



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences



Pedro Miguel Caridade Gomes

Student No. XXXXXXXXXX

Development of an Open-Source Python Toolbox for Heart Rate Variability (HRV)

Thesis submitted to the
University of Applied Sciences Hamburg for the degree of
Master in Biomedical Engineering

Supervisors:

Prof. Dr. Petra Margaritoff, University of Applied Sciences Hamburg
Hugo Silva, PhD, PLUX wireless biosignals, S.A. and IT - Instituto de Telecomunicações

Hamburg, 2018

This work was supported by:



**Hochschule für Angewandte
Wissenschaften Hamburg**
Hamburg University of Applied Sciences

Faculty Life Sciences | Department of Biomedical Engineering
Ulmenliet 20 | 21033 Hamburg | Germany



WIRELESS BIOSIGNALS S.A.

Av. 5 de Outubro, 70-8 | 1050-059 Lisbon | Portugal



**instituto de
telecomunicações**

Instituto Superior Técnico - Torre Norte - Piso 10
Av. Rovisco Pais, 1 | 1049 - 001 Lisboa | Portugal

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

**Development of an Open-Source Python Toolbox
for Heart Rate Variability (HRV)**

Pedro Miguel Caridade Gomes

Abstract

Heart Rate Variability (HRV) is a continuously growing research sector, for which an increasing number of new measures have been introduced over the recent decades. For this reason, many software tools have been developed to support researchers of this sector. However, closed-source tools prevent source code access to developers, while many open-source solutions face different issues, such as limited methods of HRV feature extraction, lack of technical documentation, or support for less mainstream programming languages. The goal of this work is to provide a fully open-source Python 2.7 toolbox named pyHRV for HRV research and application development.

The implementation of this toolbox is supported by several open-source packages to compute Time Domain, Frequency Domain, and Nonlinear HRV parameters. As for the evaluation, HRV parameters have been computed from 50 Normal-to-Normal Interval (NNI) series of 5 minutes in duration and 50 NNI series of 60 minutes in duration using pyHRV and KUBIOS HRV, the reference software. The NNI series show no sign of pathological arrhythmias.

A multilevel package architecture has been implemented for pyHRV, which gives the user the following computational options using a single line of code: (Level 1) computation of all HRV parameters, (Level 2) computation of domain-specific parameters, or (Level 3) computation of individual parameters. In-code and support documentation is provided for support the implementation of pyHRV. Error catching capabilities (e.g. automatic second to millisecond conversion) have been implemented to reduce the occurrence of errors and increase the toolbox's robustness. pyHRV computes a total of 78 HRV parameters (23 Time Domain, 48 Frequency Domain, 7 Nonlinear), from which in a direct comparison with KUBIOS 12 have achieved identical results, with 38 parameters showing marginal differences, and 26 showing significant differences, thus, computing questionable results.

Overall, pyHRV provides a reliable, versatile, robust and user-friendly toolbox for HRV researchers and application developers using the Python 2.7 programming language. pyHRV has been publicly released on the GitHub repository system.

Acknowledgments

I would like to take the opportunity to start this work by thanking those persons who have supported (or endured) me during my studies and during the development of pyHRV, the HRV toolbox developed within the scope of this work. First of all, I would like to thank Hugo Silva for giving me the opportunity to develop the open-source pyHRV toolbox which allowed me to further explore the field of ECG signals - a signal that has earned my special interest over the last months - and for the continued outstanding support and valuable lessons at PLUX. I would also like to thank Prof. Petra Margaritoff for supporting this work, especially as it is thanks to her, that I have achieved the software development skills that I have now during my studies at the HAW Hamburg.

A tremendous 'thank you!' goes out to my family, especially my parents, Fernanda and Manuel, and my brother Diogo, who have supported me in every way possible throughout the last years to help me get to the point where I am right now. Nothing could pay respect to the support that I have experienced from them. This thank you is also directed to Larissa Montenegro, who has supported me many times during this work... and who endured my moods whenever my *"let's-change-this-parameter-and-see-what-happens"* approach during desperate debugging sessions did not provide positive results.

Finally, I would also like to thank my friends and colleagues, who made all the years at the HAW Hamburg (or in Hamburg in general) an unforgettable experience.

May you all be repaid with many *Pastéis de Nata* in the future.

List of Figures

1	Stages of cardiac tissue excitation and ECG curve profiles	7
2	ECG electrode placement for Einthoven Leads.	9
3	ECG signal with falsely detected R-peak due to motion artifacts.	10
4	ECG signal at different sampling frequencies.	11
5	Relationship between R-peaks, NN intervals, and NN interval differences.	15
6	Sample ECG signal and Tachogram with the derived NNI and HR series.	16
7	Block diagram of the SDNN Index computation.	18
8	Block diagram of the SDANN computation.	18
9	Visualization of the geometrical parameters	22
10	Sample PSD computed using the Welch's method	27
11	Sample PSD computed using the Autoregressive method	28
12	Sample PSD computed using the Lomb-Scargle periodogram.	29
13	Comparison of Poincaré plots with low and high HRV.	33
14	The Poincaré plot and parameters	34
15	Detrended Fluctuation Analysis plot	38
16	BITalino (r)evolution ECG sensor.	40
17	BITalino (r)evolution Board kit.	40
18	Illustration of the intended workflow.	41
19	Flowchart of the evaluation metrics computation process.	45
20	Tachogram comparison (outliers vs. filtered).	46
21	Multilevel architecture of the Heart Rate Variability (HRV) package.	50
22	Sample ECG signals with different interval lengths.	53
23	Flowchart of the pyHRV function architecture.	59
24	Flowchart of the <code>tools.check_input()</code> function.	60
25	Example KUBIOS NNI histogram.	69
26	PSD NFFT Comparison - Welch vs. Lomb	71
27	Illustration of the Differential Index computation [9].	73

List of Tables

1	Overview of time domain parameters.	14
2	Overview of frequency domain parameters [86].	25
3	Frequency bands for ECG acquisitions of different durations.	25
4	Overview of nonlinear parameters.	32
5	Third party packages & versions used in this work.	44
6	Selected parameter settings set in the toolbox and KUBIOS software to compute HRV results on the same datasets and using equal parameters.	47
7	pyHRV package modules, files and descriptions.	50
8	Functions of the Tools Module.	52
9	Functions of the time domain module.	55
10	Frequency Domain Module functions.	57
11	Functions of the nonlinear parameters module.	58
12	Evaluation results for the Time Domain parameters (pyHRV vs. KUBIOS)	63
13	Evaluation results for the parameters extracted with Welch's method (pyHRV vs. KUBIOS)	64
14	Evaluation results for the frequency domain parameters extracted using the Lomb-Scargle PSD (pyHRV vs. KUBIOS)	65
15	Evaluation results for the parameters extracted using the Autoregressive Method (pyHRV vs. KUBIOS)	66
16	Evaluation results for the Nonlinear parameters (pyHRV vs. KUBIOS)	67
A1	Time domain parameter keys and report descriptions.	XVI
A2	Frequency domain parameter keys and report descriptions.	XVI
A3	Nonlinear parameter keys and report descriptions.	XVII
A4	Plot keys.	XVII

List of Equations

1	Normal-to-Normal intervals (NNI)	14
2	Normal-to-Normal interval differences	15
3	Heart rate	15
4	Standard deviation of successive NN intervals (SDNN)	17
5	Mean of SDNN values of long-term ECG acquisitions (SDNN-Index)	18
5	Standard Deviation of the Mean NN intervals of 5min segments (SDANN)	18
7	Root Mean Square of the Successive NN Interval Differences (RMSSD)	19
8	Standard Deviation of Successive Differences (SDSD)	20
9	Ratio between NNx and Total Number of NN intervals (pNNx)	21
10	Triangular Index (TI)	22
11	Least squares method for the computation of the TINN parameter.	23
12	Triangular Interpolation of the NNI Histogram (TINN)	23
13	Absolute power of a frequency band	29
14	Total power of all frequency bands (discrete integration)	30
15	Total power of all frequency bands (sum of absolute powers)	30
16	Relative power of in a frequency band	30
17	Logarithmic power of a frequency band	30
19	Normalized powers of the LF or HF frequency band	31
20	LF/HF ratio	31
21	Standard deviation along the minor axis of the Poincaré plot (time domain parameter based)	34
22	Standard deviation along the minor axis of the Poincaré plot (vector based)	34
23	Standard deviation along the major axis of the Poincaré plot (time domain parameter based)	35
24	Standard deviation along the major axis of the Poincaré plot (vector based)	35
25	Area of the fitted ellipse in the Poincaré plot	35
26	Sample Entropy - Template vector	36
27	Sample Entropy - Chebyshev distance of vector pairs $i + n$ and $k + n$	36

28	Sample Entropy - Count condition of the parameter A	36
29	Sample Entropy - Count condition of the parameter B	36
30	Sample Entropy - Negative logarithm of the A and B ratio	36

List of Python Snippets

1	Minimum working example of the Python code used for the intended workflow.	41
2	Minimum working example of the Tool Module's functions.	54
3	Minimum working example of the Time Domain functions.	55
4	Minimum working example of the Frequency Domain functions.	57
5	Minimum working example of the Nonlinear Parameter functions.	59
6	General architecture of the HRV parameter functions shown on the example of the RMSSD function.	61
A1	Minimum working example of the <i>OpenSignalsReader</i> package and class. .	XIX

List of Acronyms

ANS	Autonomic Nervous System
AR	Autoregression
BPM	Beats per Minute
DFA	Detrended Fluctuation Analysis
ECG	Electrocardiography
FB	Frequency Band
FFT	Fast Fourier Transform
GUI	Graphical User Interface
HF	High Frequency
HR	Heart Rate
HRV	Heart Rate Variability
HRVA	Heart Rate Variability Analysis
IBI	Inter-Beat-Interval
IDE	Integrated Development Environment
LF	Low Frequency
LSP	Lomb-Scargle Periodogram
NNI	Normal-to-Normal Interval
NNx	Number of NN Interval Differences Greater than x milliseconds
PNS	Parasympathetic Nervous System
pNNx	Ratio between NNx and Total Number of NN intervals

PPG	Photoplethysmography
PSD	Power Spectral Density
RMSSD	Root Mean Square of the Successive NN Interval Differences
RRI	Interval Between Successive R peaks
SamPen	Sample Entropy
SD	Standard Deviation
SDANN	Standard Deviation of the Mean of NN Intervals in all 5 minute Segments
SDNN	Standard Deviation of Successive NN Intervals
SDNNI	Standard Deviation of Successive NN Intervals Index
SDSD	Standard Deviation of Successive Differences
SNS	Sympathetic Nervous System
TINN	Triangular Interpolation of the NNI Histogram
TI	Triangular Index
ULF	Ultra Low Frequency
VLF	Very Low Frequency

Contents

List of Figures	v
List of Tables	vi
List of Equations	vii
List of Python Snippets	ix
List of Acronyms	x
1 Introduction	1
1.1 A Brief History of Heart Rate Variability	1
1.2 State-of-the-Art of Heart Rate Variability Measures	2
1.3 Motivation	3
1.4 Goal of this Work	4
2 Fundamentals of Heart Rate Variability	6
2.1 Physiological Background	6
2.1.1 Heart Rate Regulation Mechanisms	6
2.1.2 Electrocardiography and Electrocardiogram	7
2.2 Single Lead ECG Signals	8
2.2.1 Acquisition of Electrocardiography Signals	8
2.2.2 Signal Filtering, RR Intervals and NN Intervals	9
2.2.3 Sampling Frequency Selection	11
2.3 Overview of Heart Rate Variability	13
2.3.1 Time Domain Parameters	13
2.3.1.1 NN Interval Parameters	14
2.3.1.2 Heart Rate	15
2.3.1.3 Tachogram	16
2.3.1.4 SDNN	16
2.3.1.5 SDNN Index	17
2.3.1.6 SDANN	18

2.3.1.7	RMSSD	19
2.3.1.8	SDSD	20
2.3.1.9	NNx and pNNx	20
2.3.2	Geometrical Parameters	21
2.3.2.1	Triangular Index	21
2.3.2.2	Triangular Interpolation of NNI Histogram (TINN)	22
2.3.2.3	Physiological Context of Geometrical Parameters	23
2.3.3	Frequency Domain Parameters	24
2.3.3.1	Frequency Components	25
2.3.3.2	Welch's Periodogram	26
2.3.3.3	Autoregression Periodogram	26
2.3.3.4	Lomb-Scargle Periodogram	28
2.3.3.5	Frequency Parameters	29
2.3.4	Nonlinear Parameters	32
2.3.4.1	Poincaré Plot	32
2.3.5	Sample Entropy (SampEn)	35
2.3.6	Detrended Fluctuation Analysis (DFA)	37
3	Materials and Methods	39
3.1	Intended Workflow	39
3.2	Third-Party Tools, Software and Packages	42
3.3	Evaluation Procedures	44
4	Results	49
4.1	Implementation of the HRV Toolbox	49
4.1.1	Package Architecture	49
4.1.1.1	HRV Tools Module	51
4.1.1.2	Time Domain Module	54
4.1.1.3	Frequency Domain Module	56
4.1.1.4	Nonlinear Parameters Module	58
4.1.2	Function Architecture	59
4.2	Evaluation	63
4.2.1	Time Domain Parameters	63
4.2.2	Frequency Domain Parameters	64
4.2.3	Nonlinear Parameters	67
5	Conclusions	68
5.1	Discussion	68

5.2 Future Work 72
 Bibliography I

Appendices **X**

A Sample HRV Report XI
 B Parameter Keys & Report Descriptions XV
 C Developed Support Packages XVIII
 D BITalino (r)evolution Board Datasheet XX
 E BITalino (r)evolution ECG Sensor Datasheet XXIV
 F OpenSignals (r)evolution Datasheet XXVII
 G OpenSignals (r)evolution File Format Description XXIX

Chapter 1

Introduction

1.1 A Brief History of Heart Rate Variability

The Chinese physician Wang Shu He (180-270 A.D.) was one of the first documented physicians to have had observed the pulse-to-pulse variability of sickened patients and recognized its role as indicator of mortality stating that *"if the pattern of the heartbeat becomes as regular as the tapping of a woodpecker or the dripping of rain from the roof, the patient will be dead in four days"* [54]. Although other physicians have investigated the human pulse and its nature even the centuries before (between ca. 200 B.C. and ca. 200 A.A.) and despite the technical limitations of that time - especially compared to the modern times of multi-lead Electrocardiography (ECG) acquisition and processing systems - Wand Shu He had discovered a remarkable indicator for cardiac health, which only centuries after his observation has been continued to be investigated [8].

In modern medicine, the importance of pulse-to-pulse variability as measure of cardiac and overall health has been established due to the extensive research conducted over the last decades. This has been made possible thanks to the fast pace of technical advancements in the medical field [1, 86]. While Wang Shu He measured the pulse using a palpation method, by placing three fingers on different locations near the patient's wrist, modern physicians and researchers use ECG or Photoplethysmography (PPG) systems and computational methods to extract the Heart Rate (HR) information and to derive a variety of parameters to assess its variability. Thanks to these efforts, this method of Heart Rate Variability Analysis (HRVA) has seen an increasing growth in application fields and number of parameters that can be derived from the Inter-Beat-Interval (IBI) series, i.e. the

intervals between successive heart beats [8, 32].

In the 1960's, the first Time Domain parameters have been introduced, which consisted of a series of basic statistical parameters derived from the IBI series. In the following three decades, many methods of frequency analysis and non-linear assessments have been suggested and partially added to the family of HRV parameters. This increase in HRVA methods have primarily been made possible thanks to the technical advancements in computational resources, which allowed the computation of more complex methods [8]. However, a practical problem existed for cardiologists, where due to the great variety of computational methods, and the lack of knowledge in the field of HRV, misleading physiological interpretation of the results could potentially occur. The *Task Force of The European Society of Cardiology and The North American Society of Pacing Electrophysiology* tackled this problem, conducted a more in-depth investigation of suggestion methods, and issued the HRV guidelines ("Heart Rate Variability: Standard of Measurement, Physiological Interpretation, and Clinical Use") in 1996, where a selection of suitable parameters has been made according to clinical standards and physiological context known at the time [86]. Nowadays, the role of HRV is well-known in modern medicine. However, its appearance in clinical applications is still rare due to its complexity and yet lack of a general and standardized selection of parameter values that define a healthy subject. It is, however, a very popular research field with approx. 1100 HRV related works being published within the first 9 months of 2018 alone¹.

1.2 State-of-the-Art of Heart Rate Variability Measures

The HRV guidelines managed to conduct a rigorous and in-depth review of the suggested HRV measures at the time. This review did not only put many measures into clinical and physiological contexts, but also established standards for minimum requirements for ECG signal acquisitions and HRV parameters [86]. For instance, many of the suggested Time Domain parameters (e.g., SDNN, RMSSD) have been thoroughly investigated over the last two decades, and their importance has been recognized [24, 48, 93, 98]. In the Frequency Domain, the application of the Fast Fourier Transform (FFT)-based Welch's and the Autoregression (AR) methods of Power Spectral Density (PSD) estimation have been established as popular tools for

¹Number of works based on the search results of the search term "heart rate variability" in the `pubmed.gov` platform (search term in the title/abstract, human research only)

spectral analysis of IBI variability [42, 58]. However, new measures have been introduced to HRVA, especially for the Nonlinear analysis domain, where the guidelines failed to establish common standards due to the lack of clinical evidence.

The Poincaré plot, for example, is a popular scatter plot of IBI series often found in HRV research. This tool provides a graphical visualization of the overall HRV, which facilitates the identification of healthy patients vs. patients with existing cardiac conditions, solely based on the form of the scatter plot [25, 46, 102]. Additionally, other measures for the analysis of non-linearity have been introduced such as the Approximate Entropy, the Sample Entropy (SamPen), Detrended Fluctuation Analysis (DFA), and Multi-Scale Entropy analysis, which are commonly in HRV research [22, 64, 73, 82]. In the Frequency domain, an additional method of PSD estimation has also been introduced, the Lomb-Scargle Periodogram (LSP) [27]. This method, in comparison with FFT and AR methods, has the main advantage that it is cable compute PSD estimation for irregularly sampled times series, which is the case in IBI series. Reason for which it is an interesting measure for newer HRV applications, such as real-time HRVA [71].

1.3 Motivation

There are many HRV software libraries available in the open- and closed-source domain, with some of them being well-known in the field of HRVA. However, their functionality or usability exhibits several shortcomings when it comes to software development and integration. For instance, commercially available tools, such as the well-established *KUBIOS HRV*² software provide well-designed Graphical User Interface (GUI), but lack insights into the source code underneath the GUI with no possibility of extracting single algorithms or features for integration into custom software. Additionally, although free versions of such commercial tools are mostly available, those versions have only limited functionality (e.g. limited duration of processable signals), and the non-restricted usage of all features is only available after the purchase of the software [65].

Available open-source alternatives, on the other hand, do not require purchases of the software and provide open source codes, allowing users to modify the code to fit their specific needs (e.g. extracting only a selection of algorithms). Such solutions are openly available, with many of them being found on the popular *GitHub*

²Developed by Kubios Oy, Kuopio, Finland; <https://www.kubios.com>

repository system³. These tools are also available in a variety of different programming languages, including Swift, Python, Matlab and R [6, 60, 72, 74]. However, these open-source solutions face three major problems. First, the most sophisticated solutions are usually found for less mainstream programming languages, which are designed for specific fields of application or purposes, such as the case of the *RHRV* project for the R programming language [74]. Second, these solutions often provide only limited documentation or poorly commented source code, which complicates the usage and integration of the available libraries [6]. This issue can be frustrating for new entrants in the already highly complex field of HRV. Third, in the specific case of the Python programming language, the available solutions are still limited in the number of the computed HRV parameter functions, thus failing to provide the full range of parameter computations as recommended by the HRV guidelines and newer measures [6]. Forth, some available tools may be argued to be too complex in their use, which may also be an additional hurdle for new entrants [95]. Finally, the solutions above do not provide any evaluation metrics on their performance, i.e. they have not been tested against gold standard solutions.

1.4 Goal of this Work

The goal of this work is to develop a robust, versatile, and user-friendly HRV software toolbox named pyHRV for the Python programming language. This toolbox shall comply with the HRV guidelines, should be easily extended in the future to be updated with new achievements in HRV research. Implementation of error-catching features with understandable exception raisings and automatic data conversion to suitable data formats (ensuring the use of the highly efficient NumPy array format) as well as physical units (converting input data in *s* to *ms*) are intended to ensure the robustness of this toolbox. Versatility is ensured by providing both functions designed for the computation of a single, specific HRV parameter and functions which compute a series of HRV parameters, allowing users to either call individual algorithms or entire domain-specific algorithm sets with a single line of code. This toolbox will then be an extension to the open-source biosignal processing toolbox *BioSPPy* [17]. Additionally, a step-by-step-like written source code style with comprehensive function, variable and parameter names is applied in this toolbox to help and guide the users - especially new entrants in the field of HRV - through the implemented HRV algorithms, and their application. This is supported by a

³Web-based version control service using Git (<https://github.com>)

thorough publicly available technical documentation (API reference) on ReadTheDocs, a platform to support continuous documentation of open-source software⁴. User-friendliness is also supported by the implementation of a multilevel toolbox architecture. This architecture is designed to support the software development process of different types of users, allowing them to either compute individual parameters, entire parameter series of a specific domain, or to compute the entire set of HRV parameters over all domains using only a single line of code. Finally, the results computed with pyHRV are validated, by determining evaluation metrics from the computed results in direct comparison with the HRV gold standard software KUBIOS.

⁴<https://docs.readthedocs.io/>

Chapter 2

Fundamentals of Heart Rate Variability

2.1 Physiological Background

2.1.1 Heart Rate Regulation Mechanisms

The Autonomic Nervous System (ANS) is a subsystem of the central nervous system of the human body, which influences the functionality of the internal organs, including the heart. This system can be divided into three divisions of which two, the Parasympathetic Nervous System (PNS) and the Sympathetic Nervous System (SNS), play a significant role in the regulation of cardiac and other vital activities (e.g. respiration) [55]. The ANS responds to sensory input from the environment or the body itself and responds by stimulating physiological processes through the SNS or by inhibiting them through the PNS. For this reason, the SNS has also become known as the *fight or flight* division, which increases the activity of physiological mechanisms to ensure that all the necessary resources are available in situation of high stress [3]. The PNS, on the other hand, has become known as the *rest and digest* division which reduces the activity of the controlling organs and is linked to relaxation [3, 55]. The activity of these counterpart divisions play a critical role in heart rate regulation, which can be observed and analysed using HRVA methods. For instance, it is known that physical or emotional stress results in an increase in HR which can be indexed as increased activity of the SNS [91]. This alteration in HR is the fundamental information used in HRVA, from which a series of features can be derived to identify the dominant physiological mechanisms

within this process [1].

2.1.2 Electrocardiography and Electrocardiogram

The Electrocardiogram is one of the most characteristic signals of the human body, consisting of the sum of the electrical activity in the cardiac tissue, that triggers the pumping mechanisms of the heart, i.e. the heartbeat. It is the result of an ECG acquisition, one of the most important and well-established methods of cardiac monitoring, where the electrical potential of the cardiac activity is measured using purpose-built ECG sensors and surface electrodes placed on a subject's skin. Its characteristic form is achieved from the successive contraction of different sections of the cardiac tissue, with each section being present in the signal itself.

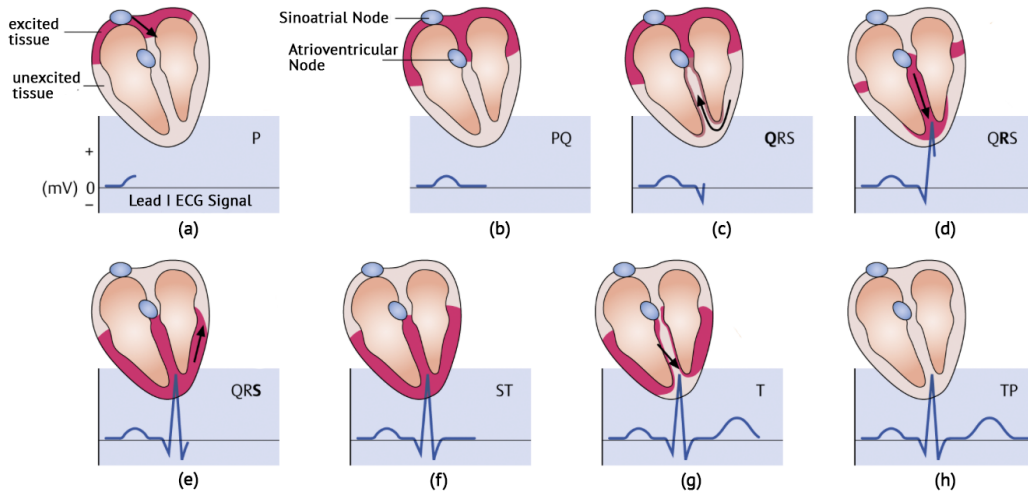


Figure 1: Successive excitement stages of the different cardiac tissues throughout the heart, during a heartbeat and the respective curve profiles of the characteristic ECG signal [29].

Figure 1 shows the different stages of electrical excitation propagating throughout the different tissue sections of the heart in a specific, successive order during a heartbeat. At the beginning of a heartbeat cycle, the heart chambers are filled with blood which will be pumped out of the heart (deoxygenated blood \rightarrow lungs, oxygenated blood \rightarrow body). The signal waveform resulting from the heartbeat is initiated at the Sinoatrial Node, Figure 1 (a), the heart's primary pacemaker that is closely interconnected with the ANS and SNS, therefore being the main influencer of the HR. This excitation, followed by the excitation of the Atrioventricular Node, 1 (b), results in a slight elevation of the ECG signal known as the P- and PQ-segments. Afterwards, the propagation proceeds down the inner ventricular walls, stimulates the heart chambers, and causes the heart to contract and to empty its chambers, Figure 1 (c-e). For HRVA, this signal characteristic is the most important one of

the ECG signal as it is the crucial segment that defines the chamber contraction in the signal from which the fundamental IBI series, i.e. the measured time between successive heartbeats, are derived for HRV parameter computation. Depending on the methods being applied to extract this event from the ECG signal, the entire QRS-segment (shown in Figure 1 c-e) is used as a fiducial complex for heartbeat detection, although the most commonly used methods are designed for the detection of the R-peak only, Figure (d). After the heartbeat, the heart restores its original volume, to allow the chambers to be filled with blood again. During this action, some repolarisation occurs along the chambers resulting in the T-segment of the wave. The entire process is then repeated for each heartbeat [29].

2.2 Single Lead ECG Signals

HRVA is based on datasets of IBI, i.e. interval durations between successive heartbeats. Conventional applications in medical research and practices derive these datasets from ECG signals [3, 51, 52, 91]. It must be noted that other sensors besides ECG sensors can be used to derive series of IBI data on which HRVA can be conducted. For instance, sensors measuring blood flow variations such as Photoplethysmography (PPG) sensors or SpO₂ sensors, with the capability of measuring fluctuations in blood oxygen saturation can be used for this purpose, given that the nature of both signals is found in the heart's pumping mechanism [34, 79, 97].

The work presented in this thesis is focused on single-lead ECG signals derived from the Einthoven Leads. For this reason, this section is intended to explain and raise awareness of technical specifications, processes, and concerns that should not be neglected when using single lead ECG signals for HRVA.

2.2.1 Acquisition of Electrocardiography Signals

Modern cardiac monitoring methods in medical applications acquire ECG data in standardized electrode configurations of 3, 6, 12 and even 22 ECG leads, among other possible and application-specific leads not named here [28, 39]. Although any of the acquired leads could technically be used for HRVA - after all, the signal peak (R-peak) is the most critical required information - commonly found derivations being used in HRV research are the Einthoven Leads [3, 51, 52, 91]. The 3 bipolar Einthoven leads (also known as the *Standard Limb Leads*) are acquired by placing positive (+) and negative (-) electrodes at the skin surface of a subject's right arm

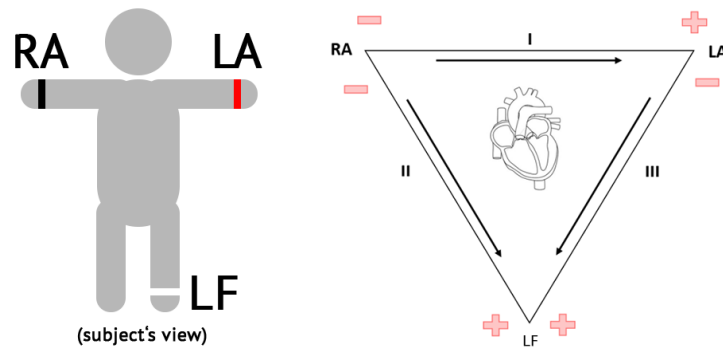


Figure 2: Illustration of the electrode placement for the Einthoven Leads (left) and the Einthoven-Triangle (right) [89].

(RA), left arm (LA) and left foot (LF) where the potential difference of the heart is measured in the following directions as illustrated in Figure 2:

- Lead I: RA(-) \rightarrow LA(+)
- Lead II: RA(-) \rightarrow LF(+)
- Lead III: LA(-) \rightarrow LF(+)

The Lead I derivation has the advantage of its simplicity, which makes it attractive even for applications outside the medical field. In the recent past, off-the-person ECG acquisition methods have been investigated and even implemented in end-consumer products using the Lead I configuration [18].

2.2.2 Signal Filtering, RR Intervals and NN Intervals

In most cases, signal processing methods must be applied on single lead ECG signals before conducting a HRVA, given that noise or other signal artifacts can influence the overall results. For this reason, filters are applied on the signal to filter out noise, ectopic heartbeats, baseline wandering (e.g. T-P knot algorithm, high-pass filter) and signal artifacts (e.g. motion artifacts) before applying R-peak detection algorithms to prevent the detection of false R-peaks, i.e. to prevent the detection of signal peaks that are not caused by an actual heartbeat. Many algorithms and signal processing methods, both in the analog and digital domain, have already been proposed for this purpose, such as the application of high-pass and band-pass filters, morphological transformation algorithms, wavelet transformation, and many others [13, 62, 63]. Due to the complexity of such signal processing methods, and given that there is already an extensive amount of literature about such methods, those

will not be further discussed in this work. Nonetheless, their importance must not be neglected.

After filtering the signal, R-peak or QRS-complex detection algorithms (e.g. *Pan Tompkins Algorithm* [68]) are applied to identify the R-peaks in the ECG signal. This information allows one to measure the duration between successive R-peaks upon which the HRV parameters can be computed.

In HRV applications, two commonly used terms for the derived IBI series are used: RR-intervals and NN-intervals. RR-intervals, i.e. Interval Between Successive R peaks (RRI), are the durations between detected R-peaks in the ECG signal, where it is often not immediately clear whether this includes or not falsely detected R-peaks, due to signal artifacts or ectopic heartbeats. These falsely detected R-peaks could have unwanted influences in the HRVA and distort the actual results. For instance, when conducting frequency analysis, such 'R-peaks' can introduce incorrect frequency components and significantly alter the results due to the distortion of the fundamental RRI series [69]. An example of such a problem is illustrated in Figure 3 (top), where it can be observed that an additional (false) RRI has been detected leading to a significant separation of a single RRI of 760ms into two individual intervals of 532ms and 228ms, thus significantly altering the actual information.

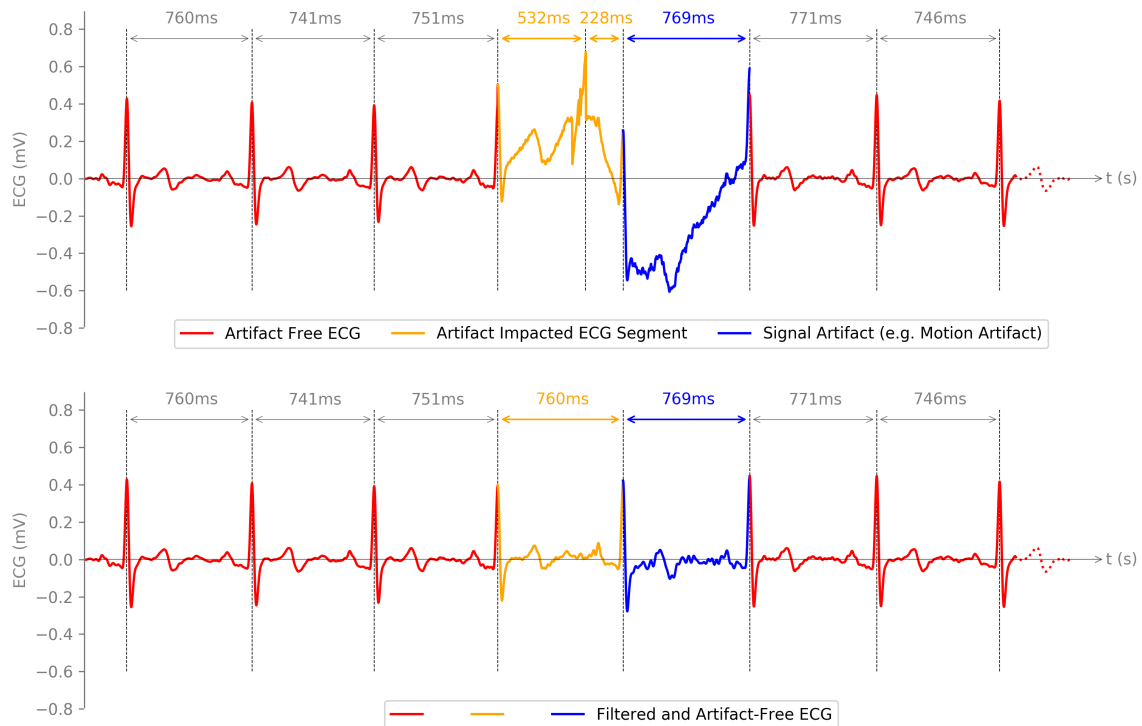


Figure 3: ECG signal with falsely detected R-peak due to motion artifacts.

Normal-to-Normal Interval (NNI) also measure intervals between detected R-peaks,

however, it emphasizes that signal (pre-)processing techniques have been applied to the ECG signal before the application of R-peak detection algorithms, thus, providing interval measures between real R-peaks with no influences of falsely detected signal peaks or ectopic heartbeats. An example of a filtered signal with detected normal R-peaks and RRI - or in this case NNI - intervals is illustrated in Figure 3 (bottom). The present work uses the NNI series as the fundamental input for HRVA.

2.2.3 Sampling Frequency Selection

The selection of a proper sampling frequency for ECG recordings is a technical aspect that must not be neglected, as it can have a significant impact on the accuracy of HRV results. Detection of the QRS-complexes (or of the R-peaks only) in the ECG signal, should be as accurate as possible to ensure a good determination of NNI. Figure 4 shows the issues resulting from the use of low sampling frequencies. Higher sampling frequencies (e.g. 1000Hz; left) provide a higher number of ECG samples near R-peaks allowing more accurate detection of R-peaks and NNI. Lower sampling frequencies, on the other hand, can vary significantly in distance between the actual R-peak location and the available ECG samples due to the decreased number of samples within a set interval (e.g. 250Hz or 100Hz; middle and right respectively¹). These inaccuracies in R-peak detection can significantly influence the measurement of NNI and HRV results [1].

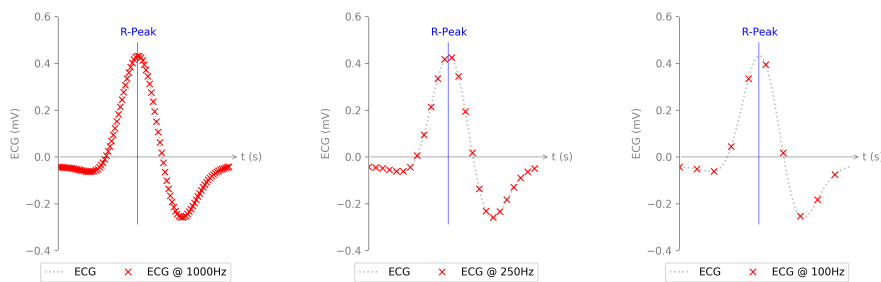


Figure 4: Example of ECG data with different sampling frequencies to illustrate the influence of sample sparsity in R-peak detection as a results of lower sampling frequencies. This inaccuracy can cause NNI variations that greatly influence HRV analysis results.

The guidelines propose a minimum sampling frequency of 128Hz when interpolation is conducted on the signal to increase the accuracy in R-peak detection [86]. Madhiani et al. [52] even went one step further and investigated the reliability of 50Hz

¹ECG samples 250Hz and 100Hz sampling frequencies are downsampled signals of the 1000Hz signal (Figure 4, left)

sampling frequency, considering the importance of such low frequencies in wireless and wearable body monitoring platforms to enable greater power efficiency. Their research showed that reasonable time domain results could be achieved by applying a cubic spline interpolation on ECG data sampled at 50Hz from healthy subjects. However, it must be noted that the cubic spline interpolation is a suitable method for R-peak interpolation in ECG signals of healthy individuals, which containing no pathological arrhythmia, as the symmetric characteristics of the QRS-complex allow an accurate R-peak interpolation. This might be less accurate in non-symmetric QRS-complexes of subjects with existing heart conditions where these symmetric characteristics are distorted (e.g. ST segment elevation) [52]. Hejjel et al. [35] on the other hand have searched for the most adequate sampling frequency, i.e. the frequency at which an interpolation is not required and where the use of higher sampling frequencies would not provide any benefit for ECG signal processing methods. According to their findings, they recommended 1kHz as the most suitable option for ECG acquisitions, providing high accuracy without the need of an additional interpolation.

2.3 Overview of Heart Rate Variability

Generally speaking, HRV describes the variation of IBI from which time, frequency and nonlinear components can be extracted, computed and analysed to assess cardiac and overall health. It is the ability of the HR regulating mechanisms of the human body to react to sensory inputs such as physical or emotional stress or relaxation and other outer influences, by adjusting the HR - and therefore the IBI - accordingly [1]. A simple example of this ability can be observed by comparing the HR of a subject in a resting position with the HR of the same subject during or after physical activity, where an increase in HR can be experienced. Savonen et al. [76] have also observed increases in HR with increasing intensities of physical training. In such cases, the heart is able to adjust its performance by increasing the HR which ensures that freshly oxygenated blood is distributed within the body to keep up with the increased consumption of oxygen in the body in physically or emotionally stressful situations is satisfied at a suitable rate [11].

2.3.1 Time Domain Parameters

The Time Domain parameters contain statistical information from three different classes as recommended by the HRV guidelines: (a) parameters derived directly from NNI or instantaneous HR (e.g., SDNN, SDANN), (b) those derived from the differences between successive NNI (e.g. SDD, RMSSD, NNx, pNNx), and (c) geometrical parameters that are derived from the NNI histogram (e.g., Triangular Index, TINN) [86]. This section aims to present the commonly used time domain parameters including information and equations about their computation and physiological context. An overview of these parameters is presented in Table 1.

Table 1: Overview of time domain parameters.

Time Domain Parameters		
Parameter	Unit	Description
NNI	ms	NN interval parameters (min, max, mean, max difference)
HR	BPM	General heart rate parameters (min, max, mean)
SDNN	ms	Standard deviation of NN intervals
SDNN-i	ms	Mean of SDNN values of long-term ECG acquisitions
SDANN	ms	Std. deviation of the mean NN intervals in 5min segments
SDSD	ms	Std. deviation of successive NN interval differences
RMSSD	ms	Root mean square of successive NN interval differences
NNx	-	Number of NN interval differences greater the threshold x
pNNx	%	Ratio between NNx and total number of NN intervals
NN50	-	Number of NN interval differences greater 50ms
pNN50	%	Ratio between NN50 and total number of NN intervals
NN20	-	Number of NN interval differences greater 20ms
pNN20	%	Ratio between NN20 and total number of NN intervals
Geometrical Parameters		
TriIndex	-	Triangular index (# of NNI/max of NNI histogram)
TINN	ms	Baseline width of the NNI histogram

2.3.1.1 NN Interval Parameters

The computation of a NNI series from a dataset with n R-peaks will returns a dataset with $n - 1$ NNI elements as two R-peaks are required to determine the interval between them and given that the last R-peak R_n has no successive R-peak to conduct such an operation (compare with Figure 5). The computation of a NNI is done according to Equation 1 for $0 < i < n - 1$.

$$NN_j = R_{j+1} - R_j \quad (1)$$

with: NN_j : NN interval i in [ms]
 RR_j : Current R-peak in [ms] or [s]
 RR_{j+1} : Successive R-peak in [ms] or [s]

Conventional parameters extracted from this dataset are minimum NNI, maximum NNI, mean NNI duration and the maximum difference between successive intervals. For the latter parameter, an additional computation has to be conducted which create a dataset of length $n - 2$ with the differences between adjacent NNI according to Equation 2. The relationship between R-peaks, NNI, and NNI differences is illustrated in Figure 5.

$$\Delta NN_j = |NN_{j+1} - NN_j| \quad (2)$$

with: ΔNN_j : NN interval difference in [ms]
 NN_j : Current NNI in [ms]
 NN_{j+1} : Successive NNI in [ms]

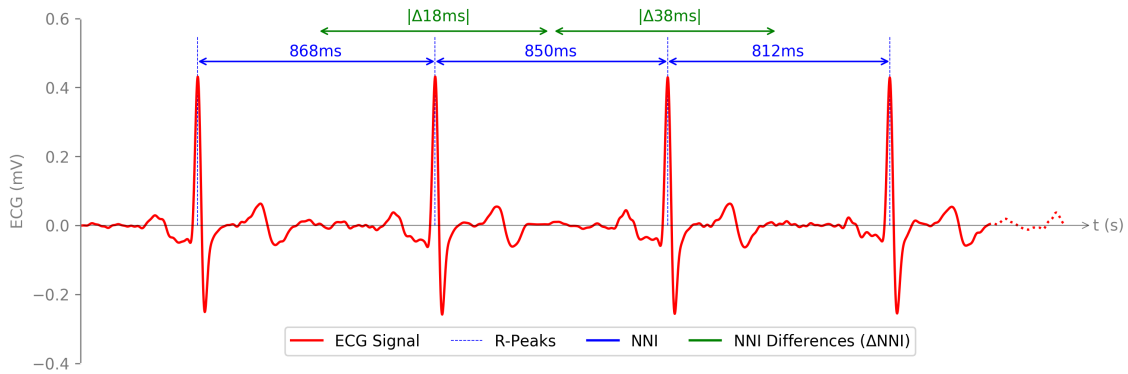


Figure 5: Relationship between the ECG signal (red) R-peaks, NN intervals (blue), and NN interval differences (green). From a signal with n R-peaks and $n - 1$ NN intervals, $n - 2$ NN interval differences can be derived.

The NN parameters are usually not very meaningful parameters when used alone, especially when based on a single NNI. For instance, the maximum or minimum value of an NNI series alone does not provide enough significant information to make any proper HRV assessment of a data set. For this reason, these parameters should be used in relation with other time domain parameters presented on the following pages.

2.3.1.2 Heart Rate

Heart Rate (HR) is the standard measure to indicate the number of occurring heartbeats per minute (bpm) [2]. It is computed according to Equation 3 and is a user-friendly measure of general monitoring of cardiac activity and the heart's response to physical or emotional stress [1, 86].

$$HR = \frac{60000[ms]}{NNI[ms]} = \frac{60[s]}{NNI[s]} \quad (3)$$

with: HR : Heart rate in [bpm]
 NNI : NN interval in [ms] or [s]

The HR is known to adjust to physical and emotional conditions of the subject as observed by Tolg et al. [91] in research conducted on subjects experiencing performance anxiety in virtual reality environments, where increasing HR have been observed with the increase of emotional stress. The most commonly used parameters which are directly related to HR and used in HRVA are minimum HR, maximum HR, mean HR and difference between maximum and minimum HR [1, 82].

2.3.1.3 Tachogram

HR series are often plotted with the NNI series in a plot named Tachogram, where both series are plotted against their temporal occurrence. This plot provides a clearer visual representation of both datasets, thus being a useful tool for a primary visual inspection of HRV. Figure 6 (bottom) shows an example Tachogram with NNI and HR series derived from a ECG signal (top).

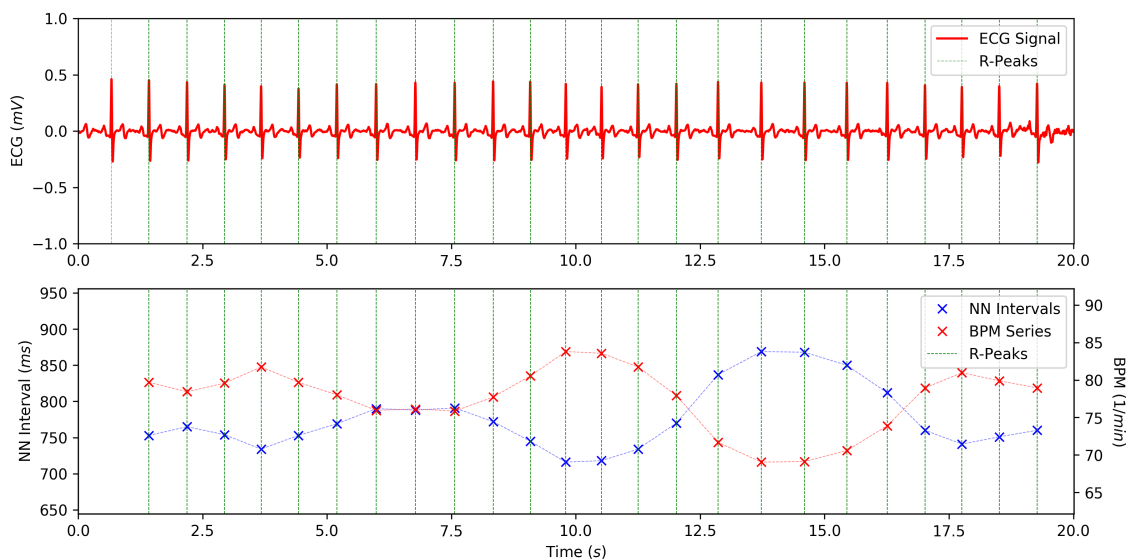


Figure 6: Sample ECG signal (top) and Tachogram with the derived NNI and HR series (bottom).

2.3.1.4 SDNN

Standard Deviation (SD) is generally a measure to quantify the amount of dispersion in a dataset. The Standard Deviation of Successive NN Intervals (SDNN) quantifies this variation in NNI durations around the mean value of the NNI, and is computed

according to Equation 4.

$$SDNN = \sqrt{\frac{1}{n-1} \cdot \sum_{j=1}^n (NN_j - \overline{NN})^2} \quad (4)$$

with: $SDNN$: Standard deviation of the NNI series in [ms]
 n : Number of NN intervals in [ms]
 NN_j : NN interval i in [ms]
 \overline{NN} : Mean of NN intervals in [ms]

SDNN serves as a measure for the activity of the entire autonomic nervous system (the aggregated ANS and PNS) which is ultimately reflected in the NNI. ECG datasets with low HRV, i.e. low variation in NNI durations, have been observed in subjects with existing medical conditions (e.g., after myocardial infarction, diabetes mellitus) and lead to small SDNN results, while higher SDNN results were observed in datasets acquired from healthy subjects with greater NNI variations [81, 93]. As for other HRV parameters, it is recommended to determine SDNN values on ECG datasets with conventional durations of 5 minutes or 24 hours, although it has to be noted that the duration itself influences the result given that the total variance does naturally increase with the duration of the recording session [86]. For this reason, it is recommended to split long-term acquisitions into shorter segments (with 5 minutes of duration), and to individually analyse each segment to counteract against the lack of robustness of the SD against outliers or non-stationary data series, as it is done with the Standard Deviation of Successive NN Intervals Index (SDNNI) parameter.

Salahuddin et al. [75] have studied the possibilities and reliability of shorter recording periods as an alternative to conventional 5 minutes short-term recordings, to meet the increasing demand of mobile applications for clinical monitoring where data acquisitions may only be possible during short intervals. Proposals of 60 seconds recordings have shown to be insufficient for reliable measurements of SDNN values, while recordings of 240 seconds in duration have shown to be good surrogates for SDNN values derived from 5 minute recordings [4].

2.3.1.5 SDNN Index

The Standard Deviation of Successive NN Intervals Index (SDNNI) is an analysis parameter based on SDNN taken into account the considerations of long-term ECG acquisitions, and is most commonly applied to acquisitions of 24 hours in duration.

For this parameter, the 24 hour ECG recording is split into segments of 5 minutes in duration resulting in a total of 288 segments (24 hours · 60 minutes = 1440 minutes → 1440 minutes / 5 minutes = 288 (segments)). It is the mean of all SDNN values computed from all 288 segments of the long-term ECG acquisition, and it is computed through the steps shown in Figure 7, with the mean value being computed according to Equation 5 [86].

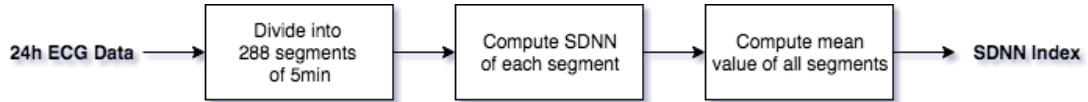


Figure 7: Block diagram of the SDNN Index computation.

$$SDNNI = \frac{1}{N} \cdot \sum_{j=1}^n SDNN_j \quad (5)$$

with: $SDNNI$: SDNN index in [ms]
 n : Number of computed SDNN values from 5 minute segments
 $SDNN_j$: SDNN of the 5 minute segment j in [ms]

It is assumed that the SDNNI is primarily a measure of the ANS impact on the HRV as it does also correlate with the Low Frequency (LF) spectral components (which represent ANS activity) in 24 hour datasets [92].

2.3.1.6 SDANN

The SDANN is the SD of the mean of NNI within individual 5 minute segments in long-term ECG acquisitions [1]. Similar to the process followed in the computation of the SDNNI, the NNI series of a 24 hour ECG acquisition is split into 288 segments of 5min in duration, followed by the computation of the mean value of each segment. Afterwards, the SD of all mean values is computed. A visualization of the individual steps of this computation is shown in Figure 8 and shown in Equation 6.

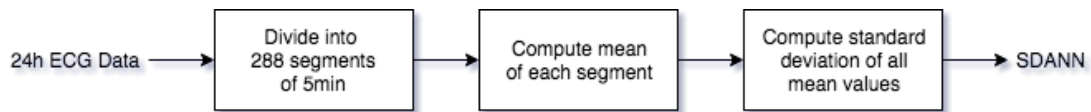


Figure 8: Block diagram of the SDANN computation.

$$SDANN = \sqrt{\frac{1}{n-1} \cdot \sum_{j=1}^n (\overline{NNI}_j - \overline{\overline{NNI}})^2} \quad (6)$$

with: $SDANN$: Standard deviation of the mean of NNI in all 5 minute segments in [ms]
 n : Number of 5 minute segments
 \overline{NNI}_j : Mean value of the 5 minute segment j in [ms]
 $\overline{\overline{NNI}}$: Mean value of the means of the 5 minute segments in [ms]

2.3.1.7 RMSSD

The RMSSD is the root mean of the sum of the squares differences between successive NN intervals. This parameter is determined by first computing all differences between adjacent NN intervals (see Equation 2), which are then used to compute the RMSSD parameter according to Equation 7 [1, 86].

$$RMSSD = \sqrt{\frac{1}{n-1} \cdot \sum_{j=1}^n \Delta NNI_j^2} \quad (7)$$

with: $RMSSD$: Root mean square of successive NNI differences in [ms]
 n : Number of computed NNI differences
 ΔNNI_i : NNI differences in [ms]

RMSSD measures the short-term variations of successive NNI and provides information about the activity of the PNS. It is often used as an indicator for rehabilitation, fitness, and health and is the primary parameter in the time domain for estimations of vagally mediated changes in HRV [55]. High RMSSD indicates the ability of rapid adaptations of the HR, while low RMSSD indicates a low ability of HR adjustments, generally caused by physical, mental stress, or sickness with, the latter cause being observed (e.g.) in patients with diabetes mellitus where significantly lower RMSSD results could be derived when compared to healthy subjects [81].

Similarly to the SDNN, this parameter is generally applied on short-term ECG recordings, although Thong et al. [90] have investigated the applicability and reliability of this parameter in ultra short-term recordings of 10 seconds, showing a high correlation between the obtained values and values from 5 minute recordings. This correlation opens new possibilities of RMSSD-based monitoring applications in sports research and medical monitoring, by using only small sample sizes which

provides an almost immediate biofeedback parameter. This is a characteristic which can be useful for the development of wearable systems in both medical and consumer applications.

2.3.1.8 SDSD

The SDSD is the SD of the differences of successive NNI and shares a great similarity with the RMSSD parameter. However, the major difference between the SDSD and the RMSSD is its dependence on the stationary characteristics of the NNI series. This parameter is computed according to Equation 8.

$$SDSD = \sqrt{\frac{1}{n-1} \cdot \sum_{j=1}^n (\Delta NNI_j - \overline{\Delta NNI})^2} \quad (8)$$

with: $SDSD$: Standard deviation of the successive NNI differences in [ms]
 n : Number of NNI differences
 ΔNNI_j : NNI difference j in [ms]
 $\overline{\Delta NNI}$: Mean of NNI differences in [ms]

The SDSD provides the same results as the RMSSD parameter when applied on stationary time series, i.e. time series with no base level shifting of the signal or time-independent spectra, thus sharing a similar physiological context as the RMSSD [42, 65]. However, as the SD is not robust against outliers or non-stationary time series, an application of this parameter on such datasets will not provide proper results, reason for which the usage of the RMSSD parameter is recommended. Both the RMSSD and SDSD share a similar physiological context and clinical application.

2.3.1.9 NNx and pNNx

NNx parameters are based on the total count of differences between successive NNI which are greater than a specific threshold x in milliseconds. Ewing et al. [26] were the first to propose and explore this parameter in long-term recordings, with a threshold set to 50 milliseconds, establishing it as a standard threshold used in conventional HRVA. This proposal urged from the observation and comparison of ECG datasets from healthy subjects and subjects with diabetes mellitus, where healthy subjects showed significantly higher random occurrences of greater differences between successive NNI. The pNNx parameter has been added as a second parameter

computed as the division of the NNx value by the total number of ΔNNI [86]. This parameter facilitates the interpretation of the NNx parameters, as it provides a ratio rather than an integer number, which is harder to interpret without if the total number of ΔNNI is unknown.

$$pNNx = \frac{NNx}{n} \quad (9)$$

with: NNx : Number of $\Delta\text{NNI} > x$ in [ms]
 n : Number of ΔNNI differences

Newer studies have re-examined the 50 millisecond threshold to assess the applicability of threshold variations ranging from 1 to 100 milliseconds. These studies show that thresholds lower than 50 milliseconds were able to create significant distinctions between healthy and pathological patients, where NN50 parameters failed to do so. However, these thresholds have been criticized for depending too much on the methods of statistical comparison, when compared with the 50ms threshold, and worked only in explicit comparisons rather than in general applications [57, 80].

2.3.2 Geometrical Parameters

The geometrical parameters are derived from the histograms of NNI series, with a standard bin size set to 7.8125 milliseconds due to the minimum recommended sampling frequency of 128Hz as set forth by the HRV guidelines ($1/128\text{Hz} = 0.0078125\text{s} = 7.8125\text{ms}$) [86]. Although ECG acquisitions are nowadays usually performed with a diversity of sampling frequencies (e.g. 1000Hz), this bin size has been considered as the reference bin size for comparison purposes. Any deviation is usually noted in the HRV results [35, 86]. An example of a NNI histogram is visualized in Figure 9, with detailed explanations about the most commonly used parameters being presented throughout the following pages.

2.3.2.1 Triangular Index

The Triangular Index (TI) is the most basic parameter that can be derived from the NNI histogram. This parameter is the division between the integral over the entire histogram, i.e. the total number of NNI of the distribution, and the distribution's maximum $D[X]$ with X being the bin containing the distribution's maximum. The

formula for the computation of this parameter is presented in Equation 10, with an example of a distribution with a given maximum $D(X)$ visualized in Figure 9 [86].

$$TriangularIndex = \frac{n}{D[X]} \quad (10)$$

with: n : Total number of NNIs
 $D[X]$: Distribution's maximum
 X : Bin at which the distribution's maximum occurs in [ms]

2.3.2.2 Triangular Interpolation of NNI Histogram (TINN)

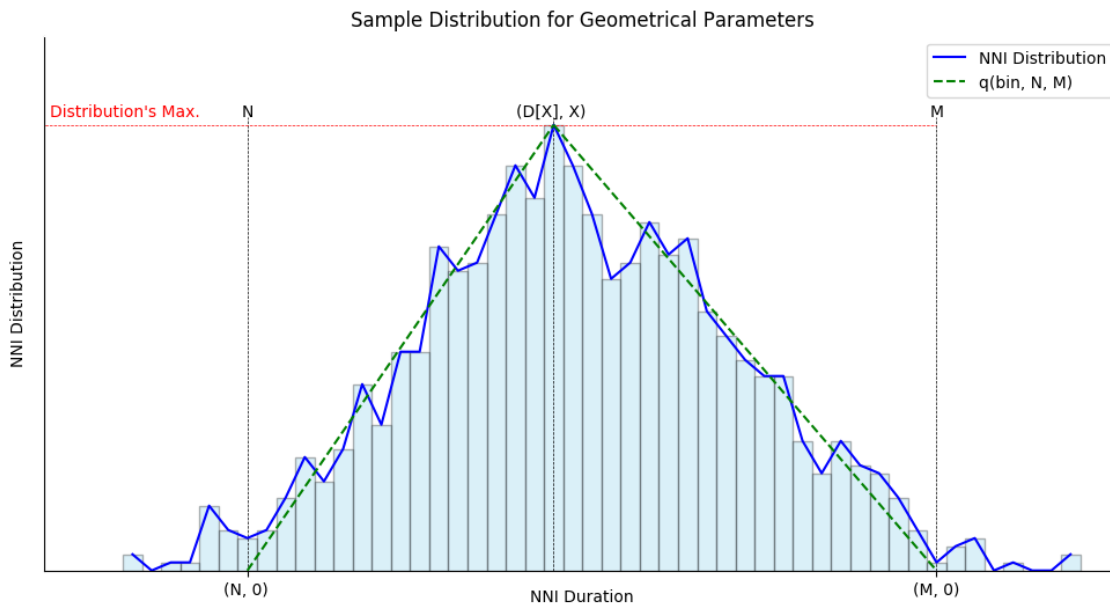


Figure 9: Histogram of an artificial NNI series to show the geometrical parameters with the distributions maximum $D[X]$ and the resulting function of the triangular interpolation in dependence of the N and M parameters.

The TINN is defined by the baseline width of the histogram, based on the approach of finding the triangular interpolation with the best fit to the NNI histogram. For this approach, a triangle is fit in the NNI histogram with its corners being set at the points $(N, 0)$, $(X, D(X))$, and $(M, 0)$ where $N < X$ and $M > X$. The linear functions between the points $(N, 0)$ and $(X, D(X))$ and the points $(X, D(X))$ and $(M, 0)$ construct the triangular function $q(\text{bin}, N, M)$ which, for values outside the $[N, M]$ interval, is 0. N and M are then set by finding the $q(t, N, M)$ with the lowest error, i.e. deviation, from the original distribution function using the least-squares method as presented in Equation 11. Afterwards, the computation of the baseline

width of the histogram (TINN) can be computed by Equation 12 [86, 96].

$$\Delta = \operatorname{argmin}\left\{\sum_{b=b_{min}}^{b_{max}} (D - q(\operatorname{bin}, N, M))^2\right\} \quad (11)$$

$$TINN = M - N \quad (12)$$

with: Δ : Triangular interp. with the best fit to the NNI histogram
 b_{min} : Lowest bin containing non-zero value of the distribution in [ms]
 b_{max} : Highest bin containing non-zero value of the distribution in [ms]
 D : NNI distribution
 $q(\operatorname{bin}, N, M)$: Triangular interpolation function
 N : Left corner of the interp. triangle with the best fit to the NNI histogram in [ms]
 M : Right corner of the interp. triangle with the best fit to the NNI histogram in [ms]
 $TINN$: Baseline width of the histogram in [ms]

An example of a triangular interpolation with the defined points in dependence of N , M , and $D(X)$ (and X) is presented in the histogram in Figure 9. The distribution function D (blue) is defined by the histogram of the NNI series with $q(\operatorname{bin}, N, M)$ (green) being defined by the N and M values that generate a triangular interpolation with the lowest error compared to D .

2.3.2.3 Physiological Context of Geometrical Parameters

Given that the NNI histogram is highly dependent on the standard deviation of the NNI series, i.e. the SDNN which has been observed to be significantly decreased in subjects with existing medical conditions (e.g. diabetes mellitus), its form and the derived geometrical parameters are known to be a measure of overall HRV [81, 86]. Similarly to the case of the SDNN parameter, both the TI and TINN have been observed to be smaller in patients with existing medical conditions where reduced HRV could be found. A lower SDNN leads to a decrease in overall histogram width while causing an increase in the histogram's values around its mean NNI (smaller SD \rightarrow compressed histogram \rightarrow smaller histogram baseline and increased D) [81, 84, 94].

2.3.3 Frequency Domain Parameters

Frequency analysis is generally a method to determine periodicity in a given signal or dataset. It has been introduced in the 1970's to the field of HRV research as a new and interesting measure to reveal periodicities in NNI. With this method, the estimation of a PSD in the form of a periodogram (a commonly used tool to estimate the power in a signal in dependence of its frequencies) is used to identify dominant, high powered frequencies and to link them to physiological mechanisms [8, 86].

The general procedure of frequency domain analysis starts with the computation of a PSD, followed by splitting the PSD into Frequency Band (FB)s and deriving a series of overall parameters related with specific frequency bands. The guidelines, among other HRV related publications, recommend different methods for the computation of the PSD, which can be classified as (a) non-parametric and (b) parametric methods. Although both provide meaningful results, each method has its advantages over the other: (a) non-parametric methods require no prior knowledge about the signal model or its form. Additionally, these methods can be efficiently computed with simple algorithms at high processing speeds (e.g., FFT and derived metrics, Lomb-Scargle periodogram). (b) Parametric methods, on the other hand, provide smoother spectral components and, accurate PSD estimations even on small sample numbers [86]. However, parametric methods are highly dependent on the selection of suitable parameters to ensure the accuracy of the PSD estimation, as such methods assume that the given data fits a specific signal model for which it is necessary to define a suitable estimation model order. This can constitute a practical issue, as the ideal model order varies in dependence of the ECG duration and no *one-size-fits-all* order is available [15].

The frequency domain parameters are derived from the methods presented on the following pages. Additionally, this section aims to present some example of the relationship between physiological mechanisms and FB, the most commonly used frequency analysis methods, and the computation of the frequency parameters as recommended by the HRV guidelines shown in Table 2 [86]. It has to be noted that, due to the high complexity of the different PSD estimation methods, only their HRV specific implementation will be presented rather than going into detail into their mathematical background.

Table 2: Overview of frequency domain parameters [86].

Frequency Domain Parameters - FFT Based		
Parameter	Unit	Description
Total Power	ms^2	Power over all FC
Absolute Power	ms^2	Power of each FC
Relative Power	%	Relative power of each FC
Log(Power)	-	Natural logarithm of FC's absolute power
Normalized Power	-	Normalized powers of each LF & HF components
Peak Frequency	Hz	Frequency where maximum power of the FC occurs
LF/HF Ratio	%	Ratio between the LF & HF components

2.3.3.1 Frequency Components

In HRVA, the computed PSD is segmented into three FB, when analyzing short-term recordings of usually 5 minutes in duration, or four frequency bands when analyzing long-term recordings of usually 24 hours, in duration as presented in Table 3. Segmenting the PSD into specific frequency bands provides a method to determine dominant frequencies in a NNI series, which are ultimately linked to autonomic responses of a subject's body.

Table 3: Frequency bands for ECG acquisitions of different durations, as recommended by the HRV guidelines [86].

Component	Short-term (5 minutes)		Long-term (24 hours)	
	Min [Hz]	Max [Hz]	Min [Hz]	Max [Hz]
ULF	-	-	0.000	0.003
VLF	0.003	0.040	0.003	0.040
LF	0.040	0.150	0.040	0.150
HF	0.150	0.400	0.150	0.400

The Ultra Low Frequency (ULF) band is the only FB applied on long-term recordings. For the time being, it is still uncertain which physiological rhythms have the greatest influence in this frequency band, although it is highly hypothesized that circadian rhythms (periodic biological rhythms with 24 hour periods, e.g. sleep/wake cycle) may be the most dominant [82, 83, 99]. By this hypothesis, applying this FB on short-term recordings would not provide much added value for analytical purposes.

The Very Low Frequency (VLF) band ranges between 0.003Hz (or in some cases 0Hz if the ULF is not computed) and 0.04Hz [86]. Research has shown that thermoregulatory mechanisms influence the energy in this FB [55]. Additionally, it has been observed that lower power within this FB may be associated with pathological

causes (e.g. inflammation) [45].

The LF band ranges between 0.04Hz and 0.15Hz, and reflects primarily baroreceptor activity (blood pressure regulation) in resting positions. However, it has to be noted that low breathing periods (approx. 7 to 10 seconds) can also influence the power in this frequency band [82].

The last FB, the High Frequency (HF) band, ranges between 0.15Hz and 0.4Hz, and is essentially known to reflect respiratory influences in the HRV and, therefore, reflecting vagal inhibition (inhaling \rightarrow increases HR) and activation (exhaling \rightarrow decreases HR), which are the fundamental mechanisms of respiratory regulation. The respiratory influence causing these fluctuations in HR is known as *Respiratory Sinus Arrhythmia* [55, 82, 103].

2.3.3.2 Welch's Periodogram

The FFT based PSD estimation is a non-parametric method, i.e. a method that does not make any assumption on the model which generates the NNI data [85]. This method requires some pre-processing of the NNI time series before it can be properly computed. First, the NNI series has an unknown and uneven sampling rate, reason for which it needs to be resampled to provide a dataset with evenly spaced intervals. This step is required, as the FFT is designed to provide information about the periodicity of an evenly sampled signal, which results would be distorted when applied on unevenly sampled time series. The resampling is usually achieved by application of a cubic-spline interpolation to the NNI series and by deriving a new time series with a 4Hz sampling frequency from the interpolated signal [86]. However, it has to be noted that this resampling process can have a low-pass filtering effect on the original NNI series [43, 86]. Afterwards, every sample in this dataset is detrended, i.e. subtracted by the dataset's mean value, to remove the DC-offset, which would be reflected in the PSD as a high energy component around the 0Hz frequency [41]. In HRVA applications, the recommended method for the PSD estimation is the Welch's Method [86, 100]. An example of a FFT based PSD using the Welch's method can be seen in Figure 10.

2.3.3.3 Autoregression Periodogram

The Autoregression (AR) based PSD estimation is a parametric method. This estimation method assumes that each value of a given time series does linearly

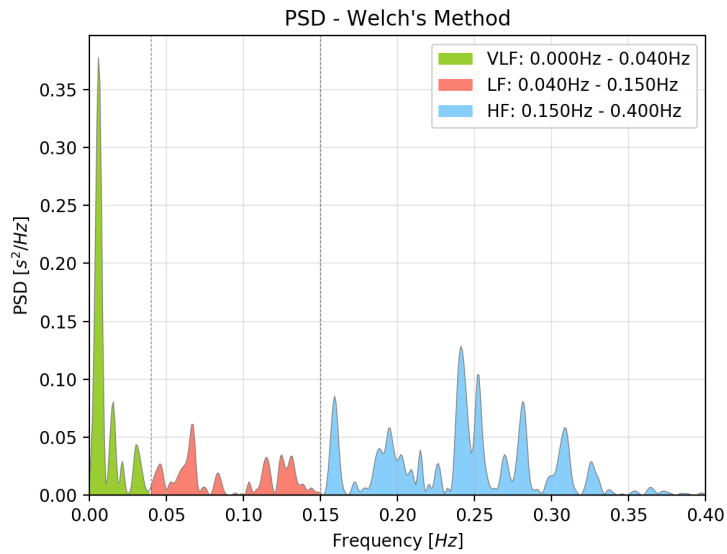


Figure 10: Sample PSD computed using the Welch's method (NNI series segment extracted from: [31]).

depend on one or multiple previous values. Due to this assumption, it differs from the FFT-based method as the latter method does not assume the time series to be generated by a specific signal model [15, 40]. As previously stated, this method has the advantages of providing smoother spectral components and providing accurate PSD estimations. However, due to the signal model assumptions, the selection of a suitable model order, i.e. the number of previous samples being used for the estimation of future samples, may cause a practical problem as investigated by Kuusela et al. [42]. Choosing too low model orders can smooth and vanish PSD peaks, while too high model orders can increase the influence of noise present the NNI series. Reason for which the selection of suitable model orders has been thoroughly investigated, with recommended model orders ranging from 9 to 25 with 16 being the recommended minimum order for short NNI segments [10, 58]. An example of an Autoregressive PSD plot is shown in Figure 11.

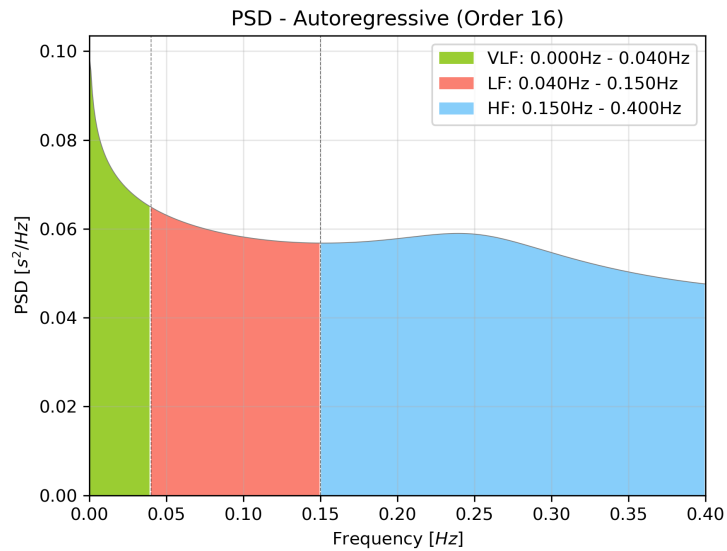


Figure 11: Sample PSD computed using the Autoregressive (Yule-Walker) method (NNI series segment extracted from: [31]).

2.3.3.4 Lomb-Scargle Periodogram

The Lomb-Scargle Periodogram (LSP) is a method for PSD estimation in non-uniformly sampled datasets, and has originated from such problems in astrophysics. Although the cause for non-uniformity in this field is of entirely different nature compared to the causes for non-uniformity in NNI series (e.g. observational restrictions due to bad weather conditions vs. spontaneous HR regulation), the obstacles for PSD estimations for unevenly sampled datasets remain the same, and require resampling and interpolation methods to minimize distortions in the frequency domain, when applying traditional estimation methods (e.g. FFT) [44, 49, 61, 77].

The LSP uses the least-squares method to fit sinusoids to a dataset to compute a FFT-like PSD. This method requires fewer additional steps than traditional FFT-based PSD, considering that it does not require any pre-processing of the dataset to conduct a resampling or interpolation in order to deal with the dataset's non-uniformity. However, this method does have its limitations, given that it requires a warm-up phase, during which a minimum amount of samples or acquisition duration for which the lower limit has yet to be further investigated. [19, 42, 61]. An example of a possible HRV based LSP with the segmentation into the different FB is shown in Figure 12.

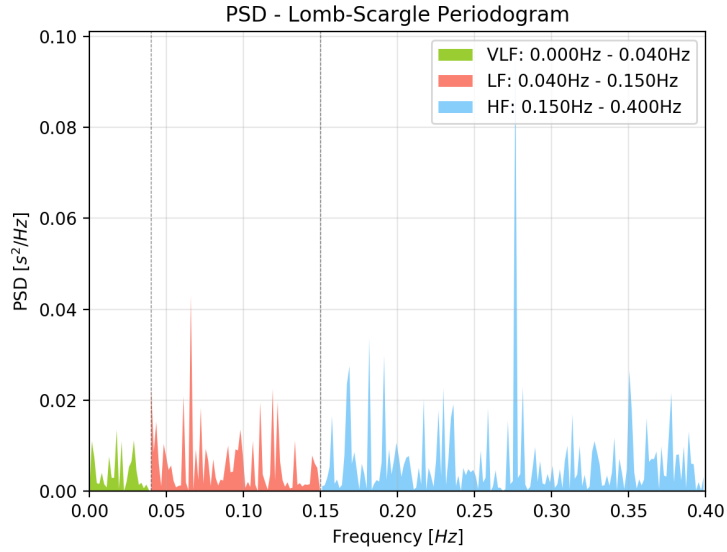


Figure 12: Sample PSD computed using the Lomb-Scargle periodogram (NNI series segment extracted from: [31]).

2.3.3.5 Frequency Parameters

The absolute power of each FB is computed by integration of the PSD over the component's frequency band. As the signal used for HRVA is of discrete nature, the sum of the power samples over the FB multiplied by the frequency resolution is equal to the sum over the same interval (Equation 13).

$$P_{abs,z} = \Delta f \cdot \sum_{f=f_{min}}^{f_{max}} S \quad (13)$$

with: z : Frequency band (ULF, VLF, LF or HF) [n.u.]
 $P_{abs,z}$: Absolute power of the frequency band z in [ms^2]
 Δf : Frequency resolution in [Hz]
 $f_{min,z}$: Minimum frequency of the frequency band z in [Hz]
 $f_{max,z}$: Maximum frequency of the frequency band z in [Hz]
 S : PSD function (unit: [ms^2])

The total power can either be computed by integrating the signal over the interval between the lowest frequency band limit (lower limit of the ULF or VLF at 0Hz, depending on which is being used) and the highest frequency of the HF frequency band (at 0.4Hz). The total power is computed as the sum of the absolute powers of all frequency bands (Equation 15).

$$TP = \Delta f \cdot \sum_{f_{min}}^{f_{max}} S \quad (14)$$

with: TP : Total power in [ms^2]
 Δf : Frequency resolution in [Hz]
 f_{min} : Minimum frequency of the lowest frequency band (ULF or VLF) in [Hz]
 f_{max} : Maximum frequency of the HF frequency band in [Hz]
 S : PSD function (unit: [ms^2])

$$TP = P_{ULF} + P_{VLF} + P_{LF} + P_{HF} \quad (15)$$

with: TP : Total power in [ms^2]
 P_{ULF} : Absolute power of the ULF band in [ms^2]
 P_{VLF} : Absolute power of the VLF in [ms^2]
 P_{LF} : Absolute power of the LF in [ms^2]
 P_{HF} : Absolute power of the HF in [ms^2]

Relative powers are the ratio between the absolute power of the individual frequency band and the total power of the PSD within the specified minimum and maximum frequency interval, and are therefore expressed as percent (Equation 16). Newer suggested implementations of the frequency domain parameters also provide the logarithmic value of the absolute powers (Equation 17) [66].

$$P_{rel,z} = \frac{P_{abs,z}}{TP} \cdot 100\% \quad (16)$$

with: z : Frequency band (ULF, VLF, LF or HF) [n.u.]
 $P_{rel,z}$: Relative power of FB z in [%]
 $P_{abs,z}$: Absolute power of the FB z in [ms^2]
 TP : Total power in [ms^2]

$$P_{log,z} = \log(P_{abs,z}) \quad (17)$$

with: z : Frequency band (ULF, VLF, LF or HF) [n.u.]
 $P_{log,z}$: Logarithmic power value of FB z [n.u.]
 $P_{abs,z}$: Absolute power of the FB z in [ms^2]

Normalized powers are usually only computed for the LF and HF frequency band and are computed as presented in Equations 18 and 19 [14].

$$P_{norm,LF} = \frac{P_{abs,LF}}{P_{abs,LF} + P_{abs,HF}} * 100 \quad (18)$$

$$P_{norm,HF} = \frac{P_{abs,HF}}{P_{abs,LF} + P_{abs,HF}} * 100 \quad (19)$$

with: $P_{norm,LF}$: Normalized power of the LF band [n.u.]
 $P_{norm,HF}$: Normalized power of the HF band [n.u.]
 $P_{abs,LF}$: Absolute power of the LF band in [ms^2]
 $P_{abs,HF}$: Absolute power of the HF band in [ms^2]

The LF/HF ratio can be computed as the ratio between the absolute powers in the LF and HF FB (Equation 20)[86].

$$\frac{LF}{HF} = \frac{P_{abs,LF}}{P_{abs,HF}} \quad (20)$$

with: LF/HF : LF/HF ratio [n.u.]
 $P_{abs,LF}$: Absolute power of the LF band in [ms^2]
 $P_{abs,HF}$: Absolute power of the HF band in [ms^2]

2.3.4 Nonlinear Parameters

The Nonlinear parameters are intended to enhance nonlinear characteristics and unpredictability of NNI series, which are caused by the different complex physiological dynamics of the human body that lead to HRV (e.g. SNS vs. PNS) [8, 82]. The HRV guidelines named multiple methods to derive a variety of parameters, but failed to establish a selection of reliable and meaningful nonlinear methods for clinical contexts, due to the lack of scientific evidence and technological advancements at the time [86].

Over the last recent years, a variety of mathematical methods for nonlinearity analysis have been applied in HRV research. However, many of them have not been found to be suitable for HRVA due to their complexity or lack of sufficient scientific backing, with only a few methods being commonly found in HRV research [8, 42]. Table 4 presents a selection of the most commonly found methods [42, 70, 82, 87, 102].

Table 4: Overview of nonlinear parameters.

Nonlinear Parameters		
Parameter	Unit	Description
Poincaré Plot	-	Scatter plot of NN intervals (NN_{i+1} against NN_i)
- SD1	<i>ms</i>	Standard deviation along the minor axis
- SD2	<i>ms</i>	Standard deviation along the major axis
- S	-	Area of the fitted ellipse (focus points SD1 & SD2)
- SD1/SD2	%	Ratio between SD1 & SD2
Sample Entropy	-	Sample entropy of the NNI series
DFA	beats	Detrended fluctuation analysis

2.3.4.1 Poincaré Plot

The *Poincaré plot* is a scatter plot where a given NNI_j is plotted against its successor NNI_{j+1} . It is a graphical tool for HRV analysis of a NNI dataset, which allows a rapid first judgment of a subject's health, as the shape of the scatter plot provides a visual representation of the overall HRV [87]. Woo et al. [102] have investigated the patterns of healthy subjects with advanced heart failure and observed torpedo or comet-like shaped Poincaré plots in datasets of healthy subjects while circular, chaotic or fan-like shape, with multiple clusters of points, were observed in datasets of patients with the named heart conditions.

Figure 13 (a) demonstrates an example Poincaré plot of a NNI dataset with relatively

low HRV, as the majority of points is closely clustered around the dataset's mean point (at $NN_{i+1} = NN_i = \overline{NN}$, which in this case is 775.19ms). Figure 13(b) on the other hand demonstrates an example Poincaré plot of an NNI dataset with high HRV, with points being widely spread around the dataset's mean value along the plot's major axis (identity line with $y = x$), creating a comet-like shape.

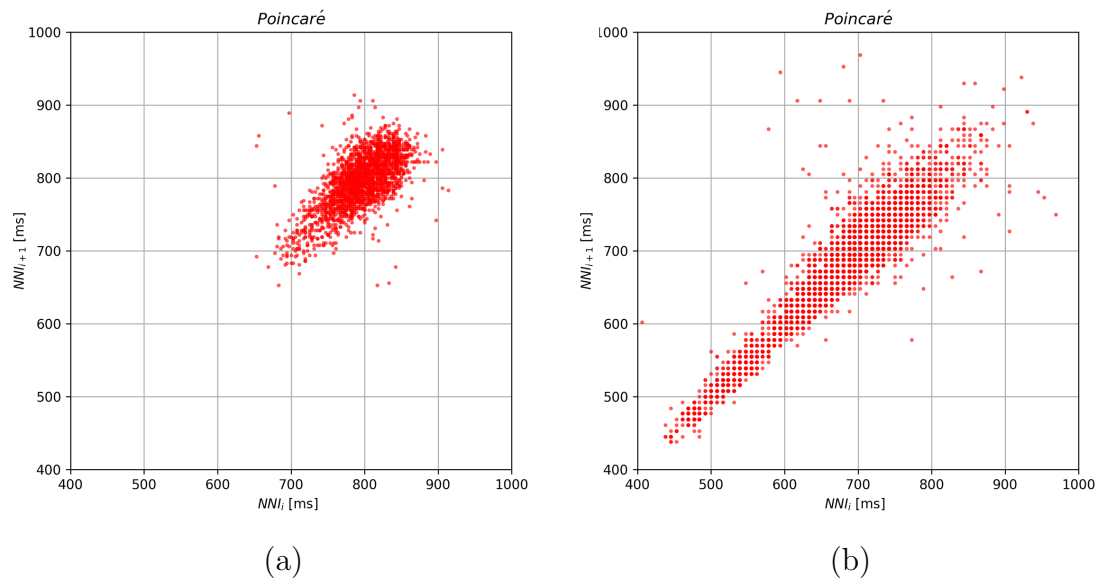


Figure 13: (a) Poincaré plot of an NNI dataset of a normal subject while in rest. The points of the plot are closely clustered around the dataset's mean value. (b) Poincaré plot of an NNI dataset with high HRV. The points of the plot are widely spread along the identity line and create a comet-like shape (dataset source: [31]).

Besides the plot itself, the Poincaré method provides parameters that can be used for a more detailed analysis of the NNI based scatter plot: SD1, SD2, ellipse area, and SD1/SD2 ratio. SD1, the standard deviation of the points perpendicular to the major axis along the minor axis, is a measure of instant beat-to-beat variability, i.e. short-term variability, and is correlated with the HF powers of the NNI series [12, 33, 87]. This parameter can either be computed based on the SDDS time domain parameter - the standard deviation of successive differences (see Chapter 2.3.1.8) - according to Equation 21, or on a vector based computation as suggested by Tayel et al. [87] according to Equation 22 - as the variance of the difference of the $\overrightarrow{NN_{i+1}}$ and $\overrightarrow{NN_i}$ vectors [12]. The SD1 parameter is usually visualized in the plot as a vector with its origin at the mean of the scatter plot and the length of SD1 along the axis perpendicular to the major axis, as visualized in Figure 14 with the SD1 vector (green arrow) and the minor axis (green dotted line).

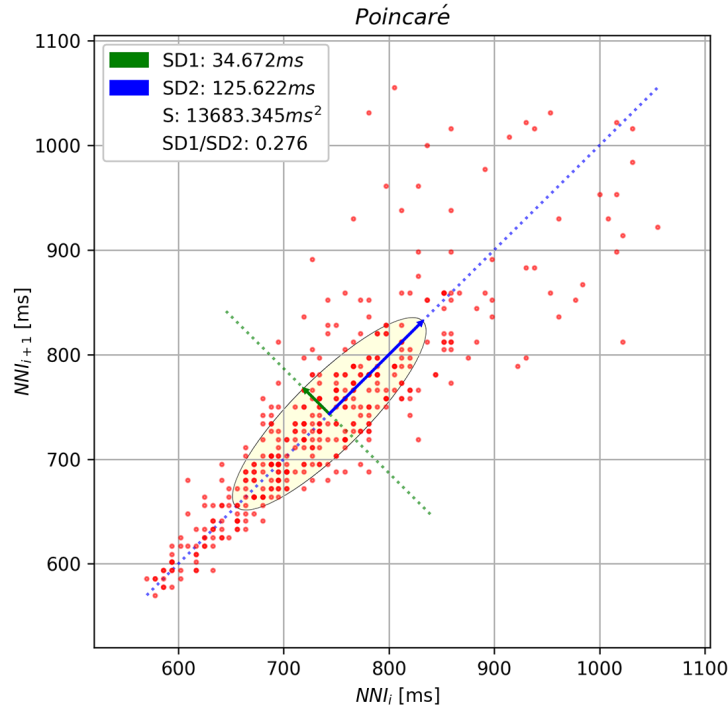


Figure 14: Poincaré plot depicting of the SD1 parameter (green arrow) along the axis perpendicular to the identity line (green dotted), the SD2 parameter (blue arrow) along the identity line (blue dotted) and the ellipse with its center at the mean of the scatter plot.

$$SD1 = \sqrt{\frac{1}{2} \cdot S D S D^2} \quad (21)$$

$$SD1 = \frac{1}{\sqrt{2}} \cdot \sigma(\overrightarrow{NN_{i+1}} - \overrightarrow{NN_i}) \quad (22)$$

with: $S D S D$: Standard deviation of successive differences in [ms]
 $\overrightarrow{NN_{i+1}}$: Vector of successive NN intervals in [ms]
 $\overrightarrow{NN_i}$: Vector of current NN intervals in [ms]

SD2, on the other hand, is the standard deviation of the points along identity line, and is a measure of both short and long-term variability that correlates with the LF power [82]. This parameter can be either computed based on the SDNN - the standard deviation of successive NNI (see Chapter 2.3.1.4) - and the S D S D time domain parameters according to equation 23, or also on a vector based computation, as suggested by Tayel et al. [87] in Equation 24 - as the variance of the sum of the $\overrightarrow{NN_{i+1}}$ and $\overrightarrow{NN_i}$ vectors [12]. The SD2 parameter is commonly depicted as a vector with its origin at the mean of the scatter plot and the length of SD2 along the identity

line, as presented in Figure 14, with the SD2 vector (blue arrow) and the identity line (blue dotted line). Additionally, the SD1 and SD2 ratio, which measures both the randomness of an NNI series and, with a sufficiently long monitoring duration, the balance between sympathetic and parasympathetic activity, is another important parameter of the Poincaré parameter family [47, 82].

$$SD2 = \sqrt{2 \cdot SDNN^2 - \frac{1}{2} \cdot SDSD^2} \quad (23)$$

$$SD2 = \frac{1}{\sqrt{2}} \cdot \sigma(\overrightarrow{NN_{i+1}} + \overrightarrow{NN_i}) \quad (24)$$

with: $SDNN$: Standard deviation of successive NN intervals in [ms]
 $SDSD$: Standard deviation of successive differences in [ms]
 $\overrightarrow{NN_{i+1}}$: Vector of successive NN intervals in [ms]
 $\overrightarrow{NN_i}$: Vector of current NN intervals in [ms]

Another commonly adapted method used in Poincaré based analysis is the fitting of an ellipse in the plot, with dependence of the \overline{NN} , SD1 and SD2 parameters, providing another visual highlight of such parameters. For this, the center of the ellipse is placed at the mean of the plot (i.e. $NN_{i+1} = NN_i = \overline{NN}$, in this case with 775.19ms) with the ellipse's focus points being the ends of the SD1 and SD2 vectors, as shown in Figure 14 (yellow ellipse). Tayel et al. [87] have shown that the area S (in in [ms²]) of the ellipse is greater in healthy subjects due to the overall higher HRV and very small for critical cases of subjects with existing heart conditions, reason for which the area of the ellipse can also be computed with accordance to Equation 25 [87].

$$S = \pi \cdot SD1 \cdot SD2 \quad (25)$$

2.3.5 Sample Entropy (SampEn)

The SamPen is a measure of complexity or irregularity of a NNI series. It is a vector-based computation, which is an improved version of the regular Approximate Entropy algorithm as it can be applied on small sample sizes [42, 82].

The SamPen function ($SamPen$ or sometimes found as $SampEn(m, r, N)$), is parametric function that depends on the embedding dimension m , the tolerance r , and

the number of NNI data points. The embedding dimension m defines the length of the vector $U_m(i)$ that is created from the original NNI series as presented in Figure 26.

$$U_m(i) = (NNI_i, NNI_{i+1}, \dots, NNI_{i+m-1}) \quad (26)$$

$$i = 1, 2, 3, \dots, n - m + 1$$

with: $U_m(i)$: Template vector
 n : Number of NNI
 m : Embedding dimension

Next, the distance function $d(U_m(i), U_m(j))$ is computed using the Chebyshev or Euclidean distance, to determine the maximum distance between the vector pairs $U_m(i)$ and $U_m(j)$ (with $i \neq j$) constructed from the template vector resulting in Equation 26.

$$d(U_m(i), U_m(j)) = \max\{|NNI_{i+n} - NNI_{j+n}|\} \quad (27)$$

From these functions, the parameters A and B are computed as the number of vector pairs that are smaller than the defined tolerance r , i.e.

$$A = \text{count}\{d(U_{m+1}(i), U_{m+1}(j)) < r\} \quad (28)$$

of the length $m + 1$ and

$$B = \text{count}\{d(U_m(i), U_m(j)) < r\} \quad (29)$$

of the length m . Finally, the SamPen is the negative logarithm of the ratio between A and B .

$$\text{SamPen} = -\log\left(\frac{A}{B}\right) \quad (30)$$

In HRVA, the recommended default values for m and r are set to 2 and $2SDNN$ [73]. Statistically, the SamPen is the negative logarithm of the probabilities of both the template vector with the length m and the template vector with the length

$m + 1$ having distances $< r$ within the tolerance limits (i.e. the probability that the additional data point $m + 1$ does not vary more than the tolerance $r \rightarrow$ low HRV).

Physiologically, low HRV is present in the NNI series due to low fluctuations in successive NNI. In this case, the predictability of the a vector of $m + 1$ in length and m are similar, i.e. the $m + 1$ data point does not exceed the specified tolerance levels, thus resulting in low SamPen values (e.g. $A = B = 1 \Rightarrow SampEn = -\log(1/1) = -\log(1) = 0$). High HRV on the other hand least to a lower predictability of a given additional $m + 1$ point due to the greater fluctuation between successive NNI durations, i.e. more $m + 1$ data points reach outside the specified tolerance levels. This decrease of $m + 1$ predictability results in smaller A values, thus decreasing the overall SamPen (e.g. $A = 0.1, B = 1 \Rightarrow SampEn = -\log(0.1/1) = 1$).

2.3.6 Detrended Fluctuation Analysis (DFA)

The DFA is an index that expresses the correlation within the NNI series in long-term acquisitions [65, 82]. This method has been introduced by Peng et al. and is computed as follows [70]:

First, the sum of the NNI series is computed and detrended by the series' mean.

$$Y[k] = \sum_{i=1}^n (NNI_i - \overline{NNI}) \quad (31)$$

with: $Y[k]$: Sum of the NNI series in [ms]
 n : Number of NNI
 \overline{NNI} : Mean of the NNI series in [ms]

Next, $Y[k]$ is segmented into individual segments of length m , for which a line is fitted to each of the segments using the least-squares, and the removal of any local trends. The fluctuation of each segment is then computed by the root-mean-square method of the detrended discrete integral function as presented in Equation 32.

$$F[m] = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y[i] - Y_m[i])^2} \quad (32)$$

The computation of $F[m]$ is repeated over different segment lengths m , which for HRVA applications commonly range between 4 and 16 beats NNI to determine short-term fluctuations and between 17 and 64 beats NNI for long-term fluctuations

[101]. The results are plotted in a double logarithmic plot containing the fluctuation results, with individual linear functions of slopes α_1 and α_2 fitted to the short and long-term fluctuations. An example plot of the DFA results is presented in Figure 15².

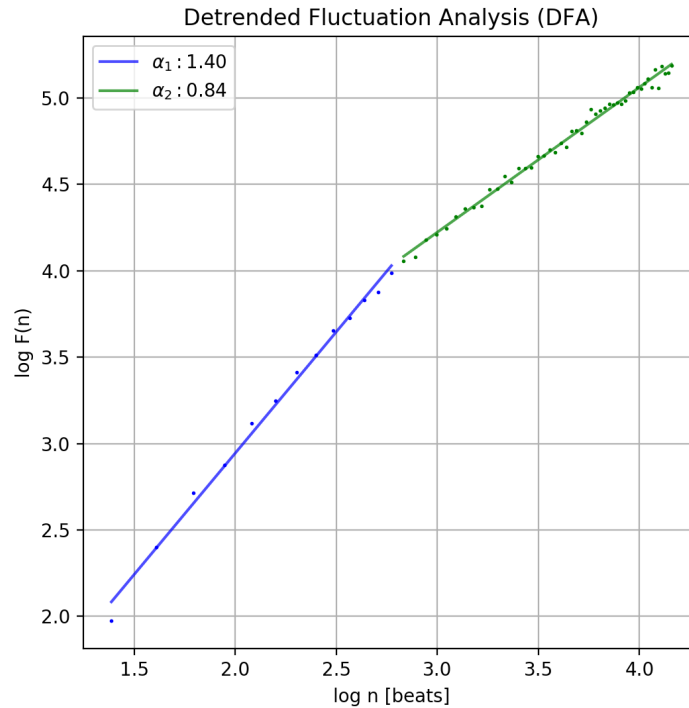


Figure 15: Resulting plot of a *Detrended Fluctuation Analysis* visualizing the short-term fluctuations (blue) and long-term fluctuations (green) [31].

²DFA computed based on a randomly selected 1 hour interval of NNI data from the MIT-BIH NSRDB database (signal 19090) using pyHRV

Chapter 3

Materials and Methods

3.1 Intended Workflow

The pyHRV toolbox developed in the scope of this work is intended to provide functions to compute HRV parameters and other useful functionalities and features often required, found, and used in HRV research applications to facilitate the use and increase the usability of this toolbox (e.g., export of HRV results, generating HRV reports). Although this toolbox's use is not restricted to specific devices but rather intended to be used with any ECG signal in combination with the *BioSPPY* toolbox (e.g. extract R-peaks) or NNI series, independently of the acquisition device being used, the intended workflow adopted in this work is focused on supporting PLUX's *BITalino (r)evolution (BITalino)* biosignal acquisition hardware toolkits, their compatible *OpenSignals (r)evolution* software and the open-source *BioSPPY* BioSPPy toolbox for biosignal processing in Python, for which the pyHRV toolbox developed in this work will be integrated in the future. This approach is designed to demonstrate a possible workflow and usage example of this toolbox in real research applications.

BITalino boards are Arduino-like modular devices with single lead ECG sensors (among other physiological sensors) designed for rapid-prototyping purposes in research, development, and education (e.g. classroom support and student projects) [7]. The BITalino ECG sensor used within this work (for testing purposes and to ensure compatibility of the intended workflow with BITalino devices and this toolbox) is shown in Figure 16. The BITalino system (Board Kit) used in this work is presented in Figure 17. The datasheets of the ECG sensor and the BITalino Board Kit can be found in Appendices D and E.

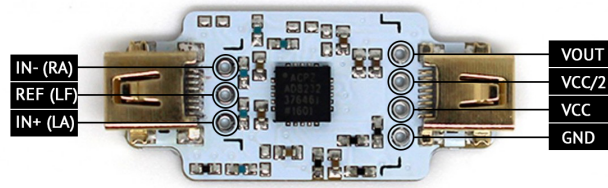


Figure 16: BITalino (r)evolution ECG sensor with open pins for power and analog signal output (right) and pinout for a Lead I single channel ECG setup (left).

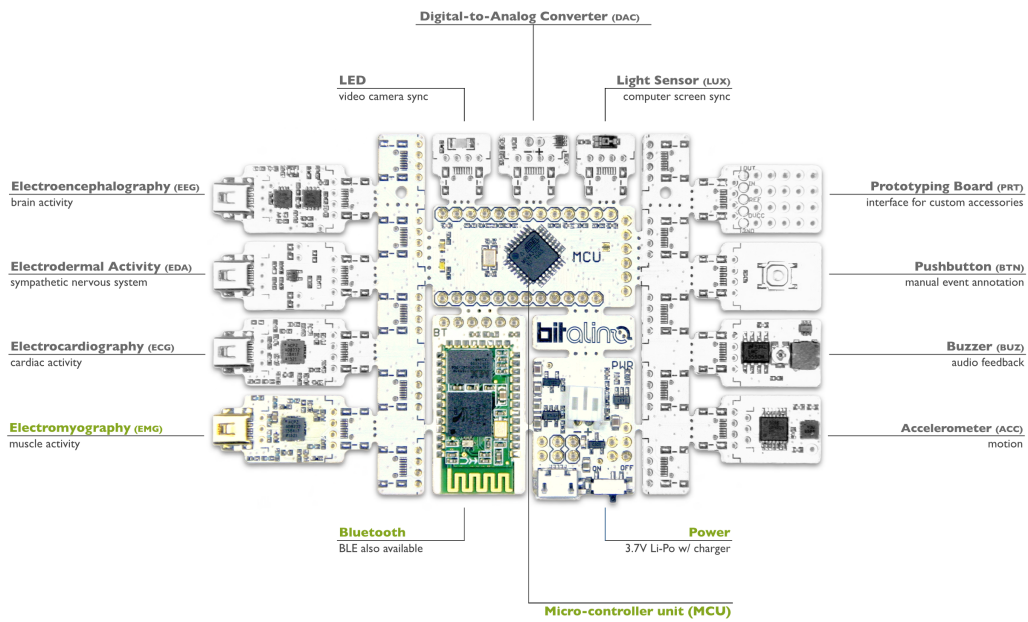


Figure 17: BITalino (r)evolution Board Kit. The most important modules for the acquisition of a single channel ECG signal are highlighted in color.

This workflow is split into two major steps: (a) signal acquisition and (b) script/software development in Python. In step (a) users are intended to use BITalino devices and the OpenSignals (r)evolution software to acquire ECG signals and to export the raw digitized signals in OpenSignals .TXT files. Afterwards in step (b), the ECG signal(s) are imported and converted using the additional *opensignalsreader* support packages developed within this work, which facilitates the signal import and automatically converts the acquired signals into their original units (mV) (see Appendix C.1 and Appendix G)¹. This step is followed by the use of the *BioSPPy* toolbox to filter the ECG signal(s) and extract the R-peak locations upon which the HRV are computed. The entire process is illustrated in Figure 18.

¹The BITalino device must be configured correctly in the OpenSignals (r)evolution software, i.e. the ECG channel must be configured as *ECG*, not as *RAW* or *CUSTOM*.

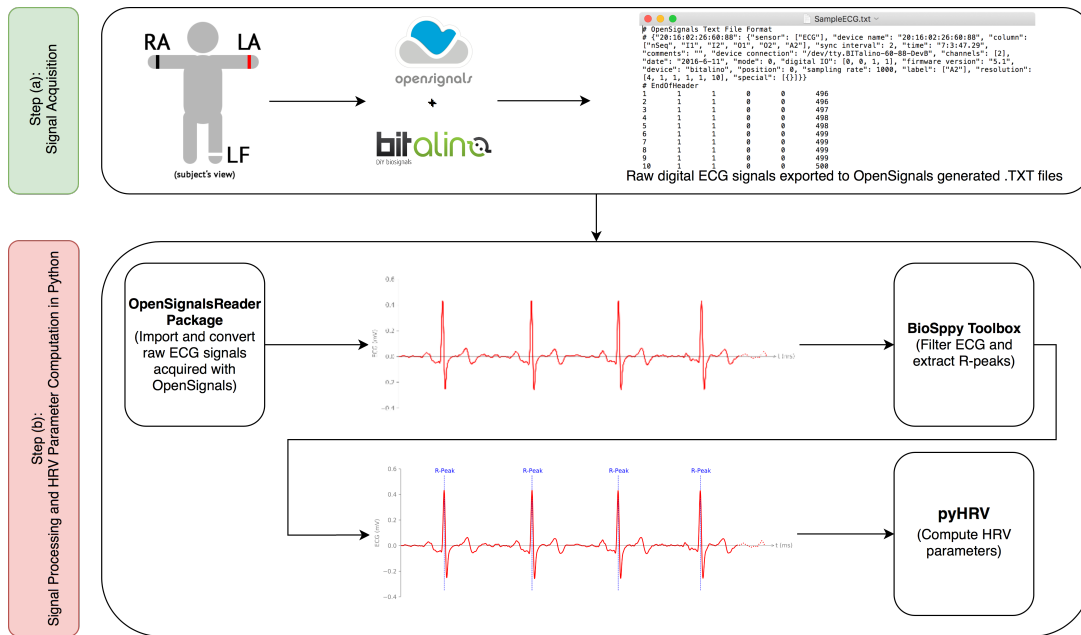


Figure 18: Illustration of the intended workflow from the ECG electrode placement and signal acquisition until the application of the HRV toolbox.

A minimal working example demonstrating step (b) of this workflow in the corresponding Python code is presented in the Python Snippet 1 which shows that only a few lines of code are necessary to go from the ECG signal import until the HRV parameter computation.

```

1 # Import packages (OpenSignals, BioSPPy and pyHRV)
2 from opensignalsreader import OpenSignalsReader
3 from bisppy.signals.ecg import ecg
4 from pyhrv.hrv import hrv
5
6 # Load ECG signal acquired with BITalino and OpenSignals
7 ecg_signal = OpenSignalsReader('SampleECG.txt').signals('ECG')
8
9 # Filter ECG signal and Extract R-peak locations
10 rpeaks = ecg(ecg_signal, show=False)[2]
11
12 # Compute HRV parameters
13 results = hrv(rpeaks=rpeaks)

```

Python Snippet 1: Minimum working example of the Python code required to import ECG signals acquired from and stored in OpenSignals files as .TXT files (line 7), followed by ECG filtering and R-peak extraction using the BioSPPy toolbox (line 10) upon which the HRV parameters are computed using the HRV toolbox (line 13).

3.2 Third-Party Tools, Software and Packages

The main goal of this work was to create a novel toolbox named pyHRV for the Python programming language; as such, multiple Python open-source libraries are used for the computation of HRV parameters, file and data management, and plotting the acquired signals and results. The development of this toolkit has been conducted in a virtual environment created with the *Conda* package and virtual environment management software, using a Python v.2.7.15 distribution and the *PyCharm Community 2017.2* Integrated Development Environment (IDE).

The *BioSPPy* package (v.0.5.1) is an open-source toolbox for biosignal processing in which the HRV toolbox developed in this work will be integrated in the future [17]. For this reason, the HRV toolbox follows the same Python format and documentation style guide as set forth by the *NumPy/SciPy* style format [53] (e.g. for the docstrings²), and uses *BioSPPy* functions throughout its modules. For instance, the `biosppy.signals.ecg.ecg()` function is used in the example code implemented in each module, to demonstrate how to properly conduct signal pre-processing using *BioSPPy* (e.g., ECG filtering, R-peak extraction), upon which the functions of the HRV toolbox can be applied. Additionally, results generated by the HRV toolbox are wrapped in objects of the *BioSPPy ReturnTuple* class found in the `biosppy.tools` module. This package-specific class combines the advantages of Python dictionaries, where values are stored in a mutable array-like object without a fix order and indexed using keywords (*dictionary keys*)³, and Python tuples, an immutable array-like object that is indexed as regular arrays with sequential numbers. These advantages of immutability of the tuple format, combined with the advantages of the key-based value indexing of dictionaries, facilitates the access of storing and reading parameter results with a single object, without the need of additional index tracking (as would be the case when using regular arrays or tuples).

The *NumPy* package provides an efficient array data type which is far more superior - in both memory size and processing time - to Python's native array-like list format [67]. Series of R-peak locations, NNI, or PSD results computed within this toolbox's functions are stored and processed in the *NumPy* array format. Additionally, functions to determine basic statistical parameters, such as the minimum, maximum or mean value of a series, as well as array manipulation functions used in this work are

²`docstrings` provide information about the function and are the output of the `help()` function when used in the Python interpreter (e.g. `help(sdnn)` prints the `docstring` of the SDNN function in the terminal).

³Imagine a dictionary named `results` which stores a SDNN value (e.g. 800ms) with a string-key named `sdnn`: `results['sdnn'] = 800`

conducted using functions of this package.

Frequency domain parameters are derived from PSD estimations obtained using different estimation methods. The computation of the FFT-based PSD estimation uses the interpolation (`scipy.interpolate.interp1d()`) and Welch's method (`scipy.signal.welch()`) algorithms from the *SciPy* (v.1.1.0) package for scientific computing in Python [38]. The LSP has been computed using SciPy's `scipy.signal.lombscargle()`. The AR PSD has been computed using the Yule-Walker algorithm implemented in the *Spectrum* package (`spectrum.pyule()`, v.0.5.2) [20, 40]. For the computation of the Nonlinear parameters (SamPen and DFA), an additional package known as *nolds* (*NOnLinear measures for Dynamical Systems*) (v.0.4.1) has been used, which has been specifically developed for the computation and analysis of nonlinear measures in datasets.

All functions with plotting capabilities use the *Matplotlib* package (v.2.2.2) [37]. *Matplotlib* is the most commonly used package for data visualization and plotting in Python, supported by many GUI development frameworks (e.g. TKinter⁴), and compatibility can be useful for future users of the pyHRV toolbox developed within the scope of this work and their application development.

Additional native Python packages have been used in this toolbox such as the *datetime* package to create timestamps in files generated by this toolbox (e.g. reports), the *os* package to help generate operating system compatible file paths for file management purposes, and the *warnings* package to trigger warnings printed to the Python console to provide additional information to the user whenever required (e.g. insufficient number of NNI samples to compute a parameter). Due to their implementation in the Python distribution, these packages do not provide individual version numbers, reason for which they will be listed as v.2.7.15 according to the Python version used during the development of this work. The exception here is the *JSON* package (v.2.0.9), which has been used to handle data import and export between Python and .JSON files. The .JSON format is a convenient format to store data in dictionary-like formats. Additionally, it is also human-readable making it more user-friendly compared to other file formats. In this work, the *JSON* library has been used to read and write .JSON files storing parameter-specific information such as the parameter *keys* of the *BioSPPy ReturnTuple* objects, the parameter descriptions used when exporting HRVA results in HRV report files, and to read and write the results of HRVA stored as *BioSPPy ReturnTuple* objects. An overview of all third-party packages used in the HRV toolbox are listed in Table 5.

⁴Matplotlib example code: https://matplotlib.org/gallery/user_interfaces/embedding_in_tk_canvas_sgskip.html

Table 5: Third party packages & versions used in this work.

Third-Party Packages & Versions			
Package	Version	Usage	Sources
biosppy	0.5.1	Biosignal processing toolbox	[17]
matplotlib	2.2.2	2D plotting library	[37]
numpy	1.15.1	Efficient array manipulation library	[67]
scipy	1.1.0	Welch’s method & signal interpolation	[38]
spectrum	0.5.2	Spectral analysis in Python	[20]
nolds	0.4.1	Nonlinear measures for nonlinear parameters	[78]
json	2.0.9	Import & Export of HRV results, keys and descriptions ⁿ	-
os	-	Tools for operating system-specific functionality ⁿ	-
datetime	-	Manipulation of dates and times ⁿ	-
warnings	-	Issues user warning notifications ⁿ	-

ⁿ Native Python package integrated in Python 2.7.15

3.3 Evaluation Procedures

In order to verify the correct functionality of the developed functions and tools of this toolbox, the developed parameter computation and helper functions (e.g. signal segmentation for SDANN and SDNN Index parameters) have been thoroughly tested and evaluated using the *KUBIOS HRV* software as reference software. *KUBIOS* is a *MATLAB* based software and, at the time being, a popular HRV software in HRV research, thus being considered as the *gold standard* among HRV tools [51, 52, 65, 66, 91]. The version used for the benchmarking in the scope of this work has been v.3.1. The approach followed in order to provide proper evaluation metrics, is summarized in Figure 19.

Step 1: Series of NNI have been selected from the `physionet.org` platform [31]. Although the physiological context(s) of the computed parameter results were neglected in this evaluation approach - after all, this method is intended to measure the correct functionality of the developed functions only - the NNI series used for this purpose were taken from the *MIT-BIH Normal Sinus Rhythm Database (MIT-BIH)*. This dataset consists of long-term ECG recordings of 18 healthy subjects (5 men between 26 and 40 years and 13 women between 20 and 50 years) with no observable pathological ECG arrhythmias, from which the NNI series have been downloaded directly from the *physionet* using the platforms *PhysioBank ATM (Automated Teller Machine)* [31]. The use of the NNI series has been preferred over the use of the raw ECG signals, to avoid possible differences in the implemented ECG pre-processing

and R-peak detection algorithms between *BioSPPy* and *KUBIOS*. Such variations could cause differences in the extracted R-peak series and the derived NNI series, which could ultimately influence the parameters results. Additionally, NNI series detrending has been deactivated in the *KUBIOS* software to avoid further processing of the NNI series in the *KUBIOS* software, which may have not be reproduced with *pyHRV*. This way, it is ensured that both tools are computing the HRV parameters on the exact same NNI series.

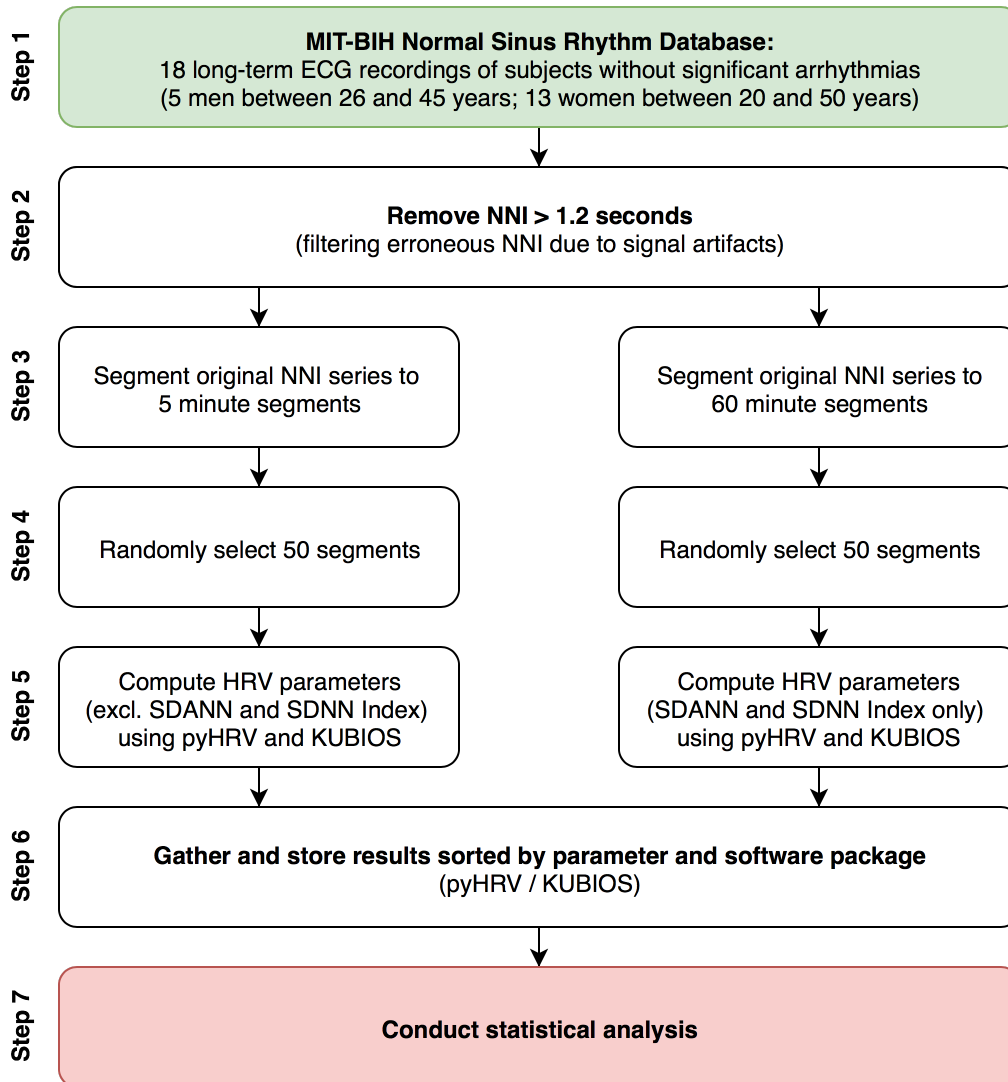


Figure 19: Flowchart of the evaluation metrics computation process.

Step 2: Outliers have been removed from the NNI series which have been introduced in the ECG recordings due to signal artifacts (e.g., movement artifacts, interrupted electrode-skin contact). In this series, NNI of up to approx. 42 seconds in duration ($\hat{=}0.7$ BPM) could be found which have been removed by applying a threshold of

$1.2s^5$ as maximum allowed NNI. If not removed, these outliers could distort the Tachogram and the HRV parameter results and cause issues in the signal segmentation process used in the next step. An example of such issues is shown in Figure 20 (top), where the outliers stand out significantly when compared to the rest of the NNI. The result of the filtered signal is shown in Figure 20 (bottom).

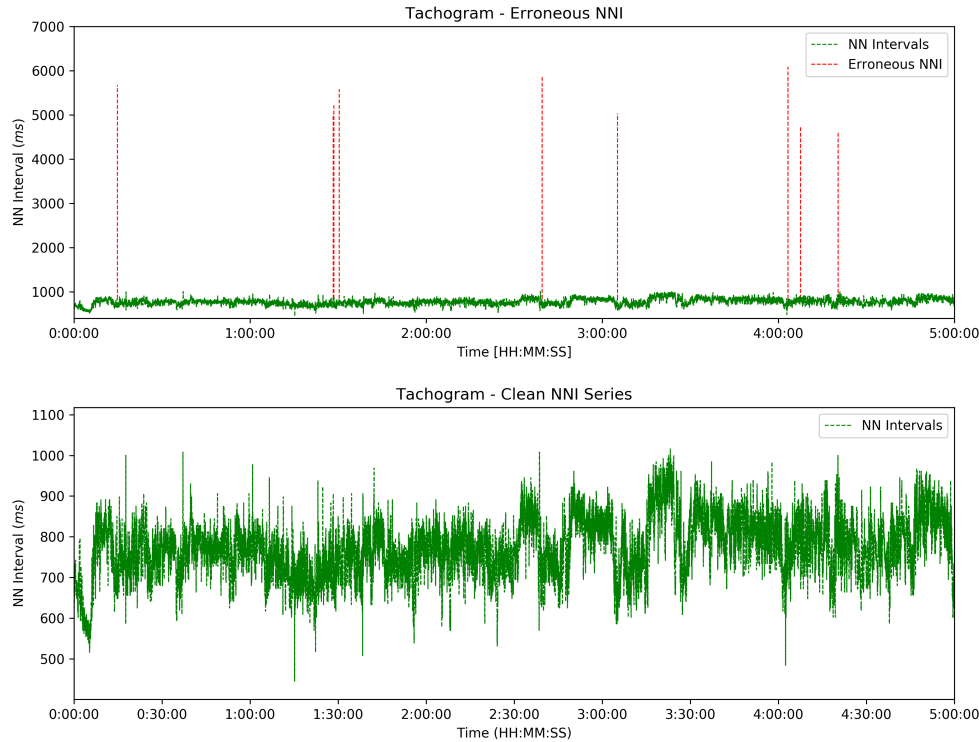


Figure 20: Tachogram comparison of a NNI series with NNI outliers caused by signal artifacts during the ECG acquisition (top), and the NNI series after applying a NNI threshold of 1.2 seconds (bottom) (signal 19090 [31]).

Step 3: The NNI series were split into segments of shorter durations, upon which the HRV parameters could be computed and statistically compared. Given that 18 series of NNI could be insufficient to conduct a significant comparison, the segmentation process using the `pyhrv.tools.segmentation()` function has been conducted to extract a greater number of NNI series from the original MIT-BIH database. A duration of 5 minutes per segment has been set for the computation of most of the HRV parameters, except for the SDANN and SDNN Index parameters, which required segments of longer duration as their computation itself conducts an independent segmentation process (see Chapters 2.3.1.5 and 2.3.1.6). For these parameters, a duration of 60 minutes has been selected.

Step 4: 50 segments of each duration have been randomly selected using the *NumPy*

⁵1.2 seconds $\hat{=}$ 50 BPM; 10 BPM lower than the lower limit of normal HR in healthy adult individuals according to the *American Heart Association* [2]

`numpy.random.randint()` function to randomly select indices of the generated segment series.

Step 5: The HRV parameters have been computed using `pyHRV` and *KUBIOS* and the results have been sorted by parameter. In order to derive comparable results, some parameter specific settings had to be equally set in `pyHRV` and *KUBIOS*. These settings consist of a selection of parameter values (e.g., frequency bands, embedding dimensions) that are listed in the table below.

Table 6: Selected parameter settings set in the toolbox and *KUBIOS* software to compute HRV results on the same datasets and using equal parameters.

Parameter Settings	
Parameter Specific Settings	Set Value(s)
NNI detrending method	None ^k
HR parameters computed as average of X heart beats	1 beat ^k
VLF Frequency Band	0.00Hz - 0.04Hz
LF Frequency Band	0.04Hz - 0.15Hz
HF Frequency Band	0.15Hz - 0.40Hz
FFT - Interpolation frequency	4Hz
FFT - Window width	300s
FFT - Window overlap	50 %
Lomb - Smoothing	None ^k
AR - Model order	16
Sample Entropy - Embedding dimension	2
Sample Entropy - Tolerance	0.2 · SD
DFA - Short term fluctuations	4 - 16 beats
DFA - Long term fluctuations	17 - 64 beats

^k *KUBIOS* HRV specific parameter settings only.

HRV results generated with *KUBIOS* were created by manually importing each NNI segment into *KUBIOS* and exporting the results in report files (.TXT, .MAT, and .PDF formats) of which a total of 200 *KUBIOS*⁶ individual reports of each file format have been created as follows:

- 50 reports for the evaluation of FFT, time domain and nonlinear results
- 50 reports for the evaluation of AR results
- 50 reports for the evaluation of LSP results
- 50 reports for the evaluation of the SDANN Index and SDANN results

⁶The extensive amount of *KUBIOS* reports and NNI segments have also been generated to provide enough data for the evaluation of new HRV features implemented in future versions of the toolbox.

Step 6: The results were exported using the toolbox’s `hrv.tools.hrv_export()` function and were stored in the .JSON file format. Additionally, the results have been sorted and summarized by parameter leading to data series of 50 results for each parameter⁷.

Step 7: A statistical analysis has been conducted. Although the NNI series have been randomly selected, it can be argued that the computed HRV results lost their characteristics of a random variable as their computation follow a non-random and structured method. Given that most statistical analysis methods depend on random variables (e.g. Chi-Square), their use might not be the most suitable option for the evaluation of the computed HRV results. Instead, the following statistical method has been used.

The mean values, $\overline{P_x}$ and $\overline{K_x}$, and the SD, $\sigma(P_x)$ and $\sigma(K_x)$ of each parameter series has been computed for the pyHRV (P_x) and KUBIOS (K_x) results, with x being one of the computed HRV parameters. Additionally, the absolute difference Δ_x between the $\overline{P_x}$ and $\overline{K_x}$ values has been computed. Finally, the pyHRV results have been classified into 3 categories, *Optimal*, *Acceptable*, and *Rejected*, depending on Δ_x and the $\sigma(K_x)$:

Optimal: $\Delta_x = 0$

The computed parameter results are identical, therefore, no difference exists between $\overline{P_x}$ and $\overline{K_x}$

Acceptable: $0 < \Delta_x \leq \sigma(K_x)$

The computed parameter results are not identical, but the difference is \leq the SD of the KUBIOS results. The difference is acceptable, as the differences are not significant.

Divergent: $\Delta_x > \sigma(K_x)$

The computed parameter results are not identical and a significant difference exists. These computed results are questionable.

⁷Import of the KUBIOS results has been achieved due to the development and use of the *kubios* support package (see Appendix C.2)

Chapter 4

Results

The implemented pyHRV toolbox has been developed in the Python package format, which allows developers to use it as an open-source library in their own software development applications. pyHRV has primarily been developed under and for Python 2.7 and has been publicly released on the GitHub repository system¹. An extensive API reference, which is intended to support users to implement the pyHRV functions, has been released on the ReadTheDocs platform². The entire list of computable parameters and the respective keys are presented in Appendix B.

4.1 Implementation of the HRV Toolbox

4.1.1 Package Architecture

The `toolbpx` contains 2 package specific files (Table 7 files #1 and 2) required for Python-specific purposes (e.g., package initialization, version tracking, storing package metadata), and the `Tools` module (Table 7 files #3), which provides fundamental functions for the computation of HRV base data (e.g. NNI series) and other useful features for HRV research (e.g. data import and export). The 4 core modules of this package (Table 7 files #4 to 8) contain all the parameter functions to compute the Time Domain, Frequency Domain, and Nonlinear parameters. Additionally, the `hrv.keys.json` file contains all the key and labels for the computed HRV parameters stored in the *BioSPPy* ReturnTuple objects (Table 7 file #8; see also Annex B).

¹pyHRV on GitHub: <https://github.com/PGomes92/pyhrv>

²pyHRV API Reference: <https://pyhrv.readthedocs.io/en/latest/>

The HRV feature extraction functions of the pyHRV toolbox have been implemented and categorized into three levels, which are intended to facilitate the usage and increase the usability of the toolbox according to the needs of the user or programming background. This multilevel-architecture is illustrated in Figure 21 and explained in greater detailed below.

Table 7: pyHRV package modules, files and descriptions.

pyHRV Package Files			
#	Module	File Name	Description
1	Package Init.	<code>__init__.py</code>	Package initialization file
2	Version	<code>__version__.py</code>	Package version tracker
3	Tools	<code>tools.py</code>	Fundamental tools and functions
4	HRV	<code>hrv.py</code>	Contains the <code>hrv()</code> function
5	Time Domain	<code>time_domain.py</code>	Time Domain parameter functions
6	Frequency Domain	<code>frequency_domain.py</code>	Frequency Domain parameter functions
7	Nonlinear	<code>nonlinear.py</code>	Nonlinear parameter functions
8	HRV Keys	<code>hrv_keys.json</code>	Parameter keys and labels

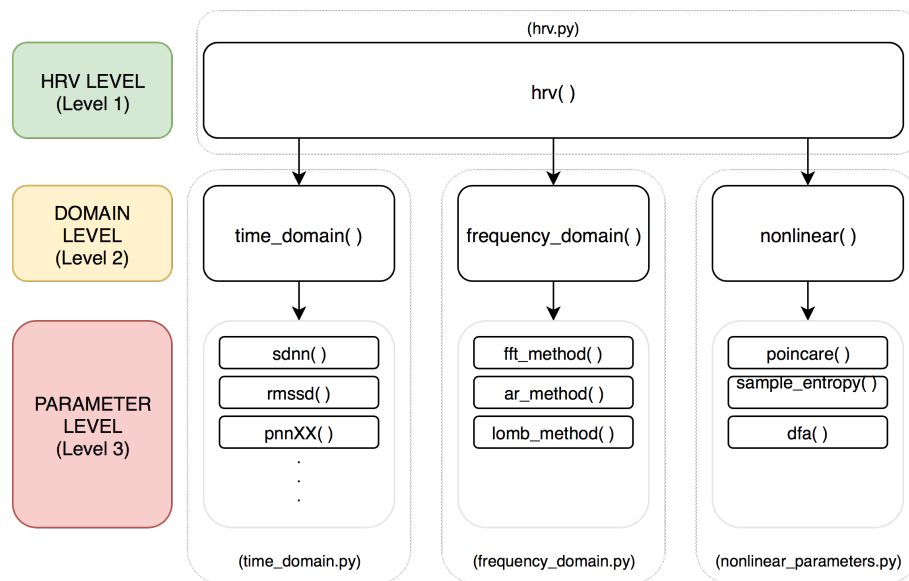


Figure 21: Multilevel architecture of the HRV package.

Level 1 - HRV Level: Consists of a single function that allows users to compute the full range of HRV parameters using only a single line of code by calling the `hrv()` function found in the `hrv.py`. This function calls all underlying HRV parameter functions and returns the bundled results in a `biosppy.utils.ReturnTuple()` object. User-specific settings or parameters for the computation of these parameters can be passed to the `hrv()` function if needed. Otherwise, the pre-set default values will

be used. This function can be especially useful for users with little programming backgrounds.

Level 2 - Domain Level: Consists of module/domain functions intended for users that only want to compute all the parameters of a specific domain. Using the example of the time domain, the `time_domain()` function calls all the parameter functions of this domain and returns the results bundled in a `biosppy.utils.ReturnTuple()` object. As in the previous level, user-specific settings and parameters can be passed to the domain functions, which are found in the respective modules. The module level function can be found in the respective domain modules.

Level 3 - Parameter Level: Contains parameter-specific functions for users that only want to compute single, specific parameters (individually) (e.g. `sdnn()` returns the SDNN parameter). This allows the user to select only the parameters or features required for their specific application with the advantage of avoiding unwanted additional processing time and memory spent on non-required parameters. User-specific settings and parameters can be made directly when overloading the function. The module level function can be found in the respective domain modules.

4.1.1.1 HRV Tools Module

The *Tools Module* (file: `tools.py`) contains functions with general purpose and key functionalities for the entire pyHRV toolbox, among other useful features designed to improve the usability of the entire toolbox. All functions are listed in Table 8 and can be categorized into four types of functions: (a) data series that generate data directly (functions #1-7), (b) plotting functions, (c) data import and export functions, and (d) data verification functions.

Functions of category (a) generate fundamental data series that are required for the computation of HRV parameters such as the NNI, Δ NNI, HR or signal segmentation series. Additionally, the `time_vector()` computes a time vector from a given ECG signal, which can be used for ECG visualization purposes. The `join_tuples()` joins multiple `biosppy.utils.ReturnTuple()` objects and returns them in a single, bundled object as used in the package level and domain level functions of this toolbox.

Functions of category (b) provide signal plotting facilities, commonly used in HRV applications. The resulting plots of the `plot_ecg()` and `tachogram()` functions are shown in Figure 22 and Figure 6 (see also Section 2.3.1.3, page 16). The plot layout generated by the `plot_ecg()` function resembles medical-grade ECG paper with the

Table 8: Functions of the Tools Module.

Tools Module		
#	Function Name	Description
1	<code>nn_intervals^s</code>	Computes a series of NNI from a series of R-peak locations
2	<code>nn_diff^s</code>	Computes a series of Δ NNI from a series of NNI
3	<code>heart_rate^s</code>	Computes HR value(s) from NNI or R-peak data
4	<code>segmentation^s</code>	Segmentation of R-peaks into individual segments of specified duration (e.g 5 minute) for computation of SDNN-index or SDANN)
5	<code>time_vector^s</code>	Computes time vector for an input ECG signal with specified sampling frequency
6	<code>join_tuples^s</code>	Bundles multiple <i>biosppy.utils.ReturnTuple()</i> objects
7	<code>std^s</code>	Computes standard deviation of an input array
8	<code>plot_ecg^s</code>	Plots ECG signal on a medical-grade ECG paper styled figure
9	<code>tachogram^s</code>	Plots NNI Tachogram
10	<code>hrv_report^s</code>	Creates a HRV report file in .TXT or .CSV format
11	<code>hrv_export^s</code>	Exports HRV results to .JSON file
12	<code>hrv_import^s</code>	Imports HRV results from hrv_export() .JSON file
13	<code>check_input^s</code>	Verifies and converts NNI and R-peak input data
14	<code>nn_format^s</code>	Verifies that NNI series is in <i>ms</i> format (converts <i>s</i> to <i>ms</i>)
15	<code>check_interval^s</code>	Checks interval boundaries of a given interval and fixes overlaps
16	<code>._check_limits^w</code>	Helper function for the check_interval() function
17	<code>._check_fname^w</code>	Verifies if given file name does not exists; creates a new file name otherwise

^s = strong function; callable outside the module

^w = weak function; only callable inside the module

t-axis divisions being automatically adjusted depending on the visualized signal interval to provide better visualization of the ECG signal.

The functions of category (c) are intended to facilitate the export and import of HRV features computed using this pyHRV. The [hrv_export\(\)](#) function exports results stored in a [biosppy.utils.ReturnTuple\(\)](#) object into a .JSON file (e.g., to permanently save results, export data to 3rd party software); latter these can be imported using the [hrv_import\(\)](#). Additionally, HRV reports in .TXT or .CSV format can be created using the [hrv_report\(\)](#) function. An example of such a report can be found in Appendix A.

Functions of the last category (d) are designed for data verification purposes and to catch erroneous data or data formats, which are either automatically fixed due to the implemented default values (and mechanisms), or that raise exceptions and notify

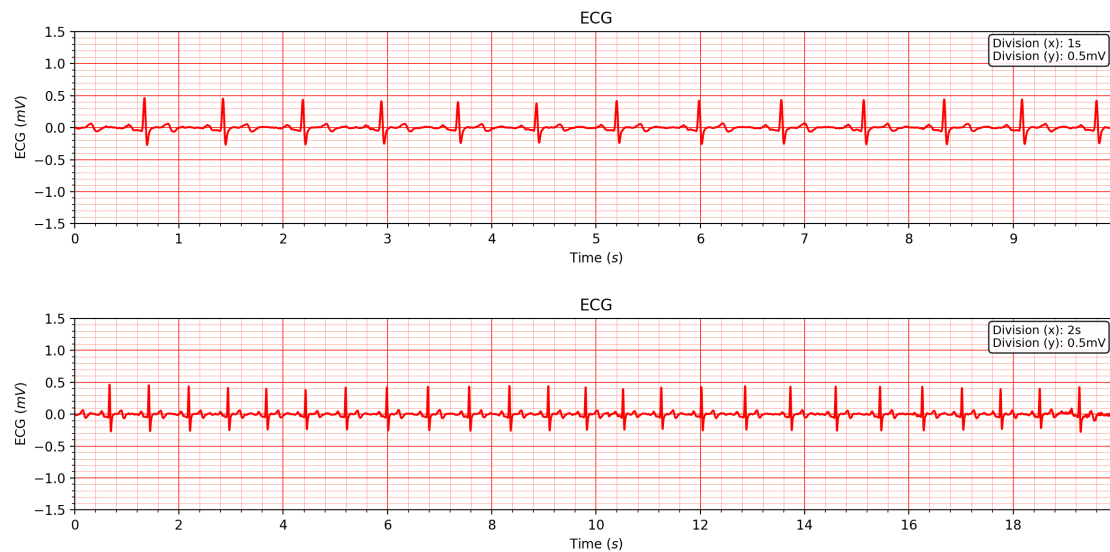


Figure 22: Sample ECG signal plotted with different intervals. The divisions are adjusted depending on the visualization interval (Top: 10 seconds signal interval with 1 second divisions; Bottom: 20 seconds signal interval with 2 seconds divisions).

the user. The `check_input()` and `nn_format()` functions ensure that the NNI series are available in the correct *NumPy* array in millisecond format. The `check_interval()` and its helper function `_check_limits()` verify interval limits and correct them if the intervals are overlapping or incorrect (e.g., lower boundary > upper boundary, used to check FB limits). The `_check_fname()` function is used by the functions of the category (c) and checks if the provided output file name to which data is meant to be written exist, and, if the file name exists, generates a new file name (e.g. `ReportFile.txt` → `ReportFile_1.txt`) to avoid accidental overwriting of files.

A minimum working example of the Tool Module's function is presented in the Python Snippet 2.


```

1 # Import libraries
2 import pyhrv.tools as tools
3 import pyhrv.time_domain as td
4 from bisppy.signals.ecg import ecg
5 from opensignalsreader import OpenSignalsReader
6
7 # Load sample ECG signal & extract R-peaks using BioSppy functions
8 ecg_raw = OpenSignalsReader('SampleECG.txt').signals['ECG']
9 ecg_filtered, rpeaks = ecg(ecg_raw, show=False)[1:3]
10
11 # Plot ECG & Tachogram (first 10s visualized by default)
12 tools.ecg_plot(ecg_filtered)
13 tools.tachogram(ecg_filtered)
14
15 # Compute NNI series and an example parameter of the Time Domain (e.g.
16     SDNN)
17 nni = tools.nn_intervals(rpeaks)
18 results = td.sdnn(nni)
19
20 # Create HRV report, JSON export & import
21 hrv_report(results)
22 hrv_export(results, '/my/path/', 'SampleExport')
23 hrv_import(results, filename='/my/path/SampleExport')

```

Python Snippet 2: Minimum working example of the Tools Module's functions.

4.1.1.2 Time Domain Module

The *Time Domain Module* (file: [time_domain.py](#)) contains all functions to compute the Time Domain parameters presented in Table 1, Chapter 2.3.1. Functions of the *Tools Module* are used within this module to compute the series of NNI, Δ NNI, and HR from which the time domain parameters are computed. In this module, each function has been implemented according to the function architecture presented in Chapter 4.1.2, and computes one specific time domain parameter (e.g. SDNN) or parameter set (e.g. basic statistical parameters of NNI series). These functions are listed in Table 9) and implemented according to the Equations 4-11 and the methods presented in Chapter 2.3.1. Additionally, the implemented [time_domain\(\)](#) function has been added, which calls all of the functions #1-11 in accordance with the package architecture presented in Chapter 4.1.1.

A minimum working example to demonstrate the code required for the computation of an individual time domain parameter, as well as for the set of Time Domain

parameters using the `time_domain()` function is shown in Python Snippet 3.

Table 9: Functions of the time domain module.

Time Domain Module		
#	Function Name	Description
1	<code>nn_parameters</code>	Computation of NNI parameters (min, max, mean)
2	<code>nn_differences_parameters</code>	Computation of Δ NNI parameters (min, max, mean)
3	<code>hr_parameters</code>	Heart rate parameters (min, max, mean)
4	<code>sdnn</code>	Standard deviation of a NNI series
5	<code>sdnn_index</code>	Standard deviation of the means of 5 minute segments of long-term recordings
6	<code>sdann</code>	Mean of the standard deviations of 5 minute segments of long-term recordings
7	<code>rmssd</code>	Root mean of squared Δ NNI
8	<code>sdsd</code>	Standard deviation of Δ NNI
9	<code>nnXX</code>	Number & percentage of Δ NNI greater than threshold <i>XX</i> in a NNI series
10	<code>nn50</code>	Number & percentage of Δ NNI greater than 50 milliseconds in a NNI series
11	<code>nn20</code>	Number & percentage of Δ NNI greater than 20 milliseconds in a NNI series
12	<code>time_domain</code>	Calls all of the functions above and returns results in a single <i>ReturnTuple</i> object

```

1 # Import libraries
2 import pyhrv.time_domain as td
3 from bisppy.signals.ecg import ecg
4 from opensignalsreader import OpenSignalsReader
5
6 # Load sample ECG signal & extract R-peaks using BioSppy functions
7 ecg_signal = OpenSignalsReader('SampleECG.txt').signals['ECG']
8 rpeaks = ecg(ecg_signal, show=False)[2]
9
10 # OPTION 1: Compute parameters & methods individually
11 results = td.sdnn(rpeaks=rpeaks)
12 print(results['sdnn'])
13
14 # OPTION 2: Compute all Time Domain parameters & methods using
15 results = td.time_domain(rpeaks=rpeaks)
16 print(results['sdnn'])

```

Python Snippet 3: Minimum working example of the Time Domain Module's individual parameter functions (in this case `sdnn()`) and the module level function `time_domain()`.

4.1.1.3 Frequency Domain Module

The *Frequency Domain Module* (file: [frequency_domain.py](#)) contains all functions to compute the Frequency Domain parameters listed in Table 2, Chapter 2.3.3 and presented in Chapter 2.3.3.5. These parameters are derived from the application of the 3 different frequency analysis methods presented in Chapters 2.3.3.2 to 2.3.3.4, i.e. they are applied to each of the methods generating method-specific results.

The FFT-based PSD estimation method has been implemented with the use of tools from the *SciPy* package. This implementation uses the [scipy.interpolate.interp1d\(\)](#) function for the interpolation and re-sampling of the NNI series, followed by the computation of the PSD estimation using the [scipy.signal.welch\(\)](#) algorithm (Welch's method). The implementation of the Lomb-Scargle based PSD estimation has been implemented using the [scipy.signal.lombscargle\(\)](#) function of the *SciPy* package [38]. The implementation of the AR PSD uses the [scipy.interpolate.interp1d\(\)](#) function for the interpolation and re-sampling of the NNI series, followed by the computation of the PSD estimation using the [spectrum.pyule\(\)](#) class of the *Spectrum* package [20]. For this function, a default model order of 16 has been set, according to the minimum recommended model order for short NNI segments [10].

The functions that compute the FFT-based and Lomb-Scargle frequency analysis methods are listed in Table 10 (functions #1 to 3), and have been implemented as strong functions of this module (i.e. they can be used outside the module). This module includes several additional functions to verify the correctness of the selected frequency band limits (function #4), proceed with signal segmentation according to the selected frequency bands (functions #5 and 6), compute the frequency domain parameters according to the Equations 13-20 (function #7), and plot the resulted PSD (function #8). These functions have been implemented as weak functions (i.e. they can only be used within the module) due to their nature of being helper functions for the functions #1 to 2.

A minimum working example of this module's functionality is presented in the Python Snippet 4.

Table 10: Frequency Domain Module functions.

Frequency Domain Module		
#	Function Name	Description
1	welch_psd ^s	Computes the FFT-PSD (plot & parameters)
2	lomb_psd ^s	Computes the Lomb-Scargle PSD (plot & parameters)
3	ar_psd ^s	Computes the Autoregressive PSD (plot & parameters)
4	._check_frequency_bands ^w	Verifies the correctness of frequency band limits (e.g. preventing overlapping bands)
5	._get_frequency_indices ^w	Segments indices of the PSD frequency series according to the frequency band boundaries
6	._get_frequency_arrays ^w	Segments sub-series of the PSD frequency series according to frequency band boundaries
7	._compute_parameters ^w	Computes Frequency Domain features
8	._plot_psd ^w	Plots PSD and adds legends (optional)
9	frequency_domain ^s	Calls all of the functions above and returns results as a single biosppy.utils.ReturnTuple() object

s = strong function; callable outside the module

w = weak function; only callable inside the module

```

1 # Import libraries
2 import pyhrv.tools as tools
3 import pyhrv.frequency_domain as fd
4 from biosppy.signals.ecg import ecg
5 from opensignalsreader import OpenSignalsReader
6
7 # Load sample ECG signal & extract R-peaks using BioSppy functions
8 ecg_signal = OpenSignalsReader('SampleECG.txt').signals['ECG']
9 rpeaks = ecg(ecg_signal, show=False)[2]
10
11 # Compute NNI series
12 nni = tools.nn_intervals(rpeaks)
13
14 # OPTION 1: Compute parameters & methods individually
15 results = fd.welch_psd(nni, show=True)
16 print(results['fft_abs'])
17
18 # OPTION 2: Compute all frequency parameters & methods
19 results = fd.frequency_domain(nni)
20 print(results['fft_abs'])
21 print(results['lomb_abs'])

```

Python Snippet 4: Minimum working example of the Frequency Domain Module's individual parameter functions (here: [fft_analysis\(\)](#)) and the module level function [frequency_domain\(\)](#).

4.1.1.4 Nonlinear Parameters Module

The *Nonlinear Parameter Module* (file: [nonlinear.py](#)) contains functions to compute the nonlinear HRV parameters listed in Table 11.

Table 11: Functions of the nonlinear parameters module.

Nonlinear Parameters Module		
#	Function Name	Description
1	<code>poincare</code>	Plots Poincaré plot and computes related parameters (SD1, SD2, ellipse area S, SD1/SD2 ratio)
2	<code>sample_entropy</code>	Sample Entropy
3	<code>dfa</code>	Detrended Fluctuation Analysis for short and long term fluctuations (α_1 & α_2)
4	<code>nonlinear</code>	Calls all of the nonlinear functions above and returns results in a single biosppy.utils.ReturnTuple() object

The [poincare\(\)](#) function generates the Poincaré plot presented in Chapter 2.3.4.1 and derives the Poincaré parameters SD1, SD2, ellipse area (S), and SD1/SD2 ratio. The computation of the SD1 and SD2 parameters has been implemented according to the time domain based computations presented in Equations 21 and 23. The computation of the ellipse area S has been implemented according to Equation 25 with the SD1/SD2 ratio computed as the division of both parameters.

The [sample_entropy\(\)](#) function has been implemented with the use of the *nolds* package with the default values set to embedding dimension = 2 and tolerance = 0.2 SDNN [73]. The DFA algorithm of the *nolds* package has been used and adjusted for the HRV specific application with default values set to 4-16 beats for short-term fluctuations and 17-64 for long-term fluctuations [101]. Additionally, error-catching capabilities have been implemented to catch errors if a DFA could not be conducted due to the lack of NNI samples (e.g. signal duration too short). In this case, an empty plot is returned and a warning message triggered to the user, otherwise a plot based on the DFA results will be returned as seen in Figure 15.

Python Snippet 5 presents a minimum working example of this module's functions using the example of the [dfa\(\)](#) and [nonlinear\(\)](#) functions.

```

1 # Import libraries
2 import pyhrv.nonlinear as nl
3 import pyhrv.tools as tools
4 from biosppy.signals.ecg import ecg
5 from opensignalsreader import OpenSignalsReader
6
7 # Load sample ECG signal & extract R-peaks using BioSppy functions
8 ecg_signal = OpenSignalsReader('SampleECG.txt').signals['ECG']
9 rpeaks = ecg(ecg_signal, show=False)[2]
10
11 # Compute NNI series
12 nni = tools.nni_intervals(rpeaks)
13
14 # OPTION 1: Compute parameters individually
15 results = nl.dfa(nni)
16 print(results['dfa'])
17
18 # OPTION 2: Compute all nonlinear parameters
19 results = nl.nonlinear(nni)
20 print(results['dfa'])

```

Python Snippet 5: Minimum working example of the Nonlinear Parameter Module’s individual parameter functions (here: `dfa()`) and the module level function `nonlinear()`.

4.1.2 Function Architecture

The functions of this toolkit were implemented according to a 4-element architecture, designed to improve their usability and robustness. This architecture is visualized in the flow-chart presented in Figure 23.

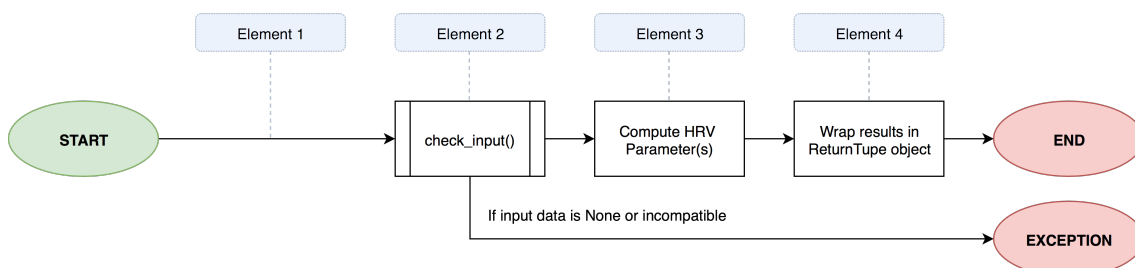


Figure 23: Flowchart of the function architecture used in parameter functions of the pyHRV toolkit.

Element 1: Consists of a comprehensive `docstring` containing a short description of the function, the input and output parameters, a list of possible exceptions being raised and, in some cases, further information about the correct usage of the function.

This element does not have an influence in the function's code, but is considered good practice in Python programming, to provide in-code documentation for the user.

Element 2: Verifies the correctness of the input data that the datasets used for feature extraction have the right structure and units; this is done using the implemented `tools.check_input()` function. Depending on the provided input data, this step follows one of the following processes:

- (a) Input: R-peak series in [ms] or [s]
 - ⇒ Computation of NNI series in milliseconds and *NumPy* array format
- (b) Input: NNI series in [ms] or [s]
 - ⇒ Computation of NNI series in milliseconds and *NumPy* array format
- (c) Input: None or incompatible data format (not *list* or *NumPy* array)
 - ⇒ Exception raised with comprehensive error descriptions to help the user during debugging processes

This procedure is visualized in the flowchart in Figure 24. It must be noted that the automatic second to millisecond conversion is conducted depending on the result of the maximum NNI and a threshold of 10. If the NNI series maximum is 10, then it is assumed that the input data is provided in *ms* given that a NNI of 10s ($\hat{=}6BPM$) is highly unlikely to be of physiological nature. Additionally, this straightforward method assumes that the NNI series does not contain any erroneous NNI.

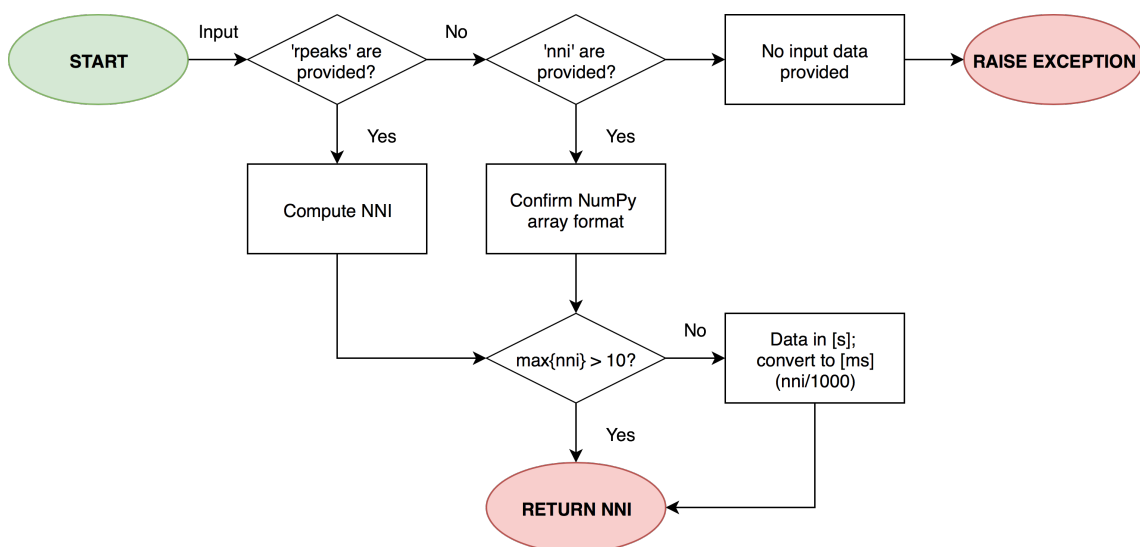


Figure 24: Flowchart of the `tools.check_input()` function.

Element 3: Computation of the HRV parameters according to the formulas, methods, and algorithms presented in Chapter 1.

Element 4: The results are wrapped and returned in a [biosppy.utils.ReturnTuple\(\)](#) object.

```

1 def rmssd(nni=None, rpeaks=None):
2     """Computes root mean of squared differences of successive NN
3         Intervals.
4
5         Parameters
6         -----
7         nni : array
8             NN intervals in [ms] or [s].
9         rpeaks : array
10            R-peak times in [ms] or [s].
11
12            Returns (biosppy.utils.ReturnTuple Object)
13            -----
14            rmssd : float
15                RMSSD value in [ms].
16
17            Notes
18            -----
19            .. Only one type of input data is required.
20            .. If both 'nni' and 'rpeaks' are provided, 'rpeaks' will be chosen
21               over the 'nni' and the 'nni' data will be computed from the 'rpeaks'
22               '.
23            .. NN and R-peak series provided in [s] format will be converted to
24               [ms] format.
25            """
26     # Check input
27     nn = tools.check_input(nni, rpeaks)
28
29     # Compute RMSSD
30     nnd = tools.nni_diff(nn)
31     rmssd = np.sum(x**2 for x in nnd)
32     rmssd = np.sqrt(1. / nnd.size * rmssd)
33
34     # Output
35     return utils.ReturnTuple((rmssd, ), ('rmssd', ))

```

Python Snippet 6: General architecture of the toolbox parameter functions shown on the example of the RMSSD function.

An example of this function architecture is shown in Python Snippet 6. A comprehensive docstring is provided at the beginning of the function (Element 1, lines 2-23), follow by the verification of the input data (Element 2, line 25), the computation of HRV feature, which in this case is the RMSSD parameter, (Element 3, line 28-29), and finally the output in a `biosppy.utils.ReturnTuple()` object (line 32-34).

This architecture has not been implemented in the functions of the Tools Module (described in more detail in the following pages) as it might not be the most appropriate form of output. For instance, functions that create the NNI or Δ NNI series return their results in the regular *NumPy* array as there is no need for a key-based indexing method to access the data.

4.2 Evaluation

4.2.1 Time Domain Parameters

The evaluation results for Time Domain comparison of the pyHRV and KUBIOS results are presented in Table 12. Evaluation metrics could be computed from 12 of the 23 computable Time Domain parameters using pyHRV, with 11 parameters being not comparable as these are not computed by KUBIOS.

Table 12: Evaluation results for the Time Domain parameters (pyHRV vs. KUBIOS)

Time Domain Evaluation metrics				
Parameter	Unit	pyHRV	KUBIOS	Δ
Mean NNI	[<i>ms</i>]	836.884 ± 143.429	836.884 ± 143.429	0.000
Min HR	[<i>bpm</i>]	57.754 ± 7.508	57.754 ± 7.508	0.000
Max HR	[<i>bpm</i>]	91.048 ± 16.342	91.048 ± 16.342	0.000
Mean HR	[<i>bpm</i>]	74.505 ± 13.545	73.953 ± 13.422	0.552
SDNN	[<i>ms</i>]	65.937 ± 26.291	65.937 ± 26.291	0.000
SDNN Index	[<i>ms</i>]	60.018 ± 16.965	60.338 ± 17.354	0.320
SDANN	[<i>ms</i>]	41.397 ± 18.281	42.113 ± 19.331	0.716
RMSSD	[<i>ms</i>]	42.485 ± 22.611	42.484 ± 22.611	0.001
NN50	[–]	58.120 ± 46.789	58.120 ± 46.789	0.000
pNN50	[%]	17.187 ± 15.440	17.187 ± 15.440	0.000
Triangular Index	[–]	9.128 ± 3.503	9.470 ± 3.886	0.342
TINN	[<i>ms</i>]	110.313 ± 59.100	305.200 ± 104.915	194.887

Optimal

Acceptable

Divergent

Uncomparable parameters: NNI (Min, Max), Δ NNI parameters, NN20, pNN20, TINN N and M

Δ : Difference between pyHRV and KUBIOS mean value

Overall 10 of the 11 comparable results lie within the optimal (A) and acceptable (B) ranges, with the TINN parameter being the only divergent (C) parameter. The TINN’s difference between the means is almost twice as high as the KUBIOS mean result with pyHRV’s mean value at roughly only a third of the KUBIOS mean ($SD = 104.915ms$ vs. $\Delta = 194.887ms$), thus showing a significant difference. Although KUBIOS does not return the N and M values of the TINN parameter computation, it can be assumed that these parameters are also divergent given that they are the base parameters for the computation of the TINN parameter (see Equation 12, page 23).

The mean of the NNI, minimum HR, SDNN, NN50, and pNN50 parameter results show optimal performance being identical to the KUBIOS results. It can be assumed that the uncomparable NN20 and pNN20 parameters also achieve optimal

results given that these parameters are computed by the same fundamental function as the NN50 and pNN50 parameters ([tools.time.domain.nnXX\(\)](#)). The mean HR ($SD = 13.422bpm$ vs. $\Delta = 0.552bpm$), SDNN index ($SD = 17.352ms$ vs. $\Delta = 0.320ms$), Standard Deviation of the Mean of NN Intervals in all 5 minute Segments (SDANN) ($SD = 19.331ms$ vs. $\Delta = 0.716ms$), RMSSD ($SD = 22.611ms$ vs. $\Delta = 0.001ms$), and the Triangular index ($SD = 3.886[-]$ vs. $\Delta = 0.342[-]$) lie within acceptable limits, with the RMSSD being close the values computed by KUBIOS, with a marginal difference of $1\mu s$.

4.2.2 Frequency Domain Parameters

The evaluation results of the Frequency Domain parameters consist of a comparison of 16 parameters for the three PSD estimations according to the Welch’s, Lomb-Scargle and Autoregressive methods, resulting in a total of 48 comparable parameters.

Table 13: Evaluation results for the parameters extracted with Welch’s method (pyHRV vs. KUBIOS)

Welch’s Method Evaluation Metrics				
Parameter	Unit	pyHRV	KUBIOS	Δ
Peak Frequencies (VLF)	[Hz]	0.012 ± 0.009	0.012 ± 0.010	0.000
Peak Frequencies (LF)	[Hz]	0.071 ± 0.023	0.071 ± 0.023	0.000
Peak Frequencies (HF)	[Hz]	0.226 ± 0.066	0.226 ± 0.066	0.000
Abs. Powers (VLF)	[ms^2]	2361.001 ± 2968.427	2325.941 ± 2963.510	35.060
Absolute Powers (LF)	[ms^2]	1518.788 ± 1292.658	1510.333 ± 1313.584	8.455
Absolute Powers (HF)	[ms^2]	883.304 ± 1561.402	884.503 ± 1568.995	1.199
Log Powers (VLF)	[log]	7.192 ± 1.078	7.168 ± 1.088	0.024
Log Powers (LF)	[log]	6.954 ± 0.931	6.933 ± 0.953	0.021
Log Powers (HF)	[log]	6.115 ± 1.030	6.102 ± 1.050	0.013
LF/HF ratio	[-]	3.234 ± 2.241	3.239 ± 2.280	0.005
Total Power	[ms^2]	4763.093 ± 4214.227	4722.274 ± 4243.256	40.819
Relative Powers (VLF)	[%]	45.752 ± 20.912	45.594 ± 21.027	0.158
Relative Powers (LF)	[%]	34.643 ± 14.488	34.572 ± 14.607	0.071
Relative Powers (HF)	[%]	19.605 ± 17.882	19.793 ± 18.033	0.188
Normalized Powers (LF)	[-]	67.638 ± 19.651	67.381 ± 19.915	0.257
Normalized Powers (HF)	[-]	32.362 ± 19.651	32.548 ± 19.894	0.186

Optimal
 Acceptable
 Divergent

Δ : Difference between pyHRV and KUBIOS mean value

The Frequency Domain parameters computed from the Welch’s method are presented in Table 13, and show overall optimal and acceptable results. pyHRV’s peak

frequencies in each frequency band are identical to the results obtained with KUBIOS. Differences can be found for the rest of the parameters, although marginal, given that the mean results exist within the acceptable ranges (see Table 13).

Table 14: Evaluation results for the frequency domain parameters extracted using the Lomb-Scargle PSD (pyHRV vs. KUBIOS)

Lomb-Scargle Method Evaluation Metrics				
Parameter	Unit	pyHRV	KUBIOS	Δ
Peak Frequencies (VLF)	[Hz]	0.017 \pm 0.010	0.011 \pm 0.009	0.006
Peak Frequencies (LF)	[Hz]	0.093 \pm 0.032	0.065 \pm 0.019	0.028
Peak Frequencies (HF)	[Hz]	0.290 \pm 0.045	0.227 \pm 0.066	0.063
Abs. Powers (VLF)	[ms ²]	216.763 \pm 49.541	2709.483 \pm 3700.016	2492.720
Absolute Powers (LF)	[ms ²]	608.678 \pm 155.304	1508.002 \pm 1109.925	899.324
Absolute Powers (HF)	[ms ²]	1463.945 \pm 267.442	826.661 \pm 1454.514	637.284
Log Powers (VLF)	[log]	5.354 \pm 0.221	7.309 \pm 1.038	1.955
Log Powers (LF)	[log]	6.380 \pm 0.251	7.038 \pm 0.806	0.658
Log Powers (HF)	[log]	7.271 \pm 0.190	6.133 \pm 0.911	1.138
LF/HF ratio	[-]	0.417 \pm 0.077	3.316 \pm 2.111	2.899
Total Power	[ms ²]	2289.386 \pm 414.363	5045.996 \pm 4600.503	2756.610
Relative Powers (VLF)	[%]	9.558 \pm 1.777	46.894 \pm 20.095	37.336
Relative Powers (LF)	[%]	26.422 \pm 3.418	34.626 \pm 13.803	8.204
Relative Powers (HF)	[%]	64.020 \pm 3.843	18.432 \pm 17.325	45.588
Normalized Powers (LF)	[-]	29.225 \pm 3.820	69.102 \pm 18.750	39.877
Normalized Powers (HF)	[-]	70.775 \pm 3.820	30.809 \pm 18.725	39.966

Optimal
 Acceptable
 Divergent

Δ : Difference between pyHRV and KUBIOS mean values

The Frequency Domain parameters computed from the Lomb-Scargle method are presented in Table 14. The evaluation results of this method identify half of the parameters within acceptable ranges, with the other half providing rejected results. No optimal results are computed by pyHRV. The Peak Frequencies show acceptable results for the VLF ($SD = 0.009Hz$ vs. $\Delta = 0.006Hz$) and HF ($SD = 0.066Hz$ vs. $\Delta = 0.063Hz$) with the LF results being rejected due to a greater difference than the KUBIOS results' SD ($SD = 0.019Hz$ vs. $\Delta = 0.028Hz$). Other acceptable results could be achieved for the Absolute Powers in all FB, the Total Power, and the Relative and Logarithmic Powers in LF band. The remaining parameters have been rejected. It is worth mentioning that no consistency over FB-specific results can be observed. For instance, the Absolute Powers provide acceptable results in the VLF and HF band. However, parameters derived from these parameters (Log. Powers and Relative Powers) show rejected results in these bands. This inconsistency influences additional parameters which depend on the Absolute Powers of the HF

band (LF/HF ratio and Normalized Powers), leading to the rejection of the results. The evaluation results of the Autoregressive parameters are presented in Table 15. Overall the results of 6 parameters lie within acceptable ranges, with 10 parameter results being divergent due to differences greater than the SD of the KUBIOS results. The Peak Frequencies lie within acceptable ranges for every FB. Other acceptable parameter results were computed for the Absolute Powers ($SD = 2666.353ms^2$ vs. $\Delta = 645.599ms^2$), and Log. Powers ($SD = 0.978ms^2$ vs. $\Delta = 0.742ms^2$) of the VLF band, and the Relative Powers of the LF frequency band ($SD = 14.716\%$ vs $\Delta = 6.466\%$). However, all remaining results lie outside the acceptable 1 SD limits. It should be noted that overall the Absolute Powers, with the exception of the VLF band, are significantly higher and lie multiple SD apart from the KUBIOS results, being almost 4.3 times higher in the LF band ($SD = 1137.171ms^2$ vs. $\Delta = 4858.550ms^2$) 8 times higher in the HF band ($SD = 1455.146ms^2$ vs. $\Delta = 11512.643ms^2$).

Table 15: Evaluation results for the parameters extracted using the Autoregressive Method (pyHRV vs. KUBIOS)

Autoregressive Method Evaluation Metrics				
Parameter	Unit	pyHRV	KUBIOS	Δ
Peak Frequencies (VLF)	[Hz]	0.000 ± 0.000	0.012 ± 0.015	0.012
Peak Frequencies (LF)	[Hz]	0.040 ± 0.000	0.045 ± 0.018	0.005
Peak Frequencies (HF)	[Hz]	0.155 ± 0.021	0.206 ± 0.071	0.051
Abs. Powers (VLF)	[ms^2]	2940.517 ± 62.056	2294.918 ± 2666.343	645.599
Absolute Powers (LF)	[ms^2]	6411.443 ± 159.130	1552.893 ± 1137.171	4858.550
Absolute Powers (HF)	[ms^2]	12379.985 ± 572.088	867.342 ± 1445.146	11512.643
Log Powers (VLF)	[log]	7.986 ± 0.021	7.244 ± 0.978	0.742
Log Powers (LF)	[log]	8.766 ± 0.025	7.056 ± 0.847	1.710
Log Powers (HF)	[log]	9.423 ± 0.046	6.217 ± 0.903	3.206
LF/HF ratio	[−]	0.518 ± 0.015	3.086 ± 1.943	2.568
Total Power	[ms^2]	21731.945 ± 748.929	4716.715 ± 3671.023	17015.230
Relative Powers (VLF)	[%]	13.539 ± 0.299	44.497 ± 19.693	30.958
Relative Powers (LF)	[%]	29.515 ± 0.492	35.981 ± 14.719	6.466
Relative Powers (HF)	[%]	56.946 ± 0.766	19.482 ± 17.338	37.464
Normalized Powers (LF)	[−]	34.138 ± 0.670	67.972 ± 19.013	33.834
Normalized Powers (HF)	[−]	65.862 ± 0.670	31.949 ± 19.006	33.913

Optimal

Acceptable

Divergent

Δ : Difference between pyHRV and KUBIOS mean value

4.2.3 Nonlinear Parameters

The evaluation results for the Nonlinear parameters are listed in Table 16. Overall, the 7 comparable parameters provide acceptable results with the SD2/SD1 ratio showing even optimal results. Marginal differences are present for the SD1 ($16.015ms$ vs. $\Delta = 0.044ms$) and SD2 ($SD = 35.962ms$ vs. $0.123ms$) parameters, with the SamPen results achieving almost optimal results ($SD = 0.366ms$ vs. $\Delta = 0.001$). The highest relative differences are present in the results of the DFA α_1 ($SD = 0.240[-]$ vs. $\Delta = 0.022[-]$) and α_2 ($SD = 0.199[-]$ vs. $\Delta = 0.0047[-]$) parameters. The area of the fitted ellipse in the Poincaré plot could not be compared, as this parameter is not computed by KUBIOS. However, given that this parameters is computed based on a multiplication of the SD1 and SD2 parameters with Pi (see Equation 25, page 35), and considering that these parameters have low relative differences, it can be assumed that this parameter also lie within acceptable limits.

Table 16: Evaluation results for the Nonlinear parameters (pyHRV vs. KUBIOS)

Nonlinear Parameters Evaluation Metrics				
Parameter	Unit	pyHRV	KUBIOS	Δ
SD1	[ms]	30.041 ± 15.988	30.085 ± 16.015	0.044
SD2	[ms]	87.171 ± 35.907	87.294 ± 35.962	0.123
SD2/SD1	[-]	3.204 ± 1.306	3.204 ± 1.306	0.000
Sample Entropy	[-]	1.332 ± 0.365	1.331 ± 0.366	0.001
DFA - α_1	[-]	1.204 ± 0.264	1.182 ± 0.240	0.022
DFA - α_2	[-]	0.886 ± 0.215	0.839 ± 0.199	0.047

Optimal
 Acceptable
 Divergent

Uncomparable parameter: Area of fitted Ellipse (Poincaré)

Δ : Difference between pyHRV and KUBIOS mean value

Chapter 5

Conclusions

5.1 Discussion

The presented work provides an open-source toolbox for HRV named pyHRV for the Python 2.7 programming language. The implemented multilevel package and function architectures allow user to compute the entire set of computable HRV parameters (Level 1 - HRV Level), domain-specific parameter sets (Level 2 - Domain Level), or individual parameters only (Level 3 - Parameter Level). The implemented error catching capabilities allow a reliable and robust use of this toolbox, even in cases where unsuitable input data formats are provided (NNI series in seconds) or where the selected method parameters are in conflict with the length of the analysed NNI series, as in the case of the `dfa()` function. The implemented functions compute a total of 87 parameters of the Time Domain (23), Frequency Domain (48) and Nonlinear (7) parameters from NNI series extracted from single-lead ECG signals. Additionally, pyHRV provides a series of tools with useful features for researchers and developers of HRV applications such as ECG and Tachogram plotting, HRV results exporting, importing, and HRV report generation.

The parameter functions have been tested and evaluated in a direct comparison with the KUBIOS HRV software. For this purpose, 50 NNI series have been processed by both pyHRV and KUBIOS, which results have been compared and evaluated. The evaluation has been conducted by computing the difference between the mean of each series of parameter results and the SD of the KUBIOS results. Parameters with no differences between pyHRV and KUBIOS results have been classified as optimal parameter results. Differences lower or equal than the SD of the KUBIOS results were, although not optimal, considered acceptable, while parameter results

with more significant differences have been classified as divergent. In the latter, it means that the pyHRV function computed questionable results, which might require further investigation in future versions of the pyHRV toolbox. Overall, 12 HRV parameters computed with pyHRV have shown optimal results (identical to the KUBIOS results), with 38 parameter being within acceptable ranges and 26 parameters presenting divergent results, .

In the Time Domain, the only concern lies within the computation of the TINN parameters, where significant differences could be identified. Thorough revision of the `tinn()` function is required for future versions of pyHRV, to identify possible code segments that originate the differences and needs to be revised. On the other hand, it is necessary to identify potential differences in the KUBIOS software. For instance, pyHRV uses the 7.815 milliseconds bin size as recommended by the HRV guidelines [86]. However, in KUBIOS, the bin size being used is not specified. For instance, Figure 25 shows the histogram of a NNI series where 4 bins can be found within an approximated interval of 100 milliseconds resulting in a bin size of (approx.) 25 milliseconds, rather than the recommended 7.8125 milliseconds. With the information provided, it is unclear whether this is merely a graphical simplification of the NNI histogram or whether this does, in fact, compute the TINN parameter on different bin sizes. Besides, KUBIOS does not provide information about the number of NNI of each bin.

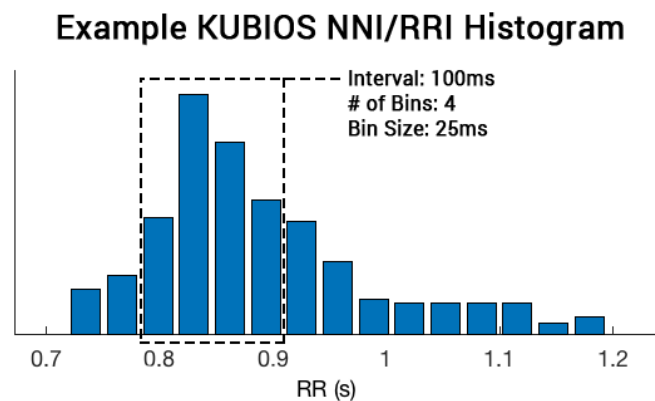


Figure 25: Example KUBIOS NNI histogram.

In the frequency domain, no adjustments are needed to the implemented Welch function (`welch_psd()`), as all parameters lie within optimal or acceptable ranges. The results of Lomb-Scargle method `lomb_psd()` differ significantly from the KUBIOS results with most of the parameters being rejected due to significant differences. For this reason, this functions requires additional development in future pyHRV versions, where the following aspects should be investigated: (1) frequency grid optimization,

(2) moving average filter application, and (3) alternative algorithms.

(1) The implementation of the Lomb-Scargle method uses SciPy's `lombscargle()` function to compute the PSD. SciPy's algorithm requires the specification of a frequency grid, i.e., an array of frequencies, for which the PSD is computed. In pyHRV, the frequency grid is generated in dependence of the selected number of PSD samples (NFFT) using the `nfft` input parameter [38]. This algorithm is susceptible to the definition of the NFFT, which can greatly alter the results. For instance, increasing the NFFT in the Welch's method increases the frequency resolution of the computed PSD, as shown in Figure 26 (a) (NFFT= 2^8) compared to Figure 26 (b) (NFFT= 2^{12}). However, for the Lomb-Scargle method, the NFFT selection has a different effect, where lower NFFT provides PSD estimations in which the dominant frequencies are more likely to be identified, as seen in Figure 26 (c), while an increase of the NFFT causes a more uniform PSD as seen in Figure 26 (d), resembling the white noise on which basis the Lomb-Scargle method is computed [49]. This behaviour must be further investigated to find a suitable NFFT value for the Lomb-Scargle method in HRV applications. In the case of the present pyHRV version, the selection of a suitable NFFT lies within the responsibility of the user.

(2) In combination with the NFFT review, it might be considered to investigate the use of moving average filters to improve the results of the Lomb-Scargle PSD, by smoothing the spectral components which can also be found in other HRV applications [66].

(3) Ultimately, in case that none of the previous methods provides reasonable results, it might be considered to investigate the use of other Lomb-Scargle algorithms for HRV specific purposes. For instance, the *AstroPy* package's `Lomb()` class has implemented methods to automatically determine the most suitable frequency grid, and has been successfully tested in HRV applications [71, 88]. However, first tests have shown that although this feature provides promising results, a noticeable difference in run-time and system resources use does exist, which needs to be improved in future work.

Although the pyHRV Autoregressive function has resulted in more rejected parameters than the Lomb-Scargle results (10 rejected parameters vs. 6), this method's results show a significant difference compared to the Lomb-Scargle results that, with other additional factors, might argue in favor of its functionality and reliability rather than the Lomb-Scargle method. As with the other methods, the majority of the Frequency Domain parameters have been computed based on the Absolute Powers of the Autoregressive PSD. Differences on these parameters have shown to

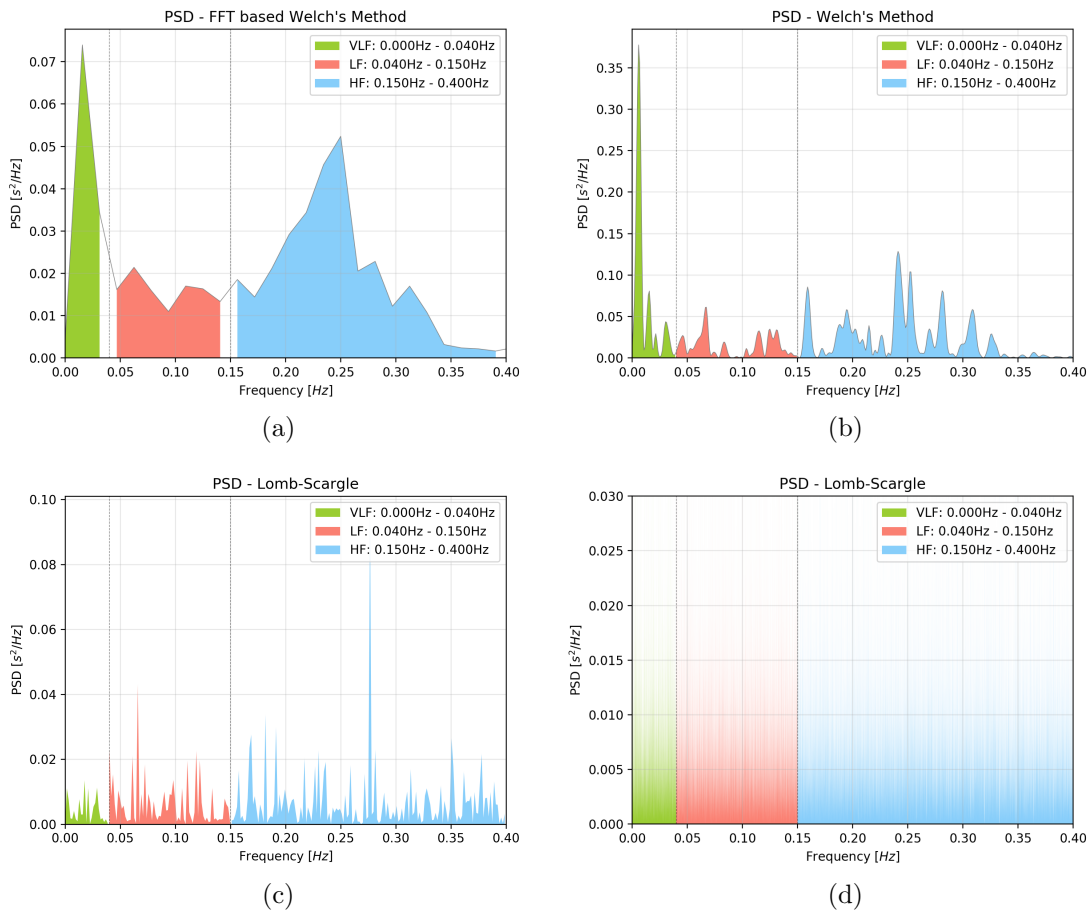


Figure 26: Comparison of the effects of NFFT selection on the PSD estimation methods (a: Welch's method with $nfft=2^8$, b: Welch's method with $nfft=2^{16}$, c: Lomb-Scargle method with $nfft=2^8$, d: Lomb-Scargle method with $nfft=2^{16}$).

be significantly higher in this method when compared to the KUBIOS results, especially for the LF and HF bands where the Absolute Powers are multiple times (4.2 and 8) greater than the KUBIOS results. However, when analysing the resulting Peak Frequencies, it is noticeable that the results are in accordance with the KUBIOS results. Given that the activity of a variety of physiological mechanisms is derived from the dominant frequencies within the individual FB, it can be argued that this method provides more reliable results than the implemented Lomb-Scargle function, which fails to accurately compute dominant frequencies in the LF band.

As for the differences in the Absolute Powers, the following aspects are assumed to be the cause of this effect. First, various models exist for AR PSD estimations, which may differ in their fundamental computational methods (e.g., Burg, Yule). KUBIOS does not specify the method being used for its implemented PSD estimation which makes it difficult to ensure an entirely fair comparison. Second, while KUBIOS allows the user to set the specifications of the moving average filter being applied

to this method, the `spectrum.pyule()` function comes with built-in functions that automatically determine the optimal moving average filter specifications. These differences may be the leading causes of the increased PSD estimation. However, it is essential to acknowledge the usefulness of the pyHRV method, given that the dominant frequencies and alterations can still be identified, thus, being a reliable tool for HRV applications.

The comparison of the Nonlinear parameters have shown positive results, with 6 out of 7 parameters showing acceptable results, and only marginal errors found when compared to the KUBIOS results, and the SD2/SD1 ratio achieving even optimal results. The functions of this module do not require any further optimization.

5.2 Future Work

In addition to the results discussed in the previous section, which require further revision and improvements in some parameter computation functions of the Time and Frequency Domains, the developed pyHRV toolbox will experience continued development with the implementation of new HRV methods and features in the future. Also, on a programming side, general improvements should be conducted such as adding full support for the Python 3 programming language. On a HRV feature extraction and method implementation side, the state-of-the-art methods described next can be implemented, which have been proposed by HRV researchers over the recent years.

Björkander et al. [9] have proposed a new geometrical Time Domain parameter named Differential Index, which is computed based on the same NNI histogram as shown in Figure 27. This parameter is computed from long-term ECG recordings and requires no pre-filtering of the NNI series. For this parameter, the width of the histogram at 10 000 (R1) and 1000 (R2) NNI intervals is measured, and a triangular function is computed. The Differential Index is the baseline of the computed triangular function and is an index for overall HRV, with increasing baseline width presenting higher HRV.

For the Frequency Domain, German-Zallo [30] has investigated the use of a Wavelet Package Transform as an alternative method to the FFT based PSD estimation method, to measure the power variance and to identify dominant frequencies in a NNI series. His approach has shown good time-frequency resolution and comparable results for the LF/HF ratio, however, it still requires further investigation. Adding

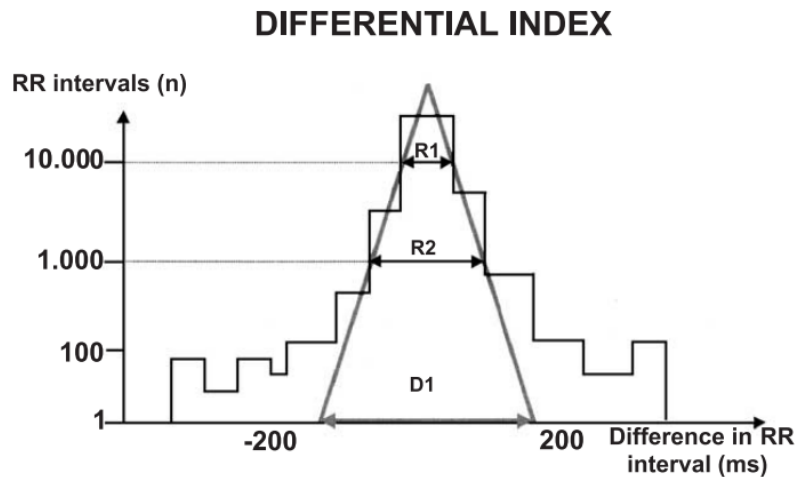


Figure 27: Illustration of the Differential Index computation [9].

this method to pyHRV would provide the tools for researchers to investigate this method in further detail. Another interesting method would be the implementation of adaptive frequency bands, as proposed by Long et al. [50], to increase the accuracy of identifying dominant HR regulation mechanisms in sleep and wake states. Here, the LF and HF frequency bands are adjusted based on time-frequency analysis of the NNI series, to compensate the time-varying behavior of the dominant frequencies in sleep vs. wake states.

Costa et al. [21] have developed and proposed a new method to quantify the complexity of physiologic time series, the Multiscale Entropy. Traditional Nonlinear entropy methods (e.g. SamPen) can cause misleading results when applied on ECG recordings with pathological cardiac arrhythmia, where higher entropy values can be found compared to healthy individuals even when the underlying HR regulation systems show no sign of pathological influence. The Multiscale Entropy method analyses the entropy of a NNI series based on multiple time scales, thus avoiding the misleading characteristics of the traditional methods, and can now also be found in other fields of research (e.g. analysis of EEG signals) for which pyHRV might also be of added value [5, 16, 22, 23, 36]

Another interesting option, the implementation of existing and recent HRV methods, could be the exploration and development of new methods that could be of additional value for the pyHRV toolbox. For instance, one possibility could be to focus on variations of commonly used parameters or methods that have shown to be easier to interpret such as the Poincaré plot [33, 56, 59, 94]. The Poincaré plot is one of the most comprehensive geometrical methods based on a NNI_j vs. NNI_{j+1} scatter plot. The form of the scatter plot has been thoroughly investigated and its importance for

the identification of healthy (high) or pathological (low) HRV is widely recognized [102]. However, little to no research has yet been conducted to investigate scatter plots of other intervals (e.g., NNI_j vs. NNI_{j+2} , NNI_j vs. NNI_{j+3}) and their potential capability to identify different scatter plot patterns as results of different cardiac conditions.

Bibliography

- [1] Lohninger Alfred. *Herzratenvariabilität - Das HRV-Praxis-Lehrbuch*. Vienna: facultas Universitätsverlag, 2017.
- [2] American Heart Association. *All About Heart Rate (Pulse)*. 2018. URL: <http://www.heart.org/en/health-topics/high-blood-pressure/the-facts-about-high-blood-pressure/all-about-heart-rate-pulse> (visited on 06/17/2018).
- [3] Bradley Appelhans and Linda Luecken. “Heart rate variability as an index of regulated emotional responding”. In: *Review of General Psychology* 10.3 (2006), pp. 229–240.
- [4] Hyun Baek et al. “Reliability of Ultra-Short-Term Analysis as a Surrogate of Standard 5-Min Analysis of Heart Rate Variability”. In: *Telemedicine and e-Health* 21.5 (2015), pp. 404–414.
- [5] Heiko Balzter et al. “Multi-Scale Entropy Analysis as a Method for Time-Series Analysis of Climate Data”. In: *Climate* 3.1 (2015), pp. 227–240.
- [6] Rhenan Bartels. *Heart Rate Analysis Package*. 2016. URL: <https://github.com/rhenanbartels/hrv> (visited on 08/20/2018).
- [7] Diana Batista, Hugo Silva, and Ana Fred. “Experimental characterization and analysis of the BITalino platforms against a reference device”. In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2017), pp. 2418–2421.
- [8] George Billman. “Heart rate variability - A historical perspective”. In: *Frontiers in Physiology* 2 NOV.November (2011), pp. 1–13.
- [9] Inge Björkander et al. “Differential index, a novel graphical method for measurements of heart rate variability”. In: *International Journal of Cardiology* 98.3 (2005), pp. 493–499.
- [10] Anita Boardman et al. “A study on the optimum order of autoregressive models for heart rate variability”. In: *Physiological Measurement* 23.2 (2002), pp. 325–336.

-
- [11] Annette Bopp. *Handbuch Herz und Kreislauf - Vorbeugen, erkennen, heilen*. 2nd ed. Berlin: Stiftung Warentest, 1996, pp. 13–21.
- [12] M. Brennan, M. Palaniswami, and P. Kamen. “Do existing measures of Poincaré plot geometry reflect nonlinear features of heart rate variability?” In: *IEEE Transactions on Biomedical Engineering* 48.11 (2001), pp. 1342–1347.
- [13] Lewis Brown and Shivaram Arunachalam. “Real-time T-P knot algorithm for baseline wander noise removal from the electrocardiogram”. In: *Biomedical Sciences Instrumentation* 45.April (2009), pp. 65–70.
- [14] Robert Burr. “Interpretation of Normalized Spectral Heart Rate Variability Indices In Sleep Research: A Critical Review”. In: *NeuroImage* 9.2 (2014), pp. 1–6.
- [15] Robert L. Burr and Marie J. Cowan. “Autoregressive spectral models of heart rate variability. Practical issues”. In: *Journal of Electrocardiology* 25.SUPPL. (1992), pp. 224–233.
- [16] Michael A. Busa and Richard E.A. van Emmerik. “Multiscale entropy: A tool for understanding the complexity of postural control”. In: *Journal of Sport and Health Science* 5.1 (2016), pp. 44–51.
- [17] Carlos Carreiras et al. *BioSPPy - Biosignal Processing in Python*. URL: <https://github.com/PIA-Group/BioSPPy/> (visited on 07/15/2018).
- [18] Carlos Carreiras et al. “Comparative Study of Medical-grade and Off-the-Person ECG Systems”. In: *Proceedings of the International Congress on Cardiovascular Technologies* (2013), pp. 115–120.
- [19] João de Carvalho et al. “Development of a Matlab software for analysis of heart rate variability”. In: *6th International Conference on Signal Processing, 2002*. 2.May (2014), pp. 1488–1491.
- [20] Thomas Cokelaer and Juergen Hasch. “‘Spectrum’: Spectral Analysis in Python”. In: *The Journal of Open Source Software* 2.18 (2017), p. 348.
- [21] Madalena Costa, Ary Goldberger, and C.-K. Peng. “Multiscale Entropy Analysis of Complex Physiologic Time Series”. In: *Physical Review Letters* 92.8 (2004), pp. 6–9.
- [22] Madalena Costa, Ary L. Goldberger, and C. K. Peng. “Multiscale entropy analysis of biological signals”. In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 71.2 (2005), pp. 1–18.
- [23] Julie Courtiol et al. “The multiscale entropy: Guidelines for use and interpretation in brain signal analysis”. In: *Journal of Neuroscience Methods* 273 (2016), pp. 175–190.

-
- [24] Ilaria Coviello et al. “Prognostic Role of Heart Rate Variability in Patients with ST-Segment Elevation Acute Myocardial Infarction Treated by Primary Angioplasty”. In: *Cardiology* 124.1 (2013), pp. 63–70.
- [25] Marianne P.C. de Rezende Barbosa et al. “Effects of functional training on geometric indices of heart rate variability”. In: *Journal of Sport and Health Science* 5.2 (2016), pp. 183–189.
- [26] David Ewing, James Neilson, and Paul Travis. “New method for assessing cardiac parasympathetic activity using 24 hour electrocardiograms”. In: *British Heart Journal* 52.4 (1984), pp. 396–402.
- [27] D. S. Fonseca et al. “Lomb-scargle periodogram applied to heart rate variability study”. In: *ISSNIP Biosignals and Biorobotics Conference, BRC 2* (2013), pp. 8–11.
- [28] Johnson Francis. “ECG monitoring leads and special leads”. In: *Indian Pacing and Electrophysiology Journal* 16.3 (2016), pp. 92–95.
- [29] Georg Thieme Verlag. *EKG: Verlauf der EKG-Kurve und Vektorschleife*. 2018. URL: <https://viamedici.thieme.de/lernmodule/physiologie/ekg+verlauf+der+ekg-kurve+und+vektorschleife> (visited on 10/04/2018).
- [30] Zoltan Germán-Salló. “Wavelet based HRV analysis”. In: *IFMBE Proceedings* 44 (2014), pp. 229–232.
- [31] Ary Goldberger et al. “The MIT-BIH Normal Sinus Rhythm Database - PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.” In: *Circulation* 101 23.June 13 (2000), pp. 215–220.
- [32] Nathalia Gomes and Arthur Sá. “Pulse Waveform Analysis of Chinese Pulse Images and Its Association with Disability in Hypertension”. In: *JAMS Journal of Acupuncture and Meridian Studies* 9.2 (2016), pp. 93–98.
- [33] Evgeniya Gospodinova et al. “Analysis of Heart Rate Variability by Applying Nonlinear Methods with Different Approaches for Graphical Representation of Results”. In: *International Journal of Advanced Computer Science and Applications* 6.8 (2015), pp. 38–45.
- [34] Clinton Goss and Eric Miller. “Dynamic Metrics of Heart Rate Variability”. In: (2013), pp. 1–4.
- [35] Laszlo Hejje and Elizabeth Roth. “What is the adequate sampling interval of the ECG signal for heart rate variability analysis in the time domain?” In: *Physiological Measurement* 25.6 (2004), pp. 1405–1411.
- [36] Anne Humeau-Heurtier. “The multiscale entropy algorithm and its variants: A review”. In: *Entropy* 17.5 (2015), pp. 3110–3123.

-
- [37] John D. Hunter. “Matplotlib - A 2D graphics environment”. In: *Computing in Science and Engineering* 9.3 (2007), pp. 99–104.
- [38] Eric Jones et al. *SciPy: Open source scientific tools for Python*. 2001. URL: <http://www.scipy.org/>.
- [39] D L Justis and W T Hession. “Accuracy of 22-lead ECG analysis for diagnosis of acute myocardial infarction and coronary artery disease in the emergency department: a comparison with 12-lead ECG”. In: *Ann Emerg Med* 21.1 (1992), pp. 1–9.
- [40] Maya Kallas et al. “Prediction of time series using Yule-Walker equations with kernels”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 1 (2012), pp. 2185–2188.
- [41] Ulrich Karrenberg. *Signale - Prozessen - Systeme: Eine multimediale und interaktive Einführung in die Signalverarbeitung*. 7th. Düsseldorf: Springer Vieweg, 2017, pp. 525–529.
- [42] Tom Kuusela. “Methodological Aspects of Heart Rate Variability Analysis”. In: *Heart Rate Variability (HRV) Signal Analysis: Clinical Applications* (2013), pp. 9–42.
- [43] Pablo Laguna, George B Moody, and Roger G Mark. “Power spectral density of unevenly sampled data by least-square analysis: Performance and application to heart rate signals”. In: *IEEE Transactions on Biomedical Engineering* 45.6 (1998), pp. 698–715.
- [44] Pablo Laguna et al. “Power Spectral Density of Unevenly Sampled Heart Rate Data”. In: (1995), pp. 157–158.
- [45] Rachel Lampert et al. “Decreased heart rate variability is associated with higher levels of inflammation in middle-aged men”. In: *American Heart Journal* 156.4 (2008), 759.e1–7.
- [46] Imad Libbus et al. “Quantitative evaluation of heartbeat interval time series using Poincaré analysis reveals distinct patterns of heart rate dynamics during cycles of vagus nerve stimulation in patients with heart failure”. In: *Journal of Electrocardiology* 50.6 (2017), pp. 898–903.
- [47] Studies I N Logic. “Poincaré Plots in Analysis of Selected Biomedical Signals”. In: *Studies in Logic, Grammar and Rhetoric* 35.48 (2013), pp. 117–127.
- [48] Herbert Löllgen. “Neue Methoden in der kardialen Funktionsdiagnostik”. In: *Deutsches Ärzteblatt* 45.31 (1999), pp. 2029–32.
- [49] N. R. Lomb. “Least-Squares Frequency Analysis of Unequally Spaced Data”. In: *Astrophysics and Space Science* 39 (1976), pp. 447–462.

-
- [50] Xi Long et al. “Time-frequency analysis of heart rate variability for sleep and wake classification”. In: *IEEE 12th International Conference on Bioinformatics and BioEngineering, BIBE 2012* (2012), pp. 85–90.
- [51] Jürgen Lorenz et al. “Subjective and Autonomic Responses to a 3D-Virtual Reality Game”. In: *Proceedings of the 11th European Conference on Games Based Learning, ECGBL 2017*. Graz, 2017, pp. 387–393.
- [52] Shadi Mahdiani et al. “Is 50 Hz high enough ECG sampling frequency for accurate HRV analysis?” In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2015-Novem* (2015), pp. 5948–5951.
- [53] Numpydoc Maintainers. *numpydoc docstring guide*. 2018. URL: <https://numpydoc.readthedocs.io/en/latest/format.html> (visited on 05/20/2018).
- [54] Peter Maud and Carl Foster. *Physiological Assessment of Human Fitness*. Champaign: Human Kinetics, 2006, pp. 39–62.
- [55] Rollin McCraty. *Science of the Heart - Exploring the Role of the Heart in Human Performance*. Vol. 2. Boulder Creek: HeartMath Institute, 2015.
- [56] Paolo Melillo, Marcello Bracale, and Leandro Pecchia. “Nonlinear Heart Rate Variability features for real-life stress detection. Case study: students under stress due to university examination”. In: *Biomedical Engineering Online* 10.96 (2011).
- [57] J. Mietus et al. “The pNNx files: re-examining a widely used heart rate variability measure”. In: *Heart* 88.4 (2002), pp. 378–380.
- [58] Eduardo Miranda Dantas et al. “Spectral analysis of heart rate variability with the autoregressive method: What model order to choose?” In: *Computers in Biology and Medicine* 42.2 (2012), pp. 164–170.
- [59] Ș. Mirescu and S. W. Harden. “Nonlinear Dynamics Methods for Assessing Heart Rate Variability in Patients with Recent Myocardial Infraction”. In: *Romanian J. Biophys.* 22.2 (2012), pp. 117–124.
- [60] Kosuke Miyoshi. *Heart Rate Variability (HRV) Analysis Tool with Polar H7/H6 on MacOSX*. 2016. URL: <https://github.com/miyosuda/heart-rate-monitor> (visited on 08/20/2018).
- [61] George B Moody. “Spectral Analysis of Heart Rate Without Resampling”. In: (1993), pp. 715–718.
- [62] A. Muthuchudar and S. Santosh Baboo. “A Study of the Processes Involved in ECG Signal Analysis”. In: *International Journal of Scientific and Research Publications* 3.1 (2013), pp. 2250–3153.

-
- [63] Dib Nabil and F. Bereksi Reguig. “Ectopic beats detection and correction methods: A review”. In: *Biomedical Signal Processing and Control* 18 (2015), pp. 228–244.
- [64] Jose Naranjo et al. “Heart Rate Variability: a Follow-up in Elite Soccer Players Throughout the Season”. In: *International Journal of Sports Medicine* 36.11 (2015), pp. 881–886.
- [65] Juha Niskanen et al. “Software for advanced HRV analysis”. In: *Computer Methods and Programs in Biomedicine* 76.1 (2004), pp. 73–81.
- [66] Juha-pekka Niskanen and Perttu O Ranta-aho. *Kubios User Manual*. 2017.
- [67] Travis Oliphant. *A guide to NumPy*. USA: Trelgol Publishing, 2006.
- [68] Jiapu Pan and Willis J. Tompkins. “A Real-Time QRS Detection Algorithm”. In: *IEEE Transactions on Biomedical Engineering* BME-32.3 (1985), pp. 230–236.
- [69] Mirja Peltola. “Role of editing of R-R intervals in the analysis of heart rate variability”. In: *Frontiers in Physiology* 3.May (2012), pp. 1–10.
- [70] C. K. Peng et al. “Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series”. In: *Chaos* 5.1 (1995), pp. 82–87.
- [71] Guilherme Ramos, Miquel Alfaras, and Hugo Gamboa. “Real-Time Approach to HRV Analysis”. In: *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies*. Vol. 4. Biostec. 2018, pp. 208–215.
- [72] John Ramshur. “Design, Evaluation, and Application of Heart Rate Variability Analysis Software (HRVAS)”. Master Thesis. University of Memphis, 2010.
- [73] Joshua S Richman et al. “Physiological time-series analysis using approximate entropy and sample entropy”. In: *Am J Physiol Hear Circ Physiol* (2008), pp. 2039–2049.
- [74] L. Rodríguez-Liñares et al. “An open source tool for heart rate variability spectral analysis”. In: *Computer Methods and Programs in Biomedicine* 103.1 (2011), pp. 39–50.
- [75] Lizawati Salahuddin et al. “Ultra Short Term Analysis of Heart Rate Variability for Monitoring Mental Stress in Mobile Settings”. In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2007, pp. 4656–4659.

-
- [76] Kai Savonen et al. “Heart rate response during exercise test and cardiovascular mortality in middle-aged men”. In: *European Heart Journal* 27.5 (2006), pp. 582–588.
- [77] J. D. Scargle. “Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data”. In: *The Astrophysical Journal* 263 (1982), p. 835. arXiv: 0003129v2 [hep-ph].
- [78] C. Schoel. *NOnLinear measures for Dynamical Systems (nolds)*. 2016. URL: <https://github.com/CSchoel/nolds{\#}nonlinear-measures-for-dynamical-systems-nolds> (visited on 10/02/2018).
- [79] Felix Scholkmann, Jens Boss, and Martin Wolf. “An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals”. In: *Algorithms* 5.4 (2012), pp. 588–603.
- [80] Home Search et al. “Statistics and graphs for heart rate variability :” in: *Physiological Measurement* 24 (2003), pp. 8–14.
- [81] P Seyd et al. “Time and frequency domain analysis of heart rate variability and their correlations in diabetes mellitus”. In: *International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering* 2.3 (2008), pp. 85–88.
- [82] Fred Shaffer and J. P. Ginsberg. “An Overview of Heart Rate Variability Metrics and Norms”. In: *Frontiers in Public Health* 5.September (2017), pp. 1–17.
- [83] Fred Shaffer, Rollin McCraty, and Christopher L. Zerr. “A healthy heart is not a metronome: an integrative review of the heart’s anatomy and heart rate variability”. In: *Frontiers in Psychology* 5.September (2014), pp. 1–19.
- [84] Naiara Maria Souza et al. “Geometric indexes of HRV in type 1 diabetes”. In: *Curr Res Cardiol* 3.2 (2016), pp. 38–42.
- [85] Umberto Spagnolini. *Statistical Signal Processing in Engineering*. John Wiley & Sons Ltd., 2017, p. 267.
- [86] Task Force of the European Society of Cardiology and the North American Society of Pacing And Electrophysiology. “Heart Rate Variability - Standards of measurement, physiological interpretation, and clinical use”. In: *European Heart Journal* 17 (1996), pp. 354–381.
- [87] Mazhar B Tayel and Eslam I Alsaba. “Poincaré Plot for Heart Rate Variability”. In: *International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering* 9.9 (2015), pp. 708–711.

-
- [88] The Astropy Collaboration et al. “The Astropy Project: Building an inclusive, open-science project and status of the v2.0 core package”. In: *ArXiv e-prints* (2018).
- [89] The Student Physiologist. *The ECG Leads Polarity And Einthovens-Triangle*. 2017. (Visited on 10/05/2018).
- [90] Tran Thong et al. “Accuracy of ultra-short heart rate variability measures”. In: *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*. EMBC 2003. 2003, pp. 2424–2427.
- [91] Boris Tolg et al. “Re-test reliability of subjective and autonomic responses in a VR performance-anxiety test”. In: *Proceedings of the 11th European Conference on Games Based Learning, ECGBL 2017*. Graz, 2017, pp. 671–676.
- [92] Ken Umetani et al. “Twenty-Four Hour Time Domain Heart Rate Variability and Heart Rate : Relations to Age and Gender Over Nine Decades”. In: *Journal of the American College of Cardiology* 31.3 (1998), pp. 593–601.
- [93] Merete Vaage-Nilsen et al. “Recovery of autonomic nervous activity after myocardial infarction demonstrated by short-term measurements of SDNN”. In: *Scandinavian Cardiovascular Journal* 35.3 (2001), pp. 186–191.
- [94] Luiz Carlos Marques Vanderlei et al. “Geometric indexes of heart rate variability in obese and eutrophic children.” In: *Arquivos brasileiros de cardiologia* 95.1 (2010), pp. 35–40.
- [95] Adriana Nicholson Vest et al. “An open source benchmarked toolbox for cardiovascular waveform and interval analysis”. In: *Physiological Measurement* (2018).
- [96] Marcus Vollmer. “A robust, simple and reliable measure of heart rate variability using relative RR intervals”. In: *Computing in Cardiology* 42.6 (2016), pp. 609–612.
- [97] Federico Wadehn, Yue Zhao, and Hans-andrea Loeliger. “Heart Rate Estimation in Photoplethysmogram Signals using Nonlinear Model-Based Pre-processing”. In: *Computing in Cardiology* 42 (2015), pp. 633–636.
- [98] Hui Min Wang and Sheng Chieh Huang. “SDNN/RMSSD as a surrogate for LF/HF: A revised investigation”. In: *Modelling and Simulation in Engineering 2012* (2012).
- [99] Jim Waterhouse, Yumi Fukuda, and Takeshi Morita. “Daily rhythms of the sleep-wake cycle”. In: *Journal of Physiological Anthropology* 31.1 (2012), pp. 1–14.

- [100] Peter Welch. “The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms”. In: *IEEE Transaction on Audio and Electroacoustics* AU-15.2 (1967), pp. 70–73.
- [101] Keith Willson et al. “Relationship between detrended fluctuation analysis and spectral analysis of heart-rate variability”. In: *Physiological Measurement* 23.2 (2002), pp. 385–401.
- [102] Mary Woo et al. “Patterns of beat-to-beat advanced heart failure heart rate variability in”. In: *American Heart Journal* 123.3 (1992), pp. 704–710.
- [103] F Yasuma and J Hayano. “Respiratory Sinus Arrhythmia - Why Does the Heartbeat Synchronize With Respiratory Rhythm?” In: *CHEST* 125.2 (2004), pp. 683–690.

Appendices

Appendix A

Sample HRV Report

SampleReport.txt

```
# BIOSPPY HRV REPORT - v.0.2.5
# Sampling Rate      ;          1000Hz
# Date Time          ; 2018-11-18_18-48-17
# File               ; SampleFile
# Device             ; BITalino revolution Board
# Identifier/MAC     ; A0:B1:C2:D3:F4:H5
# Resolution         ; 10-bit
```

COMMENTS

This is a sample report.

NN INTERVALS ms ; n/a

Time Domain

```
Number of NNIs - ; 4684.000
Mean NNI ms      ; 768.438
Minimum NNI ms   ; 562.000
Maximum NNI ms   ; 1188.000
Mean Delta-NNI ms ; 42.199
Minimum Delta-NNI ms ; 0.000
Maximum Delta-NNI ms ; 360.000
Mean Heart Rate bpm ; 78.990
Minimum Heart Rate bpm ; 50.505
```


A. Sample HRV Report

Maximum Heart Rate bpm	; 106.762
Standard Deviation of Heart Rate bpm	; 8.305
SDNN ms	; 85.357
SDNN Index ms	; 82.586
SDANN ms	; 22.347
SDSD ms	; 43.391
RMSSD ms	; 60.523
NN50 -	; 1338.000
pNN50 %	; 28.571
NN20 -	; 3008.000
pNN20 %	; 64.232
Triangular Index -	; 11.509
TINN ms	; 0.000
TINN N ms	; 0.000
TINN M ms	; 0.000

Frequency Domain - FFT Welch's Method

Frequency Bands Hz	;
- None	
- 0.0, 0.04	
- 0.04, 0.15	
- 0.15, 0.4	
Peak Frequencies Hz	;
- 0.0185546875	
- 0.0400390625	
- 0.150390625	
Absolute Powers Hz	;
- 2050.743568530359	
- 2942.537244842369	
- 1648.0967694750875	
Logarithmic Powers Hz	;
- 7.62595772272	
- 7.9870274966	
- 7.40737642809	
Relative Powers %	;
- 30.878286062618386	
- 44.306127879882	
- 24.815586057499615	
Normalized Powers -	;
- 64.09871132538694	
- 35.901288674613056	
Total Power ms ²	; 6641.378

LF/HF Ratio - ; 1.785
 Number of PSD Samples - ; 4096.000
 Window Function - ; hamming
 Resampling Frequency Hz ; 4.000
 Interpolation Method - ; cubic

 Frequency Domain - Autoregression Method

Frequency Bands Hz ;
 - None
 - 0.0, 0.04
 - 0.04, 0.15
 - 0.15, 0.4
 Peak Frequencies Hz ;
 - 0.0
 - 0.0400390625
 - 0.150390625
 Absolute Powers ms² ;
 - 2779.1536684617004
 - 6175.5637927489115
 - 12477.064306903205
 Logarithmic Powers Hz ;
 - 7.92990172453
 - 8.7283554598
 - 9.43164738243
 Relative Powers % ;
 - 12.967441058010968
 - 28.814980758794906
 - 58.21757818319412
 Normalized Powers - ;
 - 33.10827707364245
 - 66.89172292635755
 Total Power ms² ; 21431.782
 LF/HF Ratio - ; 0.495
 Number of PSD Samples - ; 4096.000
 AR Order - ; 16.000

 Frequency Domain - Lomb-Scargle Method

Frequency Bands Hz ;
 - None

- 0.0, 0.04
 - 0.04, 0.15
 - 0.15, 0.4
 Peak frequencies Hz ;
 - 0.02250980392156863
 - 0.04341176470588235
 - 0.2347450980392157
 Absolute powers Hz ;
 - 15.068433975736726
 - 46.19183177960316
 - 114.84694635936958
 Logarithmic powers Hz ;
 - 2.71260209057
 - 3.83280298115
 - 4.7436003407
 Relative powers % ;
 - 8.556398000282764
 - 26.22938108265298
 - 65.21422091706425
 Normalized powers - ;
 - 28.683670053520075
 - 71.31632994647991
 Total power ms² ; 176.107
 LF/HF ratio - ; 0.402
 Number of PSD Samples - ; 256.000
 Moving Average Window Size - ; n/a

 Nonlinear Methods

SD1 ms ; 42.797
 SD2 ms ; 112.837
 SD2/SD1 - ; 2.637
 Ellipse Area S ms² ; 15170.898
 Sample Entropy - ; 1.250
 DFA alpha 1 long term fluctuation - ; 1.197
 DFA alpha 2 long term fluctuation - ; 0.882
 None None ; n/a

Appendix B

Parameter Keys & Report Descriptions

Table A1: Time domain parameter keys and report descriptions.

Time Domain Keys & Descriptions		
Parameter	Key	Report Description
NNI	nn_intervals	NN INTERVALS
$\#ofNNI$	nni_counter	Number of NN Intervals [-]
NN_{mean}	nni_mean	Mean NN Interval [ms]
NN_{min}	nni_min	Minimum NN Interval [ms]
NN_{max}	nni_max	Maximum NN Interval Difference [ms]
ΔNN_{mean}	nni_diff_mean	Mean NN Interval [ms]
ΔNN_{min}	nni_diff_min	Minimum NN Interval [ms]
ΔNN_{max}	nni_diff_max	Maximum NN Interval Difference [ms]
HR_{mean}	hr_mean	Mean Heart Rate [BPM]
HR_{min}	hr_min	Minimum Heart Rate [BPM]
HR_{max}	hr_max	Maximum Heart Rate [BPM]
HR_{std}	hr_std	Standard deviation of the HR [BPM]
$SDNN$	sdnn	SDNN [ms]
$SDNN_{Index}$	sdnn_index	SDNN Index [ms]
$SDANN$	sdann	SDANN [ms]
$SDSD$	sdsd	SDSD [ms]
$RMSSD$	rmssd	RMSSD [ms]
$NN50$	nn50	NN50 [-]
$pNN50$	pnn50	pNN50 [%]
$NN20$	nn20	NN20 [-]
$pNN20$	pnn20	pNN20 [%]
$TriIndex$	tri_index	Triangular index [-]
$TINN$	tinn	TINN [ms]
N	tinn_n	TINN N [ms]
M	tinn_m	TINN M [ms]

Table A2: Frequency domain parameter keys and report descriptions.

Frequency Domain Keys & Descriptions		
Parameter	Key	Report Description
$f - Bands$	X_bands	Frequency bands (Hz)
$Peakf$	X_peak	Peak frequencies (Hz)
$AbsolutePowers$	X_abs	Absolute powers [ms]
$log(Powers)$	X_log	Logarithmic powers (log)
$RelativePowers$	X_rel	Relative powers[%]
$NormalizedPower$	X_norm	Normalized powers [-]
$TotalPower$	X_total	Total power [ms]
LF/HF	X_ratio	LF/HF ratio [-]

With X = 'fft' for FFT based frequency methods, 'ar' for Autoregression methods, or 'lomb' for Lomb-Scargle methods.

Table A3: Nonlinear parameter keys and report descriptions.

Nonlinear Keys & Descriptions		
Parameter	Key	Report Description
<i>SD1</i>	sd1	SD1 [<i>ms</i>]
<i>SD2</i>	sd2	SD2 [<i>ms</i>]
<i>SD2/SD1</i>	sd_ratio	SD2/SD1 ratio [–]
<i>EllipseArea</i>	ellipse_area	Ellipse Area S [<i>ms</i> ²]
<i>SampleEntropy</i>	sampen	Sample Entropy [–]
α_1	dfa_short	DFA alpha 1 (short-term fluctuation) [–]
α_2	dfa_long	DFA alpha 2 (long-term fluctuation) [–]

Table A4: Plot keys.

pyHRV plot figure keys.	
Plot	Key
<i>ECG</i>	ecg_plot
<i>Tachogram</i>	tachogram_plot
<i>Poincare</i>	poincare_plot
<i>WelchPSD</i>	fft_plot
<i>ARPSD</i>	ar_plot
<i>LombPSD</i>	lomb_plot
<i>NNIHistogram</i>	nni_histogram
<i>TINNHistogram</i>	tinn_histogram
<i>TriIndexHistogram</i>	tri_histogram

Appendix C

Developed Support Packages

C.1 OpenSignalsReader Package

The *OpenSignalsReader* package is designed to facilitate the import of sensor signals acquired with *BITalino (r)evolution* using PLUX's *OpenSignals (r)evolution* software into simple Python data containers. These containers consist of nested *NumPy* arrays, lists, and dictionaries to store sensor signals and acquisition metadata (e.g., device type, sampling frequency, sampling resolution), which are ultimately stored in an `OpenSignalsReader()` object. Imported signals are converted by default into their original units using the transfer functions implemented in the `bitalino_transfer_function.py` module taken from the sensor-specific datasheets¹.

In this work, this package is only used to import ECG signals acquired with BITalino devices for testing purposes of pyHRV and to provide application examples using ECG acquisition hardware, which can be found in the API reference.

The open-source *OpenSignalsReader* package has been made available GitHub and can be found under the following link:

<https://github.com/PGomes92/opensignalsreader>

¹BITalino Datasheets: <http://bitalino.com/en/learn/documentation>

Python Snippet 7 demonstrates a minimum working example of the use of this package and the *OpenSignalsReader* class.

```
1 # Import libraries
2 from opensignalsreader import OpenSignalsReader
3
4 # Read OpenSignals acquisition file & plot all signals
5 acq = OpenSignalsReader('SampleECG.txt', show=True)
6
7 # Plot ECG signal only
8 acq.plot('ECG')
9
10 # Access ECG signal
11 acq.signals('ECG')
```

Python Snippet 7: Minimum working example of the *OpenSignalsReader* package and class to import *BITalino (r)evolution* sensor data stored in *OpenSignals (r)evolution* file.

C.2 Kubios Package

The *kubios* package is designed to facilitate the export of NNI series in KUBIOS-friendly .TXT files and to import HRV parameter results computed with KUBIOS and stored in KUBIOS .TXT report files.

In this work, this package has been used within the evaluation process, to read KUBIOS results stored in the KUBIOS report files.

The open-source *kubios* package has been made available on GitHub and can be found under the following link:

<https://github.com/PGomes92/kubios>

Appendix D

BITalino (r)evolution Board Datasheet

(Source: http://bitalino.com/datasheets/REVOLUTION_BITalino_Board_Kit_Datasheet.pdf)

BITalino (r)evolution Board Kit Data Sheet

BBK 160616

SPECIFICATIONS

- > **Sampling Rate:** 1, 10, 100 or 1000Hz
- > **Analog Ports:** 4 in (10-bit) + 2 in (6-bit) + 1 auxiliary in (battery) + 1 out (8-bit)
- > **Digital Ports:** 2 in (1-bit) + 2 out (1-bit)
- > **Communication:** Bluetooth or BLE
- > **Range:** up to ~10m (in line of sight)
- > **Sensors:** EMG; ECG; EDA; EEG; ACC; LUX; BTN
- > **Actuators:** LED; BUZ
- > **Size:** 100x65x6mm
- > **Battery:** 500mA 3.7V LiPo (rechargeable)
- > **Consumption:** ~65mA
- > **Accessories:** 1x 3-lead cable; 1x 2-lead cable; 5x Electrodes; 1x ProtoBIT

FEATURES

- > All-in-one ready-to-use design
- > Snappable blocks
- > Raw data acquisition
- > On-board battery charger
- > Easy-to-use
- > Affordable

APPLICATIONS

- > Psychophysiology
- > Biomedical projects
- > Electrical engineering
- > Human-Computer Interaction
- > Robotics & Cybernetics
- > Physiology studies
- > Biofeedback

GENERAL DESCRIPTION

Our signature BITalino (r)evolution Board kits have an all-in-one hardware design, with all the blocks pre-connected between them and ready-to-use, making it perfect for biosignal exploration and lab activities. The kit includes all the basic accessories needed to get started, namely the hardware modules, battery, cables and electrodes. Along with our cross-platform OpenSignals software, it enables instant biosignal data visualization and recording out-of-the-box.

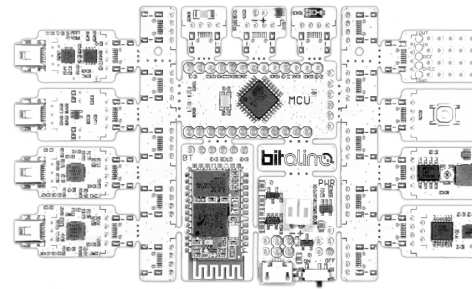


Fig. 1. All-in-one design ready to use out-of-the-box; available with Bluetooth (Fig. 4) or BLE (Fig. 5).

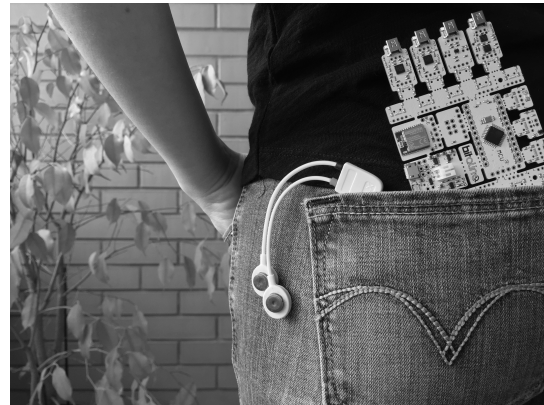


Fig. 2. The BITalino (r)evolution Board kits enable you to have your own personal biomedical research system.

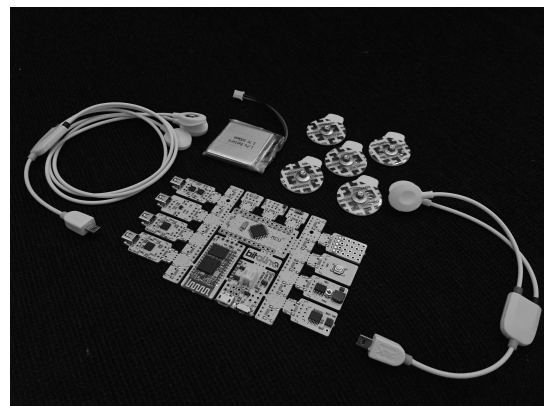


Fig. 3. Parts and accessories included in the kit.

bitalino

PLUX – Wireless Biosignals, S.A.
Av. 5 de Outubro, n. 70 – 8.
1050-059 Lisbon, Portugal
bitalino@plux.info
<http://bitalino.com/>

REV A

© 2016 PLUX 

This information is provided "as is," and we make no express or implied warranties whatsoever with respect to functionality, operability, use, fitness for a particular purpose, or infringement of rights. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.



BEWARE: DIRECT OR INDIRECT COUPLING TO THE MAINS MAY RESULT IN SHOCKING HAZARD



BITalino (r)evolution Board Kit Data Sheet

FUNCTIONAL BLOCKS

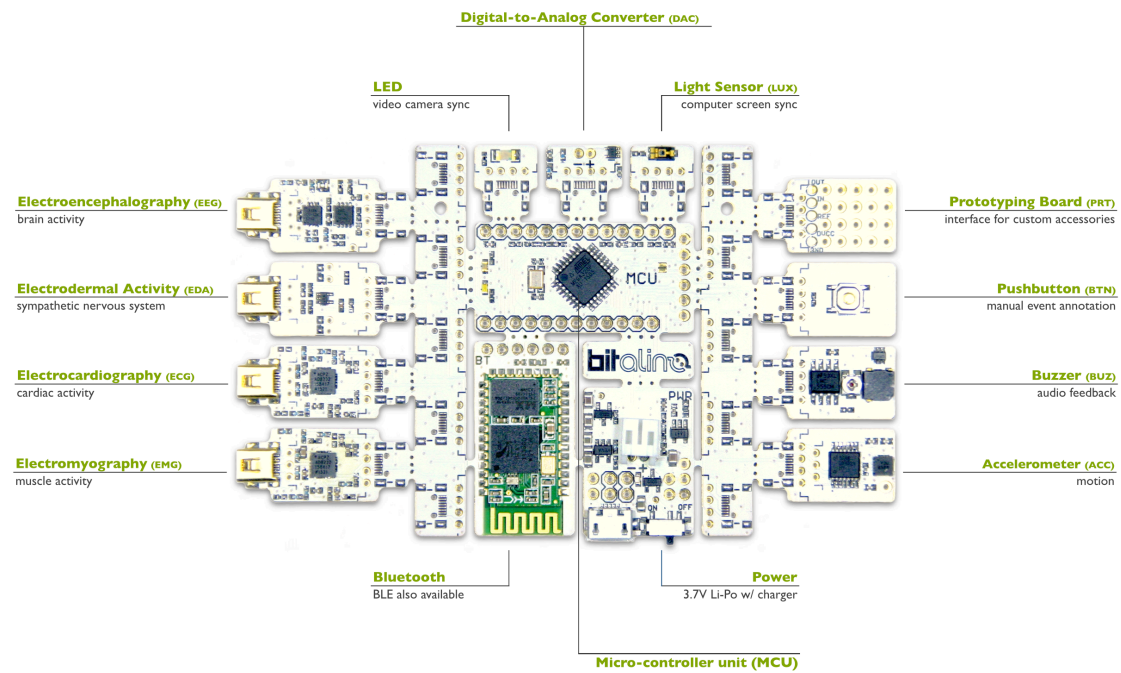


Fig. 4. BITalino (r)evolution Board with Bluetooth connectivity.

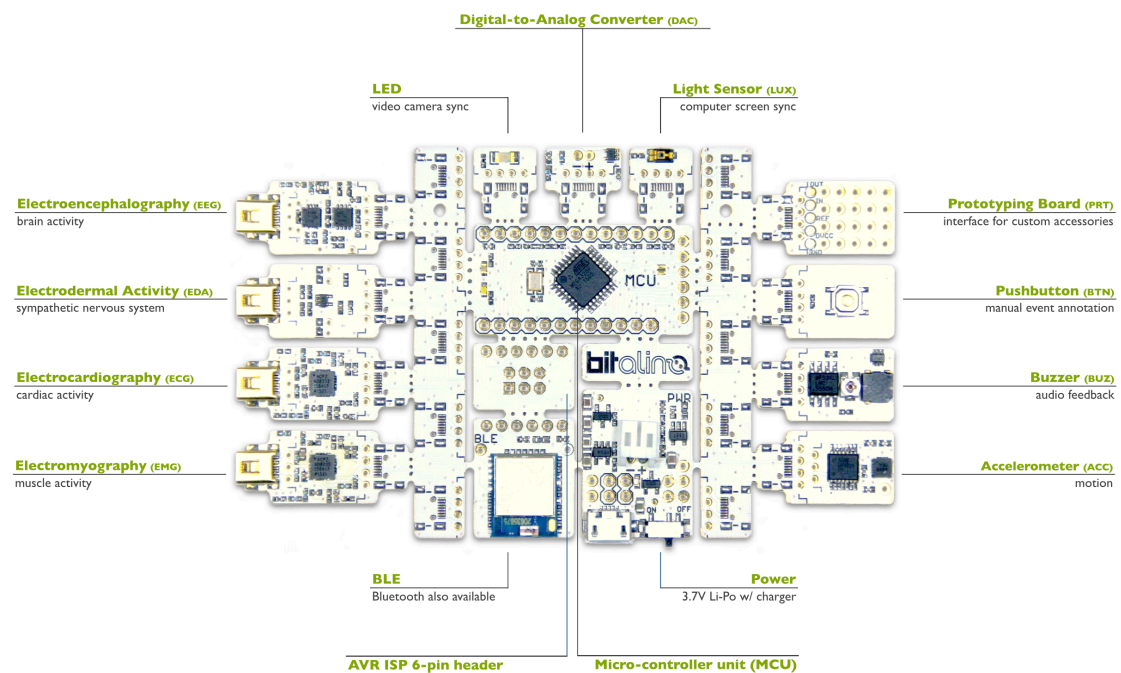


Fig. 5. BITalino (r)evolution Board with BLE connectivity.

BITalino (r)evolution Board Kit Data Sheet

PACKAGING

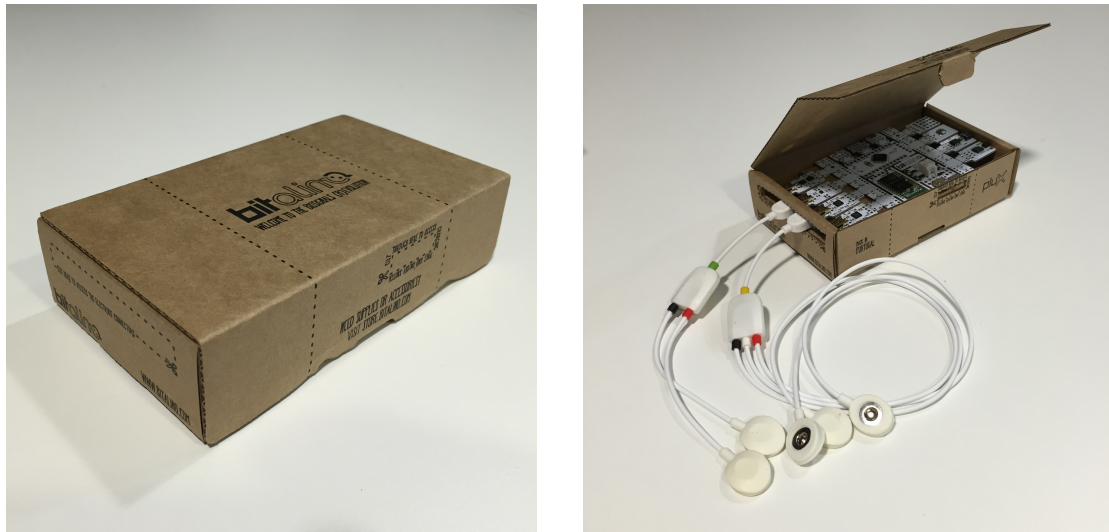


Fig. 6. BITalino (r)evolution Board kits ship in a convenient eco-friendly packaging that can double as an enclosure.

ORDERING GUIDE

Part #	Description
KIT-REV-BOARD-BT-UCE6	BITalino (r)evolution Board kit with Bluetooth connectivity and UC-E6 connectors
KIT-REV-BOARD-BLE-UCE6	BITalino (r)evolution Board kit with BLE connectivity and UC-E6 connectors

Appendix E

BITalino (r)evolution ECG Sensor Datasheet

(Source: http://bitalino.com/datasheets/REVOLUTION_ECG_Sensor_Datasheet.pdf)

Electrocardiography (ECG) Sensor Data Sheet

ECG 100716

SPECIFICATIONS

- > **Gain:** 1100
- > **Range:** $\pm 1.5\text{mV}$ (with $V_{CC} = 3.3\text{V}$)
- > **Bandwidth:** 0.5-40Hz
- > **Consumption:** $\sim 0.17\text{mA}$
- > **Input Voltage Range:** 2.0-3.5V
- > **Input Impedance:** 7.5GOhm
- > **CMRR:** 86dB

FEATURES

- > Bipolar differential measurement
- > Pre-conditioned analog output
- > High signal-to-noise ratio
- > Small form factor
- > Raw data output
- > Easy-to-use
- > "On-the-person" and "off-the-person" use

APPLICATIONS

- > Heart rate & heart rate variability
- > Human-Computer Interaction
- > Biometrics
- > Affective computing
- > Physiology studies
- > Psychophysiology
- > Biofeedback
- > Biomedical devices prototyping

GENERAL DESCRIPTION

Heartbeats are triggered by bioelectrical signals of very low amplitude generated by a special set of cells in the heart (the SA node). Electrocardiography (ECG) enables the translation of these electrical signals into numerical values, enabling them to be used in a wide array of applications. Our sensors allow data acquisition not only at the chest ("on-the-person"), but also at the hand palms ("off-the-person"), and works both with pre-gelled and most types of dry electrodes. The bipolar configuration is ideal for low noise data acquisition.

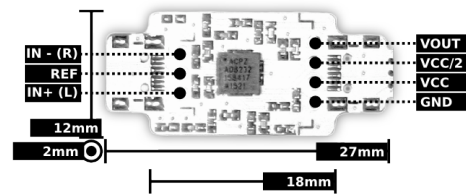


Fig. 1. Pin-out and physical dimensions.

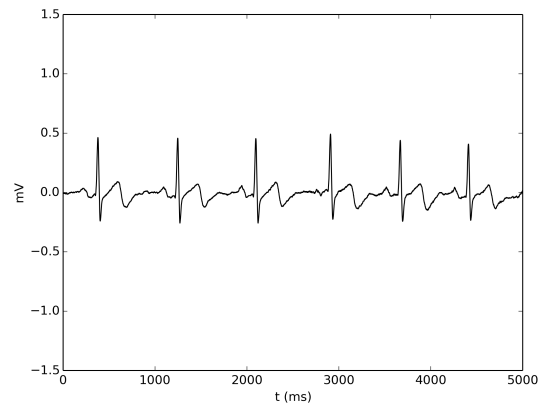


Fig. 2. Typical raw ECG data (acquired with BITalino (r)evolution) using an Einthoven triangle configuration.



Fig. 3. Example of a 1-lead placement with IN+ & IN- on the collarbones and REF on the iliac crest.

bitalino

PLUX – Wireless Biosignals, S.A.
Av. 5 de Outubro, n. 70 – 8.
1050-059 Lisbon, Portugal
bitalino@plux.info
<http://bitalino.com/>

REV A

© 2016 PLUX 

This information is provided "as is," and we make no express or implied warranties whatsoever with respect to functionality, operability, use, fitness for a particular purpose, or infringement of rights. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.



BEWARE: DIRECT OR INDIRECT COUPLING TO THE MAINS MAY RESULT IN SHOCKING HAZARD



Electrocardiography (ECG) Sensor Data Sheet

TRANSFER FUNCTION

[-1.5mV, 1.5mV]

$$ECG(V) = \frac{\left(\frac{ADC}{2^n} - \frac{1}{2}\right) \cdot VCC}{G_{ECG}}$$

$$ECG(mV) = ECG(V) \cdot 1000$$

$VCC = 3.3V$ (operating voltage)

$G_{ECG} = 1100$ (sensor gain)

$ECG(V)$ – ECG value in Volt (V)

$ECG(mV)$ – ECG value in millivolt (mV)

ADC – Value sampled from the channel

n – Number of bits of the channel¹

ORDERING GUIDE

Part #	Description
SENS-ECG-NC	Electrocardiography (ECG) sensor without connectors
SENS-ECG-UCE6	Electrocardiography (ECG) sensor with UC-E6 sockets on both sides for seamless plug & play connection to a BITalino (r)evolution Plugged or Core
SENS-ECG-SHER	Electrocardiography (ECG) sensor with a Molex Sherlock 4-pin socket on one side and a Molex Sherlock 3-pin socket on the other for easy power and signal cable connection or pin breakout using PCB wires

¹ The number of bits for each channel depends on the resolution of the Analog-to-Digital Converter (ADC); n BITalino the first four channels are sampled using 10-bit resolution ($n = 10$) while the last two may be sampled using 6-bit ($n = 6$)

Appendix F

OpenSignals (r)evolution Datasheet

(Source: http://bitalino.com/datasheets/OpenSignals_Datasheet.pdf)

OpenSignals (r)evolution Software Data Sheet

OSR 160616

SPECIFICATIONS

- > **Supported Devices:** BITalino; BITalino (r)evolution; biosignalsplux; motionplux
- > **Simultaneous Devices:** up to 3
- > **File Formats:** TXT; HDF5
- > **Plugins:** Video synchronization; Electromyography (EMG) analysis; Heart Rate Variability (HRV); Electrodermal Activity (EDA) events; Respiration (PZT & RIP) analysis

FEATURES

- > Real-time data visualization & recording
- > Visualization of recorded data
- > Feature extraction plugins with reporting
- > Synchronized multi-device recording
- > Multiplatform support
- > User-friendly GUI

APPLICATIONS

- > Psychophysiology
- > Biomedical projects
- > Computer science
- > Electrical engineering
- > Human-Computer Interaction
- > Robotics & Cybernetics
- > Physiology studies
- > Biomechanics
- > Biofeedback

GENERAL DESCRIPTION

Our cross-platform OpenSignals software enables instant data visualization and recording from any PLUX device. With a web-based GUI and a Python backend, OpenSignals combines high performance data handling and computing with user-friendly interfaces. The modular architecture enables it to be expanded with plugins for real-time or offline data processing and information extraction. Recorded data can be stored in a standard ASCII tab delimited file format or in the modern and highly efficient HDF5 format.



Fig. 1. Main screen.

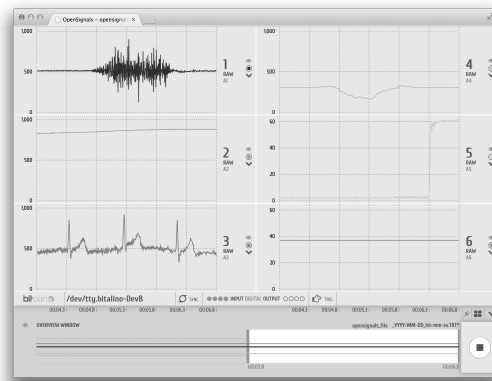


Fig. 2. Example of a visualization screen.

```
opensignals_file_2015-03-28_12-23-55.txt
# OpenSignals Text File Format
# ("/dev/tty.bitalino-DevB-1": {"sensor": ["RAW", "RAW", "RAW", "RAW", "RAW", "RAW"],
"device name": "/dev/tty.bitalino-DevB-1", "sync interval": 2, "time": "12:24:2.911",
"comments": "", "label": "fmsig", "d1": "D2", "d2": "D3", "d3": "D4", "A1": "A2", "A3": "A4",
"A5": "A6"}, "channels": [1, 2, 3, 4, 5, 6], "date": "2015-3-28", "mode": 0, "digital
ID": [0, 0, 0, 1, 1, 1, 1], "firmware version": "3.3a_18032014", "device": "BITalino",
"position": 0, "sampling rate": 1000, "resolution": [4, 1, 1, 1, 1, 10, 10, 10, 6,
6]})
# EndOfHeader
1 515 0 527 316 1 34
2 513 0 527 316 1 34
3 513 0 527 316 0 34
4 513 0 527 316 0 34
5 513 0 527 316 1 34
6 513 0 527 316 1 34
7 513 0 527 316 1 34
8 513 0 527 316 1 34
9 513 0 527 316 1 34
10 513 0 527 316 1 34
11 513 0 527 316 1 34
12 513 0 527 316 1 34
13 513 0 527 316 0 34
14 513 0 527 316 0 34
15 513 0 527 316 1 34
0 513 0 527 316 1 34
1 515 0 527 316 1 34
2 513 0 527 316 1 34
3 513 0 527 316 1 34
4 513 0 527 316 1 34
5 513 0 527 316 1 34
6 513 0 527 316 1 34
7 513 0 527 316 0 34
8 513 0 527 316 0 34
9 513 0 527 316 1 34
10 515 0 527 316 1 34
```

Fig. 3. Example of a recording stored in ASCII format.



PLUX – Wireless Biosignals, S.A.
Av. 5 de Outubro, n. 70 – 8.
1050-059 Lisbon, Portugal
plux@plux.info
<http://www.plux.info/>

REV A

© 2016 PLUX 

This information is provided "as is," and we make no express or implied warranties whatsoever with respect to functionality, operability, use, fitness for a particular purpose, or infringement of rights. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.

Appendix G

OpenSignals (r)evolution File Format Description

(Source: http://bitalino.com/datasheets/OpenSignals_File_Formats.pdf)

OpenSignals (r)evolution File Formats Description

OSF 080517

FEATURES

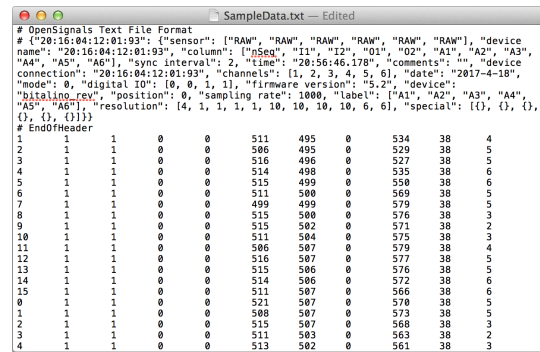
- > Data visualization & recording
- > Feature extraction plugins with reporting
- > Multiplatform support
- > User-friendly GUI

APPLICATIONS

- > Psychophysiology
- > Biomedical projects
- > Computer science
- > Electrical engineering
- > Human-Computer Interaction
- > Robotics & Cybernetics
- > Physiology studies
- > Biomechanics
- > Biofeedback

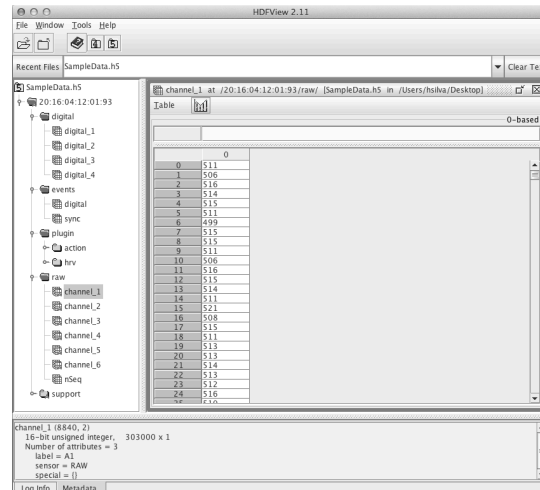
GENERAL DESCRIPTION

OpenSignals records data in ASCII tab delimited files (Fig. 1) and/or in the modern and highly efficient HDF5 format (Fig. 2). If the same format is followed, any third-party software can also export or record data in such a way that OpenSignals may be used as visualizer.



```
# OpenSignals Text File Format
# ("20:16:04:12:01:93": {"sensor": ["RAW", "RAW", "RAW", "RAW", "RAW"], "device
name": "20:16:04:12:01:93", "column": ["nSeq", "I1", "I2", "O1", "O2", "A1", "A2", "A3",
"A4", "A5", "A6"], "sync interval": 2, "time": "20:56:46.178", "comments": "", "device
connection": "20:16:04:12:01:93", "channels": [1, 2, 3, 4, 5, 6], "date": "2017-4-18",
"mode": 0, "digital I/O": [0, 0, 1, 1], "firmware version": "5.2", "device":
"BITalino_rev", "position": 0, "sampling rate": 1000, "label": ["A1", "A2", "A3", "A4",
"A5", "A6"], "resolution": [4, 1, 1, 1, 1, 10, 10, 10, 10, 6, 6], "special": [{"}, {}, {}],
{}}, {}}, {}))
# EndOfHeader
1 1 1 0 0 511 495 0 534 38 4
2 1 1 0 0 506 495 0 529 38 5
3 1 1 0 0 516 496 0 527 38 5
4 1 1 0 0 514 498 0 535 38 6
5 1 1 0 0 515 499 0 538 38 6
6 1 1 0 0 511 500 0 569 38 5
7 1 1 0 0 499 499 0 579 38 5
8 1 1 0 0 515 500 0 576 38 3
9 1 1 0 0 515 502 0 571 38 2
10 1 1 0 0 511 504 0 575 38 3
11 1 1 0 0 506 507 0 579 38 4
12 1 1 0 0 516 507 0 577 38 5
13 1 1 0 0 515 506 0 576 38 5
14 1 1 0 0 514 506 0 572 38 6
15 1 1 0 0 511 507 0 566 38 6
0 1 1 0 0 521 507 0 578 38 5
1 1 1 0 0 508 507 0 573 38 5
2 1 1 0 0 515 507 0 568 38 3
3 1 1 0 0 511 503 0 563 38 2
4 1 1 0 0 513 502 0 561 38 3
```

Fig. 1. Example ASCII file for BITalino (r)evolution; **Col. 1:** Sample sequence number (4-bit) generated on the device to facilitate the detection of missing data; **Col. 2 & 3:** State of the digital inputs I1 & I2; **Col. 3 & 4:** State of the digital outputs O1 & O2; **Col. 5-:** Raw data for each analog input acquired during the recording session (in this case A1-A6).



channel_1 at /20.16.04.12.01.93/raw/ [SampleData.h5 in /Users/holwa/Desktop]

0-based	
0	511
1	506
2	516
3	514
4	515
5	511
6	499
7	515
8	515
9	511
10	506
11	516
12	515
13	514
14	511
15	521
16	508
17	515
18	511
19	513
20	513
21	514
22	513
23	512
24	516
25	516

channel_1 (8840, 2)
16-bit unsigned integer, 303000 x 1
Number of attributes = 3
label = A1
sensor = RAW
special = []

Fig. 2. Example of a recording stored in HDF5 format.



PLUX – Wireless Biosignals, S.A.
Av. 5 de Outubro, n. 70 – 8.
1050-059 Lisbon, Portugal
plux@plux.info
<http://www.plux.info/>

REV A

© 2017 PLUX 

This information is provided "as is," and we make no express or implied warranties whatsoever with respect to functionality, operability, use, fitness for a particular purpose, or infringement of rights. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.

OpenSignals (r)evolution

File Formats Description

OSF 080517

METADATA FIELDS SUMMARY

Field	Description
<i>channels</i>	Set of analog inputs selected for acquisition in the recording session that generated this file (last columns of the file)
<i>column</i>	Meaning of each column (in ASCII files) or group with acquired data (in HDF5 files) for a given device ¹
<i>comments</i>	User-defined text with notes or other content of interest to the file
<i>date</i>	Day, month and year in which the file was recorded
<i>device</i>	Type of recording device used to collect the data (e.g. biosignalsplux, BITalino (r)evolution)
<i>device connection</i>	Logical address used to establish a connection with the device ²
<i>device name</i>	Friendly name manually assigned by the user to the device
<i>digital IO</i>	Set of digital channels available in this device (0 – Input; 1 – Output)
<i>firmware</i>	Version of the embedded software running on the device that acquired the data
<i>label</i>	Array with labels manually assigned by the user to each of the analogs channels
<i>mode</i>	Specifies the acquisition mode in which the device was used (0 – Regular acquisition; 1 – Multiple synchronized devices; 2 – Started by a triggering signal)
<i>resolution</i>	Resolution with which each of the columns (in ASCII files) or group with acquired data (in HDF5 files) are represented
<i>sampling rate</i>	Number of samples per second with which data has been acquired by the device on each channel
<i>sensor</i>	Array with the type of sensor connected to each analog port (e.g. as selected from the options available in the device configuration panel on OpenSignals)
<i>special</i>	List of non-standard channels acquired (e.g. SpO2 sensors with I2C interface connected on biosignalsplux hubs).
<i>sync interval</i>	Time interval (in seconds) at which a digital signal is sent by a “pacemaker” thread to a single device (used when the sync mode in on OpenSignals for synchronized data acquisition using multiple devices)
<i>position</i>	Order in which the block of columns corresponding to a given device appears in each line of the file (0 – First sequence of columns; 1 – Second sequence of columns; etc.)
<i>time</i>	Time at which the first sample was received by the software

ASCII TEXT FORMAT

OpenSignals ASCII text files have two parts, namely a header section and data section.

The header section has 3 header lines, each starting with the pound (#) character, given that it is automatically ignored by ASCII text file loading functions in common scientific computing programming languages (e.g. `loadtxt` in Python).

The header section begins after the line with the content “# OpenSignals Text File Format” and ends after the line “# EndOfHeader”. The second line has the metadata stored as a JSON object containing each device's MAC address as top-level keys. For each top-level key, there is a JSON object as value, containing the acquisition settings used for that

¹ e **nSeq** s the sample sequence number generated on the device **ix** s the state of digital input **Ox** s the state of digital output **x** and **Ax** s the raw data sampled by the device

² e.g. **AA:AA:AA:AA:AA:AA** represents a connection over a Bluetooth socket **0.0.0.0:0000** represents a connection over TCP/IP and **COM1** or **/dev/tty.BITalino-AA-AA-DevB** represents a connection over a serial port

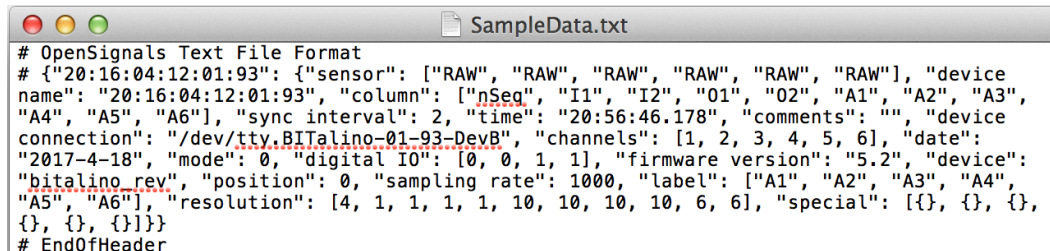
OpenSignals (r)evolution

File Formats Description

OSF 080517

device in the recording session that produced the file in the form of key / value fields (described earlier).

JSON has been chosen as the metadata representation format, given that in most scientific computing languages it can be directly unraveled as a dictionary, and it has parsers available for most mainstream programming languages.



```
# OpenSignals Text File Format
# {"20:16:04:12:01:93": {"sensor": ["RAW", "RAW", "RAW", "RAW", "RAW", "RAW"], "device
name": "20:16:04:12:01:93", "column": ["nSeq", "I1", "I2", "O1", "O2", "A1", "A2", "A3",
"A4", "A5", "A6"], "sync interval": 2, "time": "20:56:46.178", "comments": "", "device
connection": "/dev/tty.BITalino-01-93-DevB", "channels": [1, 2, 3, 4, 5, 6], "date":
"2017-4-18", "mode": 0, "digital I0": [0, 0, 1, 1], "firmware version": "5.2", "device":
"bitalino_rev", "position": 0, "sampling rate": 1000, "label": ["A1", "A2", "A3", "A4",
"A5", "A6"], "resolution": [4, 1, 1, 1, 1, 10, 10, 10, 10, 6, 6], "special": [{}, {}, {}
{}], {}, {}}}}
# EndOfHeader
```

Fig. 3. Example of an ASCII text file header.

On the data section, each column contains the signals labeled in the `column` field (see the caption on Fig. 1 for an example). To facilitate automated post-processing of the data, the resolution with which each column is recorded is stored in the `resolution` field (e.g. useful for a specific sensor's transfer function found in the corresponding sensor datasheet).

If the acquisition includes multiple devices, the metadata is stored following the same logic for each device, side by side, according to the `position` header field.

HDF5 FORMAT

OpenSignals HDF5 files comprise one group for each device used, represented by its MAC address. Each device's group includes metadata fields containing the acquisition settings used for that device in the recording session that produced the file (described earlier) and 5 sub-groups:

- > **digital**: includes a dataset for each digital channel (input channels followed by the output channels);
- > **events**: includes a dataset for digital events and dataset for sync events;
- > **plugin**: includes a group of datasets for each plugin used/processed;
- > **raw**: includes a dataset for the sample sequence number generated by the device and a dataset for each analog input selected for acquisition (each analog dataset includes the label and sensor type attributes).
- > **support**: includes a group of datasets with support information (mean, standard deviation, ...) for each zoom level available for each channel, either analog or digital (`t`: initial time for the sample group; `mean`: average, `sd`: standard deviation; `mx`: minimum value; `Mx`: maximum value; `mean_x2`: average of the sample group's 2nd power).

OpenSignals (r)evolution File Formats Description

OSF 080517

HRV Issue.h5

channel_1 at /20:16:04:12:01:93/raw/ [HRV Issue.h5 in /Users/hsilva...]

Index	Value
0	511
1	506
2	516
3	514
4	515
5	511
6	499
7	515
8	515
9	511
10	506
11	516
12	515
13	514
14	511
15	521
16	508
17	515
18	511
19	513
20	513
21	514
22	513
23	512
24	516
25	510
26	519
27	514
28	502
29	504
30	513

20:16:04:12:01:93 (800, 2)
Group size = 5
Number of attributes = 16
channels = 1,2,3,4,5,6
comments =
date = 2017-4-18
device = bitalino_rev

Fig. 4. Example of an HDF5 file structure.

Statement of Authorship

[ENGLISH] Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

[GERMAN] Eigenständigkeitserklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt und indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Ich weiß, dass die Arbeit in digitalisierter Form daraufhin überprüft werden kann, ob unerlaubte Hilfsmittel verwendet wurden und ob es sich – insgesamt oder in Teilen – um ein Plagiat handelt. Zum Vergleich meiner Arbeit mit existierenden Quellen darf sie in eine Datenbank eingestellt werden und nach der Überprüfung zum Vergleich mit künftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfältigungs- und Verwertungsrechte werden dadurch nicht eingeräumt. Die Arbeit wurde weder einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.