uc3m | Universidad Carlos III de Madrid

Biomedical Engineering degree

Academic course: 2016-2017

Bachelor Thesis

"OPTIMIZATION OF A WINDKESSEL SYSTEM FOR ARTERIAL SIMULATION"

Author: Laura López Acedo

Supervisor:

Manuel Desco Menéndez

Leganés, October 2017

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

ACKNOWLEDGEMENTS

At first, I want to thank to my supervisor Manuel Desco, for offering me the opportunity to continue with this project. I really appreciate your advices and guidance.

I do not want to forget to all people of the LIM, where everything started last summer. You brought enjoyment to my internship and made me learn so much.

I also want to express my gratitude to all the people working in the Bioengineering Laboratories, specially to Guillermo Vizcaíno, for brightening my dark days.

Thanks to my family and friends. You made an incredible effort to understand this project, and your keenness was contagious. Your support has been very important for me.

Thanks to Mario for your infinite patience and unconditional love. We will be unbreakable.

SUMMARY

Cardiovascular diseases are the leading cause of death worldwide. In this context, prevention is of great interest. A profound understanding of cardiovascular physiology and hemodynamics is required to achieve prevention; it should be acquired during the educational stage not only in medical field but also in other academic disciplines. In recent years, mathematical models have become a valuable alternative to simulate possible pathologies that help its understanding before facing them in a real context. Therefore, the aim of this Bachelor Thesis is the optimization of an arterial system simulation tool for educational purposes. The system consists of a physical simulator communicating with the computer through a MATLAB program and a theoretical simulation tool also implemented in MATLAB. Both parts describe the arterial system based on the Windkessel mathematical model, a simple arterial model but accurate enough for this project purposes. The final result is a more efficient tool, with an improved design and user-friendly related software platforms.

A discussion on the legal frame of the project, its possible socio-economic impact and estimated project budget is included in the document.

It is expected that the developed tool may be used by students of the Universidad Carlos III de Madrid, as a practical complement to the theoretical lessons on Anatomy and Physiology subject of the Biomedical Engineering degree. Thus, a practice guideline is included, specifying the instructions of use of the system and suggesting questions regarding the obtained results.

Key Words:

Cardiovascular system, mathematical modeling, education, MATLAB, arterial Windkessel.

RESUMEN

Las enfermedades cardiovasculares constituyen la primera causa de muerte en todo el mundo. En este contexto, la prevención se vuelve aún más importante. Para lograrlo se necesita un hondo conocimiento de la fisiología y hemodinámica que debe ser adquirido durante la etapa formativa de los futuros profesionales, tanto del sector médico como de un sector más técnico. En esta era, los modelos matemáticos son una alternativa cada vez más valiosa para la simulación de posibles patologías que ayude a su entendimiento antes de enfrentarse a ellas en un contexto real. Por ello, el objetivo de este proyecto es la optimización de una herramienta de simulación del sistema arterial con fines educativos. El sistema consta de un simulador físico que se comunica con un ordenador a través de un programa de MATLAB y de una simulación teórica también implementada en MATLAB. Ambas partes se basan en el modelo matemático Windkessel, un modelo arterial simple pero suficientemente exacto para lo requerido en este trabajo. El resultado final es una herramienta más eficiente, con un diseño mejorado y que permite al usuario una navegación sencilla.

Se incluye una discusión sobre el marco legal del proyecto, su posible impacto socioeconómico y el coste estimado asociado.

Se espera que el sistema pueda ser utilizado por los alumnos de la Universidad Carlos III como complemento práctico a las lecciones teóricas de Anatomía y Fisiología en el grado de Ingeniería Biomédica. Por tanto, se propone una guía de laboratorio que incorpora las instrucciones de uso del sistema y preguntas para los estudiantes acerca de los resultados observados.

Palabras clave:

Sistema cardiovascular, modelo matemático, educación, MATLAB, arterial Windkessel.

INDEX

List of figures

List of tables

1. INTRODUCTION

1.1. Antecedents

Cardiovascular function is essential for life, but it declines with age. As the general population ages, more individuals are prone to suffer cardiovascular diseases. In fact, cardiovascular disease is the leading cause of death among many regions, mainly in developed regions like Europe and the United States of America [1] [2]. In recent decades, medicine trend is not only to cure, but to prevent conditions and diseases, and there is a lot of investment from governments on it.

Regarding cardiovascular diseases, the prioritization is not only prevention but also promotion of positive cardiovascular health. Health behaviors are constantly transmitted to population, including healthy diet, physical activity and tobacco and alcohol avoidance [3].

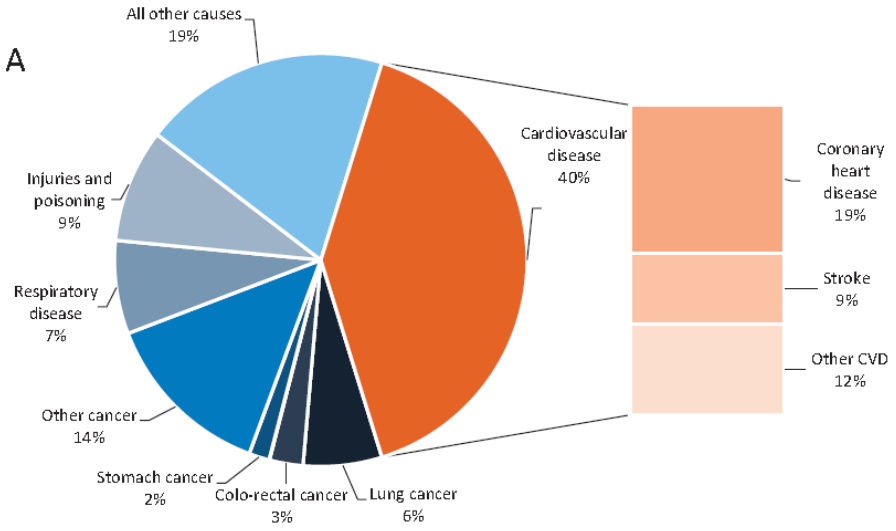


Figure 1. Proportion of all male deaths due to major causes in Europe in 2016 [2].

Simulation is one of the tools to achieve prevention, as it helps to understand function. They allow physicians to gain a better understanding of cardiovascular diseases (CVD) consequences and provide a powerful tool for researchers to further develop drugs and other tools in a safe way. Data from simulation can predict some CVD effects, and help to achieve a deep understanding of anatomy and physiology. They therefore constitute a good starting point for further investigation. However, this is not the only application. Models constitute an advantageous tool in education, being a complement in nursing and medicine fields, among others.

There are many categories of simulation models, but the generation of most of them requires knowledge of both, cardiovascular physiology and pathology and technical

background. Models are based on complex mathematic approaches, only comprehensible for people with a technical profile. This is just one of the areas in which a Biomedical Engineer can be of huge help.

Biomedical engineering is the result of the evolution of the modern health care system previously commented. Although the denomination is new, the technology potential in health care practices has been exploited for decades.

Biomedical engineers' role is to apply engineering's own analytical and synthetic methodologies to study living organisms. Their aim is to apply the scientific knowledge to solve real-world problems in a cost-effective way [4]. The area of expertise is huge and is growing, including development of new diagnostic imaging systems, design of biomedical sensors, study of biomechanics and the used here, modeling of the physiologic systems of the human body, among others.

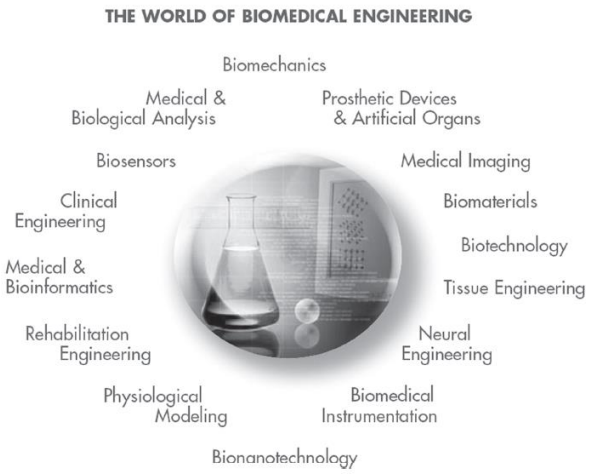


Figure 2. The world of biomedical engineering [4].

1.2. Motivation

The main motivation of this project is the availability of an arterial simulator prototype at the Bioengineering laboratories of the Universidad Carlos III de Madrid (UC3M), together with the interest of the author in the cardiovascular system and bioinstrumentation.

Biomedical engineering students at UC3M study the subject Anatomy and Physiology I and II, which contribute to the multidisciplinary program that defines the degree. To help in the practical teaching of this subject, the University acquired an arterial simulator prototype to be used for practical exercises. It was intended to help not only to better

understand physiology, but to serve as a complement of other topics like programming, electronics, etc.

After some years of previous work, the device was finally operational at the beginning of this project, and a MATLAB interface had been created to enable its use. Besides, another MATLAB simulation interface had been programmed, that simulated the arterial behavior in a purely theoretical way. In both cases there were some weaknesses of the system at the beginning of this project. Therefore, there was an opportunity of optimizing the physical device and improving the related programming tools so that the system could be finally useful for the students. It was interesting to take advantage of the modular nature of the previous work, and define the objectives of this thesis starting from that point.

This project, then, emerges from the opportunity of optimizing an arterial model consisting of a hydraulic-pneumatic device and a theoretical simulation intended to be used for educational purposes.

1.3. Objectives

The main objective of this thesis is to continue a previous work [5] to set up an arterial system simulation tool for training. Two different parts constitute this Bachelor Thesis:

- Optimization of a pneumatic-hydraulic device that models cardiovascular system with a physical Windkessel configuration [6] and a related software platform to acquire data from the system.
- Optimization of a software platform for the theoretical simulation of aortic hemodynamics based on Windkessel mathematical model.

The specific goals of the project are listed below:

1. To identify the current situation of the physical system.
2. To define the hardware and software requirements and plan the steps to follow based on the current situation of the system.
3. To accomplish hardware development and implementation.
4. To accomplish software development and implementation.
5. To create a new laboratory practice guide for the students and generate adequate documentation for the project.

1.4. Outline of the manuscript

Th first chapter of the manuscript is a general introduction, where the project is put into context and the author expresses the motivation for the work and the objectives to be accomplished. From this point on, the document is divided into four sections.

In the second chapter, a description of the theoretical concepts used during this thesis is included. It provides background on cardiovascular physiology and modeling, ending with a presentation of the state of the art.

The third chapter includes a detailed description of the physical device used for this work and the author's contributions; there is an intensive explanation of the project development, the software implementations and the laboratory practice guide.

The fourth chapter presents a brief discussion on the legal framework for this project, its potential impact in society and a detailed budget breakdown.

Finally, the fifth chapter divides into two parts: a conclusions section and a description of the possible lines of future work.

The last pages are reserved for the references and annexes.

2. BACKGROUND

2.1. Cardiovascular system anatomy and physiology

The cardiovascular system has three components: blood, heart and blood vessels. This manuscript makes a brief description of heart and blood vessels, specifically arteries, since those are the components more related to this project.

2.1.1. The heart

The heart is a relatively small organ lying in the mediastinum, between the vertebral column and the sternum. It continuously pumps blood that circulates through all the body to maintain homeostasis. The heart wall consists of three layers (epicardium, myocardium and endocardium), each of them composed of different tissues that finally allow the synchronized movement that characterizes this organ.

Internally, the heart comprises two pumps: a right heart and left heart. Each of these pumps has two chambers termed atrium (upper part) and ventricle (lower part). Each atrium acts as a primer pump by helping to move the blood into the ventricle; that is, atria are receiver chambers. The "doors" allowing blood to reach ventricles are the atrioventricular valves (bicuspid or mitral and tricuspid valves), from now on referred as A-V valves. When the blood has traversed these valves, the ventricles force the blood to then leave the heart and to go to the rest of the body, traversing valves known as semilunar valves. The ejected blood goes into blood vessels called arteries, which transport it to the rest of the vascular tree. The blood pumped by the right ventricle is ejected to the pulmonary artery and then it travels through the pulmonary circulation, while the blood pumped by the left ventricle is ejected into the aorta and it goes through the peripheral circulation. The force required in each ventricle is therefore different, since the resistance of each blood circuit is different [7].

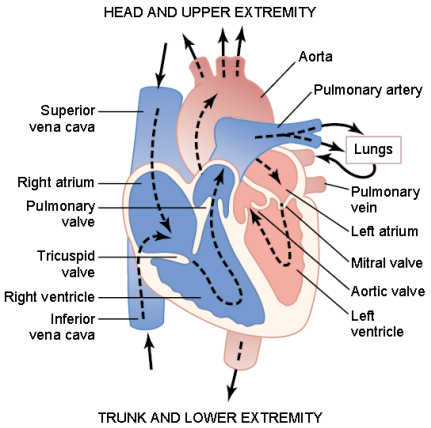


Figure 3. Structure of the heart and course of blood flow through it [7].

The blood exchange within the heart is a set of synchronized events known as cardiac cycle, that repeats itself in each heartbeat. The cardiac cycle consists of two main stages in which atria and ventricles alternatively contract and relax: systole and diastole. The addition in time of the two stages represents one heartbeat. Therefore, as the heart rate increases, the contraction and relaxation phases duration decreases.

During systole, A-V valves are closed and therefore the blood accumulates in the atria. When systole is over and the ventricular pressures fall to low values, the increased atrial pressures push the A-V valves to open, and the ventricles fill of blood while atria empty.

At a given moment, the ventricular pressure is very high, which causes the A-V to close. There is a period in which there are not open valves, since the ventricles need an additional time to push the semilunar valves to open against the pressure in aorta and pulmonary artery. In this period, called isovolumetric contraction, there is ventricle contraction but not emptying.

Immediately after the semilunar valves open, ejection of blood from the left and right ventricles to the aorta and pulmonary artery, respectively, occurs.

At the end of the cycle, ventricular pressures decrease and ventricles relax, and the pressure in the arteries rises, as they have just been filled. As a consequence, the semilunar valves close, leading to the isovolumetric relaxation. Finally, the A-V valves open again to begin another cycle.

A basic parameter of the cardiac cycle is the stroke volume of the ventricle, which is represented by the difference between the end diastolic volume (EDV) and the end systolic volume (ESV):

$$SV = EDV - ESV$$

Related to stroke volume is the cardiac output (CO). It is the volume of blood pumped by the heart over one minute:

$$CO = HR \cdot SV$$

Where HR is the heart rate measured in beats per minute.

From these two parameters it is possible to extract information useful to understand possible heart malfunctioning.

Measure	Typical value	Normal range
End diastolic volume (EDV)	130 ml	65-240 ml
End systolic volume (ESV)	60 ml	16-143 ml
Stroke volume (SV)	70 ml	55-100 ml
Ejection fraction ($\boldsymbol{E_f}$)	65%	55-70%
Heart rate (HR)	75 bpm	60-100 bpm
Cardiac output (CO)	5.25 L/min	4.00 – 8.00 L/min

Table 1. Typical values in healthy 70 kg male and human normal range of hemodynamic parameters. Values extracted from Tortora's book [8].

It has been explained that the A-V valves act by preventing blood backflow from ventricles to atria during systole, and the semilunar valves act by preventing blood backflow from the aorta and pulmonary arteries to the ventricles during diastole. The opening and closing of these valves is passive, which means that the pressure gradient drives its movement.

2.1.2. Blood vessels in the circulatory system

When the blood leaves the heart, it is transported through the circulatory system to all the organism. Circulatory system function is to transport blood to all tissues to service all their needs, from supplying nutrients, oxygen and hormones to the transport of waste products away. Circulation is divided into systemic and pulmonary circulation: systemic circulation supplies blood to all the body except the lungs and pulmonary circulation supplies blood to the lungs.

There are five different types of blood vessels in the circulatory system: arteries, arterioles, capillaries, venules and veins. From the ventricles, the blood flows to the aorta and pulmonary artery, and from them to other arteries. Arteries have strong walls which allow to transport blood at high pressure and with high velocity. The last stop on the arterial system are the arterioles, which release blood into the capillaries. These capillaries are in charge of exchanging substances between the blood and interstitial fluids. Venules then collect the output of capillaries and finally transport blood to veins, which transport blood back to the heart.

Physiological parameters as the blood flow and pressure vary in each of the functional segments of the vascular system. Considering the scope of this project, the most interesting vessel type is the artery so the arterial system will constitute the main object of attention in the next lines.

2.1.3. Arterial system hemodynamics

Arteries can be divided into elastic and muscular depending on their anatomic composition. Large arteries are elastic to accept the flow pulse ejected from the heart under great pressure. In these arteries blood flows with almost no resistance and they conduct blood to smaller size arteries. Smaller muscular arteries and arterioles present a high resistance to flow because of their smaller diameter, and regulate direct flow to tissues by changing this diameter [9]. Together, the elastic and muscular vessels allow the blood to constantly flow into the capillaries in a smooth way.

The first step for the blood in the systemic circulation when it leaves the heart is the aorta, where the pressure is the highest of all the human organism. The blood pressure is the force exerted by the blood in a vascular segment per unit of area and it is different in each of the phases of the cardiac cycle. During systole, the blood flows from the left ventricle to the aorta while the aortic valve is open, and then to the systemic arteries with varying pressure. In healthy young adults, the walls of these arteries stretch due to the blood passing through them to reach a pressure of 120 mm Hg (systolic pressure). During diastole, when the valve closes and the ventricle stops ejecting blood, the elastic walls maintain a high pressure in the arteries. In the aorta, the diastolic pressure barely falls about 80 mm Hg, since the blood flow is continuous through the peripheral vessels to the veins. Figure 4 shows these pressure variations along all the cardiac heart phases.

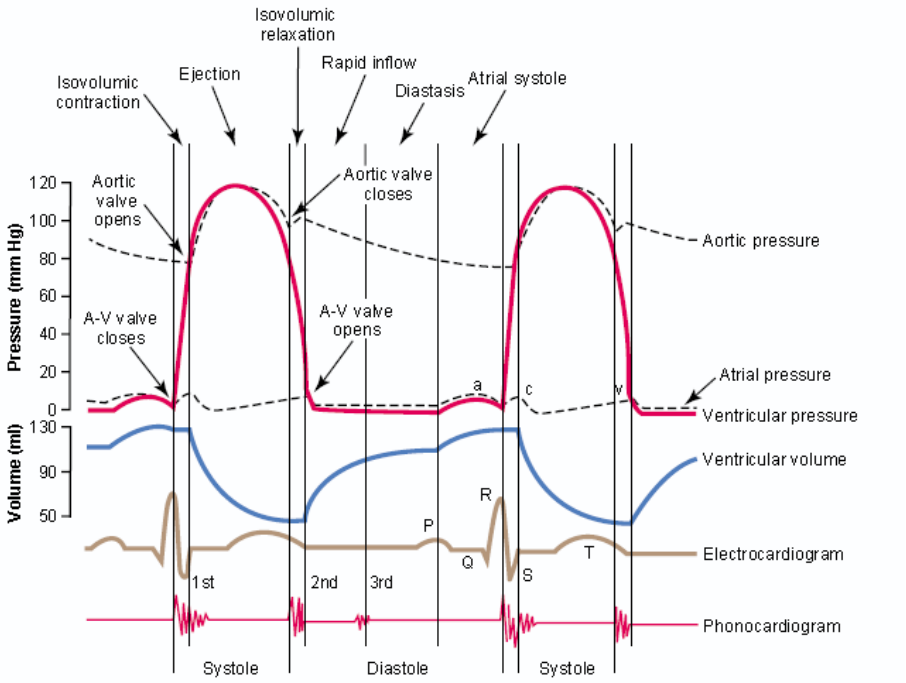


Figure 4. Events of the cardiac cycle for left ventricular function [7].

As the blood circulates though the different sections of vascular system, the pressure falls up to 0 mm Hg when the blood reaches again the heart through the cava vein.

The pressure's behavior is the same in the right ventricle and pulmonary artery, but with pressure values up to six times lower since the lung circuit resistance is much lower.

Basic cardiovascular hemodynamics cannot be explained without mentioning blood flow, defined as the amount of blood passing through a given point of the vascular tree in a given period of time.

2.2. Cardiovascular system pathology

As it has been already explained, the cardiovascular system has three parts, although in this work only two of them have been explained: the heart and the blood vessels. Therefore, any of these parts can fail and cause cardiovascular diseases (CVD).

Regarding the heart, one potential issue is to have "electrical" problems. Although it has not been detailed in the lines above, the heart is electrically stimulated to pump in a synchronized sequence of events. If there is a problem in the stimulation there will be an abnormal heart rhythm known as arrhythmia [10]. This condition may be harmless or life-threatening.

The problem may also be in the heart circulation itself: if the vessels supplying blood to the heart do not carry enough blood, the blood flow will decrease, as in the case of a coronary artery disease. The complete obstruction of flow in a coronary artery may cause myocardial infarction (heart attack).

Also inside the heart, both the A-V and the semilunar valves may not fully open or let blood backflow to the heart chambers. In the most severe cases, they may need to be surgically replaced by artificial ones.

The next section of the cardiovascular system is the circulation. There can be "mechanical" problems in the circuit and loads creating an imbalance between blood supply and metabolic demand. It causes a reduced blood flow resulting in a circulatory shock and it sometimes require surgical procedures like the placement of stents or grafts to restore the flow [11].

Another common condition is the hypertension, defined as high blood pressure that may damage the blood vessels and increases the risk of heart diseases. Aneurysm, atherosclerosis, infection, congenital defects, etc. are more examples of the very extensive disorders and diseases that can be found in this system.

All these pathologies are frequent and constitute a major problem for modern medicine worldwide. According to the WHO (World Health Organization), ischemic heart disease and stroke were the two leading causes of mortality at global level in 2015 [12], which reveals the importance of not only treat but prevent CVD. Therefore, models find their reason of living in helping to understand the cardiovascular system and giving answers to medicine professionals.

2.3. Cardiovascular modeling

In general terms, models are tools that help to understand the functionality of a real system by simplifying it. There are numerous modeling methods that should be evaluated depending on their application. Applications of physiological models include the design of new medical devices and collecting more information about the complex functioning of these systems [13]. One possible classification is to divide models between physical and mathematical. While physical models are a simple copy of an object, mathematical models select relevant features and numerically describe them to study a given object.

A cardiovascular model describes the main hemodynamics of the cardiovascular system using a minimal number of parameters. These tools can be used to represent the state of the cardiovascular system in a variety of physiological, pathological and clinical conditions [14]. The mathematical description of the cardiovascular system provides computer based simulations of dynamic processes of this system that play an important role in the comprehension of the complex interactions that characterize this system [13]. A better understanding of the system leads to improved medical treatments. In addition, it may help training students and reduce the number of experiments using animals.

Mathematically, a set of differential equations describe the behavior of the system in a selected number of dimensions, from zero to three dimensions, introducing several idealized assumptions to solve these equations [15]. The number of parameters used to describe a given system depends on the complexity and purpose of the model. It is possible to classify the models based on the number of dimensions they consider.

Zero-dimensional models or lumped parameter models consider uniform distribution of pressure, volume and flow rate in every single compartment. A compartment is each of the segments in which the circulation is partitioned: each one is described differently according to the local characteristics of the blood vessel, and finally all compartments are combined to yield a final model of the circulation system or a part of it. These models are useful when studying the global behavior of the system for specific physiological

conditions. Although they do not include spatial information, it can be estimated by setting different compartments, each of one being homogeneous and described by a different lumped parameter model [13]. Therefore, there are two types of lumped models: single compartment and multicompartment lumped parameter models.

Higher dimensional models allow insight into phenomena that lumped model do not describe. One-dimensional models describe the variation of the velocity of flow through the length of the blood vessel. Two-dimensional models account for radial changes of the flow velocity assuming a tube with axial symmetry. Three-dimensional models are the most complex ones, including the description of flow in vessel bifurcations and inside ventricles among others. These three methods are referred to as distributed parameter models. Distributed models have been used to study the effects of viscoelasticity, wave reflections and the relation of peripheral to central pressure waves among others [15]. Differences in the application of lumped and distributed models is shown in Table 2.

The model		The application
Distributed parameters	3D	Study of the local flow in 3-dimensional areas.
	2D	Study of local flow in vessels with axial symmetry.
	1D	Pulsed wave reflection effect in systemic circulation.
Lumped parameters	0D	Cardiovascular dynamics in the whole CVS. Pressure & flow changes in local areas of circulation.

Table 2. Comparison of model methods to study the cardiovascular system [13].

2.4. Windkessel model

The Windkessel model is a lumped, single-compartment model developed by the physiologist Otto Frank in 1899. Frank quantitatively described the arterial network in terms of compliance and resistance, suggesting that the pressure variations are related to the elasticity of the arteries [6].

In many cardiovascular models, and specifically in the Windkessel model, the arterial system is considered as a hydraulic network under the action of a pulsatile pump (the heart). Based on the analogy between the blood flow and the current in an electric circuit, it can be stated that the blood flow through a vessel of the circulatory system depends on:

- The pressure difference between the two ends of the vessel (ΔP).
- The vascular resistance resulting from friction between the blood and the inner vessel endothelium (R).

With these two data, it is possible to calculate the blood flow (F) through a vessel using an analogy of the Ohm's law. This law states the relationship between voltage and current in an ideal conductor [16] but its analogous in vascular physiology is:

$$F = \frac{\Delta P}{R}$$

Where $\Delta P = P_1 - P_2$.

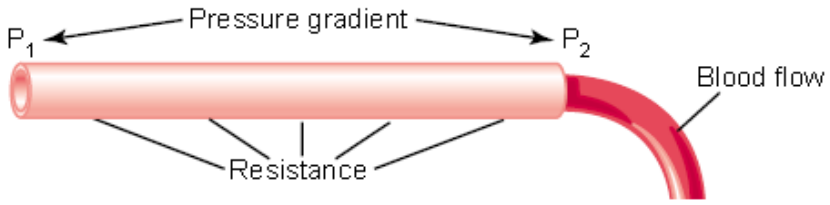


Figure 5. Interrelationships among pressure, resistance, and blood flow [6].

This relationship explains the basic hemodynamics of the circulation. However, the heart is a volumetric pump, which in its electrical analogy means that the heart is a current source. Therefore, the algebraic form of the Ohm's law that applies in this case is:

$$\Delta P = F \cdot R$$

The resistance R is described by the Poiseuille's law:

$$R = \frac{8 \cdot \eta \cdot l}{\pi \cdot r^4}$$

Where η is the blood viscosity, l the length of the vessel and r the radius of the vessel.

According to this definition, the resistance to flow mainly depends on the smallest arteries and the arterioles [17].

As it has been explained section 2.1.2, blood flows through different types of vessels, arranged in series. According to Ohm's law, when elements arrange in series, the total resistance to blood flow is equal to the sum of individual resistances, that is, the contribution of each vessel to the total resistance. Therefore, the so-called total peripheral resistance is equal to the sum of resistance of all arteries, arterioles, capillaries, venules and veins (the different conducts) [7]:

$$R_{total} = R_1 + R_2 + R_3 + \cdots + R_k$$

But blood vessels extend through the body to supply blood to all tissues, and therefore parallel circuits are formed. For vessels with this arrangement, the total resistance may be expressed as:

$$\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots + \frac{1}{R_k}$$

The Windkessel model begins from this basic relationship between flow and pressure, in which resistors of the equivalent electrical circuit represent the resistance to blood flow of the arterial system. It considers veins as low-resistance, zero-pressure far fields and therefore ignores them.

This model has several limitations. The arterial Windkessel does not explain wave transmission and wave travel as it is a lumped model. Blood flow distribution and local vascular changes cannot be described either. However, it is a fairly accurate approximation of the global effects [14].

Fluid dynamics	**Physiolog. variables**	**Electrical analogue**
Pressure P[Pa = J/m^3]	Blood press. [mmHg]	Voltage U [V = J/C]
Flow rate Q[m^3/s]	Blood flow rate [L/s]	Current I [A = C/s]
Volume V[m^3]	Blood volume [L]	Charge q [C]
Viscosity η	Bl. res. $\mathbb{R} = \frac{8\eta\mathcal{N}\ell}{\pi r^4}$	Electrical resistance R
Elastic coefficient	Vessel's wall compl.	Capacitor's capac. C
Inertance	Blood inertia	Inductor's inertance L
Poiseuille's law:	$Q = \frac{\Delta P}{\mathbb{R}} = \frac{\Delta P \pi r^4}{8\eta\ell}$	*Ohm's law*: $I = \frac{\Delta U}{\mathbb{R}}$

Table 3. Analogy between electrical and physiological variables [13].

2.4.1. 2-element Windkessel model

In addition to resistance, other dynamical properties can be estimated by different electrical circuit elements, such as compliance and inductance [18].

Capacitance or compliance is defined as the volume of blood that can be stored in a segment of a vessel per each unit increment of pressure. It describes the elasticity of the vessel walls and changing cross-sectional area:

$$Compliance = \frac{\Delta V}{\Delta P} = \frac{A}{\rho \cdot g}$$

Where A is the cross-sectional area of the vessel, ρ is the blood density and g is the acceleration due to gravity.

Therefore, large arteries with small resistive properties are called compliant arteries.

The so-called 2-element Windkessel model describes the arterial system by means of a capacitor and a resistor arranged in parallel. The capacitor represents the compliance due to blood vessels elasticity and the resistor represents the dissipative nature of peripheral vessels. Hence, the pressure-flow relation at the aorta is described by two parameters or elements, as shown in the equivalent circuit below:

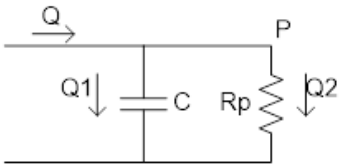


Figure 6. Electric representation of the 2-element Windkessel model.

Applying Ohm's and Kirchhoff's law as stated previously in this chapter the circuit can be solved and the pressure-flow relationship is described in the discrete domain as:

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

The full mathematical development can be found in Annex A.

Years after this model was proposed, aortic flow could be measured thanks to the development of the electromagnetic flow meter [14], and it revealed the limitations of this first Windkessel version. The 2-element Windkessel model fails in describing high frequency components; it explains aortic pressure decay in diastole but shows discrepancies with the measured systolic pressure.

2.4.2. 3-element Windkessel model

Some years later, Westerhof [17] introduced characteristic impedance as a new element to predict the pressure in systole. The new element can be considered a link between lumped Windkessel and wave travel, as the impedance equals the wave speed times the blood density divided by the blood vessel cross-sectional area. A resistor element represents characteristic impedance, since they both have the same units, and the final circuit can be observed in Figure 7. However, this simplification of the input impedance causes small errors in the low frequency range [17]. The good thing about the 3-element Windkessel model is that the additional resistor improves the simulation of the flow and

pressure through the entire cardiac cycle and the high frequency components are accurately predicted.

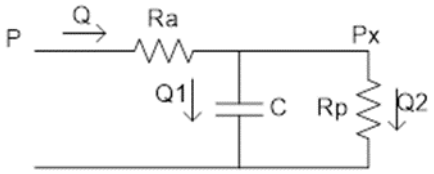


Figure 7. Electric representation of the 3-element Windkessel model.

Solving the electrical circuit in the same way as before:

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

The full mathematical development can be found in Annex A.

2.4.3. 4-element Windkessel model

This Windkessel configuration incorporates a new electrical circuit element, the inductance, which describes the inertial or time delay effects due to the mass of the fluid. It accounts for inertial effects and depends of the geometry of vessel and the density of blood:

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

The 4-element model is the most complex one: Stergiopulos [18] added an inductance element to represent the total arterial inertance, since it describes better the impedance of the blood vessels and decrease errors in the low frequency range.

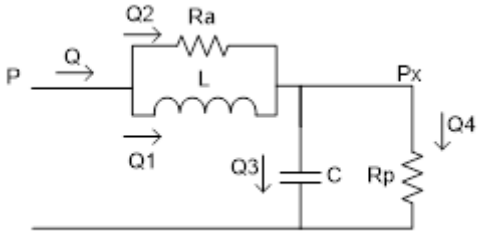


Figure 8. Electric representation of the 4-element Windkessel parallel model.

The solution to the circuit represented in Figure 8 is:

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

The full mathematical development can be found in Annex A.

The inductive element can be included in series or in parallel with other elements; for instance, the series inertance has effect only in the arterial impedance at high frequencies and not at low frequencies. Although this model includes the greater number of parameters, there is an open debate on which model is the best one, whether the 3- or the 4-element models, since the inertance is estimated with difficulty and can lead to error [17].

The comparison of measured and predicted aortic impedance for the different Windkessel configurations is quantitatively represented in Figure 9, which gives an idea on the strengths and weaknesses of each model:

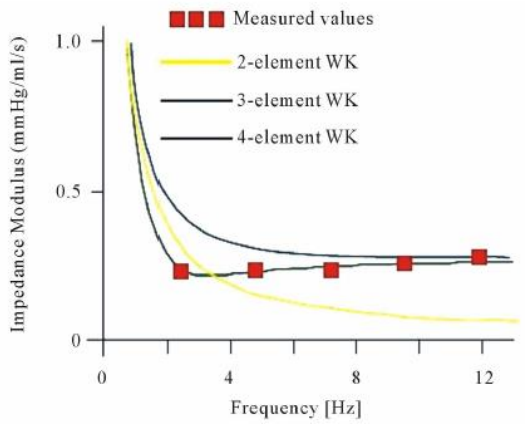


Figure 9. Comparison of predicted and measured aortic impedances [12]

2.5. State of the art

This section analyzes the existing options of models and tools to help in the understanding of cardiovascular system, especially in academic environments.

One of the most used tools in education are the anatomical models. They are intended to be used by medicine or biology professors to help students in their classes, as well as

to facilitate to physicians explanations to their patients. They have evolved from simple shapes to incredibly detailed physical reconstructions made of high quality materials.

Many companies worldwide offer a wide assortment of 3D models, including the heart and circulation models [19] [20]. But they have limitations: although these structures may be very useful for medicine students, they are not so useful describing multiple pathologies in real-time. For instance, the model shown below is relatively expensive and is only useful to visualize atherosclerosis.

Figure 10. Atherosclerosis anatomical model with transversal arterial section, 2 pieces [21].

To know how the internal structures dynamic properties vary, simulation tools are needed. One of the most complete tools are the manikin-based simulations. Current manikins offer a long list of features to simulate many scenarios. They are connected to a computer that can display blood pressure, arterial wave forms and pulmonary artery wave forms, among other parameters [22]. One of the most realistic and innovative emergency patient simulators is the SimMan 3G [23]. It can display both neurological and physical symptoms, and regarding vascular system there are several accesses to vessels and blood pressure, ECG rhythms and pulse are displayed.

The real-time information they provide is realistic and prepares for real emergency situations, but they still do not explain the vascular hemodynamics and they are not very useful for people with poor medical background. In addition, manikins' cost is very high.

Simulation based on mathematical models is used both for training and for research to achieve prevention of cardiovascular diseases. There exist highly-developed software tools for vascular simulation, like the SimVascular open source software [24]. After years of work of several university groups, the latest software version enables users to run simulations using models and boundary condition values included in a library of computational models. The software includes graphical user interfaces that enable to create 3D models from previously segmented medical imaging data; then meshes are

generated and finally there is a simulation tool using three-dimensional incompressible Navier-Stokes equations [25] to describe vascular pressure, flow, resistance, etc.

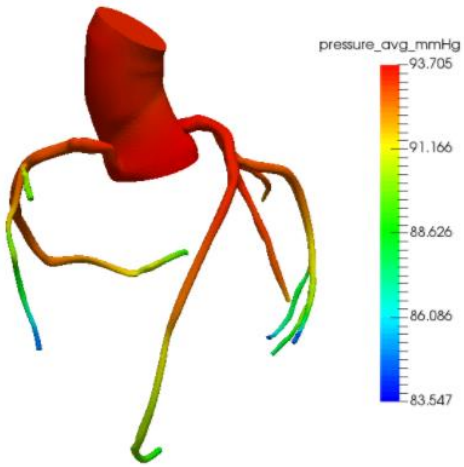


Figure 11. Pressure from coronary simulation [25].

This tool and other similar ones are more sophisticated than the one developed in this thesis, since the mathematical models they use are three-dimensional distributed models; but they could be used for the same purpose and introduce the concept of mathematical modeling to provide a more technical point of view of human physiology.

MATLAB community offers a wide variety of resources including toolboxes, courses, documentation, forum, etc. in which users worldwide contribute with their knowledge in MATLAB programming. One example is the Simulink course called "Quantitative Human Systems Physiology" developed by Dr. Bradley Sutton, which includes a cardiovascular module. Arterial system is modeled in two ways: first, examining only a compliant artery and secondly including also the left ventricle. Both are explained by simple differential equations based on the mathematical models presented in Hoppensteadt and Peskin "Modeling and Simulation in Medicine and Life Science" [26]. This is just one of the latest examples available from the MATLAB web page related to physiological simulation and its connection with the generated code.

Computational simulation applications go beyond training under- and post-graduated students. Once models are validated, they are a very useful tool for research and even for customized medicine. New models explain the impact of surgical vascular interventions on blood flow and predict the consequent repair. These simulations predict at micrometric level some physical variables such as velocity and shear stress, which is useful in stent design where such precision cannot be obtained [27].

However, mathematical models are not ideal, and some attention has been put on comparing both, physical and numerical models, as in the present work. For instance,

Barry J. Doly and his team focused on studying the aortic aneurysm rupture locations comparing the results predicted by a mathematical simple model and the ones obtained from idealized silicone models. The predicted locations from both models were similar, but more suitable physical models were required to improve the method so that it could be applied to patient-specific models [28].

Hybrid modeling of the circulatory system is achieved by merging numerical and physical models. This approach enables to connect the mathematical equations describing the circulation in a computer program to a physical part by means of hybrid interfaces such as impedance transformers (converters) that serve as gates. It improves repeatability and long-term stability while reducing costs and enhancing flexibility [29].

3. PROJECT DEVELOPMENT

This project is focused on continuing a preliminary work done by Nuria Peña Pérez, a biomedical engineer from UC3M, in her bachelor thesis [5]. Therefore, not every part has been developed from scratch, but optimized instead. The reader may find some references in the manuscript that help to understand and separate the contributions of each student.

3.1. User requirements

The complete educational tool is composed of two different platforms, both based on the Windkessel model:

1. Theoretical Windkessel simulator:

It consists of a numerical simulation of aortic pressure and flow waveforms based on the 2-, 3- and 4-element parallel Windkessel models. The equations defining these model configurations were solved and integrated into a software platform that allows the user to modify parameters to understand their contribution in cardiovascular diseases and conditions.

2. Physical Windkessel simulator:

The employed device is a pneumatic-hydraulic simulator of the cardiovascular system. It has a compliant and a resistance elements modeling different components of the vascular tree, according to a 2-element Windkessel configuration. Further details on the system design can be found in 3.3.1. Data from sensors placed in the physical device are transmitted to the computer and then processed and integrated into a second software platform that acquires, processes and displays data from the physical system into the computer. The final interface's role is to display the real-time pressure and flow waveforms data.

User requirements were planned with the supervisor of this project, Manuel Desco, who is also the responsible for the physical simulator. The users will be the professors and students, and therefore the requirements were established thinking on the teaching and students' needs.

To comply with those needs, the hardware and software requirements are listed below:

General requirements:

- To use MATLAB programming language for the software parts. MATLAB was the language employed in previous projects and the language that Biomedical Engineering students at UC3M learn and commonly use in practices and homeworks during the degree. Although students are not required to program anything, it is convenient that they feel comfortable with the environment they are intended to use.
- To display exact data in all the interfaces, with proper units allowing the user to modify parameters without possibility of error.
- User interfaces must be user-friendly and with the same layout since they will be part of the same laboratory tool. It must be clear the common environment of both tools to better enable the comparison between physical and mathematical models.
- A new practice guide for students must be written. It should include a brief introduction to the Windkessel model in an understandable way and a clear separation between the two parts of the practice, theoretical and physical. It should give simple explanations of the mode of use of the device and suggest questions in concordance with the background for physicians and engineers. Finally, the student must be asked to compare both models and see which one is better depending on the situation.

Specific requirements:

Theoretical Windkessel simulator requirements are:

- To ensure the correctness of the full mathematical development of the models and its related MATLAB program.
- To include comparison of pressure and flow, incorporate the option of saving the displayed figure for later quantitative analysis and display predefined default data in any situation.

Physical Windkessel simulator requirements:

- To diagnose problems and repair the previous physical system.
- To develop a new communication protocol between the sensors and the computer, in order to speed-up the acquisition process, to provide more precise data and to reduce computational burden.
- The data must be displayed in real-time and a .txt file should be generated to store incoming data for possible further processing.

3.2. Theoretical Windkessel simulator

3.2.1. Theoretical simulation program structure

When this project began, there was an initial interface that simulated the arterial Windkessel numerically. The equations defining this lumped model are solved by using Kirchhoff's and Ohm's laws, finding the pressure as a function of the flow. The 2-, 3- and 4-element parallel models were included in a simple MATLAB interface. The present project started from that point, to finally produce an updated version based on the Windkessel model in a user-friendly interface.

The model proposed by Otto Frank is not new, hence the mathematical equations relating flow and pressure have been computed in many ways. The first step for this work was to revise the previously developed full mathematical development, understand it and correct several small mistakes that could be the result of fast typing or format errors. The full corrected mathematical development can be found in Annex A.

The next step was to integrate the model solution into a MATLAB interface. MATLAB is a high-level language of technical computing known worldwide and it has a broad range of applications: machine learning, signal processing, image processing, etc. [30]. It offers a tool called MATLAB GUI (Graphical User Interface) for building applications with user interfaces, so that there was an option to create a customized interface based on the Windkessel model.

Concerning the programming structure, the final Windkessel tool is organized into eleven MATLAB functions:

- Six functions for the implementation of the mathematical Windkessel model, with two functions for each of the models (2-, 3- and 4-element models). These functions perform the calculations required for plotting the pressure, flow, pressure flow loops and pressure and flow together. They include the conversion factors necessary to go from medical units to International System units.
- Three functions to program the GUI of each of the three mentioned models. These functions define the three windows appearance and functionality by using subroutines (callback functions) that at first define the objects and then how they will work. They are the bulk of the work of this thesis and the most extensive ones.

- One function describing the main menu window of the GUI. It is the one the user must run to start the tool. Initial variables are defined as well as the main window containing three push buttons a pop-up menu.
- One function to code the option of saving the currently displayed figure which will be stored in a .fig file.

All MATLAB scripts have been programmed following MATLAB style guidelines as reference [31] [32]. Programming guidelines aim is to help producing code that is comprehensible for both the reader and the programmer. Software programming conventions assure correctness and clarity and facilitates sharable and maintainable scripts [32] [33]. This is particularly useful for this project, since the developed tool could be shared for educational purposes and there might be future work to add new lumped models.

Annex D includes more information on the developed programs.

3.2.2. Theoretical simulation features

The user, in this case the Biomedical Engineering student, is expected to modify the parameters that intervene in Windkessel mathematical model thanks to the developed tool and then to interpret the results. To facilitate this task to students and to develop a newer and more exact tool several changes have been accomplished. It has been possible to add new functionalities thanks to the modular structure of the codes. These modifications will be explained from this point on with the help of some screenshots to illustrate them.

The complete tool has five different windows: a welcome window, three windows for the modification of the 2-, 3- and 4-element parallel Windkessel models and a window for saving the displayed figure. In each of the Windkessel windows the user can choose to display the aortic pressure, aortic flow, aortic PQ-Loop or pressure and flow together. The parameters defining each model can be modified and two buttons to reset to default values and to save the represented curve are available. A pop-up menu to change between models and to return to the main menu is present in all the windows.

When the interface opens, the welcome window appears and allows the user to choose one of the models by clicking one push buttons. The user also can choose the model to be employed from the pop-up menu mentioned a few lines above. The following figure shows the main menu window:

Figure 12. Main window of the theoretical interface, to 'Welcome' the user.

When the user presses any of the buttons, this window disappears and the panel associated to the selected model appears. Some default values have been predefined and the resultant pressure waveform corresponding to these values according to Windkessel model is shown. The default values have been chosen based on the range of normal values for a young healthy man. In Figure 13 the pre-defined *Aortic pressure* curve for the 2-element model is shown. When the user selects any of the other variables to display (*Aortic flow*, *Aortic PQ-Loop* and *Pressure and Flow*) from the selection panel the default values for that variable are shown.

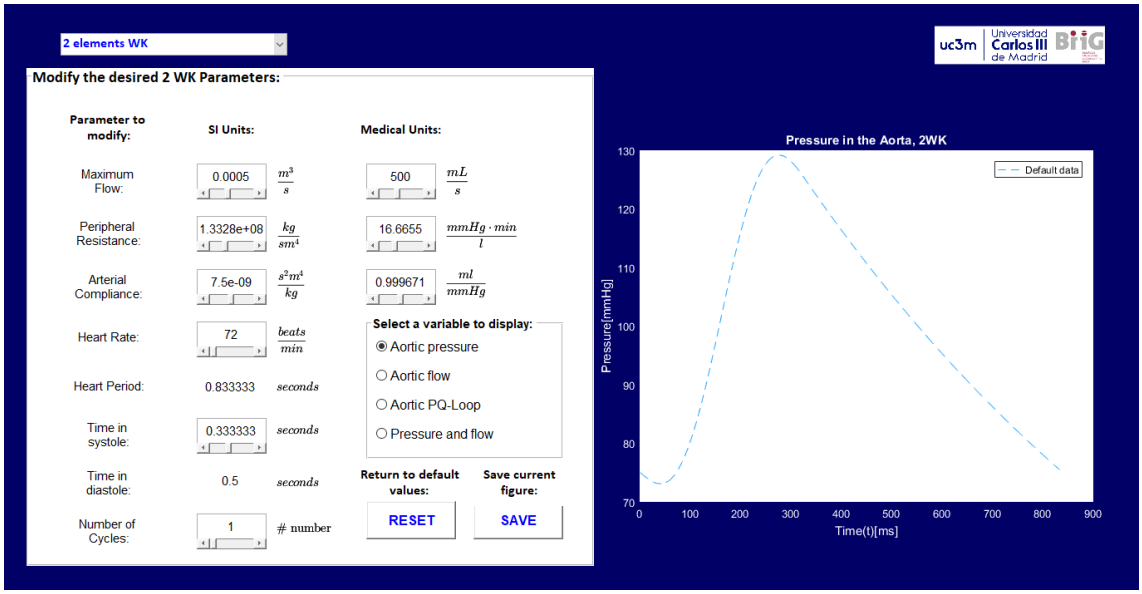


Figure 13. 2-element WK panel when it is just opened.

One of the requirements regarding this issue was to keep this reference line visible together with the waveform resulting from the user's modifications. This was thought to help data interpretation. For example, decreased aortic compliance is related with systolic hypertension and therefore with increased cardiovascular risk [12]. It will be easy for the user to detect the increased pressure if both, the normal and the altered pressure waveforms are displayed at the same time.

Thus, the default data is displayed together with the current data for every variable in all models, so that any deviation from standard values is visible.

Another upgrade consisted on indicating the systole and diastole phases for pressure and flow variables; it makes no sense in the PQ-Loop case as there is not time axis in this curve. As explained in chapter two, the systole and diastole are the two phases of the cardiac cycle. The identification of each of these phases is very useful to understand the hemodynamics of the vascular system, and that is why it was thought to be helpful to include it. The identification of the flow is easy: according to Windkessel model, during diastole there is not flow. However, in the case of pressure it is not so clearly deduced. Thus, there are vertical dashed lines crossing the displayed data at the instant of time of each change in phase and each period is properly labeled. Figure 14 shows it in an example:

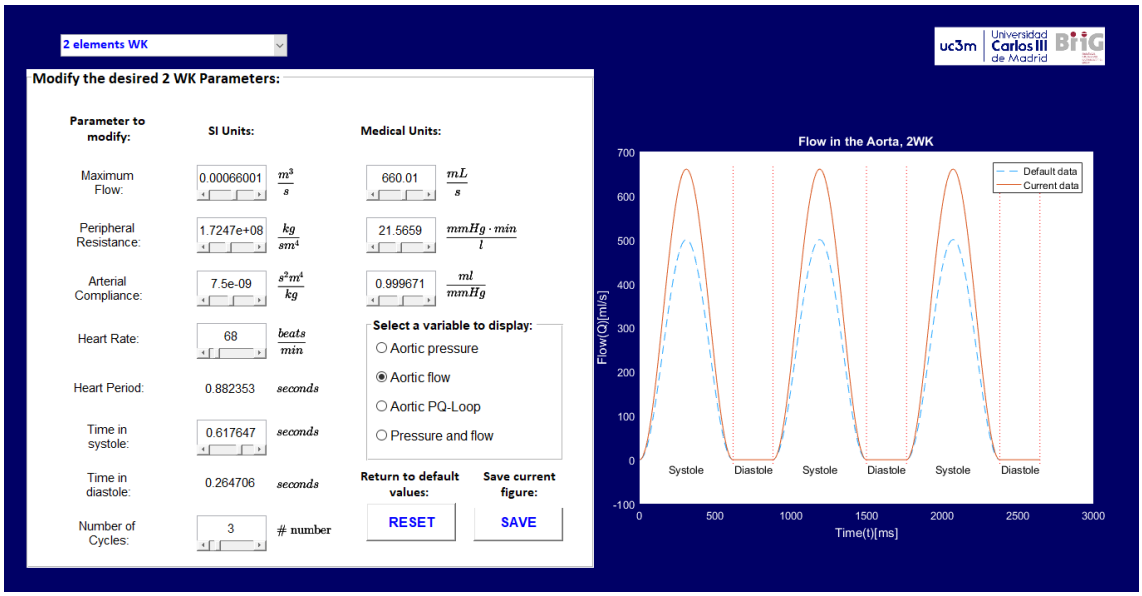


Figure 14. Flow in the 2-element WK window.

The updated version of the interface includes a fourth option in the selection panel. The user can select the possibility of representing both the pressure and the flow at the same time. It thus allows the user to compare the progression of the two waveforms together, contextualizes them and reinforces the idea of trying to better understand the systole and diastole concepts. The observation of both variables at the same time allows the

student to think: when the blood flow goes to zero in diastole, the pressure starts to decrease and when the flow starts to increase at the beginning of systole, the pressure increases with it; thus, when blood passes through the aorta, it generates some pressure.

The next figure shows how it is also useful to display more than one cycle to see the pressure variations and trend for the selected parameters' values.

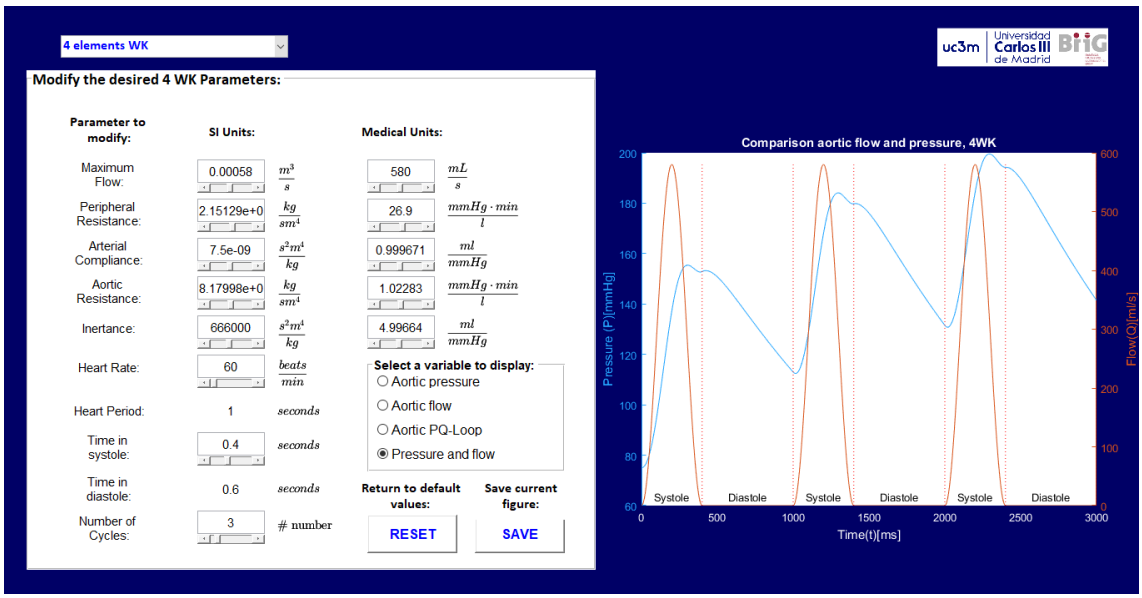


Figure 15. Aortic flow and pressure for the 4-element WK.

Another implementation has been to include the option of saving the currently displayed curve. When the user clicks the *SAVE* push button a new panel opens that asks the user for the directory where to store the figure and the name of the file. The user has to introduce two valid strings and click on *Save figure*. Automatically a .fig file is generated and the panel closes so that the user can continue working with the simulation interface.

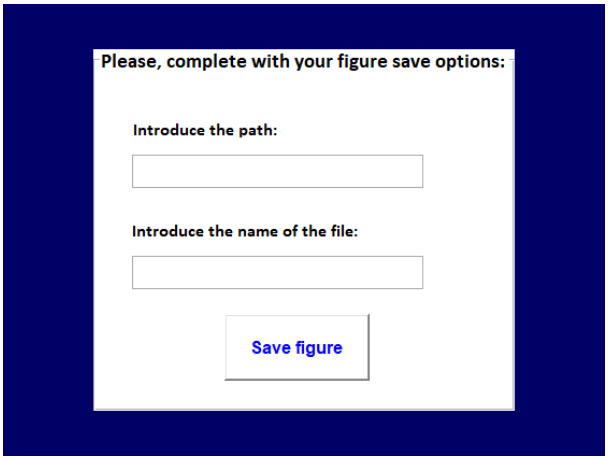


Figure 16. A new window appears when the user clicks SAVE from any of the model windows.

Biomedical Engineering students will answer to the questions presented in the cardiovascular practice guideline created for this project and included in the Annex B. That is the reason to add the *SAVE* option: .fig files have many tools for further analysis (zoom in and out, select data, change figure properties…) that may be helpful.

Another new feature of this simulation tool is the possibility to return to the default values from any variable. The user just must press the *RESET* button, and all the modifications up to that point will be discarded. All the parameters are set to their initial values and the displayed variable resulting from applying the Windkessel equations is shown.

The main utility this MATLAB GUI is to be useful for Biomedical Engineering students to better understand cardiovascular physiology. Therefore, the idea has been to develop a user-friendly tool with added value with respect the previous one. The incorporation of these new features has led to the change of the older layout design to a newer one capable of including all buttons, visually simpler and more attractive.

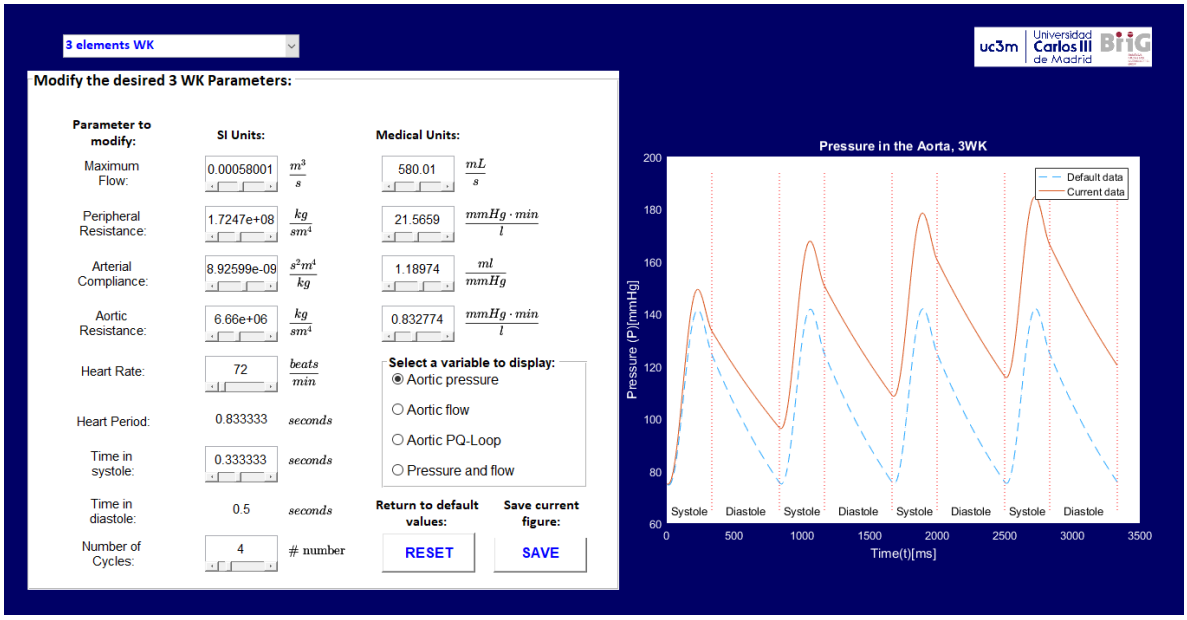


Figure 17. Four cycles of aortic pressure for the 3-element WK.

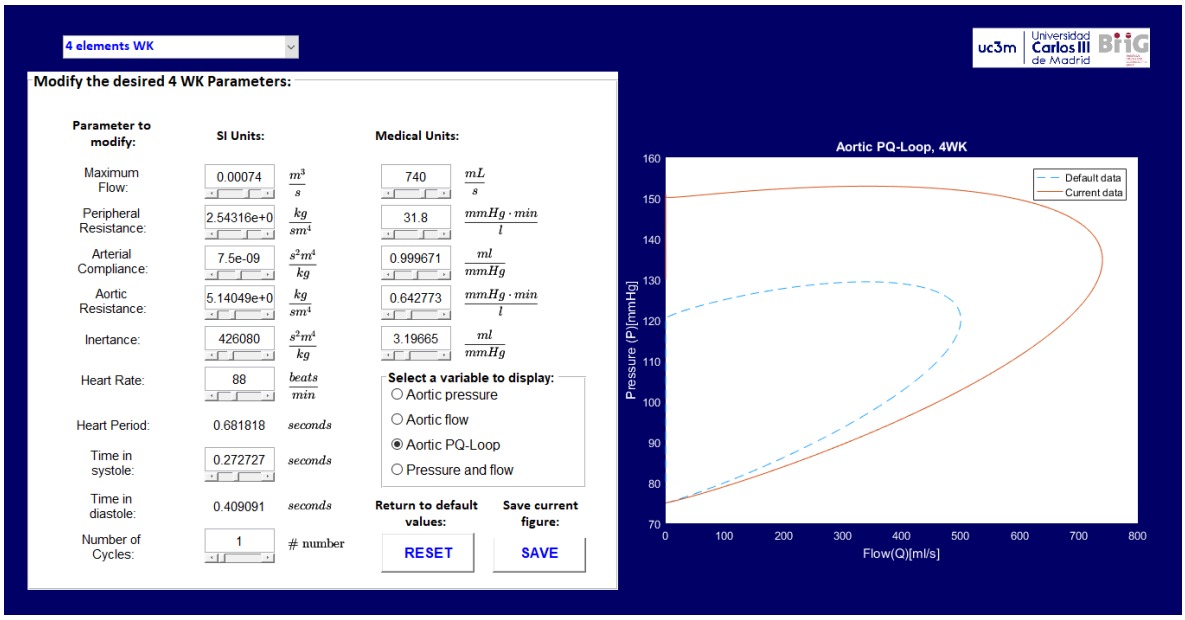


Figure 18. Aortic pressure-flow loop for the 4-element WK. Pressure-volume loops are important to analyze the cardiac cycle, but as Windkessel model input is the flow, only pressure-flow loops can be represented.

3.3. Physical Windkessel simulator

3.3.1. Description of physical device

The second part of the work consists on the optimization of a pneumatic-hydraulic prototype that simulates the arterial system by means of a physical 2-element Windkessel model. The device is located at the Bioengineering Laboratories of the Polytechnic School of Universidad Carlos III de Madrid. It was initially manufactured by the company SEDECAL (Sociedad Española de Electromedicina y Calidad, S.A.), a Spanish company, leader in OEM design and manufacturing of high frequency X-Ray generators and X-Ray systems, among other areas of expertise [34].

The initial prototype never worked very well. Even if it worked, there was not a communication protocol between the simulator and the computer, so no data could be obtained. But some university professors thought it could be useful for some laboratory practices and it has been kept at the Bioengineering laboratories.

The platform consists of several subsystems: an electronic controller, an electric circuit and a hydraulic-pneumatic circuit, which work together to model the arterial system according to a 2-element Windkessel model. When the author began this thesis, the device was composed of many elements; the main ones corresponding to each subsystem that are listed below and highlighted in Figure 19:

1. Main switch

2. Membrane pump simulating the heart
3. Pressure-vacuum regulators (rotatory knobs)
4. Heart cycle controls, allowing to manually modify the heart rate and duty cycle
5. Retention valves
6. Compliant element modeling the elastic large arteries
7. Water fluid Reservoir
8. Peripheral resistance modeling smaller vessels
9. Flow transducer, recording flow data just after the peripheral resistance
10. Flow port, sending data to the computer
11. Pressure transducer (11' is still part of the pressure transducer), recording simulated aortic pressure
12. Pressure port, sending data to the computer

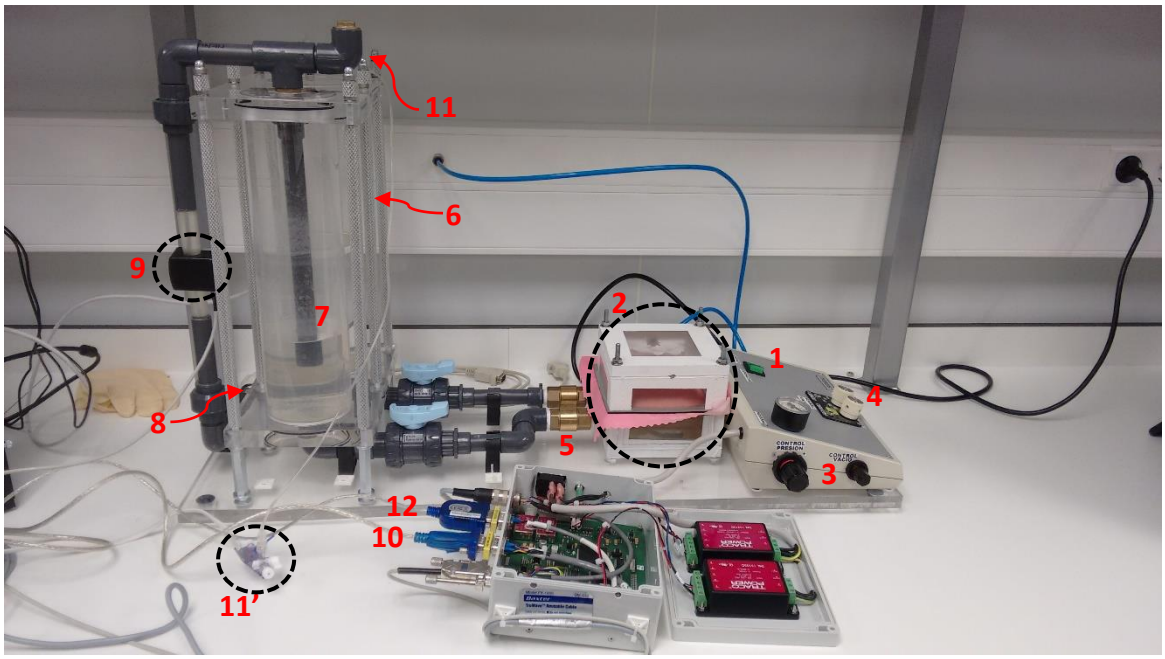


Figure 19. Original physical device. This photograph is prior to the software control mode inclusion. It has been not possible to find documentation after its inclusion and previous to this thesis.

In this simulator water substitutes the blood. It circulates through the system and some data of interest are recorded by means of two sensors placed at different points of the device.

The fluid reservoir contains the water that will circulate when the device turns on and ensures that there is water always available in case of escape or failed interconnection between the two columns. If the faucet is open, water may flow through the PVC pipe and reaches the part of the system modeling the heart.

The heart membrane is just a latex glove, since it is an elastic material but strong enough to move the liquid. It is a very simplified version of the heart muscle walls, ignoring that

the heart is a two chambers organ. However, it is enough to produce a realistic pulsatile pumping and it forces the water to go out the box through the other pipe with a periodic movement set by the user.

The pressure-vacuum regulators are responsible for inducing this movement: an alternating pressure-vacuum state causes the glove to move up and down, which corresponds to the diastolic and systolic phases, respectively, of the cardiac cycle. Some air is introduced into the heart when an electric valve is activated, causing the membrane to move downwards and pushing the water out of the box to the rest of the hydraulic circuit. The introduced air is then removed from the heart chamber when the valve is deactivated and some water enters again causing the membrane to move upwards. This process periodically repeats at the time specified by the user with the heart cycle controls. The user can choose the strength of this oscillation by playing with the rotatory knobs, but the heart should oscillate always around the middle of the box to avoid a "heart attack"; in practice, if the glove is excessively forced it can break and it would need to be substituted.

The pressure regulation referred in the lines above does not correspond to the aortic pressure measured in the compliant element, but to the method of controlling the heart pumping. It would represent the modification of the cardiac output instead of any pressure.

The heart cycle controls also play an important role. When the heart membrane oscillates as desired, these controls must be regulated to simulate a specific heart rate and duty cycle. The initial electronic implementation to achieve this control included a LM 555 timer, which was in charge of activating and deactivating the previously mentioned electric valves inducing the pump. Then, as the user modifies the heart cycle rotatory knobs, which are two potentiometers, the timing of the circuit is also being modified.

In the system previously upgraded the heart rate and duty cycle could be selected via software, instead of manually moving any control buttons. There is a switch allowing the user to select either the manual or the software control mode. In this second case, the serial lines of the sensor are used for transmitting the timing information. The voltage of the serial interfacing pins is set to HIGH or LOW alternatively to activate and deactivate the electrical valve. Only the DTR pins of the serial interface not being used at that moment can be used for the timing control. For example, if pressure data is being acquired in software mode, the flow serial lines are the ones alternatively set to HIGH and LOW to generate the beat. Therefore, it is not possible to record data from the pressure and flow sensors at the same time.

The pressure-vacuum regulators and heart cycle controls electronics are inside the control case, whose complete arrangement can be found in Figure 20. It can be observed how the compressed air is generated and the connection with the electronic board necessary for the control via software.

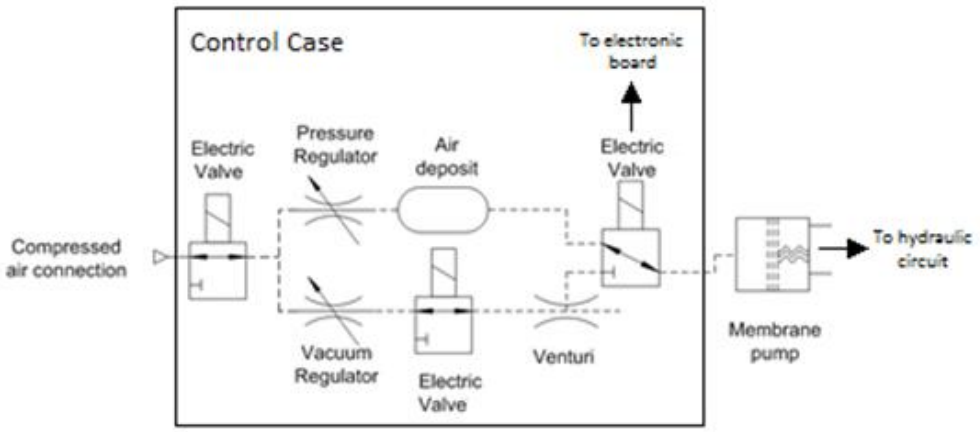


Figure 20. Control Case components. Scheme developed using Microsoft Office VISIO.

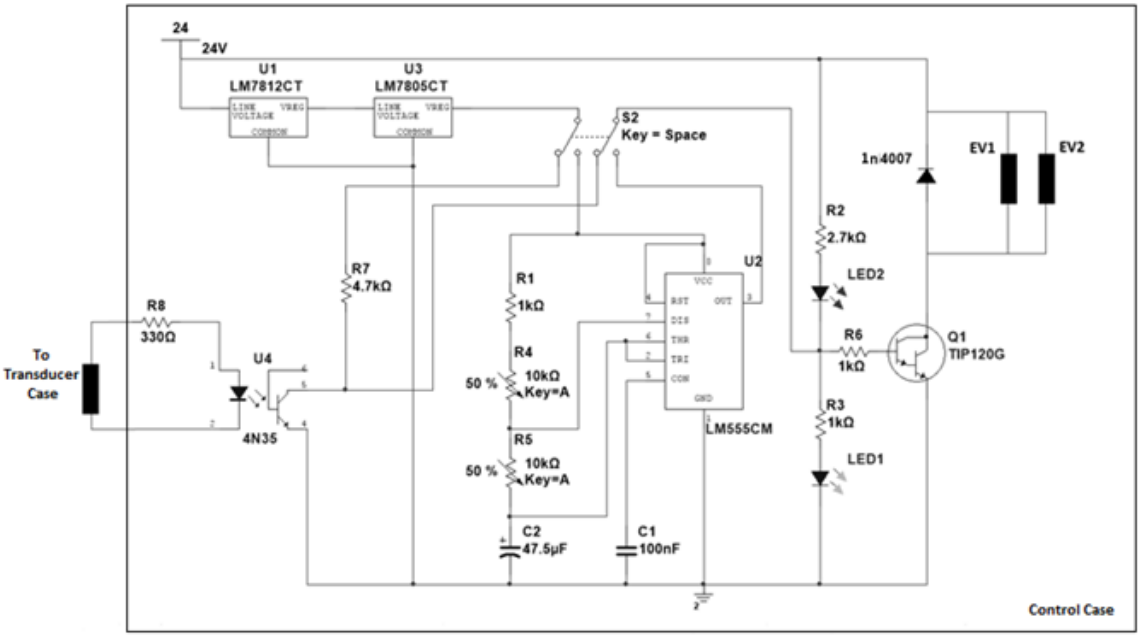


Figure 21. Electronic circuit inside the control case. The switch, LM 555 and connection to transducer case can be observed. Circuit designed using MULTISIM.

Continuing with the hydraulic circuit, in every beat water goes out the heart box through the other PVC pipe and reaches the second column; it is possible because of the presence of two brass retention valves that assume the role of the semilunar valves by preventing backflow firstly to the reservoir fluid and secondly to the heart box. Therefore, the water movement only has one direction as desired.

The second column is a compliant element modeling large human arteries like the aorta. The water going in and out generates a pressure that is recorded by a pressure sensor place on the top of the column. The pressure transducer is connected to an electronic module that transmits pressure waveforms to the computer via serial interface. The electronic details and communication protocol with the computer will be explained in section 3.4.

The water does not accumulate and fills the second column, but goes again to the fluid reservoir. Both columns are communicated through a needle valve that is half open. Needle valves are a type of valves allowing precise regulation of flow at low flow rates as the ones in this system. The conduct communicating the two parts is the peripheral resistance element modeling smaller vessels. There is a flow sensor placed in this pipe, connected to a flow measuring board which transmits the signal by serial communication, same as the pressure board. In section 3.4. the description of the sensor and the module is included.

Finally, inside the transducer case there is an electronic board including several elements. The sensor boards require 5 V to work, hence there are two voltage regulators (from 24 to 12 V and from 12 to 5 V) to accomplish this level. The board includes heat dissipation components and secure connections. The transducer boards are also inside the transducer case, and they connect directly to the computer via RS-232 interfacing. Since the computer does not have this type of connections, two USB to RS-232 serial adapter cables round the physical device design out.

3.3.2. Fixing problems and improvements on physical device

One of the steps required in this project was to fix a breakdown of the device. After the rupture of the membrane, water reached the control and the transducer case. Most components could be dried and cleaned and some connections were substituted and most wires welded again.

The second problem was that the retention valves became obstructed because small pieces of the latex glove got stuck on them, and the valves themselves were also damaged. If the only problem were the glove residues, it would be solved by cleaning the non-return valves and leaving them unobstructed. However, the water did not flow so they were replaced by new ones.

In order to replace the valves and any other piece of the pneumatic section it was necessary to disassemble the whole device, which difficulted the maintenance and was time-consuming. Once the system was disassembled, it was decided to add some extra components to facilitate the maintenance by making the device design as modular as

possible. Different plumbing pieces were integrated, with restrictions due to the space limitations. For instance, 3 brass connections were incorporated to facilitate replacement of parts of the pneumatic-hydraulic subsystem.

Enhancements also included the renewal of gaskets, brass pieces, latex glove, PVC pipe and stainless-steel rods. These materials substituted worn-out, broken or less suitable prior materials. Therefore, the chances of mechanical problems have been reduced and the present design is more modular.

Regarding the retention valves, to avoid their obstruction due to residues some filters were placed at the entrance and exit of the box where the membrane is located. These filters were envisioned to stop any small pieces of glove that may break, dirt or rests of the silicone used to seal some pieces. The filters were 40 μm cell strainers manufactured by the Biologix group, coupled to the system thanks to a 3D printed ferrule piece, manufactured by courtesy of the mechanical department of the university. As expected, the flow resistance increased, which is incompatible with the system purpose and they were finally not included in the final design. The problem remains unsolved, which means that the latex glove must be periodically replaced.

One possible question regarding the device is whether the 2-element Windkessel model is good enough for the project purposes or more elements should be added to improve its accuracy. It is true that the original Windkessel model is not the most complex model, but it is simple enough to explain it during one lesson to Biomedical Engineering students. Measurements provide good approximations on the pressure waveform compared with the literature, although it would not be acceptable for clinical training. Since the purpose of the project is to develop an educational tool and the system is just part of the whole tool, the system is considered valid, without adding more elements.

3.3.3. Communication protocol: program structure

Previously to this work, a communication protocol between the sensors and the computer had been developed in MATLAB, the same programming language deployed in the case of the theoretical software platform.

In this software, the process of data acquisition and decoding was too slow, and the somehow obscure implementation did not follow the indications of the transducer datasheets. Therefore, this protocol has been completely rebuilt to a more general and robust one, structured in a modular way, same as the previous programs.

Concerning the program structure, the reading interface consists of a modularized structure of MATLAB functions:

- Two functions to acquire data coming from the flow sensor, for the manual and software pulse controls. These two functions include the instructions to communicate the serial port of the computer with the MATLAB software and the received data processing. They include subroutines (callback functions) to regulate the acquisition and decoding and the pulse timing in the case of software control.
- Two functions to acquire data coming from the pressure sensor, for the manual and software control mode. These two functions include the instructions to communicate the serial port of the computer with the MATLAB software and the received data processing. They include subroutines (callback functions) to regulate the acquisition and decoding and the pulse timing in the case of software control.
- One function describing the main menu window of the GUI. It is the one the user must run to start the tool. Initial variables are defined as well as the main window containing four push buttons and a pop-up menu to select the reading option.
- Four different functions that implement the interface layout. They correspond to each of the four options available from the main window: to read pressure or flow using manual or software control.

All MATLAB scripts have been programmed using the same MATLAB style guidelines as in the simulation interface program [31] [32] to facilitate further improvements.

Annex D includes more information on the developed programs.

3.3.4. Communication protocol: sensors and reading of data

Both sensors have a corresponding board located into the transducer case of the physical device. Boards communicate with the computer trough RS-232 interface but since modern computers do not integrate RS-232 ports, a RS-232-to-USB converter cable was used.

All serial ports send and receive data one bit at a time: the communication is bi-directional. Bytes of information are transmitted asynchronously using three lines: ground, transmit and receive. Therefore, it is possible to transmit data on one line while receiving data on another at the same time [35]. This enables to have a software pulse control. Four parameters define the way the communication between both devices takes place:

- Baud rate: number of bits transfers per second.
- Data bits: actual data bits in a transmission (for instance, 8 data bits conform one packet of bits).

- Stop bits: to indicate the end of the transmission for a single packet.
- Parity: to check errors in the communication.

RS-232 (Recommend Standard 232) is a widespread serial connection, somehow abandoned nowadays. This standard specifies the use of 25-pin or 9-pin connectors of type 'D', being the last type is the one used for this project. In scheme below all the connections and signal directions for the used pins are shown.

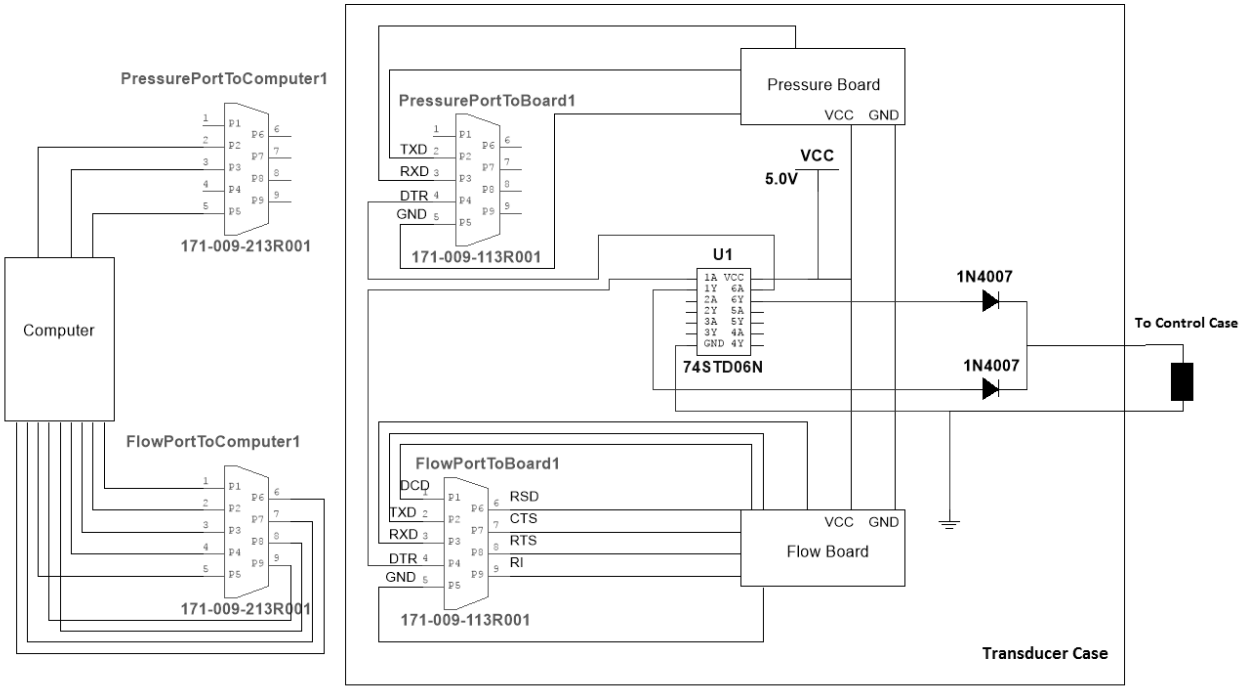


Figure 22. Electronic circuit inside the transducer case. Remember that the DTR pin of the transducer not being used for reading is used for sending a signal in the software control mode.

Although the two transducers use RS-232 serial protocol to communicate with the computer they code data use different parameters and encode data in different way. Thus, different routines were developed in MATLAB for the different cases.

Pressure transducer:

The module used is the EG02000, a two-channel invasive pressure module, although only one channel is used for this work [36]. The module is connected to a sensor with sensitivity of 5 μV/V/mmHg and zero range of ±70 mm Hg. Its pressure accuracy is ±1 % and the pressure range goes from −99 to +310 mm Hg, which is adequate for our purposes.

The board transmits three types of regular data packets: waveform packets, status packets and value packets. Each packet carries useful information such as the mean arterial pressure or the pulse rate using different number of bytes. These packets are

transmitted in 9-bit values since it is not economical to transmit 16-bit values for the pressures. Therefore, the receiver side is responsible for decoding the data so that at the end pressure values from −99 to +310 mm Hg are displayed. To generate these negative quantities, a value of −100 must be subtracted from all pressures. To calibrate to aortic values, +70 must be added.

Concerning the developed acquisition protocol, the first step to read the data is to open the port in the proper way. To meet the specifications of the technical manual, the serial port must be defined as 9600 baud rate, 8 data bits, one stop bit and no parity.

Then, a callback function executes when a specified number of bytes are available to be read in an input buffer previously created. The incoming bytes are automatically detected, read and stored in a variable.

For extracting the data of interest, a timer object with period of 0.1 seconds has been created. Every time it executes, it detects all data new from the previous execution and the "filtering" process starts. The pressure waveform information is contained in the waveform packets, hence they are the only ones interesting for this protocol.

	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 1 **Sync**	1	1	0	0	Bit 8 Wave 1	Bit 7 Wave 1	Bit 8 Wave 2	Bit 7 Wave 2
Byte 2 **Wave 1**	0	Bit 6 Wave 1	Bit 5 Wave 1	Bit 4 Wave 1	Bit 3 Wave 1	Bit 2 Wave 1	Bit 1 Wave 1	Bit 0 Wave 1
Byte 3 **Wave 2**	0	Bit 6 Wave 2	Bit 5 Wave 2	Bit 4 Wave 2	Bit 3 Wave 2	Bit 2 Wave 2	Bit 1 Wave 2	Bit 0 Wave 2

Table 4. Information contained in a waveform packet. The information of the measured pressure is contained in Wave 1, included in bytes 1 and 2 [36].

Therefore, the first step inside the timer callback function is to identify the type of packet being received to discard everything that is not a waveform packet. Then, as these packets contain three bytes, the identification of each of them is performed. The conversion of the values of interest from binary to decimal value and the calibration with some constant values gives the final pressure value. The obtained pressure values are stored in another variable and plotted in real time thanks to a global MATLAB loop. To decode a single pressure value the performed operation is:

$$Pressure\ value = decimal\ Wave1 + (binary$$

$$Sync(6) * 2\text{^}7 + binary\ Sync(5) * 2\text{^}8) - 100 + 70$$

Also, inside the timer callback function a text file is generated every time that it executes. The .txt file is constantly updated so that a sudden stop of the program does not

eliminate all the previous records. It is useful for students that may want to store the data to plot it later or analyze it quantitatively.

Finally, once the desired number of samples have been acquired, the serial port closes and the timer is stopped and eliminated.

For the case of software pulse control, the process is the same but with the addition of two more timer objects to control the cardiac cycle parameters. The DTR (Data Terminal Ready) line of the flow transducer is used to send information as the communication is bi-directional. The two timers are synchronized so that the DTR pin of the flow transducer is in the logic HIGH during systole and LOW during diastole.

Flow transducer:

The flow transducer is the DIGIFLOW unit manufactured by Em-tec [37]. It measures liquid volume flow by ultrasonic transit-time method. Its flow range varies from 0 to 9.99 LPM and flow resolution is 0.01 LPM, which is good enough for this simulation [38].

The transmitted information to the computer is a data string like the one shown in Table 5. The string of characters contains nine parameters separated by blanks. According to the user manual, the fourth parameter corresponds to the actual flow value in ml/min, which is the interesting data for displaying it. The rest of parameters provide information on the board temperature, calibration data, status information, etc. The data string finishes by a carriage return and a line feed terminator characters.

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

Table 5. String transmitted through serial port communication by the DIGIFLOW unit [38].

The signal from the sensor is transmitted to the DIGIFLOW board, and from there to the computer. Different MATLAB functions have been created to acquire data from this sensor and finally display the real-time flow in the reading interface implemented for this part of the project.

Same as in the pressure case, the first step is to know the settings of the serial communication port. To meet the specifications, the serial port is defined as 38400 baud rate, 8 data bits, one stop bit and no parity.

Therefore, a callback function executes when a terminator character is read and the MATLAB function is used to read the incoming data. Therefore, one string is read every time it executes and it is stored in a variable.

To extract the parameters of interest from those strings, a timer object with period of 0.1 seconds has been created, same as in the case of the pressure. In this case, it is only needed to take the parameter that is in the fourth position of all the new strings from the previous execution. The extracted data is divided by 60 in order to make a unit conversion (from ml/min to ml/s) and stored in a new variable.

As in the case of the pressure, within the timer callback function a text file is generated every time that it executes, in the same way and with the same purposes.

When the desired number of samples have been acquired, the serial port closes and the timer is stopped and eliminated.

For the case of software pulse control, the process is the same but with the addition of two more timer objects to control the cardiac cycle parameters. They are synchronized so that the DTR pin of the pressure transducer is in the logical HIGH during systole and LOW during diastole.

3.3.5. Graphical user interface

The final acquisition protocol is totally new, but the previous MATLAB GUI layout has been partially preserved. Some parts of the layout have been modified so that the appearance is the same for the two developed software platforms The main window that appears when the user executes the program is the following one:

Figure 23. Main window of the reading interface, to 'Welcome' the user.

The welcome window allows the user to choose between pressure and flow and between manual and software control modes. The user can click one of the push buttons to select

the desired experiment; these buttons are only available at the main window. The user also can choose it from a pop-up menu that is present in all the panels of the interface.

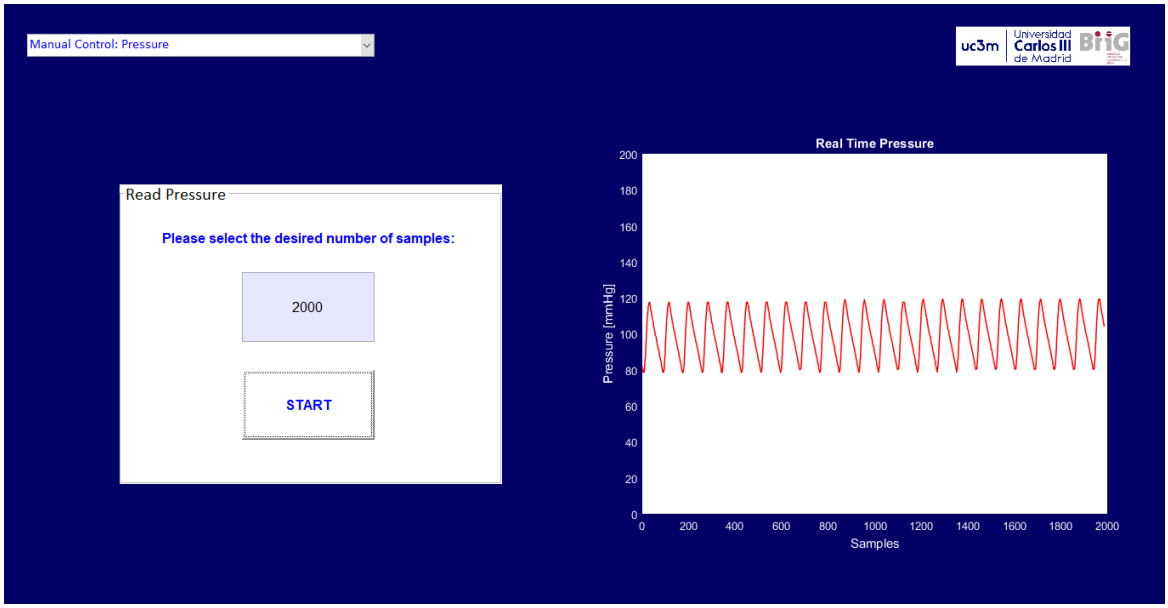


Figure 24. Manual control record of pressure (2000 samples).

The user can choose the number of samples to read from the sensor. The record can be constant if the rotatory knobs of the device are not altered as in Figure 24. Real-time modifications are shown if knobs are rotated during the record, like the flow record in the window below.

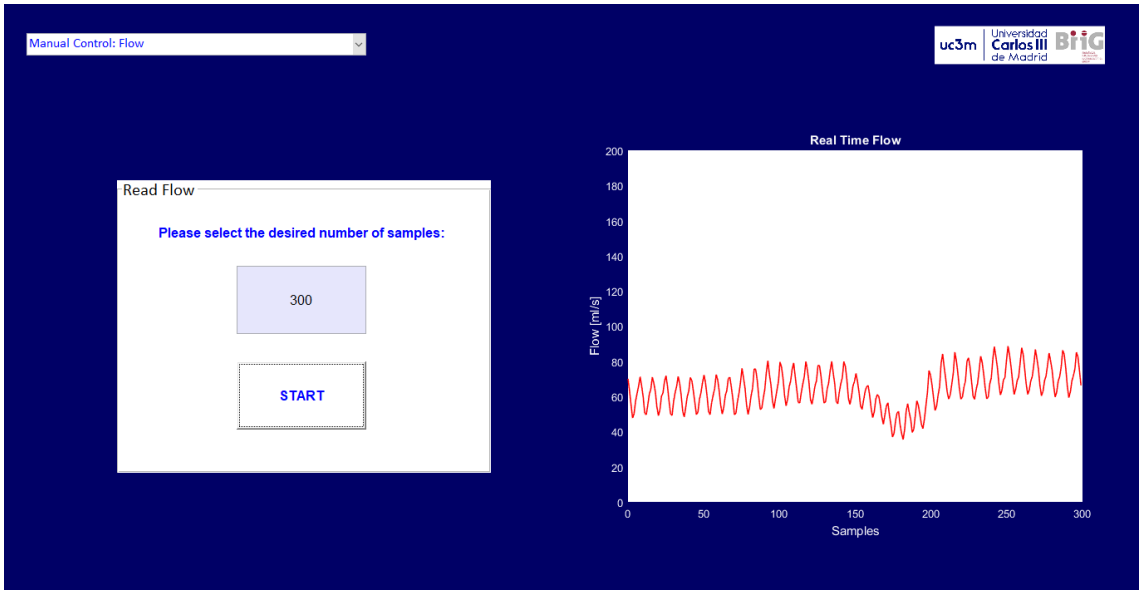


Figure 25. Manual control record of flow (300 samples).

In the case of software control, the user can choose between three options for heart rate and duty cycle. These are the options to send the pulse through the serial line at the moments defined as systole and diastole. The plain pattern at the first samples

corresponds to the time it takes for the user to turn the rotatory knobs. At first, vacuum regulators are set to zero and they must be quickly set to zero once the record is finishing.

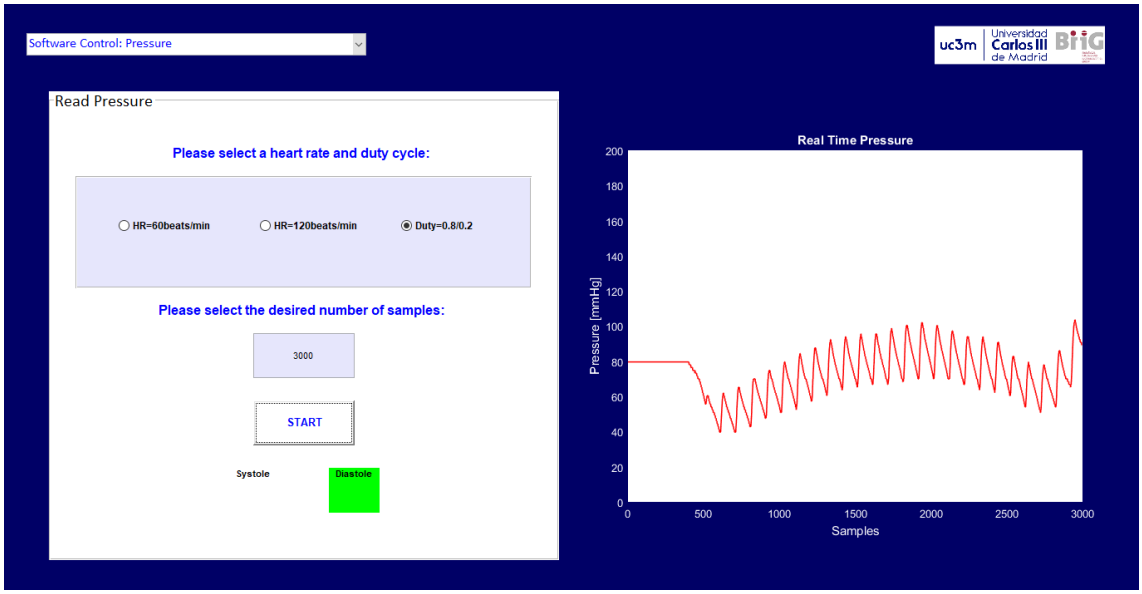


Figure 26. Software control record of pressure (3000 samples).

3.4. Laboratory practice guide

According to the user requirements, a new laboratory practice guide has been written, consisting of the following parts:

- Equipment used. It includes a brief presentation of the materials used for the practice.
- Practice outline. The core of the practice is divided into 5 sections:
  - Introduction: theory of Windkessel model.
  - Section 1: instructions of use of the theoretical simulator and questions related to it.
  - Section 2: instructions of use of the physical simulator and its associated interface and questions related to it.
  - Section 3: questions regarding the recording data, to evaluate students' comprehension of the model.
  - Section 4: questions comparing data from theoretical and physical simulator.

The complete practice guide can be found in Annex B.

4. DISCUSSION

4.1. Legal framework

Regarding possible intellectual property, this work does not involve any invention. The prototype was acquired years ago by the BiiG group and although it has been modified for specific purposes, there is not novelty for the market. Regarding the software platform, the mathematical procedure is not new and the purpose is that it can be used by the students that may need the tool, without author's benefit. Therefore, there is no need of patent or copyright protection.

In absence of the first manufacturer and as considerable modifications of the system have been accomplished, authors of projects related to this system can be considered the new manufacturers. It is important to consider the standard UNE-EN ISO 12100:2012 [39]. It guarantees that a machine or device is safe and evaluates risks associated to its use. During manufacturing, mechanical, electrical and thermal risks are present. To avoid these risks and to maintain the safety during the process it is convenient to consider a quality standard that helps to optimize the design and check that legal criteria are fulfilled. This methodology for safe process based on experience and knowledge has been useful for the new manufacturers during the process.

With respect to the location, the Bioengineering laboratories at UC3M fulfill all the safety standards stated by the Biosafety Comission of Bioengineering and Aerospace department. The set of specifications to maintain these quality standards are detailed in the safety handbook available for all the laboratory personnel.

No animals or hazardous substances were employed during the project development; therefore, this work is exempt from the Spanish order 53/2013 [40].

The Windkessel system is not thought to have an industrial application, but to be used in the Bioengineering Laboratories at Universidad Carlos III de Madrid for training students. However, it is interesting to consider the possibility of commercializing the device in the future, for it to be used in other universities. If the simulator is intended to be sold, it should obtain the CE marking. The European Commission affixes the CE marking to a product in order to declare that it meets all the legal requirements inside the European Economic Area [41]. To obtain the marking, the manufacturer has to provide the technical documentation accrediting that the product meets all requisites. The first step is to identify the requisites that are applicable to each product, since they vary between sectors. Secondly, the manufacturer is responsible of verifying that all legal requirements are fulfilled. There are two possible ways: either to follow the

corresponding standard during the manufacturing process or that the manufacturer provides the technical documentation related to the design of the device and to the possible risks derived from its use [42]. This process has not been followed during this thesis because the scope of this work was not the commercialization of the physical system, but it is interesting to start considering the idea and to carry out a deeper analysis on this aspect in a near future.

4.2. Socio-economic impact

Teaching is a challenging work, particularly in a degree like Biomedical Engineering that is becoming one of the most attractive degrees in Spain, as is demonstrated by the high score required to enter the degree. Many concepts taught during the degree are still far from the real situations engineers face during their career, and that is why practical simulations seem a good option to reduce that distance.

In a two-hour practice, students would be able to efficiently understand cardiovascular physiology and pathology, and would reinforce basic concepts of electronics and MATLAB programming. This type of practices perfectly reflects the multidisciplinary profile of biomedical engineers.

This project is not expected to have a direct impact on industry or society, as it is thought to be used as an educational tool. If the developed tools are found to be of interest, it is not discarded to commercialize the physical device so that other education centers could benefit from it, but this is still far from reality, and therefore there is not important socio-economic impact derived from this project other than a better teaching for the students.

Finally, this project does not pose any ethical issue, as it lacks any risks, animal experimentation or other ethical considerations.

4.3. Project budget

The next tables show the estimated budget breakdown for the project.

HUMAN RESOURCES			
Status	**Cost per hour (€)**	**Time investment (h)**	**Cost (€)**
Technical engineer	35	60	2,100
Supervisor (PhD.)	60	30	1,800
Biomedical engineering student	6	600	3,600
Total			**7,500**

Table 6. Human resources costs.

Work hours per month: 60

PERISHABLE MATERIALS			
Description	**Cost per unit (€)**	**Units**	**Cost (€)**
Non return valve ½	1.81	2	3.62
Brass link 3 pieces ½ M-F	4.11	2	8.22
Threaded fitting	0.30	4	1.20
Gasket	0.25	2	0.50
Teflon	0.20	2	0.40
PVC component	0.12	2	0.24
Tangit PVC glue	4.17	1	4.17
Latex glove	0.35	4	1.40
Stainless Steel bar	*	4	*
Cable	*	1	*
USB 2.0 Extension Cable	9.74	1	9.74
Total			**29.49**

Table 7. Perishable materials costs

*These materials were provided by the laboratory without any cost.

The following table shows the material used in the project with a depreciation of 20% at five years.

OTHER MATERIALS				
Description	**Initial cost (€)**	**Cost per year (€)**	**Dedication (months)**	**Cost (€)**
Windkessel prototype (SEDECAL)	5,000	1,000	10	833.33
Personal laptop	529	105.80	10	88.17
Internet	723.60	723.60	10	603
Microsoft Office	99	99	10	82.5
MATLAB software	119	119	10	99.17
Total				**3,313.17**

Table 8. Other materials costs

Indirect costs are the 20% of the human and all the material costs. Therefore, they are estimated to be 2,168.53 €.

SUMMARY OF COSTS	
Category	**Cost (€)**
Human resources	7,500
Fungible materials	29.49
Other materials	3,313.17
Indirect cost	2,168.53
Total cost without VAT	13,011.19
VAT (21%)	2,732.35
Total cost	**15,743.54**

Table 9. Summary of estimated costs

5. CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

The main objective was to set up a final version of an arterial system simulation tool for training students. The individual elements that form the complete device have been modified, replaced or optimized depending on the specific user requirements. The main goal has been fully reached by complying with those requirements and the final set-up is the result of the following small attainments:

1. The pneumatic-hydraulic system is working and ready to be used. Enhancements on the design have resulted into a safer and more efficient system less susceptible to failure and easier to maintain: components are longer-lasting and effortless to replace.
2. The device to computer communication protocol is successfully transmitting and receiving data from flow and pressure sensors in both manual and software control modes. The new protocol is faster, more robust and follows the indications of the transducer technical manuals to decode the data of interest.
3. The reading MATLAB user interface successfully incorporates data from the physical device and has a new, more attractive appearance for the user.
4. The theoretical simulation MATLAB user interface has been improved by the addition of new features that allow a great number of navigation options and provide a solid background of aortic modeling to students. The appearance has been modified to be similar to the layout of the hardware simulator interface.
5. The cardiovascular practice guide has been rewritten adapted to the new updates. It provides instructions of use of the tool and helps students to reinforce key concepts.

5.2. Future work

Although the tool is ready to be used there are features that could be improved to make a more precise control of parameters.

Regarding the software control mode of the physical simulator, it could be useful to increase the number of options the user has for the cardiac cycle parameters. Right now, the user can choose between three different options for the heart rate and duty cycle. By increasing the number of possibilities, the software mode would be more sensitive, useful and similar to the manual control mode.

Another possible implementation would be the substitution of the basic needle valve controlling the flow volume between the two columns of water by a flow proportional valve. These valves provide adjustment of flow volumes by different positioning of spools: an analogue input signal is transformed into a corresponding proportional opening section at the output. They use associated electronic controls to provide a precise positioning [43]. The design of the control circuit was begun during this project, comparing different model variables like maximum allowable pressure, output signal or hysteresis. However, its implementation would be complex (new power source, microcontroller and communication with the computer, among others, would be needed) and it was decided to set it aside. In the future, it would be interesting to do a viability analysis to check if there is a real benefit from the incorporation of this new system.

The simulation interface could include the 4-element series Windkessel so that the remaining configuration of this lumped model is included. It was not included because the 4-element parallel model was proved to have a poor performance at high frequencies [5]. However, it could be useful at low heart rates and in terms of interpretation for students. The mathematical development and its related MATLAB program have been already developed, thus the work would be to integrate it into the simulation interface tool.

REFERENCES

[1] D. Mozaffarian, et al., "Executive Summary: Heart Disease and Stroke Statistics—2015 Update", *Circulation-American Heart Association*, vol. 131, pp. 434-441, 2015.

[2] N. Townsend et al., "Cardiovascular disease in Europe: epidemiological update 2016", *European Heart J.*, vol. 37, pp. 3232-3245, 2016.

[3] CDC (2015). *Behaviors That Increase Risk for Heart Disease* [Online]. Available: https://www.cdc.gov/heartdisease/behavior.htm. [Accessed: 05- Mar- 2017].

[4] M. Pavlovic, *Bioengineering. A Conceptual Approach*, 1st ed. Cham, Switzerland: Springer, 2015.

[5] N. Peña, "Windkessel Modeling of the Human Arterial System", Bachelor Thesis, UC3M, Madrid, 2016.

[6] O. Frank, "Die Grundform des arteriellen Pulses," in *Zeitschrift für Biologie*, vol. 37, pp. 483–526, 1899.

[7] J. Hall and A. Guyton, *Textbook of medical physiology*, 11th ed. Philadelphia, Pa.: Saunders, 2005.

[8] G. Tortora and B. Derrickson, *Principles of anatomy & physiology*, 13th ed. Hoboken, N.J.: Wiley, 2013.

[9] M. Desco, Class Lecture, Topic: "Cardiovascular System I (anatomy)." Escuela Politécnica Superior, UC3M, Leganés, Mar. 2016.

[10] American Heart Association (2016). *About Arrhythmia* [Online]. Available: http://www.heart.org/HEARTORG/Conditions/Arrhythmia/AboutArrhythmia/About-Arrhythmia_UCM_002010_Article.jsp#.WZIeRVGrTIU. [Accessed: 17- Jun- 2017].

[11] J. J. Vaquero, Class Lecture, Topic: "Cardiovascular Applications." Escuela Politécnica Superior, UC3M, Leganés, Sep. 2016.

[12] World Health Organization (2016). *Global Health Estimates 2015: Deaths by Cause, Age, Sex, by Country and by Region, 2000-2015* [Online]. Available: http://www.who.int/healthinfo/global_burden_disease/estimates/en/index1.html [Accessed: 17- Jun- 2017].

[13] I. Kokalari, T. Karaja and M. Guerrisi, "Review on lumped parameter method for modeling the blood flow in systemic arteries," in *J. Biomedical Sci. and Eng.*, vol. 6, pp. 92-99, 2013.

[14] N. Westerhof, F. Bosman, C. J. de Vries and A. Noordergraaf, "Analog Studies of the Human Systemic Arterial Tree," in *J. Biomechanics*, vol. 2. pp. 121-143, 1969.

[15] N. Westerhof, N. Stergiopulos, M. Noble, "Distributed Models and Tube Models" in *Snapshots of Hemodynamics*, 2nd ed. New York: Springer, 2010.

[16] R. Nave (2016). *Ohm's Law* [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electric/ohmlaw.html. [Accessed: 28- Jun- 2017].

[17] N. Westerhof, J. W. Lankhaar and B.E. Westerhof, "The Arterial Windkessel,", in *Med. & Biol. Eng. & Comput.* J., vol. 47(2), pp. 131-41, 2009.

[18] N. Stergiopulos, B.E. Westerhof and N. Westerhof, "Total Arterial Inertance as the Fourth Element of the Windkessel Model," in *Amer. J. of Physiology— Heart and Circulatory Physiology*, vol. 276(1), pp. 81-88, 1999.

[19] Universal Medical (2017). *Vascular System Models* [Online]. Available: http://www.universalmedicalinc.com/all-products/education/anatomical-models/vascular-system-models.html. [Accessed: 05- Jul- 2017].

[20] Anatomy Warehouse (2017). *Shop for Anatomical Models and Body Figures* [Online]. Available: https://www.anatomywarehouse.com/anatomical-models. [Accessed: 05- Jul- 2017].

[21] 3B Scientific (2017). *Modelo de arteriosclerosis con sección transversa de la arteria, de 2 piezas* [Online]. Available: https://www.3bscientific.es/modelo-de-arteriosclerosis-con-seccion-transversa-de-la-arteria-de-2-piezas-g40-3b-scientific,p_33_304.html. [Accessed: 05- Jul- 2017].

[22] Johns Hopkins Medicine (2017). *Manikin-Based Simulations* [Online]. Available: http://www.hopkinsmedicine.org/simulation_center/training/manikin_based_simulations/index.html [Accessed: 05- Jul- 2017].

[23] Laerdal Medical (2017). *SimMan 3G* [Online]. Available: http://www.laerdal.com/us/products/simulation-training/emergency-care-trauma/simman-3g/ [Accessed: 05- Jul- 2017].

[24] A. Updegrove et al., "SimVascular: An Open Source Pipeline for Cardiovascular Simulation", *Ann. of Biomedical Eng.*, vol. 45, pp. 525-541, Mar. 2017.

[25] SimVascular Development Team (2017). *SimVascular* [Online]. Available: http://simvascular.github.io/ [Accessed: 05- Jul- 2017].

[26] IllinoisWiki (2017). *Simulation and Measurement of Human Physiology Courses* [Online]. Available: https://wiki.illinois.edu/wiki/display/SimMeasPhysio/Home [Accessed: 05- Jul- 2017].

[27] J. Martorell et al., Engineered arterial models to correlate blood flow to tissue biological response, *Ann. of the New York Academy of Sci.*, vol. 1254, pp. 51-56, Apr. 2012.

[28] Barry J. Doyle et al., An Experimental and Numerical Comparison of the Rupture Locations of an Abdominal Aortic Aneurysm, *J. Endovasc. Ther.*, vol. 16(3), pp. 322-335, Jun. 2009.

[29] M. Darowski et al., A new hybrid (hydro-numerical) model of the circulatory system, B. Polish Ac. Sci., vol. 61(4), pp. 993-1003, 2013.

[30] The MathWorks, Inc (2017). *MATLAB Product Description* [Online]. Available: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html [Accessed: 20- Jun- 2017].

[31] R. Johnson, *The Elements of MATLAB Style*, 1st ed. Cambridge, United Kingdom: Cambridge Univ. Press, 2010.

[32] MATLAB Style guidelines 2.0, 2nd ed., R. Johnson. 2014.

[33] B. Kernighan and R. Pike, *The Practice of Programming*, 1st ed. Addison–Wesley Publishing, 1999.

[34] SEDECAL (2017). *The company: About us* [Online]. Available: http://www.sedecal.com/sedecal.com/en/empresa/empresa.php [Accessed: 14- Jul- 2017].

[35] National Instruments Corporation (2017). *RS-232, RS-422, RS-485 Serial Communication General Concepts* [Online]. Available: http://www.ni.com/white-paper/11390/en/ [Accessed: 16- Jul- 2017].

[36] Medlab, "Invasive blood pressure OEM module Data Sheet", Jun. 2012.

[37] Em-tec (2017). [Online]. Available: https://www.em-tec.com/ [Accessed: 17- Jul- 2017].

[38] Em-tec, "DIGIFLOW – User Manual", 2010.

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

[40] Ministerio de la Presidencia (2017) *Documento BOE-A-2013-1337. Real Decreto 53/2013* [Online]. Available: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2013-1337 [Accessed: 23- Jul- 2017].

[41] European Commission (2017). *CE Marking* [Online]. Available: http://ec.europa.eu/growth/single-market/ce-marking/index_en.htm [Accessed: 23- Jul- 2017].

[42] Tu Europa (2017). *Marcado CE* [Online]. Available: http://europa.eu/youreurope/business/product/ce-mark/index_es.htm [Accessed: 24- Jul- 2017].

[43] Festo (2013). *Proportional Valves* [Online]. Available: https://www.festo.com/wiki/en/Proportional_valves [Accessed: 28- Jul- 2017].

ANNEXES

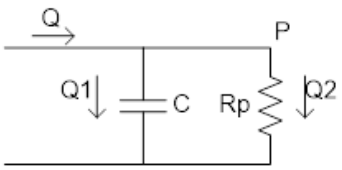
ANNEX A - WINDKESSEL MODEL. MATHEMATICAL DEMONSTRATION

	ELECTRICITY	PHYSIOLOGY
Kirchhoff's current law	$\sum_{k=1}^{n} I_k = 0$	$\sum_{k=1}^{n} Q_k = 0$
Kirchhoff's voltage law	$\sum_{k=1}^{n} V_k = 0$	$\sum_{k=1}^{n} P_k = 0$
Ohm's law	$I = \frac{V}{R}$	$Q = \frac{P}{R}$

Analogy of main electricity laws with physiological properties.

2-element Windkessel

First, solve the electrical circuit by using Kirchhoff's laws and Ohm's law as stated in Table above:



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

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

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

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

The equation may be transformed into Laplace domain, which can be used to perform the impedance analysis. Now, transforming every term to the discrete domain and solving for the pressure:

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

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

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

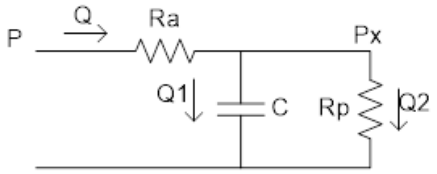
$$\frac{R_p \cdot Q(i) \cdot \Delta t}{C} = P(i) \cdot \left[R_p + \frac{\Delta t}{C}\right] - P(i-1) \cdot R_p$$

The final solution is:

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

3-element Windkessel

The circuit is solved in the same way as in the previous case:



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

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

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

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

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

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

The relationship between pressure and flow is found, and, again, Laplace definition can be useful for the impedance analysis.

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

Now, solving for the pressure in the discrete domain:

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

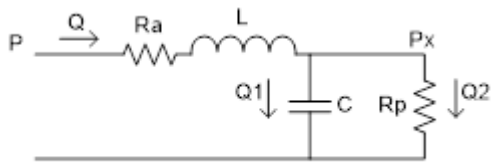
$$R_p\cdot\left\{Q(i)\cdot\left[1+\frac{R_a}{R_p}+\frac{C\cdot R_a}{\Delta t}\right] - Q(i-1)\cdot\frac{C\cdot R_a}{\Delta t}\right\} = P(i)\cdot\left(\frac{C\cdot R_p}{\Delta t}+1\right) - P(i-1)\cdot\frac{C\cdot R_p}{\Delta t}$$

Finally:

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

4-element Windkessel Series

The circuit is solved in the same way as in the previous cases:



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

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

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

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

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

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

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

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

Transforming it into the Laplace domain:

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

Transforming into a function of the previous samples:

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

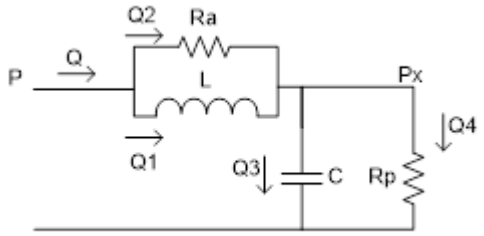
$$Q(i) \cdot \left[\left(1 + \frac{R_a}{R_p}\right) + \frac{\left(\frac{L}{R_p} + C \cdot R_a\right)}{\Delta t} + \frac{C \cdot L}{\Delta t^2}\right] - Q(i-1) \cdot \left[\frac{\left(\frac{L}{R_p} + C \cdot R_a\right)}{\Delta t} + \frac{2 \cdot C \cdot L}{\Delta t^2}\right] + Q(i-2) \cdot \frac{C \cdot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

Finally:

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

4-element Windkessel Parallel

In this case, the analysis is performed in the Laplace domain and later transformed into continuous temporal domain. This is due to the complexity of this model.



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

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

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

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

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

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

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

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

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

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

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

Now, transforming every term 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:

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

Practice: Cardiovascular Physiology

Equipment used

Simulation Interfaces

Two MATLAB GUI (Graphical User Interfaces):

- Theoretical simulation on Windkessel model.
- Interface displaying real time data acquired from the Windkessel hydraulic device.

Windkessel Hydraulic Device

Prototype manufactured by the company SEDECAL.

Practice Outline

Introduction

Physiologist Otto Frank proposed the Windkessel model to account for the blood pressure and flow waveforms by the simulation of arteries in terms of peripheral resistance and arterial compliance. Arterial network is modeled by equivalent electrical circuits in which resistor elements represent the dissipative nature of peripheral vessels and capacitors represent the compliance due to blood vessels elasticity. Later on, the incorporation of arterial resistance and inertance gave rise to more complex models based in the same electrical analogy. Observe in the figure below the different Windkessel configurations:

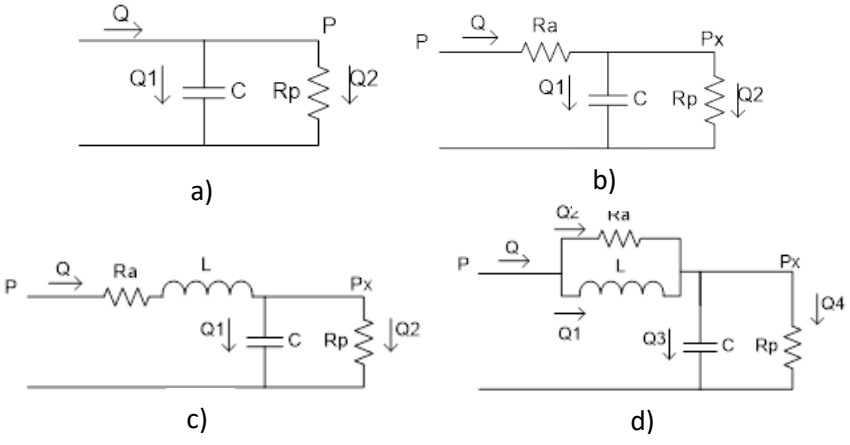


Figure 1. a) 2-element, b) 3-element, c) 4-element series and d) 4-element parallel Windkessel models.

These circuits can be solved by using the analogy of Ohm's law to find pressure-flow relationship.

$$\Delta P = F \cdot R$$

Now, there are more advanced mathematical approaches that describe the cardiovascular system in a more accurate way, but Windkessel is still very useful to understand the basic concepts that you need become acquainted with as Biomedical Engineering students.

In this practice you will learn on arterial system hemodynamics by using this model in a theoretical and practical way.

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:

Figure 2. Main window of theoretical Windkessel simulator.

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. You will see that the different heart cycle steps are pointed out. 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 that 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. Draw the aortic flow that patient will have. Which would be the systolic and diastolic pressure values for that patient? Based on what you observe in the 2-element Windkessel window, do you think they are physiologically normal? **Tip:** to observe drastic changes you may need to represent several heart cycles.

3: Go to the 4-element model, vary each parameter independently (peripheral resistance, arterial compliance, aortic resistance and inertance) 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/Vacuum controls (rotatory knobs)
3. Timing option switch (software versus manual)
4. Heart

5. Heart cycle controls (rotatory knobs)
6. Compliant element
7. Fluid reservoir
8. Peripheral resistance
9. Flow transducer location/Flow RS-232 port
10. Pressure transducer location/Pressure RS-232 port
11. Flow sensor

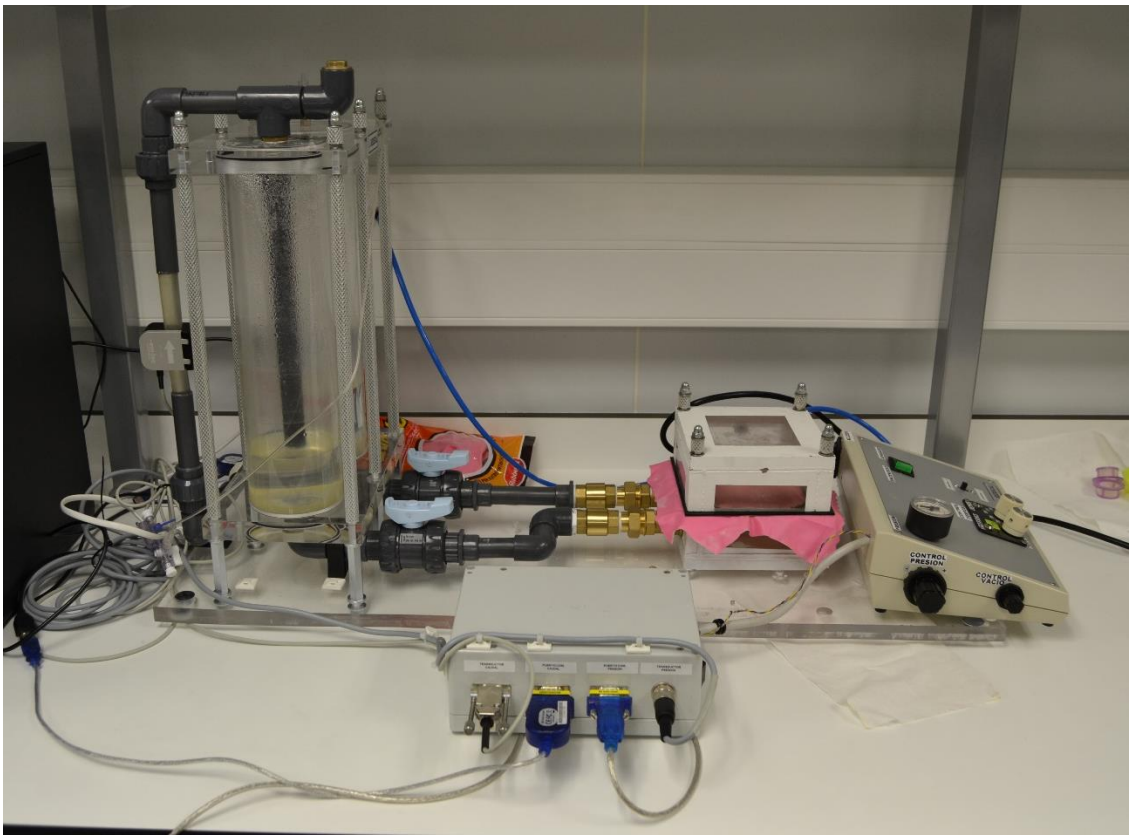


Figure 3. Physical Windkessel simulator.

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. You should see the following window:

Figure 4. Main window of physical Windkessel simulator interface.

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: First, let's read the pressure from the Manual Pulse Control. Change the number of samples to 2000 and press start 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 text files:

- **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. Which are the systolic and diastolic pressure values for each waveform?

Section 4

Finally, let's relate the data acquired from the theoretical simulator and the recorded data.

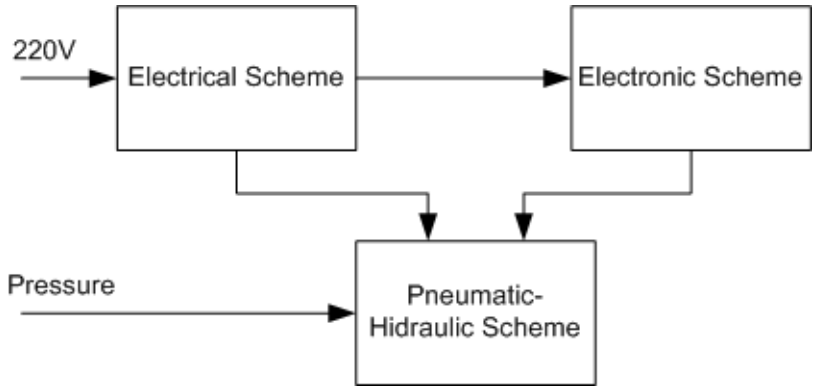
1: Compare the recorded pressure waveform and the pressures from Section 1:

- **a)** To which model (2, 3 or 4-element) do you think the obtained pressure is more similar to? Why?
- **b)** 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.

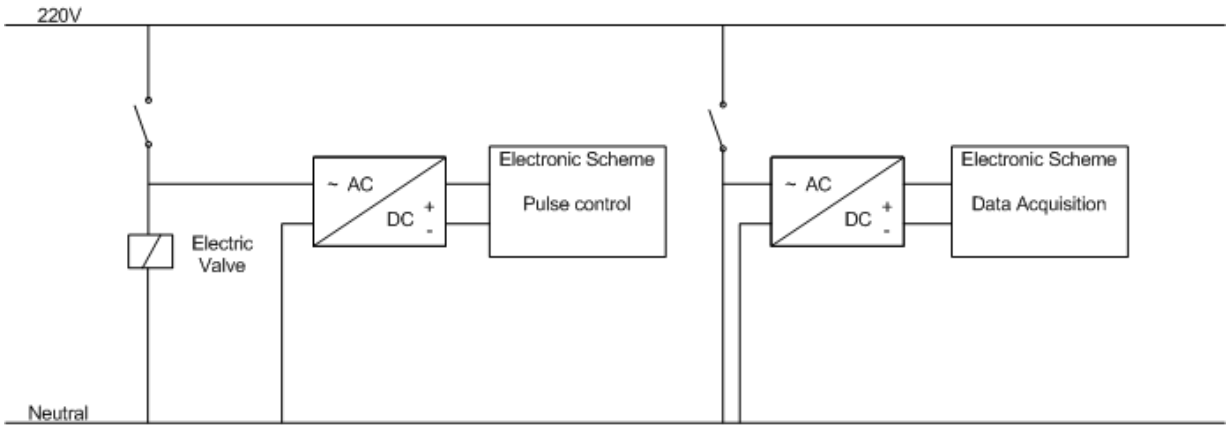
2: Recorded and theoretical pressure waveforms are very similar not only in their shape, but also in their range of values; but the two flow waveforms fluctuate between different values. Can you guess why?

ANNEX C – PHYSICAL DEVICE DIAGRAMS, SCHEMES AND PICTURES

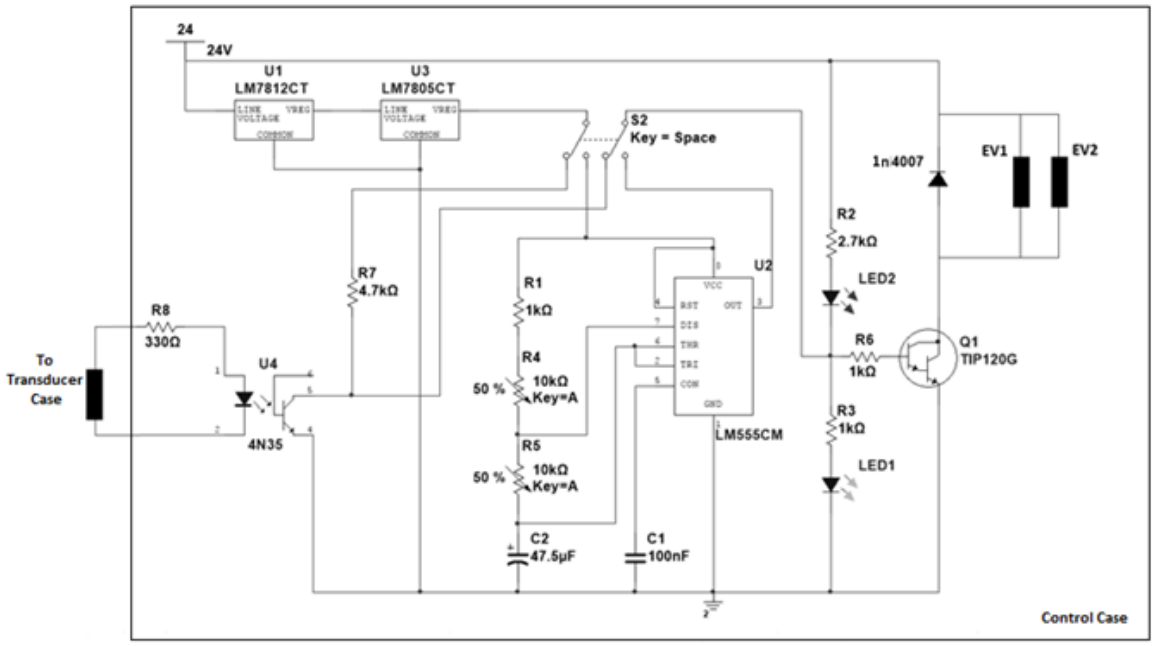
Pneumatic-hydraulic, electronic and electric levels of the device and their relation:



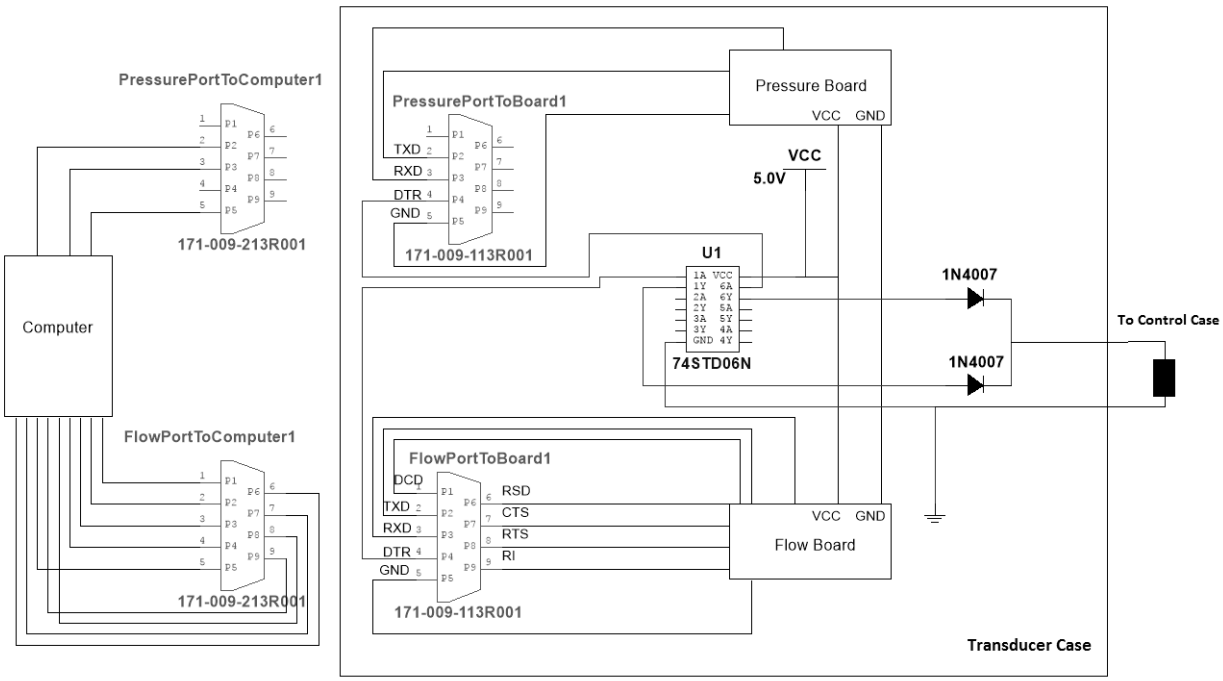
Electrical scheme:



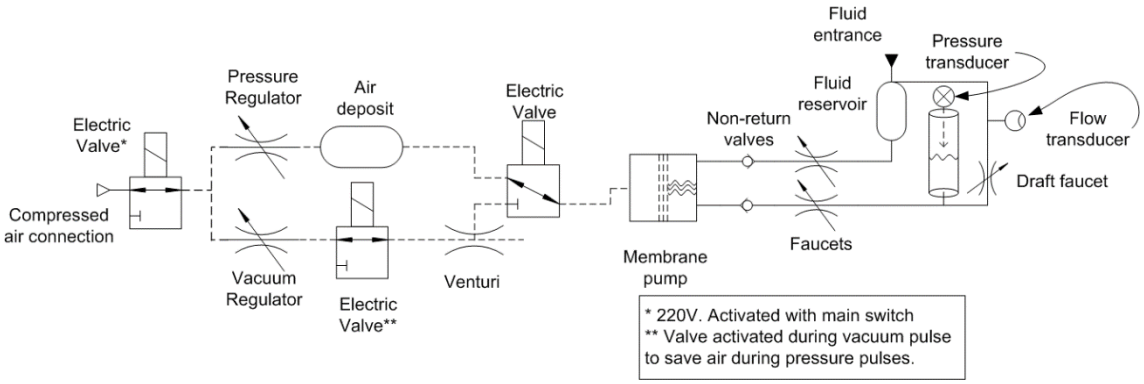
Electronic scheme control case:



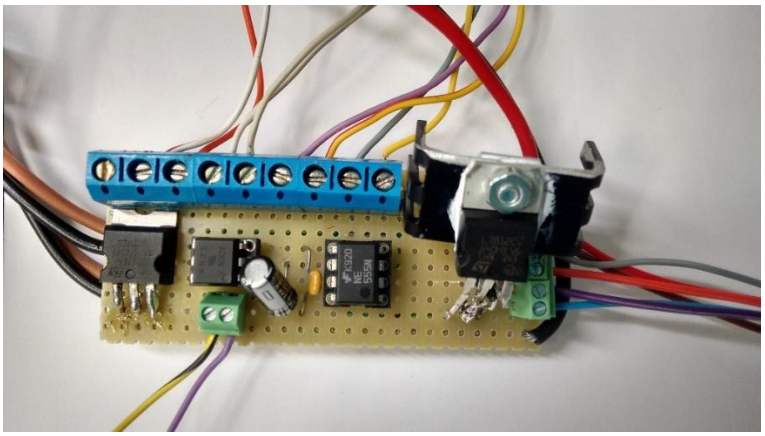
Electronic scheme transducer case:



Pneumatic-hydraulic scheme:

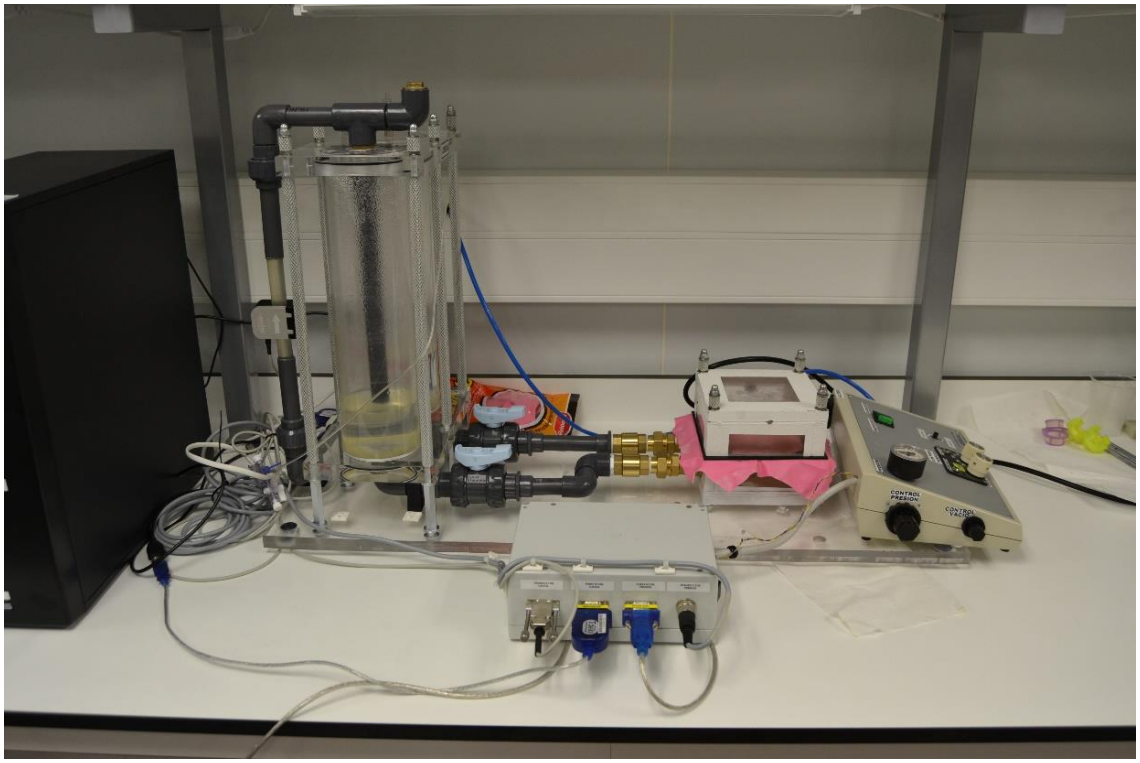


Electronic board inside transducer case:



Outside control case:

Full physical system:



ANNEX D – MATLAB CODES

SIMULATION INTERFACE:

This theoretical platform is constituted by 11 MATLAB functions. Representative parts have been selected and are shown in the following pages.

1. Compute 2-element Windkessel model:

The whole function is included below.

```
function [pressureMedUnits,flowMedUnits] = compute2WK(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
timeCardiacCycle = 60/handles.data(1);
timeDiastole = timeCardiacCycle-handles.data(2);

%%
% Define flow function
flow =
zeros(handles.data(4)*(int16(round(timeCardiacCycle,3)/0.001)),1);
% For 1 to the desired number of periods:
aux = 1;
for i = 1:handles.data(4)
    for t = 0.001:0.001:round(timeCardiacCycle,3)
        % 0.001 is the time interval between consecutive samples
        if (t <= handles.data(2))
            %When t<systole duration
            flow(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
            flow(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
flowMedUnits = flow*10^6;

%%
% Define Pressure function for a 2 element WK
pressure =
zeros(handles.data(4)*(int16(round(timeCardiacCycle,3)/0.001)),1);
pressure(1) = 10000;
% It will have the same number of elements as the flow
for i = 2:1:length(flow)
    % The following equation is obtained from the numerical resolution
to
    % the equation modeling the two-element-configuration circuit.
    pressure(i) =
(((handles.data(5)*flow(i)*0.001)/handles.data(6))+handles.data(5)*pre
ssure(i-1))/(handles.data(5)+0.001/handles.data(6));
end
% Again, transform to medical units for display purposes.
pressureMedUnits = pressure*760/101300;
end
```

2. Compute 2-element Windkessel model for default values:

This function is equal to the previous one, but it is only executed for the original data (default) to draw the reference line in the 2-element WK model.

```
function [pressureMedUnits,flowMedUnits] = compute2WKoriginal(handles)
%%
% Compute_2WK
% This function performs the calculations required for plotting the
% pressure, flow and pressure volume loops for the default data.
%%
% Defining some variables from the global ones
timeCardiacCycle=60/handles.dataOriginal(1);
timeDiastole=timeCardiacCycle-handles.dataOriginal(2);

%%
% Define flow function
flow =
zeros(handles.dataOriginal(4)*(int16(round(timeCardiacCycle,3)/0.001))
,1);
% For 1 to the desired number of periods:
aux = 1;
for i = 1:handles.dataOriginal(4)
    for t = 0.001:0.001:round(timeCardiacCycle,3)
        % 0.001 is the time interval between consecutive samples
        if (t <= handles.dataOriginal(2))
            %When t<systole duration
            flow(aux) =
handles.dataOriginal(3)*(sin((pi*t)/handles.dataOriginal(2)))^2;
            % Flow is defined as a sine squared function during
systole
        end
```

```
        if (t > handles.dataOriginal(2))
            % When t>systole duration
            flow(aux) = 0;
            % And as zero during diastole
        end
        aux = aux+1;
    end
end
% As the global variables are defined in SI units, convert them to
medical
% units for display purposes
flowMedUnits = flow*10^6;

%%
% Define Pressure function for a 2 element WK
pressure =
zeros(handles.dataOriginal(4)*(int16(round(timeCardiacCycle,3)/0.001))
,1);
pressure(1) = 10000;
% It will have the same number of elements as the flow
for i = 2:1:length(flow)
    % The following equation is obtained from the numerical resolution
to
    % the equation modeling the two-element-configuration circuit.
    pressure(i) =
(((handles.dataOriginal(5)*flow(i)*0.001)/handles.dataOriginal(6))+han
dles.dataOriginal(5)*pressure(i-
1))/(handles.dataOriginal(5)+0.001/handles.dataOriginal(6));
end
% Again, transform to medical units for display purposes.
pressureMedUnits = pressure*760/101300;
end
```

3. Compute 3-element Windkessel model:

It has the same inputs and outputs of functions 1 and 2. The structure is the same, but the calculations are performed for the 3-element model.

```
function [pressure2MedUnits,flowMedUnits] = compute3WK(handles)
```

4. Compute 3-element Windkessel model for default values:

It has the same inputs and outputs of functions 1, 2, and 3. The structure is the same, but the calculations are performed for the 3-element model for default data.

```
function [pressure2MedUnits,flowMedUnits] =
compute3WKoriginal(handles)
```

5. Compute 4-element Windkessel parallel model:

It has the same inputs and outputs of functions 1, 2, 3 and 4. The structure is the same, but the calculations are performed for the 4 parallel elements model.

```
function [pressure4MedUnits,flowMedUnits] =
compute4WKparallel(handles)
```

6. Compute 4-element Windkessel parallel model for default values:

It has the same inputs and outputs of functions 1, 2, 3, 4 and 5. The structure is the same, but the calculations are performed for the 4-element parallel model for default data.

```
function [pressure4MedUnits,flowMedUnits] =
compute4WKoriginalparallel(handles)
```

7. Save figures displayed in model panels:

The input is the figure displayed at that moment. It can be pressure, flow, pressure-flow loop or pressure and flow together, with the parameters specified in the panel in the moment of saving the figure.

There is not output; a .fig file is generated when executing it.

```
function saveFigure(fig_store)
```

8. Development of panel simulating the 2-element Windkessel model:

Due to the excessive length of the function only a small part of it is included below. The output is the execution of the interface itself.

```
function WK_2(WK2_PANEL,handles,ax)
%%
% WK_2
% This function executes the window for specifically simulating the
% 2-element Windkessel
% INPUTS:
%   WK2_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];
%   ax: contains axes properties defined in WK_Main_Interface_Reading
%       handles.data=handles.dataOriginal;
%       handles.additional=[Sec SItoFlow FlowtoSI SItoHRU HRUtoSI
SItoHCU HCUtoSI SItoHLU HLUtoSI];

%%
%DEFINING ALL THE ELEMENTS FOR THE PANEL

%Defining titles:
titleParameters =
uicontrol('Parent',WK2_PANEL,'Style','text','FontName','Calibri',...
    'Units','normalized','FontWeight','Bold','Position',[0.07 0.87
0.15 0.07],...
```

```
    'FontSize',12,'BackgroundColor',[1 1 1],'String','Parameter to
modify:');
SIunits =
uicontrol('Parent',WK2_PANEL,'Style','text','FontName','Calibri',...
    'Units','normalized','FontWeight','Bold','Position',[0.29 0.85
0.15 0.07],...
    'FontSize',12,'BackgroundColor',[1 1 1],'String','SI Units: ');
MEDunits =
uicontrol('Parent',WK2_PANEL,'Style','text','FontName','Calibri','Font
Weight','Bold',...
    'Units','normalized','Position',[0.59 0.85 0.15
0.07],'FontSize',12,...
    'BackgroundColor',[1 1 1],'String','Medical Units: ');

%Defining Reset Button
reset_button = uicontrol('Parent',WK2_PANEL,'Style', 'pushbutton',
'String', 'RESET','FontWeight','Bold',...
    'BackgroundColor',[1 1 1],'Units','normalized','Position', [0.6
0.05 0.16 0.08],...
    'FontSize',12,'ForegroundColor',[0 0
1],'Callback',@RESET_btn_Callback );
reset_text =
uicontrol('Parent',WK2_PANEL,'Style','text','FontName','Calibri','Font
Weight','Bold',...
    'Units','normalized','Position',[0.59 0.13 0.18
0.07],'FontSize',12,...
    'BackgroundColor',[1 1 1],'String','Return to default values: ');

%Defining Save Button
save_button = uicontrol('Parent',WK2_PANEL,'Style', 'pushbutton',
'String', 'SAVE','FontWeight','Bold',...
    'BackgroundColor',[1 1 1],'Units','normalized','Position', [0.79
0.05 0.16 0.08],...
    'FontSize',12,'ForegroundColor',[0 0
1],'Callback',@SAVE_btn_Callback );
save_text =
uicontrol('Parent',WK2_PANEL,'Style','text','FontName','Calibri','Font
Weight','Bold',...
    'Units','normalized','Position',[0.78 0.12 0.19
0.08],'FontSize',12,...
    'BackgroundColor',[1 1 1],'String','Save current figure: ');

%Defining Selection group buttons
bg = uibuttongroup('Parent',WK2_PANEL, 'Title','Select a variable to
display: ','FontWeight',...
    'Bold','Visible','on','FontSize',11,
'Units','normalized','Position', ...
    [0.6 0.22 0.35 0.3], 'BackgroundColor',[1 1
1],'SelectionChangedFcn',@b_selection);

%Create four radio buttons in the selection group
r1 = uicontrol(bg,'Style','radiobutton', 'String','Aortic
pressure','FontSize',12,...
    'Position',[10 115 150 40],'BackgroundColor',[1 1 1]);
r2 = uicontrol(bg,'Style','radiobutton', 'String','Aortic
flow','FontSize',12,...
    'Position',[10 80 150 40],'BackgroundColor',[1 1 1]);
r3 = uicontrol(bg,'Style','radiobutton', 'String','Aortic PQ-
Loop','FontSize',12,...
    'Position',[10 45 150 40],'BackgroundColor',[1 1 1]);
```

```
r4 = uicontrol(bg,'Style','radiobutton', 'String','Pressure and
flow','FontSize',12,...
    'Position',[10 10 150 40],'BackgroundColor',[1 1 1]);

%Defining all buttons
HR_title =
uicontrol('Parent',WK2_PANEL,'Style','text','Units','normalized',...
    'Position',[0.07 0.42 0.15
0.07],'FontSize',11,'BackgroundColor',[1 1 1],'String','Heart Rate:');

HR_edit =
uicontrol('Parent',WK2_PANEL,'Style','Edit','Units','normalized',...
    'Position',[0.3 0.45 0.125
0.055],'String',handles.data(1),'FontSize',11,'Callback',
@HR_edit_Callback);
%Units
D='$$\frac{beats}{min}$$';
h=annotation(WK2_PANEL,'textbox',[0,0,1,1],'string',D,'interpreter','l
atex',...
    'FitBoxToText','on','Position',[0.43 0.43 0.15
0.07],'LineStyle','none','FontSize',11);

HR_slider = uicontrol('Parent',WK2_PANEL,'Style',
'slider','Units','normalized','SliderStep',[0.1 0.1],...
    'Min',60,'Max',140,'Value',handles.additional(1),'Position', [0.3
0.43 0.125 0.023],'Callback', @HR_slider_Callback);

T_title =
uicontrol('Parent',WK2_PANEL,'Style','text','Units','normalized',...
    'Position',[0.07 0.317 0.15
0.07],'FontSize',11,'BackgroundColor',[1 1 1],'String','Heart
Period:');

T_text =
uicontrol('Parent',WK2_PANEL,'Style','Text','BackgroundColor',[1 1
1],'Units','normalized',...
    'Position',[0.3 0.335 0.12
0.05],'String',handles.additional(1)/handles.data(1),...
    'FontSize',11, 'Callback', @T_text_Callback);

(. . .)

%Callback functions definitions
    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)));

        handles.dataOriginal(1)=handles.data(1);
```

```
handles.dataOriginal(2)=(2/5)*(handles.additional(1)/handles.dataOrigi
nal(1));
        handles.dataOriginal(4)=handles.data(4);
        [Pdefault,Qdefault]=compute2WKoriginal (handles);

        %Compute and plot
        if (strcmp(bg.SelectedObject.String,'Aortic pressure') == 1)
            cla reset
            [P,Q]=compute2WK(handles);
            handles.current_data=P;
            %Plot pressures
            plot(Pdefault,'--');
            hold on
            plot(handles.current_data);
            end_systole=zeros(1,handles.data(4));
            end_systole(1)=0;
            end_diastole=zeros(1,handles.data(4));
            end_diastole(1)=0;
            for i=0:handles.data(4)-1

end_systole(i+2)=round(1000*handles.data(2)+i*60*1000/handles.data(1))
;
                line([end_systole(i+2) end_systole(i+2)],[min(P)-
10,max(P)+10],'LineWidth',0.3,...
                    'Color','red','LineStyle',':');

end_diastole(i+2)=round((i+1)*60*1000/handles.data(1));
                line([end_diastole(i+2) end_diastole(i+2)],[min(P)-
10,max(P)+10],'LineWidth',0.3,...
                    'Color','red','LineStyle',':');
                text((end_systole(i+2)+end_diastole(i+1))/2,min(P)-
10,'Systole','HorizontalAlignment', 'center');
                text((end_systole(i+2)+end_diastole(i+2))/2,min(P)-
10,'Diastole','HorizontalAlignment', 'center');
            end
            ylabel('Pressure (P)[mmHg]')
            xlabel('Time(t)[ms]')
            title('Pressure in the Aorta, 2WK')
            legend('Default data','Current data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Aortic flow') == 1)
            cla reset
            [P,Q]=compute2WK(handles);
            handles.current_data=Q;
            %Plot flow
            plot(Qdefault,'--');
            hold on
            plot(handles.current_data);
            end_systole=zeros(1,handles.data(4));
            end_systole(1)=0;
            end_diastole=zeros(1,handles.data(4));
            end_diastole(1)=0;
            for i=0:handles.data(4)-1

end_systole(i+2)=round(1000*handles.data(2)+i*60*1000/handles.data(1))
;
                line([end_systole(i+2) end_systole(i+2)],[min(Q)-
10,max(Q)+10],'LineWidth',0.3,...
                    'Color','red','LineStyle',':');
```

```
end_diastole(i+2)=round((i+1)*60*1000/handles.data(1));
                line([end_diastole(i+2) end_diastole(i+2)],[min(Q)-
10,max(Q)+10],'LineWidth',0.3,...
                     'Color','red','LineStyle',':');
                text((end_systole(i+2)+end_diastole(i+1))/2,min(Q)-
20,'Systole','HorizontalAlignment', 'center');
                text((end_systole(i+2)+end_diastole(i+2))/2,min(Q)-
20,'Diastole','HorizontalAlignment', 'center');
            end
            ylabel('Flow(Q)[ml/s]')
            xlabel('Time(t)[ms]')
            title('Flow in the Aorta, 2WK')
            legend('Default data','Current data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Aortic PQ-Loop') ==
1)
            cla reset
            [P,Q]=compute2WK(handles);
            %Plot P-Q loop

plot(baseFlow{1,handles.data(4)},basePressure{1,handles.data(4)},'--')
            hold on
            plot(Q,P)
            xlabel('Flow(Q)[ml/s]')
            ylabel('Pressure (P)[mmHg]')
            title('Aortic PQ-Loop, 2WK')
            legend('Default data','Current data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Pressure and
flow')==1)
            cla reset
            legend('hide')
            [P,Q]=compute2WK(handles);
            yyaxis left
            plot(P);
            ylabel('Pressure (P)[mmHg]')
            yyaxis right
            plot(Q);
            ylabel('Flow(Q)[ml/s]');
            xlabel('Time(t)[ms]');
            end_systole=zeros(1,handles.data(4));
            end_systole(1)=0;
            end_diastole=zeros(1,handles.data(4));
            end_diastole(1)=0;
            for i=0:handles.data(4)-1

end_systole(i+2)=round(1000*handles.data(2)+i*60*1000/handles.data(1))
;
                line([end_systole(i+2)
end_systole(i+2)],[min(Q),max(Q)],'LineWidth',0.3,...
                     'Color','red','LineStyle',':');

end_diastole(i+2)=round((i+1)*60*1000/handles.data(1));
                line([end_diastole(i+2)
end_diastole(i+2)],[min(Q),max(Q)],'LineWidth',0.3,...
                     'Color','red','LineStyle',':');

text((end_systole(i+2)+end_diastole(i+1))/2,min(Q)+15,'Systole','Horiz
ontalAlignment', 'center');
```

```
text((end_systole(i+2)+end_diastole(i+2))/2,min(Q)+15,'Diastole','Hori
zontalAlignment', 'center');
            end
            title('Comparison aortic flow and pressure, 2WK')
        end
        ax.Title.Color='white';
        ax.XAxis.Color='white';
        hold off
    end

(. . .)


    function RESET_btn_Callback(source,callbackdata)
        %Reset all parameters to original values
        handles.dataOriginal(1)=72;
        handles.dataOriginal(2)=(2/5)*(60/handles.dataOriginal(1));
        handles.dataOriginal(4)=1;
        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(nCycles_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(nCycles_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 (strcmp(bg.SelectedObject.String,'Aortic pressure') == 1)
            cla reset
            [P,Q]=compute2WK(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')
            legend('Default data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Aortic flow') == 1)
            cla reset
            [P,Q]=compute2WK(handles);
            handles.current_data=Q;
            %Plot flow
            plot(handles.current_data,'--');
            ylabel('Flow(Q)[ml/s]')
            xlabel('Time(t)[ms]')
            title('Pressure in the Aorta, 2WK')
            legend('Reference data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Aortic PQ-Loop') ==
1)
            cla reset
            [P,Q]=compute2WK(handles);
            plot(Q,P,'--')
            xlabel('Flow(Q)[ml/s]')
            ylabel('Pressure (P)[mmHg]')
            title('Aortic PV-Loop, 2WK')
            legend('Default data')
            ax.YAxis.Color='white';
        elseif (strcmp(bg.SelectedObject.String,'Pressure and
flow')==1)
            cla reset
            legend('hide')
            [P,Q]=compute2WK(handles);
            yyaxis left
            plot(P);
            ylabel('Pressure (P)[mmHg]')
            yyaxis right
            plot(Q);
            ylabel('Flow(Q)[ml/s]');
            xlabel('Time(t)[ms]');
            title('Comparison aortic flow and pressure, 2WK')
        end
        ax.Title.Color='white';
        ax.XAxis.Color='white';
        hold off
    end


pause(0.35)
%THIS WILL ONLY EXECUTE ONCE
[P,Q]=compute2WK(handles);
handles.current_data=P;
cla reset
%Plot pressures
plot(handles.current_data,'--');
ylabel('Pressure[mmHg]')
xlabel('Time(t)[ms]')
title('Pressure in the Aorta, 2WK')
legend('Default data')
ax.Title.Color='white';
ax.XAxis.Color='white';
ax.YAxis.Color='white';
```

```
(. . .)

end
```

9. Development of panel simulating the 3-element Windkessel model:

It has the same inputs and outputs of function 8. The structure is the same, but adapted to the 3-element WK model.

```
function WK_3(WK3_PANEL,handles,ax)
```

10. Development of panel simulating the 4-element Windkessel parallel model:

It has the same inputs and outputs of functions 8 and 9. The structure is the same, but adapted to the 4-element WK parallel model.

```
function WK_4(WK4_PANEL,handles,ax)
```

11. Main window and definition of global variables:

The inputs of the other functions are defined here. When it is executed the main window appears.

```
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
heartRate = 72;
timeSystole = (2/5)*(60/heartRate);
maxFlow = 5e-4; %[m^3/s]=4.98 l/min
nCycles = 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=[heartRate timeSystole maxFlow nCycles 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

%Set a new color order:
co = [0.1451    0.6549    1.0000
    0.8500    0.3250    0.0980];
set(groot,'defaultAxesColorOrder',co)

%%
%Create Window Components:
%CREATE MAIN FIGURE
f =
figure('Visible','off','Toolbar','none','Menubar','none','Color',[0 0
0.4],'Name','Windkessel Interface', 'NumberTitle', 'off');

%CREATE WELCOME TEXT
txtWelcome =
uicontrol('Style','text','Units','normalized','FontWeight','Bold',...
    'Position',[0.25 0.75 0.5
0.1],'FontSize',20,'ForegroundColor','white',...
    'String','WELCOME!', 'FontName','Century','BackgroundColor',[0 0
0.4]);

%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',12,'Position', [0.05 0.85 0.2 0.1],...
    'Callback', @Menu);
```

```
(. . .)

%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',[0.02 0.04 0.5 0.85]);
WK3_PANEL = uipanel('FontSize',14,'FontWeight','Bold',...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.02 0.04 0.5 0.85]);
WK4_PANEL = uipanel('FontSize',14,'FontWeight','Bold',...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.02 0.04 0.5 0.85]);

%DEFINE THE AXES WHERE THE SIMULATED VARIABLES WILL BE PLOTTED
axesLogo = axes('Visible','on','Units','normalized','Position',[0.82
0.88 0.15 0.1]);
imshow('logo_azul.png');

ax = axes('Visible','off','Units','normalized','Position',[0.56 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,callback)
        nMenu = source.Value;
        thisName = source.String;
        nameMenu = thisName{nMenu};
        %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';
            legend('hide')
            %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,ax)
            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,ax)
            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,ax)
            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 buttons
%available at the main panel. The difference is that these buttons are
only
%available at the main window, while the pop-up menu is visible and
%accessible at any window.

%2 element 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,ax)
        WK2_PANEL.Title = 'Modify the desired 2 WK Parameters: ';
        WK2_PANEL.Visible = 'on';
        ax.Visible = 'on';
    end
```

```
%3 element 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,ax)
        WK3_PANEL.Title = 'Modify the desired 3 WK Parameters: ';
        WK3_PANEL.Visible = 'on';
        ax.Visible = 'on';

    end
%4 element 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,ax)
        WK4_PANEL.Title = 'Modify the desired 4 WK Parameters: ';
        WK4_PANEL.Visible = 'on';
        ax.Visible = 'on';
    end
end
```

READING INTERFACE:

The interface acquiring data from the physical device is constituted by 9 MATLAB functions. Representative parts have been selected and are shown in the following pages.

1. Read flow in manual control mode:

```
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
delete(instrfindall);
s = serial('COM4','BaudRate',38400,'DataBits',8,'StopBits',1);
s.terminator = 'LF'; % Specify terminator character: line feed
s.BytesAvailableFcnMode = 'terminator'; %a bytes-available event
occurs when
% the terminator specified by the Terminator property is read
s.BytesAvailableFcn = @mycallback_read_data; %The bytes-available
event executes
% the callback function specified for the BytesAvailableFcn property.
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 TIMER:
%TIMER FOR DECODING FLOW DATA
paint_timer = timer('TimerFcn',@mycallback_paint_timer
,'BusyMode','drop',...
    'StartDelay',1,'Period',0.01,'ExecutionMode','fixedRate');

%START THE TIMER
start(paint_timer)

%INITIALIZE VARIABLES
rawflow_vector = [];
cont_read1 = 0;
ind = 1;
acq = [];

%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 timer, close serial port
        stop(paint_timer)
        delete(paint_timer)
        fclose(s)
        delete(s)
        break
    end
end

%%
%DEFINE CALLBACK FUNCTIONS

%Function reading data
    function mycallback_read_data (obj, event)
        if (s.BytesAvailable ~= 0)
            cont_read1=cont_read1+1;
            acq(cont_read1,:) = fscanf(s,'%f');
        end
    end

%DECODING SECTION OF THE CODE
    function mycallback_paint_timer (obj, event)
        %Obtain meaningful value and store it in a vector
        for i = ind:size(acq,1)
            rawflow_vector(i) = acq(i,4)/60;
        end
```

```
        ind = size(acq,1);

        %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);
    end

end
```

2. Read flow in software control mode:

It has the same inputs and outputs of function 1. The structure is the same, but adapted to the software control mode.

```
function Read_Trial_Flow_AND_Pulses(handles,panel)
```

3. Read pressure in manual control mode:

It has the same inputs and outputs of functions 1 and 2. The structure is the same, but adapted to the pressure data acquisition.

```
function Read_Trial_Pressure(handles)
```

4. Read pressure in software control mode:

```
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 AREAS THAT WILL BE HIGHLIGHTED WHEN 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
delete(instrfindall);
v = serial('COM4','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);
u.BytesAvailableFcnMode = 'byte';
u.BytesAvailableFcn = @mycallback_read_data;
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 DECODING DATA
paint_timer = timer('TimerFcn',@mycallback_paint_timer
,'BusyMode','drop',...
    'StartDelay',1,'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(paint_timer)


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


%MAIN LOOOP:it will constantly execute until stop condition
while(1)
    % 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
    refreshdata

    %STOP CONDITION: if enough samples have been read, stop
    if (length(real_pressure) > =handles.data(1))
        stop(t)
```

```
        stop(d)
        stop(paint_timer)
        delete(t)
        delete(d)
        delete(paint_timer)

        pause(1)
        fclose(v)
        delete(v)
        fclose(u)
        delete(u)
        break
    end
end

%%
%DEFINE CALLBACK FUNCTIONS

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

%Function decoding data
    function mycallback_paint_timer (obj, event)
        for i = ind:length(rawpressure)-1
            sync = dec2bin(rawpressure(i),8);
            if isequal(sync(1:4),'1100') == 1
                wave = dec2bin(rawpressure(i+1),8);
                if strcmp(wave(1),'0') == 1
                    real_pressure(cont) =
(bin2dec(wave)+(str2double(sync(6))*2^7)+(str2double(sync(5))*2^8)-
100)*1.6; %-100: calibration to receive negative values
                    real_pressure(cont) = real_pressure(cont)+70;
%Calibrate to aortic values
                    cont = cont+1;
                end
            end
        end

        ind = length(rawpressure);

        %Save data in a file
        fileID2 = fopen('PressureData.txt','w');
        fprintf(fileID2,'%6s\n','Real Time Pressure');
        fprintf(fileID2,' %d \n',real_pressure);
        fclose(fileID2);
    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 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
```

5. Creation of window for manual flow reading:

Its inputs are the panel corresponding to manual flow recording, the global structure with some common data and the axes for plotting the data.

```
function Read_manual_flow (manual_flow_PANEL,handles,ax)
```

6. Creation of window for software flow reading:

```
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_button = uicontrol('Parent',software_flow_PANEL,'Style',
'pushbutton', 'String', 'START','FontWeight','Bold',...
    'BackgroundColor',[1 1 1],'Units','normalized','Position', [0.4
0.25 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 0.47 0.9
0.1],'FontSize',12,...
    '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 0.40 0.2
0.1],'String',handles.data(1),'Callback', @x_max_edit_Callback);

%CREATE TEXT INSTRUCTIONS
selectHeartRate =
uicontrol('Parent',software_flow_PANEL,'Style','text','ForegroundColor
',[0 0 1],...
    'Units','normalized','Position',[0.05 0.82 0.9
0.1],'FontSize',12,...
    '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 0.60 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)
        ax.XLim = [0 handles.data(1)];
        %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
```

7. Creation of window for manual pressure reading:

Its inputs are the same of functions 5 and 6. The structure is the same but adapted to manual pressure reading.

```
function Read_manual_pressure (manual_pressure_PANEL,handles,ax)
```

8. Creation of window for software pressure reading:

Its inputs are the same of functions 5, 6 and 7. The structure is the same but adapted to software pressure reading.

```
function Read_software_pressure (software_pressure_PANEL,handles,ax)
```

9. Main window and definition of global variables:

```
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

heartRate = 60; % Heart rate [beats/min]
timeCardiacCycle = 60/heartRate; % Heart period [s]
duty = 2/5; % Fraction of period occupied by systole
delay = duty*timeCardiacCycle; % Time occupied by systole [s]

handles.vector = [74 74 74 75 75 75 75 (...)];
% Define Global Variables, handles will be an input to all the other
% functions used
handles.data = [x_max y_min y_max heartRate timeCardiacCycle 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 0
0.4],'Name','Reading Interface', 'NumberTitle', 'off');

%CREATE WELCOME TEXT
txtWelcome =
uicontrol('Style','text','Units','normalized','ForegroundColor','white
',...
    'Position',[0.25 0.75 0.5
0.1],'FontSize',20,'FontWeight','Bold',...
    'String','WELCOME!', 'FontName','Century','BackgroundColor',[0 0
0.4]);

%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',12,'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',18,'ForegroundColor','white',...
    'BackgroundColor',[0 0
0.4],'FontName','Calibri','FontWeight','Bold',... %white
    'Position',[0.1 0.1 0.35 0.6]);
```

```
hp_software = uipanel('Title','Software Pulse Control:
','FontSize',18,'ForegroundColor','white',...
    'BackgroundColor',[0 0
0.4],'FontName','Calibri','FontWeight','Bold',...
    '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',14,'FontWeight','Bold',...
    'FontName','Calibri','BackgroundColor',[0.55 0.75
0.75],'Callback', @manual_pressure);

(. . .)

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

%CREATE THE PANELS FOR READING FUNCTIONS: only visible in their
respective
%windows
manual_pressure_PANEL = uipanel('FontSize',15,...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.10 0.20 0.33 0.50]);
manual_flow_PANEL = uipanel('FontSize',15,...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.10 0.20 0.33 0.50]);
software_pressure_PANEL = uipanel('FontSize',15,...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.04 0.05 0.45 0.8]);
software_flow_PANEL = uipanel('FontSize',15,...
    'BackgroundColor','white','FontName','Calibri',...
    'Visible','off','Position',[0.04 0.05 0.45 0.8]);

%CREATE THE AXES FOR DISPLAYING THE DATA
axesLogo = axes('Visible','on','Units','normalized','Position',[0.82
0.88 0.15 0.1]);
imshow('logo_azul.png');

ax = axes('Visible','off','Units','normalized','Position',[0.55 0.15
0.4 0.6],'XLim',[0 x_max],'YLim',[y_min y_max]);
ax.XAxis.Color = 'white';
ax.YAxis.Color = 'white';
ax.Title.Color = 'white';



%%
%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)
        nMenu = source.Value;
        thisName = source.String;
        nameMenu = thisName{nMenu};
        %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

        (. . .)

        end
    end
%A similar process is developed if the user pushes one of the buttons
%available at the main panel. The difference is that these 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
```