

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO
CURSO SUPERIOR DE ENGENHARIA ELÉTRICA**

GUSTAVO FLORE CAVENAGO

**CONSTRUÇÃO DE UMA INCUBADORA DE OVOS AUTOMATIZADA COM
ARDUINO COM CONTROLE DE TEMPERATURA UTILIZANDO PID,
FUZZY E GPC**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2016

GUSTAVO FLORE CAVENAGO

**CONSTRUÇÃO DE UMA INCUBADORA DE OVOS AUTOMATIZADA COM
ARDUINO COM CONTROLE DE TEMPERATURA UTILIZANDO PID,
FUZZY E GPC**

Trabalho de conclusão de curso apresentado à
Coordenação de Engenharia Elétrica da Universi-
dade Tecnológica Federal do Paraná como requi-
sito parcial para obtenção do título de “Graduação
em Engenharia Elétrica”.

Orientador: Prof. Dr. Rodrigo Rodrigues Sumar

CORNÉLIO PROCÓPIO
2016



Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica



FOLHA DE APROVAÇÃO

Gustavo Flore Cavenago

Construção de uma incubadora de ovos automatizada com arduino com controle de temperatura utilizando PID, Fuzzy e GPC

Trabalho de conclusão de curso apresentado às 10:20hs do dia 18/11/2016 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

Prof(a). Dr(a). Rodrigo Rodrigues Sumar - Presidente (Orientador)

Prof(a). Dr(a). Kleber Romero Felizardo - (Membro)

Prof(a). Dr(a). Emerson Ravazzi Pires da Silva - (Membro)

Dedico este trabalho à Deus, minha noiva, famílias e amigos!

AGRADECIMENTOS

Primeiramente agradeço a Deus, por todas as bênçãos e maravilhas que Ele tem feito por mim, mesmo sem merecimento sua graça me alcançou, me dando o dom da vida, me lavando nas águas do Santo Batismo e prometendo a coroa de vida eterna se eu for firme e fiel.

Agradeço pela minha noiva, que também é graças a Deus, e que se não fosse por ela não teria conhecido a graça de Deus, não teria estudado em Cornélio e muito menos redigiria este trabalho. Por sempre ter me apoiado e ter tido paciência durante a faculdade e realização do TCC, estando ao meu lado nos momentos bons e ruins, não deixando faltar o amor e a compreensão. Sendo minha principal motivação para querer crescer e dar o melhor de mim, cuidando para fazer tudo certo e o melhor para nós, como casal.

Agradeço pelas minhas famílias, tanto por parte de Pai, Mãe, Madrasta e da minha Noiva, por terem me apoiado e acreditado em mim, sempre me animando e ajudando em tudo o que fora preciso para estar de pé lutando no dia a dia, sem deixar faltar nada. Este trabalho de conclusão de curso é mais de vocês do que meu, pois sem a mão de vocês para me empurrar poderia eu nunca ter chegado onde cheguei !

Por último, e não menos importante, agradeço a todos meus amigos, que tinha antes da faculdade e os que fui fazendo durante a mesma, onde nestes estão incluídos todos que moraram comigo, estudaram, estiveram ao meu lado, ensinaram (professores que não poderei esquecer suas contribuições na minha vida acadêmica e profissional) e serviram (colaboradores da universidade que sempre fizeram ótimos almoços e jantas, me trataram super bem, limpavam o ambiente para que pudéssemos estudar, estavam dispostos a resolverem os problemas burocráticos, entre outros). Sem estes amigos faltaria a faculdade, lugar pra morar, motivação para estudar e tudo quanto fora acrescentado pelos mesmos.

Everything should be as simple as it can be, but not simpler.
(EINSTEIN, Albert, 1950)

Tudo deveria se tornar o mais simples possível, mas não simplificado. (EINSTEIN, Albert, 1950)

RESUMO

CAVENAGO, Gustavo Flore. **CONSTRUÇÃO DE UMA INCUBADORA DE OVOS AUTOMATIZADA COM ARDUINO COM CONTROLE DE TEMPERATURA UTILIZANDO PID, FUZZY E GPC**. 2016. 100 f. Trabalho de conclusão de curso – Curso Superior de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

A avicultura tem papel fundamental na vida das pessoas, sendo responsável por produzir intensivamente a carne de frango, comercializada nos mercados. Tal produção é devida às incubadoras de ovos, que automatiza e propicia um meio favorável ao desenvolvimento do ovo. Assim, foco principal deste trabalho foi analisar os métodos PID, Fuzzy e GPC no controle da temperatura de uma incubadora de ovos, garantindo que a temperatura se mantenha na referência desejada (principal garantia de qualidade). Construiu-se um protótipo com suporte para 12 ovos e automatizado com a plataforma Arduino. Durante o estudo percebeu-se a necessidade de interagir e modificar alguns métodos na tentativa de obter melhores respostas da temperatura interna, utilizando algoritmos genéticos para otimização dos parâmetros dos controladores. Após o estudo implementou-se o controlador escolhido e verificou na prática o funcionamento do protótipo, utilizando um controlador simples (duplamente proporcional) de fácil implementação, atendendo aos requisitos da proposta inicial do trabalho com eficácia.

Palavras-chave: Incubadora de ovos. Arduino. PID. Fuzzy. GPC.

ABSTRACT

CAVENAGO, Gustavo Flore. **CONSTRUCTION OF AN EGGS INCUBATOR AUTOMATED WITH ARDUINO WITH TEMPERATURE CONTROL USING PID, FUZZY AND GPC.** 2016. 100 f. Trabalho de conclusão de curso – Curso Superior de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

The poultry industry farming has a fundamental role in the lives of people, being responsible for intensively producing chicken meat, marketed in the markets. Such production is due to egg incubators, which automates and provides a favorable environment for egg development. Thus, the main focus of this work was to analyze the PID, Fuzzy and GPC methods in the temperature control of an egg incubator, ensuring that the temperature is maintained at the desired reference (main quality assurance). A prototype was built with support for 12 eggs and automated with the Arduino platform. During the study the need to interact and modify some methods was verified in the attempt to obtain better internal temperature responses, using genetic algorithms to optimize the parameters of the controllers. After the study, the chosen controller was implemented and the prototype operation was verified, using a simple controller (double proportional) of easy implementation, meeting the requirements of the initial proposal of the work with effectiveness.

Keywords: Eggs incubator. Arduino. PID. Fuzzy. GPC.

LISTA DE FIGURAS

FIGURA 1	– Fluxograma do trabalho.	11
FIGURA 2	– Percentual da mortalidade embrionária no período de incubação.	12
FIGURA 3	– Exemplo da lógica Fuzzy.	16
FIGURA 4	– Funções de pertinências da variável de temperatura.	16
FIGURA 5	– Fluxograma do Algoritmo Genético.	17
FIGURA 6	– Microcontrolador ATmega328p utilizado no Arduíno.	18
FIGURA 7	– Incubadora de ovos finalizada.	20
FIGURA 8	– Entrada degrau e saída em "s" transladada para zero.	21
FIGURA 9	– Entrada discreta controlada e saída próximo à referência.	22
FIGURA 10	– Entrada Degrau e <i>fit</i> de 1 polo e 0 zeros.	23
FIGURA 11	– Entrada Degrau e <i>fit</i> de 2 polo e 1 zeros.	23
FIGURA 12	– Entrada discreta parcialmente controlada e <i>fit</i> de 2 polo e 1 zeros. ...	24
FIGURA 13	– Entrada discreta parcialmente controlada e <i>fit</i> de 1 polo e 0 zeros com atraso.	25
FIGURA 14	– Simulação do controle PI discreto.	27
FIGURA 15	– Temperatura de saída pelo método ZN.	27
FIGURA 16	– Energias pelo método ZN.	28
FIGURA 17	– Convergência dos resultados analisados no processo do AG.	29
FIGURA 18	– Temperatura de saída pelo método AG.	29
FIGURA 19	– Energias pelo método AG.	30
FIGURA 20	– Comparação dos métodos de Ajuste ZN e AG.	30
FIGURA 21	– Simulação do controlador Fuzzy aplicado à planta.	31
FIGURA 22	– Funções pertinências da entrada manual (input).	32
FIGURA 23	– Funções pertinências da saída manual (output).	32
FIGURA 24	– Curva/superfície para o Fuzzy manual.	33
FIGURA 25	– Regras para o Fuzzy manual (Funções de entrada que ativam as funções de saída).	33
FIGURA 26	– Visualização das regras para o Fuzzy manual.	34
FIGURA 27	– Temperatura interna com Fuzzy ajustado manualmente.	35
FIGURA 28	– Energias de controle Fuzzy ajustado manualmente.	35
FIGURA 29	– Convergência do processo de otimização pelo AG.	36
FIGURA 30	– Funções pertinências da entrada manual (input).	36
FIGURA 31	– Funções pertinências da saída manual (output).	37
FIGURA 32	– Curva/superfície para o Fuzzy manual.	37
FIGURA 33	– Visualização das regras para o Fuzzy manual.	38
FIGURA 34	– Temperatura interna com Fuzzy ajustado com AG.	38
FIGURA 35	– Energias de controle Fuzzy ajustado com AG.	39
FIGURA 36	– Comparação das respostas PI-AG, FUZZY-MANUAL e FUZZY-AG. ...	39
FIGURA 37	– Variações da leitura do sensor de temperatura.	40
FIGURA 38	– Simulação do controlador duplo proporcional aplicado à planta.	41
FIGURA 39	– Convergência do processo de otimização pelo AG.	42
FIGURA 40	– Resposta da temperatura interna.	42
FIGURA 41	– Energias fornecidas pelos controladores.	43
FIGURA 42	– Comparação do resultado do PP-AG com PI-AG e FUZZY-AG.	43

FIGURA 43 – Comparação do resultado com PI-AG e FUZZY-AG com zoom.	44
FIGURA 44 – Resposta da temperatura interna GPC-manual.	45
FIGURA 45 – Energia fornecida pelo controlador GPC-manual.	46
FIGURA 46 – Convergência do processo de otimização pelo AG.	46
FIGURA 47 – Temperatura interna de saída da incubadora GPC-AG.	47
FIGURA 48 – Energia fornecida pelo controlador GPC-AG.	47
FIGURA 49 – Comparação das temperaturas internas de saída da incubadora com GPC-manual e GPC-AG.	48
FIGURA 50 – Comparação das energias dos controladores GPC-manual e GPC-AG.	48
FIGURA 51 – Comparação das energias dos controladores GPC-manual e GPC-AG.	49
FIGURA 52 – Simulação do controle PID+P-ZN discreto.	50
FIGURA 53 – Temperatura de saída pelo método PID+P-ZN.	51
FIGURA 54 – Energias pelo método PID+P-ZN.	51
FIGURA 55 – Convergência dos resultados analisados no processo do AG.	52
FIGURA 56 – Temperatura de saída pelo método AG.	53
FIGURA 57 – Energias pelo método AG.	53
FIGURA 58 – Comparação dos métodos de Ajuste ZN e AG para o PID+P.	54
FIGURA 59 – Comparação dos controladores para mesma referência.	55
FIGURA 60 – Comparação dos controladores para mesma referência com zoom. ..	55
FIGURA 61 – Controlador escolhido - PP-AG.	56
FIGURA 62 – Pinagem do microcontrolador Atmega328p.	57
FIGURA 63 – Pinagem do microcontrolador Atmega328p.	58
FIGURA 64 – Tabela de dados obtida pela leitura serial do microcontrolador.	59
FIGURA 65 – Resposta real da temperatura da incubadora.	61
FIGURA 66 – Elementos de madeira para sustentação principal.	64
FIGURA 67 – Projeto do esquemático no programa Proteus [®]	65
FIGURA 68 – Geração dos arquivos da primeira placa.	66
FIGURA 69 – Confecção manual PCI.	67
FIGURA 70 – Correção das falhas nas trilhas.	68
FIGURA 71 – Corrosão da placa de cobre.	69
FIGURA 72 – Furos da PCI.	70
FIGURA 73 – Montagem final da PCI.	70
FIGURA 74 – Produção da segunda PCI.	71
FIGURA 75 – Metalização dos furos.	72
FIGURA 76 – Segunda PCI finalizada.	73
FIGURA 77 – Montagem do ovoscópio (lâmpada 40W, soquete, cabo e plugue macho para tomada).	74
FIGURA 78 – Montagem final da incubadora.	74

SUMÁRIO

1	INTRODUÇÃO	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	AVICULTURA	12
2.2	CONTROLADORES INDUSTRIAIS	12
2.2.1	Controlador Proporcional Integrativo Derivativo (PID)	12
2.2.2	Controle Preditivo Generalizado	14
2.3	SISTEMAS INTELIGENTES ARTIFICIAIS	15
2.3.1	Sistemas Fuzzy	15
2.3.2	Algoritmos Genéticos	17
2.4	MICROCONTROLADORES	17
3	CONSTRUÇÃO DA INCUBADORA DE OVOS	19
4	PROJETO DOS CONTROLADORES	21
4.1	IDENTIFICAÇÃO DO SISTEMA	21
4.2	CONTROLE PI	24
4.3	CONTROLE FUZZY	28
4.4	CONTROLE PP	34
4.5	CONTROLE GPC	41
4.6	CONTROLE PID	49
4.7	ESCOLHA DO CONTROLADOR	54
5	IMPLEMENTAÇÃO DA PROGRAMAÇÃO	57
6	CONSIDERAÇÕES FINAIS	60
	REFERÊNCIAS	62
	APÊNDICE A – CONSTRUÇÃO DA INCUBADORA DE OVOS	63
	APÊNDICE B – CÓDIGOS DE PROGRAMAÇÃO - MATLAB® E ARDUINO®	75
B.1	CONTROLADOR PI	75
B.1.1	Função custo para o AG	78
B.2	CONTROLADOR PP	78
B.2.1	Função custo para o AG	79
B.3	CONTROLADOR FUZZY	80
B.3.1	Função custo para o AG	84
B.4	CONTROLADOR GPC	85
B.4.1	Função custo para o AG	89
B.5	CONTROLADOR PID	90
B.5.1	Função custo para o AG	93
B.6	EMBARCAÇÃO ARDUINO	93
	ANEXO A – CONTROLADOR PREDITIVO GENERALIZADO	98

1 INTRODUÇÃO

Com a constante demanda de novas tecnologias e aplicação de conceitos modernos na solução de problemas atuais do mercado, a área de automação e controle tem se destacado, por oferecer resoluções eficientes e econômicas às questões que têm surgido ao longo do tempo, como acionamento de máquinas, fontes chaveadas de energia elétrica, conversores de potência, entre outros.

Na avicultura há um constante aumento das necessidades tecnológicas para auxílio do seu funcionamento pleno e aumento de rendimento, visto que o Brasil tem elevado consumo de carne de frango e ovos. Para uma pecuária intensiva, utiliza-se de recursos automatizados a fim de garantir a integridade dos materiais e a maior produção com o menor custo. Apesar da existência de incubadoras eficientes, o presente trabalho tem a finalidade de não apenas estudar formas de controle nas incubadoras, mas também abordar toda a parte de projeto, desenvolvimento, implementação e análise das mesmas, verificando a interdisciplinaridade do curso na solução de problemas.

Existem muitos tipos de técnicas de controle, mas serão abordadas algumas que são vistas no curso (Fuzzy e PID - Proporcional Integrativo Derivativo) e outra (GPC - Controle Preditivo Generalizado) mais avançada, retratando as diferenças entre elas, os diferentes métodos de abordagem e considerações a fim de se determinar uma técnica eficiente e passível de implementação.

O trabalho consiste em unir elementos elétricos (sensores, atuadores, microprocessador, resistor, leds, entre outros) para fazer o controle de temperatura da incubadora. O fluxograma presente na Figura 1, mostra as etapas principais seguidas para realização do projeto.

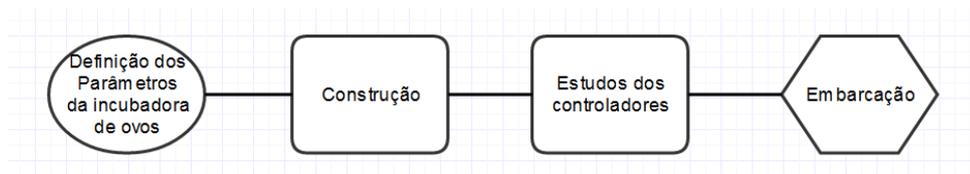


Figura 1 – Fluxograma do trabalho.

Fonte: Autoria Própria.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 AVICULTURA

A partir da década de 50, o Brasil deu início à avicultura, que se intensificou na década de 70, mudando seu polo produtor de São Paulo para região Sul. O consumo preponderante de carne bovina migrou para carne de frango, face à alta produção e a baixa nos preços devido a maximização de ganho de peso das aves. Em 2006 o Brasil atinge o patamar de segundo maior produtor do continente americano, sendo possível atingir tais resultados em decorrência da implementação de tecnologias como as incubadoras de ovos artificiais, gerando um aumento significativo da produção de aves devido a sua capacidade de chocar milhares de ovos ao mesmo tempo (ALMEIDA, 2008).

O período de incubação é tão importante quanto o pré e o pós incubatório, pois é nesta etapa que o embrião irá se desenvolver até que haja o nascimento do pinto. As taxas de mortalidade independem da modalidade do ovo e variam em torno de 1% a 5%, como pode ser visto na Figura 2 . Esta mortalidade não está apenas relacionada com a incubação, mas ela é o meio onde ocorre o desenvolvimento dos ovos e a aparição das falhas destes, assim para grandes avicultores esta porcentagem deve diminuir ainda mais para evitar prejuízos e garantir melhor qualidade na incubação dos ovos (ROSA; AVILA, 2000).

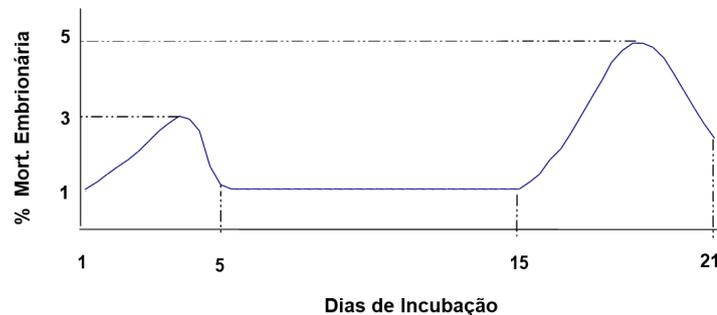


Figura 2 – Percentual da mortalidade embrionária no período de incubação.

Fonte: (ROSA; AVILA, 2000)

2.2 CONTROLADORES INDUSTRIAIS

2.2.1 Controlador Proporcional Integrativo Derivativo (PID)

Técnicas convencionais de controle têm perpetuado nas indústrias até os dias atuais. O controlador PID (Proporcional, Integrativo e Derivativo) ainda lidera em aplicações de controle por todo o mundo, persistindo por possuir uma alta capacidade de solucionar problemas e ser de fácil implementação, tanto analógica como digitalmente, além do baixo custo e a disponibilidade de ferramentas que possibilitam seu ajuste automático. O ajuste (sintonia) do PID vem sendo estudado até hoje, pois não há técnicas mais simples que solucionem os mesmos problemas de maneira mais eficiente (BERTACHI et al., 2013).

O PID é uma das aplicações mais difundidas e utilizado controle industrial, por sua fácil implementação, de forma digital ou analógica, pelo baixo custo e por ser um sistema que controla diversos tipos de plantas (sistemas a serem controlados) em malha fechada. A parcela proporcional é o ganho que o erro significa para o controle, ou seja, sendo o erro (e) entre o valor de referência (ref) e a saída atual (y), a energia aplicada (u) será proporcional a este erro. Assim quanto maior a constante de proporcionalidade (kp) mais rápida será a resposta do sistema. Na equação 1 tem-se os termos associados para o domínio do tempo (t) e da frequência (s), onde esta última é mais utilizada devido as funções de transferências e análises dos sistemas estarem todos neste domínio (OGATA, 2001).

$$u_p(t) = kp.e(t) \Rightarrow U_p(s) = kp.E(s). \quad (1)$$

Outra parcela do controlador PID é a integrativa, que tem a finalidade de armazenar o erro da saída em relação à referência e somar na energia aplicada proporcionalmente ao sistema, de forma a eliminar este erro. Conforme a equação 2, o termo de integral caracteriza a parcela integrativa, sendo representado no domínio da frequência pelo termo $\frac{1}{s}$. Durante o projeto do ganho ki , quanto maior for o mesmo, mais rápido o sistema elimina o erro, porém como o sistema aumenta sua velocidade, a inércia faz com que a resposta ultrapasse valores pré-definidos causando uma oscilação no sistema.

$$u_i(t) = ki \int_0^t e(t)dt \Rightarrow U_i(s) = \frac{ki}{s}E(s). \quad (2)$$

Por fim, a parcela derivativa tem a finalidade de detectar variações do erro e "reagir" proporcional ao ganho kd e à taxa de variação do mesmo, inserindo no sistema uma energia para contrabalancear a variação indesejada. Conforme a equação 3, o termo de derivada caracteriza a parcela derivativa, sendo representado no domínio da frequência pelo termo s . Durante o projeto do ganho kd , quanto maior for o mesmo, menor serão as taxas de variações do sistema, aumentando o amortecimento, assim há uma dificuldade maior do controlador em seguir uma referência que se altera com o passar do tempo. Sendo este termo muito incômodo à ruídos, que dão saltos na leitura e podem causar uma interpretação de mudança repentina para o sistema, ocasionando uma reposta indesejada do mesmo.

$$u_d(t) = kd \frac{d[e(t)]}{dt} \Rightarrow U_d(s) = kdsE(s). \quad (3)$$

Uma das formas de unir os três tipos de controladores acima é de forma paralela, onde a energia total ($u(t)$ ou $U(S)$) é resultado da soma das parcelas de energia proporcional, integrativa e derivativa, formando assim o controlador PID convencional representado pelas equações 4 e 5 (OGATA, 2001).

$$u(t) = kp.e(t) + ki \int_0^t e(t)dt + kd \frac{d[e(t)]}{dt}, \quad (4)$$

$$U(s) = kp.E(s) + \frac{ki}{s}E(s) + kdsE(s). \quad (5)$$

As aplicações desta modalidade de controladores são as mais diversas, estando presentes em diversos produtos do mercado atual, como por exemplo, servomotores, que têm sensores de posição interna, que determinam junto de um circuito (analógico ou digital), controlam a posição do rotor de acordo com a entrada de referência dada pelo usuário.

A determinação dos ganhos k_p , k_i e k_d , pode ser feita de várias formas, sendo as mais famosas técnicas de sintonias: Ziegler Nichols, Broida, Sundaren, entre outros. Cada uma das técnicas visa a generalizar uma fórmula, a fim de manter um parâmetro do controle fixo, porém são técnicas que podem precisar de um ajuste fino posterior.

2.2.2 Controle Preditivo Generalizado

Controle preditivo se baseia nos cálculos de ações futuras para tomar a ação atual, ou seja, é um controlador que atua de forma inteligente quanto ao estado presente do sistema a ser controlado e das condições futuras do mesmo, dada uma energia de controle calculada. Tem sido usado em indústrias para substituir os controladores convencionais como o PID. Por se tratar de uma técnica de controle moderno (controle em espaço de estados) e nova no mercado, ainda existem muitas pesquisas sobre o assunto, sendo adotado como referência de controle preditivo para este trabalho apenas a tese de doutorado do professor Rodrigo R. Sumar em (SUMAR, 2008).

A modelagem de plantas variantes no tempo é muito complexa devido a sua alteração nos parâmetros dependente do tempo, ou seja, se antes de iniciar um sistema for identificado seus parâmetros, analiticamente ou experimentalmente, de tempos em tempos deverá ser feita outra identificação devido à mudança dos mesmos. Por exemplo, as incubadoras de ovos podem ter seus sensores e atuadores (resistências metálicas para aquecer) danificadas devido à alta umidade relativa do ar, mudando seus parâmetros ao longo do tempo, até se degradarem a ponto de provocarem uma falha. Em cálculos de predição da resposta que utiliza o modelo da planta, se a mesma for mutável ao longo do tempo, o controle pode não atuar de forma correta.

Os métodos de modelagem usando entradas e saídas podem ser divididos em 4 categorias, como está na tabela 1. Quando existe o modelo da planta é porque foi obtido ou calculado o mesmo, já sem modelo da planta (*model-free*) são analisadas somente as entradas e saídas, que na verdade possui o modelo de forma empírica. Porém, a análise pode ser feita *offline* - extração das características antes de iniciar a operação - ou *online* - extração ao longo da operação, visando atualização do modelo.

Tabela 1 – Classificação dos controladores com base nos dados experimentais.

	Modelo da Planta	Sem Modelo da Planta
<i>Offline</i>	Projeto Indireto do Controle	Projeto Direto do Controle
<i>Online</i>	Adaptativo Indireto	Adaptativo Direto

Fonte: (SUMAR, 2008)

Dentro do modelo *free-model* de implementação *online* (principal atrativo desta técnica) pode-se dividir o mesmo em duas técnicas: sintonia *model-free* e controle *model-free*. As técnicas de sintonia *model-free* têm por finalidade determinar os parâmetros dos

controladores pelos dados de entrada e saída, utilizando critérios de desempenho em malha fechada, que, das técnicas a mais simples e mais robusta é a GPC (*Generalized Predictive Control* = Controle Preditivo Generalizado). Já as técnicas de controle *free-model* se adapta apenas com os dados de entrada e saída, onde as leis de controle geralmente são derivadas, mas em alguns casos utilizam de representações simplificadas da planta.

O autor define bem o conceito do GPC.

O controlador preditivo generalizado é uma extensão do controlador de variância mínima, cuja lei de controle, resultado da minimização do erro de seguimento de previsão sobre um horizonte de tempo finito, apresenta desempenho mais robusto que os controladores discretos convencionais e trata processos com atrasos variantes ou dinâmicas não-modeladas, de fase não-mínima e sistemas instáveis em malha aberta. (SUMAR, 2008, p.97).

2.3 SISTEMAS INTELIGENTES ARTIFICIAIS

2.3.1 Sistemas Fuzzy

Sistemas com Inteligência Artificial têm seu espaço garantido no futuro, devido ao grande avanço tecnológico, que permite um ambiente propício para seu desenvolvimento. Esta nova categoria de conhecimento permite a abordagem de sistemas similares aos biológicos, podendo atribuir a uma máquina, características semelhantes às de animais, sistemas naturais, entre outros. Este novo ponto de vista conceitual é emulado com os processadores e memórias presentes na atualidade, usando métodos iterativos podem resolver problemas com alto grau de complexidade.

A inteligência artificial (IA) tem habilidades para usar conhecimentos prévios para a resolução de problemas complexos e realizar tarefas que necessitem estar em constante aprendizado. O que antes era feito somente por um ser humano, hoje pode ser substituído por um pequeno circuito eletrônico, que não erra e nem cansa ao realizar suas funções. Todos os parágrafos a respeito de inteligência artificial são baseados nas notas de aulas de Sistemas Inteligentes, que por sua vez são adaptações do livro do professor Danilo H. Spatti (SILVA; SPATTI; FLAUZINO, 2010).

Redes neurais artificiais são ótimas candidatas para analisar as entradas dos parâmetros do sistema e tomar decisões. Tem sido usado amplamente nas pesquisas de sistemas incertos e limitados, por sua alta capacidade de identificar padrões e atuar rapidamente. Se assemelham as estrutura cerebrais reais, por meio de aprendizado e generalização de padrões. Por isso, devem passar por uma espécie de treinamento, onde as interações sinápticas artificiais se consolidam de forma a obter uma interação com o meio externo, ajustado para atender às especificações desejadas.

Fuzzy, por outro lado, tem sido usado amplamente em controles não lineares, por não precisar de dados de treinamento e ter uma alta abstração das variáveis de forma qualitativa. É uma abordagem não exata, que permite expressar incertezas e imprecisões do sistema, independentemente da modelagem matemática, baseado nas regras linguísticas do "especialista", que definirá as regras de funcionamento do mesmo. Este tipo de sistema assemelha-se com o processo de ocultamento de informação do cérebro, onde são escondidos as variáveis e números, que o cérebro utiliza para fazer o controle de nosso corpo,

processando as informações de formas incertas e facilitando a compreensão.

Diferente da lógica clássica, a lógica Fuzzy (nebulosa) não tem fronteiras de separação bem definidas, o tornando único por poder trazer o modelo de pensamento humano para máquinas, a fim de tratar dados qualitativos e esconder valores quantitativos. Um exemplo, mostrado na Figura 3, que retrata a separação da lógica clássica para Fuzzy. Pois se pode estabelecer limites para separar o dia quente do dia frio (não quente), que computacionalmente representaria fazer uma função if (se), condição de verificação, mas para Fuzzy, se tratarmos de uma temperatura de 26°C, neste exemplo, não representaria o frio, e sim pouco frio e pouco quente. Esta quebra de fronteira, permite que ações que seriam tomadas somente se frio, ou somente se quente, sejam ponderadas para compor uma ação que carregam informações de frio e de quente.



Figura 3 – Exemplo da lógica Fuzzy.

Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

A abordagem qualitativa se dá por uma linguagem mais simples, como alto, médio, baixo, médio alto, entre outros, que são chamadas funções de pertinências, que irão fazer a ponderação (interface) da entrada desta variável com os sistemas Fuzzy. Um exemplo, seria a interpretação da velocidade para um sistema Fuzzy, onde não há valores exatos do que é uma alta ou baixa velocidade, mas sim da representatividade (parcela) que este tem de baixa, média ou alta. Na Figura 4, têm-se os valores considerados muito baixo, baixo, médio, alto e muito alto que se diz a variável, porém a função de pertinência retorna um valor com os percentuais para cada um dos estados, restando a lógica Fuzzy, a partir da sua base de regras, definir qual será o resultado de saída ou ação a ser tomada. Também é possível notar que as funções de pertinências se sobrepõem umas com as outras, e suas formas são de vários tipos, no caso trapezoidal e triangular.

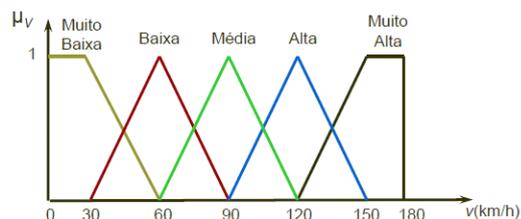


Figura 4 – Funções de pertinências da variável de temperatura.

Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

2.3.2 Algoritmos Genéticos

Algoritmos genéticos, é um conceito recentemente muito estudado, que visa aplicar as leis de Mendel de forma interativa computacionalmente. Semelhante à natureza, genes são informações (no caso armazenadas em forma de vetor), que para um determinado sistema ou estudo tem uma característica, ou seja, uma expressão gênica. De forma simples e objetiva o algoritmo genético cria então "genes" aleatórios, baseado nos limites mínimos e máximos pré-estabelecidos, que se deseja otimizar e inicia o processo de interações computacionais. Estas são: cruzamento dos genes e mutação (similar a natureza, mas com valores especificados e quantizados) e seleção natural dos "filhos"(resultado dos cruzamentos), eliminando os genes que possuem piores resultados nas simulações. Repetem-se os passos de cruzamento e seleção natural até que a população final seja apenas um tipo de gene.

Durante o *loop* do algoritmo tem-se o que entendemos por seleção natural, onde os genes são testados e os que geram os piores resultados, são eliminados. A Figura 5, mostra o funcionamento interno da função.

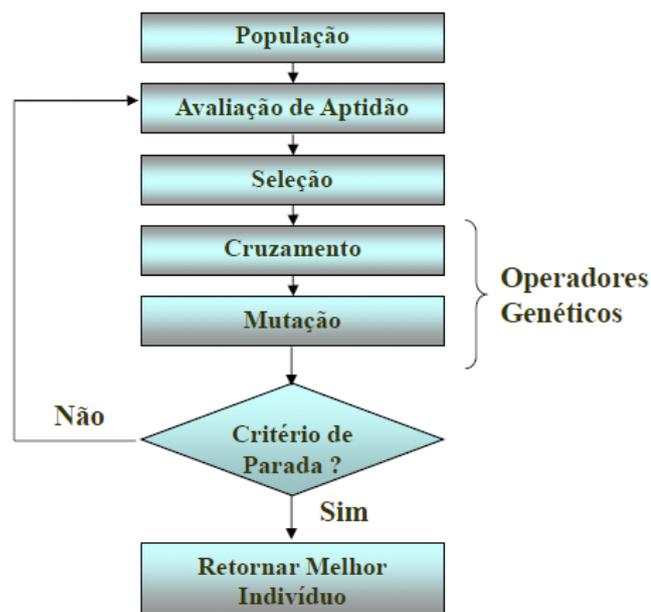


Figura 5 – Fluxograma do Algoritmo Genético.

Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010).

2.4 MICROCONTROLADORES

Microcontroladores são nada mais do que computadores miniaturizados, pois têm uma estrutura de comunicação com periféricos (entrada e saída), memória de programa, de dados e uma unidade "pensante", ou seja, o microprocessador. Por muitos anos se desenvolve novas tecnologias em cima destes componentes, gerando quase tudo o que se

encontra a nossa volta, como celular, computador, televisão, rádio, equipamentos de som e imagem, máquinas para produção de roupas, entre outros.

A plataforma Arduino junto do microcontrolador da família da ATMEL, por exemplo, tem sido utilizado na maioria dos projetos caseiros, devido a sua versatilidade e facilidade de implementação, onde qualquer pessoa é capaz de promover programas simples com esta nova plataforma. Na Figura 6, tem um exemplo de um Arduino sendo usado para projetos pessoais.

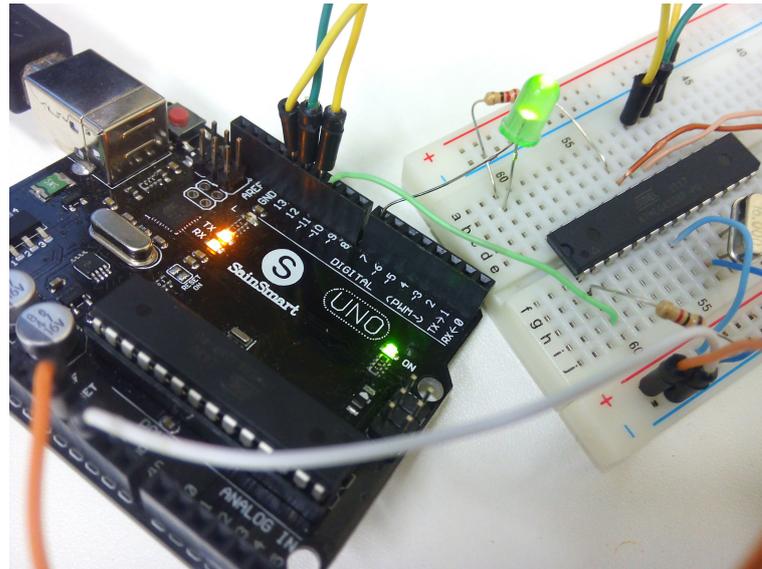


Figura 6 – Microcontrolador ATmega328p utilizado no Arduino.

Fonte: Autoria Própria.

O Arduino, para sistemas lentos, consegue realizar as operações de forma rápida e precisa, armazenando os dados necessários, se tratando de programas pouco exigente computacionalmente. O mesmo pode abrigar no seu interior um algoritmo que represente as lógicas de inteligência artificial, tanto para implementar o controlador Fuzzy (já calculado previamente), preditivo (equações e matrizes pré-definidos) ou o PI, sendo apenas necessário a implementação de uma simples conta numérica com as constantes já determinadas.

3 CONSTRUÇÃO DA INCUBADORA DE OVOS

A definição qualitativa dos parâmetros (elementos necessários) foi realizada de acordo com as especificações de (ALMEIDA, 2008) e por analogia tentativa e erro, pois a partir do protótipo é possível conhecer a real dinâmica do sistema.

A construção foi realizada visando economizar recursos financeiros, reaproveitando materiais usados, lixos eletrônicos, uso de ferramentas comuns e adaptações técnicas para suprir as necessidades da máquina.

Os principais materiais utilizados e suas respectivas finalidades estão listados abaixo:

- Resistência elétrica $\approx 200W/127V$ (Aquecimento da temperatura);
- *Cooler* interno (ventilação interna para homogeneização da temperatura interna);
- *Cooler* umidificação (forçagem de passagem do ar pela câmara com água);
- Pote com água $\approx 500ml$ (aumento da umidade relativa do ar);
- Gaveta vazada com divisórias (facilitação da respiração do ovo e rolagem pelas divisórias);
- Ovoscópio - lata + lâmpada (foco de luz para auxílio de visualização dos ovos para verificação da condição interna do embrião);
- 3 leds (indicação das condições atuais de funcionamento);
- Botão/*jumper* (zera a contagem de tempo do relógio interno);
- Botão fim de curso (detectar abertura da tampa da incubadora);
- Caixa de madeira $\approx 30x30x25cm$ (ambiente de incubação);
- Abridor de DVD (responsável por fornecer o movimento linear de rolagem dos ovos);
- Placa de fenolite e componentes eletrônicos (ligação, acionamento e leitura dos circuitos elétricos);
- Microcontrolador ATMEGA328p (circuito integrado responsável por fazer a leitura, processamento e comando);
- Fonte de alimentação 12V (fornecer energia para o sistema eletrônico);
- Fios e conectores (promover a ligação entre os componentes);
- Sensores DHT11 e DHT22 (leitura da umidade e temperatura);
- CP2102 (conversor USB/*Serial* para comunicação do computador com o microcontrolador).

Na caixa de madeira foram feitos 2 furos (na parte superior e na parte inferior) a fim de promover a circulação de ar devido a convecção natural do ar com a diferença de temperatura, pois o ar deve manter níveis normais de oxigênio, sendo o mesmo consumido pelos ovos.

A resistência foi colocada sobre um suporte e o *cooler* posicionado com a saída de ar incidindo sobre a resistência, a fim de misturar o ar frio com o ar quente.

Toda a construção da incubadora está mostrada no Apêndice A, mostrando todas as etapas de cada processo, inclusive da fabricação das PCBs (placas de circuito impresso). A Figura 7 mostra a incubadora finalizada com todos os elementos instalados e já em operação (também programada).



Figura 7 – Incubadora de ovos finalizada.

Fonte: Autoria própria.

4 PROJETO DOS CONTROLADORES

4.1 IDENTIFICAÇÃO DO SISTEMA

Em (OGATA, 2001), pode-se analisar um sistema térmico como um circuito elétrico formado por: uma fonte (aquecedor fornecendo energia de entrada), um capacitor (capacidade de absorver calor da incubadora) e um resistor (determina as perdas do sistema térmico para o meio e por sua vez a velocidade do mesmo). Sendo representado por um sistema de primeira ordem, ou seja, função transferência de *Laplace* igual a um polo e um zero.

Foram coletadas duas amostras, degrau máximo de potência da resistência (127V ligado direto) e um controle discreto para estabilizar próximo à temperatura de referência, que se irá trabalhar. Para uma entrada degrau de 1 (significando 100% de PWM = 127V), como na Figura 8, tem-se uma saída em forma de "s", apresentando como máximo a máxima diferença de temperatura que a incubadora pode possuir com a temperatura externa, visto que deve ser transladada a temperatura medida para iniciar em zero (temperatura inicial), obtendo assim a característica da mesma pelo ganho de temperatura.

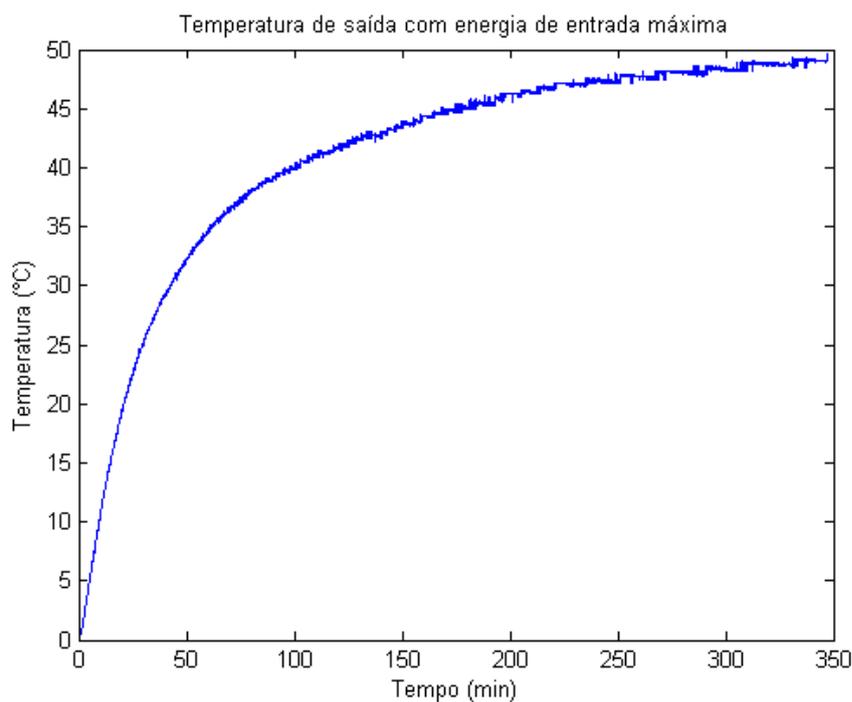


Figura 8 – Entrada degrau e saída em "s" transladada para zero.

Fonte: Autoria Própria.

Na entrada com controlador discreto semiajustado, ou seja, com ganhos determinados por tentativa e erro, desejou-se obter uma saída oscilatória. Coletando todas as entradas e saídas, foi possível usar esse sistema para estimar a função de transferência linearizada próxima do ponto de operação do sistema. A Figura 9, mostra a resposta da temperatura à entrada controlada.

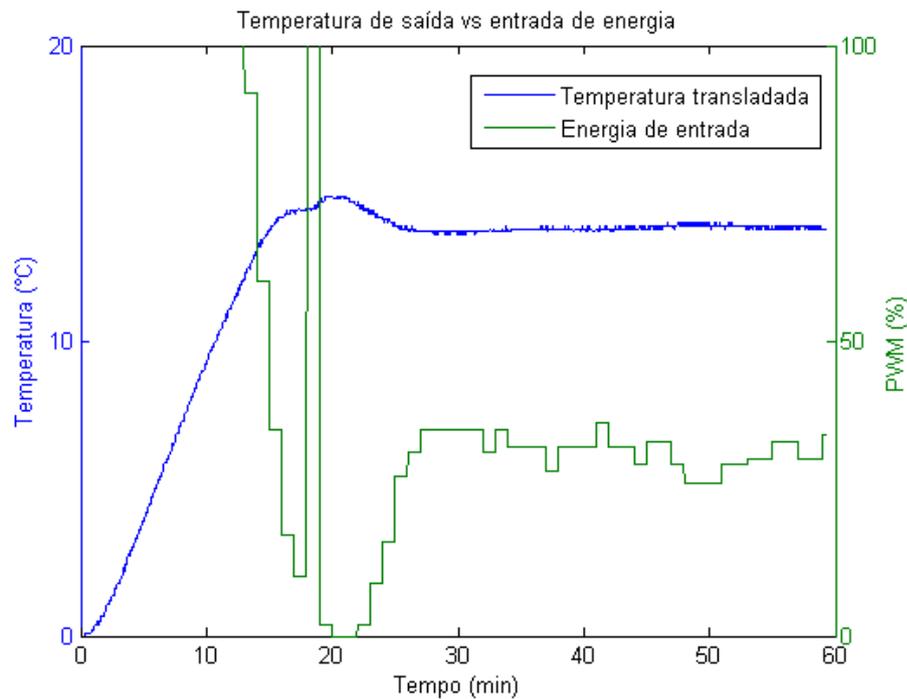


Figura 9 – Entrada discreta controlada e saída próximo à referência.
Fonte: Autoria Própria.

Utilizou-se a função *tfest* do *Matlab*[®] para identificar a função de transferência de primeira ordem (tf1) do sistema, obtendo a Figura 10, cujo valor do ajuste (fit) foi de 87,8% e está representado pela equação 6.

$$tf1(s) = \frac{0,0216}{s + 0,0004442} \quad (6)$$

Da mesma forma identificou-se a função transferência de segunda ordem (tf2) da mesma entrada ao degrau, obtendo a Figura 11, cujo valor do ajuste (fit) foi de 98% e está representado pela equação 7. Aumentou-se a ordem do sistema (polos e zeros), pois verificou-se na prática que este tipo de função representa melhor o sistema como um todo. Esta identificação, por estar com o degrau em valor máximo da saída, apresenta uma boa precisão com a incubadora trabalhando próximo ao ponto de estabilidade máxima. Mostrando também o maior ganho de temperatura que pode ser dado pela resistência.

$$tf2(s) = \frac{0,02422s + 5,208e - 06}{s^2 + 0,000893s + 1,031e - 07} \quad (7)$$

Porém, como o sistema real possui imprecisões e não linearidades, fez-se uma identificação utilizando os dados de entrada e saída presentes no controle discreto parcial (PI de teste), obtendo a função transferência (tf3) dada pela equação 8. Onde a mesma utilizou-se de um sistema de segunda ordem (2 polos e 1 zero), apresentando maior fidelidade no sistema, a qual pode ser percebida na Figura 12, cujo valor de ajuste é de 97,96%.

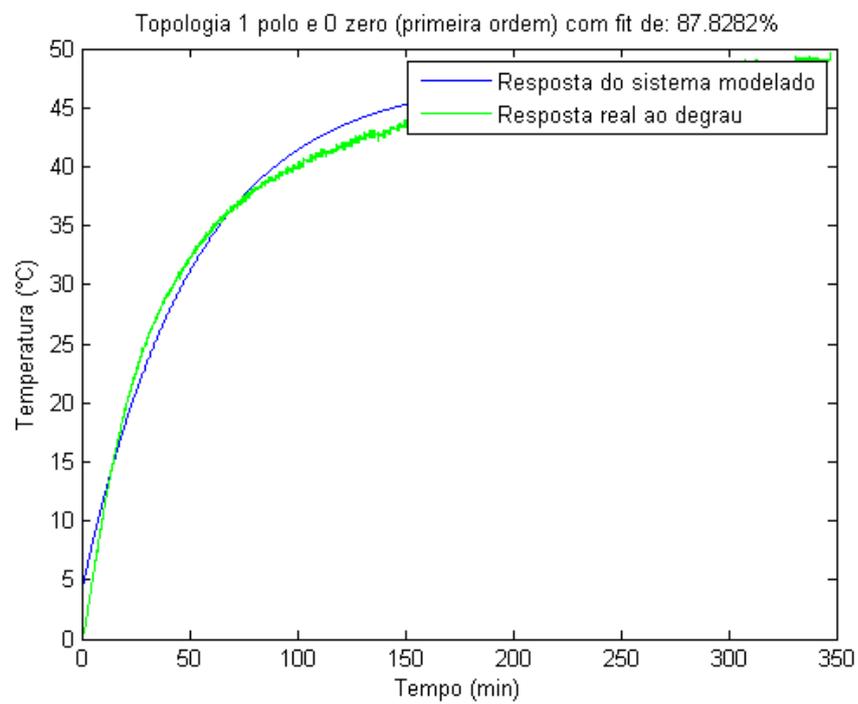


Figura 10 – Entrada Degrau e *fit* de 1 polo e 0 zeros.

Fonte: Autoria Própria.

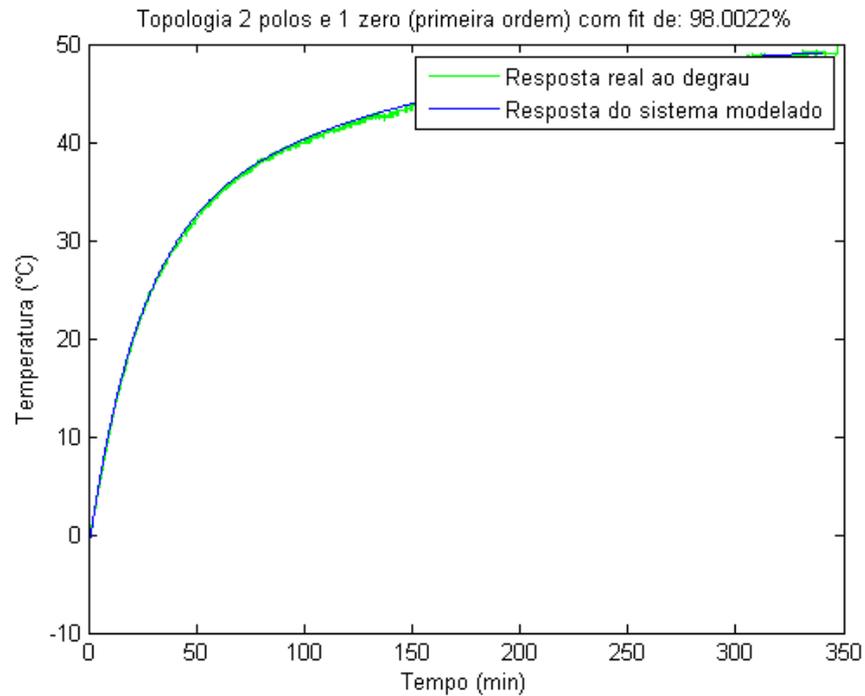


Figura 11 – Entrada Degrau e *fit* de 2 polo e 1 zeros.

Fonte: Autoria Própria.

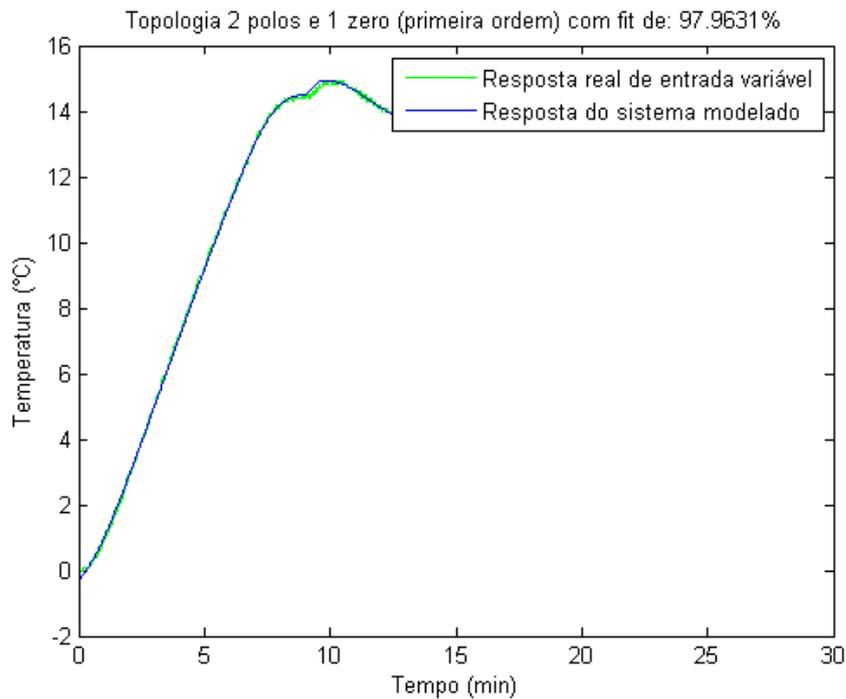


Figura 12 – Entrada discreta parcialmente controlada e fit de 2 polo e 1 zeros.

Fonte: Autoria Própria.

$$tf3(s) = \frac{0,01109s + 0,0005504}{s^2 + 0,01363s + 1,217e - 05} \quad (8)$$

Para a primeira estimativa do PI, fez-se a modelagem como sistema de primeira ordem, com um atraso, gerando a equação 9 e a Figura 13 mostrando seu desempenho, que apresenta pontas (picos ingrimes diferentes do modelo real) mas serve como uma aproximação.

$$tf4(s) = \frac{46,188e^{-55,785s}}{1134,4s + 1} \quad (9)$$

4.2 CONTROLE PI

Foi escolhido primeiramente o controlador PI no lugar do PID, excluindo a parcela derivativa, devido aos ruídos presentes no leitor de temperatura, que poderiam causar uma oscilação do sistema, pois a parcela derivativa amplifica ruídos de alta frequência. Também por motivo de ser um sistema bem lento, sendo todas as derivadas de variação relativamente baixas (sem picos de alteração da temperatura).

Mesmo o sistema sendo contínuo, o controle é feito de forma discreta, mas foi calculado primeiro os ganhos pelo método de *Ziegler-Nichols* em regime contínuo (resposta aproximada do discreto com tempo de amostragem relativamente curto). Porém, para se fazer o ajuste fino, usou-se algoritmos genéticos para sintonizar os ganhos k_p e k_i a fim de

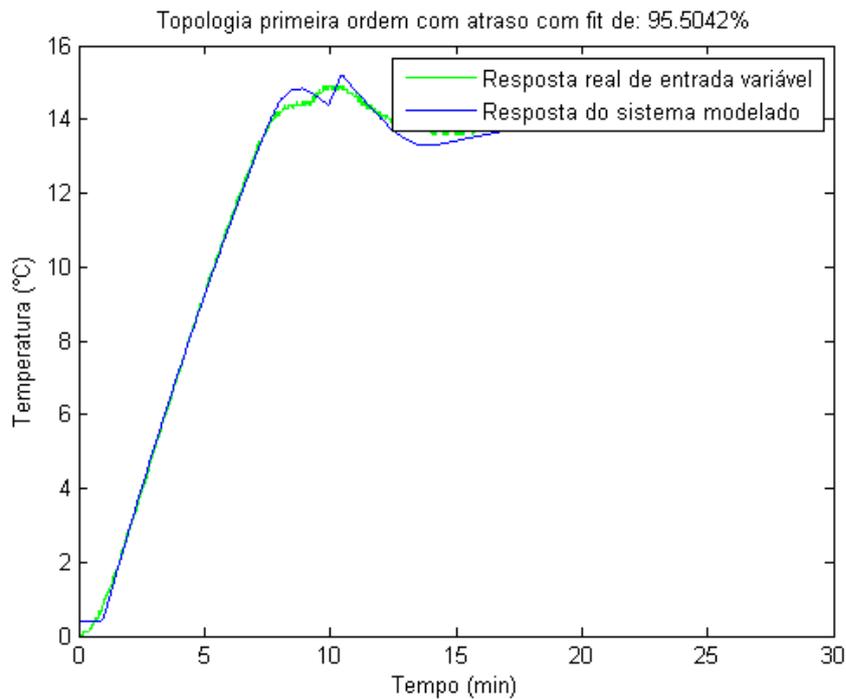


Figura 13 – Entrada discreta parcialmente controlada e fit de 1 polo e 0 zeros com atraso.
Fonte: Autoria Própria.

melhorar a resposta em regime permanente e uma atuação mais rápida (menor tempo fora da temperatura de referência).

Por se tratar de um sistema com saturação de entrada, ou seja, a resistência que eleva a temperatura não pode assumir valores negativos e nem além da sua capacidade máxima (ligada diretamente em 127V sem interrupção de tempo). É colocado então um saturador, tanto na simulação quanto na programação, a fim de limitar a entrada de energia. O controlador por sua vez pode, devido ao integrador, acumular valores de energia acima do limite máximo ou abaixo do mínimo. Para corrigir este problema, adicionou um termo chamando *anti-windup*, como proposto por (BOHN; ATHERTON, 1995). Que por sua vez, subtrai do integrador uma parcela proporcional (de constante ka) à diferença de saída do controlador e da saída do saturador, atuando então quando forem extrapolados os limites do mesmo.

Como é dado em (OGATA, 2001), tem-se que uma planta analisada com entrada degrau e que apresenta uma curva em "s" (primeira ordem com atraso), podem ser calculados os ganhos kp e ki através das fórmulas na Tabela 2, onde $ki = \frac{kp}{Ti}$. Usando os valores obtidos na equação 9 e comparando com a equação característica do sistema, tem-se a equação 10.

$$tf4(s) = \frac{46,188e^{-55,785s}}{1134,4s + 1} = \frac{Ke^{-Ls}}{Ts + 1} \quad (10)$$

Assim determinou-se os valores: $kp = 18,3$ e $ki = 0,09$. Estes serão os valores bases para serem colocados no algoritmo genético, a fim de realizar o ajuste fino. Porém, como o sistema é real e utiliza a constante ka *anti-windup* subtraindo o integrador de acordo

Tabela 2 – Ajuste dos ganhos para entrada degrau de Ziegler-Nichols.

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{T}{L}$	$2L$	$0.5L$

Fonte: (OGATA, 2001).

com a diferença entre o valor de saída do controlador e o valor de saída após a saturação, será atribuído o valor base de ka como metade do valor de ki , pois ambos interagem com o integrador, mas são proporcionais a erros diferentes. Num teste rápido verificou-se esta proporção.

O Algoritmo genético procura minimizar o erro apresentado em regime permanente pra cada controlador estudado, assim como o ajuste é estocástico, então os limites máximos e mínimos para cada valor será o valor base $\pm 100\%$, ou seja, de zero até o dobro do valor encontrado, pois são muitas as combinações. Assim, tem-se o número da população de 200 indivíduos, 20 gerações, 10% de eliminação e 20% de mistura de informações, não trabalhando com a mutação. Foi utilizado a *toolbox* do Matlab[®] para a realização do procedimento.

A avaliação de aptidão é feita através de uma função a parte, que irá ser minimizada pelo algoritmo, neste caso a mesma minimiza o quadrado do erro em relação à temperatura de referência. Esta função é chamada dentro do algoritmo genético e retorna o valor de aptidão, no caso, a somatória dos erros (sempre no mesmo período de tempo analisado), gerando assim a área de erro quadrática, onde o valor é sempre positivo e quanto pior o controlador (maior lentidão de partida e maiores oscilações), maior será o valor da área quadrática. Fez-se necessário realizar uma adaptação na função a fim de penalizar as oscilações que persistiam no tempo, multiplicando o tempo instantâneo pelo erro, ponto a ponto, aumentando o peso dos erros que perpetuam ao longo da simulação.

A simulação utilizada na função de aptidão é o modelo mais próximo do real, ou seja, possui a função transferência (tf3) que melhor representa o sistema próximo do ponto de operação ($36,5^\circ\text{C}$), trabalha com os cálculos discretizados, assim como o microcontrolador, e o tempo de discretização está ajustado para ser o mesmo do real (1 minuto). Os resultados da simulação são transferidos para o Matlab[®], podendo ser usado pela função para achar os valores a serem minimizados. A Figura 14, mostra o esquema para a simulação da planta pelo *Simulink*[®].

Aplicando os valores encontrados de k_p e k_i pelo método ZN (*Ziegler-Nichols*) e fazendo $ka = ki/10$ (dividido por 10 foi o melhor número testando na prática, atribuindo valores aleatórios), obtêm-se as Figuras 15 e 16, que relatam respectivamente a temperatura interna na incubadora e as energias (antes e depois do saturador). O mesmo teve um

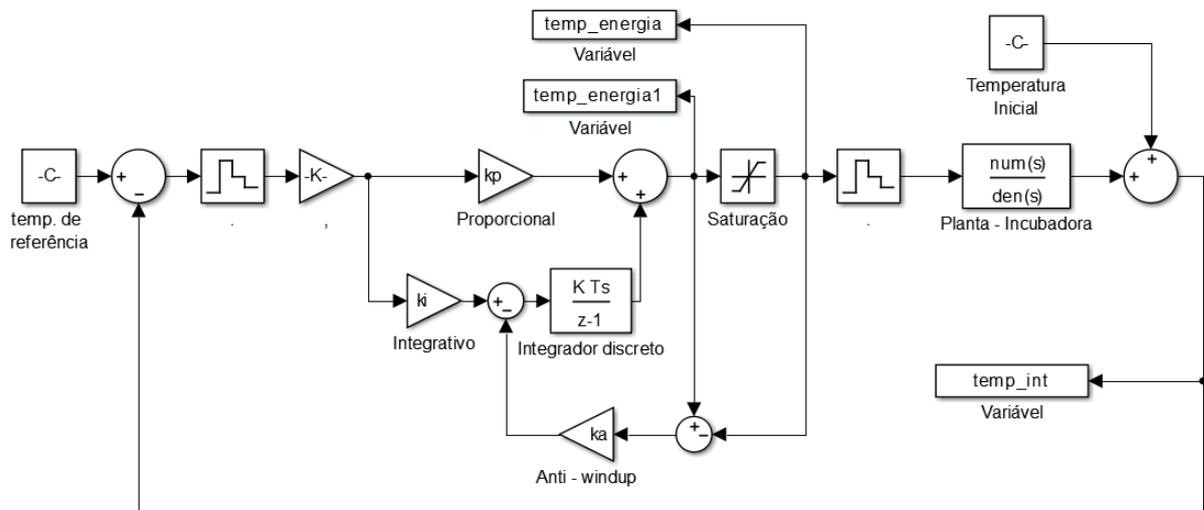


Figura 14 – Simulação do controle PI discreto.

Fonte: Autoria Própria.

resultado muito satisfatório, apesar de se definir o ka manualmente, foi possível chegar à um valor que apresentasse uma resposta boa. Percebe-se a atuação do saturador quando o mesmo corta a energia, limitando em 1 = 100% = máximo e 0% = mínimo de PWM da tensão sobre a resistência.

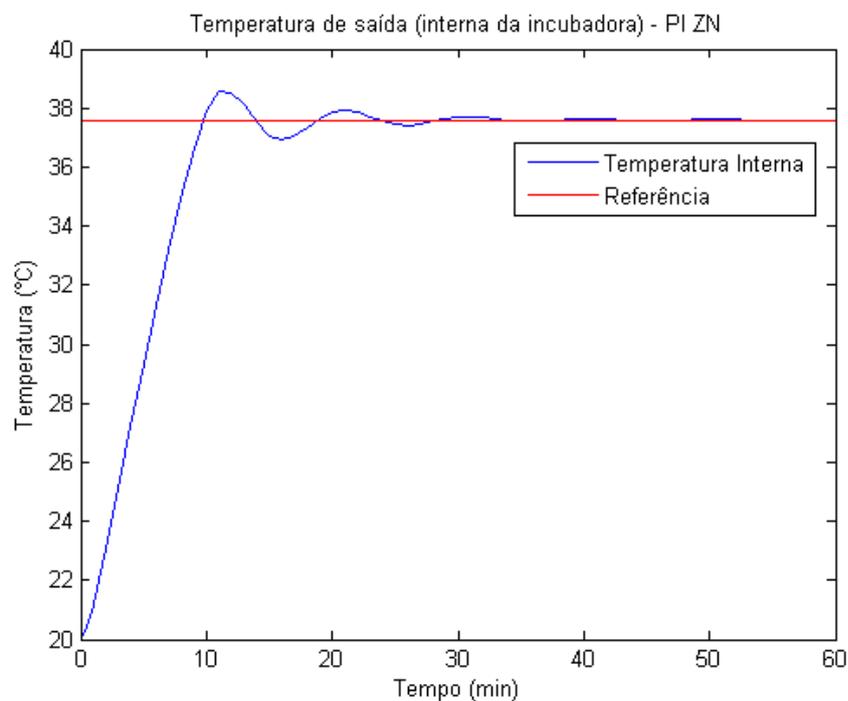


Figura 15 – Temperatura de saída pelo método ZN.

Fonte: Autoria Própria.

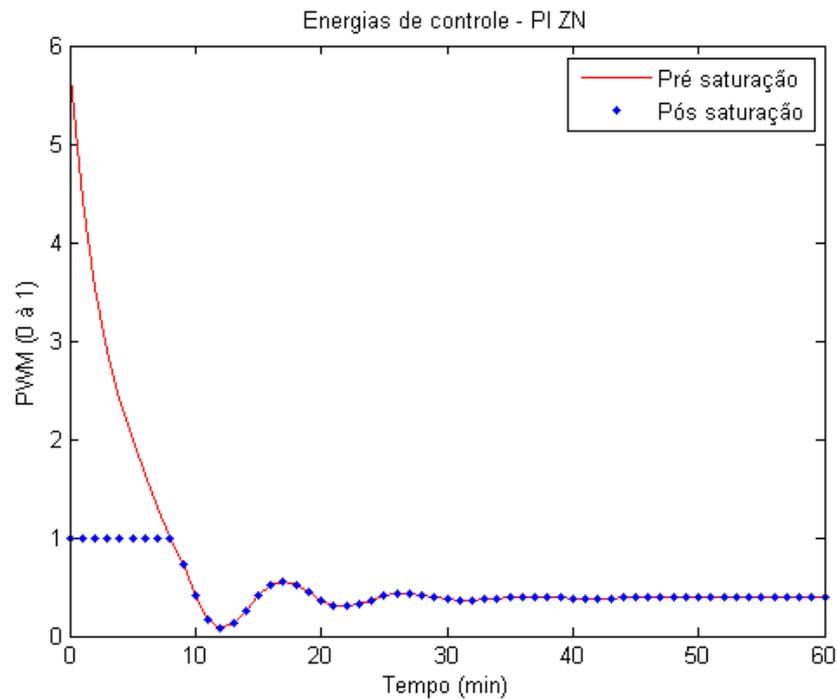


Figura 16 – Energias pelo método ZN.

Fonte: Autoria Própria.

Aplicando o AG (algoritmo genético), obtém os valores otimizados: $k_p = 22,2758$, $k_i = 0,11143$ e $k_a = 0,013403$, sendo os mesmos diferentes dos encontrados por ZN, mas que por sua vez apresentaram uma melhora de apenas 1,5% (se tratando da análise do quadrado da área), mas visualmente, percebe uma redução significativa da ultrapassagem da temperatura sobre a referência, sendo esta redução de 85%. Também com uma estabilização próxima da referência mais precoce. A Figura 17, mostra os resultados dos indivíduos gerados pelo AG, convergindo para um número cada vez menor, ou seja, encontrando os parâmetros que melhor desempenham a função de ajuste escolhida. As Figuras 18 e 19, mostram a análise das respostas do sistema (simulado) para a mesma referência e temperatura externa.

O método de AG poderia encontrar os mesmos resultados, partindo de uma sintonia manual, sem a necessidade de partir do método ZN, delimitando os limites de busca através de conhecimentos práticos (estocásticos) do sistema. A comparação dos dois métodos de controle PI com seus respectivos ajustes é dado pela Figura 20.

4.3 CONTROLE FUZZY

Fuzzy significa "nebuloso", pois o conhecimento de um especialista é passado para a máquina para que a mesma atue de maneira semelhante ao ser humano. Como neste projeto não estudou-se um controle manual de que possa ser tomado como base, criou-se uma lógica para implementar o sistema inteligente de uma forma pouco diferente do convencional, pois deseja-se fazer um Fuzzy apenas para corrigir o erro da temperatura

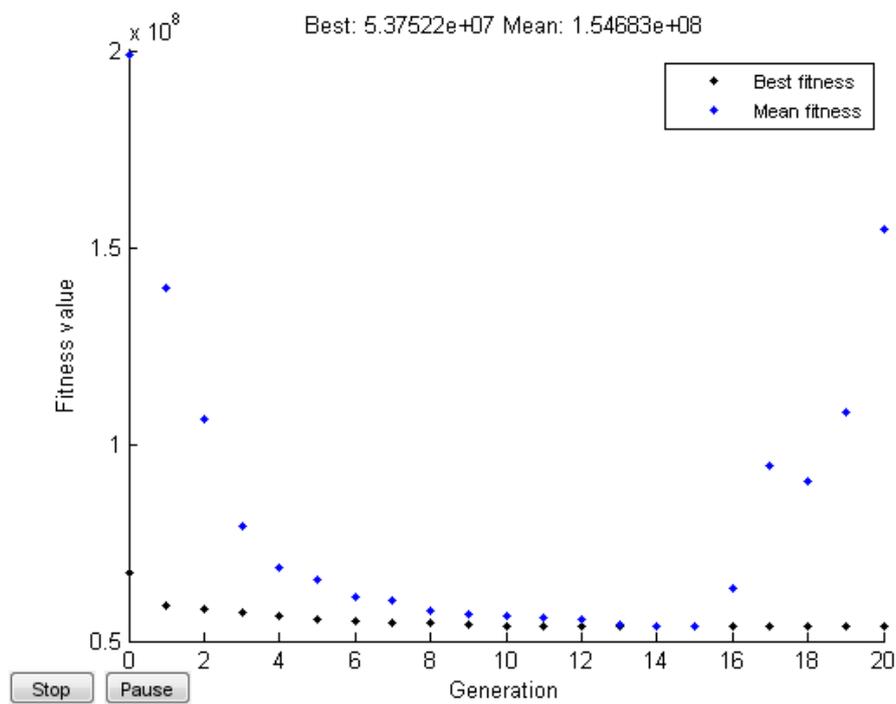


Figura 17 – Convergência dos resultados analisados no processo do AG.
Fonte: Autoria Própria.

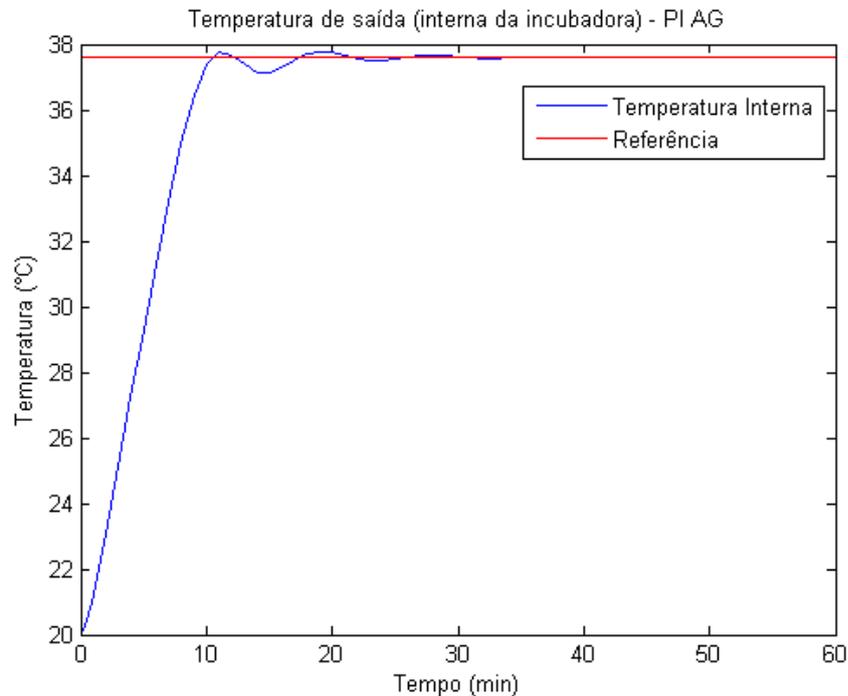


Figura 18 – Temperatura de saída pelo método AG.
Fonte: Autoria Própria.

interna com a referência. Mas existe outra variável que interfere na temperatura, esta é a diferença de temperatura entre o ambiente interno e externo, causando a diminuição da

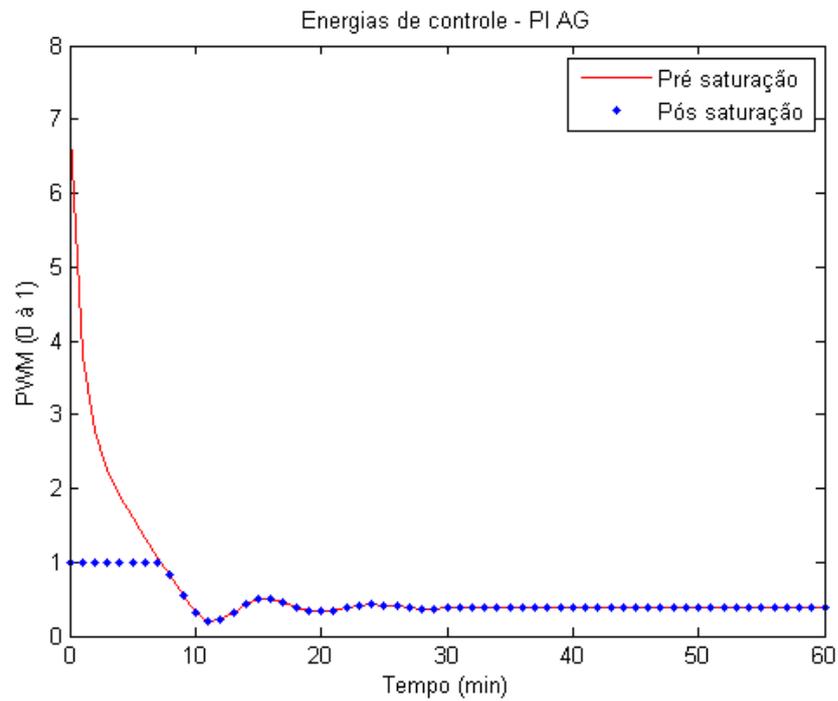


Figura 19 – Energias pelo método AG.
Fonte: Autoria Própria.

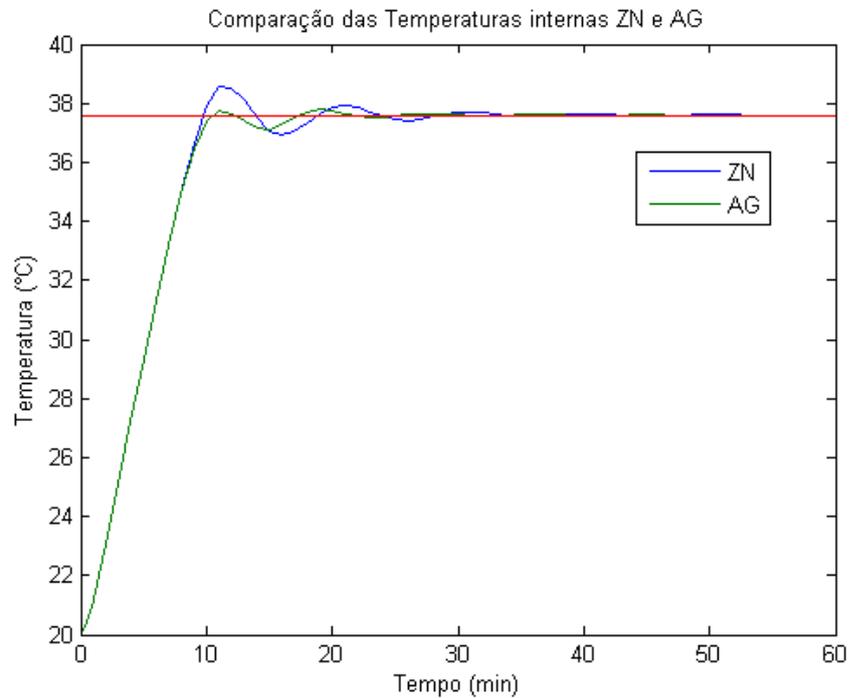


Figura 20 – Comparação dos métodos de Ajuste ZN e AG.
Fonte: Autoria Própria.

temperatura interna devido à convecção do ar, que circula livremente e proporcionalmente pela incubadora.

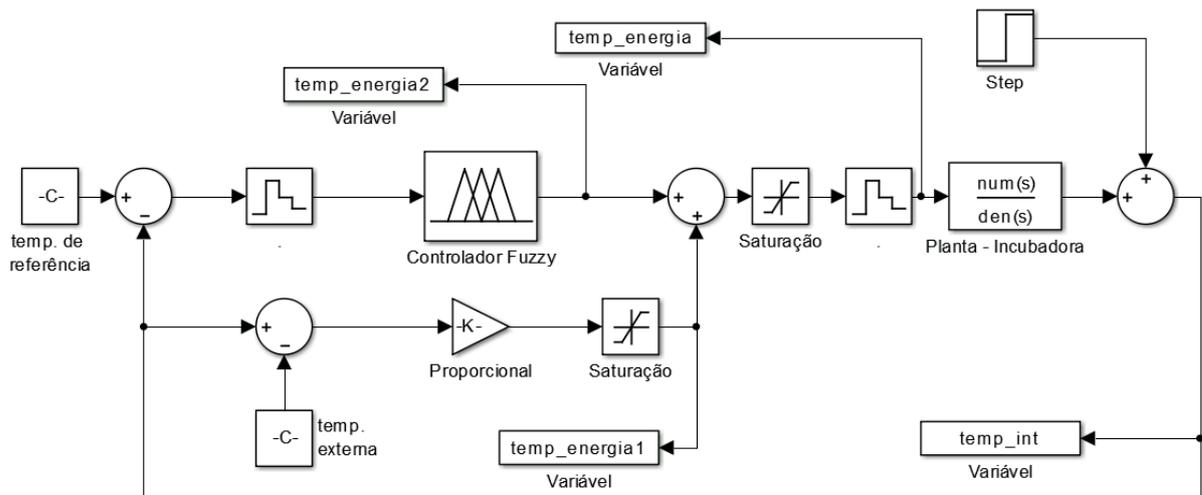


Figura 21 – Simulação do controlador Fuzzy aplicado à planta.

Fonte: Autoria Própria.

Como a convecção do ar é proporcional à diferença de temperatura, é possível calcular a relação (aproximadamente linear) da diferença de temperatura pela saída em PWM, o qual a mesma será usada para ser um controle proporcional paralelo ao Fuzzy, assim o Fuzzy se encarrega apenas de ajustar a temperatura interna de acordo com a referência, deixando a parcela de decaimento para o controlador proporcional, o que causa uma redução na oscilação da resposta e elimina a necessidade de usar valores diferenciais, que da mesma forma que no PI, causaria uma interferência devido à oscilação na leitura do sensor. Além do controle puramente Fuzzy, ter uma maior complexidade. A equação 11 mostra o cálculo do ganho proporcional auxiliar (k), sendo os valores pegos do PI, aplicado experimentalmente na planta próximo ao ponto de linearização.

$$k = \frac{PWM}{\Delta T} = \frac{0,39}{16,7} = 0,022. \quad (11)$$

Conhecendo a energia necessária para manter uma certa temperatura no interior da chocadeira, cabe ao Fuzzy controlar apenas as variações da temperatura interna, olhando para a referência e usando o erro como entrada do sistema inteligente, o qual também se comportará como um controlador proporcional, ofertando em sua saída um sinal que irá até 1 (100% PWM) e um número negativo, ajustado pelo AG. A configuração proposta está representada na Figura 21.

Fez-se um protótipo usando a (*toolbox*) do Fuzzy e ajustou-se manualmente (*script*), de acordo com os valores do PI como suporte e posteriores simulações para complementar a resposta. Consecutivamente, otimizadas as funções de pertinências com o AG, e as regras mantidas fixas. As funções pertinências de entrada e saída estão nas Figuras 22 e 23. Gerando a curva (resposta de acordo com a entrada), as regras de ativação das funções (par entrada/saída) e a visualização das regras (verificação dos valores e como ocorre a distribuição das respostas) nas respectivas Figuras: 24, 25 e 26.

Simulando então o Fuzzy proposto, ajustado manualmente, encontra-se a res-

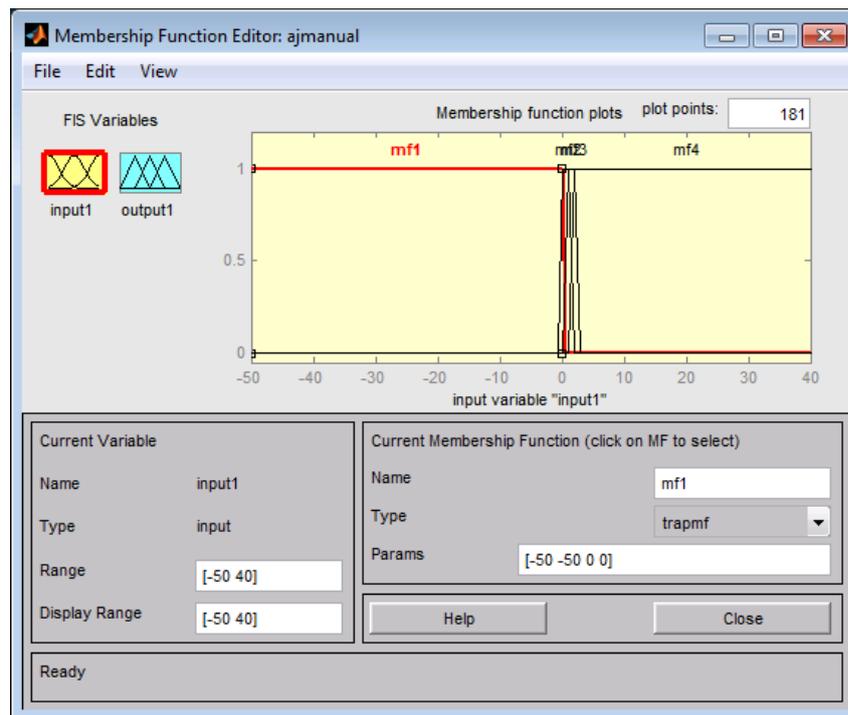


Figura 22 – Funções pertinências da entrada manual (input).

Fonte: Autoria Própria.

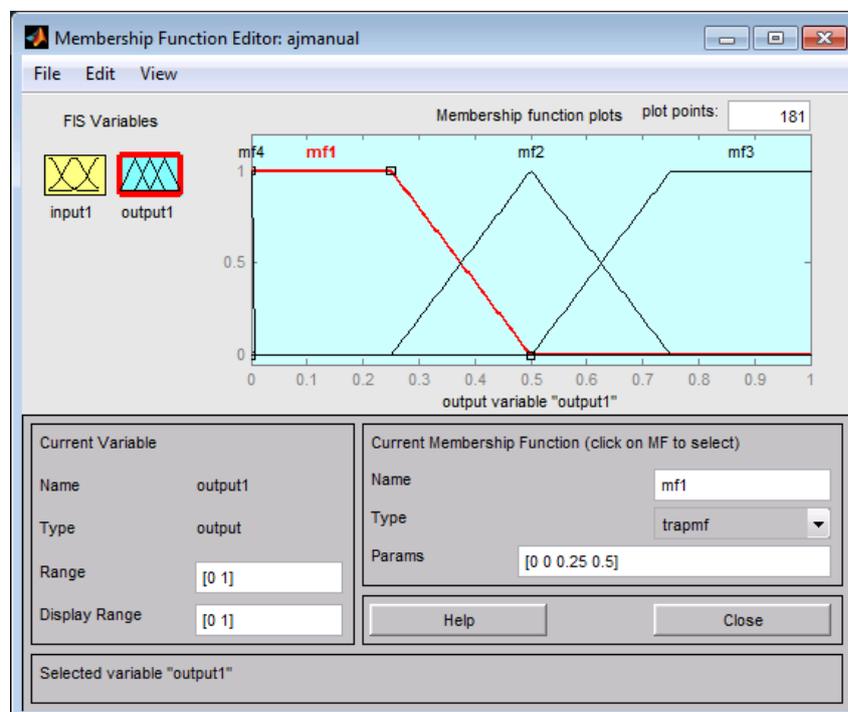


Figura 23 – Funções pertinências da saída manual (output).

Fonte: Autoria Própria.

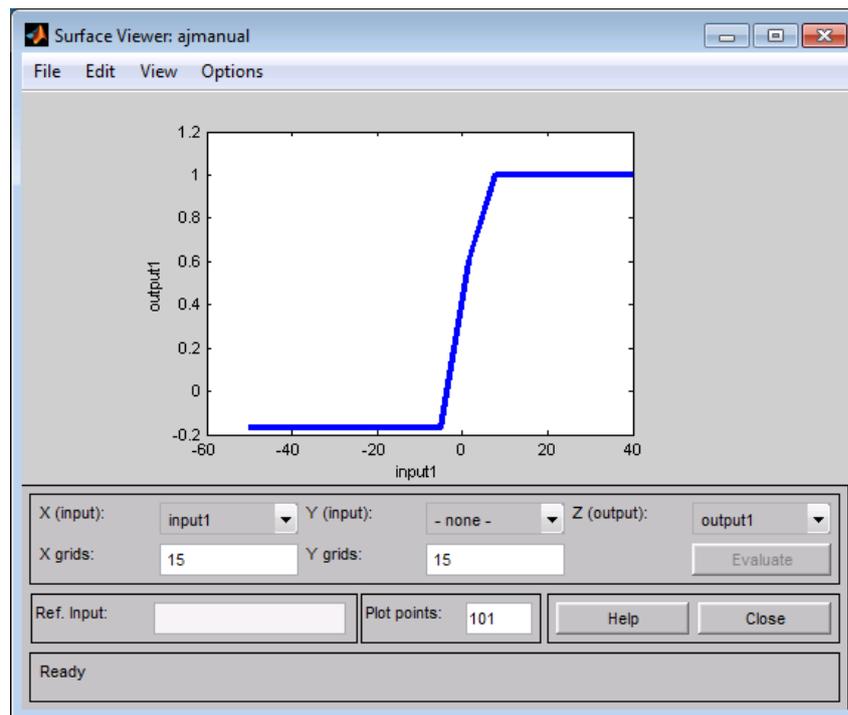


Figura 24 – Curva/superfície para o Fuzzy manual.

Fonte: Autoria Própria.

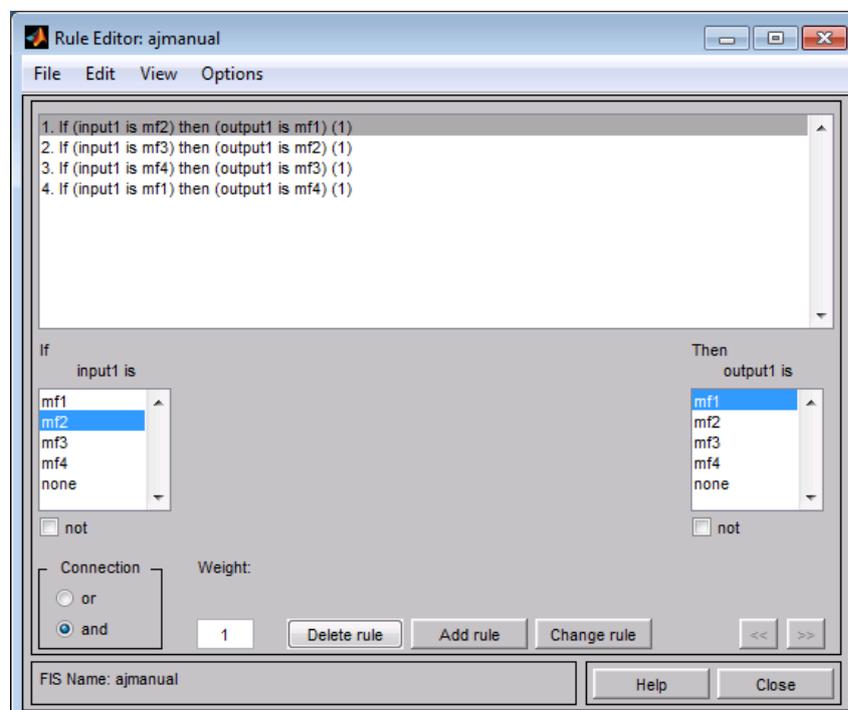


Figura 25 – Regras para o Fuzzy manual (Funções de entrada que ativam as funções de saída).

Fonte: Autoria Própria.

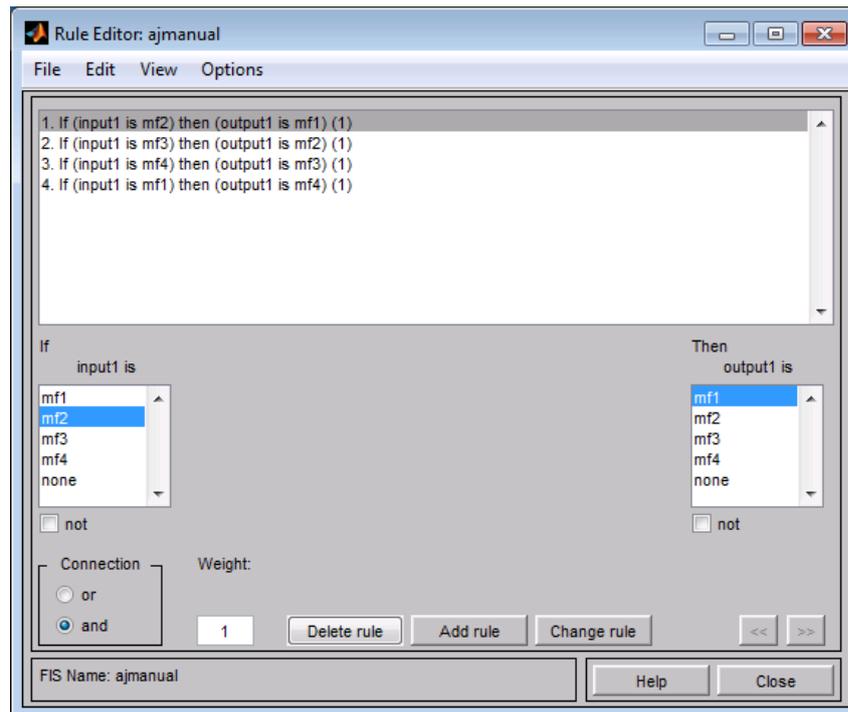


Figura 26 – Visualização das regras para o Fuzzy manual.

Fonte: Autoria Própria.

posta da temperatura interna na Figura 27, sua respectiva energia na Figura 28, sendo a energia final composta pela parcela proporcional devido à temperatura externa e a outra parcela a atuação do sistema Fuzzy.

Agora fazendo o ajuste otimizado com AG dos valores das funções pertinências e mantendo as regras iguais, tem-se os resultados: Convergência do AG, função pertinência de entrada, de saída, curva Fuzzy, visualização das regras, temperatura interna e energias em suas respectivas Figuras: 29, 30, 31, 32, 33, 34 e 35.

Percebe-se uma melhora na resposta do Fuzzy otimizado, mas que fica melhor visualizado na Figura 36 que compara o PI ajustado com AG (melhor do controle convencional) com os dois controladores Fuzzy. As energias de controle estão com menores oscilações e a resposta com menor pico de ultrapassagem da referência e menor tempo de estabilização, diminuindo também a amplitude das oscilações.

4.4 CONTROLE PP

Após a realização do controle Fuzzy, usando um controlador proporcional auxiliar, responsável por fornecer a energia necessária para o sistema manter as suas perdas, percebeu-se que o Fuzzy atuava apenas como um controlador proporcional ao erro da temperatura interna com a referência. Logo, fez-se outro modelo de controle, trocando o Fuzzy por um controlador proporcional e mantendo o controle das perdas, já conhecido, resultando num sistema proporcional - proporcional (PP).

O controle proposto tem sua regulação ao erro apenas com um elemento proporcional, sendo sua saída muito sensível à variações por imprecisões na leitura (sensor de

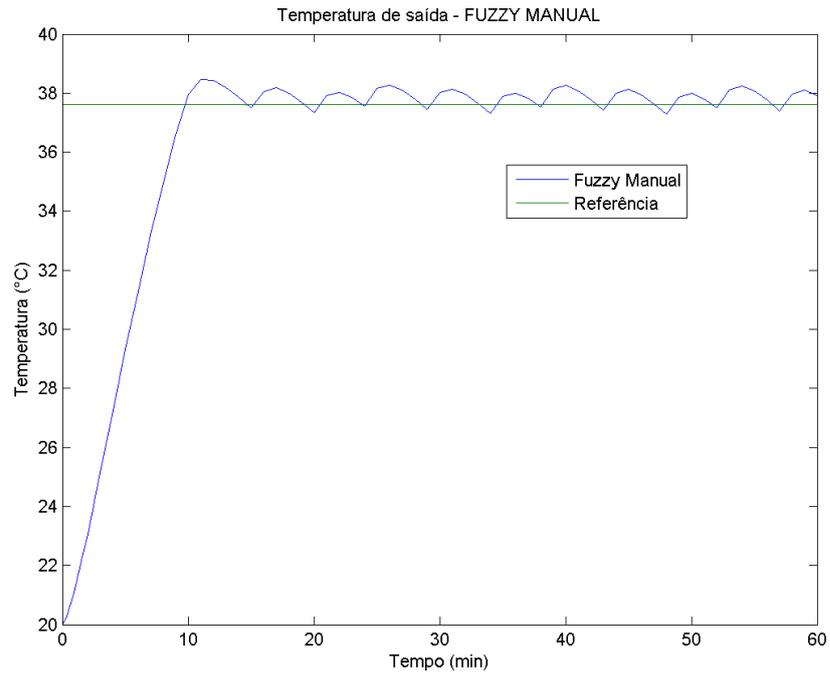


Figura 27 – Temperatura interna com Fuzzy ajustado manualmente.
Fonte: Autoria Própria.

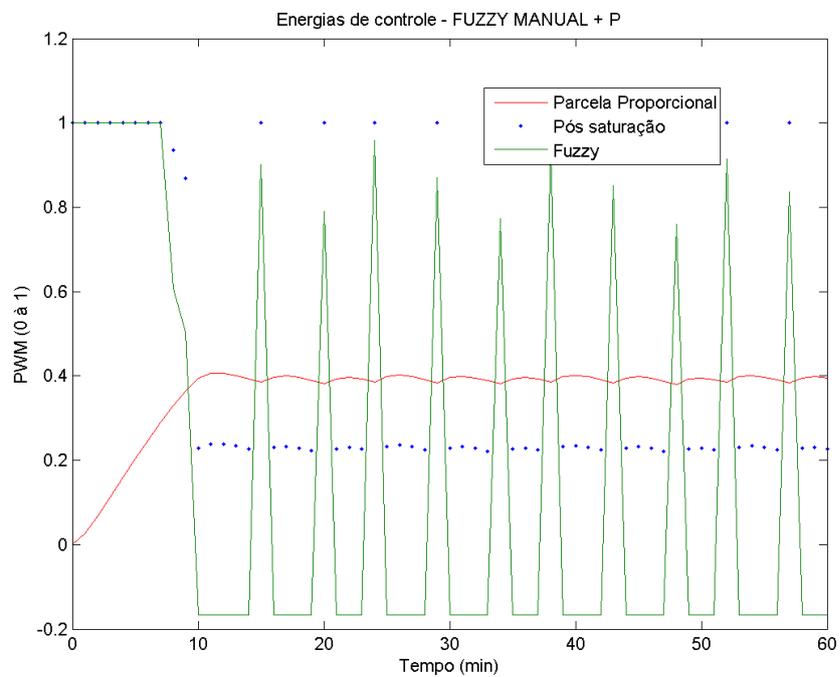


Figura 28 – Energias de controle Fuzzy ajustado manualmente.
Fonte: Autoria Própria.

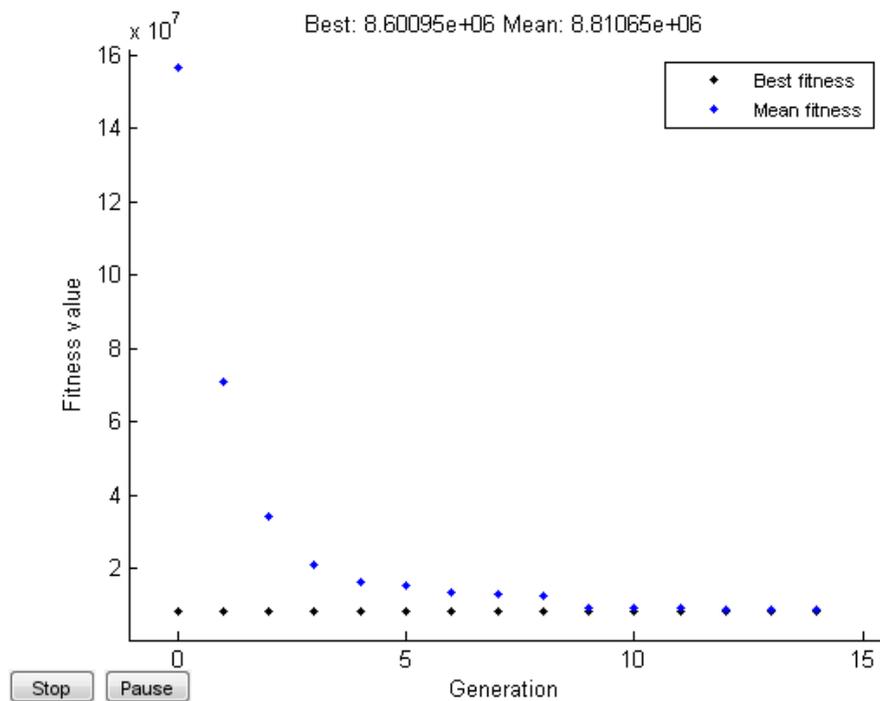


Figura 29 – Convergência do processo de otimização pelo AG.

Fonte: Autoria Própria.

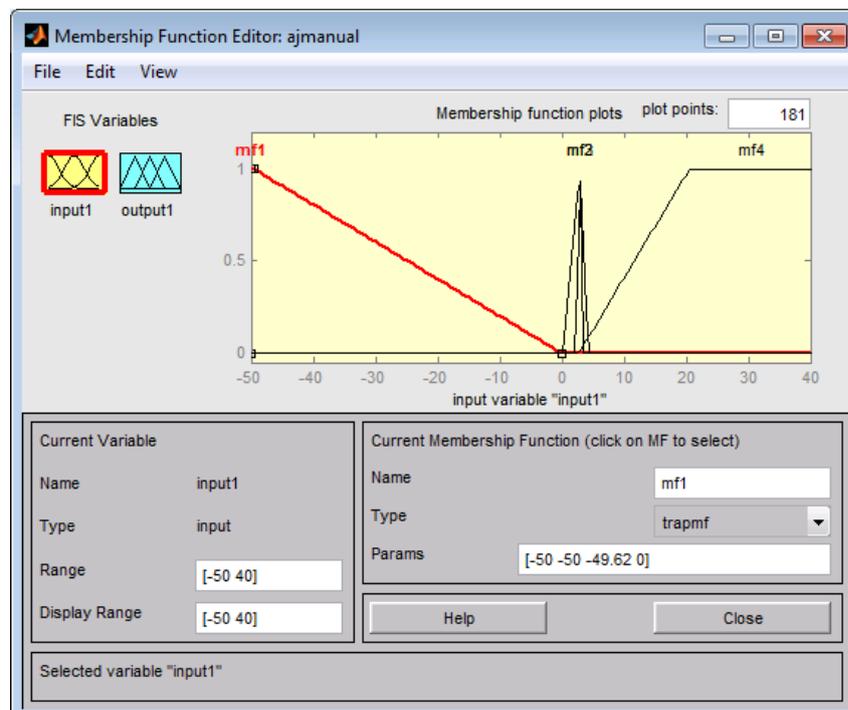


Figura 30 – Funções pertinências da entrada manual (input).

Fonte: Autoria Própria.

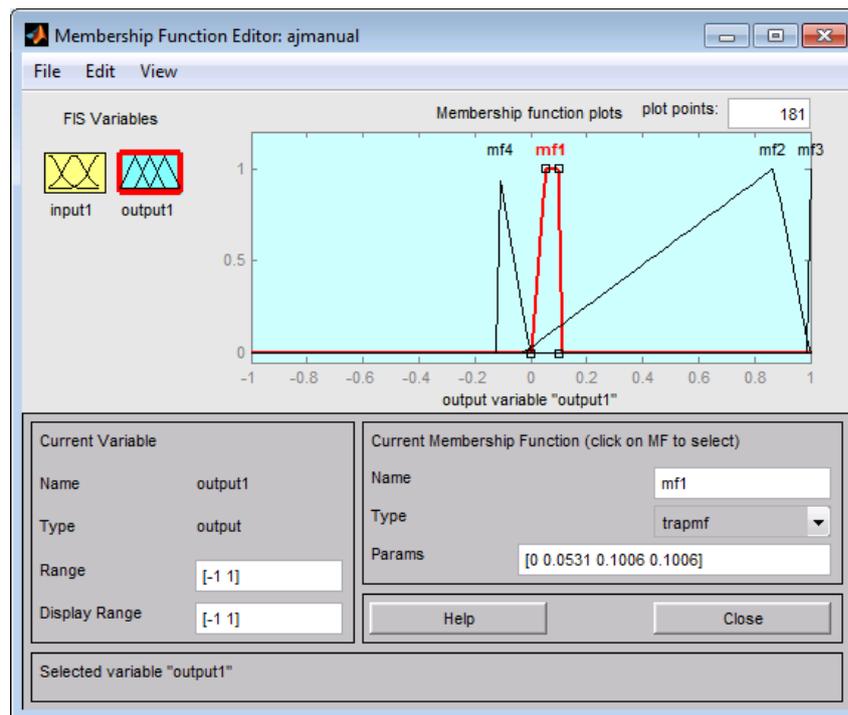


Figura 31 – Funções pertinências da saída manual (output).

Fonte: Autoria Própria.

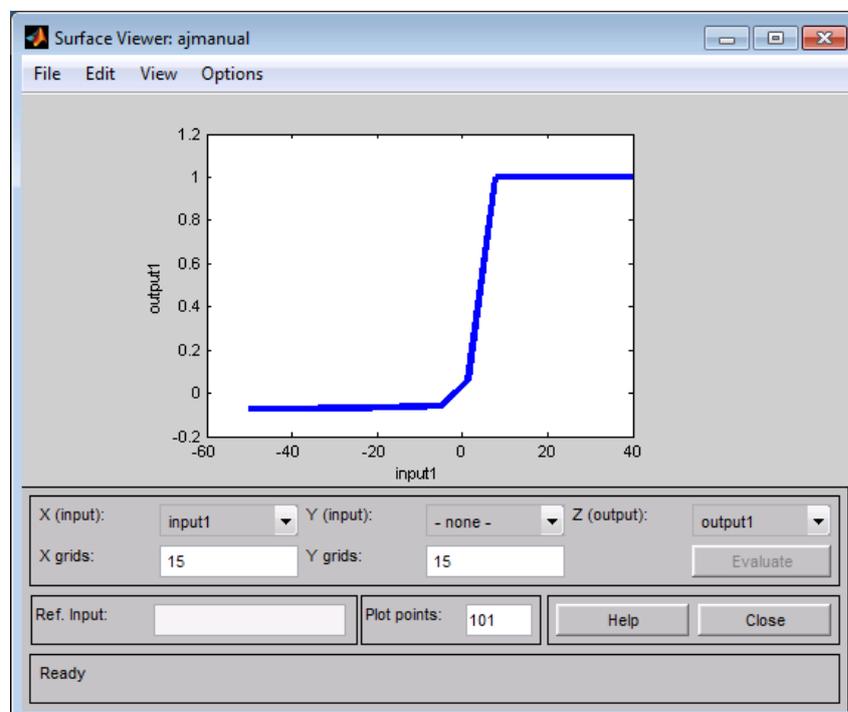


Figura 32 – Curva/superfície para o Fuzzy manual.

Fonte: Autoria Própria.

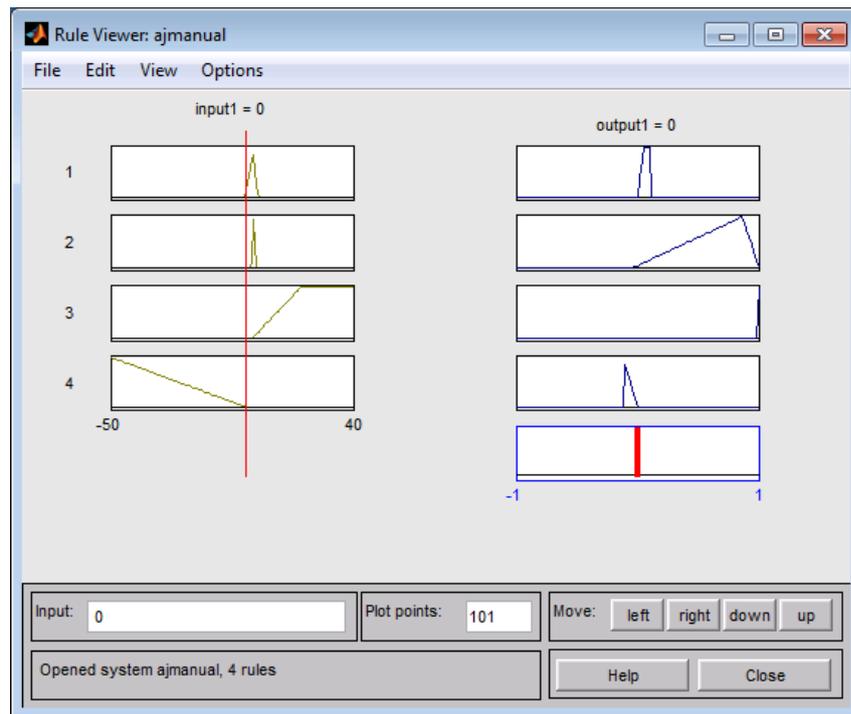


Figura 33 – Visualização das regras para o Fuzzy manual.
Fonte: Autoria Própria.

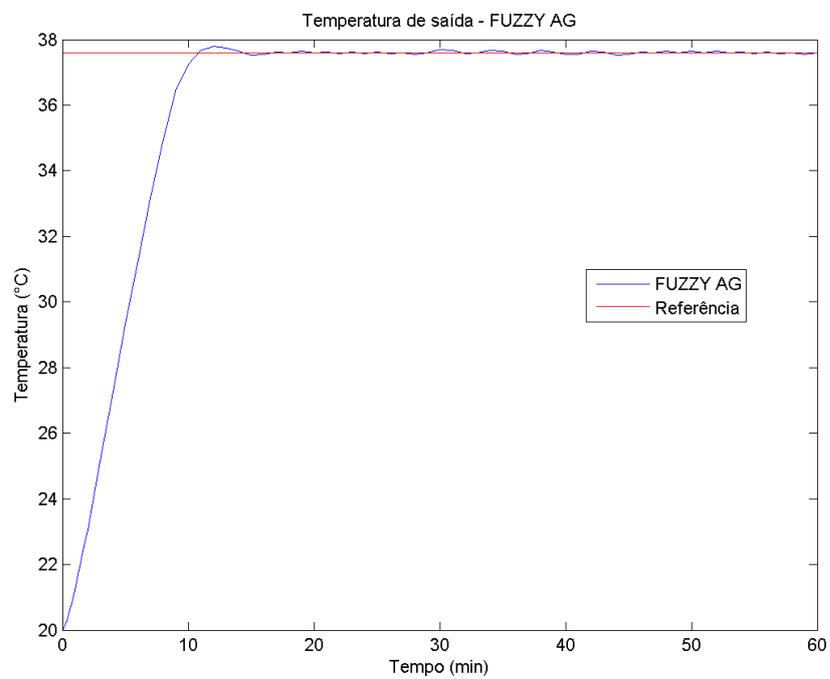


Figura 34 – Temperatura interna com Fuzzy ajustado com AG.
Fonte: Autoria Própria.

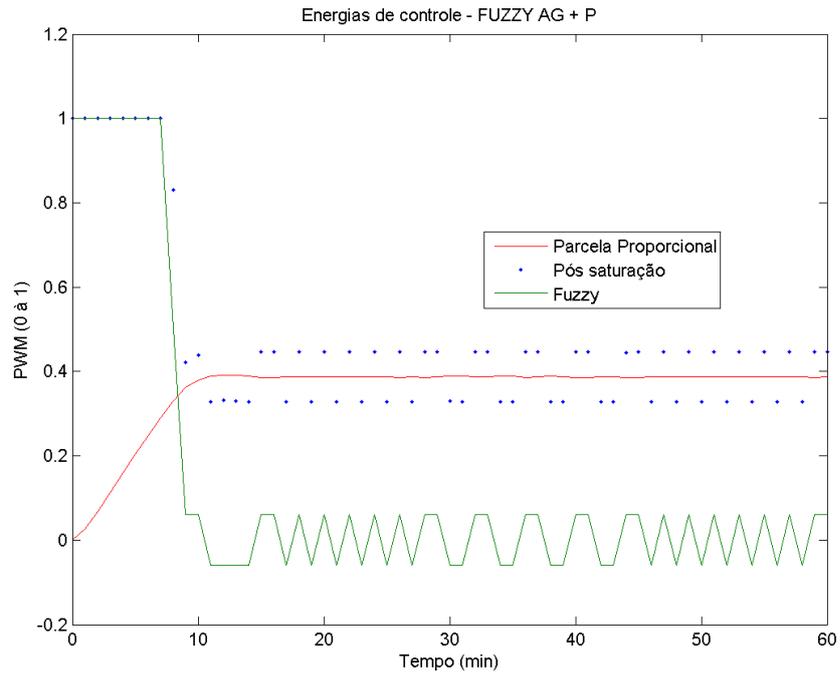


Figura 35 – Energias de controle Fuzzy ajustado com AG.

Fonte: Autoria Própria.

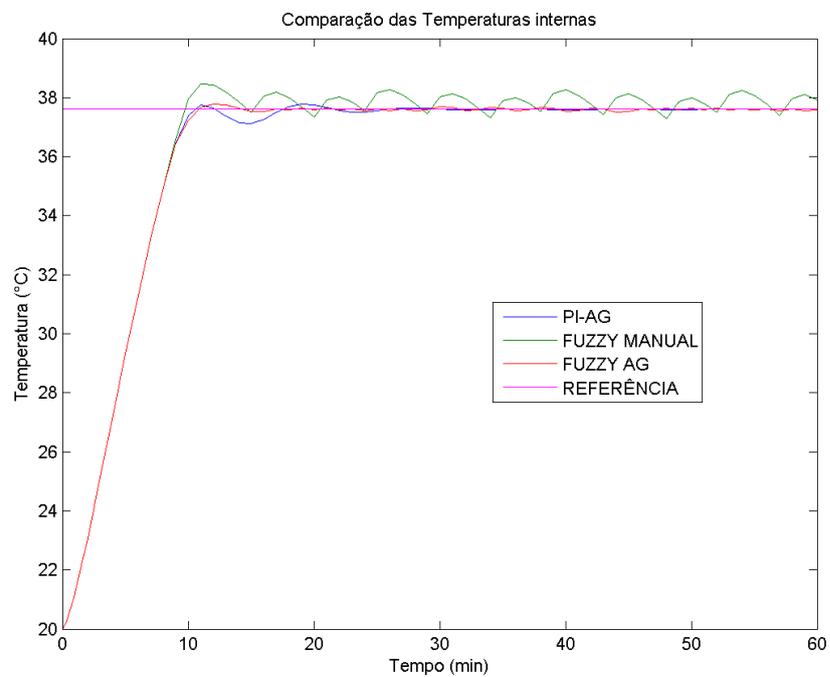


Figura 36 – Comparação das respostas PI-AG, FUZZY-MANUAL e FUZZY-AG.

Fonte: Autoria Própria.

temperatura) que, pela Figura 37, representa um trecho de leitura cujo valor médio é 46,6 e tem uma variação máxima de 0,2 em torno da média. Devido à variação, fez-se o uso de um sinal de interferência na simulação, somando a saída final para variar com valores aleatórios entre -0,2 e +0,2, simulando a imprecisão da leitura, como é possível ser visualizado na Figura 38.

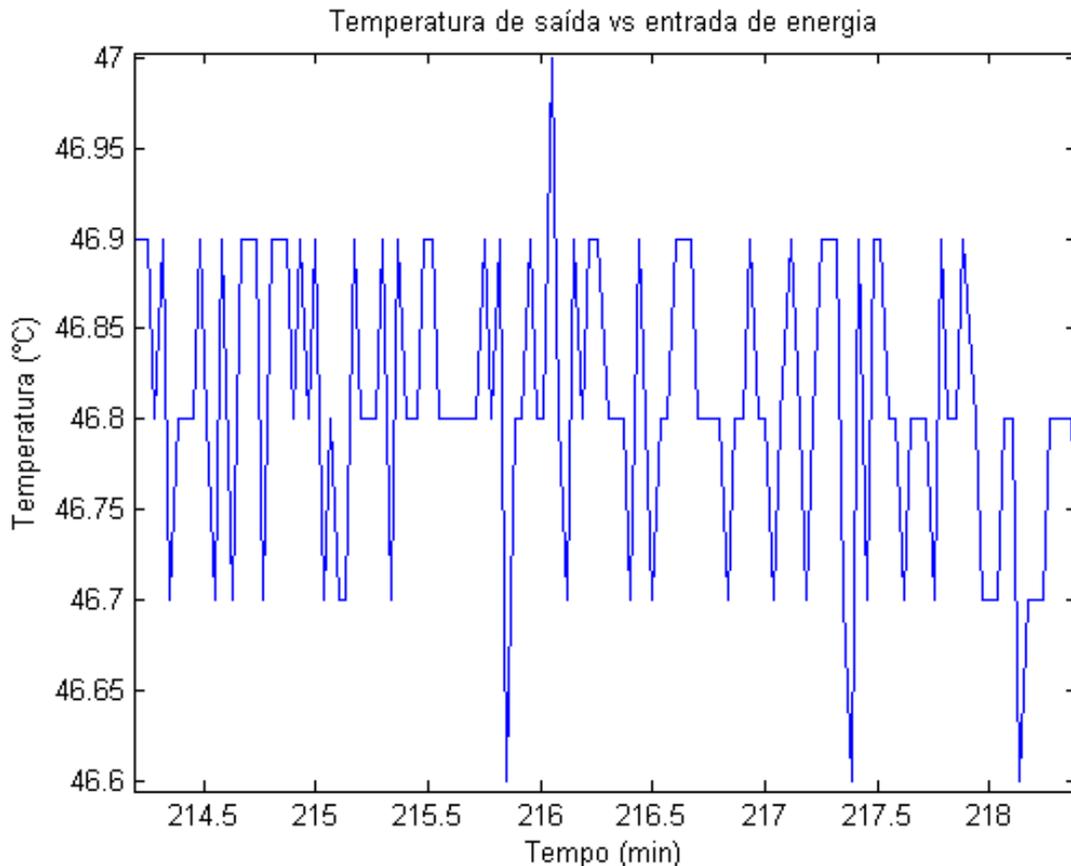


Figura 37 – Variações da leitura do sensor de temperatura.

Fonte: Autoria Própria.

Com a simulação esquematizada e conhecido o valor do controlador proporcional que mantém as perdas do sistema ($k=0,022$), restou encontrar apenas o valor do controlador proporcional segundo o erro da temperatura, sendo realizado o AG para encontrar o ganho que apresentasse menor área de erro ao decorrer do tempo (mesmo critério para as outras sintonias dos controladores).

Nas configurações do AG os limites de busca foram entre 0 e 0,3, população inicial de 30 com 10 gerações e 10% de morte e cruzamento, adicionando uma mutação gaussiana, onde alguns elementos, aleatoriamente, recebem uma alteração, possibilitando encontrar valores melhores, no caso do AG estar parado em um ponto de mínimo local (valor que aparentemente representa a melhor solução do sistema). As configurações são para apenas 1 variável, sendo sua complexidade computacional reduzida em relação ao Fuzzy, que tinham mais elementos a serem sintonizados.

Após rodar o programa, obtêm-se a convergência dos resultados buscados pelo

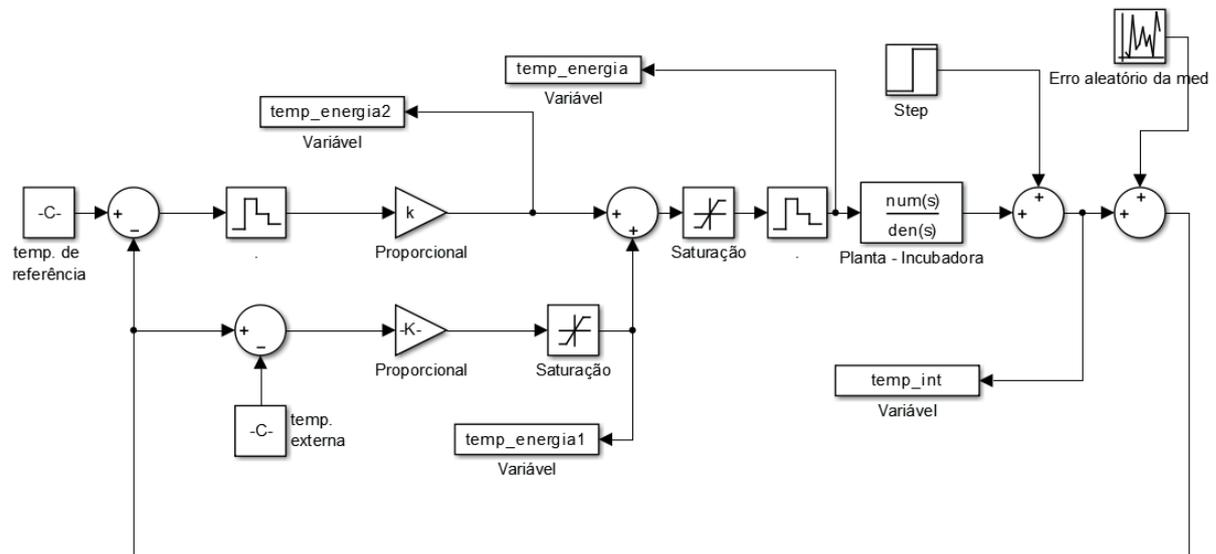


Figura 38 – Simulação do controlador duplo proporcional aplicado à planta.

Fonte: Autoria Própria.

AG para $k_{p2} = 0,1646$ (parcela proporcional do controle), a resposta do sistema e as energias dos controladores, estão nas Figuras 39, 40 e 41. Fazendo uma comparação entre a resposta obtida com os sistemas de controle PI-AG e Fuzzy-AG e analisando com *zoom*, Figuras 42 e 43, percebe-se que existe uma ultrapassagem do valor da referência de 0.4 graus, aproximadamente o dobro do Fuzzy, porém o mesmo estabiliza (de forma oscilatória devido à imprecisão da leitura) primeiro que o controlador PI-AG. Lembrando que os controladores PI e Fuzzy não estão levando em consideração oscilações de leitura.

Apesar de ser um sistema de controle relativamente simples, por usar dois controladores proporcionais, e conhecendo o comportamento das perdas do sistema, o que antes precisa de uma parcela integrativa do controlador, pode-se criar uma lógica simples de controle, tanto analógica (com resistores e amplificadores operacionais) como digital (programa fazendo leitura e aplicando os ganhos). Visto que o mesmo se comporta semelhante aos outros controladores, porém se o sistema ao longo do tempo mudar suas características e assim seu coeficiente de perdas, o mesmo apresentará um erro estacionário.

4.5 CONTROLE GPC

Utilizou-se para o GPC as equações e exemplos vistos em (SUMAR, 2008), cujo Apêndice E (Demonstração do GPC) está no Anexo A deste trabalho, servindo como suporte para as equações. E o código do *Matlab*[®] foi fornecido pelo mesmo e editado de acordo com as necessidades do projeto da incubadora.

O GPC utilizado é do tipo "Projeto indireto do Controle", ou seja, o modelo da planta existe e é definido *offline*. O modelo já foi encontrado em termos da frequência (*tf3*), mas deve ser convertido para função transferência discreto, no *Matlab*[®] a função "*c2d*"

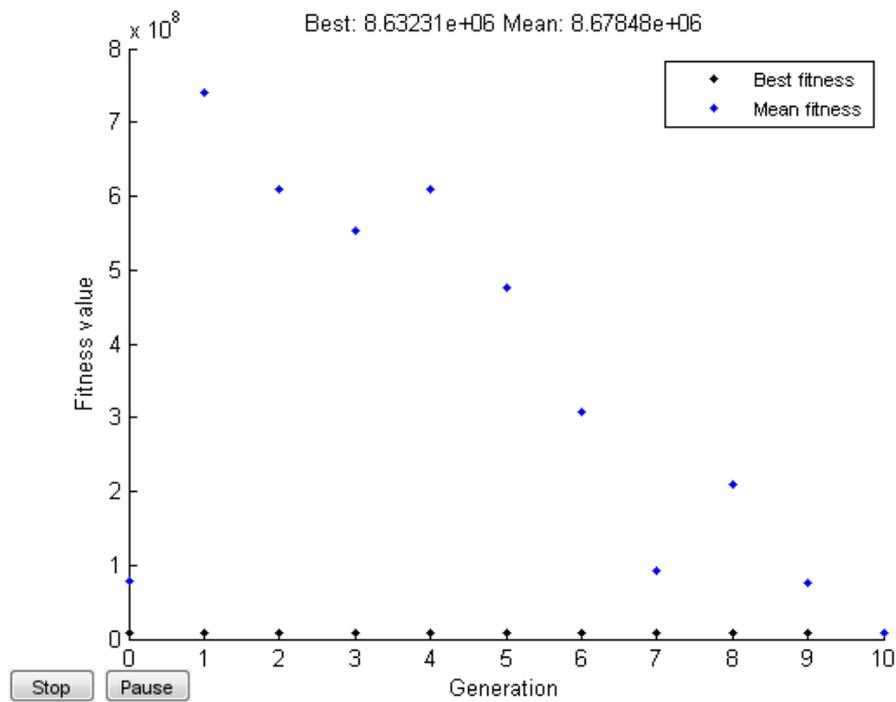


Figura 39 – Convergência do processo de otimização pelo AG.

Fonte: Autoria Própria.

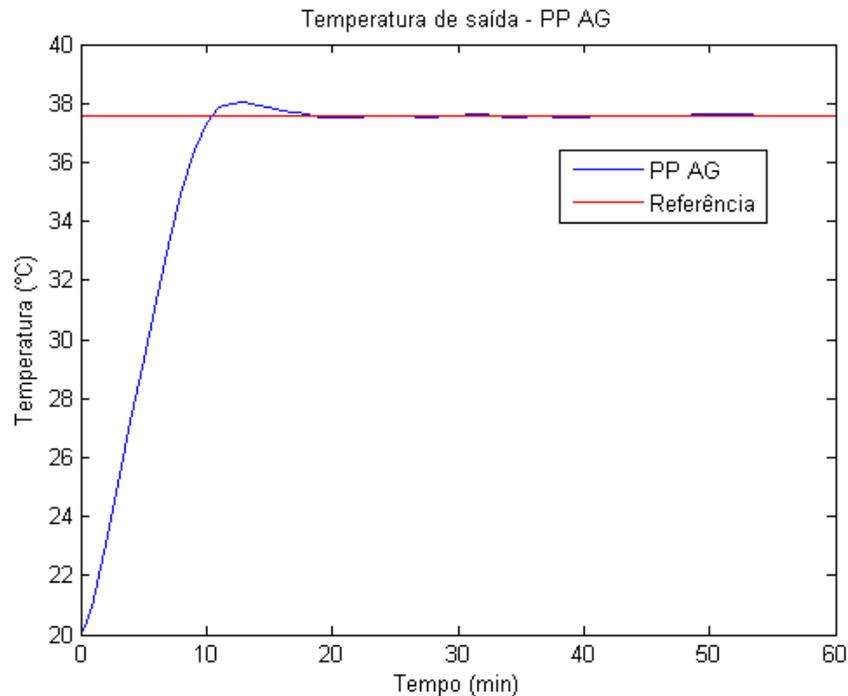


Figura 40 – Resposta da temperatura interna.

Fonte: Autoria Própria.

faz este processo. Na equação 12 está a função (tf3) na forma contínua em função de s , aplicando a transformação, com período de amostragem de 60 segundos (definido em

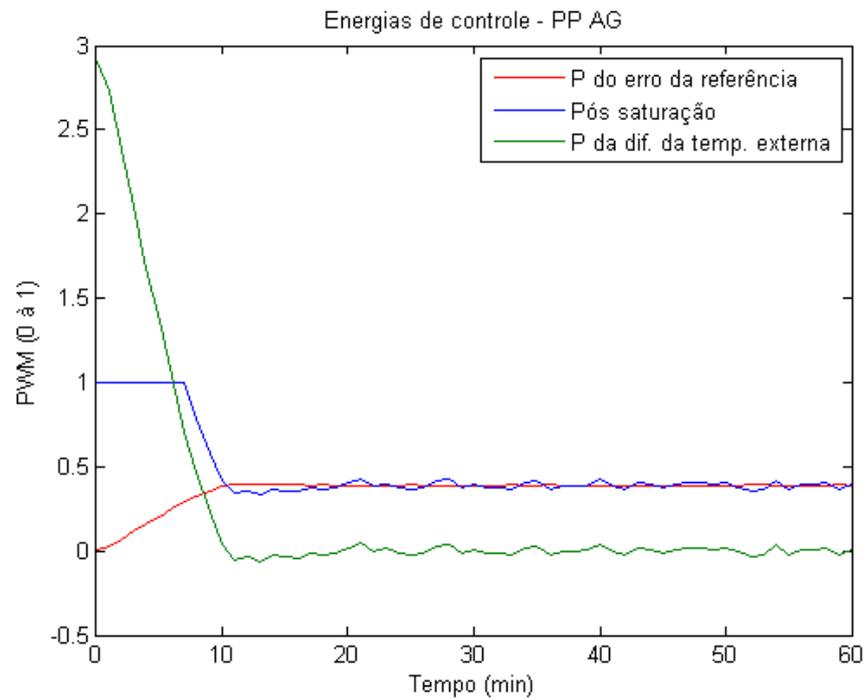


Figura 41 – Energias fornecidas pelos controladores.

Fonte: Autoria Própria.

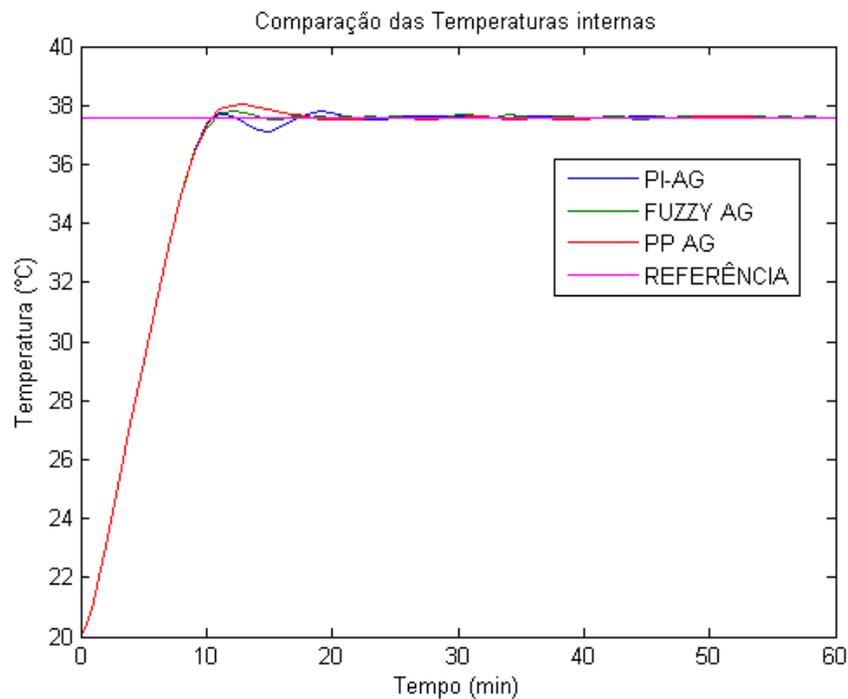


Figura 42 – Comparação do resultado do PP-AG com PI-AG e FUZZY-AG.

Fonte: Autoria Própria.

programação), tem-se a equação 13, agora em função das amostras discretas z .

$$tf3(s) = \frac{0,01109s + 0,0005504}{s^2 + 0,01363s + 1,217e - 05}; \quad (12)$$

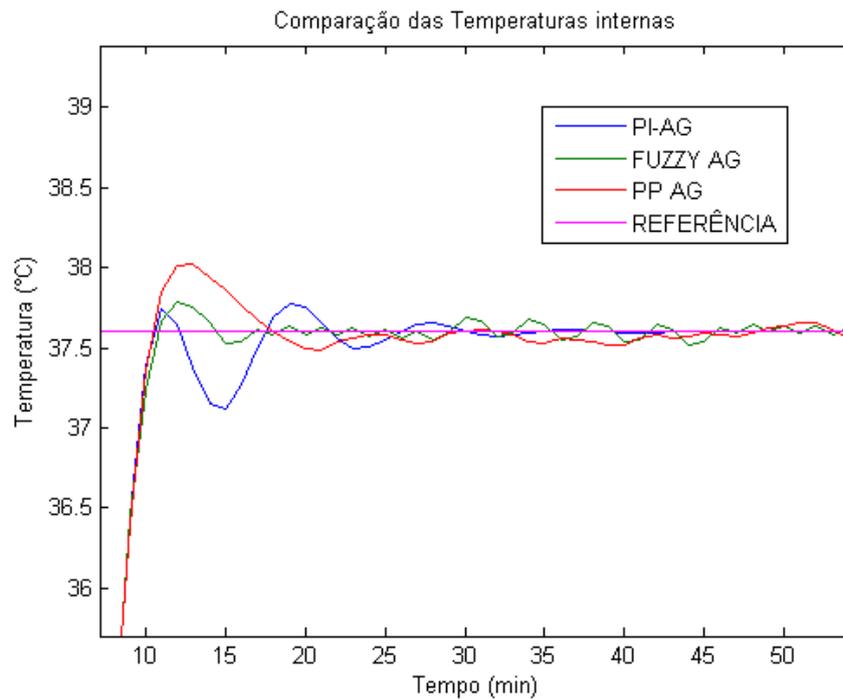


Figura 43 – Comparação do resultado com PI-AG e FUZZY-AG com zoom.
Fonte: Autoria Própria.

$$tf3(z) = \frac{1,216z + 0,132}{z^2 - 1,411z + 0,4413} \quad (13)$$

A equação 14 mostra a representação discreta do modelo da planta a ser controlada, onde A é o denominador da função discreta ($A_{(z^{-1})} = 1 + a_1z^{-1} + a_2z^{-2} + \dots + a_nz^{-n}$) e B o numerador da mesma, na forma ($B_{(z^{-1})} = b_0 + b_1z^{-1} + b_2z^{-2} + \dots + b_mz^{-m}$). Assim, arruma-se a equação 13, dividindo o numerador e o denominador por z^{-2} e colocando o termo z^{-1} em evidência no denominador, resultado na equação 15. Os vetores A e B podem ser determinados comparando as equações, encontrando: $A_{(z^{-1})} = 1 - 1,411z^{-1} + 0,4413z^{-2}$ e $B_{(z^{-1})} = 1,216 + 0,132z^{-1}$.

$$A_{(z^{-1})}\Delta y^{(k)} = z^{-1}B_{(z^{-1})}\Delta u^{(k)}; \quad (14)$$

$$tf3(z) = \frac{z^{-1}(1,216 + 0,132z^{-1})}{1 - 1,411z^{-1} + 0,4413z^{-2}} \quad (15)$$

Para projeto do GPC, testou-se várias topologias (heurísticamente) variando o número do horizonte de previsão da saída (N_2), do controle (N_u) e o fator de atuação (σ), que por sua vez compõem os parâmetros ajustáveis do controlador. O número de interações são de 60, ou seja, cada interação é de 60s dando um total de 60 minutos ou 1 hora, facilitando para comparar com os outros controladores que foram simulados neste mesmo tempo. O mesmo vale para a temperatura inicial de 20 °C e referência de 37,6 °C. Porém, como a maior contribuição deste controlador é a resposta para uma mudança de referên-

cia, e na incubadora irá ter mudanças ao longo do período, fez-se uma mudança no meio da simulação da referência para 36,7 °C (temperatura a partir do 18° dia de incubação) para analisar a resposta do mesmo.

Manualmente encontrou os parâmetros: $N2 = 10$, $Nu = 7$ e $\sigma = 0,5$. Assim simulando o sistema controlado com GPC obteve a resposta da temperatura na Figura 44 e a energia do sistema na Figura 45. Percebe-se que o sistema encontrado tem resultados apreciáveis (baixo *overshoot*, erro nulo e antecipação de mudança de referência) e sua energia de controle tem uma forma amortecida.

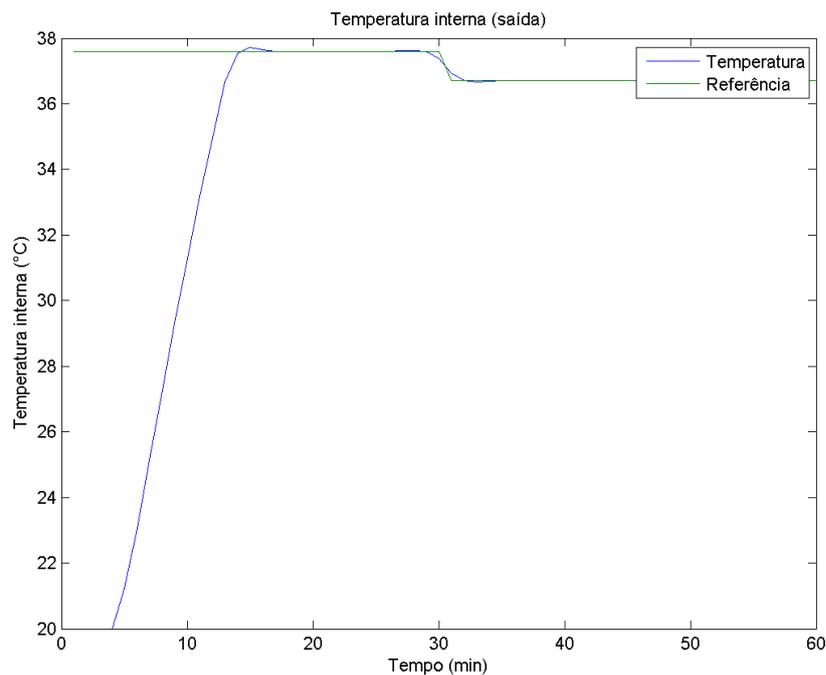


Figura 44 – Resposta da temperatura interna GPC-manual.

Fonte: Autoria Própria.

Como existem vários métodos para se determinar os coeficientes do GPC, decidiu-se usar o AG para esta situação, usando o mesmo critério nos outros controladores para minimização da função da área do erro ao longo do tempo. Configurou-se o AG para uma população de 500 indivíduos e 10 gerações. O resultado apresentado pelo algoritmo dos parâmetros foi: $N2 = 56$, $Nu = 8$ e $\sigma = 0,222$. Com 3,5 minutos aproximadamente o AG apresentou uma convergência satisfatória (valor médio se encontra com o melhor indivíduo) na Figura 46, tendo como saída da temperatura a Figura 47 e energia de controle a Figura 48.

Não foi feita a análise usando as oscilações (perturbações) do sensor de temperatura, visto que se o mesmo fosse implementado no Arduíno deveria ter um filtro em *software* (linhas de códigos). Comparando a resposta do GPC manual com o ajustado com o AG tem-se os dados: temperatura de saída e energia de controle, respectivamente, Figuras 49 e 50. Percebe-se uma redução do pico de ultrapassagem do controle com AG em relação ao manual, porém o tempo de assentamento foi o mesmo. A energia disposta

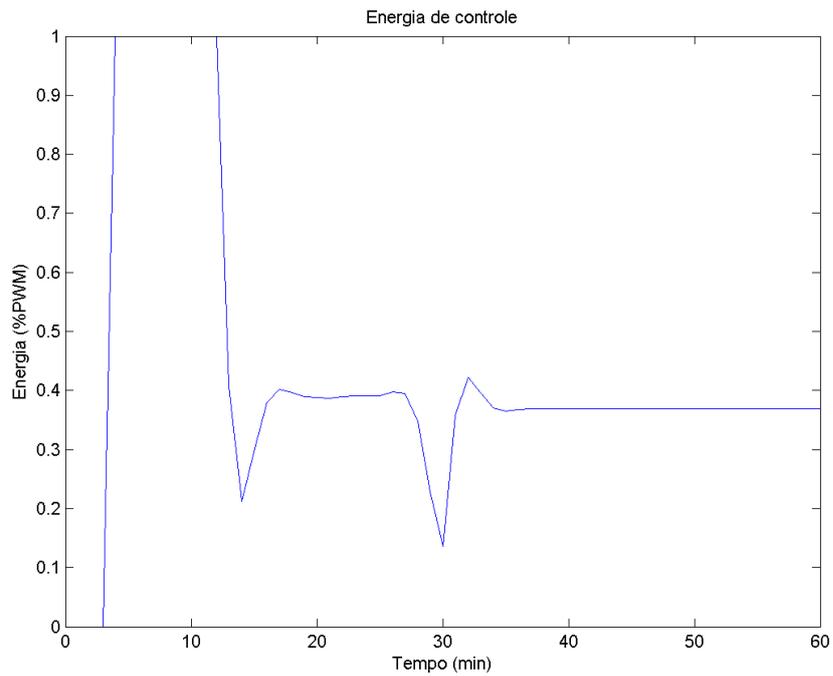


Figura 45 – Energia fornecida pelo controlador GCP-manual.
Fonte: Autoria Própria.

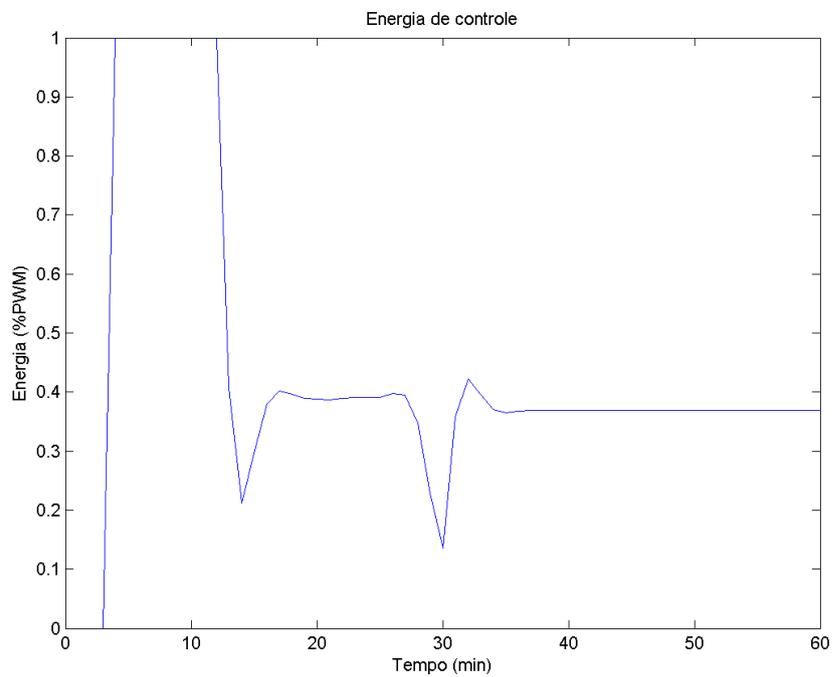


Figura 46 – Convergência do processo de otimização pelo AG.
Fonte: Autoria Própria.

pelos controladores são próximas, tendo o AG diminuído pouco a amplitude do sinal de controle, mas com maior oscilação.

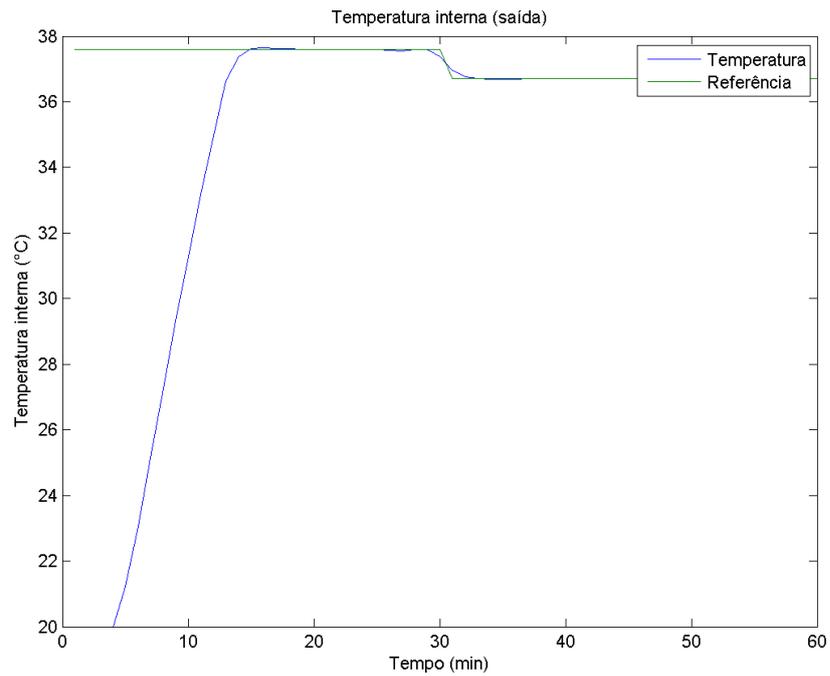


Figura 47 – Temperatura interna de saída da incubadora GPC-AG.
Fonte: Autoria Própria.

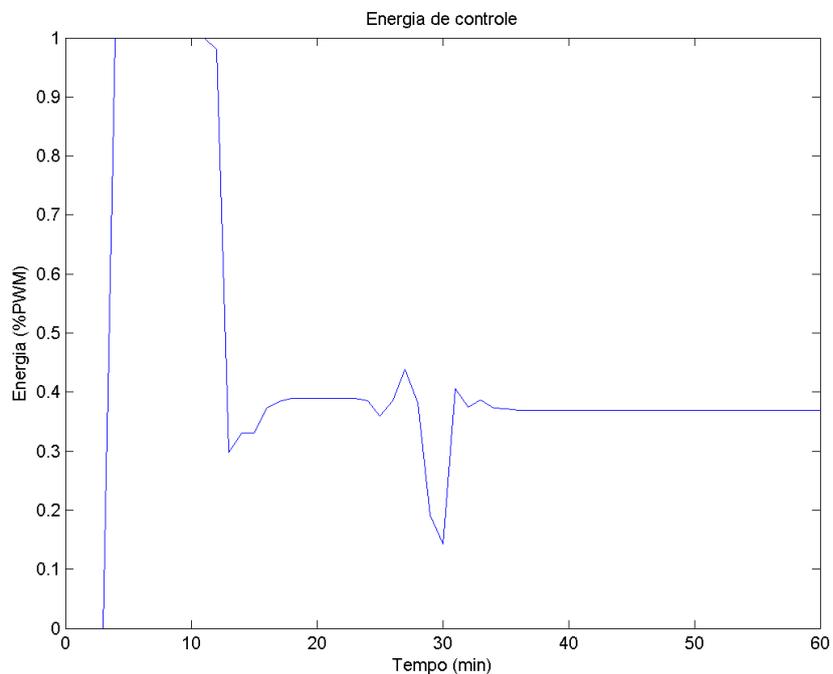


Figura 48 – Energia fornecida pelo controlador GPC-AG.
Fonte: Autoria Própria.

Para efeito de comparação, foi plotado com zoom na Figura 51, as respostas dos GPCs junto das topologias de controle, estudadas até o presente momento (apenas re-

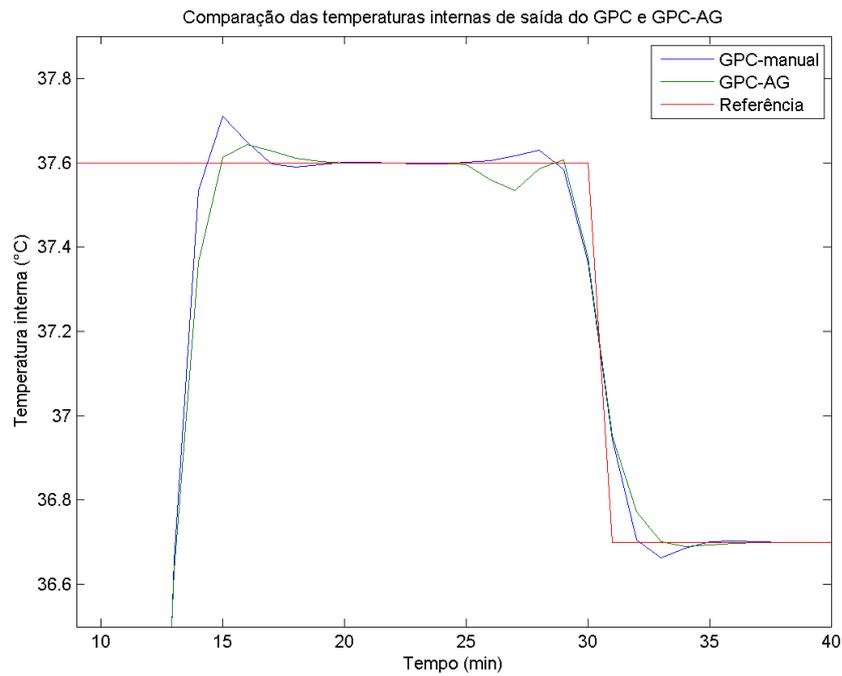


Figura 49 – Comparação das temperaturas internas de saída da incubadora com GPC-manual e GPC-AG.

Fonte: Autoria Própria.

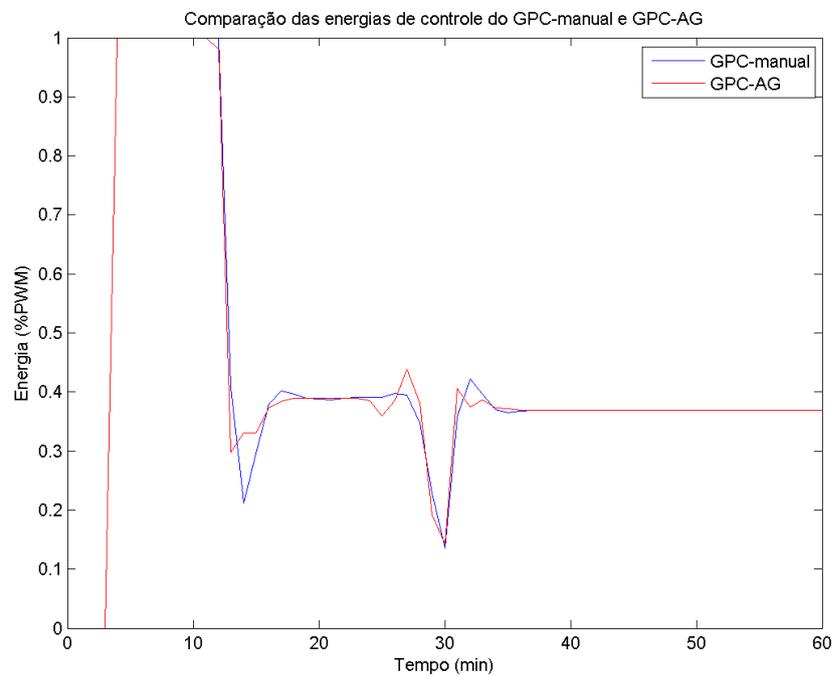


Figura 50 – Comparação das energias dos controladores GPC-manual e GPC-AG.

Fonte: Autoria Própria.

sultados ajustados com AG). Percebe-se que ambos GPCs têm os menores índices de ultrapassagem da referência, e seu tempo de assentamento é menor do que o Fuzzy (o mais rápido até o estudo do controlador PP). E apesar de não ter usado a mesma referência para os outros controladores, percebe-se que o GPC atua sempre antes da referência mudar, visto que ele é um controle preditivo, logo todos os outros controladores teriam um atraso com relação ao GPC.

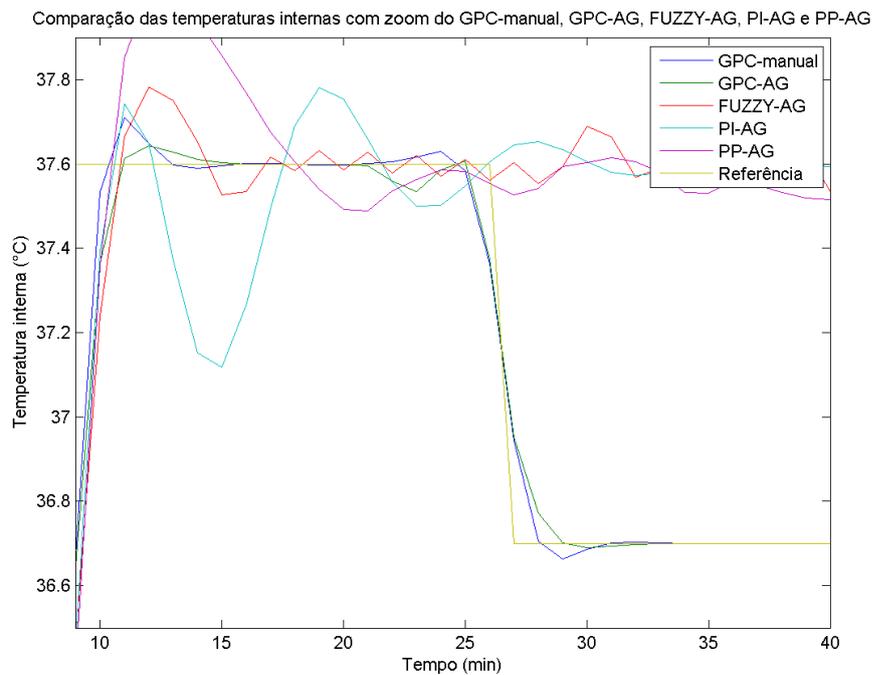


Figura 51 – Comparação das energias dos controladores GPC-manual e GPC-AG.

Fonte: Autoria Própria.

4.6 CONTROLE PID

Inicialmente retirou-se a parte derivativa do controlador, mas posteriormente, durante as análises das energias, percebeu-se que a parcela derivativa não amplificaria os ruídos a ponto de instabilizar o sistema, pois o sistema é consideravelmente lento. Assim foram seguidos os mesmos passos do controlador PI, mas utilizado a última linha da tabela 2, onde considera o ganho kd . Sendo também considerado para o termo de "anti-windup" (ka) como sendo um décimo do termo integrativo (ki).

Realizando os cálculos dos parâmetros de acordo com a tabela e fazendo $ki = \frac{kp}{Ti}$ e $kd = kp.Td$, tem-se os ganhos para ZN: $kp = 24,4$, $ki = 0,219$, $kd = 680,6$ e $ka = 0,022$. Usando a simulação proposta na Figura 52, onde considera-se variações na leitura (bloco "Erro aleatório da medida", entre -0,2 e 0,2) e perturbação de decaimento da temperatura ("Abertura de tampa- perda de 1°C) no instante de 30 minutos. Sendo a mesma semelhante à simulação do PI e do PP, pois mesclou-se a parcela proporcional da diferença de temperatura interna com a externa.

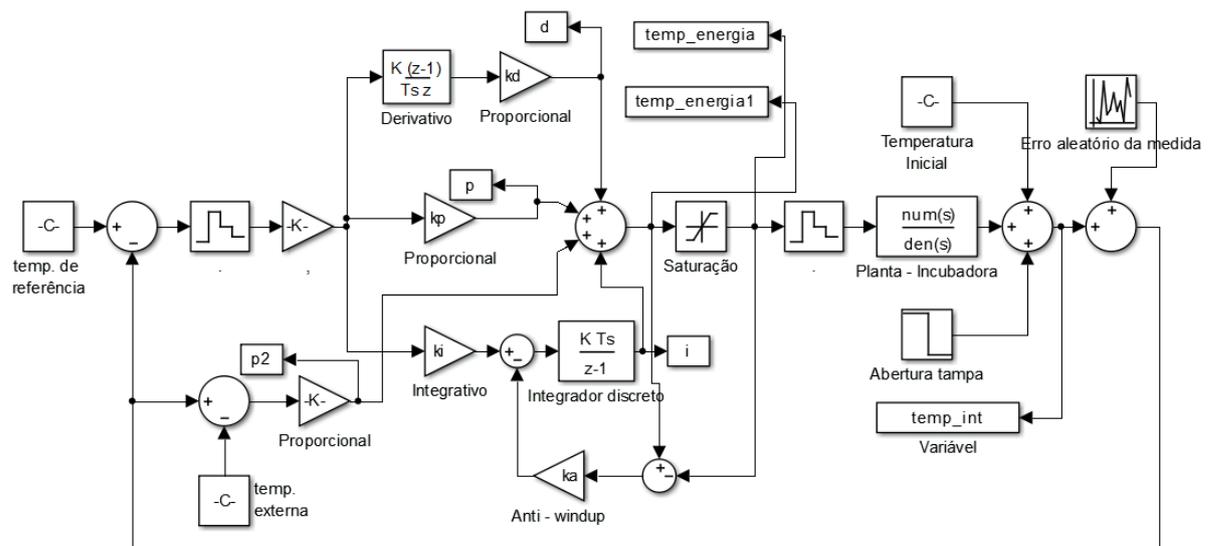


Figura 52 – Simulação do controle PID+P-ZN discreto.

Fonte: Autoria Própria.

O PID+P, por estar mesclado com o segundo proporcional, não necessitaria da parcela integrativa, pois o 2º proporcional poderia fornecer a energia referente à perda de energia. Como na prática o sistema adquire parcelas não lineares e outras influencias não consideradas na identificação, o termo proporcional será apenas uma parcela próxima do real, sendo o integrador responsável por corrigir as falhas instantâneas oferecidas pela incubadora.

Usando os dados do ZN, obtêm-se então os resultados: resposta da temperatura interna e energias de controle nas respectivas Figuras 53 e 54. Percebendo uma alta oscilação na variável de controle, sendo refletido pelas oscilações das energias proporcional, integrativa e derivativa.

Aplicando o AG para otimizar os valores, utilizou-se uma população inicial de 100 indivíduos, 20 gerações de interações, 10% de mortalidade e 30% de cruzamento, sendo os os limites máximos e mínimos em função dos ganhos ZN: $kp = (kpzn_{80\%} - kpzn_{120\%})$, $ki = (kizn_{40\%} - kizn_{120\%})$, $kd = (0 - kdzn_{200\%})$ e $ka = (0 - kizn_{50\%})$. Não iniciando todos em zero para que a busca não tivesse um espaço muito amplo de valores.

A função custo (fit) do AG é semelhante ao do PI, porém, neste caso houve uma falha entre as combinações dos ganhos que causavam instabilidades do sistema, então adicionou-se um multiplicador, que quando a variável integrativa ficasse muito alta (sinal de falha), a função custo (ainda produzindo um resultado possível devido a saturação), retornava à integral da função, multiplicado por 3, inviabilizando o resultado, que ao longo do processo seria naturalmente excluído. Também adicionado uma ponderação fazendo a mesma simulação, usando outra função de referência (fora do ponto linearizado), fazendo a necessidade de calcular um ganho ki diferente do mínimo (não iniciado em zero por este motivo também), visto que para qualquer mudança dos parâmetros da chocadeira é necessário que haja uma correção constante, neste caso responsabilidade do termo

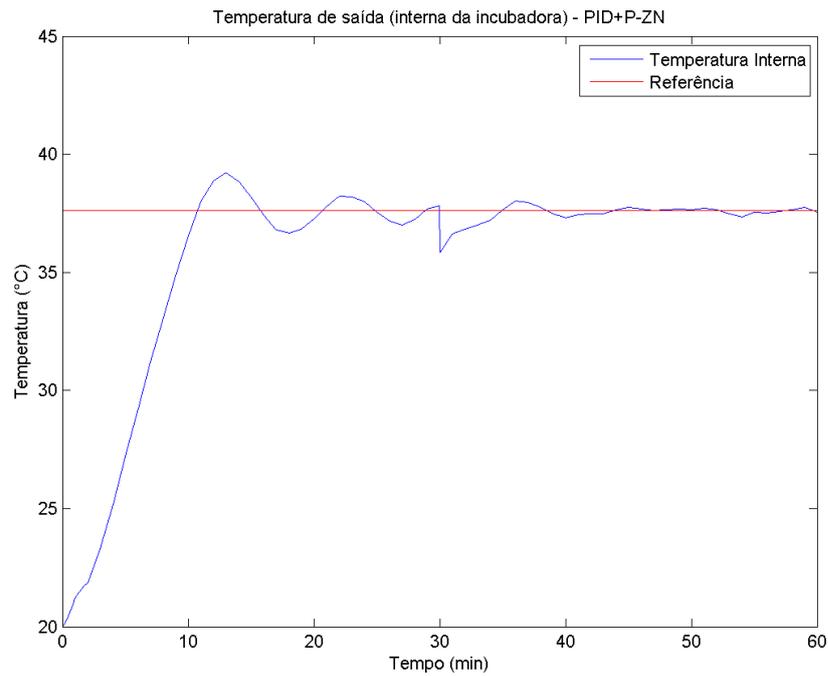


Figura 53 – Temperatura de saída pelo método PID+P-ZN.
Fonte: Autoria Própria.

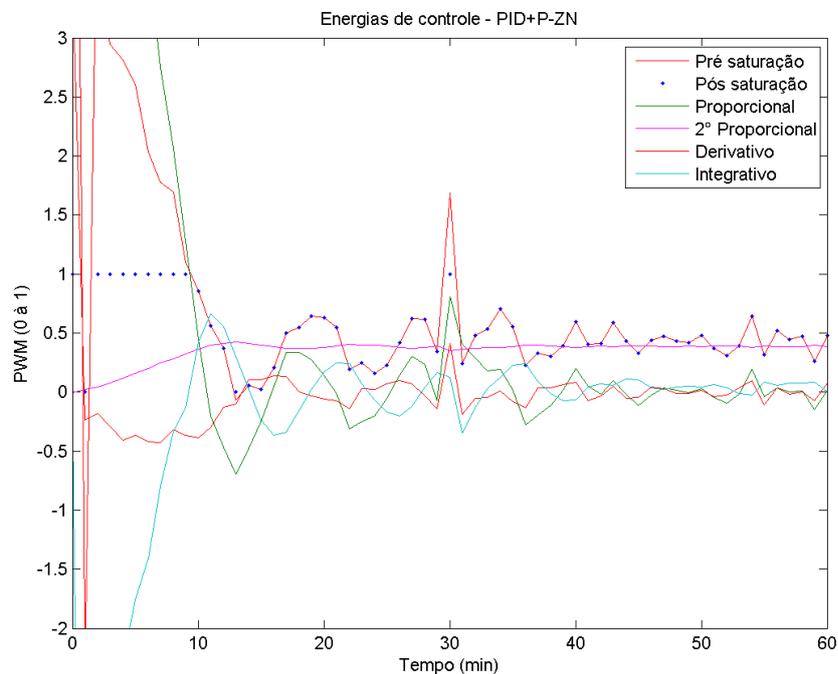


Figura 54 – Energias pelo método PID+P-ZN.
Fonte: Autoria Própria.

integrativo.

Aplicando o AG (algoritmo genético), obtém-se os valores otimizados: $k_p = 28,8$,

$k_i = 0,158$, $k_d = 635,7$ e $k_a = 0,0116$. A Figura 55, mostra os resultados dos indivíduos gerados pelo AG convergindo para um número cada vez menor, ou seja, encontrando os parâmetros que melhor desempenham a função de ajuste escolhida, com algumas variações divergentes devido ao cruzamento que resta parâmetros causadores de instabilidade. E apesar de parecer que uma solução inicial se manteve, durante o andamento do processo a melhora é significativa à pequenas áreas, mas que em termos de valores não são significativos no visualmente no gráfico. As Figuras 56 e 57, mostram a resposta da temperatura interna e das energias de controle respectivamente. Visto que as oscilações foram reduzidas significativamente, ainda existindo devido à incerteza nas medidas de temperatura.

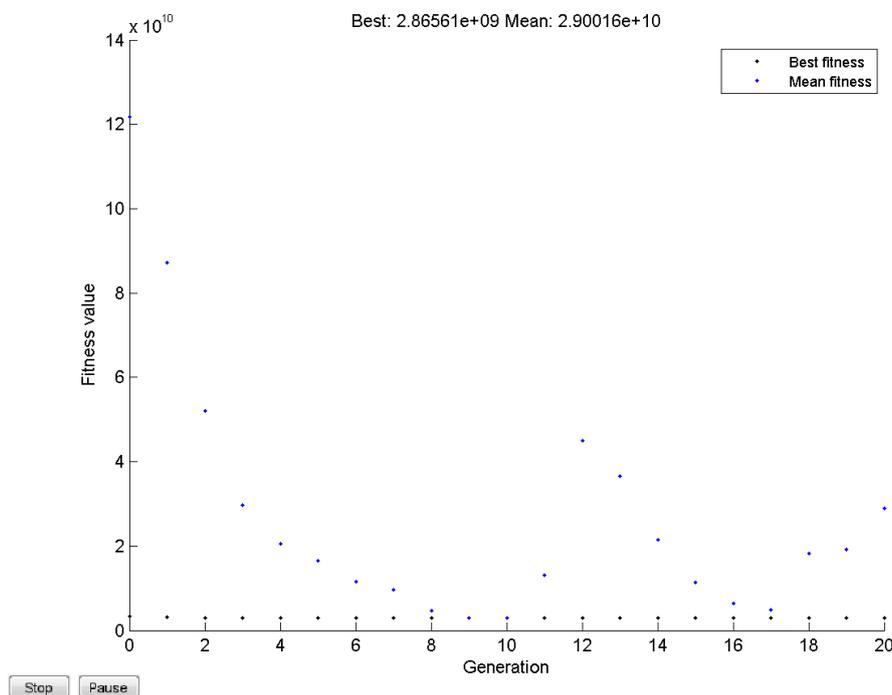


Figura 55 – Convergência dos resultados analisados no processo do AG.

Fonte: Autoria Própria.

Compara-se na Figura 58 a temperatura de saída para os dois casos analisados e percebe-se que o sistema AG encontrado se assemelha ao mesmo encontrado no PI, pois os ganhos k_p e k_i são próximos e o termo derivativo não tem influência sobre o sistema (podendo ser desconsiderado). O Ajuste por AG teve uma melhora muito evidente, ao contrário do PI. O 2º proporcional tem uma vantagem durante a implementação prática, que se o sistema por algum motivo reiniciar e for perdida a variável integrativa, o mesmo já fornece uma estimativa muito próxima do valor real de energia necessária pra manter a diferença de temperatura atual, permitindo ao termo integrativo se restabelecer de forma a não prejudicar tanto quando sem o mesmo.

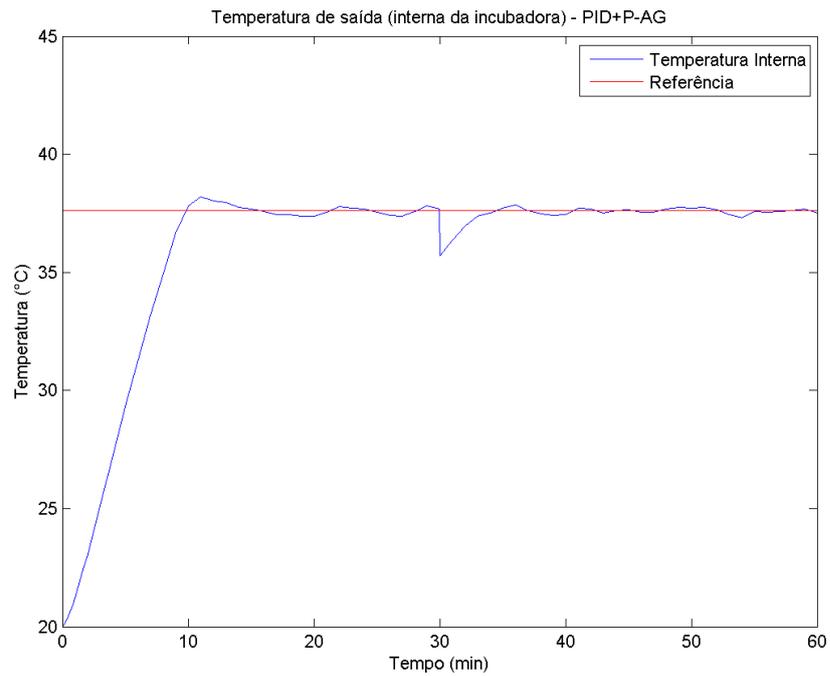


Figura 56 – Temperatura de saída pelo método AG.

Fonte: Autoria Própria.

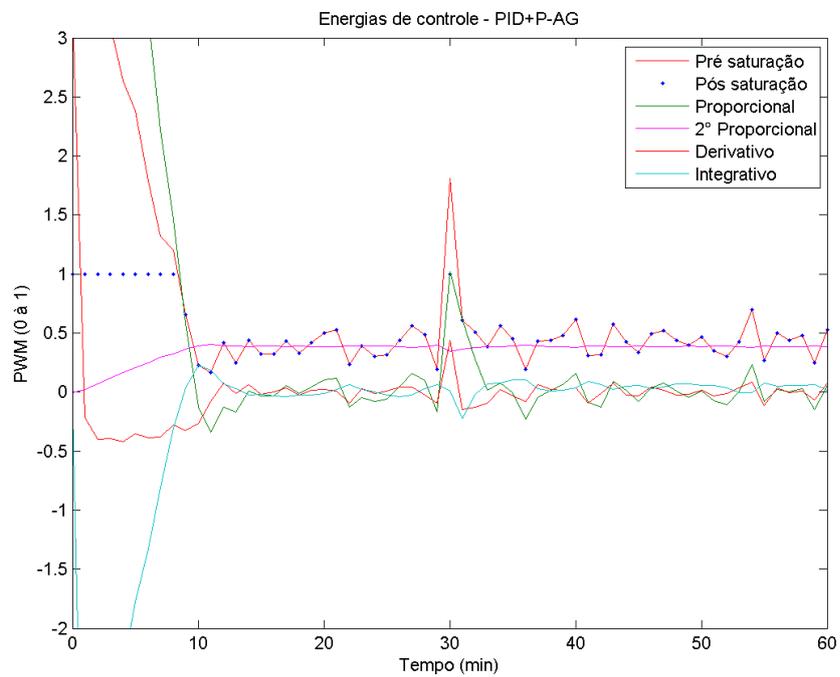


Figura 57 – Energias pelo método AG.

Fonte: Autoria Própria.

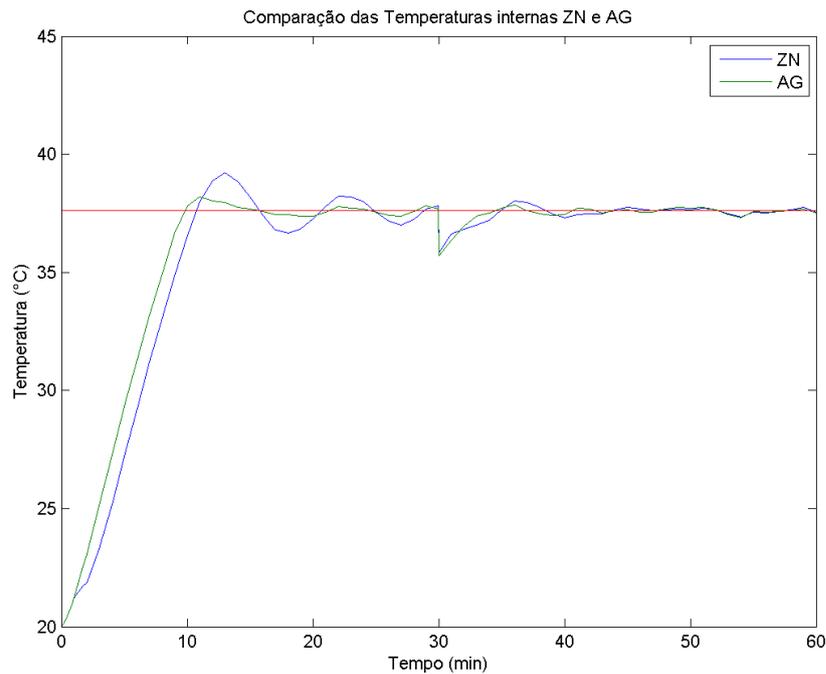


Figura 58 – Comparação dos métodos de Ajuste ZN e AG para o PID+P.

Fonte: Autoria Própria.

4.7 ESCOLHA DO CONTROLADOR

Para a decisão do controlador a ser implementado, foi simulado todos os controladores ajustados com AG (melhores resultados individuais) seguindo a mesma referência, sob as mesmas condições iniciais, uma pequena alteração na equação do sistema (simulando uma alteração de parâmetros devido a perda de características naturais) e considerando ruídos do sensor de temperatura de $-0,2^{\circ}\text{C}$ a $0,2^{\circ}\text{C}$ para todos os controladores. Assim gerou-se a Figura 59, onde é possível perceber, numa visão macroscópica, que todos os controladores estão seguindo a referência, que tem seu valor alterado em 30 min e todos apresentam pequenas oscilações (devido à simulação da falha de leitura).

Analisando o gráfico com zoom, sobre a região de interesse (regime permanente), tem-se a Figura 60. Nesta é possível comparar todas as respostas dos controladores, mesmo que visualmente, percebe-se que o controlador que tem menor erro e sem atrasos é o GPC, visto que sua capacidade preditiva tem alta melhor desempenho para reduzir picos de ultrapassagem da referência e atuação antecipada visto uma mudança na referência (cancelando o atraso). Porém a implementação do GPC exigiria um grande esforço computacional do microcontrolador, visto que são realizadas muitas interações para a determinação da resposta do controlador, dificultando também a programação do mesmo.

Durante a escrita deste trabalho foi utilizado o controlador PI-AG adaptado, adicionando a segunda parcela proporcional relativa à diferença de temperatura interna e externa, visto que já se havia conseguido 12 ovos chocados para se testar a incubadora. Percebeu-se que o controlador não utilizou da parcela integrativa, sendo o segundo proporcional totalmente responsável por manter a diferença de temperatura interna e externa,

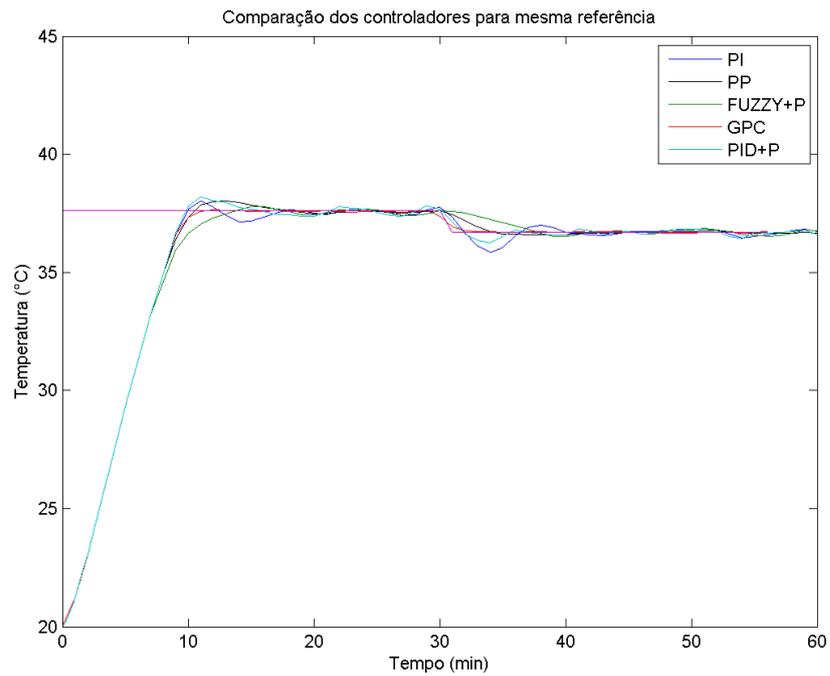


Figura 59 – Comparação dos controladores para mesma referência.
Fonte: Autoria Própria.

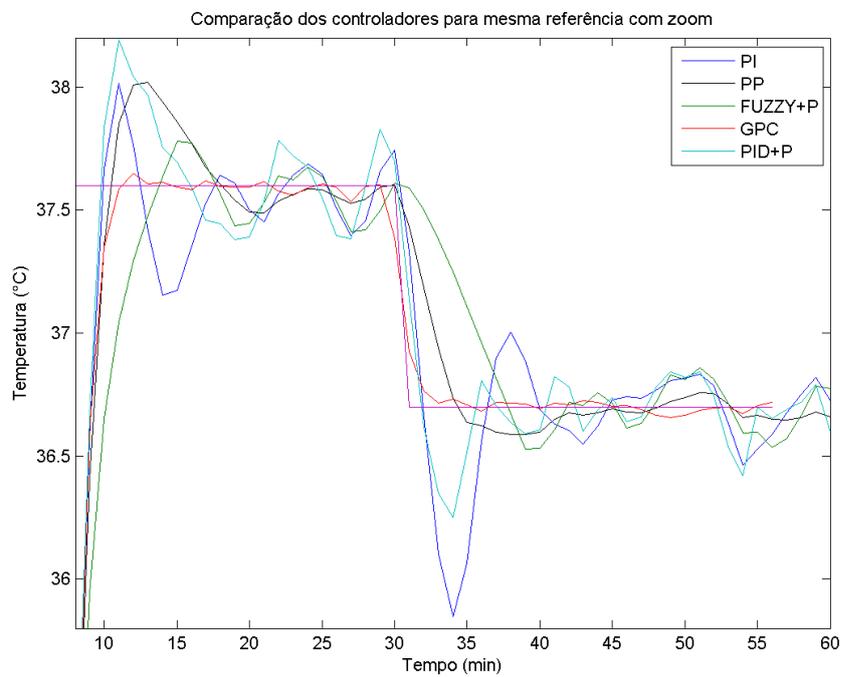


Figura 60 – Comparação dos controladores para mesma referência com zoom.
Fonte: Autoria Própria.

não gerando o erro estacionário como esperado devido à imprecisões ou variações do sistema.

Escolheu-se então para ser implementado o controlador PP, visto que suas propriedades foram observadas nos controladores PI, PID+P e Fuzzy+P, com modificações conceituais, mas de mesmo esquema de funcionamento (parcela proporcional para corrigir o erro e parcela integrativa para manter a perda de temperatura). Como é possível ver na Figura 61 o controlador possui um alto valor de *overshoot* da referência mas não repete o mesmo fato na volta da mudança de referência, logo por ser apenas no início e considerando que incubadora sofrerá pequenas variações na temperatura este tempo de ultrapassagem não será prejudicial para o sistema. Visto que por não possuir o integrador, caso a incubadora seja reiniciada, não haverá perda de variável e não terá oscilação na temperatura.

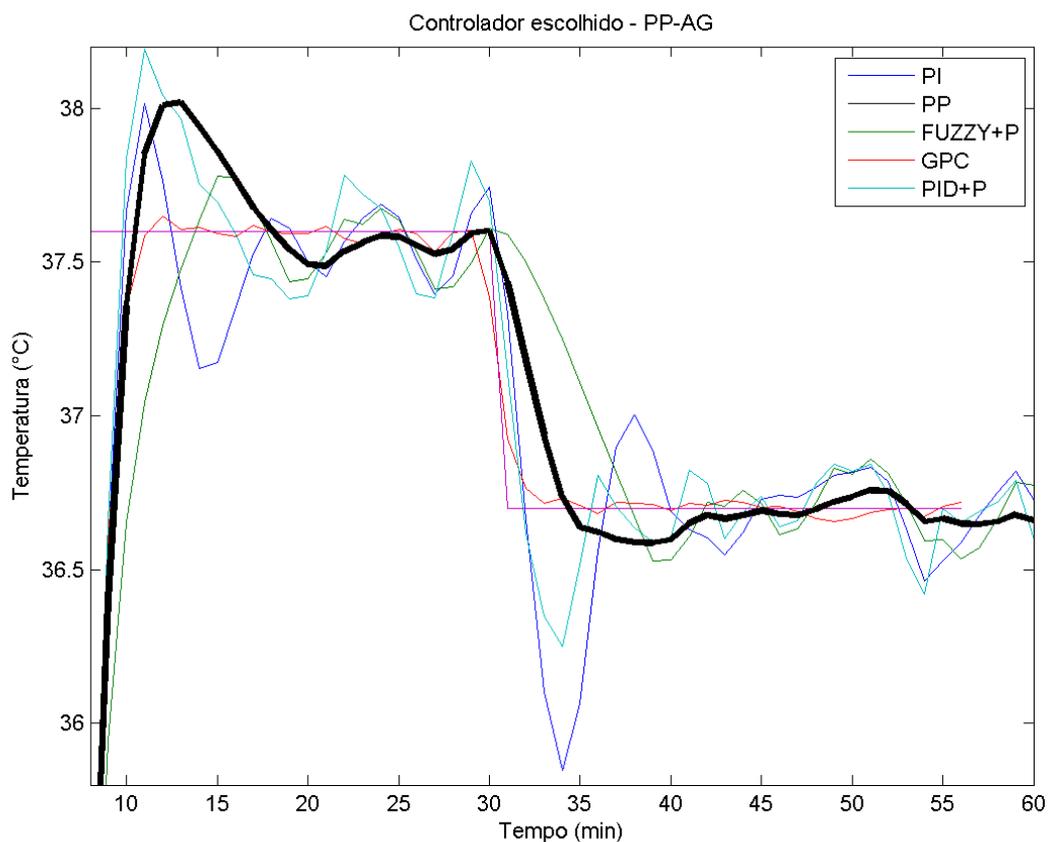


Figura 61 – Controlador escolhido - PP-AG.

Fonte: Autoria Própria.

5 IMPLEMENTAÇÃO DA PROGRAMAÇÃO

Utilizou-se a plataforma Arduino® para programar o microcontrolador Atmega328p (utilizado no Arduino UNO), pois o mesmo contém 28 pinos, dentre eles: leitura analógica, escrita analógica (PWM), leitura/escrita digital, alimentação e comunicação serial. O esquema de pinagem fornecido pelo fabricante está mostrado na Figura 62, onde foi feito o esquema de ligações segundo o manual.

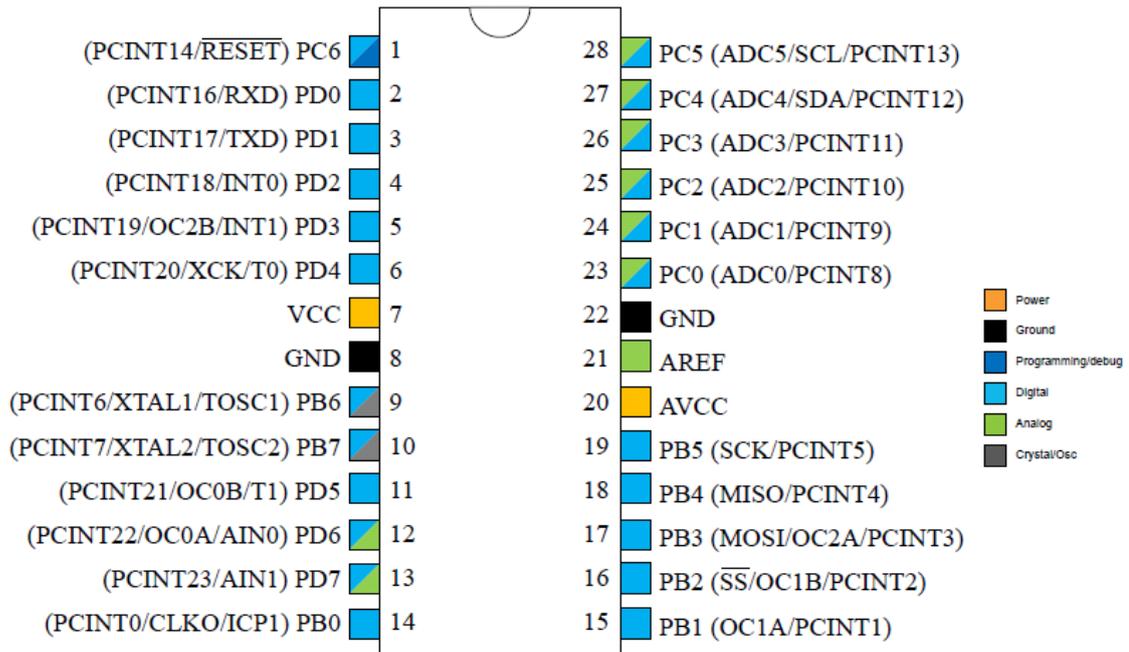


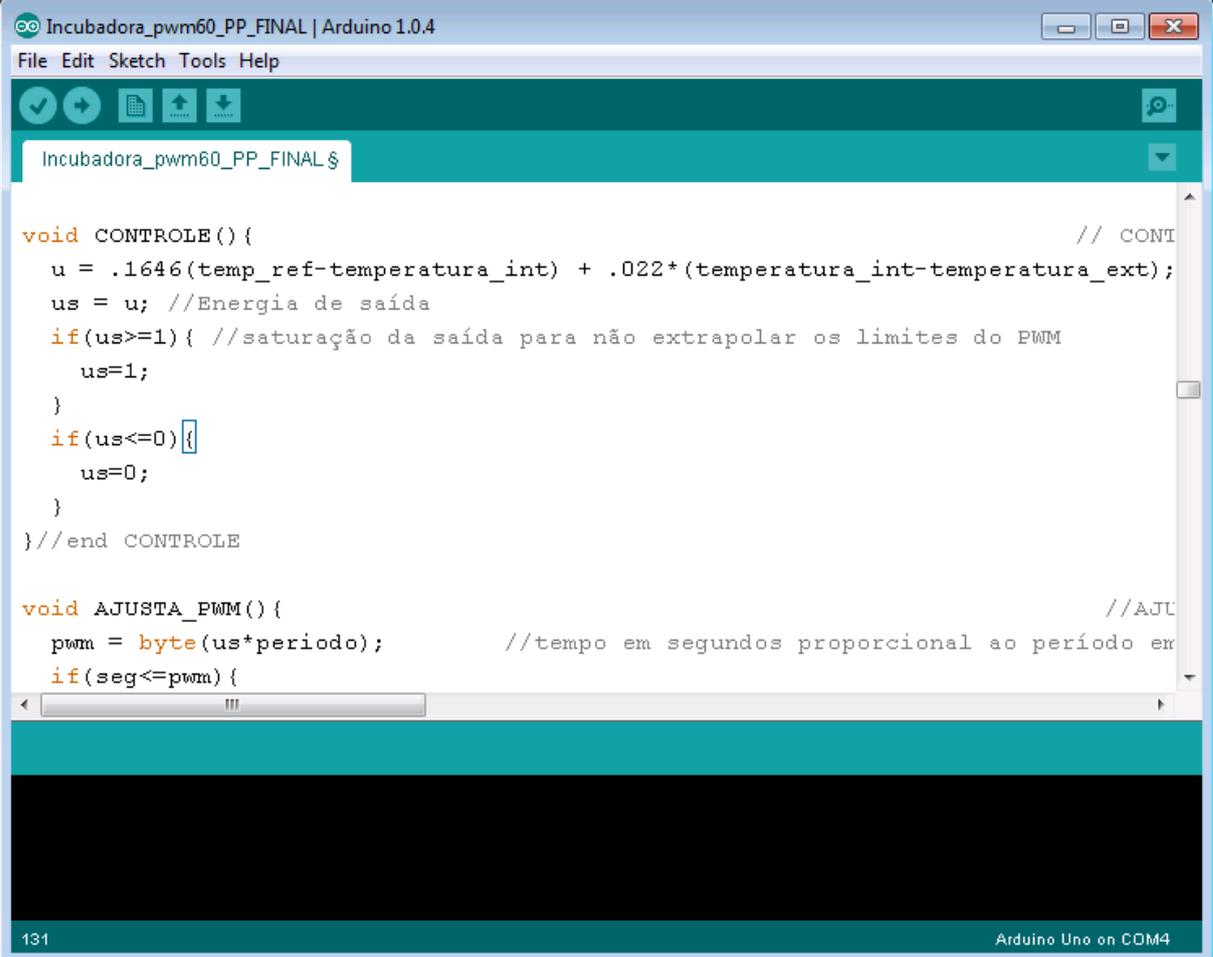
Figura 62 – Pinagem do microcontrolador Atmega328p.

Fonte: (COORPORATION, 2011)

A programação feita no Arduino está disponibilizada no Apêndice B, onde consta todas as declarações de variáveis (cabeçalho inicial), definições dos pinos e configurações principais (*void setup()*), a rotina principal em *loop* (repetição infinita) e as funções/procedimentos (procedimento é uma função que não retorna valores) auxiliares para facilitar a fluidez da escrita do código e fácil alteração.

O controle então se resume à algumas linhas de códigos que podem ser vistas na Figura 63, onde "u" representa a energia do controlador e "us" a mesma energia pós saturador e a sequência de *ifs* seguintes que limitam a energia PWM entre 0 e 1 (saturador).

A embarcação obteve sucesso ao se conectar a incubadora de ovos no computador via serial, usando o conversor USB/Serial e, obter no monitor do Arduino os dados retornados em valores, mostrando que todos os sensores estão funcionando corretamente e o controle atuando da forma desejada. A Figura 64 mostra os primeiros segundos de informações fornecidos pela incubadora (2 segundos de resolução), dados que servem de apoio para as análises da incubadora, mas que para seu funcionamento rotineiro não se fará necessário.



```
Incubadora_pwm60_PP_FINAL | Arduino 1.0.4
File Edit Sketch Tools Help

Incubadora_pwm60_PP_FINAL $

void CONTROLE() { // CONTROLE
  u = .1646(temp_ref-temperatura_int) + .022*(temperatura_int-temperatura_ext);
  us = u; //Energia de saída
  if(us>=1){ //saturação da saída para não extrapolar os limites do PWM
    us=1;
  }
  if(us<=0){
    us=0;
  }
} //end CONTROLE

void AJUSTA_PWM() { //AJUSTAR
  pwm = byte(us*período); //tempo em segundos proporcional ao período em segundos
  if(seg<=pwm){

```

131 Arduino Uno on COM4

Figura 63 – Pinagem do microcontrolador Atmega328p.

Fonte: Autoria própria.

Chocadeira Conectada!!															
DIA	HORA	MIN	SEG	SEG_ABS	TEMP_RF	TEMP_I	TEMP_E	HUM_REF	HUM_I	HUM_E	PWM	U	US	PERIODO	
0	0	1	3	62	37,6	28,9	25	60	59,1	36	100	1,52	1	60	
0	0	1	5	65	37,6	28,9	25	60	59,1	36	100	1,52	1	60	
0	0	1	7	66	37,6	29	25	60	59,7	36	100	1,52	1	60	
0	0	1	9	69	37,6	29	25	60	59,7	36	100	1,52	1	60	
0	0	1	11	70	37,6	29,1	25	60	59,7	36	100	1,52	1	60	
0	0	1	13	73	37,6	29,1	25	60	59,7	36	100	1,52	1	60	
0	0	1	15	74	37,6	29,1	25	60	59,8	36	100	1,52	1	60	
0	0	1	17	77	37,6	29,2	25	60	60	36	100	1,52	1	60	
0	0	1	19	78	37,6	29,3	25	60	60,1	36	100	1,52	1	60	
0	0	1	21	81	37,6	29,4	25	60	60,2	36	100	1,52	1	60	
0	0	1	23	82	37,6	29,5	25	60	60,4	36	100	1,52	1	60	
0	0	1	25	85	37,6	29,5	25	60	60,4	36	100	1,52	1	60	
0	0	1	29	89	37,6	29,6	25	60	60,5	36	100	1,52	1	60	
0	0	1	33	93	37,6	29,8	25	60	60,7	36	100	1,52	1	60	
0	0	1	37	96	37,6	29,9	25	60	60,8	36	100	1,52	1	60	
0	0	1	41	100	37,6	30	25	60	61,4	36	100	1,52	1	60	
0	0	1	45	104	37,6	30,1	25	60	61,4	36	100	1,52	1	60	
0	0	1	47	107	37,6	30,2	25	60	61,6	36	100	1,52	1	60	
0	0	1	49	108	37,6	30,2	25	60	61,6	36	100	1,52	1	60	
0	0	1	51	111	37,6	30,2	25	60	61,6	36	100	1,52	1	60	
0	0	1	53	112	37,6	30,2	25	60	61,6	36	100	1,52	1	60	
0	0	1	55	115	37,6	30,3	25	60	61,6	36	100	1,52	1	60	
0	0	1	57	116	37,6	30,5	25	60	61,6	36	100	1,52	1	60	
0	0	1	59	119	37,6	30,5	25	60	61,6	36	100	1,52	1	60	
0	0	2	1	120	37,6	30,5	25	60	61,5	36	100	1,52	1	60	
0	0	2	3	123	37,6	30,5	25	60	61,5	36	100	1,29	1	60	
0	0	2	5	124	37,6	30,7	25	60	61,5	36	100	1,29	1	60	

Figura 64 – Tabela de dados obtida pela leitura serial do microcontrolador.

Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

O presente trabalho obteve êxito em concluir a proposta de construção da incubadora utilizando Arduino e controlar a temperatura através de um método de controle selecionado dentre outros métodos estudados.

A construção foi realizada utilizando conceitos obtidos dentro e fora da faculdade, interagindo com a área da mecânica, pois se fez uso de ferramentas e materiais mecânicos. As áreas utilizadas da elétrica são muitas, dando suporte para todos os dimensionamentos de componentes, esquemas de ligação, projeto de circuitos, acionamentos elétricos, orçamento e compra de componentes eletrônicos, entre outros. A área de avicultura contribuiu para definições dos parâmetros de projeto, ajudando a aumentar o horizonte de conhecimento sobre aspectos biológicos e de interações do mundo natural com as máquinas. Assim, englobando vários conceitos em um único projeto.

O estudo dos controladores teve grande impacto para o aumento do conhecimento sobre os mesmos, visto que as análises foram feitas de forma profunda e, de forma prática, foram abordados vários controladores, utilizando uma topologia cada e passando por um processo de otimização, no caso o AG. Apesar da análise de vários tipos de controladores, no decorrer do desenvolvimento do trabalho conclui-se que um método mais simples seria capaz de resolver o problema em questão, não necessitando implementar métodos complexos. A familiarização com o ambiente de simulação e geração de resultados é o mais importante, visto que sabendo utilizar a ferramenta (Matlab[®]) é possível aplicar os conceitos aprendidos para quase todos os tipos de plantas e situações físicas.

Todos os programas utilizados (menos parte do código do controle GPC), foram elaborados desde o princípio, utilizando como base livros e materiais acadêmicos obtidos durante o curso. Como são utilizados muitos ambientes de programação e simulação, saber lidar com este tipo de ferramenta é fundamental para um engenheiro moderno. O Matlab[®] é uma ferramenta muito "poderosa" no sentido de auxiliar em simulações, com o auxílio de *toolboxes* prontas para várias aplicações e análises (como foram vistas algumas). O Arduino[®] foi optado no lugar de outras plataformas de programação, por possuir uma linguagem mais acessível e livre (*open-source*), tendo várias bibliotecas de componentes disponibilizadas na internet gratuitamente e facilitando o uso do microcontrolador ATMEGA328p, que por sua vez é de baixo custo.

Para validar a construção da incubadora, fez-se um teste de uma hora (real e simulado), mostrado pela Figura 65, onde a temperatura estabilizou próximo da referência, apresentando baixas oscilações em regime permanente, devido aos erros de leitura, presentes no sensor de temperatura e lentidão da variação causada pela inércia térmica.

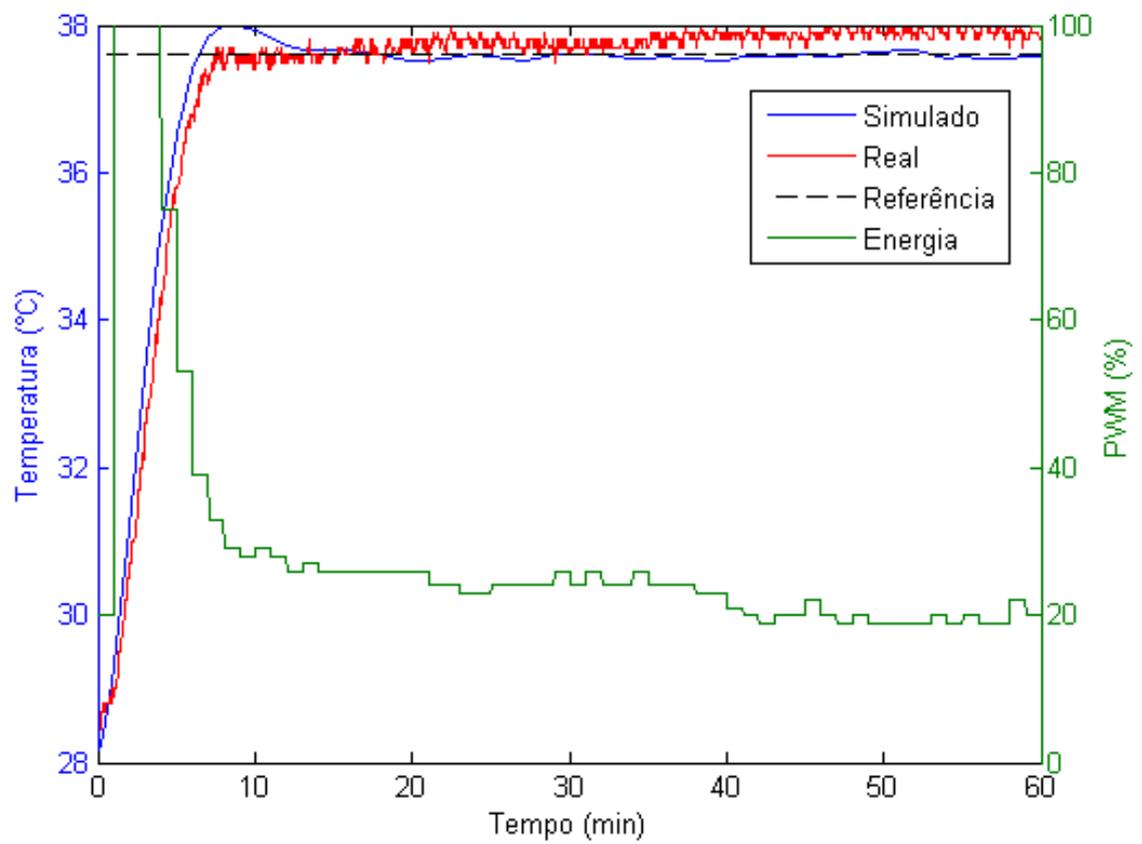


Figura 65 – Resposta real da temperatura da incubadora.

Fonte: Autoria Própria.

REFERÊNCIAS

- ALMEIDA, P. M. Incubação artificial. Jatobá, GO, 2008.
- BERTACHI, A. H.; SILVA, L. R. B.; SUMAR, R. R.; ANGÉLICO, B. A.; GOEDTEL, A. Controle de um processo multivariável em uma planta didática industrial utilizando redes neurais. 2013.
- BOHN, C.; ATHERTON, D. An analysis package comparing pid anti-windup strategies. **IEEE Control Systems**, IEEE, v. 15, n. 2, p. 34–40, 1995.
- CAMPOS, T. d. S. **Chocadeira Artesanal Automatica**. [S.l.]: Video disponível em <<https://www.youtube.com/watch?v=DvaUdZoCniE>>. Acesso em: 06 fev. 2016, 2014.
- COORPORATION, A. **Atmel ATmega328P Datasheet**. 2011.
- OGATA, K. **Modern control engineering**. [S.l.]: Prentice Hall PTR, 2001.
- ROSA, P. S.; AVILA, V. d. Variáveis relacionadas ao rendimento da incubação de ovos em matrizes de frangos de corte. Embrapa Suínos e Aves, 2000.
- SILVA, I. d.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas. **São Paulo: Artliber**, 2010.
- SUMAR, R. R. Análise e projeto de controladores discretos com aprendizado direto. Florianópolis, SC, 2008.

APÊNDICE A – CONSTRUÇÃO DA INCUBADORA DE OVOS

A construção da incubadora de ovos não necessita de muita descrição, pois foi feita de forma simples e usando ferramentas comuns, sendo as imagens quase autoexplicativas. A confecção das PCIs também é relativamente simples, mas devido à vasta gama de métodos encontrados na internet e por diferentes alunos, fez-se uma abordagem resumida do método mais utilizado por alunos e professores do CIPECA (Centro Integrado de Pesquisa em Controle e Automação), mostrando os métodos e materiais mais usados.

Também são apresentados os procedimentos para realização de uma PCI usando a máquina S63 da LPKF[®], disponibilizado pelo CIPECA para as coordenações, para realização de prototipagem.

Na Figura 66, é apresentada a estrutura de madeira da incubadora, onde abrigará todos os elementos, tanto interno quanto externo.

Segue na Figura 67, o esquema de ligação (esquemático) feito pelo programa Proteus[®], onde são feitas todas as ligações e definições dos elementos a serem utilizados, facilitando no momento de montar a placa de circuito impresso, pois no Ares deste programa (parte onde se constrói a PCI) é possível definir os tamanhos das placas (*edge*) e então usar o comando *auto-placer* para posicionamento automático dos elementos dentro dos limites estipulados, fazendo alguns ajustes posteriores.

Ainda no programa, como é visto na Figura 68, configura-se a função *auto-router* (guia *design rule manager*) usando na *Net Class* (*POWER* e *SIGNAL*) as configurações presentes na Imagem (a), onde as trilhas ficam relativamente largas e são feitas todas as trilhas possíveis na parte de baixo (*Bottom Cooper*), sendo realizadas posteriormente, as ligações na parte de cima (*Top Cooper*) de forma manual. Após a confecção das trilhas, são exportados os arquivos, em imagem/PDF, para impressão à laser no papel fotográfico, criando neste papel uma cera que quando em contato com a placa e sob 200°C (ferro de passar ou prensa térmica) transfere o desenho para a mesma. Para a realização da segunda placa é exportado o arquivo *Gerber*, que contém as informações necessárias para a fresa S63 confeccionar as trilhas e furar corretamente.

Para a primeira PCI foi feito o processo manual presente nas Figuras 69, 70, 71, 72 e 73, respectivamente, transferência dos desenhos para a placa usando um ferro de passar com folha sulfite protegendo a placa de ficar em contato direto e com fita crepe para segurar o papel fotográfico; apresentação e correção das falhas das trilhas devido à má transferência do desenho, usando uma caneta de marcação permanente de ponta fina; corrosão do cobre usando a solução de percloro (ácido), verificando durante a agitação até completa remoção do cobre nas áreas fora das trilhas; furação usando uma furadeira e broca de 1mm, centralizando do furo a fim de facilitar a entrada dos componentes; finalização da placa posicionando um a um e realizando a solda, também colocando fios para realizar as ligações faltantes (que deveriam estar na parte de cima para placa dupla face).

Na segunda PCI, usando a prototipadora, tem-se as Figuras 74, 75 e 76, respectivamente, fabricação usando a máquina com placa de fibra de vidro dupla face; metalização

dos furos (ligações entre trilhas de ambos os lados) usando o kit *EasyContac*; finalização da placa posicionando os elementos e soldando os mesmos (algumas soldas na camada superior *Top layer*) e posterior envernização para proteção do cobre contra oxidação e possíveis curto circuitos acidentais.

O ovoscópio para visualização dos ovos está representado na Figura 77, onde a tampa (não mostrada) possui uma isolação contra a luz e um furo no meio, para que possa-se colocar o ovo no furo e visualizar dentro do mesmo.

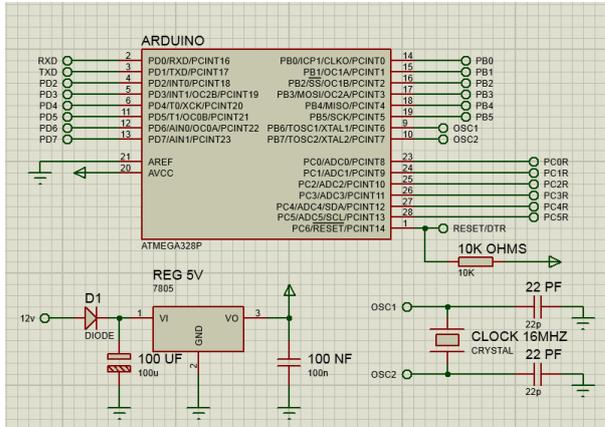
Por fim a montagem final da incubadora, representada pela Figura 78, onde é possível visualizar a disposição de todos os elementos dentro e fora da mesma.



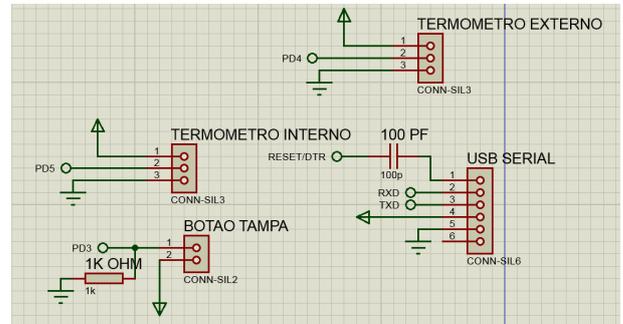
(a) Caixa de madeira - esqueleto da incubadora. (b) Bandeja para acomodação dos ovos.

Figura 66 – Elementos de madeira para sustentação principal.

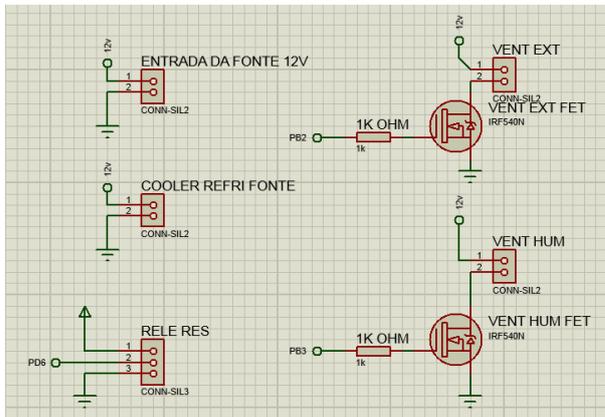
Fonte: Autoria própria.



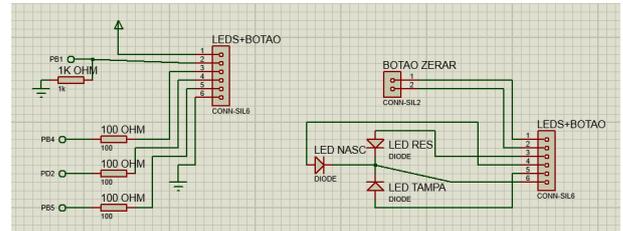
(a) Arduino + fonte de tensão + ressonador.



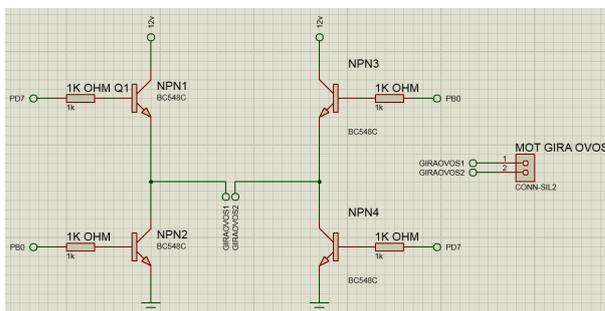
(b) Sensores de temp. + Botão da tampa + USB.



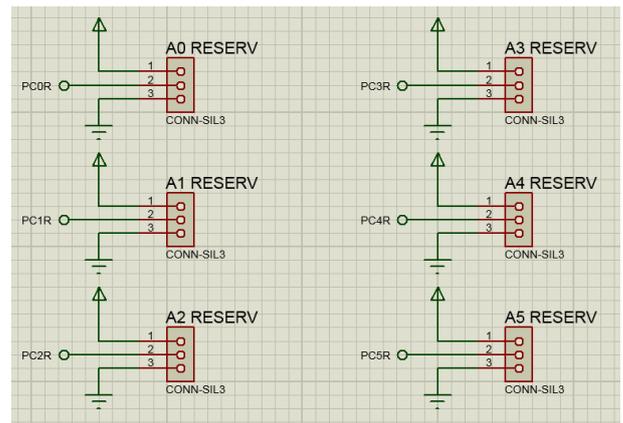
(c) Conectores + ventiladores.



(d) Leds de identificação + botão de zerar o tempo.



(e) Ponte H para giragem dos ovos.



(f) Circuitos reservados para possível expansão futura.

Figura 67 – Projeto do esquemático no programa Proteus®.

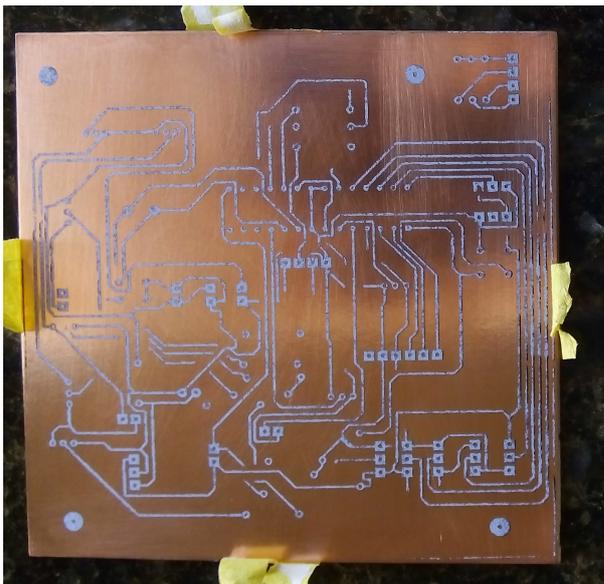
Fonte: Autoria própria.



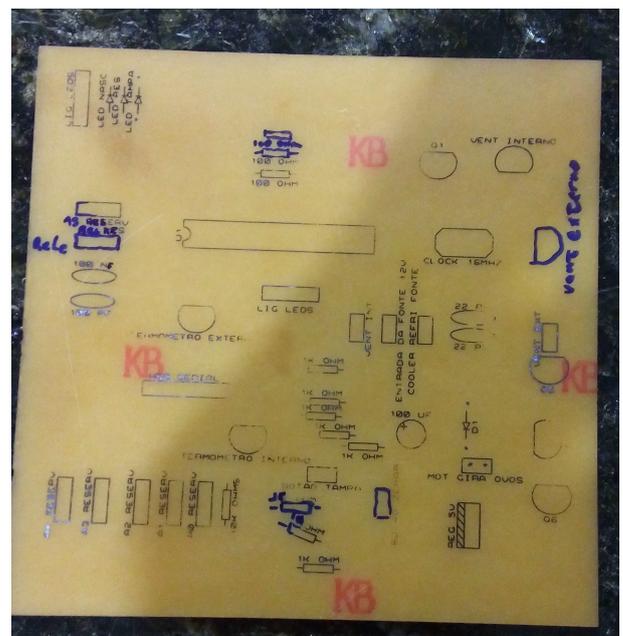
(a) Elementos utilizados para confecção.



(b) Fenolite + desenhos em papel fotogr fico. (c) Ferro de passar + fenolite envolto em papel sulfite.



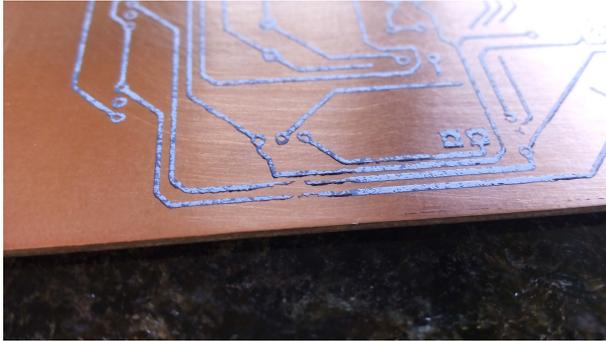
(d) Resultado das trilhas de cobre.



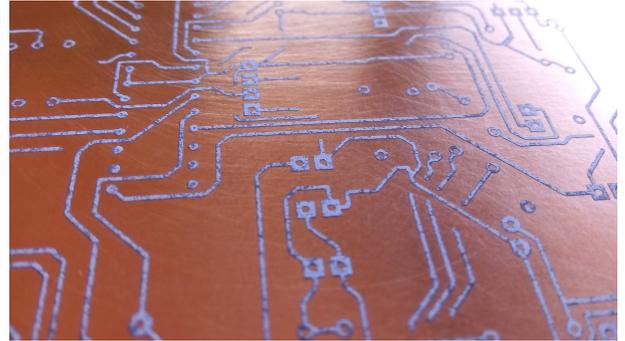
(e) Resultado das marcações dos componentes + correções de falhas.

Figura 69 – Confecção manual PCI.

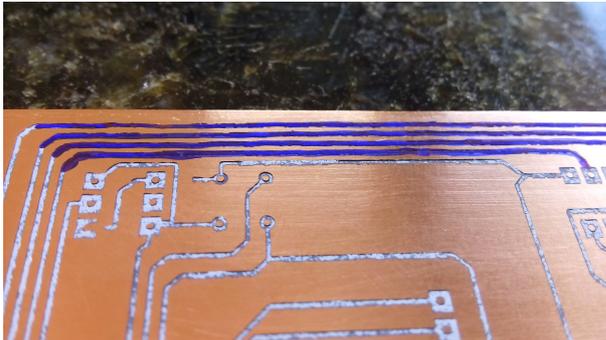
Fonte: Autoria pr pria.



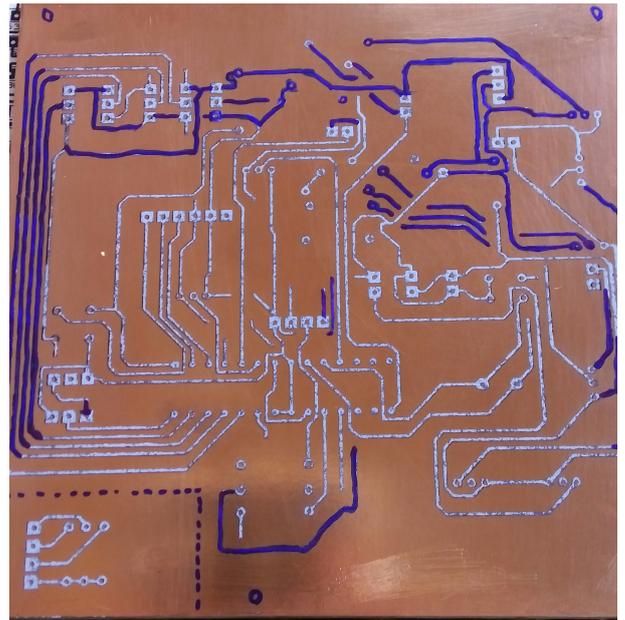
(a) Detalhe de falhas nas trilhas.



(b) Outras falhas nas trilhas.



(c) Correção das falhas com caneta de marcação permanente.



(d) Todas as falhas corrigidas + aumento da espessura de trilhas finas (devido a falhas).

Figura 70 – Correção das falhas nas trilhas.

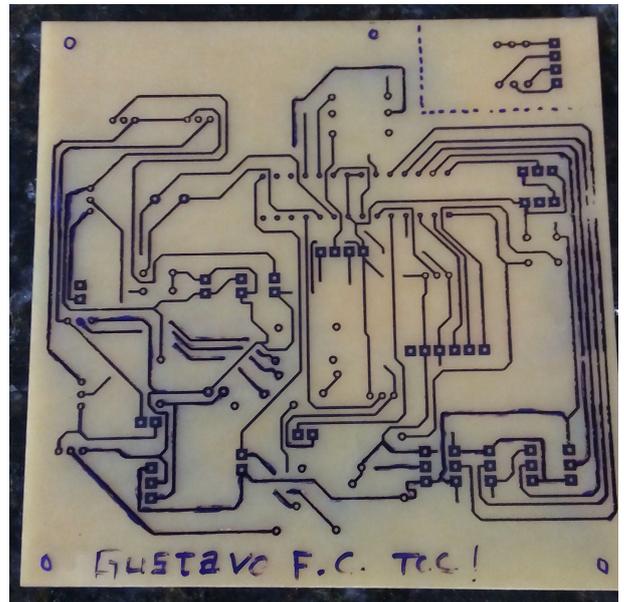
Fonte: Autoria própria.



(a) Início da corrosão (cobre avermelhado).



(b) Terminando a corrosão (cor do fenolite por quase toda a placa).



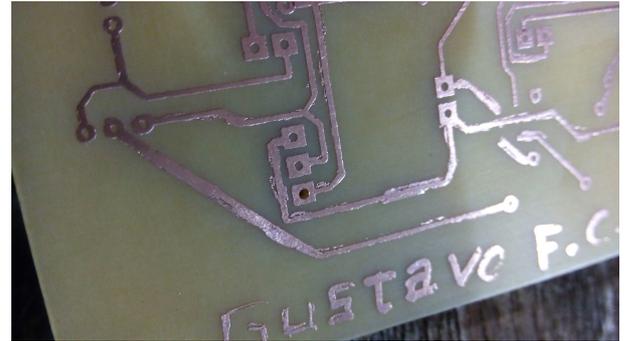
(c) Corrosão finalizada (cobre apenas nas trilhas).

Figura 71 – Corrosão da placa de cobre.

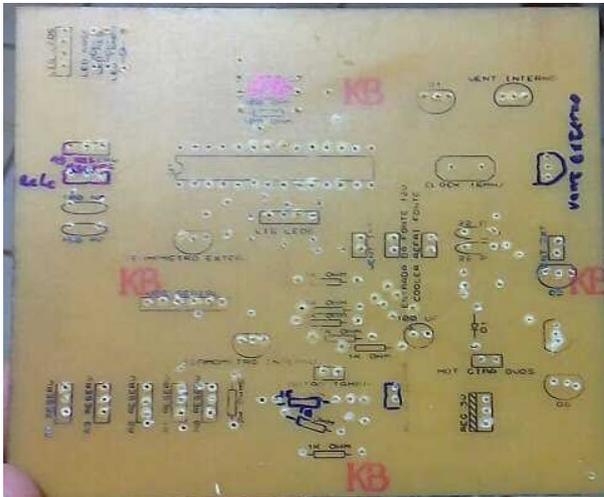
Fonte: Autoria própria.



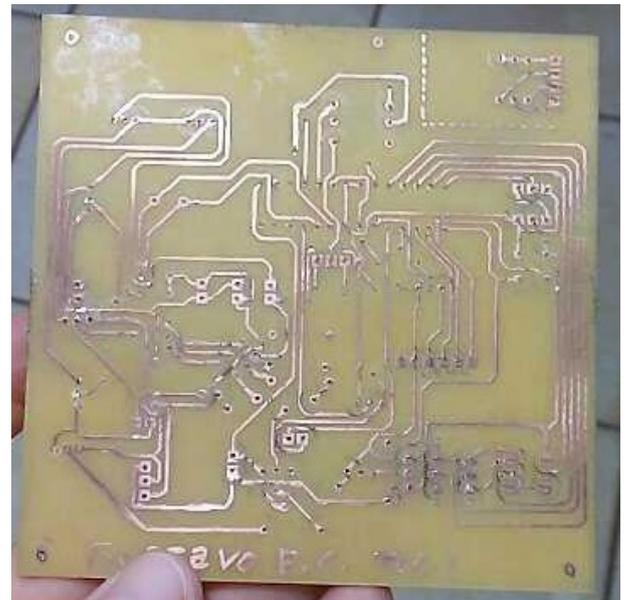
(a) Broca de 1mm para furar.



(b) Exemplo de furo centralizado.



(c) Resultado dos furos na parte superior.



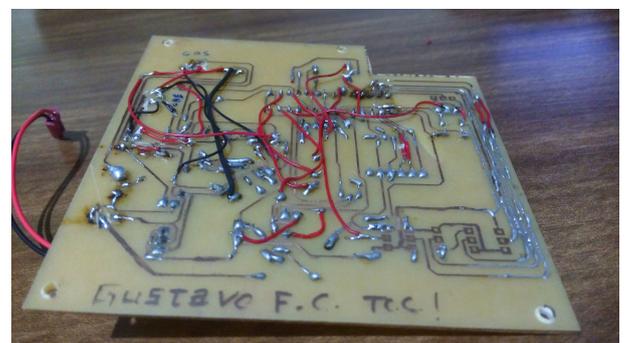
(d) Resultado dos furos na parte inferior.

Figura 72 – Furos da PCI.

Fonte: Autoria própria.



(a) Posicionamento dos componentes.



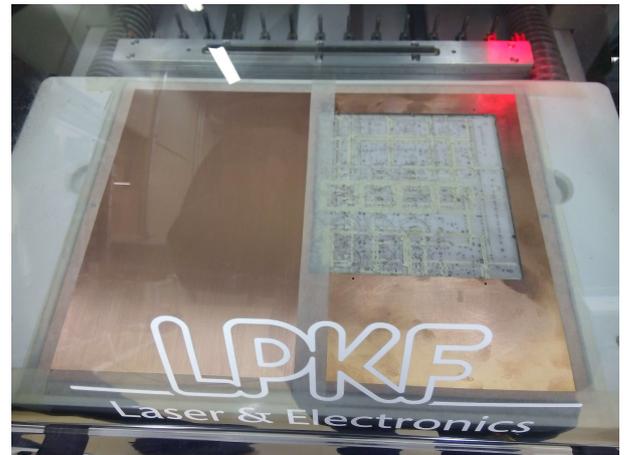
(b) Solda e ligações faltantes.

Figura 73 – Montagem final da PCI.

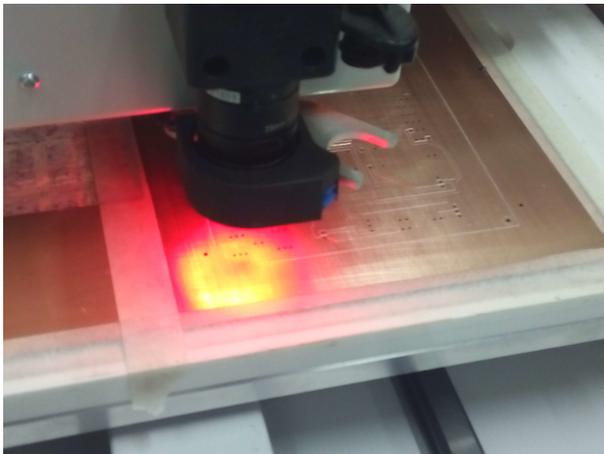
Fonte: Autoria própria.



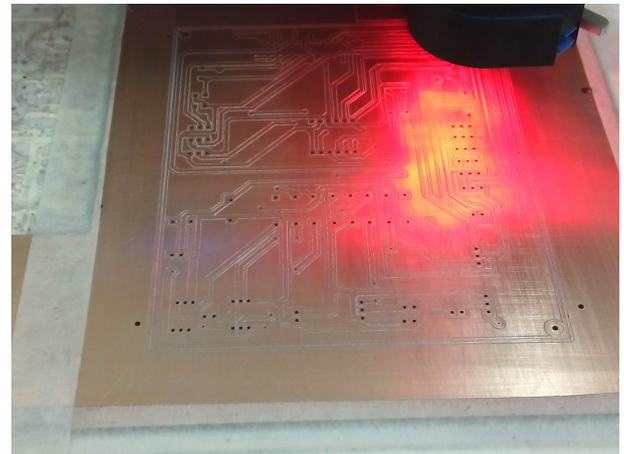
(a) LPKF S63 - Prototipadora.



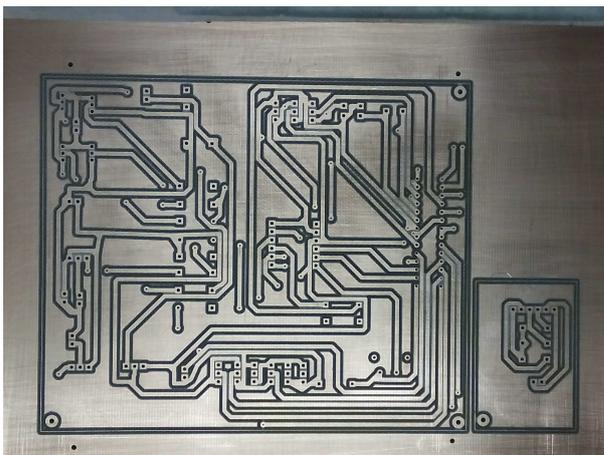
(b) Placa de fibra de vidro de dupliface.



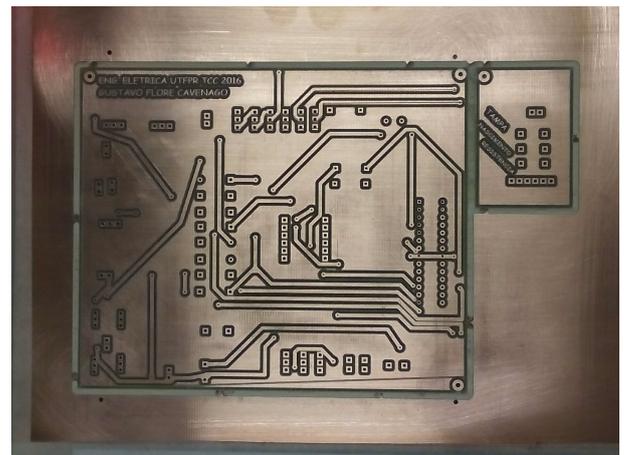
(c) Processo de desbaste do cobre mais furos na placa (*bottom*).



(d) Processo de desbaste do cobre (*top*).



(e) Placa pronta (*bottom*).



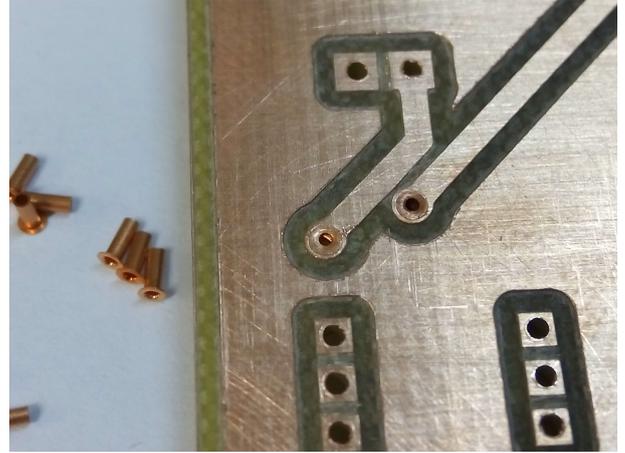
(f) Placa pronta (*top*).

Figura 74 – Produção da segunda PCI.

Fonte: Autoria própria.



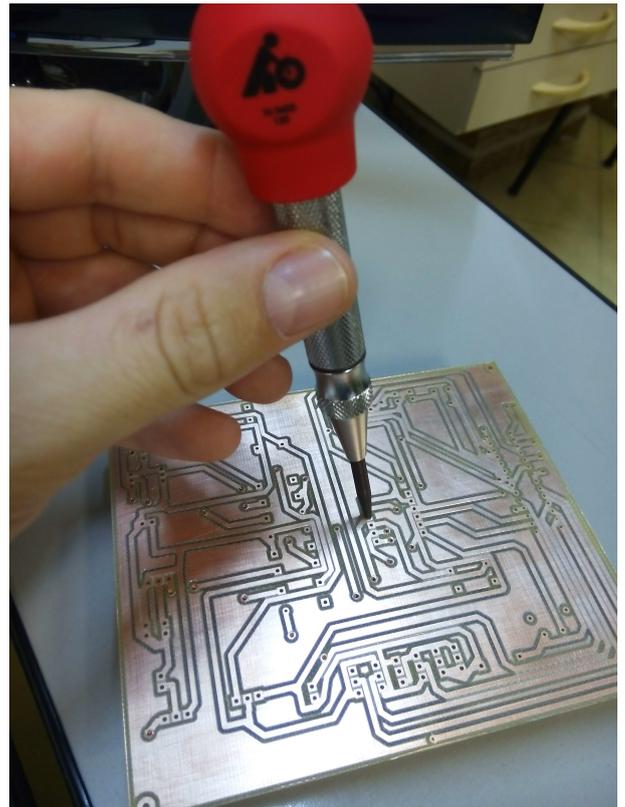
(a) LPKF EasyContac 0,6mm.



(b) Posicionamento dos elementos de cobre nos furos.



(c) Base e presa para fixação dos metalizadores.



(d) Fixação pressionado a punção do lado oposto.

Figura 75 – Metalização dos furos.

Fonte: Autoria própria.



Figura 77 – Montagem do ovoscópio (lâmpada 40W, soquete, cabo e plugue macho para tomada).

Fonte: Autoria própria.



(a) Parte de dentro sem gaveta.



(b) Parte traseira.

Figura 78 – Montagem final da incubadora.

Fonte: Autoria própria.

APÊNDICE B – CÓDIGOS DE PROGRAMAÇÃO - MATLAB® E ARDUINO®

Neste apêndice, se encontra o código referente ao controlador escolhido (PP), bem como sua função utilizada pelo AG e o código do Arduino. Sendo os mesmos autoexplicativos, devido aos comentários presentes. Constando sempre a última data de modificação.

B.1 CONTROLADOR PI

```

19 title('Temperatura de saída vs
      entrada de energia');
20 ylabel('Temperatura (°C)');
21 xlabel('Tempo (min)');
22 legend('Temperatura transladada', '
      Energia de entrada')
23 figure
24 plot(s./60,10*ones(length(s),1),'g'
      ,s./60,tint1,s./60,us.*10,'r');
25 title('Temperatura de saída vs
      entrada de energia');
26 ylabel('Temperatura (°C)');
27 xlabel('Tempo (min)');
28 legend('100% PWM', 'Temperatura
      transladada', 'Energia de entrada
      ')
29 %% MODELAGEM
30 tf1 = tfest(data,1,0); %função
      transferencia de primeira ordem
31 [saida1, fit1] = compare(data,tf1);
      %compara o sistema identificado
      com o medido, usando o mesmo
      sinal de entrada do sistema
32 %aplicando um degrau na função
      estimada
33 figure
34 plot(saida1.s./60,saida1.y,'b',s2
      ./60,tint12,'g')
35 title(['Topologia 1 polo e 0 zero (
      primeira ordem) com fit de: ',
      num2str(tf1.report.fit.
      FitPercent),'%'])
36 ylabel('Temperatura (°C)')
37 xlabel('Tempo (s)')
38 legend('Resposta do sistema
      modelado', 'Resposta real ao
      r');
1 % Gustavo Flore Cavenago Ra:1578375
2 % Atualização 22/09/2016
3 %% LEITURA DOS DADOS (EXCEL)
4 s = xlsread('aquecimento', '
      teste_pid1', 'e:e'); %tempo
5 s = s - min(s);
6 tint = xlsread('aquecimento', '
      teste_pid1', 'g:g'); %temperatura
      interna
7 tint1 = tint - min(tint); %
      colocando a referencia temporal
      em zero
8 us = xlsread('aquecimento', '
      teste_pid1', 'p:p'); %temperatura
      interna
9 s2 = xlsread('aquecimento', '
      ESQUENTANDO', 'e:e'); %tempo
10 s2 = s2 - min(s2);
11 tint2 = xlsread('aquecimento', '
      ESQUENTANDO', 'g:g'); %
      temperatura interna
12 tint12 = tint2 - min(tint2); %
      colocando a referencia temporal
      em zero
13 us2 = ones(length(s2),1);
14 %% CONTRUÇÃO DOS DADOS
15 data = iddata(tint12,us2,2); %
      resposta ao degrau
16 data2 = iddata(tint1,us); %resposta
      para entrada variante no tempo
17 figure
18 plot(s2./60,tint12,s2./60,us2.*40,'
      r');

```

```

    degrau')
39 tf1 %mostrar a função transferencia
    obtida
40 tf2 = tfest(data,2,1); %função
    transferencia de segunda ordem
41 [saida3, fit3] = compare(data, tf2);
    %compara o sistema identificado
    com o medido, usando o mesmo
    sinal de entrada do sistema
42 figure
43 plot(s2./60, tint12, 'g', saida3.s
    ./60, saida3.y, 'b')
44 title(['Topologia 2 zeros e 1 polos
    (primeira ordem) com fit de: '
    num2str(fit3), '%'])
45 ylabel('Temperatura (°C)')
46 xlabel('Tempo (s)')
47 legend('Resposta real ao degrau',
    'Resposta do sistema modelado')
48 tf2
49 tf3 = tfest(data2,2,1); %função
    transferencia de segunda ordem
50 [saida3, fit3] = compare(data2, tf3);
    %compara o sistema identificado
    com o medido, usando o mesmo
    sinal de entrada do sistema
51 figure
52 plot(s./120, tint1, 'g', saida3.s./60,
    saida3.y, 'b')
53 title(['Topologia 2 zeros e 1 polos
    (primeira ordem) com fit de: '
    num2str(fit3), '%'])
54 ylabel('Temperatura (°C)')
55 xlabel('Tempo (s)')
56 legend('Resposta real de entrada
    variável', 'Resposta do sistema
    modelado')
57 tf3
58
59 tf4 = procest(data2, 'p1d'); %função
    transferencia de segunda ordem
60 [saida4, fit4] = compare(data2, tf4);
    %compara o sistema identificado
    com o medido, usando o mesmo
    sinal de entrada do sistema
61 figure
62 plot(s./120, tint1, 'g', saida4.s./60,
    saida4.y, 'b')
63 title(['Topologia primeira ordem
    com atraso com fit de: ', num2str
    (fit4), '%'])
64 ylabel('Temperatura (°C)')
65 xlabel('Tempo (s)')
66 legend('Resposta real de entrada
    variável', 'Resposta do sistema
    modelado')
67 tf4
68 %% Determinação dos Parâmetros kp e
    ki – Ziegler–Nichols
69 %página 697 ogata
70 kpzn = .9*tf4.Tp1/tf4.Td;
71 kizn = kpzn*.3/tf4.Td;
72 pole(feedback(series(tf3, tf([kpzn
    kizn]/55.6719,[1 0])),1))
73 zero(feedback(series(tf3, tf([kpzn
    kizn]/55.6719,[1 0])),1))
74 figure
75 rlocus(feedback(series(tf3, tf([kpzn
    kizn]/55.6719,[1 0])),1))
76 figure
77 step(feedback(series(tf3, tf([kpzn
    kizn]/55.6719,[1 0])),1))
78 %% Determinação dos Parâmetros kp,
    ki e ka – Algoritmo genético
79 if 0 %mudar pra 1 para rodar o
    algoritmo
80 %definição dos parâmetros
81 nvars=3; %numero de variáveis
82 lb = [0 0 0]; %lower bounds
    limite inferior +- 100%
83 ub = [kpzn*2 kizn*2 kizn*2/2];
    %upper bounds limite
    superior Universo de
    discurso
84 SizePop=200; %numero de
    individuos inicial
85 NumGeracoes = 20; %numero de
    gerações
86 options = gaoptimset('
    PopulationType','
    DoubleVector',
    'PopulationSize',...

```

```

87     SizePop, 'Generations',      114 plot(temp_int.time./60,temp_int.
        NumGeracoes, '          data,temp_int.time./60,37.6*ones
        StallGenLimit',          (1,length(temp_int.time)), 'r')
        NumGeracoes, 'PlotFcns'  115 title('Temperatura de saída (
        ,...                       interna da incubadora) – PI ZN')
88     @gaplotbestf, 'EliteCount'   ;
        ,.1*SizePop, '          116 ylabel('Temperatura (°C)')
        CrossoverFraction',.2)%117 xlabel('Tempo (min)')
        ,...                       118 legend('Temperatura Interna', '
89     %'MutationFcn',              Referência')
        @mutationgaussian);      119 figure
90     [x fval ef out pop score]= ga(120 plot(temp_energia1.time./60,
        @gx,nvars,[],[],[],[],lb,ub temp_energia1.data, 'r',
       ,[],options);            temp_energia.time./60,
91     disp(['Os melhores valores são: temp_energia.data, '.')
        kp= ',num2str(x(1)), ' ki= 121 title('Energias de controle – PI ZN
        ,...                       ');
92     num2str(x(2)), ' ka= ',      122 ylabel('PWM (0 à 1)')
        num2str(x(3))]);          123 xlabel('Tempo (min)')
93     save pi                       124 legend('Pré saturação', 'Pós
94     end                            saturação')
95     %% Plotagem e verificação das 125 y=trapz(temp_int.time,((temp_ref–
        especificações PI convencional temp_int.data).^2).*temp_int.
96     load pi %para não precisar rodar o time)
        AG a todo momento        126 %% Plotagem e verificação das
97     kp=x(1);                       especificações PI Ajustado com
98     ki=x(2);                       AG
99     ka=x(3);                       127 load pi %para não precisar rodar o
100    %kp= 8.3341; ki= 0.032829; ka=  AG a todo momento
        0.01135; MELHOR          128 kp=x(1);
101    kp= kpzn; ki= kizn; ka= kizn/10; 129 ki=x(2);
102    st = 60; %sample time (do controle 130 ka=x(3);
        contínuo)                131 %kp= 22.2758 ki= 0.11143 ka=
103    temp_ini=20;                    0.013403; MELHOR
104    temp_ref = 37.6;%temperatura ideal 132 %kp= kpzn; ki= kizn; ka= kizn/10;
        até o 18° dia, depois 36.8 133 st = 60; %sample time (do controle
105    tempo_sim = 1*60*60; % 20 minutos contínuo)
106    load('pi','tf3');              134 temp_ini=20;
107    run simulacao_pi.slx            135 temp_ref = 37.6;%temperatura ideal
108    options = simset('SrcWorkspace', até o 18° dia, depois 36.8
        'Current');              136 tempo_sim = 1*60*60; % 20 minutos
109    sim('simulacao_pi.slx',tempo_sim, 137 load('pi','tf3');
        options);                138 run simulacao_pi.slx
110    temp_intzn = temp_int;          139 options = simset('SrcWorkspace', '
111    temp_energiazn = temp_energia; Current');
112    temp_energia1zn = temp_energia1; 140 sim('simulacao_pi.slx',tempo_sim,
113    figure                          options);

```

```

141 temp_intag = temp_int;
142 temp_energiaag = temp_energia;
143 temp_energia1ag = temp_energia1;
144 figure
145 plot(temp_int.time./60,temp_int.
    data,temp_int.time./60,37.6*ones
    (1,length(temp_int.time)), 'r')
146 title('Temperatura de saída (
    interna da incubadora) – PI AG')
    ;
147 ylabel('Temperatura (°C)')
148 xlabel('Tempo (min)')
149 legend('Temperatura Interna', '
    Referência')
150 figure
151 plot(temp_energia1.time./60,
    temp_energia1.data, 'r',
    temp_energia.time./60,
    temp_energia.data, '.')
152 title('Energias de controle – PI AG
    ');
153 ylabel('PWM (0 à 1)')
154 xlabel('Tempo (min)')
155 legend('Pré saturação', 'Pós
    saturação')
156 y=trapz(temp_int.time,((temp_ref–
    temp_int.data).^2).*temp_int.
    time)
157 figure
158 plot(temp_intzn.time./60,temp_intzn
    .data,temp_intag.time./60,
    temp_intag.data,temp_int.time
    ./60,37.6*ones(1,length(temp_int
    .time)), 'r')
159 title('Comparação das Temperaturas
    internas ZN e AG');
160 ylabel('Temperatura (°C)')
161 xlabel('Tempo (min)')
162 legend('ZN', 'AG')

B.1.1 Função custo para o AG

1 function y = gx(x)
2 kp=x(1);
3 ki=x(2);
4 ka=x(3);
5 load('pi', 'tf3')

6 st = 60; %sample time (do controle
    contínuo)
7 temp_ini=10;
8 temp_ref = 37.6;%temperatura ideal
    até o 18° dia, depois 36.8
9 tempo_sim = 30*60;
10 load('aquisicao', 'tf1');
11 run simulacao_pi.slx
12 options = simset('SrcWorkspace', '
    Current');
13 sim('simulacao_pi.slx',tempo_sim,
    options);
14 y=trapz(temp_int.time,((temp_ref–
    temp_int.data).^2).*temp_int.
    time); %integral com quadrado
    para eliminar a parte negativa e
    dar peso maior para maiores
    erros
15 end

B.2 CONTROLADOR PP

1 % Gustavo Flore Cavenago Ra:1578375
2 % Atualização 24/09/2016
3 close all; clear; clc;
4 %% Valores das funções pertinências
    ajustadas com AG
5 if 0 %mudar pra 1 para rodar o
    algoritmo
6 tic %tic toc mede o tempo que o
    programa leva pra realizar
    a operação entre os termos
    %definição dos parâmetros
7 lb = 0; %lower bounds limite
    inferior
8 ub = .3; %upper bounds liminte
    superior Universo de
    discurso
9 nvars=length(lb); %numero de
    variáveis
10 SizePop=30; %numero de
    individuos inicial
11 NumGeracoes = 10; %numero de
    gerações
12 options = gaoptimset('
    PopulationType', '
    DoubleVector', '

```

```

PopulationSize', ...
14     SizePop, 'Generations',
        NumGeracoes, '
        StallGenLimit',
        NumGeracoes, 'PlotFcns'
        , ...
15     @gaplotbestf, 'EliteCount'
        ,.1*SizePop, '
        CrossoverFraction'
        ,.1, ...
16     'MutationFcn',
        @mutationgaussian);
17     [x fval ef out pop score]= ga(
        @gxp,nvars,[],[],[],[],lb,ub,
       ,[],options);
18     save pp
19     toc
20 end
21 %% Ajustando os parâmetros
    encontrados e simulando para
    virificar a resposta
22 load pp
23 k=x;
24 st = 60; %sample time (do controle
    contínuo)
25 temp_ini=20;
26 temp_ext = temp_ini;
27 temp_ref = 37.6;%temperatura ideal
    até o 18° dia, depois 36.8
28 tempo_sim = 1*60*60; % 1hora
29 load('pi','tf3');
30 run simulacao_pp.slx
31 options = simset('SrcWorkspace','
    Current');
32 sim('simulacao_pp.slx',tempo_sim,
    options);
33 %%
34 save pp
35 figure
36 plot(temp_int.time./60,temp_int.
    data,temp_int.time./60,37.6*ones
    (1,length(temp_int.time)), 'r')
37 title('Temperatura de saída – PP AG
    ');
38 ylabel('Temperatura (°C)')
39 xlabel('Tempo (min)')
40 legend('PP AG','Referência')

41 figure
42 plot(temp_energia1.time./60,
    temp_energia1.data,'r',
    temp_energia.time./60,
    temp_energia.data,temp_energia2.
    time./60,temp_energia2.data)
43 title('Energias de controle – PP AG
    ');
44 ylabel('PWM (0 à 1)')
45 xlabel('Tempo (min)')
46 legend('P do erro da referência','
    Pós saturação','P da dif. da
    temp. externa')
47 load('pi','temp_intag')
48 load('fu','temp_intfu')
49 figure
50 plot(temp_intag.time./60,temp_intag
    .data,temp_intfu.time./60,
    temp_intfu.data,temp_int.time
    ./60,temp_int.data,temp_int.time
    ./60,37.6*ones(1,length(temp_int
    .time)), 'm')%, 'linewidth',2)
51 title('Comparação das Temperaturas
    internas');
52 ylabel('Temperatura (°C)')
53 xlabel('Tempo (min)')
54 legend('PI-AG','FUZZY AG','PP AG',
    'REFERÊNCIA')

```

B.2.1 Função custo para o AG

```

1 function y = gxp(x)
2 k=x;
3 st = 60; %sample time (do controle
    contínuo)
4 temp_ini=20;
5 temp_ext = temp_ini;
6 temp_ref = 37.6;%temperatura ideal
    até o 18° dia, depois 36.8
7 tempo_sim = 1*60*60; % 1hora
8 load('pi','tf3');
9 run simulacao_pp.slx
10 options = simset('SrcWorkspace','
    Current');
11 sim('simulacao_pp.slx',tempo_sim,
    options);
12 y=trapz(temp_int.time,((temp_ref-
    temp_int.data).^2).*temp_int.

```

```

time); %integral com quadrado
para eliminar a parte negativa
dar peso maior para maiores
erros com o passar do tempo
13 end
B.3 CONTROLADOR FUZZY
1 % Gustavo Flore Cavenago Ra:1578375
2 % Atualização 24/09/2016
3 %faz-se primeiramente um protótipo
do fuzzy na toolbox.
4 close all; clear; clc;
5 %% Determinação dos parâmetros
Ajuste manual
6 li = -50; %limites inferior e
superior
7 ls = 40;
8 %Determinação da entrada INPUT
9 y = [1 50];
10 z = [50 1 1 1 1 1];
11 x1(1) = li+z(1);
12 x1(2) = 0;
13 x1(3) = 0;
14 a=0;
15 for i=4:8
16     if rem(i,3)==0
17         x1(i) = a - z(i-2);
18         a=x1(i);
19     else
20         x1(i) = a + z(i-2);
21         a=x1(i);
22     end
23 end
24 par = [li li x1 ls-sum(y(1:2)) ls-y
(2) ls ls]; %todos os parâmetros
das funções pertinência
25 if par(3)>0
26     par(3)=0;
27 end
28 for i=6:length(par)
29     if par(i)<0
30         par(i)=0;
31     end
32     if par(i)>ls
33         par(i)=ls;
34     end
35 end
36 %determinou-se a não necessidade de
mais de 1 função na parte
negativa
37 %Determinação da saída OUTPUT
38 t = .25*ones(1,7);
39 parout = [0 0 t(1) sum(t(1:2))...
40     sum(t(1:2))-t(3) sum(t(1:4))-2*
t(3) sum(t(1:5))-2*t(3) ...
41     sum(t(1:5))-2*t(3)-t(6) sum(t
(1:7))-2*t(3)-2*t(6) 1 1 ...
42     -1 -1 -1];
43 for i=1:length(parout)
44     if parout(i)<0
45         parout(i)=0;
46     end
47     if parout(i)>1
48         parout(i)=1;
49     end
50 end
51 %preenchimento das funções
pertinências
52 matFuzzy = readfis('ajmanual.fis');
%Lê o arquivo fuzzy.fis
53 matFuzzy.input.range = [li ls];
54 %para input
55 a=0;
56 for i=1:4
57     b=length(matFuzzy.input.mf(1, i)
).params);
58     matFuzzy.input.mf(1, i).params
= par(a+1:a+b);
59     a=a+b;
60 end
61 % para output
62 a=0;
63 for i=1:4
64     b=length(matFuzzy.output.mf(1,
i).params);
65     matFuzzy.output.mf(1, i).params
= parout(a+1:a+b);
66     a=a+b;
67 end
68 save fu
69 %regras foram preenchidas a mão na
toolbox

```



```

101         x=[20 20 0 2 2 2 2 1 .3 0 125
            .1 1 1 1 1 .15]; % MEUS126
            PARÂMETROS AJUSTADOS 127
            MANUALMENTE 128
102     else 129
103         %load('fu','x'); %ultimo x130
            encontrado pelo ag 131
104         x = [17.8547285241353 132
            19.4315409350815 133
            0.383982203939155 134
            2.90556777513905 135
            1.42206842295596 136
            2.21119292823393
            0.974200153178055
            0.161263042559407 137
            0.0531023287891157 138
            0.0475366528023041 139
            0.129979724962128
            0.896077167205572 140
            0.961172925610557
            0.388148728610636
            0.720630727312426 141
            0.118621542730570];
105     end
106     %Determinação da entrada INPUT 142
107     li = -50; %limites inferior e 143
            superior 144
108     ls = 40; 145
109     y = x(1:2); 146
110     z = x(3:8); 147
111     x1(1) = li+z(1); 148
112     x1(2) = 0; 149
113     x1(3) = 0; 150
114     a=0; 151
115     for i=4:8
116         if rem(i,3)==0 152
117             x1(i) = a - z(i-2);
118             a=x1(i);
119         else 153
120             x1(i) = a + z(i-2); 154
121             a=x1(i); 155
122         end 156
123     end 157
124     par = [li li x1 ls-sum(y(1:2))
            ls-y(2) ls ls]; %todos os 158
            parâmetros das funções
            pertinência 159
            if par(3)>0
                par(3)=0;
            end
            for i=6:length(par)
                if par(i)<0
                    par(i)=0;
                end
                if par(i)>ls
                    par(i)=ls;
                end
            end
            %determinou-se a não
            necessidade de mais de 1
            função na parte negativa
            %Determinação da saída OUTPUT
            t = x(9:end);
            parout = [0 t(1) sum(t(1:2))
                sum(t(1:2)) ...
                sum(t(1:2))-t(3) sum(t(1:4))
                )-2*t(3) sum(t(1:5))-2*t
                (3) ...
                sum(t(1:5))-2*t(3)-t(6) sum
                (t(1:7))-2*t(3)-2*t(6) 1
                1 ...
                -t(end) -t(end) 0];
            for i=1:length(parout)
                if parout(i)<-1
                    parout(i)=-1;
                end
                if parout(i)>1
                    parout(i)=1;
                end
            end
            % preenchimento das funções
            pertinências
            matFuzzy = readfis('ajmanual.
                fis'); %Lê o arquivo fuzzy.
                fis
            matFuzzy.input.range = [li ls];
            %para input
            a=0;
            for i=1:4
                b=length(matFuzzy.input.mf
                    (1, i).params);
                matFuzzy.input.mf(1, i).
                    params = par(a+1:a+b);
                a=a+b;

```

```

160     end                                191     figure
161     % para output                        192     plot(temp_energia1.time
162     matFuzzy.output.range = [-1 1];      ./.60,temp_energia1.data ,
163     a=0;                                  'r',temp_energia.time
164     for i=1:4                              ./.60,temp_energia.data , '
165         b=length(matFuzzy.output.mf      .' ,temp_energia2.time
166             (1, i).params);              ./.60,temp_energia2.data)
167         matFuzzy.output.mf(1, i). 193     title('Energias de controle
168             params = parout(a+1:a+b)      - FUZZY MANUAL + P');
169         ;                                  194     ylabel('PWM (0 à 1)')
170         a=a+b;                             195     xlabel('Tempo (min)')
171     end                                    196     legend('Parcela
172     st = 60; %sample time (do            Proporcional', 'Pós
173         controle contínuo)                saturação', 'Fuzzy')
174     temp_ini=20;                          197     end
175     temp_ext = temp_ini;                   198     end
176     temp_ref = 37.6;%temperatura          199     figure
177     ideal até o 18° dia, depois 200     plot(temp_int.time./60,temp_int.
178     36.8                                    data,temp_int.time./60,37.6*ones
179     tempo_sim = 1*60*60; % 1hora           (1,length(temp_int.time)), 'r')
180     load('pi','tf3');                     201     title('Temperatura de saída – FUZZY
181     run simulacao_fuzzy.slx                AG');
182     options = simset('SrcWorkspace' 202     ylabel('Temperatura (°C)')
183         , 'Current');                     203     xlabel('Tempo (min)')
184     sim('simulacao_fuzzy.slx',          204     legend('FUZZY AG', 'Referência')
185         tempo_sim,options);              205     figure
186     if aa==1                               206     plot(temp_energia1.time./60 ,
187         temp_intmeu = temp_int;            temp_energia1.data , 'r' ,
188         temp_energiameu =                 temp_energia.time./60 ,
189             temp_energia;                  temp_energia.data , '.' ,
190         temp_energia2meu =                 temp_energia2.time./60 ,
191             temp_energia2;                  temp_energia2.data)
192         temp_energia1meu =                 207     title('Energias de controle – FUZZY
193             temp_energia1;                  AG + P');
194         surfview(matFuzzy)                 208     ylabel('PWM (0 à 1)')
195         ruleview(matFuzzy)                 209     xlabel('Tempo (min)')
196         figure                             210     legend('Parcela Proporcional', 'Pós
197         plot(temp_int.time./60 ,           saturação', 'Fuzzy')
198             temp_int.data,temp_int. 211     load('pi','temp_intzn','temp_intag'
199             time./60,37.6*ones(1,          )
200             length(temp_int.time))) 212     figure
201         title('Temperatura de saída 213     plot(temp_intag.time./60,temp_intag
202             – FUZZY MANUAL');              .data,temp_intmeu.time./60 ,
203         ylabel('Temperatura (°C)')         temp_intmeu.data,temp_int.time
204         xlabel('Tempo (min)')              ./.60,temp_int.data,temp_int.time
205         legend('Fuzzy Manual', '          ./.60,37.6*ones(1,length(temp_int
206             Referência')                    .time)), 'm')%, 'linewidth',2)

```

```

214 title('Comparação das Temperaturas
internas');
215 ylabel('Temperatura (°C)')
216 xlabel('Tempo (min)')
217 legend('PI-AG', 'FUZZY MANUAL', '
FUZZY AG', 'REFERÊNCIA')
218 surfview(matFuzzy)
219 ruleview(matFuzzy)

B.3.1 Função custo para o AG

1 function y = gxf(x)
2 matFuzzy = readfis('ajmanual.fis');
    %Lê o arquivo fuzzy.fis
3 %Determinação da entrada INPUT
4 li = -50; %limites inferior e
    superior
5 ls = 40;
6 y = x(1:2);
7 z = x(3:8);
8 x1(1) = li+z(1);
9 x1(2) = 0;
10 x1(3) = 0;
11 a=0;
12 for i=4:8
13     if rem(i,3)==0
14         x1(i) = a - z(i-2);
15         a=x1(i);
16     else
17         x1(i) = a + z(i-2);
18         a=x1(i);
19     end
20 end
21 par = [li li x1 ls -sum(y(1:2)) ls -y
    (2) ls ls]; %todos os parâmetros
    das funções pertinência
22 if par(3)>0
23     par(3)=0;
24 end
25 for i=6:length(par)
26     if par(i)<0
27         par(i)=0;
28     end
29     if par(i)>ls
30         par(i)=ls;
31     end
32 end

33 %determinou-se a não necessidade de
    mais de 1 função na parte
    negativa
34 %Determinação da saída OUTPUT
35 t = x(9:end);
36 parout = [0 t(1) sum(t(1:2)) sum(t
    (1:2)) ...
37     sum(t(1:2))-t(3) sum(t(1:4))-2*
    t(3) sum(t(1:5))-2*t(3) ...
38     sum(t(1:5))-2*t(3)-t(6) sum(t
    (1:7))-2*t(3)-2*t(6) 1 1 ...
39     -t(end) -t(end) 0];
40 for i=1:length(parout)
41     if parout(i)<=-1
42         parout(i)=-1;
43     end
44     if parout(i)>1
45         parout(i)=1;
46     end
47 end
48 %preenchimento das funções
    pertinências
49 matFuzzy.input.range = [li ls];
50 %para input
51 a=0;
52 for i=1:4
53     b=length(matFuzzy.input.mf(1, i)
    .params);
54     matFuzzy.input.mf(1, i).params
    = par(a+1:a+b);
55     a=a+b;
56 end
57 % para output
58 matFuzzy.output.range = [-1 1];
59 a=0;
60 for i=1:4
61     b=length(matFuzzy.output.mf(1,
    i).params);
62     matFuzzy.output.mf(1, i).params
    = parout(a+1:a+b);
63     a=a+b;
64 end
65 %%
66 st = 60; %sample time (do controle
    contínuo)
67 temp_ini=20;
68 temp_ext = temp_ini;

```

```

69 temp_ref = 37.6;%temperatura ideal até o 18° dia, depois 36.8
70 tempo_sim = 1*60*60; % 1hora
71 load('pi','tf3');
72 run simulacao_fuzzy.slx
73 options = simset('SrcWorkspace','Current');
74 sim('simulacao_fuzzy.slx',tempo_sim, options);
75 y=trapz(temp_int.time,((temp_ref-temp_int.data).^2).*temp_int.time); %integral com quadrado para eliminar a parte negativa dar peso maior para maiores erros
76 end

B.4 CONTROLADOR GPC

1 % Gustavo Flore Cavenago Ra:1578375
2 % Atualização 06/10/2016
3 % Adaptado do professor Rodrigo Rodrigues Sumar
4 close all;clear;clc;
5 %% Ajustado a mão
6 load('pi','tf3');
7 sysc = tf(tf3.num,tf3.den);
8 [sysd] = c2d(sysc,60,'zoh'); % função transferencia para discreto(z)
9 sys = tf(sysd.num,sysd.den,60,'Variable','z^-1')
10 A = cell2mat(sys.den);
11 B = cell2mat(sys.num);
12 B = B(2:end); % B dividido por z-1
13 Alfa = [1 -1];
14 nit=60;N2=10;Nu=7;sigma=.5;
15 temp_ini = 20;ref=37.6-temp_ini; novaref=36.7-temp_ini;
16 % ——— CONDICÕES INICIAIS
17 na=size(A,2);
18 nb=size(B,2);
19 for k=1:nit+N2
20     y(k)=0;u(k)=0;du(k)=0;yp(k)=0;e(k)=0;
21 end
22 for j=1:N2
23     alfa(j)=0;
24 end
25 % ——— GERACAO DA REFERENCIA
26 yr(1:round(nit/2))=ref;
27 yr(round(nit/2)+1:nit+N2)=novaref;
28 % ——— CALCULO DE ALFA
29 alfa=conv(A,Alfa);
30 % ——— MONTAGEM DA MATRIZ M
31 auxb=[B zeros(1,N2)];
32 m(1)=auxb(1);
33 for j=2:N2
34     mini=min([j na]);
35     suma=0;
36     for i=2:mini
37         suma=suma-alfa(i)*m(j-i+1);
38     end
39     m(j)=auxb(j)+suma;
40 end
41 for j=1:N2
42     for i=1:Nu
43         if j>i
44             M(j,i)=m(j+1-i);
45         elseif j==i
46             M(j,i)=m(1);
47         else
48             M(j,i)=0;
49         end
50     end
51 end
52 % ——— CALCULO DE V
53 V=inv(M'*M+sigma*eye(size(M'*M)))*M';
54 % ——— PROCESSO
55 for k=4:nit
56     auxy=0;
57     for l=2:na
58         auxy=auxy-A(l)*y(k-(l-1));
59     end
60     for l=1:nb
61         auxy=auxy+B(l)*u(k-l);
62     end
63     y(k)=auxy;
64 %     al = -.02*ones(1,length(y)) + .04*rand(1,length(y)); % perturbação aleatória entre -2 2.

```

```

65 %      y = y+al; %perturbação entra apenas na leitura do preditor o
que vai causar uma falha no calculo da energia do controle
66
67 % ----- CALCULO DO PREDITOR N2 PASSOS A FRENTE
68 alfa1=-alfa(2:size(alfa,2));
69 alfa2=zeros(1,na);
70 for i=1:na
71     auxyp=0;
72     for l=1:na
73         auxyp=auxyp+alfa1(l)*y(k-l+i)+alfa2(l)*yp(k-l+i)+auxb(l+i)*du(k-i);
74     end
75     yp(k+i)=auxyp;
76     alfa2(i)=alfa1(i); alfa1(i)=0;
77 end
78 for i=na+1:N2
79     auxyp=0;
80     for l=1:na
81         auxyp=auxyp+alfa2(l)*yp(k-l+i);
82     end
83     yp(k+i)=auxyp;
84 end
85 %      y=y-al; %retira a perturbação da saída real
86 % ----- CALCULO DO PRIMERO CONTROLE INCREMENTAL
87 for i=1:N2
88     ye(k+i)=yr(k+i)-yp(k+i);
89 end
90 i=1;
91 w(i)=0;
92 for j=1:N2
93     w(i)=w(i)+V(i,j)*ye(k+j);
94 end
95 du(k)=w(1);
96 u(k)=u(k-1)+du(k);
97 %saturador
98 if u(k)>1
99     u(k)=1;
100 end
101     if u(k)<0
102         u(k)=0;
103     end
104 end
105 % ----- RESULTADOS DE SIMULACAO
106 t=1:nit;
107 subplot(221),plot(t,y(t)+temp_ini,t,yr(t)+temp_ini) %subplot para análise geral e auxiliar para definir os valores iniciais.
108 title('Temperatura interna (saída)')
109 xlabel('Tempo (min)')
110 ylabel('Temperatura de saída (°C)')
111 subplot(222),plot(t,yr(t)+temp_ini,t,yp(t)+temp_ini),title('yr');
112 subplot(223),plot(t,u(t),t,ones(1,length(t),'r—')),title('controle')
113 subplot(224),plot(t,ye(t)),title('erro');
114 figure
115 plot(t,y(t)+temp_ini,t,yr(t)+temp_ini)
116 title('Temperatura interna (saída)')
117 xlabel('Tempo (min)')
118 ylabel('Temperatura interna (°C)')
119 legend('Temperatura','Referência')
120 saveas(gcf,'respgpc','png')
121 ym=y;
122 figure
123 plot(t,u(t))
124 title('Energia de controle')
125 xlabel('Tempo (min)')
126 ylabel('Energia (%PWM)')
127 saveas(gcf,'egpc','png')
128 um=u;
129 %% GPC com AG :D
130 if 0 %mudar pra 1 para rodar o algoritmo
131     tic %tic toc mede o tempo que o programa leva pra realizar a operação entre os termos
132     %definição dos parâmetros
133     lb = [4 1 0]; %lower bounds limite inferior

```

```

134   ub = [100 100 5]; %upper bounds
        limite superior Universo
        de discurso
135   nvars=length(lb); %numero de
        variáveis
136   SizePop=500; %numero de
        individuos inicial
137   NumGeracoes = 10; %numero de
        gerações
138   options = gaoptimset( '
        PopulationType', '
        DoubleVector', '
        PopulationSize', ...
139       SizePop, 'Generations',
        NumGeracoes, '
        StallGenLimit',
        NumGeracoes, 'PlotFcns'
        ,...
140       @gaplotbestf, 'EliteCount'
        ,.2*SizePop, '
        CrossoverFraction', .1)
        ,...
141   %'MutationFcn',
        @mutationgaussian);
142   [x fval ef out pop score]= ga(
        @gxg,nvars,[],[],[],[],lb,ub,
       ,[],options);
143   save gpc
144   toc
145   end
146   saveas(gcf,'aggpc','png')
147   load gpc
148   %% Ajustado com AG
149   load gpc
150   sysc = tf(tf3.num,tf3.den);
151   [sysd] = c2d(sysc,60,'zoh'); %
        função transferencia para
        discreto(z)
152   sys = tf(sysd.num,sysd.den,60, '
        Variable', 'z^-1')
153   A = cell2mat(sys.den);
154   B = cell2mat(sys.num);
155   B = B(2:end); % B dividido por z-1
156   Alfa = [1 -1];
157   x(1:2) = round(x(1:2)); %deve ser
        numeros inteiros
158   nit=60;N2=x(1);Nu=x(2);sigma=x(3);
        %AJUSTADO com AG
159   temp_ini = 20;ref=37.6-temp_ini;
        novaref=36.7-temp_ini;
160   % ——— CONDICAOES INICIAIS
161   na=size(A,2);
162   nb=size(B,2);
163   for k=1:nit+N2
164       y(k)=0;u(k)=0;du(k)=0;yp(k)=0;e
        (k)=0;
165   end
166   for j=1:N2
167       alfa(j)=0;
168   end
169   % ——— GERACAO DA REFERENCIA
170   yr(1:round(nit/2))=ref;
171   yr(round(nit/2)+1:nit+N2)=novaref;
172   % ——— CALCULO DE ALFA
173   alfa=conv(A,Alfa);
174   % ——— MONTAGEM DA MATRIZ M
175   auxb=[B zeros(1,N2)];
176   m(1)=auxb(1);
177   for j=2:N2
178       mini=min([j na]);
179       suma=0;
180       for i=2:mini
181           suma=suma-alfa(i)*m(j-i+1);
182       end
183       m(j)=auxb(j)+suma;
184   end
185   for j=1:N2
186       for i=1:Nu
187           if j>i
188               M(j,i)=m(j+1-i);
189           elseif j==i
190               M(j,i)=m(1);
191           else
192               M(j,i)=0;
193           end
194       end
195   end
196   % ——— CALCULO DE V
197   V=inv(M'*M+sigma*eye(size(M'*M)))*M
        ';
198   % ——— PROCESSO
199   for k=4:nit
200       auxy=0;

```



```

277 figure
278 plot(t,ym(t)+temp_ini,t,y(t)+
      temp_ini,t,yr(t)+temp_ini)
279 title('Comparação das temperaturas
      internas de saída do GPC e GPC-
      AG')
280 xlabel('Tempo (min)')
281 ylabel('Temperatura interna (°C)')
282 legend('GPC-manual','GPC-AG','
      Referência')
283 axis([9 40 36.5 37.9])
284 saveas(gcf,'compag','png')
285 figure
286 plot(t,um(t),t,u(t),'r')
287 title('Comparação das energias de
      controle do GPC-manual e GPC-AG')
288 xlabel('Tempo (min)')
289 ylabel('Energia (%PWM)')
290 legend('GPC-manual','GPC-AG')
291 saveas(gcf,'compu','png')
292 figure
293 plot(t(4:end)-4,ym(4:nit)+temp_ini,
      t(4:end)-4,y(4:nit)+temp_ini,
      temp_intfu.time./60,temp_intfu.
      data,temp_intpi.time./60,
      temp_intpi.data,temp_intpp.time
      ./60,temp_intpp.data,t(4:end)-4,
      yr(4:nit)+temp_ini)
294 title('Comparação das temperaturas
      internas de saída do GPC-manual,
      GPC-AG, FUZZY-AG, PI-AG e PP-AG')
295 xlabel('Tempo (min)')
296 ylabel('Temperatura interna (°C)')
297 legend('GPC-manual','GPC-AG','FUZZY-
      AG','PI-AG','PP-AG','Referência')
298 saveas(gcf,'compagt','png')
299 figure
300 plot(t(4:end)-4,ym(4:nit)+temp_ini,
      t(4:end)-4,y(4:nit)+temp_ini,
      temp_intfu.time./60,temp_intfu.
      data,temp_intpi.time./60,
      temp_intpi.data,temp_intpp.time
      ./60,temp_intpp.data,t(4:end)-4,
      yr(4:nit)+temp_ini)
301 title('Comparação das temperaturas
      internas com zoom do GPC-manual,
      GPC-AG, FUZZY-AG, PI-AG e PP-AG')
302 xlabel('Tempo (min)')
303 ylabel('Temperatura interna (°C)')
304 axis([9 40 36.5 37.9])
305 legend('GPC-manual','GPC-AG','FUZZY-
      AG','PI-AG','PP-AG','Referência')
306 saveas(gcf,'compagtz','png')

```

B.4.1 Função custo para o AG

```

1 function h = gxg(x)
2 x = round(x);
3 load('gpc','tf3');
4 sysc = tf(tf3.num,tf3.den);
5 [sysd] = c2d(sysc,60,'zoh'); %
      função transferencia para
      discreto(z)
6 sys = tf(sysd.num,sysd.den,60,'
      Variable','z^-1');
7 A = cell2mat(sys.den);
8 B = cell2mat(sys.num);
9 B = B(2:end); % B dividido por z-1
10 Alfa = [1 -1];
11 x(1:2) = round(x(1:2)); %deve ser
      numeros inteiros
12 nit=60;N2=x(1);Nu=x(2);sigma=x(3);
      %=====> variaveis ajustadas
      pelo AG
13 temp_ini = 20;ref=37.6-temp_ini;
      novaref=36.7-temp_ini;
14 % ----- CONDICAOES INICIAIS
15 na=size(A,2);
16 nb=size(B,2);
17 for k=1:nit+N2
18     y(k)=0;u(k)=0;du(k)=0;yp(k)=0;e
      (k)=0;
19 end
20 for j=1:N2
21     alfa(j)=0;
22 end
23 % ----- GERACAO DA REFERENCIA
24 yr(1:round(nit/2))=ref;
25 yr(round(nit/2)+1:nit+N2)=novaref;
26 % ----- CALCULO DE ALFA

```

```

27  alfa=conv(A,Alfa);
28  % ——— MONTAGEM DA MATRIZ M
29  auxb=[B zeros(1,N2)];
30  m(1)=auxb(1);
31  for j=2:N2
32      mini=min([j na]);
33      suma=0;
34      for i=2:mini
35          suma=suma-alfa(i)*m(j-i+1);
36      end
37      m(j)=auxb(j)+suma;
38  end
39  for j=1:N2
40      for i=1:Nu
41          if j>i
42              M(j,i)=m(j+1-i);
43          elseif j==i
44              M(j,i)=m(1);
45          else
46              M(j,i)=0;
47          end
48      end
49  end
50  % ——— CALCULO DE V
51  V=inv(M'*M+sigma*eye(size(M'*M)))*M';
52  % ——— PROCESSO
53  for k=4:nit
54      auxy=0;
55      for l=2:na
56          auxy=auxy-A(l)*y(k-(l-1));
57      end
58      for l=1:nb
59          auxy=auxy+B(l)*u(k-l);
60      end
61      y(k)=auxy;
62  % ——— CALCULO DO PREDITOR
63      N2 PASSOS A FRENTE
64      alfa1=-alfa(2:size(alfa,2));
65      alfa2=zeros(1,na);
66      for i=1:na
67          auxyp=0;
68          for l=1:na
69              end
70              yp(k+i)=auxyp;
71              alfa2(i)=alfa1(i); alfa1(i)
72                  =0;
73              end
74              for i=na+1:N2
75                  auxyp=0;
76                  for l=1:na
77                      auxyp=auxyp+alfa2(l)*yp
78                          (k-l+i);
79                  end
80                  yp(k+i)=auxyp;
81              end
82              % ——— CALCULO DO PRIMEIRO
83              CONTROLE INCREMENTAL
84              for i=1:N2
85                  ye(k+i)=yr(k+i)-yp(k+i);
86              end
87              i=1;
88              w(i)=0;
89              for j=1:N2
90                  w(i)=w(i)+V(i,j)*ye(k+j);
91              end
92              du(k)=w(1);
93              u(k)=u(k-1)+du(k);
94              %saturador
95              if u(k)>1
96                  u(k)=1;
97              end
98              if u(k)<0
99                  u(k)=0;
100             end
101         end
102         t=4:nit;
103         h=trapz(t,((yr(t)-y(t)).^2).*t); %
104             integral com quadrado para
105             eliminar a parte negativa e dar
106             peso maior para maiores erros
107             com o passar do tempo
108     end
109
110     B.5 CONTROLADOR PID
111     % Gustavo Flore Cavenago Ra:1578375
112     % Atualização 22/09/2016
113     close all;clear;clc;

```

```

4 %% Determinação dos Parâmetros kp,          ,...
   ki e kd – Ziegler–Nichols                29     num2str(x(2)), ' kd= ',
5 %página 697 ogata                          num2str(x(3)), ' ka= ',
6 load('pi', 'tf4', 'tf3')                  num2str(x(4))]);
7 T = tf4.Tp1;                               30     toc
8 L = tf4.Td;                                31     save pid
9 kpzn = 1.2*T/L;                            32     saveas(gcf, 'figuras/pid/agr.png
10 kizn = kpzn/(2*L);                        ')
11 kdzn = kpzn/(.5*L);                       33 end
12 kazn = kizn/10;                           34 %% Plotagem e verificação das
13 %% Determinação dos Parâmetros kp,         especificações PI convencional
   ki, kd e ka – Algoritmo genético        35 kp=kpzn;
14 if 0 %mudar pra 1 para rodar o            36 ki=kizn;
   algoritmo                                37 kd=kdzn;
15     tic                                    38 ka=kazn;
16     %definição dos parâmetros             39 st = 60; %sample time (do controle
17     lb = [kpzn*.8 kizn*.4 0 0]; %         contínuo)
   limite inferior                          40 temp_ini=20;
18     ub = [kpzn*1.2 kizn*1.2 kdzn*2        41 temp_ext=temp_ini;
   kizn/2]; %limite superior                42 temp_ref = 37.6;%temperatura ideal
19     nvars=length(ub); %numero de         até o 18° dia, depois 36.8
   variáveis                                43 tempo_sim = 1*60*60; % 20 minutos
20     SizePop=100; %numero de              44 load('pi', 'tf3');
   individuos inicial                       45 run simulacao_pid.slx
21     NumGeracoes = 20; %numero de        46 options = simset('SrcWorkspace', '
   gerações                                  Current');
22     options = gaoptimset('              47 sim('simulacao_pid.slx', tempo_sim,
   PopulationType', '                        options);
   DoubleVector', '                          48 temp_intzn = temp_int;
   PopulationSize', ...                      49 temp_energiazn = temp_energia;
23     SizePop, 'Generations',              50 temp_energia1zn = temp_energia1;
   NumGeracoes, '                            51 figure
   StallGenLimit',                           52 plot(temp_int.time./60, temp_int.
   NumGeracoes, 'PlotFcns'                  data, temp_int.time./60, 37.6*ones
   ,...                                       (1, length(temp_int.time)), 'r')
24     @gaplotbestf, 'EliteCount'           53 title('Temperatura de saída (
   ,.1*SizePop, '                            interna da incubadora) – PID+P–
   CrossoverFraction', .3)%                 ZN');
   ,...                                       54 ylabel('Temperatura (°C)')
25     %'MutationFcn',                       55 xlabel('Tempo (min)')
   @mutatingaussian);                        56 legend('Temperatura Interna', '
26                                           Referência')
27 [x fval ef out pop score]= ga(           57 axis([0 60 20 45])
   @gxpzn, nvars, [], [], [], [], lb        58 saveas(gcf, 'figuras/pid/tpizn.png')
   ub, [], options);                          59 figure
28 disp(['Os melhores valores são:         60 plot(temp_energia1.time./60,
   kp= ', num2str(x(1)), ' ki= '           temp_energia1.data, 'r',

```

```

temp_energia.time./60,
temp_energia.data, '.', p.time
./60, p.data, p2.time./60, p2.data
'm', d.time./60, d.data, i.time
./60, i.data)
61 title('Energias de controle – PID+P
–ZN');
62 ylabel('PWM (0 à 1)')
63 xlabel('Tempo (min)')
64 legend('Pré saturação', 'Pós
saturação', 'Proporcional', '2°
Proporcional', 'Derivativo', '
Integrativo')
65 axis([0 60 –2 3])
66 saveas(gcf, 'figuras/pid/enpizn.png')
67 %% Plotagem e verificação das
especificações PI Ajustado com
AG
68 load pid %para não precisar rodar o
AG a todo momento
69 for i=1:length(x) %condição para
eliminar numeros abaixo de zero
gerados pela mutação
70 if x(i) < 0
71 x(i)=0;
72 end
73 end
74 kp=x(1);
75 ki=x(2);
76 kd=x(3);
77 ka=x(4);
78 st = 60; %sample time (do controle
contínuo)
79 temp_ini=20;
80 temp_ext=temp_ini;
81 temp_ref = 37.6;%temperatura ideal
até o 18° dia, depois 36.8
82 tempo_sim = 1*60*60; % 20 minutos
83 load('pi', 'tf3');
84 run simulacao_pid.slx
85 options = simset('SrcWorkspace', '
Current');
86 sim('simulacao_pid.slx', tempo_sim,
options);
87 temp_intag = temp_int;
88 temp_energiaag = temp_energia;
89 temp_energia1ag = temp_energia1;
90 figure
91 plot(temp_int.time./60, temp_int.
data, temp_int.time./60, 37.6*ones
(1, length(temp_int.time)), 'r')
92 title('Temperatura de saída (
interna da incubadora) – PID+P–
AG');
93 ylabel('Temperatura (°C)')
94 xlabel('Tempo (min)')
95 legend('Temperatura Interna', '
Referência')
96 axis([0 60 20 45])
97 saveas(gcf, 'figuras/pid/tpiag.png')
98 figure
99 plot(temp_energia1.time./60,
temp_energia1.data, 'r',
temp_energia.time./60,
temp_energia.data, '.', p.time
./60, p.data, p2.time./60, p2.data,
'm', d.time./60, d.data, i.time
./60, i.data)
100 title('Energias de controle – PID+P
–AG');
101 ylabel('PWM (0 à 1)')
102 xlabel('Tempo (min)')
103 legend('Pré saturação', 'Pós
saturação', 'Proporcional', '2°
Proporcional', 'Derivativo', '
Integrativo')
104 axis([0 60 –2 3])
105 saveas(gcf, 'figuras/pid/enpiag.png')
106 figure
107 plot(temp_intzn.time./60, temp_intzn
.data, temp_intag.time./60,
temp_intag.data, temp_int.time
./60, 37.6*ones(1, length(temp_int
.time)), 'r')
108 title('Comparação das Temperaturas
internas ZN e AG');
109 ylabel('Temperatura (°C)')
110 xlabel('Tempo (min)')
111 legend('ZN', 'AG')
112 axis([0 60 20 45])
113 saveas(gcf, 'figuras/pid/comp.png')

```

B.5.1 Função custo para o AG

```

1 function y = gxpil(x)
2 for i=1:length(x) %condição para
    eliminar numeros abaixo de zero
    gerados pela mutação
3     if x(i) < 0
4         x(i)=0;
5     end
6 end
7 kp=x(1);
8 ki=x(2);
9 kd=x(3);
10 ka=x(4);
11 st = 60; %sample time (do controle
    contínuo)
12 temp_ini=20;
13 temp_ext=temp_ini;
14 temp_ref = 37.6;%temperatura ideal
    até o 18° dia, depois 36.8
15 tempo_sim = 1*60*60; % 20 minutos
16 load('pi','tf3');
17 run simulacao_pid.slx
18 options = simset('SrcWorkspace','
    Current');
19 sim('simulacao_pid.slx',tempo_sim,
    options);
20 y1=trapz(temp_int.time,((temp_ref-
    temp_int.data).^2).*temp_int.
    time.^2); %integral com quadrado
    para eliminar a parte negativa
    e dar peso maior para maiores
    erros
21 load('pi','tf2')
22 tf3=tf2; % para ter uma variação no
    modelo
23 load('pi','tf3');
24 run simulacao_pid.slx
25 options = simset('SrcWorkspace','
    Current');
26 sim('simulacao_pid.slx',tempo_sim,
    options);
27 y2=trapz(temp_int.time,((temp_ref-
    temp_int.data).^2).*temp_int.
    time.^2); %integral com quadrado
    para eliminar a parte negativa
    e dar peso maior para maiores

```

```

    erros
28 relacao = .7; % 70% de peso para os
    ganhos que satisfazem a planta
    modelada com tf3. O resto para a
    planta modelada em tf2 (
    temperaturas mais altas)
29 y=relacao*y1 + (1-relacao)*y2; %
    ponderação final ("multi
    objetivo")
30 if abs(i.data(end))>1e10 %
    eliminando os controles que dão
    problema no integrador.
31     y=y*3;
32 end
33
34 end

```

B.6 EMBARCAÇÃO ARDUINO

```

1 //Gustavo Flore Cavenago
2 //TCC – atualização 14/10/2016
3 //Controle discreto usando
    controlador PP
4 //BIBLIOTECAS
5 #include <dht.h> //DOS LEITORES DE
    HUMIDADE E TEMPERATURA
    SIMULTÂNEOS (VER CÓDIGO: 2 OU
    MAIS SENSORES DE HUMIDADE E
    TEMPERATURA JUNTOS)
6 #include <FlexiTimer2.h> //tem que
    ser este pois outros usam o
    timer1, e timer2 é usado só na
    função tone();
7 #include <EEPROM.h>
8 dht DHT;
9 //DEFINES UTILIZADOS
10 #define DHT11_PIN 4 // PARA SENSOR
    EXTERNO (MAIS SIMPLES)
11 #define DHT21_PIN 5 //PARA SENSOR
    INTERNO AM2301 DE PRECISÃO
12 //DEFINIÇÃO DOS PINOS UTILIZADOS
13 const byte led_nasceu = 2; //LED que
    indica se os ovos já estão
    proximo do tempo de nascer (
    ficar atento)
14 const byte interrupt_button = 3; //
    botão da tampa da chocadeira,

```

```

    detecção de abertura
15 const byte res = 6; //
    resistência acionada pelo relê ,
    que funciona invertido high
    desliga e low liga
16 const byte gira_ovo_tras = 7; //
    aciona a ponte H para sentido
    que volta os ovos
17 const byte gira_ovo_frente = 8; //
    aciona ponte H para empurrar
18 const byte botao_zerar = 9; //botão
    zerar na frente da chocadeira
    para começar novamente a
    contagem de tempo
19 const byte ventilador_ext = 10; //
    controla o cooler externo para
    troca de ar. usar PWM para poder
    controlar a velocidade do ar.
20 const byte ventilador_hum = 11; //
    controla os coolers internos de
    humidade para giro
21 const byte led_res = 12; //led que
    indica período em que a
    resistência está sendo acionada
22 const byte led_tampa = 13; //led
    tampa
23 //VARIÁVEIS UTILIZADAS
24 float temp_ref = 37.6; //
    temperatura de referência que
    muda de acordo com os dias do
    ovo
25 byte umid_ref = 60; //
    umidade de referencia que tambem
    muda ao longo dos dias, mas
    possui valores inteiros
    aproximados.
26 byte periodo = 60; //em segundos (
    discretização)
27 int temperatura_ext = 0;
28 float temperatura_int = 0;
29 int umidade_ext = 0;
30 float umidade_int = 0;
31 float l = 0;
32 float u = 0;
33 float us = .2; //não começar em
    zero.
34 byte pwm = 0;
35 byte seg = 0; //variáveis para
    criação do relógio interno
36 byte seg_ant = 0;
37 byte minuto = 0;
38 byte minuto_ant = 0;
39 byte hora = 0;
40 byte hora_ant = 0;
41 byte dia = 0; // numero do dia em
    que os ovos foram colocados,
    para ajustar os parâmetros.
42 byte minuto_ender = 4; //
    endereço a ser salvo o valor do
    min, segundos não precisa pois
    se perdido não fara muita
    diferença
43 byte hora_ender = 5;
44 byte hora_ant_ender = 6;
45 byte dia_ender = 7;
46 unsigned long segab = 0;
47 int tempo_giro = 2000; //0,5
    segundo
48 byte giro = 0; //controle do
    giro
49 byte giro_ender = 4; // endereço
    onde deve ser salvo na EEPROM
    pra evitar erros e comprometer
    os ovos
50 byte a = 0; //contador de giros
51 //SETUP DE INICIALIZAÇÃO//
52 void setup(){
53 //SETUP DOS PINOS UTILIZADOS
54 pinMode (led_tampa, OUTPUT); //
    led_tampa como saída 0 – 5 v
55 pinMode (interrupt_button, INPUT)
    ;// a interrupção é um pino de
    leitura !! tambem para
    verificação.
56 pinMode (res, OUTPUT); //seta a
    resistência como saída.
57 pinMode (led_res, OUTPUT);
58 pinMode (ventilador_hum, OUTPUT);
59 pinMode (ventilador_ext, OUTPUT);
60 pinMode (gira_ovo_frente, OUTPUT)
    ;
61 pinMode (gira_ovo_tras, OUTPUT);
62 pinMode (botao_zerar, INPUT);
63 pinMode (led_nasceu, OUTPUT);

```

```

64 //SETUP DA SERIAL
65 Serial.begin(9600); // INICIALIZA A
    PORTA SERIAL DE COMUNICAÇÃO
66 Serial.println("Chocadeira
    Conectada!!");
67 Serial.println("DIA\tHORA\tMIN\
    tSEG\tSEG_ABS\tTEMP_RF\tTEMP_I
    \tTEMP_E\tHUM_REF\tHUM_I\
    tHUM_E\tPWM\tU\tUS\tPERIODO\
    tCONTAGIRO"); // SEG_ABS É O
    TEMPO ABSOLUTO PARA SER USADO
    COMO BASE DE TEMPO NA PLOTAGEM
    DOS DADOS
68 //SETUP TIMER1
69 FlexiTimer2::set(1000, INCREMENTO
    ); // 1000ms period
70 FlexiTimer2::start();
71 } //END SETUP
72 //LOOP PRINCIPAL INFINITO ///
73 void loop() {
74     digitalWrite(ventilador_hum, HIGH)
    ; //aciona direto o cooler do
    humidificador.
75     if (seg%2==1 & seg!=seg_ant) {
76         LEITURA(); // LE-SE OS VALORES
            INTERNOS E EXTERNOS DE
            TEMPERATURA E UMIDADE.
77         PLOTAGEM(); // PLOTA OS VALORES
            NA SERIAL E SE USAR, UM
            DISPLAY (NÃO TEM POR
            ENQUANTO).
78         seg_ant=seg;
79     }
80     if (minuto!=minuto_ant) {
81         CONTROLE(); // CALCULA OS
            VALORES DE ENERGIA PARA SER
            APLICADO NO SISTEMA
82         minuto_ant=minuto;
83     }
84     AJUSTA_PWM(); // AJUSTAR O VALOR EM
        % DO PWM CRIADO PARA CONTROLE
        DA RESISTÊNCIA
85     TEMPO(); // CHAMA A FUNÇÃO TEMPO
        PARA AJUSTAR OS PARAMETROS DE
        TEMPO DO OVO PARA PODER
        ALTERAR OS PARÂMETROS DE
        CONTROLE
86     REGRAS(); // ONDE FARÃO OS AJUSTES
        E VERIFICAÇÕES NECESSÁRIAS.
87 } //END LOOP
88 void CONTROLE() {
89     u = .1646*(temp_ref-
        temperatura_int) + .022*(
        temperatura_int-
        temperatura_ext); // Controle P
        + P
90     us = u; // Energia de saída
91     if (us>=1) { // saturação da saída
        para não extrapolar os limites
        do PWM
92         us=1;
93     }
94     if (us<=0) {
95         us=0;
96     }
97 } //end CONTROLE
98 void AJUSTA_PWM() {
99     pwm = byte(us*periodo); //
        tempo em segundos proporcional
        ao período em que a
        resistencia ficará acionada
100     if (seg<=pwm) {
101         digitalWrite(res, HIGH);
102         digitalWrite(led_res, HIGH);
103     }
104     else {
105         digitalWrite(res, LOW);
106         digitalWrite(led_res, LOW);
107     }
108 } //END AJUSTA_PWM
109 void LEITURA() {
110     int chk = DHT.read21(DHT21_PIN);
111     umidade_int = DHT.humidity;
112     temperatura_int = DHT.temperature
        ;
113     // READ DATA EXTERNO***
114     chk = DHT.read11(DHT11_PIN);
115     umidade_ext = DHT.humidity;
116     temperatura_ext = DHT.temperature
        ;
117 } //end LEITURA
118 void INCREMENTO() { //o tempo deve
        ser apenas incrementado, pois
        estava dando problema se

```

```

    mandasse dar um print dentro de // 155
    uma interrupção que parasse // 156
    outro print. // 157
119 seg=seg+1;
120 } //END INCREMENTO // 158
121 void TEMPO() { // 159
122     minuto = EEPROM.read(minuto_ender // 160
        );
123     delay(100); // 161
124     hora = EEPROM.read(hora_ender); // 162
125     delay(100); // 163
126     hora_ant = EEPROM.read( // 164
        hora_ant_ender);
127     delay(100); // 165
128     dia = EEPROM.read(dia_ender); // 166
129     delay(100);
130     if (seg>=60) { // 167
131         seg=seg-60; // 168
132         minuto=minuto+1; // 169
133     } // 170
134     if (minuto>=60) { // 171
135         minuto=minuto-60; // 172
136         hora=hora+1;
137     }
138     if (hora>=24) { // 173
139         hora=hora-24; // 174
140         dia=dia+1; // 175
141     } // 176
142     //SALVANDO OS VALORES DE TEMPO – // 177
        MIN, HORA, HORA_ANTERIOR E DIA, // 178
        NA MEMORIA EEPROM // 179
143     EEPROM.write(minuto_ender, minuto) // 180
        ; // 181
144     delay(100);
145     EEPROM.write(hora_ender, hora); // 182
146     delay(100); // 183
147     EEPROM.write(hora_ant_ender, // 184
        hora_ant); // 185
148     delay(100); // 186
149     EEPROM.write(dia_ender, dia); // 187
150     delay(100); // 188
151     segab = seg + (minuto)*60 + (hora // 189
        )*3600 + long(dia)*86400; // 190
152     delay(500); // 191
153 } //end TEMPO // 192
154 void GIRAR() { // 193
        fica dentro da função REGRAS // 194
        giro = EEPROM.read(giro_ender);
        if (giro==0){
            digitalWrite(gira_ovo_frente ,
                HIGH);
            delay(tempo_giro);
            digitalWrite(gira_ovo_frente ,
                LOW);
            giro=1;
            EEPROM.write(giro_ender , giro);
        }
        else{
            digitalWrite(gira_ovo_tras ,
                HIGH);
            delay(tempo_giro);
            digitalWrite(gira_ovo_tras , LOW
                );
            giro=0;
            EEPROM.write(giro_ender , giro);
        }
        a++;
    } //end girar
void PLOTAGEM() { // PARA REGULAR
    MELHOR A FUNÇÃO SERIAL PRINT,
    ORGANIZANDO EM APENAS UM LUGAR.
    Serial.print(dia);
    Serial.print("\t");
    Serial.print(hora);
    Serial.print("\t");
    Serial.print(minuto);
    Serial.print("\t");
    Serial.print(seg);
    Serial.print("\t");
    Serial.print(segab); // segundos
        absolutos;
    Serial.print("\t");
    Serial.print(temp_ref);
    Serial.print("\t");
    Serial.print(temperatura_int);
    Serial.print("\t");
    Serial.print(temperatura_ext);
    Serial.print("\t");
    Serial.print(umid_ref);
    Serial.print("\t");
    Serial.print(umidade_int);
    Serial.print("\t");
    Serial.print(umidade_ext);
    Serial.print("\t");

```

```

195 Serial.print(us*100);
196 Serial.print("\t");
197 Serial.print(u);
198 Serial.print("\t");
199 Serial.print(us);
200 Serial.print("\t");
201 Serial.print(periodo);
202 Serial.print("\t");
203 Serial.println(a); //ultimo faz
    pular de linha
204 } //END PLOTAGENS
205 //INTERRUPÇÃO
206 void INTERRUPTAO() {
207     digitalWrite(led_tampa, HIGH); //
        Liga o LED (HIGH = nível
        lógico alto) mostrar que a
        tampa foi aberta
208     delay(1000); // evitar bouncing
209     while (digitalRead(3) == HIGH) {
210     }
211     digitalWrite(led_tampa, LOW);
212 } //end interrupção
213 void REGRAS() {
214     // GIRAR OVOS
215     if (dia > 3 && dia <= 18) { // irá girar
        de hora em hora e apenas entre
        os dias 3 e 18;
216         if (hora_ant != hora) {
217             GIRAR();
218             hora_ant = hora;
219         }
220     }
221     // AJUSTAR A REFERENCIA DA TEMP.
        E HUMIDADE
222     if (dia <= 18) { //até o dia 18 a
        temperatura de interna deve
        ser de 37,6°C
223         temp_ref = 37.6;
224     }
225     else { //os outros dias
        devem estar em 36,8°C
226         temp_ref = 36.8;
227     }
228     // BOTÃO RESET DO TEMPO
229     if (digitalRead(botao_zerar) ==
        HIGH) { //função para apagar
        memória EEPROM do tempo quando
        apertar o botão por 2
        segundos
230         delay(2000);
231         if (digitalRead(botao_zerar) ==
        HIGH) {
232             EEPROM.write(minuto_ender, 0);
233             EEPROM.write(hora_ender, 0);
234             EEPROM.write(hora_ant_ender
                , 0);
235             EEPROM.write(dia_ender, 0);
236             minuto_ant = 0;
237             a = 0;
238         }
239     } else {
240         GIRAR(); // para testar a
        função girar
241     }
242 }
243 // TAMPÃO INTERRUPTÃO
244 if (digitalRead(interrupt_button)
    == HIGH) {
245     INTERRUPTAO();
246 }
247 } //END REGRAS

```

ANEXO A – CONTROLADOR PREDITIVO GENERALIZADO

O controlador preditivo generalizado é uma extensão do controlador de variância mínima, cuja lei de controle, resultado da minimização do erro de seguimento de previsão sobre um horizonte de tempo finito, apresenta desempenho mais robusto que os controladores discretos convencionais e trata processos com atrasos variantes ou dinâmicas não-modeladas, de fase não-mínima e sistemas instáveis em malha aberta.

Para o projeto do controle preditivo generalizado admite-se o processo representado por modelo discreto da forma

$$A(z^{-1})\Delta y(k) = z^{-1}B(z^{-1})\Delta u(k) \quad (16)$$

onde $u(k)$ é sinal de controle e $y(k)$ é a variável mensurável. O operador Δ garante ação integral no controlador de modo a cancelar o efeito da perturbação e garantir erro em regime nulo.

Os polinômios $A(z^{-1})$ e $B(z^{-1})$ são assumidos conhecidos *a priori* e estão representados na forma

$$\begin{aligned} A(z^{-1}) &= 1 + a_1z^{-1} + a_2z^{-2} + \dots + a_nz^{-n} \\ B(z^{-1}) &= b_0 + b_1z^{-1} + b_2z^{-2} + \dots + b_mz^{-m} \end{aligned}$$

A lei de controle minimiza a seguinte função custo:

$$J(u, k) = \left\{ \sum_{j=N_1}^{N_2} [y(k+j) - y_r(k+j)]^2 + \beta \sum_{j=1}^{N_u} [\Delta u(k+j-1)]^2 \right\}, \quad (17)$$

sujeita a

$$\Delta u(k+j) = 0 \quad \text{para } j = N_u, \dots, N_2$$

onde N_1 é o horizonte mínimo de previsão, N_2 é o horizonte máximo de previsão, N_u é o horizonte de controle e β é a seqüência de ponderação da ação de controle. Por questão de simplicidade o termo z^{-1} é omitido no desenvolvimento do GPC.

Para resolver o problema da minimização da equação (17), deve-se fazer a previsão da saída j passos a frente, $y(k+j)$ para $j = N_1, \dots, N_2$, baseada nas informações conhecidas no instante k e nos valores futuros do controle incremental. Esta previsão envolve o uso da seguinte equação:

$$1 = E_j A \Delta + q^{-j} F_j \quad (18)$$

onde os polinômios E_j e F_j são definidos a partir do conhecimento de A e do horizonte de previsão j .

Utilizando-se as equações (16) e (18) obtém-se

$$y(k+j) = F_j y(k) + E_j B \Delta u(k+j-1) + E_j \xi(k+j) \quad (19)$$

O último termo da equação (19) contém informação que é independente dos sinais mensuráveis em k . Logo, a previsão de $y(k+j)$, que emprega medidas conhecidas em k é

$$\hat{y}(k+j) = F_j y(k) + E_j B \Delta u(k+j-1) \quad (20)$$

Na equação (20) o sinal $\hat{y}(k+j)$ é função dos valores dos sinais conhecidos em k e também das entradas de controle futuras que devem ser calculadas. Então, utiliza-se uma segunda identidade polinomial para separar os valores passados e futuros do controle, isto é,

$$E_j(z^{-1})B(z^{-1}) = G_j(z^{-1}) + z^{-j}\Gamma_j(z^{-1}) \quad (21)$$

que produz a seguinte expressão de previsão

$$\hat{y}(k+j) = F_j y(k) + G_j \Delta u(k+j-1) + \Gamma_j \Delta u(k-1) \quad (22)$$

ou, equivalentemente,

$$\hat{y}(k+j) = G_j \Delta u(k+j-1) + \hat{y}(k+j/k) \quad (23)$$

onde

$$\hat{y}(k+j/k) = \Gamma_j \Delta u(k-1) + F_j y(k) \quad (24)$$

é a previsão da resposta livre de $y(k+j)$, assumindo que os controles incrementais futuros depois de $(k-1)$ são nulos.

Manipulando-se as equações (18) e (21) obtém-se

$$B(A\Delta)^{-1} = G_j + q^{-j}\Gamma_j + q^{-j}BF_j(A\Delta)^{-1} \quad (25)$$

Seja o vetor \mathbf{f} composto pelas previsões da resposta livre, isto é,

$$\mathbf{f} = [\hat{y}(k+1/k), \hat{y}(k+2/k), \dots, \hat{y}(k+N_2/k)]^T \quad (26)$$

onde obtém-se as previsões de $\hat{y}(k+l)$, para $l = 1, \dots, N_2$, assumindo que $u(k+l) = 0$ para $l = 1, \dots, N_2 - 1$.

Definindo o vetor de controle incremental futuro, \hat{u} por

$$\hat{u} = [\Delta u(k), \Delta u(k+1), \dots, \Delta u(k+N_u-1)]^T \quad (27)$$

onde $\Delta u(k+j) = 0$ para $j \geq N_u$ e o vetor de saídas preditas da planta controlada

$$\hat{y} = [\hat{y}(k+1), \hat{y}(k+2), \dots, \hat{y}(k+N_2)]^T \quad (28)$$

pode-se rescrever a equação (23) de acordo com

$$\hat{y} = G\hat{u} + \mathbf{f} \quad (29)$$

onde a matriz G é composta dos parâmetros da resposta impulsiva, g_i , do modelo da

planta.

$$G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_u-1} & g_{N_u-2} & \cdots & g_0 \\ \vdots & \vdots & \cdots & \vdots \\ g_{N_2-1} & g_{N_2-2} & \cdots & g_{N_2-N_u} \end{bmatrix}$$

A matriz G é triangular inferior, de dimensão $(N_2 \times N_u)$, leva em conta a suposição sobre $\Delta u(k+j) = 0$ para $j \geq N_u$ e considera N_1 igual a 1. O efeito de alterar o valor de N_1 é apagar as linhas superiores da matriz G . Também, sendo o primeiro parâmetro da resposta impulsiva g_0 , então o atraso de transporte da planta é igual a 1. Se o atraso é maior que 1, as primeiras $d-1$ filas de G são nulas, mas se N_1 é assumido ser igual a d os primeiros elementos são não-nulos.

A minimização de J resulta no seguinte vetor de controle incremental:

$$\hat{u} = (G^T G + \lambda_u I)^{-1} G^T (y_r - f) \quad (30)$$

onde y_r é o vetor do sinal de referência definido por

$$y_r = [y_r(k+1), y_r(k+2), \dots, y_r(k+N_2)]^T \quad (31)$$

O primeiro elemento de \hat{u} é $\Delta u(k)$, e o controle atual, $u(k)$ é calculado por

$$u(k) = u(k-1) + g^T (y_r - f) \quad (32)$$

onde g^T é a primeira fila de $(G^T G + \lambda_u I)^{-1} G^T$. A ação de controle apresenta ação integral e proporciona erro em regime nulo para a referência constante, $y_r(k+i) = y_r$.

A equação (30) produz um controle incremental futuro de k até $(k+N_u-1)$ como uma estratégia em malha aberta baseada na informação disponível no instante k . O mecanismo utilizado para fechar a malha é forçar um controle realimentado no GPC para implementar somente o primeiro elemento de \hat{u} , isto é, $\Delta u(k)$, e então recalculer a solução do problema de controle ótimo para o próximo passo utilizando as medidas disponíveis em $(k+1)$ na especificação de f . Este procedimento é denominado como *Receding Horizon Control*.