FAKULTA ELEKTROTECHNIKY A INFORMATIKY STU

Krokomer

Semestrálne zadanie VRS

Dokumentácia

Patrik Buranský, Bálint Domonkos, Andrej Halán

Obsah

Zoznam obrázkov

Zadanie

Obsahuje:

- Akcelerometer LIS3DH (digitálny)
- Aplikácia (pre desktop)
- L152RE NUCLEO

Zadanie je v skratke jednoducho zadané: krokomer (pedometer). Neboli určené žiadne konkrétne podmienky vypracovania.

Link na GitHub: https://github.com/Patndrikej/BDH

Analýza údajov

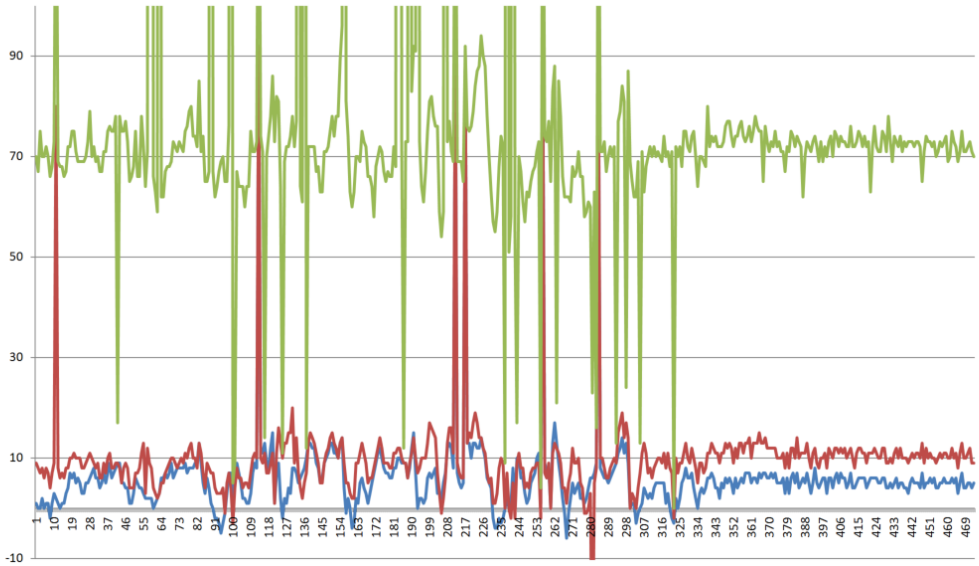
Pri návrhu aplikácie bolo potrebné, aby sme na začiatku vývoja analyzovali údaje z akcelerometra a určili tak priebeh nameraných dát.

Pre zjednodušenie analýzy sme sa rozhodli že namerané údaje budeme predbežne spracovávať v programe Excel, ktorý nám dostatočne poslúžil ako základ pre analýzu údajov.

Analýza pozostávala zo získaním údajov a ich ukladanie do textového súboru, na čo nám poslúžil program putty, cez ktorého logy sme získali použiteľné údaje na spracovanie. Údaje boli rýchlo spracované v textovom editory Sublime Text 2 kde došlo k odstránenie riadkov a prázdnych medzier a získali sme tak čisté údaje.

Získavanie údajov bolo uskutočňované pohybom akcelerometra pri vykonávaní krokov. Akcelerometer sme opatrne pripevnili na STM dosku tak, by bol stálej polohe a zabezpečili tak presnejšie získavanie údajov. Pri meraní sme akcelerometer držali najprv v dvoch polohách. Merali sme v takej polohe, že akcelerometer sme držali priamo pri tele aby sa prenášali otrasy. Najlepšie údaje sme zo začiatku získali, keď sme akcelerometer držali v ruke a pri chôdzi robili prirodzené pohyby dopredu a dozadu.

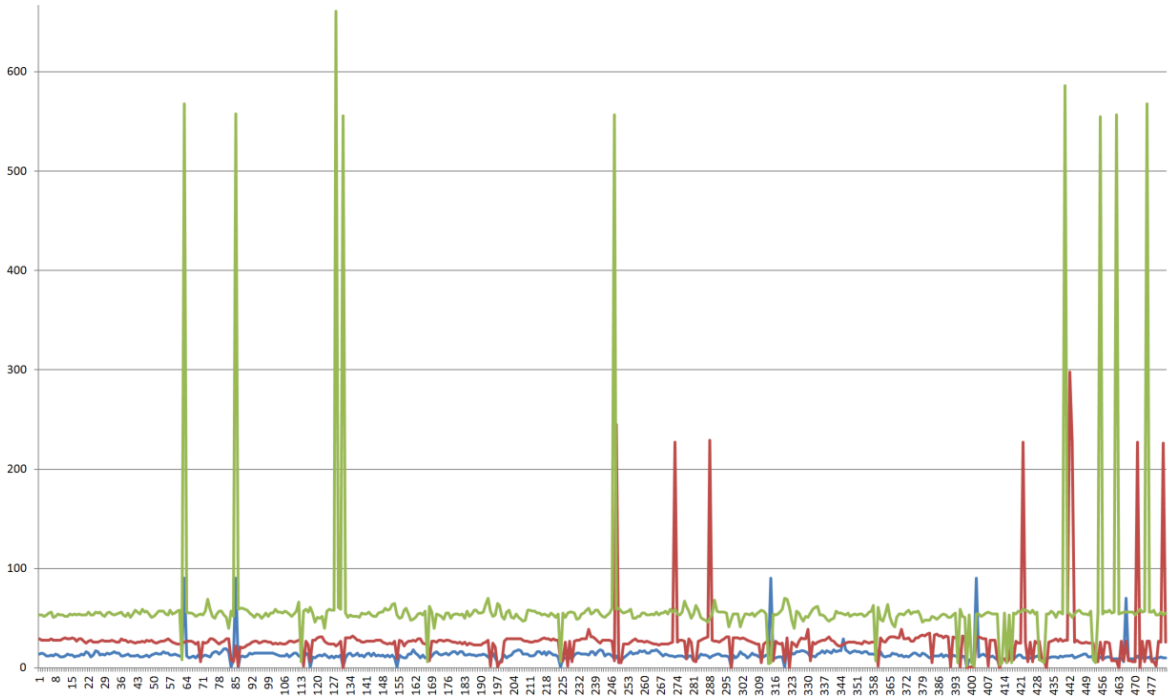
Na obrázku môžeme vidieť surové dáta získané zo všetkých osí z akcelerometra. Vidíme, že dáta sú nepriehľadné a mätúce.



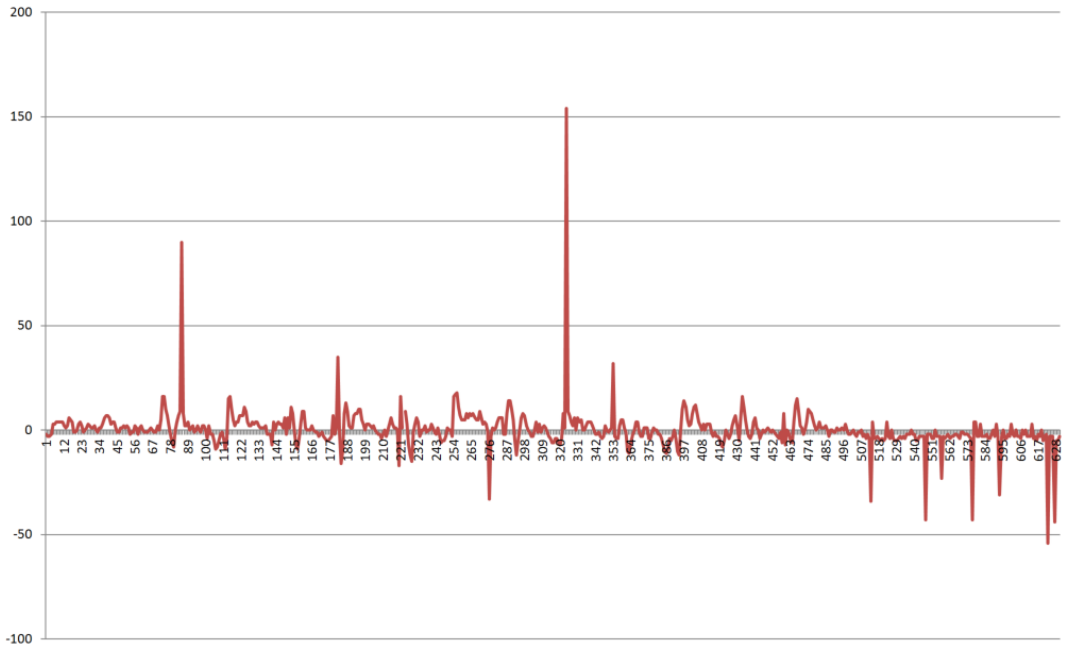
Obrázok 1 Surové dáta os x,y,z

Dokopy sme spravili vyše 50 meraní, merania sú k dispozícií v súbore excel Priebehy_dat.xlsx. Pri sledovaní všetkých priebehov je vidieť, že prvé merania sú z hodnotami ktoré neobashujú žiadne filtre. Jedná sa o surové dáta z akcelerometra. Z týchto hodnôt bolo ťažké určiť čo je krok no pomohli nám k filtrovaniu ďalších hodnôt a tak sme dosiahli efektívnejší výsledok.

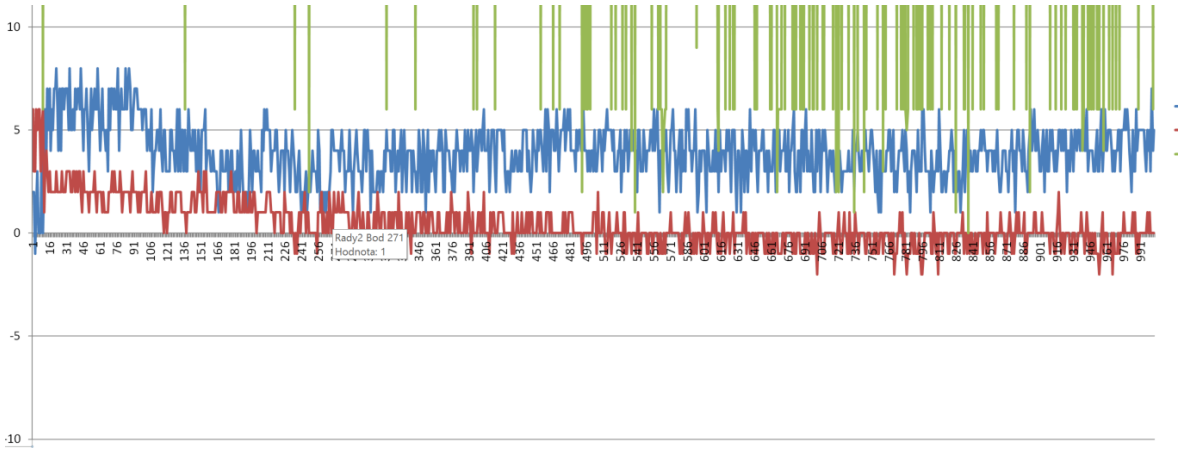
Nižšie uvedené obrázky ukazujú postupnosť meraní. Môžeme vidieť, že zo začiatku sú dáta neprehľadné. Keď sa použije filter na filtráciu údajov, údaje sú jasnejšie a viac pripomínajú priebeh kroku:



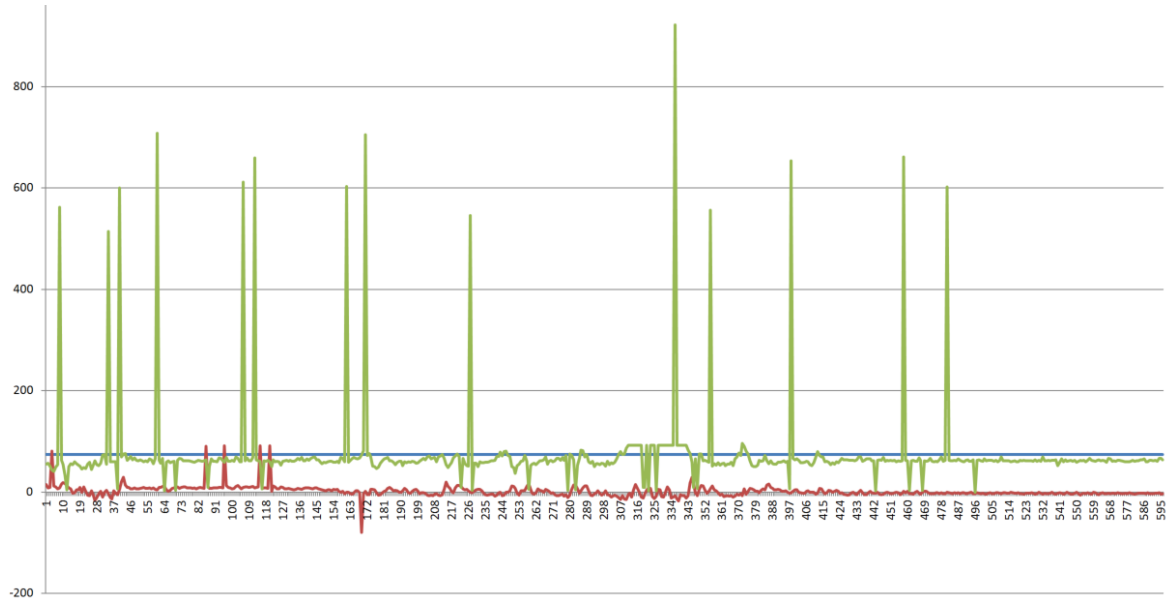
Obrázok 2 Surové dáta os x,y,z



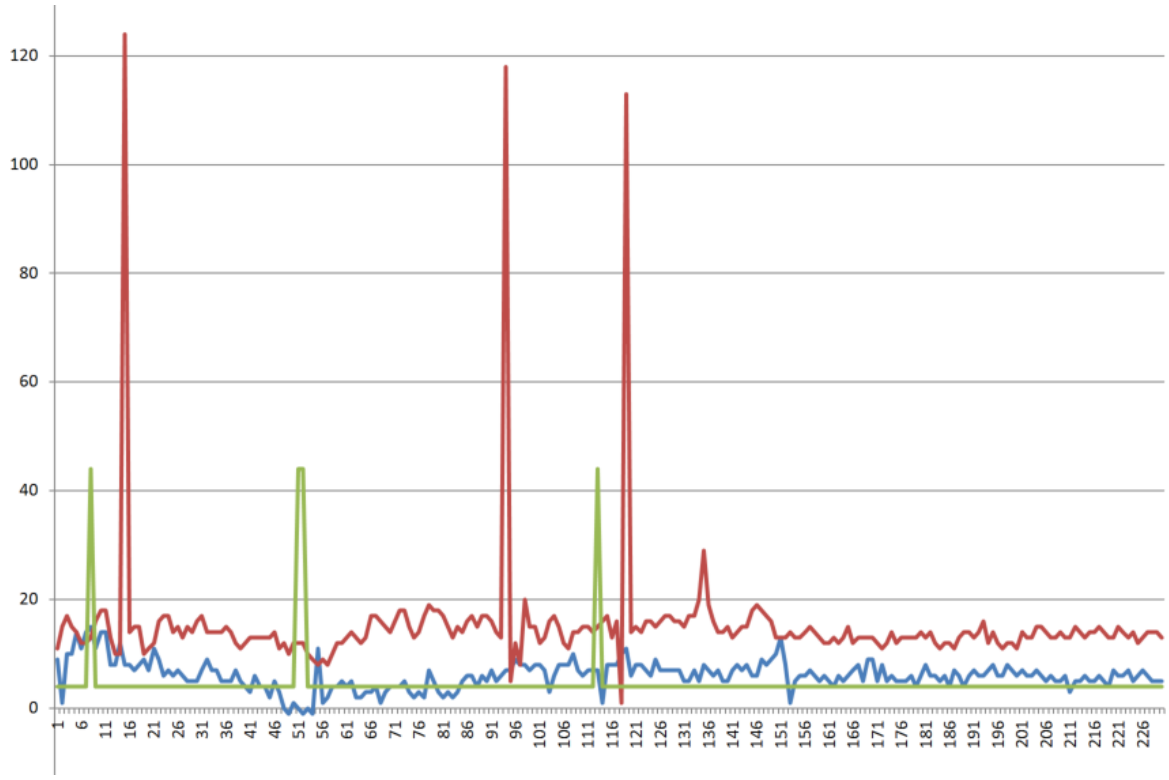
Obrázok 3 Surové dáta os y



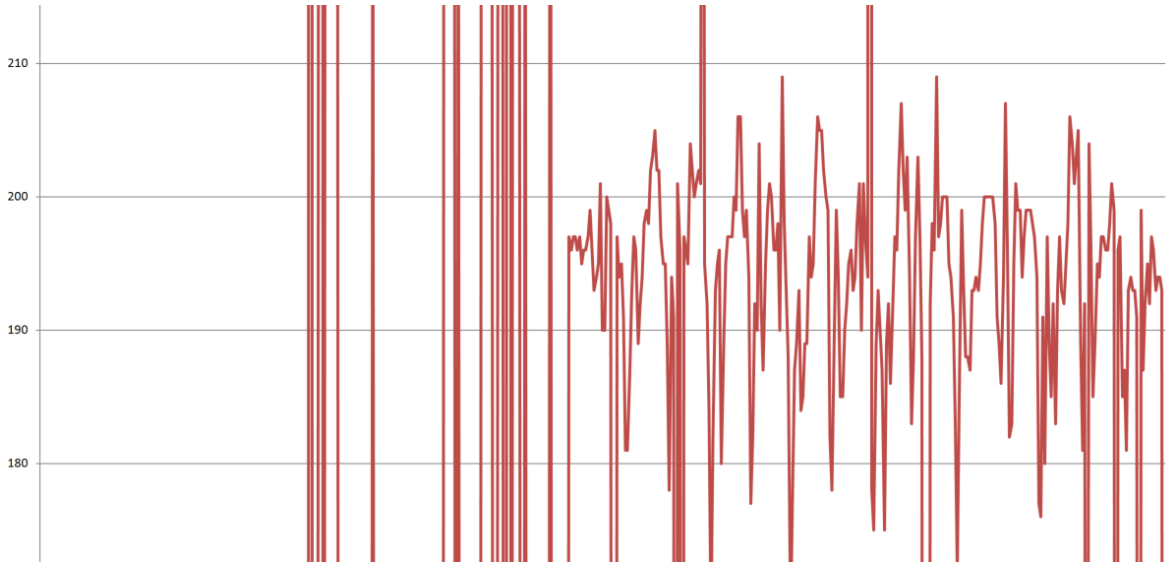
Obrázok 4 Surové dáta os x,y



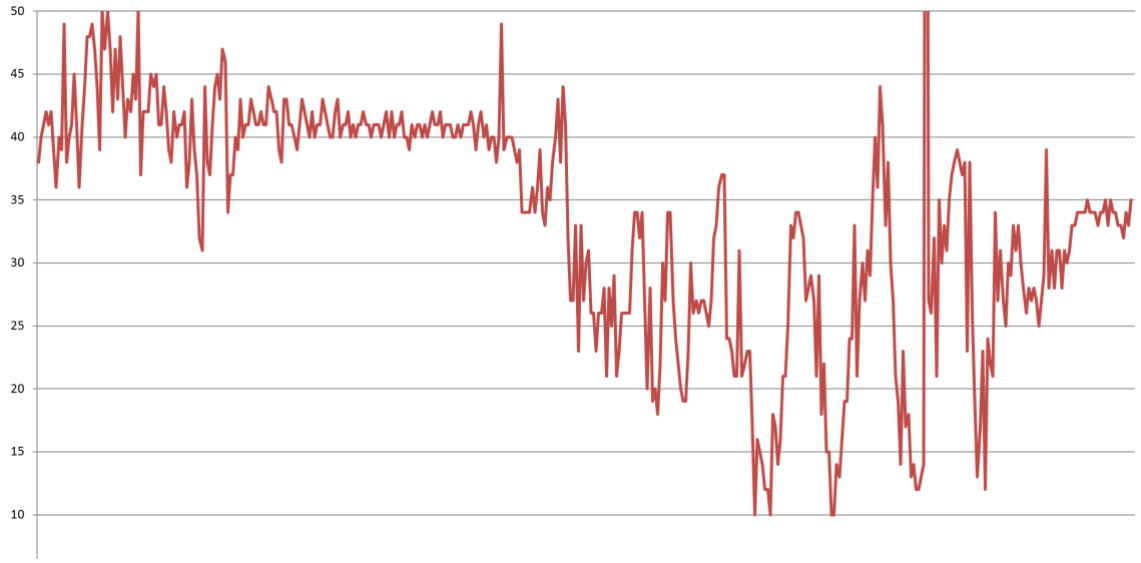
Obrázok 5 Surové dáta os y,z



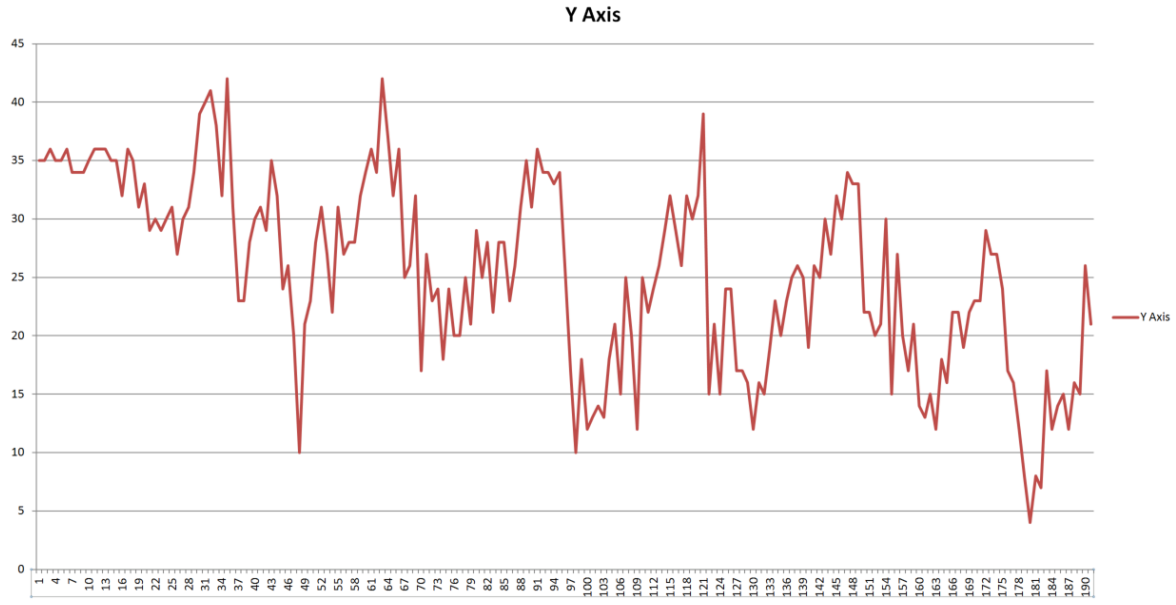
Obrázok 6 Dáta s priemerovaním 4 hodnôt, os x,y,z



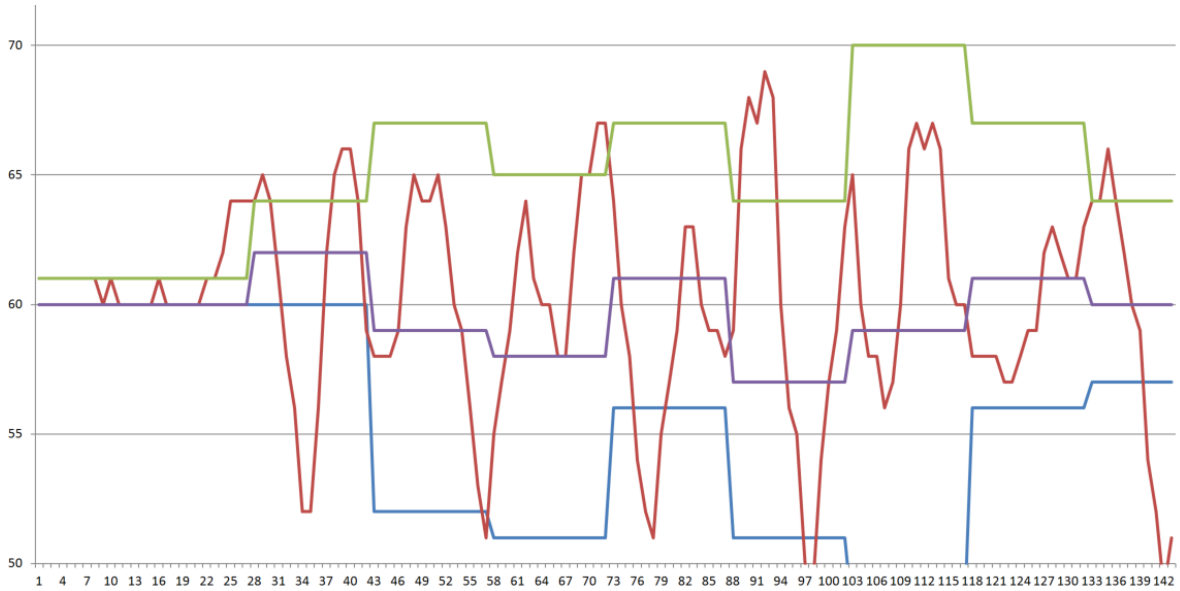
Obrázok 7 Dáta s priemerovaním 4 hodnôt, os y



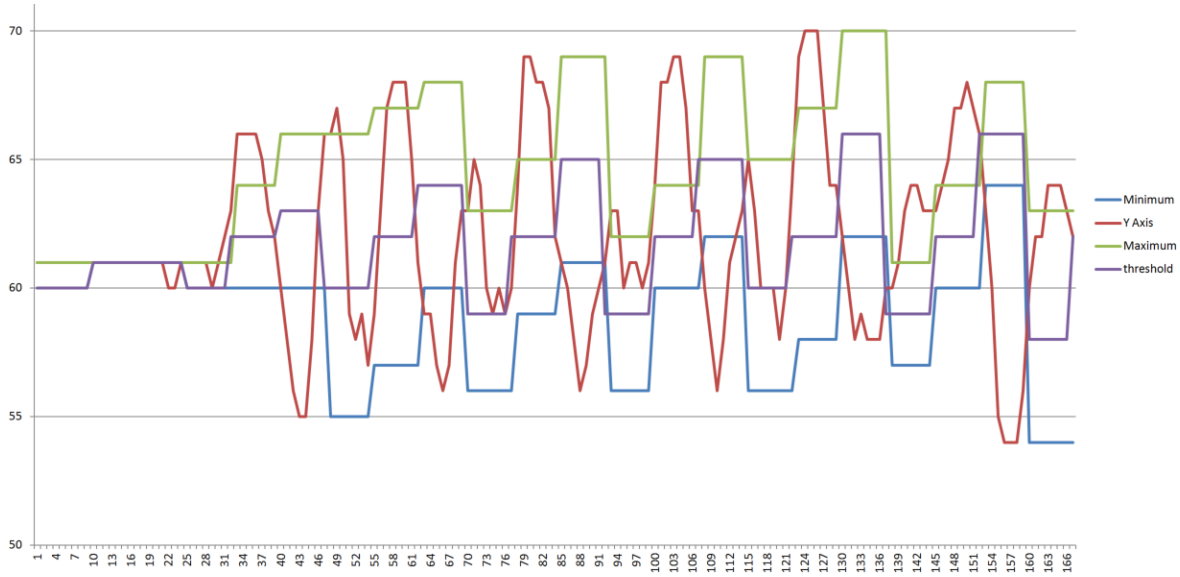
Obrázok 8 Filtrovaná os y



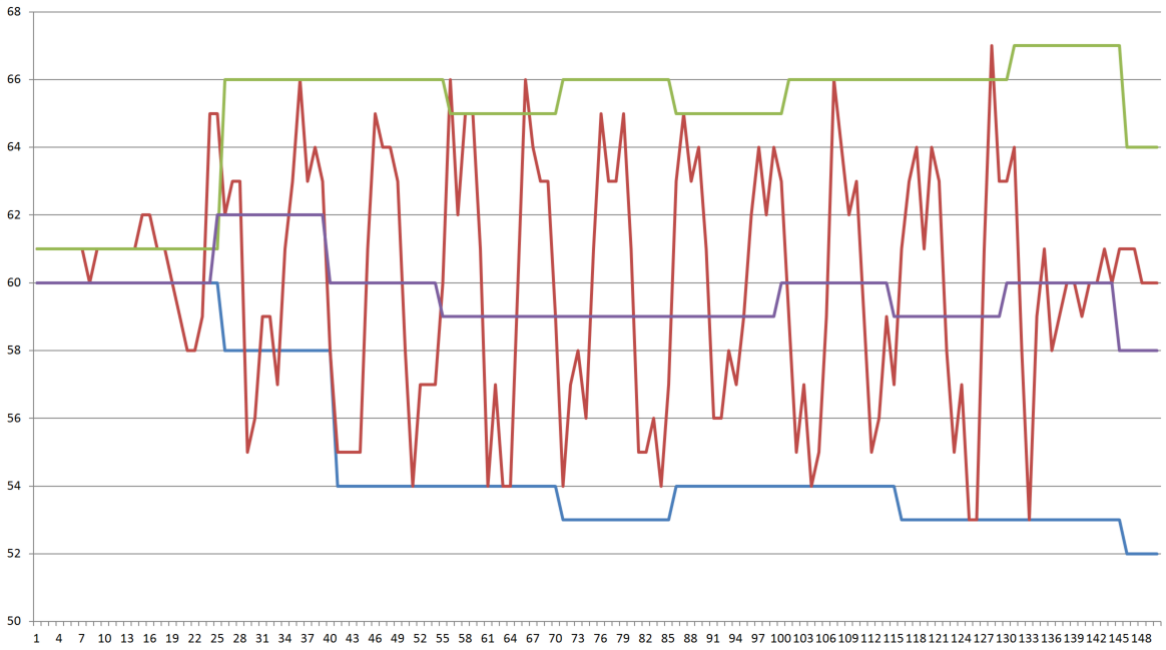
Obrázok 9 Dáta s priemerovaním 10 hodnôt, os y



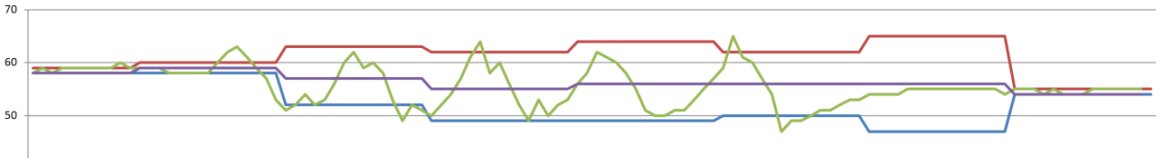
Obrázok 10 Filtrovaná os y, max,min hodnota a threshold



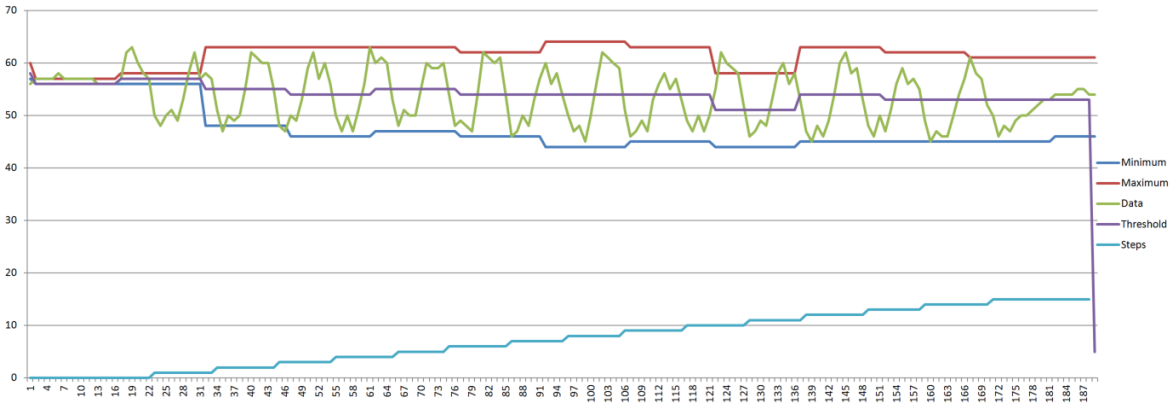
Obrázok 11 Filtrovaná os y, max,min hodnota a threshold



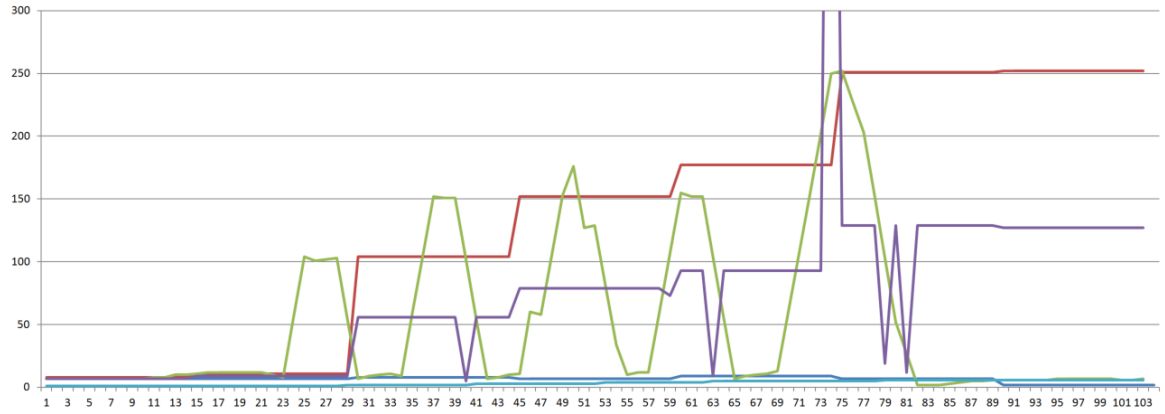
Obrázok 12 Filtrovaná os y, nová metóda na max, min hodnoty a threshold



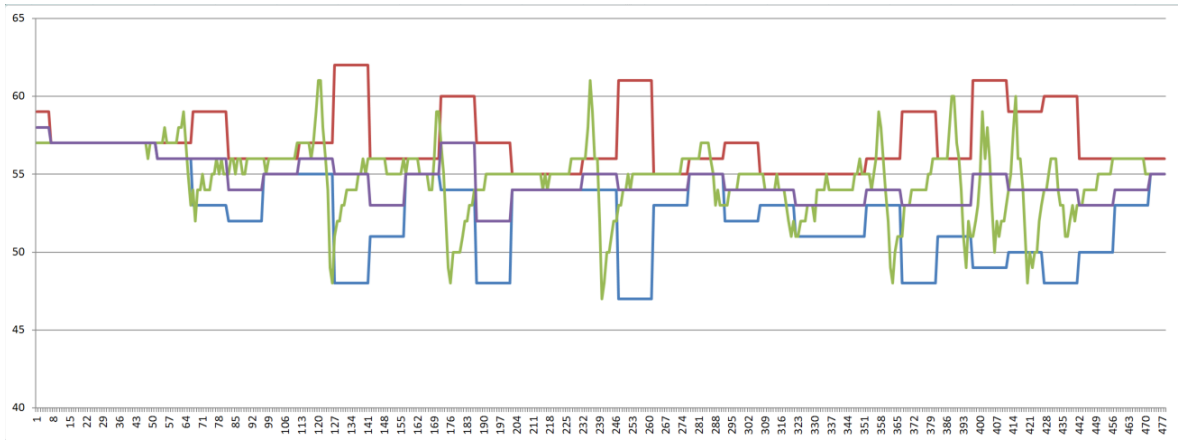
Obrázok 13 Filtrované dáta s oneskorením posielania



Obrázok 14 Filtrované dáta os y,max, min threshold, kroky



Obrázok 15 Os y vo vodorovnej polohe



Obrázok 16 Pomalé kroky a krátke kroky

Hlavná idea

Podľa hodnôt z akcelerometra chôdza pozostáva z priebehu, ktorý keby je úplne vyhladený tak sa podobá sinusu. Na krok je potrebné si určiť maximálnu, minimálnu hodnotu a hodnotu threshold čo je vlastne priemer z maximálnej a minimálnej. Takže pri chôdzi by sa hodnoty z akcelerometra mali pohybovať medzi max, min a striedavo by mala aktuálna hodnota pretnúť hodnotu thresholdu. Ak aktuálna hodnota je v dolnej oblasti (minimum – threshold), pretne threshold a dostane sa do hornej oblasti (threshold – maximum), tak sa vykoná prvá fáza kroku a to dotiahnutie zadnej nohy. Ak sa aktuálna hodnota pretne s thresholdom a prejde do spodnej oblasti z hornej, tak sa vykonala fáza dopadu na prednú nohu, čiže krok pozostáva z týchto dvoch fáz.

Implementácia

Algoritmus krokomera na STM32 L152RE

Zdrojový kód je písaný v jazyku C v Atollic TrueSTUDIO a je rozdelený do:

- *main.c*
- *header.h*
- *sources.c*

V *header.h* sa nachadza deklarácia globálnych premenných a funkcií. V *sources.c* sa nachádza implementácia funkcií deklarovaných v hlavičkovom súbore. V *main.c* sa nachádza implementácia algoritmu, ktorá je vysvetlená nižšie.

Opis funkcií

`void usart_init()` – inicializácia periférie USART

`void init_SPI1(void)` – inicializácia periférie SPI1

`void init_button()` – inicializácia periférie GPIOC

`void initialization()` – inicializácia premenných

`uint16_t getSPIdata(uint8_t adress)` – funkcia vráti hodnotu zo zadanej adresy, v našom prípade pre os y = 0x2B

`void reset_kroky()` – funkcia získa hodnotu z tlačidla, stlačené = 0 a nestlačené = 1, ak sa tlačidlo stlačilo, tak sa vynuluje premnná kroky.

`void delay(int c)` – funkcia slúži na oneskorenie v zadanom čísle c, pozostáva z jednoduchého for cyklu

`void update_min_max()` – funkcia nastaví nové maximum a minimum hodnoty podľa zbieraných hodnôt v poli `val_maxmin[]`, vo for cykle sa prejdú všetky hodnoty pola a porovnávajú sa v dvoch podmienkach pre maximum a minimum. Ak je aktuálna hodnota väčšia od uloženej, tak do uloženej sa priradí aktuálna.

`void get_Y_data()` – funkcia, ktorá získa hodnotu z osi y, následne ju pretypuje na typ `int` a skontroluje či nie je väčšia ako 255, ak hej ide o šum a do aktuálnej hodnoty sa priradí predchádzajúca hodnota `y_old`

`void shift_values()` – slúži na posunutie každej hodnoty v poli `values[]`, vo for cykle sa do hodnoty na pozícii i priradí hodnota na pozícii i+1, pričom do poslednej na pozícii 9 sa priradí hodnota z osi y.

`int get_Y_average()` – vráti vypočítaný priemer pola `values[]`

`int get_threshold()` – funkcia vráti hodnotu thresholdu, čo je vlastne priemer z maximálnej a minimálnej hodnoty osi y.

Algoritmus detekcie kroku

Ako prvé sa vykoná inicializácia všetkých premenných a periférií USART, SPI1 a GPIOC. Pomocou USART-u sú posielané dáta vo formáte ;y_min:y_max:threshold:hodntota_y:kroky;. A v desktop-ovej aplikácii sú čítané údaje vypisované na obrazovku. SPI1 slúži na komunikáciu s akcelerometrom resp. posielanie dát a periféria GPIOC na stlačenie tlačidla.

Ďalej je potrebné nastaviť akcelerometer na posielanie dát, kde na register, ktorý je na adrese 0x20 sa pošle riadiace slovo 0x67.

```
mySPI_SendData(0x20, 0x67);
```

Pred hlavným cyklom je potrebné vykonať nasledujúce kroky.

1. Získať hodnotu z osi *y*
2. Priradiť hodnotu do max, min, old, threshold, a *values[0]*
3. Cyklus while od 1 po 9 (podľa dĺžky kroka a bez prvého kroku ktorý sa už vykonal v bode 2)
4. V cykle :
    a. Získať hodnotu osi *y*
    b. Podmienka pre max, ak je aktuálna hodnota väčšia, tak do max sa priradí aktuálna
    c. To isté pre minimum osi *y*
    d. Do pola *values[i]* a *y_old* sa priradí *y*
    e. Inkrementácia *i*

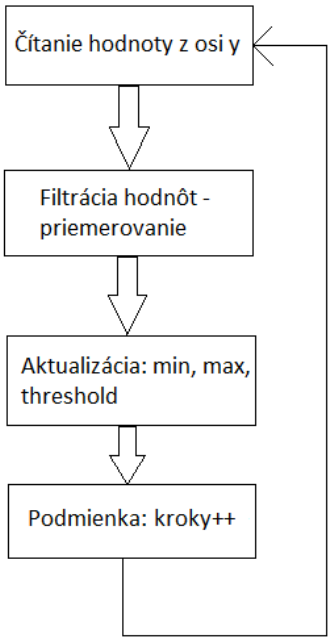
Potom sa vypočíta priemer pre *y* z pola *values*.

Ďalej Kontrola pre hodnotu min a max, ak by boli 0 alebo 255 tak sa priradí priemer do min alebo max, to je kontrola ak by sa sem dostali inicializačné hodnoty.

A výpočet thresholdu.

V hlavnom nekončenom cykle:

1. Kontrola na reset krokov
2. Čítanie hodnoty z osi *y*
3. Priemerovanie hodnôt:
    a. Funkcia *shift_values()*
    b. Do premennej *y_avg = get_Y_average()*
4. Aktualizácia hodnôt: y_max, y_min, threshold. Pre tieto hodnoty je vytvorené pole *val_maxmin[]* o veľkosti 30, v ktorom sa v každom cykle posúvajú hodnoty, ak bola pridaná posledná hodnota, tak sa zistí nové maximum, minimum a threshold.
5. Podmienka na kroky. Ak aktuálna hodnota *y_avg* je menšia ako threshold a zároveň status je 1 a zároveň platí že *y_avg* je väčšia alebo menšia o +-3, tak sa vykonal krok a premenná kroky sa zvýši o 1. Hodnota statusu sa nastaví na 1 ak hodnota aktuálnej *y_avg* je väčšia ako threshold a zároveň status je 0. Takže status 0 znamená že hodnota je v oblasti min – threshold a hodnota 1 ak je v oblasti threshold – max.
6. Odoslanie dát pomocou USART-u a použitie oneskorenia.
7. Uchovanie predchádzajúcej, resp. aktuálnej hodnoty.



Obrázok 17 Základný algoritmus krokomer

Desktopová aplikácia v C#

Aplikácia číta údaje zo sériového portu, kde je posielaný jeden údaj, počet krokov z STM, tie následne spracuje a zobrazí. Ďalej sa vypočíta vzdialenosť na základe zadanej dĺžky kroku, čiže počet krokov*dĺžka kroku. Ako ďalšie sa vypočíta priemerná rýchlosť, vzdialenosť / čas v *m/s* a násobením konštantou 3,6 dostaneme *km/h*.

Používateľská príručka

Krokometer je vnorený systém bežiaci na doske STM32L152RE, ktorý dokáže merať počet krokov pri chôdzi. Súčasťou krokomera je aj desktopová aplikácia pre zobrazenie štatistík a počtu krokov.

Použitie krokomeru :

Pre meranie hodnôt je potrebné, aby bol krokomer pripojený USB káblom k počítaču, kde je desktopová aplikácia. Ak je zariadenie pripojené k počítaču, automaticky dochádza k posielaniu údajov.

Vyresetovanie krokov sa robí pomocou stlačenia modrého USER tlačidla na STM doske. Vtedy dochádza k vyresetovaniu krokov.

Umiestnenie krokomera:

Keďže sa jedná o BETA verziu, krokomer musí byť pripojený USB káblom k počítaču, čo dosť obmedzuje v pohybu. Krokomer by mal byť správne prichytený o užívateľa alebo ruku používateľa to tak, že y – os je v zvislej polohe. Vtedy sú výsledky najpresnejšie a odchýlka krokov je minimálna.

Aplikácia:

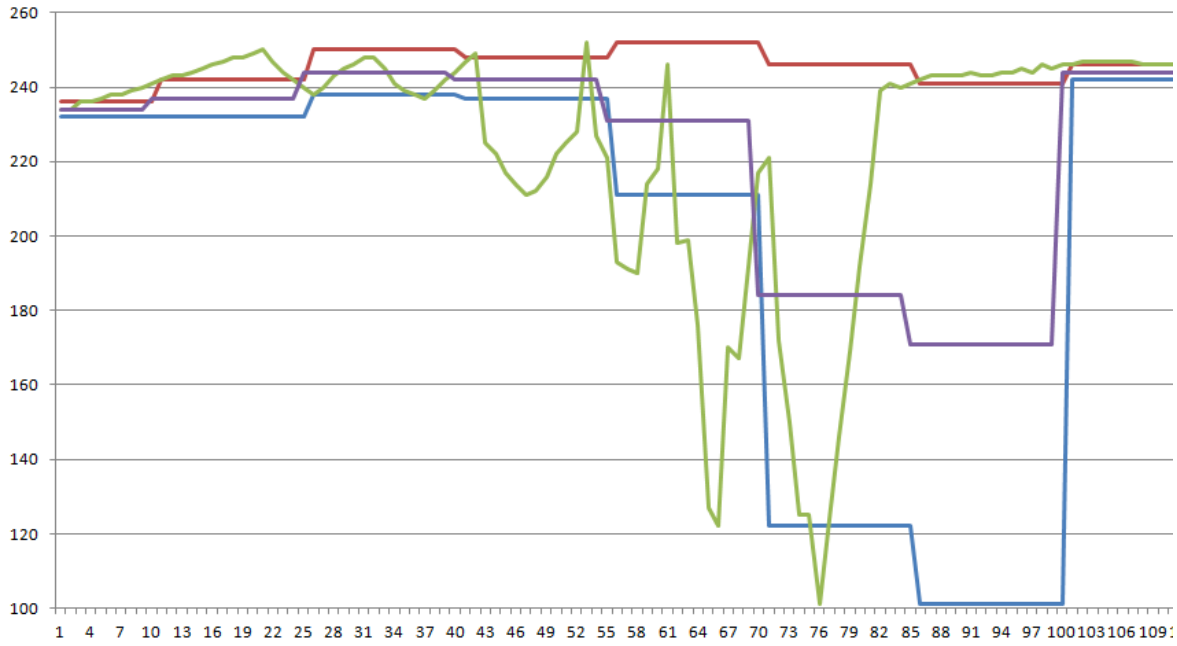
Jedná sa o jednoduchú desktopovú aplikáciu ktorá slúži na sledovanie štatistík a počet krokov, prejdených užívateľom. Dĺžku kroku je defaultne 0.8, ale je možné ju zmeniť. Aplikácia automaticky zobrazuje kroky, prejdenú vzdialenosť a priemernú rýchlosť. Je možné že pri spustení desktopovej aplikácii bude zobrazený určitý počet krokov atď, to preto že s akcelerometrom sa ľubovoľne hýbe a posiela dáta do PC, vtedy stačí resetovať STM-ko.

Záver

Dosiahnuté výsledky sú zobrazené v nasledujúcich priebehoch, ako aj v priebehoch v kapitole Analýza údajov. Obrázok 18 znázorňuje akcelerometer umiestnený:

- x-ová os vodorovne
- y-ová os vodorovne
- z-ová os zvislo

Prešlo sa 6 krokov a 6 krokov bolo detegovaných.

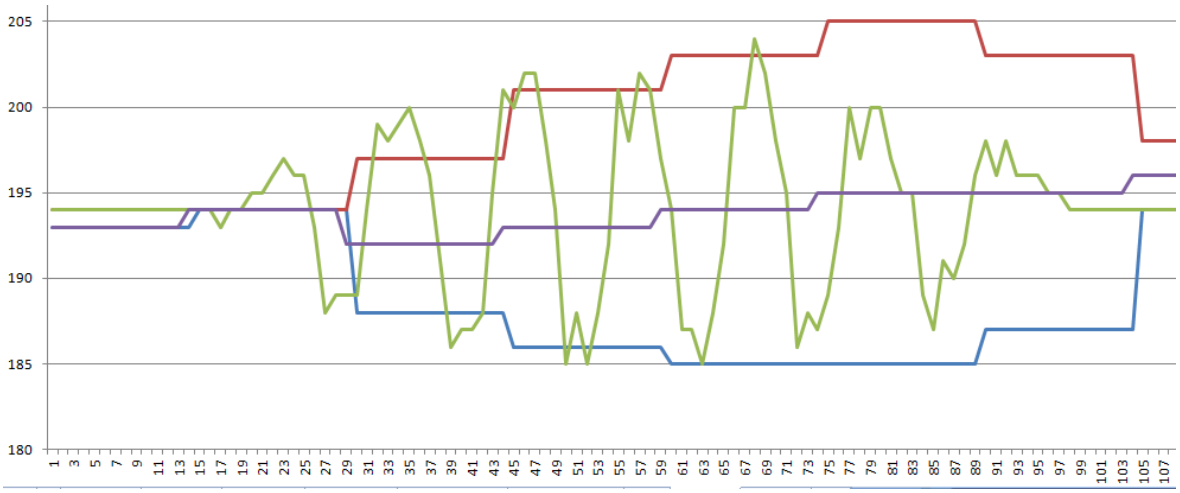


Obrázok 18 Os y vodorovne1

Obrázok 19 znázorňuje akcelerometer umiestnený:

- x-ová os vodorovne
- y-ová os zvislo
- z-ová os vodorovne

Prešlo sa 6 krokov a 6 krokov bolo detegovaných.

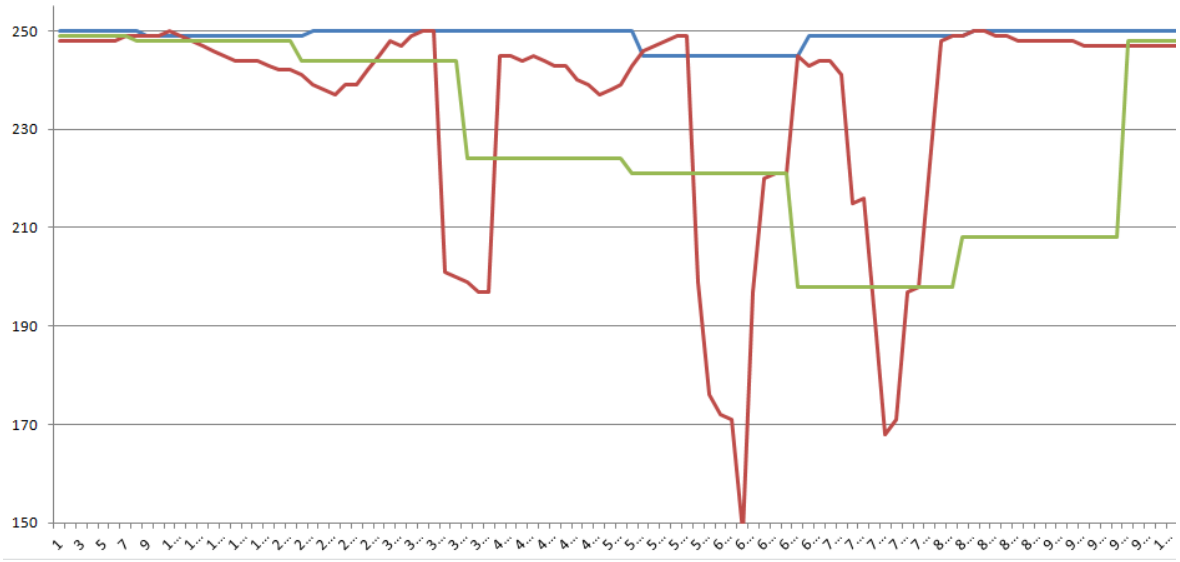


Obrázok 19 Os y zvislo

Obrázok 20 znázorňuje akcelerometer umiestnený:

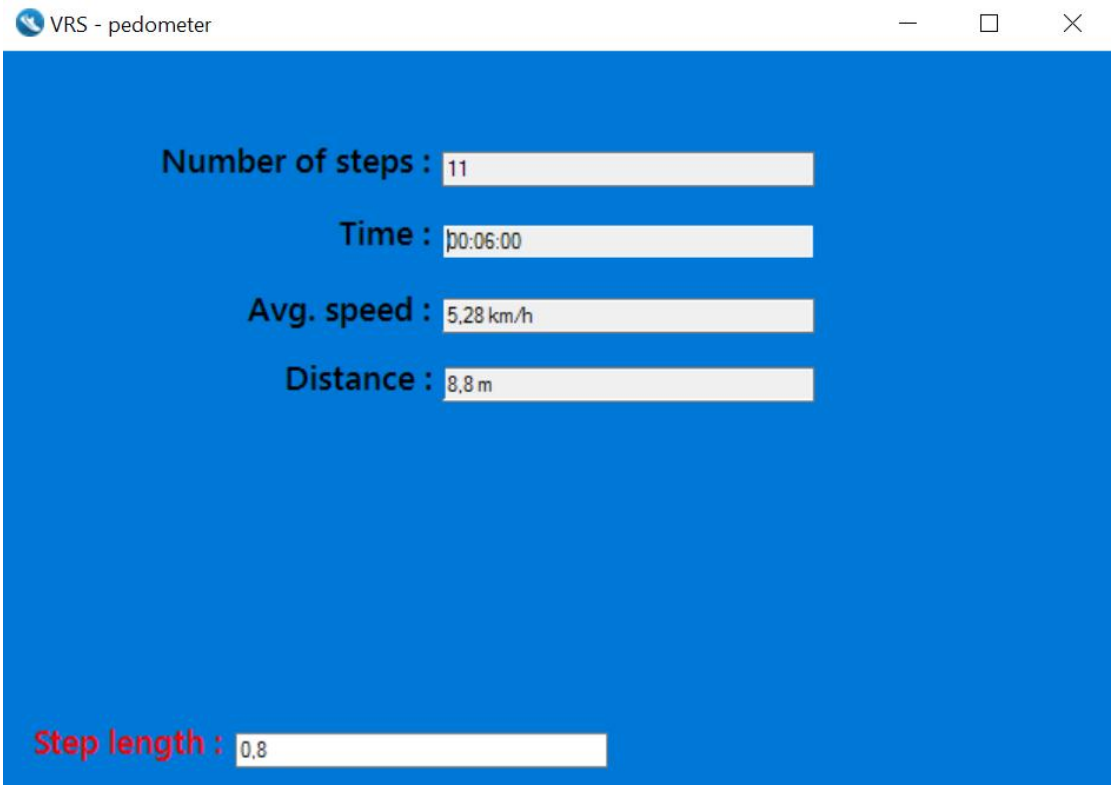
- x-ová os zvislo
- y-ová os vodorovne
- z-ová os vodorovne

Prešlo sa 5 krokov a 4 krokov bolo detegovaných.



Obrázok 20 Os y vodorovne2

Počas vypracovania sme museli zmeniť akcelerometer, nakoľko analógový akcelerometer MMA7361LC nepracoval správne a nemohli sme sa spoliehať na hodnoty z osí x,y,z. Jediný iný, ktorý sme boli schopný zohnať za krátky čas bol digitálny akcelerometer LIS3DH. Oproti analógovému, kde sa hodnoty čítajú pomocou ADC, tak je potrebné inicializovať periférium SPI1, vďaka ktorej vieme čítať hodnoty.



Obrázok 21 Desktopová aplikácia