MSc Project: Oscillometric Blood Pressure

Belinda Kneubühler 2504756K

School of Engineering
College of Science and Engineering
University of Glasgow

Submitted in fulfilment of the requirements for the
Degree of MSc Computer Systems Engineering

Supervisor:	Bernd Porr
Second Supervisor:	Kiran Ramesh

August 2020

Abstract

Abstract text goes here.

Keywords: C++, real-time, blood pressure, oscillometric, mean arterial pressure,

Contents

List of Tables

List of Figures

Acknowledgements

Acknowledgements text goes here.

Declaration

Chapter 1

Introduction

1.1 Problem Description

Traditional blood pressure measurements require an inflatable cuff that is fixed tightly around the upper arm of a patient. Pressure in the cuff is increased above the value of the expected blood pressure and slowly released while listening to the heart sounds on a stethoscope to determine blood pressure. This process is performed by a medical professional. In recent years, automatic tools have eliminated the dependency on professionals and made taking one's blood pressure at home possible and convenient.

Most automatic devices use small changes in pressure during deflation to detect blood pressure. This is the oscillometric method. It has been around for more than 40 years but has not improved much since its beginnings. Its limitations are known, and many proposed algorithms have tried to address them since. Unfortunately, without notable success. Besides, commercially available devices use proprietary, closed-source algorithms that can not be scientifically challenged.

1.2 Objectives

This project aims to examine existing and proposed algorithms for oscillometric blood pressure measurement. They will be judged based on robustness, underlying theory and applicability in an automatic real-time system.

The most promising identified algorithm is implemented using the provided hardware on a Linux system. The hardware consists of a simple, manual blood pressure cuff equipped with a pressure sensor that is connected to an analogue to digital converter (ADC). This is connected to a Linux computer. The implementation is done in C++ as an open-source project that is available on GitHub. The software is designed so the algorithm implementation can easily be adopted in other projects.

The developed application guides the user through taking a measurement and reports the results on the screen. Meanwhile, raw pressure data is recorded and stored in a file. Ideally, the project would have been concluded by performing a small set of experiments and building a database of measurements. Unfortunately, this was not possible in the current environment and the short time frame.

1.3 Outline

This report is split up into three main parts, the theory, discussed in chapter 2 Background, the implementation of the developed application, discussed in chapter 3 Algorithm, 4 Hardware and 5 Software and the discussion of the results in chapter 6.

The first part, chapter 2 Background, discusses the theory of oscillometric blood pressure measurements. It focusses on research material and algorithms suitable for implementation in real-time processing. It starts by explaining what blood pressure is and how it is can be accurately measured. Next, different algorithms for automatic oscillometric measurements are explained starting with the most popular ones and ending with proposed alternative methods. The chapter is concluded with a summary and the identified algorithm to implement in this project.

The second part, chapter 3 Algorithm discusses the implemented algorithm in detail. It is a description of the logic and principles upon which the software is built on.

Chapter 4 Hardware defines briefly the used hardware, followed by chapter, 5 Software. It discusses the implementation of the application that was developed. It focusses on the implemented architecture and structure of the application. The complete source code of the software is available on GitHub and documented using Doxygen.

The next part, chapter 6 Results and Discussion, highlights what approaches worked well to determine blood pressure from oscillometric data and what caused difficulties.

Finally, the last chapter, **??** **??** looks back at what was achieved in the project. Recommendations are given on further steps, how to develop the software and improve the algorithm.

Chapter 2

Background

This chapter discusses the findings of researching existing and proposed algorithms for determining systolic and diastolic blood pressure through the oscillometric method. This report focusses on relatively simple algorithms that can be implemented in a real-time application, which is the goal of this project.

The first part of this chapter defines what blood pressure is and how it is measured manually. Afterwards, different algorithms are explained and examined.

A brief section at the end of this chapter explores algorithms beyond the scope of this project that have been proposed in recent literature.

2.1 What is Blood Pressure?

Blood pressure is an important bio-medical measurement and often used for diagnostics in cardiovascular diseases. Most commonly, systolic and diastolic blood pressure are mentioned in pairs, but what do those numbers mean?

The heart is a pump that has two phases. When the ventricle is relaxed, the heart fills up and the diastolic blood pressure (DBP) is observed. When the heart contracts, it pushes the blood through the arterial system and the systolic blood pressure (SBP) is observed. A third characteristic is the difference between SBP and DBP, the pulse pressure (PP). Finally, the mean arterial pressure (MAP) is defined as the average pressure in the artery. Figure 2.1 shows how these pressure values are connected in one heartbeat. MAP is the area underneath the blue curve divided by the time of one pulse, indicated by the orange area. **?** Because the blood pressure is simplified as a sine wave the MAP is in the middle between SDP and DBP, usually the MAP is closer to DBP.

Blood pressure can be measured with invasive and non-invasive methods. The most accurate methods are invasive, but require professional expertise because a catheter has to be injected into the blood vessel. Subsequently, non-invasive methods are more common because they are more convenient

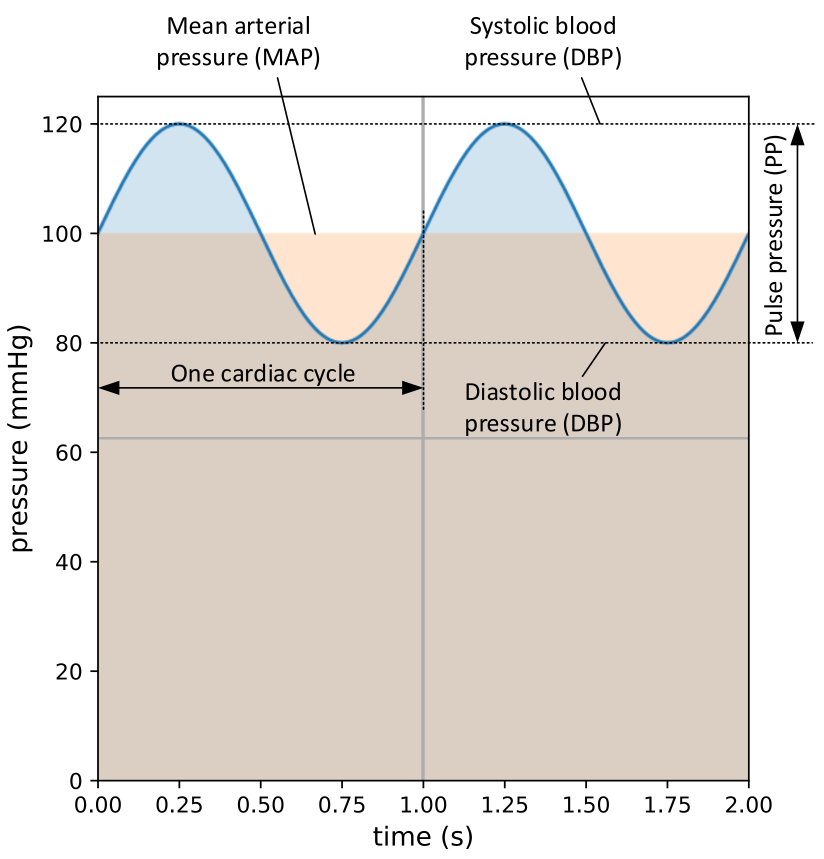

Figure 2.1: Characteristic blood pressure values, simplified as a sine wave. The values are: MAP: 100 mmHg, SBP: 120 mmHg, DBP: 80 mmHg and heart rate: 60 beats/s.

to use. With automated methods, taking blood pressure at home has become convenient and easy. However, the inaccuracies of these measurements are often ignored or even unknown.

2.2 Manual Blood Pressure Measurement

The manual method of measuring blood pressure relies on a sphygmomanometer and listening to the heart or Korotkoff sounds (named after their discoverer) with a stethoscope. This is called the *auscultatory method*. *Palpation* is another method where the pulse is felt at the wrist. It only allows detection of the SBP. The auscultatory method is described below because it provides recommendations that are true for all blood pressure measurements using a cuff.

The sphygmomanometer is shown in figure 2.2. It consists of an inflatable cuff that is connected to a rubber pump and a scale that indicates the pressure in the cuff in mmHg. The pump is equipped with a valve that can be opened to release air.

Millimetres of mercury (mmHg) is a unit to describe pressure. 1 mmHg is defined as the pressure generated by a column of mercury of 1 mm height. Millimetres of Mercury is not part of the International System of Units (SI) but was defined by it before 2019 as 1 mmHg = 133.322 Pa using standard gravity.**?** Since 2019, mmHg is no longer included in the SI brochure. However, it is still widely used in the medical field. **?**

Figure 2.2: Picture of a sphygmomanometer showing the cuff, inflation pump and mercury meter.

Intra-arteria or direct BP measurement is considered the 'gold standard', but most studies compare the developed algorithm with the auscultatory method, which underestimated SBP and overestimates DBP. **?** Nonetheless, the protocols to test automatic blood pressure monitors given by the British Hypertension Society (BHS) and their American counterpart, the Association for the Advancement of Medical Instrumentation (AAMI), both use the auscultatory method as a reference. **???**

2.2.1 Auscultatory Method

The cuff size must be appropriate for the patient's arm. The rubber bladder inside the cuff should cover more than 80 % but less than 100 % of the arm's circumference.

The cuff is completely deflated and tightly fit around the upper arm of the patient. The centre of the bladder should be over the brachial artery and the arm supported at heart level. The stethoscope is to be placed over the artery between the cuff and the patient's elbow and should not touch the cuff. This could influence the reading because it puts additional pressure on the artery and can interfere with hearing the sounds.**??**

The valve at the pump is completely closed and the cuff is inflated to a pressure of about 30 mmHg above the expected systolic pressure. This causes the artery to flatten and not let any blood through. Next, the valve is slightly opened to slowly deflate the cuff at a rate of approximately 3 mmHg/s. The artery will open and let small amounts of blood through. Systolic pressure is read when a pulse is first heard. Meanwhile, deflation continues. Diastolic pressure is read when the sounds disappear completely. Ultimately, blood can flow freely through the artery. After making sure that no further sounds can be heard by deflating for at least another 10 mmHg, the valve is opened completely to rapidly empty the cuff.(**??**)

There are some discrepancies in literature ton how to define the point of the diastolic blood pressure. Some references define it as when the Korotkoff sounds disappear completely. **??** Others, predominantly older resources, define it as the point where the sounds are muffled.**?** Here, the former is assumed, because most current literature does and some literature suggests that the auscultatory method overestimates diastolic blood pressure. **?**

2.2.2 Calculation of MAP

Traditionally, MAP is calculated as an estimate by adding a third of the PP to the DBP. Or, how it is more commonly referred to, adding SBP and twice the DBP and dividing the result by 3 (equation 2.1).

According to calculations by Bos *et al.*, this generally underestimates the MAP and they suggest to add 40% rather than a third of DBP. Furthermore, they suggest to using MAP measured by oscillometric devices rather than calculating it from values obtained through the auscultatory method. **?**

$$MAP = \frac{SBP - DBP}{3} + DBP = \frac{SBP + 2 \times DBP}{3} \tag{2.1}$$

According to Joe **?**, nurses in intensive care units (ICU) today still use the manual calculation method on the automatically obtained SBP and DBP rather than rusting the measured MAP, which introduces large errors.

2.3 Automatic Blood Pressure Measurement

Most modern automatic devices use the oscillometric method to determine blood pressure. They are used both at home and in professional environments.

Automatic blood pressure monitors measure pressure in a cuff similar to the one used in the manual method and uses the small pressure changes (oscillations) extracted from the deflation curve to estimate blood pressure. Commercially available devices are usually equipped with an electric pump and the bladder contains a pressure sensor. Otherwise, they look similar to the cuff used for manual measurements. Some devices use wrist cuffs, but those are rarely recommended. **?** The most significant problem using wrist measurements is the influence of gravity on blood pressure and the accompanying risk of systematic errors when the measurement is not taken at heart level. **?** This problem is avoided by taking blood pressure at the upper arm.

General Procedure The procedure is also similar to the manual method. The pressure in the cuff is increased to a level above the expected systolic pressure, which leads to the artery being pushed close. While releasing the pressure slowly through the valve, blood starts flowing through the artery, resulting in small increases in pressure relative to the continued deflation of the cuff. As the cuff deflates further, these relative changes increase to a maximal value before they start decreasing again.**???** The shape, magnitude or envelope of these oscillations are used in various automatic blood pressure algorithms. Methods that do not use oscillations exist, but are not discussed here.

Oscillation Extraction There are two ways the oscillometric waveform (OMW) is extracted from the deflating pressure curve. The first one is filtering. A bandpass filter with a lower cut-off frequency between 0.1 to 0.5 Hz and an upper cut-off frequency between 5 to 20 Hz is recommended. **?** Implementations usually use first **?** to sixth order **?** Butterworth filters. They are known for their flat pass-band response and good frequency response.

The second approach is to use de-trending. It requires to know the beginning of each pulse to be able to fit a line of continuously deflating pressure to the identified points. Its advantage is, that it additionally reproduces an estimated deflation curve. However, it requires additional data, for example, ECG for pulse detection.**?**

Figure 2.3: General procedure explained with example data. TODO

Ultimately, from a deflation curve as in the simulated signal in figure 2.3 on the top, the oscillations are extracted. The bottom plot in figure 2.3 shows oscillations extracted with a thrid order Butterworth bandpass filter with the cut-off frequencies at 0.5 Hz and 5 Hz. **?**

Envelope The envelope of the OMW is used by basic automatic algorithms. Formation of the envelope is achieved in different ways. The simplest way is to register only local maxima. Similar to that, and most common, is the subtraction of the following through from a local peak or interpolating the curve of local maxima and local minima to subtract them from each other. Some algorithms additionally fit a curve on the obtained oscillometric waveform envelope (OMWE) in an attempt to remove artefacts. **?**

2.3.1 Maximum Amplitude Algorithm

The maximum amplitude algorithm (MAA) is based on the assumption that the oscillations are maximal when the pressure in the cuff equals arterial pressure. Accordingly, the recorded pressure at which oscillations are maximal is considered a valid estimate of the MAP (**????**), as long as the compression chamber is kept small **?**. To avoid introducing errors, the cuff should always be tightly fit, because air volume in the cuff causes maximum oscillations to be above the true MAP. Hence, Ursio and Cristalli suggest to use the lowest pressure of the plateau of maximal oscillations as the value to assess MAP.**?**

A recent publication by Chandrasekhar *et al.* **?** employs a more complex model than the one originally introduced by Mauck *et al.***?**. They conclude that the MAA results in a weighted average of systolic and diastolic BP. According to their model, MAA underestimates MAP for higher blood pressure.

Fixed-Ratio Algorithm The fixed-ratio algorithm builds on the MAA and is based on the assumption that the systolic and diastolic blood pressure occur at specific fractions of the maximum oscillations before and after its occurrence. Determining SBP and DBP after applying the MAA was tested by **?**. They recorded Korotkoff sounds while measuring oscillations to find a ratio of oscillation amplitudes for SBP and DBP. They defined the ratio for the systolic pressure to be 0.5 and the ratio for the diastolic pulse as 0.8. These ratios were found empirically and Geddes acknowledges that the systolic pressure is overestimated and the ratio for the diastolic pulse is not constant for a range of different diastolic pressures.

Later studies tried to find accurate ratios, mostly experimentally with the ratio for SBP usually being determined between 0.45 and 0.73 and the ratio for DBP a bit higher between 0.69 and 0.83.**??**

Mathematical models confirmed that a generalised ratio cannot be found. Parameters like the age of the subject and in this regard, the arterial stiffness as well as pulse pressure influence the ratios. **?** A higher PP results in a smaller ratio for SBP and larger ratio for DBP. A stiff arterial wall causes volume changes to happen slower. While this has a small effect on the systolic ratio, the diastolic ratio decreases with the stiffness of the artery.**?**

2.3.2 Derivative Algorithm

The derivative algorithm uses the OMWE and plots it against the deflation pressure. The points of the maximal and minimal slope are determined. The pressure point where the derivative of the OMWE is maximal is assumed to be the diastolic BP and where the derivative is minimal is the systolic BP respectively. **??**

Even though mathematical models have confirmed this method to produce valid estimates of both SBP and DBP without the need for empirically obtained ratios, it is extremely vulnerable to noise and therefore not used in practical applications. **??**

Jazbinsek *et al.* evaluated the fixed-ratio and derivative algorithms. They used a device equipped with ECG and a microphone to validate their results. Various ways to form the envelope were used, including detrending, filtering and even using the low-frequency part of the audio signal of a microphone that recorded Korotkoff sounds. However, to evaluate the fixed-ratio method, they used a commercial device to find average ratios and used the values from the same device as a reference for the evaluation. Their evaluation of the derivative algorithm showed many local extrema as expected. The algorithm was only able to perform by applying additional constraints that bias the measurements significantly. For example, the extrema has to be distanced from the MAP value more than 15 mmHg. **???**

2.3.3 Other Algorithms

Since the above-mentioned algorithms use only the envelope of the oscillations to determine BP, they discard all the information that lies within the shape of a single pulse. Naturally, scientists have tried to find ways to use this information to improve BP algorithms.

There are a variety of different ideas that have been tested in small scale experiments of mostly with less than 30 test subjects. While they all claim to improve BP measurements compared to the fixed-ratio algorithm, they often lack significant evidence and mathematical validation.

Non-Fixed Ratio Sapinski **?** proposed a standard algorithm. This algorithm uses ratios to determine the SBP and DBP from MAP as above but calculates the ratios based on the oscillations. The integral of the maximal oscillation is divided by its time period and gives the amplitude of the pulse wave at systolic pressure. The amplitude at diastolic pressure is defined as the difference between the MAP amplitude and the SBP amplitude. This algorithm is based on theoretical assumptions formed form comparisons to the direct method, e.g. that the added oscillation pulse amplitudes at systolic and diastolic pressures equal the amplitude at mean pressure. Sapinski mentions that the average value of systolic oscillation is 40 % of the MAP amplitude and 60 % for the diastolic oscillation, respectively. This part is regularly quoted in other literature, without mentioning that this is not a proposed ratio. Sapinski concludes, that the proposed algorithm does not fulfil the criteria of the AAMI standard compared to the auscultatory method, but does compared to the invasive method. No further literature has been found that validate or contradict these findings.

Pulse Morphology Specific characteristics of single pulses can be examined to determine blood pressure. Following are four of the most commonly used indices. The definitions are used from Mafi *et al.***?**

- Stiffness index (SI): The height of the subject (h) divided by the time difference between systolic and diastolic peak (ΔT) (equation 2.2) is an indicator for arterial stiffness, but requires to know the person's height.
$$SI = \frac{h}{\Delta T} \tag{2.2}$$
- Augmentation index (AI): The difference of systolic (A_S) and diastolic peak (A_D), divided by the systolic peak, expressed in percentage of pulse pressure (equation 2.3).
$$AI = \frac{A_S - A_D}{A_S} \times 100\% \tag{2.3}$$
- Reflection index (RI): The of systolic (A_S) divided by the diastolic peak (A_D), expressed in percentage (equation 2.4).
$$RI = \frac{A_S}{A_S} \times 100\% \tag{2.4}$$
- $\Delta T/T$ Ratio: The time difference between systolic and diastolic peak (ΔT) is decided by the duration of the pulse (T). Both T and ΔT increase with age.

Mafi *et al.***?** plotted the indices in time and used absolute maxima or minima to determine MAP. There is a significant spike where MAP is expected. Using local maxima and minima next to the found MAP to estimate SBP and DBP has less significance. Problematic with this approach is that the reference values were measured by an Omron device with an unknown implementation of oscillometric blood pressure. Because the device did not display the measured MAP, the MAP reference was calculated using equation 2.1, which is known be wrong using accurate values of SBP and DBP and likely delivers arbitrary values when using values from an automatic device.

Another implementation from Mafi *et al.***?** is taking the '1st' derivative of each pulse and taking the maximal value of that to form another curve that looks similar to the one used in MAA. This curve is then treated equally to the fixed-ratio algorithm to determine MAP, SBP and DBP. The ratios are determined experimentally using reference measurements. Additionally, the MAP is again calculated using the faulty equation 2.1 and the 18 test subjects aged between 24 and 68 are healthy and have no history of cardiovascular disease. The authors argue that the shape is less sensitive to noise than the amplitude and therefore, their approach is more robust than MAA.

Model-Based Algorithms These algorithms use a predicted OMWE and fit it to the observed one to determine the characteristics. While they do consider factors that the standard algorithms discard, such as arterial stiffness and pulse pressure, they are vulnerable to artefacts that are not considered by the model. They could be used to validate algorithms. **?**

Neural Networks The features extracted from above, and more, are often used in neural networks (NN). However, it has been shown, that the indices can reliably only be used to determine the MAP. Moreover, some of the features are highly dependant on external factors. For example, the exact deflation rate influences the duration of the OMWE (proposed by Lee *et al.***?**) and the constraint of 2 to 3 mmHg/s given by Lim *et al.***?**, who implemented a NN using the proposed features, seems hardly enough. Similarly, the used filter by Lim *et al.*, a first-order bandpass Butterworth filter with cut-off frequencies of 0.5 Hz and 5 Hz, is likely to remove valuable information from the pulses.

2.4 Summary

Write summary of the algorithms with their advantages and disadvantages

write about which algorithm will be implemented and why

Chapter 3

Algorithm

This chapter explains the general principles behind the algorithm and separates the logic from the implementation, which will be discussed in a separate chapter, Software.

The realised algorithm is a fixed-ratio algorithm. It analyses the oscillations to find MAP form the maximum amplitude as described in 2.3.1 Maximum Amplitude Algorithm and estimates the position of the systolic and diastolic BP as where the amplitudes are at a specific ratio of the maximum. The systolic BP is evaluated while the oscillation amplitudes increase and the diastolic BP while they decrease, respectively.

The top plot in figure 3.1 shows one data set and how it is divided into different steps for processing. The first part (green) is when the user pumps up the cuff pressure. The second part (blue) is the deflation. Here, the oscillations, shown in the bottom plot, are recorded and analysed. Once these oscillations have increased to a maximum and decreased again to a certain value, the last step (red) is simply the deflation of the cuff. The core part of the algorithm works on the blue part of the data.

3.1 Preprocessing

The data is always filtered. Additionally, preprocessing decides which stage of the dataset is active. In relation to figure 3.1 stage one is green, stage two is blue and stage three is red. Figure 3.3 shows the basic diagram of the data processing. Only the deflating and filtered data is processed by the core algorithm.

3.1.1 Filtering

To remove unwanted noise from the data and to get the oscillations, two filters are used. Figure 3.3 shows the simple cascade of a low-pass (LP) and high-pass (HP) filter, resulting in a bandpass(BP) filtered output signal, the oscillogram. The results of both filter outputs are used in the algorithm.

From the research done in chapter 2, it is known that most algorithms use band pass filters of low orders up to 6th and with cut-off frequencies between 0.1 to 0.5 Hz and 5 to 20 Hz (**?**). The

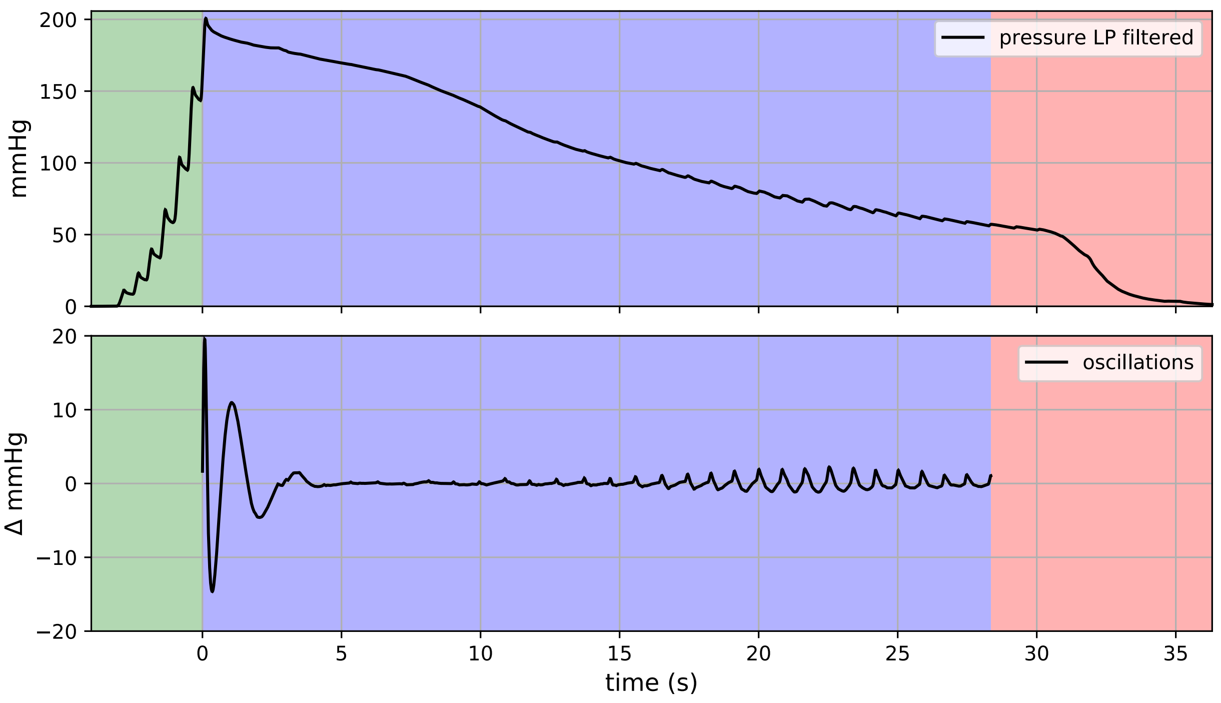

Figure 3.1: A sample data set is shown in the top plot and divided into three parts. The blue part is where oscillations are analysed. These are shown in the bottom plot.

implemented algorithm does not rely on pulse shape and can therefore use relaxed constraints. Best results were achieved with a lower cut-off frequency of 0.5 Hz. The upper cut-off frequency was fixed 10 Hz. Generally, there is very little activity between 5 to 20 Hz, but because no experiments could be made, a mean value was chosen.

The implemented filters are shown in figure 3.2. The low-pass filter has a small delay in the impulse response and the phase response shows that for very low frequencies there is a slightly higher delay. The frequency response shows a steep slope after 10 Hz and suppression of more than 20 dB of every frequency above 20 Hz. The HP filter expectedly lets the impulse pass. In the frequency response there is a small overshoot but otherwise a good suppression of DC. The bottom left plot in figure 3.2 shows that HP filter has a significantly delay of about 125 ms for frequencies around 1 Hz. This could potentially cause inaccuracies. Section 3.3.2 will show, why this is not a problem.

3.1.2 Stage Detection

The LP filtered data is analysed by the preprocessing step to detect if the second stage, the deflation, is active. Figure 3.3 shows how the preprocessing step then passes both the LP and BP filtered data to the core algorithm, symbolised by a switch.

In the first stage, the data is checked if the pressure in the cuff is high enough to start deflation and

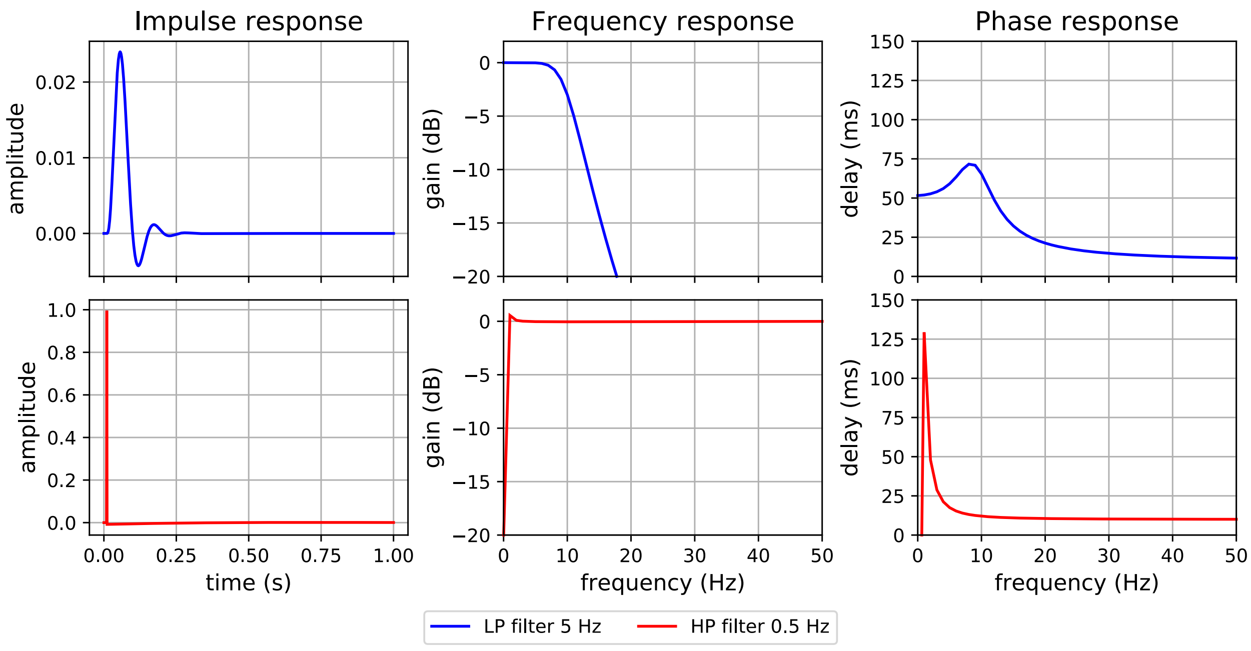

Figure 3.2: Filter.

if so, the preprocessing switches to the second stage. This value is configurable. The default value is 180 mmHg. Figure 3.3 shows how the decision is made based on the LP filtered data.

Once the second stage is active, the preprocessing checks the core algorithm if it is done. In figure 3.3 this is indicated by the arrow from the algorithm back to preprocessing. Additionally, the data is checked if it is below a value of 20 mmHg, where blood pressure measurements make no sense, and the measurement is potentially cancelled.

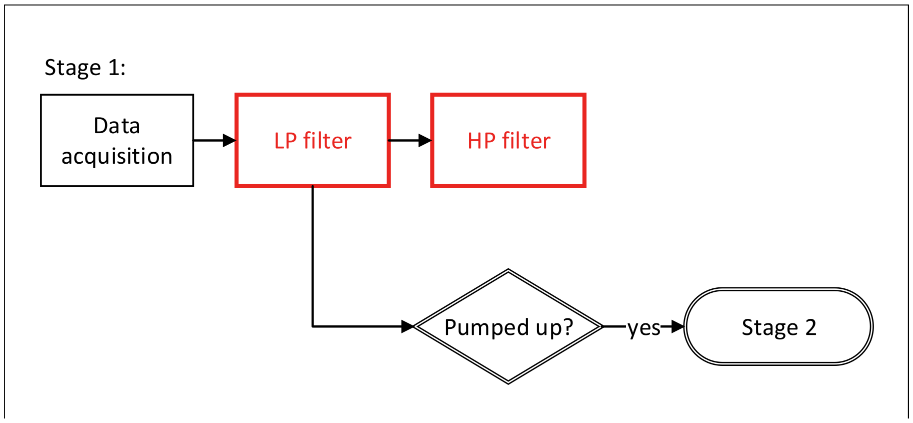

Figure 3.3: The simple basic data processing flow. The data is acquired and passed first to a low-pass filter (10 Hz) and then to a high-pass filter (0.5 Hz), resulting in a bandpass output. The first stage of the data processing only checks, if the pressure in the cuff is high enough to start deflation (stage 2).

3.2 Deflation

Figure 3.4 shows data recorded during the deflation stage. ① shows the low pass filtered data which is used in the core algorithm as a reference. ② is the BP filtered data upon which the algorithm is performed.

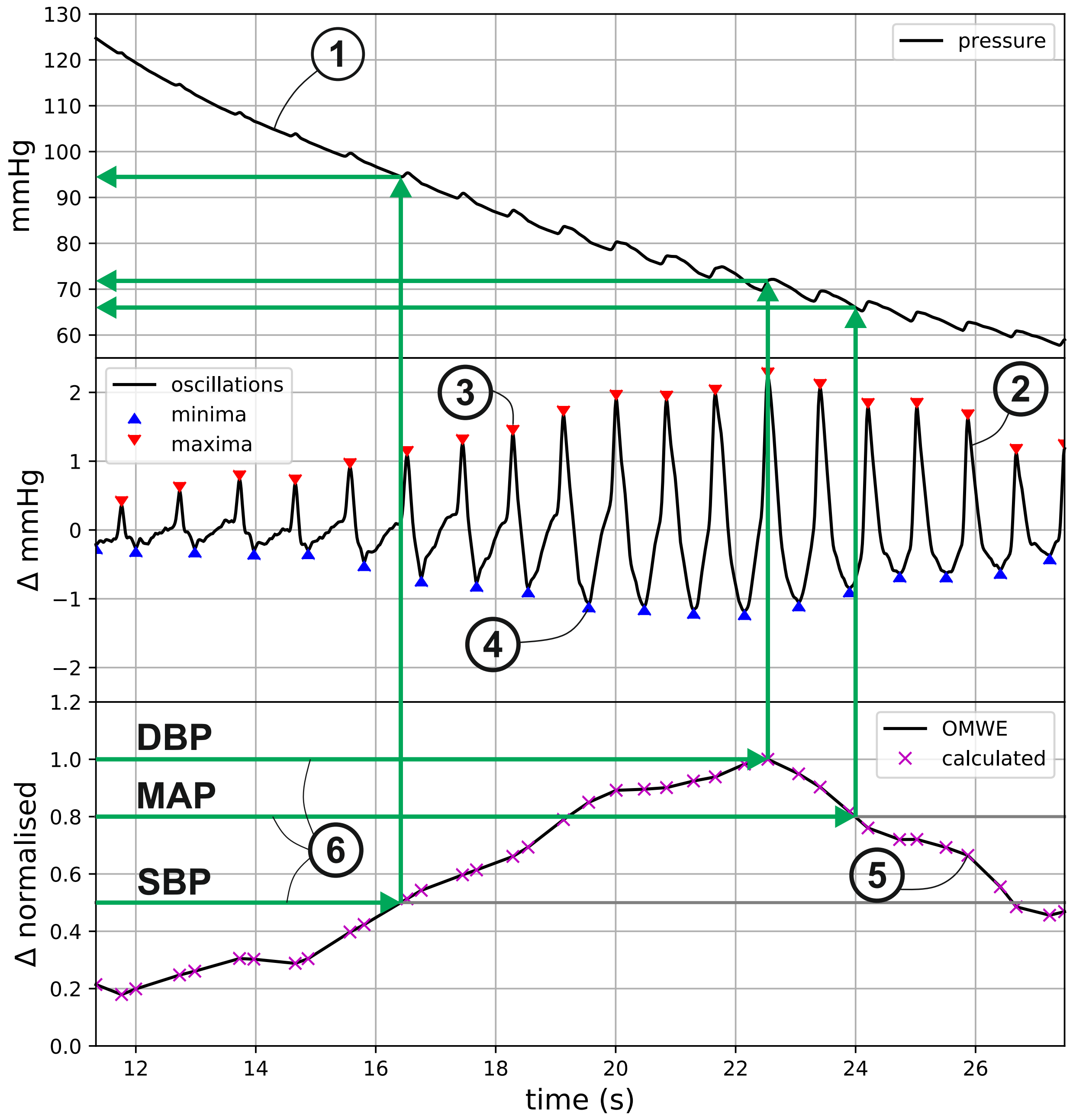

Figure 3.4: Algo explanation.

3.2.1 Extrema Detection

Maxima The next step, indicated with ③ is to detect the maxima. Looking at the oscillations, there are local maxima in the troughs between two oscillations. To avoid detecting these, a value needs to have a minimal pronunciation of 0.25 ΔmmHg to even be considered as a maxima. This limits how the quickly the first oscillations can be detected, but does not influence the algorithm because they are less than 10 % of the expected maximal amplitude.

A sample is identified as a local maxima, due to the previous and following value being lower. Next, is checked if the current and last detected maxima result in a valid heart rate given by a configurable range. The default range is 50 to 100 beats/min. In addition, if the last maxima was less than 300 ms ago, only the larger of the two is kept and the lower is discarded.

This is due to two factors that were observed in the data and are shown if figure **??**. The first, shown on the left hand side, is if the bump in the rising edge of the oscillations results in a detected maxima. This is likely due to artefacts from movement. When oscillations are small and rising, these can lead to wrong detections. This is either irrelevant because they occur before systolic BP or they will be discarded because they lead to errors.

The second issue is shown in figure **??** on the right. It occurs mainly when oscillations are decreasing and if the filtering is not optimal.

Minima Minima detection, indicated by ④ in figure 3.4, is considerably more difficult, especially in the beginning of oscillations as shown in figure **??**. Because there has to be a minima in between two maxima, minima detection is simplified. After detecting two maxima, the lowest value between them is recorded as the minima.

3.2.2 Heart Rate Detection

Heart rate is already part of the extrema detection described above. Therefore, the current heart rate is detected after each valid maxima starting from the second one. Because each pulse stems from a heart beat, each maxima represents one. By calculating the time between the last two maxima, a heart beat is estimated using equation 3.1 below.

$$pulse = \frac{60s}{t_{curMax} - t_{lastMax}} \tag{3.1}$$

After finishing the measurement, the average heart rate is calculated from all validly detected maxima.

3.3 Oscillometric Waveform Envelope

When the detected maxima, marked by the red triangles in 3.4, have reached their highest amplitude and declined again the oscillometric waveform envelope (OMWE) can be calculated.

This condition is estimated by taking the largest recorded maximum and multiplying it with the ratio for the diastolic blood pressure minus a hysteresis of 0.3 (equation 3.2. If the last recorded maxima is below this value, the extrema detection part of the algorithm (③ and ④ in figure 3.4) can be concluded. To avoid a wrong detection of this condition the largest maxima needs to be at least 1.5 δmmHg. Otherwise the condition can be detected when oscillations first start occurring and are small with large variations.

$$A_{cutoff} = A_{Max} \times (r_{DBP} - 0.3) \tag{3.2}$$

3.3.1 OMWE Calculation

To calculate the OMWE from the determined extrema, the following approaches were tested:

1. Only considering the maxima.
2. Subtracting the following minimum from each maxima.
3. Subtracting the preceding minimum from each maxima.
4. Interpolating between maxima and minima to find the corresponding values and subtracting them.

1. works decently, but discards the information in the minima. 2. works well for pressure in the systolic range, because minima follow quickly after maxima. Vice versa, 3. works well in the diastolic range. 4. requires the most computations, but is the least sensitive to noise, because it takes the time difference between minima and maxima into consideration.

When referring back to figure 3.4, ⑤ shows how the OMWE was calculated using the 4thapproach and calculating a value for each of the maxima and minima separately.

Figure **??** shows how interpolation is used to calculate an interpolated maximum at the time where a minimum occurs to then be able to subtract the minimum from the maximum. Inversely, A minimum is interpolated between two measured minima where a maximum occurs. The resulting plot is displayed in figure 3.4 at the bottom. Note that the values are normalised by the maximal value.

3.3.2 Determination of Blood Pressure Values

After calculating the OMWE, the last step is to determine the BP values from it. As explained in chapter 2 Background, MAP occurs where the OMWE has its maximum. The time, when the maximum is recorded, is compared to the pressure at that time as indicated by ⑥ in figure 3.4. To account for the oscillations in the deflation curve, the pressure value is averaged over the period of an average heart beat centred around the detected time. The average heart beat is determined from the measurement as explained in section 3.2.2 Heart Rate Detection. In the example above, this results in MAP of 71 mmHg.

Likewise, to determine SBP and DBP, the specific ratios of the maximal oscillations are searched in the OMWE. The example in figure 3.4 uses a ratio for systolic BP of 0.5 and for diastolic BP of 0.8 for simplicity.

The systolic BP is determined by stepping backwards in time from the maximal oscillations and the diastolic BP by stepping forwards in time, respectively. Once the value falls below the specific ratio, interpolation is used to find the time where the exact ratio would have occurred. Using the same approach as for the MAP, the specific values found with ⑥ for the data in figure 3.4 is 95 mmHg for SBP and 67 mmHg for DBP.

Implications of Filter Delays

As pointed out in section 3.1.1 Filtering, the original data is delayed by first the LP and then the HP filter. Since the oscillogram is based on the LP filtered data, the delay of the first filter (LP) is irrelevant. The second filter (HP) has up to 125 ms delay for certain frequencies. Figure 3.5 shows the oscillations superimposed over the deflating data. At a deflation rate of 3 mmHg/s, the error could be 0.375 mmHg. The calculated pressure is averaged over the period of an average heart rate to account for the oscillations in the deflating data and compared to the inaccuracies of the algorithm in general, this is neglectable.

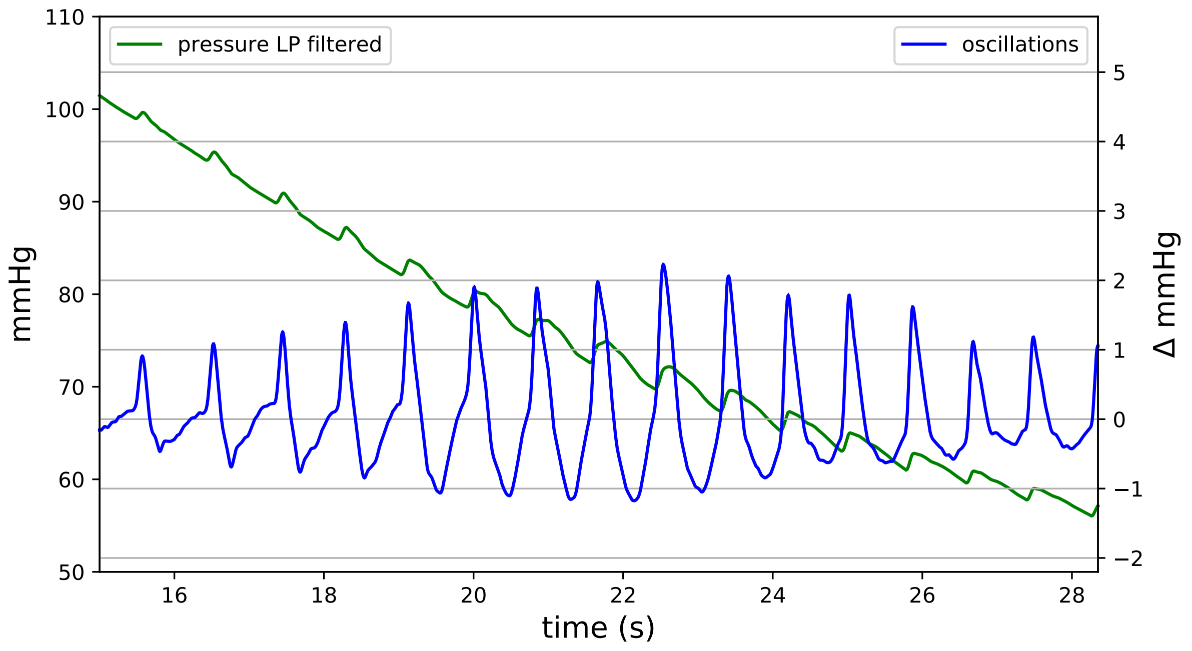

Figure 3.5: Algo explanation.

Chapter 4

Hardware

The hardware used in this project was provided. This chapter describes the basic setup upon which the developed application is built.

Overview

Figure **??** shows a schematic view of the hardware setup. The computer running the application is connected to the data acquisition device, the USB-DUX Sigma, through an USB-port. On channel 0 of the device, the pressure sensor is connected through a voltage divider with a backup capacitor. The voltage divider distributes the voltage roughly 2:1. The software should be calibrated to account for the actual values.

Pressure Sensor

The pressure sensor measures absolute pressure. It operates on 0 V supplied by the USB-DUX Sigma device. It outputs 1 V per 50 kPa measured pressure. The output range is from 0 to 4 V. The characteristic values are summarised in table **??**.

Analogue to Digital Converter

The provided analogue to digital converter (ADC) is part of a USB-DUX Sigma device. The device offers multi-channel in and outputs. However, only a single ADC channel was used. The used channel provides 24-bit resolution at a sampling rate of 1 kHz. The input range of the device is -1.375 to 1.375 V, which is not large enough for the output of the pressure sensor to be directly connected. Therefore, the above mentioned voltage divider is required.

Table **??** summarises the characteristic values of the USB-DUX Sigma device.

Chapter 5

Software

This chapter is a description of the developed software. It provides an overview of the architecture and the implementation. The projects GitHub page provides the full source code, setup instructions and Doxygen documentation (**?**).

The project is licensed under the GNU General Public License v2.0.

5.1 Overview

The application is split into the user interface (UI) and data processing. Each part runs in a thread and they are kept separated from each other. The data processing class is called `Processing` and the user interface class `Window`.

The user interface is built with Qt, an open-source widget toolkit for creating graphical user interfaces. Additionally, the subset Qt Widgets for Technical Applications (Qwt) is used to display the acquired data.

The `Processing` class handles data acquisition and processing, sending notifications to the UI with instructions on what action to perform or what values to update.

5.1.1 Class Diagram

Figure 5.1 shows a simplified class diagram of the application. It shows the Processing and Window classes in the middle with their most important attributes.

The Processing class has a ComediHandler object to acquire data, the OBPDetection object implements the core algorithm described in 3 Algorithm and the Datarecord object saves the raw data to a file. The state of the application is stored in the ProcState enum.

The Window class holds all the UI components. The class diagram in figure 5.1 only shows the high level components that are not directly from the Qt library. SettingsDialog and InfoDialog can be opened through the menu bar to access the configuration and get further information about the application. The two Plot objects are always visible, while the rest of the main window depends on the Screen enum.

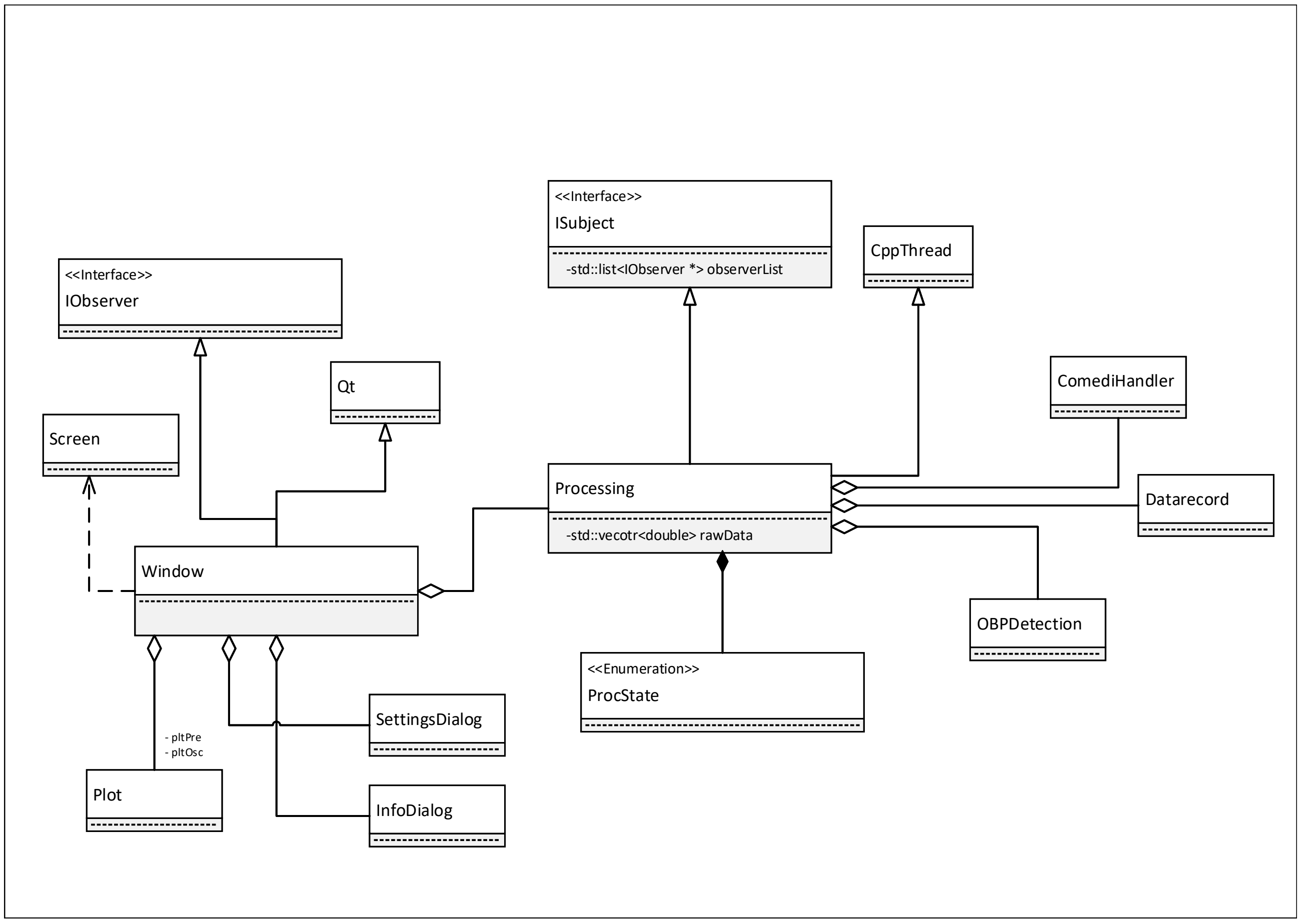

Figure 5.1: The Class Diagram. Needs work.

Communication between the two main objects is done with callbacks in an observer pattern from the `Processing` to the `Window` class. The processing class is an observable subject that the window class can register to because it implements IObserver. The window has a reference to the processing class to pass on user input.

Following is a more in-depth discussion of the above mentioned elements.

5.2 Data Processing

In the processing class, the data is acquired and processed. The acquisition is handled by a Comedi-Handler object, that abstracts handling of the underlying hardware. The data is then filtered and the observers are notified so they can display the new data.

The logic in the processing object is handled through a state machine. The OBPDetection class needs deflating pressure to extract the blood pressure characteristics from the oscillations. The state machine identifies this in the state ’Defalte’. The following section discusses the state machine in detail.

5.2.1 State Machine

Figure 5.2 shows the state machine of the processing class. It is made up of an initial state (Config) that is only entered at start-up, an ’Idle’ state and four states that are consecutively entered in an ideal measurement, called ’Inflate’, ’Deflate’, ’Empty’ and ’Results’.

For every arriving sample, the state machine is executed according to the current state and potentially switched to another one.

In every state, exept ’Config’ the arriving sample is filtered and the results are sent to the observing objects. In the ’Inflate’, ’Deflate’ and ’Empty’ states, the raw data is additionally recorded and ultimately saved as a file when exiting the ’Empty’ state.

Config After a reset, the state machine starts in the ’Config’ state, where the ambient voltage is registered. This is necessary, because the pressure sensor gives an absolute value, but pressure values are needed relative to atmospheric pressure. The raw pressure is read for 250 ms and averaged. If that value has a deviation of less than 1 mV to the maximal value, it is stored as ambient pressure and the state machine processes to the next state.

Idle In the ’Idle’ state, the processing class is waiting for the start of a measurement while filtering data and sending it to its observers. A measurement is externally triggered.

Inflate While the pressure is below the configured pump-up pressure (default of 180 mmHg), in the ’Inflate’ state, the raw data is recorded. When the pump-up pressure is surpassed, the state switches to ’Deflate’. In case the maximum data size for the recorded data is reached, the measurement is cancelled and the state switched to ’Idle’. Reaching the maximum data size takes 5 min and should never happen during a measurement.

Deflate In the ’Deflate’ state, the data is fed into the OBPDetection object, which is described below in section 3.2 Deflation. This object decided when enough data is recorded and the progression to the ’Empty’ state is possible. Alternatively, if the pressure falls below 20 mmHg, or the maximum data size is reached for the recorded data, the measurement is cancelled.

Empty This state is only needed to finish recording data until the pressure reaches zero. When this happens, the data is written to a file and the state is changed to ’Results’.

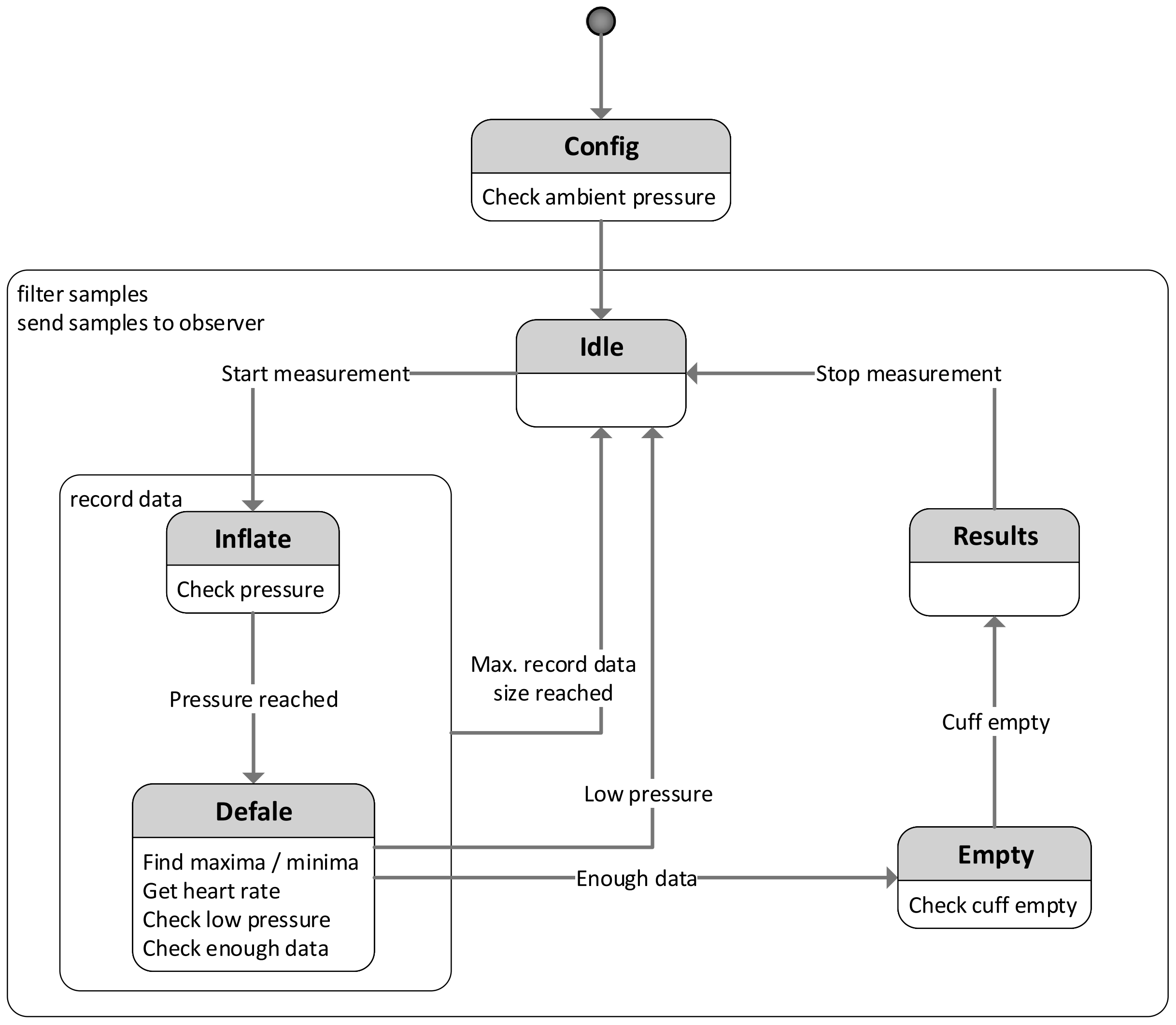

Figure 5.2: The state machine implemented in the Processing class. Needs work.

Results After a successful measurement, in this state, the results are valid. Upon leaving this state, all results are reset and a new measurement can be started. The measurement can be cancelled externally from any of the previous states. If the results state is never reached, the data will not be saved.

5.2.2 Processing Class

The processing class inherits from the CppThread class and the ISubject class. CppThread is a wrapper to the std::thread class that was written by **?** to avoid static methods and makes the inheriting class a runnable thread. Processing has an instance of ComediHandler to acquire and two IIR filter instances to pre-process the data. The raw, unfiltered data is stored in a vector that can be handed to the Datarecord instance to save it as a file. Meanwhile, the filtered data is sent to the GUI to display and handed to the OPDetection instance that performs the algorithm. Data acquisition and filtering are happening whenever the thread is running, the state machine described above decides when data is passed to the OBPDetection or stored to a file.

Thread Safety

The Processing class has public getter and setter methods, that allow objects, with access to it, to read and change configuration variables. These variables are defined as atomic, to make access to them thread-safe. Additionally, the configuration values can only be changed, when the thread is not running. This is a design choice, so a restart is necessary to apply changes and avoid changing the configuration mid-measurement.

Similarly, the booleans set by starting and stopping the measurement and stopping the thread are atomic.

5.2.3 ISubject Class

ISubject is a simple interface defined in a header file that lets the implementing class notify its observers about certain events. Each notification is realised as a protected notify-X method. The public methods are 'attach' and 'detach'. Through these, a class that implements the below described IObserver (section 5.3.4) can be attached to the subject. This stores a reference to the object in a list. When one of the notify methods is called, the notification is sent to all the observers in the list. An observer can be removed through the detach method.

The ISubject class can not be instantiated directly but has to be used to inherit from. This is achieved by having a protected constructor.

5.2.4 Data Acquisition

Access to the hardware is abstracted in the ComediHandler class. The class uses the comedi librairy (comedilib) (**?**) to read data from the hardwae device. The ComediHandler is initialised when the Processing object is created. If the initialisation fails, e.g. because there is no device connected, the application terminates and an error message is printed.

Upon a successful startup, single samples can be read from the device either as a raw integer value or as a voltage value. Optionally, the whole buffer content can be read as raw values. The application is only reading voltage values.

5.2.5 Filtering

Two filters are used to process the acquired data as described in section 3.1.1 above. The filter implementation was done by **?**. The objects are initialised when the Processing class is created. They process data sample by sample as needed by a real-time application. Whenever a new sample is passed to them, a single filtered value is returned. They are causual, so the order in which the samples are passed to it influences future outputs.

5.2.6 OBPDetection Class

The core algorithm is performed by the OBPDetection class. An object is initialised upon creation of the Processing class. The detailed description about what happens inside is provided above in the section 3.2 Deflation. Configuration values that are stored in the OBPDetection class work correspondingly to the ones stored in the Processing class directly. They can be accessed through public getter and setter methods. This could be done anytime, but should only be done when no measurement is happening. If it is done during a measurement, the behaviour can not be guaranteed.

The configuration values can only be set if they are within reasonable boundaries as defined in macros. However, a valid measurement is not guaranteed, if the values are changed from the default ones.

The data is passed to the object sample by sample like in the filters. Here, a pair of samples of pressure and oscillation data is always needed. The return value is a boolean which is true whenever a new maximum was detected in the oscillations. In this case, a new heart rate value can be read through the corresponding public method of the OBPDetection class. Additionally, the method should be checked that indicate when the measurement is finished. When the process is successfully done, the getter functions for the BP values, MAP, SBP and DBP will return valid values, otherwise they return 0.0.

5.2.7 Storing the Recorded Data

The class Datarecord stores data in a file, given a file name. There are two options, one is to store it sample by sample, the other by handing it a vector of doubles to store. If the sampling rate is supplied, it will store the values with the corresponding time. Otherwise, with the corresponding sample number. In this application, the the data is stored at the end of a measurement, so a vector is handed to the object together with a file name that represents the current date and time.

5.3 User Interface

The user interface is shown in figure 5.3 UI. It is split up into two parts. The left side accepts user input and gives instructions to the user what they have to do to take their blood pressure. The right side shows the data being acquired in real-time. There are two plots. The upper plot shows pressure data filtered with a low-pass filter of 10 Hz, the lower plot shows the data additionally high-pass filtered at

Figure 5.3: UI

0.5 Hz, which results in a bandpass filter. This is the oscillogram, the main input for the algorithm to determine the user's blood pressure.

As mentioned above, the graphical user interface (GUI) is built with Qt. The QtDesigner was used to design a first draft of the GUI, but because this does not allow integration into applications that are built outside of the Qt development environment, the whole GUI was completely built in C++ code from this draft.

5.3.1 Guided Blood Pressure Measurement

The left side of the GUI guides the user through taking blood pressure. The process is split up into five pages that are shown in figure 5.4. All pages have a dial that shows the current pressure in mmHg. The first page has information on how to prepare for the measurement. Pressing the button at the bottom of the page starts the measurement process. The button is only enabled if the processing part of the application is ready.

Figure 5.4: The screens that guide the user through taking their blood pressure in the order that they are shown.

The second page instructs the user to pump up the pressure in the cuff to a certain value. The default value is 180 mmHg. Once that value is reached, the GUI automatically switches to the next page.

This page tells the user to release the pressure in the cuff again. This should be done slowly at a rate of approximately 3 mmHg/s. There is no other feedback than the dial for how fast the pressure is being released. At the bottom of the page, the current heart rate, calculated from the latest oscillation peaks, is shown.

Once enough data is collected, the fourth page is shown. It requires the user to deflate the cuff completely to show the results. If the algorithm fails to collect enough data within a specified time (configured as 5 min) the measurement stops and goes back to the start page.

When the pressure in the cuff reaches nearly zero, the results page is shown. It displays MAP, SBP, DBP and the average heart rate during the measurement. All data is shown as whole numbers without decimals.

5.3.2 Menu

A menu bar provides access to an information pane as well as a settings pane. Both open up as dialogue windows and disable user input on the main window. Data acquisition is kept running and is displayed in the plots.

Information

The information pane shows the application version number, the licence and provides a link to the project GitHub page. Figure 5.5 shows the dialogue.

Figure 5.5: The information dialogue.

Settings

The settings dialogue (shown in figure 5.6) lets the user change some application configurations. The user is not recommended to change these if they are not aware of the consequences. The settings are stored persistently but only take effect after restarting the application. The range of accepted values is limited, to keep the algorithm working. Because not all values have been tested, the user is warned that the application might not perform reliably anymore. The values are stored once the user presses the 'OK' button. If the 'Cancel' button is clicked, the settings application is closed without saving the values.

The values can be reset by pushing the corresponding button. These changes take effect immediately.

Figure 5.6: The settings dialogue.

The handling and storing of these values is done through the QSettings class of the Qt library. It allows to save and load values by specifying a string key. If there is no key found with the provided name, the default value is selected. The default value is what is hard-coded in the Processing class.

5.3.3 Window Class

The `Window` class inherits from the `QMainWindow` class and the `IObserver` class. The QMainWindow class makes the class an executable Qt window, with Qt taking care of updating the GUI and generating events for button clicks and so on. The `IObserver` makes the class able to be registered as an observer for a subject that will send notifications to the observer. More on the `IObserver` below.

All GUI elements are set up in the constructor of `Window`, including the settings and info plane. However, they will only be shown if necessary. E.g. what page of the user instructions is shown is defined by the setting of the screen enum (currentScreen).

The Window object is instantiated with a reference to a Processing object in the constructor. This is necessary so the user inputs can be relayed and the Processing thread can be stopped when the window is closed and the application exited.

5.3.4 IObserver Class

The `IObserver` class defines the functions that the observable class (the subject) uses to notify the observer. All functions are virtual but implemented as empty functions. This way, the observing class can choose to implement and therefore listen to the notifications that it wants to and ignore the ones that it does not.

Just like the ISubject class, the IObserver class can not be instantiated directly but has to be used to inherit from.

The following methods can be implemented:

- **Ready:** Informs the observer that the subject is ready.
- **New Data:** Sends a new data pair to the observer. Each data pair consists of pressure data and oscillation data. Both are doubles.
- **Switch Screen:** Tells the observer which screen to display.
- **Results:** Sends the results to the observer. The results are three doubles for MAP, SBP and DBP.
- **Heart Rate:** Sends a new heart rate value to the observer. The value is a double.

These methods are called from a class that inherits from ISubject, which is explained in detail in section 5.2.3 above.

Thread safety

Because the methods from the `IObserver` class are called from an object, which is likely to be running in another thread than the window, it is important to implement all functions in a thread-safe manner. Qt offers a good way to do this by sending queued events.

One example of how to enable the start button, in a not thread-safe way is by accessing it directly, as follows:

```
btnStart->setDisabled(false);
```

Instead, the function `QMetaObject::invokeMethod` is called to send an event to the GUI thread. This is done by specifying the object, which function of the object to call, what arguments to set, and the type of event to send. The `Qt::QueuedConnection` will send an event into a queue that will be handled when the thread is executed. The function to call is given as a string argument. Considering the example above, this results in the following statement:

```
bool bOk = QMetaObject::invokeMethod(btnStart, "setDisabled",
 ↪  Qt::QueuedConnection, Q_ARG(bool, false));
assert(bOk);
```

The return value confirms if the connection could be made, i.e. the given string is a valid function to call for the given object. This is tested through an assert during development. The assert will fail if the boolean is not true. It is important not to have the assert around the whole function call, because it will be removed in the release build.

The plots have their underlying data set changed every time a new sample is available because a new sample is always added on the right side and an old one is discarded on the left side. This is made by changing the raw data in memory and therefore, the Qt process is not informed about that. To solve this issue, the plots are regularly updated manually. This happens in a timer event with an interval of 50 ms, which is enough for the human eye to see a continuous movement. To avoid thread safety problems, the plot objects are accessed after acquiring a mutex.

5.4 Third Party Software

The following third party software is used in the developed application:

- **Qt** 5.12.8 LTS, an open-source widget toolkit for creating graphical user interfaces. (**?**)
- **CppThread**, a wrapper to the std::thread class written by Bernd Porr to avoid static methods. (**?**)
- **iir1** An implementation of the infinite impulse response (IIR) filters for sample-by-sample, real-time processing written in C++ by Bernd Porr. Provided as a library. (**?**)
- **plog** 1.1.5, a portable, simple and extensible C++ logging library. (**?**)
- **comedilib** 0.11.0, a library that provides an interface to Comedi data acquisition devices. (**?**)

Chapter 6

Results and Discussion

This chapter highlights the important results of the project. It starts off discussing the outcomes of preliminary tests that were done offline with prerecorded data using python.

6.1 Python Algorithm Tests

Before deciding on an algorithm to implement, some test data was recorded and analysed offline using python. Different approaches of the MAA were implemented as well as the derivative algorithm. All python tests are done on the same dataset. The manually determined blood pressure was 110/70, with a large, expected error because the pressure was taken by an untrained person on themselves.

The top plot in figure x shows the by now familiar outline of a BP measurement. The bottom plot illustrates how it was analysed. The oscillations were examined for local extrema and the resulting points connected for the maxima and minima. The result is the blue line that forms the envelope around the oscillations. The cyan line is the time difference between two subsequent maxima. From this, the heart rate can be calculated. It is used to determine when to analyse the oscillations. When the heart rate is stable and changes less than 20 % per detection, the start of the oscillometric envelope is assumed. When it changes dr, the end is recognised. These points are identified in the bottom plot in figure x by the vertical lines.

6.1.1 Fixed-Ratio Algorithm

Figure y shows the same data as before. The top and middle plot are zoomed in from figure x. The bottom plot shows the different ways the OMWE can be calculated. The blue trace is simply the detected maxima connected. The green track is every minimum subtracted from its preceding maximum. Every millisecond interpolated between each of the detected maxima and afterwards, every minimum point subtracted from every maximum form the red trace. Finally, the purple track is a polynomial of 4th order fit to the points calculated in the green track.

	max value	min/max value	interpolation	polynomial fit
MAP (mmHg)	82.64	85.50	82.49	81.53
SBP (mmHg)	97.21	98.83	99.14	105.63
DBP (mmHg)	73.04	72.64	71.83	66.53

Table 6.1: Comparison of different methods to form the OMWE on the calculated BP values for MAA. MAP was estimated at the maximum of the OMWE, SBP at a ratio of 0.55 of the maximum while rising and DBP at a ratio of 0.70 of the maximum while falling.

MAP is estimated where the envelopes have their maximum as explained in chapter A. Figure z shows the OMWEs again but normalised by their maximal value for comparison. Vertical lines are drawn at 0.55 and 0.70 where SBP and DBP are estimated in this example.

Table 6.1.1 lists the values for MAP, SBP and DBP that were calculated for each of the OMWE formation methods.

All methods estimate MAP within a range of 4 mmHg, the furthest off is the min-max OMWE (green) because it is influenced by small shifts in time. Surprisingly, the OMWE using only the maximum values (blue) estimates MAP closer to the other methods. However, while oscillations are increasing and decreasing, this blue trace shown in figure z seems to lag behind the others in time and SBP is therefore estimated slightly lower, with 97 mmHg. DBP is not estimated lower with 73mmHg. This is due to a pulse peak that is happening in the deflating data. This is a weakness in the python script, it estimates the values at their exact position in the deflating plot without taking into consideration the small pulses that are still occurring there.

The purple polyfit trace does a decent job identifying MAP, but the trace hardly fits the envelope. As is shown in figure z, it is wider than the other traces and therefore estimates SBP sooner and DBP later, resulting in a considerably higher value for SBP with 106mmHg and lower value for DBP with 66mmHg.

The green OMWE that is considering minima and maxima is similar to the red interpolation one between 23 to 27s. Looking back at figure y, minima follow quickly after maxima in this period. The way this trace is calculated explains why it is slightly shifted backwards in time compared to the red one: The values are evaluated when the maxima happen and the following minimum is subtracted from it.

Altogether, the red interpolation trace looks the most continuous. Changes in timings of minima and maxima have the least effect on it. The determination of the pressure at the identified time should be done by averaging over the current pulse and not by evaluating the given sample directly from the low-pass filtered deflation curve.

6.1.2 Derivative Algorithm

Figure x shows the oscillation data plotted in time on the left-hand side. The top is the deflating pressure and the bottom the shows the oscillations.

	not normalised	normalised in time	normalised in pressure
SBP (mmHg)	94.87	94.87	94.87
DBP (mmHg)	80.76	80.76	73.94

Table 6.2: Comparison of blood pressure values calculated by the derivative algorithm for differently acquired derivatives.

The right-hand side shows the oscillations plotted in terms of pressure when they were happening. Note that this changes the x-axis and the resulting plot is an inversion in time, because the lower pressures happened later than the higher pressures.

The plot on the top right shows how this OMWE is formed from the detected maxima and minima. The bottom-right plot shows the derivative of the OMWE. The blue trace is the difference between two successive data points. Because they can be spaced unevenly, the orange trace has the values normalised by the time difference between them. The green trace normalises the values with the pressure difference. While the green trace seems to make the most sense logically, they all look similar, except for the last one being compressed compared to the others. A variation of this figure, where all derivative traces have been normalised by their maximal value for comparison, can be found in the appendix.

Taking the maximum value of the derived OMWE results in the DBP and the minimum value in the SBP as explained in section 2.3.2 Derivative Algorithm. Table x lists the values for SBP and DBP that were computed for each of the OMWE methods. SBP is identified at 94.87 and DBP at 80.87 or 73.94. The only difference in the three methods is the DBP that is estimated lower by the derivative that was normalised in pressure. This is most likely the closest to the actual value.

Other tests, using only the maxima or minima for derivation have produced plots that are even more ambiguous. For this particular example, the minima are smooth and the derivative had a clear indication for the SBP. The plots for this test are located in the appendix.

Additionally, it has to be noted that the tests were done using a very clean data set. If the deflation curve is not as stable as in the example, the derivative algorithm does not work at all. Similarly, even if the deflation is stable, but there are any other artefacts in the oscillations, e.g. from micromovements or natural blood pressure variations over time, wrong blood pressure will be measured. Figure x shows the same algorithm using non-ideal data that was recorded within minutes of the data shown above. All methods determined SBP at 84.65 mmHg and systolic DBP at 62.58 mmHg. Effectively, being 10 mmHg too low for both values compared to the previous measurement analysed by both the derivative and the fixed-ratio algorithm.

6.1.3 Summary

Table 6.1.3 lists the values of the results from the two best options for the fixed-ratio and derivative algorithm. For the fixed-ratio algorithm, the interpolated version is chosen, because it has the smoothest

	fixed-ratio interpolated	derrivative normalised in pressure
MAP (mmHg)	82.49	-
SBP (mmHg)	99.14	94.87
DBP (mmHg)	71.83	73.94

Table 6.3: Comparison of blood pressure values calculated by the fixed-ratio and the derivative algorithm.

curve. The derivative algorithm is logically and in the current measurement best for the version normalised in pressure. However, even using clean data that is optimal for the derivative algorithm, the values have little confidence. There is no clear rising and falling curve that has a clear maximum and minimum, but rather an assembly of maxima and minima. Even the smallest, changes in amplitude or time difference have a significant impact on the results. Therefore, this approach is not implemented in the application.

As expected, the fixed-ratio algorithm produced stable estimates of MAP. Of course, it also depends on how the OMWE is defined. The interpolated version has the least sensitivity to noise because it considers the time difference between minima and maxima. SBP and DBP are highly dependant on the chosen ratios. For this test, the ratios were chosen from a mean value of what was recommended in literature from chapter x.

6.2 Application Implementation

Debugging a real-time program is tricky. To assess the results, C++ calculations are compared to python. Next, Korotkoff sounds, recorded while taking a measurement, are compared to the OMWE. Finally, a series of measurements are made within a short period to test reproducibility.

6.2.1 Comparison to Python Algorithm

To make sure the in C++ implemented algorithm does what it is expected to, a measurement is performed and compared against the python algorithm. From the application, not just the raw data was saved, also low and high-pass filtered data and the OMWE that is used by the algorithm were written to a file. The raw data was analysed by the same python script described above (section x). The filtered and calculated data from the application is plotted over it in figure x. The used filters are the same and perform correctly, as can be seen in the pressure plot at the top and the oscillation plot in the middle.

The bottom plot shows the OMWE calculated by the application matches the interpolated version in python. Because interpolation is done linearly in python, only calculating the edge values results in the same trace as the fully interpolated version, with considerably fewer computations.

Table x shows the calculated values for the python and application implementation. The difference is that the application uses an average pressure value around the determined time, whereas the python script directly looks up the pressure value at the given time.

6.2.2 Korotkoff Sounds

Figure x shows a test where a stethoscope was placed under the cuff to record Korotkoff sounds while performing the measurement. The resulting values are; 109 mmHg for SBP, 77 mmHg for MAP and 71 mmHg for DBP with an average heart rate of 74 beats/min. Korotkoff sounds only allow estimations of DBP and SBP. In figure x, the green arrows are linked to the OMWE, which is displayed normalised to identify the ratios. As before, the values are 0.5 for rS and 0.8 for rD. (1) shows where the algorithm expects systolic pressure and (2) diastolic pressure. The orange arrows relate to the sounds displayed in the central plot. (3) is where the sounds first appear, (4) where they are muffled and (5) where the sounds disappear entirely. According to **?**, DBP is located where the sounds are muffled. This point is difficult to see in the plot, but easy to hear. It corresponds well with the ratio of 0.8 for this measurement. Because the ratios are known to be dependant on factors like age and health, the software allows to change them in the settings. On the other hand, if the guidelines are followed that DSP is located where Korotkoff sounds disappear (**??**), the ratio should be less than 0.5.

Altogether, the chosen ratios seem to work decently. According to **?**, the auscultatory method underestimates SBP and overestimates DBP, which fits with our findings. However, the ratios are known to be highly dependent on BP, age and health of the subject and the pressures calculated from them should only be used as indications and not fixed values.

6.2.3 Repeated Measurements

Unfortunately, the ethical application to perform experiments with test subjects was rejected given the current circumstances. Therefore, the only tests were performed on the developer.

Figure x shows how the measured values vary over 20 minutes, performing ten measurements in a row. The ratios for systolic and diastolic BP were 0.5 and 0.8. For each determined value type, the maximal, minimal and average is indicated. The individual measurements can be found in the appendix. Besides MAP, DBP and SBP, the heart rate was noted as an indicator of natural variations. The heart rate is the signal that can be registered most accurately. Both MAP and SBP have significant differences of 16 mmHg and 15 mmHg. The heart rate and DBP vary less than 10 mmHg. For the DBP, this is surprising. The research with python has shown that irregularities in amplitude are more common in the diastolic range. One explanation is because oscillations decline quicker than they increase, the pressure range is smaller for DBP. However, this explanation is based on subjective observations and might be misleading.

A manual blood pressure measurement was performed before and after the automatic ones. Before, BP was determined at 110/80 and after at 100/70. These should only be used as a reference and not as the 'right' values. The tester performed them on themselves, with the stethoscope tucked under the cuff, which is known to influence the measurement. Additionally, the person is not trained to take BP.

Still, it shows that it is likely that the average blood pressure dropped within the 20-minute time frame, which can account for part of the variations.

Chapter 7

Conclusion

This chapter discusses the outcome of the project and proposes the next steps to take.

The goal of this project was to be able to conduct experiments with test subjects and to record datasets. Unfortunately, due to the ongoing situation when this project was conducted, this was not possible. Even though all possible safety measurements were considered and approved by the University's risk assessment department, the approval process was rendered impossible to go through within the given time frame. Apart from not being able to record any datasets, the algorithm might suffer from systematic errors due to subjective characteristics in the developers blood pressure oscillations.

- valve needs to open continuously to have a steady stream of 3mmHg/s opening introduces noise
- many algorithms only work for clean data
- MAP is used for diagnostics
- why are microphones not used to determine blood pressure with the karakoff sounds?
- PTT (beyond oscillometry)

..

7.1 General Problems with Algorithms

most algorithms expect perfect data to be 'perfect'

natural variablility of blood pressure during measurement left and right arm difference 10mmHg normal

mmHg is no longer a SI unit since 2019

Bibliography

Association for the Advancement of Medical Instrumentation (AAMI), Sphygmomanometer Committee, American National Standards Institute, Inc. Non-invasive sphygmomanometers — part 2: Clinical investigation of automated measurement type. Technical report, 2013.

Charles F. Babbs. Oscillometric measurement of systolic and diastolic blood pressures validated in a physiologic mathematical model. *BioMedical Engineering Online*, 11, 2012.

Walter F Boron and Emile L Boulpaep. *Medical Physiology, 2e Updated Edition E-Book*. Elsevier health sciences, 2012.

Willem J. W. Bos, Elisabeth Verrij, Hieronymus H. Vincent, Berend E. Westerhof, Gianfranco Parati, and Gert A. Van Montfrans. How to assess mean blood pressure properly at the brachial artery level. *Journal of Hypertension*, 25, 2007.

British and Irish Hypertension Society. Bp monitors, 2020. URL `https://bihsoc.org/bp-monitors/`. Accessed: 2020-08-02.

Anand Chandrasekhar, Mohammad Yavarimanesh, Jin Oh Hahn, Shih Hsien Sung, Chen Huan Chen, Hao Min Cheng, and Ramakrishna Mukkamala. Formulas to explain popular oscillometric blood pressure estimation algorithms. *Frontiers in Physiology*, 10, 2019.

G. Drzewiecki, R. Hood, and H. Apple. Theory of the oscillometric maximum and the systolic and diastolic detection ratios. *Annals of Biomedical Engineering*, 22, 1994.

Mohamad Forouzanfar. A modeling approach for coefficient-free oscillometric blood pressure estimation. 2014.

Mohamad Forouzanfar, Hilmi R. Dajani, Voicu Z. Groza, Miodrag Bolic, Sreeraman Rajan, and Izmail Batkin. Oscillometric blood pressure estimation: Past, present, and future. *IEEE Reviews in Biomedical Engineering*, 8, 2015.

L. A. Geddes, M. Voelz, C. Combs, D. Reiner, and C. F. Babbs. Characterization of the oscillometric method for measuring indirect blood pressure. *Annals of Biomedical Engineering*, 10, 1982.

Vojko Jazbinsek. New algorithm for automatic determination of systolic and diastolic blood pressures in oscillometric measurements. 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings, 2016.

Vojko Jazbinsek, J Luznik, and Zvonko Trontelj. Non-invasive blood pressure measurements: separation of the arterial pressure oscillometric waveform from the deflation using digital filtering. 01 2005.

Vojko Jazbinsek, Janko Luznik, Stephan Mieke, and Zvonko Trontelj. Influence of different presentations of oscillometric data on automatic determination of systolic and diastolic pressures. volume 38. Annals of Biomedical Engineering, 2010.

Eddy Joe. Blood pressure measurements in the icu: Trust only the map in oscillometric devices!, 2019. URL `https://www.youtube.com/watch?v=TG0NcqKeRWE`. Accessed: 2020-08-02.

Belinda Kneubühler. Oscillometric blood pressure measurement, 2020. URL `https://github.com/itsBelinda/obp`.

Soojeong Lee, Joon-Hyuk Chang, Sang W. Nam, Chungsoo Lim, Sreeraman Rajan, Hilmi R. Dajani, and Voicu Z. Groza. Oscillometric blood pressure estimation based on maximum amplitude algorithm employing gaussian mixture regression. *IEEE Transactions on Instrumentation and Measurement*, 62(12):3387–3389, 2013.

Pooi Khoon Lim, Siew Cheok Ng, Wissam A. Jassim, Stephen J. Redmond, Mohammad Zilany, Alberto Avolio, Einly Lim, Maw Pin Tan, and Nigel H. Lovell. Improved measurement of blood pressure by extraction of characteristic features from the cuff oscillometric waveform. *Sensors (Switzerland)*, 15, 2015.

Gemma Lloyd. Clinical guidelines for measuring/ monitoring blood pressure. Technical report, Mersey Care NHS Foundation Trust, 2018.

Majid Mafi, Sreeraman Rajan, Miodrag Bolic, Voicu Z. Groza, and Hilmi R. Dajani. Blood pressure estimation using oscillometric pulse morphology. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2492–2496, Aug 2011. doi: 10.1109/IEMBS.2011.6090691.

Majid Mafi, Sreeraman Rajan, Miodrag Bolic, Voicu Z. Groza, and Hilmi R. Dajani. Blood pressure estimation using maximum slope of oscillometric pulses. *Conference proceedings (IEEE Engineering in Medicine and Biology Society. Conf.)*, 2012:3239, 2012.

G. W. Mauck, C. R. Smith, L. A. Geddes, and J. D. Bourl. The meaning of the point of maximum oscillations in cuff pressure in the indirect measurement of blood pressure-part ii. *Journal of Biomechanical Engineering*, 102:28–33, 1980.

Eoin O'Brien, James Petrie, William Littler, Michael de Swiet, Paul L. Padfield, Kevin O'Malley, Michael Jamieson, Douglas Altman, Martin Bland, and Neil Atkins. The british hypertension society protocol for the evaluation of automated and semi-automated blood pressure measuring devices with special reference to ambulatory systems. *Journal of Hypertension*, 11:S43–S63, 1993.

Organisation Intergouvernementale de la Convention du Mètre. The international system of units (si). 2006.

Sergey Podobry. Plog - portable, simple and extensible c++ logging library, 2019. URL `https://github.com/SergiusTheBest/plog`. Accessed: 2020-08-10.

Bernd Porr. Cppthread, 2020a. URL `https://github.com/berndporr/cppThread`. Accessed: 2020-08-10.

Bernd Porr. Iir1 – realtime c++ filter library, 2020b. URL `https://github.com/berndporr/iir1`. Accessed: 2020-08-10.

Dominik Pražák, Václav Sedlák, Ekrem Sınır, and František Pluháček. Changing the status of mmhg. *Accreditation and Quality Assurance*, 25(1):81–82, 2020. ISSN 1432-0517. URL `https://doi.org/10.1007/s00769-019-01414-7`.

Qt, 2018. URL `www.qt.io`. Accessed: 2020-08-10.

Maynard Ramsey. Noninvasive automatic determination of mean arterial pressure. *Medical and Biological Engineering and Computing*, 17(1):11–18, 1979. ISSN 1741-0444. URL `https://doi.org/10.1007/BF02440948`.

Richard A. Reeves. Does this patient have hypertension?: How to measure blood pressure. *JAMA: The Journal of the American Medical Association*, 273:1211–1218, 4 1995.

A. Sapinski. Standard algorithm for blood pressure measurement by sphygmo-oscillographic method. *Medical and biological engineering and computing*, 34(1):82–83, 1996.

David Schleef. Comedi - linux control and measurement device interface, 2017. URL `https://github.com/Linux-Comedi/comedilib`. Accessed: 2020-08-10.

M. Ursino and C. Cristalli. A mathematical study of some biomechanical factors affecting the oscillometric blood pressure measurement. *IEEE Transactions on Biomedical Engineering*, 43(8): 761–778, 1996.