

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL ESPINOLA LINCOLN FERREIRA DOS SANTOS

**DESENVOLVIMENTO DE UM PROTÓTIPO PARA CONTROLE DE FLUXO DE
MASSA E LEITURA DE PRESSÃO DE GASES DURANTE O TRATAMENTO
TÉRMICO DE MATERIAIS**

CURITIBA

2022

GABRIEL ESPINOLA LINCOLN FERREIRA DOS SANTOS

**DESENVOLVIMENTO DE UM PROTÓTIPO PARA CONTROLE DE FLUXO DE
MASSA E LEITURA DE PRESSÃO DE GASES DURANTE O TRATAMENTO
TÉRMICO DE MATERIAIS**

**Design of a prototype for mass flow control and gas pressure reading during
the heat treatment of materials**

Trabalho de Conclusão de Curso apresentado como requisito à obtenção do título de Bacharel em Engenharia Mecânica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador(a): Prof. Dr. Márcio Mafra
Coorientador(a): Prof. Dr. Euclides Alexandre Bernardelli.

CURITIBA

2022



[4.0 Internacional](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Esta licença permite download e compartilhamento do trabalho desde que sejam atribuídos créditos ao(s) autor(es), sem a possibilidade de alterá-lo ou utilizá-lo para fins comerciais. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

GABRIEL ESPINOLA LINCOLN FERREIRA DOS SANTOS

**DESENVOLVIMENTO DE UM PROTÓTIPO PARA CONTROLE DE FLUXO DE
MASSA E LEITURA DE PRESSÃO DE GASES DURANTE O TRATAMENTO
TÉRMICO DE MATERIAIS**

Trabalho de Conclusão de Curso de Graduação
apresentado como requisito para obtenção do título de
Bacharel em Engenharia Mecânica da Universidade
Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 14/junho/2022

Márcio Mafra

Doutorado em Ciência e Engenharia dos Materiais
Universidade Tecnológica Federal do Paraná

Euclides Alexandre Bernardelli

Doutorado em Ciência e Engenharia dos Materiais
Universidade Tecnológica Federal do Paraná

Sidney Carlos Gasoto

Mestre em Engenharia de Produção e Sistemas
Universidade Tecnológica Federal do Paraná

Walter Luís Mikos

Doutorado em Engenharia Mecânica
Universidade Tecnológica Federal do Paraná

CURITIBA

2022

Dedico este trabalho à minha família, amigos e professores que me auxiliaram nessa trajetória.

RESUMO

A engenharia de superfícies e tratamento de materiais tem uma crescente demanda por processos que garantam componentes com alterações significativas nas propriedades mecânicas, térmicas e químicas. Esses processos exigem controle constante dos parâmetros de fluxo de massa e pressão dos gases presentes no sistema. Este trabalho tem como objetivo desenvolver um protótipo para controle de sistemas com controladores de fluxo de massa de gases MKS 500 SCCM e leitura de sensores de pressão Pirani APGX-M-NW16/ALI Edwards que se adapta aos gases utilizados e as calibrações dos sensores. Além disso, forneça um método de armazenamento dos dados de medição dos sensores, *offline* e *online*. Sendo o método online, através das plataformas *Google Sheets* e *Grafana* e os método *offline* através do cartão microSD. O equipamento é manipulado via uma tela sensível ao toque onde o usuário pode configurar os sensores utilizados. O projeto aplica metodologias de desenvolvimento de produtos, tratamento de materiais, eletrônica, processamento de sinais e programação.

Palavras-chave: controle de fluxo de massa; leitura de pressão; tratamento de materiais; pirani.

ABSTRACT

There is a growing demand for processes of materials science, that can induce engineering materials to alter their mechanical, thermal, and chemical properties. These processes require constant parameter's control of the mass flow and pressure of the gases in the system. This project aims to develop a prototype to control systems with mass flow controllers MKS 500 SCCM and Pirani APGX-M-NW16/ALI Edwards pressure sensor reading. Also, it stores the measurement data from sensors by offline and online protocol. The online method is by Google Sheets and Grafana platforms and the offline method is through the microSD card. The equipment is manipulated by a touch screen where the user can configure the sensors. The research project surveys methodologies for product development, material handling, electronics, signal processing and programming.

Keywords: mass flow control; pressure reading; treatment of materials; pirani.

LISTA DE ILUSTRAÇÕES

Figura 1 - Nitretação a plasma em peça	14
Figura 2 - Diagrama esquemático do reator de plasma de baixa pressão	15
Figura 3 - Controlador de fluxo de massa.....	16
Figura 4 - Sensor Pirani	16
Figura 5– Controlador de alimentação 247D	17
Figura 6 - Estrutura interna MFC.....	19
Figura 7 - Sensor de taxa de fluxo	20
Figura 8 - Estrutura sensor pirani.....	20
Figura 9 - Pressão medida sobre por pressão real para diferentes gases	21
Figura 10 – Controlador de alimentação MKS	23
Figura 11 - Painel de controle frontal	24
Figura 12 - Painel de controle traseiro	24
Figura 13 – Controlador de alimentação Brooks Instrument	25
Figura 14 - Vista frontal controlador Brooks Instrument.....	26
Figura 15 - Vista traseira controlador Brooks Instrument	26
Figura 16 - Diagrama redução de ruídos por passa-baixa (a) entrada de sinal <i>vin</i> , (b) processo de média, (c) saída filtrada <i>vo</i>	27
Figura 17 - Filtro passa-baixa RC.....	28
Figura 18 - Curva de resposta de frequência para filtro passa-baixas	28
Figura 19 - Arduino mega	29
Figura 20 - ESP8266	30
Figura 21 – ESP32	30
Figura 22 - Esquema banco de dados relacional	32
Figura 23 - Esquema de banco de dados não relacional	32
Figura 24 - Esquema armazenamento de valores separados por vírgula	33
Figura 25 - Ciclo desenvolvimento DevOps.....	34
Figura 26 - Controle de versão Github	35
Figura 27 - Esquema de versionamento de código.....	36
Figura 28 – Esquemático simplificado do sistema de tratamento de superfície	37
Figura 29 - Sistema com o controlador de módulos	38
Figura 30 - Esquemático Hardware.....	39
Figura 31 – Análise de erro ADC1 (GPIO36) para 11 dB (0 – 3900 mV) e Wi-Fi ligado.....	40
Figura 32 – Leitura osciloscópio x Leitura ESP32.....	41
Figura 33 - Circuito leitura Pirani	42
Figura 34 - Circuito controle e leitura MFC	43
Figura 35 – Display Nextion 320x240 pixels.....	44
Figura 36 – Módulo leitura e escrita em cartão microSD	45
Figura 37 - Módulo RTC ds1302	46
Figura 38 - Fonte Knup 24V	47
Figura 39 - Módulo regulador de tensão LM2596	47
Figura 40 – Esquema eletrônico	48
Figura 41 – Rotas placa de circuito impresso	49
Figura 42 – Renderização da PCB	49
Figura 43 – Montagem gabinete e tampa	50
Figura 44 – Gabinete CAD	50

Figure 45 – Tampa CAD	51
Figura 46 – Posicionamento e configuração Cura 3D.....	51
Figura 47 - Parâmetros de impressão	52
Figura 48 – Montagem do protótipo	52
Figura 49 – Diagrama de sequência	53
Figura 50 - Diagrama de coleta de dados.....	54
Figura 51- Mapa da interface gráfica	54
Figura 52 – Menu de configurações “Settings”	55
Figura 53 – Menu de gases cadastrados.....	55
Figura 54 – Menu de controle dos sensores “Free Mode”	56
Figura 55 - Estrutura dos arquivos	57
Figura 56 - Transformação de tensão para pressão	61
Figura 57 - Esquema de abertura e fechamento.....	63
Figura 58 - Servidor <i>web</i> gerenciador de <i>WiFi</i>	64
Figura 59 - Painilha recebendo dados do ESP01	66
Figura 60 - Apps Scripts	66
Figura 61 - <i>Dashboard</i> no Grafana.....	68
Figura 62 - Consulta planilha	68
Figura 63 - Modelos de gráfico.....	69
Figura 64 – Especificações multímetro para tensão contínua	70
Figura 65 – Preparação do laboratório para realização do teste	70
Figura 66 - Leitura da pressão no reator com fluxo constante de hidrogênio à 250 <i>SCCM</i> durante 50 segundos.....	71
Figura 67 – Leitura de pressão no reator com fluxo constante de hidrogenio à 250 <i>SCCM</i> durante 35 segundos.....	71
Figura 68 – Precisão da medição do Pirani.....	72
Figura 69 - Função contínua variação de abertura de válvula	73
Figura 70 - Resultados individuais da função contínua.....	72
Figura 71 - Função pulsada 165 segundos	74
Figura 72 - <i>Dashboard</i> Grafana.....	74
Figura 73 - Taxa de aquisição x Número de MFC’s aquisitando	76

Código 1 - Função " <i>MFCPwnON</i> "	57
Código 2 - Função " <i>MFCPwmClose</i> "	58
Código 3 - Função " <i>MFCRead</i> "	58
Código 4 - Função " <i>PiraniRead</i> "	59
Código 5 - Função " <i>piraniUpdateScreen</i> "	61
Código 6 - Modo contínua	62
Código 7 - Modo pulsado.....	63
Código 8 - Função enviar para a API.....	65
Código 9 - <i>Google Sheets API</i>	66

Quadro 1 - Benchmarking de controladores.....	22
Quadro 2– Comparação Arduino Mega x ESP32	40
Quadro 3 – Pinagem RJ45 Pirani Edwards	42
Quadro 4 – Sensores Pirani compatíveis.....	43
Quadro 5 - Pinagem MFC DB9.....	43
Quadro 6 – Compatibilidade de MFC.....	44

Quadro 7 - Mapeamento de erros	61
Quadro 8 - Dados coletados pelo ESP01	75
Quadro 9 - Custo do projeto	77
Quadro 10 - Comparação entre soluções.....	77

LISTA DE ABREVIATURAS E SIGLAS

3D	Tridimensional
ASP	<i>Analog Signal Processor</i>
CSV	<i>Comma-separated values</i>
GUI	<i>Graphical user interface</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of things</i>
MFC	<i>Mass Flow Controller</i>
NoSQL	<i>Not Only Structured Query Language</i>
PCB	<i>Printed circuit board</i>
PWM	<i>Pulse Width Modulation</i>
RTC	<i>Real-time clock</i>
SCCM	<i>Standard Cubic Centimeters Per Minute</i>
SLM	<i>Standard Liter Per Minute</i>
SPI	<i>Serial Peripheral Interface</i>
SQL	<i>Structured Query Language</i>
TCC	Trabalho de conclusão de curso
UTFPR	Universidade Tecnológica Federal do Paraná

LISTA DE SÍMBOLOS

A	Ampere
Hz	Hertz
V	Volts
Ω	Ohms
kb	Kilobit
Mbps	Megabit por segundos

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Caracterização do problema e justificativa	17
1.2	Objetivos	17
2	REFERÊNCIAL TEÓRICO	19
2.1	Controladores de fluxo de massa	19
2.2	Pirani	20
2.3	Controladores de alimentação e leitura	22
2.3.1	<i>Benchmarking</i>	22
2.3.2	MKS modelo 247	23
2.3.3	<i>Brooks</i> modelo 254	24
2.4	Processamento de sinal analógico	27
2.5	Microcontrolador	29
2.5.1	Arduino	29
2.5.2	ESP01	30
2.5.3	ESP32	30
2.6	Internet das coisas (IoT)	31
2.7	Armazenamento de dados	31
2.7.1	Banco de dados relacional (SQL)	31
2.7.2	Banco de dados não relacional (NoSQL)	32
2.7.3	Valores separados por vírgula (CSV)	33
3	PROCEDIMENTOS METODOLÓGICOS	34
3.1	Metodologia de projeto	34
3.2	Controle de versão de código	35
3.3	Planejamento	36
4	DESENVOLVIMENTO DO PROTÓTIPO	39
4.1	Hardware	39
4.1.1	Microcontrolador	40
4.1.2	Circuito de leitura de pressão	41
4.1.3	Circuito de leitura e controle do fluxo de massa	43
4.1.4	Registro de dados e visualização	44
4.2	Firmware	53
4.2.1	Nextion – Interface gráfica	54
4.2.2	Arduino Mega	56

4.2.3	ESP01 – conexão <i>Wi-Fi</i>	64
4.3	Software	65
4.4	Testes do protótipo	69
4.4.1	Leitura da pressão	70
4.4.2	Leitura e controle do fluxo de massa	72
4.4.3	Módulo <i>Wi-Fi</i>	74
4.4.4	Visão geral	75
4.4.5	Custo do projeto	76
5	CONSIDERAÇÕES FINAIS	76
5.1	Trabalhos futuros	80
	REFERÊNCIAS.....	81
	APÊNDICE A – Circuito eletrônico.....	84
	APÊNDICE B – Quadro de fator de correção dos gases	87
	APÊNDICE C – Modelo 3D do case e arquivos da PCB	89
	APÊNDICE D – Código Firmware e Software	91
	APÊNDICE E – Modelo do usuário.....	94

1 INTRODUÇÃO

A crescente demanda por componentes com alterações significativas nas propriedades mecânicas, térmicas e químicas, como mais resistência ao desgaste e corrosão, faz com que empresas e pesquisadores busquem novas formas de suprir essas necessidades. Tradicionalmente, busca-se o desenvolvimento de novas ligas como forma de solução. Porém, o desenvolvimento de novas ligas exige custo e tempo, que muitas vezes se torna o fator decisivo para não prosseguir com as pesquisas. Logo, a engenharia de superfícies e materiais vêm para suprir esses requisitos de alteração nas propriedades tribológicas e triboquímicas (PINEDO, 2004).

A nitretação é um exemplo de processo que visa a melhoria das propriedades de superfície (Figura 1). O tratamento consiste em submeter o material a um ambiente de baixa pressão na presença de uma mistura gasosa contendo nitrogênio (HULTMAN, 2001). É aplicada uma diferença de potencial, através de dois eletrodos, para ionizar a mistura e o substrato é posicionado dentro de uma câmara de vácuo junto ao eletrodo de potencial negativo. Ocorre o aceleração de íons da atmosfera controlada que colidem contra o substrato, promovendo pulverização e aquecimento, o aumento de temperatura associado a presença de espécies nitretantes favorecem a modificação superficial do material ocasionando a alteração da dureza superficial, resistência mecânica, fadiga e outros (XI, 2007).

Figura 1 - Nitretação a plasma em peça



Fonte: Autor (2018)¹

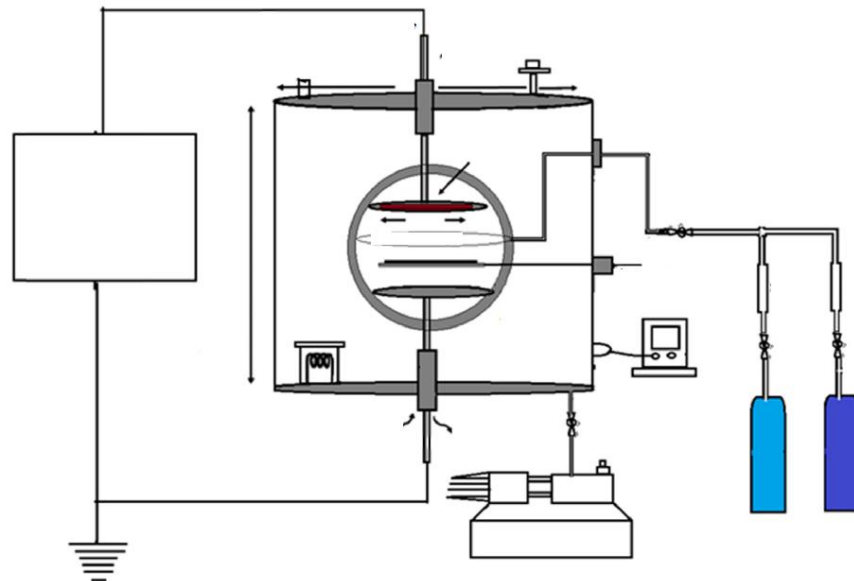
Historicamente, o processo de nitretação a plasma foi desenvolvido por Berghaus no início do século XX. Porém, devido às restrições tecnológicas da época,

¹ Ilustrações e Tabelas sem indicação de fonte são de autoria própria

o processo não atingia os parâmetros fluxo de massa e de pressão dos gases exigidos para o sistema. Assim, essa tecnologia apenas teve pleno desenvolvimento no fim do século XX, possibilitado pelos avanços tecnológicos, principalmente relacionados à eletrônica e automação (PINEDO, 2004).

A Figura 2 apresenta um exemplo de como a tecnologia está presente para o funcionamento desse processo. O esquemático representa um sistema de reator a plasma de baixa pressão, no qual a amostra (*sample*) é posicionada entre os eletrodos e a saída de gás (*gas shower*). O sistema de entrada de gás permite a mistura dos gases controlado através do controlador de fluxo de massa (*Mass flow controller*) ou *MFC*. A câmara é esvaziada pela bomba a vácuo. A pressão na câmara é medida através do sensor Pirani. Um controlador é usado para alimentar o Pirani e os *MFC* (PANDIYARAJ, 2014).

Figura 2 - Diagrama esquemático do reator de plasma de baixa pressão



Fonte: Pandiyaraj (2014)

A Figura 3 apresenta um controlador de fluxo de massa (*MFC*) que permite a introdução precisa de frações volumétricas diversas dos gases - H_2 , N_2 , Ar e O_2 - de modo a compor uma atmosfera gasosa controlada para cada material, sendo crucial para atingir os resultados desejados (HELMERSSON, 2006).

Figura 3 - Controlador de fluxo de massa



Fonte: MKS Instruments (2021)

Juntamente com o *MFC*, esse processo necessita de controle de pressão que é, geralmente, realizado em níveis de vácuo que variam entre 1 Torr e 10 Torr (133,322 Pa e 1.333,22 Pa). Essa pressão pode ser monitorada através de um sensor Pirani, apresentado na Figura 4. (TOPALLI, 2009). Segundo Alves (2001), a pressão de trabalho está diretamente relacionada com as colisões elétrons-íons-átomos, sendo um fator determinante para a realização de tratamentos de nitretação a plasma.

Figura 4 - Sensor Pirani



Fonte: Edwards Vacuum (2021)

O controlador de alimentação, Figura 5, é um *hardware* que funciona como fonte de alimentação, leitura das medições e ajuste das válvulas dos *MFC's* conectados. Logo, o sistema visa facilitar o ajuste de abertura da válvula dos equipamentos, a qual é determinado pela magnitude do sinal emitido pelo controlador para o *MFC*. Além de monitorar os valores medidos por cada equipamento conectado a ele (DILLON, 2001).

Figura 5– Controlador de alimentação 247D



Fonte: MKS Instruments (2021)

Pretende-se nesse projeto desenvolver um protótipo de controlador de alimentação e leitor de módulos *MFC/Pirani*, visando ser uma alternativa nacional do equipamento.

1.1 Caracterização do problema e justificativa

O controlador de alimentação é essencial para gerenciar o funcionamento dos módulos durante o processo de tratamento de superfícies. A necessidade de monitorar e controlar o fluxo e a pressão é de plena importância durante o tratamento de superfícies por gases. Os erros podem trazer resultados completamente diferentes do esperado. Portanto, o presente trabalho identifica a oportunidade de minimizar a interferência humana nesse processo e, assim, automatizar os sensores para melhor performance através de um microcontrolador.

Além disso, atualmente, apenas 2 empresas internacionais -MKS instruments e Brooks instruments- produzem equipamentos como esse que apenas dão suporte para módulos da mesma marca. Esses controladores têm custo elevado, podendo chegar até R\$ 57.000,00. A falta de uma tecnologia nacional faz surgir a oportunidade de desenvolvimento desse sistema com um custo mais acessível.

1.2 Objetivos

O objetivo principal deste trabalho é desenvolver um protótipo de um sistema, composto de *hardware*, *firmware* e *software*, para controle do fluxo de massa dos gases e monitoramento da pressão de gases durante tratamento de materiais.

Os objetivos específicos que devem ser cumpridos para que seja possível alcançar o objetivo geral são:

- Desenvolver um protótipo de um sistema de controle e alimentação de 4 *MFC* MKS 500 SCCM e leitura de pressão através de 1 Pirani APGX-M-NW16/ALI Edwards;
- Desenvolver modos de funcionamento programados, tais como: liga/desliga de medição dos sensores, *pre-sets* de gases comuns e alerta de erro;
- Desenvolver a placa de circuito impresso;
- Desenvolver a coleta de dados dos sensores que tenham resultados do erro de medição e precisão melhor ou equivalente ao multímetro atualmente utilizado no laboratório.

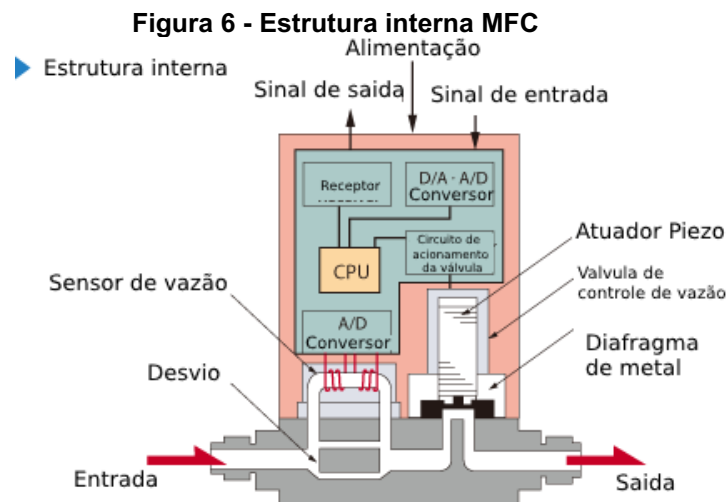
2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos teóricos necessários para o desenvolvimento do trabalho.

2.1 Controladores de fluxo de massa

O controlador de fluxo de massa controla automaticamente a taxa de fluxo de um gás de acordo com o *set point* enviado por um sinal elétrico sem ser afetado pelas condições de ou mudanças na pressão do gás (HORIBA, 2021). O *MFC* pode receber um *set point* de 0 a 100% de escala, mas normalmente é operado em 10% a 90%, onde a melhor precisão é alcançada. O dispositivo controlará a taxa de fluxo até o *set point* fornecido (CHOI, 2013).

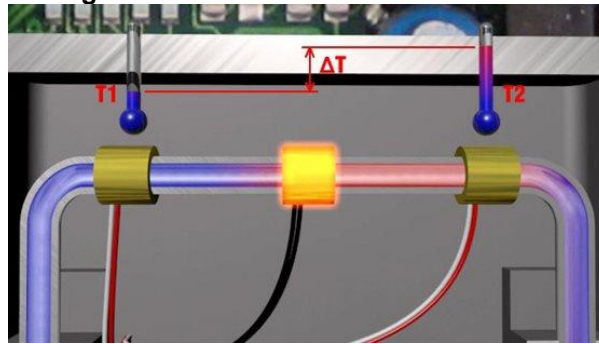
Conforme a Figura 6, a seção onde ocorre a medição de taxa de fluxo inclui um sensor, desvio, válvula de controle de taxa de fluxo e circuitos especiais. O gás é alimentado através do *inlet* e parte do gás flui para o sensor, o qual mede a taxa de fluxo, e outra para a válvula, a qual modifica a taxa de fluxo de modo que a diferença entre a taxa de fluxo medida e a taxa de fluxo recebida do sinal de *set point* seja zero (HORIBA, 2021).



Fonte: Adaptado de Horiba (2013)

O sensor de taxa de fluxo, Figura 7, consiste em um tubo de pequeno diâmetro usado para desviar uma pequena quantidade de gás do fluxo principal. Um aquecedor é instalado entre um par de termômetros. A taxa de fluxo de massa é medida por meio de fenômenos de transferência de calor entre um aquecedor e o gás fluindo dentro do tubo (TISON, 1996).

Figura 7 - Sensor de taxa de fluxo

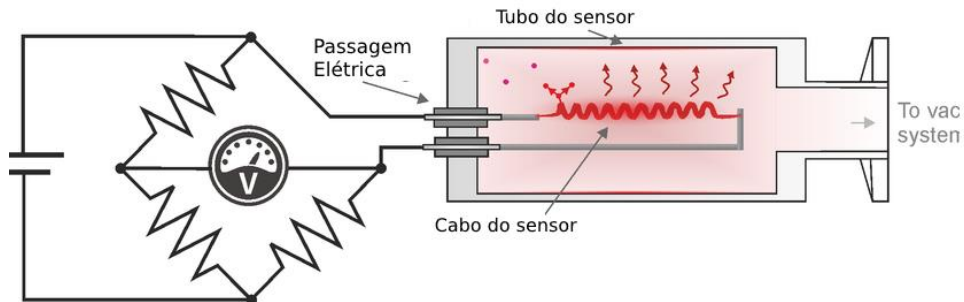


Fonte: Bronkhorst (2021)

2.2 Pirani

O sensor Pirani de vácuo, Figura 8, consiste em um filamento de metal, usualmente de platina, suspenso em um tubo em qual é conectado ao sistema de vácuo. O filamento é conectado a um circuito elétrico que permite a leitura da pressão após a calibração (LAMMERINK, 2001).

Figura 8 - Estrutura sensor pirani

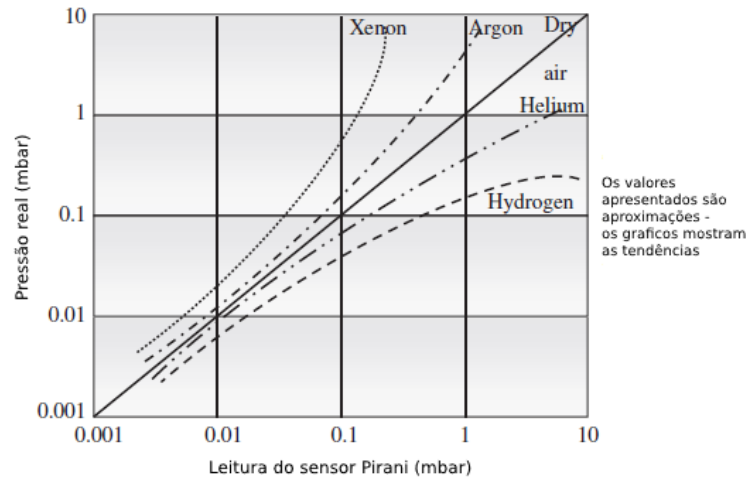


Fonte: Adaptado de Sens4 (2021)

O medidor Pirani mede a condutividade térmica do fio aquecido. Essa condutividade corresponde diretamente a pressão do vácuo para o gás circundante o qual dissipa o calor para a parede do tubo. Na faixa de fluxo molecular, a transferência térmica é proporcional à densidade do número molecular e, portanto, à pressão. Em alta pressão, a alta densidade do gás e o curto caminho livre médio entre as moléculas resultarão em alta condutividade térmica (LAMMERINK, 2001).

Segundo Martini (2012), o filamento do sensor Pirani é operado em uma ponte de Wheatstone como um dos quatro braços da ponte. Como apresentado na Figura 9, as leituras do medidor devem ser corrigidas ou calibradas para diferentes gases. Sendo as curvas definidas pelas taxas de transferência de calor para as moléculas de cada gás.

Figura 9 - Pressão medida sobre por pressão real para diferentes gases



Fonte: Adaptado de Bishop (2015)

Considerando o gráfico apresentado por Bishop (2015), se a temperatura do filamento, que é encontrada na resistência, é mantida constante, sua produção de calor será função da pressão. Porém, não é linear, pois a radiação térmica também influenciará na saída de calor. A taxa de perda total de energia W_T é dada por:

$$W_T = W_R + W_G \quad (2.1)$$

É possível examinar as funções de dependência de cada termo. Sendo dl o comprimento ao longo de um fio de diâmetro D_1 , emissividade ε_1 , e temperatura T_1 localizado ao longo do cilindro de raio r_2 , emissividade ε_2 , temperatura da vizinhança T_2 e σ a constante de Stefan-Boltzmann. Assumindo $r_2 \gg D_1$ e $T_1 > T_2$, a taxa de transferência de calor é dado por:

$$W_R = \varepsilon_1 \sigma (T_1^4 - T_2^4) 2\pi D_1 dl \quad (2.2)$$

Logo, o material ideal para o filamento do sensor apresenta uma emissividade baixa e estável, explicando a recorrência do uso de platina para o projeto desses equipamentos. Assim, fazendo o mesmo para a taxa de transferência de calor para as moléculas de gás, temos:

$$W_G = \frac{1(\gamma + 1)}{4(\gamma - 1)} \alpha \sqrt{\frac{2k}{\pi m T_2}} (T_1 - T_2) p \quad (2.3)$$

Onde $\gamma = C_p/C_v$ é a proporção de calores específicos do gás, m é a massa de uma molécula de gás e k é a constante de Boltzmann. Logo, como mencionado acima, com a equação W_G é possível definir cada curva da Figura 8 para os múltiplos gases (MARTINI, 2012).

2.3 Controladores de alimentação e leitura



O controlador de alimentação e leitura é projetado como fonte de alimentação e leitura de sinal, gerador de sinal de *set point* para controladores de fluxo de massa analógicos (*MFC's*). Em geral, é um instrumento versátil que pode ser usado separadamente ou como parte de um sistema de controle maior (MKS, 2021).

2.3.1 Benchmarking

O *benchmarking* dos controladores de alimentação e leitura similares ao desenvolvido neste trabalho é uma etapa crucial no desenvolvimento do sistema. Através desse, é possível entender e comparar as opções presentes no mercado. Entendendo quais são os pontos fortes e mínimos para o desenvolvimento de uma solução competitiva.

O Quadro 1 compara os 2 principais produtos presente no mercado para venda. Ambos são comercializados internacionalmente e revendidos no Brasil por empresas autorizadas.

Quadro 1 - Benchmarking de controladores

		
Empresa	MKS Instruments	Brook Instrument
Modelo	247	254
Dimensões (mm)	240,5x274,3x88,1	211x113x58.8
Alimentação	100 - 120 Vac	12-24 Vdc ou 100 - 240 Vac
Material case	Aluminio	ABS Cycolac
Número de canais	4	4
Tela	LED 1 linha x digito 3 1/2 0.56"	LCD 8 linhas x 40 digito
Output de sinal	0-5 volts	0-5 volts
Compatibilidade	Modulo de controle massa (apenas MKS Instruments)	Modulo de controle pressao e massa (apenas Brook Instruments)
Armazenamento de dados	Não armazena os dados	"100 anos de retenção"
Funções	Alimentação, leitura e controle de set point	Alimentação, leitura, controle de set point e armazenamento das medicoes
Orçamento	R\$ 21.395,00	R\$ 57.634,50
Revendedor/data Orçamento	Ohmini 21/10/2021	Contech 20/10/2021

2.3.2 MKS modelo 247

Analisando os dados coletados no *Benchmarking*, ambos os controladores são de pequeno porte, comportam até 4 módulos e *set point* de até 5 V baseado no fluxo da medição. Para o equipamento da MKS 247, Figura 10, temos:

Figura 10 – Controlador de alimentação MKS



Fonte: MKS Instruments (2021)

Pontos positivos

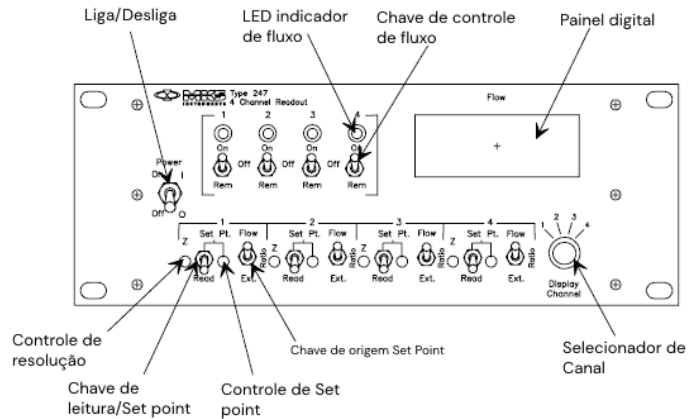
- Preço mais barato comparado ao seu concorrente.

Pontos negativos

- Não armazena e não apresenta nenhuma forma de coletar os dados medidos;
- Tela pequena, apresenta um dado por vez;
- 1 entrada DB9 para o MFC e 1 entrada DB25;
- Compatibilidade apenas com módulos da MKS;
- Apenas compatibilidade com módulos de controle de massa.

Como apresenta a Figura 11, o controle do equipamento da MKS é realizado manualmente. O controlador contém 4 potenciômetros manuais para ajuste do set point (0-5 V) e uma interface digital (*Digital Panel Meter*).

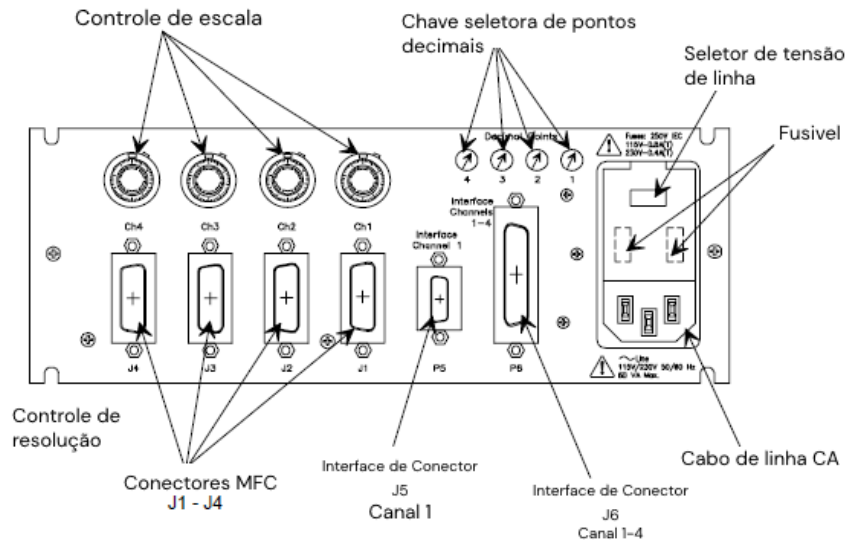
Figura 11 - Painel de controle frontal



Fonte: Adaptado de MKS Instruments (2021)

O painel de controle traseiro, apresentado na Figura 12, há apenas 1 entrada DB9 para o *MFC* e 1 entrada DB25. Além disso, o controlador contém 4 portas DB15. Os quatro potenciômetros de 10 voltas são usados para inserir o fator de escala, que reduz o sinal de saída do transdutor +5 V para que o medidor de painel digital exiba a taxa de fluxo e o ponto de ajuste diretamente em *sccm* (*Standard Cubic Centimeters Per Minute*) ou *slm* (*Standard Liter Per Minute*).

Figura 12 - Painel de controle traseiro



Fonte: Adaptado de MKS Instruments (2021)

2.3.3 Brooks modelo 254

Para o equipamento da Brooks Instrument, Figura 13, temos:

Figura 13 – Controlador de alimentação Brooks Instrument



Fonte: Brooks Instruments (2021)

Pontos positivos

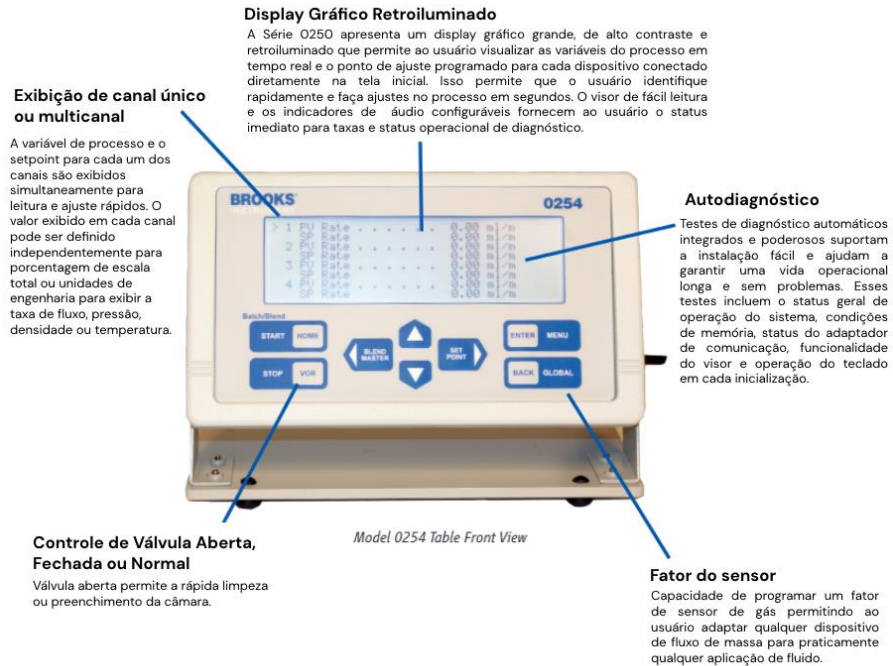
- Portabilidade;
- Tela com exibição dos 4 módulos ao mesmo tempo;
- Armazenamento interno das medições;
- Compatibilidade com módulos de controle de massa e pressão;
- Coleta externa dos dados.

Pontos negativos

- Preço alto;
- Compatibilidade apenas com módulos *Brooks Instrument*;
- Compatibilidade apenas para módulo com entrada DB15.

Como apresentado na Figura 14, o equipamento da *Brooks* é portátil e tem ajuste para uso em diversas posições. O controle do equipamento é digital. A parte frontal contém a tela para leitura das medições e os botões para controle de sensor e *set point*.

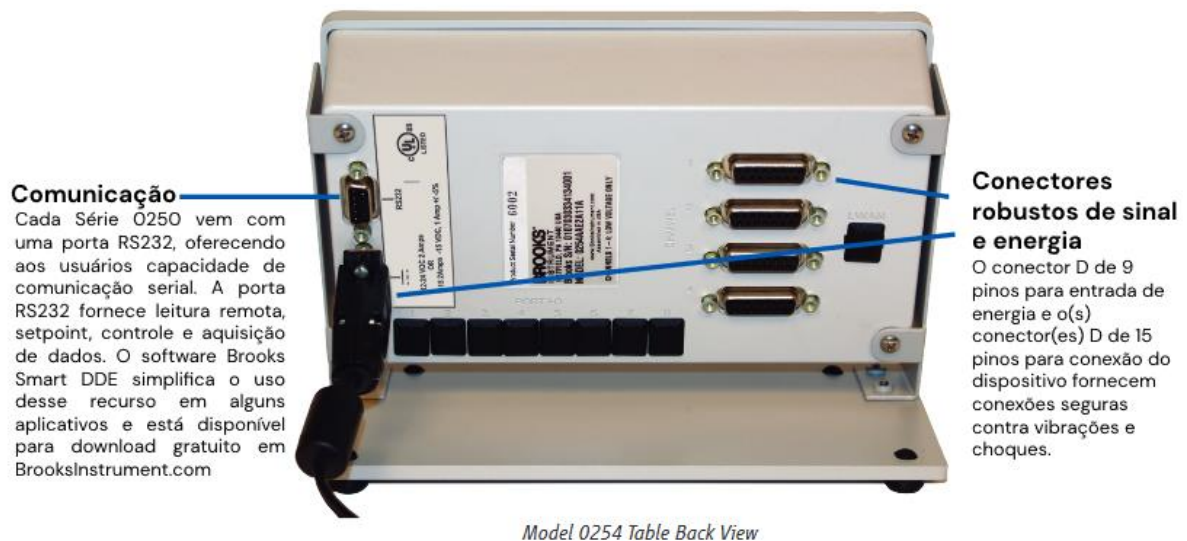
Figura 14 - Vista frontal controlador Brooks Instrument



Fonte: Adaptado de Brooks Instruments (2021)

A vista traseira do controlador, Figura 15, apresenta 4 entradas para módulos com conexão via DB15. A alimentação é feita por um conector DB9. O equipamento permite extrair os dados das medições através de uma comunicação serial.

Figura 15 - Vista traseira controlador Brooks Instrument



Fonte: Adaptado de Brooks Instruments (2021)

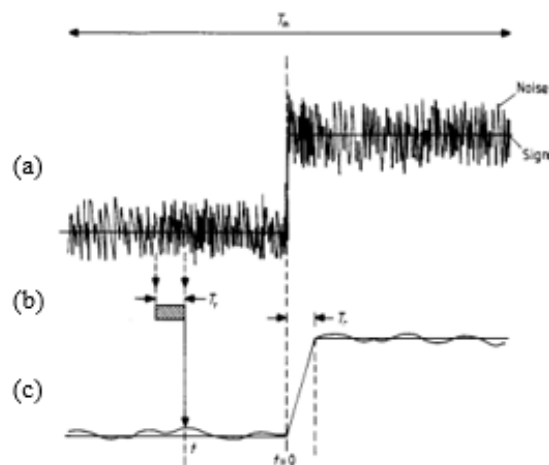
2.4 Processamento de sinal analógico

O processamento de sinal analógico (ASP) é qualquer processamento de sinal conduzido por circuitos analógicos cujo valores são normalmente representados como tensão, corrente elétrica ou carga elétrica em torno dos dispositivos eletrônicos. Um erro ou ruído afetando tais quantidades físicas resultará em um erro correspondente nos sinais representados (EL-SABA, 2015). Pode-se citar como exemplo o filtro analógico elétrico é um exemplo de processamento do sinal, no qual é usado para passar certos comprimentos de onda, ou frequências, e atenuar ou bloquear. (PAARMANN, 2001).

2.4.1.1 Filtro passa-baixa

O objetivo do filtro passa-baixa é reduzir a amplitude dos componentes indesejados do sinal, dos quais o primeiro a ser considerado é o ruído. A Figura 16 ilustra a maneira como o filtro passa-baixa reduz esse ruído (WILMSHURST, 1990).

Figura 16 - Diagrama redução de ruídos por passa-baixa (a) entrada de sinal v_{in} , (b) processo de média, (c) saída filtrada v_o



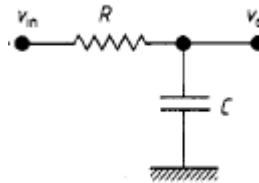
Fonte: Wilmshurst (1990)

Na Figura 15 (a) apresenta a entrada de um sinal v_{in} com ruído. O efeito do filtro é produzir a tensão de saída v_o com ruído reduzido, subsequente da "média contínua" (b), ao longo da tensão de entrada (a). No tempo t , v_o é a média de v_{in} , ao longo do período mostrado de $t - T_r$ até t . No qual, T_r é o tempo finito para o sinal de saída v_o realizar a média. Logo, T_r é o tempo de resposta do filtro. Em termos matemáticos, segundo demonstrado por Wilmshurst (1990), a equação (2.4) representa o sinal de saída v_o .

$$v_o(t) = T_t^{-1} \int_{t-T_r}^t v_{in}(t') dt' \quad (2.4)$$

Eletronicamente, o filtro passa-baixa simples consiste em um resistor em série com um capacitor em paralelo com a carga, como apresenta a Figura 17.

Figura 17 - Filtro passa-baixa RC



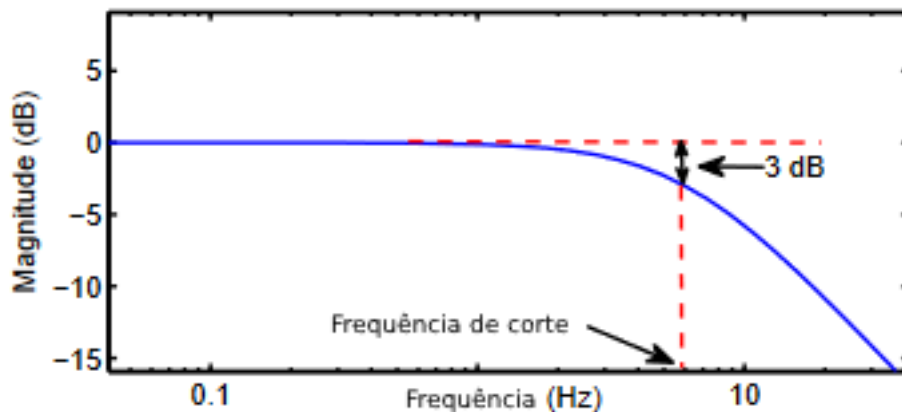
Fonte: Wilmshurst (1990)

O capacitor exibe oposição a corrente alternada e bloqueia os sinais de alta frequência, fazendo com que eles passem pela carga (WILMSHURST, 1990). A frequência de corte é determinada pela escolha da resistência (R) e da capacitância (C):

$$f_c = \frac{1}{2\pi RC} \quad (2.5)$$

Logo, um exemplo de curva de resposta de frequência para o filtro de passa-baixa é apresentado na Figura 18. Em geral, é padronizado um valor de 3 dB para a frequência de corte (WILMSHURST, 1990).

Figura 18 - Curva de resposta de frequência para filtro passa-baixas



Fonte: Adaptado de Qazizadeh (2017)

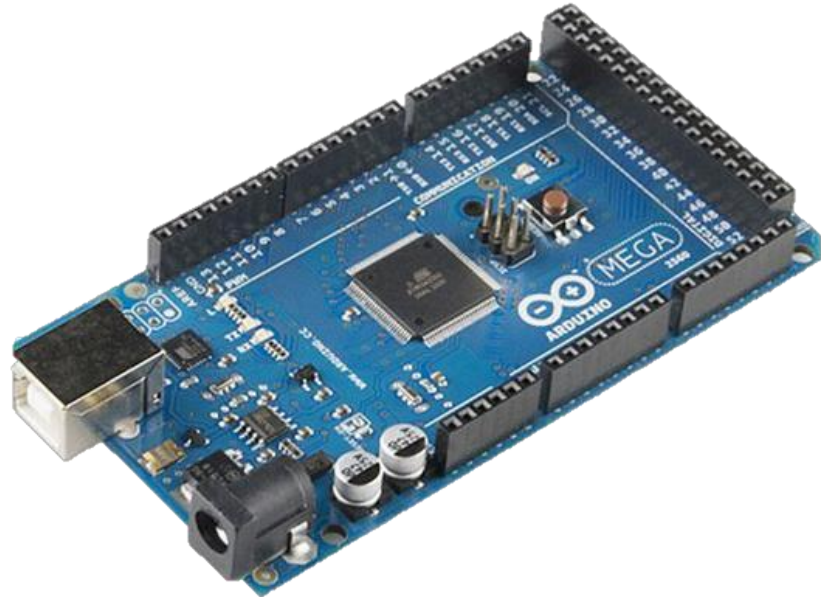
2.5 Microcontrolador

Um microcontrolador é um circuito integrado projetado para realizar uma atividade específica em um sistema embarcado. O processador central do microcontrolador interpreta os dados que recebe de seus periféricos de entrada/saída para realizar a atividade para qual foi programado (KOUTROULIS, 2001).

2.5.1 Arduino

O Arduino, representado na Figura 19, é uma plataforma de prototipagem eletrônica *open-source* que se baseia em *hardware* e *software* flexíveis. O *hardware* é um microcontrolador, em geral, baseado em um processador ATmega. Possui pinos de entrada/saída analógicos e digitais, dos quais podem ser utilizados como saídas de *PWM* (*Pulse Width Modulation*). A utilização do Arduino como produtos/protótipos de IoT, *wearables* e impressoras 3D é bastante comum. Com uma gama de modelos e especificações (ARDUINO, 2021).

Figura 19 - Arduino mega



Fonte: Arduino (2021)

O *software* Arduino é publicado como código aberto. A linguagem de programação é C ++, tendo a possibilidade de ser programado por AVR C linguagem na qual a placa se baseia. (ARDUINO, 2021).

2.5.2 ESP01

O ESP é um micro-chip de baixo custo produzido pela *Espressif Systems* que desempenha a função de um microcontrolador. A Figura 20 apresenta um ESP8266, modelo com um processador L106 de 32bits, memória flash embutida de 1 MiB, módulo *Wi-Fi* e programador funções específicas com conexão TCP/IP (ESPRESSIF, 2021).

Figura 20 - ESP8266



Fonte: Espressif (2021)

A sua linguagem de programação mais popular é C++, possibilitando ser utilizado a IDE (*Integrated Development Environment*) do Arduino, utilizando a comunicação via cabo micro-usb (ESPRESSIF, 2021).

2.5.3 ESP32

O ESP32, Figura 21, é um microcontrolador com conectividade *Wi-Fi* e Bluetooth integrada para uma ampla gama de aplicações. Com 4 MB de memória flash, o ESP32 permite criar variadas aplicações para projetos de IoT, acesso remoto, web servers e dataloggers, entre outros. O ESP32 é criado e desenvolvido pela Espressif Systems, uma empresa chinesa com sede em Xangai (ESPRESSIF, 2021).

Figura 21 – ESP32



Fonte: Espressif (2021)

A sua linguagem de programação mais popular é C++, possibilitando ser utilizado a IDE (*Integrated Development Environment*) do Arduino, utilizando a comunicação via cabo micro-usb (ESPRESSIF, 2021).

2.6 Internet das coisas (IoT)

A internet das coisas (IoT) descreve a rede de objetos físicos – coisas – que estão embarcados com sensores, *softwares* e outras tecnologias com o intuito de conectar e trocar dados com outros dispositivos e sistemas pela internet. Esses objetos variam entre objetos domésticos comuns a ferramentas industriais sofisticadas (ABDUL-QAWAY, 2015).

2.7 Armazenamento de dados

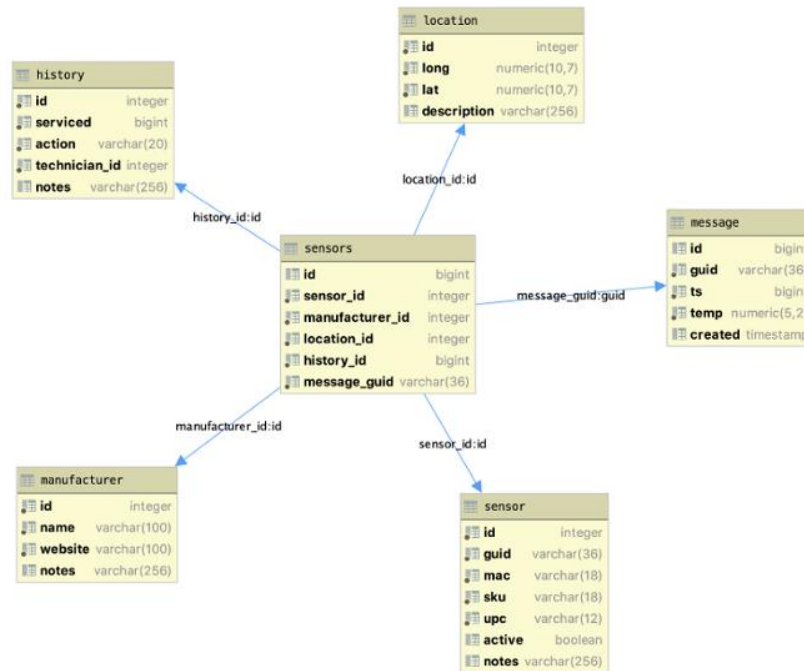
A confiabilidade dos dados é crucial nos sistemas embarcados. Referindo-se a confiabilidade como a capacidade do sistema garantir a integridade e efetividade dos dados mesmo se o sistema sofrer algum dano, como falha de memória, erro humano ou falta de alimentação (CHAO, 2012).

Existem várias formas de armazenar dados, as principais são pela sua estrutura ou pelo tipo de arquivo (texto, imagem, número) que aloca. Os mais comuns para coleta de dados de sensores são: relacional, não relacional e dados separados por vírgula (ORACLE, 2021).

2.7.1 Banco de dados relacional (SQL)

O banco de dados relacional é organizado em tabelas. Os dados têm um relacionamento predefinido entre si em cada tabela. Cada tabela é organizada por colunas e linhas (ORACLE, 2021). As tabelas são usadas para reter a informação sobre os objetos a serem representados no banco de dados. Como apresentado na Figura 22, as linhas na tabela representam uma coleção de valores relacionados de um objeto. (PEREIRA, 2009).

Figura 22 - Esquema banco de dados relacional

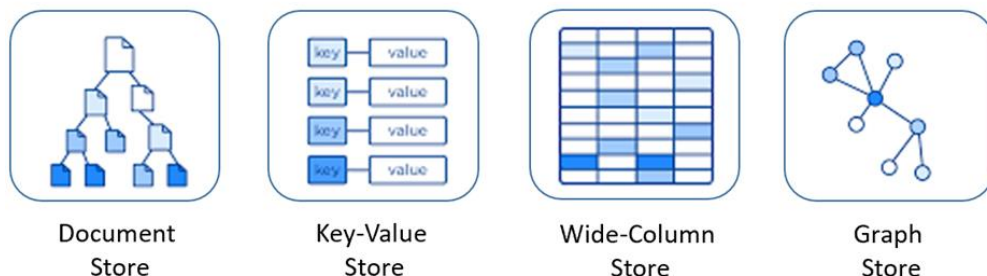


Fonte: Oracle (2021)

2.7.2 Banco de dados não relacional (NoSQL)

O banco de dados não relacional é um banco de dados de código aberto que não possui um esquema de tabelas de linhas e colunas. Como apresentado na Figura 23, ele utiliza um modelo otimizado e escalável para requisitos específicos de dados que serão armazenados, como documentos, orientação para coluna, orientação por gráficos e outros (PLECHAWSKA-WOJCIK, 2017). Logo, a especificidade e escalabilidade contribuem com sua flexibilidade e possibilitando desafios típicos de big data, variedade e velocidade de armazenamento (MICROSOFT, 2021).

Figura 23 - Esquema de banco de dados não relacional



Fonte: Microsoft (2021)

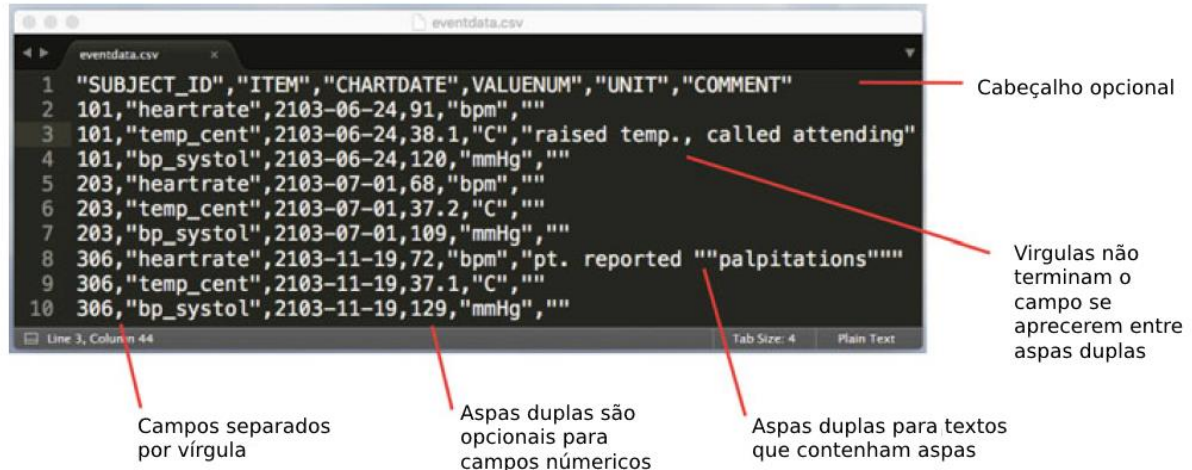
2.7.3 Valores separados por vírgula (CSV)

O formato mais simples e amplo para armazenamento de um conjunto de dados, predominantemente tabulares, é os valores separados por vírgula (CSV). Cada registro está em uma linha e os campos de dados indicados são separados por uma vírgula (AHMED, 2017).

Como apresentado na Figura 24, segundo Carvalho (2015), a recomendação geral é seguir a definição para CSV estabelecida pela *Internet Engineering Task Force*:

- Os registros estão localizados em linha separadas delimitada por uma quebra de linha;
- O último registro do arquivo não pode ter quebra de linha final;
- Uma linha de cabeçalho opcional pode existir na primeira linha do arquivo;
- Os campos que contêm quebras de linha, aspas duplas e vírgulas devem ser colocados entre aspas duplas;

Figura 24 - Esquema armazenamento de valores separados por vírgula



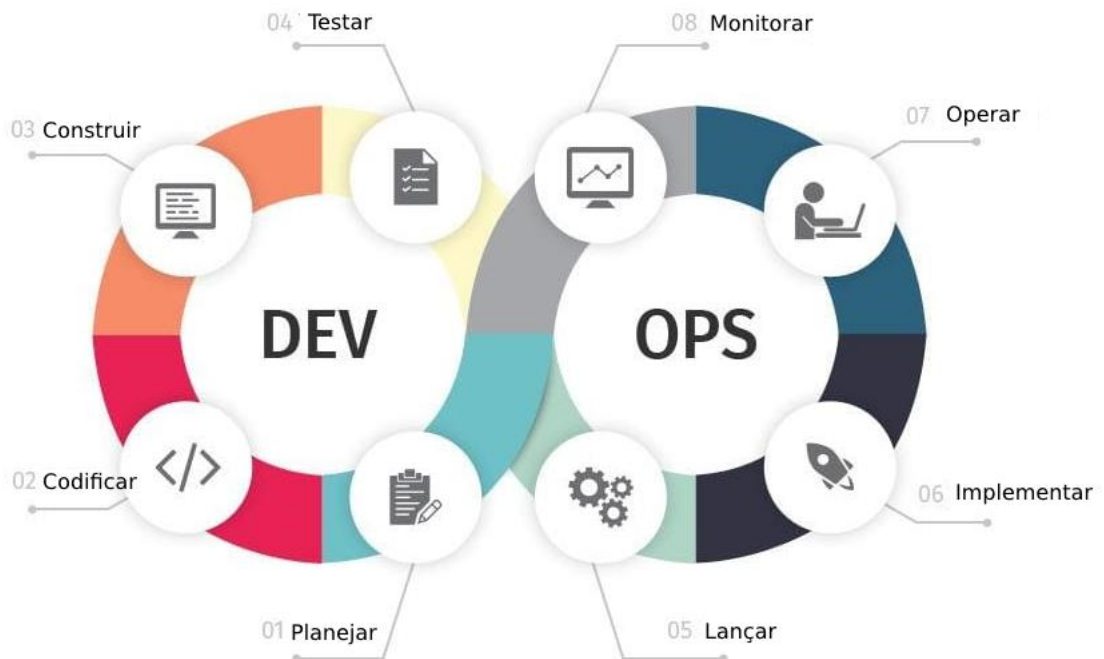
Fonte: Adaptado de Pollard (2016)

3 PROCEDIMENTOS METODOLÓGICOS

3.1 Metodologia de projeto

A metodologia DevOps foi utilizada neste trabalho. O seu objetivo é encurtar o ciclo de vida de desenvolvimento de sistemas e fornecer entrega contínua com qualidade. O DevOps permite que funções anteriormente isoladas atuem de forma coordenada e colaborativa para produzirem produtos melhores e mais confiáveis (MICROSOFT, 2021). O ciclo de vida do DevOps consiste em oito fases (Figura 25):

Figura 25 - Ciclo desenvolvimento DevOps



Fonte: Adaptado de Microsoft (2021)

1. **Planejar:** elaboração de um plano prévio antes dos desenvolvedores começarem a programar. Requisitos e *feedbacks* são coletados das partes interessadas. No caso deste trabalho, foram coletados *feedbacks* dos professores que necessitam do sistema nos laboratórios da UTFPR e dos usuários rotineiros do equipamento;
2. **Codificar:** iniciar a arquitetura do sistema. O diagrama elétrico e a preparação dos códigos são planejados nessa etapa;
3. **Construir:** consiste na montagem do protótipo físico e implementação do código para o microcontrolador ou sistema;

4. **Testar:** teste do protótipo que abrange *hardware*, *firmware* e *software*. A etapa é necessária para entender o funcionamento do sistema e se há necessidade de alterações;

5. **Lançar:** o sistema é disponibilizado para os usuários para testes mais detalhados. Logo, a coleta do *feedback* do usuário do sistema é essencial;

6. **Implementar:** O sistema é colocado em produção de acordo com o *feedback* do cliente. Apesar do lançamento para o usuário final, o sistema ainda continua em observação para melhorias;

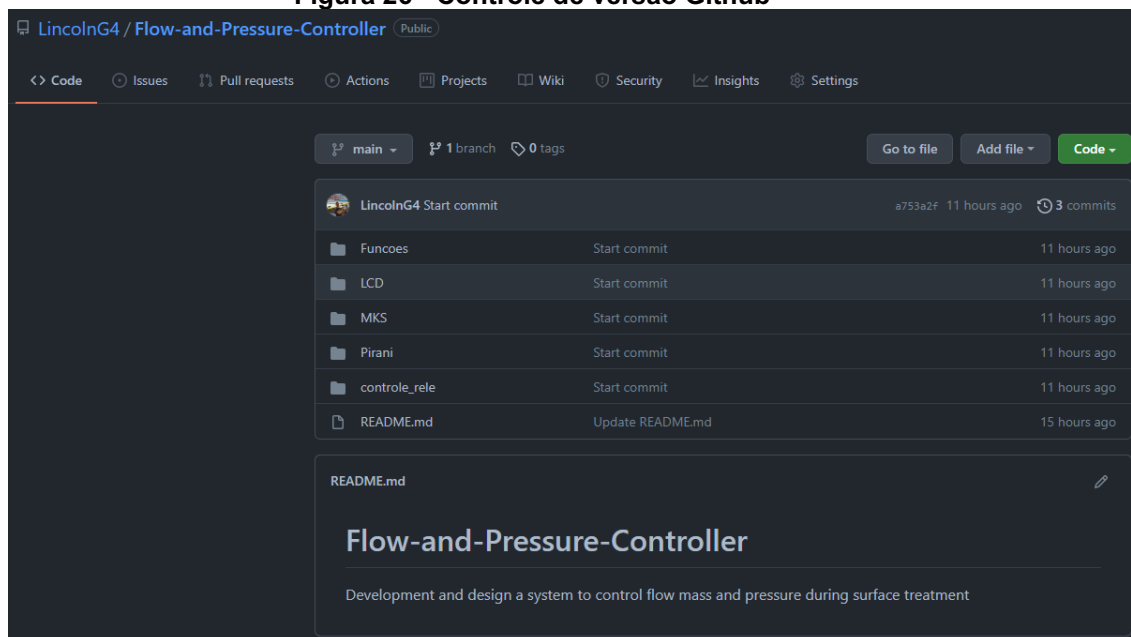
7. **Operar:** o desenvolvedor constantemente coleta *feedback* dos usuários para melhoria do sistema;

8. **Monitorar:** a fase "final" do ciclo de DevOps é monitorar o ambiente através da análise do comportamento do sistema, desempenho, erros e outros.

3.2 Controle de versão de código

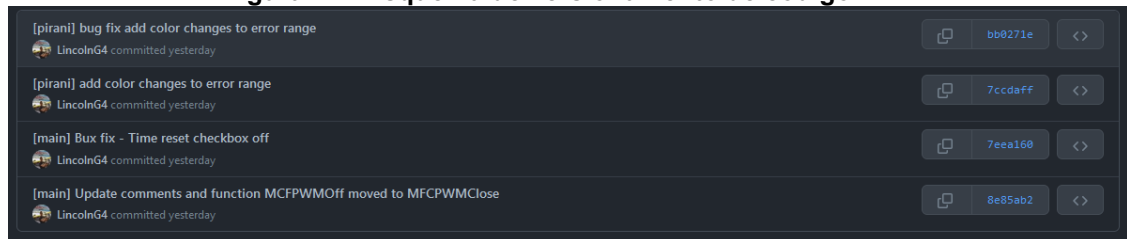
Para auxiliar a metodologia DevOps, o projeto foi armazenado e gerenciado através das ferramentas Git e Github (Figura 26), os quais permitem o controle de versão dos códigos e gerenciamento de arquivos. Esses arquivos podem ser acessados através do link: <https://github.com/LincolnG4/Flow-and-Pressure-Controller>.

Figura 26 - Controle de versão Github



O controle de versão é a prática de rastrear e gerenciar as alterações ao código-fonte ao longo do tempo (Figura 27). A ferramenta para controle de versão mantém registrado todas as modificações no código através de um banco de dados. Caso ocorra um erro, o desenvolvedor pode retornar a versão anterior (GIT, 2021). Além disso, o versionamento e armazenamento é disponibilizado em rede, auxiliando na distribuição de atualizações e correções de erros para os usuários do produto (MUDHOLKAR, 2017).

Figura 27 - Esquema de versionamento de código



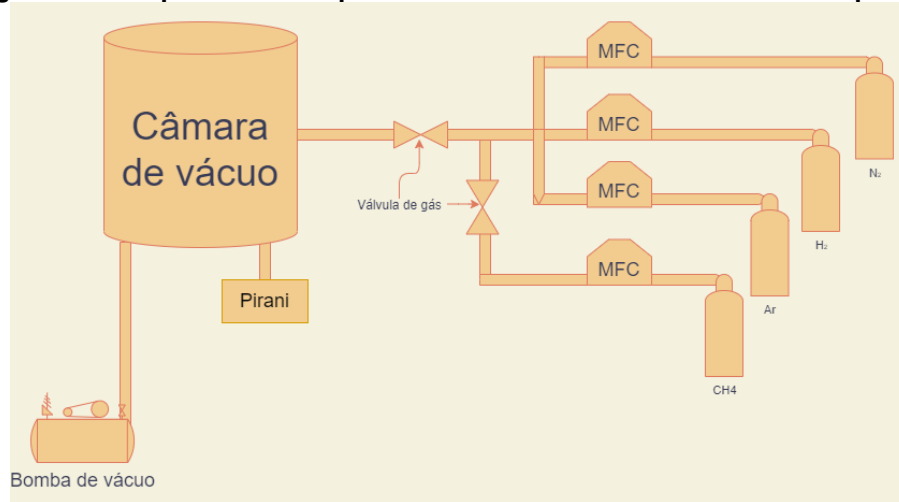
Além disso, o Github permite a realização de *fork*. O *fork* é uma cópia do repositório na qual outros desenvolvedores podem realizar mudanças sem precisar utilizar o repositório principal. Essas alterações podem ser propostas ao autor original para integrar o novo código ao projeto.

3.3 Planejamento

Nesta etapa, os requisitos do projeto foram alinhados com os professores e alunos que utilizam o reator do laboratório de tratamentos térmicos da UTFPR. Foram realizadas reuniões mensais com 4 usuários dos reatores, a fim de coletar *feedbacks* constantes e realizar testes no sistema.

Além disso, foi realizado uma visita acompanhada dos professores ao laboratório da UTFPR, a fim de entender todo o sistema de funcionamento para o equipamento. O sistema é inicialmente bombeado através de uma bomba de vácuo mecânica. O conjunto, de 4 MFC (GE50A 500 SCCM MKS Instrument), é responsável por gerenciar a passagem de cada gás que define a mistura gasosa dentro da câmara (Figura 28).

Figura 28 – Esquemático simplificado do sistema de tratamento de superfície



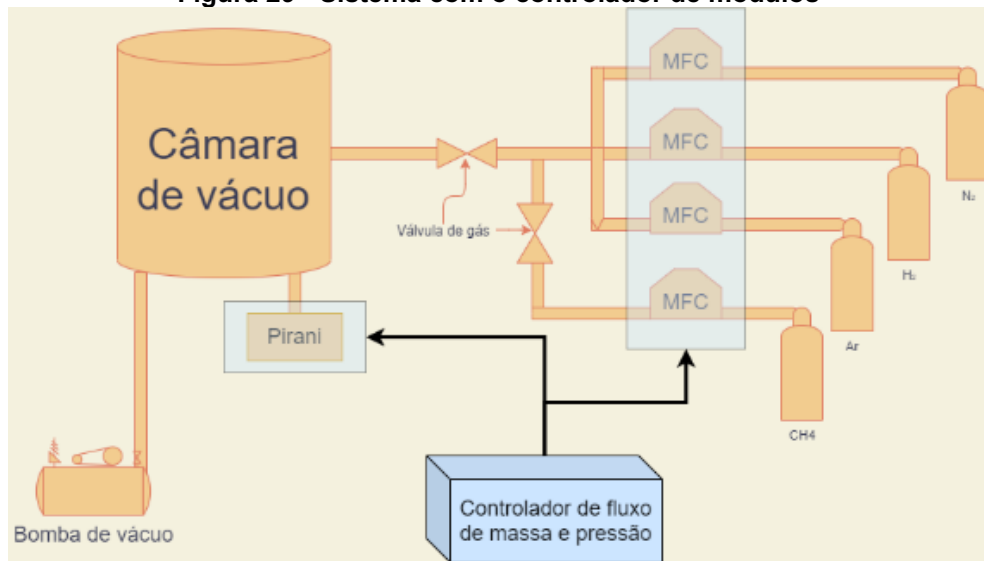
Além do desenho esquemático do sistema, foi realizada uma entrevista com os professores para entender algum dos requisitos para controlar o sistema. Logo, foi levantado, requisitos funcionais:

- Ler e controlar o fluxo de massa através de sensores MFC durante um período estipulado pelo usuário;
- Ler a pressão no reator através de um sensor Pirani;
- Gerar relatório com todas as medições realizadas;
- Calcular e corrigir as leituras de acordo com os gases utilizados;
- Alertar e prevenir erros de medição.

Os requisitos não funcionais:

- Ler e controlar o fluxo de massa através do sensor MFC GE50A 500 SCCM MKS *Instrument*;
- Entrada para 4 MFC's;
- Ler a pressão através do sensor Pirani APGX-M-NW16/ALI Edwards;
- Exibir a pressão em Torr
- O relatório deve ser gerado ao fim de cada medição.
- O relatório deve identificar sensor, data e hora de cada valor;

Portanto, o sistema final centraliza as atividades de medir a pressão através do módulo Pirani e controlar os 4 módulos de controle de fluxo de massa (Figura 29).

Figura 29 - Sistema com o controlador de módulos

4 DESENVOLVIMENTO DO PROTÓTIPO

O desenvolvimento deste projeto foi dividido em 3 grandes grupos: *hardware*, *firmware* e *software*. O *hardware* desenvolvido consiste em uma placa de circuito impresso que suporta um microcontrolador e circuitos que auxiliam a entrada e saída dos sinais elétricos. O *firmware* consiste nas rotinas programadas no microcontrolador que analisam os sinais de entrada e, a partir deles, geram os sinais de saída. O *software*, consiste em uma API desenvolvida em *JavaScript* na plataforma *Apps Script* do *Google Sheets* e o *Grafana* para criação de *dashboards*.

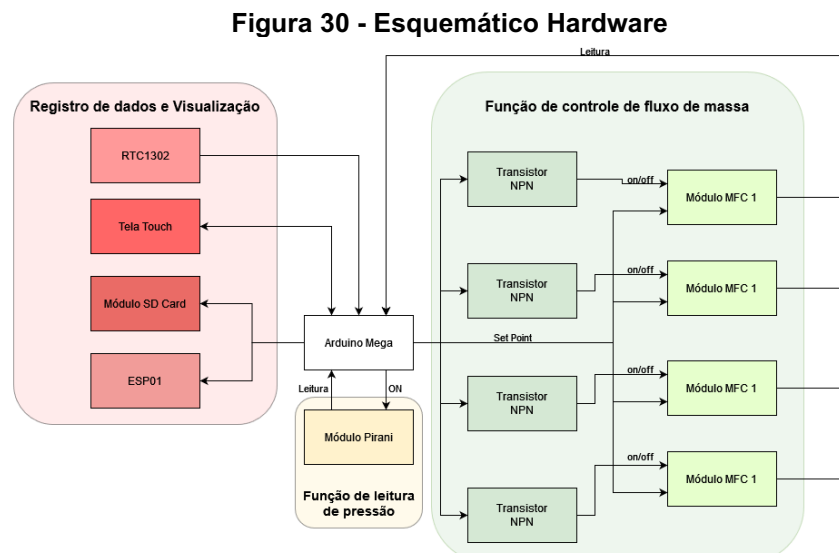
Para o *hardware*, a *PCB* e os circuitos apresentados foram desenhados e simulados no *KiCad 6.0* e *Schematic Editor*. O desenho *CAD* do *case* foi modelado no *FreeCAD 0.19* e preparação para impressão 3D no software *Cura 3D*.

O *firmware* foi escrito em *C++* utilizando o *Visual Studio Code* como editor. E o *firmware* da tela *touch screen* foi desenvolvido no software do fabricante: *Nextion Editor*. A interface gráfica do *display* foi desenhada utilizando o software *Photopea*.

Todas as ferramentas utilizadas são gratuitas e de código aberto.

4.1 Hardware

O desenvolvimento do *hardware* foi dividido em 3 categorias: “função leitura de pressão”, “função de controle de fluxo de massa” e “registro de dados e visualização”. Sendo a primeira responsável pelo tratamento do sinal do Pirani; a segunda responsável pelo tratamento do sinal do MFC; e a última responsável por coleta, *upload* e visualização dos dados gerados. Esta divisão foi esquematizada na Figura 30.



4.1.1 Microcontrolador

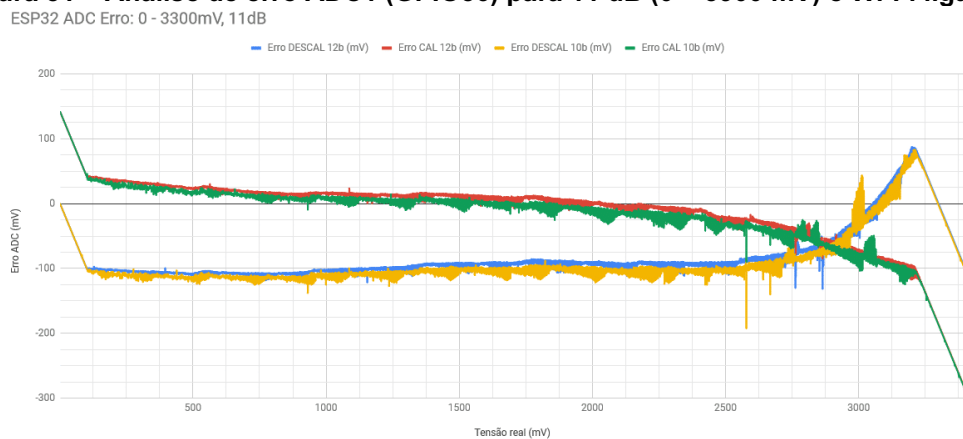
Neste trabalho, os microcontroladores Arduino Mega e ESP32 foram avaliados como possíveis soluções que atendem as necessidades do projeto. Assim, foi elaborado o Quadro 2 que visa comparar as especificações técnicas dos dois microcontroladores.

Quadro 2– Comparação Arduino Mega x ESP32

	Arduino MEGA 2560	ESP32
Alimentação	5V	3,3V
Entrada Regulada (VIN)	7 ~ 12V	5 ~ 9V
Corrente de Consumo:	Media de 70mA	Media de 80mA
Processador:	ATmega2560 RISC com até 16 MIPS	Xtensa® Dual-Core 32-bit LX6
Frequência de Operação:	0 ~ 16MHz	80MHz ~ 240MHz
Memoria FLASH:	256 KB	4MB
Memoria RAM/SRAM:	8 KB	520KB
Memória ROM/EEPROM	4 KB	448KB
Pinos de I/O:	54 pinos	34 pinos
Conversores ADC (Analogico para Digital):	16 ADC com 10-bit de resolução (1024 bits)	18 ADC com 12-bit de resolução (4096 bits)
Conversores DAC (Digital para Analógico):	-	2 DAC com 8-bit de resolução (256 bits)
WiFi:	-	2,4 GHz, 802.11 b/g/n/e/i (802.11n até 150 Mbps)
Bluetooth:	-	Bluetooth Low Energy v4.2 (BLE)
Temporizadores	4 Timers, 2 de 16-bit e 2 de 16bit	4 Timers de 64-bit
Interfaces de Módulos	I2C, SPI, 4 UART e LED PWM	SPI, SDIO, LED PWM, Motor PWM, I2S e IR

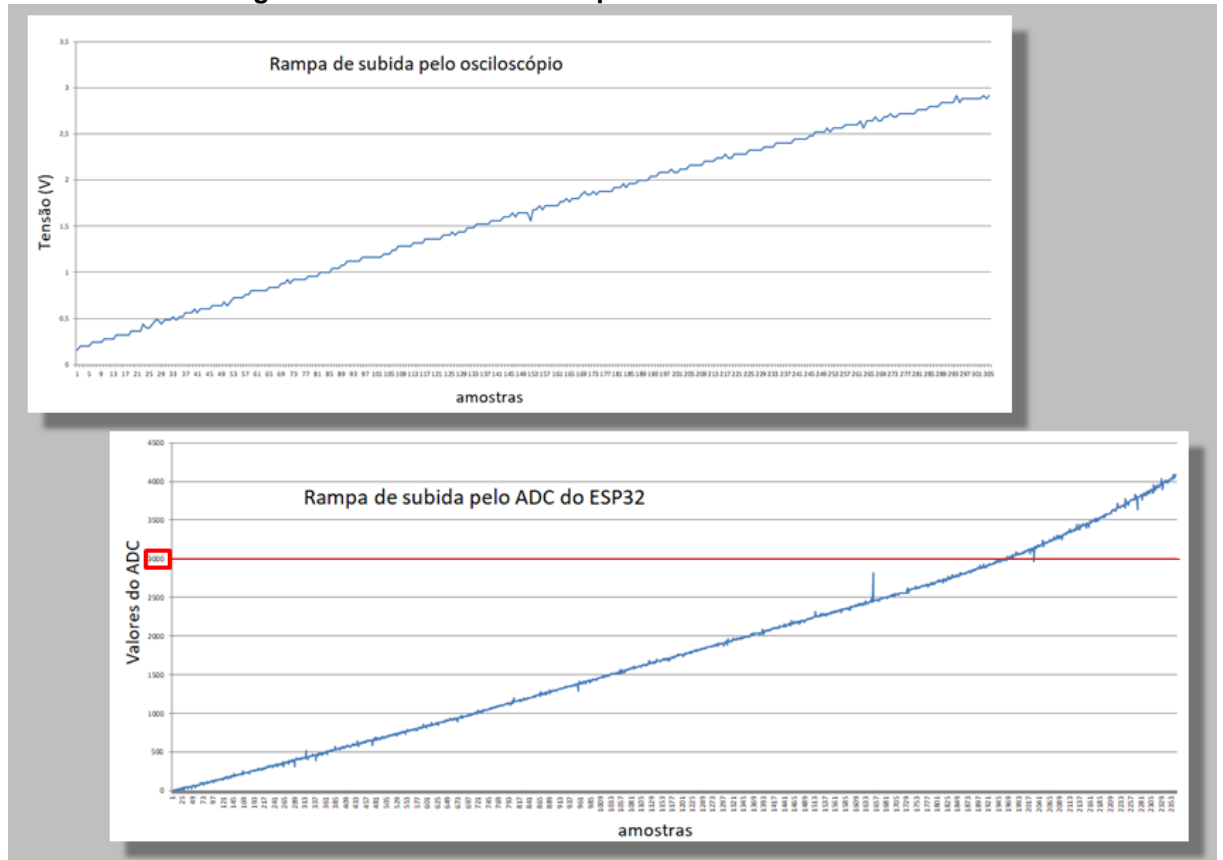
Em geral, o ESP32 possui especificações técnicas melhores de *hardware* do que o Arduino Mega. Além disso, por ele ser *dual-core*, há a possibilidade de *MultiThreading*, ou seja, paralelizar atividades em seu *software*, como por exemplo a realizar medições simultâneas dos sensores. Porém, de acordo com o manual técnico do produto (ESPRESSIF 2019) e testes realizados por outros usuários Figura 31 e 32, os resultados de medições acima de 3 V (tensão em aprox. 2.450 mV), a exatidão do ADC piora drasticamente (Figura 32). Por padrão, há $\pm 6\%$ de diferença nos resultados medidos entre os chips. Além disso, esse microcontrolador tem sinal de saída e entrada equivalente a 3 V. A redução dos sinais de entrada e a amplificação dos sinais de saída podem prejudicar a exatidão deles.

Figura 31 – Análise de erro ADC1 (GPIO36) para 11 dB (0 – 3900 mV) e Wi-Fi ligado



Fonte: Morais (2019)

Figura 32 – Leitura osciloscópio x Leitura ESP32



Fonte: Koyanagi (2018)

Assim, para usufruir de toda a capacidade do *hardware* ESP32 e das suas portas analógicas, seria necessário um longo período de testes e ajustes a fim de garantir o perfeito funcionamento. Somado a isso, este trabalho foi realizado durante as restrições da pandemia, que resultaram na redução do tempo do período letivo e em dificuldades para utilização do laboratório da universidade.

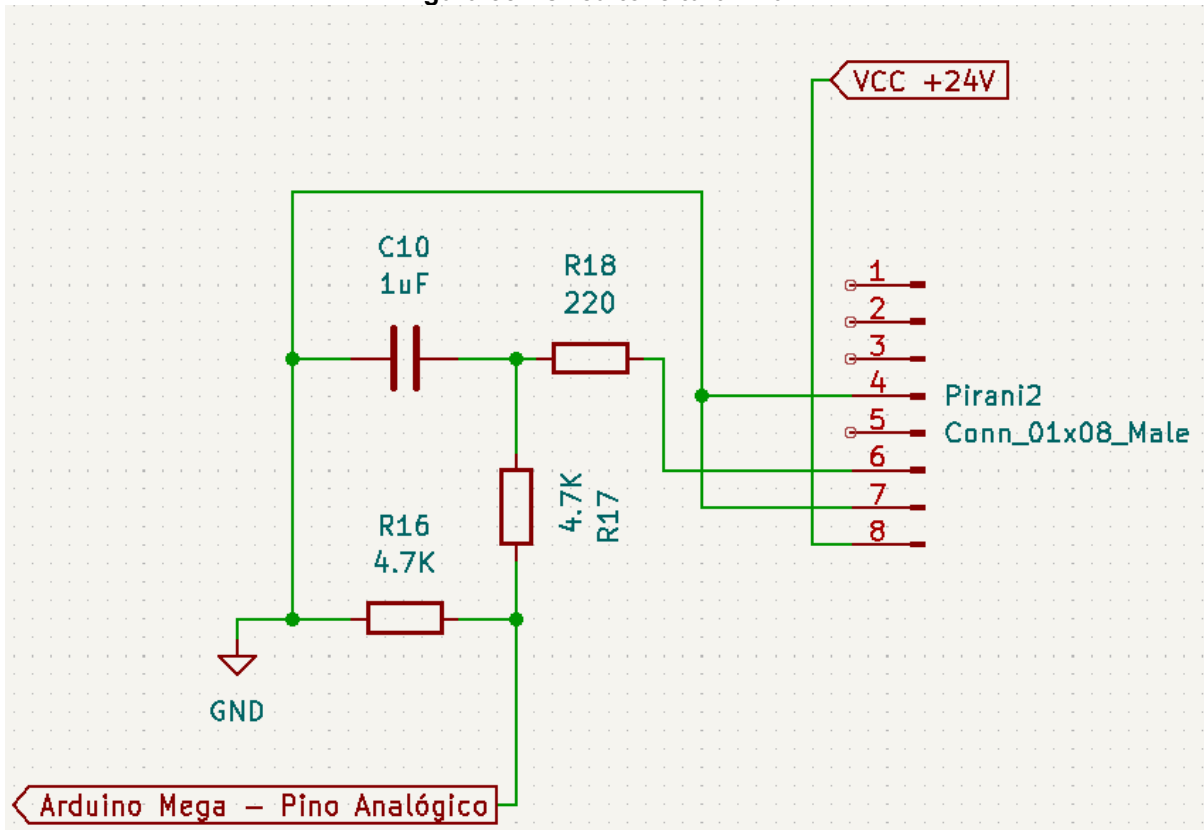
Logo, a fim de cumprir os objetivos deste trabalho dentro do período estabelecido, o Arduino Mega foi selecionado como microcontrolador. O Arduino Mega atende todos os requisitos de programação e não necessita de calibração das suas portas ADC. Junto a ele, será usado um ESP01 para gerar conexão *Wi-Fi*.

4.1.2 Circuito de leitura de pressão

O sensor Pirani Edwards APGX-M-NW16/ALI foi utilizado na leitura da pressão do reator. O sinal de saída do sensor varia entre 0 e 10 V e é submetido a um divisor de tensão, formado por dois resistores de 4.7 k Ω , e um filtro passa-baixas, um resistor de 220 Ω e capacitor de 1 μ F (Figura 33). Com isso, o sinal é reduzido pela

metade para atender a tensão máxima de 5 V do Arduino Mega. A pinagem do sensor foi representada no Quadro 3, conforme o manual Edwards (2015). A numeração da pinagem do conector Pirani na Figura 33 se difere ao Quadro 3 devido ao software inverter os pinos por conta do modelo RJ45 escolhido

Figura 33 - Circuito leitura Pirani



Quadro 3 – Pinagem RJ45 Pirani Edwards

Sinal	Pino RJ45
+15 à +36 V Alimentação	1
GND	2
Sinal de saída de medição pressão (0-10 V)	3
-	4
GND	5
-	6
-	7
-	8

4.1.2.1 Compatibilidade

O circuito foi projetado para atender sensores Pirani com alimentação mínima de 24 V e sinal de saída de 10 V. Assim, no Quadro 4, foram listados alguns dos sensores compatíveis com o *hardware*.

Quadro 4 – Sensores Pirani compatíveis

Sensor Pirani Compatíveis	Necessidade de cabo RJ45 compatível
Edwards APGX-M-NW16/ALI	Não
Edwards APG100	Não
Edwards APG200	Não
BPG402-Sx ATM to Ultra-High Vacuum Gauge	Sim
MKS EtherCAT® 901P-17020 Micro Pirani-Piezo Loadlock Vacuum	Sim
PTR 90 N PENNINGVAC 230070V02	Não
CX-PRN10 Smart Pirani Vacuum Gauge	Não

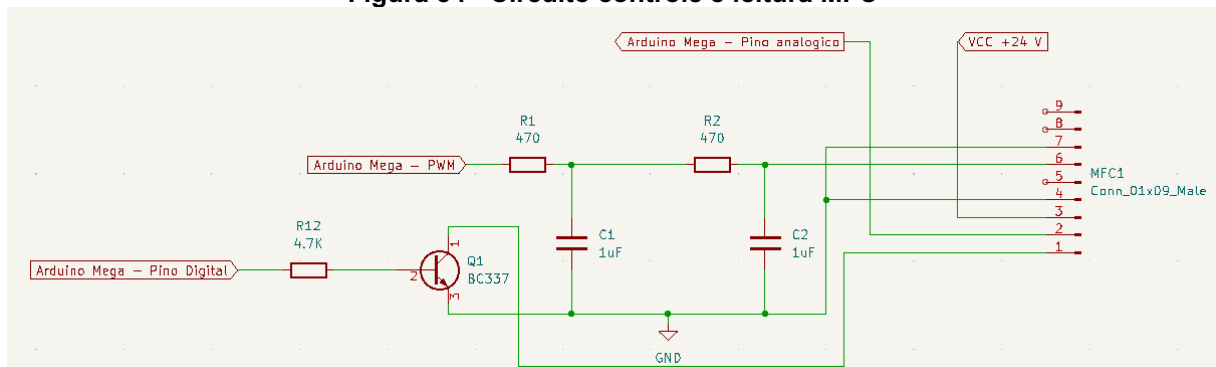
4.1.3 Circuito de leitura e controle do fluxo de massa

Foram utilizados 4 sensores *MFC GE50A 500 SCCM* da *MKS Instruments* para leitura e controle do fluxo de massa. A pinagem foi representada no Quadro 5 segundo o manual do fabricante MKS (2013).

Quadro 5 - Pinagem MFC DB9

Sinal	Pino DB9
Abertura/Fechamento válvula	1
Sinal de saída (0 - 5 V)	2
+15 à +25 VDC Alimentação	3
GND	4
Sem conexão	5
<i>Set point Input, 0-5 V</i>	6
GND	7
GND	8
Teste ponto de válvula	9

O controle da válvula (*Set point*) é submetido a um filtro pi – um resistor de 470Ω e um capacitor de 1μF cada -. Um transistor BC337 foi utilizado como chave, no pino 1, para garantir o fechamento completo da válvula (Figura 34).

Figura 34 - Circuito controle e leitura MFC

4.1.3.1 Compatibilidade

O circuito foi projetado para atender controladores de fluxo de massa com alimentação mínima de 24 V, sinal de entrada de 5 V e sinal de saída de 5 V. No Quadro 6 foram listados controladores de fluxo de massa compatíveis.

Quadro 6 – Compatibilidade de MFC

Sensor MFC Compatíveis	Necessidade de cabo DB9 compatível
G-Series GM50 - MKS	Não
MFC GF100 - Brooks Instrument	Não
GC1 Series Air Mass Flow Controller - Dakota Instruments	Sim
Whisper™ MCW-Series- Alicat	Sim
SLA5800 - Brooks Instrument	Sim
EL-FLOW® Select - Bronkhorst	Sim
MFC SLA5800 - Brooks Instrument	Não

4.1.4 Registro de dados e visualização

4.1.4.1 Tela sensível ao toque

A tela utilizada é a Nextion *serial touch* 320x240 pixels (Figura 35). Esse *display* possui um processador próprio, permitindo a renderização de imagens e funções. Assim, o Arduino prioriza as tarefas principais: leitura e processamento de sinais. A tela é composta de quatro pinos: dois pinos são para alimentação 5V e terra; dois pinos para comunicação serial (Tx e Rx).

Os pinos seriais Rx e Tx foram conectados, respectivamente, aos pinos 18 (Tx) e 19 (Rx) do Arduino Mega, equivalente ao Serial1 do microcontrolador.

Figura 35 – Display Nextion 320x240 pixels



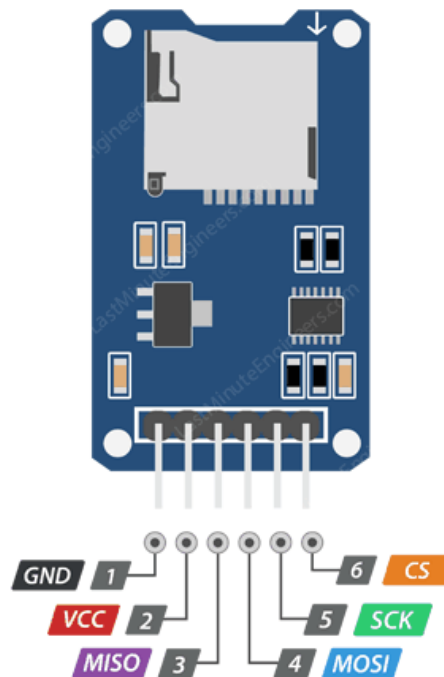
Fonte: Nextion (2022)

Outros *displays* de toque foram estudados, como o WinWin tft 2.4 que é compatível com as bibliotecas gráficas LVGL de código aberto, mas a falta de um editor de GUI e a pouca quantidade de tutoriais disponíveis foram decisivos para a escolha do Nextion como *display*.

4.1.4.2 Módulo leitura e escrita em cartão microSD

O módulo de leitura e escrita em cartão microSD, Figura 36, foi utilizado para armazenar as medições do sensor Pirani e dos controladores de fluxo de massa. Esse módulo tem capacidade de leitura de dois tipos de cartão de memória: o cartão Micro SD tradicional e o Micro SDHC -cartão de memória de alta velocidade-.

Figura 36 – Módulo leitura e escrita em cartão microSD



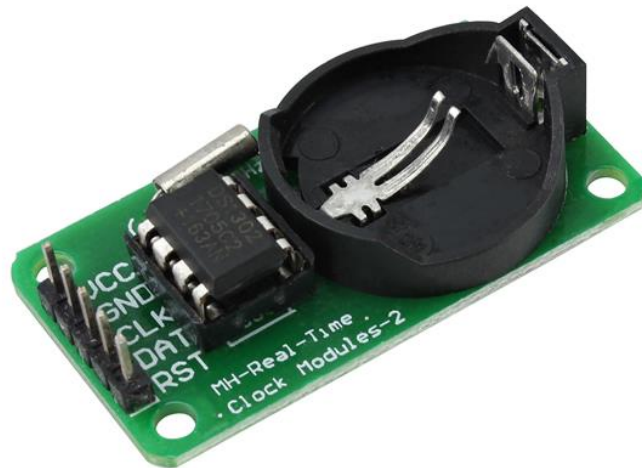
Fonte: Last Minute Engineers (2022)

Esse módulo necessita de uma comunicação SPI (*Serial Peripheral Interface*), a qual se aplica aos pinos do Arduino Mega: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (CS)

4.1.4.3 Real-time clock (RTC)

O módulo RTC ds1302 (Figura 37) foi utilizado para uma maior exatidão do controle do tempo. O módulo temporiza e registra a data e hora da coleta dos dados dos sensores que são salvos no cartão microSD.

Figura 37 - Módulo RTC ds1302



Fonte: Usina Info (2022)

O RTC necessita de uma bateria CR2032 para manter a contagem exata do tempo gravado em sua memória, mesmo quando o Arduino é desligado. A bateria tem um tempo estimado de 10 anos de vida útil.

O pino CLK é conectado ao pino 8 do Arduino e os pinos DAT e RST estão, respectivamente, conectados aos pinos 22 e 23.

4.1.4.4 ESP01 (ESP8266 Wi-Fi)

O microcontrolador ESP01 foi utilizado como módulo de conexão *Wi-Fi* para o Arduino Mega. O módulo contém 8 pinos: VCC (3.3V), GND, Rx, Tx, GPIO0, GPIO2, RST e CH_PD. Para esse projeto, foram utilizados os pinos de alimentação e o pino Rx para comunicação serial com o Arduino Mega através da porta 16 (Tx), equivalente ao Serial2 do Mega.

Através da porta serial, o Arduino envia as mensagens referente as medições dos sensores para o módulo *Wi-Fi*, que envia as mensagens para o *Google Sheets*. A função de envio para a rede será explicada com detalhes dentro do tópico de *Firmware e Software*.

4.1.4.5 Fonte de alimentação

A alimentação dos principais componentes da placa se dá com uma tensão de 24V. Assim, foi selecionado a fonte Knup (Figura 38) de 24 Volts e 4A.

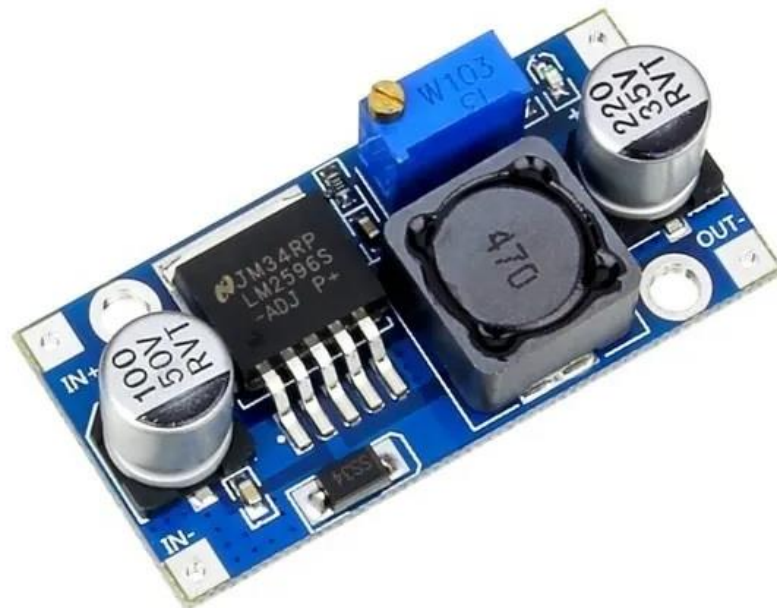
Figura 38 - Fonte Knup 24V



Fonte: Azob (2022)

O módulo regulador de tensão LM2596 (Figura 39) foi utilizado para reduzir a tensão até 7 V para alimentar o Arduino Mega com a tensão recomendada em seu *datasheet* (ARDUINO, 2022). O módulo trabalha como um conversor DC-DC no modo *Step Down*. A tensão de saída pode ser ajustada entre 1.5 e 35 V, tendo como entrada 3.2 a 40 V.

Figura 39 - Módulo regulador de tensão LM2596

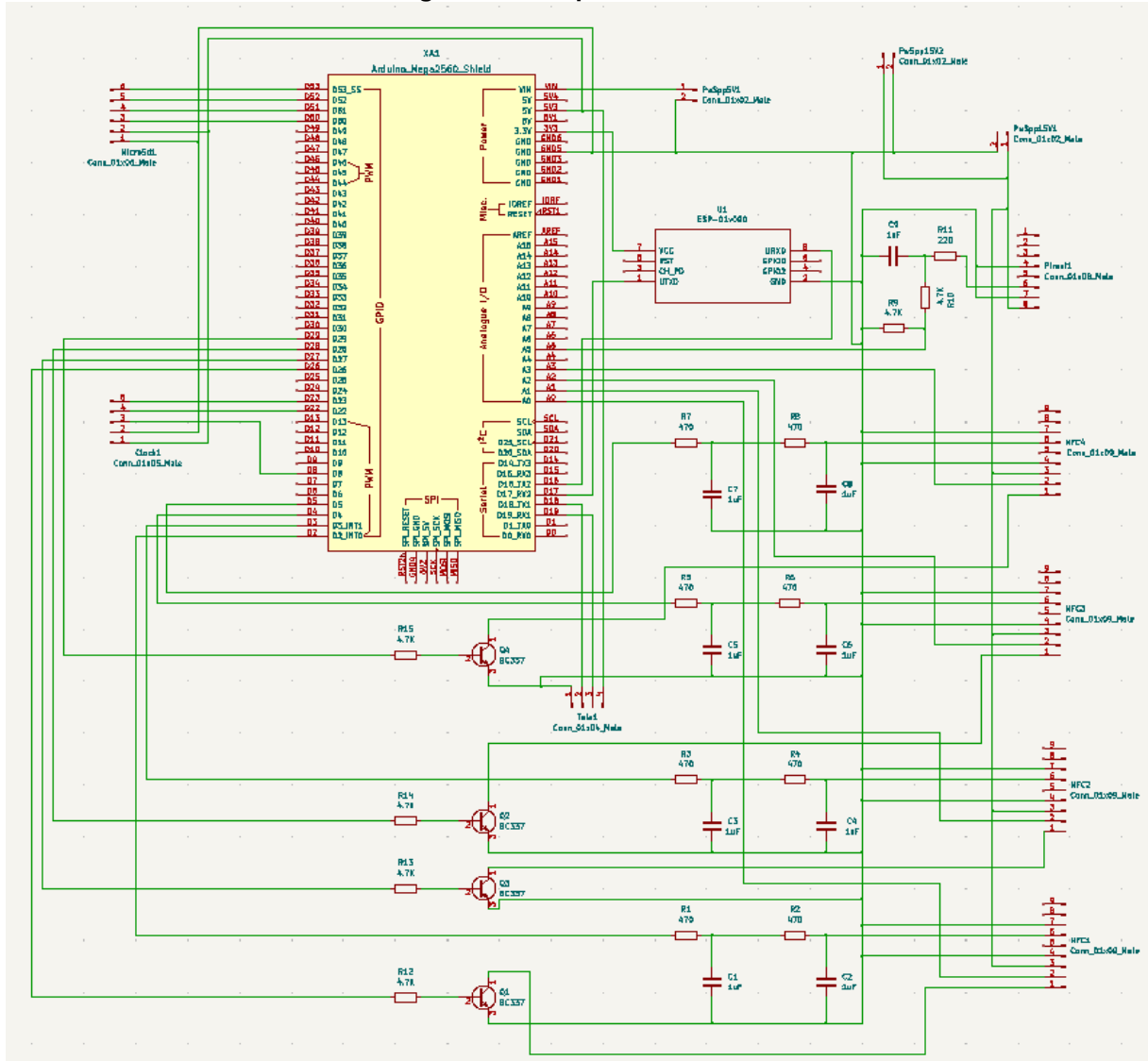


Fonte: Texas Instruments (2022)

4.1.4.6 Placa de circuito impresso

A placa de circuito impresso foi desenvolvida em formato de *shield*, que está conectada a uma placa base. O circuito eletrônico e a *PCB* foram projetadas no software KiCad 6.0 (Figura 40).

Figura 40 – Esquema eletrônico



A placa é formada por 2 camadas e dimensão 147,3 mm x 68,6 mm x 1,6 mm (Figura 41 e 42). As rotas da *PCB* foram calculadas no software Schematic Editor. Por fim, os arquivos *gerber* foram gerados para a fabricante JLCPCB (jlcpcb.com). Os arquivos *gerber* armazenam, em formato vetorial ASCII, os detalhes do projeto para configuração das máquinas -camadas de cobre, dados de perfuração, máscara de solda, legenda e outros-. A placa custou 20 reais/unidade (produto + frete).

Figura 41 – Rotas placa de circuito impresso

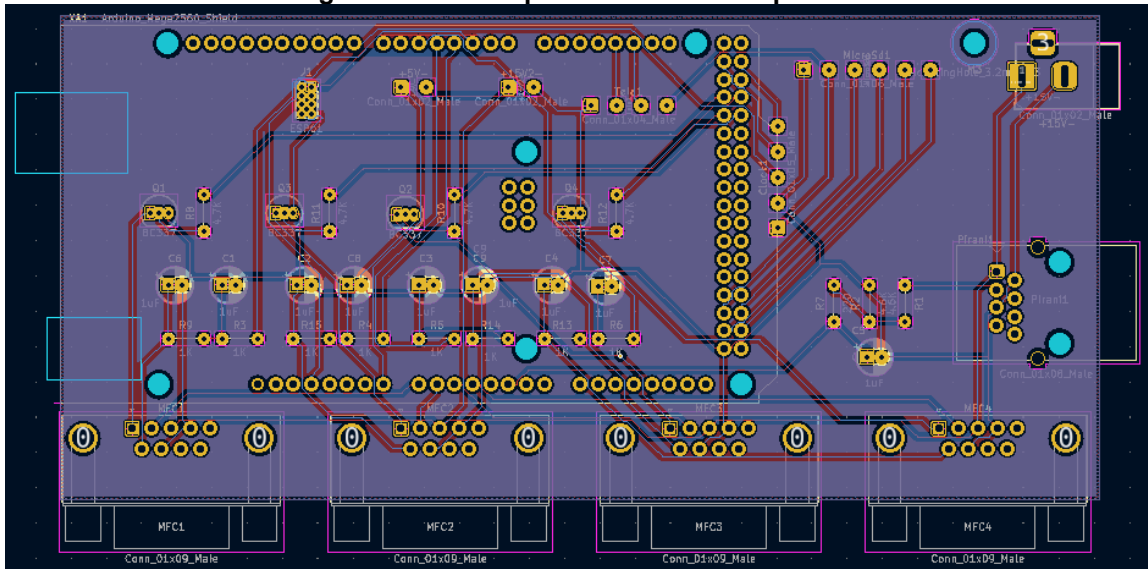
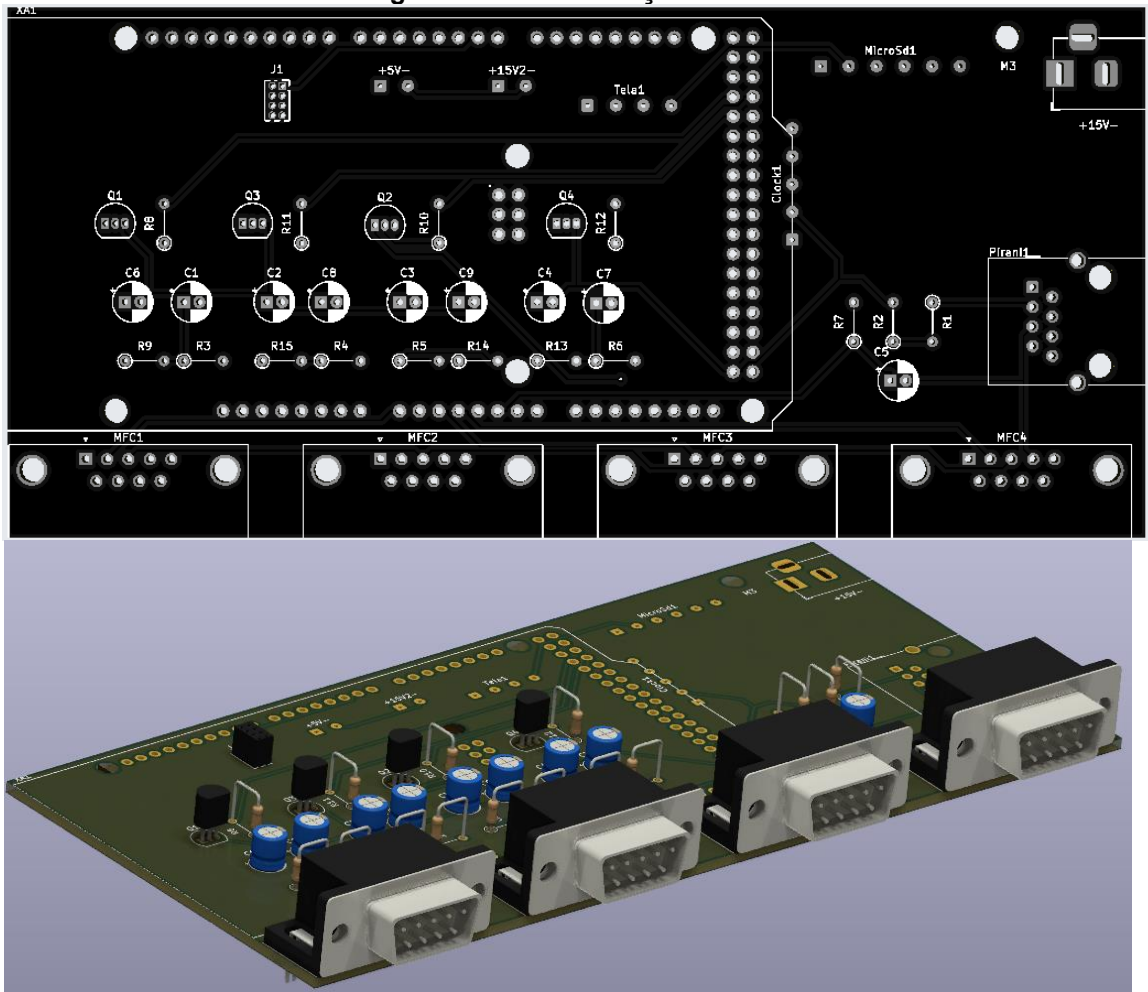


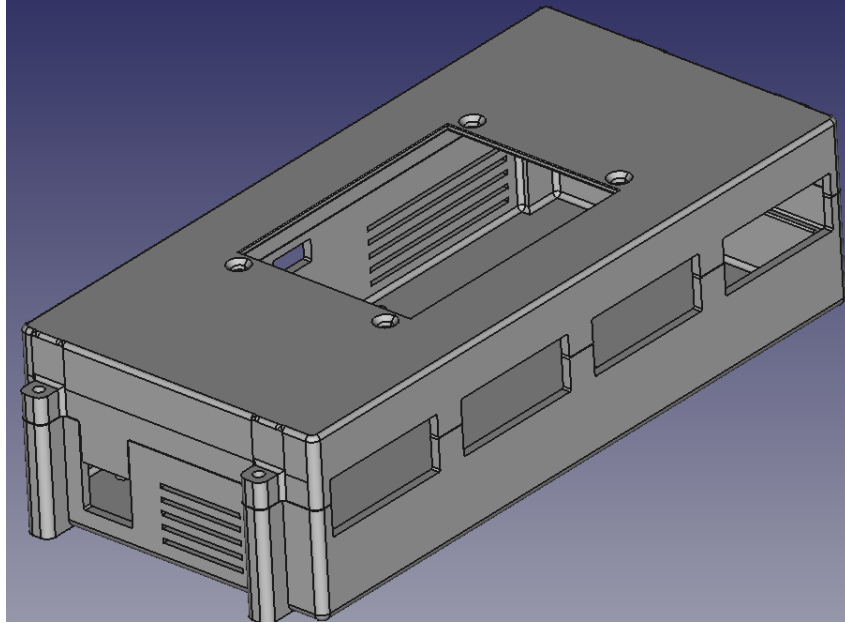
Figura 42 – Renderização da PCB



4.1.4.7 Case do produto e montagem

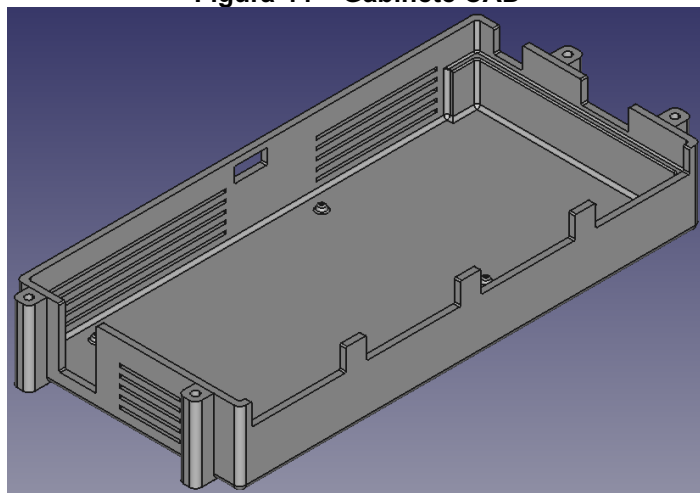
O case do protótipo é formado por um gabinete e uma tampa (Figura 43).

Figura 43 – Montagem gabinete e tampa



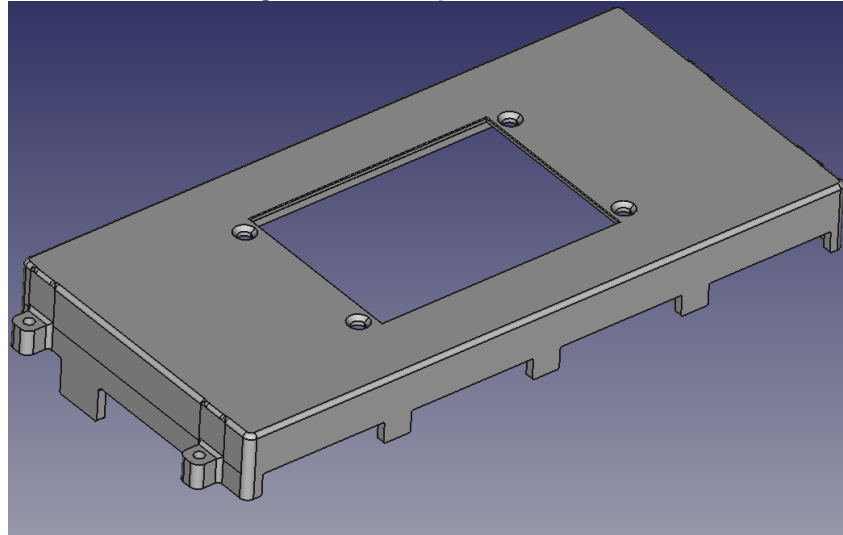
As medidas do gabinete são 165,5 x 74,6 x 25 mm (Figura 44). Ele contém frestas para ajudar no resfriamento do *hardware* já que o mesmo não possui nenhum sistema de refrigeração, como *coolers* ou ventoinhas. A montagem da *PCB shield* com o Arduino é através de encaixe e não necessita de parafusos. Além disso, existe um relevo junto a aba lateral direita para apoio da *PCB*. A tampa é parafusada ao gabinete através dos 4 suportes externos.

Figura 44 – Gabinete CAD



As medidas da tampa são 165.5 x 74.6 x 16 mm (Figura 45). Além disso, possui entrada para parafusar a tela *touch* (M3x8mm) e os apoios externos para parafusar com o gabinete (M3x10mm).

Figure 45 – Tampa CAD



O case do protótipo foi impresso 3D a partir de PLA. O PLA foi selecionado por conta do seu baixo custo, facilidade de impressão e parametrização. O *software* Cura 3D foi utilizado para a configuração da impressão (Figura 46). Foram utilizados os parâmetros: 0.2 mm de altura/camada e preenchimento de 10% (Figura 47).

Figura 46 – Posicionamento e configuração Cura 3D

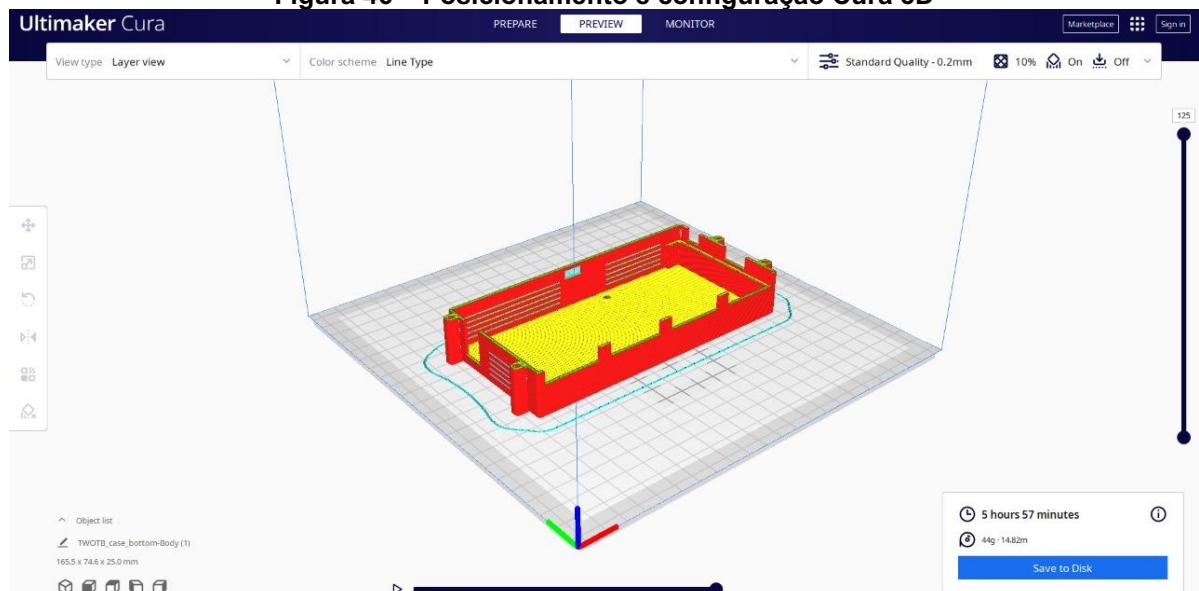
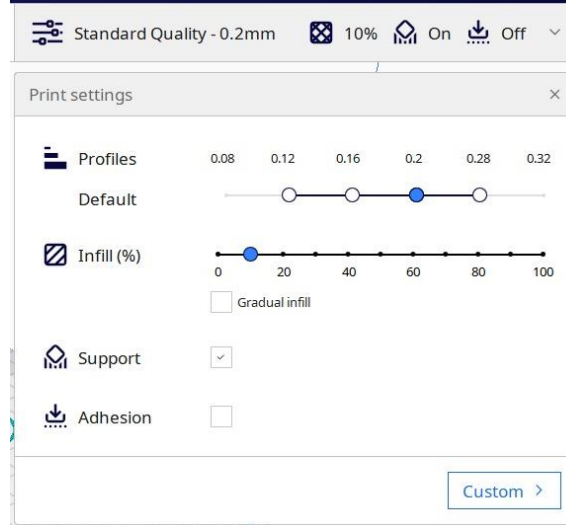


Figura 47 - Parâmetros de impressão

Assim, o resultado da montagem é apresentado na Figura 48. A parte frontal do projeto contém as entradas para os controladores de fluxo de massa e a lateral direita contém as entradas de leitura do sensor Pirani e alimentação do sistema.

Figura 48 – Montagem do protótipo

4.2 Firmware

O *firmware* é dividido em três partes: tela Nextion, Arduino Mega e ESP01. O primeiro faz o controle da interface gráfica, o segundo faz a rotina de leitura dos sensores e o último o envio das medições para a internet. O diagrama de sequência é representado na Figura 40 e o diagrama de funcionamento é representado no esquemático da Figura 50.

Figura 49 – Diagrama de sequência

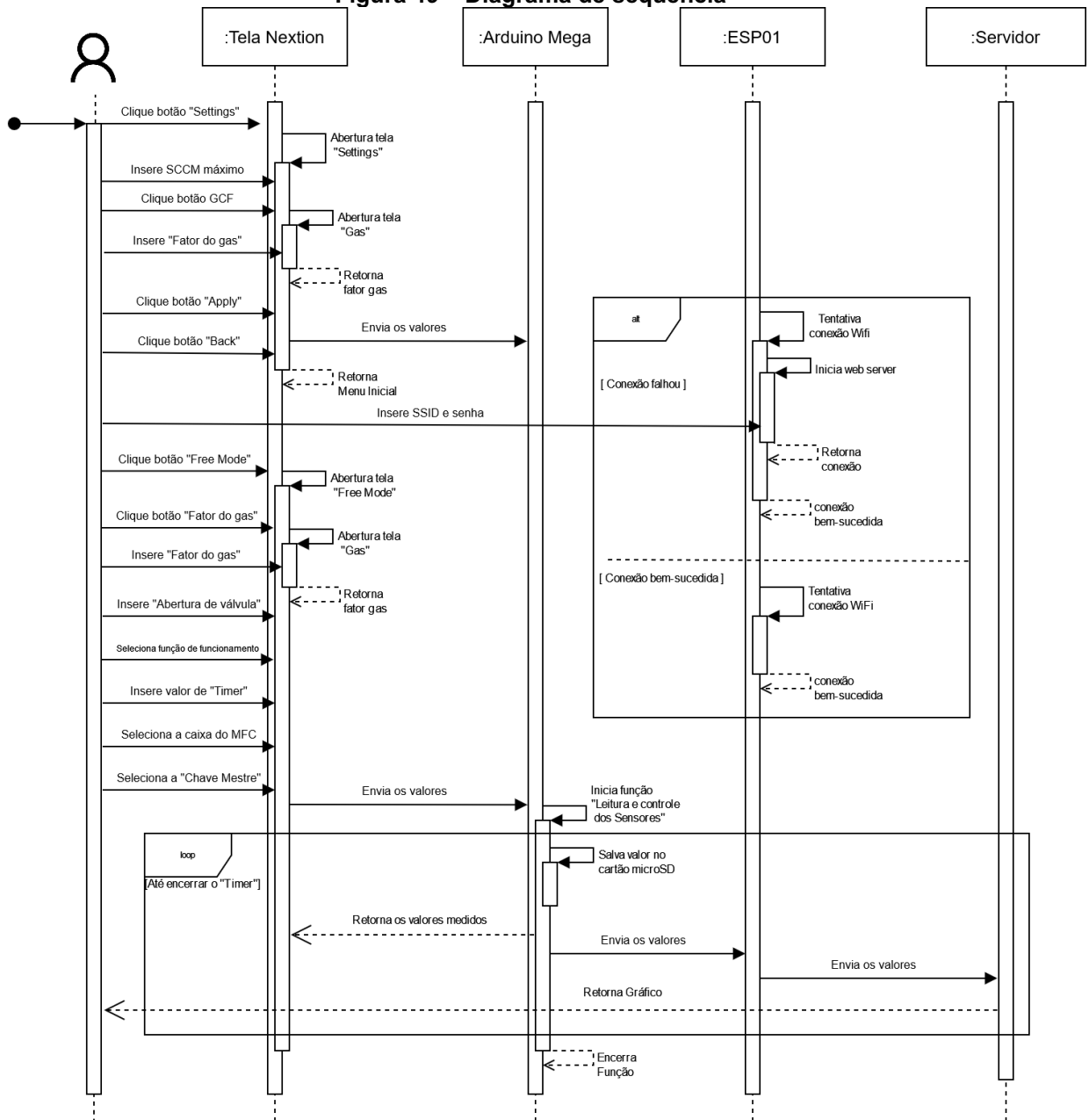
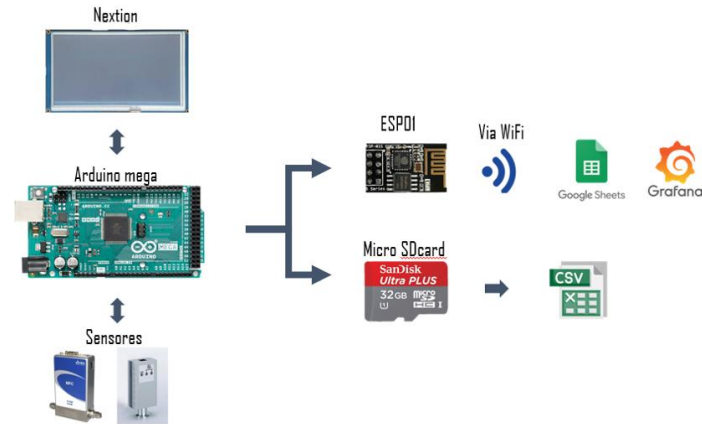


Figura 50 - Diagrama de coleta de dados

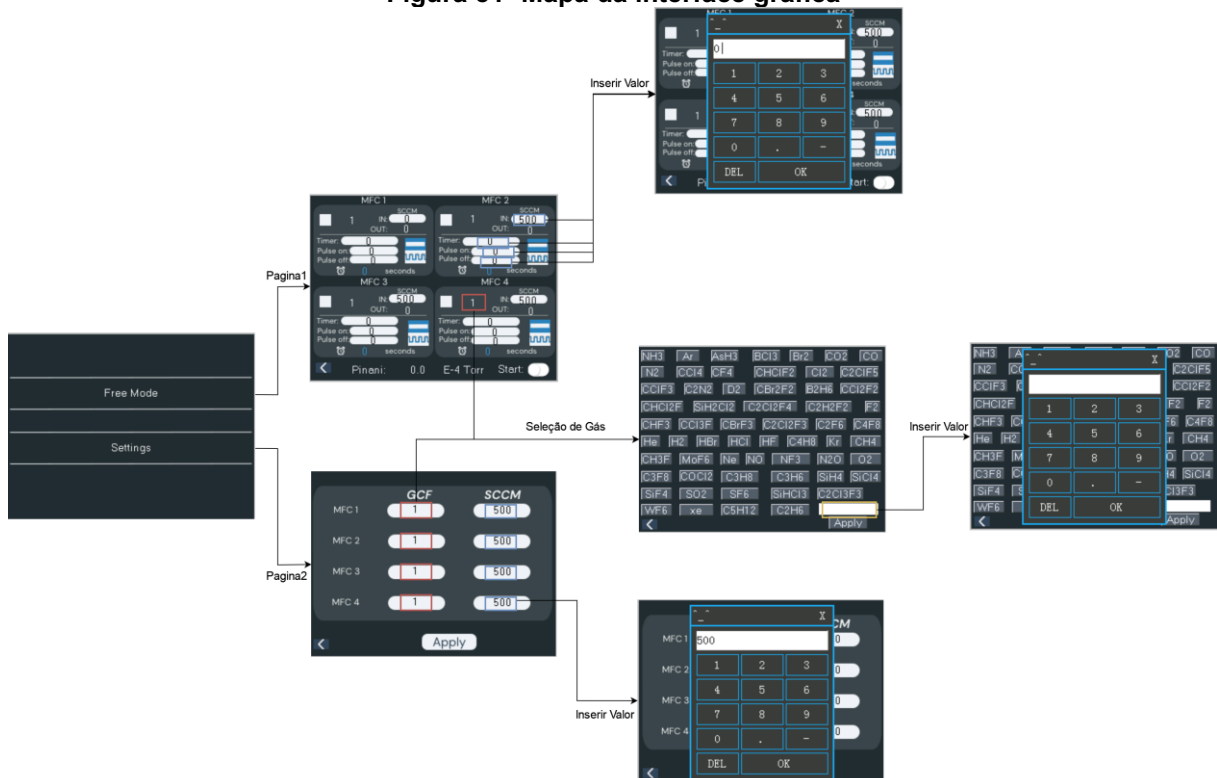


4.2.1 Nextion – Interface gráfica

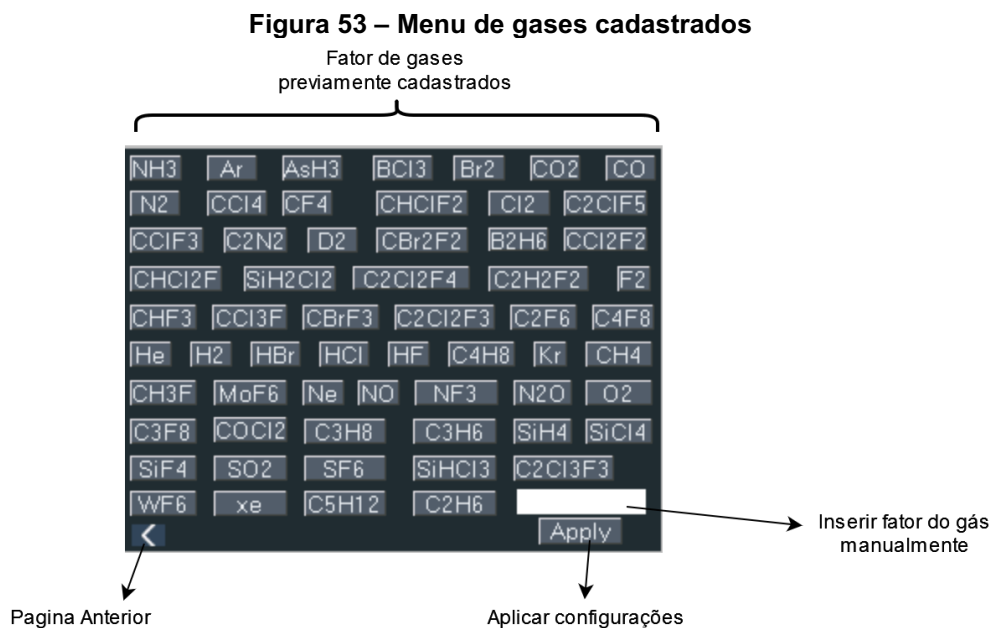
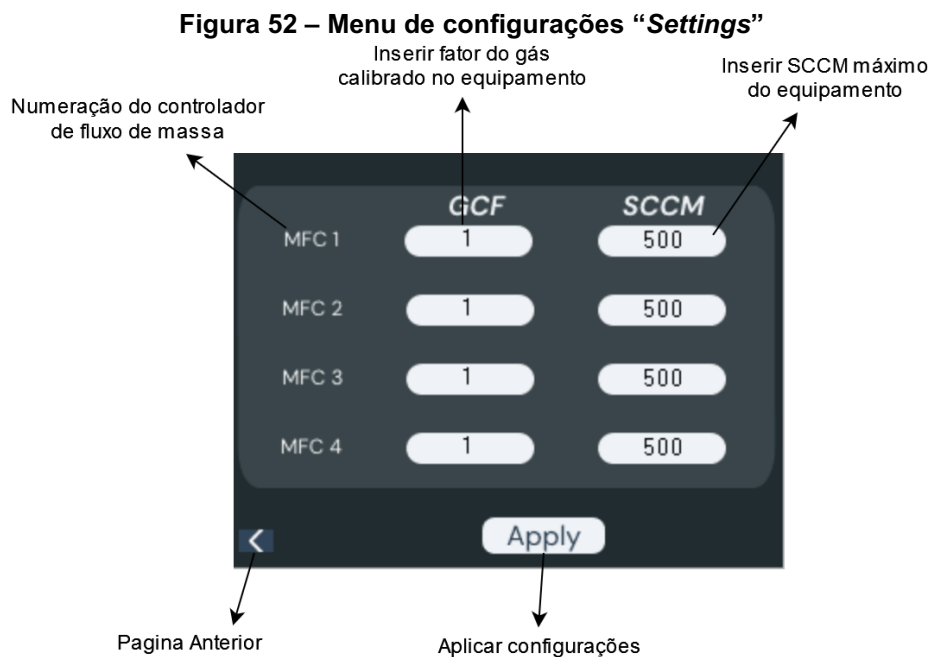
O Nextion é responsável por realizar a interface entre o usuário e o microcontrolador. A interface foi programada no *Nextion Editor*, o visual foi desenhado no *Photopea* e a paleta de cores *Flat UI Colors (2022)* foi utilizada como padrão de design.

O mapa de interação com o usuário é representado na Figura 51:

Figura 51- Mapa da interface gráfica



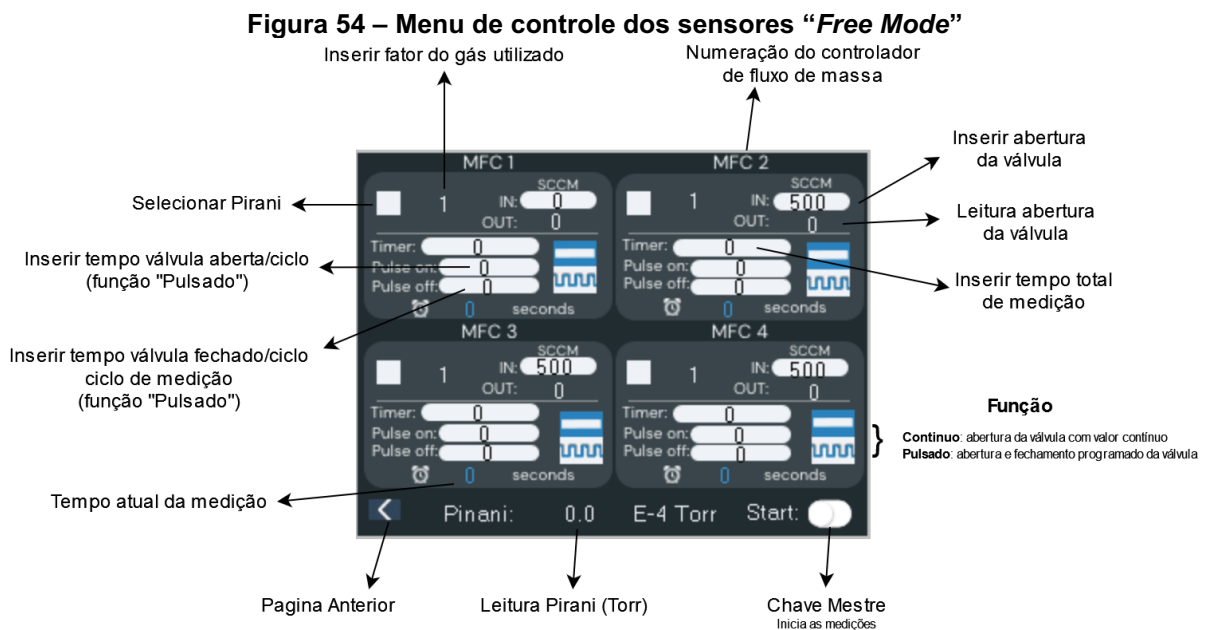
No menu de “Settings”, é possível configurar o fator de correção do gás calibrado e inserir o SCCM máximo para cada equipamento (Figura 52). Os valores dos fatores de correção dos gases cadastrados no *hardware* são selecionados (Apêndice B) com base na tabela da MKS (2022). Além disso, há a possibilidade do usuário inserir um fator manualmente (Figura 53).



A leitura dos sensores e controle dos MFC's são realizados no menu “*Free Mode*” (Figura 54). Os controladores de fluxo de massa funcionam independentes um do outro, ou seja, cada configuração de tempo, função, abertura de válvula e gás são específicas para cada um. Além disso, através das caixas de seleção, é possível selecionar quais os controladores que irão rodar durante a medição.

As funções programadas são: Contínuo e pulsado. No modo contínuo, o microcontrolador roda uma rotina de abertura constante da válvula até o fim do cronômetro inserido no *Timer* (segundos). O modo pulsado, o microcontrolador roda uma rotina de abertura e fechamento da válvula até o fim do cronômetro inserido no campo *Timer*. Os tempos dessa abertura e fechamento são inseridos nos campos “*Pulse On*” e “*Pulse Off*”.

A medição é iniciada assim que a “Chave Mestre” for ligada. Enquanto essa chave está aberta, não é possível sair do menu “*Free Mode*”.

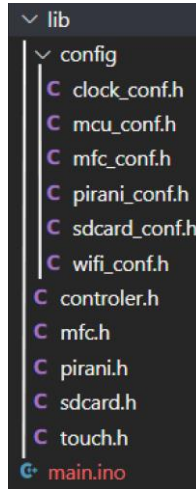


4.2.2 Arduino Mega

O Arduino Mega é responsável por armazenar as configurações dos sensores, receber os *inputs* do usuário através da tela Nextion, controlar e ler os sensores. Os códigos em C++ foram organizados em 3 partes: o código principal (“*main.ino*”), as bibliotecas de função (pasta “*lib*”), Figura 55, e os arquivos de configurações dos

equipamentos (pasta “*config*”). As principais funções e bibliotecas serão explicadas de forma geral durante este trabalho (Apêndice D).

Figura 55 - Estrutura dos arquivos



4.2.2.1 Biblioteca sensor MFC

A biblioteca do sensor MFC contém 3 funções: “*MFCPwmON*”, “*MFCPwmClose*” e “*MFCRead*”. A primeira função recebe o *input* de abertura da válvula do *MFC*, converte em um sinal *PWM* para abertura da válvula -considerando o valor de *SCCM* máximo configurado pelo usuário- e atualiza o valor na tela (Código 1).

Código 1 - Função “*MFCPwmON*”

```

/*
 * Send PWM signal to open valve
 * @param PinPWM timer timerText SCCMValveOpen CalibratedSCCM
 */
void MfcPwmON(int PWM_INPUT,int delta,NexText t,NexNumber n,uint32_t
mfcSCCM) {
    char buffer[100] = {0};
    uint32_t val;

    n.getValue(&val); //Request object from Nextion

    val = map(val, 0, mfcSCCM, 0, 255);
    analogWrite(PWM_INPUT, val);

    itoa(delta,buffer, 10);
    t.setText(buffer); //Send value to object Nextion
}

```

A função “*MFCPwmClose*” envia sinal para acionar o transistor, fechando a válvula, e o sinal *PWM* é colocado em nível lógico baixo (Código 2).

Código 2 - Função "*MFCPwmClose*"

```

/*
 * This sends PWM zero to close valve
 * @param t_calibrate MFC SCCM value
 * @return new float MFC SCCM calibration
 */
void MfcPwmClose(int PWM_INPUT,int delta,NexText t,int Close_MFC){
    char bufferclose[100] = {0};

    digitalWrite(Close_MFC, HIGH);
    analogWrite(PWM_INPUT, 0);

    itoa(delta,bufferclose, 10);
    t.setText(bufferclose);
}

```

A última função, “*MFCRead*”, lê o retorno de abertura de válvula do controlador de fluxo de massa e corrige a saída SCCM de acordo com o fator de correção do gás e da calibração do equipamento utilizado (Código 3).

Código 3 - Função "*MFCRead*"

```

/*
 * This read MFC and applies gas correction factor
 * @param MFC_number T_print textMFCOut GasFactor mfcSCCM
 * @return MeasureMFC
 */
float MFCRead(int MFC,NexText t,NexText t_out, float
Calibrated_MFC,uint32_t mfcSCCM){
    char mfcOutputString[10];
    float gas_factor_in;
    float output_MFC_factor;
    char bufferText[10];

    MFC_V =
(mfcSCCM*(sinalMKS/SinalMCU)*((SinalMCU*(analogRead(MFC)))/bitsMCU))/SinalM
CU;

    memset(bufferText, 10, sizeof(bufferText));
    t_out.getText(bufferText, sizeof(bufferText));

    gas_factor_in = atof(bufferText);
    output_MFC_factor = float(MFC_V)*(gas_factor_in/Calibrated_MFC);

    dtostrf(output_MFC_factor, 1, 2, mfcOutputString);
}

```

```

t.setText(mfcOutputString);

return output_MFC_factor;
}

```

4.2.2.2 Biblioteca sensor Pirani

A biblioteca do sensor Pirani contém 2 funções: “*PiraniRead*”, “*piraniUpdateScreen*”. A primeira função lê a medição do sensor e converte de Volts para Torr. Porém, caso haja controladores de fluxo de massa rodando em paralelo, a função corrige a medição aplicando o fator de correção dos gases, considerando a porcentagem de fluxo de entrada para cada controlador (Código 4).

Código 4 - Função "PiraniRead"

```

/*
 * Read Pirani
 * This get signal from Pirani, converts from V to Torr and applies a
 * correction factor considering the gas %.
 * If no MFC is turned ON, it won't apply a correction factor
 * @params Checkbox1_4, GasFactor1_4, MFC_valve
 * @return Pirani in torr
 */
float PiraniRead(uint32_t checkbox1,uint32_t checkbox2,uint32_t
checkbox3,uint32_t checkbox4,NexText t_gas1,NexText t_gas2,NexText
t_gas3,NexText t_gas4,int mcfOutput1,int mcfOutput2,int mcfOutput3,int
mcfOutput4){
    float Pirani_V = 0;
    float totalFlow=0;
    float correctionFactor=1;
    char bufferText[10];
    char bufferText2[10];
    char bufferText3[10];
    char bufferText4[10];
    float gas_factor_1 = 1;
    float gas_factor_2 = 1;
    float gas_factor_3 = 1;
    float gas_factor_4 = 1;

    Pirani_V =
(sinalPirani/SinalMCU) * ((SinalMCU * (analogRead(Read_Pirani))) / bitsMCU);

    if(checkbox1==0 && checkbox2==0 && checkbox3==0 && checkbox4==0){ //if
mfc's are turned off
        Pirani_Torr = 10000.00 * (pow(10, (Pirani_V - 6.125)));
        return Pirani_Torr;
    }
}

```

```

else{ //mfc's are turned off, calculate % gas and apply correction factor
to Pirani measure
    if(checkbox1==1){

        totalFlow = totalFlow + mcfOutput1;

        memset(bufferText, 10, sizeof(bufferText));
        t8.getText(bufferText, sizeof(bufferText));
        gas_factor_1 = atof(bufferText);
    }
    if(checkbox2==1){

        totalFlow = totalFlow + mcfOutput2;

        memset(bufferText2, 10, sizeof(bufferText2));
        t19.getText(bufferText2, sizeof(bufferText2));
        gas_factor_2 = atof(bufferText2);
    }
    if(checkbox3==1){

        totalFlow = totalFlow + mcfOutput3;

        memset(bufferText3, 10, sizeof(bufferText3));
        t20.getText(bufferText3, sizeof(bufferText3));
        gas_factor_3 = atof(bufferText3);
    }
    if(checkbox4==1){

        totalFlow = totalFlow + mcfOutput4;

        memset(bufferText4, 10, sizeof(bufferText4));
        t21.getText(bufferText4, sizeof(bufferText4));
        gas_factor_4 = atof(bufferText4);
    }

    correctionFactor = checkbox1*(mcfOutput1/totalFlow)*gas_factor_1 +
checkbox2*(mcfOutput2/totalFlow)*gas_factor_2 +
checkbox3*(mcfOutput3/totalFlow)*gas_factor_3 +
checkbox4*(mcfOutput4/totalFlow)*gas_factor_4;
    Pirani_Torr = 10000.00*(pow(10, ((Pirani_V*correctionFactor)-
6.125))); // convert V to Torr and applies correction Factor
    return Pirani_Torr;
}
}

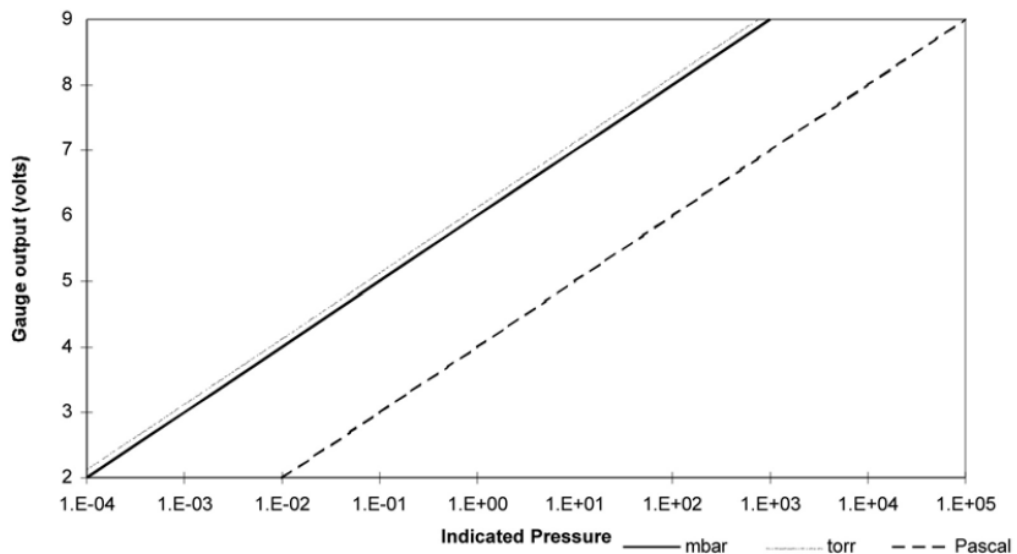
```

A última função atualiza a tela Nextion com o valor de Pirani calculado na função anterior (Código 5). Porém, conforme o manual do fabricante Edwards (2007), as medições abaixo de 2 V e acima de 9 V são faixas de erro -Figura 56-. Os erros mapeados são apresentados no Quadro 7. Assim, a função alerta o usuário do erro colorindo de vermelho o valor de medição na tela.

Código 5 - Função "piraniUpdateScreen"

```
void piraniUpdateScreen(float Pirani_V){
  dtostrf(Pirani_V, 4, 2, buff);
  t10.setText(buff); // Print on screen the Pirani value
  if(Pirani_V<0.7498 || Pirani_V>7498942.0933){
    t10.Set_font_color_pco(57959); //Set Object to change color white
  }
  else{
    t10.Set_font_color_pco(61342); //Set Object to change color
  }
}
```

Figura 56 - Transformação de tensão para pressão



Fonte: Edwards (2007)

Quadro 7 - Mapeamento de erros

Indicação de erro	Sinal de saída
Filamento quebrado	9,5 V
Erro de calibração	9,6 V

4.2.2.3 Modo de operação

Os controladores de fluxo de massa têm 2 modos de operação: contínuo e pulsado. O modo de operação contínuo mantém a válvula aberta com valor constante

até o fim do cronômetro configurado pelo usuário (Código 6). Além disso, durante a operação, as medições são salvas no arquivo “CSV”.

Código 6 - Modo contínuo

```

/*
 * MFC - Linear
 * This 'If' starts the Linear function that controls the valve with
 constant value for a period of time.
 * During the ON period, it saves all measure into a CSV file in microSDcard
 */

if(MFC1_mode_linear==1 && MFC1_mode_pulse==0){
  MFC_CSV_1 = SD.open("MFC1.csv", FILE_WRITE); //Create a CSV file

  if(timer_enable == 0){ //first time run
    n1.getValue(&timerMFC1); // Get time to be Opening&Running
    start_mfc1 = unix; // Get time start reference
    timer_enable = 1;
  }

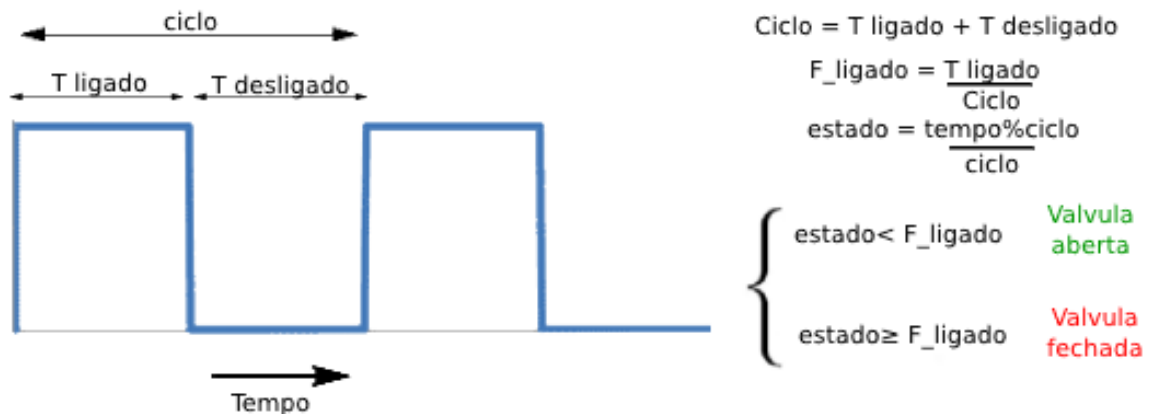
  delta = unix - start_mfc1;

  if(timerMFC1 >= delta){
    digitalWrite(Close_MFC1, LOW); // Open valve
    MfcPwmON(PWM_INPUT1,delta,t15,n0,mfcSCCM1);
    MFC1_value = MFCRead(MFC,t11,t8,Calibrated_MFC_1,mfcSCCM1);
    String message1 = TimeString + "," + String(MFC1_value);
    WriteSD(MFC_CSV_1,message1);
  }
  else{
    MfcPwmClose(PWM_INPUT1,delta_pulse,t15,Close_MFC1);
    c0.setValue(0); //uncheck checkbox object Nextion
    checkbox1 = 0;
    timer_enable = 0;
    int delta = 0;
  }
}

```

O modo de operação pulsado (Código 7) varia entre a válvula aberta e fechada até o fim do cronômetro. Ou seja, a medição se encerra ao fim do cronometro independentemente do estado da válvula. O cálculo dessa variação de estado é explicado na Figura 57.

Figura 57 - Esquema de abertura e fechamento



Código 7 - Modo pulsado

```

/*
 * MFC - Pulse
 * This 'If' starts the Pulse function that controls the valve switching
 from ON and OFF.
 * During the ON/OFF period, it saves all measure into a CSV file in micro
 SDcard
 */
if(MFC1_mode_pulse==1 && MFC1_mode_linear==0){

    //CREATE SAVE A SD CARD
    MFC_CSV_1 = SD.open("MFC1.csv", FILE_WRITE);

    if(timer_enable_pulse == 0){
        n1.getValue(&timerMFC1_pulse); // Get total timer
        n8.getValue(&timerMFC1_pulse_on); // Get timer to open pulse
        n9.getValue(&timerMFC1_pulse_off); // Get timer to close pulse
        totall = timerMFC1_pulse_on + timerMFC1_pulse_off;
        cycle_on = timerMFC1_pulse_on/float(totall);
        start_mfc1_pulse = unix;
        timer_enable_pulse = 1;
        //Serial.println("[INFO] - MFC1 - Turned ON");
    }

    delta_pulse = unix - start_mfc1_pulse;

    if(timerMFC1_pulse > delta_pulse){
        if( ((delta_pulse%totall)/float(totall)) < cycle_on ){ // open pulse
            digitalWrite(Close_MFC1, LOW); //OPEN VALVE
            MfcPwmON(PWM_INPUT1,delta_pulse,t15,n0,mfcSCCM1);
            MFC1_value = MFCRead(MFC,t11,t8,Calibrated_MFC_1,mfcSCCM1);
            String message1 = TimeString + "," + String(MFC1_value); //
            WriteSD(MFC_CSV_1,message1);
        }
    }
}

```

```

else{ // close pulse
  MfcPwmClose(PWM_INPUT1,delta_pulse,t15,Close_MFC1);
  MFC1_value = MFCRead(MFC,t11,t8,Calibrated_MFC_1,mfcSCCM1);
  String message1 = TimeString + "," + String(MFC1_value);
  WriteSD(MFC_CSV_1,message1);
}

}

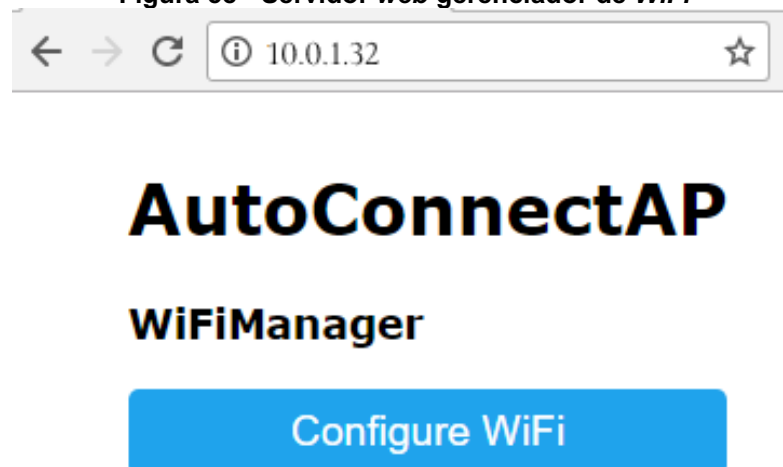
else {
  MfcPwmClose(PWM_INPUT1,delta_pulse,t15,Close_MFC1);
  c0.setValue(0); //Send to object to close Nextion Checkbox
  checkbox1 = 0;
  delta_pulse= 0;
  timer_enable_pulse = 0;
}
}
}

```

4.2.3 ESP01 – conexão *Wi-Fi*

O ESP01 é responsável por coletar as medições do Arduino Mega e enviar para o *Google Sheets* através da API desenvolvida na plataforma *App Script* do *Google* (Código 8). Além disso, esse microcontrolador é responsável por gerenciar a conexão *Wi-Fi*. Ao ligar, caso a conexão *Wi-Fi* falhe, o ESP01 inicia um servidor *web* para que o usuário troque a conexão e senha (Figura 58). Esse servidor tem IP fixo (10.0.1.32) e o nome da conexão é "UTFPR_IOT_10.0.1.32" e a senha foi disponibilizada para os usuários.

Figura 58 - Servidor web gerenciador de *WiFi*



Código 8 - Função enviar para a API

```

/*
 * Send data to API
 * This process message and sends a 'GET' request to Google Sheets API
 * @params Measure Pirani, MFC 1 , MFC 2, MFC 3, MFC 4
 */
void sendData(float Pirani_ESP, float MFC1_ESP, float MFC2_ESP, float
MFC3_ESP, float MFC4_ESP) {

    //Processing data and sending data
    String string_Pirani_ESP = String(Pirani_ESP);
    String string_MFC1_ESP = String(MFC1_ESP);
    String string_MFC2_ESP = String(MFC2_ESP);
    String string_MFC3_ESP = String(MFC3_ESP);
    String string_MFC4_ESP = String(MFC4_ESP);

    // prepare header to API
    String url = "/macros/s/" + GAS_ID + "/exec?Pirani=" + string_Pirani_ESP
+ "&MFC1=" + string_MFC1_ESP+ "&MFC2=" + string_MFC2_ESP + "&MFC3=" +
string_MFC3_ESP + "&MFC4=" + string_MFC4_ESP;

    //API GET
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
        "Host: " + host + "\r\n" +
        "User-Agent: BuildFailureDetectorESP8266\r\n" +
        "Connection: close\r\n\r\n");
}

```

4.3 Software

O *software* é responsável pela telemetria do equipamento. O banco de dados *NoSQL Firebase* e o armazenamento no *Google Sheets* foram avaliados. O *Firebase* é o mais indicado devido à velocidade de processamento e disponibilidade. Porém, como a ideia do projeto é ser um protótipo funcional e minimizar custos, foi utilizado o *Google Sheets* como repositório devido a sua gratuidade e a possibilidade de enviar um maior número de mensagens do que a versão gratuita do *Firebase*.

Assim, através de uma planilha no *Google Sheets* (Figura 59), é possível gerar uma *API* no *Google Apps Script* (Figura 60) para processar as mensagens que são enviadas através de um *url*. O código 9 de processamento é escrito na linguagem *JavaScript*.

Figura 59 - Painilha recebendo dados do ESP01

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Pirani		Date	MFC1		Date	MFC2		Date	MFC3		Date
2	2022-05-20 9 59 02	24022		2022-05-20 9 59 02	0		2022-05-20 9 59 02	500		2022-05-20 9 59 02	321		2022-05-20 9 59 02
3	2022-05-20 9 59 08	12843		2022-05-20 9 59 08	0		2022-05-20 9 59 08	500		2022-05-20 9 59 08	321		2022-05-20 9 59 08
4	2022-05-20 9 59 24	34661		2022-05-20 9 59 24	96		2022-05-20 9 59 24	500		2022-05-20 9 59 24	321		2022-05-20 9 59 24
5	2022-05-20 9 59 30	11935		2022-05-20 9 59 30	95		2022-05-20 9 59 30	500		2022-05-20 9 59 30	321		2022-05-20 9 59 30
6	2022-05-20 9 59 36	30428		2022-05-20 9 59 36	94		2022-05-20 9 59 36	500		2022-05-20 9 59 36	321		2022-05-20 9 59 36
7	2022-05-20 9 59 41	16154		2022-05-20 9 59 41	94		2022-05-20 9 59 41	500		2022-05-20 9 59 41	321		2022-05-20 9 59 41
8	2022-05-20 9 59 47	8174		2022-05-20 9 59 47	94		2022-05-20 9 59 47	500		2022-05-20 9 59 47	321		2022-05-20 9 59 47
9	2022-05-20 9 59 54	21398		2022-05-20 9 59 54	92		2022-05-20 9 59 54	500		2022-05-20 9 59 54	321		2022-05-20 9 59 54
10	2022-05-20 10 00 00	9063		2022-05-20 10 00 00	91		2022-05-20 10 00 00	500		2022-05-20 10 00 00	321		2022-05-20 10 00 00
11	2022-05-20 10 00 05	5410		2022-05-20 10 00 05	91		2022-05-20 10 00 05	500		2022-05-20 10 00 05	321		2022-05-20 10 00 05
12	2022-05-20 10 00 11	23539		2022-05-20 10 00 11	90		2022-05-20 10 00 11	500		2022-05-20 10 00 11	321		2022-05-20 10 00 11
13	2022-05-20 10 00 17	3706		2022-05-20 10 00 17	90		2022-05-20 10 00 17	500		2022-05-20 10 00 17	321		2022-05-20 10 00 17
14	2022-05-20 10 00 24	6269		2022-05-20 10 00 24	91		2022-05-20 10 00 24	500		2022-05-20 10 00 24	321		2022-05-20 10 00 24
15	2022-05-20 10 00 30	2995		2022-05-20 10 00 30	90		2022-05-20 10 00 30	500		2022-05-20 10 00 30	321		2022-05-20 10 00 30
16	2022-05-20 10 00 36	31935		2022-05-20 10 00 36	91		2022-05-20 10 00 36	500		2022-05-20 10 00 36	321		2022-05-20 10 00 36

Figura 60 - Apps Scripts

```

1  /**
2  * Function doGet: Parse received data from GET request,
3  * get and store data which is corresponding with header row in Google Spreadsheet
4  */
5  function doGet(e) {
6  Logger.log( JSON.stringify(e) ); // view parameters
7  var result = 'Ok'; // assume success
8  if (e.parameter == 'undefined') {
9  result = 'No Parameters';
10 }
11 else {

```

Código 9 - Google Sheets API

```

/**
 * Function doGet: Parse received data from GET request,
 * get and store data which is corresponding with header row in Google
 * Spreadsheet
 */
function doGet(e) {
  Logger.log( JSON.stringify(e) ); // view parameters
  var result = 'Ok'; // assume success
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = "****chave apagada****"; // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet(); //
get Active sheet
    var newRow = sheet.getLastRow() + 1;
    var rowData = [];
    d=new Date();

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);

```

```

switch (param) {
    case 'Pirani': //Parameter
        rowData[0] = Utilities.formatDate(d, 'America/Sao_Paulo', 'yyyy-
MM-dd HH:mm:ss'); // Timestamp in column A
        rowData[1] = value; //Value in column B
        result = 'Written on column B';
        break;
    case 'MFC1': //Parameter
        rowData[3] = Utilities.formatDate(d, 'America/Sao_Paulo', 'yyyy-
MM-dd HH:mm:ss'); // Timestamp in column A
        rowData[4] = value; //Value in column C
        result += ' ,Written on column C';
        break;
    case 'MFC2': //Parameter
        rowData[6] = Utilities.formatDate(d, 'America/Sao_Paulo', 'yyyy-
MM-dd HH:mm:ss'); // Timestamp in column A
        rowData[7] = value; //Value in column C
        result += ' ,Written on column D';
        break;
    case 'MFC3': //Parameter
        rowData[9] = Utilities.formatDate(d, 'America/Sao_Paulo', 'yyyy-
MM-dd HH:mm:ss'); // Timestamp in column A
        rowData[10] = value; //Value in column C
        result += ' ,Written on column E';
        break;
    case 'MFC4': //Parameter
        rowData[12] = Utilities.formatDate(d, 'America/Sao_Paulo', 'yyyy-
MM-dd HH:mm:ss'); // Timestamp in column A
        rowData[13] = value; //Value in column C
        result += ' ,Written on column F';
        break;
    default:
        result = "unsupported parameter";
}
}
Logger.log(JSON.stringify(rowData));
// Write new row below
var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
newRange.setValues([rowData]);
}
// Return result of operation
return ContentService.createTextOutput(result);
}
/**
 * Remove leading and trailing single or double quotes
 */
function stripQuotes( value ) {
    return value.replace(/^[\"']|['\"]$/g, "");
}
}

```

O *Grafana Cloud*, camada gratuita, foi utilizado como servidor em nuvem para conectar à planilha do *Google Sheets* com os dados de telemetria e transformar em dashboards (Figura 61). Os dados são atualizados automaticamente a cada 6 segundos. As consultas são realizadas através de um conector através da API que solicita os valores das colunas da planilha através de uma chave secreta que apenas o usuário final tem acesso (Figura 62).

Figura 61 - Dashboard no Grafana



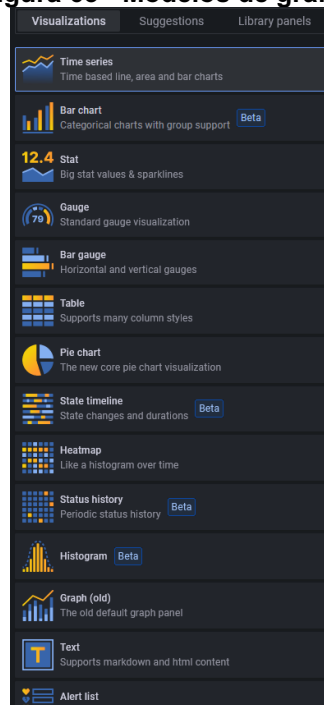
Figura 62 - Consulta planilha

The image shows the configuration interface for a Google Sheets data source in Grafana. The title is '(Google Sheets)'. There are four main configuration rows:

- Spreadsheet ID:** A text field containing a redacted ID, with a red arrow pointing to it and the text 'Chave de acesso' below.
- Range:** A text field containing 'Sheet1!D:E'.
- Cache Time:** A dropdown menu set to '5s'.
- Use Time Filter:** A toggle switch that is currently turned on.

Além disso, o Grafana permite criar outras funções durante a visualização (soma, adição, derivadas etc.), alterar configurações dos gráficos (cor, texto, nome das variáveis, unidades de medida etc.) e sendo possível utilizar mais de 20 modelos de gráficos (Figura 63).

Figura 63 - Modelos de gráfico



4.4 Testes do protótipo

Os testes realizados visaram certificar que o *hardware*, *firmware* e *software* estão desempenhando suas funções e, assim, cumprindo os objetivos do projeto. O Arduino Mega original utilizado queimou durante o desenvolvimento e para os testes do protótipo final foi utilizado um Arduino Mega genérico o que pode ter ocasionado em alterações no resultado.

Para o sensor Pirani, foi verificado o sinal gerado e comparado ao sinal de referência do multímetro digital ICEL (2020) modelo MD-6111, calibração de fábrica, em escala de 20 V e exatidão $\pm 0,75\%$, representado como “exatidão” no manual (Figura 64). O sinal de referência apresentado é a média dos valores medidos pelo multímetro no período da medição. Além disso, foi avaliado sua repetibilidade considerando: mesmo procedimento de medição, mesmo instrumento de medição, mesmo local e repetição em um intervalo de 1 minuto (JCGM:100, 2008).

Para os controladores de fluxo de massa, foi comparado os valores de *PWM* enviados ao MFC com os valores lidos no microcontrolador. Para o software, foi enviado dados simulados pela porta serial do Arduino, já que não foi possível realizar o teste no laboratório por conta do sinal de rede, roteado pelo celular, ser fraco. Além disso, o ESP01 não está programado para conectar-se à rede *Wi-Fi* por certificado PEAP e MSCHAPv2, utilizado na rede disponibilizada pela UTFPR.

Figura 64 – Especificações multímetro para tensão contínua

ESCALA	RESOLUÇÃO	EXATIDÃO	IMPEDÂNCIA	SOBRECARGA
200mV	100 μ V	$\pm(0,5\%+5d)$	$\geq 10M\Omega$	250VDC/ACpico
2V	1mV	$\pm(0,5\%+3d)$		1.000VDC/ VACpico
20V	10mV			
200V	100mV			
1.000V	1V	$\pm(1,0\%+10d)$		

Fonte: ICEL (2021)

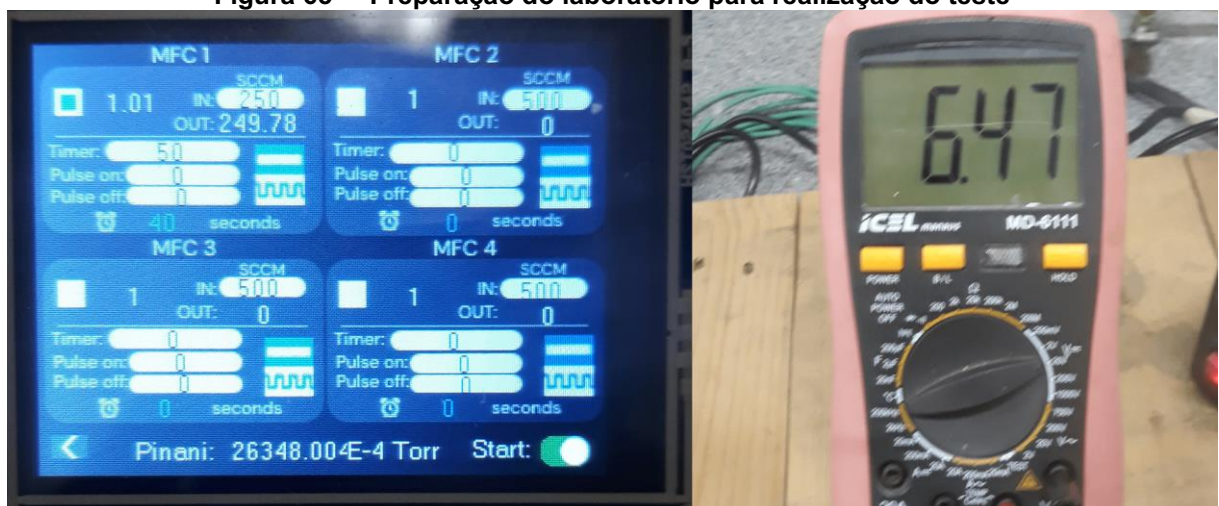
Em todos os testes foi utilizado o gás hidrogênio -fator de correção igual a 1.01- e controladores de fluxo de massa calibrados para nitrogênio -que possui o fator de correção igual 1-. O tempo de duração para cada teste foi variado aleatoriamente para garantir que o protótipo atende diferentes valores.

Todos os testes foram realizados utilizando o protótipo final e com a presença dos usuários do reator, assim, foi coletado e implementado os *feedbacks* da utilização do protótipo.

4.4.1 Leitura da pressão

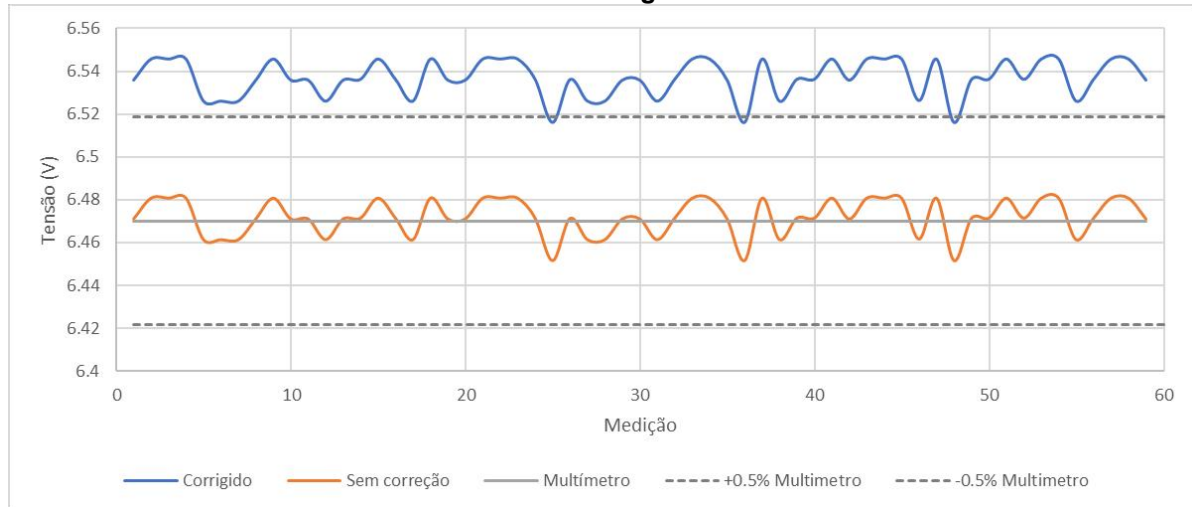
O teste da função de leitura do Pirani consistiu na medição da pressão no reator do laboratório de tratamentos térmicos da UTFPR durante 50 segundos com um fluxo constante de hidrogênio à 250 SCCM. Foi utilizado o sinal do multímetro como referência (Figura 65).

Figura 65 – Preparação do laboratório para realização do teste



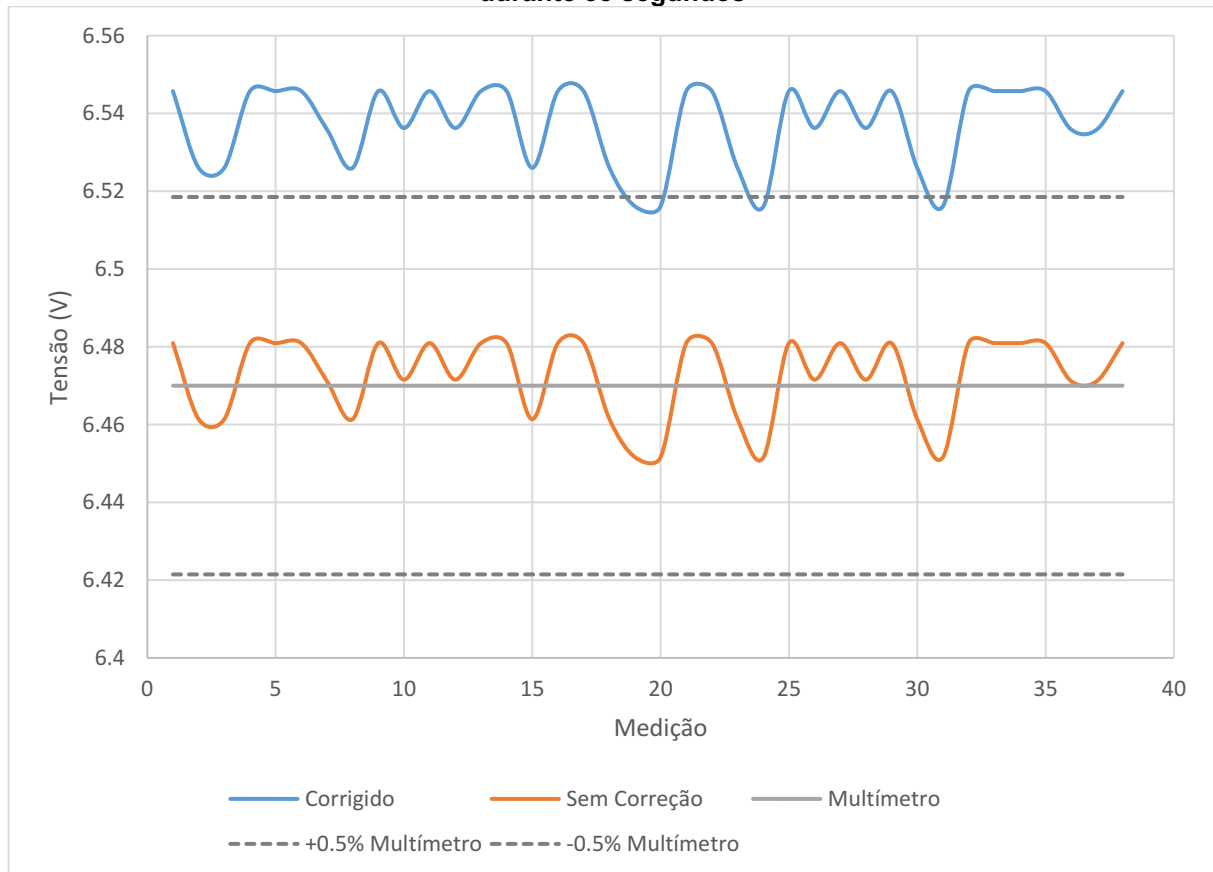
O resultado do teste pode ser visto na Figura 66. A curva azul é o resultado gerado pelo protótipo através do arquivo CSV salvo no cartão microSD. A curva laranja é o resultado encontrado se o fator de correção do gás for retirado.

Figura 66 - Leitura da pressão no reator com fluxo constante de hidrogênio à 250 SCCM durante 50 segundos

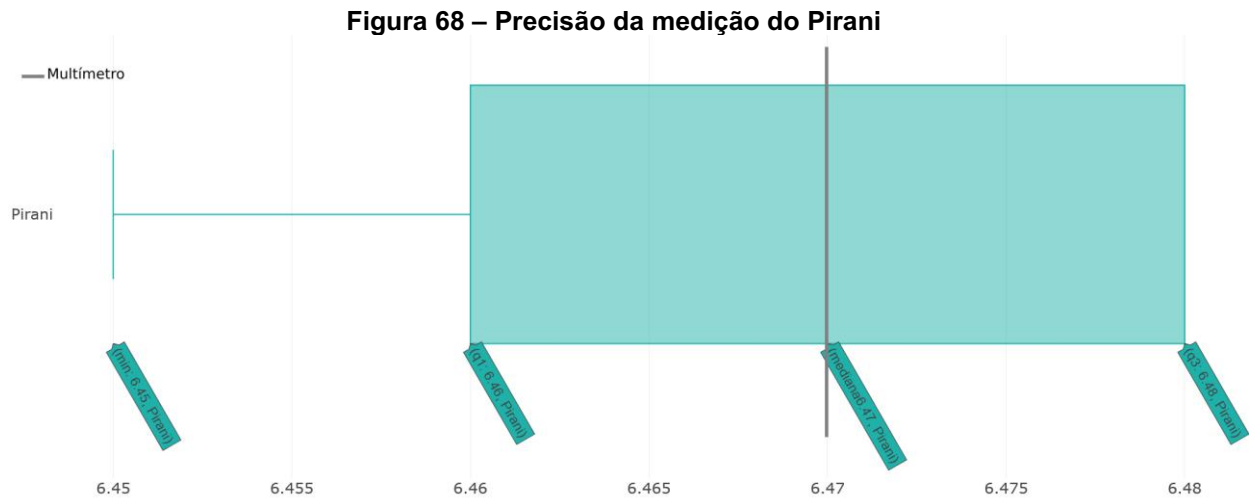


Como mencionado anteriormente, o teste foi repetido a fim de comprovar a sua repetibilidade e que as variáveis no microcontrolador se reiniciaram corretamente. O segundo teste tendo duração de 35 segundos (Figura 67).

Figura 67 – Leitura de pressão no reator com fluxo constante de hidrogenio à 250 SCCM durante 35 segundos



A precisão da medição da leitura do Pirani foi representada na Figura 68. O cálculo se baseia no desvio padrão amostral das duas medições e desconsiderando os valores corrigidos.



4.4.2 Leitura e controle do fluxo de massa

4.4.2.1 Função contínua

O teste da função contínua foi realizado durante 130 segundos. Foram realizados 5 testes variando a abertura da válvula entre: 10, 15, 50, 100, 250 e 500 SCCM (Figura 69 e 70). Segundo manual da MKS (2013), o *MFC* suporta um controle de abertura dentre 2% a 100%, que é equivalente entre 10 SCCM a 500 SCCM. Porém, o protótipo não conseguiu controlar o sensor com valores abaixo de 15 SCCM, por isso, o teste para 10 SCCM foi interrompido antes do fim.

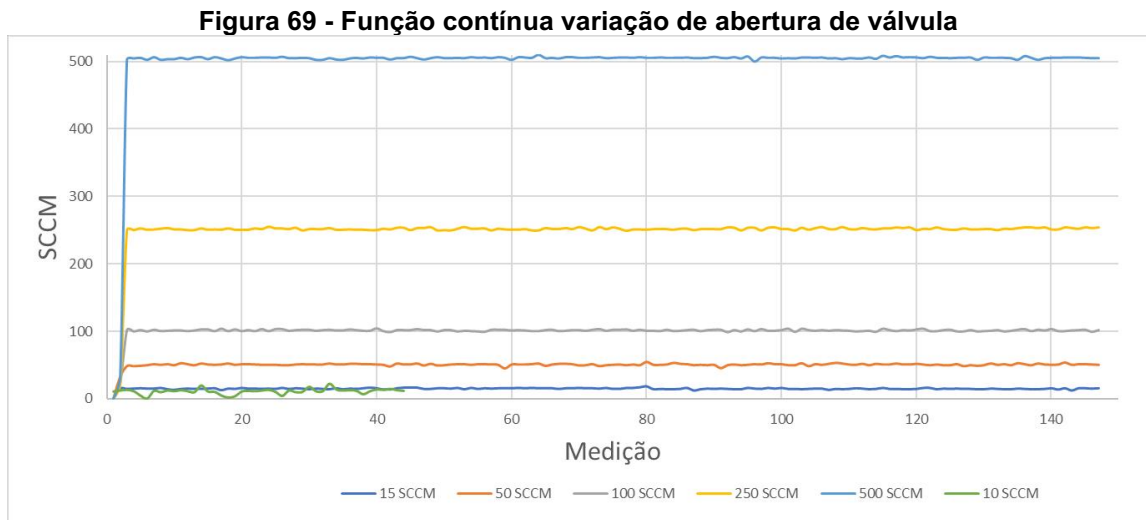
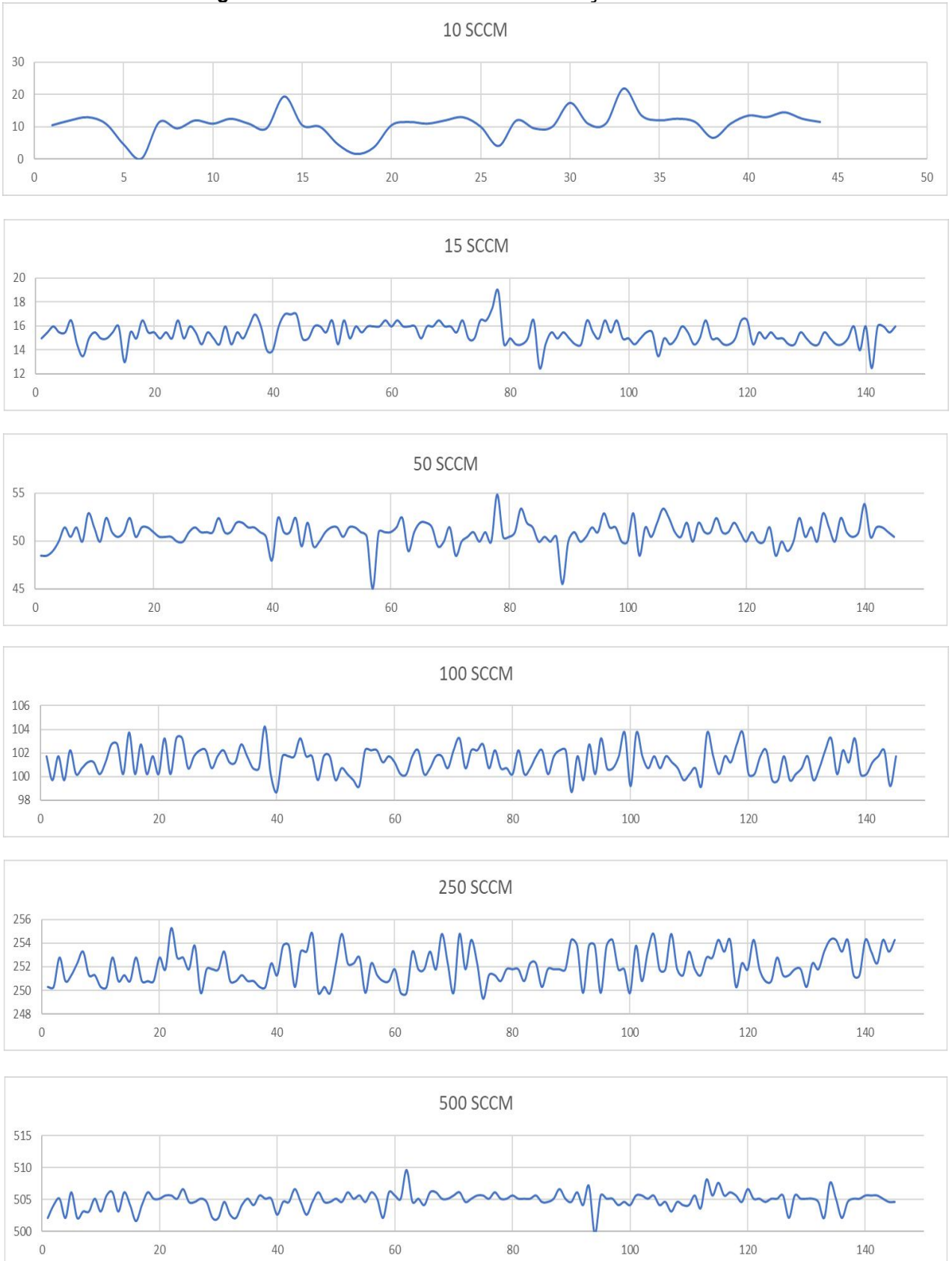
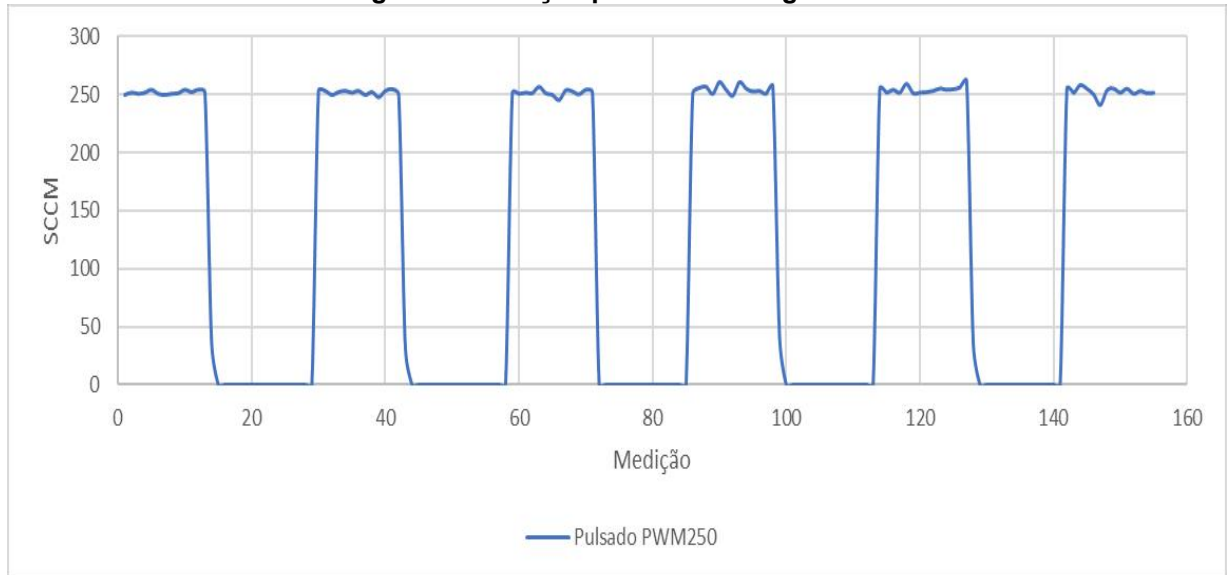


Figura 70 - Resultados individuais da função contínua

4.4.2.2 Função pulsada

O teste da função pulsada foi realizado durante 143 segundos com um pulso de 13 segundos com válvula aberta e 13 segundos com a válvula fechada. Foi utilizado um fluxo constante de hidrogênio à 250 SCCM (Figura 71).

Figura 69 - Função pulsada 165 segundos



4.4.3 Módulo Wi-Fi

O teste do envio das medições via *Wi-Fi* foi realizado durante 300 segundos utilizando a função contínua. Como foi mencionado anteriormente, os dados aleatórios foram simulados através do software Nextion e enviados via porta serial do Arduino (Figura 72). Porém não foi possível gerar os dados simulados do sensor Pirani. O ESP01 limita-se a enviar dados em um intervalo mínimo de 6 segundos (Quadro 8).

Figura 70 - Dashboard Grafana

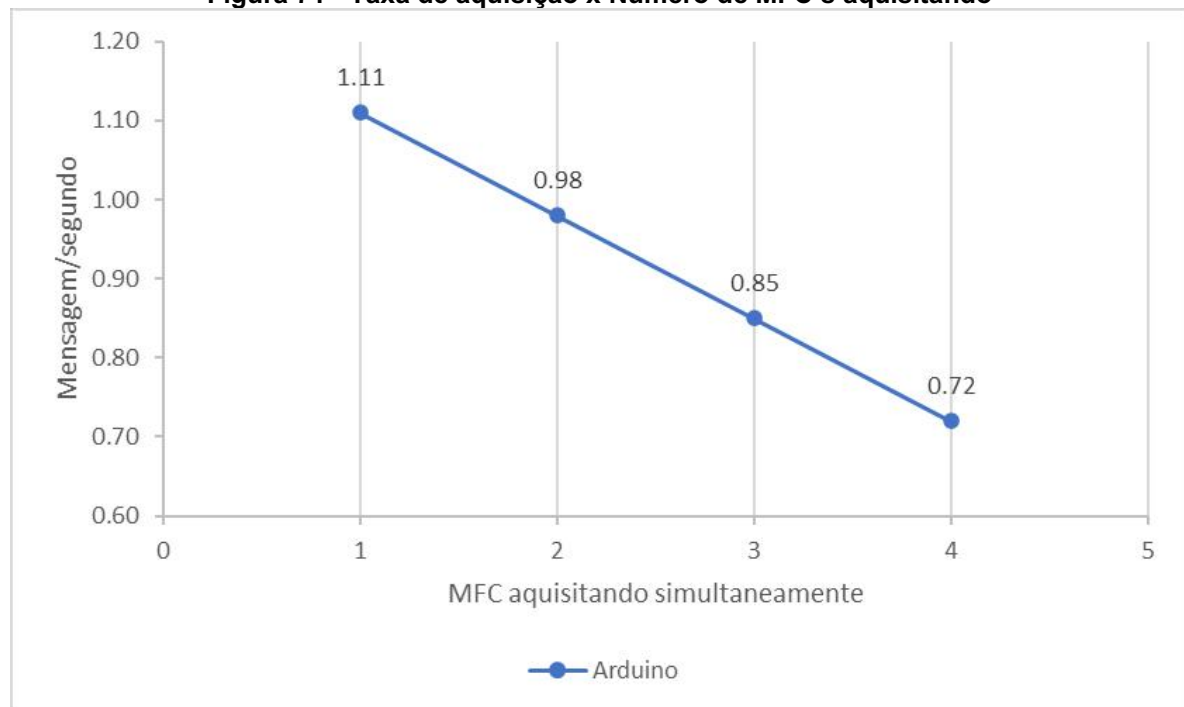


Quadro 8 - Dados coletados pelo ESP01

PIRANI		MFC1		MFC2		MFC3		MFC4	
Data e hora	medição	Data e hora	medição	Data e hora	medição	Data e hora	medição	Data e hora	medição
05/06/2022 22:36:52	0.01	05/06/2022 22:36:52	98.24	05/06/2022 22:36:52	180.35	05/06/2022 22:36:52	169.6	05/06/2022 22:36:52	174
05/06/2022 22:36:58	0.01	05/06/2022 22:36:58	98.24	05/06/2022 22:36:58	85.04	05/06/2022 22:36:58	137.34	05/06/2022 22:36:58	153.4
05/06/2022 22:37:03	0.01	05/06/2022 22:37:03	0	05/06/2022 22:37:03	76.74	05/06/2022 22:37:03	131.48	05/06/2022 22:37:03	148.09
05/06/2022 22:37:10	0.01	05/06/2022 22:37:10	0	05/06/2022 22:37:10	75.76	05/06/2022 22:37:10	131.96	05/06/2022 22:37:10	145.65
05/06/2022 22:37:17	0.01	05/06/2022 22:37:17	0	05/06/2022 22:37:17	76.25	05/06/2022 22:37:17	129.52	05/06/2022 22:37:17	145.16
05/06/2022 22:37:23	0.01	05/06/2022 22:37:23	0	05/06/2022 22:37:23	76.74	05/06/2022 22:37:23	131.48	05/06/2022 22:37:23	147.61
05/06/2022 22:37:29	0.01	05/06/2022 22:37:29	100.2	05/06/2022 22:37:29	175.95	05/06/2022 22:37:29	159.82	05/06/2022 22:37:29	150
05/06/2022 22:37:38	0.01	05/06/2022 22:37:38	100.68	05/06/2022 22:37:38	181.33	05/06/2022 22:37:38	164.22	05/06/2022 22:37:38	154
05/06/2022 22:37:44	0.01	05/06/2022 22:37:44	101.66	05/06/2022 22:37:44	181.33	05/06/2022 22:37:44	167.16	05/06/2022 22:37:44	155
05/06/2022 22:37:51	0.01	05/06/2022 22:37:51	101.17	05/06/2022 22:37:51	181.33	05/06/2022 22:37:51	167.16	05/06/2022 22:37:51	155
05/06/2022 22:37:57	0.01	05/06/2022 22:37:57	102.15	05/06/2022 22:37:57	181.82	05/06/2022 22:37:57	167.16	05/06/2022 22:37:57	159
05/06/2022 22:38:03	0.01	05/06/2022 22:38:03	102.15	05/06/2022 22:38:03	183.28	05/06/2022 22:38:03	169.11	05/06/2022 22:38:03	158
05/06/2022 22:38:09	0.01	05/06/2022 22:38:09	102.15	05/06/2022 22:38:09	181.82	05/06/2022 22:38:09	168.13	05/06/2022 22:38:09	158
05/06/2022 22:38:16	0.01	05/06/2022 22:38:16	101.66	05/06/2022 22:38:16	182.8	05/06/2022 22:38:16	166.67	05/06/2022 22:38:16	156
05/06/2022 22:38:22	0.01	05/06/2022 22:38:22	102.64	05/06/2022 22:38:22	183.77	05/06/2022 22:38:22	166.18	05/06/2022 22:38:22	159
05/06/2022 22:38:28	0.01	05/06/2022 22:38:28	102.64	05/06/2022 22:38:28	182.31	05/06/2022 22:38:28	166.67	05/06/2022 22:38:28	159
05/06/2022 22:38:34	0.01	05/06/2022 22:38:34	102.15	05/06/2022 22:38:34	182.31	05/06/2022 22:38:34	166.67	05/06/2022 22:38:34	157
05/06/2022 22:38:41	0.01	05/06/2022 22:38:41	102.64	05/06/2022 22:38:41	182.31	05/06/2022 22:38:41	166.18	05/06/2022 22:38:41	159
05/06/2022 22:38:47	0.01	05/06/2022 22:38:47	103.13	05/06/2022 22:38:47	182.31	05/06/2022 22:38:47	167.64	05/06/2022 22:38:47	156
05/06/2022 22:38:53	0.01	05/06/2022 22:38:53	103.62	05/06/2022 22:38:53	182.8	05/06/2022 22:38:53	167.16	05/06/2022 22:38:53	159
05/06/2022 22:38:59	0.01	05/06/2022 22:38:59	103.13	05/06/2022 22:38:59	182.8	05/06/2022 22:38:59	166.18	05/06/2022 22:38:59	159
05/06/2022 22:39:07	0.01	05/06/2022 22:39:07	102.64	05/06/2022 22:39:07	183.77	05/06/2022 22:39:07	168.13	05/06/2022 22:39:07	158
05/06/2022 22:39:13	0.01	05/06/2022 22:39:13	104.11	05/06/2022 22:39:13	182.8	05/06/2022 22:39:13	167.16	05/06/2022 22:39:13	157
05/06/2022 22:39:18	0.01	05/06/2022 22:39:18	103.13	05/06/2022 22:39:18	182.31	05/06/2022 22:39:18	167.16	05/06/2022 22:39:18	159
05/06/2022 22:39:25	0.01	05/06/2022 22:39:25	102.64	05/06/2022 22:39:25	183.28	05/06/2022 22:39:25	168.13	05/06/2022 22:39:25	159
05/06/2022 22:39:31	0.01	05/06/2022 22:39:31	103.13	05/06/2022 22:39:31	183.77	05/06/2022 22:39:31	166.67	05/06/2022 22:39:31	156
05/06/2022 22:39:37	0.01	05/06/2022 22:39:37	103.13	05/06/2022 22:39:37	182.8	05/06/2022 22:39:37	167.64	05/06/2022 22:39:37	158
05/06/2022 22:39:43	0.01	05/06/2022 22:39:43	104.11	05/06/2022 22:39:43	182.31	05/06/2022 22:39:43	169.6	05/06/2022 22:39:43	155
05/06/2022 22:39:50	0.01	05/06/2022 22:39:50	102.64	05/06/2022 22:39:50	183.28	05/06/2022 22:39:50	166.18	05/06/2022 22:39:50	159
05/06/2022 22:39:56	0.01	05/06/2022 22:39:56	103.13	05/06/2022 22:39:56	183.28	05/06/2022 22:39:56	165.69	05/06/2022 22:39:56	159
05/06/2022 22:40:02	0.01	05/06/2022 22:40:02	103.13	05/06/2022 22:40:02	182.31	05/06/2022 22:40:02	166.67	05/06/2022 22:40:02	158
05/06/2022 22:40:08	0.01	05/06/2022 22:40:08	103.62	05/06/2022 22:40:08	182.8	05/06/2022 22:40:08	167.64	05/06/2022 22:40:08	159
05/06/2022 22:40:14	0.01	05/06/2022 22:40:14	106.55	05/06/2022 22:40:14	188.17	05/06/2022 22:40:14	165.69	05/06/2022 22:40:14	0
05/06/2022 22:40:19	0.01	05/06/2022 22:40:19	107.04	05/06/2022 22:40:19	187.19	05/06/2022 22:40:19	167.64	05/06/2022 22:40:19	0
05/06/2022 22:40:25	0.01	05/06/2022 22:40:25	107.04	05/06/2022 22:40:25	186.71	05/06/2022 22:40:25	167.64	05/06/2022 22:40:25	0
05/06/2022 22:40:32	0.01	05/06/2022 22:40:32	107.04	05/06/2022 22:40:32	187.19	05/06/2022 22:40:32	166.67	05/06/2022 22:40:32	0
05/06/2022 22:40:39	0.01	05/06/2022 22:40:39	107.53	05/06/2022 22:40:39	187.68	05/06/2022 22:40:39	168.13	05/06/2022 22:40:39	0
05/06/2022 22:40:44	0.01	05/06/2022 22:40:44	107.04	05/06/2022 22:40:44	188.66	05/06/2022 22:40:44	166.67	05/06/2022 22:40:44	0
05/06/2022 22:40:51	0.01	05/06/2022 22:40:51	107.04	05/06/2022 22:40:51	188.17	05/06/2022 22:40:51	166.67	05/06/2022 22:40:51	0
05/06/2022 22:40:57	0.01	05/06/2022 22:40:57	104.11	05/06/2022 22:40:57	186.71	05/06/2022 22:40:57	168.13	05/06/2022 22:40:57	0
05/06/2022 22:41:02	0.01	05/06/2022 22:41:02	107.04	05/06/2022 22:41:02	187.68	05/06/2022 22:41:02	167.16	05/06/2022 22:41:02	0
05/06/2022 22:41:11	0.01	05/06/2022 22:41:11	107.53	05/06/2022 22:41:11	186.22	05/06/2022 22:41:11	168.13	05/06/2022 22:41:11	0
05/06/2022 22:41:17	0.01	05/06/2022 22:41:17	107.04	05/06/2022 22:41:17	186.22	05/06/2022 22:41:17	168.13	05/06/2022 22:41:17	0
05/06/2022 22:41:23	0.01	05/06/2022 22:41:23	107.04	05/06/2022 22:41:23	186.71	05/06/2022 22:41:23	167.64	05/06/2022 22:41:23	0
05/06/2022 22:41:29	0.01	05/06/2022 22:41:29	106.55	05/06/2022 22:41:29	186.22	05/06/2022 22:41:29	165.2	05/06/2022 22:41:29	0
05/06/2022 22:41:36	0.01	05/06/2022 22:41:36	106.06	05/06/2022 22:41:36	187.19	05/06/2022 22:41:36	168.13	05/06/2022 22:41:36	0
05/06/2022 22:41:41	0.01	05/06/2022 22:41:41	106.55	05/06/2022 22:41:41	187.19	05/06/2022 22:41:41	163.73	05/06/2022 22:41:41	0
05/06/2022 22:41:48	0.01	05/06/2022 22:41:48	108.02	05/06/2022 22:41:48	186.71	05/06/2022 22:41:48	165.69	05/06/2022 22:41:48	0
05/06/2022 22:41:55	0.01	05/06/2022 22:41:55	107.04	05/06/2022 22:41:55	187.19	05/06/2022 22:41:55	169.6	05/06/2022 22:41:55	0
05/06/2022 22:42:02	0.01	05/06/2022 22:42:02	111.44	05/06/2022 22:42:02	0	05/06/2022 22:42:02	0	05/06/2022 22:42:02	0
05/06/2022 22:42:08	0.01	05/06/2022 22:42:08	110.95	05/06/2022 22:42:08	0	05/06/2022 22:42:08	0	05/06/2022 22:42:08	0
05/06/2022 22:42:14	0.01	05/06/2022 22:42:14	111.93	05/06/2022 22:42:14	0	05/06/2022 22:42:14	0	05/06/2022 22:42:14	0
05/06/2022 22:42:21	0.01	05/06/2022 22:42:21	111.44	05/06/2022 22:42:21	0	05/06/2022 22:42:21	0	05/06/2022 22:42:21	0
05/06/2022 22:42:26	0.01	05/06/2022 22:42:26	111.44	05/06/2022 22:42:26	0	05/06/2022 22:42:26	0	05/06/2022 22:42:26	0

4.4.4 Visão geral

A taxa de aquisição do protótipo foi calculada com equipamento variando o número de *MFC* ligados simultaneamente (Figura 73). Há uma diminuição do tempo de aquisição das medições pelo fato do Arduino não permitir as atividades simultâneas (*Multithreading*), assim, as medições são sequenciais. Além disso, ainda é possível otimizar a programação para aumentar a taxa de aquisição, porém a redução da taxa por número de controladores ligados ainda permanecerá. Assim, a ordem de prioridade de medição é crescente iniciando no *MFC 1*.

Figura 71 - Taxa de aquisição x Número de MFC's aquisitando

4.4.5 Custo do projeto

O Tabela 1 apresenta o custo do protótipo. O cálculo do orçamento não considera os custos dos equipamentos como o sensor Pirani e o módulo de controle de fluxo de massa. Além disso os valores de frete foram retirados de cada produto.

Tabela 1 - Custo do projeto

Componente	Unidade	Preço/unidade (R\$)	Total
Arduino Mega	1	R\$ 200.00	R\$ 200.00
Capacitor 1uF	9	R\$ 0.23	R\$ 2.00
Case 3D	1	R\$ 45.00	R\$ 45.00
Conector DB9	4	R\$ 3.00	R\$ 12.00
Conector RJ45	1	R\$ 4.00	R\$ 4.00
Conector Alimentação	1	R\$ 2.50	R\$ 2.50
Fonte de alimentação 24V	1	R\$ 30.00	R\$ 30.00
Modulo Leitor de microSD	1	R\$ 12.00	R\$ 12.00
Modulo LM2596 StepDown	1	R\$ 15.00	R\$ 15.00
Modulo WiFi ESP8266	1	R\$ 20.00	R\$ 20.00
PCB	1	R\$ 10.00	R\$ 10.00
Resistor 4.7k Ohms	6	R\$ 0.05	R\$ 0.30
Resistor 470 Ohms	8	R\$ 0.05	R\$ 0.40
Resistor 220 Ohms	1	R\$ 0.05	R\$ 0.05
RTC 1302	1	R\$ 10.00	R\$ 10.00
Tela Nextion 320x240	1	R\$ 120.00	R\$ 120.00
Transistor BC337	4	R\$ 0.27	R\$ 1.08
FRETE			R\$ 60.00
TOTAL			R\$ 544.33

O Quadro 9 apresenta o comparativo em uma visão geral das especificações entre o protótipo e os produtos usados como referência.

Quadro 9 - Comparação entre soluções

			
Empresa	MKS Instruments	Brook Instrument	-
Modelo	247	254	v1.0
Dimensões (mm)	240,5x274,3x88,1	211x113x58.8	165,5x74,6x41
Alimentação	100 - 120 Vac	12-24 Vdc ou 100 - 240 Vac	24 Vdc
Material case	Aluminio	ABS Cicolac	PLA
Número de canais	4	4	4 MFC e 1 Pirani
Tela	LED 1 linha x digito 3 1/2 0.56"	LCD 8 linhas x 40 digito	LCD 320x240 Sensível ao toque
Output de sinal	0-5 volts	0-5 volts	0-5 volts
Compatibilidade	Modulo de controle massa (apenas MKS Instruments)	Modulo de controle pressao e massa (apenas Brook Instruments)	MFC: Quadro 4 Pirani: Quadro 6
Armazenamento de dados	Não armazena os dados	"100 anos de retenção"	Cartão microSD até 64GB e 15GB em nuvem
Funções	Alimentação, leitura e controle de set point	Alimentação, leitura, controle de set point e armazenamento das medicoes	Alimentação, leitura, controle de set point, modo de operação e armazenamento das medicoes offline e online
Orçamento	R\$ 21,395.00	R\$ 57,634.50	R\$ 545.00
Revendedor/data Orçamento	Ohmini 21/10/2021	Contech 20/10/2021	Autor 20/05/2022

Assim as funcionalidades do protótipo são:

- Ler e controlar até 4 controladores de fluxo de massa;
- Ler a pressão do sistema;
- Abertura de forma contínua da válvula durante período estipulado pelo usuário;
- Abertura de forma pulsada da válvula durante período estipulado pelo usuário;
- Correção da leitura da pressão e fluxo de massa de acordo com o gás utilizado;
- Customizar o valor do fator de correção de gás;
- Customizar especificações técnicas de cada equipamento MFC (fluxo máximo do equipamento e fator de gás calibrado);
- Gerar relatório online e offline das medições.

5 CONSIDERAÇÕES FINAIS

O protótipo cumpriu com a funcionalidade proposta e as necessidades para o laboratório da UTFPR. Foi possível controlar simultaneamente os 4 MFC MKS 500 SCCM, ler as medições do sensor Pirani APGX-M-NW16/ALI Edward e ter todas as medições salvas no microSD *card*. Os resultados da leitura do sensor Pirani se mostraram mais precisos que as medições realizadas no multímetro, tendo $\sigma=0,01$. Para o controle da válvula MFC, as funções contínua e pulsada cumpriram com os objetivos apesar de apresentar picos no sinal durante o processo. Esses picos podem ser solucionados através de um Arduino Mega original e alteração da resolução de bits do sinal PWM. Além disso, é possível monitorar o tratamento térmico em um período de 6 segundos entre medições através das plataformas do *Google Sheet* e *Grafana*. A taxa de aquisição do protótipo (0,72 mensagens por segundo em carga máxima de trabalho) ficou abaixo do esperado. Porém, essa taxa pode ser facilmente corrigida com a otimização do código.

Assim, o resultado do protótipo mostra uma melhoria significativa no processo de medição e controle dos sensores durante o tratamento térmico de matérias. O sistema criado permite um maior controle do fluxo de gases no reator e aquisição dos dados coletados.

Embora os resultados estejam aquém de uma solução comercial, é de se destacar a inovação implementada deste projeto, visto que não foi encontrado equipamento que permita diferentes modos de operação automatizados, conexão com a internet, interface gráfica amigável, integração do sensor Pirani no equipamento e preço acessível.

O desenvolvimento do projeto continuará mesmo após a entrega do trabalho de conclusão do curso. A replicação do projeto do zero exige conhecimento técnico, porém para diminuir essa barreira, foi desenvolvido um manual auxiliar os futuros usuários que queiram contribuir no desenvolvimento do projeto (Apêndice E).

Como resultado acadêmico, este trabalho possibilitou o aprendizado e aplicação de conhecimentos em diversas áreas tanto técnicas (sistema) quanto sociais (analisar e entender as necessidades do usuário). Integrando conhecimentos de engenharia mecânica, eletrônica e computação. Além disso, colaborando com o desenvolvimento de pesquisas dentro do laboratório da UTFPR.

5.1 Trabalhos Futuros

Os resultados obtidos mostram que o sistema desenvolvido funciona corretamente dentro do que foi proposto, assim, atendendo todos os requisitos do projeto. Além disso, o protótipo poderia ter aprimoramentos:

- Utilização de um hardware com *Wi-Fi* embutido e com mais de um núcleo, assim, permitindo multitarefas;
- Utilização de um processador de fácil integração na placa PCB;
- Otimização da programação;
- Garantir a não utilização de um *hardware* genérico;
- Utilização de um roteador de internet dentro do laboratório;
- Remodelar o case para montagem por encaixe clicáveis ao invés de parafusos;
- Função para alteração de data e hora do RTC;

REFERÊNCIAS

- ABDUL-QAWAY, A. **The Internet of Things (IoT): An Overview**. SUMAIT University, 2015.
- AHMED, S. **Modern Data Formats for Big Bioinformatics Data Analytics**. International Journal of Advanced Computer Science and Applications, vol. 8, n. 4, 2017.
- ALVES, C. **Nitretação a plasma fundamentos e aplicações**. v1, p. 6-16, 2001.
- ARDUINO. **Tech specs**. Disponível em: < <https://store.arduino.cc/products/arduino-mega-2560-rev3>>. Acesso em: 02 de abril de 2022.
- ARDUINO. **What is Arduino?**. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction/>>. Acesso em: 30 de outubro de 2021.
- AZOB. **Knup fonte**. Disponível em: < <https://www.azob.com.br/none-37680710>>. Acesso em: 05 de maio de 2022.
- BISHOP, C. **Vacuum Deposition onto Webs, Films and Foils**. Computer Methods in Applied Mechanics and Engineering, v. 3, p. 62–63, 2015.
- BROOKS INSTRUMENTS. **Mass Flow Controllers & Meters**. Disponível em: <<https://www.brooksinstrument.com/en/products/mass-flow-controllers>>. Acesso em: 06 de outubro de 2021.
- BRONKHORST. **Thermal mass flow measurement working principle**. Disponível em: <<https://www.bronkhorst.com/int/service-support-1/technologies/thermal-mass-flow-measurement/>>. Acesso em: 06 de outubro de 2021.
- CARVALHO, P. **Information Visualization for CSV Open Data Files Structure Analysis**. University François Rabelais of Tours, Tours, França, 2015.
- CHAO, L. **Research on reliability design of data storage for Embedded System**. School of Information Science and Technology, China, 2012.
- CHOI, Y. **Characteristic test methods of the thermal mass flow controller**. Division of Physical Metrology, Korea Research Institute of Standards and Science, 2013.
- DILLON, R. O. **Thermochromic VO₂ sputtered by control of a vanadium-oxygen emission ratio**. Department of Electrical Engineering and Center for Microelectronic and Optical Materials Research University of Nebraska, 2001.
- EDWARDS VACUUM. **Active Gauges and Controllers**. Disponível em: < <https://www.edwardsvacuum.com/en/our-products/vacuum-measurement/active-gauges>>. Acesso em: 06 de outubro de 2021.
- EDWARDS VACUUM. **Instructions manual – Linear Active Pirani Gauge**. Estocolmo, Suécia, 2007.

EL-SABA, M. H. **Fundamentals of Analog & Digital Signal Processing**. Ain Shams University, 2015.

ESPRESSIF. **ESP32 Series - Datasheet**. Disponível em: <https://datasheet.lcsc.com/lcsc/1912261832_Espressif-Systems-ESP32-D0WD_C130167.pdf>. Acesso em: 01 de janeiro de 2022.

ESPRESSIF. **ESP8266 Series of Modules**. Disponível em: <<https://www.espressif.com/en/products/socs/esp8266/>>. Acesso em: 30 de outubro de 2021.

FLAT UI COLLORS. **UI COLORS PALLETE**. Disponível em: <<https://flatuicolors.com/palette/defo/>>. Acesso em: 07 de abril de 2022.

GIT. **Branching and Merging**. Disponível em: <<https://git-scm.com/about>>. Acesso em: 01 de novembro de 2021.

HELMERSSON, U. **Avaliação de dados de medição — Guia para a expressão de incerteza de medição**. BIPM, 2008. Disponível em: <http://www.inmetro.gov.br/noticias/conteudo/iso_gum_versao_site.pdf>

JCGM 100. **Ionized physical vapor deposition (IPVD): a review of technology and applications**. Linköping University, 2008.

HORIBA. **What is a Mass Flow Controller?**. Disponível em: <https://www.horiba.com/en_en/fluid-measurement-and-control/>. Acesso em: 11 de outubro de 2021.

HULTMAN, L. **Handbook of Hard Coatings – Deposition Technologies, Properties and Applications**. Ed. Bunshaw, p. 111, 2001.

ICEL. **Manual de instruções do multímetro digital modelo MD-6111**. Brasil, 2020.

KOKAJ, A. **Fast fourier transform and a complimentary filter-based control of a robotic system**. Vienna University of Technology, 2018.

KOUTROULIS, E. **Development of a Microcontroller-Based, Photovoltaic Maximum Power Point Tracking Control System**. IEEE transactions on power electronics, vol. 16, n 1, janeiro, 2001.

KOYANAGI, F. **Do You Know About ESP32 ADC Adjustment?** Disponível em: <<https://www.instructables.com/Do-You-Know-About-ESP32-ADC-Adjustment>>. Acesso em: 15 de Dezembro de 2021.

KUMAR, U. **Signal denoising using Fourier Analysis in Python**. Disponível em: <<https://www.earthinversion.com/techniques/signal-denoising-using-fast-fourier-transform/>>. Acesso em: 30 de outubro de 2021.

LAMMERINK, T. S. **Pirani pressure sensor with distributed temperature sensing**. University of Twente, 2001

LAST MINUTE ENGINEERING. **Interfacing Micro SD Card Module with Arduino**. Disponível em: <<https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/>>. Acesso em: 05 de maio de 2022.

MARTINI, I. **Copper for particle accelerators: electron stimulated desorption and study of hydrogen content measurement by laser ablation (msc thesis)**. Scuola di ingegneria dei processi industriali, 2012.

MICROSOFT. **Relational vs. NoSQL data**. Disponível em: <<https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data/>>. Acesso em: 31 de outubro de 2021.

MKS INSTRUMENTS. **Gas Correction Factors for Thermal-based Mass Flow**. Disponível em: < <https://www.mks.com/n/gas-correction-factors-for-thermal-based-mass-flow-controllers>>. Acesso em: 06 de maio de 2022.

MKS INSTRUMENTS. **MKS G-Series Digital Mass Flow Controllers - Instruction Manual**. Estados Unidos da América, Janeiro, 2013.

MKS INSTRUMENTS. **Multi-gas, Multi-range, Elastomer-sealed, 5 - 50,000 sccm Mass Flow Controllers**. Disponível em: < <https://www.mksinst.com/f/ge50a-mass-flow-controller>>. Acesso em: 06 de outubro de 2021.

MORAIS, José. **ESP32 – Analisando e corrigindo o ADC interno**. Disponível em: < <https://www.embarcados.com.br/esp32-adc-interno>>. Acesso em: 15 de Dezembro de 2021.

MUDHOLKAR, P. **A Beginner's Guide to Git and GitHub**. Thakur Institute of Management Studies, Career Development & Research, Mumbai, India, 2017.

NEXTION. **What's Nextion?**. Disponível em: <<https://nextion.tech/>>. Acesso em: 05 de maio de 2022.

ORACLE. **O Que É um Banco de Dados Relacional ?**. Disponível em: <<https://www.oracle.com/br/database/what-is-a-relational-database/>>. Acesso em: 31 de outubro de 2021.

PAARMANN, L. D. **Design and Analysis of Analog Filters: a Signal Processing Perspective**. Kluwer academic publishers, v. 1, p. 23–58, 2001.

PANDIYARAJ, K. **Influence of non-thermal plasma forming gases on improvement of surface properties of low-density polyethylene (LDPE)**. Applied Surface Science, v 307, p 109-11915, 2014.

PEREIRA, M. **Implementação de um Banco de Dados Orientado a Objetos a Partir de um Modelo Dimensional**. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, 2009.

PINEDO, C. **Nitreção por plasma de aços inoxidáveis**. Heat Tech – Tecnologia em Tratamento Térmico e Engenharia de Superfície. Brasil, 2004.

PLECHAWSKA-WOJCIK, M. **Implementação de um Banco de Dados Orientado a Objetos a Partir de um Modelo Dimensional**. Lublin University of Technology, 2017.

POLLARD, T. **Secondary Analysis of Electronic Health Records**. MIT Critical Data, v. 1, cap. 11, p. 101-113, 2016.

QAZIZADEH, A. **On active suspension in rail vehicles**. KTH Royal Institute of Technology, Stockholm, 2017.

SENS4. **What is the working principle of the Pirani gauge?**. Disponível em: <<https://sens4.com/pirani-working-principle.html>>. Acesso em: 06 de outubro de 2021.

SHENOI, B. A. **Introduction to Digital Signal Processing and Filter Design**. Department of Electrical Engineering, Wright State University, v. 1, p. 29-43, 2006.

SPEIDEL, J. **Introduction to digital communications**. Springer, v.2, p. 3–49, 2021.

TEXAS INSTRUMENTS. **What is the working principle of the Pirani gauge LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator**. Disponível em: <https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1653608885656&ref_url=https%253A%252F%252Fwww.google.com%252F>. Acesso em: 05 de maio de 2022.

TISON, S. A. **A critical evaluation of thermal mass flow meters**. National Institute of Standards and Technology, 1996.

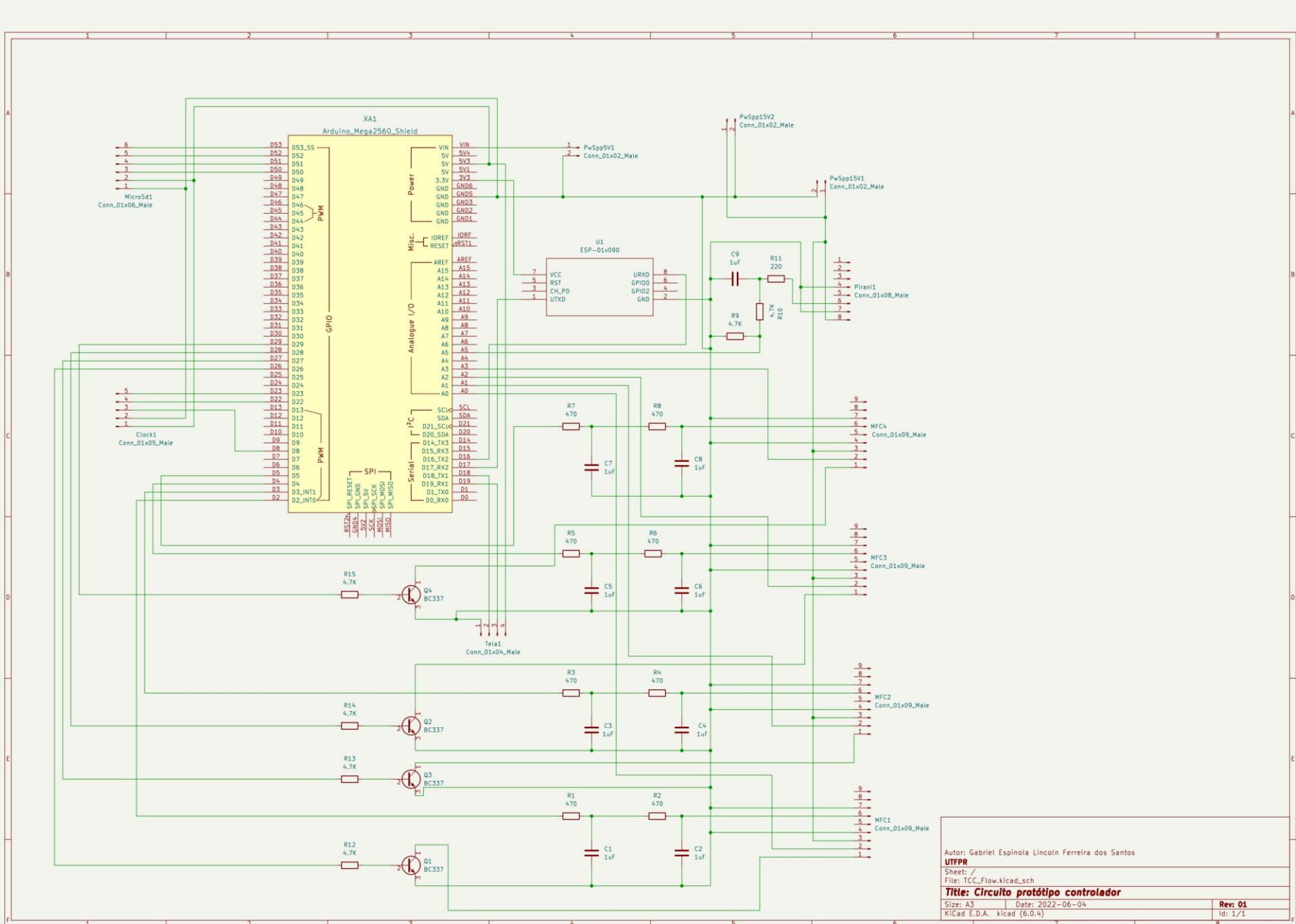
TOPALLI, E. S. **Pirani Vacuum Gauges Using Silicon-on-Glass and Dissolved-Wafer Processes for the Characterization of MEMS Vacuum Packaging**. IEEE sensors journal, v 9, n. 3, 2009

USINA INFO. **Arduino RTC DS1302 Projeto Verificando o Tempo**. Disponível em: <<https://www.usinainfo.com.br/blog/arduino-rtc-ds1302-projeto-verificando-o-tempo/>>. Acesso em: 05 de maio de 2022.

XI, Y. **Improvement of corrosion and wear resistances of AISI 420 martensitic stainless steel using plasma nitriding at low temperature**. Northwestern Polytechnical University, 2007.

WILMSHURST, T. H. **Signal recovery from noise in electronic instrumentation**. Department of Electronics and Computer Science, University of Southampton, v. 2, p. 9-24, 1990.

APÊNDICE A - Circuito Eletrônico



Autor: Gabriel Espinola Lincoln Ferreira dos Santos
UTFRPR
Sheet: /
File: ICC_Flow.kicad_sch
Title: Circuito protótipo controlador
Size: A3 | Date: 2022-06-04 | Rev: 01
Kicad E.D.A. kicad (6.0.4) | Id: 1/1

APÊNDICE B - Quadro de fator de correção de gases

Gas	GFC
NH3	0.73
Ar	1.39
AsH3	0.67
BCl3	0.41
Br2	0.81
CO2	0.7
CO	1
N2	1
CCl4	0.31
CF4	0.42
Cl2	0.86
CHClF2	0.46
C2ClF5	0.24
CClF3	0.38
C2N2	0.61
D2	1
CBr2F2	0.19
B2H6	0.44
CCl2F2	0.35
CHCl2F	0.42
SiH2Cl2	0.4
C2Cl2F4	0.22
C2H2F2	0.43
F2	0.98
CHF3	0.5
CCl3F	0.33
CBrF3	0.37
C2Cl2F3	0.2
C2F6	0.24
C4F8	0.164
He	1.45
H2	1.01
HBr	1
HCl	1
HF	1
C4H8	0.29
Kr	1.543
CH4	0.72
CH3F	0.56
MoF6	0.21
Ne	1.46
NO	0.99
NF3	0.48
N2O	0.71
O2	0.993
C3F8	0.17
COCl2	0.44
C3H8	0.36
C3H6	0.41
SiH4	0.6
SiCl4	0.28
SiF4	0.35
SO2	0.69
SF6	0.26
SiHCl3	0.33
C2Cl3F3	0.2
WF6	0.25
xe	1.32
C5H12	0.22
C2H6	0.5

APÊNDICE C - Modelos 3D do case e arquivos da *PCB*

Os modelos 3D do case podem ser encontrados no repositório do Github do autor:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/tree/main/Hardware/CAD>

Os arquivos de desenvolvimento da PCB podem ser encontrados no repositório do GitHub do autor:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/tree/main/Hardware/PCB%26Eletronic>

APÊNDICE D - Códigos *Firmware* e *Software*

Os códigos referentes ao firmware podem ser encontrados no repositório do GitHub do autor:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/tree/main/project>

Arduino:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/tree/main/project/Arduino>

- **Main.ino:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/main.ino>

- **Controler.h:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/controler.h>

- **Mfc.h:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/mfc.h>

- **Pirani.h:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/pirani.h>

- **Sdcard.h:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/sdcard.h>

- **Touch.h:**

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/touch.h>

- **clock_conf.h:**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/clock_conf.h

- **mcu_conf.h**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/mcu_conf.h

- **mfc_conf.h**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/mfc_conf.h

- **pirani_conf.h**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/pirani_conf.h

- **sdcard_conf.h**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/sdcard_conf.h

- **wifi_conf.h**

https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/Arduino/lib/config/wifi_conf.h

ESP01:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/blob/main/project/ESP01/esp01.ino>

LCD:

<https://github.com/LincolnG4/Flow-and-Pressure-Controller/tree/main/project/LCD>

APÊNDICE E - Manual do usuário

O manual do usuário pode ser encontrado no repositório do GitHub do autor:
<https://github.com/LincolnG4/Flow-and-Pressure-Controller>