VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE ZALOŽENÉ NA AKCELERÁTORU GRAVITAČNÍHO ZRYCHLENÍ

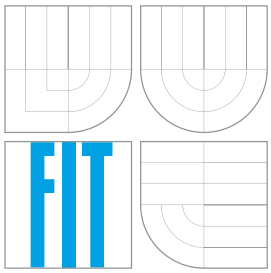
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB HORNÍK

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE ZALOŽENÉ NA AKCELERÁTORU GRAVITAČNÍHO ZRYCHLENÍ

GRAVITY ACCELEROMETER BASED APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB HORNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOSEF STRNADEL, Ph.D.

BRNO 2009

Abstrakt

Práce se zabývá možnostmi využití akcelerometru v praxi. Cílem práce bylo vybrat několik typických aplikací založených na akcelerátorech gravitačního zrychlení a realizovat je. Akcelerometr byl k dispozici na vývojovém kitu Freescale MC1321x. Výsledná aplikace je implementována v jazyce C++ spolu s knihovnou wxWidgets pro vytvoření GUI.

Abstract

Bachelor's thesis deals with the practical use of accelerometer. The goal of this work was to select a few typical applications based on accelerometer and to implement it. Accelerometer was available on the Freescale development kit MC1321x. The resulting application is implemented in C++ with wxWidgets library for creating GUI.

Klíčová slova

akcelerometr, MC1321x, MCU, mikrokontrolér, MMA7260Q, flash, wxwidgets, COM port

Keywords

accelerometer, MC1321x, MCU, microcontroller, MMA7260Q, flash, wxwidgets, COM port

Citace

Jakub Horník: Aplikace založené na akcelerátoru gravitačního zrychlení, bakalářská práce, Brno, FIT VUT v Brně, 2009

Aplikace založené na akcelerátoru gravitačního zrychlení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D.

.....
Jakub Horník
16. května 2009

Poděkování

Rád bych poděkoval Ing. Josefu Strnadelovi, Ph.D. za poskytnuté konzultace a rady, a za zapůjčení vývojového kitu Freescale MC1321x.

Obsah

1 Úvod **3**

2 Akcelerometr **4**
2.1 Konstrukce akcelerometru . . . 4
2.1.1 Kapacitní akcelerometry . . . 4
2.1.2 Piezoelektrické akcelerometry . . . 5
2.1.3 Tepelné akcelerometry . . . 6
2.2 Parametry akcelerometru . . . 6
2.3 Použití akcelerometrů . . . 7
2.3.1 Pokročilé využití akcelerometrů - inerciální navigace . . . 8

3 Vývojový kit **9**
3.1 Popis kitu . . . 9
3.1.1 Akcelerometr MMA7260Q . . . 9
3.1.2 MC13213 . . . 10
3.1.3 Modul 1321x-Network Coordinator Board . . . 10
3.1.4 Modul 1321x-Sensor Reference Board . . . 11
3.2 Programování kitu . . . 13
3.3 Demonstrační aplikace akcelerometru . . . 14
3.3.1 Princip aplikace . . . 14

4 Realizace aplikace **16**
4.1 Řídící program kitu . . . 16
4.1.1 Ukládání dat do trvalé paměti . . . 16
4.1.2 Ovládání modulu . . . 17
4.1.3 Princip programu . . . 19
4.2 PC aplikace . . . 20
4.2.1 Hlavní okno . . . 20
4.2.2 Třída ComPort . . . 23
4.2.3 Logovací okno . . . 24
4.2.4 Raw data . . . 25
4.2.5 Krokoměr . . . 26
4.2.6 Měření náklonu . . . 29
4.2.7 Měření okamžité rychlosti . . . 33

5 Závěr **37**

A Obsah CD **40**

B Návod pro překlad projektu **41**
B.1 Řídící program modulu SRB 41
B.2 Aplikace Akcelerometr 41

Kapitola 1

Úvod

Má bakalářská práce, jak již z názvu vyplývá, se zabývá využitím obvodu akcelerometru. Akcelerometry jsou důležitými součástmi v mnoha technických odvětvích. Někde si jejich přítomnost ani neuvědomujeme, jinde je jejich použití, např. v důsledku medializace, známé.

Díky akcelerometrům je cestování, a případná kolize automobilů bezpečnější, jelikož zde slouží jako senzor, spouštějící airbag. Jedná se o nejběžnější využití akcelerometru v praxi. Jsou nepostradatelnou součástí avioniky[1] každého moderního letadla, ať už malých jednomístných strojů, tak velkých dopravních letadel. V posledních letech se stále častěji nasazuje i ve spotřební elektronice. V mobilních telefonech, PDA zařízeních, kde slouží např. pro natáčení displeje podle pozice v telefonu nebo herní ovladače pro videohry. Tyto příklady však nejsou jediné možné využití akcelerometru. Druhá kapitola se proto zabývá principy akcelerometru a možnosti jeho využití.

Cílem bakalářské práce je využít akcelerometr, obsažený ve vývojovém kitu Freescale MC1321x, který popisuje třetí kapitola.

Návrhem a realizací se zabývá čtvrtá kapitola. Aplikaci lze rozdělit na část řídící pro mikrokontrolér a programovou, běžící na PC, která zobrazuje naměřená data z akcelerometru.

V závěru jsou shrnuty poznatky, získané během vypracování bakalářské práce, a jsou nastíněny další možnosti pokračování.

[1] Avionika (slovo vzniklo z francouzského avionyka) je souhrný název pro vybavení letadel elektrickými a elektronickými přístroji. Jako třeba autopilot, navigace nebo palubní počítač.

Kapitola 2

Akcelerometr

Akcelerometr je zařízení, které je schopno měřit zrychlení jak dynamické (síla působící na snímač v pohybu), tak gravitační (působení gravitace). Princip spočívá v měření rozdílu mezi kinematickým zrychlením, měřeným uvnitř inerciálního[1] prostoru (akcelerometru) a gravitačním zrychlením. Dle principu ekvivalence [15], formulovaném Albertem Einsteinem, jsou tyto hodnoty v klidovém stavu totožné. Rozdílem získáme vektor mající směr a hodnotu zrychlení působící na čidlo akcelerometru.[5, 8, 13]

2.1 Konstrukce akcelerometru

V dnešní době je pro výrobu akcelerometrů velmi používána technologie MEMS (Micro Electro-Mechanical Systems) [14]. Ta umožňuje konstrukci elektrických a mechanických struktur o velikosti μm, které se spolu vyskytují na ploše jediného čipu. Akcelerometry založené na této technologii jsou velmi kompaktní, mechanicky odolné a vynikají spotřebou energie. Mezi další typy akcelerometrů, které již nevyužívají technologie MEMS, patří např. akcelerometry se seismickou hmotou. Existují také elektromechanické akcelerometry, ale vyznačují se vysokou cenou a mají navíc příliš velké rozměry a jsou vhodné pouze pro průmysl.

Vyrábějí se akcelerometry, umožňující měřit zrychlení v jedné ose, i komplexní obvody, měřící zrychlení ve všech třech osách.

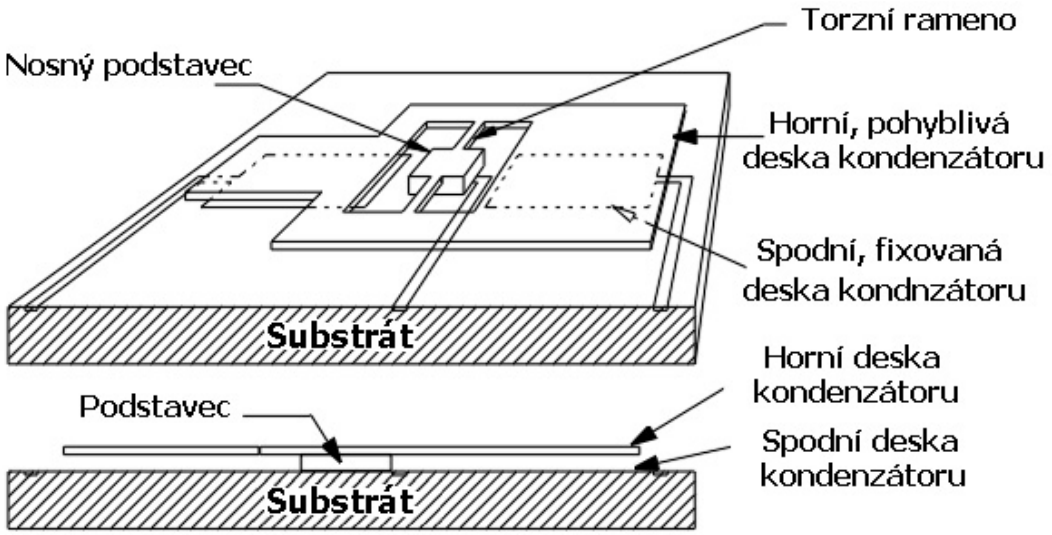
Akcelerometry využívající technologii MEMS jsou většinou založené na kapacitním principu. Mezi akcelerometry se seismickou hmotou patří piezoelektrické. Existují i speciální tepelné akcelerometry.[8]

2.1.1 Kapacitní akcelerometry

Dnes nejpoužívanější technologie používána pro konstrukci akcelerometrů. Je založena na principu proměnného deskového kondenzátoru. Jednu elektrodu tvoří deska, která se skládá ze dvou nestejně velkých křídel a je uprostřed pomocí torzních ramen ukotvena k podstavci. Druhá elektroda kondenzátoru je na tomto podstavci fixně umístěna. Za normálních okolností jsou obě křídla desky v jedné rovině a systém je vyvážen. To se však mění, pokud na systém začne působit zrychlení kolmé k rovině křídel. Vlivem asymetrie začne na širší křídlo působit větší sila, než na druhé a v důsledku se deska nakloní podle osy torzních

[1] Jako inerciální se ve fyzice označuje taková soustava, v níž platí 1. Newtonův pohybový zákon (zákon setrvačnosti)

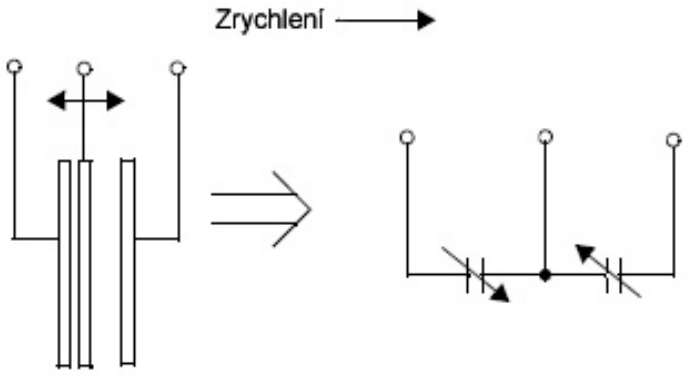
ramen ve směru působení této síly. Dojde ke změně šířky dielektrika mezi elektrodami kondenzátoru a tím i jeho kapacity.[10] Ukázka principu kapacitního MEMS akcelerometru je na obrázku 2.1 jeho fyzikálního modelu na obrázku 2.2.



Obrázek 2.1: Uspořádání mechanických prvků kapacitního akcelerometru[10]

Vyznačují se nízkou cenou a velmi dobrou přesností. Nevýhodou je omezená odolnost proti nárazům (stovky g).

Na tomto principu je postaven i akcelerometr, který obsahuje čip MMA7260Q, použitý v kitu MC1321x.



Obrázek 2.2: Fyzikální model a jeho ekvivalentní obvod kapacitního akcelerometru

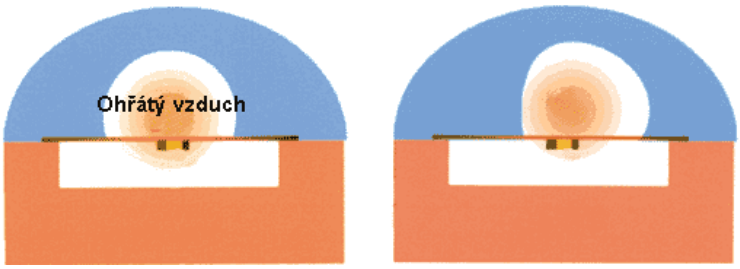
2.1.2 Piezoelektrické akcelerometry

Síla, vznikající zrychlením, působí na hmotu snímače, která stlačuje piezoelektrický krystal. Ten generuje elektrický náboj úměrný stlačení. Protože je hmota snímače konstantní, elektrický náboj je úměrný zrychlení působícím na snímač. Tento náboj je pak konvertován na napětí nebo proud podle typu a využití akcelerometru.

Tyto snímače nemohou být použity pro měření konstantního zrychlení, protože nedokážou měřit frekvence nižší než 0.1 Hz. Piezoelektrické akcelerometry patří mezi senzory výrobně jednodušší konstrukce. Jejich nevýhodou jsou větší rozměry.

2.1.3 Tepelné akcelerometry

Speciální typy akcelerometrů, které pro měření zrychlení nevyužívají pohyblivých části. Uvnitř snímače je kapsle, vyplněná plynem, v jejíž středu je odporový materiál. Tím protéká proud a zahřívá plyn, který ve svém okolí vytváří teplotní gradient. Na povrchu kapsle jsou vhodně umístěná teplotní čidla (vždy dvojice umístěná proti sobě, tvořící jednu osu snímače). Pokud na akcelerometr nepůsobí zrychlení, zaznamenávají čidla stejnou teplotu. Vlivem zrychlení dojde k naměření rozdílných teplot mezi čidly. Rozdíl této teploty je proto úměrný zrychlení působící na snímač. Ukázka funkce tepelného akcelerometru je na obrázku 2.3. Tyto akcelerometry vynikají svou odolností vůči nárazům (až 50.000g).



Obrázek 2.3: Princip tepelného akcelerometru. Důsledkem zrychlení se ohřátý vzduch pohybuje ve stejném směru jako působící síla[9]

2.2 Parametry akcelerometru

Jako každé zařízení mají i akcelerometry charakteristické vlastnosti a parametry, které vymezují oblast použitelnosti. Mezi nejdůležitější parametry, na které je potřeba brát zřetel, při výběru vhodného akcelerometru jsou:[5]

- **Dynamický rozsah**
  Udává maximální amplitudu, kterou lze změřit (kterou lze na snímač působit), než se snímač poškodí. Je uváděn v násobku g.
- **Frekvenční odezva**
  Odezva mechanické soustavy uvnitř senzoru na skokovou změnu zrychlení. Je to frekvenční rozsah, v němž výstupní hodnota signálu akcelerometru má výrobcem stanovenou dovolenou odchylku.
- **Horní frekvenční limit**
  Frekvence, kdy výstupní signál překročí výrobcem stanovenou dovolenou odchylku. Souvisí s mechanickou rezonancí daného snímače.
- **Dolní prahová frekvence**
  Frekvence, při níž výstupní signál začíná klesat nebo jeho přesnost překračuje dovo-

lenou mez. Není to zcela nulový signál, avšak citlivost velmi rychle s nižší frekvencí klesá.

- **Rezonanční frekvence**
  Je to frekvence, při níž snímač rezonuje. Frekvenční měření se snímači zrychlení by nemělo překročit tuto hodnotu.
- **Mezní hodnota napětí**
  Kritická velikost napájecího napětí, které ještě senzor nezničí.
- **Citlivost**
  Je to výstupní napětí snímače při měření určitého zrychlení vyjádřené v g. Uvádí se v mV/g.
- **Teplotní vliv (citlivost na teplotu)**
  Výstupní napětí na stupeň Celsia měřené teploty. Snímače jsou teplotně kompenzovány s cílem udržení změn výstupního signálu v daných limitech v daném rozsahu teploty.
- **Teplotní rozsah**
  Rozsah, při němž je zaručena správná funkčnost akcelerometru. Typický rozsah je -50 až 120 stupňů Celsia.

2.3 Použití akcelerometrů

Jak již bylo zmíněno v úvodu, mají akcelerometry široké uplatnění v průmyslu a v poslední době i ve spotřební elektrotechnice. Zde je stručný výčet, kde se lze s akcelerometry setkat.

- Měření zrychlení v automobilech, senzor airbagu, apod.
- Dynamické brzdové systémy v automobilech
- Inerciální navigační přístroje (viz podkapitola 2.3.1)
- Měření vibrací, seismické aktivity
- Navigace robotických ramen v průmyslu
- Navigace všesměrového robota
- Měření náklonu
- Stabilizace obrazu ve videokamerách
- Detekce pádu HDD
- Mobilní telefony a PDA–přetáčení obrazu, scrolování textu náklonem
- Herní ovladače (např. Nintendo Wii)
- Krokoměry
- V medicíně (např. AED defibrilátor)

S akcelerometrem jsme se mohli setkat i na FIT v rámci soutěže autíček řízených mikrokontrolérem, kde jeho důležitým úkolem bylo určování směru jízdy, a tedy umožnit efektivně měnit rychlost.

2.3.1 Pokročilé využití akcelerometrů - inerciální navigace

Inerciální navigační systém je naprosto autonomní systém, který je určen k navigačnímu vedení letadla nebo jiného pohybujícího se prostředku (lodě, kosmické rakety, ponorky, řízené střely).

Inerciální navigační systém se skládá z měřící jednotky obsahující akcelerometry a gyroskopy a z navigačního počítače, který vyhodnocuje data z měřících zařízení. Na rozdíl od všech ostatních navigačních systémů, je inerciální navigace zcela soběstačná a nezávislá na okolním prostředí, tj. systém nepotřebuje vnější objekty jako například družice a je odolný vůči vnějším vlivům jako je počasí, magnetické poruchy, elektronické rušení a zkreslení signálu.

Principem je počítání aktuální polohy a rychlosti pohybujícího se objektu od zadaného počátečního bodu, na základě údajů zrychlení získaných z akcelerometrů v jednotlivých osách pohybu.[12]

Kapitola 3

Vývojový kit

3.1 Popis kitu

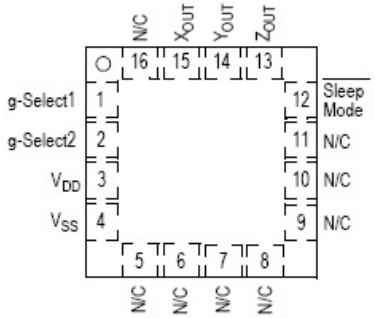
Vývojový kit 1321x[1] patří do druhé generace ZigBee platformy firmy Freescale. Sestává ze dvou rozdílných referenčních modulů:

- 1321x-Network Coordinator Board (NCB)
- 1321x-Sensor Reference Board (SRB)

Moduly mezi sebou komunikují pomocí nízkopříkonové radiové RF komunikace v pásmu 2.4GHz. Pro připojení k PC slouží USB port. Oba moduly lze napájet pomocí dvou AA baterií.

3.1.1 Akcelerometr MMA7260Q

Tříosý kapacitní akcelerometr pro měření menších zrychlení, který je součástí SRB modulu. Pracuje ve 4 úrovních rozlišení, umožňujících měřit zrychlení s rozsahem 1,5g , 2g, 4g a 6g (viz tabulka 3.1). Vyznačuje se nízkým proudovým odběrem 500μA. V režimu spánku dosahuje čip odběru 3μA. Rozmístění pinů je znázorněno na obrázku 3.1.



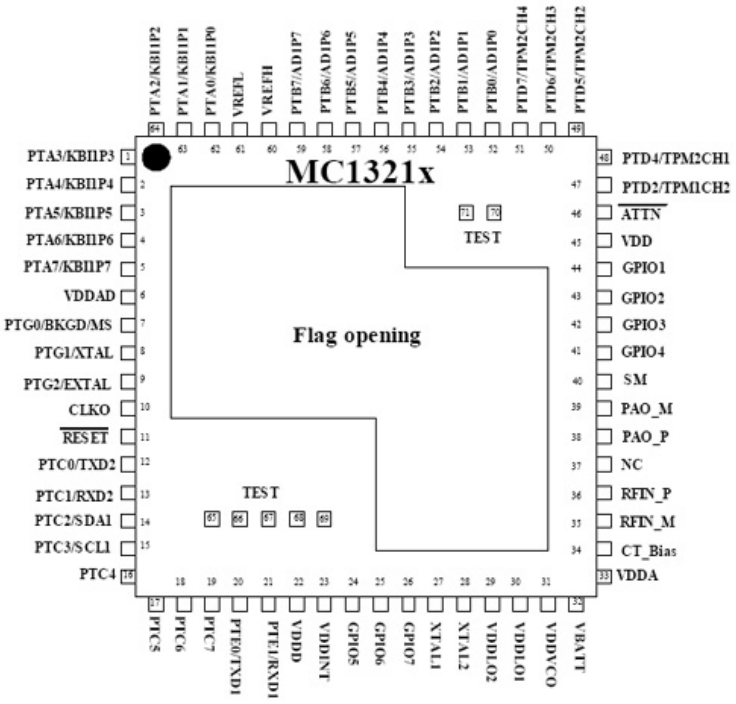
Obrázek 3.1: Rozložení pinů na čipu MMA7260Q (pohled shora)[3]

Akcelerace [g]	Citlivost [mV / g]	Frekvenční rozsah [Hz]	Zpoždění [ms]
1,5	800	350 / 150	1
2	600	350 / 150	1
4	300	350 / 150	1
6	200	350 / 150	1

Tabulka 3.1: Režimy akcelerometru MMA7260Q

3.1.2 MC13213

Čip MC13213 SiP v sobě integruje mikrokontrolér MC9S08GT spolu s transceiverem MC1320x. Mikrokontrolér vychází z rodiny HCS08 a nabízí 60KB flash paměti a 4KB paměti RAM. RF transceiver MC1320x je kompatibilní s IEEE 802.15.4 a pracuje v pásmu 2,4 GHz. Zahrnuje v sobě 1mW zesilovač, VCO oscilátor a kódovací / dekódovací obvody. Rozložení pinů je znázorněno na obrázku 3.2.



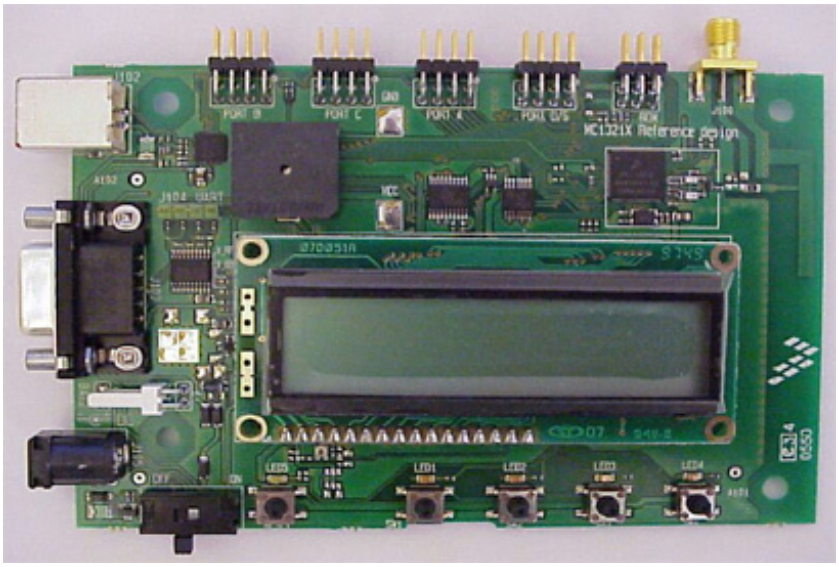
Obrázek 3.2: Mikrokontrolér MC13213 (pohled shora)[2]

3.1.3 Modul 1321x-Network Coordinator Board

Modul 1321x-NCB (obrázek 3.3) je založen na čipu MC13213 SiP. Pro propojení s PC nabízí rozhraní USB a RS232.

Součástí modulu jsou:[1]

- Programovatelný dvouřádkový LCD displej
- 4 spínače (na přípravku označené jako SW1, SW2, SW3 a SW4)



Obrázek 3.3: Modul 1321x-NCB[1]

- 4 LED diody (označené LED1, LED2, LED3 a LED4)
- Tlačítko Reset
- Vypínač
- Konektory USB a RS232
- Napájecí konektor
- 2x3 pinový konektor BDM (Background Debug Module) pro programování a debugování přes USB Multilink modul
- 4x 8 pinový konektor pro připojení externích periferií (vyvedené porty MCU)
- Vestavěná anténa
- SMA RF konektor pro připojení externí antény

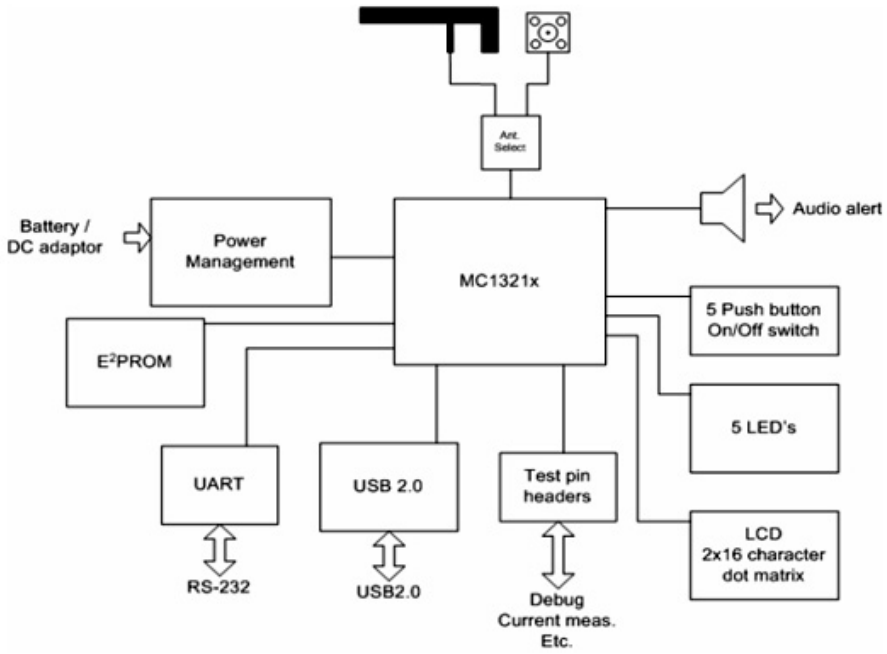
Blokové schéma modulu 1321x-NCB je na obrázku 3.4.

3.1.4 Modul 1321x-Sensor Reference Board

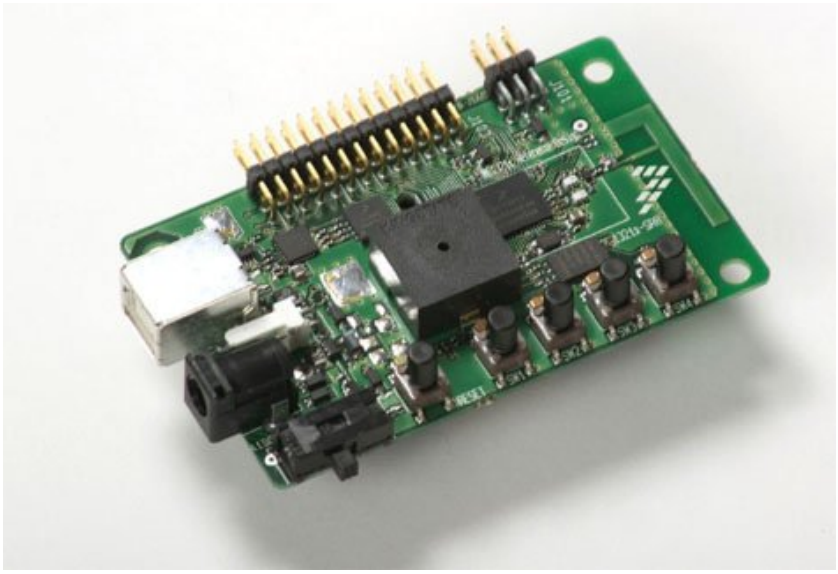
Modul 1321x-SRB (obrázek 3.5) je založen na shodné platformě jako NCB ale neobsahuje LCD displej a převodník RS232. Navíc má akcelerometr MMA7260Q.

Součástí modulu jsou:[1]

- 4 spínače (na přípravku označené jako SW1, SW2, SW3 a SW4)
- 4 LED diody (označené LED1, LED2, LED3 a LED4)
- Tlačítko Reset



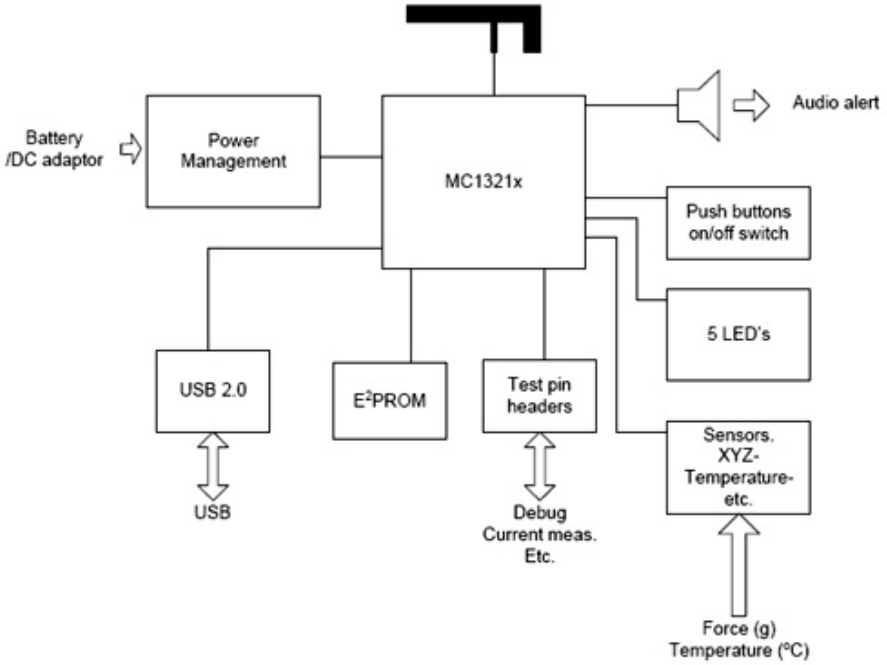
Obrázek 3.4: Blokové schéma modulu 1321x-NCB[1]



Obrázek 3.5: Modul 1321x-SRB[1]

- Vypínač
- Konektor USB
- Napájecí konektor
- 2x3 pinový konektor BDM (Background Debug Module) pro programování a debugování přes USB Multilink modul
- 26 pinový konektor pro přístup ke specifickým pinům MCU a RF
- Vestavěná anténa
- SMA RF konektor pro připojení externí antény

Blokové schéma modulu 1321x-SRB je na obrázku 3.6.



Obrázek 3.6: Blokové schéma modulu 1321x-NCB[1]

3.2 Programování kitu

Pro programování modulů kitu slouží zařízení P&E USB HCS08/HCS12 MULTILINK. Kromě programování umožňuje i ladit program skrze BDM rozhraní. Programátor je zobrazen na obrázku 3.7.

Popis programátoru Multilink:[6]

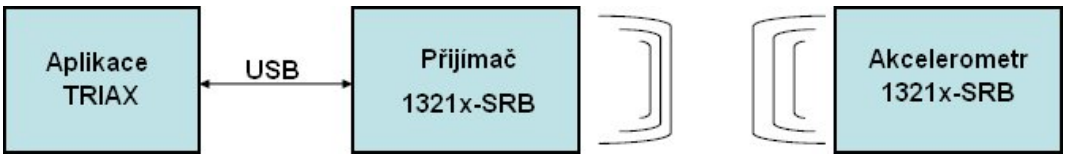
- Připojení přes USB rozhraní z počítače do Multilink zařízení umožňuje rychlé a snadné programování a ladění.

Obrázek 3.7: Programátor P&E USB HCS08/HCS12 MULTILINK

- Napájeno z USB rozhraní - není potřeba žádný samostatný napájecí zdroj.
- Výstupní napětí: 1.6V-5.25V
- Výstupní rekvence: 16Khz-50Mhz
- Kompatibilní s HCS08, RS08, HCS12, HC12 a ColdFire V1
- Automatická detekce frekvence pro mikrokontroléry rodiny HCS08 a HCS012

3.3 Demonstrační aplikace akcelerometru

Ke kitu jsou dodávány testovací aplikace založené na Simple MAC (SMAC). Jich je celá řada, ale pro tuto práci je nejdůležitější demonstrační aplikace využívající akcelerometr. Využívají se oba moduly SRB, kdy jeden je pomocí USB připojen k PC a slouží jako přijímač. Druhý modul slouží jako akcelerometr a údaje, pomocí bezdrátového přenosu, přenáší prvnímu. Na PC je spuštěna ukázková aplikace TRIAX, která naměřené údaje zobrazuje (viz obrázek 3.8).

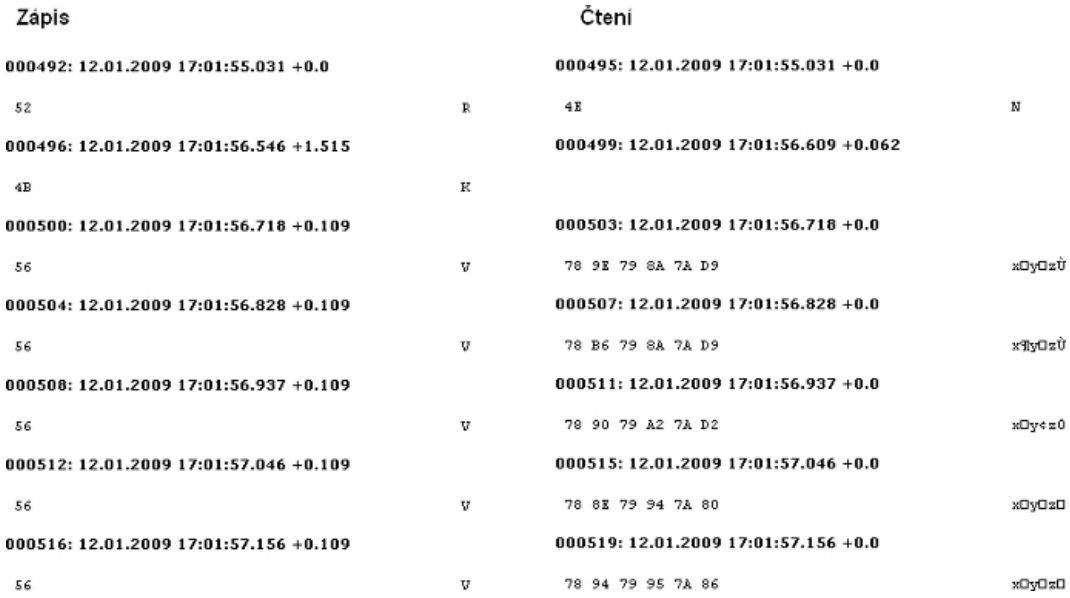


Obrázek 3.8: Schéma demonstrační aplikace akcelerometru

3.3.1 Princip aplikace

Po připojení modulu k PC se namapuje na volný COM port, pomocí kterého lze s modulem komunikovat. Po spuštění demoaplikace TRIAX se tento port manuálně zvolí. Aplikace poté

pomocí dohodnuté syntaxe ověří přítomnost SRB modulu (zašle bajt 52h), který odpoví 4Eh. Následuje zaslání bajtu 4Bh, na který však ze strany SRB není implementována žádná reakce. Aplikace se poté periodicky dotazuje přijímacího modulu na aktuální stav akcelerometru (zasíláním bajtu 56h). Odpověď je ve formátu x*y*z*, kde * je 8b číslo udávající velikost akceleračního zrychlení v daném směru (viz obrázek 3.9). Přijímač si pamatuje poslední hodnotu, kterou poslal modul akcelerometru pomocí bezdrátového přenosu.



Obrázek 3.9: Ukázka komunikace aplikace Triax s modulem SRB

Kapitola 4

Realizace aplikace

Úkolem je vytvořit jednoduchou aplikaci, která by demonstrovala několik typických aplikací, založených na akcelerátorech gravitačního zrychlení (např. měření sklonu nakloněné roviny, krokoměr). Akcelerometr by měl mít možnost ukládat naměřená data offline, bez připojení k PC, a zobrazit je až později.

Tato kapitola popisuje návrh a implementaci aplikace. Tu lze rozdělit na řídící programy modulů a na aplikaci, běžící na počítači, která přijíma data od akcelerometru a dle zvoleného módu je patřičně vizualizuje. Schéma komunikace mezi PC aplikací a modulem akcelerometru je shodné, jak bylo popsáno v kapitole 3.3.

4.1 Řídící program kitu

Jako základ pro řídící program byl použit demonstrační program dodávaný ke kitu, a byl upraven pro potřeby aplikace. Nepotřebné funkce byly odstraněny a naopak přidány nové. Program pro bezdrátový přenos dat mezi moduly byl využit beze změny. Je použit režim akcelerometru 1,5g.

4.1.1 Ukládání dat do trvalé paměti

Jedním z požadavků při návrhu aplikace byla možnost ukládat naměřená data do trvalé paměti a zobrazit je až po připojení k počítači. Zde se nabízely dvě možnosti:

- vyměnitelná SD karta
- vestavěná FLASH paměť pro program

Ukládání na SD kartu

Využití SD karty má výhodu vyměnitelnosti, a tedy teoreticky neomezené kapacity. Jelikož SD karta podporuje SPI komunikaci, je práce s ní velice jednoduchá. Zde se však narazilo na problém ze strany mikrokontroléru, který využívá SPI pro komunikaci s transceiverem. Simulovat SPI přes zbývající V/V rozhraní nebylo taktéž možno, jelikož nebyl k dispozici dostatečný počet vstupů/výstupů.

Ukládání na pamět FLASH

Mikrokontrolér obsahuje 64kB vestavěné FLASH paměti pro program a konstanty. Do této paměti lze zapisovat i za běhu programu a může být použita pro ukládání naměřených dat z akcelerometru. Samotný program zabírá na FLASH pamětí asi 20kB dat, a pro demonstrační účely zbývá dostatečná kapacita.

Zvolená varianta

Vzhledem k již zníměným problémům u SD karty byla zvolena varianta druhá. Data se do FLASH ukládají za sebou v posloupnosti data X, data Y, data Z. Aby se omezil počet zápisů do FLASH, neukládá se do ni informace o prvním volném bajtu. Při inicializaci se toto místo sekvenčně vyhledá od počáteční pozice pro data a vytvoří se ukazatel v paměti RAM.

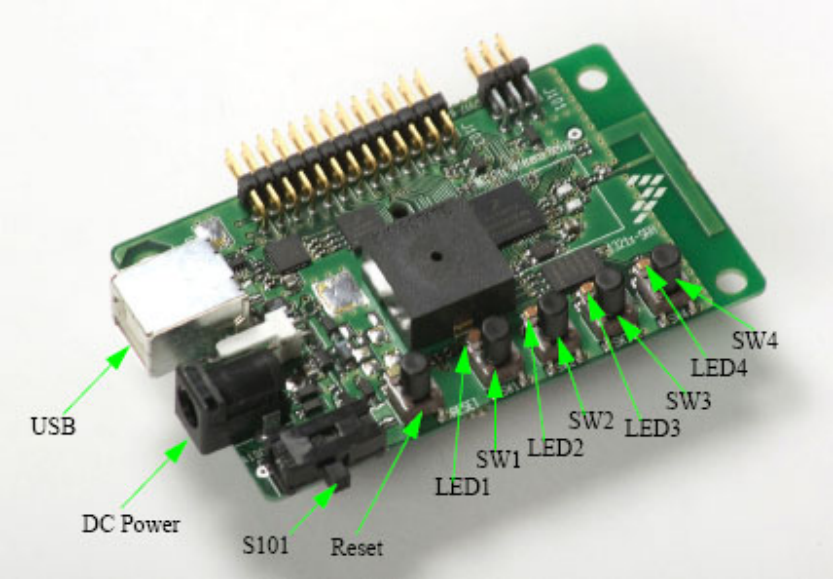
```
void findVectorStart(void)
{
    // cyklus dokud nenarazím na první volný bajt
    while( *((UINT8*)u16flashVector) != 0xFF) {
       u16flashVector  += 3;               // posun o 3 bajty
       if(u16flashVector > VECTOREND) {    // konec vymezené paměti dat
          u8vectorFull = 1;
          break;
       }
    }
}
```

Pro zápis slouží funkce `Program_Byte(Address, data)`, které po přepisu do C vypadá následovně:

```
if (FSTAT&0x10){                    // FACCERR je nastaveno
  FSTAT = FSTAT | 0x10;             // zápis 1 na FACCERR
}
// zápis
(*((volatile unsigned char *)(Address))) = data;
FSTAT = 0x80;                       // nastavení FCBEF na 1
_asm NOP;                           // čekat 4 cykly
...
// kontrola jestli FACCERR nebo FVIOL je nastaveno
if (FSTAT&0x30){
  return 0xFF;                      // jestli ano, chyba
}
while ((FSTAT&0x40)==0){            // čekání na konec operace
}
```

4.1.2 Ovládání modulu

Kit se ovládá pomocí pětice spínačů (reset a 4 funkční tlačítka SW1 až SW4). Rozmístění ovládacích prvků je znázorněno na obrázku 4.1.



Obrázek 4.1: Popis ovládacích prvků a konektorů modulu SRB

Pomocí spínače SW1 se vybírá jeden z módu akcelerometru. Výběr je indikován rozsvícením LED v binárním kódu, odpovídajícím pořadí, a zvukovým signálem. Stavový diagram popisující tyto módy a jejich ovládání je naznačen na obrázku 4.2.

Výčet módu:

- režim přijímače (PC_RADIO_STATE)
- režim akcelerometru (XYZ_STATE)
- režim čtení dat z FLASH (XYZ_FLASH_STATE)

Režim přijímače

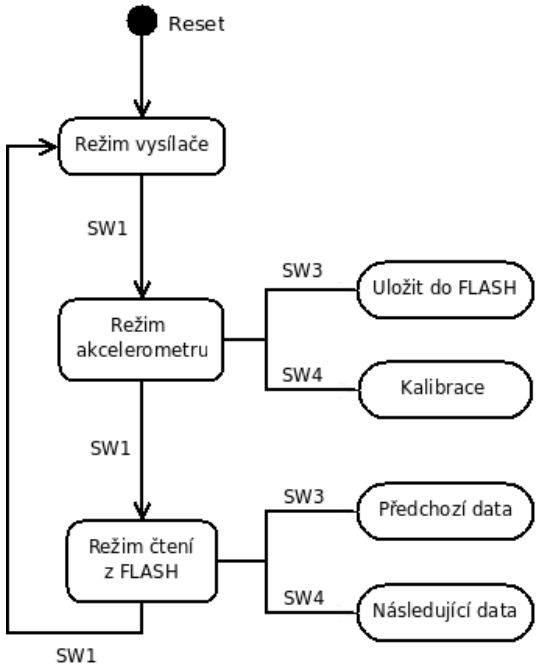
V tomto režimu modul přijímá údaje z druhého zařízení, které je v módu akcelerometru, pomocí bezdrátové sítě. Aktuální data jsou uloženy v paměti a pokud jsou vyžádána od PC aplikace, jsou poslány sériovým rozhraním.

Režim akcelerometru

Pokud dojde ke změně jedné z os akcelerometru, jsou data přeposlána, pokud je v dosahu, přijímači. Změna je indikována blikáním LED1. Stiskem tlačítka SW4 dojde ke kalibraci modulu a nastavení na počáteční hodnoty. Stiskem tlačítka SW3 dojde k uložení aktuálních hodnot os X,Y a Z do FLASH paměti.

Režim čtení dat z FLASH

V tomto režimu je možné procházet data, uložená ve FLASH paměti a přes přijímač je posílat do PC aplikace pro zobrazení. Mezi daty se přepíná stiskem tlačítka SW3 pro předchozí a SW4 pro následující data. Změna je doprovázena zvukovým signálem. Při dosažení konce nebo začátku je zvukový signál výraznější.



Obrázek 4.2: Stavový diagram módů SRB

4.1.3 Princip programu

Samotný řídící program pracuje v jedné velké smyčce, kterou lze rozdělit do dvou fází.

V první fázi se provede obslužná rutina dle zvoleného režimu činnosti. Aktuální režim modulu je v proměnné `gi8AppStatus`.

```
switch (gi8AppStatus) {
    case PC_RADIO_STATE:
        PC_Radio_App();
        break;
    case ...
}
```

Ve druhé fázi se reaguje na události (stisk tlačítka, přetečení čítače), které vznikly v tomto cyklu, nebo se nestihly detekovat a zpracovat v předchozím. Veškeré tyto události jsou zaznamenány do proměnné `gu16Events` a po provedení patřičné reakce z ní odstraněny.

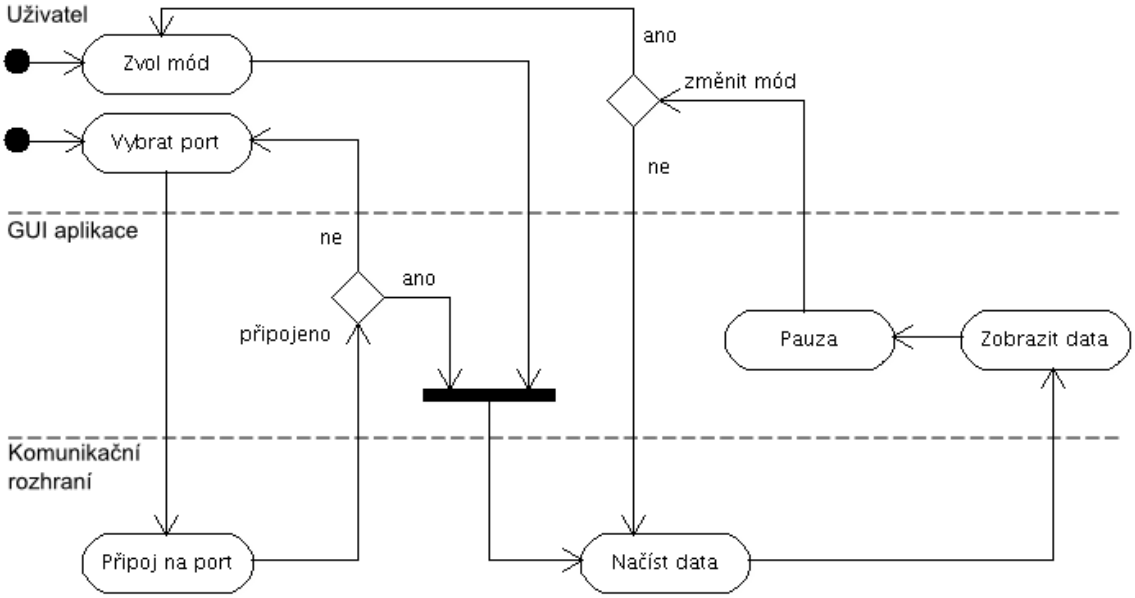
Provedení akce na konkrétní událost vypadá následovně:

```
// stisk tlačítka SW3
if ((gu16Events & KBI4_EVENT) != 0) {
    gu16Events &= ~KBI4_EVENT; // odstranění události
    // provedení akce
    if(gi8AppStatus == XYZ_FLASH_STATE) {
        vectorPrevData();
    }
    ...
}
```

4.2 PC aplikace

Pro implementaci aplikace byl zvolen jazyk C++ spolu s knihovnou wxWidgets[11] pro vytvoření grafického uživatelského rozhraní. Aplikaci lze rozdělit na rozhraní pro komunikaci s modulem akcelerometru a GUI pro zobrazení získaných dat. Komunikační rozhraní je psáno pro platformu Microsoft Windows, GUI aplikace je multiplatformní.

Na obrázku 4.3 je naznačena činnost aplikace. Uživatel si zvolí mód aplikace a vybere komunikační port. Pokud připojení proběhne úspěšně, začne se aplikace periodicky dotazovat akcelerometru o data, která se následně, dle zvoleného módu, patřičně zobrazí. V této smyčce je možné libovolně měnit zobrazený mód.



Obrázek 4.3: UML diagram aktivit popisující činnost aplikace

V následujících podkapitolách jsou popsány jednotlivé části programu.

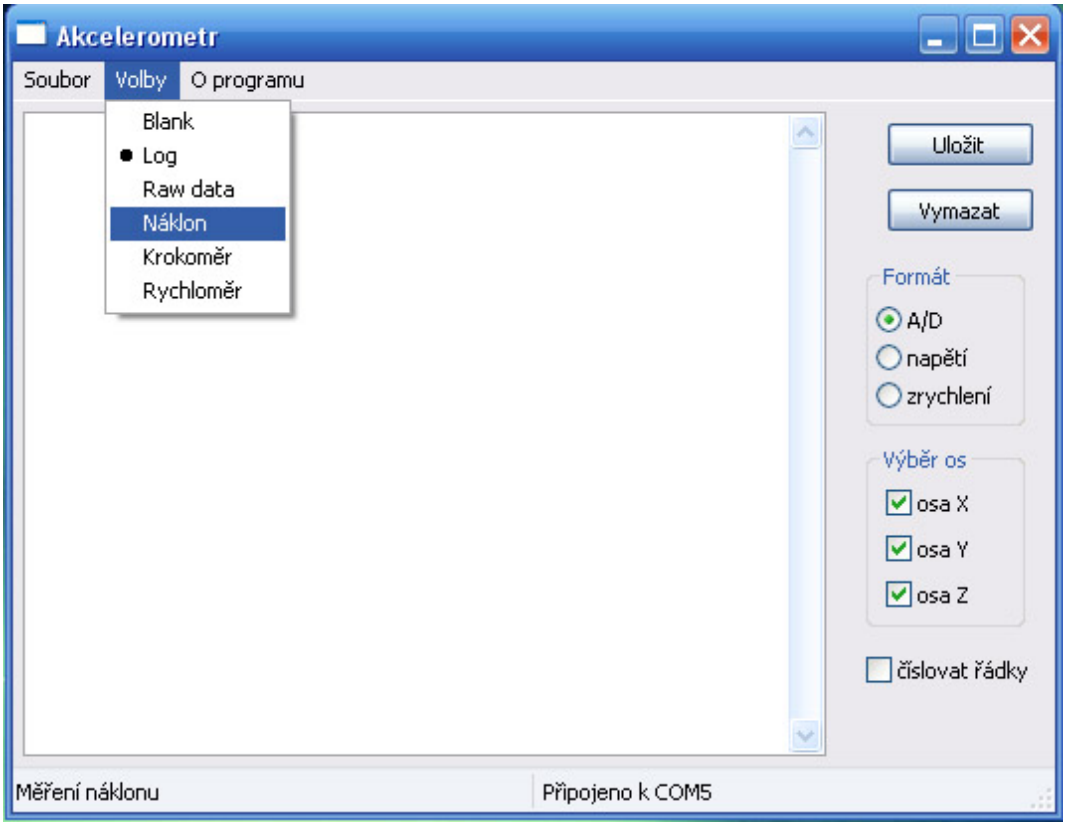
4.2.1 Hlavní okno

Hlavní okno aplikace tvoří třída `Akcelerometr`. Ta poskytuje základní prvky pro ovládání aplikace jako menu a statusbar. Pomocí menu lze přepínat mezi panely, které představují jednotlivé módy akcelerometru. Ukázka aplikace viz obrázek 4.4.

Aplikace se ovládá pomocí menu v horní části okna. Základní volby jsou:

- Soubor
- Volby
- O programu

V menu Soubor se lze pomocí položek připojit, popř. odpojit od akcelerometru nebo ukončit program.



Obrázek 4.4: Ukázka aplikace Akcelerometr

V menu Volby lze přepínat mezi těmito módy:

- Logování
- Raw data
- Krokoměr
- Měření náklonu
- Měření okamžité rychlosti

Třída `Akcelerometr` je navrhnuta a napsána tak, aby umožňovala jednoduché přidání nebo odebrání jednotlivých panelů. V hlavičkovém souboru `main.h` je obsažen výčet `ePanel`, kde jsou všechny uvedeny. V proměnné `m_panelSet` je uložen identifikátor právě zvoleného panelu.

```
enum ePanel
{
    PANELBLANK,
    PANELLOG,
    ...
};
```

Každý panel má svou položku v menu, která při výběru vyvolá událost, v níž je zavolána metoda `ChangeView(ePanel view)`. Ta slouží k přepínání mezi jednotlivými panely, kdy uvolní z paměti původní a nahradí ho zvoleným panelem. Metodu tvoří přikaz switch, který na základě identifikátoru panelu, předaného parametrem, provede jeho vytvoření a umístění do hlavního sizeru okna, čímž dojde k jeho zobrazení.

```
void Akcelerometr::ChangeView(ePanel view)
{
    // odstranění aktuálního panelu
    bMainSizer->Clear( true );

    // nastavení aktualního panelu dle parametru funkce
    switch(view)
    {
    //prazdný panel
    case PANELBLANK:
        // vytvoření panelu
        m_panelBlank = new PanelBlank( this, GetClientSize() );
        // umístění do hlavního sizeru okna
        bMainSizer->Add( m_panelBlank, 1, wxEXPAND | wxALL, 0 );
        // nastavení aktualního panelu
        m_panelSet = PANELBLANK;
        break;

    case ...
    }
    ...
}
```

Pokud je zvolen jeden z panelů (vyjma PANELBLANK – prázdné okno), a je úspěšně připojen akcelerometr, dochází v intervalu 100ms (nastavuje se v hlavičkovém souboru) k volání metody `AccelData` třídy `ComPort` pro získání aktuálních dat akcelerometru. Poté je volána metoda `RefreshData()`, která data zašle aktuálně zvolenému panelu. Výběr probíha příkazem switch na základě proměnné `m_panelSet`, kde je uložen identifikátor aktuálního panelu.

```
void Akcelerometr::RefreshData(void)
{
    // výběr podle aktuálního panelu
    switch(m_panelSet)
    {
    case PANELLOG:
        // došlo ke změně dat
        if( databuff != datalast) m_log->RefreshData(databuff);
        break;
    ...
    }
}
```

4.2.2 Třída ComPort

Pro komunikaci s kitem slouží třída `ComPort`, která poskytuje rozhraní potřebné pro komunikaci s akcelerometrem. Využívá funkci `CreateFile` knihovny windows.h umožnující vytvořit sériové spojení přes COM port.

```
m_comHandle=CreateFile( PortName,  // textový řetězec s číslem portu
    GENERIC_READ | GENERIC_WRITE,  // čtení i zápis
    0,                             // otevřít s exklusivním přístupem
    NULL,
    OPEN_EXISTING,                 // otevřít pouze existující
    NULL,                          // nejde přijímat i vysílat zároveň
    NULL);
```

Komunikace s kitem probíha podle předem dohodnuté konvence, jak již bylo popsáno v kapitole 3.3. Po připojení se po sériovém portu zašle zpráva `R`. Pokud je připojen modul akcelerometru, odpoví zprávou `N`. Po tomto ověření se aplikace periodicky dotazuje na nová data zasláním zprávy `V`. Od modulu obdrží řeťezec s daty, která jsou převedena do struktury `sAccel`. V této struktuře se předávají data mezi všemi moduly (třídami).

```
/* Struktura pro uložení dat z akcelerometru */
struct sAccel
{
    int x, y, z;
    sAccel() : x(0), y(0), z(0) {}
    sAccel(int _x, int _y, int _z) : x(_x), y(_y), z(_z) {}
    sAccel(const sAccel& p) : x(p.x), y(p.y), z(p.z) {}

    sAccel operator=(const sAccel & p)
    {
      x = p.x; y = p.y; z = p.z;
      return p;
    }
};
```

Popis metod

- `bool OpenPort(int ComNumber)`
  Základní metoda třídy `ComPort`. Vytvoří sériovou komunikaci na COM portu, zadaném parametrem `ComNumber`. Vrací `true` při úspěšném provedení, jinak `false`.

- `bool ClosePort()`
  Uzavře sériové spojení. Vrací `true` při úspěchu.

- `int ReadData(char* buffer,int size)`
  Přečte data ze sériového portu a uloží je do zásobníku `buffer` s kapacitou `size`. Metoda vrací počet přečtených bajtů.

- `int WriteData(char* buffer,int size)`
  Zapíše `size` bajtů z bufferu na sériový port. Metoda vrací počet zapsaných bajtů.

- `int AccelInit()`
  Metoda ověří, pomocí dohodnuté syntaxe, zda komunikuje s akcelerometrem. Vrací 0 při neúspěchu nebo počet přijatých bajtů při úspěchu.
- `int AccelData(sAccel &data)`
  Přečte aktuální hodnoty zrychlení z akcelerometru a uloží je do struktury data, předané odkazem.
- `bool IsConnected()`
  Vrací `true`, pokud je připojeno, jinak `false`.

4.2.3 Logovací okno

Tato aplikace slouží pro logování hodnot z akcelerometru.

Popis metod

Třída `PanelLog`.

- `void RefreshData(const sAccel &data)`
  Jediná veřejná metoda třídy. Slouží k předání aktuálních hodnot zrychlení z akcelerometru, a zajistí zobrazení do logu.
- `void OnBtnClear( wxCommandEvent& event )`
  Metoda události volána při stisku tlačítka *Vymazat*. Odstraní data z logovacího okna.
- `void OnBtnSave( wxCommandEvent& event )`
  Metoda události volána při stisku tlačítka *Uložit*. Otevře dialog pro ukládání souboru a uloží výstupu z logu.
- `void OnChkboxLine( wxCommandEvent& event )`
  Metoda události volána při zatrhnutí políčka *číslovat řádky*.
- `void OnChkbox{X|Y|Z}( wxCommandEvent& event )`
  Metody události volané při zatrhnutí políček pro výběr zobrazovaných os.

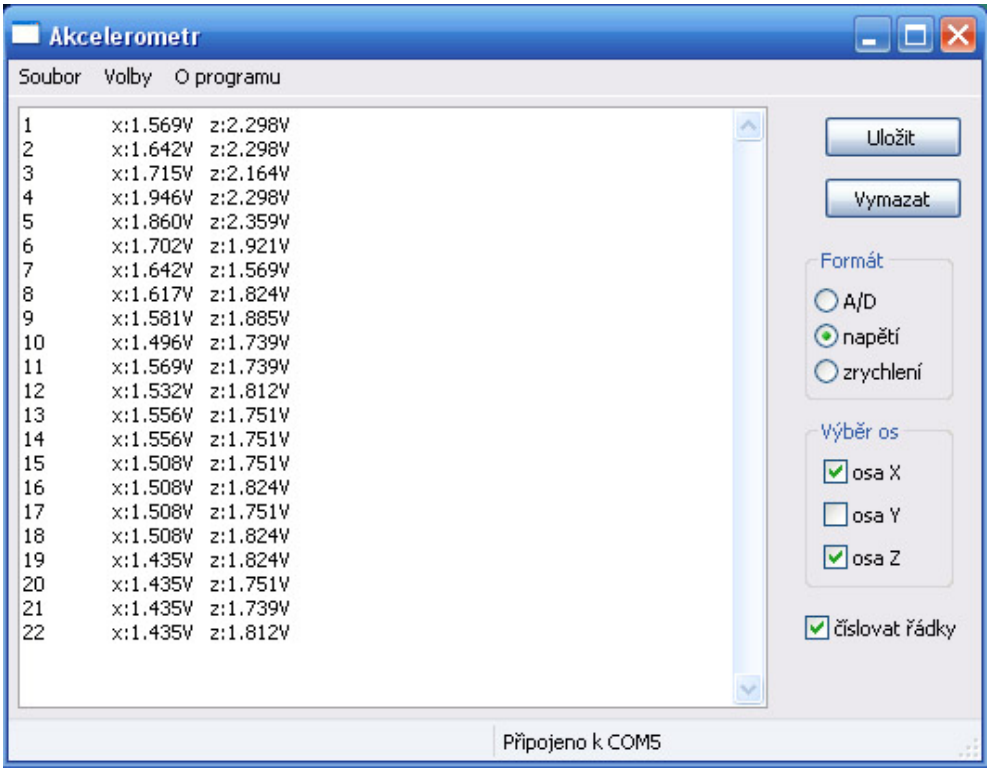
Popis aplikace

Ukázka aplikace je na obrázku 4.5. Hlavní část panelu tvoří textové pole wxTextCtrl, do kterého se ve zvoleném formátu zapisují údaje z akcelerometru. V pravé části se nacházejí ovládací prvky.

Lze zvolit, v jakém formátu se mají data zobrazovat:

- hodnota A/D převodu [0 - 255]
- napěťová úroveň ve voltech
- velikost zrychlení v g

Logovaná data je možné uložit do textového souboru nebo smazat stiskem příslušného tlačítka. Dále je možné si vybrat pouze konkrétní osu a číslovat řádky. Takový textový výstup je následně možné graficky zobrazit pomocí externích programů. Aktuální data z akcelerometru se předávají zavoláním metody `RefreshData`.



Obrázek 4.5: Ukázka módu Logovací okno

4.2.4 Raw data

Popis aplikace

Panel slouží pro zobrazení aktuálních údajů z akcelerometru ve třech základních formátech:

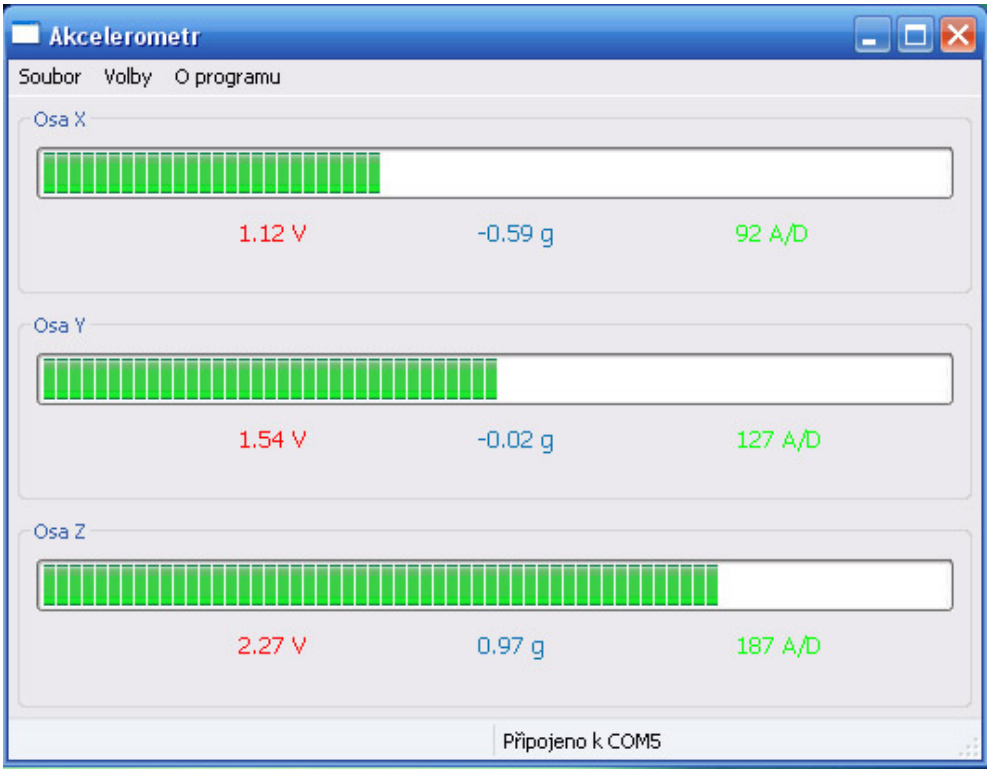
- číselný výstup A/D převodu [0-255]
- hodnota napětí ve voltech
- velikost zrychlení v dané ose v g (-1,5 až 1,5g), kde znaménko určuje směr

Pro lepší vizualizaci údajů slouží wxWidgets modul wxGauge[11], což je ukazatel průběhu používaný u instalátorů. Aktuální data z akcelerometru se předávají zavoláním metody `RefreshData`, což je, vyjma konstruktoru a destruktoru, jediná metoda této třídy. Ukázka panelu viz obrázek 4.6.

Vztahy pro převod mezi těmito třemi veličinami jsou následující:

- Převod A/D na volty dle vzorce $U = 0,01216 \cdot d$, kde d je hodnota A/D převodu a U je výsledné napětí ve voltech.
- Převod A/D na hodnotu zrychlení dle vzorce $a = 0,016387 \cdot (d - 128)$, kde d je hodnota A/D převodu a a je výsledné zrychlení v jednotce g. Pro $d = 0$ je zrychlení $0g$.

Tyto vzorce byly odvozeny dle převodních vztahů použitých u demonstrační aplikace Triax a dle [3]. Jsou k dispozici jako globální funkce `digToV(int d)` a `digToV(int d)` definované v hlavičkovém souboru `conversion.h`.



Obrázek 4.6: Ukázka módu Raw data

Popis metod

Třída `PanelGauge`.

- `void RefreshData(const sAccel &data)`
  Slouží k předání aktuálních hodnot zrychlení z akcelerometru, a zajistí jejich zobrazení.

4.2.5 Krokoměr

Popis aplikace

Aplikace pomocí údajů zrychlení umožňuje detekovat a počítat kroky při chůzi. Podle zadané délky kroku následně počítá ušlou vzdálenost a průměrnou rychlost. Ukázka aplikace viz obrázek 4.7.

Ušlá vzdálenost se vypočítá pomocí vzorce $s = poc\_k \cdot delka\_k$, kde poc_k je počet kroků a $delka\_k$ délka jednoho kroku.

Pro výpočet průměrné rychlosti slouží vzorec $v = s/t$[4], kde s je ušlá vzdálenost a t je doba chůze.

Rychlost lze pomocí radioboxu zobrazit v těchto formátech:

- m/s
- km/h
- $kroků/min$

Obrázek 4.7: Ukázka módu Krokoměr

Princip detekce kroku

Detekce kroku pracuje na jednoduchém principu a slouží pouze k demonstraci aplikace. Základem je počítání absolutní hodnoty rozdílu dvou po sobě jdoucích hodnot. Pokud je rozdíl větší, než aktuálně nastavená hodnota citlivosti, vyhodnotí se tato událost jako krok. Dalším předpokladem je, že doba mezi dvěma kroky je větší než 200ms (pouze pro demonstraci, v reálu to může být při běhu i méně), a po tento časový údaj se po detekci kroku neprovádí porovnávání. Tím dojde k potlačení nežádoucích zákmitů vzniklých po dokročení.

Pro měření je využívaná pouze osa y akcelerometru (delši strana modulu). Modul by měl být přichycen k noze tak, aby osa y byla kolmo na směru pohybu. Výše popsaná detekce kroku je implementována v metodě `RefreshData`.

```
void PanelPedom::RefreshData(const sAccel &data)
{
    if(isRunning && !delay)                                    [1]
    {
        int dy = data.y - prev.y;                              [2]
        if( abs(dy) >= sensi )                                 [3]
        {
            steps++;                                           [4]
            distance += steplen;                               [5]
            ...
            m_delay->Start(200, true);                         [6]
            delay = true;                                      [7]
        }
    }
        prev = data;                                           [8]
}
```

Jestliže je aplikace spuštěna a neběží čekací rutina od předešlé detekce `[1]`, vypočte se absolutní hodnota rozdílu dvou po sobě jdoucích hodnot `[2]`. Pokud je tato hodnota větší, než nastavená úroveň citlivosti `[3]`, je detekován krok `[4]` a jeho délka připočtena k celkové vzdálenosti `[5]`. Poté je spuštěn časovač, který po stanovené době nastaví proměnnou `delay` na `false` `[6]`. Proměnná `delay` je po tento časový okamžik nastavená jako `true` `[7]` a zamezí splnění podmínky `[1]`. Aktuální hodnota se poté stává poslední `[8]`.

Popis metod

Třída `PanelPedom`.

- `void RefreshData(const sAccel &data)`
  Slouží k předání aktuálních hodnot zrychlení z akcelerometru, a zajistí jejich vyhodnocení.
- `void UpdateTime()`
  Aktualizuje zobrazovaný čas. Hodnota je přepočítána na hodiny, minuty a sekundy a zobrazena.
- `void UpdateSteps()`
  Aktualizuje zobrazovaný počet kroků. Aktuální hodnota je uložená v proměnné `steps`
- `void UpdateSpeed()`
  Aktualizuje zobrazovanou průměrnou rychlost podle zvoleného formátu.
- `void UpdateDistance()`
  Aktualizuje zobrazovanou vzdálenost. Aktuální hodnota je uložená v proměnné `distance`.
- `void OnBtnStart(wxCommandEvent& event)`
  Metoda události volané při stisku tlačítka *Start/Stop*.
- `void OnBtnReset(wxCommandEvent& event)`
  Metoda události volané při stisku tlačítka *Reset*.

- `void OnSliderStep(wxCommandEvent& event)`
  Metoda události volaná při změně posuvníku délky kroku.
- `void OnSliderSensi(wxCommandEvent& event)`
  Metoda události volané při změně citlivosti.
- `void OnRadioMode(wxCommandEvent& event)`
  Metoda události volané při volbě formátu rychlosti.
- `void OnTimer(wxTimerEvent& event)`
  Metoda časovače volána každou jednu sekundu. Slouží pro počítání uběhlého času a aktualizaci zobrazovaných údajů.
- `void OnDelay(wxTimerEvent& event)`
  Zpoždění, kdy nedochází k porovnávání hodnot z akcelerometru.

Pokročilá implementace

Dosavadní princip detekce je pouze demonstrativní a jeho procentuální úspěšnost závisí na vhodně zvolené citlivosti. Pro lepší využitelnost této aplikace je podstatné realizovat pokročilejší algoritmus detekce kroku. Dále lze doplnit např. měřič spálených kalorií, využitelný při běhu.

Ovládání aplikace

Aplikace se ovládá pomocí voleb v sekci *Nastavení*.

Posuvníkem *Délka kroku* se nastaví vzdálenost jednoho kroku. Od této hodnoty se odvozuje ušlá vzdálenost i rychlost pohybu.

Posuvník *Citlivost* slouží pro změnu citlivosti detekce kroku. Hodnota 0% představuje nejmenší citlivost.

V části *Formát* lze zvolit zobrazovaný formát rychlosti.

Stiskem tlačítka *Start*, a za podmínky, že je připojen akcelerometr, dojde ke spuštení aplikace. Tlačítkem *Stop* se zastaví.

Tlačítko *Reset* slouží pro vynulování veškerých dat i časomíry.

4.2.6 Měření náklonu

Popis aplikace

Tento panel je plně vytvořen pomocí kreslících metod wxWidgets. Každá osa má svůj vlastní ukazatel náklonu. Ten se skládá ze čtyř částí:

- kružnice s popisy úhlů `CreateBitmap`
- šipka ukazující na hodnotu náklonu `DrawArrow`
- textový popisek velikosti náklonu `DrawValue`
- rámec ohraničující oblast ukazatele `DrawRect`

U této aplikace lze využít možnosti modulu akcelerometru, ukládat naměřené hodnoty náklonu do vestavěné FLASH paměti, a po připojení k PC si mezi nimi přepínat (viz kapitola 4.1).

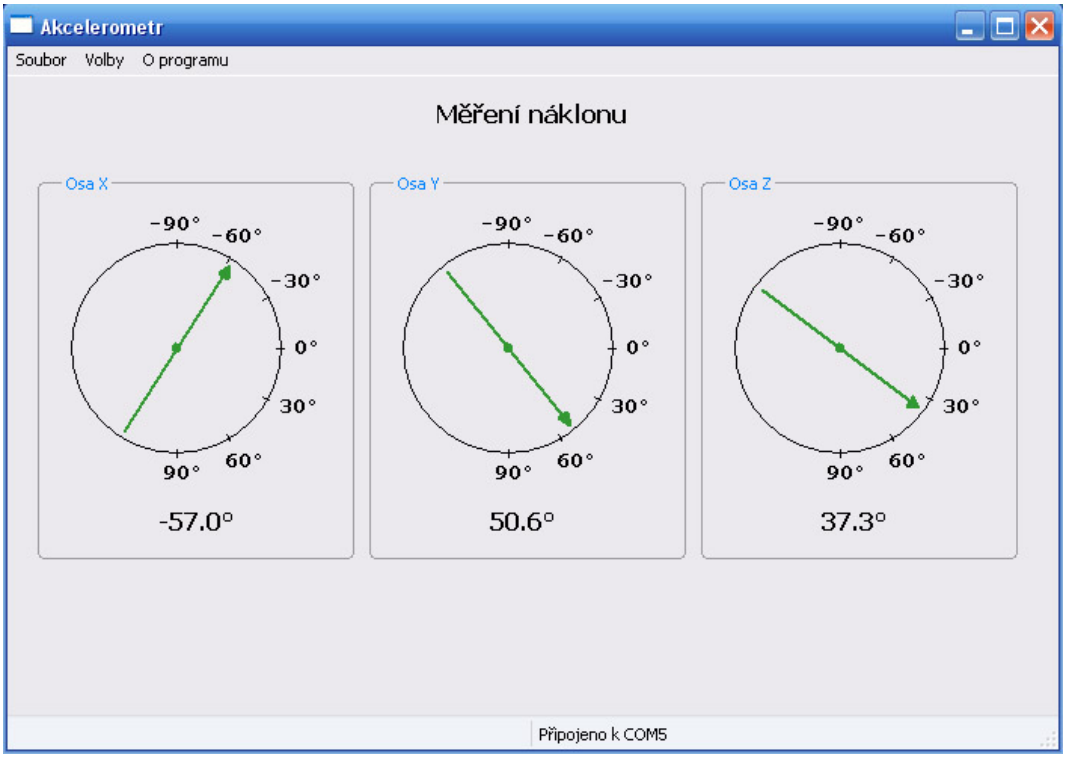
Při inicializaci dojde k vytvoření bitmapy kružnice s popisy, která je společná pro všechny tři osy a uloží se do paměti. Ta je poté pomocí wxWidgets metody `Blit` 3x vykreslena na plochu (pro každou osu). Následně dochází k vykreslení šipek, které mají úhel podle hodnot zrychlení, a textových popisků hodnot náklonu. Nakonec dojde k vykreslení rámů okolo kružnic.

Překreslení probíhá na základě události `OnPaint()`, volané při zneplatnění obsahu kreslící plochy, či vynuceně, zavoláním příkazu `Refresh()`. Pro zamezení zbytečnému překreslování plochy pracuje metoda `OnPaint` ve dvou režimech, podle nastavené hodnoty `s_resize`. Pokud je hodnota `true`, dojde k překreslení celého panelu. To je využito při inicializaci, posunu, změně velikosti okna (`EVT_SIZE`) nebo při označení okna (`EVT_SET_FOCUS`). Hodnota `s_resize` se automaticky, po každém překreslení, nastaví na `false`. Proto bez opětovného vyvolání dojde ke kompletnímu překreslení pouze jednou. Pokud je hodnota `s_resize` `false`, dojde k překreslení pouze šipky a údaje o úhlu, a to pouze, pokud se hodnota liší od předchozí. Děje se tak pomocí metody `Blit`, kdy se bitmapou šipka přemaže a vykreslí se nová.

```
void PanelTilt::OnPaint(wxPaintEvent &WXUNUSED(event))
{
    ...
    // došlo ke změně hodnoty x nebo dc mimo platnost
    if(s_prev.x != s_act.x || b_resize)
    {
        // přemazání bitmapou
        ddc.Blit(22,80,210,282,&m_dc,0,0);
        // vykreslení šipky
        DrawArrow(ddc, wxPoint(112,180), 70, s_act.x);
        // vykreslení hodnoty úhlu
        DrawValue(ddc,wxPoint(110,285), s_act.x);
    }
    ...
    // došlo k zneplatnění okna
    if( b_resize)
    {
        // nadpis
        DrawTitle(ddc);
        // orámování osy x
        DrawRect(ddc, wxPoint(20,70), ... );
        ...
    }

    b_resize = false;
}
```

Aktuální data z akcelerometru se předávají zavoláním metody `RefreshData(const sAccel &data)`. Ukázka aplikace viz obrázek 4.8.



Obrázek 4.8: Ukázka módu Měření náklonu

Převod dat na hodnoty náklonu

Při naklonění modulu ve směru jednotlivých os dochází ke změně výstupního napětí akcelerometru, což lze využít pro měření náklonu. Hodnota A/D převodu 128 představuje vyváženou polohu 0 stupňů. Pohybem na jednu stranu, dle zvolené osy, se hodnota zmenšuje, na druhou stranu zvětšuje. Nevýhodou je, že nelze měřit větší náklon, než 90 stupňů oběma směry, protože po dosažení těchto mezních veličin již hodnota A/D převodu neklesá (neroste), ale vrací se zpět k 128.

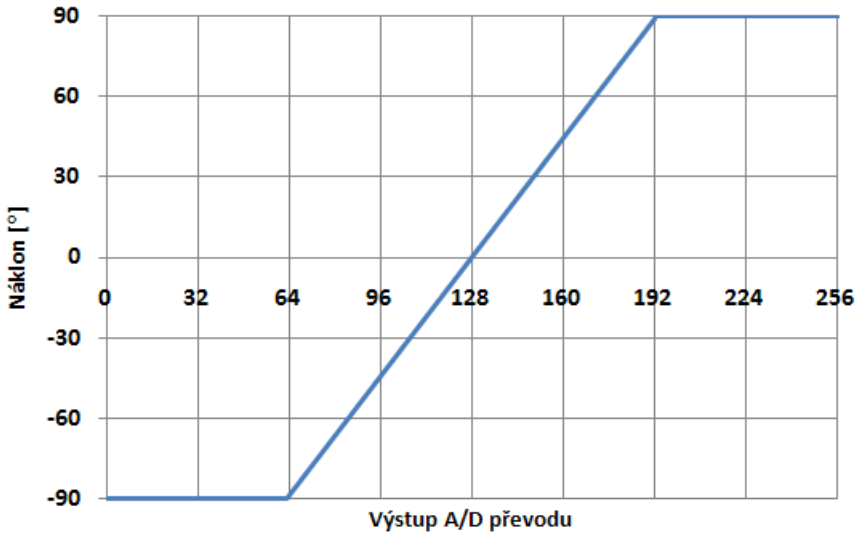
Pro převod z hodnot A/D převodu (rozsah 0–255) na velikost náklonu slouží globální funkce `digToDegree(int d)`. Převod probíhá dle vzorce $s = 0,703125 \cdot d - 90$, kde d je velikost A/D převodu a s je výsledný náklon ve stupních. Konstanta je zvolena tak, aby při $d = 0$ byl náklon $-90°$, při $d = 256$ byl $90°$.

```
double digToDegree(int d)
{
    return 0.703125*d - 90.0;
}
```

Metoda pro vykreslení šipky je taktéž navrhnuta pro rozsah A/D převodu 0–255, odpovídající -90° až +90°. Zde se však narazilo na problém, kdy při použitém módu akcelerometru 1,5g je tento rozsah jiný, a to 63–193. Graf znázorňující toto rozložení je na obrázku 4.9. Bylo proto potřeba jej převést pomocí následujících příkazů metody `RefreshData`:

```
// kompenzace pro mód 1,5g
#define COMPENS 1.9692
...
// převod na korektní rozložení
if(s_act.x > 128 ) s_act.x = 128 + (s_act.x-128)*COMPENS;
else s_act.x = 128 - (128 - s_act.x)*COMPENS;

// oříznutí hodnot mimo rozsah
if(s_act.x > 256) s_act.x = 256;
if(s_act.x < 0) s_act.x = 0;
```



Obrázek 4.9: Graf převodu hodnot A/D na velikost náklonu

Popis metod

Třída `PanelTilt`.

- `void RefreshData(const sAccel & data)`
  Slouží k předání aktuálních hodnot zrychlení z akcelerometru, a zajistí jejich zobrazení.
- `void CreateBitmap(wxDC & p_dc, int x, int y, int r)`
  Vytvoří bitmapu s rozměry $x \cdot y$ a s barvou pozadí jako plátno `p_dc`. Vykreslí se kružnice s poloměrem `r` a popisky úhlů.
- `void DrawArrow(wxDC & dc, wxPoint s, int r, int ad)`
  Do kružnice se středem `s` a poloměrem `r` vykreslí šipku, která bude mít náklon odpovídající hodnotě `ad` (číselný výstup A/D převodu).
- `void DrawRect(wxDC & dc, wxPoint s, wxPoint d, wxString title);`
  Vykreslí rámec z počátečního bodu `s` a rozměry `d` a popisem `title`.

- `void DrawValue(wxDC & dc,wxPoint s, int ad)`
  Vykreslí číselnou hodnotu náklonu na pozici `s`.
- `void DrawTitle(wxDC & dc)`
  Vykreslí nadpis na předem danou pozici.
- `void OnPaint(wxPaintEvent & event)`
  Překreslí panel. Metoda je voláná při zneplatnění obsahu nebo voláním wxWidgets funkce `Refresh()`.

Pokročilá implementace

Tato aplikace funguje bez problémů. Pro lepši znázornění lze vytvořit zobrazení náklonu ve 3d režimu.

Ovládání aplikace

Tato aplikace neobsahuje žádné ovládací prvky. Veškeré činnosti spojené s vykreslováním probíhají samostatně, bez nutnosti interakce uživatele. Jedinou podmínkou je mít připojený modul s akcelerometrem (zobrazeno ve statusbaru).

U této aplikace lze využít možnosti modulu akcelerometru, přepínat mezi naměřenými hodnotami náklonu. Modul musí být v režimu čtení z flash. Mezi daty se přesouvá tlačítky SW3 a SW4.

4.2.7 Měření okamžité rychlosti

Popis aplikace

Aplikace umožňuje z údajů zrychlení, získaných z modulu akcelerometru, vypočítat okamžitou rychlost a uraženou vzdálenost. Z ní se určí průměrná rychlost pohybu. Směr pohybu lze určit výběrem požadované osy. Ukázka aplikace viz obrázek 4.10.

Aplikace umožňuje zobrazit údaje rychlosti ve dvou základních veličinách:

- m/s
- km/h

Okamžitá rychlost je vypočtena pomocí vzorce $v_x = v_{x-1} + \Delta a_x t$ [4], kde v_{x-1} je rychlost vypočítaná v předchozím kroku a $\Delta a_x t$ je změna rychlosti za čas t. Počáteční rychlost pohybu je nulová ($v_0 = 0$).

Vzdálenost je počítána pomocí vzorce $s_x = s_{x-1} + \Delta v_x t$, kde s_{x-1} je vzdálenost spočtená v předchozím kroku a $\Delta v_x t$ je vzdálenost uražená při rychlosti v_x za čas t. Počáteční hodnota v_x je nulová.

Čas t trvání jednoho kroku lze měnit v hlavičkovém souboru a je standardně nastaven na $100ms$.

Veškeré veličiny jsou počítány v metodě `RefreshData`. Té jsou předávány aktuální data a časový údaj, který udává délku od posledního volání funkce.

```
void PanelSpeed::RefreshData(const sAccel &data, const int delta)
{
    time += delta;                                                    [1]
    int tmp = 0;
    switch( axis )                                                    [2]
    {
        case AX:
        tmp = data.x;
        break;
        ...
    }
    ...
    accel = digToG( tmp )*GCONST;                                     [3]
    actspeed += accel*(delta/1000.0);                                 [4]
    distance += abs(actspeed) * (delta/1000.0);                       [5]
    if(time>0) avgspeed = distance / ( (double)time/1000);            [6]
    ...
}
```

Nejprve je v těle funkce přičten časový údaj od předešlého volání `[1]`. Ze zaslaných dat je zvolena osa, která je vybraná v nastavení aplikace `[2]`. Její hodnota je převedena na velikost zrychlení v jednotce $m \cdot s^{-2}$ `[3]`. Z této veličiny je vypočten rychlostní přírůstek za čas `delta` a připočten k aktuální rychlosti `[4]`. Z aktuální rychlosti je určen přírůstek vzdálenosti za čas `delta [5]`. Z celkové vzdálenosti a času je vypočtena průměrná rychlost pohybu `[6]`.

Nevýhodou této aplikace je, že sebemenší vibrace, i naklonění modulu, se projeví vznikem zrychlení, a tedy změnou okamžité rychlosti. Aplikaci lze tedy použít pouze v prostředí s maximálním útlumem vibrací (např. výtahová kabina), kde se tyto rušivé faktory neprojeví. Pro vylepšení naměřených údajů by šel použít například Kalmanův filtr, ale to je mimo složitost této práce.

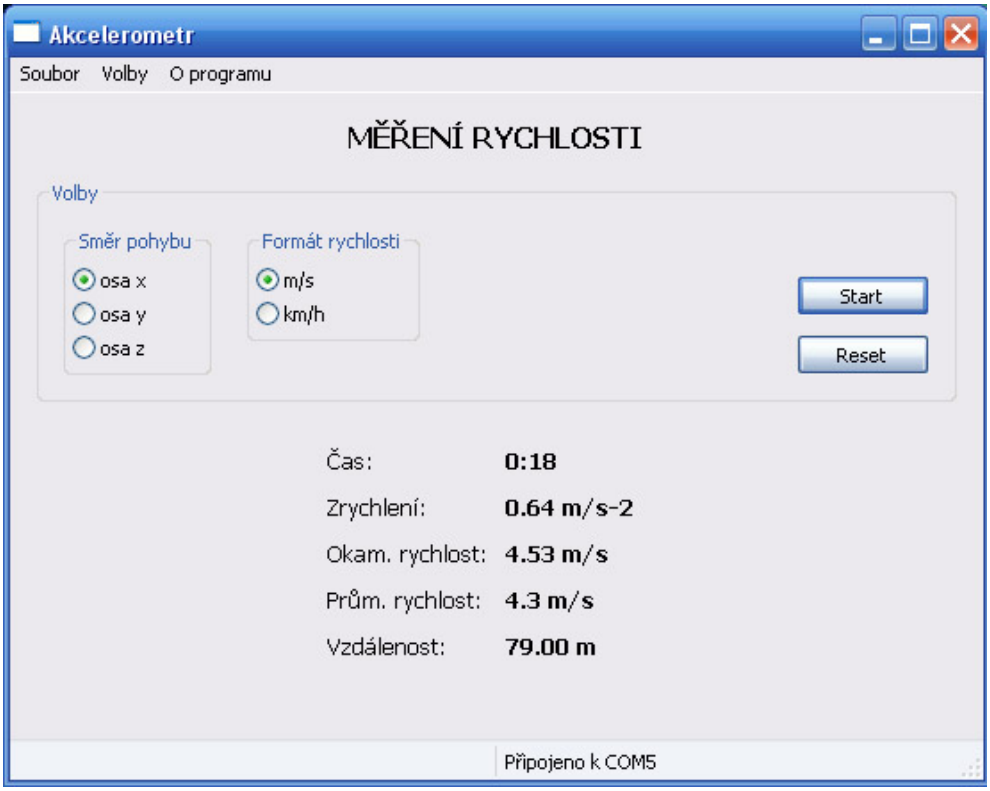
Kalmanův filtr

Kalmanův filtr je výkonný rekurzivní filtr, který umožňuje získat čistý signál a hodnoty ze zašuměného signálu nebo jinak znehodnoceného souboru hodnot, i bez jakéhokoliv poznatku o rušení. Prakticky lze takto matematicky zjistit hodnoty, které jsou přímým měřením těžko zjistitelné, protože se při samotném aktu měření do získaných hodnot indukují chyby měřících přístrojů nebo okolní působící šum a rušení.

Algoritmus pracuje ve dvou krocích:

- predikce nového stavu
- korekce integrací nového měření

Používá se například u senzorů (teplotní, polohový), pro správnou interpretaci měřených hodnot, které bývají zatíženy určitou chybou.[7]



Obrázek 4.10: Ukázka módu Měření rychlosti

Popis metod

Třída `PanelSpeed`.

- `void RefreshData(const sAccel & data, const int delta)`
  Slouží k předání aktuálních hodnot zrychlení z akcelerometru. Proměnná `delta` udává čas v ms od posledního zavolání této funkce a slouží k určení změny okamžité rychlosti.
- `void UpdateTime()`
  Aktualizuje zobrazovaný čas. Hodnota je přepočítána na hodiny, minuty, sekundy a zobrazena.
- `void UpdateValues()`
  Aktualizuje veškeré zobrazované údaje vyjma času.
- `void ResetValues()`
  Vynuluje veškeré údaje.
- `void OnBtnStart(wxCommandEvent& event)`
  Metoda události volané při stisku tlačítka Start/Stop.
- `void OnBtnReset(wxCommandEvent& event)`
  Metoda události volané při stisku tlačítka Reset.
- `void OnRadioFormat(wxCommandEvent& event)`
  Metoda události volaná při výběru zobrazované jednotky rychlosti.

- `void OnRadioAxis(wxCommandEvent& event)`
  Metoda události volaná pří výběru osy pohybu.

Pokročilá implementace

Pokud by byla vyřešena nevýhoda dosavadní implementace, popsána výše, např. pomocí Kalmanova filtru, či jinými metodami, lze aplikaci dále rozvíjet. Již by bylo možno kromě dosavadních veličin určit i dráhu pohybu a graficky zobrazit.

Ovládání aplikace

Veškeré ovládací prvky aplikace jsou v sekci *Volby*. V části *Směr pohybu* lze zvolit, z jaké osy akcelerometru se budou data počítat.

V části *Formát rychlosti* se vybírá formát, ve kterém je výsledná rychlost zobrazena.

Stiskem tlačítka *Start*, a za podmínky, že je připojen akcelerometr, dojde ke spuštení aplikace. Tlačítkem *Stop* se zastaví.

Tlačítko *Reset* slouží pro vynulování veškerých dat i časomíry.

Kapitola 5

Závěr

Tato práce se zabývala využitím modulu akcelerometru. Výsledkem je aplikace, která demonstruje několik typických aplikací založených na akcelerátorech gravitačního zrychlení.

První aplikace umožňuje měřit náklon ve všech třech osách akcelerometru v rozsahu -90 až +90 stupňů. U této aplikace lze využít možnosti SRB modulu ukládat naměřená data do FLASH paměti a po připojení k PC mezi nimi procházet.

Druhou aplikací je krokoměr. Pomocí jednoduchého algoritmu umožňuje detekovat kroky, a z nich odvodit průměrnou rychlost pohybu a ušlou vzdálenost. Samotná detekce není stoprocentní a vyžaduje podrobné nastavení citlivosti. Zde bych viděl jedno z možných rozšíření práce, a to implementaci pokročilého detekčního algoritmu.

Třetí aplikace slouží k odhadu okamžité rychlosti. Z údajů zrychlení umožňuje určit aktuální a průměrnou rychlost pohybu a vzdálenost. Jelikož je samotný akcelerometr velice citlivý na vibrace, které výrazně ovlivňují výsledné měření, je využití této aplikace pouze v ideálu blízkým podmínkám (např. výtahová kabina). Jako další možnost rozšíření práce je implementace filtru pro odhad plovoucího průměru, aby se zlepšila odolnost proti vibracím. Vhodné by bylo využít Kalmanův filtr.

Dalšími aplikacemi jsou logovací okno, které má podrobné možnosti nastavení a umožňuje naměřená data uložit do textového souboru a aplikace pro zobrazení údajů z akcelerometru ve třech veličinách (zrychlení, napětí a výsledek A/D převodu). Přidat další aplikace v rámci rozšíření práce je jednou z dalších možností.

Jelikož je program psán v multiplatformním wxWidgets, je možné rozšířit funkčnost i na jiné platformy (např. Linux). Zde zbývá pouze implementovat komunikaci mezi počítačem a modulem SRB pro daný operační systém.

Literatura

[1] Freescale Semiconductors: 13213 Evaluation Kits User Guide. http://www.freescale.com/files/rf_if/doc/user_guide/13213EVKUG.pdf.

[2] Freescale Semiconductors: MC13211/212/213. www.freescale.com/files/rf_if/doc/data_sheet/MC1321x.pdf.

[3] Freescale Semiconductors: Three Axis Low-g Micromachined Accelerometer. http://www.sparkfun.com/datasheets/Accelerometers/MMA7260Q-Rev1.pdf.

[4] Halliday D., W. J., Resnick R.: *Fyzika*. VUTIUM, 2000, iSBN 80-214-1869-9.

[5] Omega, technické reference: Akcelerometry [online]. http://www.omegaeng.cz/prodinfo/Accelerometers.html, [cit. 2009-02-22].

[6] PE micro: USB BDM Multilink [online]. http://www.pemicro.com/products/product_processor.cfm?family=2, [cit. 2009-04-28].

[7] Robotika.cz: Měření rychlosti [online]. http://robotika.cz/guide/filtering/en, [cit. 2009-04-18].

[8] RobotWiki: Akcelerometr [online]. http://wiki.kn.vutbr.cz/robot/index.cgi?akcelerometr, [cit. 2009-02-22].

[9] Sensorsmag: A Micromachined Thermal Accelerometer [online]. http://archives.sensorsmag.com/articles/0601/98/main.shtml, [cit. 2009-04-18].

[10] Silicon Designs: Capacitive accelerometers [online]. http://www.silicondesigns.com/tech.html, [cit. 2009-04-18].

[11] Smart J., Roebling R., Zeitlin V., Dunn R.: wxWidgets Online manual [online]. http://docs.wxwidgets.org/stable/, [cit. 2009-04-18].

[12] Stavovčík, B.: *Obecná navigace*. CERM, 2008, iSBN 978-80-7204-576-1.

[13] Wikipedie, o. e.: Akcelerometry [online]. http://en.wikipedia.org/wiki/Accelerometer, [cit. 2009-02-19].

[14] Wikipedie, otevřená encyklopedie: MEMS [online]. http://en.wikipedia.org/wiki/Microelectromechanical_systems, [cit. 2009-02-19].

[15] Wikipedie, otevřená encyklopedie: Princip ekvivalence [online].
`http://en.wikipedia.org/wiki/Equivalence_principle`, [cit. 2009-02-19].

Dodatek A

Obsah CD

Složky:

```
/bin                    - obsahuje spustitelný program Akcelerometr
/install                - instalační soubory pro wxWidgets a SMAC 4.2
/src/Akcelerometr       - projekt aplikace Akcelerometr
/src/Accel_V3.0         - řídící program kitu
/tex                    - zdrojové soubory technické zprávy
/zigbee                 - image cd ZigBee Evaluation Kit
```

Soubory:

```
xhorni00_bp.pdf         - text bakalářské práce
xhorni00_bp_print.pdf   - text bakalářské práce pro tisk
Readme
```

Dodatek B

Návod pro překlad projektu

B.1 Řídící program modulu SRB

Program byl testován na následujícím programovém vybavení:

- **Freescale Code Warrior IDE 5**
  - dostupný spolu s kitem
- **SMAC 4.2**
  - demonstrační aplikace pro MC1321x
  - dostupné na přiloženém CD v adresáři install/SMAC_4.2

Instalace

1. Nainstalovat Test tool z CD ZigBee Evaluation Kit (dostupné s kitem nebo jako image na přiloženém cd bakalářské práce `zigbee\zigbee_img.iso`)
2. Nainstalovat SMAC 4.2 (defaultně `C:\Program Files\Freescale\SMAC 4.2`)
3. Přepsat původní složku `Accel_V3.0` nacházející se v `...\SMAC 4.2\S08\apps\` složkou z přiloženého CD z adresáře `...\src\Accel_V3.0`
4. Otevřít v Code Warrior
   `...\SMAC 4.2\S08\apps\Accel_V3.0\3.1\Accelerometer V3.mcp`
5. Vybrat cílový modul 13213-SRB a přeložit
6. Pomocí programátoru nahrát do modulu SRB

B.2 Aplikace Akcelerometr

Aplikace byla vyvíjena na následujícím programovém vybavení:

- **Microsoft Visual Studio 2008**
- **wxWidgets 2.8.9 pro Windows**
  - dostupné na přiloženém CD `/install/wxWidgets`

Instalace

1. Nainstalovat wxWidgets (defaultně `C:\wxWidgets`)

2. Přeložit knihovny wxWidgets:
   Ve Visual Studiu otevřít `C:\wxWidgets\build\msw\wx.sln`
   Vybrat konfiguraci (Debug / Release) a spustit překlad (F7)

3. Z přiloženého cd BP zkopírovat složku s projektem `Akcelerometr`
   (umístěnoou `\src\Akcelerometr`) do `C:\wxWidgets\samples` a ve Visual Studiu otevřít `C:\wxWidgets\samples\Akcelerometr\akcelerometr.vcproj` (projekt musí být 2 úrovně pod kořenovým adresářem `C:\wxWidgets\`).

4. Přeložit projekt Akcelerometr se stejnou konfigurací jako u bodu 2.