NTNU – Trondheim
Norwegian University of
Science and Technology

A Scientific Tool for Real-time Acquisition and Analysis of Cardiovascular Measurements

Including ultrasound and other physiological
data sources

Morten Smedsrud Wigen

Electronics System Design and Innovation
Submission date: December 2013
Supervisor: Ilangko Balasingham, IET
Co-supervisor: Lasse Løvstakken, Institutt for sirkulasjon og bildediagnostikk

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Oppgavebeskrivelse

En programvare for studier av karsystemet, for både eksperimentelle og kliniske forsøk, skal utvikles i denne oppgaven. Data fra både ultralyd og andre fysiologiske kilder, som trykk, blodstrøm og volum, skal kunne mottas i sanntid. Datakildene må synkroniseres for at målingene skal kunne kombineres og sammenlignes.

B-mode og Color Flow skal være støttet av det utviklede programmet, slik at data(hovedsakelig hastighet) kan ekstraheres fra områder i bilde.

Brukergrensesnittet skal være intuitivt for akkvisisjon og analyse, og sørge for muligheter for funksjonell lagring av data i et format som kan importeres til annen programvare, som Matlab, for ytterligere analyse.

Problem description

In this thesis a software framework for cardiovascular research shall be developed for use in both experimental and clinical studies. The software should allow for simultaneously acquisition of digitally streamed ultrasound images and analog / digital data from physiological measurements such as pressure, flow, and volume. It should further allow for synchronization, combination, and comparison of these non-invasive and invasive data. Both B-mode and color flow ultrasound modalities shall be supported and it should further be possible to extract data from specific regions in these images (e.g. velocity traces). The software should provide an intuitive user interface for data acquisition and initial analysis, and provide storage functionality of data in an open format for further analysis in standard software (e.g Matlab).

Preface

This diploma has been written at the Department of Electronics and Telecommunications at NTNU, Trondheim, and carried out under the Department of Circulation and Medical Imaging at St. Olavs Hospital. While the main ideas for the software has been the same since development started during spring 2013, it has been totally rewritten for the diploma.

My supervisor during the master thesis has been Lasse Løvstakken, who has contributed with great ideas and helpful guidance throughout the semester. I also want to thank Gabriel Kiss for sharing his knowledge about computer programming, Idar Kirkeby Garstad and the clinical team at his department for including me in their trials and lastly Ilangko Balasingham for being my formal adviser.

Oslo, December 2013
Morten Smedsrud Wigen

Abstract

The heart is a complex organ that is yet to be fully understood, and much effort and resources are put into research in the field of cardiology. To acquire analyzable measurements of the heart's properties, different devices are used. Echocardiography uses ultrasound as an imaging modality specialized for this purpose. Other measurements are acquired invasively, with their respective devices. Even though these signals serve a purpose alone, they can give new meaning if processed jointly.

This thesis presents an analysis and acquisition tool for centralization of these measurements under one interface. Work has been put into development of a system that acquires data, at adequate rates, and is organized into an interface that allows intuitive user-interaction and practical project organization. The result of the developed application can be of importance for other scientists' future work, as it gives the user the opportunity to interpret real-time data in a currently non-existing manner.

Sampling rates up to $499.5 \pm 3.91Hz$. Hz was achieved from a general acquisition unit, and a frame rate of 40 fps for the ultrasound frames. The delay between the acquisition units was found to be 200 ± 15.5ms, were the mean was successfully adjusted for through a developed synchronization method.

Contents

1 Introduction **7**
- 1.1 Motivation . . . 7
- 1.2 State of research . . . 8
- 1.3 Goals . . . 8
- 1.4 Thesis outline . . . 8

2 Theory **11**
- 2.1 Physiology . . . 11
- 2.2 Medical instrumentation . . . 16
  - 2.2.1 Ultrasound . . . 16
  - 2.2.2 Coronary catheterization . . . 22

3 Methodology **25**
- 3.1 Data acquisition . . . 25
  - 3.1.1 Sampling . . . 25
  - 3.1.2 Filtering . . . 26
  - 3.1.3 Calibration . . . 27
- 3.2 Ultrasound streaming . . . 28
- 3.3 Traces from ultrasound data . . . 28
- 3.4 Synchronization . . . 30
- 3.5 Implementation . . . 30
  - 3.5.1 Language and libraries . . . 30
  - 3.5.2 Multiple threading . . . 31
  - 3.5.3 Timing . . . 32
  - 3.5.4 File format . . . 33
  - 3.5.5 Design and GUI . . . 33
- 3.6 Experimental set-ups . . . 33
  - 3.6.1 Pig trials . . . 34
  - 3.6.2 Human trial . . . 36

4 Results **37**
- 4.1 GUI implementation . . . 37
  - 4.1.1 Start-up menu . . . 37
  - 4.1.2 Main application window . . . 40
- 4.2 Data acquisition . . . 48
  - 4.2.1 Acquisition from the DAQ . . . 48
  - 4.2.2 Streaming . . . 48
- 4.3 Synchronization . . . 49
- 4.4 In vivo results . . . 52
- 4.5 Saving of data . . . 62
- 4.6 Importing to Matlab . . . 64

5 Discussion **67**
5.1 GUI and usability 67
5.2 Data acquisition 67
5.3 Ultrasound streaming 68
5.4 Synchronization 69
5.5 In vivo results 69
5.6 Project files 70

6 Conclusions **71**

7 Appendix **75**

A GUI elements **75**

B Flow derivation **77**

C C++ code **78**
C.1 Data retrieval from data buffer 78
C.2 Insert data to image ring buffer 80
C.3 Data retrieval from image buffer 80
C.4 Control of sampling rate 82

D Matlab code **83**
D.1 Load session 83
D.2 Load recording 83
D.3 Load measurement 84

E Connections of instruments **87**

List of Figures

1 The cardiocascular system . . . 11
2 Cross section of the heart . . . 12
3 The heart cycle . . . 13
4 Pressure and volume plots . . . 15
5 Transesophageal probe . . . 18
6 M-mode imaging . . . 19
7 Color flow image . . . 21
8 TVI . . . 22
9 Catheter placement . . . 23
10 FIR filter . . . 26
11 FIR filter - impulse respons . . . 27
12 FIR filter - frequency respons . . . 27
13 Angle correction . . . 29
14 Threads . . . 32
15 Pig trial . . . 35
16 Experimental set-up for pig trials . . . 35
17 Instruments in operation room . . . 36
18 Experimental set-up for human trial . . . 36
19 New project tab . . . 38
20 Load project tab . . . 39
21 The *Mixed view* tab . . . 41
22 The *Ultrasound* tab . . . 43
23 The *NIDAQ* tab . . . 45
24 DAQ toolbox . . . 46
25 The synchronization view. show the synchronization signals before and after the delay is found. The plots does however not look completely aligned, this is due to difference in the render time between the plots, as discussed later in the thesis. . . . 47
26 Synchronization validation . . . 51
27 After synch delay . . . 52
28 Volume filtering . . . 53
29 Pressure data . . . 54
30 PV loops in the application . . . 55
31 PV-loop plot . . . 56
32 ECG comparison . . . 57
33 Flow rate trace . . . 58
34 Flow rate plot . . . 59
35 Strain rate screen shot . . . 60
36 Strain rate plot . . . 61
37 Project file organization . . . 63
38 Data viewer in Matlab 1 . . . 64
39 Data viewer in Matlab 2 . . . 65
40 Two screens 1; the NIDAQ tab is dragged an external display. . . . 75

41 Two screens 2; mixed view, with only the ultraound image represented, dragged to an external display. 76
42 Two screens 3; another combination of mixed view on laptop with ultrasound three DAQ plots and PV-plot, and the DAQ-tab on the external display. 76
43 Sentron pressure interface. A T-coupling is used as the pressure signal was used both by the DAQ and in the Leycom Sigma 5DF. 87
44 Leycom Sigma 5DF. Takes pressure as input to show PV-plots on another screen. Volume is generated by the unit and is available through an output channel. ECG is also available. 87
45 NI USB-6251; Connections to the DAQ. Channel 0: Volume, channel 1: Flow rate(not used), channel 2: ECG, channel 3: Pressure from Millar catheter(not used), channel 4: Pressure from Leycom 88

List of Tables

1 DAQ samplings rate 48
2 Ultrasound frame rate 49
3 Streamed ultrasound ECG 49

Nomenclature

ADC	Analogue to digital converter
API	Application programmable interface
AUX	Auxilary
AV	Atrioventricular
bpm	Beats per minute
CF	Color flow
CF	Color flow
CW	Continuous wave
Ea	Arterial elastance
EDPVR	End diastolic pressure volume ratio
EDV	End diastolic volume
EF	Ejection fraction
ESPVR	End systolic pressure volume ratio
ESV	End systolic volume
FIR	Finit impulse repsonse
FPS	Frames per second
fps	Frames per second
FR	Frame rate
GUI	Graphical user interface
PRF	Pulse repetition frequency
PW	Pulsed wave
SA	Sinoatrial
SL	Semilunar
SR	Samplings rate
TVI	Tissue velocity imaging
TVI	Tissue velocity imaging
VTK	Visualization Tool Kit
VTK	Visualization Toolkit

1 Introduction

A scientific tool for heart studies is presented in this thesis. With ultrasound data streamed over IP and other physiological data collected from a general acquisition unit, an implementation has been made to centralize these in a PC environment. The application is written in C++ and tailor made to obtain an effective system with a flexible user experience for non-technical personnel.

Acquisition of quantitative and qualitative measurements is one of the strength of medical science, as physical conditions can be investigated based on statistical studies and previously obtained knowledge. A variety of acquisition devices are today used to collect data with different interpretation possibilities.

Ultrasound technology has had a great impact on studies of the cardiovascular system[14] as it provides real-time images of its anatomy and dynamics. Ultrasound gives the user the ability to examine anatomy in multiple dimensions, measure velocities of moving objects inside the body, and other derived analytical measures. As the technology uses pressure waves for acquisition, examinations are considered as harmless as well as cheap, which makes it a low threshold method.

Conductance catheters are another group of instruments that are much used for monitoring a subject's physical condition. In relevance for heart studies, ventricular pressure and volume, aortic flow and ECG are examples of useful measurements. These are usually represented as traces on a display. Catheter data are acquired invasively and the use will hence be restricted by higher requirements for certified personnel and working environment than ultrasound.

1.1 Motivation

If integrating ultrasound data with other physiological parameters, studies on the cardiovascular system can be performed in a new manner. A combination of these sources can be used for both studies on relations between variables, but also for validation of data acquired from ultrasound examination. If parameters derived from ultrasound data can be shown to replace invasive measurements, as the ones retrieved from catheters, studies that now are restricted to animals can be performed on human patients.

Three examples on current projects that could benefit from such a system is:

- Cardiac power by ultrasound; uses ultrasound for blood flow measurements in addition to invasively measured blood pressure, to measure the heart's energy deliverance. An animal model has been developed at St. Olavs, Trondheim, but a better integrated system is needed for patient studies.

- A heart catheterization project by post doc Brage Amundsen, NTNU, for studies on relationships between ultrasound measurements in systolic and diastolic function and invasive pressure from the heart chambers.

- Studies on blood flow regulation in the kidney arteries. A scientific project proposed by prof Sven Erik Ricksten, Gøteborg og prof Gerald diBona, Iowa, in anticipation of an integrated system for blood pressure and blood flow measurements.

1.2 State of research

A similar application as the one described here has not been found in any published papers. The main difference between this application and related software is the real-time capabilities that lets a user study the different data sources, including ultrasound, under one interface during examination.

The project has its origin from a program developed by Ole Omejer that acquired data from multiple sources with a multichannel acquisition unit. Ultrasound, however, was imported off-line [24]. The experience gained from Ole's project in combination with the development of a streaming client for ultrasound data, made this project possible.

1.3 Goals

To evolve the possibilities for cardiovascular studies, this application shall be capable of acquire signals from an ultrasound scanner as well as other data sources (e.g. conductance catheters) connected to an acquisition unit. These data must be synchronized and calibrated, if needed, with effective methods to provide functional acquisition and analysis. A flexible, dynamic and intuitive graphical user interface should make the functionality easy available for a user with no technical background.

Data has to be acquired at adequate rates to maintain data integrity, this goes to both sampling frequency as well as frame rates. Capabilities as signal filtering, parameter extraction from ultrasound data and pressure-volume plots should be functions that extend the usability of the system.

As a scientific tool the application should also provide functionality for project management, in a intuitive manner. A project file should be capable of holding recorded data in a way that makes post processing and analysis possible.

1.4 Thesis outline

The first section of this thesis is covering theory relevant for understanding of the cardiovascular system's physiology, and some how to retrieve relevant measurements through medical instrumentation.

The methodology section is related to processing of signals and data acquired in the application. This is followed up by the main aspects for a functional implementation. Lastly, the experimental set-ups used for the results are described.

The *results* section is divided into six sub sections. The first is a walk-through of the developed GUI for the application. The second describes the

results for data acquisition from the ultrasound scanner and the acquisition unit. Then a sub section lists the results from the synchronized functionality of the application. Sub section four contains results from the experimental set-ups and how the application worked in practice. The two last sections cover saving of data and exportability with Matlab integration.

Discussion of the obtained results are available in section 5, where some ideas for improvement is commented as well. The last main section contains conclusions based on the problem statement.

2 Theory

As the source of the measurements is the heart, its main functions are described under this section's physiology part. This is followed up with theory concerning medical instrumentation.

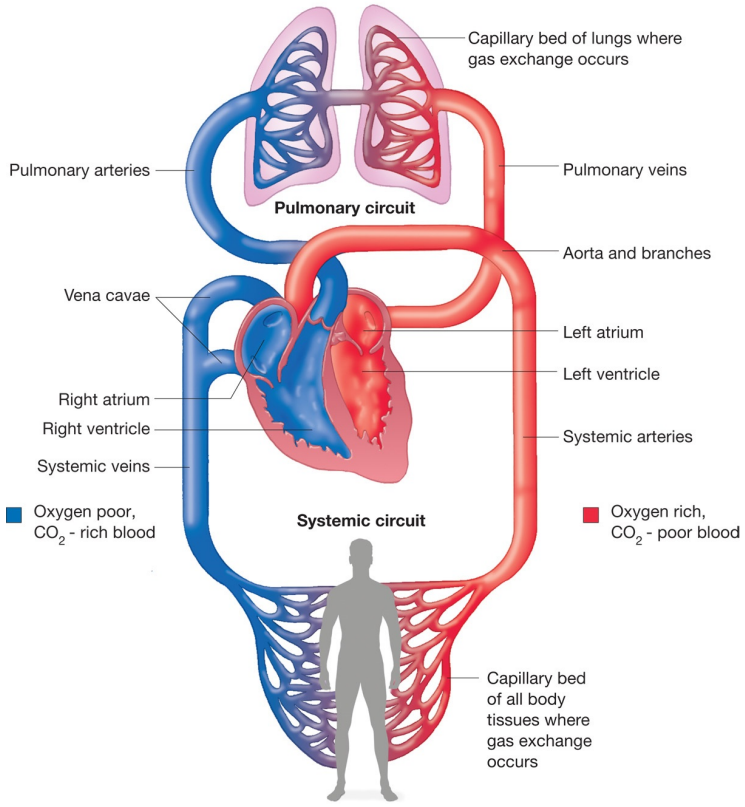


Figure 1: The cardiovascular system. The figure distinguishes the function of the four heart chambers. The left side distributes oxygen rich blood, while the right pumps deoxygenated blood to the lungs. [2]

2.1 Physiology

The cardiovascular system has one of the body's most crucial tasks; to distribute nutrients with the blood to and from the body's cells. It is comprised of the heart, the blood and the blood vessels. To make models and characteristics of these structures, their properties are parameterized. Together with the heart's function, some of the physical relations are found in this section.

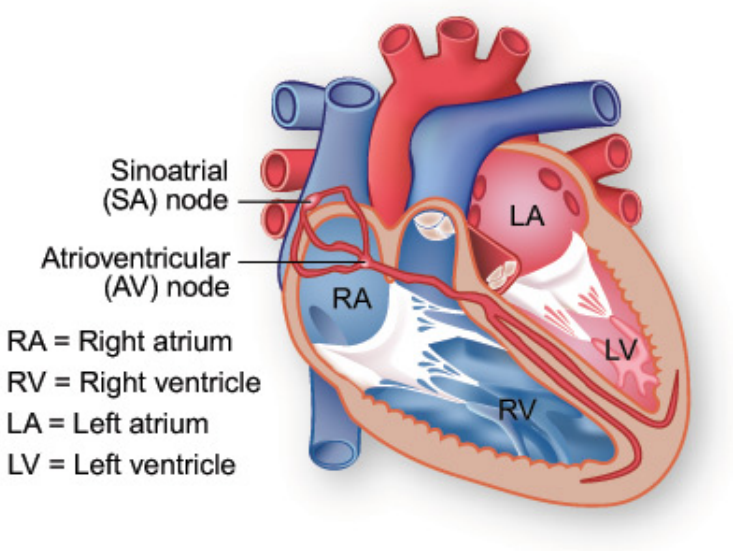


Figure 2: Cross section of the heart; showing all four chambers, and nodes. When the signal for heart contraction is triggered it travels from the SA node, to the AV node and further down the septum.[9]

The heart cycle The heart is build up by four chambers; two atria and to ventricles. The right atrium receives deoxygenated blood from the body, from where it flows into the right ventricle that in turn pumps it into the lungs in exchange for oxygen-rich blood. From the lungs the blood is transported to the left atrium and then into the left ventricle, where it is pumped into the aorta for transportation throughout the entire body.

In the heart there are also four valves. Two connects each of the atria to the respective ventricles, atrioventricular(AV) valves. The two others are called semilunar(SL) valves and are located at the output of the ventricles.

The pump function of the heart is possible due to the heart's structure, surrounding muscles, valves and its internal signaling system, triggered from the sinoatrial (SA) node. The hormone level and the autonomous nervous center control the frequency of the signals.

The signals to contract the heart muscles can be picked up and represented with electrocardiogram (ECG). A healthy heart will have a recognizable characteristic shape, and deviations from this can be helpful for medical diagnosis. The different parts of the ECG signal are named with the letters; P, Q, R, S and T, see figure 3, where the QRS-interval is the most significant contribution.

The interval where the ventricles are filled with blood is called diastole, while the phase where it is emptied is called systole. The heart cycle, from the beginning of diastole to the end of systole, can be described as follows:

a) During early diastole the AV valves are open, and SL valves closed, while the heart relaxes after the previous systole. Due to inflow of blood to the ventricles, their volume increases to $\sim 80\%$ of total volume.

b) At the end of the diastole, the SA node gives the initial signal from the top of the right atrium; this causes contraction of the muscles surrounding the right and left atrium and pumps blood into the ventricles. This fills the residual ventricle volume. After this stage the AV valves are closed.

c) Due to a delay in the AV node, the ventricles are filled with blood before the signal is passed through to the muscles that are responsible for the contraction of those chambers. When they do contract, the pressure in the ventricles will first increased, before the SL valves opens and pumps the blood into lungs and body.

d) At the end of this phase the SL valves closes, before the heart relaxes and then opening of the AV valves.[27]

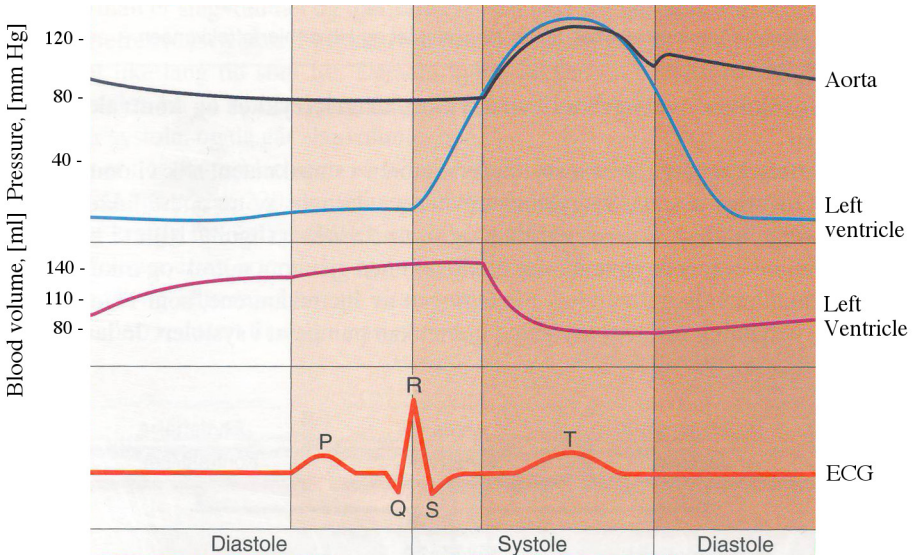


Figure 3: Graph of the hemodynamic properties pressure, volume and ECG, over a heart's cycle. The ECG's characteristic points are marked with the letters P, Q, R, S and T. Diastolic and systolic phases are indicated at the bottom.[27]

Heart function In the heart cycle there are some defined parameters that are useful to make quantitative measurements of the heart function. In figure 4a, end systolic volume(ESV) and end diastolic volume(EDV) are marked. These quantities are defined to be the instant volume of a ventricle at the end of the systolic and diastolic phase, which corresponds to the minimum and maximum

volume of the heart, respectively. From those values, the stroke volume of the heart can be calculated with equation 1.

$$SV = EDV - ESV \tag{1}$$

This corresponds to the amount of blood that is pumped out in a single heartbeat. The stroke volume has three main regulators; the end diastolic volume; total peripheral resistance, which is the bloods frictional force throughout the body; and last contractility.

A measure for the contractility of the heart is given by the ejection fraction (EF). This number tells how much of the full volume the heart is able to compress.

$$EF = \frac{SV}{EDV} \tag{2}$$

The stroke volume can be multiplied with heart rate to get the cardiac output (Q), described by equation 3.

$$Q = SV * HR \tag{3}$$

The cardiac output is the volume of blood pumped out from the heart every minute.

To measure the stroke work of the heart, the product between the cardiac output and the pressure is integrated over time.

$$PWR = \int Q * Pdt \tag{4}$$

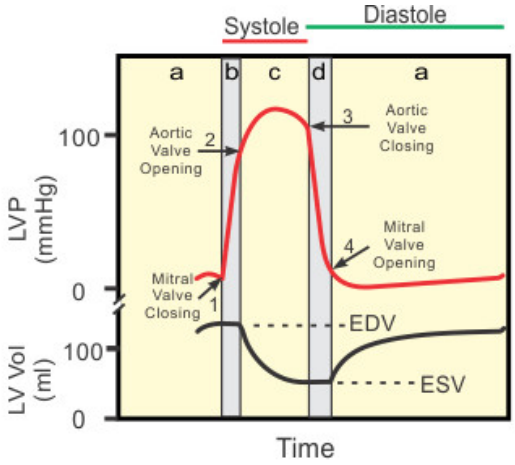
In a similar manner pressure-volume diagrams can also be used to analyze the work done by the heart. They are then plotted as seen in figure 4b, with pressure on the y-axis and volume on the x-axis. These measurements can be performed both for the right and the left ventricle.

The trace that is drawn from this plot is called the PV-loop, as the plots goes in a loop, against the clock with time. The different intervals, a, b, c and d, as described in the previous paragraph are also indicated in figure 4a and 4b. [10]

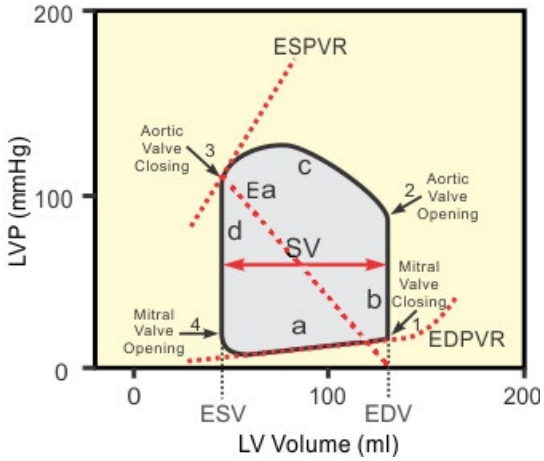
Heart load Afterload and preload are the mechanical loads imposed on the ventricle during ejection and at the end of the diastole respectively. The first mentioned parameter is indicted by the aortic pressure, ejection wall stress, total peripheral resistance and arterial impedance, while the second is dependent on the end-diastolic volume, end-diastolic pressure and the end-diastolic wall stress. By imposing different loads on the heart, heart function parameters can be extracted from the PV-loops. In figure 4b there are three indicated lines;

- End diastolic pressure volume ratio (ESPVR); Maximum volume for the heart, independent on load. Can be used to describe the contractility. The ratio is usual approximated as linear.

- End systolic pressure volume ratio (EDPVR); describes the stiffness of the heart walls, and its passive filling properties.
- Arterial elastance (Ea); affected by the arterial load. Gives a description of peripheral pressure.



(a) The left ventricle's pressure and volume plotted over a heart's cycle.



(b) The PV loop, based on previous plot. The Ea(added to original figure), ESPVR and EDPVR lines are indicated.

Figure 4: These figures shows the relationship between volume, pressure and time.[11]

Flow rate in vessels

An alternative for calculating the cardiac output from the stroke volume is by measuring from the flow through the aorta, which leads the blood out of the left ventricle of the heart, see figure **??**. Flow rate through a vessel can be described by equation 5.

$$\widetilde{Q} = \frac{\pi v_{max} r^2}{2} \tag{5}$$

Where $v_{max}$is the maximum velocity in the vain, which will be located at the center, and r is it's radius. See appendix B for details concerning this derivation. [28]

Strain Strain is a measure for stretch, or alternatively deformation, of an object. It is defined by equation 6

$$\varepsilon = \frac{L - L_0}{L} \tag{6}$$

Where L is the initial length and L_0 is the length after a stretch. Strain rate is the change in strain per time unit:

$$\dot{\varepsilon} = \frac{v_1(t) - v_0(t)}{L} \tag{7}$$

Where $v_1(t) - v_0(t)$ is the relative velocity between two moving points and L is the distance between them.

While velocity gives an impression on the global movement, strain rate subtracts motion due to tethering, and can hence be used as a measure of local heart function. This can be helpful for locating stiff tissue. As those regions will move due to heart motion, but will however not have internal movement due to the lack of contraction.[21]

2.2 Medical instrumentation

2.2.1 Ultrasound

Ultrasound imaging is a term that covers a broad span of ultrasonic imaging techniques. B-mode, M-mode and 3D ultrasound can be used to study anatomy, while continuous wave (CW) Doppler, pulsed wave (PW) Doppler, color flow (CF) and tissue velocity imaging (TVI) are used to analyze dynamics in the body, by measuring velocities inside the imaged field. Some of these techniques are described in this section.

Ultrasound imaging is a relatively cheap medical tool that can provide accurate and cost-effective measurements for diagnosis of patients in a non-radiative way, and is considered safe within the diagnosis specifications.

Another valuable property is the real-time acquisition and visualization of the imaged sector. This provides possibilities to view moving organs as the heart.[15]

The following paragraphs describe the relevant ultrasound modalities for this thesis, in addition to some that are not used but useful for conceptual understanding.

Ultrasound probes

The probe is the transducer for both transmission and reception of the signals used in ultrasound. Pietzo electric material is used to convert electric potential to mechanical force. By applying alternating voltage over these elements, mechanical waves can be transmitted in to a body, if there is an impedance coupling between the elements and the imaged object.

There are a variety of probes that provides different imaging views. The two main types are:

- Linear array; Data from the imaged object is collected from a shifting interval of the probe elements. And will hence get a squared image with approximately the same width as the probe.
- Phased array; Data from these probes are gathered with a steered beam. This is done by applying different phases on the elements. The resulting image is fan-shaped, due to collection of data in a polar coordinate manner.[19]

While most probes are applied on the skin non-invasively, the transesophageal probes are inserted in the esophagus. This obtains a closer view of the heart. The probe is a phased array, and can be steered from a steering handle. The movement of the probe is; up and down the esophagus; physical rotation of the probe; flexing the tip mechanically in two directions; and in multiplanar probes, the view can be rotated as well. Placement of such a probe can be seen in figure 5.[23]

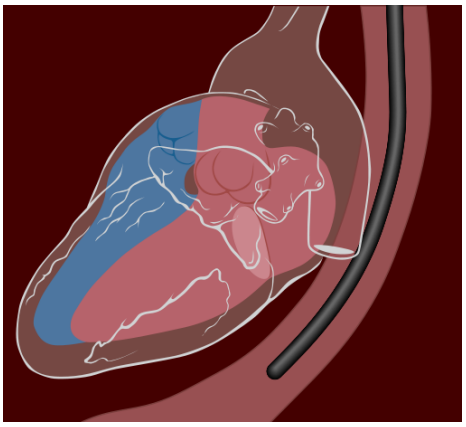


Figure 5: A transesophageal probe inserted to a position close to the heart.[16]

M-mode imaging

M-mode is the simplest form for anatomic imaging with ultrasound. It images one strip of a volume over time. The principle of time-of-flight is used to calculate where a reflection comes from, after a wave is transmitted into the region of interest of the body. Equation 8 describes this relationship.

$$r = \frac{ct}{2} \tag{8}$$

Where t is the time it takes for a pressure wave to travel a distance r into an object and back. The velocity for the pressure wave in the imaged object is defined by c.

During examination pressure waves are sent with intervals equal the time it takes for a wave to reach the deepest range of interest and back. The rate at which the beams are sent is called the pulse repetition frequency (PRF).

$$PRF = \frac{c}{2r_{max}} \tag{9}$$

For each sent beam, the received echo is processed and the magnitude of the signal can be plotted at the corresponding range in the image. As the consecutive waves are sent, a time varying image can be displayed.

Figure 6 shows an example of m-mode imaging. In modern ultrasound machines M-mode is usually combined with B-mode(see next paragraph) images, to select the strip to be studied from a wider sector, as seen in figure 6.

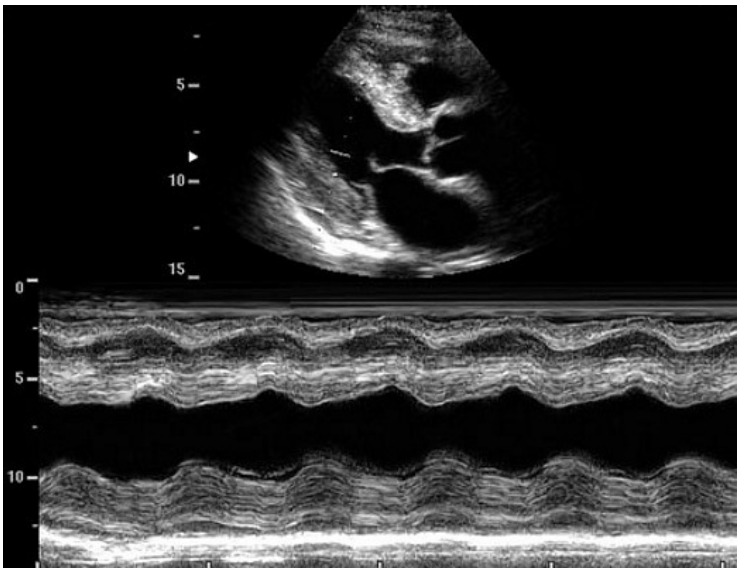


Figure 6: M-mode imaging of the heart; the m-mode image is selected(pointed line) from the B-mode image at the top. Times moves along the x-axis [4]

B-mode imaging

With a phased or linear array, the beam can be steered or moved in more than one direction while scanning. When this is done adequately fast, data from a two dimensional sector can be gathered at a rate fast enough to display time varying two dimensional images in real-time. The frame rate for a B-mode image will naturally be lower than for M-mode as a full frame can be rendered first after a whole sector has been scanned. Number of frames per second (FPS), at its simplest form, is described by equation 10.

$$FPS = PRF * N_{lines} \tag{10}$$

Where $N_{lines}$is the number of scan lines in the width of the imaged sector.

3D imaging

3D ultrasound expands the dimension further by retrieving data from an extra direction. This requires an extensive number of beam forming techniques in order to gather enough data from the imaged volume to display the images at an adequate frame rate. To interpretable these data, segmentation has to be applied.

Continuous wave (CW) Doppler

To get information of the velocity of moving objects in the body, the Doppler Effect can be utilized by transmitting a wave with a frequency. A moving object will generate a Doppler shift when it reflects such a wave, given by equation 11.

$$f_D = f_0 \frac{2v}{c} cos\alpha \tag{11}$$

Where f_0 is the carrier frequency, v is the object's velocity and α is the angle between the beam direction and the velocity vector.

In CW Doppler a continuous wave is transmitted into the region of interest, at the same time it is continuously received[1]. Due to this lack of pulsing, the reflected echo cannot be associated with the range it comes from; hence will the received Doppler spectrum contain Doppler shifts from the whole illuminated region. A constructive effect of this continuous wave is an, in theory, infinite range of the frequency spectrum.

Pulsed wave (PW) Doppler

PW Doppler is a technique to handle the ambiguities in range for CW Doppler. As the name implies, the method transmits pulsed Doppler beams. This makes it possible to link the received signals to a range, as described by equation 8. However, when the pulse is divided the maximum Doppler shift is limited by the Nyquist limit, given by equation 12.

$$f_{D_{max}} = \frac{PRF}{2} \tag{12}$$

Due to the Nyquist limit the highest measurable velocity is limited by the following equation:

$$v_{max} = \frac{c * PRF}{4 f_0 cos\alpha} \tag{13}$$

In practice an operator can navigate in B-mode images to select the region from where the Doppler beam is focused, and hence where the Doppler spectrum originates from.

Color flow (CF) imaging

A technique used to collect velocity data over a full region of interest is called color flow imaging. The velocity data obtained is drawn on top of B-mode images with color codes, hence the name.

There are three parameters that are estimated for each resolution cell in this modality; the power of the signal, the velocity and the bandwidth. These parameters are calculated from the autocorrelation of the complex demodulated signal[3].

[1] Usually done by splitting the probe array in two; on for transmit and one for receive.

$$P = R(0) \tag{14}$$

$$\bar{\omega} = \frac{1}{T_{PRF}} arg\left[R(T_{PRF})\right] \tag{15}$$

$$=> \widetilde{v} = \frac{c * arg\left[R(T_{PRF})\right]}{4\pi T_{PRF} f_0} \tag{16}$$

$$B^2 = \frac{2}{T_{PRF}^2}\left[1 - \frac{|R(T_{PRF})|}{R(0)}\right] \tag{17}$$

Where $R(\tau)$ is the autocorrelation function of the signal. And arg is the angle component of the complex autocorrelation. T_{PRF} is the time between sampling of each sample volume. While the velocity is usually the parameter that is displayed for interpretation, the power and bandwidth are used to calculate whether tissue data or velocity data should be displayed for the specific volume.

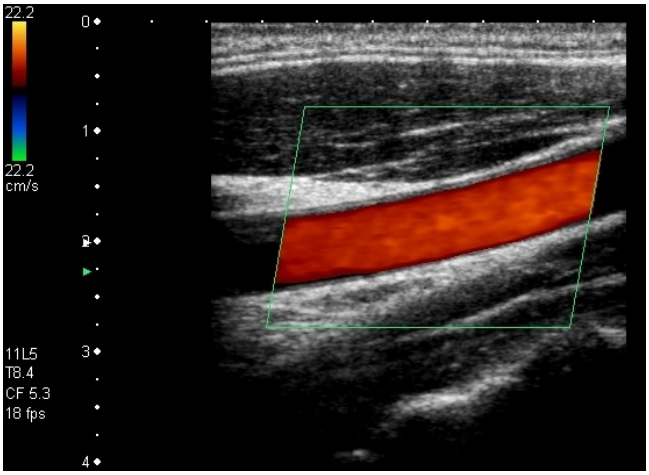


Figure 7: Color flow image of the carotid.. The framed region is selected for color flow calculations.[7]

Tissue velocity imaging (TVI)

Where CF is optimized for visualizing and quantifying blood flow, TVI is made for tissue velocity. To obtain these data the same principles as for CF is used, however, the equipment has to operate on lower sensitivity, as the velocity data from tissue is lower than from blood. [12] Figure 8 shows an example of TVI imaging.

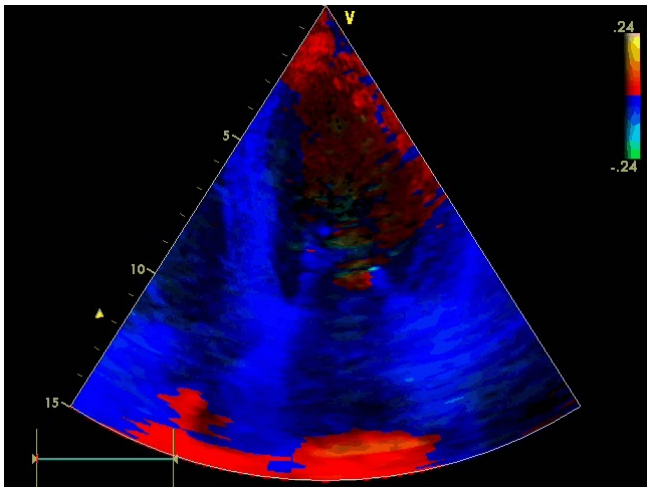


Figure 8: TVI image. The entire image is color coded. Stronger contrast can be seen where there is tissue.[8]

2.2.2 Coronary catheterization

For acquisition of cardiac measurements, catheters are used. These are wires thin enough to be inserted into a vessel, from where it can be further moved into a desired position in a certain vessel or heart chamber. Figure 9 shows an example on insertion into the left ventricle.

While there are catheters for many purposes within diagnostic and intervention, only conductive catheters(referred to as catheters) are relevant in this thesis. These catheters use the bloods conductivity for measurements as pressure, volume and ECG. [5]

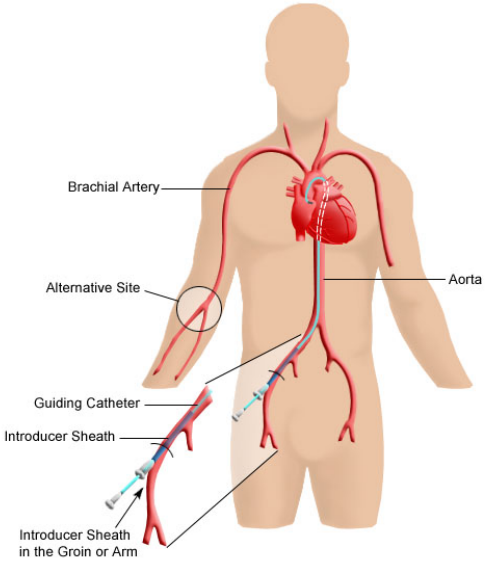


Figure 9: Catheter placement; here inserted to get access to the left ventricle. [22]

3 Methodology

This section describes theory more directly relevant for the application, as well as ideas around the implementation and experimental set-ups.

3.1 Data acquisition

Data acquisition in this application comes from two sources: an ultrasound scanner; and a general acquisition unit, called a DAQ from now on. The DAQ is used as an interface between different medical instrumentation units and the application described.

3.1.1 Sampling

When analog data is acquired and converted to digital format, it is quantified by an analogue to digital converter (ADC), in both time and amplitude. The amplitude quantification is defined by the number of bits in the resulting value. The range for digital values are defined by equation 18.

$$range = 2^n \tag{18}$$

Where n is the number of bits.

The quantification in time is defined by the frequency of the ADC. Where the relationship between the sampling interval, T, and the frequency, f, is defined by equation 19.

$$f = \frac{1}{T} \tag{19}$$

The Nyquist criteria state that the maximum frequency that can be reconstructed after sampling is half of the sampling frequency. This means that data must be sampled at a frequency twice the highest frequencies components that the signal of interest has.

It has been shown that physiological pressure and flow waves in mammals are limited to around 10 harmonics [1], where the fundamental frequency is related to the body weight. The fundamental frequency equals bpm/60, see equation 20.

$$bpm = 282 * weight^{-0.32} \tag{20}$$

The maximum frequencies, when resting, will therefore be found around:

$$f_{max} = 47 * weight^{-0.32} \tag{21}$$

Needed sampling rate, due to the Nyquist criteria, is two times f_{max}. For a test environment involving rat measurements, a sampling rate $> 150Hz$ can be necessary, as flow and pressure will have the highest frequency components of the measured signals. Volume data from the heart are assumed to have the

same variations as flow and pressure, and hence can be sampled under the same requirements.

Minimum sampling rate for ECG has been found to be 250 Hz[18], which makes it the signal with the highest requirements for sampling.

3.1.2 Filtering

Filtering of signals can be valuable to remove high frequency noise. The Nyquist criteria will also play a role for filtering, as you can only filter frequencies lower than half the sampling rate.

A finite impulse response (FIR) filter can be used by calculating it's coefficients using an implementation of the Parks McClellan algorithm[20][26].

The structure of a FIR filter is shown in figure 10. It can be seen that the filter is causal as the output of the filter is only dependent on previous samples. This makes the impulse response finite and the impulse response will be equal its coefficients.

An example of a FIR-filter's impulse is plotted in figure 11, where the x-axis values represents the coefficients $b_0,\ b_{1...}b_N$, from figure 10. This shows that there will be a delay in the filtered signal that will equal half the filter length, where the maximum is found.

Figure 11 shows the frequency spectrum to the same filter example. As this is a digital filter the frequency axis is normalized by half the sampling rate. The Parks McClellan algorithm provides coefficients with linear phase as seen in the same figure. This means that all frequencies have the same delay.

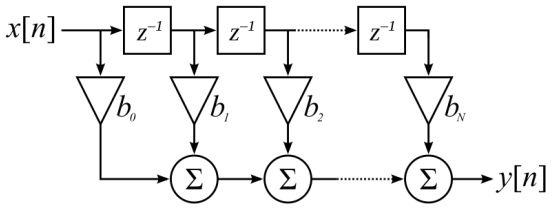


Figure 10: A FIR filter schematically drawn. The Z^{-1} elements represents a delay. $b_0,\ b_{1...}b_N$ are the filter coefficients

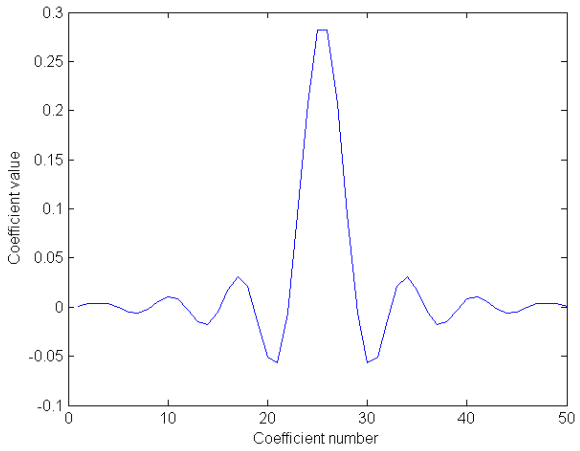


Figure 11: Example of a FIR filter's impulse response with an order of 50.

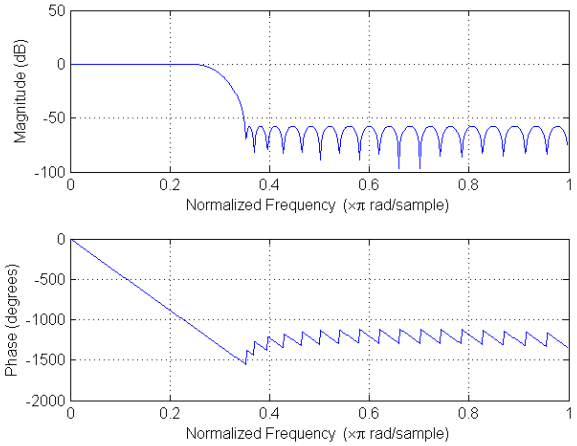


Figure 12: Example of the same FIR filter's frequency response example. The cut-off frequency is found around 0.3.

3.1.3 Calibration

As data is sampled from a voltage potential, they will have their unit defined thereby. Hence is calibration of these sampled data necessary to obtain the unit

they actually represent. The relationship between the sampled physical data and the transmitted voltage value is assumed to be linear, and can hence be described with the following relationship:

$$y = ax + b \tag{22}$$

Where y is the calibrated signal; x, is the received signal; a, is the linear constant and b is the offset constant. The constants a and b can be set by using two known reference values, from at two different times; and two received values, sampled at the respectively same times. The two calibration constants can then be set with equations 23 and 24. The unit for pressure, Pa, is used as an example.

$$a = \frac{y_2 - y_1}{x_2 - x_1} \left[\frac{Pa}{V}\right] \tag{23}$$

$$b = y_2 - ax_2 \, [Pa] \tag{24}$$

It becomes obvious that the calibrated signal, y from equation 22, now get the right unit.

The method relies on precise calibration signals provided from the instruments.

3.2 Ultrasound streaming

To gather data from the ultrasound machine to the application described in this paper, a streaming client is used. This functionality is developed at the Institute for Circulation and Image Diagnosis, which lies under the Medical Faculty at NTNU, Trondheim. The client supports streaming of many of the ultrasound modalities, including B-mode, CF and TVI imaging, which are of most relevance in this application. Streaming of ECG and an auxiliary signal are also supported. The client establishes a connection with the scanner, and as it receives data over IP it converts and sorts them into a VTK format. These data are then made available through functions from the library provided by the streaming client.

The maximum supported streaming rate for the streaming client is:

$$FR_{max} = 60 \tag{25}$$

3.3 Traces from ultrasound data

Flow and strain rates are parameters that can be derived from the velocity information that is streamed from the ultrasound scanner.

Flow rate

Equation 5 can be used as an approximation for flow, under the assumptions described. The radius of the vain can be set given by the operator.

The acquired velocity data will be the beam-direction component of the actual velocity. This can however, be corrected by applying equation 26.

$$v_{corrected} = \frac{v_{received}}{-sin(a+b)} \tag{26}$$

Where the angles a and b are the beam and probe tilt[2] respectively, as illustrated in figure 13. The beam tilt angle, a, is provided by the streaming client, while the angle b must be set. This can be done by the user, by picking two points in the direction of the flow. And from those coordinates calculate the angle.

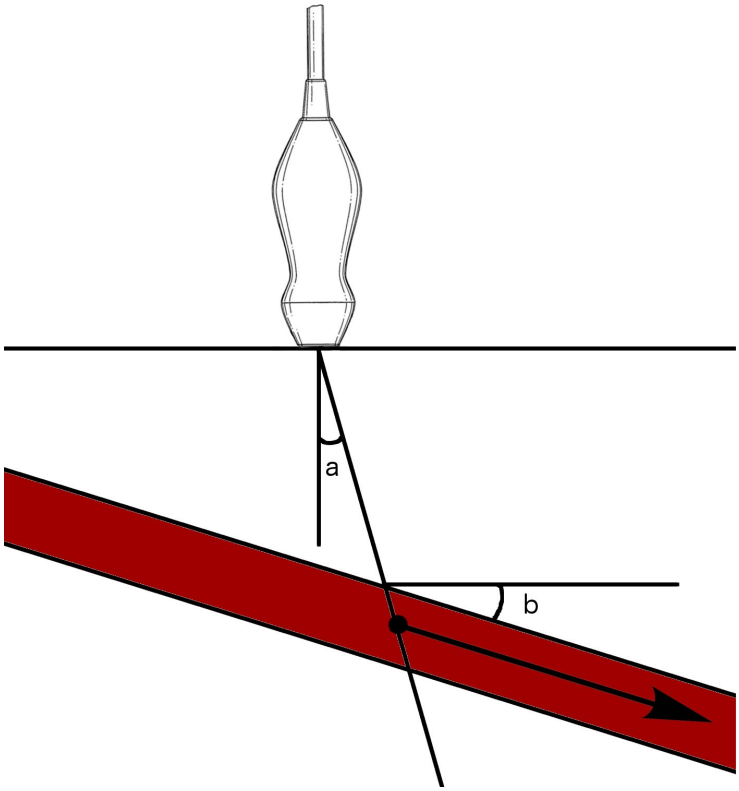


Figure 13: The relevant angles for angle correction. a is the beam tilt, while b is the probe tilt, relevant to the blood flow.

[2]Related to the blood flow direction

Strain rate

To calculate strain, equation 7 can be used. Two reference points are needed to calculate the relative velocity between them, in addition to the distance that separates the points.

3.4 Synchronization

As data acquisitions are collected from two different systems, synchronization is needed for analysis under one time reference. The ultrasound scanner has an auxiliary (AUX) input channel, which can be streamed with the streaming client, in the same way as the ECG signal. This is used to send a reference signal in order to find the delay between the two systems.

This is done by transmitting a known signal from the application, and adding this signal with the local time reference to a buffer. The signal is then transmitted, from one of the DAQ's output channels, to the AUX input at the ultrasound scanner. The streamed AUX signal is continuously added to a buffer at the application side, with time stamp from the local time reference at reception.

The transmitted signal is a short spike to easily distinguish the time difference between the sent and received signals. The method to find this time difference is a search for the largest amplitudes of the two signals. The time duration between the two respective peak tops' time stamps will then be used as the synchronization delay between the two.

There are two assumption that are made for this method to be accurate:

1. There is no or a negligible delay in the acquisition instruments that are connected to the DAQ.

2. The delay from acquisition at the scanner to the reception in the application is not fluctuating.

3.5 Implementation

3.5.1 Language and libraries

The program has been written i C++, compiled with Microsoft Visual studio 2010. QTs API has been used for the graphical user interface (GUI), which is a cross platform framework. It supports most of the elements used in conventional GUI applications.

For graphical rendering and visualization of the different data, both from the DAQ and the ultrasound frames data, the Visualization Toolkit(VTK) has been used, which is also cross platform. QT and VTK has implemented objects that make it possible to link the graphics constructed with the VTK libraries to QT objects in the GUI.

An objective of the implementation has been to separate the GUI from the bottom platform of the application. In that way, it's easier to later change the GUI implementation to for example a license free library for commercial use.

3.5.2 Multiple threading

While the application is running, there are a variety of tasks to be handled; GUI interaction, rendering, data acquisition, data transmission and more. Some of these different tasks have different requirements for timing that has to be met, and cannot all be run under one thread as consecutive tasks. These tasks have been divided into the numbered list under, and illustrated in figure 14. The three first are threads at the bottom of the application's hierarchy, relative to the distance to the displayed data.

1. DAQ acquisition thread; controls the sampling from the DAQ. All data is time stamped and added to a buffer that is available from the plotting buffer thread.

2. Ultrasound data thread; is the link to the streaming client. Acquires ultrasound frames, ECG and AUX, and sorts them into buffers.

3. DAQ transmission thread; transmits data for synchronization and adds the samples to a buffer with local timestamps.

4. Plotting buffer thread; Makes sure that the right interval of the data buffer is inserted in the plotting buffer. The sampling interval in the plotting buffer is lower than in the acquisition buffer, to assure data integrity for saving.

5. GUI thread; due to the QT implementation, all interaction with the GUI has to be done from this thread. The rendering functions for ultrasound frames and the different plots are called from here, in addition to user interaction.

The program's class structure is build up based on the same hierarchy as the threads. The arrows in the figure are related to the flow of data, and the function calls for retrieval between the classes, goes in opposite direction.

Even though multiple threads are used to prevent queuing of tasks, they will be sharing resources. One example is the buffers, which are used both in the acquisition thread and in the plotting thread. In order to prevent data ambiguities when accessing, the common resource has to be locked to one thread at a time, which will regularly cause one thread to wait for another. However, the effect of this should be low compared to a non-threading environment.

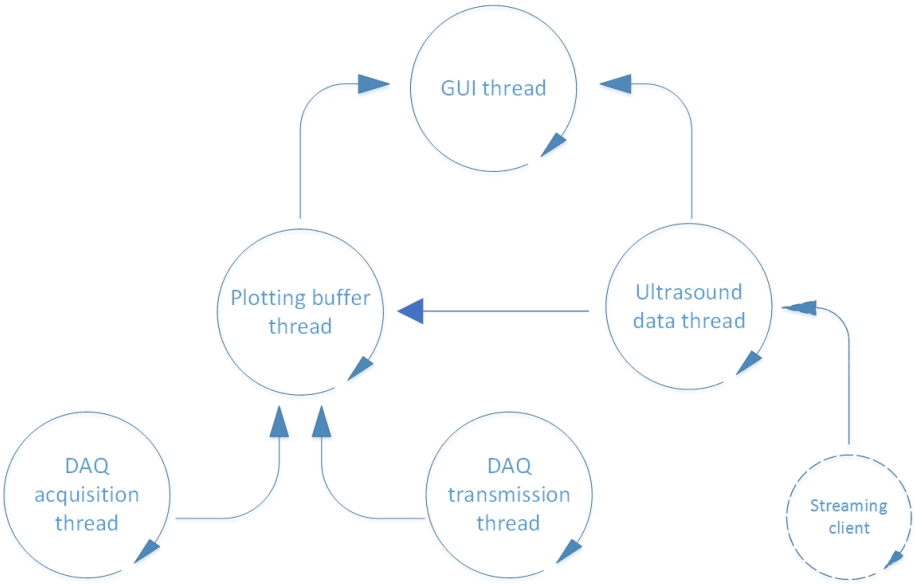


Figure 14: The separation of threads in the application; the arrows indicate the direction of data flow.

3.5.3 Timing

The application has an internal clock that is initialized from zero at start-up, called *last time*. All data gets a time stamp from this clock at reception, with adjusted delay if synchronization has been applied.

The plotting buffer uses another time called *plotting time* which is equal to the *last time* if the program wants to show the latest data. However, if the user moves in time the *plotting time* gets a reference delay from the *last time*, and will continue to play with this distance from *last time*. This makes it possible to move in time. Data is retrieved from the data buffers based on a time reference. To prevent a search through the entire buffer at each data retrieval a index is saved when data is picked, which is used as the starting point for the consecutive sample retrieval.

When a recording is started a time stamp is saved for the *start time* and another one for the *end time*. If the user chooses to save, this interval is collected from the buffers and saved as a recording to the projects file..

All trace data, which includes DAQ data, streamed ECG and AUX and the traces from ultrasound is saved to infinite buffers in memory. The frame buffers however, have ring-buffers. These have a fixed size, where new data overwrites older data when the buffer fills up. This was done to prevent the application from using more than the maximum amount of memory available for a process, which is about 1.8 GB.

3.5.4 File format

Development of a functional file format has been a key feature for the application. It was decided to use the HDF file format, due to its C++ compatibility, and exportability possibilities like integration with Matlab. HDF has a file reader that can be used to view the content of saved HDF files.

As the program is designed to be used for clinicians, a set-up was constructed to handle projects, sessions and multiple recordings;

- Projects; is designed to contain all data associated with a project, as for example a scientific study where data is collected over a larger time period. A project is a placeholder for multiple session, in addition to a project name and comments.

- Sessions; if the user wishes to acquire or view data in real-time, a new session has to be created. A session contains all information associated with an acquisition set-up; a name, comment, date, user name, all settings and recordings. The session settings specifies; the active DAQ channels and their respective settings as calibration, plot color, filter settings, measurement name and the synchronization constant; and also the ultrasound streaming settings. If the user record data they are saved in relation to a session. It is also intended that the user can open an existing session that contains recordings for analysis of already recorded data. This is, however, not fully supported yet..

- Recordings; contains all raw data, including their respective time stamps. This includes all data from the DAQ, ultrasound frames, ECG from the scanner and flow and strain traces if active. The recordings are also represented with a name.

3.5.5 Design and GUI

The user interface has been an evolving and ongoing process through the development of the application. It was important to make it possible for the user to be in charge of the setup, in order to make it flexible for different applications. The main set-up had to have functionality for a flexible number of plots from the DAQ and to choose whether to include ultrasound streaming or not.

Another goal of the GUI implementation was to find a functional way to organize the different plots and data settings in a way where all settings are easy available without taking up to much space from the data plots.

3.6 Experimental set-ups

For testing on actual data, two experimental set-ups were conducted, where the main testing were performed on pigs and the other on a human candidate. Both tests included ultrasound streaming with a GE vivid E9 scanner[6].

The application is compatible and tested with National Instruments' USB-6251 and USB-6259[13] for acquisition of measurement data(DAQ) through its

BNC inputs. The DAQ were connected to a computer with USB, which made it possible to integrate reception and transmission of data from the application through its C application programmable interface (API).

The USB-6251 has 8 input and 2 output channels while USB-6259 has 16 input and 4 output, all with BNC connectors. Both has a sampling rate of 1.25MSamples/s, which is shared by all active channels. The resolution is 16 bits, over [-10, 10] Volts.

3.6.1 Pig trials

At the operation room the USB-6259 was used to connected to the relevant instruments. Two conductance catheters were used for pressure, volume and ECG measurements. These was both placed in the left ventricle of the pig by the clinical personnel that lead the operation.

A Leycom Sigma 5DF was used for the volume and ECG measurements, and a Sentron Pressure Measurement System for pressure. Both devices had BNC outputs that could be connected to the DAQ. Calibration of both instruments was done by setting a fixed value from the devices and then use the application's calibration functionality.

A transesophageal probe was used to acquire ultrasound data. Non-invasive ECG measurement was also acquired by the ultrasound machine.

Figure 16 shows this schematically and figure 17 shows the equipment in the trial environment. For further details regarding the connection set-up, see appendix E.

Figure 15: Pig trial; here imaged with an x-ray machine. This was however removed during testing.

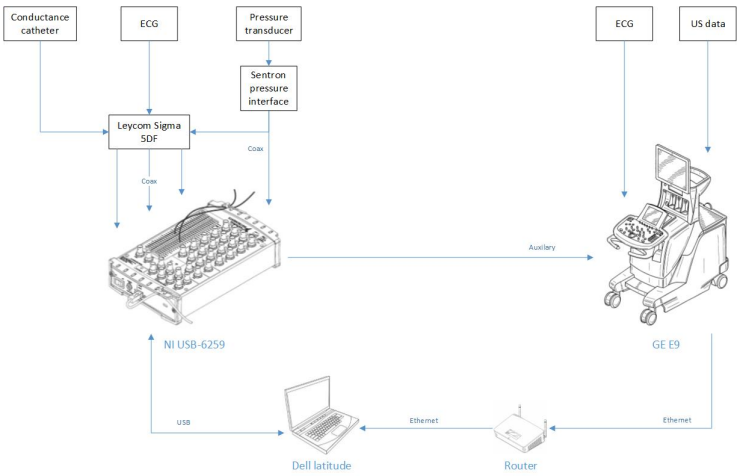


Figure 16: Experimental set-up for the pig trial of the two acquisitions units; the ultrasound scanner and the DAQ.

Figure 17: Instrument rack and the ultrasound machine in the background.

3.6.2 Human trial

This was a simpler set-up conducted at the ultrasound lab, and involved only an ultrasound machine. It was set up to do additional testing on the trace data from the streamed ultrasound images. The scanner's ECG was also connected to the test person.

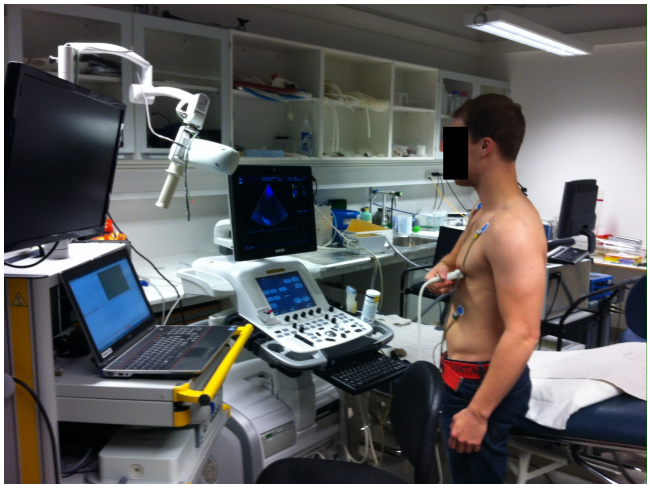


Figure 18: Experimental set-up for human trial. Only an ultrasound scanner with connected ECG measurements was used.

4 Results

This section covers the most relevant results from the testing, with a focus on the GUI design, achieved sampling/frame rates, synchronization, usability and file handling.

The figures from the practical results usually contain a screen shot to show how the program worked in practice, with relevant plots in a separate figure. The latter figure will show the calibrated values, while the screen shots were not necessarily taken after calibration was applied.

4.1 GUI implementation

An overview and naming of the GUI elements used can be found on QT's web pages[17].

4.1.1 Start-up menu

At start-up the user is presented for a menu to initialize the session. This menu has two main tabs, one for generating a new project and the other for loading a project, as shown in figure 19 and 20, respectively.

If a new project is selected the user can choose to add information to the editable fields under the *Project* frame and on the left side in the *Session* frame. On the right side of the latter the set-up is managed. Here the user has the choice between loading a set of settings from a previous session, or a new custom made set-up. The ultrasound settings are set with the check-boxes, while in the DAQ settings the user can choose to add or remove an input, and for each input choose its channel, as well as a preset type of measurement.

The other way of starting a session is to choose to load a project. The user will here choose a project from the top combo box and from there choose to either construct a new session or loading an existing for analysis of recorded data. The new session page in the toolbox menu gives the same possibilities as when loading a project. Under the page for loading a session, the available sessions are listed. When one is selected, the user is presented for a generated description of the session based on the settings made and user defined comments. The loading functionality is, as mentioned, not fully developed and will not be studied further.

When all settings are set, the user starts the program with the *Start session* button.

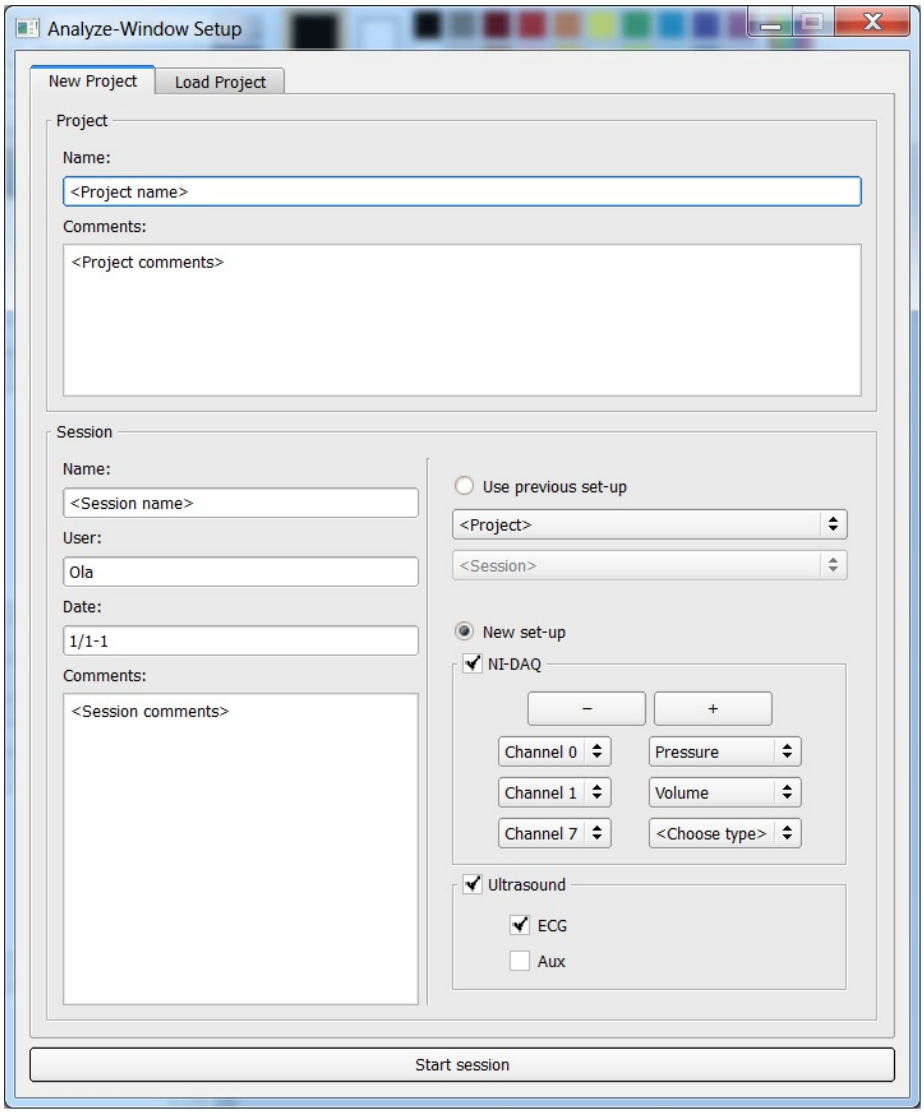


Figure 19: The *New project tab*. When a new project is selected, the user must start up a new session. Settings can be loaded from previous sessions or a new set-up can be chosen.

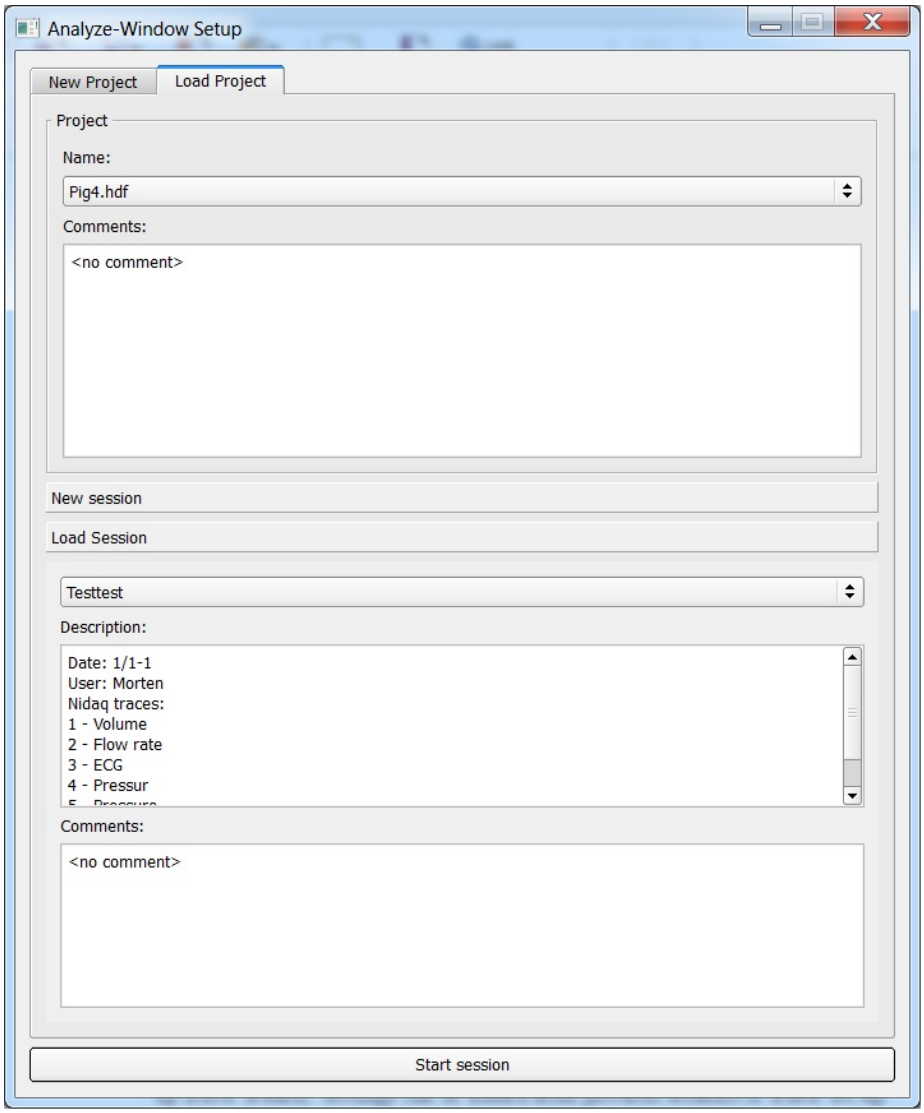


Figure 20: The *Load project tab*. The user can here choose to start a new session for the project or open an existing. When a project is loaded

4.1.2 Main application window

The main application is organized into an upper and a bottom section. The bottom section handles time and the capturing capabilities.

A slide bar lets the user move in time. When this is dragged the time line plot follows the movement of the slider and the red marker at the right side of the plot, will represents the corresponding time of the sliders position. Capturing capabilities is found in the center of the bottom.

The upper top and main section of the main window is organized in tabs. These can also be floating windows which gives flexibility for the organization of the windows. The different tabs are further described in the following section.

Mixed view The idea behind the mixed view tab is to display information content to the user with as few disturbing elements as possible. The user has the ability to choose which elements to view, and the settings for the different data is done within the tab their data originate from.

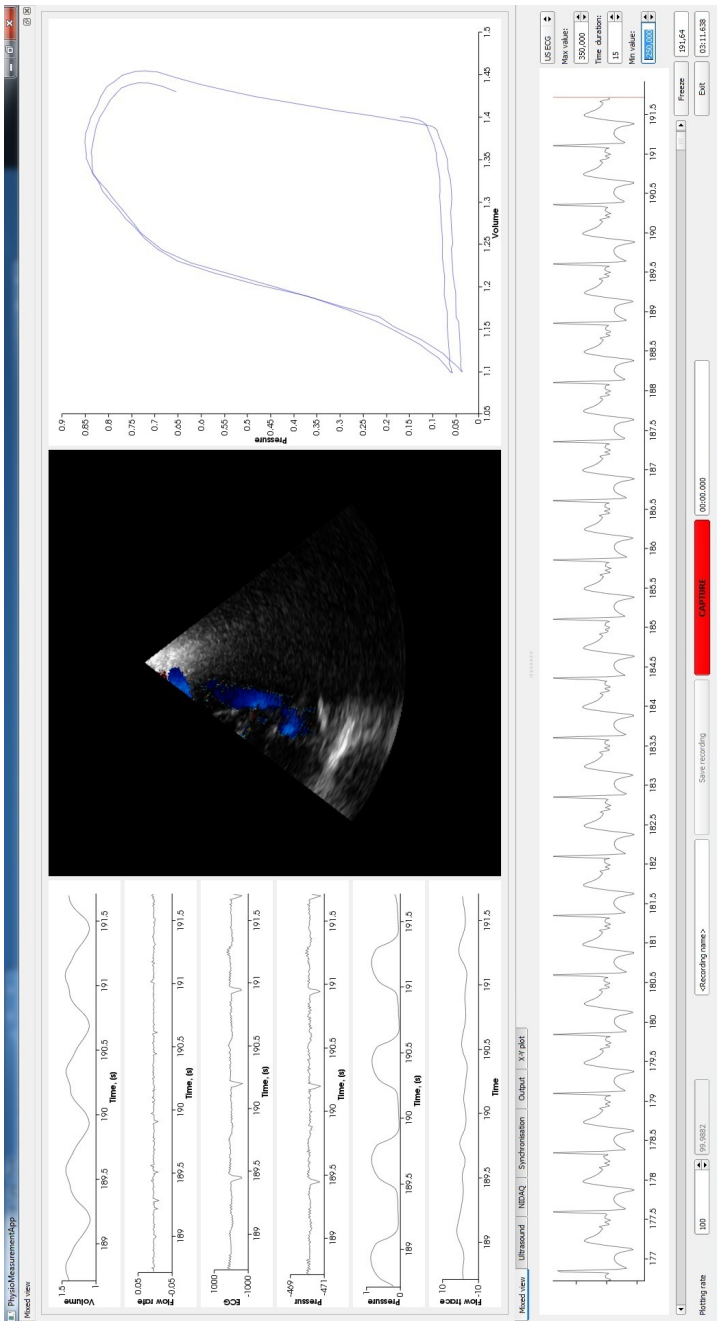


Figure 21: The *Mixed view* tab; this is where data from the different sources are gathered in one view. The column to the left displays all trace data, the middle shows the ultrasound frames, while the plot to the right is the X-Y plot currently plotting pressure and volume. The plot on the bottom plot is used as a time reference, represented by ECG values.

Ultrasound In the ultrasound tab, the supported settings for the streamed data, except AUX, can be seen and adjusted. The ultrasound frames, ECG, flow trace and strain trace have their respective settings page in the toolbox. ECG and ultrasound frames can here be chosen to be included in mixed view.

The flow rate estimation lets the user generate a trace plot calculated from the ultrasound velocity data. To start, the user activates the plot and selects a trace point from the ultrasound images. The trace point is intended to be selected from the center of the vain, and with a radius chosen from the spin box, equation 5 calculates and plots the flow rate continuously. A second trace point can also be selected, and makes it possible for angle correction and averaging. The second point is selected from another center point in the vain with a small distance from the first point. The line these points make should be in the direction of the blood, as shown in figure 13. The beam tilt angle is collected from the streaming client. The number of averaging point between the two trace points can be selected by the user.

Strain rate estimation is based on equation 7, and the user's interaction by selecting two trace point for calculation of relative velocity. The length between the points is also specified by the user with a spinbox.

Filtering, as described in section 3.1.2, can be applied to both trace plots.

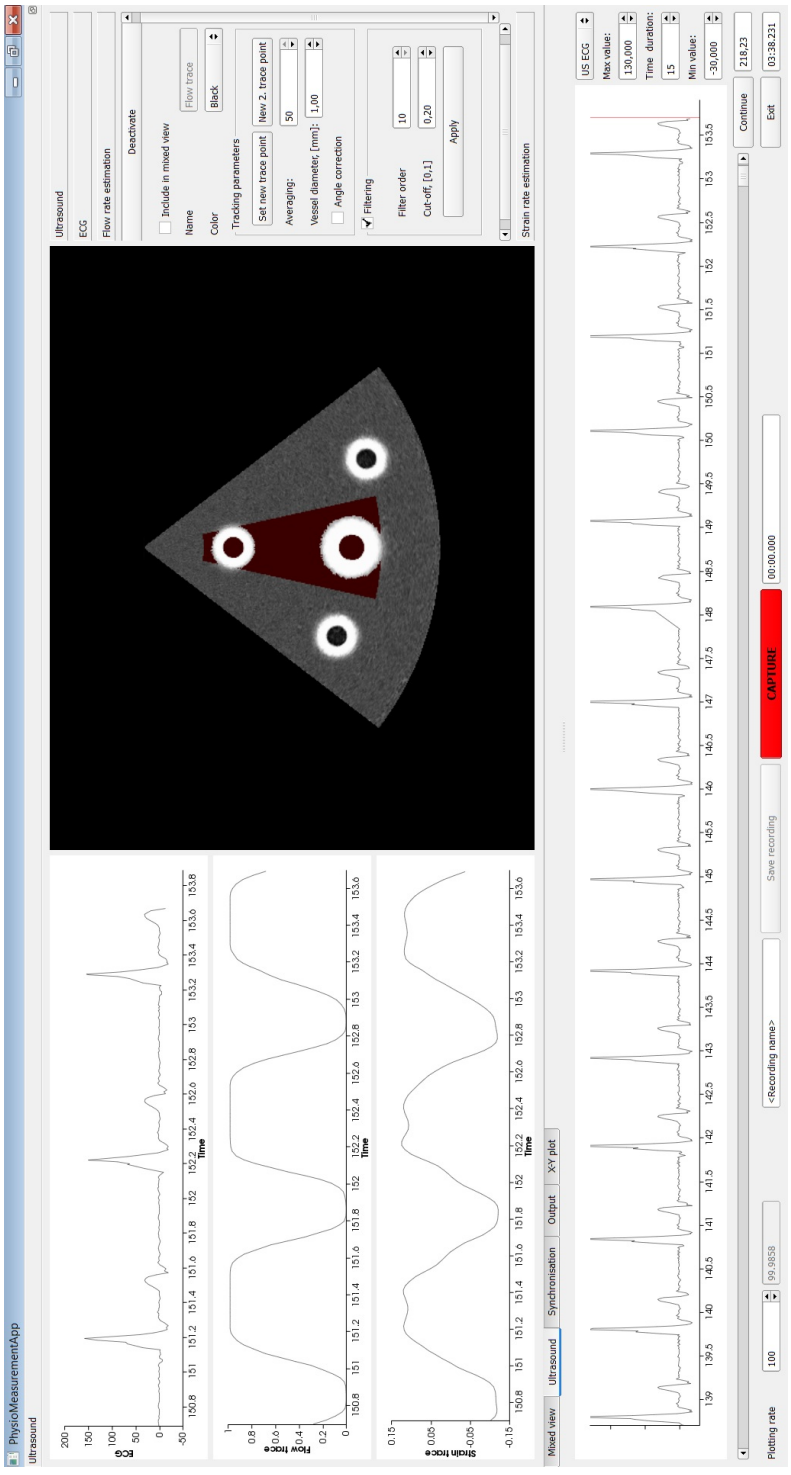


Figure 22: The *Ultrasound* tab; ultrasound frames and ultrasound traces with their respective settings are located here.

NIDAQ All activated channels from the (National Instruments) DAQ are plotted in this tab. The plots have their settings in the toolbox on the right side of the plots. One plot have its settings in the toolbox at a time. Which one is selected from the combo box on the top. The different pages has the following functionality, confer with figure 24.:

- Measurement labels; this gives information regarding which channel the trace originate from and how it should be displayed in terms of label, color and the choice of including it in mixed view.
- Calibration; From equation 23 and 24, calibration can be set with the input from two reference values and two received values. The made calibration constants can be applied to see the result on the graph, and saved to file for later use if wanted. Unit conversion has not been implemented, but is prepared for later work.
- Sampling; the sampling rate can be chosen by the user from the combo box. Window length for the plot can also be set. This will also affect the window lengths for the plots if they are added to mixed view. The settings made on this page are identical for all DAQ measurements.
- Filtering; gives the user the ability to activate filtering on the respective plot, according to the implemented filter implementation described in section 3.1.2.
- Statistics; by pushing the *Update* button the user get the mean and standard deviation of the measurement within the window length presented on the respective displays, which is especially useful during calibration.

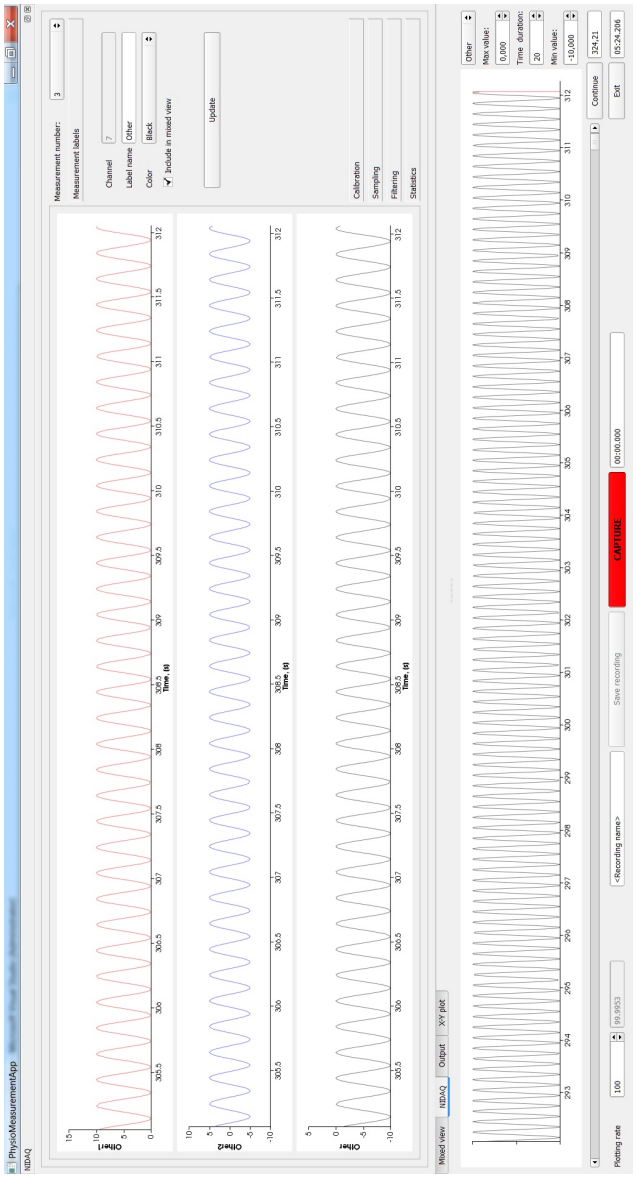


Figure 23: The *NIDAQ* tab. All active channels are plotted here. With relevant settings on the right side.

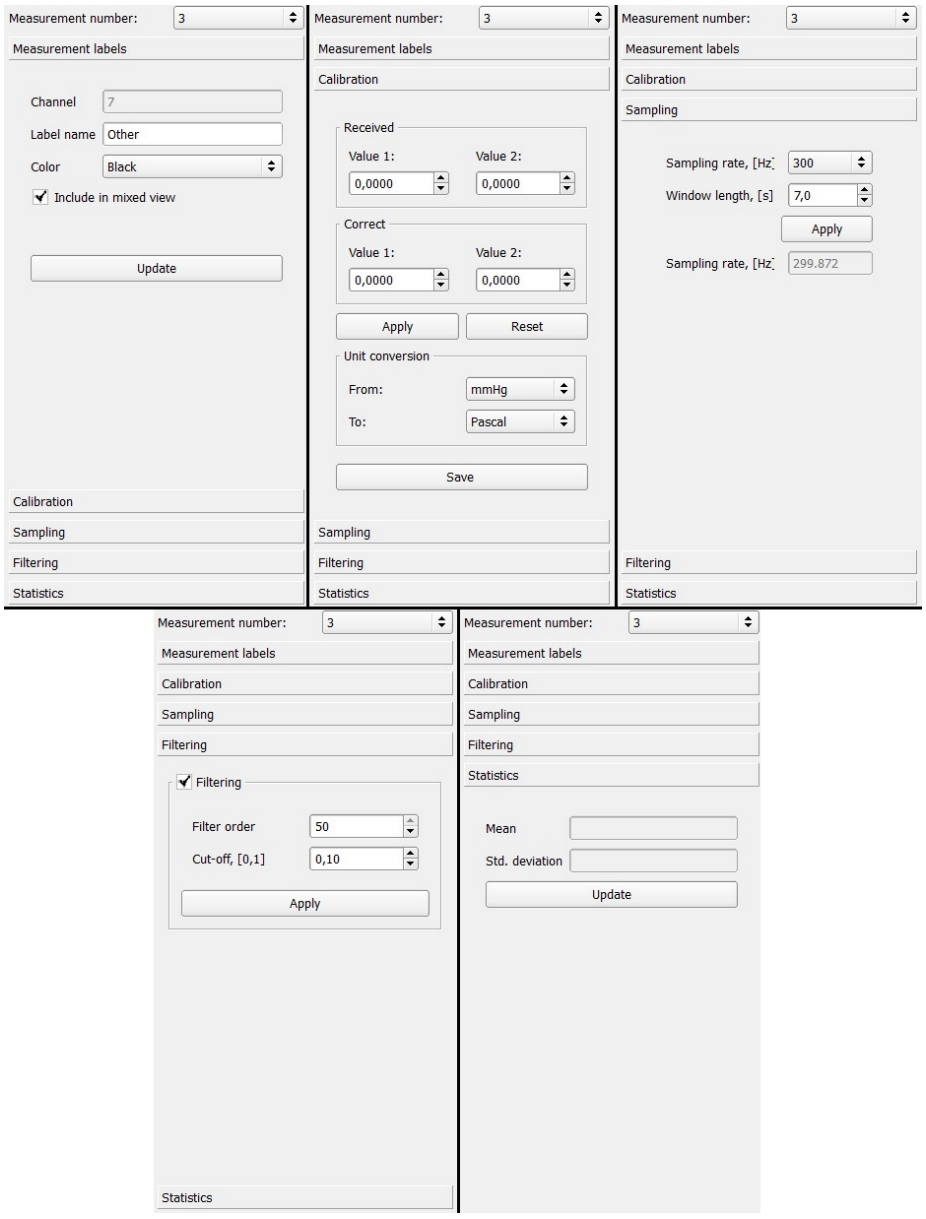
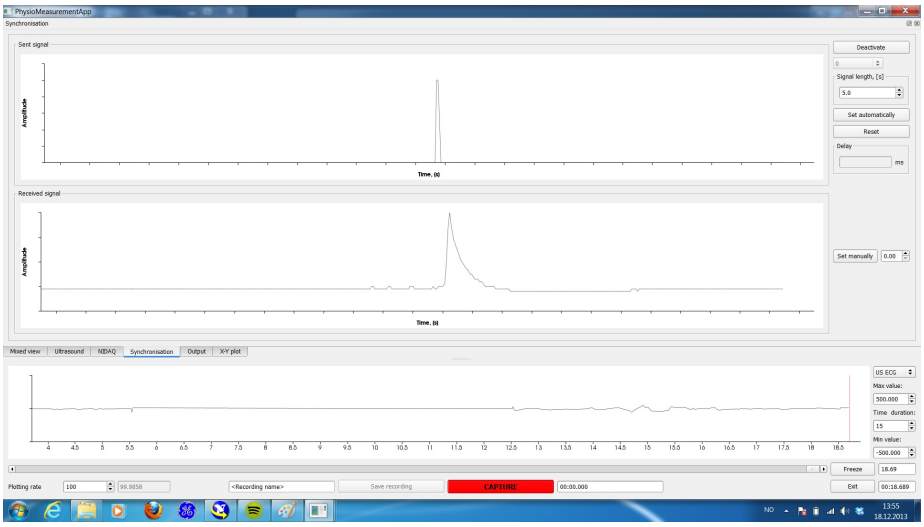


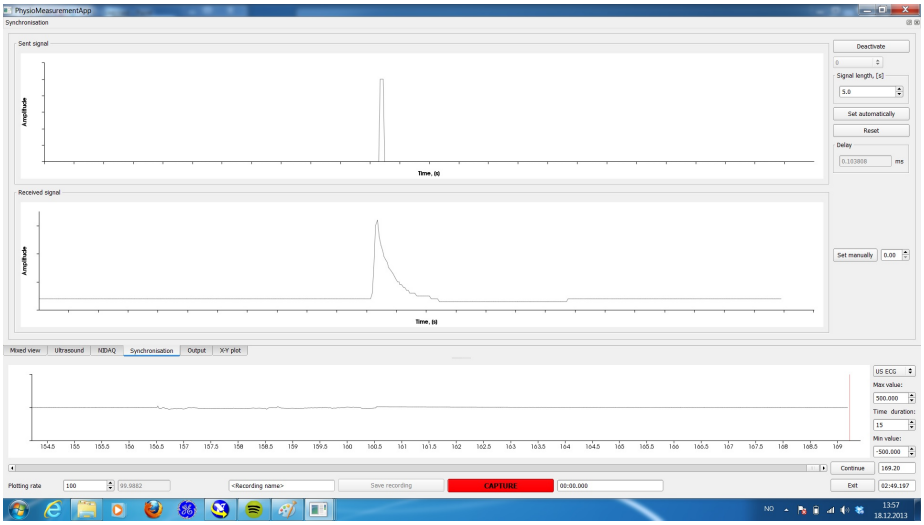
Figure 24: DAQ toolbox; shows all available settings for the DAQ plots

Synchronization The synchronization process is described in section 3.4, and is applied in the synchronization tab. The duration of the synchronization signal is set in the spin box. One signal duration has one reference peak.

When one peak from each signal, transmitted and received, is visible in the plot the *Set automatically* button will find the delay between the two signals. This constant is then applied to the DAQ signals, as this is the unit with the smallest delay, and saved to the session file.



(a) Before sync



(b) After synch

Figure 25: The synchronization view. show the synchronization signals before and after the delay is found. The plots does however not look completely aligned, this is due to difference in the render time between the plots, as discussed later in the thesis.

General GUI functionality The GUI provided in the application supports flexibility for organizing the windows, beyond the mixed views possibilities. The tabs can be undocked and moved to for example an external display, or side-by-side with other docked or undocked windows. Closing of tabs is also possible. See appendix A for a selection of set-ups.

4.2 Data acquisition

There are three different sampling rates for data in the application; one for the DAQ data; one for the ultrasound frames, also called frame rate, which will also be the sampling rate for the velocity data used in the flow and strain traces; the third is the sampling rate for ECG and AUX from the scanner. The ECG and AUX samples from the scanner are transmitted in bulks of multiple samples, and does hence achieve a higher sampling rate.

4.2.1 Acquisition from the DAQ

The sampling rate (SR) for the DAQ can be set by the operator within the application. By varying this rate under different recordings the effective sampling rate was found by calculating the time interval between the saved samples. It became obvious that the uniformity of the sampling rate was dependent on the rate, as the deviation increases at higher rates. There is also an upper bound around 500Hz. The recordings where acquired with 8 active channels, which were all available channels on the device. No remarkable changes were found with fewer active channels.

Set SR	Mean achieved SR	Mean deviation, [%]	Std. deviation	Max deviation, [%]
50	50.00	-0.01	0.02	-2.00
100	100.0	-0.02	0.06	-4.23
200	199.9	-0.03	0.16	-5.69
300	299.8	-0.05	0.90	-27.68
400	399.8	-0.06	2.57	-58.59
500	499.5	-0.10	3.91	-64.76
600	504.0	-16.0	22.1	-65.45

Table 1: DAQ samplings rate; based on data sampled over 1 minute. Max deviation is deviation from mean, not the set SR.

4.2.2 Streaming

The same procedure, as in the previous section, was used to find the frame rate (FR) for the ultrasound images, see table 2. This will also corresponds to the

sampling rate of the velocity data that are used for strain and flow rates.

The ring buffers had to be restricted to$\sim$ 1000 elements in size if velocity data was transferred, which includes CF and TVI, and could be doubled with b-mode only. This restriction had impact on the saved data, as the maximum interval was limited to the size of the buffer. A buffer size of 1000 would correspond to $\sim$ 25 seconds of saved data.

Set FR	Mean achieved FR	Mean deviation, [%]	Std. deviation	Max deviation, [%]
10	10.0	0.07	0.35	10.6
20	20.1	0.29	1.17	25.7
30	30.2	0.66	2.87	52.9
40	38.1	-4.80	5.05	34.4
50	39.6	-20.9	6.83	37.9
60	41.1	-31.5	6.65	32.4
70	40.4	-42.2	6.79	29.4

Table 2: Ultrasound frame rate; Max deviation is deviation from mean, not the set SR.

ECG SR on scanner	SR in the application	Mean dev, [%]	Std. dev.	Max dev., [%]
600	600.0	≈ 0	≈ 0	≈ 0

Table 3: Streamed ultrasound ECG. The signal is sampled at the scanner side, which makes the signals seemingly flawless.

4.3 Synchronization

The synchronization set-up was used to find the delay between the unit,. and was found to be:

$$\gamma_{delay} = 0.20s \tag{27}$$

The variation of the delay from the scanner to the application was measured. This was done by recording the difference between the scanner's time stamp on the images and the local time at reception of the frames over 15 minutes. The standard deviation of these differences was found to be:

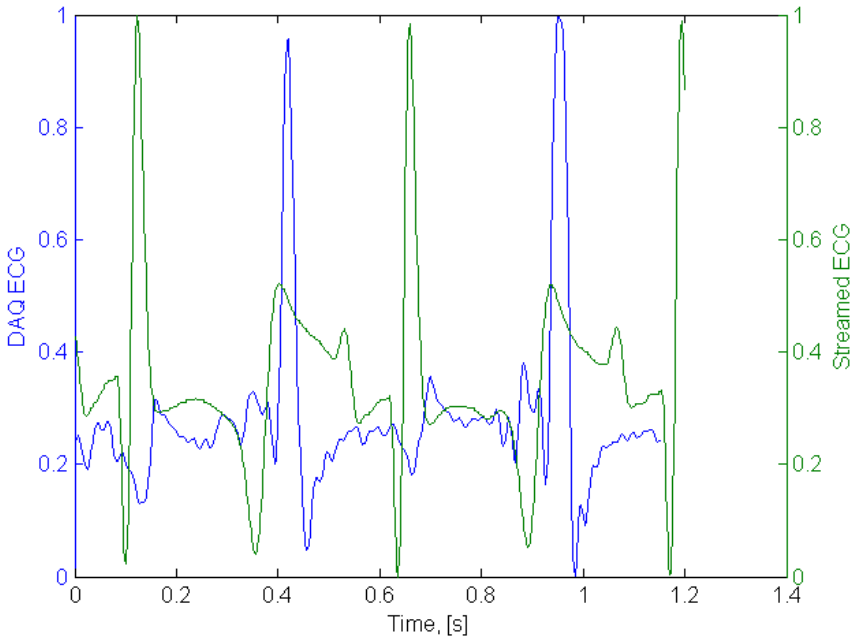
$$\sigma_{delay} = 15.5187ms \tag{28}$$

$$100 * \frac{\sigma_{delay}}{\gamma_{delay}} = 7.5\% \tag{29}$$

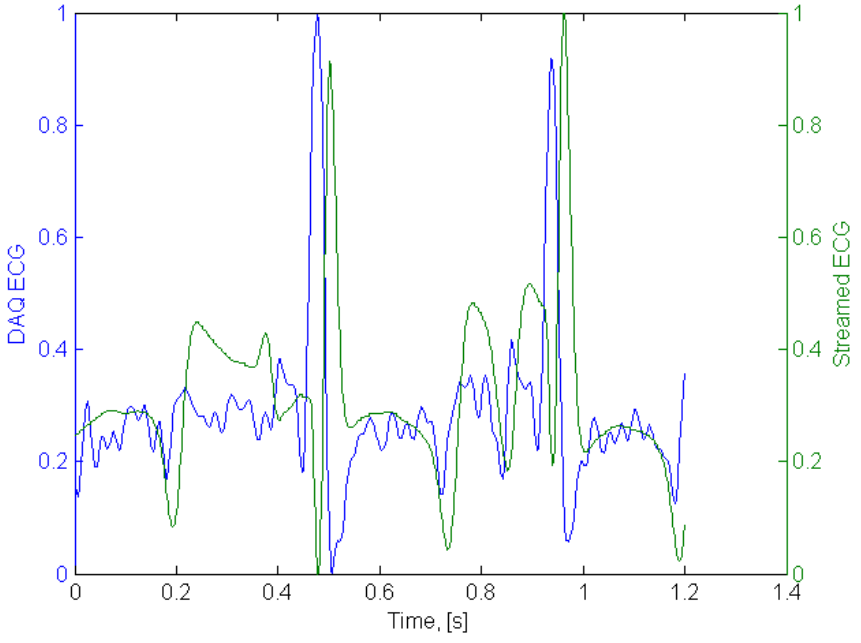
With a frame rate of 40 this deviation will correspond to a deviation of ± 0.62 frames per second.

The ECG signal can be used to validate the synchronization, as both data sources acquires this signal. However, the two ECG signals have different sources, as one is measured with a catheter and the other non-invasively, this will cause an actual time-shift between them, but will still be an indicator of delay between the acquisition units.

Figure 26a and 26b shows recorded data from before and after synchronization is applied. The synchronization functionality aligns the plots, there was however a deviation between them also after synchronization. Figure 27 shows the shift between the signals after synchronization, which measured to be $25ms$.



(a) Before synch



(b) After synch

Figure 26: Synchronization validation, ECG values from the DAQ and the ultrasound scanner, Before and after applied synchronization.

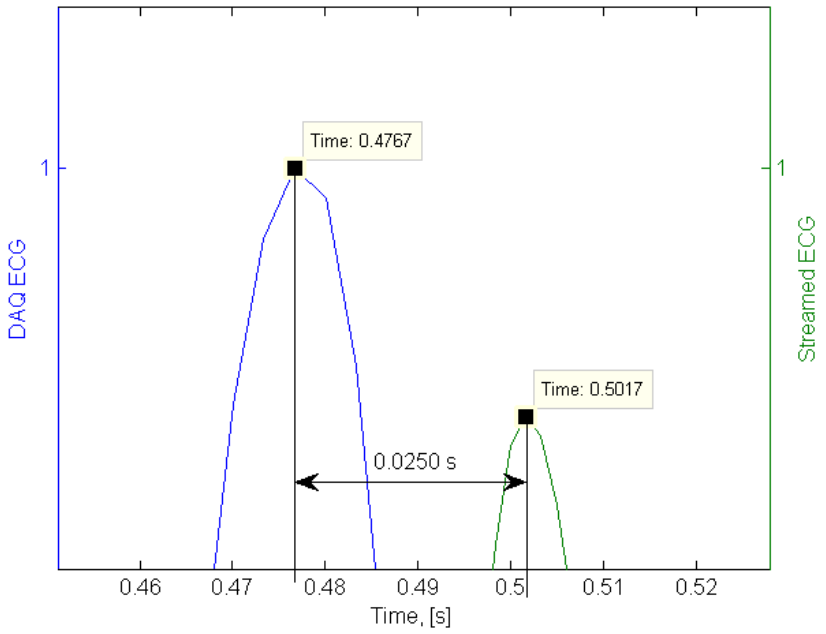


Figure 27: After synch delay

4.4 In vivo results

The range of the output signals from the used instruments were all within the limits of the DAQ, [-10,10] Volts. All the instruments acquired by the DAQ had supporting calibration methods that where compatible with the integrated interface, as described in section 3.1.3.

The data was recorded from the left ventricle of the pigs heart. As the pig had a blood infection, that resulted in sepsis, the values were abnormal.

Volume The volume data from the Leycom was inferred by the power outlet and its harmonics. This was solved with applying the low pass filter with order 50 and a cut-off frequency set to 30Hz. The result can be seen in figure 28.

The relatively high heart rate, which can be seen by the period of the signal, is caused by the sepsis.

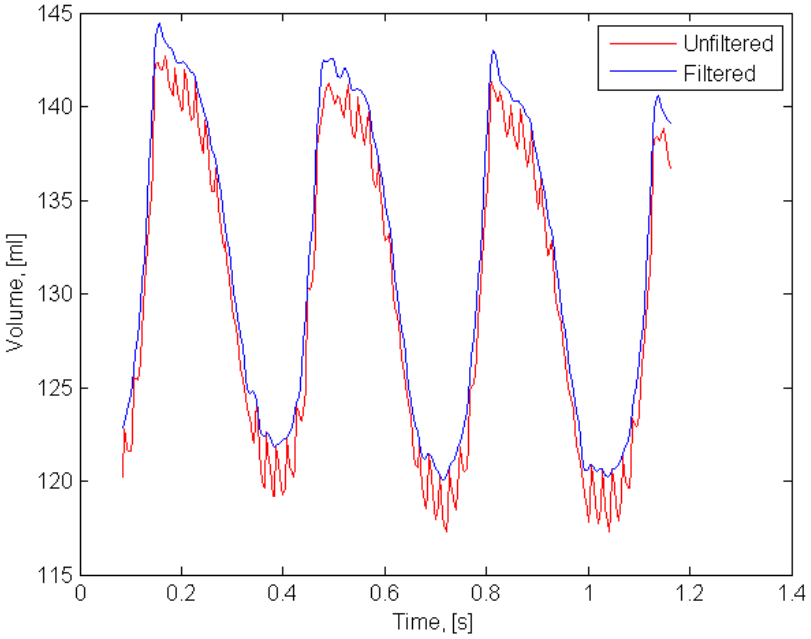


Figure 28: Volume, with and without filtering

Pressure The pressure data supplied by the Sentron Pressure Measurement System had no need for filtering, due to its smooth curve. The curve shows low pressure values due to sepsis.

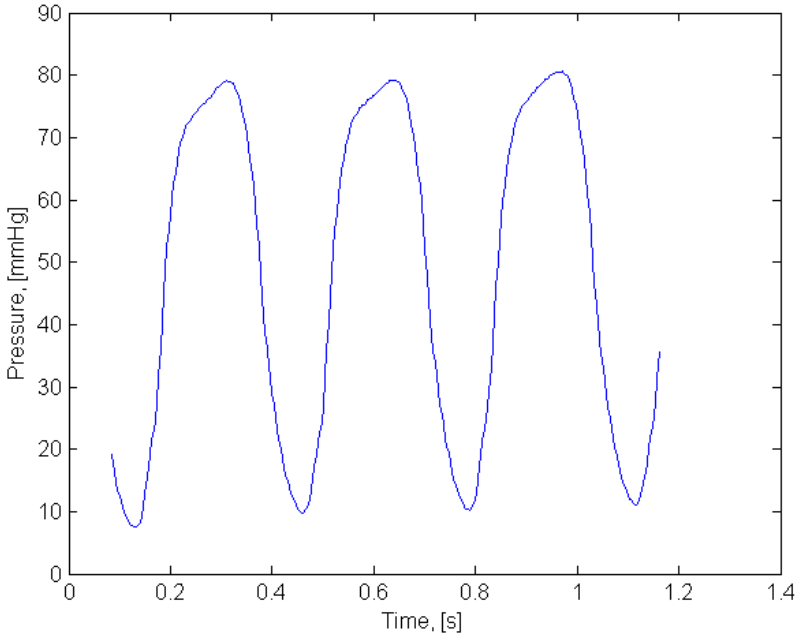
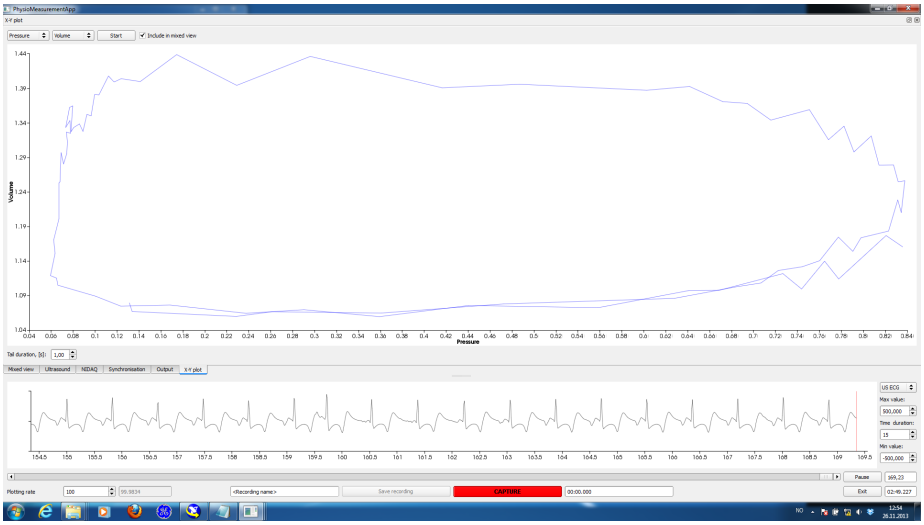


Figure 29: Pressure data. The smooth plot makes it unnecessary for filtering.

PV-loop By choosing the pressure measurement from the Sentron device and the volume measurement from the Leycom unit, the application produced the PV loop in figure 30. The effect of filtering the volume curve can also be seen. Figure 31 shows a resulting PV plot alone.



(a) Before filtering

(b) After filtering

Figure 30: PV loops in the application

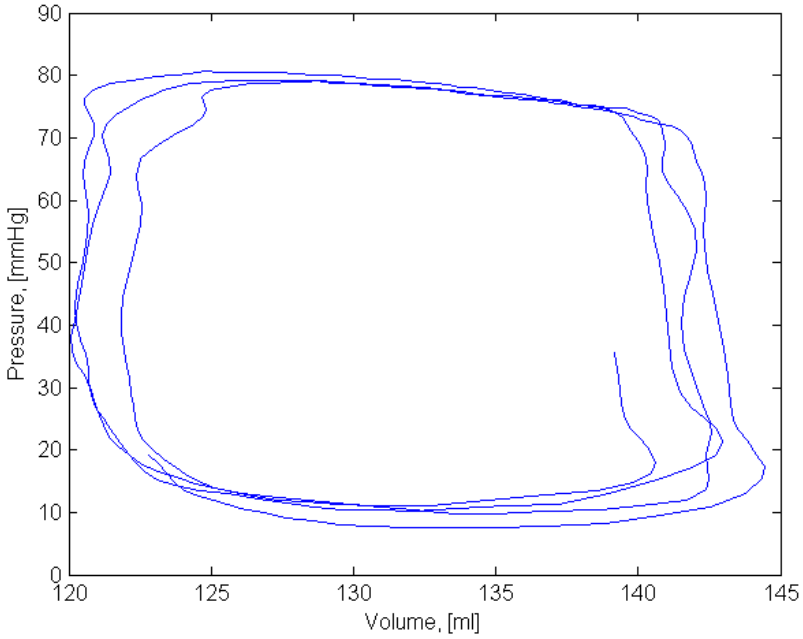
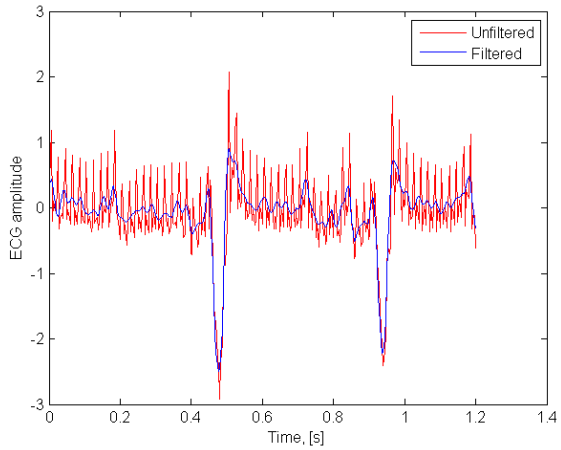


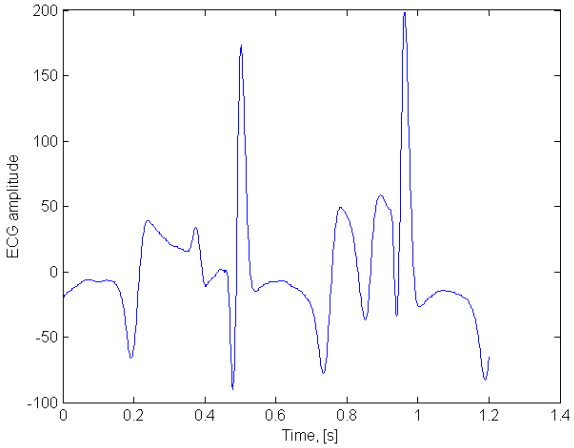
Figure 31: PV-loop plotted.

ECG The two ECG measurements have different sources and acquisition methods, invasive and non-invasive, and had hence some differences in their signals. It is however easy to notice the periodic peak in both signals, which both originate from the r-point in the ECG signal, as seen in figure 3.

The Leycom's ECG signal was could be filtered with the same filter specifications as the volume curve, as their noise sources was the same.



(a) Leycom ECG, with and without filtering



(b) Streamed ECG

Figure 32: ECG comparison. The Leycom ECG has an opposite polarity than the streamed.

Flow rate Figure 33 shows the blood flow from the aorta, where a flow rate trace was calculated from. The diameter was set to cm instead of mm, but this is adjusted for in figure 34. The blood flow had a direction towards the probe and was hence not corrected for.

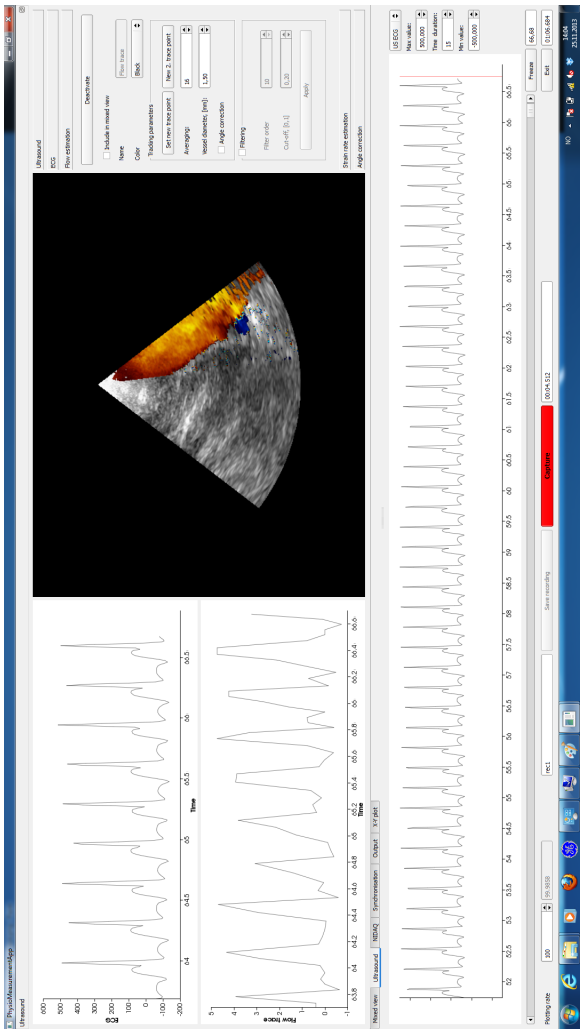


Figure 33: Flow rate trace from the Aorta.

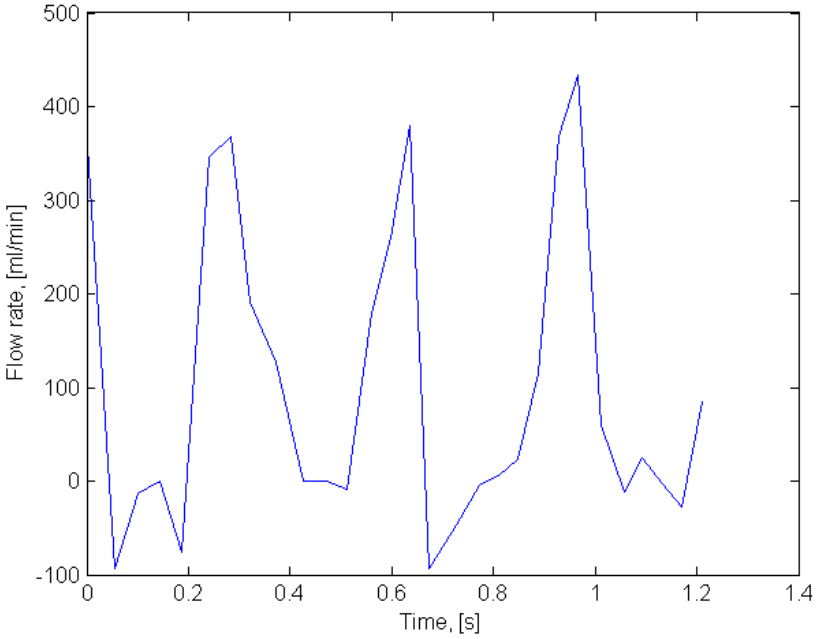


Figure 34: Flow rate plot

Strain The streaming client is not fully developed for TVI transmission. The velocity data is transmitted as it should, but is not correctly represented graphically, as seen in figure 35. The colors are very dominant, compared to figure 8 where the velocity image is semi-transparent. This made it difficult to choose the right trace points.

Two tests were performed with the second experimental set-up, where one of the measurements was a trace of velocity data alone, with no strain calculations. The second was intended to demonstrate the strain rate calculation. The results can be seen in figure 36.

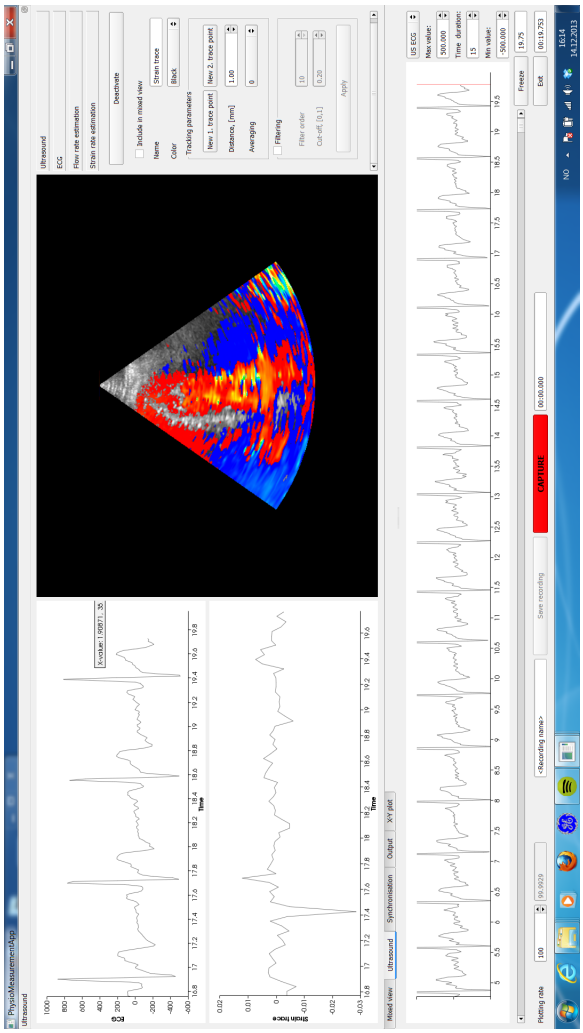
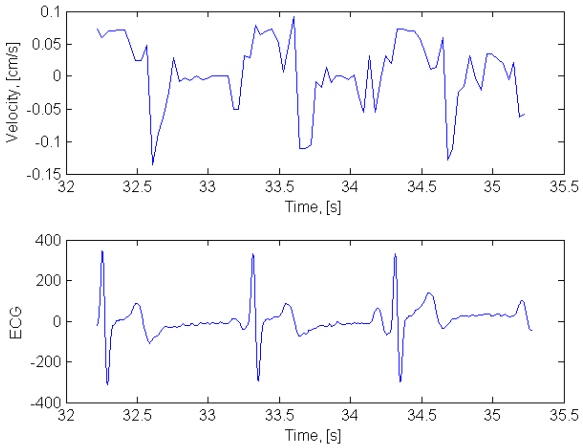
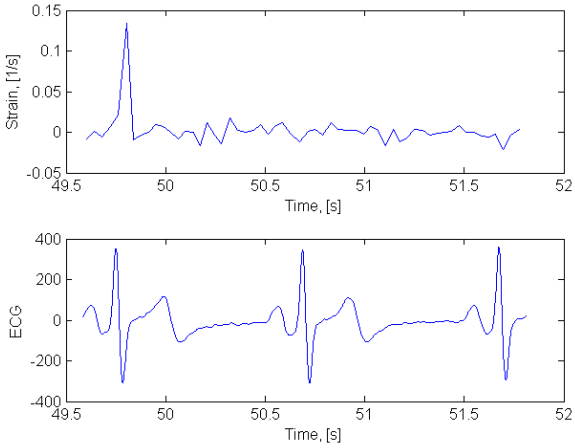


Figure 35: Strain rate screen shot



(a) Velocity plot



(b) Strain plot

Figure 36: Strain rate plot

4.5 Saving of data

Figure 37 shows how a project file is organized. This example project file, named *Stenosis project*, contains two sessions (in red); "Pig - healthy" and "Pig - stenosis stage 1", and a "Project comment" element. The comment is a string that contains a user-defined description of the project.

Each of the sessions contains some meta data and all related settings. By saving the session's settings, a user is able to start a new session based on another session's settings file.

The recordings contain all raw data. This includes the DAQ's traces and streamed ultrasound data and ECG. And as seen in the figure; multiple recordings can be saved in a session; "No load", "pulmonial hypertemsjon1" and "pulmonial hypertemsjon2" in this example.

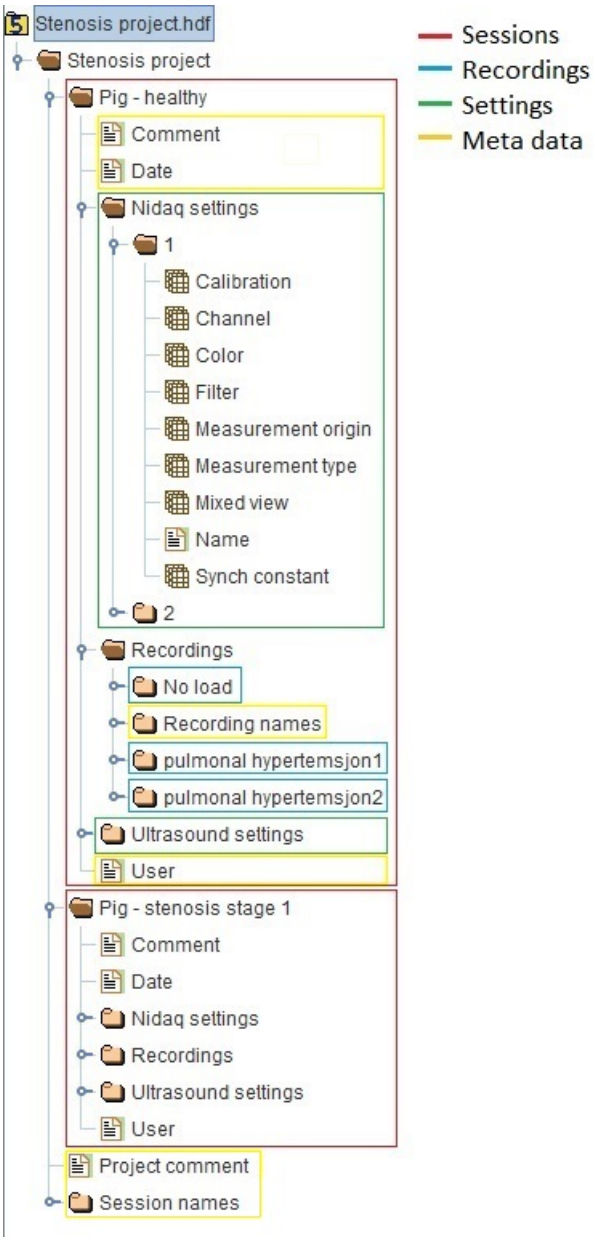


Figure 37: Project file organization, with explaining color codes.

4.6 Importing to Matlab

Scripts were made in Matlab to show capabilities for exportation, and post-acquisition analysis. Figure 38 and 39 shows examples where the project files are loaded into Matlab and a selection of measurements are chosen to be plotted. ECG is used to divide the plots into heart cycles, and the user can choose how many of these to plot. The black vertical line in the ECG plot is the time reference for the viewed image, and can be moved by the user to compare the ultrasound image with the measurements at a given time. A PV plot for the period is also constructed with a marked Ea-line.

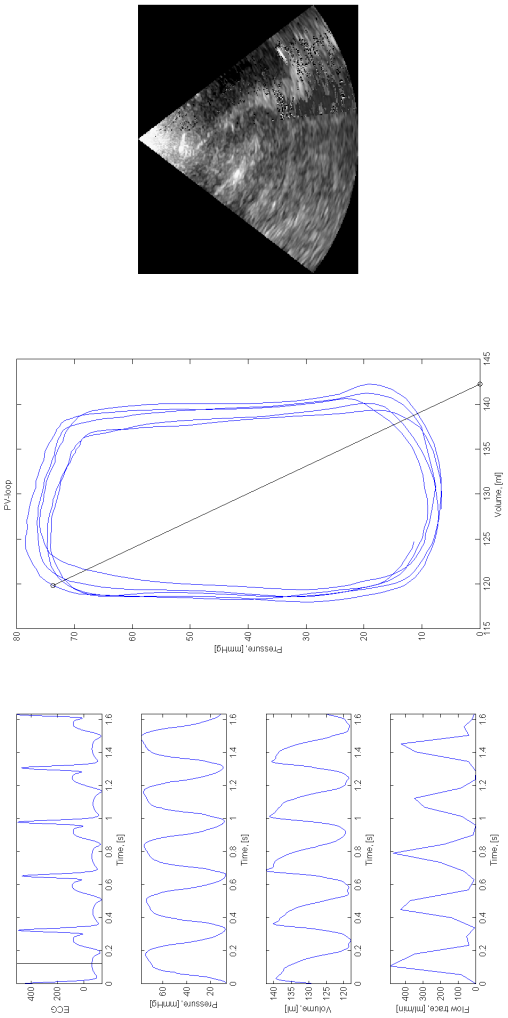
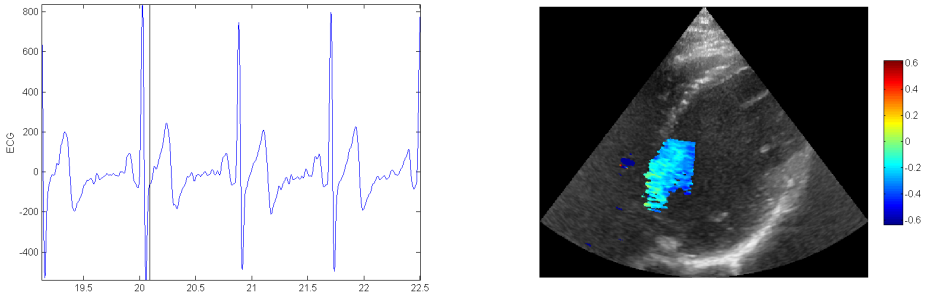
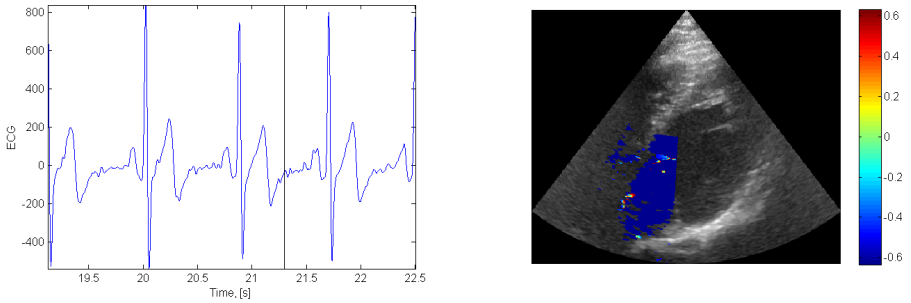


Figure 38: Data viewer in Matlab; Data acquired from pig trials, except the image that is imported from another sequence for illustrative purposes.



(a) Heart image from early systole. The heart volume is around it's maximum.



(b) Heart image from early diastole. Approximately fully contracted.

Figure 39: Data viewer in Matlab; data obtained from the experimental set-up with a human candidate.

5 Discussion

5.1 GUI and usability

The GUI provided the elements that were needed to interact with the functions implemented in the application. QT's variety of GUI elements made it possible to design a well-organized view.

The tabbed windows were used as a basis for the set-up by dedicating one tab for each of the different measurement groups. These measurement tabs concentrated the available settings to those relevant for the respective measurement. These tabs alone did however not support a single view for all the measurements. A "mixed view" tab was there for constructed, in where the user could choose which measurements to include. A section at the bottom of the application window, made easy access to functions that was intended to be used independent on the active tab. This includes recording functionality and the time bar.

There are, however, some GUI features that have room for improvement. For now the testing has been performed close to the laptop screen, while in practice the program will be displayed on an external monitor with some distance to the personnel. The font size should therefore be an adjustable setting for the operator, and perhaps also the thickness of the plot lines.

Another aspect for increased user experience would be to change the way that the graphs are plotted. As it is now, the whole plot moves with time, from right to left. The continuous rendering of this moving plot makes the appearance a bit blurry, and it can be hard to focus on one point in time due to the movement. This could be solved by making the plot start from the left side of the plot and move to the right, when it reaches the end can start over. An additional flaw with the plots is that they are not rendered synchronously. This causes the different plot's time-axis to be slightly shifted related to each other. The reason for this is to reduce the amount of time a thread is holding a on to a resource.

Interaction with the ultrasound images could have a closer integration with the GUI. It does not provide any marker points for the trace points in the image. This made it difficult to remember the exact places from where the data was acquired from. The different length needed for the traces could also have been automatized to make the acquisition less user dependent.

The testing has for now been restricted to the developer, and not by end users of the application. This reduces the objectivity, and the a-priory knowledge of the program by the developer will shadow some aspects of the program's usability. The described evaluation will therefore be limited to some degree.

5.2 Data acquisition

The sampling rates achieved was found to be higher than the minimum sampling frequency for physiological data on humans, $\sim 250Hz$ for ECG traces. The highest achieved sampling rate was $499.5 \pm 3.91Hz$. However, the standard

deviation increases with the sampling rate. Requirements for uniform sampling will be dependent on the use, and can have effect on for filtering. However, At this rate it will not have any affect for the naked eye, and filtering of the signals acquired during these experimental studies gave good results.

The filtering implementation worked as intended. It was easy to update the filter characteristics, which made it convenient to find the right filter order and cut-off for the different signals during acquisition. There is, however, a flaw with the filtering functionality. When the filtering is applied there will be a delay on the signal, as described in section 3.1.2. There is no compensation for this delay in the viewed data, which will cause them to be out of synch with the other measurements. However, as the saved measurements contain the raw data, they are not corrupted by this.

5.3 Ultrasound streaming

The application has shown to be compatible with three of the supported ultrasound modalities provided by the streaming client and the scanner; b-mode, CF and TVI. Due to the limitations in the maximum velocity achieved with both TVI and CF, pulsed Doppler is a much used modality that lacks support in the application. This could have been useful as the application is designed for heart imaging, with high velocity content.

Due to motion of the heart there is need for high frame rates. The traces acquired from these frames add additional requirements for sufficient rate. The limit from the streaming client is 60 frames per second(fps). This is not met in the application, which has an upper limit below 40 fps. At this rate there are additional deviations, which causes a non-uniform rate. Sampling of these data will result in inaccurate filtering.

Rendering and acquisition of frame data are parallel threads that are both time consuming. As these threads shares a buffer, one insert and the other retrieves, there will be queues to prevent ambiguities from the buffer. This was the bottle neck in the data chain, and effectively decreases the frame rate. Due to the size of the frame data, the maximum frames that could be saved were limited to around 25 seconds, which is probably too short time for practical use. This could however be solved by instead of saving data after a recording is finished, dumping it to disk during recording.

Velocity data was successfully acquired from both the CF and the TVI modalities, but at a rate limited by the streamed ultrasound frame rate. The flow data showed an obvious pattern and repeating shape with the same rate as the other data. The actual values could, however, not be validated as no direct flow measurements were available. But the same approach was used in earlier version of the same software[25], where results showed correlation between the accurate flow rate and the approximated.

TVI was used to measure strain rate from the heart wall. Two types of tests were performed, where one showed a successful trace of a velocity trace only, where the other attempted to measure actual strain rate. The latter turned out to be a noisy, without any recognizable repeated patterns as it should have

been. This can probably be explained due to the static trace point placed on the image frame, rather than a tracking trace on the heart. It was also difficult to make accurate trace points as the color coding covered the tissue image. In addition no visible markers where set on the image, which made it hard to place the second trace point at an accurate position related to the first.

5.4 Synchronization

Synchronization between the ultrasound scanner and the DAQ was developed to be a simple and useful implementation. The DAQ's output functionality made the synchronization set-up to an uncomplicated affair. With the synchronization cable connected to the scanner, only two buttons had to be pushed in the application to find, and apply, the delay constant between the to devices. Conveniently this constant was saved to the project file, available for later session.

The synchronization tab provided a view that let the user get an impression of the delay between the units, both before and after applied synchronization. This delay was treated as constant through the session, but the variance was found to be 7.5% of the calculated delay. This variability could have a observable impact when comparing data. This could, however, possibly be solved by continuously calculating the time difference between local application time and local scanner time, and use this change to adjust the delay between the units.

The synchronization technique depends on no or a negligible delay through the instruments that are connected to the DAQ. This has, however, not been properly evaluated, due to lack of methods for testing. But the ECG signals that are acquired from both systems has been compared, and gives an indication of successful synchronization, but not a definite answer as the two signals are acquired from different places in the body. The assumption has been proved to be adequate enough for similar applications, as described in Ole Omejer's Master thesis[24].

5.5 In vivo results

The first time a set-up is used; there is need for some initialization of synchronization, calibration and changing of names with correct units. As this can take some time, it was very convenient with the ability to load a full set-up for the consecutive trials.

It was valuable to see the application working under realistic environments. This gave understanding on how data acquired from the devices where linked together, compared to a testing environment where independent test data has been used. The data acquired from the test was also valuable to show exportability to Matlab, where a simple analysis program was made. It showed that obtained results could be linked back to theory. It could however have been beneficial with additional data, as recordings under different loading conditions on the heart, to further show how obtained data could be supported by theory, for example the ESPVR and EDPVR lines.

Validation of the obtained trace data from the ultrasound frames was difficult, as no reference measurement was available at sight. However, the flow shape was shown to follow the pulse. But further investigation of these measurements is needed for validation.

The importance of a stable program was a valuable lesson. The application is still a beta version, and does suffer from some bugs that occasionally caused the program to crash. This is events that degrades the user experience, and will not be acceptable in a finished product, as important data can be lost.

5.6 Project files

The file system developed was a successful integration in the application. The possibility to load settings from earlier sessions made the start-up process a quick affair.

As the files can be opened with an HDF viewer, the user can after acquired data go in and for example what calibration constants the measurements was based on, or the synchronization delay. The HDF viewer provides copy functionality, so saved data samples can be copied into any program. Integration with Matlab has also be shown to be efficient. The code used for this can be found in appendix D.

Final testing of the functionality remains to be performed by the end user of the application, but the development has been a result of input from users of Ole Omejer's similar application which is currently in use by the same end user as the application described here.

6 Conclusions

The development of a flexible framework for acquisition of data from both an ultrasound scanner and other measurements, through an acquisition unit, has made the main objectives of this thesis achieved.

Real-time acquisition of data from these sources - with signal processing possibilities as calibration, filtering, synchronization, data extraction from ultrasound images and combination of different signals - has given the program functionality that is not yet provided by any other available software.

Ultrasound frame rate and extraction of strain measurements does, however, not provide sufficient performance at this stage.

The developed GUI is one of the applications biggest strength with its flexibility and usability for a variety of applications. The possibility that comes with the constructed file system makes it easy to save a used set-up as well as acquired data. The representation of saved data has also been shown to be easy accessible from other general programs, as Matlab.

The framework has a robust platform that allows for expansions for future work. More ultrasound modalities, as pulsed Doppler, and compatibility for other acquisition units could enhance the possibilities further.

References

[1] *Hemodynamics*. Williams and Wilkins, 1989.

[2] The circulatory system, 2013.

[3] B. Angelsen. *Ultrasound Imaging. Waves, Signals, and Signal Processing*. 2000.

[4] M. Co. *Heart, amyloidosis, M-mode*. 2013.

[5] D. C. e. a. Dugdale. Cardiac catheterization, 2012.

[6] General Electric co. *Technical Publications - Vivid E9*, 2008.

[7] S. A. Gilani. Common carotid artery, 2010.

[8] Hockhdrucklabor. Echocardiography, 2012.

[9] T. H. Institute. Cross-section view, 2013.

[10] R. E. Klabunde. Cardiovascular physiology concepts, 2011.

[11] R. E. Klabunde. Pressure volume cycle, 2012.

[12] W. N. e. McDicken. Colour doppler velocity imaging of the myocardium. 1992.

[13] National Instruments. *NI 625x Specifications*.

[14] Nema. How medical imaging has transformed health care in the u.s., 2006.

[15] A. Olsson. *Ekokardiografi*. TrycksakSpecialisten, 2010.

[16] m. i. Patrick J. Lynch. transesophageal echocardiography ultrasound diagram, 2006.

[17] QT. Widgets and layouts, 2013.

[18] P. R. e. a. Rijnbeek. Minimum bandwidth requirements for recording of pediatric electrocardiograms. 2001.

[19] K. K. Shung. The principle of multidimensional arrays. 2002.

[20] I. H. Software. Parks mcclellan algorithm, 2013.

[21] A. Støylen. Strain rate imaging. 2001.

[22] J. H. H. System. Cardiac catherization, 2013.

[23] I. R. Thomson. Basic transesophageal, 2013.

[24] O. Øvergaard Omejer. A system for the acquisition and analysis of incvasive and non-invasive measurements used to quantify cardiovascular performance. Master's thesis, NTNU, 2011.

[25] M. S. Wigen. A centralized tool for live acquisition and analysis of ultrasound and other physiological data. 2013.

[26] Wikipedia. 2013.

[27] V. B. Wyller. *Det friske og det syke mennesket*. Akribe, 2005.

[28] C. Yunus and J. Cimbala. Flow in pipes. In *Fluid Mechanics*, chapter 8. McGrawHill, 2004.

7 Appendix

A GUI elements

Here are som examples of different ways to organize the application with multiple screens.

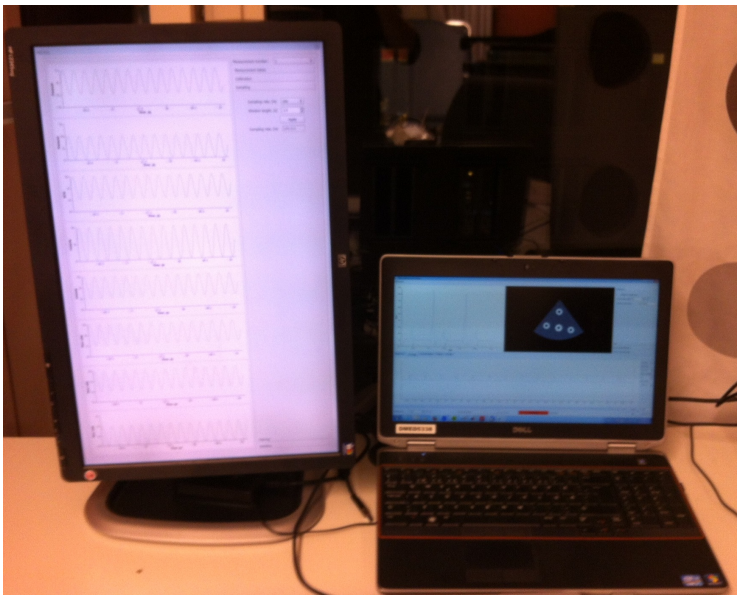


Figure 40: Two screens 1; the NIDAQ tab is dragged an external display.

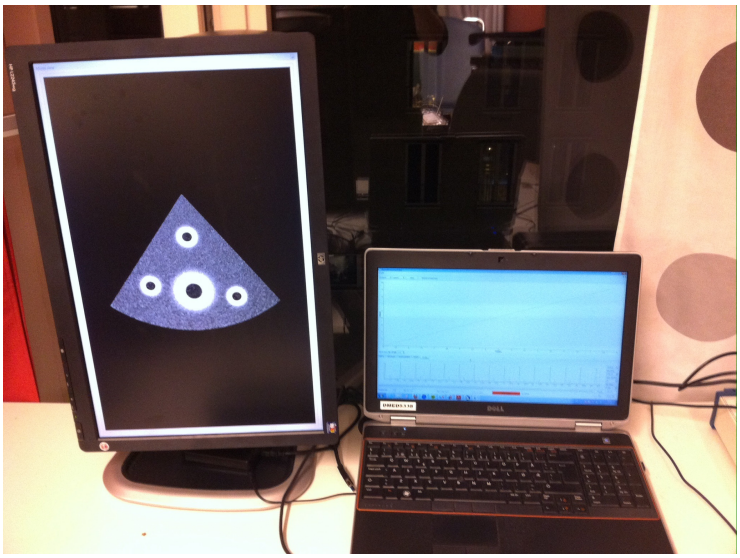


Figure 41: Two screens 2; mixed view, with only the ultraound image represented, dragged to an external display.

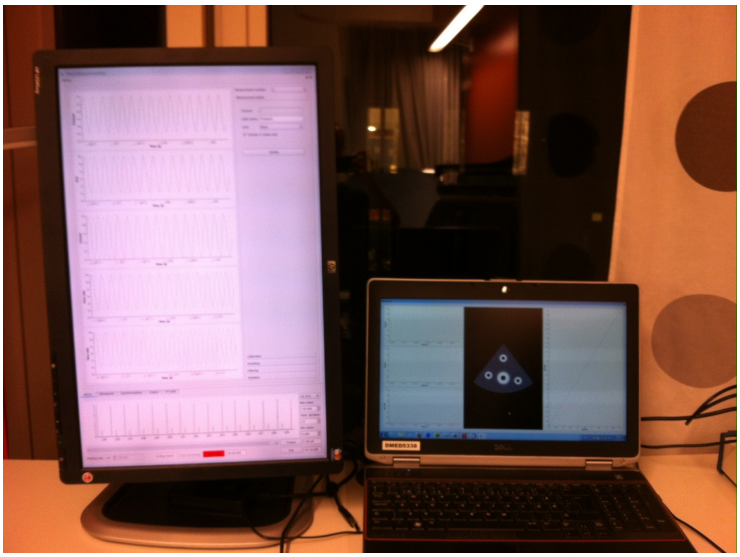


Figure 42: Two screens 3; another combination of mixed view on laptop with ultrasound three DAQ plots and PV-plot, and the DAQ-tab on the external display.

B Flow derivation

Flow rate can be described by equation 30.

$$\dot{Q} = \frac{dV}{dt} \tag{30}$$

This can also be written as:

$$Q = v_{avg} * A \tag{31}$$

By assuming laminar flow in a circular pipe the following approximation can be applied:

$$v_{avg} \approx \frac{v_{max}}{2} \tag{32}$$

And the area will be:

$$A = \frac{\pi * D^2}{4} \tag{33}$$

Which combined gives the approximation for flow in a vain:

$$\widetilde{Q} = \frac{\pi v_{max} r^2}{2} \tag{34}$$

C C++ code

C.1 Data retrieval from data buffer

```
1  double getBufferValueFromTimeReference
2      (
3      int measurementIndex,
4      double timeReference,
5      iterator& dataIt,
6      iterator& timeIt,
7      bool filteringOn
8      )
9  {
10 //Index for measurement buffer
11     int index = measurementIndex;
12
13     //Initialization of time and data iterators
14     iterator dataIterator;
15     iterator timeIterator;
16 // start at beginning of buffer
17     if(lastTimeStampIndex==0)
18     {
19         timeIterator =
20             timeBuffer[index].begin();
21         dataIterator =
22             dataBuffer[index].begin();
23     }
24 //precausion only
25     else if(
26         timeBuffer[index].begin() + lastTimeStampIndex >=
27             timeBuffer[index].end()
28         )
29     {
30         timeIterator =
31             timeBuffer[index].begin();
32         dataIterator =
33             dataBuffer[index].begin();
34     }
35 //if timeReference is smaller than previously used time
       stamp-> start at beginning
36     else if(
37         *(timeBuffer[index].begin()+lastTimeStampIndex-1)
38         > timeReference
39         )
40     {
41         timeIterator =
```

```
42              timeBuffer[index].begin();
43          dataIterator =
44              dataBuffer[index].begin();
45      }
46  //If timeReference is bigger that prev. used time stamp->
        start search from prev index
47      else
48      {
49          timeIterator =
50              timeBuffer[index].begin()+lastTimeStampIndex;
51          dataIterator =
52              dataBuffer[index].begin()+lastTimeStampIndex;
53      }
54
55
56  //Starts the actual search for the closest timestamp in
        relation to the timeReference.
57      for(
58          timeIterator;
59          timeIterator<timeBuffer[index].end()-1;
60          timeIterator++
61          )
62      {
63          double value1 =
64              abs(*timeIterator - timeReference);
65          double value2 =
66              abs(*(timeIterator+1) - timeReference);
67          if(value1<value2)
68          {
69              break;
70          }
71          dataIterator++;
72      }
73  //Updates the index for next time the function is called
74      lastTimeStampIndex =
75          std::distance(timeBuffer[index].begin(),
                timeIterator);
76
77  //Updates input iterators
78      timeIt=timeIterator;
79      dataIt=dataIterator;
80  //returns unfiltered if not activated.
81      if(!filteringOn)
82          return *dataIt;
83      else
84      {
```

```
85 //If filtering is activated-> return a filtered sample
86         return std::inner_product(firCoeff[index].begin()
               , firCoeff[index].end(), dataIterator-firCoeff
               [index].size(), 0.0);
87         else
88 //If buffer is smaller than number coeficients-> dont
       filter
89             return *dataIt;
90     }
91 }
```

C.2 Insert data to image ring buffer

```
1  //Add scan converted image frame to ring buffer
2  addCurrentData(vtkSmartPointer<vtkImageData>
       scanConverted, double timeStamp)
3  {
4  //Index of the current ring buffer insertion element
5      currentScanConvertFrameIndex=(
           currentScanConvertFrameIndex+1)%bufferSize;
6      index=currentScanConvertFrameIndex;
7
8  //If the image buffer is not yet filled up, the current
       buffer size is updated
9      if(index>maxSCRingBufferIndex)
10         {
11             maxSCRingBufferIndex = index;
12         }
13
14 //Adds the data to the buffer
15     scanConvertedImages[index]=vtkSmartPointer<
           vtkImageData>::New();     scanConvertedImages[index
           ]->DeepCopy(scanConverted);
16 //Adds the timestamp to the data
17     scanConvertedTimeStampVector[index]=timeStamp;
18
19 }
```

C.3 Data retrieval from image buffer

```
1  double updateNextScanConvertedFrame(double timeReference)
       {
2  //Starts the search for the closest time at the previous
       frame
3      int prevSCIndex = previousScanConvertedFrameIndex;
4
```

```
5  //If timeReference is smaller than the previous time
       reference->
6  //start search at beginning of ring buffer
7      if(timeReference < scanConvertedTimes[prevSCIndex])
8      {
9          if(maxSCRingBufferIndex<bufferSize)//Before
               buffer is filled up
10             startIndex=0;
11         else //After buffer is filled
12             startIndex=currentScanConvertFrameIndex+1;
13     }
14     else //if not search starts from previous time
           reference index
15         startIndex=prevSCIndex;
16
17     if(scanConvertedTimes.size()>1)
18     {
19 //This will be the new index
20         int index;
21         for(index=startIndex; index!=
               currentScanConvertFrameIndex; index=(index+1)%
               bufferSize)
22         {
23             //Find the closest time
24             if(abs((scanConvertedTimes[index] -
                   timeReference))
25                 <=abs((scanConvertedTimes[(index+1)%
                       bufferSize] - timeReference)))
26                 {
27                     break;
28                 }
29         }
30
31
32
33         echoWidget->
34             UpdatePlaneTexture(scanConvertedImages[index
                   ]);
35 //current index saved for next frame searc
36         prevSCIndex=index;
37         //returns the time of the updated framw
38         return scanConvertedTimes[index];
39     }
40     else
41         return -1;
42
```

```
43 }
```

C.4 Control of sampling rate

```
1  void getNewSample(double &sample, double &time)
2  {
3  //Sleeps the thread until the right time has gone between
4  //previous sample was retrieved and the current time
5      while((getCurrentTimeSample()-previousTimeStamp) < 1/
           sampRate)
6          {
7              Sleep(0);
8          }
9
10     //Calculate current sampling rate
11     currentFrameRate=1/(getCurrentTimeSample() -
           previousTimeStamp);
12
13     previousTimeStamp = getCurrentTimeSample();
14
15 //Sample and the current time stamp values are given to
       the input variables.
16     sample = getLatestSample();
17     time = getCurrentTimeSample();
18 }
```

D Matlab code

D.1 Load session

```
1  function [ currentSession ] = getSession( fileName )
2  %GETSESSION Opens the file, basd in filename(project)
3  file = H5F.open(fileName);
4  rootGroup = H5G.open(file, fileName(1:end-4));
5  sessionNames = H5G.open(rootGroup, 'Session names');
6  info = H5G.get_info(sessionNames);
7  numberOfSessions = info.nlinks;
8
9  %Lists available sessions
10 fprintf('Sessions:\n');
11 for i=1:numberOfSessions
12     sessionString = H5D.read(H5D.open(sessionNames,
           num2str(i)));
13     fprintf('%i: %s\n', i, sessionString);
14 end
15
16 %Asks the user to select session from project
17 pressed = input('Enter session number', 's');
18
19 %Opens session to return from function
20 currentSession = H5G.open(rootGroup,H5D.read(H5D.open(
       sessionNames, num2str(pressed))));
21
22 end
```

D.2 Load recording

```
1  function [ currentRecording ] = getRecording(
       currentSession )
2  %GETRECORDING Opens a recording from a session
3
4  recordingsGroup = H5G.open(currentSession, 'Recordings');
5  recordingNames = H5G.open(recordingsGroup, 'Recording
       names');
6  info = H5G.get_info(recordingNames);
7  numberOfRecordings = info.nlinks;
8
9  %Lists available recordings
10 fprintf('Recordings:\n');
11 for i=1:numberOfRecordings
12     recordingString = H5D.read(H5D.open(recordingNames,
           num2str(i)));
```

```
13      fprintf('%i: %s\n', i, recordingString);
14  end
15
16  %Asks the user to select recording from session
17  recordingNumber = input('\nEnter recording number', 's');
18
19  %Opens recording to return
20  currentRecording =
21      H5G.open(recordingsGroup,H5D.read(H5D.open(
            recordingNames, num2str(recordingNumber))));
22
23
24  end
```

D.3 Load measurement

```
1   function [ samples, timestamps, name ] = getMeasurement(
        currentSession, recording )
2   %GETMEASUREMENT Takes session and recording as input and
        returns a sample set, with its %timestamps and its
        name
3
4   %Opens settings group
5   nidaqGroup = H5G.open(currentSession, 'Nidaq settings');
6   ultrasoundGroup = H5G.open(currentSession, 'Ultrasound
        settings');
7   info = H5G.get_info(nidaqGroup);
8   numberOfNidaqSettings = info.nlinks;
9
10  %Lists all available measuremetns based on settings
11  fprintf('Measurements:\n');
12  numberOfMeasurements=numberOfNidaqSettings;
13  for i=1:numberOfNidaqSettings
14      recordingGroup = H5G.open(nidaqGroup, num2str(i));
15      recordingString =  H5D.read(H5D.open(recordingGroup,
            'Name'));
16      fprintf('%i: %s\n', i, recordingString);
17  end
18
19  streamingOn =  H5D.read(H5D.open(ultrasoundGroup, 'Active
        ultrasound streaming'));
20  if(streamingOn)
21      fprintf('%i: %s\n', numberOfNidaqSettings+1, '
            Ultrasound ECG');
22      fprintf('%i: %s\n', numberOfNidaqSettings+2, 'Flow
            trace(if any)');
```

```
23      fprintf('%i: %s\n', numberOfNidaqSettings+3, 'Strain
            trace(if any)');
24      numberOfMeasurements=numberOfNidaqSettings+3;
25  end
26  measurementNumber=-1;
27  %Asks the user for a measurement based on the list
28  while((measurementNumber<1)||(measurementNumber>
        numberOfMeasurements))
29      measurementNumber = input('Enter measurement number')
            ;
30  end
31
32  measurementNumberIndex = num2str(measurementNumber-1);
33  measurementNumber = num2str(measurementNumber);
34
35  %Finds the chosen measurement from the recordings
36  if(str2num(measurementNumber)<=numberOfNidaqSettings)
37      nidaqTraces = H5G.open(recording, 'Nidaq traces');
38      samplesDataset = H5D.open(nidaqTraces, ['Measurement
            samples ' measurementNumberIndex]);
39      timeStampDataset = H5D.open(nidaqTraces, ['
            Measurement times ' measurementNumberIndex]);
40      nameGroup=H5G.open(nidaqGroup, measurementNumber);
41      nameDataset=H5D.open(nameGroup, 'Name');
42      name=H5D.read(nameDataset);
43      samples = H5D.read(samplesDataset);
44      timestamps = H5D.read(timeStampDataset);
45
46  elseif(str2num(measurementNumber)==(numberOfNidaqSettings
        +1))
47      usECG = H5G.open(recording, 'Other traces');
48      samplesDataset = H5D.open(usECG, 'ECG data samples');
49      timeStampDataset = H5D.open(usECG, 'ECG time samples
            ');
50      name='us ECG';
51      name=name';
52      samples = H5D.read(samplesDataset);
53      timestamps = H5D.read(timeStampDataset);
54      %timestamps=timestamps-0.19;
55
56  elseif(str2num(measurementNumber)==(numberOfNidaqSettings
        +2))
57      flowTrace = H5G.open(recording, 'Other traces');
58      samplesDataset = H5D.open(flowTrace, 'Flow trace data
            samples');
59      timeStampDataset = H5D.open(flowTrace, 'Flow trace
```

```
            time samples');
60          name='Flow trace, [ml/min]';
61          name=name';
62          samples = H5D.read(samplesDataset);
63          %samples=samples*100;
64          timestamps = H5D.read(timeStampDataset);
65          %timestamps=timestamps-0.19;
66
67      elseif(str2num(measurementNumber)==(numberOfNidaqSettings
            +3))
68          strainTrace = H5G.open(recording, 'Other traces');
69          samplesDataset = H5D.open(strainTrace, 'Strain trace
                data samples');
70          timeStampDataset = H5D.open(strainTrace, 'Strain
                trace time samples');
71          name='Strain trace';
72          name=name';
73          samples = H5D.read(samplesDataset);
74          timestamps = H5D.read(timeStampDataset);
75          %timestamps=timestamps-0.19;
76      end
77
78      end
```

E Connections of instruments

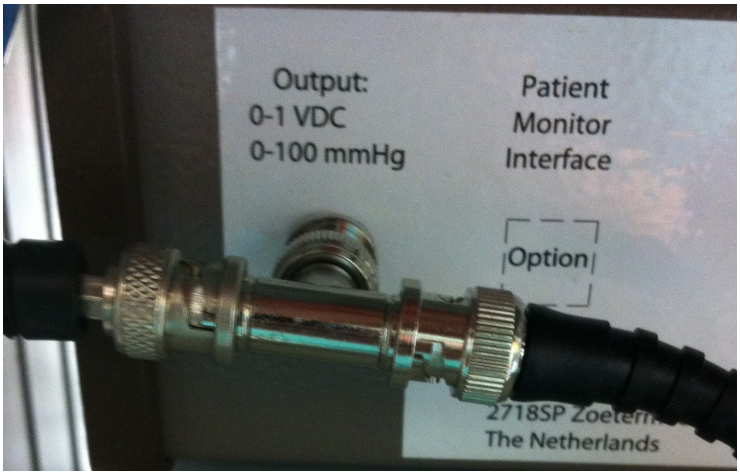


Figure 43: Sentron pressure interface. A T-coupling is used as the pressure signal was used both by the DAQ and in the Leycom Sigma 5DF.

Figure 44: Leycom Sigma 5DF. Takes pressure as input to show PV-plots on another screen. Volume is generated by the unit and is available through an output channel. ECG is also available.

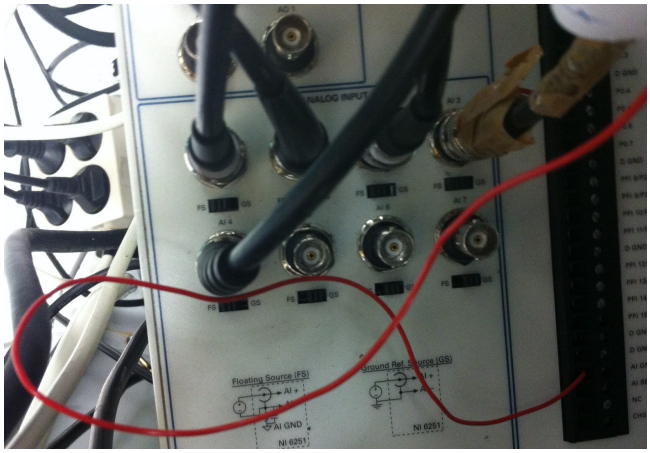


Figure 45: NI USB-6251; Connections to the DAQ. Channel 0: Volume, channel 1: Flow rate(not used), channel 2: ECG, channel 3: Pressure from Millar catheter(not used), channel 4: Pressure from Leycom