VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta strojního inženýrství

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Michal Havlíček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

MOŽNOSTI ANALÝZY SIGNÁLU NA PLATFORMĚ ARDUINO
SIGNAL ANALYSIS OPTIONS ON ARDUINO PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Michal Havlíček

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Rostislav Huzlík, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Michal Havlíček**
Studijní program:	Strojírenství
Studijní obor:	Kvalita, spolehlivost a bezpečnost
Vedoucí práce:	**Ing. Rostislav Huzlík, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Možnosti analýzy signálu na platformě Arduino

Stručná charakteristika problematiky úkolu:

Předmětem práce je průzkum možného použití různých variant platformy Arduino pro základní analýzu signálu. V rámci práce student vytvoří program pro výpočet střední a efektivní hodnoty a Fourierovu transformaci.

Cíle bakalářské práce:

Seznámení s možnostmi platformy Arduino.
Seznámení s postupem výpočtu stejnosměrné hodnoty, efektivní hodnoty harmonického signálu a Fourierovy transformace.
Vytvoření programu, který v rámci signálu provede výpočet střední hodnoty, efektivní hodnoty a Fourierovu transformaci a výsledky přenese do počítače.
Otestování vytvořeného programu a zhodnocení možných omezení pro použití platformy Arduino pro analýzu signálů.
Závěr a doporučení pro praxi.

Seznam doporučené literatury:

TŮMA, Jiří, 1997. Zpracování signálů získaných z mechanických systémů užitím FFT. Praha: Sdělovací technika. ISBN 80-901936-1-7.

VODA, Zbyšek, 2015. Průvodce světem Arduina. Bučovice: Martin Stříž. ISBN 978-80-87106-90-7.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce se zabývá možností analýzy signálu na platformě Arduino, vytvořením programu, pomocí kterého je možné v rámci signálu provést výpočet střední hodnoty, efektivní hodnoty a Fourierovy transformace a tyto výsledky přenese do počítače. Rovněž se zabývá testováním vytvořeného programu a zhodnocením možných omezení při využití platformy Arduino.

ABSTRACT

The aim of this bachelors thesis is to analyse signal with Arduino platform by creation of program which is able to perform calculations of signal´s mean value, efficiency value, Fourier transform and subsequent export of these results into computes. Furthermore it examines testing of this program and evaluation of possible limitations with usage of Arduino platform.

KLÍČOVÁ SLOVA

Platforma Arduino, analýza signálu, střední hodnota, efektivní hodnota, Arduino IDE, FFT

KEYWORDS

Arduino platform, signal analysis, DC value, RMS value, Arduino IDE, FFT

BIBLIOGRAFICKÁ CITACE

HAVLÍČEK, Michal. *Možnosti analýzy signálu na platformě Arduino.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/129503. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. Vedoucí práce Rostislav Huzlík.

PODĚKOVÁNÍ

Tímto děkuji panu Ing. Rostislavovi Huzlíkovi, Ph.D. za cenné připomínky a rady týkající se zpracování bakalářské práce

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Rostislava Huzlíka, Ph.D. a s použitím literatury uvedené v seznamu.

V Brně dne 10.9.2020 ……

Havlíček Michal

OBSAH

1 ÚVOD

Strojírenský průmysl představuje pro lidstvo nepostradatelné odvětví, neboť obsahuje, jak návrh, tak výrobu veškerých zařízení a prolíná se tak napříč všemi sférami lidského života. Neustálým zlepšování a narůstajícím rozmachem strojírenství ale přichází také větší požadavky na produktivitu strojů a kvalitu vyráběných výrobků, což nutí výrobce zařízení zaměřovat se neustále více na problematiku zpracování signálu, ať už jde o měření různých veličin či přenos a zpracování signálu. Získáním možná co nejvíce hodnot během výroby produktu následně usnadňuje řízení celých procesů a při správné zpracování může představovat nemalou finanční úsporu.

Zpracováním signálu se zabývá samostatný vědeckotechnický obor na rozhrání aplikované matematiky a elektrotechniky. Obecně lze říct, že signál přenáší určitý typ zprávy a tato zpráva může obsahovat určité množství informací. Tento obor se tedy zabývá způsoby a možnostmi tyto informace z přenášené zprávy získat a dále je zpracovat. Pro účely získání právě těch informací, které jsou podstatné pro daný úkol, kterým se člověk zrovna zaobírá.

Tato bakalářská práce se zabývá možností analýzy signálů na platformě Arduino. Jak již bylo zmíněno, samotný signál může mít různé podoby, které si v jedné z kapitol této práce rozebereme. Dále budou vytvořeny programy, které budou schopny zpracovávat příchozí harmonický signál a vracet nám konkrétní velikost střední hodnoty, efektivní hodnoty a Fourierovi transformace různých druhů signálů. Následně budou vypočtené hodnoty zobrazované v reálném čase přímo v počítači. V poslední řadě se budou napsané programy testovat, zda správně fungují a jestli dané hodnoty vycházejí správně. Poslední částí bude zhodnocení celé práce a možných omezení při využití platformy Arduino.

2 ARDUINO

Arduino je platforma s mikrokontrolerem, jehož zakladateli jsou David Cuartielles a Massimo Banzi. Projekt Arduino vznikl roku 2005 v Italském městě Ivrea a je pojmenován po Arduinovi Ivrejském. Roku 2006 získal ocenění Prix Ars Electronica. Arduino je velmi cenově dostupné Tento projekt je od svého začátku volně dostupný všem uživatelům. Projekt vyvíjí snahu podpořit výuku informatiky a seznámit studenty s řízením různých zařízení za pomoci programování. Pro ty, co si chtějí Arduino postavit sami, jsou dostupné návrhy plošných spojů.

Obr. 1) Arduino Community Logo [15]

Arduino desky se nejčastěji osazují procesory ATMega328, ATMega168, ATMega8, ATMega2560 a ATMega1280. Piny procesorů jsou na desck8ch vyvedeny na konektory, přes které se pak připojují jiná zařízení, nayvané tzv. “Shieldy“. Ty lze zakoupit již hotové. Vyrábí se například desky s LCD či segmentovými displeji, desky s reproduktory, diodami, tlačítky, klávesnicemi nebo i drivery pro řízení krokových motorů, či servomotorů. Existují různé typy desek. Od sebe se liší počtem I/O pinů, použitým čipem, možnými sběrnicemi, PWM, typem připojení USB, pamětí EEPROM, flash či digitálními a analogovými piny. K dostání jsou například desky , Duemilanove, Uno, Due, Mega, Mega2560, Leonardo, Fio či Nano.

Programy pro Arduino tvoří typicky nekonečnou smyčku s možnou kontrolou svého okolí a následnou reakcí. Řídící program je tvořen v prostředí Arduino IDE a do desky je pak vložen až hotový. K počítači se desky připojují pro naprogramování přes USB port, který je dále na desce konvertován na RS-232. Programovací jazyk Wiring je velmi podobný známému jazyku C/C++. K pohodlnému programování neodmyslitelně patří právě vývojové prostředí (IDE) napsané v jazyce Java a dostupné v několika jazycích. Prostředí má několik základních funkcí jako je kontrola chyb v programu, nahrání programu do desky nebo sériový monitor. Zde lze sledovat dění na sériové lince. Software obsahuje ale i několik hotových jednoduchých programů a knihoven, které stačí jen nahrát do vývojové desky

Velikou výhodou je, že tato platforma má kolem sebe velmi četnou komunitu, která vytváří velké množství materiálů pro samostudium a tvůrčí činnost. Například tutoriály, projekty a další. Open-source a open-hardware povaha platformy přispívají k tomu, že existuje i velké množství klonů desek Arduino, které jsou finančně velmi dostupné, stejně jako moduly a senzory pro platformu Arduino, a proto si případní zájemci o tuto problematiku mohou pořídit svou vlastní vývojovou desku a vybavení k ní a pokračovat s experimentováním i v domácích podmínkách. [1], [2], [5].

Další důležitou vlastností Arduina v rámci této bakalářské práce jsou možnosti ad převodníku na procesorech. Ten je nastaven na rozsah 0 až 3,3 voltu. Proto není schopen

vykreslovat zpracované signály do záporných hodnot. Nejsme tedy schopni dosáhnout symetrie k nule například u sinusového signálu. Pro tento účel bychom museli využít extérní elektroniku.

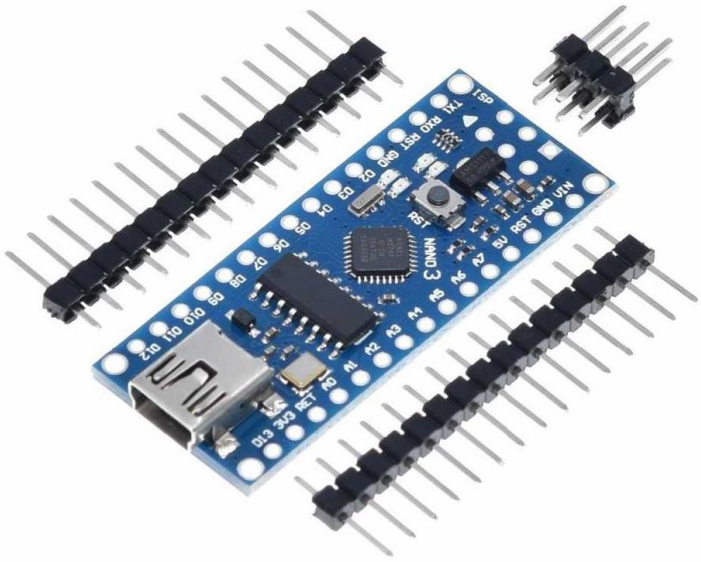
2.1 Přehled Arduino desek

Po téměř 15 letech existence platformy Arduino bylo vytvořeno mnoho desek, z nichž je nejpoužívanější pravděpodobně Arduino UNO. Kromě té existuje například verze Mega (v současné době konkrétně Mega 2560), jejímž hlavním rozdílem je větší počet GPIO pinů a větší paměť. Dále třeba desky Nano a Micro, u kterých byla nejpodstatnějším kritériem co nejmenší velikost. Kromě nich ještě existují další desky, které se již tolik nepoužívají. Příkladem může být třeba Yún s mnohem výkonnějším čipem, díky němuž může na desce fungovat linuxová distribuce. Dále ještě Lilypad, deska primárně zaměřená na tvorbu nositelné funkční elektroniky jako třeba blinkry na kolo umístěné na bundě, ovládané například senzory v rukavicích. Nejběžnější desky, tedy (UNO, Mega, Nano, Micro, ale i další) používají procesory od firmy Atmel. A to konkrétně typy ATmega328(P), ATmega32U4 nebo ATmega2560, ty pracující na frekvenci 16 MHz. Paměť programu a dat je oddělená (Harvardská architektura). Velikost pamětí se také u jednotlivých desek liší [7] [9] [21].

- **Arduino UNO**
  Byla první a dosud i zdaleka nejpoužívanější deskou platformy Arduino. V současné době se prodává už ve svojí třetí revizi (značeno „R3“, nebo „Rev3“). Deska používá procesor ATmega328. Jejími hlavními přednostmi jsou kompromis mezi velikostí, cenou a snadným přístupem k pinům. Pro Arduino Uno je také koncipována většina „shieldů“, ty již byli popsány výše. Ty se nasadí na konkrétní úsek pinů a dodávají desce dodatečnou funkcionalitu. Například internetové připojení. V současné době jsou ovšem již na ústupu. Programování a napájení je zajištěno přes vestavěný USB-B port, přičemž napájení může být řešeno i pomocí vestavěného kulatého konektoru případně přímo přes piny Vcc a GND. Jedná se rovněž o první desku, kterou v této bakalářské práci použijeme pro testování programů [7] [21].

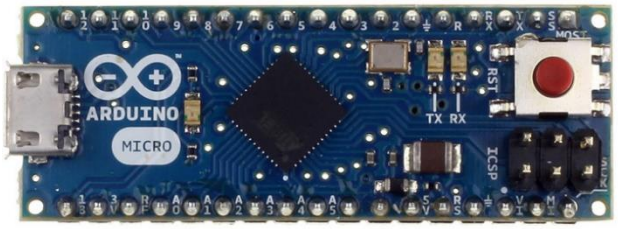
Obr. 2) Aduino Uno Arduino UNO [18]

- **Arduino Mega**
  Tato deska vznikla rozšířením desky Arduino UNO o větší paměť a větší počet pinů, což je také jeho hlavní výhodou. Využívá se tam, kde již nedostačuje počet pinů Arduina UNO, případně je také vhodné pro lidi s velkými prsty, jelikož větší velikost desky umožňuje lepší připojování k pinům. [21]
- **Arduino Nano**
  Je deska, která má opět prakticky totožnou funkcionalitu s deskou Arduino UNO, ale je určeno pro použití s opačnými potřebami než Arduino Mega. Nano je vytvořeno s cílem být co nejmenší. Programování probíhá přes mini-USB. Menší rozměry předurčují Arduino Nano k trvalé vestavbě do projektů. Například použití v modelech letadel, nebo všude tam, kde velikost a hmotnost hrají zásadní roli. Deska je osazena osmibitovým mikrokontrolerem ATmega328. Obsahuje 8 analogových vstupů, 6 kanálů PWM a 14 digitálních I/O pinů. Napájení může být řešeno opět přímo z USB portu, nebo přes zabudovaný stabilizátor, který může mít na vstupu napětí 6-20 V. Maximální výstupní proud každého programovatelného pinu je až 40 mA Arduino Nano obsahuje vnější či vnitřní přerušení. Na pinech čipu je i sběrnice SPI, ale i I2C. Při zacyklení procesoru nebo vyžaduje-li to situace lze použít resetovacího tlačítka umístěného přímo na desce. Původně se mělo jednat o první využitou desku v této bakalářské práci, bohužel však z důvodů nefunkčnosti objednaného kusu musel být nahrazen verzí Leonardo Pro Micro [7] [21].



Obr. 3) Arduino Nano [16]

- **Arduino Micro**
  Stejně jako Arduino Nano je zástupcem miniaturních desek určených spíše pro trvalé zabudování do projektu, kde záleží na co nejmenší velikosti a váze. Hlavním rozdílem je ovšem použitý procesor: ATmega32U4, který má zabudovaný USB převodník. Díky tomu se Micro může chovat jako standardní vstupní periferie počítače. Může tedy být použit například jako myš nebo klávesnice a zároveň takové zařízení lze naprogramovat. [21]



Obr. 4) Arduino Micro [19]

- **Arduino Leonardo Pro Micro**
  V této práci je jako druhá deska využita nadstavba desky Micro s názvem Leonardo Pro Micro. Dále v celé práci již bude označována pouze jako Micro. Je tím však myšlena právě tato nadstavba. Od varianty Micro se liší pouze počtem menším počtem vstupů. Na desce je osazeno 12 digitálních a 4 analogové I/O piny. Kde deska Micro má osazeno 20 digitálních a 12 analogových I/O pinů. Dalším rozdílem je váha která je v nadstavbě Leonardo pouhých 5gramů oproti Micru se 7gramy. Tato rozdíly jsou však pro naše využití zanedbatelné. [21]

- **Arduino Due**
  Jedná se o vývojovou desku založenou na procesoru Atmel SAM3X8E ARM Cortex-M3. Je to první Arduino deska založená na 32bitové architektuře. Má 54 digitálních I/O pinů (z nichž 12 je možné použít pro PWM výstupy), 12 analogových vstupů, 4 UART (hardwarové sériové porty), taktovací frekvenci 84 MHz, USB umožňující OTG připojení, 2 DAC (digital to analog) piny, 2x TWI, napájecí konektor, SPI čtečku, JTAG čtečku, resetovací tlačítko a tlačítko pro smazání paměti. Na rozdíl od ostatních desek běží na 3,3 V. Maximální napětí na I/O piny je 3,3 V. Vyšší napětí může poškodit vývojovou desku. Arduino due je třetí deska kterou budeme využívat v této práci [7] [21].

Obr. 5) Arduino DUE [17]

3 ARDUINO IDE A JAZYK WIRING

Jak již bylo zmíněno na začátku, Arduino je názvem projektu (platformy), sdružující pod sebou jak hardwarové prostředky, tak vlastní jazyk a vývojové prostředí. V následující části bude stručně popsán programovací jazyk Wiring a programovací prostředí Arduino IDE. Nastíněn bude i princip programování desek.

3.1 Jazyk Wiring

Nejprve je třeba uvést, že samotný jazyk Wiring byl vytvořen v roce 2003 jako diplomová práce Hernanda Barragána. Později, v roce 2005, Massimo Banzi naklonoval repositář jazyka Wiring a se svým týmem ho začal vyvíjet pod platformou Arduino [11]. Samostatný jazyk Wiring sice stále existuje, ovšem je na okraji zájmu. Pojem „jazyk Wiring“, tedy v této práci (a v drtivé většině případů na internetu) odkazuje na klon původního jazyka Wiring pod platformou Arduino a takto zde také bude uvažován.

Jazyk Wiring při svém vzniku cílil na zjednodušení práce umělcům a designerům, kteří používají elektroniku a kteří se do té doby museli učit relativně složité jazyky. Základem tohoto je jazyk C++. Označení „jazyk“ ve spojitosti s Wiringem ovšem může být diskutabilní, po stránce technické má spíše povahu řekněme rozsáhlé knihovny nebo frameworku pro jazyk C++.

Základní myšlenkou je „zapouzdřit“ složitou interakci s mikrokontrolery a nahradit je jednoduše použitelnými funkcemi. Například pokud chci nastavit pin jako výstupní není třeba nastavovat žádné bity v registrech, stačí zavolat funkci **pinMode()**. Ta vše provede a tím odstíní programátora od zásahu do samotného mikrokontroleru. Toto umožňuje velmi jednoduše programovat velkou škálu mikrokontrolerů pomocí stejných funkcí. V jazyce Wiring lze programovat kromě desek Arudino například i desky od společnosti Espressif s čipy ESP32 nebo ESP8266.

Programy pro Arduino se přezdívají „sketche“ a používají příponu .ino. Tato přípona je však požadována pouze u hlavního souboru, který obsahuje funkce **setup()** a **loop().** Pokud je výsledný kód rozdělen do více souborů, ostatní soubory již mohou využívat standardní přípony jazyka C nebo C++ (c., .cpp, h.). Celý projekt ovšem musí být umístěn ve složce, jejíž jméno se shoduje se jménem hlavního souboru, tedy souboru s příponou .ino.

Celá struktura programů pro Arduino se dělí do dvou hlavních funkcí: **setup()** a **loop()**. Ve funkci **setup ()** se nachází ta část kódu, která se provádí pouze jednou. Jedná se tedy většinou o nastavení pinů (vstupní nebo výstupní) nebo inicializaci připojených komponent. Funkce **loop()**, jak již název napovídá, vykonává hlavní smyčku programu [5] [6] [21].

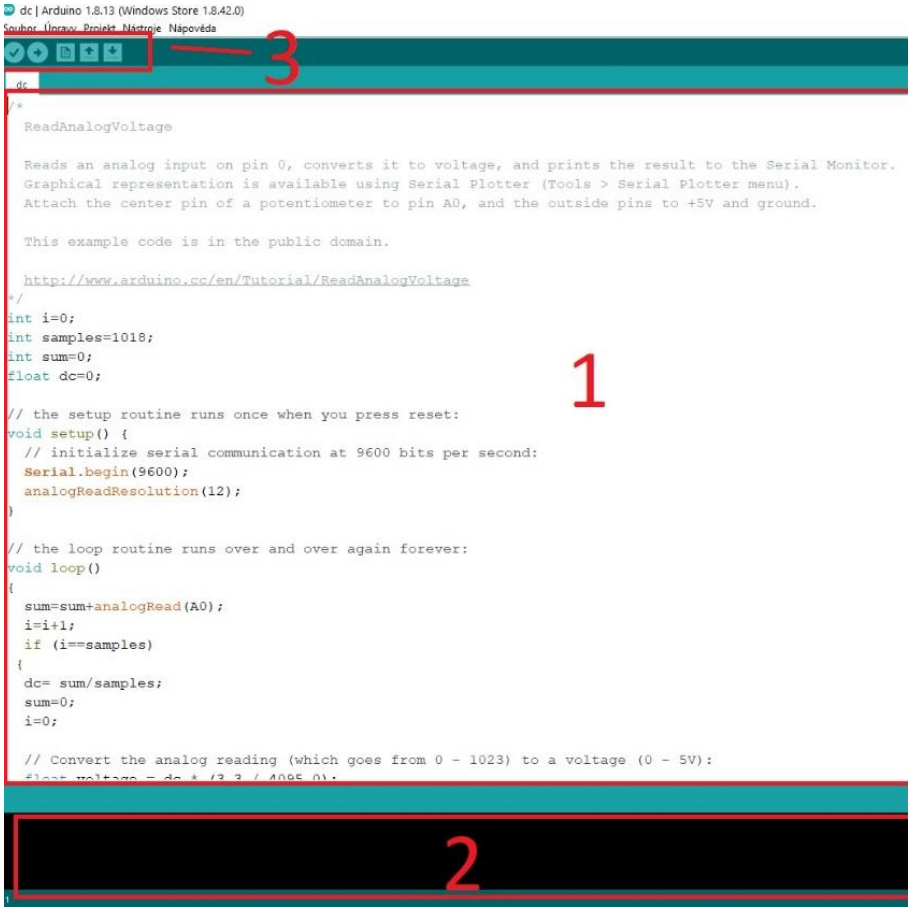
3.2 Arduino IDE

Jak již bylo zmíněno, oficiálním vývojovým prostředím pro platformu Arduino je Arduino IDE, velmi jednoduché vývojové prostředí. Ve své podstatě se jedná pouze o textový editor s funkcí nahrávání programů a zvýrazňování syntaxe. Arduino IDE bohužel nepodporuje například našeptávání nebo ladicí funkce. Problémem je také práce na více souborech. Ty sice mohou být otevřené zároveň, ale Arduino IDE nepodporuje rychlý přesun na místo definice/deklarace.

Na Obr. 6 můžeme vidět hlavní obrazovku tohoto vývojového prostředí. Kromě hlavní části, určené pro samotný kód (1), se ve spodní části nachází místo, kde se vypisují ladící

informace při nahrávání. Také je zde uživatel informován o průběhu překladu a nahrávání programu (2). Vlevo nahoře pod standardní navigační lištou se nachází (zleva doprava) tlačítka pro překlad, nahrání programu, vytvoření nového souboru, otevření stávajícího souboru a uložení (3) Připojení desky Arduino a její nastavení se provádí nejprve fyzickým připojením a poté přes nabídku „Tools“, kde je potřeba vybrat správné položky v nabídkách „Board“, „Processor“ a „Port“. První dvě položky jsou dány deskou, která je aktuálně používaná. Při použití originálních desek Arduino se COM port rozpozná jednoduše: v závorce za názvem portu je napsán typ desky, která je k danému portu připojena. U neoriginálních desek je třeba využít systému pokusů a omylů, pokud je tedy v nabídce více COM portů. Případně je možné otevřít si správce zařízení. Zde vidíme všechny aktuálně připojené porty. Port, který při odpojení desky zmizí, je s největší pravděpodobností portem, na kterém je deska připojená. Nyní stačí desku opět připojit a vybrat identifikovaný port.

Následné programování desky probíhá tlačítkem „Upload“ v horní části obrazovky (oblast 3). Pomocí nabídky „File→Open“ nebo klávesové zkratky „Ctrl + O“ je možné otevřít stávající soubor. Pomocí „File→New“ nebo „ctrl + N“ vytvořit. Pod „File→Examples“ se dají najít předpřipravené příklady použití Arduina, stejně tak jako příklady z přídavných knihoven, pokud má uživatel nějaké nainstalované.



Obr. 6) Ukázka prostředí Arduino IDE

3.2.1 Instalace přídavných knihoven

Arduino IDE má již po instalaci nespočet předinstalovaných knihoven, občas je ovšem nutné doinstalovat další. To se provádí přes nabídku „Sketch→Include library→Manage libraries“ v případě standardních Arduino knihoven. Pokud však bude nutno nainstalovat knihovnu, která se v této nabídce nevyskytuje (například ze serveru GitHub), provádí se tak přes „Sketch→Include library→Add .ZIP library“. Knihovnu je nutné mít nachystanou v .zip archivu. Po těchto krocích se doporučuje Arduino IDE restartovat [7] [9].

3.3 Matlab a Simulink

Arduino je možné s výhodami programovat a ovládat i přes prostředí Matlab a simulink.

Matlab je programovací jazyk a interaktivní vývojové prostředí. Zprvu byl určen především pro práci s maticemi, avšak dnes již zvládá mnohem více složitějších funkcí. První komerční verze byla vypuštěna v roce 1985 společností MathWorks. Do té doby byl Matlab zdarma. Matlab je napsaný pomocí programovacích jazyků C/C++, Fortranu a Javy. [10] Jeho aktuální verze je R2020a. V dnešní době je využívaný především inženýry, akademickými pracovníky a studenty. Tento program je sice placený, ovšem díky Matlab Campus Wide licenci je Matlab pro studenty a zaměstnance VUT bezplatný.

Simulink je grafické prostředí pro modelování a simulaci dynamických systémů. Jedná se o rozšíření Matlabu umožňující uživateli vytvářet bloková schémata, znázorňující reálné systémy. Je možné v něm modelovat například fyzikální soustavy, systémy pro zpracovávání signálů nebo algoritmy řídicích systémů. [4]

Matlab spolu se Simulink mohou být využity pro komunikaci s rozličným hardwarem, přičemž hobby platformy Arduino, Raspberry Pi nebo LEGO MINDSTORMS (EV3) jsou zdarma. Tato práce se zabývá platformou Arduino, tudíž ostatní výše zmíněný hardware zde není více popsán. Pro práci s Arduinem je nutné nainstalovat dvě rozšíření distribuovaná společností MathWorks. Jedná se o Matlab Support Package for Arduino Hardware. Toto rozšíření umožňuje sériovou komunikaci mezi Arduinem a Matlabem. Uživatel může zapisovat nebo číst data ze senzorů, nebo například pohybovat servomotorem přes Arduino a ihned vidí výsledky skrze Matlab, aniž by proběhla kompilace. Druhým rozšířením je Simulink Support Package for Arduino Hardwar. Jedná se o rozšíření, které umožňuje programování Arduino v Simulinku [3].

4 VÝPOČET STEJNOSMĚRNÉ HODNOTY, EFEKTIVNÍ HODNOTY HARMONICKÉHO SIGNÁLU A FOURIEROVY TRANSFORMACE

V této kapitole budeme pojednávat o způsobu zpracování signálu. Pro tyto účely musíme nejprve zavést základní rozdělení. Tím jsou signály typu:

Determinované průběhy lze vyjádřit matematickými funkcemi, které jednoznačně určují jejich funkční hodnoty v jednotlivých časových okamžicích. Pro konkrétní časové okamžiky, lze dosazením vypočítat konkrétní hodnotu veličiny. Ještě můžeme determinované signály dělit z matematického hlediska, a to na **spojité a nespojité.**

Z hlediska matematických postupů užívaných při analýze je dobré rozdělit determinované průběhy na **stacionární**, **periodické** a **neperiodické**. Přičemž Stacionární a periodické průběhy se mohou označit jako **ustálené**.

Naproti tomu **nedeterminované** (stochastické) průběhy mají charakter náhodných procesů, tedy jejich hodnoty lze očekávat jen s určitou pravděpodobností. Patří sem např. problematika šumů v elektrických obvodech [12] [13].

4.1 Stacionární a periodické veličiny

Stacionární průběh nemění velikost a smysl v čase. V praxi se používá pojmu **stejnosměrný**, přestože se nejedná o významově zcela adekvátní výraz (není v něm zahrnuta ona neměnnost velikosti). **Periodický** průběh je takový průběh, jehož hodnoty se opakují po určité době *T*, ta je nazývána **perioda**. Jednotkou jsou sekundy [*s*]. Pro periodický průběh veličiny platí vzorec

$$x(t + k \cdot T) = x(t) \tag{1}$$

,kde *k* je libovolné celé číslo. Převrácená hodnota periody je **frekvence** *f*

$$f = \frac{1}{T} \tag{2}$$

s jednotkou Hertz [*Hz*]. Běžně se užívá názvu **kmitočet**.

Periodické průběhy je dobré dále dělit podle těchto znaků. Obecný periodický průběh s rozdílnou kladnou a zápornou plochou v rámci periody se označuje jako **kmitavý**. Periodický průběh, který nabývá pouze kladné nebo pouze záporné veličiny, se nazývá **pulsující**. Typickými průběhy tohoto typu jsou např. jednocestně i dvojcestně usměrněný harmonický proud, nebo v praxi i často využívané obdélníkové impulsy.
K nejvýznamnějším souměrně střídavým průběhům, patří průběh **harmonický.** Ty nalézají největší uplatnění v praxi ale i v teoretických úvahách různorodého charakteru [12] [13].

4.2 Střední a efektivní hodnota harmonického signálu

V rámci zadání bakalářské práce je třeba seznámit se s postupem výpočtu stejnosměrné hodnoty, efektivní hodnoty harmonického signálu a Fourierovy transformace. Harmonické průběhy lze matematicky popsat funkcemi sinus a kosinus. Pomocí funkce sinus můžeme například psát pro harmonický průběh veličiny

$$x(t) = x_m \cdot \sin(\omega \cdot t + \psi_i) \tag{3}$$

,kde x_m je **amplituda** (maximální hodnota), ω je **úhlová frekvence** definovaná jako

$$\omega = 2 \cdot \pi \cdot f = \frac{(2 \cdot \pi)}{T} \tag{4}$$

s jednotkou [$rad \cdot s^{-1}$], a konstanta ψ_i značí **počáteční fázi** tzv. (fázový úhel) **[** *rad* **]**. Jak je z předchozích rovnic patrné, obecně je periodická funkce zcela určena časovým průběhem za dobu jedné periody. V určitých využitích ale vystačíme pouze se znalostí určitých **charakteristických hodnot** průběhu. Patří sem například **Maximální hodnota.** To je největší absolutní hodnota, které periodická funkce během periody nabývá. Značí se zpravidla indexem *m*. V praxi se setkáme také s označením *max*. Je-li třeba rozlišit kladnou a zápornou maximální hodnotu, které jsou různě velké, používají se indexy *m+* a *m-*. **Střední hodnota v době jedné periody** (**stejnosměrná složka**) je průměrná hodnota časové funkce za dobu jedné periody.

$$X_0 = \frac{1}{T}\int_0^T x|(t)dt \tag{5}$$

Definice obecněji použitelná se jeví tzv.
aritmetická střední hodnota, definovaná pomocí absolutní hodnoty následujícím vztahem.

$$X_{DC} = \frac{1}{T}\int_0^T |x(t)|\,dt \tag{6}$$

Jedná se o střední hodnotu dvojcestně usměrněné veličiny v době jedné periody. Uvedená charakteristická hodnota nachází využití v měřicí technice.
Pro účely této bakalářské práce bude využito diskrétní varianty tohoto vzorce v následujícím tvaru:

$$X_{DC} = \frac{1}{n}\sum_{i=1}^{n} x(i) \tag{7}$$

V technické praxi se hojně užívá **efektivní hodnota** periodického průběhu veličiny, neboť vyjadřuje její energetické účinky. Označuje se velkými písmeny bez indexu, pouze v případě možné záměny se může užít indexu *ef*, nebo *rms* (od anglického *root-mean-square*).

$$X_{rms} = \sqrt{\frac{1}{T}\int_0^T x^2(t)\,dt} \tag{8}$$

Stejně jako pro střední hodnotu i pro efektivní bude dále využit diskrétní tvar tohoto vzorce.

$$X_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x^2(i)} \tag{9}$$

Efektivní hodnotu je třeba v praxi často stanovit měřením. Proto se běžně měřicí přístroje (ampérmetry, voltmetry) „cejchují“ právě v hodnotách efektivních. Praktická realizace převodníků efektivní hodnoty je však obvodově a finančně náročnější než realizace převodníků pro měření hodnoty střední. Proto se v levných měřicích přístrojích používá převodníků pro měření střední hodnoty. Hodnota efektivní se pak získává pomocí cejchování přístroje vynásobením tzv. činitelem tvaru. [12] [13]

4.3 Fourierova transformace

Průběh signálu se běžně znázorňuje v čase neboli v časové oblasti tzv. doméně. Posuzování časového průběhu signálu vhodně doplňují i vlastnosti reprezentované rozkladem na soubor elementárních funkcí. Nejpřirozenější pro technické aplikace je nejen v obor u kmitání mechanických systémů rozklad na soubor harmonických funkcí. Ty se liší amplitudou, úhlovou frekvencí a svou počáteční fázi. Jestliže se tedy u souboru harmonických signálů znázorní závislost amplitudy a počáteční fáze na frekvenci, tak je signál znázorněn ve frekvenční oblasti tzv. doméně. Rozklad periodické funkce se spojitým časem na kombinaci harmonických signálů se nazývá Fourierova (nekonečná) řada. Pro obecné funkce neperiodické se používá Fourierova transformace.

V této kapitole budou popsány vlastnosti signálů s důrazem na frekvenční oblast, která je zvláště vhodná pro analýzu periodických diagnostických signálů. Tento typ signálů se analyzuje ve frekvenční oblasti mnohem přehledněji než v časové oblasti. Časovou oblast preferujeme pro znázornění signálu pokud v jeho spektru zůstanou složky jen užitečné pro posouzení jeho časového průběhu. K vytvoření takového časového průběhu jsou potřebné různé úpravy, které se provádějí právě ve frekvenční oblasti. Nástrojem k analýze signálů je tedy Fourierova integrální transformace. Pro různé aplikační oblasti lze nalézt řadu učebnic a příruček. [12] [13]

4.3.1 Fourierovy řady

Charakteristické pro periodickou funkci je rovnost navzájem posunutých funkčních hodnot x(t)= x(t + iT) , kde T je perioda a i = ±1, ±2, ±3, ... je její násobek. Tuto funkci x(t), lze rozvinout do tvaru (bodově) konvergentní Fourierovy nekonečné řady, za podmínky, že je po úsecích hladká. To znamená, že tato funkce a její první derivace má konečný počet bodů nespojitostí a konečné jednostranné limity. Definiční vzorce jsou pak následující

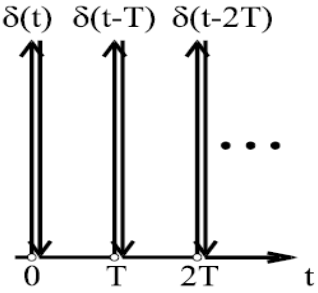
$$X_{(t)} = \sum_{k=-\infty}^{+\infty} F_k e^{\left(j\frac{2\pi}{T}kt\right)} \tag{10}$$

$$F_k = \frac{1}{T}\int_0^T x(t)\, e^{\left(-j\frac{2\pi}{T}kt\right)} dt, k = 0, \pm 1, \pm 2, .. \tag{11}$$

, kde F_k, k = 0, ±1, ± 2,... jsou koeficienty Fourierovy řady. V bodech nespojitosti je Součet Fourierovy řady roven aritmetickému průměru jednostranných limit. Fourierovu řadu v exponenciálním tvaru s harmonickými funkcemi můžeme převést na řadu, která obsahuje funkce sinus a kosinus. Tento rozklad nebude dále rozvíjen, protože vše je směřováno k interpretaci signálů jen pomocí rotujících vektorů. Fourierova řada tedy představuje rozklad signálu na nekonečný počet dvojic vektorů rotujících proti sobě. K definici Fourierovy řady je nutno ještě dodat, že signál x(t) nemusí být nutně reálná funkce času. Může se jednat také o komplexní funkce času. Ta je vhodná pro záznam orbitů. Mezi komplexní funkcí x(t) a koeficienty Fk platí Parsevalova rovnost

$$\frac{1}{T}\int_0^T |x(t)|^2\, dt = \sum_{k=-\infty}^{+\infty} |F_k|^2 \tag{12}$$

Pro symetrické koeficienty řady (11) platí, že jsou vzájemně komplexně sdruženy $F_k = F^*_k$, tj. jejich reálné části jsou shodné a imaginární části jsou opačné, jestliže je periodická funkce reálná. Tedy $Re\{F_{-k}\} = R\{F_k\}$, $Im\{F_{-k}\} = -Im\{F_k\}$. Pro koeficient F_k s indexem k = 0 je jeho imaginární složka nulová, Im{F}= 0. Koeficient F_0 je roven harmonickému signálu s nulovou frekvencí. Tedy vlastně konstanta ve významu střední hodnoty signálu, která se rovněž nazývá stejnosměrná složka signálu. Dále bude potřebný rozklad periodické funkce, která je složena z Diracových funkcí. Tato funkce se nazývá tzv. Diracův hřeben a je znázorněna na obr. Obr. 7. Jde tedy o funkci periodickou s periodou T, a tak ji lze rozložit na Fourierovu řadu. Koeficienty Forierovy řady lze vypočítat užitím filtrační vlastnosti Dirakovy funkce. [12] [13]



Obr. 7) Periodická funkce z Diracových impulsů převzato z [12]

4.3.2 Vlastnosti diskrétní Fourierovy transformace

Vzorec pro Fourierovou transformaci periodického signálu ve spojitém čase obsahuje nekonečnou posloupnost koeficientů F_k, k = 0, ±1, ± 2, ..., ve významu posloupnosti ploch Diracových funkcí. Při zpětné transformaci pak vystupují pouze koeficienty Fk ve významu amplitud příslušných harmonických složek signálu s úhlovou frekvencí 2πk/T, kde T je perioda signálu. Při vzorkování dle výše napsaného rozboru vznikají určité požadavky na vztah mezi frekvencí vzorkování a frekvenčním rozsahem měřeného signálu. Tato část kapitoly o Fourierově transformaci se bude zajímat o odvození vztahů, které předpokládají vzorkování a opakování signálu. Toto významově mění obecnou Fourierovou transformaci na tzv. diskrétní Fourierovou transformaci (DFT). Definice diskrétní Fourierovy transformace pozměněna na vztahy, které jsou invariantní ke vzorkovací frekvenci je.

$$F_k = \sum_{i=0}^{N-1} x_i \, exp\left(-\mathrm{j}\frac{2\pi}{N} k \, i\right). \; k = 0,1,2 \ldots . N-1 \tag{13}$$

$$x_i = \frac{1}{N} \sum_{k=0}^{N-1} F_k \, exp\left(\mathrm{j}\frac{2\pi}{N} k \, i\right). \; i = 0,1,2 \ldots . N-1 \tag{14}$$

Vzorec (13), tzv. přímá diskrétní Fourierova transformace, lze použít pro výpočet koeficientů F_k, k = 0, ±1, ± 2,..., pokud to jsou obecně komplexní čísla. Tyto koeficienty reprezentují pro každou frekvenci N-násobek poloviční velikosti vektorů rotujících proti sobě vzájemně opačnou úhlovou rychlostí. Koeficienty můžeme vypočítat všechny, pokud potřebujeme úplný ucelený obraz o frekvenčním složení signálu. Pro případ zájmu jen o některé frekvenční složky signálu vypočítáme jen ty, které požadujeme. Druhý vzorec (14), inverzní diskrétní Fourierova transformace se využívá v syntéze vzorkovaných hodnot signálu pro zvolené anebo cíleným způsobem upravené koeficienty F_k. [12] [13]

4.3.3 Algoritmus FFT a jeho vlastnosti

Z definičního vzorce přímé DFT pro záznam o délce N vyplívá, že k vyčíslení všech koeficientů, F_k, k = 0, 1, ..., N - 1, je třeba N^2 sčítání a také N^2 násobení komplexních čísel. Metodu značného urychlení výpočtu objevili Cooley a Tukey. Pro tuto metodu se rozšířil název rychlá Fourierova transformace (Fast Fourier Transform – FFT). Podstatou metody FFT je volba zvláštní délky záznamu, a to $N = 2^m$, kde m je přirozené číslo. Tato volba, která se označuje v angličtině radix 2, vede k předem daným, jasným, délkám záznamů např. N = 128, 256, 512, 1024, 2048, 4096, 8192, které jsou blízké k dekadické řadě. Další autoři později navrhli postupy také pro rychlý výpočet transformace DFT pro délky záznamu $N = r^m$ nebo $N = r_1 r_2 \ldots r_m$ s volbou r nebo r_1, r_2 .., která je různá od 2. Délka záznamu jako mocnina 2 je velmi běžná u FFT analyzátorů. Při měření přenosů je zapotřebí souběžně transformovat dvě posloupnosti. [12] [13]

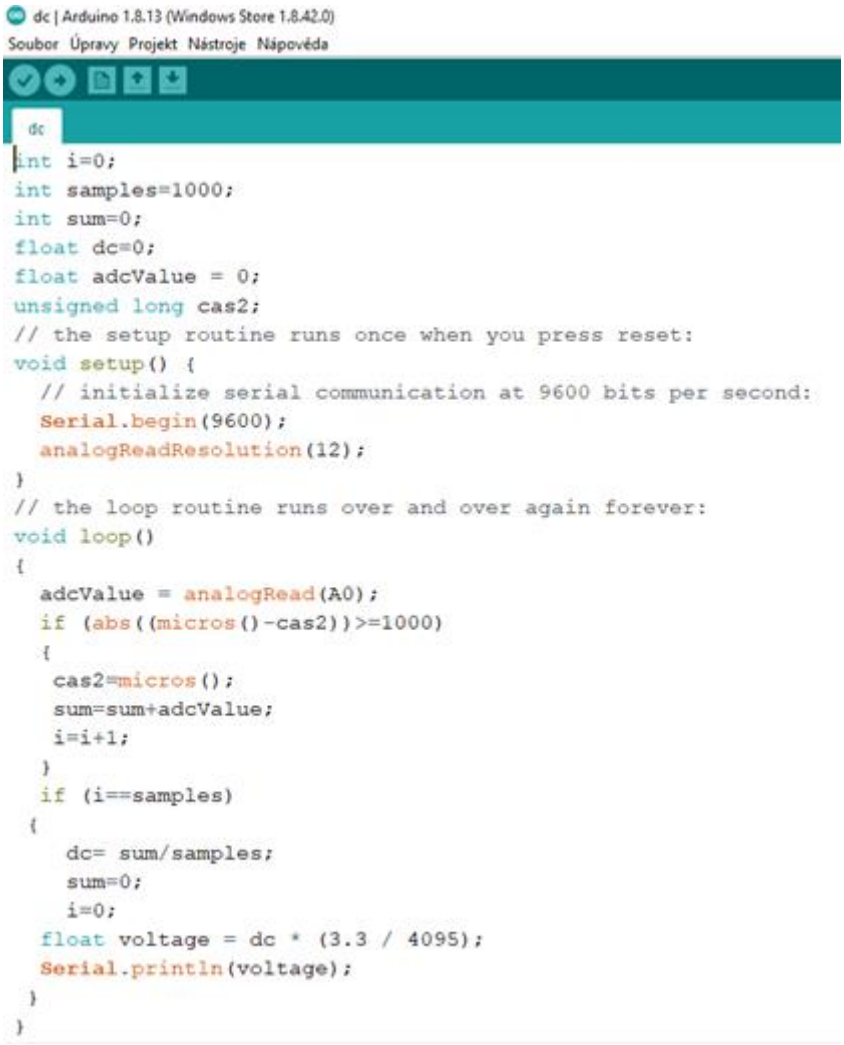
5 TVORBA PROGRAMŮ

Hlavním úkolem práce je vytvořit a odzkoušet funkčnost programů pro výpočet střední hodnoty, efektivní hodnoty a rychlé Fourierovi transformace.

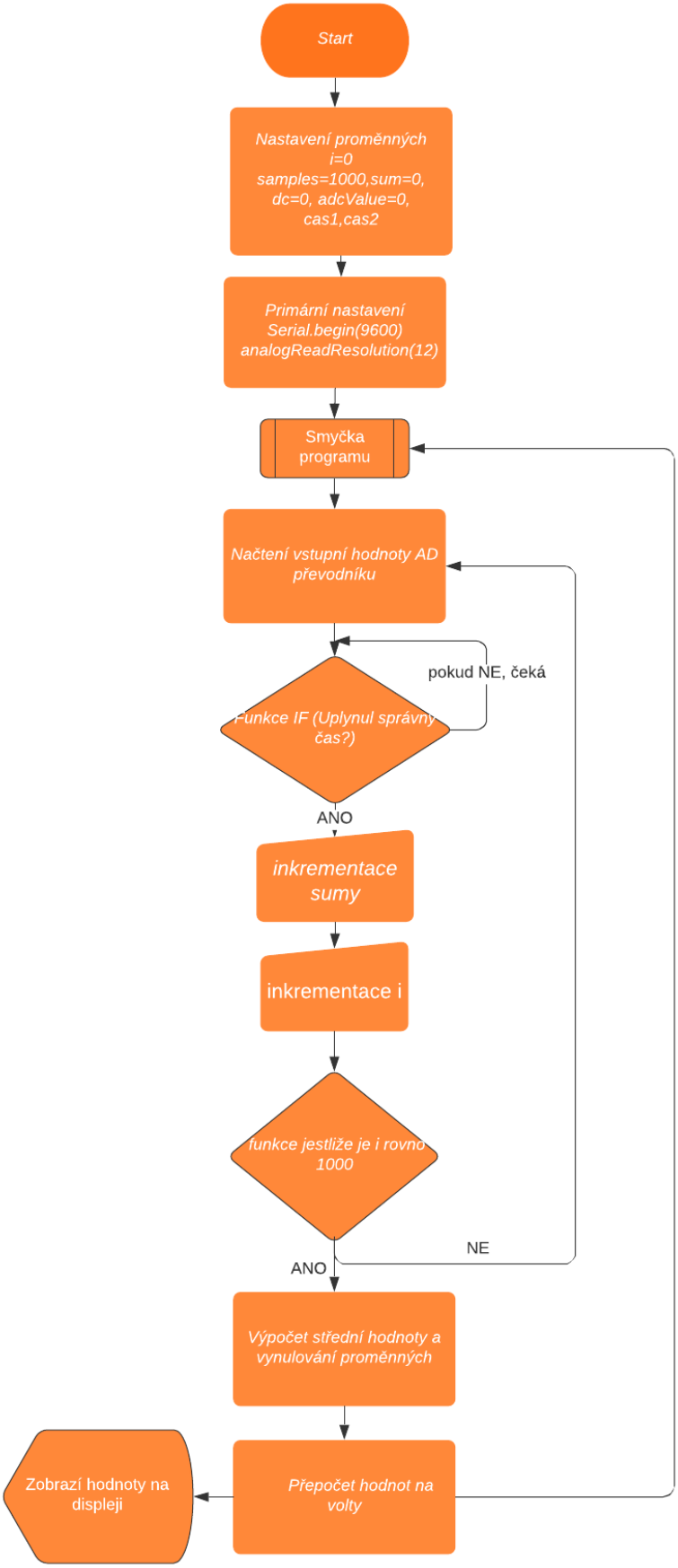
Všechny tyto programy budou vytvořeny v prostředí Arduino Ide popsaného výše. Dále v této části budou jednotlivé programy detailně popsány ve formě zdrojového kódu a jejich funkčnost znázorněna na vývojových diagramech. Všechny programy budou spolu s prací odevzdány jako příloha.

5.1 Střední hodnota (DC)

Pro tvorbu prvního programu budeme využívat rovnici pro výpočet střední hodnoty v diskrétním tvaru (7). Tento vzorec je potřeba zapsat v prostředí IDE do tvaru kódu, který bude v reálném čase neustále vyhodnocovat přicházející signál. Na následujícím na.Obr. 8 můžeme vidět zdrojový kód programu. Ten bude postupně popsán a vysvětlen. Jeho funkčnost si lze lépe představit pomocí vývojovéh diagramu na Obr. 9.



Obr. 8) Zdrojový kód DC hodnoty pro desku Due



Obr. 9) Vývojový diagram DC hodnoty

Nejprve je potřeba definovat si potřebné proměnné. První proměnnou bude i, které reprezentuje kolikátý vzorek měření probíhá. Samples určuje kolik vzorků celkem bude program měřit. Sum využijeme pro vyjádření hodnoty sčítaných výsledků z jednotlivých vzorků. Proměnná dc nám bude vyjadřovat konečnou střední hodnotu příchozího signálu. Další bude adcValue, ta slouží pouze pro načtení aktuální hodnoty na převodníku. Nakonec je definována časová proměnná cas2. Ta slouží pro funkci časování, která bude popsána dále v této kapitole.

Následuje **void setup()**, kde provedeme základní nastavení pro daný program, které se nastaví vždy jen jednou při nahrání programu, případně při resetu desky. Zde můžeme vidět příkaz **s Srial.begin(9600),** který spustí komunikaci Arduina v rychlosti 9600 bitů za sekundu. Druhým nastavením je **analogReadResolution(12)**, kterým nastavíme AD převodník na 12bitů. Tento příkaz však používáme pouze v případě použití desky DUE, ta totiž jako jediná z desek, které jsou pro tuto práci využity, má 12bitový AD převodník. Zbytek má 10bitový AD převodník, proto je v našem případě u desek Uno a Micro potřeba tento příkaz odstranit, protože touto možností nedisponují.

Nyní již přicházíme ke. smyčce programu neboli **void loop()**. Obsah této části se automaticky provádí stále dokola. Zde si v první řadě načteme aktuální hodnoty na AD převodníku do proměnné adcValue. V dalším kroku použijeme funkci **if()**. Ta má za úkol spustit další krok programu pouze v případě, že uběhla 1milisekunda času. Tím docílíme správného fungování a časování smyčky, aby v průběhu delšího času program nedošlo k zacyklení nebo jiné chybě.O Pokud bude splněna podmínka časování, provede se součet sumy a aktuální hodnoty adcValue, tedy vstupních hodnot. Zároveň se každou iterací přičítá hodnota pro i(tedy kolikátá iterace proběhla). Pokračujeme opět rozhodovací funkcí **if()**. Ta je nastavená tak, že pokud bude hodnota i rovna počtu měření které chceme (v našem případě 1000). Přejde program k výpočtu střední hodnoty, vezme sum a podělí jej počtem vzorků. Dále opět nastaví hodnoty sum a i na 0, díky čemuž se cyklus po dokončení programu může spustit znovu.

Poté v předposledním kroku přepočítá výslednou hodnotu digitální informace na konkrétní hodnotu střední hodnoty ve Voltech Tento přepočet budeme ještě dále rozebírat v kapitole o testování programů. Poté zobrazí tuto hodnotu na výstupu v počítači na sériový monitor pomocí příkazu **Serial.println()**.

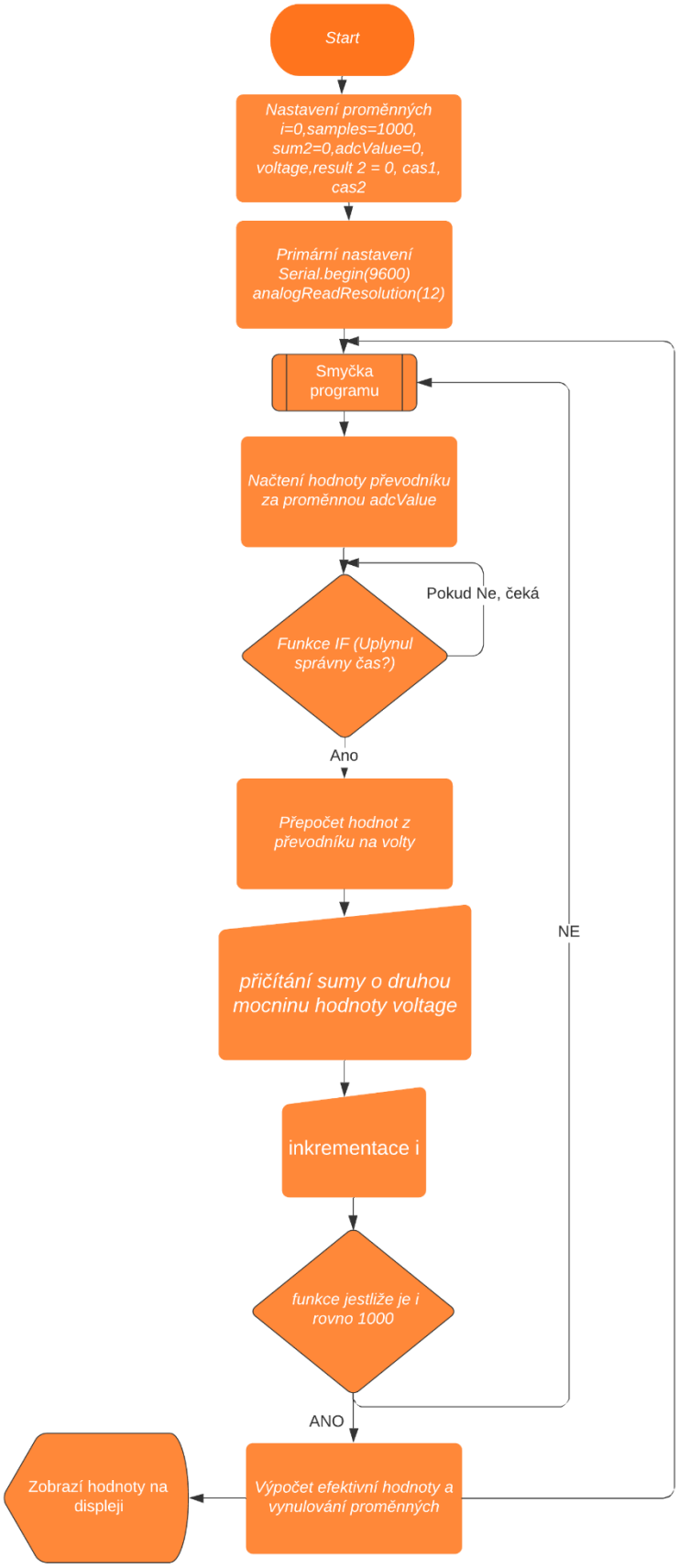
5.2 Efektivní hodnota (rms)

Pro tvorbu druhého programu budeme využívat rovnici pro výpočet efektivní hodnoty v diskrétním tvaru (9). Tento vzorec je potřeba opět zapsat v prostředí IDE do tvaru kódu, který bude v reálném čase neustále vyhodnocovat přicházející signál. Na následujícím obrázku Obr. 10 můžeme vidět zdrojový kód programu. Ten bude postupně popsán a vysvětlen. Stejně jako u střední hodnoty slouží pro lepší pochopení tohoto programu jeho vývojový diagram na obrázku Obr. 11.


```
float i=0;
float samples=1000;
float sum2 = 0;
float adcValue = 0;
float voltage;
float result2 = 0;
unsigned long cas2;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  analogReadResolution(12);
}
void loop()
{
  adcValue = analogRead(A0);
  if (abs((micros()-cas2))>=1000)
  {
   cas2=micros();
   float voltage = adcValue * (3.3 / 4095);
   sum2 = sum2 +voltage*voltage;
   i=i+1;
}
  if(i == samples)
  {
    result2 = sum2 / samples;
    result2 = sqrt(result2);
    Serial.println(result2);
    sum2=0;
    i=0;
  }
}
```

Obr. 10) Zdrojový kód RMS pro desku DueS



Obr. 11) Vývojový diagram RMS hodnoty

Na začátku programu si opět musíme vydefinovat proměnné vstupující do programu. Těmi jsou i reprezentující konkrétní číslo probíhajícího vzorku. Samples reprezentují kolik vzorků má být provedeno. Sum2 reprezentuje součet naměřených hodnot v druhé mocnině podle použitého vzorce. AdcValue odkazuje na konkrétní hodnoty čtené převodníkem Arduina. Voltage budeme využívat opět na přepočtení digitálních počítaných hodnot na hodnoty napětí a budou pro něj platit stejná pravidla jako u předchozího programu. Další proměnnou je result2 obsahující již konkrétní efektivní hodnotu, která je dle využívaného vzorce odmocněna. Nakonec je definována časová proměnná cas2. Ta slouží pro funkci časování.

Do **void setup()** vkládáme opět funkce pro zahájení komunikace, a protože se jedná o program psaný pro desku Due nachází se zde také převedení na 12bitové rozhraní.

Samotná smyčka programu je pak podobná předchozímu programu, ovšem z drobnými změnami podle použitého vzorečku. V první řadě se opět za proměnnou adcValue pomocí funkce **analogRead()** na čtení analogového vstupu Arduina dosadí konkrétní hodnota na převodníku. Následuje rozhodovací funkce **if()**, ta stejně jako u hodnoty DC slouží k dodržení časování vzorkování. Tedy pokud čas odpovídá dané hodnotě, provede se další část programu. Hodnota načtená do adcValue je nyní v dalším kroku převedena na hodnotu napětí. U předchozího programu byl tento krok použit až na konci cyklu. Zde ovšem musíme zařadit tento převod dříve, protože hodnoty, které zde dále dostáváme musejí být mocněny na druhou. Tím by došlo k přetečení proměnné, protože dostáváme příliš vysoká čísla. To by znemožnilo programu správně fungovat. V dalším kroku tedy počítáme sumu všech hodnot již převedených na Volty v druhé mocnině. S každým novým vzorkem se nám zde tedy sčítají již zmíněné sumy a proměnná i. Po dosažení 1000 iterací, tedy jakmile i dosáhne právě tohoto čísla bude splněna podmínka rozhodovací funkce **if()**, která spustí další část programu.

Zde se již vypočítá výsledná hodnota result2 tím, že nejprve sum2(tedy umocněný součet napětí všech provedených iterací) podělí právě počtem těchto vzorků a poté provede odmocnění druhou odmocninou. Dále tuto výslednou hodnotu zobrazí na sériový monitor. Nakonec vynuluje proměnné sum2 a i, aby se cyklus mohl opět opakovat.

5.3 FFT

Třetí a poslední program zpracovává příchozí signál pomocí již výše zmiňovaného algoritmu rychlé Fourierovi transformace. Pro tento program bylo využito jedné z přednastavených knihoven, které Arduino nabízí, jak bylo popisováno v druhé kapitole. Tato knihovna se jmenuje arduinoFFT. Program této předpřipravené knihovny však musel být pro náš účel zcela předělán. Využito z jeho původního kódu byli pouze některé funkce pro výpočet samotného algoritmu. Dále byla z této knihovny byla využita celá část zobrazovacího kódu, který vypisuje vypočtené hodnoty na sériový monitor. Proto nebude tato část kódu více popisována ani vysvětlena Její průběh je více popsán na stránkách autorů [20]. Program bude popsán pomocí zdrojového kódu na obrázcích Obr. 12 a Obr. 13. Vzhledem k rozsahu tohoto programu nebude dále přiložen ani vývojový diagram programu. Jeho velikost by neumožnovala vložit jej přímo do bakalářské práce a tím pádem by nepomohl s pochopením řešeného problému.

FFT_01 §

```
#include "arduinoFFT.h"
arduinoFFT FFT = arduinoFFT();

const uint16_t samples = 1024;
const double signalFrequency = 1000;
const double samplingFrequency = 1000;
const uint8_t amplitude = 100;
unsigned long cas2;
double valuesVoltageArr[samples];
double valuesVoltageArr_img[samples];
double adcValue;
int i=0;


#define SCL_INDEX 0x00
#define SCL_TIME 0x01
#define SCL_FREQUENCY 0x02
#define SCL_PLOT 0x03

void setup()
{
  Serial.begin(115200);
  Serial.println("Ready");
  analogReadResolution(12);
}
```

Obr. 12) Zdrojový kód FFT 1/2 pro desku Due

FFT_01 | Arduino 1.8.13 (Windows Store 1.8.42.0)

Soubor Úpravy Projekt Nástroje Nápověda

FFT_01 §

```
  analogReadResolution(12);
}

void loop()
{
  adcValue = analogRead(A0);
if (abs((micros()-cas2))>=1000)
  {
  cas2=micros();
  valuesVoltageArr[i]=adcValue * (3.3 / 4095.0);;
  valuesVoltageArr_img[i]=0;
  i=i+1;
  }
  if (i==samples)
  {
  Serial.println("Data:");
  PrintVector(valuesVoltageArr, samples, SCL_TIME);

   FFT.Windowing(valuesVoltageArr, samples, FFT_WIN_TYP_RECTANGLE, FFT_FORWARD);
     Serial.println("vážená data:");
     PrintVector(valuesVoltageArr, samples, SCL_TIME);
   FFT.Compute(valuesVoltageArr, valuesVoltageArr_img, samples, FFT_FORWARD); /*
      Serial.println("spocitane Real values:");
      PrintVector(valuesVoltageArr, samples, SCL_INDEX);
   FFT.ComplexToMagnitude(valuesVoltageArr, valuesVoltageArr_img, samples); /* C
       Serial.println("Computed magnitudes:");
       PrintVector(valuesVoltageArr, (samples >> 1), SCL_FREQUENCY);
   double x = FFT.MajorPeak(valuesVoltageArr, samples, samplingFrequency);
     Serial.println(x, 6);
  i=0;
   while(1);
  }
```

Obr. 13) Zdrojový kód FFT 2/2 pro desku DUE

Na obrázku Obr. 12 můžeme vidět začátek programu.Na začátku programu být použity příkazy **#include “arduinoFFT.h”** a **arduinoFFT FFt = ArduinoFFT().** Ty umožnují použití a spuštění knihovny na které byl program napsaný a které využívá. Konkrétně funkce **include()** zavolá příslušnou knihovnu a druhý příkaz pak vytvoří objekt daného typu ve kterém jsou schovány jednotlivé funkce této knihovny, odkud jsou poté volány.

Dále jsou definované proměnné. V první řadě jsou definované konstantní pomocí funkce **const** (tedy jejich hodnota se nemůže měnit). Zde je definovaný počet vzorků v cyklu samples=1024. Tato hodnota se liší od předchozích případů, protože jak již bylo zmíněno v kapitole 4.3. Počet vzorků FFT musí být vždy mocnina dvojky. V našem případě tedy 2 na 10. Dalšími konstantami jsou signalFrequency, samplingFrequency a amplitude. Ty jsou využívány při zpracování signálu. Poté je definována proměnná cas2, její využití je totožné z DC a RMS. Stejně tak totožné využití zde mají adcValue a i. Navíc jsou zde ještě definované pole hodnot o velikosti samples pro hodnoty reálných a imaginárních složek komplexních čísel s pojmenováním valuesVoltageArr a valuesVoltaArr_img.

Poté jsou pomocí funkce **#define** přiřazeny další čtyři konstantní proměnné v hexadecimální soustavě, SCL_INDEX, TIME, FREQUENCY A PLOT. Ty budou využity v programu na správné zobrazování výsledků. O jejich funkci lze získat více informací na stránkách knihovny[20]

Void setup() obsahuje stejné funkce jako programy DC a RMS, změnou zde je, že komunikace je nastavena na vyšší rychlost z důvodu složitosti programu. Dále se zde nachází příkaz **Serial.println()**, pro vypsání slova Ready při načtení programu. Abychom věděli, že program se nahrál správně.

Void loop(), obsahuje vše potřebné pro výpočet a zobrazení hodnot FFT. V Prvním kroku stejně jako u ostatních programů nejprve načte data z analogového vstupu pinu 0 a načte je za proměnnou adcValue. Následuje rozhodovací funkce **if().** Ta zde má stejný význam jako dříve. Tedy zajistit, aby se smyčka opakovala v pravidelných milisekundových intervalech. V dalším kroku se již hodnoty z převodníku převádějí na hodnotu ve Voltech a rovnou se načítají do pole reálných hodnot. Imaginární pole je zde uvedeno na hodnotu 0 a také probíhá iterace proměnné i, také stejně jak v obou předchozích programech. Poté následuje další podmíněná funkce **if()**. Jakmile tedy i dosáhne hodnoty 1024 spustí se sekvence pro výpočet FFT z načtených dat. Sekvence je nastavená tak, aby po výpočtu vždy na sériový monitor napsala, jaká data budou vypsány a hned pod jejich název vypíše vektor naměřených nebo spočítaných hodnot. Jako první rovnou vypíše data, která byla nahraná do pole reálných hodnot spolu se vzorkovacím časem. Poté pomocí funkce **FFT.windowing()** vytvoří data proložená okenní funkcí, tato funkce v našem případě není nezbytně nutná, protože správné vzorkování máme ošetřeno první funkcí **if().** V dalším kroku z těchto dat spočítá reálnou hodnotu FFT pomocí funkce **FFT.Compute().** Data ihned opět vypíše. V dalším kroku funkce **FFT.ComplexToMagnitude()** vypočítá absolutní hodnoty FFT v závislosti na frekvenci a hodnoty opět vypíše do sériového monitoru. V posledním kroku dosadí za proměnnou x pomocí funkce **FFT.MajorPeak()** hodnotu frekvence s největší hodnotou FFT. Tuto hodnotu nakonec zase vypíše. Nakonec nastaví proměnnou i na nulovou hodnotu pro možnost opakovat cyklus. Pro lepší přehlednost výsledků, aby se měření neopakovalo stále dokola, je ještě na konci této sekvence funkce **while()**, díky ní se měření provede pouze jednou. Pokud bych chtěl měřit stále dokola v klasické smyčce, stačí tento příkaz odebrat.

```
void PrintVector_frekv(double *vData, uint16_t bufferSize)
{
  int hodnota;
  for (uint16_t i = 0; i < bufferSize; i++)
    {
      if (i==0)
      {
      hodnota=1;
      }
      else
      {
      hodnota=2;
      }
     double abscissa;
     /* Print abscissa value */
         abscissa = ((i * 1.0 * samplingFrequency) / samples);
      Serial.print(abscissa, 6);
      Serial.print("Hz");
      Serial.print(" ");
      Serial.println(hodnota*vData[i]/samples, 4);
;
```

Obr. 14) Zobrazovací funkce upravených hodnot

Na obrázku Obr. 14 můžeme vidět část programu pro zobrazování upravených hodnot Fouriera. Pro zobrazení frekvence byla vytvořena vlastní funkce, která byla upravena na základě původní funkce na výpis tak, aby ukazovala přepočtenou hodnotu amplitudy dle následujících podmínek. Kde hodnota 1024 je hodnota proměnné samples.

$$\text{Pro nultou složku výpočet} \cdot \frac{1}{1024} \tag{15}$$

$$\text{Pro všechny další složky výpočty} \cdot \frac{2}{1024} \tag{16}$$

Tato funkce je potřeba z důvodu otestování a porovnání programu s hodnotami v excelu, které již mají hodnoty přepočteny tímto způsobe. Tím zajistíme porovnatelnost. Pro účely zobrazení dat v sériovém plotru budeme stále využívat zobrazovací funkci původní knihovny. Jak již bylo zmíněno na začátku této kapitoly, tato část byla převzata z původní knihovny, proto se odkazuji na stránku vývojářů pro více informací. [21]

6 OTESTOVÁNÍ PROGRAMŮ

Pro testování programů byli využity desky Arduino Leonardo Pro Micro, Uno a Due. Pro které byl vybrán zdroj signálu, který bude dál popsán.

6.1 Zdroj signálu

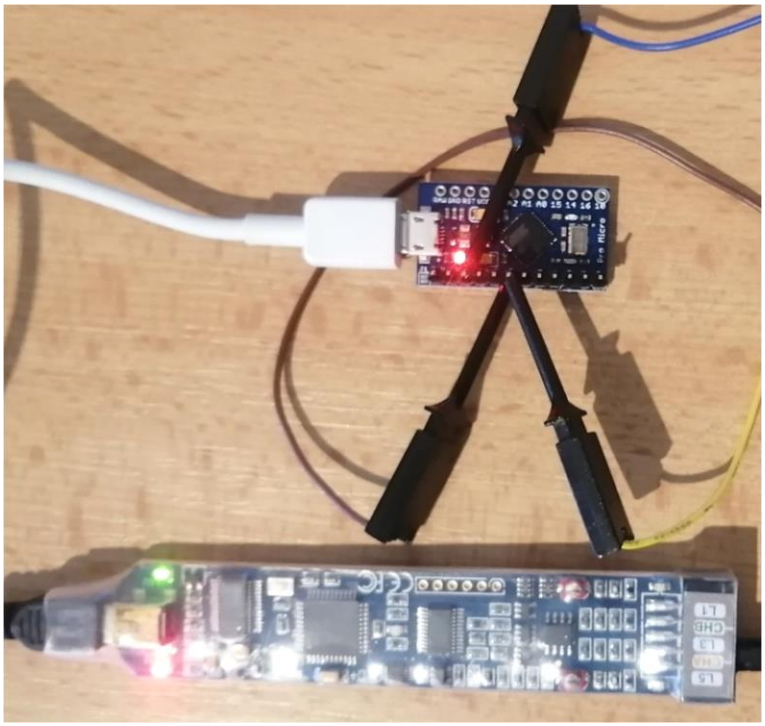
V prvním kroku pro úspěšné splnění zadání bakalářské práce je vybrat si a připojit k využívaným deskám Arduino zdroj signálu, který budeme dále zpracovávat.

6.1.1 BitScope Micro Analyzer and Scope

BitScope "Micro" Model 5 je první osciloskop pro smíšený signál, který obsahuje výkonný analyzátor logického protokolu, generátor průběhů & vzorů, spektrální analyzátor a záznamník dat v jednom malém, lehkém a vodotěsném balení napájeném z USB. Je plně uživatelsky programovatelný. [14] Osciloskop generuje libovolné průběhy analogového signálu, díky čemuž můžeme zkoušet aplikovat níže popsané programy pro výpočty střední a efektivní hodnoty a FFT na různé signály. Zároveň lze pomocí software Bitscope DSO v reálném čase signál měnit případně upravovat jeho hodnoty.

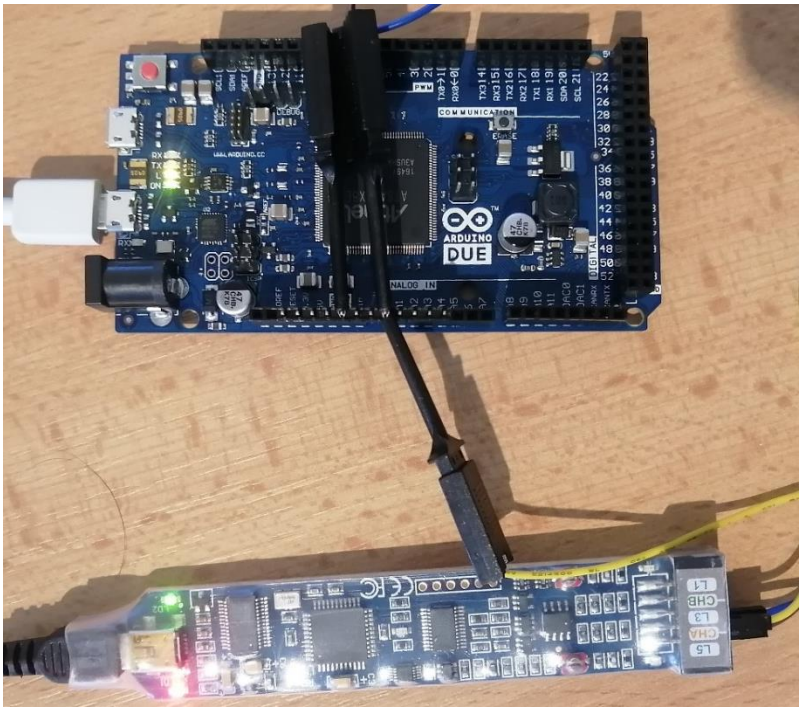
6.1.2 Připojení Arduina k Bitscope

Na obrázku Obr. 15 lze vidět připojení desky Arduino Mico. Připojení je jednoduché. Je provedeno pomocí rychle upínacích konektorů s výsuvnými čelistmi. To umožňuje rychlé přepojení a možnost napojení bez trvalého připájení. Modrým kabelem jsou propojené piny GND Arduina a Biscope. Hnědý kabel je připojený na pinu AWG tedy osciloskopu a k Arduinu připojen na libovolný pin pro analogový vstup. Posední žlutý kabel je na Arduinu připojený ke stejnému pinu jako osciloskop. K Bitscope je připojený na pin ChaA, tedy k prvnímu kanálu analyzátoru signálu. Díky tomu vidíme, jaký přesně signál nám do Arduina přichází v reálném čase v Bitscope DSO. [8]



Obr. 15) Připojení Arduino Leonardo Pro Micro k Bitscope

Na obrázku Obr. 16 můžeme vidět připojení desky Due, kde samotné zapojení je stejné jako u předchozího Arduina Micro a to stejné připojení bude využito i pro desku Arduino Uno.



Obr. 16) Připojení Arduino Due k Bitscope

6.2 Časování

Pro správnou funkci programu je potřeba vyřešit správně časování. Jedná se o nastavení programu tak, aby jeden cyklus programu trval jednu milisekundu. Tuto podmínku máme ošetřenou pomocí časové funkce **if()**, která je detailně popsána v kapitole 5 o tvorbě programu. Problém zde spočívá v tom, že pokud by deskám s menším operačním výkonem trvalo zpracování programu delší dobu, program by nemusel fungovat správně. Pro hlubší pochopení bude dál popsán způsob, jak zjistit dobu provedení jednoho cyklu našich programů jednotlivými deskami.

ReadAnalogVoltage

```
int i=0;
int samples=1000;
int sum=0;
float dc=0;
float adcValue = 0;
unsigned long cas1;
unsigned long cas2;
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
//  analogReadResolution(12);
}
// the loop routine runs over and over again forever:
void loop()
{
  cas1=micros();
  adcValue = analogRead(A0);

  if (abs((micros()-cas2))>=1000)
  {
   cas2=micros();
   sum=sum+adcValue;
   i=i+1;
  }
  if (i==samples)
  {
    dc= sum/samples;
    sum=0;
    i=0;
  float voltage = dc * (3.3 / 4095);
  Serial.println(voltage);
  cas2=micros()-cas1;
  Serial.println(cas2);
}
}
```

Obr. 17) časování programu DC hodnoty

Postup si ukážeme na programu pro výpočet střední hodnoty. Protože je program jednoduchý, je celý cyklus proveden za několik mikrosekund. To by mohl být problém, protože pokud by běžel znovu ve smyčce, vykresloval by pouze stejné hodnoty stále dokola.

Na obrázku Obr. 17 můžeme vidět, jak bylo zjištění délky zpracování a vypsání provedeno. Pro účely této části práce musí být upravený zdrojový kód pro výpočet hodnoty DC. Tato úprava byla provedena pro všechny desky, abychom mohly tyto časy zjistit, ovšem program s touto úpravou již nebude zahrnut do příloh, protože se jedná o jednoduchou úpravu. Nejprve byla zavedena proměnná čas1. Na začátek smyčky programu bylo pomocí funkce **micros()** zadáno, aby se začal počítat čas 1. Na konec smyčky výpočtu byli přidány příkazy pro výpočet délky načtení, zpracování a vypsání posledního vzorku signálu. Tento čas je nakonec zobrazen na sériový monitor. Jak jsem již zmínil, tento postup byl použit pro všechny programy na všech deskách Arduino využitých v této práci s výjimkou programu FFT. Zde nebylo zahrnuto vypsání dat na sériový port pro jejich rozsáhlost by to zabralo příliš času.

Do tabulky 1 níže jsou vypsány časy zpracování programů jednotlivými deskami.

Tab. 1) Délka zpracování programů jednotlivými deskami

Deska	DC hodnota	RMS hodnota	FFT
Micro	450µs	505 µs	--
Uno	430 µs	480 µs	--
Due	80 µs	102 µs	132 000 µs

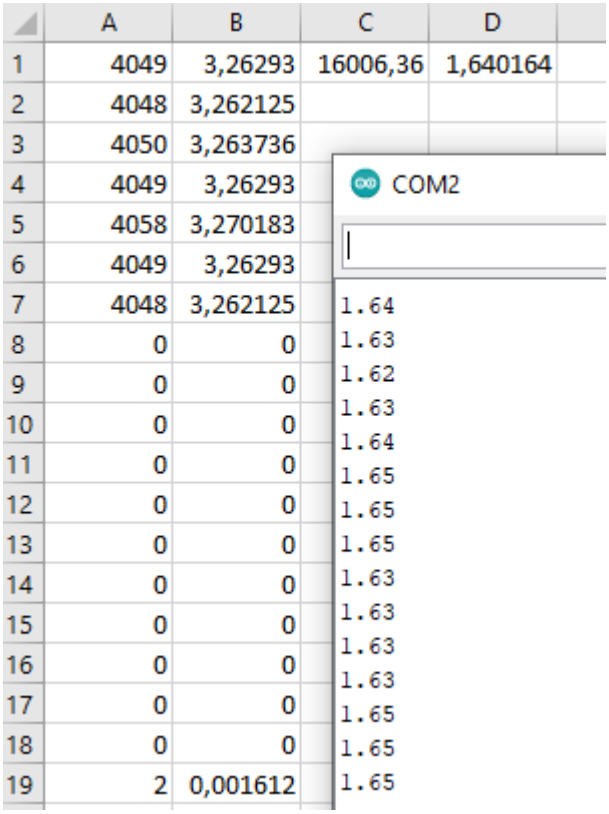
Pozor, pro lepší přehlednost nebylo v tabulce zahrnuzo zpracování všech námi požadovaných vzorků(samples). K těmto časům pak ještě musíme pro DC a RSM 1000 vzorků tedy 1sekunda a pro FFT 1024 vzorků, tedy 1,024sekundy. Tento čas vychází z dříve popsané podmínky na délku trvání jednoho vzorku jednu milisekundu. Výsledný čas pro provedení celých programů je tedy součet hodnot z tabulky s časem všech vzorků.

Z tabulky vyplívá, že je mezi mikrokontrolery na deskách Micro a Uno není příliš velký rozdíl, protože časy zpracování se liší pouze v řádu desítek mikrosekund. Zároveň ani jedna z desek nezvládla zpracovat program FFT, obě hlásili malou dynamickou paměť pro daný problém. Naopak jde jasně vidět, že deska Arduino Due je mnohonásobně výkonnější, rozdíl u jednoduchých programu DC a RMS je v řádu stovek mikrosekund.

6.3 Střední hodnota (DC)

Pro testování programu na vyhodnocení střední hodnoty byl použit obdélníkový vstupní signál v nastavení 20 Hz. Pro toto nastavení ukazovali všechny tři Arduino desky výsledek pohybující se okolo hodnoty 1,65.

Výsledek byl ověřen přepočtením hodnot převzatých z AD převodníku Arduina pomocí excelové tabulky dle vzorce č.(7). Tato hodnota odpovídá. Máme tedy jistotu, že výpočet v programu je prováděn správně. Vidět to můžeme na obrázku Obr. 18 pozici D1.



	A	B	C	D
1	4049	3,26293	16006,36	1,640164
2	4048	3,262125		
3	4050	3,263736		
4	4049	3,26293		
5	4058	3,270183		
6	4049	3,26293		
7	4048	3,262125		
8	0	0		
9	0	0		
10	0	0		
11	0	0		
12	0	0		
13	0	0		
14	0	0		
15	0	0		
16	0	0		
17	0	0		
18	0	0		
19	2	0,001612		

Obr. 18) Výpočet DC z hodnot AD převodníku

Dle výpočtu střední hodnoty pomocí střídy (rovnice č. 17), což je poměr mezi délkou pulzu a délkou celé periody. Se nám tento výsledek ověřil. Hodnota středního napětí dle tohoto vzorečku vychází 1,65, tedy jsme ověřili správnost fungování tohoto programu pro všechny používané desky Arduino.

$$s = \frac{\tau}{T} \tag{17}$$

Důležité pro správné výsledky je mít správně nastavený program, protože Arduino Micro a Uno pracují obě na 10bitovém AD převodníku s max napětím 5 Voltu. Kdežto Due disponuje 12bitovým AD převodníkem, ale jeho maximální napětí může být pouze 3,3 Voltu.

Tomuto je tedy potřeba podřídit nastavení programu. Pro desky Micro a Uno použijeme převod voltage = dc · (5/1024) Pro desku Due pak bude převod na Volty ve tvaru voltage = x · (3,3/4095) Tedy vynásobí výslednou digitální hodnotu počítané veličiny a pro přepočet na volty ji vynásobí závorkou. Ta obsahuje na prvním místě maximální hodnotu napětí, druhá hodnota je maximální hodnota (úroveň) na AD převodníku. Toto platí pro všechny programy a dále už tato informace nebude opakována, nicméně je potřebné na to stále myslet.

6.4 Efektivní hodnota (rms)

Pro testování programů na vyhodnocení efektivní hodnoty bylo využito Referenčního osciloskopu se schopností měřit a rovnou zobrazovat efektivní hodnotu vstupního signálu. Připojili jsme tedy tento osciloskop na zdroj signálu zároveň s Arduinem a porovnávali hodnoty které ukazují. Všechny hodnoty budou v následující tabulce.

Tabulka 2) Výsledky hodnot RMS

	Obdélníkový signál	**Sinusový signál**	**Rampový signál**
Referenční osciloskop	2,37 V	2,07 V	1,95 V
Arduino Due	2,32 V	2,03 V	1,89 V
Arduino Uno	2,41 V	2,08 V	1,95 V

Protože desky Uno a Micro mají stejný mikrokontroler rozhodli jsme se odzkoušet variantu Uno, protože Micro má malé piny na připojení konektorů.

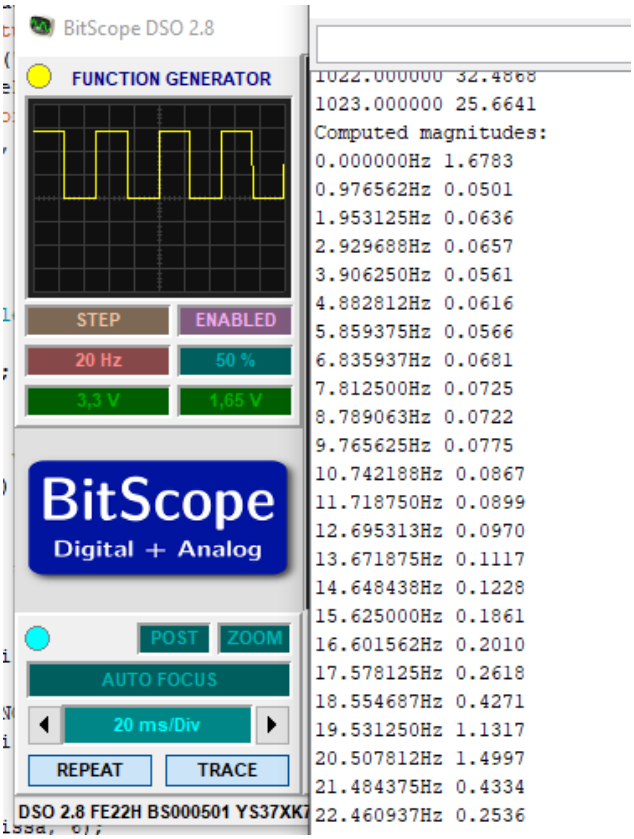
Jak je dle tabulky patrné, hodnoty Efektivních hodnot se liší pouze minimálně od hodnot naměřených na deskách Arduino. Malé rozdíly ve výsledcích jsou pravděpodobně způsobeny přesností měření na AD převodníku Arduina a osciloskopu. Tento program je tedy také ověřen a zcela funkční.

6.5 FFT

Jak již bylo dříve zmíněno, desky Micro a Uno bohužel nedokážou tento program nahrát z důvodu malé paměti, proto musí být odzkoušen pouze na desce Due. Pro komplexní odzkoušení budeme testovat tři druhy signálu. První bude sinusový, druhý signál bude rampový a poslední pak obdélníkový. Pro každý druh signálu budou porovnány tři frekvence :20 Hz, 50 Hz a 200 Hz.

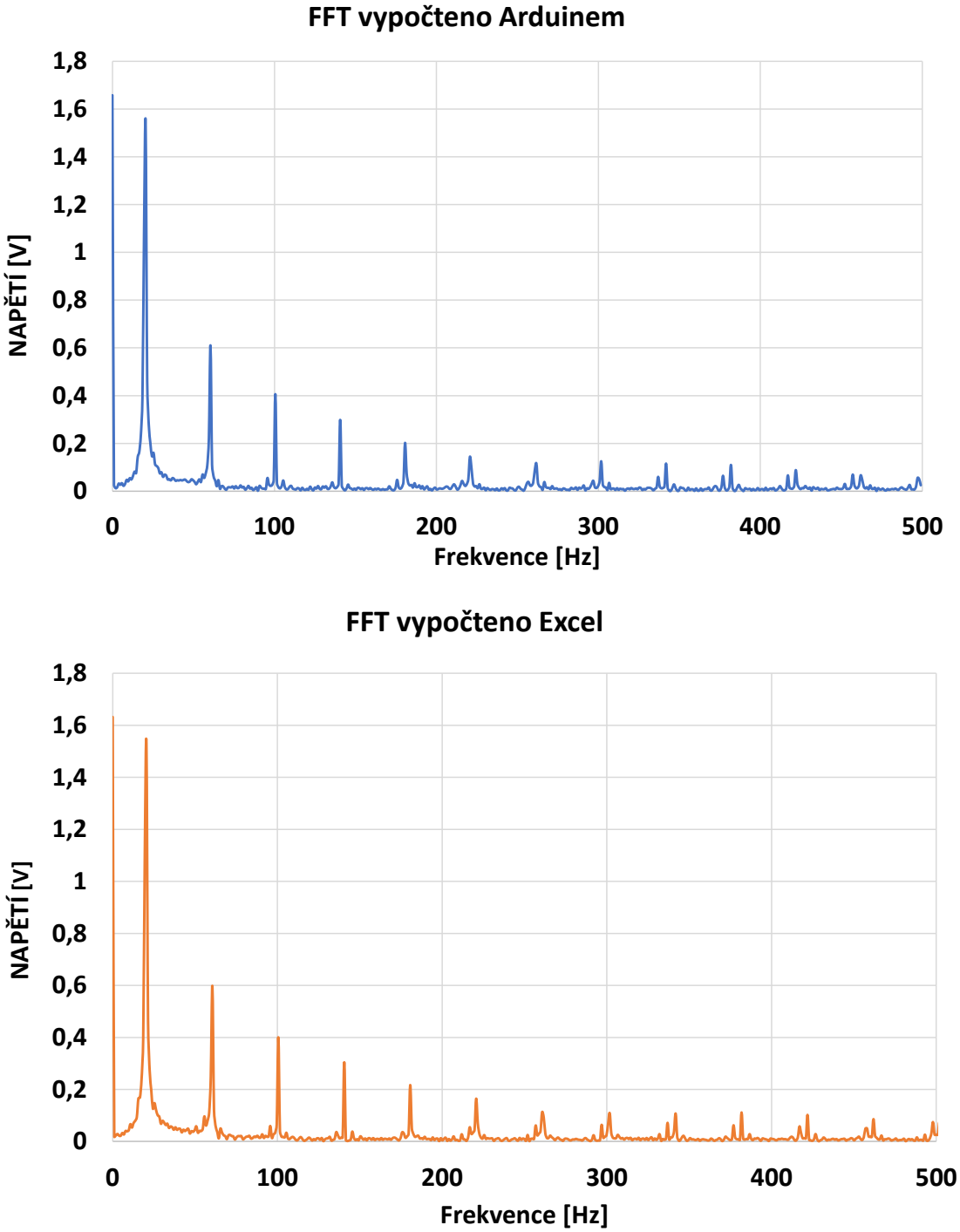
Pro ověření správnosti výpočtu algoritmu FFT prováděných pomocí desky Due bylo využito tabulek Excelu. Ty umožňují doinstalování doplňku s názvem Analytické nástroje, ve kterých nalezneme právě algoritmus FFT. Pro výpočet přes Excel budeme potřebovat nejprve načíst data z AD převodníku vyčtené ze sériového monitoru. Pomocí funkce **PrintVector(valuesVoltageArr, samples, SCL_TIME)**. Požadované hodnoty vyplníme do Excelu a ten nám postupně dopočítá komplexní čísla, frekvenci v Hz a amplitudu. Tento postup zopakujeme pro všechny frekvence a druhy signálů. Následuje ještě přepočet hodnot vysvětlený na konci kapitoly 5.3. Postup otestování programu je tedy zkontrolovat, jestli Arduino dokáže správně vypočíst frekvenci maximálního vrcholu a porovnat tuto hodnotu s Excelem a vstupním signálem.

Na obrázku Obr. 19 vidíme vlevo okno generátoru Bitscope kde můžeme vyčíst nastavenou frekvenci a hodnotu 1,65V a vpravo hodnoty v sériovém monitoru prostředí Arduino IDE. Správnost výpočtu ověřujeme prvně nultou frekvencí, kdy hodnota FFT musí vystřelit na střední hodnotu, ta se liší pouze v řádu setin voltu což je v pořádku. Jako druhý znak pak musíme pozorovat, zda frekvence 20 Hz odpovídá největšímu peaku. Toto taktéž sedí protože vidíme, že na dané frekvenci je opravdu nejvyšší napětí vyjma skoku na nulté frekvenci takže tyto hodnoty jsou správně.



Obr. 19) Výsledky výpočtu Arduina a porovnání se vstupním signálem

Nyní je potřeba ověřit, zda hodnoty v Excelu vycházejí stejně jako v Arduinu. To bude nejlépe vidět při porovnání grafů vytvořených z dat vypočtených oběma způsoby. Vložené grafy jsou vyhodnocení obdélníkového signálu s frekvencí 20 Hz.



Obr. 20) Grafy hodnot FFT pro porovnání Arduino a Excel

Z grafů lze vyčíst, že hodnoty se hodnoty přibližně shodují.

Stejným postupem, jak je naznačeno byli zkontrolovány všechny typy signálů a frekvencí uvedených na začátku této kapitoly. Všechny hodnoty seděli a byli správné, tento program tedy můžeme také považovat za úspěšně otestovaný.

7 ZÁVĚR

Cílem bakalářské práce bylo v prvním kroku seznámit se s platformou Arduino. Zde se jednalo především o vytvoření všeobecného přehledu, jak tato platforma funguje a jaké přináší možnosti, rozdělení druhů desek a jejich praktického použití.

V dalším kroku bylo nutné seznámit se s problematikou matematiky a postupem výpočtů zabývajících se zpracováním signálu. Tyto výpočty zpracování signálu pochopit do takové míry, aby mohly být aplikovány v praktické části této bakalářské práce. Především s výpočtem střední hodnoty, efektivní hodnoty a Fourierovi transformace.

Praktická část práce se zabývá vytvořením fungujících programů pro výpočet těchto hodnot. Tato část práce byla nejnáročnější, především pak tvorba algoritmu FFT, pro který musela být využita předinstalovaná knihovna v prostředí Arduino IDE. Problematická byla, jak na pochopení principu, tak na její přeprogramování. Změnou zdrojových kódů bylo dosaženo schopnosti programu načítat příchozí data z osciloskopu, ukládat si je do datových polí a dále je vyhodnocovat. Díky funkci na vypsání do sériového monitoru, kterým disponuje vývojové prostředí Arduino IDE, je zobrazení zpracovaných dat velice jednoduché.

Posledním krokem bylo ověřit funkčnost programů. Na všech typech desek, které byly v této práci použity proběhlo testování vytvořených programů. Všechny tyto programy jsou zcela funkční. Tedy až na FFT, kde slabší typy mikrokontrolerů na deskách Micro a Uno tento algoritmus nedokážou zpracovat z důvodu malé dynamické paměti, kterou disponují.

Dokonce i deska Due má své omezení. Viz tab.1) kde čas FFT programu je 0,135 sekundy. Když k této hodnotě přičteme čas načtení všech 1024 vzorků, dostaneme se na celkový čas 1,159 sekundy na zpracování relativně malého vzorku. Kdyby množství vzorků bylo větší, čas na zpracování by byl mnohonásobně větší.

Jedná se ale o obecný problémem algoritmů podobného typu vlastně na všech platformách, na kterých je můžeme zpracovávat. Arduino je proto obecně vhodná platforma na opravdu široké využití dokonce i v takto náročných použitích jako bylo zde testováno.

Výstupem této bakalářské práce je posouzení využitelnosti platformy Arduino v oboru zpracování signálu. Velký nedostatek platformy je v tom, že není možné softwarově řešit offset signálu. V případě, že bychom měli zpracovávat bipolární signál, tedy signál pohybující se od záporných do kladných hodnot. Výsledky, které by programy uváděly by byly špatně. Řešením může být využití externí elektroniky pro úpravu offsetu do tvaru signálu nad nulovou osu. Konkrétní tak v případě využívaných desek Micro a Uno by se pak jednalo o rozsah 0-5 Voltu. V případě desky Due se pak jedná o hodnoty 0-3,3 Voltu.

8 SEZNAM POUŽITÝCH ZDROJŮ

[1] WIKIPEDIA. Arduino [online]. [cit. 2016-5-5]. Dostupné z: https://cs.wikipedia.org/wiki/Arduino

[2] Creative Commons Attribution-ShareAlike 3.0 License. Arduino IDE [online]. [cit. 2016- 5-5]. Dostupné z: http://arduino.cz/arduino-ide/

[3] ZIMEK, Tomáš. Arduino: programování v prostředí Matlab/Simulink. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 77 s. Vedoucí bakalářské práce doc. Ing. Radomil Matoušek, Ph.D.

[4] Humusoft. Simulink. Humusoft [online]. Humusoft 1991-2020 [cit. 2020-05-07]. Dostupné z: https://www.humusoft.cz/matlab/simulink/

[5] Arduino [online]. 2020 [cit. 2020-04-10]. Dostupné z https://www.arduino.cc/.

[6] Wiring [online]. 2020 [cit. 2020-04-10]. Dostupné z https://www.wiring.org.co/.

[7] VODA, Zbyšek a tým HW Kitchen. Průvodce světem Arduina. 2. vydání. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.

[8] SVOBODA, Aleš. Vše o napájení Arduina. In navody.arduino-shop.cz [online]. 2018-09-03 [cit. 2020-04-10]. Dostupné z: https://navody.arduino-shop.cz/technikuv-blog/napajeniarduina.html/.

[9] SELECKÝ, Matúš. *Arduino: uživatelská příručka*. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 978-80-251-4840-2.

[10] Mathworks. Articles. A Brief History of MATLAB [online]. Mathworks, 2018 [cit 2020-04-20] Dostupné z https://www.mathworks.com/company/newsletters/articles/a-brief-historyof-matlab.html

[11] Barragán, H. The Untold History of Arduino [online]. Dostupné z: https://arduinohistory.github.io.

[12] TŮMA, Jiří. *Zpracování signálů získaných z mechanických systémů užitím FFT*. Praha: Sdělovací technika, 1997. ISBN 80-901936-1-7.

[13] BRANČÍK, Lubomír. *Elektrotechnika 1*. Brno: VUTIUM, 2004. ISBN 80-214-2607-.

[14] An Avnet Company. *Cz.farnell* [online]. Anglie, 2020 [cit. 2020-09-10]. Dostupné z: h12ttps://cz.farnell.com/bitscope/bs05u/oscilloscope-logic-analyzer-2/dp/2432906

[15] loudimg [online]. 2020 [cit. 2020-09-09]. Dostupné z: https://ce8dc832c.cloudimg.io/width/340/foil1@886bcaf43e476b2bd42e4ac2fc3ab9aba5e27f1c/_cs_/2020/05/5eccde629dc20/arduino_logo.png

[16] alicdn [online]. 2020 [cit. 2020-08-10]. Dostupné z https://ae01.alicdn.com/kf/Hed8c075ce0eb46f68c780564b70fc8e7H/Nano-Board-CH340-ATmega328P-Without-USB-Cable-Compatible-with-Arduino-Nano-V3-0-Without-Cable.jpg_q50.jpg

[17] tecwizdom [online]. 2020 [cit. 2020-09-07]. Dostupné z: https://www.tecwizdom.com/wp-content/uploads/2018/11/ARDUINO-DUE-1.jpg

[18] distrelec [online]. 2020 [cit. 2020-07-09]. Dostupné z. https://www.distrelec.cz/Web/WebShopImages/landscape_large/9-/01/arduino-a000066.jpg

[19] katodo [online]. 2020 [cit. 2020-06-09]. Dostupné z. https://katodo.com/228/arduino-micro-atmega32u4-compatible-.jpg
[20] *GitHub,Inc.* [online]. 2020 [cit. 2020-09-10]. Dostupné z: https://github.com/kosme/arduinoFFT
[21] VODA, Zbyšek a tým HW Kitchen. *Průvodcem světem Arduina*. 1. Bučovice: Martin Stříž, 2018.

8 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

8.1 Seznam zkratek a symbolů

Zkratka – Význam

cos – cosinus

DC - střední hodnota veličiny

DFT – Fourierova transformace

f – kmitočet

FFT – rychlá Fourierova transformace

F_k – koeficient Fourierovy řady

GND – zkratka uzemnění v elektronice (z anglického ground)

If – rozhodovací funkce

k – libovolné periodické číslo

m + - kladná maximální hodnota

m- - záporná maximální hodnota

mA – jednotka elektrického proudu miliampér

max – maximální hodnota

MHz – jednotka frekvence – megahertz

Rad jednotka rovinného úhlu – radián

RMS – efektivní hodnota (root-mean-square) veličin

sin – sinus

T – Perioda

USB – universální sériová sběrnice

V – jednotka elektrického napětí – volt

xm - amplituda

π – matematická konstanta Ludolfovo číslo

ω – úhlová frekvence

ψi - konstanta – počáteční fázový úhel

8.2 Seznam Obrázků

8.3 Seznam tabulek

9 SEZNAM PŘÍLOH

1. Zdrojové kódy