



Expertise
and insight
for the future

Johanna Löfblom

Prototyping with Movesense Platform – Breathing Application

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

04 June 2019

PREFACE

Writing this thesis was short but very intensive period just prior deadline, including many cups of coffee and several sleepless nights. However, gathering knowledge took almost two years, and included several Movesense meetups, two Junction hackathons and many hours of investigation and reflection. I can only hope that most of that information ended up in this thesis in a coherent way.

I have been inspired by positive computing and intend to follow best practices it provides. I will gladly continue to work with Movesense platform, since it offers great possibilities to build the final breathing application and potentially whole wellbeing toolbox.

I thank Movesense developers for their support and guidance. I am grateful of Junction experience and inspiring ideas my team members shared. I am humbled by flexibility that my supervisor showed towards me. And most of all I am lucky to have supportive family nudging me forward.

May this thesis spread positive vibes and flow your mind.

Vantaa, 04 June 2019
Johanna Löfblom

Author Title	Johanna Löfblom Prototyping with Movesense Platform – Breathing Application
Number of Pages Date	71 pages 04 June 2019
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Head of Degree Program
<p>This thesis explored new Movesense platform for development of wearable solutions. Long-term reusability of sensor and openness of platform lead to selection. Prototyping was guided by user experience design and positive computing practices. Benefits of diaphragmatic breathing made it optimal starting point for wellbeing toolbox of applications.</p> <p>This study was carried out by collecting information of different assets of the platform, and of specific application related knowledge, like measurement and use of breathing, and application of technology for wellbeing and human potential. Findings from pre-study were used as well, and study continued the design process as Develop or Ideation phase.</p> <p>Implementation experimented with APIs, sensors, modules, plugins, libraries, file formats, scripts, apps and different development tools in the Movesense platform or associated with the platform. And most of all with ideas and thoughts provided both by the literature and open community.</p> <p>The outcome was four prototypes implemented, (and six planned for) ranging from data handling, visualization, wireless communication, and fusing of data sources. Furthermore collected knowledge from both pre-study and this study will be used for building the complete solution in the next phase of design process.</p>	
Keywords	Prototyping, Movesense, Platform, Breathing Application, Positive Computing

Contents

Preface

Abstract

List of Abbreviations

1	Introduction	1
1.1	Background	1
1.2	Objective	2
1.3	Research Method	3
2	Movesense platform	4
2.1	Movesense Architecture	6
2.1.1	Movesense Device Overview	8
2.1.2	Movesense Accessories and Additional Tools	9
2.1.3	Movesense Libraries and Services overview	10
2.2	Movesense Device	10
2.2.1	Sensors	11
2.2.2	Device API	13
2.2.3	Device App and Modules	17
2.2.4	Device Communication	21
2.3	Movesense Mobile App	24
2.3.1	Mobile Library	24
2.3.2	Mobile API	24
2.3.3	Mobile App	25
2.4	Setup Movesense system	25
2.4.1	Upgrade Apps	26
2.4.2	Setup Device Build Environment	27
2.4.3	Setup Mobile App Build Environment	28
2.5	Build Movesense Device Apps	29
2.5.1	App Development Recommendations	29
2.5.2	Device Simulator on PC	32
2.6	Movesense Community Projects and Plugins	33
2.6.1	Holon.ist Open World Mobile Music Platform	34
2.6.2	FSenSync Synchronised Sensing Over Wi-Fi	35
2.6.3	KÄVELI Platform for Patient Monitoring and Diagnostics	35
2.6.4	Additional Third Party Tools and Methods	36
2.6.5	Third Party Movesense Plugins – Unity and Xamarin	37

2.6.6	Partner Services and Solutions	40
3	Breathing Application Specific Knowledge	44
3.1	User Experience Principles for Developing Health and Fitness Wearables	44
3.2	Positive Computing	45
3.3	Definition of Breathing Application	49
3.3.1	Respiratory System	49
3.3.2	Importance of Breathing Exercises	50
3.3.3	Measuring Breathing	53
4	Implementation of Breathing Application Prototype	58
4.1	Road Map for Breathing Application Prototypes	58
4.2	Setup for First Prototype – Handling Breathing Data	59
4.3	Setup for Second Prototype – Unity Plugin	61
4.4	Setup for Third Prototype – GATT Client with Gyroscope	63
4.5	Setup for Fourth Prototype – Mood and breathing	65
5	Results and Analysis	68
6	Conclusions	70

References

List of Abbreviations

AAR	Android Archive Software Library
AFE	Analog-Front-End
AHRS	Attitude and Heading Reference System
ANS	Autonomic Nervous System
API	Application Programmable Interface
APK	Android Package
ARM	Advanced RISC Machines
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
CNS	Central Nervous System
CO ₂	Carbon Dioxide
CR ddt	IEC prefix for Li-MNO ₂ coin cells, with dd diameter, tt thickness in mm.
CSV	Comma Separated Value
CV	Coefficient of Variation
DFU	Device Firmware Update
DIY	Do-It-for-Yourself
EE	Energy Expenditure
EEG	Electroencephalography
EEPROM	Electrically Erasable Programmable Read-Only Memory
ECG	Electrocardiography
EPOC	Post-Exercise Oxygen Consumption
FAQ	Frequently Asked Questions
FIR	Finite Impulse Response
GAP	General Access Profile
GATT	General Attribute Profile
GCC	GNU Compiler Collection
HR	Heart Rate
HREx	Exercise Heart Rate
HRR	Heart Rate Recovery
HRV	Heart Rate Variability
HTTP	Hypertext Transfer Protocol
IBI	Inter-Beat-Interval
IDE	Integrated Development Environment
IIR	Infinite Impulse Response
IMU	Inertial Measurement Unit

iOS	iPhone Operating System
KVM	Kernel-based Virtual Machine
LTO	Link Time Optimization
MCU	Microcontroller Unit
MDS	Movesense Device Service Library
MEMS	Micro-Electro-Mechanical Systems
MHR	Maximum Heart Rate
MIDI	Musical Instrument Digital Interface
MVVM	Model-View-Viewmodel
OSC	Open Sound Control Protocol
OTA	Over-The-Air
PCU	Processor Controller Unit
PDU	Protocol Data Unit
RAM	Random Access Memory
REST	Representational State Transfer
RIP	Respiratory Inductance Plethysmography
RF	Radio Frequency
rMSSD	Root Mean Square of the Successive Differences
RR	Respiratory Rate
RR-interval	Peak-to-Peak Interval between consecutive heart beats
RSA	Respiratory Sinus Arrhythmia
RSC	Running Speed and Cadence Profile
SDNN	Standard Deviation
SDNN Index	Mean of the Standard Deviations
SNR	Signal-to-Noise-Ratio
SoC	System on Chip
STL	Standard Template Library
VO ₂	Oxygen Consumption
YAML	YAML Ain't Markup Language

1 Introduction

Nowadays tracking different kind of activities has become common, whether it is steps taken during the day or hours slept during the night. In other words people are increasingly focused on wellbeing aspects of life, trying to find ways to avoid stress altogether or recover from it. Not just individuals but also institutions from national level organisations to private companies aim to prevent health issues and accidents. As a consequence there are plenty of initiatives for health, fitness and wellness data collection, management and use while not enough solutions.

Increasingly more knowledge of human physiology and behaviour is acquired partially through holistic approaches and partially through scientific advances. In addition, means to proof different hypothesis or destroy the myths have increased. However, that knowledge is not applied adequately to create user-friendly solutions. Instead, solutions are often developed from technology and company perspectives. This study tries to remedy that by applying user experience principles and positive computing methods in the development of a breathing application.

1.1 Background

Exploration of different building blocks for breathing application is a third phase in the Double diamond experience design process called Develop phase as shown in Figure 1 below. First two phases were part of the Master's thesis in Industrial Management and narrowed down user experience principles for developing health and fitness wearables in the Define phase (Löfblom 2017). In other words, defined principles will be used for analysing prototypes implemented in Develop phase. More details will be provided of both principles and analysis, in Chapter 3 for the principles and in Chapter 5 for the analysis.

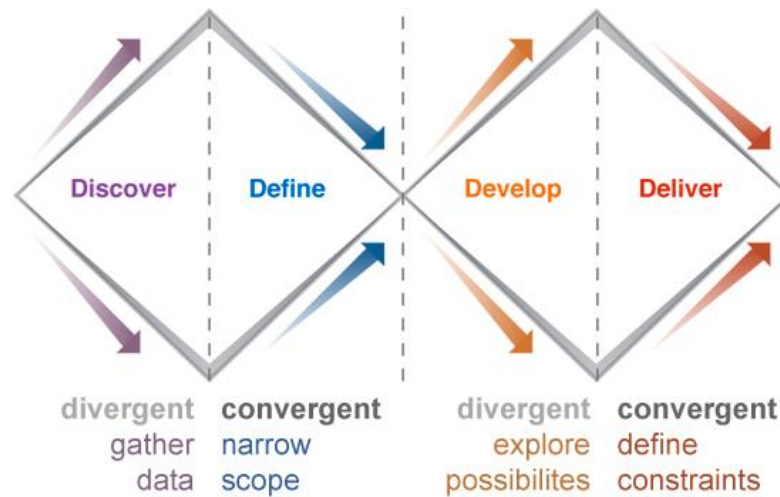


Figure 1. Double diamond experience design process. (Design Council IDEO.org 2008).

Movesense platform was selected for the implementation of prototypes, due its open development environment and affordability for do-it-yourself (DIY) prototyping. In addition, Suunto possesses extensive expertise in wearable motion sensing and tracking in harsh environments, thus their sensors are likely to be trustworthy and robust. Furthermore, Movesense sensor device is reusable hence offering long-term use options.

Breathing was selected as a first motion detection target due to its applicability to all humans as a simple way of stress relief. Breathing out relaxes body and breathing in deeply gives volume to body. In addition to breathing application, other wellness and fitness applications could be gathered into a toolkit, to be picked and used when needed.

1.2 Objective

Create prototypes of (diaphragmatic) breathing application using Movesense (sensor) platform (an open development environment for motion sensing solutions) and evaluate prototypes with respect to the principles and technical constraints to narrow options for final definition of implementable product/system. Outcome of the project will be a set of prototypes implemented using Movesense platform and evaluated against defined criteria.

1.3 Research Method

The study progressed following a typical software development process for small projects in the pre-study phase. Since the purpose of the study was to use prototyping to find how breathing application could be developed with selected Movesense platform, the availability of different options for further study was emphasised. The study progressed so that first, Movesense platform was explored and its structure described. Secondly, study of existing knowledge of breathing application creation was investigated especially how to measure breathing and design prototype for wellbeing of user. Thirdly, understanding gained from the two first parts was used in order to create breathing application prototypes. Fourthly, prototypes were analysed and next steps defined.

So in other words this study consists of six chapters. Chapter 2 describes assets of Movesense platform. Chapter 3 examines existing knowledge specific for breathing application. And then Chapter 4 combines understanding acquired from the Chapter 2 and 3 and uses it in creating prototypes of breathing application. Chapter 5 analyses outcome and reflects on it with respect to the objective. And then last Chapter 6 experiments with the ideas for the future motion tracking wellbeing applications.

2 Movesense platform

Platform can mean many things, even in the context of product platform, thus generalised definition usable for product development could be: “A platform is a collection of core assets that are reused in order to achieve a competitive advantage.” (Kristjansson et al. 2004). Such definition well describes Suunto’s intention with the Movesense platform that company itself often describes as a sensor platform and an open development environment for motion sensing solutions. According to Terho Lahtinen also plenty of resources are saved and efforts better focused where it matters (Movesense Launch 2017):

“Developing, designing, and manufacturing hardware from scratch can take years and millions of dollars, draining the momentum of startups and enterprises alike. With Movesense, companies can leapfrog development to the actual application, focusing on their area of expertise and what will actually deliver value to customers.”

Another advantage would be to engage athletes and boost niche markets currently underserved (Movesense Launch 2017):

“I’ve read estimates that there are more than 8,000 sports played in the world, but most of them are not measured at all. In every sport, however, there are participants who want to improve, and measurement can help, Movesense democratizes motion data, and by opening the SDK (development environment), we expect developers to make possible applications and data-driven insights that we haven’t even considered yet.”

Initially Movesense sensors were developed for Spartan sportwatch series as an internal project but then expanded to other uses to benefit larger group. Currently, over 800 developer kits are delivered in more than 30 countries. Furthermore 10 partners already launched their products in the market and 40 companies are close to producing products to the market. There are three main product groups, sports, health and IoT.

Suunto is certifying the Movesense sensor as class 2a medical device in Europe and once the approval is done it will help partner companies to certify their own specific health related solutions as medical devices. Existing development kits, sensors and accessories are available from the webshop (Movesense shop 2019). There is growing community supporting developers, and meetings are held every six months in Suunto office in Vantaa. And February 2019 meetup was first time arranged in Düsseldorf during ISPO sports trade fair, where Movesense platform received ISPO award in the Digi-

tal Health and Fitness category. Currently co-creating partners are searched through innovation contest open for members of the ISPO Open Innovation Community. Movesense is a registered trademark of Suunto Oy, a subsidiary of Amer Sports Corporation. However, typically partners brand their own products with their own logo as well.

Amer Sports also provides opportunities for Movesense partner companies through its Sports Tracker application. One opportunity provided is to build awareness of new product or service through advertisements to millions of users in over 200 countries around the world. Another opportunity provided is to validate even hypothesis besides new product or service with quick surveys. Third opportunity provided is integration of new service to Sport Tracker by using API or other negotiated means. (Movesense Sports Tracker 2018).

Sports Tracker is a free to use sports tracking application that Amer Sports bought in 2015 to accelerate Suunto's digital services development and strengthen Suunto's ability to provide world-class digital sports services (Suunto Sports-Tracker 2015). In 2018 Suunto announced to retire its own online training platform Movescount in 2020 and launched Suunto app instead and full transition of digital services is still ongoing. (Suunto Transition 2018). According to sports blogger Dcraimaker the new Suunto online platform is built based on Sports Tracker platform.

Soon to be finalised acquisition of Amer Sports (or more than 90% of its shares) by the Investor Consortium might bring in new resources even for development in the Suunto subsidiary. Furthermore the resource network might expand in the Chinese market according to the actual tender offer. (Dcraimaker Transition 2019) In addition, running and running camps are becoming more popular in China and there is an increasing demand for services especially provided in Chinese language(s) according to running coach and translator Kuofeng Hsu in his presentation in the First Beat HRV Summit 2019 (First Beat HRV Summit Wearables 2019). Since Anta Sports is one of the investors in the Consortium, it is interesting to see what happens in the long-term in cooperation between previous competitors. Anta is not well-known sports brand in Europe, yet, but in China it is biggest Chinese brand, Adidas and Nike being biggest foreign brands.

The result of recent events is that Movesense network is expanding and core assets of platform are getting more robust. Medical certification of sensor will benefit community with improved documentation and well defined boundaries to Movesense libraries in both sensor and mobile. Movesense architecture will be described next. For readability sensor will be called Device, not sensor or sensor device. Similarly device communicating with the sensor will be called Mobile Phone even if it could potentially be tablet, laptop, raspberry Pi processor, or such device in the real implementation. Additionally, accelerometer is used in most of the examples to improve readability.

2.1 Movesense Architecture

Movesense platform includes a tiny device with multiple sensors for measuring motion and human physiology, a set of developer tools, APIs, accessories for attaching or embedding the device, and support both from Movesense developers in Suunto and from the network of partners, developers and users of Movesense solutions for tracking motion. In other words those assets allow new Movesense developers to get their own application or service into use fast.

Essential entities of Movesense architecture are device and mobile phone together with the communication interface between entities. Libraries for both entities and sample apps using those libraries provide a starting point for new dedicated application or service creation. Core Movesense platform consists of Movesense device library and Movesense mobile library with different configuration combinations of modules and features depending on the use case. In Figure 2 device and mobile libraries are shown as grey blocks while device and mobile apps are shown as red circles.

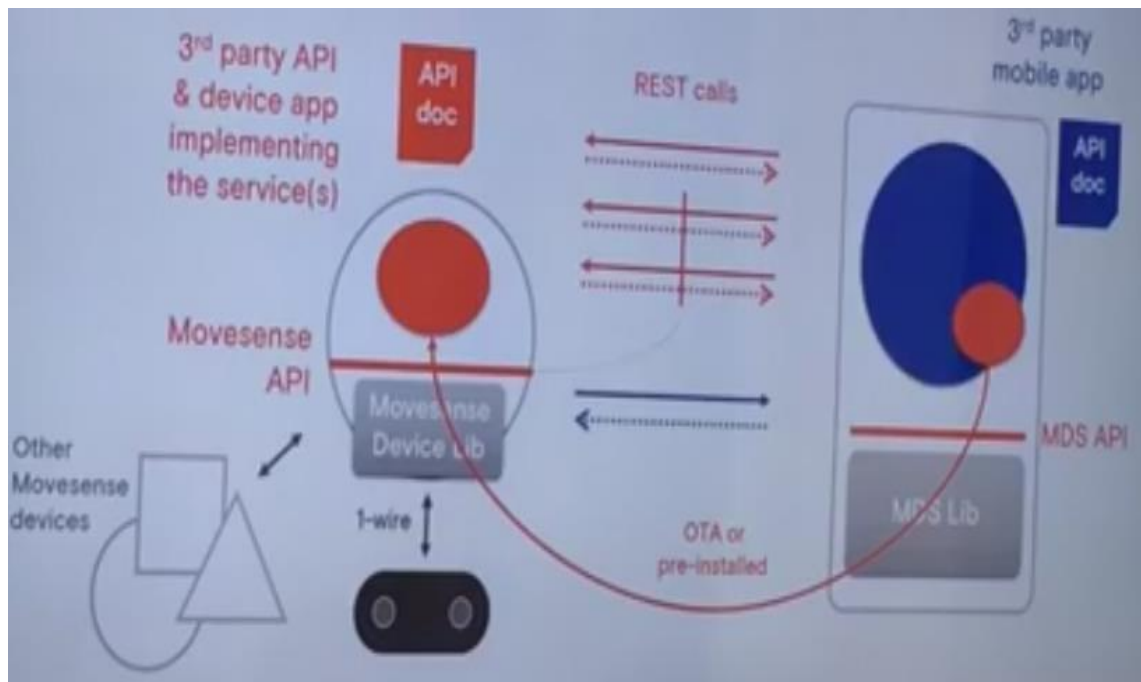


Figure 2. Movesense system architecture. (Movesense meetup1 video 2017)

Device and mobile phone form server-client connection for communication over wireless Bluetooth Low Energy (BLE) and Application Programming Interface (API) uses REST type format for (service) messages between apps. Representational state transfer (REST) technology API is based on HTTP protocol. Number of available Web APIs has been constantly growing since 2009 (ProgrammableWeb in Restcase 2015). Thus, REST type API will be easy to use for developers accustomed to use REST APIs earlier.

Device app is upgradable by Mobile app Over-the-Air (OTA) connection. Ready built sample apps for device are downloadable from Bitbucket repository to the mobile phone (app) for OTA upgrade. However mobile sample apps need to be built in chosen Android or iOS development environment after downloading apps.

1-wire connector allows snapping additional slave sensors to the device. Connectors can even have their own Gear ID making devices context aware. Other accessories like chest straps, wrist straps and clips are available. Furthermore, sensors can be taped or integrated directly to textiles e.g. tops, jackets and so on.

Before mentioned development kit contains a chest strap together with two devices and information how to connect to the Bitbucket to get access to Movesense files. After

development environment was opened in the end of 2017 it is enough to buy one device from the Movesense shop, in order to get started with Movesense development with freely downloadable Movesense files. Figure 3 below shows repositories currently available in the Bitbucket for Movesense community.

Suunto / Movesense-community

Repositories






Repository	Last updated
 Movesense-device-lib	2019-05-20
 movesense-docs	2017-09-19
 movesense-ios-mqtt-gateway	2019-04-01
 Movesense-mobile-lib	2019-05-17
 movesense-raspi-gateway	2019-03-19

Figure 3. Movesense-community repositories.

Main repositories are marked with red circle. Device files are in Movesense-device-lib and Mobile phone files in the Movesense-mobile-lib repository. Movesense-docs contains documentation and other repositories are separate community projects. Repositories in the Bitbucket can be cloned to the local repository for development. Details of main repository structures will be described in Chapter 2.2 and 2.3.

2.1.1 Movesense Device Overview

Movesense device is slightly bigger than two euro coin size and contains multiple sensors packaged into water and shock resistant black plastic. Figure 4 shows both back and front views of the device. Device uses CR 2025 lithium coin cell battery that provides even months of operating time, depending on the user application. Device is able to store measurements in the non-volatile memory, hence reducing need to keep BLE wireless connection powered unnecessarily and same time saving battery lifetime. Red led provides visual signals if programmed to do so.

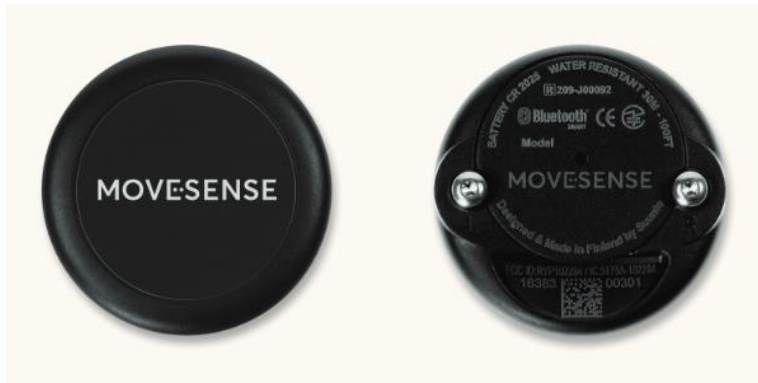


Figure 4. Movesense Device front view with logo and back view with 2 metal studs. Note: image is scaled into real size of device. (Movesense Sample Readme file)

Currently there are two types of devices one with slightly different hardware. Both types look alike, except for HR mark labelled back side of device straight on the right bottom corner under battery. Heart rate and ECG monitoring require HR type (especially in higher sampling frequencies essential). Current hardware version is G and label is visible in the back side of device. In addition to that, in the front side, there is a groove on the edge of the device, which older version did not have. Both HR and non-HR device types use same hardware version with different assembled components. Medical certified device mentioned earlier will be the third type of device, as soon as approval process finishes.

2.1.2 Movesense Accessories and Additional Tools

Several different accessories can be purchased f.ex chest straps, wrist bands, sticker connectors, 1-wire connectors or connectors to be attached to textiles. Smart connectors contain ID that makes the device context aware. (Movesense shop 2019)

Connectors are snap-in type, containing holes where device's metal studs are connected. Dimensions and other useful information is available in the Movesense Developer pages in case integration to textile or other object is needed. (Movesense connector 2019)

Device repository contains also mechanical drawing of debug pins for building own debugger circuit, if of-the-self debuggers, like JLink, are not available, and it OTA upgrade is not usable. Fastest upgrade is achieved with Programming and debugging jigs, however high cost might be an impediment. (Movesense Debug 2019)

2.1.3 Movesense Libraries and Services overview

Device repository is called Device library and it contains samples (Samples) and core library (MovesenseCoreLib) and few other subfolders like tools, migration and mechanics. All the sample apps including source code and binary files can be found in samples. And under core library, Include contains header files, and Resources contains API files in yaml-format inside subfolders called Movesense-api and Whiteboard, which contains whiteboard service definitions used for communication. Following list shows what subfolders and files are under Movesense-api:

```
/movesense-device-lib/MovesenseCoreLib/resources/movesense-api
  -info, time, types
1  /comm          -1wire, ble, ble_gattsvc
2  /component     - ds24l65, eeprom, led, lsm6ds3, max3000x, nrf52
3  /meas         -acc, ecg, gyro, hr, imu, magn, temp
4  /mem          -datalogger, logbook
5  /misc         -calibration, gear, manufacturing
6  /system       -debug, energy, memory, mode, settings, states
7  /ui           -ind
```

Listing 1. Movesense-api subfolders and yaml-format files.

Under Meas can be found measurement service definitions, and for example accelerometer provides basic REST-type services with commands, like get, put, post, delete, and with parameters, like sampling frequency and range. APIs are described more in Chapter 2.2. Under Component are hardware service definitions related to sensors, non-volatile memory and arm processor which will be described next.

2.2 Movesense Device

The Movesense uses nRF52832 microcontroller unit (MCU) from Nordic Semiconductors. The nRF52832 is an advanced low-power system on chip (SoC) including a radio for BLE 4.0, 2,4GHz communication. Multiprotocol radio has hardware support for Bluetooth 5 and is compatible with ANT protocol.

The nRF52832 SoC is designed for low-power applications and it can be used with a supply voltage from 1.7 to 3.6 volts. The central processing unit (CPU) of the nRF52832 use 32-bit ARM Cortex-M4F equipped with 512kB flash memory and 64kB

RAM memory. Size of non-volatile external memory or EEPROM is 3Mbit. (Nordicsemiconductor n.d). Sensors will be described next.

2.2.1 Sensors

Movesense uses low-power components to enable low power usage and long battery life. Following list shows product name and manufacturer of electronic components that were used in building Movesense device:

- Accelerometer and gyroscope combo LSM6DSL from STMicroelectronics
- Magnetometer LIS3MDL from STMicroelectronics
- Temperature sensor TMP112 from Texas Instruments
- Heart rate sensor MAX30003 Biopotential Analog Front-End Maxim Integrated

Linear acceleration sensor is capable to deliver useful tri-axis data about motion, acceleration and shocks, or even capable to calculate position by using AHRS algorithms or High Performance Attitude and Heading Reference System algorithms. Sampling frequencies available are 12.5Hz, 26Hz, 52Hz, 104Hz, 208Hz, but highest frequencies might be limited by BLE bandwidth, unless sensor storage is used to send bigger packets at a same time. Similarly different acceleration sensitivities can be configured, but care have to be taken not to do it during active subscription to avoid strange behavior. Ranges available are $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$ full scale. Accelerator power consumption is low, however for efficient work with accelerometer the power consumption and accelerometer sampling frequency should be taken into consideration.

Tri-Axis Gyroscope measures tilt and angular velocity in units of degree per second (dps). Similarly to accelerator sampling frequencies and range can be configured. Sampling frequencies available are same than for accelerometer, and ranges available are ± 125 , ± 245 , ± 500 , ± 1000 , ± 2000 dps full scale. And Tri-Axis Magnetometer measures magnitude of the magnetic field changes of planet. Ranges available are ± 2 , ± 4 , ± 8 and ± 16 gauss full scale.

Movesense device is equipped with analog front-end (AFE) which is capable of capturing ECG signals with one channel. And same component is used for heart rate capturing and both average heart rate and peak-to-peak interval (RR) also called inter-beat-interval (IBI) is provided. IBI or RR value can be used to quantify heart rate variability

(HRV). AFE has also stud contact detection e.g. snapping into chest strap will wake up device. Additionally temperature sensor can be used to measure device's internal temperature and give values in units of Kelvins [K] but converted to Celsius degrees [°C]. Accuracy is $\pm 0.5^{\circ}\text{C}$ (max) from 0°C to $+65^{\circ}\text{C}$ and $\pm 1.0^{\circ}\text{C}$ (max) from -40°C to $+125^{\circ}\text{C}$. Axis layout for the inertial sensors is typical as for Android mobile device, in case device is worn on the chest strap in vertical position so that device's metal studs are from left to/from right in horizontal position. Following list explains position in each axis X (right), Y (down), Z (through device):

- positive x-axis is to the right while wearing it
- positive y-axis is downwards while wearing it
- positive z-axis is through the device while wearing it

For calibration purposes there is calibration API in the Misc subfolder under Resources, that instructs how to set (put) device in calibration state for three minutes and set mode between 0 and 3 from, no mode set (0), predefined tilt sequence (1), predefined magnetic field sequence (2) or random tilt sequence (3). In Mode 3 device changes the state itself but in modes 1 and 2 calibrator has to set mode. States are Idle (0), sequence on (1), step on (2), step done (3) and sequence done (4), and for last state timeout may need tuning since it takes quite long time to get response.

Additional instructions about calibration were given in the stack overflow under Move-sense tagged questions and answers. Answer was given by an experienced user working with Movesense (in Suunto):

"There is no "built-in" bias calibration in the framework/API. The reasoning is that it would take memory from everybody (not all apps need it) while not working for all the use cases. Bias-removal is also a feature with quite straight forward implementation:

- Make sure that sensor is still for the duration of the bias calibration. The duration depends on application, sample rate and accuracy required (this very application specific).
- Measure Acc + Gyro (i.e. /Meas/IMU6) with the sample rate that you normally use
- run sanity check on the acc & gyro data you receive for example stdev of each axis (no movement, nor rotation!)
- calculate average of each axis => this is the bias correction that you need to subtract from the actual measurements before you do anything else with them.

Since both average and stdev can be calculated incrementally there is no need to reserve RAM-buffers for the calibration.”

Besides sensors in the device also additional sensors can be connected over the 1-wire bus by snapping device to connector with 1-wire capabilities. Polarity matters in the bus so better know which one is voltage and which one ground when connecting. Again connector adds context awareness and could be added to sports gear so that device would know gear by its gear ID and service it accordingly. Device APIs will be described next.

2.2.2 Device API

Device APIs are also called Movesense APIs, since they are used for both device and mobile, and they are documented in Readme.md file under Device repository. However there are two kinds of APIs external and internal, of which external ones are first described and then most important internal APIs as well. For readability, APIs are split to measurement related, system related and other APIs, since the original list is very long. First each table is shortly described and then accelerometer API is described in more detail with listings from Acc.yaml file. API files and folders are located under Move-sense-api and this is the complete path from the root of device repository:

/movesense-device-lib/MovesenseCoreLib/resources/movesense-api

Each sensor has measurement services it provides to client and additionally there is imu service that contains all inertial sensor service definitions in same file. All measurement services are shown in the Table 1 below.

Table 1. Device APIs enabling subscribing measurements.

Resource	Description
/Meas/Acc	API enabling subscribing linear acceleration data (based on accelerometer).
/Meas/ECG	API for the electrocardiography measurement.
/Meas/Gyro	API enabling subscribing angular velocity data (based on gyroscope).
/Meas/HR	API enabling subscribing heart rate data.

Resource	Description
/Meas/Imu	API for synchronous access to motion data (accelerometer, gyroscope, magnetometer)
/Meas/Magn	API enabling subscribing magnetic field data (based on magnetometer).
/Meas/Temp	API enabling reading or subscribing temperature data (based on thermometer).

System related services, that are common to all apps, are listed in Table 2 below. System APIs include battery state, memory, system state reading and subscribing to debug services or controlling the main system state or setting device parameters.

Table 2. Device APIs enabling subscribing for system services.

Resource	Description
/System/Debug	API for subscribing messages from device.
/System/Energy	API for reading the battery state.
/System/Memory	API for reading memory state.
/System/Mode	API for controlling the main system state (e.g. factory sleep).
/System/Settings	Settings API for a Movesense device.
/System/State	API for reading some states.

Other different types of services are listed below in Table 3. Those APIs include Bluetooth management, accessing time and device information, reading or writing memory, storing data with Datalogger or reading it from Logbook, and even getting unique ID for gear that device is attached. We will look into the content of one of the API files and to go through service API commands in used for accelerometer services.

Table 3. Device APIs enabling subscribing for other services than measurements and system.

Resource	Description
/Comm/Ble	API for managing BLE.
/Component/Eeprom	API for writing and reading the EEPROM memory/ies.
/Info	API for accessing generic device information.
/Mem/DataLogger	Generic logger capable of logging max. 8 different resources.
/Mem/Logbook	Generic Logbook from where the logged data can be read.
/Misc/Gear	API to get the globally unique Movesense ID associated with the physical gear.
/Time	API for accessing different time related services.
/Ui/Ind	API for controlling the LED.

File format used for API files is YAML (YAML Ain't Markup Language). Yaml is a human-readable data-serialization language. It is commonly used for configuration files, but could be used in many applications where data is being stored (e.g. debugging output) or transmitted (e.g. document headers). (Wikipedia) It is generated by using Swagger and current version is 2.0.

The API of each Movesense app, module, service and component will have definition in the form of a YAML file with the following structure:

```
1 info: - version, title, description, x-api-type, x-api-required
2 paths:- command: description, parameters: name, in, description, required, schema, $ref;
responses: description
3 parameters: - $ref parameter name: name, in, required, type, format
4 definitions: - $ref definition name: required, properties: name: description, type, items: type,
format
```

In the YAML file structure info is described first in Line 1. Then in Line 2, paths to all API related services like Acc and their commands like put or post are described and if they have \$ref path to either parameters like SampleRate or definitions like AccData then those are described as in Line 3 and Line 4 respectively. As an example, put command for Config service command for Acc API is described below:

Paths:...

```

/Meas/Acc/Config:
  get:...
  put:
    description: |
      Set linear acceleration measurement configuration.
    parameters:
      - name: config
        in: body
        description: New configurations for the accelerometer.
        required: true
        schema:
          $ref: '#/definitions/AccConfig'
    responses:
      200:
        description: Operation completed successfully
      503:
        description: |
          Not allowed to change configuration at the mo-
          ment. E.g. when active subscriptions exist.
...
Definitions: ...
  AccConfig:
    required:
      - GRange
    properties:
      GRange:
        description: |
          Acceleration range for example if set 2 the range is -2..+2 G
        type: integer
        format: uint8

```

Note as a special case parameters can be on the command level in the API structure and not as attributes to the commands like in SampleRate or SampleRate Subscription services as can be seen in below API:

```

Paths:...
  /Meas/Acc/{SampleRate}:
    parameters:
      - $ref: '#/parameters/SampleRate'

  /Meas/Acc/{SampleRate}/Subscription:
    parameters:
      - $ref: '#/parameters/SampleRate'
...
parameters:
  SampleRate:
    name: SampleRate
    in: path
    required: true
    type: integer
    format: int32

```

Other Acc services and their possible commands are listed below:

- 1 Info: - get
- 2 Config: - get,put
- 3 {SampleRate}: – parameters
- 4 {SampleRate}/Subscription: - parameters, post, delete

Command post of SampleRate subscription service subscribes to periodic linear acceleration measurements, and command delete unsubscribes from the service. More details of these Movesense-api services can be read from the device repository.during development.

Another important source of APIs in the resources is Whiteboard. It is a communication library containing services, clients, timers, threading and external communications. Whiteboard threads are called ExecutionContext and there are two which are called application and meas, no other ExecutionContext are in use currently.

LaunchableModule is so called whiteboard-aware module which runs in an ExecutionContext (wb thread) and it has lifecycle callbacks called initModule, startModule, stopModule and deinitModule.

Then there are two server-client related components called ResourceProvider, which is wb REST service which has request callbacks like onGetRequest, onPutRequest and so on, and ResourceClient, which makes requests to internal and external whiteboard services using request methods like asyncGet, asyncPut, an so. More about whiteboard can be looked at the resources in device repository. The structure of device app providing accelerometer data to client will be looked at next.

2.2.3 Device App and Modules

C/C++ is used for programming with some limitations derived from limited resources of thread stack, cstack and RAM. Moreover, standard template library (STL) is not used, nor dynamic memory. Asynchronous APIs automatically optimise power consumption, but developer must avoid hogging the execution by making busy-loops or such. Call – callback -structure is used, and REST-type commands with additional publish-subscribe pattern will be used in programming device app.

Following sample app IntegerAccelerometerApp.cpp in Listing 2 shows app structure used, especially text in bold font emphasising app specific parts. Most of the core features called core modules are optional and need to be set true in order to include those into the build. Here only Bluetooth service (BleService) and communication (BLE_COMMUNICATION) are enabled. By default only Startup provider is included in app with empty provider list between BEGIN and END providers. However, if service needs to be run within Startup provider module it needs to be added into the list in MOVESENSE_PROVIDER_DEF, and count of providers in both BEGIN and END is incremented accordingly. Naturally, IntegerAccelerometerService.h header file needs to be added too.

```
#include "IntegerAccelerometerService.h"
#include "movesense.h"

MOVESENSE_APPLICATION_STACKSIZE(1024)

MOVESENSE_PROVIDERS_BEGIN(1)

MOVESENSE_PROVIDER_DEF(IntegerAccelerometerService)

MOVESENSE_PROVIDERS_END(1)

MOVESENSE_FEATURES_BEGIN()
// Explicitly enable or disable Movesense framework core modules.
// List of modules and their default state is found in documentation
OPTIONAL_CORE_MODULE(DataLogger, false)
OPTIONAL_CORE_MODULE(Logbook, false)
OPTIONAL_CORE_MODULE(LedService, false)
OPTIONAL_CORE_MODULE(IndicationService, false)
OPTIONAL_CORE_MODULE(BleService, true)
OPTIONAL_CORE_MODULE(EepromService, false)
OPTIONAL_CORE_MODULE(BypassService, false)
OPTIONAL_CORE_MODULE(SystemMemoryService, false)
OPTIONAL_CORE_MODULE(DebugService, false)
OPTIONAL_CORE_MODULE(BleStandardHRS, false)
OPTIONAL_CORE_MODULE(BleNordicUART, false)
OPTIONAL_CORE_MODULE(CustomGattService, false)

// NOTE: It is inadvisable to enable both Logbook/DataLogger and EepromService without
// explicit definition of Logbook memory area (see LOGBOOK_MEMORY_AREA macro in move-
sense.h and eeprom_logbook_app).
// Default setting is for Logbook to use the whole EEPROM memory area.

// NOTE: If building a simulator build, these macros are obligatory!
DEBUGSERVICE_BUFFER_SIZE(6, 120); // 6 lines, 120 characters total
DEBUG_EEPROM_MEMORY_AREA(false, 0, 0)
LOGBOOK_MEMORY_AREA(0, 384 * 1024);

APPINFO_NAME("Sample acc 16 bit data app");
APPINFO_VERSION("1.1.0");
```

```

APPINFO_COMPANY("Movesense");

// NOTE: SERIAL_COMMUNICATION macro has been DEPRECATED
BLE_COMMUNICATION(true)
MOVESENSE_FEATURES_END()

```

Listing 2. IntegerAccelerometerApp.cpp contents listed.

Some debug buffers, logbook and non-volatile memory allocation are required for app if it needs to provide data for PC Simulator use. This sample app was recently modified to use 16 bit integer numbers instead of 32 bit floating point numbers for accelerator values, and thus was chosen as an example of accelerometer use.

File structure for IntegerAccelerometerApp sample app can be seen in Figure 5 below taken from the local repository but identical to Bitbucket device repository contents. Root folder for sample app is called accel_int16_app and it contains the source code for sample app. Basic functionality of content is following:

- CMakeLists.txt with instructions how to build app
- IntegerAccelerometerApp.cpp main app (see Listing 2 above)
- IntegerAccelerometerApp.yaml with main app API definitions
- IntegerAccelerometerService.cpp Service app
- IntegerAccelerometerService.h header file for service app
- Service-api-spec –subfolder is a whiteboard resources subfolder
 - IntegerAccelerometerService.yaml with app specific API definitions
- ResourceForwardingTracker.cpp additional app
- ResourceForwardingTracker.h header for additional app

movesense-device-lib > samples > accel_int16_app > Search ac...









Name	Date modified	Type
 service-api-spec	16.5.2019 12:03	File folder
 CMakeLists	16.5.2019 12:03	Text Document
 IntegerAccelerometerApp.cpp	16.5.2019 12:03	C++ Source
 IntegerAccelerometerApp	16.5.2019 12:03	YAML File
 IntegerAccelerometerService.cpp	16.5.2019 12:03	C++ Source
 IntegerAccelerometerService.h	16.5.2019 12:03	C/C++ Header
 ResourceForwardingTracker.cpp	16.5.2019 12:03	C++ Source
 ResourceForwardingTracker.h	16.5.2019 12:03	C/C++ Header

Figure 5. IntegerAccelerometerApp sample app file structure.

Typically subfolder Service-api-spec is called wresources containing app specific API file and if that is not needed it contains empty resource.cpp file for build purposes so that CMake is able to generate core libraries. API will be explained in the next chapter. However, first contents of sample app definition file IntegerAccelerometerApp.yaml is shown in Listing 3 below, describing execution contexts (executionContexts) and APIs (apis). Typically two types of execution contexts are used one for measurement (meas) and one for application. Each execution context has lists number of requests, responses, datapoints, and stack size, execution priority, external thread state which here is true since application service is running as an external thread within Startup provider module. APIs lists name of additional api services which here is IntegerAccelerometerService, api identification number, and default execution service used which here is measurement execution context.

```
# Type of the document

wbres:
  version: "2.0"
  type: "root"

# Execution context definitions
executionContexts:
  application:
    numberOfDpcs: 8
    numberOfRequests: 20
    numberOfResponses: 25
```

```

    externalThread: true # we run this execution context in main
thread
    stackSize: 768
    priority: normal
meas:
    numberOfDpcs: 9
    numberOfRequests: 25
    numberOfResponses: 10
    stackSize: 768
    priority: normal

apis:
    IntegerAccelerometerService.*:
        apiId: 100
        defaultExecutionContext: meas

```

Listing 3. IntegerAccelerometerApp.yaml sample app definitions.

Note that app_root.yaml file which is a typical app definition file contains configuration for the whiteboard which is part of the Movesense and to change it without known exactly the effects might change the stability of the platform. However, in this example app app_root.yaml was replaced by IntegerAccelerometerApp.yaml file and its service extends general app_root.yaml content to include apis section at the end of the file defining the service like IntegerAccelerometerService and all file types of it to use whiteboard service called meas for measurements by default. Whiteboard communication library will be described in next section. Device communication is discussed next.

2.2.4 Device Communication

Bluetooth low energy (BLE) is used in communication between device and mobile phone. Current BLE version is 4.0 and next major device app version will have BLE version 5 which means speed, range and broadcasting improvements. Compared to Bluetooth 4.2, Bluetooth 5 has (semiconductorstore n.d):

- 2x the data rate
- 4x the range
- 8x the broadcast capability

Improvements will mean increase of range from 50 meters to 200 meters (or inside 10 meters and 40 meters respectively) and eight times bigger data packets will double the

data rate. Combined with the Link Time Optimisation (LTO) provided by the coming compiler GCC 9 it will make OTA upgrade five to six times faster than with current tools (Meetup 15 May 2019).

BLE protocol is divided in different layers as seen in Figure 6. Application layer can directly communicate only with the Generic Access Profile (GAP) layer defining roles for device communication. Broadcaster sends advertisements and observer receives packets. Peripheral sends information for connection creation and central sends request to establish connection with peripheral. Multiple GAP roles can be used at the same time in BLE protocol stack. (TI n.d)

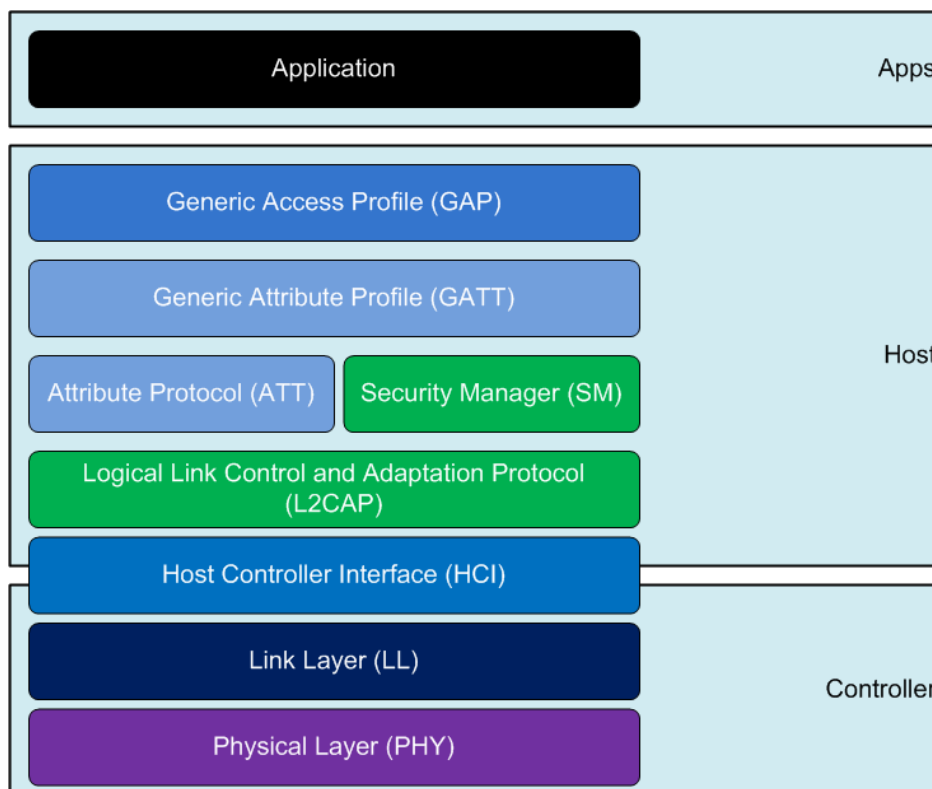


Figure 6. Bluetooth Low Energy protocol stack. (TI n.d)

GATT is the second highest BLE protocol layer built on Attribute Protocol (ATT). GATT includes Services and Characteristics, like for example Battery Service using the battery level as a characteristic. GATT deals only with actual data transfer procedures and formats. GATT provides the reference framework for all GATT-based profiles, which

cover precise use cases and ensure interoperability between devices from different vendors. (Bluetooth n.d)

There are four different types of advertising packets or Protocol Data Units (PDU):

First three PDUs are used for indirect broadcasting:

- Connectable ADV_IND,
- Scannable ADV_SCAN_IND, but not connectable
- Non-connectable ADV_NONCONN_IND, but no scan response, and
- Connectable ADV_DIRECT_IND, which can be used directly to intended receiver sending only 6 Byte device address of intended receiver and no scan response.

Advertisements can contain up to 31 Bytes of data called payload and same amount in scan response to devices scanning for advertisements. Up to three channels can be used for advertising, and advertising interval in the same channel can be between 20ms and 10.24 seconds. In other words, different combinations of channels can be used and time between consecutive advertisements can be up to 10,24 seconds, depending on balance between power saving and interaction needs. Wi-Fi causes least interference in advertising channels 37, 38 and 39. Figure 7 below shows timing of packets during advertising.

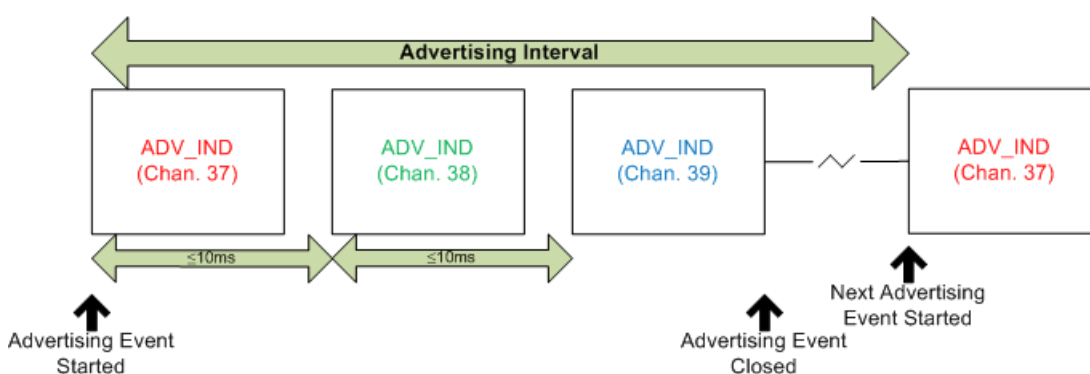


Figure 7. BLE advertising interval, three channels allocated for advertisements, and maximum advertising time per channel. (TI n.d)

Bluetooth advertisement was originally intended for the discovery of the peripheral devices, but it can be used for transmitting data. Field tests conducted by Portaankorva

confirm its usability for real-time monitoring of the team sports, ideally indoors with shorter distances, and with data recorded in the internal memory prior broadcasting it. However quality validation requires further testing. (Portaankorva 2018).

2.3 Movesense Mobile App

Mobile phones are typical display devices for sensor data but tablets and laptops can be used as well provided those incorporate Bluetooth. Among different operating systems Android and iOS are best supported with buildable sample apps. Developers in the community have provided additional ways to create apps with Xamarin Visual Studio (plugin made by AndyCW) or Unity (plugin). Apps are buildable from provided source code to get sensor data for further analysing. With the exception of Showcase app which is already built and downloadable from mobile repository (Downloads). Display devices also take care of updating device app over-the-air (OTA).

2.3.1 Mobile Library

Mobile phone repository is called mobile library, in other words Movesense Device Service (MDS) library. It consists of subfolders for Android and iOS operating systems. Under those subfolders there are multiple sample codes, for Android Movesense aar-format file, Showcaseapp and then samples with four different types. Root folder can be cloned to local repository for building it and then apk-file can be downloaded and installed in the mobile phone for testing for example subscribing for measurements from device.

2.3.2 Mobile API

Similar symmetrical REST-type API services and commands are used by the mobile app. Those device APIs can be found in Readme.md file in device repository. Moreover available API services were introduced in Section 2.2.2. However, following services and commands are seen in the mobile phone screen when using Showcase sample app that has connected to the device and subscribed to receive data:

Reading values from sensor

- Values from sensor displayed in textView:

- Get Request
- Led Sensor
- Heart Rate

Values from sensor displayed in graphView:

- Linear Acceleration
- Magnetic Field
- Angular Velocity

Next section will described mobile apps.

2.3.3 Mobile App

After library dependency is added to the application project, run the application in the Android Studio. And build the sample app or alternatively use Showcase sample app which is directly downloadable from the mobile repository as apk file. How to connect with mobile app to device will be explained next.

Scanning for devices runs automatically after start of the app. When Android BLE Adapter finds the device it will be shown in a list. And when device is visible on the list, device is clickable and connection to the selected device starts. After connection, it is possible to create Get Request in most of the scenarios by clicking on GET Button.

In led sensor device/app, there is a switch for on/off led. This request will return single value from device or error if something went wrong. Then, it is possible to create Subscribe Request by moving / clicking on switch component. This request will return values from sensor until Unsubscribed by moving switch to the left side or error if something went wrong.

There are different mobile sample apps to build and more can be found in the documentation in the mobile repository. It is also advisable to look what mobile phones have been tested with Movesense and whether those passed the test.

2.4 Setup Movesense system

Setting up Movesense systems usually starts by selecting latest Movesense mobile sample app from mobile store and then uploading also Nordic Semiconductor tool for device upgrade using OTA DFU packages. Then when those are installed also device sample app binary can be uploaded from the device repository (or local repository) to the phone and selected for upload once connection between device and mobile phone is initiated and successful. Next section will describe process in more details.

2.4.1 Upgrade Apps

Mobile sample apps can be found in app stores but if there is a need to get one from sample repository then it needs to be built first, unless Showcase sample app is used, which can be downloaded from Mobile repository Downloads (in the left side menu). Currently Android sample app is renamed to Showcase sample app and latest version is 1.9.6. Movesense perform some tests with different mobile phone brands and versions and test results of compatibility can be found in the mobile repository either movesense wiki or link from Readme.md file in mobile repository. For connecting GATT sample deviceapp, in order to get standard GATT packages, also nRF Connect mobile app could be used, or any other device capable of getting BLE packages.

Device app can be updated OTA by using either mobile phone app or Nordic Semiconductor's nRF Toolbox app which is downloadable from the app stores for free. DFU packages require nRF Toolbox app and apk type files will be installed with Android mobile phone app as shown in the Figure 8 below. Make sure that mobile app and device app are able to communicate by BLE, in other words in close proximity to one another.

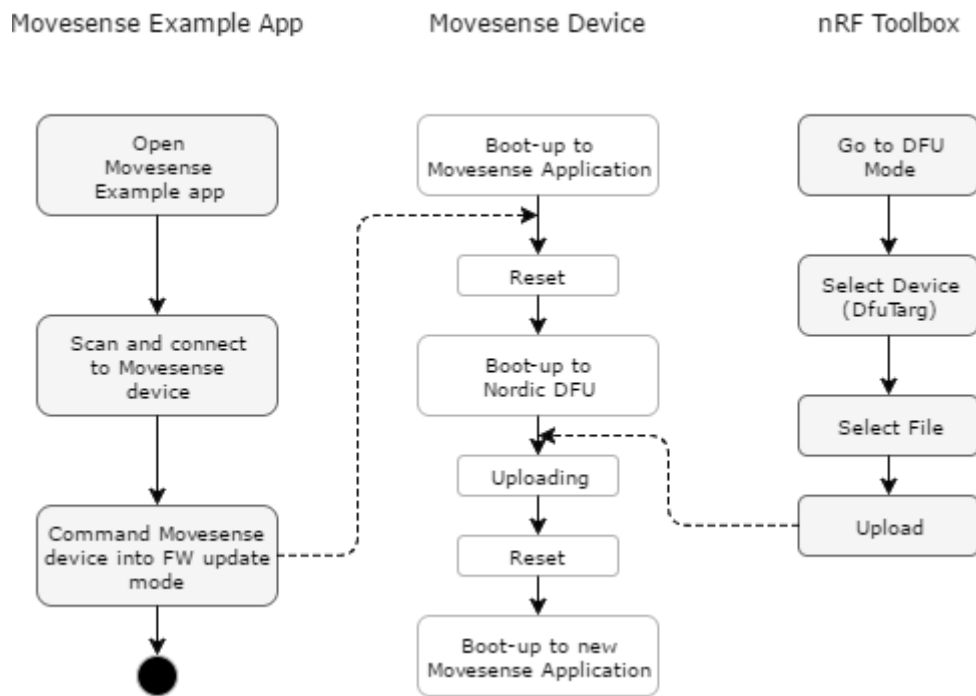


Figure 8. OTA upgrade process for device using DFU package.

Bootloader updates and hardware changes are always problematic for OTA upgrade so it is always good to be cautious and check change log and versions of device app (firmware), and device hardware. Latest hardware version today is G and device app version 1.9.0. There might be way to reset device that has failed upload by “finger dance” on metal studs in the back of device, so it might be worth to try to ask Movesense support more instructions in case upgrade fails and device becomes unresponsive. Bootloader uploads might require use of jig tool, but again some service might be asked from community members without buying own tool.

2.4.2 Setup Device Build Environment

Toolchain instructions are found in the Device repository in the readme.md file. For latest build device build version gcc compiler needed to be update, thus when downloading new device versions from the device repository it is recommended to check possible changes in tools as well.

Setup instructions contain Android, iOS, Linux and Windows operating systems. Additionally Vagrant installation is fastest and sets up isolated development environment, however it is referred as experimental in the instructions, mostly due that it is community provided and not written by Movesense. However according to Vagrant instructions there is some benefit to use it: “Vagrant provides an easy, reproducible environment for setting up development environments. It provides an isolated environment with all the dependencies set up without any version clashes or corner case bugs. However, how to manually install JIG support need to be checked from Linux installation instructions for Vagrant.”

In short Windows setup requires following tools that can be found from links:

- [GNU Toolchain for ARM Embedded 2017q4](#)
- [Ninja build tool](#) (to be placed in C:/bin and as PATH variable)
- [cmake >=3.6](#) (exe location as PATH variable)
- [nrfutil package & python 2.7](#) for OTA firmware update package creation (Nordic provides pre-compiled Windows executable)
- [Visual Studio Redistributable 2015](#)
- install pyyaml package using pip (In case: ImportError: No module named yaml)

2.4.3 Setup Mobile App Build Environment

After cloning sources from repository Android Studio can be run. There is official Android Developers site in case more guidance is needed:

<https://developer.android.com/studio/index.html>

And Movesense Android App Developer Guide is here for more information how to set environment, for example integrate MDS library (module) in an Android app and project:

<https://bitbucket.org/suunto/movesense-docs/wiki/Mobile/Android/Main.md>

Before using Google Drive Api, there is a need to generate own Api key using debug.keystore file. debug.keystore is generated by Android studio. Default Path:

C:/Users/USER/.android/debug.keystore

Generate Api key using debug.keystore:

Google instructions: <https://developers.google.com/drive/android/get-started>

Google developer console:

https://console.developers.google.com/flows/enableapi?apiid=drive&credential=client_key

Application using library in .aar format, need to put library in to the root of application folder. After copying library to the root of application folder, dependency needs to be added to Android Studio project. Example .aar file: mdslib-1.0.0.aar. In the application Module build.gradle add library dependencies.

2.5 Build Movesense Device Apps

Make myBuild subfolder under the local device repository (under app subfolder) to build device app there. As an example building hello_world sample app with cmake C/C++ -compiler using debug tag uses following commands and parameters:

```
> cmake -G Ninja -DMOVESENSE_CORE_LIBRARY=../MovesenseCoreLib/ -DCMAKE_TOOLCHAIN_FILE=../MovesenseCoreLib/toolchain/gcc-nrf52.cmake
../samples/hello_world_app
```

Release build instructions can be found in Readme.md file under device repository. Build DFU package for OTA upgrade like this: > ninja dfupkg, which places the OTA package named movesense_dfu.zip in the build folder.

Bug reports and other feedback are needed to allow faster recovery and influence in road map of Movesense. Issues need to be raised in Bitbucket. Stack overflow also have Movesense related questions and answers, and questions assigned there need to be tagged with 'movesense' (i.e. include [tag:movesense] in the question body). Device repository have FAQ.md for Frequently Asked Questions which is edited online with Bitbucket.. So good practice is to check whether question already has an answer or whether it is already known issue prior re-raising it.

2.5.1 App Development Recommendations

Starting point of development is to know what kind of data is needed by the user and then figure out which tools and methods work best for specific use case. App creation can be started from one of the sample apps by selecting one that is most suited for the data collection. One possibility for data collection is to measure with Showcase App, if

possible, and then use DataLogger as a fallback (Android samples/DataLoggerSample). Then setup the system and look at the data collected, try to find the right sampling rate to serve app development. Remember the limitations of the system for sensors the selectable G-ranges, sensitivity, and so on. Whereas for data memory there is 400kbps bandwidth and limited size, same with BLE theoretically 128kB/s but in reality less. Get to know manufacturers data sheets and protocol documentation when needed.

Data signals must be calibrated in the code, and some of the steps to do that were already given in Chapter 2.2.1 describing sensors. If data filters need to be tuned find the parameters or change the filter altogether. Again it might be easier to start with higher level languages prior going to change code in the device level. Simulate, if possible to proof filtering solution works and then port it to device. Filters in C/C++ might be easier to include in the device code. Some references to Butterworth IIR filters were given in the code, and tuning those existing filters might be enough for the app. Furthermore, it often saves time to reuse code.

More experienced developers suggested making figure eight for magnetometer and closed loop to gyroscope and accelerometer to ease the calibration effort. The point being to use integration to zero with closed loop and keep gravity steady but 2D level integrates to zero. Accelerometer needs to eliminate gravity. Gyroscope needs daily calibration for slow, random bias changes. And Magnetometer has constant offset due soft and hard iron effects and linear and non-linear effect caused by plane rotation, except Y direction unless in North Pole. So for magnetometer circling device in pattern of eight allows easy calculation of average value. Integral of gyroscope value compensates accelerometer values. And there is one degree freedom in rotation. In order to eliminate noise coming from the movement, for gyroscope start position then turning device for example on the top of the table and return to starting point to stop and check gyroscope data. Also for accelerometer calibration in walking or ice-skating 45-degree angle could be used for movement (gait, skating sweep) and then at stop with feet in same position and speed would go to zero in optimal case, thus error in each measurement point could be eliminated. So in other words start and stop in the same place.

So get the movement data from device sensor first, create the algorithm prior, and only after that import it to device. And then first modify mobile app, and after that, device app in simulator, and at the end device app itself. However, it depends on developer

skills, what is easy, and what is not, but generally higher the abstraction level of programming language more easy it is to understand.

If one is working with multiple devices and need to synchronize those, then one can do it by creating recognizable signal in all of the devices at the same time, to catch it in the logs easier. Additionally, two devices capable of waking up by touch to studs, could be synchronized, by touching studs with another device's studs., However, lag in sensor (seed) or in network or elsewhere might cause lagging, and thus delays. Mixing old and new mobile phones could be quite asynchronous accelerometers. Synchronizing over WI-Fi could be tried rather than direct BLE messaging, and has been tried in FSenSync project by Fröger. More details of his project can be seen in Chapter 2.6.2.

Power saving is important design factor, not so much in debug version but in release version it is recommended to let device sleep when idle to save battery life. Especially BLE connection is power hungry. While there is some difference between streaming big data and not sending any data during connection, even keeping the BLE connected uses quite a bit of power.

Movesense's core software architecture has been designed to be asynchronous and event based. That naturally leads to good results for power saving under normal operation. If there are no active subscriptions, the processor sleeps and the sensor enters "power OFF" mode. The sensor will be enabled as soon as there is data request, either internally or over Whiteboard communication. While the basic architecture is designed for power saving, there are a few things that must be taken into account to achieve good power consumption figures.

There is new battery measurement method implemented in device app version 1.5.0. Capacity calculation is based on voltage and battery's internal resistance. For this device app version path system/energy contains also internal resistance information provided in ohms. With every calculation, battery measurements accuracy is getting higher, so good practice is to measure battery periodically to have better view of battery capacity. Battery level is provided in 10% steps except last 10% which are divide into 1% steps. Note, that in extreme low temperature voltage seems to be zero even it is not, this is due chemical processes generating power that are slow to stabilize at the occurrence of change like hit or shake.

To have the most precise values from battery, measurement has to be performed without any additional load. Good practice is to unsubscribe all power consuming features before getting battery level (example: accelerometer, gyroscope, magnetometer).

There is no possibility (Hardware limitation) to use magnetometer or gyroscope without turning on accelerometer. Accelerometer is responsible for data transfers from IMU.

2.5.2 Device Simulator on PC

Device Simulator is called PC Simulator in the documentation which can be found in Documents repository under EmbeddedSoftware subfolder. In the same folder is Build how-to subfolder for setting up the Simulator build environment. But same tools than for device build environment are needed except version for ARM tool [GNU Toolchain for ARM Embedded \(5-2016-q3\)](#), and [Visual Studio 2015 or 2017](#). The simulator build creates a Windows executable that is based on the Movesense simulation framework. And no other operating system, nor Visual Studio version are currently supported.

Benefits for building device code in Simulator are first of all that usable memory size is much larger, so no need to think about code optimisation to fit in device itself. Secondly, errors in functionality have less dramatic effects in simulated environment than in the device. Thirdly, the real data what needs to be measured can be tested with Simulator. Or simulated device data can be used, although it is simple 1G acceleration, no rotation, fixed magnetic field, etc. And all csv files, with real or simulated data, need to be in the working directory (under sample app subfolder). For example Showcase app can subscribe to all sensors for data, and store it in csv format in the mobile phone's Movesense folder.

However few modifications need to be made in csv file in order to use it in simulation. First of all name of the file needs to be changed for example for IMU sensors name needs to be accelerometer.csv, and it needs to be in the working directory (sample app subfolder). And then, in csv file two first rows are reserved for headers and actual data must be separated by comma like in example given below for accelerometer data:

```
LoopingTimestamp: {number in ms}
Timestamp, /Meas/Acc/x: Accelerometer x-axis data [m/s^2], /Meas/Acc/y: Accelerometer y-axis data [m/s^2], /Meas/Acc/z: Accelerometer z-axis data [m/s^2]
1141504.0,509689,9.588383,0.734623,
1141581.0,.21152,9.502238,0.480975,
1141658,0.368508,9.475916,0.538404,
```

In case no actual data is available, Simulator can provide test data.

On the other hand, there are disadvantages to Simulator use, that must be considered, for example if it is needed to build Bluetooth capability or interface for 1-wire or Gear/ID or wake-up the device with some triggering event, those features are not supported in Simulator. Furthermore simulated environment is close to real one but still lagging, for example not all system states are supported and tick rate is slightly different: 1000 Hz whereas, for device it is 1024 Hz.

In order to build simulated device app in Visual Studio few things need to be set up first. Start by cloning device repository and creating myBuild subfolder for build files in the local repository (under sample app folder). Then compile with cmake either debug-ger or release version in the myBuild folder. Note that release needs to be explicitly chosen by adding parameter: `-DCMAKE_BUILD_TYPE=Release`. Last build step is to run ninja on the command line.

In order to access simulated device app, once it has started, and registered whiteboard communication on TCP port 7044, whiteboard command line tool `wbcmd.exe` is used for sending configuration and subscription commands to simulated device. Info command will give port information. Simulator UI shows the tick counter, used/free RAM, current working directory and i/o componets which are LED (right square on the bottom of the screen) and analog swith visual presentation. Here are some whiteboard commands:

```
wbcmd.exe --port TCP127.0.0.1:7044 --path /Info
wbcmd.exe --port TCP127.0.0.1:7044 --path /Meas/Acc/13 --op subscribe
```

Finally, after source code has been modified and needs to be run, build can be made within Visual Studio, as long as simuBuild subfolder is created in the local repository (under sample app folder) for it prior building.

2.6 Movesense Community Projects and Plugins

First Movesense partners tested Beta versions of Movesense in the beginning of 2017. First meetup in summer 2017 presented few demos ranging from rehabilitation guidance to fencing training. Movesense has also been presented in different global trade fairs like CES or sport trade fairs like ISPO, bringing cooperating partners to network.

Following example solutions present various use cases of Movesense assets and provide both ideas and tools for new application development.

2.6.1 Holon.ist Open World Mobile Music Platform

One interesting platform using Movesense sensors, as well as many others is Holon.ist, an open world mobile music platform that lets users create their own musical mappings from real-world data. Supported Apple iOS devices all feature 9-axis motion sensing (imu sensing) with sensor fusion. One view of app is shown in Figure 9 on the right side. Holon.ist data sources include:

- Altitude and atmospheric pressure
- User defined postures
- Custom locations with distance and azimuth
- GPS based readings include course, speed, lon/lat
- Local weather and solar/lunar tracking
- HealthKit: flights climbed, walked distance, heart rate variability etc.
- IoT data with JavaScript editor

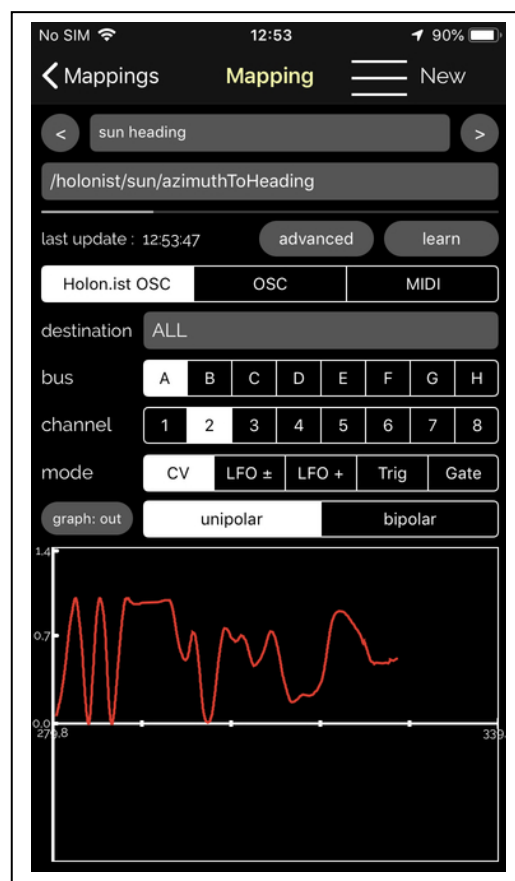


Figure 9. Holon.ist mobile screen view.

Besides above described features, sensors can be used to control VCV Rack. And other Integrations with Ableton Live, Reaktor and Eurorack module from Instruments of Things offer easily expandable system. Music creation is supported with adaptive tools like native support for Open Sound Control (OSC) communication protocol with two-way communication between devices, and Musical Instrument Digital Interface (MIDI) over USB, Bluetooth, Wi-Fi, and Inter-App Audio. In other words it is a mixed reality solution for electronic music creation from movement and other mapped sources, thus

offering interaction with environment in immersive way. (Movesense FB 2019) Look for Holonic Systems for further details of its different parts.

2.6.2 FSenSync Synchronised Sensing Over Wi-Fi

Another sensor fusion project, presented by Fröger K. in the second meetup end of 2018, was FSenSync system, used for synchronizing different sensors, video, and audio for the scientific research and different installation purposes, Desktop server took care of synhronization over Wi-Fi with repeated time requests to Android mobile phones within 3ms window. Bluetooth was used between devices. System allows real-time interaction between audience and performers (improvising dancers) using touch screens, Moodmetric rings and other sensing and annotation means. All is controlled from the desktop, like streaminig, recording, file downloads and uploads, and visualisation of output from all sources using python (Anaconda) scripts. Sensor data is saved to csv file in mobile phone.

According to Fröger different versions of accelerometers in mobile phones and smartwatches will have different sensitivities (old 2G, new much more) and lagging which needs to be adjusted to get synchronized signals from simultaneous activities. Synchronising through Bluetooth would not be reliable enough unless same model of mobile phones would be used. Up to five Movesense devices sending accelerometer data with 52Hz sampling frequency successfully connected to mobile phone models from Xiaomi, Nexus, Samsung, LG and Lenovo. However, devices were within two meters distance from mobile phone. When at least one out of five devices additionally sent ECG data, only Xiaomi mobile phones managed all five successfully. Device test table is visible from github. (Fröger 2018)

2.6.3 KÄVELI Platform for Patient Monitoring and Diagnostics

Ruokolainen J. from Datamo Oy presented KÄVELI project in the Movesense meetup 15 May 2019, and encouraged to look at the real-time ecosystem platform for patient monitoring and diagnostics, and especially its mobile microservice architecture that removes the need to use cloud storage for sensitive health data. Below is the figure taken from JMIR Research Protocols medical publication, presenting the data-collection system KÄVELI project used.

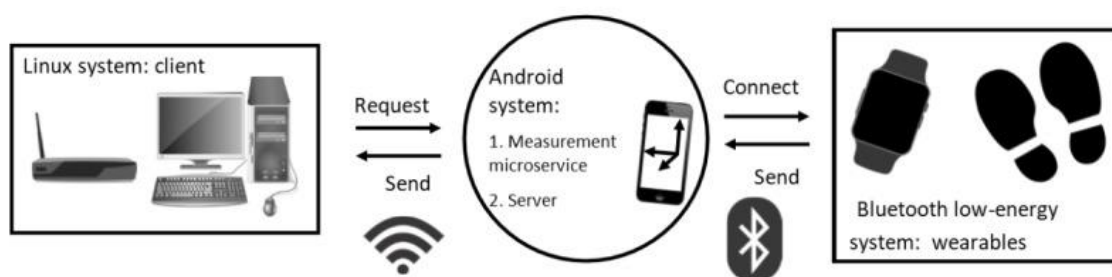


Figure 10. An overview of the structure of the data-collection system used in KÄVELI project.

The concept of back- and front-end is flipped since the smartphone acts as a server behind the simple user interface, and the personal computer acts as a client who is requesting information according to the data-collection frequency set by the personal computer user. In such mobile microservice architecture all sensors and wearable devices act as independent services with their own architecture and the use of common interface between the server and the services allows data to be collected simultaneously and asynchronously. In KÄVELI project Movesense device was worn on the wrist and measured the rest tremor or detected attenuated arm swing during walking in order to detect symptoms of under- or overmedication of patients with Parkinson disease. (Jauhiainen et al., 2019)

2.6.4 Additional Third Party Tools and Methods

One project connected Movesense device to the IBM Watson IoT Platform using gateway to pass through messages. Movesense device broadcasted messages as Bluetooth advertisement packets, embedding into them additional measurement data. On the receiving end in the cloud messages were decoded on Node-RED editor in Cloud Foundry App, and content of one message is shown in Listing 4 below, including heart rate value in line 8 and activity intensity in line 7.

```

8   27.2.2019 14.25.12
9   iot-2/type/g/id/MSGW-007/evt/msg/fmt/json : msg : Object
10  object
11  topic: "iot-2/type/g/id/MSGW-007/evt/msg/fmt/json"
12  payload: object
13
14  ACT: 600.0024851921480149    'xyz' sum intensity in 1sec window

```

```

15  HR: 62.20000076293945      heart rate (bpm)
16  MAC: 12345678901234      MAC address of the sensor
17  msgID: 74                running message ID (Uint32)
18  RSSI: 191                RSSI value (191-256= -65dB)
19
20  deviceId: "MSGW-007"
21  deviceType: "g"
22  eventType: "msg"
23  format: "json"

```

Listing 4. Bluetooth advertisement packet content with additional data embedded in line 7 and 8. (Movesense Raspbian 2019)

Script, setup and document files can be found in the root of the Movesense Community folder in the Bitbucket repository in the folder named movesense-raspi-gateway. Python, with the PyBluez module, was used for scripting and patch file creation (in setup folder), and shell script for installation file creation. Installation files and commands were then transferred through terminal to the Raspbian Raspberry Pi processor to both install and configure processor. Additionally, link to the device app sample code, named activity_broadcast_app, can be found in the Readme file, under source code folder. Furthermore, link to the IBM Developer recipes is provided for instructions to create gateway using iOS mobile phone (or Android one too using the same link).

A Mobile phone could also act as a gateway passing data from device to the cloud to be analysed in the IBM Watson IoT Platform. Linnakko has made step-by-step guide for this as an IBM developer recipe in IoT-cookbook (Linnakko 2018). Instructions for both iOS and Android mobile phones are provided. First Watson IoT service has to be created. Then mobile app code has to be downloaded, built and installed. Source code for mobile app is downloadable from branches in Movesense Mobile Library in Bitbucket repository. These so called feature names are ios-mqtt for iOS and android-mqtt-phase2 for Android mobile phone and they are made by Symbio. After mobile app is running Movesense device connection has to be made to get data collection started. Final step is to develop applications in the IBM Cloud.

2.6.5 Third Party Movesense Plugins – Unity and Xamarin

Developing mobile apps with cross-platform tools like Unity and Xamarin is possible with the help of Movesense Plugins. Movesense Sensor Plugin version 1.0 for Unity

was released in September 2018 and it is downloadable from the asset store. Figure 11 below shows two Movesense devices, chest strap with connector, and Movesense logo.



Figure 11. Movesense Sensor Plugin in Unity Asset Store.

Plugin contains different scenes, materials and scripts, as well as examples and documentation, all of which will very likely ease and speed development of an iOS or Android mobile app and its integration to the device sensor. Movesense Sensor Plugin is both published and supported by Movesense. (Unity 2018)

Plugin.Movesense is the plugin for Xamarin that allows .NET developers to build common code to both iOS, and Android targets or mobile phones owing to C# bindings around the native iOS libmds.a and Android Mdslib.aar libraries. Plugin.Movesense package includes other two packages MovesenseBindingiOS and MovesenseBindingAndroid as dependencies so only the Plugin.Movesense is added to project of Microsoft Visual Studio 2019. Earlier version is not usable since Xamarin tools in it do not support java8 features that plugin needs or the native plugin library for Android, at least. (MovesenseDotNet 2019) Plugin contains the Xamarin .NET API for Movesense and is available on NuGet. Latest plugin version 2.1.0-beta contains major changes related to native libraries. (Plugin Movesense 2019).

Xamarin Forms allows single UI implementation for all platforms while native UI implementations allow over 75% of code to be shared between targets or mobile phones (Movesense Xamarin 2018). Three app samples are available in the github, two simple ones and one with more complex structure using full Model-view-viewmodel (MVVM) leading to better separation between development of the UI presentation (view) and business logic (model) parts of app. Nevertheless function of this structurally complex sample app GraphPlotSample is as simple as to connect to device sensor and subscribe readings from accelerometer to graph plot those readings. Other sample app is BasicDemo reading device info and battery level and then triggering alert and toggling led. Third sample app is CustomServiceSample demonstrating how to connect to a custom resource exposed by an app running on the (Movesense) device. Following example gets JSON response as a string from device sample app hello_world_app:

```
var helloWorldResponse = await
Plugin.Movesense.CrossMovesense.Current.ApiCallAsync<string>(movesenseDevice,
Plugin.Movesense.Api.MdsOp.GET, "/Sample/HelloWorld");
```

And its API and Constructor are shown below. ApiCallAsync is a base class for all Mds API calls, and defined according to version 2.x which uses object instead of name of the device:

Constructor

```
MdsLibrary.Api.ApiCallAsync(IMovesenseDevice,Plugin.Movesense.Api.MdsOp,System.String,
System.String)
```

Base class for all Mds API calls

Name	Description
movesenseDevice:	IMovesenseDevice for the device
restOp:	The type of REST call to make to MdsLib
path:	The path of the MdsLib resource
body:	JSON body if any

In order to even deserialize JSON, the return type T needs to be defined and passed in (mobile) app. The parameters passed in the call to ApiCallAsync are the IMovesenseDevice object for the device, the type of operation (GET, POST, PUT, DELETE) and the path to the resource. Same technique can be used for REST end-

points not currently mapped to Movesense.NET API methods. List of API resources currently available, as well as installation instructions can be found from the Movesense .NET site. (MovesenseDotNet 2019)

The Plugin and sample apps are created by Wigley Andy, although open source package Plugin.BluetoothLe is used for device discovery implementation that is not managed by Plugin.Movesense. (BluetoothLe 2018) Next section will present partner company solutions especially those who also provide services for others.

2.6.6 Partner Services and Solutions

Currently twelve companies working with Movesense technology are offering also developing services. Some of those companies have their solutions on the market including Konect Motion, Runteq and Symbio whose solution and service will be presented next. Complete list of partners and showcases can be found on the Movesense site for further exploration.

Konect Motion is one of the latest partner companies, it is known for the speed and reaction time training tools for athletes. Konect Motion is based in U.S. but provides services globally not just sports but also consumer electronics, medical, aerospace, media and other customers. They are able to provide custom tool kits, data science solutions or complete products.

Runteq is one of the earliest partner companies, it offers its software library, Sports4IMU for licensing e.g. the basic sports features: Heart Rate Monitor (HRM), Running Speed and Cadence (RSC) and Runteq Running Biomechanics (RBS). Also In-Device App Framework is licensable as part of Runteq customer projects. Other design and development services are provided as well, including algorithm design, architecture design and custom Bluetooth GATT specification and implementation. Runteq Sports4IMU provides meaningful kinematics and kinetic parameters provided by Runteq's own Zoi running technique analysis device and Movesense-based Runteq smart running sensor. Figure 12 below contributes details to the Runteq Sports4IMU and divides it to three functional parts: Running sports bluetooth profiles, Motion analysis and In-device app development.

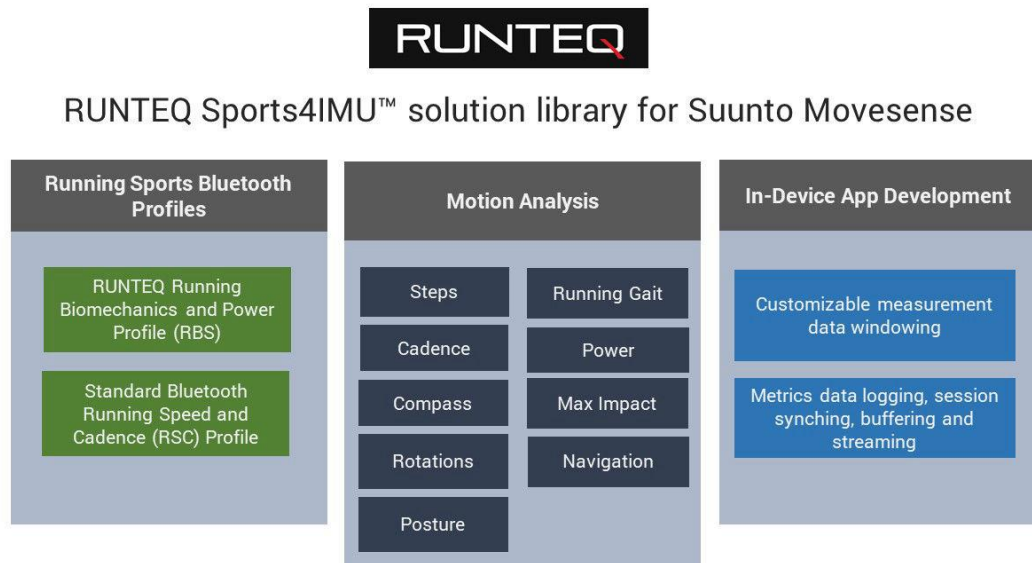


Figure 12. Runteq Sports4IMU™ solution library for Suunto Movesense consists of three functional parts: Bluetooth Profiles for Running sports (green), Motion analysis (black) and In-Device app development. (blue).

Figure 13 below shows simplified architecture of the Runteq Sports4IMU and divides it to four functional parts: Movesense Device (grey blocks in the top and bottom layers), Bluetooth LE GATT Services (blue interface on top and two green boxes at the top layer named RBS and RSC), Motion analysis (green boxes at the bottom layer named algorithm library and API) and In-device app development (green block at the top layer named Custom and dark green blocks at the middle layer).

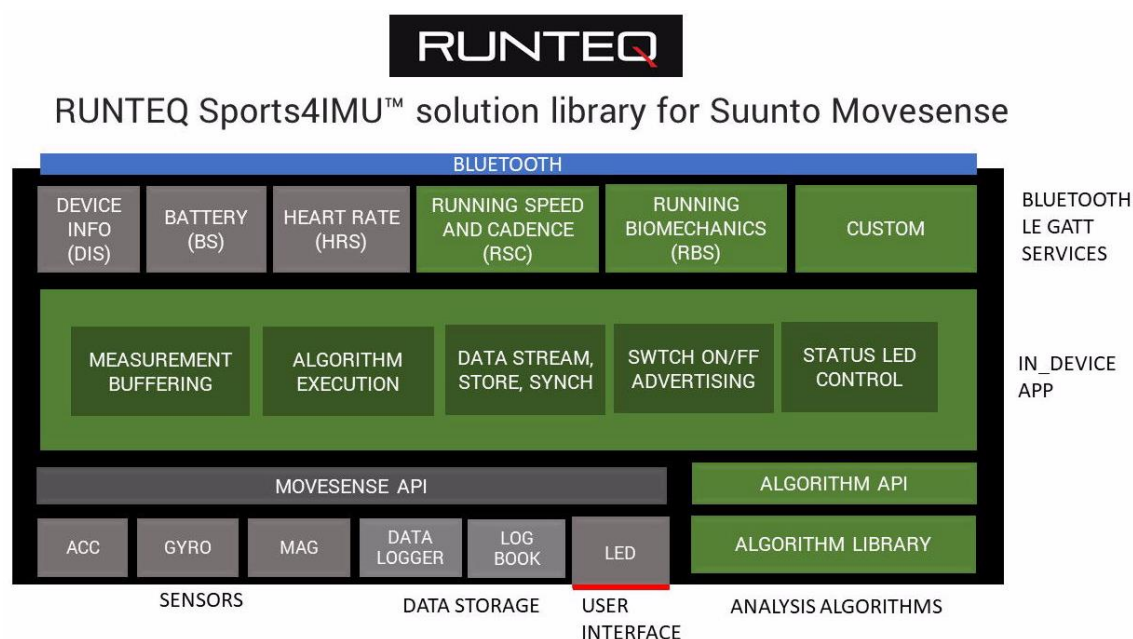


Figure 13. Runteq Sports4IMU™ solution library for Suunto Movesense simplified architecture consists of four functional parts: Movesense Device (grey blocks), Bluetooth LE GATT services (blue and two green blocks: RSC and RBS on top), Motion analysis (two green blocks at the bottom right corner: algorithm library and API), In-Device App (green block: custom on top and dark green blocks in the middle).

Chinese innovation incubator COMB+ invested in Runteq in 2018, and as a result of that the next generation wearable running sensor and AI-driven digital coaching service is planned to be launched into Chinese market in 2019. Since China has planned, in its latest national plan, to get third of its population to exercise by 2020, there is a great potential for Runteq to coach at least part of those 435 million Chinese people looking for suitable form of exercise. COMB+ has offices in Helsinki and Beijing providing both entrepreneurship services to and investing in early-stage startups in the technology and cultural industries. Already tens of other Nordic startups have participated COMB+'s Sino Track accelerator program. (Runteq 2019)

Another partner company providing development services is Symbio. In 2018 they also looked for developers for their own Movesense project. The Smart Safety concepts is targeted to hazardous workspace use to prevent accidents by identifying unsafe behaviour, unauthorized personnel, entering no-go zones, and medical issues as can be seen in Figure 14 below.



Figure 14. Symbio Smart Safety solution integrates Movesense sensor device in the work wear. Symbio logo is visible on the top of the device. (Symbio 2018)

Besides safety also productivity improvements will be included such as improvements in people and material flows. The concept is based on cloud backend connecting already existing system with Movesense platform. (Symbio 2018)

This Chapter 2 has now examined most important assets that Movesense platform offers. Thus, next Chapter 3 continues gathering additional knowledge needed for breathing application development.

3 Breathing Application Specific Knowledge

Breathing is a natural physiological movement partially unconscious, but still muscles have to work to contract and release muscles along the breathing in and out. In this study context it is both health related factor and movement measuring factor, so both of those fields are explored for applicable knowledge. However starting point in all development should be user, thus we start from principles and methods available to guide the development in order to create positive experiences for users of wearable solutions.

3.1 User Experience Principles for Developing Health and Fitness Wearables

Results from another thesis are used as a starting point of prototype development as explained in the introduction in Chapter 1. Those results were derived from studying health and fitness wearable user experiences both through literature study and through user interviews. That knowledge was then synthesized into definition of five principles that could guide the development of next wearable solutions. Especially in terms of interaction, long-term usability and 24/7 usability which are main areas requiring better solutions to enhance wellbeing of user. Those principles are shown in Figure 15 below.

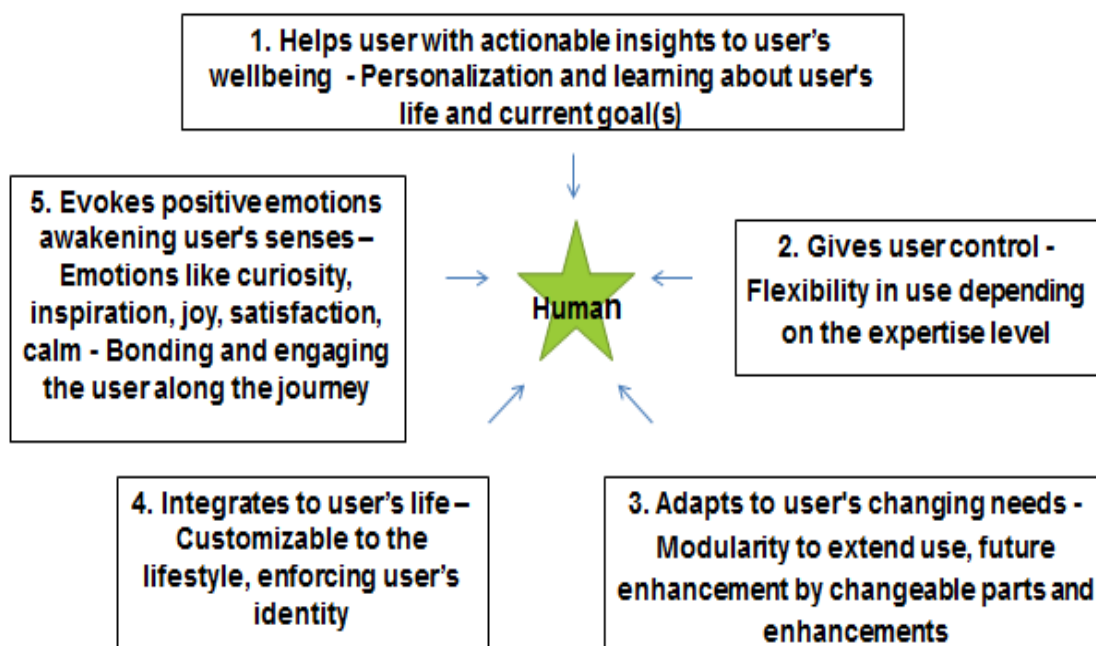


Figure 15. The final proposal for the user experience principles for developing health and fitness wearables and smartwatches. (Löfblom 2017).

Actionable insights, giving control, adapting to user's changing needs, integrating to user's life and evoking positive emotions were identified as good building blocks for future wearables. Those are needs from the human perspective that must be kept in the focus during development process. While developers focus in improving personalization, flexibility, modularity, customizability, and engagement aspects of the solution. Those ideal principles are something that development should aim to create using current technology, tools and methods. Next section will cover some of the methods provided by positive computing that explains why and how technology should serve well-being and enhance human potential.

3.2 Positive Computing

Positive computing is a term used for the design and development of technology to support psychological wellbeing and human potential. Due growing interest in social good derived from public concern about effects of digital technology, it reflects focus on humanistic values in many different disciplines. Calvo R and Peters D synthesise theory, knowledge and empirical methodologies from a variety of fields and create framework for positive computing. (2014) In this section we will repeat some parts of that knowledge potentially usable for the breathing application.

Initially, comparing the Double Diamond process diagram, used in pre-study, and shown in Chapter 1.1, and the modified version in Figure 16 below, confirms how well-being and ethical aspects are emphasized both during the process and after it. In addition to that, the modified Double diamond process is called The Responsible Design Process, renaming the Develop phase to Ideation phase and the next phase Prototypes phase instead of Deliver phase. Despite the naming differences both diagrams have initial problem that is defined after first diamond and implemented solution after second diamond. But most importantly responsible design process evaluates wellbeing and ethical impact of use of solution after the process. So, looking into different wellbeing factors will help to uncover potential factors usable for breathing application.

The Responsible Design Process

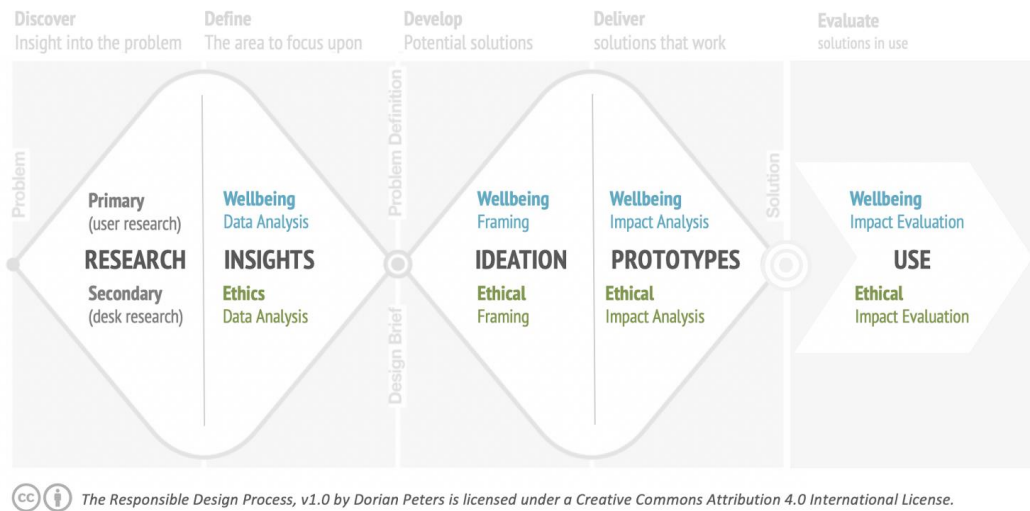


Figure 16. The modified Double diamond design process called The Responsible Design Process. (Peters n.d)

In summary, wellbeing factors for positive computing can be divided to three main parts self (intrapersonal), social (interpersonal), and transcendent (extra-personal). Social (interpersonal) factors include interaction with others, whereas transcendent (extra-personal) factors include personally not-known persons. Social factors are gratitude and empathy, whereas compassion and altruism belong to transcendent factors. Furthermore, factors belonging to self, are positive emotions, motivation and engagement, self-awareness, mindfulness, and resilience. All these wellbeing factors have own theories, strategies and methods in framework of positive computing. (Calvo & Peters 2014: 85)

The consistency of movement, slow or fast depending on person, was the key indicator of flow in the study conducted by Yano, Ara and Csikszentmihalyi. Results were based on comparison between activity diary entries and (accelerometer) sensor data recordings. (Yano et al. 2012; Calvo et al. 2014: 150). Result shows individual variation that could be used in personalization of breathing application. Additionally, allowing for non-absolute categories in presentation of analysis of personal data to support inferences about states of mind or behaviour, induces adaptiveness or creativity if done with conditional conception (Langer & Piper 1987 in Calvo & Peters 2014).

In order to avoid distraction, autonomy and minimalism could be used in design, in other words, honouring user agency as to when, where, and how help occurs, as well as subtlety, not to startle or distract, but improve self-regulation of attention. The Breath-Walk Aware System and the Slow Floor were given as examples of technologies supporting mindfulness in daily activities. The Sonic Cradle is another embodied experience where detected breathing patterns shape the ambient sound around cradle. Experience caused similar effects than mindfulness meditation, like clarity of mind and loss of intention. The HAPIfork on the other hand promotes slow eating and records time and speed of eating, correlates data with sleep, meal, and relaxation times, and even connects to desktop app for coaching advice and stats from collected data and self-reports. Online mindfulness training can significantly decrease perceived stress according to study of impact of such training programs (Krusche, Cyhlarova, King, and Williams, 2012 in Calvo & Peters 2014).

Dweck (2006 in Calvo & Peters 2014) has also concept of growth mindset and fixed mindset, meaning that those with fixed mindset limit themselves by thinking that they can not do something and thus avoid obstacles whereas growth mindset believes their abilities can be enhanced and grow with time. So this has implications in goal setting in popular fitness applications. How to create both sense of control for people with fixed mindset, and feeling of engagement for people with growth mindset.

Kauer et.al. (2012 in Calvo & Peters 2014) showed evidence that mobile phone self-reports can help reduce the brooding component of rumination (which is the component with strongest correlation to depression). This could be used as a way to increase user's emotional self-awareness in breathing application, which is needed in habit changing practice. In order to support self-awareness and reflection, designing for self-compassion, gratitude, and mindfulness helps prevent comparison, self-criticism, narcissism, and entitlement. In addition to that feedback for making better decisions will be more effective than providing specific instruction, since users often know more about their context than developers. (Calvo & Peters 2014: 172)

Peak – end rule is another effective way to design, relying on what is remembered afterwards of painful experience. Kahnemann found out that only average of peak pain and end pain during painful operation was remembered afterwards, meaning that the length of painful experience is less significant. (2011 in book of Calvo & Peters 2014)

Similarly Norman (2009 in Calvo & Peters 2014) argues that seeking a good remembered experience is more important than seeking a perfect moment-by-moment user experience, since emotions fade faster than cognitions. Such results might give some ideas of moment-to-moment experiences versus remembered experiences and need to be reflected in the design.

Positive emotion increases creativity and problem solving. Norman defines three levels for emotional reactions to design the visceral, behavioural, and reflective experience. In short summary those levels are for visual, interaction and value related emotional reactions to experiences. (2005 in Calvo & Peters 2014)

Gilbert and Choden (2013 in Calvo & Peters 2014) describe two physiological systems linked to two groups of positive emotions; the excitement and drive system and the affiliative and soothing system. Too much drive leads to over self-focus and indifference to suffering of others, whereas kindness and feeling connected to people will help balance the sympathetic and parasympathetic nervous system. Even so that compassion and self-compassion are sources of resilience and mental flourishing.

In addition to that, synchronous movement can enhance cooperative ability, compassion, and even altruism according to research on dance, rituals, and movement done by Behrends et al (Behrends, Müller, & Dziobek 2012 in Calvo & Peters 2014). Moreover, according to appraisal theories (Ortony, Clore, & Collins 1988 in Calvo & Peters 2014) likelihood of compassion arising can be increased by following three factors, relevance or similarity, goal congruence or fairness, and empowerment or ability to cope. According to Emma Seppälä (2013 in Calvo & Peters 2014) practice of compassion will be as important for health as physical exercise and a healthful diet, it is taught and applied widely, and empirically validated techniques for cultivating compassion are widely accessible. There are physiological signals and facial expressions unique to compassion, but affective computing has not yet attempted to detect compassion (Calvo, D'Mello, Gratch, & Kappas 2014 in Calvo & Peters 2014). Evidence exist that pro-social games develop altruism and other prosocial behaviours, and that embodied experience of helping facilitates the transfer of this behaviour to the real world (Rosenberg, Baughman, & Bailenson 2013 in Calvo & Peters 2014).

Furthermore Fredrickson (2001 in Calvo & Peters 2014) states that there are two ways to experience more positive emotions; to decrease negative or to increase positive

emotions. In addition, she lists ten positive emotions: joy, gratitude, serenity, interest, hope, pride, amusement, inspiration, awe, and love, which she considers significant to wellbeing and phenomenologically distinct.

To conclude, above findings can be used pragmatically either to redesign, actively integrate or dedicatedly promote wellbeing in technology design. Next section will define breathing application by asking why breathing exercises are important and how breathing could be measured. For example both self-awareness and mindfulness are increased by breathing exercises used in cognitive behavioural therapy.

3.3 Definition of Breathing Application

Maximum wellbeing impact of breathing exercises will provide incentive for breathing application creation, whereas measurement practice will constitute limiting factors for breathing application. Sensors provided for tracking breathing movement especially accelerometer will further fine-tune the investigation of best practice in measuring breathing in following sections. These definitions for breathing application will be used in specifying requirements for prototypes.

3.3.1 Respiratory System

The autonomous nervous system (ANS) consists of sympathetic and parasympathetic parts, in other words the accelerator and the brake pedal parts, where the sympathetic nervous system controls the flight or fight response, and parasympathetic nervous system calms us down. Similarly the inner organs are affected so that bronchi, the main passageway to lungs, dilates and heartbeat accelerates according to the sympathetic nervous system whereas bronchi constricts and heartbeat slows according to the parasympathetic nervous system. In other words those systems work in opposition to each other in order to maintain homeostasis or the dynamic equilibrium within the body. (StudyLib The Nervous System)

Homeostatic control of our respiratory system happens for example during exercise when high concentration of carbon dioxide (CO₂) is detected by carotid arteries in the neck and aorta (artery) causing chain of nerve impulses from respiration control centre (medulla) to diaphragm and intercostal (in between ribs) muscles in the chest to in-

crease breathing rate and depth to return carbon dioxide (CO_2) level to normal. (Smith, 2015)

The normal rate of inspiration and expiration, the respiratory rate (RR), is about 16 times a minute in an adult. Cells transfer oxygen and glycogen to energy, water and carbon dioxide. Air has 21% of oxygen that is breathed in and 17% of it is breathed out. Inspiration is an active process caused by muscular contraction, mainly of the diaphragm and intercostal muscles. The diaphragm is a sheet of skeletal muscle curving below lungs. During inspiration the diaphragm contracts and flattens out to allow expansion of lungs and during expiration it recoils back to its curved position. Expiration is typically a passive process mainly caused by elastic recoil of the diaphragm and relaxation of the intercostal muscles between the ribs. For really deep inspiratory breaths the external intercostal muscles contract more to pull on the rib cage and sternum more, to raise and draw it out further. Other muscles can contribute in that too. (StudyLib The Respiratory System) Position of diaphragm within thorax can be seen in Figure 17 below.



Figure 17. Position of diaphragm in the thorax at the bottom of the rib cage. (Beyer 2001)

Diaphragm is positioned at the bottom of rib cage in the thorax, and its connection point to torso is lower at the back than in front. Next chapter grounds out reasons for exercising diaphragm.

3.3.2 Importance of Breathing Exercises

Most important benefit of diaphragmatic (abdominal) breathing is reduction of stress, but also deep breaths are more efficient allowing full exchange of oxygen and carbon dioxide in body. Breathing exercises like Diaphragmatic and Rib-stretch breathing improve deep breathing. Taking deep breaths will voluntarily regulate autonomic nervous system (ANS), which then lowers heart rate, regulates blood pressure and helps relax, all of which help decrease stress hormone cortisol in body. However, such exercises may increase anxiety of those already suffering of anxiety like those diagnosed with generalized anxiety disorder (GAD). (Healthline 2017)

Other breathing types are shallow, tidal, and forced breathing. Shallow breathing or chest breathing draws minimal breath into the lungs and uses the intercostal (in between ribcage) muscles instead of diaphragmatic muscle. Tidal breathing is normal, resting breathing (12 – 20 breaths/minute for adult), whereas forced breathing is deep inhalation and fast exhalation typically used to measure maximum capacity of lungs. On the other hand in quiet breathing exhalation is passive and non-forced. Paced breathing is low, deep, diaphragmatic breathing. Box breathing is breathing in, holding, breathing out, and holding in equal amount of seconds in each step and it increases alertness. Different breathing patterns exist like 4-7-8 for relaxing and calming effect. Breathing through mouth or nose or alternately left or right nostril is also part of breathing exercises. (Iceman) Wim Hof Method calls short burst breaths in from nose and out from mouth, like blowing air into balloon, as Power breaths, which increases heart rate and opens senses. Whereas, holding breath for 10 minutes after full inhale is called Recovery breath. Sequence of breathing exercises has retention of breath after full relaxed inhale and exhale in between power and recovery breaths. Doing such sequence of breathing exercises doubles oxygen consumption in body, visible 45 minutes afterwards prior it starts slowly decaying. (Wim Hof Method 2018)

Holding breath accumulates the carbon dioxide in cells, blood and lung, and trigger impulses from the respiratory centre part of brain. The signals include strong, painful, and involuntary contractions or spasms of the diaphragm and the muscles in between ribs. The point at which carbon dioxide causes pain is called critical line and it is individual. Hyperventilation can push that line back but it might be dangerous to overdrive brain. Hyperventilation is fast breathing in and out. (Magana 2012)

Slow breathing (<10 breaths/minute) used in meditation is affecting brain areas related to increasing positive emotions and reducing negative ones, as can be derived from following quotation:

“The main effects of slow breathing techniques cover autonomic and central nervous systems activities as well as the psychological status. Slow breathing techniques promote autonomic changes increasing Heart Rate Variability and Respiratory Sinus Arrhythmia paralleled by Central Nervous System (CNS) activity modifications. EEG studies show an increase in alpha and a decrease in theta power. Anatomically, the only available fMRI study highlights increased activity in cortical (e.g., prefrontal, motor, and parietal cortices) and subcortical (e.g., pons, thalamus, sub-parabrachial nucleus, periaqueductal gray, and hypothalamus) structures. Psychological/behavioral outputs related to the abovementioned changes are increased comfort, relaxation, pleasantness, vigor and alertness, and reduced symptoms of arousal, anxiety, depression, anger, and confusion” (Zaccaro et al. 2018)

However, both attentional-based and breath-based techniques used in meditation can lead to similar states, and furthermore nostril-based respiration could be one of the key factors behind positive psychophysiological effects. Thus, systematic review made by Zaccaro et al. created checklist of nine items for future research to promote a more standardized research on breathing techniques. (expanded from taxonomy for meditation of Nash and Newberg (2013) in Zaccaro et al. 2018):

- I Specifying whether breath is consciously attended or not
- II Specify if other techniques are associated with breathing (e.g., “feeling the breath in the body,” sounds with mouth, breath-related mantras, breath-related imagery, etc.)
- III Specify the mean breathing frequency and, if present, any significant breathing frequency variations
- IV Specify whether during respiration the air passes through the mouth or through the nostrils (both, left, right, alternate), or through both mouth and nostrils
- V Specify the presence and the duration of inspiration (if any) and expiration pauses (if any)
- VI Specify the Inspiration/Expiration ratio
- VII Specify whether the breath is thoracic or abdominal
- VIII Specify (if applicable) what type of metronome is used
- IX Specify (if applicable) the air pressure during the inspiratory phases.”

Nevertheless exercising the diaphragm has many practical applications and it is used by singers (Bohemian vocal studio 2018) and deep divers (Campbell 2016), to whom it

is essential skill to manage diaphragmatic breathing. However, good posture and breathing technique are essential to all humans, so in order to activate (GuerrillaZen Fitness 2015) or release (Abelson 2015) diaphragm, exercising at home is one way to do it. In order to get started, short videos showing how and explaining why can be reached through our reference links above. While doing exercises, concentrate breathing two thirds of air in abdomen and one third in chest, according to GuerrillaZen Fitness. Next section studies how to measure breathing.

3.3.3 Measuring Breathing

Breathing is typically measured from air flow with spirometer or from amount of oxygen in the blood with pulse oximeter. However, since breathing is movement, it can also be measured from thorax or abdomen with accelerometer. In the diaphragmatic deep breathing, movement of the thoracic diaphragm should be 3-5 cm, but can be even 7-8 cm on well-conditioned person. Volume of abdomen and intercostal respiration could be also measured with Respiratory Inductance Plethysmography (RIP) –belt, using induction.

Breathing measurement with accelerometer could be equivalent of measuring sedentary daily activities. In such measurement accelerometer is typically attached to waist. And signal is formed from movement of the body, gravitational acceleration, external vibration and acceleration due to loose attachment of the accelerometer. For intensity of physical activity, activity counts are used, and digital integration is mostly used to calculate those. (Tikkanen 2014) Test done for Fibion showed that position of the accelerometer matter since thigh-worn one was similar to Actigraph for longer tests but pocket-worn not. Large random measuring error was problematic for shorter tests for both accelerometers. (Peer 2018)

According to Crouter et al. (2006 in Tikkanen 2014) general Actigraph devices tend to overestimate walking and sedentary activities and underestimate most other activities. Moreover equations might not work well for classifying other than moderate activity. During free-living activities, single regressions over- and underestimate energy expenditure (EE). Static work or strength training is not detected. Moreover removal of accelerometer could be interpreted as non-activity, unless HR is measured as well. So we will also look HR in this section.

EE and oxygen uptake tend to be linearly related within an individual throughout most of the aerobic work range, so EE estimation of HR is based on that, However during light activity or inactivity, there is almost no slope between HR and EE. (Rennie et al. 2001 in Tikkanen 2014) And individual variability exists, and additional factors like food intake, emotions, body position, ambient temperature, muscle groups used and type of muscle activity also affect HR, not just EE. (Livingstone 1997 in Tikkanen 2014)

For filtering Tikkanen used 0.5-11Hz band-pass filter with finite impulse response to decrease effects of gravity and high frequency artefacts. (Van Someren et al. 1996 in Tikkanen 2014) Signal was also rectified and moving average with 10s window was used. Activity counts were calculated for each axis separately. Sum vector was calculated with equation $xyz = (x^2 + y^2 + z^2)^{1/2}$. Activity counts were calculated as area under the curve (integration) from 60s time periods.

Pakkala used 13Hz sampling frequency and Infinite Impulse Response (IIR) bandpass filter with corner or cut-off frequencies 0.1-0.4Hz with pole and zero value 4. IIR filter is easier to implement and filters selectively frequencies, whereas Finite Impulse Response (FIR) filter is linear and affects equally to all signal frequencies. Butterworth filter has flat stop- and passbands and not very steep transition-band. Additionally interpolation factor was 265 and decimation factor 68, so. $12.5 * 265 / 68 = 48,7$. (13Hz is currently 12,5Hz in Movesense specifications, whereas actual value is probably somewhere in between according to Pakkala.). Respiratory rate (RR) was calculated using Fourier transformation with 30 seconds periods with window of two minutes moving average, which was less prone to error than looking for peaks and threshold of respiration signal, and also more independent of position. Euclidean norms or sum vector of axis were calculated. SNR-ratio of z-axis of accelerometer was lowest. (2018) Since Movesense device was used although for paediatric sleep studies and also in order to compare Movesense with other sensors like Emfit, still it gives a good starting point for respiration signal handling in breathing application for adults.

The heart is a pump that beats approximately 72 times per minute and sends blood to the lungs to be re-oxygenated. Amount of blood pumped out of the heart each minute is called cardiac output. It is calculated by multiplying heart rate by stroke volume which is the amount of blood pumped out of the heart in each beat or contraction. Trained athlete has higher cardiac output than non-athlete and larger stroke volume. The resting heart rate for the average person is between 70 and 90 beats per minute (bpm) but

for endurance athletes it may be less than 50 bpm. (StudyLib The Respiratory System) Endurance improves and fat burns when training happens at 50% to 60% from the maximum heart rate (MHR). Aerobic training happens at 60% to 70% from MHR and anaerobic training at 10% higher rate. MHR start out approximately at 220 beats per minute and falls by one beat each year, or in more exactly falls by half a beat each year after 30 years. The oxygen consumption (VO_2) increases during the physical activity and even after it so called excess post-exercise oxygen consumption (EPOC) continues.

Measuring heart rate variability (HRV) provides a great holistic view on the health. Relation between HRV, HR and breathing is explained in Figure 18 below.

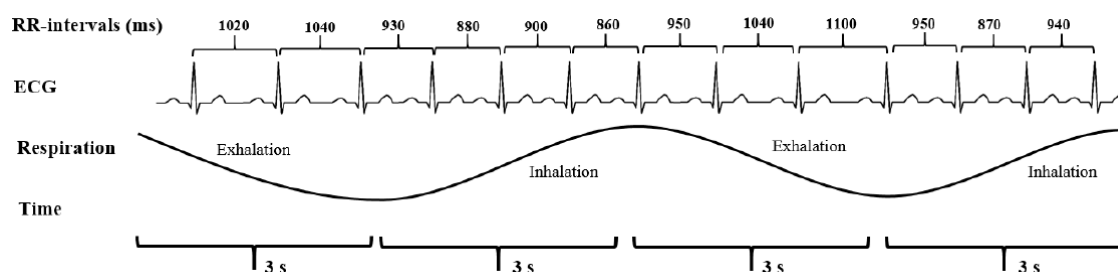


Figure 18. Electrocardiogram (ECG) exhibiting respiratory sinus arrhythmia (RSA). Heart rate increases and thus the time between successive RR-intervals get shorter during inhalation (inspiration) and longer during exhalation (expiration). This fluctuation in the time between the successive RR-intervals is called HRV. (Heartmath n.d).

So deep breathing increases the HRV but increase in HRV does not automatically increase HR or vice versa. However, the natural relationship between HR and amount of HRV exists, and it is called cycle length dependence, meaning that less variability occur at higher HRs, while at lower HRs there is more time between heartbeats, so variability naturally increases, but if variability does not increase that is an indication of heart problems.

Measurement of HRV can be done in time domain, that is simplest to calculate, or frequency domain, which provides means to adequately quantify autonomic dynamics or determine the rhythmic activity generated by the different physiological control systems (Heartmath n.d). Depending on calculated statistical parameter, either 24 hour measurement or 5 minute window is typically used in time domain to quantify the amount of variance in the interbeat interval (IBI). Standard deviation (SDNN) where all cyclic

components are measured uses generally a 24 hour measurement, and root mean square of successive differences (rMSSD) which is a reflection of Vagal Tone is generally calculated on 5 minute window, and the mean of the standard deviations (SDNN Index) uses both so that a 24 hour measurement is divided into 288 five-minute segments for calculation of average of these 288 values (Heartmath; Collier 2012).

HRV-guided training has been popular since study of Kiviniemi et al. in 2007 (Flatt 2017). In accordance with that, HRV is most useful for daily measurements in individual sports characterised by high metabolic component and high training volume, but not sports characterized by neuromuscular and cognitive function like strength training. In addition to that, in team sports daily measurements might be too much trouble. Figure 19 below describes decision chart for HR measurement type, depending on the measurement frequency, that is most practical for defining training loads to improve performance.

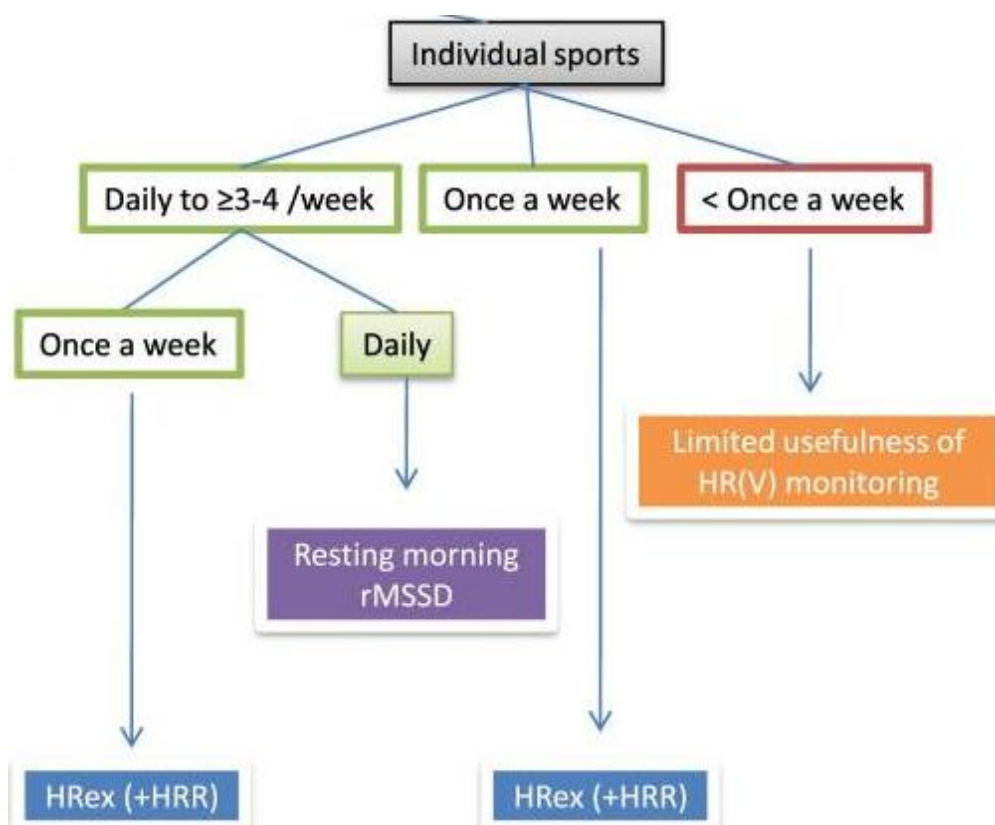


Figure 19. Decision chart for the selection of heart rate (HR) measures based on sport participation and implementation possibilities. HRV, heart rate variability; rMSSD, logarithm of the square root of the mean of the sum of the squares of differences between adjacent normal R-R intervals measured after exercise; HRex, submaximal exercise heart rate; HRR, heart rate recovery. (Buchheit 2014)

According to Buchheit (2014), HR not measured daily but rather once a week could use submaximal exercise HR (HR_{ex}), which is measured from last minute of five minute exercise done in submaximal HR zone, and possibly together with HR recovery measured after exercise. In order to assess meaningful changes in HRV daily measurements are needed, taken from resting morning HRV after previous day exercise and using $\ln \text{rMSSD}$, logarithm of the square root of the mean of the sum of the squares of differences between adjacent normal R-R intervals (beat-to-beat intervals), for HRV calculation.

While the mean HRV, taken from weekly values of $\ln \text{rMSSD}$, reflects average vagal or parasympathetic activity, the Coefficient of Variation (CV) represents perturbations occurring therein, such as the stress (decreases in HRV) and recovery (return to at or above baseline) response to training and lifestyle events. CV is calculated as the standard deviation (of the HRV data) divided by the mean HRV value and expressed as a percentage. Both mean HRV and CV are needed to evaluate direction of trend (increasing or decreasing) and amount of fluctuation within. Trends differ based on athlete, training phase, fitness level, sport and so on, so interpretation of unique HRV is needed, often including non-training factors, like wellness questionnaires, as well. Following quotation clarifies complexity: (Flatt 2017)

“Few studies have reported HRV CV values (regarding $\ln \text{rMSSD}$) but the following will summarize most of what’s available. Elite male triathletes during baseline training have been reported to have a supine CV of $6.7 \pm 2.9\%$ while recreational endurance athletes had CV values of $10.1 \pm 3.4\%$ ²¹ and 12.7% .²² Collegiate women’s soccer (NAIA) players have demonstrated average CV values of $6.7 \pm 3.5\%$.¹⁰ from supine measures during the offseason while NCAA D-1 players have shown CV values of $7.7 \pm 3.3\%$ from seated measures during preseason. The CV from seated HRV measures for elite male rugby players during a short training camp was 7.65% .²³ Professional indoor male soccer players had seated CV values of around 10% during the early preseason which improved to an average of about 7% in the late preseason.²⁴” (Flatt 2017)

So, the position HRV is measured in, supine (lower) or standing (higher), impacts the CV, as well. Furthermore, individual CV values range from 1% to more than 20%. In general, 2-6% for fit athletes and 7-12% for less fit, for collegiate athletes in soccer, swimming an American Football, or among variety of professional athletes 6-8% could be used as common average for CV values. (Flatt 2017) So, applying similar practical approach for both HR and breathing measurement should be wise. Implementation of breathing application prototypes will be described in next chapter.

4 Implementation of Breathing Application Prototype

According to objective set of prototypes was to be created and initial requirement will be given based on definition of breathing application as well as knowledge from literature and Movesense platform. Next section will define prototypes for breathing application to be implemented.

4.1 Road Map for Breathing Application Prototypes

Initial prototype needs to handle data from the sensor. Since Pakkala (2018) used IIR Bandpass filter with cut-off frequencies 0,1-0,4Hz with pole and zero value 4. An in addition to that, interpolation factor was 265 and decimation factor 68, so. $12,5 * 265 / 68 = 48,7$, where 12,5 is sampling frequency. Respiratory rate (RR) was calculated using Fourier transformation with 30 seconds periods with window of two minutes moving average. However for first prototype threshold and peak detection will be enough. Pakkala used Matlab so other implementation possibilities need to be looked at.

Breathing measurement will be done by standing or sitting but without movement. Breathing pattern should be symmetrical and deep, which will help in seeing it better visualised. Symmetry of breathing in and out should be maintained. Alternatively Box breathing with 4-4-4-4 could be used. Device will be attached to the chest strap worn on abdomen just above navel as seen in Figure 20 below, and so that Movesense logo is readable. Thus z-axis will be of interest, more than other axis.



Figure 20. Position of Movesense device above navel just below rib cage during breathing measurements. Movesense device is connected to chest strap typically used for heart rate measurements from chest.

Second prototype will look at Unity and Movesense sensor plugin to find out more about the visual implementation. Minimalist and non-disruptive mobile app will be required according to the positive computing. So, both interaction and visualisation will be targets.

Third prototype will look at connectivity and architecture and try to add gyroscope together with accelerometer to measure breathing, and see what custom GATT service offers. With gyroscope more movements could be added, for example lying down while doing breathing exercises and turning to one side and then other could be enough.

Several devices could be used if needed for separate device apps.

Fourth prototype will look at data collection from different sources to get context to breathing measurements, since measurements alone are not enough to recognize causes of stress. Building insight through feedback from user and triggering self-reflection on user are targets of this prototype. As well as defining what are steps to combine data from different sources.

Fifth prototype will look HR data use together with accelerometer and potentially gyroscope data. Consider usability, reliability and power saving options for using different sets of sensors.

Sixth prototype will create a system, which consist of options like IBM Watson, different GATT clients, KVM virtualisation, in the other words other than mobile phones too. Uncover what are pros and cons of each system. Several of earlier prototypes could be used in order to build complete system (if there are enough sensors to do that).

4.2 Setup for First Prototype – Handling Breathing Data

First prototype was built with Movesense tools except filtering part that taken care in Octave open source version of Matlab, since filter code was already made in other thesis (Pakkala 2018). Latest apps in mobile phone and device are used. Device is worn on waist with chest strap, so that device is above navel, and Movesense logo is readable and not upside down). Abdomen location is used to get bigger breathing movement (3-8 cm) in z-axis. Breath relaxed fashion, stand or sit.

Install apps: Showcase app version 1.9.6 (latest version) was downloaded from Bitbucket (Downloads) and installed. After which nrF Toolbox took care of loading and OTA upgrade to device hr_wakeup_app version 1.9.0 (latest version) was used, attached to chest belt. Box breathing with 4-4-4-4 was used, with equal 4 seconds in – hold – out – hold breath.

Connection process: First enable Bluetooth and Location information. Then open Mobile app, and make sure device is woken up by touching both metal studs and if led blinks it has woken-up. Then look device from the list on the mobile phone to see if it is shown on the list. If yes, select device from the list. (Follow instructions on the screen in case something has changed since last use, and wait, since connection may take 10-30s).

Subscribe to service: Look at the left side of Figure 21 which shows menu structure. After device has connected, select service Linear Acceleration to subscribe to, and then push the slide if default (lowest) sample frequency is adequate for sensor data, otherwise change the frequency to suitable by typing it in under subscription button. 12,5Hz is lowest frequency, so just swipe subscription slide to right and wait for measurements to come to mobile app.

Visualise measurements: Look at the right side of Figure 21 which shows text values on top z-value last in green and graph with z-axis shown green in the middle with minimum peak. Y-axis is at the top and x-axis in the bottom of graph. During breathing in curve starts to go down, hold it flattens out, when breathing out curve starts to go up. (it kind of goes opposite than abdomen movement).

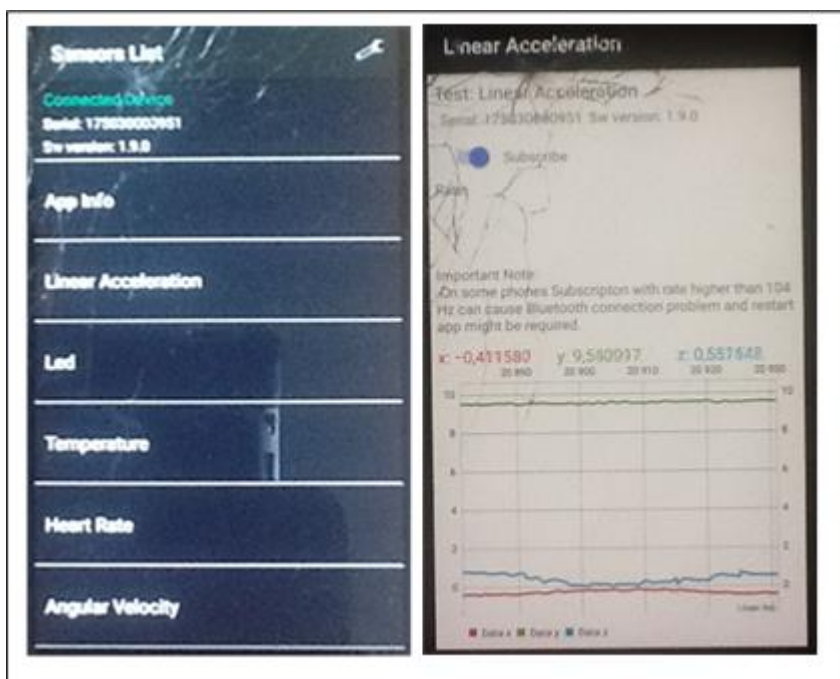


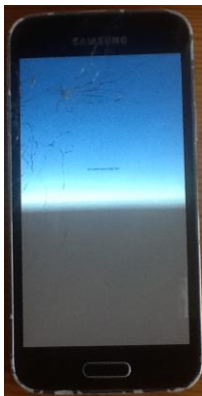
Figure 21. Views from Showcase_app: First view Menu structure or available services. Second view Accelerometer service data of breathing visible in Z-axis shown in blue (middle curve).

Handle measurement file: Once enough measurements is saved, close the mobile app, and device will go to sleep as well, when connection goes down. Take the file from Movesense folder in the mobile phone and open it in Excel (or other tool). File is of csv type (comma separated value). See file structure for PC Simulator running in Chapter.2.5.2. File can be opened in Octave, Matlab, or Jupyter python editor or any other way that is needed for filtering purposes. Matlab code from Pakkala is available from Theseus.fi in appendix of the thesis. Same tool that takes care of filtering can probably visualize the results as well. PC Simulator can also read filtered csv file and show it as long as the file headers are modified as instructed in Chapter 2.5.2. For future purposes C/C++ filter can be derived from open source and ported to device code so that instead of raw data, less and more directly usable data would be sent OTA. In next section second prototype setup is described.

4.3 Setup for Second Prototype – Unity Plugin

Second prototype was built by using Movesense Sensor Plugin for Unity in Visual Studio 2015 and with MDS library in order to build Android MoveCare mobile app that could be used with Google cardboard glasses by placing phone into the glasses. Fig-

ure 22 below shows view during recording of measurements to the mobile phone in dat file format. Measurements are displayed in the middle of the screen but very small font is used that is more unobtrusive for user.



Several sensors were intended to be used for knees, elbows and torso, however due some connection issues two sensors were used most of the time, additional sensors were to detect gait related movement. Mokka version 0.6.2 was used for sensors.dat file visualisation (offline), but prior that dat file format was converted to trc file format. Part of python code from convert.py is presented in Listing 5 below. Note that the other sensor is commented out for clarity of example.

Figure 22. MoveCare app view for Accelerometer data recording.

```
#!/usr/bin/python

dt = 0.05
numMarkers = 1 #2
file_in = "sensors.dat"
file_out = "sensors.trc"

markers_prev_pos = {
    "950" : [ 0.0, 1.0, 0.0 ]#,
    # "951" : [ 0.0, 1.0, 0.0 ]
}
markers_prev_vel = {
    "950" : [ 0.0, 0.0, 0.0 ]#,
    # "951" : [ 0.0, 0.0, 0.0 ],
}
markers_prev_timestamp = {
    "950" : 0#,
    # "951" : 0
}
...
# parse
# write
...
fo.write("DataRate          CameraRate  NumFrames  NumMarkers  Units          Orig-
DataRate    OrigDataStartFrame          OrigNumFrames\n")
fo.write("%.2ft%.2ft%d\t%d\t%s\t%.2ft%d\t%d\n" % (20.0, 120.0, cnt_out_lines, numMarkers,
"m", 120.0, 1, cnt_out_lines))
fo.write("Frame")
...

```

Listing 5. File Convert.py for converting Accelerometer data from dat format to trc format.

So after conversion, next step was to transfer collected data to other visualization software called Mokka. Mokka is an open source and cross-platform software to easily analyse biomechanical data in C3D format and many other formats like trc (Track Row Column) from Motion Analysis Corporation. Mokka visualises in 3D and 2D markers' trajectories, force platforms, segments, but also joint angles, forces, moments, as well as analog signals like EMGs. It makes exploration of data from different research projects and different hardware easy. (Mokka)

Test procedure was created to include both gait control and breathing into the same training. Training was initially targeted to child user thus emphasizing visual guidance through the test. Another purpose was to reduce stress level of child in hospital by game as attention catcher and concentration facilitator. Offline Mokka part was never solved to be online over Wi-Fi as intended. Test.trc file was used in testing the tool and file structure used by Mokka timeline is visible in Listing 6 below

```
Frame number,5
First frame,1
Point frequency,120
Analog frequency,120
```

```
Time,Marke,,
s,m,m,m
,X,Y,Z
0,0,0,0
0.00833333,0,0,0
0.0166667,0,10,1
0.025,10,10,1
0.0333333,0,0,0
```

Listing 6. Test.trc file used when experimenting with Mokka and then exported as text.

Unity offers also possibilities for gaming features and 3D visualisation to be used for breathing app, however learning those skills in more advanced level was not prioritised for this study project. Mastering all tools and all features would take more time than provided for it for thesis writing purposes. Instead, Bluetooth advertisement and GATT service was looked at in next section.

4.4 Setup for Third Prototype – GATT Client with Gyroscope

The third prototype was done for two reasons, first to use Bluetooth advertisement and second to get additional gyroscope data. Modification was done into the Custom GATT Service Client quickly so that Interval Characteristics handle was commented out and instead Measurement Characteristics handle was used as shown in Listing 7 below.

```
CustomGATTSvcClient::CustomGATTSvcClient()
: ResourceClient(WBDEBUG_NAME(__FUNCTION__), WB_EXEC_CTX_APPLICATION),
  LaunchableModule(LAUNCHABLE_NAME, WB_EXEC_CTX_APPLICATION),
  mMeasIntervalSecs(DEFAULT_MEASUREMENT_INTERVAL_SECS),
  mTemperatureSvcHandle(0),
  mMeasCharHandle(0),
//  mIntervalCharHandle(0),
  mIntervalCharResource(whiteboard::ID_INVALID_RESOURCE),
  mMeasCharResource(whiteboard::ID_INVALID_RESOURCE),
  mMeasurementTimer(whiteboard::ID_INVALID_TIMER)
{
}
```

Listing 7. Modified Custom GATT Service Client to handle extra gyroscope measurement.

Furthermore, service command INDICATE was changed to NOTIFY as shown in Listing 8 below. Another change had to be done for the actual handling of characteristics after onNotify API command is handled. In this version Interval characteristic handle is skipped and thus not shown here but else if code is processed instead where Interval times related code is commented out and Gyroscope data is asynchronously subscribed or unsubscribed depending on Notification flag's state.

```
// Define the CMD characteristics
WB_RES::GattProperty measCharProp = WB_RES::GattProperty::NOTIFY;
....
case WB_RES::LOCAL::COMM_BLE_GATTSVC_SVCHANDLE_CHARHANDLE::LID:
{

WB_RES::LOCAL::COMM_BLE_GATTSVC_SVCHANDLE_CHARHANDLE::SUBSCRIBE::ParameterListRef parameterRef(rParameters);
    if (parameterRef.getCharHandle() == mIntervalCharHandle)
    {
...
    }
    else if (parameterRef.getCharHandle() == mMeasCharHandle)
    {
        const WB_RES::Characteristic &charValue = value.convertTo<const WB_RES::Characteristic &>();
        bool bNotificationsEnabled = charValue.notifications.hasValue() ? charValue.notifications.getValue() : false;
        DEBUGLOG("onNotify: mMeasCharHandle. bNotificationsEnabled: %d", bNotificationsEnabled);
        // Start or stop the timer
        if (mMeasurementTimer != whiteboard::ID_INVALID_TIMER)
```

```

//      {
//          stopTimer(mMeasurementTimer);
//          mMeasurementTimer = whiteboard::ID_INVALID_TIMER;
//      }
//      if (bNotificationsEnabled) {
//          asyncSubscribe(WB_RES::LOCAL::
MEAS_GYRO_SAMPLERATE(),AsyncRequestOptions::Empty,52) ;
//          } else {
//          asyncUnsubscribe(WB_RES::LOCAL::
MEAS_GYRO_SAMPLERATE(),AsyncRequestOptions::Empty,52) ;
//          }
//          mMeasurementTimer = startTimer(mMeasIntervalSecs*1000, true);
//      }
//      }
//      break;
//      }

```

Listing 8. Modified Custom GATT Service Client onNotify extra measurement characteristics are handled and timer disabled.

This code also eliminated reporting of temperature and accelerometer data, even if it is not shown in Listing 7 and 8 above. However, both values can be derived from the existing sample apps by testing with two separate apps and devices same time and synchronizing the data using timestamps when data is combined.

In this prototype the server side in mobile phone is not yet implemented, however nrF Connect is able to connect to the device app enabling testing. Similar flipped client-server architecture was used in KÄVELI project explained in Chapter 2.6.3. And importance of maintaining interoperability in GATT was explained in Chapter 2.2.4. The functionality should allow listeners of Bluetooth advertisements to receive Bluetooth advertisement packets and reach the measurements from the 31 Byte payloads. Fourth prototype will be presented in next section.

4.5 Setup for Fourth Prototype – Mood and breathing

Breathing data alone is not enough to understand cause of stress that makes breathing shallower upper chest breathing, thus additional data is needed. This prototype added Daylio mood reporting to be added with the breathing data to later analyse and identify causes of stress and potentially try to prevent those situations with the help of breathing notifications or another less disruptive means like nudging or tunes. However free Daylio app only gives one tag per day and uses text string in its exported csv files. Thus csv file was modified and mood code was added in front of mood text string to be

used later. There are moods from radiant coded as 0 and awful coded as 4 and good, neutral and bad are in between. In Table 4 below is listed few moods related to activities.

Table 4. Daylio report with added mood code from 0 to 4 or radiant to awful.

full_date	date	weekday	time	mood_code	mood	activities	note
2019-06-03	June 3	Monday	06:06	4	awful	"date"	"Blind date gone bad."
2019-05-30	May 30	Thursday	09:27	0	rad	"relax"	"Read a good book"
2019-05-29	May 29	Wednesday	20:00	2	meh	"cleaning"	"
2019-05-28	May 28	Tuesday	12:02	1	good	"dog"	"
2019-05-28	May 28	Tuesday	11:45	1	good	"work"	"Thesis read references"

Additionally data from Oura cloud storage was taken to show night time respiratory rate and average resting heart rate curves from eight consecutive nights visible in Figure 23 below and csv file contents in Table 5 below. Typically respiration and hear rate changes in night have similar direction but not always as Wed 27 values indicate, slightly lower resting heart rate while slightly higher respiratory rate than previous night is measured.

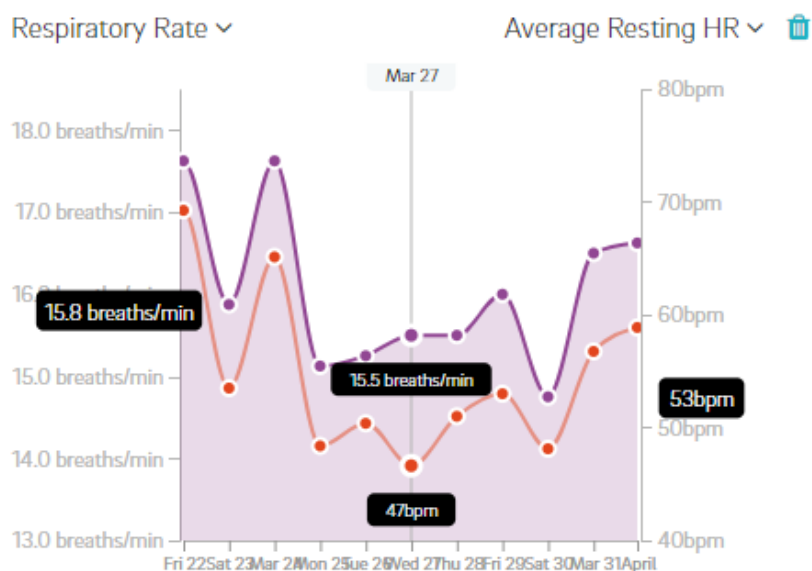


Figure 23. Average respiratory rate and resting HR from eight consecutive nights measured with Oura ring and graph copied from cloud.ouraring.com. Wed 27 shows slightly different direction for respiratory (higher) and HR (lower) compared to other values.

Values in Table 10 below could be used in combination of Movesense device values either to analyze both night time and day time respiration and HR, or in verifying results of measurements done night time with both Oura ring and Movesense device. Other relevant attributes like average HRV, calculated from night time rMSSD values could, be useful as well. In general Oura measures night time sleep and daytime activity and calculate readiness score from those. Multiple Oura variables could be useful for breathing application.

Table 5. Respiratory rate and average resting heart rate from Oura ring in csv file.

```
date,Respiratory Rate,Average Resting Heart Rate
2019-03-22,17.625,69.25
2019-03-23,15.875,53.5
2019-03-24,17.625,65.125
2019-03-25,15.125,48.375
2019-03-26,15.25,50.375
2019-03-27,15.5,46.625
2019-03-28,15.5,51
2019-03-29,16,53
```

Moodmetric ring could provide measurements indicating changes in skin conductance used for detecting arousal, thus allowing detection of stress reactions. Measurements are provided as GATT advertisement messages, to be either read from ring or received as notifications in interval of 3 to 16 samples per seconds from ring. Further, mobile sample application for Android and iOS is provided as part of Moodmetric SDK. Additionally accelerometer data is provided allowing additional source of verification of Movesense sensors measurements.

So offline combination of different sensor data is possible. Combined data could then be analysed for insights of behavioural patterns that could increase awareness of user and allow reflection and decision for change to happen. Furthermore change could be visible in continued measurements. More sources of either automatic detection or self-reporting could be added to system in order to create more holistic view of user's life. Those ideas and more will be looked at in Chapter 6. Next chapter will summarize theory sections in Chapters 2 and 3 and implementation sections in this chapter.

5 Results and Analysis

This chapter analyses outcome of this study with respect to the objective for this study. And in addition to that reflects on ideas emerging from the human-centred design and open platforms. This study was continuation for the Double diamond design process, after the first diamond ended in defining principles for development of health and fitness wearables from the user experience perspective. This study project was a Develop phase in the design process and aimed prototyping with open Movesense platform assets in building prototypes of breathing application.

Assets of open platform included APIs for connecting with both device app and mobile app using REST type protocol and JSON messaging format. And standard Bluetooth Low Energy wraps those messages and carries them between client and server. Another asset is community involved in Movesense technology development. In the community source code, plugins, ideas and solutions are shared in hope of cooperation and partnership and eventually profit from launched solutions. In two years number of partners has grown to twelve and showcases to eleven, and forty others are soon launching solutions. And numerous projects, research papers, and experiments have been created as well, during last two years since opening of Movesense platform.

User experience that was in the core of my earlier thesis handled very much the same research than positive computing. Although, positive computing much more fully synthesized theory and provided concrete strategies and methods to apply in the field of technology development, as well as in psychology or economy. Change of mindset is slow but on-going, since wellbeing is increasingly underscored and acknowledged even not yet fully understood. However, looking things from another perspective, user or human perspective is not only refreshing for technology-minded person, but also essential for successful implementation of wearable solutions for health and wellness.

Applying technology in health or wellness context is rewarding, since it is easy to relate to, furthermore it gives greater purpose for work. On the other hand, slow change of structures in healthcare is frustrating, but on the other hand, change need to be managed carefully not to be life threatening. Converging consumer market and medical market creates opportunities for self-care solutions that could potentially change epidemic chronic illness statistics not to mention the uplift in quality of life.

Prototypes created in this study were attempts to explore possibilities of open Move-sense platform and same time apply some of the positive computing and user experience lessons learned. First prototype concentrated on data handling, second prototype on visualization, third prototype on communication, fourth prototype on fusing data sources. And fifth was planned to be about sensor fusion, and sixth about system, looking into cloud, virtualization, and beyond mobile app and sensor in the network. As a collection of building blocks those prototypes are valuable to me, but not necessarily to the community. One of the reasons is that those ideas and implementations get old so fast while platform is developing. Another reason is that topic might not be interesting enough when compared to sports and health, wellness will be left last. However, future ideas will be handled more in the next and final section.

6 Conclusions

In retrospective, the objective was good and so was the plan of prototypes but the execution was not rigorous enough to complete all planned six prototypes. So, prototyping continues for a while, maybe takes more holistic approach by introducing that toolkit aspect, the ultimate goal of not one tool but set of tools for wellbeing. Movesense platform has last year introduced module concept, so instead of library or app, single modules could be saved and reused in the repository. This could be a nice next step to think about, both in terms of learning and giving back to community.

The topic of wearables, wellbeing and open platforms will be of interest in the future as well. Thus, this opportunity to get deeper into the knowledge of the subjects will not be wasted. Like was mentioned in the introduction, there are plenty of problems and not enough solutions. Picking one to start with will be easy, and hopefully meaningful. Reusability of wearables will encompass all three subjects, since open platforms are enablers of reusability. And reusability is opposite of consumerism, which is considered to be detrimental for wellbeing in individual and global level. So having high hopes of Movesense device being fully functional in 2027, maybe apps or modules or LOCs will look different but still the hardware is the same. So in other words, two problems solved since long-term use of wearables has been extremely rare, at least in the low-end products, which might not be used even six months.

Even reusability of Movesense device or sensor looks good on user perspective, allowing measurement of motion, posture and heart rate, and detecting defined patterns depending on current tracking demands of user, combining that data with other data from other sources to create holistic view of user's life is not so easy. Devices using standard Bluetooth communication might not allow direct access but instead access through cloud storage or mobile apps. Rings like Moodmetric, is example of direct access, whereas Oura is example of cloud access, although app to app connection is possible. Further, hardware related delays might cause synchronization issues between devices, thus upper level synchronization is required where such delays can be eliminated, like in FSenSync case. Consequently, sensor fusion often requires aggregation of data in mobile phone with cloud connection or with synchronization over local Wi-Fi network. Furthermore, combining csv files is required, in order to aggregate data, unless direct API calls are allowed. Nevertheless, aggregation of data will only be first step in using it, while next and bigger step will be to analyse data to detect patterns.

Patterns related to wellbeing data are often behavioural, reflecting user beliefs and actions, and thus, are much needed if user decides to change behaviour. Presenting data in user-meaningful ways is challenge for developer, and so is giving reflective feedback. Solution should also have intuitive interface to ease up use and feedback in form of notifications or alarms, which should be subtle and non-destructive not to steal attention from task at hand. Enough control of appearance of interface should be allowed for sense of control and autonomy for user. Most important wellbeing factors usable for breathing application are those of self, so called intrapersonal factors, like positive emotions, motivation and engagement, self-awareness, mindfulness, and resilience. Self-compassion and gratitude could also be applicable for breathing application as a form of cognitive behavioural activities in order to reframe negative thoughts to positive and verifying effect on respiratory rate.

References

- Abelson B. (2015) Releasing the Diaphragm. [online] Available at: <https://www.youtube.com/watch?v=O84oKZhy2JU> [Accessed 5 March 2019]
- Bluetooth (n.d) GATT Overview, Bluetooth SIG, available [online] Available at <https://www.bluetooth.com/specifications/gatt/> [Accessed 9 May 2019]
- BluetoothLe (2018) ACR Bluetooth LE Reactive Plugin for Xamarin and Windows. [online] Available at: <https://www.nuget.org/packages/Plugin.BluetoothLE> [Accessed 30 May 2019]
- Bohemian Vocal Studio (2018) What is Diaphragmatic Breathing Singing? [online] Available at: <https://bohemianvocalstudio.com/how-to-sing-from-your-diaphragm-4> [Accessed 5 March. 2019]
- Buchheit M. (2014). Monitoring training status with HR measures: do all roads lead to Rome? *Frontiers in physiology*, 5, 73. doi:10.3389/fphys.2014.00073
- Calvo R. & Peters D. (2014). *Positive Computing: Technology for Wellbeing and Human Potential*. 10.7551/mitpress/9764.001.0001.[book]
- Campbell S. (2016) How To Stretch your Diaphragm: Uddiyanabhandha : Yoga for Freediving with Sara Campbell. [online] Available at: https://www.youtube.com/watch?v=Tx6HFZD_O8o [Accessed 5 March. 2019]
- Design Council (2008). The 'double diamond' design process model. [online] Available at: <http://webarchive.nationalarchives.gov.uk/20080821115409/designcouncil.org.uk/en/about-design/managingdesign/the-study-of-the-design-process/> [Accessed 5 May 2017]
- Dcrainmaker Movesense (2017) First Look: Suunto's new Movesense advanced sensor platform. [online] Available at: <https://www.dcrainmaker.com/2017/01/first-look-suuntos-new-movesense-advanced-sensor-platform.html> [Accessed 27 Nov 2016]
- Dcrainmaker Transition (2019) Suunto announces decommission plans for Movescount platform, impacting some watches. [online] Available at: <https://www.dcrainmaker.com/2019/01/suunto-announces-decommission-plans-for-movescount-platform-impacting-some-watches.html> [Accessed 29 May 2019]
- First Beat HRV Summit Wearables (2019) Key Takeaways About the Future of Wearables from Firstbeat Summit 2019 [online] Available at: <https://www.firstbeat.com/en/blog/key-takeaways-about-the-future-of-wearables-from-firstbeat-summit-2019/> [Accessed 29 May 2019]

- Flatt A. (2017) Improving HRV Data Interpretation with the Coefficient of Variation. [online] Available at: <https://elitehrv.com/improving-hrv-data-interpretation-coefficient-variation> [Accessed 13 May 2019]
- Fröger K. (2018) Fröger Analytics FSenSync synchronized multisensory streaming system. [online] Available at: <http://forger.fi/fsensync/> [Accessed 29 Nov 2018]
- Heartmath (n.d) Chapter 03: Heart Rate Variability: An Indicator of Self-Regulatory Capacity, autonomic Function and Health in: Science of the Heart [ebook] Available at: <https://www.heartmath.org/research/science-of-the-heart/heart-rate-variability/> [Accessed 10 Jun. 2016]
- GuerrillaZen Fitness (2015) Breathing Exercises For Posture | Diaphragm Activation. [online] Available at: <https://www.youtube.com/watch?v=EdGC4Qr3hfM> [Accessed 5 March. 2019]
- Healthline (2018) What Is Diaphragmatic Breathing? [online] Available at: <https://www.healthline.com/health/diaphragmatic-breathing> [Accessed 10 March. 2019]
- Jauhiainen M. et al. (2019) Identification of Motor Symptoms Related to Parkinson Disease using Motion-Tracking Sensors at Home (KÄVELI): Protocol for an Observational Case-Control Study [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6456828/> [Accessed 15 May 2019]
- Kristjansson, A. H., Jensen, T., & Hildre, H. P. (2004). The term platform in the context of a product developing company. In DS 32: Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia. [online] Available at: <https://www.designsociety.org/publication/19770/THE+TERM+PLATFORM+IN+THE+CONTEXT+OF+A+PRODUCT+DEVELOPING+COMPANY> [Accessed 23 May 2019]
- Linnakko T. (2018) Connect a Suunto Movesense sensor to IBM Cloud. [online] Available at: <https://developer.ibm.com/recipes/tutorials/connect-a-suunto-movesense-sensor-to-ibm-cloud/> [Accessed 5 May 2018]
- Löfblom J. (2017) Defining User Experience Principles for Developing Health and Fitness Wearables and Smartwatches. [online] Available at: <http://www.theseus.fi/handle/10024/129310> [Accessed at 4 June 2017]
- Nordicsemiconductor (n.d) nRF52832 Product Specification v1.4. [online] Available at: https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf [Accessed at 3 May 2018]
- Magana (2012) Waiting to inhale: why it hurts to hold your breath. [online] Available at: <https://www.scq.ubc.ca/waiting-to-inhale-why-it-hurts-to-hold-your-breath/> [Accessed at 4 June 2017]

- Mokka (2013) Mokka, Motion kinematic & kinetic analyser. [online] Available at: <https://biomechanical-toolkit.github.io/mokka/> [Accessed at 25 November 2018]
- Motion-labs systems (2019) Downloadable software applications. [online] Available at: https://www.motion-labs.com/index_downloads.html [Accessed at 3 June 2018]
- Movesense (2018) Movesense site with news, showcases, opportunities, developers links. [online] Available at: <https://www.movesense.com/> [Accessed at 5 May 2018]
- Movesense DevKit (2017) Movesense DevKit Launched Publicly. [online] Available at: <https://www.movesense.com/news/2017/06/movesense-publicly-launches-software-developer-kit-accelerate-motion-sensing-technology/> [Accessed at 5 May 2018]
- Movesense CES (2019) Five Companies to Launch Their Movesense Products at CES 2019 with Suunto. [online] Available at: <https://www.movesense.com/news/2018/12/five-companies-to-launch-their-movesense-products-at-ces-2019-with-suunto/> [Accessed at 10 May 2018]
- Movesense Connector (2019) Movesense connector pinout. [online] Available at: https://www.movesense.com/wp-content/uploads/2017/01/Movesense_Connector_Tech_Sheet_001-1_20170102.pdf [Accessed at 10 May 2019]
- Movesense Developers (2017) Movesense Developers. [online] Available at: <https://www.movesense.com/developers/> [Accessed at 10 August 2017]
- Movesense May meetup (2019) Movesense Meetup was Full of Energy and Enthusiasm. [online] Available at: <https://www.movesense.com/news/2019/05/energy-and-enthusiasm-in-movesense-meetup/> [Accessed at 16 May 2019]
- Movesense news 01 (2019) Movesense Tools saved customers over three decades in wearable development. [online] Available at: <https://www.movesense.com/news/2019/01/movesense-tools-saved-customers-over-three-decades-in-wearable-development/> [Accessed at 16 May 2019]
- Movesense Raspi (2019) Movesense raspi gateway in Bitbucket. iOS [online] Available at <https://bitbucket.org/suunto/movesense-raspi-gateway/src/master/> [Accessed at 16 May 2019]
- Movesense Xamarin (2018) New Beta of Movesense net released now support Xamarin Android and Xamarin iOS [online] Available at: <https://www.movesense.com/news/2018/07/new-beta-of-movesense-net>

released-now-supports-xamarin-android-and-xamarin-ios/ [Accessed at 16 May 2019]

Movesense meetup1 (2017) Movesense First Meetup. [online] Available at: <https://www.movesense.com/news/2017/06/highlights-first-movesense-developer-meetup/> [Accessed at 16 June 2017]

Movesense Meetup1 Video (2017) Video stream of first Movesense meetup in Suunto office in Helsinki [video online] Available at: <https://www.facebook.com/MovesenseOfficial/videos/308016116314477/> [Accessed at 16 June 2017]

MovesenseDotNet (2019) Movesense .NET SDK for Xamarin Android and Xamarin iOS. [online] Available at: <https://github.com/AndyCW/MovesenseDotNet> [Accessed at 30 May 2019]

Movesense Firstbeat (2019) Firstbeat Introduces new Firstbeat Sports Sensor with Movesense. [online] Available at: <https://www.movesense.com/news/2019/05/firstbeat-introduces-new-firstbeat-sports-sensor-with-movesense/> [Accessed at 21 May 2019]

Movesense Sports Tracker (2018) Build Awareness for your Movesense Based Product in Sports Tracker. [online] Available at: <https://www.movesense.com/news/2018/12/build-awareness-for-your-movesense-based-product-in-sports-tracker/> [Accessed at 29 May 2019]

MyData Declaration (2017) [online] Available at: <https://mydata.org/declaration/>

Pakkala O. (2018) Movesense-anturin soveltuvuus hengityksen tutkimiseen. . [online] Available at: <https://www.theseus.fi/handle/10024/148294> [Accessed at 1 December 2018]

Peer J. (2018) Reliability and validity of a new accelerometer-based device for detecting physical activities and energy expenditure. [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6186411/> [Accessed at 4 June 2019]

Peters D. (n.d) The Responsible Design Process. [online] Available at: <http://www.positivecomputing.org/p/process.html> [Accessed at 1 June 2018]

Portaankorva (2018) Evaluation Of Bluetooth Low Energy Technology For Team Sports Real-Time Physiological Monitoring. [online] Available at: <https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/26570/portaankorva.pdf?sequence=1&isAllowed=y> [Accessed at 1 June 2019]

Restcase (2015) REST APIs and their Gain Added Importance on the Rise in Application Integration Design. . [online] Available at: <https://blog.restcase.com/rest-apis-and-their-gain-added-importance-on-the-rise-in-application-integration-design/> [Accessed at 2 June 2019]

Runteq (2019) [online] Available at: <http://www.runteq.com/> [Accessed at 29 May 2019]

- Scitech Europa (2018) The collecting of motion related and physiological field data quickly. [online] Available at: <https://www.scitecheuropa.eu/movesense-physiological-field-data/87018/> [Accessed at 22 May 2019]
- Semiconductorstore (2017) Bluetooth 5 versus Bluetooth 4.2, what's the difference? [online] Available at: <https://www.semiconductorstore.com/blog/2017/Bluetooth-5-versus-Bluetooth-4-2-whats-the-difference/2080/> [Accessed at 31 May 2019]
- Simtk (2018) OpenSim Marker (.trc) Files. [online] Available at: [https://simtk-confluence.stanford.edu:8443/display/OpenSim/Marker+\(.trc\)+Files](https://simtk-confluence.stanford.edu:8443/display/OpenSim/Marker+(.trc)+Files) [Accessed at 25 May 2018]
- Skawinski K., Roca F.M, Peirovifar P (2018) Exercise recognition and workout supervision using Deep Learning techniques. [online] Available at: https://mycourses.aalto.fi/pluginfile.php/911154/mod_folder/content/0/Report_Group03.pdf?forcedownload=1 [Accessed at 22 May 2019]
- Smith (2015). Unit 2: The Continuation of Life. Chapter 24: Regulating Mechanisms. Slide 23: SUMMARY: The effect of Exercise on the Respiratory System: Homeostatic control. An example of negative feedback control. [online] Available at: <https://slideplayer.com/slide/5937341/> [Accessed at 28 May 2019]
- StudyLib The Nervous System (n.d) Science/Health Science/Neurology (MHR) Chapter 11: The Nervous System, Section 11.3: The Peripheral Nervous System. The Autonomic System. [online] pp.396-399 Available at: <https://studylib.net/doc/18072623/chapter-11-the-nervous-system> [Accessed at 28 May 2019]
- StudyLib The Respiratory System (n.d) Science/Health Science/Cardiology (n.d) Apply Basic Exercise Science to Exercise Instruction. Theory Workbook. [online] Available at: <https://studylib.net/doc/10183724/respiratory-system> [Accessed at 28 May 2019]
- Suunto Sports-Tracker (2015) Sports Tracker joins the Amer Sports Digital Services family to help accelerate the group's digital capabilities. [online] Available at: <https://www.suunto.com/News/Sports-Tracker/> [Accessed at 29 May 2019]
- Suunto Transition (2018) Digital Service Transition. [online] Available at: <https://www.suunto.com/transition> [Accessed at 29 May 2019]
- Symbio [2018] Symbio Smart Safety datasheet [pdf] Available at: <https://www.movesense.com/wp-content/uploads/2018/11/Symbio-Smart-Safety-data-sheet.pdf> [Accessed at 29 May 2019]
- Swagger usage (2019) Swagger (software) Wikipedia [online] Available at: [https://en.wikipedia.org/wiki/Swagger_\(software\)](https://en.wikipedia.org/wiki/Swagger_(software)) [Accessed at 26 May 2019]

- Tekniikka & talous (2016) Suunto ja Sports Tracker tiivistävät yhteistyötä [online] Available at: <https://www.tekniikkatalous.fi/tpaiva/suunto-ja-sports-tracker-tiivistavat-yhteistyota-6306064> [Accessed at 29 May 2019]
- TI (n.d) Bluetooth Low Energy Scanning and Advertising. Texas instruments developers. [online] Available at: http://dev.ti.com/tirex/content/simplelink_academy_cc2640r2sdk_1_12_0_1_16/modules/ble_scan_adv_basic/ble_scan_adv_basic.html [Accessed at 9 May 2019]
- Tikkanen O. (2014) Physiological loading during normal daily life and exercise assessed with electromyography. [online] Available at: https://jyx.jyu.fi/bitstream/handle/123456789/44381/978-951-39-5813-8_vaitos01102014.pdf?sequence=1&isAllowed=y [Accessed at 15 May 2019]
- Unity (2018) Movesense Sensor Plugin in the Unity asset store. [online] Available at: <https://assetstore.unity.com/packages/tools/integration/movesense-sensor-plugin-118242> [Accessed at 28 Nov 2018]
- Wim Hof Method (2018) Breathing Exercises of Wim Hof Method. [online] Available at: <https://www.wimhofmethod.com/breathing-exercises> [Accessed at 21 Nov 2018]
- Yano K., Lyubomirsky S. and Chanceloer J. (2012) Can Technology Make You Happy? Yes, and it can make your office a better place to work, too. Article in IEEE Spectrum. [online] Available at: <https://spectrum.ieee.org/at-work/innovation/can-technology-make-you-happy> [Accessed at 28 May 2019]
- Zaccaro, A. et. al. (2018) How Breath-Control Can Change Your Life: A Systematic Review on Psycho-Physiological Correlates of Slow Breathing. [online] Available at: <https://www.frontiersin.org/articles/10.3389/fnhum.2018.00353/full> [Accessed at 28 May 2019]