UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE FÍSICA
BACHARELADO EM ENGENHARIA FÍSICA

BRUNA FACHIN

AUTOMATIZAÇÃO DO PROCESSO DE PRODUÇÃO DE DISSULFETO DE MOLIBDÊNIO

PORTO ALEGRE
2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE FÍSICA
BACHARELADO EM ENGENHARIA FÍSICA

BRUNA FACHIN

AUTOMATIZAÇÃO DO PROCESSO DE PRODUÇÃO DE DISSULFETO DE MOLIBDÊNIO

Trabalho de diplomação em Engenharia Física entregue à Universidade Federal do Rio Grande do Sul como requisito parcial para a obtenção do grau de Bacharel em Engenharia Física.
Orientador: Prof. Dr. Gabriel Vieira Soares

PORTO ALEGRE
2021

AGRADECIMENTOS

Primeiramente, agradeço à minha mãe, Eva, por todos os esforços realizados para que eu tivesse as oportunidades que tive, por todas as vezes que deixou de fazer algo por ela para permitir que eu fizesse por mim, por todas as vezes que me entendeu e apoiou nas escolhas que fiz: tu és meu maior exemplo. Ao lado dela, meu padrasto, Mauro, por estar sempre presente e disposto a me ajudar incondicionalmente sempre que precisei.

À Enfitec Júnior, por ter sido minha família dentro do curso de Engenharia Física, por ter me acolhido e me ensinado a crescer tanto no âmbito profissional quanto pessoal ao longo dos dois anos que tive a oportunidade de fazer parte da empresa júnior, e por estar presente também na execução deste trabalho. Ao CTA, pelas contribuições neste projeto, e em especial ao Renan, por me salvar quando precisei de ajuda de última hora.

Agradeço também a alguns professores que fizeram a diferença ao longo da minha trajetória: começando pelo Ismael Lima, professor de física do ensino médio, que sempre me incentivou a seguir esse caminho e acreditou no meu potencial, ao Leandro Langie por ter me incentivado a acreditar em mim mesma e a não desistir logo no início da graduação, e ao Gabriel Soares, por me ajudar a enfrentar todas as dificuldades encontradas ao longo deste trabalho.

“It takes courage to make it in this land
So don’t forget, but forgive every man
And prosperity’s river, it will forever flow
Honor is among us, honor is all we know”
Rancid - Honor is all we know

RESUMO

Por meio da tecnologia, o mundo está cada vez mais conectado, possibilitando a transmissão de informações e comunicação em uma velocidade jamais vista. Com isso, a demanda por dispositivos eletrônicos, como smartphones, vem aumentando consideravelmente e as pessoas se tornam cada vez mais dependentes do seu uso, exigindo, por consequência, maior qualidade e funcionalidade desses aparelhos. Para que essa necessidade seja suprida, a tecnologia relacionada deve estar sempre sendo aprimorada, e um componente chave que precisa fazer parte dessa evolução tecnológica é o transistor. A partir de materiais bidimensionais como o dissulfeto de molibdênio, é possível avançar mais um passo para o aperfeiçoamento e redução das dimensões de um transistor – porém, é necessário que esse material seja estudado a fundo para que possa ser desenvolvido com a melhor qualidade possível e cumprir o objetivo esperado dele. No Laboratório de Superfícies e Interfaces Sólidas do Instituto de Física (FQSIS) da Universidade Federal do Rio Grande do Sul, encontra-se um reator CVD utilizado para síntese de dissulfeto de molibdênio, entretanto, o processo realizado no LASIS ainda é muito manual e pouco prático, prejudicando os resultados finais. Dessa forma, o objetivo deste trabalho de diplomação em Engenharia Física II é o desenvolvimento de uma aprimoração no processo de produção do dissulfeto de molibdênio feito no Instituto de Física da UFRGS, a partir da automatização do controle de fluxo de entrada dos gases de arraste necessários ao processo. Essa automatização é colocada em prática por meio da programação de uma interface visual via computador, utilizando o software Processing, e programação de um circuito com Arduino, visando maior praticidade de manuseio pelo usuário e maior precisão na síntese do composto.

Palavras-chave: materiais bidimensionais, transistor, automatização de processos, eletrônica, programação.

ABSTRACT

Through technology, the world is increasingly connected, enabling the transmission of information and communication at a speed never seen before. As a result, the demand for electronic devices, such as smartphones, has been increasing considerably and people have become increasingly dependent on their use, therefore demanding greater quality and functionality in these devices. For this need to be met, the related technology must always be improved, and a key component that needs to be part of this technological evolution is the transistor. From two-dimensional materials such as molybdenum disulfide, it is possible to go one step further towards improving and reducing the dimensions of a transistor – however, this material needs to be studied in depth so that it can be developed with the best possible quality and fulfill his expected goal. In the Laboratory of Solid Surfaces and Interfaces of the Physics Institute (FQSIS) of the Federal University of Rio Grande do Sul, there is a CVD reactor used for molybdenum disulfide synthesis, however, the process carried out in LASIS is still very manual and not practical, hampering the final results. Thus, the objective of this degree work in Physical Engineering II is the development of an improvement in the molybdenum disulfide production process carried out at the UFRGS Institute of Physics, through the automation of the inlet flow control of the needed carrier gases to the process. This automation is put into practice by programming a visual interface through a computer, using a software named Processing, and programming a circuit with Arduino, aiming at greater user-friendliness and greater precision in the synthesis of the compound.

Keywords: two-dimensional materials, transistor, process automation, electronics, programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

BJT – transistor de junção bipolar;

MOSFET – transistor de efeito de campo feito de metal, óxido e semicondutor;

hBN – nitreto de boro hexagonal;

MoS_2 – dissulfeto de molibdênio;

TMD – *transition metal dichalcogenides* (dicalcogeneto de metal de transição);

FLG – *few-layer graphene* (grafeno de poucas camadas);

MBG FET – transistor de efeito de campo feito com MoS_2, hBN e FLG;

CVD – *chemical vapor deposition* (deposição química a partir da fase vapor);

LASIS – Laboratório de Superfícies e Interfaces Sólidas do Instituto de Física;

MoO_3 – óxido de molibdênio;

S – enxofre sólido;

MFC – *mass flow controller* (controlador de fluxo de massa);

sccm – *standard cubic centimetres per minute* (centímetros cúbicos padrão por minuto);

db15 – conector tipo “D” de 15 pinos e duas fileiras;

IDE Arduino – *Arduino integrated development environment* (ambiente de desenvolvimento integrado Arduino);

I3D – impressão tridimensional;

PWM – *pulse width modulation* (modulação de largura de pulso);

PCI – placa de circuito impresso;

CTA – Centro de Tecnologia Acadêmica.

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

Globalização é um conceito relacionado à expansão de culturas, economia e política a nível mundial que ocorre de forma intrínseca ao ser humano, com grandes exemplos como na época das expedições marítimas no século XVI e exploração de novas terras, tendo seu impacto fomentado pelas consequentes revoluções industriais que permitiram avanços tecnológicos e econômicos de impacto mundial. Esse processo de expansão foi amplamente acelerado após a Terceira Revolução Industrial, conhecida também como Revolução Técnico-Científico-Informacional, que elevou o conceito de globalização a um novo patamar com as tecnologias relacionadas a transportes e em especial à comunicação[1].

Dispositivos eletrônicos portáteis como *smartphones* são amplamente utilizados na atualidade e trazem conexão com o mundo todo a partir de poucos cliques, porém sua simplicidade de uso não corresponde diretamente à complexidade de sua arquitetura: essa tecnologia descende de anos de evolução e estudos em componentes eletrônicos. Em 1930, foi patenteado o conceito do transistor de efeito de campo, tendo sido colocado em prática apenas em 1960. Desde então, o transistor foi incorporado em circuitos integrados e tornou-se um dos componentes mais importantes para a indústria eletrônica[2].

Sua importância se dá pelas funções como amplificador, chave ou controle de corrente elétrica em uma carga, assim como o transistor de junção bipolar (BJT), primeiro transistor a ser inventado com objetivo de substituir válvulas termoiônicas[3]. Entretanto, o transistor de efeito de campo feito de metal, óxido e semicondutor, chamado de MOSFET, apresenta várias vantagens em relação ao BJT por poder ser feito em dimensões muito menores, além de apresentar baixo consumo de energia e dissipação de potência, tornando seu uso ideal para dispositivos eletrônicos[4].

Um MOSFET é constituído por três eletrodos, chamados de fonte, dreno e porta (isolada do resto do sistema por um dielétrico), como demonstrado na imagem abaixo:

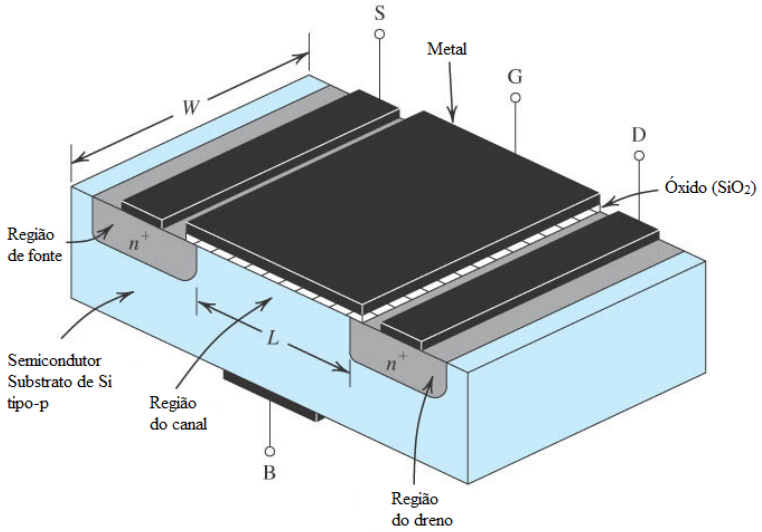


Figura 1 – Estrutura física do MOSFET vista em perspectiva. Retirada de [5].

A região de canal desse componente é comumente feito de Silício, um material semicondutor tridimensional, que permite a passagem de corrente elétrica entre a fonte e o dreno a partir da geração de um campo elétrico quando é aplicada uma tensão no eletrodo da porta[5].

O aumento exponencial da quantidade desse componente dentro de um circuito integrado desde 1960 é diretamente responsável pelo avanço na qualidade e funcionalidade de dispositivos eletrônicos, além de sua redução de custos[6]. Para isso, ocorre uma constante miniaturização de sua estrutura que permite essa exponencialidade sem prejudicar características importantes dos dispositivos, como por exemplo a necessidade de serem portáteis e leves.

A miniaturização dos transistores foi observada pelo cofundador da Intel Corporation Gordon Moore em 1965, que previu que a quantidade de transistores em um circuito integrado dobraria a cada dois anos, custando o mesmo valor para sua produção[7]. Sua previsão ficou conhecida como Lei de Moore e acabou se tornando um objetivo para as indústrias de semicondutores, fazendo-as investirem muitos recursos para alcançar essa meta[8].

Entretanto, essa previsão pode estar chegando ao fim, devido a uma desaceleração da indústria na inserção de cada vez mais transistores em um circuito integrado, pois componentes cada vez menores impõem rigor cada vez maior nas etapas do seu processo de fabricação[5]. Um dos motivos para essa desaceleração é decorrente da diminuição da região do canal do transistor, pois quando ele apresenta a mesma ordem de magnitude da região de depleção entre fonte e dreno,

ocorrem fenômenos chamados de efeitos de canal curto. Um desses efeitos é a redução da barreira de potencial induzida pelo dreno, que acaba permitindo a passagem de alguns elétrons entre a fonte e o dreno mesmo que não haja tensão sendo aplicada[9].

Problemas como esse inserem impedimentos no caminho da indústria de transistores, gerando a necessidade de serem estudadas soluções que podem ir desde a melhora dos métodos de fabricação até o desenvolvimento de novas tecnologias[6]. Uma das soluções estudada atualmente que permite que essa tecnologia possa continuar avançando é a substituição dos materiais tridimensionais utilizados no transistor como metal, óxido e semicondutor por outros mais tecnológicos, como os materiais bidimensionais.

Materiais bidimensionais são estruturas que se diferem dos tridimensionais por serem feitos de camadas únicas de átomos[10], um dos mais conhecidos e primeiro a ser descoberto é o grafeno, sintetizado pela primeira vez em 2004 a partir da esfoliação mecânica do grafite tridimensional. Algumas das suas características que conferem tanta importância a esse material para a tecnologia são sua alta condutividade térmica em temperatura ambiente, alta condutividade elétrica e alta transparência, podendo ser utilizado em diversas aplicações como eletrodos transparentes e células solares[11].

Após a descoberta do grafeno que garantiu o Prêmio Nobel de Física de 2010 para os cientistas que o estudaram, vários outros materiais bidimensionais começaram a ser sintetizados em monocamadas por processos parecidos com o do grafeno e estão sendo estudados para diferentes usos de acordo com as propriedades físicas adquiridas nessa síntese. Um exemplo é o nitreto de boro hexagonal (hBN), que adquire propriedades isolantes em temperatura ambiente, além de ter outras características similares ao grafeno como alta resistência mecânica e alta transparência[12].

Outro material bidimensional que vem sendo muito estudado é o dissulfeto de molibdênio (MoS_2) que faz parte da família dos dicalcogenetos de metal de transição (*transition metal dichalcogenides*, TMD) e segue a fórmula química MX_2 (sendo M o metal e X o calcogeneto)[13]. Ao ser sintetizado em monocamadas, adquire propriedades semicondutoras interessantes para aplicação em dispositivos eletrônicos[11].

Esses materiais têm grande potencial para aplicação na indústria de dispositivos eletrônicos, já havendo estudos para elaboração de um transistor baseado em materiais bidimensionais, com o canal feito de MoS_2, dielétrico que isola a porta feito de hBN e porta feita de grafeno com poucas camadas (*few-layer graphene*, FLG). Uma construção desse tipo permite uma melhor mobilidade de efeito de campo operando com baixa tensão de porta, além de gerar um componente final com alta flexibilidade e transparência óptica, exibindo pouca modificação de desempenho em altos níveis de tensão[14].

A imagem abaixo exemplifica a construção desse transistor chamado de MBG FET, feito com materiais bidimensionais:

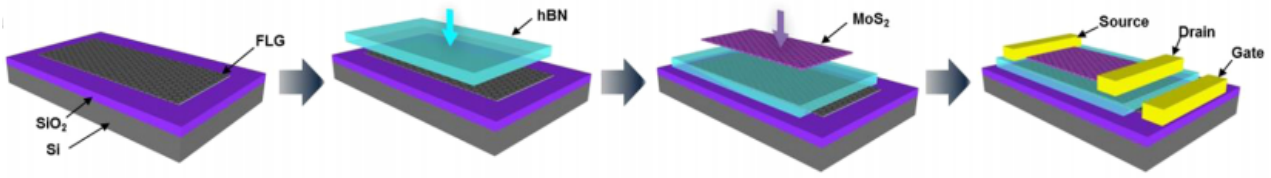


Figura 2 – Esquemático da fabricação de dispositivos MBG[12].

Esse tipo de construção em camadas é chamado de heteroestruturas de Van Der Waals, que consiste no empilhamento de camadas atômicas de materiais bidimensionais que interagem entre si através da força de Van Der Waals. A partir dessas interações, os materiais adquirem novas propriedades e funcionalidades, tornando possível a construção de dispositivos eletrônicos como os transistores[13].

Para que esses materiais bidimensionais possam ser aplicados em larga escala para dispositivos eletrônicos, é necessário garantir uma síntese de alta qualidade, sendo importante a evolução dos estudos na área. Uma forma de sintetizar esses materiais em camadas macroscópicas que facilitem seu estudo e caracterização é com o processo de deposição química a partir da fase vapor (CVD)[15]. Esse processo se dá pela formação de um filme fino por deposição a partir de uma reação química na superfície do substrato, utilizando substâncias ou gases que contêm os elementos formadores do filme fino[16].

Na Universidade Federal do Rio Grande do Sul, há reatores CVD para síntese de grafeno e de MoS_2 no Laboratório de Superfícies e Interfaces Sólidas do Instituto de Física (LASIS). Entretanto, várias partes do processo nesses reatores são feitas

de forma manual, tornando o resultado da síntese muito menos preciso do que o necessário e prejudicando a qualidade dos filmes sintetizados.

A partir disso, surgiu a proposta de projeto deste Trabalho de Diplomação em Engenharia Física, que visa uma aprimoração do processo de produção de materiais bidimensionais por meio de automatizações, com foco no reator CVD onde é feita a síntese de dissulfeto de molibdênio no LASIS.

1.2 Objetivos

Para a síntese do dissulfeto de molibdênio por CVD, é necessária a utilização de um substrato inicial para deposição, o qual é comumente feito de um óxido depositado sobre silício, podendo ser óxido de silício (SiO_2/Si). Esse substrato é inserido no tubo de quartzo do reator CVD junto com os precursores do MoS_2, que são uma fonte sólida de óxido de molibdênio (MoO_3) colocada sobre um cadinho a 5cm de distância do substrato e outra fonte também sólida de enxofre (S) colocada sobre um cadinho a 20cm de distância do substrato. A fonte de S é colocada mais distante do substrato pois ela necessita de uma temperatura mais baixa para sublimação, enquanto que a fonte de MoO_3 é colocada próxima ao substrato para que atinjam a mesma temperatura ao mesmo tempo, conforme a rampa de temperatura programada inicialmente para cada finalidade[17].

A figura abaixo exemplifica a disposição do substrato e fontes internamente ao tubo do reator CVD:

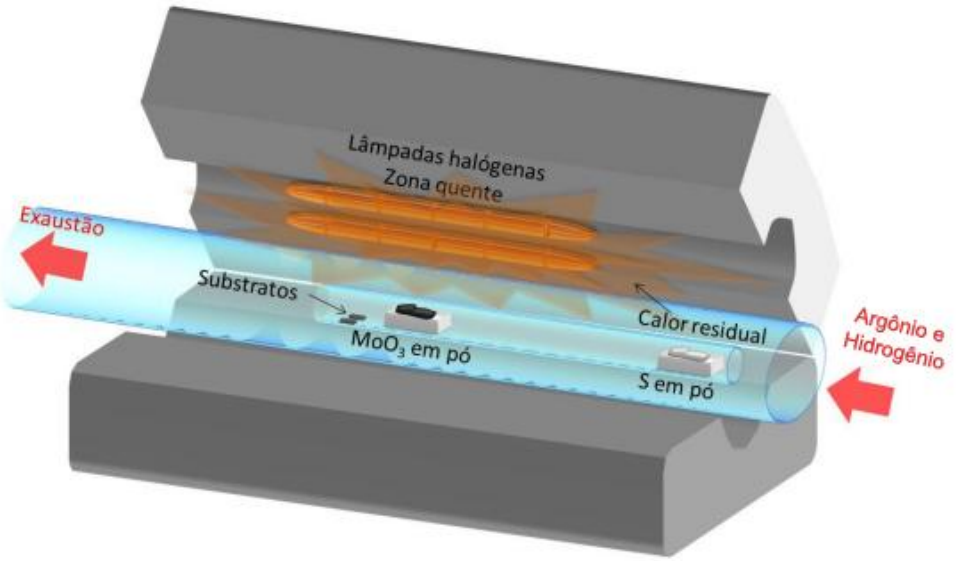


Figura 3 – Representação esquemática do reator CVD utilizado para síntese de MoS_2. Adaptada de [17].

Após a inserção do substrato e fontes no reator, é realizado o bombeamento da câmara com uma bomba mecânica, na saída para o sistema de exaustão do tubo, para que a pressão diminua até atingir em torno de 0,4 Torr, criando um nível médio de vácuo. Ao atingir esse valor, são inseridos os gases argônio (Ar) e hidrogênio (H_2) no lado oposto do tubo até que a pressão aumente para em torno de 5,0 Torr, sendo mantida constante ao longo do experimento[17]. O gás argônio é utilizado pois realiza a função de arraste dos precursores, enquanto que o gás hidrogênio age na limpeza da superfície do substrato, não sendo obrigatório para a síntese do MoS_2.

A constância da pressão ao longo do processo é garantida pela escolha do fluxo dos gases inseridos, que devem ser selecionados no início do experimento, de modo a se relacionarem diretamente com a taxa de bombeamento no sistema de exaustão do reator. Assim, mantém-se uma taxa constante de entrada e de saída de gases no tubo de quartzo, resultando numa pressão também constante.

É iniciado então o aquecimento do tubo a partir das lâmpadas halógenas, na qual é possível ajustar, por meio de um controlador, uma rampa de temperatura para que ela aumente gradualmente até chegar na temperatura desejada ao final do tempo programado para a síntese. Essa temperatura final pode atingir em torno de 500 a 700 ºC no centro do tubo, dependendo do objetivo do experimento, e um valor menor na região afastada onde fica o enxofre. Finalmente tem-se o crescimento dos filmes de MoS_2 assim que as fontes começam a sublimar e se depositam sobre o substrato[17].

Atualmente no LASIS, várias dessas etapas são realizadas de forma manual, como por exemplo a contabilização do tempo total do experimento, que é feita pelo usuário por meio de um cronômetro, sem conexão com o sistema de liberação dos gases nem do controlador de temperatura. Levando em conta que o tempo total do experimento interfere diretamente na quantidade de camadas depositadas de MoS_2[17], essa forma de controle atual acarreta incertezas no resultado da síntese inerentes ao processo, pois depende de ação manual para iniciar e terminar a contagem.

Outro processo feito de forma manual é relacionado aos gases argônio e hidrogênio. Para sua liberação, é necessário abrir manualmente os cilindros que os contêm, e no caminho até o reator os gases passam primeiro por um controlador de

fluxo cada um, que irá modular o valor desejado deles antes de serem inseridos no tubo do reator CVD. Entretanto, para esse ajuste dos valores de fluxo nos controladores, é utilizada uma chave de fenda para girar um potenciômetro que fica acoplado em cada um até chegar no valor desejado, mostrado no visor do equipamento em tempo real.

Além da dificuldade gerada para o usuário nesse ajuste manual, assim que os cilindros são abertos os gases já são liberados no reator antes mesmo de o valor do fluxo estar corretamente selecionado nos controladores. Assim, já se pode considerar o tempo despendido no ajuste dos fluxos e sua falta de praticidade inerentemente negativos, podendo ainda serem gerados problemas na manutenção da pressão interna do reator e posteriormente na qualidade das camadas sintetizadas do MoS_2.

Analisando esses problemas na liberação e ajuste do fluxo dos gases e no controle do tempo do experimento, foi desenvolvido este projeto de automatização do processo de produção de dissulfeto de molibdênio, que visa o controle de forma remota dos parâmetros supracitados. Ao final do projeto, é objetivado um aumento da praticidade para o usuário do reator CVD e da qualidade na síntese do MoS_2 ao serem suprimidas incertezas e erros inerentes aos processos manuais.

2. FUNDAMENTAÇÃO TEÓRICA

Para execução das automatizações propostas, será enfocado o controle, por meio de um computador, do tempo total do experimento e do fluxo de cada um dos gases utilizados na síntese do MoS_2. Esse processo todo pode ser executado utilizando os controladores de fluxo já existentes no laboratório de uma maneira diferente da qual eles estão sendo utilizados atualmente. Dessa forma, para compreender posteriormente o método que será utilizado, é necessário entender como funciona esse equipamento e como adaptá-lo para melhorar a performance do reator CVD.

2.1 Controladores de fluxo

Um controlador de fluxo, chamado em inglês de *mass flow controller* (MFC), é um equipamento que pode ser acoplado em cilindros de gases ou outros fluidos para regular o fluxo deles por meio de um sistema interno como o exemplificado na imagem abaixo:

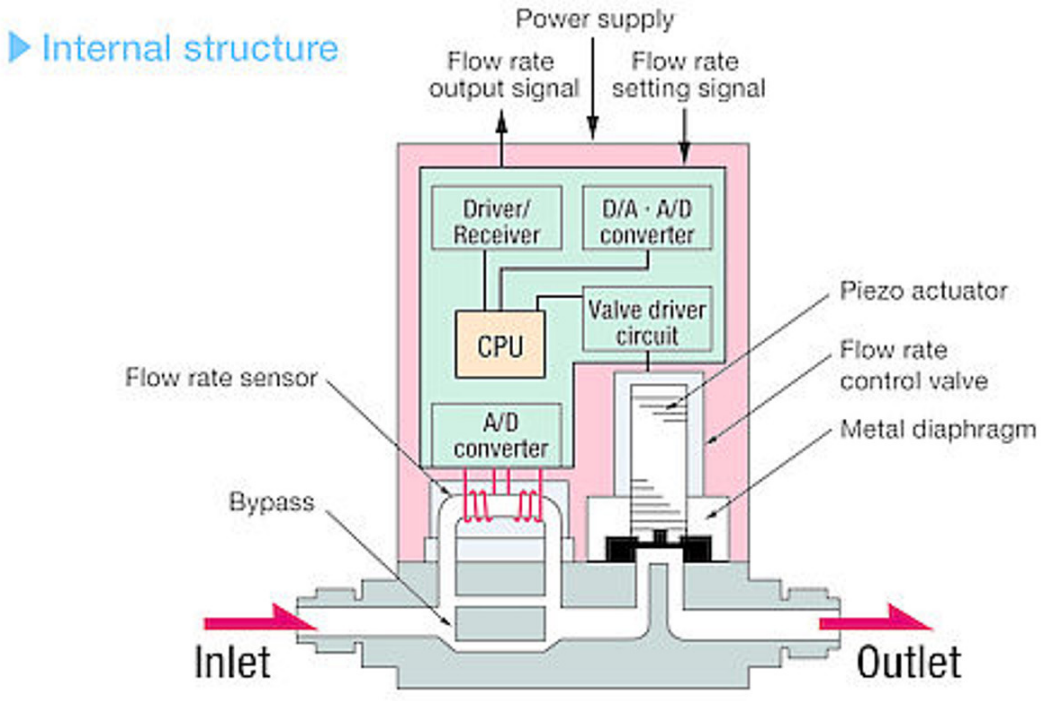


Figura 4 – Esquemático interno de um MFC. Retirado de [18].

Ao entrar no MFC, o gás em questão se divide entre dois caminhos, sendo que um deles, chamado de *bypass*, apenas direciona uma parte desse gás para a

saída do controlador sem passar por nenhum processo. O outro caminho no qual o gás passa após ser dividido é onde ocorrerá a medição do fluxo atual desse gás. A geometria dos dois capilares por onde o gás se divide é construída de modo que o fluxo entre ambos se relacione, de acordo com princípios de capilaridade de mecânica dos fluidos[19].

Neste segundo caminho pelo qual o gás passa, o fluxo é detectado por um transdutor como uma variação de temperatura proporcional, chamada de fluxo de calor. Esse valor detectado é posteriormente convertido em sinal elétrico pelo transdutor e enviado a um circuito de correção e amplificação que o transforma numa variação de 0 a 5V. Ao mesmo tempo, ele é enviado para um sistema de comparação, que compara o valor do fluxo atual do gás com o valor que foi definido para sua saída por meio do ajuste manual do usuário. Esse valor de diferença entre os fluxos definidos pelo usuário e o fluxo inicial do gás é enviado para o circuito de condução da válvula[18].

O sistema da válvula é a última etapa para ajuste do fluxo antes de ocorrer a saída do gás e funciona por meio de um atuador piezo. Esse atuador recebe o sinal elétrico produzido previamente e o converte em uma ação mecânica, produzindo uma movimentação na válvula proporcional ao valor de fluxo desejado para a saída do gás. Assim, o volume pelo qual o gás pode passar na saída do MFC é aumentado ou diminuído, de modo que a diferença entre o fluxo definido e o fluxo atual seja o mais próxima de zero possível[18]. O circuito de controle do MFC está constantemente realizando essas medidas comparativas e corrigindo o valor de saída por compensação da válvula quando necessário, mantendo o fluxo de saída constante[19].

O modelo de controlador de fluxo utilizado no laboratório é o FMA5512A da marca Omega com alcance de 0 a 500 ml/min, unidade que também é conhecida e mais utilizada como “sccm” (*standard cubic centimetres per minute*, ou centímetros cúbicos padrão por minuto). Ele foi inicialmente construído para funcionamento com gás metano (CH_4), porém, por meio de uma calibração, é possível adaptá-lo para o gás desejado. No LASIS, existe um modelo adaptado para o argônio e outro para o gás hidrogênio, que são como os da imagem abaixo:

Figura 5 – Foto do controlador de fluxo utilizado no LASIS.

A calibração do MFC para adaptação ao gás de interesse é feita por meio de um cálculo de um fator K que incorpora a densidade do gás (*d*) e o coeficiente de calor específico (*Cp*) dele, pela equação abaixo:

$$K = \frac{1}{d.Cp} \quad (1)$$

Caso a faixa de operação do fluxo do equipamento não seja alterada, então esse fator K pode ser calculado da seguinte forma:

$$K = \frac{Qa}{Qr}\frac{Ka}{Kr} \quad (2)$$

Nessa equação, "*Qa*" é o fluxo do gás que será utilizado, "*Qr*" é o fluxo do gás para o qual o MFC foi projetado para operar, e os "K"s são os fatores referentes a cada um deles, calculados pela equação (1). Com essa conversão, é possível utilizar um MFC projetado para o gás metano e adaptá-lo para o uso com o argônio ou gás hidrogênio, fazendo uma calibração do valor do fluxo que aparece no visor do controlador.

2.2 Pinagem do MFC

Os controladores de fluxo podem ser comandados de duas formas: local ou remotamente. Atualmente, a seleção dos valores de fluxo desejado é feita manualmente, como já explicado, por meio do modo local de funcionamento do MFC. Para a execução das automatizações propostas, é necessário alterá-lo para

funcionamento no modo remoto por meio de uma combinação das ligações dos *jumpers* que controlam a válvula.

Na imagem abaixo, temos um esquemático visual dos *jumpers* de controle, que funcionam conectando dois pontos de acordo com o propósito esperado (como por exemplo os pontos 2 e 3 ou 1 e 2), por isso seu ajuste é feito pela escolha de dois números, como na tabela da imagem. O número central (no exemplo dado, o pino central é o de número 2) sempre estará conectado, mudando apenas a posição da outra ponta do *jumper* para cima (pino 3) ou para baixo (pino 1), com conexões apenas verticalmente.

FUNCTION		NJ1A	NJ1B	NJ1C	NJ1D	NJ1E
Remote	0 to 5 Vdc 2% cutoff ON	2 - 3	5 - 6	8 - 9	10 - 11	13 - 14
	0 to 5 Vdc 2% cutoff OFF					14 -15
	4 to 20 mA 2% cutoff ON	1 - 2	4 - 5	7 - 8	10 - 11	13 - 14
	4 to 20 mA 2% cutoff OFF					14 - 15
Local	2% cutoff ON	2 - 3	5 - 6	8 - 9	11 - 12	13 -14
	2% cutoff OFF					14 - 15

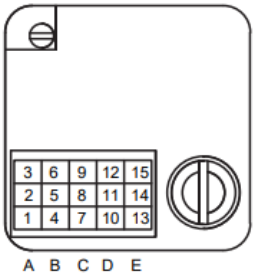


Figura 6 – Configuração dos *jumpers* de controle da válvula do MFC. Retirado de [19].

As diversas combinações de pinos que podem ser feitas giram em torno da escolha entre modo local ou remoto, além de ser possível no modo remoto definir se o fluxo será detectado por meio de uma entrada proporcional a ele de tensão (V) ou corrente elétrica (A). O terceiro ajuste que pode ser feito por meio da combinação dos pinos é o *cutoff*, um comando que, quando ligado, tem o poder de interromper a energia da válvula do MFC quando o valor do fluxo definido pelo usuário for de menos de 2% do valor total da escala.

Uma segunda configuração que deve ser feita no MFC, após a troca de posição de alguns *jumpers*, é o correto ajuste dos valores enviados à pinagem de entrada de um conector tipo “D” de 15 pinos e duas fileiras. Esse conector, quando o MFC está configurado no modo local, funciona apenas para energizar o

equipamento e é ligado por um cabo diretamente na tomada. Entretanto, quando se deseja controlar remotamente o equipamento, existe uma série de configurações que devem ser feitas pelo conector db15.

Seus pinos também devem ser utilizados em pares no modo remoto, já que quase todos recebem parâmetros em unidade de tensão, sendo necessário sempre ter uma diferença de potencial entre dois pinos para o correto entendimento do valor enviado ao MFC. O ajuste é feito de acordo com as conexões demonstradas na imagem abaixo:

PIN	FUNCTION
1	0 to 5 VDC Flow Signal Common
2	0 to 5 VDC Flow Signal Output
3	Common
4	Open (Purge)
5	Common, Power Supply
6	(unassigned)
7	+12 VDC (Optional +24 VDC*) Power Supply
8	Remote Setpoint Input
9	4 to 20 mA (-) Flow Signal Return (use with 14)
10	Remote Setpoint Common (use with 8)
11	+5VDC Reference Output for Remote Setpoint
12	Valve Off Control
13	Auxiliary +12 VDC (Optional +24 VDC*) Power Output (For Loads <100 mA)
14	4 to 20 mA (+) Flow Signal Output
15	Chassis Ground

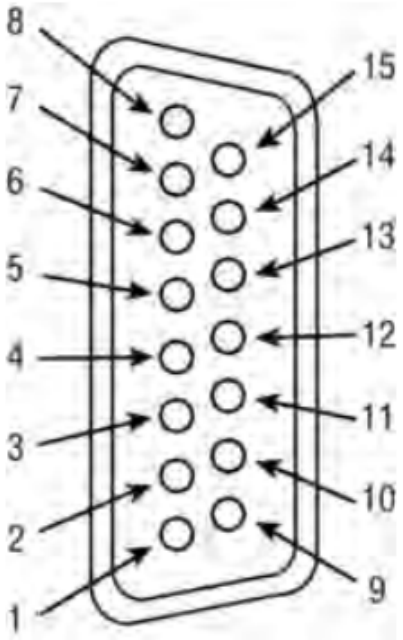


Figura 7 – Configuração do conector db15 do MFC. Retirado de [19].

Por meio desse conector e do correto sinal aplicado em cada um dos 15 pinos, o MFC deve receber os seguintes parâmetros: ser energizado com a mesma tensão que recebia previamente no modo local, receber o valor do fluxo ajustado de forma remota que pode ser em unidades de tensão ou corrente elétrica, e receber também uma referência para comparação dos valores máximo e mínimo do fluxo. É possível também controlar a abertura e fechamento da válvula, porém esse ajuste é opcional e pode atrapalhar os resultados finais.

A partir do entendimento e ajuste dessas configurações do MFC, se torna possível controlá-lo por meio de um circuito e programação via computador, realizando assim as automatizações visadas.

3. METODOLOGIA

Para alcançar os objetivos esperados com esse projeto, se torna necessário fazer o uso de diversas ferramentas que, em conjunto, darão o retorno esperado ao final dele. Dessa forma, o primeiro passo é iniciar a conexão com os MFCs a partir dos seus respectivos conectores tipo “D” de 15 pinos, sendo necessário utilizar cabos db15, próprios para isso, conectados em um circuito com um microcontrolador que envie corretamente os sinais esperados para cada um desses pinos.

Na fase de obtenção desses sinais, é necessário realizar a programação do microcontrolador por meio de um código, para que ele aja como o esperado. Além disso, os sinais precisam ser passados para esse código que comanda o microcontrolador por meio de um computador, se fazendo relevante então a construção, também por meio de um software programável, de uma interface visual que faça a conexão entre o usuário do sistema e a programação do microcontrolador.

Portanto, o método utilizado para ajuste das configurações dos controladores de fluxo depende de um aparato que envolve um circuito, um microcontrolador, um software de programação do microcontrolador e um software de programação da interface visual. Além disso, foram utilizados mecanismos que contribuíram para a melhor versão final do projeto, como um computador *netbook* portátil, a impressão da placa de circuito e a impressão tridimensional de uma caixa para alocar todo o *hardware*.

Para facilitar o entendimento de todo esse sistema, cada um dos mecanismos utilizados que contribuíram para o resultado final do projeto serão explicados de forma isolada a seguir.

3.1 Circuito e microcontrolador

Os cabos db15 são a forma de conexão que há entre toda a programação e os controladores de fluxo, dessa forma, se faz necessária a existência de um *hardware* que interprete e envie os sinais recebidos pelo computador para esses cabos. Para isso, foi escolhida uma placa de circuito Arduino®, um tipo de circuito impresso que já vem com um microcontrolador incluso, como o da imagem abaixo:

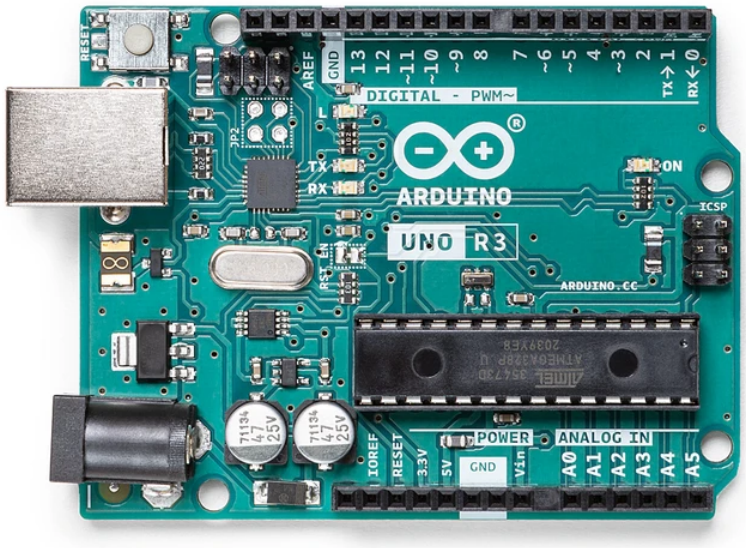


Figura 8 – Placa de circuito Arduino Uno R3. Retirado de [20].

Sua escolha se deu pela usabilidade simples e eficiente, cumprindo todos os requisitos necessários para a execução desse projeto, além de ter um custo baixo. Apesar de ser um circuito que não permite uso comercial dos projetos realizados com ele, é necessário considerar que este projeto foi executado para aplicação no LASIS, dentro da universidade, tornando o Arduino a melhor opção para esse caso em que não é requisitado um microcontrolador com licença comercial.

A tabela abaixo indica a configuração do Arduino utilizado:

Tabela 1 – Especificações do circuito Arduino utilizado. Retirada de [20].

Modelo	Arduino Uno R3
Microcontrolador	ATmega328
Tensão de Operação	5V
Tensão de Entrada	7 a 12V
Portas Digitais	14 (6 para PWM)
Portas Analógicas	6

Para programar o Arduino, foi empregado o ambiente de desenvolvimento integrado (IDE Arduino) original dele, que tem licença de uso gratuita. Nesse

software, é utilizada uma linguagem de programação própria, similar à linguagem C++, e com ele é possível executar grande parte das ações necessárias para o projeto.

3.2 Interface Visual

Visando o aumento da praticidade na experiência do usuário do sistema, é importante implementar uma interface visual computadorizada que permita a inserção dos parâmetros de entrada necessários ao projeto. Não é viável a digitação deles diretamente no código programado na IDE Arduino, pois podem ocorrer modificações não intencionais no código que atrapalhem seu funcionamento.

Pensando nesse ponto, foi utilizado um segundo ambiente de desenvolvimento integrado chamado Processing, com características similares ao do Arduino. Esse *software* também conta com licença gratuita e permite a programação de uma interface por meio da escrita de um código em linguagem própria e similar à C++, tal como na IDE Arduino[21].

Com o Processing, é possível inserir imagens de fundo, escrever textos na tela, realizar a programação de botões clicáveis que executem ações, dentre outras inúmeras formas de programar uma interface visual. Outra vantagem do seu uso é que quando a programação da interface está concluída, é possível gerar um arquivo executável que abre diretamente a tela programada, sem necessidade de abrir o *software*. Dessa forma, não há contato do usuário do sistema com o código da interface, ocasionando uma redução de possíveis problemas relacionados a alterações indevidas dele.

A conexão do Processing com o Arduino se dá por comunicação serial, utilizando o cabo USB do circuito conectado no computador e as portas seriais dele (portas digitais 0 e 1), que ficam reservadas para essa passagem de parâmetros selecionados na interface visual ao microcontrolador. O código que estiver salvo no Arduino pode executar ações com as variáveis recebidas nas portas seriais normalmente, da mesma forma que com as variáveis criadas internamente.

3.3 Computador

Para o funcionamento desse programa que gera a interface visual, é necessário a conexão e instalação do sistema todo em um computador. Entretanto, deve-se levar em conta que não havia inicialmente computadores no LASIS e que o

sistema precisa estar próximo ao reator CVD para conexão dos cabos, sem atrapalhar a disposição dos equipamentos no laboratório. Assim, foi definida a utilização de um computador do tipo *netbook*, que conta com a facilidade de ser portátil e também com dimensões reduzidas pela metade ao ser comparado com um *notebook* tradicional.

Apesar de ser um computador com menor capacidade de processamento, ele cumpre com as funcionalidades necessárias ao sistema. Além disso, sua escolha impacta positivamente no espaço físico do local, pois não interfere na disposição dos equipamentos já instalados no laboratório, inclusive podendo ser transportado para outro local caso seja necessário fazer a manutenção do sistema.

3.4 Placa de circuito impresso

Embora grande parte do funcionamento deste projeto se dê a partir da programação desenvolvida, é necessário se utilizar de algumas conexões básicas via *hardware*. Para os testes iniciais, foi utilizada uma *protoboard* com conexões feitas com *jumpers*, pois permite a alteração dessas ligações de forma prática. Porém, é importante avaliar que o sistema ficará instalado no laboratório, podendo sofrer choques mecânicos e ser movimentado de forma não intencional.

Assim, foi feita uma parceria com o Centro de Tecnologia Acadêmica (CTA) da UFRGS para prototipação da placa de circuito impresso com os equipamentos do laboratório deles[22]. A placa impressa corresponde a essas conexões que precisam ser feitas via *hardware*, visando aumento da qualidade do sistema desenvolvido e garantia de seu funcionamento.

3.5 Impressão 3D

Um último mecanismo utilizado para conclusão deste projeto, que também dá enfoque na qualidade e garantia de funcionamento do protótipo desenvolvido, é a elaboração de uma caixa para armazenamento do *hardware* e circuito Arduino. A importância dessa etapa vai ao encontro da ideia de redução de problemas como choques mecânicos ou cabos que possam ser desconectados de forma não proposital.

Para que a caixa de armazenamento do *hardware* cumpra com essas funcionalidades exigidas, é necessário que ela tenha dimensões precisas e aberturas em pontos estratégicos para passagem dos cabos conectores. Partindo

desse ponto, foi executado um projeto paralelo em parceria com a Enfitec Júnior (empresa júnior do curso de Engenharia Física) para construção dessa caixa por meio de uma impressão tridimensional (I3D), pois assim se torna possível definir todas suas dimensões e aberturas necessárias aos cabos precisamente.

4. RESULTADOS E DISCUSSÃO

Após a definição da metodologia e mecanismos utilizados para alcançar os objetivos finais deste projeto, deu-se início então a uma série de atividades para elaboração do projeto. Cada uma das etapas seguidas ao longo do processo de execução será descrita nesta seção.

4.1 Testes iniciais

A primeira etapa após a definição da metodologia de trabalho foi a aquisição da placa de circuito Arduino, para dar início à programação na sua IDE de origem. O código preliminar elaborado tinha o objetivo de receber duas variáveis correspondentes aos valores de fluxo (em sccm) e uma terceira variável correspondente ao tempo (em minutos) desejado do experimento. Ambas eram digitadas manualmente no código, apenas para viabilizar os testes iniciais de funcionamento com o Arduino, sendo utilizadas posteriormente ao longo do programa.

A primeira função desse código inicial tinha o objetivo de testar se os valores de fluxo estavam dentro dos limites permitidos pelos MFCs, considerando zero como mínimo e 500 sccm como valor máximo. Caso ambos os valores estivessem dentro desses limites, então uma nova função de envio dos parâmetros aos MFCs seria chamada.

Essa segunda função foi criada com o objetivo de converter os valores de fluxo da unidade de sccm para um valor correspondente em volts, considerando uma rampa de conversão de 0 a 500 sccm para 0 a 5 V, e enviar essas tensões correspondentes às portas do Arduino por meio de programação. O tempo selecionado para o experimento é utilizado para controlar quando esse envio será zerado, ou seja, após quanto tempo as tensões enviadas às portas do Arduino voltarão a ser valores nulos, sinalizando o fim do experimento.

Como o Arduino utilizado não tem portas de escrita analógicas (as portas analógicas existentes são apenas para leitura), foi necessário utilizar de programação específica para contornar esse problema e viabilizar o uso de portas digitais ao enviar os valores de tensão para o Arduino. Além de ajustes na programação, é necessário utilizar filtros específicos na saída dessas portas digitais

que garantam a conversão para valores analógicos, sendo construídos em uma *protoboard* para facilitar os testes.

A programação e os filtros utilizados na adaptação das portas digitais para escrita de valores analógicos serão explicitados em detalhes nas seções seguintes, quando o código e *hardware* finais serão apresentados.

Como grande parte do trabalho foi executado de casa devido à pandemia, foi realizado um teste com dois LEDs ligados nas saídas que iriam para os MFCs. O objetivo era comprovar, por meio da variação da intensidade no brilho dos LEDs, que as alterações de valores de fluxo digitadas no programa estavam sendo enviadas corretamente ao *hardware*. Assim, relaciona-se que uma maior intensidade no brilho do LED corresponde a uma maior tensão recebida por ele. A foto abaixo demonstra a conexão feita para possibilitar esses testes caseiros:

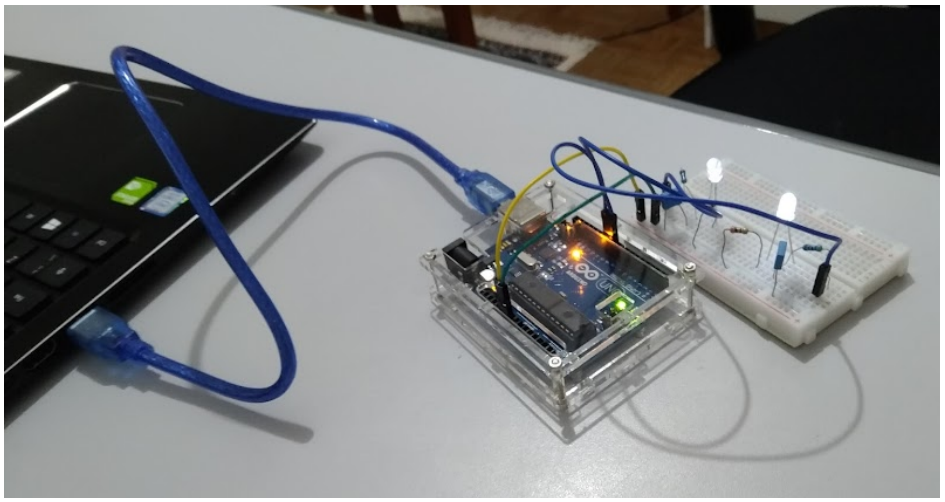


Figura 9 – Foto da construção caseira do sistema para testes iniciais.

Após obter essa construção preliminar e comprovação de funcionamento, foi adquirido um conector tipo D de 15 pinos com uma saída fêmea (para conexão no MFC) e outra com rabicho, ou seja, com os 15 fios soltos, para iniciar os testes com os MFCs. Esses fios soltos foram utilizados para conexões na protoboard, de forma que poderiam ser facilmente trocados de ligação para execução dos testes no laboratório.

Abaixo tem-se uma foto de como essas ligações foram feitas no laboratório:

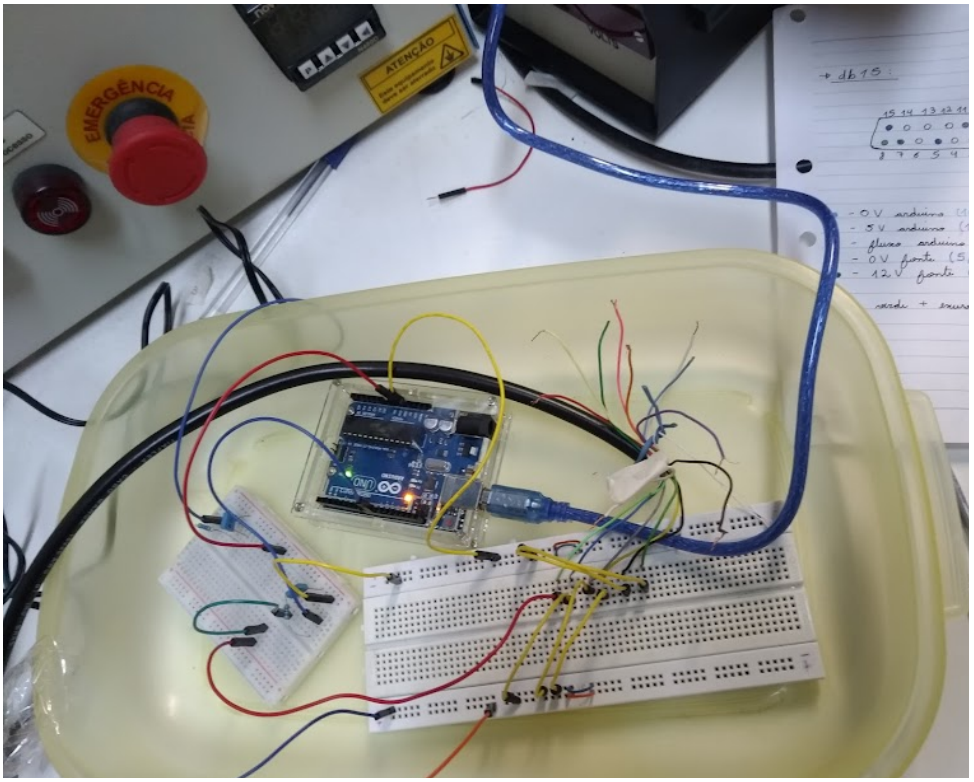


Figura 10 – Foto do sistema instalado no laboratório para testes iniciais.

Para facilitar os testes, o sistema foi conectado a apenas um MFC, visando também o não desperdício dos gases utilizados. Houve muita dificuldade nessa etapa, pois a configuração fornecida pelo manual dos MFCs para utilizá-los no modo remoto não era descrita claramente.

Foi necessário fazer várias trocas nas conexões desses 15 fios que ligam o sistema do Arduino ao MFC, e também nos *jumpers* internos do próprio controlador de fluxo, até encontrar a combinação que funcionasse corretamente. Após várias tentativas, o MFC utilizado reconheceu a entrada do valor do fluxo enviada de forma remota e agiu de acordo com o esperado inicialmente. As ligações finais do conector db15 serão explicadas na seção correspondente a ele.

Com o sistema preliminar viabilizado e aprovado nos testes, deu-se início a fase de elaboração do protótipo final, para garantir maior qualidade no projeto a ser entregue.

4.2 Código da interface

O primeiro item a ser planejado e executado após o funcionamento do sistema preliminar foi a interface visual no computador. Essa é uma das partes mais

importantes do projeto, pois é o que garantirá um dos objetivos finais dele: facilitar o manuseio pelo usuário.

A interface foi programada no *software* Processing e sua versão final ficou da seguinte maneira:

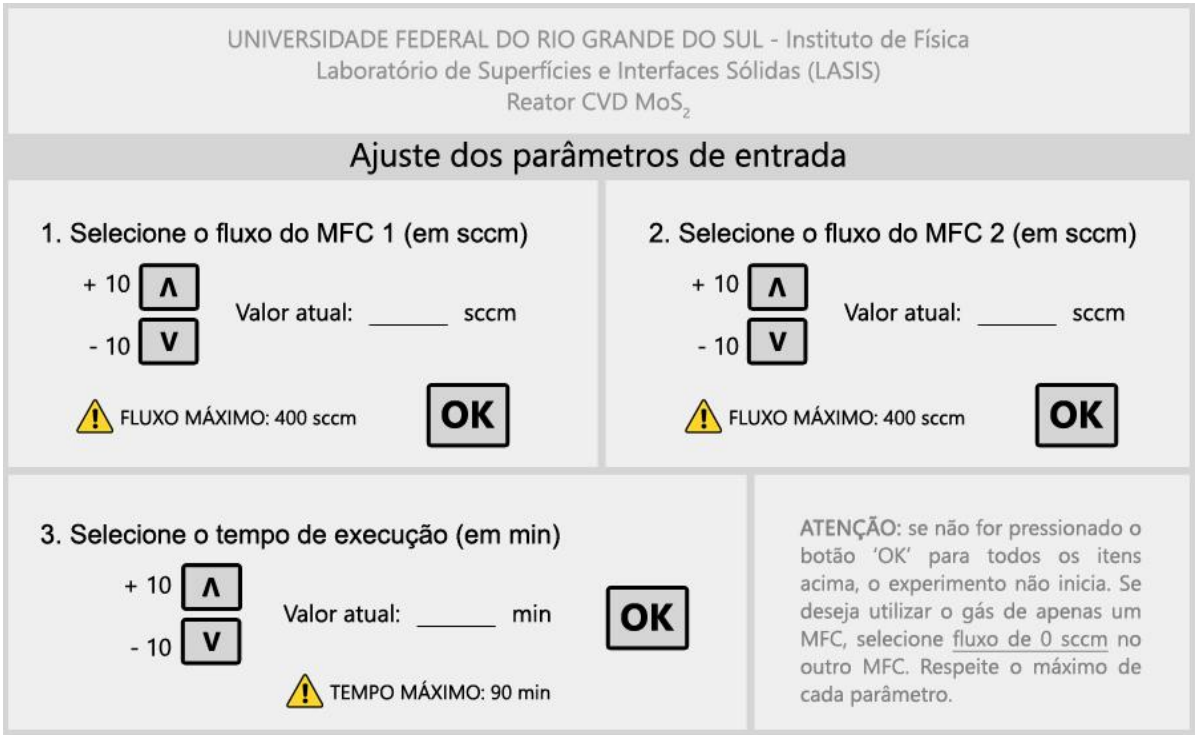


Figura 11 – Imagem final da interface programada com o Processing.

Os valores máximos de 400 sccm para os fluxos e de 90 minutos para o tempo do experimento foram definidos pois, em laboratório, nunca são utilizados parâmetros maiores do que esses no reator CVD para síntese de MoS_2, além de diminuir a complexidade da passagem dos valores do Processing ao Arduino via porta serial.

A interface programada foi pensada para ser o mais intuitiva e prática possível, contando com botões que aumentam ou diminuem cada parâmetro em 10 unidades, do valor mínimo ao máximo que pode ser selecionado. Cada parâmetro é selecionado apenas uma vez e muda automaticamente na tela. Quando chegar no valor desejado, apenas é necessário pressionar o botão “OK” do parâmetro, executando esse mesmo passo para os outros dois.

A estrutura do código escrito no Processing deve conter as declarações iniciais, uma função correspondente a ações que ocorrem apenas uma vez

(chamada de *setup*) e uma função de *loop* (chamada de *draw*, em referência ao desenho gerado na tela), que ocorre repetidamente. O *loop* não foi utilizado nesse caso, pois toda a programação necessária foi executada por meio de uma função específica que programa os botões, já que tudo que ocorre na tela é direcionado por cliques neles.

A programação que gera a interface elaborada consistiu de três blocos básicos para funcionamento: as configurações iniciais, programação dos botões e envio dos parâmetros ao Arduino (também parte da programação dos botões). O código na íntegra se encontra no apêndice deste documento, mas seus blocos básicos serão explicados a seguir.

4.2.1 Configurações iniciais

Para que o Processing permita a passagem de informações ao Arduino, é necessário declarar no início do documento uma série de códigos que o farão se conectar às portas seriais do Arduino, que deve estar conectado no computador. O recorte abaixo indica a parte do código referente a essa configuração inicial:

```
17 /********************************************* CONFIGURAÇÃO SERIAL ******
18 //
19 import processing.serial.*;
20 Serial myPort;
21 //
22 /********************************************* CONFIGURAÇÃO INICIAL *****
23 //
24 void setup() {
25   myPort = new Serial(this, Serial.list()[0], 9600); //conexão com a porta
26   //programação da imagem da janela
27   size(897,549);
28   PImage bg;
29   bg = loadImage("bginterface.jpg");
30   background(bg);
31   //programação dos textos
32   textSize(20);
33   textAlign(CENTER);
34   fill(0);
35   text(0, 303, 240);
36   text(0, 760, 240);
37   text(0, 340, 466);
38 }
39 //
```

Figura 12 – Recorte do código da interface referente à configuração inicial.

Nas linhas 19 e 20 tem-se a inicialização do contato com o Arduino fora dos *loops* essenciais do programa, enquanto que na linha 25 é feita a conexão com a porta serial escolhida (neste caso foi a porta digital 0, pois é a que funciona para recebimento de informações pelo Arduino). É selecionada também a *baud rate* de 9600, valor que deve estar igual ao configurado no Arduino e que se refere à taxa de transferência em bits por segundo[24].

O restante da configuração inicial é referente à imagem básica que veremos na tela. Como esse *software* processa o código digitado linha após linha, o primeiro item a ser inserido é o plano de fundo, que foi desenhado no *software* Photoshop CS3. Após, é feita a configuração de tamanho, fonte e cor dos textos que serão alterados na tela, sobrepondo essa imagem de fundo.

O único item da tela que não faz parte da imagem de fundo é o valor dos parâmetros selecionados, pois ele precisa variar dinamicamente conforme o usuário pressiona os botões. Todo o restante da interface consiste na imagem de fundo produzida. Preferiu-se gerar a imagem de fundo com todo o conteúdo fixo, apesar da possibilidade de ser todo programado pelo *software*, pois o código se torna mais sucinto.

4.2.2 Programação dos botões

Existe uma função no Processing chamada *mouseClicked* na qual é possível detectar cliques feitos na tela dentro de uma determinada área. Essa área é contabilizada em *pixels*, com o canto superior esquerdo da tela sendo o zero e crescendo no sentido do canto inferior direito. Sabendo as posições inicial e final nos eixos (X e Y) de um certo desenho na tela, podemos configurá-lo para se tornar um botão, ou seja, executar uma ação quando for detectado um clique dentro desta área.

A imagem abaixo traz um recorte do código que exemplifica a programação de um botão:

```
48 void mouseClicked() {
49   if (mouseX > 103 && mouseX < 147 && mouseY > 196 && mouseY < 230)
50   {
51     fluxo1 = fluxo1 + 10; //se o botão de +10 foi pressionado, incrementa
52     fill(239,239,239);
53     stroke(239,239,239);
54     rect(275,224,60,16);
55     fill(0);
56     text(fluxo1, 303, 240); //mostra na tela o valor sendo incrementado
57   }
```

Figura 13 – Recorte do código da interface referente à programação de botões.

Nessa parte do código, há um laço do tipo *if* que executa o que está dentro dele caso seja detectado um clique na área pré-determinada, que é determinada pelos valores máximo e mínimo de cada eixo. A ação a ser executada nesse recorte indica que, se o botão foi pressionado, então o valor do fluxo será incrementado em 10 unidades, e o novo valor será mostrado na tela, sobrepondo o anterior.

Essa mesma lógica é utilizada para todos os botões de incremento ou decremento programados na interface, apenas adaptando a fórmula para cada caso.

4.2.3 Envio dos parâmetros

A última etapa do programa é justamente o envio dos parâmetros selecionados nos botões ao Arduino. Porém, a passagem dos argumentos não pode ser feita diretamente, pois a porta serial do Arduino só recebe um algarismo por vez, e os valores de fluxo podem ter até três algarismos cada, além do parâmetro de tempo que terá até dois algarismos. Assim, é necessário separar esses números nos algarismos que os compõem, totalizando oito variáveis.

Nos valores de fluxo, o isolamento dos algarismos da centena do número foi feito a partir de uma simples divisão por cem, obtendo-se assim um valor de zero a quatro. Já na casa da dezena, foi necessário um cálculo mais complexo, no qual obtêm-se o algarismo isolado por meio de duas funções diferentes: caso o valor do fluxo seja menor do que 100 sccm, então a dezena será obtida apenas dividindo-se por dez; entretanto, caso o valor do fluxo seja maior que 100 sccm, a dezena é obtida a partir do resto da divisão dele por cem, dividido novamente por dez. O algarismo das dezenas obtido em ambas as fórmulas retorna um valor que pode ir de zero a nove.

Não é necessário o envio do algarismo correspondente à unidade, pois ele sempre será zero, já que os parâmetros aumentam ou decrescem de dez unidades sempre. Por último, tem-se o parâmetro de tempo, no qual apenas foi necessário dividir o valor selecionado por dez, retornando valores de zero a nove, para simplificar sua passagem pela porta serial.

As fórmulas descritas estão na tabela abaixo, na qual "x" é o algarismo desejado e "p" é o valor inicial do parâmetro:

Tabela 2 – Fórmulas de separação de algarismos dos parâmetros de entrada.

Algarismo isolado	**Fórmula**
Centena do fluxo	$x = \frac{p}{100}$
Dezena do fluxo (p<100)	$x = \frac{p}{10}$
Dezena do fluxo (p>100)	$x = \frac{p\%100}{10}$
Dezena do tempo	$x = \frac{p}{10}$

Ao serem obtidos os algarismos isolados de cada parâmetro, é necessário enviá-los, um por vez, para a porta serial do Arduino. Entretanto, não é suficiente apenas passar esses números sequencialmente ao Arduino, pois ele pode se perder na detecção. Para enviar o algarismo sem a chance de erros, é necessário que o valor passado pela porta serial carregue informações sobre qual MFC corresponde àquele valor de fluxo, se é um valor correspondente a dezena ou centena ou até mesmo se é o parâmetro de tempo.

Sabe-se que o caractere passado pela porta serial pode ser um número, uma letra ou símbolo, e ainda há a diferenciação das letras caso sejam maiúsculas ou minúsculas. Tendo isso em vista, elaborou-se um código, por meio das letras do alfabeto e diferenciando-as em maiúsculas e minúsculas, para envio desses parâmetros sem que haja confusão no recebimento deles pelo Arduino.

O código elaborado está descrito na tabela abaixo:

Tabela 3 – Código de relação dos algarismos dos parâmetros.

Correspondência	Intervalo de referência	Tipo do caractere	Intervalo final
Centena do fluxo MFC 1	0 - 4	letra minúscula	a - e
Dezena do fluxo MFC 1	0 - 9	letra minúscula	f - o
Centena do fluxo MFC 2	0 - 4	letra maiúscula	A - E
Dezena do fluxo MFC 2	0 - 9	letra maiúscula	F - O
Dezena do tempo	0 - 9	número	0 - 9

O valor do tempo pode ser passado diretamente após isolamento do algarismo da dezena pois, com a conversão dos outros algarismos correspondentes aos fluxos em letras, não há chance para erros na passagem dos parâmetros.

Ao analisar esse envio de informações, pode-se entender o motivo pelo qual foi necessário definir valores máximos para os parâmetros, pois caso fossem maiores, seria necessário adição de caracteres até completar a correspondência com eles. Outro motivo que impacta nos caracteres escolhidos é o intervalo de dez unidades de acréscimo ou decréscimo nos parâmetros, pois se tornaria inviável a passagem de valores com intervalos de uma unidade apenas, devido a quantidade de caracteres a serem convertidos.

Um último problema que pode ocorrer no envio dos parâmetros é a seleção de valores maiores que o permitido, já que não há impedimento da seleção deles na interface, como por exemplo a escolha de um fluxo acima de 600 sccm ou um tempo de 150 minutos. Para tanto, foi inserido também um trecho no código que promove a alteração de qualquer valor acima do permitido, convertendo-o no valor máximo daquele parâmetro. Assim, caso seja selecionado um fluxo maior que 400 sccm, ele será forçado a voltar a 400, e caso seja selecionado um tempo maior que 90 minutos, ele será forçado a voltar a 90.

Por fim, esses algarismos são passados para o Arduino por meio de um trecho do código que escreve o caractere na porta serial assim que os botões de “OK” são pressionados na tela, utilizando a mesma função *mouseClicked*. Ao ser

pressionado o botão de “OK” para o MFC 1, seus dois parâmetros são passados em sequência, ocorrendo o mesmo para o MFC 2 e para o tempo. A sequência a ser passada não interfere, pois o Arduino também estará programado para interpretar corretamente o recebimento desses caracteres.

4.3 Código do Arduino

O código escrito na IDE Arduino foi a primeira etapa iniciada neste projeto, porém, só foi concluído após a fase de programação da interface visual. Isso ocorreu pois a interface criada no Processing modifica a forma que a IDE Arduino recebe os parâmetros escolhidos pelo usuário e que serão tratados ao longo do código, tendo sido feitas modificações significativas nele após a conclusão do código no Processing.

Todos os programas escritos na IDE Arduino dependem de duas funções principais: a de *setup*, que contém as configurações iniciais que só serão executadas uma vez ao longo do código, e a de *loop*, que irá rodar infinitamente enquanto o Arduino estiver ligado. Neste projeto, a função inicial de *setup* apenas configura as portas a serem utilizadas como saída no Arduino e inicializa a conexão serial, enquanto que o *loop* engloba quatro funções criadas que são essenciais ao projeto.

O código completo se encontra no apêndice, entretanto, na imagem abaixo é trazido um recorte dele que demonstra a sequência de funções que foram criadas internamente ao *loop* principal do programa:

```
//
void loop()
{
  interface(); //chama função que recebe valores do Processing
  conversao(); //chama função que converte os algarismos recebidos
  start = teste(); //chama função que testa os limites do fluxo
  if (start == 1) //início do experimento
  {
    envio(); //chama função que envia valores aos MFCs
  }
}
//
```

Figura 14 – Recorte do código do Arduino referente a ordem das funções.

4.3.1 Função “interface”

Essa função tem como propósito receber do Processing, por meio do código de caracteres criado previamente, os parâmetros que foram selecionados na interface pelo usuário. Ela é constituída por um teste que indica que, se a porta serial estiver detectando o recebimento de um caractere, o programa irá compará-lo com todos os quarenta caracteres criados até que haja uma igualdade.

Quando o programa detectar uma igualdade de caracteres, ele o converterá novamente para um algarismo, de acordo com o código explicitado na tabela 3. Ou seja, é feito o processo inverso que ocorre no Processing: nesse caso, é recebido o caractere codificado e ele é transformado novamente em um algarismo que compõe os números dos parâmetros escolhidos pelo usuário.

Na imagem abaixo temos um recorte do código que demonstra parte das conversões feitas nessa função, relativo ao algarismo da centena do fluxo do MFC 1:

```
void interface()
{
  do
  {
    if (Serial.available()) //testa se a comunicação serial está disponível
    {
      switch(Serial.read()) //transforma caractere recebido pela porta serial em algarismos
      {
        case 'a':
          ceml = 0;
          flagceml = 1;
          break;
        case 'b':
          ceml = 1;
          flagceml = 1;
          break;
        case 'c':
          ceml = 2;
          flagceml = 1;
          break;
        case 'd':
          ceml = 3;
          flagceml = 1;
          break;
        case 'e':
          ceml = 4;
          flagceml = 1;
          break;
```

Figura 15 – Recorte do código do Arduino referente a valores da porta serial.

Por fim, essa função ocorre em *loop* até que seja detectado o recebimento de todos os algarismos necessários para compor os parâmetros, por meio de quatro

variáveis do tipo *flag*, criadas uma para cada algarismo: das centenas e dezenas dos dois fluxos, e da dezena do parâmetro de tempo. Assim, são criadas cinco variáveis, uma correspondente a cada algarismo que é necessário ser recebido pelo Arduino, e o programa só dá continuidade às funções seguintes quando todas essas *flags* trocarem de zero para um, indicando recebimento de todos os algarismos essenciais para reconstrução dos parâmetros.

4.3.2 Função “conversão”

A função que dá sequência ao código após recebimento e decodificação dos algarismos é justamente a função que irá reconstruir os parâmetros ao valor inicial selecionado pelo usuário. Os cálculos executados nesta função estão demonstrados na imagem abaixo:

```
//
void conversao() //converte os algarismos recebidos
{
  fluxo1 = ((cem1)*100) + ((dez1)*10);
  fluxo2 = ((cem2)*100) + ((dez2)*10);
  tempo = t*10;
}
//
```

Figura 16 – Recorte do código do Arduino referente à reconstrução dos valores.

Para ambos os fluxos, é multiplicado o algarismo da centena por cem e o da dezena por dez, sendo somados para resultar novamente em um valor que vai de 0 a 400 sccm. Já para o parâmetro de tempo, o valor é apenas multiplicado por dez, resultando novamente em um parâmetro que está dentro dos limites de 0 a 90 minutos.

4.3.3 Função “teste”

Após a reconstituição dos parâmetros, os valores de fluxo passam por uma função de teste que garante que só será feito o envio dos parâmetros ao Arduino caso eles sejam ambos maiores que zero. Na prática, isso não se faz necessário, já que o programa só entra nessa função quando já recebeu os valores selecionados pelo usuário. Porém, é importante manter esse teste caso ocorram erros de seleção

na interface visual e uma garantia extra de correto funcionamento do código do Arduino.

Ao garantir que ambos os fluxos são maiores que zero, eles passam por uma conversão para os limites inferior e superior utilizado nas portas PWM do Arduino. É necessário realizar essa conversão pois o Arduino não possui portas de escrita analógicas, ou seja, em teoria não é possível enviar valores analógicos do código ao microcontrolador, apenas o caminho inverso. Porém, com algumas adaptações no circuito e no código escrito, torna-se viável o uso de portas digitais para essa função.

As portas digitais passíveis de serem utilizadas com PWM (*pulse width modulation*, que significa uma modulação da largura de pulso) são as sinalizadas com o sinal gráfico til “~” no design do circuito Arduino. Nessas portas, o PWM funciona a partir do envio de um *duty cycle*, uma onda quadrada com pulsos determinados por valores de 0 a 255. Um valor de pulso nulo indica que a tensão de saída será sempre nula também, enquanto que um valor de pulso de 255 indica que a tensão de saída será sempre de 5 V. Valores intermediários de pulsos indicam a porcentagem do período em que o ciclo estará no estado ligado de 5 V[25].

Na imagem abaixo é demonstrada essa onda quadrada com diferentes valores de pulso:

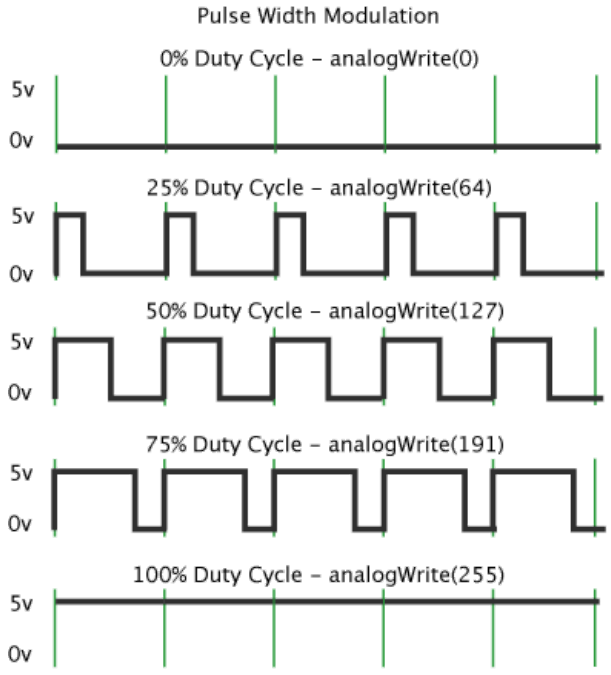


Figura 17 – D*uty cycle* para diferentes valores do pulso. Retirado de [25].

A função *analogWrite()* demonstrada na imagem é a que será utilizada para escrita desses valores no Arduino posteriormente na função de envio, e o valor que é utilizado nela é uma conversão dos valores de fluxo (que podem ser de 0 a 500 sccm) para valores de ciclo do PWM (de 0 a 255), em escala diretamente proporcional[26]. O valor máximo utilizado para o fluxo nesse caso é de 500 sccm, apesar de não ser possível enviar valores acima de 400 pela interface, pois facilita a conversão direta para o ciclo do PWM e da tensão de saída.

Esse cálculo da tensão de saída (V_{out}) correspondente se dá pela seguinte equação, retirada de [27], na qual o *duty cycle* é a porcentagem correspondente ao período em que o ciclo está no estado ligado e *Vcc* é a tensão de alimentação do Arduino (5 V):

$$Vout = \frac{duty\ cycle\ .Vcc}{100} \tag{3}$$

Assim, sabe-se a tensão de saída analógica na porta digital a partir do PWM, já que a modulação de valores intermediários de tensão ocorre com a repetição rápida desse padrão de onda na porta digital de saída, gerando sinais analógicos estáveis, e a adição de um filtro capacitivo na saída do Arduino. As adaptações relacionadas ao *hardware* serão explicadas posteriormente.

Sabendo disso, a função de teste no código foi escrita da seguinte forma:

```
//
int teste() //verificação dos limites de fluxo
{
  if (fluxo1 > 0 || fluxo2 > 0) //testa se os valores são maiores do que zero
  {
    pwm1 = map (fluxo1, 0, 500, 0, 255); //conversão dos limites de fluxo
    pwm2 = map (fluxo2, 0, 500, 0, 255); //conversão dos limites de fluxo
    return 1; //retorna 1 se pelo menos um dos valores de fluxo for > 0
  }
  else
  {
    return 0; //retorna 0 se ambos os fluxos estiverem zerados
  }
}
```

Figura 18 – Recorte do código do Arduino referente à função de teste.

Caso ambos fluxos sejam não nulos, a função matemática *map* realiza uma regra de três para converter o valor dos fluxos e é enviado um retorno de 1 da função de teste. Ela retorna 0 se ambos os fluxos estiverem zerados.

4.3.4 Função “envio”

A última função do código tem o propósito de escrever os valores nas portas digitais caso a função de teste retorne 1, como na imagem abaixo:

```
//
void envio()
{
  t_envio = tempo*60000; //conversão do tempo de minutos para milissegundos
  t_atual = millis();
  while(t_atual <= t_envio) //envia fluxos enquanto o tempo atual > selecionado
  {
    analogWrite(mfc1, pwm1); //enviando valor selecionado de fluxo ao MFC 1
    analogWrite(mfc2, pwm2); //enviando valor selecionado de fluxo ao MFC 2
    t_atual = millis();
  }
  if(t_atual >= t_envio)
  {
    analogWrite(mfc1, 0); //zerando valor de fluxo enviado ao MFC 1
    analogWrite(mfc2, 0); //zerando valor de fluxo enviado ao MFC 2
    start = 0; //zera flag que controla início de experimento
  }
}
//
```

Figura 19 – Recorte do código do Arduino referente à função de envio.

A primeira ação a ser tomada é a conversão do tempo do experimento selecionado pelo usuário na interface para milissegundos, pois assim se torna viável compará-lo com o tempo que o programa está rodando, com a função *millis()*. Essa função apenas retorna o tempo que o programa já levou pra executar o código, em milissegundos, quando chamada[28].

Assim, o código foi escrito de maneira que os fluxos sejam enviados constantemente às portas do Arduino pela função *analogWrite()* enquanto que o tempo que o programa já rodou seja menor que o tempo selecionado pelo usuário. Quando esse valor se tornar maior, então será enviado fluxo nulo aos MFCs pelas portas do Arduino, sinalizando o fim do programa e do experimento.

4.4 Filtragem do sinal

Um item essencial para o correto funcionamento desse sistema, que não tem a possibilidade de fazer parte da programação, é a existência de filtros capacitivos

nas saídas PWM do Arduino. Esses filtros são colocados antes dos sinais serem enviados aos MFCs, para garantia de recebimento correto do sinal modulado pelo código.

São utilizados filtros capacitivos, em especial o filtro do tipo passa-baixa, pois ele funciona como um atenuador dos pulsos gerados pelo PWM nas portas digitais do Arduino, deixando passar apenas uma porcentagem do maior sinal recebido nas portas pelo *duty cycle*, que é sempre de 5 V. O circuito desse filtro é ligado como o da imagem abaixo:

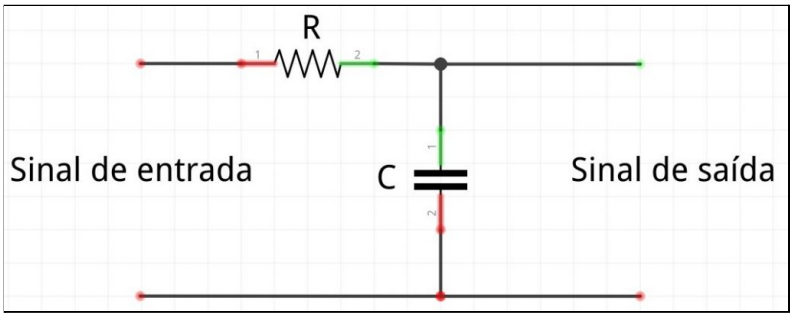


Figura 20 – Ligações de um filtro capacitivo passa-baixa. Retirado de [29].

Quando são utilizados sinais alternados na entrada do filtro passa-baixa, o capacitor apresenta uma certa resistência à passagem da tensão que é chamada de reatância capacitiva (*Xc*), dada pela equação abaixo, na qual *f* é a frequência do sinal e *C* é a capacitância do capacitor utilizado:

$$Xc = \frac{1}{2\pi fC} \tag{4}$$

Essa reatância capacitiva tende a ser alta para sinais de baixa frequência, resultando numa alta resistência à passagem do sinal pelo caminho do capacitor. Somado a isso, tem-se que a corrente elétrica terá maior facilidade de passar pelo caminho com menor resistência que, no caso do filtro utilizado, é o caminho de interesse que é direcionado aos MFCs, recebendo apenas frequências abaixo da frequência de corte[29].

Em um filtro passa-baixa, o resistor utilizado no caminho serve apenas para a definição dessa frequência de corte que divide o que serão consideradas altas ou baixas frequências, definindo a partir de qual valor elas serão atenuadas. O cálculo da frequência de corte é derivado da equação (4), se considerarmos que *Xc* é uma resistência e utilizarmos o valor do resistor a ser colocado no circuito do filtro[29].

No caso deste projeto, a frequência de corte deve ser a mesma frequência utilizada nas portas do Arduino, pois é ele que determina o período de cada ciclo PWM, sendo geralmente o valor de 340 Hz, exceto para algumas portas que não foram utilizadas neste trabalho. A partir dessa frequência, calculou-se os valores necessários para o resistor e capacitor do filtro, procurando adaptá-los para valores comerciais existentes para esses componentes. Assim, o valor da resistência do resistor ficou em 4,7 kΩ e da capacitância do capacitor 100nF.

4.5 Conexões dos MFCs

O sinal modulado pelo Arduino e que passa pelos filtros é de grande importância, entretanto ele é apenas um dos sinais que deve ser enviado aos MFCs. Para correta interpretação dos valores de fluxo enviados por meio de tensão, os MFCs devem receber tanto a referência de tensão mínima e máxima utilizadas pelo Arduino (0 e 5 V), quanto uma tensão de 12 V para sua própria alimentação.

A alimentação de 12 V do MFC é realizada diretamente pela entrada db15 quando ele está configurado para ser utilizado no modo local. Porém, quando é alterado para o modo remoto, essa alimentação deve chegar em algum dos pinos de entrada desse conector db15. A configuração a ser utilizada, fornecida pela marca do produto no manual, está descrita na figura 7.

Levando em conta esses pontos, foram testadas e definidas as tensões a serem colocadas em cada um dos pinos do conector db15, e a solução final que funcionou corretamente nos testes no laboratório é a descrita na tabela abaixo:

Tabela 4 – Configuração da pinagem utilizada no conector db15.

Pinos db15	5 e 7	8	10 e 11	15
Sinal (V)	0 e 12 V	fluxo	0 e 5 V	0 V

Nos pinos 5 e 7, foi utilizada uma fonte chaveada de 12 V para fornecer essa diferença de potencial, enquanto que nos pinos 10 e 11 os valores são as referências de *ground* (terra) e tensão máxima fornecidas pelo Arduino, para comparação com o valor recebido no pino 8, que será variável entre 0 e 5 V. É importante o envio dessa referência aos MFCs justamente para que, caso o valor recebido no pino 8 varie um pouco do que deveria ser, ele irá modular perante os

limites máximo e mínimo recebidos nas portas 10 e 11 que também terão a mesma variação, não acarretando problemas na conversão de tensão para fluxo que é feita dentro dos MFCs.

Outro item que precisou ser alterado nos MFCs para seu correto funcionamento nos testes foi a configuração dos *jumpers* internos deles, que é dada pela figura 6 também retirada do manual. Nesse caso, foi necessário alterar os jumpers da terceira coluna, que estavam configurados para 11 e 12 e foram colocados em 10 e 11. Essa movimentação se refere à troca do modo local para modo remoto.

Na teoria, essa seria a única alteração necessária, considerando a tabela demonstrada na figura 6. Porém, durante os testes no laboratório, o MFC não dava continuidade à modulação do fluxo internamente, mostrando sinal de erro. A partir disso e após muitas tentativas de alteração até mesmo nas conexões do db15, foi alterada a última coluna de *jumpers*, de 13 - 14 para 14 - 15, desligando a função *cutoff*.

Não era cogitada uma alteração desse tipo no planejamento do projeto, pois o MFC deve funcionar da mesma forma com ela ligada. Entretanto, essa função estava interrompendo o seguimento do experimento justamente porque o valor de fluxo recebido pelos MFCs diferia muito do valor esperado para aquela tensão. Após desligar essa função e o MFC funcionar como o esperado remotamente, percebeu-se que o valor mostrado na tela dele estava consideravelmente abaixo do que deveria, em torno de 7% do valor que era digitado no código.

Essas diferenças provavelmente se deram pela precariedade do teste, já que foi utilizado um cabo db15 com uma das pontas abertas e decapada conectada diretamente na protoboard, onde foram feitas as outras conexões com a fonte de 12 V e o Arduino. Esses acontecimentos implicam a necessidade de uma possível calibração ao final do projeto.

4.6 Hardware

Grande parte das ações necessárias a este projeto são executadas nos códigos escritos nos softwares. Apesar disso, é necessária a criação de um circuito adicional simples para correta distribuição dos sinais que serão enviados aos MFCs.

Um dos itens que deve estar presente no *hardware* são os filtros passa-baixa, já explicados previamente, nas saídas do Arduino que contém a tensão referente

aos fluxos selecionados. Além disso, as conexões que alimentam os MFCs com tensão de 12 V e que enviam a ele a referência de voltagem do Arduino (0 a 5 V) também precisam passar por um circuito.

Dessa forma, foi elaborado inicialmente o circuito em protoboard para testes no laboratório, com as seguintes conexões demonstradas no diagrama unilateral abaixo, construído no software Fritzing[30]:

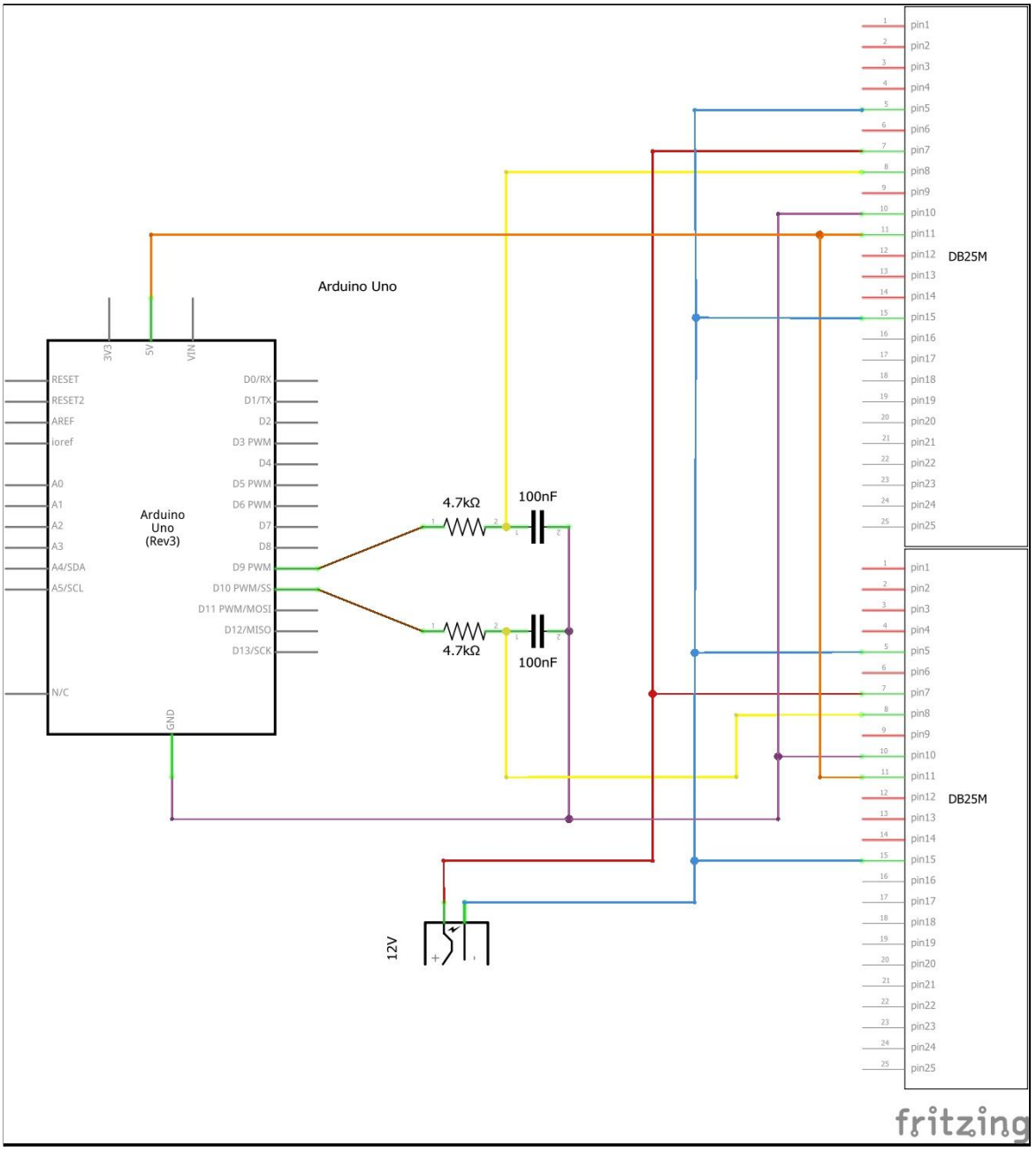


Figura 21 – Diagrama unilateral do *hardware*.

Na imagem, os conectores db25 demonstrados não são os corretamente utilizados pois o Fritzing não continha as entradas de 15 pinos, porém o diagrama não perde sua função para facilitar o entendimento do *hardware*. A partir desse diagrama, é possível ter uma visão ampla de como funciona a distribuição dos sinais já explicitados.

Após os testes no laboratório e a confirmação de funcionamento desse circuito, foi então encomendada uma placa de circuito impresso (PCI) com a Enfitec Júnior, empresa júnior do curso de Engenharia Física. O *sketch* unilateral acima foi transformado e adaptado em um esquemático para impressão da placa de circuito utilizando os equipamentos e suporte do CTA, Centro de Tecnologia Acadêmica.

O desenho é inicialmente impresso em papel *transfer* de forma espelhada em uma impressora a *laser*, após isso ele é passado para uma chapa a partir de uma prensa térmica e cortado com uma guilhotina do tamanho ideal para o uso. Então, a placa já com o desenho das conexões é imersa em percloreto de ferro por cerca de 20 a 25 minutos, sendo posteriormente limpa e perfurada conforme necessário para utilização de parafusos. Por fim, os componentes são soldados com estanho na posição definida na placa.

O desenho da placa de circuito pode ser encontrado no apêndice, e as conexões da placa impressa ficaram da seguinte maneira:

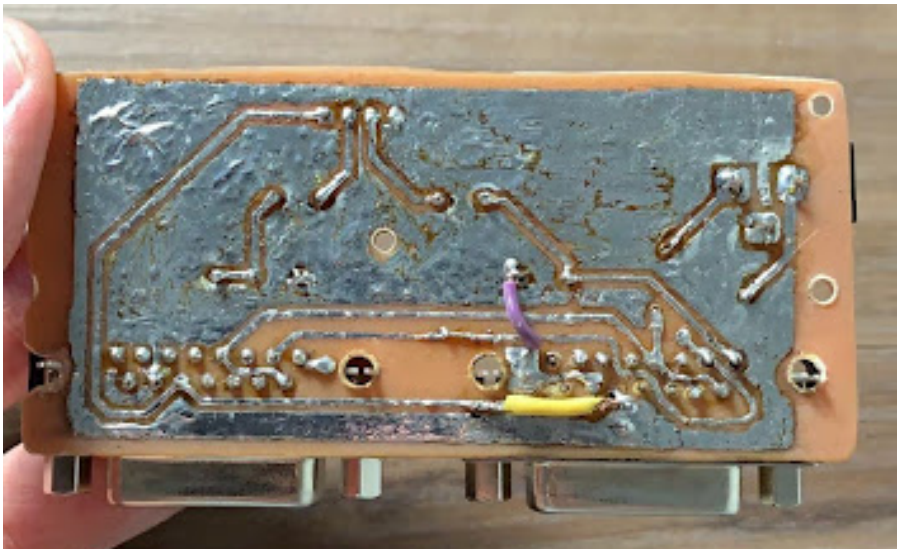


Figura 22 – Foto das conexões da placa de circuito impresso.

Pode-se perceber que a placa é feita de forma artesanal para baratear os custos com o projeto, o que acarretou em algumas desconexões enquanto o sistema era manuseado, já que as linhas de conexões acabavam raspando e perdendo contato. Além disso, os componentes soldados ficaram pouco firmes, exigindo maior cuidado no manuseio e inserção dos cabos. A imagem abaixo demonstra a visão superior da placa impressa, com os componentes soldados:

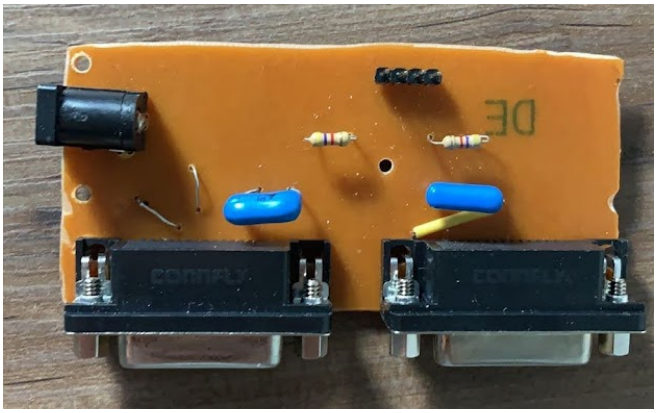


Figura 23 – Foto dos componentes soldados na placa de circuito impresso.

4.7 Caixa em impressão 3D

Por conter muitos componentes e cabos conectados entre o circuito impresso e o Arduino, foi desenvolvida, também com a Enfitec Júnior, uma caixa para ser feita na impressora 3D da empresa júnior. Essa caixa foi desenvolvida de modo a comportar a placa de circuito impresso e o Arduino, além de haver furos apenas nas regiões em que é necessário a saída dos cabos. Os cabos que devem sair da caixa são os dois conectores db15 que ligam a PCI aos MFCs, o cabo da fonte de 12 V que se conecta na PCI para alimentar os MFCs por meio de um conjunto de pinos do conector db15 e o cabo USB que conecta o Arduino ao computador utilizado.

O desenho técnico feito pela Enfitec Júnior com as dimensões corretas de cada aresta e dos furos se encontra no apêndice deste documento, mas abaixo tem-se uma foto das peças já impressas:

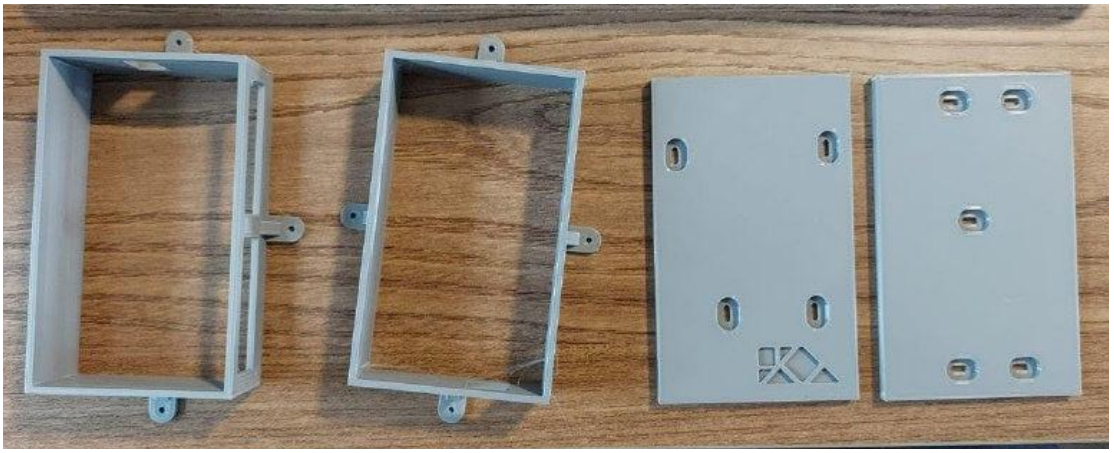


Figura 24 – Foto das peças impressas da caixa.

A caixa foi impressa em quatro partes separadamente, pois facilita a impressão 3D e diminui os custos dela. Assim, posteriormente as duas arestas e as tampas foram parafusadas para manter as peças unidas e as abas centrais coladas. Outra vantagem da caixa poder ser desmontada é caso haja necessidade de manutenção do circuito, ou se algum componente queimar, pois se torna mais prático abrir a apenas uma das tampas da caixa.

A imagem abaixo demonstra um desenho tridimensional esquemático da caixa montada com todas as peças:

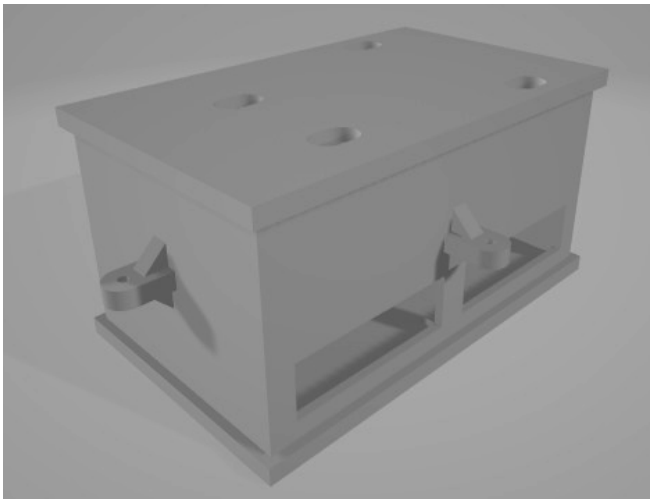


Figura 25 – Desenho tridimensional da caixa montada.

4.8 Computador

Por fim, o último item que completa toda a estrutura utilizada para funcionamento do projeto é um computador, o qual foi doado por um dos professores

colaboradores do Laboratório de Superfícies e Interfaces Sólidas para redução dos custos do projeto. Foi utilizado um *netbook* da marca HP devido a suas dimensões reduzidas e de fácil transporte caso necessário.

Ao ligar o *netbook*, é encontrado um arquivo de imagem na área de trabalho intitulado “Mini manual de instruções”, que é indicado para ser lido antes da utilização do sistema. Após ler o documento e realizar o passo a passo indicado, o usuário pode abrir a pasta que contém o arquivo executável, também encontrada na área de trabalho, que inicializa diretamente a interface visual, pronta para ser utilizada.

4.9 Mini manual de instruções

O mini manual de instruções elaborado é um guia completo e sucinto dos passos que devem ser seguidos para a correta utilização do sistema. Apesar do objetivo deste projeto ser justamente facilitar o manuseio pelo usuário, é necessário que haja um documento indicativo do uso correto caso surjam dúvidas, visando uma melhor gestão do conhecimento sobre o projeto.

O manual desenvolvido pode ser observado na imagem abaixo:

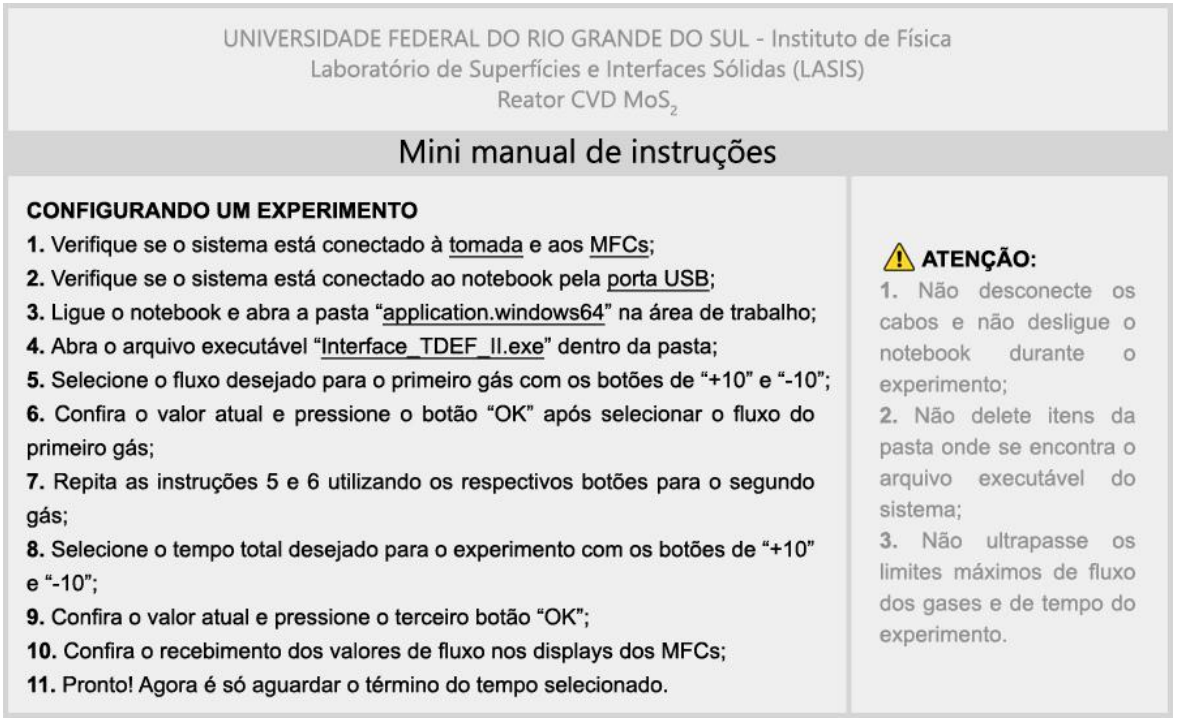


Figura 26 – Imagem do mini manual de instruções.

4.10 Funcionamento

Após a execução de todas as etapas supracitadas, há o momento de juntar todo o aparato desenvolvido e realizar novos testes no laboratório para aferir o funcionamento e realizar possíveis ajustes caso necessário. Como já mencionado, foi denotado ao longo dos testes que a placa de circuito impresso estava dando problemas de mau contato com frequência, necessitando de forma recorrente que as conexões fossem refeitas por meio de solda com estanho. Para tentar resolver o problema, foi adaptado um mecanismo para sustentação dos cabos, impedindo que eles fossem facilmente removidos ou movimentados, o que estava acarretando a desconexão de linhas do circuito.

A montagem final é apresentada na imagem abaixo:

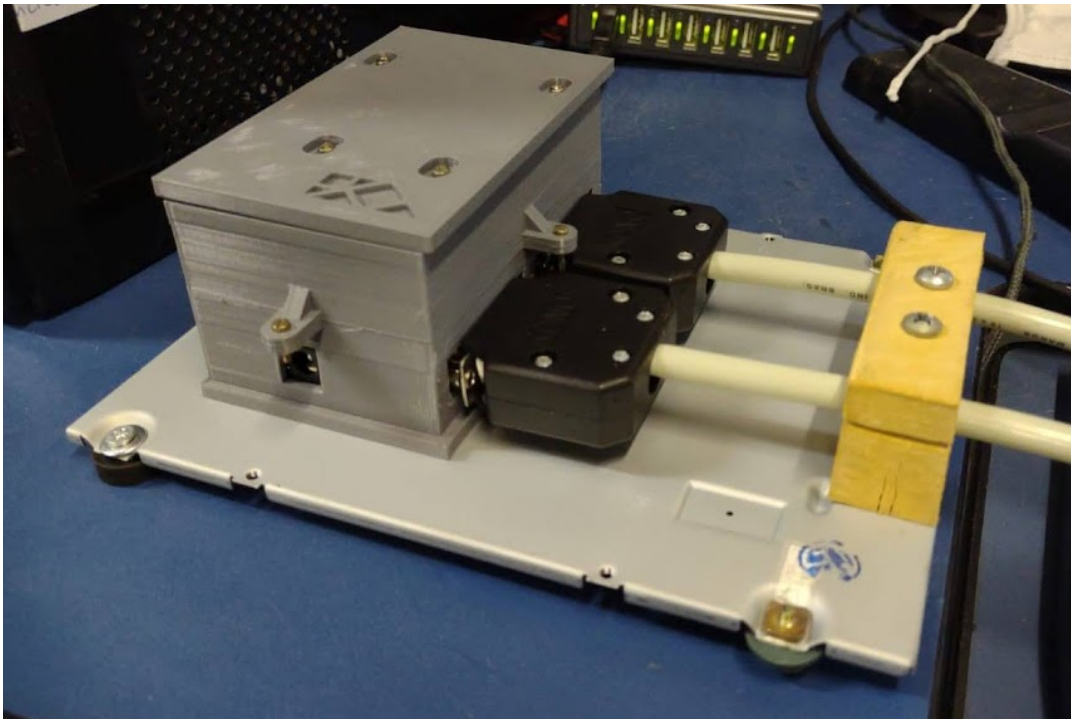


Figura 27 – Foto da estrutura do sistema montado com sustentação dos cabos.

O sistema foi testado e aprovado para ser instalado, realizando com sucesso a mesma função de quando foi verificado seu funcionamento no laboratório com a montagem em protoboard e sem a interface. Por fim, foi definido um momento para instalar todo o sistema no laboratório, realizar a calibração e síntese de uma amostra com ele, para posterior análise.

Entretanto, ainda não foi possível realizar sua instalação devido ao reator estar sendo utilizado naquele momento para outros projetos no laboratório e também a dificuldade de agendamento de novas datas devido a compromissos pessoais, ficando em aberto um momento futuro para instalação e calibração. Apesar disso, foi feito um registro do sistema no local onde ficará instalado no laboratório:

Figura 28 – Foto do sistema no local onde será instalado.

4.11 Custos

Para esse projeto, os itens que foram inicialmente utilizados para testes como uma *protoboard*, LEDs, resistores, capacitores e *jumpers*, não precisaram ser adquiridos, pois já os tinha em casa. Dessa forma, os únicos gastos no início do projeto foram com a placa de circuito Arduino, que era necessária para teste dos código criados, os cabos db15 e uma fonte de 12 V, ambos também para viabilização dos testes iniciais.

Posteriormente, foram necessários gastos para as melhorias no design do protótipo, o que inclui a placa de circuito impresso e a impressão 3D. Os valores gastos com cada item estão descritos na tabela abaixo, juntamente com o custo total para execução do projeto:

Tabela 5 – Custos dos componentes adquiridos para o projeto.

Material	**Custo**
Arduino Uno R3	R$ 92,90
2 Cabos db15 montados	R$ 146,00
Fonte DC 12 V chaveada	R$ 26,90
Placa de circuito impresso	R$ 100,00
Caixa em impressão 3D	R$ 93,00
TOTAL	R$ 458,80

5. CONCLUSÕES

Este projeto teve um prazo de execução longo em relação ao que era esperado inicialmente, tendo sido iniciadas as atividades práticas no início de 2021, devido principalmente às dificuldades impostas pela pandemia de Coronavírus. Além disso, como a ideia inicial do projeto era muito simples, não era esperado ir ao encontro de tanta dificuldade nos testes para atingir o correto funcionamento do sistema, problema que estendeu a previsão de entrega final. Essa dificuldade se deu especialmente devido ao manual do controlador de fluxo utilizado ser muito antigo e pouco específico, exigindo muitas mudanças no projeto ao longo dos testes para descobrir a combinação da pinagem de entrada do MFC que o faria funcionar como o esperado.

Apesar desses itens problemáticos, o resultado final foi satisfatório e superou as expectativas iniciais do projeto, que contava com a possibilidade de não ser viável a execução das partes finais, como por exemplo a impressão da placa de circuito e da caixa, já que eram itens opcionais e poderiam ser omitidos caso o prazo final estivesse ainda mais apertado ou não fosse acessível financeiramente e logisticamente. Outro item que superou as expectativas iniciais foi o desenvolvimento da interface visual via computador, já que exigia o contato e aprendizado com um software e mecanismos pouco explorados ao longo da graduação em Engenharia Física.

Através da conclusão de todos os itens relacionados ao projeto, tem-se a projeção de que os futuros usuários do sistema observem na prática o cumprimento dos objetivos esperados, relacionados ao aumento da praticidade e redução do tempo empregado para a seleção dos parâmetros desejados. Outro objetivo concluído é a contribuição para que o manuseio do reator CVD seja mais intuitivo, já que é necessário seguir os mesmos passos sempre para realizar uma síntese nele. Somado a isso, é dado que a automatização de processos aumenta a confiabilidade nos parâmetros definidos e resulta em um processo de produção com menos incertezas[31], agregando maior qualidade nas sínteses realizadas com o reator CVD.

Uma sugestão para projetos futuros é realizar a troca do microcontrolador para um modelo que tenha licença de uso comercial, explorando novas possibilidades para o projeto, como a implementação desse sistema em outros

reatores parecidos na universidade, por exemplo. Outra sugestão é o investimento financeiro em uma placa de circuito impresso de maior qualidade, que não terá suas conexões danificadas tão facilmente, reduzindo a possibilidade de manutenção futura.

A partir da pesquisa e desenvolvimento de projetos como este em ambiente universitário e percebendo a possibilidade de expansão dele para novos horizontes, é notável o quanto se tem de conhecimento agregado e maior independência na execução de projetos na universidade pública, fomentando um futuro necessário para a educação no Brasil.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] *O QUE É GLOBALIZAÇÃO? POLITIZE.* Disponível em: <https://www.politize.com.br/globalizacao-o-que-e/>. Acesso em: set. 2021

[2] FRANK, D. J., et al. Device Scaling Limits of Si MOSFETs and Their Application Dependencies. Proceedings of the IEEE, março 2001, nº 3, vol. 89, p. 259-288

[3] *DOMINE O TRANSISTOR DE JUNÇÃO BIPOLAR. EMBARCADOS.* Disponível em: <https://www.embarcados.com.br/domine-o-transistor-de-juncao-bipolar/>. Acesso em: set. 2021

[4] *O TRANSISTOR DE EFEITO DE CAMPO. ELETTOAMICI.* Disponível em: <https://www.elettroamici.org/pt/il-transistor-ad-effetto-di-campo/>. Acesso em: set. 2021

[5] SEDRA, A. S., SMITH, K. C. Microeletrônica. 4ª ed., São Paulo, Editora Pearson, 2000, p. 330

[6] BATHAEI, F. Z. Silicon device miniaturization and its effect on processing techniques. Microelectronics Journal, 1989, nº 4, vol. 20

[7] MOORE, G. E. Cramming more components onto integrated circuits. Electronics, 19 abril 1965, nº 8, vol. 38

[8] *MOORE'S LAW. WIKIPEDIA.* Disponível em: <https://en.wikipedia.org/wiki/Moore%27s_law>. Acesso em: set. 2021.

[9] D'AGOSTINO, F., QUERCIA, D. Short-Channel Effects in MOSFETs. Introduction to VLSI design (EECS 467), 11 dezembro 2000

[10] *MATERIAIS BIDIMENSIONAIS, QUASE INVISÍVEIS E DE ENORME POTENCIAL. FAPESP*. Disponível em: <https://agencia.fapesp.br/materiais-bidimensionais-quase-invisiveis-e-de-enorme-potencial/29332/>. Acesso em: set. 2021.

[11] ZHU, H., LI, X. Two-dimensional MoS_2: Properties, preparation, and applications. Journal of Materiomics, 2015, nº 1, p. 33-44

[12] WANG, J., MA, F., SUN, M. Graphene, hexagonal boron nitride, and their heterostructures: properties and applications. The Royal Society of Chemistry, 2017, nº 7, p. 16801-16822

[13] BARBOSA, D. Q. *Éxcitons intra- e inter-camada em heteroestruturas de Van Der Waals controlados por campo elétrico perpendicular.* 2021. 67f. Dissertação de Mestrado - Universidade Federal do Ceará, Fortaleza

[14] LEE, G. H., et al. Flexible and Transparent MoS_2 Field-Effect Transistors on Hexagonal Boron Nitride-Graphene Heterostructures. American Chemical Society, 2013, nº 9, vol. 7, p. 7931-7936

[15] ELIBOL, K., et al. Grain boundary-mediated nanopores in molybdenum disulfide grown by chemical vapor deposition. The Royal Society of Chemistry, Nanoscale, 2017, nº 9, p. 1591-1598

[16] XIA, L. Importance of nanostructured surfaces. Bioceramics, Elsevier Series on Advanced Ceramic Materials, 2021, p. 5-24

[17] FEIJÓ, T. O. *Crescimento de heteroestruturas de Van der Waals visando a aplicações em nanoeletrônica*. 2021. 132f. Tese de doutorado - Universidade Federal do Rio Grande do Sul, Porto Alegre

[18] *MASS FLOW CONTROLLER. HORIBA.* Disponível em: <https://www.horiba.com/en_en/fluid-measurement-and-control/>. Acesso em: set. 2021

[19] *FMA 5400A/FMA 5500A MASS FLOW CONTROLLERS. OMEGA.* Disponível em: <https://assets.omega.com/manuals/M5372.pdf>. Acesso em: set. 2021

[20] *ARDUINO UNO REV3. ARDUINO STORE.* Disponível em: <https://store-usa.arduino.cc/products/arduino-uno-rev3/?selectedStore=us>. Acesso em: set. 2021

[21] *PROCESSING OVERVIEW. PROCESSING.* Disponível em: <https://processing.org/overview/>. Acesso em: set. 2021

[22] *A BANCADA DOS HIPEROBJETOS. CTA.* Disponível em: <http://cta.if.ufrgs.br/projects/bancada-dos-hiperobjetos/wiki>. Acesso em: set. 2021

[23] *INÍCIO. ENFITEC JÚNIOR.* Disponível em: <https://www.ufrgs.br/enfitecjunior/>. Acesso em: set. 2021

[24] *INTERFACE LED COM ARDUINO E PROCESSING. JEKNOWLEDGE.* Disponível em: <http://jeknowledge.github.io/academy-articles/interface-led-com-arduino-e-processing>. Acesso em: set. 2021

[25] *PWM. ARDUINO.* Disponível em: <https://www.arduino.cc/en/Tutorial/Foundations/PWM>. Acesso em: set. 2021

[26] *ANALOGWRITE(). ARDUINO.* Disponível em: <https://www.arduino.cc/reference/pt/language/functions/analog-io/analogwrite/>. Acesso em: set. 2021

[27] *PWM DO ARDUINO. EMBARCADOS.* Disponível em: <https://www.embarcados.com.br/pwm-do-arduino/>. Acesso em: set. 2021

[28] *MILLIS(). ARDUINO.* Disponível em: <https://www.arduino.cc/reference/pt/language/functions/time/millis/>. Acesso em: set. 2021

[29] *FILTRO CAPACITIVO PASSA-BAIXA E PASSA-ALTA. MUNDO PROJETADO.* Disponível em: <https://mundoprojetado.com.br/filtro-capacitivo-passa-baixa-e-passa-alta/>. Acesso em: set. 2021

[30] *INÍCIO. FRITZING.* Disponível em: <https://fritzing.org/>. Acesso em: out. 2021

[31] *AUTOMAÇÃO INDUSTRIAL E A CONFIABILIDADE DOS PROCESSOS. INSTRUMENTAÇÃO E CONTROLE.* Disponível em: <https://instrumentacaoecontrole.com.br/automacao-industrial-e-a-confiabilidade/>. Acesso em: out 2021

7. APÊNDICES

7.1 Código Processing

```
/*********************************************************************************************************
                            UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL - INSTITUTO DE FÍSICA
                                LABORATÓRIO DE SUPERFÍCIES E INTERFACES SÓLIDAS (LASIS)

                                    TRABALHO DE DIPLOMAÇÃO EM ENGENHARIA FÍSICA II - 2021/1
                        TÍTULO: Automatização do processo de produção de dissulfeto de molibdênio
                                            AUTORA: Bruna Fachin - 00275790
                                           ORIENTAÇÃO: Gabriel Vieira Soares
*********************************************************************************************************
//
//
//
/******************************************** CONFIGURAÇÃO SERIAL *********************************
//
import processing.serial.*;
Serial myPort;
//
/******************************************** DECLARAÇÃO DAS VARIÁVEIS ****************************
//
int fluxo1 = 0, fluxo2 = 0, tempo = 0, cem1 = 0, dez1 = 0, cem2 = 0, dez2 = 0, tdez = 0;
//
/******************************************** CONFIGURAÇÃO INICIAL ********************************
//
void setup() {
  myPort = new Serial(this, Serial.list()[0], 9600); //conexão com a porta serial do arduino
  //programação da imagem da janela
  size(897,549);
  PImage bg;

  bg = loadImage("bginterface.jpg");
  background(bg);
  //programação dos textos
  textSize(20);
  textAlign(CENTER);
  fill(0);
  text(0, 303, 240);
  text(0, 760, 240);
  text(0, 340, 466);
}
//
/********************************************** LOOP DE AÇÕES ***********************************************
//
void draw() {
//não necessita de nenhum código em loop
}
//
/********************************************** PROGRAMAÇÃO DOS BOTÕES ***************************************
//
void mouseClicked() {
  if (mouseX > 103 && mouseX < 147 && mouseY > 196 && mouseY < 230)
  {
    fluxo1 = fluxo1 + 10; //se o botão de +10 foi pressionado, incrementa no fluxo do MFC 1
    fill(239,239,239);
    stroke(239,239,239);
    rect(275,224,60,16);
    fill(0);
```

```
    text(fluxo1, 303, 240); //mostra na tela o valor sendo incrementado
  }
  if (mouseX > 103 && mouseX < 147 && mouseY > 237 && mouseY < 270)
  {
    fluxo1 = fluxo1 - 10; //se o botão de -10 foi pressionado, decrementa no fluxo do MFC 1
    fill(239,239,239);
    stroke(239,239,239);
    rect(275,224,60,16);
    fill(0);
    text(fluxo1, 303, 240); //mostra na tela o valor sendo decrementado
  }
  if (mouseX > 561 && mouseX < 605 && mouseY > 196 && mouseY < 230)
  {
    fluxo2 = fluxo2 + 10; //se o botão de +10 foi pressionado, incrementa no fluxo do MFC 2
    fill(239,239,239);
    stroke(239,239,239);
    rect(733,224,60,16);
    fill(0);
    text(fluxo2, 760, 240); //mostra na tela o valor sendo incrementado
  }
  if (mouseX > 561 && mouseX < 605 && mouseY > 237 && mouseY < 270)
  {
    fluxo2 = fluxo2 - 10; //se o botão de -10 foi pressionado, decrementa no fluxo do MFC 2
    fill(239,239,239);
    stroke(239,239,239);
    rect(733,224,60,16);
    fill(0);

    text(fluxo2, 760, 240); //mostra na tela o valor sendo decrementado
  }
  if (mouseX > 135 && mouseX < 179 && mouseY > 421 && mouseY < 455)
  {
    tempo = tempo + 10; //se o botão de +10 foi pressionado, incrementa no tempo
    fill(239,239,239);
    stroke(239,239,239);
    rect(311,450,60,16);
    fill(0);
    text(tempo, 340, 466); //mostra na tela o valor sendo incrementado
  }
  if (mouseX > 135 && mouseX < 179 && mouseY > 461 && mouseY < 495)
  {
    tempo = tempo - 10; //se o botão de -10 foi pressionado, decrementa no tempo
    fill(239,239,239);
    stroke(239,239,239);
    rect(311,450,60,16);
    fill(0);
    text(tempo, 340, 466); //mostra na tela o valor sendo decrementado
  }
//
/******************************************** CONEXÃO COM O ARDUINO ********************************************
//
  if(mouseX > 319 && mouseX < 381 && mouseY > 286 && mouseY < 333) //envio dos parâmetros do MFC 1 após ser dado OK
  {
    if(fluxo1 > 400)
    {
```

```
  fluxo1 = 400; //força que o fluxo máximo seja 400sccm
}
cem1 = fluxo1/100; //separa centena do número
switch (cem1) //envio da centena do número
{
   case 0:
     myPort.write("a");
     break;
   case 1:
     myPort.write("b");
     break;
   case 2:
     myPort.write("c");
     break;
   case 3:
     myPort.write("d");
     break;
   case 4:
     myPort.write("e");
     break;
}
if (fluxo1 < 100) //envio da dezena do número para fluxo <100
{
   dez1 = fluxo1/10; //separa dezena do número
   switch (dez1)
   {
     case 0:

       myPort.write("f");
       break;
    case 1:
      myPort.write("g");
      break;
    case 2:
      myPort.write("h");
      break;
    case 3:
      myPort.write("i");
      break;
    case 4:
      myPort.write("j");
      break;
    case 5:
      myPort.write("k");
      break;
    case 6:
      myPort.write("l");
      break;
    case 7:
      myPort.write("m");
      break;
    case 8:
      myPort.write("n");
      break;
    case 9:
```

```
          myPort.write("o");
          break;
      }
   }
   else //envio da dezena do número para fluxo >100
   {
      dez1 = (fluxo1%100)/10; //separa dezena do número
      switch (dez1)
      {
        case 0:
          myPort.write("f");
          break;
        case 1:
          myPort.write("g");
          break;
        case 2:
          myPort.write("h");
          break;
        case 3:
          myPort.write("i");
          break;
        case 4:
          myPort.write("j");
          break;
        case 5:
          myPort.write("k");
          break;

       case 6:
          myPort.write("l");
          break;
        case 7:
          myPort.write("m");
          break;
        case 8:
          myPort.write("n");
          break;
        case 9:
          myPort.write("o");
          break;
      }
   }
   fluxo1 = 0; //zera variável do fluxo do MFC 1 após envio
}
if(mouseX > 777 && mouseX < 839 && mouseY > 286 && mouseY < 333) //envio dos parâmetros do MFC 2 após OK
{
   if(fluxo2 > 400)
   {
     fluxo2 = 400; //força que o fluxo máximo seja 400sccm
   }
   cem2 = fluxo2/100; //separa centena do número
   switch (cem2) //envio da centena do número
   {
      case 0:
        myPort.write("A");
```

```
      break;
    case 1:
      myPort.write("B");
      break;
    case 2:
      myPort.write("C");
      break;
    case 3:
      myPort.write("D");
      break;
    case 4:
      myPort.write("E");
      break;
}
if (fluxo2 < 100) //envio da dezena do número para fluxo <100
{
    dez2 = fluxo2/10; //separa dezena do número
    switch (dez2)
    {
      case 0:
        myPort.write("F");
        break;
      case 1:
        myPort.write("G");
        break;
      case 2:
        myPort.write("H");

        break;
      case 3:
        myPort.write("I");
        break;
      case 4:
        myPort.write("J");
        break;
      case 5:
        myPort.write("K");
        break;
      case 6:
        myPort.write("L");
        break;
      case 7:
        myPort.write("M");
        break;
      case 8:
        myPort.write("N");
        break;
      case 9:
        myPort.write("O");
        break;
    }
}
else //envio da dezena do número para fluxo >100
{
    dez2 = (fluxo2%100)/10; //separa dezena do número
```

```
        switch (dez2)
        {
          case 0:
            myPort.write("F");
            break;
          case 1:
            myPort.write("G");
            break;
          case 2:
            myPort.write("H");
            break;
          case 3:
            myPort.write("I");
            break;
          case 4:
            myPort.write("J");
            break;
          case 5:
            myPort.write("K");
            break;
          case 6:
            myPort.write("L");
            break;
          case 7:
            myPort.write("M");
            break;
          case 8:

            myPort.write("N");
            break;
          case 9:
            myPort.write("O");
            break;
        }
      }
      fluxo2 = 0; //zera variável do fluxo do MFC 2 após envio
    }
    if(mouseX > 454 && mouseX < 516 && mouseY > 438 && mouseY < 485) //envio do parâmetro de tempo após OK
    {
      if(tempo > 90)
      {
        tempo = 90; //força que o tempo máximo seja 90min
      }
      tdez = tempo/10; //separa dezena do número
      myPort.write(tdez);
      tempo = 0; //zera variável do tempo após envio
    }
}
//
/********************************************** FIM DO PROGRAMA ***************************************
```

7.2 Código Arduino

```
/*****************************************************************************************************
                            UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL - INSTITUTO DE FÍSICA
                               LABORATÓRIO DE SUPERFÍCIES E INTERFACES SÓLIDAS (LASIS)

                                  TRABALHO DE DIPLOMAÇÃO EM ENGENHARIA FÍSICA II - 2021/1
                          TÍTULO: Automatização do processo de produção de dissulfeto de molibdênio
                                              AUTORA: Bruna Fachin - 00275790
                                            ORIENTAÇÃO: Gabriel Vieira Soares
*****************************************************************************************************
//
//
//
/********************************************** DEFINIÇÃO DOS PINOS ******************************
//
#include <stdlib.h> //biblioteca padrão
#define mfc1 9 //definindo entrada do MFC 1 no pino 9 do arduino
#define mfc2 10 //definindo entrada do MFC 2 no pino 10 do arduino
//
/********************************************** DECLARAÇÃO DAS VARIÁVEIS *************************
//
boolean flagcem1 = 0, flagdez1 = 0, flagcem2 = 0, flagdez2 = 0, flagt = 0, start = 0;
int cem1 = 0, dez1 = 0, cem2 = 0, dez2 = 0, t = 0;
int fluxo1 = 0, fluxo2 = 0, tempo = 0, pwm1 = 0, pwm2 = 0;
unsigned long t_atual = 0, t_final = 0, t_envio = 0;
//
/********************************************** CONFIGURAÇÃO INICIAL *****************************
//
void setup()
{

  pinMode(mfc1, OUTPUT); //define o pino 9 do arduino como saída (para escrita)
  pinMode(mfc2, OUTPUT); //define o pino 10 do arduino como saída (para escrita)
  Serial.begin(9600); //inicia comunicação em série com o Processing
}
//
/********************************************** LOOP DE AÇÕES ************************************
//
void loop()
{
  interface(); //chama função que recebe valores do Processing
  conversao(); //chama função que converte os algarismos recebidos
  start = teste(); //chama função que testa os limites do fluxo
  if (start == 1) //início do experimento
  {
    envio(); //chama função que envia valores aos MFCs
  }
}
//
/********************************************** CONEXÃO COM O PROCESSING *************************
//
void interface()
{
  do
  {
    if (Serial.available()) //testa se a comunicação serial está disponível
    {
      switch(Serial.read()) //transforma caractere recebido pela porta serial em algarismos
```

```
{
  case 'a':
    ceml = 0;
    flagceml = 1;
    break;
  case 'b':
    ceml = 1;
    flagceml = 1;
    break;
  case 'c':
    ceml = 2;
    flagceml = 1;
    break;
  case 'd':
    ceml = 3;
    flagceml = 1;
    break;
  case 'e':
    ceml = 4;
    flagceml = 1;
    break;
  case 'f':
    dezl = 0;
    flagdezl = 1;
    break;
  case 'g':

    dezl = 1;
    flagdezl = 1;
    break;
  case 'h':
    dezl = 2;
    flagdezl = 1;
    break;
  case 'i':
    dezl = 3;
    flagdezl = 1;
    break;
  case 'j':
    dezl = 4;
    flagdezl = 1;
    break;
  case 'k':
    dezl = 5;
    flagdezl = 1;
    break;
  case 'l':
    dezl = 6;
    flagdezl = 1;
    break;
  case 'm':
    dezl = 7;
    flagdezl = 1;
    break;
```

```
case 'n':
  dez1 = 8;
  flagdez1 = 1;
  break;
case 'o':
  dez1 = 9;
  flagdez1 = 1;
  break;
case 'A':
  cem2 = 0;
  flagcem2 = 1;
  break;
case 'B':
  cem2 = 1;
  flagcem2 = 1;
  break;
case 'C':
  cem2 = 2;
  flagcem2 = 1;
  break;
case 'D':
  cem2 = 3;
  flagcem2 = 1;
  break;
case 'E':
  cem2 = 4;
  flagcem2 = 1;

   break;
 case 'F':
   dez2 = 0;
   flagdez2 = 1;
   break;
 case 'G':
   dez2 = 1;
   flagdez2 = 1;
   break;
 case 'H':
   dez2 = 2;
   flagdez2 = 1;
   break;
 case 'I':
   dez2 = 3;
   flagdez2 = 1;
   break;
 case 'J':
   dez2 = 4;
   flagdez2 = 1;
   break;
 case 'K':
   dez2 = 5;
   flagdez2 = 1;
   break;
 case 'L':
   dez2 = 6;
```

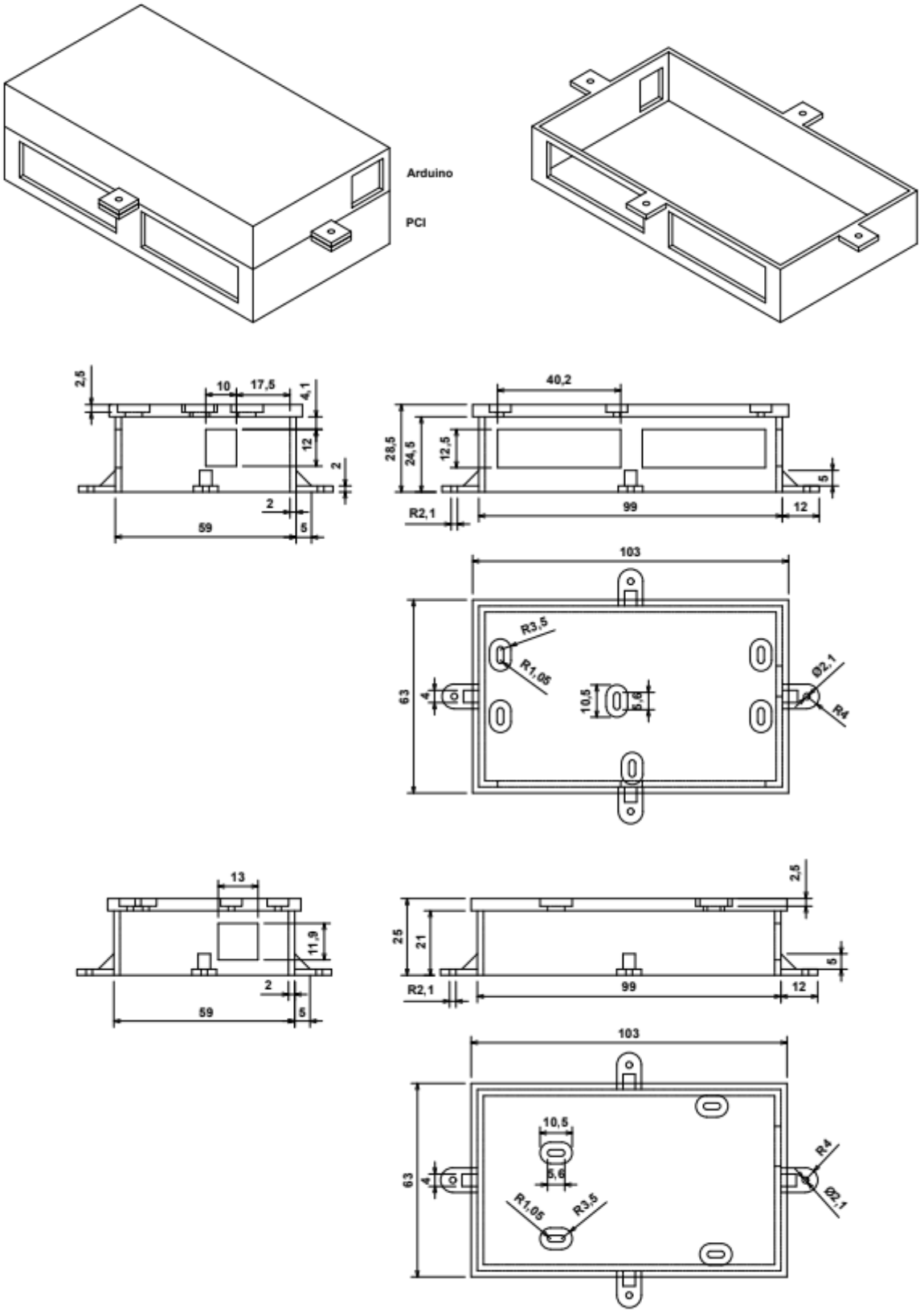
```
  flagdez2 = 1;
  break;
case 'M':
  dez2 = 7;
  flagdez2 = 1;
  break;
case 'N':
  dez2 = 8;
  flagdez2 = 1;
  break;
case 'O':
  dez2 = 9;
  flagdez2 = 1;
  break;
case 0:
  t = 0;
  flagt = 1;
  break;
case 1:
  t = 1;
  flagt = 1;
  break;
case 2:
  t = 2;
  flagt = 1;
  break;
case 3:

  t = 3;
  flagt = 1;
  break;
case 4:
  t = 4;
  flagt = 1;
  break;
case 5:
  t = 5;
  flagt = 1;
  break;
case 6:
  t = 6;
  flagt = 1;
  break;
case 7:
  t = 7;
  flagt = 1;
  break;
case 8:
  t = 8;
  flagt = 1;
  break;
case 9:
  t = 9;
  flagt = 1;
  break;
```

```
      }
    }
  }
  while(flagceml == 0 || flagdezl == 0 || flagcem2 == 0 || flagdez2 == 0 || flagt == 0); //roda até receber tudo
}
//
/********************************************** CONVERSÃO DOS VALORES **********************************************
//
void conversao() //converte os algarismos recebidos para valor de fluxo e tempo
{
  fluxol = ((ceml)*100) + ((dezl)*10);
  fluxo2 = ((cem2)*100) + ((dez2)*10);
  tempo = t*10;
}
//
/********************************************** TESTE DOS FLUXOS **********************************************
//
int teste() //verificação dos limites de fluxo
{
  if (fluxol > 0 || fluxo2 > 0) //testa se os valores são maiores do que zero
  {
    pwml = map (fluxol, 0, 500, 0, 255); //conversão dos limites de fluxo para os limites do pwm
    pwm2 = map (fluxo2, 0, 500, 0, 255); //conversão dos limites de fluxo para os limites do pwm
    return 1; //retorna 1 se pelo menos um dos valores de fluxo for maior que zero
  }
  else
  {

    return 0; //retorna 0 se ambos os fluxos estiverem zerados
  }
}
/********************************************** ENVIO DOS FLUXOS PELO TEMPO DEFINIDO **********************************************
//
void envio()
{
  t_envio = tempo*60000; //conversão do tempo de minutos para milissegundos
  t_atual = millis();
  while(t_atual <= t_envio) //envia fluxos enquanto o tempo atual é menor que o selecionado
  {
    analogWrite(mfc1, pwml); //enviando valor selecionado de fluxo ao MFC 1
    analogWrite(mfc2, pwm2); //enviando valor selecionado de fluxo ao MFC 2
    t_atual = millis();
  }
  if(t_atual >= t_envio)
  {
    analogWrite(mfc1, 0); //zerando valor de fluxo enviado ao MFC 1
    analogWrite(mfc2, 0); //zerando valor de fluxo enviado ao MFC 2
    start = 0; //zera flag que controla início de experimento
  }
}
//
/********************************************** FIM DO PROGRAMA **********************************************
```

7.3 Desenho técnico impressão 3D



7.4 Esquemático da placa de circuito impresso