UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”
CAMPUS SÃO JOÃO DA BOA VISTA

GUSTAVO DE BARROS OTAVIANO

Gerador de medidas para teste de bancada em Controladores de Pulverização:

Desenvolvimento e Automatização

São João da Boa Vista
2024

Gustavo de Barros Otaviano

Gerador de medidas para teste de bancada em Controladores de Pulverização:

Desenvolvimento e Automatização

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Eletrônica e de Telecomunicações do Campus de São João da Boa Vista, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Eletrônica e de Telecomunicações.

Orientador: Prof. Dr. Wilian Miranda dos Santos

São João da Boa Vista
2024

São João da Boa Vista
2024

O87g

Otaviano, Gustavo de Barros

Gerador de medidas para teste de bancada em controladores de pulverização: : desenvolvimento e automatização / Gustavo de Barros Otaviano. -- São João da Boa Vista, 2024

58 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Telecomunicações) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, São João da Boa Vista

Orientador: Wilian Miranda dos Santos

1. Agricultura de precisão. 2. Equipamento de pulverização. 3. Pulverização. 4. Pulverização e polvilhação na agricultura. 5. Sistemas embarcados (Computadores). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”

FACULDADE DE ENGENHARIA

CAMPUS DE SÃO JOÃO DA BOA VISTA

GRADUAÇÃO EM ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

TRABALHO DE CONCLUSÃO DE CURSO

GERADOR DE MEDIDAS PARA TESTE DE BANCADA EM CONTROLADORES DE PULVERIZAÇÃO: DESENVOLVIMENTO E AUTOMATIZAÇÃO

Aluno: Gustavo de Barros Otaviano

Orientador: Prof. Dr. Wilian Miranda dos Santos

São João da Boa Vista, 18 de novembro de 2024

Banca Examinadora:

Wilian Miranda dos Santos (Orientador)

Welder Fernandes Perina (Examinador)

André Alves Ferreira (Examinador)

A ata de defesa com as respectivas assinaturas dos membros encontra-se no prontuário do aluno (Expediente nº 052/2021)

São João da Boa Vista, 18 de novembro de 2024

DADOS CURRICULARES

GUSTAVO DE BARROS OTAVIANO

NASCIMENTO	28/06/1998 – Santa Rita do Passa Quatro / SP
FILIAÇÃO	Luiz Rogério Otaviano
	Ariadne Garbulho de Barros Otaviano
1º GRAU	SESI 255
2º GRAU	CEC COC – Santa Rita do Passa Quatro
3º GRAU	Engenharia Eletrônica e de Telecomunicações
	UNESP – Campus São João da Boa Vista

A quem me apoiou e contribuiu para que este projeto fosse realizado. Aos meus pais, familiares, orientadores e amigos.

A quem me apoiou e contribuiu para que este projeto fosse realizado. Aos meus pais, familiares, orientadores e amigos.

Agradecimentos

Gostaria de agradecer primeiramente, à minha família que me apoiou constantemente durante todos esses anos e me proporcionou educação, conhecimento e oportunidades ao longo dos anos. Ao professor Rafael Penchel que me ajudou muito durante a graduação e fez eu me apaixonar pelo curso com sua didática. Ao meu gestor Guilherme que me ajudou e ensinou muito durante meu estágio. Ao professor Wilian que me apoiou e auxiliou durante este projeto. E aos meus amigos Caio, Gabriel, Bruno, Paulo e Gean pela amizade e companheirismo durante toda a graduação.

Este trabalho contou com o apoio das seguintes entidades:
Faculdade de Engenharia de São João da Boa Vista – UNESP
Bomsistema - Sistemas de Pressurização de Fluidos

Este trabalho contou com o apoio das seguintes entidades:
Faculdade de Engenharia de São João da Boa Vista – UNESP
Bomsistema - Sistemas de Pressurização de Fluidos

“Quando algo é importante o suficiente, você faz, mesmo que as probabilidades não estejam a seu favor.”
(Elon Musk)

Resumo

O agronegócio está em constante crescimento, se desenvolvendo a cada dia. No Brasil, o agronegócio representa por cerca de 10% do PIB, incluindo a produção agrícola e pecuária, além do processamento de alimentos, fibras e bioenergia. Ele está diretamente ligado à vida da população, onde é um dos principais responsáveis pela alimentação e transporte. Com o aumento populacional, o seu crescimento há de ser proporcional, para suprimir as necessidades, e com isso, é necessário a otimização de processos, como a erradicação de pragas, melhoria na colheita, e aceleração de processos. No projeto desenvolvido, levou-se em consideração, a necessidade da pulverização, que é responsável por erradicar as pragas da lavoura. Atualmente, para otimizar a pulverização e diminuir gastos, os sistemas de pulverização possuem um controlador, responsável por alterar a quantidade de produto utilizado de acordo com a velocidade da máquina e da quantidade de produto necessário por área. Além de ser capaz de ligar ou desligar seções do sistema, para que não haja pulverizações a mais do que o necessário. A fim de reduzir ainda mais esses gastos e facilitar o desenvolvimento deste controlador, o projeto foi idealizado, para criar medidas simuladas de maneira precisa, possibilitando assim, aferir medidas e calibrar o controlador em uma bancada de teste, sem que seja necessário se deslocar até o campo.

Palavras-chave: Agricultura; agricultura de precisão; controlador de pulverização; pulverização; automação.

Abstract

The agribusiness sector is in constant growth, evolving daily. In Brazil, agribusiness accounts for approximately 10% of the GDP, including agricultural and livestock production, as well as the processing of food, fibers, and bioenergy. It is directly connected to the lives of the population, serving as one of the main contributors to food supply and transportation. With population growth, its expansion must be proportional to meet rising needs, necessitating the optimization of processes such as pest eradication, harvest improvement, and process acceleration. The developed project considered the necessity of spraying, which is crucial for eradicating pests from crops.

Currently, to optimize spraying and reduce costs, spraying systems are equipped with a controller responsible for adjusting the amount of product used according to the machine's speed and the required quantity per area. It also controls the activation or deactivation of system sections to prevent unnecessary spraying. To further reduce costs and facilitate the development of this controller, the project was created to simulate precise measurements, enabling the calibration and validation of the controller on a test bench without the need to travel to the field.

Keywords: Agriculture, precision agriculture, spray controller, spraying, automation.

Lista de Figuras

Figura 1 - Painel controlador [12] (à direita) e monitor de um controlador de pulverização com GPS agrícola [13] (à esquerda). 21
Figura 2 - Módulo controlador com conexões rotuladas. 25
Figura 3 - Fonte de alimentação com múltiplas tensões de saída. 26
Figura 4 - Circuitos de filtragem e estabilização dos sinais analógicos 27
Figura 5 - Circuito Conversor PWM com leitor de tensão. 28
Figura 6 - Sinal de saída referente à vazão quando a Vazão de entrada é igual a 0L/min. .. 37
Figura 7 - Circuito gerador de 4 a 20mA. 38
Figura 8 - Gráfico tensão/Valor analógico, variando o valor analógico de 0 a 4095. 41
Figura 9 - Gráfico tensão/Valor analógico, variando o a tensão de 0 a 600mV. 42
Figura 10 - Gráfico tensão/Valor analógico, onde a tensão varia de 600mV até seu valor máximo. 43
Figura 11 - Circuito regulador. 48
Figura 12 - Circuito de leitura do controlador de corte de seções. 49
Figura 13 - Layout Final do LCD. 63
Figura 14 - Amostragem da velocidade quando o potenciômetro atinge o valor máximo.. 63
Figura 15 - Frequência referente à velocidade máxima (15km/h). 64
Figura 16 - Frequência referente à uma velocidade igual a 6,24km/h 65
Figura 17 - Sinal reajustado para nível lógico baixo quando a velocidade é igual a 0km/h. 66
Figura 18 - Frequência referente à vazão máxima (50L/min). 67
Figura 19 - Tensão medida quando a pressão amostrada é igual a 2Bar. 68
Figura 20 - Pressão amostrada referente à uma pressão de 628mV. 69
Figura 21 - Reguladora em estado ideal. 69
Figura 22 - Reguladora em estado ideal. 70
Figura 23 - Reguladora ordenando elevar a vazão 70
Figura 24 - Conectores referentes à reguladora e ao leitor de seções. 71
Figura 25 - Simulação da conexão do chicote elétrico, indicando que a seção está ligada. 71
Figura 26 - Seção geral ativa. 72
Figura 27 - Seção 1 ativa. 72
Figura 28 - Seção 2 ativa. 72
Figura 29 - Seção 3 ativa. 73

Sumário

1. Introdução 21
   1.1. Motivação 21
   1.2. Objetivo 22
   1.3. Organização do texto 22
2. Desenvolvimento 24
   2.1. Microcontrolador 24
   2.2. Fonte Reguladora 25
   2.3. Simulador de velocidade e vazão 26
      2.3.1. Circuito esquemático 26
      2.3.2. Software de Automação 29
   2.4. Gerador de 4 a 20mA 37
      2.4.1. Circuito esquemático 37
      2.4.2. Análise do comportamento de tensão 39
      2.4.3. Software de Automação 44
   2.5. Reguladora e Leitor de Corte de seções 47
      2.5.1. Circuito esquemático 47
      2.5.2. Explicação do código 51
3. Resultados 62
   3.1. Layout da Tela 62
   3.2. Sinal referente à Velocidade 63
   3.3. Sinal referente à Vazão 66
   3.4. Sinal referente à Pressão 67
   3.5. Leitura da reguladora 69
   3.6. Leitura da condição das seções 70
4. Conclusão 74

Referências 75

Apêndice A – Código de automação completo 77

1. Introdução

1.1. Motivação

Os sistemas de pulverização, assim como todo o setor agropecuário, se desenvolvem a cada dia e a concorrência aumenta cada vez mais. As indústrias precisam se adequar ao meio e prover desenvolvimento de tecnologias novas, fazendo isso de maneira rápida, prática e com baixo custo, para que assim, obtenha um lucro maior em menor tempo. Para isso é necessário reduzir etapas de processo, sem que haja queda na qualidade do produto. De acordo com o Embrapa, um dos problemas atuais dentro do setor agropecuário, é a necessidade de testar equipamentos em campo, pois é necessário verificar se a vazão, velocidade, pressão da bomba e seções do sistema de pulverização estão funcionando corretamente, e isso requer muito tempo e dinheiro [1], logo, com um gerador de medidas, é possível realizar o teste em bancada e solucionar essas dificuldades.

Figura 1 - Painel controlador [12] (à direita) e monitor de um controlador de pulverização com GPS agrícola [13] (à esquerda).

Fonte: Inel Agricultura, 2024

A fim de solucionar este problema, surgiu a ideia do projeto, sendo possível realizar um sistema embarcado para se utilizar em uma bancada de testes. Isso permite que um

sistema controlador de pulverização seja desenvolvido e testado dentro de um laboratório, sendo necessário ir a campo somente para validação final.

1.2. Objetivo

O objetivo principal do projeto, é criar um sistema embarcado que gera medidas simuladas de maneira precisa, para que o controlador de pulverização seja desenvolvido e calibrado de maneira correta. O projeto deve gerar sinais de velocidade, vazão e pressão que será utilizado para o controlador verificar se há necessidade de aumentar, diminuir ou manter a vazão. O sistema funciona de forma que, caso a velocidade aumente, será necessário suprir essa diferença aumentando a vazão, mantendo uma quantidade de produto por área igual, e para essa vazão aumentar, é necessário aumentar a pressão do sistema. Além dos sinais transmitidos, também é de extrema importância receber sinais e analisar se o controlador está realizando suas funções corretamente. As funções que serão recebidas do controlador referem-se à reguladora e ao sistema de corte de seções, onde a reguladora indicará se será necessário aumentar ou diminuir a vazão e as seções serão ligadas ou desligadas de acordo com a rota de pulverização.

1.3. Organização do texto

O trabalho está organizado da seguinte maneira:

No capítulo 2 é apresentado todo o processo de desenvolvimento de cada circuito individual para embarcá-los na placa PCB e o software utilizado para validar se o funcionamento está correto. O esquemático foi gerado através do *software* Ki CAD e o seu *software* foi desenvolvido utilizando o Arduino IDE e o módulo ESP32 DEVKIT V1 [4].

No Apêndice A é apresentado o script definitivo, que possibilita o funcionamento de todos os circuitos de maneira simultânea. Ademais, o circuito é comentado e explicado para definir a necessidade de cada função implementada.

O capítulo 3 afere os resultados obtidos após alcançar as medidas de forma simultânea, onde demonstra que os circuitos não interferem entre si e apresentam as medidas de acordo com o esperado.

Por fim, no capítulo 4, são apresentadas as conclusões do projeto.

2. Desenvolvimento

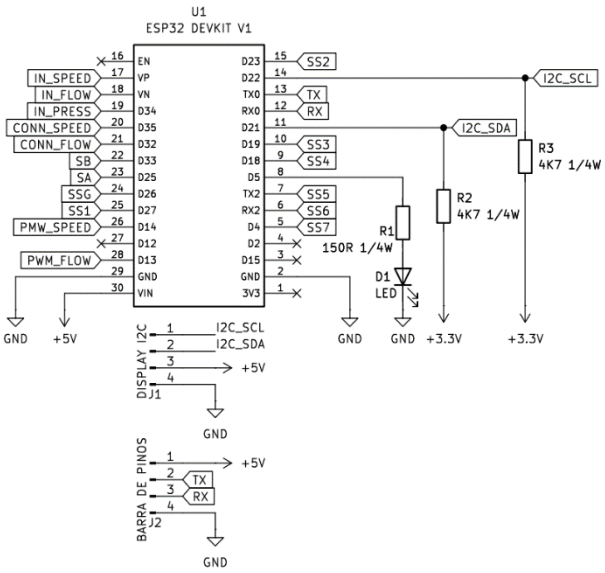
O dispositivo é um gerador de sinais utilizado como para simular o funcionamento de um sistema de pulverização. Com isso, é necessário simular todas as variáveis necessárias que afetam a pulverização, como velocidade, pressão, vazão e quantidade de seções. Como o objetivo, é um sistema automatizado e otimizado, um regulador, que verifica a necessidade de aumentar ou diminuir a pressão, também é necessário.

Para o desenvolvimento do *hardware*, foi-se utilizado o *software* KiCad 8.0 e o auxílio do simulador Multisim, e para o desenvolvimento do *software*, foi utilizado o Arduino IDE versão 1.8.19.

2.1. Microcontrolador

Na Figura 2, é demonstrada as conexões definidas para o microcontrolador. De acordo com o *datasheet* do ESP32 DEVKIT V1 [4], foi definido os pinos ideais para realizar cada função. No caso das conexões I2C, é especificado que o SCL e SDA devem ser conectados nas portas 22 e 21, respectivamente, e que para apresentar o funcionamento correto, é necessário um resistor de pullup de $4k7\Omega$.

Figura 2 - Módulo controlador com conexões rotuladas.



Fonte:Desenvolvida pelo autor, 2024

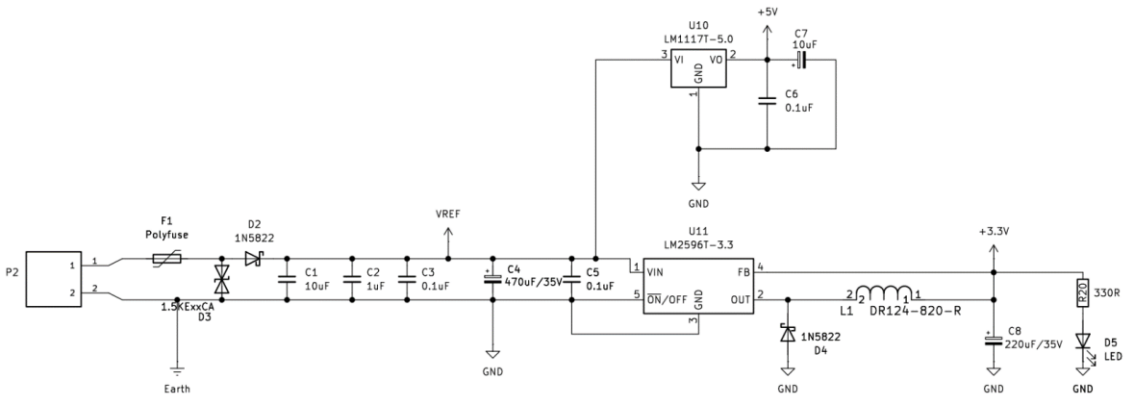
Já o LED (D1) será utilizado para monitorar se o microcontrolador está ligado ou não, ou seja, sempre que o microcontrolador estiver conectado à alimentação, o LED receberá o comando para se manter em nível lógico alto.

O resistor (R1) foi definido como 150Ω seguindo a condição, de alimentar o LED com uma corrente de 20mA. De acordo com o *datasheet* do LED, é possível determinar qual a tensão necessária em cima do LED para atingir 20mA, e assim, subtraindo da tensão total emitida pelo microcontrolador (+3.3V), é possível determinar o valor do resistor necessário.

2.2. Fonte Reguladora

A fonte, demonstrada na Figura 3, possui uma alimentação de entrada igual a 12V, e utilizando um conversor de tensão de 5V $(LM1117T-5.0)$[8] e um regulador de tensão de 3,3V $(LM2596T-3.3)$[7], ela é capaz de alimentar o circuito com todas as tensões necessárias, sem ser necessário utilizar a tensão de saída do microcontrolador.

Figura 3 - Fonte de alimentação com múltiplas tensões de saída.



Fonte: Desenvolvida pelo autor, 2024

Os capacitores do circuito têm como função, filtrar o ruído, tanto de entrada, quanto de saída, para que o sinal seja mais limpo e estável. Como o regulador de tensão é do tipo Buck, o Indutor (L1) é necessário, pois ele armazena energia quando o transistor de comutação está conectado e libera quando está desligado, suavizando e estabilizando a corrente de saída. O LED (D5), tem a mesma função que o LED (D1), que é sinalizar que a fonte está ligada.

Já os demais componentes estão acoplados no circuito para fornecer proteção e segurança. O poli fusível foi inserido para fornecer uma proteção contra sobrecorrente, o diodo 1.5KExxCA (D3), é um diodo supressor que quando conectado entre a alimentação e o terra, protege o circuito contra surtos de tensão que podem ocorrer na entrada. Por fim, os diodos D2 e D4 ($1N5822$) são diodos Schottky inseridos com o intuito de fornecer proteção contra picos de tensão inversa.

2.3. Simulador de velocidade e vazão

2.3.1. Circuito esquemático

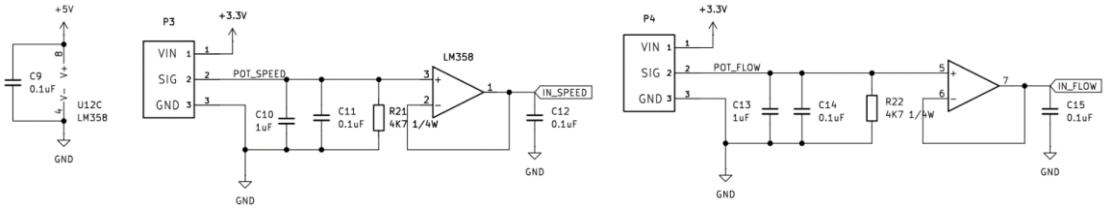
2.3.1.1. Controle de potenciômetros

O circuito de controle de potenciômetros tem como objetivo enviar um sinal analógico para o módulo controlador. Seria possível enviá-lo diretamente, sem estabelecer

este circuito, porém, a instabilidade e quantidade de ruídos são muito grandes, trazendo muitas desvantagens.

A fim de corrigir isto, o circuito referente na Figura 4 foi desenvolvido. O amplificador operacional ($LM358$)[5] age como um buffer, mantendo a integridade do sinal, minimizando ruídos, interferências e distorções, os capacitores filtram o sinal de entrada para inserir no amplificador, diminuindo ainda mais o ruído. E o resistor age como um pulldown, evitando ponto flutuante quando a tensão é próxima de 0V. Os conectores serão ligados à potenciômetros de $10K\Omega$.

Figura 4 - Circuitos de filtragem e estabilização dos sinais analógicos



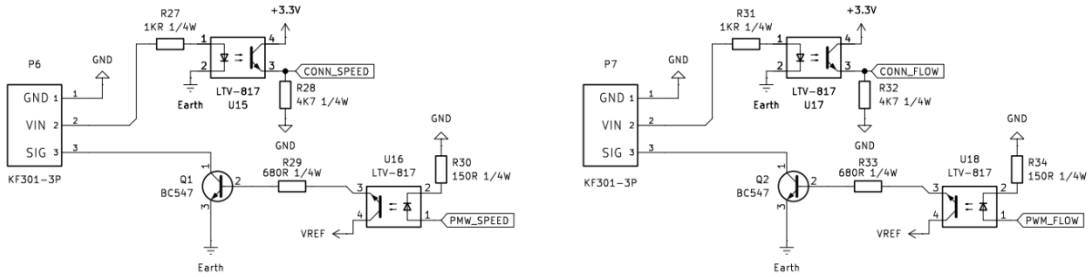
Fonte: Desenvolvido pelo autor, 2024

2.3.1.2. Circuito conversor PWM

O circuito conversor PWM, mostrado na Figura 5, realiza duas funções através de um chicote elétrico, ele emite um sinal PWM do módulo para o controlador de pulverização e recebe uma tensão de entrada de 12V para verificar se o cabo está conectado corretamente.

Levando em consideração a tensão de entrada (12V) no conector, ela acionará o LED presente no opto acoplador, ativando-o, emitindo um sinal de 3,3V até o módulo controlador, indicando que ele está conectado.

Figura 5 - Circuito Conversor PWM com leitor de tensão.



Fonte: Desenvolvido pelo autor, 2024

A corrente necessária para ativar o LED do opto acoplador $(LTV - 817)$ deve estar entre 10mA e 20mA, portanto, através da equação 1, onde V_{in} é a tensão de entrada, R é a resistência e I é a corrente de ativação. Utilizando um resistor de $1k\Omega$, tem-se uma corrente de 12mA. Satisfazendo a condição da corrente necessária.

$$\frac{V_{in}}{R} = I \tag{1}$$

Já o Resistor de pulldown inserido, possui valor de $4{,}7k\Omega$ pois por especificação, um resistor de pulldown deve ter entre $4{,}7k\Omega$ e $47k\Omega$.

Agora, analisando o sinal enviado, quando um pulso positivo é enviado, o LED ativará o opto acoplador e assim, um sinal de 12V será emitido para o transistor BC547. Assim, pela equação 2, define-se como V_E, a tensão emitida, $V_{D_{LED}}$, a queda de tensão interna do LED e $V_{D_{BC}}$, a queda de tensão no BC547 e *R,* a resistência. Aplicando-a, é determinados os resistores de 150Ω, levando em consideração a corrente necessária para a ativação do LED interno do opto acoplador.

$$I = \frac{V_E - V_{D_{LED}} - V_{D_{BC}}}{R} \tag{2}$$

Já os resistores de 680Ω foram calculados a fim de manter uma corrente de aproximadamente 15mA, deste modo, considerando a equação 3, obtendo uma corrente de 15,6mA .

$$I = \frac{V_E - V_{D\,LED}}{R} \quad (3)$$

Por conta do posicionamento do transistor BC547 e do Controlador de pulverização que será conectado na saída, o sinal emitido é o inverso do sinal original, ou seja, quando o módulo emitir um nível lógico alto, a saída estará em nível lógico baixo, e vice e versa. Isso ocorre pois quando chega tensão no coletor do transistor, ele envia o sinal para o terra. Já quando o contrário ocorre, há nível lógico alto estável na saída, o que também é ocasionado por conta de um resistor de pullup conectado na entrada do controlador de pulverização.

Apesar desta mudança, nenhum problema com o sinal será acarretado, já que o sinal tem como característica, um "duty cycle" de 50%, portanto o tempo de trabalho não alterará, apesar de funcionar em nível lógico inverso. A única diferença é quando a frequência for igual a 0Hz, pois o sinal se manteria em nível lógico alto o tempo todo.

2.3.2. Software de Automação

2.3.2.1. Controle de potenciômetros

A definição de portas definidas na seção 2.1, no esquemático do módulo ESP32 DEVKIT V1, já as variáveis serão utilizadas para aplicar um filtro exponencial com os valores obtidos e manter os valores apresentados estáveis.

```
#define SpeedPin 36
#define FlowPin 39
// Variáveis para Speed
float filteredSpeed = 0;
const float alphaSpeed = 0.1;
unsigned long previousMillisSpeed = 0;
const unsigned long intervalSpeed = 100;
// Variáveis para Flow
float filteredFlow = 0;
const float alphaFlow = 0.1;
```

```
unsigned long previousMillisFlow = 0;
const unsigned long intervalFlow = 100;

void setup() {
  Serial.begin(115200);
  // Configuração dos pinos de entrada para sensores
  pinMode(SpeedPin, INPUT);
  pinMode(FlowPin, INPUT);
}
```

O setup do programa tem a função somente de inicializar a serial e definir as portas como pinos de entrada.

```
float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 100.0;
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}
```

O filtro aplicado é feito pela EMA (Média Móvel Exponencial), que é aplicado no setor de investimento, usualmente em *trading*. Ela consiste no princípio da média móvel comum, que utiliza os últimos dados obtidos, para gerar uma linha de tendência suave, porém, a diferença é que a média móvel comum, utiliza todos os valores da média com pesos iguais, estabelecendo a mesma relevância entre eles, já a média móvel exponencial, atribui maior relevância a dados mais recentes.

Deste modo, mapeando o potenciômetro com os novos valores máximos e mínimos, e determinando α, que é o fator de suavização definido por

$$\alpha = \frac{2}{N+1} \tag{4}$$

onde N é igual ao Período, obtendo um resultado melhor e mais estável na serial.

```
void loop() {
  unsigned long currentMillis = millis();
```

```
  // Processamento para Speed
  if (currentMillis - previousMillisSpeed >= intervalSpeed) {
    previousMillisSpeed = currentMillis;
    filteredSpeed = applyExponentialFilter(analogRead(SpeedPin),
filteredSpeed, alphaSpeed, 0, 1500);
  }
  // Processamento para Flow
  if (currentMillis - previousMillisFlow >= intervalFlow) {
    previousMillisFlow = currentMillis;
    filteredFlow = applyExponentialFilter(analogRead(FlowPin),
filteredFlow, alphaFlow, 0, 6500);
  }
  Serial.print ("Speed: ");
  Serial.print (filteredSpeed);
  Serial.print ("     Flow: ");
  Serial.println (filteredFlow);
}
```

E por fim, aplicando o filtro para a velocidade e vazão, é possível analisar se a leitura está correta e estável.

2.3.2.2. Circuito conversor PWM

Para gerar o PWM, no ESP32, a funções ledc são utilizadas, e para conseguir acessá-las corretamente, é necessário adicionar a biblioteca $< Arduino.h >$, e para gerá-la é necessário definir os canais que vão se utilizar, a resolução do sinal e qual vai ser o ciclo de trabalho do sinal.

```
 #include <Arduino.h>

// Velocidade (km/h)
#define SpeedPin 36
#define SpeedPWM 14
```

```
// Vazão (L/min)
#define FlowPin 39
#define FlowPWM 13

// Variáveis para Velocidade
float filteredSpeed = 0;
const float alphaSpeed = 0.2;
unsigned long previousMillisSpeed = 0;
const unsigned long intervalSpeed = 100;

// Variáveis para PWM da Velocidade
float SpeedFrequency = 0;
float SpeedPeriod = 0;

// Variáveis para Vazão
float filteredFlow = 0;
const float alphaFlow = 0.2;
unsigned long previousMillisFlow = 0;
const unsigned long intervalFlow = 100;

// Variáveis para PWM da Vazão
float FlowFrequency = 0;
float FlowPeriod = 0;
```

Há 16 canais para se utilizar, podendo definir de 0 a 15. Já a resolução é predefinida como 8 bits, logo, com uma resolução de 8 bits, tem-se $2^n = 2^8 = 256$ valores possíveis para definir o ciclo de trabalho, o qual varia de 0 a 255. Deste modo, para um ciclo de 50%, utiliza-se *duty cycle* = 127.

```
void setup() {
  Serial.begin(115200);

  pinMode(SpeedPin, INPUT);
  pinMode(FlowPin, INPUT);
```

```
  pinMode(SpeedPWM, OUTPUT);
  pinMode(FlowPWM, OUTPUT);

  // Configuração dos canais PWM para os dois sinais
  ledcSetup(SpeedChannel, 0, pwmResolution);  // Inicialização
do canal de velocidade
  ledcAttachPin(SpeedPWM, SpeedChannel);

  ledcSetup(FlowChannel, 0, pwmResolution);  // Inicialização do
canal de vazão
  ledcAttachPin(FlowPWM, FlowChannel);
}
```

Adicionando agora, no setup, as saídas dos sinais PWM e a inicialização dos canais de pressão e de vazão.

```
float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 100.0; // Escala para 0 a 15.00
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}

void SpeedEquation() {
  SpeedFrequency = 60 * filteredSpeed / 3.6;
  SpeedPeriod = (SpeedFrequency > 0) ? (1000.0 / SpeedFrequency)
: 0;

  if (filteredSpeed <= 0.01) {
    SpeedFrequency = 0;
    SpeedPeriod = 0;
  }
}

void FlowEquation() {
```

```
  FlowFrequency = 1015 * filteredFlow / 60;
  FlowPeriod = (FlowFrequency > 0) ? (1000.0 / FlowFrequency) :
0;

  if (filteredFlow <= 0.01) {
    FlowFrequency = 0;
    FlowPeriod = 0;
  }
}
```

As funções “SpeedEquation” e “FlowEquation”, são definidas para transformar os valores de velocidade e pressão para suas frequências respectivas, tornando possível gerar um sinal PWM.

Para a velocidade, é necessário transformar a velocidade, representada por V, na equação 5, já que será necessário utilizar a unidade de medida de m/s.

$$V\left[\frac{km}{h}\right] = \frac{V[m/s]}{3.6} \tag{5}$$

Por fim, para gerar a frequência, é necessário verificar a cada minuto, como é demonstrado na equação 6.

$$F_{velocidade} = 60 \times \frac{V[m/s]}{3.6} \tag{6}$$

Já a frequência da vazão é dada por

$$F_{vazão} = FL[L/s] \times P_{FL} \tag{7}$$

onde FL representa a vazão e P_{FL} representa o fator de pulsos por unidade de vazão.

Porém, a vazão é dada por L/min, então é necessário realizar a seguinte alteração

$$F_{vazão} = \frac{FL[L/s] \times P_{FL}}{60} \tag{8}$$

o valor de P_{FL} é igual a 1015.

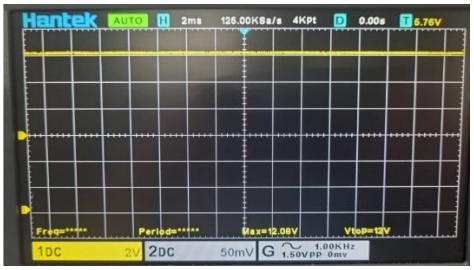
```
void loop() {
  unsigned long currentMillis = millis();

  // Processamento para Speed
  if (currentMillis - previousMillisSpeed >= intervalSpeed) {
    previousMillisSpeed = currentMillis;
    filteredSpeed = applyExponentialFilter(analogRead(SpeedPin),
filteredSpeed, alphaSpeed, 0, 1500);

    SpeedEquation();

    // Print SPEED
    Serial.print("V: ");
    Serial.print(filteredSpeed);
    Serial.print(" km/h");
    //Serial.print(" SFreq: ");
    //Serial.print(SpeedFrequency);
    //Serial.print(" Hz");
    //Serial.print(" SP: ");
    //Serial.print(SpeedPeriod);
    //Serial.print(" ms    ");

    // Controle do sinal de saída para Speed usando PWM
    if (SpeedFrequency > 0.5) {
      ledcSetup(SpeedChannel, SpeedFrequency, pwmResolution);
// Configura a nova frequência
      ledcWrite(SpeedChannel, HalfDutyCycle);
    } else {
      ledcWrite(SpeedChannel, 0); // Para o sinal PWM se a
frequência for 0
    }
  }

  // Processamento para Flow
  if (currentMillis - previousMillisFlow >= intervalFlow) {
```

```
    previousMillisFlow = currentMillis;
    filteredFlow = applyExponentialFilter(analogRead(FlowPin),
filteredFlow, alphaFlow, 0, 5000);

    FlowEquation();

    // Print FLOW
    Serial.print("               F: ");
    Serial.print(filteredFlow);
    Serial.println(" L/min");
    //Serial.print(" FFreq: ");
    //Serial.print(FlowFrequency);
    //Serial.print(" Hz");
    //Serial.print(" FP: ");
    //Serial.println(FlowPeriod);
    //Serial.print(" ms");

    // Controle do sinal de saída para Flow usando PWM
    if (FlowFrequency > 0.5) {
      ledcSetup(FlowChannel, FlowFrequency, pwmResolution);  //
Configura a nova frequência
      ledcWrite(FlowChannel, HalfDutyCycle );
    } else {
      ledcWrite(FlowChannel, 0); // Para o sinal PWM se a
frequência for 0
    }
  }
}
```

No loop, foi adicionado as equações ledc para configurar as frequências na saída e apresenta os valores na serial para ser possível comparar os valores medidos com os que serão apresentados no LCD.

E como dito na seção 2.3.1.2, quando a frequência é nula, o sinal se mantém em estado lógico alto, demonstrado na Figura 6, com o auxílio do osciloscópio.

Figura 6 - Sinal de saída referente à vazão quando a Vazão de entrada é igual a 0L/min.



Fonte: Desenvolvida pelo autor, 2024

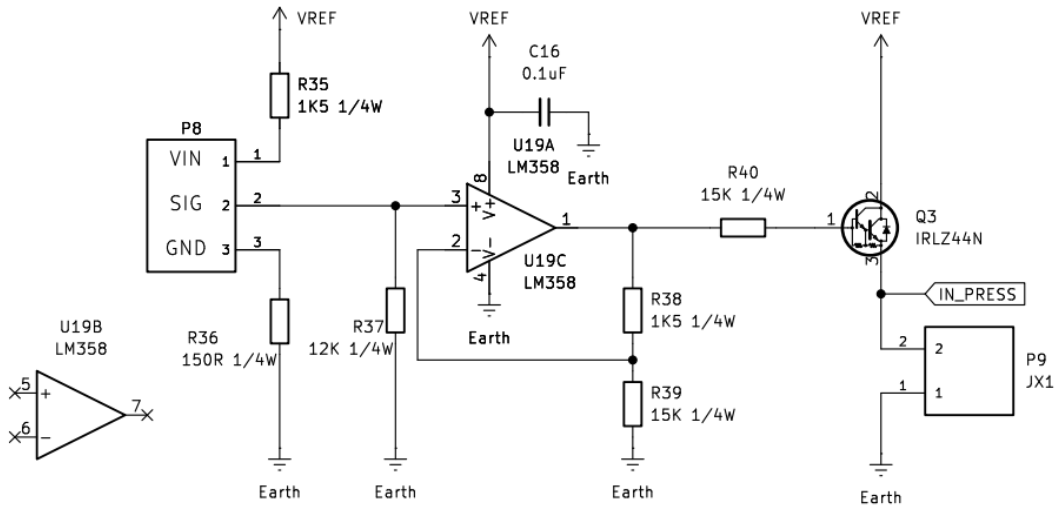
2.4. Gerador de 4 a 20mA

2.4.1. Circuito esquemático

O circuito gerador de 4 a 20mA é necessário para gerar dados equivalente à pressão do sistema de pulverização. Para se medir a pressão, é utilizado um transdutor de pressão de 10Bar conectado à saída. [3, 11]

O circuito é demonstrado na figura 7, onde o transdutor lê a pressão de acordo com a corrente recebida, sendo que a corrente equivalente a 0Bar e 10Bar é igual a 4mA e 20mA respectivamente. Para o sistema que será implementado, só há necessidade de se analisar a pressão de 0 a 10Bar, portanto, o gerador só precisa cobrir esta faixa.

Figura 7 - Circuito gerador de 4 a 20mA.



Fonte: Desenvolvida pelo autor, 2024

O conector de saída (P9) será conectado em um circuito do controlador que possui uma carga de 150Ω, logo toda consideração feita para o desenvolvimento dele, foi em cima desta carga. Ou seja, considerando a corrente desejada, foi necessário desenvolvê-lo para enviar um sinal de saída entre 600mV e 3000mV.

E então utilizando o Multisim e testando o circuito na protoboard, foi possível ver qual o melhor valor para cada resistor.

Os resistores R35 e R36 são posicionados para limitar a faixa de tensão que será emitida pelo potenciômetro, que neste projeto está sendo utilizado com uma resistência de $1K\Omega$. Já o resistor R37 funciona como um resistor de pulldown para evitar pontos flutuantes na entrada do amplificador.

O resistor R38 age como um limitador para a realimentação do amplificador, enquanto o R39 serve como resistor de pulldown. Já o resistor R40 limita a tensão que entra no Transistor $(IRLZ44N)$[10].

A faixa de operação do circuito em si, não foi limitada a exatamente de 600mV a 3000mV, que seria o caso ideal. A sua operação está de 0 a 3300mV, porém, é possível corrigir esta defasagem ao desenvolver o código de automação.

2.4.2. Análise do comportamento de tensão

Para que seja possível determinar o comportamento da tensão, é necessário analisar suas medidas de acordo com o valor analógico do potenciômetro.

```
#define PressPin 34
const float alpha = 0.01;
float emaValue = 0;
int RLoad = 150;
float Voltage = 0;
float Current = 0;
float Press = 0;
void setup() {
  Serial.begin(115200);
  pinMode(PressPin, INPUT);
  emaValue = analogRead(PressPin);
}
void loop() {
  // Aplicando filtro exponencial
  emaValue = applyExponentialFilter(analogRead(PressPin),
emaValue, alpha, 0, 409500);
  Serial.print(4095);
  Serial.print(",");
  Serial.print(0);
  Serial.print(",");
  Serial.println(emaValue);
}
float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 100.0;
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}
```

Este código foi desenvolvido para filtrar a leitura de pressão, para obter grande estabilidade, para que assim, a medição seja mais precisa e exata. Para obter uma curva sem flutuações constantes, foi-se adicionado 2 faixas limitadores, referentes ao valor máximo e mínimo do valor analógico, ou seja, 4095 e 0, respectivamente.

Deste modo, utilizando com o auxílio do multímetro, foi realizado a medição de tensão na saída do circuito, e a cada 50mV de diferente, foi aferido qual o valor analógico correspondente no potenciômetro. Assim, obteve-se os dados referentes a Tabela 1.

Tabela 1 - Medidas Valor analógico x Tensão[mV]

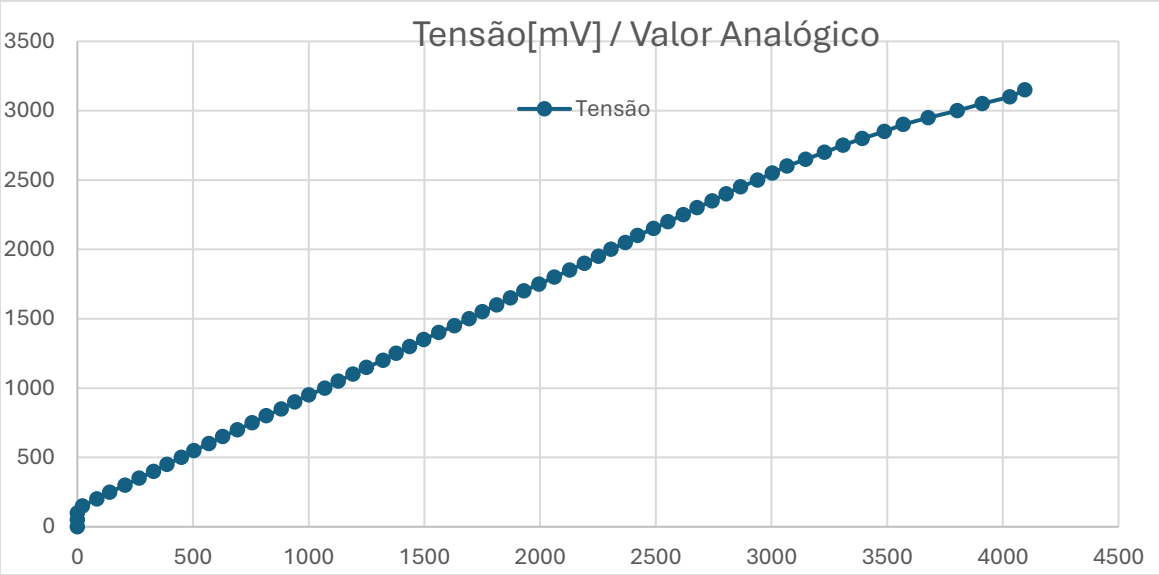
V. ANALÓGICO	TENSÃO	V. ANALÓGICO	TENSÃO
0	0	**1872**	1650
0	50	**1930**	1700
0	100	**1995**	1750
23	150	**2061**	1800
84	200	**2128**	1850
140	250	**2191**	1900
207	300	**2252**	1950
267	350	**2307**	2000
329	400	**2368**	2050
387	450	**2421**	2100
450	500	**2491**	2150
504	550	**2553**	2200
569	600	**2619**	2250
628	650	**2679**	2300
692	700	**2744**	2350
755	750	**2805**	2400
817	800	**2867**	2450
882	850	**2940**	2500
939	900	**3003**	2550
1000	950	**3067**	2600
1069	1000	**3148**	2650
1128	1050	**3229**	2700

1191	1100	**3309**	2750
1250	1150	**3393**	2800
1321	1200	**3488**	2850
1378	1250	**3570**	2900
1436	1300	**3678**	2950
1497	1350	**3803**	3000
1562	1400	**3911**	3050
1630	1450	**4030**	3100
1694	1500		
1750	1550		
1813	1600		

Fonte: Desenvolvida pelo autor, 2024

Na Figura 8 é possível observar o comportamento da curva gerada através dos valores da tabela 1.

Figura 8 - Gráfico tensão/Valor analógico, variando o valor analógico de 0 a 4095.

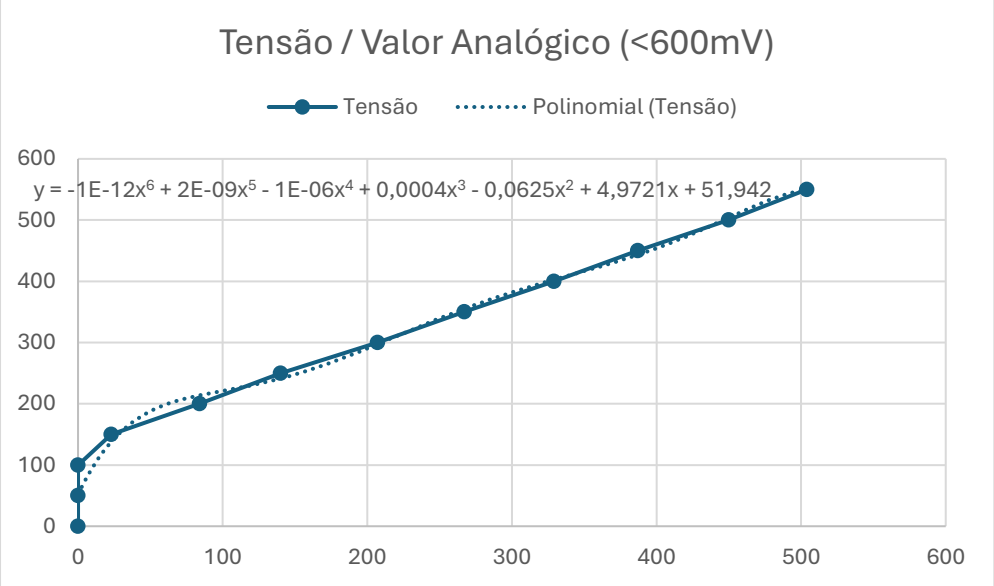


Fonte: Desenvolvida pelo autor, 2024

Ao obter este resultado, é possível perceber que dentro da faixa que será utilizada (600 a 3000mV), a operação é linear, sem muitas variações. Porém, para obter o melhor resultado, é necessário separar a faixa de operação desejada, e obter as curvas para cada caso.

Primeiramente, definimos um novo gráfico com o valor analógico variando de 0 a 569, já que pela tabela, é visto que corresponde a 600mV, obtendo assim, o gráfico apresentado na Figura 9.

Figura 9 - Gráfico tensão/Valor analógico, variando o a tensão de 0 a 600mV.



Fonte: Desenvolvida pelo autor, 2024

Analisando-o separadamente, é possível notar a curva quando seu valor analógico é próximo a 0, e assim, com o intuito de aproximar este resultado, foi criada uma equação polinomial de grau 6. O grau da equação polinomial foi definida após determinar as equações de grau 3 a 9 e analisar qual o melhor RMSE obtido dentre elas.

Considerando $Valor\ Analógico = x$, a tensão é definida para esta faixa como

$$V = A\,x^6 + B\,x^5 - C\,x^4 + D\,x^3 - E\,x^2 + F\,\mathrm{x} + \mathrm{G} \tag{9}$$

onde os coeficientes da equação 9 são demonstrados na Tabela 2.

Tabela 2 - Coeficientes da equação 9: elaborada pelo autor

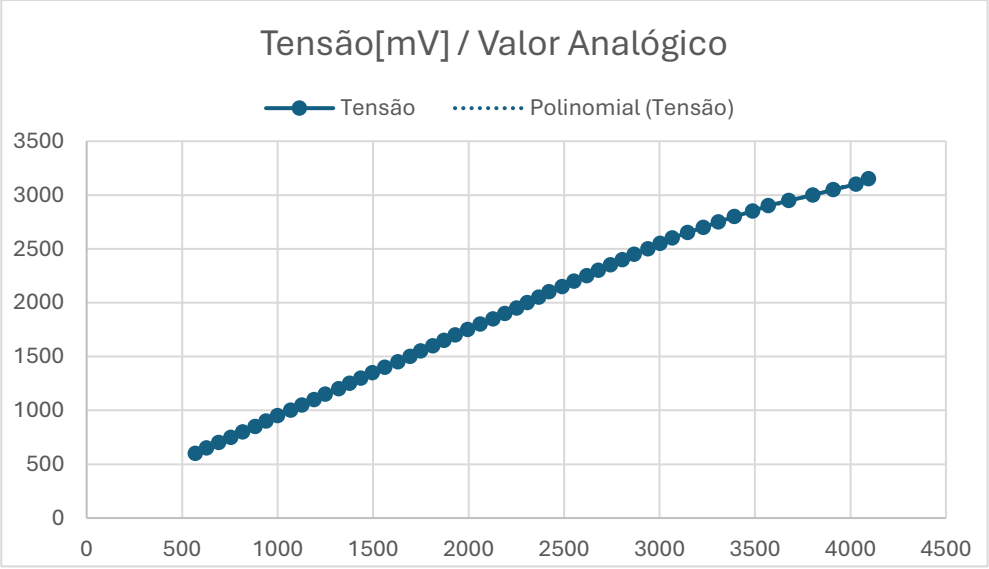
A	-10^{-12}
B	2×10^{-9}
C	10^{-6}
D	0,0004
E	0,0625
F	4,9721
G	51,942

Fonte: Desenvolvida pelo autor, 2024

Isto foi realizado para aferir a tensão mais precisamente e conseguir visualizar no leitor serial do software, a fim de facilitar a leitura, sem a necessidade do uso do multímetro.

Por fim, avaliando a faixa a partir de 600mV, a curva se mantém linear até uma tensão igual a 3000mV, porém após este ponto, sofre variações, demonstrado na Figura 10.

Figura 10 - Gráfico tensão/Valor analógico, onde a tensão varia de 600mV até seu valor máximo.



Fonte: Desenvolvida pelo autor, 2024

Neste caso, a equação polinomial utilizada foi de grau 9, e esta equação obteve um valor tão preciso que não é possível vê-la no gráfico pois há sobreposição entre as duas curvas. A equação da curva foi determinada pela equação 10

$$V = A\,x^9 - B\,x^8 + C\,x^7 - D\,x^6 + E\,x^5 - F\,x^4 + G\,x^3 - H\,x^2 + I\,x - J \qquad (10)$$

onde os coeficientes da equação 10 são demonstrados na Tabela 3.

Tabela 3 - Coeficientes da equação 10: elaborada pelo autor

A	$1{,}5 \times 10^{-27}$
B	$3{,}11 \times 10^{-23}$
C	$2{,}79 \times 10^{-19}$
D	$1{,}41 \times 10^{-15}$
E	$4{,}41 \times 10^{-12}$
F	$8{,}75 \times 10^{-9}$
G	$1{,}10 \times 10^{-5}$
H	$8{,}30 \times 10^{-3}$
I	$4{,}24$
J	$446{,}05$

Fonte: Desenvolvida pelo autor, 2024

Por fim, ao obter todos estes dados, é possível implementar o comportamento da tensão no código.

2.4.3. Software de Automação

Utilizando o mesmo filtro para estabilizar a leitura analógica, foi adicionado a condição para cada equação apresentada nos gráficos anteriormente e foram definidas as equações para converter a tensão em corrente e em medida de pressão.

Como dito previamente, a corrente foi dada como

$$I = \frac{V}{R_{carga}} \qquad (11)$$

sendo que R_{carga} é a resistência da carga do controlador de pulverização, que tem valor igual a 150Ω.

E a pressão, sabendo que 0Bar e 10Bar correspondem a 4mA e 20mA respectivamente, a variação de corrente por unidade de medida de pressão é igual a 1.6mA.

Desse modo, a equação para o cálculo da pressão equivalente é

$$P[Bar] = \frac{I - I_{mín}}{1.6} \tag{17}$$

A qual P, I e I_{min} correspondem a pressão, corrente desejada e corrente mínima, respectivamente. Assim, é possível observar no leitor serial, o valor analógico, a tensão, a corrente e a pressão no circuito.

```
#define PressPin 34
const float alpha = 0.01;
float emaValue = 0;
int RLoad = 150;
float Voltage = 0;
float Current = 0;
float Press = 0;
void setup() {
  Serial.begin(115200);
  pinMode(PressPin, INPUT);
  emaValue = analogRead(PressPin);
}
void loop() {
  // Aplicando filtro exponencial
  emaValue = applyExponentialFilter(analogRead(PressPin),
emaValue, alpha, 0, 409500);
  double term1, term2, term3, term4, term5, term6, term7, term8,
term9, term10;

  if (emaValue < 569) {
```

```
    term1 = 0;
    term2 = 0;
    term3 = 0;
    term4 = -1e-12 * pow(emaValue, 6);
    term5 = 2e-9 * pow(emaValue, 5);
    term6 = -1e-6 * pow(emaValue, 4);
    term7 = 0.0004 * pow(emaValue, 3);
    term8 = -0.0625 * pow(emaValue, 2);
    term9 = 4.9721 * emaValue;
    term10 = 51.942;
    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
term7 + term8 + term9 + term10;
  } else if (emaValue >= 569 && emaValue <= 4095) {
    term1 = 1.50252771e-27 * pow(emaValue, 9);
    term2 = -3.11119751e-23 * pow(emaValue, 8);
    term3 = 2.78904543e-19 * pow(emaValue, 7);
    term4 = -1.41158374e-15 * pow(emaValue, 6);
    term5 = 4.41013361e-12 * pow(emaValue, 5);
    term6 = -8.75245621e-09 * pow(emaValue, 4);
    term7 = 1.09593789e-05 * pow(emaValue, 3);
    term8 = -8.30095519e-03 * pow(emaValue, 2);
    term9 = 4.23846656 * emaValue;
    term10 = -446.05;
    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
term7 + term8 + term9 + term10;
  }
  Current = Voltage / RLoad;
  Press = (Current - 4) / 1.6;
  if (Press < 0) {
    Press = 0;
  }
  Serial.print("emaValue: ");
  Serial.print(emaValue);
  Serial.print("    Voltage: ");
  Serial.print(Voltage);
  Serial.print(" mV");
  Serial.print("    Current: ");
```

```
  Serial.print(Current);
  Serial.print(" mA");
  Serial.print("    Press: ");
  Serial.println(Press);

}
float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 100.0;
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}
```

2.5. *Reguladora e Leitor de Corte de seções*

2.5.1. *Circuito esquemático*

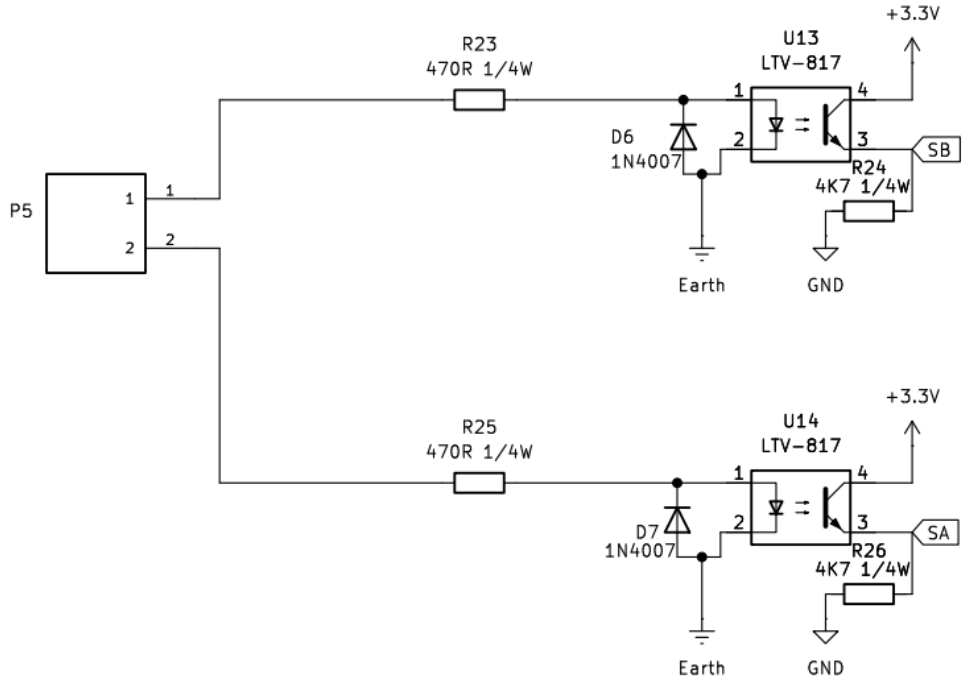
2.5.1.1. Reguladora

A reguladora tem como função verificar a necessidade de aumentar ou diminuir a vazão. Isto é necessário pois, quando a pulverização é realizada, é preciso que a quantidade correta de produtos químicos seja despejada na plantação. E para que isso ocorra de maneira desejada, a velocidade da máquina deve ser precisa conforme o cálculo estabelecido previamente à execução. Caso a velocidade aumente, a área será coberta com menos produtos que o necessário, e para isso, é necessário aumentar a vazão; para aumentar a vazão, é necessário aumentar a pressão no sistema, e o contrário também é verdadeiro.

Por conta disso, para acusar esta variação, através de um chicote elétrico de duas vias, ela emite um sinal de 12V para o controlador caso seja necessário realizar uma ação e não emite sinal caso não seja necessário realizar ação alguma.

Para mudar a ação que deve ser realizada, no caso, elevar ou reduzir a vazão, ela inverte a polaridade do sinal. Na Figura 11, é possível observar o circuito desenvolvido para controlar a reguladora, onde P5 é um conector de 2 pinos.

Figura 11 - Circuito regulador.



Fonte: Desenvolvido pelo autor, 2024

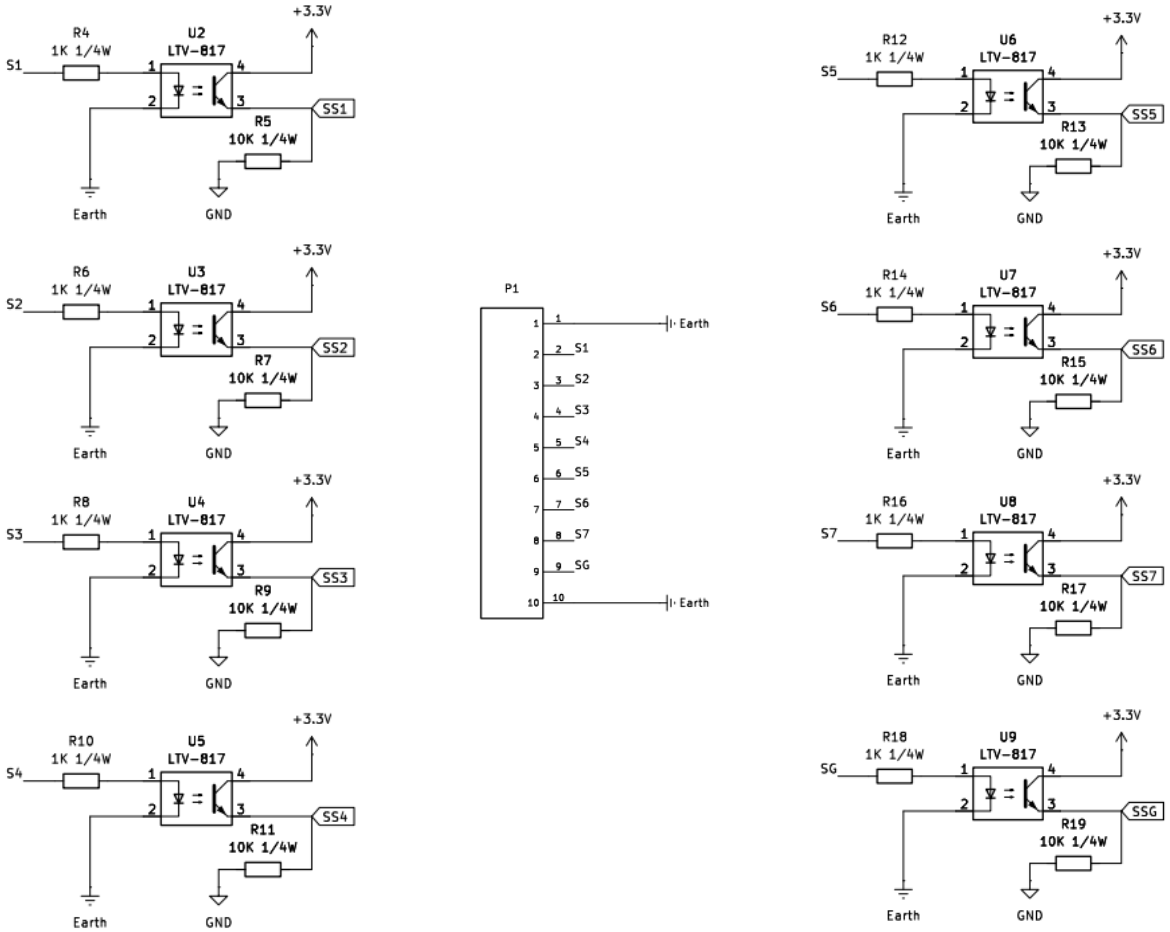
Este circuito envia um sinal de 12V que ativará o LED interno do opto acoplador, conduzindo um sinal de 3,3V para o módulo ESP32 através de SA ou SB, dependendo da polaridade.

Este é um circuito que fornece um isolamento galvânico, com uma diferença, que é a necessidade de um diodo realizando o barramento entre a entrada e a saída do LED interno. Isso ocorre, pois como a tensão percorre por ambas as portas do conector, não há um aterramento específico, e quando não há tensão no circuito específico, ele gera flutuação. Logo, o diodo é utilizado para reparar este problema.

2.5.1.2. Leitor de corte de seções

O circuito de corte de seções, apresentado na Figura 12, é necessário para verificar se as seções estão conectadas ou não, tal que, as seções são conjuntos de bicos, separados por setorização. O motivo disso, é que para economizar com os produtos químicos, o controlador desliga seções ao sobrepor uma área que fora pulverizada, para não pulverizar o mesmo local mais de uma vez, evitando desperdício.

Figura 12 - Circuito de leitura do controlador de corte de seções.



Fonte: Desenvolvido pelo autor, 2024

Para desenvolver o leitor de corte de seções, só foi preciso inserir um isolamento galvânico através de um opto acoplador ($LTV - 817$)[9], onde sempre que o circuito estiver conectado, o módulo irá receber o sinal e indicar que a conexão existe.

O resistor na entrada do LED interno serve para ajustar a corrente e o resistor conectado ao GND age como pulldown.

```
const int sectionPins[] = {27, 23, 19, 18, 17, 16, 4, 26, 25,
33};
String sectionStates[10];
String sectionLabels[] = {"S1", "S2", "S3", "S4", "S5", "S6",
"S7", "SG", "SA", "SB"};
```

```
String Make;
void setup() {
  Serial.begin(115200);
 for (int i = 0; i < 10; i++) {
    pinMode(sectionPins[i], INPUT);
  }
}
void loop() {
  // Controle de sessão e leitura adicional
  for (int i = 0; i < 8; i++) {
    sectionStates[i] = (digitalRead(sectionPins[i]) == LOW) ?
"0" : "1";
  }
  // Controle adicional de SA e SB
  bool SSA = digitalRead(sectionPins[8]);
  bool SSB = digitalRead(sectionPins[9]);
  if (SSA && !SSB) {
    Make = "Eleva";
  } else if (!SSA && SSB) {
    Make = "Reduz";
  } else {
    Make = "Ideal";
  }
  String output;
    for (int i = 0; i < 8; i++) {
     output += "  " + sectionLabels[i] + ": " +
sectionStates[i];
     }
    output += "  SA: " + String(SSA) + "  SB: " + String(SSB) +
"  Action: " + Make;
    Serial.println(output);
```

2.5.2. Explicação do código

Para ser possível aplicar as funções necessárias para o configurar o LCD I2C e os pulsos PWM, é necessário adicionar as seguintes bibliotecas

```
#include <LiquidCrystal_I2C.h>
#include <Arduino.h>
```

Assim, é possível iniciar o LCD, definindo o seu endereço, número de linhas e colunas. Para alterar o endereço, é necessário modificar o hardware do LCD, onde contém 3 pinos, A0, A1 e A2, alterando o nível lógico de cada. Portanto, foi utilizado um lcd 20x4 com endereçamento 0x27 para o projeto.

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

Para iniciar o código, é necessário definir as portas utilizadas para cada função

```
// Pinos utilizados
#define SpeedPin  36
#define SpeedPWM  14
#define ConnSpeed 35

#define FlowPin   39
#define FlowPWM   13
#define ConnFlow  32

#define PressPin  34

#define LedPin    5
```

Onde a reguladora e cada representação de seção é definida através de um arranjo

```
const int sectionPins[] = {27, 23, 19, 18, 17, 16, 4, 26, 25,
33};
String sectionStates[10];
String sectionLabels[] = {"S1", "S2", "S3", "S4", "S5", "S6",
"S7", "SG", "SA", "SB"};
```

```
String Make;

// Variáveis Conectores
String ConnS;
String ConnF;
```

Posteriormente, é necessário definir as variáveis e constantes que vão ser utilizadas e seus valores iniciais.

```
// Variáveis para Pressão
const float alpha = 0.15;
float emaValue = 0;
int RLoad = 150;
float Voltage = 0;
float Current = 0;
float Press = 0;

// Variáveis para Velocidade
float filteredSpeed = 0;
const float alphaSpeed = 0.15;
unsigned long previousMillisSpeed = 0;
const unsigned long intervalSpeed = 100;

// Variáveis para PWM da Velocidade
float SpeedFrequency = 0;
float SpeedPeriod = 0;

// Variáveis para Vazão
float filteredFlow = 0;
const float alphaFlow = 0.15;
unsigned long previousMillisFlow = 0;
const unsigned long intervalFlow = 100;

// Variáveis para PWM da Vazão
float FlowFrequency = 0;
float FlowPeriod = 0;
```

Para configurar os canais para gerar o pulso de velocidade e vazão, é necessário definir a resolução, o ciclo de trabalho e qual canal vai ser utilizado. Os canais podem variar de 0 a 15, onde todos os canais possuem mesma propriedade, a resolução é definida de acordo com o módulo utilizado e é necessário consultar no datasheet, no caso do ESP32 DEVKIT V1, a resolução é de 8 bits. Já o ciclo de trabalho é definido de acordo com o período da onda. Como a onda ideal para este caso deve estar metade do período em nível lógico alto, assume-se um *duty cycle* igual a 50%. Ou seja, como 8 bits, possui 256 valores, variando de 0 a 255, metade é referente a 128 valores, variando de 0 a 127. Portando, o ciclo utilizado é igual a 127.

```
// Configurações do canal PWM
const int SpeedChannel = 1;
const int FlowChannel = 2;
const int pwmResolution = 8;  // Resolução de 8 bits
const int HalfDutyCycle = 127; // Ciclo de de 50% para 8 bits
```

O setup deve conter a inicialização serial, que no ESP32 é definido por 115200, diferentemente do Arduino, que se utiliza 9600, e a inicialização do LCD e dos canais PWM. Além disso, se define se as portas serão entradas ou saídas.

A definição da leitura do potenciômetro referente à pressão foi realizada dentro do setup, todavia não seria mandatório.

```
void setup() {
  Serial.begin(115200);

  pinMode(SpeedPin, INPUT);
  pinMode(ConnSpeed, INPUT);
  pinMode(FlowPin, INPUT);
  pinMode(ConnFlow, INPUT);

  pinMode(PressPin, INPUT);
  emaValue = analogRead(PressPin);

  pinMode(LedPin, OUTPUT);
```

```
  digitalWrite(LedPin, HIGH);

  for (int i = 0; i < 10; i++) {
    pinMode(sectionPins[i], INPUT);
  }

  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 1);
  lcd.print("     CONTROLLER     ");
  lcd.setCursor(0, 2);
  lcd.print("     SIMULATOR!     ");
  delay(3000); // Aguarda 3 segundos
  lcd.clear(); // Limpa a tela para exibir as próximas
informações

  // Configuração dos canais PWM para os dois sinais
  ledcSetup(SpeedChannel, 0, pwmResolution);  // Inicialização
do canal de velocidade
  ledcAttachPin(SpeedPWM, SpeedChannel);

  ledcSetup(FlowChannel, 0, pwmResolution);  // Inicialização do
canal de vazão
  ledcAttachPin(FlowPWM, FlowChannel);
}
```

Em seguida, foi especificada as funções utilizadas para estabelece o filtro de leitura do sinal analógico e a definição das frequências equivalentes à velocidade e vazão. Foi determinada também, a função que mostrará as informações na tela LCD.

```
float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 10.0; // Escala para 0 a 15.00
```

```
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}

void SpeedEquation() {
  SpeedFrequency = 60 * filteredSpeed / 3.6;
  SpeedPeriod = (SpeedFrequency > 0) ? (1000.0 / SpeedFrequency)
: 0;


  if (filteredSpeed <= 0.01) {
    SpeedFrequency = 0;
    SpeedPeriod = 0;
  }
}

void FlowEquation() {
  FlowFrequency = 1015 * filteredFlow / 60;
  FlowPeriod = (FlowFrequency > 0) ? (1000.0 / FlowFrequency) :
0;

  if (filteredFlow <= 0.01) {
    FlowFrequency = 0;
    FlowPeriod = 0;
  }
}
```

Os dados apresentados na tela foram definidos como variáveis do tipo $char$, onde representavam cada linha do LCD e seu respectivo tamanho. E deste modo, usando a função $snprintf$, a string é armazenada e formatada em um buffer de forma segura, evitando overflow de memória. Assim, definimos qual a string utilizada, o seu tamanho e a informação contida.

Finalmente, o LCD define a posição de cada linha e as apresenta.

```
Void tela() {
  char line0[20], line1[20], line2[20], line3[20];
```

```
  snprintf(line0, sizeof(line0), " R:%s   SPD:%.1f", Make,
filteredSpeed);
  snprintf(line1, sizeof(line1), " FLW:%.1f   PRS:%.1f",
filteredFlow, Press);
  snprintf(line2, sizeof(line2), "G 1 2 3 4 5 6 7 S F ");
  snprintf(line3, sizeof(line3), "%s %s %s %s %s %s %s %s %s
%s",
           sectionStates[7].c_str(), sectionStates[0].c_str(),
sectionStates[1].c_str(),
           sectionStates[2].c_str(), sectionStates[3].c_str(),
sectionStates[4].c_str(),
           sectionStates[5].c_str(), sectionStates[6].c_str(),
ConnS.c_str(), ConnF.c_str());

  lcd.setCursor(0, 0);
  lcd.print(line0);
  lcd.setCursor(0, 1);
  lcd.print(line1);
  lcd.setCursor(0, 2);
  lcd.print(line2);
  lcd.setCursor(0, 3);
  lcd.print(line3);
}
```

Por fim, é preciso definir o loop executável, que atualiza ininterruptamente, realizando todos os processos.

```
void loop() {
  unsigned long currentMillis = millis();
```

Para definir as seções, utiliza o arranjo criado anteriormente e define-se que a leitura digital é baixa. Caso for verdadeira a condição, será retornado 0, caso contrário, 1.

```
   for (int i = 0; i < 8; i++) {
    sectionStates[i] = (digitalRead(sectionPins[i]) == LOW) ?
"0" : "1";
  }
```

A leitura da reguladora será condicional, onde ao realizar a leitura digital, estabelecerá as condições para elevar, reduzir ou manter a vazão no sistema de pulverização.

```
  // Controle adicional de SA e SB
  bool SSA = digitalRead(sectionPins[8]);
  bool SSB = digitalRead(sectionPins[9]);

  if (SSA && !SSB) {
    Make = "Eleva";
  } else if (!SSA && SSB) {
    Make = "Reduz";
  } else {
    Make = "Ideal";
  }
```

O processamento da velocidade e da vazão são realizadas igualmente, alterando somente seus valores máximos e a equação utilizada para frequência, que fora determinada anteriormente no código.

Outra adaptação que ocorreu no código, foi a alteração na função `ledcWrite`, pois como há inversão do nível lógico na saída do pulso PWM, quando a frequência for igual a 0Hz, o módulo a manterá em nível lógico alto, para que o controlador receba um sinal em nível lógico baixo.

```
  // Processamento para Speed
  if (currentMillis - previousMillisSpeed >= intervalSpeed) {
    previousMillisSpeed = currentMillis;
    filteredSpeed = applyExponentialFilter(analogRead(SpeedPin),
filteredSpeed, alphaSpeed, 0, 150);

    SpeedEquation();

    // Controle do sinal de saída para Speed usando PWM
    if (SpeedFrequency > 0.1) {
      ledcSetup(SpeedChannel, SpeedFrequency, pwmResolution);
// Configura a nova frequência
```

```
      ledcWrite(SpeedChannel, HalfDutyCycle);
    } else {
      ledcWrite(SpeedChannel, 255); // Para o sinal PWM se a
frequência for 0
    }
  }

  // Processamento para Flow
  if (currentMillis - previousMillisFlow >= intervalFlow) {
    previousMillisFlow = currentMillis;
    filteredFlow = applyExponentialFilter(analogRead(FlowPin),
filteredFlow, alphaFlow, 0, 500);

    FlowEquation();

    // Controle do sinal de saída para Flow usando PWM
    if (FlowFrequency > 0.1) {
      ledcSetup(FlowChannel, FlowFrequency, pwmResolution);  //
Configura a nova frequência
      ledcWrite(FlowChannel, HalfDutyCycle);
    } else {
      ledcWrite(FlowChannel, 255); // Para o sinal PWM se a
frequência for 0
    }
  }
```

Para definir se há a conexão dos conectores que farão a leitura da velocidade e da vazão, foi inserido uma leitura analógica com um ponto limiar arbitrário igual a 512, que corresponde a aproximadamente 12,5% do valor total. Este valor foi baseado em um teste empírico, que apresentou flutuações quando a condição não era especificada.

Essa flutuação não apresenta problema para o sistema, já que se trata de uma tensão DC.

```
  ConnS = (analogRead(ConnSpeed) > 512) ? "1" : "0";
 ConnF = (analogRead(ConnFlow) > 512) ? "1" : "0";
```

Enfim, após aplicar o filtro na leitura analógica reminiscente à pressão as equações pré-determinadas anteriormente são aplicadas de acordo com o valor analógico definido através das medições realizadas pelo teste empírico utilizando o multímetro.

```
   emaValue = applyExponentialFilter(analogRead(PressPin), 
emaValue, alpha, 0, 40950);

  double term1, term2, term3, term4, term5, term6, term7, term8, 
term9, term10;

  if (emaValue < 569) {


    emaValue = applyExponentialFilter(analogRead(PressPin), 
emaValue, alpha, 0, 40950);

  double term1, term2, term3, term4, term5, term6, term7, term8, 
term9, term10;

  if (emaValue < 569) {

    Voltage = 0;
  } else if (emaValue >= 569 && emaValue <= 4095) {
    term1 = 1.50252771e-27 * pow(emaValue, 9);
    term2 = -3.11119751e-23 * pow(emaValue, 8);
    term3 = 2.78904543e-19 * pow(emaValue, 7);
    term4 = -1.41158374e-15 * pow(emaValue, 6);
    term5 = 4.41013361e-12 * pow(emaValue, 5);
    term6 = -8.75245621e-09 * pow(emaValue, 4);
    term7 = 1.09593789e-05 * pow(emaValue, 3);
    term8 = -8.30095519e-03 * pow(emaValue, 2);
    term9 = 4.23846656 * emaValue;
    term10 = -446.05;

    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
```

```
term7 + term8 + term9 + term10;
  }

  Current = Voltage / RLoad;
  Press = (Current - 4) / 1.6;

  if (Press < 0) {
    Press = 0;
  }

  tela();
}


  } else if (emaValue >= 569 && emaValue <= 4095) {
    term1 = 1.50252771e-27 * pow(emaValue, 9);
    term2 = -3.11119751e-23 * pow(emaValue, 8);
    term3 = 2.78904543e-19 * pow(emaValue, 7);
    term4 = -1.41158374e-15 * pow(emaValue, 6);
    term5 = 4.41013361e-12 * pow(emaValue, 5);
    term6 = -8.75245621e-09 * pow(emaValue, 4);
    term7 = 1.09593789e-05 * pow(emaValue, 3);
    term8 = -8.30095519e-03 * pow(emaValue, 2);
    term9 = 4.23846656 * emaValue;
    term10 = -446.05;

    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
term7 + term8 + term9 + term10;
  }

  Current = Voltage / RLoad;
  Press = (Current - 4) / 1.6;

  if (Press < 0) {
    Press = 0;
  }
```

```
  tela();
}
```

3. Resultados

O código completo, é referente ao código utilizado na compilação final do sistema, constituindo o funcionamento de todos os circuitos do sistema, diferentemente dos códigos anteriores, que eram códigos isolados para comprovar o funcionamento de cada circuito.

Para desenvolvê-lo, foi preciso adaptar determinadas funções que causavam interferência entre si, como por exemplo a função *millis*. É possível analisa-lo no Apêndice A.

Para apurar os resultados, foi necessário utilizar um multímetro para aferir a tensão de saída correspondente à pressão, um osciloscópio para analisar a saída dos pulsos PWM, uma fonte de tensão para simular os sinais da reguladora, do leitor de corte de seções e do conector dos sinais PWM. Além disso, foi necessário utilizar um resistor de $4{,}7k\Omega$ para ser equivalente ao resistor de pullup e um resistor de 150Ω equivalente à resistência da carga presente no leitor de pressão, ambos contidos no controlador.

3.1. Layout da Tela

O layout da tela foi determinado como mostra a Figura 13, onde na primeira linha, a reguladora mostra se necessita de alguma ação ou se está ideal e qual a velocidade amostrada. Na segunda linha, é demonstrada a vazão e a pressão, enquanto na terceira e quarta linha, verifica-se se os conectores referentes às seções, velocidade e pressão estão conectados.

Figura 13 - Layout Final do LCD.

Fonte: Desenvolvida pelo autor, 2024

3.2. Sinal referente à Velocidade

Para avaliar a velocidade precisamente, é preciso verificar se a velocidade e o pulso PWM na saída são correspondentes. Adicionando um resistor de pullup de $4{,}7k\Omega$ na saída, com o auxílio do osciloscópio, fora analisado o pulso equivalente à velocidade. Como é demonstrado na figura 14.

Figura 14 - Amostragem da velocidade quando o potenciômetro atinge o valor máximo.

Fonte: Desenvolvida pelo autor, 2024

Obtendo a frequência equivalente à velocidade medida, é possível obter o período. Tendo em vista que o valor máximo da velocidade é igual a 15, seu período corresponde a 4ms. Assim, utilizando o osciloscópio, é visível que se obteve o valor esperado, como demonstrado na Figura 15.

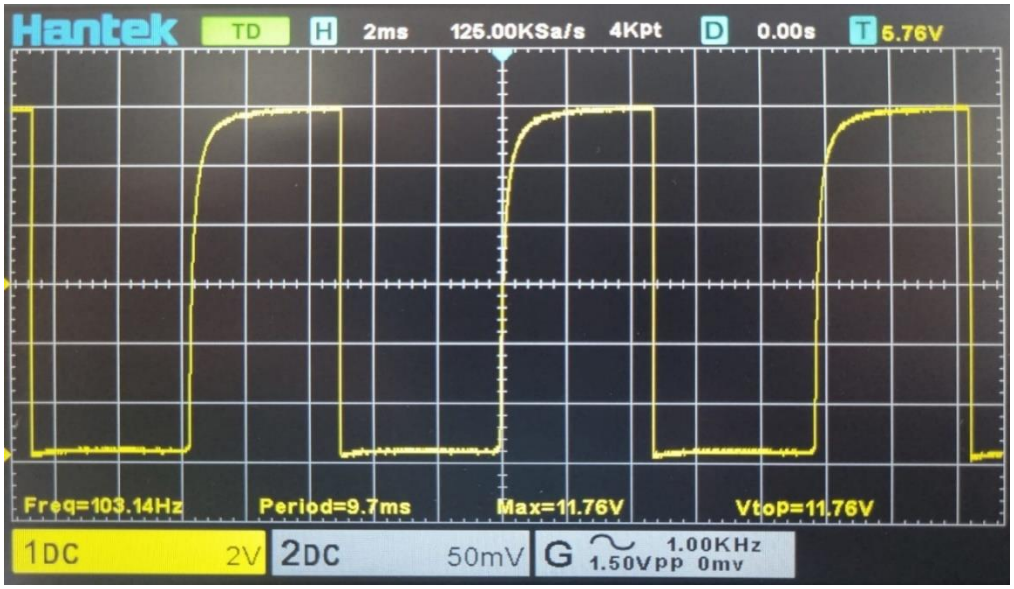
Figura 15 - Frequência referente à velocidade máxima (15km/h).

Fonte: Desenvolvida pelo autor, 2024

Demonstrando assim que o sistema está funcionando corretamente. Porém, para obter uma precisão melhor, é necessário analisar para as outras medidas. Desse modo, usando uma velocidade de valor aleatório para teste, com uma velocidade de 6,24 km/h, o esperado é um período de 9,61ms.

Na Figura 16, podemos perceber que há uma diferença de aproximadamente 0.1ms entre o valor teórico e o valor medido. Isso ocorre pois como foi aplicado um filtro exponencial para estabilizar o sinal, o valor amostrado no LCD é uma média, não mostrando o valor exato.

Figura 16 - Frequência referente à uma velocidade igual a 6,24km/h

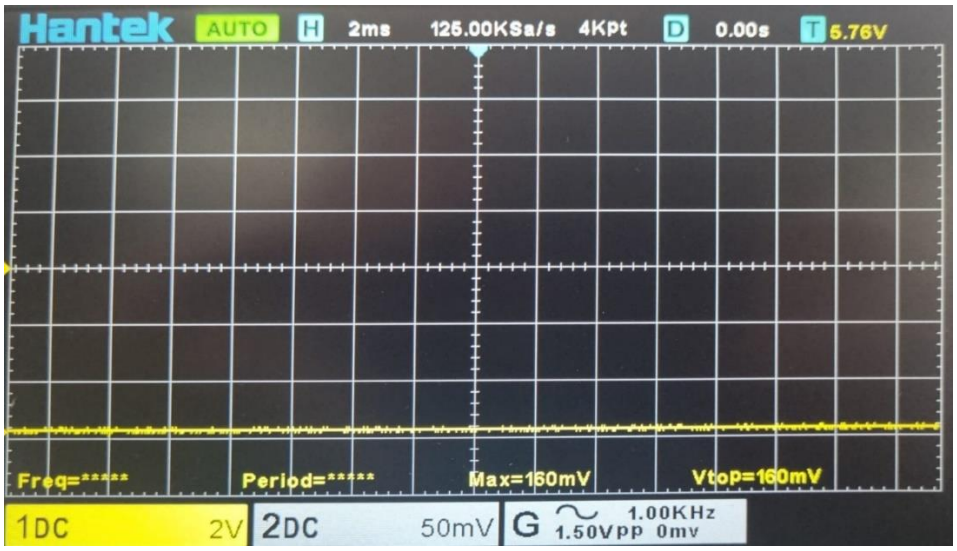


Fonte: Desenvolvida pelo autor, 2024

Porém, considerando o período igual a 9,7ms, a velocidade correspondente seria 6,18km/h, e como essa diferença é mínima, não afeta no comportamento do sistema.

Por fim, quando não ter movimento, ou seja, velocidade igual a 0, com o auxílio do código, foi feita a alteração do sinal de saída, estabelecendo o sinal em nível lógico baixo. Logo, na Figura 17, é visível que o sinal se mantém em nível lógico baixo.

Figura 17 - Sinal reajustado para nível lógico baixo quando a velocidade é igual a 0km/h.



Fonte: Desenvolvida pelo autor, 2024

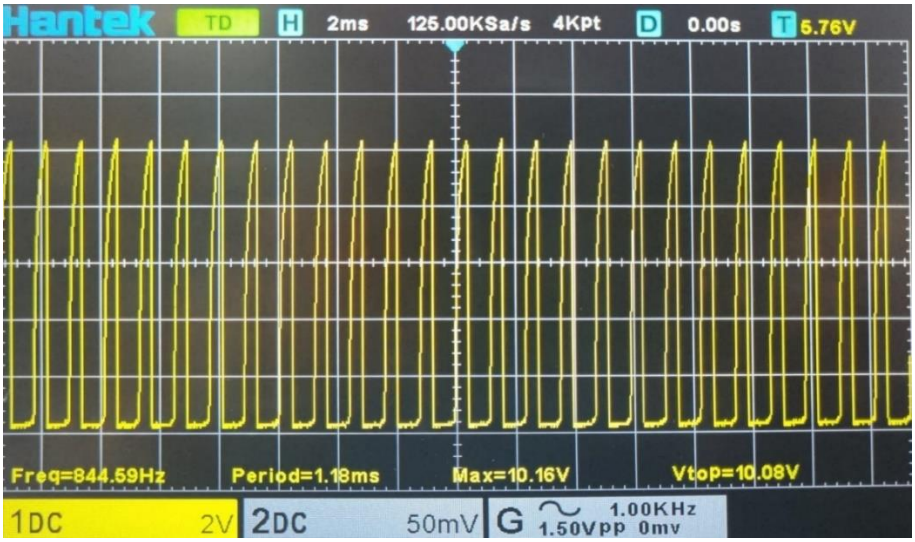
Com base nos resultados apresentados, foi validado que o funcionamento do circuito gerador de medidas de velocidade, assim como o respectivo código, está conforme o esperado.

3.3. Sinal referente à Vazão

Dado que o circuito gerador de medidas referente à vazão é idêntico ao circuito referente à velocidade, é esperado que o seu funcionamento esteja correto. Tendo em vista que as únicas diferenças são o seu valor máximo e o cálculo para obter a frequência.

Utilizando a equação 8, para seu valor máximo $(50L/\min)$, o período é igual a 1,18L/min, assim, com o auxílio do osciloscópio, é demonstrado na Figura 18 que o funcionamento está como o esperado.

Figura 18 - Frequência referente à vazão máxima (50L/min).



Fonte: Desenvolvido pelo autor, 2024

Dado que o circuito apresenta mesmo comportamento referente a velocidade e o código utilizado segue mesmo padrão, e como é observado nos resultados, o sinal de saída está sendo gerado como o esperado, sem variações relevantes para afetar o funcionamento do sistema.

3.4. Sinal referente à Pressão

O circuito gerador de 4 a 20mA, é o que sofreu mais problemas ao ser desenvolvido, já que houve a necessidade ajustar a pressão igual a 0 de acordo com a leitura do potenciômetro, pois o circuito não manteve a tensão mínima em 600mV como era o desejado, e sim, em 0V. Além disso, como o circuito não foi processado pelo módulo, e só teve o seu valor amostrado, foi necessário remapear através de equações para que fosse possível apresentar a pressão referente à corrente na saída do circuito. A consequência disso, é que o valor é aproximado pela equação, logo, a exatidão do valor será menor.

Para aferir seus resultados, uma carga de 150Ω foi adicionada na saída do circuito, e com o auxílio de um multímetro, foi medida a tensão na carga, tornando possível a comparação com a pressão amostrada no LCD e verificando sua exatidão.

Seguindo o mesmo princípio da corrente, utilizando a pressão [Bar] e a tensão [mV], sendo:

$$V = P_{amostrada} \times \left(\frac{V_{máx} - V_{mín}}{P_{máx} - P_{mín}}\right) + V_{mín} \tag{18}$$

onde $P_{amostrada}$, $V_{máx}$, $V_{mín}$, $P_{máx}$ e $P_{mín}$ correspondem à pressão amostrada, tensão máxima, tensão mínima, pressão máxima e pressão mínima respectivamente. Sabendo que os seus valores correspondentes são 3000mV, 600mV, 10Bar e 0bar respectivamente.

A tensão equivalente para uma pressão de 2Bar seria igual a 1080mV, assim, como demonstrado na Figura 19, ao realizar a medição com um multímetro, obteve-se 1025mV.

Figura 19 - Tensão medida quando a pressão amostrada é igual a 2Bar.

Fonte: Desenvolvida pelo autor, 2024

Essa diferença era esperada, pois há uma diferença entre o cálculo e a medida, já que para manter o sinal estável, foi utilizado uma média móvel. Além disso, foi realizada uma medida abaixo e próxima a 600mV no multímetro para analisar a resposta obtida no LCD. Com uma tensão definida em 580mV, a pressão mostrada no LCD continuou em 0, como o esperado, já quando a tensão foi definida em 628mV, a pressão alterou para 0,1Bar, como visto na Figura 20.

Figura 20 - Pressão amostrada referente à uma pressão de 628mV.

Fonte: Desenvolvida pelo autor, 2024

Assim, foi verificado que o circuito está funcionando corretamente, apesar da leve variação apresentada nos valores do LCD, ocasionada pela aproximação da equação.

3.5. Leitura da reguladora

Para realizar a medição na reguladora, foi necessário utilizar uma fonte variável ajustada em 12V. Seu uso foi necessário para simular o sinal que é transmitido pelo controlador e define a ação que deve ser tomada. Mantendo a conexão desligada, é esperado que ela mostre que a vazão é ideal. Nas Figuras 21 – 23, é possível analisar o comportamento da reguladora quando não havia tensão no circuito e quando havia tensão em cada pino. A reguladora, não tem um lado específico, já que dependerá da conexão do controlador, ou seja, dependendo da polaridade e posição do conector do controlador, portanto é necessário fazer uma alteração na ação que deve ser realizada, sendo algo situacional.

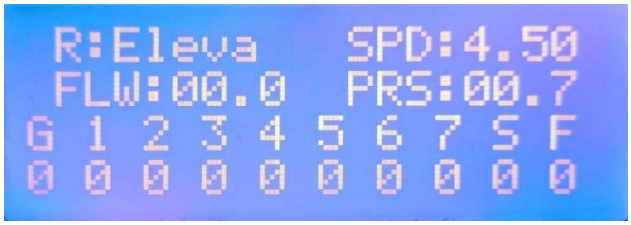
Figura 21 - Reguladora em estado ideal.

Fonte: Desenvolvida pelo autor, 2024

Figura 22 - Reguladora em estado ideal.

Fonte: Desenvolvida pelo autor, 2024

Figura 23 - Reguladora ordenando elevar a vazão

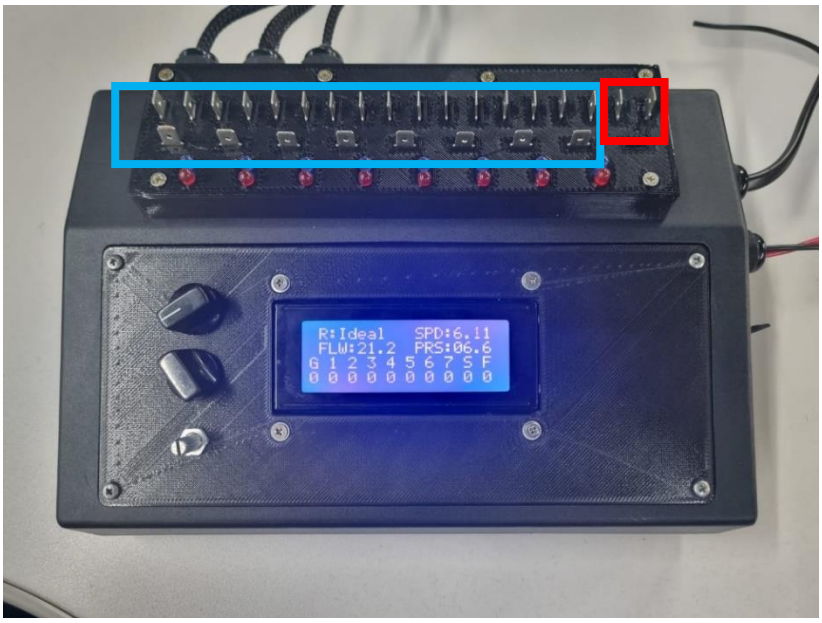


Fonte: Desenvolvida pelo autor, 2024

3.6. Leitura da condição das seções

Para realizar a leitura, a mesma fonte foi utilizada, na mesma configuração. No caso do conector que gerenciará as seções, diferentemente da reguladora, possui 3 pinos. Como visto na Figura 24, os pinos destacados em azul são referentes às seções, já o conector destacado em vermelho é referente à reguladora. Os conectores possuem 3 pinos, para que ele consiga ligar/desligar seções e mostre se o conector está conectado ou não.

Figura 24 - Conectores referentes à reguladora e ao leitor de seções.



Fonte: Desenvolvida pelo autor, 2024

Caso o conector seja conectado, o LED vermelha se acende para indicar que há conexão entre eles. Pois assim, se houver conexão e o LED não acender, sabe-se que o cabo ou o conector está com defeito. Diferente deste pino, o outro ligará e desligará as seções através do controlador, acendendo LED azul somente se o controlador enviar sinal, mesmo que esteja conectado, como demonstrado na Figura 25.

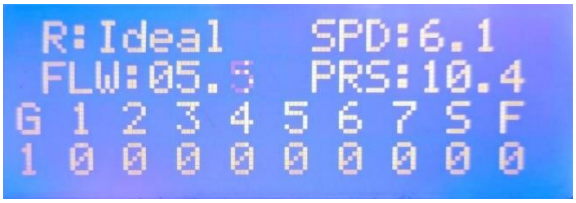
Figura 25 - Simulação da conexão do chicote elétrico, indicando que a seção está ligada.

Fonte: Desenvolvida pelo autor, 2024

Assim, sendo possível analisar se está ativo ou não, através do LED, e não somente pelo LCD, que simultaneamente apresenta qual seção está ligada. Com isso haverá ocorrerá a mudança do estado lógico baixo (0), para o estado lógico alto (1).

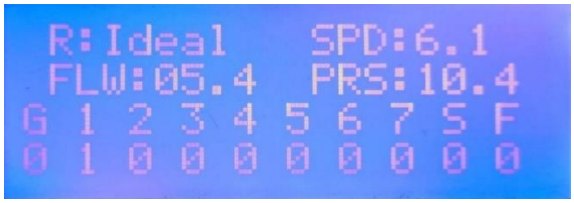
Para realizar esse teste, a conexão foi alternando entre os pinos, iniciando pelo pino da extremidade da esquerda e prosseguindo ao testar seus conectores adjacentes, obtendo o resultado demonstrado nas Figuras 26 – 29.

Figura 26 - Seção geral ativa.



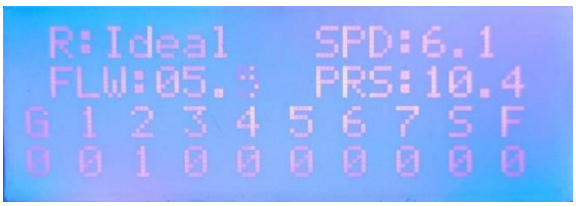
Fonte: Desenvolvida pelo autor, 2024

Figura 27 - Seção 1 ativa.



Fonte: Desenvolvida pelo autor, 2024

Figura 28 - Seção 2 ativa.



Fonte: Desenvolvida pelo autor, 2024

Figura 29 - Seção 3 ativa.

Fonte: Desenvolvida pelo autor, 2024

Como os circuitos de todos os conectores são iguais e apresentam o mesmo funcionamento, não há a necessidade de testar todos os conectores para validar o resultado. Assim, observando os quatro conectores da esquerda, é visível que a mudança de estado lógico está funcionando corretamente, como o esperado.

O mesmo ocorre com os status de velocidade e frequência, que para demonstrar conexão, possuem um circuito semelhante ao circuito de leitura de corte de seções, onde foi validado usando o mesmo método.

4. Conclusão

O objetivo principal do projeto é facilitar e diminuir gastos, possibilitando que o controlador seja testado e configurado em um ambiente de testes, e para que seja possível, as medidas simuladas devem ser precisas.

Ao analisar e obter os resultados de cada circuito, de maneira conjunta para levar em consideração o funcionamento simultâneo dos circuitos, é visto que a fonte reguladora está funcionando corretamente, alimentando o circuito como o necessário. Os circuitos referentes à vazão e velocidade apresentaram exímia precisão, onde a frequência medida apresentava menos de 5Hz de diferença da frequência esperada, sendo irrelevante para a medida.

O gerador de 4 a 20mA, foi o circuito que apresentou mais divergências e "problemas", pois o circuito não foi limitado na faixa de operação desejada. Porém, isso foi corrigido ao realizar uma limitação pelo software. Outra divergência causada por este circuito, foi que a visualização da pressão no LCD não é feita pelo valor de tensão de saída, e apesar do controlador estar fazendo a leitura da corrente precisamente, o módulo realiza a leitura pelo valor analógico do potenciômetro, e como a curva não é linear, apresenta variações por conta da aproximação feita pela equação da curva. Apesar disso, a variação é muito baixa e como a corrente de saída é lida corretamente pelo controlador, ambas as dificuldades não trarão consequências, validando o uso do circuito.

Os demais circuitos estão sujeitos ao mesmo princípio, onde recebem um sinal do controlador para realizar certas ações, e assim, é possível analisar se o controlador está enviando sinais corretos. A reguladora realiza a ação imposta pelo sinal recebido, alterando a ação quando a polaridade é alterada, do mesmo modo em que o leitor de corte de seções e os indicadores de conexão referente à velocidade e pressão mostram corretamente a mudança de estado lógico, assim que o conector envia sinal para o circuito.

Concluindo-se que o projeto, apesar de apresentar características não desejadas, está funcionando perfeitamente, não apresentando problemas de medições. Deste modo, o circuito geral é válido e aplicável na indústria agropecuária, atingindo seu objetivo inicial, podendo substituir a máquina pulverizadora ao realizar testes de controlador.

Referências

[1] CHAIM, A.; PERES, M.; PESSOA, Y. **Métodos para Calibração de Pulverizadores**. [s.l: s.n.]. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/130723/1/2006PL-049.pdf>.

[2] **Rashid, Muhammad H.** *Power Electronics: Circuits, Devices, and Applications*. 4ª ed. Nova York: Pearson, 2013. ISBN: 9780133125900.

[3] **Professor Bairros.** (2019, outubro 30). *Conversor tensão corrente 1 a 5V para 4 a 20 mA* [Vídeo]. YouTube. https://www.youtube.com/watch?v=C2yk-W0px2Q

[4] **ESP-WROOM-32 datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/1179101/ESPRESSIF/ESP-WROOM-32.html>.

[5] **LM358N datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/3073/MOTOROLA/LM358N.html>.

[6] **PCF8574 datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/18212/PHILIPS/PCF8574.html>.

[7] **LM2576T datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/42464/SEMTECH/LM2576T.html>.

[8] **LM1117-5.0 datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/141839/CYSTEKEC/LM1117-5.0.html>.

[9] **LTV817 datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/86744/LITEON/LTV817.html?gad_source=1&gclid=Cj0KCQjwmOm3BhC8ARIsAOSbapVO-8JxXQdmiGge48hemjBIhIYeZrfbuJmfYlY82Xvaqp015ouArvQaArbUEALw_wcB>.

[10] **IRLZ44N datasheet(PDF)**. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/pdf/1010480/ISC/IRLZ44N.html>.

[11] **Sedra, Adel S., & Smith, Kenneth C.** *Microelectronic Circuits*. 7ª ed. Nova York: Oxford University Press, 2014. ISBN: 9780199339136.

[12] **Controlador De Vazão Pulverização 6 Seções Tecnomark**. Disponível em: <https://produto.mercadolivre.com.br/MLB-5041033440-controlador-de-vazo-pulverizaco-6-secoes-tecnomark-_JM>. Acesso em: 18 nov. 2024.

[13] **Controlador de Pulverização T1000 - Inel Agricultura**. Disponível em: <https://inel.ind.br/controlador-de-pulverizacao-t1000-2/>. Acesso em: 18 nov. 2024.

Apêndice A – Código de automação completo

```
#include <LiquidCrystal_I2C.h>
#include <Arduino.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

// Pinos utilizados
#define SpeedPin  36
#define SpeedPWM  14
#define ConnSpeed 35

#define FlowPin   39
#define FlowPWM   13
#define ConnFlow  32

#define PressPin  34

#define LedPin    5

const int sectionPins[] = {27, 23, 19, 18, 17, 16, 4, 26, 25,
33};
String sectionStates[10];
String sectionLabels[] = {"S1", "S2", "S3", "S4", "S5", "S6",
"S7", "SG", "SA", "SB"};
String Make;

// Variáveis Conectores
String ConnS;
String ConnF;

// Variáveis para Velocidade
float filteredSpeed = 0;
const float alphaSpeed = 0.15;
unsigned long previousMillisSpeed = 0;
```

```
const unsigned long intervalSpeed = 100;

// Variáveis para PWM da Velocidade
float SpeedFrequency = 0;
float SpeedPeriod = 0;

// Variáveis para Vazão
float filteredFlow = 0;
const float alphaFlow = 0.15;
unsigned long previousMillisFlow = 0;
const unsigned long intervalFlow = 100;

// Variáveis para PWM da Vazão
float FlowFrequency = 0;
float FlowPeriod = 0;

// Configurações do canal PWM
const int SpeedChannel = 1;
const int FlowChannel = 2;
const int pwmResolution = 8;  // Resolução de 8 bits
const int HalfDutyCycle = 127; // Ciclo de de 50% para 8 bits

// Variáveis para Pressão
const float alpha = 0.15;
float emaValue = 0;
int RLoad = 150;
float Voltage = 0;
float Current = 0;
float Press = 0;

void setup() {
  Serial.begin(115200);

  pinMode(SpeedPin, INPUT);
  pinMode(ConnSpeed, INPUT);
  pinMode(FlowPin, INPUT);
  pinMode(ConnFlow, INPUT);
```

```
  pinMode(PressPin, INPUT);
  emaValue = analogRead(PressPin);

  pinMode(LedPin, OUTPUT);
  digitalWrite(LedPin, HIGH);

  for (int i = 0; i < 10; i++) {
    pinMode(sectionPins[i], INPUT);
  }

  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 1);
  lcd.print("     CONTROLLER     ");
  lcd.setCursor(0, 2);
  lcd.print("     SIMULATOR!     ");
  delay(3000); // Aguarda 3 segundos
  lcd.clear(); // Limpa a tela para exibir as próximas
informações

  // Configuração dos canais PWM para os dois sinais
  ledcSetup(SpeedChannel, 0, pwmResolution);  // Inicialização
do canal de velocidade
  ledcAttachPin(SpeedPWM, SpeedChannel);

  ledcSetup(FlowChannel, 0, pwmResolution);  // Inicialização do
canal de vazão
  ledcAttachPin(FlowPWM, FlowChannel);
}

void loop() {
  unsigned long currentMillis = millis();

  for (int i = 0; i < 8; i++) {
    sectionStates[i] = (digitalRead(sectionPins[i]) == LOW) ?
```

```
"0" : "1";
  }

  // Controle adicional de SA e SB
  bool SSA = digitalRead(sectionPins[8]);
  bool SSB = digitalRead(sectionPins[9]);

  if (SSA && !SSB) {
    Make = "Eleva";
  } else if (!SSA && SSB) {
    Make = "Reduz";
  } else {
    Make = "Ideal";
  }

  // Processamento para Speed
  if (currentMillis - previousMillisSpeed >= intervalSpeed) {
    previousMillisSpeed = currentMillis;
    filteredSpeed = applyExponentialFilter(analogRead(SpeedPin),
filteredSpeed, alphaSpeed, 0, 150);

    SpeedEquation();

    // Controle do sinal de saída para Speed usando PWM
    if (SpeedFrequency > 0.5) {
      ledcSetup(SpeedChannel, SpeedFrequency, pwmResolution);
// Configura a nova frequência
      ledcWrite(SpeedChannel, HalfDutyCycle);
    } else {
      ledcWrite(SpeedChannel, 0); // Para o sinal PWM se a
frequência for 0
    }
  }

  // Processamento para Flow
  if (currentMillis - previousMillisFlow >= intervalFlow) {
    previousMillisFlow = currentMillis;
```

```
    filteredFlow = applyExponentialFilter(analogRead(FlowPin),
filteredFlow, alphaFlow, 0, 500);

    FlowEquation();

    // Controle do sinal de saída para Flow usando PWM
    if (FlowFrequency > 0.5) {
      ledcSetup(FlowChannel, FlowFrequency, pwmResolution);  //
Configura a nova frequência
      ledcWrite(FlowChannel, HalfDutyCycle);
    } else {
      ledcWrite(FlowChannel, 255); // Para o sinal PWM se a
frequência for 0
    }
  }

  ConnS = (analogRead(ConnSpeed) > 512) ? "1" : "0";
  ConnF = (analogRead(ConnFlow) > 512) ? "1" : "0";

  emaValue = applyExponentialFilter(analogRead(PressPin),
emaValue, alpha, 0, 40950);

  double term1, term2, term3, term4, term5, term6, term7, term8,
term9, term10;

  if (emaValue < 569) {
    term1 = 0;
    term2 = 0;
    term3 = 0;
    term4 = -1e-12 * pow(emaValue, 6);
    term5 = 2e-9 * pow(emaValue, 5);
    term6 = -1e-6 * pow(emaValue, 4);
    term7 = 0.0004 * pow(emaValue, 3);
    term8 = -0.0625 * pow(emaValue, 2);
    term9 = 4.9721 * emaValue;
    term10 = 51.942;
```

```
    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
term7 + term8 + term9 + term10;
  } else if (emaValue >= 569 && emaValue <= 4095) {
    term1 = 1.50252771e-27 * pow(emaValue, 9);
    term2 = -3.11119751e-23 * pow(emaValue, 8);
    term3 = 2.78904543e-19 * pow(emaValue, 7);
    term4 = -1.41158374e-15 * pow(emaValue, 6);
    term5 = 4.41013361e-12 * pow(emaValue, 5);
    term6 = -8.75245621e-09 * pow(emaValue, 4);
    term7 = 1.09593789e-05 * pow(emaValue, 3);
    term8 = -8.30095519e-03 * pow(emaValue, 2);
    term9 = 4.23846656 * emaValue;
    term10 = -446.05;

    Voltage = term1 + term2 + term3 + term4 + term5 + term6 +
term7 + term8 + term9 + term10;
  }

  Current = Voltage / RLoad;
  Press = (Current - 4) / 1.6;

  if (Press < 0) {
    Press = 0;
  }

  tela();
}

float applyExponentialFilter(int rawValue, float filteredValue,
float alpha, int minValue, int maxValue) {
  float scaledValue = map(rawValue, 0, 4095, minValue, maxValue)
/ 10.0; // Escala para 0 a 15.00
  return (alpha * scaledValue) + ((1 - alpha) * filteredValue);
}

void SpeedEquation() {
  SpeedFrequency = 60 * filteredSpeed / 3.6;
```

```
  SpeedPeriod = (SpeedFrequency > 0) ? (1000.0 / SpeedFrequency)
: 0;

  if (filteredSpeed <= 0.01) {
    SpeedFrequency = 0;
    SpeedPeriod = 0;
  }
}

void FlowEquation() {
  FlowFrequency = 1015 * filteredFlow / 60;
  FlowPeriod = (FlowFrequency > 0) ? (1000.0 / FlowFrequency) :
0;

  if (filteredFlow <= 0.01) {
    FlowFrequency = 0;
    FlowPeriod = 0;
  }
}

void tela() {
  char line0[20], line1[20], line2[20], line3[20];

  snprintf(line0, sizeof(line0), " R:%s   SPD:%.1f", Make,
filteredSpeed);
  snprintf(line1, sizeof(line1), " FLW:%.1f   PRS:%.1f",
filteredFlow, Press);
  snprintf(line2, sizeof(line2), "G 1 2 3 4 5 6 7 S F ");
  snprintf(line3, sizeof(line3), "%s %s %s %s %s %s %s %s %s
%s",
           sectionStates[7].c_str(), sectionStates[0].c_str(),
sectionStates[1].c_str(),
           sectionStates[2].c_str(), sectionStates[3].c_str(),
sectionStates[4].c_str(),
           sectionStates[5].c_str(), sectionStates[6].c_str(),
ConnS.c_str(), ConnF.c_str());
```

```
  lcd.setCursor(0, 0);
  lcd.print(line0);
  lcd.setCursor(0, 1);
  lcd.print(line1);
  lcd.setCursor(0, 2);
  lcd.print(line2);
  lcd.setCursor(0, 3);
  lcd.print(line3);
}
```