Filipe Bernardo

Licenciado em Ciências de Engenharia Física

Desenvolvimento duma Unidade de Injeção Vascular Programável

Dissertação para obtenção do Grau de Mestre em

Engenharia Física

Orientador: Paulo A. Ribeiro, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientador: João Goyri O'Neill, Professor Catedrático, Faculdade de Ciências Médicas da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2019

Desenvolvimento duma Unidade de Injeção Vascular Programável

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "unlthesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/unlthesis [2] http://www.di.fct.unl.pt

Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu orientador Professor Doutor Paulo Ribeiro pela sua enorme disponibilidade e excelente acompanhamento durante a elaboração desta dissertação.

Agradeço às oficinas do Departamento de Física e aos seus técnicos pelo tempo disponibilizado e pelo conhecimento transmitido, tanto em técnicas como em métodos de operacionamento da maquinaria disponível nas oficinas.

Agradeço o suporte financeiro da parte da FEDER, através do Programa Operacional Factores de Competitividade - COMPETE e à Fundação para a Ciência e a Tecnologia - FCT, pelo projecto UID/FIS/00068/2013.

Ao Tiago Lopes pela sua assistência na área de instrumentação.

Resumo

Para uma melhor formação dos médicos e para o estudo anatómico do corpo humano uma boa preservação a longo prazo dos cadáveres é necessária, esta obtida através de técnicas de embalsamamento. O embalsamamento é uma técnica de preservação de cadáveres que por um tratamento químico reduz a presença e o crescimento de micro-organismos retardando assim a putrefacção.

Um dos tratamentos químicos que se pode usar para o embalsamento é a injecção de dietilenoglicol, $C_4H_{10}O_3$, no cadáver pelo sistema circulatório. Assim, para esse fim, pretende-se desenvolver um perfusor de rede vascular para a injecção de dietilenoglicol, que visa a conservação do cadáver, ou de um polímero, para produção de moldes para estudo. Isto implica um controlo de fluxo e temperatura dos fluidos conservantes e de moldagem.

Este perfusor usará uma bomba pulsada, uma unidade de controlo baseada num raspberry pi e um ecrã tátil onde será possível visualizar e/ou controlar vários parâmetros inerentes à perfusão, como o fluxo, temperatura, pressão vascular entre outros.

Em relação aos sistemas de perfusão já existentes, este irá possibilitar uma melhor e mais controlada perfusão o que corresponde a uma conservação mais duradoura e tecidos com uma aparência mais natural o que permite um melhor estudo da anatomia do corpo e uma melhor experiência em praticas cirúrgicas.

Palavras-chave: Embalsamento, perfusor de rede vascular, conservação de cadáveres, injecção de dietilenoglicol, moldes de órgãos.

Abstract

For a better training of physicians and for the anatomical study of the human body a good long-term preservation of the cadavers is necessary, this is obtained through embalming techniques. Embalming is a corpse preservation technique which, by chemical treatment, reduces the presence and growth of microorganisms, thereby retarding putrefaction.

One of the chemical treatments that can be used for embalming is the injection of diethylene glycol, $C_4H_{10}O_3$, into the corpse by the circulatory system. Thus, for this purpose, it is intended to develop a vascular network perfusor for the injection of diethylene glycol, for the preservation of the cadaver, or of a polymer, for the production of molds for study. This involves flow and temperature control of the preservative and molding fluids.

This perfuser will use a pulsed pump, a raspberry pi-based control unit and a touch screen where it will be possible to view and/or control various parameters inherent to the perfusion, such as flow, temperature, vascular pressure and others.

In relation to existing perfusion systems, this will allow a better and more controlled perfusion which corresponds to a more lasting preservation and tissues with a more natural appearance which allows a better study of the anatomy of the body and a better experience in surgical practices.

Keywords: Embalming, vascular network perfusor, corpse preservation, diethylene glycol injection, organ molds.

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Capítulo 1

Introdução

A doação de cadáveres para fins de investigação, apesar de ter vindo a aumentar, ainda tem números muito baixos e muitas das intenções de doação nunca chegam a ser feitas seja por esquecimento ou desconhecimento dos familiares.

Face às estatísticas, é necessário melhorar a preservação dos cadáveres e o seu armazenamento. Assim, em 2006, na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL) em colaboração com o Departamento de Anatomia da Faculdade de Ciências Médicas da mesma universidade (FCM-UNL), surge um tema com o objectivo de desenvolver um sistema pré-protótipo para bombeamento de fluidos de conservação em cadáveres humanos.

Este sistema de perfusão[1] tem desde então sido usado no embalsamamento de cadáveres na FCM-UNL para a dissecação, produção de moldes de órgãos, formação e treino dos estudantes de medicina. Tendo apresentado resultados merecedores de melhoramentos surge então o tema para esta dissertação de mestrado.

Neste trabalho é descrito com algum detalhe conceitos de embalsamamento dando ênfase às técnicas mais relevantes para esta dissertação. São também introduzidos conceitos físicos presentes na hemodinâmica e como estes podem influenciar o processo de embalsamamento. Seguidamente é descrito o desenvolvimento do perfusor, materiais e equipamentos usados, código desenvolvido e electrónica montada. É ainda demonstrado os resultados e a analise dos mesmos. Por ultimo, as conclusão retiradas deste trabalho e suas perspectivas futuras.

[1] Perfusão é a passagem de fluido através do sistema circulatório ou sistema linfático para um órgão ou um tecido.

Capítulo 2

Revisão bibliográfica

2.1 Métodos de embalsamamento

O estudo anatómico de um cadáver humano requer a preservação deste a longo prazo, através de técnicas de embalsamamento. O embalsamamento é um processo químico que visa a preservação e a sanitização do corpo humano, por tempo indefinido.

O processo de embalsamamento pode envolver quatro fases o embalsamamento arterial, o embalsamamento de cavidades, o embalsamamento hipodérmico e o embalsamamento de superfície, sendo as duas ultimas complementares das primeiras.

2.1.1 Embalsamamento arterial

O embalsamamento arterial consiste na injecção da solução de embalsamamento no sistema arterial sanguíneo com drenagem do sangue pelas veias. Este método pode ser utilizado no corpo inteiro, grandes secções ou em regiões específicas do corpo[1]. Normalmente a artéria escolhida é a carótida ou femoral, sendo feita uma incisão com profundidade suficiente para que com um gancho a artéria seja levantada e cortada, sendo posteriormente nela inserido um tubo através do qual se vai introduzir a solução de embalsamamento no cadáver.

Uma vez injectada a solução, esta vai percorrer as artérias, arteríolas e capilares, difundindo-se pelos tecidos, percorrendo preferencialmente o percurso que oferece menor resistência. A pressão e a taxa de fluxo da solução, são controladas

com vista a uma melhor distribuição da solução. Durante o embalsamamento, o operador poderá também de ir massajando o cadáver, de modo a evitar a formação de coágulos possam obstruir os vasos sanguíneos e para ir verificar o progressivo sucesso da técnica aplicada.

2.1.2 Embalsamamento de cavidades

O embalsamamento arterial apenas permite o tratamento do tecido muscular, pele e superfície dos órgãos. No entanto, no interior dos órgãos o processo de putrefacção provocado pelas bactérias ainda decorre, comprometendo a conservação do cadáver.

O embalsamamento de cavidades implica a aspiração de líquidos e gases que eventualmente se acumulem no interior dos órgãos abdominais e torácicos. Este processo é realizado através de um *trocar*, ilustrado na figura seguinte.

Figura 2.1: *Trocar* para embalsamamento de cavidades[2]

Este instrumento corresponde a um tubo metálico longo com um bocal de lâminas finas numa extremidade e com um conector na outra, que estabelece a ligação a uma bomba de sucção. As lâminas permitem posicionar o instrumento de troca de forma a criar um ponto de entrada na cavidade direccionado para o órgão cujo conteúdo se pretende aspirar. Quando o conteúdo das cavidades for totalmente aspirado segue-se a fase de injecção de fluido de embalsamamento. A solução de embalsamamento utilizada no preenchimento das cavidades é semelhante à utilizada no embalsamamento arterial, devendo apenas ser mais ácida, de forma a garantir a maior firmeza dos tecidos[3].

2.1.3 Embalsamamento Hipodérmico

Por vezes o embalsamamento arterial não é totalmente eficaz na distribuição da solução de preservação pelo corpo. O embalsamamento hipodérmico é aplicado, como método suplementar, quando os compostos químicos de preservação não chegam a determinadas regiões através do embalsamamento arterial. Isto pode

ser originado pela formação de coágulos que obstruem a passagem da solução para as regiões afectadas ou por o corpo ter sido insuficientemente massajado durante o processo. Este tipo de técnica requer a utilização de uma seringa e uma agulha, sendo a solução injectada localmente e directamente na pele a solução de preservação[3].

2.1.4 Embalsamamento de superfície

O embalsamamento de superfície é efectuado em regiões do corpo danificadas onde a solução de preservação não é distribuída uniformemente, nomeadamente feridas e queimaduras[3]. Nestes métodos são utilizados derivados do formol sob a forma de gel ou aerossol directamente na superfície da região afectada.

2.2 Métodos de injecção de fluidos em cadáveres

O embalsamamento de um cadáver deve ter em conta os seguintes factores:

- Vasos adequados para injecção e drenagem
- Distribuição e difusão da solução de embalsamamento
- Volume e qualidade da solução injectada
- Pressão de injecção
- Fluxo de injecção

A injecção da solução no sistema vascular é realizada pelas artérias pois estas não possuem válvulas, ao contrário das veias[1].

São conhecidos seis técnicas de injecção de fluidos em cadáveres humanos: gravidade, balão ou seringa, combinação da gravidade com seringa, bomba manual, bomba de pressão, bomba de centrífuga.

Das técnicas referidas, a da injecção por gravidade é o mais tradicional, segura, simples e menos dispendiosa. Consiste num recipiente de vidro cheio de fluido de embalsamamento, com uma saída inferior, por onde o líquido escoa. O recipiente é elevado acima do cadáver e o fluido desloca-se para o interior do sistema vascular. A pressão de injecção do fluido na rede vascular depende da altura do recipiente em relação ao cadáver. Este método providencia uma injecção lenta e estável, que promove a distribuição lenta mas eficaz da solução de embalsamamento pelos tecidos do corpo humano, o que representa uma vantagem relativamente a outros

sistemas mecânicos, nos quais a injecção é realizada a velocidades superiores e em que parte do fluido pode atravessar a rede vascular sem ser absorvido pelos tecidos[1].

O método da seringa ou balão faz uso de uma seringa ou de um balão de borracha, cujas extremidades ligam a cânulas. A cânula localizada a montante do balão está inserida num recipiente que contém solução de embalsamamento. Quando o balão é apertado, o fluido desloca-se para o interior do corpo, atravessando o balão e uma cânula que liga o balão ao corpo. A injecção de um cadáver pode ser bastante rápida através deste método, dependendo da frequência com que o balão é apertado. Por ser um método totalmente manual, a injecção exige a presença constante de um operador, que bombeia o fluido para o interior do sistema vascular a uma pressão desconhecida[1].

Os métodos acima descritos podem ser combinados e melhorados. Por exemplo, pode ser utilizado um recipiente elevado em relação ao cadáver, em série com o balão, que a jusante liga ao cadáver através de uma cânula. O corpo pode ser embalsamado pelo método gravitacional, mas a pressão e o fluxo de bombeamento podem ser aumentadas periodicamente apertando o balão, o que constitui uma vantagem relativamente aos dois métodos anteriores[1].

Por vezes é utilizado um sistema manual de injecção vascular que difere dos métodos referidos pelo facto de permitir a criação de vácuo para aspiração[1]. Tal como o método do balão ou seringa, este método exige a presença constante de um operador, sendo que o líquido é bombeado a uma pressão desconhecida.

Alternativamente aos métodos manuais, têm sido também implementados métodos automáticos ou semiautomáticos, que recorrem a bombas de pressão ou de centrifugação, que não requerem a presença do operador junto do equipamento. A pressão de bombeamento pode ser estabilizada e controlada pelo operador variando a velocidade de bombeamento ou alternativamente actuando numa válvula de controlo de fluxo. Em todo o caso todos estes sistemas requerem uma monitorização cuidadosa por parte do operador, já que pecam pela falta de automatismos.

Capítulo 3

Princípios físicos

3.1 Mecânica dos Fluidos

O sistema vascular sanguíneo pode ser compreendido como um sistema complexo de condutas. O estudo do escoamento e distribuição de um fluido pelo sistema vascular requer a compreensão de alguns conceitos da mecânica dos fluidos inerentes a um processo hemodinâmico num sistema de condutas, bem como a forma como estes se relacionam entre si.

3.1.1 Propriedades de um fluido

Um fluido é uma substância que se deforma continuamente quando submetida a uma tensão de cisalhamento[1], não importando o quão pequena possa ser essa tensão. Um subconjunto das fases da matéria, os fluidos incluem os líquidos, os gases, os plasmas e, de certa maneira, os sólidos plásticos. Os fluidos compartilham a propriedade de não resistir à deformação e apresentam a capacidade de fluir[4]. Ao estudar o comportamento de um fluido é importante considerar algumas das suas propriedade físicas, nomeadamente a sua densidade, isotropia, continuidade, compressibilidade, tensão superficial e viscosidade.

A densidade de uma substância define-se como o quociente entre a massa e o volume ocupado pela mesma. Desta forma pode-se dizer que a densidade mede o grau de concentração de massa em determinado volume. A densidade, ρ, é uma

[1] Tensão de cisalhamento, tensão tangencial, ou ainda tensão de corte ou tensão cortante é um tipo de tensão gerado por forças aplicadas em sentidos iguais ou opostos, em direções semelhantes, mas com intensidades diferentes no material analisado.

característica da substância a uma determinada temperatura e a sua unidade SI é quilograma por metro cúbico, $kg.m^{-3}$. Os fluidos em geral apresentam a propriedade da isotropia, ou seja, apresentam as mesmas propriedades físicas independentemente da direcção considerada e considera-se que a distribuição da matéria é contínua. A compressibilidade de um fluido está associada à variação do seu volume quando sujeito a uma pressão. A tensão superficial é um efeito físico que ocorre na camada superficial de um líquido que leva a sua superfície a comportar-se como uma membrana elástica e é expressa no Sistema Internacional em Newton por metro, $N.m^{-1}$. A viscosidade é uma propriedade dos fluidos que representa a resistência à deformação[5]. É a medida de resistência ao fluxo das moléculas de um líquido quando elas deslizam umas sobre as outras. É uma medida inversa à de fluidez.

Considere-se um fluido entre duas placas paralelas infinitas em largura e comprimento, conforme esquematizado na figura 3.1. A placa superior é colocada em movimento sob a acção de uma força F, adquirindo uma velocidade v. A placa inferior permanece em repouso e as camadas do fluido junto às placas permanecem em contacto com estas devido às forças de adesão entre o fluido e as placas. Assim, a camada superior do fluido move-se com a velocidade da placa, cujo módulo é v, e a camada junto à placa inferior não se move. De cima para baixo as camadas vão apresentando um contínuo de velocidades entre v e 0.

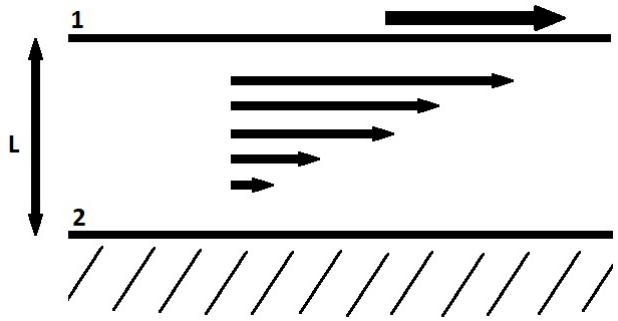


Figura 3.1: Escoamento de um fluido. 1 - Placa móvel; 2 - Placa imóvel.

O módulo da força F, necessária para manter o movimento da placa superior com velocidade v constante, é directamente proporcional à área, A, da placa e ao módulo da velocidade, v, e inversamente proporcional à distância entre placas, L. Assim chega-se à seguinte expressão:

$$F = \eta \frac{Av}{L} \qquad (\mathrm{N.m^{-2}}) \tag{3.1}$$

sendo que η corresponde ao coeficiente de viscosidade do fluido, que depende da substância e da sua temperatura.

Tendo em conta que a tenção de corte, τ, é dada por $\frac{F}{A}$, e na expressão 3.1 F está aplicada sobre A, então essa expressão pode ser escrita como:

$$\tau = \eta \frac{v}{L} \qquad (\mathrm{N.m^{-2}}) \tag{3.2}$$

ou, de um modo geral,

$$\tau = \eta \frac{\Delta v}{\Delta L} \qquad (\mathrm{N.m^{-2}}) \tag{3.3}$$

A expressão 3.3 representa a 2ª Lei de Newton para a viscosidade e o fluido para a qual ela é aplicada denomina-se fluído newtoniano. Esta lei mostra que a tensão de corte é proporcional ao gradiente da velocidade sendo a constante de proporcionalidade é o coeficiente de viscosidade[6].

3.1.2 Lei de Poiseuille

Considerando-se o escoamento dum fluido viscoso em regime laminar através de uma conduta com secção e espessura constantes. Esta conduta é homogénea e isotrópica. Apesar dos vasos sanguíneos não serem nem homogéneos nem isotrópicos, esta corresponde a uma primeira aproximação para compreender a mecânica do sistema vascular[7].

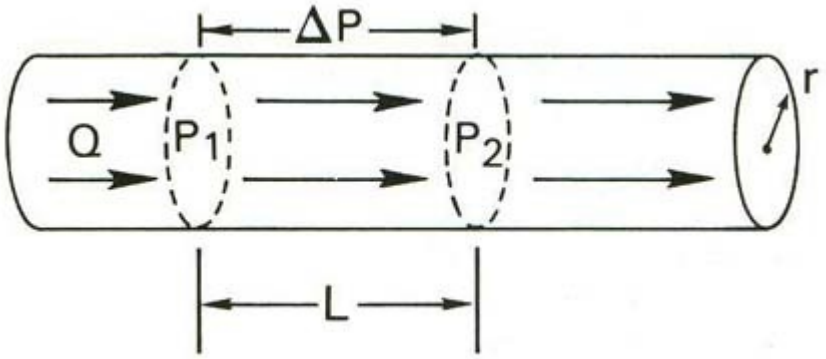


Figura 3.2: Modelo de um vaso do sistema vascular: Q – fluxo; l - comprimento; r - raio da conduta; ΔP - diferença de pressão. Adaptado de[8]

O fluxo é definido como a quantidade de volume que atravessa um elemento infinitesimal de secção da conduta por unidade de tempo[7]. Caso seja constante, é descrito por uma relação entre o raio da conduta, r, e pela velocidade média do fluido, $v_{média}$.

$$Q = \pi r^2 v_{média} \qquad (\mathrm{m^3.s^{-1}}) \tag{3.4}$$

Sobre este fluido actuam forças de viscosidade que se opõem ao movimento e que conduzem a uma diminuição de velocidade média. Estas forças são mais intensas junto das paredes da conduta, diminuindo gradualmente à medida que se aproxima do centro desta, onde a força é mínima[9]. Nos vasos sanguíneos mais largos, as forças de viscosidade não têm um grande efeito no fluxo, pelo que a variação de pressão entre dois pontos da conduta é mínima. No entanto, para vasos sanguíneos mais pequenos a viscosidade causa uma queda de pressão, pelo que P1 > P2. Nestes casos, o fluxo Q entre dois pontos da conduta relaciona-se com a diferença de pressão, ΔP, viscosidade do fluido, η, com o raio, r, e comprimento, l, da conduta, pela seguinte expressão

$$Q = \frac{\pi \Delta P}{8\eta l} r^4 \qquad (\mathrm{m}^3.\mathrm{s}^{-1}) \tag{3.5}$$

Esta é conhecida como lei de Poiseuille e quando combinada com a equação 3.4 obtém-se a velocidade média do fluido na conduta

$$v_{média} = \frac{\Delta P}{8\eta l} r^2 \qquad (\mathrm{m}.\mathrm{s}^{-1}) \tag{3.6}$$

A lei de Poiseuille tem várias aplicações no sistema circulatório humano. À medida que o fluido percorre o sistema arterial sanguíneo desde as artérias mais largas até às mais pequenas, o fluxo diminui significativamente, pois é directamente proporcional a r^4. Isto significa que o fluxo ao longo do sistema circulatório é controlado pela geometria da conduta[9].

3.1.3 Resistência ao escoamento

Os vasos sanguíneos formam uma rede de condutas que transporta o sangue do coração para os vários tecidos do corpo humano. As artérias são os vasos responsáveis pelo transporte do sangue desde o coração até aos tecidos. À medida que se afastam do coração, as artérias ramificam-se em vasos de menor diâmetro até que no interior dos tecidos se encontram sob a forma de capilares.

Para melhor se entender a resistência ao escoamento dum fluido ao longo dos vários capilares é possível fazer uma analogia com o fenómeno de resistência à passagem de corrente num circuito eléctrico RLC[2]. Esta analogia deve-se ao facto da circulação sanguínea envolver o movimento de um fluido com massa, ao qual está associado uma indutância no momento de aceleração ou desaceleração; o fluido é viscoso e está sujeito a forças de resistência ao movimento nas paredes dos

[2]Circuito elétrico consistindo de uma resistência (R), um indutor (L), e um condensador (C)

vasos; e os vasos são elásticos pelo que sofrem alterações de volume, produzindo efeitos capacitivos[10].

Num circuito eléctrico, a resistência depende da disposição das cargas. Se estas se encontrarem em paralelo, a resistência equivalente vai diminuir com o número de cargas instaladas. No caso das cargas estarem associadas em série, a resistência equivalente vai aumentar com o número de cargas.

Para determinar a resistência ao escoamento de um fluido ao longo dos capilares é possível recorrer a uma analogia com a lei de Ohm, em que $R = \frac{V}{I}$. Nesta analogia, a diferença de potencial eléctrico, V, é substituída pela diferença de pressão, ΔP, a resistência eléctrica, R, é substituída pela resistência ao escoamento do fluido, R_e, e a corrente eléctrica, I, é substituída pelo fluxo do escoamento, Q. Nestes termos, a resistência ao escoamento é dada por:

$$R_e = \frac{\Delta P}{Q} \qquad (\text{N.s.m}^{-5}) \tag{3.7}$$

Sendo que através da equação 3.5 que descreve a Lei de Poiseuille podemos assim concluir que a resistência ao escoamento ao longo de um vaso sanguíneo é dada por:

$$R_e = \frac{8\eta l}{\pi r^4} \qquad (\text{N.s.m}^{-5}) \tag{3.8}$$

Na mesma linha de raciocínio, a indutância de uma artéria, L, em analogia com a indutância de uma bobina, é descrita pela expressão:

$$L = \frac{\rho l}{\pi r^2} \qquad (\text{Kg.m}^{-4}) \tag{3.9}$$

em que ρ é a densidade do fluido. Por sua vez, a capacitância corresponde à complacência[3] da artéria, C. A complacência depende do volume da artéria, V, e da pressão exercida pelo fluido nas paredes da artéria, ΔP. É dada pela expressão:

$$C = \frac{\Delta V}{\Delta P} \qquad (\text{N}^{-1}.\text{m}^5) \tag{3.10}$$

3.1.4 Regimes de escoamento

Na análise do escoamento de fluidos deve-se distinguir dois regimes: o regime laminar e o regime turbulento. No regime laminar o fluxo é relativamente lento, a velocidade de escoamento num dado ponto é independente no tempo e a direcção de escoamento é paralela às paredes do tubo condutor. Para velocidade mais elevadas surgem movimentos de rotação das partículas, que geram movimentos

[3] [Fisiologia] Capacidade de distensão de certas estruturas elásticas como os vasos sanguíneos.

desordenados com flutuações de velocidade ao longo do tempo, o que corresponde a um regime turbulento. Entre estes dois regimes de escoamento existe uma região de transição, na qual o escoamento é misto, flutuando entre os regimes laminar e turbulento, antes de se tornar totalmente turbulento[7].

O regime do escoamento de um fluido depende da sua viscosidade, η, e densidade, ρ, da velocidade de escoamento, v, e da geometria do condutor, que num vaso sanguíneo será dependente do raio, r, podendo ser determinada a partir do número de Reynolds, R_e, que é definido por:

$$R_e = 2\frac{\rho v r}{\eta} \tag{3.11}$$

O regime é laminar para R<2000 e turbulento para R>4000. Para valores entre 2000 e 4000 tem-se um regime de transição, no qual o fluido apresenta características dos dois regimes[7].

3.2 Mecânica dos vasos sanguíneos

3.2.1 Lei de Hooke

A lei de Hooke é a lei da física relacionada à elasticidade de corpos, que serve para calcular a deformação, não permanente, causada pela força exercida sobre um corpo, tal que a força, F, é igual ao deslocamento da massa, x, a partir do seu ponto de equilíbrio vezes a constante elástica do corpo, k[11]:

$$F = kx \qquad \text{(N)} \tag{3.12}$$

3.2.2 Deformação da parede arterial

As forças que actuam numa artéria provocam uma deformação instantânea. Essa deformação vai dar lugar a forças interiores que a mantêm em equilíbrio estático. Se estas forças forem eliminadas, a artéria volta à sua forma inicial. Este fenómeno designa-se por elasticidade da artéria.

Considere-se a área de secção de uma conduta com raio interno r e espessura h. Nas artérias, a razão entre a espessura e o raio interno está compreendida entre 0,1 e 0,15.

$$\frac{h}{r} \cong 0,1 \; a \; 0,15 \tag{3.13}$$

A mecânica dos vasos sanguíneos não depende apenas da sua geometria, pois compreende também as suas propriedades elásticas que são descritas pelo seu

módulo de elasticidade, E[7]. A deformação instantânea é chamada de deformação elástica. Isto verifica-se sempre que a deformação não seja demasiado elevada. Para pequenas deformações, inferiores a 10%, aplica-se a lei de Hooke. O alongamento experimentado por um material elástico, ao ser submetido à acção de uma força deformadora, é directamente proporcional a esta, sempre que a referida força não ultrapasse determinado limite, designado de limite de elasticidade, Sy, e que depende do material em questão. Quando este limite é ultrapassado a deformação deixa de ser elástica e o material fica permanentemente deformado, resultando numa deformação plástica[7].

A resposta elástica de uma artéria não é linear. As artérias podem estar sujeitas a grandes deformações em que não obedecem à lei de Hooke, tornando-se mais rígidas quando a tensão ou deformação exercida sobre elas aumenta. A expansibilidade da artéria surge como resposta a um aumento da pressão arterial como consequência de um aumento do volume de fluido. Para uma dada variação de volume, se a variação de pressão for elevada, a capacidade arterial vai ser menor e as propriedades elásticas da artéria diminuem. Quando a elasticidade é boa, as artérias dilatam facilmente e não há um aumento de pressão significativo, pelo que existe uma grande capacidade arterial.

À medida que as artérias se vão afastando do coração, vão ficando mais musculosas, aumentando assim a percentagem de músculo liso[4] na parede arterial. A contracção ou relaxamento do músculo liso tem influência sobre a elasticidade arterial, a contracção do músculo liso diminui a elasticidade, ou seja, aumenta a rigidez arterial e a relaxação do músculo liso vai ter o efeito contrário.

[4]Músculo liso é um tecido muscular de contração involuntária e lenta, encontra nas paredes de órgãos ocos, tais como os vasos sanguíneos, na bexiga, no útero e no trato gastrointestinal. Nos vasos sanguíneos tem o papel de impulsionar o sangue.

Capítulo 4

Desenvolvimento e automatização do sistema de perfusão

Em colaboração com o Departamento de Anatomia da Faculdade de Ciências Médicas da Universidade Nova de Lisboa foi desenvolvido em 2006 um pré-protótipo de perfusor para embalsamamento de cadáveres. Em 2008, foi efectuado um protótipo de um perfusor para embalsamento de cadáveres, em colaboração com o Departamento de Anatomia a Faculdade de Ciências Medicas. Este era constituído por um deposito de fluido onde o mesmo era aquecido, sendo posteriormente bombeado para o interior do cadáver. Para aquecimento do fluido foi utilizado um sensor de temperatura e uma resistência de aquecimento, inseridos dentro do deposito. O controlo (ON/OFF) da resistência foi executado através do accionamento de um relé de estado sólido. Relativamente ao controlo de fluxo, este foi efectuado através da aquisição do mesmo, utilizando um sensor instalado à saída da bomba de diafragma. Sendo esta accionada por um potenciómetro digital ou analógico, de acordo com o modo seleccionado previamente pelo utilizador para o funcionamento do sistema, automático ou manual, respectivamente.

4.1 Projecto do Perfusor

A utilização deste protótipo no embalsamamento permitiu compreender o funcionamento do sistema de perfusão bem como as suas limitações. Segundo os responsáveis da FCM, este sistema de perfusão alcançou desempenhos superiores às bombas centrífugas previamente utilizadas, muito embora existam ainda

alguns componentes que merecem ser melhorados.

Ao longo do processo de embalsamamento observado registou-se o mau funcionamento dos sensores de pressão e de fluxo, já que os respectivos medidores não indicavam valores passíveis de serem lidos. Também pelo facto da bomba não ter qualquer informação acerca do fluxo a que opera, não era possível saber exactamente o volume de solução injectado no cadáver. No que diz respeito ao controlo de temperatura, foi possível conferir uma acentuada demora no aquecimento do liquido do deposito e na estabilização dessa temperatura, devido às dimensões do reservatório (15L). O equipamento apresenta dimensões excessivas para o local onde é implementado o que também dificulta a sua movimentação para junto do cadáver.

Assim, para melhorar estes aspectos e implementar novas funcionalidades, foi projectado um novo sistema totalmente programável e automático, mais compacto, sistema de aquecimento mais eficaz, controlo do fluxo mais rigoroso e com registo dos dados sobre o espécime. Tendo em conta as implementações pretendidas, o novo sistema proposto encontra-se esquematizado na figura 4.1.

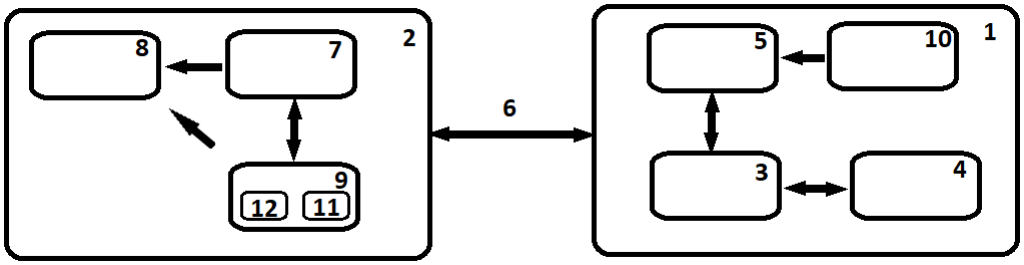


Figura 4.1: Esquema de blocos do perfusor desenvolvido: 1 - Unidade de controlo; 2 - Unidade e bombeamento; 3 - Computador de placa única[1](Raspberry Pi); 4 - Interface gráfica (Ecrã táctil); 5 - Electrónica auxiliar da unidade de controlo; 6 - Comunicação entre as duas unidades pelo cabo RJ-45; 7 - Electrónica auxiliar da unidade de bombeamento; 8 - Bomba de diafragma; 9 - Deposito de aquecimento; 10 - Sensores de pressão; 11 - Sensor de temperatura; 12 - Carga de aquecimento

As opções e os componentes utilizados no desenvolvimento deste sistema serão detalhados nas secções seguintes.

4.1.1 Unidade de Controlo

Na unidade de controlo encontra-se um Raspberry Pi onde corre um programa completamente dedicado, este é o core da unidade de controlo. Tem-se também

[1]Computador de placa única é um computador onde todos os componentes electrónicos necessários para o seu funcionamento estão situados numa única placa de circuito impresso.

um ecrã táctil onde é demonstrada uma interface gráfica desenhada à medida das necessidades implementadas. No ecrã o utilizador também pode inserir comandos disponíveis na interface. Ainda na unidade de controlo há uma electrónica auxiliar que faz ponte entre o Raspberry Pi e algumas periféricos como a iluminação de controlo ou os sensores de pressão.

4.1.1.1 Raspberry PI

Raspberry Pi é uma série de computadores de placa única do tamanho reduzido, que se pode conectar a um monitor, usa um teclado e um rato padrão e foi desenvolvido no Reino Unido pela Fundação Raspberry Pi.

Neste projecto é usado um Raspberry Pi 3 modelo B, este contem um processador 1.2GHz 64-bit quad-core ARMv8 CPU, 1 GB de RAM, Bluetooth 4.1 e wireless 802.11n. Este é alimentado a 5V DC.

São também usados 2 módulos complementares do Raspberry Pi. Um modulo RTC[2], uma vez que o próprio Raspberry Pi não tem um incluído e para registo é importante anotar a data e hora do embalsamamento. O modulo RTC usado é da marca PiFace usa uma conexão I^2C e uma pilha de 3V. O segundo modulo usado é um conversor de sinal analógico para digital, uma vez que o sinal dos sensores de pressão é analógico. O modulo usado foi um ADS1115 de 16bits, este usa uma conexão I^2C.

4.1.1.2 Ecrã táctil

A interação do utilizador com o equipamento é feita através de um ecrã táctil, neste o operador tem acesso a um interface gráfica onde pode controlar o funcionamento do perfusor bem como monitorizar o processo corrente.

O ecrã táctil usado neste projecto é da marca Element14 tem 7" e uma resolução de 800 x 480 pixels.

4.1.1.3 Electrónica auxiliar

Foi também necessário implementar alguma electrónica auxiliar, esta consiste principalmente num sistema de relés para controlar a iluminação e numa fonte de alimentação uma vez que o Raspberri Pi é alimentado a 5V DC e a unidade de controlo a 250V AC.

Usando o programa Kicad foi desenhada uma placa de circuito eléctrico, apêndice A.1. Esta foi impressa em papel de fotografia, foi depois gravada numa placa

[2]Relógio de tempo real (RTC ou real-time clock, em inglês) é um relógio de computador que mantém o controlo do tempo presente.

para circuitos eléctricos (PCB) aplicando calor. Usando Cloreto de ferro(III), $FeCl_3$, o cobre exposto da PCB é erodido ficando só os caminhos de cobre protegidos pela impressão.

4.1.2 Caixa

Como invólucro para ambas as unidades, de controlo e bombeamento, foram usadas caixas já previamente desenhadas e desenvolvidas pelo laboratório, no apêndice A.2 tem-se vários ângulos do modelo 3D usado para a construção das caixas. Devido a novas implementações ao sistema foram feitas algumas alterações às caixas. Nas figuras 4.2 e 4.3 tem-se fotos do sistema já montado. O material das caixas é aço-inoxidável pois este é fácil de limpar e tem uma longevidade maior no ambiente em que vai ser expostos.

Figura 4.2: Foto da unidade de controlo.

Figura 4.3: Fotos da unidade de bombeamento.

4.1.3 Unidade de bombeamento

A unidade de bombeamento divide-se em 4 sub-blocos principais: uma bomba de diafragma, electrónica auxiliar, o deposito de aquecimento e electro-válvulas para controlar a direcção do fluxo. Estes serão detalhados nas secções seguintes.

4.1.3.1 Bomba de diafragma

A escolha da bomba no procjeto teve em conta essencialmente a quantidade de líquido injectado por unidade de tempo.A quantidade de solução de embalsamamento injectada num cadáver é aproximadamente 10 L. Num sistema de funcionamento manual, onde a presença de um operador é indispensável, pretende-se que esta quantidade seja injectada no cadáver a uma taxa relativamente elevada, a fim de evitar a exposição prolongada do operador aos efeitos potencialmente nocivos da solução de embalsamamento. No entanto, num processo automatizado, tendo em vista uma optimização da distribuição da solução, a perfusão deve ser longa, pelo que se pretende que o fluxo seja baixo. É ainda necessário ter em conta que a pressão de bombeamento deve ser suficientemente elevada para que solução consiga ultrapassar qualquer obstáculo que possa eventualmente surgir no interior do sistema vascular.

Atendendo a estas características importantes, foi escolhida uma bomba Blackstone BL 15, 4.4, que tem a capacidade de bombear até cerca de 20 L/h, funcionando numa gama de pressão desde 1 bar até 8 bar.

Figura 4.4: Bomba de diafragma Blackstone BL15

Esta bomba tinha originalmente um controlo externo de fluxo, usando um potenciómetro, na parte frontal da bomba, que lhe permite ajustar a percentagem

de fluxo desde 0% a 100% da máxima capacidade de dosagem. Esta característica foi removida e substituída para que o controle passe a ser feito pela unidade de controlo. As características da bomba estão resumidas na tabela 4.1.

Tabela 4.1: Características técnicas da bomba de diafragma.

Características técnicas	
Marca	Blackstone
Tensão de Alimentação	220/240 V a 50/60 Hz
Temperatura de funcionamento	0 a +50 ºC
Revestimento exterior	Polipropileno reforçado
Potência máxima absorvida	200 W
Materiais	Cabeça da bomba em Kynar; Diafragma e válvulas em Teflon; Válvulas de revestimento e O-Rings em Viton

Esta bomba é constituída por materiais de alta qualidade, com elevada resistência à corrosão, como o *Kynar*, o *Teflon* e o *Viton*, e que fornecem protecção às partes em contacto com os químicos corrosivos.

O bombeamento é realizado através do deslocamento de um solenóide[3], resultando no menor movimento das peças e consequentemente numa redução da probabilidade da ocorrência de falhas mecânicas. Trata-se de uma bomba de precisão pois cada batida do pistão é exactamente igual à batida anterior e à batida seguinte, o que garante um fluxo constante na saída[12].

Estas características garantem uma gama de fluxo e de pressão adequada às necessidades da perfusão, bem como longevidade face às características corrosivas da solução de embalsamamento. O facto de se tratar de uma bomba de diafragma garante uma perfusão pulsada, isto é, com um período de relaxamento entre dois picos de pressão, correspondentes ao deslocamento sucessivo do solenóide. Esta característica favorece a perfusão, pois permite que os vasos e os tecidos se adaptem ao fluido, permitindo que este se difunda uniformemente por todas as zonas do corpo.

4.1.3.2 Electrónica auxiliar

Tal como na unidade de control, a unidade de bombeamento também requer electrónica auxiliar. Aqui está consiste um sistema de relés que controla a iluminação e as electro-válvulas de fluxo, numa ponte retificadora para a bomba e numa ventoinha para extrair o ar quente da caixa.

[3] solenóide é um condutor enrolado em forma de espiral

Foram escolhidos relés do estado solido com acoplamento óptico, para isolar fisicamente as zonas de elevado potencial electrico e as de baixo potencial electrico, uma vez que a bomba e as válvulas são cargas indutivas e geram picos de alta tensão. Sem este isolamento estes pico podem danificar o raspberri pi entre outros componentes.

4.1.3.3 Electro-válvulas de fluídos

Uma vez sendo o modo de aquecimento opcional, o deposito de aquecimento não é usado quando este esta desligado. Assim é necessário definir o caminho que o fluído deve de tomar. Para isso é usado um sistema de electro-válvulas, representado na figura 4.5, controlado pelo Raspberry Pi. As válvulas 4 e 5 estão ambas abertas quando o modo de aquecimento está ligado e fechadas quando este está desligado, a válvula 6 tem o comportamento oposto.

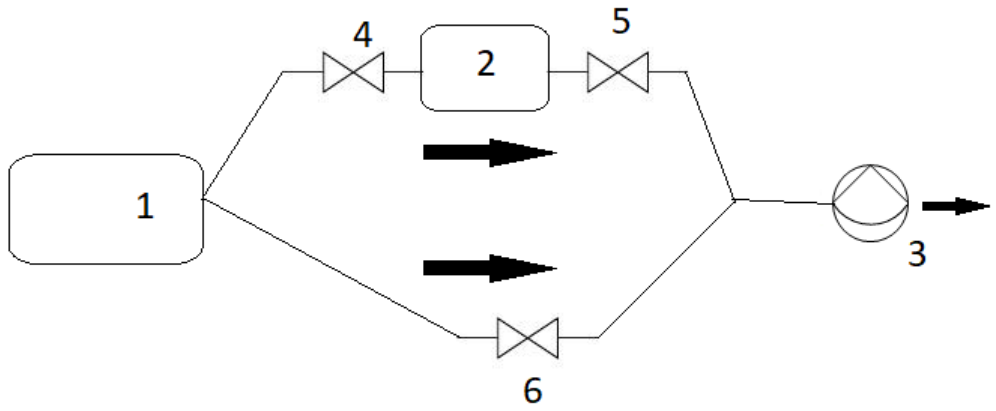


Figura 4.5: Esquema do sistema de válvulas: 1 - Reservatório externo do fluído embalsamador; 2 - Deposito de aquecimento; 3 - Bomba de diafragma; 4 - Electro-válvula à entrada do deposito de aquecimento; 5 - Electro-válvula à saída do deposito de aquecimento; 6 - Electro-válvula à entrada da bomba. As setas indicam o sentido geral do fluído.

4.1.3.4 Deposito de aquecimento

O deposito de aquecimento consiste num recipiente interno à unidade de bombeamento onde, usando uma carga de aquecimento, o fluído é aquecido e mantido à temperatura pretendida enquanto novo fluído frio entra e o já aquecido sai.

Para o recipiente foi usado um já previamente concebido pelo laboratório, no apêndice A.3 tem-se um modelo 3D usado para a construção de deposito.

4.1.3.5 Carga de aquecimento

Para aquecer o fluído é usado uma resistência de 500W. Esta é controlada pelo Raspberry Pi atravez de um controlador PID[4]. Os parâmetros para o controlador PID foram estimados e depois ajustados manualmente por tentativa e erro até se obter um comportamento do aquecimento desejado.

4.1.3.6 Sensor de temperatura

O sensor de temperatura encontra-se dentro do deposito de aquecimento, consiste num sensor DS18B20 embutido num invólucro de aço-inoxidável. Este usa uma conexão I^2C para comunicar com o Raspberry Pi.

4.1.4 Sensor de pressão

Na perfusão é importante medir a pressão em vários pontos do sistema vascular, de forma a não ocorrerem roturas das condutas. Para esse efeito, será necessário efectuar a aquisição da pressão à saída da unidade de bombeamento e em outros pontos, do sistema vascular, à consideração do operador.

Para medir a pressão foram escolhidos sensores piezoeléctricos, de pressão manométrica que usam como referencia a pressão atmosférica. São constituídos por um diafragma de silício com montagem em ponte de *Wheatstone*[5], figura 4.6. As características do mesmo estão resumidas na tabela 4.2.

[4]Controlador proporcional integral derivativo, controlador PID ou simplesmente PID, é uma técnica de controle de processos que une as ações derivativa, integral e proporcional, fazendo assim com que o sinal de erro seja minimizado pela ação proporcional, reduzido a zero pela ação integral e obtido com uma velocidade antecipativa pela ação derivativa.

[5]Ponte de *Wheatstone* é um esquema de montagem de elementos elétricos que permite a medição do valor de uma resistência elétrica desconhecida

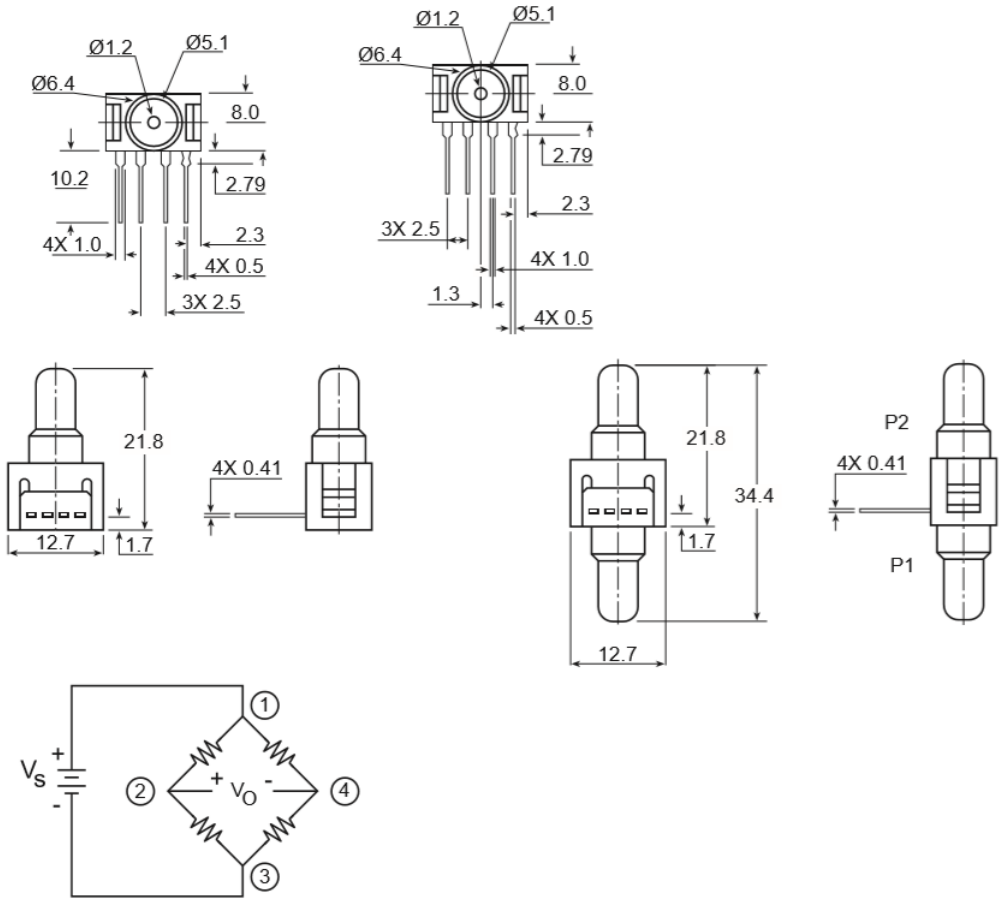


Figura 4.6: Especificações do sensor de pressão: 1 - Pin 1, V_s(+); 2 - Pin 2, Output(+); 3 - Pin 3, V_s(-) (Terra); 4 - Pin 4, Output(-). Adaptado de [13].

Tabela 4.2: Características técnicas do sensor de pressão.

Características técnicas	
Marca	Honeywell
Tensão de Alimentação	10V DC
Temperatura de funcionamento	- 40ºC a + 85ºC
Gama de pressão	0 a 5psi
Tempo de resposta	1ms
Sensibilidade	10mV/psi

Para calcular os parâmetros da recta de calibração foi usada uma coluna de água. Por este método sabendo a altura da coluna e a secção do sensor pode-se calcular a pressão.

4.2 Código desenvolvido

O código desenvolvido pode ser dividido em duas partes: o programa de controlo e a interface gráfica. A primeira consiste no código que carrega a interface gráfica assim que o programa é iniciado e executa os vários comandos que o utilizador pode inserir usando a interface gráfica. Também é a parte responsável pelo tratamento dos dados recebidos e pelo controlo dos sinais de saída. O código desenvolvido encontra-se no apêndice A.4.

A interface gráfica foi desenhada tendo em consideram dos requisitos pretendidos.

Os comandos disponíveis ao utilizador, através de botões na interface gráfica (figuras 4.7 e 4.8), são os seguintes:

- Desligar o equipamento. Quando pressionado este botão abre uma janela de confirmação.
- Guardar os dados do corrente processo de perfusão num ficheiro txt, este pode ser aberto depois por Excel^{TM} para tratar os dados. Ao pressionar este botão abre uma janela para seleccionar o destino do ficheiro, na unidade de controlo o utilizador tem acesso a duas portas de USB onde pode colocar uma *pen* e guardar o ficheiro na mesma.
- Inserir ou modificar espécime. Ao pressionar este botão uma nova interface gráfica e carregada, nesta o utilizador pode inserir dados sobre o espécime.
- Mudar o modo de funcionamento entre automático ou manual. No mudo manual o utilizador indica o fluxo pretendido e o processo só parara por ordem do operador, este também pode ir mudando o fluxo a qualquer momento. No mudo automático o utilizador pode seleccionar diferentes fluxos e intervalos de tempo para os mesmos.
- Desligar/ligar modo de aquecimento. Ao ligar o modo de aquecimento o utilizador pode definir uma temperatura, também mudar a mesma a qualquer momento.
- Iniciar/parar processo. Quando iniciado o processo o utilizador já não pode mudar o modo de funcionamento nem o aquecimento até parar o processo.

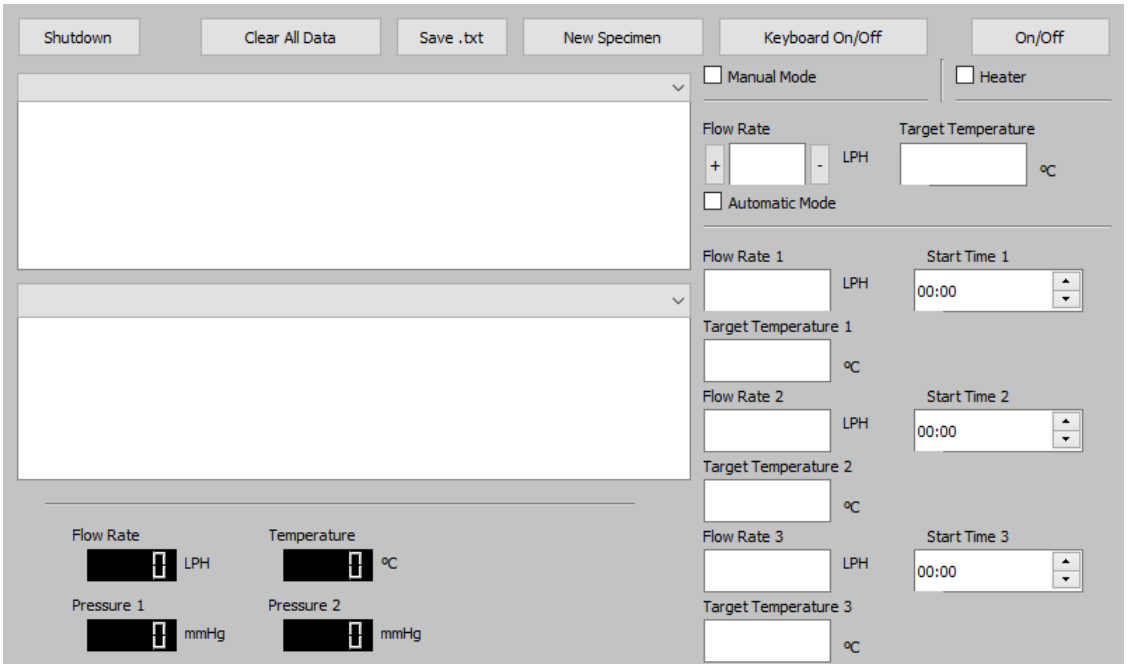


Figura 4.7: Interface gráfica principal

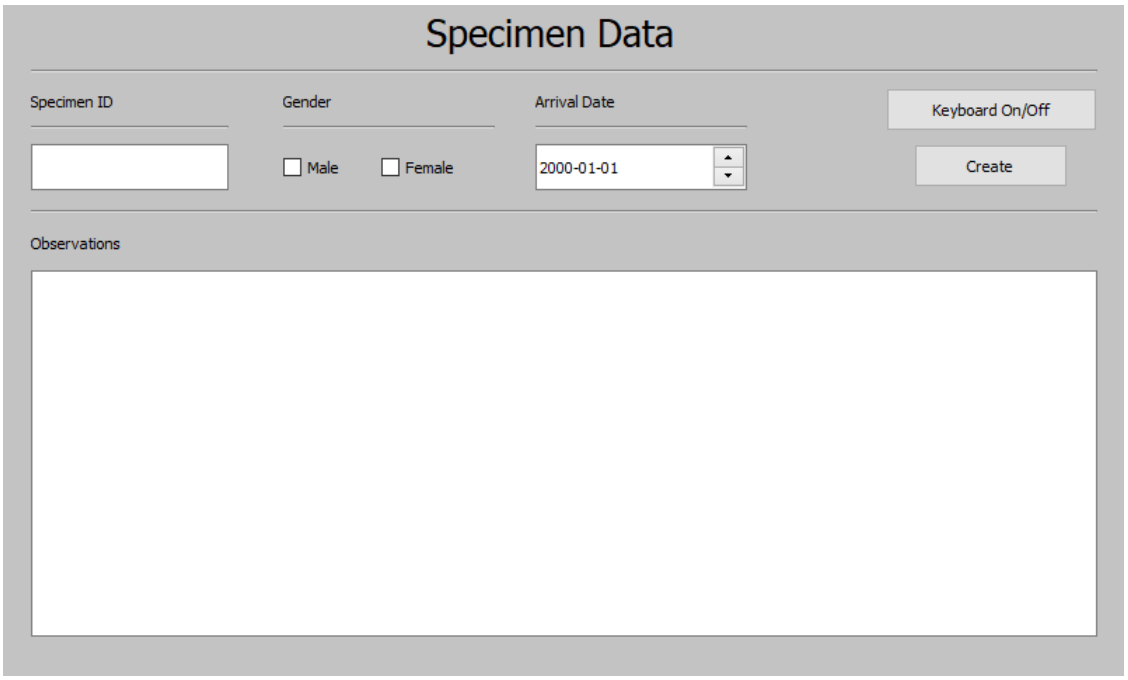


Figura 4.8: Interface gráfica para informação sobre o espécime

Capítulo 5

Conclusões

O equipamento desenvolvido constitui uma grande evolução em relação ao anterior, pois foram introduzidas soluções inovadoras que permitiram corrigir muitas das fragilidades verificas.

Foram introduzidas melhorias significativas na medição da pressão periférica, no sistema de aquecimento, na interface com o utilizador e nas dimensões do equipamento. Foi ainda incorporado um novo sistema de controlo de fluxo programável pelo utilizador, que permite definir previamente todos os fluxos e respectivas temporizações, ao longo de todo o processo. É também permitido ao utilizador ver e guardar os dados da perfusão.

Conclui-se assim que o sistema desenvolvido, além de ser uma solução totalmente inovadora no sector, apresenta uma enorme evolução face ao prototipo anterior, revelando-se de enorme importância no estudo e optimização de técnicas de embalsamamento.

Bibliografia

[1] Robert G. Mayer. *Embalming: History, Theory and Practice*. McGrawHill, 2006.

[2] From Wikimedia Commons. *Example of a reusable trocar*. [Consultado em: 2019/01/30]. URL: https://en.wikipedia.org/wiki/Trocar#/media/File:Trocar.jpg.

[3] M. L. Ajmani. *Embalming: principles and legal aspects*. 1st edition 1998.

[4] Hugh D. Young e Roger A. Freedman. *Física II: Termodinâmica e Ondas (12ª ed.)* Addison Wesley, 2008.

[5] Keith Symon. *Mechanics (3ª ed.)* Addison Wesley, 1971.

[6] Ronald L. Panton. *Incompressible Flow (4ª ed.)* John Wiley & Sons, 2013.

[7] Lee Waite e Jerry Fine. *Applied Biofluid Mechanics*. McGrawHill, 2007.

[8] Smith e Kampine 1990. *Poiseuille's Law*. [Consultado em: 2018/12/03]. URL: http://www.cai.md.chula.ac.th/lesson/lesson4711/html/pic1.html.

[9] Vincent Coletta. *College Physics*. McGrawHill, 1995.

[10] M. Zamir. *The Physics of Coronary Blood Flow*. AIP Press, 2005.

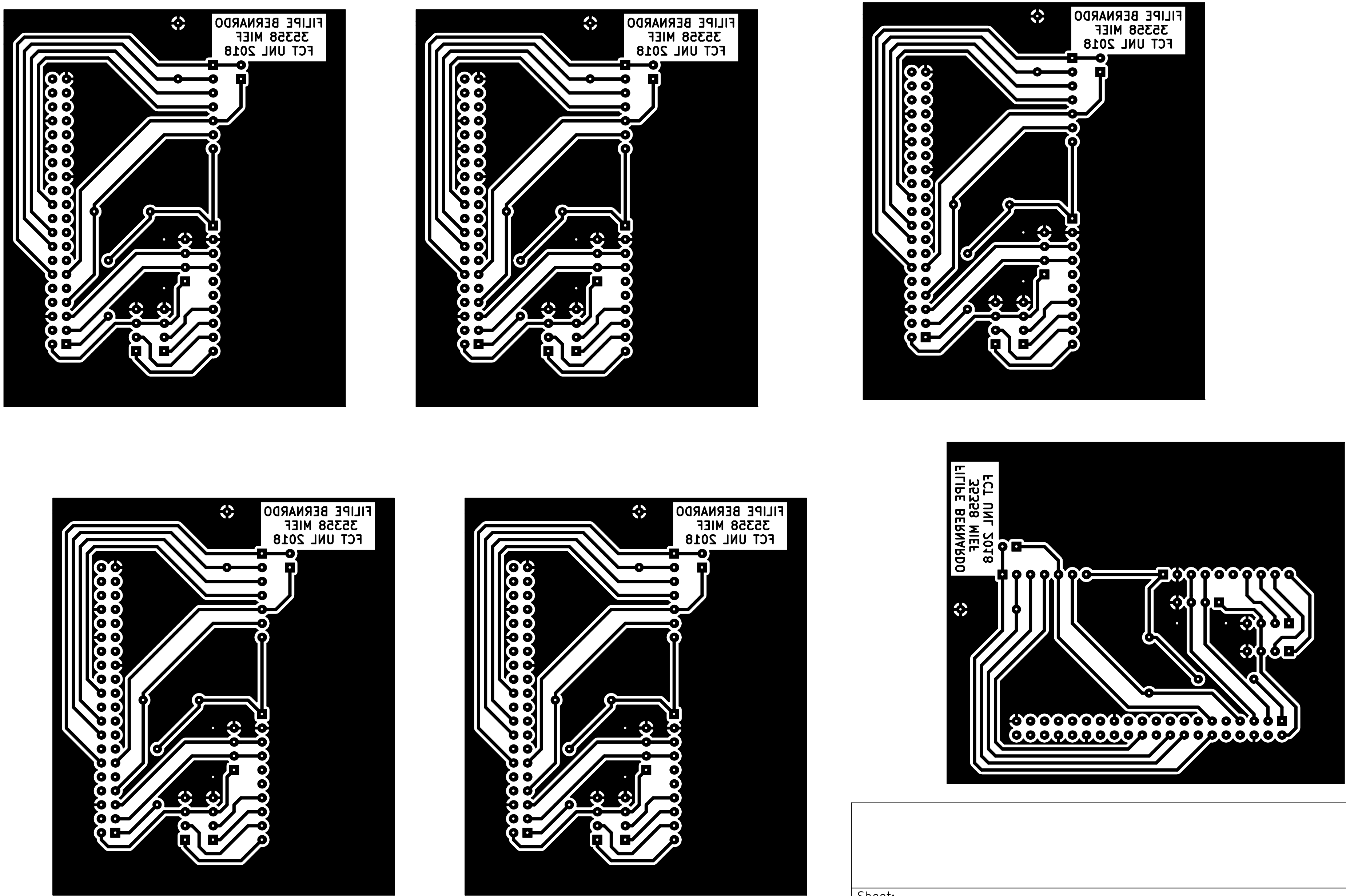
[11] David Halliday. *Física 2, volume 1 (5ª ed.)* LTC, 2004.

[12] *Manual de instruções da bomba Blackstone BL15*.

[13] *Documentação técnica do sensor 26PCBFA6D da Honeywell*.

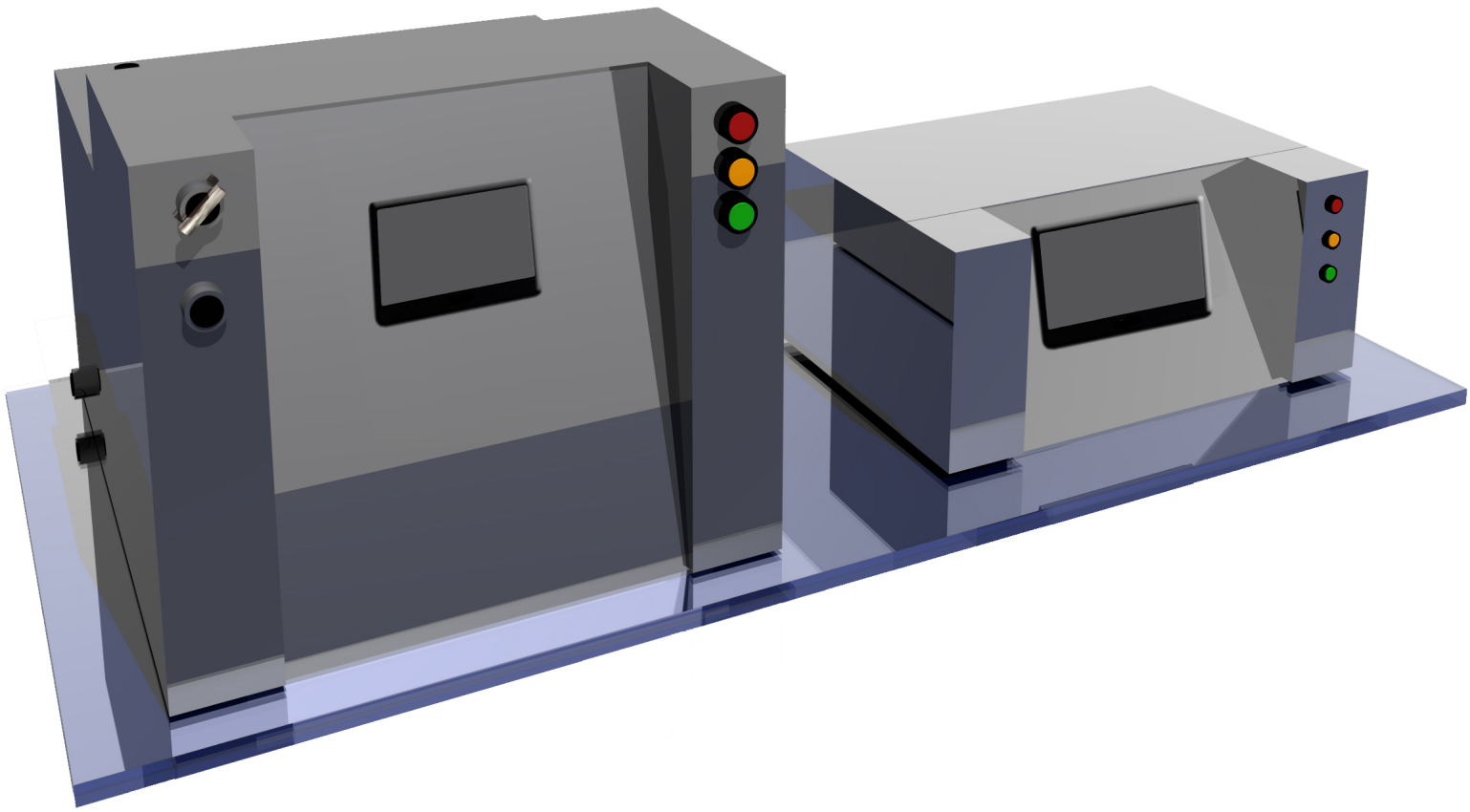
Apêndice

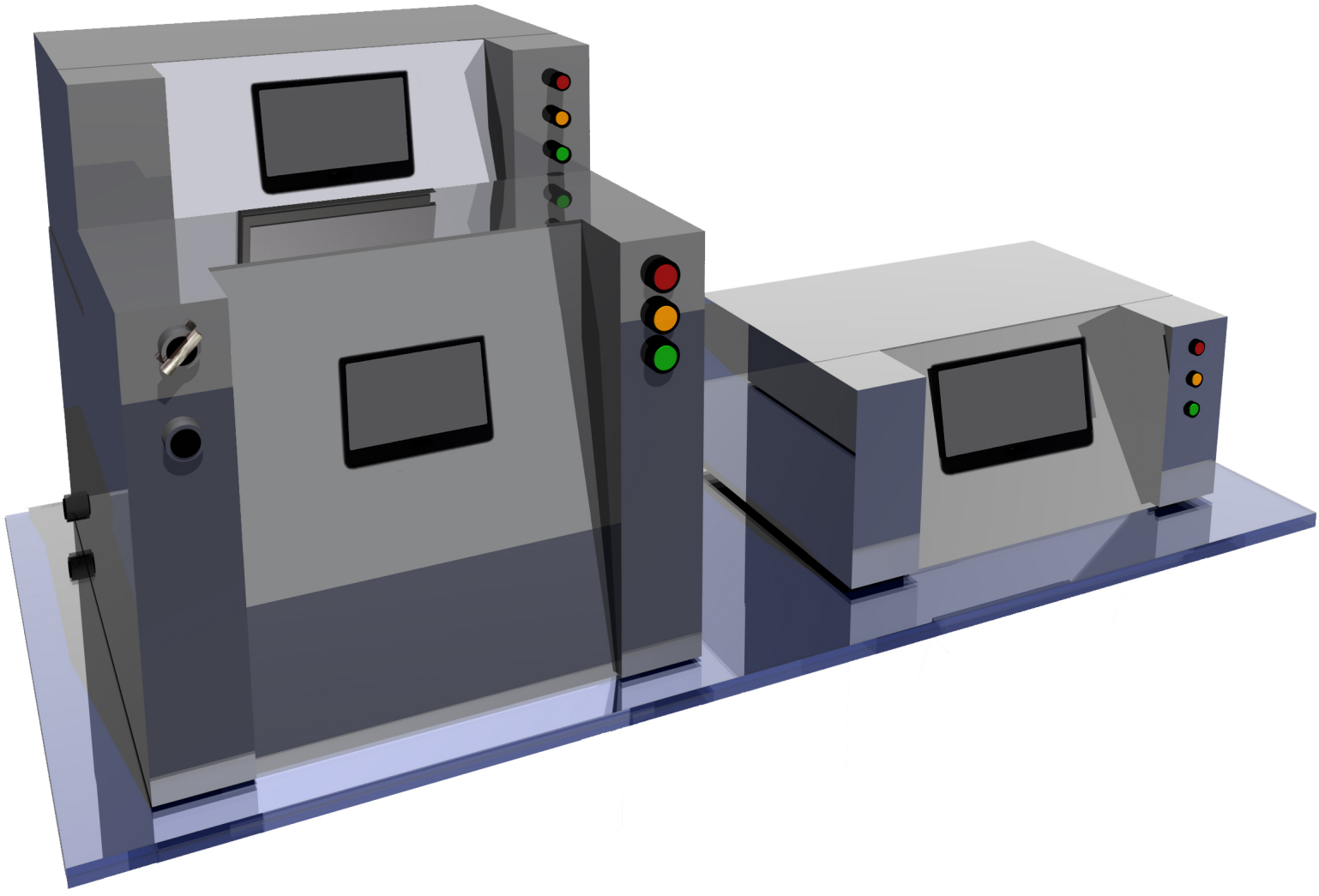
A.1 Esquemático em Kicad

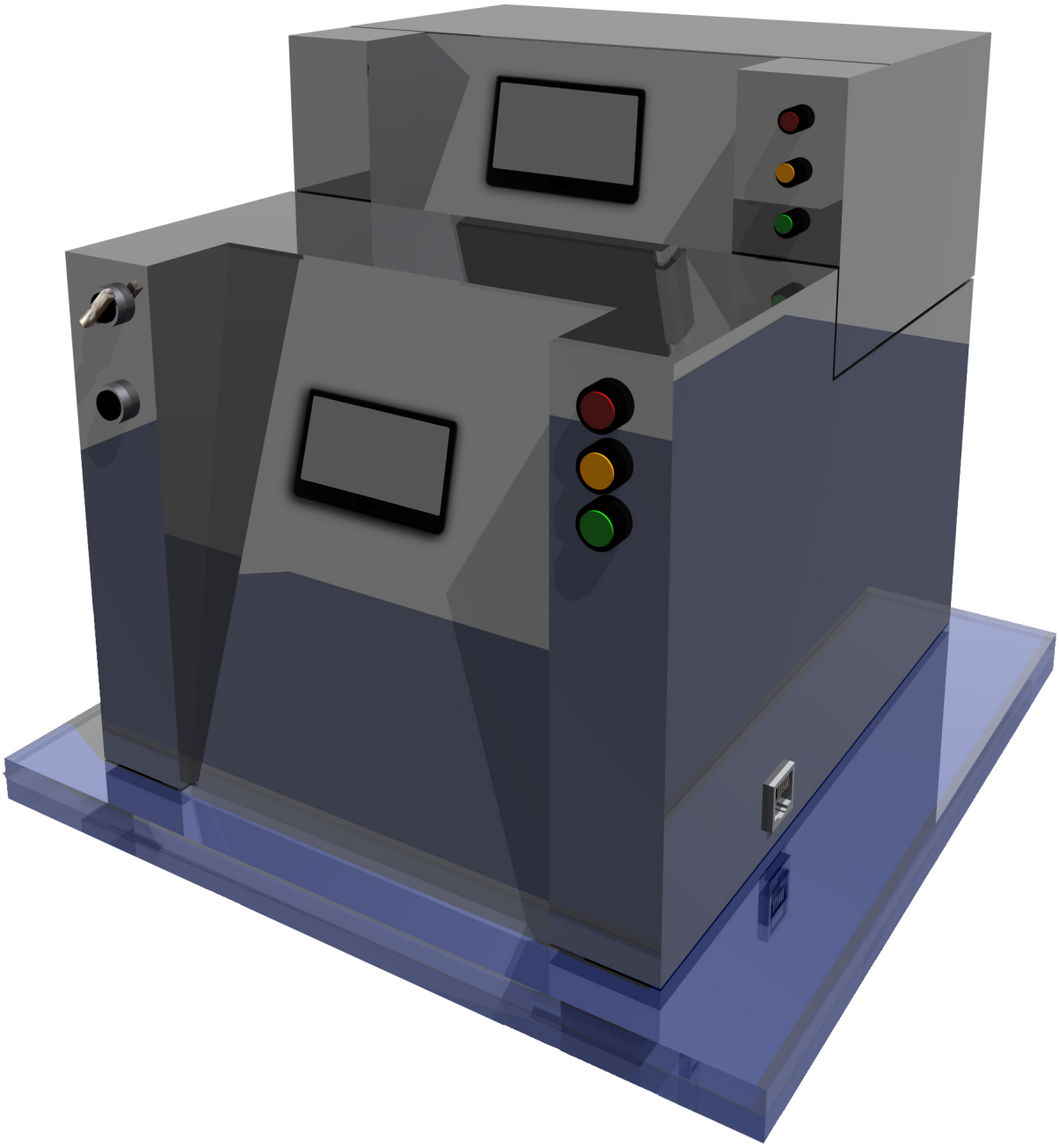


Sheet:		
File: PlacaFilipe.kicad_pcb		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. pcbnew 4.0.7		Id: 1/1

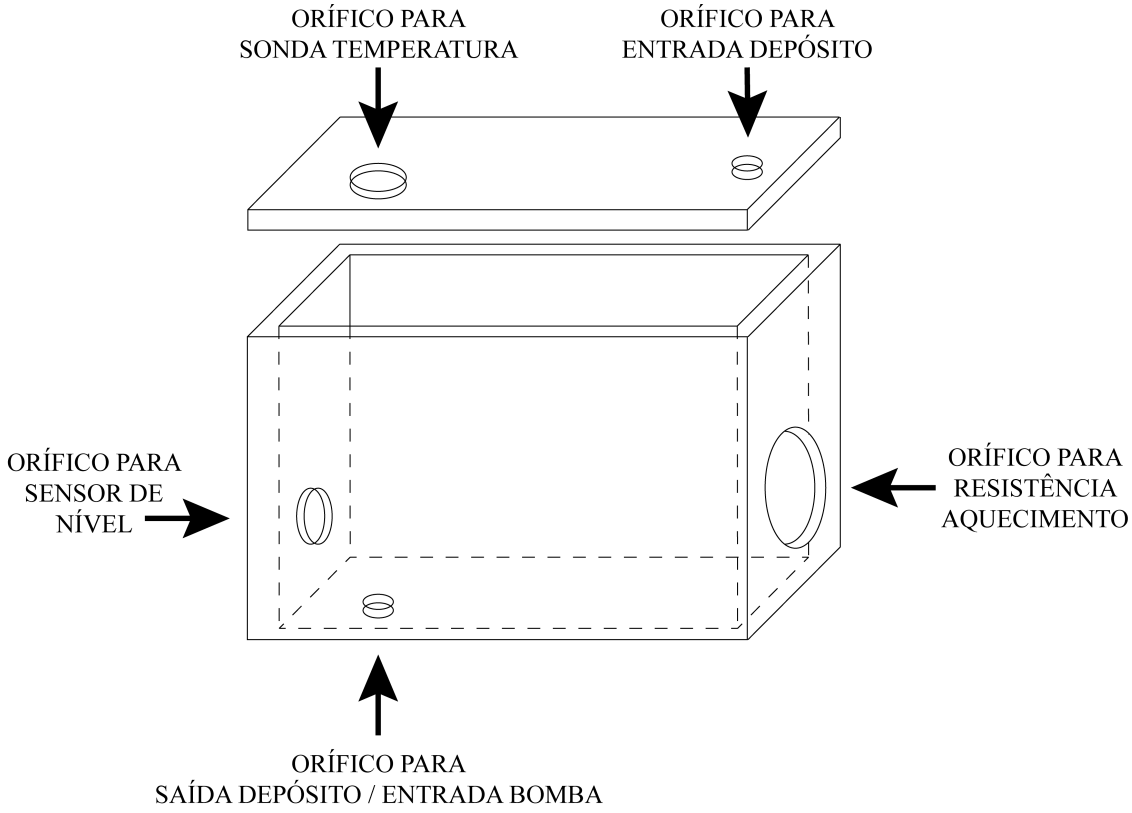
A.2 Modulo 3D do perfusor







A.3 Modelo 3D do deposito de aquecimento



A.4 Código Python

A.4.1 Código principal

Listagem A.1: Código principal

```
1  # -*- coding: utf-8 -*-
2  ################
3  ## Main file ##
4  ################
5
6  import sys, os, datetime, time
7  from PyQt4 import QtGui, QtCore
8  from threading import Thread
9  import thread
10 from multiprocessing import Process
11
12 sys.path.append('/home/pi/Desktop/')
13 from main_ui import Ui_main_ui
14 from data_ui import Ui_data_ui
15 from temp import *
16 from press import *
17 from pulse import *
18 from pid import *
19
20 global gender, arrival_date, observations, on, auto_mode, ID, heater,
       read_status
21 gender = "Male"          #Male by default
22 on = False               #System start at off state
23 auto_mode = False        #Manual mode by default
24 arrival_date = "␣"
25 observations = "␣"
26 ID = "␣"
27 heater = False           #Inside reservatory isn't use by default
28 global temperature, pressure1, pressure2, flow
29 #Data arrays
30 global x, yt, yp1, yp2, yf, t
31 x = []
32 yt = []
33 yp1 = []
34 yp2 = []
35 yf = []
36 t = 0
37
38 class Window(QtGui.QMainWindow, Ui_main_ui):
39
```

```
40      def __init__(self, parent=None):
41
42          QtGui.QMainWindow.__init__(self, parent)
43          self.setupUi(self)
44          self.setGeometry(0, 0, 800, 480)
45          QtGui.QApplication.setStyle(QtGui.QStyleFactory.create("cleanlooks"))
46          self.power_button.setText("On")
47          self.power_button.setStyleSheet(QtCore.QString.fromUtf8("background-
        color: rgb(0, 255, 0);"))
48
49          self.choice_graph_1.addItem("Pressure 1")
50          self.choice_graph_1.addItem("pressure 2")
51          self.choice_graph_1.addItem("Temperature")
52          self.choice_graph_1.addItem("Flow")
53
54          self.choice_graph_2.addItem("Pressure 2")
55          self.choice_graph_2.addItem("Flow")
56          self.choice_graph_2.addItem("Temperature")
57          self.choice_graph_2.addItem("Pressure 1")
58
59          self.target_temp.setText("0")
60          self.flow_rate.setText("0")
61          self.flow_rate_1.setText("0")
62          self.flow_rate_2.setText("0")
63          self.flow_rate_3.setText("0")
64          self.target_temp_1.setText("0")
65          self.target_temp_2.setText("0")
66          self.target_temp_3.setText("0")
67
68          self.flow_rate_1.setEnabled(False)
69          self.flow_rate_2.setEnabled(False)
70          self.flow_rate_3.setEnabled(False)
71          self.target_temp_1.setEnabled(False)
72          self.target_temp_2.setEnabled(False)
73          self.target_temp_3.setEnabled(False)
74          self.time_1.setEnabled(False)
75          self.time_2.setEnabled(False)
76          self.time_3.setEnabled(False)
77          self.target_temp.setEnabled(False)
78          self.real_temp.setStyleSheet(QtCore.QString.fromUtf8("background-color:
        rgb(195, 195, 195);\n""))
79
80
81          self.shutdown_button.clicked.connect(self.shutdown)
82          self.new_specimen_button.clicked.connect(self.dataopen)
```

```
83          self.save_button.clicked.connect(self.file_save)
84          self.manual_mode.toggle()
85          self.manual_mode.stateChanged.connect(self.check_automatic)
86          self.automatic_mode.stateChanged.connect(self.check_manual)
87          self.power_button.clicked.connect(self.on_off)
88          self.choice_graph_1.activated[str].connect(self.change_graph1)
89          self.choice_graph_2.activated[str].connect(self.change_graph2)
90          self.clear_data_button.clicked.connect(self.clear_data)
91          self.add_flow_plus.clicked.connect(self.add_flow)
92          self.add_flow_minus.clicked.connect(self.add_flow)
93          self.keyboard_button.clicked.connect(keyboard)
94          self.heater.stateChanged.connect(self.heater_status)
95
96      def heater_status(self):
97
98          global heater
99
100         if self.heater.isChecked():
101             heater = True
102             change_valve(heater)
103             thread_3 = Process(target=self.level_status())
104             thread_3.start()
105             self.real_temp.setStyleSheet(QtCore.QString.fromUtf8("background-
    color:␣rgb(0,␣0,␣0);\n"""))
106
107             if self.manual_mode.isChecked():
108                 self.target_temp.setEnabled(True)
109
110             elif self.automatic_mode.isChecked():
111                 self.target_temp_1.setEnabled(True)
112                 self.target_temp_2.setEnabled(True)
113                 self.target_temp_3.setEnabled(True)
114
115         elif not self.heater.isChecked():
116             heater = False
117             change_valve(heater)
118             self.level_indicator.setStyleSheet(QtCore.QString.fromUtf8("
    background-color:␣rgb(195,␣195,␣195);\n""font:␣25␣16pt␣\"Roboto\";\n""color:
    ␣rgb(195,␣195,␣195);"))
119             self.real_temp.setStyleSheet(QtCore.QString.fromUtf8("background-
    color:␣rgb(195,␣195,␣195);\n"""))
120             self.target_temp_1.setEnabled(False)
121             self.target_temp_2.setEnabled(False)
122             self.target_temp_3.setEnabled(False)
123             self.target_temp.setEnabled(False)
```

```
124
125     def level_status(self):
126
127         level = check_level()
128
129         if level==0:
130             self.level_indicator.setText("␣␣␣Fluid␣Level\n␣␣␣␣␣␣␣␣Low")
131             self.level_indicator.setStyleSheet(QtCore.QString.fromUtf8("
    background-color:␣rgb(255,␣0,␣0);\n""font:␣25␣16pt␣\"Roboto\";\n""color:␣rgb
    (0,␣255,␣0);"))
132             QtGui.qApp.processEvents()
133             QtGui.qApp.processEvents()
134
135         else:
136             self.level_indicator.setText("␣␣␣Fluid␣Level\n␣␣␣␣␣␣␣␣High")
137             self.level_indicator.setStyleSheet(QtCore.QString.fromUtf8("
    background-color:␣rgb(0,␣255,␣0);\n""font:␣25␣16pt␣\"Roboto\";\n""color:␣rgb
    (255,␣0,␣0);"))
138             QtGui.qApp.processEvents()
139             QtGui.qApp.processEvents()
140
141         while heater:
142
143             if check_level() != level:
144                 level = check_level()
145
146                 if level==1:
147                     self.level_indicator.setText("␣␣␣Fluid␣Level\n␣␣␣␣␣␣␣␣High")
148                     self.level_indicator.setStyleSheet(QtCore.QString.fromUtf8("
    background-color:␣rgb(0,␣255,␣0);\n""font:␣25␣16pt␣\"Roboto\";\n""color:␣rgb
    (255,␣0,␣0);"))
149                     QtGui.qApp.processEvents()
150                     QtGui.qApp.processEvents()
151
152                 elif level==0:
153                     stop_all_channels()
154                     self.level_indicator.setText("␣␣␣Fluid␣Level\n␣␣␣␣␣␣␣␣Low")
155                     self.level_indicator.setStyleSheet(QtCore.QString.fromUtf8("
    background-color:␣rgb(255,␣0,␣0);\n""font:␣25␣16pt␣\"Roboto\";\n""color:␣rgb
    (0,␣255,␣0);"))
156                     QtGui.qApp.processEvents()
157                     QtGui.qApp.processEvents()
158
159             else:
160                 pass
```

```
161
162             time.sleep(.1)
163
164         QtGui.qApp.processEvents()
165         QtGui.qApp.processEvents()
166
167     def add_flow(self):
168
169         if self.flow_rate.toPlainText() == "" or self.flow_rate.toPlainText()[0]
     == "␣":
170             old = 0
171         else:
172             old = float(self.flow_rate.toPlainText())
173
174         button = self.sender()
175
176         if button == self.add_flow_minus and old >= 0.5:
177             new = old - 0.5
178             self.flow_rate.setText(str('%.1f' % new))
179         elif button == self.add_flow_plus and old <= 14.5:
180             new = old + 0.5
181             self.flow_rate.setText(str('%.1f' % new))
182
183     def clear_data(self):
184
185         global x, yt, yp1, yp2, yf, t
186         choice = QtGui.QMessageBox.question(self, 'Clear␣All␣Data', "Are␣you␣
        sure␣you␣want␣to␣delete␣all␣data?", QtGui.QMessageBox.Yes |QtGui.QMessageBox.
        No, QtGui.QMessageBox.No)
187
188         if choice == QtGui.QMessageBox.Yes:
189             x = []
190             yt = []
191             yp1 = []
192             yp2 = []
193             yf = []
194             t = 0
195         else:
196             pass
197
198     def change_graph1(self, text):
199
200         if text == 'Temperature':
201             try:
202                 self.graphicsView_1.plot(x,yt, clear=True)
```

```
203             except:
204                 pass
205 
206         elif text == "Flow":
207             try:
208                 self.graphicsView_1.plot(x,yf, clear=True)
209             except:
210                 pass
211 
212         elif text == "Pressure 1":
213             try:
214                 self.graphicsView_1.plot(x,yp1, clear=True)
215             except:
216                 pass
217 
218         elif text == "Pressure 2":
219             try:
220                 self.graphicsView_1.plot(x,yp2, clear=True)
221             except:
222                 pass
223 
224         else:
225             pass
226 
227         QtGui.qApp.processEvents()
228         QtGui.qApp.processEvents()
229 
230     def change_graph2(self, text):
231 
232         if text == 'Temperature':
233             try:
234                 self.graphicsView_2.plot(x,yt, clear=True)
235             except:
236                 pass
237 
238         elif text == "Flow":
239             try:
240                 self.graphicsView_2.plot(x,yf, clear=True)
241             except:
242                 pass
243 
244         elif text == "Pressure 1":
245             try:
246                 self.graphicsView_2.plot(x,yp1, clear=True)
247             except:
```

```
248                 pass
249 
250         elif text == "Pressure␣2":
251             try:
252                 self.graphicsView_2.plot(x,yp2, clear=True)
253             except:
254                 pass
255 
256         else:
257             pass
258 
259         QtGui.qApp.processEvents()
260         QtGui.qApp.processEvents()
261 
262     def on_off(self):
263 
264         global on
265         stop_all_channels()
266 
267         if on:
268             on = False
269             self.power_button.setText("On")
270             self.power_button.setStyleSheet(QtCore.QString.fromUtf8("background-
    color:␣rgb(0,␣255,␣0);"))
271             self.clear_data_button.setEnabled(True)
272 
273         else:
274             on = True
275             self.power_button.setText("Off")
276             self.power_button.setStyleSheet(QtCore.QString.fromUtf8("background-
    color:␣rgb(255,␣0,␣0);"))
277             self.clear_data_button.setEnabled(False)
278             QtGui.qApp.processEvents()
279             thread_2 = Thread(target = self.cycle(), args = ())
280             thread_2.start()
281 
282     def shutdown(self):
283 
284         choice = QtGui.QMessageBox.question(self, 'Shutdown!', "Are␣you␣sure␣you
    ␣want␣to␣shutdown␣the␣system?\nAny␣unsaved␣data␣will␣be␣lost.", QtGui.
    QMessageBox.Yes | QtGui.QMessageBox.No, QtGui.QMessageBox.No)
285 
286         if choice == QtGui.QMessageBox.Yes:
287             os.system('sudo␣shutdown␣now')
288         else:
```

```
289             pass
290
291     def dataopen(self):
292
293         DATA = Text(self)
294         thread_1 = Thread(target = DATA.show(), args = ())
295         thread_1.start()
296
297     def file_save(self):
298
299         initialPath = str(ID) + '.txt'
300         name = QtGui.QFileDialog.getSaveFileName(self, 'Save File', initialPath)
301         savefile(name)
302
303     def check_automatic(self):
304
305         global auto_mode
306
307         if self.manual_mode.isChecked() == self.automatic_mode.isChecked():
308             self.automatic_mode.toggle()
309         else:
310             pass
311
312         if self.automatic_mode.isChecked():
313             auto_mode = True
314             self.target_temp.setEnabled(False)
315             self.flow_rate.setEnabled(False)
316             self.flow_rate_1.setEnabled(True)
317             self.flow_rate_2.setEnabled(True)
318             self.flow_rate_3.setEnabled(True)
319             self.time_1.setEnabled(True)
320             self.time_2.setEnabled(True)
321             self.time_3.setEnabled(True)
322
323             if heater:
324                 self.target_temp_1.setEnabled(True)
325                 self.target_temp_2.setEnabled(True)
326                 self.target_temp_3.setEnabled(True)
327             else:
328                 self.target_temp_1.setEnabled(False)
329                 self.target_temp_2.setEnabled(False)
330                 self.target_temp_3.setEnabled(False)
331
332     def check_manual(self):
333
```

```
334         global auto_mode
335 
336         if self.manual_mode.isChecked() == self.automatic_mode.isChecked():
337             self.manual_mode.toggle()
338         else:
339             pass
340 
341         if self.manual_mode.isChecked():
342             auto_mode = False
343             self.flow_rate.setEnabled(True)
344             self.flow_rate_1.setEnabled(False)
345             self.flow_rate_2.setEnabled(False)
346             self.flow_rate_3.setEnabled(False)
347             self.target_temp_1.setEnabled(False)
348             self.target_temp_2.setEnabled(False)
349             self.target_temp_3.setEnabled(False)
350             self.time_1.setEnabled(False)
351             self.time_2.setEnabled(False)
352             self.time_3.setEnabled(False)
353 
354             if heater:
355                 self.target_temp.setEnabled(True)
356             else:
357                 self.target_temp.setEnabled(False)
358 
359     def read_data(self, n):
360 
361         global temperature, pressure1, pressure2, pressure3, flow
362 
363         if n == 4:
364             temperature = read_temp()
365             yt.append(temperature)
366             self.real_temp.display(temperature)
367         elif n == 1:
368             pressure1 = read_press(1)
369             yp1.append(pressure1)
370             self.real_press_1.display(pressure1)
371         elif n == 2:
372             pressure2 = read_press(2)
373             yp2.append(pressure2)
374             self.real_press_2.display(pressure2)
375         elif n == 3:
376             flow = float(self.flow_rate.toPlainText())
377             yf.append(flow)
378             self.real_flow_rate.display(flow)
```

```
379         else:
380             pass
381 
382     def cycle(self):
383 
384         self.automatic_mode.setEnabled(False)
385         self.manual_mode.setEnabled(False)
386         self.heater.setEnabled(False)
387 
388         global t
389 
390         if auto_mode:
391             time_1 = self.time_1.time().toPyTime()
392             time_2 = self.time_2.time().toPyTime()
393             time_3 = self.time_3.time().toPyTime()
394 
395             time_start = time.time()
396 
397         if heater:
398 
399             while on:
400 
401                 start = time.time()
402 
403                 threads = []
404 
405                 x.append(t)
406 
407                 for n in range(1, 5):
408 
409                     thread = Thread(target=self.read_data, args=(n,))
410                     thread.start()
411 
412                     threads.append(thread)
413 
414                 t += 2
415 
416                 if not auto_mode:
417                     target_0 = float(self.target_temp.toPlainText())
418                     flow_0 = float(self.flow_rate.toPlainText())
419 
420                     if flow_0 > 15:
421                         flow_0 = 15
422                         self.flow_rate.setText("15")
423                     elif flow_0 < 0:
```

```
424                         flow_0 = 0
425                         self.flow_rate.setText("0")
426 
427                 if auto_mode:
428                     target_1 = float(self.target_temp_1.toPlainText())
429                     target_2 = float(self.target_temp_2.toPlainText())
430                     target_3 = float(self.target_temp_3.toPlainText())
431                     flow_1 = float(self.flow_rate_1.toPlainText())
432                     flow_2 = float(self.flow_rate_2.toPlainText())
433                     flow_3 = float(self.flow_rate_3.toPlainText())
434 
435                     if flow_1 > 15:
436                         flow_1 = 15
437                         self.flow_rate_1.setText("15")
438                     elif flow_1 < 0:
439                         flow_1 = 0
440                         self.flow_rate_1.setText("0")
441 
442                     if flow_2 > 15:
443                         flow_2 = 15
444                         self.flow_rate_2.setText("15")
445                     elif flow_2 < 0:
446                         flow_2 = 0
447                         self.flow_rate_2.setText("0")
448 
449                     if flow_3 > 15:
450                         flow_3 = 15
451                         self.flow_rate_3.setText("15")
452                     elif flow_3 < 0:
453                         flow_3 = 0
454                         self.flow_rate_3.setText("0")
455 
456                 for thread in threads:
457 
458                     thread.join()
459 
460                 if not auto_mode:
461 
462                     start_pulse(0, 11, dutycicle(target_0, temperature))
463 
464                     if len(yf) < 2:
465                         start_pulse(flow_0, 12, 0)
466                     elif yf[len(yf)-1] != yf[len(yf)-2]:
467                         start_pulse(flow_0, 12, 0)
468 
```

```
469                 elif auto_mode:
470
471                     if time.time() - time_start < seconds(self.time_1.time().
    toPyTime()):
472                         stop_all_channels()
473
474                     elif time.time() - time_start > seconds(self.time_1.time().
    toPyTime()) and time.time() - time_start < seconds(self.time_2.time().
    toPyTime()):
475                         start_pulse(0, 11, dutycicle(target_1, temperature))
476                         start_pulse(flow_1, 12, 0)
477
478                     elif time.time() - time_start > seconds(self.time_2.time().
    toPyTime()) and time.time() - time_start < seconds(self.time_3.time().
    toPyTime()):
479                         start_pulse(0, 11, dutycicle(target_2, temperature))
480                         start_pulse(flow_2, 12, 0)
481
482                     elif time.time() - time_start > seconds(self.time_3.time().
    toPyTime()):
483                         start_pulse(0, 11, dutycicle(target_3, temperature))
484                         start_pulse(flow_3, 12, 0)
485
486                 self.change_graph1(self.choice_graph_1.currentText())
487                 self.change_graph2(self.choice_graph_2.currentText())
488                 QtGui.qApp.processEvents()
489                 QtGui.qApp.processEvents()
490
491                 elapsed = (time.time() - start)
492                 try:
493                     time.sleep(2.0 - elapsed)
494                 except:
495                     if elapsed > 2:
496                         t += (-2 + elapsed)
497                     else:
498                         pass
499
500         elif not heater:
501
502             while on:
503
504                 start = time.time()
505
506                 threads = []
507
```

```
508                 x.append(t)
509
510                 for n in range(1, 3):
511                     thread = Thread(target=self.read_data, args=(n,))
512                     thread.start()
513
514                     threads.append(thread)
515
516                 t += 2
517
518                 if not auto_mode:
519                     flow_0 = float(self.flow_rate.toPlainText())
520
521                     if flow_0 > 15:
522                         flow_0 = 15
523                         self.flow_rate.setText("15")
524                     elif flow_0 < 0:
525                         flow_0 = 0
526                         self.flow_rate.setText("0")
527
528                 if auto_mode:
529                     flow_1 = float(self.flow_rate_1.toPlainText())
530                     flow_2 = float(self.flow_rate_2.toPlainText())
531                     flow_3 = float(self.flow_rate_3.toPlainText())
532
533                     if flow_1 > 15:
534                         flow_1 = 15
535                         self.flow_rate_1.setText("15")
536                     elif flow_1 < 0:
537                         flow_1 = 0
538                         self.flow_rate_1.setText("0")
539
540                     if flow_2 > 15:
541                         flow_2 = 15
542                         self.flow_rate_2.setText("15")
543                     elif flow_2 < 0:
544                         flow_2 = 0
545                         self.flow_rate_2.setText("0")
546
547                     if flow_3 > 15:
548                         flow_3 = 15
549                         self.flow_rate_3.setText("15")
550                     elif flow_3 < 0:
551                         flow_3 = 0
552                         self.flow_rate_3.setText("0")
```

```
553
554                 for thread in threads:
555                     thread.join()
556
557                 if not auto_mode:
558
559                     if len(yf) < 2:
560                         start_pulse(flow_0, 12, 0)
561                     elif yf[len(yf)-1] != yf[len(yf)-2]:
562                         start_pulse(flow_0, 12, 0)
563
564                 elif auto_mode:
565
566                     if time.time() - time_start < seconds(self.time_1.time().
    toPyTime()):
567                         stop_all_channels()
568                     elif time.time() - time_start > seconds(self.time_1.time().
    toPyTime()) and time.time() - time_start < seconds(self.time_2.time().
    toPyTime()):
569                         start_pulse(flow_1, 12, 0)
570                     elif time.time() - time_start > seconds(self.time_2.time().
    toPyTime()) and time.time() - time_start < seconds(self.time_3.time().
    toPyTime()):
571                         start_pulse(flow_2, 12, 0)
572                     elif time.time() - time_start > seconds(self.time_3.time().
    toPyTime()):
573                         start_pulse(flow_3, 12, 0)
574
575                 self.change_graph1(self.choice_graph_1.currentText())
576                 self.change_graph2(self.choice_graph_2.currentText())
577                 QtGui.qApp.processEvents()
578                 QtGui.qApp.processEvents()
579
580                 elapsed = (time.time() - start)
581                 try:
582                     time.sleep(2.0 - elapsed)
583                 except:
584                     if elapsed > 2:
585                         t += (-2 + elapsed)
586                     else:
587                         pass
588
589         self.automatic_mode.setEnabled(True)
590         self.manual_mode.setEnabled(True)
591         self.heater.setEnabled(True)
```

```
592
593 class Text(QtGui.QMainWindow, Ui_data_ui):
594
595     def __init__(self, parent=None):
596         QtGui.QMainWindow.__init__(self, parent)
597         self.setupUi(self)
598         self.setGeometry(0, 0, 800, 480)
599         QtGui.QApplication.setStyle(QtGui.QStyleFactory.create("cleanlooks"))
600
601         self.create_specimen_button.clicked.connect(self.closedata)
602         self.male_check.toggle()
603         self.male_check.stateChanged.connect(self.check_female)
604         self.female_check.stateChanged.connect(self.check_male)
605         self.arrival_date.setDate(datetime.date.today())
606         self.keyboard_button.clicked.connect(keyboard)
607
608     def check_male(self):
609
610         if self.male_check.isChecked() == self.female_check.isChecked():
611             self.male_check.toggle()
612         else:
613             pass
614
615     def check_female(self):
616
617         if self.male_check.isChecked() == self.female_check.isChecked():
618             self.female_check.toggle()
619         else:
620             pass
621
622     def closedata(self):
623
624         global gender, arrival_date, observations, ID
625
626         if self.female_check.isChecked():
627             gender = "Female"
628         else :
629             gender = "Male"
630
631         arrival_date = self.arrival_date.date().toPyDate()
632
633         observations =  self.observations.toPlainText()
634
635         ID = self.id.toPlainText()
636
```

```
637         self.hide()
638 
639 def savefile(name):
640 
641     f = open(name, "w")
642     f.write("Specimen ID: %s\n" % ID)
643     f.write("Specimen Arrival Date: %s Specimen gender: %s\n" %(arrival_date,
        gender))
644     f.write("Specimen Observations:\n%s\n" %(observations))
645     f.write("Date: %s\n" %(datetime.date.today()))
646     f.write("Time: %s\n" %(datetime.datetime.now().time()))
647     f.write("time(s) flow(LPH) temperature(£C) pressure1(mmHg) pressure2(mmHg)
        pressure3(mmHg)\n")
648     n = 0
649 
650     while n < len(x):
651 
652         f.write("%f %f %f %f %f %f\n" %(x[n], yf[n], yt[n], yp1[n], yp2[n], yp3[
        n]))
653         n += 1
654 
655     f.close()
656 
657 def seconds(time):
658 
659     s = time.hour * 3600 + time.minute * 60
660     return s
661 
662 def keyboard():
663 
664     os.system('killall florence&')
665     os.system('florence&')
666     os.system('florence move 0,0')
667 
668 app = QtGui.QApplication(sys.argv)
669 GUI = Window()
670 #Start the ui
671 GUI.show()
672 
673 sys.exit(app.exec_())
```

A.4.1.1 Módulos personalizados

Listagem A.2: Módulo de nível

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(16, GPIO.IN)

def check_level(self):

    level = 0

    try:
        level = GPIO.input(16)
        return level
    except:
        return level
```

Listagem A.3: Módulo de PID

```
########################################
## File for the PID value calculation ##
########################################

global B, P, I, inter, error, power
inter = 0
#The values can be manually ajusted here
B = 22
P = 6
I = 2

#Return the resistor up time needed in %
def dutycicle(target, temperature):

    global error, inter, power
    error = (target - temperature)*1000
    inter = inter + error
    power = B + ((P * error) + (I * inter)/100)/100
    return power
```

Listagem A.4: Módulo de pressão

```
# -*- coding: utf-8 -*-
#include <OneWire.h>
############################
## Pressure sensors file ##
############################

import Adafruit_ADS1x15 as ADS
```

```
8
9  GAIN = 1
10
11 adc = ADS.ADS1015()
12
13 #Return the diferencial pressure on the nth sensor
14 def read_press(n):
15
16     if n == 1:
17         raw = adc.read_adc(0, gain=GAIN) - adc.read_adc(1, gain=GAIN)
18         if raw != 0:
19             pressure = (32767/raw)*4.095 #recta de calibraÅÅo
20             return pressure
21         else:
22             pressure = 0
23             return pressure
24
25     elif n == 2:
26         raw = adc.read_adc(2, gain=GAIN) - adc.read_adc(3, gain=GAIN)
27         if raw != 0:
28             pressure = (32767/raw)*4.095 #recta de calibraÅÅo
29             return pressure
30         else:
31             pressure = 0
32             return pressure
```

Listagem A.5: Módulo de sinal

```
1  ########################
2  ## GPIO control file ##
3  ########################
4
5  import RPi.GPIO as GPIO
6
7  GPIO.setwarnings(False)
8
9  #GPIO setup
10 GPIO.setmode(GPIO.BOARD)
11 GPIO.setup(11, GPIO.OUT)
12 GPIO.setup(12, GPIO.OUT)
13 GPIO.setup(16, GPIO.IN)
14 GPIO.setup(18, GPIO.OUT)
15 GPIO.setup(22, GPIO.OUT)
16
17 #By default the inside reservatory isn't use
18 GPIO.output(18, 1) #Open the valve directly to the pump
```

```
19 GPIO.output(22, 0) #Close the valve to the inside reservatory
20
21 global pump, resistor, resistor_status
22 pump = GPIO.PWM(12, 1)
23 resistor = GPIO.PWM(11, 1)
24 pump.start(0)
25 resistor.start(0)
26
27 #Start or change the pump's/resistor's control
28 def start_pulse(speed, channel, duty):
29
30     global pump
31     global resistor
32
33     if channel == 12:
34         if speed > 0:
35             frq = speed * 0.1309
36             DC = 7.5 * frq
37             pump.ChangeFrequency(frq)
38             pump.ChangeDutyCycle(DC)
39         else:
40             pump.ChangeDutyCycle(0)
41
42     if channel == 11:
43         resistor.ChangeDutyCycle(_dutycicle(duty))
44
45     else:
46         pass
47
48 #Fix the value for the dutycycle between 0 and 100
49 def _dutycicle(duty):
50
51     if duty > 0 and duty < 100:
52         return duty
53     elif duty >= 100:
54         return 100
55     else:
56         return 0
57
58 #Stop the pump and the resistor
59 def stop_all_channels():
60
61     pump.ChangeDutyCycle(0)
62     resistor.ChangeDutyCycle(0)
63
```

```
64 #Stop the pump or the resistor
65 def stop_channel(channel):
66 
67     if channel == 11:
68         resistor.ChangeDutyCycle(0)
69     elif channel == 12:
70         pump.ChangeDutyCycle(0)
71     else:
72         pass
73 
74 #Open and close or Close and open the valve directly
75 #to the pump or to the inside reservatory respectively
76 def change_valve(n):
77 
78     if not n:
79         GPIO.output(18, 1)
80         GPIO.output(22, 0)
81     elif n:
82         GPIO.output(18, 0)
83         GPIO.output(22, 1)
84     else:
85         pass
86 
87 #Return 0 or 1 for low or high fluid level respectively
88 def check_level():
89 
90     return int(GPIO.input(16))
```

Listagem A.6: Módulo de temperatura

```
1  # -*- coding: utf-8 -*-
2  #include <OneWire.h>
3  ##############################
4  ## Temperature sensor file ##
5  ##############################
6 
7  import os
8  import time
9 
10 os.system('modprobe␣w1-gpio')
11 os.system('modprobe␣w1-therm')
12 
13 #Sensor's directory
14 temp_sensor = '/sys/bus/w1/devices/28-000006157784/w1_slave'
15 
16 #Read the sensor and returns its output
```

```
17 #In case of file not found return t=00 to avoid error
18 def _temp_raw():
19
20     try:
21         f=open(temp_sensor, 'r')
22         lines = f.readlines()
23         f.close()
24     except:
25         lines = ['YES','t=00']
26     return lines
27
28 #Extract the temperature from the sensor's output and return it in ºC
29 def read_temp():
30
31     lines = _temp_raw()
32     while lines[0].strip()[-3:] != 'YES' :
33         time.sleep(0.2)
34         lines = _temp_raw()
35
36     temp_output = lines[1].find('t=')
37
38     if temp_output != -1:
39         temp_string = lines[1].strip()[temp_output+2:]
40         temp = float(temp_string)/1000.0
41         return temp
42     else:
43         return 0
```

A.4.2 Interface gráfica

A.4.2.1 Interface gráfica principal

Listagem A.7: Código para a interface gráfica principal

```
1  # -*- coding: utf-8 -*-
2  ########################################
3  ## Setup file for the primary ui ##
4  ########################################
5
6  from PyQt4 import QtCore, QtGui
7
8  try:
9      _fromUtf8 = QtCore.QString.fromUtf8
10 except AttributeError:
11     def _fromUtf8(s):
```

```
12          return s
13
14  try:
15      _encoding = QtGui.QApplication.UnicodeUTF8
16      def _translate(context, text, disambig):
17          return QtGui.QApplication.translate(context, text, disambig, _encoding)
18  except AttributeError:
19      def _translate(context, text, disambig):
20          return QtGui.QApplication.translate(context, text, disambig)
21
22  class Ui_main_ui(object):
23      def setupUi(self, main_ui):
24          main_ui.setObjectName(_fromUtf8("main_ui"))
25          main_ui.resize(801, 473)
26          main_ui.setStyleSheet(_fromUtf8("background-color:␣rgb(195,␣195,␣195);")
        )
27
28          self.shutdown_button = QtGui.QPushButton(main_ui)
29          self.shutdown_button.setGeometry(QtCore.QRect(10, 10, 89, 29))
30          self.shutdown_button.setObjectName(_fromUtf8("shutdown_button"))
31
32          self.power_button = QtGui.QPushButton(main_ui)
33          self.power_button.setGeometry(QtCore.QRect(690, 10, 101, 29))
34          self.power_button.setObjectName(_fromUtf8("power_button"))
35
36          self.clear_data_button = QtGui.QPushButton(main_ui)
37          self.clear_data_button.setGeometry(QtCore.QRect(140, 10, 131, 29))
38          self.clear_data_button.setObjectName(_fromUtf8("clear_data_button"))
39
40          self.save_button = QtGui.QPushButton(main_ui)
41          self.save_button.setGeometry(QtCore.QRect(280, 10, 81, 29))
42          self.save_button.setObjectName(_fromUtf8("save_button"))
43
44          self.manual_mode = QtGui.QCheckBox(main_ui)
45          self.manual_mode.setGeometry(QtCore.QRect(500, 40, 131, 24))
46          self.manual_mode.setObjectName(_fromUtf8("manual_mode"))
47
48          self.label = QtGui.QLabel(main_ui)
49          self.label.setGeometry(QtCore.QRect(740, 110, 21, 19))
50          self.label.setObjectName(_fromUtf8("label"))
51
52          self.label_2 = QtGui.QLabel(main_ui)
53          self.label_2.setGeometry(QtCore.QRect(600, 100, 41, 19))
54          self.label_2.setObjectName(_fromUtf8("label_2"))
55
```

```
56          self.target_temp = QtGui.QTextEdit(main_ui)
57          self.target_temp.setGeometry(QtCore.QRect(640, 100, 91, 31))
58          self.target_temp.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
59          self.target_temp.setTextInteractionFlags(QtCore.Qt.TextEditorInteraction
        )
60          self.target_temp.setObjectName(_fromUtf8("target_temp"))
61
62          self.label_3 = QtGui.QLabel(main_ui)
63          self.label_3.setGeometry(QtCore.QRect(500, 80, 81, 19))
64          self.label_3.setObjectName(_fromUtf8("label_3"))
65
66          self.flow_rate = QtGui.QTextEdit(main_ui)
67          self.flow_rate.setGeometry(QtCore.QRect(518, 100, 55, 31))
68          self.flow_rate.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣255,␣
        255);\n""font:␣25␣10pt␣\"Roboto\";"))
69          self.flow_rate.setObjectName(_fromUtf8("flow_rate"))
70
71          self.line = QtGui.QFrame(main_ui)
72          self.line.setGeometry(QtCore.QRect(500, 60, 291, 20))
73          self.line.setStyleSheet(_fromUtf8(""))
74          self.line.setFrameShape(QtGui.QFrame.HLine)
75          self.line.setFrameShadow(QtGui.QFrame.Sunken)
76          self.line.setObjectName(_fromUtf8("line"))
77
78          self.label_4 = QtGui.QLabel(main_ui)
79          self.label_4.setGeometry(QtCore.QRect(640, 80, 151, 19))
80          self.label_4.setObjectName(_fromUtf8("label_4"))
81
82          self.graphicsView_1 = PlotWidget(main_ui)
83          self.graphicsView_1.setGeometry(QtCore.QRect(10, 70, 481, 121))
84          self.graphicsView_1.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);"))
85          self.graphicsView_1.setObjectName(_fromUtf8("graphicsView_1"))
86
87          self.automatic_mode = QtGui.QCheckBox(main_ui)
88          self.automatic_mode.setGeometry(QtCore.QRect(500, 130, 161, 24))
89          self.automatic_mode.setObjectName(_fromUtf8("automatic_mode"))
90
91          self.line_2 = QtGui.QFrame(main_ui)
92          self.line_2.setGeometry(QtCore.QRect(500, 150, 291, 20))
93          self.line_2.setFrameShape(QtGui.QFrame.HLine)
94          self.line_2.setFrameShadow(QtGui.QFrame.Sunken)
95          self.line_2.setObjectName(_fromUtf8("line_2"))
96
```

```
97          self.label_9 = QtGui.QLabel(main_ui)
98          self.label_9.setGeometry(QtCore.QRect(660, 170, 101, 20))
99          self.label_9.setObjectName(_fromUtf8("label_9"))
100
101         self.time_1 = QtGui.QTimeEdit(main_ui)
102         self.time_1.setGeometry(QtCore.QRect(650, 190, 121, 31))
103         self.time_1.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣255,␣
        255);"))
104         self.time_1.setObjectName(_fromUtf8("time_1"))
105
106         self.label_5 = QtGui.QLabel(main_ui)
107         self.label_5.setGeometry(QtCore.QRect(600, 190, 41, 19))
108         self.label_5.setObjectName(_fromUtf8("label_5"))
109
110         self.label_14 = QtGui.QLabel(main_ui)
111         self.label_14.setGeometry(QtCore.QRect(500, 270, 91, 20))
112         self.label_14.setObjectName(_fromUtf8("label_14"))
113
114         self.label_6 = QtGui.QLabel(main_ui)
115         self.label_6.setGeometry(QtCore.QRect(500, 170, 91, 20))
116         self.label_6.setObjectName(_fromUtf8("label_6"))
117
118         self.flow_rate_1 = QtGui.QTextEdit(main_ui)
119         self.flow_rate_1.setGeometry(QtCore.QRect(500, 190, 91, 31))
120         self.flow_rate_1.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
121         self.flow_rate_1.setObjectName(_fromUtf8("flow_rate_1"))
122
123         self.label_8 = QtGui.QLabel(main_ui)
124         self.label_8.setGeometry(QtCore.QRect(600, 250, 21, 19))
125         self.label_8.setObjectName(_fromUtf8("label_8"))
126
127         self.target_temp_1 = QtGui.QTextEdit(main_ui)
128         self.target_temp_1.setGeometry(QtCore.QRect(500, 240, 91, 31))
129         self.target_temp_1.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
130         self.target_temp_1.setObjectName(_fromUtf8("target_temp_1"))
131
132         self.label_7 = QtGui.QLabel(main_ui)
133         self.label_7.setGeometry(QtCore.QRect(500, 220, 171, 20))
134         self.label_7.setObjectName(_fromUtf8("label_7"))
135
136         self.label_11 = QtGui.QLabel(main_ui)
137         self.label_11.setGeometry(QtCore.QRect(660, 270, 101, 20))
138         self.label_11.setObjectName(_fromUtf8("label_11"))
```

```
139
140         self.label_16 = QtGui.QLabel(main_ui)
141         self.label_16.setGeometry(QtCore.QRect(660, 370, 101, 20))
142         self.label_16.setObjectName(_fromUtf8("label_16"))
143
144         self.time_2 = QtGui.QTimeEdit(main_ui)
145         self.time_2.setGeometry(QtCore.QRect(650, 290, 121, 31))
146         self.time_2.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣255,␣
    255);"))
147         self.time_2.setObjectName(_fromUtf8("time_2"))
148
149         self.time_3 = QtGui.QTimeEdit(main_ui)
150         self.time_3.setGeometry(QtCore.QRect(650, 390, 121, 31))
151         self.time_3.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣255,␣
    255);"))
152         self.time_3.setObjectName(_fromUtf8("time_3"))
153
154         self.target_temp_3 = QtGui.QTextEdit(main_ui)
155         self.target_temp_3.setGeometry(QtCore.QRect(500, 440, 91, 31))
156         self.target_temp_3.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
157         self.target_temp_3.setObjectName(_fromUtf8("target_temp_3"))
158
159         self.label_18 = QtGui.QLabel(main_ui)
160         self.label_18.setGeometry(QtCore.QRect(600, 450, 21, 19))
161         self.label_18.setObjectName(_fromUtf8("label_18"))
162
163         self.label_12 = QtGui.QLabel(main_ui)
164         self.label_12.setGeometry(QtCore.QRect(600, 290, 41, 19))
165         self.label_12.setObjectName(_fromUtf8("label_12"))
166
167         self.label_19 = QtGui.QLabel(main_ui)
168         self.label_19.setGeometry(QtCore.QRect(500, 370, 91, 20))
169         self.label_19.setObjectName(_fromUtf8("label_19"))
170
171         self.label_10 = QtGui.QLabel(main_ui)
172         self.label_10.setGeometry(QtCore.QRect(500, 320, 171, 20))
173         self.label_10.setObjectName(_fromUtf8("label_10"))
174
175         self.label_13 = QtGui.QLabel(main_ui)
176         self.label_13.setGeometry(QtCore.QRect(600, 350, 21, 19))
177         self.label_13.setObjectName(_fromUtf8("label_13"))
178
179         self.flow_rate_2 = QtGui.QTextEdit(main_ui)
180         self.flow_rate_2.setGeometry(QtCore.QRect(500, 290, 91, 31))
```

```
181         self.flow_rate_2.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
182         self.flow_rate_2.setObjectName(_fromUtf8("flow_rate_2"))
183 
184         self.label_15 = QtGui.QLabel(main_ui)
185         self.label_15.setGeometry(QtCore.QRect(500, 420, 171, 20))
186         self.label_15.setObjectName(_fromUtf8("label_15"))
187 
188         self.flow_rate_3 = QtGui.QTextEdit(main_ui)
189         self.flow_rate_3.setGeometry(QtCore.QRect(500, 390, 91, 31))
190         self.flow_rate_3.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
191         self.flow_rate_3.setObjectName(_fromUtf8("flow_rate_3"))
192 
193         self.target_temp_2 = QtGui.QTextEdit(main_ui)
194         self.target_temp_2.setGeometry(QtCore.QRect(500, 340, 91, 31))
195         self.target_temp_2.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""font:␣25␣10pt␣\"Roboto\";"))
196         self.target_temp_2.setObjectName(_fromUtf8("target_temp_2"))
197 
198         self.label_17 = QtGui.QLabel(main_ui)
199         self.label_17.setGeometry(QtCore.QRect(600, 390, 41, 19))
200         self.label_17.setObjectName(_fromUtf8("label_17"))
201 
202         self.real_flow_rate = QtGui.QLCDNumber(main_ui)
203         self.real_flow_rate.setGeometry(QtCore.QRect(60, 390, 64, 23))
204         self.real_flow_rate.setStyleSheet(_fromUtf8("background-color:␣rgb(0,␣0,
    ␣0);"))
205         self.real_flow_rate.setObjectName(_fromUtf8("real_flow_rate"))
206 
207         self.label_22 = QtGui.QLabel(main_ui)
208         self.label_22.setGeometry(QtCore.QRect(190, 370, 101, 19))
209         self.label_22.setObjectName(_fromUtf8("label_22"))
210 
211         self.label_24 = QtGui.QLabel(main_ui)
212         self.label_24.setGeometry(QtCore.QRect(50, 420, 91, 19))
213         self.label_24.setObjectName(_fromUtf8("label_24"))
214 
215         self.real_temp = QtGui.QLCDNumber(main_ui)
216         self.real_temp.setGeometry(QtCore.QRect(200, 390, 64, 23))
217         self.real_temp.setStyleSheet(_fromUtf8("background-color:␣rgb(0,␣0,␣0);\
    n"""))
218         self.real_temp.setObjectName(_fromUtf8("real_temp"))
219 
220         self.real_press_1 = QtGui.QLCDNumber(main_ui)
```

```
221         self.real_press_1.setGeometry(QtCore.QRect(60, 440, 64, 23))
222         self.real_press_1.setStyleSheet(_fromUtf8("background-color:␣rgb(0,␣0,␣
    0);"))
223         self.real_press_1.setObjectName(_fromUtf8("real_press_1"))
224 
225         self.real_press_2 = QtGui.QLCDNumber(main_ui)
226         self.real_press_2.setGeometry(QtCore.QRect(200, 440, 64, 23))
227         self.real_press_2.setStyleSheet(_fromUtf8("background-color:␣rgb(0,␣0,␣
    0);"))
228         self.real_press_2.setObjectName(_fromUtf8("real_press_2"))
229 
230         self.label_23 = QtGui.QLabel(main_ui)
231         self.label_23.setGeometry(QtCore.QRect(270, 390, 21, 19))
232         self.label_23.setObjectName(_fromUtf8("label_23"))
233 
234         self.label_25 = QtGui.QLabel(main_ui)
235         self.label_25.setGeometry(QtCore.QRect(130, 440, 61, 19))
236         self.label_25.setObjectName(_fromUtf8("label_25"))
237 
238         self.label_26 = QtGui.QLabel(main_ui)
239         self.label_26.setGeometry(QtCore.QRect(270, 440, 61, 19))
240         self.label_26.setObjectName(_fromUtf8("label_26"))
241 
242         self.label_21 = QtGui.QLabel(main_ui)
243         self.label_21.setGeometry(QtCore.QRect(130, 390, 41, 19))
244         self.label_21.setObjectName(_fromUtf8("label_21"))
245 
246         self.label_20 = QtGui.QLabel(main_ui)
247         self.label_20.setGeometry(QtCore.QRect(50, 370, 81, 19))
248         self.label_20.setObjectName(_fromUtf8("label_20"))
249 
250         self.line_3 = QtGui.QFrame(main_ui)
251         self.line_3.setGeometry(QtCore.QRect(30, 350, 421, 16))
252         self.line_3.setFrameShape(QtGui.QFrame.HLine)
253         self.line_3.setFrameShadow(QtGui.QFrame.Sunken)
254         self.line_3.setObjectName(_fromUtf8("line_3"))
255 
256         self.label_27 = QtGui.QLabel(main_ui)
257         self.label_27.setGeometry(QtCore.QRect(190, 420, 91, 19))
258         self.label_27.setObjectName(_fromUtf8("label_27"))
259 
260         self.graphicsView_2 = PlotWidget(main_ui)
261         self.graphicsView_2.setGeometry(QtCore.QRect(10, 220, 481, 121))
262         self.graphicsView_2.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);"))
```

```
263         self.graphicsView_2.setObjectName(_fromUtf8("graphicsView_2"))
264 
265         self.new_specimen_button = QtGui.QPushButton(main_ui)
266         self.new_specimen_button.setGeometry(QtCore.QRect(370, 10, 131, 29))
267         self.new_specimen_button.setObjectName(_fromUtf8("new_specimen_button"))
268 
269         self.choice_graph_1 = QtGui.QComboBox(main_ui)
270         self.choice_graph_1.setGeometry(QtCore.QRect(10, 50, 481, 21))
271         self.choice_graph_1.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""selection-background-color:␣rgb(0,␣0,␣0);"))
272         self.choice_graph_1.setObjectName(_fromUtf8("choice_graph_1"))
273 
274         self.choice_graph_2 = QtGui.QComboBox(main_ui)
275         self.choice_graph_2.setGeometry(QtCore.QRect(10, 200, 481, 25))
276         self.choice_graph_2.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
    255,␣255);\n""selection-background-color:␣rgb(0,␣0,␣0);"))
277         self.choice_graph_2.setObjectName(_fromUtf8("choice_graph_2"))
278 
279         self.add_flow_plus = QtGui.QPushButton(main_ui)
280         self.add_flow_plus.setGeometry(QtCore.QRect(500, 100, 16, 31))
281         self.add_flow_plus.setObjectName(_fromUtf8("add_flow_plus"))
282 
283         self.add_flow_minus = QtGui.QPushButton(main_ui)
284         self.add_flow_minus.setGeometry(QtCore.QRect(575, 100, 16, 31))
285         self.add_flow_minus.setObjectName(_fromUtf8("add_flow_minus"))
286 
287         self.keyboard_button = QtGui.QPushButton(main_ui)
288         self.keyboard_button.setGeometry(QtCore.QRect(510, 10, 151, 29))
289         self.keyboard_button.setObjectName(_fromUtf8("keyboard_button"))
290 
291         self.heater = QtGui.QCheckBox(main_ui)
292         self.heater.setGeometry(QtCore.QRect(680, 40, 81, 24))
293         self.heater.setObjectName(_fromUtf8("heater"))
294 
295         self.level_indicator = QtGui.QLabel(main_ui)
296         self.level_indicator.setGeometry(QtCore.QRect(330, 380, 131, 71))
297         self.level_indicator.setStyleSheet(_fromUtf8("background-color:␣rgb(195,
    ␣195,␣195);\n""font:␣25␣18pt␣\"Roboto\";\n""color:␣rgb(195,␣195,␣195);"))
298         self.level_indicator.setObjectName(_fromUtf8("level_indicator"))
299 
300         self.line_4 = QtGui.QFrame(main_ui)
301         self.line_4.setGeometry(QtCore.QRect(660, 40, 20, 31))
302         self.line_4.setFrameShape(QtGui.QFrame.VLine)
303         self.line_4.setFrameShadow(QtGui.QFrame.Sunken)
304         self.line_4.setObjectName(_fromUtf8("line_4"))
```

```
305
306         self.retranslateUi(main_ui)
307         QtCore.QMetaObject.connectSlotsByName(main_ui)
308
309     def retranslateUi(self, main_ui):
310         main_ui.setWindowTitle(_translate("main_ui", "Form", None))
311         self.shutdown_button.setText(_translate("main_ui", "Shutdown", None))
312         self.power_button.setText(_translate("main_ui", "On/Off", None))
313         self.clear_data_button.setText(_translate("main_ui", "Clear All Data",
    None))
314         self.save_button.setText(_translate("main_ui", "Save .txt", None))
315         self.manual_mode.setText(_translate("main_ui", "Manual Mode", None))
316         self.label.setText(_translate("main_ui", "ºC", None))
317         self.label_2.setText(_translate("main_ui", "LPH", None))
318         self.label_3.setText(_translate("main_ui", "Flow Rate", None))
319         self.label_4.setText(_translate("main_ui", "Target Temperature", None))
320         self.automatic_mode.setText(_translate("main_ui", "Automatic Mode", None
    ))
321         self.label_9.setText(_translate("main_ui", "Start Time 1", None))
322         self.label_5.setText(_translate("main_ui", "LPH", None))
323         self.label_14.setText(_translate("main_ui", "Flow Rate 2", None))
324         self.label_6.setText(_translate("main_ui", "Flow Rate 1", None))
325         self.label_8.setText(_translate("main_ui", "ºC", None))
326         self.label_7.setText(_translate("main_ui", "Target Temperature 1", None)
    )
327         self.label_11.setText(_translate("main_ui", "Start Time 2", None))
328         self.label_16.setText(_translate("main_ui", "Start Time 3", None))
329         self.label_18.setText(_translate("main_ui", "ºC", None))
330         self.label_12.setText(_translate("main_ui", "LPH", None))
331         self.label_19.setText(_translate("main_ui", "Flow Rate 3", None))
332         self.label_10.setText(_translate("main_ui", "Target Temperature 2", None
    ))
333         self.label_13.setText(_translate("main_ui", "ºC", None))
334         self.label_15.setText(_translate("main_ui", "Target Temperature 3", None
    ))
335         self.label_17.setText(_translate("main_ui", "LPH", None))
336         self.label_22.setText(_translate("main_ui", "Temperature", None))
337         self.label_24.setText(_translate("main_ui", "Pressure 1", None))
338         self.label_23.setText(_translate("main_ui", "ºC", None))
339         self.label_25.setText(_translate("main_ui", "mmHg", None))
340         self.label_26.setText(_translate("main_ui", "mmHg", None))
341         self.label_21.setText(_translate("main_ui", "LPH", None))
342         self.label_20.setText(_translate("main_ui", "Flow Rate", None))
343         self.label_27.setText(_translate("main_ui", "Pressure 2", None))
```

```
344         self.new_specimen_button.setText(_translate("main_ui", "New␣Specimen",
    None))
345         self.add_flow_plus.setText(_translate("main_ui", "+", None))
346         self.add_flow_minus.setText(_translate("main_ui", "-", None))
347         self.keyboard_button.setText(_translate("main_ui", "Keyboard␣On/Off",
    None))
348         self.heater.setText(_translate("main_ui", "Heater", None))
349         self.level_indicator.setText(_translate("main_ui", "␣␣Fluid␣Level", None
    ))
350
351 from pyqtgraph import PlotWidget
```

A.4.2.2 Interface gráfica da informação sobre o espécime

Listagem A.8: Código para a interface gráfica da informação sobre o espécime

```
1  # -*- coding: utf-8 -*-
2  ########################################
3  ## Setup file for the secundary ui ##
4  ########################################
5
6  from PyQt4 import QtCore, QtGui
7
8  try:
9      _fromUtf8 = QtCore.QString.fromUtf8
10 except AttributeError:
11     def _fromUtf8(s):
12         return s
13
14 try:
15     _encoding = QtGui.QApplication.UnicodeUTF8
16     def _translate(context, text, disambig):
17         return QtGui.QApplication.translate(context, text, disambig, _encoding)
18 except AttributeError:
19     def _translate(context, text, disambig):
20         return QtGui.QApplication.translate(context, text, disambig)
21
22 class Ui_data_ui(object):
23     def setupUi(self, data_ui):
24         data_ui.setObjectName(_fromUtf8("data_ui"))
25         data_ui.resize(800, 480)
26         data_ui.setStyleSheet(_fromUtf8("background-color:␣rgb(195,␣195,␣195);")
    )
27
28         self.male_check = QtGui.QCheckBox(data_ui)
```

```
29          self.male_check.setGeometry(QtCore.QRect(200, 100, 61, 33))
30          self.male_check.setObjectName(_fromUtf8("male_check"))
31
32          self.female_check = QtGui.QCheckBox(data_ui)
33          self.female_check.setGeometry(QtCore.QRect(270, 100, 81, 33))
34          self.female_check.setObjectName(_fromUtf8("female_check"))
35
36          self.label = QtGui.QLabel(data_ui)
37          self.label.setGeometry(QtCore.QRect(200, 60, 31, 19))
38          self.label.setObjectName(_fromUtf8("label"))
39
40          self.line = QtGui.QFrame(data_ui)
41          self.line.setGeometry(QtCore.QRect(200, 80, 151, 16))
42          self.line.setFrameShape(QtGui.QFrame.HLine)
43          self.line.setFrameShadow(QtGui.QFrame.Sunken)
44          self.line.setObjectName(_fromUtf8("line"))
45
46          self.observations = QtGui.QTextEdit(data_ui)
47          self.observations.setGeometry(QtCore.QRect(20, 190, 761, 261))
48          self.observations.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);"))
49          self.observations.setObjectName(_fromUtf8("observations"))
50
51          self.label_2 = QtGui.QLabel(data_ui)
52          self.label_2.setGeometry(QtCore.QRect(20, 160, 111, 21))
53          self.label_2.setObjectName(_fromUtf8("label_2"))
54
55          self.label_3 = QtGui.QLabel(data_ui)
56          self.label_3.setGeometry(QtCore.QRect(270, 0, 241, 41))
57          self.label_3.setObjectName(_fromUtf8("label_3"))
58
59          self.line_2 = QtGui.QFrame(data_ui)
60          self.line_2.setGeometry(QtCore.QRect(20, 40, 761, 16))
61          self.line_2.setFrameShape(QtGui.QFrame.HLine)
62          self.line_2.setFrameShadow(QtGui.QFrame.Sunken)
63          self.line_2.setObjectName(_fromUtf8("line_2"))
64
65          self.create_specimen_button = QtGui.QPushButton(data_ui)
66          self.create_specimen_button.setGeometry(QtCore.QRect(650, 100, 110, 31))
67          self.create_specimen_button.setObjectName(_fromUtf8("
        create_specimen_button"))
68
69          self.label_4 = QtGui.QLabel(data_ui)
70          self.label_4.setGeometry(QtCore.QRect(380, 60, 161, 19))
71          self.label_4.setObjectName(_fromUtf8("label_4"))
```

```
72
73          self.line_3 = QtGui.QFrame(data_ui)
74          self.line_3.setGeometry(QtCore.QRect(380, 80, 151, 16))
75          self.line_3.setFrameShape(QtGui.QFrame.HLine)
76          self.line_3.setFrameShadow(QtGui.QFrame.Sunken)
77          self.line_3.setObjectName(_fromUtf8("line_3"))
78
79          self.arrival_date = QtGui.QDateEdit(data_ui)
80          self.arrival_date.setGeometry(QtCore.QRect(380, 100, 151, 33))
81          self.arrival_date.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣
        255,␣255);"))
82          self.arrival_date.setObjectName(_fromUtf8("arrival_date"))
83
84          self.line_4 = QtGui.QFrame(data_ui)
85          self.line_4.setGeometry(QtCore.QRect(20, 140, 761, 16))
86          self.line_4.setFrameShape(QtGui.QFrame.HLine)
87          self.line_4.setFrameShadow(QtGui.QFrame.Sunken)
88          self.line_4.setObjectName(_fromUtf8("line_4"))
89
90          self.keyboard_button = QtGui.QPushButton(data_ui)
91          self.keyboard_button.setGeometry(QtCore.QRect(630, 60, 151, 31))
92          self.keyboard_button.setObjectName(_fromUtf8("keyboard_button"))
93
94          self.id = QtGui.QTextEdit(data_ui)
95          self.id.setGeometry(QtCore.QRect(20, 100, 141, 33))
96          self.id.setMinimumSize(QtCore.QSize(0, 31))
97          self.id.setStyleSheet(_fromUtf8("background-color:␣rgb(255,␣255,␣255);")
        )
98          self.id.setObjectName(_fromUtf8("id"))
99
100         self.label_5 = QtGui.QLabel(data_ui)
101         self.label_5.setGeometry(QtCore.QRect(20, 60, 101, 19))
102         self.label_5.setObjectName(_fromUtf8("label_5"))
103
104         self.line_5 = QtGui.QFrame(data_ui)
105         self.line_5.setGeometry(QtCore.QRect(20, 80, 141, 16))
106         self.line_5.setFrameShape(QtGui.QFrame.HLine)
107         self.line_5.setFrameShadow(QtGui.QFrame.Sunken)
108         self.line_5.setObjectName(_fromUtf8("line_5"))
109
110         self.retranslateUi(data_ui)
111         QtCore.QMetaObject.connectSlotsByName(data_ui)
112
113     def retranslateUi(self, data_ui):
114         data_ui.setWindowTitle(_translate("data_ui", "Frame", None))
```

```
115         self.male_check.setText(_translate("data_ui", "Male", None))
116         self.female_check.setText(_translate("data_ui", "Female", None))
117         self.label.setText(_translate("data_ui", "Sex", None))
118         self.label_2.setText(_translate("data_ui", "Observations", None))
119         self.label_3.setText(_translate("data_ui", "<html><head/><body><p␣align
        =\"center\"><span␣style=\"␣font-size:20pt;\">Specimen␣Data</span></p></body
        ></html>", None))
120         self.create_specimen_button.setText(_translate("data_ui", "Create", None
        ))
121         self.label_4.setText(_translate("data_ui", "Arrival␣Date", None))
122         self.keyboard_button.setText(_translate("data_ui", "Keyboard␣On/Off",
        None))
123         self.label_5.setText(_translate("data_ui", "Specimen␣ID", None))
```