UNIVERSIDADE FEDERAL DE MATO GROSSO

FACULDADE DE ENGENHARIA – CAMPUS VÁRZEA GRANDE

RELATÓRIO DO PROJETO- ESTUFA.

KELVIN VINICIUS DA SILVA MAGALHÃES

CUIABÁ, 2019.

UNIVERSIDADE FEDERAL DE MATO GROSSO

FACULDADE DE ENGENHARIA – CAMPUS VÁRZEA GRANDE

RELATÓRIO DO PROJETO- ESTUFA.

Trabalho elaborado para fins avaliativos da disciplina Sistema de Controle 1, da Faculdade de Engenharia, Prof. Me Daniel Miranda Cruz.

KELVIN VINICIUS DA SILVA MAGALHÃES

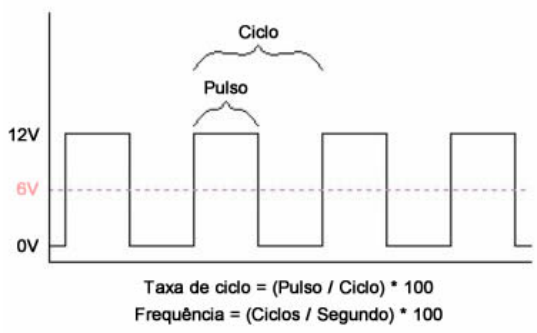
CUIABÁ, 2019.

Sumário

1. Introdução

Porque controlar a temperatura de um ambiente fechado? Nos dias de hoje o controle de variáveis dos mais diversos tipos são essenciais para a automação de processos, no século passado controlar a temperatura era extremamente complicado e sem precisão, sem equipamentos como controladores e sensores de alta precisão, o trabalho quase sempre era manual e propicio a erros; mas o que mudou de lá pra cá? Com a criação e desenvolvimento da tecnologia dos controladores e sensores tornou possível fazer aplicações automáticas, ou seja, uma vez instalado o controle de temperatura por exemplo funcionaria sem nenhum tipo de auxilio humano, exceto quando trata-se de manutenção, e esse controle de temperatura é amplamente utilizado em diversos setores como indústria, lazer, medicinal, militar e outros , e devido a essa ampla utilização o seu custo caiu, podendo construir um pequeno sistema de controle de temperatura barato mais eficiente como será desenvolvido neste projeto.

Para elaboração deste projeto será necessário controlar a velocidade de um ventilador e isso será feito por meio de PWM (‘pulse width Modulation’) refere-se ao conceito de pulsar rapidamente um sinal digital em um condutor. Além de várias outras aplicações, esta técnica de modulação pode ser utilizada para simular uma tensão estática variável e é comumente aplicada no controle de motores elétricos, aquecedores, LEDs ou luzes em diferentes intensidades ou frequências [1].



ura 5 – *PWM* e termos

Em termos de definição, temos:

Ciclo ou Período – o intervalo de tempo entre a subida de um pulso (dado em segundos);
Frequência – a taxa de bordas de subida de um pulso (dado em Hz ou ciclos por segundo). É simplesmente o inverso do período;
Taxa de Ciclo – tempo no período em que o pulso está ativo ou alto, dividido pelo tempo de ciclo (é dado em porcentagem do período completo)

Figura 1: PWM

Fonte: [1]

Também temos os sistemas de controle que realizam as correções no sistema, o controle PID (Proporcional Integral Derivativo) é uma das técnicas mais empregadas quando se deseja realizar o controle de variáveis contínuas. O controle PID consiste em um algoritmo matemático, que tem por função o controle preciso de uma variável em um sistema, permitindo

ao sistema operar de forma estável no ponto de ajuste desejado, mesmo que ocorram variações ou distúrbios que afetariam sua estabilidade [2].

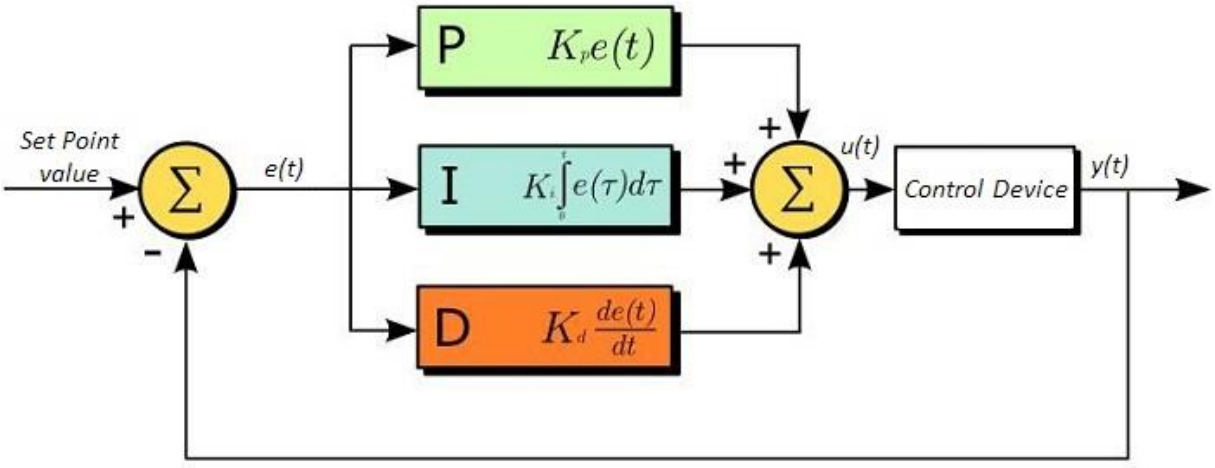


Figura 2:PID

Fonte: [3]

Além da estrutura física da planta feita de madeira, será utilizado um microcontrolador de baixo custo, Arduino Mega, responsável por controlar o sistema, também faz a aquisição de sinais e controla o atuador.

2. Desenvolvimento

2.1 Definição da Equipe

A equipe será constituída de apenas um integrante, o autor do relatório.

2.2 Divisão de tarefas

Todas as atividades relacionadas ao trabalho ficaram a cargo do autor do relatório.

2.3 Cronograma

Existem 2 cronogramas, o estático que trata-se do cronograma inicial na figura abaixo e o dinâmico que é atualizado constantemente e pode ser acompanhado através do link: https://trello.com/b/feuC7YD6/estufa .

CRONOGRAMA PROJETO DE SISTEMA DE CONTROLE 1 - Estufa- Kelvin Magalhães			
Atividade	**Dia /mês**	**Situação**	**Observação**
E1 - Definição da equipe	14/dez	Concluído	
E1 - Divisão de tarefas	14/dez	Concluído	
E2.1 - Escolha do processo	15/fev	Validação	
E2.1 - Estudo e suas características	21/dez	Validação	
E2.1 -Definição da placa controladora/Aquisição de sinal	15/fev	Concluído	
E2.1 -Definição do sistema de medição	15/fev	Validação	
E2.1 - Definição do sistema de atuação	15/fev	Validação	
E2.1 - Simulação e entendimento do modelo	15/fev	Validação	
E2.1 - Listagem de materiais	15/fev	Validação	
E2.2 - Aquisição de materiais	20/fev	Validação	
E2.2 - Eletrônica em funcionamento	20/fev	Validação	
E2.2 - Montagem e teste do sistema de medição	26/fev	Validação	
E2.2 - Montagem e sistema de atuação	26/fev	Validação	
E2.2 - Aquisição de sinal via placa controladora	26/fev	Validação	
E2.3 - Ensaio e identificação em malha aberta e modelage	01/mar	Não iniciado	
E2.3 - Comparação do modelo teórico desenvolvido	01/mar	Não iniciado	
E2.3 - Projeto de um controlador simples	01/mar	Não iniciado	
E2.3 - Ajuste de período de amostragem	01/mar	Não iniciado	
E2.3 - Discretização da lei de controle	01/mar	Não iniciado	
E2.3 - Implementação do algoritmo	09/mar	Não iniciado	
E2.3 - Definição das especificações	09/mar	Não iniciado	
E2.3 - Projeto dos parâmetros de ajuste do controlador	09/mar	Não iniciado	
E2.4 - Projeto de estrutura PID2DOF	15/mar	Não iniciado	
E2.4 - Projeto usando lugares da raízes	15/mar	Não iniciado	
E2.4 - Comparação do desempenho dos controladores	15/mar	Não iniciado	
E2.4 - PID - Industrial: Tratando Saturação e modo de segu	15/mar	Não iniciado	
E3 - Sistema de controle em funcionamento	05/abr	Não iniciado	
E3 - Relatório Final	05/abr	Não iniciado	
E3 - Apresentação oral e experimental	05/abr	Não iniciado	
E3 - Proposta de melhoria e trabalhos futuros	05/abr	Não iniciado	

Figura 3: Cronograma

Fonte: Própria.

2.4 Escolha do processo

O processo escolhido trata-se de um sistema de controle de temperatura em um ambiente, um modelo de estufa em pequena escala, onde temos uma lâmpada incandescente de 70 W que fornece o aquecimento necessário ao sistema, um sensor de temperatura LM35 que é responsável por medir a temperatura do ambiente e um pequeno ventilador que será controlado pela temperatura do ambiente, ou seja, se a temperatura está alta em relação á um padrão definido o ventilador terá uma velocidade alta , já se a temperatura for abaixo do referencial o ventilador terá uma velocidade baixa ou desligará. Aqui temos a simulação da construção física do projeto.

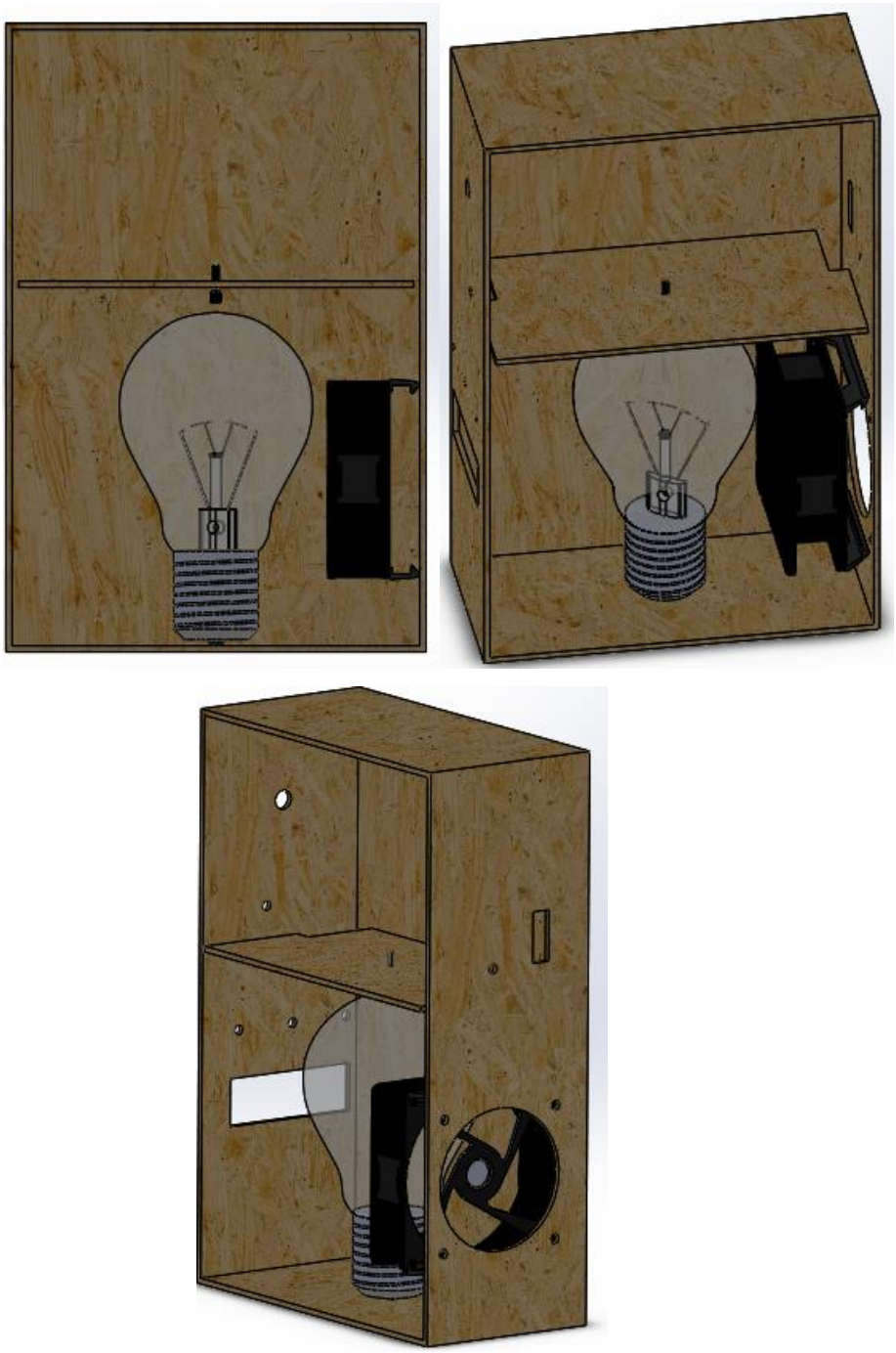


Figura 4: Modelagem da Planta

Fonte: Própria.

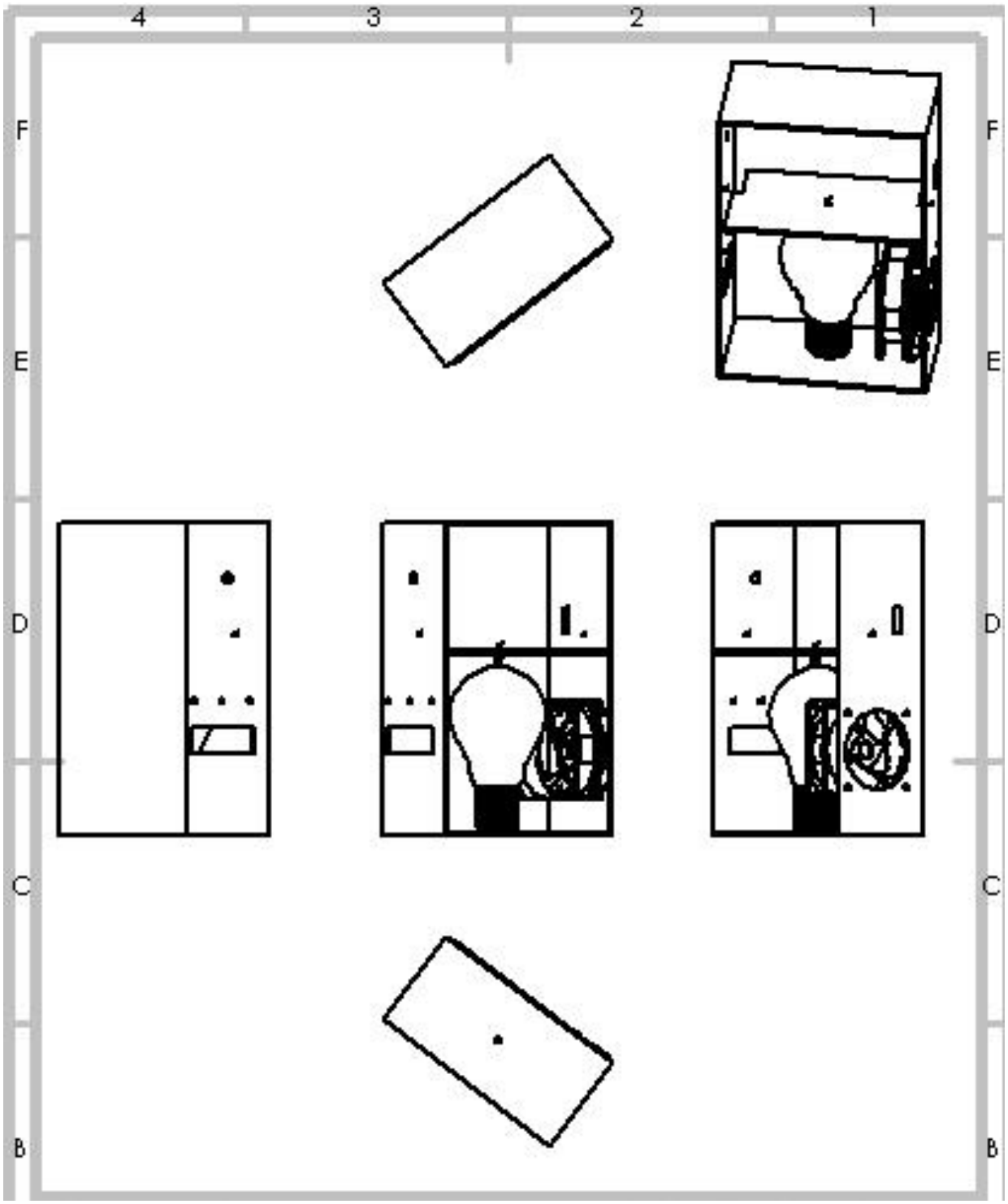
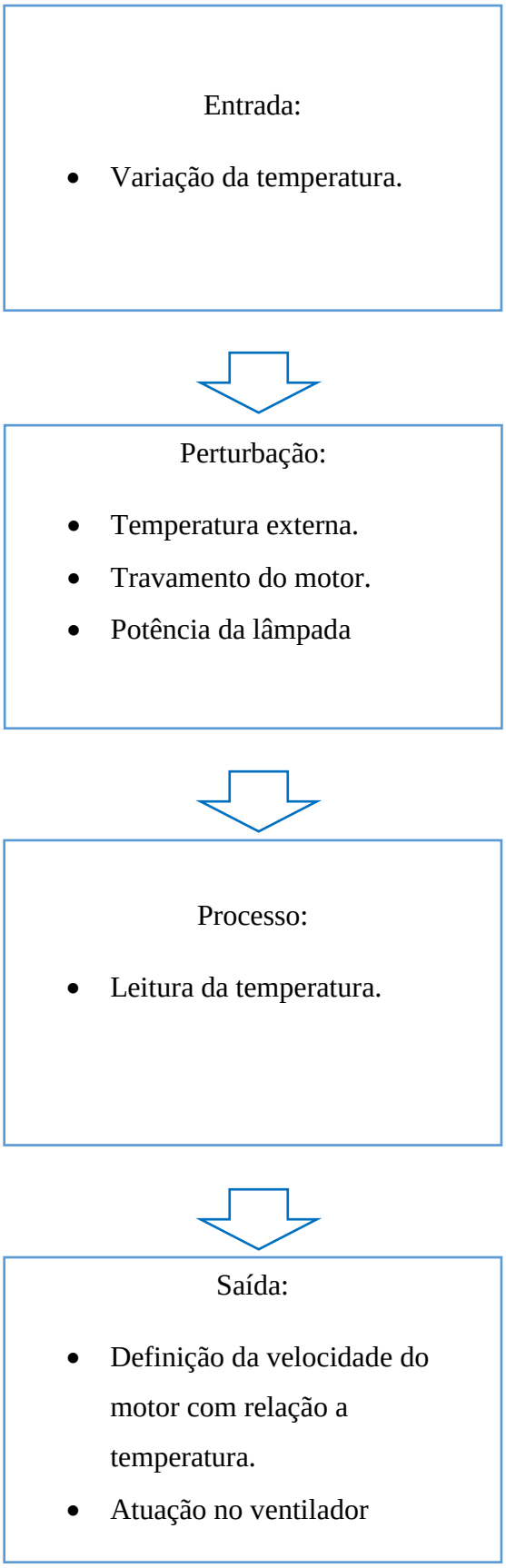


Figura 5: Desenho da Construção

Fonte: Própria.

E este é o fluxograma do sistema:



2.5 Estudo e suas características

Para simular o aquecimento de temperatura será utilizado uma lâmpada incandescente , que trata-se de uma lâmpada amplamente utilizada nas residências até início do século XXI, que tem um baixo custo de aquisição mas que tem maior consumo quando comparada a lâmpadas utilizadas atualmente que são as , Fluorescentes e LED , esse alto consumo está diretamente relacionado ao seu aquecimento, boa parte da energia consumida torna-se calor, por ter baixo custo o sistema ser em pequena escala é a fonte de calor ideal . Para simular o sistema de ventilação, foi utilizado um cooler de computador que tem a função de retirar calor de um ambiente fechado, o que também o torna ideal para esta aplicação. O sistema de medição é constituído por um sensor de temperatura de baixo custo que tem faixa de medição entre -55°C a +150°C. Além de um microcontrolador Arduino, que se trata de um controlador amplamente difundido na área acadêmica e de baixo custo também. Tal controlador foi escolhido pelo custo e pela quantidade de informação disponível sobre ele. Todo este conjunto de componentes ficaram alojados numa caixa de madeira.

O sistema funcionará da seguinte forma, ligamos a chave liga/desliga da lâmpada que aquecerá e será monitorado a temperatura do sistema através do sensor de temperatura LM35 até que ela estabilize (acompanhando pelo scope ou display do Simulink), definimos uma temperatura desejada que seria o Setpoint, e aguardamos a ação do controlador que atua na velocidade do ventilador até que a temperatura desejada seja atingida e mantida . Lembrando que se deve respeitar os limites de temperatura que o ventilador consegue atingir, pois, é de baixa potência e levando em consideração que a temperatura é uma variável lenta e pode demorar alguns segundos para que se note diferença. Podemos aplicar perturbações como tentar travar o cooler, desligar a lâmpada ou diminuir sua potência e ver como o sistema reage.

2.6 Definição da placa controladora/Aquisição de sinal

A placa controladora utilizada será o Arduino MEGA que contém um chip ATMEGA 2560, essa escolha foi feita devido ao baixo custo deste microcontrolador, a facilidade de acesso a informação sobre ele. A aquisição de sinais se dará por meio da comunicação via cabo serial do microcontrolador, com a aquisição de sinais e controle sendo gerenciadas pelo Simulink.

Temos a disposição o seguinte microcontrolador:

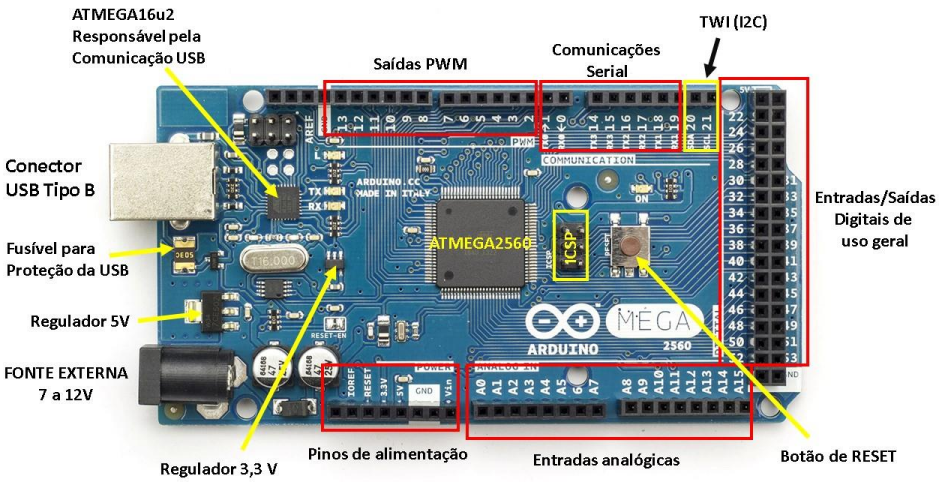


Figura 6: Arduino Mega

Fonte: [4]

Baseada no microcontrolador ATmega2560, possui 54 pinos de entradas e saídas digitais onde 15 destes podem ser utilizados como saídas PWM. Possui 16 entradas analógicas, 4 portas de comunicação serial. Além da quantidade de pinos, ela conta com maior quantidade de memória que Arduino UNO, sendo uma ótima opção para projetos que necessitem de muitos pinos de entradas e saídas além de memória de programa com maior capacidade [4].

2.7 Definição do sistema de medição

O sistema de medição será baseado em um único componente, o sensor de temperatura LM35.

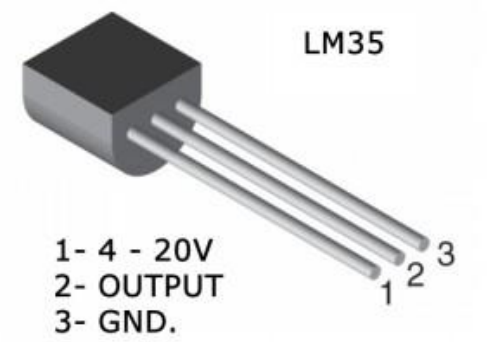


Figura 7: MPU 6050

Fonte: [5]

O Sensor de Temperatura LM35 é um componente eletrônico digital desenvolvido para ser aplicado nos mais diversos ambientes, medindo temperaturas entre -55°C a +150°C. Para

que o sensor de temperatura entre em funcionamento, é necessário estar conectado junto a uma plataforma de prototipagem, neste caso será o Arduino. [5]

Ele apresenta uma tensão de saída linear relativa a temperatura, e retorna um valor analógico referente a temperatura que o ambiente está apresentando. Para conversão dos valores de tensão para temperatura utilizamos a seguinte fórmula:

$$Temperatura(C) = \frac{leitura(V) * 5 * 100}{1023}$$

Equação 1

2.8 Definição do sistema de Atuação

O sistema de atuação consiste na ação de controle que atua no ventilador , temos a temperatura mensurada pelo sensor de temperatura, o controlador recebe essa temperatura aplica o controle e retorna os dados aplicando a correção no atuador.

2.9 Simulação e entendimento do modelo

Para simular o funcionamento, temos que abrir o programa Matlab, ir até o toolbox Simulink e abrir o arquivo ‘Controle_principal. O toolbox Simulink serve como supervisório. Para permitir esta conexão Matlab e Arduino, foi necessário adicionar a biblioteca ‘Arduino from support Simulink’ que não é nativa do software, isto pode ser feito diretamente no software. E só funcionou no Matlab 2015 com um Arduino Mega, anteriormente estava-se trabalhando com Arduino Uno, mas devido a limitações no Simulink, a placa controladora foi trocada. A configuração é simples, basta alterar no Simulink o tempo de simulação, definir como ‘External’ e colocar em execução verificando o funcionamento. Um exemplo abaixo:

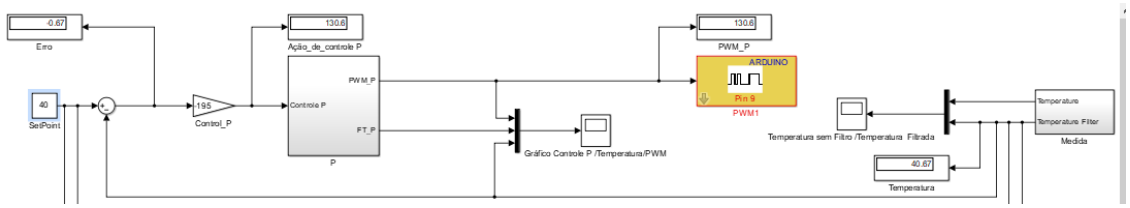


Figura 8: Diagrama Controle P

Fonte: Própria.

O SetPoint é a referência de temperatura desejada e pode ser alterada em tempo real, já para os outros parâmetros só podem ser alterados parando a simulação. Podemos observar

alguns valores diretamente na tela do Simulink como mostrado acima são eles: temperatura, erro, controle PWM, ação de controle ou podemos abrir o scope'Gráficos_controle_temperatura/PWM' e verificar os dados em forma de gráfico. Para o controle proporcional temos o seguinte gráfico abaixo, com a curva em rosa representando o controle Proporcional, em azul a temperatura e em amarelo o PWM aplicado ao motor.

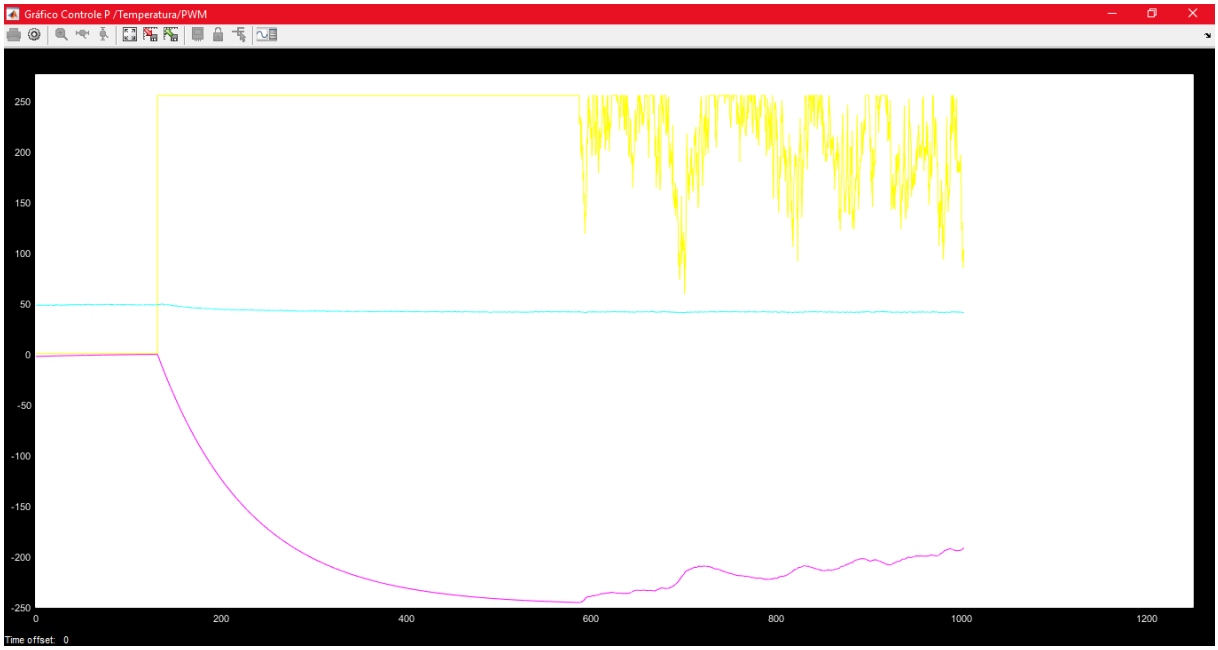


Figura 9: Gráficos controlador P

Fonte: Própria.

2.10 Listagem de materiais

Quantidade	Componentes	Valor
1	Arduino Uno/Mega	R$ 40,00
1	Suporte Lampada Incadescente	R$ 6,00
1	Lampada Incadescente 70W	R$ 10,00
1	Cooler Computador	R$ 10,00
1	Circuito de controle de velocidade(TIP 31C, Diodo, Led)	R$ 5,00
1	Sensor de Temperatura LM35	R$ 11,00
1	Fonte 12 Vdc	R$ 10,00
1	Protoboard	R$ 10,00
1	Caixa de Madeira	R$ 50,00
-	Jumpes	R$ 1,00
	TOTAL	R$ 153,00

Figura 10: Lista de componentes

Fonte: Própria.

2.11 Aquisição de materiais

Materiais já foram adquiridos, o custo foi especificado na *seção 2.10.*

2.12 Eletrônica em funcionamento

A disposição dos componentes eletrônicos está indicada abaixo:

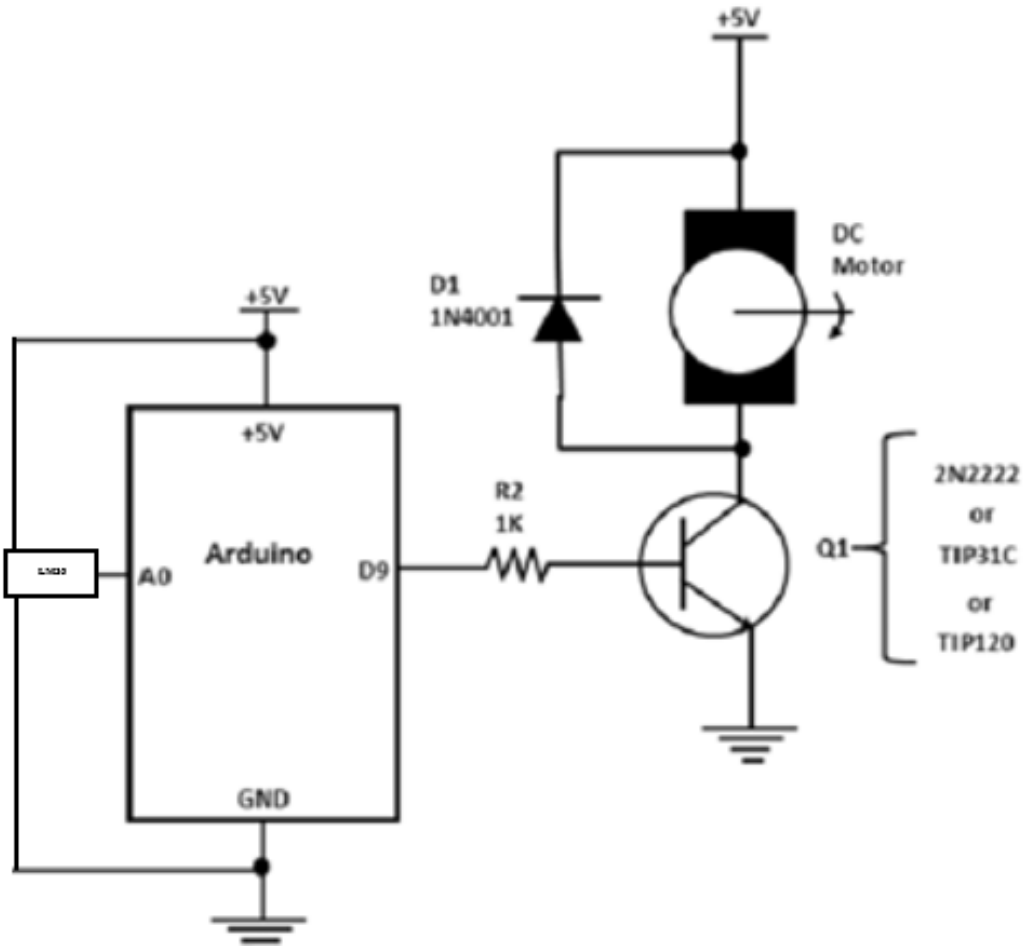


Figura 11: Diagrama eletrônico

Fonte: [6]

O funcionamento eletrônico dos componentes ocorre da seguinte forma.

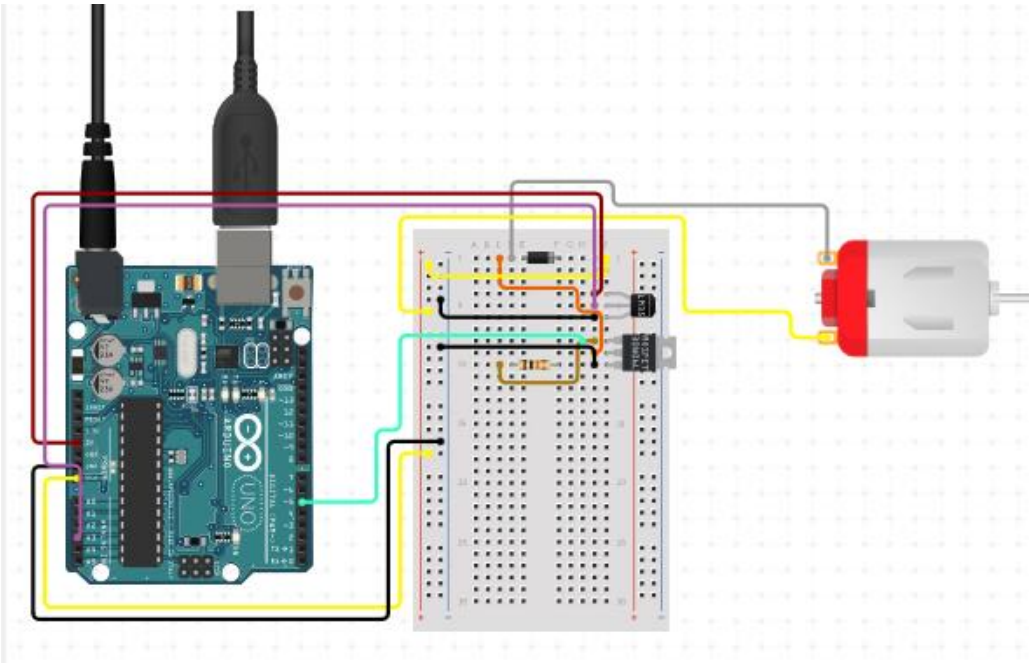


Figura 12: Simulação Circuito IO

Fonte:Própria.

Microcontrolador (Arduino Mega): É o cérebro do sistema, pois, faz a aquisição de sinais e tratamento das variáveis, ou seja, quem controla o sistema. É alimentado pelo cabo serial conectado ao computador.

Lâmpada Incandescente: Responsável pelo fornecimento de calor ao sistema, é ligada diretamente na rede AC através de um adaptador para tomada, em 127 V e tem potência de 70W.

Sensor de temperatura LM35: Ligado diretamente ao Arduino que fornece a alimentação de 5V necessária, e seu pino 2 trata-se do pino de dados que é responsável por enviar os valores analógicos medidos, a temperatura é uma unidade analógica.

Ventilador: Responsável pera retirada de calor do sistema, com corrente máxima de 0.22 A, que é ligado a uma fonte externa de 12Vdc que fornece a corrente necessária para o bom funcionamento do sistema, caso fosse diretamente no Arduino haveria problemas com a velocidade do motor, pois, a corrente é baixa.

Circuito de Controle de velocidade: Para realizar o controle de velocidade do motor DC foi necessário um transistor TIP 31C que permitiu o uso da função PWM (Pulse Width Modulation), onde a velocidade pode ser variada de 0 a 255(8 bits). Também foi necessário um

resistor de 1 kΩ e um diodo, e uma fonte externa para construção do circuito. Abaixo está o circuito elaborado e como foi conectado ao microcontrolador.

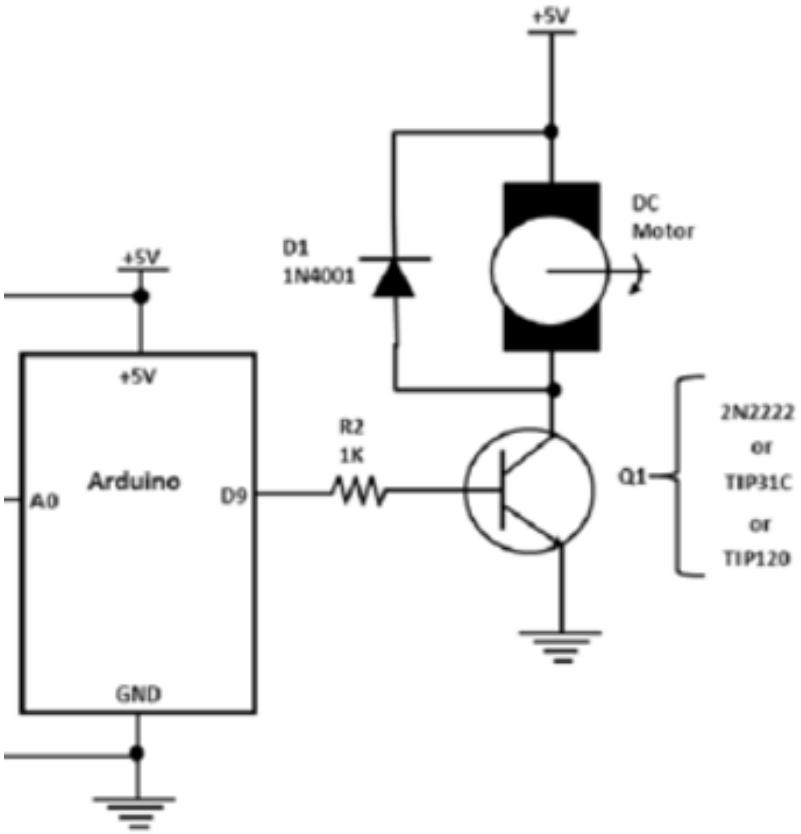


Figura 13: Diagrama eletrônico

Fonte: [6]

Fonte externa 12 VDC: Tem a função de alimentar o circuito de controle do motor DC, trata-se de uma fonte de 12VDC com 2 A de corrente, que é suficiente para atender o circuito.

2.13 Montagem e teste do sistema de medição

O sistema de medição foi acoplado a um suporte de madeira para ficar numa posição estratégica na caixa, próximo a lâmpada e ao ventilador.

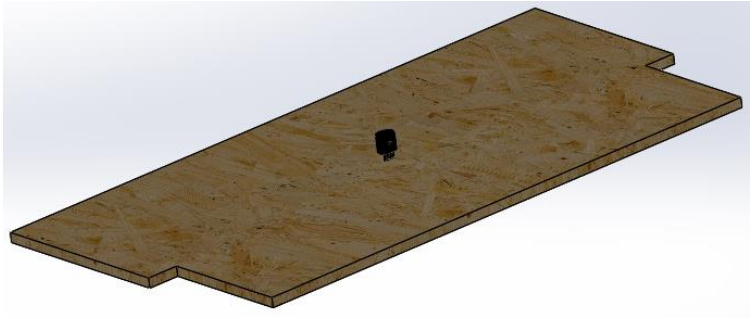


Figura 14: Sistema de medição

Fonte: Própria.

Temos 3 pinos, um para alimentação de 5 V e GND conectados diretamente no Arduino, e um pino de dados analógicos que está conectado ao pino A0 do Arduino. E a temperatura é requerida pelo controlador a cada 1 segundo.

2.14 Montagem e sistema de atuação

O sistema de medição foi acoplado a caixa de madeira para ficar numa posição estratégica na caixa, próximo a lâmpada e ao sensor de temperatura.

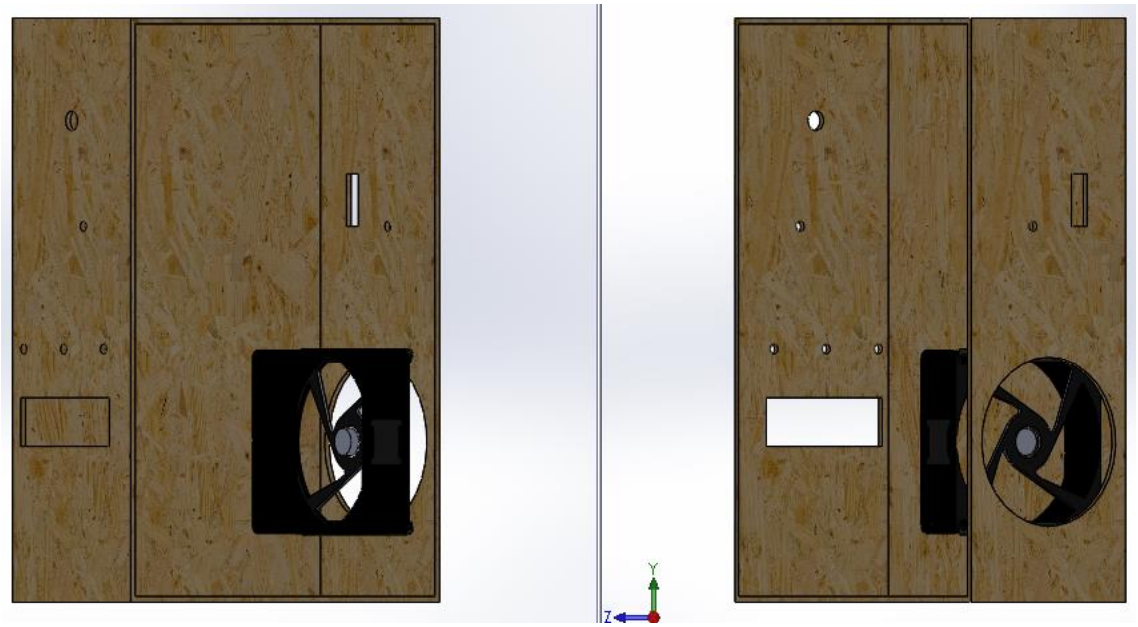


Figura 15: Sistema de Atuação

Fonte:Própria.

Para controlar o ventilador temos 2 pinos, apenas de alimentação. Então é necessário um circuito que permita a alteração da tensão aplicada ao motor, isso será feito por meio do circuito de controle de motor DC já citado anteriormente, que é conectado ao pino 9 de PWM do controlador, ou seja, a tensão aplicada ao ventilador será controlada pelo controlador e a

tensão está diretamente relacionada a velocidade do ventilador e será mapeada da seguinte forma: O PWM tem uma faixa de atuação de 0 a 255 , sendo 255 igual ao valor máximo de tensão no caso 5 V e 0 igual ao valor mínimo de tensão que é 0 V e essa tensão será variada entre 0 a 5 V conforme a ação de controle.

2.15 Aquisição de sinal via placa controladora

A aquisição de sinais será feita via comunicação serial USB, os dados serão recebidos no toolbox do Matlab- Simulink. A taxa de amostragem foi definida em $Ts = 0.1$s (segundos) para que o controle tenha resposta a tempo de atuar no sistema sem que ele sature, pelas características do sensor que tem um tempo de leitura de 2s, que é muito maior que a amostragem; Para corrigir essa inconsistência de ter valores não realísticos na medida, foi utilizado um filtro que coleta dados de 2 em 2 segundos e também evita ruídos na medição.

$$Filtro = \frac{1}{2s + 1}$$

Equação 2

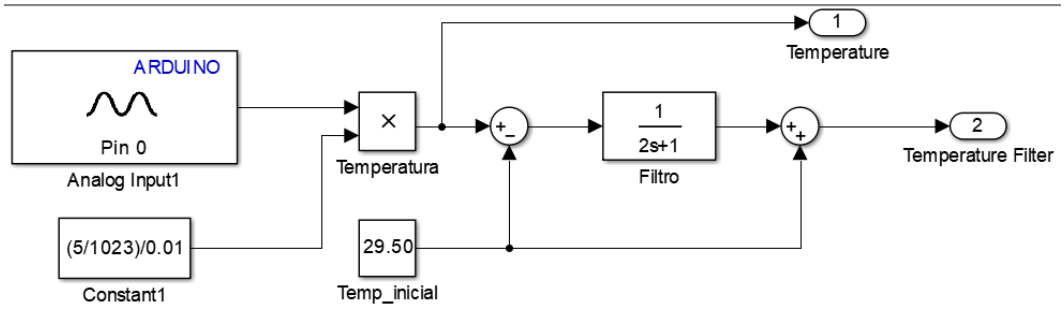


Figura 16: Diagrama de Temperatura com Filtro

Fonte: Própria.

2.16 Ensaio e identificação em malha aberta e modelagem do sistema

Através do ensaio em malha aberta foi possível identificar a função transferência do sistema.

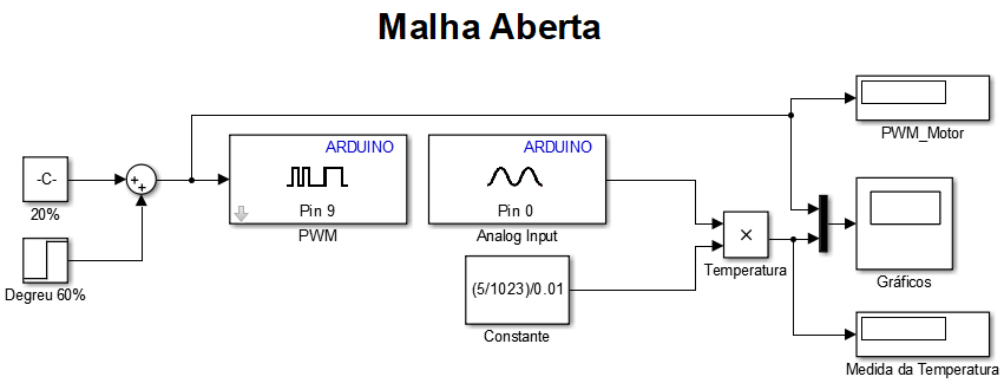


Figura 17: Malha Aberta

Fonte: Própria

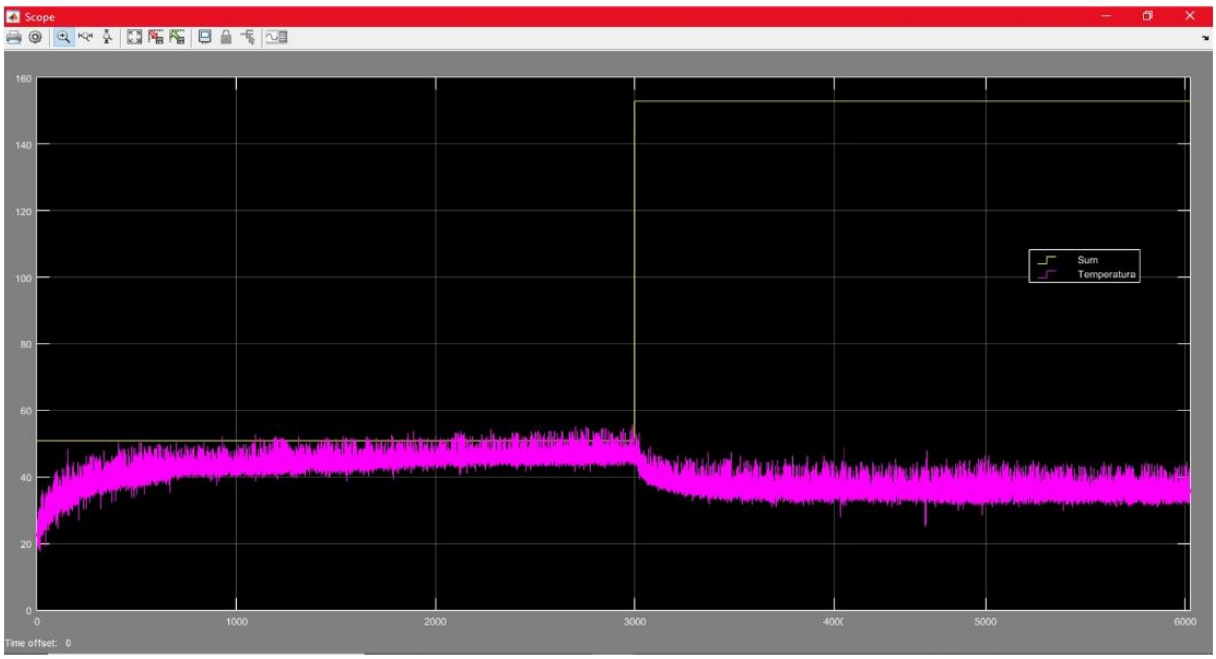


Figura 18: Ensaio Malha Aberta

Fonte: Própria.

O ensaio foi feito da seguinte forma:

1- Definiu-se em 20% de PWM a atuação no ventilador com a lâmpada ligada, até que a temperatura se estabiliza.

$$255 \rightarrow 100\%$$

$$PWM \rightarrow 20\%$$

Equação 3

Logo,

$$PWM20\% = 51.$$

Então o sinal PWM aplicado no atuador é de 51. Após um tempo de 1000 segundos já é possível identificar uma estabilização em 44°C.

2- Aplicou-se um degrau, alterando a atuação no ventilador de 20% para 60% de PWM com a lâmpada ligada, até que a temperatura se estabiliza novamente.

Logo,

Analogamente ao passo 1, $PWM60\% = 153$.Após um tempo de 1000 segundo já é possível identificar uma estabilização em 36°C.

3- Para calcular o ganho estático Ke da função transferência, temos a seguinte fórmula:

$$Ke = \frac{(\Delta T * 0.632) - T0}{Uf - Ui} = \frac{\big((44 - 36) * 0.632\big) - 44}{60 - 20} = -0.9736$$

Equação 4

Onde,

T0 – temperatura inicial

Uf – PWM final

Ui – PWM inicial

O ganho estático é negativo, pois a variável temperatura é reversa a ação do atuador.

4- Para cálculo do τ utilizamos a seguinte fórmula:

$$T(63\%) = (-\Delta T * 0.632) + 44 = 39°\text{C}$$

Equação 5

Verificando nos dados o sistema atinge a temperatura de 39°C depois de 100 segundos.

Logo,

$$\tau = 100s$$

5- Para encontrar a função transferência, basta aplicar na fórmula os valores obtidos:

$$G(s) = \frac{Ke}{\tau s + 1} = \frac{-0.9736}{100s + 1}$$

Equação 6

Como já era esperado trata-se de uma função de primeira ordem, comum em sistemas térmicos.

2.17 Modelo Teórico

Para modelagem teórica irei considerar a transferência de calor apenas por convecção forçada, ou seja, o calor dissipado apenas pelo ventilador.

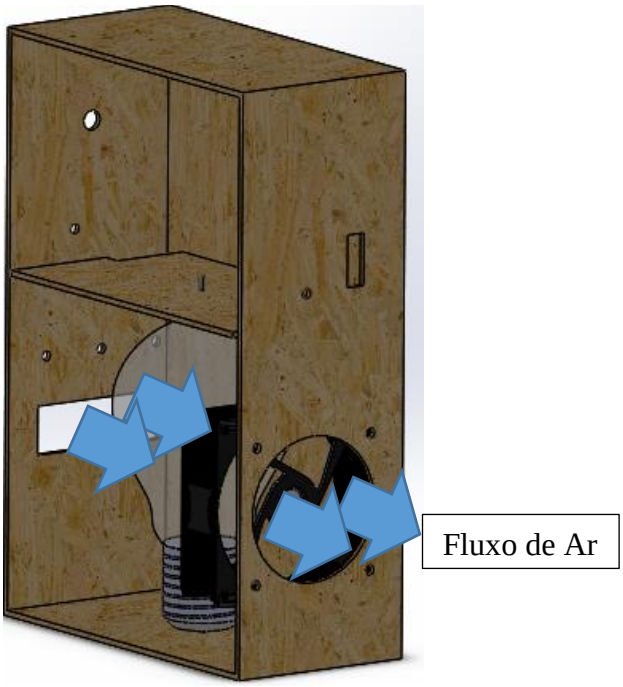


Figura 19: Direção do fluxo de ar

Fonte: Própria.

Temos que:

$$\frac{dQ}{dt} = h * As * (\frac{dTamb}{dt} - \frac{dTint}{dt})$$

Equação 7

Onde,

Q- é taxa de variação de calor do sistema

h- Coeficiente convectivo (constante)

As- Área da seção normal ao ventilador

Tamb- Temperatura ambiente externa

Tint- Temperatura interna

Com a taxa de calor, é possível encontrar o fluxo de calor.

$$\frac{dQ''}{dt} = \frac{\frac{dQ}{dt}}{As}\left(\frac{dTamb}{dt} - \frac{dTint}{dt}\right)$$

Equação 8

Então pode-se fazer uma analogia que o fluxo de calor pode ser visto como vazão de um fluido, neste caso o ar, mas porque fazer esta analogia? Precisa-se relacionar fluxo de calor com velocidade do ventilador e dessa forma conseguimos.

Se,

$$\frac{dQ''}{dt} = \frac{dV}{dt}$$

Equação 9

$$\frac{dV}{dt} = \frac{dv}{dt} * As$$

Equação 10

Onde $\frac{dV}{dt}$ é a vazão e $\frac{dv}{dt}$ é a velocidade, que podemos definir como:

$$\frac{dv}{dt} = \frac{\frac{dV}{dt}}{As}$$

Equação 11

$$\frac{dv}{dt} = \frac{\frac{dQ''}{dt}}{As}$$

Equação 12

$$\frac{dv}{dt} = \frac{\frac{\frac{dQ}{dt}}{As}\left(\frac{dTamb}{dt} - \frac{dTint}{dt}\right)}{As}$$

Equação 13

$$\frac{dv}{dt} = \frac{\frac{h * As * (\frac{dTamb}{dt} - \frac{dTint}{dt}))}{As}}{As} * (\frac{dTamb}{dt} - \frac{dTint}{dt})$$

Equação 14

Desta forma temos o sistema não linearizado:

$$\frac{dv}{dt} = \frac{h * (\frac{dTamb}{dt} - \frac{dTint}{dt})^2}{As}$$

Equação 15

E a velocidade ficará no intervalo 0 a 255, sendo referente ao PWM. Sendo necessário fazer o processo de linearização da função e depois fazer transformada de Laplace, mas como a planta terá modelagem através da pratica, foi desenvolvido só o modelo não linear teoricamente.

2.18 Comparação do modelo teórico desenvolvido

Realizando uma breve comparação entre o modelo teórico e o real, é possível notar que realmente o que afeta o sistema é a variação da temperatura externa e interna, já que a área e o coeficiente convectivo são constantes. Durante os testes práticos, a temperatura ambiente sempre tinha variação, em alguns casos variações grandes de até 8°C, ou seja, a modelagem podia não atender , isso se deve ao ponto de operação o qual o sistema foi modelado. Através da modelagem teórica foi possível entender como relacionar os diversos tipos de variáveis que envolve um sistema térmico e com resultados na pratica foi possível ver estes conceitos aplicados fisicamente.

2.19 Projeto de um controlador simples

Para calcular um controlador simples basta adicionar um controle proporcional integral (PI) em malha fechada, por alocação de polos para encontrar Kp e Ti [7].

1- Deseja-se que o sistema seja acelerado, então adicionamos um polo 4 vezes mais rápido que o $\tau = 100s$.

$$FTMFdesejada = \frac{-0.9736}{(25s+1)^2}$$

Equação 16

$$n = \frac{\tau}{4} = \frac{100}{4} = 25$$

Equação 17

$$Kp = \frac{2n-1}{K} = \frac{(2*25)-1}{-\frac{1}{4}} = -196$$

Equação 18

$$Ti = \frac{\tau 1(2n-1)}{n^2} = \frac{25*39}{25^2} = 1.56$$

Equação 19

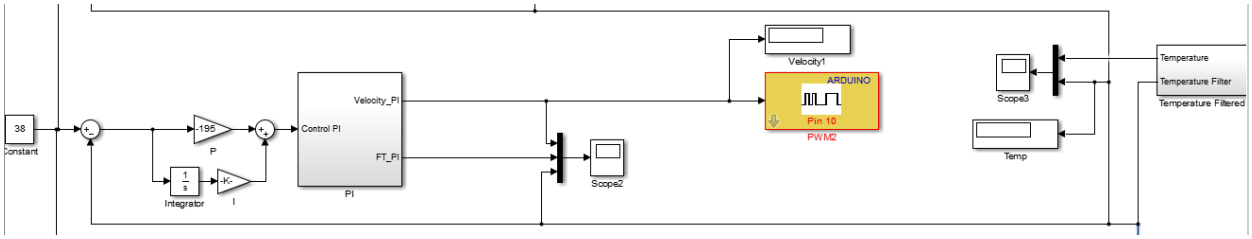


Figura 20:Controle PI

Fonte: Própria

2.20 Ajuste de período de amostragem

A taxa de amostragem foi definida em $Ts = 0.1s$ para que o controle tenha resposta a tempo de atuar no sistema sem que ele sature, pelas características do sensor que tem um tempo de leitura de 2s, que é muito maior que a amostragem; Para corrigir essa inconsistência de ter valores não realísticos na medida, foi utilizado um filtro que coleta dados de 2 em 2 segundos e também evita ruídos na medição.

$$Filtro = \frac{1}{2s+1}$$

Equação 20

2.21 Implementação do algoritmo

Implementação está dividida em 2 partes, o algoritmo no Matlab (‘code.m’) e o diagrama de blocos contendo a simulação no Simulink (‘Controle_principal.slx’), também foi elaborado o controle pela IDE do Arduino, arquivo ‘PID_Estufa.ino’. Estão anexados junto a este material.

2.22 Definição das especificações

Para especificações foram fixados alguns parâmetros, são eles:

1- $\tau R = \tau L/4$, ou seja, o tal rápido devia ser 4 vezes mais rápido que o tau lento, o tau lento foi obtido na função transferência encontrada, $\tau L = \tau = 100s$, portanto, $\tau R = 25s$.
2- Foi adotado um sistema super-amortecido, ou seja, ξ > 1, onde, os polos são negativos, diferentes.

2.23 Projeto dos parâmetros de ajuste do controlador

$$G(s) = \frac{-0.9736}{100s + 1}$$

Equação 21

Por alocação de polos, foi desejado um polo em 25s, ou seja, um polo 4 vezes mais rápido que o τ, ficando desta forma [7]:

$$G1(s) = \frac{-0.9736}{(100s + 1) * (25s + 1)} = \frac{-0.9736}{2500s^2 + 125s + 1}$$

Equação 22

Como a menor constante de tempo agora é o τR =25s. Podemos calcular $\omega 0 = \frac{1}{\beta \tau R}$,onde β=2/3, α=1 e ξ =1.1.

$$\omega 0 = \frac{1}{\frac{2}{3} * 25} = 0.06$$

Equação 23

Calculando as constantes para o controle PID, Kp, Ti e Td.

$$Kp = \frac{\tau R * \tau L * \omega 0^2(1 + 2 * \alpha * \xi) - 1}{Ke} = \frac{25 * 100 * 0.06^2(1 + 2 * 1 * 1.1) - 1}{-0.9736} = -26.7$$

Equação 24

$$Ti = \frac{\tau R * \tau L * \omega 0^2(1 + 2\alpha\xi) - 1}{\tau R * \tau L * \alpha * \omega 0^3} = \frac{25 * 100 * 0.06^2(1 + 2 * 1 * 1.1) - 1}{25 * 100 * 1 * 0.06^3} = 48.15$$

Equação 25

$$Td = \frac{\tau R * \tau L * \omega 0(\alpha + 2\xi) - \tau R - \tau L}{\tau R * \tau L * \omega 0^2(1 + 2\alpha\xi) - 1} = \frac{25 * 100 * 0.06(1 + 2 * 1.1) - 25 - 100}{25 * 100 * 0.06^2(1 + 2 * 1 * 1.1) - 1} = 12.5$$

Equação 26

Logo, por alocação de polos de 2ª ordem, temos os parâmetros calculados para o PID $Kp = -26.7, Ti = 48.15$ e $Td = 12.5$. Porém, este modelo não apresentou resposta satisfatória na pratica, principalmente tratando-se de Kp,então foi necessário realizar testes de tentativa e erro até achar os parâmetros ideais de funcionamento que são: $Kp = -195;\ Ti = \frac{1}{48.15} = 0.02;\ Td = 12.5$, isso pode ocorrer por diversos motivos, o teste realizado em malha aberta podia não ser a melhor escolha, o ambiente de teste de modelagem inicial e testes finais foram diferentes, então o ponto de operação é diferente em até 8° C. É muito comum realizar testes empíricos para sistemas de controle.

2.24 Projeto de estrutura PID2DOF

Devido a inconsistência de variáveis entre Simulink e Arduino , não consegui implementar a estrutura PID2DOF no Simulink, quando era adicionado o bloco PID no Simulink e adicionado os valores de $Kp, Ti\ e\ Td$ o controle deixava de funcionar, o bloco não aceitava a realimentação que continha a medição do processo, e como o bloco PID2DOF é apenas uma variação do bloco PID , ocorreu o mesmo problema.

Como havia desenvolvido o sistema primeiramente na IDE do Arduino e só depois foi passado para o Simulink, o controle PID2DOF e controle PID funcionou, como a IDE do Arduino é limitada em relação a ajuste de escala do gráfico, quantidade de variáveis e outros problemas pelo plotter Serial não foi possível mostrar os gráficos ,mas abaixo está a implementação que funcionou na planta, conforme mostrado durante o desenvolvimento do projeto.

PID_Estufa

```
class PID{
public:

  double error, errorP;
  double sample;
  double lastSample;
  double kP, kI, kD, b=1.0, c=1.0;
  double P, I, D;
  double pid;

  double setPoint;
  long lastProcess;

  PID(double _kP, double _kI, double _kD){
    kP = _kP;
    kI = _kI;
    kD = _kD;
  }

  PID_2DOF(double _kP, double _kI, double _kD, double _b, double _c){
    kP = _kP;
    kI = _kI;
    kD = _kD;
  }

```

PID_Estufa

```

  void addNewSample(double _sample){
    sample = _sample;
  }

  void setSetPoint(double _setPoint){
    setPoint = _setPoint;
  }

  double process(){
    // Implementação P ID
    error = sample - setPoint;
    errorP = sample - setPoint*b;
    float deltaTime = (millis() - lastProcess) / 1000.0;
    lastProcess = millis();

    //P
    //P = error * kP;
    P = errorP*kP;
    //I
    I = I + (error * kI) * deltaTime;

    //D
    //D = (lastSample - sample) * kD / deltaTime;
    D = (c*lastSample - sample) * kD / deltaTime;
    lastSample = sample;

```

Figura 21: Controle PID e PID2DOF parte 1

Fonte: Própria

PID_Estufa

```
51
52      // Soma tudo
53      pid = P + I + D;
54
55      if (pid >=255) pid = 255;
56      else if (pid <= 0) pid = 0;
57
58
59      return pid;
60    }
61  };
62
63  //double A = 0.25*0.25;  //area da estrutura
64  // double Vmax = 20.63;  // variavel maxima
65  double Tmax= 75.00;
66  double setpoint_temp = 39; // temperatura desejada
67  int controlePwm; // pwm do motor
68
69  #define pCONTROLE 9
70  PID meuPid(-195, 0.02, 12.5);// ajuste Kp,Ki,Kd
71  const int LM35 = A0; // Define o pino que lera a saída do LM35

73  void setup()
74  {
75    Serial.begin(9600);
76    meuPid.setSetPoint(setpoint_temp);
77    meuPid.addNewSample(1000);
78    pinMode(pCONTROLE, OUTPUT);
79  }
80
81
82  void loop()
83  {
84
85    double temp = (float(analogRead(LM35))*5/(1023))/0.01; // solicitar temperatur do lm35
86    meuPid.addNewSample(temp);
87    controlePwm = (meuPid.process());
88    analogWrite(pCONTROLE, controlePwm);
89    int Pwm = map(controlePwm, 0,65,0,255);
90    Serial.println(temp);
91    //Serial.print("    ,");
92    //Serial.print(setpoint_temp);
93    //Serial.print("    ,");
94    Serial.println(Pwm/10);
95    delay(1000);
96  }
```

Figura 22: Controle PID e PID2DOF parte 2

Fonte: Própria

2.25 Discretização da lei de controle

A descretização foi feita em cima do modelo obtido com alocação de polos através do Matlab utilizando a função ‘c2dm’ com método Tustin.

```
18      %% Discretizacao
19 -    numS     = [-0.9736];
20 -    denS     = [2500 125 1];
21 -    Ts       = 0.1;
22 -    [numZ,denZ]   =   c2dm(numS,denS,Ts,'tustin')
```

```
Command Window
Gpid =

        -0.9736
  --------------------
  2500 s^2 + 125 s + 1

Continuous-time transfer function.


numZ =

   1.0e-05 *

   -0.0971   -0.1942   -0.0971


denZ =

    1.0000   -1.9950    0.9950
```

Figura 23: Discretização

Fonte: Própria.

2.26 Projeto usando lugares das raízes

O LGR foi obtido através da função rlocus do Matlab. Temos apenas 2 polos e o sistema é estável, pois, encontram-se no semi-plano-esquerdo.

```
%% LGR
Ke= -0.9736;
num=Ke;
den=[2500 125 1];
G=tf(num,den)
%step(G);
```

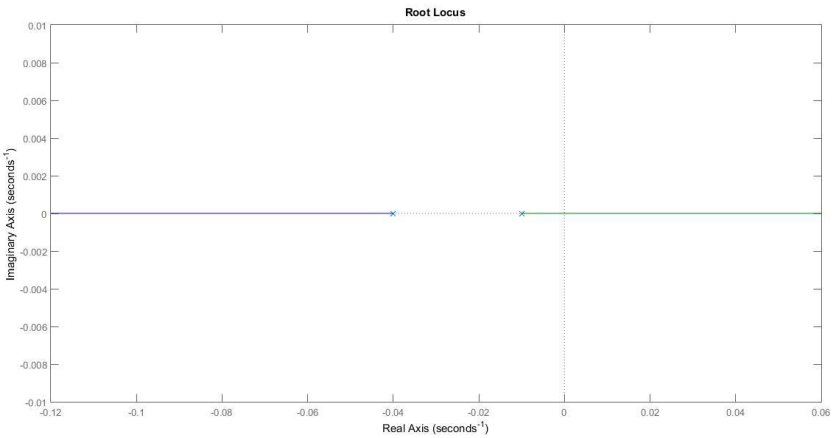


Figura 24: LGR

Fonte: Própria

2.27 Comparação do desempenho dos controladores

Antes de comparar o desempenho dos controladores vale lembrar que o modelo obtido tem ganho negativo, já que a temperatura é inversa a ação do ventilador, a curva em rosa é a ação de controle, a curva em amarelo é o PWM aplicado ao ventilador e em azul trata-se da medição de temperatura. Um outro ponto a considerar é que nesse sistema não é interessante aplicar perturbações, a temperatura é uma variável muito lenta, então por mais que se aplique perturbações o sistema irá demorar a responder, mas conseguirá retornar a referência, em qualquer um dos controladores abaixo.

Todos os controladores foram aplicados ao mesmo ensaio, temperatura inicial de aproximadamente 50°C com a lâmpada ligada, e o controle começa a atuar assim que o sistema é ligado.

O primeiro teste foi realizado com controle proporcional que obteve uma resposta satisfatória como podemos observar na imagem abaixo ,mas que demorou a chegar próximo ao Setpoint e ele não atinge o Setpoint pois o erro estacionário não permite que se atinja perfeitamente o valor desejado, porém, se um pequeno erro for aceitável em um sistema em relação ao valor desejado e não houvesse problema com o tempo até atuação ,seria um controle útil, por ser simples.

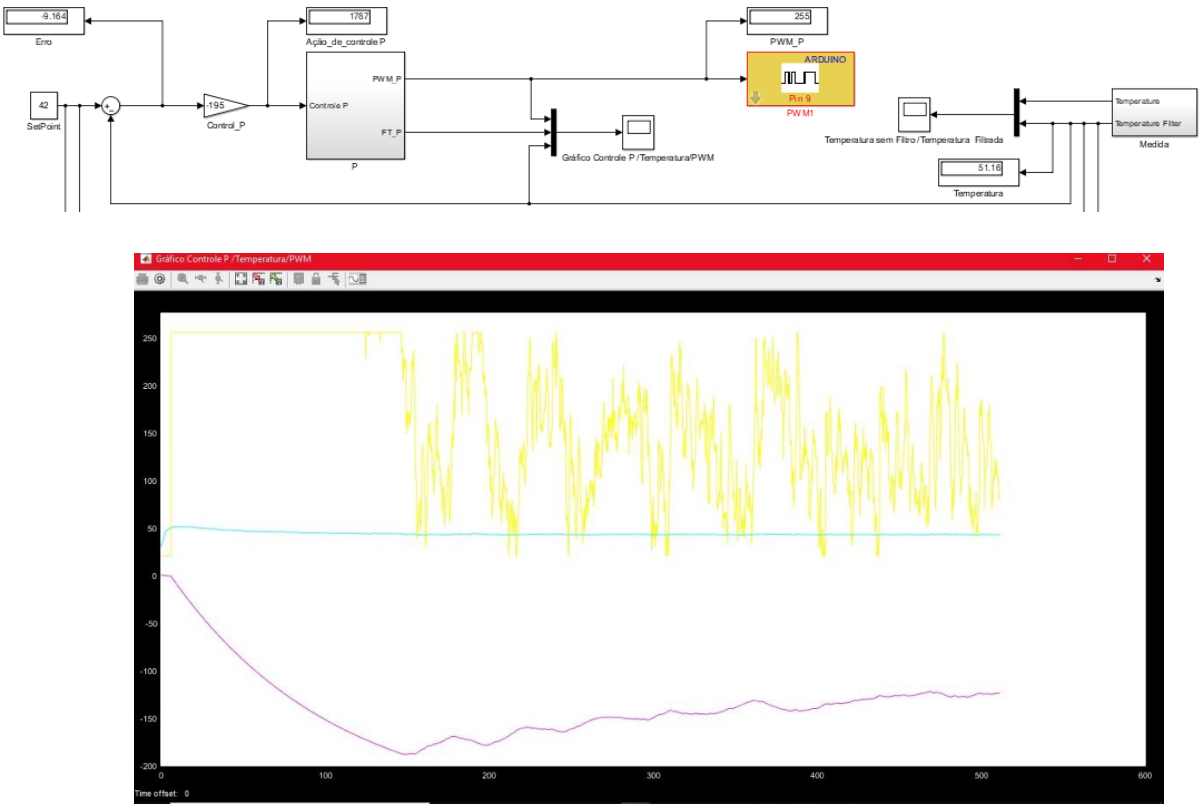


Figura 25: Diagrama de Blocos e Gráficos Controle P

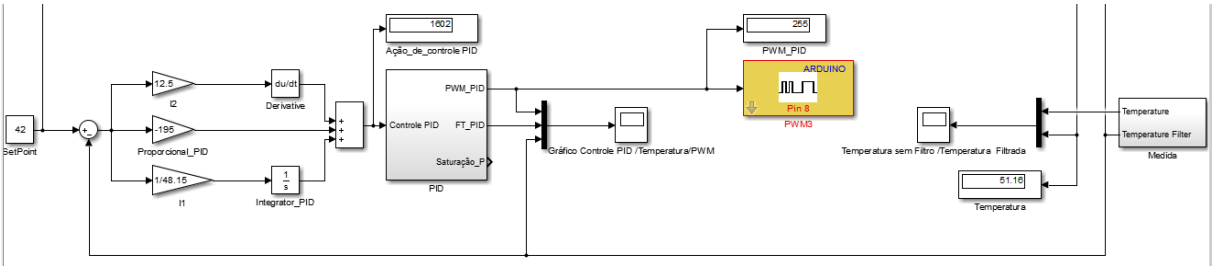
Fonte:Própria

Já para o controle proporcional integral foi adicionado um integrador ao sistema que tem como objetivo corrigir o erro estacionário, como podemos ver na imagem , trata-se de uma respostas mais rápida que o controle proporcional, mas não atingiu o Setpoint também, provavelmente o controlador elaborado não era o mais indicado ,pois, instabilizava com facilidade se os parâmetros de Kp e Td fossem minimamente alterados, mas apresentou resposta melhor que o controlador proporcional em questão de tempo.

Figura 26:Diagrama de Blocos e Gráficos Controle PI

Fonte: Própria

E o último controlador é o controle PID, onde temos a ação derivativa. Foi o que apresentou melhor resposta em questão de atingir o valor do Setpoint e o tempo que levou para atingi-lo, portanto seria o controle mais indicado para este sistema.



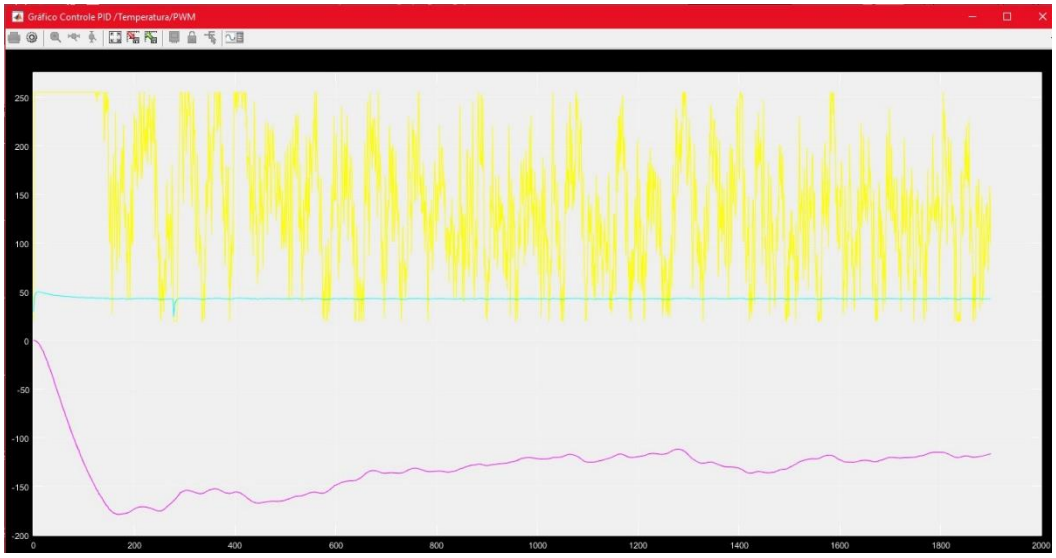


Figura 27:Diagrama de Blocos e Gráficos Controle PID

Fonte: Própria

2.28 PID - Industrial: Tratando Saturação e modo de seguimento

Devido a limitações físicas, é necessário adicionar saturador ao sistema, pois, ação de controle no atuador só pode atuar entre 20 a 255, sendo 20 o limite mínimo em que o ventilador consegue rodar, abaixo disso não existe controle, e o limite superior é o 255 que é o máximo de fluxo que o ventilador suporta. Mas o que faremos com essa ação de controle quando ela for maior que 255, podemos usar o AntiWindup que é uma correção e melhora o controle, essa ferramenta vai conectada ao integrador como pode se observar na imagem abaixo.

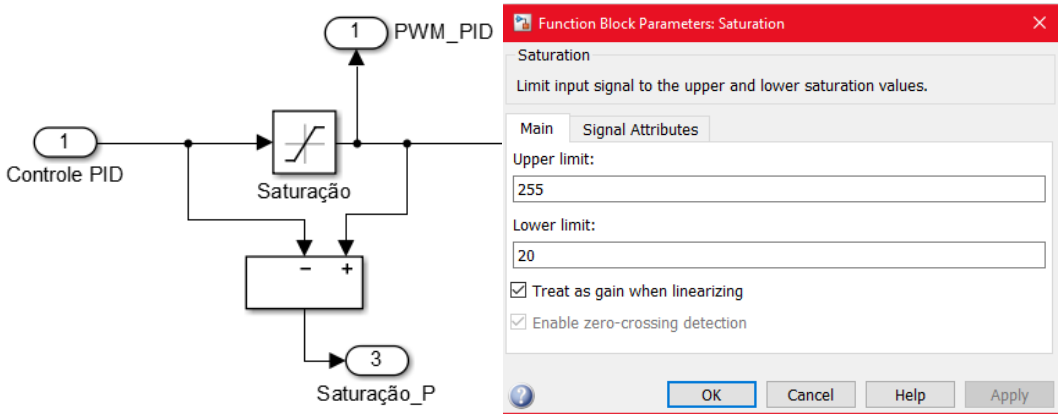


Figura 28: Saturação

Fonte: Própria

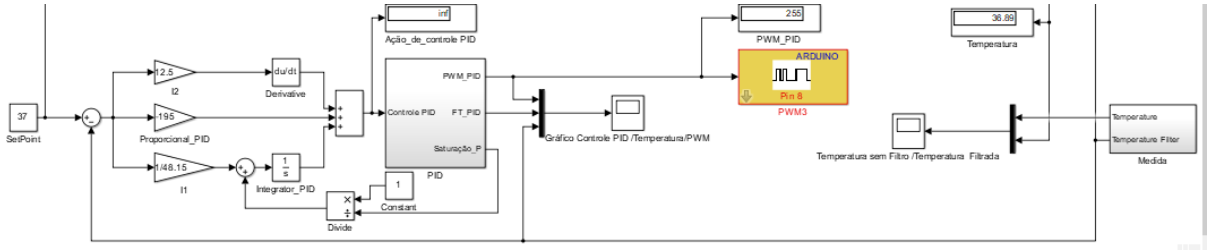


Figura 29:PID com AntiWindup

Fonte: Própria.

2.29 Trabalhos Futuros

Como trabalhos futuros fica o segmento deste projeto podendo incluir outros tipos de controle como controle em cascata, antecipativo, controle avançado e outros, seria interessante adicionar mais variáveis ao sistema tanto de medição quanto de atuação, poderia ser por exemplo a umidade dentro do sistema tornando-se um sistema multivariável e assim mais complexo.

Outra melhoria seria utilizar um display que se indica a temperatura e a ação do ventilador podendo assim acompanhar o sistema sem a necessidade de se ter um computador do lado, seria um sistema embarcado.

Trocar o sensor de temperatura por um que tenha maior estabilização na leitura, um grande problema deste projeto foi o ruído na leitura dos dados e a troca do sensor de temperatura seria interessante.

3. Conclusão

O projeto permitiu realizar um comparativo de sistemas de controle, neste caso o controle PID foi o mais indicado devido as suas características, como a temperatura é uma variável lenta e era necessário acelerar o sistema sem que se ocorre a instabilização o controle PID teve a melhor resposta. Trabalhar com sistemas térmicos é interessante porque são processos muito comuns na indústria e este projeto deu uma noção de como cada característica externa pode afetar.

Outro tópico fundamental deste trabalho são as diferenças dos modelos reais e dos modelos teóricos, houve grande diferença dos controles calculados, sistemas não lineares são complexos e os parâmetros devem ser calculados como referencial de onde partir, mas não são garantia que aquele sistema vai funcionar exatamente como foi projetado.

4. Bibliografia

1. SILVEIRA, C. B. Citisystems. **O que é PWM e Para que Serve?** Disponivel em: <https://www.citisystems.com.br/pwm/>. Acesso em: 13 de Abril de 2019.

2. FREITAS, C. M. Embarcados. **Controle PID em sistemas embarcados**, 2014. Disponivel em: <https://www.embarcados.com.br/controle-pid-em-sistemas-embarcados/>. Acesso em: 13 de Abril de 2019.

3. THORLABS. **PID Basics**. Disponivel em: <https://www.thorlabs.com/tutorials.cfm?tabID=5dfca308-d07e-46c9-baa0-4defc5c40c3e>. Acesso em: 13 de Abril de 2019.

4. AQEEL, A. Introduction to Arduino Mega 2560. **Theengineeringprojects**, 2018. Disponivel em: <https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-mega-2560.html>. Acesso em: 13 de Abril de 2019.

5. USINAINFO. Usinainfo. **Sensor de temperatura LM35 para Projetos**. Disponivel em: <https://www.usinainfo.com.br/sensor-de-temperatura-arduino/sensor-de-temperatura-lm35-para-projetos-3099.html>. Acesso em: 18 de fevereiro de 2019.

6. C2O. **Sistema Didático com Bombeamento, Comutação e Detecção**. Disponivel em: <http://www.c2o.pro.br/hackaguas/ar01s09.html>. Acesso em: 13 de Abril de 2019.

7. PHR. **Alocação de polos para sistemas de 2ª ordem**. Automação, controle e instrumentação para o setor do Petróleo e Gas. Florianópois-SC, p. 1-8.

8. SOUZA, F. Embarcados. **Arduino MEGA 2560**, 2014. Disponivel em: <https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 13 de Abril de 2019.