UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA – INSTITUTO DE FÍSICA

CURSO DE ENGENHARIA FÍSICA

LAÍS ROSA DE OLIVEIRA

AUTOMATIZAÇÃO DO SISTEMA DE LIBERAÇÃO DE PRECURSORES EM UM REATOR CVD UTILIZADO PARA A PRODUÇÃO DE GRAFENO

Porto Alegre

2018

LAÍS ROSA DE OLIVEIRA

AUTOMATIZAÇÃO DO SISTEMA DE LIBERAÇÃO DE PRECURSORES EM UM REATOR CVD UTILIZADO PARA A PRODUÇÃO DE GRAFENO

Monografia apresentada à Universidade Federal do Rio Grande do Sul como parte dos requisitos para obtenção do título de Bacharel em Engenharia Física.

Orientador: Prof. Dr. Gabriel Vieira Soares

Porto Alegre

2018

AGRADECIMENTOS

Primeiramente a Deus, por me dar a vida, por me proteger e por permitir que pessoas especiais cruzassem meu caminho.

Aos meus pais, Nadir e Leidizar, que sempre fizeram o possível e o impossível para me apoiar e me ver feliz. Às minhas irmãs, meus cunhados, meus sobrinhos e toda minha família, pelo amor, pelo apoio e incentivo durante toda a minha vida.

Ao meu orientador, professor Gabriel Soares, pela ajuda, atenção e grande paciência comigo na elaboração deste trabalho.

À Taís, pelos anos de amizade e companheirismo, pela ajuda, pela atenção, pelas dicas e por não medir esforços em me auxiliar no que fosse preciso para desenvolver este projeto. Não tenho palavras para agradecer!

A todos os meus amigos, em especial ao Ítalo e ao Samuel, pelas constantes demonstrações de apoio e incentivo.

Ao Everton, por não me deixar desistir e pelas diversas idas às lojas de componentes eletrônicos sempre que precisei de um material para implementar o projeto.

Aos professores integrantes da banca avaliadora, Cristiano Krug e Fabiano Bernardi, pelas correções e sugestões no desenvolvimento deste trabalho.

Aos professores Marcelo Barbalho e Cristiano Krug, integrantes da COMGRAD de Engenharia Física, pelo auxílio na resolução de todas as questões burocráticas, viabilizando a defesa deste trabalho.

Aos meus colegas de trabalho, que diariamente manifestaram apoio, torcida e preocupação, em especial ao Leo Rolim, que me salvou dos loops infinitos.

A todos vocês e aqueles que sempre torceram por mim, meu sincero agradecimento!

“Só a descoberta desperta. Só a invenção prova que se pensa de verdade a coisa que se pensa, seja qual for essa coisa. Só o sopro criativo dá a vida, pois a vida inventa”.

Michel Serres

SUMÁRIO

RESUMO

O presente trabalho tem por objetivo desenvolver um sistema de controle de vazão de gases precursores em um reator utilizado para produzir grafeno. Neste reator são utilizados os gases hidrogênio (H_2) e metano (CH_4) para formar grafeno através da técnica de deposição química a partir da fase de vapor (CVD). Um dos parâmetros a serem controlados durante a síntese de grafeno por CVD é a vazão dos precursores, que deve ser muito bem estabelecida e controlada com precisão. Tal controle era feito através do ajuste manual de um controlador de vazão mássica (MFC), o que trazia dificuldades de operação e poderia ser fonte de diversos erros e imprecisões experimentais, podendo resultar na obtenção de um material com qualidade inferior à esperada. O presente projeto se propôs a automatizar o controle da vazão dos gases precursores, produzindo um sistema de controle formado por uma placa Arduino e uma interface com o usuário contendo um display LCD e um conjunto de botões de controle, permitindo ao operador do reator apenas selecionar as vazões desejadas, enquanto o sistema faz o controle eletronicamente dos MFCs. O resultado do projeto foi satisfatório, necessitando ainda de ajustes para a correta conexão do dispositivo com os MFCs.

PALAVRAS-CHAVES: Grafeno, Deposição Química na Fase de Vapor (CVD), Arduino, Controlador de Vazão Mássica (MFC), Automatização.

ABSTRACT

The main objective of this work is the development of a control system of the gas flow in a reactor used to produce graphene. In this reactor, two gases, hydrogen (H_2) and methane (CH_4), are the precursors of the graphene, which is obtained by the technique called Chemical Vapor Deposition (CVD). One of the parameters to be controlled during the synthesis of graphene by CVD is the flow of the precursors, which must be very well established and controlled accurately. Since the control was done through the manual adjustment of a Mass Flow Controller (MFC), some difficulties in operation were found and it could be the source of errors and experimental inaccuracies, resulting in a material with inferior quality than the expected one. This project develops a device to automate the control of the gases flow, formed by an Arduino board and an user interface containing a LCD display and a set of control buttons, allowing the operator to select the desired flows, while the system electronically controls the MFCs. The result of the project was satisfactory, but the system still requires adjustments for the correct connection of the device with the MFCs.

KEY-WORDS: Graphene, Chemical Vapor Deposition (CVD), Arduino, Mass Flow Controller (MFC), Automatization.

1. INTRODUÇÃO

Em virtude do seu conjunto de propriedades, o grafeno é um material que possui um grande potencial de se tornar revolucionário para a ciência e para a tecnologia. Sabe-se que, dentre outras características, ele é condutor elétrico [1] e térmico [2], é transparente a luz visível [3] e possui alta resistência mecânica e flexibilidade [4]. O fato de um único material possuir todas estas propriedades faz com que o grafeno venha sendo muito estudado ultimamente e cogitado para aplicações de inovação em diversas áreas, como eletrônica, instrumentação, medicina, fotônica, microeletrônica e energia [1].

No Laboratório de Superfícies e Interfaces Sólidas da Universidade Federal do Rio Grande do Sul (LASIS), o grafeno é produzido pela técnica chamada de Deposição Química a partir da Fase de Vapor – CVD (do inglês, *Chemical Vapor Deposition*), em um reator que utiliza os gases metano e hidrogênio como precursores. Neste reator, o controle da vazão dos gases é feito por dois controladores de vazão mássica – MFCs (do inglês, *Mass Flow Controllers*), um para cada gás, que podem ser ajustados de forma local (manual) ou remota (utilizando um dispositivo de interface). O ajuste manual, utilizado até então, traz algumas limitações e desvantagens, pois dificulta um controle ágil e preciso.

Um estudo recente sobre o grafeno produzido neste reator mostrou que o crescimento homogêneo do material depende diretamente da vazão de metano e de hidrogênio no sistema, pois a vazão mássica dos precursores interfere no processo de nucleação do grafeno [5]. Portanto, é necessário que este parâmetro seja bem definido para obter filmes de grafeno com melhor qualidade, já que pequenas mudanças na vazão influenciam diretamente o resultado final do experimento.

Tendo em vista o contexto tecnológico e o caráter promissor das pesquisas envolvendo o grafeno, surge a necessidade de aprimorar o sistema de liberação de precursores do reator CVD do LASIS. Embora existam no mercado soluções de controle automatizadas, como a solução

da *Omega Engineering*, empresa fabricante dos MFCs utilizados neste sistema, que oferece um dispositivo pronto para este propósito, seu custo elevado faz com que soluções alternativas sejam pesquisadas e desenvolvidas.

Com o intuito de diminuir os custos de uma solução convencional, a plataforma Arduino foi eleita como peça fundamental deste projeto, pois apresenta baixo custo e facilidade de manuseio e programação. Neste projeto, o Arduino será utilizado, juntamente com os MFCs, para desenvolver o sistema de controle automatizado da vazão de gases precursores no reator.

2. OBJETIVOS

O objetivo deste trabalho é automatizar o sistema de liberação de precursores do reator CVD do LASIS, através do projeto e construção de um dispositivo microcontrolado que atue como interface entre o usuário e o sistema, sendo capaz de ajustar remotamente os controladores de vazão mássica presentes nas linhas de gases.

O dispositivo projetado será composto basicamente por uma placa Arduino, um display LCD e um conjunto de botões de controle, como ilustra de forma esquemática a figura a seguir.

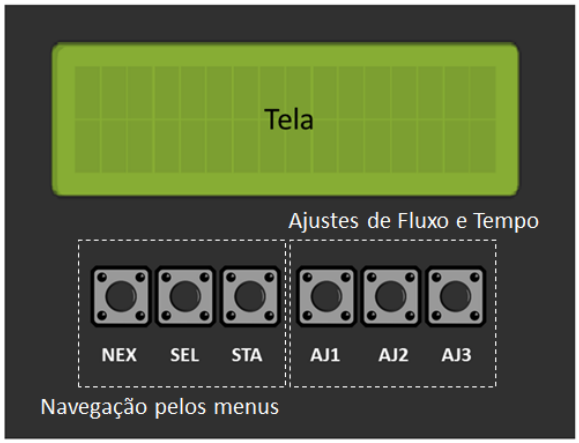

Figura 1: Esquema de montagem do dispositivo.

O dispositivo exibirá um menu com diversas telas de ajuste, que serão exibidas, uma de cada vez, à medida que o usuário pressiona um botão determinado. Em cada uma dessas telas será possível ajustar os parâmetros do experimento, como a vazão de cada um dos gases e o tempo de duração de cada etapa do experimento.

3. REVISÃO DE CONCEITOS

3.1 GRAFENO

O grafeno é um membro da classe dos materiais bidimensionais (2D), isto é, materiais formados por uma única camada de átomos. No grafeno, esta monocamada é composta por átomos de carbono no estado de hibridização sp^2, conectados por três ligações covalentes. Seus átomos estão arranjados em uma estrutura cristalina hexagonal, separados por um ângulo de 120° [6]. O grafeno é o elemento básico da formação de outros compostos de carbono com diferentes dimensões, pois dependendo da forma como é arranjado ele pode formar os fulerenos, os nanotubos ou o próprio grafite [6, 7]. A figura abaixo ilustra a estrutura hexagonal do grafeno e as demais estruturas formadas a partir dele.

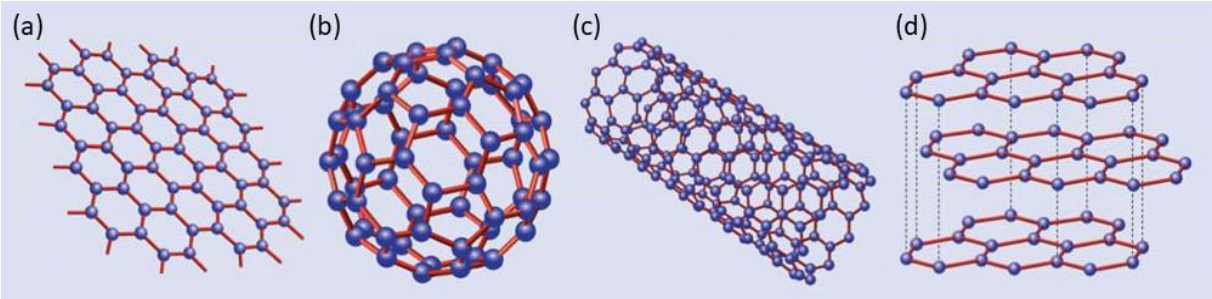

Figura 2: (a) Estrutura cristalina hexagonal do grafeno. Camadas de grafeno formando outras estruturas como os fulerenos (b), os nanotubos (c) e o grafite (d). Adaptado de [7].

Métodos de Produção

Na primeira vez que o grafeno foi isolado, em 2004, Andre Geim e Konstantin Novoselov utilizaram o método da esfoliação mecânica [6], feito que lhes rendeu o Prêmio Nobel em 2010. De lá para cá diversas técnicas vêm sendo estudadas e as que mais se destacam são: a esfoliação mecânica, a esfoliação química em fase líquida, a sublimação de carbetos e a deposição química a partir da fase de vapor.

(i) Esfoliação mecânica

A esfoliação mecânica consiste em utilizar uma fita adesiva para esfoliar uma amostra de grafite de alta pureza, fazendo com que uma fina camada de átomos de carbono (grafeno) seja

transferida para a fita. Na sequência, o grafeno é transferido para outro substrato, como o silício, por exemplo [6]. Sua principal vantagem é a simplicidade do método e a alta qualidade do grafeno obtido; em contrapartida, suas maiores desvantagens são a falta de reprodutibilidade, a área limitada da amostra e a irregularidade do formato e do tamanho do material [8], que depende do tamanho e do formato da fita utilizada.

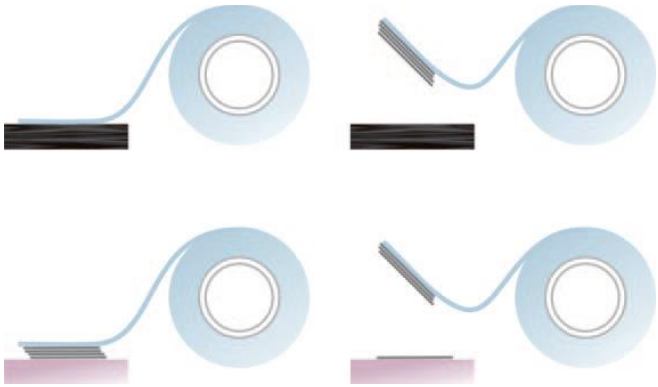

Figura 3: Produção de grafeno por esfoliação mecânica. Reproduzido de [9].

(ii) Esfoliação química em fase líquida

A esfoliação química em fase líquida trata de submeter uma mistura de grafite e de um solvente orgânico a uma fonte de energia (geralmente ondas ultrassônicas), fazendo com que as ligações de Van der Waals presentes entre as camadas de grafeno que compõem o grafite sejam quebradas, isolando, desta forma, o grafeno [10]. Esta técnica apresenta as maiores vantagens no que diz respeito a produção, pois é uma técnica que pode ser executada em escala industrial. Entretanto, é limitada por apresentar um grafeno com baixa qualidade eletrônica [8].

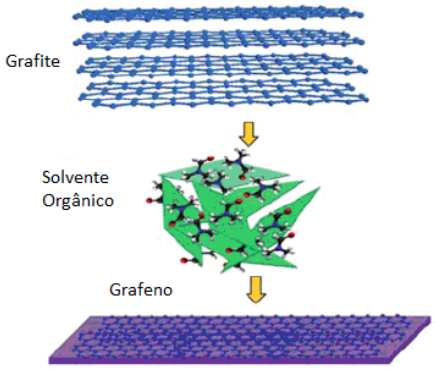

Figura 4: Produção de grafeno por esfoliação química. Adaptado de [11].

(iii) Sublimação de carbetos

A sublimação de carbetos consiste no tratamento térmico em atmosfera inerte de um tipo de carbeto, geralmente o carbeto de silício (SiC), até que o elemento ligante com o carbono sublime e reste apenas uma camada de grafeno [1], como ilustra a figura abaixo. Suas vantagens são a regularidade do grafeno obtido e a ausência da etapa de transferência de substrato; porém, a técnica é limitada pelo tamanho do substrato [8].

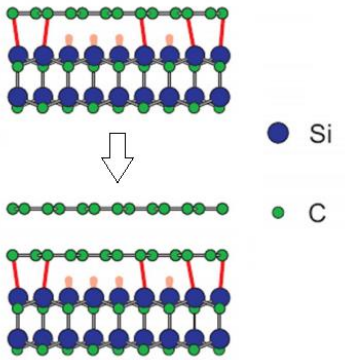

Figura 5: Produção de grafeno por sublimação de carbetos. Adaptado de [12].

(iv) Deposição química a partir da fase de vapor (CVD)

A produção de grafeno por CVD trata de submeter um substrato metálico ou com uma camada metálica (geralmente cobre ou níquel) a um tratamento térmico na presença de um precursor gasoso à base de carbono. A uma certa temperatura a molécula do gás é quebrada, fazendo com que os átomos de carbono provenientes do precursor sejam aderidos ao substrato, formando um filme fino de grafeno [1, 10].

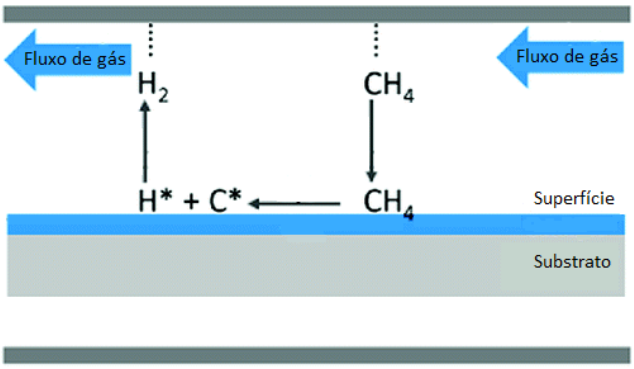

Figura 6: Produção de grafeno por CVD. Adaptado de [13].

A técnica de CVD é muito promissora, pois resulta em monocamadas de grafeno de excelente qualidade [8], além de ser possível sua produção em larga escala, limitada apenas pela dimensão do sistema. Outro fator interessante em relação a esta técnica é seu custo de produção em massa, que é um dos menores, comparado às demais técnicas [1]. A figura abaixo ilustra a relação entre o custo de produção em massa e a qualidade do grafeno obtido para as técnicas citadas.

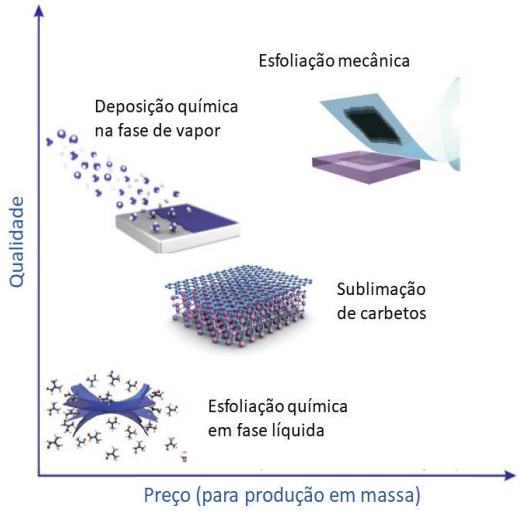

Figura 7: Relação entre diferentes técnicas de obtenção de grafeno em termos de qualidade e custo de produção. Adaptado de [1].

Para obter grafeno através de CVD, alguns parâmetros experimentais precisam ser controlados, como a qualidade do substrato, a temperatura e pressão da câmara e a vazão de gases. Dentre outros fatores, o controle destes parâmetros condicionará a produção de um material de boa qualidade [14].

Propriedades e Aplicações

Dentre as principais propriedades físicas do grafeno, pode-se destacar sua condutividade elétrica, com limite de mobilidade de elétrons de cerca de 200.000 $cm^2/V.s$ [7] e sua transparência à luz visível, apresentando uma transmitância maior que 97% e caracterizando-o como um dos poucos condutores elétricos transparentes [3]. Além disso, o grafeno apresenta uma alta resistência mecânica, com uma tensão de ruptura de 130 GPa, cerca de 100 vezes superior ao aço e um módulo

de Young de 1 TPa, caracterizando-se, também, como um material flexível [4]. Ademais, o grafeno também apresenta diversas outras propriedades, como elevada condutividade térmica, ferromagnetismo induzido por defeitos estruturais e transporte eletrônico balístico [6].

As aplicações do grafeno contemplam diversas áreas [1], desde o uso na fabricação de baterias e sensores, passando pelas células solares e até mesmo para armazenamento de hidrogênio em células combustíveis não poluentes. Uma das áreas de maior interesse de cientistas e engenheiros atualmente e uma das mais promissoras para aplicação do grafeno é a eletrônica flexível, pois – sendo um material condutor, transparente e, ao mesmo tempo, resistente mecanicamente e flexível – o grafeno tem os pré-requisitos fundamentais para o desenvolvimento desta tecnologia [15]. As figuras abaixo ilustram diferentes aplicações do grafeno na indústria.

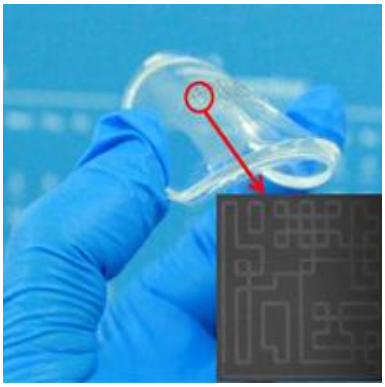

Figura 8: Circuito eletrônico de grafeno construído sobre substrato flexível. Reproduzido de [15].

Figura 9: Bateria produzida por uma empresa da China utilizando grafeno. Reproduzido de [16].

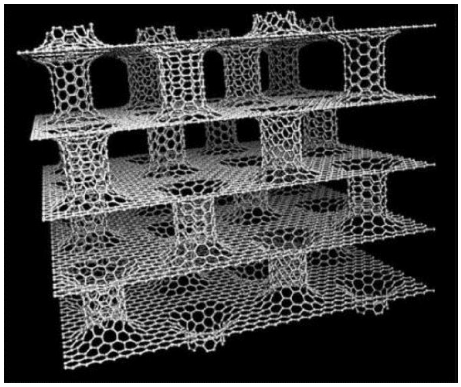

Figura 10: Camadas de grafeno conectadas por nanotubos de carbono que poderiam armazenar 6,1% do seu peso em hidrogênio. Reproduzido de [17].

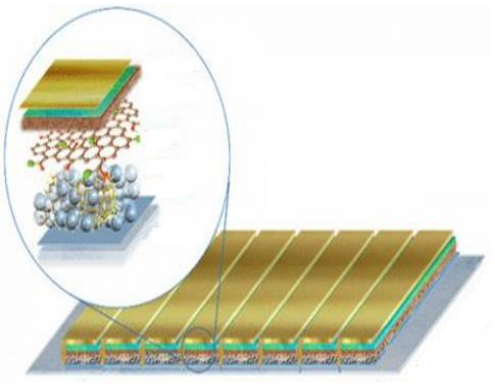

Figura 11: Esquema de uma célula solar produzida com uma camada de grafeno. Adaptado de [18].

3.2 REATOR CVD

O LASIS possui um reator CVD utilizado para a produção de grafeno, construído em 2014, que opera utilizando precursores gasosos (H_2 e CH_4) e possui uma geometria horizontal. É composto por três partes principais: o sistema de liberação de precursores, a câmara de reação e o sistema de exaustão, conforme ilustra o esquema abaixo.

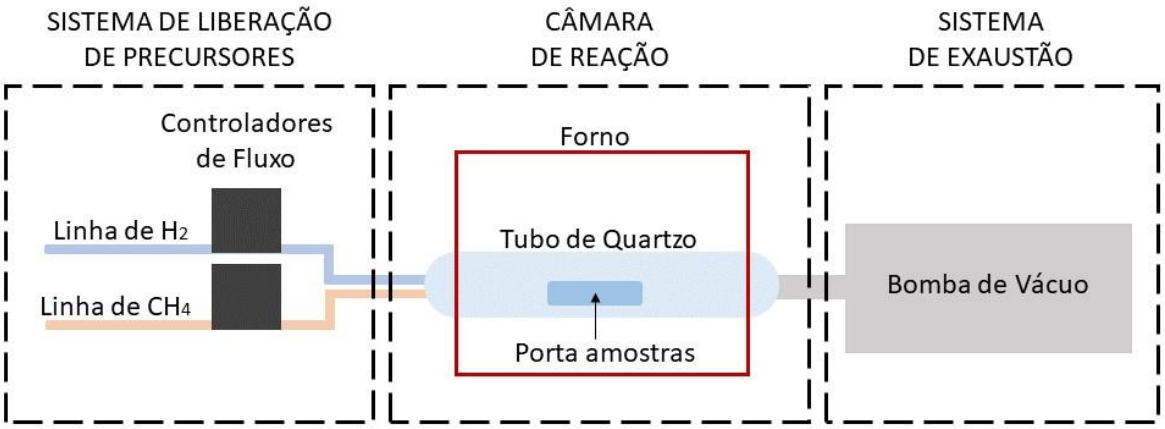

Figura 12: Três partes principais do reator CVD do LASIS.

O sistema de liberação de precursores tem como principal função fornecer os gases que irão fluir para dentro da câmara de reação. Neste reator, tal sistema é formado por uma linha de CH_4, utilizado como precursor do grafeno, e por outra linha de H_2, usado para tratamento térmico do substrato, limpeza e eliminação do óxido nativo. Além disso, também existem dois controladores de vazão mássica, colocados em cada uma das linhas, responsáveis pelo controle da injeção de H_2 e CH_4 durante o processo.

A câmara de reação conta com o sistema de aquecimento, que consiste em um forno resistivo, um controlador de temperatura e um tubo de quartzo. No interior do tubo de quartzo fluem os gases precursores onde está o porta-amostras. É neste local que o substrato está colocado e onde ocorre a formação do grafeno.

O sistema de exaustão é formado por uma bomba de vácuo mecânica, que tem como principais funções realizar a limpeza da atmosfera antes das reações e manter o fluxo de gases no interior da câmara durante o processo de obtenção de grafeno.

3.3 CONTROLADOR DE VAZÃO MÁSSICA (MFC)

Um controlador de vazão mássica é um dispositivo que pode detectar, medir e controlar taxas de vazão de diferentes gases. Neste projeto, dois MFCs da marca *Omega*, modelo FMA 5500A, que possuem intervalo de medição[1] de 0 a 500 sccm, são utilizados para compor o sistema de injeção de precursores do reator.

Figura 13: MFC modelo FMA 5500A da marca Omega.

Princípio de Funcionamento

O princípio de funcionamento do MFC é o *by-pass*, ilustrado na figura a seguir, em que o gás que entra no dispositivo é dividido em duas partes: uma pequena porção flui através de um tubo capilar e o restante flui normalmente através do canal principal. A geometria do canal e do tubo capilar são projetados para garantir o regime de escoamento laminar, de modo que as vazões medidas no tubo sejam diretamente proporcionais à vazão total do gás [19].

[1] O sccm (do inglês, *standard cubic centimeter per minute*) é uma unidade de vazão volumétrica de um fluido que corresponde a 1 cm^3/min em condições normais de temperatura e pressão (0 °C e 1 atm).

O sensor de vazão opera pelo princípio térmico de detecção, em que uma pequena quantidade de calor é introduzida no tubo capilar por meio de um elemento aquecedor e a temperatura é medida em dois pontos do tubo através de um sensor térmico do tipo RTD (do inglês, *Resistance Temperature Detector*) [20]. O calor é transferido através da fina parede do tubo capilar para o gás e à medida que a vazão se estabelece no sistema, o mesmo é transportado pelo gás entre os dois RTDs. O diferencial de resistência dos sensores é detectado eletronicamente e a diferença de temperatura medida é linearmente proporcional à vazão instantânea do gás. O circuito de controle do MFC compara continuamente a saída de vazão mássica com o valor selecionado (*setpoint*). Os desvios do ponto de ajuste são corrigidos por uma válvula de compensação, mantendo a vazão desejada constante [19].

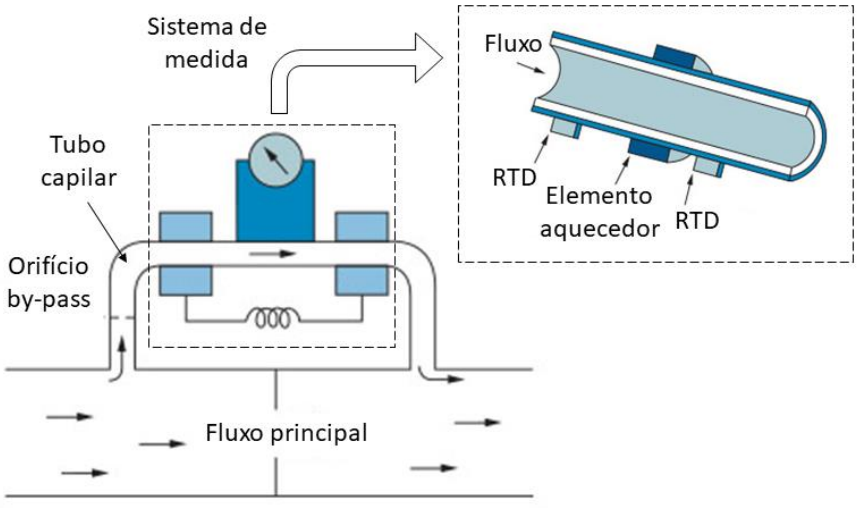

Figura 14: Esquema de funcionamento do MFC e detalhe do sistema térmico de medição. Adaptado de [20].

Modos de Ajuste

A escolha do *setpoint* no MFC pode ser feita de duas formas: através de um controle local no dispositivo ou de forma remota, através da comunicação por uma porta serial [19]. Atualmente, o controle é feito manualmente, através de giro de um pequeno parafuso na lateral do controlador, com o auxílio de uma chave de fenda. O esquema abaixo ilustra a vista posterior

do controlador MFC, detalhando seus principais componentes e mostrando seu controle manual.

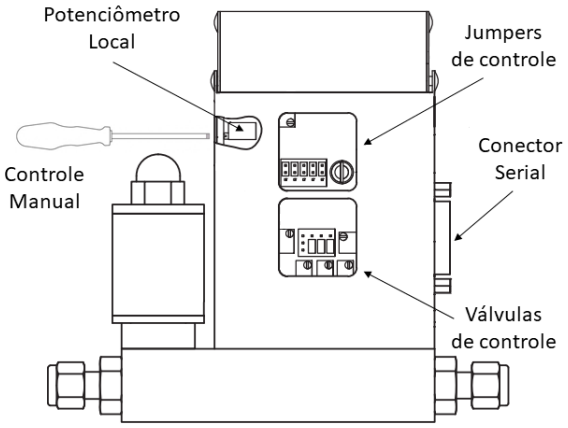

Figura 15: Principais componentes de controle do MFC. Adaptado de [19].

O controle de forma remota pode ser feito através da conexão do MFC com um dispositivo via porta serial. É possível controlar a vazão dos gases através de um sinal aplicado entre os pinos 8 e 10, conforme o mapa de pinos abaixo.

Pino	Função
1	Sinal de saída do fluxo (comum)
2	Sinal de saída do fluxo (0 – 5 V)
3	Controle da válvula de abertura (comum)
4	Controle da válvula de abertura
5	Alimentação (comum)
6	(sem atribuição)
7	Alimentação (+12 V)
8	Entrada do sinal de controle (0 – 5 V)
9	Sinal de saída do fluxo (4 – 20 mA (-))
10	Entrada do sinal de controle (comum)
11	Sinal de referência para o controle (5 V)
12	Controle da válvula de desligamento
13	Alimentação auxiliar (+12 V)
14	Sinal de saída do fluxo (4 – 20 mA (+))
15	Terra

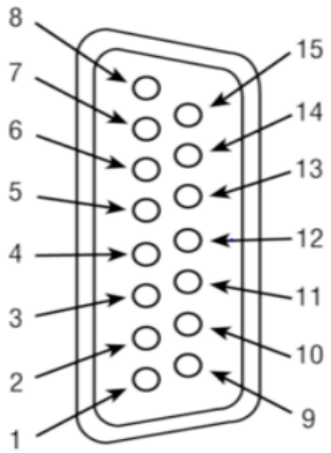

Figura 16: Configuração dos pinos de conexão do MFC. Adaptado de [19].

Para ativar o modo remoto, é necessário modificar a configuração dos jumpers de controle mostrados na figura anterior. A tabela abaixo mostra as combinações para cada propósito:

Ajuste	Jumpers				
	NJ1A	NJ1B	NJ1C	NJ1D	NJ1E
Local	2-3	5-6	8-9	11-12	14-15
Remoto	2-3	5-6	8-9	10-11	14-15

Tabela 1: Configuração dos jumpers do MFC. Adaptado de [19].

3.4 PLATAFORMA ARDUINO

Arduino é uma plataforma de prototipagem eletrônica de hardware livre, projetada com um microcontrolador Atmel AVR e com suporte de entradas e saídas em seus pinos analógicos e digitais, permitindo as mais diversas aplicações. O objetivo dos criadores do Arduino era desenvolver uma ferramenta que permitisse uma prototipagem rápida, barata e simples.

O grande diferencial da plataforma está na fácil programação do microcontrolador que nela está embarcado, graças ao uso de uma interface de programação própria, a IDE (do inglês, *Integrated Development Environment*) [21]. A IDE compila os códigos e os transfere para a placa via cabo USB, onde são gravados na memória interna do microcontrolador. O código é escrito em uma linguagem própria, similar a C++, que utiliza diversas bibliotecas, permitindo a comunicação da placa com dispositivos externos como o display de LCD utilizado neste projeto.

Com o intuito de facilitar a implementação deste projeto e de diminuir custos, a plataforma Arduino foi escolhida como principal componente, estabelecendo o controle dos MFCs e se comunicando com o usuário através da implementação de uma interface.

Especificações Arduino Mega 2560

Existem, atualmente, diversos modelos de placas Arduino e similares disponíveis no mercado, com diferentes componentes e características. Para este trabalho, foi escolhida a placa Arduino Mega 2560, por melhor atender às necessidades do projeto, em termos de tensão de operação, corrente elétrica e quantidade de pinos analógicos e digitais disponíveis.

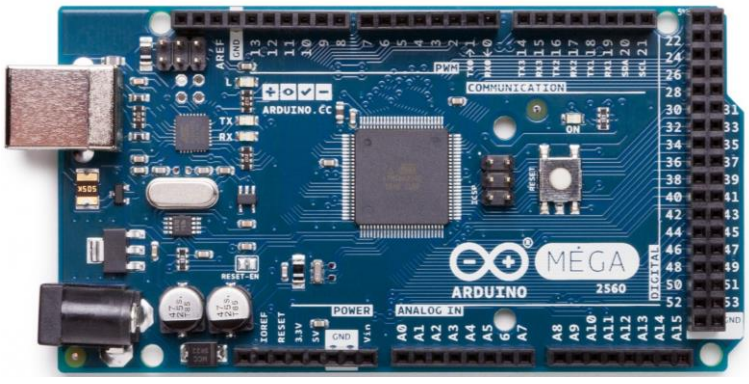

Figura 17: Arduino Mega 2560. Reproduzido de [22].

O Arduino Mega 2560 possui um microcontrolador ATmega2560, que efetua o processamento dos sinais lidos através dos pinos digitais e analógicos da placa. Tais pinos podem ser configurados como entrada ou saída através do software desenvolvido, permitindo que o Arduino leia os sinais provenientes de um sistema e que exerça o controle sobre outros dispositivos. A tabela a seguir mostra as especificações da placa utilizada.

Microcontrolador	ATmega2560
Tensão de operação	5 V
Alimentação (Recomendada)	7 – 12 V
Alimentação (Limite)	6 – 20 V
Pinos Digitais	54
Pinos Analógicos	16
Corrente DC nos pinos digitais	40 mA
Corrente DC no pino 3,3 V	50 mA
Memória Flash	256 kB
SRAM	8 kB
EEPROM	4 kB
Frequência do processador	16 MHz

Tabela 2: Especificações do Arduino Mega 2560. Adaptado de [22].

Configuração dos Pinos

Além de ler os sinais provenientes de um sensor, é possível utilizar os pinos digitais para exercer controle sobre diversos componentes quando colocados no modo *output*. Os pinos conseguem fornecer uma corrente significativa (em torno de 40 mA), podendo, por exemplo, acender LED's, controlar servos, relés e outros componentes, inclusive os MFCs.

A função que permite colocar as saídas dos pinos digitais em *high* ou *low* é a *digitalWrite*, porém é importante notar que os pinos irão fornecer 5 V de saída quando o estado for colocado em *high* e 0 V quando colocado em *low*.

Outra opção e mais interessante para este projeto, é gerar ondas moduladas, ou PWM (do inglês, *Pulse Width Modulation*), utilizando uma função onde é possível definir a porcentagem do tempo em que a onda permanece em nível lógico alto (*duty cycle*). Esta configuração provoca a mudança no valor médio da onda, que por sua vez, está relacionado com a tensão entregue no pino correspondente. O duty cycle varia de 0 a 255, onde 0 significa que o sinal de saída é 0 V, e 255 significa que o sinal de saída é 5 V. Os valores intermediários fornecem ondas quadradas, que apresentam variações nos tempos em que o pino permanece em *high* e em *low* [23].

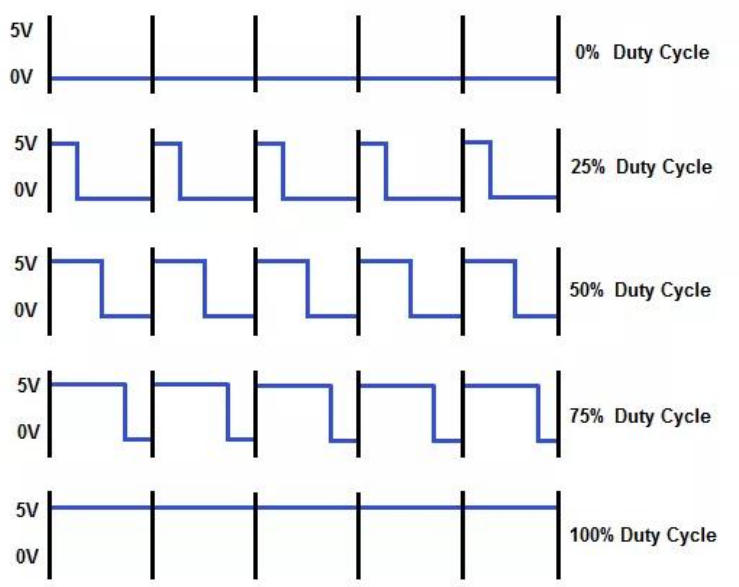

Figura 18: Exemplo de geração de sinais por modulação de largura de pulso (PWM). Adaptado de [24].

Observando a figura acima é possível concluir que a terceira onda está 50% do período em nível baixo e 50% em nível alto, fazendo com que a tensão média deste sinal seja 50% da tensão total. Já o segundo sinal está apenas 25% do tempo em nível alto, fazendo com que sua tensão média seja apenas 25% da tensão total.

Este efeito de variação de período é a base para geração de um sinal analógico, entretanto, o mesmo não pode ser utilizado sem passar por um filtro de frequência, pois uma saída analógica deve resultar em uma tensão contínua. Conforme explicado na próxima seção, é possível converter as saídas PWM em tensão contínua utilizando um filtro “passa-baixa”.

3.5 FILTRO RC PASSA-BAIXA

Um filtro passa-baixa é um circuito passivo que permite apenas a passagem de sinais de tensão e corrente em frequências abaixo de um certo limite (a frequência de corte), atenuando os sinais cuja frequência ultrapassar esse valor. Um circuito RC, formado por um resistor e um capacitor, como o apresentado na figura a seguir, pode ser empregado como um filtro passa-baixa, onde V_e é o sinal de entrada e V_s é o sinal de saída.

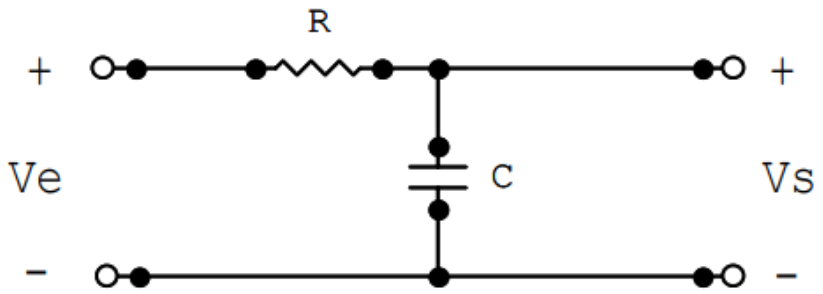

Figura 19: Filtro RC passa-baixa. Reproduzido de [25].

Sabemos que um resistor tem sua resistência fixa e independe da frequência do sinal aplicado no circuito, entretanto, um capacitor apresenta uma reatância capacitiva (X_C) que depende da frequência do sinal de entrada e que a variação dessa reatância capacitiva é inversamente proporcional à frequência do sinal, conforme a expressão abaixo, onde f é a frequência do sinal e C a capacitância do capacitor:

$$X_C = \frac{1}{2\pi f C}$$

Tal expressão nos permite concluir que quanto maior a frequência do sinal aplicado, menor será a reatância capacitiva. Portanto, em frequências muito altas, o capacitor se comporta como um curto-circuito. Desta forma, a maior parcela da tensão de entrada estará sobre o resistor e a tensão sobre o capacitor de saída será muito pequena. Podemos dizer que o circuito "permite a passagem" de sinais de alta frequência.

Já para o caso de frequências menores sendo aplicadas ao circuito, maior será a reatância capacitiva. Portanto, para frequência zero (sinal contínuo), o capacitor se comporta como um

circuito aberto. Desta forma, a maior parcela da tensão de entrada estará sobre o capacitor de saída. Podemos dizer que o circuito apresentado “deixa passar” sinais de baixa frequência.

A frequência de corte do circuito pode ser ajustada escolhendo corretamente os valores de resistência e capacitância dos componentes, que são relacionados através da seguinte expressão:

$$f_C = \frac{1}{2\pi RC}$$

4. METODOLOGIA

4.1 INTERFACE COM O USUÁRIO

Componentes

O componente responsável por conectar o usuário ao controlador, permitindo o acompanhamento do ajuste das variáveis e a seleção do valor desejado, é o display. Neste projeto, é utilizado um display LCD 16x2, que exibe uma matriz de caracteres de 16 colunas por 2 linhas, como mostrado na figura abaixo.

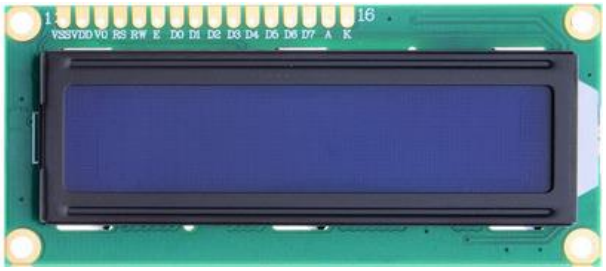

Figura 20: Display LCD 16x2.

O controle dos caracteres mostrados no display LCD é feito via software, através da biblioteca *LiquidCrystal.h* presente na IDE do Arduino. A conexão entre o display e o Arduino é feita através de seus pinos digitais. A tabela abaixo mostra a configuração de pinos do display LCD e seus pinos correspondentes no Arduino.

Pino	**Função**	**Conexão Arduino**
V_{SS}	Alimentação (Terra)	GND (terra)
V_{DD}	Alimentação (5 V)	VCC (5 V)
V_0	Contraste da tela	VCC via potenciômetro
RS	Seleção entre dado (1) ou instrução (0)	Pino digital
R/W	Seleção entre leitura (1) ou escrita (0)	Pino digital
E	Habilita (1) ou desabilita (0) o controle	Pino digital
D0 – D7	Barramento de dados	Pinos digitais
A	Ânodo do LED de luz de fundo	VCC
K	Cátodo do LED de luz de fundo	GND

Tabela 3: Conexões entre o display LCD e o Arduino. Adaptado de [26].

Outro componente importante para a interface entre o sistema e o usuário é o conjunto de botões, que alternam seus estados entre 0 e 1 lógico conforme pressionados. Esta informação é processada pelo Arduino através de cada pino digital conectado aos botões. A conexão dos botões é feita através de um resistor de *pull-up* ligado ao V_{CC}. O esquema abaixo ilustra os dois estados lógicos providos pelo circuito, onde nota-se que o botão atua como uma chave, fazendo com que o pino digital do Arduino seja ligado diretamente ao GND (*low*) ou aos 5 V (*high*). Neste projeto, os pinos de entrada para os botões foram configurados com a função *INPUT_PULLUP*, que aciona um resistor de 20 kΩ presente no Atmega 2560, eliminando a necessidade de implementar resistores externos no circuito.

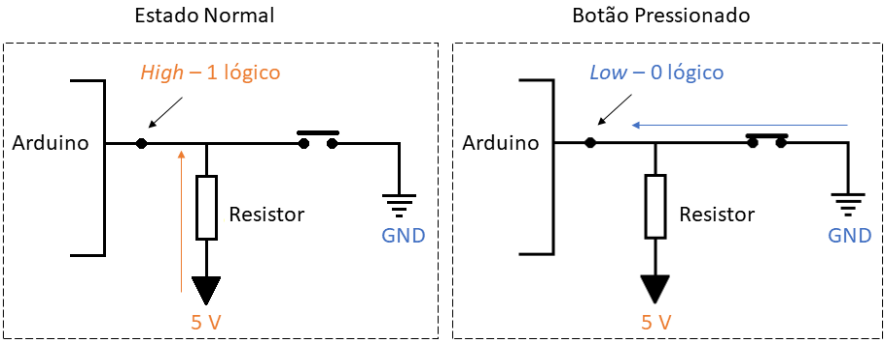

Figura 21: Circuito de *pull-up* estabelecendo as duas configurações lógicas do botão.

Lógica de Menus

O usuário irá navegar por um conjunto de menus pressionando o botão "NEXT". Em cada tela será possível ajustar um dos parâmetros do experimento através dos botões de ajuste "AJ1", "AJ2" e "AJ3". Dependendo do parâmetro que está sendo ajustado, os botões de ajuste incrementam valores diferentes nas varáveis. A tabela abaixo mostra os valores dos incrementos nos botões de ajuste em cada parâmetro.

Botão/Parâmetro	Vazão	Tempo
AJ1	+100 sccm	+10 min
AJ2	+10 sccm	+1 min
AJ3	+1 sccm	+ 5 s

Tabela 4: Definição dos ajustes via botões.

É interessante para o experimento que a vazão dos gases aumente de forma lenta até atingir o valor desejado. Por este motivo foi implementada uma “rampa”, onde o usuário escolhe o intervalo de tempo necessário para o gás em questão chegar ao valor final. Assim, nas primeiras 4 telas serão ajustados as vazões e as rampas de cada gás.

O ajuste será feito pressionando o botão “SELECT” e quando o valor do parâmetro estiver piscando na tela será possível determinar seu valor pressionando os botões “AJs” correspondentes. Para confirmar a escolha é necessário pressionar “SELECT” novamente, assim o valor na tela irá parar de piscar e o valor da variável será registrado.

Figura 22: Telas de ajuste de vazão e rampa para cada gás.

Da mesma forma, nas próximas 3 telas será possível o ajuste dos tempos característicos do experimento. No sistema deste reator, o experimento é dividido em três partes: Num primeiro momento, na fase de “recozimento” o substrato é tratado termicamente apenas com H_2 a uma certa temperatura e durante um certo tempo. Na sequência é iniciada a fase de “crescimento”, onde o CH_4 é adicionado ao sistema para fornecer os átomos de carbono que formarão o grafeno. Nesta etapa, os dois gases fluem no sistema. A última etapa, o “resfriamento”, consiste de aguardar o resfriamento da amostra produzida enquanto apenas o H_2 continua no sistema. Estes três tempos também serão definidos pelo usuário através dos menus, conforme ilustra a figura abaixo.

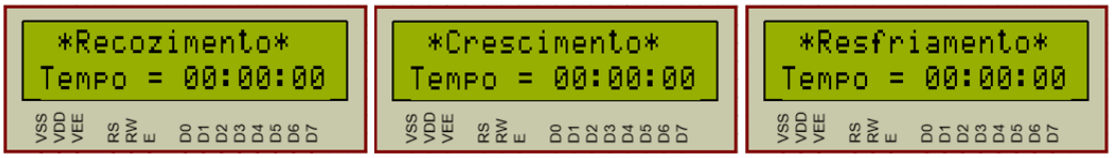

Figura 23: Telas de ajuste dos tempos de experimento.

Quando todos os parâmetros estão escolhidos e salvos pelo usuário, o experimento está pronto para iniciar. Para isso, o usuário deve pressionar a tecla “START” e ao responder à mensagem de confirmação, iniciando de fato o experimento, o Arduino passará a enviar o valor de tensão correspondente à vazão determinada durante o tempo escolhido.

Figura 24: Mensagem de confirmação ao pressionar a tecla "START".

4.2 CONTROLE DE VAZÃO

Como visto na seção 3.3, o MFC pode ser controlado através de um sinal analógico de tensão entre 0 e 5 V aplicado entre os pinos 8 e 10 de seu conector [19]. O usuário irá definir através dos botões de ajuste um valor entre 0 e 500 que corresponderá à vazão do gás em sccm. Este valor será convertido pelo programa em tensão e será fornecido pelo Arduino. Esta configuração é similar a um potenciômetro, onde, no limite inferior, 0 sccm corresponde a um *setpoint* de 0 V, e no limite superior, 500 sccm corresponde a um *setpoint* de 5 V.

O sinal de tensão é gerado utilizando a função *analogWrite*, que gera sinais PWM de acordo com o valor escolhido para o *duty-cycle*, conforme explicado anteriormente. Para converter as ondas quadradas em sinais contínuos, utiliza-se a seguinte configuração nos filtros passa-baixa: um capacitor de 100 nF e um resistor de 4,7 kΩ, que, de acordo com a equação que calcula a frequência de corte, temos f_c = 340 Hz.

4.3 CONTROLE DE TEMPO

O controle de tempo é feito exclusivamente via software, utilizando a função *millis* do Arduino. Esta função é simplesmente um contador que armazena o tempo (em milissegundos) que o controlador está ligado. Através de ajustes no código é possível utilizar estas contagens para definir intervalos dentro da lógica do programa. É possível, também, utilizar para este

propósito a função *delay*, mais conhecida e mais utilizada nos projetos com Arduino, entretanto a função *delay* de fato pausa a execução do programa em uma determinada linha e só volta a executá-lo após o tempo determinado, fazendo com que a leitura de sensores ou de botões, como no caso deste projeto, seja interrompida. Com a função *millis* não temos este problema, visto que ela utiliza os registradores internos do microcontrolador para efetuar a contagem, sendo executada paralelamente com o restante do código [27].

4.4 CIRCUITO

O circuito foi projetado e testado primeiramente no software de simulação *Proteus 8.5 Professional*, uma ferramenta que permite projetar os mais diferentes circuitos e simular seu funcionamento. O *Proteus* possui diversas bibliotecas repletas de componentes eletrônicos, incluindo todos os componentes usados no presente projeto.

O layout final do circuito, mostrando todos os componentes e suas conexões, pode ser visto a seguir.

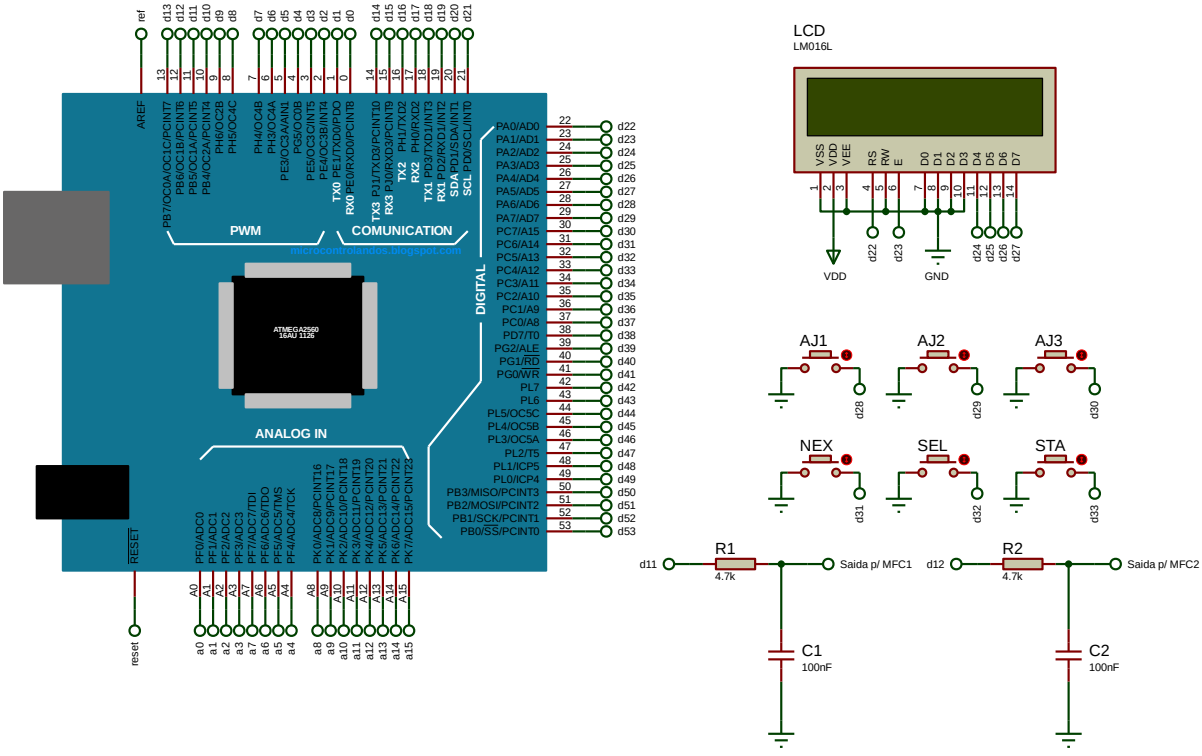

Figura 25: Esquema do circuito projetado no simulador.

Uma funcionalidade interessante do simulador é o uso de *labels* para determinar as conexões do circuito: nota-se que cada pino do Arduino e do display, assim como os terminais dos botões e dos filtros possuem uma legenda atribuída, o que quer dizer que dois terminais com as mesmas legendas estarão conectados. Como por exemplo, o terminal do botão “AJ1” tem a legenda “d28”, assim como o pino digital 28 do Arduino tem a legenda “d28”, significando, portanto, que o botão “AJ1” está conectado ao pino 28 do Arduino.

4.5 CÓDIGO

O código desenvolvido controla as informações que aparecem no display, interpreta as seleções do usuário, através da leitura dos parâmetros de controle do circuito, e gera a saída PWM, fazendo com que o bloco de controle do MFC receba o sinal de tensão correspondente ao valor de vazão desejado.

Para executar todas estas tarefas, foram desenvolvidas diversas funções dentro do código. Sabe-se que o código implementado no Arduino possui dois blocos principais: o bloco de *setup*, onde as variáveis e os pinos utilizados são definidos e o bloco de *loop* onde o código permanece rodando repetidamente.

Para o loop foram desenvolvidas duas funções, uma delas, a função *menus()*, testa continuamente se os botões foram pressionados e incrementa o valor de uma varável de controle, a variável “menu”, que representa o menu selecionado. A outra função, chamada de *show()*, interpreta o valor da varável “menu” e mostra o menu correspondente. Para isso, a função é baseada em um bloco *switch...case* na variável “menu”, que alterna a instrução de acordo com o valor da variável de referência.

Em cada menu será possível definir uma variável diferente do experimento e, para isso, existe em cada caso uma função auxiliar, que permite que o programa grave o valor de cada uma das variáveis à medida que as mesmas são definidas pelo usuário.

Outra função desenvolvida e executada dentro de todas as funções auxiliares é a função *readbut()*, que efetua a leitura dos sinais dos botões de ajuste, para identificar quando forem pressionados e incrementar as variáveis correspondentes de acordo com a Tabela 3. Sua estrutura é baseada em blocos *if...else* que executam as instruções de incrementar as variáveis apenas se os botões de ajuste forem pressionados e soltos.

A tabela abaixo resume todas as funções desenvolvidas.

Função	Tarefa	Menu	Executada em:
msg()	Mostrar a mensagem inicial ao ligar o sistema.	-	setup()
menus()	Incrementar a variável dos menus quando os botões de controle forem pressionados.	-	loop()
show()	Interpretar a variável acima e chamar as funções auxiliares de cada menu.	-	loop()
readbut()	Monitorar o estado dos três botões de ajuste e definir quais valores foram selecionados.	-	todas as funções de ajuste
setflow1()	Mostrar a tela de ajuste e salvar o valor ajustado para a vazão do gás 1.	1	show()
rampa1()	Mostrar a tela de ajuste e salvar o valor ajustado para a rampa do gás 1.	2	show()
setflow2()	Mostrar a tela de ajuste e salvar o valor ajustado para a vazão do gás 2.	3	show()
rampa2()	Mostrar a tela de ajuste e salvar o valor ajustado para a rampa do gás 2.	4	show()
time1()	Mostrar a tela de ajuste e salvar o valor ajustado para o tempo de “recozimento”.	5	show()
time2()	Mostrar a tela de ajuste e salvar o valor ajustado para o tempo de “crescimento”.	6	show()
time3()	Mostrar a tela de ajuste e salvar o valor ajustado para o tempo de “resfriamento”.	7	show()
start()	Mostrar a mensagem de confirmação de início e chamar a função de ajuste da saída.	8	show()
outpwm()	Usar os valores de tensão de tempo salvos para gerar o sinal de saída conforme definido.	-	start()

Tabela 5: Funções desenvolvidas para o código do controlador.

O bloco principal de código é responsável por mostrar os menus no display, ler os botões de controle e de ajuste e gravar os valores ajustados. Esse bloco segue o seguinte fluxograma:

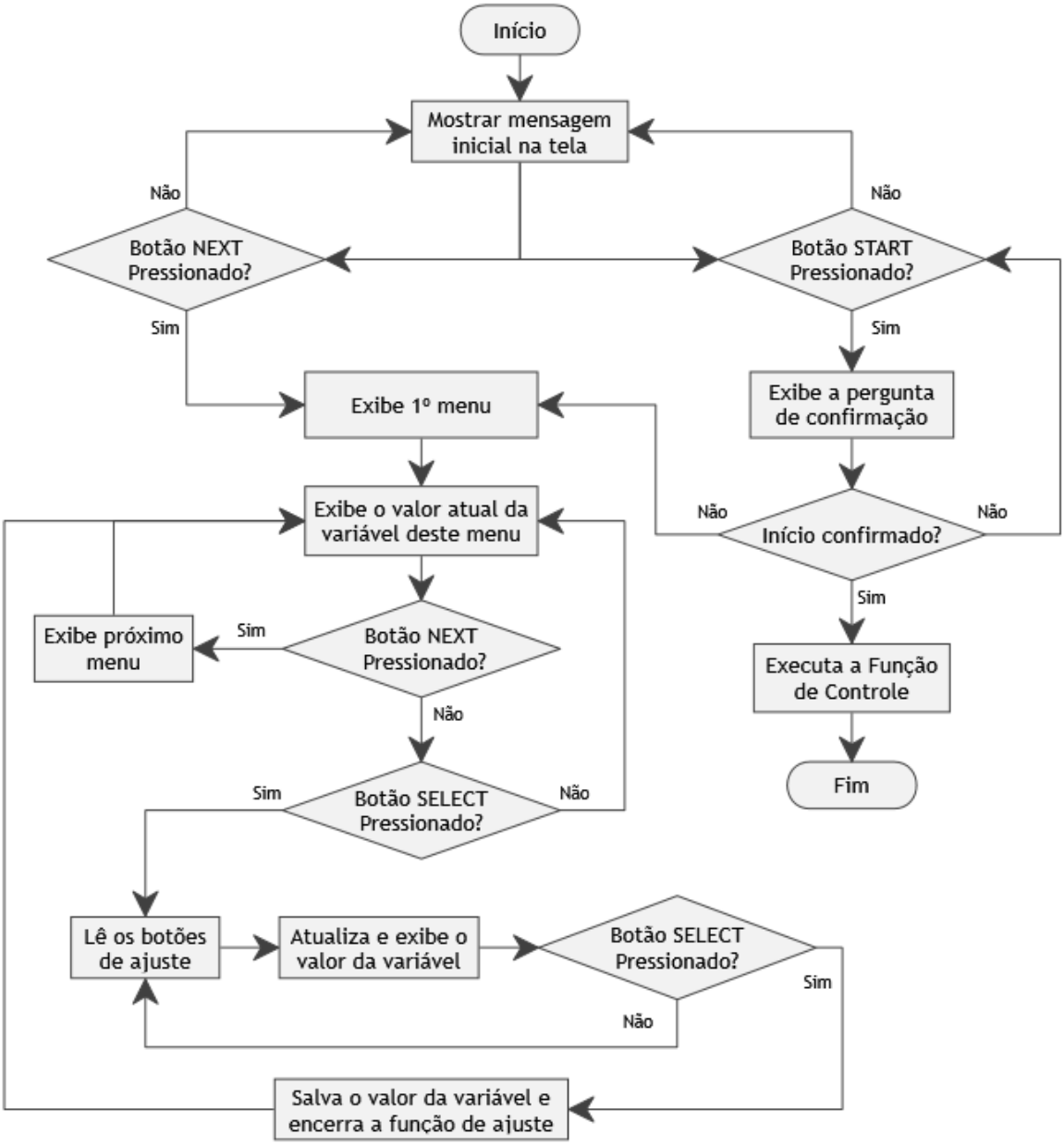

Figura 26: Fluxograma do bloco principal do código.

O código completo, com o desenvolvimento de todas as funções auxiliares está disponível no Anexo deste trabalho.

4.6 MONTAGEM FINAL

Para conectar adequadamente o dispositivo projetado ao MFC, foi utilizado um cabo metálico de 15 vias, cada uma conectada a um pino do conector tipo “D” (padrão do MFC). Nesse cabo, cada via foi identificada e conectada à parte do circuito correspondente, seguindo o mapa de pinos do conector do MFC (Figura 15).

Para alimentação do MFC foi conectada aos pinos correspondentes (5 e 7) uma fonte de tensão de 12 V. Para alimentação do Arduino, foi conectada uma fonte de 9 V no mesmo.

4.7 TESTES

Os seguintes testes foram realizados com o objetivo de verificar o funcionamento do controlador após sua montagem final:

Tensão vs. Vazão

Após o desenvolvimento do controlador, foi testada qual seria sua saída em tensão para diferentes vazões escolhidas pelo usuário. Para isso, utilizou-se um multímetro digital e verificou-se a tensão de saída do circuito. O principal objetivo deste teste é verificar a linearidade do sinal, isto é, se a cada incremento no valor de vazão temos um incremento proporcional na saída de tensão.

Rampa de Tensão

Também foi executado um teste que verificou o implemento da rampa de vazão, com o objetivo de observar a variação gradual da saída conforme os parâmetros definidos pelo usuário. Assim como no teste anterior, o principal ponto a ser verificado é a linearidade da rampa, isto é, se a cada instante decorrido desde o início do experimento a saída teve seu valor aumentado gradualmente, até atingir seu valor máximo ao final do intervalo de tempo em questão. Novamente foi utilizado um multímetro digital para registrar a tensão de saída. Além disso, a

saída também foi aplicada a um LED, para observar o comportamento de seu brilho e verificar se, de fato, havia um aumento gradual.

Simulação de experimento completo

Um experimento completo também foi simulado, monitorando as duas saídas de tensão e utilizando dois LEDs para representar cada um dos MFCs. Todos os parâmetros de vazão e tempo foram ajustados no controlador da mesma forma que se ajustaria em um experimento real. O objetivo foi verificar a sincronia dos tempos definidos, isto é, se durante cada intervalo do experimento apenas o LED correspondente ao MFC que deveria estar liberando a vazão, de fato estava recebendo a saída de tensão do controlador.

Conexão a um MFC

O protótipo do controlador foi conectado a um dos MFCs em uma linha de gás argônio para testes iniciais. Visto que tanto o metano, quanto o hidrogênio são gases perigosos e caros, preferiu-se utilizar um gás inerte e evitar possíveis acidentes. O objetivo deste teste é, de fato, verificar o MFC sendo controlado pelo dispositivo desenvolvido.

5. RESULTADOS E DISCUSSÃO

5.1 DISPOSITIVO

As imagens abaixo mostram o dispositivo projetado, com todas suas conexões. A segunda imagem detalha o conjunto de botões de controle e o display. Sabe-se que um dispositivo montado em *protoboard* está sujeito a mal contato entre os fios e os componentes. O ideal é transformar o dispositivo abaixo em uma placa de circuito impressa, onde todos os componentes são conectados através de solda e alocados no interior de um invólucro apropriado.

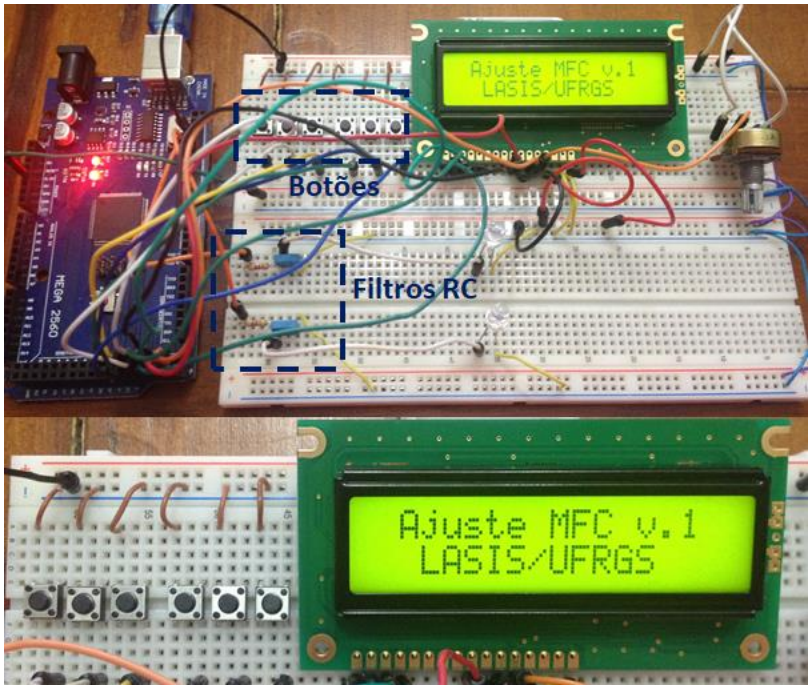

Figura 27: Dispositivo projetado.

5.2 TESTES

Tensão vs. Vazão

Como observamos abaixo, a saída de tensão do circuito projetado responde linearmente de acordo com a escolha da vazão, confirmando que um incremento no valor de vazão gera uma saída proporcional em tensão. Percebe-se, ainda, que o valor máximo de tensão, que teoricamente deveria ser 5 V, é, na verdade, 4,45 V. Essa diferença é provavelmente devido às perdas do circuito e ao filtro RC.

Em termos práticos, essa diferença no valor máximo não afeta o MFC, pois é possível, através do pino de referência, informar ao MFC qual é o valor máximo da escala que fará o seu controle, para que, internamente, seja feita a correspondência.

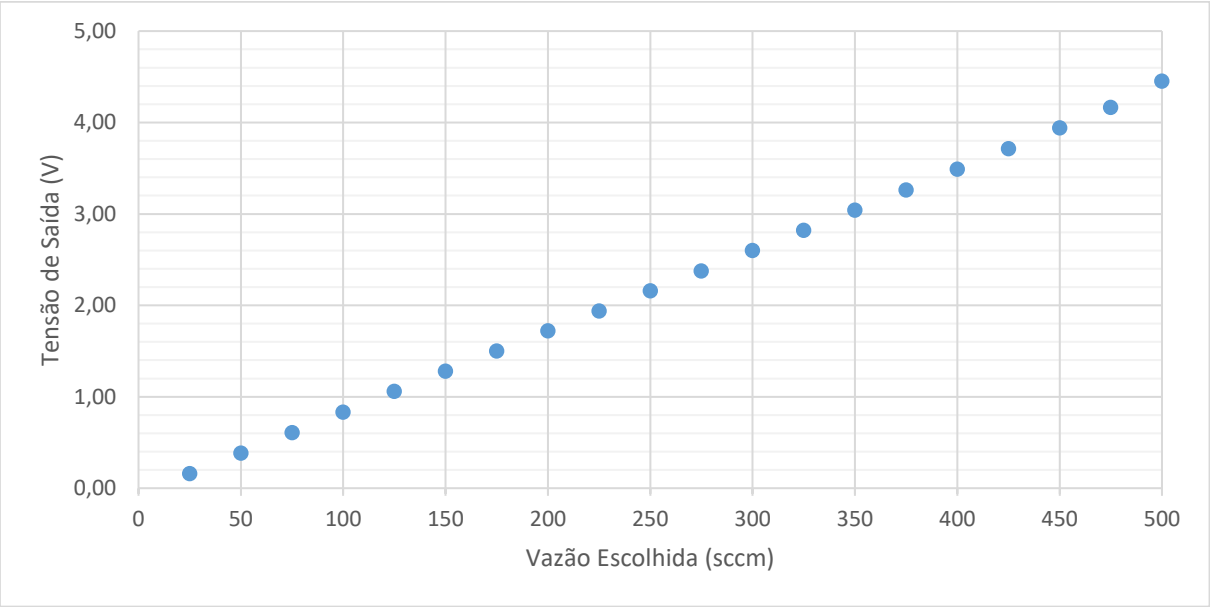

Figura 28: Resposta de tensão para cada vazão selecionado.

Rampa de Tensão

Para melhor observar os dados, foi ajustada uma rampa de 180 s para dois vazões diferentes: 300 e 500 sccm. Observou-se, para ambos os casos, a subida gradual no valor de tensão de saída, até atingir seu valor máximo (4,37 V para 500 sccm e 2,60 V para 300 sccm), ao final do tempo determinado.

Além disso, foi possível observar aumento gradual do brilho do LED que estava conectado à saída do circuito. A figura abaixo mostra as respostas do circuito a este teste.

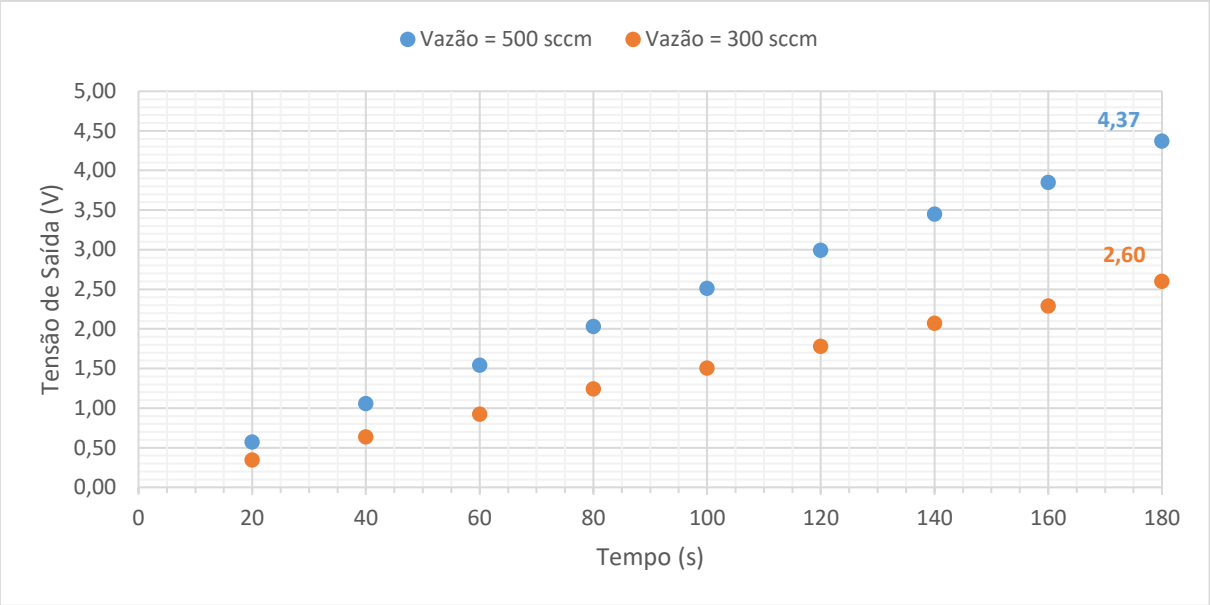

Figura 29: Rampa de subida de tensão.

Simulação de experimento completo

Para simular um experimento completo, foram escolhidos os seguintes parâmetros[2]:

Vazão H_2	300 sccm
Rampa H_2	30 s
Vazão CH_4	500 sccm
Rampa CH_4	30 s
Tempo de Recozimento	60 s
Tempo de Crescimento	60 s
Tempo de Resfriamento	30 s

Tabela 6: Parâmetros utilizados na simulação de um experimento completo.

Como se observa na figura abaixo, as saídas permaneceram ligadas nos intervalos de tempo definidos. A saída 1, correspondente ao MFC da linha de H_2, demorou 30 s para atingir o valor final de 2,60 V, correspondente aos 300 sccm ajustados, e permaneceu com esta tensão durante 2,5 minutos, tempo correspondente à soma dos tempos das 3 etapas. Já a saída 2, correspondente ao MFC da linha de CH_4, começou a elevar sua tensão 30 s antes de finalizar o tempo de recozimento, para que ao final do tempo de rampa e início do tempo de crescimento estivesse

[2] Os tempos escolhidos são bem menores que os tempos reais utilizados em um experimento. Entretanto, para melhor execução do teste e melhor visualização dos resultados, foram escolhidos tais parâmetros.

com sua saída no valor final de 4,37 V, correspondente aos 500 sccm ajustados, permanecendo nesta configuração por 1 min, correspondente ao tempo da etapa 2 do experimento.

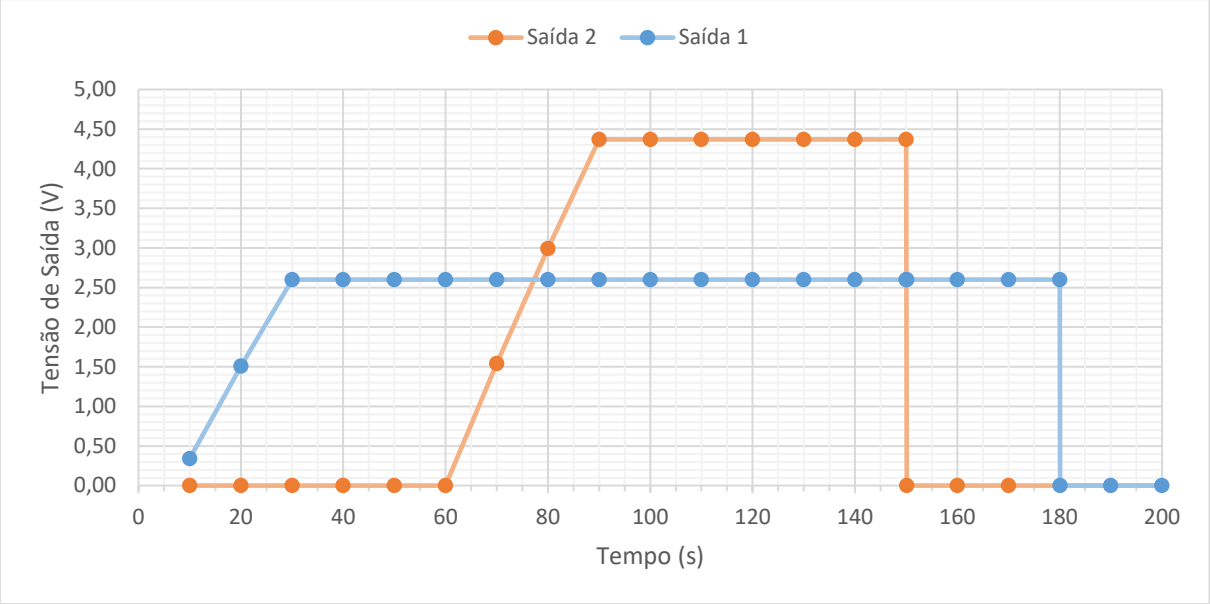

Figura 30: Respostas das duas saídas quando simulado um experimento completo.

Conexão a um MFC

O teste de conexão a um MFC foi iniciado e não pode ser finalizado, pois a pressão dentro do forno começou a aumentar indefinidamente quando o ajuste foi colocado em modo remoto. Embora o controlador estivesse gerando uma saída nula em tensão, provavelmente o MFC não estava interpretando o sinal. Por ser perigoso lidar com o aumento de pressão de gases no reator, este teste foi interrompido e não houve tempo hábil de refazê-lo após ajustes no circuito.

6. CONCLUSÕES

A proposta deste trabalho foi desenvolver um dispositivo de interface com o usuário que permitisse a escolha de uma vazão de gás e fosse capaz de gerar uma saída em tensão correspondente ao valor escolhido, para, então, ser usado como controlador no modo remoto de dois MFCs que controlam a injeção de precursores em um reator que produz grafeno por CVD. Com este intuito, foram desenvolvidos um circuito eletrônico, uma interface com usuário, composta por um display LCD e um conjunto de botões, e um código para a placa Arduino.

Tanto o circuito quanto o código projetado exercem a função de ajuste de vazão, gerando uma saída em tensão proporcional à vazão definida pelo usuário. Além disso, foi possível implementar o controle de parâmetros de tempo, como o tempo de rampa para cada saída e o tempo de duração de cada etapa do experimento.

No que diz respeito à conexão com o MFC, aconteceram imprevistos e erros nos ajustes de parâmetros que impediram o sucesso dos testes de controle.

De maneira geral, pode-se concluir que a solução desenvolvida é válida e apresenta bom comportamento no que diz respeito à geração do sinal de controle, necessitando ainda de ajustes na fase de conexão ao MFC.

7. SUGESTÕES PARA FUTUROS TRABALHOS

Algumas melhorias que podem ser implementadas neste projeto:

- Ajustar os parâmetros de conexão do dispositivo projetado com os MFCs;
- Transferir o dispositivo da protoboard para uma placa de circuito impressa;
- Integrar o controle de vazão com os controles de temperatura e pressão.

REFERÊNCIAS

[1] K. S. Novoselov, V. I. Fal′ko, L. Colombo, P. R. Gellert, S. M. G. e K. Kim, "A roadmap for graphene," *Nature,* vol. 490, pp. 192-200, 2012.

[2] A. A. Balandin, "Thermal properties of graphene and nanostructured carbon materials," *Nature Materials,* vol. 10, pp. 569-581, 2011.

[3] Z. S. F. Bonaccorso, T. Hasan e A. C. Ferrari, "Graphene photonics and optoelectronics," *Nature Photonics,* vol. 4, pp. 611-622, 2010.

[4] C. Lee, X. Wei, J. W. Kysar e J. Hone, "Measurement of the elastic properties and intrinsic strength of monolayer graphene," *Science,* vol. 321, pp. 385-388, 2008.

[5] T. O. Feijó, *Dissertação de Mestrado - Crescimento de grafeno por CVD e sua interação físico-química com hidrogênio,* Porto Alegre, 2017.

[6] A. K. Geim e K. S. Novoselov, "The rise of graphene," *Nature Materials,* vol. 6, pp. 183-191, 2007.

[7] A. H. C. Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov e A. K. Geim, "The electronic properties of graphene," *Reviews of Modern Physics,* vol. 81, pp. 109-162, 2009.

[8] L. Colombo, R. M. Wallace e R. S. Ruoff, "Graphene growth and device integration," *Proceedings of the IEEE,* vol. 101, pp. 1536-1556, 2013.

[9] K. S. Novoselov, "Graphene: Materials in the Flatland (Nobel Lecture)," *Angew. Chem. Int.,* vol. 50, pp. 6986-7002, 2011.

[10] E.-Y. Choi, W. S. Choi, Y. B. Lee e Y.-Y. Noh, "Production of graphene by exfoliation of graphite in a volatile organic solvent," *Nanotechnology,* vol. 22, pp. 1-6, 2011.

[11] K. P. Loh, Q. Bao, P. K. Anga e J. Yanga, "The chemistry of graphene," *Journal of Materials Chemistry,* vol. 20, pp. 2277-2289, 2010.

[12] M. Sprinkle, J. Hicks, A. Tejeda, A. Taleb-Ibrahimi, P. L. Fèvre, F. Bertran, H. Tinkey, M. C. Clark, P. Soukiassian, D. Martinotti, J. Hass e E. H. Conrad, "Multilayer epitaxial graphene grown on the SiC surface: structure and electronic properties," *J. Phys. D: Appl. Phys,* vol. 43, p. 374006, 2010.

[13] S. Bhaviripudi, X. Jia, M. S. Dresselhaus e J. Kong, “Role of kinetic factors in chemical vapor deposition synthesis of uniform large area graphene using copper catalyst,” *Nano Letters,* vol. 10, pp. 4128-4133, 2010.

[14] M. P. Lavin-Lopez, J. L. Valverde, M. C. Cuevas, A. Garrido, L. Sanchez-Silva, P. Martinezb e A. Romero-Izquierdo, “Synthesis and characterization of graphene: influence of synthesis variables,” *Phys. Chem. Chem. Phys.,* vol. 16, pp. 2962-2970, 2014.

[15] X. Dai, J. Wu, Z. Qian, H. Wang, J. Jian, Y. Cao, M. H. Rummeli, Q. Yi, H. Liu e G. Zou, “Ultra-smooth glassy graphene thin films for flexible transparent circuits,” *Science Advances,* vol. 2, p. e1601574, 2016.

[16] Graphene-Info, “Dongxu unveils an impressive graphene-enhanced battery,” [Online]. Available: https://www.graphene-info.com/dongxu-unveils-impressive-graphene-enhanced-battery. [Acesso em 15 Dezembro 2017].

[17] G. K. Dimitrakakis, E. Tylianakis e G. E. Froudakis, “Pillared Graphene: A New 3-D Network Nanostructure for Enhanced Hydrogen Storage,” *Nano Letters,* vol. 8, nº 10, pp. 3166-3170, 2008.

[18] Graphene-Info, “Graphene-perovskite large area solar cell achieves record efficiency,” [Online]. Available: https://www.graphene-info.com/graphene-perovskite-large-area-solar-cell-achieves-record-efficiency. [Acesso em 15 Dezembro 2017].

[19] Omega Engineering, Inc., *FMA 5400A/FMA 5500A Mass Flow Controllers User's Guide,* 2009.

[20] Omega Engineering, Inc., “Thermal Mass Flow Working Principle, Theory and Design,” [Online]. Available: http://www.omega.com/technical-learning/thermal-mass-flow-working-principle-theory-and-design.html. [Acesso em 9 Julho 2017].

[21] Arduino, “Arduino Software (IDE),” [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Acesso em 13 Outubro 2017].

[22] Arduino, “Arduino Mega 2560,” [Online]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Acesso em 14 Outubro 2017].

[23] Arduino, “AnalogWrite,” [Online]. Available: https://www.arduino.cc/en/Reference/AnalogWrite. [Acesso em 14 Dezembro 2017].

[24] Arduino, “Arduino - PWM,” [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Acesso em 5 Dezembro 2017].

[25] F. L. R. Mussoi, *Resposta em Frequência de Filtros Passivos,* Florianópolis: CEFET/SC - Gerência Educacional de Eletrônica, 2004.

[26] Arduino, “LiquidCrystal Library,” [Online]. Available: https://www.arduino.cc/en/Reference/LiquidCrystal. [Acesso em 13 Outubro 2017].

[27] Arduino, “Millis,” [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/millis/. [Acesso em 16 Dezembro 2017].

[28] M. McRoberts, Arduino Básico, São Paulo: Novatec, 2011.

ANEXO: CÓDIGO

```
//=============================================================================================//
// UNIVERISIDADE FEDERAL DO RIO GRANDE DO SUL                                                  //
// TRABALHO DE DIPLOMAÇÃO EM ENGENHARIA FÍSICA II - 2017/2                                     //
//                                                                                             //
// Automatização do sistema de liberação de precursores em um reator CVD utilizado para       //
// produção de grafeno.                                                                        //
//                                                                                             //
// Autora: Laís Rosa de Oliveira                                                               //
// Orientador: Prof. Dr. Gabriel Vieira Soares                                                 //
//=============================================================================================//

// ================================== BIBLIOTECAS ==========================================//
//                                                                                             //
#include <LiquidCrystal.h>   // biblioteca do display LCD                                     //
//                                                                                             //
//=============================================================================================//

// ================================== HARDWARE =============================================//
//                                                                                             //
#define aj1 28               // botão de ajuste 1 no pino 28                                  //
#define aj2 29               // botão de ajuste 2 no pino 29                                  //
#define aj3 30               // botão de ajuste 3 no pino 30                                  //
#define nex 31               // botão next no pino 31                                         //
#define sel 32               // botão select no pino 32                                       //
#define sta 33               // botão start no pino 33                                        //
//                                                                                             //
LiquidCrystal screen(22,     // display LCD   // RS no pino 22                               //
                     23,                     // EN no pino 23                                 //
                     24,                     // D4 no pino 24                                 //
                     25,                     // D5 no pino 25                                 //
                     26,                     // D6 no pino 26                                 //
                     27);                    // D7 no pino 27                                 //
//                                                                                             //
//=============================================================================================//

// ============================== VARIÁVEIS GLOBAIS ========================================//
//                                                                                             //
boolean       flag1 = 0,          // flag do botão de ajuste 1                                //
              flag2 = 0,          // flag do botão de ajuste 2                                //
              flag3 = 0,          // flag do botão de ajuste 3                                //
              flagsel = 0,        // flag do botão select                                     //
              flagsta = 0,        // flag do botão start                                      //
              flagnex = 0,        // flag do botão next                                       //
              flagmode = 0;       // flag do modo de ajuste                                   //
char          menu = 0;           // armazena o valor do menu atual                           //
int           flow = 0,           // armazena o valor dos fluxos em sccm                      //
              hour = 0,           // armazena a componente de horas do tempo                  //
              minute = 0,         // armazena a componente de minutos do tempo                //
              second = 0,         // armazena a componente de segundos do tempo               //
              second_r = 0,       // armazena os segundos da rampa                            //
              flow1 = 0,          // fluxo do gás 1 em sccm                                   //
              flow2 = 0,          // fluxo do gás 2 em sccm                                   //
              hour1 = 0,          // componente de horas do tempo de recozimento              //
              minute1 = 0,        // componente de minutos do tempo de recozimento            //
              second1 = 0,        // componente de segundos do tempo de recozimento           //
              hour2 = 0,          // componente de horas do tempo de crescimento              //
              minute2 = 0,        // componente de minutos do tempo de crescimento            //
              second2 = 0,        // componente de segundos do tempo de crescimento           //
              hour3 = 0,          // componente de horas do tempo de resfriamento             //
              minute3 = 0,        // componente de minutos do tempo de resfriamento           //
              second3 = 0,        // componente de segundos do tempo de resfriamento          //
              second_r1 = 0,      // tempo em segundos de rampa do gás 1                      //
              second_r2 = 0;      // tempo em segundos de rampa do gás 2                      //
//                                                                                             //
//=============================================================================================//

// ============================ DECLARAÇÃO DE FUNÇÕES ======================================//
//                                                                                             //
void readbut();      // leitura dos botões de ajuste                                         //
void setflow1();     // define o valor do fluxo do gás 1                                     //
void setflow2();     // define o valor do fluxo do gás 2                                     //
void rampa1();       // define o tempo em segundos até atingir o fluxo do gás 1              //
void rampa2();       // define o tempo em segundos até atingir o fluxo do gás 2              //
```

```
void time1();         // define o valor do tempo de recozimento                          //
void time2();         // define o valor do tempo do crescimento                          //
void time3();         // define o valor do tempo de resfriamento                         //
void menus();         // troca de menu quando o botão next é pressionado                 //
void show();          // mostra menu atual                                               //
void msg();           // mostra a mensagem da tela inicial                               //
void start();         // inicia o experimento                                            //
void outpwm();        // define as saídas pwm para os MFCs                               //
//                                                                                       //
//=======================================================================================//

// ============================ CONFIGURAÇÃO INICIAL ======================================//
//                                                                                       //
void setup() {                                                                           //
  //                                                                                     //
  screen.begin(16, 2);            // inicializa o display LCD                          //
  msg();                          // mostra mensagem inicial                           //
  for (int i = 28; i < 34; i++)                                                        //
    pinMode(i, INPUT_PULLUP);     // define os pinos 28 a 33 como entrada              //
  for (int i = 11; i < 13; i++)   // usa o resistor interno de pull-up                 //
    pinMode(i, OUTPUT);           // define os pinos 11 e 12 como saída                //
}                                                                                        //
//                                                                                       //
//=======================================================================================//

// ================================== LOOP ===============================================//
//                                                                                       //
void loop() {                                                                            //
  //                                                                                     //
  menus();                                                                             //
  show();                                                                              //
}                                                                                        //
//                                                                                       //
//=======================================================================================//

// ============================== MENSAGEM INICIAL ========================================//
//                                                                                       //
void msg() {                                                                             //
  //                                                                                     //
  screen.setCursor(1, 0);                                                              //
  screen.print("Ajuste MFC v.1");  // mensagem inicial na linha 1                       //
  screen.setCursor(2, 1);                                                              //
  screen.print("LASIS/UFRGS");     // mensagem inicial na linha 2                      //
}                                                                                        //
//                                                                                       //
//=======================================================================================//

// ============================== CONTROLE DOS MENUS ======================================//
//                                                                                       //
void menus() {                                                                           //
  //                                                                                     //
  if (!digitalRead(nex)) flagnex = 1;     // se next for pressionado, seta a flag em 1  //
  if (digitalRead(nex) && flagnex) {      // se next for solto e a flag estiver setada  //
    flagnex = 0;                          // limpa flag                                 //
    screen.clear();                       // limpa display                              //
    menu++;                               // incrementa menu                            //
    if (menu > 7) menu = 1;               // lógica de menus circulares                 //
  }                                                                                      //
  if (!digitalRead(sta)) flagsta = 1;     // se start for pressionado, seta a flag em 1 //
  if (digitalRead(sta) && flagsta) {      // se start for solto e a flag estiver setada //
    flagsta = 0;                          // limpa flag                                 //
    screen.clear();                       // limpa display                              //
    menu = 8;                             // seta o menu em 8 que chama a função start  //
  }                                                                                      //
}                                                                                        //
//                                                                                       //
//=======================================================================================//

// ============================== EXIBIÇÃO DOS MENUS ======================================//
//                                                                                       //
void show() {                                                                            //
  //                                                                                     //
  switch (menu) {                                                                      //
    case 1: setflow1(); break;      // 1° menu: função de ajuste do fluxo do gás 1     //
    case 2: rampa1(); break;        // 2° menu: função de ajuste da rampa do gás 1     //
    case 3: setflow2(); break;      // 3° menu: função de ajuste do fluxo do gás 2     //
```

```
    case 4: rampa2(); break;        // 4° menu: função de ajuste da rampa do gás 2          //
    case 5: time1(); break;         // 5° menu: função de ajuste do tempo de recozimento    //
    case 6: time2(); break;         // 6° menu: função de ajuste do tempo de crescimento    //
    case 7: time3(); break;         // 7° menu: função de ajuste do tempo de resfriamento   //
    case 8: start(); break;         // 8° menu: função de início do experimento             //
  }                                                                                         //
}                                                                                           //
//                                                                                          //
//==========================================================================================//

// ========================= LEITURA DOS BOTÕES DE AJUSTE ================================//
//                                                                                          //
void readbut() {                                                                            //
  //                                                                                        //
  if (!digitalRead(aj1)) flag1 = 1;         // testa se o botão aj1 foi pressionado       //
  if (digitalRead(aj1) && flag1) {          // testa se o botão aj1 foi pressionado e solto //
    flag1 = 0;                              // reinicializa a flag                        //
    flow += 100;                            // incrementa 100 na variável flow            //
    if (flow > 500) flow = 0;               // define o fluxo máximo de 500 sccm          //
    minute += 10;                           // incrementa 10 na varável minute            //
    if (minute >= 60) {                                                                   //
      hour += 1;                            // se minute = 60, incremeta hour em 1        //
      minute -= 60;                         // e reinicia minute                          //
    }                                                                                     //
  }                                                                                       //
  if (!digitalRead(aj2)) flag2 = 1;         // testa se o botão aj2 foi pressionado       //
  if (digitalRead(aj2) && flag2) {          // testa se o botão aj2 foi pressionado e solto //
    flag2 = 0;                              // reinicializa a flag                        //
    flow += 10;                             // incrementa 10 na variável flow             //
    if (flow > 500) flow = 0;               // define o fluxo máximo de 500 sccm          //
    minute += 1;                            // incrementa 1 na varável minute             //
    if (minute >= 60) {                                                                   //
      hour += 1;                            // se minute = 60, incremeta hour em 1        //
      minute -= 60;                         // e reinicia minute                          //
    }                                                                                     //
  }                                                                                       //
  if (!digitalRead(aj3)) flag3 = 1;         // testa se o botão aj3 foi pressionado       //
  if (digitalRead(aj3) && flag3) {          // testa se o botão aj3 foi pressionado e solto //
    flag3 = 0;                              // reinicializa a flag                        //
    flow += 1;                              // incrementa 1 na variável flow              //
    if (flow > 500) flow = 0;               // define o fluxo máximo de 500 sccm          //
    second += 5;                            // incrementa 5 na variável second            //
    second_r += 5;                          // incrementa 5 na varável second para rampa  //
    if (second >= 60) {                                                                   //
      minute += 1;                          // se second = 60, incrementa 1 em minute     //
      second = 0;                           // e reinicia second                          //
    }                                                                                     //
  }                                                                                       //
  if (!digitalRead(sel)) flagsel = 1;       // testa se select foi pressionado            //
  if (digitalRead(sel) && flagsel) {        // testa se select foi pressionado e solto    //
    flagsel = 0;                            // limpa a flag                               //
    flagmode = !flagmode;                   // encerra modo de ajuste                     //
  }                                                                                       //
}                                                                                         //
//                                                                                        //
//========================================================================================//

// ==================== DEFINIÇÃO DOS TEMPOS DO EXPERIMENTO ==============================//
//                                                                                        //
// time1: define o tempo de recozimento do substrato sob fluxo de H2                     //
void time1() {                                                                            //
  //                                                                                      //
  screen.setCursor(0, 0);                                                                 //
  screen.print(" *Recozimento*");         // cabeçalho da tela de ajuste                  //
  screen.setCursor(0, 1);                                                                 //
  screen.print("Tempo = ");                                                               //
  if (!flagmode) {                        // modo de ajuste não selecionado               //
    if (!digitalRead(sel)) flagsel = 1;   // testa select para abrir o modo de ajuste     //
    if (digitalRead(sel) && flagsel) {    // se o botão select for pressionado e solto    //
      flagsel = 0;                        // limpa a flag                                 //
      flagmode = !flagmode;               // troca o estado da flag de ajuste             //
      hour = 0;                           // zera o valor de leitura dos botões           //
      minute = 0;                         // zera o valor de leitura dos botões           //
      second = 0;                         // zera o valor de leitura dos botões           //
    }                                                                                     //
    screen.setCursor(8, 1);                                                               //
```

```
    if (hour1 < 10) screen.print("0");    // mostra o tempo no formato 00:00:00          //
    screen.print(hour1);                  // mostra o valor atual                        //
    screen.print(":");                                                                   //
    if (minute1 < 10) screen.print("0");  // mostra o tempo no formato 00:00:00          //
    screen.print(minute1);                // mostra o valor atual                        //
    screen.print(":");                                                                   //
    if (second1 < 10) screen.print("0");  // mostra o tempo no formato 00:00:00          //
    screen.print(second1);                // mostra o valor atual                        //
  }                                                                                      //
  else {                                  // modo de ajuste selecionado                  //
    screen.setCursor(8, 1);                                                              //
    if (hour1 < 10) screen.print("0");    // mostra o tempo no formato 00:00:00          //
    screen.print(hour1);                  // mostra o valor atual                        //
    screen.print(":");                                                                   //
    if (minute1 < 10) screen.print("0");  // mostra o tempo no formato 00:00:00          //
    screen.print(minute1);                // mostra o valor atual                        //
    screen.print(":");                                                                   //
    if (second1 < 10) screen.print("0");  // mostra o tempo no formato 00:00:00          //
    screen.print(second1);                // mostra o valor atual                        //
    for (int i = 0; i < 25; i++) {                                                       //
      readbut();                          // lê o valor selecionado nos botões           //
      hour1 = hour;                       // atribui o valor da leitura à variável       //
      minute1 = minute;                   // atribui o valor da leitura à variável       //
      second1 = second;                   // atribui o valor da leitura à variável       //
      delay(10);                                                                         //
    }                                                                                    //
    screen.setCursor(8, 1);                                                              //
    screen.print("  :  :  ");             // pisca a tela                                //
    for (int i = 0; i < 25; i++) {                                                       //
      readbut();                          // continua a leitura dos botões               //
      hour1 = hour;                       // atribui o valor da leitura à variável       //
      minute1 = minute;                   // atribui o valor da leitura à variável       //
      second1 = second;                   // atribui o valor da leitura à variável       //
      delay(10);                                                                         //
    }                                                                                    //
  }                                                                                      //
}                                                                                        //
//.....................................................................................//
//                                                                                       //
// time2: define o tempo de crescimento de grafeno em CH4                                //
void time2() { // mesma lógica de time1() adaptada para as variáveis de time2()        //
  //                                                                                     //
  screen.setCursor(0, 0);                                                              //
  screen.print(" *Crescimento*");                                                      //
  screen.setCursor(0, 1);                                                              //
  screen.print("Tempo = ");                                                            //
  if (!flagmode) {                                                                     //
    if (!digitalRead(sel)) flagsel = 1;                                                //
    if (digitalRead(sel) && flagsel) {                                                 //
      flagsel = 0;                                                                     //
      flagmode = !flagmode;                                                            //
      hour = 0;                                                                        //
      minute = 0;                                                                      //
      second = 0;                                                                      //
    }                                                                                  //
    screen.setCursor(8, 1);                                                            //
    if (hour2 < 10) screen.print("0");                                                 //
    screen.print(hour2);                                                               //
    screen.print(":");                                                                 //
    if (minute2 < 10) screen.print("0");                                               //
    screen.print(minute2);                                                             //
    screen.print(":");                                                                 //
    if (second2 < 10) screen.print("0");                                               //
    screen.print(second2);                                                             //
  }                                                                                    //
  else {                                                                               //
    screen.setCursor(8, 1);                                                            //
    if (hour2 < 10) screen.print("0");                                                 //
    screen.print(hour2);                                                               //
    screen.print(":");                                                                 //
    if (minute2 < 10) screen.print("0");                                               //
    screen.print(minute2);                                                             //
    screen.print(":");                                                                 //
    if (second2 < 10) screen.print("0");                                               //
    screen.print(second2);                                                             //
    for (int i = 0; i < 25; i++) {                                                     //
```

```
      readbut();                                                                    //
      hour2 = hour;                                                                 //
      minute2 = minute;                                                             //
      second2 = second;                                                             //
      delay(10);                                                                    //
    }                                                                               //
    screen.setCursor(8, 1);                                                         //
    screen.print("  :  :  ");                                                       //
    for (int i = 0; i < 25; i++) {                                                  //
      readbut();                                                                    //
      hour2 = hour;                                                                 //
      minute2 = minute;                                                             //
      second2 = second;                                                             //
      delay(10);                                                                    //
    }                                                                               //
  }                                                                                 //
}                                                                                   //
//.....................................................................................//
//                                                                                   //
// time3: define o tempo de resfriamento da amostra sob fluxo de H2                 //
void time3()  { // mesma lógica de time1() adaptada para as variáveis de time3()    //
  //                                                                                //
  screen.setCursor(0, 0);                                                           //
  screen.print(" *Resfriamento*");                                                  //
  screen.setCursor(0, 1);                                                           //
  screen.print("Tempo = ");                                                         //
  if (!flagmode) {                                                                  //
    if (!digitalRead(sel)) flagsel = 1;                                             //
    if (digitalRead(sel) && flagsel) {                                              //
      flagsel = 0;                                                                  //
      flagmode = !flagmode;                                                         //
      hour = 0;                                                                     //
      minute = 0;                                                                   //
      second = 0;                                                                   //
    }                                                                               //
    screen.setCursor(8, 1);                                                         //
    if (hour3 < 10) screen.print("0");                                              //
    screen.print(hour3);                                                            //
    screen.print(":");                                                              //
    if (minute3 < 10) screen.print("0");                                            //
    screen.print(minute3);                                                          //
    screen.print(":");                                                              //
    if (second3 < 10) screen.print("0");                                            //
    screen.print(second3);                                                          //
  }                                                                                 //
  else {                                                                            //
    screen.setCursor(8, 1);                                                         //
    if (hour3 < 10) screen.print("0");                                              //
    screen.print(hour3);                                                            //
    screen.print(":");                                                              //
    if (minute3 < 10) screen.print("0");                                            //
    screen.print(minute3);                                                          //
    screen.print(":");                                                              //
    if (second3 < 10) screen.print("0");                                            //
    screen.print(second3);                                                          //
    for (int i = 0; i < 25; i++) {                                                  //
      readbut();                                                                    //
      hour3 = hour;                                                                 //
      minute3 = minute;                                                             //
      second3 = second;                                                             //
      delay(10);                                                                    //
    }                                                                               //
    screen.setCursor(8, 1);                                                         //
    screen.print("  :  :  ");                                                       //
    for (int i = 0; i < 25; i++) {                                                  //
      readbut();                                                                    //
      hour3 = hour;                                                                 //
      minute3 = minute;                                                             //
      second3 = second;                                                             //
      delay(10);                                                                    //
    }                                                                               //
  }                                                                                 //
}                                                                                   //
//                                                                                  //
//==================================================================================//
```

```
// ========================== DEFINIÇÃO DOS TEMPOS DO RAMPA ==============================//
//                                                                                        //
void rampa1() {                                                                           //
  //                                                                                      //
  screen.setCursor(0, 0);                                                                 //
  screen.print("Ajuste MFC1: H2");              // cabeçalho da tela de ajuste          //
  screen.setCursor(1, 1);                                                                 //
  screen.print("Rampa = ");                                                               //
  if (!flagmode) {                             // modo de ajuste não selecionado       //
    if (!digitalRead(sel)) flagsel = 1;        // testa select para abrir o modo de ajuste //
    if (digitalRead(sel) && flagsel) {         // se o botão select for pressionado e solto //
      flagsel = 0;                             // limpa a flag                         //
      flagmode = !flagmode;                    // troca o estado da flag de ajuste     //
      second = 0;                              // zera o valor de leitura dos botões   //
    }                                                                                     //
    screen.setCursor(9, 1);                                                               //
    if (second_r1 < 100 && second_r1 >= 10)    // se o valor de fluxo tiver dois dígitos //
      screen.print(" ");                       // mantém o valor ajustado à direita    //
    if (second_r1 < 10)                        // se o valor do fluxo tiver um dígito  //
      screen.print("  ");                      // mantém o valor ajustado à direita    //
    screen.print(second_r1);                   // mostra o valor atual do gás 1        //
    screen.print(" s");                        // mostra a unidade                     //
  }                                                                                       //
  else {                                       // modo de ajuste inicializado          //
    screen.setCursor(9, 1);                                                               //
    if (second_r1 < 100 && second_r1 >= 10)    // se o valor de fluxo tiver dois dígitos //
      screen.print(" ");                       // mantém o valor ajustado à direita    //
    if (second_r1 < 10)                        // se o valor do fluxo tiver um dígito  //
      screen.print("  ");                      // mantém o valor ajustado junto à direita //
    screen.print(second_r1);                   // mostra o valor atual                 //
    screen.print(" s");                        // mostra a unidade                     //
    for (int i = 0; i < 25; i++) {                                                        //
      readbut();                               // lê o valor selecionado nos botões    //
      second_r1 = second_r;                    // atribui o valor selecionado à variável //
      delay(10);                                                                          //
    }                                                                                     //
    screen.setCursor(9, 1);                                                               //
    screen.print("     ");                    // pisca a tela                         //
    for (int i = 0; i < 25; i++) {                                                        //
      readbut();                               // contiinua a leitura dos botões       //
      second_r1 = second_r;                    // atribui o valor da leitura à variável //
      delay(10);                                                                          //
    }                                                                                     //
  }                                                                                       //
}                                                                                         //
//.......................................................................................//
//                                                                                        //
void rampa2() { // mesma lógica de rampa1() adaptada com as variáveis de rampa2()       //
  //                                                                                      //
  screen.setCursor(0, 0);                                                                 //
  screen.print("Ajuste MFC2: CH4");                                                       //
  screen.setCursor(1, 1);                                                                 //
  screen.print("Rampa = ");                                                               //
  if (!flagmode) {                                                                        //
    if (!digitalRead(sel)) flagsel = 1;                                                   //
    if (digitalRead(sel) && flagsel) {                                                    //
      flagsel = 0;                                                                        //
      flagmode = !flagmode;                                                               //
      second = 0;                                                                         //
    }                                                                                     //
    screen.setCursor(9, 1);                                                               //
    if (second_r2 < 100 && second_r2 >= 10)                                               //
      screen.print(" ");                                                                  //
    if (second_r2 < 10)                                                                   //
      screen.print("  ");                                                                 //
    screen.print(second_r2);                                                              //
    screen.print(" s");                                                                   //
  }                                                                                       //
  else {                                                                                  //
    screen.setCursor(9, 1);                                                               //
    if (second_r2 < 100 && second_r2 >= 10)                                               //
      screen.print(" ");                                                                  //
    if (second_r2 < 10)                                                                   //
      screen.print("  ");                                                                 //
    screen.print(second_r2);                                                              //
    screen.print(" s");                                                                   //
```

```
    for (int i = 0; i < 25; i++) {                                                    //
      readbut();                                                                      //
      second_r2 = second_r;                                                           //
      delay(10);                                                                      //
    }                                                                                 //
    screen.setCursor(9, 1);                                                           //
    screen.print("     ");                                                            //
    for (int i = 0; i < 25; i++) {                                                    //
      readbut();                                                                      //
      second_r2 = second_r;                                                           //
      delay(10);                                                                      //
    }                                                                                 //
  }                                                                                   //
}                                                                                     //
//                                                                                    //
//====================================================================================//

// ================== FUNÇÕES QUE DEFINEM OS FLUXOS DO EXPERIMENTO ======================//
//                                                                                    //
void setflow1() {                                                                     //
  //                                                                                  //
  screen.setCursor(0, 0);                                                             //
  screen.print("Ajuste MFC1: H2");          // cabeçalho da tela de ajuste          //
  screen.setCursor(0, 1);                                                             //
  screen.print("Fluxo = ");                                                           //
  if (!flagmode) {                         // modo de ajuste não selecionado       //
    if (!digitalRead(sel)) flagsel = 1;    // testa select para abrir o modo de ajuste //
    if (digitalRead(sel) && flagsel) {     // se o botçao select for pressionado e solto //
      flagsel = 0;                         // limpa a flag                         //
      flagmode = !flagmode;                // troca o estado da flag de ajuste     //
      flow = 0;                            // zera o valor de leitura dos botões   //
    }                                                                                 //
    screen.setCursor(8, 1);                                                           //
    if (flow1 < 100 && flow1 >= 10)        // se o valor de fluxo tiver dois dígitos //
      screen.print(" ");                   // mantém o valor ajustado à direita da tela //
    if (flow1 < 10)                        // se o valor do fluxo tiver um dígito  //
      screen.print("  ");                  // mantém o valor ajustado à direita da tela //
    screen.print(flow1);                   // mostra o valor atual do gás 1        //
    screen.print(" sccm");                 // mostra a unidade                     //
  }                                                                                   //
  else {                                   // modo de ajuste inicializado          //
    screen.setCursor(8, 1);                                                           //
    if (flow1 < 100 && flow1 >= 10)        // se o valor de fluxo tiver dois dígitos //
      screen.print(" ");                   // mantém o valor ajustado à direita da tela //
    if (flow1 < 10)                        // se o valor de fluxo tiver um dígito  //
      screen.print("  ");                  // mantém o valor ajustado à direita da tela //
    screen.print(flow1);                   // mostra o valor do gás 1 sendo atualizado //
    screen.print(" sccm");                 // mostra a unidade                     //
    for (int i = 0; i < 25; i++) {                                                    //
      readbut();                           // lê o valor selecionado nos botões    //
      flow1 = flow;                        // atribui valor da leitura à variável  //
      delay(10);                                                                      //
    }                                                                                 //
    screen.setCursor(8, 1);                                                           //
    screen.print("        ");             // pisca a tela                         //
    for (int i = 0; i < 25; i++) {                                                    //
      readbut();                           // continua a leitura                   //
      flow1 = flow;                        // atribui valor da leitura à variável  //
      delay(10);                                                                      //
    }                                                                                 //
  }                                                                                   //
}                                                                                     //
//....................................................................................//
//                                                                                    //
void setflow2() { // mesma lógica de setflow1() adaptada com as variáveis do gás 2    //
  //                                                                                  //
  screen.setCursor(0, 0);                                                             //
  screen.print("Ajuste MFC2: CH4");                                                   //
  screen.setCursor(0, 1);                                                             //
  screen.print("Fluxo = ");                                                           //
  if (!flagmode) {                                                                    //
    if (!digitalRead(sel)) flagsel = 1;                                               //
    if (digitalRead(sel) && flagsel) {                                                //
      flagsel = 0;                                                                    //
      flagmode = !flagmode;                                                           //
      flow = 0;                                                                       //
```

```
    }                                                                                                    //
    screen.setCursor(8, 1);                                                                              //
    if (flow2 < 100 && flow2 >= 10)                                                                      //
      screen.print(" ");                                                                                 //
    if (flow2 < 10)                                                                                      //
      screen.print("  ");                                                                               //
    screen.print(flow2);                                                                                 //
    screen.print(" sccm");                                                                               //
  }                                                                                                      //
  else {                                                                                                 //
    screen.setCursor(8, 1);                                                                              //
    if (flow2 < 100 && flow2 >= 10)                                                                      //
      screen.print(" ");                                                                                 //
    if (flow2 < 10)                                                                                      //
      screen.print("  ");                                                                               //
    screen.print(flow2);                                                                                 //
    screen.print(" sccm");                                                                               //
    for (int i = 0; i < 25; i++) {                                                                       //
      readbut();                                                                                         //
      flow2 = flow;                                                                                      //
      delay(10);                                                                                         //
    }                                                                                                    //
    screen.setCursor(8, 1);                                                                              //
    screen.print("        ");                                                                            //
    for (int i = 0; i < 25; i++) {                                                                       //
      readbut();                                                                                         //
      flow2 = flow;                                                                                      //
      delay(10);                                                                                         //
    }                                                                                                    //
  }                                                                                                      //
}                                                                                                        //
//                                                                                                       //
//=======================================================================================================//

//============================ FUNÇÃO QUE INICIA O EXPERIMENTO ==============================//
//                                                                                                       //
void start() {                                                                                           //
  //                                                                                                     //
  unsigned long time_rampa1_ms = 1000 * second_r1,                                                       //
                time_rampa2_ms = 1000 * second_r2,                                                       //
                time_recoz_ms = 1000 * ((hour1 * 60 * 60) + (minute1 * 60) + second1),                   //
                time_cresc_ms = 1000 * ((hour2 * 60 * 60) + (minute2 * 60) + second2),                   //
                time_resfr_ms = 1000 * ((hour3 * 60 * 60) + (minute3 * 60) + second3);                   //
  int flowpwm1 = map(flow1, 0, 500, 0, 255),                                                             //
      flowpwm2 = map(flow2, 0, 500, 0, 255);                                                             //
  screen.setCursor(0, 0);                                                                                //
  screen.print("  Ok, Iniciar?");                                                                        //
  screen.setCursor(0, 1);                                                                                //
  screen.print("S-->SEL N-->NEX");                                                                       //
  if (!digitalRead(sel)) flagsel = 1;                                                                    //
  if (digitalRead(sel) && flagsel) {                                                                     //
    flagsel = 0;                                                                                         //
    menu = 0;                                                                                            //
    screen.clear();                                                                                      //
    screen.setCursor(0, 0);                                                                              //
    screen.print("Ajuste realizado");                                                                    //
    screen.setCursor(2, 1);                                                                              //
    screen.print("com sucesso!");                                                                        //
    unsigned long zero_ms = millis();                                                                    //
    outpwm(flowpwm1, flowpwm2,                                                                           //
           time_rampa1_ms, time_rampa2_ms,                                                               //
           time_recoz_ms, time_cresc_ms, time_resfr_ms, zero_ms);                                        //
  }                                                                                                      //
}                                                                                                        //
//                                                                                                       //
//=======================================================================================================//

//======================== AJUSTE DA SAÍDA PWM COM RAMPA ================================//
//                                                                                                       //
void outpwm(int pwm1,                                                                                    //
            int pwm2,                                                                                    //
            unsigned long t_ramp1, unsigned long t_ramp2,                                                //
            unsigned long t_int1, unsigned long t_int2,                                                  //
            unsigned long t_int3, unsigned long M0) {                                                    //
  unsigned long t_now,                                                                                   //
           M1 = M0 + t_ramp1,                                                                            //
```

```
          M2 = M1 + t_int1 - t_ramp2,                                                    //
          M3 = M1 + t_int1,                                                              //
          M4 = M3 + t_int2,                                                              //
          M5 = M4 + t_int3;                                                              //
  int pwm_ramp1, pwm_ramp2;                                                              //
  t_now = millis();                                                                      //
  while (t_now < M1) {                                                                   //
    pwm_ramp1 = pwm1 * (t_now - start) / t_ramp1 ;                                       //
    analogWrite(11, pwm_ramp1);                                                          //
    analogWrite(12, 0);                                                                  //
    t_now = millis();                                                                    //
  }                                                                                      //
  t_now = millis();                                                                      //
  while (t_now < M2) {                                                                   //
    analogWrite(11, pwm1);                                                               //
    analogWrite(12, 0);                                                                  //
    t_now = millis();                                                                    //
  }                                                                                      //
  t_now = millis();                                                                      //
  while (t_now < M3) {                                                                   //
    pwm_ramp2 = pwm2 * ( t_now - (t_ramp1 + start + t_int1 - t_ramp2)) / t_ramp2;        //
    analogWrite(11, pwm1);                                                               //
    analogWrite(12, pwm_ramp2);                                                          //
    t_now = millis();                                                                    //
  }                                                                                      //
  t_now = millis();                                                                      //
  while (t_now < M4) {                                                                   //
    analogWrite(11, pwm1);                                                               //
    analogWrite(12, pwm2);                                                               //
    t_now = millis();                                                                    //
  }                                                                                      //
  t_now = millis();                                                                      //
  while (t_now < M5) {                                                                   //
    analogWrite(11, pwm1);                                                               //
    analogWrite(12, 0);                                                                  //
    t_now = millis();                                                                    //
  }                                                                                      //
}                                                                                        //
//                                                                                       //
//=======================================================================================//
```