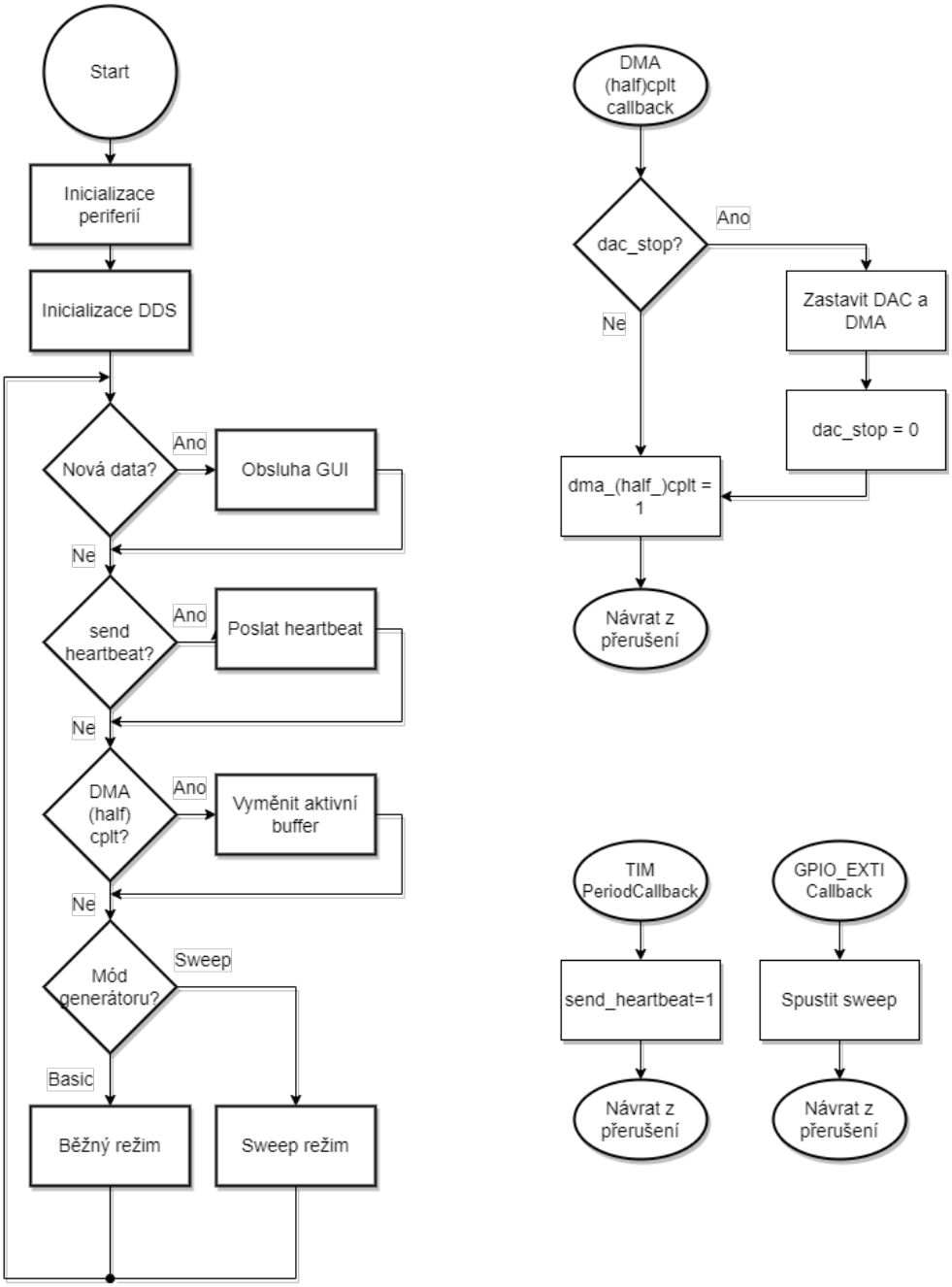
Následuje inicializace vlastní knihovny DDS a nastavení prvotních parametrů generovaných signálů, spuštění periferií TIM6, TIM3, USB/UART a začátek přenosu hodnot pro DAC pomocí DMA. Tím je dokončena veškerá potřebná inicializace a program vstupuje do hlavní smyčky, ze které již kromě obsluhy přerušení nevystoupí. Průchod programem je znázorněn vývojovým diagramem na obrázku 3.23.

V hlavní smyčce programu jsou pravidelně kontrolovány příznaky, které indikují, zda se stala některá událost, na kterou je třeba reagovat. První kontrolovanou událostí je příjem dat přes USB nebo UART. Pokud od poslední kontroly přišly nové znaky, je funkci `GUI_ParseInput` předán ukazatel na buffer přijatých znaků spolu s počtem přijatých znaků a funkce tyto znaky rozklíčuje. Nejefektivnější by bylo zpracovávat přijatá dat po celých zprávách, ovšem protože délka zprávy není mikrokontroléru předem známa a počet přijatých znaků může být při kontrole libovolný, provádí se zpracování dat i po přijetí jediného bajtu.

V pořadí druhou událostí je pravidelné odesílání tzv. heartbeatu. Toto slouží k ověření, zda je zařízení připojeno k DataPlotteru. Pokud uběhl čas určený periodou časovače TIM3 (0,5 s), je zavolána funkce `GUI_Heartbeat`, která odešle DataPlotteru heartbeat zprávu. Tento mechanismus je podrobněji vysvětlen v kapitole 3.5.3 o komunikačních protokolech.

Další událostí, která pravidelně nastává, je jeden ze dvou callbacků, které spouští modul DMA - `HAL_DAC_ConvCpltCallbackCh1` nebo `HAL_DAC_ConvHalfCpltCallbackCh1`. Tyto callbacky jsou spuštěny vždy, když DMA inkrementováním zdrojové adresy dojde do poloviny nebo na konec bufferu. Pokaždé, když se tak stane, znamená to, že DMA začíná číst z té druhé poloviny bufferu a ta první je tedy připravena na ukládání nových hodnot. Samotné callbacky se vykonávají v obsluze přerušení a nastavují pouze příznak `dma_cplt` nebo `dma_half_cplt`. Úprava hranic bufferu pro ukládání nových hodnot (proměnné `buf_idx` a `buf_idx_end`) a signalizace volného bufferu (proměnná `dma_buffer_full`) je provedena až v hlavní smyčce po kontrole uvedených příznaků.

Po kontrole těchto událostí jsou provedeny případné výpočty nových hodnot signálu. V závislosti na zvoleném režimu se vykoná buďto větev programu pro generování sweepu nebo běžných signálů.



Obr. 3.23: Vývojový diagram celého programu

Program pro běžný režim generátoru

Chod programu v obou režimech je řízen stavovým automatem, který zajišťuje správný začátek generování signálu a přechod mezi běžným režimem a sweepem. V běžném režimu signálového generátoru je tento stavový automat jednoduchý a sestává v podstatě ze tří po sobě jdoucích kroků, jak je znázorněno stavovým a vývojovým diagramem na Obrázku 3.24.

Průchod touto částí programu je řízen proměnnými `dma_buffer_full` a `basic_state`. Příznak `dma_buffer_full`, jak název napovídá, oznamuje, zda je polovina bufferu určená k ukládání hodnot již zaplněna. Pokud je zaplněna, není třeba nic dalšího dělat a zbytek této sekce programu je přeskočen. Proměnná `basic_state` určuje ve kterém stavu stavového automatu se program nachází a může nabývat hodnot `Init, Start` nebo `Run`.

Na začátku je tato proměnná ve stavu `Init`. Při prvním průchodu je tedy vykonána větev `Init`, která zastaví DAC a DMA, vymaže případné hodnoty v bufferu a nastaví hodnotu `basic_state` na `Start`. Poté se řízení vrací zpět na začátek dříve popsané hlavní smyčky, kde se provede kontrola a případná obsluha událostí. Nakonec řízení opět vstoupí do této sekce programu.

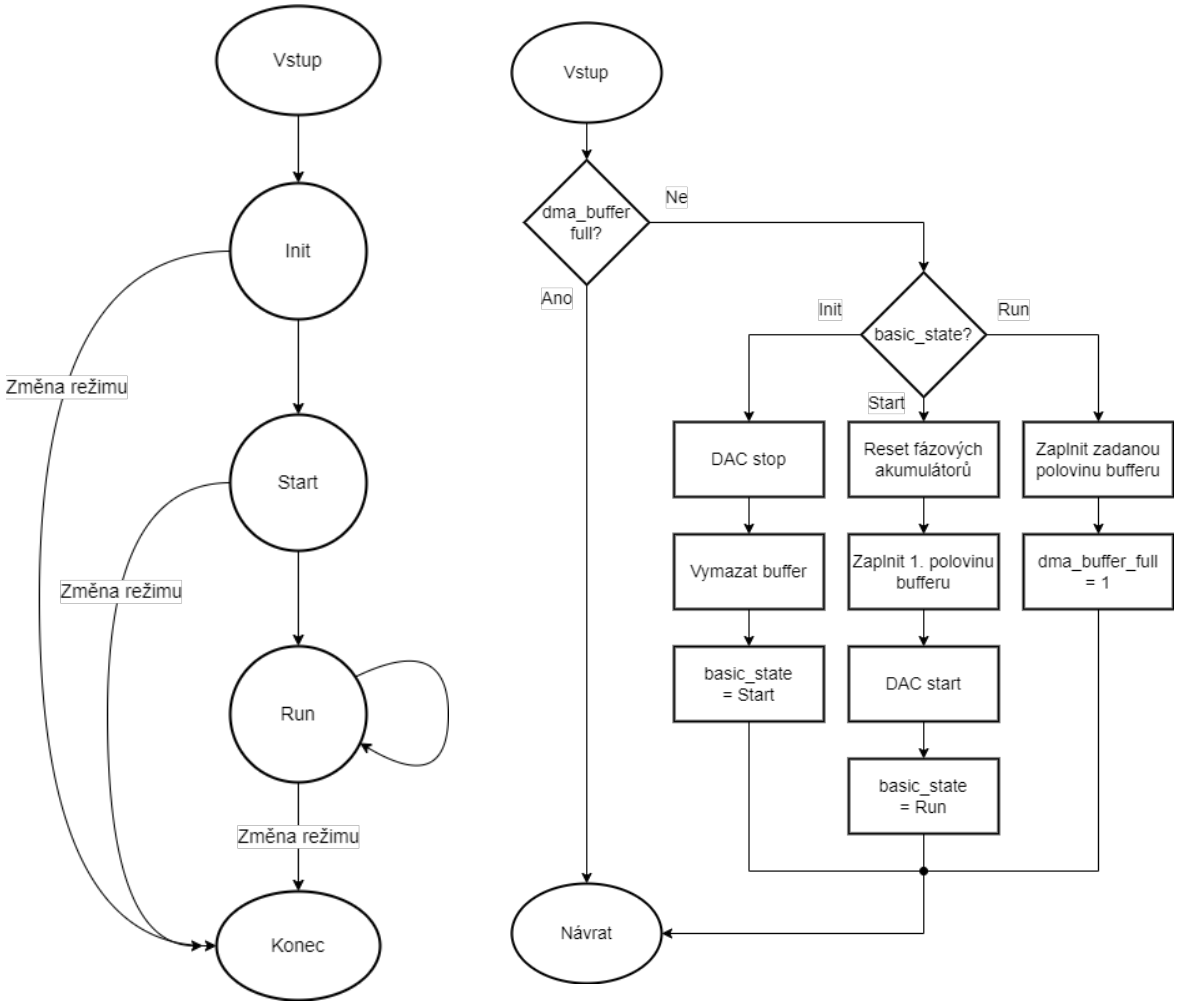
Při druhém průchodu je již proměnná `basic_state` na hodnotě `Start` a je tedy vykonána odpovídající větev programu. V ní je resetována hodnota fázových akumulátorů, dále je zaplněna první polovina bufferu hodnotami signálu, a nakonec je nastavena hodnota `basic_state` na `Run`.

V dalším průchodu se vykoná větev Run, která pomocí funkce `DDS_CalculateHalfBuffer` vypočítá budoucí hodnoty signálu a zaplní jimi jednu polovinu bufferu. Tato polovina je zadána proměnnými `buf_idx` a `buf_idx_end`, které jsou vždy aktualizovány v reakci na callbacky DMA `HAL_DAC_ConvCpltCallbackCh1` a `HAL_DAC_ConvHalfCpltCallbackCh1`. Po zaplnění dané poloviny bufferu hodnotami je nastaven příznak `dma_buffer_full`. V dalších průchodech bude díky tomuto příznaku tato sekce programu přeskakována, dokud tento příznak nebude v reakci na jeden z callbacků DMA znovu vynulován. Ve větvi `Run` se nemění hodnota proměnné `basic_state` a program tak v každém dalším průchodu vykonává pouze tuto větev. Ke změně stavu může dojít pouze externě, v obsluze GUI, pokud uživatel zadá příkaz ke změně režimu na sweep.

Program pro režim sweep

Režim sweep je, co se týče toku programu, odlišný v tom, že sweep je možno spustit triggerem a ukončit po dosažení požadovaného počtu opakování. V rozhraní lze nastavit pět typů triggerování: `None, Rising, Falling, Both` a `Auto`. V režimu None je jediným způsobem spuštění tlačítko GO v grafickém rozhraní. V režimech `Rising, Falling` a `Both` lze spustit sweep kromě tlačítka GO také zvolenou hranou (hranami) na pinu PA8. Nakonec režim `Auto` znamená, že sweep je automaticky opakován donekonečna. Možnost zastavit a spustit sweep po určitém počtu opakování si vyžaduje o něco složitější stavový automat. Průchod tímto stavovým automatem je řízen proměnnými `sweep_state`, `sweep_type`, `dma_buffer_full` a `burst_rep_counter`.

Ve stavu `Init` je nejprve zastaveno generování signálu, poté jsou vypočítány parametry sweepu (počáteční a koncový inkrement fáze, celkový počet vzorků), následně jsou připraveny LUT s generovaným signálem a se signálem pro triggerování externího osciloskopu. Nakonec je resetován fázový akumulátor a počitadlo vypočítaných vzorků signálu. Přechod do dalšího stavu je automatický a bez podmínek.

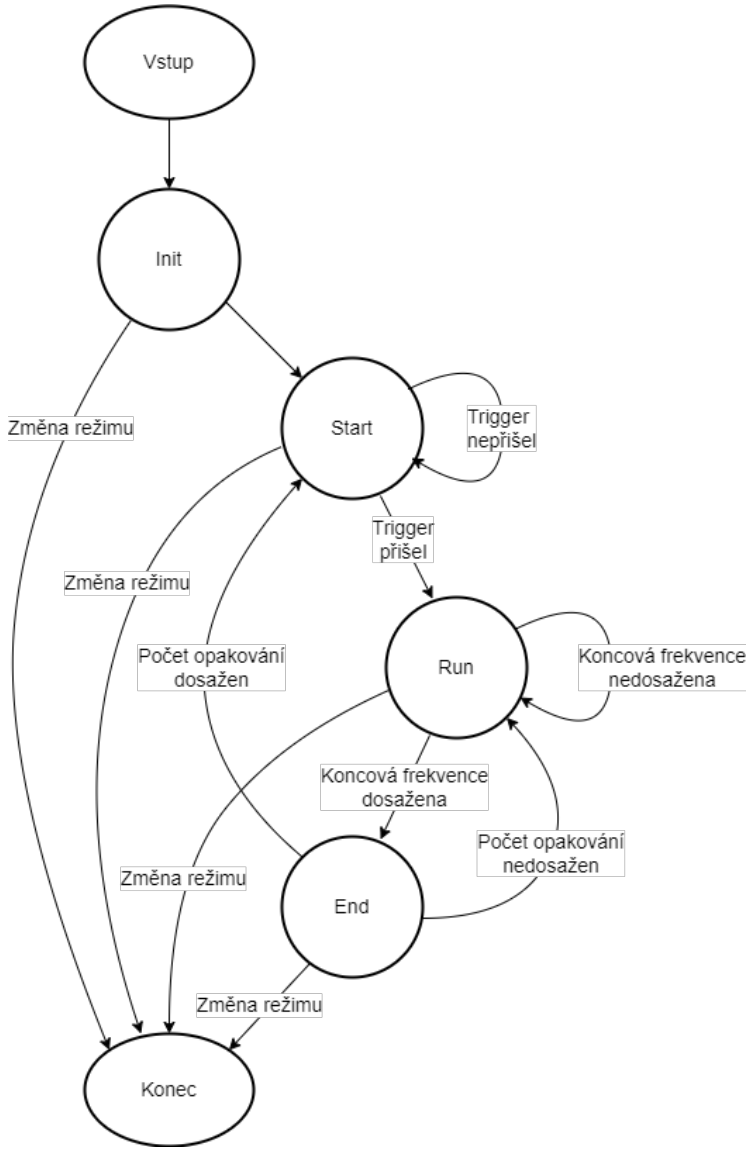


Obr. 3.24: Stavový (vlevo) a vývojový (vpravo) diagram běžného režimu přístroje

Ve stavu `Start` program čeká na trigger. Dokud se tak nestane, neděje se v tomto stavu nic a řízení je ihned předáno zpět hlavní smyčce. V okamžiku, kdy je obdržen trigger, je po zaplnění bufferu hodnotami aktivováno DAC a DMA. Jedině tímto lze přejít do stavu `Run`.

Ve stavu `Run` je nejprve zkontrolováno, zda je již buffer zaplněn hodnotami pomocí příznaku `dma_buffer_full`. V případě, že tento příznak není nastaven a je tedy možno do bufferu zapisovat, je zaplněna polovina bufferu novými hodnotami signálu. Poté je korigována chyba frekvence pomocí `DDS_CorrectFrequencyMidsweep` a nakonec je zkontrolováno, zda již byla dosažena požadovaná koncová frekvence. Pokud byla dosažena, přechází se do stavu `End`.

Ze stavu `End` jsou možné přechody do dvou dalších stavů. Pokud ještě nebyl proveden požadovaný počet sweepů, je pouze resetována fáze a počitadlo vypočítaných vzorků a poté je program navrácen do stavu `Run`, kde je pokračováno v generování dalšího sweepu. V případě že právě proběhl poslední sweep, je nastaven příznak `dac_stop`, čímž je zajištěno, že po zavolání dalšího DMA callbacku bude DAC a DMA vypnuto. Řízení se pak vrací do stavu `Start`, kde program čeká na další trigger. Celý tento mechanismus je znázorněn stavovým diagramem na obrázku 3.25.



Obr. 3.25: Stavový diagram režimu sweep

3.6.3 Program algoritmu DDS

Princip implementace DDS algoritmu a dvojitého bufferování již zde byl popsán v obecné rovině. Z pohledu struktury programu jsou pro DDS algoritmus stěžejní funkce `DDS_LUT_Precompute` a `DDS_CalculateHalfBuffer`, a její obdoby pro sweep režim `DDS_CalculateSweepLinHalfBuffer` a `DDS_CalculateSweepLogHalfBuffer`.

První zmíněná funkce `DDS_LUT_Precompute` je vykonávána po každé změně tvaru signálu. Jejím úkolem je na základním tvaru signálu provést všechny operace, které je zbytečné provádět v každém kroku znovu. Z Flash paměti jsou načítány hodnoty vybraného základního tvaru signálu, je jim upravena amplituda, offset a jejich hodnota je omezena na maximální hodnoty DAC. Výsledné hodnoty jsou uloženy v RAM v `LUT1_RAM` a `LUT2_RAM`.

Na funkci `DDS_CalculateHalfBuffer` a její obdoby pro sweep zbývá pouze inkrementovat fázi a vybírat dle ní správné hodnoty. Tyto hodnoty vybrané z `LUT1_RAM` nebo `LUT2_RAM` jsou ukládány

do bufferu `dac_values`. Děje se tak v cyklu, který má předem určený počet opakování pomocí počátečního a koncového indexu pro ukládání v bufferu. Na pozadí celého programu tato data DMA vyčítá a přenáší do registru DAC.

Kromě zmíněných zásadních funkcí byly implementovány i funkce pro nastavování jednotlivých parametrů, jako například frekvence, fázového a napěťového offsetu nebo počtu opakování sweepu. Ke všem těmto „setterům“ existují i „wrapper“ funkce, které je adaptují pro volání z obsluhy GUI. Tyto funkce mají vždy stejné jméno, ale s příponou `_GUI`. Například funkce `DDS_SetFreq_GUI` přijímá jako argument univerzální strukturu typu `GUI_var_t`, vybere z ní potřebné parametry, zkontroluje jejich validitu a nakonec s těmito parametry zavolá funkci `DDS_SetFreq`, která uloží hodnotu frekvence a vykoná výpočet inkrementu.

3.6.4 Program obsluhy GUI

Funkce `GUI_ParseInput` přijímá jako argumenty pointer na buffer s přijatými daty a počet přijatých bajtů. K rozklíčování, o který příkaz jde dochází po jednotlivých znacích pomocí stavového automatu. Nejprve je nalezena dvojice znaků `##`. Poté je dle následujícího znaku určen příkaz a podle něj probíhá další zpracování dat. Příjem „arbitrary“ signálu i „heartbeat“ zpráva již byly rozebrány v kapitole o komunikačních protokolech.

Propojení DDS algoritmu a GUI je uskutečněno skrze nastavování hodnot proměnných příkazem `##V` z DataPlotteru. Každý ovládací prvek v QML rozhraní má své jméno a s ním v mikrokontroléru spojenou strukturu uchovávající informace jako typ proměnné, „setter“ funkce (nqpříklad `DDS_SetFreq_GUI`) a „getter“ funkce (`DDS_GetFreq_GUI`). Po přijetí `##V` zprávy je rozklíčován název proměnné, který je porovnán se seznamem názvů proměnných. Pokud je nalezena shoda, uloží se index v seznamu a nalezne se přiřazená struktura. Poté je s pomocí zjištěného typu proměnné rozklíčována nová hodnota proměnné a nakonec je s touto hodnotou zavolána příslušná „setter“ funkce. Po nastavení nové hodnoty je DataPlotteru pro jistotu poslána právě nastavená hodnota k zobrazení (kdyby byla nastavovaná hodnota „setter“ funkcí omezena).

3.6.5 Příjem arbitrary signálu

Tento přístroj nabízí uživateli možnost nahrát vlastní libovolný signál, pokud si nevybere ze tří základních (sinus, trojúhelník, rampa). Tento signál lze nahrát jako .csv (Comma Separated Values) soubor. Tento soubor se musí řídit následujícími pravidly:

- Soubor musí začínat buďto rovnou hodnotami nebo jedním řádkem popisů sloupců
- Hodnoty musí být ve sloupcích
- Pokud popisy sloupců obsahují názvy „index“ nebo „value“
  - Odpovídající sloupce jsou použity jako indexy a hodnoty
- Pokud popisy sloupců nejsou rozeznány jako „index“ ani „value“ nebo nejsou popisy přítomny vůbec
  - Pokud je sloupců více
    - První sloupec je brán jako indexy hodnot
    - Druhý sloupec je brán jako hodnoty
    - Ostatní sloupce jsou ignorovány
  - Pokud je sloupec jen jeden
    - Tento sloupec je brán jako hodnoty s po sobě jdoucími indexy
    - Pokud je hodnot méně, než 2048, je zbytek doplněn nulami
- Maximální délka signálu: 2048 vzorků (maximální index: 2047)
- Hodnoty signálu
  - Maximální hodnota: 2047
  - Minimální hodnota: -2048
  - Hodnoty mimo tento rozsah jsou omezeny na tyto hodnoty
  - Desetinná čísla jsou povolena, ale část za desetinnou čárkou je zahozena
- Pokud je dostupný sloupec s indexy hodnot, je možno využít lineární interpolace mezi hodnotami. Jestliže se dva po sobě jdoucí indexy liší o více než 1, jsou chybějící hodnoty interpolovány.

Uživatel signálového generátoru však nemusí studovat pravidla návrhu .csv souboru. Pro jednoduchý návrh a následné exportování signálů byly vytvořeny pomocné šablony pro programy Microsoft Excel a Matlab.

3.6.6 Dekódování CSV souboru

Z přijatého .csv souboru je třeba dekódovat hodnoty signálu a jim přiřazené indexy v LUT. Pro tento účel byla napsána vlastní funkce pro postupné dekódování souboru a ukládání získaných hodnot. Protože soubor přichází po paketech, bylo třeba implementovat tuto funkci tak, aby bylo možno z ní po dosažení konce paketu vystoupit a po přijetí dalšího paketu pokračovat, kde skončila. Výsledná implementace této funkce zpracovává data po jednom bajtu a je tedy možno ji přerušit po jakékoliv délce dat. Stavový automat této funkce se skládá z následujících stavů:

- `CSV_START`
  - Přeskočí případné nealfanumerické znaky na začátku souboru (například označení kódování)
- `CSV_READ_FIELD`
  - Po jednom kopíruje znaky do pomocného pole, dokud nenarazí na „;“, „\r“ nebo „\n“
- `CSV_PARSE_HEADER`
  - Hledá v prvním řádku string „index“ a „value“
  - Nastavuje, ze kterých sloupců jsou brány hodnoty a indexy (pokud jsou přítomny)
- `CSV_PARSE_FIELD`
  - Převede zkopírovaný string na číselnou hodnotu a uloží ji jako index nebo hodnotu podle aktuálního sloupce
  - Poté co v aktuálním řádku získá index i hodnotu provede případnou interpolaci a uloží hodnotu(-y) do Flash paměti
- `CSV_END`
  - Vrátí všechny použité proměnné na začáteční hodnotu
  - Zpracuje všechny koncové znaky.

3.6.7 Ukládání do Flash paměti

Přijatý arbitrary signál je automaticky ukládán do Flash paměti mikrokontroléru, aby ho nemusel uživatel znovu nahrávat po každém zapnutí přístroje. Pro LUT uživatelského signálu je v paměti pomocí úpravy linker skriptu vyhrazen prostor o velikosti 4096 Bajtů začínající na adrese 0x0801F000.

Přístup k Flash paměti je specifický tím, že ji lze mazat a zapisovat do ní pouze po větších blocích. Před zápisem do Flash paměti je vždy třeba nejprve dotčenou oblast vymazat. Nejmenší samostatně vymazatelný blok paměti Flash je jedna stránka, což činí 2048 Bajtů. Velikost LUT tedy odpovídá přesně dvěma stránkám, a proto je také adresa zarovnána na hranice stránek. Obě stránky jsou vymazány vždy, když se mikrokontrolér připravuje na přijetí uživatelského signálu.

Po vymazání paměti a přijetí a dekódování prvních hodnot je možno začít ukládat do Flash paměti. Zapisovat lze pouze po blocích o velikosti 64 bitů, a tak jsou dekódované 16bitové hodnoty vždy nejdříve ukládány do dočasného pole v RAM a poté vždy po čtyřech ukládány najednou.

3.6.8 USB

Mikrokontrolér STM32G431 disponuje hardwarovým modulem pro USB, který podporuje rychlostní standard Full Speed (teoretické maximum 12 Mb/s či 1,5 MB/s). Díky tomu se dá mikrokontrolér připojit přímo k PC bez nutnosti převodníku, jako je tomu u UART, a navíc je toto připojení znatelně rychlejší. Pro obsluhu USB komunikace na mikrokontroléru byla použita výrobcem nabízená knihovna USB Device pro třídu zařízení CDC (Communications Device Class). Díky tomu se po

připojení k PC mikrokontrolér automaticky enumeruje jako virtuální COM port (VCP) a na straně PC s ním lze pracovat stejně, jako se sériovým COM portem (UART). Použitá knihovna nabízí API v podobě funkcí `CDC_Transmit_FS`, `CDC_Receive_FS` a `CDC_TransmitCplt_FS`. Předávání dat mezi těmito funkcemi a aplikací funguje na základě předávání ukazatele na buffer a počtu bajtů, které obsahuje.

Odesílání dat přes USB

Buffer pro data k odeslání je implementován podobně jako buffer hodnot pro DAC, tedy je rozdělen na dvě poloviny. Zatímco jedna polovina bufferu je předána k odeslání, do druhé poloviny je mezitím stále možno zapisovat další data k odeslání a program se tak nemusí zastavovat. Jen v případě, že zapisovaná polovina bufferu je zaplněna dříve, než je odeslána minulá polovina, počká program na dokončení odesílání, aby se předešlo přepsání dat v právě odesílaném bufferu. Jinak není program odesíláním nijak blokován. Pro zápis dat k odeslání do bufferu byly do API poskytovaného knihovnou `usbd_cdc` přidány funkce `CDC_Print` a `CDC_SendBuffer`, které zajišťují zápis do správné poloviny bufferu a její následné odeslání. Odeslat polovinu bufferu je možno buď pomocí funkce `CDC_SendBuffer` anebo je tak učiněno automaticky ve funkci `CDC_Print` po zaplnění dané poloviny.

Příjem dat přes USB

Pro přijatá data je vyhrazen buffer o více než dostačující velikosti 256 B. Z pohledu zpracování dat na mikrokontroléru je vítanou vlastností USB možnost odmítat po přijetí zprávy další data, dokud to není znovu povoleno. Tím je zajištěno, že mikrokontrolér vždy stihne data zpracovat než přijdou další.

3.6.9 UART

Druhý používaný mikrokontrolér, STM32F303RE na desce Nucleo, sice také disponuje nativním USB rozhraním, ale to je bohužel vyvedeno pouze na hřebenový konektor a je tedy velmi nepraktické jej používat. Namísto toho se u této desky počítá s připojením pomocí portu USART2, který je připojen k vestavěnému ST-LINKu. Ten kromě funkce programátoru a debuggeru také zastává funkci USB-UART převodníku a na PC se tedy též enumeruje jako virtuální COM port. Nevýhodou tohoto přístupu je omezená rychlost připojení ve srovnání s USB. S běžnou rychlostí 115200 baud/s je teoretické maximum 11,52 kB/s a při maximální rychlosti podporované DataPlotterem (2250000 baud/s) je to 225kB/s.

Vývoj programu probíhal nejdříve na mikrokontroléru STM32G431 a až později byl přenesen na STM32F303. Pro zjednodušení přechodu byla snaha ponechat API zprostředkovávající komunikaci co možná nejpodobnější, ale nakonec bylo třeba pro efektivní využití UART toto rozhraní pozměnit.

Příjem dat přes UART

Nejefektivnějším způsobem, jak přijímat data skrze UART je s použitím DMA, které zajistí uložení příchozích dat bez zásahu procesoru. Jinak by musel procesor každý přijatý bajt uložit v obsluze přerušení. Rozhraní UART ale na rozdíl od USB neumožňuje pozastavit příjem dat, a tak bylo třeba upravit způsob jejich ukládání. Zatímco přijímací buffer USB ukládá vždy jeden paket, který je zpracován před příchodem dalšího, UART přijímá kontinuálně, a proto jsou data pomocí DMA ukládána do kruhového bufferu. Během kontroly událostí v hlavní smyčce je z rozdílu ukazatele DMA oproti minulé kontrole vypočítán počet přijatých bajtů a společně s ukazatelem na začátek nově přijatých dat je předán ke zpracování obsluze GUI. Jde pravděpodobně o netradiční řešení, ale zdá se být spolehlivé a nevyžaduje žádná přerušení.

Odesílání dat přes UART

Odesílání dat je implementováno shodně s USB, jen s tím rozdílem, že buffer je odeslán pomocí DMA přes UART. Nečekaným poznatkem zde bylo, že na rozdíl od USB a oproti očekávání není po dokončení odesílání vyvoláno přerušení, které by o tom program informovalo.

Kapitola 4

Zhodnocení výsledků

Za výsledek této práce se dají považovat dvě věci. V prvé řadě je to implementace signálového generátoru jako Software Defined přístroje na mikrokontrolérech STM32 s využitím DDS algoritmu. Druhým výsledkem je prostudování možností způsobu navržení ovládacích panelů pro přístroje SDI pomocí jazyka QML a s tím osvětlení nových možností ovládání pro budoucí podobné SDI.

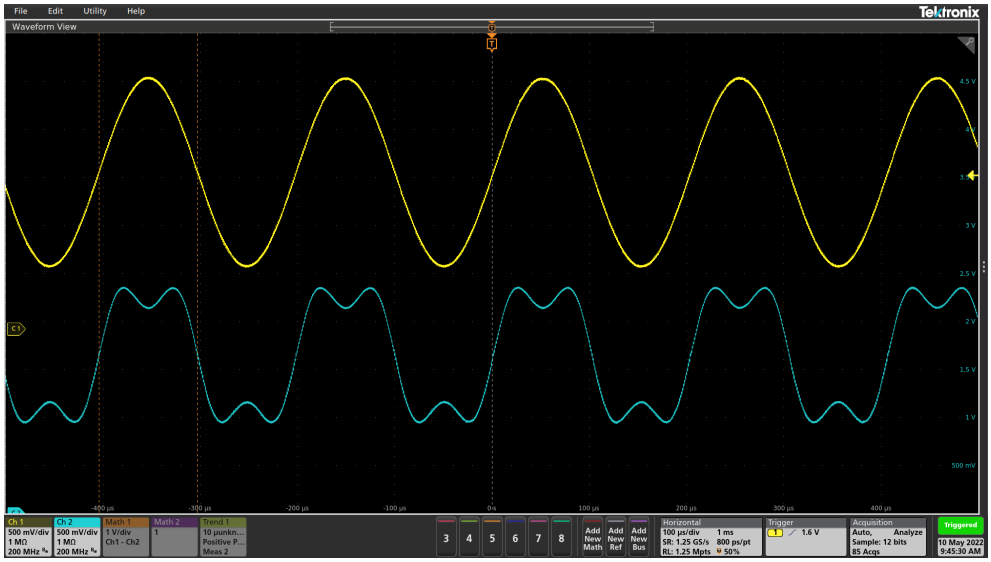
4.1 Implementace signálového generátoru

Navržený signálový generátor umožňuje generovat periodické signály ve dvou režimech: běžný režim a frekvenční „sweep“. V běžném režimu generuje dva nezávislé periodické signály. Ve „sweep“ režimu je na prvním kanálu generován signál s logaritmicky či lineárně se měnící frekvencí, zatímco na druhém kanálu je generován trigger signál pro jednoduché triggerování osciloskopu. V obou režimech lze nastavit frekvenci (u sweepu počáteční a koncovou), amplitudu, napěťový offset a tvar signálu, který lze též nahrát jako .csv soubor. V běžném režimu lze též určit fázový posun každého signálu.

Z počátku vývoje byl problém dosáhnout požadované vzorkovací frekvence 1 MHz, ale po nalezení a odstranění problematických částí programu byla tato frekvence s rezervou dosažena i na mikrokontroléru STM32F303, který je více než dvakrát pomalejší než STM32G431. Druhý jmenovaný mikrokontrolér byl dokonce úspěšně testován až do vzorkovací frekvence 4 MHz v režimu sweep a 5,25 MHz v běžném režimu. Přiložený firmware používá vzorkovací frekvenci 1 MHz v souladu se specifikací DAC modulu pro jistotu, že nedochází ke zkreslení například příliš dlouhým časem ustálení, nízkým „slew rate“ či jinými chybami. Pro porovnání je však přiložena i varianta se vzorkovací frekvencí 4 MHz. Zdrojem původních problémů byly příliš univerzální a neoptimalizované funkce pro obsluhu přerušení, které byly součástí HAL knihoven. Tyto funkce byly nejdříve optimalizovány, ale nakonec se jako lepší řešení ukázalo tato přerušení vůbec nepoužívat.

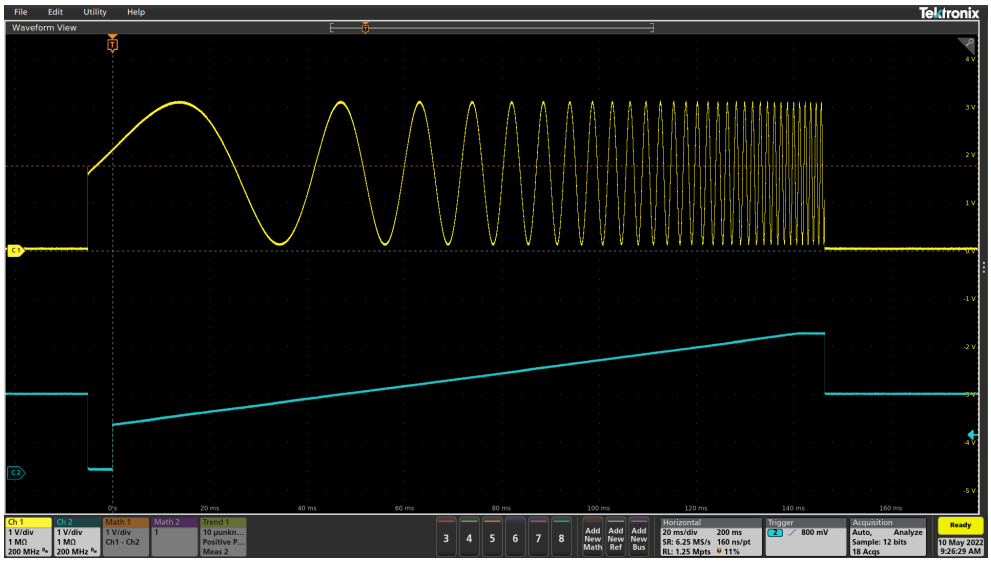
V souvislosti s touto prací byl v rámci předcházejícího semestrálního projektu též navrhnut podobný softwarem definovaný generátor, jehož předností je generování čtyř nezávislých interních signálů, které lze váhovat a sčítat do dvou výstupů. Pouze malá část vyvinutého programu byla využita i v této práci, avšak semestrální projekt dobře posloužil pro osvojení algoritmu DDS a zpracování digitálních signálů na mikrokontroléru.

Následující snímky z osciloskopu zobrazují signály generované signálovým generátorem navrženým v této práci. Obrázek 4.1 ukazuje příklad dvou signálů o frekvenci 5 kHz, z nichž jeden (světle modrý) je uživatelem nahraný „arbitrary“ signál.



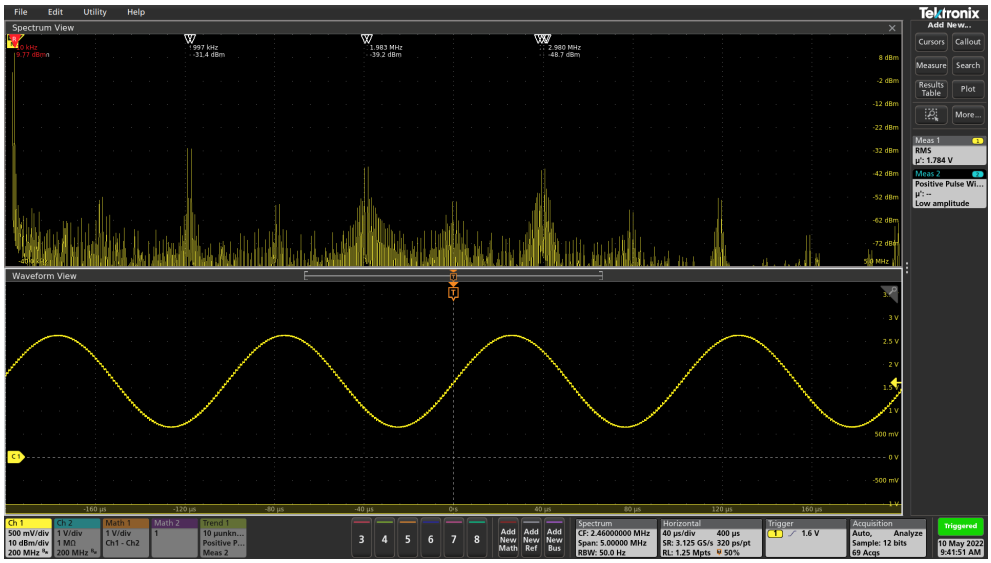
Obr. 4.1: Ukázka generování harmonického a „arbitrary“ signálu

Obrázek 4.2 ilustruje využití režimu frekvenční logaritmický „sweep“. První kanál (žlutý) generuje signál s logaritmicky rostoucí frekvencí od 10 Hz do 1 kHz, zatímco druhý kanál generuje signál pro jednoduché triggerování osciloskopu.



Obr. 4.2: Ukázka režimu frekvenční sweep

Následující obrázek 4.3 zobrazuje harmonický signál o frekvenci 10 kHz a jeho spektrum, ve kterém jsou jasně rozeznatelné obrazy generovaného signálu okolo násobků vzorkovací frekvence. Odstup signálu od nežádoucích špiček ve spektru (SFDR) je v tomto případě 41,17 dB a SNR je odhadem 70 dB. Tyto hodnoty jsou poměrně nízké ve srovnání s laboratorním přístrojem a je to dáno omezeným výkonem mikrokontroléru a absencí rekonstrukčního filtru.



Obr. 4.3: Ukázka spektra generovaného harmonického signálu o frekvenci 10 kHz

4.1.1 Možnosti dalšího vývoje přístroje

Vyvinutá implementace stále skýtá oblasti, ve kterých lze tento přístroj rozvíjet. Tyto nápady zatím nebyly implementovány, protože nebyly prioritou pro splnění cílů této práce. Jejich případná implementace v budoucnu tak rozvine tento přístroj i v aspektech, které nebyly v zadání požadovány.

- Otestování vzorkovací frekvence 4 MHz
  - Na mikrokontorléru STM32G431 se podařilo dosáhnout vzorkovací frekvence 4 MHz, což výrazně zlepšuje kvalitu reprodukce signálu a použitelný frekvenční rozsah. Tato frekvence je však 4x vyšší, než maximální frekvence specifikovaná v datasheetu DAC modulu. Je tedy třeba otestovat, zda nedochází ke zkreslení či chybám.
- Využití vestavěných operačních zesilovačů
  - Interním propojením DAC modulů se zabudovanými operačními zesilovači je možno vyvést výstupy DAC na jiné piny. Také tím lze využít DAC moduly v mikrokontroléru STM32G431, které mají garantovanou maximální vzorkovací frekvenci 15 MHz, ale nemají výstupní buffer a výstupy nemají vyvedeny na pin.
  - Zabudované operační zesilovače lze teoreticky použít jako aktivní rekonstrukční filtry (například Sallen-Key dolní propust 2. řádu). Je však třeba ověřit, že frekvenční rozsah operačních zesilovačů je pro toto použití dostačující.

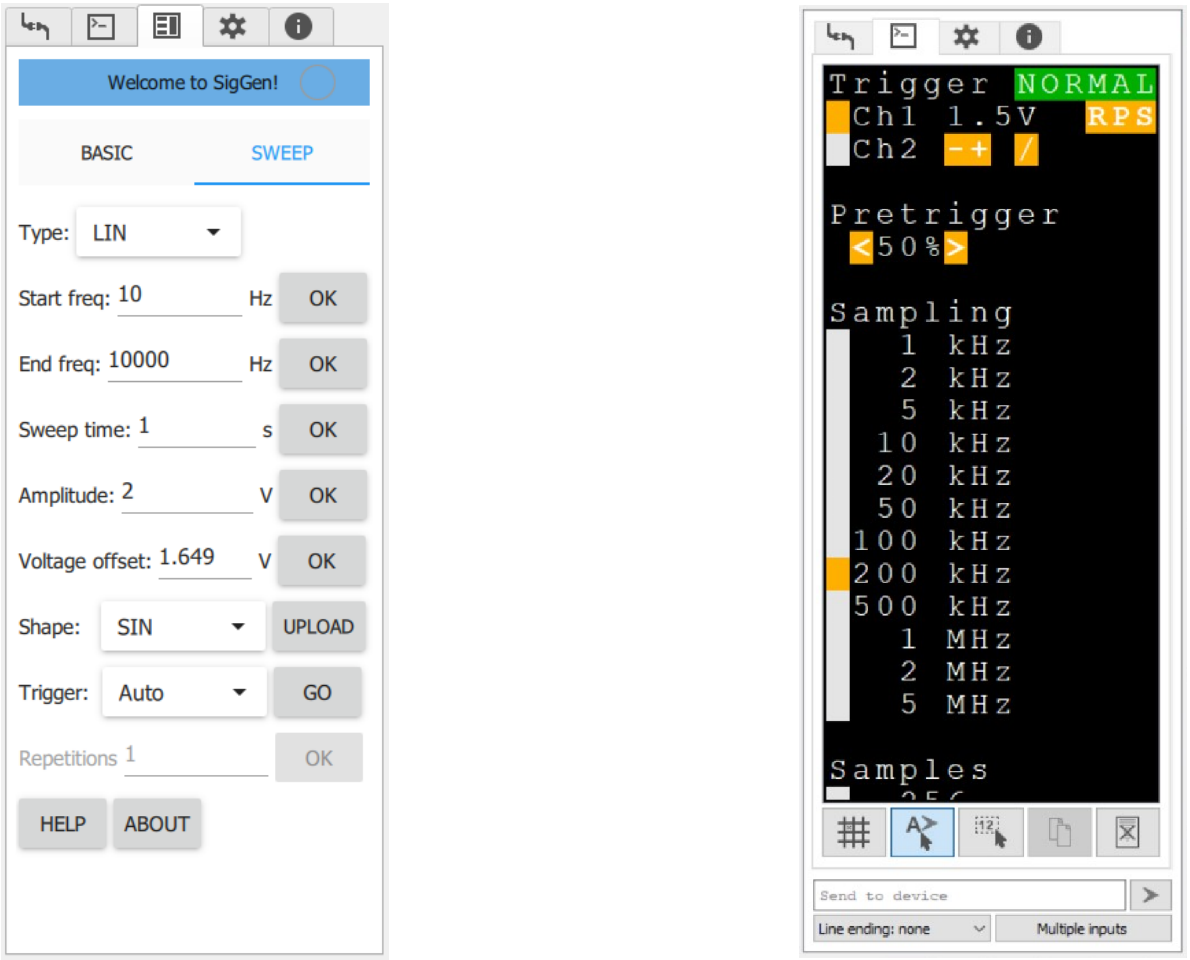
- Využití zabudovaných funkcí DAC modulů
  - Moduly DAC mají zabudované funkce pro jednoduché generování některých signálů. Například funkce generování šumu se jeví jako užitečná.
- Možnost automatické detekce připojení krystalového oscilátoru
  - Automatickým rozhodováním o zdroji hodinového signálu na základě detekce krystalového oscilátoru lze zajistit možnost lepší stability frekvence generátoru bez nutnosti zvýšit složitost obvodu na nepájivém poli.
- Možnost automatického ladění interního oscilátoru mikrokontroléru podle USB
  - Hodinový signál sběrnice USB má garantovanou přesnost alespoň 0,25 %, což je přibližně o řád lepší než interní oscilátor HSI16. Správným propojením časovačů a hodinových signálů v mikrokontroléru by mohlo jít automatizovaně ladit HSI16 za běhu programu. Tuto hypotézu je však třeba ještě prověřit.

4.2 Uživatelské rozhraní

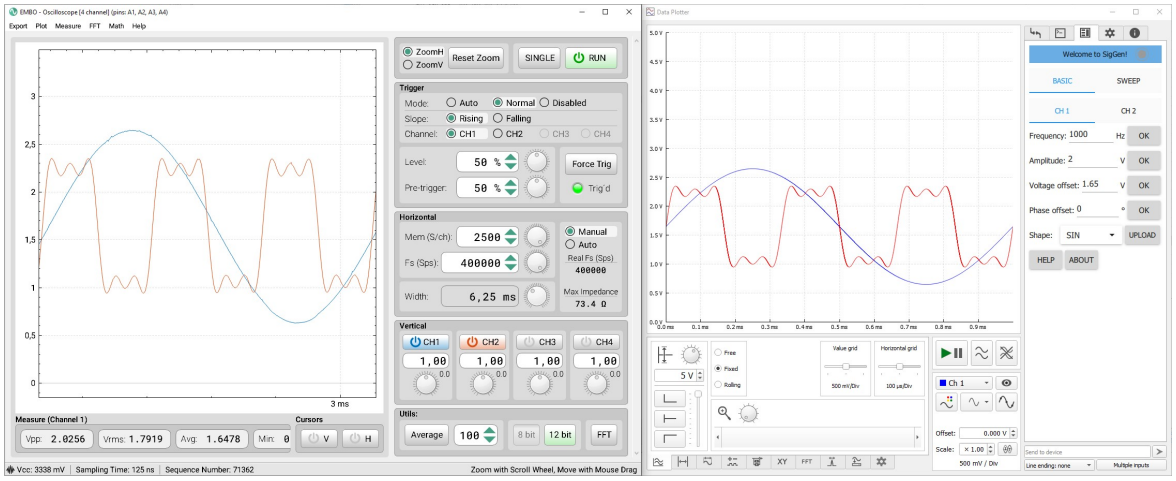
Pro ovládání tohoto přístroje bylo vyvinuto grafické rozhraní postavené na PC aplikaci DataPlotter [1] doplněné o vlastní definici ovládacího panelu přístroje. Navržené řešení využívá nový přístup k definici uživatelského rozhraní pomocí QML jazyka. Rozhraní je řízeno mikrokontrolérem, zatímco aplikace DataPlotter a implementovaný ovládací panel slouží jen pro zobrazení ovládacích prvků a interakci uživatele s nimi.

Oproti minulým implementacím ovládacích rozhraní Software Defined přístrojů založených na TUI terminálu je použité řešení vzhledem i ovládáním podobné nativní PC aplikaci. Interakce s ovládacím panelem je založena na běžných úkonech, jako je úprava hodnoty v textovém poli nebo kliknutí na tlačítko. Ovládací panel je tak k nerozeznání od PC aplikace, a přitom je stále řízen mikrokontrolérem. Na obrázku 4.4 jsou porovnána rozhraní osciloskopu vyvinutého s aplikací DataPlotter a rozhraní této práce navržené v jazyce QML. Na stejný prostor na obrazovce lze přehledně umístit mnohem více informací, ke kterým je zároveň okamžitý a intuitivní přístup.

Aplikace DataPlotter byla původně navržena pro SDI osciloskop, avšak i tento signálový generátor využívá oblast grafu zobrazujícího naměřené hodnoty. V případě signálového generátoru však nejde o naměřené hodnoty, ale o vypočtené hodnoty, které mikrokontrolér generuje. Uživatel tak přímo ve stejné aplikaci ihned vidí vliv právě upraveného nastavení na generovaný signál. Obrázek 4.5 ukazuje vpravo aplikaci DataPlotter jako rozhraní signálového generátoru s náhledem generovaného signálu a vlevo program EMBO, který slouží jako osciloskop zobrazující generovaný signál.

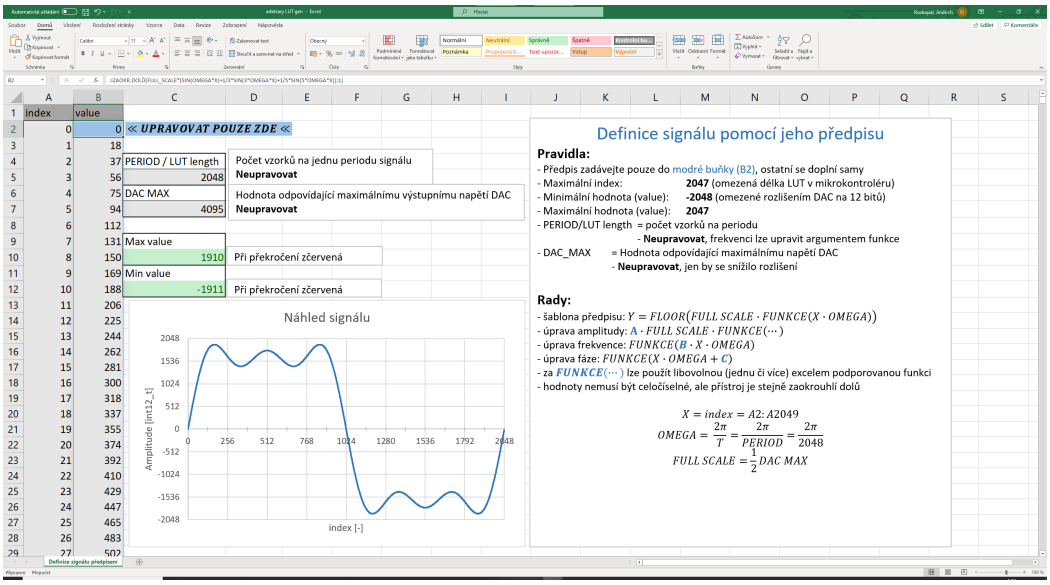


Obr. 4.4: Srovnání ovládacího panelu této práce (vlevo) a TUI terminálu (vpravo)



Obr. 4.5: Aplikace DataPlotter (vpravo) zobrazující rozhraní signálového generátoru a aplikace EMBO (vlevo) zobrazující generovaný signál naměřený osciloskopem

Pro zjednodušení návrhu vlastních „arbitrary“ signálů pro tento signálový generátor byly navrženy šablony pro programy Microsoft Excel a Matlab. Uživatel pouze zadá předpis požadovaného signálu a exportuje šablonu jako .csv soubor, který pak lze do přístroje jednoduše nahrát skrze jeho uživatelské rozhraní. Obrázek 4.6 ukazuje tuto pomůcku v programu Microsoft Excel.



Obr. 4.6: Šablona pro vytváření „arbitrary“ signálů pro navržený signálový generátor v programu Excel

4.2.1 Možnosti dalšího vývoje rozhraní

- Využití pokročilejších struktur jazyka QML
  - Navržené rozhraní aktuálně používá spíše základní postupy a objekty jazyka QML. Využitím pokročilejších jazykových struktur lze ušetřit velikost QML souboru
- Možnost nastavování hodnot pomocí kolečka myši
  - Změna hodnoty frekvence pomocí kolečka myši umožní jednodušší jemné i hrubé ladění frekvence například při hledání rezonance

Kapitola 5

Závěr

Cílem práce bylo navrhnout a realizovat na mikrokontrolérech STM32G431KBT6 a STM32F303RET6 dvoukanálový funkční generátor zaměřený na využití v laboratorní výuce.

Byl navržen program, který z obou zadaných mikrokontrolérů dokáže vytvořit dvoukanálový signálový generátor s možností nahrát „arbitrary“ signál a s režimem frekvenčního „sweepu“. Program využívá ke generování signálu algoritmus DDS upravený pro implementaci na mikrokontroléru. Program se na obou mikrokontrolérech podařilo optimalizovat do té míry, že výsledná vzorkovací frekvence je omezena pouze zabudovaným DAC modulem na 1 MHz. Přístroj disponuje kromě běžného režimu generování periodických signálů též režimem frekvenční „sweep“, který buďto lineárně nebo logaritmicky plynule mění generovanou frekvenci signálu. Uživateli je umožněno si v obou režimech přístroje vybrat jeden ze tří základních tvarů signálu nebo si nahrát libovolný vlastní signál ve formě .csv souboru.

Kromě samotné implementace signálového generátoru na mikrokontroléru bylo též pro tento přístroj vyvinuto uživatelské rozhraní založené na PC aplikaci DataPlotter s implementací vlastních grafických a ovládacích prvků pomocí jazyka QML. Tento nový přístup k definici uživatelského rozhraní nabízí vytvořit rozhraní využívající stejné ovládací prvky, jako běžné PC aplikace a je tak uživatelsky přívětivější, než předešlé implementace pomocí textového terminálu. Toto rozhraní je však stále plně definováno a řízeno mikrokontrolérem.

Tento přístroj nalezne využití v laboratorní výuce na Katedře měření ČVUT FEL, kde bude používán jako jednoduchá náhrada klasického laboratorního signálového generátoru. Navržený přístroj sice nedosahuje stejných parametrů, jako laboratorní generátor, ale zato si jej studenti mohou jednoduše sestavit na nepájivém poli. Velkou výhodou oproti laboratornímu přístroji je dostupnost tohoto přístroje pro studenty. Narozdíl od laboratorního přístroje, tento si může student odnést domů například na práci na semestrálním projektu nebo si jej může kdokoli sestavit sám. Použité dva mikrokontroléry navíc již podporují více podobných Software Defined přístrojů a tento signálový generátor se k nim tedy přidává.

S ohledem na výše uvedené výsledky si myslím, že lze tvrdit, že cíle práce byly dosaženy.

Literatura

[1] MAIER, Jiří. Univerzální GUI pro osciloskopické PC aplikace [online]. České Vysoké Učení Technické v Praze, Fakulta Elektrotechnická, Technická 2, 166 27, Praha 6 - Dejvice, 2021 [cit. 2022-04-18]. Dostupné z: http://hdl.handle.net/10467/94674. Bakalářská práce. České Vysoké Učení Technické v Praze, Fakulta Elektrotechnická, Katedra elektromagnetického pole. Vedoucí práce Doc. Ing. Jan Fischer, CSc.

[2] PAŘEZ, Jakub. Softwarove definovaný osciloskop s mikroradicem STM32F103 [online]. České Vysoké Učení Technické v Praze, Fakulta Elektrotechnická, Technická 2, 166 27, Praha 6 - Dejvice, 2021 [cit. 2022-05-18]. Dostupné z: http://hdl.handle.net/10467/94788. Diplomová práce. České Vysoké Učení Technické v Praze, Fakulta elektrotechnická, Katedra měření. Vedoucí práce Doc. Ing. Jan Fischer, CSc.

[3] HLADÍK, Jiří, ed. Single chip software defined instrumentation for educational purposes. In: HUSNÍK, Libor. Proceedings of the International Student Scientific Conference Poster – 21/2017. 1. Faculty of Electrical Engineering, Printing Office, CTU in Prague: Czech Technical University in Prague, 2017. ISBN 978-80-01-06153-4.

[4] Analog signal generators. *Rohde-schwarz.com* [online]. [cit. 2022-05-18]. Dostupné z: https://www.rohde-schwarz.com/us/products/test-and-measurement/analog-signal-generators_333521.html

[5] OWON AG051F 5MHz 1-Channel Arbitrary Waveform Generator. *Owontechnology.eu* [online]. [cit. 2022-05-15]. Dostupné z: https://www.owontechnology.eu/product/1318411/owon-ag051f-5mhz-1-ch-arbitrary-waveform-generator

[6] A Technical Tutorial on Digital Signal Synthesis. *Analog.com* [online]. , 1999 [cit. 2022-04-16]. Dostupné z: https://www.analog.com/media/en/training-seminars/tutorials/450968421DDS_Tutorial_rev12-2-99.pdf

[7] MURPHY, Eva a Colm SLATRERY. Ask The Application Engineer-33: All About Direct Digital Synthesis. *Analog.com* [online]. 2004 [cit. 2022-05-01]. Dostupné z: https://www.analog.com/en/analog-dialogue/articles/all-about-direct-digital-synthesis.html

[8] PINI, Art. The Basics of Direct Digital Synthesizers (DDSs) and How to Select and Use Them. *Digikey.com* [online]. 2019, 2019-03-20 [cit. 2022-04-05]. Dostupné z: https://www.digikey.cz/en/articles/the-basics-of-direct-digital-synthesizers-ddss

[9] MT-085: Fundamentals of Direct Digital Synthesis (DDS). *Analog.com* [online]. 2009, 2009 [cit. 2022-05-18]. Dostupné z: https://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf?doc=cn0304.pdf

[10] VEDRAL, Josef a Michal JANOŠEK. 9 Sampling, Quantization, Coding, Reconstruction. *Moodle.fel.cvut.cz* [online]. [cit. 2022-05-18]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/321381/mod_resource/content/3/9_Sampling%2C%20quantization%2C%20coding%2C%20reconstruction.pdf

[11] KESTER, Walt. MT-001: Taking the Mystery out of the Infamous Formula,"SNR = 6.02N + 1.76dB,"and Why You Should Care. *Analog.com* [online]. 2009 [cit. 2022-05-18]. Dostupné z: https://www.analog.com/media/en/training-seminars/tutorials/mt-001.pdf

[12] MALCOVATI, P. a F. MALOBERTI. 17 Interface Circuitry and Microsystems. MEMS: A Practical Guide to Design, Analysis, and Applications (pp.901-942) [online]. 2006. Berlin, Heidelberg: Springer, 2006, 921 - 929 [cit. 2022-05-18]. ISBN 978-3-540-33655-6. Dostupné z: https://www.researchgate.net/publication/226649538_17_Interface_Circuitry_and_Microsystems

[13] Understanding Frequency Performance Specifications. *Ni.com* [online]. Feb 4, 2020 [cit. 2022-05-18]. Dostupné z: https://www.ni.com/cs-cz/support/documentation/supplemental/06/understanding-frequency-performance-specifications.html#

[14] FEHLHABER, William. Signal Analysis II: Linear vs Logarithmic Sine Sweep. *Linkedin.com* [online]. 25.9.2018 [cit. 2022-05-18]. Dostupné z: https://www.linkedin.com/pulse/signal-analysis-ii-linear-vs-logarithmic-sine-sweep-william-fehlhaber

[15] QML Applications | Qt 5.7. *Doc.qt.io* [online]. 2016 [cit. 2022-05-18]. Dostupné z: https://doc.qt.io/archives/qt-5.7/qmlapplications.html

[16] MAIER, Jiří. Data protocol guide cz.pdf. *Github.com* [online]. 4 Oct 2021 [cit. 2022-05-18]. Dostupné z: https://github.com/jirimaier/DataPlotter/blob/master/documentation/Data%20protocol%20guide%20cz.pdf

[17] MAIER, Jiří. Manual.pdf. *Github.com* [online]. 17 Mar 2022 [cit. 2022-05-18]. Dostupné z: https://github.com/jirimaier/DataPlotter/blob/master/documentation/Manual.pdf

[18] Use STM32F3/STM32G4 CCM SRAM with IAR Embedded Workbench®, Keil® MDK-ARM, STMicroelectronics STM32CubeIDE and other GNU-based toolchains. *St.com* [online]. February 2021 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/application_note/dm00083249-use-stm32f3-stm32g4-ccm-sram-with-iar-ewarm-keil-mdk-arm-and-gnu-based-toolchains-stmicroelectronics.pdf

[19] STM32G4 Series advanced Arm®-based 32-bit MCUs - Reference manual. *St.com* [online]. February 2022 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/reference_manual/rm0440-stm32g4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[20] Datasheet - STM32G431x6 STM32G431x8 STM32G431xB. *St.com* [online]. October 2021 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32g431rb.pdf

[21] RM0316 Reference manual: STM32F303xB/C/D/E, STM32F303x6/8, STM32F328x8, STM32F358xC, STM32F398xE advanced ARM®-based MCUs. *St.com* [online]. January 2017 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/reference_manual/dm00

043574-stm32f303xb-c-d-e-stm32f303x6-8-stm32f328x8-stm32f358xc-stm32f398xe-advanced-arm-based-mcus-stmicroelectronics.pdf

[22] STM32F303xD STM32F303xE: ARM® Cortex®-M4 32b MCU+FPU, up to 512KB Flash, 80KB SRAM, FSMC, 4 ADCs, 2 DAC ch., 7 comp, 4 Op-Amp, 2.0-3.6 V. *St.com* [online]. October 2016 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32f303rd.pdf

[23] STM32 Nucleo-64 boards (MB1136) - UM1724. *St.com* [online]. August 2020 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

[24] YIU, Joseph. A Beginner's Guide on Interrupt Latency - and Interrupt Latency of the Arm Cortex-M processors. *Community.arm.com* [online]. April 1, 2016 [cit. 2022-05-18]. Dostupné z: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/beginner-guide-on-interrupt-latency-and-interrupt-latency-of-the-arm-cortex-m-processors

[25] PM0214 Programming manual: STM32 Cortex®-M4 MCUs and MPUs programming manual. *St.com* [online]. March 2020 [cit. 2022-05-18]. Dostupné z: https://www.st.com/resource/en/programming_manual/dm00046982-stm32-cortex-m4-mcus-and-mpus-programming-manual-stmicroelectronics.pdf

[26] Description of STM32F4 HAL and low-layer drivers. *St.com* [online]. June 2021 [cit. 2022-05-19]. Dostupné z: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf

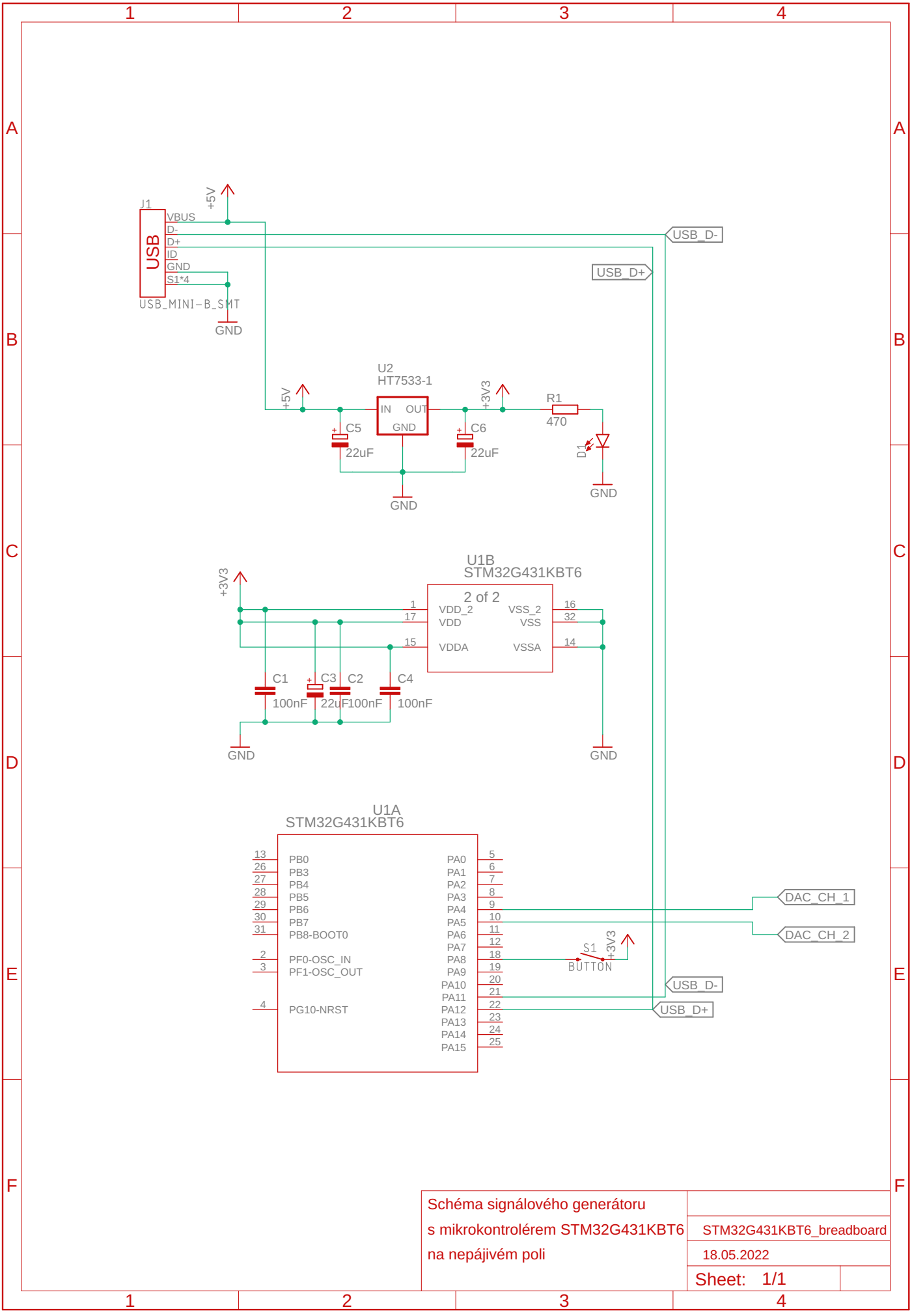
[27] How to place a variable at a given absolute address in memory (with GCC). *Stackoverflow.com* [online]. [cit. 2022-05-19]. Dostupné z: https://stackoverflow.com/questions/4067811/how-to-place-a-variable-at-a-given-absolute-address-in-memory-with-gcc

[28] Using Serial Wire Trace with CubeIDE (aka a new level of debugging). *Embedblog.eu* [online]. [cit. 2022-05-19]. Dostupné z: http://embedblog.eu/?p=673

Seznam příloh

Příloha A

Schéma – Signálový generátor s mikrokontrolérem STM32G431KBT6


1
2
3
4
A
B
C
D
E
F
J1
USB
VBUS
D-
D+
ID
GND
S1*4
USB_MINI-B_SMT
+5V
GND
USB_D-
USB_D+
U2
HT7533-1
IN
OUT
GND
C5
22uF
C6
22uF
+3V3
R1
470
D1
U1B
STM32G431KBT6
2 of 2
VDD_2
VDD
VDDA
VSS_2
VSS
VSSA
C1
100nF
C3
22uF
C2
100nF
C4
100nF
U1A
STM32G431KBT6
PB0
PB3
PB4
PB5
PB6
PB7
PB8-BOOT0
PF0-OSC_IN
PF1-OSC_OUT
PG10-NRST
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
S1
BUTTON
DAC_CH_1
DAC_CH_2
Schéma signálového generátoru
s mikrokontrolérem STM32G431KBT6
na nepájivém poli
STM32G431KBT6_breadboard
18.05.2022
Sheet: 1/1

Příloha B

Obsah CD

1. diplomova-prace.zip
   - Soubory diplomové práce v LaTeX.
2. SigGen_STM32G431KBT6.zip
   - Zdrojové soubory signálového generátoru pro STM32G431KBT6.
3. SigGen_STM32F303RET6.zip
   - Zdrojové soubory signálového generátoru pro STM32F303RET6.
4. SigGen_STM32G431KBT6.bin
   - Binární soubor programu pro STM32G431KBT6.
5. SigGen_STM32F303RET6.bin
   - Binární soubor programu pro STM32F303RET6.
6. SigGen_STM32G431KBT6_4MHz.bin
   - Binární soubor programu ve verzi se vzorkovací frekvencí 4 MHz pro STM32G431KBT6.
7. SigGen_LUT_sablona.xlsx
   - Šablona pro vytváření „arbitrary“ signálů pro program MS Excel.
8. SigGen_LUT_sablona.m
   - Šablona pro vytváření „arbitrary“ signálů pro program Matlab.
9. SigGen_navod.pdf
   - Návod na použití přístroje.