**Régis Rolim  
Dayane Carneiro  
Victor Freitas**

**Carolina Barrozo**

**Introdução**

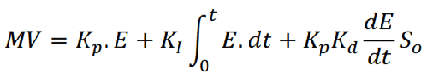
O Controlador PID basicamente significa "Proporcional, Integral,Derivativo". Cada um dos três termos são elementos secundários do controlador total que são somados em conjunto (para calcular um ganho de controle) para corrigir um erro em algumas medidas (ângulo, distância, temperatura etc.) entre a posição atual do sistema e a posição desejada. Existem muitos controladores lineares que podem ser usados ​​para esta situação, mas o PID é o mais usado.

**Teoria**

O controle PID (Proporcional Integral Derivativo) é uma das técnicas mais empregadas na realização do controle de variáveis contínuas. O controle PID consiste em um algoritmo matemático, que tem por função o controle preciso de uma variável em um sistema, que permite ao sistema operar de forma estável no ponto de ajuste desejado, mesmo que ocorram variações ou distúrbios que afetariam sua estabilidade.  Sua aplicação pode ser encontrada em qualquer aplicação que necessite de controle de variáveis contínuas como:

* Pressão;
* Vazão;
* Nível;
* Temperatura;
* Rotação;
* Posicionamento;
* Controle de tensão em fontes chaveadas.

**Equação controle PID**



Kp = 2

Ki = 0.1

Kd = 0

**E = 0**

MV: Variável manipulada.

Kp: Ganho proporcional.

Ki: Ganho integral.

Kd: Ganho derivativo.

E: Erro ou desvio.

S0: Saída inicial do controlador.

O erro é a diferença entre o valor desejado (setpoint) e o valor real da variável. Por exemplo no nosso projeto

**Características do controle PID**

**Proporcional**

O controle proporcional elimina as oscilações da variável, tornando o sistema estável, mas não garante que a mesma esteja no valor desejado (setpoint), esse desvio é denominado off-set. A ação proporcional trabalha corrigindo o erro do sistema, multiplicando o ganho proporcional pelo erro, dessa forma agindo com uma maior amplitude de correção a fim de manter a estabilidade da variável.

**Integral**

O controle integral elimina o desvio de off-set, fazendo com que a variável permaneça próximo ao valor desejado para o sistema mesmo após um distúrbio,  ou seja a variável permanece próximo ao set-point mesmo que ocorra uma variação brusca nas condições de operação. A ação integral realiza a integração do erro no tempo, portanto quanto maior for o tempo de permanência do erro no sistema, maior será a amplitude da ação integral.

**Derivativo**

Quando o controlador sofre variações bruscas, o sinal de saída do controlador PID pode atingir seu limite máximo saindo da região linear normal de controle, fazendo com que o atuador seja acionado até seu limite de capacidade. Em outras palavras ocorre a saturação do sinal de controle. Este fato faz com que o loop de controle seja desfeito, pois o atuador permanecerá no seu limite máximo independentemente da saída do processo ou máquina controlada. Se a ação integral for utilizada, o erro continuará a ser integrado e o termo integral tende a se tornar muito grande. Esse fenômeno é denominado "windup".

Para corrigir o efeito windup o controlador PID deve possuir em seu algoritmo rotinas de "reset" da ação integral, que impede que o termo integral continue a ser atualizado quando a saída atinge seu  limite máximo.

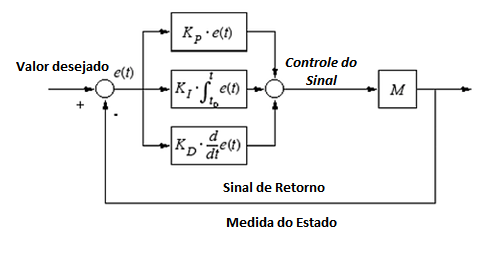
A ação derivativa fornece ao sistema uma ação antecipada, evitando previamente que o desvio se torne maior, quando o processo se caracteriza por ter uma correção lenta comparada com a velocidade do desvio.

A ação derivativa tem sua resposta proporcional à taxa de variação da variável do processo, aumentando a velocidade de resposta do sistema caso a presença do erro seja detectada. Quando o sistema a ser controlado possui maior velocidade de resposta, como por exemplo controle de rotação de motores, a ação derivativa pode ser desativada, pois não há necessidade de antecipar a resposta ao erro, pois o sistema pode corrigir rapidamente seu valor, para desativar a ação derivativa basta tornar seu valor igual a zero.

É comum a utilização das combinações P+I e P+I+D, de modo geral em sistemas com boa velocidade de resposta, como a rotação de motores, podem ser utilizados controladores apenas PI

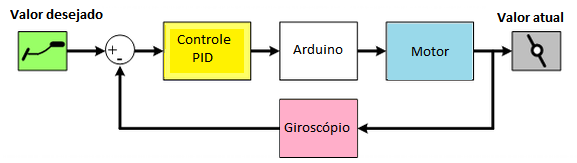
O caso do Arduino uno, pois apesar de possuir conversor A/D este não possui conversor D/A, utilizando portanto sua saída PWM.

**Fluxograma do PID**



Representação matemática do algoritmo de controle

**Fluxograma do sistema**

****

**Valores de calibracao:**

**Os valores de Kp, Kd e Ki precisam ter uma boa escolha pois suas faixas de trabalho que definem o funcionamento do controlador.**

**O controlador proporcional (KP) foi escolhido com base no ganho necessário para a estabilidade rápida do movimento.**

**Nos testes iniciais foram escolhidos valores arbitrarios para testes e foi observado que valores muito altos apresentavam mais instabilidade do que o movimento sem controlador. Com isso uma boa faixa de funcionamento são valores pequenos como algo em torno de 2 a 6.**

**Para o nosso Kp foi escolhido o valor dois por ser o que apresentou a menor variação brusca.**

**O controlador integrativo (KI) possui um valor muito baixo para que o seu peso entre na retificação final do valor. Foi escolhido o valor Kp = 0.1, pois o sistema se trata de valores muito próximos e o erro entre os motores naturalmente devem ser pequeno.**

**O controlador derivativo(KD) foi colocado como nulo pois nos testes ele apresentou aumentar o distúrbio do sistema. Mesmo com valores muito pequenos como 0.1 o sistema ainda ficava igual ou pior. Então foi colocado um valor nulo pois talvez não apresentar tanta diferença com um valor tao pequeno.**

**Programação em PID**

Para manter uma linearidade mais suave no movimento foi necessário a aplicação de um controlador PID.

float set\_point = 0;

float angulo\_atual = 0;

float erro = 0;

float erro\_anterior = 0;

float D\_erro;

float kp = 2; // Ganho proporcional.

float ki = 0.1; // Ganho Integrativo.

float kd = 0; // Ganho Derivativo.

int dt = 0;

float i = 0;

float d = 0;

int tempo\_atual = 0;

int tempo\_anterior = 0;

int velocidade1 = 0;

int velocidade2 = 0;

int dados = 0;

int ang = 0;

int SetPoint = 0;

// Motor 1 (Esquerda);

int M1P1 = 13;

int M1P2 = 12;

// Motor 2 (Direita);

int M2P1 = 8;

int M2P2 = 7;

Sobre a configuração da ponte H e giroscópio:

A Ponte H possui 4 pinos de habilitação para indicar o caminho de fechamento e o sentido da corrente, que consequentemente, afeta o sentido de giro do motor, determinando o sentido de movimento do carro.

Como a movimentação em testes foi feita apenas para frente, os pinos da ponte H foram declarados apenas no primeiro ciclo do arduno.

Na mesma função e definido os parâmetros de comunicação com o giroscópio. (Definido o boud de comunicação serial).

// Codigo de setup:

void setup(){

pinMode(M1P1, OUTPUT);

pinMode(M1P2, OUTPUT);

pinMode(M2P1, OUTPUT);

pinMode(M2P2, OUTPUT);

Serial.begin(115200);

sensor\_setup();

while(first <= 400){

sensor\_loop();

}

}

Logica do controlador:

A aplicação do controlador consiste em aplicar um certo fator de correção perante o erro apresentado na saída. Para medir o erro foi necessário colocar o valor desejado da saída, nomeado de setpoint, e o erro e gerado através da diferença entre o valor do ângulo gerado pelo giroscópio e o setpoint desejado na saída. A variação do erro utilizada na lógica da derivada e feita pela diferença entre o erro anterior gerado pela saída e o erro atual após um ciclo de correção.

Para conseguirmos monitorar o valor de tempo em que o sistema está em funcionamento e utilizada a função Millis da biblioteca do arduino para pegar o tempo em milesegundos.

Todas essas variáveis precisam ser monitoradas a cada ciclo para poder ter uma tomada de decisão diferente no ciclo seguinte, com isso elas ficam implementadas no void loop.

Bloco de código( Void Loop):

void loop(){

sensor\_loop();

ang = Euler[0];

erro\_anterior = erro;

erro = SetPoint - ang; // O Erro È gerado pela diferenÁa do valor que se quer adquirir ( SetPoint) menos o valor da saida, No caso a percepÁ„o de variaÁ„o na saida È feita pela mediacao do angulo.

D\_erro = erro - erro\_anterior; // Variacao do erro para equacao da derivada.. Equivalente ao Dt de uma equacao.

tempo\_anterior = tempo\_atual;

tempo\_atual = millis();

Controle();

}//Fim do loop

Função controle: Nessa função é feito o cálculo do controlador PID, através dos valores adquiridos pelo void loop, para isso e definido também a variação de tempo do sistema.

O cálculo da variável integrativa e derivativa foi feita através da transformação do modelo matemático em linhas de programação.

Após o cálculo das variáveis, elas são implementadas na saída PWM, fazendo a diferença do valor desejado da saída, para mais e para menos. E feito o bloqueio para caso o valor passe do máximo ou do mínimo da saída.

Bloco de controle:

void Controle(){

dt = tempo\_atual - tempo\_anterior;

i = i + ki\*erro\*dt; // Somatório da integral. Ganha peso conforme o tempo passa e o erro ainda se mantem relevante.

d = i + kd\*D\_erro; // Aumenta a eficiência do sistema para que consiga chegar mais rápido ao setpoint. Se torna mais eficiente quando o erro È maior.

// Controle da energicaÁ„o da Ponte H para que o carro possa ir para frente.

digitalWrite(M1P1, HIGH); // 5v

digitalWrite(M1P2, LOW); // 0v

digitalWrite(M2P1, HIGH); // 5v

digitalWrite(M2P2, LOW); // 0v

velocidade1 = 240 + (kp\*erro+i+d); // Enviando o valor desejado na saída + a filtragem de erro feita pelo PID.

velocidade2 = 240 - (kp\*erro+i+d); // Enviando o valor desejado na saída - a filtragem de erro feita pelo PID.

if(velocidade1 > 255) // Teto para que o valor não extrapole o máximo permitido na saída PWM.

velocidade1 = 255;

if(velocidade2 > 255)

velocidade2 = 255;

if(velocidade1 < 0) // Piso para que o valor não extrapole o mínimo permitido na saída PWM.

velocidade1 = 0;

if(velocidade2 < 0)

velocidade2 = 0;

analogWrite(6, velocidade1);

analogWrite(5, velocidade2);

}

**Bibliografia**

**Referência:**KOCHAKARN, Pawit. **Arduino : DC Motor Position Control using PID.**Disponível em: <https://www.techkingdom.org/single-post/2016/12/12/PID-Controller-Basics-DC-Motor-Position-Control-using-Arduino>. Acesso em: 14 dez. 2016.

**Citação com autor incluído no texto:**Kochakarn (2016)

**Citação com autor não incluído no texto:**(KOCHAKARN, 2016)

**Referência:**FREITAS, Carlos Márcio. **Controle PID em sistemas embarcados.**Disponível em: <https://www.embarcados.com.br/controle-pid-em-sistemas-embarcados/>. Acesso em: 17 mar. 2014.

**Citação com autor incluído no texto:**Freitas (2014)

**Citação com autor não incluído no texto:**(FREITAS, 2014)