Development of VREACT, a tool for biosignal acquisition and visualization capable of real-time heart rate variability analysis

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplom-Ingenieur

im Rahmen des Studiums
Biomedical Engineering

eingereicht von
Fatih Kartal
Matrikelnummer 01425899

an der
Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität Wien

Betreuer:
Projektass. Dipl.-Ing. Florian Thürk
Ao.Univ.Prof. Dipl.Ing. Dr.techn. Eugenijus Kaniusas

Abstract

In order to ensure high safety measures during perioperative stage, patients in intensive care units (ICU's), operating theatres or recovery rooms are continuously monitored. Although the technologies and tools in use are constantly improving, perioperative organ injury, as cause of single- or multiple organ failure, is still a major risk for patients in this stage. Current monitoring set-ups in the perioperative setting provide basic parameters like electrocardiogram (ECG) or oxygen saturation (SpO2). While these signals are important for general health status tracking, they lack to offer a predictive value regarding cardiovascular complications. Studies have shown that changes in the heart rate variability (HRV) correlate strongly with a high risk in multiple organ failure making this parameter an early marker for diagnosis. In an effort to introduce HRV to perioperative monitoring, a new tool was developed within the frame of this work. "Vital-signs REal-time Analysis for Clinical Translation", VREACT, is an easy-to-use tool that allows for acquisition and continuous recording of high resolution biosignals from standard monitoring equipment. For the purpose of improving mortality risk analysis and prediction, our tool was enhanced by so called "modules", which are derived biosignals that can use data from the monitors in order to easily introduce novel clinical parameters to perioperative monitoring. A semi online peak detection algorithm was implemented and applied on the ECG data in order to allow for HRV analysis. Afterwards different established methods for HRV inspection in the time domain as well as in the frequency domain were added as modules. VREACT also provides a sophisticated feature called "PatientViewer", which enables real-time visualization of available modules. Finally, a modified version of VREACT called "VREACTquick" was developed, with the aim of handling long-term recordings without requiring user interaction. For the practical part, the performance of our tool was tested during real conditions in the General Hospital of Vienna. VREACT succeeded in recording over 15 patients from 3 different care units for over 2 hours. The PatientViewer managed to plot different modules correctly during 1 hour long tests for 5 randomly chosen patients. VREACTquick proved itself in an endurance test by collecting data from 2 different care units for over 3 days. For the future we expect our software to be used in various studies conducted in collaboration with the General Hospital of Vienna. Furthermore VREACT should constantly be improved by new modules and could also serve as a powerful acquisition tool in big data projects. In the course of this work one conference paper was submitted.

Kurzfassung

Das kontinuierliche Überwachen von Vitalparametern gewährleistet die Sicherheit von Patienten aus Intensivstationen, Operationssälen oder Aufwachräumen, während der perioperativen Phase. Trotz ständiger Entwicklung der modernen Medizintechnik, bleiben durch Multiorganversagen versursachte perioperative Organschäden weiterhin ein großes Risiko. Aktuelle Überwachungssysteme stellen nur grundlegende Biosignale wie das Elektrokardiogram (EKG) oder die Sauerstoffsättigung (SpO2) dar. Diese Signale sind zwar wichtig um den generellen Gesundheitszustand festzustellen, erlauben jedoch keine Voraussagen zu kardiovaskulären Komplikationen. Diverse Studien haben gezeigt, dass die Herzratenvariabilität (HRV) ein verlässlicher Risikoindikator für die Diagnose von Multiorganversagen ist. Um die HRV in das perioperative Umfeld einzuführen, haben wir im Rahmen dieser Arbeit eine neue Software entwickelt. „Vital-signs REal-time Analysis for Clinical Translation“, kurz VREACT, ist ein intuitives Tool, das die kontinuierliche Erfassung und Verarbeitung von hoch aufgelösten Biosignalen aus Überwachungssystemen ermöglicht. Darüber hinaus wurde unser Programm mit sogenannten „Modulen“ erweitert. Module sind abgeleitete Parameter, die mittels Daten aus den Überwachungssystemen neue Biosignale in die perioperative Überwachung einführen. Mit Hilfe eines semi online R-Zacken Detektionsalgorithmus, wurden aus EKG Daten kontinuierlich die Herzfrequenz berechnet um in weiterer Folge die HRV in der Zeit- und Frequenzdomäne zu analysieren. VREACT bietet zudem noch einen eigenen „PatientViewer“ der die Echtzeit-Darstellung der verfügbaren Module ermöglicht. Im Zuge der Entwicklung wurde eine modifizierte Version namens „VREACTquick“ veröffentlich. Diese Version erlaubt es Langzeitaufnahmen durchzuführen ohne dabei die Beaufsichtigung eines Benutzers zu benötigen. Für den praktischen Teil dieser Arbeit wurde die Software unter echten Bedingungen im Allgemeinen Krankenhaus (AKH) Wien getestet. VREACT bestand dabei erfolgreich eine 2 stündige Messung mit über 15 Patienten aus 3 verschiedenen Stationen. Der PatientViewer konnte die Daten verschiedener Module von 5 zufällig gewählten Patienten für jeweils über eine Stunde visualisieren. VREACTquick bestand einen Langzeittest in dem es Daten aus 2 verschiedenen Stationen über 3 Tage lang aufnahm. Für die Zukunft hoffen wir, dass unsere Software in Studien in Kooperation mit dem AKH Wien verwendet wird. Darüber hinaus, könnte VREACT ständig mit neuen Modulen erweitert werden und in Big Data Projekten Verwendung finden. Im Zuge dieser Arbeit wurde ein Conference Paper eingereicht.

Acknowledgments

First of all I want to thank DI Florian Thürk for his constant support and guidance throughout every phase of this work. Furthermore I want to thank my colleagues Mr. Jakub Matta and Mr. Max Schnetzinger for their help and share of work that made this project possible. A big thanks goes to Prof. Eugenijus Kaniusas who kept me inspired and interested in the field of biomedical engineering.

Most importantly I want to thank my family and my best friend and partner in life Ute Schreiner for being there for me whenever I was in need of support.

Table of Contents

List of Figures

List of Tables

List of Abbrevations

Abbrevation	Definition
AMI	Acute myocardial infarction
ANS	Autonomic nervous system
API	Application programming interface
AV	Atrioventricular
BA	Bland - Altman
BP	Blood pressure
DLL	Dynamic Link Library
ECG	Electrocardiogram
EEG	Electroencephalogram
FFT	Fast Fourier Transformation
GUI	Graphical user interface
HF	High frequency
HR	Heart rate
HRV	Heart rate variability
HTI	Heart rate variability triangular index
ICU	Intensive care unit
ICW	Intensive care window
IDE	Integrated development environment
LF	Low frequency
LOA	Limits of agreement
LOI	Line of identity
NN	Normal-to-normal
MVC	Model – View – Controller
PNS	Parasympathetic nervous system
RSA	Respiratory sinus arrhythmia
SA	Sinoatrial
SNS	Sympathetic nervous system

SpO2	Oxygen saturation
UI	User interface
ULF	Ultra low frequency
UML	Unified Modelling Language
VLF	Very low frequency
VREACT	Vital-signs Real-time Analysis for Clinical Translation

List of Symbols

Symbol	Definition
$D_{i(maj)}$	Distance of point (RR_i,RR_{i+1}) in the Poincaré plot from the major axis
$D_{i(min)}$	Distance of point (RR_i,RR_{i+1}) in the Poincaré plot from the minor axis
f_C	Heart rate
f_S	Sampling frequency
N	Total number of samples
NN	Normal-to-normal R-R interval
NN_{mean}	Mean value of all normal-to-normal R-R intervals
NN_n	*n*th NN interval
NN_{n+1}	*n+1*th NN interval
$RMSSD$	Root mean square of the difference of successive RR intervals
$\overline{\boldsymbol{RR}}$	Mean of **RR**
$\boldsymbol{RR}$	RR interval series used in Poincarè plot
RR_n	*n*th RR interval
RR_{n+1}	*n+1*th RR interval
$SD1$	Short term variability in the Poincaré plot
$SD2$	Long term variability in the Poincaré plot
$SDNN$	Standard deviation of all normal-to-normal R-R intervals
S_p	Spectral power
V_S	Stroke volume

1. Introduction

Continuous monitoring of physiological parameters is a standard procedure that is applied in ICUs and operating theatres of hospitals. It gives real – time information of the condition of a patient and can also deliver trend data to improve clinical decision making. Although anaesthesia and surgery related deaths have decreased in the last decade [1], perioperative organ injury, as cause of single- or multiple organ failure, is still one of the top causes for deaths in first world countries [2] [3]. Unfortunately, standard monitoring systems only allow the visualization of basic vital signs that are not sufficient for proactive reacting. A solution for this, is the monitoring of the HRV parameter, which is known to be an early marker for multiple organ failure [4] [5].
In this work, we developed an easy-to-use tool, “Vital-signs REal-time Analysis for Clinical Translation”, VREACT, which allows for acquisition of high resolution biosignals obtained directly from Dräger Infinity Delta monitors [6], in the local network of a hospital. VREACT not only enables the simultaneous and continuous recording of signals coming from every patient that it is connected to, but also includes its own “PatientViewer” for real – time visualization. Furthermore VREACT has the capability to extend standard biosignals, delivered by the monitoring systems, by so called “modules” (derived biosignals). Therefore our software introduces novel clinical parameters (such as HRV) to perioperative monitoring. VREACT was developed by Mr. Jakub Matta, who was responsible for the main graphical user interface (GUI) and establishing and handling connections to the Dräger monitors in general, and me, responsible for the development of the modules, the PatientViewer and VREACTquick (a modified version of VREACT).

1.1. Anatomical and physiological background

In this chapter the anatomy and physiology of the human heart will be explained. Furthermore the role and functionality of the cardiovascular system will be discussed.

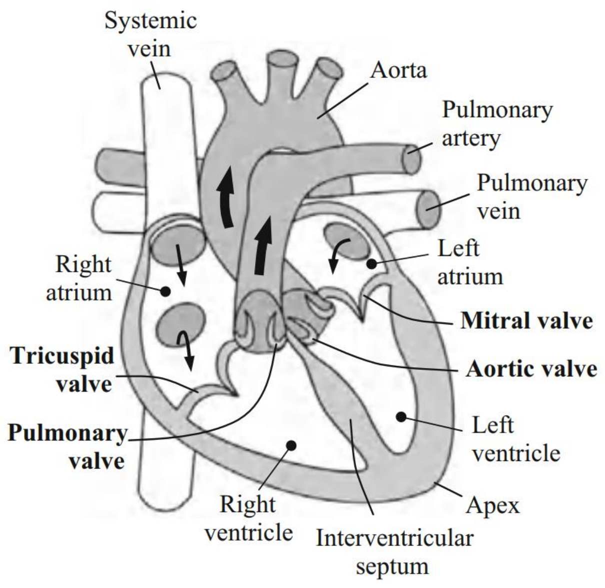


Figure 1: Anatomy of the human heart [7].

1.1.1. Cardiac anatomy

The human heart is a muscular pump that collects and pumps blood through a system of veins and arteries. It lies in the thorax, where it is protected by the rib cage, posterior to the sternum and on the superior surface of the diaphragm. The heart is covered by the pericardium, which has outer and inner layers with a lubricant in between. This allows the inner layer to glide against the outer layer and therefore movement and expansion of the heart. As seen in Figure 1, the heart consists of four separate chambers. The upper chambers, otherwise called atria, collect blood from veins whereas the lower ones, called ventricles, pump the blood out of the heart. The atrioventricular (AV) valves, which are called tricuspid (right side) and bicuspid (left side), maintain the one way direction of the blood flow between the atria and ventricles. The semilunar valves, named pulmonary valve and aortic valve, maintain the same for blood leaving the ventricles. The right atrium of this system collects blood from the body through the vena cava. The pressure in the atrium rises until the tricuspid valve opens and blood enters the right ventricle. Afterwards blood is pumped into the lungs through the pulmonary artery. On the left side the oxygenated blood is collected from the pulmonary vein and then pumped into the body through the aorta [8][9][7].

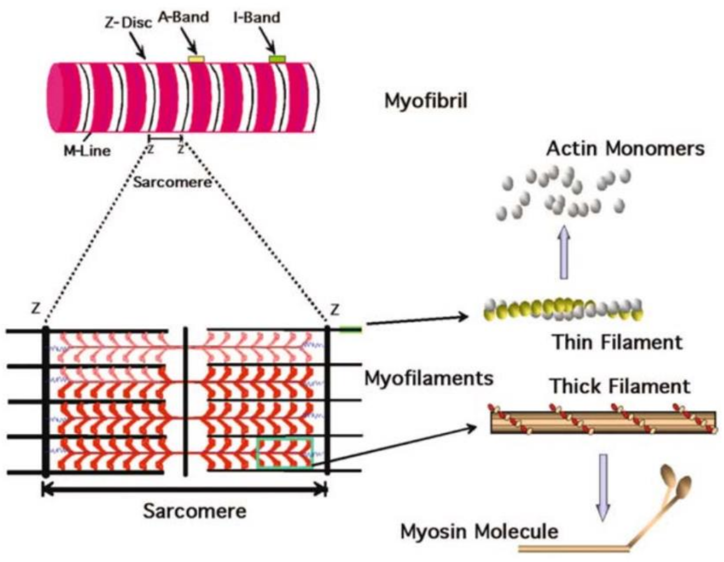


Figure 2: Structure of a sarcomere. Actin and myosin are arranged in filaments with interaction that is the molecular basis of muscle contraction [10].

1.1.2. Cardiac physiology

Sarcomeres

Cardiac muscle cells contain bundles of myofibrils, which are organized into contractile functional units, called sarcomeres. In Figure 2 the structure of a sarcomere is shown. The borders of sarcomeres are built by a protein matrix called the Z line, which consists of the protein a – actinin. Each sarcomere has a lattice of thicker and thinner protein filaments. Thin filaments reach from the Z line towards the centre, are about 1µm long and are globular subunits of the protein actin. Thick filaments consist of the protein myosin and have myosin heads that extend from the filament. These heads connect to the thin filaments and give the ability shorten the muscle upon contraction. The contraction is initiated with an influx of calcium into the sarcoplasmic reticulum. This leads to an alteration in the angle of the myosin actin connections, which results in an overlap of these proteins and therefore in a shortening of sarcomere. The area of a sarcomere with thick filaments is called the A band and the area between two A bands is known as the I band [9][10].

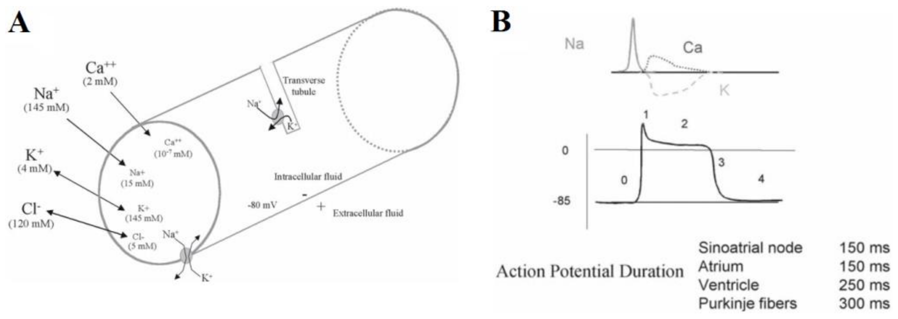


Figure 3: Ionic distribution for cardiac cells (A) and Ionic conductance changes during the ventricular cardiac action potential (B) [10].

Action potential

There are two ways for cardiac cells to communicate with each other. The first way is through mechanical bonds formed by protein – protein associations at the membrane surfaces allowing the transmission of forces across the myocardium. The second way is through gap junctions that create electrical connections between cells [10].

In Figure 3A, the ionic composition inside and outside of a cell is shown. This specific distribution of ions is maintained by ion pumps, channel proteins and ion exchange proteins and is responsible for the cell membrane potential, which is about -90mV. A depolarization of the cell occurs through an increase in sodium permeability. As seen in Figure 3B, this process marks the beginning of a so called action potential. Phase 0 is the transition from the resting potential to depolarization. In phase 1 a repolarization occurs due to the closing of sodium channels. Now voltage – gated calcium and potassium channels are activated. The influx of calcium sustains the depolarized state and the efflux of potassium drives the membrane potential back to a negative membrane potential, which leads to a positive plateau in phase 2. As the calcium channels start to close, the potassium channels dominate and the cell repolarizes (phase 3 to phase 4) [9][10].

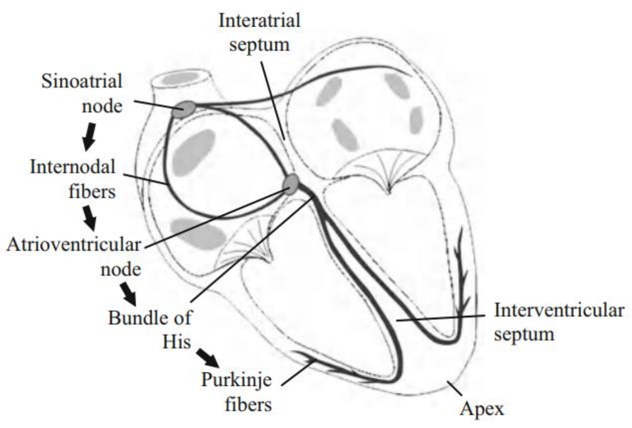


Figure 4: The excito – conductive system of the heart [7].

Pacemaker cells

The excito – conductive system of the heart consists of modified cardiac cells also called pacemaker cells. These cells are grouped in nodes and bundles and are able to generate and conduct electric stimulation. Due to leak channels, pacemaker cells have an unstable resting potential. This means their resting potential gradually rises until a depolarization is initiated.

In Figure 4 the components of the excito – conductive system are shown. It consists of the sinoatrial (SA) node, the AV node, the Bundle of His with a right and a left bundle branch. The branches contain the Purkinje fibres that are connected with contractile myocytes. The SA node is the main pacemaker with a heart rate (f_C) of about 70 per minute. f_C is measured in beats per minute and is controlled by the autonomic nervous system (ANS). It generates action potentials which spread from the atria to the ventricles. It is important to note that the atria are electrically isolated from the ventricles in order to deny simultaneous contraction. Action potentials cross the AV node in the interatrial septum. This is the only conductive connection between the atria and the ventricles. Afterwards the electrical path passes the Bundle of His and finally separates into both ventricles via the Purkinje fibres. If for some reason the SA node stops working as a pacemaker, the AV node steps in as a secondary pacemaker with f_C of about 50 per minute. Also the Bundle of His and Purkinje fibres can step in as tertiary pacemakers with a f_C of about 30 per minute. As seen in Figure 3B the action potential duration is different for each pacemaker [9][7][10].

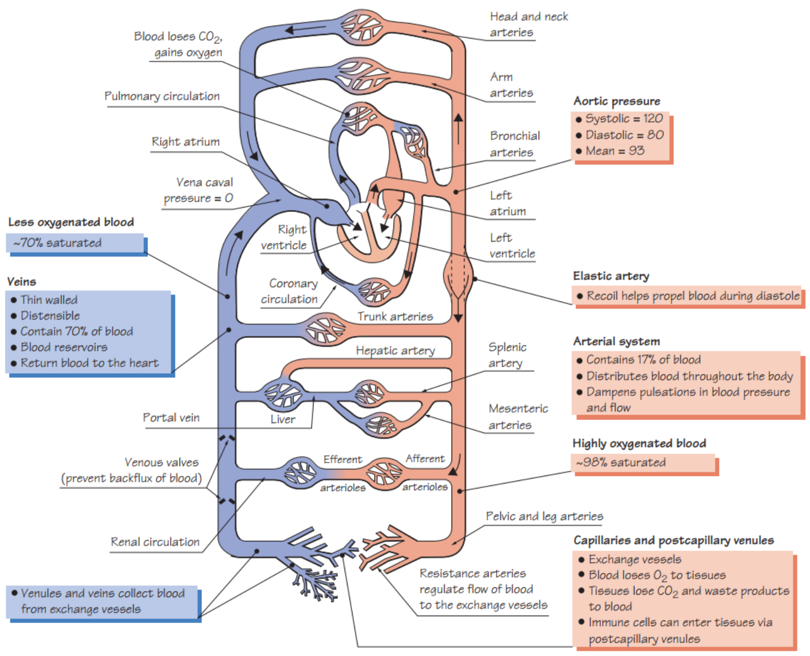


Figure 5: Overview of the human cardiovascular system. It consists of the heart, blood vessels and blood. The systemic circulation refers to the blood flow from the left ventricle to the right atrium and the pulmonary circulation to the blood flow from the right ventricle to the left atrium. [11].

1.1.3. Cardiovascular system

In Figure 5, a detailed picture of the cardiovascular system can be seen. This system consists of the heart, blood vessels and blood. Its functions are distribution of oxygen, water, nutrients and hormones, removal of carbon dioxide and waste products, contribution to the immune system and thermoregulation. Blood is supplied through the arteries to organs and body regions. The arteries divide into smaller arterioles where they converge into venules. The venules collect the deoxygenated blood from the exchange vessel and pass it to the right atrium of the heart. This circulation is called the systemic circulation. The second circulation is called the pulmonary circulation, in which the deoxygenated blood is pushed out of the right ventricle into the lungs, where gas exchange processes occur. Afterwards the oxygenated blood enters the left atrium of the heart [7][11].

Arterial walls consist of three layers. The inner layer contains endothelial cells and connective tissue with elastic fibres. The middle layer contains smooth muscles that are controlled by sympathetic nerve fibres. These nerve fibres are responsible for vasoconstriction (contraction of vessels) and vasodilation (relaxation of vessels), which are mechanisms that directly influence blood flow. Finally the outer layer consists of thick collagen fibres and elastic fibres. Venous walls also have three layers and their structure is similar to that of arteries. The inner layer consists of endothelial cells, the middle layer of

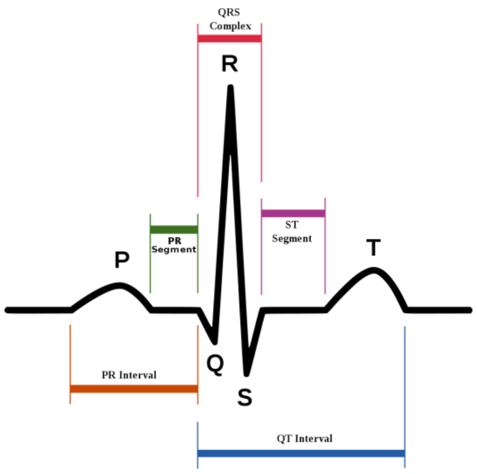


Figure 6: Schematic of a typical ECG recording [12].

smooth muscles and elastin and the outer layer of connective tissue. The blood flow through veins is maintained by skeletal muscles in the vicinity, by the respiratory pump and the smooth muscles in the venous walls [7][11].

1.2. Electrocardiogram

As described in 1.1.2 the heart contracts due to electrical processes. A measurement of these electrical activities leads to a biosignal called the ECG.

In Figure 6 the shape of a typical ECG signal for one heart cycle is shown. It consists of the following parts:

- P – wave: depolarization of atria
- PR – segment: electrical conduction between atria and ventricles
- QRS – complex: depolarization of ventricles
- T – wave: repolarization of ventricles

The ECG is a widely used and an important clinical parameter for diagnosis of various cardiac conditions. It is a method that gives direct information about changes and abnormalities in the cardiac rhythm. A parameter that can be easily extracted from the ECG is the f_C. It is calculated by measuring the time difference between two successive R – peaks [7][13][14].

1.3. Heart rate variability

HRV is the beat to beat variation in either heart rate or the time difference between two successive R – peaks [15]. In this section the regulation of HRV by the ANS will be explained. Furthermore different approaches for HRV analysis in the time and frequency domain will be introduced and discussed.

1.3.1. Regulation by the autonomic nervous system

The ANS has two main components, the sympathetic nervous system (SNS) and the parasympathetic nervous system (PNS). The PNS is the “rest” system that is concerned with promoting the conservation of energy. It is mediated by the release of acetylcholine by the vagus nerve toward the heart which slows down the diastolic depolarization, increases ventricular refractory period and therefore decreases f_C. The SNS is the “fight or flight” system that responds to threatening situations. It intervenes by the release of hormones like adrenalin and is induced via splanchnic nerves, the neuronal activation of beta receptors in the heart (SA node and AV node), the acceleration of the slow diastolic depolarization and a decrease in cardiac refractory period, resulting in an increased f_C [7][16]. The PNS and the SNS oppose and balance each other. As SNS begins to rise the f_C, the PNS goes into action in order to tone it down again. Since the PNS innervates the SA node directly via the vagus nerve, it is able to respond much quicker than the SNS. This ability is very important since it enables the body to react to tachycardia very quickly. HRV constantly changes due to autonomic influences on the SA node. During inspiration the R – R intervals shorten (f_C increases) and during expiration they get longer (f_C decreases) [17]. This phenomenon is called respiratory sinus arrhythmia (RSA) and occurs due to following mechanism. During inspiration the filling of the right side of the heart increases (see respiratory pump in 1.1.3). Because of this the left ventricle is compressed and consequently a decrease in stroke volume (V_S) occurs. The ANS makes up for the smaller V_S by an increase in f_C [7]. Overall HRV is mostly dependent on the circadian rhythm with a predominant sympathetic activity during day and vagal activity during night. A reduction of HRV is highly associated with an increase in cardiac mortality [18].

1.3.2. HRV analysis in the time domain

In the time domain the standard deviation of all normal-to-normal R-R intervals (SDNN), the root mean square of the difference of successive RR intervals (RMSSD), the sum of all normal-to-normal (NN) intervals greater than 50ms (NN50) and NN50 divided by total number of n NN intervals (pNN50) are commonly used parameters [19]. Furthermore geometric techniques like Poincaré Plot (scatter plot) and sample density histograms are applied [20].

SDNN

$$NN_{mean} = \frac{1}{n}\sum_{i=1}^{n} NN_i \tag{1}$$

$$SDNN = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(NN_i - NN_{mean})^2} \tag{2}$$

In Formula 2, the calculation for SDNN is shown. SDNN strongly depends on recording durations making comparisons on measurements of different lengths invalid. Generally it is reported that SDNN calculations on short term 5 minutes (includes short term HF components) and long term 24 hour (includes both short term HF components and long term LF components) recordings are appropriate [19][21]. Average value in healthy subjects is about 141 $\pm$ 39ms (mean $\pm$ standard deviation) for long term recordings [20].

RMSSD

$$RMSSD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(NN_{i+1} - NN_i)^2} \tag{3}$$

In Formula 3, the calculation for RMSSD is shown. RMSSD is a parameter that is used for short term HRV measurements [19][20][22]. Since short term variability is purely controlled by the parasympathetic system, this parameter measures the parasympathetic modulation of heart rate. Average value in healthy subjects is about 27 $\pm$ 12ms [20].

NN50

As already mentioned, NN50 is a measure for number of normal-normal intervals that exceed 50ms. NN50 is also a measure for short term HRV and highly correlates with RMSSD. In direct comparison, RMSSD has better immunity to ectopic beats and nicer statistical properties, which makes it the preferred parameter for short time measurements [19][20].

pNN50

pNN50 is the fraction of NN50 intervals as a proportion of the total number of NN intervals. Since this parameter is directly derived from NN50, it naturally is a measure for short time HRV and also highly correlates with RMSSD [19][20].

Poincaré plot

The Poincaré plot (named after Henri Poincaré, a French scientist), in the context of HRV, is a scatter plot of RR or NN intervals against the following RR or NN interval [20].

As seen in Figure 7, the plot resembles a cloud along the line of identity (LOI) which is the line where $RR_n = RR_{n+1}$. The shape of this cloud gives valuable information about the HRV of a patient and can be measured as the dispersion of points perpendicular to the LOI (short term variability) and along the LOI (long term variability) [20].

In order to characterize the Poincaré plot mathematically, an ellipse – fitting technique is used. The major axis of the fitted ellipse is in alignment with the LOI (slope of 45°, see Formula 4) and the minor axis is perpendicular to the LOI (slope of 135°, see Formula 5) and passes through the centroid of the plot [20].

$$RR_n = RR_{n+1} \qquad (4)$$

$$RR_n + RR_{n+1} = 2\overline{\boldsymbol{RR}} \qquad (5)$$

In Formula 5, $\boldsymbol{RR}$ represents the RR interval series used in Poincarè plot and $\overline{\boldsymbol{RR}}$ the mean value of this series. The dispersion of points along the major axis is a measure for the length of the plot and the dispersion along the minor axis a measure for the width of the plot [20].

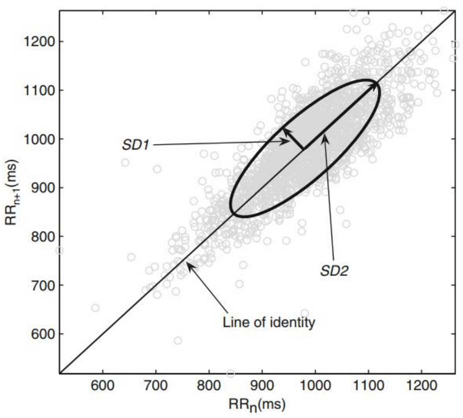


Figure 7: Poincaré plot of RR intervals in HRV data. An ellipse is fitted for SD1 and SD2 calculation [20].

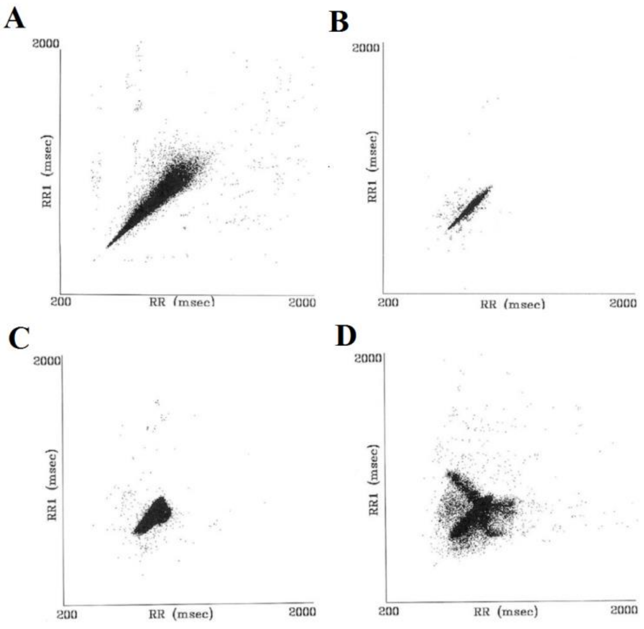


Figure 8: Different HRV Poincaré plots collected from patients. Healthy patient with comet pattern (A), heart failure patient with torpedo pattern (B), heart failure patient with fan pattern (C) and heart failure patient with complex pattern (D). Figure adapted from Woo et al [23].

$$D_{i(\min)} = \frac{RR_i - RR_{i+1}}{\sqrt{2}} \qquad (6)$$

$$D_{i(\text{maj})} = \frac{RR_i + RR_{i+1} - 2\overline{\boldsymbol{RR}}}{\sqrt{2}} \qquad (7)$$

$$SD1 = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1} D_{i(\min)}^2} \qquad (8)$$

$$SD2 = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1} D_{i(\mathrm{maj})}^{2}} \qquad (9)$$

Formulas 6 and 7 are distance measurements for points P(RRi,RRi+1) from the minor (6) and major axis (7). Finally, the short (8) and long term variabilities (9) of the plot are expressed in Formulas 8 and 9, with N being the total number of RR intervals [20].

In Figure 8 different possible patterns for Poincaré plots are shown. The pattern in Figure 8A is called a “comet” pattern. It represents the lengthening of RR intervals, which is an indication for increased HRV found in healthy subjects. In Figure 8B a “torpedo” pattern be seen. It indicates that changes between consecutive RR intervals are minimal. Figure 8C shows a “fan” pattern, meaning the data has a small increase in RR interval length. It is associated with greater dispersion in consecutive RR intervals. The last pattern, seen in Figure 8D, is called a “complex” pattern. It has clusters of points with distinct gaps in between. The RR intervals change stepwise and represent a lack of graded relationship between successive intervals (nonlinear behaviour) [23].

Histogram

The histogram provides a visual overview of the density of RR intervals. Furthermore it can be used to extract the HRV triangular index (HTI) from it. For this, the total number of RR intervals (integral of the density curve) is divided by the most frequent RR interval length (height of the histogram). 5 minute epochs of HRV data can be used for HTI calculations and it is reported that HTI > 20.42 is an indicator for arrhythmia [25][26].

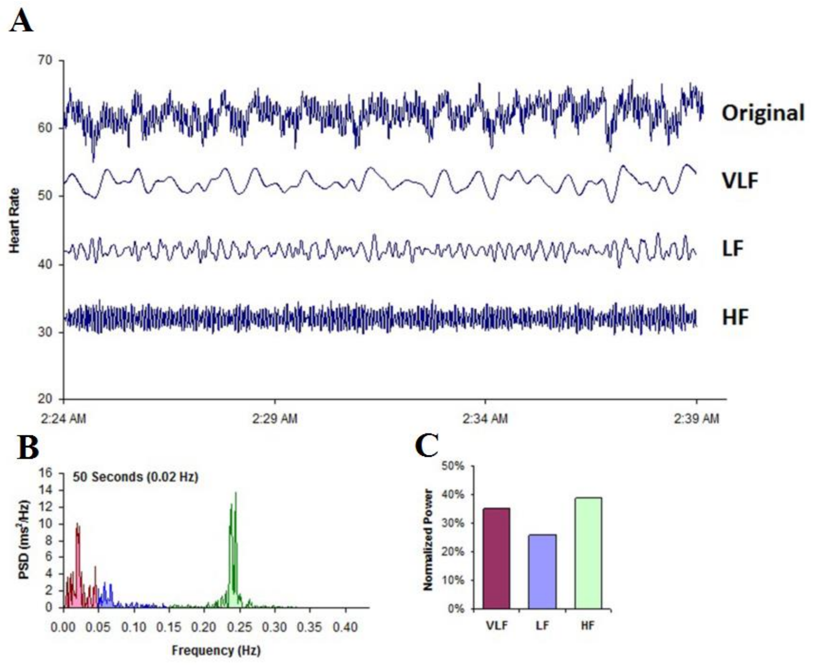


Figure 9: Typical HRV recording over a period of 15 minutes during resting conditions in a healthy subject. In (A) the original HRV waveform and the waveform after applying different filters can be seen. The filtering results in waveforms of the VLF, LF and HF band. In (B) the power spectra and in (C) the percentage of power in each band can be seen [24].

1.3.3. HRV analysis in the frequency domain

Spectral analysis of HRV gives important information on the ANS. It is applied on short term and long term recordings. Fast Fourier Transformation (FFT) is used in order to split HRV into ultra low frequency (ULF), very low frequency (VLF), low frequency (LF) and high frequency (HF) components (see Figure 9) [19][20].

Ultra low frequency band

The ULF band ($\leq$ 0.003Hz) can be inspected in recordings with a period of at least 24 hours. Although there is no consensus regarding the mechanisms that generate ULF power, circadian rhythm may be the primary driver [24]. It is assumed that core body temperature and metabolism could also be contributors [26].

Very low frequency band

The VLF band (0.003Hz – 0.04Hz) requires a recording of at least 5 minutes, but is advised to be monitored over 24 hours. VLF power is greatly associated with all – cause mortality [27][28]. There is still uncertainty for activity within this band, since no known physiological rhythms show correlation [20][26].

Low frequency band

The LF band (0.04Hz – 0.15Hz) is recorded from HRV data with a period of at least 2 minutes. This band mainly reflects baroreceptor activity (vasomotor oscillations) and is influenced by both the SNS and the PNS [20][26].

High frequency band

The HF band (0.15Hz – 0.4Hz) is recorded over a minimum 1 minute period. It reflects PNS activity and corresponds to HRV related to the respiratory cycle (see RSA in 1.3) [20][26]. HF power is highly correlated with pNN50 and RMSSD [29].

LF/HF ratio

Since LF power contains both SNS and PNS activity and HF power only contains PNS activity, the LF/HF ratio (sometimes used inversely) is a measure of parasympathetic/sympathetic balance [26].

1.4. Related work

HRV was used in a wide range of studies for the purpose of health status and risk of mortality analysis. This section will give a brief overview of related publications that focused on the HRV parameter and also the development of similar biosignal acquisition tools.

Many clinical studies of HRV analysis in the spectral domain were conducted. Winchell et al [30] developed an automated system for real – time spectral analysis of HRV data and used it to study mortality in a surgical ICU population. They calculated the total spectral power, which is a measure of overall autonomic activity, and HF/LF every 6 hours. After conducting 7994 measurements in 742 patients, they concluded that both low total spectral power and high HF/LF indicate increased mortality.

Bigger Jr. et al investigated spectral information of HRV in two different studies over the same population of 715 patients with acute myocardial infarction (AMI). In the first study [31], they inspected power in the ULF, VLF, LF and HF bands over a time period of 24 hours. ULF and VLF power had stronger associations with all – cause mortality, cardiac death and arrhythmic death than power in LF and HF. In their second study [32], they tested the ability of shorter recordings in mortality predictions. For this, they compared the performance of their 24 hour predictions with predictions in shorter segments (2, 5, 10 and 15 minutes). They concluded that power spectral measures from shorter recordings were similar to those calculated from over 24 hours and can therefore be used to predict all – cause mortality and sudden cardiac death.

Extensive studies on the effect of AMI on the SDNN were conducted. Kleiger et al [33] tested the hypothesis of HRV being a predictor in long – term survival after AMI with 808 patients who survived AMI. They found that the risk of mortality was 5.3 times higher in the group with SDNN < 50ms than the group with SDNN > 100ms.
La Rovere et al [34] carried out a study with 1284 patients with a recent (<28 days) AMI. During 21 months 44 of the patients died and 5 suffered a non – fatal cardiac arrest. Their conclusion was that a SDNN < 70ms is a significant factor for cardiac death.
In another study, Erdogan et al [35] researched the significance of impaired HRV after AMI in patients in whom early reperfusion was attempted. For this they measured SDNN in 412 patients treated with direct coronary angioplasty within 12 hours of symptom onset. They found a strong correlation between SDNN < 50ms and mortality but also noted that SDNN had a low positive predictive value.

Thayer et al [36] performed a meta – analysis of neuroimaging studies on the relationship between HRV and regional cerebral blood flow. They identified several regions with significant associations and proposed that HRV may be an important marker for stress (defined as perception of threat) by linking it to activity in the amygdala.

The “Vital Recorder” was developed and published by Lee and Jung [37]. Their lightweight tool offers high resolution recordings and a multi-functional UI for real time visualization, while having low CPU usage. Important features are the “track mode” which allows for tracking of all available signals and a timeline with sample history. Furthermore the user can enter events to given time stamps and make notes about administered medications. The second mode is called the “monitor mode” and basically mimics the display of a standard monitoring device by visualizing the most important vital data. Furthermore Vital Recorder can import and load data from the Physionet database, as it supports the .edf file format. Although this tool is compatible with more than 20 different monitoring devices, establishing connections to individual monitors takes effort and requires a higher technical skill level. The communication is maintained by the RS-232C serial port of the monitors. In addition analog-to-digital converters may be required to obtain data from the analog port of a monitor and a communication protocol needs to be set up. Finally, the Dräger Infinity Delta is not supported by Vital Recorder making this tool not applicable in the General Hospital of Vienna.

An open source tool, named “Intensive Care Window” (ICW), was introduced by Stylianides et al [38]. Their project contains two main parts. The first part is the “ICW Bedside Controller Middleware”, which is an API that allows for communication with monitoring devices that based their communication protocols on the medical information bus standard. The second part is the “ICW Application”, which is a GUI that makes use of their own API in order to visualize signals in real time. This tool can read the internal settings of the monitoring device and import important information like alarm thresholds. Although ICW is able to perform post-processing, this function only allows for novel parameters after finishing a recording. This makes it unable to provide HRV data in real-time. Furthermore, like the already mentioned Vital Recorder, it also uses the RS-232C serial port for connections which makes it necessary to have the computer in the vicinity of a monitor while only allowing for recordings and visualization from one monitor at a time.

1.5. Dräger Infinity Delta

The Dräger Infinity Delta is a state-of-the-art monitoring device that is capable of displaying various vital parameters. In Figure 10, a picture of the monitoring device can be seen. The Infinity Delta series is designed to be portable, meaning patients in ICUs don’t have to be disconnected during relocation. Parameters like 3-, 5-, 6- and 12-lead ECG, respiration, ST segment analysis, $etCO_2$ (the level of carbon dioxide released at the end of expiration), bispectral index (depth of anaesthesia), electroencephalogram (EEG) , multiple temperatures, invasive and non-invasive blood pressure and full arrhythmia can be continuously monitored [6].

The Infinity Delta monitors are connected to the so called Infinity Network. The Infinity Gateway is a client – server application that allows computers within the hospital network to access these monitors in order to view patient information. For this, the application is installed on a computer (client) that is part of the hospital network. The server portion runs on a computer that is connected directly both to the hospital network and to the Infinity network.

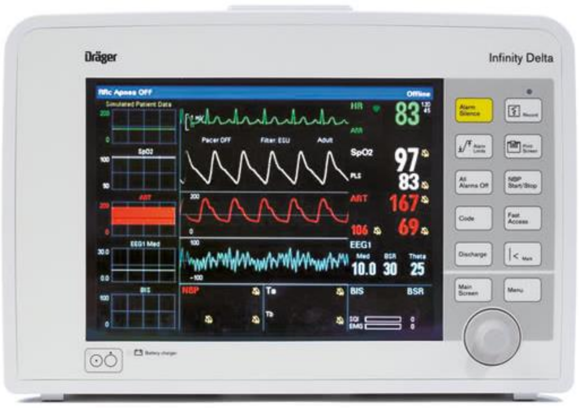
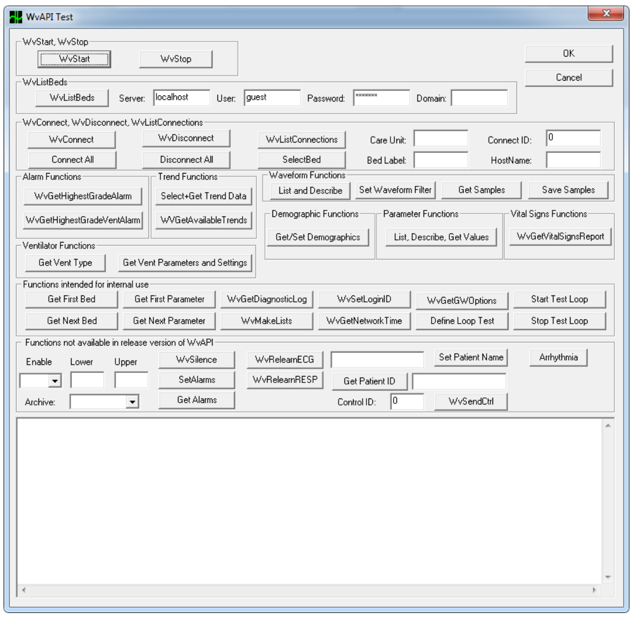


Figure 10: Dräger Infinity Delta. A state-of-the-art monitoring device [6].



Server: localhost
User: guest
Password: *******
Domain:
Care Unit:
Connect ID: 0
Bed Label:
HostName:
Enable
Lower
Upper
Archive:
Control ID: 0

Figure 11: User Interface of the Dräger Infinity Delta software.

Unfortunately the user interface (UI) (Figure 11) of the included software is unnecessarily complicated and therefore unpleasant to use. Furthermore it only allows users to save the data recorded within the last 10 seconds.

1.6. Qt Framework

The source code of the original Dräger Infinity Delta software was written in C++ using the Microsoft Visual Studio integrated development environment (IDE) [39]. The UI was developed using the Microsoft Foundation Class Library [40].

For this project we decided to use the Qt IDE and framework. Qt offers various tools and libraries for creating dynamic UI's using the "signals & slots" mechanism [41]. In addition there are several options for creating and maintaining threads in the application. We kept C++ as our programming language, since it allowed us to transfer and understand some of the existing code more easily.

1.6.1. QObject

The QObject class is a very important component of the Qt framework, because all other Qt objects derive from it. QObject provides functions like QObject::connect() and QObect::disconnect(), which are essential for the usage of signals & slots [42]. Furthermore, all classes that use signals & slots or QThread have to implement the Q_OBJECT macro. This macro will be translated into C++ source code by the Meta-Object Compiler. Finally, the translated code will be compiled and linked with the class's implementation [43].

1.6.2. Signals & Slots

Qt offers the signals & slots mechanism as a central feature of their framework. It enables easy communication between different objects.

In Figure 12, a schematic illustration of the signals & slots can be seen. A signal is emitted when a certain event occurs. Qt Widgets Modules provide a set of UI elements that have many predefined signals. A slot is defined as a function that is triggered by a signal it is connected to. Widgets also have many predefined slots, but it is always possible to create your own signals & slots. An example for this mechanism would be to click a button (signal) in order to change the background colour of a window (slot):

```
connect(button, SIGNAL(clicked()), window, SLOT(changeColor()));
```

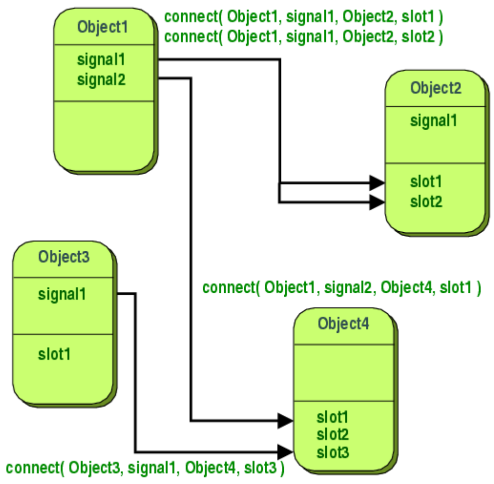

connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object1
signal1
signal2
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

Figure 12: Signals and slots mechanism of the Qt framework. Different object communicate with each other by emitting signals that are connected to specific slots [41].

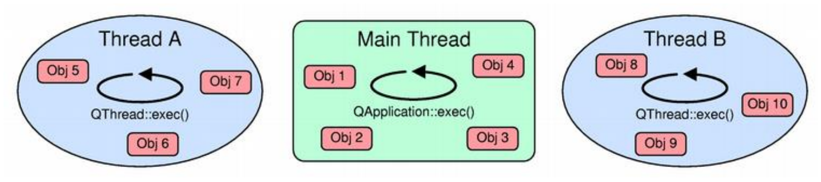


Figure 13: Threading in the Qt Framework. Objects always live in the thread they are created in and can't be directly accessed from other threads [44].

1.6.3. Threading

Threads are an important tool for parallelizing tasks, meaning that different tasks can be executed at the same time. This is especially important for keeping GUIs responsive. As mentioned before, Qt offers different solutions for threading. In this section only the QThread class will be explained, since it was used in this project. Figure 13 pictures the principle of threading. Objects always live in the thread they are created in. You can't delete or access them directly from other threads. GUIs always run in their own thread and therefore can't be accessed from other threads. This is why signals & slots are very important for GUI applications. By creating connections, communication between the GUI and objects from other thread can be established [44].

In order to use the QThread class for threading, the following steps are necessary:

1) QThread myThread;
2) this → moveToThread(&myThread);
3) myThread.start();

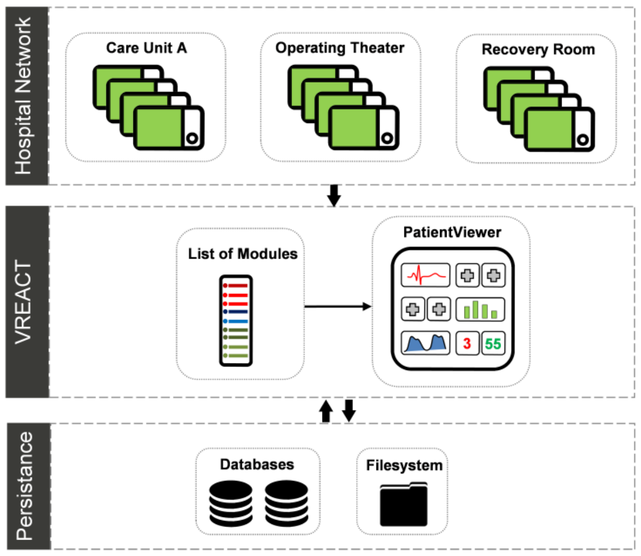


Figure 14: Structure of the project. It consists of data acquisition from the patient monitors followed by a processing step. Afterwards the data is stored and can be visualized by using the PatientViewer feature.

In the first step the QThread object is created. In step two, the object that should be processed in the new thread (referenced by “this”), has to call its function “moveToThread” and pass the QThread object by reference. In the final step the thread has to be started.

1.7. Project structure

The project was split into different isolated tasks in order to allow for independent workflow. Figure 14 summarizes the project structure. My colleague, Mr. Jakub Matta, was responsible for establishing the communication between VREACT and the monitoring devices located in the local network of the hospital. Furthermore he implemented the logic that enables data acquisition and storage (hospital network and persistence sections in the figure). My part was to cover the processing and visualization of the acquired biosignals. This involved the creation of the so called BioSignal class that wraps the incoming information into processable objects and allows for novel clinical parameters (”modules”), and a dedicated PatientViewer that allows for real time visualization capable of different plot types (see VREACT section in the figure). In addition I was responsible for the implementation of several modules for HRV analysis. For this, I implemented an R – peak detection algorithm and various HRV parameters in the time and frequency domain. Apart from that, I developed a modified version called “VREACTquick” which is meant to be used for long term recordings and is capable of managing changes in bed assignments and encrypting patient information.

2. Software architecture

In this chapter, the underlying architecture of our software is explained. The main goals were to keep it readable, reusable and extendable. As depicted in Figure 15, this was achieved by separating the application into three logically independent layers [45]:

- The presentation layer is mainly composed of the main window, which allows the user to connect to beds, start/stop recordings and open the PatientViewer, and the PatientViewer itself, with selectable biosignals (or “modules”) and different widget components for visualization.
- The business layer covers the classes responsible for the management of monitoring devices, structuring of clinical parameters like ECG or SpO2 and the modules that extend these parameters.
- The data access layer utilizes the API of the Infinity Gateway in order to provide functions that establish and manage connections between the client side and the monitoring devices. In addition, this layer manages the recorded data via a filesystem.

2.1. Model – View – Controller design pattern

The Model – View – Controller (MVC) design pattern was used throughout development. The idea of this approach is to decouple components from each other in order to increase code reusability and readability. As seen in Figure 16, the MVC pattern is divided into following parts [46]:

- The controller acts as the “brain” of this configuration. It accepts inputs from the user and can manipulate the view and/or the model.
- The model contains the data and can send a signal when its content changes in order to update the view component accordingly.
- The view is responsible for presenting the data of the model.

MVC was used for every GUI in this application. This means both, the main UI as well as the PatientViewer have their own components that were necessary for keeping this design pattern.

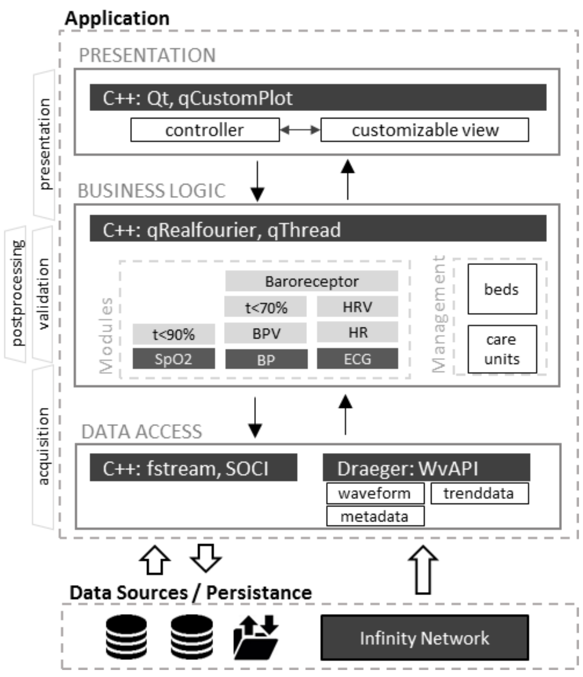


Figure 15: Layer structure of the software. Data of the monitoring devices is acquired over the Infinity Network. The raw signals are further processed within specialized modules (e.g. HRV for ECG). Finally the user is able to store all the data locally or visualize them using the PatientViewer functionality [45].

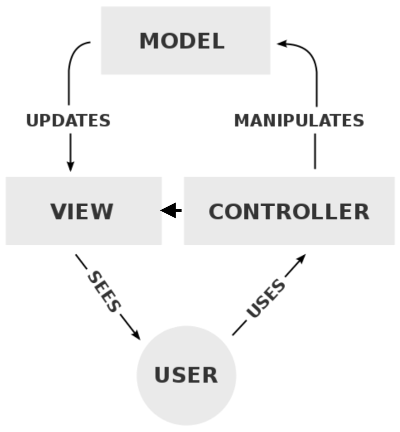


Figure 16: Diagram of interactions within the MVC pattern [47].

2.2. Presentation layer

There are differences in the presentation layer for VREACT and VREACTquick. Both versions have a main UI, with slightly different functionalities, that starts with the execution of the program. VREACT additionally offers the PatientViewer that allows for real – time visualization of biosignals. The version specific features will be explained in the Software implementation section.

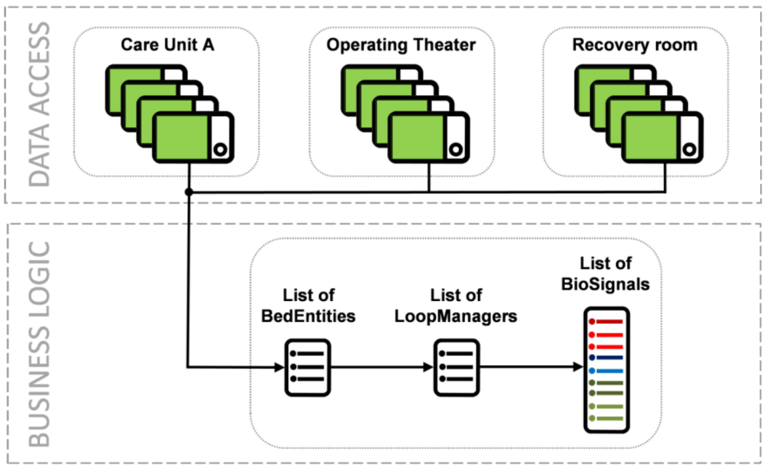


Figure 17: Communication between the data access layer and the business layer. The business logic contains classes that maintain the monitoring devices and the data gathered from them.

2.3. Business layer

As mentioned before, the business layer contains classes for handling of monitoring devices and their respective biosignals.

Figure 17 illustrates the communication between the data access and the business layer. The monitoring units are gathered in a list of bed entities. These entities contain meta information about the patient and the connection status. For each bed entity a loop manager is created. The purpose of the loop manager is to constantly ask the respective monitoring device for new physiological data and, if necessary, to create new biosignal objects. The biosignal class is the heart of the modular design and will be explained in detail in 2.3.3.

2.3.1. BedEntity class

The BedEntity class represents instances of real beds in a hospital. It contains information about the connection status and the following patient metadata:

- Patient name
- Patient ID
- Bed label
- Care unit

The information listed above is sent by a monitor, is accessed via the Dräger API and is used in the UIs for patient identification. As seen in Figure 17, a BedEntity is created for every monitoring unit that is registered. If a successful connection was established the BedEntity object will be provided with its own loop manager.

2.3.2. LoopManager class

Loop manager objects are assigned uniquely to instances of the BedEntity class and run in their own threads. Once a BedEntity has a successful connection status, meaning the monitoring device is communicating with the software, a timer signal starts triggering the slot_getSamples() function of the loop manager object every second. This slot works through the following steps sequentially:

```
1.  for each BioSignal in mapper do
2.      if BioSignal has not received data in the last 30 seconds then
3.          remove BioSignal
4.      end if
5.  end for
6.  for each WaveForm signal requested via API call do
7.      if WaveForm is already in the mapper then
8.          append samples of WaveForm to its BioSignal object
9.      else
10.         instantiate new BioSignal object for the WaveForm signal
11.     end if
12.     call countDown function of BioSignal
13. end for
14. repeat steps 6 to 13 for VitalSign signals
15. if record button was pushed then
16.     start CSV export for received data
```

2.3.3. BioSignal class

The BioSignal class was introduced in order to maintain and extend the biosignals received from the monitoring devices. These devices can deliver two different biosignal types, the WaveForms and the VitalSigns. Biosignals of the WaveForm type are continuous signals with 200 Hz (e.g. ECG), 100 Hz and lower sampling rates. Pre – calculated parameters (e.g. HR or SpO2) are VitalSigns and have a sampling frequency (f_s) of 1 Hz.

Constructors

A BioSignal object can be created by using either one of two constructors. The first one is used for biosignals that are retrieved directly from a monitor:

BioSignal(WvData* p_model, Type t, QStringList plotTypes, QString name, QString labelName, QString xLabel, QString yLabel, int samplingRate, QColor plotColor, QString numUnit, int bufferSize, int plotBufferSize)

The second constructor is called in order to create derived signals:

BioSignal(BioSignal* p_parentSig, QStringList plotTypes, bool plottable, bool exportable, int calcCounter, int bufferSize, int plotBufferSize)

UML class diagram

In order to give a good overview of the interactions between the BioSignal class and other classes, a “Unified Modelling Language” (UML) class diagram was created. The diagram in Figure 18 depicts the relations between objects created from the BioSignal, WvData and LoopManager classes. As already mentioned, the LoopManager is responsible for gathering of biosignal samples and appending them to their according buffers. For that, it recognizes if new signals arrive and appends them to a BioSignal list that can be found in the WvData object (model). Each LoopManager can hold one instance of a WvData object and vice versa. A WvData object holds multiple instances of BioSignals gathered in a list. Furthermore BioSignal objects hold a list of derived signals which explains the relation with itself. The signals and slots of the BioSignal class are hinted in the diagram and will be explained later in more detail.

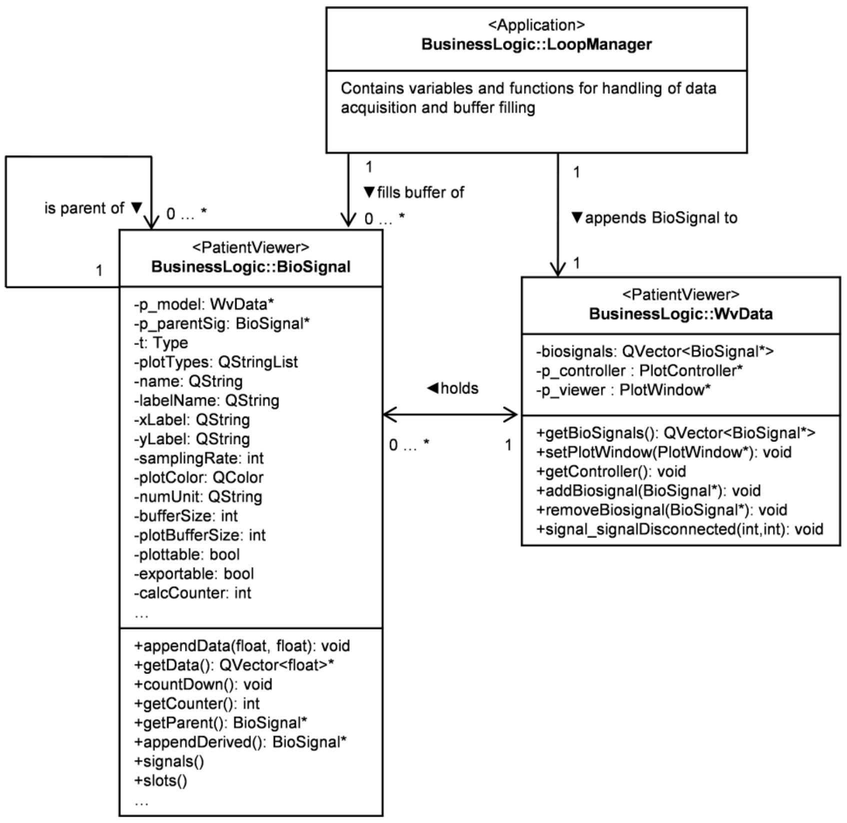


Figure 18: UML class diagram for describing the BioSignal class and its interaction with other classes. The relations between the BioSignal, WvData and LoopManager class were visualized by arrows and numbers. The arrows give information about which class holds the instance in the relationship and the number on both sides depict the number of possible instance that can be generated. Furthermore a brief summary on the relation and its direction is displayed via text and an arrow.

Buffer variables

Every BioSignal object has its own buffer that it has to maintain. Every second the loop manager provides the object with new samples that have to be stored in the buffer. This buffer is limited by the variable bufferSize (in seconds). For example if the buffer for an ECG signal should be limited to 5 minutes, bufferSize is set to 300. By doing so a sample size depending on the f_s is calculated and the buffer starts popping the oldest data points once it exceeds the limit. In addition a variable named plotBufferSize has to be set. This variable does the same as described above but for the buffer of the plots. These two buffers are treated separately, since the plots keep their own internal buffer. If the histogram and the scatter plot were activated, buffers for these plot types will be created and continuously filled.

Parameter calculation

Each derived BioSignal has a calcCounter variable that is subtracted by one after every second. Once this variable equals to 0, it is restored to its start value and the slot_calculateData() function is triggered. This slot function initiates the start of an algorithm that is responsible for the calculation of new samples of a derived biosignal. As an example if the calcCounter variable is set to 5, after every 5 seconds the SDNN parameter will be calculated and stored in its buffer. We refer to these derived BioSignals as “modules” and they will be explained in 2.3.4 in more detail.

Signals & slots

As explained in 1.6.2, signals & slots are used for the communication between objects. In Table 1 signals of the BioSignal class will be listed and described.

Signal	Functionality
signal_appendScatter	Connects to a PlotController slot and triggers filling of the scatter plot data buffer.
signal_appendHisto	Connects to a PlotController slot and triggers filling of the histogram data buffer.
signal_dataArrived	Connects to a PatientViewer slot and triggers drawing of the wave plot.
signal_peakArrived	Connects to a PatientViewer slot and triggers .drawing of detected R – peaks.
signal_scatterArrived	Connects to a PatientViewer slot and triggers drawing of the scatter plot.
signal_histoArrived	Connects to a PatientViewer slot and triggers drawing of the histogram.
signal_numArrived	Connects to a PatientViewer slot and updates the displayed numeric value.

Table 1: Signals of the BioSignal class.

In Table 2 slots of the BioSignal class will be described.

Slot	Functionality
slot_plotData	This slot is used by WaveForm signals coming from the Dräger Monitor. It is triggered by the timeout signal of a timer and plots the biosignal
slot_calculateData	This slot is triggered by a signal that is emitted as soon as the calcCounter variable hits 0. In here the algorithms of derived biosignals calculate new samples.

Table 2: Slots of the BioSignal class.

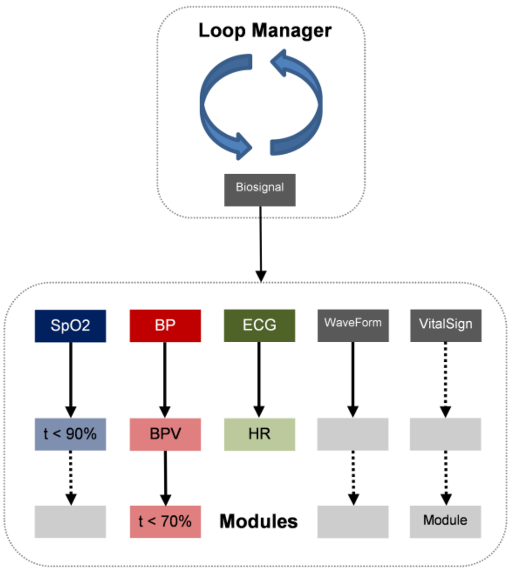


Figure 19: Modular encapsulation of derived biosignals. Whenever the connected monitor delivers new biosignals, the loop manager checks for compatible modules.

2.3.4. Modules

VREACT extends the capabilities of standard monitors by various algorithms that deliver novel clinical parameters. These parameters are encapsulated in so called “modules”, meaning they have their own class that inherits from the BioSignal class, and are instantiated by using the BioSignal constructor for derived signals (e.g. SDNN derived from HR).

Figure 19 shows how biosignals received from the monitoring devices (VitalSigns or WaveForms) are processed by the loop manager. The loop manager checks if there are any compatible modules implemented that can derive from these signals. For example if there is a Hypoxemia module implemented that derives from the SpO2 VitalSign, the loop manager will filter this SpO2 signal from a list of signals and then pass its pointer to the constructor of the Hypoxemia module:

```
1. for each VitalSign requested via API call do
2.         if the label of the VitalSign is SpO2 then
3.                 instantiate a new Hypoxemia object and pass the pointer of the
           SpO2 VitalSign to it
4.         end if
5. end for
```

The Hypoxemia module object will be created using the BioSignal constructor for derived signals. Derived biosignals can be parents of further derived parameters and one signal can have multiple instances of these child signals. If there are no modules for arriving WaveForms or VitalSigns, they will simply not be filtered out of the list and no modules will be instantiated

Module	Parent	Children	Functionality
PeakTime	ECG WaveForm	HeartRate	detects R – peaks
HeartRate	PeakTime	SDNN, RMSSD, SD1, SD2 and HeartRateVariabilityInterpolation	calculates f_c
SDNN	HeartRate	---	calculates SDNN
RMSSD	HeartRate	---	calculates RMSSD
SD1	HeartRate	---	calculates SD1
SD2	HeartRate	---	calculates SD2
HeartRateVariability Interpolation	HeartRate	HeartRateVariabilityFFT	interpolates f_c
HeartRateVariability FFT	HeartRateVariability Interpolation	HeartRateVariabilityVLF, HeartRateVariabilityLF and HeartRateVariabilityHF	performs forward FFT
HeartRateVariability VLF	HeartRateVariability FFT	---	calculates relative spectral power for VLF
HeartRateVariability LF	HeartRateVariability FFT	---	calculates relative spectral power for LF
HeartRateVariability HF	HeartRateVariability FFT	---	calculates relative spectral power for HF
Hypoxemia	SpO2 VitalSign	---	counts cumulative time of hypoxemia

Table 3: Overview of implemented modules.

Table 3 gives an overview of all implemented modules. The software architecture was kept in a way to allow for easy implementation and integration of new modules. For doing so, the module needs a pointer to the signal it is derived from and the algorithm has to be implemented in the slot_calculateData function. After that, the calculation for the module will be triggered every time its calcCounter variable is decremented to 0.

PeakTime

The PeakTime module was implemented as a non plottable BioSignal. This means this module will not show up in the PatientViewer and therefore cannot be selected for plotting. Its purpose is to detect R – peaks from ECG WaveForms (parent signal) delivered by the monitors and then to forward the time stamps of the peaks to its child, the HeartRate module.

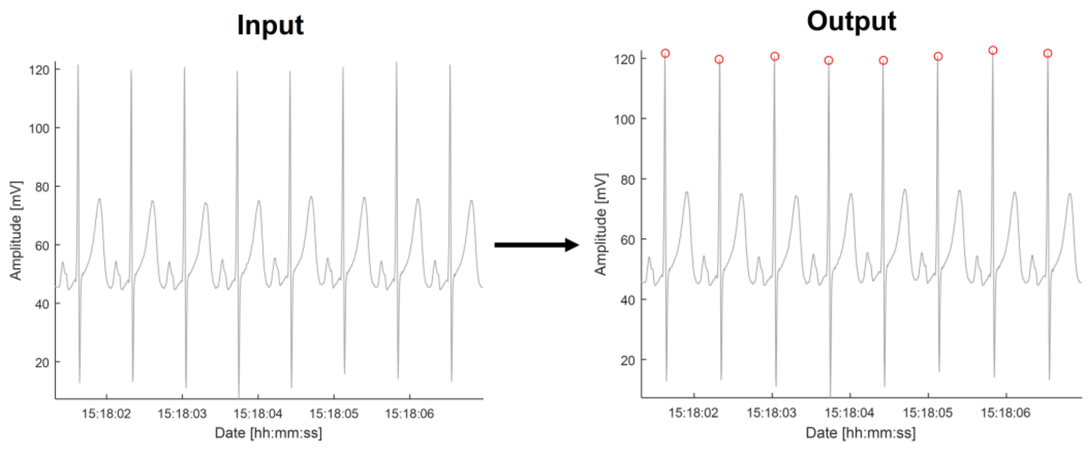


Figure 20: Input-output plot of the PeakTime module. It detects the R-peaks in an ECG signal and saves the time stamps.

The R – peak detection algorithm was provided by Bachler M. and his team from the Austrian Institute of Technology. It detects R – peaks in real time by continuously monitoring the amplitude of the first derivative of the signal. In addition, statistical characteristics of the signal are processed in order to prevent the algorithm from detecting motion artefacts as peaks. If a peak was identified, a classification is initiated that distinguishes between premature ventricular contractions and normal QRS complexes. The provided algorithm can be used for semi online peak detection and has proven to be very robust [48].

Constructor parameter	Value
BioSignal* p_parentSig	pECG … pointer to ECG WaveForm (parent)
QStringList plotTypes	{} … has no plot types
bool plottable	false … can't be plotted
bool exportable	false … not exported to CSV file
int calcCounter	1 … is calculated every second
int bufferSize	10 … buffer contains 10 seconds of data
int plotBufferSize	0 … has no plotbuffer

Table 4: Constructor parameters of the PeakTime module.

Algorithm implementation

```
1. if parent buffer has at least 10 seconds of ECG data do
2.     check for peaks within the last 10 seconds using the provided library
3.     if new peaks are found do
4.         add time stamps of the peaks into the buffer
5.     end if
6. end if
```

Figure 20 shows the input and output of the PeakTime module. As soon as the ECG signal has at least 10 seconds of data in its buffer, the algorithm will start to detect each R – peak.

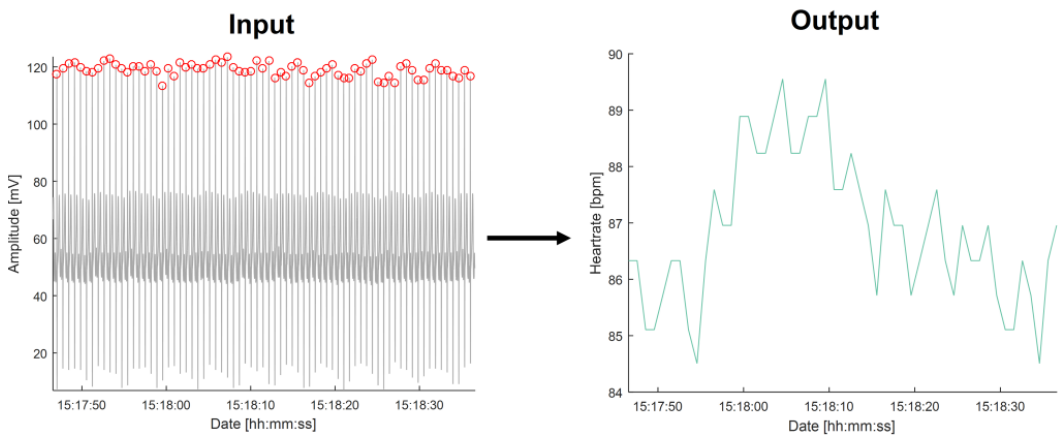


Figure 21: Input-output plot of the HeartRate module. It calculates the heart rate by using the peak times located in the parent module.

HeartRate

The HeartRate module calculates the f_C in beats per minute. The HeartRate module is the parent of following modules: SDNN, RMSSD, SD1, SD2 and HeartRateVariabilityInterpolation.

Constructor parameter	Value
BioSignal* p_parentSig	pPT … pointer to PeakTime module (parent)
QStringList plotTypes	{"wave","numeric","scatter"} … can be plotted as wave plot, numeric value or scatter plot
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	1 … is calculated every second
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 5: Constructor parameters of the HeartRate module.

Algorithm implementation

```
1. if parent buffer has at least 2 samples do
2.     calculate difference between the last two time
       stamps:
       $RR[s] = X_n - X_{n-1}$
3.     convert the value to beats per minute:
       $HR[bpm] = \frac{60}{RR}$
4.     append value to the buffer
5. end if
```

Figure 21 shows the input and output of the Heartrate module. It uses the detected peaks in order to calculate the RR - intervals in seconds and then converts them into bpm.

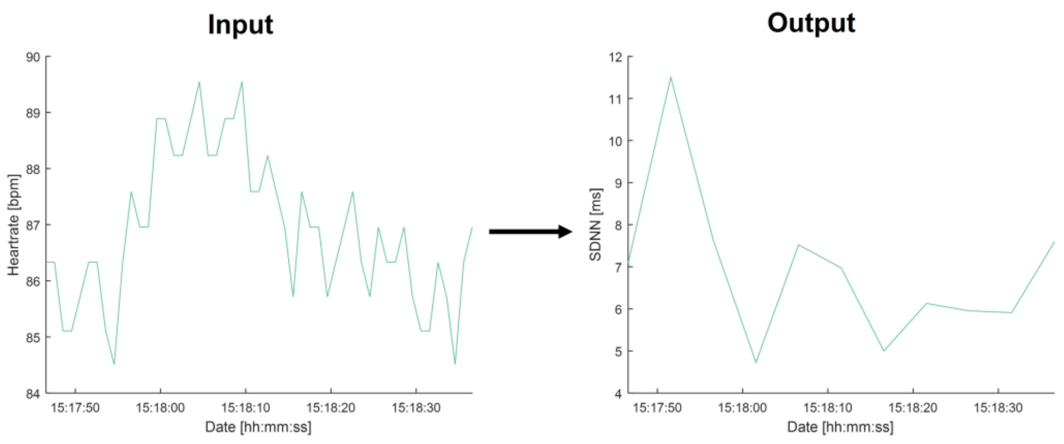


Figure 22: Input-output plot of the SDNN module. It calculates the SDNN from the heart rate data.

SDNN

This module calculates the SDNN parameter for the last 5 minutes of RR – interval data (see Figure 22).

Constructor parameter	**Value**
BioSignal* p_parentSig	pHR … pointer to HeartRate module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 6: Constructor parameters of the SDNN module.

Algorithm implementation

```
1. if parent buffer has at least 5 minutes of data do
2.     calculate the mean value:
       $NN_{mean}\ [s] = \frac{1}{n}\sum_{i=1}^{n} NN_i$
3.     calculate the SDNN parameter:
       $SDNN\ [ms] = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(NN_i - NN_{mean})^2} * 1000$
4.     append value to the buffer
5. end if
```

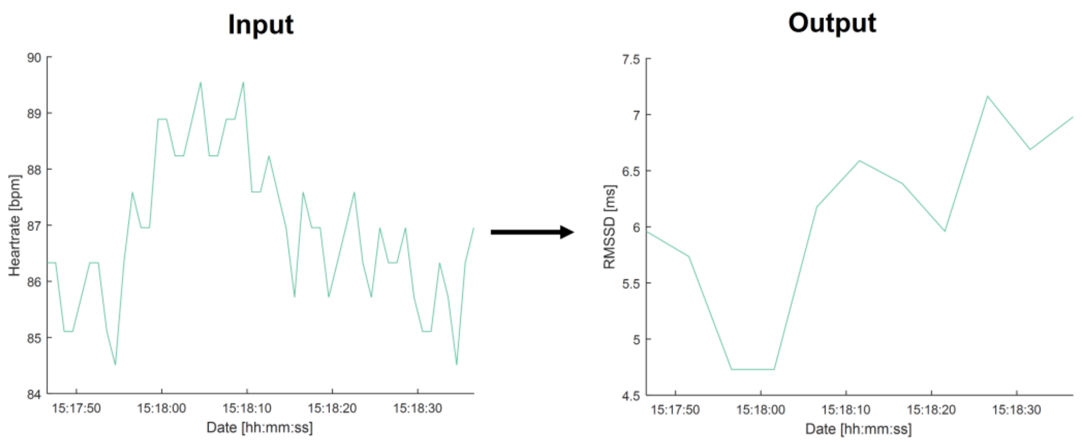


Figure 23: Input-output plot of the RMSSD module. It calculates the RMSSD from the heart rate data.

RMSSD

This module calculates the RMSSD parameter for the last 5 minutes of RR – interval data (see Figure 23).

Constructor parameter	**Value**
BioSignal* p_parentSig	pHR … pointer to HeartRate module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 7: Constructor parameters of the RMSSD module.

Algorithm implementation

```
1. if parent buffer has at least 5 minutes of data do
2.     calculate the RMSSD parameter:
       $RMSSD\ [ms] = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(NN_{i+1} - NN_i)^2} * 1000$
3.     append value to the buffer
4. end if
```

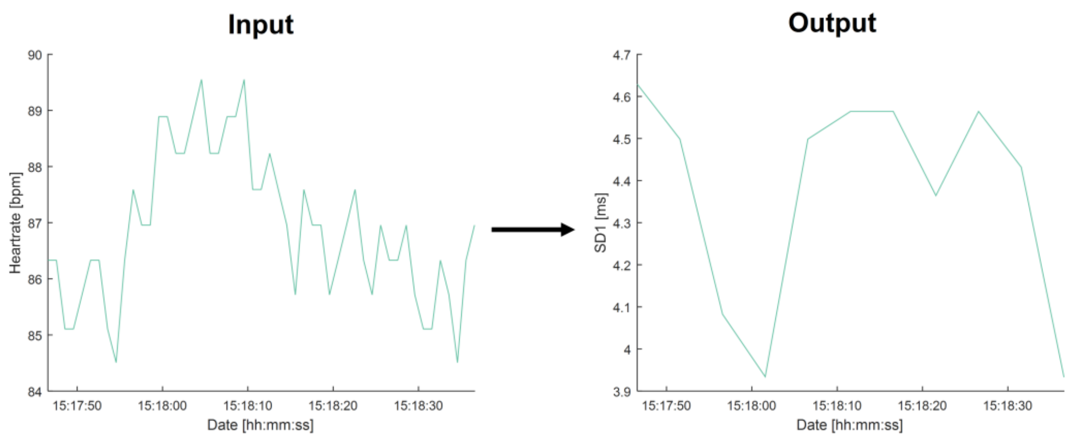


Figure 24: Input-output plot of the SD1 module. It calculates the SD1 from the heart rate data.

SD1

This module calculates the SD1 parameter for the last 5 minutes of RR – interval data (see Figure 24).

Constructor parameter	Value
BioSignal* p_parentSig	pHR … pointer to HeartRate module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 8: Constructor parameters of the SD1 module.

Algorithm implementation

```
1.  if parent buffer has at least 5 minutes of data do
2.      calculate the SD1 parameter:
        $SD1\,[ms] = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\frac{NN_i - NN_{i+1}}{\sqrt{2}}\right)^2} * 1000$
3.      append value to the buffer
4.  end if
```

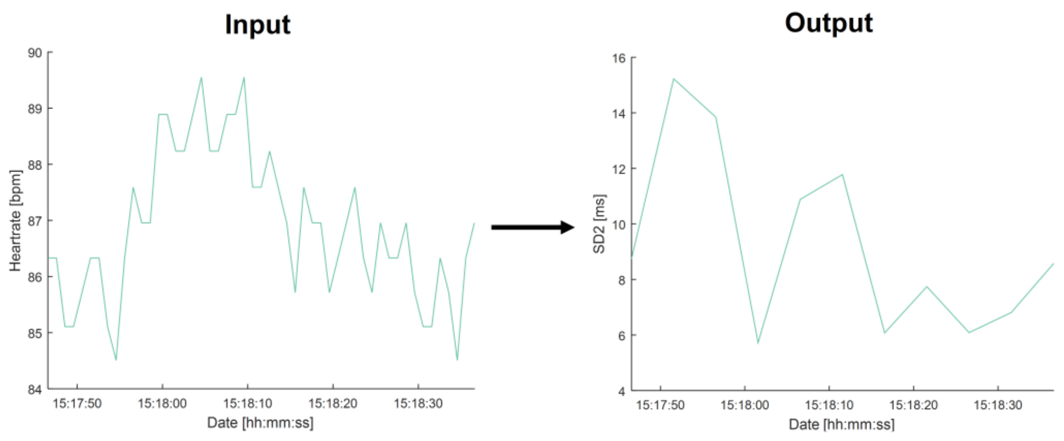


Figure 25: Input-output plot of the SD2 module. It calculates the SD2 from the heart rate data.

SD2

This module calculates the SD2 parameter for the last 5 minutes of RR – interval data (see Figure 25).

Constructor parameter	**Value**
BioSignal* p_parentSig	pHR … pointer to HeartRate module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 9: Constructor parameters of the SD2 module.

Algorithm implementation

```
1.  if parent buffer has at least 5 minutes of data do
2.      calculate the mean value:
        $NN_{mean}\,[s] = \frac{1}{n}\sum_{i=1}^{n} NN_i$
3.      calculate the SD2 parameter:
        $SD2\,[ms] = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\frac{NN_i+NN_{i+1}-2NN_{mean}}{\sqrt{2}}\right)^2} * 1000$
4.      append value to the buffer
5.  end if
```

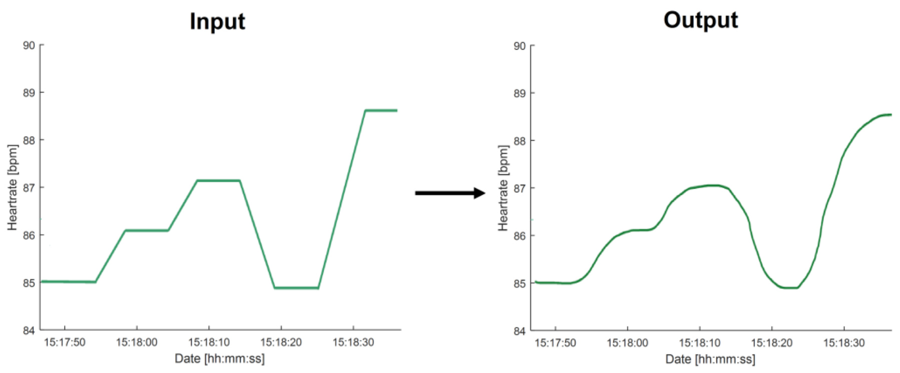


Figure 26: Input-output plot of the HeartRateVariabilityInterpolation module. It interpolates the HR data in order to allow for further FFT calculations.

HeartRateVariabilityInterpolation

The RR interval time series consists of non – uniformly spaced samples. In order to apply FFT on this series, the data needed to be interpolated (see Figure 26). A f_s of 4Hz was proposed for a majority of cases and is appropriate for the study of HRV [49]. The HeartRateVariabilityInterpolation module is not plottable and not exportable. It is used as an intermediate module that provides interpolated data for the FFT analysis. Newton's interpolation algorithm was used, which allowed for a continuous interpolation between the last two samples. This module is the parent of the HeartRateVariabilityFFT module.

Constructor parameter	Value
BioSignal* p_parentSig	pHR … pointer to HeartRate module (parent)
QStringList plotTypes	{} … has no plot types
bool plottable	false … can't be plotted
bool exportable	false … not exported to CSV file
int calcCounter	1 … is calculated every second
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	0 … has no plotbuffer

Table 10: Constructor parameters of the HeartRateVariabilityInterpolation module.

Algorithm implementation

```
1. **if** parent buffer has at least 2 samples **do**
2.     evaluate the first two coefficients of
       Netwon's interpolation polynomial:
       $a_0 = f(x_0)$
       $a_1 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$
3.     loop through time stamps between the last
       sample in the interpolated buffer f($x_0$) and the
       last sample in the parent buffer f($x_1$) in steps
       of 0.25s (→ 4Hz)
4.     in each step calculate the interpolated
       sample:
       $f(x) = a_0 + a_1 * (x - x_0)$
5.     append values to the buffer
6. **end if**
```

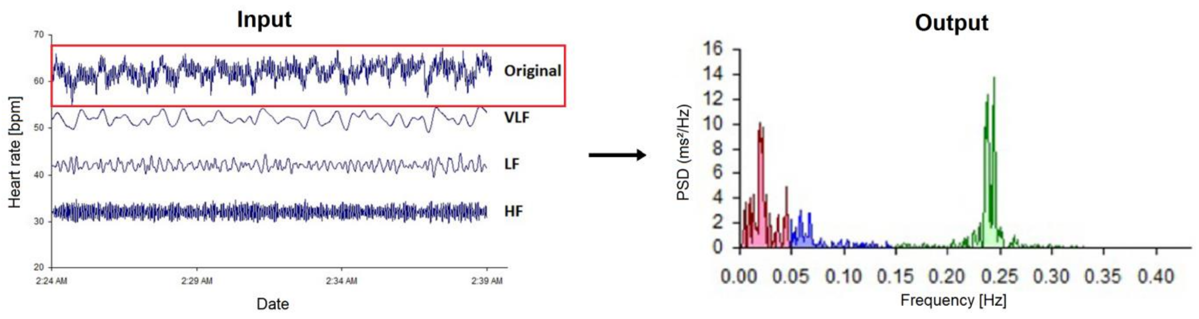


Figure 27: Input-output plot of the HeartRateVariabilityFFT module. FFT analysis is conducted on the interpolated HR data (marked with a red rectangle on the left) which results in a power distribution in the frequency bands. Image adapted from Shaffer et al [24].

HeartRateVariabilityFFT

As described in 1.3.3, HRV can be analysed in the frequency domain. In order to achieve this, a Qt – based C++ library, named QRealFourier, was used. This library provides functions for forward and inverse FFTs [50].

The HeartRateVariabilityFFT module is not plottable and not exportable. As seen in Figure 27, it performs FFT on the last 256 seconds of interpolated HR data. Afterwards it delivers the normalized power spectrum to its children modules HeartRateVariabilityVLF, HeartRateVariabilityLF and HeartRateVariabilityHF.

Constructor parameter	Value
BioSignal* p_parentSig	pHRVI … pointer to HeartRateVariabilityInterpolation module (parent)
QStringList plotTypes	{} … has no plot types
bool plottable	false … can't be plotted
bool exportable	false … not exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	0 … has no plotbuffer

Table 11: Constructor parameters of the HeartRateVariabilityFFT module.

Algorithm implementation

```
1. if parent buffer has at least 1024 samples (256 seconds of data) do
2.     perform forward FFT on the last 1024 samples in the parent buffer using the QRealFourier library
3.     calculate the absolute value of each sample
4.     normalize power spectrum by dividing each sample by the maxima
5.     append normalized samples and according frequencies to the buffer
6. end if
```

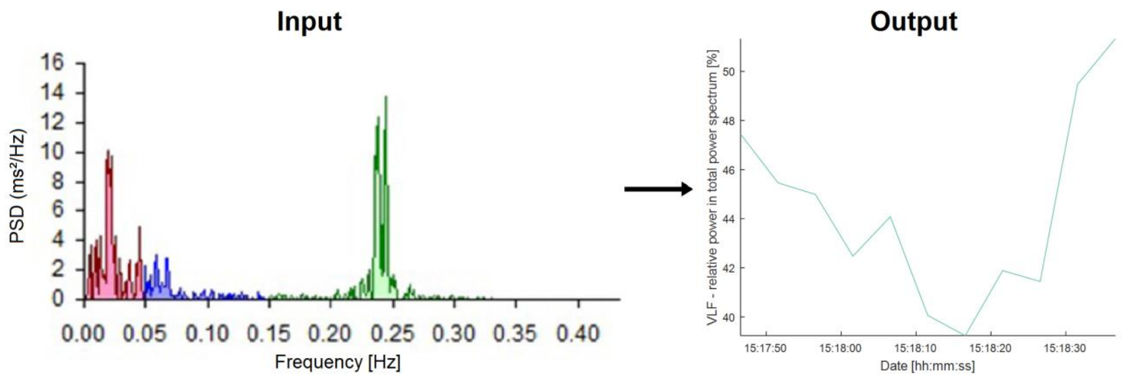


Figure 28: Input-output plot of the HeartRateVariabilityVLF module. This module constantly calculates the relative power in the VLF band from the total power over all bands. Image partially adapted from Shaffer et al [24].

HeartRateVariabilityVLF

This module calculates the relative power of the VLF band for the last 256 seconds of data. The output plot in Figure 28 shows that the power ratio is calculated relative to the total power found in the frequency band from 0Hz to 0.4Hz.

Constructor parameter	**Value**
BioSignal* p_parentSig	pHRVFFT … pointer to HeartRateVariabilityFFT module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 12: Constructor parameters of the HeartRateVariabilityVLF module.

Algorithm implementation

```
1. if parent buffer is not empty do
2.     sum the normalized spectral power $S_p$ for frequencies $f \leq 0.05$Hz:
       $PARTIAL = \int_{0Hz}^{0.05Hz} S_p(f)$
3.     sum the normalized spectral power $S_p$ for frequencies $\leq 0.4$Hz:
       $TOTAL = \int_{0Hz}^{0.4Hz} S_p(f)$
4.     calculate relative power percentage:
       $VLF\ [rel.power] = \frac{PARTIAL}{TOTAL}$
5.     append value to the buffer
6. end if
```

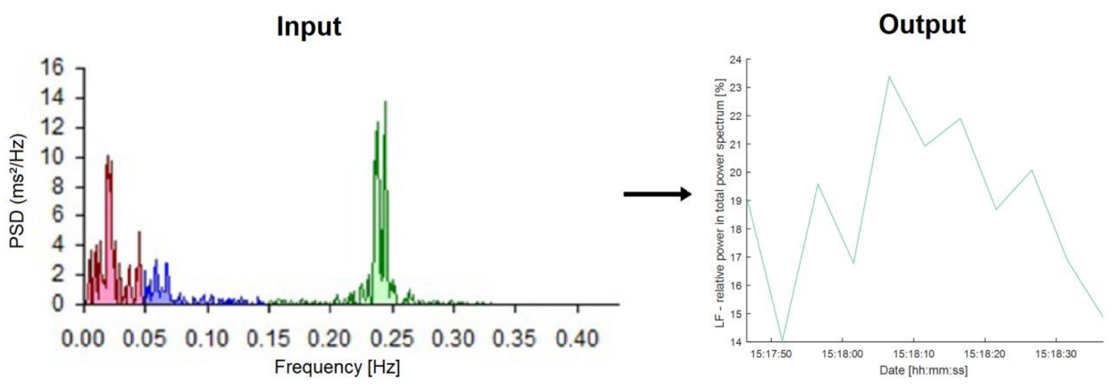


Figure 29: Input-output plot of the HeartRateVariabilityLF module. This module constantly calculates the relative power in the LF band from the total power over all bands. Image partially adapted from Shaffer et al [24].

HeartRateVariabilityLF

This module calculates the relative power of the LF band for the last 256 seconds of data. The output plot in Figure 29 shows that the power ratio is calculated relative to the total power found in the frequency band from 0Hz to 0.4Hz.

Constructor parameter	Value
BioSignal* p_parentSig	pHRVFFT … pointer to HeartRateVariabilityFFT module (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 13: Constructor parameters of the HeartRateVariabilityLF module.

Algorithm implementation

```
1.  if parent buffer is not empty do
2.      sum the normalized spectral power $S_p$ for
        frequencies $0.05Hz < f \leq 0.15Hz$:
        $PARTIAL = \int_{0.05Hz}^{0.15Hz} S_p(f)$
3.      sum the normalized spectral power $S_p$ for
        frequencies $\leq 0.4Hz$:
        $TOTAL = \int_{0Hz}^{0.4Hz} S_p(f)$
4.      calculate relative power percentage:
        $LF\ [rel.power] = \frac{PARTIAL}{TOTAL}$
5.      append value to the buffer
6.  end if
```

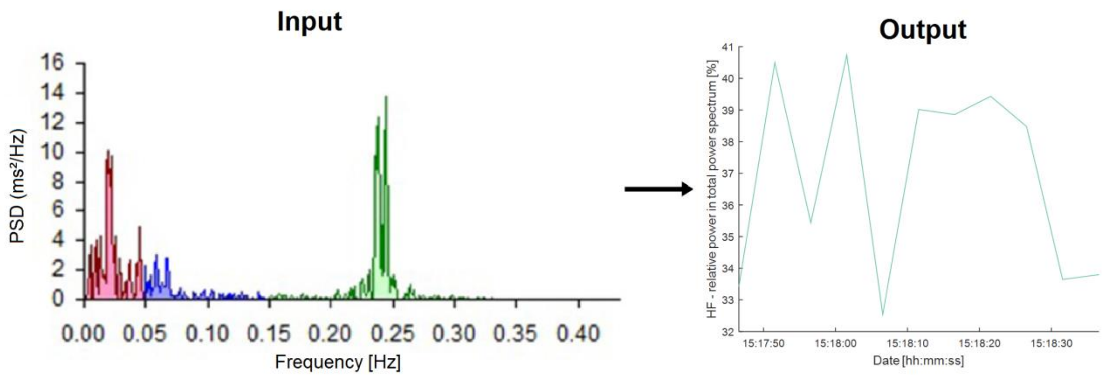


Figure 30: Input-output plot of the HeartRateVariabilityHF module. This module constantly calculates the relative power in the HF band from the total power over all bands. Image partially adapted from Shaffer et al [24].

HeartRateVariabilityHF

This module calculates the relative power of the HF band for the last 256 seconds of data. The output plot in Figure 30 shows that the power ratio is calculated relative to the total power found in the frequency band from 0Hz to 0.4Hz.

Constructor parameter	**Value**
BioSignal* p_parentSig	pHRVFFT … pointer to HeartRateVariabilityFFT module (parent)
QStringList plotTypes	{ "wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	5 … is calculated every 5 seconds
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 14: Constructor parameters of the HeartRateVariabilityHF module.

Algorithm implementation

```
1. if parent buffer is not empty do
2.     sum the normalized spectral power $S_p$ for
       frequencies $0.15Hz < f \leq 0.4Hz$:
       $PARTIAL = \int_{0.15Hz}^{0.4Hz} S_p(f)$
3.     sum the normalized spectral power $S_p$ for
       frequencies $\leq 0.4Hz$:
       $TOTAL = \int_{0Hz}^{0.4Hz} S_p(f)$
4.     calculate relative power percentage:
       $HF\ [rel.power] = \frac{PARTIAL}{TOTAL}$
5.     append value to the buffer
6. end if
```

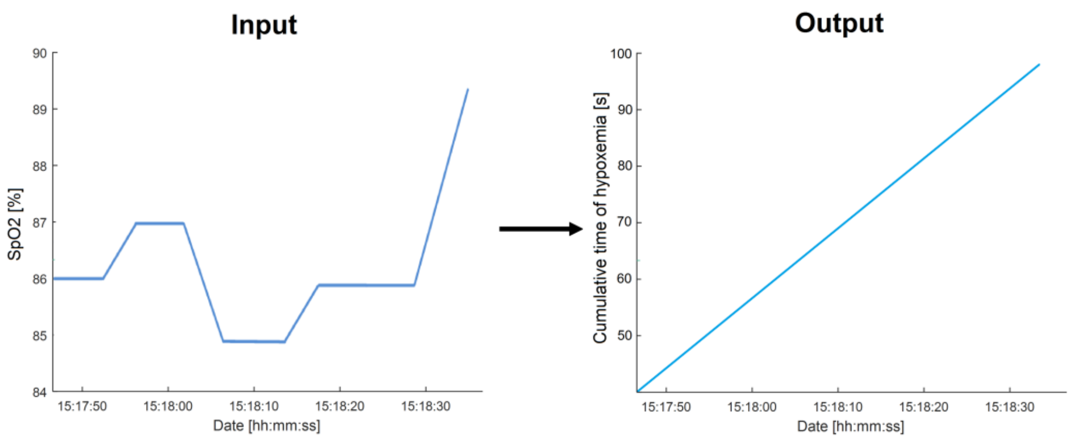


Figure 31: Input-output plot of the Hypoxemia module. This module counts up the seconds in which the patient had SpO2 samples below 90%.

Hypoxemia

The Hypoxemia module calculates the cumulative time of hypoxemia using the SpO2 VitalSign (parent signal) coming directly from the monitor (see Figure 31).

Constructor parameter	Value
BioSignal* p_parentSig	pSPO … pointer to SpO2 VitalSign (parent)
QStringList plotTypes	{"wave","numeric"} … can be plotted as wave plot or numeric value
bool plottable	true … can be plotted
bool exportable	true … exported to CSV file
int calcCounter	1 … is calculated every second
int bufferSize	300 … buffer contains 300 seconds of data
int plotBufferSize	300 … plotbuffer contains 300 seconds of data

Table 15: Constructor parameters of the HeartRateVariabilityHF module.

Algorithm implementation

```
1. if value of the last sample in the parent buffer is < 90 do
2.     append value to the buffer
3. end if
```

2.4. Data access layer

The data access layer was built using the Infinity Delta API. This API is provided in the form of a Dynamic Link Library (DLL), which is a file that contains a shared library of functions and resources. Due to dynamic linkage, different executables can call this information during runtime. The provided DLL is called WvAPI.dll and contains several functions for establishing connections to the monitoring devices and retrieving biosignal samples from them (as seen in Figure 17).

Function	Description
WvStart	Performs necessary initializations.
WvStop	Performs necessary clean up.
WvListBeds	Fills in a list of beds that are currently online at the Infinity Gateway.
WvConnect	Opens a connection to a bed.
WvDisconnect	Closes a connection to a bed.
WvListConnections	Lists beds that are currently connected.
WvListWaveforms	Fills in a list of waveforms that are currently being collected from the bed.
WvDescribeWaveform	Fills in a structure with data for a given waveform.
WvGetWaveformSamples	Fills a buffer with samples for the specified waveform.
WvGetVitalSignsReport	Creates a vital signs report for the specified device.

Table 16: Important functions provided by WvAPI.dll.

2.5. Integration into the Infinity Network

In order to establish a connection, the measurement system had to be setup with Windows Server 2012 and MySQL Server. Furthermore several hardware address filters, firewall rules and Infinity Gateway settings had to be adjusted. Finally, a security dongle provided by Dräger had to be plugged in for unlocking the hardware API [42]. Since repeating these steps for each computer would be too cumbersome, we decided to clone the whole system and convert it into a VirtualBox environment [48]. By doing this, we not only created an easy solution for the distribution of the preconfigured system, but also a back-up for fast recovery.

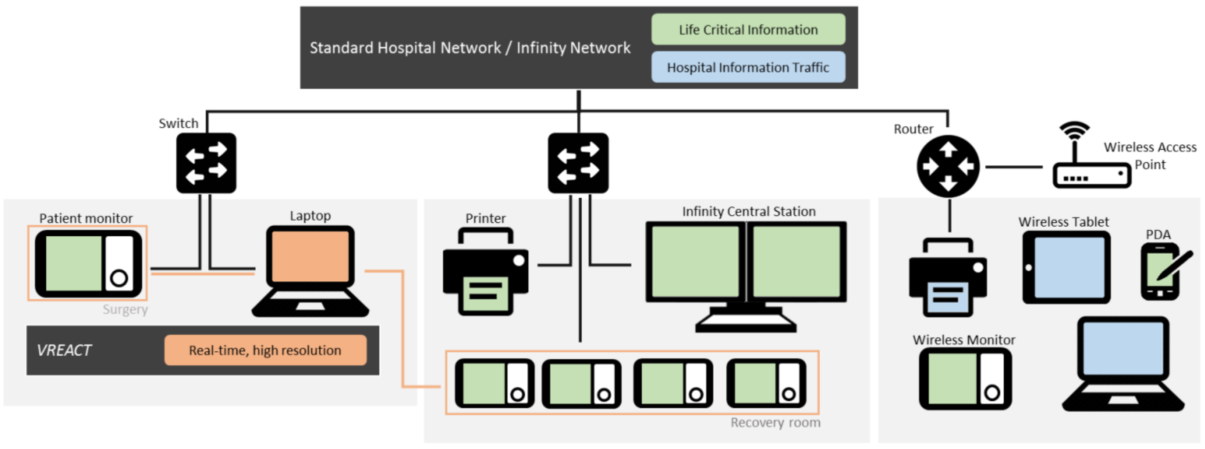


Figure 32: Integration of VREACT into the Infinity Network. VREACT is able to establish connections to all available monitors within this network [45].

Figure 32 depicts the integration of VREACT into the Infinity Network. The secure local network of the hospital is used for data transfer of life critical information (e.g. vital signs for monitoring of patients) and health information traffic (e.g. metadata of patients). This integration allowed us to connect to Infinity Delta monitors within ICUs, operating theatres and recovery rooms.

3. Software implementation

In order to overcome the problems mentioned in 1.5, we developed VREACT, a sophisticated tool that not only enables the continuous recording of biosignals provided by the monitors, but also includes a UI (“PatientViewer”) for visualizing them. Furthermore “VREACTquick”, a more compact version that excludes the PatientViewer, but includes additional features for long term recordings was released.

3.1. VREACT

VREACT is meant to be used for short term recordings (minutes to hours) and continuous and remote visualization of the vital signs of a patients. It allows for connections from single patients up to multiple care units simultaneously. In Figure 33 an illustration of the software structure for VREACT can be seen. Data is collected from beds within care units of the hospital network. Afterwards the raw signals are encapsulated in modules and used as parents of new derived signals. Algorithms are continuously calculating new samples of the derived modules. The final data can be visualized by the PatientViewer and/or can be recorded by saving the sample into a CSV file. Details of the specific parts in Figure 33 were covered in the Software architecture section.

3.1.1. Main GUI

Figure 34 depicts the main GUI of VREACT. It appears with starting of the executable and contains information about available care units, bed labels, patient IDs and patient names. The user is able select whole care units or just single patients by checking the boxes left to their labels. After making the selection, connections can be established by clicking the connect button. By doing so every checked bed that was not connected yet will be connected and every unchecked bed that was already connected will be disconnected. Furthermore the GUI allows for setting of the file directory of recordings and the button to start or stop them. If connections were established successfully, the PatientViewer symbol will be enabled and can be clicked in order to visualize all biosignals of a bed.

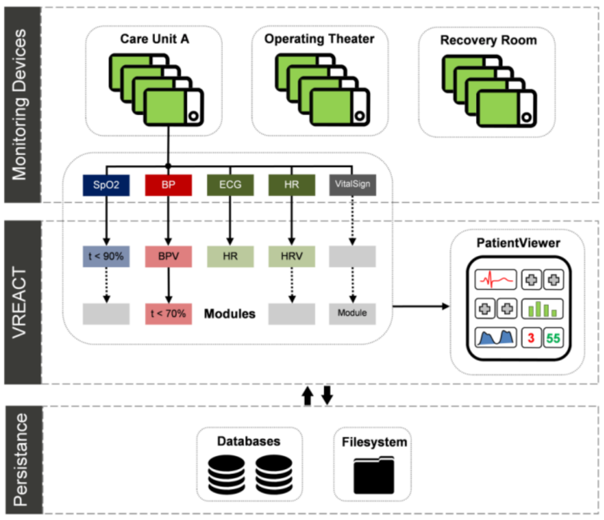
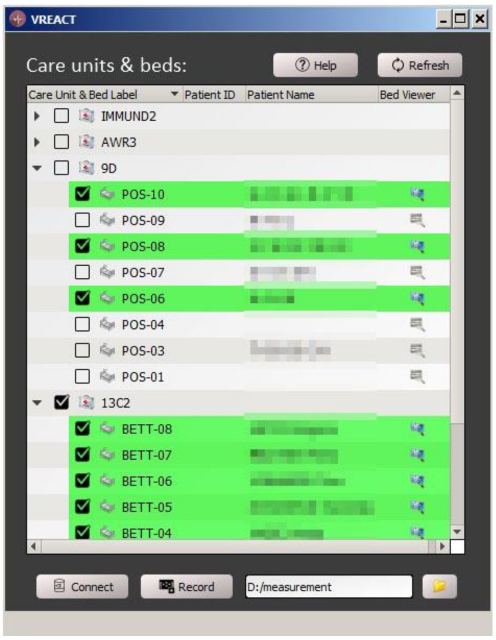


Figure 33: Structure of VREACT. Data of the monitoring devices is acquired over the Infinity Network. VREACT gathers all the available biosignals of a patient in a list. The raw signals are further processed within specialized modules (e.g. HRV for ECG). Finally the user is able to store all the data locally or visualize them using the PatientViewer functionality.


VREACT

Care units & beds:

Help | Refresh

Care Unit & Bed Label	Patient ID	Patient Name	Bed Viewer
☐ IMMUND2			
☐ AWR3			
☐ 9D			
☑ POS-10		[redacted]	
☐ POS-09		[redacted]	
☑ POS-08		[redacted]	
☐ POS-07		[redacted]	
☑ POS-06		[redacted]	
☐ POS-04			
☐ POS-03		[redacted]	
☐ POS-01			
☑ 13C2			
☑ BETT-08		[redacted]	
☑ BETT-07		[redacted]	
☑ BETT-06		[redacted]	
☑ BETT-05		[redacted]	
☑ BETT-04		[redacted]	

Connect | Record | D:/measurement

Figure 34: Main GUI of VREACT. It displays every care unit and their respective beds found within the Infinity Network. If connection to a bed was successfully established, the user can click on the PatientViewer symbol in order to visualize the signals. The patient names were blurred.

3.1.2. PatientViewer

The PatientViewer extends VREACT with real time visualization capabilities. For this it is provided with data stored in the buffers of each plottable BioSignal object. On the left top side all available signals are listed. This list is updated as soon as new signals are detected or existing ones are removed. The user can select a signal simply by clicking on it. Derived signals are accessed by expanding the parent signal. Once a signal is selected, the available plots for this signal will be enabled. The PatientViewer was implemented using a slot

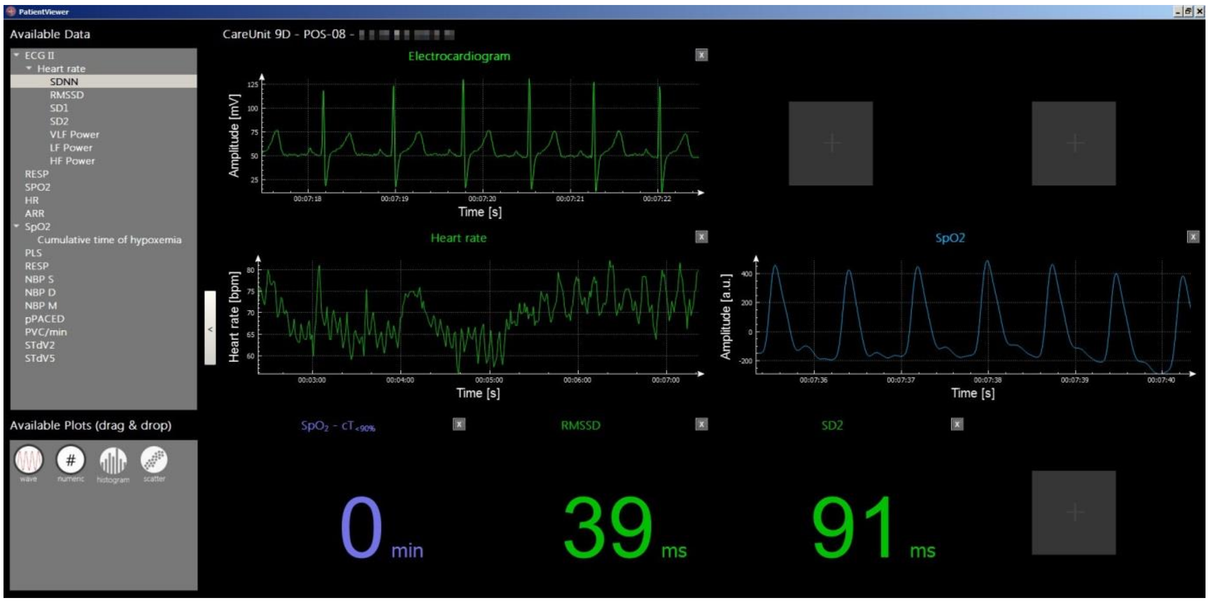


Figure 35: The PatientViewer allows for real time visualization of biosignals. Derived signals can be accessed by expanding the parent signals. In the available plots section the user can choose the type of data representation via drag & drop mechanism. Numeric values take up one slots and graphs take up to slots. The name of the patient was blurred.

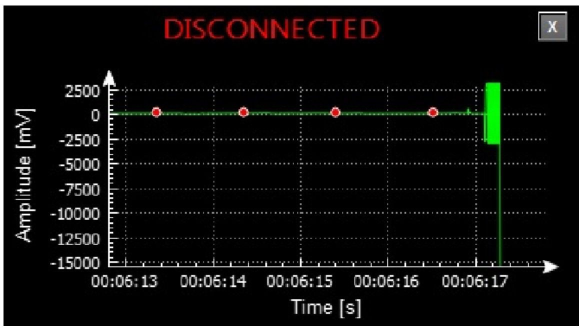


Figure 36: Detection of disconnected signals in the PatientViewer

mechanism. This means that plots take up specific amount of empty slots. Whereas numeric values take up one slot, graphs, like waveforms, histograms or scatter plots, take up two slots. Plotting can be started by dragging and dropping an available plot into the desired slot. The colours of signals as well as their plot labels are set in the BioSignal constructor. Derived signals are assigned slightly lighter colours than their parents. Numeric values are represented by displaying the last value in the buffer of the signal. Their units are either directly set by the information delivered by the monitor or are within the constructor of a derived BioSignal. Wave plots have a buffer of up to 5 minutes. Once a signal exceeds this limit, the first samples in its buffer are removed and new samples are appended. This allows for smooth and continuous plotting. The drag & drop mechanism allows for overwriting of already taken slots. Doing so will close and clear the plot buffer of the former plot. In addition to that the user can close a plot by simply clicking on the X symbol on the right of the label name.

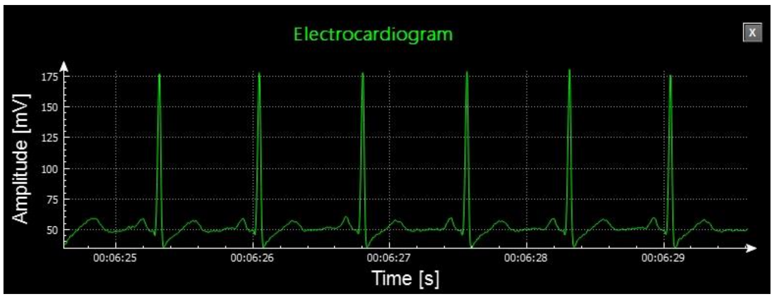


Figure 37: An ECG plotted by the PatientViewer. The y – axis shows the amplitude in mV and the x – axis contains time information. This plot type is called “Wave”.

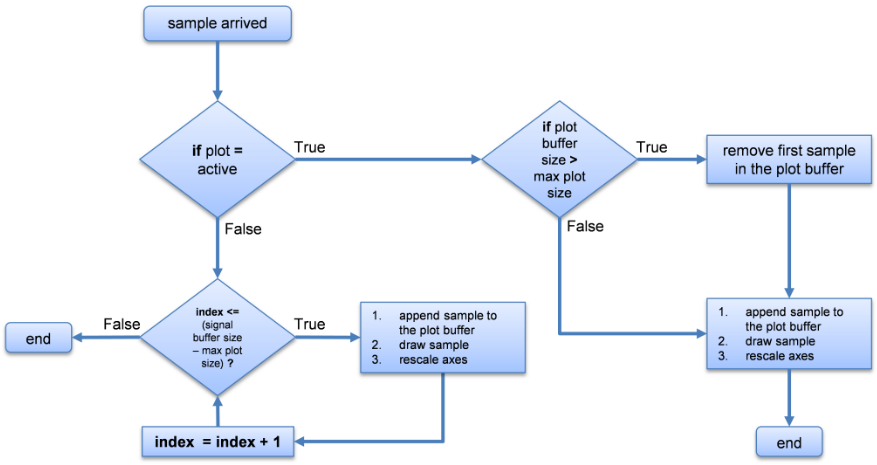

sample arrived
if plot = active
True
False
if plot buffer size > max plot size
True
False
remove first sample in the plot buffer
index <= (signal buffer size – max plot size) ?
False
True
end
1. append sample to the plot buffer
2. draw sample
3. rescale axes
1. append sample to the plot buffer
2. draw sample
3. rescale axes
index = index + 1
end

Figure 38: Flow diagram of the wave plots. After a sample arrives, the logic checks whether a plot is already active or not. If it was active already samples will simply be added to the plot and if not, a new plot will be created and filled with all recent samples.

In Figure 36 the automatic detection of removed biosignals can be seen. As soon as a sensor stops delivering data for more than 5 minutes, it will be removed from the list of available data and all active plots related to it will display “DISCONNECTED”.

Plot types

As already mentioned, signals can be represented by different plot types. In this section each type will be explained in more detail.

Wave plots

The wave type is used for continuous plotting of signals and takes up two slots. In Figure 37 an ECG signal plotted as a wave type can be seen. The amount of visualized samples is defined by the plot buffer size. For ECG signals this size is set to 6 seconds. Figure 38 describes the executed steps in the form of a flow diagram.

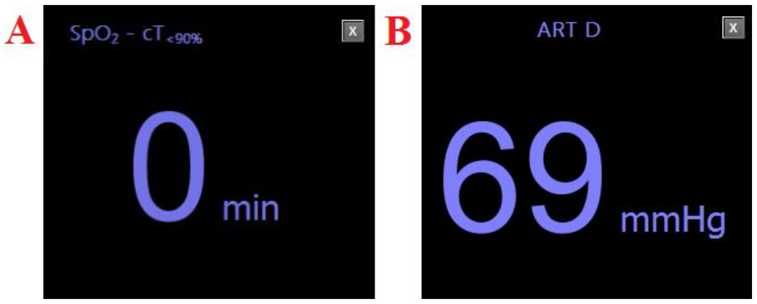


Figure 39: Numeric values displayed by the PatientViewer. Picture A shows the cumulative time of hypoxemia in minutes and picture B shows the diastolic blood pressure in mmHg.

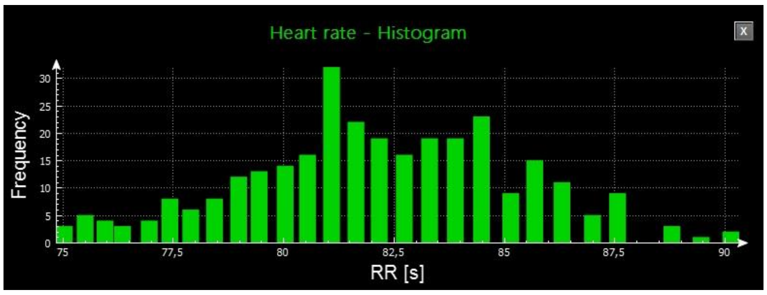


Figure 40: Histogram of heart rate samples visualized by the PatientViewer. The y – axis contains information about the frequency and the x – axis depicts the values.

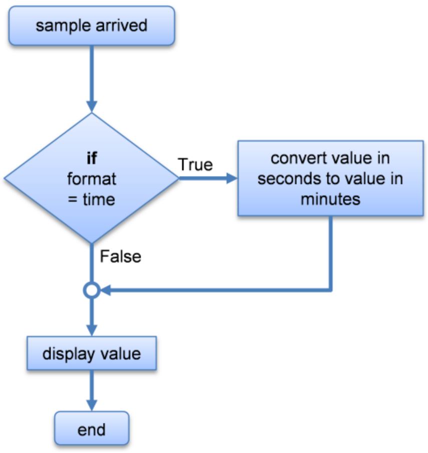
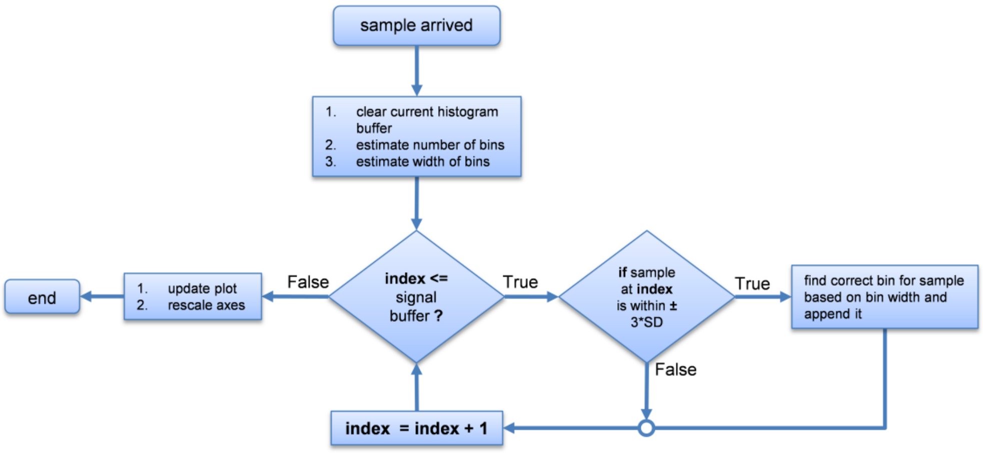


Figure 41: Flow diagram of numeric plots. After a sample arrives, the logic checks whether the value should be displayed in the time format or not. If it has time format, the value will be converted from seconds to minutes before being displayed.

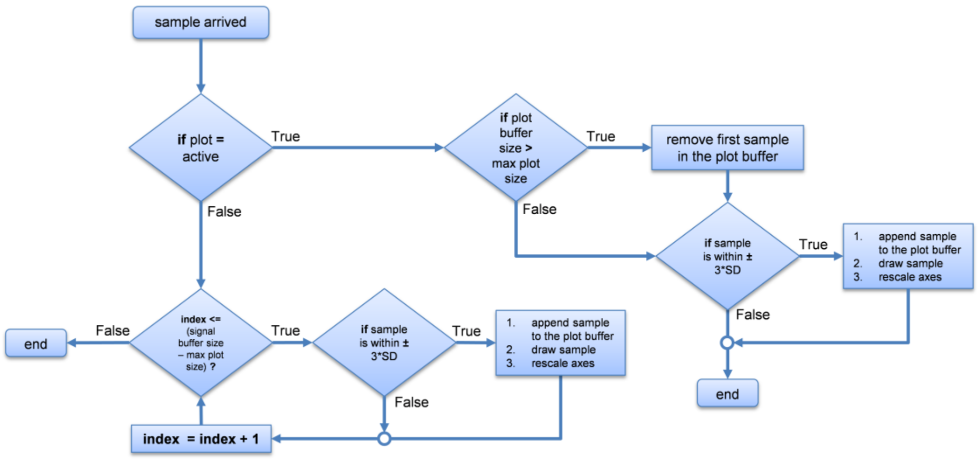
Numeric plots

The numeric type is mainly used for VitalSign signals coming from the Dräger monitors. This type takes up one free slot and continuously displays the most recent value in the signal buffer. Figure 39 shows the cumulative time of hypoxemia in minutes and the diastolic blood pressure in mmHg. Both parameters are visualized as numerics. Figure 41 describes the executed steps in the form of a flow diagram.



sample arrived
1. clear current histogram buffer
2. estimate number of bins
3. estimate width of bins
index <= signal buffer ?
True
if sample at index is within ± 3*SD
True
find correct bin for sample based on bin width and append it
False
False
1. update plot
2. rescale axes
end
index = index + 1

Figure 42: Flow diagram of histogram plots. After a sample arrives calculations are started in order to create bins and append samples based on the bin width.



sample arrived
if plot = active
True
if plot buffer size > max plot size
True
remove first sample in the plot buffer
False
if sample is within ± 3*SD
True
1. append sample to the plot buffer
2. draw sample
3. rescale axes
False
end
False
index <= (signal buffer size – max plot size) ?
False
end
True
if sample is within ± 3*SD
True
1. append sample to the plot buffer
2. draw sample
3. rescale axes
False
index = index + 1

Figure 43: Flow diagram of the scatter plots. After a sample arrives, the logic checks whether a plot is already active or not. If it was active already samples will be added to the plot and if not, a new plot will be created and filled with all recent samples.

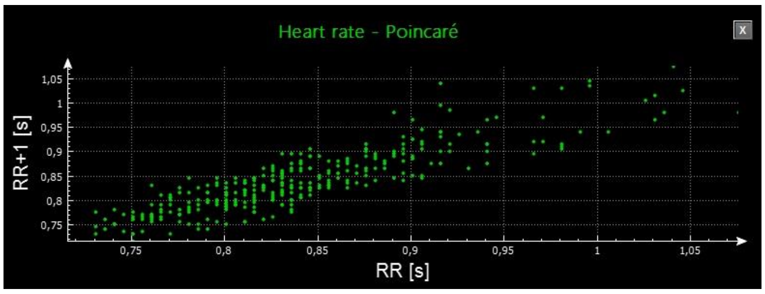


Figure 44: Scatter plot (Poincaré plot) of heart rate samples visualized by the PatientViewer. The y – axis depicts RR+1 intervals and the x – axis depicts RR intervals.

Histogram

The histogram plot visualizes the frequency of certain samples in real time and occupies two slots. Figure 40 pictures a histogram displayed by the PatientViewer. The distribution of the histogram contains information about the HTI as explained in 1.3.2. Figure 42 describes the executed steps in the form of a flow diagram.

Scatter plot

The scatter plot contains information on the relation between successive RR intervals and it occupies two slots. Figure 44 shows a Poincaré plot displayed by the PatientViewer. As described in 1.3.2, the distribution of the sample cloud can help identifying cardiac diseases. Figure 43 describes the executed steps in the form of a flow diagram.

Drag & drop mechanism

In order to allow the user to choose between various plot types and the slots for visualizing them, a drag & drop mechanism was introduced. If no signals are selected the available plot types stay disabled. Once a user makes his selection, the types that are compatible become enabled. The plot symbols have to be dragged into the dedicated slots. If the cursor is not in the vicinity of a slot while dropping, no event will occur. If the symbol is dropped over an already active slot, the former plot will be replaced with the new one and its plot buffer will be cleared.

3.1.3. Libraries

QCustomPlot

In order to implement customizable and dynamic plots, the QCustomPlot library was used [51]. This library is a Qt C++ widget that is well documented. By promoting Qt widgets in the UI creator to QCustomPlot objects, all the functionality within the library can be accessed. This enables creating sophisticated plots that can quickly adapt to changes in the dataset.

3.1.4. Debugging

Since VREACT establishes connections by using external libraries, the behaviour was sometimes very hard to understand. In order to gain more information, the software was debugged extensively. Debugging was mainly split in two different parts: under simulated conditions by establishing connections to a test monitor, and under real conditions in the network of the General Hospital of Vienna.

The simulated conditions mainly helped us to understand the behaviour of functions within the WvAPI library and the exception handling. We discovered that connections to beds were lost as soon as the communication timeout reached a certain threshold. On top of that, Dräger delivered new IDs for these beds afterwards. As a counter measure we repeatedly ask for connection details of each connected bed. This ensures that beds stay connected and keep their IDs throughout the execution of the software.

Debugging under real conditions was mostly for identifying problems regarding performance and multi-threading. Here we saw that connecting to too many beds at the same time had impacts on the PatientViewer, since it was not able to keep up with updating the plots. Furthermore the log file caused a crash due to different threads trying to write into the file simultaneously.

3.1.5. Test results

VREACTs recording capabilities were tested by my colleague Mr. Jakub Matta, who successfully conducted a 2 hour recording from 3 different care units with a total of 15 patients. The PatientViewer was tested during various conditions by me. Besides smaller UI tests that verified basic functionality and responsiveness, the viewer passed 1 hour long visualization tests of data from 5 different patients. During those tests different parameters were plotted and often times closed or replaced by other parameters. Although final tests were successful, the PatientViewers performance drops if VREACT is connected to multiple beds at the same time. The test protocol of VREACT can be found in the appendix.

3.2. VREACTquick

VREACTquick is a modified version of VREACT that offers handling of long term recordings and name encryption. In this version connections are only allowed to care units but not to single beds. VREACTquick introduces handling of changes in the list of patients. For long term recordings it is crucial to identify new assignments of patients to beds. If a patient was disconnected for a while but returns to his bed, his data will be stored in the same folder as it was before. If the software recognizes that a new patient is assigned to a bed, it will clear all the memory that was allocated for the old patient and then will create a new BedEntity. By doing so, memory leaks are prevented. Furthermore if a new bed is added, it will be instantly connected or recorded if its care unit was already connected or being recorded. The algorithm works through the following steps sequentially:

```
1. if list of beds contains new bed then
2.         if bed information is equal to that of an already existing bed then
3.                 delete the old BedEntity
4.         end if
5.         create a new BedEntity
6.         if already existing bed was connected then
7.                 connect the new BedEntity
8.         end if
9. end if
```

Since the architecture is based on VREACT, this section will only cover the extra features.

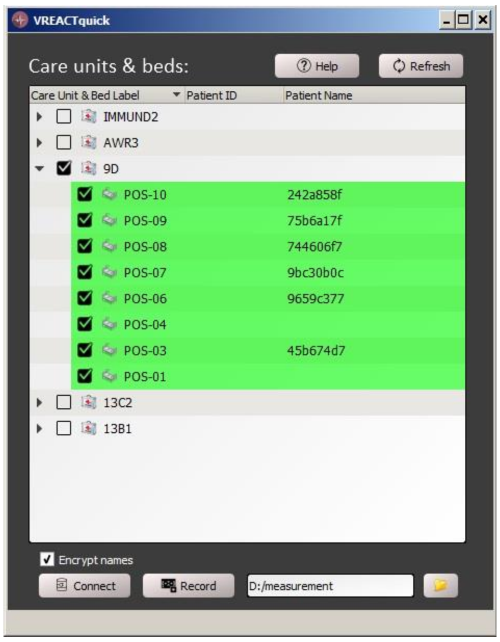
3.2.1. Main GUI

The main GUI is structured similarly to VREACT. As seen in Figure 45, in this version patient names can be encrypted by checking the box above the connect button. The names are encrypted by using a SHA-1 function and are represented by 8 digits. In order to encrypt the names in recordings, the encryption box has to be checked before starting to record. Additionally, the VREACTquick GUI does not allow for connections to single beds. The user can only use the checkboxes left to the care unit labels.

3.2.2. Libraries

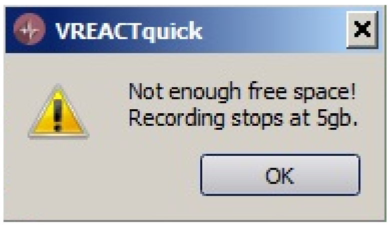
libHaru

In order to allow for name decryption in the recordings, the libHaru library was used [52]. This library provides methods to create password protected PDF files. Each time a new file directory for a bed is created, VREACTquick will check if name encryption is enabled and if it is, a protected PDF file will be created.



Care Unit & Bed Label	Patient ID	Patient Name
IMMUND2		
AWR3		
9D		
POS-10		242a858f
POS-09		75b6a17f
POS-08		744606f7
POS-07		9bc30b0c
POS-06		9659c377
POS-04		
POS-03		45b674d7
POS-01		
13C2		
13B1		

Figure 45: Main GUI of VREACTquick. It extends the VREACT GUI by a new checkbox that enables name encryption of patients. Furthermore the user can only select whole care units.



Not enough free space!
Recording stops at 5gb.

Figure 46: VREACTquick stops recording if there is not enough free disk space.

3.2.3. Debugging

Debugging VREACTquick was a difficult task. Since this tool had to pass long-term recording tests within the hospital, continuous tracking of events was necessary. For this, the logged events were reduced by a significant amount in order to avoid crashes due to multi-threading. After a while memory leaks were recognizable. This issue was caused due to buffers that were filled for the PatientViewer, but since this feature was excluded, the data was never accessed and cleared. Apart from that, pointers to BedEntities were lost before the memory was cleared which again resulted in a memory leak.

Another task was to ensure that the disk has enough free space for further recordings. In order to introduce a safety mechanism, the windows API was used to check the current free disk space on the chosen file directory. Figure 46 shows the warning message that pops up as soon as the free space reaches 5 gigabytes. This will also stop the ongoing recording.

3.2.4. Test results

VREACTquick passed an endurance test that consisted of non-stop data recording from two different care units with over 20 patients. The recording was stopped after approximately 82 hours and collected about 35 gigabyte of data. The appendix section contains plots of different biosignals of one patient. These plots show that VREACTquick is capable of high resolution recordings (e.g. ECG signals with f_s of 200Hz) as well as low resolution recordings (e.g. HR signals with f_s of 1Hz or SDNN signals with f_s of 0.2Hz). In addition, the tool succeeded in encrypting the patient names and detecting changes in the assignment of patients to beds in order to create new file directories. The test protocol of VREACTquick can be found in the appendix.

Validation of the R - Peak detection algorithm

In order to validate the performance and accuracy of the implemented R – Peak detection algorithm, the calculated f_C was compared to the f_C delivered by the Dräger monitors. For this, the agreement of the two datasets was examined using a Bland – Altman (BA) difference plot. It needs to be mentioned that the f_C of the Dräger monitor is always an averaged value over a few seconds, which causes it to miss sudden spikes or drops. As a result of that both data sets needed to be averaged by a 5 minute window in order to make them more comparable. In addition the Dräger f_C is usually delayed by a few seconds and therefore does not allow for direct comparison. Because of this, the data sets were synchronized as accurate as possible using cross-correlation. Figure 47 shows the BA plots for 4 different patients. The plots contain information about differences between samples (y – axis) and about the mean value of each of these difference pairs (x – axis). Furthermore the BA plots contain information about the bias ("Median of Δ" line) and the limits of agreement (LOA, “97.5 quantile” and “2.5 quantile” dashed lines) which include 95% of all data. The percentiles were used instead of SD since the data is not normally distributed. In general the LOA are rather small which means the differences between the samples are small. However, the datasets also contain bigger outliers. This was mainly due to ECG sensors being detached from the patient which affected the peak detection of the algorithm while the monitor seemed to be able to deliver correct values.

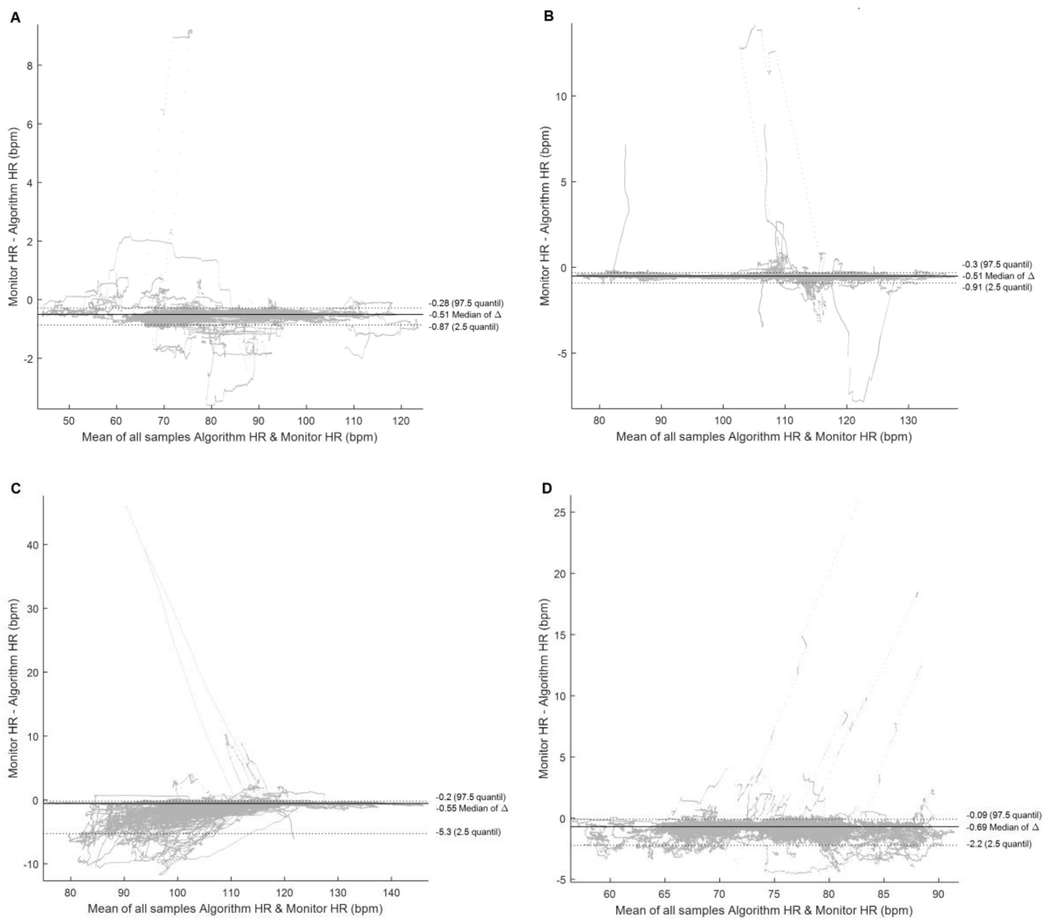


Figure 47: Bland – Altman plots of 4 different patients. On the y – axis the differences between the HR samples provided by the monitors and the HR samples calculated by the implemented algorithm ($s_{monitor} – s_{alg}$) are displayed. The x – axis shows the mean value of each sample pair ($s_{monitor} + s_{alg}$ / 2). The bias (median of all Δ, solid line) and the limits of agreement (2.5 quantil and 97.5 quantil, dashed lines) are also calculated and visualized.

Patient	**A**	**B**	**C**	**D**
Bias (median of Δ**)**	-0.51	-0.51	-0.55	-0.69
Upper LOA	-0.28	-0.3	-0.2	-0.09
Lower LOA	-0.87	-0.91	-5.3	-2.2
95% confidence interval	0.59	0.61	5.1	2.11

Table 17: Bias (median of Δ), limits of agreement and the 95% confidence interval of patients A, B, C and D.

Table 17 lists the results of the BA analysis. Overall the bias in each dataset is negative meaning the f_C samples of the algorithm are slightly higher than the f_C samples of the monitor. This is due to the initial averaging behaviour of the monitors. By averaging the samples, sudden spikes are not registered leading to an overall lower f_C signal. Whereas the 95% confidence intervals for patients A and B are very small, for patients C and D they are bigger. As mentioned before, this was mainly due to disturbances on the patient bed side like motion artefacts and detachment of sensors.

4. Discussion

The objective of this work was to develop software for recording and monitoring of biosignals through the safe hospital network. While the focus was on providing HRV related parameters, the software architecture was kept in a way that allows for implementation of new modules very easily. During the course of this thesis a modified version was released which solely focuses on long term recordings while encrypting personal patient information.

The development process started in the facilities of TU Wien, where we had direct access to a Dräger monitor. During this period we were able to change and adjust the settings of the monitor and test the output of the Dräger API. After the basic functionality was implemented and all tests with one monitor succeeded, we transferred our setup to the General Hospital of Vienna where we had real conditions with several care units and beds.

Throughout development we overcame several difficulties, some of them specific to the Dräger API. As an example, accessing the internal buffer of the Dräger monitors turned out to be rather complicated. We realized that the monitors will always release and clear the content of their whole buffer when data is requested. This means the amount of returned samples depends on the time difference between two requests. In addition to that, a lower limit of 40 samples locks the buffer. This means if the monitor has not stored at least 40 samples, it will not deliver anything. By adjusting the request frequency to 1 second, we were able to maintain a constant size (about 200 samples) of the internal Dräger buffer.

Connections to beds were established by using functions provided by the Dräger API. In early development there were issues with keeping connections alive. After a successful connection, Dräger delivers a unique bed ID. If there is no communication to this bed for a certain amount of time, the connection is not maintained anymore. The big issue with this was that the monitor changed the bed ID when the list of beds was requested. Therefore we had to make sure to not allow the IDs to change. This was done by setting up a timer that would continuously exchange data with the connected bed.

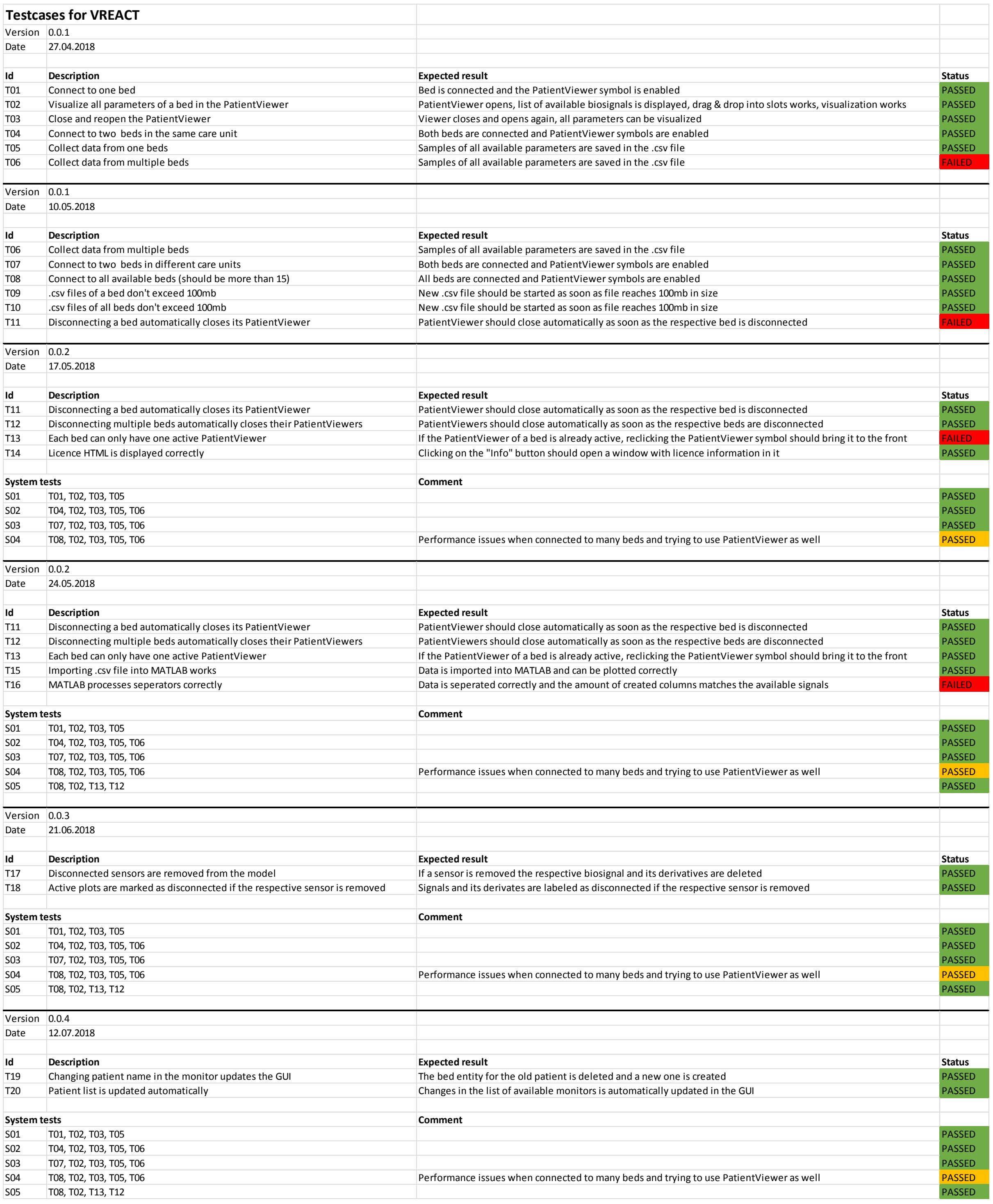
Since C++ does not offer a garbage collector, the code is usually very prone to memory leaks. It was important to track if derived signals would automatically be deleted if their parents are deleted. This was tested successfully in various scenarios with multiple derived parameters and also non exportable signals. Moreover BedEntities and their dedicated LoopManagers had to be cleared if a certain bed was removed from the list. During the long-term test of VREACTquick a big memory leak was detected due to a buffer that was supposed to fill up plots and then clear itself. Since this version had no PatientViewer, the buffers were not cleared leading to a big memory leak that crashed the software after a few hours. In the end all detected memory leaks were fixed.

Our software introduces HRV as a novel parameter to the perioperative environment. The user can record or visualize parameters like SDNN, RMSSD, SD1, SD2, VLF-, LF-, or HF – Power in real time. This enables healthcare professionals to make proactive decisions based on the information that VREACT offers them. Furthermore our tool can be used in studies in order to gather huge amount of data over long periods of time. Not only will it record basic vital signs delivered by the monitors, but also the data stored in the buffers of the derived parameters. Compared to other proposed tools, VREACT does not need a physical connection to the monitoring devices. The only requirement is a stable connection to the hospital network. Because of this, VREACT can connect to multiple monitors simultaneously.

For testing under real conditions we cooperated with the General Hospital of Vienna. There we had access to multiple care units and many beds within the local network. VREACTquick was able to record data from two different care units simultaneously over a period of about 82 hours. VREACT successfully recorded data from over 15 patients from 3 different care units for over 2 hours. Furthermore the PatientViewer passed one hour long plot tests for 5 different patients. The implemented R – peak detection algorithm was validated by comparing the calculated f_C to the monitor f_C using the Bland – Altman analysis. The validation was done with the data of four different patients. Overall the vast majority of the samples (> 95%) were within the limits of agreement. Outliers occurred due to different approaches in the calculation of f_C between the monitors (average values over a few seconds) and our algorithm (saves each RR – interval separately), and the detachment of ECG sensors which causes wrong peak detections of the algorithm for a short time. Another problem arises when a user connects to multiple beds simultaneously. By doing so, the PatientViewer starts to suffer from performance issues. The plots can’t update regularly anymore which in the worst case leads to empty plots. This issue depends on how many signals each bed has and how many algorithms are actively calculating. In conclusion the functionality of our software is very satisfactory but there is still room for improvement.

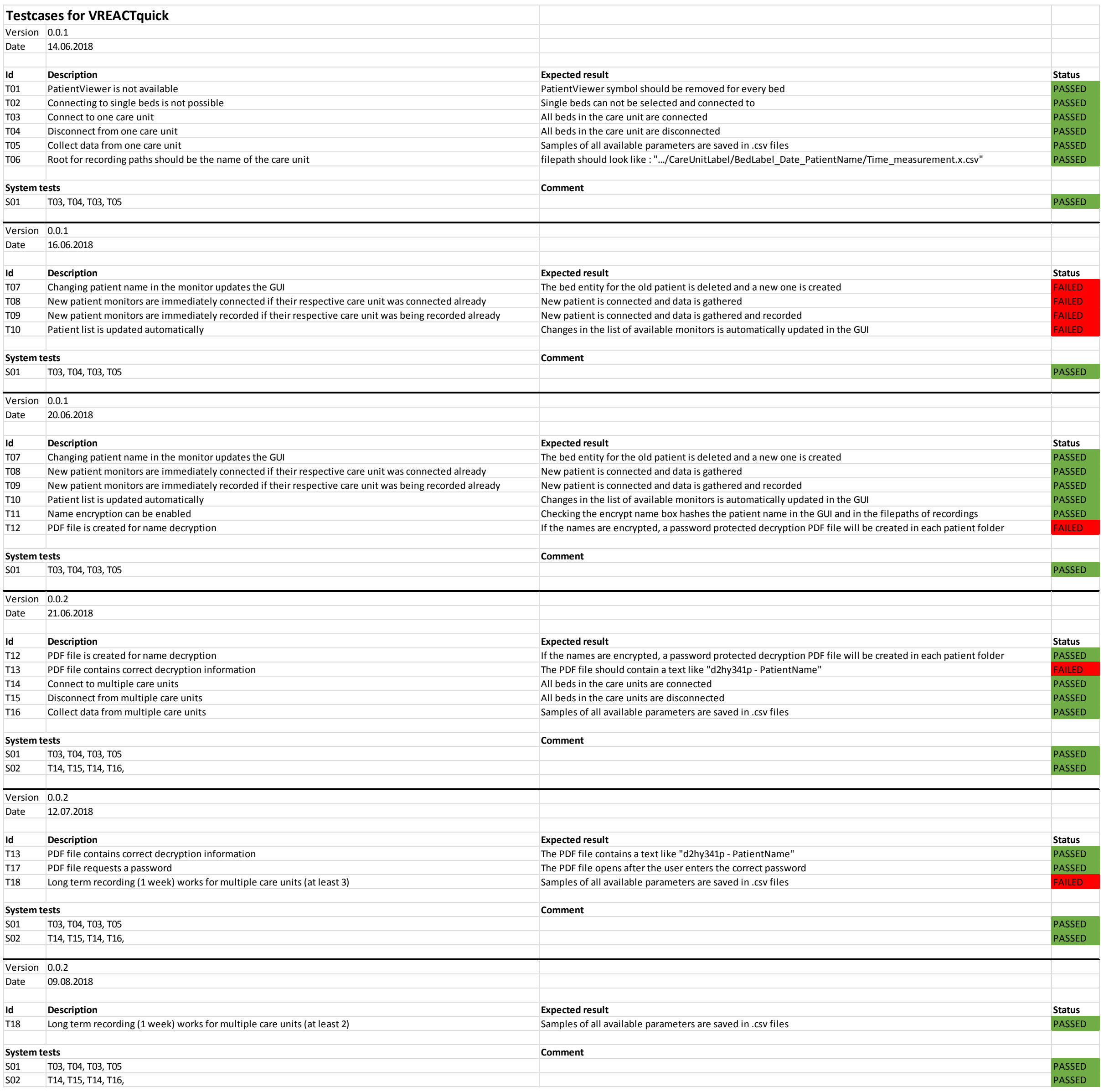
As of right now, VREACT is ready for usage in the perioperative environment. Our test protocols fulfil all the criteria that were set in the beginning of the project. Our group prepared a solid and extensible framework that future students and interested scientist can use and enhance. Therefore we hope that VREACT will continuously be extended by new modules. This is also important for making our tool more versatile and a valuable acquisition tool for future studies. Furthermore, the functionality of VREACTquick suits big data projects since it handles changes in the patient list automatically and can record data over long periods of time. The next step will be a handover of our tools to health care professionals in order to receive feedback from end users.

Appendix



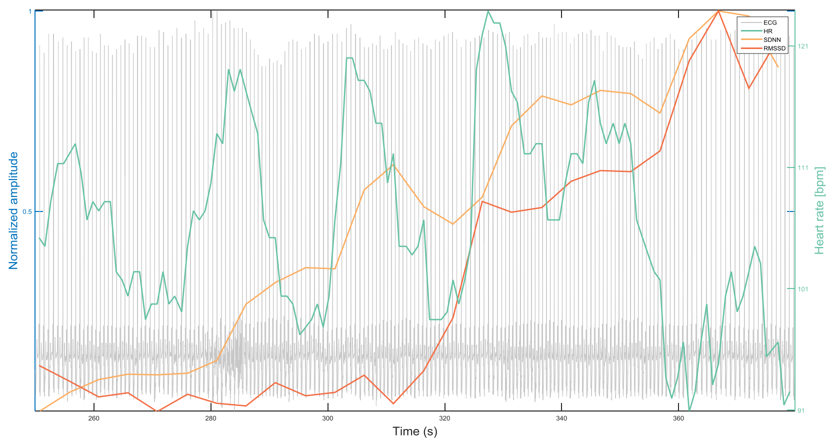
Testcases for VREACT			
Version	0.0.1		
Date	27.04.2018		
Id	**Description**	**Expected result**	**Status**
T01	Connect to one bed	Bed is connected and the PatientViewer symbol is enabled	PASSED
T02	Visualize all parameters of a bed in the PatientViewer	PatientViewer opens, list of available biosignals is displayed, drag & drop into slots works, visualization works	PASSED
T03	Close and reopen the PatientViewer	Viewer closes and opens again, all parameters can be visualized	PASSED
T04	Connect to two beds in the same care unit	Both beds are connected and PatientViewer symbols are enabled	PASSED
T05	Collect data from one beds	Samples of all available parameters are saved in the .csv file	PASSED
T06	Collect data from multiple beds	Samples of all available parameters are saved in the .csv file	FAILED
Version	0.0.1		
Date	10.05.2018		
Id	**Description**	**Expected result**	**Status**
T06	Collect data from multiple beds	Samples of all available parameters are saved in the .csv file	PASSED
T07	Connect to two beds in different care units	Both beds are connected and PatientViewer symbols are enabled	PASSED
T08	Connect to all available beds (should be more than 15)	All beds are connected and PatientViewer symbols are enabled	PASSED
T09	.csv files of a bed don't exceed 100mb	New .csv file should be started as soon as file reaches 100mb in size	PASSED
T10	.csv files of all beds don't exceed 100mb	New .csv file should be started as soon as file reaches 100mb in size	PASSED
T11	Disconnecting a bed automatically closes its PatientViewer	PatientViewer should close automatically as soon as the respective bed is disconnected	FAILED
Version	0.0.2		
Date	17.05.2018		
Id	**Description**	**Expected result**	**Status**
T11	Disconnecting a bed automatically closes its PatientViewer	PatientViewer should close automatically as soon as the respective bed is disconnected	PASSED
T12	Disconnecting multiple beds automatically closes their PatientViewers	PatientViewers should close automatically as soon as the respective beds are disconnected	PASSED
T13	Each bed can only have one active PatientViewer	If the PatientViewer of a bed is already active, reclicking the PatientViewer symbol should bring it to the front	FAILED
T14	Licence HTML is displayed correctly	Clicking on the "Info" button should open a window with licence information in it	PASSED
System tests		**Comment**	
S01	T01, T02, T03, T05		PASSED
S02	T04, T02, T03, T05, T06		PASSED
S03	T07, T02, T03, T05, T06		PASSED
S04	T08, T02, T03, T05, T06	Performance issues when connected to many beds and trying to use PatientViewer as well	PASSED
Version	0.0.2		
Date	24.05.2018		
Id	**Description**	**Expected result**	**Status**
T11	Disconnecting a bed automatically closes its PatientViewer	PatientViewer should close automatically as soon as the respective bed is disconnected	PASSED
T12	Disconnecting multiple beds automatically closes their PatientViewers	PatientViewers should close automatically as soon as the respective beds are disconnected	PASSED
T13	Each bed can only have one active PatientViewer	If the PatientViewer of a bed is already active, reclicking the PatientViewer symbol should bring it to the front	PASSED
T15	Importing .csv file into MATLAB works	Data is imported into MATLAB and can be plotted correctly	PASSED
T16	MATLAB processes seperators correctly	Data is seperated correctly and the amount of created columns matches the available signals	FAILED
System tests		**Comment**	
S01	T01, T02, T03, T05		PASSED
S02	T04, T02, T03, T05, T06		PASSED
S03	T07, T02, T03, T05, T06		PASSED
S04	T08, T02, T03, T05, T06	Performance issues when connected to many beds and trying to use PatientViewer as well	PASSED
S05	T08, T02, T13, T12		PASSED
Version	0.0.3		
Date	21.06.2018		
Id	**Description**	**Expected result**	**Status**
T17	Disconnected sensors are removed from the model	If a sensor is removed the respective biosignal and its derivatives are deleted	PASSED
T18	Active plots are marked as disconnected if the respective sensor is removed	Signals and its derivates are labeled as disconnected if the respective sensor is removed	PASSED
System tests		**Comment**	
S01	T01, T02, T03, T05		PASSED
S02	T04, T02, T03, T05, T06		PASSED
S03	T07, T02, T03, T05, T06		PASSED
S04	T08, T02, T03, T05, T06	Performance issues when connected to many beds and trying to use PatientViewer as well	PASSED
S05	T08, T02, T13, T12		PASSED
Version	0.0.4		
Date	12.07.2018		
Id	**Description**	**Expected result**	**Status**
T19	Changing patient name in the monitor updates the GUI	The bed entity for the old patient is deleted and a new one is created	PASSED
T20	Patient list is updated automatically	Changes in the list of available monitors is automatically updated in the GUI	PASSED
System tests		**Comment**	
S01	T01, T02, T03, T05		PASSED
S02	T04, T02, T03, T05, T06		PASSED
S03	T07, T02, T03, T05, T06		PASSED
S04	T08, T02, T03, T05, T06	Performance issues when connected to many beds and trying to use PatientViewer as well	PASSED
S05	T08, T02, T13, T12		PASSED

Test protocol of VREACT. The content and expected result of each test case is listed. On the right side the status of a test is documented

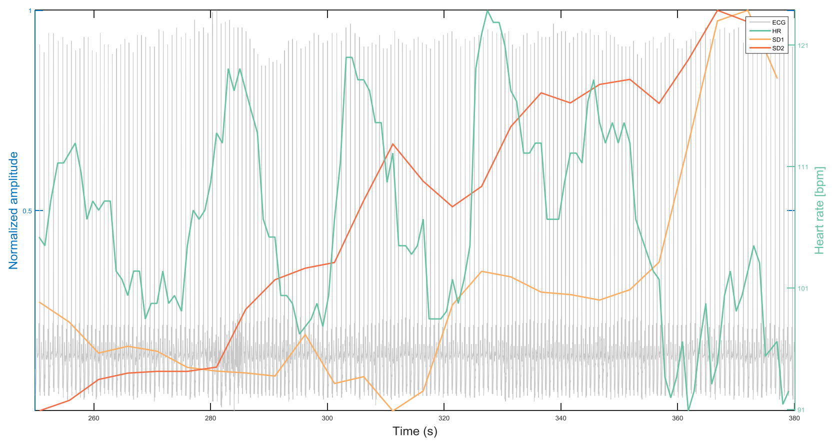


Testcases for VREACTquick			
Version	0.0.1		
Date	14.06.2018		
Id	**Description**	**Expected result**	**Status**
T01	PatientViewer is not available	PatientViewer symbol should be removed for every bed	PASSED
T02	Connecting to single beds is not possible	Single beds can not be selected and connected to	PASSED
T03	Connect to one care unit	All beds in the care unit are connected	PASSED
T04	Disconnect from one care unit	All beds in the care unit are disconnected	PASSED
T05	Collect data from one care unit	Samples of all available parameters are saved in .csv files	PASSED
T06	Root for recording paths should be the name of the care unit	filepath should look like : ".../CareUnitLabel/BedLabel_Date_PatientName/Time_measurement.x.csv"	PASSED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
Version	0.0.1		
Date	16.06.2018		
Id	**Description**	**Expected result**	**Status**
T07	Changing patient name in the monitor updates the GUI	The bed entity for the old patient is deleted and a new one is created	FAILED
T08	New patient monitors are immediately connected if their respective care unit was connected already	New patient is connected and data is gathered	FAILED
T09	New patient monitors are immediately recorded if their respective care unit was being recorded already	New patient is connected and data is gathered and recorded	FAILED
T10	Patient list is updated automatically	Changes in the list of available monitors is automatically updated in the GUI	FAILED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
Version	0.0.1		
Date	20.06.2018		
Id	**Description**	**Expected result**	**Status**
T07	Changing patient name in the monitor updates the GUI	The bed entity for the old patient is deleted and a new one is created	PASSED
T08	New patient monitors are immediately connected if their respective care unit was connected already	New patient is connected and data is gathered	PASSED
T09	New patient monitors are immediately recorded if their respective care unit was being recorded already	New patient is connected and data is gathered and recorded	PASSED
T10	Patient list is updated automatically	Changes in the list of available monitors is automatically updated in the GUI	PASSED
T11	Name encryption can be enabled	Checking the encrypt name box hashes the patient name in the GUI and in the filepaths of recordings	PASSED
T12	PDF file is created for name decryption	If the names are encrypted, a password protected decryption PDF file will be created in each patient folder	FAILED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
Version	0.0.2		
Date	21.06.2018		
Id	**Description**	**Expected result**	**Status**
T12	PDF file is created for name decryption	If the names are encrypted, a password protected decryption PDF file will be created in each patient folder	PASSED
T13	PDF file contains correct decryption information	The PDF file should contain a text like "d2hy341p - PatientName"	FAILED
T14	Connect to multiple care units	All beds in the care units are connected	PASSED
T15	Disconnect from multiple care units	All beds in the care units are disconnected	PASSED
T16	Collect data from multiple care units	Samples of all available parameters are saved in .csv files	PASSED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
S02	T14, T15, T14, T16,		PASSED
Version	0.0.2		
Date	12.07.2018		
Id	**Description**	**Expected result**	**Status**
T13	PDF file contains correct decryption information	The PDF file contains a text like "d2hy341p - PatientName"	PASSED
T17	PDF file requests a password	The PDF file opens after the user enters the correct password	PASSED
T18	Long term recording (1 week) works for multiple care units (at least 3)	Samples of all available parameters are saved in .csv files	FAILED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
S02	T14, T15, T14, T16,		PASSED
Version	0.0.2		
Date	09.08.2018		
Id	**Description**	**Expected result**	**Status**
T18	Long term recording (1 week) works for multiple care units (at least 2)	Samples of all available parameters are saved in .csv files	PASSED
System tests		**Comment**	
S01	T03, T04, T03, T05		PASSED
S02	T14, T15, T14, T16,		PASSED

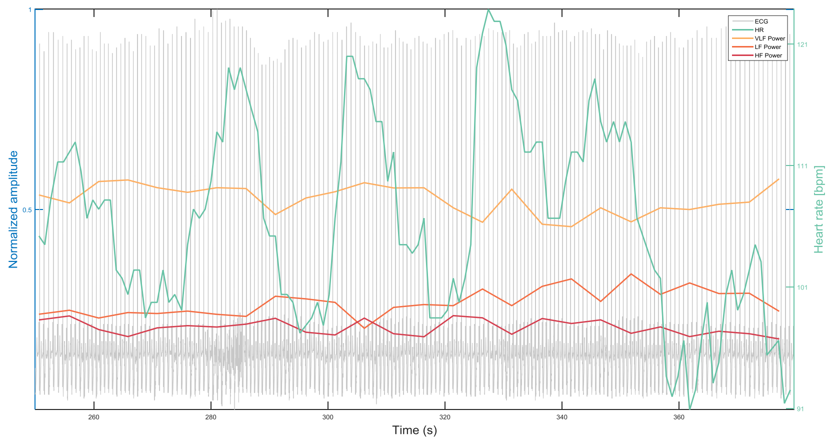
Test protocol of VREACTquick. The content and expected result of each test case is listed. On the right side the status of a test is documented



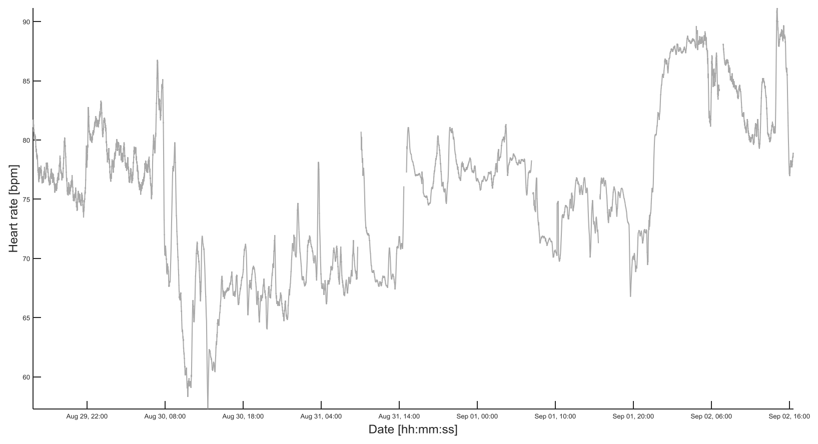
Visualization of data calculated and recorded by VREACTquick. This plot represents the relation between the ECG, HR, SDNN and RMSSD signals. On the right y-axis the HR in bpm and on the left y-axis the normalized amplitudes of the remaining signals can be seen. The x – axis shows the time stamps in seconds.



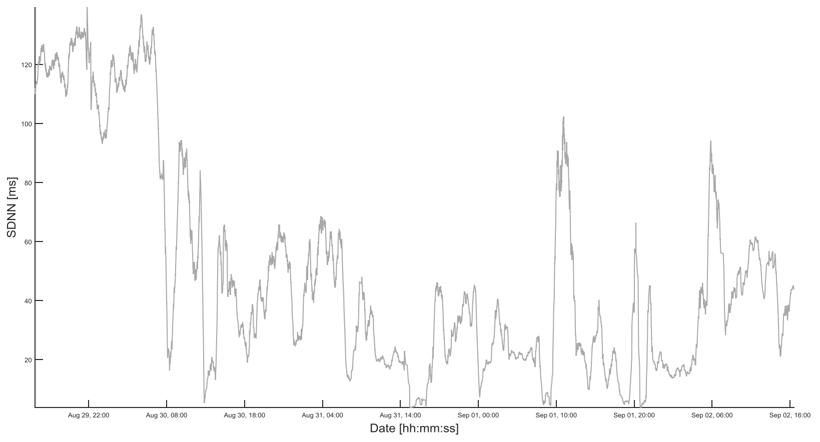
Visualization of data calculated and recorded by VREACTquick. This plot represents the relation between the ECG, HR, SD1 and SD2 signals. On the right y-axis the HR in bpm and on the left y-axis the normalized amplitudes of the remaining signals can be seen. The x – axis shows the time stamps in seconds.



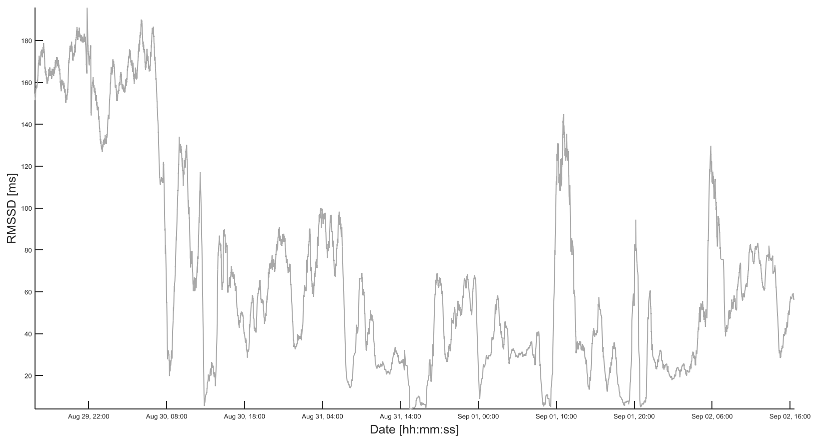
Visualization of data calculated and recorded by VREACTquick. This plot represents the relation between the ECG, HR, VLF Power, LF Power and HF Power signals. On the right y-axis the HR in bpm and on the left y-axis the normalized amplitudes of the remaining signals can be seen. The x – axis shows the time stamps in seconds.



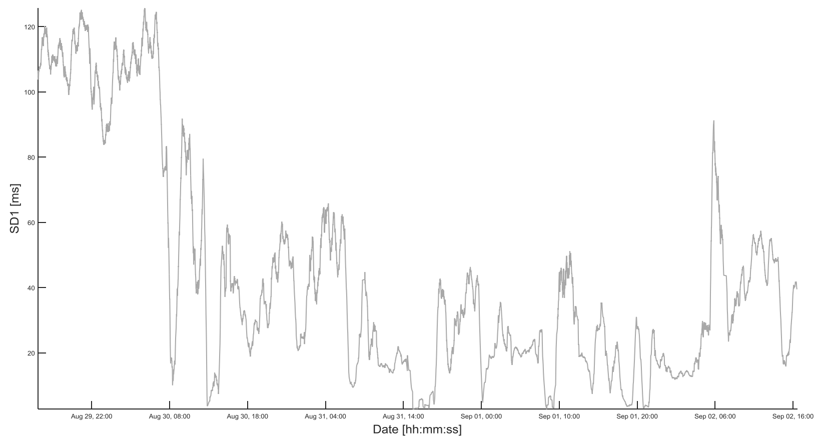
Visualization of a long-term HR signal calculated and recorded by VREACTquick. The signal was averaged by a 10 minute window. The y – axis depicts the HR in bpm and the x – axis shows the time in date format. The signal is discontinuous due to sporadic detachment of the ECG sensors.



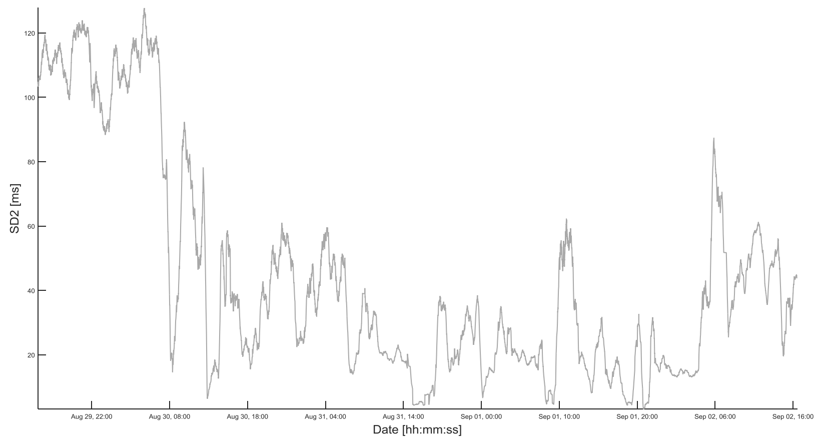
Visualization of a long-term SDNN signal calculated and recorded by VREACTquick. The signal was averaged by a 30 minute window. The y – axis depicts amplitude in ms and the x – axis shows the time in date format.



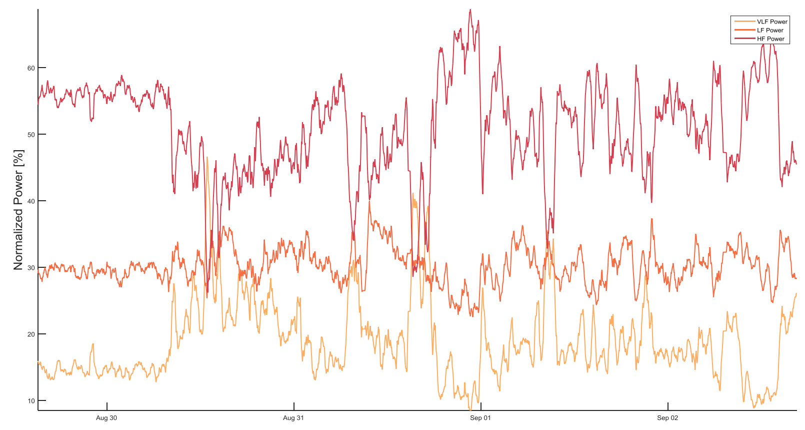
Visualization of a long-term RMSSD signal calculated and recorded by VREACTquick. The signal was averaged by a 30 minute window. The y – axis depicts the amplitude in ms and the x – axis shows the time in date format.



Visualization of a long-term SD1 signal calculated and recorded by VREACTquick. The signal was averaged by a 30 minute window. The y – axis depicts the amplitude in ms and the x – axis shows the time in date format.



Visualization of a long-term SD2 signal calculated and recorded by VREACTquick. The signal was averaged by a 30 minute window. The y – axis depicts the amplitude in ms and the x – axis shows the time in date format.



Visualization of long-term spectral power measurements for VLF, LF and HF power calculated and recorded by VREACTquick. The signals were averaged by a 30 minute window. The y – axis depicts the normalized amplitude and the x – axis shows the time in date format.

References

[1] K. Bartels, E. T. Karhausen, A. Clambey, A. Grenz, and H. K. Eltzschig, "Perioperative Organ Injury," *Anesthesiology*, vol. 119, no. 6, pp. 1474–1489, 2013.

[2] S. M. Lobo *et al.*, "Early determinants of death due to multiple organ failure after noncardiac surgery in high-risk patients," *Anesth. Analg.*, vol. 112, no. 4, pp. 877–883, 2011.

[3] V. D. Mayr *et al.*, "Causes of death and determinants of outcome in critically ill patients," *Crit. Care*, vol. 10, no. 6, pp. 1–13, 2006.

[4] G. C. Green, B. Bradley, A. Bravi, and A. J. E. Seely, "Continuous multiorgan variability analysis to track severity of organ failure in critically ill patients," *J. Crit. Care*, vol. 28, no. 5, p. 879.e1-879.e11, 2013.

[5] J. Pontet *et al.*, "Heart Rate Variability as Early Marker of Multiple Organ Dysfunction Syndrome in Septic Patients," *J. Crit. Care*, vol. 18, no. 3, pp. 156–163, 2003.

[6] Dräger, "Monitoring and It solutions for supporting patient safety and care," 2015. [Online]. Available: https://www.draeger.com/Products/Content/infinity_bedside_solutions_br_9051805_en.pdf. [Accessed: 18-Mar-2018].

[7] E. Kaniusas, *Biomedical signals and sensors I: Linking physiological phenomena and biosignals*. Springer Heidelberg Dordrecht London New York, 2012.

[8] A. J. Weinhaus and R. Kenneth P., "Anatomy of the Human Heart," *Handb. Card. Anatomy, Physiol. Devices Third Ed.*, pp. 51–79, 2005.

[9] F. M. Filipoiu, *Atlas of Heart Anatomy and Development*. 2014.

[10] V. A. Barnett, "Cardiac Myocytes," *Handb. Card. Anatomy, Physiol. Devices, Third Ed.*, pp. 113–136, 2005.

[11] P. I. Aaronson, J. P. T. Ward, and M. J. Connolly, *The Cardiovascular System at a Glance*, Fourth Edi. John Wiley & Sons, Ltd, 2013.

[12] Wikimedia Foundation, "Schematic diagram of normal sinus rhythm for a human heart as seen on ECG." [Online]. Available: https://en.wikipedia.org/wiki/Electrocardiography#/media/File:SinusRhythmLabels.svg. [Accessed: 09-Mar-2018].

[13] S. A. Jones, *ECG Success: Exercises in ECG Interpretation*. F. A. Davis Company, 2008.

[14] A. Bayés de Luna, *Basic Electrocardiography: Normal and Abnormal ECG Patterns*. 2008.

[15] T. F. of the E. S. of C. and the N. A. S. of P. and Electrophysiology, "An introduction to heart rate variability : methodological considerations and clinical applications," *Front. Physiol.*, vol. 6, no. February, pp. 2013–2015, 2015.

[16] N. R. Bush, Z. K. Caron, K. S. Blackburn, and A. Alkon, "Measuring Cardiac Autonomic Nervous System (ANS) Activity in Toddlers - Resting and Developmental Challenges," *J. Vis. Exp.*, no. 108, pp. 1–12, 2016.

[17] G. E. Billman, "Heart rate variability - A historical perspective," *Front. Physiol.*, vol. 2 NOV, no. November, pp. 1–13, 2011.

[18] F. Lombardi and P. K. Stein, "Origin of heart rate variability and turbulence: An appraisal of autonomic modulation of cardiovascular function," *Front. Physiol.*, vol. 2 DEC, no. December, pp. 1–7, 2011.

[19] T. F. of the E. S. of C. the N. A. S. of P. Electrophysiology, "Heart Rate Variability: Standards of Measurement, Physiological Interpretation, and Clinical Use," *Eur. Heart*

J., vol. 17, pp. 354–381, 1996.

[20] M. Brennan, M. Palaniswami, and P. Kamen, “Poincaré plot interpretation using a physiological model of HRV based on a network of oscillators,” *Am. J. Physiol. - Hear. Circ. Physiol.*, vol. 283, no. 5, p. H1873 LP-H1886, Nov. 2002.

[21] F. R. Gilliam, J. P. Singh, C. M. Mullin, M. McGuire, and K. J. Chase, “Prognostic value of heart rate variability footprint and standard deviation of average 5-minute intrinsic R-R intervals for mortality in cardiac resynchronization therapy patients,” *J. Electrocardiol.*, vol. 40, no. 4, pp. 336–342, 2007.

[22] C. M. DeGiorgio *et al.*, “RMSSD, a Measure of Heart Rate Variability, Is Associated With Risk Factors For SUDEP: The SUDEP-7 Inventory,” *Epilepsy Behav.*, vol. 19, no. 1, pp. 78–81, 2011.

[23] M. A. Woo, W. G. Stevenson, D. K. Moser, R. B. Trelease, and R. M. Harper, “Patterns of beat-to-beat advanced heart failure heart rate variability in,” *Am. Hear. J.*, pp. 704–710, 1992.

[24] F. Shaffer, R. McCraty, and C. L. Zerr, “A healthy heart is not a metronome: an integrative review of the heart’s anatomy and heart rate variability,” *Front. Psychol.*, vol. 5, no. September, pp. 1–19, 2014.

[25] A. Jovic and N. Bogunovic, “Electrocardiogram analysis using a combination of statistical, geometric, and nonlinear heart rate variability features,” *Artif. Intell. Med.*, vol. 51, no. 3, pp. 175–186, 2011.

[26] F. Shaffer and J. P. Ginsberg, “An Overview of Heart Rate Variability Metrics and Norms,” *Front. Public Heal.*, vol. 5, no. September, pp. 1–17, 2017.

[27] H. Tsuji *et al.*, “Reduced Heart Rate Variability and Mortalit Risk in an Elderly Cohort The Framingham Heart Study,” pp. 878–883, 1994.

[28] M. Hadase *et al.*, “Very low frequency power of heart rate variability is a powerful predictor of clinical prognosis in patients with congestive heart failure.,” *Circ. J.*, vol. 68, no. 4, pp. 343–347, 2004.

[29] R. E. Kleiger, P. K. Stein, and J. T. Bigger, “Heart rate variability: Measurement and clinical utility,” *Ann. Noninvasive Electrocardiol.*, vol. 10, no. 1, pp. 88–101, 2005.

[30] R. J. Winchell and D. B. Hoyt, “Spectral analysis of heart rate variability in the ICU: a measure of autonomic function,” *J Surg Res*, vol. 63, no. 1, pp. 11–16, 1996.

[31] J. T. Bigger, J. L. Fleiss, L. M. Rolnitzky, and R. C. Steinman, “Frequency domain measures of heart period variability to assess risk late after myocardial infarction,” *J. Am. Coll. Cardiol.*, vol. 21, no. 3, pp. 729–736, 1993.

[32] J. T. Bigger, J. L. Fleiss, L. M. Rolnitzky, and R. C. Steinman, “The ability of several short-term measures of RR variability to predict mortality after myocardial infarction,” *Circulation*, vol. 88, no. 3, pp. 927–934, 1993.

[33] R. E. Kleiger, J. P. Miller, J. T. Bigger, and A. J. Moss, “Decreased Heart Rate Variability and Its Association with Increased Mortality After Acute Myocardial Infarction,” *Am J Cardiol*, vol. 59, pp. 258–282, 1987.

[34] M. T. La Rovere, J. T. Bigger, F. I. Marcus, A. Mortara, and P. J. Schwartz, “Baroreflex sensitivity and heart-rate variability in prediction of total cardiac mortality after myocardial infarction. ATRAMI (Autonomic Tone and Reflexes After Myocardial Infarction) Investigators.,” *Lancet (London, England)*, vol. 351, no. 9101, pp. 478–84, 1998.

[35] A. Erdogan *et al.*, “Prognostic value of heart rate variability after acute myocardial infarction in the era of immediate reperfusion,” *Herzschrittmachertherapie und Elektrophysiologie*, vol. 19, no. 4, pp. 161–168, 2008.

[36] J. F. Thayer, F. Åhs, M. Fredrikson, J. J. Sollers, and T. D. Wager, “A meta-analysis of heart rate variability and neuroimaging studies: Implications for heart rate variability

as a marker of stress and health,” *Neurosci. Biobehav. Rev.*, vol. 36, no. 2, pp. 747–756, 2012.

[37] H. C. Lee and C. W. Jung, “Vital Recorder- A free research tool for automatic recording of high-resolution time-synchronised physiological data from multiple anaesthesia devices,” *Sci. Rep.*, vol. 8, no. 1, pp. 1–8, 2018.

[38] N. Stylianides, M. D. Dikaiakos, H. Gjermundrod, G. Panayi, and T. Kyprianou, “Intensive care window: real-time monitoring and analysis in the intensive care environment.,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 1, pp. 26–32, 2011.

[39] Microsoft, “Microsoft Visual Studio.” [Online]. Available: https://www.visualstudio.com/de/vs/. [Accessed: 05-Jun-2018].

[40] Microsoft, “Microsoft Foundation Class Library.” [Online]. Available: https://docs.microsoft.com/en-us/cpp/mfc/mfc-desktop-applications. [Accessed: 05-Jun-2018].

[41] Qt, “Signals & Slots.” [Online]. Available: http://doc.qt.io/archives/qt-4.8/signalsandslots.html. [Accessed: 05-Jun-2018].

[42] Qt, “QObject.” [Online]. Available: https://doc.qt.io/qt-5/qobject.html. [Accessed: 10-Jun-2018].

[43] Qt, “The Meta-Object System.” [Online]. Available: https://doc.qt.io/archives/qt-4.8/metaobjects.html. [Accessed: 10-Jun-2018].

[44] Qt, “Threads and QObjects.” [Online]. Available: https://doc.qt.io/archives/qt-4.8/threads-qobject.html. [Accessed: 10-Jun-2018].

[45] F. Thürk *et al.*, “Real-time assessment of perioperative clinical parameters based on high resolution vital signs recordings,” *Submitt. to Med. Meas. Appl.*, 2018.

[46] Qt, “Model/View Programming.” [Online]. Available: http://doc.qt.io/archives/qt-4.8/model-view-programming.html. [Accessed: 17-Jun-2018].

[47] Wikimedia Foundation, “The model, view, and controller (MVC) pattern relative to the user.” [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/200px-MVC-Process.svg.png. [Accessed: 08-Jul-2018].

[48] M. Bachler, C. Mayer, B. Hametner, S. Wassertheurer, and A. Holzinger, “Online and Offline Determination of QT and PR Interval and QRS Duration in Electrocardiography,” *LNCS 7719 - Pervasive Comput. Networked World*, no. November, pp. 1–15, 2012.

[49] D. Singh, K. Vinod, and S. Saxena, “Sampling frequency of the RR interval time series for spectral analysis of heart rate variability,” *J. Med. Eng. Technol.*, vol. 28, no. 6, pp. 263–272, 2004.

[50] H. Stallmann, “QRealFourier.” [Online]. Available: https://github.com/visore/QRealFourier. [Accessed: 02-Jul-2018].

[51] QCustomPlot, “QCustomPlot.” [Online]. Available: https://www.qcustomplot.com/. [Accessed: 15-Aug-2018].

[52] LibHaru, “libHaru.” [Online]. Available: http://libharu.org/. [Accessed: 15-Aug-2018].

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, 14.09.2018

....
Fatih Kartal