ADO – RESUMO DO CONTROLE PID DE POTÊNCIA EM CORRENTE ALTERNADA – ARDUINO E TRIAC

ANDRÉ RICHARD SANCHES

FERNANDO MARTINS FERREIRA

HIGOR CABRAL CAVALCANTE

RAFAEL PEQUINO FREIRE

31 DE MARÇO DE 2025

Sumário

Introdução:

Este resumo irá explorar a implementação de um sistema de controle Proporcional-Integral-Diferencial (PID) para regularmos uma potência em corrente alternada, utilizando um Arduino e um TRIAC (Componente eletrônico que controla correntes alternadas). O objetivo principal é controlar a potência dissipada em uma carga resistiva, como uma resistência de aquecimento, ajustando dinamicamente o ângulo de fase da onda senoidal da rede elétrica. Esse tipo de controle é amplamente utilizado em aplicações industriais e domésticas que exigem ajustes precisos de potência.

1. Introdução ao Controle de Potência em Corrente Alternada

O controle de potência em corrente alternada pode ser realizado de diferentes formas. Neste projeto, comparamos duas abordagens principais:

1. **Controle por chaveamento on/off (relé)**: Método simples, no qual a carga é ligada ou desligada completamente. É utilizado em aplicações como controle de temperatura de geladeiras e aquecedores residenciais, mas pode apresentar mudanças indesejadas na temperatura.
2. **Controle por ajuste do ângulo de fase com PID**: Técnica mais avançada que permite um ajuste contínuo da potência entregue a carga, reduzindo oscilações e melhorando a eficiência energética do aparelho. Se tornando semelhante ao funcionamento de um chuveiro elétrico com regulagem eletrônica de temperatura.

O controle PID desempenha um papel fundamental ao minimizar a diferença entre o valor desejado (set point) e o valor medido, ajustando a potência de forma contínua para manter a estabilidade térmica do sistema.

2. Sensor de Passagem por Zero: Sincronização com a Rede Elétrica

Para garantir a precisão no controle da potência, é essencial sincronizar o acionamento do TRIAC com a rede elétrica (110V ou 220V). Isso é feito através da detecção da passagem por zero da onda chamada senoidal, que ocorre duas vezes por ciclo (50 Hz ou 60 Hz, dependendo da rede elétrica utilizada).

O circuito de detecção de passagem por zero consiste em:

- **Optoacoplador (4N25 ou similar)**: Componente que gera um pulso para o Arduino sempre que a tensão da rede cruza o ponto zero.
- **Interrupções no Arduino**: O arduino processa esses pulsos mencionados para determinar o momento ideal de disparo do TRIAC.

Esse processo é fundamental para evitar disparos aleatórios e garantir um controle preciso da potência.

3. Controle de Potência com TRIAC

O controle da potência entregue à carga resistiva é realizado pelo ajuste do ângulo de fase da onda senoidal, utilizando um TRIAC acionado por um optoacoplador MOC3020. O funcionamento ocorre da seguinte maneira:

1. O Arduino recebe o sinal de passagem por zero.
2. Um tempo de atraso é calculado com base no valor de potência desejado.
3. Após esse atraso, o TRIAC é acionado, permitindo a condução da corrente pelo restante do ciclo da onda senoidal.

Os principais componentes do circuito são:

- **MOC3020**: Optoacoplador para isolar a lógica do Arduino da rede elétrica.
- **BTA12-600**: TRIAC que comuta a carga resistiva.
- **H11AA1**: Optoacoplador usado para detectar a passagem por zero.
- **Resistores e capacitores**: Para garantir o funcionamento correto do circuito de acionamento.

4. Código para Controle da Potência

O código implementado no Arduino gerencia o acionamento do TRIAC com base no tempo de atraso necessário para ajustar a potência. O seguinte trecho de código demonstra esse controle:

```
#define loadR 4

volatile int power = 100;

void zero_crosss_int()

{
```

```
  int powertime = (32*(256-power));
  delayMicroseconds(powertime);
  digitalWrite(loadR, HIGH);
  delayMicroseconds(8.33);
  digitalWrite(loadR, LOW);
}

void setup()
{
  Serial.begin(9600);
  pinMode(loadR, OUTPUT);
  attachInterrupt(0, zero_crosss_int, RISING);
}

void loop() {
  power=10;
  delay(10000);
  power=60;
  delay(10000);
  power=120;
  delay(10000);
  power=180;
  delay(10000);
  power=240;
  delay(10000);
}
```

Neste código:

- A função zero_crosss_int() calcula o tempo de atraso antes de acionar o TRIAC, baseado na potência desejada.
- O Arduino utiliza interrupções para capturar a passagem por zero e realizar os ajustes necessários.
- No loop(), diferentes níveis de potência são testados a cada 10 segundos.

Parte 4.1: Introdução ao Controle PID

O Controle Proporcional, Integral e Derivativo (PID) é um mecanismo de controle por realimentação muito utilizado em sistemas industriais. Ele ajusta a variável manipulada (MV) para minimizar a diferença entre o valor atual da variável de processo (PV) e o set point (SP). O PID é composto por três termos:

- **Proporcional (P):** Responde ao erro presente.
- **Integral (I):** Considera a soma dos erros passados.
- **Derivativo (D):** Atua antecipando futuros erros.

Ajustar corretamente os coeficientes KpK_p, KiK_i e KdK_d é essencial para obter um controle eficiente. Dependendo da necessidade, pode-se usar apenas algumas das ações (PI, PD, P ou I).

Exemplo prático: o controle de temperatura da água no chuveiro ilustra o funcionamento de um sistema de controle em malha fechada, onde o cérebro humano atua como controlador PID.

Parte 5: Definição dos Parâmetros do PID

Cada sistema exige ajustes específicos para KpK_p, KiK_i e KdK_d, dependendo do comportamento desejado:

- **K_p:** Define a intensidade da resposta ao erro atual.
- **K_i:** Reduz erro residual acumulado ao longo do tempo.
- **K_d:** Ajuda a minimizar oscilações e estabilizar o sistema.

Os parâmetros podem ser ajustados de três formas:

1. **Tentativa e erro:** Ajustes manuais baseados na observação da resposta do sistema.
2. **Métodos heurísticos:** Como o método de Ziegler-Nichols, que determina KpK_p, KiK_i e KdK_d a partir da resposta oscilatória do sistema.

3. **Combinação das abordagens:** Uso do método heurístico como ponto de partida, refinado manualmente.

Parte 6: Implementação no Arduino

A biblioteca PID_v1 simplifica a implementação do controle PID no Arduino. O código de exemplo configura um sistema de controle de temperatura, utilizando um sensor LM35 e um TRIAC para regular a potência aplicada a uma carga resistiva. O programa:

1. Define os coeficientes KpK_p, KiK_i e KdK_d.
2. Mede a temperatura.
3. Calcula o sinal de controle (MV) usando a biblioteca PID.
4. Ajusta a potência do TRIAC conforme necessário.

Essa abordagem facilita a aplicação do PID em diversos projetos, desde controle de processos até robótica.