POLITECNICO DI TORINO

Master's degree course in Biomedical Engineering

Master's Degree Thesis

Development of Automated Testing Algorithms for Graphical User Interface and Pulse Wave Velocity Evaluation on LDV Signals

Supervisors
Filippo Molinari
Massimo Salvi
Silvia Seoni

Candidate
Francesco Capuzzo

July 2023

Abstract

According to the World Health Organization, Cardiovascular diseases are the leading cause of death in the world nowadays. Over the years, numerous approaches have been tried in the attempt to detect and prevent them, resulting in the selection of physiological risk factors responsible for generating these pathologies. Among the various detectors, aortic stiffness is considered one of the most accurate indicators for preventing the development of diseases related to the cardiovascular system. It is currently assessed by measuring Pulse Wave Velocity (PWV) in the carotid-femoral segment and, over the years, efforts have been made to calculate this variable as simply and, above all, non-invasively as possible. As part of the EU-funded InSiDe project, a device has been developed which is able to calculate PWV from LDV signals obtained from Laser Doppler technology. The device is equipped with a handset, used by the healthcare provider to detect the LDV signals at two critical points of the patient, and a personal computer with a Graphical User Interface (GUI), where the clinical data of the patient can be entered, and the signals can be visualised during the acquisition. The aim of the present work is the implementation of automatic controls to be integrated within the GUI, as tests, relating to the input of patient data by an operator. With the help of the Unittest library, it was possible to configure the individual tests independently, collectively incorporated inside a single algorithm. The implementation of these controls is crucial in the development of the patient report in order to avoid any unforeseen events or misunderstandings in future consultations. An automatic algorithm was therefore created that would generate confirmation or warning alerts based on the input of individual physiological parameters. The data and ranges of values used for some parameters were taken from a report compiled by the InSiDe project containing the list of checks to be performed in the GUI, for others they were chosen arbitrarily in a reasonable manner, while waiting for feedback and confirmation from the clinical staff. A further aim of the work was to implement a function in the Python environment for evaluating PWV from LDV signals, which would be integrated into the GUI. The displacement signals utilised came from acquisitions performed with the help of the CARDIS device, a European project preceding the current InSiDe, on a total of 100 patients at the Hopital Europen Georges Pompidou (HEGP) in Paris. The technique used in the function to assess PWV is a median of the median PWVs, calculated from the estimated Pulse Transit Time and fiducial points inside the signals. The existing function was designed in the Matlab programming language, the contribution of this thesis work was the implementation of it in the Python environment, so that it could be integrated within the GUI algorithm and allow a more accurate estimation of the signals acquired by the InSiDe device.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

According to the World Health Organization (WHO), Cardiovascular diseases (CVD) are described as a "group of disorders of hearth and blood vessels"[1] and they are currently the leading cause of death worldwide. In the United States especially, Heart diseases (HD) remained the primary cause of death in 2020, even in the presence of COVID-19, with the number of total deaths caused by CVD increased in the last decade[2]. Arterial stiffness is a significant factor for future developments of CVD: it depends directly on arterial pressure and it's determined by factors as the vascular muscle tone or contents within the vessels walls and its assessment may be considered as marker for CVD prevention. While the systemic arterial stiffness can only be evaluated with circulation models, with modified Windkessel models making use of pulse contour analysis, regional and local arterial stiffness can be measured non-invasively pursuing different techniques, although the use of the Pulse wave velocity (PWV) is considered one of the best, non-invasive and powerful method to determine arterial stiffness. PVW can be evaluated along different regions of the arterial tree and its accuracy improve with longer measurement distances and better arterial segments shape.
The PVW Carotid-Femoral measurement is considered the "gold-standard" for the evaluation of the arterial stiffness, particularly the aortic stiffness, which is considered the principal predictor and contributor of HD events[3]. Consequently, a non-invasive measurement of aortic stiffness could facilitate large-scale population screenings for accurate predictions of CVD risk and the implementation of preventive therapies for individuals deemed at low or moderate risk. To enable such screenings, a non-invasive, portable, cost-effective device capable of measuring aortic stiffness and PWV is required.
Several devices have been introduced, with different measuring methods and sensors: two of the most popular are the ®Complior[4], which makes use of piezoelectric sensors, and ®SphygmoCor[5], that works with applanation tonometers. However, both devices are strongly reliant on the operator experience and knowledge and the correct position of the probes. As a consequence, from 2015 to 2019, the

Community Research and Development Information Service (CORDIS), together with other contributors and participants, developed a mobile, low-cost screening device called CARDIS. This device is based on a silicon photonics integrated laser vibrometer, which enables the detection of arterial stiffness, stenosis, and heart dyssynchrony.
The Laser Doppler Vibrometer (LDV) integrated into CARDIS allows for non-contact measurement of skin surface displacement and velocity with exceptionally high resolution, down to femtometer size, extracting the motion information from the Doppler shift of the reflected laser beam[6]. The arterial stiffness and the PWV evaluation through the LDV are recently demonstrated as a valid alternative than applanation tonometry or Doppler Ultrasound technique. The CARDIS device works with a silicon photonics chip that contains the optical functionality of a multi-beam LDV. The project ended in 2019, after performing different clinical feasibility studies collecting robust datasets from healthy subjects and from cardiovascular risks subjects[7, 8, 9].

1.1 InSiDe project

Funded from the European Union's Horizon 2020 Research and Innovation Programme, the InSiDe project consists of the development of a multi-beam LDV device for the accurate measurement of pulse-generated movements. The evaluation is done with an optical interferometer that detects the change in frequency and phase of laser light reflected from a target, created by the Doppler effect if the target is moving. The development of the InSiDe device builds on the previous CARDIS project, completed in January 2019, and aims to create an even more effective and high-performance measurement tool[10].

1.1.1 InSiDe demonstrator device

The main components of the InSiDe device are two battery-powered, wireless communicating handpieces with four sensing beams. They are designed for combined applications, ideal for measuring cardiac contractions or local stenosis, or as separate units for synchronised pulse wave measurements and consecutive carotid-femoral PWV assessments.
The two handpieces are divided into master handpiece and slave handpiece (Figure 1.1). The former is equipped with the trigger start button for managing the acquisition, while the latter is only provided with the measuring and data transfer system towards the master handpiece. Both of them have led indicators for battery and signal levels. As shown in the component schematic in (Figure 1.2), the master handpiece is equipped with a sensing system consisting of a photonic package hermetically sealed,a photonic integrated circuit (PIC), amplifiers and a

demodulation module, which provides local real-time demodulation of raw signals to displacement time series data.

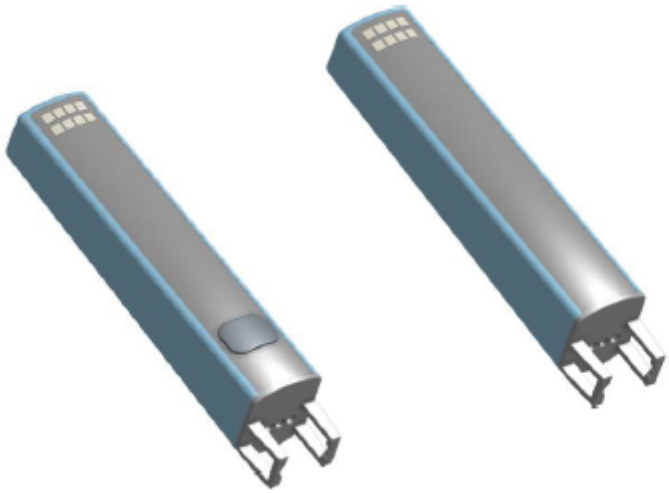

Figure 1.1: *Master handpiece and Slave handpiece of the InSiDe demonstrator device*

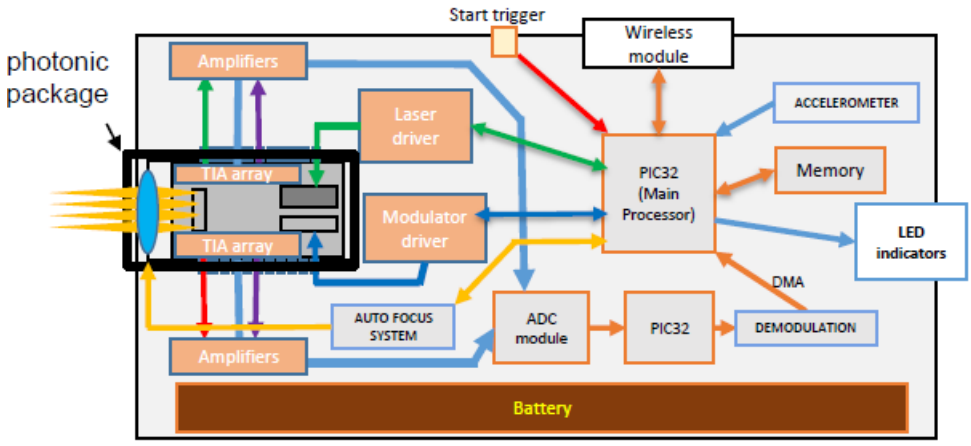

Figure 1.2: *System diagram of the handpiece*

The InSiDe demonstrator device is designed with a 4-beam homodyne interferometer,implemented on a silicon chip, in order to capture timed traces of mechanical displacement and skin vibrations using LDV with high precision thanks to the 1310nm laser assembly. The reflection beam and a reference beam from the same laser source are combined in a 90-degree optical hybrid, and the resulting beams are detected by photodetectors (PDs). The optical system is equipped by confocal

lenses, optical antennas and autofocus lenses with higher numerical aperture to increase light pick up and adjust focal point to match skin position.

1.1.2 InSiDe objectives

The main objective of the InSiDe project is the development and release of a hand-held wireless clinical investigational device.
This device must be provided with the following features:

- Measure aortic stiffness by assessing PWV and provide on-screen display of results
- Quantify arterial stenosis
- Record traces of cardiac contractions by marking fiducial points and calculating time parameters
- Transfer recorded data to external servers for future verification and evaluation

1.2 Arterial stiffness and PWV

This work contributed to the implementation of a PWV evaluation algorithm in the Python environment, to be integrated within the GUI functionality. The calculation code of the PWV, designed and developed by third parties, was initially created using Matlab software, however, the necessity to combine it with the functionality of the device Interface required a transposition of the algorithm into the Python environment, where the main code of the GUI structure is based. Before presenting the functionalities of the PWV evaluation algorithm and its main steps, it was preferred to introduce the theoretical and experimental principles related to arterial stiffness and Pulse Wave Velocity, in order to contextualise the purposes of the creation of this computation algorithm.

1.2.1 Definition of arterial stiffness

Large-calibre arteries such as the ascending and descending aorta, or the subclavian artery, are characterised by a high percentage of elastin and a layer of smooth muscle cells within their tunica media. This peculiarity allows them to withstand the stress created by variations in blood pressure during the systolic phase and release the accumulated elastic energy in the following diastolic phase, pushing the blood volume towards the peripheral organs. The role of this muscular layer of the arterial wall is to work as a capacitor, storing and releasing energy and

blood volume. In this way, the discontinuous flow of cardiac output is progressively attenuated until it becomes a steady flow at the capillaries, according to an effect known in medicine as the 'Windkessel effect' (Figure 1.3).
Factors such as age or a very bad way of living cause deterioration of the muscular layer, making it stiff and fibrous, compromising its propulsive capacities necessary for the correct functioning of the cardiac cycle. Consequently, as the stiffness of the arterial wall increases, so does the probability of the developing of a cardiovascular disease. Monitoring of arterial stiffness values is therefore essential to prevent the occurrence of CVD.

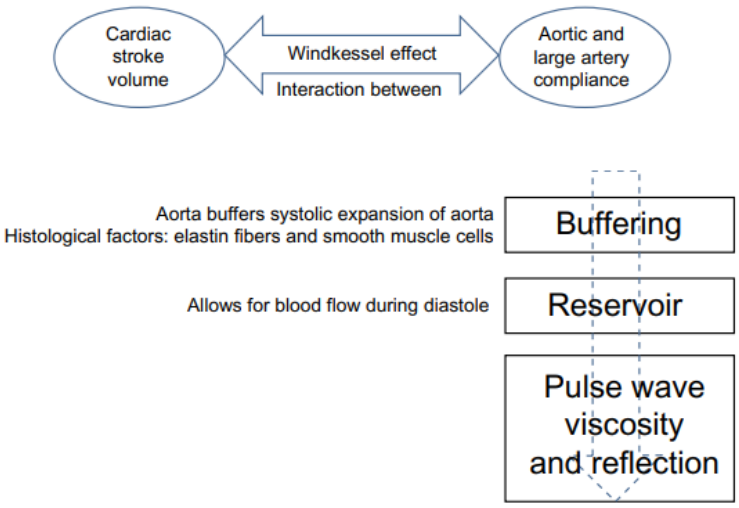

Figure 1.3: *Windkessel effect: the muscle layer of the aorta absorbs the energy and attenuates the pulsatile flow* [11]

1.2.2 Arterial stiffness measurement methods

The nature of the phenomenon of arterial stiffness brings with it certain restrictions when measuring it: to be able to measure stiffness directly, the instantaneous variation in blood pressure and diameter in the same arterial section must be measured simultaneously and accurately. To obtain these readings, invasive methods of measurement using intra-arterial accesses and pressure transducers must be applied.
Nowadays, however, non-invasive techniques have also been developed for estimating arterial stiffness such as pulse waveform, ultrasound methods and magnetic resonance techniques or the augmentation index, defined as "the contribution of the reflected wave to the central-pressure waveform by expressing augmented pressure as a percentage of pulse pressure"[13].
One of the most widely used reference techniques is the measurement of pressure wave velocity inside the vessel, called **Pulse Wave Velocity** (PWV). It has been

recognised by the European Society of Hypertension within the guidelines for the management of hypertension as an indicator of harmfulness and danger for individuals at risk[14].PWV can be measured non-invasively using techniques such as Ultrasound Doppler or MRI and at different locations in the human body such as the carotid-femoral tract, which is now considered the gold-standard for assessing arterial stiffness by calculating PWV, also known as Aortic PWV, which is used to derive aortic stiffness.

1.2.3 PWV definition

By theoretical definition, PWV is a measure of the space travelled by the pressure wave in a given period of time:

$$PWV = \frac{dx}{dt}$$

However, relations between the Pulse Wave Velocity and the distensibility of a vessel have been developed throughout history: the **Moens-Koertweg** equation, as a matter of fact, puts these two quantities in relation with a square root [15]:

$$PWV = \sqrt{\frac{E_{\text{inc}} \cdot h}{2 \cdot r \cdot \rho}}$$

Thus, from the wave velocity it is possible to determine the elastic modulus of the vessel (E_{inc}), knowing also other geometric parameters of the vessel (r and h) and physiological parameters of the blood (ρ).
Subsequently, the relationship known as the **Frank/Bramwell-Hill** equation is developed, where there is a quadratic proportion between the velocity and the distensibility of a vessel D [16]:

$$PWV = \sqrt{\frac{V \cdot dP}{\rho \cdot dV}}$$

1.2.4 PWV measurement methods

There have been considerable technological advances since Moens and Korteweg together with Bramwell and Hill conducted their researches; however, the fundamental principle of PWV measurement has remained unchanged. It is based on measuring the time required for the pressure wave to cover a certain distance. To achieve this, it is essential to accurately measure the transit time of the wave and the distance travelled.
The measurement of transit times, called **Pulse Transit Times** (PTT), involves determining the time difference between specific points on the waveforms of the carotid and femoral arteries.The selection of these characteristic points depends on

the type of waveform being analyzed (such as flow, pressure, or diameter distension) and the algorithm employed to detect them. Among the various algorithms utilised, the two most widely used are the intersecting tangent algorithm or 'foot-to-foot' method (Figure 1.4)(adopted, for example, by the SphygmoCor system and manual evaluation) and the algorithm based on the point of maximum systolic ejection (used, for example, by the Complior system), although the difference in PWV estimation between the two could be as much as 15% of the measurement [17]. As mentioned earlier, the measurement of the wave path, especially the carotid-femoral path, is also extremely important for the accuracy of the assessment. A direct measurement of the tract is the simplest method but leads in most cases to an overestimation of PWV, up to 30% [18].
Different studies have proposed mathematical equations to calculate this quantity indirectly; however, the technique that allows the most accurate path length estimation is the direct measurement of the carotid-femoral tract (L_{CF}) scaled by a factor of 0.8. Indeed, the overestimation using this measurement method is only 0.4% [19].

$$dx = 0.8 \cdot L_{CF}$$

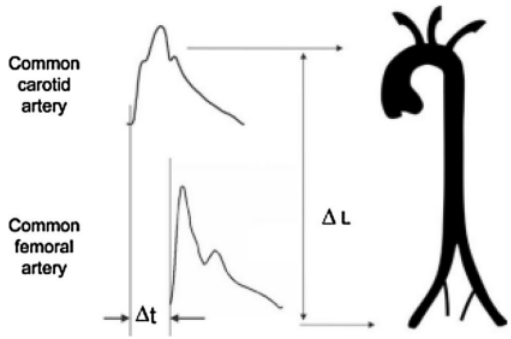

Figure 1.4: *PWV measurement through the foot-to-foot method, example of cf-PWV measured between the feet of the two waveforms* [20]

It is possible to record the pulse waves using different techniques such as tonometry, piezoelectricity, Doppler effect (ultrasonography) and impedance measurement.

1.2.5 Tonometric techniques

The most widely used methodology for measuring the pressure wave is applanation tonometry. It consists of a small pressure of an artery by a pressure sensor (tonometer) against the bone immediately underneath, obtaining a highly accurate

recording of the transited wave, almost the same as the one acquired with an intra-arterial transducer [12]. Tonometric devices capable of simultaneously recording pressure wave transitions have been developed to obtain an automatic measurement of PWV.
The SphygmoCor System, with a single applanation tonometer, records the two proximal and distal waveforms in a short time, automatically calculating the PWV based on the transit time of the wave at the two sites. Specifically, the PWV is calculated by subtracting the latency time between the R-wave of the ECG and the proximal wave with the time between the R-wave and the distal wave.
The Complior System, on the other hand, determines the PTT by correlating the carotid pressure wave and the femoral pressure wave, acquired simultaneously by mechanical sensors applied to the patient's skin.

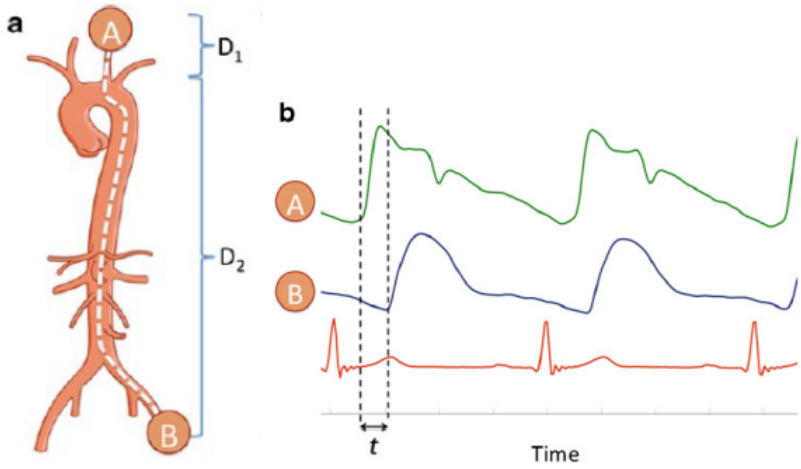

Figure 1.5: *Sequential measurement of cfPWV (A = measurement site and form carotid waveform, B = measurement site and femoral waveform; D1 = aortic-carotid distance aortic-carotid distance, D2 = aortic-femoral distance)[21]*

1.2.6 Ultrasonographic techniques

Using distension waves acquired by high-definition ultrasound probes, Pulse Wave Velocity can be derived. The SphygmoCor device calculates PWV by recording the pressure wave at two arterial sites (e.g., the common carotid artery and the femoral artery) over a short period of time and using an ECG recording to determine the time delay from the R-wave. Typically, measurements are taken at the root of the subclavian artery and near the bifurcation of the abdominal aorta. This method has been used to demonstrate the predictive value of aortic PWV in terms

of cardiovascular events in diabetic patients and appears to provide more accurate PWV values than those obtained at the carotid-femoral level, although no specific advantages have been demonstrated to date [12].

Chapter 2

Graphic device Interface

The device's Graphical User Interface (**GUI**) allows the user to view the signal trends acquired by the probes on a monitor and assess their quality for any possible recording and evaluation of quantities such as Pulse Transit Time and Pulse Wave Velocity. It is also provided with text lines that can be filled in with the patient's data and corresponding physiological parameters before acquisition. There are also indicators of the quality of the received signal, the device's battery level and the instantaneous strength of the wireless connection. The present work contributed to the automation of alphanumeric tests and checks on the inputs in the text boxes of the patient's data by implementing external functions that, once integrated into the Interface algorithm, simulate the user's actions by generating a considerable number of combinations and while verifying the feedback received. Before presenting the functioning of the implemented tests, it was decided to include a description of the Interface configuration and the communication protocol used between the server and the client.

2.1 Graphic User Interface configuration

The Interface was created with the help of QtDesigner software[22] and characterised by the Python PyQt5 library[23]. The Interface is presented to the user in three main windows (tabs): the first is for the acquisition of signals by means of two handpieces for the following calculation of the PWV between the two chosen points (Figure 2.2); the second is for TSD recording, for acquisition by means of a single handpiece and on-site evaluation of any cardiovascular pathologies (Figure 2.3); the last tab handles the connection between the device and the server, with buttons for creating a socket connection and text lines with connection status messages and parameters such as the IP address. When the software starts up, a Login pop-up window (Figure 2.1) appears asking the user to enter their credentials (Username

and Password) into the appropriate text boxes. Users who have access to the device are pre-registered and classified into two different levels: User and SuperUser. Normal users (User) have full access to the first two tabs of the Interface, dedicated to PWV and TSD, while in the last tab only the button concerning the connection of the device with the Interface remains enabled (Figure 2.4). User management with promotion/removal options is only allowed for admins with SuperUser status (Figure 2.5).
The layout of the first two tabs is very similar: the main difference consists in the absence of one of the two signal display windows within the TSD recording tab, since this type of acquisition is carried out using a single handpiece. The central part of the tab dedicated to PWV is divided into two distinct boxes called "*Graph*" and "*Measurement status*". Inside the first box there are two windows for the visualisation of the signals acquired by two handpieces, four buttons next to the name of each of the two graphs indicating the deselectable/selectable acquisition channels, and the 'Save' and 'Reset' buttons, which respectively allow the acquisition to be saved or the settings to be reset to default conditions. There is also a drop-down box where it is possible to select the variable to be plotted in the graph between "*Acceleration*" and "*Displacement*", and a text box where the time interval that can be displayed in the graph is selected, which is the x-axis of the Cartesian plane where the signals are plotted. The second box, on the other hand, presents instant status indicators of the device, with LEDs dedicated to the state of the autofocus, progress bars indicating the quality of the signal being acquired, a long progress bar relating to the state of the measurement status, and finally indicators of the battery and Wi-Fi signal of the single handpieces.

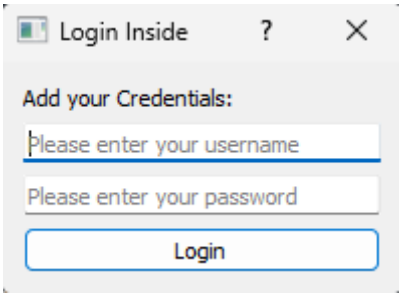

Figure 2.1: *Pop-up window of Login*

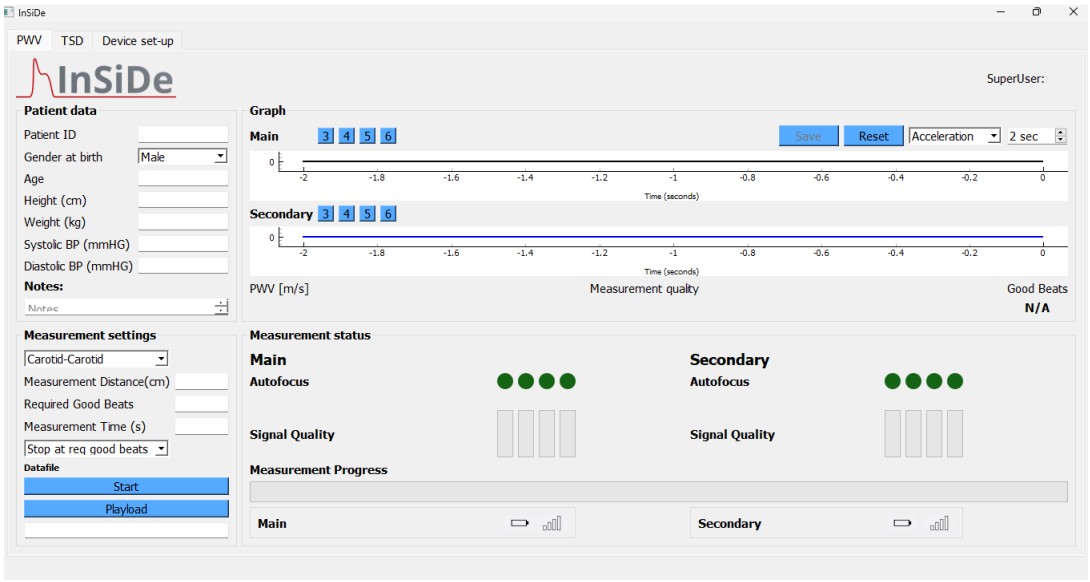

Figure 2.2: *First Tab of the Interface: PWV acquisition*

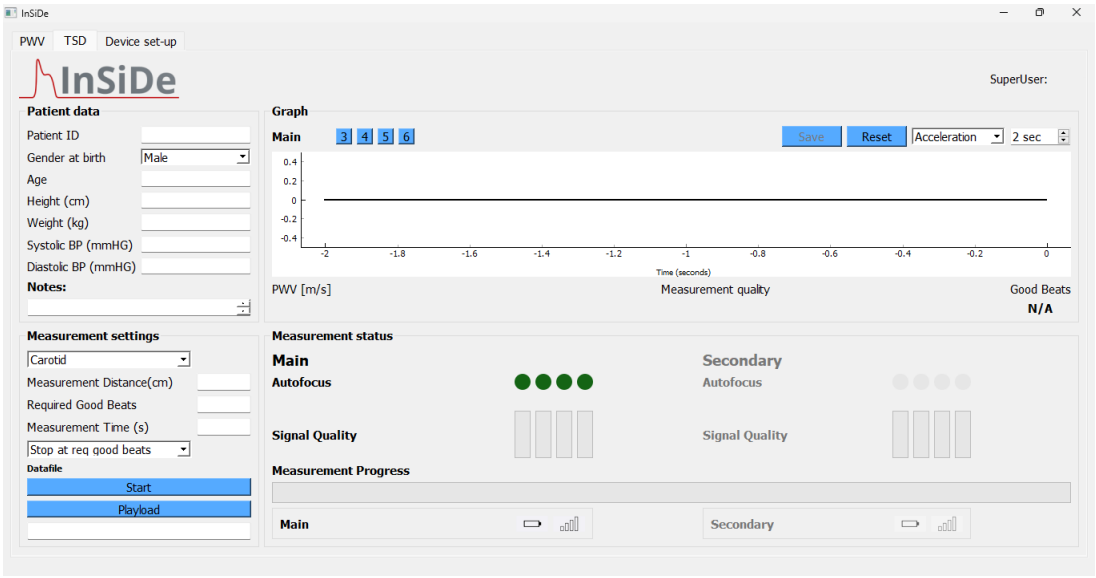

Figure 2.3: *Second tab of the Interface: TSD acquisition*

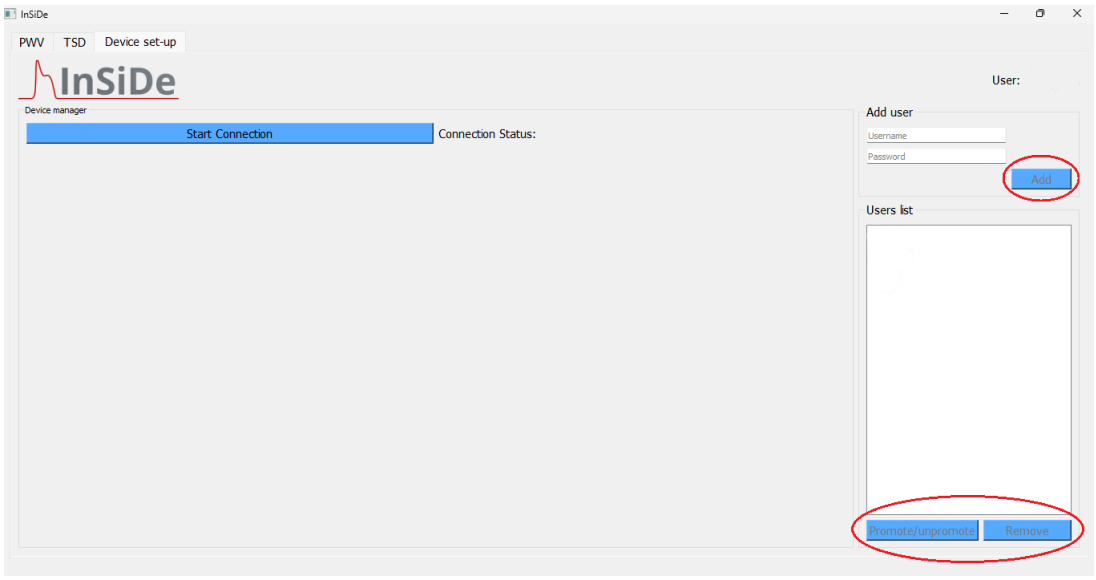

Figure 2.4: *Third tab as a User status: only connection buttons available, while the admin options (red circles) are not enabled*

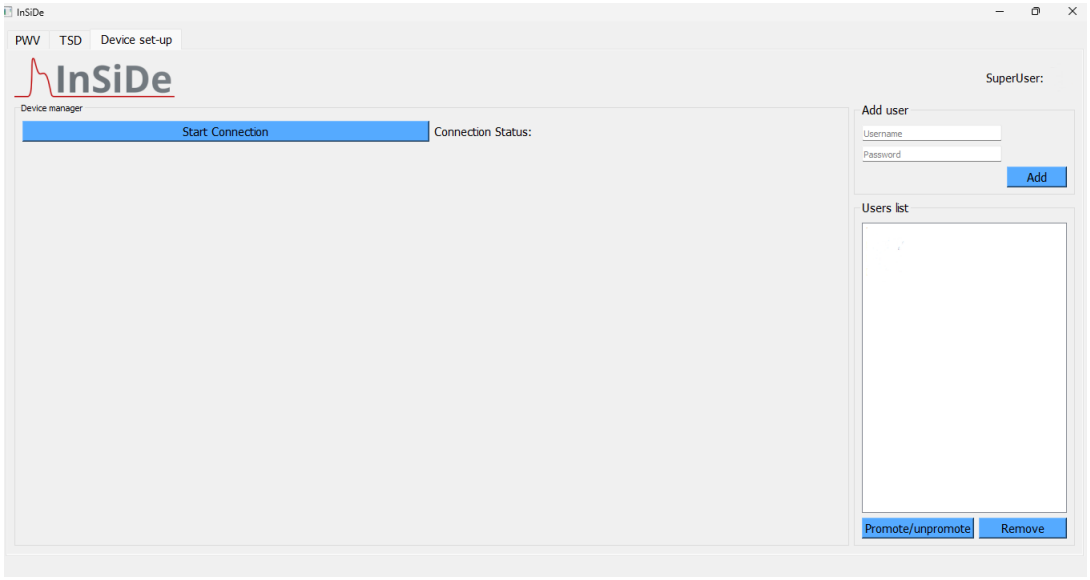

Figure 2.5: *Third tab of the Interface: Connection and user management with Super User status*

Before starting signal acquisition, the user must correctly enter the patient's clinical data in the appropriate text boxes. The data required as input from the Interface are as follows:

- **First Name and Last Name:** named by the Patient ID box
- **Gender:** selectable by a drop-down box with the options: "*Male*", "*Female*", "*Fluid*"
- **Age** (years)
- **Height** (centimetres)
- **Weight** (kilos)
- **Systolic and Diastolic Pressures** (mmHg)

The user can, at his or her preference, enter notes or comments to be taken into consideration later for any evaluations. Finally, you must select the type of measurement you want to take and enter the physiological parameters used such as the distance between the two acquisition points. The data entry group called 'Measurement settings' indeed contains a drop-down box where it is possible to: (1) Select the type of sampling chosen from the options '*Carotid-Carotid*', '*Femoral-Carotid*' and '*Chest-Carotid*'; (2) Entry data in the three text boxes relating respectively to the distance between the acquisition points (Measurement Distance), the number of beats required to consider the acquisition valid (Required Good Beats) and the maximum sampling measurement time (Measurement Time). The last two parameters are two minimum criteria to be reached in order to stop signal acquisition, and the user has the freedom to choose which of the two to select.
As soon as the data and parameter input phase is completed, signal acquisition begins with the two handpieces after pressing the "*Start*" button. Immediately before starting the sampling, the Interface performs checks on the data entered by the user, verifying whether each of them satisfies precise conditions. The operation of these checks and the external tests implemented to perform them will be explained in detail later.
For the acquisition to begin, the user must be sure to fill in the data on obligatory parameters such as the measurement distance, which is fundamental for the evaluation of the PWV, or the stopping conditions of the measurement, which can be selected from the "Required Good Beats" and "Measurement Time" parameters.If a compulsory parameter is not filled in, the software does not allow the beginning of the acquisition and the Interface warns the user of the missing parameter with a warning window (Figure 2.6, Figure 2.7). It is also possible, however, to carry out the measurement in the case of some missing data, which are not considered compulsory for evaluation purposes, in that case the Interface will warn the user

of these data deficits but will allow the choice of proceeding with the acquisition (Figure 2.8).

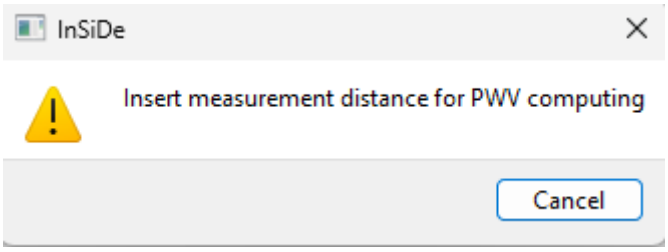

Figure 2.6: *Warning window in case of Measurement Distance missing data*

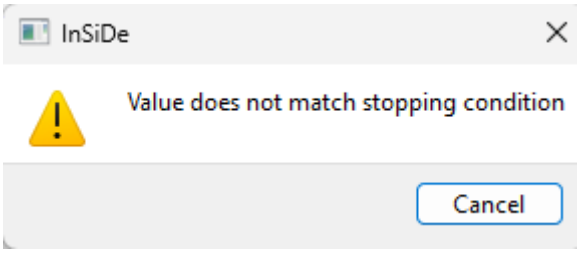

Figure 2.7: *Warning window in case of Stopping conditions missing data*

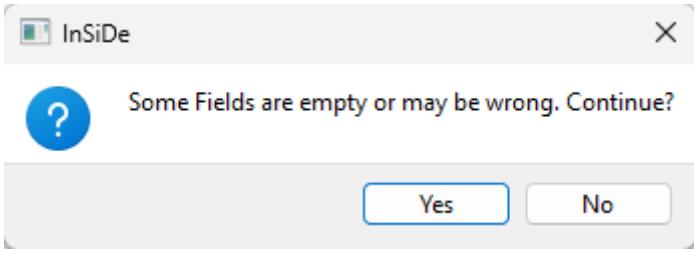

Figure 2.8: *Warning window in case of not compulsory missing data: signal acquisition is enabled*

During the measurement, the signal plot update functions are activated, for the correct display of the signal being acquired in the graph, together with the update functions of the measurement progress bars and signal quality. The latter are updated by a continuous score generated by the signal acquisition algorithm with a range from 0 to 100 and their colouring varies according to the input value of this score: Red for values: 0 < score < 25; Yellow for values: 25 <= score < 50; Green for values: score >= 50. At the end of the measurement the user has the option of either save the signal recording made or discard it to start a new one (Figure 2.9). During saving, in addition to the acquired signals, all clinical data of the

patient is stored in a predefined folder. These signals can be loaded and displayed later thanks to the "*Playload*" button, which retrieves the previously saved signals in "*.bin*" format. Once the recording is complete, the PWV estimation algorithm is launched with the initial user-defined physiological parameters and the acquired signals, which will provide the Pulse Transit Time intervals that are essential for calculating the PWV.

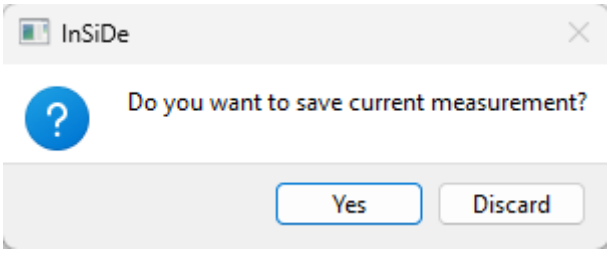

Figure 2.9: *Pop-up window with the signal saving/discard choice for the user*

2.2 Communication protocol

The communication protocol between the device's handset and the GUI establishes the presence of a bidirectional **TCP Socket** type connection, capable of exchanging data from the client (handset) and the server (GUI), and back in the opposite direction. The client, through this connection, is able to send data packages holding information such as the battery level of the handpieces, the quality of the signal being acquired and many others. The server will process the incoming data and perform the resulting operations depending on the information received. The handset communicates with the server with data packages of two different types according to the information it wants to transmit:

- Frame size of **1460** bytes divided as follows: Command bytes + 1459 Data bytes. The first byte, called Command byte, defines the type of information transmitted by the client (Buttons pressed, Autofocus completed, Errors raised etc.). This packet is sent immediately after being created in the client, without any waiting time

- Frame size of **3600** bytes for the DMA Data, divided as follows: Command bytes (0x85) + 31 Data bytes + 3568 DMA data bytes. DMA Data contain the signal acquisition from the single handpiece of the device, divided into 223 samples of each of the 4 channels, multiplied by a factor of 4 for the float format (Figure 2.10). Frames containing the DMA Data are sent by the client every 16ms

The data sampling frequency of the server is **13937.5Hz** (corresponding to 223 samples sent every 16ms). The received data is finally processed and displayed as a signal within the Interface.

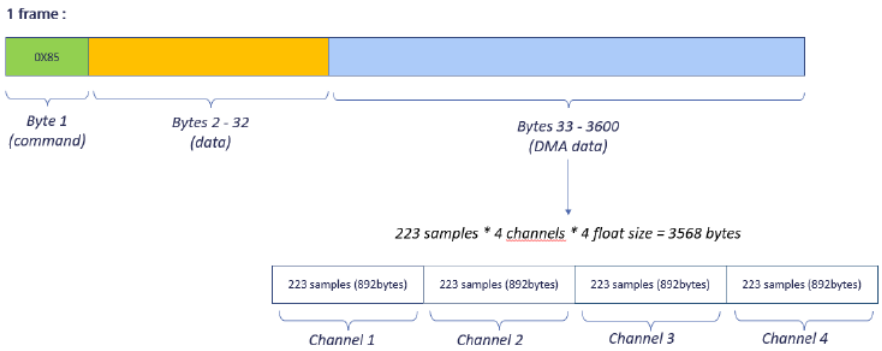

Figure 2.10: *Frame protocol for DMA Data reading*

Chapter 3

Materials and methods

In order to validate the device and verify its correct functioning, it is necessary to carry out various tests and evaluate the responses issued, providing adjustments in the event of errors generated. This work focuses on the alphanumeric checks performed by the Graphical User Interface when entering the patient's clinical data, which are indispensable for the correct classification of the individual patient in order to avoid any unforeseen errors during diagnosis or clinical evaluation by the doctor. The tests were initially carried out in manual mode, by entering precise data in the appropriate text boxes and waiting for a specific output from the Interface: no warning in the case of correctly entered data, warning in pop-up mode in the case of incorrect data.
In order to be able to validate the operation of the Interface, the volume of data to be used and tested is considerably high, so that a manual verification of these data by an operator would have required a considerable amount of time. The necessity to automatize these controls through algorithms using an extensive database was extremely relevant.

3.1 Unittest module

3.1.1 Unittest library framework

The implementation of an algorithm to carry out this series of tests automatically was achieved by using the Python library called *Unittest*[24]. This library provides a wide range of tools and functions necessary for performing tests. Unittest is equipped with a base class called '**TestCase**', which allows the creation of other subclasses where specific tests will be implemented independently from one another. The other main classes available in the library are:

- TestSuite: it is permitted to group tests under the same subclass and allows the operator to execute them at the same time

- TestLoader: it is possible to collect individual tests and groups of tests in a single code

- TestResult: it allows the receiving of information about which tests were successful and which ones showed errors

In this work, only the base class 'TestCase' was used, since the tests to be implemented had to be carried out independently.

3.1.2 Unitest work adaption

In this work, each test is an independent unit within the Unittest class and it is named after the common word '*test*', a convention that notifies the operator which subclasses represent a test and which do not. Their operation is based on the **assertion** method, several different types are provided by the base class (Table 3.1), where the output generated by the individual test function is compared with the expected result, depending on the input data that was used, or check whether the input has generated exceptions, warnings and log messages.

Method	Checks that
assertEqual(a, b)	a == b
assertNotEqual(a, b)	a != b
assertTrue(x)	bool(x) is True
assertFalse(x)	bool(x) is False
assertIs(a, b)	a is b
assertIsNot(a, b)	a is not b
assertIsNone(x)	x is None
assertIsNotNone(x)	x is not None
assertIn(a, b)	a in b
assertNotIn(a, b)	a not in b
assertIsInstance(a, b)	isinstance(a, b)
assertNotIsInstance(a, b)	not isinstance(a, b)

Table 3.1: *Assert methods in Unittest library*

In the event of a match between expected result and received output, the test will be classified as '*Passed*'. Otherwise, an Exception warning will be generated and the test will be classified as '*Failure*'. In the event of a considerable number of tests to be executed, Unittest allows variables to be defined, which will be recalled automatically before each individual test, through the '**setUp**' method, located at the beginning of the algorithm. After the individual tests have been set up, the input data to be entered are defined in order to verify the generated response. The algorithm can then also be run from the command line, by defining the file path first and then executing the file. It is also possible to specify the necessity of more details on the execution of the tests with the command '-v', which allows the prompt to compile the information of each individual test and its result on the screen.

3.2 Database

The main aim of this work was to automatise the alphanumerical controls carried out by the GUI as part of the InSiDe device project. The tests implemented were created taking as a reference a report drawn up previously, which contained tables of all the tests to be performed manually, each associated with an identification code and a description of all the steps to be followed.

3.2.1 Data organization

In total, there were 11 checks relating to the input of the patient's data in the report provided, expanding the count to the individual steps of each test, the number reached 45. However, for many of these steps, it was impossible to automatise them inside an algorithm for the following reasons: (1) for each insertable parameter, the first control step is a visual verification by the operator that the Interface provides the text box or drop-down box necessary for the successful accomplishment of the data input; (2) checks on the update of the indexes related to parameters available from drop-down boxes or on the correct storage of variables would have exceedingly modified the functionality of the main GUI code.

3.2.2 Tests database structure

The final number of controls configured within the algorithm is 25, each relating to a single step in 14 macrocategories, each representing the patient parameters tested. Of these 25 implemented controls, 18 are the automatic version of the tests in the previously mentioned report drawn up during the 'InSiDe' project, which were originally planned to be carried out manually. These in fact maintained the main title of the individual test as the identification code assigned to it inside the report,

just as the representative number of the automated step is the same as that of the report. The remaining 7 controls are checks performed by the GUI of correctly entered values in order to ensure that there were no incorrectly raised warnings even in the case of regularly entered data. All steps contain a precise description of the type of control performed (e.g. Input values with more than three digits) and an accurate range of values used (Table 3.2).For each test performed by the algorithm, all values entered into the individual function are saved and copied into a list dedicated to the individual step, named in the following ways according to the control (Table 3.3):

- List_+ Parameter tested name (e.g. *List _Ages*)
- List_+ Test identification code_+ Number of the step performed (e.g. *List _INSI511 _2*)

These lists of values are created in the algorithm and saved as external files in the "*.json*" format, so that they can be easily called up by external codes and any incorrect or inconsistent values in the test can be checked as quickly as possible.

Range type	**Minimum Value**	**Step value**	**Maximum Value**
Age parameter	0	1	120
Height/Weight parameter	0.5	0.5	220
Sys/Dia pressure	100/60	1	300/260
Wrong age values	121	1	999
Wrong height/weight values	221	1	999
Wrong pressure values	301	1	999
Values with min. 4 digits	1000	1	10000

Table 3.2: *Values ranges applied for the tests*

Test Title	Step	Step description	Values list
Test Patient name	No	Strings of fixed length, with alphabetic uppercase, lowercase and space characters	List_names
Test Ages		Input values for the Age input box	List_ages
Test INSI-511	2	Strings of fixed length with characters different from digits	List_INSI511_2
	3	Input values with more than three digits	List_INSI511_3
	5	Input values greater than 120 for the Age input box	List_INSI511_5
Test Heights		Input values for the Height input box	List_heights
Test INSI-512	2	Strings of fixed length with characters different from digits	List_INSI512_2
	3	Input values with more than three digits	List_INSI512_3
	5	Input values greater than 220 for the Height input box	List_INSI512_5
Test Weights		Input values for the Weight input box	List_weights
Test INSI-513	2	Strings of fixed length with characters different from digits	List_INSI513_2
	3	Input values with more than three digits	List_INSI513_3
	5	Input values greater than 220 for the Weight input box	List_INSI513_5
Test Systolic pressure		Input values for the Systolic pressure input box	List_systolic
Test Diastolic pressure		Input values for the Diastolic pressure input box	List_diastolic
Test Pressures		Systolic pressure values greater than Diastolic pressure values	List_pressures
Test INSI-514	2	Strings of fixed length with characters different from digits	List_INSI514_2
	3	Strings of fixed length with characters different from digits	List_INSI514_3
	5	Input values with more than three digits	List_INSI514_5
	8	Diastolic values greater than systolic values	List_INSI514_8
	9	Input values greater than 300 for the Height input box	List_INSI514_9
	10	Input values greater than 300 for the Height input box	List_INSI514_10
Test INSI-517	2	Strings of fixed length with characters different from digits	List_INSI517_2
Test INSI-518	2	Strings of fixed length with characters different from digits	List_INSI518_2
Test INSI-519	2	Strings of fixed length with characters different from digits	List_INSI519_2

Table 3.3: *List of the implemented tests with name, number of each step, description of the steps and name of the external list for values storage*

3.3 Automated tests on patient data

3.3.1 Tests algorithm configuration

The algorithm for automated tests was developed in two separate parts with different functionalities:

1. The main code is responsible for running the tests via the Python library '*Unittest*', which can also be called from prompts outside the development environment

2. The secondary code containing validation functions for values it receives as input, generating output to be sent to the main code

Within the main script, the organisation is dictated by the rules of the library, each test is implemented with its own function independent of the others, each named according to the parameter being tested or with the identification code, always making explicit the step being performed. All ranges containing the values to be used in the tests are initialised by the '*SetUp*' function, which is called by the algorithm each time one of the tests is executed.In the '*SetUp*' function, in addition to the ranges of values to be used in the tests, there is also a variable called '*iter_num*', which defines the number of times each value in the chosen range is repeated and entered as input in the validation functions. It was chosen to repeat the input action to improve the accuracy of the output generated by the validation functions. Each test in the main code, at runtime, calls up a specific range of values to be used. Each of these values is sent as input to the related validation function contained in the secondary code, simulating manual input by a user.
Consequently, in the secondary code, there are functions specialised in the validation of the values received as input. They are divided according to the parameter tested, with a total of 10 functions. The purpose of these functions is the checking of each individual value received as input, comparing it with the validity range of the parameter, with the subsequent generation of a feedback output that is sent to the test blocks of the main code for the final evaluation of success or failure of the test. The functions are named according to the parameter considered by the validation in this way: Validate_+ name of the tested parameter (e.g. *Validate_age*). The check on the individual value is functionalized to the parameter being considered, so the following differentiation of validity criteria is configured according to the type of parameter:

- In the case of numeric type parameters admitting only **integer** numbers (e.g. Age, Systolic and Diastolic Pressure), the criteria for validating the value received as input are:

    1. The value must consist of digits only, no other characters are allowed

2. The value must not exceed the maximum number of digits allowed
3. The value must be within the validity range of the parameter

- In the case of numeric type parameters which also admit the **decimal** digit (e.g. Height and Weight), the criteria for validating the value received as input are the same as mentioned above for parameters with integer numbers, however, the format of the value admitted as input can only be of two types:

  1. Integer number (e.g. 99)
  2. Digit - Comma - Digits (e.g. 99.5)

  This restriction of permitted formats was created to exclude all non-numeric characters entered as input in parameters requiring digits, while admitting the character '*Comma*', which was chosen as the only character separating integer digits from decimal digits.

- In the case of **text** type parameters (e.g. Patient Name), the only validation condition for the value received as input is the presence of a variable in text string format. The value is not considered valid if it consists of an empty string.

Once the control criteria of each validation function have been defined, they generate a specific output after comparing the value received as input with the conditions explained above, sending it as feedback to the main code, specifically to the single test that had sent to validate the value.The outputs generated are of different types depending on the conditions that were passed by the input value, attempting to simulate the feedback from the GUI at the moment of entering the data:

1. Functions validate the value and send it back to the tests in a variable format to be compared with the expected output

2. Functions generate a warning message explaining the error encountered when validating the value received as input (e.g. "Too many digits", "Characters different from digits") and send it back to the test functions for comparison with the expected output

The '*Unittest*' library uses assertion methods to check and report failures as passing test criteria. They compare an output received from an external source with a previously programmed expected output and report the result of this comparison, depending on the predefined criterion of the individual method chosen. Each Test Case, having called up a specific range of values to be used, sends as input to the validation functions every single value of the range, repeating this operation for a specific number of times according to the iteration variable created at the beginning of the algorithm, waiting to receive feedback from the functions after the

value validation operations.
Each Test Case has an expected output initialised, which will be compared with the incoming output from the validation functions, via the assertion methods. Two different types of assertion methods are used:

1. The expected output is a validated value, to be compared with all values in the range used in the relevant test. If the expected value is matched against one of the values in the range, the test is considered passed.

2. The expected output is a warning message containing the error caused by the input value. If the match between the expected output and the output generated by the validation functions is positive, the test is considered passed.

At the beginning of the test session, the algorithm executes all the Test Cases individually, displaying the result of each one on the screen with the words '*Passed*' or '*Failed*', depending on the success or failure of the test (Figure 3.1, Figure 3.2). If a test case is detected as a failure, the algorithm indicates on the screen which test has failed and, in the section below the results, reports an explanation of the failure (Figure 3.3), thus a mismatch between the output generated by the validation functions and the expected output of the test. It also reports which input values to the functions produced an output different from the expected one, immediately identifying the incorrect values within the validation range.

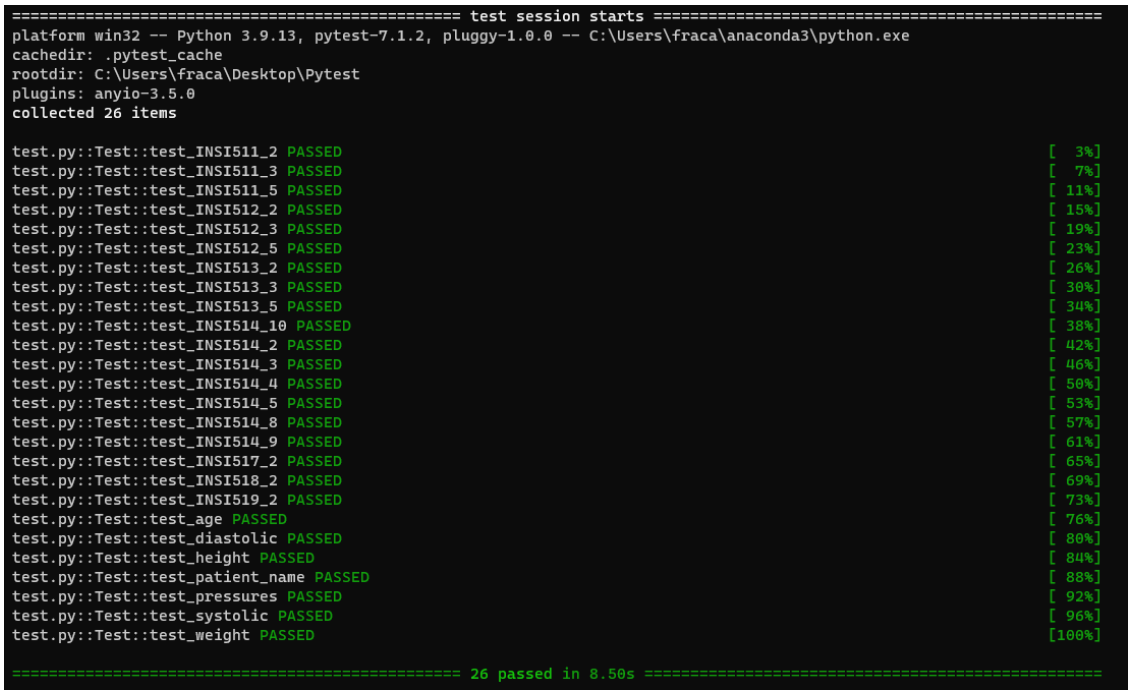

Figure 3.1: *Example of prompt view in case of all tests successfully performed*

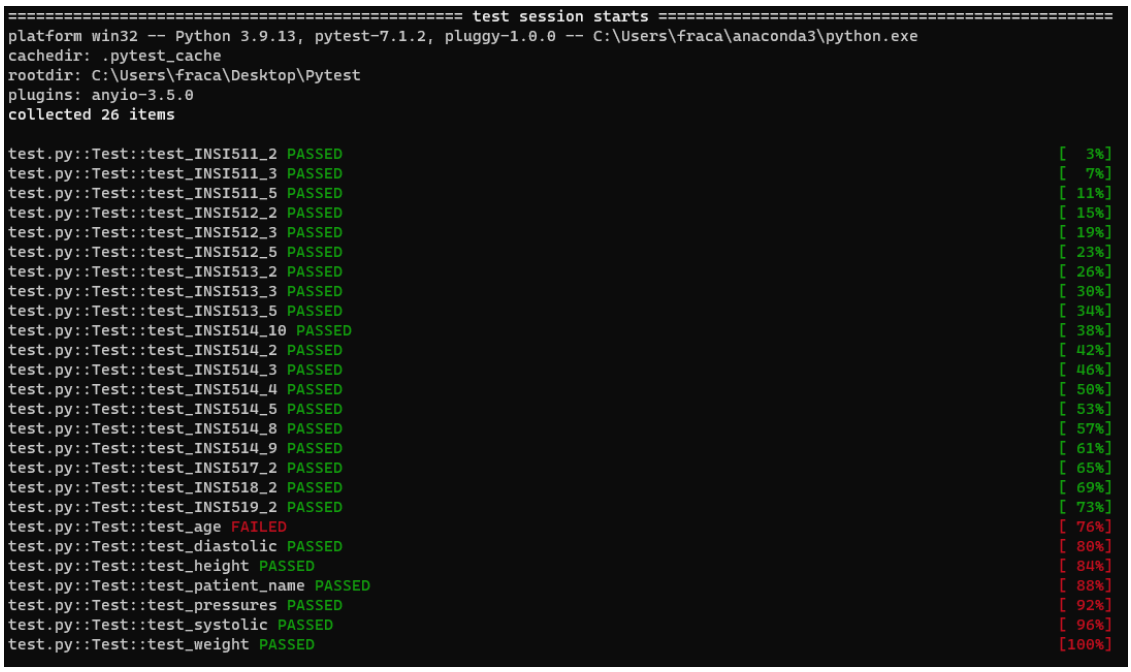

Figure 3.2: *Example of prompt view in case of one test failed*

```
=================================================== FAILURES ===================================================
_____________________________________________________ Test.test_age _____________________________________________________

self = <test.Test testMethod=test_age>

    def test_age(self):
        list_ages = []

        for i in islice(cycle(self.age_range), len(self.age_range) * self.iter_num) :

            list_ages.append(str(i))

>           assert update_data.validate_age(self, i) in self.age_range

test.py:77:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

self = <test.Test testMethod=test_age>, value = 121

    def validate_age(self,value):
        self.age = value
        self.age_range = np.arange(0,121,1)

        if str(self.age).isdigit() == True:
            if self.age not in self.age_range and len(str(self.age)) <= 3:
>               raise ValueError('Some fields are empty or maybe wrong. Continue?')
E               ValueError: Some fields are empty or maybe wrong. Continue?

update_data.py:27: ValueError
============================================= short test summary info ==============================================
FAILED test.py::Test::test_age - ValueError: Some fields are empty or maybe wrong. Continue?
========================================== 1 failed, 25 passed in 7.03s ============================================
```

Figure 3.3: *Test failure explanation by the algorithm, defining which incorrect value/values has generated the wrong output*

3.4 PWV evaluation algorithm transposition

The PWV evaluation algorithm was previously developed by studies external to this work in the Matlab environment. Therefore, in order to be able to integrate its calculation function within the GUI, whose code is entirely in Python, a transposition of the algorithm between the two programming languages was required. The main points of the algorithm and the solutions adopted to recreate them in the new environment will be explained below [26].

3.4.1 PWV evaluation algorithm configuration

The PWV calculation algorithm was developed using a database of LDV acquisitions performed on a total of 100 patients with the CARDIS device at the Hopital Europen Georges Pompidou (HEGP) in Paris, France. [29].
The time duration of each acquisition is 20 seconds, divided into 12 tracks representing the individual channels of the device's handpieces, and the sampling frequency is 10 kHz. At the end of the recording, the LDV displacement signals are saved, which will be used for the calculation of acceleration, which provides clear peaks within the pattern, easily detectable by a function and thus used in the final PWV calculation. Starting with the displacement signals, a sixth-order, low-pass IIR filter is applied, followed by a gradient operation for two times, until acceleration

signals are obtained. After each applied gradient, the filter is reapplied to the result of the operation, bringing the total number of filterings to 3.

The acceleration signals of each channel obtained are resampled at a frequency of 1 kHz and can be analysed for peaks identification using the **template matching** technique. It uses an approach involving the translation of a pattern over the entire image, followed by the calculation of the correlation between the pattern and the window covering the image, thanks to the use of two-dimensional convolution. The templates required for this method to work were created by Seoni et al., during a study that also validated this technique for peak identification [8]. The length of the carotid templates provided is 200ms, while the femoral ones are 400ms. Thus, the template matching method is implemented between the acceleration signals and the constructed template, calculating the correlation coefficients and the subsequent threshold, which is necessary for the identification of signal peaks.Thanks to the threshold obtained from the cross-correlation, signal peaks above it are identified and the indices of the identified points are saved within a array. These will represent the number of beats recognised by the template matching. For the PTT calculation, the vector with the highest number of recognised beats among the 12 channels is chosen and will be taken as the reference for the peak indices.

The Δt between two successive peaks is calculated after their recognition within a 200ms time window. If a femoral peak is detected after a carotid peak within the given time period, they are identified as a pair of the same beat and the delay period between them is evaluated as the PTT. The calculated time intervals in each channel are saved and the median between them is evaluated to obtain an acquisition parameter relative to the individual carotid channel compared with all femoral channels.
The PWV is computed by the ratio of the space travelled, evaluated as the length of the carotid-femoral segment scaled by a factor of 0.8, and all previously calculated medians of the PTT. This will result in median PWV values of each carotid channel with all femoral channels. The final output provided by the algorithm consists of a PTT value and a PWV value, both obtained from the median of the PTT and PWV. In addition, the standard deviation between the median PTT and PWV (PTT STDV and PWV STDV) values of the channels is calculated and integrated into the generated output.

The contribution of this work, as mentioned, was the transposition of this algorithm into the Python environment, so as to be able to integrate the PWV calculation function within the functionality of the GUI of the InSiDe device. Python libraries were used that could replicate the work of the functions of the Matlab environment in the same way, in particular the libraries *'Numpy'* and *'Scipy.signal'* [27, 28].

The former provided the appropriate functions for building the arrays and matrices hosting the variables that were being calculated, as well as useful commands for computing the medians of the PTT and PTT values found and the gradients for that of the acceleration signals. The second was fundamental for carrying out the correlation operations within the template matching and filtering of the LDV displacement signals, with functions for constructing and application of the filter.

Chapter 4

Results and discussions

4.1 Results

The evaluation of the efficiency of the algorithm implemented in the Python environment was carried out by comparing the acceleration signals obtained from the filtering and derivation operations from the LDV displacement signals and the output values of Pulse Wave Velocity, its standard deviation and the number of peaks identified by the two algorithms. Two samples taken by the CARDIS device during a previously mentioned study were used for verification purposes [29]. They were classified in terms of signal quality on the basis of a visual quality score assessed by an experienced operator during the same study: respectively, the two signals were classified as '*Excellent*' and '*Good*'.
A visual comparison was then performed by plotting both acceleration signals simultaneously, one obtained in the Matlab code and the other obtained in the Python code, to check whether the two patterns coincided as expected (Figure 4.1, Figure 4.2). Finally, the output values from the algorithms of the two different environments were compared: Pulse Wave Velocity (PWV), Standard Deviation of PWV (STDV PWV), Total Number of Recognised Peaks (PKS) (Table 4.1).

Output parameter	**Matlab**	**Python**
Excellent signal PWV	4.758974358974359	4.758974358974359
Excellent signal STDV PWV	0.085766322226529	0.085766322226529
Good signal PWV	4.595744680851064	4.595744680851064
Good signal STDV PWV	0.595061840205240	0.595061840205240
Excellent signal PKS	236	240
Good signal PKS	268	268

Table 4.1: *Output parameters from Matlab and Python algorithm: Pulse Wave Velocity, Standard Deviation of PWV and Number of identified Peaks*

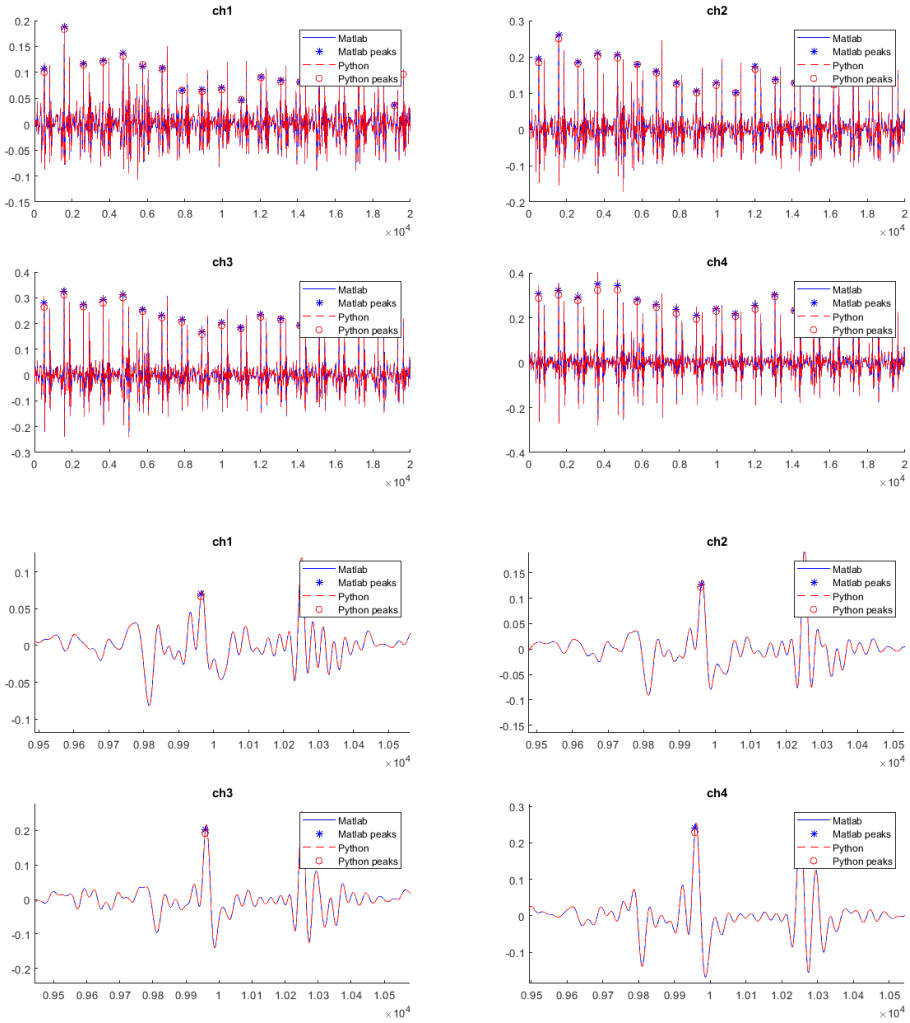

Figure 4.1: *Comparison between acceleration signals from Matlab (blue) and Python (red) and detail of one the peaks recognised by the two algorithm on 'Excellent' classified LDV signal*

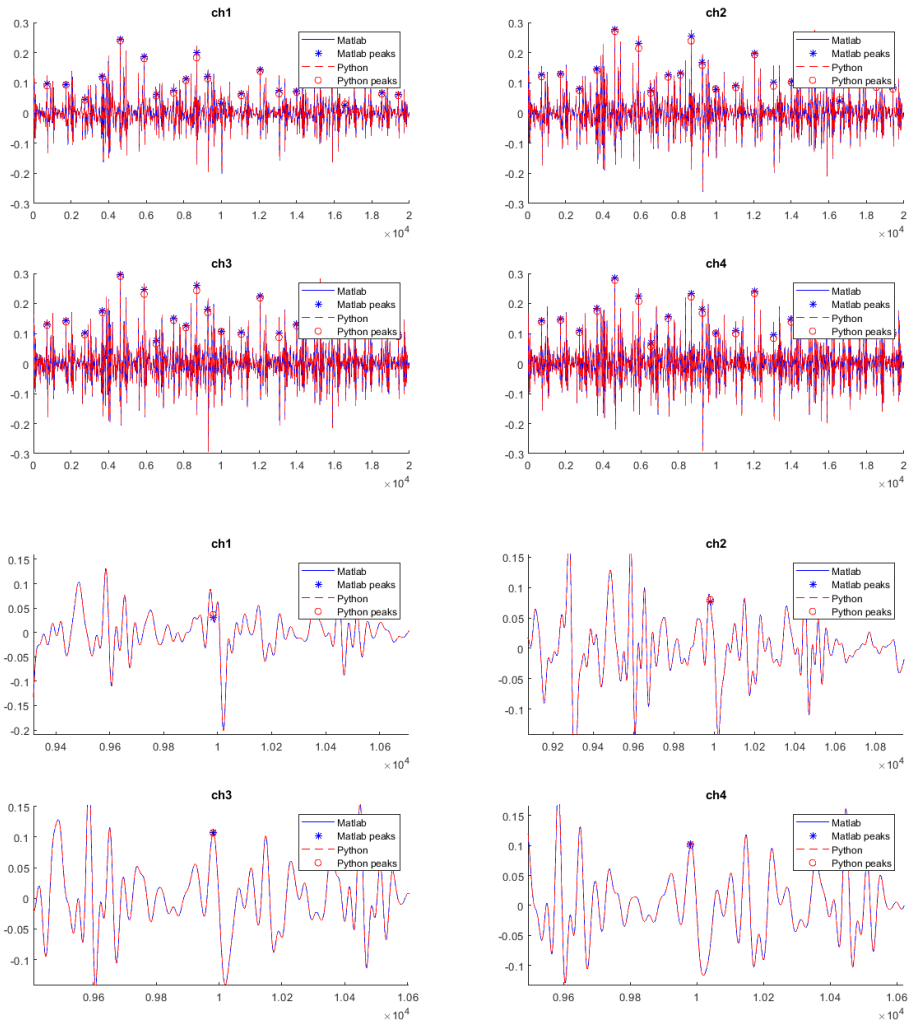

Figure 4.2: *Comparison between acceleration signals from Matlab (blue) and Python (red) and detail of one the peaks recognised by the two algorithm on 'Good' classified LDV signal*

4.2 Discussions for future developments

As shown in the graphs above, the pattern of the acceleration signal calculated using the Python algorithm is exactly the **same** as the one calculated using the Matlab code, confirming the efficiency and the accuracy of the transposition of the algorithm between the two programming environments.The filtering and gradient functions, despite the same initial coefficients and parameters between the two algorithms, resulted in the same acceleration signal vector, even though the programming language is different.
The results generated by the algorithms in the respective outputs also confirmed the transposition performance. The Pulse Wave Velocity values between the two algorithms are **identical**, as are the standard deviations calculated with the excellent and good quality signals. However, they did not affect the computation of the PWV values because all of them were peaks found in the carotid signal and, since they either did not have a corresponding peak in the same beat in the femoral signal or were too close to a subsequent peak in the carotid signal, they were not included in the spectral matching and PTT computation.This mismatch resulted in a difference in peak identification of only 1.7%, which decreased further on signals of considerably longer duration.

The automation of tests regarding the patient data to be entered as input into the Interface was presented within the InSiDe project and provided to them to verify its efficiency. A possible improvement of the test algorithm would be the greater involvement of the Python library '*Unittest*', which has numerous functions that are useful for the more specific implementation of test cases and allow a more detailed configuration of the algorithm to be implemented. For time reasons, it was decided to use only the base class of the library, which in any case provided the necessary tools for the realisation of the work, however, a more detailed study of the functions of this library would allow a considerable improvement in the building of testing codes. In addition, for the computational costs, it was decided to maintain a restricted database to validate the testing algorithm. Computational power tools would increase the chances of testing the algorithm on a more robust database, increasing the effectiveness of the testing algorithm. Finally, all the value ranges used within the test functions were chosen by taking them from the report developed by the InSiDe project and refined with personal choices, so a confirmation or an opinion from medical experts on them would allow the validation of the values used and a certainty for the control configuration in the medical device Interface developed during the **InSiDe** project.

Bibliography

[1] *World Health Organization*. URL: https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1.

[2] Connie W. Tsao et al. "Heart disease and stroke Statistics 2023 update: A report from the American Heart Association". In: *Circulation* 147.8 (2023). DOI: 10.1161/cir.0000000000001123.

[3] S. Laurent et al. "Expert consensus document on arterial stiffness: Methodological issues and clinical applications". In: *European Heart Journal* 27.21 (2006), pp. 2588–2605. DOI: 10.1093/eurheartj/ehl254.

[4] Jordi Calabia et al. "Doppler ultrasound in the measurement of Pulse Wave Velocity: Agreement with the Complior Method". In: *Cardiovascular Ultrasound* 9.1 (2011). DOI: 10.1186/1476-7120-9-13.

[5] Mark Butlin and Ahmad Qasem. "Large artery stiffness assessment using SPHYGMOCOR technology". In: *Pulse* 4.4 (2016), pp. 180–192. DOI: 10.1159/000452448.

[6] *Lasser Doppler Vibrometer*. URL: https://en.wikipedia.org/wiki/Laser_Doppler_vibrometer.

[7] *Cardis project*. URL: https://cordis.europa.eu/project/id/644798.

[8] Silvia Seoni et al. "Template matching and matrix profile for signal quality assessment of carotid and femoral laser Doppler vibrometer signals". In: *Frontiers in Physiology* 12 (2022). DOI: 10.3389/fphys.2021.775052.

[9] Yanlu Li et al. "Silicon Photonics-based laser Doppler vibrometer array for carotid-femoral pulse wave velocity (PWV) measurement". In: *Biomedical Optics Express* 11.7 (2020), p. 3913. DOI: 10.1364/boe.394921.

[10] *Inside site*. URL: https://www.inside-h2020.eu/.

[11] Salil Sethi et al. "Aortic stiffness: Pathophysiology, clinical implications, and approach to treatment". In: *Integrated Blood Pressure Control* (2014), p. 29. DOI: 10.2147/ibpc.s59535.

[12] Pucci G. Schillaci G. *La rigiditá arteriosa:quali applicazioni pratiche?* URL: http://www.sisa.it/upload/GIA_2010_n0_3.pdf.

[13] M VYAS et al. "Augmentation index and central aortic stiffness in middle-aged to elderly individuals". In: *American Journal of Hypertension* 20.6 (2007), pp. 642–647. DOI: 10.1016/j.amjhyper.2007.01.008.

[14] Ian B. Wilkinson et al. "Artery society guidelines for validation of non-invasive haemodynamic measurement devices: Part 1, Arterial Pulse Wave Velocity". In: *Artery Research* 4.2 (2010), p. 34. DOI: 10.1016/j.artres.2010.03.001.

[15] *Moens-Korteweg equation*. URL: https://en.wikipedia.org/wiki/Moens%E2%80%93Korteweg_equation.

[16] *Pulse Wave Velocity*. URL: https://en.wikipedia.org/wiki/Pulse_wave_velocity.

[17] Sandrine C. Millasseau et al. "Evaluation of carotid-femoral pulse wave velocity". In: *Hypertension* 45.2 (2005), pp. 222–226. DOI: 10.1161/01.hyp.0000154229.97341.d2.

[18] Marek W Rajzer et al. "Comparison of aortic pulse wave velocity measured by three techniques: Complior, SPHYGMOCOR and Arteriograph". In: *Journal of Hypertension* 26.10 (2008), pp. 2001–2007. DOI: 10.1097/hjh.0b013e32830a4a25.

[19] Thomas Weber et al. "Noninvasive determination of carotid-femoral pulse wave velocity depends critically on assessment of travel distance: A comparison with invasive measurement". In: *Journal of Hypertension* 27.8 (2009), pp. 1624–1630. DOI: 10.1097/hjh.0b013e32832cb04e.

[20] Samir Sulemane et al. "Subclinical markers of cardiovascular disease predict adverse outcomes in chronic kidney disease patients with normal left ventricular ejection fraction". In: *The International Journal of Cardiovascular Imaging* 33.5 (2017), pp. 687–698. DOI: 10.1007/s10554-016-1059-x.

[21] Precisi S. *Valutazione dell'elasticitá carotidea durante esercizio fisico*.

[22] *QtDesigner*. URL: https://doc.qt.io/qt-6/qtdesigner-manual.html.

[23] *PyQt5 library*. URL: https://pypi.org/project/PyQt5/.

[24] *Unittest library*. URL: https://docs.python.org/3/library/unittest.html.

[25] *Matlab*. URL: https://it.mathworks.com/products/matlab.html?s_tid=hp_products_matlab.

[26] Tafuri E. *Valutazione real-time della qualitá dei segnali LDV e calcolo della PWV*. 2022.

[27] *Numpy library*. URL: https://numpy.org/.

[28] *Scipy Signal library*. URL: https://docs.scipy.org/doc/scipy/reference/signal.html.

[29] L. Marais et al. “Measurement of aortic stiffness by laser Doppler vibrometry”. In: *Journal of Hypertension* 37 (2019). DOI: 10.1097/01.hjh.0000570292.62996.dd.