Universidade Federal de Itajubá - Campus Itabira
Introdução aos sistemas de Controle
Prof. Matheus Henrique Marcolino

Mateus Eusébio de Lacerda

Controle PID de Luminosidade Usando Microcontrolador.

19 de agosto de 2021
Itabira

1 Controle PID

Controladores PID são usados em vários sistemas para controlar diferentes grandezas físicas como temperatura, velocidade, posição, tensão elétrica, e entre outras. Um dos motivos desse tipo de compensador ser tão popular é sua simplicidade de projeto, pois somente ajustando três parâmetros, conseguimos projetar o controlador. Na Figura 1, temos um exemplo de um sistema controlado com um controlador PID.

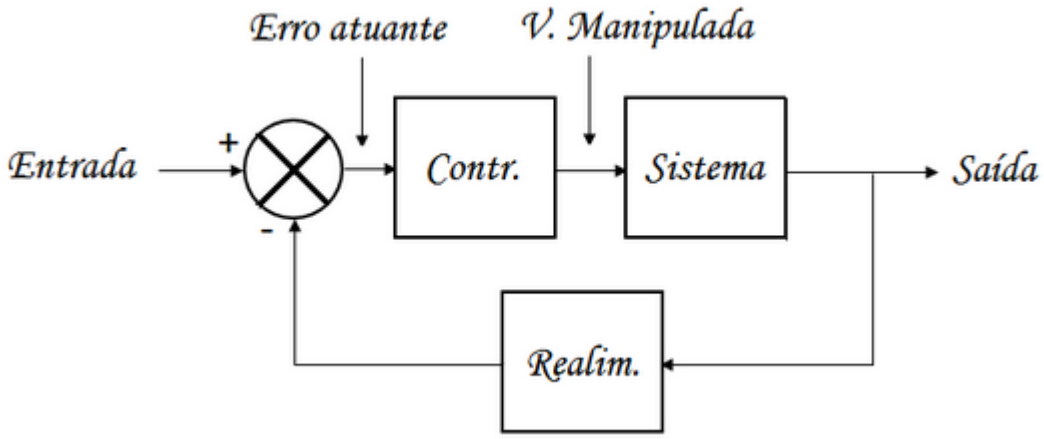

Figura 1 – Malha com Controlador PID

Esses parâmetros são os que dão nome ao controlador: Proporcional(P) , Integral(I) e Derivativo(D). Onde, cada um deles é responsável por ajustar alguma característica do sistema. A escolha correta desses parâmetros (Tunagem do Controlador) é importantíssima para obter a resposta desejada do sistema.

1.1 Ganho Proporcional

Como podemos ver na figura 2, o ganho proporcional(P), que é descrito pela equação 1.1, de modo que Kp é a constante de proporcionalidade e $e(t)$ o sinal do erro.

$$P = Kp.e(t) \tag{1.1}$$

A parte proporcional do Controlador descrita na equação 1.1, retorna um sinal de controle proporcional ao sinal do erro $e(t)$, fazendo com que, quanto maior o erro do sistema maior é o sinal de controle enviado para estabilizar o sistema,

Figura 2 – Ganho Proporcional

1.2 Parte Integral

A parte integrativa pode ser descrita na equação 1.2, ela têm como função diminuir o erro em regime permanente do sistema, como podemos ver na figura 3, e na equação 1.3, valores maiores de Ki(e consequêntemente valores menores de Ti) tendem a reduzir o erro de regime permanente do sistema mais rapidamente.

$$I = \frac{Kp}{Ti}.\int erro.dt = Ki.\int erro.dt \tag{1.2}$$

$$Ki = \frac{Kp}{Ti} \tag{1.3}$$

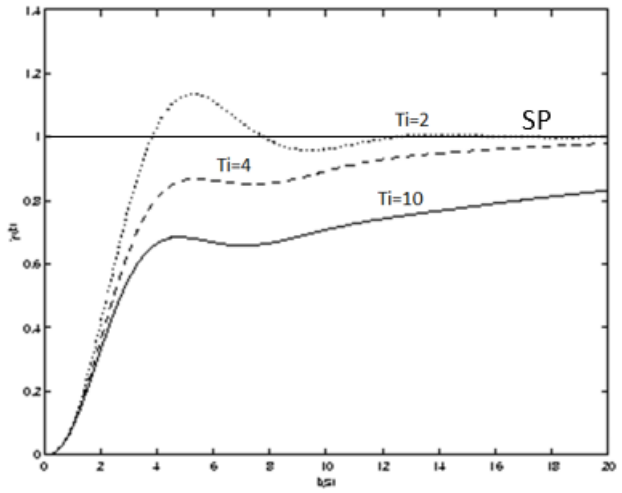

Figura 3 – Resposta da Parte Integral do Controlador

1.3 Parte Derivativa

A parte derivativa é descrita pela equação 1.4, e tem como função diminuir a resposta oscilatória do sistema, de modo que, valores corretos de Kd na equação 1.5, tendem a

diminuir as oscilações na resposta do circuito, como podemos ver na figura 4.

$$D = Kp.Td\frac{d(erro)}{dt} = Kd.\frac{d(erro)}{dt} \tag{1.4}$$

$$Kd = Kp.Td \tag{1.5}$$

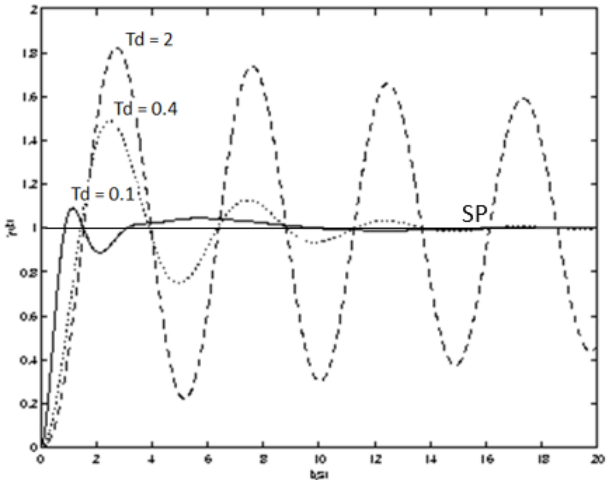

Figura 4 – Resposta da Parte Derivativa do Controlador

1.3.1 A resposta do Compensador

De acordo com os parâmetros escolhidos de Kp, Ki e Kd, teremos uma resposta diferente do sistema. Podemos, inclusive, usar um controlador P,(somente proporcional) PI(proporcional Integrativo) ou PD(proporcional e derivativo), dependendo do sistema que queremos controlar e da resposta que desejamos obter. Na figura 5, temos as respostas de sistemas com cada um desses tipos de controladores.

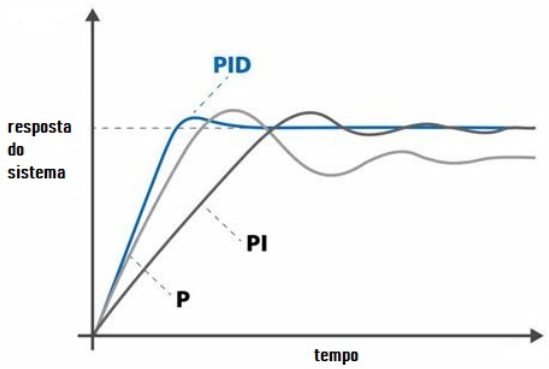

Figura 5 – Resposta dos controlador

Pela figura 5, podemos observar que o controlador P causa uma resposta considerável no sistema, porém, essa resposta está com oscilações e erro em regime permanente.

O Controlador PI corrige o erro em regime permanente, e o PID corrige, além do erro, as oscilações.

1.4 O Sistema de Iluminação

1.4.1 Malha

O objetivo do sistema, cuja malha pode ser vista na figura 6 é garantir uma iluminação $y(t)$ constante e igual ao setpoint $r(t)$ para o ambiente que ele for empregado, levando em consideração as mudanças externas de iluminação $w(t)$, como por exemplo a iluminação que vêm do sol ou outras fontes.

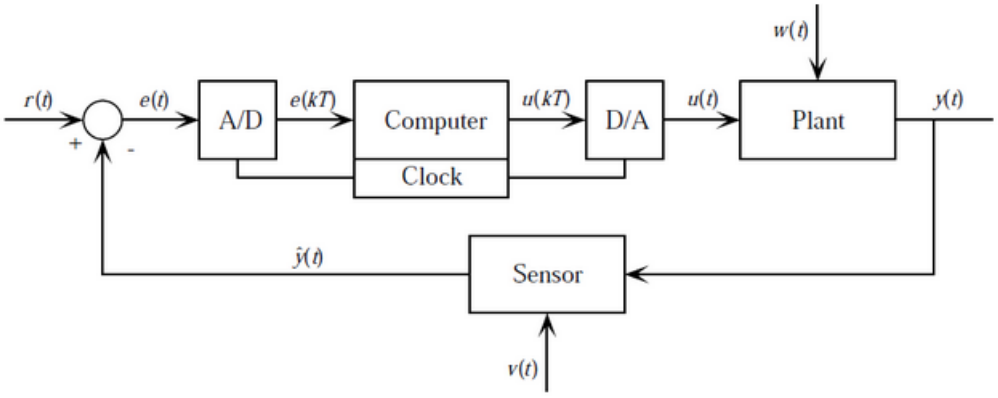

Figura 6 – Malha do Sistema

Onde:

- $r(t)$: Setpoint
- $e(t)$: Erro
- $e(kT)$: Erro Discretizado
- $u(kT)$: Sinal de Controle discretizado
- $u(t)$: Sinal de Controle
- $w(t)$: Fontes de Iluminação Externa
- $y(t)$: Iluminação do Sistema
- $v(t)$: Ruído no sensor
- $y'(t)$: Leitura de Iluminação do Sensor

1.4.2 Hardware

Para fazer uma simulação do circuito, foi projetado um Hardware que terá dois atuadores (dois LEDs) para gerar luminosidade baseado no nosso sinal de controle $u(t)$ e um Sensor LDR (Light Depending Resistor) que retornará para o microcontrolador o valor de luminosidade lido $y'(t)$, que é a luminosidade ambiente $w(t)$ somada à luminosidade gerada pelos atuadores $u(t)$.

Enquanto que o setpoint $r(t)$, o cálculo do erro $e(kT)$ e o cálculo do sinal de controle discretizado $u(kT)$ são feitos pelo software no nosso microcontrolador.

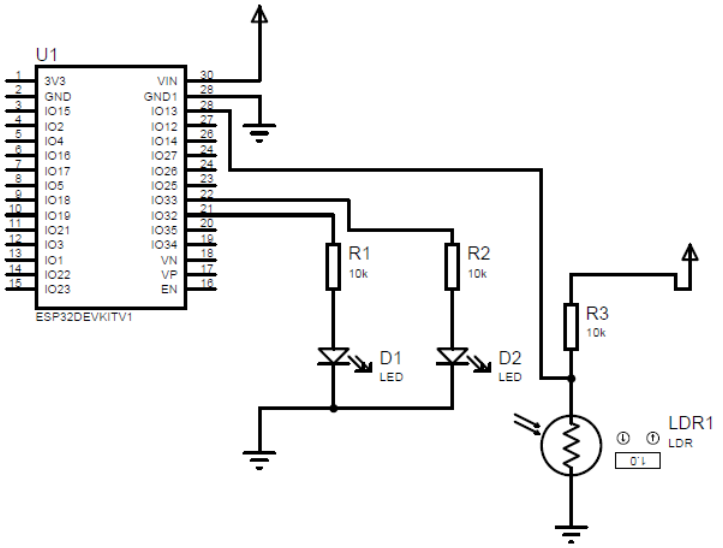

Figura 7 – Hardware

1.4.3 Software

A função mais importante do nosso microcontrolador realiza o cálculo do sinal de controle. Essa função recebe o valor de luminosidade lido *output* e o valor de *setpoint*, realizando faz os cálculos dos blocos presentes na figura 8

```
1  float controle_PID(float output,float setpoint)
2  {
3    static float output_anterior,integral;
4
5    float        erro,proporcional,derivativa;
6
7    erro = setpoint - output;
8
9    proporcional = kp*erro;
10
11   integral = integral + (erro*ki) * (dt);
```

```
12
13    derivativa = ((output_anterior - output)*kd) / (dt);
14
15    output_anterior = output;
16
17    float sinal_de_controle=(proporcional+integral+derivativa);
18
19    return sinal_de_controle;
20  }
```

Podemos comparar o código com a figura 8, na linha 7, ele faz o cálculo do erro. Na linha 9 é calculado o sinal de controle Proporcional, assim como descrito na figura e na equação 1.1. Na linha 11 ele faz o cálculo da perte Integral, assim como na equação 1.2. Na linha 13 é feito o cálculo de parte derivativa, assim como na equação 1.4. Na linha 15 salvamos o sinal de saída para axiliar cálculo da derivada da próxima iteração, enquanto que na linha 17 somamos as três partes do controlador, no sinal de controle, que é retornado na linha 19.

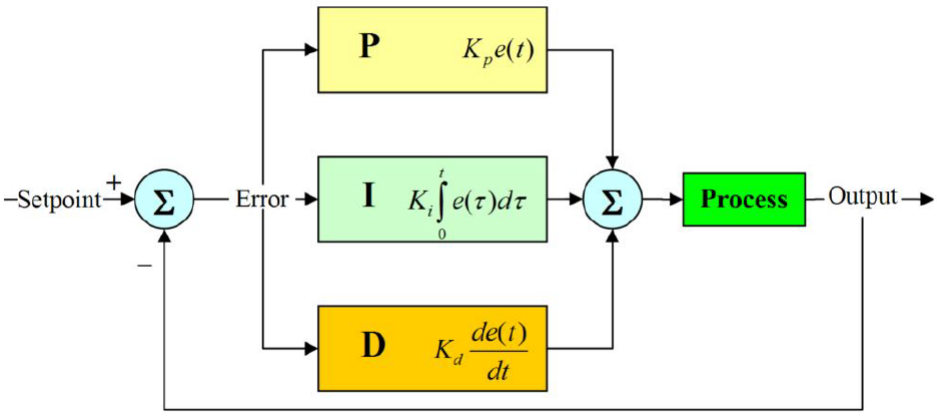

Figura 8 – Diagrama do código

1.5 Tutorial

1.5.1 Materiais necessários

Para utilizar o Projeto, não é necessário utilizar um microcontrolador Arduíno, mas ele têm que executar o Framework do Arduíno. No meu caso foi utilizado um ESP32-DevkitV1.

Para o Atuador são necessários dois LEDs e dois resistores para limitação de corrente de cada LED.

Para o sensor, foi utilizado um Dimmer com módulo sensor de luz LDR, mais informações podem ser encontradas em: <https://www.arduinoecia.com.br/dimmer-modulo-sensor-de-luz>

1.5.2 Programação do Microcontrolador

O ato de colocar um software num microcontrolador pode ser chamado de programação do microcontrolador. Para fazer isso, vamos usar a Interface de Desenvolvimento do Arduíno, que pode ser acessada em: <https://www.arduino.cc/>

Mas primeiro vamos garantir que o Hardware está ligado como na figura 9, de modo que o pino A0 do sensor é ligado no pino 13 do microcontrolador e s pinos 32 e 33 do microcontrolador são ligados nos resistores R1 e R2, de cada LED.

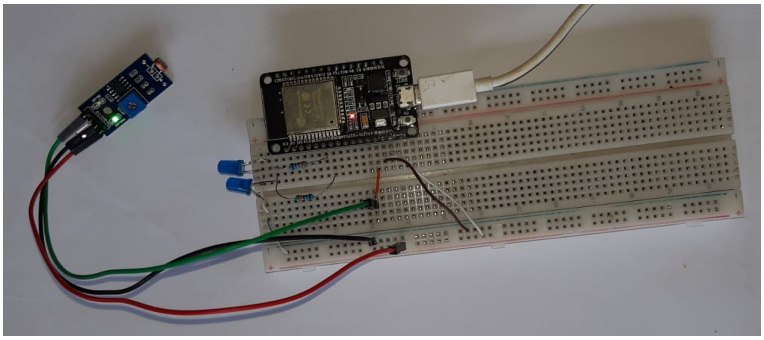

Figura 9 – Ligação do Circuito na Protoboard

Em seguida, vamos abrir a IDE do Arduíno, mostrada na figura 10, e em seguida:

1. Colar o Código Disponibilizado.
2. Conectar o Microcontrolador na Porta USB.
3. Selecionar o nosso microcontrolador em Ferramentas>Placa.
4. Selecionar a Porta disponível em Ferramentas>Porta.
5. Verificar o Código usando Sketch>Verificar/Compilar.
6. Carregar o código compilado no microcontrolador usando Sketch>Carregar.

Assim, se tudo der certo, o nosso microcontrolador estará lendo os dados do sensor e aplicando a lógica do controle PID.

Para visualizar a leitura dos valores e o sinal de controle em tempo real, basta ir em Ferramentas > Plotter Serial, que vai exibir a janela da figura 11.

Onde, em azul temos o valor de Leitura da Luminosidade$y(t)$, em Vermelho temos o sinal do Compensador PID $u(t)$ e em Amarelo temos o Setpoint $r(t)$.

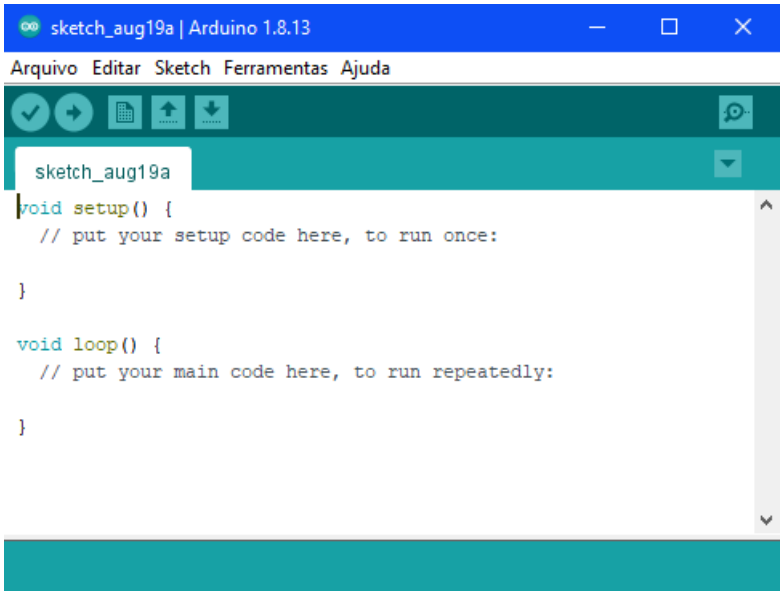

Figura 10 – IDE do Arduíno

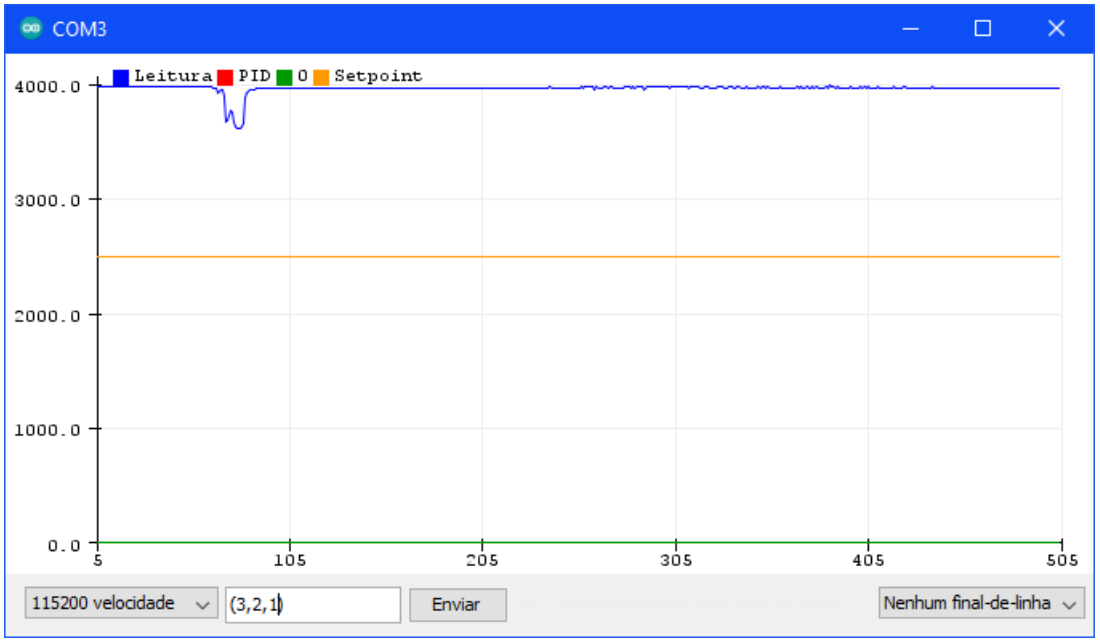

Figura 11 – Plotter Serial

Na parte de baixo da figura, podemos enviar os valores de Kp, Ki e Kd desejados. Por exemplo, na figura 11, estamos enviando Kp=3, Ki=2 e Kd=1.

Referências