VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2017 Matej Racek


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# SYSTÉM SLEDOVÁNÍ POHYBU A SPÁNKU POMOCÍ PEBBLE TIME SMARTWATCH

GAIT AND SLEEP MONITORING SYSTEM USING PEBBLE TIME SMARTWATCH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE
AUTHOR

Matej Racek

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jiří Mekyska, Ph.D.



BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

***Student:*** Matej Racek ***ID:*** 164380
***Ročník:*** 3 ***Akademický rok:*** 2016/17

**NÁZEV TÉMATU:**

## Systém sledování pohybu a spánku pomocí Pebble Time Smartwatch

**POKYNY PRO VYPRACOVÁNÍ:**

V rámci této práce bude pro prostředí iOS či Android vytvořena aplikace sledující pohyb a spánek. K tomu budou využita data z akcelerometru chytrých hodinek Pebble Time Smartwatch. Data bude rovněž možné exportovat v CSV formátu.

**DOPORUČENÁ LITERATURA:**

[1] SILVIA DE LIMA, AL.; HAHN, T.; VRIES, NM.; et al. Large-Scale Wearable Sensor Deployment in Parkinson's Patients: The Parkinson@Home Study Protocol. JMIR Res Protoc, 2016, roč. 5, č. 3, s. e172.

[2] ZHAN, A.; LITTLE, M.; HARRIS, D.; et al. High Frequency Remote Monitoring of Parkinson's Disease via Smartphone: Platform Overview and Medication Response Detection. CoRR abs/1601.00960, 2016.

***Termín zadání:*** 1.2.2017 ***Termín odevzdání:*** 8.6.2017

***Vedoucí práce:*** Ing. Jiří Mekyska, Ph.D.
***Konzultant:***

**doc. Ing. Jiří Mišurec, CSc.**
*předseda oborové rady*

## ABSTRAKT

Táto práca sa zaoberá sledovaním pohybu a spánku pomocou inteligentných hodiniek Pebble Time. V rámci tejto práce je vyvinutá aplikácia pre prostredie Android. Teoretická časť sa zo začiatku zaoberá rešeršou na danú problematiku, porovnaním riešení, základnou terminológiou. Ďalej sú v práci popísané samotné inteligentné hodinky Pebble, Pebble SDK, možnosť vývoju v jeho prostredí a taktiež využitie dát z akcelerometra. Praktická časť sa venuje návrhu, následnej implementácie s ukážkami kódu a testovaniu systému pre sledovanie spánku a pohybu.



## KĽÚČOVÉ SLOVÁ

Pebble, Android, SDK, akcelerometer



## ABSTRACT

This thesis deals with Gait and sleep monitoring system using Pebble Time Smartwatch. As part of this work, 2 Android applications are developed. The theoretical part deals with research on the given issue, comparison of solutions, basic terminology. Furthermore, Pebble, Pebble SDK, the possibility of development in its environment as well as the use of accelerometer data are described in the work. The practical part deals with design, implementation with samples of code and testing of the sleep and movement monitoring system.



## KEYWORDS

Pebble, Android, SDK, accelerometer



RACEK, Matej. *Systém sledování pohybu a spánku pomocí Pebble Time Smartwatch*. Brno, 2017, 54 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Jiří Mekyska, Ph.D.

# VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Systém sledování pohybu a spánku pomocí Pebble Time Smartwatch“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ..... .....

podpis autora

## POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi Ing. Jiřímu Mekyskovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.


Brno ..... .....

podpis autora


Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

## POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ..... .....

podpis autora


EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

# OBSAH

# ZOZNAM OBRÁZKOV

# ZOZNAM TABULIEK

# ZOZNAM VÝPISOV

# ÚVOD

Technológia nositeľných zariadení je neoddeliteľnou súčasťou riešenia pre poskytovanie zdravotnej starostlivosti rastúcej svetovej populácie. Poskytnutím prostriedkov na vykonávanie telemedicíny tj. monitorovanie, zaznamenávanie a prenos fyziologických signálov zo strany pacientov, by mohli zmierniť zaťaženie zdravotníckeho personálu a využívanie priestorov v nemocniciach pre vyššiu a rýchlejšiu starostlivosť. Navyše použitie nositeľných technológií v profesiách, kde sú pracovníci vystavení nebezpečenstvu, by mohlo pomôcť zachrániť životy a ochrániť zdravotnícky personál [15].

Náplňou bakalárskej práce bolo oboznámiť sa s Pebble SDK a naprogramovať aplikáciu pre monitorovanie spánku a pohybu. Rešerš na danú problematiku s porovnaním iných riešení je popisovaný v kapitole 1.

Kapitola 2 sa zaoberá stručným popisom hodiniek Pebble Time, Pebble SDK, jeho vývojom, získavaním a využitím dát z akcelerometra taktiež možnosťou záznamu dát z akcelerometra a možnosti logovania. V samostatnej 3 kapitole je popísané navrhnutie systému sledovania pohybu a spánku kde je na tento návrh vytvorený do blokových diagramov s popisom jednotlivých blokov a procesov, takisto aj spracovanie surových dát a exportovanie z aplikácie.

Nasledujúca kapitola 4 je už úvodom k samotnému praktickému riešenie a zároveň obohacuje predchádzajúcu o vizuálny výsledok implementácie prvotného návrhu ako aj ukážok hlavnej častí kódu príslušnej aplikácie. Testovaním funkčnosti systémov sa zaoberá 5 kapitola kde sa bude rozoberať zvlášť testovanie pri pohybe ruky a pohybe počas spánku. Posledná 6 kapitola je vlastne vyhodnotením dosiahnutých výsledkov testovania.

# 1 TEORETICKÁ ČASŤ PRÁCE

## 1.1 Nositeľné zariadenia v medicíne

Nositeľné technológie sú základným kameňom medicínskej informatiky, celkové využitie nositeľných biosenzorov a ich aplikácie sú stále ešte predmetom viacerých štúdií [1]. Tieto biosenzory sú k dispozícii v rôznych formách, ako sú napríklad klobúky, košele, prstene, pásy, náramky, topánky, ponožky, okuliare, kontaktné šošovky, náhrdelníky a hodinky.

## 1.2 Základná terminológia

Pre lepšie chápanie danej problematiky a špecifických termínov je potrebné ozrejmiť si základnú terminológiu.

**API (Application programming interface)** – Ide o zbierku funkcií a tried (ale aj iných programov), ktoré určujú akým spôsobom sa majú funkcie knižníc volať zo zdrojového kódu programu.

**UI (User interface)** – Jedná sa o programy a zariadenia, ktoré sú k dispozícii používateľovi systému na spracovanie dát. Používateľské rozhranie definuje tú časť systému, ktorú môže používateľ používať.

**SDK (Software development kit)** – Jedná sa o súbor nástrojov pre vývoj softvéru, ktorý umožňuje vytváranie aplikácií pre určitú softvérovú platformu, hardvérovu platformu, herné konzoly, operačný systém, alebo iné platformy.

**IDE (Integrated development environment)** – je softvérový balík, ktorý konsoliduje základné nástroje, ktoré vývojári potrebujú na písanie a testovanie softvéru.

**FreeRTOS** – FreeRTOS je trieda RTOS (real-time operating system), ktorá je navrhnutá tak, aby bola dostatočne malá na to, aby fungovala na mikrokontroléroch - hoci jej použitie nie je obmedzené na aplikácie mikrokontrolérov [6].

**Framework** – Je softvérová štruktúra, ktorá slúži ako podpora pri programovaní a vývoji a organizácií iných softwarových projektov. Môže obsahovať podporné programy, knihovne API, podporu pre návrhové vzory alebo doporučené postupy pri vývoji.

**Realm DB** – Open-source objektový systém správy databáz, dostupný pre využitie na viacerých distribučných platformách mobilných zariadení (Android, iOS, Windows ...).

**UUID (Universally unique identifier)** – Jedná sa o 128-bitové číslo použité na identifikáciu informácií v počítačových systémoch.

**Buffer** – Oblasť úložiska fyzickej pamäte, ktorá sa používa na dočasné ukladanie údajov.

**URI (Uniform resource identifier)** – Kompaktný reťazec znakov používaný na identifikáciu alebo pomenovanie zdroja dát.

**Watchapp** – Aplikácia bežiaca až po spustení na inteligentných hodinkách.

**Watchface** – Aplikácia na inteligentných hodinkách bežiaca na popredí.

## 1.3 Systémy sledujúce životné funkcie

### 1.3.1 Veľkokapacitné nasadenie senzorov u pacientov s Parkinsonovou chorobou

Dvojfázová pozorovacia štúdia zahŕňajúca 1000 pacientov s Parkinsonovou chorobou a 250 fyzioterapeutov. Stav ochorenia sa hodnotí pomocou Parkinsonovej progresívnej stupnice, ktorú vykonávajú certifikovaní fyzioterapeuti. Účastníci nosia sadu senzorov (smartwatch, smartfón a detektor pádu) a používajú ich spolu s prispôsobenou aplikáciou pre smartfón (Fox Insight) 24 hodín denne, 7 dní v týždni počas 3 mesiacov. Senzory zabudované v inteligentných hodinkách a detektore pádu sa môžu použiť na odhad fyzickej aktivity, tremoru, kvality spánku a pádov. Príznaky pádu pri príjme liekov sa merajú prostredníctvom vlastných správ pacientov v aplikácií na mobile. Prvá fáza sa zameria na uskutočniteľnosť študijného protokolu. V druhej fáze matematici zanalyzujú príslušné súhrnné štatistické údaje zo surových dát nameraných senzormi, ktoré sa porovnajú s klinickými výsledkami [15].

#### Aplikácia Fox Insight

Aplikácia Fox Insight je vytvorená a vyvinutá spoločnosťou Intel Corp. pre mobilné zariadenia Android a iPhone. Táto aplikácia získava 50 záznamov za sekundu z akcelerometra z hodiniek Pebble a odhaduje úrovne aktivity, tremor a analýzu pohybu spánku pomocou špeciálnych algoritmov spustených v aplikácii. Aplikácia zobrazuje tieto odhadované množstvá používateľovi pomocou grafov a súhrnných správ o zhromaždených údajoch [15].

### 1.3.2 Monitorovanie dennej aktivity pomocou nositeľných zariadení

Testovanie zahŕňa 40 ľudí pričom každý nosil deväť zariadení počas 24 hodín: Actigraph GT3X +, activPAL, Fitbit One, GENEactiv, Jawbone Up, LUMOback,

Nike Fuelband, Omron krokomer a Z-Machine. Analýza zahŕňa strednú absolútnu percentuálnu chybu a testy ekvivalencie [14].

24 hodinový cyklus je rozdelený do týchto aktivít:

- Spánok.
- Sedenie.
- Mierna fyzická aktivita.
- Stredná až silná fyzická aktivita.

**Prehľad zariadení**

| **Zariadenie** | **Softvér** |
|---|---|
| Z-Machine | Z-machine Data Viewer |
| activPALvt | ActivPAL3 v7.1.18 |
| GT3X+ | Actilife 6 |
| HJ-112 Pocket Pedometer | Zobrazenie na displeji |
| Fitbit One | iPhone aplikácia synchronizovaná s PC |
| GENEactiv Original | GENEactiv PC softvér |
| Jawbone UP | iPhone aplikácia |
| LUMOback | iPhone aplikácia |
| Fuelband | iPhone aplikácia synchronizovaná s PC |

Tab. 1.1: Prehľad softvéru použitých zariadení.

### 1.3.3 Nepretržité sledovanie tremoru pomocou prenosného systému založeného na Smartwatch

Nový vysoko prenosný systém bol používaný na sledovanie tremoru nepretržite počas každodenného života. Skladá sa z inteligentných hodiniek , smarfónu a vzdialeného servera. Bol uskutočnený experiment zahŕňajúci osem pacientov s ET (Essential Tremor). Priemerná účinná doba zberu údajov na pacienta bola 26 (± 6,05) hodín. Hodnotiaca stupnica Fahn-Tolosa-Marin Tremor (FTMTRS) bola prijatá ako zlatý štandard na klasifikáciu tremoru a validáciu výkonnosti systému. Kvantitatívna analýza závažnosti tremoru v rôznych časových intervaloch je validovaná [16].

**Monitorovací systém**

Tento trojvrstvový systém sa skladá z Pebble Smartwatch, ktorý obsahuje trojosový akcelerometer a Bluetooth 4.0 na zaznamenávanie údajov o pohybe ruky uží-

vateľa, smartfón s OS android na príjem dát z Pebble a ich uploadovanie na vzdialený server, multi-platformovú databázu NoSQL MongoDB2 na vzdialenom serveri, na ukladanie a analýzu dát. Informácie zhromaždené pomocou tohto systému zahŕňajú hodnoty zrýchlenia z akcelerometra [16].

# 2 PEBBLE TIME SMARTWATCH

Pebble používa procesor architektúry ARM Cortex-M3 ako svoj CPU a prevádzkuje vlastný Pebble OS, čo je spoločnosťou Pebble upravená verzia FreeRTOS [11]. To sa líši od jeho konkurentov, ako napríklad Samsung Galaxy Gear 3 a Sony SmartWatch 2, v tom, že nie je samostatné zariadenie a preto vyžaduje spárovanie s mobilným telefónom v rámci Bluetooth [2].

Obr. 2.1: Pebble Time

Spracovanie údajov nie je vykonané na hodinkách Pebble, ale poslané na spárovaný smartfón [3]. Tento systém spárovania hodiniek so smartfónom je veľmi šetrný pre baterku Pebble, ktorú je potrebné nabiť len raz za niekoľko dní [3]. Okrem toho Pebble používa monochromatický displej, so snímačom okolitého svetla, ktorý reguluje jas obrazovky v závislosti od svetelných podmienok [3]. Napriek použitiu troj-osového akcelerometra a magnetometra, ktorý poskytuje vstupné údaje založené na okolitom prostredí, Pebble stále dokáže udržať energetickú efektivitu [4].

Používanie operačného systému, ktorý závisí od niekoľkých hardvérových zdrojov a vonkajšieho dizajnu, obsahujúci štyri tlačidlá na interakciu medzi človekom a počítačom, znamená že interakcia používateľa s Pebble je jednoduchá a intuitívna [8]. Pebble predstavuje multiprogramovateľnú platformu, na ktorej písané aplikácie na platforme android môžu byť spustené. Toto univerzálne použiteľné počítačové riešenie obsahuje veľa výhod nositeľného počítača, ako je pohodlie a vždy prístupné vyhľadávanie údajov a zároveň poskytuje stabilnejšie prostredie v porovnaní s inými inteligentnými hodinkami ako je spotreba energie a komplikované užívateľské rozhrania.

Nutnosťou pre vývoj Pebble Smartwatch je Pebble SDK (PebbleKit), ktorý je voľne dostuptný pre inštaláciu na Linux a MacOS X. Pebble taktiež poskytuje webový vývojový framework prostredníctvom CloudPebble dostupný pre Windows [5].

## 2.1 Pebble SDK

Pebble SDK sa skladá z viacerých frameworkov, ktoré sú organizované ich funkciami. Každý framework zahŕňa aplikačné rozhranie a poskytuje prístup do softvérových knižníc podporovaných operačným systémom Pebble.

### 2.1.1 Pebble Smartwatch API

Pebble smartwatch API dovoľuje vývojárom rozvíjať aplikácikácie bežiacich priamo na hodinkách. Delí sa na:

Pebble C API

Používa sa na vytváranie watchapps a watchfaces v jazyku C. Pebble C API sa môže použiť v kombinácií s každým PebbleKit API pre rozšírenie funkčnosti aplikácie. C SDK je delené do týchto 6 hlavných modulov:

- Foundation.
- Graphics.
- User Interface.
- Smartstrap.
- Worker.

Pebble JavaScript API

Vstavané JavaScript API dovoľuje vývojárom písať watchfaces v JavaScripte spustiteľné cez JerryScript engine. Skladá sa z týchto modulov:

- CanvasRenderingContext2D.
- Console.
- Date.
- Rocky.

### 2.1.2 PebbleKit API

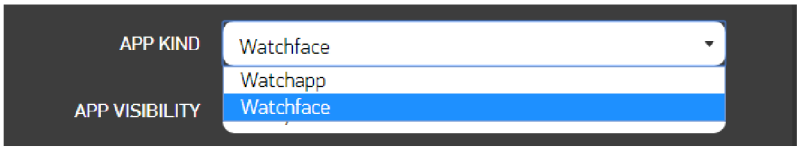
PebbleKit API ponúka možnosť rozšírenia funkcionality aplikácie na inteligentných hodinkách cez komunikáciu s aplikáciou nainštalovanou na mobilnom zariadení, zamerané na **PebbleKit JS**, **PebbleKit iOS** a **PebbleKit Android**. Záleží už len na vývojárovi ktoré z aplikačných rozhraní mu je najbližšie.

## 2.2 Možnosti vývoju

V súčasnosť sú dostupné 2 možnosti pre vývoj aplikácií Pebble a to sú:

1. Inštalácia Pebble SDK priamo na OS Linux alebo Mac OS.
2. Používanie webového CloudPebble IDE.

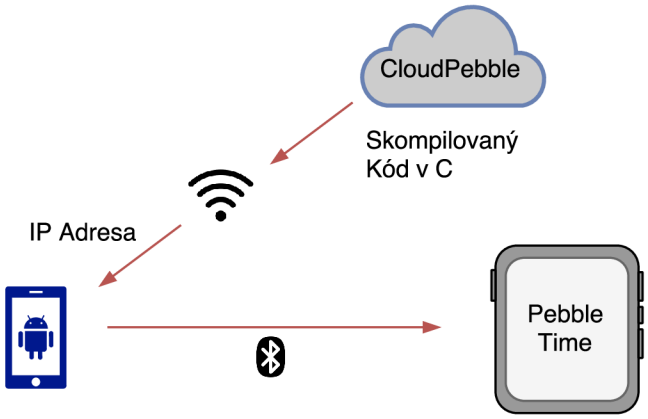
Pebble aplikácie sú rozdelené do dvoch kategórií a to watchfaces a watchapps viz obr.2.2 [9]. Watchfaces sú vyvinuté s úmyslom spustenia na predĺžené časové obdobie, v ktorom sa informácie aktualizujú v pravidelných časových intervaloch a to buď v minútach alebo sekundách, zobrazujúce sa užívateľovi. [9] Watchfaces nemôžu reagovať na interakciu používateľa prostredníctvom vstavaných tlačidiel iným než tlačidla späť, alebo pomocou senzorov hodiniek. Sú preto vyvinuté tak, aby pracovali s najmenším množstvom využitia systému. [10] Watchapps poskytuje používateľom viac funkcií ako watchfaces a sú schopné reagovať na interakcie používateľa pomocou tlačidiel. Informácie, ktoré sledujú zobrazovanie aplikácií, je možné aktualizovať aj častejšie, a sú schopné reagovať oveľa rýchlejšie na interakcie užívateľa, ako sú watchfaces [9]. Watchapps umožňuje programátorovi použiť viac systémových prostriedkov ako je potrebné. V programe CloudPebble IDE programátor dokáže určiť, či chce vyvíjať watchface alebo watchapp.




Obr. 2.2: Výber typu aplikácie v CloudPebble IDE.

Pre aplikácie inteligentných hodiniek, ktoré potrebujú komunikovať so spárovaným telefónom, musia byť inicializované v kóde priamo na smartfóne pre spoluprácu s Pebble. Toto sa dosiahne použitím open source knižnice PebbleKit, ktorá je k dispozícii od Pebble [10]. Knižnica PebbleKit je dostupná pre Apple iOS a mobilných operačných systémov Android a je implementovaná v programovacích prostrediach C a Java. Natívna aplikácia Pebble môže byť nainštalovaná na smartfón, a je k dispozícii v obchode Google Play. Spravuje akýkoľvek PebbleKit kód vygenerovaný aplikáciou, ktorá sa pokúša komunikovať s hodinkami [9].

Watchfaces a watchapps, ktoré sú vyvýjané v IDE CloudPebble, sú odosielané do telefónu pomocou WiFi. Obslužné rozhranie CloudPebble umožňuje zadanie IP adresy telefónu v aktuálnej sieti WiFi. Program písaný v jazyku C, ktorý je zostavený v prostredí CloudPebble IDE, je odoslaný na zadanú adresu IP cez sieť WiFi, kde nátivná aplikácia Pebble zachycuje a balí ich do súboru .pbw, ktorý sa odošle na inteligentné hodinky. Súbor .pbw sa pošle do hodiniek cez Bluetooth, kde sa aj aplikuje.




Obr. 2.3: Vizuálne zobrazenie ako je watchface / app poslaná z prostredia CloudPebble do hodiniek.

Nasledujúca časť sa bude zaoberať prácou s integrovaným akcelerometrom na hodinkách Pebble Time, ktorý je hlavnou súčasťou pri navrhovaní a vyvíjaní aplikácie pre monitorovanie pohybu a spánku.

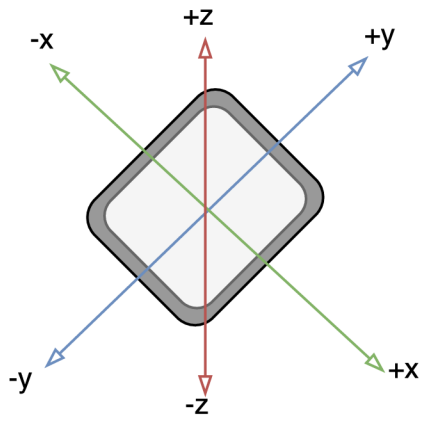
## 2.3 Získavanie dát z akcelerometra

Senzor akcelerometra obsahujú každé inteligentné hodinky Pebble. Umožňuje zhromažďovanie údajov o zrýchlení a orientácii v watchapps a watchfaces. Dáta sú k dispozícií dvomi spôsobmi, z ktorých každý je vhodný pre rôzne typy watchapp:

- Tap events (Udalosti dotyku) - Spustí udalosť, kedykoľvek dôjde k významnému zaklepaniu alebo zatraseniu hodiniek.
- Data batches (Dávky dát) - Umožňuje zhromažďovanie dávok údajov v určitých intervaloch. Veľmi užitočné pre zber údajov z akcelerometra.

Ako významný zdroj pravidelných spätných volaní by mal byť akcelerometer, použitý čo najmenej, aby sa Pebble hodinky mohli prepnúť do režimu `sleep` a šetriť energiu. Napríklad prijímanie údajov v dávkach raz za sekundu je energeticky šetrnejšie ako prijímanie jedinej vzorky 25 krát za sekundu [12].

Pebble akcelerometer je orientovaný podľa následujúceho diagramu:




Obr. 2.4: Zobrazenie smeru osi x, y a z akcelerometra.

V API sa každá hodnota osi obsiahnutá vo vzorke `AccelData` meria v milli-Gs. Akcelerometer je kalibrovaný na meranie maximálneho zrýchlenia ± 4G. Rozsah možných hodnôt pre každú os je preto -4000 až +4000.

Objekt `AccelData` obsahuje pole `did_vibrate`, nastavené na hodnotu `true` ak bol vibračný motor aktívny počas odberu vzoriek. To by mohlo tieto vzorky pozmeniť kvôli vibráciám na hodinkách, a mali by byť zahodené. Pole `timestamp` umožňuje sledovanie získaných údajov akcelerometra v čase.

### 2.3.1 Používanie odberu udalostí dotyku

Pri použití odberu udalostí, vývojár dokáže reagovať na dotyk a zatrasenie hodiniek po každej z 3 osí. Tieto udalostí sú príjímané po zaregistrovaní funkcie `AccelTapHandler`:

```
1
2 static void accel_tap_handler(AccelAxisType axis, int32_t
3 direction) {
4   // Vznik udalosťi
5 }
```

Výpis 2.1: Funkcia AccelTapHandler.

Parameter `axis` opisuje, po ktorej osi bola detekovaná udalosť. Parameter `direction` je nastavený na 1 pre pozitívny smer a $-1$ pre negatívny smer.

Zaregistrovanie je možné kedykoľvek pridať alebo odstrániť. Po zaregistrovaní bude funkcia `accel_tap_handler` volaná vždy, keď nastane udalosť.

```
1
2 // Zaregistrovanie odberu udalostí
3 accel_tap_service_subscribe(accel_tap_handler);
4
5 // Odregistrovanie odberu udalostí
6 accel_tap_service_unsubscribe();
7 }
```

Výpis 2.2: Registrácia odberu udalostí.

### 2.3.2 Používanie dávky dát

Dáta z akcelerometru dokážu byť príjmané aj ako jedna várka dát za určitej vzorkovacej frekvencie po zaregistrovaní do služby `accel_data_service`:

```
1
2 uint32_t num_samples = 3;  // Počet vzorkov pre jednu dávku
3
4 // Zaregistrovanie do odberu dávky dát
5 accel_data_service_subscribe(num_samples, accel_data_handler);
6 }
```

Výpis 2.3: Registrovanie dávky dát.

Funkcia `accel_data_service` je volaná, hneď ako je nová dávka údajov pripravená na použitie pre watchapp. Miera ich výskytu je diktovaná dvoma nastaveniami:

- Vzorkovacia frekvencia - počet vzoriek, ktoré sú merané pomocou akcelerometra za sekundu.
- Počtom vzoriek na dávku.

## 2.4 Možnosť logovania dát

Pebble SDK tiež obsahuje rozhranie Datalogging API. Čo je veľmi užitočné pre aplikácie, u ktorých je možné odosielať dáta v dávkach v časových intervaloch. Taktiež umožňuje ukladanie dát do buffera o veľkosti 640 kB, pokiaľ nie je k dispozícii pripojenie k smartfónu.

Ak sa zaznamenávajú údaje počas odpojenia hodiniek, prenesú sa na natívnu mobilnú aplikáciu Pebble v dávkach na spracovanie pri ďalšom pripojení. Údaje sa

potom prenášajú na akúkoľvek aplikáciu PebbleKit pre OS Android alebo PebbleKit iOS, ktorá ju chce spracovať. Týmto spôsobom sa dá efektívne ušetriť na výdrži batérie hodiniek kedže bude v stave spánku [13].

### 2.4.1 Zberanie dát

Datalogging dokáže zachytiť všetky hodnoty, ktoré sú kompatibilné s jednou z hodnôt `byte_array`, `unsigned_integer` a `integer`, ktoré majú rovnaký zdroj dát, vrátane dát z akcelerometra alebo magnetometra.

#### Zahájenie spojenia

Dáta sa zaznamenávajú do relácie s jedinečným identifikátorom alebo značkou, ktorá umožňuje, aby jedna aplikácia obsahovala viacero typov logovania dát o údajoch pre rôzne typy údajov. Najprv sa definujte identifikátor, ktorý sa má použiť:

```
1 // ID logu
2 #define TIMESTAMP_LOG 1
```

Pred tým než sa dáta po prvykrát zalogujú musí byť najprv vytvorené spojenie. Buď počas inicializácie aplikácie alebo tesne pred prvým ukladaním informácií:

```
1 static void init() {
2   // Zahájenie spojenia
3   s_session_ref = data_logging_create(TIMESTAMP_LOG,
4   DATA_LOGGING_INT, sizeof(int), true);
5 }
```

#### Logovanie dát

Po vytvorení spojenia začína zber údajov. Volaním funkcie `data_logging_log()` sa pridáva nový záznam do logu označeného poskytnutou premennou `DataLoggingSessionRef`. Úspešnosť každého logovania sa dá skontrolovať pomocou premennej `DataLoggingResult`:

```
1 const int value = 16;
2 const uint32_t num_values = 1;
3
4 // Zaloguje sa 1 hodnota
5 DataLoggingResult result = data_logging_log(s_session_ref,
6  &value, num_values);
```

```
7
8  // Pri neúspešnom zalogovaní, vypíše chybovú správu
9  if(result != DATA_LOGGING_SUCCESS) {
10   APP_LOG(APP_LOG_LEVEL_ERROR, Error logging data:
11   %d, (int)result);;
12 }
```

Výpis 2.4: Logovanie dát.

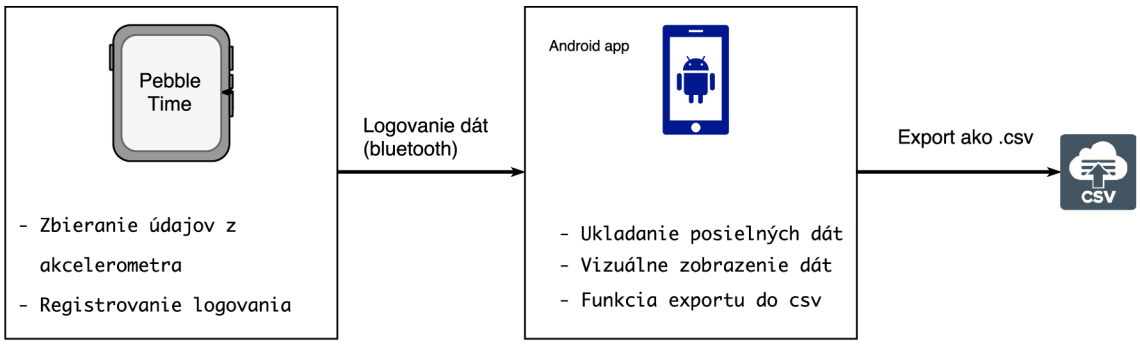
### Ukončenie spojenia

Po zalogovaní všetkých dát alebo pri zatváraní aplikácie je potrebné spojenie ukončiť aby sa jasne naznačilo že buď dáta sú prenesené alebo sú uložené pre neskorší prenos.

```
// Ukončenie spojenia a synchronyzovanie dát
data_logging_finish(s_session_ref);
```

# 3 NÁVRH SYSTÉMU SLEDOVANIA POHYBU A SPÁNKU

Pri navrhovaní sa museli brať do úvahy 3 faktory a to získanie, posielanie a spracovanie dát z akcelerometra. Nakoľko pri meraní pohybu ruky je potrebné získavať dáta v reálnom čase v určitom intervale, ale druhej strane pri sledovaní spánku stačí logovať dáta do telefónu ako celkovú dávku za celú noc preto sme sa rozhodli vytvoriť 2 návrhy zvlášť pre pohyb a spánok.

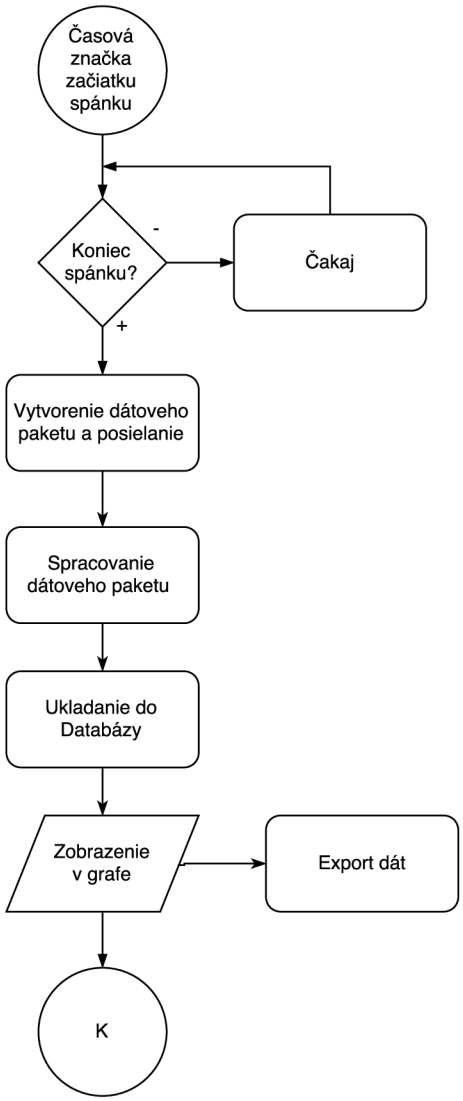



Obr. 3.1: Grafické zobrazenie návrhu systému.

## 3.1 Monitorovanie spánku

Pri návrhu systému ako celku sa vychádzalo z predpokladu, že sa jedná o rozsiahlejšiu aplikáciu, ktorú bude možné v budúcnosti rozširovať čo umožňuje jednoduché úpravy a dopĺňanie funkcionality aj po jej dokončení. Zmyslom návrhu je aby aplikácia na mobile tak aj na hodinkách bola ľahko použiteľná, prehľadná a vizuálne príťažlivá.

Úlohou hodinkovej aplikácie je zberanie dát z akcelerometra. Táto nameraná hodnota sa posiela do aplikácie na smartfóne, je vykreslená do grafu a uložená v databáze. Spárovanie zariadení pomocou bluetooth nie je potrebné po celej dobu spánku, iba pri posielaní dát z hodiniek do smartfónu. Pomocou funkcie exportu do CSV je možné tieto dáta neskôr analyzovať. Následujúci diagram zobrazuje celkový návrh aplikácie monitorovania spánku:

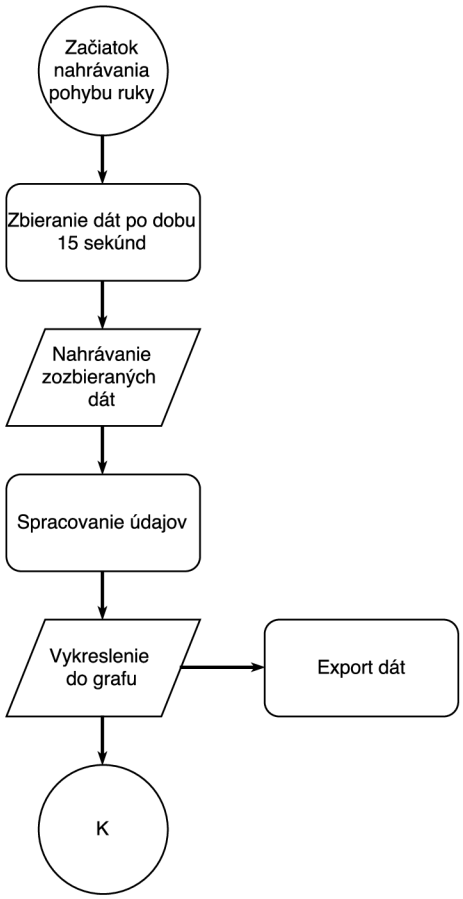



Obr. 3.2: Vývojový diagram monitorovania spánku.

- **Vytvorenie časovej značky** - Tento počiatočný proces slúži na to aby sa užívateľovi pred spaním vytvorila časová značka začiatku spánku cez vstavané tlačidlo. Následne aplikácia čaká dokiaľ nie je táto časová značka zakončená.
- **Dátový paket** - Po určení celej časovej značky sa na hodinkách spustí do popredia príslušná watchapp s textom pre potvrdenie zaslania dát. Hodinky dáta zabalia a zašlú pre spracovanie.
- **Spracovanie dát** - V telefóne sa tieto dáta rozbalia, zoradia podľa času a následne sa orežú podľa vytvorenej časovej značky užívateľom.
- **Uloženie dát** - Už spracované dáta sa uložia do databázy ako jeden objekt. Tomuto objektu sa priradí jeho unikátne UUID.
- **Zobrazenie v grafe** - Následne sa celý databázový objekt zobrazí v GUI
- **Export** - Vytvorený objekt dát je možne pomocou funkcie exportovať do CSV.

## 3.2 Meranie pohybu ruky

Tento návrh je podobný prvému s rozdielom v získavaní záznamov a jeho posielaní do telefónu. Tento systém závisí výhradne na komunikáciu v reálnom čase medzi Pebble hodinkami a Android smartfónom. V tomto prípade sú získavané surové dáta z akcelerometra na hodinkách a posielané simultánne do aplikácie na smartfóne. Po ukončení merania sú dáta vykreslené do grafu po osi X, Y a Z. Taktiež sú tieto dáta exportované ako CSV súbor na následnú analýzu.

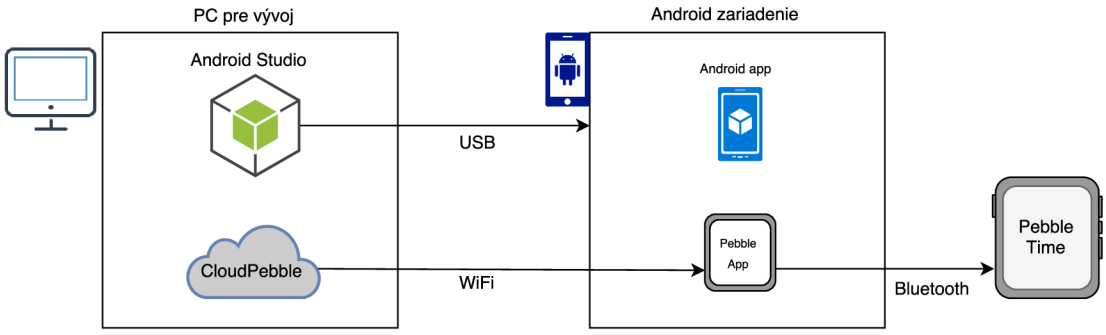



Obr. 3.3: Vývojový diagram zaznámenávania pohybu ruky.

- **Zahájenie merania pohybu** - V počiatočnom procese sa začnú merať surové dáta z akcelerometra hodiniek po osiach X, Y a Z po dobu 15 sekúnd
- **Nahranie dát** - Merané dáta sa nahrávajú a odosielajú po častiach a to 5 paketov po 5 záznamov z každej osi za sekundu.
- **Spracovanie dát** - Po uplynutí intervalu merania sa odoslané časti dát zoradia do jedného objektu.
- **Zobrazenie v grafe** - Následne sa všetky získané dáta zobrazia na grafe.
- **Export** - Po vyobrazení dát, rovnako ako pri meraní spánku je možnosť tieto dáta exportovať do CSV a hlbšie analyzovať.

# 4 IMPLEMENTÁCIA SYSTÉMU

Ako už bolo spomenuté v práci, Pebble využíva pre vytváranie aplikácií vlastné SDK. Na základe toho využívame niektoré vstavané API pri programovaní aplikácie bežiacich priamo na hodinkách. Keďže Pebble nie je navrhnuté aby dokázalo ukladať veľké množstvo dát a zároveň ich spracovať tak tieto watchapp sú výhradne určené len ako zdroj a vysielač dát, následne spracovanie má na starosti už príslušná mobilná aplikácia. Komunikácia medzi mobilným zariadením a hodinkami je výhradne riešená cez bluetooth. Pre mobilnú aplikáciu sme použili `PebbleKit Android API` pre možnosť komunikácie a `DataLogging API` zber a logovanie dát zhodiniek. Pebble SDK momentálne nie je k dispozícii na priamu inštaláciu pre OS Windows, tak pre vytváranie aplikácií na inteligentných hodinkách je vybrané prostredie CloudPebble IDE. Dôvodom je jednoduché použitie, kompilácia kódu a jeho následného ladenia. Samotné android aplikácie sú programované cez vývojové prostredie Android Studio. Obe android aplikácie sú vytvorené pre kompatibilitu s Android OS verziou 5.0 a vyššie.



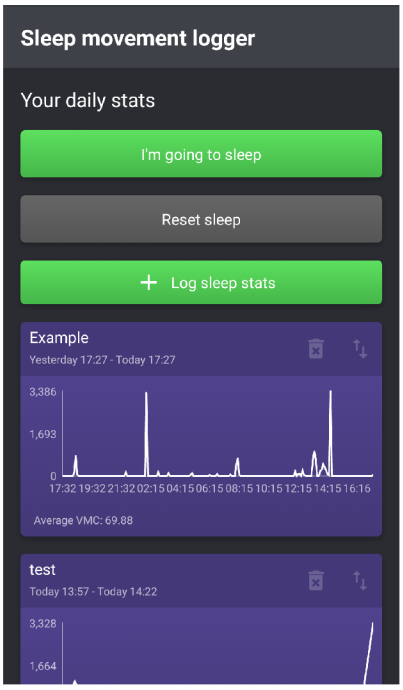

Obr. 4.1: Grafické zobrazenie vývoju aplikácií.

## 4.1 Sleep movement logger

Aplikácia Sleep movement logger slúži pre monitorovanie pohybu počas spánku užívateľa. Pre meranie pohybu používa namerané dáta z akcelerometra ktoré sú následne cez `HealthService API` spriemerované do hodnoty VMC. VMC udáva rozsah pohybu ruky v číselnej podobe za jednu minútu. Tieto hodnoty sa po ukončení spánku na základe časového intervalu určeného pred spánkom a po zobudení odošlú z hodiniek do telefónu, následne uložia do pamäti telefónu do databázy a vykreslia do grafu.

### Hlavná aktivita

Hlavná štruktúra aplikácie **Sleep movement logger** je zapisaná v Java triede `MainActivity.java`. Táto trieda slúži na zobrazenie UI pre užívateľa po spustení aplikácie. V triede sú taktiež definované metódy, funkcie, datatypy a premenné.




Obr. 4.2: Hlavná aktivita aplikácie.

Celkový vizuálny vzhľad a rozloženie objektov aplikácie je uložený v XML súboroch v zložkách **drawable** a **layout**.

### Nastavenie PebbleKit Android

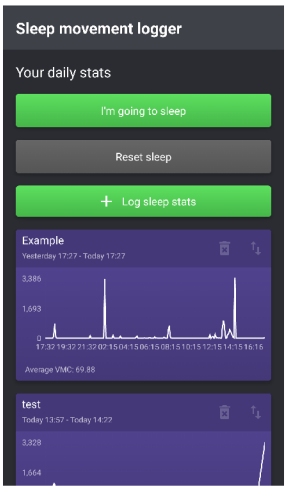
Predtým než dokáže android aplikácia komunikovať a vymieňať si dáta s hodinkou Pebble je potrebné zadefinovať nastavenie PebbleKit Android do projektu aplikácie v súbore `build.gradle`:

```
1 dependencies {
2   compile 'com.getpebble:pebblekit:4.0.1'
3 }
```

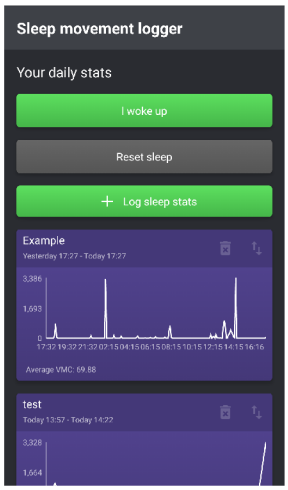
Výpis 4.1: Nastavenie PebbleKit.

### Monitorovanie spánku

Pre spustenie monitorovania spánku je potrebné si najprv vytvoriť časovú značku cez tlačidlo **I'm going to sleep**. Hneď po stlačení sa zmení text tlačítka na **I woke**




Obr. 4.3: Stlačenie tlačídla.




Obr. 4.4: Zmena textu na tlačidle.

**up**. Časový údaj sa uloží v aplikácií a bude čakať pre uzavretie intervalu spánku ďalším stlačením tlačidla. Tlačítko **Reset** slúži na odstránenie celej časovej značky.

Za použitia knižnice `Hawk`[1] sa údaj o časovej značke uloží pre ďalšie použitie ako kľúč `SLEEP_START_TIME_KEY`.

```
1  public void onSleepButtonClick() {
2      if (Hawk.contains(SLEEP_START_TIME_KEY)) {
3          sleepButtonText.setText(Resources.getString
4          (R.string."I'm_going_to_sleep"));
5       }
6       else {
7          Hawk.put(SLEEP_START_TIME_KEY, DateTime.now().
8          getMillis());
9          sleepButtonText.setText(Resources.getString
10         (R.string."I_woke_up"));
11     }
12 }
```

Výpis 4.2: Kód zobrazujúci metódu stlačenia tlačídla.

[1]Hawk - slúži ako zabezpečené úložisko pre ukladanie kľúčov.

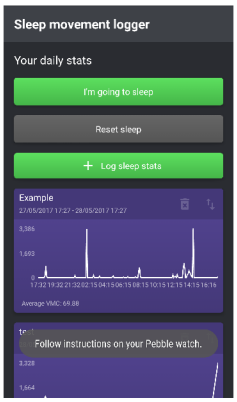
### Zbieranie a logovanie VMC dát

Ako už bolo spomenuté skôr aplikácia na inteligentných hodinkách používa pre získanie dát z akcelerometra HealthService API. Za využitia objektu `HealthMinuteData`, ktorý obsahuje viacero typov údajov súvisiacich s činnosťou a ktoré sa zaznamenávajú na minútovej bázi, je vybratá metrika VMC. Následná ukážka kódu zobrazuje použitie tejto metriky.

```
1  loggingSession = data_logging_create(VMC_LOG,
2  DATA_LOGGING_BYTE_ARRAY, sizeof(LogData), true);
3    for (int hour = LOG_HOURS; hour > 0; hour--) {
4      minuteData = (HealthMinuteData*)
5      malloc(MAX_RECORDS * sizeof(HealthMinuteData));
6
7      time_t start = time(NULL) - (hour * SECONDS_PER_HOUR);
8      time_t end = time(NULL) - ((hour - 1) * SECONDS_PER_HOUR);
9
10     uint32_t numRecords = health_service_get_minute_history
11     (minuteData, MAX_RECORDS, &start, &end);
12
13     result = DATA_LOGGING_NOT_FOUND;
14     for (uint16_t minute = 0; minute < numRecords; minute++) {
15       data->timestamp = start + (SECONDS_PER_MINUTE * minute)
16       * (MAX_RECORDS / numRecords);
17       data->is_data_valid = !minuteData[minute].is_invalid;
18       data->vmc = minuteData[minute].vmc;
19       result = data_logging_log(loggingSession, data, 1);
20       psleep(10);
21     }
22
23     free(minuteData);
24   }
```

Výpis 4.3: Získanie VMC dát použitím HealthService API.

Požiadavka zalogovania dát do telefónu sa inicializuje stlačením tlačítka **Log sleep stats** čo vyvolá okno s inštrukciami a zároveň sa do popredia na hodinkách zobrazí hlavný program, ktorá požiada užívateľa o stlačenie vstavaného tlačidla **SELECT**.




Obr. 4.5: Inštruktážne okno.

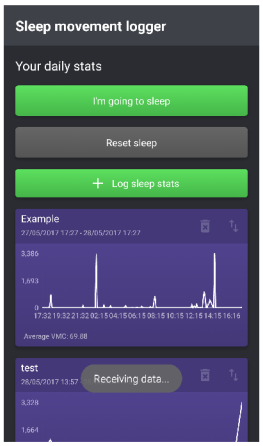

Obr. 4.6: Hlavné okno hodinkovej aplikácie.

Metódou `initPebbleReceiver` sa posielané dáta za posledných 24 hodín ukladajú do bufferu vo forme dátového poľa. Tieto dáta obsahujú samotný údaj o čase a dátume posledného merania.

```
1  private void initPebbleReceiver() {
2  logDataBuffer = new ArrayList<>();
3  // Príchod dát z hodiniek
4  dataLogReceiver = new PebbleKit.PebbleDataLogReceiver
5  (WATCHAPP_UUID) {
6
7    public void receiveData(Context context, UUID logUuid,
8    Long timestamp, Long tag, byte[] data) {
9      if (logDataBuffer.size() == 0) {
10       Toast.makeText(getApplicationContext(),
11       "Receiving_data...",
12       Toast.LENGTH_SHORT).show();
13     }
14   LogDataItem logDataItem = LogDataItem.fromByteArray(data);
15   logDataBuffer.add(logDataItem);
16   Timber.d("Data_received:" + logDataItem.toString());
17   }
18
19
20   public void onFinishSession(Context context, UUID logUuid,
21   Long timestamp, Long tag) {
22   Timber.d("Datalogging_session_ended");
23   showAddDialog();
24       }}  };
```

Výpis 4.4: Logovanie dát do aplikácie na mobile.

Užívateľovi sa proces posielania dát zobrazí na informačnom okne s textom **Receiving data**, zároveň sa po úspešnom poslaní dát zobrazí na hodinkách **Log successful**. Po spracovaní sú dáta pripravené pre ukladanie do databázy.

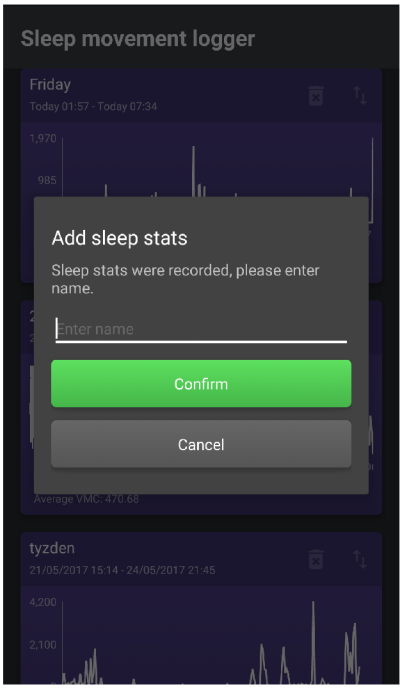



Obr. 4.7: Informačné okno posielania.


Obr. 4.8: Úspešné poslanie dát.

### Ukladanie do databázy a vykresľovanie do grafu

Dokončením logovania dát sa do popredia aktivuje dialógové okno pre zadanie názvu aktuálnej dávke nameraných hodnôt počas spánku.




Obr. 4.9: Dialógové okno uloženia dát.

```
1  private void showAddDialog() {
2      AddSleepStatsDialog.show(MainActivity.this,
3      new AddSleepStatsDialog.DialogEventListener() {
4
5        public void onDialogConfirmed(String name) {
6          saveData(name);
7        }
8
9        public void onDialogCanceled() {
10       logDataBuffer.clear();
11       }
12     });
13   }
```

Výpis 4.5: Zobrazenie dialogového okna.

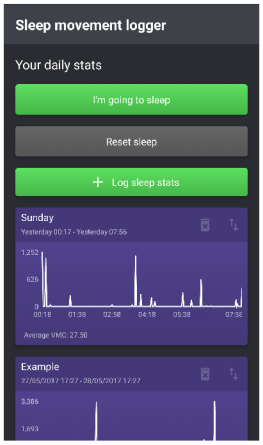
Uloženie dát do databázy je možné zrušiť tlačidlom **Cancel**. Potvrdzovacím tlačidlom **Confirm** sa vytvorí nový objekt v databáze so zadaným názvom a unikátnym náhodne vygenerovaným UUID, kde sa celkový blok dát vytriedi postupne od počiatočného času až po koncový. Zároveň sa po uložení vyprázdni aj buffer, kde dané dáta boli dočasné uložené. Databáza je reprezentovaná použitou knižnicou `Realm DB`.

```
1  private void saveData(final String name) {
2  realm.executeTransaction(new Realm.Transaction() {
3
4  public void execute(Realm realm) {
5  // Vytvaranie objektu v DB s unikatnym ID
6  SleepStats sleepStats = realm.createObject
7  (SleepStats.class, UUID.randomUUID().toString());
8
9  // Zoradenie prijatých dát podla datumu
10 Collections.sort(logDataBuffer, new Comparator<LogDataItem>()
11 {
12 public int compare(LogDataItem o1, LogDataItem o2)
13 {
14 if (o1.timestamp.getMillis() < o2.timestamp.getMillis()) {
15     return -1;
16   } else if (o1.timestamp.getMillis()
17   > o2.timestamp.getMillis()) {
18   return 1;
19   } else {
20     return 0;
```

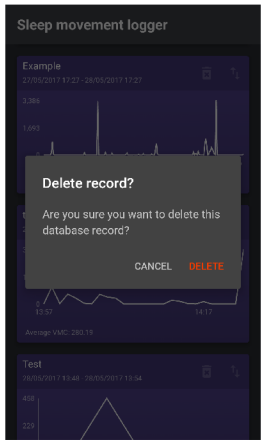
```
21    }
22 }});
23      if (Hawk.contains(SLEEP_START_TIME_KEY)) {
24      TimeRange range = new TimeRange();
25
26      range.setStart(new DateTime((long)
27      Hawk.get(SLEEP_START_TIME_KEY)));
28      range.setEnd(new DateTime());
29
30      List<LogDataItem> rangedDataBuffer = new ArrayList<>();
31      for (LogDataItem item : logDataBuffer) {
32      if (item.timestamp.isAfter(range.start) &&
33      item.timestamp.isBefore(range.end)) {
34      rangedDataBuffer.add(item);
35    }
36 }
```

Výpis 4.6: Metóda uloženia dát do databázy.

Vykreslenie dát do grafu prebieha v metóde `setupRecycler()` pomocou objektu **Adapter**, ktorý sa napojí na **RecyclerView** a vytvorí **Viewholder** z vybraných dát, ktoré zobrazí v podobe grafu. Pre vykreslenie grafu je použitá knižnica `williamchart:2.4.0`. Daný objekt je možné aj odstrániť. Graf taktiež zobrazuje priemerný pohyb počas noci v podobe **Average VMC**.




Obr. 4.10: Vytvorenie objektu dát.




Obr. 4.11: Odstránenie objektu.

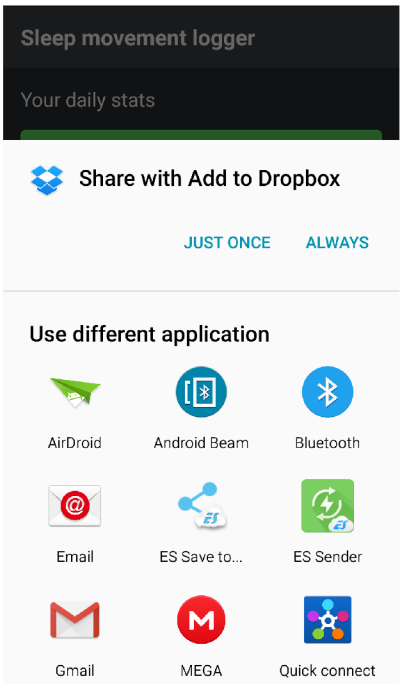
```
1  private void setupRecycler() {
2  dailyRecyclerLayoutManager = new LinearLayoutManager(this,
3  LinearLayoutManager.VERTICAL, true);
4  dailyRecyclerLayoutManager.setStackFromEnd(true);
5  dailyStatsAdapter = new
6  DailyStatsAdapter(dailyStatsEventListener);
7  dailyStatsRecycler.setLayoutManager
8      (dailyRecyclerLayoutManager);
9  dailyStatsRecycler.setAdapter(dailyStatsAdapter);
10 dailyStatsRecycler.setNestedScrollingEnabled(false);
11
12     // Vytiahne data z databazy a naplni sa adapter
13     RealmResults<SleepStats> dbResults = realm.where
14     (SleepStats.class).findAll();
15     for (SleepStats sleepStats : dbResults) {
16       dailyStatsAdapter.addItemWithoutNotify(sleepStats);
17       dailyStatsAdapter.notifyDataSetChanged();
18     }
19   }
```

Výpis 4.7: Metóda vykreslenia databázového objektu na grafe.

### Export do CSV

Export dát prebieha v metóde `onExportClick`, kde pomocou android komponentu FileProvider sa vrácia URI[2] CSV súboru. Pri získaní URI cesty sa vytvorí objekt Intent[3] so záznamom cesty ku CSV súboru, ktorému sa nastaví akcia zdieľania. Zavolaním novej aktivity `startActivity(Intent)` sa už sám Android postará o zdieľanie, zobrazí sa okno zdieľania pre vybratie typu exportu.

[2]URI - Kompaktný reťazec znakov používaný na identifikáciu alebo pomenovanie zdroja dát.
[3]Intent - Android objekt pomocou ktorého prebieha komunikácia medzi viacerými aplikáciami.




Obr. 4.12: Možnosť exportu dát.

```
1
2  public void onExportClick(int adapterPosition) {
3  if (ContextCompat.checkSelfPermission(MainActivity.this,
4    Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
5    PackageManager.PERMISSION_GRANTED) {
6      Toast.makeText(MainActivity.this, "No_permission",
7      Toast.LENGTH_SHORT).show();
8      ActivityCompat.requestPermissions(MainActivity.this,
9      new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}
10     , 0);
11     return;
12       }
13 Uri uri = FileProvider.getUriForFile(MainActivity.this,
14 BuildConfig.APPLICATION_ID + ".provider", tempFile);
15   Intent sendIntent = new Intent();
16   sendIntent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
17   sendIntent.setAction(Intent.ACTION_SEND);
18   sendIntent.putExtra(Intent.EXTRA_STREAM, uri);
19   sendIntent.setType("text/csv");
20   startActivity(sendIntent);
```

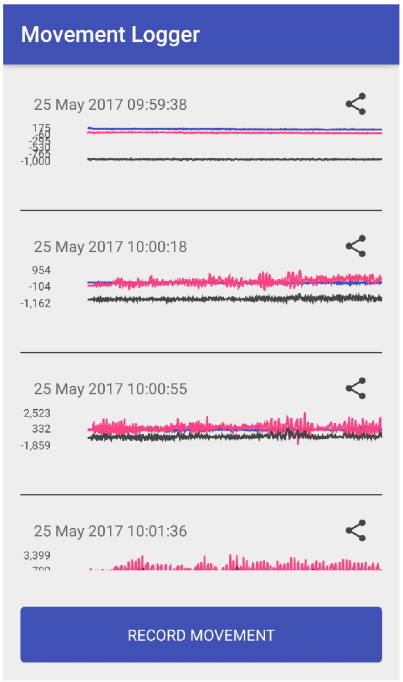
Výpis 4.8: Vytvorenie aktivity exportu dát.

## 4.2 Movement logger

Aplikácia Movement logger slúži pre zhromažďovanie a zaznamenávanie vzoriek údajov o zrýchlení pri tremore ruky. V tomto prípade sa za použitia AccelerometerService API zbierajú surové dáta maximálneho zrýchlenia v jednotkách `milli-G`. Tieto záznamy môžu nadobúdať rozsah od -4000 až po +4000 po každej z osi X, Y a Z. Záznam dát prebieha v reálnom čase po stanovenú dobu 15 sekúnd so vzorkovacou frekvenciou 25 Hz čo znamená že výsledkom merania je 375 vzoriek po každej z osí. Dáta sa bez prerušenia ukladajú do buffera na mobile a po ukončení merania sa daný blok hodnôt vykreslí do grafu. Dáta je možné taktiež exportovať ako súbor .csv prostredníctvom viacero metód (email, uloženie na cloud ...).

### Hlavná aktivita

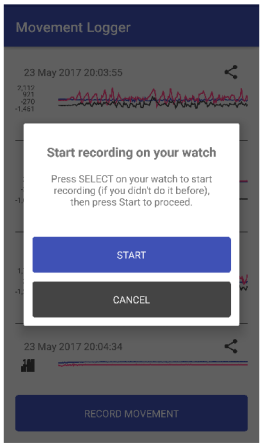
Obdobne ako v predošlej aplikácií hlavná štruktúra chodu aplikácie **Movement logger** je definovaná v Java triede `MainActivity.java`. Vizuálny vzhľad a rozloženie objektov aplikácie je definovaný v zdrojových zložkách **drawable** a **layout**.




Obr. 4.13: Hlavné okno aplikácie movement logger.

### Zber dát

Stlačením **Record movement** sa do popredia aplikácie zobrazí dialógove okno s inštruktážnym textom pre užívateľa. Zároveň sa spustí do popredia hlavné okno hodinkovej aplikácie.




Obr. 4.14: Dialógove okno záznamu.


Obr. 4.15: Aplikácia na Pebble.

```
1  public void showRecordDialog() {
2    final MaterialDialog dialog =
3    new MaterialDialog.Builder(this)
4
5      .customView(R.layout.dialog_record_info, false)
6      .cancelable(true)
7      .build();
8    TextView okButton = ButterKnife.findById
9      (dialog, R.id.button_ok);
10   TextView cancelButton = ButterKnife.findById
11     (dialog, R.id.button_cancel);
12   okButton.setOnClickListener(new View.OnClickListener() {
13   public void onClick(View view) {
14     dialog.dismiss();
15     presenter.startRecording();
16       }
17     });
18 cancelButton.setOnClickListener(new
19 View.OnClickListener()
20     {
21     public void onClick(View view) {
22         dialog.dismiss();
23       }
```

```
24      });
25      dialog.show(); }
```

Výpis 4.9: Vytvorenie dialógového okna.

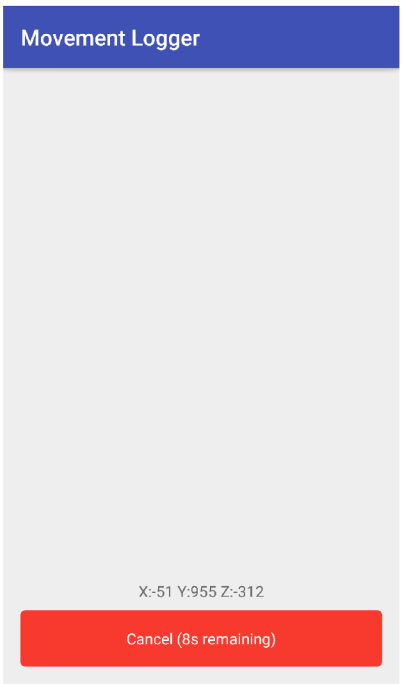
Stlačenie vstavaného tlačítka **SELECT** spustí posielanie údajov pohybu z akcelerometera, ktorý začne vypisovať počet hodnôt v tvare packets/s. Jeden paket tvorí 5 meraní po každej z 3 osí.


Obr. 4.16: Zber dát z akcelerometra.

```
1  static void select_click_handler(ClickRecognizerRef
2  recognizer, void *context) {
3    if(!s_sending) {
4      // Začiatok posielania dát
5      s_sending = true;
6      accel_service_set_sampling_rate(ACCEL_SAMPLING_25HZ);
7      accel_data_service_subscribe(SAMPLES_PER_UPDATE,
8      accel_data_handler);
9      text_layer_set_text(s_text_layer, "Started");
10     if(s_packets_per_second_timer) {
11       app_timer_cancel(s_packets_per_second_timer);
12     }
13     s_packets_per_second_timer = app_timer_register(1000,
14     packets_per_second_handler, NULL);
15     window_set_background_color(s_window, GColorOrange);
16   }
```

Výpis 4.10: Posielanie údajov o pohybe ruky.

Zahájenie záznamu príjmaných dát sa inicializuje tlačidlom **Start** v aplikácií na smartfóne po ktorom sa začne odpočet 15 sekúnd nahrávania. Posielanie a prijímanie dát prebieha v reálnom čase.



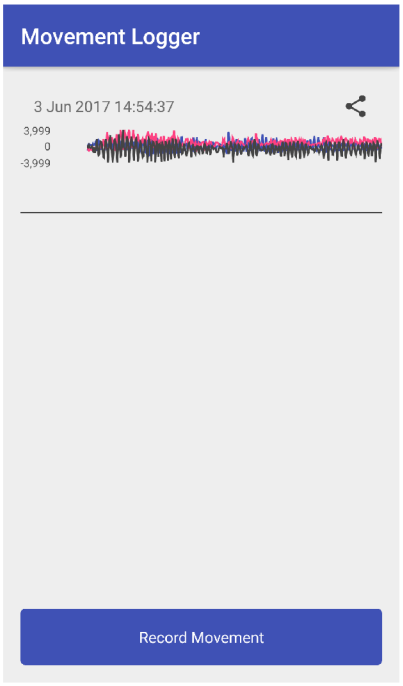

Obr. 4.17: Záznamenávanie pohybu ruky.

```
1  public void receiveData(Context context, int transactionId,
2  PebbleDictionary data) {
3  PebbleKit.sendAckToPebble(context, transactionId);
4    for (int i = 0; i < SAMPLES_PER_UPDATE; i++) {
5      AccelSample sample = new AccelSample();
6      sample.setX(data.getInteger((i * ELEMENTS_PER_PACKAGE))
7      .intValue());
8      sample.setY(data.getInteger((i * ELEMENTS_PER_PACKAGE)
9      + 1).intValue());
10     sample.setZ(data.getInteger((i * ELEMENTS_PER_PACKAGE)
11     + 2).intValue());
12     presenter.onSampleReceived(sample);
13       }
14     }
```

Výpis 4.11: Záznam dát po osi X, Y a Z.

### Vykreslenie do grafu

Prijaté záznamy sa priamo ukladajú do objektu **Adapter** ktorý po skončení časového intervalu vytvorí **Viewholder** nahraných dát a vykresli ich do grafu. Táto aplikácia na smarfóne taktiež používa knižnicu `williamchart: 2.4.0.` pre vykresľovanie grafu.




Obr. 4.18: Vykreslenie grafu nového záznamu.

Aplikácia dané záznamy neukladá do žiadnej databázy iba ich vykreslí po ukončení merania.

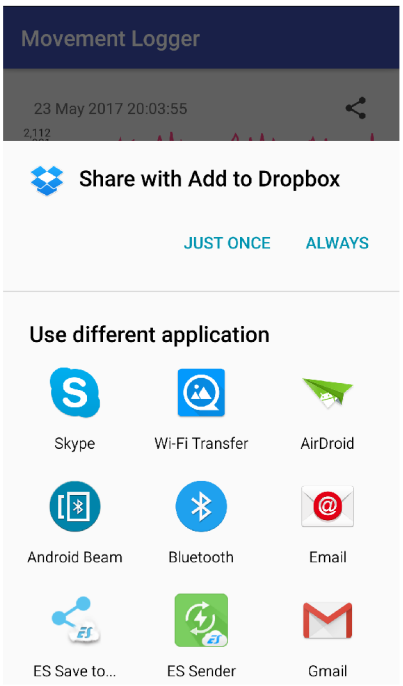
```
1 public void initRecycler() {
2   LinearLayoutManager layoutManager = new LinearLayoutManager
3     (this, LinearLayoutManager.VERTICAL, false);
4   adapter = new LogSessionAdapter(viewHolderListener);
5   recycler.setLayoutManager(layoutManager);
6   recycler.setAdapter(adapter);
```

Výpis 4.12: Metóda zobrazenia nového záznamu.

### Export dát

Obdobne ako pri aplikácií **Sleep movement logger**, k exportu celého záznamu dát je využitý komponent `FileProvider`, ktorý vracia URI cestu daného CSV súboru

pre vytvorenie novej Intent aktivity pre následné zdieľanie. Zobrazí sa zdieľacie okno a zvyšok už zariadi sám Android.




Obr. 4.19: Funkcia exportu do CSV.

```
1  public void sendFileIntent(File file) {
2    Uri uri = FileProvider.getUriForFile(MainActivity.this,
3    BuildConfig.APPLICATION_ID + ".provider", file);
4    Intent sendIntent = new Intent();
5    sendIntent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
6    sendIntent.setAction(Intent.ACTION_SEND);
7    sendIntent.putExtra(Intent.EXTRA_STREAM, uri);
8    sendIntent.setType("text/csv");
9    startActivity(sendIntent);
10   }
```

Výpis 4.13: Metóda exportu do CSV.

# 5 TESTOVANIE SYSTÉMU

Testovanie bolo rozdelené na 2 časti a to testovanie pri spánku a pri pohybe ruky. Monitorovanie spánku prebiehalo po dobu 2 týždňov počas ktorých bola inteligentná hodinka Pebble nosená počas noci a pomocou aplikácie na smartfóne sa dané hodnoty pohybu zaznamenávali pre ďalšiu analýzu. Následne sme tieto záznamy spánku exportovali ako súbor CSV a zisťovali sme kvalitu spánku.

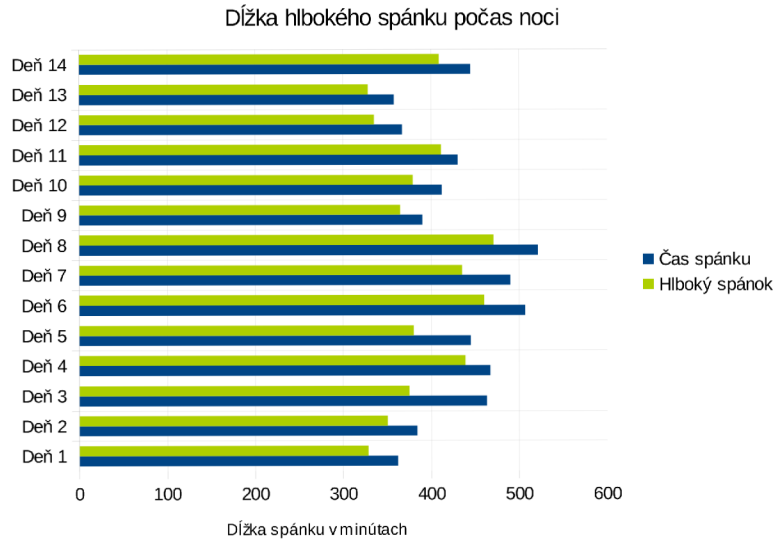
Meranie pohybu ruky bolo zaznamenávané po dobe 15 sekúnd v dvoch štádiách a to pri kľudovom stave ruky a pri simulácií tremoru.

# 6 VÝSLEDKY ŠTUDENTSKEJ PRÁCE

Všetky namerané dáta spánku sme spracovali a vyhodnotili do 3 metrík a to:

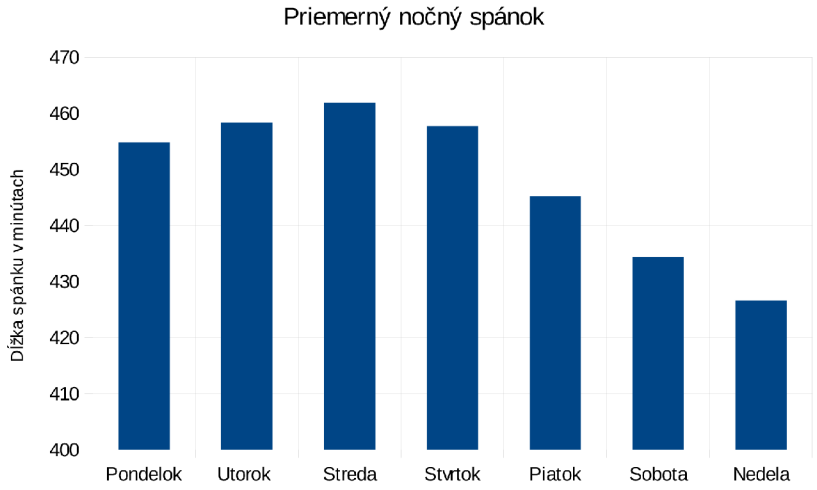
- Dĺžka hlbokého spánku počas noci
- Priemerný spánok počas 2 týždňov
- Priemerný pohyb počas noci

Nasledujúci graf zobrazuje kvalitu spánku každého dňa ako dĺžku hlbokého spánku oproti celkovému času v minútach.



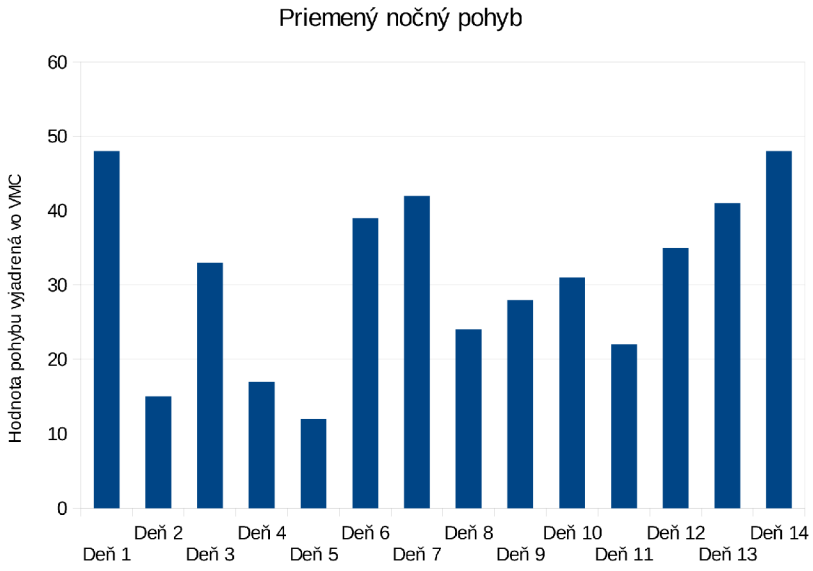

Obr. 6.1: Odmeraný hlboký spánok.

Hodnota priemerneho času spánku určuje v ktorý deň v týždni som mal najdlhší spánok na základe priemeru dvoch rovnakých dni dohromady.



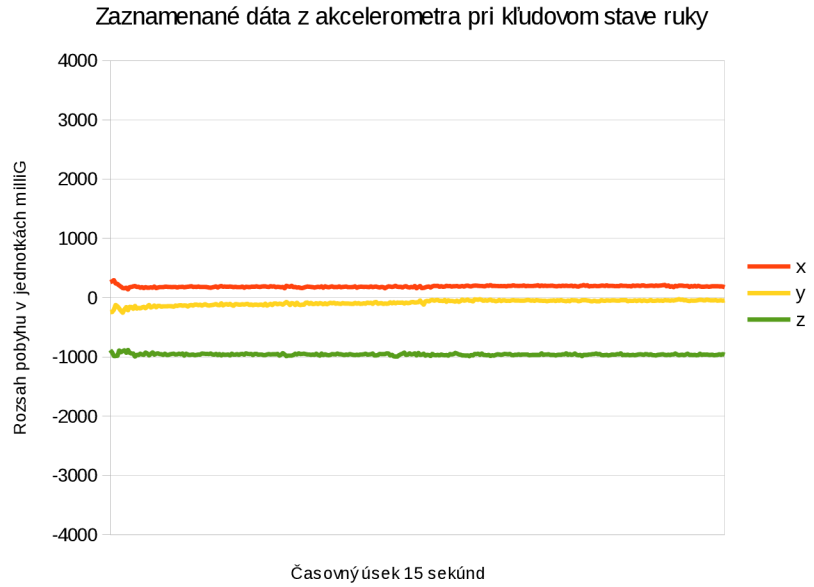

Obr. 6.2: Priemerný nočný spánok v daný deň za posledné 2 týždne.

Posledný graf určuje úroveň aktivity počas noci ku danému dňu merania. Všetky namerané hodnoty každej minúty boli zjednotené do výsledného pohybu počas spánku.

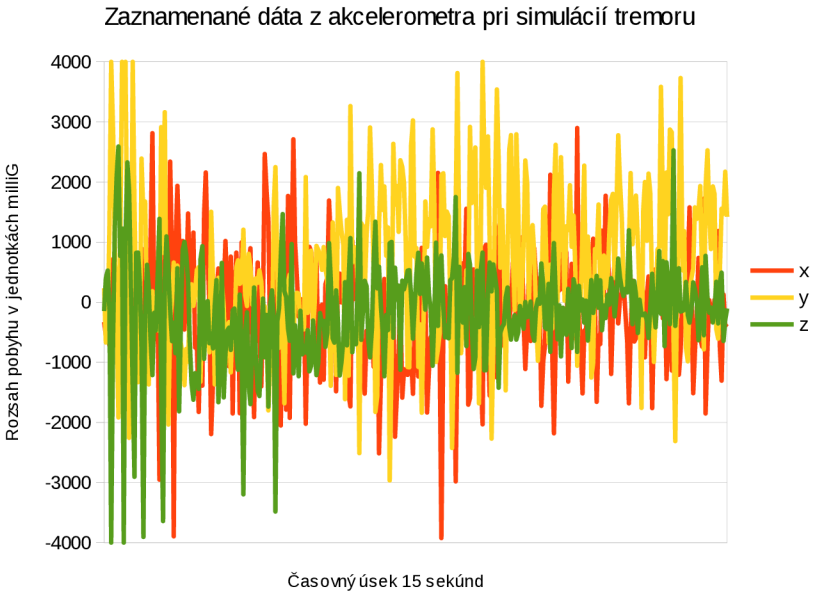



Obr. 6.3: Priemerný nočný pohyb počas 2 týždnov.

Pri meraní pohybu ruky sme použili exportované hodnoty záznamu a následne sme ich vyniesli do dvoch grafov a to pri meraní kľudového stavu a pri simulácií tremoru.




Obr. 6.4: Meranie pri kľudovom stave ruky.




Obr. 6.5: Simulácia tremoru.

# 7 ZÁVER

V prvej časti bakalárskej práce sme sa zamerali na krátky popis nositeľných zariadení vo využití v medicíne, taktiež sme popísali rôzne použitia pre sledovanie životných funkcií, a vysvetlenie základnej terminológie. Ďalej je v práci popis samotných hodiniek Pebble Smartwatch, Pebble SDK, použitie akcelerometra a možnosti spracovania a logovania dát. Cieľom práce bolo vytvoriť aplikácie pre sledovanie spánku a zaznamenávanie pohybu ruky za využitia práve akcelerometra hodiniek a následný export do CSV.

Pre vývoj aplikácií na hodinkách sme použili online vývojové prostredie CloudPebble, ktoré bolo užívateľsky priateľské na použitie a kompiláciu kódu. Veľkou výhodou prostredia je možnosť emulácie vytvoreného kódu pre rýchle otestovanie funkcionality. Pebble SDK má priamo na stránkach spoločnosti Pebble vlastnú sekciu s dôkladne pripravenou dokumentáciou, ktorá je veľmi užitočná pre každého vývojára. Samotné android aplikácie boli vyvinuté v prostredí Android Studio využívajúci nový kompilovací nástroj Gradle.

Aplikácia monitorovania spánku používa ako zdroj dát HealthService API metriku VMC, ktorá je priemerná minútová hodnota celkového pohybu ruky. Pre záznam nie je nutné mať spárované obe zariadenia cez Bluetooth počas celej doby zbierania dát hodinkami, čo šetrí baterku na hodinkách tak aj na smartfóne. Doba spánku je určená užívateľom kde cez interaktívne tlačítká v aplikácií na smartfóne vytvára časovú značku začiatku a konca spánku. Tieto záznamy sa ukladajú do databázy a vykresľujú do grafu s údajom o pohybe a čase. Všetky uložené záznamy majú možnosť exportu ako CSV súbor. Aplikácia sledovania pohybu ruky využíva priamo surové hodnoty z akcelerometra ako zdroj dát pre záznam. Oproti monitorovaniu spánku zbieranie, posielanie a spracovanie prebieha v reálnom čase tým pádom je potrebné mať obe zariadenia pri zázname spárované. V aplikácií su vstavané informačne okná čo zľahčuje jej použiteľnosť pre každého užívateľa. Záznam trvá 15 sekúnd po ktorom sa všetky hodnoty vykreslia do grafu. Táto aplikácia taktiež umožňuje export dát. V poslednej časti boli získané záznamy spánku spracované do 3 grafov slúžiacich pre určenie kvality spánku. Pri meraní pohybu ruky sme dané dáta použili pre vynesenie 2 grafov a to pri kľudovom stave a pri simulácií tremoru.

Stanovený cieľ práce bol splnený v plnom rozsahu s možnosťou rozšírenia funkcionality u oboch aplikácií. V prípade pokračovania, by dané systémy mohli byť aplikované v praxi na niekoľkých pacientov buď s poruchami spánku alebo trpiacimi tremorom ruky.

# LITERATÚRA


[1] AJAMI, Sima a Fotooheh TEIMOURI. Features and application of wearable biosensors in medical care [online]. [cit. 2017-06-01]. DOI: 10.4103/1735-1995.172991. ISBN 10.4103/1735-1995.172991. Dostupné z URL: <http://www.jmsjournal.net/text.asp?2015/20/12/1208/172991>

[2] BIEBER, Gerald, Thomas KIRSTE a Bodo URBAN. Ambient interaction by smart watches. Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '12 [online]. New York, New York, USA: ACM Press, 2012, , 1- [cit. 2017-05-07]. DOI: 10.1145/2413097.2413147. ISBN 9781450313001. Dostupné z URL: <http://dl.acm.org/citation.cfm?doid=2413097.2413147>

[3] BIEBER, Gerald, Marian HAESCHER a Matthias VAHL. Sensor requirements for activity recognition on smart watches [online]. [cit. 2017-05-07]. DOI: 10.1145/2504335.2504407. ISBN 10.1145/2504335.2504407. Dostupné z URL: <http://dl.acm.org/citation.cfm?doid=2504335.2504407>

[4] CHYLA, Peter. Remote display for smartphone [online]. 2013 [cit. 2017-05-07]. Dostupné z URL: <http://hdl.handle.net/10211.2/3759>

[5] CloudPebble. Cloud pebble. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://cloudpebble.net/>

[6] Freertos. FreeRTOS [online]. [cit. 2017-05-07]. Dostupné z URL: <http://www.freertos.org/RTOS.html>

[7] KAEWKANNATE, Kanitthika a Soochan KIM. A comparison of wearable fitness devices [online]. [cit. 2017-05-01]. DOI: 10.1186/s12889-016-3059-0. Dostupné z URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4877805/>

[8] NARAYANASWAMI, C. a M.T. RAGHUNATH. Application design for a smart watch with a high resolution display [online]. [cit. 2017-05-08]. DOI: 10.1109/ISWC.2000.888452. ISBN 10.1109/ISWC.2000.888452. Dostupné z URL: <http://ieeexplore.ieee.org/document/888452/>

[9] Pebble. Pebble developer guide. [online]. [cit. 2017-05-03]. Dostupné z URL: <https://developer.getpebble.com/2/guides/>

[10] Pebble. Working with the PebbleKit JavaScript Framework. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://developer.getpebble.com/2/guides/javascript-guide.html>

[11] Pebble. (2014d). Develop for pebble. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://developer.getpebble.com/>

[12] Pebble. Accelerometer. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://developer.pebble.com/guides/events-and-services/accelerometer/>

[13] Pebble. DataLogging. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://developer.pebble.com/guides/communication/datalogging/>

[14] ROSENBERGER, MARY E., MATTHEW P. BUMAN, WILLIAM L. HASKELL, MICHAEL V. MCCONNELL, LAURA L. CARSTENSEN a Yolanda ALADRO. Twenty-four Hours of Sleep, Sedentary Behavior, and Physical Activity with Nine Wearable Devices [online]. [cit. 2017-06-03]. DOI: 10.1249/MSS.0000000000000778. ISBN 10.1249/MSS.0000000000000778. Dostupné z URL: <http://content.wkhealth.com/linkback/openurl?sid=WKPTLP:landingpage>

[15] RUTHERFORD, Jesse Jayne. Wearable Technology [online]. [cit. 2017-06-01]. DOI: 10.1109/MEMB.2010.936550. ISBN 10.1109/MEMB.2010.936550. Dostupné z URL: <http://ieeexplore.ieee.org/document/5463002/>

[16] ZHENG, Xiaochen, Alba VIEIRA CAMPOS, Joaquín ORDIERES-MERÉ, Jose BALSEIRO, Sergio LABRADOR MARCOS a Yolanda ALADRO. Continuous Monitoring of Essential Tremor Using a Portable System Based on Smartwatch [online]. [cit. 2017-06-03]. DOI: 10.3389/fneur.2017.00096. ISBN 10.3389/fneur.2017.00096. Dostupné z URL: <http://journal.frontiersin.org/article/10.3389/fneur.2017.00096/full>

[17] Xu S, Zhang Y, Jia L, Mathewson KE, Jang KI, Kim J, et al. Soft microfluidic assemblies of sensors, circuits, and radios for the skin. Science. [online]. [cit. 2017-05-01]. Dostupné z URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5334130/>

# ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

| | |
|---|---|
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| SDK | Software Development Kit |
| UI | User Interface |
| URI | Uniform Resource Identifier |
| UUID | Universally Unique Identifier |
| VMC | Vector Magnitude Count |

# ZOZNAM PRÍLOH

# A OBSAH PRILOŽENÉHO CD

- Elektronická verzia práce.
- Zdrojové kódy Pebble a Android aplikácií.
- Inšalačné súbory Android aplikácií.

# B ŠTRUKTÚRA PRILOŽENÉHO CD

- `/` ..... koreňový adresár priloženého CD
  - `xracek09.pdf`
  - `Sleep` ..... adresár aplikácií merania spánku
    - `SleepMovement.apk`
    - `Android`
      - `hodziny-receiver`
        - `.gradle`
        - `.idea`
        - `app` ..... adresár obsahujúci zdrojové kódy
        - `build` ..... vygenerované súbory po kompilácií
        - `gradle` ..... kompilačný nástroj
        - `build.gradle` ..... konfigurácia kompilátora
    - `Pebble`
      - `hodziny`
        - `build` ..... zkompilovaná Pebble aplikácia
        - `src` ..... zdrojové kódy Pebble aplikácie
  - `Movement` ..... adresár aplikácií merania pohybu ruky
    - `MovementLogger.apk`
    - `Android`
      - `hodziny-receiver`
        - `.gradle`
        - `.idea`
        - `app`
        - `build`
        - `gradle`
        - `build.gradle`
    - `Pebble`
      - `hodziny`
        - `build`
        - `src`