Politecnico di Torino

Politecnico di Torino

Master of Science in Biomedical Engineering
A.Y. 2022/2023
March 2023

Enhancing the research toward wearable solutions for continuous noninvasive blood pressure monitoring

Supervisors:
Professor Carla Fabiana Chiasserini
Professor Guido Pagana

Candidate:
Francesca Boschi

Abstract

Wearable devices are potential power instruments for facing major health-related challenges such as the aging population, chronic diseases, and hospital service delivery. While wearables are nowadays widely used for monitoring physical activity and active life, also through some physiological parameters, their use in the clinical environment remains limited and challenging. Among the many reasons for the limited use of wearable solutions in the clinical field is the lack of a complete and updated regulatory framework, including specific guidelines to establish the accuracy of wearables and the ownership of the cost burden. Not to mention the acceptability of such devices among patients and healthcare providers in terms of comfort and trust. The European Horizon 2020-funded SINTEC project (Soft Intelligence Epidermal Communication Platform) aims to develop soft, sticky, and stretchable devices for specific use in the clinical field, particularly for continuous blood pressure monitoring (BP). Monitoring BP is essential for hypertension diagnosis and monitoring both in a clinical and home environment, especially in a scenario where deaths due to hypertension are increasing in high-income countries, according to WHO (World Health Organization). Different methods have been developed for BP monitoring including, invasive, noninvasive, intermittent, continuous, and cuffless techniques but still, a noninvasive technique for continuous BP monitoring remains a challenge. This thesis works aims to add major robustness to motion artifacts (MA) to the algorithm developed for the SINTEC project, exploiting the electrocardiographic (ECG) and photoplethysmographic (PPG) signals to extrapolate BP measurements through the Machine Learning (ML) linear regression technique. The novelty proposed in this thesis work includes the acquisition of the accelerometer signals, adding an extra filtering possibility through a Least Mean Square (LMS) adaptive filter, and extra features cleaning through thresholding based on the accelerometer signals. Specifically, a combination of both techniques has allowed for an average of 25 % reduction of MAE (Mean Absolute Error) in highly corrupted signals, allowing in some cases to obtain an acceptable low error according to the AAMI/ISO/ESH guidelines from acquisitions that gave nonacceptable MAE. For testing the validity of these techniques, signals have been acquired through SHIMMER (Sensing Health with Intelligence, Modularity, Mobility, and Experimental Reusability) devices in healthy subjects in different conditions, such as sitting still

on a chair, standing up and down from the chair, doing small walking, and doing small hand gestures. Furthermore, this thesis work aims to enhance the useability of this new BP monitoring technique by proponing a GUI (Graphical User Interface) for better determinate input parameters necessary for a good BP estimation, to enhance the accessibility to this technology also to users without programming background and allowing the transferability of the system for signals that will be recorded with the intended SINTEC final devices. Finally, the GUI speeds up the research process for the best ML coefficients needed in the algorithm. In conclusion, this thesis work aims to contribute to the research for a wearable solution for BP monitoring that is accurate, highly comfortable for the patient, easy to set up, and use for both patients and physicians to be officially recognized in a clinical environment.

Contents

Acronims table

Table 1: Acronyms

Acronym	**Meaning**
ABPM	Ambulatory BP monitoring
APIs	Application Programming Interfaces
APG	acceleration photoplethysmogram
ARS	Accelerometer reference signal
AV	Atrio ventricular
BP	Blood Pressure
BTLE	low-energy Bluetooth
CVD	Cardiovascular disease
DBP	Diastolic blood pressure
DoF	Degree of Freedom
ECG	electrocardiogram
Fat-IBC	Fat Intra Body Communication
GUI	Graphical User Interface
HBPM	Home BP monitoring
HR	Heart Rate
HRV	Heart rate variability
ICU	Intensive Care Unit
IMU	inertial measurement unit
IoT	Internet of Things
IR-LEDs	infrared light emitting diodes
LMS	Least Mean Squares
MA	Motion Artifact
ML	Machine learning
MLR	Multivariate Linear Regression
PCB	Printed Circuit Board
PPG	Photoplethysmography
PTT	Pulse transmit time
PW	Pulse wave

Table 1 – Acronyms

Acronym	**Meaning**
PWV	Pulse Wave Velocity
SBP	Systolic blood pressure
UML	Unified Modeling Language
WHO	World Health Organization

Chapter 1

Introduction

This thesis work is based on improving the SINTEC medical device algorithm by enlarging the testing database, proposing a reduction strategy for motion artifacts (MA), and developing a graphical user interface (GUI). SINTEC – Soft Intelligence epidermal communication platform is a Horizon 2020-funded project [1] that will provide soft, sticky, and stretchable sensor patches for multiple usages, such as active life and clinical monitoring [2]. This thesis focuses only on the clinical aspect, which is the monitoring of systolic blood pressure (SBP), diastolic blood pressure (DBP), and heart rate (HR) of the device wearer. This work has been carried out at LINKS Foundation, which has been operating for about 20 years at national and international levels and aims to promote, lead and enhance innovation processes through research projects with strong innovative potential that can create an impact on the productive and public sector, dealing with an international context [3].

1.1 Wearables in the clinical field

The increase in the aging population, chronic diseases, and hospitalization costs are just a few of the many healthcare challenges for future decades. Wearable healthcare solutions such as advanced sensors, new technology exploiting new communication platforms, and the Internet of Things (IoT) are potential solutions for facing these main challenges [4]. Nowadays wearables devices are largely used in sports and physical activity monitoring, whereas their clinical use is still limited and remains a challenge. However, it is a matter of fact that there has been a growth in the availability of wearables-based analytic platforms with the potential of enhancing the quality and accessibility of healthcare everywhere from hospital intensive care units (ICUs) to in-home chronic disease management [5] [6]. The use of wearables in hospitals could enhance patient monitoring, adding major comfort to the patient himself and facilitating mobility between hospital wards. For example, the use of multimodal sensors with tailored alarm systems in Intensive Care Units

(ICUs) can ameliorate continuous patient monitoring. Wearables integrating machine learning and artificial intelligence systems are a powerful tool for enhancing the development of predictive models and diagnostics [4]. On the market, there are wearables proposed also for different health-related applications outside of hospitals like cardiovascular and gastrointestinal monitoring, neurology, and mental health, maternal, pre - and neonatal care, pulmonary health, and environmental exposures [6]. The main challenges for the success of wearables in the clinical field include the integration in a complex already existing system of care and the a priori target disease definition [5]. Acceptance from both patients and physicians is essential, which can be increased by a proper support service defining utilities and limitations of wearable technologies and access to the application programming interfaces (APIs) and raw data in machine-readable format [4], [6]. Other challenges include defining the ownership of the wearables cost burden either on the healthcare structure or on the patient and eventual reimbursements. Nevertheless, is the regulatory framework that must ensure prevention from sensor measurement inaccuracy, misinterpretation, data protection, encryption, and patient general security [6]. From the technical point of view, the main challenges are sensor accuracy, battery life, integration of data from different systems, sensor body placement, communication protocols, signals quality, and robustness [4],[5],[6]. Accuracy is hampered by noises such as electromagnetic interference of power line, poor quality of contact between the electrode and the skin, baseline wanders caused by respiration, electrosurgical instruments, movement of the patient's body or, for example in ECG electrodes drying out [4],[5],[6]. SINTEC project faces these challenges with innovative technical solutions like rigid-stretch PCB technology with stretchable substrate and liquid alloy, smart patches for Fat-IBC, and low-energy Bluetooth (BTLE) communication. Explored sensors include optical sensors for photoplethysmography (PPG), pressure sensors for tonometry, as well as electrodes for ECG and bioimpedance acquisition [2].

1.2 Hypertension and BP monitoring methods

1.2.1 Hypertension

Ischemic heart disease and stroke are the top two causes of death, in terms of the total number of lives lost, with a combined total of over 2.5 million fatalities in 2019 only in high-income countries. Even if high income is the only category of income group in which there have been decreasing numbers of deaths from these two cardiovascular diseases (CVDs), deaths from hypertensive heart disease are rising. Reflecting a global trend, this disease has risen from the 18th leading cause of death to the 9th according to WHO [7]. Hypertension is a health condition characterized by high values of blood pressure in the arteries, due to the amount of

blood pumped by the heart and the arteries' resistance to the blood flow [8]. Hypertension is the strongest or one of the strongest risk factors for almost all different cardiovascular diseases acquired during life, cerebral stroke, and renal failure [9]. The main short and long-term consequences linked to high values of BP are stroke, coronary heart disease, heart failure, and cardiovascular death. Whereas the main long-term consequences are hypertensive cardiomyopathy, heart failure with preserved ejection fraction, atrial fibrillation, valvular heart disease, aortic syndromes, peripheral arterial disease, chronic kidney disease, dementias, diabetes mellitus, and erectile dysfunction [10]. In most hypertension cases, a precise, identifiable, and curable cause does not exist, the high BP values are due to an alteration of the complex regulatory pressure system [8]. Factors that increase the risk of developing hypertension are:

- Family history of premature cardiovascular disease (men $<$ 55 years; women $<$ 65 years)
- Age (men $>$ 55 years; women $>$ 65 years)
- Obesity (body mass index $\geq$ 30 kg/m^2 ($\frac{\text{weight}}{\text{height}^2}$))
- Diabetes
- Smoking
- Imbalance of sodium and potassium
- Alcohol
- Stress (both physical and mental)
- Sedentary lifestyle [8],[9].

The distinction between high normal blood pressure and hypertension is based on arbitrary cut-off values [9], as shown in Fig 1.1. Treatments after a hypertension diagnosis include lifestyle changes (balanced diet, physical activity, etc), weight loss, and drug therapy depending on the specific case (ACE, Calcium antagonists, diuretics, etc.) [8]. Hypertension is the level at which intervention to lower blood pressure has documented preventive benefits, especially prevention of the age-related increase in BP would substantially reduce the vascular consequences usually attributed to aging [10],[11].

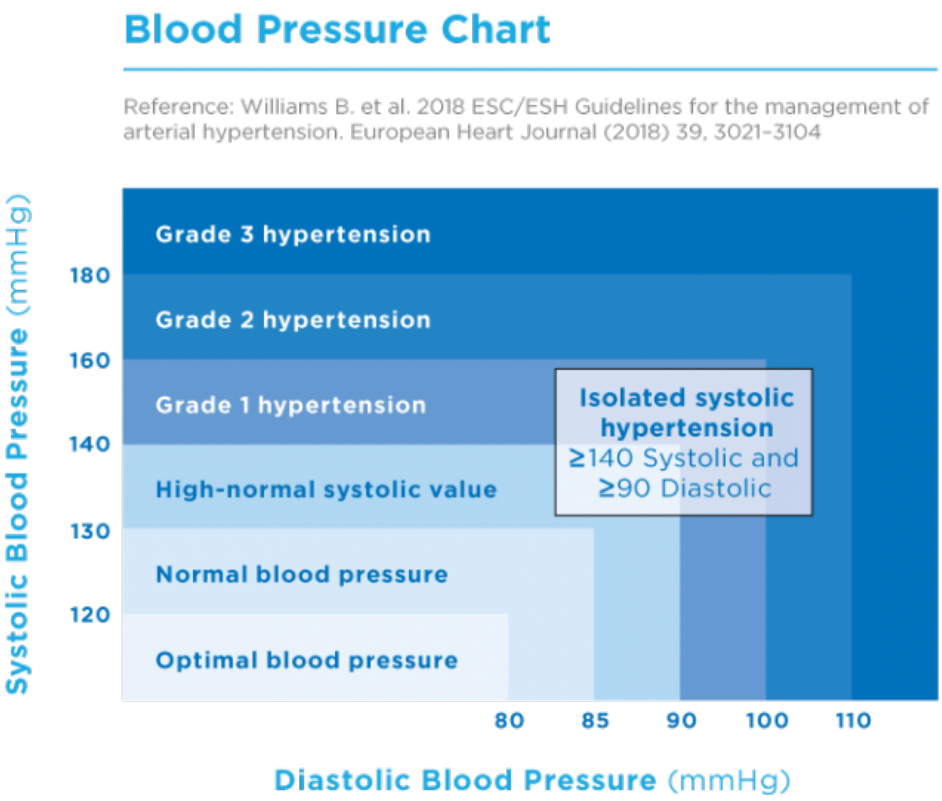

Figure 1.1: BP cut-off values [10]

1.2.2 BP monitoring methods

Monitoring BP is important to detect and treat hypertension, it is also highly recommended out-of-office monitoring such as ambulatory BP monitoring (ABPM) and home monitoring (HBPM), for this reason, many invasive and noninvasive techniques have been developed along with wearable solutions [12],[13].

Invasive continuous method

The clinical reference method is the direct measurement of BP via arterial cannulation [12], due to the precise and continuous measurements. This technique exploits an arterial catheter placed into an artery; the radial artery is the most common sampling site, Fig 1.2. This technique is mostly used in acute and critical care settings because it allows immediate recognition of alterations along with blood sample collection. Arterial cannulation must be performed by a trained operator to reduce risks of complication, classified as rare, like embolism, lesions of nerves, vessels, and ischemia [12],[14].

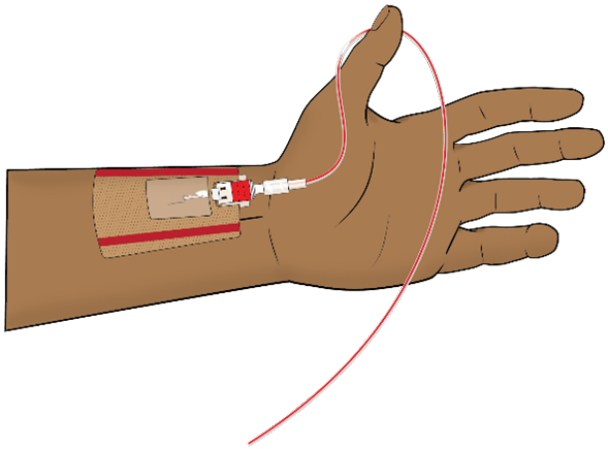

Figure 1.2: Arterial cannulation [15]

Noninvasive intermittent methods

The auscultatory method, a manually cuff-based method using a mercury sphygmomanometer and a stethoscope, is a noninvasive intermittent technique considered the gold standard in absence of a direct BP measurement. The SBP and DBP values are estimated from the appearance and disappearance of the Korotkoff sound, (pulsatile circulatory sound) [16]. In this technique, an inflatable cuff linked to the mercury sphygmomanometer is placed in the right upper arm of the patient, while the stethoscope is placed distantly. Air is inflated in the cuff until the cuff is completely occluded and the pressure inside is over a referred maximal arterial pressure, then the cuff is gradually deflated. During the deflation, it is noted the pressure when the first Korotkoff sound has been listened to, which indicates SBP, and the pressure when the last Korotkoff sound has been listened to indicates DBP, [16], [17]. The method is shown in Fig 1.3. This technique has the main disadvantage of providing intermittent values and needing a trained operator and a quiet environment [12]. The accuracy of this technique relies on the hearing acuity of the observer, the sensitivity of the stethoscope, and the amplitude and waveform morphometry of the Korotkoff sounds, plus the determination of DBP is challenging [16].

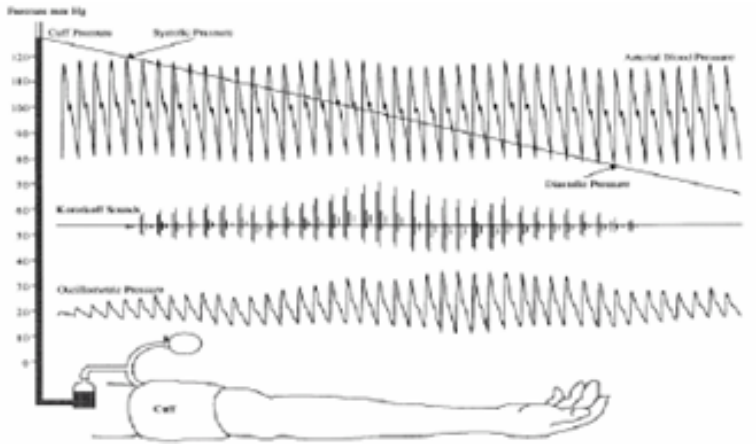

Figure 1.3: Auscultatory method [18]

The oscillometric method is an automated and intermittent technique for BP estimation exploiting an inflatable cuff as shown in Fig 1.4. The cuff can detect oscillations caused by pressure waves and determine SBP and DBP through algorithms comparing the maximum volume change rate. This method has shown accuracy problems, such as an underestimation of DBP values and overestimation in SBP [12],[16],[19].

Figure 1.4: Oscillometric method [20]

Noninvasive continuous methods

The arterial applanation tonometry is a noninvasive continuous method for BP measurements that exploits a precise pressure transducer, such as a tonometer. The arterial pulse wave is measured in a superficial artery with a flat bone beneath. The vertical displacement caused by the applying force of the tonometer is proportional to arterial pressure [12],[21]. The issue of the need for an examiner to handle the tonometer is solved by new-generation devices, such as T-Line system,Fig 1.5, (Tensys Medical, San Diego, CA, USA), which allows automated measurements [12].

Figure 1.5: Arterial applanation [22]

The volume clamp is a noninvasive continuous method for BP measurements exploiting an inflatable finger cuff containing an infrared transmission plethysmograph as shown in Fig 1.6. The sensor is sensible to arterial volume alteration and the blood pressure can be derived from the pressure needed in the cuff to maintain the artery diameter constant [12],[23].

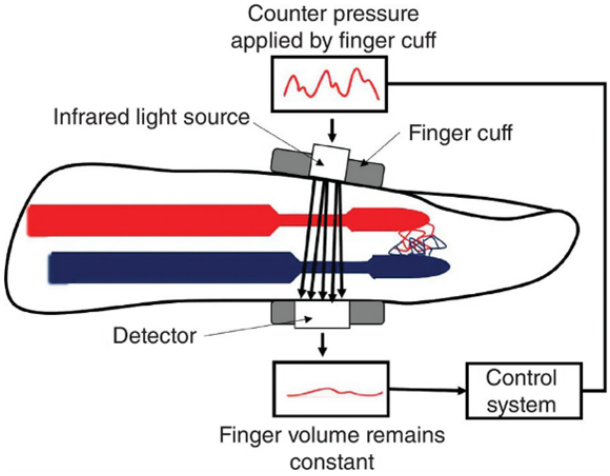

Figure 1.6: Volume clamp [24]

Wearables solutions

The above-mentioned techniques for BP estimation are cuff-based and face the common problem of inaccuracy due to wrong cuff size, sensibility to motion artifact, and discomfort for patients as a cause of the painful cuff inflation [12],[25]. For all these reasons, patents for wearable solutions for BP monitoring have increased in the last decade [25]. Omron HeartGuide (Omron Corp., Kyoto, Japan),Fig 1.7, is a wearable device for BP measurement, that has been validated in the laboratory environment according to the protocol from the American National Standards Institute, Inc/Association for the Advancement of Medical Instrumentation/International Organization for Standardization (ANSI/AAMI/ISO) 81060-2:2013 guideline [13],[26]. The device works like oscillometric devices thanks to the inflatable extra-stiff band [25], BP and heart rate values are read manually one minute after a button is pushed. It also allowed sleep monitoring and physical activity tracking [26]. Even if the device has an inflatable element, patients find it more comfortable, less intrusive, and less burdensome [13].

Figure 1.7: Omron HeartGuide [26]

Other wearable solutions recently evaluated are a finger-wearable monitor that exploits a capacitive tactile sensor array pushed by a pump-driven pneumatic bladder, and soft pressure sensors using wrinkled thin films worn on the wrist. Promising are sensors exploiting Pulse Transit Time (PTT) and ECG, but also machine learning techniques. The latter can overcome the challenge in BP estimation of eliminating unwanted features and calculating BP from optimal data [11].

1.3 Objective of work

This thesis work aims to perfect the algorithm that will be used by SINTEC final intended device for monitoring BP during physical activity and clinical settings, Fig 1.8. The final intended SINTEC device is a cuffless, noninvasive strain gauge flexible and stretchable sensor for continuous BP monitoring, that will be applied on the skin by a transparent patch. This thesis aims to perfect PPT and HR feature extraction from PPG and ECG signals through motion artifact reduction, enlarging the database for ML techniques, and developing a GUI for simplifying ML coefficients extraction and evaluation for BP estimation.

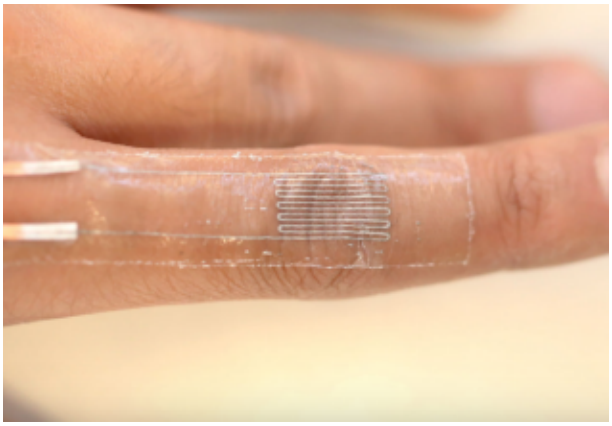

Figure 1.8: SINTEC attended final device [2]

Chapter 2

Physiological signals

SINTEC attended final device exploits ECG and PPG signals for extracting HR and PTT that will be used as features for BP estimation through ML techniques. Features reliability will be tested through the use of the accelerometer signals recorded at the same time as ECG and PPG signals.

2.1 ECG

The electrocardiogram, ECG, is a noninvasive method for monitoring heart electrical activity. The ECG is an extremely important tool for physicians to establish if heart electrical activity manifests anomalies. The ECG is the recording of the electrical current flow which crosses the heart during the cardiac cycle. ECG signal reflects how action potentials of the entire cellular population are generated. Since heart electrical activity is highly synchronized, electrical potential corresponding at the different heart electrical phases is relatively high and can be recorded on the skin surface [27]. ECG is a non-stationary signal [28] with a bandwidth that ranges from 0.05 Hz to 125 Hz and an amplitude 5mV that is morphologically interpreted.

2.1.1 ECG recording techniques

ECG recordings are based on Willem Einthoven technique, based on an imaginary equilateral triangle built around the heart,Fig 2.1. The Einthoven triangle vertices are in the left arm, right arm, and left leg. On each triangle angle, a couple of electrodes are located and linked to a device for voltage registration such as an oscillometer or a chart recorder, this is a bipolar registration and each couple of electrodes is called the lead. Each lead, indicated with a roman number, measures the difference in electrical potential from the positive and negative electrodes. The waveform direction depends on if the potential difference is positive or negative [27]. The three leads indicated by Einthoven are:

- Lead I: detection of potential difference between the left arm and right arm
- Lead II: detection of potential difference between the left leg and the right arm
- Lead III: detection of potential difference between the left leg and the left arm

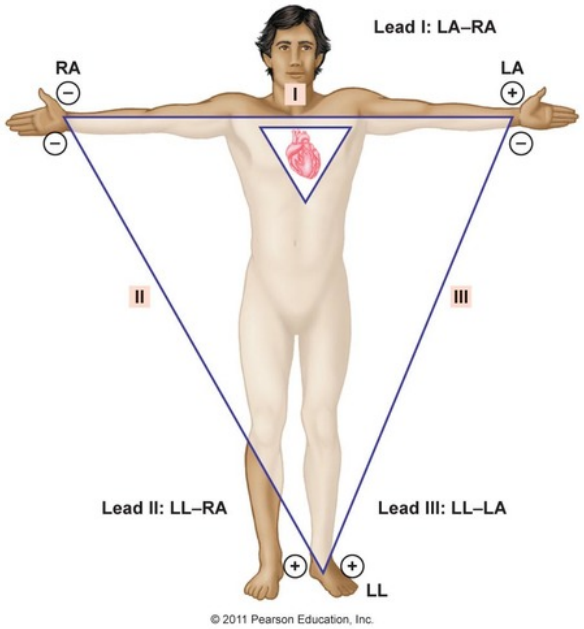

Figure 2.1: Einthoven triangle [27]

The most common ECG electrodes are Ag/AgCl surface electrodes, characterized by a low noise level and low electrode-skin interface impedance. Surface Ag/AgCl electrodes are non-polarizable electrodes that charge to cross the electrode-electrolyte interface [29].

2.1.2 ECG waveform

ECGs are recorded on a millimetric paper with a standard rate of 25mm/s and an amplitude of 10 mm/mV. A physiological ECG waveform,Fig 2.2, shows three main characteristics:

- The P wave indicates atrial depolarization, it is an upward deflection
- The QRS complex indicates ventricular depolarization, which is a succession of downward and upward deflections
- The T wave indicates atrial repolarization, it is an upward deflection

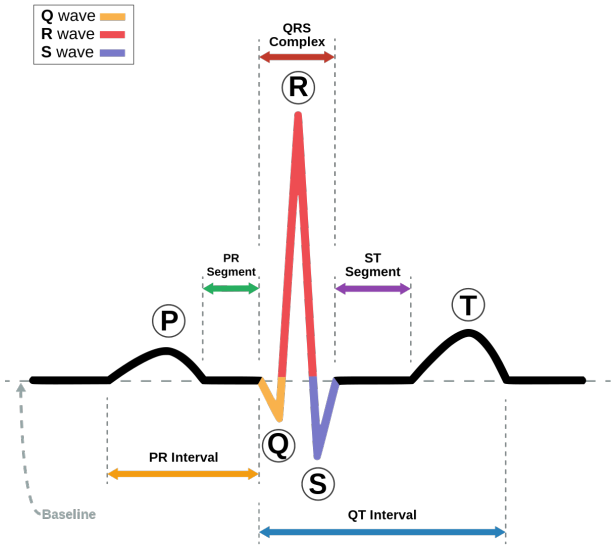

Figure 2.2: ECG waveform [30]

Heart pathophysiological state can be detected by analyzing intervals and segments of the ECG:

- P-Q (or P-Q) interval is a time estimation of the conduction through the AV node. It is calculated as the time between the beginning of P wave and QRS complex.
- Q-T interval is a time estimation of ventricles contraction, called systole. It is calculated as the time between the beginning of QRS complex and the end of T wave.
- T-Q segment is a time estimation ventricles relaxation, called diastole. It is calculated as the time from the end of T wave and the beginning of QRS complex.
- R-R interval is the time between the peaks of two consecutive QRS complexes, that is the time from one heartbeat to the next. From this interval is it possible to extract the HR feature:

$$\text{HR} = \frac{60 \text{ seconds}}{\text{RR interval}} \tag{2.1}$$

2.2 PPG

Photoplethysmography (PPG) is a non-invasive technology that uses a light source and a photodetector at the surface of the skin to measure volume changes, more

commonly PPG refers to the blood volume changes (pulse waves PWs) dues to the cardiac cycle but this technology can be used also for recording air volume changes during the respiration cycle [31]. PPG main characteristic features are HR and pulse oximetry, but it has shown potential for more valuable health-related information, such as diagnosis of various cardiovascular diseases, thanks also to the study of the first and second PPG waveform derivates [32]. Interest in PPG is increased in recent years due to its advantages as a non-invasive, economic, and wearable adaptable diagnostic tool. PPG signals can be recorded from different parts of the body, such as the forehead, earlobe, torso, wrist, fingertip, and ankle, and they are mostly morphologically interpreted, however different measurement sites have different degrees of accuracy. Challenges in the use of PPG signals in wearable solutions is high susceptibility to motion artifacts caused by body movements [32].

2.2.1 PPG recording technique

A light source and a photodetector are the two main elements in a PPG monitoring device. Two configurations are possible for PPG sensor, the transmission mode, and the reflectance mode, Fig 2.3. In the transmission mode, the light source and the photodetector are divided by the tissue and the photodetector detects the light components that have not been absorbed by the tissue. In the reflectance mode, the light sensor and the photodetector are located side by side, on the same side of the tissue and the photodetector measures the reflected light. From the light intensity detection, it is possible to measure the blood volume changes [32]. The mode selection depends on the measurements site, for example, the transmission mode is preferred for fingertip and earlobe whereas the reflection mode is preferred for wrists, forearm, ankle, forehead, and torso. Both modalities have advantages and disadvantages which have to be analyzed depending on the final application. Acquisition quality is linked to the pressure applied by the sensor to the tissue, because too much pressure, especially in transmission mode, can cause a reduction of venous oscillations [32]. PPG light sources are in general infrared light-emitting diodes (IR-LEDs) or a green LED. IR-LEDs are generally used for monitoring blood changes in parts like muscles, more deeply concentrated whereas the green LED is used to measure the oxygen absorption in oxyhemoglobin and deoxyhemoglobin. Among all the colors, the green LED is selected to its highly penetrative characteristics.

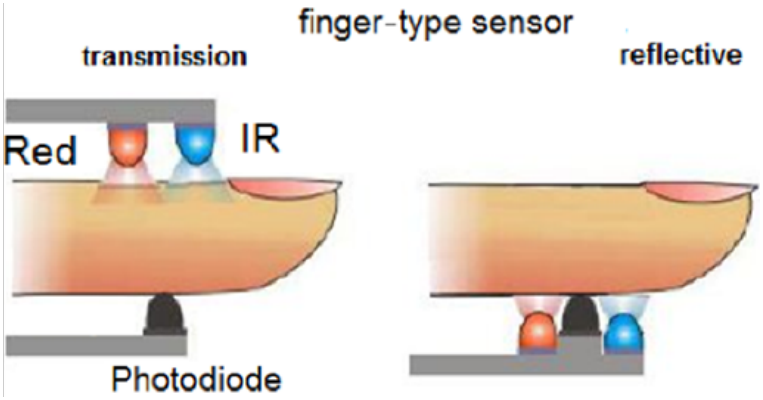

Figure 2.3: PPG sensors, on the right the transmission mode and on the left the reflectance mode [33]

2.2.2 PPG waveform

PPG signal has two components, the pulsatile (AC) strictly linked to blood volume changes dues to the cardiac cycle, and the superimposed (DC) shaped by respiration, sympathetic nervous system activity, and thermoregulation [32]. PPG waveform, Fig 2.4, shows two phases. The rising edge of the signal is called the anacrotic phase and it is linked to the heart systole whereas the falling edge is the catacrotic phase, linked to the diastole. The presence of the dicrotic notch in the falling edge reflects the presence of healthy arteries [34].

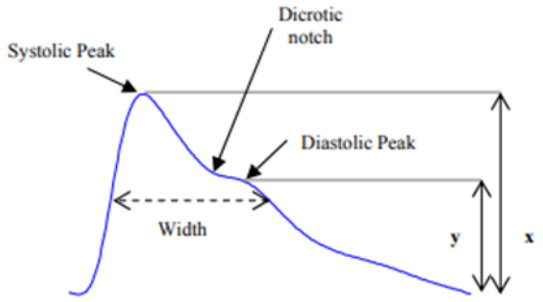

Figure 2.4: PPG waveform [34]

PPG main characteristics are:

- Systolic amplitude x, which shows the pulsatile changes of arterial blood in the nearby of the measurement site
- Pulse width, which is calculated at the half height of the systolic peak, is correlated with the systemic vascular resistance

- Pulse area, the area below the PPG curve. The area ratio between the two areas divided by the dicrotic notch is an indicator of peripheral resistance
- Peak-to-Peak, interval, Fig 2.5, which is the distance between two consecutive systolic peaks, is highly correlated with the R-R interval

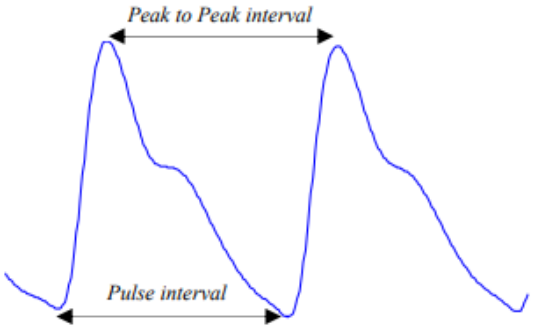

Figure 2.5: Pulse interval, Peak to peak interval [34]

- Pulse interval, which is the distance between the beginning and the end of PPG waveform. It can be used to calculate heart rate variability (HRV)

More information can be determined by the second derivative wave of PPG signal, the acceleration photoplethysmogram (APG). APG waveform is correlated with the distensibility of the carotid artery, age, blood pressure, the estimated risk of coronary heart disease, and the presence of atherosclerotic disorders [34]. APG has also been described as a potential diagnostic tool for other disorders, varying from a sensation of coldness and stress experienced by surgeons to exposure to lead, pneumonia, intracerebral hemorrhage, and acute poisoning [34].

2.3 ECG and PPG for BP monitoring

As mentioned above, there is a high correlation between the R-R interval of ECG and the peak-to-peak interval of PPG. The feature of interest for this thesis work is the pulse transit time (PTT), defined as the propagation time of a PW going from the heart to the peripheral arteries and is calculated as the time between the R-peak of the ECG and a reference point on the PW measured using PPG [35], Fig 2.6..

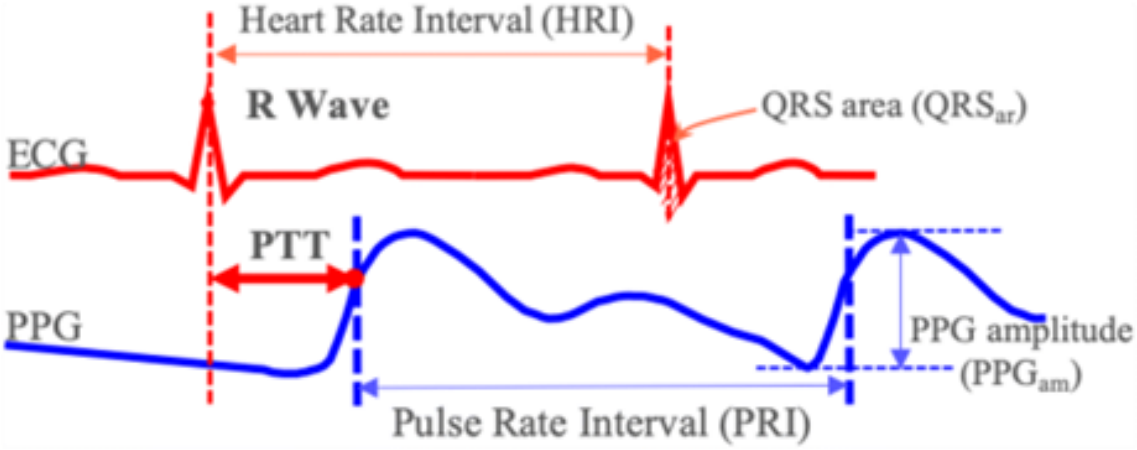

Figure 2.6: PTT [36]

PTT is inversely related to pulse wave velocity, PWV, which can be considered the gold standard for measuring arterial stiffness and can be an indicator for diagnosing cardiovascular diseases [35].PTT shows a strong relationship with BP and can be used as a feature along with HR for BP estimation through Bramwell–Hills and Moens–Kortweg's equations [37], which lead to Eq. 2.2:

$$\mathrm{BP} = \mathrm{aPTT} + \mathrm{bHR} + \mathrm{c} \tag{2.2}$$

Coefficients a, b, and c are subject-specific parameters and must be obtained through a calibration procedure. SBP and DBP can be obtained as Eq.s 2.3:

$$\begin{cases} \mathrm{BP} = a_s\mathrm{PTT} + b_s\mathrm{HR} + c_s \\ \mathrm{BP} = a_d\mathrm{PTT} + b_d\mathrm{HR} + c_d \end{cases} \tag{2.3}$$

Whereas, b_s, c_s, a_d, b_d, c_d are subject-specific coefficients for SBP and DBP estimation [37].

2.4 Accelerometer

The accelerometer is an inertial measurement unit (IMU) that measures the proper linear acceleration, that is the difference between the acceleration sensed along the sensitive axis of the sensor and the gravity acceleration. An accelerometer can be schematized as a second-order spring-mass-damper system, which can measure the acceleration as the spring compression when a mass is moved to the point that the spring can push (accelerate) the mass at the same speed as the casing or can measure the frequency of the vibrating mass-proof element it's caused because of tension changing [38], [39]. An accelerometer can be classified by the transduction mechanism employed to convert the proof-mass displacement due to acceleration.

The main transduction systems are piezoresistive, capacitive, tunneling, optical and piezoelectric. The selection must be made upon the different advantages and disadvantages of each transduction mechanism [40], Fig 2.7.

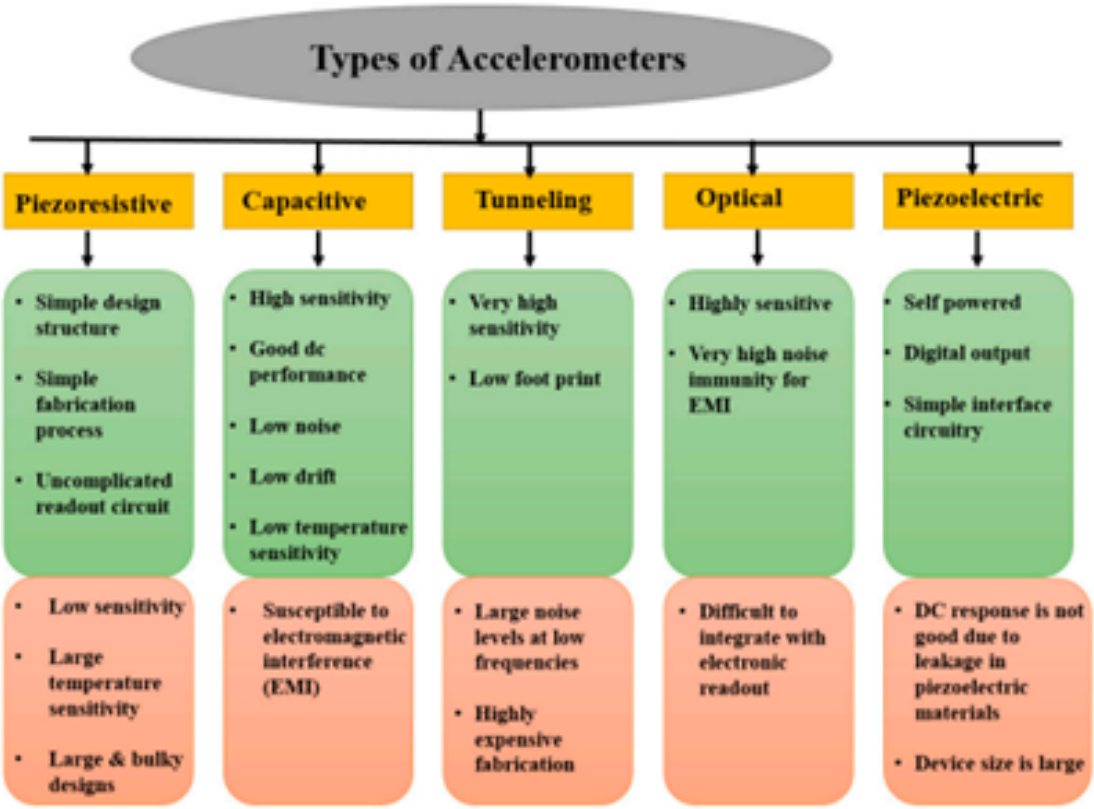

Figure 2.7: Advantages and disadvantages of various transduction schemes [40]

In the human body, the utricle, and the saccule, the otolith organs, work as a human linear accelerometer. Otolith organs are part of the vestibular system, which is responsible for the ability to sense body movement and maintain balance, they sense the direction and the speed of linear acceleration and the position (tilt) of the head [41]. When a force, due to an example at the tilting of the head, leads to a displacement of the otoliths relative to the connective tissue, the displacement is sensed by the hair cell that bending polarizes the cell and induces excitation, Fig. 2.8.

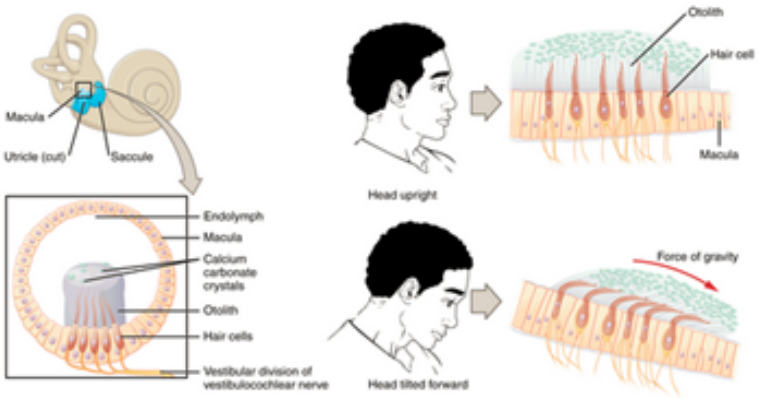

Figure 2.8: Utricle ans saccule [42]

2.4.1 Use of accelerometer in health wearables applications

Accelerometers, along with the other IMU sensors gyroscope, and magnetometers, are widely used for health wearable applications for motion detection. Accelerometer signals are largely used in algorithms for balance disorders evaluation or monitoring. From accelerometers and gyroscopes signals is it possible to extract important features such as gait speed, and step and stride length. Accelerometers can be used in balancing prostheses, detecting body orientation, and providing information to the patient through mechanical actuators. Accelerometer use can space from remote patient surveillance in ICUs to improving radiation oncology. Accelerometers can be used for measuring the intensity of physical activity and hereby energy expenditure in both sports or rehabilitation exercises [43]. In the aim of this thesis work, accelerometer signals will be acquired simultaneously with PPG and ECG to detect potential MA that could affect the wrong PTT and HR features extraction and reduce the accuracy of BP estimation.

Chapter 3

Materials and methods

For the aim of this thesis work a 40 ECG, PPG, and accelerometer signals have been added at the Shimmer database [44] . The new signals include the recording of 6 subjects (4 females and 2 males), all aged between 23-25 years, for a more detailed population description, please refer to APPENDIX D. Before proceeding with the signals acquisition, each subject read and signed the informed consent with the attached information note related to SINTEC project. For the full informed consent, refer to APPENDIX A.

3.1 Devices, sensors, and software

The signals database has been created by using Shimmer devices. Shimmer is an Irish company founded in 2008, a pioneer and leader in the development of wearable wireless sensing solutions. Shimmer wireless sensor platform's main characteristics are the possibility to record and communicate data in real-time, low-power wireless communication, and a large storage capacity [45],[46]. For the aim of this thesis work, two Shimmer3 ECG Units, one Shimmer3 GSR+ Unit, a Consensys Base6, and ConsesysPRO Software have been used. Omron HeartGuide,Fig. 1.7, has been used simultaneously for BP monitoring, which values will be used to train the algorithm.

3.1.1 ECG device and sensor

ECG signals have been recorded through Shimmer3 ECG- Unit device, Fig 3.1, Shimmer Biophysical snap leads, Fig 3.2, and Covidien Ag/AgCl snap on electrodes, Fig 3.3. Shimmer3 ECG-Unit is a lightweight device with compacted dimensions (65 x 32 x 12 mm) with an EEPROM memory of 2048 bytes. The Shimmer3 ECG/EMG-Unit has a software configurable gain, data rate, and right-leg drive for common-mode interference rejection, with an input differential dynamic range of

approximately 800 mV and a bandwidth of 8.4 kHz. The device has 4 input channels and a reference channel. Input protections include ESD and RF/EMI filtering, current limiting, and defibrillation protection [47]. The integrates 10 DoF inertial sensing via an accelerometer, gyroscope, magnetometer, and altimeter. Covidien electrodes have an adhesive side with non-irritating gel and a latex-free foam to prevent allergic reactions [48].

Figure 3.1: Shimmer3 ECG unit

Figure 3.2: Shimmer Biophysical snap leads

Figure 3.3: Covidien Ag/AgCl electrodes

3.1.2 PPG device and sensor

PPG signals have been recorded through the Shimmer3 GSR+Unit device, Fig 3.4, and Shimmer Pulse ear-clip, Fig 3.5. Shimmer3 GSR+Unit is suitable for measuring the electrical characteristics or skin conductance and PPG signal. Shimmer3 GRS+ is a lightweight device, with compacted dimensions (65 x 32 x 12 mm) with an EEPROM memory of 2032 bytes. The device has 3 input channels, two SSR inputs, and an auxiliary analog/digital input that can be used for PPG recording through a Shimmer Pulse ear clip. Input protections include RF/EMI filtering and current limiting, and GSR inputs include defibrillation protection [49]. The ear lobe optical pulse circuitry of the Shimmer Pulse ear clip includes an onboard amplifier and filter circuit for initial conditioning of the signal [50].

Figure 3.4: Shimmer3 GSR+ Unit

Figure 3.5: Shimmer ear clip

3.1.3 Accelerometer device and sensor

Accelerometer signals have been recorded by enabling the accelerometer in the ECG Shimmer3-Unit. In particular, the accelerometer is part of MEMS InvensSense ICM-20948, which combines a 3-axis gyroscope,3 -axis accelerometer, and compass and it is particularly suited for tablets, wearable sensors, and IoT applications [48],[51].

3.1.4 ConsensensysPRO software and Consensys Base6

ConsensysPro software, Fig 3.6, and Consensys Base6,Fig 3.7, or Shimmer dock, have been designed to be used together to make easier the programming, configuration, data recording, and data storage from different Shimmer devices at the same time. The software keeps track of the setup parameters of the Shimmer configuration, even after a firmware update, to enhance the useability for multiple trial sessions.

Figure 3.6: ConsensysPRO software [52]

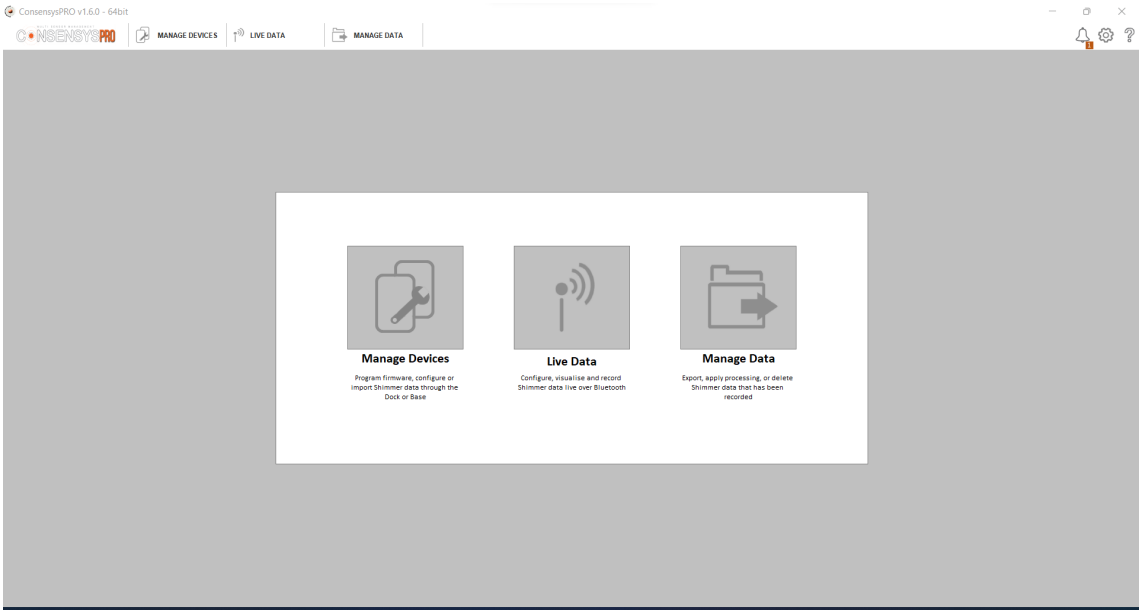

Figure 3.7: Consensys Base6

3.2 Protocols

The signals acquisition protocol foresees 20 minutes recording where the patient is relaxed and sat, and he/she is asked to perform some controlled movements during the recording such as getting up and having a small walk, moving his/her hands while holding the sensors. Acquisitions have been made at different hours of the day. The procedure starts when the Shimmer dock is switched on with all the necessary Shimmer devices correctly seen by ConsensysPro software.

3.2.1 Sensors placement

ECG is recorded with the I lead, LA-RA. After the patient skin is cleaned with an alcohol swab, an electrode is placed on the right and another on the left of the patient's chest, and the reference electrode is placed on the right leg. The active electrode on the right is inserted in the white pin of the Shimmer3 ECG-Unit, the active electrode on the left is inserted in the black pin, and the reference electrode in the green pin. PPG is recorded using the Shimmer ear clip on the patient's left finger. Due to the high light sensibility of PPG signals, the patient's finger with the ear clip is covered with a black strip. Tightening too much the black strip around the clip could compromise signal quality. The ear clip is inserted in the black pin of the Shimmer3 GSR+ device. Omron HeartGuide is located on the patient's right wrist. During the recording, the patient will hold the PPG and

accelerometer device in the left hand and the ECG device in the right hand. The final experimental set up is shown in Fig 3.8.

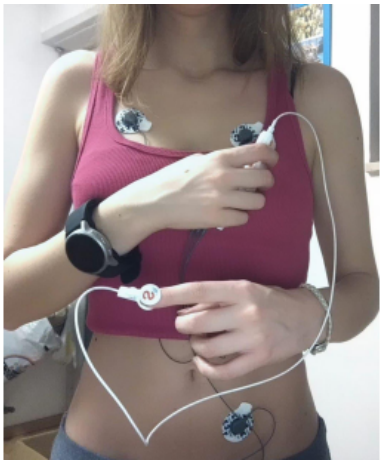

Figure 3.8: Sensor set up [44]

3.2.2 Software configuration

After linking the Shimmer dock to the PC and switching on all Shimmer devices it is necessary to launch ConsensysPRO software. After clicking on "Manage Devices", it is important to check that all three Shimmers appeared correctly on the graphic and in the device list. After selecting all the devices, it is necessary to program the firmware by clicking on the "FIRMWARE" button. Firmware programming is done by clicking on "SDLog" which allows for synchronization between multiple Shimmers when logging into the SD card, Fig 3.9. Then it is necessary to configure the session trial:

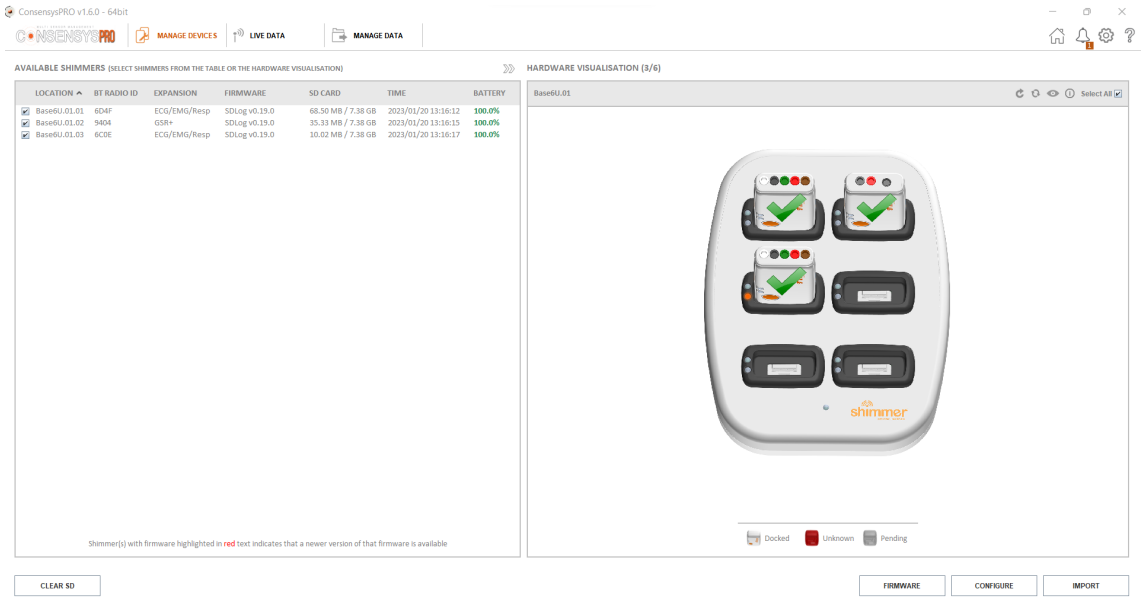

Figure 3.9: Shimmers correctly selected, ready for firmware writing

- Choose a trial name: i.e. *ProvaSubject*1
- Select the sampling rate: 504.12Hz
- Choose Mode based on estimated logging duration: disabling Sync Devices and selecting Undock/Dock
- For the ECG recording: select the intended Shimmer3 ECG Unit, in Sensor, select only ECG. No algorithm was selected, Fig 3.10.

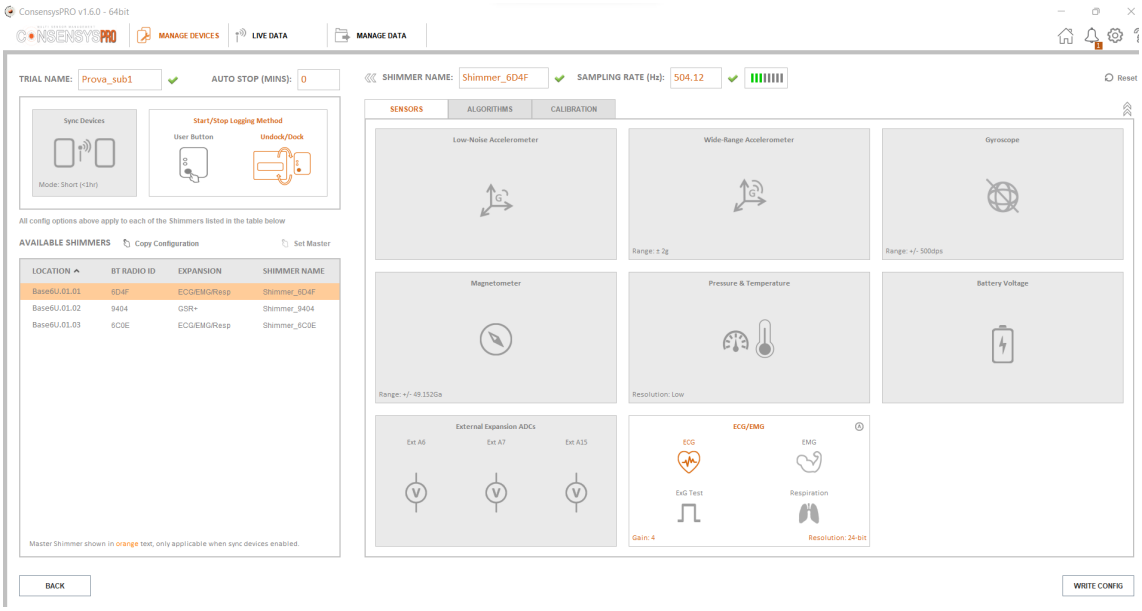

Figure 3.10: ECG configuration

- For the PPG recording: select the GRS+ Unit, in Sensor select only PPG. No algorithm was selected, Fig 3.11.

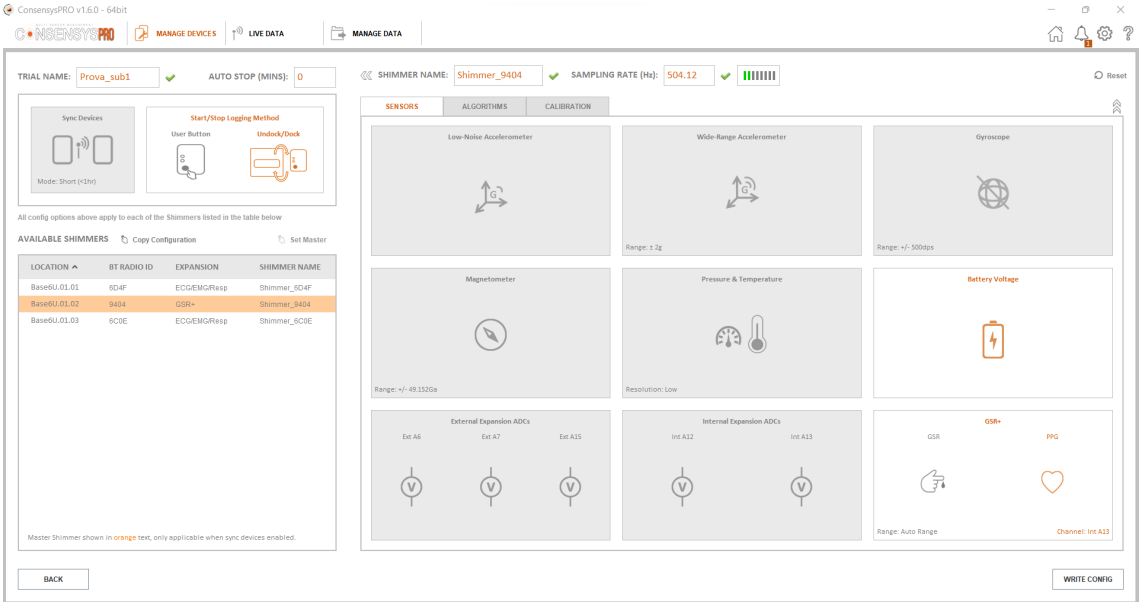

Figure 3.11: PPG configuration

- For the accelerometer recording: select the intended Shimmer3 ECG Unit, in

Sensor selecting only Low-noise Accelerometer. No algorithm was selected, Fig 3.12.

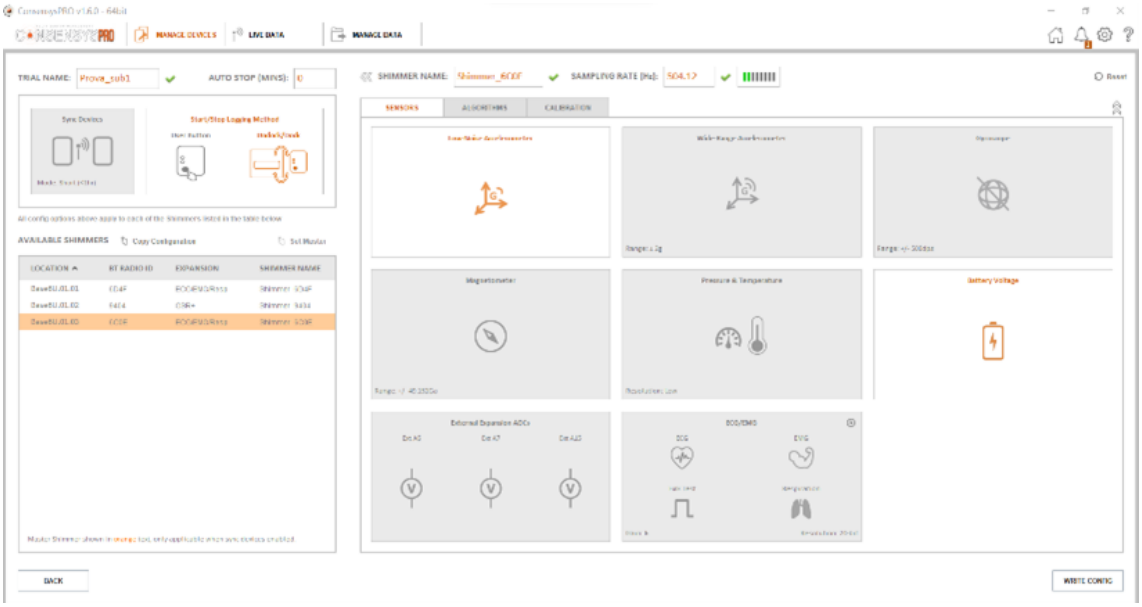

Figure 3.12: Acceleration configuration

3.2.3 Recording

Now, when the Shimmers devices will be undocked, the recording will start. Right after, Omron upper button will be pushed, and it will start a one-minute BP recording. When Omron finishes its measurement,Fig 3.13, the time and SBP, and DBP values will be stored in an Excel spreadsheet. During the recording, it is required at the patient, that is relaxed and sat, to have a small walk, or move his/her hands on purpose one or two times, in random order and intensity, during the recording. The recording is over when 20 minutes are passed, and the devices are docked again on the base.

3.2.4 Import data

To import data, it is necessary to select all the devices in Manage devices and click on "IMPORT". Then, select the correct trial between the available trials, and through the double fleshes import the trial in "TRIALS FOR NEXT STAGE", then click on "NEXT". After the import is done, click on "DONE". Then, select the folder where stored the files and save them in .mat format. Now signals are ready to be analyzed.

3.2.5 Unix conversion and .csv file

For BP estimation through ECG and PPG signals is it necessary to train the algorithm with Omron measurement. The Excel spreadsheet with time in format HH: MM, SBP, and DBP values in columns is saved in a .csv file after converting the time in UNIX format, Fig 3.14. It is important when converting to UNIX, to set up a local time area.

Figure 3.13: Omron measurement [53]

G7

	A	B	C
1	1673611980	125	93
2	1673612040	133	66
3	1673612100	141	80
4	1673612160	127	72
5	1673612340	138	67

Figure 3.14: Unix time stamp amd SBP,DBP values

3.3 Algorithm overview

The algorithm was developed in Python, which is an interpreted, object-oriented, high-level programming language with dynamic semantics [53]. The algorithm can be divided into seven main sections:

1. Signal preparing. In this section, ECG and PPG signals recorded through ConsesysPRO are loaded. The first operation is the remotion of the possible noise introduced by the undocking of the devices, for this reason, 20 seconds of samples are removed. For the correct algorithm, work is necessary to align, cut at the same length, and synchronized all of the tree signals.

2. Signal filtering. In this section, the ECG signal is filtered to remove the baseline through the envelope function. The PPG signal is filtered with a seven-order Butterworth filter and then also PPG baseline is removed through the envelope function.

3. Peaks detection. In this section, the user must set up an ECG and PPG threshold to detect signal peaks through the python function findpeaks.py. The algorithm implements a control that excludes extra peaks that appear in a 0.5 second window. The outputs of this section are an array containing only zeros and peaks values and the relative time stamp array for both PPG and ECG signals

4. Feature extraction. In this section HR and PTT features are extracted from R and P peaks array. HR is calculated from the timestamps of the two following R peaks. PTT is calculated from the timestamps of an R peak and the timestamp of the first S peaks that follows the R peak in time. After saving HR and PTT in arrays with the relative timestamps, the features are cleaned by removing HR and PTT values that are out of range mean+/- SD. The last step of this section is the HR and PTT interpolation along the whole signal's timestamp.

5. Omron HeartGuide data preparing. During signal acquisition, Omron HeartGuide returns punctual values of SBP and DBP each minute. On the .csv file is reported the Unix time stamp when the recording finishes, for this reason, before using Omron data, it is necessary to subtract 60 seconds from the Unix time stamp. After, it is necessary to interpolate Omron values along the whole signal's timestamp array.

6. Feature reduction. This section aims to reduce errors due to device synchronization. To achieve that, the previous extracted features HR and PTT along with SBP and DBP values extrapolated from the Omron are averaged with 10 seconds time intervals and then are resampled with a shorter time stamp array.

7. Regression analysis. In this section, a Multivariate Linear Regression (MLR) has been implemented and tested on dataset divisions without any time interval.

The full algorithm flowchart is shown in, Fig 3.15.

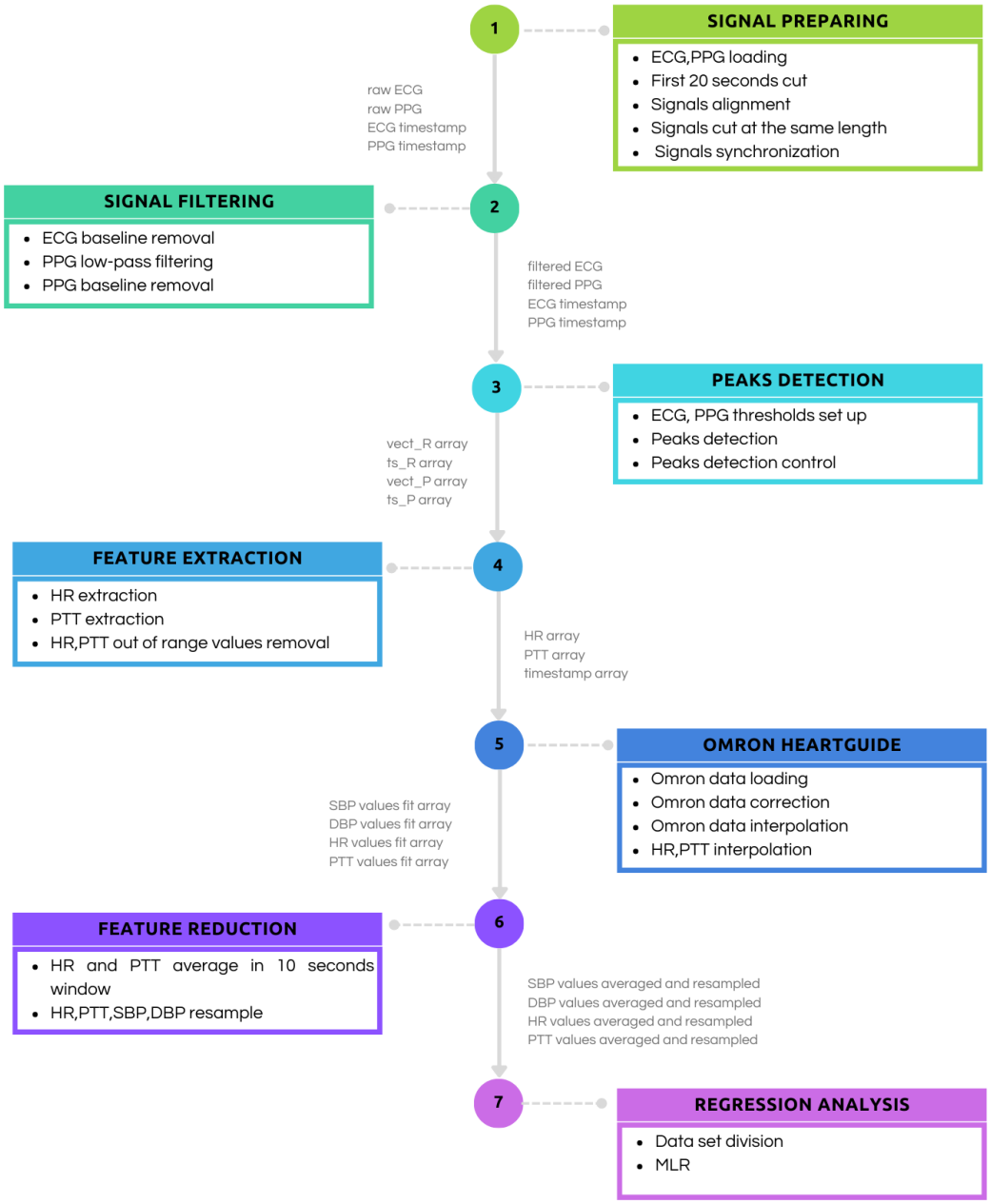

Figure 3.15: Algorithm flowchart

Chapter 4

MA reduction strategies techniques and GUI design

SINTEC final device is intended to monitor BP both in a clinical environment and in active life, including athletics performance evaluation [2]. For this reason, this thesis work investigates possible techniques to enhance the robustness of the proposed algorithm to motion artifacts, (MA). Breathing, and muscle contraction, for example, can temper the quality of features extraction from ECG signals a cause of the motion artifact introduced by the unstable contact of the electrodes on the skin [54]. In PPG signals, even the slightest movement can distort PPG waveform [55],[56]. The techniques for MA reduction introduced by this thesis work required a three-axis accelerometer. To enhance the usability and transferability of the proposed algorithm, a GUI has been developed which allows the combinations of all the proposed techniques, saving data and faster the whole process of research. A schema of the new introductions is shown in Fig 4.1

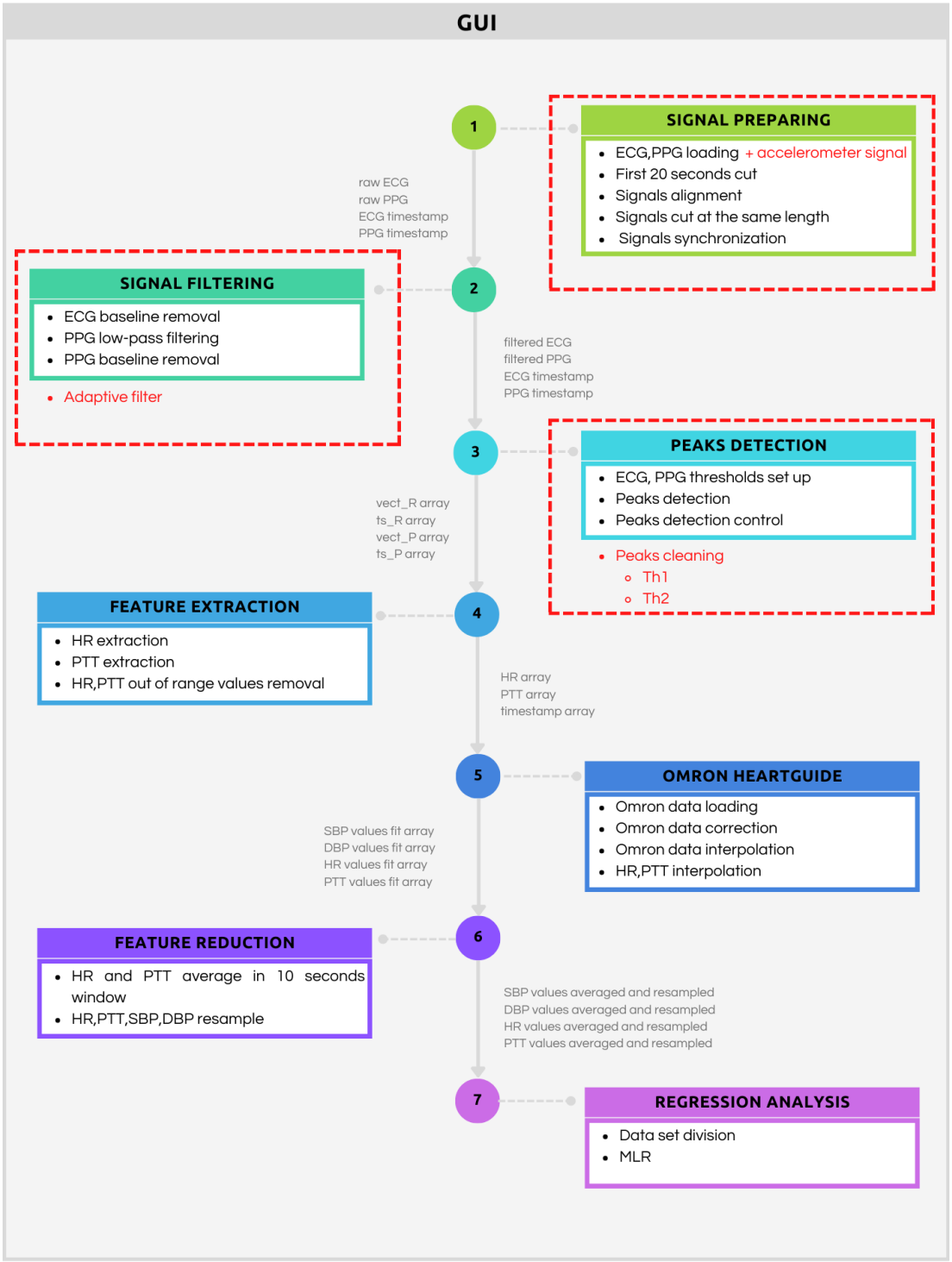

Figure 4.1: Algorithm modifications

4.1 Introduction of the accelerometer

The three-axis accelerometer signals have been used for introducing two techniques, the first foresees the construction of an LMS adaptive filter, and the second analyzes the accelerometer signals variations for discarding ECG and PPG features, R and P peaks, which can be highly compromised by MA. The accelerometer calibration has been checked by analyzing the norm of the three accelerometer signals recorded along the three sensitive axes during a five-minute recording where the sensor was placed on a desk.The following techniques are based on the norm of the acceleration signal, computed as Eq. 4.1:

$$ACC = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} \tag{4.1}$$

Where acc_x, acc_y, acc_z are the Min/Max normalized values of the accelerometer signals recorded respectively along axis x, y and z. The acceleration signals have been used also for computing a singular values decomposition, SVD, where the diagonal elements of matrix S contain information about the noise level. SVD is computed as Eq. 4.2:

$$A = USV^T \tag{4.2}$$

Where A is containing Min/Max normalized values of the accelerometer signals, S is the diagonal matrix containing the singular values which represent the importance of each mode of the signal, U and V matrices represent the corresponding mode shapes. The adaptative filter strategy has been selected after analyzing the power spectrum of normalized ECG, normalized PPG, and the norm of accelerometer signals recorded during standing up from a chair. In Fig 4.2, it is possible to see how the three signals contain most of their spectral information in the same bandwidth, this makes it difficult to denoising the biological signals from MA by traditional low pass, high pass, and bandpass filters.

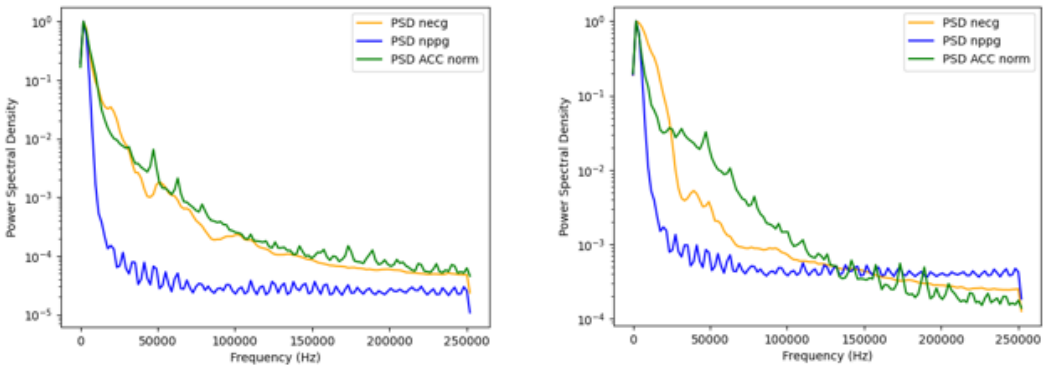

Figure 4.2: PSD analysis of normalized ECG, PPG and ACC signals recorded during standing up from a chair in Subject 7 and Subject 8. The PSD has been evaluated with a Welch periodogram with a 50 % overlap and a modified Hann window that reduces variance by using a non-overlapping sum of squared Hann windows.

4.1.1 Adaptive filter

To reduce MA in ECG and PPG signals, a Least Mean Squares (LMS) adaptive filter has been proposed. LMS is widely used due to its simplicity, stability, and robustness and its reduced computational cost compared to other adaptive filters, which makes LMS also suitable for real-time applications [57],[58]. LMS adaptive filter requires a desired signal, which is the target signal that the adaptive filter aims to produce, an input signal, which is the signal that will be processed by the filter, a specified filter length, which determines the number of coefficients in the filter, a specified step size, which is a scalar number needed for the stability and convergence of the filter and a specified number of iterations, which is the number of times the filter updates. The outputs of LMS adaptive filter are a filtered signal, which is the estimated desired signal obtained by filtering the input signal through the adaptive filter, and an error signal, which is the difference between the desired signal and the filtered signal. It is used to update the filter coefficients in each iteration. The adaptive filter starts initializing the error signal and the output signal as arrays of zeros, then a loop over the number of iterations specified starts. For each iteration, the output signal is computed as the scalar product of the input signal and the filter coefficients. Then, the error signal is computed as the difference between the desired signal and the output signal, and the filter coefficients are updated by combing the previous coefficients, the step size, the error signal computed just before, and the input signal. The final filtered signal and the final error signal are outputted when all n-iterations have been performed. The LMS adaptive filter proposed in this thesis work uses the accelerometer signals as the target, and the

noisy PPG and ECG signals as inputs, in this way it is possible to obtain the denoised PPG and ECG signals as the output error signal, Figure 3. The filter length can be selected in a range of experimentally selected values (7,20) and the step size has been experimentally selected at 0.01. The number of iterations has been set equal to the length of the input signal. The accelerometer signals used as the signal target have been experimentally selected as the norm of Min/Max normalized values of the accelerometer signals recorded respectively along axis x, y, and z (1). Also, the input signals, noisy ECG or noisy PPG, are Min/Max normalized before entering the adaptive filter. After the adaptive filtering, both PPG and ECG are denormalized and they are ulteriorly filtered to remove the baseline.

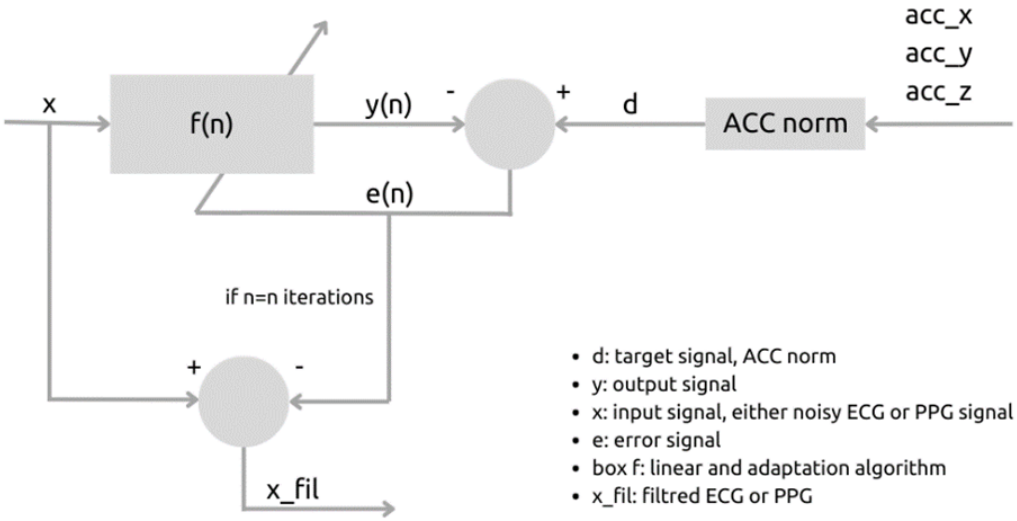

Figure 4.3: LMS Adaptive filter using the norm of the acceleration signals as the target, the noisy ECG or PPG as inputs, the step size set at 0.01 and the number of iterations equals to the length of the input signal

The LMS adaptive filter proposed in this thesis work uses the accelerometer signals as the target, and the noisy PPG and ECG signals as inputs, in this way it is possible to obtain the denoised PPG and ECG signals as the output error signal, Fig 4.3. The filter length can be selected in a range of experimentally selected values (7,20) and the step size has been experimentally selected at 0.01. The number of iterations has been set equal to the length of the input signal. The accelerometer signals used as the signal target have been experimentally selected as the norm of Min/Max normalized values of the accelerometer signals recorded respectively along axis x, y, and z Eq. 4.1. Also, the input signals, noisy ECG or noisy PPG, are Min/Max normalized before entering the adaptive filter. After the adaptive filtering, both PPG and ECG are denormalized and they are ulteriorly filtered to

remove the baseline.

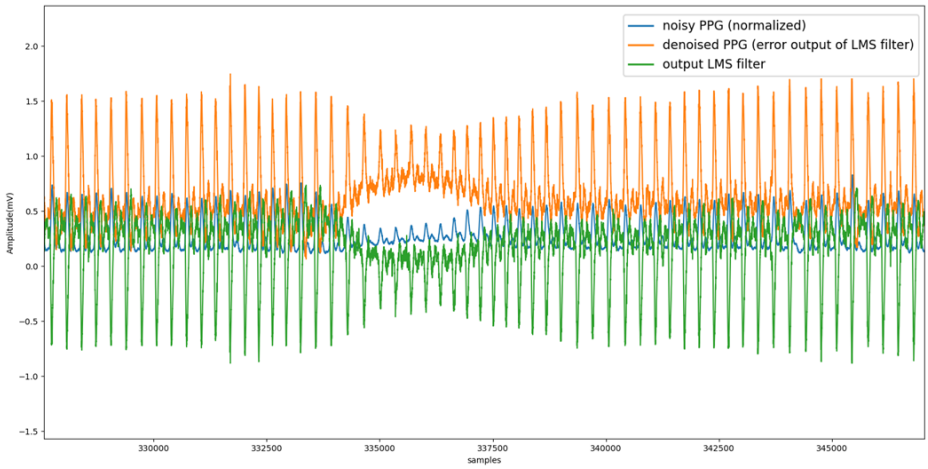

Figure 4.4: Min/Max normalized PPG signal of subject 7 before (blue) and after (orange) been filtered with a LMS Adaptive filter. In green, it is shown the output signal

In Fig 4.4 is shown a normalized PPG signal before and after being filtered with the LMS adaptive filter. Is it possible to notice how during the MA the signal amplitude decreases, while the amplitude of the LMS filtered signal is less affected, and in general in the filtered signals peaks are more defined.

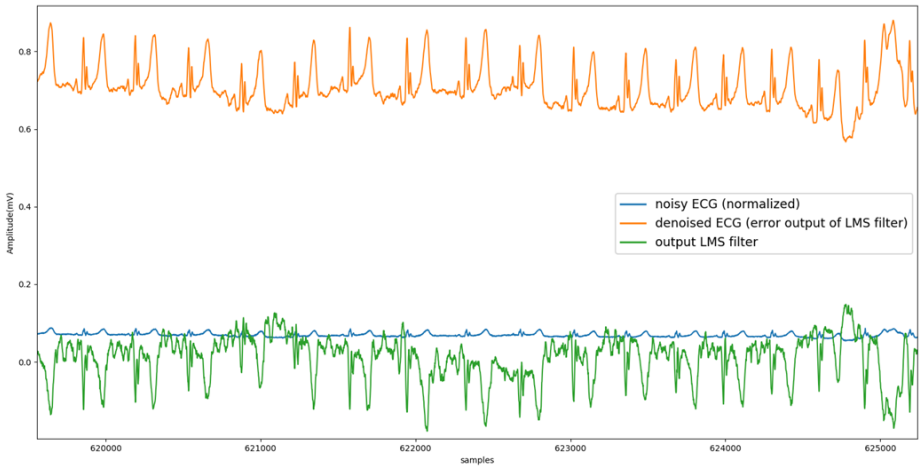

Figure 4.5: Min/Max normalized ECG signal of subject 7 before (blue) and after (orange) been filtered with a LMS Adaptive filter. In green, it is shown the output signal

In Fig 4.5 is shown a normalized ECG signal of subject 7 before and after being filtered with the LMS adaptive filter. In this case, is quite visible how the ECG is generally less attenuated by the noise and peaks are more visible.

4.1.2 Peaks cleaning

The second proposed approach used accelerometer signals to reduce the impact of MA in BP estimation by discarding R and P peaks which can be potential fake peaks due to MA. This approach foresees scrolling the accelerometer signal of reference with windows of various lengths, and if the reference acceleration signal is outside a determined range, peaks detected in the same timestamp of the acceleration value outside the range, are discarded. Two are the reference accelerometer signal proposed, the first is the acceleration norm 4.1, while the second is the S component of the SVD decomposition 4.2. If the reference signal is the acceleration norm, it can be divided into windows of one minute, 30 seconds, or one window of the same length as the reference signal itself. Due to its nature, if the reference signal is composed of the diagonal element of S, the window is usually considered the same length as S. On each window, the mean and the standard deviation of the reference accelerometer signal (ARS) are computed in order to set the upper and lower limit of the range. The default upper and lower limits of the range are computed as:

$$\begin{cases} \text{Upper limit} = \text{mean(ARS)} + 2 \times \text{SD(ARS)} \\ \text{Lower limit} = \text{mean(ARSl)} - 2 \times \text{SD(ARS)} \end{cases} \tag{4.3}$$

The width of the range can be changed by selecting another multiplicative coefficient (1.5,2, 3). The peaks cleaning process is based on the timestamp of the elements of the reference acceleration signal outside of the determined range. The process foresees the creation of a binary signal of the same length as the reference acceleration signal where the value zero indicates that the reference acceleration signal is inside the range and one if it is outside the range. The binary signal peaks are smothered by filling the gaps between the peaks that are less distant than an experimental threshold set at ten times the sampling frequency. Through this binary signal, the arrays containing the R and P peaks previously determined are cleaned. Specifically, peaks that correspond to the time interval where the binary signal is one are discarded for BP estimation. The peaks cleaning process is illustrated in Fig 4.6.

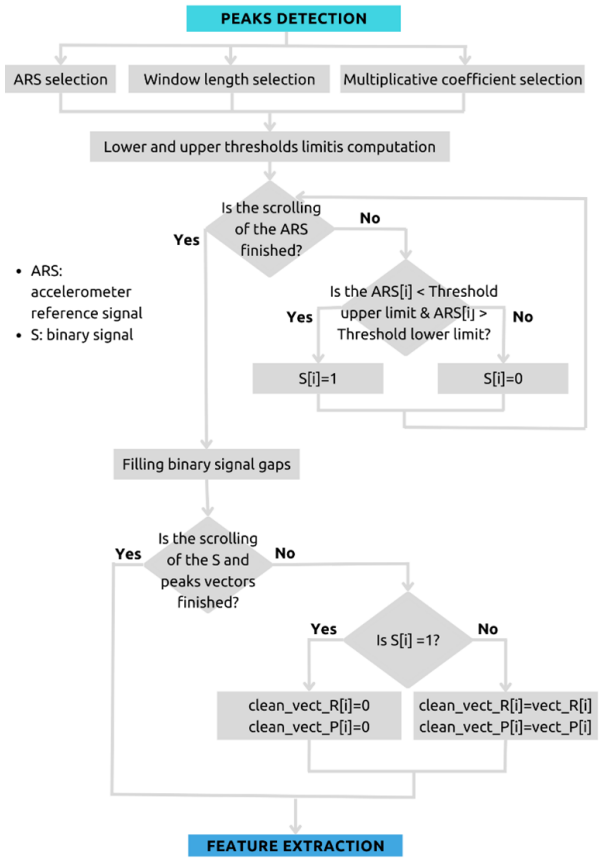

Figure 4.6: Peaks cleaning process

4.2 Graphical User Interface (GUI)

A graphical user interface (GUI) is a type of user interface that facilitates user interaction with systems through graphical icons, visual indicators, buttons, and widgets instead of a command-line interface. Before developing a GUI, is it important to analyze and specify the requirements needed to successfully implement a service. A Service is generally implemented in five major phases, which are the service definition, the service design, the service implementation, the service delivery, and the service decommission [59]. This thesis work focalizes only on the first three phases.

4.2.1 Service Definition

The service definition phase is intended to identify users, requirements, targets, goals, and the overall functionality of the GUI.

Purpose and goals

This GUI is intended to be used in the SINTEC project to increase the useability of the proposed algorithm, to fast the research for the best results, and to enhance the transferability of the whole BP evaluation process from the Shimmer devices to SINTEC devices.

User requirements

This GUI is intended to be used by researchers, engineers, or doctors, who are not supposed to possess programming skills but who are familiar with the steps needed to evaluate BP through ECG and PPG signals. The major need is a tool that allows an easy way to visually determined the best ECG and PPG thresholds for peaks detection, for the final goal of obtaining the best BP estimation in terms of MAE and standard deviation. It is in the interest of the researcher the possibility to visualize the SBP and DBP predictions in contrast with the reference signal extrapolated from the Omron device. Furthermore, researchers are interested in strategies allowing the reduction of the effects of MA on the prediction of BP. At the same time, since researchers could come from different backgrounds, is it important to guide the user through the possible choices with default setups and limited possible choices. The GUI should have the possibility to save the results obtained and allow multiple consecutive sessions of signal analysis and BP estimation.

System functionality

The GUI should allow the uploading of previously recorded .mat files which are the ECG, PPG, and accelerometer signals, and should allow also the uploading of a .csv

file containing the Omron data. The output data of the service deployed by the GUI should be a .csv file containing at least the HR feature, statistical parameters of the analyzed session, and the coefficients for BP determined. The GUI should allow the visualization of the ECG, PPG signals with relatives determined peaks, acceleration signals, BP in terms of predicted and real. The GUI should give the possibility of selecting the best filtering systems, to allow users to select the best thresholds for peaks detection through visual inspection. The user should be allowed to decide if utilize or not strategies for motion artifact cleaning.

Technical specifications The GUI is developed for working at least on a personal computer with Windows 11 operating system installed and the python 3.3 version installed. The personal computer should have at least enough data storage available to store the previously recorded signals and the .csv file elaborated from the GUI.

4.2.2 Service Design

The service design phase is intended to determine the layout and visual appearance of the GUI, and define GUI widgets, and interactions for deploying the requirements. Unified Modelling Language (UML) diagrams are a valid visual tool for identify the requirements previously identified.

Synopsis diagram

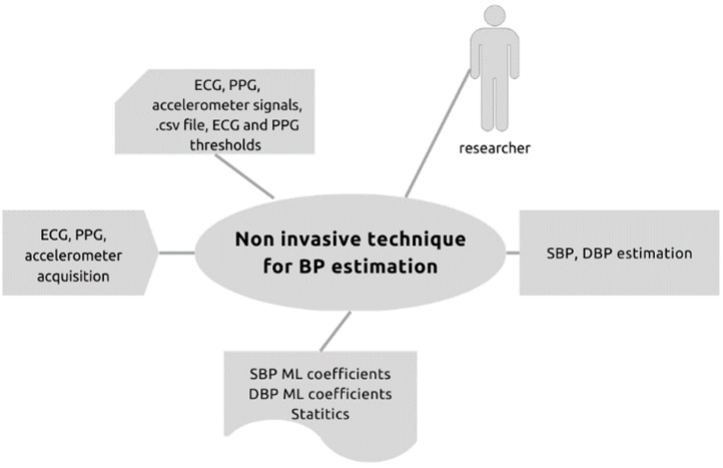

Figure 4.7: Synopsis diagram

The synopsis diagram, Fig 4.7, allows to visually analyze the event that triggers the service intended to deploy, which is the signals acquisition. The diagram shows that the main user in the process is the researcher, the inputs data needed are the signals and the selected values of thresholds for the peaks detection, while the output data are the ML coefficients extrapolated to reach the final goal which is the SBP and DBP estimation with a non-invasive technique for BP monitoring.

Use case diagram

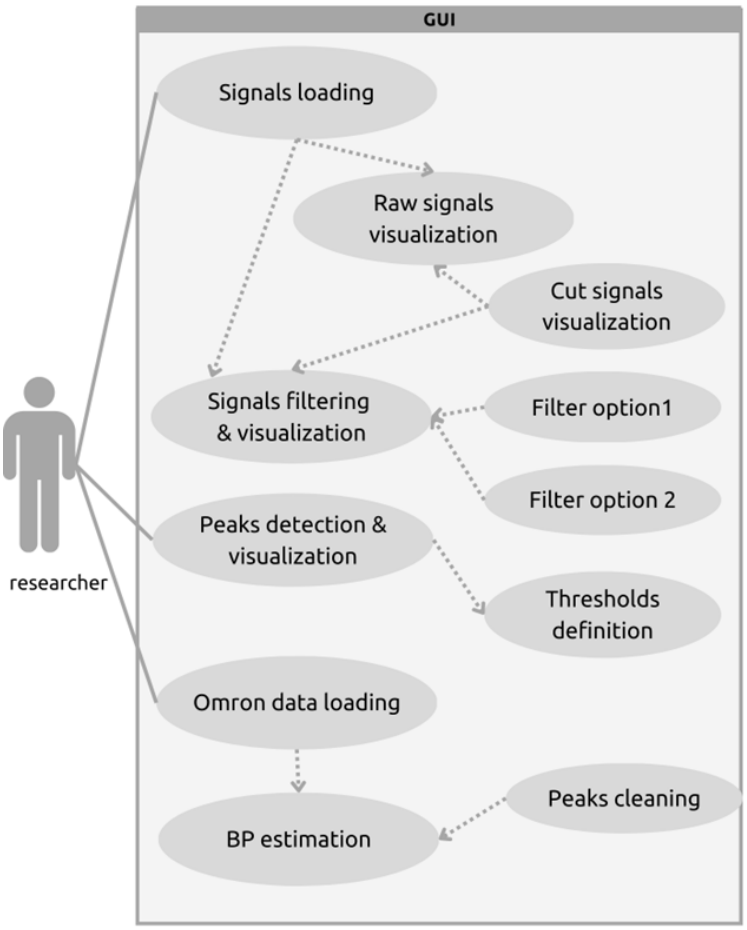

Figure 4.8: Use case diagram

The use case diagram, Fig 4.8, allows to visually capture the system requirements and the interactions between them and the actors. In particular, the diagram shows that the possibility of signals visualization is a ¡include¿ relationship with the GUI functionality of loading signals, which means that signals visualization is

possible only after the user loads the signals. The functionality of visualizing only determined parts of the signal is a ¡extend¿ relationship because it is not strictly necessary for the general functioning of the GUI. The filtering options and the thresholds definitions have also ¡extend¿ relationship because the general functioning of the GUI is assured by default values. The BP estimation has an ¡include¿ relationship with the Omron data loading because Omron data are necessary for the ML technique for BP estimation, whereas the functionality of peaks cleaning has a ¡extend¿ relationship because it is not strictly needed for the general functioning of the GUI.

GUI blueprint and widgets selection

In Fig 4.9, is represented the initial blueprint for the GUI layout and design. Widgets that have been selected for deploying the GUI requirements are:

Figure 4.9: GUI blueprint with widgets selection

Buttons

Buttons allow a specific functionality.

- 1: Button “UPLOAD”, for uploading ECG, PPG, accelerometer signals, and Omron file
- 2: Button “SAVE”, for saving data
- 3: Button “CLEAR”, for clearing everything before starting a new session

- 14: Button "PLOT", for visualizing cut signals and signals with the options selected with the menu 11, and entry boxes 12, 13.
- 24: Button "PREDICT", for predicting BP with the previous selections

Labels

- 4: Texted information showing the .mat file names loaded
- 15: Texted information showing the .csv file name loaded
- 16: Statics data about the BP estimation

Entry Box

User input by keyboard.

- 5: Plot signal from a certain minute in a certain format
- 6: Plot signal to a certain minute in a certain format
- 12: ECG threshold
- 13: PPG threshold

Radio buttons

For selection options where at least one selection is needed.

- 7: For visualizing raw signals
- 8: For visualizing filtered signals
- 9: For visualizing the first filtering option
- 10: For visualizing the seconds filtering option
- 11: For visualizing R & S peaks
- 20: For selecting the threshold 1 for peaks cleaning
- 21: For selecting the threshold 2 for peaks cleaning

Menuboxes

For user selection in a restricted number of options.

- 11: For selecting the filter length of the adaptive filter
- 22: For selecting the window duration
- 23: For selecting the standard deviation

Checkboxes

For user selection, when a selection is not strictly needed.

- 17: For visualizing the real SBP
- 18: For visualizing the real DBP
- 19: For enabling peaks cleaning

Plots

- 25: ECG signal visualization
- 26: PPG signal visualization
- 27: SBP visualization
- 28: DBP visualization

Chapter 5

Results

Signals have been processed with the ECG and PPG thresholds shown in Table 1 of Appendix B, the results of the new algorithm testing contribute to the enlargement of the already existing database of signals recorded with Shimmer devices for the SINTEC project, resulting in a final database of 90 measurements recorded across 12 subjects. The new acquisitions have been analyzed through the new GUI deploying the strategies for MA removal. The tables in Appendix C show all the strategies and parameters that have been used.

5.1 Algorithm test

The ANSI/AAMI/ISO 81060-2:2018 Universal Standard for the Validation of Blood Pressure Measuring Devices is the reference standard for evaluating the accuracy of this algorithm for continuous non-invasive BP monitoring [60]. The guideline provides two criteria, Criterion1 refers to the accuracy and precision requirements that the device must meet when compared to a reference measurement method, whereas Criterion 2 refers to testing the device in a variety of conditions [61]. For the aim of this thesis work, only Criterion 1 has been investigated. According to the population number of this case study (90 measurements), is it possible to refer to the guideline error tolerance for a general population with at least 85 measurements, which states a maximum MAE tolerance of 10 mmHg and a maximum SD tolerance of 8 mmHg [60]. However, it is necessary to specify that, the control device used for this thesis work, the Omron Heart Guide device, has an accuracy of $\pm$ 3 mmHg, which leads, according to the error propagation theory [62], to considering valid measurements with:

$$\begin{cases} MAE < 7mmHg \\ SD < 8mmHg \end{cases} \tag{5.1}$$

The number of validated measurements for the already existing Shimmer measurements and the new measurements according to 5.1 is shown in Table 5.1 :

Table 5.1: Number of valid measurements according to ANSI/AAMI/ISO 81060-2:2018 and error propagation theory in the already existing Shimmer database and the new measurements

	Valid already existing Shimmer measurements	Valid new Shimmer measurements	% Total of valid measurements
MAE, SD 5.1 according to the error propagation theory	50/50	33/40	92%

It is worth noticing that the average MAE from the new measurements is superior to the average MAE from the already existing measurements, while there are no apparent differences in terms of average SD, as shown in Table 5.2 .

Table 5.2: Average MAE and Average SD calculated on the 50 already existing measurements and the 40 new measurements of the Shimmer database

	Already existing Shimmer measurements	New Shimmer measurements
Average MAE (mmHg) ± Average SD (mmHg)	1.76 ± 1.67	4.30 ± 1.79

There are some explanations, the first one refers to the signal quality that has decreased after one year of non-use between the two acquisition sessions. During signals acquisition in the second session, signal quality, especially for ECG signals was lower, resulting in attenuated signals with R peaks generally being no higher than 0.2 mV, instead of 1.25 mV as in the first session. This attenuation may have increased errors in detecting the features necessary for BP estimation. The second explanation refers to the voluntary movements introduced during signals acquisition that introduce rapid variation in BP values that are not detectable with one-minute Omron measurements. The interpolations of the same Omron measurement over a minute when a MA was present, may have caused a major difference in terms of MAE. In addition, an increase in MAE could be due to the loss of accuracy of the Omron itself, caused by the deterioration of mechanical parts due to its oscillometric nature. Overall, in terms of the ANSI/AAMI/ISO 81060-2:2018 guideline, the algorithm has shown acceptable accuracy.

5.2 MA reduction strategies test

Between the 40 new database acquisitions processed with the original algorithm, 16 had returned values of MAE and SD that were not compliant with ANSI/AAMI/ISO 81060-2:2018 guideline and the error propagation theory 5.1. Thus, the MA reduction techniques deployed through the GUI have allowed reducing the number of non-valid measurements from 16 to 7 and allowed a general average MAE improvement of 25% considering all 40 measurements.

Table 5.3: MA removal techniques improvements

	SBP MAE (%)	DBP MAE (%)	Mean value of SBP and DBP MAE (%)
Improvement (%)	21	21	25

As shown in 5.3, the new measurements have beneficiated a 21% reduction of MAE in both SBP and DBP estimations. Considering for each measurement the balance improvement, which is the mean value between the SBP MAE and DBP MAE improvement, the measurements have beneficiated from a general balance improvement of the 25%. Table 3 of Appendix C shows all the improvements for each measurement. These results have been obtained by processing the signals first with the GUI setup corresponding to the original algorithm and then investigating the best combination of MA reduction technique and parameters to obtain an error reduction in terms of MAE. Seven combinations have been analyzed, the first three combinations are entirely based on MA removal through thresholding with different time windows, a combination is the use of the adaptive filter, and the other three techniques are a combination of adaptively filtered signals with MA removal through thresholding with different time windows. For each combination, some parameters can be settled specifically for that measurement. These parameters are the accelerometer signal of reference used in thresholding, the adaptive filter length, and the standard deviation used for determining the upper and lower limit in the thresholding. The seven combinations are:

1. Default filter + thresholding with the length of a window corresponding to the full signals length

2. Default filter + thresholding with a window length corresponding to one minute

3. Default filter + thresholding with a window length corresponding to 30 seconds

4. Adaptive filter

5. Adaptive filter + thresholding with a window length corresponding to the full signals length

6. Adaptive filter + thresholding with a window length corresponding to one minute

7. Adaptive filter + thresholding with a window length corresponding to 30 seconds

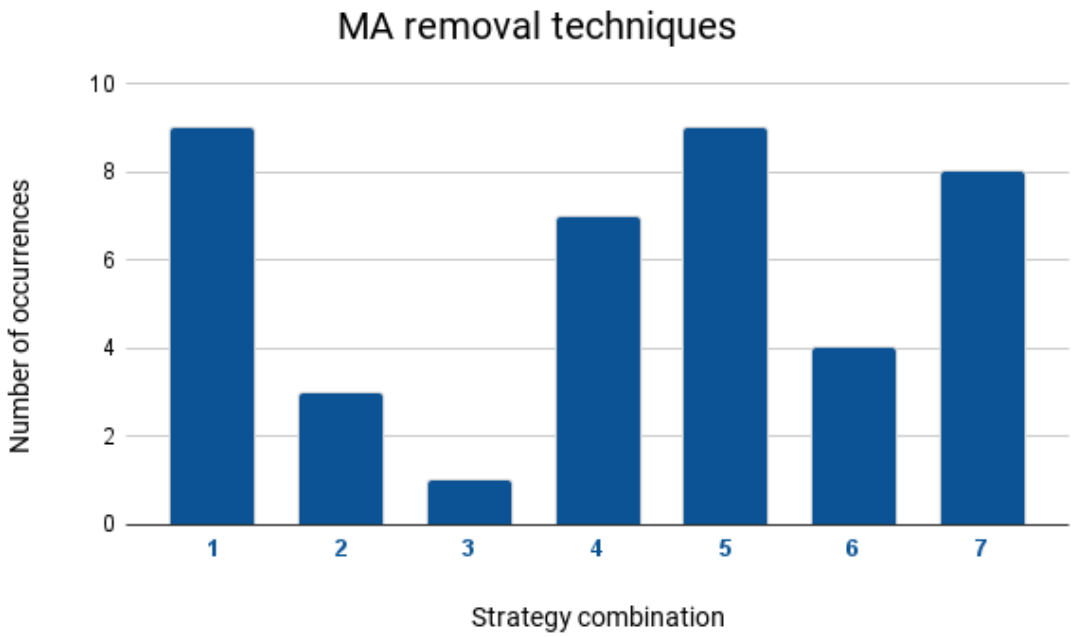

Figure 5.1: MA removal techniques. The x-axis represents the number associated with each of the analyzed strategies, while the y-axis represents the number of times each strategy was found to be better in terms of MAE.

From Fig 5.1 is possible to notice that combinations 1 and 5 have given the best results for almost half of the 40 measurements (In Fig 5.1, 41 measurements have been considered because one measurement has the best results with two of the possible combinations). Both combinations 1 and 5 use the peaks cleaning technique with a window length corresponding to the full signal length. Also, combinations 7 and 4 have returned good results, both techniques foresee at least one adaptively filtered signal but combination 7 uses also the peaks cleaning technique with a window length of 30 seconds, while combination 5 does not use this technique. It is possible to conclude that both the techniques proposed, the peaks cleaning and the adaptive filter, have equally contributed to reducing the MA effects in

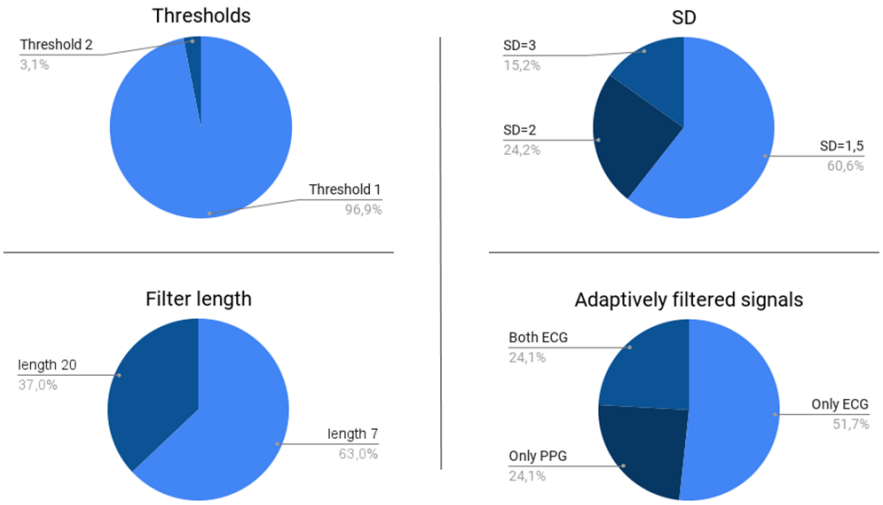

Figure 5.2: The four pie charts represent the user-selectable parameters for finding the best combination in terms of MAE. The parameters in question are the type of threshold to use, the standard deviation to determine the upper and lower limits for thresholding, the length of the filter, and which signal to filter with the LMS filter.

terms of MAE for BP estimation. As mentioned above, each combination was customized through the GUI functionalities with some params choices, for example, in combinations that exploit the peaks cleaning technique, it is possible to select the accelerometer signal of reference between Threshold 1 (ACC norm) and Threshold 2 (SVD decomposition) and the SD necessary for determining the upper and lower limit. For combinations that exploit the adaptive filter, it is possible to select which signal to filter (ECG, PPG, or both) and the filter length. From Fig 5.2 it is possible to notice that Threshold 1 has returned the best results in almost all the measurements. SD set to 1.5 and the filter length equal to 7 have been used more often than the other parameters. In almost half of the measurements that have used the adaptive filter, both ECG and PPG signals have been adaptively filtered.

5.3 ML coefficients analysis

Based on the previous study conducted using the Shimmer database, a statistical analysis was performed on the regression coefficients of the new measurements to investigate whether the values for the same subject remained relatively constant over time. The statistical analysis was conducted on the measurement of the subject

which the major number of measurements and using the mean value Eq. 5.2 and the variance Eq. 5.3 as statistical tools. The mean value (µ) is defined as the central tendency of a numerical distribution and is calculated as the average value of the distribution [63], that in this study case is the average value of a regression coefficient, estimated through:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.2}$$

The variance (σ^2) is a measure of the dispersion of a numerical distribution from its mean value and is defined as the quadratic deviation of the distribution from the arithmetical mean [64]:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2 \tag{5.3}$$

The statistical analysis was conducted over a 24-year-old woman, subject 7, which was the subject with the major number of measurements, and compared with the statistical analysis previously conducted on subjects 1 and 3.

Table 5.4: Mean value of regression coefficients of subject 7

MEAN (μ)	a_{sd}	b_{sd}	a_d	b_d
	4.73	0.04	4.73	0.04

Table 5.5: Variance of regression coefficients of subject 7

VARIANCE (σ^2)	a_s	b_s	a_d	b_d
	164.43	0.01	164.43	0.01

In Table 5.4 and Table 5.5 are illustrated the mean values and variances calculated from the coefficients of subject 7. As previously mentioned, the coefficients refer to Eq. 2.3 from Section 2.3, a_s and a_d refer to the PTT features whereas b_s and b_d refer to the HR feature. As expected, coefficients related to PTT are affected with a higher variability than the coefficients associated with the HR feature, which is extremely low.

In Table 5.6 is illustrated how the variance of the regression coefficients related to ante- meridiem (AM) and post-meridiem (PM) recordings. It is possible to notice that the variability of as , ad coefficients from PM signal acquisitions is higher, almost double respect the variance of as , ad coefficients in AM recording. BP variability can be due to different factors, including the proximity to meals [65] as in most PM acquisitions that have been acquired or immediately after lunch

Table 5.6: Variance of regression coefficients related to AM and PM recording of subject 7

	Variance σ^2			
	a_s	b_s	a_d	b_d
AM	114.87	0.014	114.87	0.014
PM	201.15	0.013	201.15	0.013

or just before dinner. Although a similar behavior can also be observed from the measurements on subjects 1 and 3 [44], the coefficients appear to be substantially different even in subjects with similar BMI. This could indicate that calibrating the algorithm based on BMI may not be a viable approach. These results have been evicted also in the previous study on the Shimmer databases as shown in the tables of Appendix E.

5.4 GUI first release

The previous results have been obtained by analyzing signals with the first GUI prototype, as shown in the following figures.

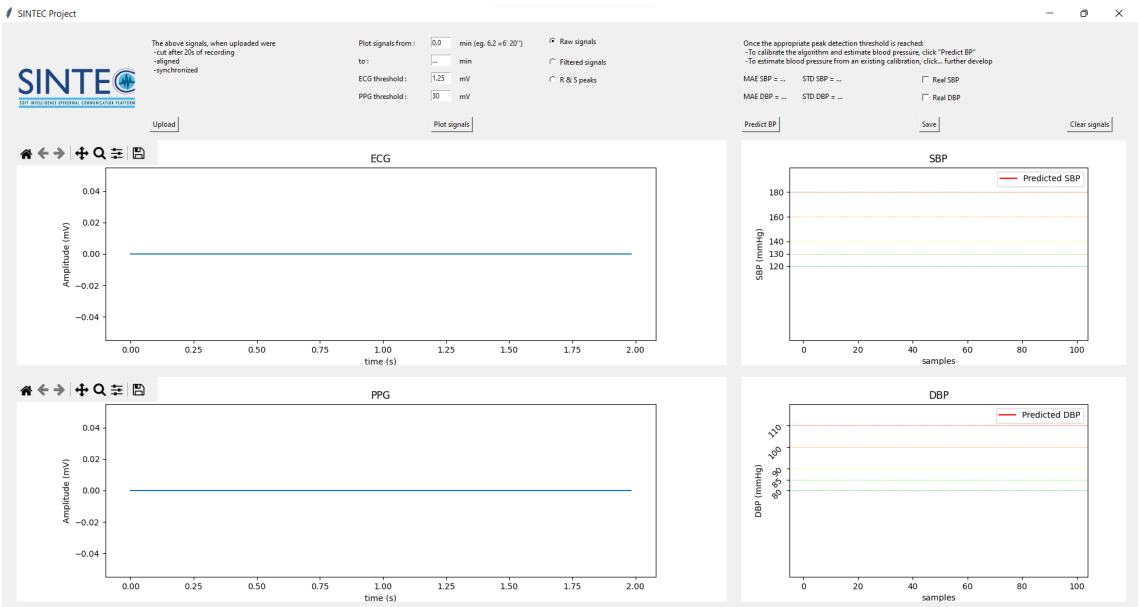

Figure 5.3: GUI first release homepage. Most of the functionalities are disabled because no signals have been uploaded

The GUI homepage, Fig 5.3, appears with a zero signal for ECG and PPG

canvases, and two blank canvases for SBP and DBP. Labels indicate that no signals have been uploaded and for this reason, radio buttons enabling the different visualizations are disabled.

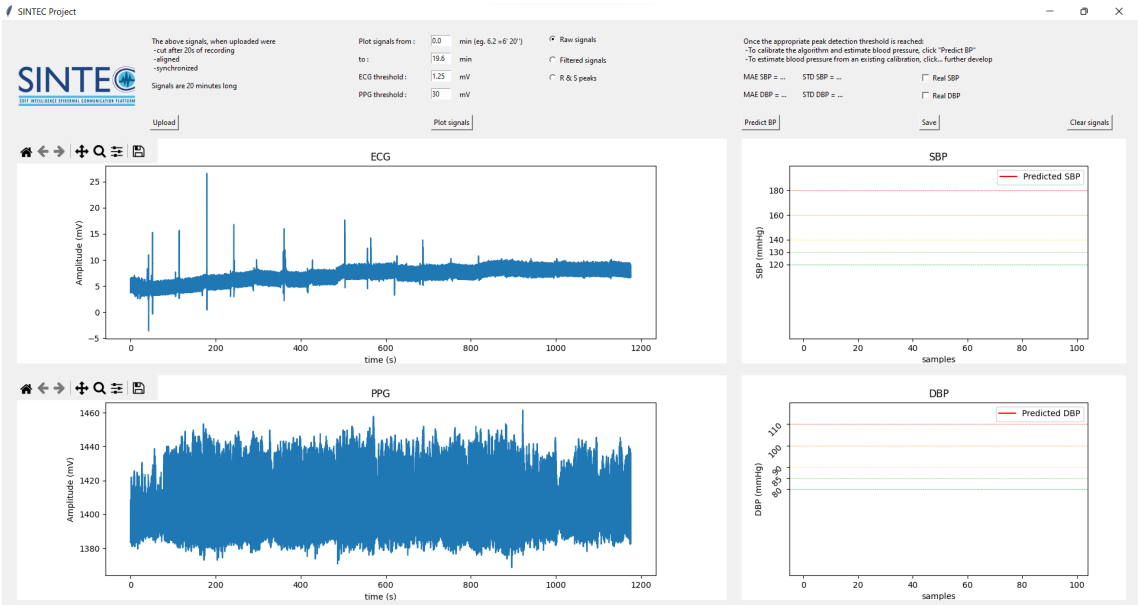

Figure 5.4: Uploading. Users can select which signals upload from their own files

When the button "UPLOAD" is pushed, a file dialog window opens and indicates to the user which signals to upload. When a signal is uploaded, the label updates and indicates which file has been uploaded. In Fig 5.4, it is shown the ECG uploading label updated after and the file dialog indicates to the user to select a PPG signal.

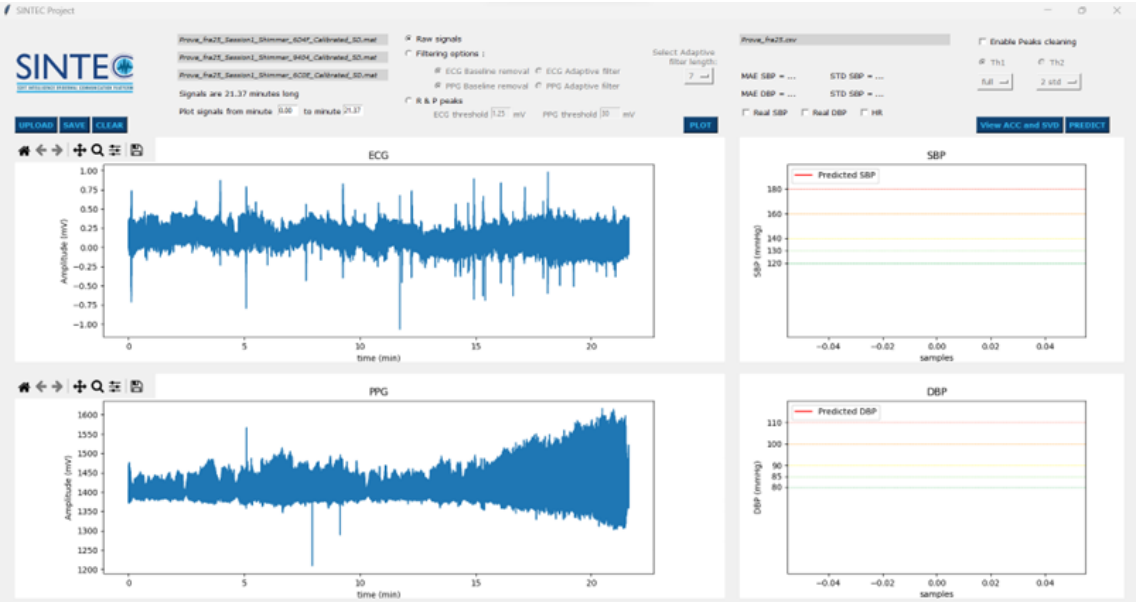

Figure 5.5: The raw signals shown in the canvases are cut, aligned and synchronized ECG and PPG signals

In Fig 5.5, it is shown the result of the uploading. The radio button selected by default indicates that the raw signals are displayed. The label indicating the length of the signals is updated as also the default values in the entry box allowing to see parts of the signals. The radio button options allow the visualization of filtered signals and peaks are enabled.

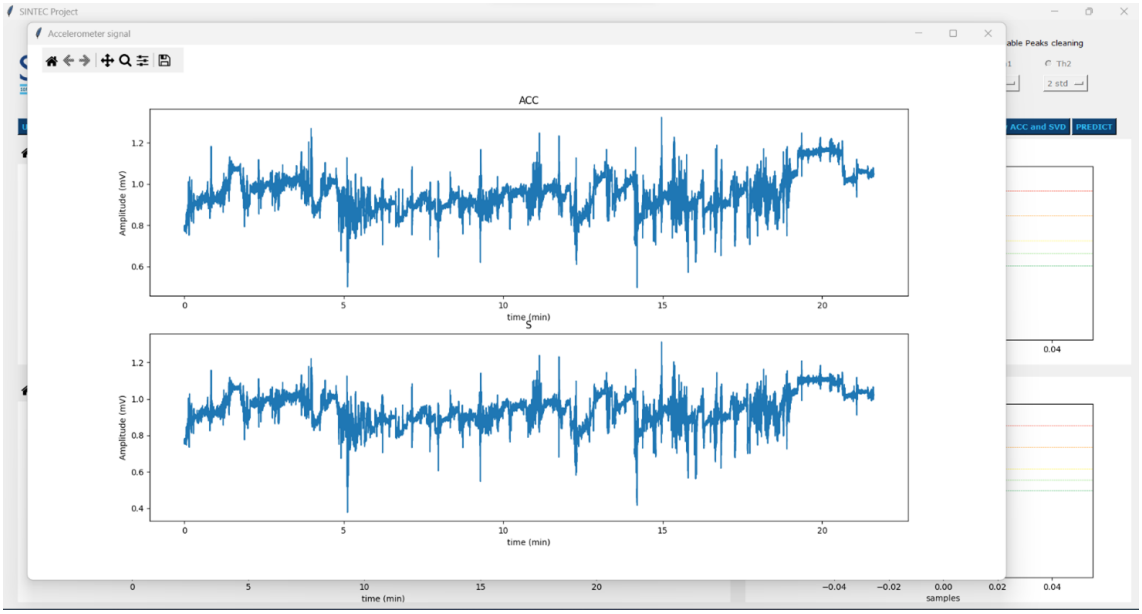

Figure 5.6: The second window allows the users to visualize the two ARS, the ACC norm and the S components of SVD

In Fig 5.6, it is shown the second window opened when button “VIEW ACC and SVD” is pushed. In the first canvas is shown the first accelerometer reference signal, the acceleration norm, whereas the second canvas shows the second accelerometer reference signal, the S components of SVD decomposition. A navigation toolbox simplifies signal visualization.

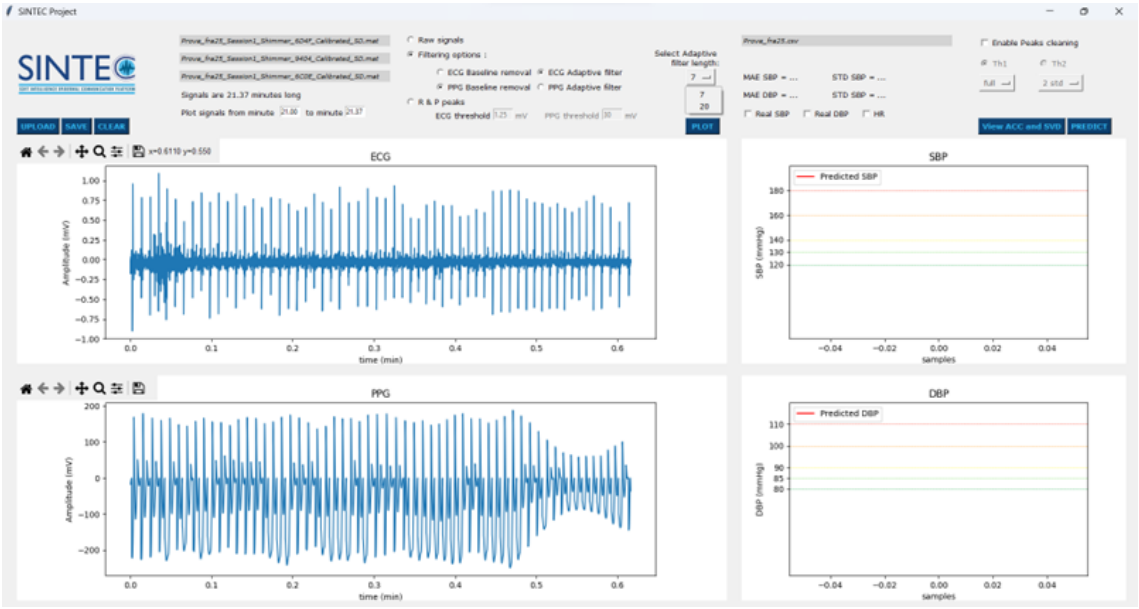

Figure 5.7: The User can select which filtering option returns is most suitable for his case

In Fig 1.7, it is shown how the entry boxes allow the visualization of the same time-specified part of the signals, while the navigation bars allow the user to visualize some part of the respective canvas. It is also possible to notice that, when the filtering option radio button is selected, the filtering options are enabled as the menu allowing to select the adaptive filter length that most suits the signals.

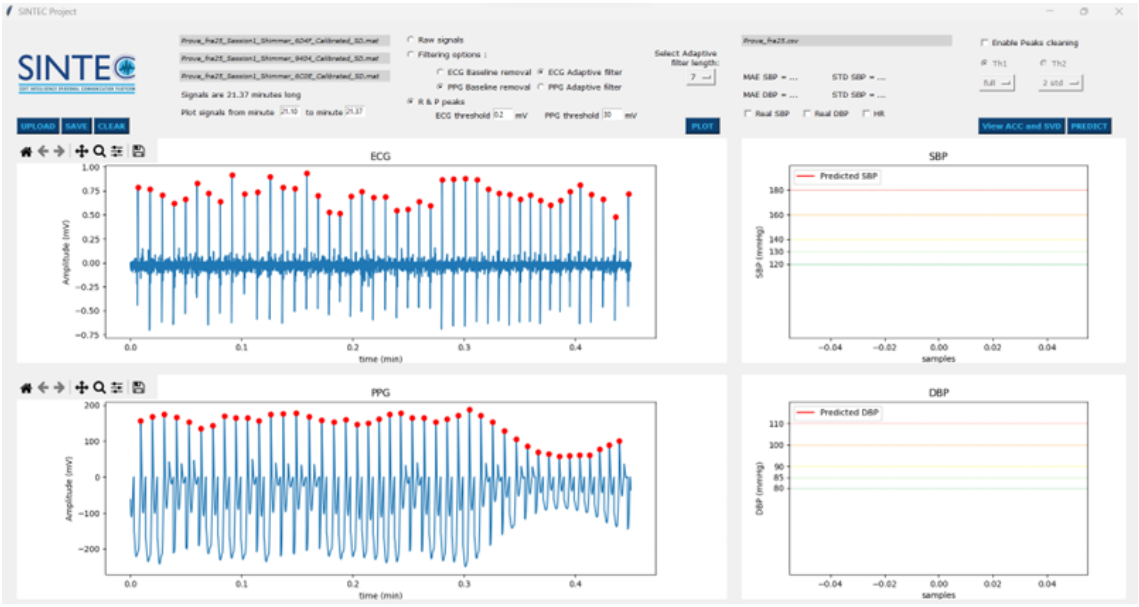

Figure 5.8: The User can manually insert the desired ECG and PPG thresholds

In Fig 5.8 is shown how the GUI allows the visualization of the R and P peaks determined with the threshold specified in the entry boxes. To update the visualization when the values of the thresholds change, it is necessary to click on the button “PLOT”.

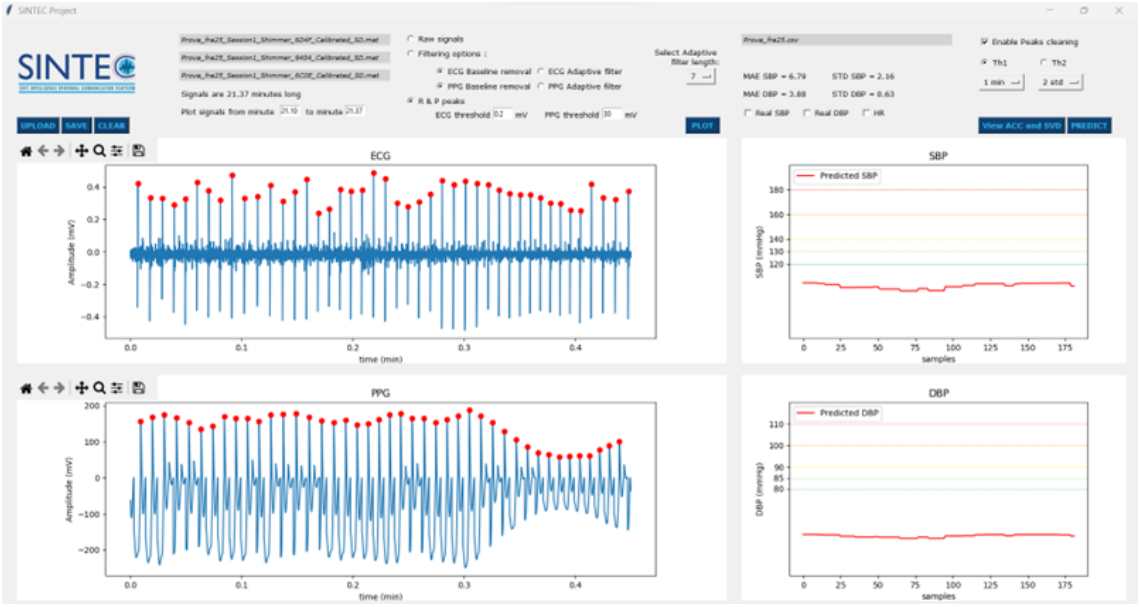

Figure 5.9: The User can manually insert the desired ECG and PPG thresholds

In Fig 5.9, it is shown the results of SBP and DBP predictions in terms of MAE, SD, and plot when the peaks cleaning functionality for MA removal is selected. In particular, the checking of the "Enable peaks cleaning" checkbox enables the selection of the Threshold radio buttons. In this case, Threshold 1 was selected and the user has chosen a window length of one minute for MA removal and a standard deviation set to 2.

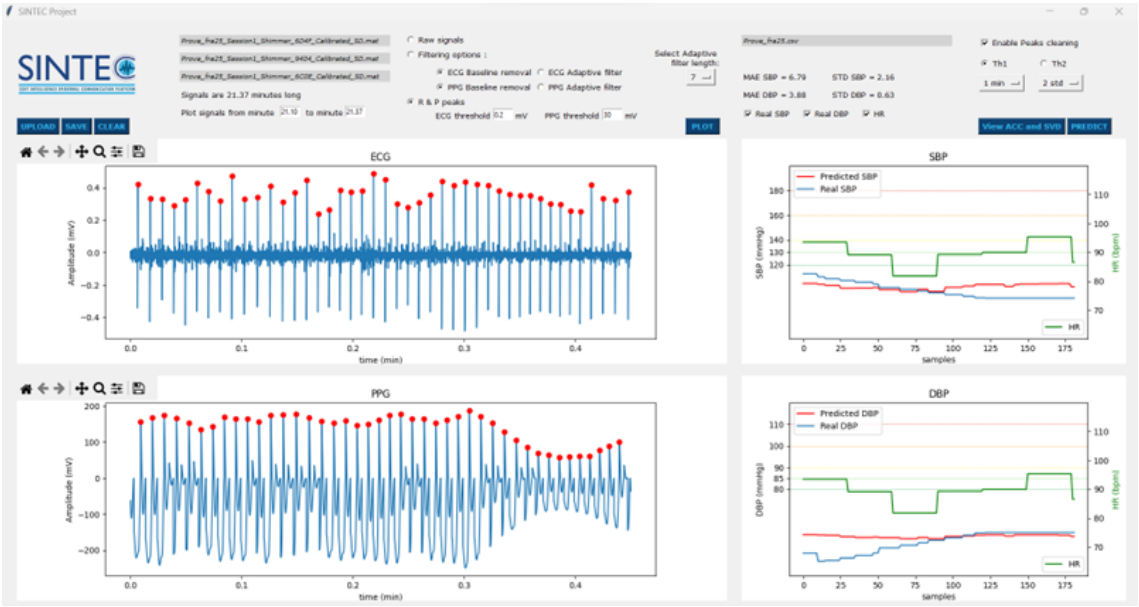

Figure 5.10: The User can check the checkboxes for visualizing also the BP measured by the Omron and the HR

In Fig 5.10, it is shown the visualization of the reference SBP and DBP measured with the Omron and HR on the same plot where the predicted SBP and DBP are visualized when the checkboxes are checked.

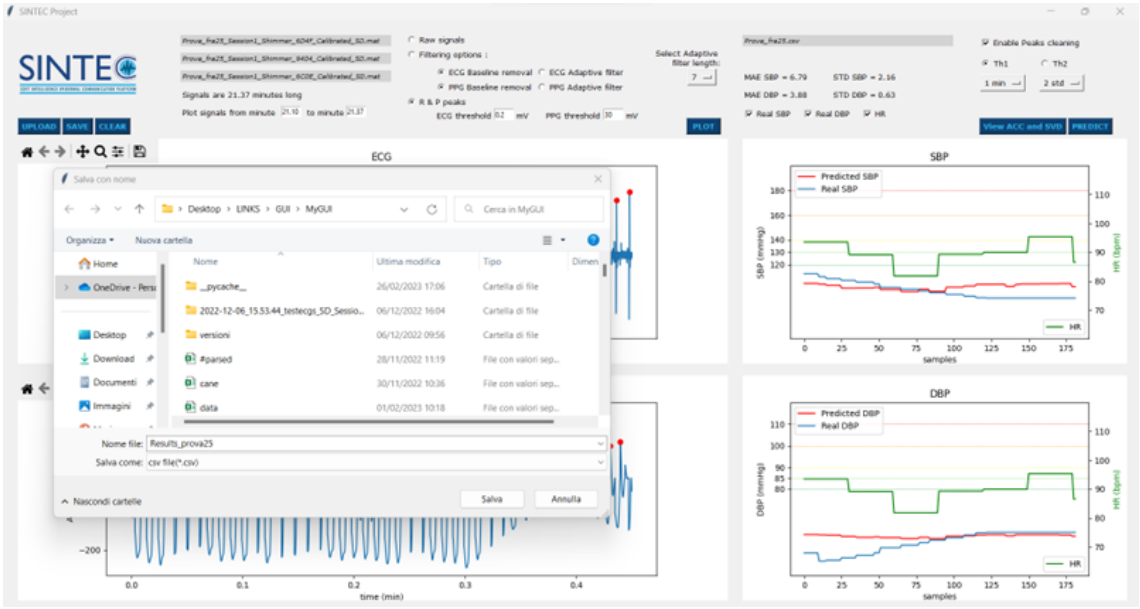

Figure 5.11: The User can save the produced data in a .csv file

In Fig 5.11 is shown the file dialog that is opened when the user clicked on "SAVE" button and allows to save the values of predicted SBP, predicted DBP, HR, MAE and SD of the estimation that has been made in a .csv file.

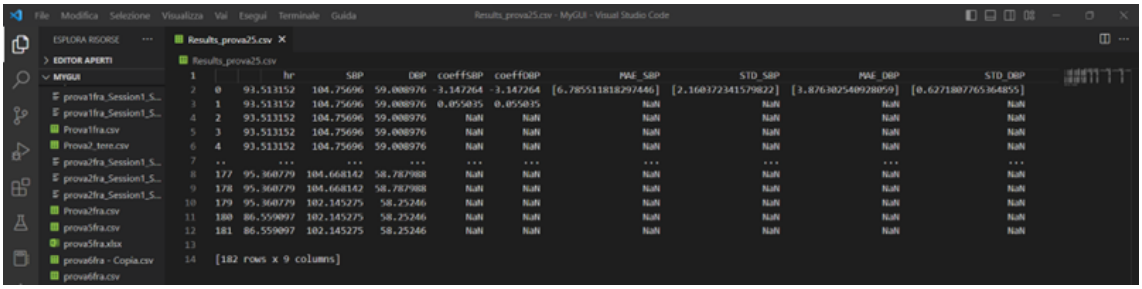

Figure 5.12: Example of CSV file

In Fig 5.12 is shown the .csv file that has just been saved.

Figure 5.13: GUI ready for a new session after a "CLEAR"

In Fig 5.13, it is shown how the GUI is back to the Homepage configuration when the button "CLEAR" is pushed after a session.

Chapter 6

Conclusion

This thesis work aimed to continue the study of a non-invasive continuous BP estimation technique through features extracted from ECG and PPG signals, enlarging the Shimmer database, and obtaining BP values through ML linear regression technique with an acceptable low error according to the AAMI/ISO/ESH guidelines. These results have been obtained on signals corrupted with motion artifacts and processed with MA removal techniques developed in this thesis work and deployed through a user-friendly GUI specifically created for this type of investigation. The novelties introduced in the algorithm by this thesis work foresee the introduction of the accelerometer signal acquisition that has been used for building a technique based on thresholding to eliminate possible corrupted HR and PTT features and to build an LMS adaptive filter. The GUI has been used in the research of the best setup for signal processing, accuracy testing, and coefficient extraction by enabling the different techniques on the same signal analysis session and setting parameters according to the specific case. These techniques for MA removal have been proposed with the scope to be introduced in the future on the wearables SINTEC sensors, which are intended to monitor BP everywhere at every time, during any kind of activity [2].

6.1 Calibration issues and accuracy estimation

An aspect that still requires further investigation is the calibration issue. BMI has been studied to verify if it is possible to calibrate the algorithm with the same coefficients for subjects with similar BMI. However, subjects with similar BMI have shown very different coefficients, indicating that BMI may not be a sufficiently discriminative characteristic to standardize calibration. Currently, calibration is personalized for each subject, but the different variability in AM and PM acquisitions suggests that calibration could be further personalized for the same subject according to the time of day of the measurement. It is worth noticing also that

the algorithm has been tested only for 20 minutes recordings in laboratory while the subject was sitting or doing small movements, it will be interesting to study deeper the calibration issue on SINTEC devices, which are intended for long time measurements and in different outdoor and indoor situations. The question is if the calibration will be still necessary for SINTEC measurements, which is uncomfortable and not feasible to investigate at the moment, with the actual Shimmer devices. Research could be focused also on enhancing methods for evaluating the accuracy of non-invasive continuous BP estimation techniques, overcoming the issue of the non-continuous measurements from control devices [66].

6.2 Toward edge computing and real-time application

A possible future improvement for wearable devices could be the deployment of an edge computing paradigm. This paradigm refers to a decentralized architecture where the IT services are at the "edge" of the network, much closer to the wearables than cloud servers. This paradigm has the potential to power the wearables in terms of processing real-time data and conducting in-depth analysis, extending the battery life of wearables, reducing the latency, and improving the accuracy of data processing. This kind of technology could also lead to the development of a GUI for real-time data visualization, enabling a faster verification of signal quality before an acquisition session, the correct electrodes location, and enabling faster signal processing [67]. An edge computing architecture could also optimize data storage, a critical aspect related to health data, that reach about 50 petabytes per year in a hospital [68], plus the fact that patient health data information are kept within the device and inference at the edge, the security and the vulnerability from attacks and breaches is enhanced.

Appendix A

This appendix contains the informed consent that has been signed by each subject before proceeding with the recording of the physiological signals. The information note relating to the SINTEC project is shown below the informed consent.

CONSENSO INFORMATO

Titolo della ricerca: Implementazione di un algoritmo per il monitoraggio della pressione arteriosa in modo non invasivo partendo dai segnali elettrocardiografici (ECG) e Fotopletismografici (PPG).
Sperimentatore: Dott.ssa Francesca Boschi Tesista magistrale del Politecnico di Torino Tesi svolta presso LINKS Foundation Torino (TO), Italia
Io sottoscritto/a _____ nato/a a _____ il _____
indirizzo _____ telefono _____
Dichiaro
Di partecipare volontariamente allo studio Implementazione di un algoritmo per il monitoraggio della pressione arteriosa in modo non invasivo partendo dai segnali elettrocardiografici (ECG) e Fotopletismografici (PPG) avente lo scopo di Testare l'accuratezza dell'algoritmo per la stima della pressione sanguigna in modo non invasivo nell'ambito del progetto europeo SINTEC.

- Di aver ricevuto dalla dott.ssa Francesca Boschi esaurienti spiegazioni in merito alla richiesta di partecipazione alla ricerca, in particolare sulle finalità e procedure.
- Di aver avuto la possibilità di porre domande e di aver avuto risposte soddisfacenti su tutta la sperimentazione.
- Di essere stato informato sui possibili rischi o disagi ragionevolmente prevedibili.
- Di essere consapevole che la partecipazione è volontaria.
- Di essere stato assicurato:
  - che potrò ritirarmi dalla sperimentazione già iniziata in qualsiasi momento senza l'obbligo da parte mia di motivarne la decisione;

- che i dati saranno utilizzati con le finalità indicate nello studio;
- che è mio diritto avere accesso alla documentazione che mi riguarda;
- che una copia del consenso informato e della documentazione di cui ho preso visione rimarrà in mio possesso;
- che per ogni problema o per eventuali ulteriori informazioni potrò rivolgermi a: Dott.ssa Francesca Boschi

Pertanto, confermo di aver avuto risposte esaurienti a tutti i miei quesiti e, preso atto della situazione illustrata ACCONSENTO

LIBERAMENTE, SPONTANEAMENTE E IN PIENA COSCIENZA ALLA SPERIMENTAZIONE PROPOSTAMI. Dichiaro inoltre di

essere a conoscenza della possibilità di revocare il presente consenso in qualsiasi momento prima, durante e dopo l'avvio della sperimentazione.

Data registrazione	Ora inizio	Ora fine	Firma del partecipante

Eventuali testimoni presenti (nome, cognome, firma):_____

Firma del partecipante: Firma dello sperimentatore:

OPPURE

NON ACCONSENTO

LIBERAMENTE, SPONTANEAMENTE E IN PIENA COSCIENZA ALLA SPERIMENTAZIONE PROPOSTAMI.

Data:_____

Eventuali testimoni presenti (nome, cognome, firma):_____

Firma del partecipante: Firma dello sperimentatore:

NOTA INFORMATIVA

Chi siamo e cosa è il Progetto SINTEC? Il progetto SINTEC- Soft intelligence epidermal communication platform è un progetto europeo nato a giugno del 2019 che si propone di sviluppare una tecnologia innovativa in grado di monitorare la salute di chi la indossa. SINTEC è supportato da un'ampia varietà di organizzazioni con esperti in tutti i settori d'interesse come l'ingegneria, lo studio dei materiali, l'elettronica, la medicina, lo sport, l'economia e la divulgazione. Gran parte delle attività vengono svolte utilizzando un approccio interdisciplinare. Il progetto vuole rispondere alla necessità di sviluppare nuove tecnologie di interconnessione, non invasive e che non interferiscano con la vita del soggetto che le indossa.

I dispositivi indossabili smart sono il passo successivo nell'evoluzione dei dispositivi indossabili di Internet of Things (IoT). Il primo obiettivo che il progetto SINTEC si propone di raggiungere è dimostrare i vantaggi di questa nuova tecnologia nell'ambiente clinico. Per far ciò si sta sperimentando una tecnologia PCB con substrato estensibile e lega liquida con l'integrazione di sistemi embedded complessi all'interno del substrato al fine di applicarla in differenti situazioni complesse della vita di tutti i giorni.

Il contributo di LINKS foundation è orientato all'analisi ed elaborazione dei segnali fisiologici registrati, nello specifico del segnale elettrocardiografico (ECG) e fotopletismografico (PPG). La prima parte del lavoro è focalizzata sulla valutazione delle prestazioni di sensori e antenne tramite confronto con i dispositivi allo stato dell'arte presenti nei laboratori. I sensori esplorati saranno elettrodi per segnali elettrofisiologici (bioimpedenza) per ECG e sensori ottici (LED e fotodiodo) per PPG. Il risultato di queste analisi ha orientato ed orienterà l'architettura del sistema finale che è in continuo aggiornamento. Verranno inoltre integrati e testati firmware e algoritmi per l'estrazione dei parametri fisiologici (es. Frequenza cardiaca, Pressione sanguigna ecc.). I risultati ottenuti dalla validazione forniranno feedback per l'ottimizzazione del dispositivo.

Ulteriori informazioni sul Progetto SINTEC sono disponibili qui: https://www.sintec-project.eu/ Firmando il modulo di consenso ivi allegato, si accetta di partecipare allo studio Implementazione di un algoritmo per il monitoraggio della pressione

arteriosa in modo non invasivo partendo dai segnali elettrocardiografici (ECG) e Fotopletismografici (PPG).

Qual è lo scopo della sperimentazione? Lo studio ha lo scopo di testare l'accuratezza dell'algoritmo per la stima della pressione sanguigna in modo non invasivo partendo dai segnali ECG e PPG registrati con dispositivi wearable (Shimmer3 ECG e GRS+).

Sono obbligato a partecipare? No. La decisione di partecipare alla sperimentazione dipende solo da Lei. È completamente volontaria.

Come si svolgerà la sperimentazione? Ai volontari sarà richiesto di indossare i dispositivi Shimmer e il dispositivo di riferimento Omron Heartguide. Dovranno restare in posizione rilassata per 20 minuti durante i quali verranno registrati i segnali ECG e PPG in maniera continuativa ed i valori di pressione sistolica e diastolica ogni minuto.

Quanto dura la sperimentazione? La Sua partecipazione alla sperimentazione durerà fino alla fine della registrazione dei segnali, i volontari che lo desidereranno potranno partecipare più volte alla registrazione.

Dovrò sostenere spese? No. La Sua partecipazione alla sperimentazione sarà completamente gratuita.

Cosa dovrò fare se decido di partecipare alla sperimentazione? Le sarà consegnata questa nota informativa, da leggere e conservare. Le sarà chiesto di firmare il modulo di consenso, ivi allegato. Le sarà richiesto di dedicarci il tempo necessario per la registrazione dei segnali.

Potrò cambiare idea dopo aver accettato di partecipare? Sì. Lei potrà decidere di ritirare il consenso e interrompere la partecipazione alla sperimentazione, in qualsiasi momento, anche a studio avviato, senza dover fornire giustificazioni. Qualora decidesse di ritirare il consenso, Le chiediamo di inviare una comunicazione al seguente recapito: francesca.boschi@studenti.polito.it e valeria.figini@linksfoundation.com

Come saranno usati i miei dati personali? I Suoi dati personali saranno resi anonimi, nessuna informazione che La identifichi o La renda identificabile, direttamente o indirettamente, o che possa fornire informazioni sulle Sue caratteristiche, le Sue abitudini, il Suo stile di vita, le Sue relazioni personali, la Sua situazione economica, verrà conservata. I segnali registrati verranno elaborati per la stima della pressione sanguigna e i risultati ottenuti non potranno in alcun modo ricondurre a

Lei.

I miei dati saranno sfruttati commercialmente? I Suoi dati non saranno in alcun modo sfruttati commercialmente.

Con chi verranno condivisi i miei dati personali? I Suoi dati personali saranno resi anonimi e solo i risultati da essi ottenuti saranno condivisi con il consorzio SINTEC.

Chi devo contattare nel caso in cui abbia delle domande o reclami da sottoporre? Nel caso in cui avesse domande o reclami relativi alla sperimentazione, può contattare: francesca.boschi@studenti.polito.it e valeria.figini@linksfoundation.com

Quali benefici potrò avere partecipando alla sperimentazione? Lei parteciperà ad uno studio che punta ad avere un impatto rivoluzionario sulla vita dei pazienti per le patologie cardiovascolari. Si spera che fornendo un mezzo di monitoraggio conveniente e affidabile, aumenti il numero di persone che rilevano sistematicamente la propria pressione sanguigna. In questo modo si potrebbe prevenire l'insorgenza o la degenerazione di malattie cardiovascolari che sono ancora oggi la prima causa di mortalità nel mondo.

Cosa accadrà ai risultati della sperimentazione? I risultati della sperimentazione saranno resi anonimi, quindi Lei non sarà identificabile. I risultati anonimizzati saranno utilizzati ai fini del Progetto SINTEC, condivisi con il consorzio SINTEC, in conferenze nazionali e internazionali, e pubblicati su riviste scientifiche.

Verranno effettuate riprese fotografiche o videografiche durante la mia partecipazione alla sperimentazione? Durante la Sua partecipazione alla sperimentazione, non saranno scattate fotografie e non saranno effettuati filmati delle sessioni di test.

Quali potrebbero essere i rischi? Il prelievo dei segnali avviene in modo non invasivo ed indolore, si percepirà solo una pressione al polso su cui verrà posizionato il dispositivo Omron HeartGuide (smartwatch con cuffia che restituisce i valori pressori gonfiandosi e sgonfiandosi). I rischi relativi alla fase di prelievo sono quelli legati all'eventuale mal posizionamento dei sensori con conseguente registrazione di segnali non utilizzabili. In ogni caso il soggetto potrà scegliere in totale libertà di sottoporsi nuovamente alla sperimentazione o meno.

Appendix B

This appendix shows the table containing the peaks detection thresholds, both for ECG and PPG signals, related to all measurements.

Table 1: ECG and PPG thresholds

THRESHOLDS (mV)		
Measurement	**ECG**	**PPG**
51	0.1	30
52	0.4	30
53	0.2	30
54	0.2	30
55	1	30
56	0.1	15
57	0.11	30
58	0.2	10
59	1	30
60	0.4	30
61	1	30
62	0.1	30
63	0.4	30
64	0.5	20
65	0.2	30
66	0.2	30
67	0.4	100
68	0.1	20
69	1	30
70	0.6	30
71	0.4	30
72	0.5	30
73	0.5	10
74	0.1	30
75	0.2	30

Table 1 – ECG and PPG thresholds

Measurement	**ECG**	**PPG**
76	0.2	30
77	0.5	80
78	0.1	30
79	0.15	30
80	0.2	50
81	0.1	20
82	0.1	10
83	0.1	5
86	0.1	10
87	0.5	10
88	0.4	1
89	0.2	5
90	0.2	20

Appendix C

This appendix shows the MAEs and the SDs related to all measurements and indicates which combination has been used to obtain the best results and the relative parameters that have been used. It is also shown the percentage improvement in terms of SBP and DBP MAE and the average percentage improvement for each measurement. The following bullet list shows the number corresponding to each used strategy along with its description.

1. Default filter+thresholding with a windows length corresponding to the full signals length
2. Default filter+thresholding with a windows length corresponding to one minute
3. Default filter+thresholding with a windows length corresponding to 30 seconds
4. Adaptive filter
5. Adaptive filter+thresholding with a windows length corresponding to the full signals length
6. Adaptive filter+thresholding with a windows length corresponding to one minute
7. Adaptive filter+thresholding with a windows length corresponding to 30 seconds

Table 2: MAEs and the SDs related to all measurements with the relative strategy

MAE: MEA (mmHg) ± SD (mmHg)			
Measurement	**Strategy**	**SBP**	**DBP**
51	1	3.62 ± 1.08	2.65 ± 1.14
52	5	1.44 ± 0.33	1.15 ± 1.01
53	7	6.62 ± 6.32	6.97 ± 2.75
54	6	4.98 ± 1.14	3.01 ± 0.45
55	5	1.39 ± 0.39	0.87 ± 0.26

Table 2 – MAEs and the SDs related to all measurements with the relative strategy

Measurement	**Strategy**	**SBP**	**DBP**
56	3	14.77 ± 0.79	7.61 ± 1.92
57	7	4.34 ± 0.60	2.65 ± 0.95
58	7	1.27 ± 0.49	1.18 ± 0.72
59	4	5.17 ± 3.28	3.39 ± 2.13
60	6	7 2.00 ± 3.06	4.31 ± 1.53
61	6	0.39 ± 0.42	0.51 ± 0.24
62	1	6.47 ± 2.22	6.65 ± 1.52
63	5	3.69 ± 1.06	2.66 ± 1.09
64	5	2.62 ± 0.60	1.50 ± 0.70
65	7	6.41 ± 6.44	7.08 ± 2.75
66	1	4.55 ± 1.64	3.22 ± 3.46
67	7	1.91 ± 2.39	2.71 ± 1.35
68	5	8.51 ± 3.80	5.06 ± 5.08
69	1	4.91 ± 1.15	2.78 ± 1.18
70	5	1.53 ± 1.14	1.97 ± 0.67
71	4	2.13 ± 1.25	1.92 ± 0.67
72	2	7.1 ± 9.33	3.84 ± 4.04
73	7	3.65 ± 1.42	3.71 ± 1.01
74	2	3.25 ± 1.55	1.65 ± 0.79
75	2	6.79 ± 2.16	3.88 ± 0.63
76	1	2.38 ± 1.16	1.37 ± 0.61
77	7	2.32 ± 2.24	4.89 ± 3.70
78	1	3.37 ± 1.58	3.41 ± 1.84
79	4	5.03 ± 0.88	2.20 ± 0.35
80	6	5.01 ± 1.53	2.75 ± 0.52
81	1	4.27 ± 1.47	4.12 ± 1.41
82	4	1.89 ± 1.02	5.72 ± 1.60
83	4	6.61 ± 1.30	2.04 ± 0.47
84	4	5.24 ± 0.49	5.93 ± 0.83
85	4	14.39 ± 2.64	3.38 ± 0.46
86	1	12.25 ± 3.38	8.22 ± 5.28
87	1	4.7 ± 1.51	4.94 ± 0.94
88	5	4.83 ± 0.61	4.43 ± 2.69
89	5	2.23 ± 0.87	10.86 ± 0.73
90	5	5.07 ± 4.51	7.80 ± 4.80

Table 3: Parameters used for a strategy related to a measurement

Parameters				
Measurement	**Strategy**	**Filter length**	**Th**	**SD**
51	1	-	1	1.5
52	5	7	1	2
53	7	7	1	1.5
54	6	20	1	3
55	5	20	1	1.5
56	3	-	1	3
57	7	7	1	2
58	7	20	1	1.5
59	4	7	-	-
60	(6 and 7)	7	1	3
61	6	20	1	1.5
62	1	-	(1 and2)	3
63	5	7	1	1.5
64	5	20	1	2
65	7	7	1	1.5
66	1	-	1	1.5
67	7	7	1	1.5
68	5	20	1	1.5
69	1	-	2	2
70	5	20	1	1.5
71	4	20	-	
72	2	-	1	3
73	7	7	1	1.5
74	2	-	1	1.5
75	2	-	1	2
76	1	-	1	1.5
77	7	7	1	1.5
78	1	-	1	2
79	4	7	-	-
80	6	7	1	1.5
81	1	-	1	1.5
82	4	7	-	-
83	4	20	-	-
84	4	20	-	-
85	4	7	-	-
86	1	-	1	2
87	1	-	1	2

Table 3 – Parameters used for a strategy related to a measurement

Measurement	**Strategy**	**Filter length**	**Th**	**SD**
88	5	7	1	1.5
89	5	7	1	1.5
90	5	7	1	1.5

Table 4: Improvements in terms of MAE

Improvements				
Measurement	**Strategy**	**%SBP impr.**	**%DBP impr.**	**% avg impr.**
51	1	2%	4%	3%
52	5	26%	20%	23%
53	7	35%	7%	21%
54	6	28%	36%	32%
55	5	58%	38%	48%
56	3	2%	4%	3%
57	7	25%	52%	39%
58	7	46%	20%	33%
59	4	41%	57%	49%
60	6 & 7	12%	18%	15%
61	6	83%	87%	85%
62	1	0%	5%	3%
63	5	16%	17%	17%
64	5	45%	74%	60%
65	7	48%	7%	28%
66	1	24%	30%	27%
67	7	36%	3%	20%
68	5	31%	36%	34%
69	1	5%	-2%	2%
70	5	89%	77%	83%
71	4	27%	-9%	9%
72	2	-1%	13%	6%
73	7	82%	-106%	-12%
74	2	23%	51%	37%
75	2	33%	-21%	6%
76	1	40%	43%	42%
77	7	54%	68%	61%
78	1	9%	2%	6%
79	4	15%	14%	15%
80	6	31%	42%	37%

Table 4 – Improvements in terms of MAE

Measurement	**Strategy**	**%SBP impr.**	**%DBP impr.**	**% avg impr.**
81	1	65%	-24%	21%
82	4	54%	-57%	-2%
83	4	0%	48%	24%
84	4	6%	11%	9%
85	4	14%	6%	10%
86	1	13%	33%	23%
87	1	-19%	36%	9%
88	5	0%	46%	23%
89	5	70%	12%	41%
90	5	58%	35%	47%

Appendix D

This appendix shows tables with the values of the regression coefficients related to all subjects from whom more than two valid measurements were recorded and tables showing the mean and variance values of the coefficients themselves.

Table 5: Regression coefficients subject 7

	Measurement	a_s	b_s	a_d	b_d
Subject 7	51	-8.37	0.12	-8.37	0.12
Female					
24 y/o	52	10.61	-0.03	10.61	-0.03
BMI: 19.83	53	-2.59	-0.15	-2.59	-0.15
	54	14.36	-0.02	14.36	-0.02
	55	19.62	-0.03	19.62	-0.03
	56	2.12	0.41	2.12	0.4
	57	10.54	0.03	10.54	0.03
	58	-4.74	-0.04	-4.74	-0.04
	59	23.76	0.23	23.76	0.23
	60	-6.61	-0.09	-6.61	-0.09
	61	-6.61	-0.09	-6.61	-0.09
	62	3.64	0.06	3.64	0.06
	63	-29.36	0.10	-29.36	0.10
	64	18.60	0.01	18.60	0.01
	65	3.93	0.10	3.93	0.10
	66	30.15	-0.22	30.15	-0.22
	67	20.06	0.21	20.06	0.21
	68	-2.59	-0.15	-2.59	-0.15
	69	0.57	-0.12	0.57	-0.12
	70	-8.37	0.12	-8.37	0.12
	71	10.61	-0.03	10.61	-0.03
	72	7.15	0.04	7.15	0.04
	73	44.38	0.09	44.38	0.09
	74	-0.41	0.14	-0.41	0.14

Table 5 – Regression coefficients subject 7

	Measurement	a_s	b_s	a_d	b_d
	75	-6.60	-0.09	-6.60	-0.09
	76	-3.15	0.05	-3.15	0.05
	77	6.86	0.01	6.86	0.01
	78	27.06	0.31	27.06	0.31
	79	19.50	0.18	19.50	0.18
	80	-3.66	0.09	-3.66	0.09

Table 6: Mean value of regression coefficients of subject 7

MEAN (µ)			
a_s	b_s	a_d	b_d
4.73	0.04	4.73	0.04

Table 7: Variance of regression coefficients of subject 7

VARIANCE (σ^2)			
a_s	b_s	a_d	b_d
164.43	0.01	164.43	0.01

Table 8: Regression coefficients subject 8

	Measurement	a_s	b_s	a_d	b_d
Subject 8 Female	81	-8.26	-0.33	-8.26	-0.33
24 y/o BMI: 21.8	82	11.34	0.3	11.34	0.3

Table 9: Mean value of regression coefficients of subject 8

MEAN (µ)			
a_s	b_s	a_d	b_d
9.80	-0.02	9.80	-0.02

Table 10: Variance of regression coefficients of subject 8

VARIANCE (σ^2)			
a_s	b_s	a_d	b_d
4.74	0.20	4.74	0.20

Table 11: Regression coefficients subject 9

	Measurement	a_s	b_s	a_d	b_d
Subject 9 Female	83	7.96	-0.19	7.96	-0.19
24 y/o BMI: 19.83	84	17.52	0.02	17.52	0.02

Table 12: Mean value of regression coefficients of subject 9

MEAN (µ)			
a_s	b_s	a_d	b_d
12.74	-0.09	12.74	-0.09

Table 13: Variance of regression coefficients of subject 9

VARIANCE (σ^2)			
a_s	b_s	a_d	b_d
45.7	0.02	45.7	0.02

Table 14: Regression coefficients subject 10

	Measurement	a_s	b_s	a_d	b_d
	87	2.32	0.09	2.32	0.09
Subject 10 Female					
24 y/o BMI: 20.20	88	-20.51	0.68	-20.51	0.68

Table 15: Mean value of regression coefficients of subject 10

MEAN (µ)			
a_s	b_s	a_d	b_d
-9.10	0.39	-9.10	0.39

Table 16: Variance of regression coefficients of subject 10

VARIANCE (σ^2)			
a_s	b_s	a_d	b_d
260.60	0.17	260.60	0.17

Appendix E

This appendix shows the variance of the regression coefficients relative to AM and PM acquisitions in subjects 1,3 and 7.

Table 17: Variance of regression coefficients related to AM/PM recordings on subject 1

	VARIANCE (σ^2)			
	a_s	b_s	a_d	b_d
AM	42.77	0.022	269.21	0.023
PM	358.01	0.065	437.15	0.080

Table 18: Variance of regression coefficients related to AM/PM recordings on subject 3

	VARIANCE (σ^2)			
	a_s	b_s	a_d	b_d
AM	224.70	0.070	505.29	0.100
PM	957.15	0.599	427.30	0.161

Table 19: Variance of regression coefficients related to AM/PM recordings on subject 7

	VARIANCE (σ^2)			
	a_s	b_s	a_d	b_d
AM	114.87	0.014	114.87	0.014
PM	201.15	0.013	201.15	0.013

Appendix F

This appendix presents the Python code that was implemented for the GUI in this thesis project.

```
import csv
import math
import numpy as np
import operator
import pandas as pd
import pathlib
import PIL.Image
import scipy
import scipy.io
import scipy.signal
from datetime import datetime
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
    NavigationToolbar2Tk
import matplotlib.pyplot as plt
from PIL import ImageTk, Image
from pyparsing import empty
from scipy.signal import butter, find_peaks, peak_widths
from sklearn import linear_model
from sklearn.decomposition import TruncatedSVD
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
from sklearn.svm import SVR
from statistics import *
import tkinter as tk
from tkinter import *
from tkinter import filedialog, messagebox
from GUIfunctions_ACC import *


#====================================#
FUNCTIONS
#====================================#
def minutes_to_samples(user_input,fs):
    try:
        # Convert user input to minutes and seconds
```

```
            minutes, seconds = user_input.split(".")
            minutes = int(minutes)
            seconds = int(seconds)
            # Check if seconds are no more than 60
            if seconds > 59:
                root = tk.Tk()
                root.withdraw()
                messagebox.showerror("Error", "Seconds must be no more
                    than 59")
            else:
                # Convert minutes and seconds to samples
                sample = int((minutes * 60 + seconds) * fs)
        except:
            root = tk.Tk()
            root.withdraw()
            messagebox.showerror("Error", "Invalid input format. The
                correct format is mm.ss")

        return sample

def cut_signals(s1,s2,start,end,ts_1,ts_2):
    s1_c=s1[start:end]
    s2_c=s2[start:end]
    t1=np.arange(0,len(s1_c))/504.12
    t2=np.arange(0,len(s1_c))/504.12
    ts_1=ts_1[start:end]
    ts_2=ts_2[start:end]
    return s1_c,s2_c,t1,t2,ts_1,ts_2


# 1-SIGNAL PREPARING
def signal_preparing(ecg,tsecg,ppg,tsppg,acc_x,acc_y,acc_z,tsacc):
    # Creation of numpy arrays
    ecg_head=np.zeros(len(ecg))
    ts_ecg=np.zeros(len(ecg))
    ppg_head=np.zeros(len(ppg))
    ts_ppg=np.zeros(len(ppg))
    ts_acc=np.zeros(len(acc_x))

    for i in range(len(ecg)):
        ecg_head[i]=ecg[i]
        ts_ecg[i]=tsecg[i]/1000

    for i in range(len(ppg)):
        ppg_head[i]=ppg[i]
        ts_ppg[i]=tsppg[i]/1000

    fs=504.12  # Sampling frequency (Hz)
    rec_time_mins_ppg = ((len(ppg_head)-1)/fs)/60
    t_ppg = np.arange(0,len(ppg_head))/fs
```

```
rec_time_mins_ecg = ((len(ecg_head)-1)/fs)/60
t_ecg = np.arange(0,len(ecg_head))/fs
t_acc=(np.arange(0,len(acc_x))/fs)


# Cut of the first noisy samples (first 20 seconds)
cut_int=round(20/(t_ppg[-1]/len(t_ppg)))
ind=np.arange(0,cut_int)
ppg_head=np.delete(ppg_head,ind)
ts_ppg=np.delete(ts_ppg,ind)
t_ppg=np.delete(t_ppg,ind)

cut_int=round(20/(t_ecg[-1]/len(t_ecg)))
ind=np.arange(0,cut_int)
ecg_head=np.delete(ecg_head,ind)
ts_ecg=np.delete(ts_ecg,ind)
t_ecg=np.delete(t_ecg,ind)

cut_int=round(20/(t_acc[-1]/len(acc_x)))
ind=np.arange(0,cut_int)
acc_x_head=np.delete(acc_x,ind)
acc_y_head=np.delete(acc_y,ind)
acc_z_head=np.delete(acc_z,ind)
ts_acc=np.delete(tsacc,ind)/1000
t_acc=np.delete(t_acc,ind)

# ECG PPG:Signal alignment
if ts_ecg[0]<ts_ppg[0]:
    for i in range(len(ts_ecg)):
        if ts_ecg[i]>ts_ppg[0]:
            ind=np.arange(0,i-1)
            ecg_head=np.delete(ecg_head,ind)
            ts_ecg=np.delete(ts_ecg,ind)
            t_ecg=np.delete(t_ecg,ind)
            break
else:
    for i in range(len(ts_ppg)):
        if ts_ppg[i]>ts_ecg[0]:
            ind=np.arange(0,i-1)
            ppg_head=np.delete(ppg_head,ind)
            ts_ppg=np.delete(ts_ppg,ind)
            t_ppg=np.delete(t_ppg,ind)
            break

# Acc:Signal alignment
if tsacc[0]<tsppg[0]:
    for i in range(len(tsacc)):
        if tsacc[i]>tsppg[0]:
            ind=np.arange(0,i-1)
            acc_x_head=np.delete(acc_x_head,ind)
```

```
                acc_y_head=np.delete(acc_y_head,ind)
                acc_z_head=np.delete(acc_z_head,ind)
                ts_acc=np.delete(ts_acc,ind)
                t_acc=np.delete(t_acc,ind)

                break
    else:
        for i in range(len(ts_ppg)):
            if ts_ppg[i]>ts_acc[0]:
                ind=np.arange(0,i-1)
                ppg_head=np.delete(ppg_head,ind)
                ts_ppg=np.delete(ts_ppg,ind)
                t_ppg=np.delete(t_ppg,ind)
                ecg_head=np.delete(ecg_head,ind)
                ts_ecg=np.delete(ts_ecg,ind)
                t_ecg=np.delete(t_ecg,ind)
                break

    # Signals cut at the same length
    if len(ts_ecg)>len(ts_ppg):
        t=t_ppg
        ind=np.arange(0,len(ts_ppg))
        ecg_head=ecg_head[ind]
        ts_ecg=ts_ecg[ind]

    else:
        t=t_ecg
        ind=np.arange(0,len(ts_ecg))
        ppg_head=ppg_head[ind]
        ts_ppg=ts_ppg[ind]

    acc_x_head=acc_x_head[ind]
    acc_y_head=acc_y_head[ind]
    acc_z_head=acc_z_head[ind]
    ts_acc=ts_acc[ind]
    t_acc=t_acc[ind]


    # PPG ECG  Signals synchronization
    int_t=ts_ppg[0]-ts_ecg[0]
    for i in range(len(ts_ecg)):
        if abs(ts_ppg[i]-ts_ecg[i])>int_t:
            ts_ppg[i]=ts_ecg[i]-int_t

    return ( ecg_head, ppg_head, ts_ecg, ts_ppg,acc_x_head,
       acc_y_head,acc_z_head,ts_acc)


# 2-SIGNAL FILTERING
def adaptive_filter(s,acc_x,acc_y,acc_z,filter_len):
```

```
    #MIN MAX Normalization
    ns=(s-min(s))/(max(s)-min(s))
    acc_x=(acc_x-min(acc_x))/(max(acc_x)-min(acc_x))
    acc_y=(acc_y-min(acc_y))/(max(acc_y)-min(acc_y))
    acc_z=(acc_z-min(acc_z))/(max(acc_z)-min(acc_z))

    #Acc norm
    ACC=np.sqrt(acc_x**2+acc_y**2+acc_z**2)

    #Filter parameters
    #filter_len = 7
    step_size = 0.01

    # Initialize the filter coefficients to zero
    b = np.zeros(filter_len)

    # Set the number of iterations
    n_iter = len(ns)
    s_padded = np.pad(ns, (0, filter_len-1), 'constant')
    y, error = lms_adaptive_filter(s_padded, ACC, b, step_size,
       n_iter,filter_len)
    s_filt=y*(max(s)-min(s))+min(s)
    #s_filt=hl_envelopes_idx(s_filt)
    return s_filt



def lms_adaptive_filter(x, d, b, step_size, n_iter,filter_len):
    """
    Implement the LMS adaptive filter using the input signal x,
       desired signal d,
    filter coefficients b, step size, and number of iterations.
    """
    # Initialize the error signal and the output signal
    error = np.zeros(n_iter)
    y = np.zeros(n_iter)

    # Loop through the number of iterations
    for i in range(n_iter):
        # Compute the output signal
        y[i] = np.dot(b, x[i:i+filter_len])


        # Compute the error signal
        error[i] = d[i] - y[i]

        # Update the filter coefficients
        b += 2 * step_size * error[i] * x[i:i+filter_len]
```

```
    return y, error


def hl_envelopes_idx(s, dmin=1, dmax=1, split=False):


    # locals max
    lmax = (np.diff(np.sign(np.diff(s))) < 0).nonzero()[0] + 1

    if split:
        # s_mid is zero if s centered around x-axis or more
            generally mean of signal
        s_mid = np.mean(s)
        # pre-sorting of local max based on relative position with
            respect to s_mid
        lmax = lmax[s[lmax]>s_mid]

    # global min of dmin-chunks of locals min
    lmax = lmax[[i+np.argmax(s[lmax[i:i+dmax]]) for i in range(0,
        len(lmax),dmax)]]

    s_filt=np.zeros(len(s))
    n=0
    for i in range(len(s)):
        if i==lmax[n]:
            s_filt[i]=s[i]-s[lmax[n]]
            if n<len(lmax)-1:
                n=n+1
        else:
            s_filt[i]=s[i]-s[lmax[n]]

    return s_filt

def LP_butter(s,fs):

    fNy = fs/2    # Nyquist frequency (Hz)
    ft = 50        # Cut off frequency (Hz) (experimental)
    ws=0.1         # Passaband ripple (dB) (experimental)
    wp=15          # Stopband attenuation (dB) (experimental)
    fa=30          # Attenuation frequenzy (Hz) (experimental)
    n,wn=scipy.signal.buttord(ft/fNy,fa/fNy,ws,wp)
    b,a=scipy.signal.butter(n+1,wn)                  # 7-orer low-
        pass Butterworth filter
    s2_1=scipy.signal.filtfilt(b,a,s)
    s_filt = hl_envelopes_idx(s2_1)

    return s_filt


# PEAKS DETECTION FUNCTION
```

```
def peaks_detection(s_filt,ts,th):
    time=np.arange(0,len(s_filt))/504.12
    pks=find_peaks(s_filt,height=th)
    ind_pks=pks[0]
    ts_pks=np.zeros(len(ts))
    vect_pks=np.zeros(len(ts))
    vect_pks[ind_pks]=s_filt[ind_pks]
    ts_pks[ind_pks]=ts[ind_pks]

    # Local maximus deletion
    int_t=round(0.5/(time[-1]/len(time)))
    for i in range(len(vect_pks)):
        if i>len(vect_pks)-int_t:
            break
        if vect_pks[i]>0:
            for j in range(1,int_t):
                if vect_pks[i+j]>0:
                    vect_pks[i+j]=0
                    ts_pks[i+j]=0

    return vect_pks,ts_pks

#Peaks cleaning
def peaks_cleaningNOwin(th_pc,std_pc,ecg_filt,ppg_filt,ACC,S,th_ecg
   ,th_ppg,ts_ecg,ts_ppg,fs):



    vect_R, ts_R=peaks_detection(ecg_filt, ts_ecg, th_ecg)
    vect_P, ts_P=peaks_detection(ppg_filt, ts_ppg, th_ppg)

    necg=(ecg_filt-min(ecg_filt))/(max(ecg_filt)-min(ecg_filt))
    nppg=(ppg_filt-min(ppg_filt))/(max(ppg_filt)-min(ppg_filt))
    if th_pc== 1:

        mdn=statistics.median(ACC)
        dev=np.std(ACC)
        Hs=mdn+std_pc*dev
        Hi=mdn-std_pc*dev
    elif th_pc==2:
        mdn=statistics.median(S)
        dev=np.std(S)
        Hs=mdn+std_pc*dev
        Hi=mdn-std_pc*dev
        Hs=Hs[0]
        Hi=Hi[0]

    #Binary Signal V
    V=np.zeros(len(ACC))
    for i in range(len(V)):
```

```
        if ACC[i]>Hs:
            V[i]=1
        elif ACC[i]<Hi:
            V[i]=1
    # Define the threshold distance
    threshold_distance = fs*10

    # Find the peaks in the binary signal
    peaks, _ = find_peaks(V)
    # Find the widths of the peaks
    widths = peak_widths(V, peaks)[0]
    # Find the gaps between peaks that are less than the threshold
       distance
    gaps = peaks[np.where(np.diff(peaks) < threshold_distance)[0] +
        1]
    # Fill the gaps by setting the values in those indices to 1
    V[gaps] = 1
    indexV= np.nonzero(V)[0]

    #Peaks cleaning
    clean_vect_P=np.zeros(len(vect_P))
    clean_ts_P=np.zeros(len(vect_P))
    clean_vect_R=np.zeros(len(vect_R))
    clean_ts_R=np.zeros(len(vect_R))

    for i in range(len(V)):
        if V[i]==0:
            clean_vect_P[i]=vect_P[i]
            clean_ts_P[i]=ts_P[i]

            clean_vect_R[i]=vect_R[i]
            clean_ts_R[i]=ts_R[i]
        else:
            i=i+1

    return clean_vect_R, clean_ts_R, clean_vect_P, clean_ts_P

def peaks_cleaning_win(std_pc,win_pc,ecg_filt,ppg_filt,ACC_nopad,
   th_ecg,th_ppg,ts_ecg,ts_ppg,fs):
    vect_R, ts_R=peaks_detection(ecg_filt, ts_ecg, th_ecg)
    vect_P, ts_P=peaks_detection(ppg_filt, ts_ppg, th_ppg)

    #MAX MIN Normalization
    necg=(ecg_filt-min(ecg_filt))/(max(ecg_filt)-min(ecg_filt))
    nppg=(ppg_filt-min(ppg_filt))/(max(ppg_filt)-min(ppg_filt))


    #Windowing
    n_samples = int(len(necg))
```

```
samples_per_minute = int(fs * win_pc)
# Number of one-minute segments
#n_segments = round(n_samples / samples_per_minute )
n_segments = math.ceil(n_samples / samples_per_minute )
segments = []



#ACC padding
n_samples_round=samples_per_minute*n_segments

last_value=ACC_nopad[-1]
pad_size=(n_samples_round)-len(ACC_nopad)
ACC = np.pad(ACC_nopad,( 0,pad_size), 'constant',
   constant_values=(0, last_value))



# Initialize an empty list to store the threshold values
thresholds = []

# Divide the signal into one-minute segments
for i in range(n_segments):
    start = i * samples_per_minute
    end = (i + 1) * samples_per_minute
    segment = ACC[start:end]

    # Threshold 1
    mdn = statistics.median(segment)
    dev = np.std(segment)
    Hs = mdn + std_pc*dev
    Hi = mdn - std_pc*dev
    thresholds.append((Hs, Hi))

# Separate the threshold list into two separate lists
hs = [t[0] for t in thresholds]
hi = [t[1] for t in thresholds]

# Define the threshold distance
threshold_distance = fs*10

# Initialize an empty list to store the binary signals
V = []

# Find the peaks in the binary signal for each segment
for i in range(n_segments):
    segment = ACC[i*samples_per_minute:(i+1)*samples_per_minute
       ]

    Hs, Hi = thresholds[i]
```

```
        v = np.zeros(len(segment))
        for j in range(len(v)):
            if segment[j] > Hs:
                v[j] = 1
            elif segment[j] < Hi:
                v[j] = 1
        peaks, _ = find_peaks(v)
        # Find the gaps between peaks that are less than the
            threshold distance
        gaps = peaks[np.where(np.diff(peaks) < threshold_distance)
            [0] + 1]
        # Fill the gaps by setting the values in those indices to 1
        v[gaps] = 1
        V.append(v)

    # concatenate all the binary signal in one signal
    V = np.concatenate(V)
    indexV = np.nonzero(V)[0]


    #Resize V at the same lenght of original signals
    V = np.resize(V, len(vect_R))


    #Peaks cleaning
    clean_vect_P=np.zeros(len(vect_P))
    clean_ts_P=np.zeros(len(vect_P))
    clean_vect_R=np.zeros(len(vect_R))
    clean_ts_R=np.zeros(len(vect_R))


    for i in range(len(V)):
        if V[i]==0:
            clean_vect_P[i]=vect_P[i]
            clean_ts_P[i]=ts_P[i]

            clean_vect_R[i]=vect_R[i]
            clean_ts_R[i]=ts_R[i]
        else:
            i=i+1
    return clean_vect_R, clean_ts_R, clean_vect_P, clean_ts_P


# BP ESTIME FUNCTION
def BP_estime(vect_R,ts_R,vect_P,ts_P,ecg_fil,ppg_fil,ts_ecg,ts_ppg
    ,ts_omron,sbp,dbp):

    n=0
    T=0
    found=0
```

```
ptt=np.zeros(len(ppg_fil))          # PTT array
hr=np.zeros(len(ecg_fil))           # HR array
timetable=np.zeros(len(ecg_fil))    # Timestamp array
for i in range(len(vect_R)):
    if i>=T:
        if vect_R[i]>0:
            found=0
            for j in range(i+1,len(vect_R)):
                if found==1:
                    break
                if vect_R[j]>0:
                    break
                else:
                    if vect_P[j]>0:
                        ptt[n]=ts_P[j]-ts_R[i]
                        for k in range(i+1,len(vect_R)):
                            if vect_R[k]>0:
                                hr[n]=60/(ts_R[k]-ts_R[i])
                                timetable[n]=ts_R[i]
                                n=n+1
                                T=k
                                found=1
                                break

# Zero elements deletion and arrays cut at the same length
ind=np.array(np.where(ptt==0))
ptt=np.delete(ptt,ind)
ind=np.array(np.where(hr==0))
hr=np.delete(hr,ind)
ind=np.array(np.where(timetable==0))
timetable=np.delete(timetable,ind)

if len(ptt)>len(hr):
    ind=np.arange(0,len(hr))
    ptt=ptt[ind]
    timetable=timetable[ind]
else:
    ind=np.arange(0,len(ptt))
    hr=hr[ind]
    timetable=timetable[ind]

# Arrays cleaning
mean_PTT=np.mean(ptt)
dev_PTT=np.std(ptt)
mean_HR=np.mean(hr)
dev_HR=np.std(hr)

for i in range(len(timetable)):
    if ptt[i]>mean_PTT+dev_PTT or ptt[i]<mean_PTT-dev_PTT or hr
       [i]>mean_HR+dev_HR or hr[i]<mean_HR-dev_HR:
```

```
            ptt[i]=0
            hr[i]=0
            timetable[i]=0

ind=np.array(np.where(ptt==0))
ptt=np.delete(ptt,ind)
ind=np.array(np.where(hr==0))
hr=np.delete(hr,ind)
ind=np.array(np.where(timetable==0))
timetable=np.delete(timetable,ind)


'''omronfile_path = filedialog.askopenfilename()
omronfile=Path(omronfile_path)
filename=omronfile

with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    ts_omron=np.array(list(map(float,[(line[0]) for line in
       reader])))
with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    sbp=np.array(list(map(float,[(line[1]) for line in reader])
       ))
with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    dbp=np.array(list(map(float,[(line[2]) for line in reader])
       )) '''

# Omron's time values correspond to the time in which the
   device returns the
# pressure values
ts_omron=ts_omron-60*np.ones(len(ts_omron))

# Creation of the interpolating time array
n=np.array(np.where(ts_ecg==timetable[0]))
m=np.array(np.where(ts_ecg==timetable[-1]))
ind=np.arange(n,m)
t_fit=ts_ecg[ind]

# Interpolation of the Omron's data
SBP_fit=np.interp(t_fit,ts_omron,sbp)
DBP_fit=np.interp(t_fit,ts_omron,dbp)

# Interpolation of PTT and HR values
HR_fit=np.interp(t_fit,timetable,hr)
PTT_fit=np.interp(t_fit,timetable,ptt)
```

```
    #==================
    # FEATURE REDUCTION
    #==================
    row1=feat_reduction(PTT_fit,t_fit)      # PTT
    row2=feat_reduction(HR_fit,t_fit)       # HR
    row3=feat_reduction(SBP_fit,t_fit)      # SBP
    row4=feat_reduction(DBP_fit,t_fit)      # DBP

    # Arrays resampling
    row1=np.interp(timetable,t_fit,row1)
    row2=np.interp(timetable,t_fit,row2)
    row3=np.transpose(np.interp(timetable,t_fit,row3))
    row4=np.transpose(np.interp(timetable,t_fit,row4))

    sz_train=round(0.7*len(row1))
    ind_train=np.arange(0,sz_train)
    ind_test=np.arange(sz_train,len(row1))
    trainData_PTT=row1[ind_train]
    testData_PTT = row1[ind_test]
    trainData_HR=row2[ind_train]
    testData_HR = row2[ind_test]
    X_train=np.transpose(np.array([trainData_PTT,trainData_HR]))
    X_test=np.transpose(np.array([testData_PTT,testData_HR]))
    perc=round(0.2*len(ind_test))
    hr=hr[ind_test]
    hr=mean_replacement(hr, window_size=30)


    # LINEAR REGRESSION
    regr = linear_model.LinearRegression()              # Parameters
        definition
    MLR_modelfit_SBP,MLR_modelfit_DBP,MLR_SBP_pred,MLR_DBP_pred,
       MLR_mae_SBP,MLR_mae_DBP,MLR_dev_SBP,MLR_dev_DBP=
       regression_process(regr,X_train,X_test,ind_train,ind_test,
       row3,row4)
    MLR_coeff_SBP=MLR_modelfit_SBP.coef_
    MLR_coeff_DBP=MLR_modelfit_DBP.coef_
    print(MLR_coeff_SBP)
    print(MLR_coeff_DBP)


    return hr,MLR_SBP_pred,MLR_DBP_pred,row3,row4,ind_test,
       MLR_modelfit_DBP,MLR_modelfit_SBP,MLR_mae_SBP,MLR_mae_DBP,
       MLR_dev_SBP,MLR_dev_DBP

def mean_replacement(hr, window_size=30):
    num_windows = (len(hr) + window_size - 1) // window_size
    padded_hr = np.concatenate((hr, np.full(num_windows *
       window_size - len(hr), hr[-1])))
    new_hr = np.zeros(num_windows * window_size)
```

```
    for i in range(num_windows):
        start = i * window_size
        end = (i + 1) * window_size
        window = padded_hr[start:end]
        mean = np.mean(window)
        new_hr[start:end] = np.full(window_size, mean)

    new_hr= np.resize(new_hr, len(hr))

    return new_hr



# FEATURE REDUCTION FUNCTION
def feat_reduction(feat,t_fitted):

    row=np.zeros(len(t_fitted))
    T=0
    for i in range(len(t_fitted)-1):
        if i>=T:
            for j in range(i+1,len(t_fitted)):
                # Time window of 10 seconds
                if t_fitted[j]-t_fitted[i]>=10:
                    ind1=np.arange(i,j)

                    # Feature averaging
                    vect_feat=feat[ind1]
                    val_feat=np.mean(vect_feat)
                    row[ind1]=val_feat
                    T=j
                    break

    # If the last window is smaller than 10 s, the last values are
       averaged and fitted in a 10 s time window
    for i in range(len(row)):
        if row[i]==0:
            ind1=np.arange(i,len(row))

            # Feature averaging
            val_feat=np.mean(feat[ind1])
            row[ind1]=val_feat
            break

    return row

# REGRESSION PROCESS FUNCTION
def regression_process(model,matr_train,matr_test,i_train,i_test,
   sbp,dbp):
```

```
    # SBP
    modelfit_SBP = model.fit(matr_train,sbp[i_train]) # Model
        training
    sbp_pred=modelfit_SBP.predict(matr_test)              # Model
        testing
    mae_sbp=mean_absolute_error(sbp[i_test], sbp_pred)           #
        SBP MEA (mmHg)
    dev_sbp=np.std(sbp_pred)
                                                  # SBP dev std (mmHg)
    err_sbp=abs(sbp_pred-sbp[i_test])
    n=np.array(np.where(err_sbp>5))
    num_sbp=len(np.transpose(n))

    # DBP
    modelfit_DBP = model.fit(matr_train,dbp[i_train]) # Model
        training
    dbp_pred=modelfit_DBP.predict(matr_test)              # Model
        testing
    mae_dbp=mean_absolute_error(dbp[i_test], dbp_pred)           #
        DBP MEA (mmHg)
    dev_dbp=np.std(dbp_pred)
                                                  # SBP dev std (mmHg)
    err_dbp=abs(dbp_pred-dbp[i_test])
    n=np.array(np.where(err_dbp>5))
    num_dbp=len(np.transpose(n))

    return modelfit_SBP,modelfit_DBP,sbp_pred,dbp_pred,mae_sbp,
        mae_dbp,dev_sbp,dev_dbp

#====================================#
# GUI MAIN
#====================================#
def main():
    root=tk.Tk()
    gui=MyGUI(root)
    gui.root.mainloop()


    return None


#====================================#
# GUI CLASS
#====================================#
class MyGUI:
    def __init__(self,root):
        self.root=root
        self.root.title('SINTEC␣Project')
        self.root.geometry("1900x1200")
        self.root.state('zoomed')
```

```
logo=ImageTk.PhotoImage(PIL.Image.open("logo.png"))
logo_label=Label(image=logo)
logo_label.image=logo
logo_label.place(x=25,y=55)


#==============================#
# GUI attributes default values
#==============================#

#Empty signals
self.fs=504.12
self.s2=np.zeros(100000)
self.s1=np.zeros(100000)
self.t1=np.arange(0,len(self.s1))/self.fs
self.t2=np.arange(0,len(self.s2))/self.fs
self.ts_1=np.arange(0,len(self.s1))/self.fs
self.ts_2=np.arange(0,len(self.s2))/self.fs
self.s3=[]
self.s4=[]

#Raw signals before alignement/sinc/20scut
self.ecg_head=[]
self.ppg_head=[]
self.ts_ecg=[]
self.ts_ppg=[]
self.acc_x=[]
self.acc_y=[]
self.acc_z=[]
self.ts_acc=[]

#Acceerometer signals
self.ACC=[]
self.S=[]

#Filtered signals
self.ecg_fil1=[]                                    #baseline
self.ecg_fil2=[]                                    #adaptive (7)+
    baseline
self.ecg_fil2_=[]                                   #adaptive (20)+
    baseline

self.ppg_fil1=[]                                    #baseline
self.ppg_fil2=[]                                    #adaptive (7)+
    baseline
self.ppg_fil2_=[]                                   #adaptive (7)+
    baseline

self.ecg_fil=[]                                     #selected ECG
    filtered option
```

```
self.ppg_fil=[]                                    #selected PPG
    filtered option

#HR
self.hr=[]                                         #heart rate

self.ts_omron=[]                                   #omron
    timestamp
self.sbp=[]                                        #omron
    interpolated SBP
self.dbp=[]                                        #omron
    interpolated DBP
self.s3_real=[]                                    # omron test
    set SBP
self.s4_real=[]                                    # omron test
    set DBP

#BP
self.SBP=[]                                        #ML SBP
self.DBP=[]                                        #ML SBP
self.MLR_coeff_SBP=[]                              #SBP
    coefficients
self.MLR_coeff_DBP=[]                              #DBP
    coefficients
self.MLR_mae_SBP=[]                                #MAE SBP
self.MLR_mae_DBP=[]                                #MAE DBP
self.MLR_dev_SBP=[]                                #STD SBP
self.MLR_dev_DBP=[]                                #STD DBP
self.vect_R=[]                                     #Peaks R
self.ts_R=[]                                       #timestamp
    Peaks R
self.vect_P=[]                                     #Peaks P
self.ts_P=[]                                       #timestamp
    Peaks P
self.indexR=[]                                     #Index Peaks R
self.indexP=[]                                     #Index Peaks P


#Default values for Entry Box options
self.sample_start=0                                #Entry box
    start
self.sample_end=len(self.s1)                       #Entry box end
self.th_ecg=1.25                                   #Entry box ECG
    Threshold
self.th_ppg=30                                     #Entry box PPG
    Threshold

#Default values for Radiobutton options
self.choice1 = IntVar()
self.choice1.set(0)                                #Raw signals
```

```
self.sel1=0
self.choice2 = IntVar()
self.choice2.set(1)                              #ECG Baseline
    filtering option
self.sel2=0
self.choice3 = IntVar()
self.choice3.set(1)                              #PPG Baseline
    filtering option
self.sel3=0
#self.choice_pc=IntVar()
#self.choice_pc.set(2)
self.rb_var = tk.IntVar()
self.rb_var.set(1)                               #Peaks cleaning
      threshold

#Default values for label
self.initialLenght=0

#Default values for checkboxes
self.var1 = tk.IntVar()                          #Real SBP
self.var2 = tk.IntVar()                          #Real DBP
self.var3 = tk.IntVar()                          #HR
self.checkbox_var = tk.IntVar()                  #Peaks cleaning

#Menu
self.menu1_var=tk.StringVar()                    #Windowing
    duration
self.menu2_var=tk.StringVar()                    #Standard
    deviation
self.filter_length=tk.StringVar()                #Filter lenght

#==============================#
# GUI widgets
#==============================#

#Upload labels
self.filename_label_ecg = tk.Label(root, text="Upload␣a␣ECG
   ␣signal",font=("Verdana", 8, "italic"),anchor='w',bg="#
   D3D3D3", width=50, height=1)
self.filename_label_ecg.place(x=300,y=25)

self.filename_label_ppg = tk.Label(root, text="Upload␣a␣PPG
   ",font=("Verdana", 8, "italic"),anchor='w',bg="#D3D3D3"
   , width=50, height=1)
self.filename_label_ppg.place(x=300,y=55)

self.filename_label_acc = tk.Label(root, text="Upload␣
   accelerometer␣signals",font=("Verdana", 8, "italic"),
   anchor='w',bg="#D3D3D3", width=50, height=1)
self.filename_label_acc.place(x=300,y=85)
```

```
self.filename_label_omron = tk.Label(root, text="Upload␣.
    csv␣file",font=("Verdana", 8, "italic"),anchor='w',bg="
    #D3D3D3", width=50, height=1)
self.filename_label_omron.place(x=1250,y=25)

#Signals lenght label
self.fullLength_Label=Label(root,text="Signals␣are␣...␣
    minutes␣long",font=("Verdana", 10))
self.fullLength_Label.place(x=300,y=115)

#Start label + Entry box
Label(self.root,text="Plot␣signals␣from␣minute",font=("
    Verdana", 10)).place(x=300,y=145)
self.sample_start_entry=Entry(self.root,width=5)
self.sample_start_entry.insert(0,"0.00")
self.sample_start_entry.place(x=470,y=145)

#End label + Entry box
Label(self.root,text="to␣minute␣",font=("Verdana", 10)).
    place(x=510,y=145)
self.sample_end_entry=Entry(self.root,width=5)
self.sample_end_entry.insert(0,"...")
self.sample_end_entry.place(x=580,y=145)

#Threshold ECG label + Entry box
self.th_ecg_label=Label(self.root,text="ECG␣threshold␣:",
    font=("Verdana", 10))
self.th_ecg_label.place(x=730,y=150)
self.unit_ecg=Label(self.root,text="mV",font=("Verdana",
    10))
self.unit_ecg.place(x=865,y=150)
self.th_ecg_entry=Entry(self.root,width=5)
self.th_ecg_entry.insert(0,"1.25")
self.th_ecg_entry.place(x=830,y=150)

#Threshold PPG label + Entry box
self.th_ppg_label=Label(self.root,text="PPG␣threshold␣:",
    font=("Verdana", 10))
self.th_ppg_label.place(x=915,y=150)
self.unit_ppg=Label(self.root,text="mV",font=("Verdana",
    10))
self.unit_ppg.place(x=1050,y=150)
self.th_ppg_entry=Entry(self.root,width=5)
self.th_ppg_entry.insert(0,"30")
self.th_ppg_entry.place(x=1015,y=150)

#BP statistics label
self.mae1=Label(root,text='MAE␣SBP␣=␣...',font=("Verdana",
    10))
```

```
self.mae1.place(x=1250,y=85)
self.mae2=Label(root,text='MAE␣DBP␣=␣...',font=("Verdana",
    10))
self.mae2.place(x=1250,y=115)
self.dev1=Label(root,text='STD␣SBP␣=␣...',font=("Verdana",
    10) )
self.dev1.place(x=1400,y=85)
self.dev2=Label(root,text='STD␣DBP␣=␣...',font=("Verdana",
    10))
self.dev2.place(x=1400,y=115)

#Radio Buttons
#Possibility to choose between raw, filtered, peaks
    detection
self.rb0=tk.Radiobutton(self.root,text='Raw␣signals',font=(
    "Verdana", 10),variable=self.choice1,value=0,command=
    self.update_values,state=tk.DISABLED)
                        #raw signals
self.rb0.place(x=680,y=20)
self.rb0_1=tk.Radiobutton(self.root,text="Filtering␣options
    ␣:",font=("Verdana", 10),variable=self.choice1,value=1,
    command=self.update_values)           #filtered signals
self.rb0_1.place(x=680,y=45)
self.rb0_2=Radiobutton(self.root,text="R␣&␣P␣peaks",font=("
    Verdana", 10),variable=self.choice1,value=2,command=
    self.update_values)                          #peaks
    detection
self.rb0_2.place(x=680,y=125)

#ECG filtering options
#Possibility to choose between baseline or adaptive+
    baseline filter
self.rb1_1=tk.Radiobutton(self.root,text='ECG␣Baseline␣
    removal',font=("Verdana", 10),variable=self.choice2,
    value=1,command=self.update_values,state=tk.DISABLED)
self.rb1_1.place(x=730,y=75)
self.rb1_2=tk.Radiobutton(self.root,text='ECG␣Adaptive␣
    filter',font=("Verdana", 10),variable=self.choice2,
    value=2,command=self.update_values,state=tk.DISABLED)
self.rb1_2.place(x=900,y=75)

#PPG filtering options
#Possibility to choose between baseline+LP or adaptive+
    baseline filter
self.rb2_1=tk.Radiobutton(self.root,text='PPG␣Baseline␣
    removal',font=("Verdana", 10),variable=self.choice3,
    value=1,command=self.update_values,state=tk.DISABLED)
self.rb2_1.place(x=730,y=100)
```

```
self.rb2_2=tk.Radiobutton(self.root,text='PPG␣Adaptive␣
    filter',font=("Verdana", 10),variable=self.choice3,
    value=2,command=self.update_values,state=tk.DISABLED)
self.rb2_2.place(x=900,y=100)


#Peaks cleaning
#Enabling Peaks cleaning
self.checkbox_pc = tk.Checkbutton(root, text="Enable␣Peaks␣
    cleaning",font=("Verdana", 10), variable=self.
    checkbox_var)
self.checkbox_pc.place(x=1650,y=25)
#Possibility to choose the threshold
self.rb_th1 = tk.Radiobutton(root, text="Th1",font=("
    Verdana", 10), variable=self.rb_var, value=1)
self.rb_th1.place(x=1650,y=60)
self.rb_th2 = tk.Radiobutton(root, text="Th2",font=("
    Verdana", 10), variable=self.rb_var, value=2)
self.rb_th2.place(x=1750,y=60)

#Menu Options for Peaks cleaning
# Windowing
self.menu1 = tk.OptionMenu(root, self.menu1_var, "full", "1
    ␣min", "0.5␣min")
self.min_values = {"full": 1, "1␣min": 60, "0.5␣min": 30}
self.menu1.config(font=("Verdana", 10))
self.menu1_var.set("full")
self.menu1.place(x=1650,y=90)
menu1 = tk.Menu(self.menu1, tearoff=0, font=("Verdana", 10)
    )
for option in self.min_values:
    menu1.add_command(label=option, command=lambda value=
        option: self.menu1_var.set(value))
self.menu1['menu'] = menu1

#Standard deviation
self.menu2 = tk.OptionMenu(root,self.menu2_var,"1.5␣std", "
    2␣std", "3␣std")
self.menu2.config(font=("Verdana", 10))
self.std_values = {"1.5␣std": 1.5, "2␣std": 2, "3␣std": 3}
self.menu2_var.set("2␣std")
self.menu2.place(x=1750,y=90)
menu = tk.Menu(self.menu2, tearoff=0, font=("Verdana", 10))
for option in self.std_values:
    menu.add_command(label=option, command=lambda value=
        option: self.menu2_var.set(value))
self.menu2['menu'] = menu

#Option menu adaptive filter
```

```
self.labelFilter=Label(root,text="Select␣Adaptive␣\n␣filter
   ␣length:",font=("Verdana",10),justify= RIGHT,state=
   DISABLED)
self.labelFilter.place(x=1100,y=45)
self.filter_length_options = {"7": 7, "20": 20}
self.menu_box = tk.OptionMenu(root, self.filter_length, "7"
   ,"20")
self.filter_length.set("7")
self.menu_box.place(x=1155, y=80)
self.menu_box.config(font=("Verdana", 10))
menu3 = tk.Menu(self.menu_box, tearoff=0, font=("Verdana",
   10))
for option in self.filter_length_options:
    menu3.add_command(label=option, command=lambda value=
       option: self.filter_length.set(value))
self.menu_box['menu'] = menu3
self.checkbox_var.trace("w", self.checkbox_trace)
self.rb_var.trace("w", self.rb_trace)

#Checkboxes
cb1=Checkbutton(self.root,text="Real␣SBP",font=("Verdana",
   10),variable=self.var1, onvalue=1,offvalue=0, command=
   self.plot_graphs_bp).place(x=1250,y=145)
cb2=Checkbutton(self.root,text="Real␣DBP",font=("Verdana",
   10),variable=self.var2, onvalue=1,offvalue=0, command=
   self.plot_graphs_bp).place(x=1350,y=145)
cb3=Checkbutton(self.root,text="HR",font=("Verdana", 10),
   variable=self.var3, onvalue=1,offvalue=0, command=self.
   plot_graphs_bp).place(x=1450,y=145)

#Configuration state
self.rb0_1.config(state="disabled")
self.rb0_2.config(state="disabled")
self.rb_th1.config(state="disabled")
self.rb_th2.config(state="disabled")
self.th_ecg_label.config(state="disabled")
self.unit_ecg.config(state="disabled")
self.th_ppg_label.config(state="disabled")
self.unit_ppg.config(state="disabled")
self.th_ecg_entry.config(state="disabled")
self.th_ppg_entry.config(state="disabled")
self.menu1.config(state="disabled")
self.menu2.config(state="disabled")
self.menu_box.config(state="disabled")


#Upload Button
buttonUPLOAD=Button(self.root,text="UPLOAD",font=("Verdana"
   , 10,"bold"),bg='#0F4170',fg='#31BBF5',command=self.
   upload).place(x=25,y=165)
```

```
        self.root.bind("<Return>", self.upload)
        self.plot_graphs()
        pass

        #Plot Button
        buttonPLOT=Button(self.root,text="␣PLOT␣",font=("Verdana",
            10,"bold"),bg='#0F4170',fg='#31BBF5',command=self.
            update_values).place(x=1155,y=165)
        self.root.bind("<Return>", self.update_values)
        self.plot_graphs()
        pass

        #Save button
        buttonSAVE=Button(self.root,text="SAVE",font=("Verdana",
            10,"bold"),bg='#0F4170',fg='#31BBF5',command=self.
            save_data).place(x=100,y=165)

        #Clear Button
        buttonCLEAR=Button(self.root,text="CLEAR",font=("Verdana",
            10,"bold"),bg='#0F4170',fg='#31BBF5',command=self.
            clear_graphs).place(x=155,y=165)
        self.root.bind("<Return>", self.clear_graphs)
        self.plot_graphs()
        self.plot_graphs_bp()
        pass

        #Button ACC and SVD visualization
        self.button = tk.Button(self.root, text="View␣ACC␣and␣SVD",
            font=("Verdana", 10,"bold"),bg='#0F4170',fg='#31BBF5',
            command=self.open_second_window)
        self.button.place(x=1650,y=165)

        #Evaluate Button
        buttonEVALUATE=Button(self.root,text="PREDICT",font=("
            Verdana", 10,"bold"),bg='#0F4170',fg='#31BBF5',command=
            self.evaluate).place(x=1800,y=165)
        self.root.bind("<Return>", self.evaluate)
        self.plot_graphs_bp()
        pass

#==============================#
# GUI methods
#==============================#
def checkbox_trace(self, *args):
    if self.checkbox_var.get() == 1:
        self.rb_th1.config(state="normal")
        self.rb_th2.config(state="normal")
        self.rb_trace()
    else:
        self.rb_th1.config(state="disabled")
```

```
            self.rb_th2.config(state="disabled")
            self.menu1.config(state="disabled")
            self.menu2.config(state="disabled")

    def rb_trace(self, *args):
        if self.rb_var.get() == 1:
            self.menu1.config(state="normal")
            self.menu2.config(state="normal")
        elif self.rb_var.get() == 2:
            self.menu1.config(state="disabled")
            self.menu2.config(state="normal")

    def open_second_window(self):

        self.second_window = tk.Toplevel(self.root)
        self.second_window.title("Accelerometer signal")
        self.fig5 = plt.figure(figsize=(12, 7), dpi=100)

        #Plot ACC
        self.ax1 = self.fig5.add_subplot(211)
        self.ax2 = self.fig5.add_subplot(212)
        self.ax1.set_xlabel('time (min)')
        self.ax1.set_ylabel('Amplitude (mV)')
        self.ax1.set_title('ACC')
        self.ax1.plot((np.arange(0,len(self.ACC))/self.fs)/60,self.
            ACC)

        #Plot S
        self.ax2.set_xlabel('time (min)')
        self.ax2.set_ylabel('Amplitude (mV)')
        self.ax2.set_title('S')
        self.ax2.plot((np.arange(0,len(self.S))/self.fs)/60,self.S)

        canvasbar = FigureCanvasTkAgg(self.fig5, self.second_window
            )
        canvasbar.draw()
        canvasbar.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH,
            expand=True)
        toolbar5 = NavigationToolbar2Tk(canvasbar, self.
            second_window).place(x=25,y=5)



    def clear_graphs(self,*arg):
        #Empty signals
        self.s2=np.zeros(1000)
        self.s1=np.zeros(1000)
        self.t1=np.arange(0,len(self.s1))/self.fs
        self.t2=np.arange(0,len(self.s2))/self.fs
```

```
self.s3=[]
self.s4=[]
self.s3_real=[]
self.s4_real=[]


#Raw signals before alignement/sinc/20scut
self.ecg_head=[]
self.ppg_head=[]
self.ts_ecg=[]
self.ts_ppg=[]
self.acc_x=[]
self.acc_y=[]
self.acc_z=[]
self.ts_acc=[]

#ACC
self.ACC=[]
self.S=[]

#Filtered signals
self.ecg_fil1=[]
self.ecg_fil2=[]

self.ppg_fil1=[]
self.ppg_fil2=[]

self.ecg_fil=[]
self.ppg_fil=[]

#HR
self.hr=[]                                          #heart rate

self.ts_omron=[]                                    #omron
    timestamp
self.sbp=[]                                         #omron
    interpolated SBP
self.dbp=[]                                         #omron
    interpolated DBP
self.s3_real=[]                                     # omron test
    set SBP
self.s4_real=[]                                     # omron test
    set DBP

#BP
self.SBP=[]                                         #ML SBP
self.DBP=[]                                         #ML SBP
self.MLR_coeff_SBP=[]                               #SBP
    coefficients
```

```
self.MLR_coeff_DBP=[]                          #DBP
    coefficients
self.MLR_mae_SBP=[]                            #MAE SBP
self.MLR_mae_DBP=[]                            #MAE DBP
self.MLR_dev_SBP=[]                            #STD SBP
self.MLR_dev_DBP=[]                            #STD DBP
self.vect_R=[]                                 #Peaks R
self.ts_R=[]                                   #timestamp
    Peaks R
self.vect_P=[]                                 #Peaks P
self.ts_P=[]                                   #timestamp
    Peaks P
self.indexR=[]                                 #Index Peaks R
self.indexP=[]                                 #Index Peaks P

self.initialLenght=0

#Clearing labels
self.filename_label_ecg.config(text="Upload␣a␣ECG␣signal")
self.filename_label_ppg.config(text="Upload␣a␣PPG␣signal")
self.filename_label_acc.config(text="Upload␣a␣accelerometer
    ␣signal")
self.filename_label_omron.config(text="Upload␣a␣.csv␣file")
self.fullLength_Label.config(text="Signals␣are␣...␣minutes␣
    long")
self.mae1.config(text="MAE␣SBP␣=␣...")
self.mae2.config(text="MAE␣DBP␣=␣...")
self.dev1.config(text="STD␣SBP␣=␣...")
self.dev2.config(text="STD␣SBP␣=␣...")

#Default Radio button selection
self.rb0.select()
self.rb1_1.select()
self.rb2_1.select()

# Widgets default values
self.th_ecg_entry.delete(0, END)
self.th_ecg_entry.insert(0, "1.25")
self.th_ppg_entry.delete(0, END)
self.th_ppg_entry.insert(0, "30")
self.sample_start_entry.delete(0,END)
self.sample_start_entry.insert(0,"0.00")
self.sample_end_entry.delete(0,END)
self.sample_end_entry.insert(0,"...")
self.var1.set(0)
self.var2.set(0)
self.var3.set(0)
self.checkbox_var.set(0)
self.rb_var.set(1)
self.menu1_var.set("full")
```

```
        self.menu2_var.set("2␣std")
        self.filter_length.set("7")


        #Configuration state
        self.rb0.config(state="disabled")
        self.rb0_1.config(state="disabled")
        self.rb0_2.config(state="disabled")
        self.rb_th1.config(state="disabled")
        self.rb_th2.config(state="disabled")
        self.th_ecg_label.config(state="disabled")
        self.unit_ecg.config(state="disabled")
        self.th_ppg_label.config(state="disabled")
        self.unit_ppg.config(state="disabled")
        self.th_ecg_entry.config(state='disabled')
        self.th_ppg_entry.config(state='disabled')
        self.menu1.config(state="disabled")
        self.menu2.config(state="disabled")
        self.menu_box.config(state='disabled')
        self.rb1_1.config(state="disabled")
        self.rb1_2.config(state="disabled")
        self.rb2_1.config(state="disabled")
        self.rb2_2.config(state="disabled")
        self.menu_box.config(state="disabled")

        self.labelFilter.config(state="disabled")

        self.root.bind("<Return>", self.clear_graphs)
        self.plot_graphs()
        self.plot_graphs_bp()
        pass



    def upload(self,*arg):
        #ECG
        file_path_ecg = filedialog.askopenfilename(title='Select␣
            the␣ECG␣signal')
        p_ecg=Path(file_path_ecg)
        filename_ecg=p_ecg.name
        self.filename_label_ecg.config(text=filename_ecg)
        ecg_mat=scipy.io.loadmat(filename_ecg)
        ecg=ecg_mat['Shimmer_6C0E_ECG_LA_RA_24BIT_CAL']
        tsecg=ecg_mat['Shimmer_6C0E_Timestamp_Unix_CAL']


        #PPG
        file_path_ppg = filedialog.askopenfilename(title='Select␣
            the␣PPG␣signal')
        p_ppg=Path(file_path_ppg)
```

```
filename_ppg=p_ppg.name
self.filename_label_ppg.config(text=filename_ppg)
ppg_mat=scipy.io.loadmat(filename_ppg)
ppg=ppg_mat['Shimmer_9404_PPG_A13_CAL']
tsppg=ppg_mat['Shimmer_9404_Timestamp_Unix_CAL']

#Accelerometer
file_path_acc = filedialog.askopenfilename(title='Select␣
   the␣accelerometer␣signal')
p_acc=Path(file_path_acc)
filename_acc=p_acc.name
self.filename_label_acc.config(text=filename_acc)
acc_mat=scipy.io.loadmat(filename_acc)
acc_x=acc_mat['Shimmer_6D4F_Accel_LN_X_CAL']
acc_y=acc_mat['Shimmer_6D4F_Accel_LN_Y_CAL']
acc_z=acc_mat['Shimmer_6D4F_Accel_LN_Z_CAL']
tsacc=acc_mat['Shimmer_6D4F_Timestamp_Unix_CAL']

#omron
omronfile_path = filedialog.askopenfilename(title='Select␣
   the␣coorect␣.csv␣file')
omronfile=Path(omronfile_path)
filename=omronfile
filename_omron=filename.name
self.filename_label_omron.config(text=filename_omron)

with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    ts_omron=np.array(list(map(float,[(line[0]) for line in
        reader])))
with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    sbp=np.array(list(map(float,[(line[1]) for line in
       reader])))
with open(filename) as filecsv:
    reader=csv.reader(filecsv,delimiter=";")
    dbp=np.array(list(map(float,[(line[2]) for line in
       reader])))

#====================#
# SIGNAL PREPARING
#====================#
ecg_head, ppg_head, ts_ecg, ts_ppg,acc_x_head,acc_y_head,
   acc_z_head,ts_acc =signal_preparing(ecg,tsecg,ppg,tsppg
   ,acc_x,acc_y,acc_z,tsacc)

#Raw signals
self.ecg_head=ecg_head
self.ppg_head=ppg_head
self.ts_ecg=ts_ecg
```

```
self.ts_ppg=ts_ppg
self.acc_x=acc_x_head
self.acc_y=acc_y_head
self.acc_z=acc_z_head
self.ts_acc=ts_acc

#====================#
# SIGNAL FILTERING
#====================#
self.ecg_fil1=hl_envelopes_idx(self.ecg_head) #baseline
    removal

ecgfil2=adaptive_filter(self.ecg_head,self.acc_x,self.acc_y
    ,self.acc_z,7)#adaptive
self.ecg_fil2=hl_envelopes_idx(ecgfil2)

ecgfil2_=adaptive_filter(self.ecg_head,self.acc_x,self.
    acc_y,self.acc_z,20)#adaptive
self.ecg_fil2_=hl_envelopes_idx(ecgfil2_)

self.ppg_fil1=LP_butter(self.ppg_head,self.fs) #LP +
    baseline

ppgfil2=adaptive_filter(self.ppg_head,self.acc_x,self.acc_y
    ,self.acc_z,7)
self.ppg_fil2=hl_envelopes_idx(ppgfil2) #adaptive

ppgfil2_=adaptive_filter(self.ppg_head,self.acc_x,self.
    acc_y,self.acc_z,20)
self.ppg_fil2_=hl_envelopes_idx(ppgfil2_) #adaptive



#====================#
# ACC
#====================#
#MIN MAX normalization
acc_x=(acc_x_head-min(acc_x_head))/(max(acc_x_head)-min(
    acc_x_head))
acc_y=(acc_y_head-min(acc_y_head))/(max(acc_y_head)-min(
    acc_y_head))
acc_z=(acc_z_head-min(acc_z_head))/(max(acc_z_head)-min(
    acc_z_head))

#ACC norm
self.ACC=np.sqrt(acc_x**2+acc_y**2+acc_z**2)

#SVD
acc_matrix=np.asarray([acc_x,acc_y,acc_z])
acc_matrix=np.transpose(acc_matrix)
```

```
        svd =  TruncatedSVD(n_components = 1)
        self.S = svd.fit_transform(acc_matrix)

        #====================#
        # OMRON
        #====================#
        self.ts_omron=ts_omron
        self.sbp=sbp
        self.dbp=dbp

        #Plot update (raw signals)
        self.s1=ecg_head
        self.s2=ppg_head
        self.t1=np.arange(0,len(self.s1))/self.fs
        self.t2=np.arange(0,len(self.s2))/self.fs
        self.ts_1=ts_ecg
        self.ts_2=ts_ppg
        self.sample_end=len(self.s1)
        duration_in_minutes, seconds = divmod(self.sample_end
            /504.12, 60)
        self.initialLenght= "{:.0f}.{:02d}".format(
            duration_in_minutes, int(seconds))

        #Frame/entry box update
        fullLenght = "Signals␣are␣" +str(self.initialLenght) + "␣
            minutes␣long"
        self.fullLength_Label.config(text=fullLenght)
        self.sample_end_entry.delete(0, tk.END)
        t=str(self.initialLenght)
        self.sample_end_entry.insert(0,t)

        #Radio button
        self.sel1=0
        self.rb0.config(state="normal")
        self.rb0_1.config(state='normal')
        self.rb0_2.config(state='normal')
        self.root.bind("<Return>", self.upload)
        self.update_values
        self.plot_graphs()
        pass

    def update_values(self,*arg):
        #Update  entry box values
        a=str(self.sample_start_entry.get())
        b=str(self.sample_end_entry.get())
        self.sample_start=minutes_to_samples(a,self.fs)
        self.sample_end=minutes_to_samples(b,self.fs)
        self.th_ecg=float(self.th_ecg_entry.get())
        self.th_ppg=float(self.th_ppg_entry.get())
```

```
#Update radio button selection
self.sel1=int(self.choice1.get())
self.sel2=int(self.choice2.get())
self.sel3=int(self.choice3.get())



if self.sel1==0: #Raw signal plot
    self.s1=self.ecg_head
    self.s2=self.ppg_head
    self.ts_1=self.ts_ecg
    self.ts_2=self.ts_ppg
    self.t1=np.arange(0,len(self.s1))/self.fs
    self.t2=np.arange(0,len(self.s2))/self.fs


    self.rb1_1.config(state=tk.DISABLED)
    self.rb1_2.config(state=tk.DISABLED)
    #self.rb1_3.config(state=tk.DISABLED)
    self.rb2_1.config(state=tk.DISABLED)
    self.rb2_2.config(state=tk.DISABLED)
    #self.rb2_3.config(state=tk.DISABLED)
    self.menu_box.config(state='disabled')
    self.labelFilter.config(state='disabled')
    self.th_ecg_label.config(state="disabled")
    self.unit_ecg.config(state="disabled")
    self.th_ppg_label.config(state="disabled")
    self.unit_ppg.config(state="disabled")
    self.th_ecg_entry.config(state='disabled')
    self.th_ppg_entry.config(state='disabled')

    #Signals cut
    self.s1,self.s2,self.t1,self.t2,self.ts_1,self.ts_2 =
       cut_signals(self.s1,self.s2,self.sample_start,self.
       sample_end,self.ts_1,self.ts_2)

elif self.sel1==1: #Filtered signals


    self.rb1_1.config(state=tk.NORMAL)
    self.rb1_2.config(state=tk.NORMAL)
    #self.rb1_3.config(state=tk.NORMAL)
    self.rb2_1.config(state=tk.NORMAL)
    self.rb2_2.config(state=tk.NORMAL)
    #self.rb2_3.config(state=tk.NORMAL)
    self.menu_box.config(state='normal')
    self.labelFilter.config(state='normal')
    self.unit_ecg.config(state="disabled")
    self.th_ppg_label.config(state="disabled")
    self.unit_ppg.config(state="disabled")
```

```
        self.th_ecg_entry.config(state='disabled')
        self.th_ppg_entry.config(state='disabled')


        #ECG filtering options:
        if self.choice2.get()==1: #baseline

            self.s1=self.ecg_fil1
            self.ts_1=self.ts_ecg
            self.t1=np.arange(0,len(self.s1))/self.fs


        elif self.choice2.get()==2: #adaptive
            if self.filter_length.get() == "7":
                self.s1 = self.ecg_fil2
            elif self.filter_length.get() == "20":
                self.s1 = self.ecg_fil2_


            self.ts_1=self.ts_ecg
            self.t1=np.arange(0,len(self.s1))/self.fs




        #PPG filtering options:
        if self.choice3.get()==1: #baseline +LP

            self.s2=self.ppg_fil1
            self.ts_2=self.ts_ppg
            self.t2=np.arange(0,len(self.s2))/self.fs


        elif self.choice3.get() == 2: #adaptive
            if self.filter_length.get() == "7":
                self.s2 = self.ppg_fil2
            elif self.filter_length.get() == "20":
                self.s2 = self.ppg_fil2_

            self.t2=np.arange(0,len(self.s2))/self.fs


        #Signals cut
        self.s1,self.s2,self.t1,self.t2,self.ts_1,self.ts_2 =
           cut_signals(self.s1,self.s2,self.sample_start,self.
           sample_end,self.ts_1,self.ts_2)

    elif self.sel1==2: #R & S peaks detection
        self.th_ecg_entry.config(state='normal')
        self.th_ppg_entry.config(state='normal')
```

```
        self.th_ecg_label.config(state="normal")
        self.unit_ecg.config(state="normal")
        self.th_ppg_label.config(state="normal")
        self.unit_ppg.config(state="normal")
        if self.choice2.get()==1: #baseline
            self.s1=self.ecg_fil1
        elif self.choice2.get()==2: #adaptive
            if self.filter_length.get() == "7":
                self.s1 = self.ecg_fil2
            elif self.filter_length.get() == "20":
                self.s1 = self.ecg_fil2_
        if self.choice3.get()==1: #baseline +LP
            self.s2=self.ppg_fil1
        elif self.choice3.get() == 2: #adaptive
            if self.filter_length.get() == "7":
                self.s2 = self.ppg_fil2

            elif self.filter_length.get() == "20":
                self.s2 = self.ppg_fil2_


        self.ts_1=self.ts_ecg
        self.ts_2=self.ts_ppg
        self.t1=np.arange(0,len(self.s1))/self.fs
        self.t2=np.arange(0,len(self.s2))/self.fs

        #Signals cut
        self.s1,self.s2,self.t1,self.t2,self.ts_1,self.ts_2 =
            cut_signals(self.s1,self.s2,self.sample_start,self.
            sample_end,self.ts_1,self.ts_2)

        #Peaks detection
        #ECG peaks
        self.vect_R, self.ts_R=peaks_detection(self.s1, self.
            ts_1, self.th_ecg)
        self.indexR = np.nonzero(self.vect_R)[0]

        # PPG peaks
        self.vect_P, self.ts_P=peaks_detection(self.s2, self.
            ts_2, self.th_ppg)
        self.indexP = np.nonzero(self.vect_P)[0]

    else :
       tk.messagebox.showerror(title='Error␣', message='Error␣
          in␣signal␣processing')



    self.root.bind("<Return>", self.update_values)
    self.plot_graphs()
```

```
        pass


    def evaluate(self):
        #Entry box update
        self.th_ecg=float(self.th_ecg_entry.get())
        self.th_ppg=float(self.th_ppg_entry.get())

        #Selection of the correct filtered signals
        if self.choice2.get()==1: #baseline
                self.ecg_fil=self.ecg_fil1

        elif self.choice2.get()==2: #adaptive
            if self.filter_length.get() == "7":
                self.ecg_fil = self.ecg_fil2

            elif self.filter_length.get() == "20":
                self.ecg_fil = self.ecg_fil2_


        if self.choice3.get()==1: #baseline +LP
                self.ppg_fil=self.ppg_fil1

        elif self.choice3.get() == 2: #adaptive
            if self.filter_length.get() == "7":
                self.ppg_fil = self.ppg_fil2

            elif self.filter_length.get() == "20":
                self.ppg_fil = self.ppg_fil2_


        #====================#
        # PEAKS DETECTION
        #====================#

        if self.checkbox_var.get()==0:
            self.vect_R, self.ts_R=peaks_detection(self.ecg_fil,
                self.ts_ecg, self.th_ecg)
            self.vect_P, self.ts_P=peaks_detection(self.ppg_fil,
                self.ts_ppg, self.th_ppg)
        elif self.checkbox_var.get()==1:
            if self.rb_var.get()==1: #Th1
                th_pc=1
                if self.menu1_var.get()=="full":
                    if self.menu2_var.get() != "":
                        std_pc = self.std_values[self.menu2_var.get
                            ()]
                    else:
                        std_pc = self.std_values[list(self.
                            std_values.keys())[1]]
```

```
                self.vect_R, self.ts_R, self.vect_P, self.ts_P
                    = peaks_cleaningNOwin(th_pc,std_pc,self.
                    ecg_fil,self.ppg_fil,self.ACC,self.S,self.
                    th_ecg,self.th_ppg,self.ts_ecg,self.ts_ppg,
                    self.fs)
            elif self.menu1_var.get()=="1 min" or self.
                menu1_var.get()=="0.5 min" :
                if self.menu1_var.get() != "":
                    win_pc = self.min_values[self.menu1_var.get
                        ()]

                else:
                    win_pc = self.min_values[list(self.
                        min_values.keys())[1]]

                if self.menu2_var.get() != "":
                    std_pc = self.std_values[self.menu2_var.get
                        ()]
                else:
                    std_pc = self.std_values[list(self.
                        std_values.keys())[1]]

                self.vect_R, self.ts_R, self.vect_P, self.ts_P
                    = peaks_cleaning_win(std_pc,win_pc,self.
                    ecg_fil,self.ppg_fil,self.ACC,self.th_ecg,
                    self.th_ppg,self.ts_ecg,self.ts_ppg,self.fs
                    )
        elif self.rb_var.get()==2: #Th2
            th_pc=2
            std_pc=self.std_values[self.menu2_var.get()]
            self.vect_R, self.ts_R, self.vect_P, self.ts_P =
                peaks_cleaningNOwin(th_pc,std_pc,self.ecg_fil,
                self.ppg_fil,self.ACC,self.S,self.th_ecg,self.
                th_ppg,self.ts_ecg,self.ts_ppg,self.fs)


    hr,MLR_SBP_pred,MLR_DBP_pred,row3,row4,ind_test,
        MLR_modelfit_DBP,MLR_modelfit_SBP,MLR_mae_SBP,
        MLR_mae_DBP,MLR_dev_SBP,MLR_dev_DBP=BP_estime(self.
        vect_R, self.ts_R, self.vect_P, self.ts_P,self.ecg_fil,
        self.ppg_fil,self.ts_ecg,self.ts_ppg,self.ts_omron,self
        .sbp,self.dbp)

    self.hr=hr
    self.SBP=MLR_SBP_pred
    self.DBP=MLR_DBP_pred
    self.MLR_coeff_SBP=MLR_modelfit_SBP.coef_
    self.MLR_coeff_DBP=MLR_modelfit_DBP.coef_
```

```
        self.MLR_mae_SBP=MLR_mae_SBP
        self.MLR_mae_DBP=MLR_mae_DBP
        self.MLR_dev_SBP=MLR_dev_SBP
        self.MLR_dev_DBP=MLR_dev_DBP

        self.s3=MLR_SBP_pred
        self.t3=np.arange(0,len(ind_test))
        self.s3_real=row3[ind_test]

        self.s4=MLR_DBP_pred
        self.t4=np.arange(0,len(ind_test))
        self.s4_real=row4[ind_test]

        #BP statistics label
        self.mae1.config(text="MAE␣SBP␣=␣" + str(round(self.
            MLR_mae_SBP, 2)))
        self.mae2.config(text="MAE␣DBP␣=␣" + str(round(self.
            MLR_mae_DBP, 2)))
        self.dev1.config(text="STD␣SBP␣=␣" + str(round(self.
            MLR_dev_SBP, 2)))
        self.dev2.config(text="STD␣DBP␣=␣" + str(round(self.
            MLR_dev_DBP, 2)))


        self.root.bind("<Return>", self.evaluate)
        self.plot_graphs_bp()
        pass


    def save_data(self,*arg):
        data_hr=np.asarray(self.hr)
        data_SBP=self.SBP
        data_DBP=self.DBP
        data_coeff_SBP=self.MLR_coeff_SBP
        data_coeff_DBP=self.MLR_coeff_DBP
        data_mae1=np.asarray(self.MLR_mae_SBP)
        data_mae1 = np.reshape(data_mae1,(1, 1))
        data_mae2=np.asarray(self.MLR_mae_DBP)
        data_mae2 = np.reshape(data_mae2,(1, 1))
        data_dev1=np.asarray(self.MLR_dev_SBP)
        data_dev1 = np.reshape(data_dev1,(1, 1))
        data_dev2=np.asarray(self.MLR_dev_DBP)
        data_dev2 = np.reshape(data_dev2,(1, 1))
        data = [[data_hr], [data_SBP], [data_DBP], [data_coeff_SBP
            ], [data_coeff_DBP],[data_mae1],[data_dev1],[data_mae2
            ],[data_dev2]]
```

```
        df=dict(hr=data_hr,SBP=data_SBP,DBP=data_DBP, coeffSBP=
            data_coeff_SBP,coeffDBP=data_coeff_DBP,MAE_SBP=
            data_mae1,STD_SBP=data_dev1,MAE_DBP=data_mae2,STD_DBP=
            data_dev2 )

        frame = pd.DataFrame.from_dict(df, orient='index')
        frame = frame.transpose()

        data = [('All␣types(*.*)', '*.*'),("csv␣file(*.csv)","*.csv
            ")]
        file = filedialog.asksaveasfilename(filetypes = data,
            defaultextension = data)
        with open(file,"w") as f:
            f.write(str(frame))


    def plot_graphs(self,*arg):

        #Figure Size
        dx=12 #figure width
        dy=3.8  #figure height

        if self.sel1==0:

            fig1=plt.figure(figsize=(dx,dy),dpi=100)
            plt.xlabel('time␣(min)')
            plt.ylabel('Amplitude␣(mV)')
            plt.title('ECG')
            plt.plot(self.t1/60,self.s1)

            fig2=plt.figure(figsize=(dx,dy),dpi=100)
            plt.xlabel('time␣(min)')
            plt.ylabel('Amplitude␣(mV)')
            plt.title('PPG')
            plt.plot(self.t2/60,self.s2)

        elif self.sel1==1:
            fig1=plt.figure(figsize=(dx,dy),dpi=100)
            plt.xlabel('time␣(min)')
            plt.ylabel('Amplitude␣(mV)')
            plt.title('ECG')
            plt.plot(self.t1/60,self.s1)


            fig2=plt.figure(figsize=(dx,dy),dpi=100)
            plt.xlabel('time␣(min)')
            plt.ylabel('Amplitude␣(mV)')
            plt.title('PPG')
            plt.plot(self.t2/60,self.s2)
```

```
    elif self.sel1==2:
        fig1=plt.figure(figsize=(dx,dy),dpi=100)
        plt.xlabel('time (min)')
        plt.ylabel('Amplitude (mV)')
        plt.title('ECG')
        plt.plot(self.t1/60,self.s1)
        plt.plot((self.t1[self.indexR])/60,self.s1[self.indexR
            ],'ro')

        fig2=plt.figure(figsize=(dx,dy),dpi=100)
        plt.xlabel('time (min)')
        plt.ylabel('Amplitude (mV)')
        plt.title('PPG')
        plt.plot(self.t2/60,self.s2)
        plt.plot((self.t2[self.indexP])/60,self.s2[self.indexP
            ],'ro')
    else:
        tk.messagebox.showerror(title='Error ', message='Error
            in signal processing')

    canvasbar=FigureCanvasTkAgg(fig1,self.root)
    canvasbar.draw()
    canvasbar.get_tk_widget().place(x=25,y=200)
    toolbar1 = NavigationToolbar2Tk(canvasbar, self.root).place
       (x=25,y=200)


    canvasbar=FigureCanvasTkAgg(fig2,self.root)
    canvasbar.draw()
    canvasbar.get_tk_widget().place(x=25,y=600)
    toolbar2 = NavigationToolbar2Tk(canvasbar, self.root).place
       (x=25,y=600)
    return None
pass


def plot_graphs_bp(self):

    fig3 = plt.figure(figsize=(6.5,3.8),dpi=100)
    ax1 = fig3.add_subplot(111)
    ax1.set_xlabel('samples')
    ax1.set_ylabel('SBP (mmHg)', color='black')
    ax1.set_title('SBP')

    ax1.plot(self.s3,'r',label="Predicted SBP")

    if (self.var1.get()==1):
        ax1.plot(self.s3_real,label="Real SBP")
    elif(self.var1.get()==0):
        pass
```

```
ax1.legend(loc='upper left')
ax1.set_ylim(60,200)
ax1.set_yticks([120,130,140,160,180])
ax1.axhline(120,color ='#18AD47',linestyle ="--",linewidth=
    0.5)
ax1.axhline(130, color ='#69DB4E',linestyle ="--",linewidth
   = 0.5)
ax1.axhline(140, color ='#FFF829',linestyle ="--",linewidth
   = 0.5)
ax1.axhline(160, color ='#FF9C19',linestyle ="--",linewidth
   = 0.5)
ax1.axhline(180, color ='#FA2E15',linestyle ="--",linewidth
   = 0.5)

if self.var3.get()==1:

    ax2 = ax1.twinx()
    ax2.plot(self.hr, 'green', label="HR")
    ax2.set_ylim(60, 120)
    ax2.set_yticks([ 70, 80, 90, 100, 110])
    ax2.set_ylabel('HR (bpm)', color='green')
    ax2.legend(loc='lower right')

canvasbar=FigureCanvasTkAgg(fig3,self.root)
canvasbar.draw()
canvasbar.get_tk_widget().place(x=1250,y=200)

fig4=plt.figure(figsize=(6.5,3.8),dpi=100)
ax1 = fig4.add_subplot(111)
ax1.set_xlabel('samples')
ax1.set_ylabel('DBP (mmHg)', color='black')
ax1.set_title('DBP')
ax1.plot(self.s4,'r',label="Predicted DBP")

if (self.var2.get()==1):
    ax1.plot(self.s4_real,label="Real DBP")

elif(self.var2.get()==0):
    pass
ax1.legend(loc='upper left')
ax1.set_ylim(40,120)
ax1.set_yticks([80,85,90,100,110])

plt.axhline(80,color ='#18AD47',linestyle ="--",linewidth=
   0.5)
plt.axhline(85, color ='#69DB4E',linestyle ="--",linewidth=
    0.5)
plt.axhline(90, color ='#FFF829',linestyle ="--",linewidth=
    0.5)
```

```
        plt.axhline(100, color ='#FF9C19',linestyle ="--",linewidth
            = 0.5)
        plt.axhline(110, color ='#FA2E15',linestyle ="--",linewidth
            = 0.5)


        if self.var3.get()==1:
            ax2 = ax1.twinx()
            ax2.plot(self.hr, 'green', label="HR")
            ax2.set_ylim(60, 120)
            ax2.set_yticks( [ 70, 80, 90, 100, 110])
            ax2.set_ylabel('HR (bpm)', color='green')
            ax2.legend(loc='lower right')
        canvasbar=FigureCanvasTkAgg(fig4,self.root)
        canvasbar.draw()
        canvasbar.get_tk_widget().place(x=1250,y=600)
        return None

    pass


main()
```

References

[1] "H2020 structure and budget." https://ec.europa.eu/research/participants/docs/h2020-funding-guide/grants/applying-for-funding/find-a-call/h2020-structure-and-budget$_e$*n.html.VisitedinDecember*2022.

[2] S. Consortium, "Sintec project." Visited in December 2022. https://www.sintec-project.eu/, 2022.

[3] L. Foundation, "Links foundation." Visited in December 2022. https://linksfoundation.com/chi-siamo/, 2022.

[4] M. M. Baig, H. Gholamhosseini, and M. J. Connolly, "A systematic review of wearable patient monitoring systems - current challenges and opportunities for clinical adoption," *Journal of medical systems*, vol. 41, no. 7, p. 115, 2017.

[5] M. Smuck, M. Conroy, L. Murphy, A. Edmonds, and G. Demiris, "The emerging clinical role of wearables: factors for successful implementation in healthcare," *NPJ digital medicine*, vol. 4, no. 1, p. 45, 2021.

[6] J. Dunn, R. Runge, and M. Snyder, "Wearables and the medical revolution," *Personalized medicine*, vol. 15, no. 5, pp. 429–448, 2018.

[7] W. H. Organization, "The top 10 causes of death." Visited in December 2022. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death, 2022.

[8] H. M. Care, "Ipertensione." Visited in December 2022. https://www.humanitas.it/malattie/ipertensione/, n.d.

[9] S. E. Kjeldsen, "Hypertension and cardiovascular risk: General aspects," *Pharmacological research*, vol. 129, pp. 95–99, 2018.

[10] O. Healthcare, "Diario della pressione arteriosa e scheda della pressione arteriosa: Download gratuito per tenere traccia delle misurazioni." Visited in December 2022. https://www.omron-healthcare.it/it/salute-e-stile-di-vita/salute-del-cuore/gestione-della-pressione-arteriosa/diario-della-pressione-arteriosa-e-scheda-della-pressione-arteriosa-download-gratuito-per-tenere-traccia-delle-misurazioni.html, n.d.

[11] K. Kario, "Management of hypertension in the digital era: Small wearable monitoring devices for remote blood pressure monitoring," *Hypertension (Dallas, Tex. : 1979)*, vol. 76, no. 3, pp. 640–650, 2020.

[12] A. S. Meidert and B. Saugel, "Techniques for non-invasive monitoring of arterial blood pressure," *Frontiers in medicine*, vol. 4, p. 231, 2018.

[13] K. Kario, K. Shirai, T. Otsuka, T. Kuroda, H. Takahashi, Y. Umeda, H. Ishii, S. Hoshide, and K. Eguchi, "The first study comparing a wearable watch-type blood pressure monitor with a conventional ambulatory blood pressure monitor on in-office and out-of-office settings," *Journal of clinical hypertension (Greenwich, Conn.)*, vol. 22, no. 2, pp. 135–141, 2020.

[14] H. H. Hager and B. Burns, "Artery cannulation," *StatPearls*, 7 2022.

[15] Wikipedia, "Arterial line — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Arterial_line, 2022. [Online; accessed 10-March-2023].

[16] A. Argha *et al.*, "Artificial intelligence based blood pressure estimation from auscultatory and oscillometric waveforms: A methodological review," *IEEE reviews in biomedical engineering*, vol. 15, pp. 152–168, 2022.

[17] M. University, "Auscultation — McGill University Physiology Virtual Laboratory." https://www.medicine.mcgill.ca/physio/vlab/cardio/auscul.html, N/A. [Online; accessed 10-March-2023].

[18] D. Venztas, L. Tei, and E. Cardiologist, "Signal processing and knowledge acquisition in objective blood pressure measurement with conventional clinical sphygmomanometry," *Journal of Hypertension*, vol. 22, no. S1, pp. S21–S23, 2004.

[19] P. Lewis, on behalf of the British, and I. H. S. B. P. M. W. Party, "Oscillometric measurement of blood pressure: a simplified explanation. a technical note on behalf of the british and irish hypertension society," *Journal of Human Hypertension*, vol. 33, pp. 349–351, 2019.

[20] J. E. Sharman, I. Tan, G. S. Stergiou, and E. O'Brien, "Automated 'oscillometric' blood pressure measuring devices: how they work and what they measure," *Journal of human hypertension*, 2022.

[21] M. R. Nelson, J. Stepanek, M. Cevette, M. Covalciuc, R. T. Hurst, and A. J. Tajik, "Noninvasive measurement of central vascular pressures with arterial tonometry: clinical revival of the pulse pressure waveform?," *Mayo Clinic proceedings*, vol. 85, no. 5, pp. 460–472, 2010.

[22] T. Medical, "T-line r tl-300 nibp patient monitor." Visited in December 2022: https://healthmanagement.org/products/view/nibp-patient-monitor-t-line-r-tl-300-tensys-medical, 2022.

[23] Y. Liang, X. Chen, Z. Xu, H. Chen, and Y. Li, "A clinically accurate oscillometric blood pressure algorithm." Visited in December 2022: https://www.sciencedirect.com/science/article/pii/S1521689614000706?via%3Dihub, 2014.

[24] L. Beard and D. Ashton-Cleary, "Blood pressure measurement," in *Maths, Physics and Clinical Measurement for Anaesthesia and Intensive Care*, pp. 159–173, Cambridge University Press, 2019.

[25] T. Arakawa, "Recent research and developing trends of wearable sensors for detecting blood pressure," *Sensors (Basel, Switzerland)*, vol. 18, no. 9, p. 2772, 2018.

[26] "Omron heartguide." https://www.omron-healthcare.it/it/misuratori-di-pressione/heartguide.htmlprefn1. Accessed: December 2022.

[27] C. L. Stanfield, *Fisiologia*. Edises, iv ed., 2018.

[28] E. Houssein, M. Kilany, and A. E. Hassanien, "Ecg signals classification: a review," *International Journal of Medical Engineering and Informatics*, vol. 5, no. 4, pp. 376–396, 2017.

[29] A. Albulbul, "Evaluating major electrode types for idle biological signal measurements for modern medical technology," *Bioengineering (Basel, Switzerland)*, vol. 3, no. 3, p. 20, 2016.

[30] Wikipedia, "Electrocardiography." https://en.wikipedia.org/wiki/Electrocardiography. Visited in December 2022.

[31] M. Nitzan and Z. Ovadia-Blechman, "Physical and physiological interpretations of the ppg signal," in *Photoplethysmography* (J. Allen and P. Kyriacou, eds.), pp. 319–340, Academic Press, 2022.

[32] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, and H. Nazeran, "A review on wearable photoplethysmography sensors and their potential future applications in health care," *International Journal of Biosensors & Bioelectronics*, vol. 4, no. 4, pp. 195–202, 2018.

[33] A. Kavsaoğlu, K. Polat, and M. Bozkurt, "An innovative peak detection algorithm for photoplethysmography signals: An adaptive segmentation method," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, pp. 1782–1796, 2016.

[34] M. Elgendi, "On the analysis of fingertip photoplethysmogram signals," *Current Cardiology Reviews*, vol. 8, no. 1, pp. 14–25, 2012.

[35] M. Mehrabbeik and S. Rashidi, "Estimation of cuffless systolic and diastolic blood pressure using pulse transient time," in *2019 6th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pp. 1–6, 2019.

[36] X. Ding, B. P. Yan, W. Karlen, and J. Y. Cheung, "Pulse transit time based respiratory rate estimation with singular spectrum analysis," *Medical & Biological Engineering & Computing*, vol. 58, pp. 257–266, 2020.

[37] V. Figini, S. Galici, D. Russo, I. Centonze, M. Visintin, and G. Pagana, "Improving cuff-less continuous blood pressure estimation with linear regression analysis," *Electronics*, vol. 11, p. 1442, 2022.

[38] Wikipedia, "Accelerometer," 2023. [Visited in January 2023].

[39] I. Faisal, T. Purboyo, and A. Ansori, "A review of accelerometer sensor and gyroscope sensor in imu sensors on motion capture," *Journal of Engineering and Applied Sciences*, vol. 15, pp. 826–829, 2019.

[40] Z. Mohammed *et al.*, "Monolithic multi degree of freedom (mdof) capacitive mems accelerometers," *Micromachines*, vol. 9, no. 11, p. 602, 2018.

[41] NASA, "Human vestibular system in space," 2023. [Visited in January 2023].
[42] OpenStax, "Equilibrium," 2023. [Visited in January 2023].
[43] H. Zeng and Y. Zhao, "Sensing movement: microsensors for body motion measurement," *Sensors (Basel, Switzerland)*, vol. 11, no. 1, pp. 638–60, 2011.
[44] S. Galici, "Blood pressure monitoring in a non-invasive way using regression techniques." https://github.com/sofiagalici/Blood-Pressure-Monitoring-in-a-Non-Invasive-Way-Using-Regression-Techniques, 2019.
[45] S. Sensing, "Shimmer," 2023. [Visited in January 2023].
[46] S. Sensing, "About us," 2023. [Visited in January 2023].
[47] S. Sensing, "Shimmer3 ecg unit," 2023. [Visited in January 2023].
[48] S. Sensing, "Emg-ecg electrodes," 2023. [Visited in January 2023].
[49] Shimmer, "Shimmer3 gsr+ unit," 2023. [Visited in January 2023].
[50] Shimmer, "Optical pulse ear clip," 2023. [Visited in January 2023].
[51] T. InvenSense, "Icm-20948 9-axis motion sensor," 2023. [Visited in January 2023].
[52] Shimmer, "Consensyspro software," 2023. [Visited in January 2023].
[53] P. S. Foundation, "Python: About," 2023. [Visited in January 2023].
[54] X. Xu, H. Chen, X. Chen, X. Li, and X. Li, "Adaptive motion artifact reduction based on empirical wavelet transform and wavelet thresholding for the non-contact ecg monitoring systems," *Sensors*, vol. 19, no. 13, p. 2916, 2019.
[55] C. C. Wu, I. W. Chen, and W.-C. Fang, "An implementation of motion artifacts elimination for ppg signal processing based on recursive least squares adaptive filter," in *2017 IEEE Biomedical Circuits and Systems Conference, BioCAS 2017 - Proceedings*, pp. 1–4, Institute of Electrical and Electronics Engineers Inc., 2018.
[56] T. Tanweer, S. R. Hasan, and A. Kamboh, "Motion artifact reduction from ppg signals during intense exercise using filtered x-lms," 2017.
[57] P. H. Prajapati and A. D. Darji, "Two stage step-size scaler adaptive filter design for ecg denoising," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.
[58] "Visited in february 2023." https://chat.openai.com/chat/7e1cdfe4-f124-442a-a206-e8b16ce97209, 2023.
[59] E. Patti, "Lectures from "programming for iot", a.a. 2021-2022, politecnico di torino," 2021.
[60] G. S. Stergiou and et al., "A universal standard for the validation of blood pressure measuring devices: Association for the advancement of medical instrumentation/european society of hypertension/international organization for standardization (aami/esh/iso) collaboration statement," *Journal of hypertension*, vol. 36, no. 3, pp. 472–478, 2018.
[61] L. Wang and et al., "A novel blood pressure monitoring technique by smart huawei watch: A validation study according to the ansi/aami/iso 81060-2:2018 guidelines," *Frontiers in cardiovascular medicine*, vol. 9, p. 923655, 2022.

[62] J. C. for Guides in Metrology, "Guide to the expression of uncertainty in measurement - part 6: Developing and using measurement models," Technical report JCGM 106:2020, 2020.

[63] Wikipedia, "Mean." https://en.wikipedia.org/wiki/Mean, 2023. Accessed: February 2023.

[64] Wikipedia, "Variance." https://en.wikipedia.org/wiki/Variance, 2023. Accessed: February 2023.

[65] K. Kamide and M. Kabayama, "Implications of blood pressure variations in older populations," *Hypertension Research*, vol. 42, no. 1, pp. 19–25, 2019.

[66] R. Mukkamala and et al., "Evaluation of the accuracy of cuffless blood pressure measurement devices: Challenges and proposals," *Hypertension (Dallas, Tex. : 1979)*, vol. 78, no. 5, pp. 1161–1167, 2021.

[67] X. Jin, L. Li, F. Dang, X. Chen, and Y. Liu, "A survey on edge computing for wearable technology," *Digital Signal Processing*, vol. 125, p. 103146, 2022.

[68] Intel, "Healthcare at the edge." https://www.intel.com/content/www/us/en/, n.d.