UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

WINDKESSEL MODELING OF THE HUMAN ARTERIAL SYSTEM

Bachelor Thesis
Trabajo Fin de Grado
Biomedical Engineering

Oral presentation date:

Author: Nuria Peña Pérez
Supervisor: Manuel Desco Menéndez

Leganés, June 2016

ACKNOWLEDGEMENTS

I would like express my gratitude to the following people for their support. This thesis means the end of four very special years, and as probably the greatest challenge of this period, this would not have been possible without them.

To my supervisor Manuel Desco Menéndez, for giving me the opportunity to work in this project, allowing me to work independently and being available always that guidance was needed. Thank you for trusting me.

To Guillermo Vizcaíno, for his patience, time and endless knowledge. For teaching me so much. This would not have been possible without you. To all the people working in the lab, always opening the door with a smile, thank you for making my days special and transforming my time there into a great experience.

To all my friends for being such a source of encouragement and inspiration. I feel incredibly lucky of being surrounded by such amazing people.

To all my family, for having faith in me and for their care and support to succeed in making my dreams come true. Specially, to Ana and Javier, my parents, thank you for giving me everything, I will never be able to thank you enough. To my brother Raúl, for his understanding and support.

To Álvaro, for believing in me. For letting me believe in you. It´s been such a good year.

Nuria Peña, June 2015.

ABSTRACT

Cardiovascular diseases are a major concern of our society. Millions of patients all around the world are affected by disorders such as arrhythmias or atherosclerosis. Moreover, finding new diagnostic techniques and treatments is of increased difficulty due to the complexity of cardiovascular medicine. In this context, the upcoming generations of experts must be well prepared for overcoming such a challenge. This project aims to develop an educational tool that will allow students to improve their understanding on cardiovascular fluid mechanics and physiology and will allow them to gain practical experience before dealing with real patients. A system modelling the arterial system, available at the Universidad Carlos III de Madrid, is used for this purpose. The educational tool is composed by a theoretical simulation interface and an acquisition and control program, created using MATLAB, and a practical environment based on a physical pneumatic-hydraulic device. A laboratory practice for the students has been developed describing how to work with both platforms.

Key words:

Cardiovascular physiology, arterial modelling, Windkessel model, models in education, MATLAB.

RESUMEN

Las enfermedades cardiovasculares constituyen uno de los mayores problemas médicos de la sociedad actual. Millones de pacientes a lo largo del mundo se ven afectados por enfermedades como arritmias o arterosclerosis. Además, la búsqueda de nuevos tratamientos y métodos de diagnóstico se ve complicada por la dificultad inherente a la medicina cardiovascular. En este contexto, los futuros profesionales del sector médico deben estar perfectamente preparados para afrontar el reto. En este proyecto se ha creado una herramienta educativa orientada a mejorar la compresión en los temas de fisiología y mecánica cardiovascular y a proporcionar a los estudiantes una plataforma a través de la cual completar su educación con formación práctica, de cara a futuros retos con casos reales. La herramienta propuesta está compuesta por una interfaz de simulación teórica y un programa adquisición y control, creados utilizando MATLAB, y un aparato de simulación hidroneumático. Se ha creado también una práctica de laboratorio, describiendo a los estudiantes como interactuar con ambas plataformas.

Palabras clave:
Fisiología cardiovascular, modelado arterial, modelo Windkessel, modelos de uso educativo, MATLAB.

GENERAL INDEX

CONTENTS

Figure Index

Table Index

"There is no certainty in sciences where one of the mathematical sciences cannot be applied, or which are not in relation with these mathematics".

Leonardo Da Vinci.

1. GENERAL INTRODUCTION

1.1 Motivation

Medicine has evolved greatly from its origins. As time progresses it tries to adapt to the necessities of the new society. Emerging technologies contribute to this purpose while novel focus areas offer alternative perspectives to solve medical problems. However, as new treatments are found for existing diseases, aiming to eradicate some of them, new medical challenges arise. This is the case of vascular diseases which nowadays have become one of the most worrying conditions, affecting millions of patients all around the world [1].

Biomedical engineering emerged as one of the disciplines committed to help medical sciences. It has expanded considerably in the last decades, establishing itself as a fundamental field in the improvement of disease prevention, diagnosis, treatment and rehabilitation. Through the application of engineering concepts to solve medical issues it provides a singular point of view and favors the cooperation of biologists, physicist and doctors in multidisciplinary teams. Thus, engineering has become fundamental in the healthcare industry. One of the most remarkable tools that biomedical engineering provides is the possibility of using mathematics-based sciences to model biological processes.

When dealing with vascular diseases, mathematics (together with physics, electronics, etc.) may be used to model the anatomy and physiology of the circulatory system. These models, and their implementation in physical devices, offer significant advantages against traditional techniques and methods. In vascular research, models can be used when it is complicated or not practical to create experimental conditions. This allows to observe the system's performance in specific circumstances and provides with a better understanding of it, therefore helping in the search for new treatments. In training and education, models give the students the opportunity to work with an actual experiment, when they can adjust the parameters of the system and observe its response. Real system modelling also prevents them from committing failures during experimentation, offering them an environment to practice before dealing with real patients. All of this converts modelling in a very useful tool for the field of biomedicine, and in particular of biomedical training.

This project emerged from the availability of a physical device that models the arterial system located at the laboratories of the Universidad Carlos III de Madrid. For the system to be converted in a reliable and effective educational tool, it had to be completed, improved and adjusted to the teaching necessities. Once implemented, it would constitute a very useful instrument for making use of the advantages that models offer to education. Biomedical engineering students must be prepared to overcome any type of challenge related to any of the disciplines integrated in biomedical sciences and engineering. This demands a broad education where theoretical knowledge must be completed with practical instruction. In this context, including a tool such as the proposed in this thesis, might be of huge help in the instruction of students. It would help them to learn in a more realistic environment without the need of treating directly with patients, and it would improve their understanding of a field as complex and important as vascular physiology. All of this will contribute to keep a high quality, cutting-edge instruction.

1.2 Objectives

The **general goal** of this project is to set up an educational tool for training students, focusing on the topic of human vascular fluid mechanics. Two different learning platforms will be provided, allowing the students to improve both their theoretical and practical knowledge. In this way, the **specific goals** of this project are:

- To create a “theoretical” educational software platform. The platform will be constituted by a simulation interface where the students will be able to improve their knowledge on vascular physiology. By using this platform, students will be able to observe the role that different physiological parameters (e.g., changes in the heart rate) have in the circulatory system, and to understand how they affect to aortic pressure and flow.
- To set up a “practical” environment where students can reinforce the concepts learnt during the lessons. This will make use of the physical device available at the university. The students will be able to define simulation parameters and acquire data coming from the simulated vascular device in a software platform specifically designed for this purpose.
- To design a laboratory practice that defines the workflow to be followed in the experience with both platforms. This will allow the students to observe and discuss the differences and similarities between the output given by the theoretical model itself and by its implementation in a physical device, and to see how these are related to real data from human patients.

1.3 Structure

This thesis is structured into six main chapters, each containing different subsections.

The first chapter of this thesis explains the context and the origin of the motivation that inspired this project, defining the main goals that will be pursued during its development. It also includes a brief description of the legal framework, explaining what considerations have been taken.

The second chapter provides with the necessary background required to understand the proposed solution and its implementation. It also discusses currently available techniques and devices based on the use of models that aim to contribute to the field of vascular medicine. Relevant characteristics of the different studies will be considered for achieving a feasible and successful solution to a real need.

The third chapter comments the limitations and obstacles to take into account for the development of a solution. It describes the user and technical requirements.

Along the fourth chapter the different solutions adopted for achieving each of the specific objectives are presented in detail. It explains why the selected approach accomplishes with the user requirements and describes the considerations taken for each of the elements integrating the system. In this way, the chapter includes the core of the development of the project, providing the necessary material for arriving to the first functional implementation.

The fifth chapter shows some of the results obtained and some significant figures regarding the behavior of the system and its characteristics.

Finally, a conclusion will be provided together with future work suggestions.

Additional sections are defined outside the principal chapters:

- Acknowledgements
- Abstract, including its Spanish translation
- References
- Annex, containing the following relevant information:
    - General costs
    - Practice for students
    - Schemes
    - Full Mathematical Development
    - Codes

1.4 Legal Framework

As previously explained, the system has been developed as an educational tool, aimed to be used at the Universidad Carlos III de Madrid for training students. Since the device will not be distributed, and no commercial transactions will take place, the device is exempt from following regulation stated by the European legislation.

European Union market regulation is managed by the Directorate-General for Internal market, Industry, Entrepreneurship and SMEs. Products in the European Economic Area (EEA) require an evaluation to ensure their high safety, health, and the requirements of environmental protection. This is achieved through CE marking. By affixing the CE marking on a product, the manufacturer declares that the product complies with all legal requirements for CE marking and can be sold throughout the EEA. Products that are manufactured in other countries and sold in the EEA also follow this regulation [2]. The latest legislative framework was adopted in 2008, through a packet of measures that aim to improve market surveillance and boost the quality of conformity assessments.

Since the development of the device did not require animal experimentation it is exempt from the guidelines set by order 53/2013 [3].

Moreover, since the system will not be used in clinical trial and it is not defined as medicinal product for human use, it is exempt from following the regulation stated by the European Union. In this context, all clinical trials performed in the European Union are required to be conducted in accordance with the Clinical Trials Directive [4]. On May 28th 2016 the new Clinical Trials Regulation (CTR) EU No 536/2014 will become applicable.

The device meets however UNE-EN ISO 12100 requirements specific for risk management and security in machinery design [5]. This order provides the procedure for risk identification and evaluation during the relevant phases of a machine life cycle. This includes considering aspects such as geometric and physical characteristics, electric and hydraulic risks, etc.

2. THEORETICAL FRAMEWORK

5.1. Vascular Anatomy and physiology

In order to understand the theoretical model included in this project and the functioning of the physical device, it is important to understand some of the concepts of vascular anatomy and physiology.

The cardiovascular system is composed by the heart, the blood and the blood vessels.
The heart is a relatively small, four-chambered muscle that rests on the diaphragm, close to the midline of the thoracic cavity. It is in charge of enabling the circulatory flow, pumping blood through the blood vessels. As continuous flow is demanded, the heart is required to beat approximately 100,000 every day, which supposes 2.5 billion beats in an average lifetime [6].
Fig. 1 shows the internal structure of the heart, which is divided in two sections each containing an atrium and a ventricle.

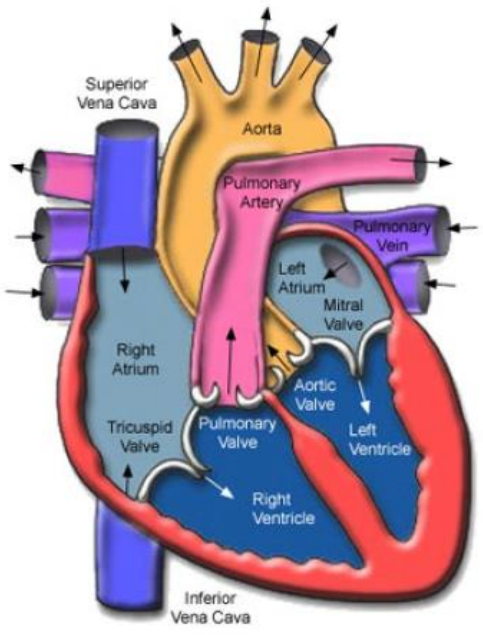


Figure 1: Internal structure of the heart [7].

The right atrium (RA) collects the un-oxygenated blood from all the body except the lungs. Blood flows from the RA, to the right ventricle (RV), where it is pumped to the lungs to receive oxygen. Oxygenated blood returns from the lungs and enters to the left atrium (LA) and travels into the left ventricle (LV). The left ventricle finally pumps the oxygenated blood to the aorta, one of the major arteries in the body. The aorta branches out into a series of major and then minor arteries (arterioles, with smaller diameter), and finally into a series of capillaries where gas exchange and diffusion occur.

To ensure the switch from deoxygenated blood entering the heart to oxygenated blood leaving the heart, the vascular system has two different subsystems: the pulmonary system and the systemic system.

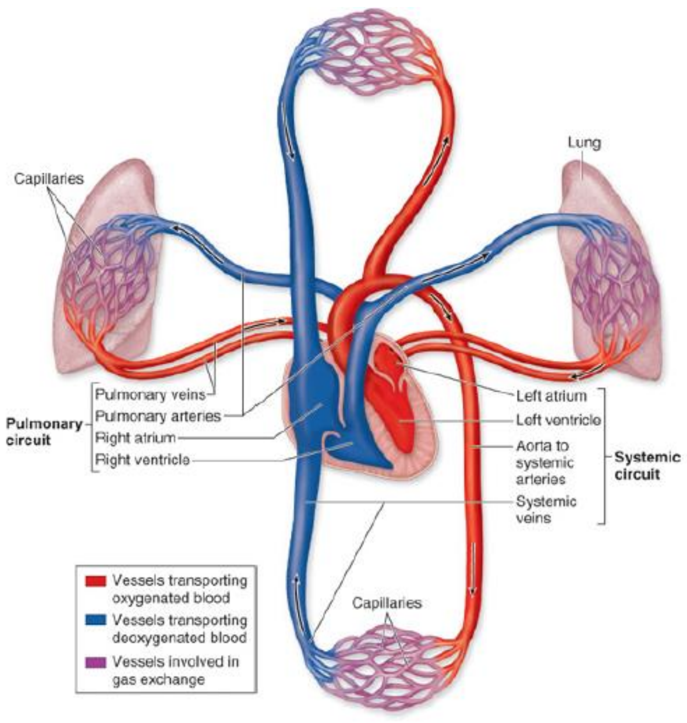


Figure 2: Subsystems of the cardiovascular system [7].

Each of the ventricles is providing blood to a different subsystem during the cardiac cycle, to guarantee no deoxygenated blood travels to systemic circulation without entering the lungs first. The cardiac cycle is typically described in two phases, diastole and systole, happening in a controlled time period T. The first step of diastole consists in the filling of the right and left atria, while the ventricles are relaxing (isovolumetric relaxation, which lowers their pressure). After the tricuspid and mitral valve open due to the increase of pressure in the atria, the second step of diastole arrives and both ventricles are filled with blood. The first stage of systole consists in the isovolumetric contraction of the right and left ventricles. Once pressure inside the ventricle is high enough the second stage starts: aortic and pulmonary semilunar valves open and blood is ejected to the aorta and pulmonary arteries respectively [8]. Fig. 3 shows each of the described steps.

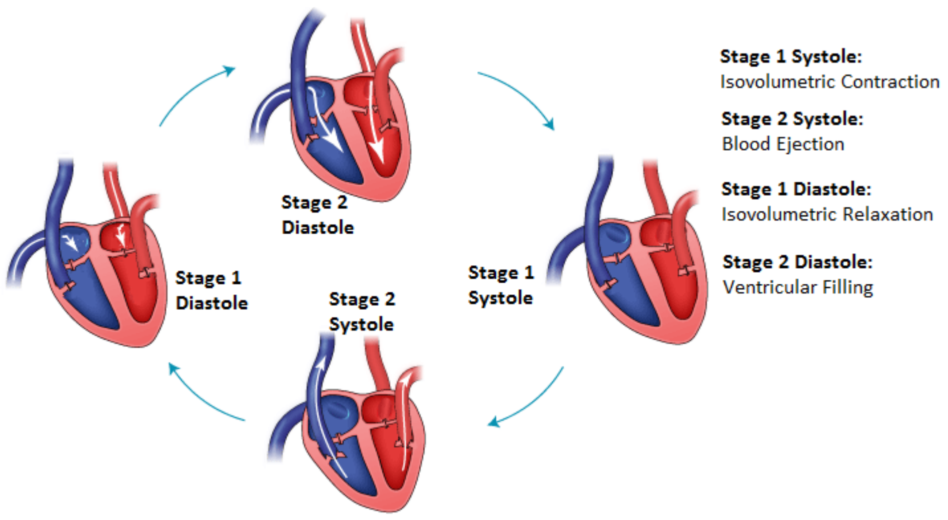


Figure 3: Steps of the heart cycle [9].

The description of systole and diastole above, demonstrates the mechanics happening at the aortic valve. It is clear that changes in ventricular pressure causes the aortic valve to open and close, and initiates the flow of blood through the valve. It is important to mention that this project will consider pressure and flow just outside the left ventricle, once the blood has traversed the aortic valve. In the fig. 4, differences between the pressures at the left atrium, ventricle and aorta are shown, together with contrasts between aortic flow and ventricular volume.

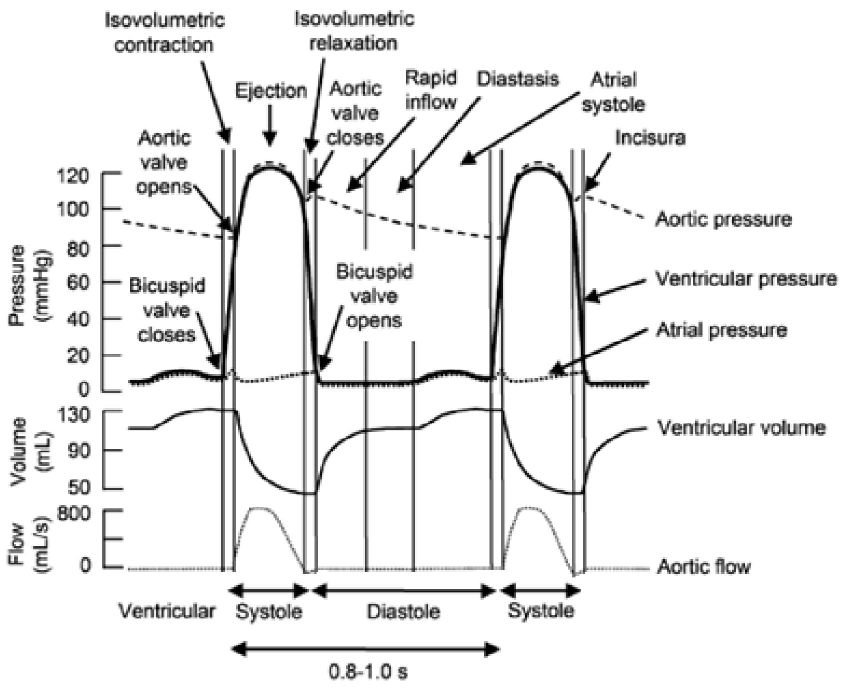


Figure 4: Comparison between aortic, ventricular and atrial pressure. Comparison between ventricular volume and aortic flow [8].

Some of the characteristics that will be used to generate the theoretical model may be recognized in the image, such as the absence of flow at the aorta during systole or the range of values for the aortic pressure (that never decreases to zero).

5.2. Cardiovascular diseases

Cardiovascular diseases are a group of disorders that are related to the circulatory system and therefore affect to the heart or the blood vessels. According to the World Health Organization [1] cardiovascular diseases are currently the leading cause of dead in the world.

Disorders such as sudden cardiac arrest, atrial fibrillation, peripheral artery disease, stroke or heart failure caused approximately 1 death each 40 seconds in the United States (USA) according to data from 2013 [10]. This issues are strongly influenced by habits such as poor diet, lack of exercise or smoking, and by other factors such as genetic predisposition. As an example, one of the major risk factors for cardiovascular diseases, high blood pressure (HBP), affects to more than 50% of the population aged between 55 and 64 in the USA (Fig. 5a). The prevalence of this condition has generally increased over the last decades (Fig. 5b). If this tendency continues, the search for a solution will require from all the efforts of the upcoming generations of experts. This is why education in the field of vascular medicine is such a critical matter, for all students aiming to work in biomedical sciences.

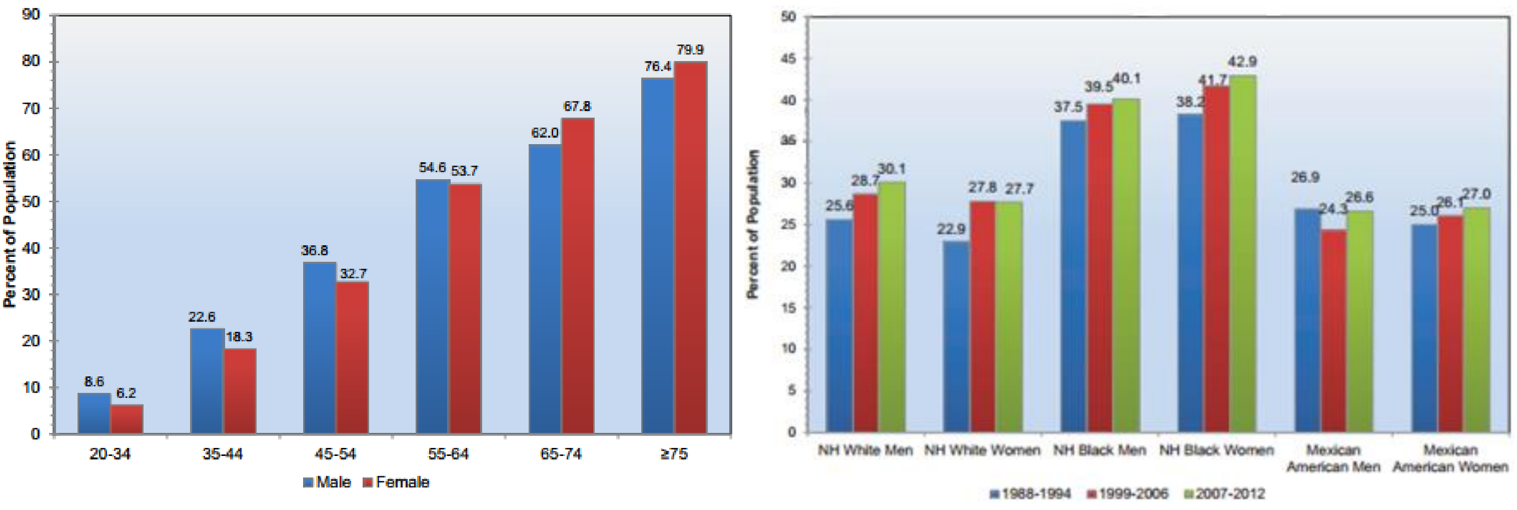


Figure 5: Prevalence of high blood pressure in adults ≥20 years of age by age and sex (National Health and Nutrition Examination Survey: 2007–2012) (a). Age-adjusted prevalence trends for high blood pressure in adults ≥20 years of age by race/ethnicity, sex, and survey (National Health and Nutrition Examination Survey: 1988–1994, 1999–2006, and 2007–2012) (b) [10].

5.3. Arterial Models

The complicated interactions between the different physiological processes and control mechanisms of the cardiovascular system, convert it in one of the most complex to understand. In order to provide a better diagnosis and a better understanding of its physiology, different approaches have been used. Nowadays, mathematical description and modeling of the human cardiovascular system plays an important role in the comprehension of cardiovascular disorders, providing tools such as computer modeling and simulation in physical devices.

Many different models exist, aiming to provide a representation of the arterial system. These models can be classified according to the features of the system they represent. It is important to take into account that in general there are not "good" or "bad" models, but models that suit the requirements of different applications [11]. Each model has a different goal but all of them aim to understand the vascular system in a non-invasive way. The principal arterial models are: Anatomically based distributed models, tube models and lumped models [12].

For the development of this project a lumped model is going to be considered. Lumped models are described as being 0-dimensional models since they consider that the distribution of the fundamental variables of the cardiovascular system (pressure, flow, etc.) is uniform along all its compartments (different organs and vessels) at every moment of time. If an experiment wants to take into account spatially distributed phenomena a different type of model must be selected. In 1-dimensional models, wave transport effect can be readily represented, and the variation of the velocity of the flow along a blood vessel can be described. In 2-dimensional models radial changes of the flow velocity can be easily represented in tubes showing axial symmetry. For some applications, where a complex description of the flow at specific locations of the arterial tree (bifurcations of the vessels, inside ventricles, through heart valves, etc.) needs to be done, 3-dimensional models may be used [11].

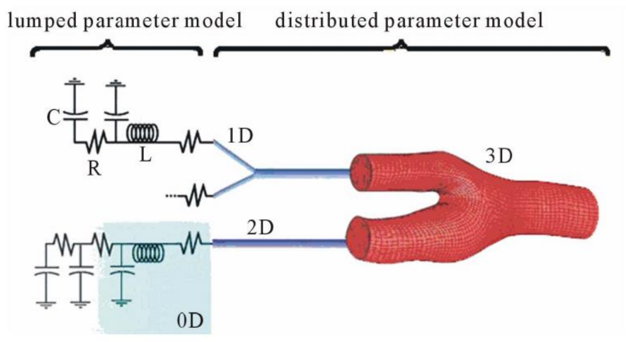


Figure 6: Dimensional description of arterial models [11]

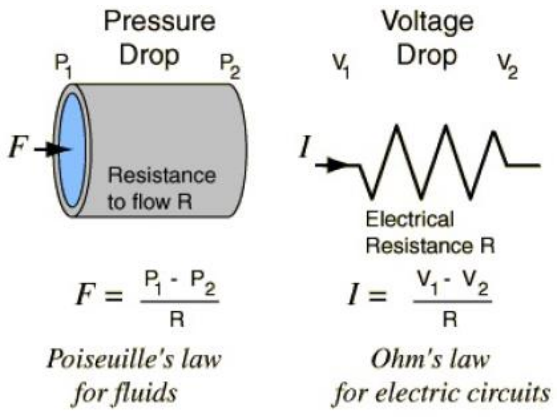
The main limitations of lumped models are therefore their incapability to study wave transmission phenomena, variations in the blood flow distribution and effects of local vascular changes. However, complexity may be added to these models by building multiple compartment models. These allow to avoid considering the entire systemic tree as a single block, permitting to establish differences in the fundamental variables between distinct segments of a vessel. Each of the segments is described as an independent lumped model. This has the advantage of allowing to develop a flexible model that provides more detail in the areas of interest [13].

For this project, as that much detail is not required, a single compartment lumped model, representing the entire systemic arterial tree will be used.

5.4. The Windkessel Model

The Windkessel model of the human arterial system is a lumped model that represents the systemic arterial tree. This model was designed by the German physiologist Otto Frank in the late 1800's [14]. The name of the model comes from the German word that means air chamber.

The Windkessel model describes the systemic arterial system as an analogy of a hydraulic or electric circuit. This may be observed in Fig. 7.



Pressure Drop
P_1
P_2
F
Resistance to flow R
$F = \frac{P_1 - P_2}{R}$
Poiseuille's law for fluids
Voltage Drop
V_1
V_2
I
Electrical Resistance R
$I = \frac{V_1 - V_2}{R}$
Ohm's law for electric circuits

Figure 7: Analogy between Ohm's and Poiseuille's law [15].

In Otto Frank's original design, the circuit contained a water pump connected to a chamber, filled with water except for a pocket of air. When the pump starts working the water compresses the air, which pushes the water outside the chamber. This analogy may be also established with an electric circuit and it resembles the mechanics of the heart. However, when considering an electric circuit, the blood is no longer represented by water, but it is

represented by electrons. The driving force of these electrons is no longer the pressure (P), but a voltage difference (U). The flow is no longer the volume change over a certain time (V), but the variation of charge over time (q), which is the intensity (I).
The main parameters of the original Windkessel model (also called two-element Windkessel model) are the peripheral resistance and the arterial compliance [16]. It order to improve it, some other parameters can be included building Windkessel models of higher complexity [17].

5.4.1. Two Elements Windkessel Model

As previously mentioned, the two-element Windkessel model describes artery behavior by means of two parameters: peripheral resistance and aortic compliance.
Peripheral resistance refers to the resistance to the flow of blood as it flows through the systemic arterial system. The higher the resistance, the higher the pressure difference for a given rate of flow.

$$R_p = \frac{Pressure\ difference}{Change\ in\ flow\ rate} = \frac{P_1 - P_2}{Q_1 - Q_2} = \frac{\Delta P}{\Delta Q}$$

Regard the similarities of the formula above with electric Ohm’s law, by using the analogies described in the previous section. This causes peripheral resistance to be represented with an electric resistor in the electrical presentation. (See Fig. 8)

$$\begin{aligned} &Hydraulic\ relationship: && \Delta P = R_p \cdot \Delta Q \\ &Ohm's\ law: && V = R \cdot I \end{aligned}$$

Hydraulic resistance depends on the length and the radius of the vessel, and the viscosity of the fluid [18]. In this way, more resistance is encountered in small vessels, which means that the part of the system accounting for most of the resistance will be arterioles and capillaries.

$$R_p = \frac{8\mu l}{\pi R^4}$$

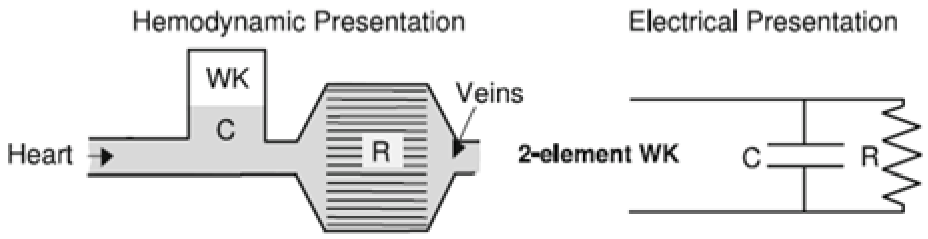


Figure 8: two-element Windkessel (hemodynamic and electrical presentation) [17]

Nevertheless, big arteries are characterized by a different parameter: compliance. Which may be defined as [18]:

$$C = \frac{A}{\rho g}$$

This parameter is related to the elasticity and extensibility of the arteries. Compliance refers to the volume change produced in an artery when subjected to a certain pressure [17]:

$$C = \frac{\Delta V}{\Delta P}$$

Realizing that the flow is the variation of volume over time, a relationship between compliance, pressure and flow may be defined:

$$Q = \frac{dV}{dt}$$

$$\Delta Q = C\frac{dP}{dt}$$

This shows why aortic compliance is represented as a capacitor in the electrical presentation, as the same procedure can be used to define capacitance (C_a) in terms of the intensity (current) and the derivative of the voltage.

$$C_a = \frac{\Delta q}{\Delta U}$$

$$I = C_a \cdot \frac{dU}{dt}$$

5.4.2. Three Elements Windkessel Model

For developing the three-element Windkessel model a new parameter is included: characteristic impedance (Z_a). The resulting model can be considered a link between a lumped model and a model including wave travel aspects. This arises from the fact that characteristic impedance is defined as wave speed times blood density divided by (aortic) cross-sectional area [17].

$$Z_a = \frac{v\rho}{A}$$

When developing a dimensional analysis, it can be observed that characteristic impedance has the same units as peripheral resistance [19]. More detail is given in section 4.1.2.

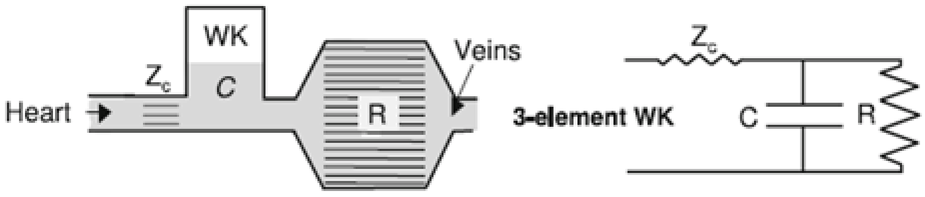


Figure 9: three-element Windkessel (hemodynamic and electrical presentation) [17]

5.4.3. Four Elements Windkessel Model

For describing a more complex Windkessel model, a parameter representing the fluidic inertia of the blood as it is cycled through the system. This parameter, called inertance (L), may be included in the model in two different ways, defining the four elements-series Windkessel model and the four elements-parallel Windkessel model [20].

Fig. 10a shows the four-element Windkessel in its series configuration, while fig. 10b shows the parallel configuration of the model.

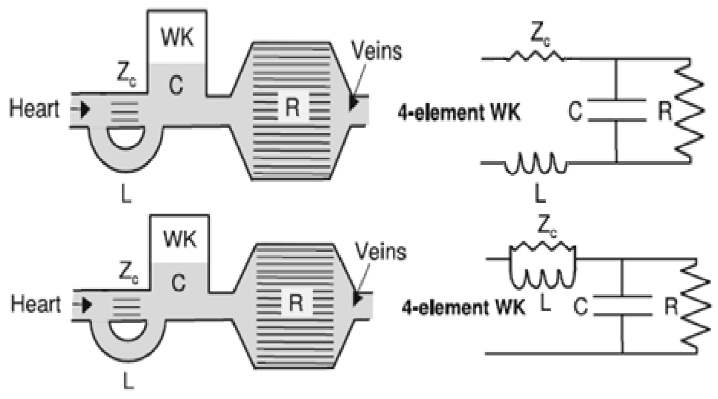


Figure 10: Four-element-series Windkessel model (a), four-element-parallel Windkessel model (b) [17]

As described by the relationship below [21], inertance depends on the length and the cross-sectional area of the vessel and on the fluid density.

$$L = \frac{l\rho}{A}$$

This parameter is related to flow and pressure through the following equation:

$$\Delta P = L\frac{dQ}{dt}$$

This shows why inertance appears as an inductor (L_i) in the electrical representation, since:

$$\Delta U = L_i\frac{dI}{dt}$$

5.5 State of the Art

One of the fields that has benefited the most from the use of models is cardiovascular medicine. Models in this area are used with three main purposes, contributing to research, training, and marketing demonstrations.

Models in Research

Nowadays models are commonly used for different applications as they offer serious advantages compared to traditional experimentation, such as cost reduction and shortening of the time required to perform the experiments. However, when using a model for research, it is important to remember that they cannot completely replace measurements as they are not fully accurate; models are only approximations to reality. Nevertheless, if the assumptions used to elaborate the model are taken into account when interpreting the results, they can be of huge help.

Although traditionally animal models were the most used with research purposes [22] and in the cardiovascular filed animals with vessels of small diameter were the most demanded [23], currently computational hemodynamics has arisen as a powerful tool for improving the efficiency of scientific experimentation, knowledge transfer and technology development [24].

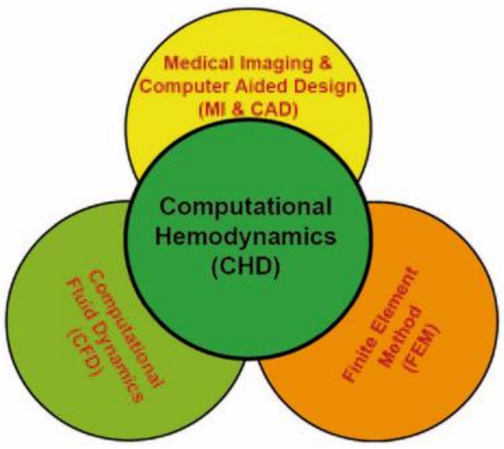

Medical Imaging & Computer Aided Design (MI & CAD)
Computational Hemodynamics (CHD)
Computational Fluid Dynamics (CFD)
Finite Element Method (FEM)

Figure 11: Techniques involved in Computational Hemodynamics [24].

This technique allows to observe flow patterns that are difficult, expensive or impossible to study using traditional (experimental) techniques [25]. In this way, lumped models such as the Windkessel model are not usually used for research purposes, being anatomical distributed models the preferred option. Most of the work developed in the field is related to graft generation [26], design of artificial valves and stents [27] and study of cerebral aneurysms [28].

Marketing Demonstrations

Models may be used as interactive tools for effectively communicating new procedure ideas or the products offered by a company. These type of models can be found at corporate meetings and customer demonstrations. As an example, the brand Pulse (Medical demonstration Models) [29] offer biomedical companies different anatomical models manufactured according to their criteria. The purpose of these models is to respond as the human organisms to the tool manufactured by the biomedical company. In this way, products manufactured (e.g. a catheter) will be showed to clients working in a realistic environment.

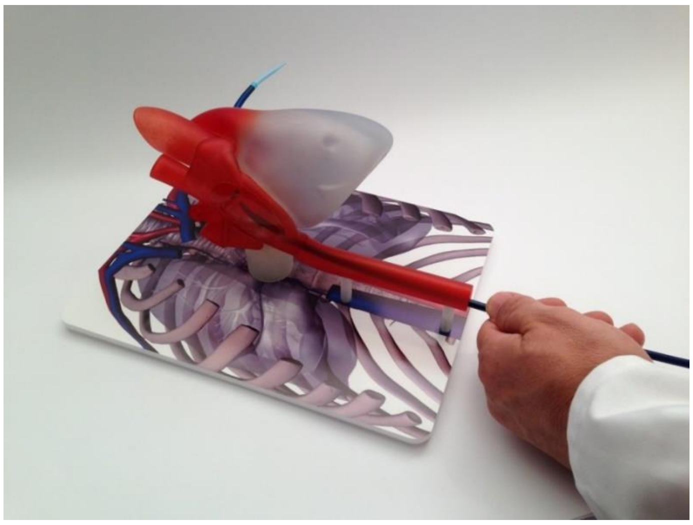


Figure 12: Model of a heart (manufactured by Pulse) used to show the performance of a device [29]

<u>Training</u>

Finally, models can be used in education. By looking at the bibliography it has been observed that there are two main types of models used in training.

Firstly, physical models can be used in education. These models are mainly used in physician training, aiming to improve patient care by offering a training environment that replicates interventional scenarios. The use of vessel, valves and organ replicas, combined with

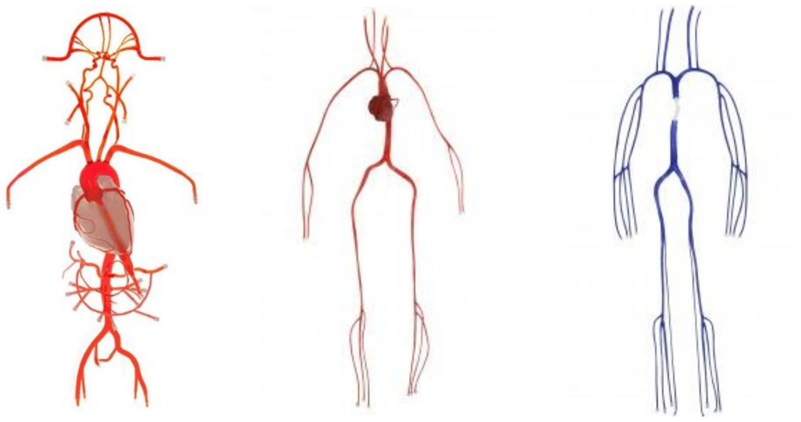
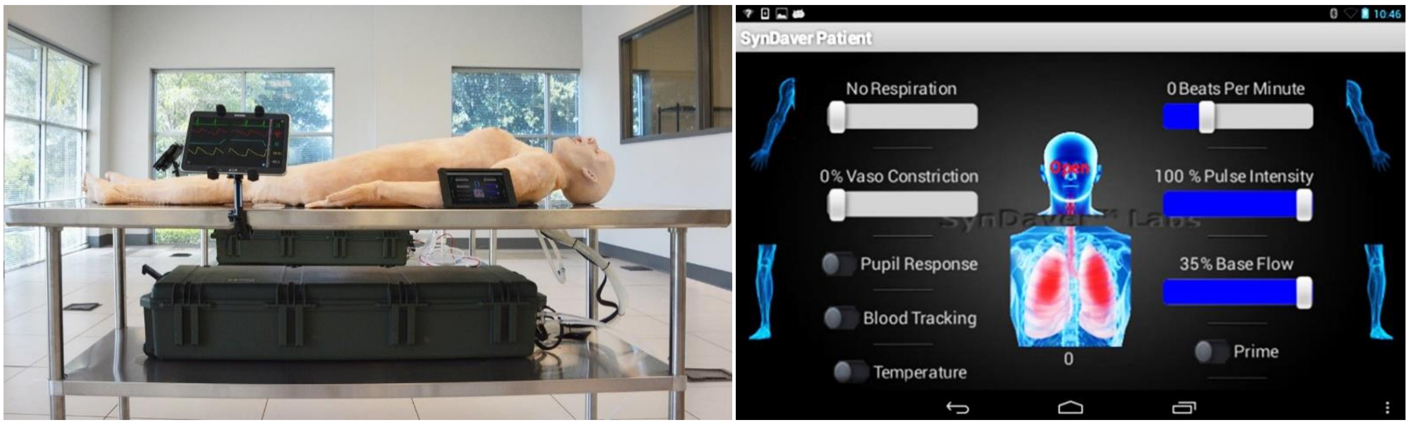


Figure 13: Different cardiovascular models offered by United Biologics [30]

diagnostic tools and devices can help in the demonstration of surgical procedures, treatment of pathologies and diagnostic techniques. Companies such as United Biologics [30] offer different products for its use in patient demonstrations, professional training or courses offered at medical schools. This models may cost around 5000$.

One of the most complete physical model observed during the bibliography review was the "*SynDaver Patient*" (Fig. 14a) [31]. It consists on a complex anatomical model of the entire human body, which includes also some of its physiological functions such as respiration rate, tidal volume, end-tidal CO2, heart rate, heart waveform, arrhythmia, systemic vasoconstriction, system-wide blood volume, body temperature, blink rate and pupil dilation. Body motions are controlled by an open-source physiology engine (Fig. 14b).

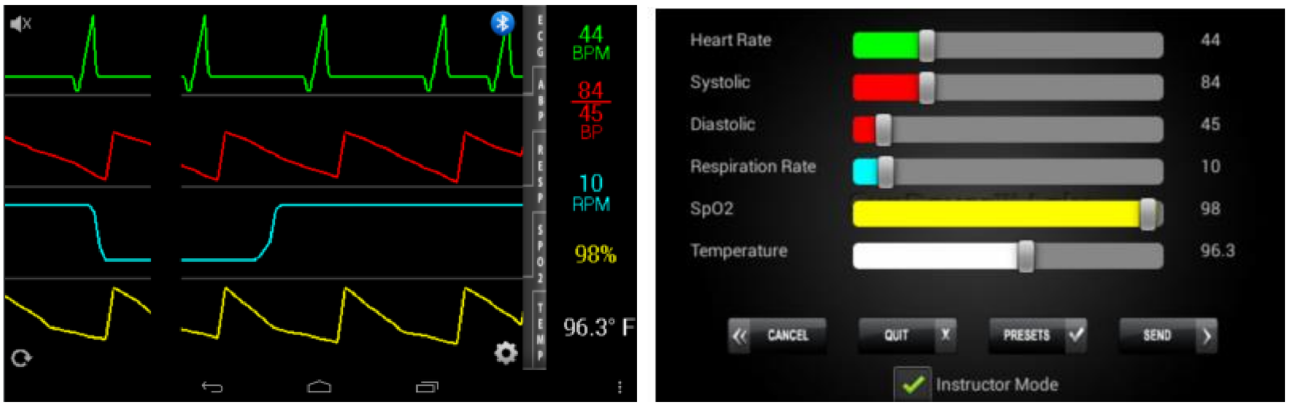
The cost of this device is of around 85000 $ [32]. This model can be ideal for preparing a physician for all type of interventions. However, since the aim of this model is to represent general body functions, cardiovascular signals are not modeled so thoroughly.



No Respiration
0 Beats Per Minute
0% Vaso Constriction
100 % Pulse Intensity
Pupil Response
35% Base Flow
Blood Tracking
Temperature
0

Figure 14: SynDaver Patient (a) and its controlling open-source engine (b) [31].

Secondly, software models can be used in education. Again, the company SynDaver Labs, offer an interactive electrocardiogram (EKG) simulator [33]. It allows the user to observe variables such as the heart rate, systolic pressure, diastolic pressure, respiration rate, SpO2, and temperature.



44 BPM
84
45 BP
10 RPM
98%
96.3° F

Heart Rate	44
Systolic	84
Diastolic	45
Respiration Rate	10
Sp02	98
Temperature	96.3

✓ Instructor Mode

Figure 15: SynDaver EKG simulator (a) and controls (b) [33].

However, observing the pressure waveform (red line), and comparing it with a measured pressure signal, still differences can be found. Meaning that the model is not completely accurate.

Several mobile applications are also available providing with different software models (mainly focusing on ECG signals) of the cardiovascular system and available to any student that has a cell-phone [34]. This proves the increasing importance of the use of cardiovascular models in education.

Two more studies related with software models must be mentioned:

- In the first place, MATLAB offers a SIMULINK toolkit specialized in cardiovascular simulation [35]. The main advantage of this toolkit is that any type of circuit configuration may be implemented by using the predefined SIMULINK programming blocks, as most of the elements present in the cardiovascular system are available (e.g. specific valves and vessels). However, this tool requires the user to know how to program and to be familiarized with SIMULINK language and structure. Moreover, the toolkit has an additional cost, even for users with a MATHWORKS license.
- In the second place, PhysioNet offers a cardiovascular simulator. The model is capable of generating reasonable human hemodynamic waveforms by using a two-element Windkessel model [36]. This tool allows access to the code in case the user wants to modify it. This seemed as a useful feature, so it was decided to use it in the developed system.

In conclusion, models are very used in the field of vascular medicine. Regarding models used in education there are many tools available for physical simulation or software simulation. However, it seems there are not many tools aiming to integrate both aspects, allowing the students to improve both their theoretical and practical education. With this project an accurate, both physical and computational, representation of selected cardiovascular variables will be achieved.

3. SYSTEM DESIGN

Different tasks were identified in order to achieve the proposed goals. The main tasks defined were:

- To identify the current situation of the system
- To establish a plan defining the steps to comply with user and technical specifications

- Regarding software simulation:
    - To develop mathematical models
    - To program the user interface
- Regarding physical simulation:
    - To upgrade the device
    - To establish device-computer communication
    - To program the user interface
- To assess the final models and write a practice for students

The context for developing a successful solution is described in three sections: system requirements, which studied the situation of the device and what features are required; software requirements, for both interfaces; and hardware requirements, regarding specific components. For more detail, specifications of individual components will be included in the annex and design considerations will be discussed in chapter four.

3.1 System requirements

The existing platform intended to be used in this project is a pneumatic-hydraulic device, located at the Universidad Carlos III de Madrid. This system is a prototype that simulates the arterial system using a Windkessel configuration, and it was manufactured by the company SEDECAL. Two major issues were considered in order to work with this device.

Firstly, the device had to be fixed as it was not working. The problem was found to be an issue with the valves and the membrane pump due to prior erroneous use.

Secondly, the configuration of the device does not enable measuring pressure or flow inside the heart, although allows easily measuring aortic pressure and flow, which are the main variables obtained using the Windkessel model. This was found not to be a problem considering that the Windkessel model is a very effective tool to explain the concepts that are required for the instruction of a biomedical engineer. It only requires few parameters and its analogy to an electric circuit makes of it an easy model to explain during lessons. Moreover, even for students who does not know the configuration of the model and with no previous background on electricity or fluid mechanics, it is easy to relate each of the parameters to simple characteristics of the vascular system (e.g. compliance to elasticity) and therefore with factors influencing diseases (e.g. decrease compliance and hypertension). In this way, the device seemed adequate for its use with educational purposes [37].

3.2 Software requirements

Two user interfaces will be developed, one in charge of the theoretical simulation and another that reads and displays data from the hydraulic device (practical simulation interface). The general user requirements for both interfaces are:

- They must be user-friendly, all buttons must be labeled and clear instructions must be provided

- No extra programming must be required, the graphic interface will be the intermediary between user and code
- Graphs must be relevant and clear, axes must be labeled and units must be the same in both interfaces for allowing easier comparisons
- They must be understandable both for engineers and physicians
- They must be simple, allowing users with poor vascular physiology background to use the interface

Moreover, both codes must be well structured and described, allowing to add more features in the future and facilitating interested users the reading of the code. In this way both codes will be structured in functions, so that each section can be easily accessed, and it will be heavily commented.

The specific user requirements for the theoretical simulation interface are:

- Navigation must be easy, so a pull down menu will be available at any moment that allows switching to different windows and returning to the origin
- User should not be required to know the characteristic value of any of the parameters. In this way, sliders will be included for a straight forward interaction with the interface and a reset button will allow the return to a reference
- Precision must be allowed, so the possibility of manually introducing values will be also included
- Both International System (SI) units and medical units must be displayed, so that users with different backgrounds will understand data values
- It must allow the user to observe aortic pressure and flow waveforms, as well as aortic pressure-volume loops.
- Different Windkessel configuration should be included so that the user can simulate simpler or more complex arterial representations.

- The interface must allow users to define and modify the main Windkessel parameters for each of the models included:
    - Parameters common to all models
        - Heart rate, which determines heart period
        - Systole duration, which determines duty cycle
        - Number of cycles to represent
        - Cardiac output
    - Two-element Windkessel
        - Peripheral resistance
        - Aortic compliance
    - Three-element Windkessel
        - Characteristic impedance
    - Four-element parallel Windkessel
        - Inertance

The specific user requirements for the practical simulation interface are:

- Navigation must be easy, so a pull down menu will be available at any moment allowing switching between the different options and returning to origin
- The user must be able to decide between reading pressure or flow data and select the amount of data desired
- The interface must allow the user to observe pressure and flow data acquired from the device in real-time
- In addition to the real-time display, a file must be generated for storing the data, so that it is downloadable by the user.
- Timing (systole or diastole cycle) was manually controlled by the user in the original device. The user should also be able to control timing through simulation software, so both working modes must be available at the interface.
- In the software timing control mode, the user must be able to select between a few selected, device-compatible, pulse rate and duty cycle options.

In both cases MATLAB has been chosen as the program environment to be used, more details on this choice are given in section 4.1.

3.3 Hardware requirements

Regarding the elements used in the computer-device communication, two transducers (one for reading flow and another one for reading pressure) will be used. Both transducers support RS-232 interface. This determined that the communication protocol used to receive and send data would be serial communication. However, in order to communicate with the serial ports at the computer, as it lacks in RS-232 ports, an USB-to-RS-232 converter cable will be used.

Regarding the components used to upgrade the device, electronic circuits designed and boards manufactured must be modular, safe and everything must be well documented. When the device was opened to modify the electronics, it was observed that documentation provided by the company did not match the actual components used and their distribution. Providing a faithful representation of the circuits used is of critical importance for allowing future maintenance and upgrades, as well as for safety issues. The modified circuits and diagrams are described in section 4.2.2. and included in the annex.

4. PROJECT DEVELOPMENT

4.1. Theoretical Simulation

Three main tasks have been identified in order to perform the theoretical simulation. Firstly, mathematical models representing each of the Windkessel models have been developed. Secondly, the necessary data for simulating the system have been obtained and the functions have been evaluated. Finally, the layout of the user interface that allows working with the models has been designed. Some design considerations for developing the theoretical simulation are:

- MATLAB has been selected as the tool for implementing and evaluating the models:
    - It is the language that is most familiar to the students at the UC3M, so using it seems appropriate in case further code manipulation is required.
    - In contrast to other languages such a C++ or Python, which are general purpose languages, MATLAB is a specialized language for technical and scientific computing.
    - It has an extensive library of predefined functions, so for many calculations there is no need of writing new subroutines or adding external libraries. This makes it extremely easy to work with mathematical functions.

- It allows to easily write programs and modify them within the built-in environments and integrated debugger.
- A lot of technical support is provided, both through the MATLAB environment and in forum discussions.
- It has its own graphical user interface development environment (GUIDE), which will be used to build the interface.

- To simulate the system an input signal is required. Also, an appropriate transfer function must be selected to obtain the desired output. Since the heart is a volumetric pump, flow has been selected as input and pressure as output. The transfer function will be defined differently for each of the Windkessel models used.

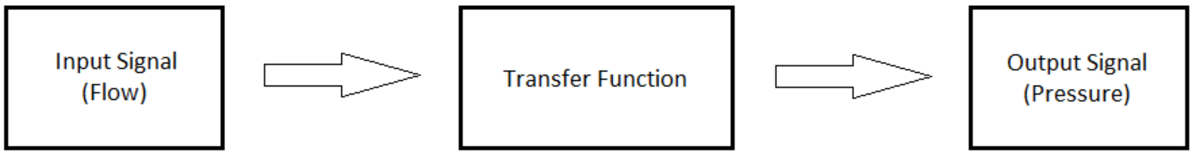


Figure 16: Generic structure of a system

- Electric analogy will be used for obtaining the equations governing each of the Windkessel configurations. Voltage and current will be labeled all the time as pressure and flow respectively for a better comprehension of the development.
- The pressure-flow relationships obtained are differential equations that need to be solved in order to plot the pressure as a function of the flow.
- In this project differential equations will be solved using a simple numerical method:
  - MATLAB, in order to plot a function, needs to assign values to each of its points in a discrete way. By using a numerical method, the function is directly discretized and implemented into MATLAB.
  - The simulated pressure signal may be discretized by expressing it as a function of a discrete input flow signal, the choice of this signal is discussed in section 4.1.2.
  - The transformations from continuous to discrete functions that will be used are:

$$f(t) \rightarrow f(i)$$

$$\frac{df(t)}{dt} \rightarrow \frac{f(i) - f(i-1)}{\Delta t}$$

$$\frac{d^2f(t)}{dt^2} \rightarrow \frac{f(i) - 2 \cdot f(i-1) + f(i-2)}{\Delta t^2}$$

Where i is the current sample and Δt is the time interval between two consecutive samples. Δt has been selected to be $0.001\ s$ so that samples are represented each $1\ ms$. This time interval seems ideal for the representation of both the flow and the pressure as these variables vary along the heart cycle and the heart cycle is not normally longer than $1\ s$.

Only main equations are included in this chapter, the full mathematical development can be checked at the annex.

4.1.1 Mathematical Models

Two-element Windkessel

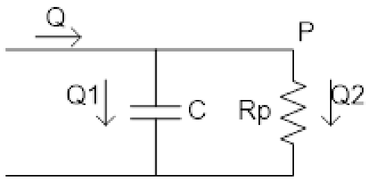


Figure 17: Two-element Windkessel diagram.

Flow may be expressed as a function of the pressure with the described equation:

$$Q(t) = C \cdot \frac{dP(t)}{dt} + \frac{P(t)}{R_p}$$

The equation may be transformed into Laplace domain, this definition will be used in section 5.1.3 to analyze the frequency behavior of the system.

$$Q = C \cdot s \cdot P + \frac{P}{R_p} = P\left(s \cdot C + \frac{1}{R_p}\right)$$

Now, solving for the pressure as a function of previous samples by using the method described in the previous section, the final solution is:

$$P(i) = \frac{1}{R_p + \frac{\Delta t}{C}} \cdot \left[Q(i) \cdot \frac{R_p \cdot \Delta t}{C} + P(i-1) \cdot R_p\right]$$

Realize that for evaluating this function, an input signal Q is needed. The obtained P signal will have the same number of points as Q. Also an initial condition $P(i_1)$ is required. The effect of choosing this initial condition is discussed in section 5.1.2.

Three-element Windkessel

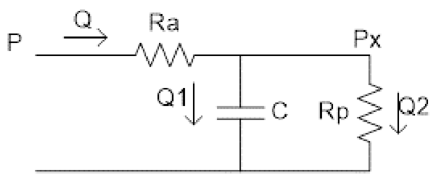


Figure 18:Three-element windkessel diagram.

The relationship between pressure and flow would be the following in temporal domain:

$$Q(t)\cdot\left(1+\frac{R_a}{R_p}\right)+C\cdot R_a\cdot\frac{dQ(t)}{dt}=C\cdot\frac{dP(t)}{dt}+\frac{P(t)}{R_p}$$

And in Laplace domain:

$$Q\cdot\left(1+\frac{R_a}{R_p}+C\cdot R_a\cdot s\right)=P\left(C\cdot s+\frac{1}{R_p}\right)$$

Again, Laplace definition will be used in the frequency behavior analysis. Finally, the current pressure sample is defined as:

$$P(i)=\frac{1}{1+\frac{C\cdot R_p}{\Delta t}}\cdot\left\{P(i-1)\cdot\frac{C\cdot R_p}{\Delta t}+R_p\cdot\left\{Q(i)\cdot\left[1+\frac{R_a}{R_p}+\frac{C\cdot R_a}{\Delta t}\right]-Q(i-1)\cdot\frac{C\cdot R_a}{\Delta t}\right\}\right\}$$

Four-element-series Windkessel

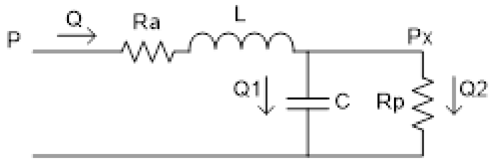


Figure 19: Four-element-series Windkessel diagram.

The solution in temporal domain:

$$Q(t)\cdot\left(1+\frac{R_a}{R_p}\right)+\frac{dQ(t)}{dt}\cdot\left(\frac{L}{R_p}-C\cdot R_a\right)+C\cdot L\cdot\frac{d^2Q(t)}{dt^2}=P(t)\cdot\left(\frac{1}{R_p}\right)+C\cdot\frac{dP(t)}{dt}$$

Solution in Laplace domain:

$$Q\cdot\left[\left(1+\frac{R_a}{R_p}\right)+s\cdot\left(\frac{L}{R_p}-C\cdot R_a\right)+C\cdot L\cdot s^2\right]=P\cdot\left(\frac{1}{R_p}+C\cdot s\right)$$

Finally:

$$P(i) = \frac{1}{\frac{1}{R_p} + \frac{C}{\Delta t}} \cdot \left\{ P(i-1) \cdot \frac{C}{\Delta t} + Q(i) \cdot \left[\left(1 + \frac{R_a}{R_p}\right) + \frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{C \cdot L}{\Delta t^2} \right] - Q(i-1) \cdot \left[\frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{2 \cdot C \cdot L}{\Delta t^2} \right] + Q(i-2) \cdot \frac{C \cdot L}{\Delta t^2} \right\}$$

Four-element-parallel Windkessel

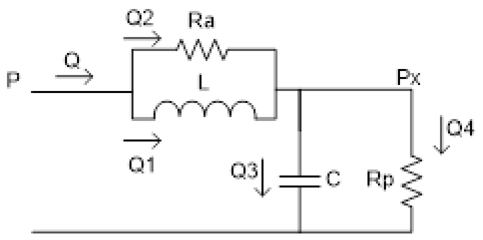


Figure 20:Four-element-parallel Windkessel diagram.

Due to the complexity of this circuit it was solved in the Laplace domain:

$$Q(s) \cdot \left[s^2 \cdot L + s \cdot \left(\frac{L}{C \cdot R_a} + \frac{L}{C \cdot R_p} \right) + \frac{1}{C} \right] \cdot C \cdot R_p \cdot R_a = \left(s \cdot C \cdot R_p \cdot R_a + R_a + L \cdot s^2 \cdot C \cdot R_p + s \cdot L \right) \cdot P(s)$$

And later transformed into temporal domain applying the transformation: $s^n = \frac{d^n}{dt^n}$

$$\frac{d^2 Q(t)}{dt^2} \cdot L \cdot C \cdot R_p \cdot R_a + \frac{dQ(t)}{dt} \cdot \left(L \cdot (R_p + R_a) \right) + Q(t) \cdot R_p \cdot R_a = \frac{d^2 P(t)}{dt^2} \cdot L \cdot C \cdot R_p + \frac{dP(t)}{dt} \cdot \left(C \cdot R_p \cdot R_a + L \right) + P(t) \cdot R_a$$

The final solution is:

$$P(i) = \frac{1}{\frac{L \cdot C \cdot R_p}{\Delta t^2} + \frac{C \cdot R_p \cdot R_a + L}{\Delta t} + R_a} \cdot \left\{ P(i-1) \cdot \left[\frac{2 \cdot L \cdot C \cdot R_p}{\Delta t^2} + \frac{C \cdot R_p \cdot R_a + L}{\Delta t} \right] - P(i-2) \cdot \frac{L \cdot C \cdot R_p}{\Delta t^2} + Q(i) \cdot \left[\frac{L \cdot C \cdot R_p \cdot R_a}{\Delta t^2} + \frac{L \cdot (R_p + R_a)}{\Delta t} + R_p \cdot R_a \right] - Q(i-1) \cdot \left[\frac{2 \cdot L \cdot C \cdot R_p \cdot R_a}{\Delta t^2} + \frac{L \cdot (R_p + R_a)}{\Delta t} \right] + Q(i-2) \cdot \frac{L \cdot C \cdot R_p \cdot R_a}{\Delta t^2} \right\}$$

Realize that for the four-element model (both series and parallel configuration) two pressure initial conditions $P(i_1)$ and $P(i_2)$ are required.

4.1.2. Simulation Data

Input signal: Flow

The input signal used for the model is the flow of blood through the aorta. Recall that the heart behaves as a flow pump. If the entire systemic arterial system (arteries, arterioles and capillaries) was considered as simply a resistor (R), opposing to the flow of blood, the relationship between flow and pressure could be defined by the following configuration of Ohm's law.

$$P = Q \cdot R$$

Where the pressure in the aorta is the dependent variable, implying that the heart would be modeled as a current source instead of as a voltage source. The opposite case would be the one represented below. This relationship may be used for example when studying the inner vascular bed of an organ, where pressure depends on external factors and so it is fixed. By adjusting resistance (i.e. contracting arterioles) flow would vary.

$$Q = \frac{P}{R}$$

In order to simulate the flow waveform, it must be defined for the two different periods of the heart cycle. During systole, aortic valve will be opened and therefore there will be flow. During diastole, aortic valve is closed, and so there will not be any flow. No negative flow can be allowed, and so signal during systole may be approximated as a squared sinusoidal.

The definition of the flow used [38] is described below:

$$Q(t) = \begin{cases} Q_0 \cdot \left(\sin\left(\frac{\pi \cdot t}{T_S}\right)\right)^2 & t \in (0, T_S) \\ 0 & t \in (T_S, T) \end{cases}$$

Where T_S is the time in systole, and T is the heart period. The time in diastole can be therefore defined as $T_D = T - T_S$. The maximum flow reached in one period is Q_0, so it can be identified as the maximum flow during one heart period.

Regular parameters for healthy adults can be observed in Table 1.

Parameter	Normal Units
$Heart\ Rate\ (HR)$	$72\ \frac{beats}{min}$
$Heart\ Period\ (T)$	$0.8\hat{3}\ \frac{s}{cycle}$
$Time\ in\ Systole\ (T_S)$	$0.33\ s$
$Time\ in\ Diastole\ (T_{D})$	$0.5\ s$
$Maximum\ Flow\ (Q_0)$	$500\,\frac{mL}{beat\cdot s}$
$Cardiac\ Output\ (CO)$	$5.9\,\frac{L}{min}$
$Stroke\ Volume\ (SV)$	$82\ \frac{mL}{beat}$

Table 1: Normal hemodynamic parameters for a healthy adult [39].

The systole duration for a healthy subject has been obtained considering the following duty cycle: $T_S = \frac{2}{5} \cdot T$. The maximum flow ejected to the aorta during each beat [lavho] may be related to the cardiac output, heart rate, time in systole and stroke volume as shown below.

$$Q_0 = \frac{2 \cdot CO}{HR \cdot T_S} = \frac{2 \cdot SV}{T_S}$$

The previous relationship arises from the fact that the stroke volume may be calculated as the area under the flow waveform. Realize simplification is possible since $CO = SV \cdot HR$.

Code in MATLAB to generate the flow is specified in the annex. The general idea is to discretize the given flow definition, evaluating the flow at different times separated by the time interval Δt. The total number of samples i_N for which the flow is to be evaluated depends on the heart period and number of heart cycles represented. As $\Delta t = 0.001s$ for a period $T = 0.833s$ the number of samples represented (and so the length of the flow vector) will be of 833. If ten heart cycles were to be represented the final vector would have a total length of 8330 samples. Fig. 21 shows the obtained flow waveform for a single heart cycle.

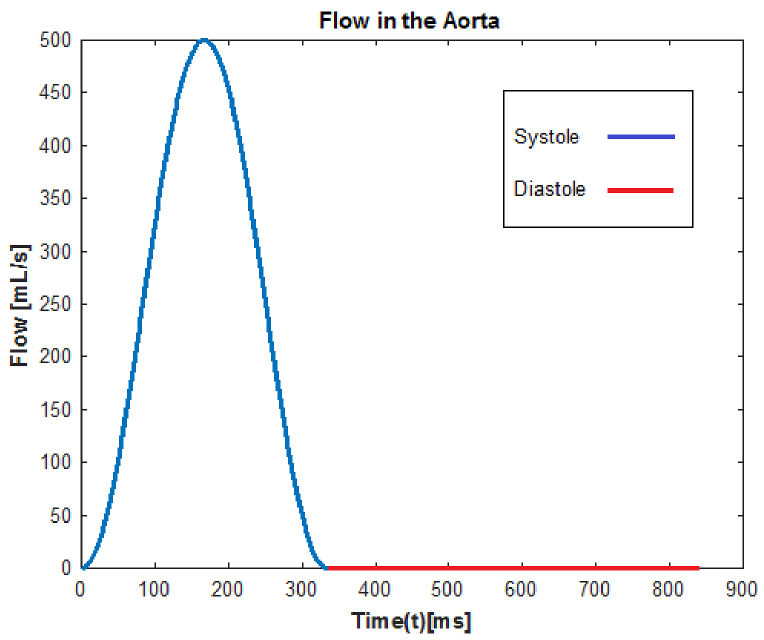


Figure 21: Flow over a heart cycle. Obtained using MATLAB.

Transfer function

As previously mentioned the transfer function is defined differently for each of the models used. In this way, the circuit configuration and the number of elements used in a model will determine its transfer function. As the input flow is considered to be the same for all models, the transfer function is also responsible for giving an output signal, different for each model.

In order to determine the transfer functions, two steps are required:

- To develop the circuit equations, which was done in section 4.1.1.
- To find the values for the different parameters: peripheral resistance, compliance, characteristic impedance and inertance (Table 2).

Parameter	**SI units**	**Medical Units**
Peripheral Resistance	$133.28 \cdot 10^6 \ \frac{kg}{s \cdot m^4}$	$16.66 \ \frac{mmHg \cdot min}{L}$
Aortic Compliance	$0.75 \cdot 10^{-8} \ \frac{s^2 \cdot m^4}{kg}$	$1 \ \frac{mL}{mmHg}$
Characteristic Impedance	$6.66 \cdot 10^6 \ \frac{kg}{s \cdot m^4}$	$0.83 \ \frac{mmHg \cdot min}{L}$
Inertance	$6.66 \cdot 10^5 \ \frac{kg}{m^4}$	$5 \ \frac{mmHg \cdot s^2}{L}$

Table 2: Windkessel parameter values for a healthy adult human [38].

Aortic impedance has the same units as resistance, this comes from the definition:

$$Z_a = \frac{v\rho}{A}$$

$$[Z_a] = \left[\frac{v\rho}{A}\right] = \left[\frac{velocity \cdot density}{area}\right] = \left[\frac{m}{s} \cdot \frac{kg}{m^3} \cdot \frac{1}{m^2}\right] = \left[\frac{kg}{s \cdot m^4}\right]$$

However, the characteristic impedance is not exactly a resistance and its role needs to be understood in terms of oscillatory phenomena. This concept is related to the question: why is it needed to add complexity to the models? Why is it not enough to work with the two-element model? There are two answers to this question:

1. Firstly, more complexity in the model means more physiological features are being represented.
2. Secondly, distinct models behave differently at distinct frequencies.

Therefore, in order to determine if the obtained transfer functions are suitable the behavior of each of the systems must be analyzed to make sure its performance is correct. This analysis is provided in chapter five, in the results section.

Output signal: Pressure

Once the transfer function is determined and the values for each of the parameters are defined, the next step is to evaluate the different transfer functions to obtain the aortic pressure. An example for the simplest case (2-element Windkessel) is developed below, and the exact same procedure is followed for higher complexity models.

Starting with the following equation:

$$P(i) = \frac{1}{R_p + \frac{\Delta t}{C}} \cdot \left[Q(i) \cdot \frac{R_p \cdot \Delta t}{C} + P(i-1) \cdot R_p\right]$$

Recall that the reference values for the required parameters are $R_p = 1\ \frac{mmHg \cdot s}{mL}$, $C = 1\ \frac{mL}{mmHg}$ and $\Delta t = 0.001s$. Considering $P(i_1) = 0\ mmHg$.

$$P(i_2) = \frac{1}{1\,\dfrac{mmHg\cdot s}{mL} + \dfrac{0.001s}{1\,\dfrac{mL}{mmHg}}} \cdot \left[Q(i_2) \cdot \frac{1\,\dfrac{mmHg\cdot s}{mL}\cdot 0.001s}{1\,\dfrac{mL}{mmHg}} + 0\cdot 1\,\frac{mmHg\cdot s}{mL} \right]$$

$$= \frac{1}{1\,\dfrac{mmHg\cdot s}{mL} + 0.001\,\dfrac{s\cdot mmHg}{mL}} \cdot \left[Q(i_2)\cdot 0.001\frac{mmHg^2 s^2}{mL^2} + 0 \right]$$

$$= \frac{1}{1.001\,\dfrac{mmHg\cdot s}{mL}} \cdot \left[Q(i_2)\cdot 0.001\frac{mmHg^2 s^2}{mL^2} \right] = \frac{0.001}{1.001}\cdot Q(i_2)\frac{mmHg\cdot s}{mL}$$

As flow units are $\frac{mL}{s}$, final units will be $mmHg$, so pressure is being evaluated correctly. This process will be automatically performed by MATLAB until $i = i_N$.

The resulting pressure waveform can be observed in Fig. 22.

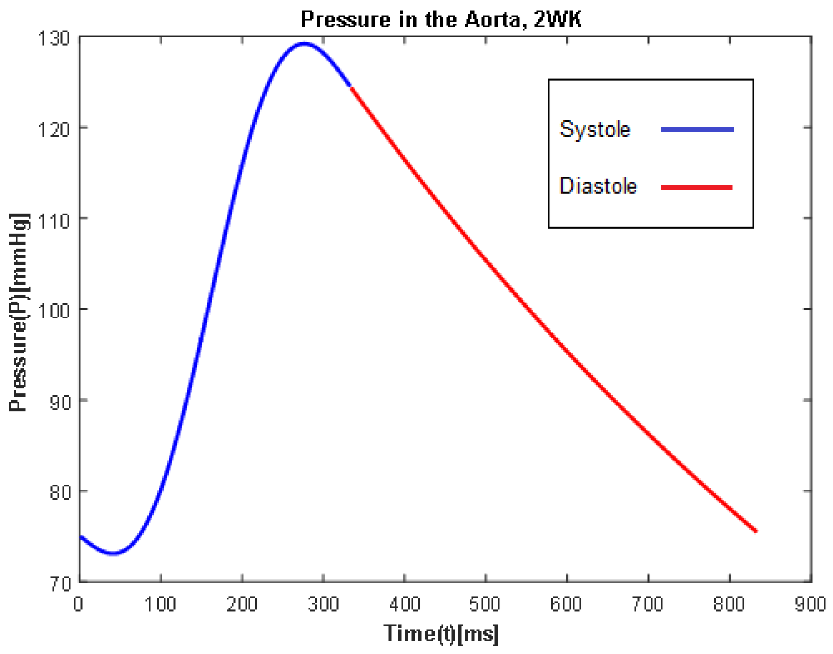


Figure 22: Pressure in the Aorta obtained using the two-element Windkessel model. Obtained using MATLAB.

4.1.3. Simulation Interface

A simulation interface has been created using GUI MATLAB. It is structured into four main different windows. The first one is the main welcome page, which allows the user to select between working with the two-element, three-element and four-element-parallel Windkessel Each of them allows displaying three different graph types: pressure, flow and pressure-volume loop. The four-element series Windkessel has not been included in the interface. Although the output pressure obtained by the model was not far from the expected, the performance of the model was observed to be unstable at high frequencies (see section 5.1.3).

Although the system is not intended to work under those conditions, it seemed enough to show the students only one model configuration with four elements, so the more stable 4-element parallel Windkessel was chosen.

Some examples of the layout of the interface are presented in this section. Fig. 23 shows the main welcome window, where the user can select between the different Windkessel model options. Fig. 24 shows the window for the four-element-parallel Windkessel, when the user has selected to display the pressure. Fig. 25 shows the same window but it displays flow. Finally, Fig. 26 shows the simulated pressure-volume loop for the same window.

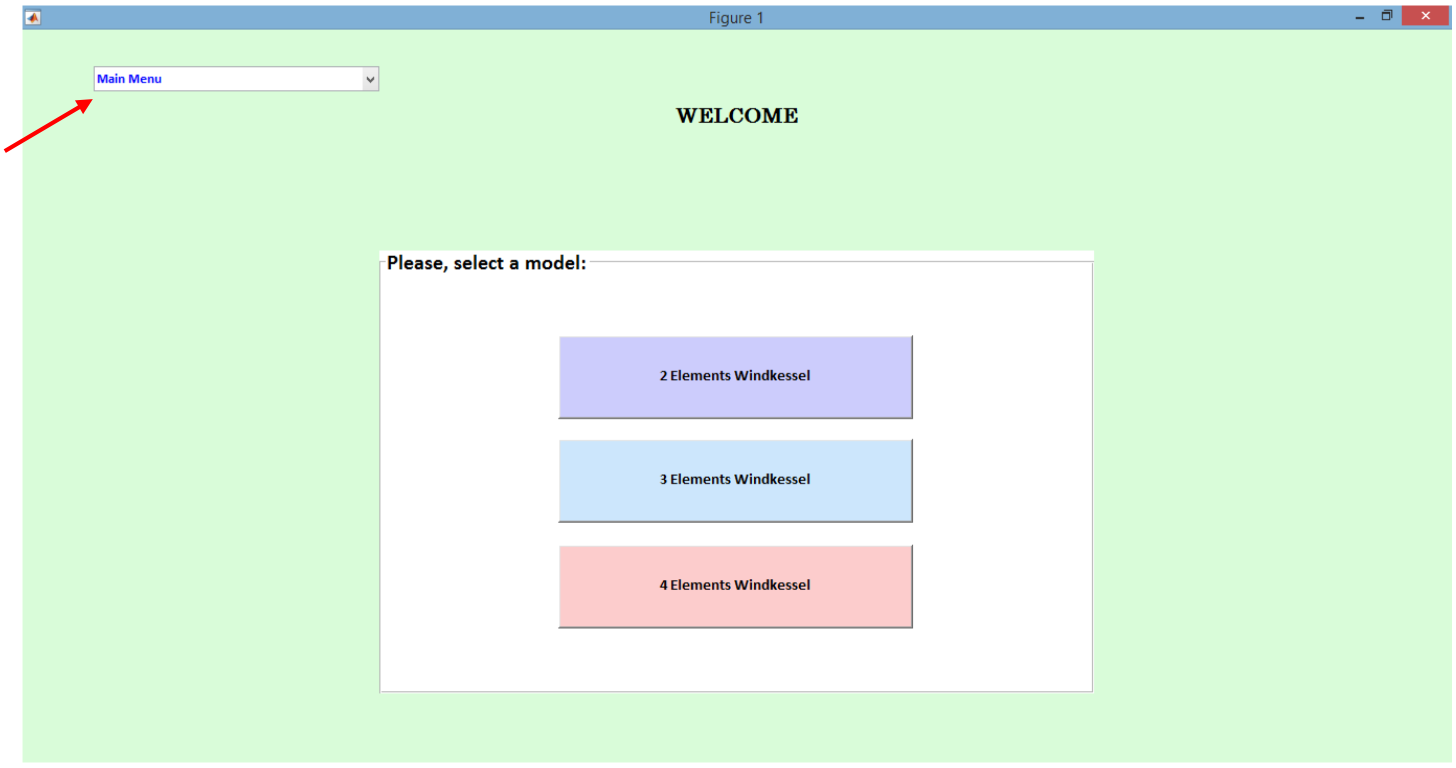
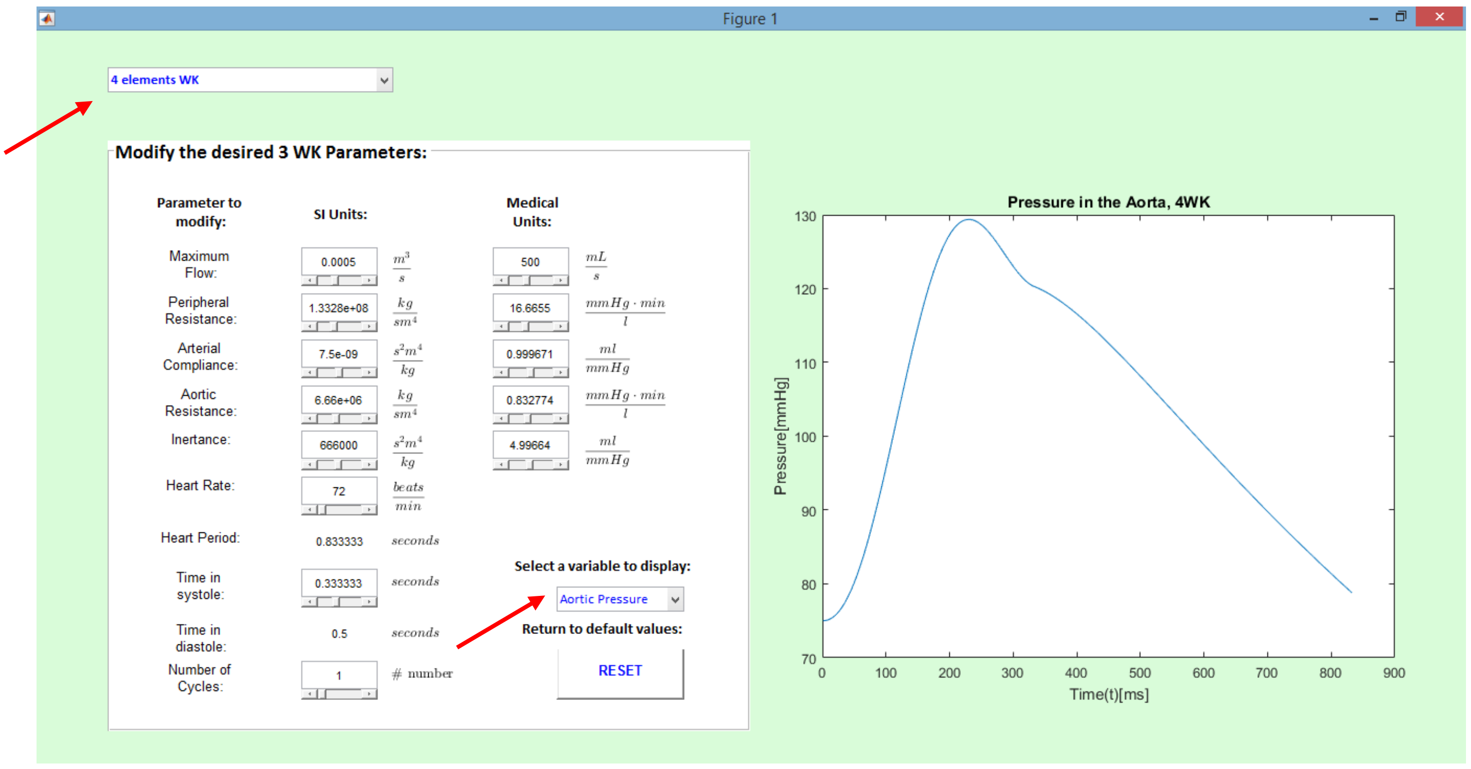


Figure 23: Simulation interface: welcome page



4 elements WK

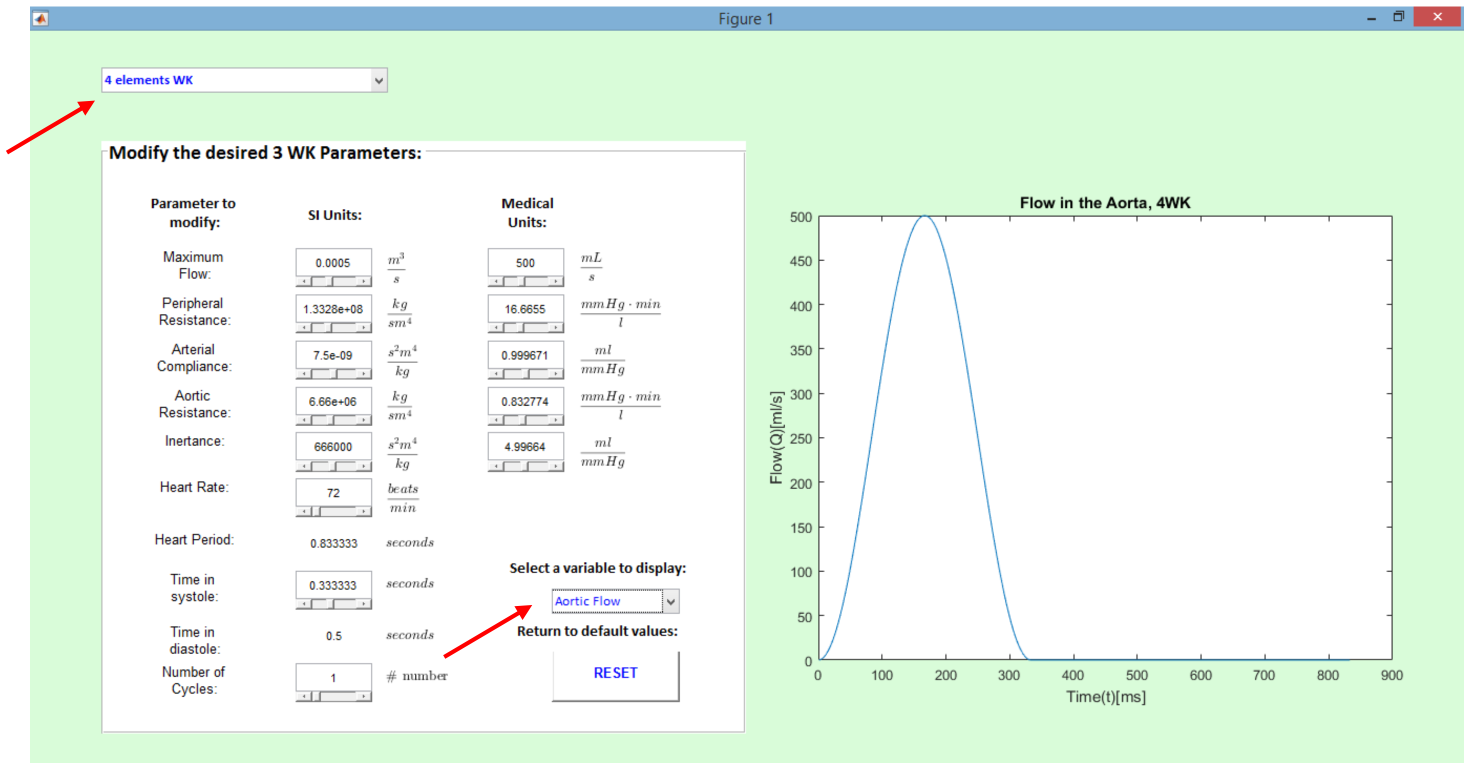
Modify the desired 3 WK Parameters:

Parameter to modify:	SI Units:		Medical Units:	
Maximum Flow:	0.0005	$\frac{m^3}{s}$	500	$\frac{mL}{s}$
Peripheral Resistance:	1.3328e+08	$\frac{kg}{sm^4}$	16.6655	$\frac{mmHg \cdot min}{l}$
Arterial Compliance:	7.5e-09	$\frac{s^2m^4}{kg}$	0.999671	$\frac{ml}{mmHg}$
Aortic Resistance:	6.66e+06	$\frac{kg}{sm^4}$	0.832774	$\frac{mmHg \cdot min}{l}$
Inertance:	666000	$\frac{s^2m^4}{kg}$	4.99664	$\frac{ml}{mmHg}$
Heart Rate:	72	$\frac{beats}{min}$		
Heart Period:	0.833333	$seconds$		
Time in systole:	0.333333	$seconds$		
Time in diastole:	0.5	$seconds$		
Number of Cycles:	1	$\#$ number		

Select a variable to display: Aortic Pressure

Return to default values: RESET

Figure 25: Simulation Interface: four-element-parallel model, pressure display.



4 elements WK

Modify the desired 3 WK Parameters:

Parameter to modify:	SI Units:		Medical Units:	
Maximum Flow:	0.0005	$\frac{m^3}{s}$	500	$\frac{mL}{s}$
Peripheral Resistance:	1.3328e+08	$\frac{kg}{sm^4}$	16.6655	$\frac{mmHg \cdot min}{l}$
Arterial Compliance:	7.5e-09	$\frac{s^2m^4}{kg}$	0.999671	$\frac{ml}{mmHg}$
Aortic Resistance:	6.66e+06	$\frac{kg}{sm^4}$	0.832774	$\frac{mmHg \cdot min}{l}$
Inertance:	666000	$\frac{s^2m^4}{kg}$	4.99664	$\frac{ml}{mmHg}$
Heart Rate:	72	$\frac{beats}{min}$		
Heart Period:	0.833333	$seconds$		
Time in systole:	0.333333	$seconds$		
Time in diastole:	0.5	$seconds$		
Number of Cycles:	1	$\#$ number		

Select a variable to display: Aortic Flow

Return to default values: RESET

Figure 245: Simulation Interface: four-element-parallel model, flow display.

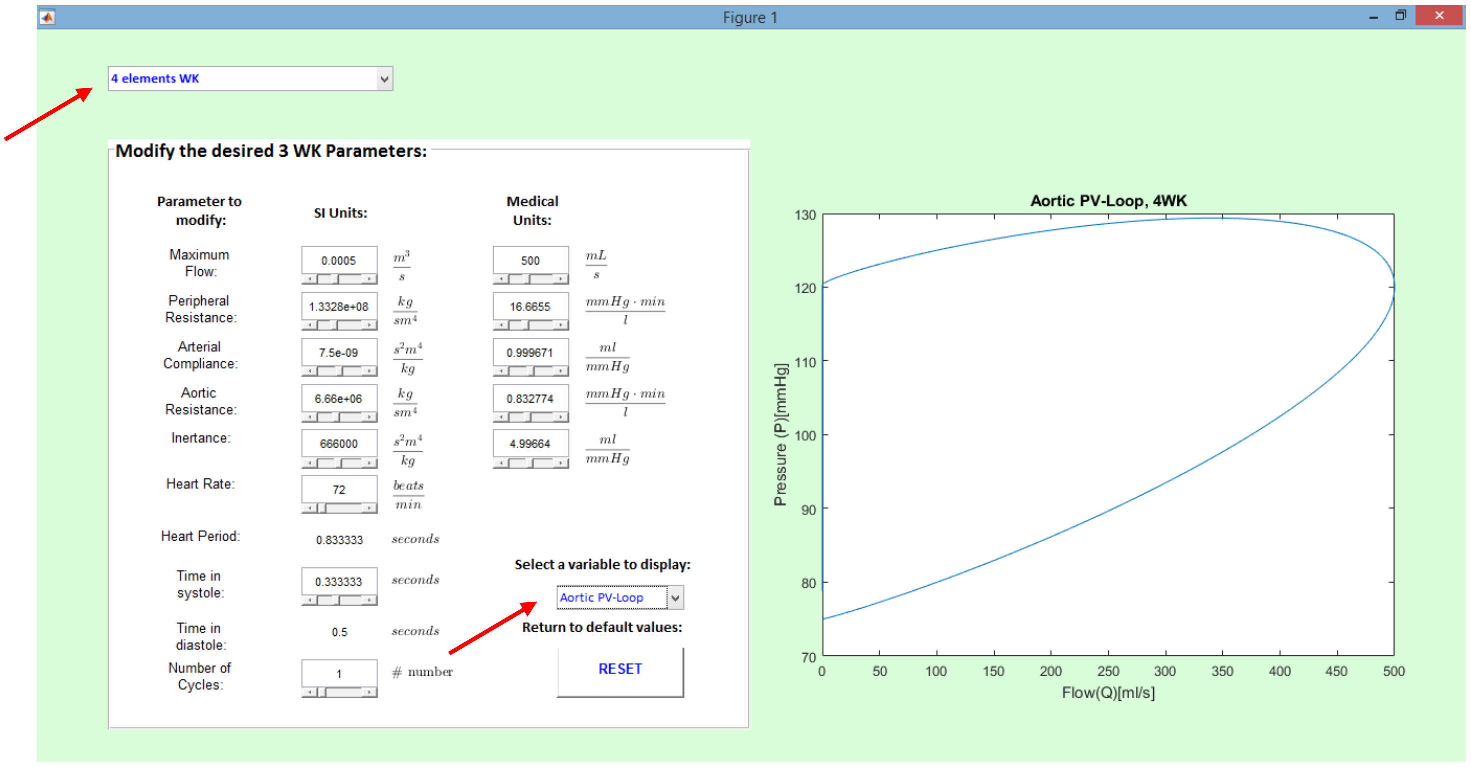


Figure 1

4 elements WK

Modify the desired 3 WK Parameters:

Parameter to modify:	SI Units:		Medical Units:	
Maximum Flow:	0.0005	$\frac{m^3}{s}$	500	$\frac{mL}{s}$
Peripheral Resistance:	1.3328e+08	$\frac{kg}{sm^4}$	16.6655	$\frac{mmHg \cdot min}{l}$
Arterial Compliance:	7.5e-09	$\frac{s^2m^4}{kg}$	0.999671	$\frac{ml}{mmHg}$
Aortic Resistance:	6.66e+06	$\frac{kg}{sm^4}$	0.832774	$\frac{mmHg \cdot min}{l}$
Inertance:	666000	$\frac{s^2m^4}{kg}$	4.99664	$\frac{ml}{mmHg}$
Heart Rate:	72	$\frac{beats}{min}$		
Heart Period:	0.833333	$seconds$		
Time in systole:	0.333333	$seconds$		
Time in diastole:	0.5	$seconds$		
Number of Cycles:	1	$\#$ number		

Select a variable to display: Aortic PV-Loop

Return to default values: RESET

Figure 26: Simulation Interface: four-element-parallel model, pressure-volume loop display.

4.2. Physical Device

The physical device is a prototype manufactured by the company SEDECAL that was acquired by the university a few years ago. The device represents the arterial systemic tree by means of several elements that will be described in the following section, each in charge of a specific function. Regarding this physical device, work has been developed in three main aspects: improving the electronics, establishing communication with the computer (both in the device-computer and computer-devices directions) and developing the necessary software to interpret acquired pressure and flow data.

4.2.1. Description of the device

The device is a hydraulic-pneumatic device, which includes electric and electronic components for control and data acquisition. Fig. 27 shows the main diagram of how the different levels of the device interact together. Detailed schemes for each of the levels are provided in the annex.

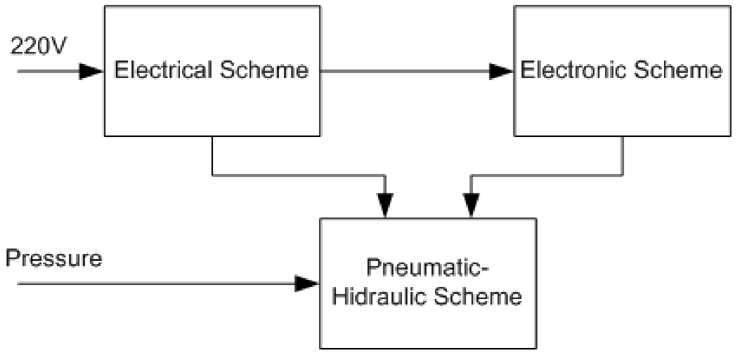


Figure 26: Main diagram of the physical device. Scheme developed using Microsoft Office VISIO.

Regarding the main parts forming the device, they are highlighted in Fig. 28. Each of the parts plays an important role, and the ensemble allows the device to behave as a physical model of the arterial system.

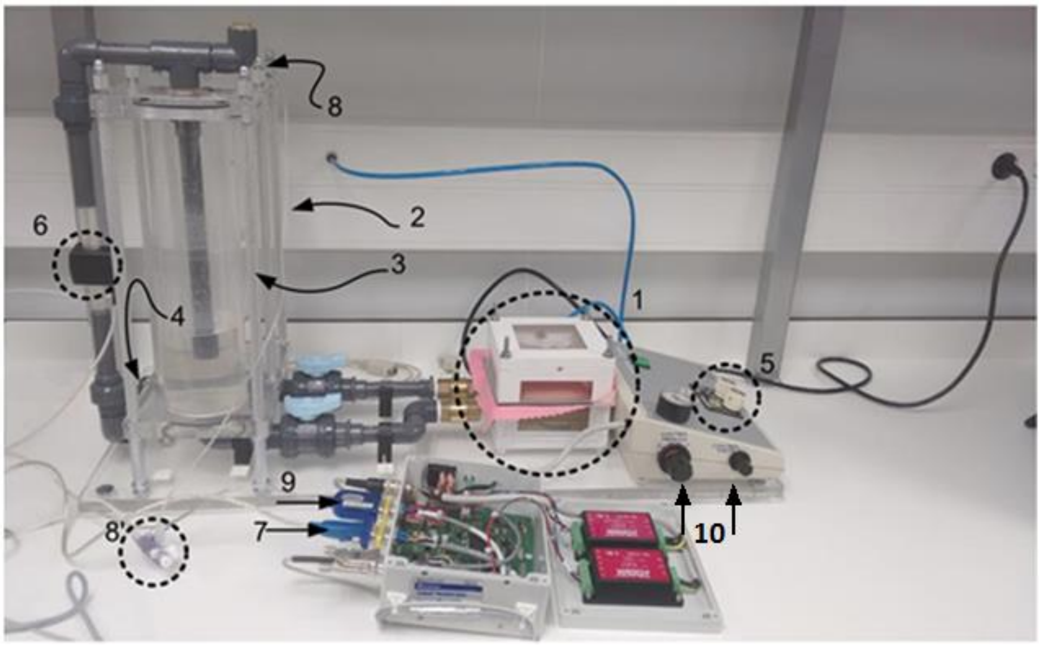


Figure 27: Picture of the physical device with the main components highlighted.

1.-Membrane pump simulating the heart.

2.-Compliant element representing the elastic large arteries.

3.-Fluid Reservoir.

4.-Peripheral resistance modeling smaller vessels.

5.-Heart cycle controls, allowing to manually modify the heart rate and duty cycle.

6.-Flow transducer, recording flow data just after the peripheral resistance.

7.-Flow port, sending data to the computer.

8.-Pressure transducer (8' is still part of the pressure transducer), recording simulated aortic pressure.

9.-Pressure port, sending data to the computer.

10.-Pressure-vacuum regulators (rotatory knobs).

Heart cycle controls and pressure-vacuum rotatory knobs are located at the control case, which serves as the link between the electronic and the pneumatic levels of the device.

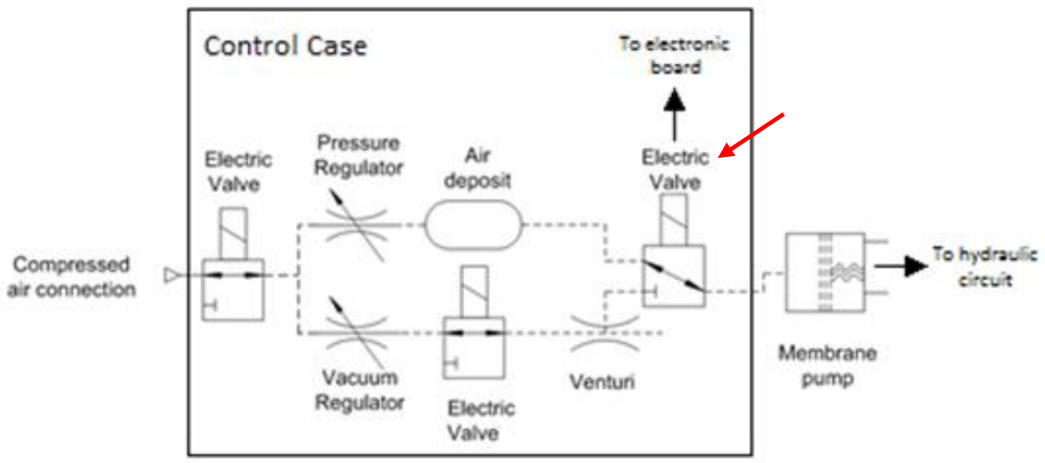


Figure 28: Pneumatic components inside the Control Case, and links to other levels of the device. Scheme developed using Microsoft Office VISIO.

In the electronic board a timing circuit, implemented with a LM 555 timer, is in charge of controlling the action of the main electric valves that induce the heart to pump. The timing of this circuit is can be modified by the heart cycle controls, which are basically two potentiometers.

The "strength" of the membrane pump depends on the pressure-vacuum regulators, which allow the flow of a determined amount of compressed air. At times specified by systole, the main electric valve is activated introducing form the air deposit the selected amount of air inside the heart chamber. This displaces the membrane downwards, which pumps water into the hydraulic circuit. At times specified as diastole, the main electric valve is no longer connected to the air deposit. It therefore helps removing the selected amount of air from the heart chamber. This causes water to flow into the heart chamber. In this way, it can be observed that the control the user has over the heart refers to the modification of its cardiac output. Aortic pressure is therefore not a parameter that the user can modify, and so, when talking about changing the pressure input of the system it will refer to changing the amount of compressed air supplied to the membrane.

4.2.2. Electronic Improvements

The existing electronics have been improved. The new design achieves the goal of allowing the user to control heart rate parameters and duty cycle from software, while keeping the manual control in case the user decides to use this one. Some problems were found in the existing board: the circuits used were not well documented, the voltage regulator was overheating and several connections were not properly secured. To solve this, the entire circuit was replaced by a new board.

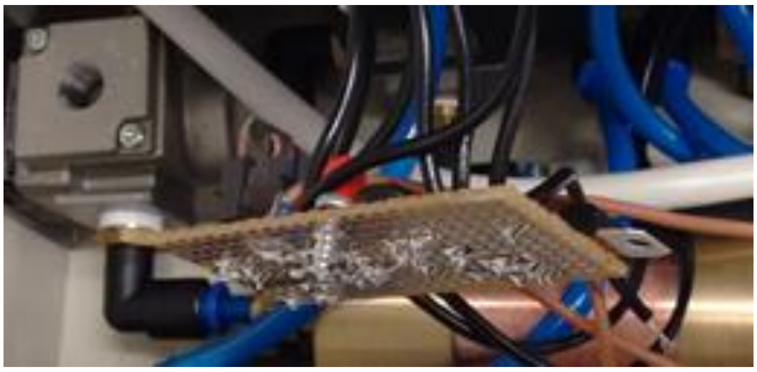


Figure 29: Original electronic board.

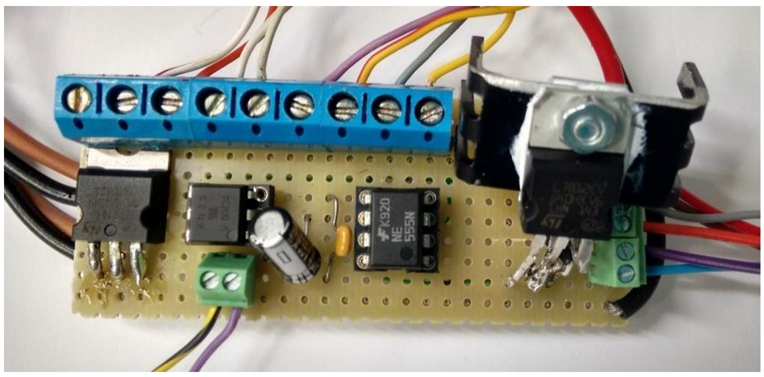


Figure 30: New electronic board.

Some advances have been included in the manufactured electronic board. In the first place, two voltage regulators have been included so that voltage decreases in two steps, first from $24\ to\ 12\ V$ and then from $12\ to\ 5\ V$, which is the level required for all the components. A heat sink and thermal conductance paste, have also been added to help dissipating the heat. This will help components to work more efficiently, avoiding overheating and lasting longer. In the second place, connection terminals have been included for all the cables communicating the circuit at the board with outside components such as the electric valves. This allows easier identification and access to the different components. Besides, this will ease further improvements as it is simpler to replace and modify components in a modular board. All connections with other elements, such as RS-232 pins, have been secured with heat shrinking tube. In this way no connections will be removed, or touched by accident, and the device will be safer. Moreover, the provided documentation is now correct and agrees with reality, so future maintenance will be much easier.

Regarding the addition of the timing control through software, extra components have been added to the board. The added elements are have been located in two different regions, one part has been placed with the main board at the control case and the other at the transducer case. The general idea is to make use of the voltage provided by one of the RS-232 pins, which can be set to a HIGH or LOW state during a specified time from the computer, to substitute the signal originally supplied by the LM 555 timer. A switch to select between supplying the signal with the timer or by using the computer has been included.

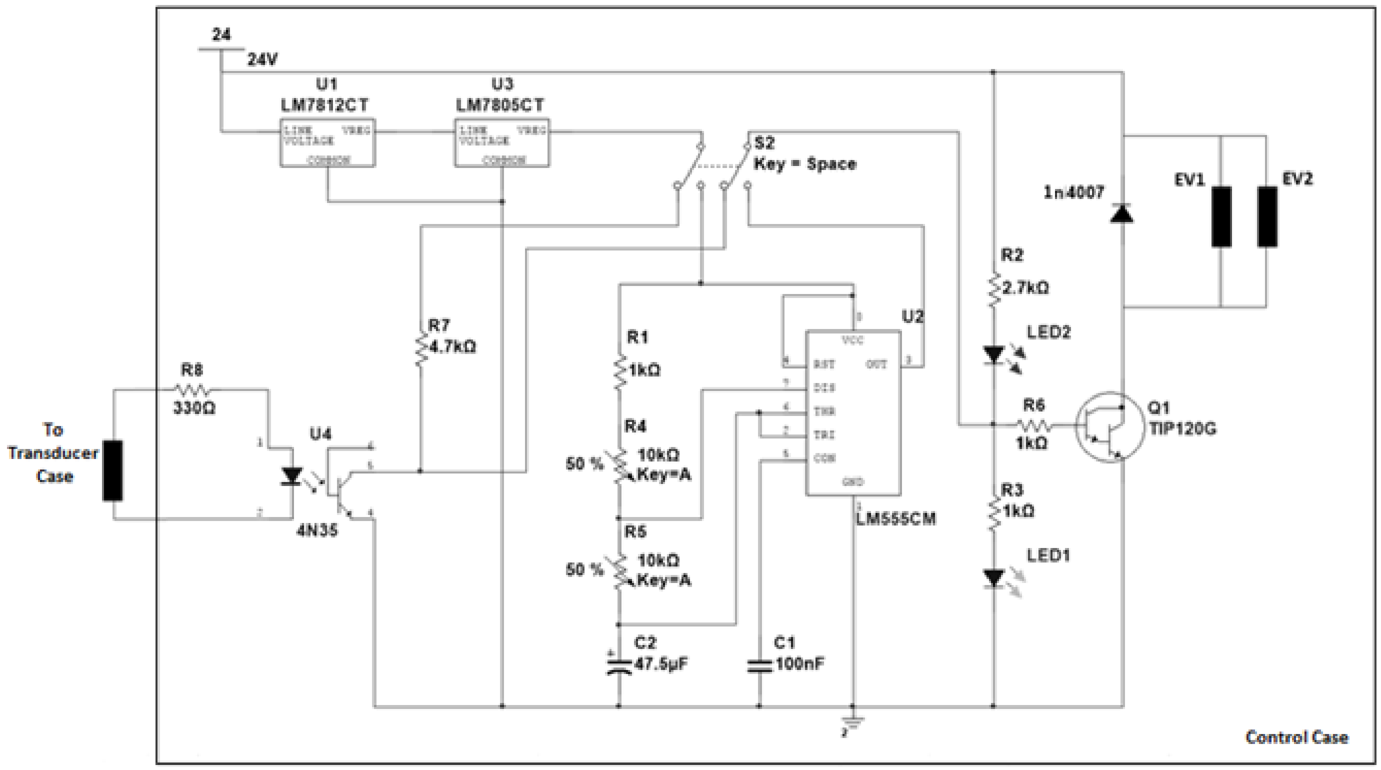


Figure 31: New electronic circuit inside Control Case. It includes a switch that allows selecting between the signal generated by the LM 555 or the signal coming from the optoisolator (which is transmitted from the circuit inside the Transducer Case). Final signal is in charge of activating the electric valves). Circuit designed using MULTISIM.

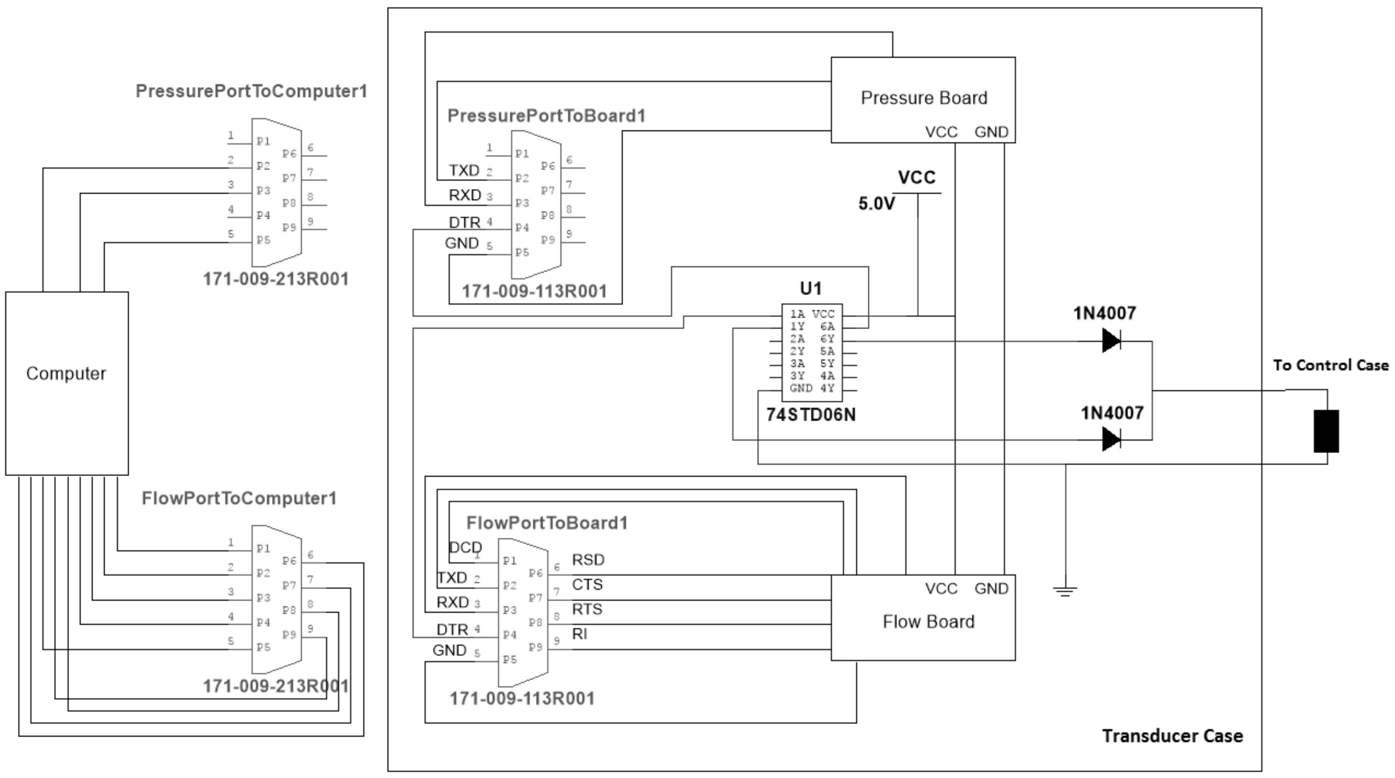
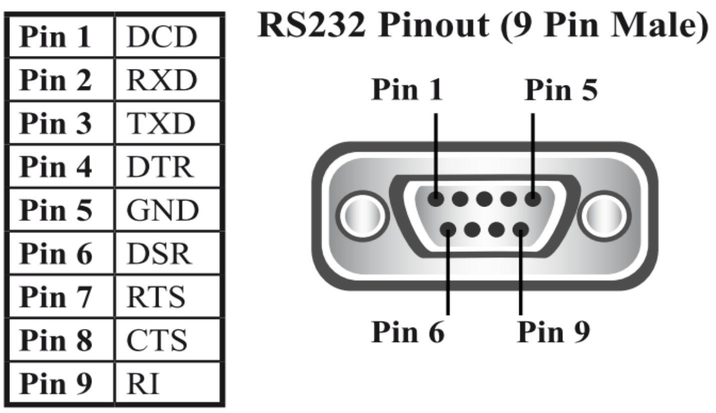


Figure 32: New electronic circuit inside Transducer Case. A signal obtained from the RS-232 interfacing the computer is sent to the Control Case. Voltage is obtained from the DTR pins of the RS-232 (see section 4.2.3), transducer not used for reading at the moment is set to periodic HIGH and LOW states. After inverting it (to avoid the constant high value of the pin belonging to the reading transducer) the resulting signal is sent to activate the electric valves.

4.2.3. Device to computer communication

As previously mentioned the communication between the device and the computer is established through serial protocol as the two transducer used are available for RS-232 interfacing. A RS-232-to-USB converter cable is used since computer has no RS-232 ports. This communication protocol has determined the way of sending pulses from the computer to the device to control heart cycle parameters. Other design options such as using an *Arduino* microcontroller were considered. Both pulses through the RS-232 and *Arduino* can be controlled using MATLAB (to have everything in the same programming environment). However, the first option was selected as the original design wanted to be preserved, and the microcontroller would require from another USB port connection.

The pin that is being used to replace the pulses created by the timer is the Data Transmission Ready pin (DTR). MATLAB allows to set this pin to a HIGH or LOW state by means of a specific function



Pin 1	DCD
Pin 2	RXD
Pin 3	TXD
Pin 4	DTR
Pin 5	GND
Pin 6	DSR
Pin 7	RTS
Pin 8	CTS
Pin 9	RI

Figure 33: RS-232 pin diagram [40].

The two available transducers communicate in a different way. The pressure transducer uses software handshaking, which means it only requires three of the nine pins of the RS-232 to establish serial communication (RXD, TXD and GND, plus V_{cc}). The flow transducer uses hardware handshaking, for which all the nine pins are required. These transducers were manufactured and distributed by different companies, they acquire and send data in a completely different way. For this reason, in order to decode and understand the information sent by each transducer each of them needs to be studied as a separate case.

Pressure transducer:

The board used is the MEDLAB EG 02000 [41]. This module works with all sensors that offer a sensitivity of$5\mathrm{\mu V/V/mmHg}$. In order to read using this transducer, the serial port will have to be defined as 9600 baud rate, 8 data bits, one stop bit, and no parity, to meet with the specifications. The transducer does not transmit 16 bit pressures values, as it is not considered to be economical. This impedes the device from sending negative pressure values. To fix this, since only pressure values in the interval from -99 to 300 mmHg have to be transmitted, all values are transmitted with an offset of +100 mmHg. This must be considered by the receiver side, and a value of 100 (decimal) must be subtracted before any display or storage of the data.

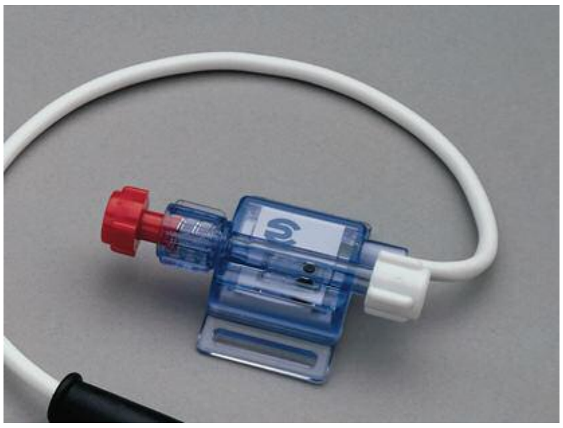


Figure 34: Pressure sensor compatible with the EG 02000 [42].

This transducer sends three types of regular data packets: waveform packets, status packets and value packets. For displaying and storing the pressure waveforms, only the waveform packet is necessary. However, other packets will affect to the filtering process.

- Waveform packets: contains three bytes.
    - The first byte is the waveform-packet-characteristic-byte which serves as identification (ID) that a waveform packet is being transmitted. It is recognized as its most significant bits are always "1100"
    - Second byte corresponds to the waveform recorder by channel one
    - Third byte corresponds to the waveform recorder by channel two

  As only channel one of the two available will be used, during the filtering process it must be considered that only the middle value of a waveform packet is a relevant value for the application.

- Status packet, which also contains three bytes. An ID byte (whose most significant bits are "1101"), and two bytes reporting the status of the transducer (e.g. if it is connected), one relative to channel one and another to channel two.
- Information packet, which contains nine bytes. An ID byte (which starts by "10"), four bytes with information relative to channel one (systolic pressure, mean arterial and diastolic pressure) and four relative to channel two.

Realize that in the decoding process a single value must be selected between at least three of the bytes transmitted. In order to be used, this transducer has been calibrated, and in the software an offset of $70\ mmHg$ is being added.

Flow transducer:

Both the board and the sensor used have been manufactured by EMTEC [43]. In order to read using this transducer, the serial port will have to be defined as 38400 baud rate, 8 data bits, one stop bit, and no parity, to meet with the specifications. This transducer works by using the ultrasonic transit-time method to measure volumetric flow through the tube as described in Fig. 36.

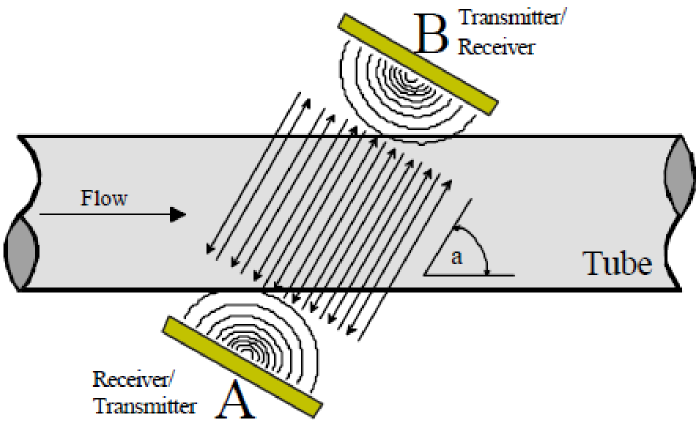


Figure 35: Ultrasonic transit-time method to measure volumetric flow [43].

Considering the values sent by the flow transducer, the filtering process will be simpler. This transducer sends a string containing different parameters each time the receiver side requests it. An example can be observed in Table 3.

Parameter	1	2	3	4	5	6	7	8	9
String	00	08	+1592	+1590	+113	+0	+1254	+1590	+454

Table 3: Data format sent by flow transducer [43].

The only parameter of interest will be the fourth one as it represents the actual flow value in $\frac{ml}{min}$. The other set of parameters represent features such as additional information for calibration purposes, status information of the flow measurement board or signal amplitude of the acoustic receiving signal. In this way, for the reading and decoding process it must be considered that the data arriving is a string, so the positions of interest within the string must be selected before converting it to a number to avoid errors.

Realize that one of the software requirements was that units of variables acquired from the device had to agree with data displayed in the theoretical interface. Pressure is being sent by default in $mmHg$, but before displaying flow data it needs to be transformed from $\frac{ml}{min}$ to $\frac{ml}{s}$, which is being done by simply multiplying by a conversion factor.

4.2.4. Reading Codes and Interface

As a strict control of the reading, decoding and sending functions must be maintained, the code developed has been structured by using timers. Timers are sections of code that are meant to happen periodically with a determinable interval. Each time their execution time arrives, they run a callback function that is defined by the programmer.

The structure of the code allows the user deciding between four different options: read pressure or flow while controlling timing through software and read pressure or flow while manually controlling timing. Once one of the four options is selected a specific code for each section is executed. This code can be described by looking at the different sections:

Manual control:

- Pressure reading: it uses one timer for reading, which executes each 0.001s. Acquired data is stored in a vector that is always available for the rest of the code. Decoding and painting the waveform is constantly being performed in the main function by means of a while loop.
- Flow reading: again one timer for reading each 0.001s. However decoding is executed also using a timer that executes each 0.1s. As the decoding step was taking more time it was necessary to specify when to perform it so that it would not affect to the painting waveform.

Software control:

It follows the same process for both pressure and flow reading. However, it adds an extra consideration:

- A timer with period equal to the selected heart period, starting without any delay is used to determine when the systole pulse (DTR set to HIGH).
- A timer with period equal to the selected heart period, starting with a delay determined by the selected duty cycle is used to send a diastole pulse (DTR set to LOW).

It is important to mention that as it is not possible to read and write at the same time in the same port, when pressure is being read pulses are sent through the flow port and when flow is being read pulses are sent through the pressure port.

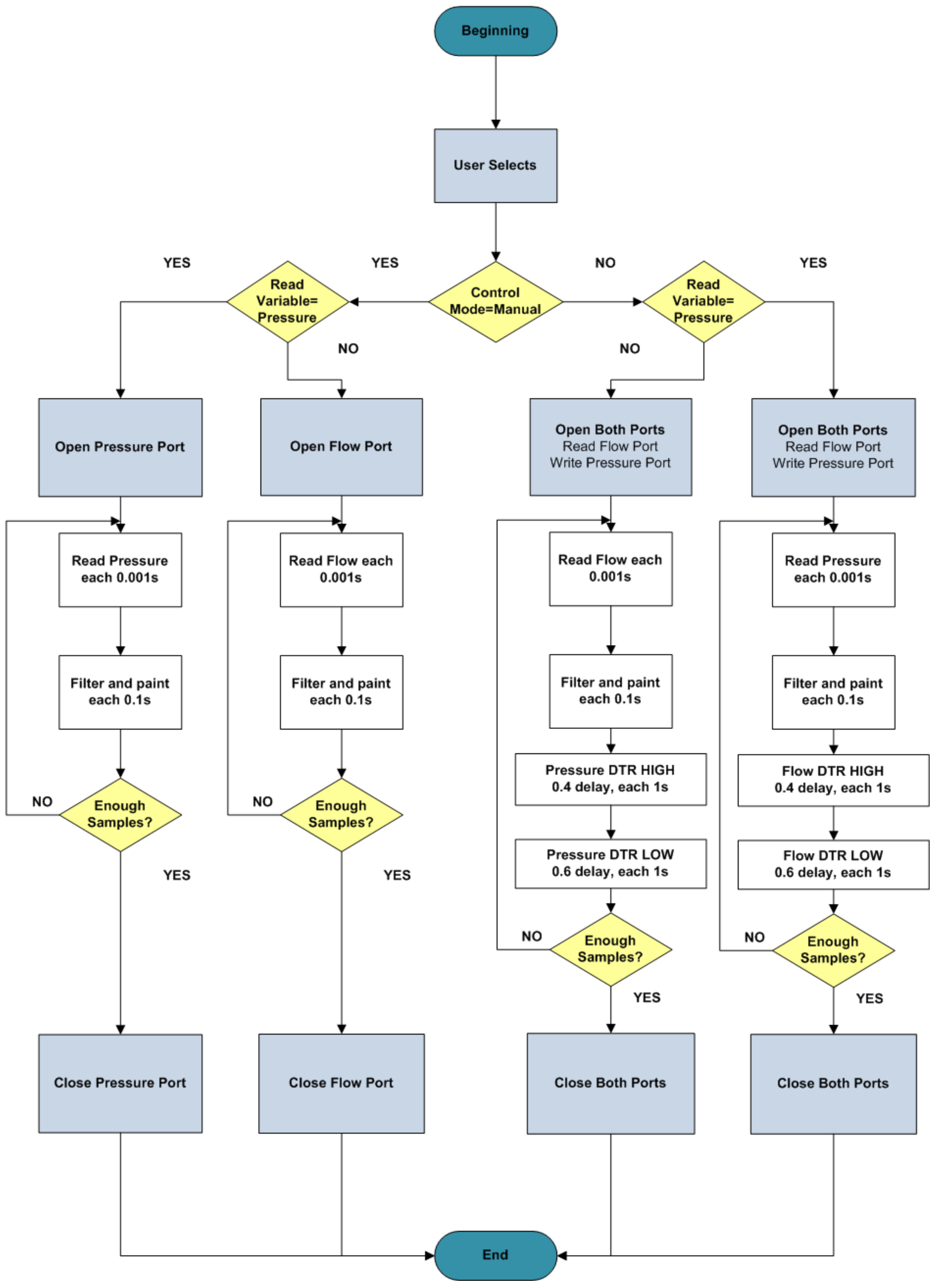

Beginning
User Selects
Control Mode=Manual
YES
NO
Read Variable= Pressure
YES
NO
Read Variable= Pressure
YES
NO
Open Pressure Port
Open Flow Port
Open Both Ports Read Flow Port Write Pressure Port
Open Both Ports Read Flow Port Write Pressure Port
Read Pressure each 0.001s
Read Flow each 0.001s
Read Flow each 0.001s
Read Pressure each 0.001s
Filter and paint each 0.1s
Filter and paint each 0.1s
Filter and paint each 0.1s
Filter and paint each 0.1s
Pressure DTR HIGH 0.4 delay, each 1s
Pressure DTR LOW 0.6 delay, each 1s
Flow DTR HIGH 0.4 delay, each 1s
Flow DTR LOW 0.6 delay, each 1s
Enough Samples?
NO
YES
Close Pressure Port
Close Flow Port
Close Both Ports
Close Both Ports
End

Figure 36: Reading Interface: general flow diagram. Obtained using Microsoft Office VISIO.

However, MATLAB has a characteristic that hampers data acquisition code for the practical simulation interface. MATLAB works sequentially unless the specialized parallel computing toolbox is being used. This means that a function cannot be executed while a different one is running; it has to wait until the previous function finishes working. This feature affects to the timers. If their callback function takes too long, it may still be running at the time of execution of the next timer. This limits the code structure possibilities, making it necessary to program fast, short and effective lines to provide a real-time data visualization. The timers that are most affected are the timers for reading and decoding data, so the most efficient way of decoding each of the transducers has tried to be found.

Regarding the design of the interface, a similar layout to one used in the theoretical simulation interface has been used. In this way it will be easier for the students to understand both interfaces during the laboratory practice. As an example, the welcome page is showed in Fig. 38. Fig. 39 shows a pressure real time recording, using the software timing control method. As it can be observed, at that exact moment software was sending a systole pulse (red mark). Fig. 40 shows a more advanced time of the same recording, when software is sending a diastole pulse (green mark).

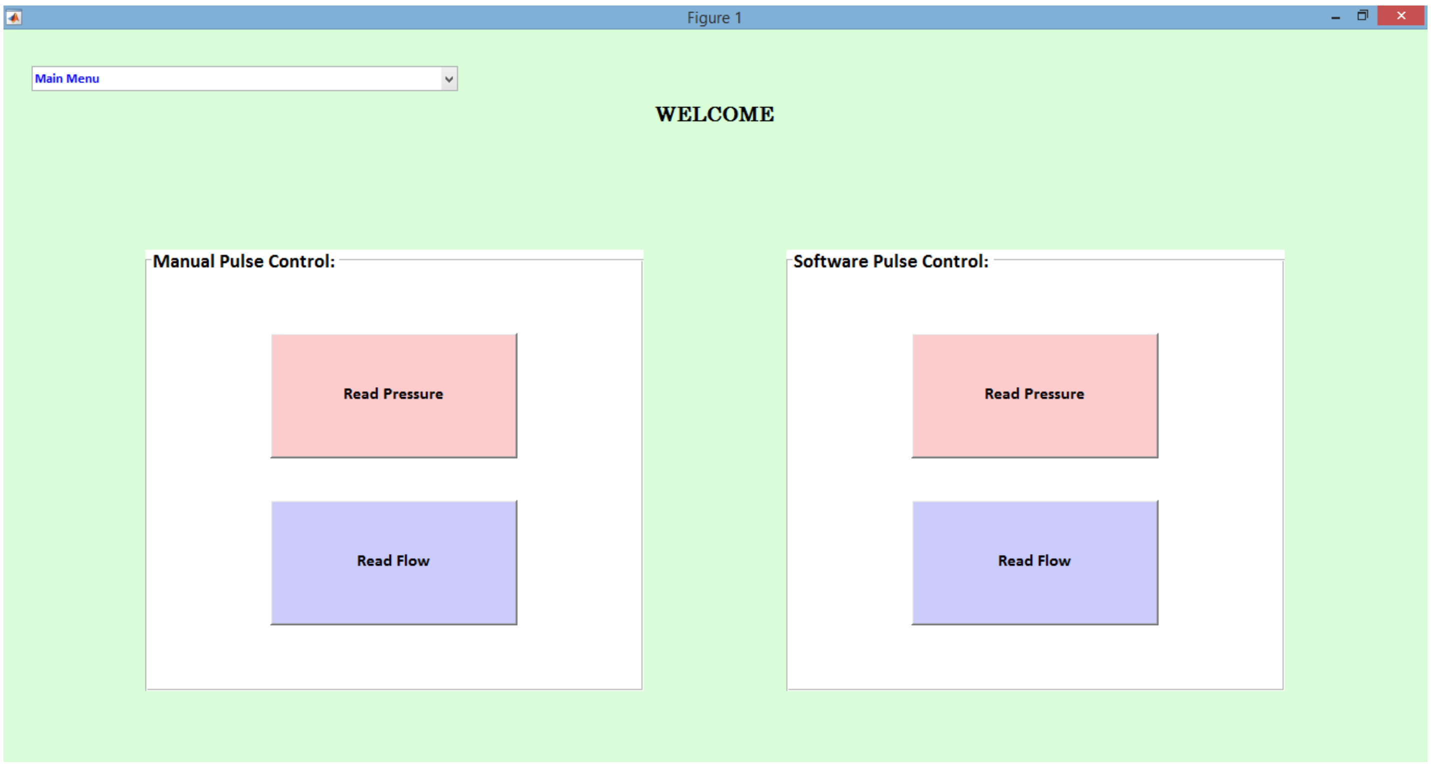


Figure 37: Reading Interface: welcome page.

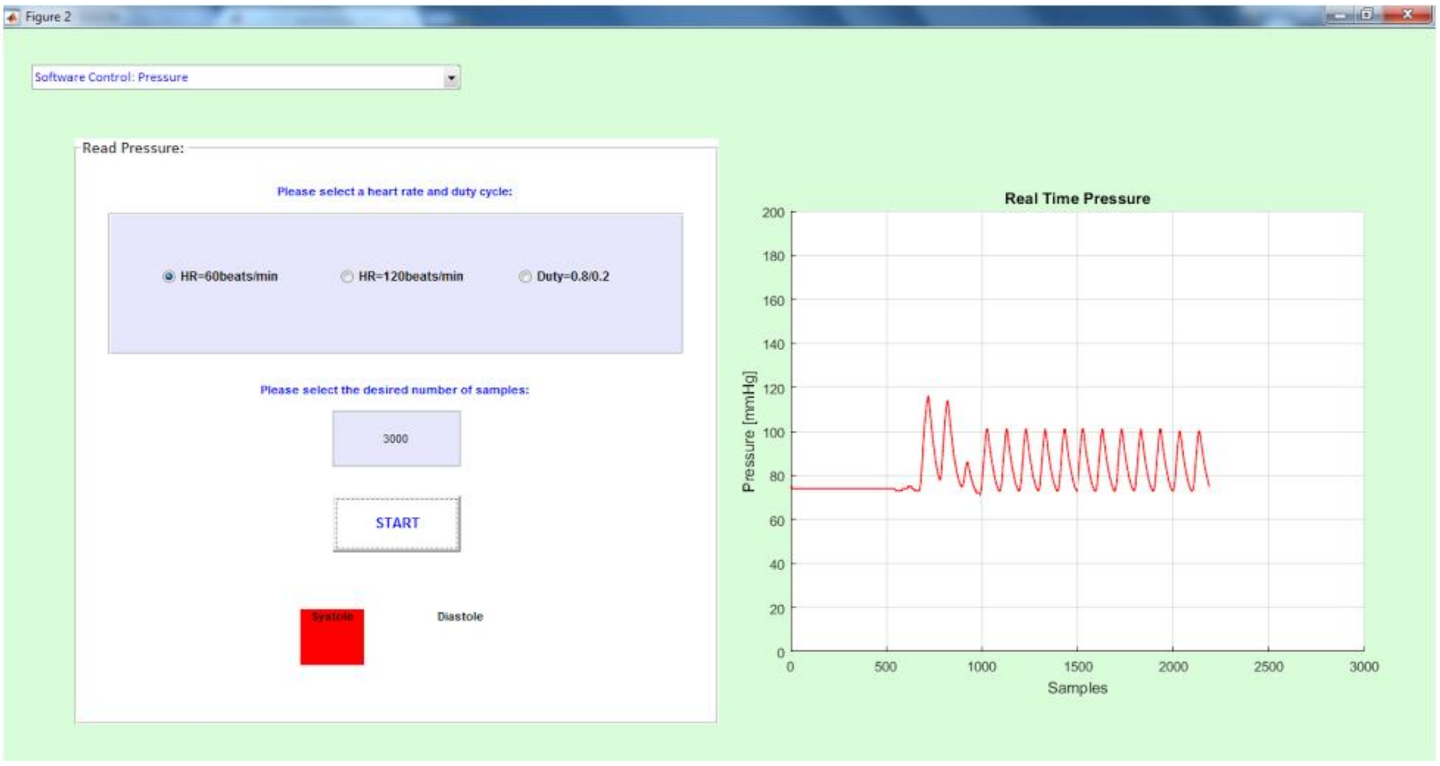


Figure 39: Reading Interface: Software timing, pressure recording (systole).

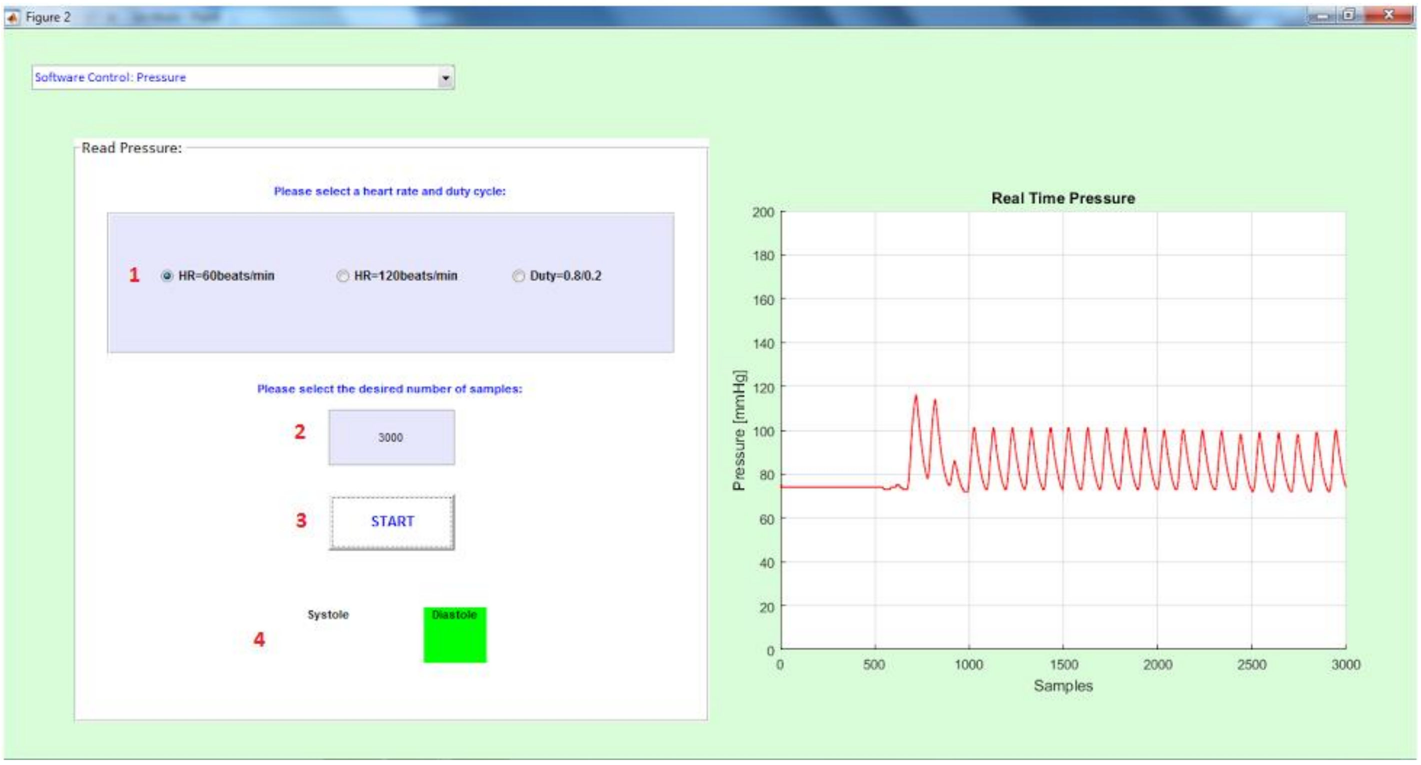


Figure 38: Reading Interface: Software timing, pressure recording (diastole).

As it can be observed the window is composed by four main parts:

1. Allows the user to select three different heart rate and duty cycle options.
2. Allows the user to decide the number of pressure samples to record.
3. Starts the recording.
4. Provides with an easy visualization of every systole and diastole pulse.

Observe the irregularity in Fig. 40 at the beginning of the recording. This is caused by the fact that the pressure and vacuum regulators are set to zero when the recording starts and once the first data (with the value of the constant offset of $70\ mmHg$) is captured, both are increased to induce the membrane to pump. The irregularity corresponds to those first membrane pumps.

4.3. Practice for Students

The developed practice for students has been created based on other Biomedical Engineering laboratory practices. It has been divided in three main parts:

- In the first part, the students will work with the theoretical simulation interface. The main objectives for this part are:
    - Allow students to become familiar with the different models and parameters, as well as with the different waveforms displayed.
    - They will answer questions on how parameters affect to physiological variables and how the different heart cycle steps affect to the aortic pressure and flow.
- In the second part, the students will work with the practical environment.
    - They will learn which are the different components integrating the device and their related physiological function.
    - They will observe how device behaves under different input conditions.
- In the third part students will answer questions on how theoretically and physically simulated relate and learn how they are different from measured physiological data.

The practice developed can be observed in the annex.

5. RESULTS

5.1. Assessment of the Models

In order to assess the performance of the developed models four main analysis have been performed:

- Analysis of the waveforms and the physiological features represented.
- Analysis of the selected initial condition.
- Analysis of the frequency behavior of the system.
- Analysis of the interface response.

5.1.1. Analysis of features represented

As it can be observed in Fig. 41, the simplest model (two-element Windkessel) is enough to successfully predict the exponential pressure decay that takes place at the aorta during diastole, when the aortic valve is closed.

$$P_{dias}(t) = P_{es} \cdot e^{\frac{-t}{RC}}$$

Where P_{es}=end-systolic aortic pressure.

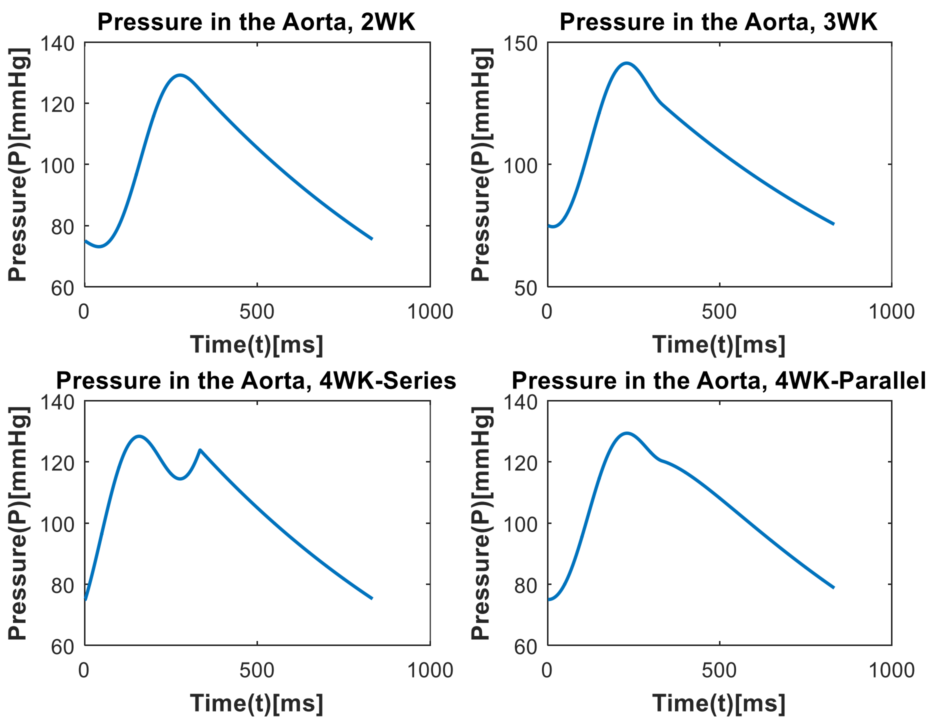


Figure 40: Comparison between obtained simulated pressure waveforms. Obtained using MATLAB.

In order to prove the accuracy of such prediction, the pressure waveform has been fitted with an exponential function (Fig. 42) that has the following parameters:

$$P(t) = a \cdot e^{b \cdot t}$$

Where $b = -0.0009999 \ \frac{1}{ms} = -0.9999 \frac{1}{s}$ and $a = 173.7 \ [mmHg]$.

The fitted curve can be evaluated using its R-square coefficient, which is given by MATLAB and is found to be $\mathrm{R-square} = \ 1$. This means that the approximation is perfect.

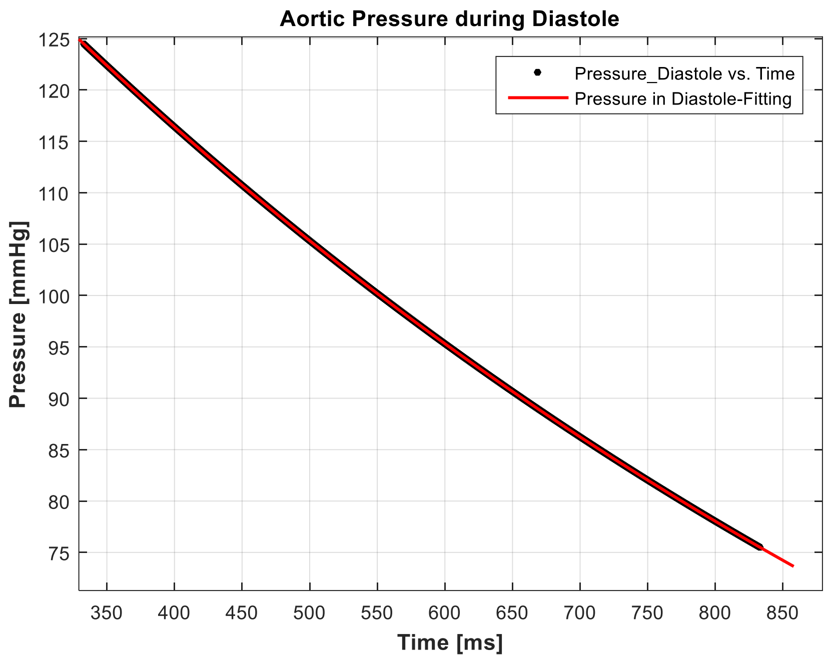


Figure 41: Aortic pressure during diastole fitting curve. Obtained using MATLAB curve fitting toolbox.

The characteristic time (τ) is an interesting parameter describing the decay and may be calculated as

$$\tau = \frac{-1}{b} = \frac{1}{0.9999\frac{1}{s}} \sim 1s = R \cdot C = 133.28 \cdot 10^{6} \ \frac{kg}{s \cdot m^{4}} \cdot 0.75 \cdot 10^{-8} \ \frac{s^{2} \cdot m^{4}}{kg} = 1s$$

Recall Fig. 22, where it could be observed that $P_{es} \sim 125 \ mmHg$. However according to the described model P_{es} should be equal to $173.7 \ [mmHg]$. This would only be achieved assuming that the entire pressure waveform over one heart cycle is modeled by the exponential equation, so that $P(t_0) = 173.7 \ mmHg$.

This shows how the two-element Windkessel model fails to accurately predict the behavior of the system during systole.

However, the obtained equation might be really helpful to calculate relevant values belonging to the decreasing section of the waveform. As an example, the value at the end of the waveform, just before the next systole starts, may be used as the diastolic pressure value, allowing to compare it with real data.

$$P(t = 833\,ms) = 173.7mmHg \cdot e^{-0.0009999\frac{1}{ms}\cdot 833ms} = 75.52\,mmHg = P_{diastolic}$$

A similar approximation may be used to calculate the systolic pressure, considering that the highest point of the waveform takes place at $t = 280\,ms$:

$$P(t = 280\,ms) = 173.7mmHg \cdot e^{-0.0009999\frac{1}{ms}\cdot 280ms} = 131.31\,mmHg = P_{systolic}$$

The poor prediction of pressure during systole is the reason why higher complexity models are needed.

Fig. 43 shows how, in addition, more complex models can better represent feature such as the specific point at which the aortic valve closes and diastole starts. In Fig. 43 it can also be observed that the main difference between the three-element Windkessel and the 4-element-parallel Windkessel is found at the pressure values, as the three-element model reaches higher values. As it can be observed function decay during diastole follows the same function in all the models although it happens later in the four-element configuration.

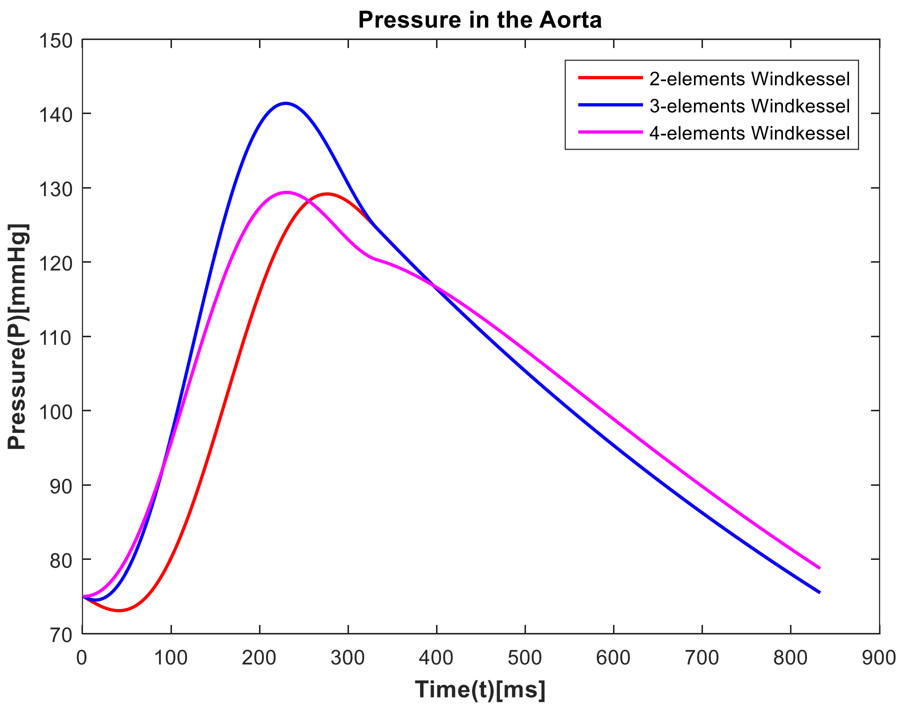


Figure 42: Comparison of the Windkessel models used in the interface. Obtained using MATLAB.

Data Origin	$P_{diastolic}$	$P_{systolic}$
Healthy Adult Human []	$60 - 90\ mmHg$	$100 - 140\ mmHg$
Two-element Windkessel	$75.52\ mmHg$	$131.31\ mmHg$
Three-element Windkessel	$\sim 75\ mmHg$	$\sim 140\ mmHg$
Four-element-parallel Windkessel	$\sim 75\ mmHg$	$\sim 130\ mmHg$

Table 4: Systolic and diastolic pressure Values comparison.

Table 4 shows how all models give pressure values within the range established by real data.

5.1.2. Initial condition analysis

The first initial condition selected was $P(i_1) = 0$, in order to check how the system evolved from resting. After observing it stabilized at $P \sim 75\ mmHg$, this value was selected as initial condition.

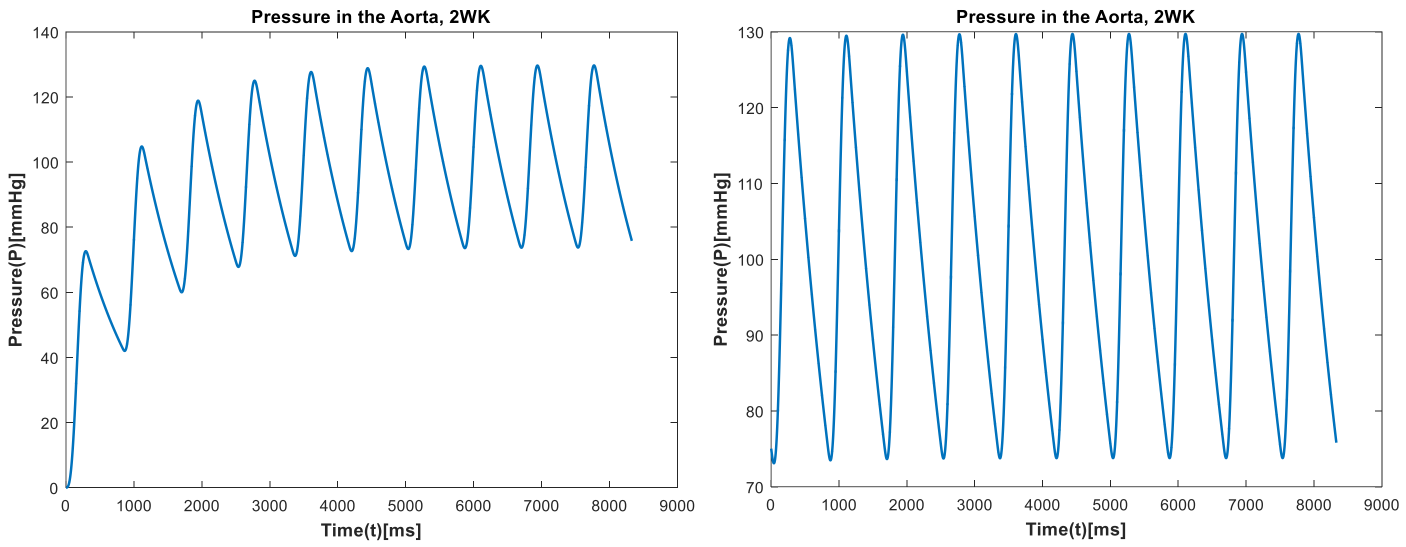


Figure 43: Comparison of the performance of the system using the two considered initial conditions. $\mathrm{P(i_1)} = 0mmHg$ *(a) and* $\mathrm{P(i_1)} = 75\mathrm{mmHg}$ *(b).*

5.1.3. Impedance analysis

The behavior of the system at different frequencies is interesting for studying since the system will work at different heart rates. Fig. 45 shows the Bode diagrams for each of the models, obtained using the Laplace definition of the model presented in section 4.1.1. The results show how all models are stable except in the case of the four-element-series Windkessel. This instability is the why it has been excluded from the theoretical simulation interface.

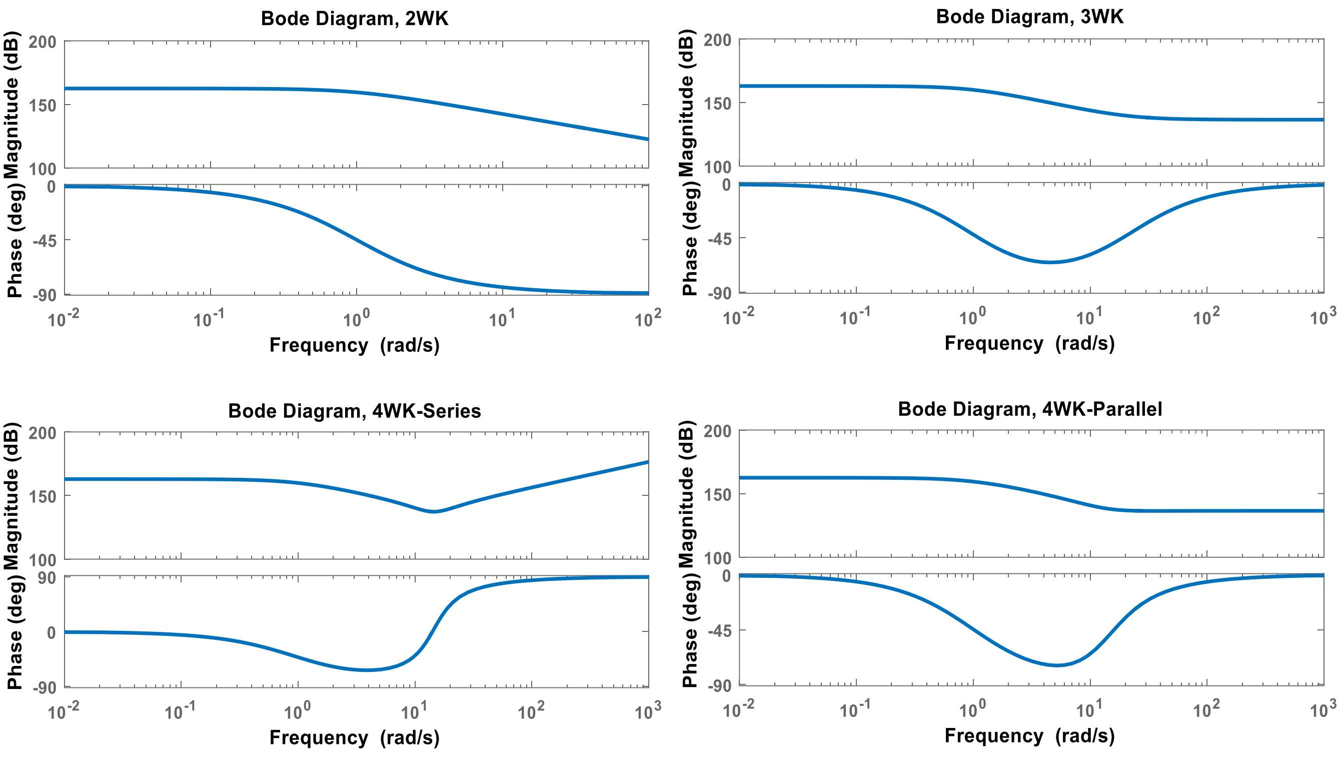


Figure 44: Bode diagrams of the different models. Obtained using MATLAB.

A comparison between the impedances of the developed models with an impedance analysis found in the literature [17] is displayed in Fig. 46. The x-axis represents the frequency using a linear scale, instead of a logarithmic one. The y-axis, instead of using a $20\log(|A|)\,dB$ transform, uses a natural scale.

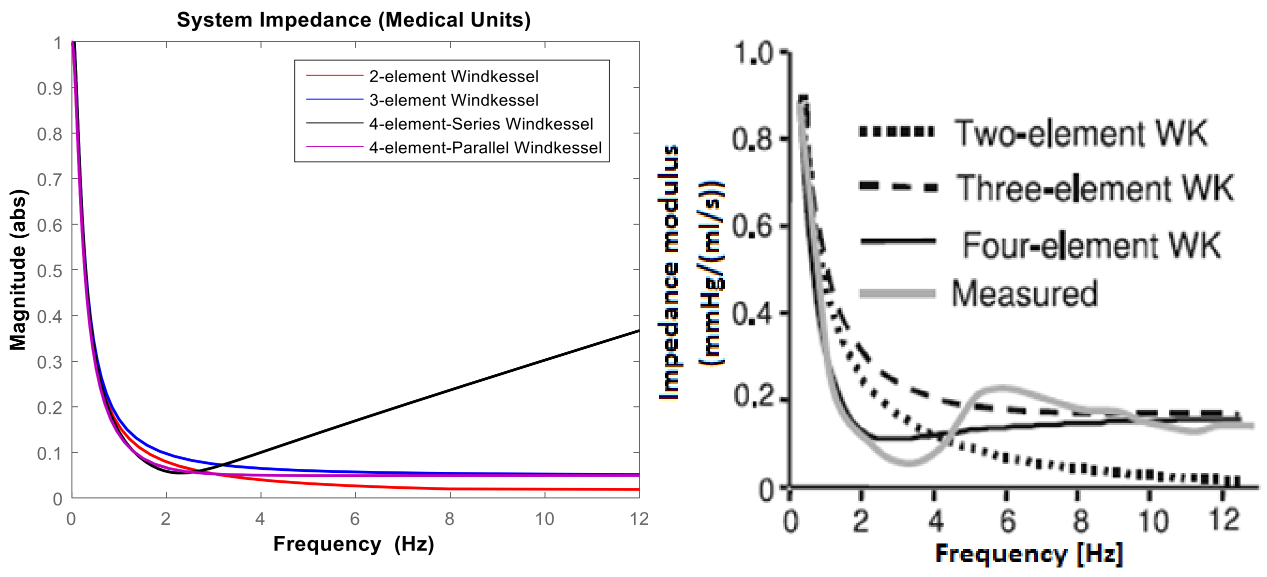


Figure 45: System impedance comparison between obtained models (a) and literature models (b). Realize in (b) the four-element model that is being used is the parallel one.

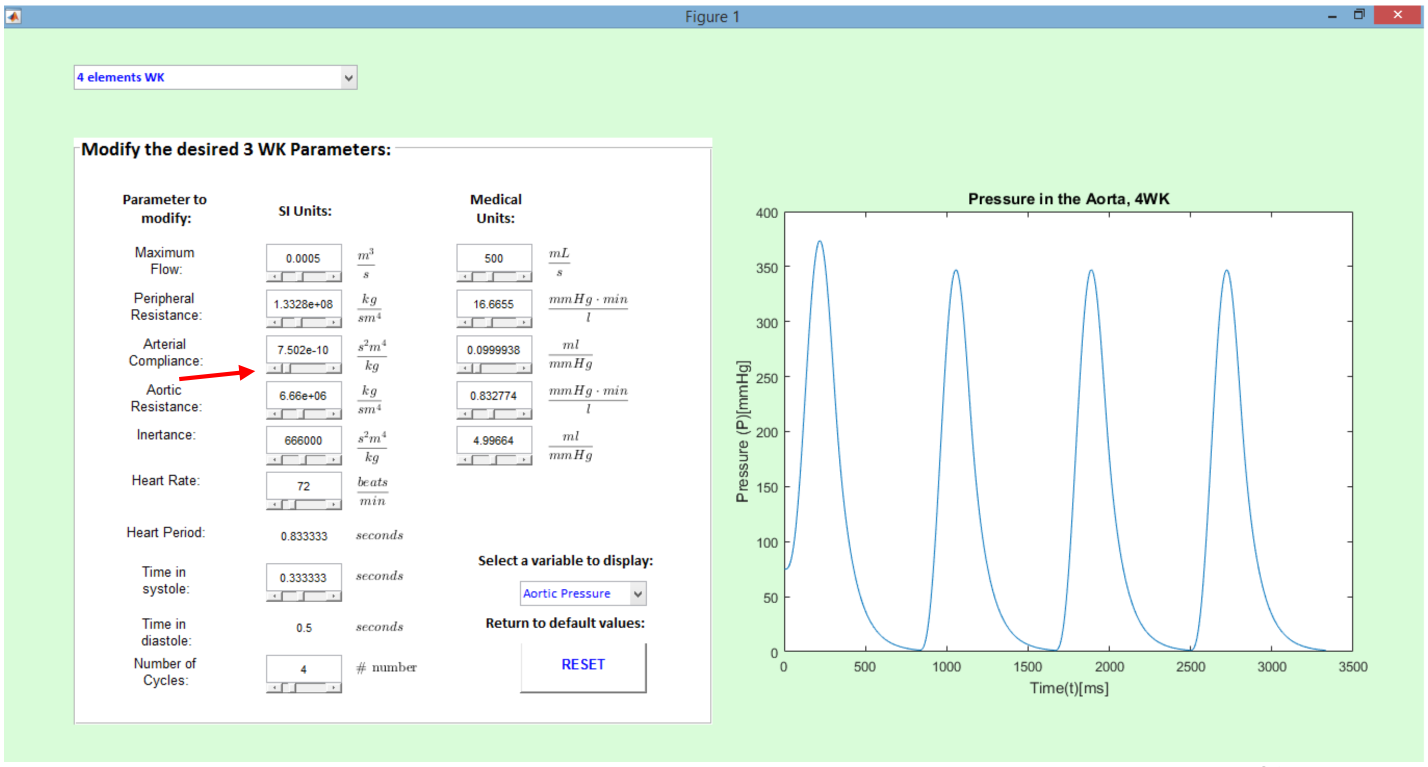
Excluding the unstable four-element-series configuration from the obtained plot, it can be observed that the developed models behave similar to the literature at different frequencies.

The need of increasing the complexity of the models can be also observed through this analysis:

- The two-element model proves to be stable but wrong, as there is not constant stability at higher heart rates (it goes to zero).
- As mentioned in chapter four the characteristic impedance is not exactly a resistor, but needs to be understood in terms of oscillatory phenomena. Due to the addition of this parameter the three-element model is stable, showing a constant correct value, at high frequencies. However it is far from the expected impedance at low frequencies.
- The four-element-parallel model has the best of both worlds. It shows a fast and correct evolution at low heart rates and perfect stability at higher frequencies.

5.1.4. Parameter modification analysis

The aim of this section is simply to show how the user by modifying a parameter in the interface can obtain the expected result in the displayed waveform. In Fig. 47 arterial compliance has been decreased. As a result pressure has increased significantly, from $P_{systolic} \sim 130\ mmHg$ (Fig. 24) to $P_{systolic} \sim 375\ mmHg$ (Fig. 47). This is expected as decreased compliance is related to arterial stiffening which is the principal cause of increasing systolic pressure with advancing years and in patients with arterial hypertension [44].



Modify the desired 3 WK Parameters:

Parameter to modify:	SI Units:		Medical Units:	
Maximum Flow:	0.0005	$\frac{m^3}{s}$	500	$\frac{mL}{s}$
Peripheral Resistance:	1.3328e+08	$\frac{kg}{sm^4}$	16.6655	$\frac{mmHg \cdot min}{l}$
Arterial Compliance:	7.502e-10	$\frac{s^2m^4}{kg}$	0.0999938	$\frac{ml}{mmHg}$
Aortic Resistance:	6.66e+06	$\frac{kg}{sm^4}$	0.832774	$\frac{mmHg \cdot min}{l}$
Inertance:	666000	$\frac{s^2m^4}{kg}$	4.99664	$\frac{ml}{mmHg}$
Heart Rate:	72	$\frac{beats}{min}$		
Heart Period:	0.833333	$seconds$		
Time in systole:	0.333333	$seconds$		
Time in diastole:	0.5	$seconds$		
Number of Cycles:	4	# number		

Select a variable to display: Aortic Pressure

Return to default values:

Figure 46: Simulation Interface: four-element-parallel Windkessel model, displaying pressure with decreased arterial compliance over four heart cycles.

5.2. Characterization of the Device

Data obtained from the device has been analyzed in order to determine if the performance of the model is successful. Fig. 48 shows the real time recording of the flow, and Fig. 49 shows the real time recording of the flow while the input pressure to the system is being altered. As it can be observed, the system success to show this alteration in real-time.

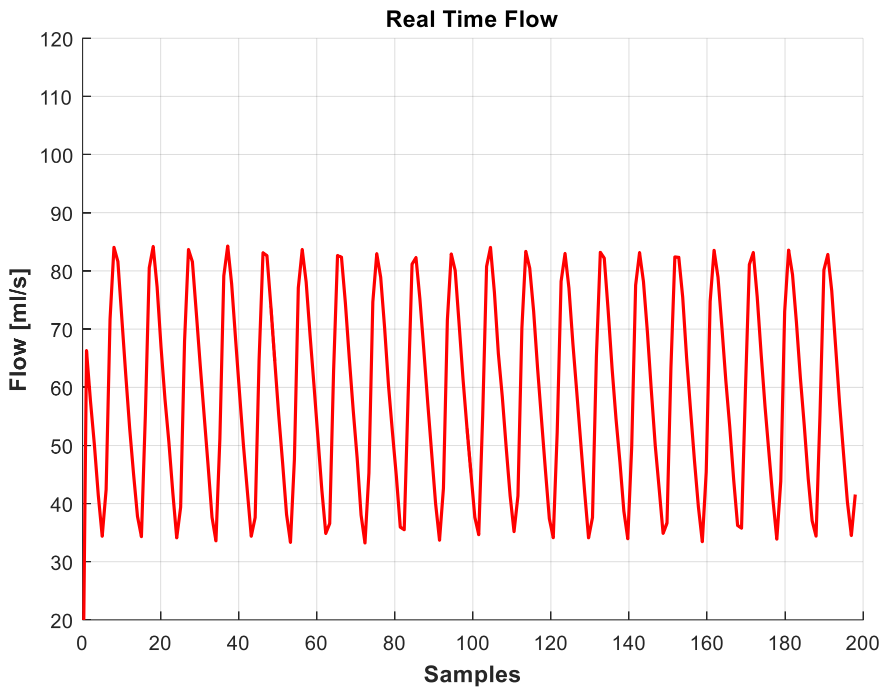


Figure 47: Real time recording of the flow. Obtained using MATLAB.

By using MATAB timing functions "*tic*" and "*toc*", it was found out that the recording was taken in $19.853369\, s \sim 20s$. As 20 periods were recorded in that time, the heart rate can be determined as $60\, \frac{beats}{min}$. This agrees with the heart rate that was set in the device.

The cardiac output of the system under these conditions can be calculated as $85\, \frac{mL}{s} \cdot \frac{60s}{min} \cdot \frac{L}{1000mL} = 5.1\, \frac{L}{min}$, which is within the normal range for a healthy adult (see section 4.1.2.).

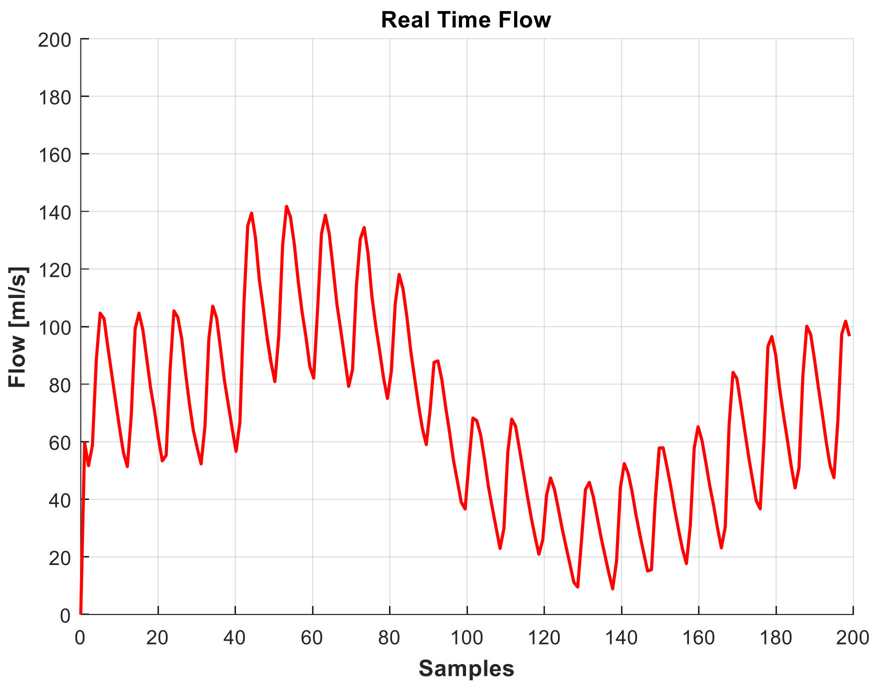


Figure 48: Real time recording of the flow while the input pressure is being altered.

The same process is followed for a pressure recording. Comparison between a normal recording and an altered input pressure can be observed in Figs. 50 and 51 respectively.

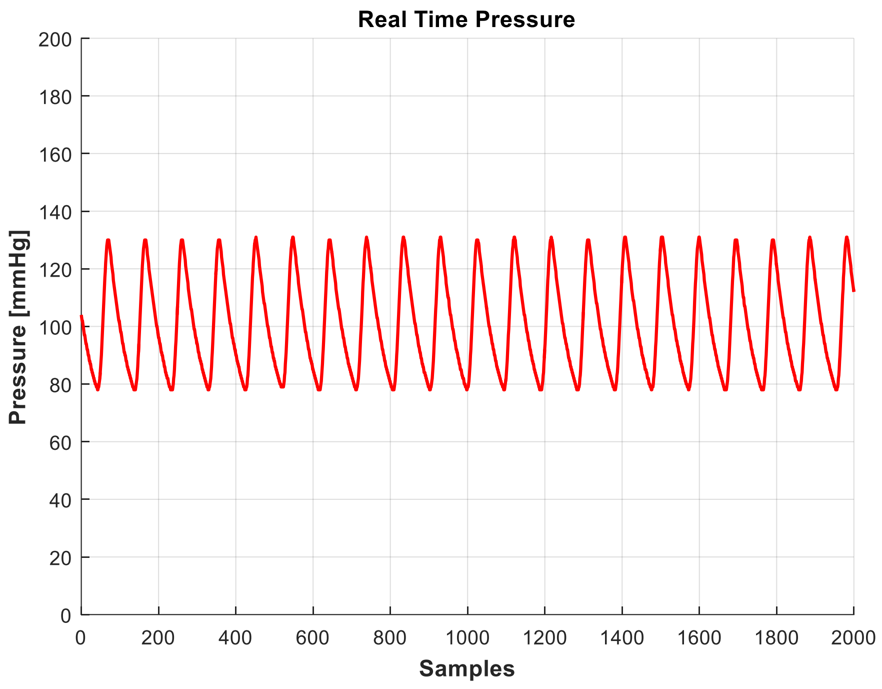


Figure 49: Real time pressure recording.

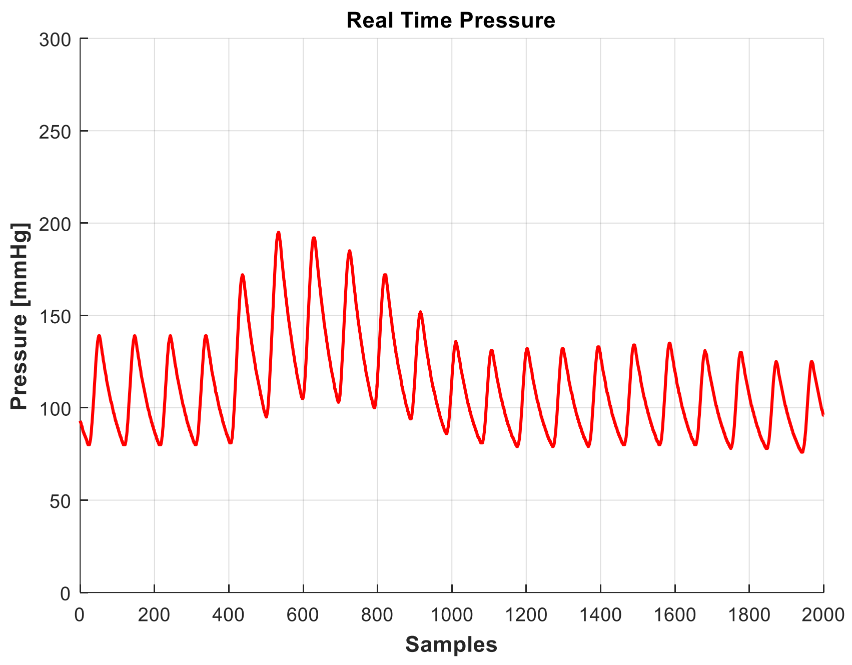


Figure 50: Real time pressure recording while input pressure to the system is being altered.

Again 20 complete periods have been recorded in $19.962132\,s \sim 20\,s$. This makes sense as heart cycle controls were not modified to perform this recording. This shows that the new electronic board functions correctly and send pulses in a regular manner.

Again, the recorded values are within the normal range for a healthy adult human, as the obtained systolic pressure is of $\sim 130\,mmHg$ and the diastolic pressure is of $\sim 80\,mmHg$. Realize that the pressure waveform is similar to the pressure obtained by the simulated 2-element Windkessel model. This was expected as the physical device models the arterial system by means of a compliant element and an adjustable valve modelling the peripheral resistance.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

The main objective of this project was to develop an educational tool that would help students to understand better the physiology of the cardiovascular system and would provide them with a practical environment for completing their education with practical instruction, before dealing with real cases.

All the three specific goals have been reached: the theoretical simulation is available, the physical device is successfully functioning and the entire platform is ready to be used by students during laboratory practices.

The models developed work well in comparison with the reviewed literature, being the four-element configuration the one giving the best results. Moreover, their code implementation is robust. Parameters are related to real elements of the cardiovascular system. User and technical requirements have been satisfied.

The system is capable of reading real time pressure and flow data, and displaying it in a user interface. The replaced electronic boards work in a safer, more efficient way and the original design of the device has been conserved.

Finally, simulated variables, both theoretical and physical, represent successfully the physiology of the human arterial system by means of a Windkessel model.

6.2 Future Work

In the future, the developed laboratory practice should be performed by Biomedical Engineering students. In order to determine how useful the educational tool is and what other improvements can be considered (e.g. to check if the interface layout is the most appropriate), they should provide feedback after the laboratory session.

Two main improvements regarding the functioning of the physical device could be implemented in the future.

Firstly, the accuracy of the software timing mode could be enhanced. At this moment, the user can successfully induce the heart to pump at different rates and with different duty cycles. However, if the timing switch is set to software mode while the code is not executing, there will be a constant voltage value at the DTR terminal, so in order to avoid the membrane to move while a recording is not being performed, pressure and vacuum regulators must be set to zero. Once the reading code starts executing, pressure can be increased but the first samples recorded will not have any meaning. Once the code stops executing, the regulators need to be set to zero again. In the future, it should be allowed to the user to control timing from the computer without concerning about the pressure regulator.

Secondly, it would be useful to allow the user reading real-time pressure and flow at the same time. In this way pressure-volume loops could be obtained and compared with the simulated ones. There are two main problems regarding this issue.

- In the first place, the differences between the transducers, the different communication velocities and data format, make very difficult to program a code with the needed timing requirements. The callback function of the timers would need to execute not one but two different decoding routines and finish in time so that they do not interrupt the next timer. This might be achieved if executing routines in parallel was possible and with a tight control of the acquired raw data.
- In the second place, if this wanted to be achieved, a different way of controlling timing through software should be considered, as at the moment the port which is not being used for the recording is being used to send the pulses.

The fact that the electronic modifications have been designed in a modular way will make easier any future work. Each module of the system may be replaced using components that offer characteristics that fit better with the nature of the research being carried out at each moment. This does not only apply to the physical modifications but also to all the codes developed, which are structured in functions so that new pieces of code can be easily integrated.

References

[1] World Health Organization. *Cardiovascular Diseases: Key Facts.* Retrieved 05 15, 2016, from http://www.who.int/mediacentre/factsheets/fs317/en/

[2] European Commission. *CE Marking.* Retrieved 06 10, 2016, from http://ec.europa.eu/growth/single-market/ce-marking/index_en.htm

[3] Ministerio de la Presidencia (Gobierno de España). (n.d.). *Real Decreto 53/2013 (Boletín oficial del Estado).* Retrieved 06 10, 2016, from https://www.boe.es/diario_boe/txt.php?id=BOE-A-2013-1337

[4] European Commission. *Medicinal Products for Human Use.* Retrieved 06 10, 2016, from http://ec.europa.eu/health/human-use/clinical-trials/regulation/index_en.htm

[5] AENOR. *Seguridad de las máquinas (ISO 12100:2012).* Retrieved 06 10, 2016, from http://www.aenor.es/aenor/actualidad/actualidad/noticias.asp?campo=4&codigo=22995&tipon=2#.V2ExlbuLTIU

[6] Tortora, G. (2013). *Princiles of Anatomy and Physiology* (13 ed.). Wiley.

[7] Truant, R. (2013). *Design of a Pulsatile Pumping System for Cardiovascular Flow PIV Experimentation.* University of Victoria, Faculty of Mechanical Engineering, Victoria, British Columbia.

[8] Herman, I. P. (2007). *Physics of the Human Body.* Springer-Verlag Berlin Heidelberg.

[9] Beltina.org. *Beltina enciklopedia of health: Cardiac Cycle.* Retrieved 05 15, 2016, from http://www.beltina.org/health-dictionary/cardiac-cycle-phases-diagram-definition.html

[10] Mozaffarian, D. (2015). *Heart Disease and Stroke Statistics-2016 Update, A Report From the American Heart Association.*

[11] Kokalari, I. (2013). Review on lumped parameter method for modeling the blood flow in systemic arteries. *Journal of Biomedical Science and Engineering, 6*, 92-99 .

[12] Alfonso, M. R. (2014). Conceptual model of arterial tree based on solitons by compartments. *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (pp. 3224 - 3227). Chicago, IL: IEEE.

[13] Ursino, M. (1998). Interaction between carotid baroregulation and the pulsating heart: a mathematical model. *American Physiological Society*.

[14] Savage, V. (2012). *Modeling Vascular Networks.* University of California, Los Angeles.

[15] Hyperphysics. *Ohm's Law-Poiseuille's Law*. Retrieved 03 17, 2016, from http://hyperphysics.phy-astr.gsu.edu/hbase/electric/watcir2.html

[16] Manning, T. S. (2002). Validity and Reliability of Diastolic Pulse Contour Analysis (Windkessel Model) in Humans. *Hypertension, 39*, 963-968.

[17] Westerhof, N. (2008). The arterial Windkessel. *Medical & Biological Engineering*.

[18] University of Ottawa. (n.d.). *Modeling Fluid Systems.* Retrieved 03 03, 2016, from http://www.site.uottawa.ca/~rhabash/ESSModelFluid.pdf

[19] Westerhof, N. (1971). An artificial arterial system for pumping hearts. *Journal of Applied Physiology, 31*, 776-781.

[20] Segers, P. (2008). Three-and-four-element Windkessel models: Assessment of their fitting performance in a large cohort of healthy middle-aged individuals. *Journal of Engineering in Medicine, 222*, 417-428.

[21] Sullivan, C. (2004). *Lumped Fluid Systems.* Retrieved 03 25, 2016, from http://www.dartmouth.edu/~sullivan/22files/Fluid_sys_anal_w_chart.pdf

[22] Biomedical Research Models. (n.d.). *Animal Models*. Retrieved 06 12, 2016, from http://www.brmcro.com/Animal-Models.html

[23] Swartz, D. D. (2013). Animal Models for Vascular Tissue-Engineering. *Current Opinion in Biotechnology, 24 (5)*, 916-925.

[24] Tu. (2015). *Computational Hemodynamics – Theory, Modelling and Applications.* Springer.

[25] Kuzmin, D. *Introduction to Computational Fluid Dynamics.* University of Dortmund, Institute of Applied Mathematics. Retrieved 06 12, 2016, from http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/lecture1.pdf

[26] Martorell, J. (2012). Engineered arterial models to correlate blood flow to tissue biological response. *Annals of the New York Academy of Sciences, 1254 (1)*, 51–56.

[27] Sotiropoulos, F. (2009). A review of state-of-the-art numerical methods for simulating flow through mechanical heart valves. *Medical & Biological Engineering & Computing, 47 (3)*, 245–256.

[28] Cebral, J. R. (2005). Characterization of Cerebral Aneurysms for Assessing Risk of Rupture By Using Patient-Specific Computational Hemodynamics Models. *American Journal of Neuroradiology, 26*, 2550-2559.

[29] Pulse. *Custom Medical Exhibit Models*. Retrieved 06 13, 2016, from http://www.pulsemdm.com/custom-medical-exhibit-models/

[30] United Biologics. *Products*. Retrieved 06 12, 2016, from http://www.unitedbiologics.com/products.html

[31] SynDaver Labs. *SynDaver Patient*. Retrieved 06 12, 2016, from http://syndaver.com/shop/syndaver/synthetic-humans/syndaver-patient/

[32] Coxworth, B. (2015, 05 14). *SynDaver Patient offers a lively alternative to cadavers*. Retrieved from http://www.gizmag.com/syndaver-patient/37516/

[33] SynDaver Labs. *EKG Simulator*. Retrieved 06 12, 2016, from http://syndaver.com/shop/synatomy/pumps-accessories/ekg-simulator/

[34] Medsby. *ECG simulator*. Retrieved 06 12, 2016, from http://explore.coimbatorestartups.com/startup/medsby

[35] Ortiz-León, G. (2013). An updated Cardiovascular Simulation Toolbox. *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)* (pp. 1901 - 1904). Beijing: IEEE.

[36] PhysioNet. *A Cardiovascular Simulator for Research*. Retrieved 06 12, 2016, from https://www.physionet.org/physiotools/rcvsim/

[37] Gates, P. E. (2006). Decline in large elastic artery compliance with age: a therapeutic target for habitual exercise. *British Journal of Sports Medicine, 40(11)*, 897–899.

[38] Hlavác, M. (2004). *Windkessel Model Analysis In MATLAB.* Biomedical Engineering, Brno. Retrieved from http://www.feec.vutbr.cz/EEICT/2004

[39] Edwards Lifesciences. *Normal Hemodynamic Parameters – Adult.* Retrieved 04 28, 2016, from http://icverpleegkundige.com/files/Heamodynamische-parameters.pdf

[40] Punto Flotante S.A. *Estándares de comunicaciones RS232, RS422, RS485.* Retrieved 05 20, 2016, from http://www.puntoflotante.net/RS485.htm

[41] MedLab GmbH. *Invasive blood pressure OEM module Data Sheet: EG 02000.* Retrieved 03 10, 2016, from http://www.medlab-gmbh.de/english/downloads/ibp_oem202.pdf

[42] MedLab GmbH. *IBP Monitoring: Invasive Blood Pressure Module*. Retrieved 05 20, 2016, from http://www.medlab-gmbh.de/english/modules/ibpmonitoring/index.html

[43] EmTec. *Digiflow: OEM Ultrasonic Flow Measuring Board.* Retrieved 03 10, 2016, from http://manualzz.com/doc/4127354/digiflow-oem-ultrasonic-flow-measuring-board---user-manual--

[44] O'Rourke, M. (1990). Arterial Stiffness, Systolic Blood Pressure, and Logical Treatment of Arterial Hypertension. *Hypertension, 15*, 339-347.

ANNEXES

I. General costs

1.-Author
Nuria Peña Pérez

2.-Departament
Biomedical and Aerospace engineering

3.-Project description	
- Title:	Windkessel Modelling of the Human Arterial System
- Duration (hours):	695

4.-Project budget (€)
17197,04

5.-Budget breakdown			
Human resources			
Category	Human-hours	Cost of human-hour	Cost(€)
Technical Engineer	105	55	5775
Total			5775
Materials			
Description		Company	Cost(€)
Windkessel prototype		SEDECAL	5000
Total			5000
Details of material cost for timing control development			
Description	Price(€/piece)	Quantity	Price(€)
Optoisolator	0,43	2	0,86
LM7812	0,19	1	0,19
LM7805	0,22	1	0,22

Resistors (SMD)	0,02	6	0,12
Capacitor	0,1	2	0,2
TIP120	0,31	1	0,31
1N4007	0,05	3	0,15
Board	4,24	1	4,24
74C04	0,41	1	0,41
Wires, connectors			0,8
Total			7,5

Other direct costs		
Description	Company	Cost(€)
Internet	Telefonica	530
Windows 8	Microsoft	99,95
Visio 2007	Microsoft	79,99
Multisim	National Instruments	640
MatLab	MathWorks	2000
Office 2013	Microsoft	79,99
Total		3429,93

6.-General costs and industrial benefit		
On direct human resources costs		
Total	924	16%

7.-Summary of costs	
Description	Budget on total costs (€)
Personnel	5775
Materials	5007,5
Funtioning costs	3429,93
Total without IVA	14212,43
IVA 21%	2984,61
Estimated total	17197,04

II. Practice: Cardiovascular Physiology

Note: The answers to questions marked in blue should include in your Lab notebook, to be evaluated.

Equipment used

Simulation Interface

In this practice we will use an interface that allows learning from different arterial models.

Windkessel Hydraulic Device

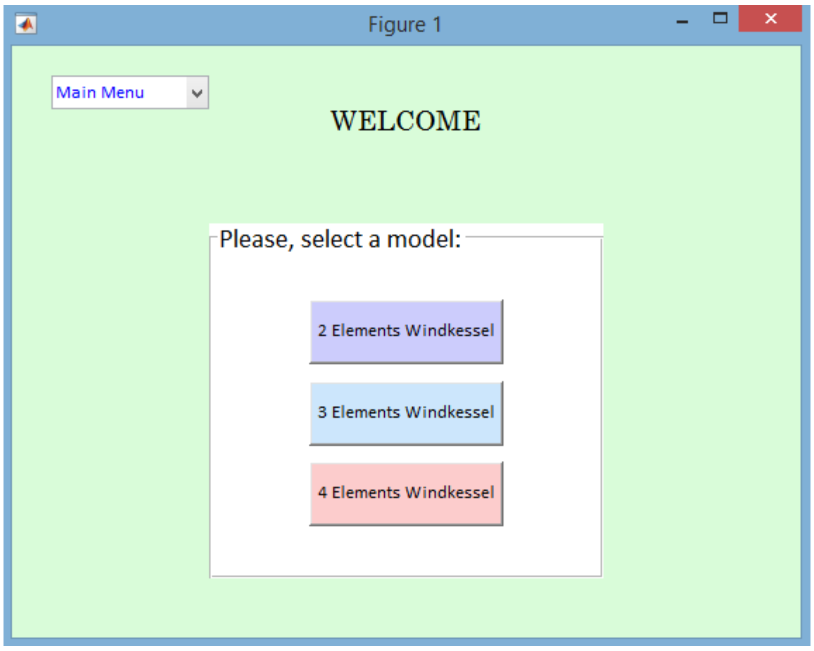
Prototype manufactured by the company SEDECAL.

Practice Outline

Section 1

Before using the hydraulic device we will get familiar with the Windkessel model using the theoretical simulator at the computer.

Download the folder "*WK_Interface_Simulation_FINAL*". Open MATLAB 2015b (or newer versions) and run the code "WK_*Main _Interface*". Make sure all the files in the compressed folder are in the MATLAB directory. You should see the following window:



Once you have accessed to the main page of the interface, please answer the following questions:

1: Access to the 2-element Windkessel window. Pay attention to the aortic pressure waveform, taking into account that by default one heart period is represented. Can you identify the different heart cycle steps? Explain the relationship each of them has with the shape of the function. Now, go to the 3 and 4 element models, what differences do you observe in the pressure and why do you think that happens? Tip: take into account adding more elements to a model means more complexity is represented.

2: Calculate the cardiac output for a patient with a maximum flow per beat of 424 ml/s, a heart rate of 180 beats/min and a systole duration of 0.1s. Include/draw the aortic flow that patient will have. Which would be the systolic and diastolic pressure values for that patient? Are they physiologically normal? (Use the 2-element Windkessel for this question) **Tip:** to observe drastic changes you may need to represent several heart cycles.

3: Compare the pressure-volume loop obtained by any of the models with the ones you have seen at class. Why are they different?

4: Go to the 4-element model, vary each parameter independently and observe the effect they have on aortic pressure and flow. **Tip:** to observe drastic changes you may need to represent several heart cycles.

- **a)** Do all parameters affect to both pressure and flow? Why?
- **b)** Relate the effect of each of the parameters with a different disease (e.g. increasing heart rate with tachycardia).

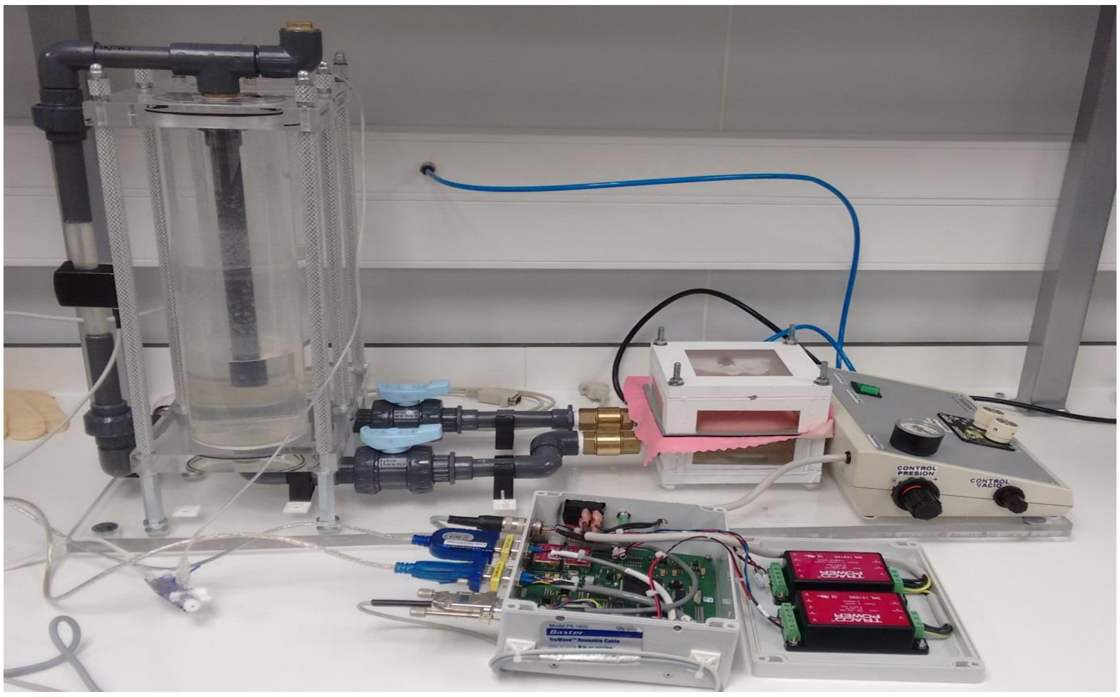
Section 2

Now, let's start working with the hydraulic device.

1: Observe the device and locate the main parts on the picture:

1. Main switch
2. Pressure connection

3. Pressure/Vacuum rotatory knobs
4. Timing option switch (software versus manual)
5. Heart
6. Heart cycle controls (rotatory knobs)
7. Compliant element
8. Fluid reservoir
9. Peripheral resistance
10. Flow transducer location/Flow RS-232 port
11. Pressure transducer location/Pressure RS-232 port



Connect the arterial simulator to the pressure connection and turn it on. **Make sure that pressure and vacuum are set to zero before switching the device on!** Connect the pressure port to the computer COM7 and the flow port to the computer COM6, switch the transducer's boards on. Open in the MATLAB directory the folder "*Read_Windkessel_Interface*" and run the main interface code.

Note- If you are going to use the software timing option make sure that you always increase pressure and vacuum after you start running the code and **set pressure and flow quickly to zero after the recording is finished**. Set the timing switch to software mode.

If you are using the manual timing option, set the timing switch to this mode. You will start hearing the pulses. You can modify both the heart rate and the duty cycle using the heart cycle controls. Increase both pressure and vacuum (turning both rotatory nodes to their right). Try to supply even pressure-vacuum quantities, so that the membrane simulating the heart remains pumping in the middle of its cavity. Otherwise you may produce a heart attack.

2: Go to pressure reading and press start for the default number of samples to observe the pressure recording in real-time. A txt file with the name "*PressureData*" will be generated after the process finishes, save this file in a different folder. Do this again, what happens when you modify the pressure/vacuum rotatory knobs during the recording? Do the same for the flow and file "*FlowData*".

Section 3

By looking at the recorded data, please answer the following questions:

1: Plot the vectors stored in each of the files and answer to the following questions:

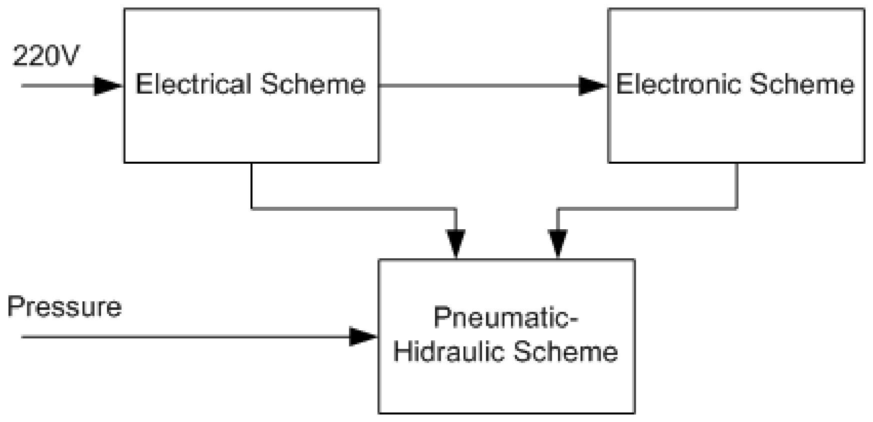
a) The heart is a flow pump. Describe the relationship it would have with the pressure assuming the arteries, arterioles and capillaries are simply exerting a resistance to the flow. However, when you varied the pressure knob during the recording the flow waveform is altered. How is this possible?

b) Plot the normal pressure together with the altered pressure:

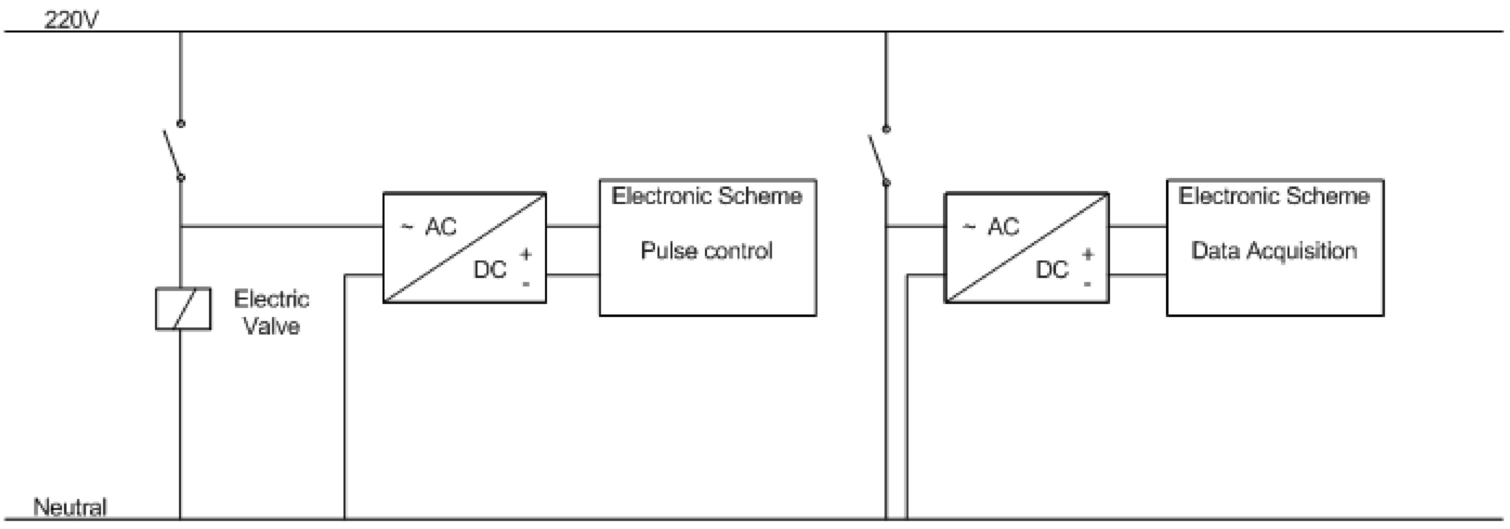
- **a.** Which are the systolic and diastolic pressure values for each waveform?
- **b.** Which of the models do you think is more similar to the obtained waveform from the ones we saw in section 1?
- **c.** Draw the electric circuit that corresponds to this model and explain the relationship each of the elements have with the parts of the hydraulic device. **Tip:** think of the role the following elements play in the hydraulic device and the location they have: heart, compliant element, fluid reservoir, peripheral resistance.

III. Schemes

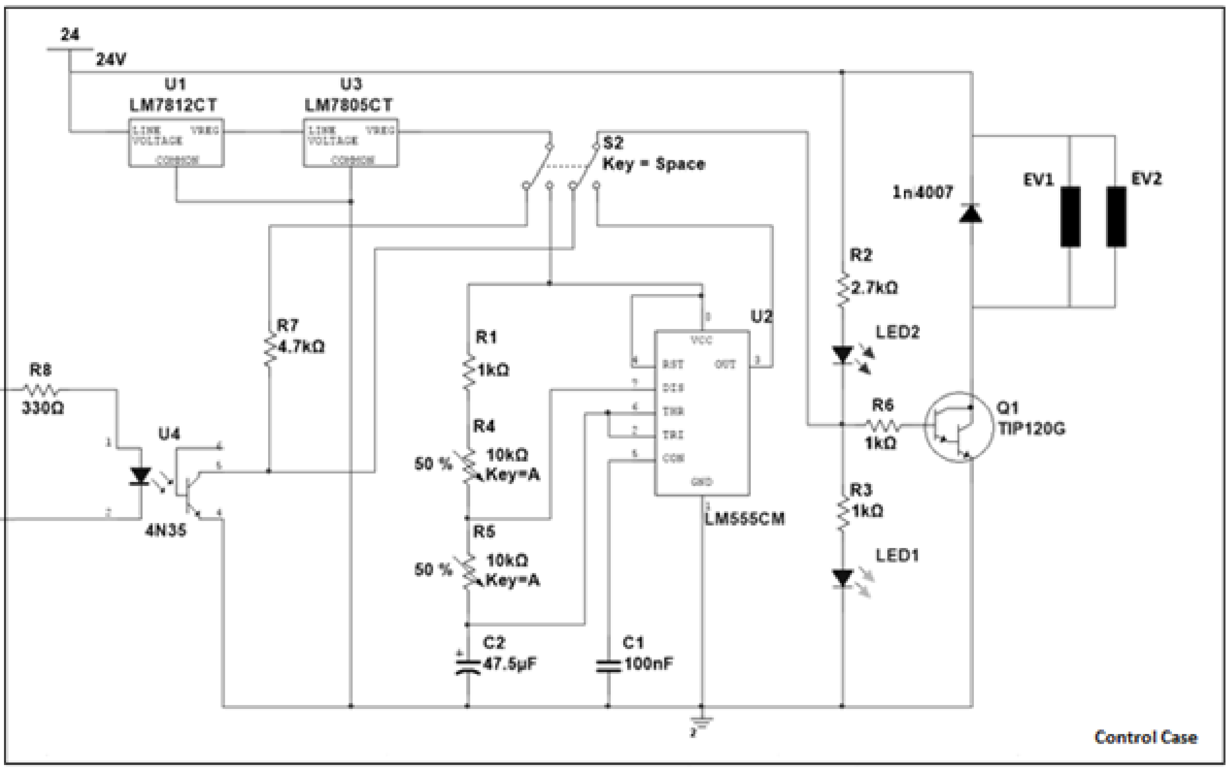
General scheme:



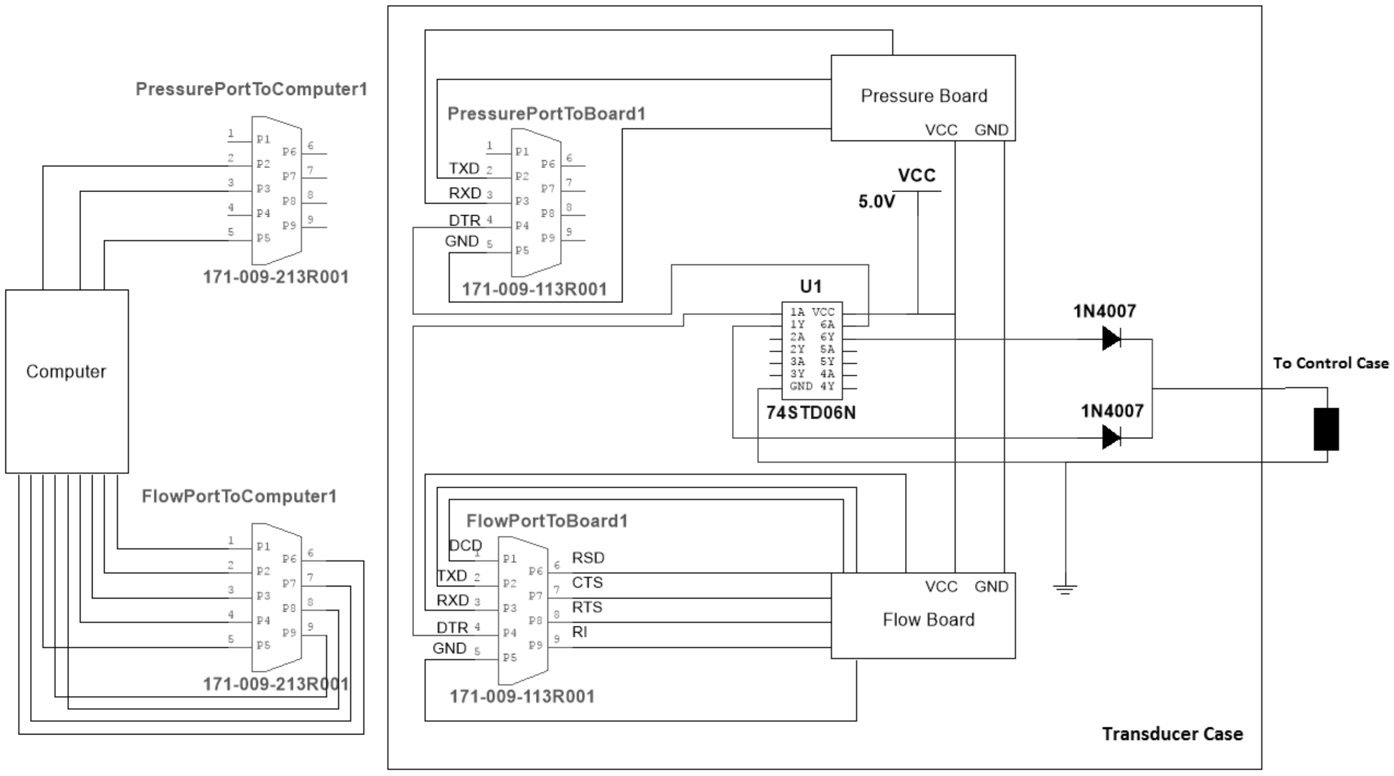
Electrical scheme:



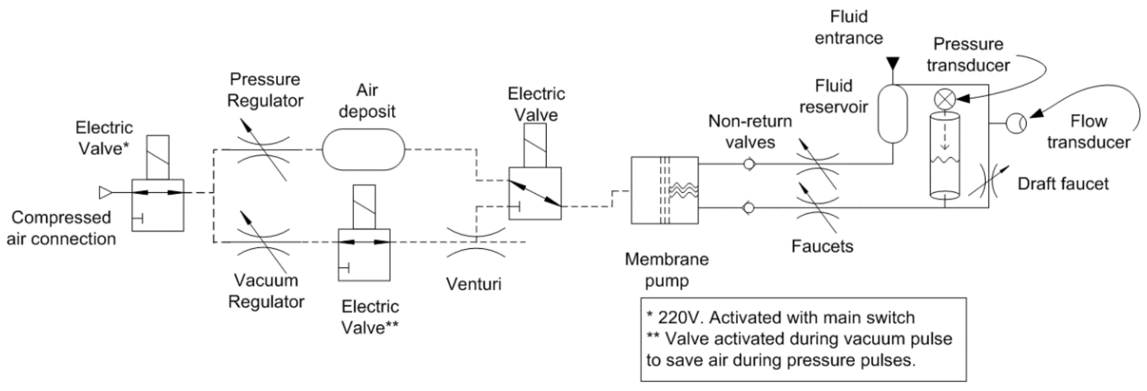
Electronic scheme: Part 1



Electronic scheme: Part 2

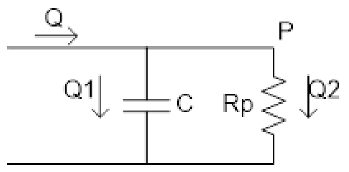


Pneumatic-Hydraulic scheme



IV. Full Mathematical Development

Two-element Windkessel



By using Kirchhoff's laws and Ohm's law, the circuit can be solved:

$$Q_1(t) = C \cdot \frac{dP(t)}{dt}$$

$$Q_2(t) = \frac{P(t)}{R_p}$$

$$Q(t) = Q_1(t) + Q_2(t)$$

Flow may be expressed as a function of the pressure with the described equation:

$$Q(t) = C \cdot \frac{dP(t)}{dt} + \frac{P(t)}{R_p} \leftrightarrow C \cdot s \cdot P + \frac{P}{R_p} = P\left(s \cdot C + \frac{1}{R_p}\right)$$

The equation may be transformed into Laplace domain, this definition will be used in the next section for performing the impedance analysis. Now, solving for the pressure as a function of previous samples:

$$Q(t) \cong Q(i)$$

$$Q(i) = C \cdot \frac{P(i) - P(i-1)}{\Delta t} + \frac{P(i)}{R_p}$$

$$\frac{R_p \cdot Q(i) \cdot \Delta t}{C} = \frac{\Delta t \cdot P(i)}{C} + P(i) - P(i-1)$$

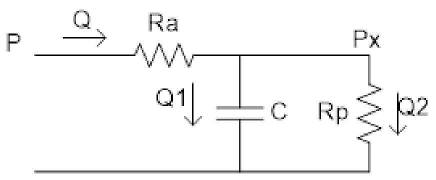
$$\frac{R_p \cdot Q(i) \cdot \Delta t}{C} = P(i) \cdot \left[R_p + \frac{\Delta t}{C}\right] - P(i-1) \cdot R_p$$

The final solution is:

$$P(i) = \frac{1}{R_p + \frac{\Delta t}{C}} \cdot \left[Q(i) \cdot \frac{R_p \cdot \Delta t}{C} + P(i-1) \cdot R_p\right]$$

Three-element Windkessel

The circuit is solved following the same procedure.



$$Q(t) = \frac{P(t) - P_x(t)}{R_a} \rightarrow P_x(t) = P(t) - Q(t) \cdot R_a$$

$$Q_1(t) = C \cdot \frac{dP_x(t)}{dt} = C \cdot \frac{d(P(t) - Q(t) \cdot R_a)}{dt}$$

$$Q_2(t) = \frac{P_x(t)}{R_p} = \frac{P(t) - Q(t) \cdot R_a}{R_p}$$

Assembling these equations:

$$Q(t) = Q_1(t) + Q_2(t)$$

$$Q(t) = C \cdot \frac{d(P(t) - Q(t) \cdot R_a)}{dt} + \frac{P(t) - Q(t) \cdot R_a}{R_p}$$

Rearranging the terms, the relationship between pressure and flow can be found:

$$Q(t) \cdot \left(1 + \frac{R_a}{R_p}\right) + C \cdot R_a \cdot \frac{dQ(t)}{dt} = C \cdot \frac{dP(t)}{dt} + \frac{P(t)}{R_p} \leftrightarrow Q \cdot \left(1 + \frac{R_a}{R_p} + C \cdot R_a \cdot s\right) = P(C \cdot s + \frac{1}{R_p})$$

Again, Laplace definition will be later useful for the impedance analysis.

Transforming every term from a continuous to a discrete function:

$$Q(i) \cdot \left(1 + \frac{R_a}{R_p}\right) + C \cdot R_a \cdot \frac{Q(i) - Q(i-1)}{\Delta t} = C \cdot \frac{P(i) - P(i-1)}{\Delta t} + \frac{P(i)}{R_p}$$

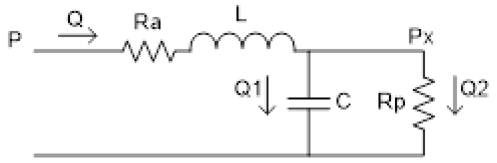
$$R_p \cdot \left\{Q(i) \cdot \left[1 + \frac{R_a}{R_p} + \frac{C \cdot R_a}{\Delta t}\right] - Q(i-1) \cdot \frac{C \cdot R_a}{\Delta t}\right\} = P(i) \cdot \left(\frac{C \cdot R_p}{\Delta t} + 1\right) - P(i-1) \cdot \frac{C \cdot R_p}{\Delta t}$$

Finally:

$$P(i) = \frac{1}{1 + \frac{C \cdot R_p}{\Delta t}} \cdot \left\{P(i-1) \cdot \frac{C \cdot R_p}{\Delta t} + R_p \cdot \left\{Q(i) \cdot \left[1 + \frac{R_a}{R_p} + \frac{C \cdot R_a}{\Delta t}\right] - Q(i-1) \cdot \frac{C \cdot R_a}{\Delta t}\right\}\right\}$$

<u>Four-element Windkessel Series</u>

The circuit is solved following the same procedure.



$$P_L(t) = L \cdot \frac{dQ(t)}{dt}$$

$$Q(t) = \frac{P(t) - P_L(t) - P_x(t)}{R_a} \rightarrow P_x(t) = P(t) - L \cdot \frac{dQ(t)}{dt} - Q(t) \cdot R_a$$

$$Q_1(t) = C \cdot \frac{dP_x(t)}{dt} = C \cdot \frac{d\left(P(t) - L \cdot \frac{dQ(t)}{dt} - Q(t) \cdot R_a\right)}{dt}$$

$$Q_2(t) = \frac{P_x(t)}{R_p} = \frac{P(t) - L \cdot \frac{dQ(t)}{dt} - Q(t) \cdot R_a}{R_p}$$

$$Q(t) = Q_1(t) + Q_2(t)$$

Assembling the previous equations:

$$Q(t) = C \cdot \frac{d\left(P(t) - L \cdot \frac{dQ(t)}{dt} - Q(t) \cdot R_a\right)}{dt} + \frac{P(t) - L \cdot \frac{dQ(t)}{dt} - Q(t) \cdot R_a}{R_p}$$

$$Q(t) = C \cdot \frac{dP(t)}{dt} - C \cdot L \cdot \frac{d^2Q(t)}{dt^2} + C \cdot R_a \cdot \frac{dQ(t)}{dt} + \frac{P(t)}{R_p} - \frac{L}{R_p} \cdot \frac{dQ(t)}{dt} - \frac{R_a}{R_p} \cdot Q(t)$$

Finally arriving to the solution in the temporal domain:

$$Q(t) \cdot \left(1 + \frac{R_a}{R_p}\right) + \frac{dQ(t)}{dt} \cdot \left(\frac{L}{R_p} - C \cdot R_a\right) + C \cdot L \cdot \frac{d^2Q(t)}{dt^2} = P(t) \cdot \left(\frac{1}{R_p}\right) + C \cdot \frac{dP(t)}{dt}$$

And to the solution in the Laplace domain:

$$Q \cdot \left[\left(1 + \frac{R_a}{R_p}\right) + s \cdot \left(\frac{L}{R_p} - C \cdot R_a\right) + C \cdot L \cdot s^2\right] = P \cdot \left(\frac{1}{R_p} + C \cdot s\right)$$

Transforming the solution into a function of the previous samples:

$$Q(i) \cdot \left(1 + \frac{R_a}{R_p}\right) + \frac{Q(i) - Q(i-1)}{\Delta t} \cdot \left(\frac{L}{R_p} - C \cdot R_a\right) + C \cdot L \cdot \frac{Q(i) - 2 \cdot Q(i-1) + Q(i-2)}{\Delta t^2}$$
$$= P(i) \cdot \left(\frac{1}{R_p}\right) + C \cdot \frac{P(i) - P(i-1)}{\Delta t}$$

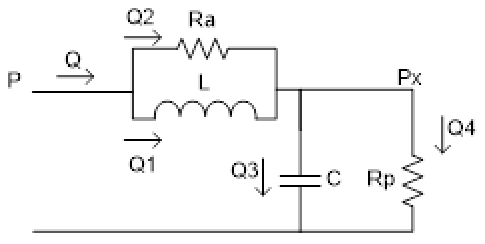
$$Q(i) \cdot \left[\left(1 + \frac{R_a}{R_p}\right) + \frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{C \cdot L}{\Delta t^2}\right] - Q(i-1) \cdot \left[\frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{2 \cdot C \cdot L}{\Delta t^2}\right] + Q(i-2) \cdot \frac{C \cdot L}{\Delta t^2}$$
$$= P(i) \cdot \left(\frac{1}{R_p} + \frac{C}{\Delta t}\right) + P(i-1) \cdot \frac{C}{\Delta t}$$

Finally:

$$P(i) = \frac{1}{\frac{1}{R_p} + \frac{C}{\Delta t}} \cdot \left\{P(i-1) \cdot \frac{C}{\Delta t} + Q(i) \cdot \left[\left(1 + \frac{R_a}{R_p}\right) + \frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{C \cdot L}{\Delta t^2}\right] - Q(i-1)\right.$$
$$\left. \cdot \left[\frac{\left(\frac{L}{R_p} - C \cdot R_a\right)}{\Delta t} + \frac{2 \cdot C \cdot L}{\Delta t^2}\right] + Q(i-2) \cdot \frac{C \cdot L}{\Delta t^2}\right\}$$

<u>Four-element Windkessel Parallel</u>

Due to the increased complexity of this model, it is analyzed directly in the Laplace domain and



later transformed into temporal domain to obtain the final solution.

$$Q(t) = \frac{P(t)}{Z_t}$$

$$Z_t = \frac{s \cdot L \cdot R_a}{s \cdot L + R_a} + \frac{\frac{1}{s \cdot C} \cdot R_p}{\frac{1}{s \cdot C} + R_p} = \frac{s \cdot L}{1 + s \cdot \frac{L}{R_a}} + \frac{\frac{1}{s \cdot C \cdot R_p}}{1 + \frac{1}{s \cdot C \cdot R_p}} =$$

$$= \frac{s \cdot L \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right) + \frac{1}{s \cdot C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)}{\left(1 + s \cdot \frac{L}{R_a}\right) \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right)} =$$

$$= \frac{s \cdot L \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right) + \frac{1}{s \cdot C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)}{1 + \frac{1}{s \cdot C \cdot R_p} + s \cdot \frac{L}{R_a} + \frac{L}{C \cdot R_p \cdot R_a}}$$

$$Q(s) \cdot \frac{s \cdot L \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right) + \frac{1}{s \cdot C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)}{1 + \frac{1}{s \cdot C \cdot R_p} + s \cdot \frac{L}{R_a} + \frac{L}{C \cdot R_p \cdot R_a}} = P(s)$$

$$Q(s) \cdot \left[s \cdot L \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right) + \frac{1}{s \cdot C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)\right] = \left(1 + \frac{1}{s \cdot C \cdot R_p} + s \cdot \frac{L}{R_a} + \frac{L}{C \cdot R_p \cdot R_a}\right) \cdot P(s)$$

$$Q(s) \cdot \left[s \cdot L \cdot \left(1 + \frac{1}{s \cdot C \cdot R_p}\right) + \frac{1}{s \cdot C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)\right] = \left[\frac{\left(s \cdot C \cdot R_p + 1\right) \cdot R_a + L \cdot s \cdot \left(s \cdot C \cdot R_p + 1\right)}{s \cdot C \cdot R_p \cdot R_a}\right] \cdot P(s)$$

$$Q(s) \cdot \left[s \cdot L \cdot \left(s + \frac{1}{C \cdot R_p}\right) + \frac{1}{C} \cdot \left(1 + s \cdot \frac{L}{R_a}\right)\right] \cdot C \cdot R_p \cdot R_a = \left(\left(s \cdot C \cdot R_p + 1\right) \cdot R_a + L \cdot s \cdot \left(s \cdot C \cdot R_p + 1\right)\right) \cdot P(s)$$

$$Q(s) \cdot \left[s^2 \cdot L + s \cdot \left(\frac{L}{C \cdot R_a} + \frac{L}{C \cdot R_p}\right) + \frac{1}{C}\right] \cdot C \cdot R_p \cdot R_a = \left(s \cdot C \cdot R_p \cdot R_a + R_a + L \cdot s^2 \cdot C \cdot R_p + s \cdot L\right) \cdot P(s)$$

Applying the transformation: $s^n = \frac{d^n}{dt^n}$

$$\frac{d^2Q(t)}{dt^2} \cdot L \cdot C \cdot R_p \cdot R_a + \frac{dQ(t)}{dt} \cdot \left(L \cdot \left(R_p + R_a\right)\right) + Q(t) \cdot R_p \cdot R_a = \frac{d^2P(t)}{dt^2} \cdot L \cdot C \cdot R_p + \frac{dP(t)}{dt} \cdot \left(C \cdot R_p \cdot R_a + L\right) + P(t) \cdot R_a$$

After transforming every term from a continuous to a discrete function:

$\frac{Q(i)-2\cdot Q(i-1)+Q(i-2)}{\Delta t^2}\cdot L\cdot C\cdot R_p\cdot R_a+\frac{Q(i)-Q(i-1)}{\Delta t}\cdot\left(L\cdot\left(R_p+R_a\right)\right)+Q(i)\cdot R_p\cdot R_a=\frac{P(i)-2\cdot P(i-1)+P(i-2)}{\Delta t^2}\cdot$ $L\cdot C\cdot R_p+\frac{V(i)-V(i-1)}{\Delta t}\cdot\left(C\cdot R_p\cdot R_a+L\right)+P(i)\cdot R_a$

$Q(i)\cdot\left[\frac{L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}+\frac{L\cdot\left(R_p+R_a\right)}{\Delta t}+R_p\cdot R_a\right]-Q(i-1)\cdot\left[\frac{2\cdot L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}+\frac{L\cdot\left(R_p+R_a\right)}{\Delta t}\right]+Q(i-2)\cdot\frac{L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}=$ $P(i)\cdot\left[\frac{L\cdot C\cdot R_p}{\Delta t^2}+\frac{C\cdot R_p\cdot R_a+L}{\Delta t}+R_a\right]-P(i-1)\cdot\left[\frac{2\cdot L\cdot C\cdot R_p}{\Delta t^2}+\frac{C\cdot R_p\cdot R_a+L}{\Delta t}\right]+P(i-2)\cdot\frac{L\cdot C\cdot R_p}{\Delta t^2}$

Finally:

$$P(i)=\frac{1}{\frac{L\cdot C\cdot R_p}{\Delta t^2}+\frac{C\cdot R_p\cdot R_a+L}{\Delta t}+R_a}\cdot\left\{P(i-1)\cdot\left[\frac{2\cdot L\cdot C\cdot R_p}{\Delta t^2}+\frac{C\cdot R_p\cdot R_a+L}{\Delta t}\right]-P(i-2)\cdot\frac{L\cdot C\cdot R_p}{\Delta t^2}+Q(i)\cdot\left[\frac{L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}+\frac{L\cdot\left(R_p+R_a\right)}{\Delta t}+R_p\cdot R_a\right]-Q(i-1)\cdot\left[\frac{2\cdot L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}+\frac{L\cdot\left(R_p+R_a\right)}{\Delta t}\right]+Q(i-2)\cdot\frac{L\cdot C\cdot R_p\cdot R_a}{\Delta t^2}\right\}$$

V. Codes

READING INTERFACE

Main Window

```
function WK_Main_Interface_Reading
% WK_MAIN_INTERFACE_READING
%
% This function opens the main window of the reading interface and
allows
% the user to decide which variable to read and timing mode to use
%
% Define some common variables
    x_max=200; % Number of samples to read, limit of the  x-axis
    y_min=0;   % Lower y-axis limit
    y_max=200; % Upper y-axis limit

    HR=60; % Heart rate [beats/min]
    T=60/HR; % Heart period [s]
    Duty=2/5; % Fraction of period occupied by systole
    Delay=Duty*T; % Time occupied by systole [s]

% Define Global Variables, handles will be an input to all the other
% functions used
    handles.data=[x_max y_min y_max HR T Duty Delay];
%In this way handles will be called for executig functions WK_2,WK_3
and WK_4


%CREATE MAIN FIGURE
    f =
figure('Visible','off','Toolbar','none','Menubar','none','Color',[0.85
0.99 0.85]);

%CREATE WELCOME TEXT
    txtWelcome = uicontrol('Style','text','Units','normalized',...
        'Position',[0.25 0.8 0.5 0.1],'FontSize',14,...
        'String','WELCOME',
'FontName','Century','BackgroundColor',[0.85 0.99 0.85]);

%CREATE POP UP MENU: it will be available for all the windows
displayed
    popup = uicontrol('Style',
'popup','Units','normalized','ForegroundColor',[0 0 1],...
       'String', {'Main Menu','Manual Control: Pressure','Manual
Control: Flow','Software Control: Pressure','Software Control:
Flow'},...
       'FontName','Calibri','FontSize',10,'Position', [0.02 0.85 0.3
0.1],...
       'Callback', @Menu);

%CREATE MAIN PANEL: only available for welcome page
    hp_manual = uipanel('Title','Manual Pulse Control:
','FontSize',14,...
         'BackgroundColor','white','FontName','Calibri',...
         'Position',[0.1 0.1 0.35 0.6]);
    hp_software = uipanel('Title','Software Pulse Control:
','FontSize',14,...
         'BackgroundColor','white','FontName','Calibri',...
         'Position',[0.55 0.1 0.35 0.6]);

%CREATE ALL MAIN PANEL BUTTONS
    %CREATE MANUAL CONTROL PRESSURE READING BUTTON
    btn_manual_pressure = uicontrol('Parent',hp_manual,'Style',
'pushbutton','Units','normalized',...
        'String', 'Read Pressure','Position',[0.25 0.55 0.5
0.3],'FontSize',11,...
        'FontName','Calibri','BackgroundColor',[0.99 0.8
0.8],'Callback', @manual_pressure);

    %CREATE MANUAL CONTROL FLOW READING BUTTON
    btn_manual_flow= uicontrol('Parent',hp_manual,'Style',
'pushbutton','Units','normalized',...
        'String', 'Read Flow','Position', [0.25 0.15 0.5
0.3],'FontSize',11,...
        'FontName','Calibri','BackgroundColor',[0.8 0.8
0.99],'Callback', @manual_flow);

    %CREATE SOFTWARE CONTROL PRESSSURE READING BUTTON
    btn_software_pressure= uicontrol('Parent',hp_software,'Style',
'pushbutton','Units','normalized',...
        'String', 'Read Pressure','Position',[0.25 0.55 0.5 0.3]
,'FontSize',11,...
        'FontName','Calibri','BackgroundColor',[0.99 0.8
0.8],'Callback', @software_pressure);

    %CREATE SOFTWARE CONTROL FLOW READING BUTTON
    btn_software_flow= uicontrol('Parent',hp_software,'Style',
'pushbutton','Units','normalized',...
        'String', 'Read Flow','Position',[0.25 0.15 0.5 0.3]
,'FontSize',11,...
        'FontName','Calibri','BackgroundColor',[0.8 0.8
0.99],'Callback', @software_flow);

%MAKE EVERYTHING VISIBLE
    f.Visible = 'on';

%CREATE THE PANELS FOR READING FUNCTIONS: only visible in their
respective
%windows
    manual_pressure_PANEL = uipanel('FontSize',12,...
        'BackgroundColor','white','FontName','Calibri',...
        'Visible','off','Position',[0.15 0.3 0.2 0.35]);
    manual_flow_PANEL = uipanel('FontSize',12,...
        'BackgroundColor','white','FontName','Calibri',...
        'Visible','off','Position',[0.15 0.3 0.2 0.35]);
    software_pressure_PANEL = uipanel('FontSize',12,...
        'BackgroundColor','white','FontName','Calibri',...
        'Visible','off','Position',[0.05 0.05 0.45 0.8]);
    software_flow_PANEL = uipanel('FontSize',12,...
        'BackgroundColor','white','FontName','Calibri',...
        'Visible','off','Position',[0.05 0.05 0.45 0.8]);

%CREATE THE AXES FOR DISPLAYING THE DATA
```

```
    ax = axes('Visible','off','Units','normalized','Position',[0.55
0.15 0.4 0.6],'XLim',[0 x_max],'YLim',[y_min y_max]);

%%

%DEFINE ALL CALLBACK FUNCTIONS
%Each of them needs as input the properties of the respective object

%For the pop-up menu
    function Menu(source,callbackdata)
        NumberMenu = source.Value;
        Names= source.String;
        NameMenu = Names{NumberMenu};
        %If user selects Main Menu in the pop-up menu, everything
should
        %dissappear and the main panel, its buttons and the welcome
text
        %should appear
        if (strcmp(NameMenu,'Main Menu')==1)
            %Everything involve in the main panel should appear
            txtWelcome.Visible = 'on';
            hp_manual.Visible = 'on';
            hp_software.Visible = 'on';
            %Everything refering to other windows must disapear
            manual_pressure_PANEL.Visible='off';
            manual_flow_PANEL.Visible='off';
            software_pressure_PANEL.Visible='off';
            software_flow_PANEL.Visible='off';
            cla(ax) %Clean the plot in the axes
            ax.Visible='off'; %Make them invisible
        %If user selects to manually control pressure in the pop-up
menu,
        %the main panel and other windows should disappear, moreover
        %fucntion created for creating the window that reads pressure
        %manually has to be executed
        elseif (strcmp(NameMenu,'Manual Control: Pressure')==1)
        %Everything not involved must disappear
            txtWelcome.Visible = 'off';
            hp_manual.Visible = 'off';
            hp_software.Visible = 'off';
            cla(ax)
            ax.Visible='off';
            manual_flow_PANEL.Visible='off';
            software_pressure_PANEL.Visible='off';
            software_flow_PANEL.Visible='off';
            %Function creating window must appear: it needs its panel
and
            %axes as input in order to locate all the specific
buttons.
            %Global structure handles its also an input
            Read_manual_pressure (manual_pressure_PANEL,handles,ax)
            manual_pressure_PANEL.Title='Read Pressure: ';
            manual_pressure_PANEL.Visible='on';
            ax.Visible='on';
        %A similar process is developed when other windows want to be
        %opened
        elseif (strcmp(NameMenu,'Manual Control: Flow')==1)
        %Everything not involved must disappear
            txtWelcome.Visible = 'off';
            hp_manual.Visible = 'off';
            hp_software.Visible = 'off';
            cla(ax)
            ax.Visible='off';
            manual_pressure_PANEL.Visible='off';
            software_pressure_PANEL.Visible='off';
            software_flow_PANEL.Visible='off';
        %Calling the function for the specific window
            Read_manual_flow (manual_flow_PANEL,handles,ax)
            manual_flow_PANEL.Title='Read Flow: ';
            manual_flow_PANEL.Visible='on';
            ax.Visible='on';
        elseif (strcmp(NameMenu,'Software Control: Pressure')==1)
        %Everything not involved must disappear
            txtWelcome.Visible = 'off';
            hp_manual.Visible = 'off';
            hp_software.Visible = 'off';
            cla(ax)
            ax.Visible='off';
            manual_pressure_PANEL.Visible='off';
            manual_flow_PANEL.Visible='off';
            software_flow_PANEL.Visible='off';
        %Calling the function for the specific window
            Read_software_pressure
(software_pressure_PANEL,handles,ax)
            software_pressure_PANEL.Title='Read Pressure: ';
            software_pressure_PANEL.Visible='on';
            ax.Visible='on';
        elseif (strcmp(NameMenu,'Software Control: Flow')==1)
        %Everything not involved must disappear
            txtWelcome.Visible = 'off';
            hp_manual.Visible = 'off';
            hp_software.Visible = 'off';
            cla(ax)
            ax.Visible='off';
            manual_pressure_PANEL.Visible='off';
            manual_flow_PANEL.Visible='off';
            software_pressure_PANEL.Visible='off';
        %Calling the function for the specific window
            Read_software_flow (software_flow_PANEL,handles,ax)
            software_flow_PANEL.Title='Read Flow: ';
            software_flow_PANEL.Visible='on';
            ax.Visible='on';
        end
    end
%A similar process is developed if the user pushes one of the butttons
%available at the main panel. The difference is that this buttons are
only
%available at the main window, while the pop-up menu is visible and
%accessible at any window.

% For the manual pressure reading button

    function manual_pressure(source,callbackdata)
    %Everything not involved must disappear.
        txtWelcome.Visible = 'off';
        hp_manual.Visible = 'off';
        hp_software.Visible = 'off';
    %Set the pop up menu to show that the user is in the manual
pressure window.
        popup.Value=2;
    %Function creating window must appear: it needs its panel and
    %axes as input in order to locate all the specific buttons.
    %Global structure handles its also an input
        Read_manual_pressure (manual_pressure_PANEL,handles,ax)
        manual_pressure_PANEL.Title='Read Pressure: ';
        manual_pressure_PANEL.Visible='on';
        ax.Visible='on';

    end

% The same process is developed for the other buttons
% For the manual flow reading button

    function manual_flow(source,callbackdata)
    %Everything not involved must disappear.
        txtWelcome.Visible = 'off';
        hp_manual.Visible = 'off';
        hp_software.Visible = 'off';
        popup.Value=3;
    %Calling the function for the specific window
        Read_manual_flow (manual_flow_PANEL,handles,ax)
        manual_flow_PANEL.Title='Read Flow: ';
        manual_flow_PANEL.Visible='on';
        ax.Visible='on';

    end

% For the software pressure reading button

    function software_pressure (source,callbackdata)
    %Everything not involved must disappear.
        txtWelcome.Visible = 'off';
        hp_manual.Visible = 'off';
        hp_software.Visible = 'off';
        popup.Value=4;
    %Calling the function for the specific window
        Read_software_pressure (software_pressure_PANEL,handles,ax)
        software_pressure_PANEL.Title='Read Pressure: ';
        software_pressure_PANEL.Visible='on';
        ax.Visible='on';


    end

% For the software flow reading button

    function software_flow (source,callbackdata)
    %Everything not involved must disappear.
        txtWelcome.Visible = 'off';
        hp_manual.Visible = 'off';
        hp_software.Visible = 'off';
        popup.Value=5;
    %Calling the function for the specific window
        Read_software_flow (software_flow_PANEL,handles,ax)
        software_pressure_PANEL.Title='Read Flow: ';
        software_flow_PANEL.Visible='on';
        ax.Visible='on';


    end
```

```
end
```

Manual Pressure Window

```
function Read_manual_pressure (manual_pressure_PANEL,handles,ax)
%
% READ_MANUAL_PRESSURE
% This function executes the window for specifically reading pressure
% INPUTS:
%   Manual_pressure_panel: panel created during
WK_Main_Interface_Reading,
%   all buttons will be located in this panel
%   Handles: global structure with some common data
%       handles.data=[x_max y_min y_max HR T Duty Delay];
%   ax: axes for ploting the data
%
%%

%DEFINING ALL THE ELEMENTS FOR THE PANEL

%CREATE READING START PUSHBUTTON
START= uicontrol('Parent',manual_pressure_PANEL,'Style', 'pushbutton',
'String', 'START','FontWeight','Bold',...
        'BackgroundColor',[1 1 1],'Units','normalized','Position',
[0.35 0.1 0.3 0.2],...
        'FontSize',10,'ForegroundColor',[0 0
1],'Callback',@START_btn_Callback );

%CREATE TEXT INSTRUCTIONS
SelectSamples =
uicontrol('Parent',manual_pressure_PANEL,'Style','text','ForegroundCol
or',[0 0 1],...
        'Units','normalized','Position',[0.05 0.7 0.9 0.2],...
        'FontWeight','Bold','String','Please select the desired number
of samples: ','BackgroundColor',[1 1 1]);

%CREATE EDIT BUTTON
x_max_edit =
uicontrol('Parent',manual_pressure_PANEL,'Style','Edit','Units','norma
lized',...
        'BackgroundColor',[0.9 0.9 0.99],'Position',[0.35 0.45 0.3
0.2],'String',handles.data(1),'Callback', @x_max_edit_Callback);

%%

%DEFINING CALLBACK FUNCTIONS

%STARTING TO READ SAMPLES FROM THE DEVICE
function START_btn_Callback(source,callbackdata)
    cla(ax)
    %As this window is the manual-control-pressure-reading window, the
    %corresponding function must be used:
    Read_Trial_Pressure(handles)
end

%READING THE NUMBER OF SAMPLES THE USER WANTS TO READ

function x_max_edit_Callback(source,callbackdata)
    %Obtain value from edit button
    handles.data(1)=str2double(get(source,'String'));
    cla(ax) %Cleaning axes to re-define them with the new x-axis limit
    ax.XLim=[0 handles.data(1)];
end


end
```

Manual Pressure Reading

```
function Read_Trial_Pressure(handles)

% Read_Trial_Pressure
% inputs:
    % Handles: global structure with general data in the form
    % handles.data=[x_max y_min y_max HR T Duty Delay];

%%

% OPENING THE PRESSURE SERIAL PORT FOR READING
u = serial('COM3','BaudRate',9600,'DataBits',8,'StopBits',1);
set(u,'InputBufferSize',5);
fopen(u);

% DEFINING THE LINE THAT WILL BE USED FOR THE PLOT
l1 = line(nan,nan,'Color','r','LineWidth',1);
title('Real Time Pressure')
xlabel('Samples')
ylabel('Pressure [mmHg]')
grid on
hold on
%%

% DEFINING THE TIMERS:

%TIMER FOR READING PRESSURE DATA
read_timer = timer('TimerFcn',@mycallback_read_timer
,'BusyMode','drop',...

'StartDelay',1,'Period',0.005,'ExecutionMode','fixedSpacing');

%START THE TIMER
start(read_timer)

%INITIALIZE VARIABLES
rawpressure=[];
rawpressure(1)=1;
real_pressure=[];

%MAIN LOOOP:it will constantly execute until stop condition
while(1)
%DECODING SECTION OF THE CODE
    positions=find(rawpressure==192 | rawpressure==196);
    positions=positions+1;
    real_pressure=rawpressure(positions(1:(length(positions)-1)));
    x100=find(real_pressure>=100);
    x28=find(real_pressure<100);
    real_pressure(x28)=real_pressure(x28)+28;
    real_pressure(x100)=real_pressure(x100)-100;

% PLOTTING SECTION OF THE CODE
    real_pressure=real_pressure+70; %Calibrate
    x=linspace(0,length(real_pressure),length(real_pressure));
    set(l1,'YData',real_pressure(1:length(real_pressure)),'XData',x);
    drawnow

%STOP CONDITION: if enough samples have been read, stop
if  (length(real_pressure)>=handles.data(1))
%Stop and delete timers, close serial ports
    stop(read_timer)
    delete(read_timer)
    fclose(u)
    delete(u)
%Save data in a file
    fileID = fopen('PressureData.txt','w');
    fprintf(fileID,'%6s\n','Real Time Pressure');
    fprintf(fileID,'%6.2f\n',real_pressure);
    fclose(fileID);
    break
end
end
%%
%DEFINE CALLBACK FUNCTIONS

%Function reading data
function mycallback_read_timer (obj, event)
if (u.BytesAvailable~=0)
    rawpressure=[rawpressure fread(u,u.BytesAvailable)'];
end
end

end
```

Manual Flow Window

```
function Read_manual_flow (manual_flow_PANEL,handles,ax)
%
% READ_FLOW_PRESSURE
% This function executes the window for specifically reading pressure
% INPUTS:
%   Manual_flow_panel: panel created during WK_Main_Interface_Reading,
%   all buttons will be located in this panel
%   Handles: global structure with some common data
%       handles.data=[x_max y_min y_max HR T Duty Delay];
%   ax: axes for ploting the data
%
%%

%%

%DEFINING ALL THE ELEMENTS FOR THE PANEL

%CREATE READING START PUSHBUTTON
START= uicontrol('Parent',manual_flow_PANEL,'Style', 'pushbutton',
'String', 'START','FontWeight','Bold',...
        'BackgroundColor',[1 1 1],'Units','normalized','Position',
[0.35 0.1 0.3 0.2],...
```

```
        'FontSize',10,'ForegroundColor',[0 0
1],'Callback',@START_btn_Callback );

%CREATE TEXT INSTRUCTIONS
SelectSamples =
uicontrol('Parent',manual_flow_PANEL,'Style','text','ForegroundColor',
[0 0 1],...
        'Units','normalized','Position',[0.05 0.7 0.9 0.2],...
        'FontWeight','Bold','String','Please select the desired number
of samples: ','BackgroundColor',[1 1 1]);

%CREATE EDIT BUTTON
x_max_edit =
uicontrol('Parent',manual_flow_PANEL,'Style','Edit','Units','normalize
d',...
        'BackgroundColor',[0.9 0.9 0.99],'Position',[0.35 0.45 0.3
0.2],'String',handles.data(1),'Callback', @x_max_edit_Callback);

%%

%DEFINING CALLBACK FUNCTIONS

%STARTING TO READ SAMPLES FROM THE DEVICE
function START_btn_Callback(source,callbackdata)
    cla(ax)
    %As this window is the manual-control-flow-reading window, the
    %corresponding function must be used:
    Read_Trial_Flow(handles)
end

%READING THE NUMBER OF SAMPLES THE USER WANTS TO READ
function x_max_edit_Callback(source,callbackdata)
    %Obtain value from edit button
    handles.data(1)=str2double(get(source,'String'));
    cla(ax) %Cleaning axes to re-define them with the new x-axis limit
    ax.XLim=[0 handles.data(1)];
end
end
```

Manual Flow Reading

```
function Read_Trial_Flow(handles)
%
% Read_Trial_Flow
% inputs:
    % Handles: global structure with general data in the form
    % handles.data=[x_max y_min y_max HR T Duty Delay];

%%

% OPENING THE FLOW SERIAL PORT FOR READING
s = serial('COM9','BaudRate',38400,'DataBits',8,'StopBits',1);
set(s,'InputBufferSize',55); %Con 55 me lee una linea=2 dato util
fopen(s);

% DEFINING THE LINE THAT WILL BE USED FOR THE PLOT
l1 = line(nan,nan,'Color','r','LineWidth',1);
title('Real Time Flow')
xlabel('Samples')
```

```
ylabel('Flow [ml/s]')
grid on
hold on
%%

% DEFINING THE TIMERS:

%TIMER FOR READING FLOW DATA
read_timer = timer('TimerFcn',@mycallback_read_timer
,'BusyMode','drop',...

'StartDelay',1,'Period',0.01,'ExecutionMode','fixedSpacing');
%TIMER FOR DECODING FLOW DATA
paint_timer= timer('TimerFcn',@mycallback_paint_timer
,'BusyMode','drop',...

'StartDelay',1,'Period',0.1,'ExecutionMode','fixedRate');

%START THE TIMER
start(read_timer)
start(paint_timer)

%INITIALIZE VARIABLES
rawflow=struct('outstr',{});
rawflow(1).outstr='00 00 +0000 +0000 +000 +0 +0000 +0000 +000';
rawflow_vector=[];
cont_read=1;
ind=1;
%MAIN LOOOP:it will constantly execute until stop condition
while(1)
% PLOTTING SECTION OF THE CODE
    x=linspace(0,length(rawflow_vector),length(rawflow_vector));

set(l1,'YData',rawflow_vector(1:length(rawflow_vector)),'XData',x);
    drawnow
    refreshdata

%STOP CONDITION: if enough samples have been read, stop
if (length(rawflow_vector)>=x_max)
%Stop and delete timers, close serial ports
    stop(read_timer)
    delete(read_timer)
    stop(paint_timer)
    delete(paint_timer)
    fclose(s)
    delete(s)
%Save data in a file
    fileID = fopen('FlowData.txt','w');
    fprintf(fileID,'%6s\n','Real Time Flow');
    fprintf(fileID,'%6.2f\n',rawflow_vector);
    fclose(fileID);
    break
end
end
%%
%DEFINE CALLBACK FUNCTIONS

%Function reading data
function mycallback_read_timer (obj, event)
    if (s.BytesAvailable~=0)
```

```
        cont_read=cont_read+1;
        rawflow(cont_read).outstr=fgets(s);
    end

end

%DECODING SECTION OF THE CODE
function mycallback_paint_timer (obj, event)
%Obtain meaningful value from a string and store it in a vector
    for i=ind:numel(rawflow)
        if (numel(rawflow(i).outstr)==55)
            if
(((strcmp(rawflow(i).outstr(13),'+'))==0)&&((strcmp(rawflow(i).outstr(
13),'-'))==0)&&((strcmp(rawflow(i).outstr(13),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(13:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(14),'+'))==0)&&((strcmp(rawflow(i).outstr(
14),'-'))==0)&&((strcmp(rawflow(i).outstr(14),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(14:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(15),'+'))==0)&&((strcmp(rawflow(i).outstr(
15),'-'))==0)&&((strcmp(rawflow(i).outstr(15),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(15:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(16),'+'))==0)&&((strcmp(rawflow(i).outstr(
16),'-'))==0)&&((strcmp(rawflow(i).outstr(16),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(16))/60;
            end
        else
            rawflow_vector(i)=0;
        end
    end
   ind=length(rawflow_vector);
end

end
```

Software Pressure Window

```
function Read_software_pressure (software_pressure_PANEL,handles,ax)
%
% READ_SOFTWARE PRESSURE
% This function executes the window for specifically reading pressure
% INPUTS:
%   Software_pressure_panel: panel created during
WK_Main_Interface_Reading,
%   all buttons will be located in this panel
%   Handles: global structure with some common data
%       handles.data=[x_max y_min y_max HR T Duty Delay];
%   ax: axes for ploting the data
%
%%

%DEFINING ALL THE ELEMENTS FOR THE PANEL

%CREATE READING START PUSHBUTTON
START= uicontrol('Parent',software_pressure_PANEL,'Style',
'pushbutton', 'String', 'START','FontWeight','Bold',...
```

```
        'BackgroundColor',[1 1 1],'Units','normalized','Position',
[0.4 0.3 0.2 0.1],...
        'FontSize',10,'ForegroundColor',[0 0
1],'Callback',@START_btn_Callback );

%CREATE TEXT INSTRUCTIONS
SelectSamples =
uicontrol('Parent',software_pressure_PANEL,'Style','text','ForegroundC
olor',[0 0 1],...
        'Units','normalized','Position',[0.05 0.5 0.9 0.1],...
        'FontWeight','Bold','String','Please select the desired number
of samples: ','BackgroundColor',[1 1 1]);

%CREATE EDIT BUTTON
x_max_edit =
uicontrol('Parent',software_pressure_PANEL,'Style','Edit','Units','nor
malized',...
        'BackgroundColor',[0.9 0.9 0.99],'Position',[0.4 0.45 0.2
0.1],'String',handles.data(1),'Callback', @x_max_edit_Callback);

%CREATE TEXT INSTRUCTIONS
SelectHR =
uicontrol('Parent',software_pressure_PANEL,'Style','text','ForegroundC
olor',[0 0 1],...
        'Units','normalized','Position',[0.05 0.85 0.9 0.1],...
        'FontWeight','Bold','String','Please select a heart rate and
duty cycle: ','BackgroundColor',[1 1 1]);

%CREATE PANNEL FOR BUTTONS SELECTING HEART CYCLE PARAMETERS
bg =
uibuttongroup('Parent',software_pressure_PANEL,'Visible','on','Units',
'normalized','Position',[0.05 0.65 0.9 0.25],...
                  'BackgroundColor',[0.9 0.9
0.99],'SelectionChangedFcn',@b_selection);

%CREATE BUTTONS SELECTING HEART CYCLE PARAMETERS
r1 =
uicontrol(bg,'Style','radiobutton','String','HR=60beats/min','FontSize
',9,'FontWeight','Bold',...
                  'Position',[50 50 120 50],'BackgroundColor',[0.9 0.9
0.99],'HandleVisibility','off');

r2 =
uicontrol(bg,'Style','radiobutton','String','HR=120beats/min','FontSiz
e',9,'FontWeight','Bold',...
                  'Position',[220 50 120 50],'BackgroundColor',[0.9
0.9 0.99],'HandleVisibility','off');

r3 =
uicontrol(bg,'Style','radiobutton','String','Duty=0.8/0.2','FontSize',
9,'FontWeight','Bold',...
                  'Position',[390 50 120 50],'BackgroundColor',[0.9
0.9 0.99],'HandleVisibility','off');

%%

%DEFINING CALLBACK FUNCTIONS

%DEFINING CODE EXECUTED BY HEART PARAMETER BUTTONS
```

```
function b_selection(source,callbackdata)
   callbackdata.NewValue.String
   %If first button is selected
   if (strcmp(callbackdata.NewValue.String,'HR=120beats/min')==1)
       panel.Visible='on';
       %They modify global variables that will be used for sending
pulses
        handles.data(4)=120; %Heart rate
        handles.data(5)=60/handles.data(4); %Heart period
        handles.data(6)=2/5; % Fraction of period occupied by systole
        handles.data(7)=handles.data(6)*handles.data(5); %Time
occupied by
        %systole
    %If second button is selected
    elseif (strcmp(callbackdata.NewValue.String,'HR=60beats/min')==1)
       panel.Visible='on';
        handles.data(4)=60;
        handles.data(5)=60/handles.data(4);
        handles.data(6)=2/5;
        handles.data(7)=handles.data(6)*handles.data(5);
   %If third button is selected
   elseif (strcmp(callbackdata.NewValue.String,'Duty=0.8/0.2')==1)
       panel.Visible='on';
        handles.data(4)=60;
        handles.data(5)=60/handles.data(4);
        handles.data(6)=8/10;
        handles.data(7)=handles.data(6)*handles.data(5);
   end

end

%READING THE NUMBER OF SAMPLES THE USER WANTS TO READ

function x_max_edit_Callback(source,callbackdata)
    %Obtain value from edit button
    handles.data(1)=str2double(get(source,'String'));
    cla(ax) %Cleaning axes to re-define them with the new x-axis limit
    ax.XLim=[0 handles.data(1)];
end

%STARTING TO READ SAMPLES FROM THE DEVICE

function START_btn_Callback(source,callbackdata)
    cla(ax)
    %As this window is the software-control-pressure-reading window,
the
    %corresponding function must be used:
    Read_Trial_Pressure_AND_Pulses(handles,software_pressure_PANEL)
    %It needs as inputs:
    % handles: to determine the different heart cycle
    %parameters to send th pulses AND the x-axis limit to know the
number
    %of samples to read before stopping.
    % It also needs the current panel to add a new display
end

end
```

Software Pressure Read

```
function Read_Trial_Pressure_AND_Pulses(handles,panel)

% Read_Trial_Pressure_AND_Pulses
% inputs:
    % Handles: global structure with general data in the form
    % handles.data=[x_max y_min y_max HR T Duty Delay];
    % Panel: the panel used in the current window is needed to add a
new display

%%

% DEFINING THE DISPLAYS THAT WILL HIGHLIGHT DEPENDING IF THE SYSTEM IS
% WORKING ON SYSTOLE OR DIASTOLE
txt_s = uicontrol('Parent',panel,'Style','text',...
        'Units','normalized','Position',[0.35 0.1 0.1 0.1],...
        'FontWeight','Bold','String','Systole','BackgroundColor',[1 1
1]);
txt_d = uicontrol('Parent',panel,'Style','text',...
        'Units','normalized','Position',[0.55 0.1 0.1 0.1],...
        'FontWeight','Bold','String','Diastole','BackgroundColor',[1 1
1]);

% OPENING THE FLOW SERIAL PORT FOR SENDING PULSES
v = serial('COM9','BaudRate',38400,'DataBits',8,'StopBits',1);
set(v,'RequestToSend','off')
set(v,'DataTerminalReady','off')
fopen(v);
% OPENING THE PRESSURE SERIAL PORT FOR READING
u = serial('COM3','BaudRate',9600,'DataBits',8,'StopBits',1);
set(u,'InputBufferSize',5);
fopen(u);

% DEFINING THE LINE THAT WILL BE USED FOR THE PLOT

l1 = line(nan,nan,'Color','r','LineWidth',1);
title('Real Time Pressure')
xlabel('Samples')
ylabel('Pressure [mmHg]')
grid on
hold on

%%

% DEFINING THE TIMERS:

%TIMER FOR READING PRESSURE DATA
read_timer = timer('TimerFcn',@mycallback_read_timer
,'BusyMode','drop',...

'StartDelay',1,'Period',0.001,'ExecutionMode','fixedSpacing');

%TIMER FOR SENDING SYSTOLE PULSE
t = timer('TimerFcn', @mycallback_t,'BusyMode','drop',...

'StartDelay',0,'Period',handles.data(5),'ExecutionMode','fixedSpacing'
);

%TIMER FOR SENDING DIASTOLE PULSE
```

```
d = timer('TimerFcn', @mycallback_d,'BusyMode','drop',...

'StartDelay',handles.data(7),'Period',handles.data(5),'ExecutionMode',
'fixedSpacing');

%START THE TIMERS
start(t)
start(d)
start(read_timer)

%INITIALIZE VARIABLES
rawpressure=[];
rawpressure(1)=1;
real_pressure=[];

%MAIN LOOOP:it will constantly execute until stop condition

while(1)
%DECODING SECTION OF THE CODE
%Find only meaningful values indicators:
    positions=find(rawpressure==192 | rawpressure==196);
%Useful values will be the ones after the previous calculated
    positions=positions+1;
    real_pressure=rawpressure(positions(1:(length(positions)-1)));
%Remove +100mmHg offset
    x100=find(real_pressure>=100);
%Recover values over 128
    x28=find(real_pressure<100);
    real_pressure(x28)=real_pressure(x28)+28;
    real_pressure(x100)=real_pressure(x100)-100;
    real_pressure=real_pressure+70; %Calibrating

% PLOTTING SECTION OF THE CODE
    x=linspace(0,length(real_pressure),length(real_pressure));
    set(l1,'YData',real_pressure(1:length(real_pressure)),'XData',x);
    drawnow

%STOP CONDITION: if enough samples have been read, stop
if (length(real_pressure)>=handles.data(1))
%Stop and delete timers, close serial ports
    stop(read_timer)
    delete(read_timer)
    stop(t)
    stop(d)
    delete(t)
    delete(d)
    fclose(v)
    delete(v)
    fclose(u)
    delete(u)
%Save data in a file
    fileID = fopen('PressureData.txt','w');
    fprintf(fileID,'%6s\n','Real Time Pressure');
    fprintf(fileID,'%6.2f\n',real_pressure);
    fclose(fileID);
    break
end
end
%%
%DEFINE CALLBACK FUNCTIONS
```

```
%Function reading data
function mycallback_read_timer (obj, event)
    if (u.BytesAvailable~=0)
    rawpressure=[rawpressure fread(u,u.BytesAvailable)'];
    end
end

%Function sending systole pulse (set DTR HIGH)
function mycallback_t (obj, event, string_arg)
    set(v,'DataTerminalReady','on')%on=high=pressure
    %Modify highlighted display
    txt_s.BackgroundColor=[1 0 0];
    txt_d.BackgroundColor=[1 1 1];
end

%Function sending diastole pulse (set DTR LOW)
function mycallback_d (obj, event, string_arg)
    set(v,'DataTerminalReady','off')
    %Modify highlighted display
    txt_d.BackgroundColor=[0 1 0];
    txt_s.BackgroundColor=[1 1 1];
end

end
```

Software Flow Window

```
function Read_software_flow (software_flow_PANEL,handles,ax)
%
% READ_SOFTWARE FLOW
% This function executes the window for specifically reading pressure
% INPUTS:
%   Software_flow_panel: panel created during
WK_Main_Interface_Reading,
%   all buttons will be located in this panel
%   Handles: global structure with some common data
%       handles.data=[x_max y_min y_max HR T Duty Delay];
%   ax: axes for ploting the data
%
%%

%DEFINING ALL THE ELEMENTS FOR THE PANEL
%CREATE READING START PUSHBUTTON
START= uicontrol('Parent',software_flow_PANEL,'Style', 'pushbutton',
'String', 'START','FontWeight','Bold',...
        'BackgroundColor',[1 1 1],'Units','normalized','Position',
[0.4 0.3 0.2 0.1],...
        'FontSize',10,'ForegroundColor',[0 0
1],'Callback',@START_btn_Callback );

%CREATE TEXT INSTRUCTIONS
SelectSamples =
uicontrol('Parent',software_flow_PANEL,'Style','text','ForegroundColor
',[0 0 1],...
        'Units','normalized','Position',[0.05 0.5 0.9 0.1],...
        'FontWeight','Bold','String','Please select the desired number
of samples: ','BackgroundColor',[1 1 1]);
```

```
%CREATE EDIT BUTTON
x_max_edit =
uicontrol('Parent',software_flow_PANEL,'Style','Edit','Units','normali
zed',...
        'BackgroundColor',[0.9 0.9 0.99],'Position',[0.4 0.45 0.2
0.1],'String',handles.data(1),'Callback', @x_max_edit_Callback);

%CREATE TEXT INSTRUCTIONS
SelectHR =
uicontrol('Parent',software_flow_PANEL,'Style','text','ForegroundColor
',[0 0 1],...
        'Units','normalized','Position',[0.05 0.85 0.9 0.1],...
        'FontWeight','Bold','String','Please select a heart rate and
duty cycle: ','BackgroundColor',[1 1 1]);

%CREATE PANNEL FOR BUTTONS SELECTING HEART CYCLE PARAMETERS
bg =
uibuttongroup('Parent',software_flow_PANEL,'Visible','on','Units','nor
malized','Position',[0.05 0.65 0.9 0.25],...
                  'BackgroundColor',[0.9 0.9
0.99],'SelectionChangedFcn',@b_selection);

%CREATE BUTTONS SELECTING HEART CYCLE PARAMETERS
r1 =
uicontrol(bg,'Style','radiobutton','String','HR=60beats/min','FontSize
',9,'FontWeight','Bold',...
                  'Position',[50 50 120 50],'BackgroundColor',[0.9 0.9
0.99],'HandleVisibility','off');

r2 =
uicontrol(bg,'Style','radiobutton','String','HR=120beats/min','FontSiz
e',9,'FontWeight','Bold',...
                  'Position',[220 50 120 50],'BackgroundColor',[0.9
0.9 0.99],'HandleVisibility','off');

r3 =
uicontrol(bg,'Style','radiobutton','String','Duty=0.8/0.2','FontSize',
9,'FontWeight','Bold',...
                  'Position',[390 50 120 50],'BackgroundColor',[0.9
0.9 0.99],'HandleVisibility','off');

%%
%DEFINING CALLBACK FUNCTIONS

%DEFINING CODE EXECUTED BY HEART PARAMETER BUTTONS
function b_selection(source,callbackdata)
   callbackdata.NewValue.String
   %If first button is selected
   if (strcmp(callbackdata.NewValue.String,'HR=120beats/min')==1)
       panel.Visible='on';
       %They modify global variables that will be used for sending
pulses
        handles.data(4)=120; %Heart rate
        handles.data(5)=60/handles.data(4); %Heart period
        handles.data(6)=2/5; % Fraction of period occupied by systole
        handles.data(7)=handles.data(6)*handles.data(5); %Time
occupied by
        %systole
    %If second button is selected
    elseif (strcmp(callbackdata.NewValue.String,'HR=60beats/min')==1)
```

```
        panel.Visible='on';
         handles.data(4)=60;
         handles.data(5)=60/handles.data(4);
         handles.data(6)=2/5;
         handles.data(7)=handles.data(6)*handles.data(5);
    elseif (strcmp(callbackdata.NewValue.String,'Duty=0.8/0.2')==1)
        panel.Visible='on';
         handles.data(4)=60;
         handles.data(5)=60/handles.data(4);
         handles.data(6)=8/10;
         handles.data(7)=handles.data(6)*handles.data(5);
    end

end

%READING THE NUMBER OF SAMPLES THE USER WANTS TO READ

function x_max_edit_Callback(source,callbackdata)
    %Obtain value from edit button
    handles.data(1)=str2double(get(source,'String'));
    cla(ax) %Cleaning axes to re-define them with the new x-axis limit
    ax.XLim=[0 handles.data(1)];
end

%STARTING TO READ SAMPLES FROM THE DEVICE
function START_btn_Callback(source,callbackdata)
    cla(ax)
    %As this window is the software-control-flow-reading window, the
    %corresponding function must be used:
    Read_Trial_Flow_AND_Pulses(handles,software_flow_PANEL)
    %It needs as inputs:
    %handles: to determine the different heart cycle
    %parameters to send th pulses AND the x-axis limit to know the
number
    %of samples to read before stopping.
    % It also needs the current panel to add a new display
end
end
```

<u>**Software Flow Read**</u>

```
function Read_Trial_Flow_AND_Pulses(handles,panel)
%
%Read_Trial_Flow_AND_Pulses
% inputs:
    % Handles: global structure with general data in the form
    % handles.data=[x_max y_min y_max HR T Duty Delay];
    % Panel: the panel used in the current window is needed to add a
new display

%%

% DEFINING THE DISPLAYS THAT WILL HIGHLIGHT DEPENDING IF THE SYSTEM IS
% WORKING ON SYSTOLE OR DIASTOLE
txt_s = uicontrol('Parent',panel,'Style','text',...
        'Units','normalized','Position',[0.7 0.1 0.05 0.05],...
        'String','Systole','BackgroundColor',[1 1 1]);
txt_d = uicontrol('Parent',panel,'Style','text',...
        'Units','normalized','Position',[0.8 0.1 0.05 0.05],...
        'String','Diastole','BackgroundColor',[1 1 1]);

% OPENING THE FLOW SERIAL PROT FOR READING
s = serial('COM9','BaudRate',38400,'DataBits',8,'StopBits',1);
set(s,'InputBufferSize',55); %With 55 bytes it reads an entire line
fopen(s);

% OPENING THE PRESSURE SERIAL PROT FOR SENDING PULSES
v = serial('COM3','BaudRate',9600,'DataBits',8,'StopBits',1);
set(v,'DataTerminalReady','off')
fopen(v)

% DEFINING THE LINE THAT WILL BE USED FOR THE PLOT
l1 = line(nan,nan,'Color','r','LineWidth',1);
title('Real Time Flow')
xlabel('Samples')
ylabel('Flow [ml/s]')
grid on
hold on
%%

% DEFINING THE TIMERS:

%TIMER FOR READING FLOW DATA
read_timer = timer('TimerFcn',@mycallback_read_timer
,'BusyMode','drop',...

'StartDelay',2,'Period',0.001,'ExecutionMode','fixedSpacing');
%TIMER FOR DECODING DATA
paint_timer= timer('TimerFcn',@mycallback_paint_timer
,'BusyMode','drop',...

'StartDelay',2,'Period',0.1,'ExecutionMode','fixedRate');

%TIMER FOR SENDING SYSTOLE PULSE
t = timer('TimerFcn', @mycallback_t,'BusyMode','drop',...

'StartDelay',0,'Period',handles.data(5),'ExecutionMode','fixedSpacing'
);

%TIMER FOR SENDING DIASTOLE PULSE
d = timer('TimerFcn', @mycallback_d,'BusyMode','drop',...

'StartDelay',handles.data(7),'Period',handles.data(5),'ExecutionMode',
'fixedSpacing');

%START THE TIMERS
start(t)
start(d)
start(read_timer)
start(paint_timer)

%INITIALIZE VARIABLES
rawflow=struct('outstr',{});
rawflow(1).outstr='00 00 +0000 +0000 +000 +0 +0000 +0000 +000';
rawflow_vector=[];
cont_read=1;
ind=1;
```

```
%MAIN LOOOP:it will constantly execute until stop condition

while(1)

% PLOTTING SECTION OF THE CODE
    x=linspace(0,length(rawflow_vector),length(rawflow_vector));

set(l1,'YData',rawflow_vector(1:length(rawflow_vector)),'XData',x);
    drawnow
    refreshdata

%STOP CONDITION: if enough samples have been read, stop
if (length(rawflow_vector)>=handles.data(1))
%Stop and delete timers, close serial ports
    set(v,'DataTerminalReady','on')
    stop(read_timer)
    delete(read_timer)
    stop(paint_timer)
    delete(paint_timer)
    stop(t)
    stop(d)
    delete(t)
    delete(d)
    set(v,'DataTerminalReady','on')
    pause(1)
    fclose(v)
    delete(v)
    fclose(s)
    delete(s)
%Save data in a file
    fileID = fopen('FlowData.txt','w');
    fprintf(fileID,'%6s\n','Real Time Flow');
    fprintf(fileID,'%6.2f\n',rawflow_vector);
    fclose(fileID);
    break
end
end

%%
%DEFINE CALLBACK FUNCTIONS

%Function reading data
function mycallback_read_timer (obj, event)
    if (s.BytesAvailable~=0)
        cont_read=cont_read+1;
        rawflow(cont_read).outstr=fgets(s);
    end
end

%DECODING SECTION OF THE CODE
function mycallback_paint_timer (obj, event)
%Obtain meaningful value from a string and store it in a vector
    for i=ind:numel(rawflow)
        if (numel(rawflow(i).outstr)==55)
            if
(((strcmp(rawflow(i).outstr(13),'+'))==0)&&((strcmp(rawflow(i).outstr(
13),'-'))==0)&&((strcmp(rawflow(i).outstr(13),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(13:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(14),'+'))==0)&&((strcmp(rawflow(i).outstr(
14),'-'))==0)&&((strcmp(rawflow(i).outstr(14),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(14:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(15),'+'))==0)&&((strcmp(rawflow(i).outstr(
15),'-'))==0)&&((strcmp(rawflow(i).outstr(15),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(15:16))/60;
            elseif
(((strcmp(rawflow(i).outstr(16),'+'))==0)&&((strcmp(rawflow(i).outstr(
16),'-'))==0)&&((strcmp(rawflow(i).outstr(16),' '))==0))
            rawflow_vector(i)=str2num(rawflow(i).outstr(16))/60;
            end
        else
            rawflow_vector(i)=0;
        end
    end
   ind=length(rawflow_vector);
end

%Function sending systole pulse (set DTR HIGH)
function mycallback_t (obj, event, string_arg)
    set(v,'DataTerminalReady','on')
    %Modify highlighted display
    txt_s.BackgroundColor=[1 0 0];
    txt_d.BackgroundColor=[1 1 1];
end
function mycallback_d (obj, event, string_arg)
    %disp('diastole')
    set(v,'DataTerminalReady','off')%off=low=vacuum
    %Modify highlighted display
    txt_d.BackgroundColor=[0 1 0];
    txt_s.BackgroundColor=[1 1 1];
end

end
```

SIMULATION INTERFACE

Due to the excesive length of all the functions only the main window, and the window and calculations for the 2-element model are displayed as routine examples.

Main Window

```
function WK_Main_Interface
%
% WK_MAIN_INTERFACE
%
% This function opens the main window of the simulation interface and
allows
% the user to observe the different Windkessel models

%%
%Define Common Variables
Heart_Rate=72;
Time_Systole= (2/5)*(60/Heart_Rate);
Max_Flow=5e-4; %[m^3/s]=4.98 l/min
Number_Cycles=1;
```

```
%2WK data variables
WK2_Rp=133.28e6; %[kg/s*m^4]
WK2_C=0.75e-8;  %[s^2*m^4/kg]

%3WK data variables
WK3_Rp=133.28e6; %[kg/s*m^4]
WK3_C=0.75e-8;  %[s^2*m^4/kg]
WK3_Ra=6.66e6;   %[kg/s*m^4]

%4WK data variables
WK4_Rp=133.28e6; %[kg/s*m^4]
WK4_C=0.75e-8;  %[s^2*m^4/kg]
WK4_Ra=6.66e6;   %[kg/s*m^4]
WK4_L=6.66e5;   %[kg/m4]

%Conversion Parameters
Sec=60;
SItoFlow=10^6;%[m^3/s*beat]
FlowtoSI=1/(10^6);%[ml/s*beat]
SItoHRU=760/(6*1.013*10^9);
HRUtoSI=(6*1.013*10^9)/760; %[mmHg*min/L]
SItoHCU=(1.013*10^11)/760;
HCUtoSI=760/(1.013*10^11); %[mL/mmHg]
SItoHLU=760/(1.013*10^8);
HLUtoSI=(1.013*10^8)/760; %[mmHg*s^2/L]

% Create structure handles that will store data to calculate pressure
% It will have three fields:
%
% 1. handles.dataOriginal= containing always the original data to
reset
% parameters when default buttons are pressed
%
% 2. handles.data= containing always initially the original data,
% but changing as parameters are modified
%
% 3. handles.additional= containing conversion parameters

handles.dataOriginal=[Heart_Rate Time_Systole Max_Flow Number_Cycles
WK2_Rp WK2_C WK3_Rp WK3_C WK3_Ra WK4_Rp WK4_C WK4_Ra WK4_L];
handles.data=handles.dataOriginal;
handles.additional=[Sec SItoFlow FlowtoSI SItoHRU HRUtoSI SItoHCU
HCUtoSI SItoHLU HLUtoSI];
%In this way handles will be called for executig functions WK_2,WK_3
and WK_4

%%
%Create Window Components:
%CREATE MAIN FIGURE
f =
figure('Visible','off','Toolbar','none','Menubar','none','Color',[0.85
0.99 0.85]);

%CREATE WELCOME TEXT
txtWelcome =
uicontrol('Style','text','Units','normalized','FontWeight','Bold',...
    'Position',[0.25 0.8 0.5 0.1],'FontSize',15,...
    'String','WELCOME', 'FontName','Century','BackgroundColor',[0.85
0.99 0.85]);
%CREATE POP UP MENU
```

```
popup = uicontrol('Style',
'popup','Units','normalized','ForegroundColor',[0 0
1],'FontWeight','Bold',...
   'String', {'Main Menu','2 elements WK','3 elements WK','4 elements
WK'},'FontName','Calibri',...
   'FontSize',10,'Position', [0.05 0.85 0.2 0.1],...
   'Callback', @Menu);
%CREATE MAIN PANEL
hp = uipanel('Title','Please, select a model: ','FontSize',15,...

'BackgroundColor','white','FontName','Calibri','FontWeight','Bold',...
     'Position',[0.25 0.1 0.5 0.6]);
%CREATE 2WK BUTTON
btn2WK = uicontrol('Parent',hp,'Style',
'pushbutton','Units','normalized','FontWeight','Bold',...
    'String', '2 Elements Windkessel','Position',[0.25 0.65 0.5
0.2],'FontSize',11,...
    'FontName','Calibri','BackgroundColor',[0.8 0.8 0.99],'Callback',
@WK_2BTN);
%CREATE 3WK BUTTON
btn3WK = uicontrol('Parent',hp,'Style',
'pushbutton','Units','normalized','FontWeight','Bold',...
    'String', '3 Elements Windkessel','Position', [0.25 0.4 0.5
0.2],'FontSize',11,...
    'FontName','Calibri','BackgroundColor',[0.8 0.9 0.99],'Callback',
@WK_3BTN);
%CREATE 4WK BUTTON
btn4WK= uicontrol('Parent',hp,'Style',
'pushbutton','Units','normalized','FontWeight','Bold',...
    'String', '4 Elements Windkessel','Position',[0.25 0.15 0.5 0.2]
,'FontSize',11,...
    'FontName','Calibri','BackgroundColor',[0.99 0.8 0.8],'Callback',
@WK_4BTN);
%Make everything visible
f.Visible = 'on';

% CREATE THE SPECIFIC PANELS FOR EACH OF THE MODELS USED:only visible
in their respective
% windows
WK2_PANEL = uipanel('FontSize',14,'FontWeight','Bold',...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.05 0.05 0.45 0.8]);
WK3_PANEL = uipanel('FontSize',14,'FontWeight','Bold',...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.05 0.05 0.45 0.8]);
WK4_PANEL = uipanel('FontSize',14,'FontWeight','Bold',...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.05 0.05 0.45 0.8]);

%DEFINE THE AXES WHERE THE SIMULATED VARIABLES WILL BE PLOTTED
ax = axes('Visible','off','Units','normalized','Position',[0.55 0.15
0.4 0.6]);

%%
%DEFINE CALLBACK FUNCTIONS
%Each of them needs as input the properties of the respective object
%For the pop-up menu
function Menu(source,callbackdata)
    NumberMenu = source.Value;
    Names= source.String;
    NameMenu = Names{NumberMenu};
    %If user selects Main Menu in the pop-up menu, everything should
    %dissappear and the main panel, its buttons and the welcome text
    %should appear
    if (strcmp(NameMenu,'Main Menu')==1)
        %Everything involved in the main panel should appear
        txtWelcome.Visible = 'on';
        hp.Visible = 'on';
        %Everything refering to other windows must disapear
        WK2_PANEL.Visible='off';
        WK3_PANEL.Visible='off';
        WK4_PANEL.Visible='off';
        cla(ax)
        ax.Visible='off';
    %If the user selects a different option, the corresponding window
must
    %be displayed.
    elseif (strcmp(NameMenu,'2 elements WK')==1)
        %Everything not involved in the window must disappear
        txtWelcome.Visible = 'off';
        hp.Visible = 'off';
        cla(ax)
        ax.Visible='off';
        WK2_PANEL.Visible='off';
        WK3_PANEL.Visible='off';
        WK4_PANEL.Visible='off';
        %Function creating window must appear: it needs its panel and
        %the global variables as inputs
        WK_2(WK2_PANEL,handles)
        WK2_PANEL.Title='Modify the desired 2 WK Parameters: ';
        WK2_PANEL.Visible='on';
        ax.Visible='on';
    elseif (strcmp(NameMenu,'3 elements WK')==1)
        %Everything not involved in the window must disappear
        txtWelcome.Visible = 'off';
        hp.Visible = 'off';
        %Call WK_3
        cla(ax)
        ax.Visible='off';
        WK2_PANEL.Visible='off';
        WK3_PANEL.Visible='off';
        WK4_PANEL.Visible='off';
        %Function creating window must appear: it needs its panel and
        %the global variables as inputs
        WK_3(WK3_PANEL,handles)
        WK3_PANEL.Title='Modify the desired 3 WK Parameters: ';
        WK3_PANEL.Visible='on';
        ax.Visible='on';
    elseif (strcmp(NameMenu,'4 elements WK')==1)
        %Everything not involved in the window must disappear
        txtWelcome.Visible = 'off';
        hp.Visible = 'off';
        cla(ax)
        ax.Visible='off';
        WK2_PANEL.Visible='off';
        WK3_PANEL.Visible='off';
        WK4_PANEL.Visible='off';
        %Function creating window must appear: it needs its panel and
        %the global variables as inputs
        WK_4(WK4_PANEL,handles)
        WK4_PANEL.Title='Modify the desired 4 WK Parameters: ';
        WK4_PANEL.Visible='on';
        ax.Visible='on';
    end
end
%A similar process is developed if the user pushes one of the butttons
%available at the main panel. The difference is that this buttons are
only
%available at the main window, while the pop-up menu is visible and
%accessible at any window.
%
%2 WINDKESSEL
function WK_2BTN(source,callbackdata)
    txtWelcome.Visible = 'off';
    hp.Visible = 'off';
    popup.Value=2; %Set to '2 elements WK';
    %Function creating window must appear: it needs its panel and
    %the global variables as inputs
    WK_2(WK2_PANEL,handles)
    WK2_PANEL.Title='Modify the desired 2 WK Parameters: ';
    WK2_PANEL.Visible='on';
    ax.Visible='on';

end
%3 WINDKESSEL
function WK_3BTN(source,callbackdata)
    txtWelcome.Visible = 'off';
    hp.Visible = 'off';
    popup.Value=3; % Set to '3 elements WK';
    %Function creating window must appear: it needs its panel and
    %the global variables as inputs
    WK_3(WK3_PANEL,handles)
    WK3_PANEL.Title='Modify the desired 3 WK Parameters: ';
    WK3_PANEL.Visible='on';
    ax.Visible='on';

end
%4 WINDKESSEL (parallel)
function WK_4BTN(source,callbackdata)
    txtWelcome.Visible = 'off';
    hp.Visible = 'off';
    popup.Value=4; %Set to '4 elements WK';
    %Function creating window must appear: it needs its panel and
    %the global variables as inputs
    WK_4(WK4_PANEL,handles)
    WK4_PANEL.Title='Modify the desired 3 WK Parameters: ';
    WK4_PANEL.Visible='on';
    ax.Visible='on';

end

end
```

2-element Windkessel Window

```
function WK_2(WK_PANEL,handles)
%
% WK_2
% This function executes the window for specifically simulating the
% 2-elements Windkessel
% INPUTS:
%   WK_panel: panel created during WK_Main_Interface_Reading,
%   all buttons will be located in this panel
%   Handles: global structure with some common data:
%       handles.dataOriginal=[Heart_Rate Time_Systole Max_Flow
Number_Cycles WK2_Rp WK2_C WK3_Rp WK3_C WK3_Ra WK4_Rp WK4_C WK4_Ra
WK4_L];
%       handles.data=handles.dataOriginal;
%       handles.additional=[Sec SItoFlow FlowtoSI SItoHRU HRUtoSI
SItoHCU HCUtoSI SItoHLU HLUtoSI];
%
%%

%DEFINING ALL THE ELEMENTS FOR THE PANEL (first make it invisible)
WK_PANEL.Visible='off';

%Defining titles:
TitleParameters =
uicontrol('Parent',WK_PANEL,'Style','text','FontName','Calibri',...
        'Units','normalized','FontWeight','Bold','Position',[0.07 0.87
0.15 0.07],...
        'FontSize',11,'BackgroundColor',[1 1 1],'String','Parameter to
modify:');
SIunits =
uicontrol('Parent',WK_PANEL,'Style','text','FontName','Calibri',...
        'Units','normalized','FontWeight','Bold','Position',[0.29 0.85
0.15 0.07],...
        'FontSize',11,'BackgroundColor',[1 1 1],'String','SI Units:
');
MEDunits =
uicontrol('Parent',WK_PANEL,'Style','text','FontName','Calibri','FontW
eight','Bold',...
        'Units','normalized','Position',[0.59 0.87 0.15
0.07],'FontSize',11,...
        'BackgroundColor',[1 1 1],'String','Medical Units: ');

%Defining Display Button:allows user to choose between displaying
%pressure,flow and PV-loop

Display = uicontrol('Parent',WK_PANEL,'Style',
'popup','Units','normalized','ForegroundColor',[0 0 1],...
   'String', {'Aortic Pressure','Aortic Flow','Aortic PV-
Loop'},'FontName','Calibri',...
   'FontSize',10,'Position', [0.7 0.15 0.2 0.1],...
   'Callback', @Display_Callback);

Display_text =
uicontrol('Parent',WK_PANEL,'Style','text','FontName','Calibri','FontW
eight','Bold',...
        'Units','normalized','Position',[0.6 0.25 0.35
0.05],'FontSize',11,...
        'BackgroundColor',[1 1 1],'String','Select a variable to
display: ');

%Defining Reset Button

RESET= uicontrol('Parent',WK_PANEL,'Style', 'pushbutton', 'String',
'RESET','FontWeight','Bold',...
        'BackgroundColor',[1 1 1],'Units','normalized','Position',
[0.7 0.05 0.2 0.1],...
        'FontSize',10,'ForegroundColor',[0 0
1],'Callback',@RESET_btn_Callback );
```

```
RESET_text =
uicontrol('Parent',WK_PANEL,'Style','text','FontName','Calibri','FontW
eight','Bold',...
        'Units','normalized','Position',[0.6 0.14 0.35
0.05],'FontSize',11,...
        'BackgroundColor',[1 1 1],'String','Return to default values:
');

%Defining all buttons
HR_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.43 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Heart Rate:');

HR_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.45 0.12
0.05],'String',handles.data(1),'Callback', @HR_edit_Callback);
%Units
D='$$\frac{beats}{min}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.43 0.15
0.07],'LineStyle','none','FontSize',10);

HR_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized','SliderStep',[0.1 0.1],...
        'Min',60,'Max',140,'Value',handles.additional(1),'Position',
[0.3 0.43 0.12 0.02],'Callback', @HR_slider_Callback);

T_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.33 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Heart
Period:');

T_text =
uicontrol('Parent',WK_PANEL,'Style','Text','BackgroundColor',[1 1
1],'Units','normalized',...
        'Position',[0.3 0.34 0.12
0.05],'String',handles.additional(1)/handles.data(1),'Callback',
@T_text_Callback);
%Units
D='$$ seconds $$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.33 0.15
0.07],'LineStyle','none','FontSize',10);

Ts_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.23 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Time in
systole:');

Ts_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.25 0.12
0.05],'String',handles.data(2),'Callback', @Ts_edit_Callback);
%Units
```

```
D='$$ seconds $$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.23 0.15
0.07],'LineStyle','none','FontSize',10);

Ts_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized','SliderStep',[0.1 0.1],...
        'Min',0,'Max',0.83,'Value',handles.data(2),'Position', [0.3
0.23 0.12 0.02],'Callback', @Ts_slider_Callback);

Td_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.13 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Time in
diastole:');

Td_text =
uicontrol('Parent',WK_PANEL,'Style','Text','BackgroundColor',[1 1
1],'Units','normalized',...
        'Position',[0.3 0.14 0.12
0.05],'String',handles.additional(1)/handles.data(1)-
handles.data(2),...
        'Callback', @Td_text_Callback);
%Units
D='$$ seconds $$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.13 0.15
0.07],'LineStyle','none','FontSize',10);

Q0_title =
uicontrol('Parent',WK_PANEL,'Style','text','BackgroundColor',[1 1
1],'Units','normalized',...
        'Position',[0.07 0.76 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Maximum
Flow:');

Q0_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.78 0.12
0.05],'String',handles.data(3),'Callback', @Q0_edit_Callback);
%Units
D='$$\frac{m^{3}}{s}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.76 0.15
0.07],'LineStyle','none','FontSize',10);

Q0_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',1.999E-4,'Max',1E-3,'SliderStep',[0.1
0.1],'Value',handles.data(3),...
        'Position', [0.3 0.76 0.12 0.02],'Callback',
@Q0_slider_Callback);

Q0_edit_med =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.6 0.78 0.12
0.05],'String',(handles.additional(2))*handles.data(3),...
```

```
        'Callback', @Q0_edit_med_Callback);
%Units
D='$$\frac{mL}{s}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.73 0.76 0.15
0.07],'LineStyle','none','FontSize',10);

Q0_slider_med = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',200,'Max',1000,'SliderStep',[0.1
0.1],'Value',(handles.additional(2))*handles.data(3),...
        'Position', [0.6 0.76 0.12 0.02],'Callback',
@Q0_slider_med_Callback);

NumberCycles_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.03 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Number of
Cycles:');

NumberCycles_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.05 0.12
0.05],'String',handles.data(4),'Callback',
@NumberCycles_edit_Callback);
%Units
D='\# number';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.03 0.15
0.07],'LineStyle','none','FontSize',10);

NumberCycles_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',1,'Max',20,'SliderStep',[0.1
0.1],'Value',handles.data(4),'SliderStep',[1/(19) 1/(19)],...
        'Position', [0.3 0.03 0.12 0.02],'Callback',
@NumberCycles_slider_Callback);

Rp2_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.65 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Peripheral
Resistance:');

Rp2_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.67 0.12
0.05],'String',handles.data(5),'Callback', @Rp2_edit_Callback);
%Units
D='$$\frac{kg}{s m^{4}}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.65 0.15
0.07],'LineStyle','none','FontSize',10);

Rp2_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
```

```
        'Min',7.997E6,'Max',3.999E8,'SliderStep',[0.1
0.1],'Value',handles.data(5),...
        'Position', [0.3 0.65 0.12 0.02],'Callback',
@Rp2_slider_Callback);

Rp2_edit_med =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.6 0.67 0.12
0.05],'String',(handles.additional(4))*handles.data(5),...
        'Callback', @Rp2_edit_med_Callback);
%Units
D='$$\frac{mmHg\cdot min}{l}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.73 0.65 0.15
0.07],'LineStyle','none','FontSize',10);

Rp2_slider_med = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',1,'Max',50,'SliderStep',[0.1
0.1],'Value',(handles.additional(4))*handles.data(5),...
        'Position', [0.6 0.65 0.12 0.02],'Callback',
@Rp2_slider_med_Callback);

C2_title =
uicontrol('Parent',WK_PANEL,'Style','text','Units','normalized',...
        'Position',[0.07 0.54 0.15
0.07],'FontSize',10,'BackgroundColor',[1 1 1],'String','Arterial
Compliance:');

C2_edit =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.3 0.56 0.12
0.05],'String',handles.data(6),'Callback', @C2_edit_Callback);
%Units
D='$$\frac{s^{2}m^{4}}{kg}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.43 0.54 0.15
0.07],'LineStyle','none','FontSize',10);

C2_slider = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',7.502E-10,'Max',1.501E-8,'SliderStep',[0.1
0.1],'Value',handles.data(6),...
        'Position', [0.3 0.54 0.12 0.02],'Callback',
@C2_slider_Callback);

C2_edit_med =
uicontrol('Parent',WK_PANEL,'Style','Edit','Units','normalized',...
        'Position',[0.6 0.56 0.12
0.05],'String',(handles.additional(6))*handles.data(6),...
        'Callback', @C2_edit_med_Callback);
%Units
D='$$\frac{ml}{mmHg}$$';
h=annotation(WK_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','la
tex',...
        'FitBoxToText','on','Position',[0.73 0.54 0.15
0.07],'LineStyle','none','FontSize',10);
```

```
C2_slider_med = uicontrol('Parent',WK_PANEL,'Style',
'slider','Units','normalized',...
        'Min',0.1,'Max',2,'SliderStep',[0.1
0.1],'Value',(handles.additional(6))*handles.data(6),...
        'Position', [0.6 0.54 0.12 0.02],'Callback',
@C2_slider_med_Callback);


%%
%Callback functionsdefinitions
function HR_edit_Callback(source,callbackdata)
    handles.data(1)=str2double(get(source,'String'));

    %As HR changes, T value displayed must also change
    set(T_text,'String',handles.additional(1)/handles.data(1));
    %Ts and Td must be recalculated too
    handles.data(2)=(2/5)*(handles.additional(1)/handles.data(1));
    set(Ts_edit,'String',handles.data(2));
    set(Td_text,'String',(handles.additional(1)/handles.data(1)-
handles.data(2)));
    set(Ts_slider, 'Max', (handles.additional(1)/handles.data(1)));
    set(Ts_slider, 'Value', (handles.data(2)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function HR_slider_Callback(source,callbackdata)
    handles.data(1)=source.Value;

    set(HR_edit,'String',(handles.data(1)));
    set(T_text,'String',(handles.additional(1)/handles.data(1)));
    handles.data(2)=(2/5)*(handles.additional(1)/handles.data(1));
    set(Ts_edit,'String',(handles.data(2)));
    set(Td_text,'String',((handles.additional(1)/handles.data(1)-
handles.data(2))));
    set(Ts_slider, 'Max', (handles.additional(1)/handles.data(1)));
    set(Ts_slider, 'Value', (handles.data(2)));
    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Ts_edit_Callback(source,callbackdata)
    %Get new Ts, store it and displayed the new Td for the same period
as
    %Td=T-Ts
if
((str2double(get(source,'String')))<=(handles.additional(1)/handles.da
ta(1)))
    handles.data(2)=str2double(get(source,'String'));

    set(Ts_slider, 'Value', (handles.data(2)));
    set(Td_text,'String',(handles.additional(1)/handles.data(1)-
handles.data(2)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
else
    handles.data(2)=(2/5)*(handles.additional(1)/handles.data(1));
    set(Ts_edit,'String',handles.data(2));
    set(Td_text,'String',(handles.additional(1)/handles.data(1)-
handles.data(2)));
    set(Ts_slider, 'Value', round(handles.data(2),3));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

end

function Ts_slider_Callback(source,callbackdata)
    handles.data(2)=source.Value;

    set(Ts_edit,'String',(handles.data(2)));
    set(Td_text,'String',((handles.additional(1)/handles.data(1)-
handles.data(2))));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Q0_edit_Callback(source,callbackdata)
    handles.data(3)=str2double(get(source,'String'));

    set(Q0_edit_med,'String',(handles.data(3)*handles.additional(2)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Q0_slider_Callback(source,callbackdata)
    handles.data(3)=source.Value;

    set(Q0_edit,'String',handles.data(3));

set(Q0_edit_med,'String',(handles.data(3)*(handles.additional(2))));

set(Q0_slider_med,'Value',round(handles.data(3)*(handles.additional(2)
)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Q0_edit_med_Callback(source,callbackdata)

handles.data(3)=(handles.additional(3))*str2double(get(source,'String'
));

    set(Q0_edit,'String',handles.data(3));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Q0_slider_med_Callback(source,callbackdata)
    handles.data(3)=handles.additional(3)*source.Value;

    set(Q0_edit_med,'String',source.Value);
    set(Q0_edit,'String',handles.data(3));
    set(Q0_slider, 'Value',handles.data(3));

    %Compute and plot
    if (Display.Value==1)
```

```
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure  (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure  (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function NumberCycles_edit_Callback(source,callbackdata)
    handles.data(4)=str2double(get(source,'String'));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure  (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure  (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function NumberCycles_slider_Callback(source,callbackdata)
    handles.data(4)=source.Value;

    set(NumberCycles_edit,'String',handles.data(4));

    %Compute and plot
    if (Display.Value==1)
```

```
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Rp2_edit_Callback(source,callbackdata)
    handles.data(5)=str2double(get(source,'String'));


set(Rp2_edit_med,'String',(handles.data(5)*handles.additional(4)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Rp2_slider_Callback(source,callbackdata)
    handles.data(5)=source.Value;

    set(Rp2_edit,'String',handles.data(5));
set(Rp2_edit_med,'String',(handles.data(5)*handles.additional(4)));

set(Rp2_slider_med,'Value',round(handles.data(5)*handles.additional(4)
));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Rp2_edit_med_Callback(source,callbackdata)

handles.data(5)=(handles.additional(5))*str2double(get(source,'String'
));

    set(Rp2_edit,'String',handles.data(5));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function Rp2_slider_med_Callback(source,callbackdata)
    handles.data(5)=(handles.additional(5))*source.Value;

    set(Rp2_edit_med,'String',source.Value);
    set(Rp2_edit,'String',handles.data(5));
    set(Rp2_slider, 'Value',handles.data(5));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function C2_edit_Callback(source,callbackdata)
    handles.data(6)=str2double(get(source,'String'));

    set(C2_edit_med,'String',(handles.data(6)*handles.additional(6)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
```

```
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function C2_slider_Callback(source,callbackdata)
    handles.data(6)=source.Value;

    set(C2_edit,'String',handles.data(6));
    set(C2_edit_med,'String',(handles.data(6)*handles.additional(6)));

set(C2_slider_med,'Value',round(handles.data(6)*handles.additional(6),
2));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function C2_edit_med_Callback(source,callbackdata)

handles.data(6)=(handles.additional(7))*str2double(get(source,'String'
));

    set(C2_edit,'String',handles.data(6));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
```

```
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function C2_slider_med_Callback(source,callbackdata)
    handles.data(6)=(handles.additional(7))*source.Value;

    set(C2_edit_med,'String',source.Value);
    set(C2_edit,'String',handles.data(6));
    set(C2_slider, 'Value',handles.data(6));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

function RESET_btn_Callback(source,callbackdata)
    %Reset all parameters to original values
    handles.data=handles.dataOriginal;
    %Modify visible values
    set(HR_edit,'String',handles.data(1));
    set(T_text,'String',handles.additional(1)/handles.data(1));
    set(Ts_edit,'String',handles.data(2));
    set(Td_text,'String',(handles.additional(1)/handles.data(1)-
handles.data(2)));
    set(Q0_edit,'String',handles.data(3));
    set(NumberCycles_edit,'String',handles.data(4));
    set(Rp2_edit,'String',handles.data(5));
    set(C2_edit,'String',handles.data(6));

set(Q0_edit_med,'String',((handles.data(3)*handles.additional(2))));

set(Rp2_edit_med,'String',((handles.data(5)*handles.additional(4))));

set(C2_edit_med,'String',((handles.data(6)*handles.additional(6))));
    %Sliders
    set(Rp2_slider,'Value',handles.data(5));

set(Rp2_slider_med,'Value',(handles.data(5)*handles.additional(4)));
    set(C2_slider,'Value',handles.data(6));

set(C2_slider_med,'Value',(handles.data(6)*handles.additional(6)));
    set(HR_slider, 'Value', (handles.data(1)));
    set(Ts_slider, 'Max', (handles.additional(1)/handles.data(1)));
    set(Ts_slider, 'Value', (handles.data(2)));
    set(NumberCycles_slider, 'Value', handles.data(4));
    set(Q0_slider, 'Value',handles.data(3));

set(Q0_slider_med,'Value',round(handles.data(3)*(handles.additional(2)
)));

    %Compute and plot
    if (Display.Value==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==2)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (Display.Value==3)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end
function Display_Callback(source,callbackdata)
    NumberMenu = source.Value;
    Names= source.String;
    NameMenu = Names{NumberMenu};
    if (strcmp(NameMenu,'Aortic Pressure')==1)
        [P,Q]=Compute_2WK (handles);
        handles.current_data=P;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Pressure (P)[mmHg]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (strcmp(NameMenu,'Aortic Flow')==1)
```

```
        [P,Q]=Compute_2WK (handles);
        handles.current_data=Q;
        %Plot pressures
        plot(handles.current_data);
        ylabel('Flow(Q)[ml/s]')
        xlabel('Time(t)[ms]')
        title('Pressure in the Aorta, 2WK')
    elseif (strcmp(NameMenu,'Aortic PV-Loop')==1)
        [P,Q]=Compute_2WK (handles);
        plot(Q,P)
        xlabel('Flow(Q)[ml/s]')
        ylabel('Pressure (P)[mmHg]')
        title('Aortic PV-Loop, 2WK')
    end
end

pause(0.35)
%THIS WILL ONLY EXECUTE ONCE
[P,Q]=Compute_2WK (handles);
handles.current_data=P;
%Plot pressures
plot(handles.current_data);
ylabel('Pressure[mmHg]')
xlabel('Time(t)[ms]')
title('Pressure in the Aorta, 2WK')

end
```

2-element Windkessel Calculations

```
function [P_medunits,Q_medunits]=Compute_2WK (handles)
%%
% Compute_2WK
% This function performs the calculations required for plotting the
% pressure, flow and pressure volume loops. This is performed using
the
% equations from the model developed. In this function the TWO
WINDKESSEL
% MODEL is implemented.
% INPUTS:
%   Handles: global structure with some required data:
%       handles.dataOriginal=[Heart_Rate Time_Systole Max_Flow
Number_Cycles
%       WK2_Rp WK2_C WK3_Rp WK3_C WK3_Ra WK4_Rp WK4_C WK4_Ra WK4_L];
%       handles.data=handles.dataOriginal;
%       handles.additional=[Sec SItoFlow FlowtoSI SItoHRU HRUtoSI
SItoHCU
%       HCUtoSI SItoHLU HLUtoSI];
%
%%
%Defining some variables from the global ones

T=60/handles.data(1);
Td=T-handles.data(2);

%%
%Define flow function
Q=zeros(handles.data(4)*(int16(round(T,3)/0.001)),1);
aux=1;
%For 1 to the desired number of periods:
```

```
for i=1:handles.data(4)
    for t=0.001:0.001:round(T,3)
        %0.001 is the time interval between consecutive samples
        if (t<=handles.data(2))
            %When t<systole duration
            Q(aux)=handles.data(3)*(sin((pi*t)/handles.data(2)))^2;
            %Flow is defined as a sine squared function during systole
        end
        if (t>handles.data(2))
            %When t>systole duration
            Q(aux)=0;
            %And as zero during diastole
        end
        aux=aux+1;
    end
end
%As the global variables are defined in SI units, convert the to
medical
%units for display purposes
Q_medunits=Q*10^6;

%%
%Define Pressure function for a 2 WK
P=zeros(handles.data(4)*(int16(round(T,3)/0.001)),1);
P(1)=10000;
%It will have the same number of elements as the flow
for i=2:1:length(Q)
    %The following equation is obtained from the numerical resolution
to
    %the equation modeling the two-element-configuration circuit.

P(i)=(((handles.data(5)*Q(i)*0.001)/handles.data(6))+handles.data(5)*P
(i-1))/(handles.data(5)+0.001/handles.data(6));
end
%Again, transform to medical units for display purposes.
P_medunits=P*760/101300;
end
```