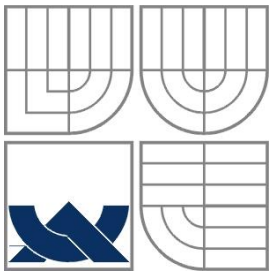
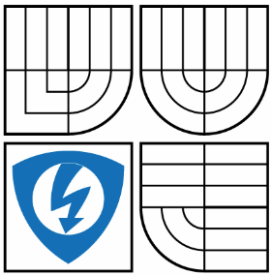


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SBĚR A ZPRACOVÁNÍ DAT S VYUŽITÍM MIKROKONTROLERU ARM

BAKALÁŘSKÁ PRÁCE

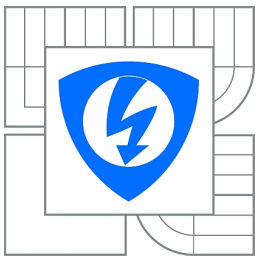
AUTOR PRÁCE
AUTHOR

Ondřej Bartík

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Petr Petyovský

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Ondřej Bartík	***ID:***	146782
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Sběr a zpracování dat s využitím mikrokontroleru ARM

POKYNY PRO VYPRACOVÁNÍ:

Cílem studenta je navrhnout a implementovat vhodnou demonstrační úlohu z oblasti měření a zpracování dat pomocí mikrokontroleru třídy ARM.
1. Seznamte se s dostupnými vývojovými kity obsahujícími mikrokontroler ARM a zhodnoťte jejich možnosti.
2. Zvolte vhodný vývojový kit a nastudujte vlastnosti jednotlivých periferií a možnostmi jejich optimální vzájemné spolupráce.
3. Navrhněte vhodnou úlohu z oblasti sběru a zpracování dat pomocí mikrokontroleru.
4. Proveďte rozbor efektivního využití jednotlivých periferií s ohledem na: propustnost dat, výpočetní výkon a spotřebu.
5. Implementujte a odlaďte úlohu jako aplikaci pro zvolený vývojový kit.
6. Realizujte experimentální měření parametrů aplikace pro sběr a zpracování dat.
7. Ověřte, zda implementace úlohy odpovídá teoretickým předpokladům v bodě 4.
8. Zhodnoťte dosažené výsledky a diskutuje problémy vzniklé při implementaci.

DOPORUČENÁ LITERATURA:

[1] Váňa, V.: ARM pro začátečníky, 1. vyd. BEN 2009, 196 s. ISBN 978-80-7300-246-6
[2] Virius, M. : Jazyky C a C++ (2. vydání), Grada 2011, ISBN 80-247-3917-5

Termín zadání:	10.2.2014	***Termín odevzdání:***	26.5.2014

Vedoucí práce: Ing. Petr Petyovský
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

Abstrakt

Bakalářská práce se zaměřuje na možnost využití mikroprocesoru s architekturou ARM Cortex-M3 pro sběr a zpracování dat. Práce se zabývá srovnáním možných vývojových prostředků a určení toho nejvhodnějšího pro zadaný úkol. Dále je vybrána jedna úloha a v rámci práce je ukázána její praktická realizace.

Abstract

This Bachelor thesis is about the possible using of microprocessor with ARM Cortex-M3 architecture for data acquisition and data processing. Thesis is also about comparison of possible development tools and choosing the most suitable for assigned task. There is also chose one task and possible implementation is shown.

Klíčová slova

Data logger pro měření zrychlení a rychlosti vibrací, ARM, Cortex-M3

Key words

Data logger for acceleration and speed of vibrations acquisition, ARM, Cortex-M3

Bibliografická citace:

BARTÍK, O. Sběr a zpracování dat s využitím mikrokontroleru ARM. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 28 s. Vedoucí bakalářské práce Ing. Petr Petyovský.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Sběr a zpracování dat s využitím mikrokontroleru ARM jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **21. května. 2014** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Petyovskému. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce. Bez jeho pomoci by tato bakalářská práce nenabyla takovéto finální podoby

V Brně dne: **21. května 2014** ……

podpis autora

Obsah

1 ÚVOD

Práce se zabývá sběrem a zpracováním dat s využitím mikrokontroleru ARM s architekturou jádra Cortex-M3 za použití vhodného vývojového kitu. Cílem práce je vybrat vhodný vývojový prostředek, kit a navrhnout vhodnou úlohu, kde se uplatní právě sběr a zpracování dat za použití mikrokontroleru. Dalším bodem zájmu práce je prostudovat možnosti vývojového prostředku a navrhnout co nejefektivnější realizaci úlohy s ohledem na maximální využití možností, které vývojový kit s použitým mikrokontrolerem nabízejí. A poslední částí je i zadanou úlohu realizovat. Realizací se rozumí co nejefektivnější využití periferií, které použitý mikrokontroler nabízí a to z hlediska výkonu ve smyslu datové propustnosti a energetické úspornosti. V rámci práce bude vybrán jeden vývojový prostředek s konkrétním mikroprocesorem s jádrem ARM Cortex-M3 u nějž budou využity všechny možnosti, které jsou vhodné pro zadanou úlohu. Dále budou v práci, detailně pospány jednotlivé části výsledné aplikace. Práce obsahuje nejprve části o výběru vhodného prostředku, dále pak část o výběru konkrétní úlohy a dále části popisující teoretický návrh jednotlivých částí celé aplikace a nakonec i části popisující praktickou realizaci celé aplikace odkazující na elektronickou přílohu práce. Práce poskytuje čtenáři pohled na konkrétní způsob realizace zadaného problému, jeho rozbor i jeho řešení. Výsledná aplikace bude realizována pomocí programovacího jazyka C. Aplikace bude navrhnuta tak aby byla použitelná v samostatném čipu, který byl vybrán a nebyla pak závislá na použitém vývojovém prostředku, kitu. V práci bude taktéž poukázáno na možné problémy v průběhu vývoje a řešení zadané úlohy a bude nastíněno jejich možné řešení. Práce tak nabízí jedno z mnoha využití architektury procesorů Cortex-M3 v oblasti sběru a zpracování dat. Práce taktéž zhodnocuje možnosti vybraného vývojového prostředku pro danou úlohu a to jak z pohledu nastínění teoretické realizace, tak i z pohledu praktické realizace. Účelem práce je taktéž čtenáře seznámit s použitou architekturou s její historií a jejími možnostmi.

2 PROCESORY ARM

2.1 Historie architektury ARM

První procesory ARM (*Advanced Risch Machine*) spatřily světlo světa v roce 1985 a byly vyvinuty firmou ARM Limited. Počínaje tyto čipy znamenaly obrovský skok jak v informačních technologiích, tak i v mnoha dalších sférách elektrotechniky a techniky vůbec. Už v tomto roku měly čipy 32 bitovou šířku slova a jejich energetická spotřeba byla nízká[1].
Robustní výpočetní výkon a možnosti využití spolu s nízkými nároky na napájení přenesly obvody do sféry mobilních aplikací. Dnes je nenajdeme ovšem jen ve valné většině mobilní elektrotechniky jako jsou mobilní telefony, tablety, fotoaparáty a mnoho dalších, ale i v drtivé většině spotřební elektrotechniky od kávovarů přes televizory až k pračkám. A to hlavně díky nízké výrobní ceně, široké škále modelů a velkému množství použitých architektur, které určují konečné možnosti využití čipu. Dnes se firma ARM Limited zabývá pouhým návrhem architektury jádra obvodu a tu pak v podobě licence prodává výrobcům čipů, kteří k návrhu jádra přidají další aspekty výsledného čipu, jako jsou integrované periferie na čipu spolu se samotným procesorem a zajistí fyzickou výrobu obvodu.
V historickém vývoji obvodů s architekturou ARM můžeme vidět několik různých rodin. Z toho v každé můžeme najít několik verzí architektur. Ale všechny rodiny mají jedno společné, všechny jsou založeny na RISC architektuře (hlavní znakem je menší instrukční sada obsahující univerzální instrukce, viz [25]). Dnes nejvíce používaná rodina obvodů nese označení Cortex. V této rodině můžeme nalézt několik verzí architektur, ale všechny se dají roztřídit do třech kategorií, viz dále.

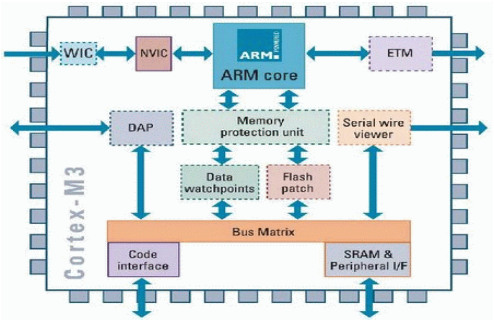
2.2 Cortex rodina obvodů

Jak již bylo zmíněno výše, obvody rodiny Cortex se dají roztřídit do tří skupin.
A to podle aplikace, pro kterou jsou určeny. Rozlišují se celkem tři kategorie a každá z nich se v označení jádra rozlišuje písmeny A, R, nebo M [2].
Architektury nesoucí označení Cortex-A jsou určeny pro robustní aplikace jako je například operační systém nebo náročné výpočetní úkony. Tyto obvody nalezneme všude tam, kde je potřeba výpočetní výkon, jako například v mobilních zařízeních s operačním systémem. Nalezneme je i ovšem v zařízeních, která využívají signálových procesů k zpracování zvuku nebo obrazu. Například hudební přehrávače, video přehrávače i rekordéry, nebo i televize.
Další kategorie označována písmenem R slouží pro aplikace, které vyžadují Real-Time procesy a zpracování dat. Tyto obvody nacházejí převážně využití v průmyslu, kde je Real-Time proces vyžadován.

Třetí a poslední skupinou, jsou obvody označovány Cortex-M a slouží převážně pro aplikace, kde je kladen důraz na nízkou energetickou spotřebu a není vyžadován maximální výkon. Tyto obvody se používají u aplikací, na které verze A a R byly zbytečně robustní. Obvody s touto architekturou nalezneme takřka všude. I v prostých domácích spotřebičích a vůbec ve zbytku spotřební elektroniky. Nicméně nacházejí uplatnění i v průmyslu. I když jsou tou nejslabší kategorií ze všech, co do výpočetního výkonu, stále jsou dostatečně robustní například pro řídicí aplikace, telekomunikační aplikace, nebo i pro aplikace v měření různých fyzikálních veličin.
Jelikož v práci byla použita právě verze Cortex-M, konkrétně Cortex-M3, je tato architektura podrobněji rozebrána a je ji věnován i následující text.

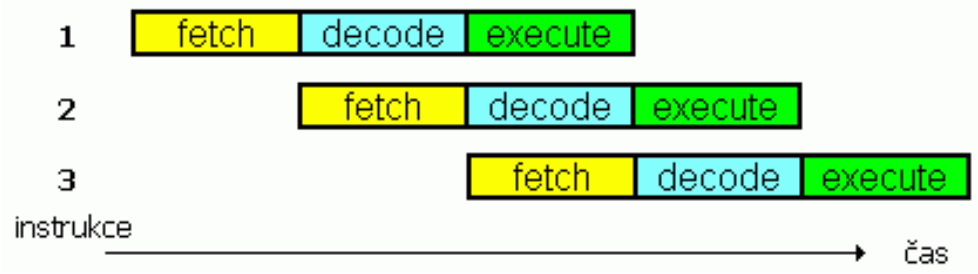
2.3 Architektura ARM Cortex

[2]Dosažení zvýšení výkonů mikroprocesorů lze docílit dvěma způsoby.
Prvním způsobem je zvýšení taktovací frekvence, ale to má za následek zvýšení spotřeby obvodu a tím pádem i zvýšení ztrát a složitosti napájecích obvodů. Druhým a efektivnějším způsobem je zlepšení algoritmů zpracovávání strojového kódu na nižší taktovací frekvenci a také zvýšení hustoty instrukcí na bajt paměti. Na základě těchto dvou myšlenek vznikla instrukční sada Thumb2. Oproti svému předchůdci má sice značné výhody, ale díky zmenšení strojové instrukce postrádá možnost tří operandových zápisů instrukcí. Tento zápis instrukce umožňuje výsledek operace ukládat do třetího operandu a zamezuje tak případné ztrátě dat, která je u použití dvou operandové instrukce nevyhnutelná, protože výsledek operace je zapsán do jednoho z operandu a tím je přepsán původní obsah. To může mít za následek v některých případech potřebu použití další instrukce navíc pro uchování dat. Nicméně snížení velikosti instrukčního slova a zvýšení efektivnosti při jeho zpracování má za následek zvýšení výkonu oproti svému přímému předchůdci v řadě minimálně o 45% a snížení objemu kódu minimálně o 28%. Dalším důležitým kritériem je výrobní cena. Moderní sofistikované obvody vyžadují moderní výrobní metody a technologie, které jsou však drahé. A proto se jádra procesorů vyrábějí o maximálním počtu tranzistorů, čítající třicet tři tisíc a jako použitá výrobní technologie je maximálně 0,18 mikronů. Okolí čipu taktéž snížilo počet součástí na minimum. Díky tomuto, ale nejen tomu, klesla cena na minimální možnou úroveň a dnes je možné zakoupit čip s architekturou Cortex-M3 dokonce i levněji než některé starší osmibitové mikrokontrolery.



Obrázek 1: architektura Cortex-M3 [3]

Jádro celé architektury je z pohledu programátora realizováno architekturou Von Neuman, Z pohledu fyzické stránky je však architektura realizována Harvardskou architekturou. Architektura Von Neuman je taková, kde je společná paměť pro data a zvlášť paměť pro program. Tím pádem jsou zde i dvě vzájemně se neovlivňující sběrnice, instrukční a datová. Zpracování instrukcí probíhá ve třech krocích v tříúrovňovém systému tzv. *pipeline*, kde v prvním kroku je instrukce vyzvednuta z paměti, v dalším dekódována a v posledním je provedena [4].



Obrázek 2: systém pipeline [5]

Z obrázku je jasná výhoda tohoto systému, kde *fetch* značí vyzvednutí instrukce z paměti, *decode* pak dekódování a *execute* konečné vykonání. Zatímco už bylo provedeno vyzvednutí první instrukce a je právě dekódována, ta další může být vyzvednuta či vyzvedávána. A pokud je první připravena k vykonání tak systém zpracovává tři instrukce, kde první je vykonávána, druhá je dekódována a třetí je vyzvedávána. Tento systém spolu s ostatními výše popsanými výhodami dělá s architektury Cortex-M3 silný a robustní prostředek pro implementací široké škály aplikací.

3 SROVNÁNÍ VÝVOJOVÝCH PROSTŘEDKŮ

Prvním úkolem práce bylo seznámení s dostupnými vývojovými kity obsahujícími mikrokontroler s jádrem ARM, zhodnotit jejich vlastnosti a vybrat ten, který bude vyhovovat dalším kritériím a požadavkům práce. Byly srovnány celkem tři kity od třech různých výrobců Atmel, ST a Energymicro, obsahující mikroprocesor s architekturou a jádrem Cortex-M3.

3.1 Kit STM32 vl Discovery

Jako první byl zvolen vývojový kit od firmy ST, STM32VLDISCOVERY [6]. Tento vývojový prostředek je poměrně chudý co do výbavy periferií na desce. Ale mezi jeho bezkonkurenční přednost patří cena, ta se pohybuje v intervalu dvě stě až tři sta korun českých. Kit obsahuje čip STM32F100RB [7], který nabízí 128KB programovou paměť FLASH, několik standartních periferií: sériová rozhraní RS232, SPI, I2C, ale i USB s možností konfigurace jako zařízení typu HOST. Dále pak několik čítačů. Z toho sedm pracuje v 16 bitovém PWM módu a čip je tak podle výrobce vhodný pro řídicí aplikace. Dále jsou na čipu obsaženy samozřejmě analogové periférie v podobě analogových převodníků. Jelikož je tato práce zaměřena na sběr a zpracování dat, kde kvalita celého procesu sběru dat je závislá právě na výkonu analogových převodníků, tak je na tyto periferie srovnávaných kitů s mikrokontrolery kladen velký důraz.
Výše zmíněný čip disponuje dvěma AD převodníky, které je možno zapojit do kaskády a využít tak obou bitových rozsahů. Každý dílčí převodník může pracovat až s 12 bitovým rozlišením a vzorkovací frekvencí 1MHz. Tyto vlastnosti jsou standardní a platí pro AD převodníky i na jiných mikrokontrolerech, nejen od firmy ST. Bohužel maximálního výkonu převodníku nelze takřka využít. Praktická zkušenost ukázala, že při několika měřeních konstantního napětí, vznikla odchylka přibližně +- 100mV. A to i při ošetření vstupu článkem typu dolní propust s časovou konstantou 0,01s a tedy s mezní frekvencí přibližně 160Hz. Z tohoto důvodu tento kit nebyl zvolen. I když v úvahu přišla i možnost použití externího AD převodníku s mnohem lepšími vlastnostmi. Ale další průzkum ukázal, že další vývojové kity mají AD převodníky s lepšími vlastnostmi a dokonce mají ošetřené analogové vstupy vhodnými filtry proti nežádoucím šumu a rušení, což kit od firmy ST vůbec nemá a je třeba použít externí ošetření.
Ale mezi další výhody kitu kromě ceny patří například možnost programování externích čipů rodiny STM32F1xx. Programování čipu, nebo externích zařízení, je dosaženo pomocí ST-LINK rozhraní. Jedná se o ladicí, *debug* a programovací nástroj. Umožňuje komunikaci s čipem pouze přes rozhraní SWD (*Single Wire Debugging*). Jedná se o rozhraní sloužící pouze k programování FLASH pamětí. Oproti standartu JTAG (*Join Test Aplication Group*), které je hojně používán ke stejnému účelu, obsahuje kromě napájení jen dva pracovní vodiče. První slouží pro přenos synchronizačního hodinového signálu a druhý pak pro přenos dat. Díky použití pouze jednoho vodiče pro přenos dat je

možné zajistit pouze *half duplex* komunikaci. Naproti tomu rozhraní JTAG umožňuje *full duplex* komunikaci se zařízením, ale toto rozhraní neslouží jen pro programování FLASH pamětí, ale i pro testování vnitřních funkcí integrovaných obvodů. K tomu je zapotřebí příslušné software prostředí, které má znalost vnitřní struktury testovaného obvodu a na základě změřených proudů, napětí vyhodnocuje stav a funkčnost obvodu. Celé programovací rozhraní (*SWD*) je implementováno do dalšího obvodu s jádrem ARM a taktéž od firmy ST. Jedná se o čip STM32F103C8T6.
Jak už bylo zmíněno, kit nemá mnoho periferií a jeho celková výbava je dost chudá. Na kitu jsou pouze dvě tlačítka, jedno slouží jako reset a druhé je uživatelské a volně programovatelné. Dále jsou na kitu dvě LED diody a krystal, který není připájen na desku, ale je umístěn v patici. To umožňuje jeho výměnu, ale také může způsobit problémy s nedostatečným elektrickým kontaktem a tedy celkovou nestabilitu obvodu hodin v mikrokontroleru. S tímto problémem se autor sice nesetkal, ale bývá to častá odpověď na problémy ze stabilitou hodinového signálu. Viz technická podpora společnosti ST[26].

Obrázek 3: kit STM32 vl Discovery[8]

3.2 Kit SAM3S-EK

Dalším porovnávaným vývojovým kitem byl kit od firmy Atmel, konrétně SAM3S-EK[9]. Tento prostředek je mnohem bohatší co se týče jeho výbavy externích periferií oproti popisovanému předchůdci. Z toho vyplývá i jasný cenový rozdíl, který dělá asi desetinásobek ceny kitu od firmy ST. Na desce kitu můžeme nalézt standardní periferie pro vstupně výstupní rozhraní čipu, jako jsou LED diody, tlačítka a dotykový panel. Jsou zde také přítomna rozhraní pro vstup mikrofonu a výstup sluchátek, připojené přes zesilovače implementované na desce přímo k vývodům A/D a D/A převodníků obsažených v čipu. Nalezneme zde i rozhraní pro SD kartu, pro sériové komunikační

linky RS232 a RS485, nebo i externí NAND FLASH paměť sloužící pro rozšíření programové paměti. Je zde možnost využití i ZigBee komunikačního modulu, pro který je na desce vyvedený konektor. Na kitu je i umístěn TFT LCD dislej s rozlišením 320x240 bodů. Je zde také přítomné USB rozhraní, ale to je napojeno na vývody vnitřní periferie čipu, kterou lze nakonfigurovat jako USB host zařízení, podobně jako u čipu STM32F100RB firmy ST. Bohužel, skrz toto ladicí rozhraní nelze ladit programové vybavení čipu.

Tento kit nemá implementován žádný debugger ani jiný ladící nástroj, podobně jako výrobek firmy ST. Na desce je ovšem vyveden konektor pro JTAG rozhraní, které je obsaženo přímo v čipu. Obsahem balení ale není žádné JTAG zařízení, a proto je třeba použít externího prostředku pro ladění, nebo nahrávání programu do programové paměti. Na kitu je ale vyvedeno sériové rozhraní UART/RS232, které je připojeno na další ladící rozhraní čipu. Bohužel toto rozhraní již nebývá běžně výbavou dnešních klasických stolních multimediálních PC, určených pro běžného uživatele (u něj se nepředpokládá možné využití tohoto rozhraní) a proto je nutné použít externího převodníku, například ze sběrnice USB. Čip disponuje ještě jedním programovacím a ladicím rozhraním a to paralelním. Ovšem toto rozhraní není na vývojovému kitu vůbec vyvedeno. To do jisté míry komplikuje práci s tímto vývojovým prostředkem a v konečném důsledku ještě navyšuje cenu, protože kit bez možnosti ladění nebo programováni paměti je nevhodný pro vývoj aplikace a bez dokoupeného potřebného zařízení i nepoužitelný. Z těchto důvodů nebyl vybrán.

I když kit nebyl vybrán pro možný návrh realizace úloh, je vhodné zmínit aspoň minimum o jeho možnostech a možném využití. Na kitu je implementován čip firmy Atmel SAM3S4C [10]. Tento čip nám nabízí kromě standardních periferií, které byly zmíněny výše, sériová rozhraní UART, USART/RS232/RS485, USB, I2C, analogových převodníky a standardní vstupně výstupní rozhraní, také čítač reálných hodin, nebo standardní čítače, které nalezneme snad u všech mikrokontrolerů, bez ohledu na šířku jejich datové sběrnice. Tyto čítače umí pracovat v režimu PWM, vhodném pro řízení motorů, nebo kvadratického enkodéru, který se taktéž využívá k řízení motorů. Dokonce mezi nimi nalezneme i dvoubitový čítač, čítající v *Gray code*, který je ideální pro řízení krokových motorů

Obrázek 4: kit SAM3S-EK firmy Atmel [11]

3.3 Kit EFM32G-DK3550

Posledním vývojovým nástrojem, který byl podroben zkoumáním a porovnáním s dvěma výše popsanými kity byl vývojový kit EFM32G-DK3550 [12] od firmy Energymicro. Tato společnost si zakládá na energeticky výhodných a přijatelných řešení. Tato filozofie se odráží i na výsledných produktech. A tak výrobky této firmy jsou nanejvýše vhodné pro mobilní aplikace a pro všechny ostatní, kde je kladen důraz na spotřebu energie. Je vhodné připomenout, že nízká spotřeba energie byla jedním z kritérií výběru hardwaru.

Nicméně cena tohoto nástroje je asi dvakrát vyšší, než cena kitu firmy Atmel a pohybuje se okolo 7500,- až 10 000,- korun. Ovšem jeho výbava a možnosti této ceně odpovídají.

Celý kit je koncipován jako stavebnice a jednotlivé díly můžou být nahrazeny jinými splňující kompatibilitu s ostatními díly. Kit se skládá celkově ze třech dílů. Základní desky, která obsahuje všechny externí periferie, prototypové desky, na které jsou vyvedeny všechny piny vstupně výstupního rozhraní čipu a univerzální plošný spoj pro realizaci potřebného, rozšiřujícího hardware, Ale pouze jednoduchého. Rozměry plošného spoje nejsou moc velké (240 pájitelných otvorů, 60 pájitelných plošek pro součástky povrchové montáže a dalších 200 pájitelných otvorů a 160 plošek, na které jsou vyvedena napětí +5V a +3.3V). Posledním dílem a zároveň tím nejdůležitějším, je tzv. Plugboard, deska, obsahující samotný čip a v závislosti na verzi i LCD segmentový displej. Výbava kitu je obdobná jako o produktu firmy Atmel. Nalezneme zde audio a analogová rozhraní, připojené k A/D a D/A převodníkům přes příslušné zesilovací a filtrační obvody. Podobně sériová rozhraní jako je USART/RS232, USB ale i Ethernet. Rozhraní pro SD kartu a taktéž TFT LCD displej s rozlišením 320x240 bodů. Dále je zde samozřejmě přítomno několik periferií v podobě tlačítek a LED diod. Bohužel, tyto periferie nejsou přímo připojeny na piny procesoru, ale jsou ovládány skrz sériové rozhraní komunikující s čipem přes rozhraní, SPI.

Na rozdíl od kitu firmy Atmel, má tento kit implementováno rozhraní pro ladění a nahrávání programu do programové paměti, které je fyzicky přístupné skrze USB konektor, což je podstatná výhoda oproti předcházejícímu kitu. Rozhraní se nazývá J-LINK a je realizováno dalším vhodným mikrokontrolerem na desce kitu. Toto rozhraní vyvinula společnost Segger a nabízí taktéž hotová řešení v podobě externích ladících zařízení. Na rozdíl od ST-LINK firmy ST, umožňuje rozhraní J-LINK i tzv. trasování. Tento proces zaznamenává chod a chování programu. A poté, na základě informací obsažených ve výsledku tohoto procesu je možné odstranit chyby programu. Slouží tedy k obdobné činnosti jako klasické ladění programu (*debug*). Klasické ladění (*debug*) pouze prochází programem instrukce po instrukci a vypisuje případná hlášení hned po vykonání aktuální instrukce, nic neukládá a nic na konci procesu ladění nevyhodnocuje tak jako proces trasování. Pomocí tohoto zařízení, je možno ladit a programovat externí zařízení, nebo jej vypnout a k programování čipu na kitu lze použít externího nástroje. Tento úkon se děje pomocí integrovaného software, který je přímo v kitu již od výroby.

Tento firmware se nestará pouze o výše zmíněné úkony, ale také skrze něj lze buď připojit, nebo odpojit externí periferie kitu k procesoru, jako jsou analogová, nebo sériová komunikační rozhraní. Další důležitou funkcí integrovaného firmware je AEM (*Advanced Energy monitoring*), sloužící k monitorování aktuální spotřeby elektrické energie nejen samotného procesoru, ale celého vývojového kitu. Tedy kompletně celé aplikace. Výrobce také poskytuje verzi pro PC, pro mnohem přesnější monitorování a tzv. *energy debug*, což je způsob kontroly spotřeby energie při chodu aplikace. Jak už bylo zmíněno, výrobce si zakládá na spotřebě elektrické energie svých produktů, a proto poskytuje možnosti a nástroje pro co možná nejlepší výsledky v tomto směru.
Na kitu je integrován čip firmy Energymicro, EFM32G890F128 [13], který sám o sobě je poměrně drahou záležitostí mezi dnešními procesory ARM. Jeho cena činní tři sta korun českých. Čip s nejnižší tržní cenou a stejnou architekturou Cortex-M3 je možné zakoupit v maloobchodní síti již od 60 korun českých za kus.
Výbava integrovaná uvnitř čipu je téměř standardní až na pár drobností, které tomuto obvodu dávají jeho cenu. Mezi standardní výbavu patří vstupně výstupní, sériová komunikační a analogová rozhraní. Dále také čítače, které zvládají jak klasický PWM výstup, tak i středovou PWM (*center aligned PWM*) užívanou při řízení motorů nebo i kvadratické dekódování. Nutnou vlastností pro řízení motorů je mód, který umožňuje do řídicích signálů měniče motoru vložit takzvanou mrtvou dobu (*dead time*). A ošetří tak hazardní stavy při spínání měniče motoru. Další zajímavostí je integrované rozhraní pro ovládání LCD segmentových displejů, bez nutnosti použití externích obvodů. Nebo také, PRS (*Peripheral Reflex System*). Tento systém staví na podobné myšlence jako DMA. Tedy spravovat nějaký proces bez nutnosti zásahu procesoru. Systém DMA slouží pro práci s pamětí a PRS slouží k interakci mezi vnitřními periferiemi čipu, bez nutného řízení procesoru. Skrze tento systém mohou chodit spouštěcí impulsy vyvolané periferiemi a určené jiným periferiím k aktivování jejich činností bez nutnosti použití přerušení. S tímto přístupem k řešení aplikací souvisí i režimy spánku procesoru. Pokud totiž není procesor potřeba, je možné jej uvést do stavu spánku a šetřit tak spotřebu energie.
Režimy spánku a vhodná, nezávislá činnost periferií, jsou klíčem k úspoře energie. Při režimu nejhlubšího spánku, je čip neaktivní a reaguje pouze na přerušení reset, spotřebovává pouze 20 nA při napájení 3V. Nicméně v tomto režimu není žádná z vnitřních periferií k dispozici. Při módech spánku, kdy není spotřeba sice tak nízká, ale dosahuje hodnot 45 μA/MHz pro první mód spánku (*Sleep mode*) a 0.9 μA pro mód hlubšího spánku (*Deep sleep mode*), mohou být aktivní periferie, které pracují se systémem PRS a nezatěžují procesor. Při režimu hlubokého spánku (*Deep sleep mode*) nejsou aktivní zdroje ani obvody pro správu hodinového signálu 32MHz. Z toho plyne fakt, že v tomto režimu nelze použít periferie, které tento zdroj hodin vyžadují pro svou činnost

Obrázek 5: Kit EFM32G-DK3550 [14]

3.4 Výběr vhodného vývojového prostředku

Díky chudé výbavě kitu od firmy ST by byl autor nucen realizovat velkou část externích zařízení a obvodů, což dělá práci složitější a náročnější. Vývojový prostředek firmy Atmel nabízel sice hardware výbavu na lepší úrovni, ale postrádal možnost ladění, nebo nahrávání programu do programové paměti. Ovšem vývojový prostředek firmy Energymicro skloubil obě výhody a přidal možnost například možnost výměny desky s mikrokontrolerem za jiný, ale kompatibilní se základní deskou kitu. A to platí i pro připojitelnou desku s pájivým polem. Díky možnosti sledování spotřeby a efektivní činnosti čipu a vnitřních periferií, právě s ohledem na spotřebu energie, se kit EFM32G-DK3550 firmy Energymicro stal jasným kandidátem a byl vybrán jako prostředek pro návrh realizace níže popsaných úloh.

ENERGY®
micro

Obrázek 6: logo firmy Energymicro [15]

4 PROGRAMOVÉ VYBAVENÍ

Ke zvolenému vývojovému kitu, viz předcházející kapitola, je dostupná poměrně široká software základna, v podobě software balíků, knihoven a nástrojů, které výrobce poskytl, nebo doporučil.
Tyto nástroje zefektivňují práci a usnadňují veškerou činnost spjatou s čipem. Mezi takové nástroje patří vývojové prostředí, doporučené výrobcem. Prostředí sice není jeho dílem, ale zaručuje stabilní chod ladicího a programovacího nástroje, který výrobce čipu upřednostňuje pro svá zařízení. Dále jsou to pak hotové knihovny pro vývoj programového vybavení. Napsané v jazyku C, sloužící jako jakési základní stavební kameny pro efektivní a snadnou práci s procesorem. O možnostech těchto nástrojů pojednávají následující podkapitoly.

4.1 Vývojové prostředí IAR Embedded

Prostředí firmy IAR Systems[16], slouží jako vývojový prostředek aplikací nejen pro procesory s architekturou ARM, ale i mnoho dalších. Jako jsou například i osmibitové, nebo jiné dvaatřiceti bitové mikrokontrolery firmy Atmel, nebo ST. Ovšem pro každý typ je potřeba jiný modul vývojového prostředí, kvůli kompilátorům, ladicím nástrojům a simulátorů. Pro potřeby této práce, bylo použito pouze modulu pro práci s ARM procesory. Společnost spolupracuje s většinou velkých výrobců mikrokontroleru a díky tomu poskytuje poměrně robustní podporu, bez ohledu na zvolený obvod či výrobce. Je nutno zdůraznit, že každý modul je poskytován zvlášť a tedy i za každý nový modul je třeba zaplatit.
Vývojové prostředí IAR Embedde, slouží k editace kódu v programovacím jazyce C, k jeho kompilaci a zahrnuje i prostředí pro ladění výsledné aplikace. Prostředí zahrnuje i simulátor, pro odladění aplikace „nanečisto“.

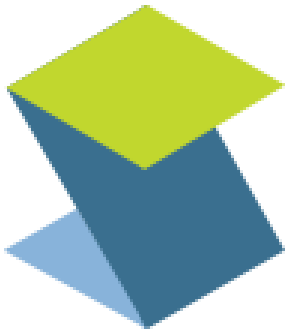
Obrázek 7: logo společnosti IAR Systems [17]

4.2 Soubor programových nástrojů

Software balík v podobě knihoven, obsahuje předpřipravené funkce pro práci se všemi vnitřními periferiemi čipu, jako jsou: čítače, analogové převodníky, vstupně či výstupní rozhraní. Ale také se prostřednictvím nich zajišťuje základní konfigurace čipu, nastavení paměti a systému hodin pro chod jádra, sběrnic a vnitřních periferií.
Díky těmto nástrojům potřeba hluboké precizní znalosti konkrétního čipu, ale programátor je díky knihovnám oddělen od podstaty procesoru, což zefektivňuje a urychluje práci. Jak už bylo zmíněno, výrobce dává k dispozici bohatou dokumentaci. A to i co se týče, právě programové podpory. K dispozici je vždy detailní popis funkce knihovny, spolu s bohatými příklady. A tak i s poměrně elementárními znalostmi je člověk schopný pochopit problematiku a navrhnout požadovanou aplikaci. A právě tento přístup je dnes v průmyslu a v praxi nejen vítaný, ale dokonce i požadovaný.

4.3 Simplicity studio

Simplicity Studio je užitečný nástroj vyvinutý právě firmou *Energymicro* a slouží ke shromáždění veškerých dat a informací poskytnutých výrobcem čipu na jedno místo a programátor nemusí zdlouhavě hledat veškeré zdroje, které jsou někdy těžce dohledatelné na serveru výrobce čipu.
Nástroj umožňuje přístup nejen k dokumentacím, ale také k užitečným nástrojům. Jedním z nich je *energyAware Commander*, který slouží pro komunikaci s kity a jejich správu. Dalším je *energyAware Designer* pro náhled nad konfigurací všech vývodů pouzdra. A v neposlední řadě je to *energyAware Profiler*, který umožňuje monitorovat spotřebu celého kitu. Dokáže rozlišit události v běhu procesoru a rozlišit je o ostatních procesů, jako například začátek, průběh a konec přerušení.



Obrázek 8: symbol Simplicity Studio [18]

5 VÝBĚR VHODNÝCH ÚLOH A NÁVRH JEJICH MOŽNÉ REALIZACE

Dalšími body zadání byli nalézt vhodné úlohy pro demonstraci možností vybraného kitu a čipu na něm implementovaném a analýzy, zda je úlohu vůbec možno realizovat. Pokud ano, tak pokud možná co nejefektivněji s ohledem na: využití periferií, propustnost dat a spotřebu celého zařízení.
Jelikož autor práce je studentem oboru automatizační a měřicí techniky, tak z toho důvodu byla vybrána úloha *Datalogger vibrací pro diagnostiku rotačních strojů.*

5.1 Datalogger vibrací pro diagnostiku točivých strojů

Tato úloha byla zadána pracovníky ústavu automatizace a měřicí techniky VUT v Brně a její kompletní zadání vypadá následovně.
Datalogger musí splňovat:

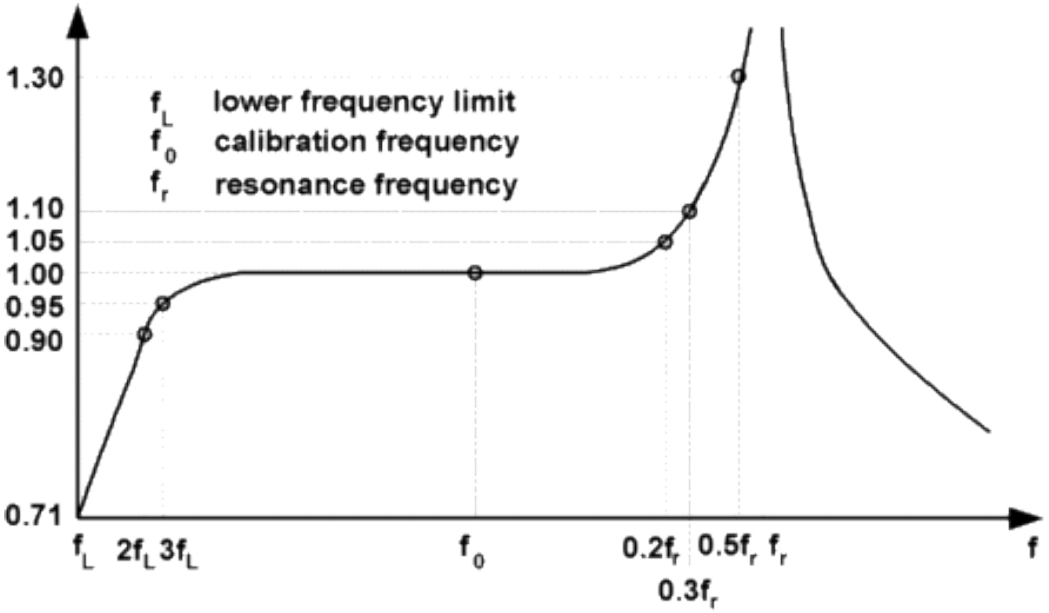
1) Zpracování analogových vibračních signálů ze dvou snímačů zrychlení (piezoelektrické snímače s výstupem ICP) s citlivostí 10mV/m/s². vzorkování signálu na frekvenci 16384 Hz. Dynamický rozsah měřených hodnot minimálně 60dB.
2) Návrh digitálních filtrů pro úpravu měřených signálů. Volitelný digitální filtr typu horní propust s možností volby mezního kmitočtu 1 Hz, 3 Hz nebo 10 Hz se strmostí minimálně 40dB/dek. Volitelný digitální filtr typu dolní propust s možností volby mezního kmitočtu 100 Hz nebo 1000 Hz a se strmostí minimálně 40dB/dek.
3) Výpočet rychlostí vibrací pomocí integrace signálu po dobu jedné sekundy.
4) Ukládání průměrné hodnoty rychlosti a zrychlení vibrací v časovém úseku 15 minut (průměr z 900 okamžitých hodnot, týdenní záznam vyžaduje paměť na 672 uložených průměrných hodnot) nebo 8 hodin (průměr z 28800 okamžitých hodnot, roční záznam vyžaduje paměť na 1095 uložených průměrných hodnot)
5) Možnost navíc - výpočet FFT v reálném čase s kmitočtovým rozlišením 1 Hz, volba oken obdélníkové/Hamming, zobrazení 5 nejvýznamnějších spektrálních čar na displeji

V následujícím textu bude nastíněna možná realizace jednotlivých bodů zadání s ohledem na vlastnosti a možnosti použitého mikrokontroleru.

5.1.1 Zpracování analogových signálu ze snímačů

Nejprve je potřeba zhodnotit obecné vlastnosti zadaných snímačů a to zda jsou vůbec

vhodné pro měření vibrací.
Piezoelektrické snímače zrychlení fungují na principu piezoelektrického jevu. Vlivem mechanické deformace na protilehlých stranách, se na sousedních protilehlých stranách vytváří napětí a generuje se náboj. Praktické uspořádání těchto snímačů musí obsahovat vhodný piezoelektrický materiál a seismickou hmotu, která je vhodně připevněna na piezoelektrický materiál a která pak spolu s působícím zrychlením vyvolává sílu a ta zapříčiní potřebnou mechanickou deformaci. Vygenerovaný náboj je pak úměrný zrychlení působící na seismickou hmotu [19].
Piezoelektrické snímače se vyznačují omezenou pracovní oblastí. Z jedné strany je to omezení způsobené neschopností měřit hodnoty při nízkých kmitočtech kvůli svodovému parazitnímu odporu, který odvádí vygenerovaný náboj na nulový potenciál a měření pak není proveditelné. Z druhé strany je to pak omezení mechanickou rezonancí seismické hmoty.
Zadané snímače jsou vybaveny ICP výstupem, *Integrated Circuit Piezoelectric*. Jedná se o integrovaný nábojový zesilovač, který stanovuje konečnou citlivost snímače v jeho pracovní oblasti. Citlivost zadaných snímačů je 10mV/m/s².



Obrázek 9: ukázka frekvenční charakteristiky piezoelektrického snímače [20]

Dále byly v zadání úlohy Datalogger jasně stanoveny parametry pro snímání signálů. Vzorkovací frekvence 16384 Hz a dynamický rozsah naměřených hodnot minimálně 60 dB. Aby bylo jasné, zda je integrovaný A/D převodník vhodný měření hodnot s těmito parametry, je třeba si nejprve zjistit jeho vlastnosti.
A/D převodník integrovaný v použitém mikrokontroleru implementovaného na kitu (ne v případné vnitřní struktuře snímače) je převodník s postupnou aproximací a je pouze jeden, ale disponuje osmi přepínatelnými kanály. Převodník má maximální rozlišení 12bitů. Toto rozlišení je konfigurovatelné na 6, 8 a 12 bitů. Pomocí převzorkování, kdy

AD převodník provede více měření za sebou a výsledné hodnoty pak sečte, lze dosáhnout rozlišení až 16 bitů, ovšem za cenu prodloužení doby převodu. Převodník má také několik vnitřních zdrojů referenčního napětí, 1,25V a 2,5V. Samozřejmostí je také možnost použití externího referenčního napětí. Pro usnadnění bylo zvoleno vnitřní reference 1,25V (+/-625mV v diferenciálním režimu A/D převodníku). Z toho vyplývá jasné omezení a to maximální možná naměřená hodnota zrychlení. Při zadané citlivosti a použité reference je tato hodna 62,5 m/s², protože je třeba počítat i se zápornými hodnotami zrychlení a charakter vibrací v čase se předpokládá sinusový. Tato hodnota nesmí být překročena! Jinak by mohlo dojít k poškození analogového převodníku a tím celého zařízení. Je možné vstup A/D převodníku ošetřit vhodným napěťovým omezením aby nedošlo ke zničení zařízení. Pro zajištění dynamického rozsahu 60 dB při nejvyšší možné změřené hodnotě odpovídající napětí 2,5V musí nejnižší změřitelná hodnota odpovídat napětí 0,0025V. Aby převodník dosáhl takového rozlišení, musí mít rozlišení alespoň 10 bitů, což převodník, který je k dispozici neumožňuje a proto je nutné zvolit vyšší rozlišení a to 12 bitů. Při takovém rozlišení a použitém referenčním napětí, dokáže změřit nejnižší hodnotu napětí rovnu 0,0006V, což je dostačující.
Zajištění zadané vzorkovací frekvence lze docílit dvěma způsoby. Prvním je nastavit AD převodník tak, aby měřil hodnoty signálů opravdu s frekvencí 16384 Hz. Z dokumentace obvodu[13] lze stanovit vztah pro konečnou dobu převodu T_{conv} v podobě:

$$T_{conv} = ((T_A + N) * OSR * N_{ch})/f_{ADC} \qquad 5.1.1.1$$

Kde T_A je počet měřicích cyklů, který slouží k nastavení konkrétní doby trvání převodu. Lze nastavit 1 až 256 těchto cyklů. Hodnota N udává rozlišení převodníku v bitech. Parametr OSR udává hodnotu, kolikrát byl převáděný signál vzorkován, respektive převzorkován pro získání jednoho výsledku převodu (více v [13]). N_{ch} je počet použitých kanálu převodníku a f_{ADC} je kmitočet samotného analogového převodníku. Tato frekvence je odvozena z hlavního hodinového obvodu procesoru a může být dále upravována pomocí vnitřní 256 bitové předděličky. Důležité však je, aby se její hodnota vždy pohybovala mezi 32 kHz a 13 MHz. Pokud se převodník nastaví tak, aby poskytoval naměřené hodnoty s frekvencí 16384 Hz, bude muset být pořád v chodu a nebudou žádné časové prodlevy, kdy by se mohl procesor například uvést do stavu nižší potřeby, který by pro aplikaci typu logger byl neocenitelný a to kvůli dlouhé době, kdy je zařízení v chodu.
Druhá možnost je nastavit dobu převodu co nejnižší a například vnitřním čítačem časovat a spouštět okamžiky měření. Tento přístup nám naopak umožní využít režimů, kdy procesor spotřebovává minimum energie, v režimů spánku. Čítače totiž nemají takové nároky na napájení jako samotný procesor a mohou běžet i při takovýchto úsporných režimech. Přerušení, které aktivuje převod A/D převodníkem ovšem přivede procesor zpět do režimu plné spotřeby. Tento jev ovšem nenastává při interakci periferií skrz výše popsaný PRS (*peripheral reflex system*), který může zajistit aktivace převodu na pokyn

čítače, aniž by byl procesor přiveden do režimu plné spotřeby. I když pro spracování naměřených dat musí být procesor v režimu plné spotřeby, pomocí PRS je možné prodloužit dobu kdy je procesor v režimu nižší spotřeby a to právě o dobu převodu A/D převodníku.

5.1.2 Návrh digitálních filtrů

Číslicové filtry jsou jednou z možností realizace diskrétních obvodů, které slouží k podobným účelům jako filtry spojité, analogové. Podobně jako jejich spojité ekvivalenty i oni mají za úkol vhodným způsobem ovlivnit tvar kmitočtového spektra signálu. Potlačují nežádoucí složky jako filtry typu horní, dolní, nebo pásmová propust [21].
Zadání úlohy Datalogger jasně udává vlastnosti filtrů. Jedná se o typy horní a dolní propust.
Oba jsou volitelné v tom smyslu, že záleží na koncovém uživateli, zda je použije, či nikoliv. Dále byly zadány mezní kmitočty jednotlivých filtrů a samozřejmě možnost jejich volby. V poslední řadě to byla strmost frekvenčních charakteristik, která byla zadána jako 40 dB/dek.
Při návrhu číslicového filtru se postupovalo následovně:

1) Návrh parametrů a vlastností ve spojité oblasti
2) Diskretizace pomocí bilineární transformace
3) Převedení do tvaru diferenčních rovnic, udávající přímo algoritmus filtru

Jako první je nastíněn návrh dolní propusti s mezní frekvencí 100 Hz. Tomuto filtru odpovídá operátorový přenos ve spojité oblasti:

$$Fs_{(s)} = \frac{1}{(Ts+1)^2} \qquad 5.1.2.1$$

Kde T je časová konstanta systémů a odpovídá hodnotě:

$$T = \frac{1}{2\pi f_c} \qquad 5.1.2.2$$

Kde f_c je mezní kmitočet. Pomocí bilineární transformace viz [21] byl tento systém transformován do diskrétní oblasti podle následujícího postupu:

$$s \sim \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}} \qquad 5.1.2.3$$

$$Fc_{(z)} = \frac{1}{(T\frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}+1)^2} = \frac{\frac{1}{1+q}+\frac{2}{1+q}z^{-1}+\frac{1}{1+q}z^{-2}}{\frac{q^2+2q+1}{1+q}+\frac{2-2q^2}{1+q}+\frac{q^2-2q+1}{1+q}} \qquad 5.1.2.4$$

Kde q odpovídá vztahu:

$$q = \frac{f_{vz}}{\pi f_c} \qquad 5.1.2.5$$

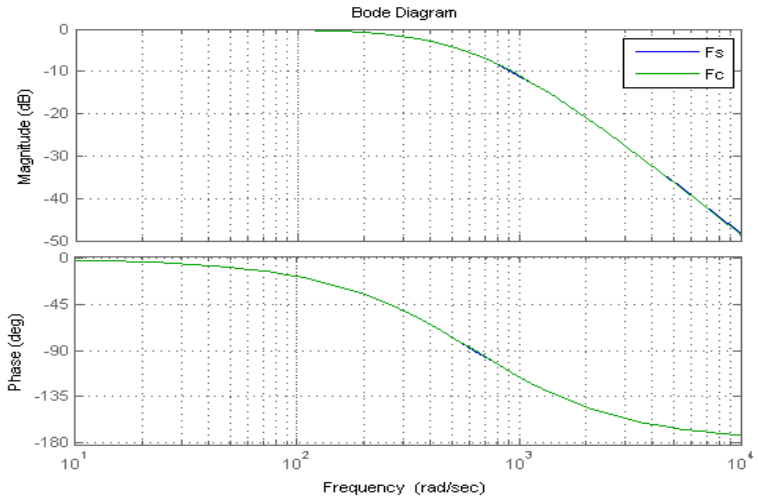
Kde parametrem f_{vz} se rozumí vzorkovací frekvence a $f_c$je mezní kmitočet filtru. Konečnou podobu přenosu filtru lze zapsat takto:

$$Fc_{(z)} = \frac{c_0 + c_1 z^{-1} + c_2 z^{-2}}{d_0 + d_1 z^{-1} + d_2 z^{-2}} \qquad 5.1.2.6$$

Význam jednotlivých koeficientů je patrný z předcházející podoby přenosu filtru. Volitelný parametr filtru, mezní kmitočet, lze změnit pouze jednoduchou úpravou parametru q. Pro zvolený kmitočet 100 Hz a vzorkovací frekvenci 16384 Hz bude přenos filtru typu dolní propust vypadat následovně:

$$Fc_{(z)} = \frac{0{,}0188 + 0{,}0376z^{-1} + 0{,}0188z^{-2}}{53{,}1519 - 102{,}3038z^{-1} + 49{,}2271z^{-2}} \qquad 5.1.2.7$$

V závěru je možno srovnat frekvenční charakteristiku v logaritmických souřadnicích spojité podoby filtru s diskrétní podobou. Z průběhů lze vidět, že diskrétní náhrada za spojitý filtr má podobné vlastnosti jako spojitá předloha.



Obrázek 10: srovnání FACH spojitého a diskrétního filtru

Posledním krokem je jen vytvoření vhodného algoritmu, který by realizoval funkci filtru. K tomuto účelu můžeme využít vlastnosti Z transformace:

$$Fc_{(z)} = \frac{0{,}0188 + 0{,}0376z^{-1} + 0{,}0188z^{-2}}{53{,}1519 - 102{,}3038z^{-1} + 49{,}2271z^{-2}} = \frac{Y_{(z)}}{U_{(z)}} \qquad 5.1.2.8$$

$$53{,}1519Y_{(z)} - 102{,}3038Y_{(z)}z^{-1} + 49{,}2271Y_{(z)}z^{-2} = 0{,}0188U_{(z)} + 0{,}0376U_{(z)}z^{-1} + 0{,}0188U_{(z)}z^{-2} \qquad 5.1.2.9$$

$$53{,}1519y_{(k)} - 102{,}3038y_{(k-1)} + 49{,}2271y_{(k-2)} = 0{,}0188u_{(k)} + 0{,}0376u_{(k-1)} + 0{,}0188u_{(k-2)} \qquad 5.1.2.10$$

$$53{,}1519y_{(k)} = 102{,}3038y_{(k-1)} - 49{,}2271y_{(k-2)} + 0{,}0188u_{(k)} + 0{,}0376u_{(k-1)} + 0{,}0188u_{(k-2)} \qquad 5.1.2.11$$

Kde poslední tvar rovnice je přímo předpis pro algoritmus chování filtru.

Obdobným způsobem postupujeme i při návrhu filtru typu horní propust. Její přenos v operátorovém tvaru ve spojité oblasti vypadá následovně:

$$Fs_{(s)} = \frac{(Ts)^2}{(Ts+1)^2} \qquad 5.1.2.12$$

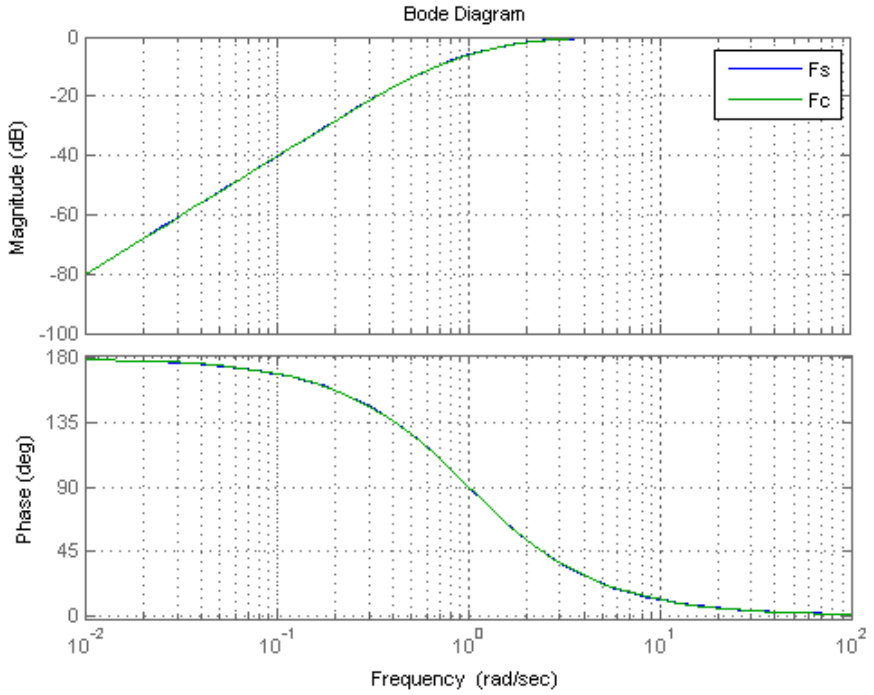
Její diskrétní ekvivalent podle stejného postupu pak vypadá:

$$Fc_{(z)} = \frac{q^2 - 2q^2z^{-1} + q^2z^{-2}}{(q^2 + 2q + 1) + (2 - 2q^2)z^{-1} + (q^2 - 2q + 1)z^{-2}} = \frac{c_0 + c_1z^{-1} + c_2z^{-2}}{d_0 + d_1z^{-1} + d_2z^{-2}} \qquad 5.1.2.13$$

Pro parametr q platí stejný vztah jako u předchozího příkladu. Pro účely demonstrace návrhu byla zvolena mezní frekvence 1 Hz a při vzorkovací frekvenci 16384 Hz výsledný přenos vypadá následovně:

$$Fc_{(z)} = \frac{1{,}0737 * 10^9 - 2{,}1475 * 10^9z^{-1} + 1{,}0737 * 10^9z^{-2}}{1{,}0738 * 10^9 - 2{,}1475 * 10^9z^{-1} + 1{,}0737 * 10^9z^{-2}} \qquad 5.1.2.14$$

Stejně jako v předchozím příkladu, můžeme srovnat frekvenční charakteristiky jednotlivých přenosů.



Obrázek 11:srovnání FACH spojitého a diskrétního filtru

I druhého filtru můžeme pozorovat přijatelnou náhradu vzhledem k spojité předloze. Nyní jako v předchozím příkladu zbývá navrhnout vhodný algoritmus. Stejným postupem dospějeme k výsledku:

$$1{,}0738 * 10^9 y_{(k)} = -2{,}1475 * 10^9 y_{(k-1)} - 1{,}0737 * 10^9 y_{(k-2)} + 1{,}0737 * 10^9 u_{(k)} - 2{,}1475 * 10^9 u_{(k-1)} + 1{,}0737 * 10^9 u_{(k-2)} \quad 5.1.2.15$$

Je ovšem důležité upozornit na skutečnost, že frekvenční charakteristiky takto navržených filtrů se periodicky opakují s periodou rovnou vzorkovací frekvenci, v tomhle případě 16384 Hz a proto je nezbytné použít před diskretizací antialiasing filtru, který potlačí všechny složky spektra s frekvencí vyšší než frekvence vzorkování. Také je nutné poukázat na fakt, že v praxi může být reálná vzorkovací frekvence odlišná od předpokládané, která je použita v návrhu. To má za následek posunutí mezního kmitočtu filtru. Mezi posunem a rozdílem vzorkovací frekvence platí přímá úměra. A také rozdíl mezi mezními kmitočty bude vetší, čím bude větší zvolená mezní frekvence. Závislost je jasná ze vztahu pro parametr q.

Praktická realizace těchto filtrů není velkou výkonovou zátěží pro použitý čip, a proto je jejich realizace splnitelná.

5.1.3 Výpočet rychlosti vibrací pomocí integrace

Pro výpočet okamžité hodnoty rychlosti je zapotřebí integrovat hodnoty zrychlení.
Jako metodu integrace byla zvolena eulerova integrační metoda.
Tato metoda se hojně používá v mikroprocesorových aplikací, díky své výpočetní nenáročnosti a jednoduché implementovatelnosti. Mechanismus metody je znázorněn v následujícím vztahu[22]:

$$v_{(t_0+h)} = v_{(t_0)+}h * a_{(t_0)} \qquad 5.1.3.1$$

Kde h je krok integrace a je roven:

$$h = 1/f_{vz} \qquad 5.1.3.2$$

Kde parametr f_{vz} je vzorkovací frekvence signálu. Pro potřeby úlohy by se velikost kroku rovnala:

$$h = \frac{1}{16384} = 6{,}1035 * 10^{-5} \qquad 5.1.3.3$$

Ani tuto metodu není nikterak velký problém integrovat do zvoleného mikrokontroleru. Je ale potřeba počítat s chybou R jednoho kroku této metody, která je přibližně rovna:

$$R \sim n * h^2 \sim h \qquad 5.1.3.4$$

Kde parametr n je kroků a h je krok integrace.

5.1.4 Ukládání průměrných hodnot rychlosti a zrychlení

Hodnoty zrychlení a rychlosti vibrací je vhodné vyjádřit jako efektivní hodnoty z intervalů, které jsou jasně definovány zadáním úlohy. Jejich délka musí být jedna sekunda, protože takto udává zadání (průměrné hodnoty, ze efektivních, které jsou získávány z jedno sekundových intervalů).

$$ef = \sqrt{\frac{1}{T}\int_0^T f^2{}_{(t)}\,dt} \qquad 5.1.4.1$$

Pro integraci ve vztahu, byla použita obdélníková metoda[23]:

$$\int_a^b f_{(t)}dt = h\sum_{i=1}^{n} f_{(x_i-1)}\,;\; h = \frac{b-a}{f_{vz}} \qquad 5.1.4.2$$

Zadáno bylo získávání průměrných hodnot ze dvou možných časových úseků. První je úsek 15 minut a tedy průměr z 900 efektivních hodnot. Na tomto intervalu bylo zadáno získávat záznamy o délce jednoho týdne. Tomu odpovídá dvakrát 672 průměrných hodnot rychlosti a zrychlení.
Aby bylo možné takovýto záznam uchovávat, je nutné mít k dispozici dostatečně velikou paměť. Za předpokladu 32 bitové velikosti jedné hodnoty je potřeba paměť o velikost 5376 B, pro jeden snímač. Zvolený mikroprocesor disponuje až 16 kB rychlou SRAM pamětí [13], požadavek je tedy splněn.
Druhý úsek činí osm hodin. Průměrné hodnoty se tedy počítají každá z 28800 efektivních hodnot a je požadováno vést roční záznam. To znamená mít k dispozici paměť pro dvakrát 1095 hodnot. Při předpokládané 32 bitové velikosti jedné hodnoty je potřeba paměť o velikosti 8760 B, opět pro jeden snímač. Bohužel tato podmínka už není splněna. Pro potřeby ukládání takového množství dat, je vhodné využít sériového rozhraní *SPI*, prostřednictvím kterého je realizována komunikace s SD kartou. Ta přináší možnost ukládat data do textového souboru a následně je formátovat do čitelnější a přijatelnější podoby.
Výrobce čipu poskytuje knihovnu, která implementuje souborové systémy FAT12, 16 a 32 a umožňuje jednoduchou práci s nimi [24] a také knihovnu realizující fyzickou vrstvu pro komunikaci s SD kartou.

5.1.5 Numerické zpracování dat

Předchozí čtyři kapitoly poskytovaly náhled na možné řešení celé problematiky sběru a zpracování dat. Popisovaly také detailně jednotlivé metodiky, jako například číslicové filtry. Tato kapitola bude částečnou rekapitulací předchozích čtyřech a poskytne náhled na konečnou a reálnou podobu řešení.
Na začátku procesu časovač skrze výše popsané *PRS* rozhraní iniciuje převod dvou kanálů A/D. Po skončení převodu systém *DMA* uloží hodnoty do paměťového zásobníku. Tyto operace jsou nezávislé na procesoru a nevyžadují od něj žádnou spolupráci. V tento okamžik je také velmi výhodné jej nechat setrvat v energeticky úsporném režimu. Takto popsaným mechanismem sběru dat je změřeno celkem 128 hodnot od každého kanálu. Zásobník takto naměřených hodnot je poté vyměněn s druhým zásobníkem a měření se opakuje. Zásobník naměřených hodnot je mezi tím zpracováván. Nejprve dojde k přepočtu naměřených hodnot z A/D na skutečné hodnoty napětí a poté, pokud je to uživatelem žádoucí, se provede filtrace tohoto bloku dat skrze dolní propust druhého řádu popřípadě horní propust druhého řádu. Po této filtraci, jsou data integrována podle metody popsané v kapitole 5.1.3. Teoretický návrh opomenul jednu věc a to, že výsledný

integrovaný signál bude zatížen chybou v podobě stejnosměrné složky úměrné frekvenci tohoto signálu. Pro odstranění této chyby je po integraci signálu aplikována filtrace signálu dolní propustí prvního řádu. Dále jsou jednotlivé vzorky původního signálu a integrovaného signálu podrobeny částečnému výpočtu efektivní hodnoty. Částečnému proto, protože se jedná pouze o sumaci kvadrátů jednotlivých hodnot. Po tomto je procesor opět uveden do energeticky výhodného režimu. Až dojde k naměření dalších 128 hodnot od každého kanálu, jsou zásobníky opět prohozeny a mechanismus zpracování dat se opakuje. Až je takto zpracováno 128 bloků, znamená to, že byl zpracován úsek signálu trvající 1 sekundu. Ze zadanou vzorkovací frekvencí 16384Hz je to právě 128*128 hodnot co dává jednosekundový úsek signálu. Zadáno je získávat průměrné hodnoty z efektivních hodnot, které jsou získány právě z jednosekundových intervalů. Tedy nyní, když je zpracováno 128*128 hodnot se provede dokončení výpočtu efektivní hodnoty, tedy vynásobení s obrácenou hodnotou periody vzorkování, tedy právě s hodnotou 16384 a po té je tento součin odmocněn, viz Kapitola 5.1.4. Takto získaná efektivní hodnota je přičtena k průběžnému součtu efektivních hodnot. Podle režimu ukládání hodnot, týdenní nebo roční (viz Kapitola 5.1.4) se sumuje buďto 900 nebo 28800 efektivních hodnot a poté co je dosaženo sumace potřebného množství hodnot je jedním z těchto dvou čísel suma podělena a takto získána průměrná hodnota efektivních hodnot je zapsána a uložena do textového souboru na SD paměťové médium. Tato posloupnost operací platí pro oba kanály i pro oba typy hodnot, tedy jak pro hodnoty zrychlení, tak i pro hodnoty rychlosti získané integrací změřených hodnot. Po získání potřebného množství středních hodnot, pro týdenní záznam je to 672 středních hodnot a pro roční záznam je to 1095 hodnot, se programově zastaví čítač, tím pádem i časování převodů A/D a na uživatelském rozhraní se signalizuje konec měření.
Z hlediska programového zpracování byla vytvořena struktura obsahující údaje o vzorkovací frekvenci, proměnné pro mezi výpočty efektivních a středních hodnot, dále počítadla již zpracovaných dat a také ukazatele na zásobníky dat. To vše kvůli jednoduší práci a lepší přehlednosti software viz příloha.

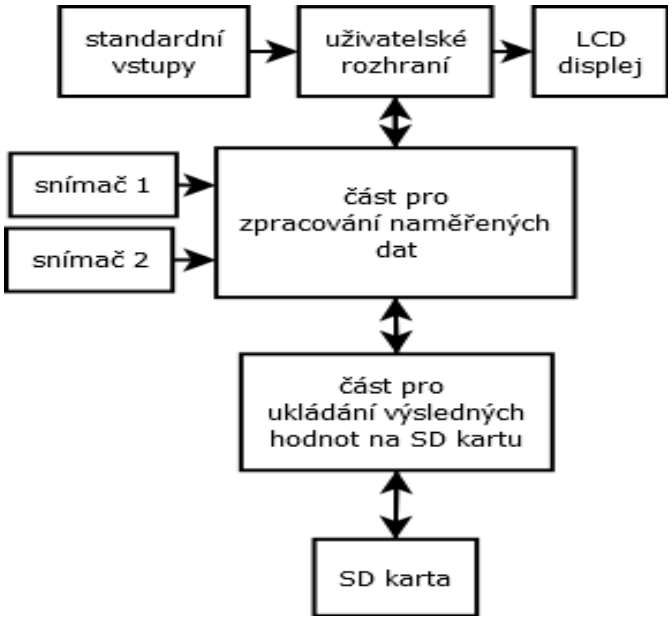
5.1.6 Návrh uživatelského rozhraní

Na vývojovém kitu je integrován TFT LCD RGB panel s rozlišením 240x180 bodů a dotykovým povrchem v rezistivním provedení. Ten se dokonale hodí pro uživatelské rozhraní. Bohužel vodiče, skrz které je tento displej připojen k procesoru na vývojovém kitu jsou ty samé vodiče, které slouží jako diferenciální vstupy pro A/D převodník a proto nelze tyhle dvě periferie provozovat v jeden okamžik. Z tohoto důvodu je uživatelské rozhraní situováno na segmentový, znakový a nedotykový LCD displej. Tímto také odpadá možnost realizace FFT algoritmu, protože by nebylo možné jeho výsledek zobrazit. Procesor disponuje integrovaným řadičem pro až 40×40 LCD segmentový displej. Jako uživatelské vstupy jsou použity standardní vstupy a výstupy procesoru. Rozhraní by mělo umožnovat uživateli konfigurovat zařízení v souladu se zadáním. Uživatel by měl být schopný nastavit vlastnosti číslicových filtrů, a zda je vůbec do

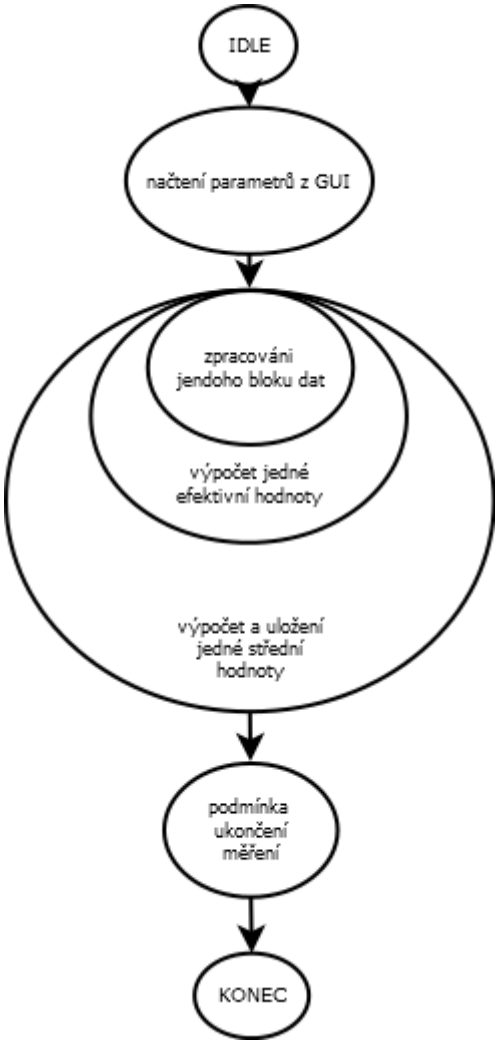
systému zapojit, dále vybrat jeden ze dvou režimů ukládání dat. Dále by byla vhodná možnost kdykoliv proces přerušit a samozřejmě i aktivovat.

5.1.7 Návrh algoritmu aplikace

Nakonec je třeba navrhnout vhodný algoritmus toho, jak se bude celá aplikace chovat, neboli v jakém sledu budou na sebe navazovat jednotlivé části programu.
Popis předpokládaného algoritmu je jednoduchý. Program čeká ve výchozím stavu, dokud nejsou nastaveny uživatelem všechny potřebné parametry měření. Poté, po spuštění měření program přejde do stavu, kdy načte uživatelskou konfiguraci, pak přejde do čekací smyčky, kde procesor může čekat v energeticky úsporném režimu. V tom samém okamžiku již probíhá měření, které nijak nepotřebuje spolupráci s procesorem. Po dokončení měření jsou prohozeny zásobníky dat a tak jak je popsáno v přecházející kapitole 5.1.5, jsou data zpracována. Obrázek č. 13 zobrazuje pouze zjednodušenou podobu algoritmu popisované aplikace. Podrobnější verzi je možné nalézt v elektronické příloze této publikace. Jak už bylo několikrát popsáno, hlavní algoritmus se skládá ze třech dílčích částí, které jsou patrné ze zjednodušené podoby vývojového diagramu algoritmu. Tyto tři části jsou: zpracování jednoho bloku naměřených dat, po zpracování potřebného množství dat následuje výpočet efektivní hodnoty a poté, po zpracování potřebného množství efektivních hodnot je získána jedna střední hodnota. Ta je pak uložena do souboru a celý cyklus se opakuje, dokud není splněna podmínka o ukončení měření. Tomuto samozřejmě předchází inicializace aplikace skrze načtení uživatelských nastavení z uživatelského rozhraní a po změření poslední střední hodnoty je ukončeno a tento konec je signalizován. Na obrázku č. 12 je vidět blokové schéma celé aplikace.



Obrázek 12: blokové schéma aplikace



Obrázek 13: zjednodušený algoritmus aplikace

5.1.8 Zhodnocení možností realizace dané úlohy

Úloha typu logger je ideální pro zvolený čip. Díky možnostem využití nízké energetické spotřeby, možnosti využití rozhraní pro SD kartu, nebo podporu pro souborové systémy *FAT* pro uchování naměřených dat a celkové využití širokého spektra periferií. V rámci této kapitoly nebyly řešeny problémy ohledně spektrální analýzy, ani možnosti uživatelského rozhraní pro konfiguraci zařízení. Těmto problémům bude eventuálně věnována praktická část.

5.1.9 Praktická realizace algoritmu

Celý algoritmus aplikace, tak jak je ho možné nalézt v příloze, byl realizován v jazyce C, vyhovující normě C99. Tento programovací jazyk je velmi rozšířený a oblíbený. Používá se pro programování procesů jak osobních počítačů (platforma Intel x86, x64), tak i pro ostatní mikroprocesorové systémy, mikrokontrolery postavených třeba právě na jádru

ARM. Tato kapitola stručně popisuje jednotlivé aspekty kódu a jejich vzájemné spolupráce. Celý kód v podobě projektu pro *IAR Embedded Workbanch IDE* je možno dohledat v elektronické příloze publikace. Také bude v této kapitole často odkazováno právě na výše zmíněnou přílohu prostřednictvím funkcí a jednotlivých souborů celého projektu.

Celý program aplikace se dá rozdělit na tři logické části. První části je vhodná konfigurace a inicializace všech periferií nutných pro měření dat a také jejich ukládání na SD paměťovou kartu. Pro upřesnění bude uveden výčet všech periferií, které jsou za výše popsané procesy odpovědné. Jedná se o: A/D převodník pro sběr a digitalizaci dat ze snímačů, čítač a časovač pro časování jednotlivých převodů, řadič DMA (*Direct Memory Access*), který zodpovídá za bezchybný přenos dat z A/D převodníku do paměti, dále pak o systém PRS (*Peripheral Reflex System*), který slouží jako prostředník pro interakci mezi čítačem a A/D převodníkem. A v poslední řade se jedná o universální synchronní přijímač a vysílač USART (*Universal Synchronous Receiver and Transmitter*), který slouží jako univerzální sériový komunikační prostředek a je konfigurován, jako sériové periferní rozhraní SPI (*Serial Peripheral Interface*) pro komunikaci s SD kartou. Ze všeho nejdříve je ale ve funkci, *main* v stejnojmenném souboru *main.c*, volána funkce *CHIP_Init*, která se stará základní inicializaci samotného procesoru. Kontroluje revizi čipu a nastavuje kalibrační hodnoty pro A/D a D/A převodníky. Po ní následují funkce *CMU_ClockEnable* a *CMU_ClockSelectSet*, které zaručují nastavení taktu samotného procesoru na maximální možnou hodnotu a to 32MHz. Po těchto funkcích následuje sled inicializačních funkcí periferií. První je funkce pro inicializaci A/D převodníku *adc_Init*, která se stará o nastavení rozlišení převodníku, výběr referenčního napětí pro převod, nastavení typu převodu, výběr vstupních kanálů pro měření, určuje způsob spouštění jednotlivých převodů, také zajišťuje připojení hodinového signálu do řídící logiky A/D převodníků a samozřejmě nastavuje dobu jednoho převodu skrz nastavení předděličky pro řídící logiku a také skrze výběru počtu měřících cyklů, což je počet cyklů hodin řídící logiky, po kterých je převeden vzorek napětí uchovaný ve vzorkovacím kondenzátoru A/D převodníku. Tento počet je vhodné volit s ohledem na výstupní impedanci zdroje signálu. Dále je volána funkce *sdc_Init*, která se stará o inicializaci komunikačního rozhraní pro SD paměťovou kartu a skrze testování této komunikace testuje její správnost. Z tohoto důvodu je nutné, aby byla SD paměťová karta vložena již před spuštěním samotné aplikace, nejen před samotným měřením. Po této funkci následuje funkce *tim_Init*, která se stará o správnou inicializaci čítače a časovače, který jak už bylo zmíněno, slouží k vhodnému časování jednotlivých převodů A/D převodníku. Funkce podobně jako u A/D převodníku připojuje k vnitřní logice čítače hodinový signál, pomocí vnitřní předděličky nastavuje rychlost čítání a v poslední řadě nastavuje limitní hodnotu pro čítání a tím pádem i vlastní periodu čítače samotného. Další funkcí je *prs_Init*. Tato funkce inicializuje systém PRS. Funkce pouze zajišťuje připojení hodin do vnitřní logiky systému a připojení výstupu z čítače na jeden z osmi kanálů systému PRS, na kterém je při každém přetečení čítače, přetečení jeho maximální hodnoty, indikován spouštěcí impuls pro převod A/D převodníku. Druhá strana tohoto kanálu systému PRS je připojena

vstup A/D převodníku, který slouží k spouštění převodu. Toto spojení ale zajišťuje funkce *adc_Init*. Další volanou funkcí v souboru *main.c* je funkce *dma_Init*. Tato funkce zajišťuje konfiguraci řadiče DMA, který slouží pro přenos dat z A/D převodníků do paměti, konkrétně do datových zásobníků *buff_1* popřípadě *buff_2*. Funkce zajišťuje nastavení velikosti jednoho přenášeného vzorku, s tím spojenou i velikost inkrementace v cílové paměti (zásobníku) a tak jako všechny předcházející funkce i ona zajišťuje přivedení hodinového signálu do vnitřní řídící logiky DMA řadiče. Hodinové signály bývají od periferií standardně implicitně odpojené z důvodu úspory energie. Nakonec funkce zajišťuje konfiguraci tzv. *call-back* funkce pojmenované *transferComplete*, která je volána vždy po dokončení celého přenosu prostřednictví DMA řadiče. Definici této funkce je možné dohledat v souboru *main.c*. Její úloha spočívá v přehození zásobníku dat tak, aby zásobník s naměřenými daty byl připraven pro zpracovávání a zásobník už se zpracovanými a už neplatným daty připraven pro nové měření. Mechanismus přehození je jednoduchý, jedná se pouze o přehození ukazatelů příslušných datových zásobníků. Fyzické překopírování obsahů jednotlivých zásobníků by bylo zbytečně časově náročné. Ušetřený čas pak procesor setrvává v energeticky úsporných režimech. Dále tato *call-back* funkce zajišťuje volání funkce *mean_val_getNlog*, která odpovídá za zpracování jednoho naměřeného bloku dat a popřípadě výpočet efektivní a střední hodnoty, nebo i zápisu do souboru. O této funkci více v odstavci níže. Poslední věcí, kterou funkce *dma_Init* zajišťuje, je nastavení prvního přenosu. Toto nastavení je vždy nutno udělat před každým přenosem dat. Nastavení zahrnuje definici zdroje dat, destinaci přenosu a v poslední řadě počet vzorků dat, které je potřeba přenést. Toto nastavení nespouští samotný přenos, pouze konfiguruje jeho parametry. Za samotné spuštění přenosu je zodpovědný A/D převodník. Další inicializační funkce je pro inicializaci uživatelského rozhraní. Jedná se o *interface_Init*, která inicializuje vstupy a výstupy pro rozhraní. Taktéž se stará o inicializaci displeje skrze funkci *SegmentLCD_Init*. Všechny deklarace popřípadě definice všech výše popsaných funkcí, lze dohledat v souboru *FW_Init.h* popřípadě *FW_Init.c* umístěných v elektronické příloze. Další inicializační funkcí, která ovšem nezajišťuje inicializaci žádné periferie je *file_Init*. Tato funkce zajišťuje inicializaci textového souboru pro záznam zpracovaných dat. Inicializace spočívá v zapsání průvodního textu do souboru, který má sloužit pro lepší orientaci v souboru pro zaznamenání zpracovaných dat.

Druhou části programu aplikace je kód zajišťující zpracování dat ve smyslu filtrace, integrace, výpočtu efektivních hodnot, výpočtů středních hodnot a zápisu na SD kartu. Pro snadnější práci s daty a jejich zpracováním byl deklarován jednou už zmiňovaný datový typ struktury *data_form*. Tato deklarace je k nalezení v souboru *numeric.h*. Struktura v sobě zapouzdřuje všechny parametry, proměnné a příznaky potřebné pro zpracování dat. Mezi ně patří: hodnota vzorkovací frekvence dat, způsob neboli mód vedení záznamu zpracovaných dat, ukazatele na datové zásobníky, proměnné pro mezi výpočty a konečné výsledky efektivních a středních hodnot vypočítaných z naměřených dat, dále struktura obsahuje informace o počtu bloků, které je třeba zpracovat pro dosažení jedné efektivní hodnoty z jednosekundového intervalu měřeného signálu, velikost

jednoho bloku (tyto dva parametry nemůže uživatel ovlivnit, jsou pevně nastavené v rámci inicializace struktury, to samé platí o vzorkovací periodě), informace o tom kolik bloků bylo již zpracováno, nebo kolik efektivních a středních hodnot bylo již zpracováno. Dále také příznak o dokončení zpracování jednoho bloku dat a také časové konstanty pro volitelné filtry typu horní a dolní propust a parametry pro povolení těchto filtrů .Dalšími dvěma parametry jsou parametry týkající se souboru pro záznam zpracovaných dat. Jedná se o ukazatel na textový soubor a ukazatel na konkrétní znak v něm (*offset*). Posledním parametrem je pak příznak, zda má být po dokončení měření vygenerován ze zpracovaných dat soubor typu *.mat kompatibilní s prostředním Matlab. Parametry týkající se filtrů a mód vedení záznamu jsou uživatelsky konfigurovatelné skrz uživatelské rozhraní, ale o něm více v kapitole 5.1.9 a o jeho praktické realizaci v odstavci níže.

Poslední věcí, kterou je třeba zmínit před popisem realizace zpracovávání dat, je fakt, že data od každého z kanálů A/D převodníku nejsou ukládána do dvou samostatných zásobníků ale jen do jednoho a to tak, že všechny sudé prvky odpovídají datům z prvního kanálu A/D převodníku, tedy prvnímu snímači a všechny liché prvky naopak odpovídají datům z druhého kanálu A/D převodníku, tedy druhému snímači.

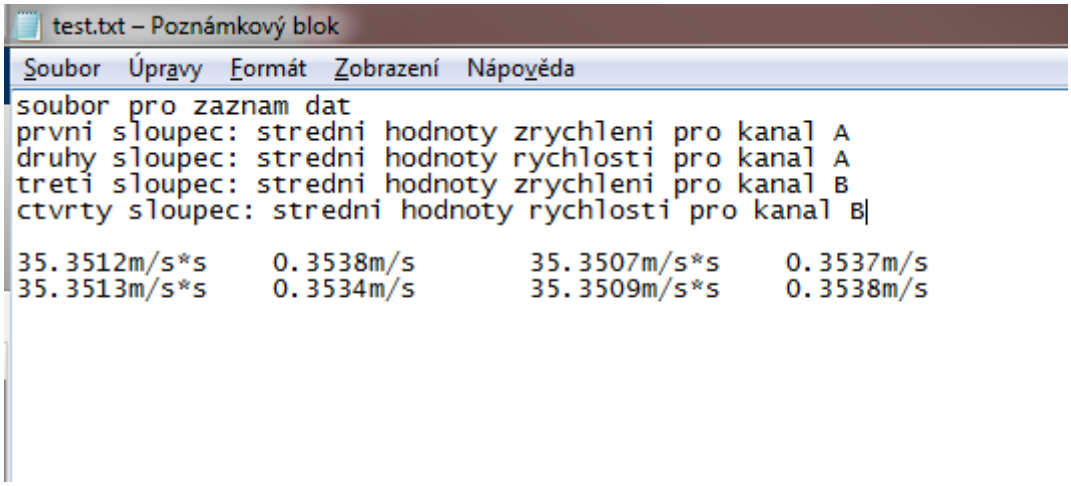
Již výše zmíněná funkce, *mean_val_getNlog*, která je za tohle vše zodpovědná, v sobě zahrnuje výpočet efektivní hodnoty, podmíněný předchozím zpracováním dostatečného množství naměřených dat (16384 vzorků, 128 bloků dat na jeden kanál) a také výpočet střední hodnoty podmíněný dostatečným množstvím zpracovaných efektivních hodnot (pro týdenní záznam 900 efektivních hodnot, pro roční záznam 28800 efektivních hodnot), jednotlivé dílčí hodnoty pro výpočty ať už efektivních nebo středních hodnot nejsou ukládány, ale hned při jejich získání, jsou připočteny k mezi výsledku počítané hodnoty.

Pokud by byl zvolen postup ukládání dílčích hodnot a výpočet efektivní nebo střední hodnoty by byl realizován až jako poslední krok, nebylo by z důvodů nedostatku paměti kam toto množství dat uložit. Funkce *mean_val_getNlog* také obsahuje volání funkce *block_process*, která je zodpovědná za zpracování jednoho bloku naměřených dat. Tato funkce je realizovaná jako konečný stavový automat, který podle uživatelského nastavení buď pomocí filtrů přefiltruje naměřená data, nebo data rovnou integruje a připočte je k mezi výsledku efektivní hodnoty. Všechny algoritmy numerických filtrů v sobě zahrnují potřebu pamatovat si předchozí hodnoty ať už vstupů, nebo i výstupů z filtrů. Tyto hodnoty jsou v paměti mikroprocesoru drženy nejen v rámci zpracování jednoho bloku dat, ale v rámci zpracování všech bloků dat. To znamená, že hodnoty předchozích hodnot vstupů a výstupů jednotlivých filtrů, získané na konci filtrace jednoho z bloků jsou použity jako počáteční podmínky předchozích hodnot vstupů a výstupů pro filtraci dalšího bloku. Tento mechanismus zajišťuje potlačení přechodových jevů způsobených filtry na každém bloku. Pravdou je, že se ale ve zpracovaných hodnotách vliv přechodného děje projeví aspoň na začátku procesu zpracování dat v rámci celého měření. Nicméně praktické měření ukázalo, že průměrování (výpočet efektivních a středních hodnot) z tak velkého množství dat, ze kterého bylo prováděno, vliv přechodného jevu

úplně potlačí. Po výpočtu jedné střední hodnoty (ve skutečnosti jde o čtyři, jedna pro zrychlení, druhá pro rychlost pro každý kanál), se tato hodnota uloží do textového souboru na SD kartu a to pomocí *CreateLogLine*. Tato funkce zajišťuje vytvoření jednoho řádku ze čtyř hodnot (aktuálně zpracované konečné střední hodnoty zrychlení a rychlosti pro oba kanály, snímače) vytvoří textový řetězec a zapíše jej na SD kartu pomocí funkce *WriteToFile*, která zajišťuje otevření souboru, zápisu textového řetězce a následné zavření souboru. Deklaraci popřípadě definici funkce *CreateLogLine* lze dohledat v souboru *LogTools.h* popřípadě *LogTools.c* a zdroje pro funkci *WriteToFile* se nacházejí v souborech *sdTool.h* a *sdTool.c* viz Kompletní aplikace pro mikrokontroler v elektronické příloze. Přepočet na hodnoty rychlosti a zrychlení z naměřených hodnot napětí ze snímačů, se provádí hned na začátku zpracování jednoho bloku naměřených dat a to pomocí zadané citlivosti snímače 10mV/m/s². Popisu uživatelského rozhraní je věnována kapitola 5.1.12 Realizace uživatelského rozhraní.

5.1.10 Uložení dat do textového souboru

V kapitole 5.1.4, byl nastíněn návrh uložení zpracovaných dat na SD kartu. Tato kapitola ukazuje, jak výsledný soubor vypadá a poskytuje návod na orientaci v něm. Na následujícím obrázku je vidět záznam z testovacího měření sinusového signálu o amplitudě 500mV. Ukládání dat bylo prováděno v režimu týdenního ukládání, ale uměle byl počet středních hodnot omezen na dvě, respektive na osm.

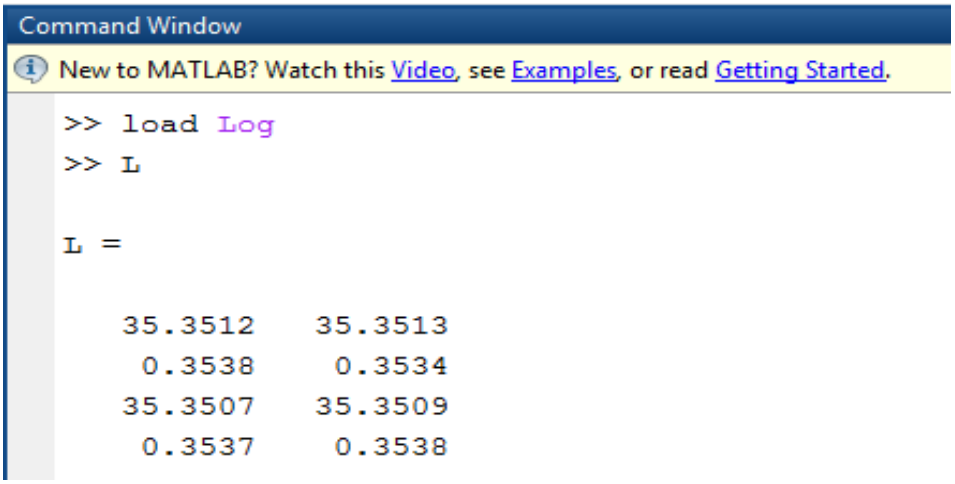


Obrázek 14: zobrazení dat v textovém souboru

Na obrázku lze vidět textový soubor, která je složen ze dvou částí. První částí je průvodní text, který slouží jako legenda. Písmena A a B u kanálů mají význam označení kanálu 1 a 2. Tato podoba byla zvolena z toho důvodu, že funkce, která extrahuje číselné údaje z textového souboru a používá je pro alternativní ukládání dat, viz následující kapitola, by milně pokládala za platný číselný údaj i číselné označení kanálů. Druhou části souboru jsou už samotná zaznamenaná data včetně jednotlivých jednotek veličin.

5.1.11 Alternativní způsob uložení dat

Ukládání dat do textového souboru je mnohdy nepraktické. Proto byla přidána možnost uložení dat do souboru typu: *.mat úrovně 4, kompatibilním s prostředními *Matlab* od verze 4.2. Tato úroveň umožňuje zapouzdřit pouze 1D a 2D datová pole (vektory a matice). Buňková pole nejsou zahrnuty. Prostředí *Matlab* je robustní nástroj pro matematické výpočty, návrh matematických a fyzikálních modelů a jejich simulace.
Struktura souboru je jednoduchá, její detailní popis je možno nalézt v [27]. Skládá se ze dvou částí: z části hlavičky a části dat. Hlavička má velikost 20 Bytů a nachází se v ní informace o datovém typu, počtu řad a sloupců matice, dále také příznak zda matice či vektor obsahuje imaginární část čísel a jako posledním parametrem hlavičky je délka textového řetězce názvu proměnné zvětšeného o jedna. Část hlavičky datový typ se skládá z několika nastavitelných bitových polí. První slouží k výběru formátu binární logiky, tak jak ji chápe a pracuje s ní konkrétní procesorová platforma (*Little Endian, Big Endian* atd..). Další konfigurovatelným polem je pole, které slouží pro zvolení datového formátu, s jakým jsou data uložena (*int, float, double* atd…). Posledním konfigurovatelným polem je pole určující typ matice. Tím se myslí, zda je numerická, textová, nebo řídká (dominující počet nul nad počtem ostatních čísel). Část dat obsahuje textový řetězec názvu proměnné, po kterém už následují data v binární podobě korespondující se zvoleným datovým typem dat v hlavičce. První se do souboru zapisují reálná data a až po nich se zapisují data, která mají význam imaginárních částí. Imaginární data nejsou povinná. Pro pochopení výsledné prezentace dat po exportu z textového souboru je třeba zmínit, že prostředí Matlab naplňuje matice po sloupcích ne po řádcích.
Funkce, která toto zajišťuje, se nazývá *GenMatFil* a její zdroje jsou dohledatelné v souborech *Txt2Mat.h* a *Txt2Mat.c* v rámci elektronické přílohy této práce. Funkce vytváří soubor *Log*.mat obsahující jednu proměnou typu matice 4*počet středních hodnot (tento parametr závisí na zvoleném módu ukládání, buďto týdenní, nebo roční) s názvem *L* a dále pomocí funkce *Txt2Flt* prochází textovým souborem a textové řetězce, které mají význam hodnot pak převádí na číselné hodnoty a ty jsou pak zapsány do souboru typu *.mat. Tato funkce prochází soubor po řádcích, ale prostředí Matlab, jak už bylo zmíněno, plní matice po sloupcích a proto je výsledná matice transponovaná oproti matici dat v textovém souboru. Tento mechanismus se děje až po skončení měření a je podmíněn uživatelskou volbou při nastavování parametru měření před jeho spuštěním. Je třeba zmínit, že tyto data neobsahují časové známky, ale i tak, v tomto provedení, poskytují možnost snadnější práce s výsledky. Pomocí Prostředí *Matlab* lze s takovými daty lehce provádět statistické výpočty nebo tyto data vyobrazovat do grafů, což se souborem typu *.txt není možné. Následující obrázek demonstruje reprezentaci dat ve vygenerovaném souboru typu *.mat:

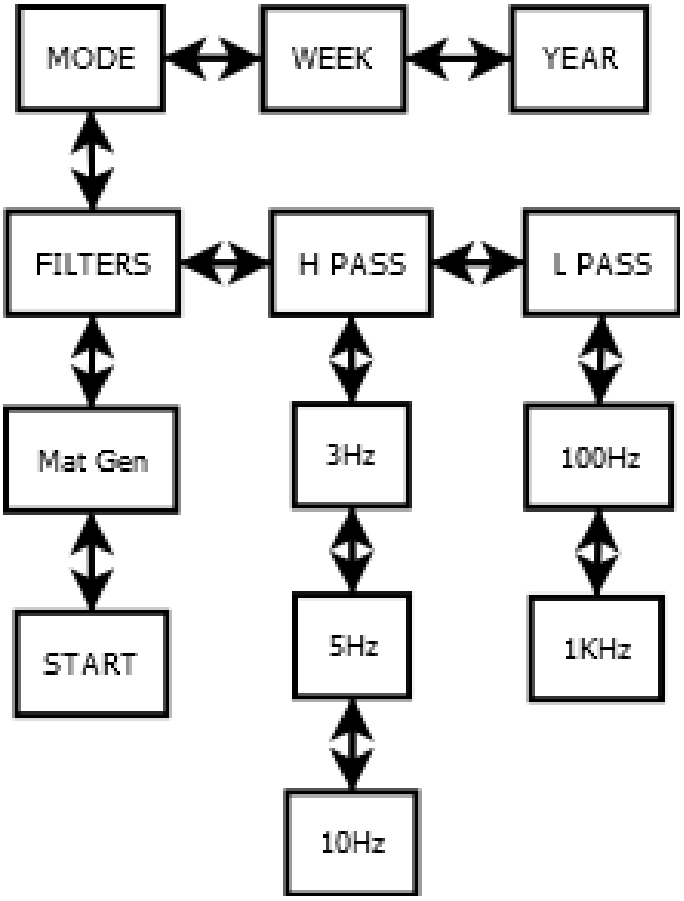


*Obrázek 15:zobrazení dat po vygenerování *.mat souboru*

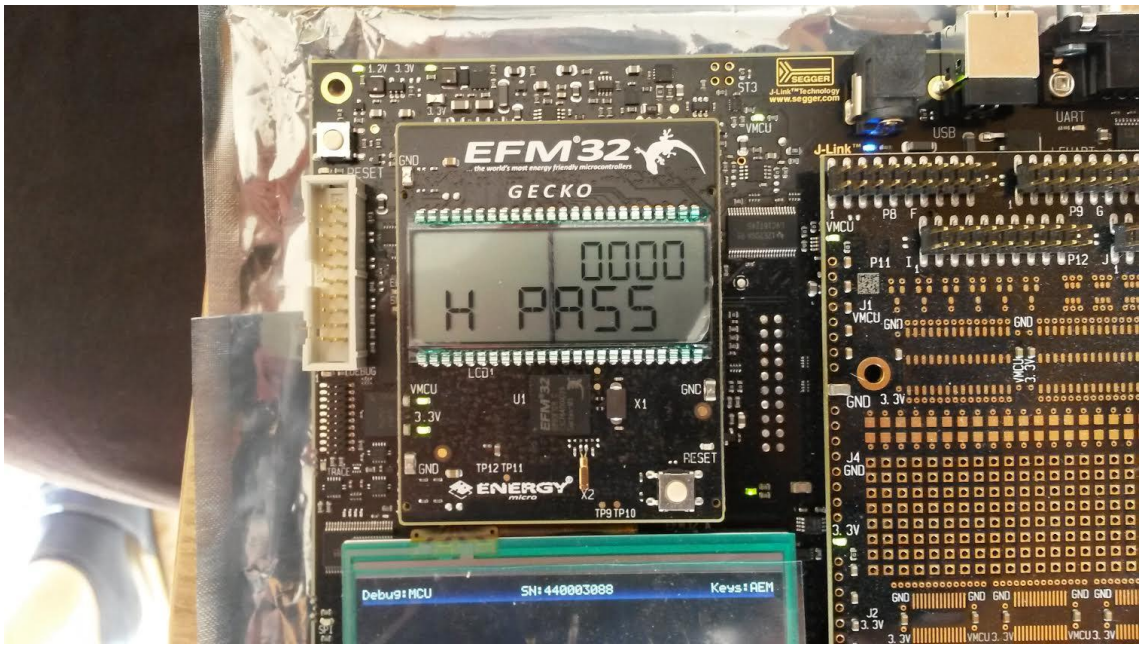
Jako zdroj dat byl použit textový soubor z předchozí kapitoly soubor obrázku je jasná transpozice matice dat oproti matici dat v textovém souboru v kapitole 5.1.10.

5.1.12 Realizace uživatelského rozhraní

Uživatelské rozhraní je nezbytnou součástí zařízení. Slouží k nastavení parametrů měření a k ovládání celého zařízení. Rozhraní je tvořeno ze dvou částí. První je LCD segmentový displej a druhou jsou standardní vstupy a výstupy mikrokontroleru.
Celkem má uživatel k dispozici šest vstupů realizovaných jako vstupy s *pullup* rezistorem 40KΩ, který na vstupu zajišťuje úroveň logické jedničky a vstup samotný je pak aktivní v úrovni logické nuly. Vstupy jsou dále vybaveny interním filtrem proti mechanickým zákmitům, který odstraňuje ze vstupního signálu pulsy o šířce 10ns až 50ns, viz technická dokumentace procesoru [28]. Pro rozhraní byl použit port A mikroprocesoru a piny 0 až 6. První čtyři piny mají význam směrových kurzorů pro orientaci v menu, pátý pin má význam potvrzení volby a šestý pin slouží pro přerušení měření. Pin 0 má význam kurzoru „vlevo“, pin 1 kurzoru „vpravo“, pin 2 kurzoru „nahoru“ a pin 3 má pak význam kurzoru „dolů“. Na každý ze vstupů lze připojit tlačítko připojené k zemi.
LCD displej slouží k zobrazení uživatelského menu, jeho struktura je patrná z následujícího obrázku č. 15. Procesor samotný disponuje řadičem pro až 40×40 segmentový LCD displej, proto pro realizaci úlohy stačí kterýkoliv jiný LCD segmentový displej se stejnými parametry a uživatelské menu bude stále realizovatelné. Toto zajišťuje přenositelnosti aplikace z vývojového prostředku na finální výrobek bez závislosti na vývojové platformě. Pomocí kurzorů se lze pohybovat po trasách vyznačených šipkami. Výběr položky je indikován názvem položky, volby menu na displeji. Některé volby lze potvrzovat pomocí pinu 4, který je k tomuto účelu určen. Platnost výběru volby je indikována číslem jedna na číselném údaji, který je vyobrazen nad aktuálně vybraným názvem aktivní položky menu, popřípadě číslem 0, když volba není vybrána, viz obrázek č. 16. Mezi takovéto položky patří například: položka *WEEK* sloužící pro potvrzení výběru módu ukládání dat, nebo položka *Mat Gen*, která slouží pro povolení vygenerování souboru typu *.mat po skončení měření.



Obrázek 16:struktura menu



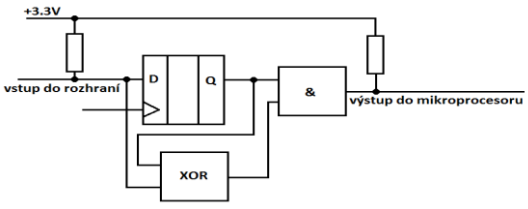
Obrázek 17: ukázka menu na LCD displeji

Na LCD Displeji je pro příklad zobrazena položka *H PASS*, která slouží pro připojení numerického filtru do algoritmu zpracování dat. Nula nad názvem položky značí, že tato volba není vybrána pomocí potvrzovacího vstupu na pinu 4, filtr není připojen. Na následujícím obrázku je ukázka, jak vypadá vybraná položka pomocí potvrzovacího vstupu.

Obrázek 18: ukázka LCD menu

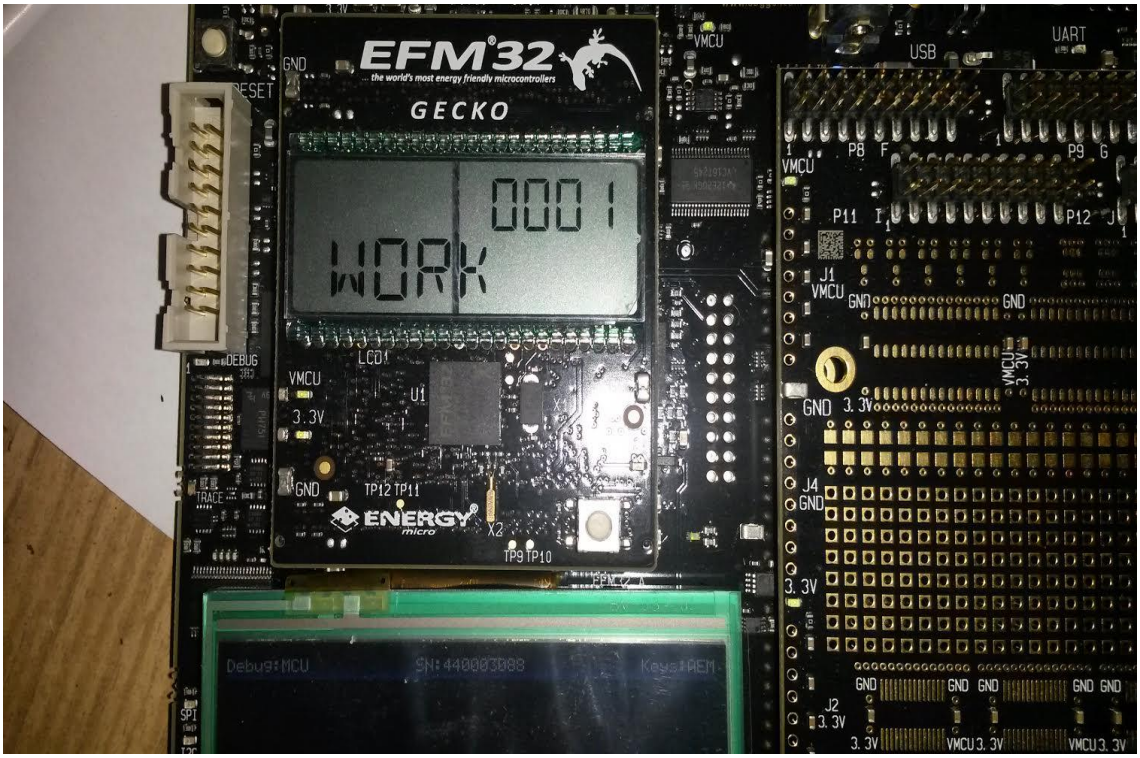
Na tomto obrázku je vidět výběr volby *Mat Gen,* která slouží pro vygenerování souboru typu *.mat. Jednička nad jejím názvem indikuje, že tato volba byla potvrzena potvrzovacím vstupem.

Po stránce programové, je menu realizováno, jako stavový automat přepínající mezi jednotlivými položkami podle toho jaký kurzor byl stisknut. Samotný algoritmus vyhodnocování stavu vstupů funguje tak, aby byla vyhodnocena jako platná informace pouze okamžik změny vstupní logické úrovně z logické jedničky do logické nuly (protože vstup je aktivní v logické nule). To je dosaženo detekcí náběžné hrany pomocí porovnání stavu při změně úrovně na vstupu a předchozímu stavu při poslední změně úrovně na vstupu. Pokud jsou tyto dva stavy různé a předchozí stav byl logická jedna, znamená to, že logická úroveň se změnila z jedničky do nuly. To pro vstup, který je aktivní v logické nule znamená, že došlo k okamžiku stisku tlačítka. Následující obrázek demonstruje, jak by se takovýto algoritmus řešil pomocí logických obvodů:



Obrázek 19: algoritmus detekce hran

Klopný obvod typu zde slouží jako paměťový element pro zapamatování předchozího stavu. V aplikaci byla pro tuto potřebu použita proměnná. Dvouvstupový logický obvod XOR plní funkci zjištění změny mezi aktuálním a předchozím stavem a nakonec logický obvod AND vyhodnocuje, k jaké změně došlo, v tom případě se na jeho výstupu objeví logická jednička tehdy, když dojde na vstupu do rozhraní ke změně z úrovně logická jedna na úroveň logická nula. Počet položek menu je jasný z obrázku č. 15. při potvrzení možnosti START, dojde ke spuštění měření a na displeji se zobrazí WORK, program nastaví dodatečné parametry pro měření, aktivuje čítač pro časování okamžiků měření vzorků ze snímačů a posléze zavolá funkci EMU_EnterEM1, která uvede procesor do energeticky výhodného módu. Tato funkce se volá pokaždé po skončení zpracování jednoho bloku dat, protože každé zavolání call-back funkce, která se automaticky volá po dokončení DMA transakce dat, se procesor z energeticky úsporného režimu probere.
Posledním nezmíněným vstupem je vstup plnící funkci přerušení měření. Tento vstup je implementován jako zdroj externího přerušení mikroprocesoru. Stejně jako ostatní vstupy rozhraní je i on aktivní v logické nule a taktéž je ošetřen filtrem proti mechanickým zákmitům. Při jeho aktivaci dojde k vyvolání přerušení, které způsobí zastavení čítače určující okamžiky převodů A/D převodníku a vynuluje a znovu inicializuje soubor pro záznam dat. Taktéž zavolá funkci menu, díky kterému může dojít k nastavení nového měření.



Obrázek 20:signalizace běžící aplikace

5.1.13 Zhodnocení praktické realizace

Výsledkem praktické realizace je funkční aplikace, která splňuje zadané kritéria měření. Aplikace je schopna počítat střední hodnoty ze zadaného časového úseku (jedna vteřina), počítat střední hodnoty z těchto efektivních hodnot dle dvou zadaných režimu ukládání dat a je schopna je ukládat do textového souboru na SD kartě. Aplikace sice nedisponuje barevným TFT LCD displejem s dotykovým rozhraním, protože vodiče nezbytné pro komunikaci mezi ním a mikrokontrolerem jsou využity jako vstupy pro A/D převodník. Nicméně realizovaná varianta se LCD segmentovým displejem a fyzickým uživatelským rozhraním variantu s TFT displejem ji dostatečně nahrazuje. Aplikace sice není schopná vyobrazovat průběžné nasnímané průběhy zrychlení vibrací a vypočítané průběhy rychlostí vibrací a také nedisponuje výpočtem rychlé Fourierovy transformace (FFT), ale za to umožňuje z naměřených dat vygenerovat soubor typu *.mat kompatibilní s prostředím Matlab, který umožňuje snadnější zpracování získaných dat.

6 VÝSLEDKY TESTOVÁNÍ APLIKACE

Tato kapitola poskytuje výsledky z testovacích měření aplikace. Z důvodů časové náročnosti testování byl počet výsledných zpracovaných středních hodnot omezen na čtyři pro režim týden (měření trvalo jednu hodinu) a na jednu hodnotu pro režim rok (měření této jedné hodnoty trvalo osm hodin). Ve skutečnosti by se však jednalo o 672 výsledků zpracování v režimu týden a 1095 výsledků pro režim rok (je třeba si uvědomit, že se jedná o počet čtveřic). Jako zdroje signálu byl použit sinusový signál s frekvencí 100Hz a amplitudou 500mv. Výsledné hodnoty měření tohoto signálu se vždy pohybovaly okolo hodnoty 353.5mV respektive 35,35m/s² pro hodnoty zrychlení a 3,535mV respektive 0,3535m/s pro hodnoty rychlosti, což odpovídá předpokládané skutečnosti podle známých vztahů pro harmonické průběhy:

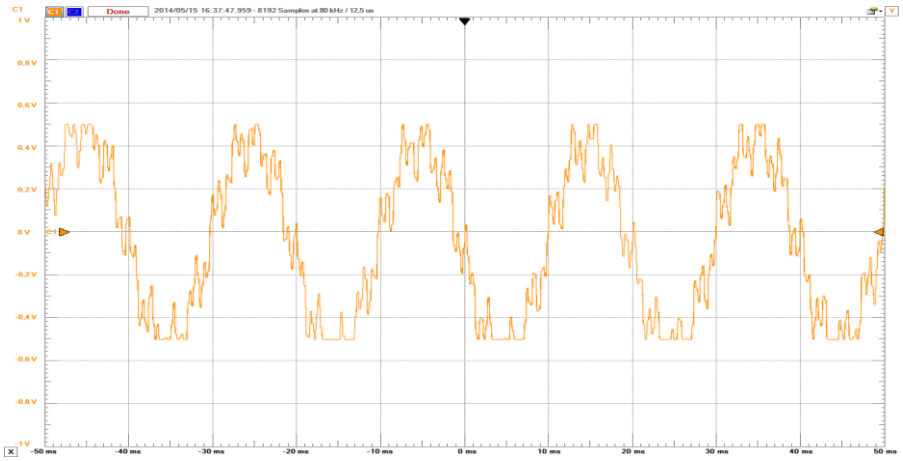
$$ef = A \times \frac{\sqrt{2}}{2} \tag{6.1}$$

Kde ef je výsledná efektivní hodnota a A je amplituda signálu.

$$v_{(t)} = \int a_{(t)} dt \tag{6.2}$$

$$\int A_a * \sin(\omega t)\, dt = -\frac{\mathrm{A}_v * \cos(\omega t)}{\omega} \tag{6.3}$$

Kde v je rychlost v čase a a je zrychlení v čase. Po zjištění správnosti výsledku byl aplikován signál tvořený součtem sinusových signálu s frekvencemi 3Hz, 5Hz, 50Hz, 400Hz a 3Khz kde složka 50Hz měla amplitudu 500mV a ostatní měly amplitudu 100mV. Tento signál se považoval za případný reálný signál ze snímačů.



Obrázek 21: referenční signál pro měření

Signál na obrázku je výše popsaná suma sinusových signálu. Referenční měření ukázalo, že efektivní hodnota signálu je 346.8mV to podle převodní konstanty snímače odpovídá 34,8m/s². V této kapitoly budou zveřejněny výsledky měření při zapojení či odpojení filtrů v měřícím systému aplikace. Pro oba kanály byl použit jeden a tentýž signál.

Tabulka 1: měření referenčního signálu v režimu týden, varianta bez filtrů

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,7891m/s²	0,6836m/s	34,6431m/s²	0,6936m/s
34,6643m/s²	0,6916m/s	34,6111m/s²	0,6906m/s
34,7765m/s²	0,7001m/s	34,7831m/s²	0,6941m/s
34,6849m/s²	0,6899m/s	34,7091m/s²	0,6943m/s

V tabulce 1 lez vidět výsledky dosažené měřením v režimu týden a bez přítomnosti numerických filtrů. Následující tabulka zobrazuje výsledky stejné konfigurace filtrů, ale v režimu rok.

Tabulka 2: měření referenčního signálu v režimu rok, varianta bez filtrů

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6911m/s²	0,6935m/s	34,6911m/s²	0,6934m/s

Následující dvě tabulky ukazují výsledky měření se zapojeným filtrem typem horní propust s mezní frekvencí 5Hz.

Tabulka 3: měření referenčního signálu v režimu týden, varianta s filtrem typu horní propust

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,7991m/s²	0,6960m/s	34,6999m/s²	0,6940m/s
34,7851m/s²	0,6957m/s	34,678m/s²	0,6936m/s
34,7919m/s²	0,6958m/s	34,7001m/s²	0,6940m/s
34,6101m/s²	0,6922m/s	34,6854m/s²	0,6937m/s

Tabulka 4: měření referenčního signálu v režimu rok, varianta s filtrem typu horní propust

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6811m/s²	0,6937m/s	34,6915m/s²	0,6938m/s

Následující dvojice tabule obsahuje data získaná s připojeným filtrem typu dolní propust s mezním kmitočtem 1KHz.

Tabulka 5: měření referenčního signálu v režimu týden, varianta s filtrem typu dolní propust

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6121m/s²	0,6922m/s	34,6971m/s²	0,6939m/s
34,6510m/s²	0,6930m/s	34,6809m/s²	0,6936m/s
34,6191m/s²	0,6924m/s	34,6432m/s²	0,6929m/s
34,5999m/s²	0,6920m/s	34,6810m/s²	0,6936m/s

Tabulka 6: měření referenčního signálu v režimu rok, varianta s filtrem typu dolní propust

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6799m/s²	0,6936m/s	34,6844m/s²	0,6937m/s

Poslední dvojice tabulek zobrazuje hodnoty získané zpracováním dat s filtry obou typů. S mezními frekvencemi 5Hz pro filtr typu horní propust a 1Khz pro filtr typu dolní propust

Tabulka 7: měření referenčního signálu v režimu týden, varianta s oběma typy filtrů

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6711m/s²	0,6934m/s	34,6771m/s²	0,6935m/s
34,6810m/s²	0,6936m/s	34,6609m/s²	0,6932m/s
34,7191m/s²	0,6944m/s	34,6732m/s²	0,6935m/s
34,8999m/s²	0,6980m/s	34,6610m/s²	0,6932m/s

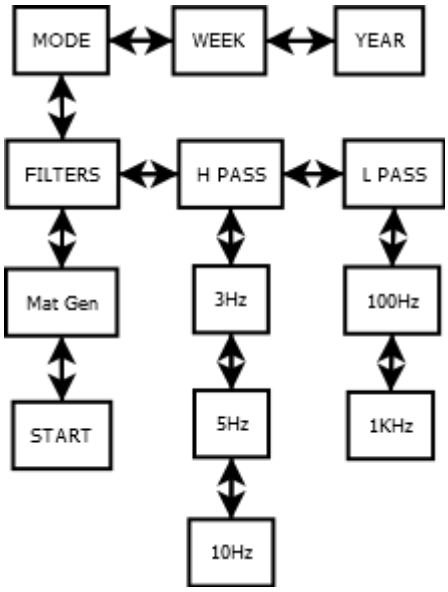
Tabulka 8: měření referenčního signálu v režimu rok, varianta s oběma typy filtrů

kanál 1		kanál 2	
střední hodnoty zrychlení	střední hodnoty rychlosti	střední hodnoty zrychlení	střední hodnoty rychlosti
34,6802m/s²	0,6936m/s	34,6798m/s²	0,6935m/s

Z naměřených hodnot lze vidět, že ani jeden z filtrů nezatěžuje výsledky viditelnou chybou. V první tabulce při měření režimu týden na kanálu číslo jedna lze pozorovat viditelné odchylky. Jelikož byl pro oba kanály použit tentýž signál, lze tyto chyby prohlásit za náhodné chyby způsobené výpočetním algoritmem. Ze znalosti efektivní hodnoty referenčního signálu lze určit maximální odchylku hodnot zrychlení, ta činí 0,2199m/s².

7 UŽIVATELSKÝ MANUÁL

Každé fyzické zařízení, u kterého se předpokládá, že bude používáno cizí osobou, by mělo mít uživatelskou příručku, která by takového uživatele provedla funkčností a objasnila mu způsoby zacházení s takovýmto zařízením. Tato kapitola slouží právě jako takováto příručka. Poskytne souhrn toho, co vyplívá z předcházejících textů této práce a konkrétně to, jak se dá zařízení nastavit a možnosti připojení snímačů.
Po zapnutí zařízení dojde k inicializaci všeho potřebného, viz kapitola 5.1.9 Praktická realizace algoritmu. Poté se na znakovém displeji objeví zpráva *MODE* říkající, že se uživatel nachází na položce *MODE* v rámci uživatelského rozhraní. Pro lepší orientaci zde bude ještě jednou uveden obrázek ilustrující strukturu uživatelského menu.



Obrázek 22: struktura uživatelského menu

Pomocí uživatelských vstupů se lze v tomto menu pohybovat a pomocí vstupu pro potvrzení lze potvrdit případnou volbu. Pro pořádek je zde znovu uvedeno fyzické umístění uživatelských vstupů. Uživatelské vstupy, se nacházejí na portu A mikrokontroleru a jeho jednotlivé piny mají následující význam: pin číslo 0 má funkci kurzoru vlevo, pin číslo 1 má funkci kurzoru vpravo, pin číslo 2 má význam kurzoru nahoru, pin číslo 3 má význam kurzoru dolů, pin číslo 4 má význam potvrzovacího vstupu a pin číslo 5 má význam vstupu pro přerušení měření a uvedení zařízení do výchozího stavu. Funkce každé položky menu bude rozepsána níže.

Položka MODE značí, že se uživatel nachází na položce MODE uživatelského menu. Pohybem vlevo pomocí patřičného vstupu se lze dostat na podpoložky, které slouží pro výběr konkrétního módu ukládání zpracovaných dat. Pohybem dolů v menu se lze dostat na položku FILTERS, přes kterou se lze dostat na položky sloužící pro detailní nastavení filtrů

Položka WEEK slouží pro zvolení módu ukládání zpracovaných dat v režimu týden, tedy ukládání hodnot co 15 minut podobu jednoho týdne. Pro výběr tohoto módu je třeba

vybrat tuto položku v menu a potvrdit potvrzovacím vstupem. Potvrzení výběru se projeví jako zobrazení čísla jedna nad názvem této položky. V Případě zobrazené nuly, výběr nastavení, které se pod touto položkou skrývá, není potvrzeno.

Položka YEAR slouží pro zvolení módu ukládání zpracovaných dat v režimu rok, tedy ukládání hodnot co 8 hodin podobu jednoho roku. Pro výběr tohoto módu je třeba vybrat tuto položku v menu a potvrdit potvrzovacím vstupem. Potvrzení výběru se projeví jako zobrazení čísla jedna nad názvem této položky. V Případě zobrazené nuly, výběr nastavení, které se pod touto položkou skrývá, není potvrzeno.

Položka FILTERS značí, že se uživatel nachází na položce FILTERS uživatelského menu. Pohybem vlevo pomocí patřičného vstupu, se lze dostat na detailní nastavení číslicových filtrů.

Položka H PASS slouží pro připojení číslicového filtru typu horní propust do měřícího systému. Pro výběr tohoto filtru je třeba vybrat tuto položku v menu a potvrdit potvrzovacím vstupem. Pod touto položkou se nacházejí položky, které slouží pro výběr konkrétní mezní frekvence filtru. Mechanismus výběru je stejný jako pro výběr zde popisované položky. Potvrzení výběru se projeví jako zobrazení čísla jedna nad názvem této položky. V Případě zobrazené nuly, výběr nastavení, které se pod touto položkou skrývá, není potvrzeno.

Položka L PASS slouží pro připojení číslicového filtru typu dolní propust do měřícího systému. Pro výběr tohoto filtru je třeba vybrat tuto položku v menu a potvrdit potvrzovacím vstupem. Pod touto položkou se nacházejí položky, které slouží pro výběr konkrétní mezní frekvence filtru. Mechanismus výběru je stejný jako pro výběr zde popisované položky. Potvrzení výběru se projeví jako zobrazení čísla jedna nad názvem této položky. V Případě zobrazené nuly, výběr nastavení, které se pod touto položkou skrývá, není potvrzeno.

Položka Mat Gen, slouží pro případné vygenerování souboru typu *.mat. Potvrzení této volby se provede potvrzením pomocí potvrzovacího vstupu.

Položka START slouží pro zahájení sběru a zpracování dat. Po potvrzení této volby, se na displeji objeví text WORK značící že zařízení pracuje.

Pro připojení snímačů na jednotlivé vstupy je nutné dodržet napěťové limity jednotlivých pinů. Minimální elektrické napětí, které lze připojit na pin mikroprocesoru je -0,6V vztažené proti společnému referenčnímu potenciálu napájení. Dále je třeba znovu uvést, že použité referenční je 1,25V tedy +/-625mV. Pokud dojde k překročení tohoto napěťového intervalu, může dojít k poškození zařízení.

Postup pro uskutečnění měření je následující:

Před zapnutím zařízení (vývojového prostředku), je třeba vložit paměťovou SD kartu se souborovým systémem FAT32. jej přítomnost je nutná pro bezchybné provedení inicializace.

Po zapnutí se na LCD displeji objeví text MODE signalizující, že veškerá inicializace proběhla v pořádku a uživatel se nyní nachází v prostředí uživatelského menu, skrze které je schopen nastavit potřebné parametry měření.

Po nastavení všech potřebných parametrů musí uživatel potvrdit volbu START a měření je tímto spuštěno. Měření je kdykoliv možné přerušit pomocí vstupu pro přerušení měření. Po tomto se zařízení vrátí do výchozího stavu.

8 ZÁVĚR

Práce poskytuje rešerši tří dostupných vývojových prostředků na trhu. První (stm32 VL Discovery) nenabízel dostatečnou výbavu z hlediska periferií na vývojovém kitu, ale jeho tržní cena je poměrně nízká, pohybuje se okolo 200 korun českých. Druhý (Atmel SAM3S-EK) poskytoval už dostatečnou výbavu, ale nebyl fyzicky dostupný. Třetí nabízel taktéž dostatečnou výbavu fyzických periferií implementovaných na kitu a navíc obsahoval procesor určený pro aplikace kde je brán ohled na spotřebu energie.

Z výše popsaných vývojových prostředků byl vybrán třetí, z důvodů obsahu procesoru určeného pro energeticky úsporné aplikace také kvůli jeho fyzické dostupnosti. Tento vývojový prostředek nese označení EFM32 DK3550 a obsahuje čip EFM32G890F128 firmy Energymicro.

Pro tento zvolený vývojový prostředek byla zvolena úloha Data logger vibrací zrychlení pro diagnostiku točivých strojů, který má zajišťovat sběr dat ze dvou piezoelektrických snímačů zrychlení a tyto hodnoty dále zpracovávat do podoby efektivních hodnot z určitých časových intervalů a následně zpracoval do tvaru středních hodnot, tyto hodnoty pak zaznamenával po dobu danou konkrétním zadáním úlohy. Součástí výsledné aplikace taktéž mělo být uživatelské rozhraní pro možnost ovládání a konfigurace zařízení a aplikace měla zahrnovat algoritmy pro numerické filtry.

Periferie obsažené v mikrokontroleru jako například DMA (Direct memory access) a PRS (Peripheral Reflex System) umožňující interakci mezi periferiemi jako je například čítač použitý pro časování okamžiků převodů A/D převodníku, nebo přenos dat, tak aby nebyla pro tyto operace nutná činnost procesoru a ten mohl setrvávat v energeticky úsporném režimu, zaručují nízkou spotřebu celé aplikace. Pracovní frekvence procesoru 32MHz a instrukční sada architektury ARM Thumb2 zaručují velkou výpočetní rychlost potřebnou pro vykonání algoritmů numerických filtrů a vykonání algoritmů pro vypracování naměřených dat.

Výsledná aplikace realizující úlohu Data logger vibrací zrychlení pro diagnostiku točivých strojů byla implementována do mikroprocesru s jádrem ARM Crotex-M3 pomocí programovacího jazyka C a vývojového prostředí IAR Embeded Workbench. Implementovaná aplikace v sobě zahrnuje zpracování naměřených dat dle zadání úlohy, umožňuje ukládat zpracované data ve dvou režimech, režim týden a režim rok, dále zajišťuje uložení zpracovaných dat do textového souboru a navíc přidává možnost vygenerovat soubor typu *.mat, který je kompatibilní s prostředím Matlab a umožňuje snadnější zpracování získaných dat. Výsledná aplikace taktéž v sobě zahrnuje uživatelské rozhraní složené z LCD segmentového displeje a fyzických vstupů, které umožňují ovládání a konfiguraci aplikace. Displej pak slouží pro zobrazení menu, které taktéž slouží pro ovládání a konfiguraci aplikace. Podrobný popis realizace implementace aplikace je obsažen v kapitole 5.1.9 praktická realizace algoritmu.

Experimentální měření parametrů výsledné aplikace bylo provedeno následovně: nejprve byl jako zdroj signálu použit signálu sinusový signál s frekvencí 100Hz a amplitudou 500mV. Výsledné průměrné hodnoty byly ověřeny jednoduchým výpočtem, viz kapitola

6 výsledky testování aplikace. Po zjištění správnosti výsledku byl vygenerován referenční signál skládající se z několika dílčích sinusových signálů a ten byl pak použit jako zdroj signálu pro testovací měření. Výsledky lze nalézt v kapitole 6 výsledky testování měření. Implementovaná aplikace zvládá dodržet kritéria dané zadáním úlohy Data logger vibrací zrychlení pro diagnostiku točivých strojů. Procesor zvládá data zpracovat mezi tím, než jsou k dispozici nová, změřená data.

Aplikace takto navrhnutá je vhodná nejen pro diagnostiku točivých strojů, pro kterou byla původně navrhnuta, ale taktéž je vhodná pro diagnostiku všech mechanických či konstrukčních zařízení, které jsou vystaveny mechanickým vibracím a je třeba zjišťovat zda, se tyto vibrace s časem mění. Při vývoji této aplikace došlo k několika komplikacím a problémům. První z nich byla neschopnost zahrnout do výsledné aplikace dotykový TFT LCD displej. Tento displej bohužel sdílí na použitém vývojovém kitu vodiče s A/D převodníkem. Pro displej mají tyto vodiče význam komunikace s mikroprocesorem, ale pro A/D převodník to jsou analogové vstupy. Další problém při vývoji aplikace byl programového charakteru. Autorovi znalosti a zkušenosti nebyly dostatečné pro vyřešení tohoto problému, ale s pomocí vedoucího práce byl problém naštěstí vyřešen. Tento problém se stal těsně před dokončením celé aplikace a celé publikace ji popisující a představoval hrozbu včasného nedokončení práce. Pomoc vedoucího práce však zajistila včasné dokončení. Práce byla autorovi přínosem. Pomohla autorovi prohloubit své znalosti v mikroprocesorové technice a ve způsobech signálového zpracování dat, jako byly například návrhy algoritmů pro číslicové filtry. Hlavním přínosem autora byl návrh algoritmu pro vygenerování souboru typu *.mat a celý koncept měřícího systému pro zpracování naměřených dat. Aplikace nabízí i prostor k vylepšení. Prvním zlepšením by mohlo být použití pokročilejší architektury Cortex-M4, která disponuje matematickým koprocesorem *FPU* (Float Processor Unit), která nabízí podporu pro operace s desetinnými čísly a zrychluje i zjednodušuje operace pro výpočty numerického zpracování naměřených dat. Jedná se o veškeré matematické operace s desetinnými čísly jako je třeba dělení, násobení, odmocniny, ale i sčítání a odečítání desetinných čísel.

Seznam obrázků

Seznam tabulek

Seznam příloh

Literatura

[1] VÁŇA, Vladimír. *ARM pro začátečníky*. 1. vyd. Praha: BEN-technická literatura, 2009. ISBN 978-80-7300-246-6.

[2] Úvod do architektury Cortex-M3 - díl. 1. In: *Pandatron.cz* [online]. 2010 [cit. 2014-01-02]. Dostupné z: http://pandatron.cz/?1252&uvod_do_architektury_cortex-m3_-_dil._1

[3] Blokové schéma jádra Cortex-M3. In: *Pandatron.cz* [online]. 2010 [cit. 2014-01-02]. Dostupné z: http://pandatron.cz/elektronika2/cortexm3_1_cortexm3.gif

[4] Úvod do architektury Cortex-M3 - díl. 2. In: *Pandatron.cz* [online]. 2010 [cit. 2014-01-02]. Dostupné z: http://pandatron.cz/?1281&uvod_do_architektury_cortex-m3_-_dil._2

[5] Pipe-line. In: *Pandatron.cz* [online]. 2010 [cit. 2014-01-02]. Dostupné z: http://pandatron.cz/elektronika2/cortexm3_2_pipeline.gif

[6] STM32VLDISCOVERY. In: *St.com* [online]. 2013 [cit. 2014-01-02]. Dostupné z: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1199/PF250863

[7] STM32F100RB Reference manual. In: *St.com* [online]. 2011 [cit. 2014-01-02]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/CD00246267.pdf

[8] STM32VLDISCOVERY. In: \ [online]. 2013 [cit. 2014-01-02]. Dostupné z: http://www.st.com/st-web-

[9] SAM3S-EK Development Board User Guide. In: Www.atmel.com [online]. 2011 [cit. 2014-01-17]. Dostupné z: http://www.atmel.com/Images/doc11031.pdf

[10] SAM3s Series Datasheet. In: Www.atmel.com [online]. 2013 [cit. 2014-01-17]. Dostupné z: http://www.atmel.com/Images/Atmel_6500_32-bit-Cortex-M3-Microcontroller_SAM3S_Datasheet.pdf

[11] SAM4. In: Http://www.freertos.org/ [online]. 2014 [cit. 2014-01-17]. Dostupné z: http://www.freertos.org/SAM4.jpg

[12] EFM32G-DK3550 User Guide. In: Www.energymicro.com [online]. 2013 [cit. 2014-01-17]. Dostupné z: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32g-dk3550-ug.pdf

[13] EFM32G890 Reference Manual. In: Www.energymicro.com [online]. 2013 [cit. 2014-01-17]. Dostupné z: http://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32G890.pdf

[14] R7904132-01. In: Http://australia.rs-online.com/ [online]. 2013 [cit. 2014-01-17]. Dostupné z: http://img-asia.electrocomponents.com/largeimages/R7904132-01.jpg

[15] Energy-micro. In: Http://ww.koreaittimes.com/ [online]. 2012 [cit. 2014-01-17]. Dostupné z:

Nf/EABkBAQADAQEAAAAAAAAAAAAAAABAgQDBf/EACMRAAMAAgICAwADA
QAAAAAAAAABAgMREjEhURMyQQQiQmH/2gAMAwEAAhEDEQA/APcUREARFntId
[16] IAR Embedded Workbench IDE User Guide IDE User Guide. In: Www.iar.com [online]. 2012 [cit. 2014-01-17]. Dostupné z: ftp://ftp.iar.se/WWWfiles/RX/webic/doc/EWRX_IDEGuide.ENU.pdf
[17] IAR_CMYK_small. In: Http://www2.renesas.eu/ [online]. 2014 [cit. 2014-01-17]. Dostupné z: https://www.google.cz/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&docid=ze-xt7-ehUNHhM&tbnid=xMdVFNlIaRsggM:&ved=0CAUQjRw&url=http%3A%2F%2Fwww2.renesas.eu%2Fap_newsletter%2FOctober-2013%2F002_iar%2Findex.html&ei=S3HZUqaDIYiw0QXb24H4Bw&bvm=bv.59568121,d.bGE&psig=AFQjCNFO0j3tMxoh4pvh4ewvkbe664sTMg&ust=1390068422854937
[18] Simplicity-studio-tn. In: Http://www.silabs.com/ [online]. 2014 [cit. 2014-01-17]. Dostupné z: http://www.silabs.com/SiteCollectionImages/Misc/simplicity-studio-tn.png
[19] BENEŠ, Petr, Jiří FIALKA, Zdeněk HAVRÁNEK, Stanislav KLUSÁČEK, Tomáš KOPECKÝ, Miroslav KRUPA, Miroslav UHER a Martin VÁGNER. Měření fyzikálních veličin. Brno, 2013. Skriptum. VUT
[20] Figure3. In: Http://www.mmf.de/ [online]. 2008 [cit. 2014-01-17]. Dostupné z: http://www.mmf.de/characteristics.htm
[21] SMÉKAL, Zdeněk a Petr SYSEL. Číslicové filtry. Brno, 2013. Dostupné z: http://foxeightyfive.org/downloads/download.php?fname=./VUT/M-TIT/8.%20semester/MCSI/cislicove_filtry_s.pdf. Skriptum. VUT Brno.
[22] Eulerova metoda. In: Http://www-troja.fjfi.cvut.cz/ [online]. 2000 [cit. 2014-01-17]. Dostupné z: http://www-troja.fjfi.cvut.cz/~limpouch/numet/ode/node3.html
[23] Numerické-metody. In: Www.slu.cz [online]. 2000 [cit. 2014-01-17]. Dostupné z: http://www.slu.cz/math/cz/knihovna/ucebni-texty/Numericke-metody/Numericke-metody.pdf
[24] FAT on SD Card. 2013.
[25] Přehled vývoje architektury a koncepcí CPU (RISC/CISC). In: Www.edux.feld.cvut.cz [online]. 2013 [cit. 2014-01-17]. Dostupné z: https://edux.feld.cvut.cz/courses/A0B36APO/lectures/12/start
[26] STMICROELECTRONICS. Www.st.com [online]. 2013 [cit. 2014-01-27]. Dostupné z: http://www.st.com/web/en/support/support.html
[27] MAT-File Format. In: Http://www.mathworks.com/ [online]. 2014 [cit. 2014-05-16]. Dostupné z:http://www.mathworks.com/help/pdf_doc/matlab/matfile_format.pdf
[28] EFM32G890 DATASHEET. In: Www.silabs.com [online]. 2009 [cit. 2014-05-16]. Dostupné z:http://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32G890.pdf