BROB - Základy robotiky

Projekt: Vizualizace dat z akcelerometru

Pavel Vejnar (134657)
Karel Vaculík (134649)

letní semestr, 2012

Obsah

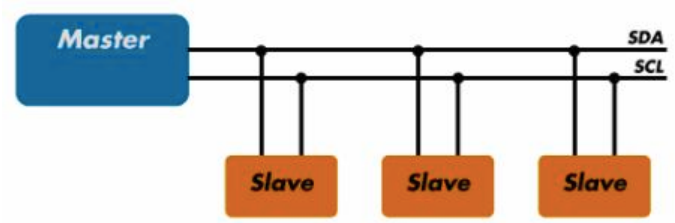
Zadání projektu

Seznamte se předloženými akcelerometry a navrhněte zapojení pro komunikaci mezi nimi a počítačem. Proměřte a porovnejte dodané moduly a udělejte vhodnou vizualizaci dat na PC.

Od akcelerometru k PC

I2C Sběrnice

Jde o moderní a často používanou digitální sběrnici. Celé sběrnici stačí pouze 2 vodiče, což celé řešení značně usnadňuje a zlevňuje. Na sběrnici je možno připojit několik zařízení. Každé má svoji jedinečnou adresu, díky které se dostaneme na konkrétní zařízení.



Obr. 01: Blokové schéma připojení zařízení (slave) na sběrnici

Výše uvedené blokové schéma znázorňuje obvyklé zapojení I2C sběrnice. Vždy je zařízení typu Master, které generuje hodinový signál a komunikuje s jednotlivými zařízeními. Hodinový signál je generován na vodiči SCL (serial clock) a po vodiči SDA (serial data) jdou data.
V praxi je ještě třeba zajistit i společnou zem GND. Dále je třeba připojit na sběrnici dva pull-up odpory. Důvodem je, že normou definovaný klidový stav je logická jednička. Jsou to odpory připojené na napájecí napětí a přivedené na SCL a SDA. Obvykle se používají v rozsahu 1-10kΩ. Čím nižší hodnota, tím jsou náběžné a sestupné hrany ostřejší a můžeme si dovolit větší rychlost komunikace. Ale jak vyplývá z Ohmova zákona, stoupne nám spotřeba, která může být zásadní vzhledem k poměru odebíraného zařízení (řádově až setiny miliampér) ku odporům (až jednotky miliampér).

Obr. 02: Princip komunikace

Princip komunikace po sběrnici je následující. Pomocí sedmi bitů je vybráno jedno ze zařízení, pracující v režimu Slave. Další bit R/W určuje, zda budeme ze zařízení číst nebo do něj zapisovat. Následně se přenesou data (8 bitů) a následuje potvrzovací ACK bit. Pro ukončení komunikace se odešle STOP bit.

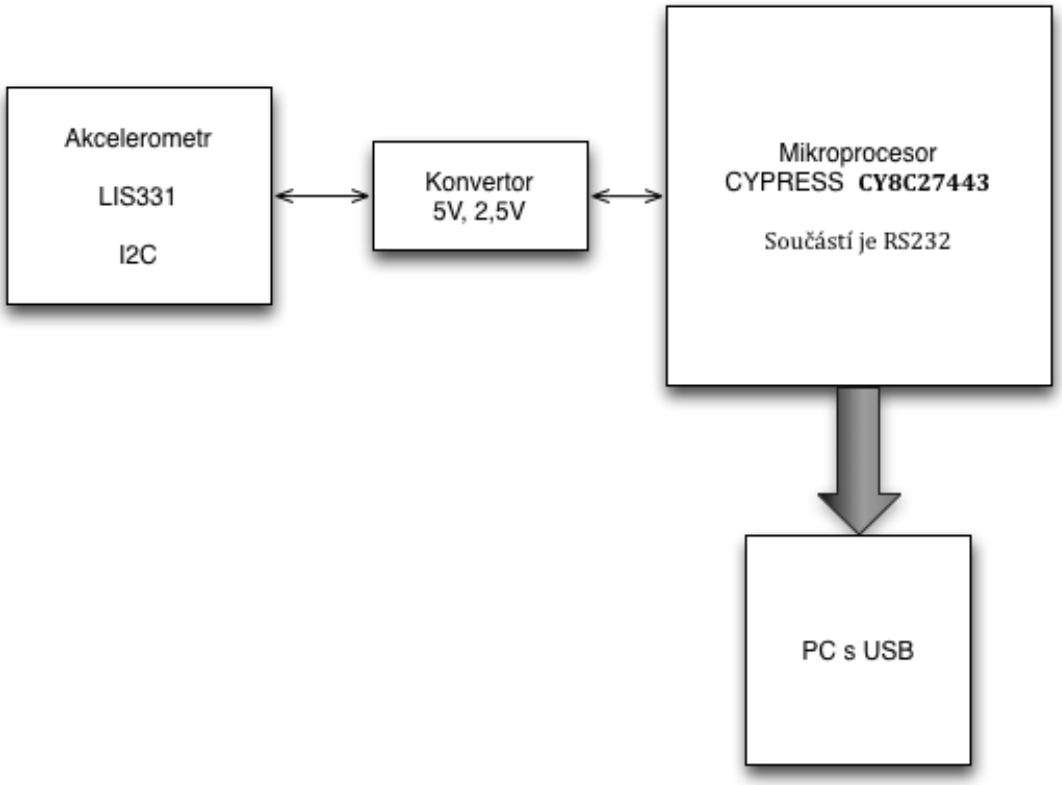
Přenosová rychlost	Označení
10 kbps	low speed mode
100 kbps	standard mode
400 kbps	fast mode
1 Mbps	fast mode +
3,4 Mbps	high speed mode

Tab. 02: Tabulka podporovaných rychlostí I2C sběrnice

Rychlost I2C sběrnice je také definována výše uvedenými standardy, však zařízení fungují i na jiných frekvencích (rychlostech).

Komunikace mezi akcelerometrem a počítačem

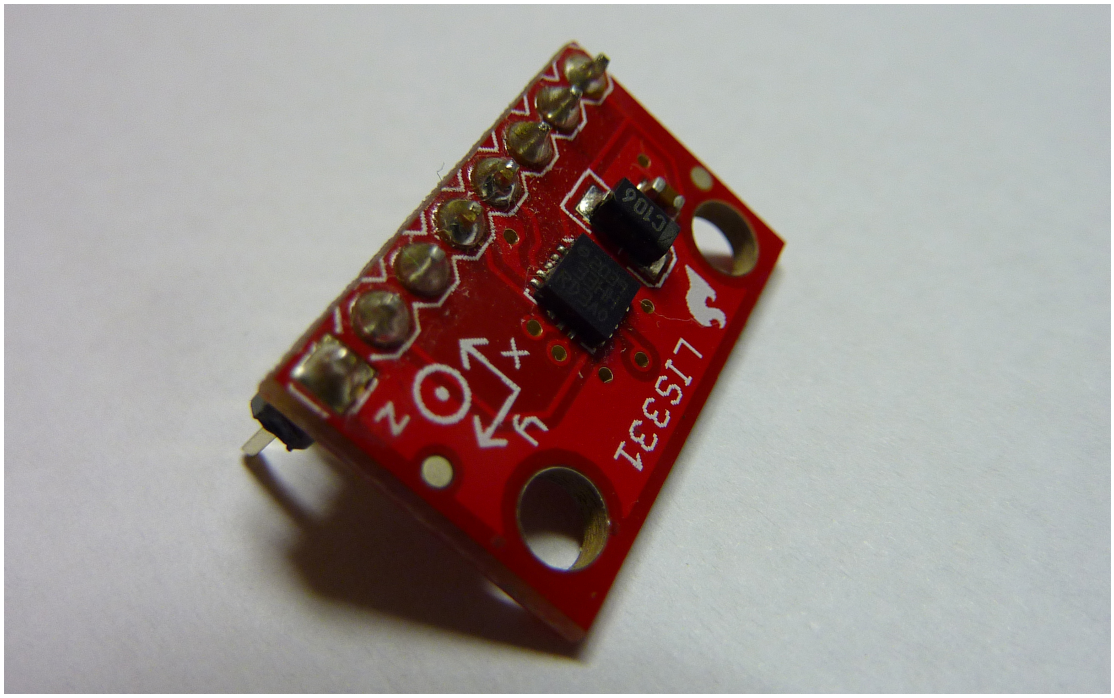
Skládá se ze tří důležitých bloků. Samotného akcelerometru, převodníku z 5V na 2,5V a mikroprocesoru s převodníkem na RS232.



Obr. 03: Blokové schéma propojení mezi akcelerometrem a PC

Akcelerometr LIS331

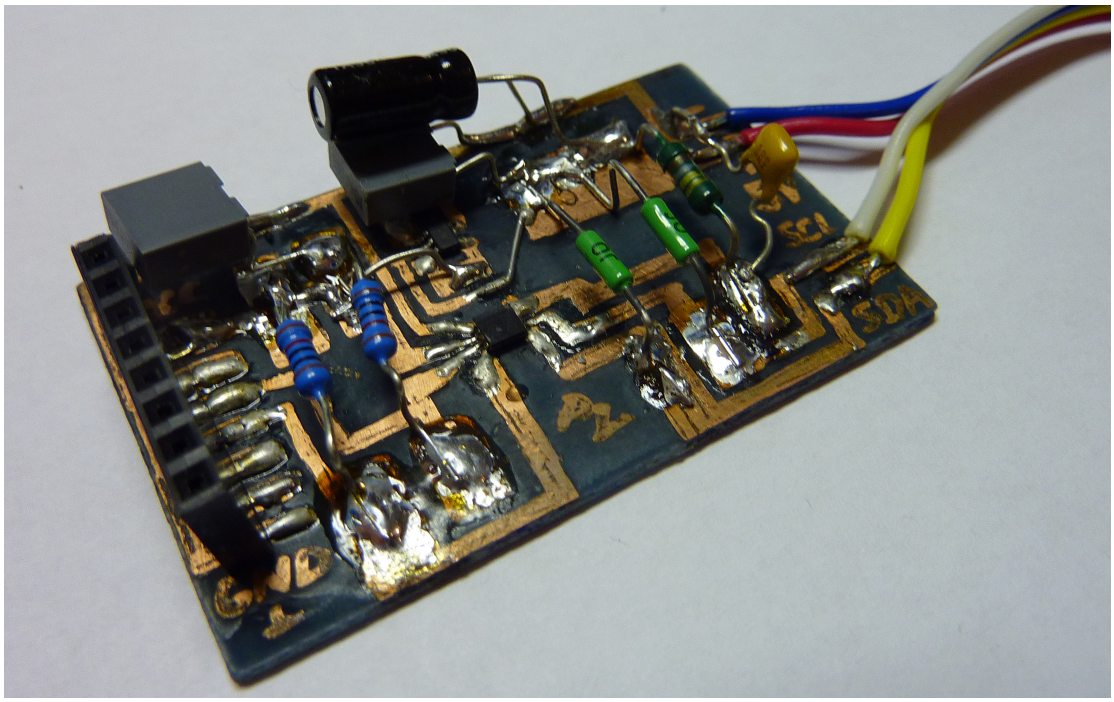
Samotný akcelerometr nebylo třeba dále upravovat, pouze stačilo přiletovat piny pro snadnější propojení na desku.



Obr. 04: Reálné foto akcelerometru LIS331

Převodník z 5V na 2,5V

Zde bylo třeba navrhnout rozumný obvod pro převod z 5V logiky, která šla z mikroprocesoru, na 2,5V logiku, která je třeba pro akcelerometr. Dalo by se to vyřešit napájením mikroprocesoru 2,5V, které podporuje, ale vznikl problém s převodníkem na RS232, který potřeboval pro své napájení 5V (řešení na již hotovém vývojovém kitu mikroprocesoru).



Obr. 05: Reálné foto převodníku 2,5V, 5V (obousměrně)

Volba 2,5V logiky by se mohla zdát neobvyklá, však samotný akcelerometr podporuje rozsah 2,16-3,6. Akcelerometr je z výroby kalibrován na 2,5 voltu, tudíž byla zvolena tahle hodnota napětí.
Jako DC-DC převodník byl zvolen vzorek od Analog Devices **ADR391**, který má výstup 2,5V, 5mA. Bylo zvoleno základní schéma zapojení z datasheetu (i s kondenzátory).

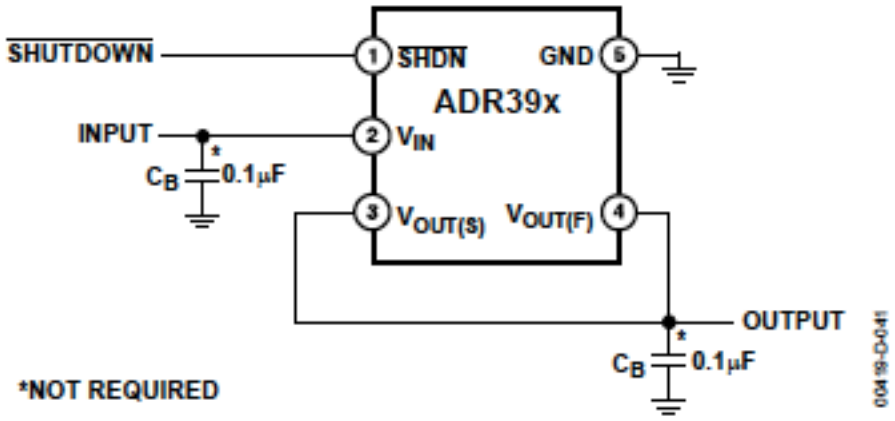
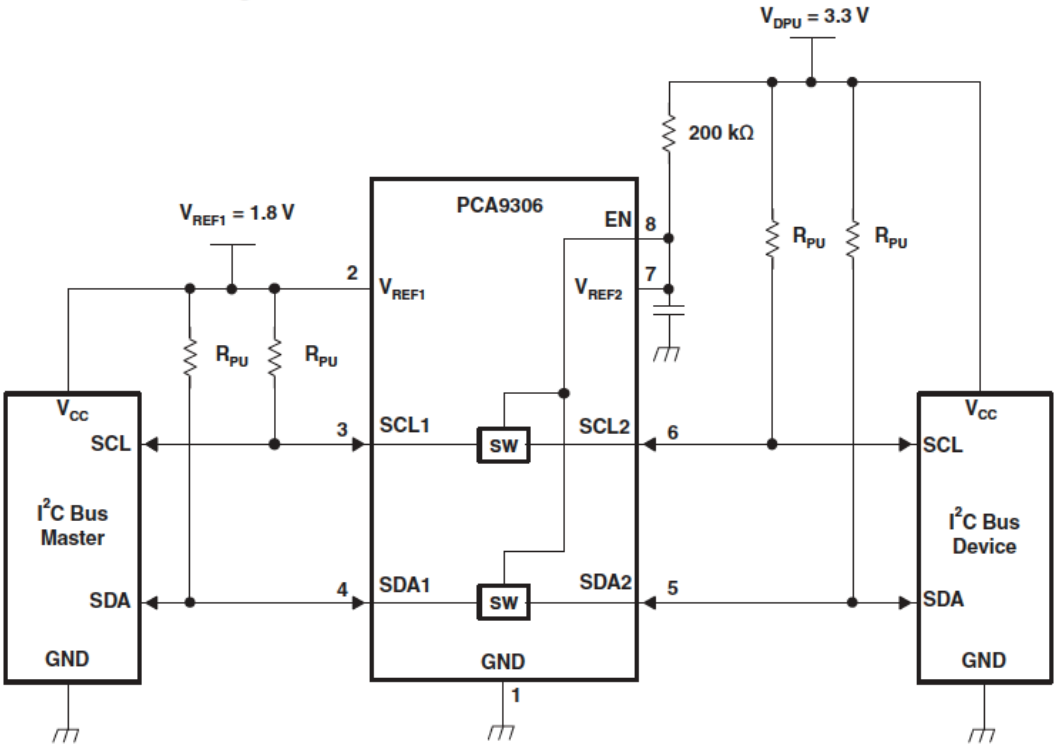


Figure 41. Basic Configuration for the ADR39x Family

Obr. 06: Schéma zapojení DC-DC měniče ADR391

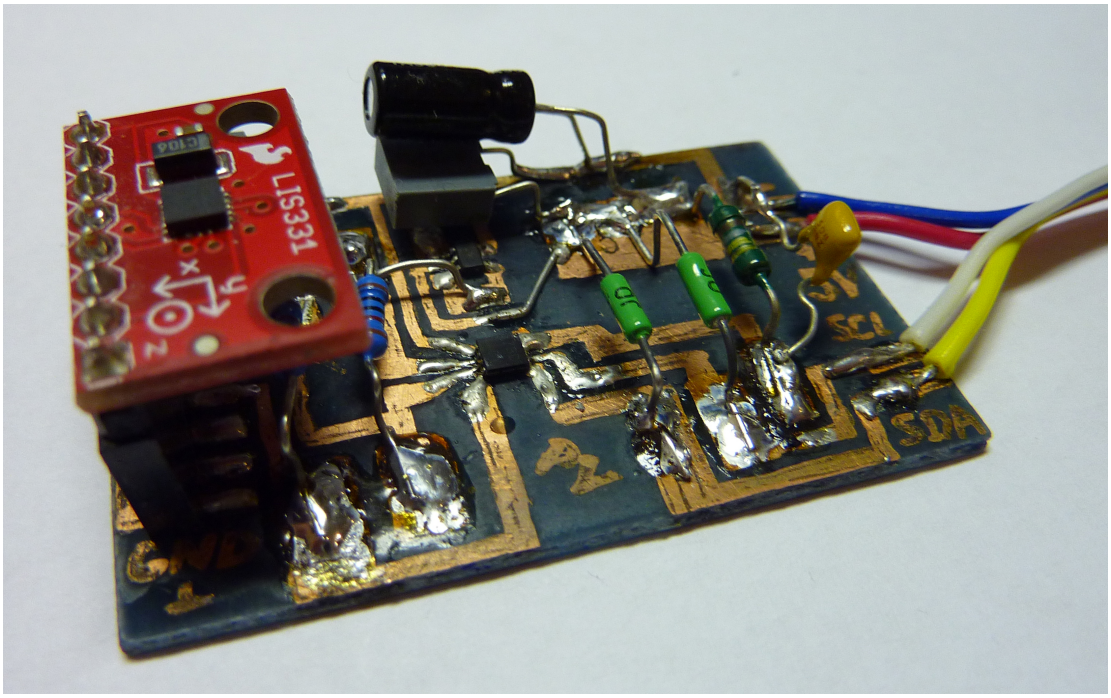
DC-DC převodník napájí samotný akcelerometr + dva pull-up odpory, které jsou součástí standartu I2C sběrnice. Původní hodnota pull-up odporů byla 1kΩ. Snadným výpočtem odběru proudu, (*I=2,5V/1kΩ=2,5mA a máme 2 odpory, takže celkový odběr odporů bylo 5mA, samotný akcelerometr má odběr maximálně 0,25mA),* se dalo zjistit, že zdroj je na hranici použitelnosti a ve špičce musí dodat **5,25mA.**
Čím menší odpor tím, je strmější náběžná hrana. V našem případě se ukázalo, že v stačí odpor 10kΩ (volba vychází ze zkušeností při navrhování), takže výsledný proudový odběr byl 0,5mA (odpory) + 0,25mA (akcelerometr) tudíž **0,75mA** nám v pohodě DC-DC převodník dodá.
Dále bylo třeba snížit napětí na logických úrovních SCL a SDA. K tomu byl využit vzorek od firmy Texas Instrument **PCA9306.** Jedná se o 2-bitový obousměrný převodník úrovní navržený speciálně pro I2C sběrnice. Podporuje více druhů převodů napětí a jedním z nich je právě z 5V na 2,5V.
Bylo využito zapojení, doporučeného výrobcem:



Obr. 07: Schéma zapojení PCA9306

Odpory ve schématu jsou zároveň pull-up odpory pro celou I2C sběrnici. Na straně s 5V logikou byly použity odpory 6k8, zde však na odběru nezáleží. Orientačně, to je to asi 1,5mA, odběru pro oba rezistory. Spotřeba samotného PCA9306 je zanedbatelná (řádově jednotky μA). Hodnoty napětí ve schématu jsou jen informativní důležité je, že na pravé straně musí být vždy vyšší napětí.

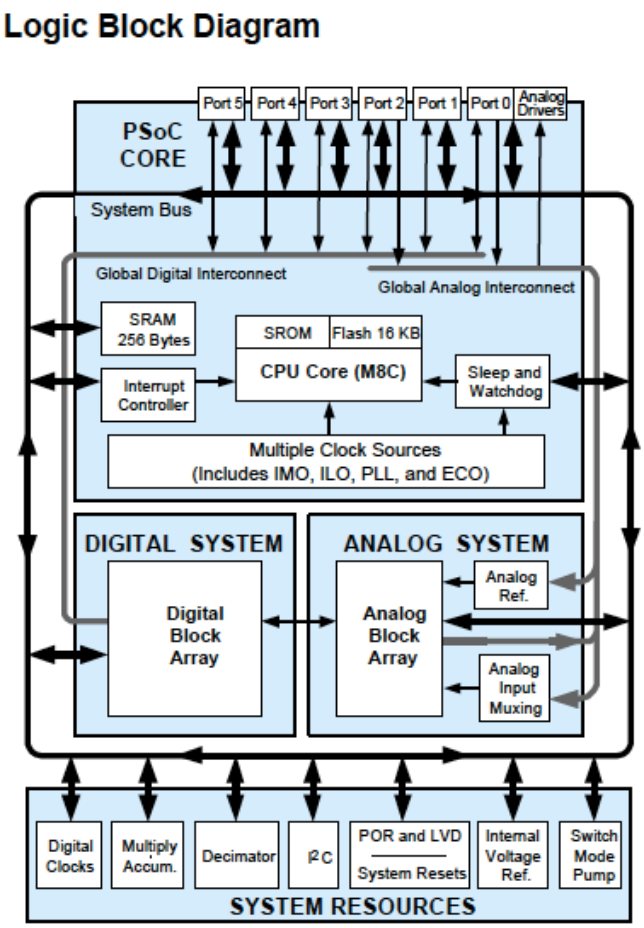
Na vstupu celého převodníku byl navíc připojen 5μF elektrolytický kondenzátor pro zlepšení stability zapojení. Ostatní kondenzátory byly fóliové, což zaručuje minimální šum, který by při dosahovaných rychlostech mohl způsobovat neočekávané poruchy.
Celý převodník je koncipován pro snadnou manipulaci a tudíž je přímo na něm přes piny osazen akcelerometr . Pro spojení s piny mikroprocesoru jsou vyvedeny 4 vodiče, (5V, GND, SCL a SDA). Více jich v našem zapojení není potřeba.



Obr. 08: Reálné foto kompletního převodníku, osazeného akcelerometrem LIS331

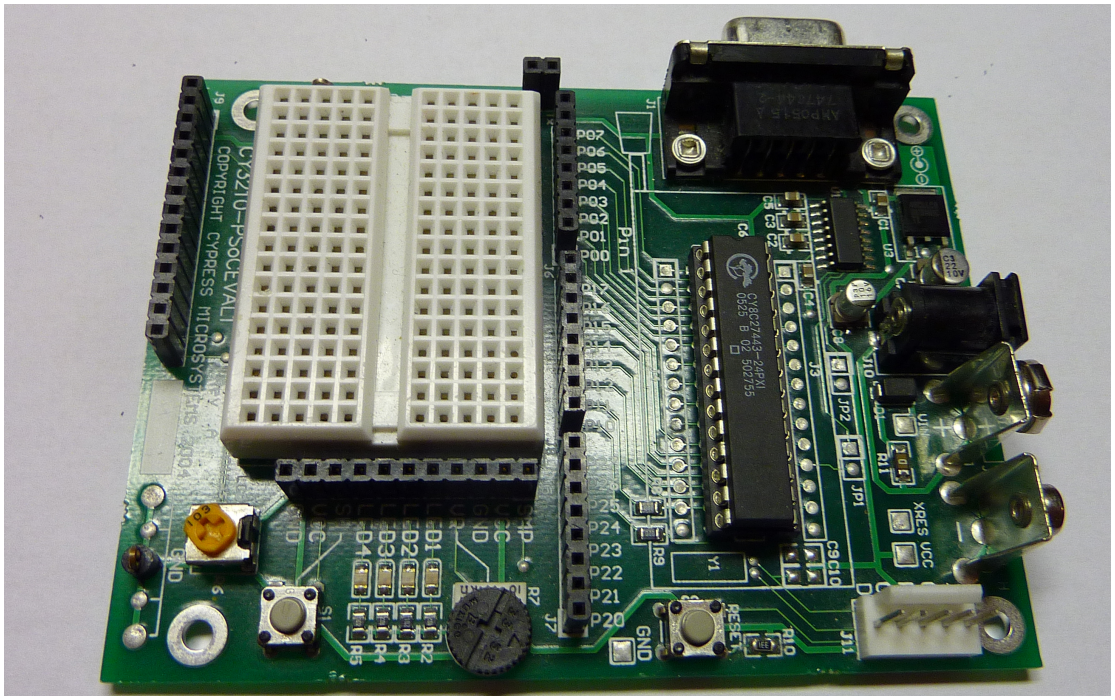
Mikroprocesor

Byl použit mikroprocesor od firmy CYPRESS s kódovým označením **CY8C27443**. Je to velice pokročilý mikroprocesor, který je o něco dále než obvykle požívaná Atmega. Jednou z výhod je například možnost volby, k čemu bude příslušná nožička mikroprocesoru využita. Obsahuje vnitřní oscilátor na 24MHz a 48MHz. Samotný procesor běží na 24MHz. 16kB programové paměti. Výstup pinu je maximálně 24mA, analogový 30mA. Napájení je 3V a 5V, může pracovat i v režimu na 1,5V či 1,8V, má však nižší proud na výstupu.



Obr. 09: Blokové schéma mikroprocesoru

Mikroprocesor byl dodán s kompletní vývojovou deskou, která obsahuje vstup pro programátor, DC-DC měnič na 5V, 4 smd led diody (červené) i s odpory, malé nepájivé pole a převodník na RS232.

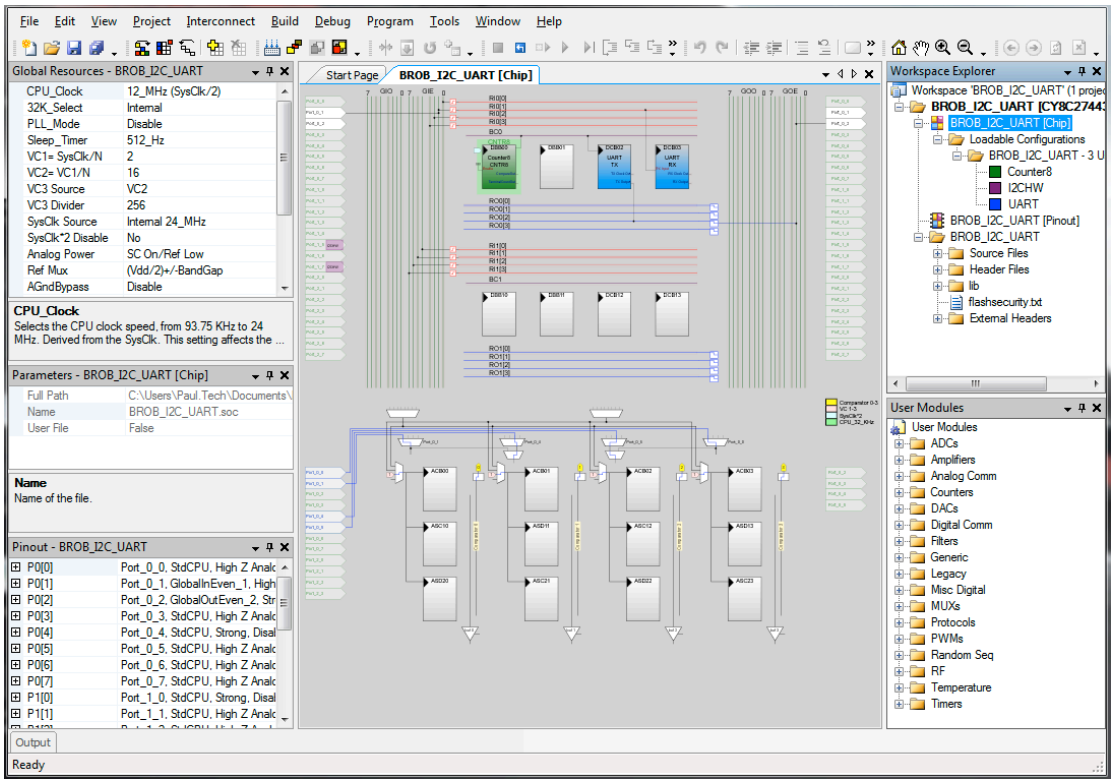


Obr. 10: Reálné foto vývojové desky, osazené mikroprocesorem CYPRESS CY8C27443

Jelikož již není obvyklé integrovat do notebooku COM port, byl použit externí převodník USB-SERIAL kvůli připojitelnosti do USB portu.

Obr. 11: Reálné foto USB-SERIAL převodníku

Popis vývojového prostředí pro mikroprocesor.
Mikroprocesor byl programován ve výrobcem dodávaném programu PSoC. Je to uživatelský příjemné programovací prostředí, vhodné i pro začátečníky. Při startu programu je na výběr programováni mikroprocesoru v jazyce Asembler nebo v C. Možnou výhodou, je napsání celého programu bez znalostí registrů v mikroprocesoru. Vše je primárně řešeno bloky a hotovými funkcemi. Ke všemu je přehledný a kompletní datasheet.



Obr. 12: Snímek obrazovky vývojového prostředí PsOCpro mikroprocesor

Princip programování komunikace

Programování mělo dvě části. První bylo rozchození UARTu, což je komunikace mezi mikroprocesorem a počítačem. Nám ve výsledku stačilo jen data odesílat do počítače, takže zpětný přenos nebylo potřeba řešit.
Druhou částí bylo rozchození I2C sběrnice formou zápisu do registrů akcelerometru. Po úspěšně navázané komunikaci bylo nutno nastavit pouze dva registry.
CTRL_REG1 na adrese 0x20, zapsat 0x27, což způsobí zapnutí napájení k akcelerometru a zapnutí všech tří os.
CTRL_REG4 na adrese 0x23, zapsat 0x40, tímhle registrem se dalo nastavit v jakém formátu chceme výslednou osu, jestli napřed horní nebo dolní část (Endian).
Výstupní data z jednotlivých os je 16 bitové číslo rozdělené do dvou registrů. Poté již následnuje čtení jednotlivých os.
Vše bylo uspořádáno do přehledných funkcí, například nastavení registru CTRL_REG1 se provedlo fukcí

```
set_CTRL_REG1();
```

Dále byly v úvodu nadefinovány jednotlivé osy

```
int XL=0x28;      //registr osy X (Low registr)
```

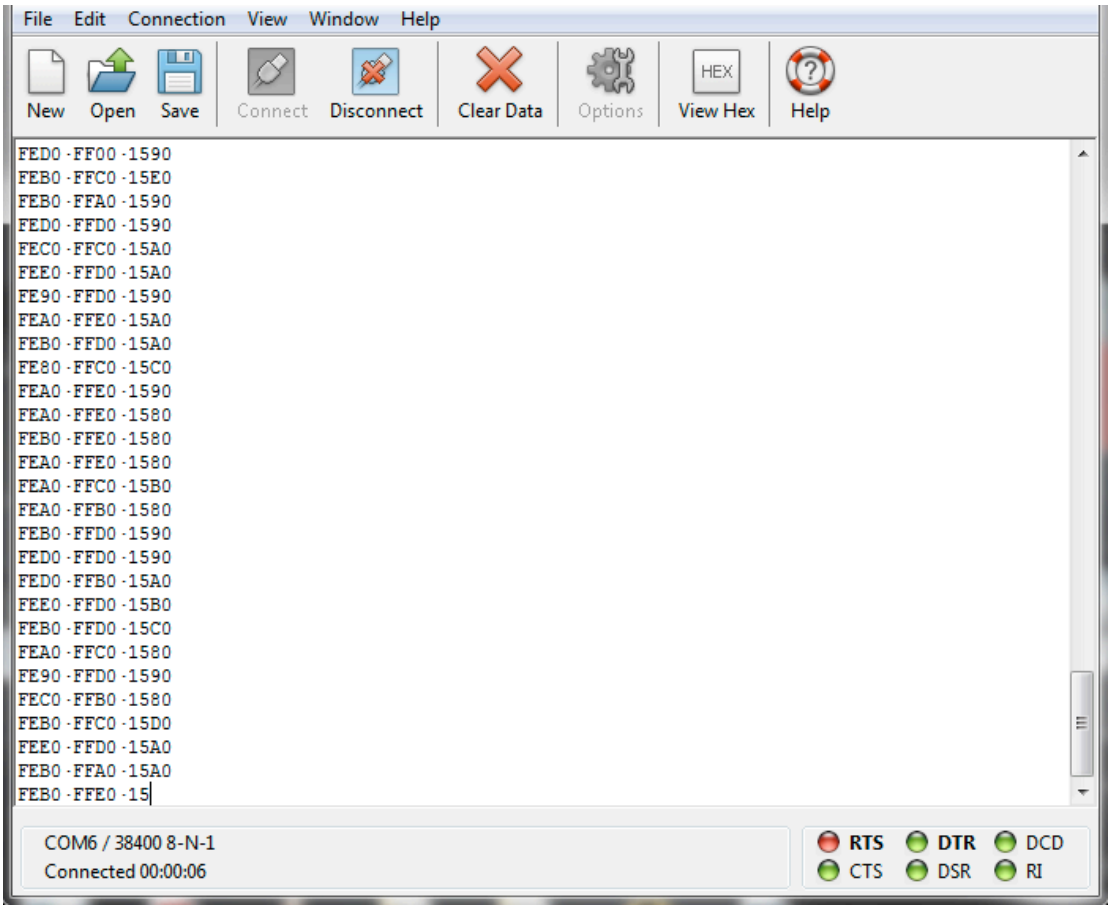
A následné stačilo do nekonečné smyčky umístit fukci, které předáme registr, který chceme přečíst a ona jej následně zapíše, přečte a vytiskne

```
OUT(XL);
```

Následující funkce tiskla na UART libovolné znaky, v našem případě další řádek

```
UART_CPutString("\n");
```

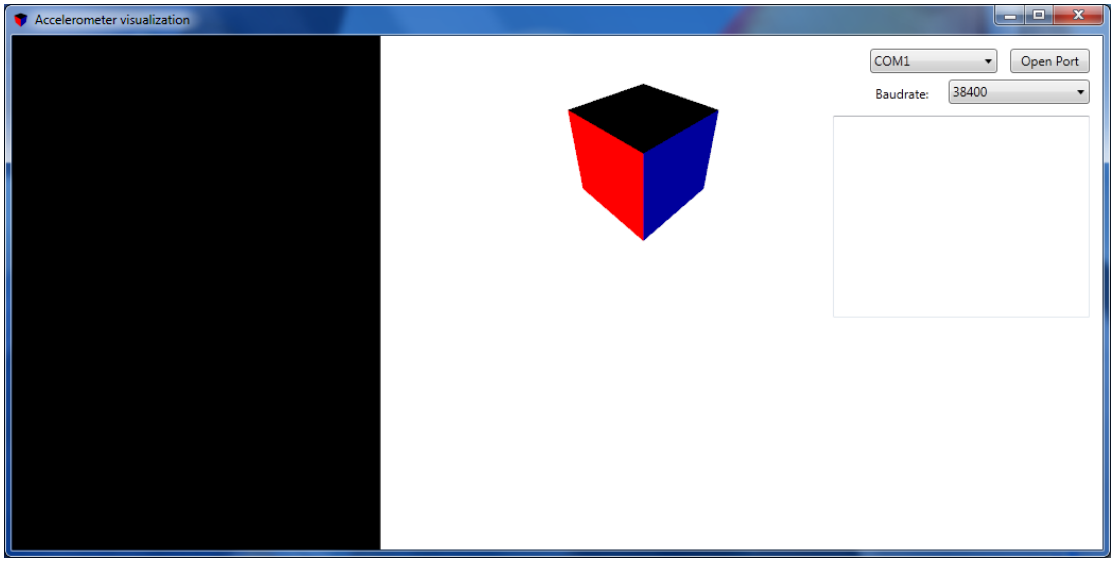
Do počítače šla tedy surová data přímo z akcelerometru. Na konzoli to vypadalo následovně:



Obr. 13: Ukázka formátu dat přímo z akcelerometru (prostředí CoolTerm)

Vizualizace dat na PC

Aplikace běžící na PC byla naprogramována v jazyce C#, k vývoji bylo použito MS Visual Studio 2010. Jako platformu jsme zvolili MS Windows. Jedním z důvodů zaměření se na tuto platformu byla knihovna Windows Presentation Foundation (dále jen WPF), která dovoluje vytvářet bohaté GUI s podporou 3D grafiky a která má dobře zpracovanou dokumentaci na webu MSDN [1].
Na obr. 12 je zobrazena aplikace po spuštění (před otevřením portu).



Obr. 14: Aplikace pro vizualizaci dat z akcelerometru po spuštění

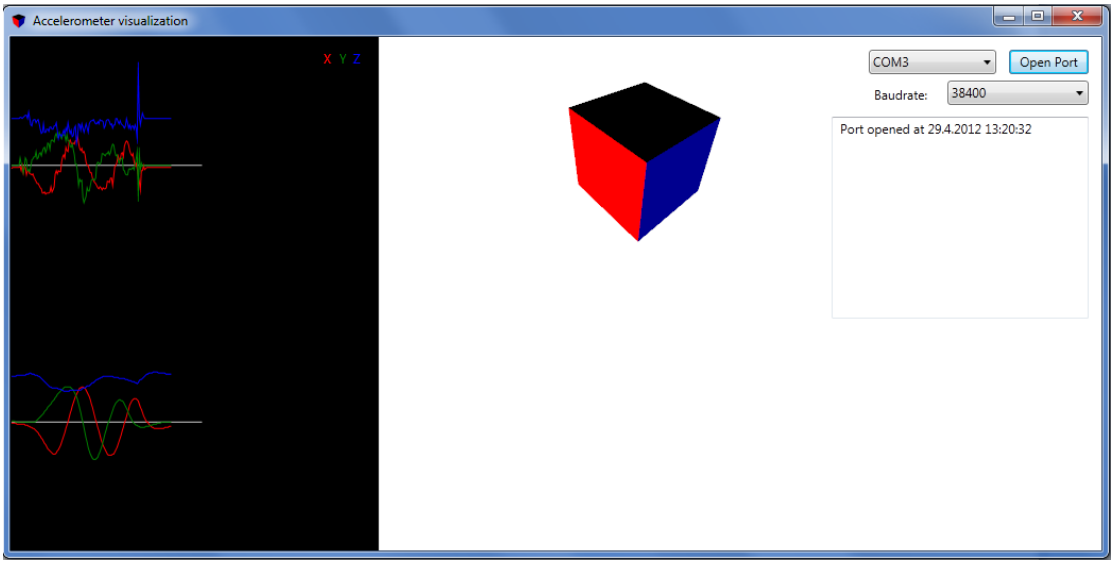
Celý proces zpracování dat ze sériového portu až po vizualizaci probíhá v několika fázích. Příklad výsledné vizualizace je znázorněn na obr. 13. Vizualizace je realizována ve dvou formách:

1) Graf posledních hodnot zrychlení pro osy X, Y, Z (viz černé plátno na obr. 13). Jednotlivé osy jsou rozlišeny červenou (X), zelenou (Y) a modrou (Z) barvou. Přesněji jsou zde obsaženy dva grafy. Horní graf, který zobrazuje čistá data tak, jak jsou získávána z akcelerometru, a dolní graf, který zobrazuje vyhlazená data pomocí Kalmanova filtru (viz dále).
2) Natočení krychle v trojrozměrném zobrazení, které sleduje reálné natočení samotného akcelerometru.

V dalších částech textu budou jednotlivé fáze programu popsány podrobněji.

Předzpracování dat ze sériového portu

V aplikaci je použito pouze čtení ze sériového portu. Pro tento způsob komunikace má jazyk C# k dispozici velmi vhodnou třídu `SerialPort`. V naší aplikaci jsme implementovali pomocnou třídu `SerialCommunicator`, která spravuje komunikační kanál právě pomocí třídy `SerialPort`.
Primárním úkolem třídy `SerialCommunicator` je otevření portu, příjem dat a odeslání dat ve vhodné podobě další části programu.



Obr. 15: Aplikace pro vizualizaci dat z akcelerometru za běhu

Před samotným otevřením portu, je potřeba vhodně nastavit tyto atributy objektu:

- `BaudRate`
- `Parity`
- `StopBits`
- `DataBits`
- `PortName`

Atribut `Parity` je nastaven napevno na hodnotu `None`, `StopBits` na `One` a `DataBits` na hodnotu 8. `BaudRate` si uživatel může zvolit pomocí seznamu v okně aplikace, stejně tak si může zvolit port, ke kterému má připojený akcelerometr. Seznam portů je vygenerován na základě právě dostupných portů pro použitý PC.
Taktéž je nutné nastavit delegáta pro `DataReceived`, který se bude starat o příjem dat ze sériového portu. V tomto delegátu realizujeme samotné čtení dat. Používáme načtení celého řádku jako `string`, nicméně k dispozici jsou i jiné metody, a to i čtení samotných bitů. V případě úspěšného otevření portu anebo při výskytu chyby je informován uživatel prostřednictvím textového rámečku, jak lze vidět například na obr. 13.

Řádek dat ze sériového portu v textové podobě má tento tvar:

HHHH HHHH HHHH

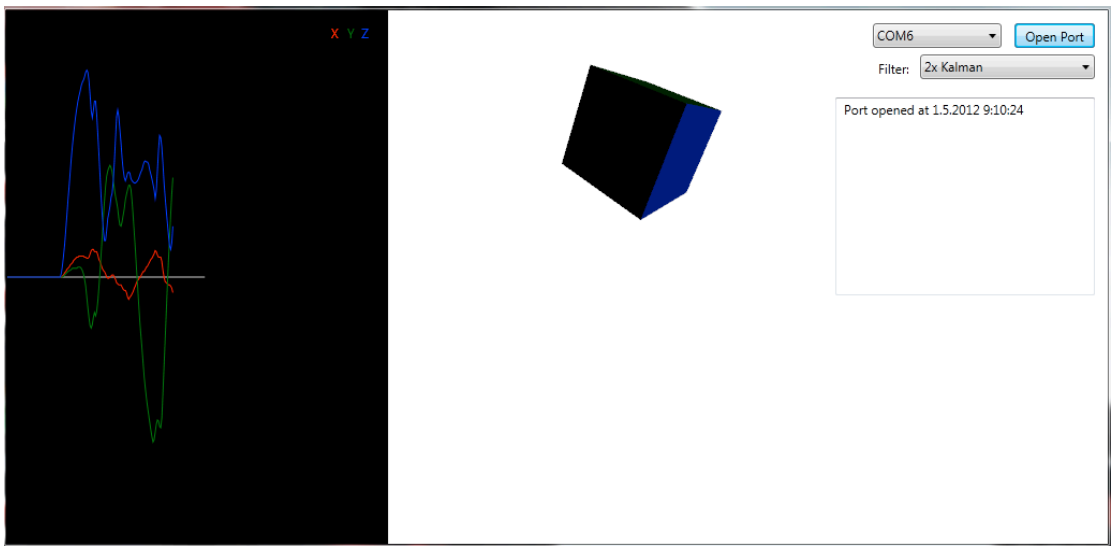
V tomto tvaru znak „H“ označuje hexadecimální číslici. Každá čtveřice kóduje 16bitové celé číslo. Postupně tato tři čísla označují hodnotu zrychlení pro souřadnice X, Y a Z. Příkladem může být tento řádek:

Pak X = 65280, Y = 65504 a Z = 5696. O rozdělení řetězce a konverzi jednotlivých čísel se před odesláním do další části programu stará taktéž třída `SerialCommunicator`.

Zpracování trojic hodnot

V další fázi je každá trojice hodnot přeposlána hlavní třídě programu, která spravuje GUI okno programu.
Tato třída se má postarat o to, aby načtené hodnoty byly patřičně vizualizovány – jednak jako graf hodnot zrychlení, jednak jako natočení krychle v trojrozměrném prostoru.
Aby bylo možné udržovat v grafu poslední historii hodnot zrychlení, ukládáme získané hodnoty do fronty. V případě nárůstu fronty do určité délky jsou nejstarší hodnoty mazány, aby se udržoval maximálně jistý počet hodnot.
Pro každou osu je udržována fronta jak pro načtená data, tak pro data vyhlazená filtrem. Celkem je tedy zapotřebí 6 front.

Jak již bylo zmíněno, pro vyhlazování dat se používá Kalmanův filtr [2]. Je to často používaný filtr, který přináší velmi slušné výsledky. Jeho charakteristickou vlastností je schopnost odhadovat nové hodnoty, což je velká výhoda v naší aplikaci, neboť můžeme filtrovat data v reálném čase a neprojevuje se nám zde zpoždění. Toto je velmi důležité, protože na základě filtrovaných dat je v našem programu prováděno otáčení krychle. Tato vlastnost Kalmanova filtru je ale zároveň i nevýhodou, protože nedokáže reagovat na prudké změny.



Obr. 16: Ukázka filtrace při prudké změně

Jak lze vidět na obrázku 13 u hrubých dat z akcelerometru, šum v datech je celkem výrazný. Pokud bychom chtěli upravit Kalmanův filtr tak, aby byl schopen vypořádat se s prudkými pohyby, těžko bychom mohli odhadnout, která

hodnota je jen vychýlením zapříčiněným šumem a která znamená skokovou změnu. Alespoň v reálném čase je to celkem nemožné. Pro naši aplikaci však výhody použití Kalmanova filtru celkově převládají nad nevýhodami.

K implementaci Kalmanova filtru jsme se nechali inspirovat zdrojovým kódem na webové stránce [3]. Nepodstatné části kódu byly vypuštěny.
V našem kódu je Kalmanův filtr realizován jako samostatná třída, jejíž instance si uchovávají aktuální hodnoty průběžně přepočítávaných parametrů algoritmu. Konstantní parametry filtru byly zvoleny experimentálně.
Pro každou osu se ve skutečnosti počítají dva průchody filtrem, aby byla řada hodnot dostatečně vyhlazená. Tzn., že pro každou osu jsou vytvořeny dvě instance filtru. První filtr bere na vstup hrubá data, druhý filtr pak bere výstup z prvního filtru. Tento modulární přístup umožňuje samozřejmě zřetězit i více filtrů, v našem případě to však výraznou změnu už nepřinese, takže by to bylo plýtvání časem výpočtu.

Aktualizace grafů

Po každé nově načtené a vyfiltrované trojici čísel se vykreslují data z front do grafů. Hodnoty se přepočítávají po každé přidané trojici, protože je potřeba jednotlivé body horizontálně posunout, abychom vytvořili efekt časového průběhu. Navíc musí být všechny hodnoty normalizovány, aby se vešly do černé vykreslovací oblasti, a vertikálně posunuty do správné části. Tyto výpočty se provádí relativně vzhledem k velikosti plochy, aby se vykreslování grafů dokázalo přizpůsobit při změně velikosti okna.

Natočení krychle

Zobrazení vyhlazených dat v grafu má spíše ilustrativní účel. Primárně jsou tato data použita pro výpočet natočení krychle v trojrozměrném prostoru, čímž se dostáváme do závěrečné fáze zpracování dat.
3D grafika je ve WPF založena na technologii Direct3D. Krychle, jak ji vidíme na obr. 12 nebo 13, je složena z 12 trojúhelníkových ploch, které jsou v prostoru určeny souřadnicemi svých vrcholů. Pro úplnost zobrazení jsou přidány zdroje světla a vhodně umístěn zdroj a směr pohledu na scénu.

Rotace 3D objektů ve WPF se provádí především určením úhlu a vektoru, kolem kterého se má objekt otáčet o tento úhel. Knihovna WPF nabízí také možnost rotace pomocí kvaternionů, v podstatě se však podle dokumentace jedná opět o otočení kolem daného vektoru o daný úhel.

V programu se používají dvě rotace, kolem vektoru (3,0,0) a (0,3,0).
Výpočet rotace byl založen na vzorcích [4]:

$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\varphi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

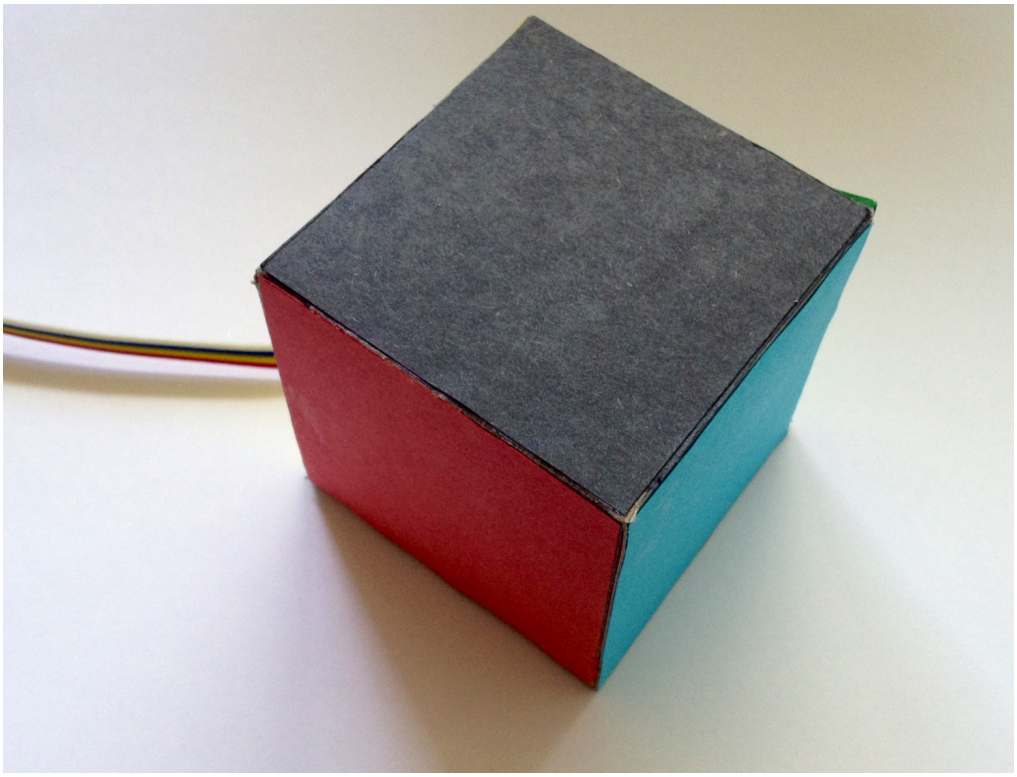
Bylo však potřeba rozšířit tyto vzorce na výpočet schopný pracovat pro celý rozsah 360° u obou rotací.

Pseudokód výpočtu pro danou trojici X, Y, Z:

```
lastZ ← 10          //libovolna hodnota na zacatek
aroundx ← false
updateCube(x, y, z)
     if (z * lastZ < 0)
          if (|x| - |y| > 0)
               aroundx ← true;
          else
               aroundx ← false;
     ρ ← Atan(x / sqrt(y*y + z*z)) * 57.2957795
     if (z < 0 ∧ aroundx)
          ρ ← 180 - ρ
     φ ← Atan(y / sqrt(x*x + z*z)) * 57.2957795
     if (z < 0 ∧ ¬aroundx)
          φ ← 180 - φ
     lastZ ← z
```

V hodnotě `lastZ` se uchovává minulá hodnota proměnné `z`. Pomocí ní se detekuje přechod mezi zápornými kladnými hodnotami. Logická proměnná `aroundx` určuje, který úhel se má při záporné hodnotě `z` upravit (kvůli rozšíření do 360°). Vypočtené hodnoty úhlů `ρ, resp. φ`, se dále použijí pro natočení kolem osy (3,0,0), resp. (0,3,0).

Na tomto místě je vhodné podotknout, že vyhlazování pomocí Kalmanova filtru je možno provádět až na vypočtených úhlech natočení. Je to jeden z možných návrhů pro budoucí rozšiřování.



Obr. 17: Pro jasnou vizualizaci jsme dali akcelerometr do papírové kostky, shodou jako v programu na PC

Závěr

Tento projekt bychom hodnotili jako velice přínosný v oblasti praktické realizace. Naučili jsme se zde spoustu nových věcí za relativně krátký čas. Například rozchození komunikace akcelerometr-PC obnášelo naučit se programovat mikroprocesor, porozumět principu funkce I2C sběrnice, zjistit jak cist data z akcelerometru (že to není jen čteni, ale i nastaveni registru), a dále alespoň základně znát „obvodařinu“ a návrh plošných spojů.
Nevýhodou projektu nám přišel až moc krátký čas na celkovou realizaci a bez alespoň základních znalostí z různých zaměření skoro nerealizovatelný.
Náš projekt byl pro 2, takže nebyl problém s rozdělením na dvě části. Samotný vizualizační program a komunikace akcelerometru – PC. Převážně jsme každý pracovali na své části a následně je spojili v jednu.
Podařilo se nám vizualizovat data z akcelerometru přímo do počítače. Jediný známý a nevyřešený problém je občasné přerušení dodávky dat z akcelerometru a občasné následné zamrznutí mikroprocesoru. Je to zcela náhodné a tudíž těžce identifikovatelné a bez osciloskopu skoro nereálné. Tomuhle problému bylo věnováno opravdu hodně času, ale přesto zůstal nevyřešen.
Byla řešena ještě jedna věc. Bezdrátová komunikace. Sice nebyla součástí zadání, ale v tomhle řešení je více než vhodná. Komunikace byla řešena přes Bluetooth. Bylo to velice jednoduché a nebylo potřeba mnoho změn, bohužel modul běžel na 57600 baudech a přenos dat byl ještě více náchylný k chybám. Těžko říci, jestli to bylo způsobeno předchozím problémem, přerušení dodávky dat nebo samotným modulem.
Projekt byl tak zajímavý, že je v plánu ho dotáhnout do podoby bezdrátové komunikace a odstranění chyb. (ve vlastním zájmu)

Reference

[1] http://msdn.microsoft.com/en-us/library/ms754130.aspx

[2] http://en.wikipedia.org/wiki/Kalman_filter

[3] http://www.manicai.net/blog/files/3565d84c26b0e46715b35d63f6d33ea5-1.html

[4] Sam Naghshineh, Golafsoun Ameri, Mazdak Zereshki. *Human Motion capture using Tri-Axial accelerometers*.
http://edge.rit.edu/content/P10010/public/PDF/HME.pdf

[5] Mikroprocesor http://www.cypress.com/l

[6] I2C http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf

[7] I2C http://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[8] Konverze napětí http://uart.cz/253/konverze-mezi-5v-a-3v-logikou/

[9] Datasheety: Lis331, CY8C27x43, PCA9306, BDD133, ADR391