

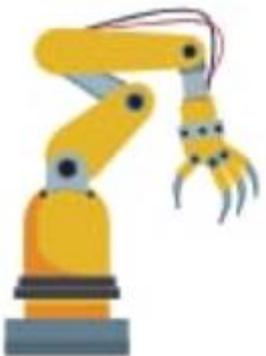


Automação Industrial

Projeto de Sensoreamento e Instrumentação por controle PID – Parte II

Prof^o Eng^o Hermom Leal, Msc.

Versão 2 – Outubro/2019



Controladores PID – Proporcional-Integral- Derivativo - (Ajuste de Parâmetros)

HERMOM LEAL MOREIRA

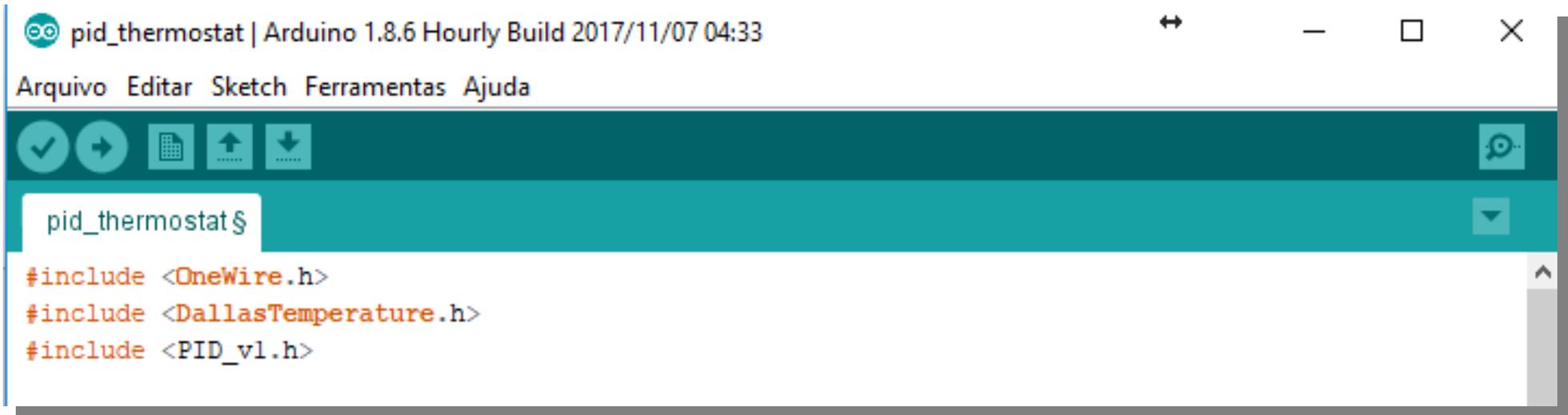
• Sumário

1. Introdução;
2. Testando o Controlador - Teste – k_p , k_i e $k_d=0$;
3. Teste P – $t_{ref}=30$, $k_p=50$, k_i e $k_d=0$;
4. Teste P – $t_{ref}=30$, $k_p=5000$, k_i e $k_d=0$;
5. Teste PID – $t_{ref}=30$, $k_p=50$, $k_i=20$ e $k_d=10$;

• Introdução

- Este experimento tem como base a utilização de um código principal denominado */pid_thermostat.ino*.
- O microcontrolador utilizado é o Arduino, sendo que, o código principal realiza a leitura de temperatura do sensor DS18B20, através de uma biblioteca do dispositivo denominado (*DallasTemperature.h*), que tem a função de realizar a leitura de temperatura externa.
- Além disto, o Arduino trabalha com bibliotecas, e para este experimento foram instaladas as bibliotecas denominadas *OneWire.h*, *Arduino-Temperature-Control-Library.h* e *PIDLibrary.h*

• Introdução



The image shows a screenshot of the Arduino IDE interface. The title bar reads "pid_thermostat | Arduino 1.8.6 Hourly Build 2017/11/07 04:33". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar contains icons for a checkmark, a right arrow, a keyboard, an up arrow, a down arrow, and a refresh icon. The main text area shows the following code:

```
pid_thermostat $  
#include <OneWire.h>  
#include <DallasTemperature.h>  
#include <PID_v1.h>
```

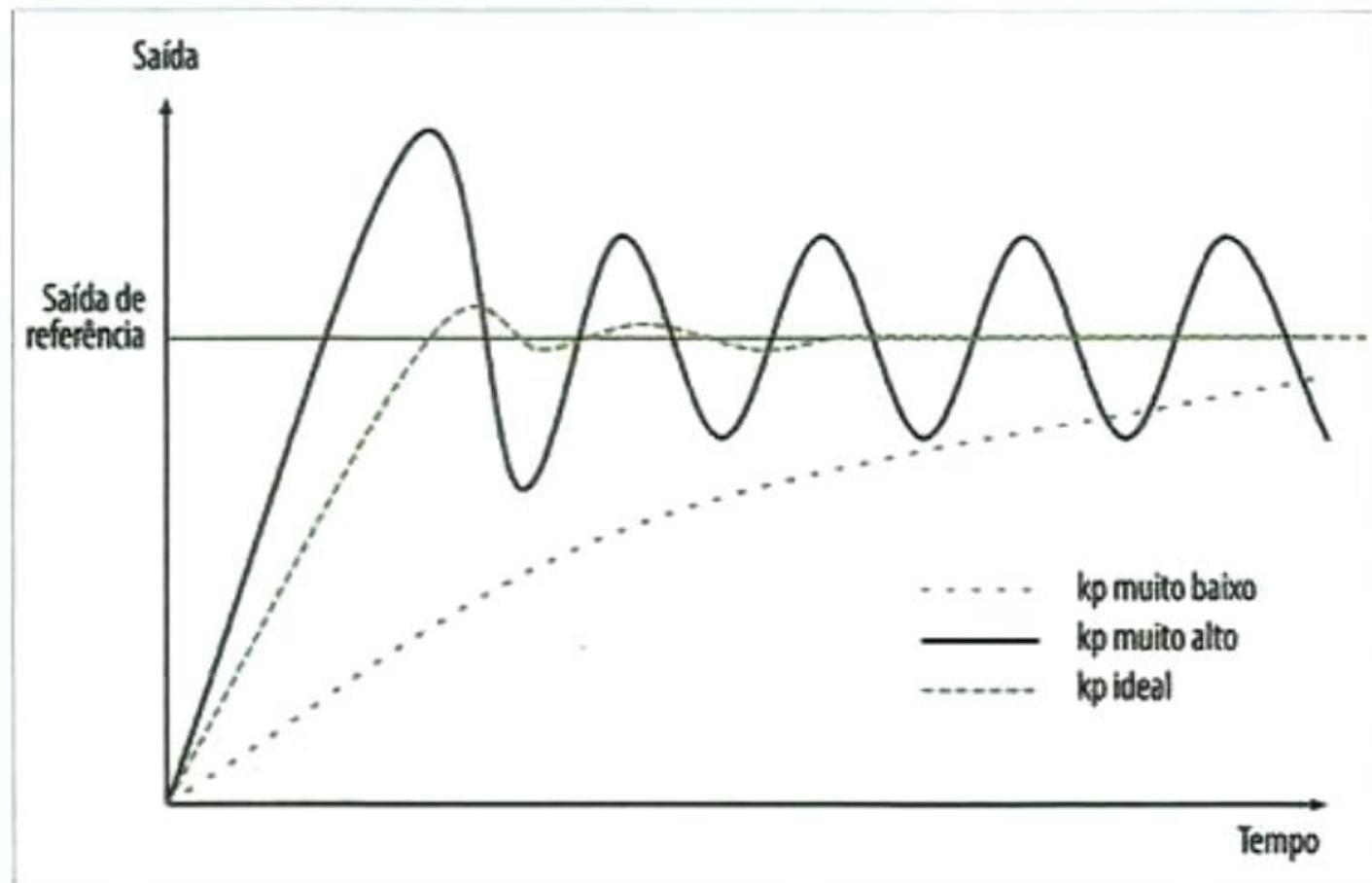
Figura 1. Bibliotecas Utilizadas

O objetivo do experimento é realizar o **teste do controlador PID, para a alteração da temperatura de referência e dos valores de k_p , k_i e k_d .**

Logo após será feito o registro dos dados obtidos na saída a fim de obter bom controle na saída.

2. Testando o Controlador;

Ao ajustar o efeito de ganho k_p (ganho) sobre a saída em um controlador proporcional, são obtidos os sinais conforme a Figura 2.



2. Testando o Controlador;

Quando os valores de **kp**, **ki** e **kd** são iguais a **(0)** o controlador não atua. Com isto, somente o sinal de temperatura recebido do sensor DS18B20, através da **entrada (Pin2)** do hardware será computada e conseqüentemente a **saída (Pin9)** não envia o sinal tipo PWM para a carga, ou seja, não ocorre a chamada retroalimentação conforme a Figura 3.

2. Testando o Controlador;

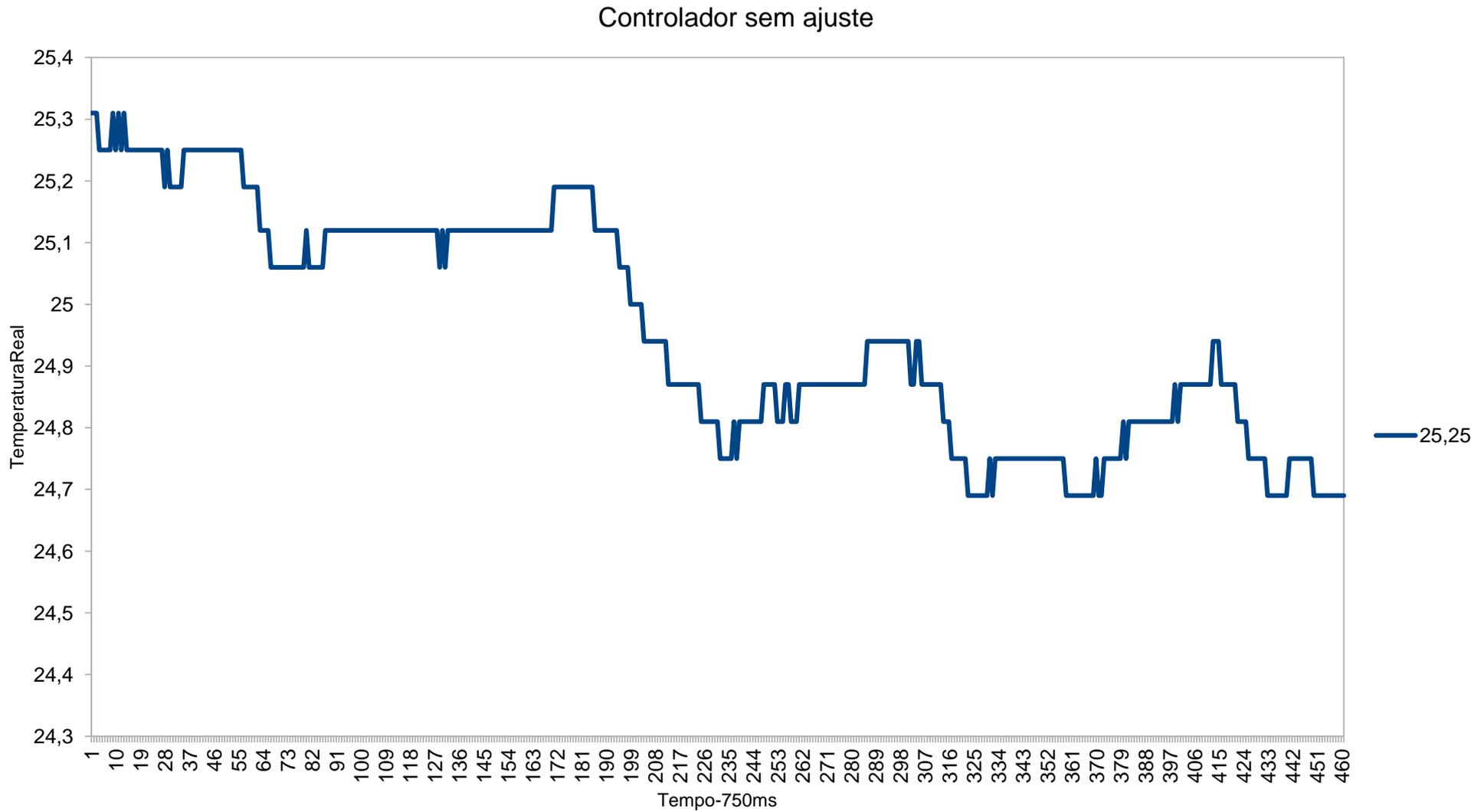
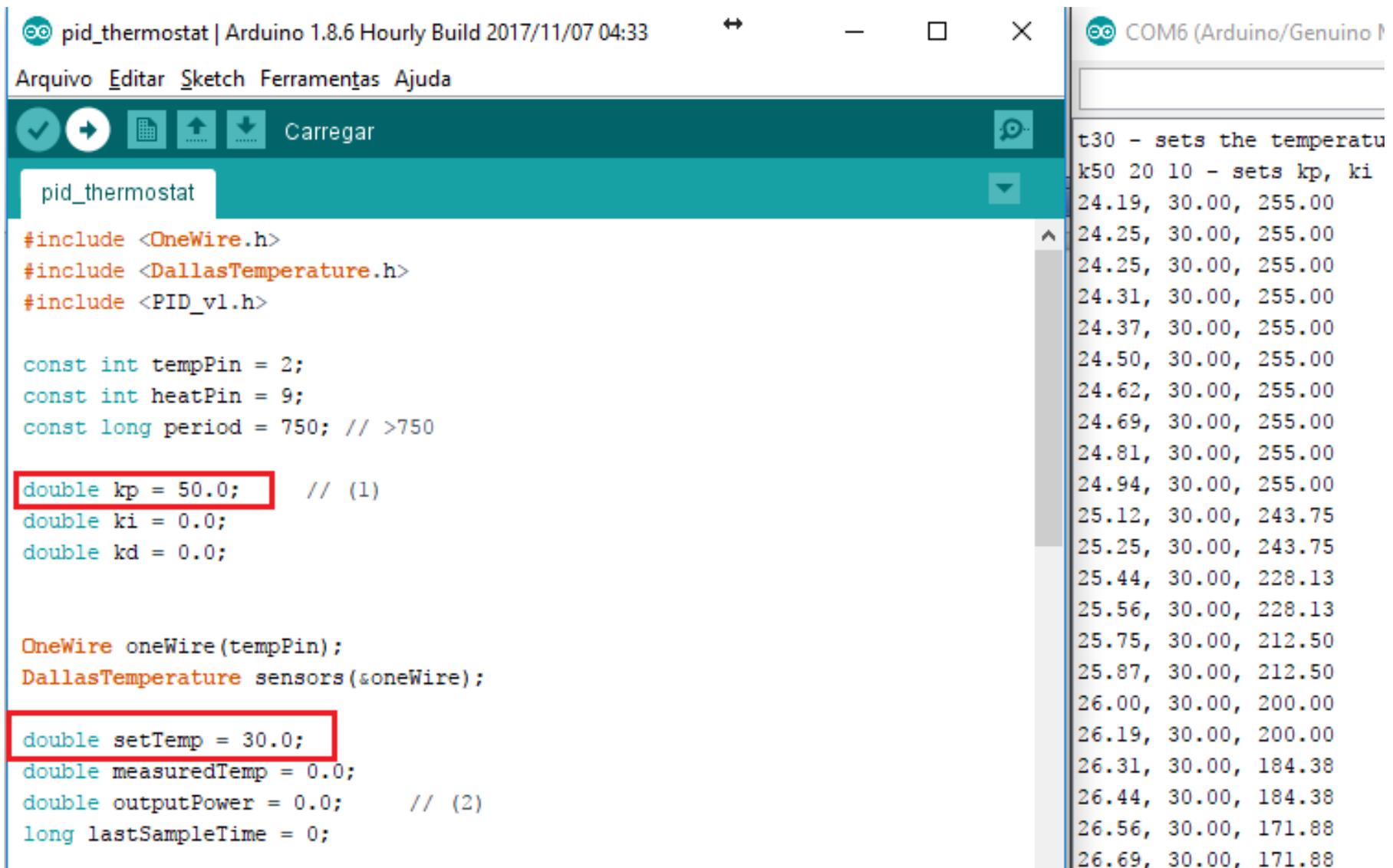


Figura 4 – Dados de saída (Output)
- Ajuste *Default*

3. Teste P – $t_{ref}=30$, $k_p=50$;

Nesta etapa são feitos ajuste no valor de temperatura de referência ($t_{ref}=30$), no valor de k_p ($k=50$). Com isto, é possível observar que a segunda coluna terá um valor fixo de 30 graus e terceira coluna terá um valor variando entre 0 e 255 referente ao sinal PWM, que buscará ajustar o valor da temperatura.

3. Teste P – $t_{ref}=30$, $k_p=50$;



The image shows the Arduino IDE interface. The main window displays the code for a PID thermostat. Two lines of code are highlighted with red boxes: `double kp = 50.0;` and `double setTemp = 30.0;`. The serial monitor on the right shows the output of the program, which consists of a series of three values separated by commas, representing the current temperature, the setpoint, and the output power.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>

const int tempPin = 2;
const int heatPin = 9;
const long period = 750; // >750

double kp = 50.0; // (1)
double ki = 0.0;
double kd = 0.0;

OneWire oneWire(tempPin);
DallasTemperature sensors(&oneWire);

double setTemp = 30.0;
double measuredTemp = 0.0;
double outputPower = 0.0; // (2)
long lastSampleTime = 0;
```

```
t30 - sets the temperatu
k50 20 10 - sets kp, ki
24.19, 30.00, 255.00
24.25, 30.00, 255.00
24.25, 30.00, 255.00
24.31, 30.00, 255.00
24.37, 30.00, 255.00
24.50, 30.00, 255.00
24.62, 30.00, 255.00
24.69, 30.00, 255.00
24.81, 30.00, 255.00
24.94, 30.00, 255.00
25.12, 30.00, 243.75
25.25, 30.00, 243.75
25.44, 30.00, 228.13
25.56, 30.00, 228.13
25.75, 30.00, 212.50
25.87, 30.00, 212.50
26.00, 30.00, 200.00
26.19, 30.00, 200.00
26.31, 30.00, 184.38
26.44, 30.00, 184.38
26.56, 30.00, 171.88
26.69, 30.00, 171.88
```

Figura 5 – Dados de entrada (Input)
- Ajuste $t_{ref}=30$, $k_p=50$

3. Teste P – $t_{ref}=30$, $k_p=50$;

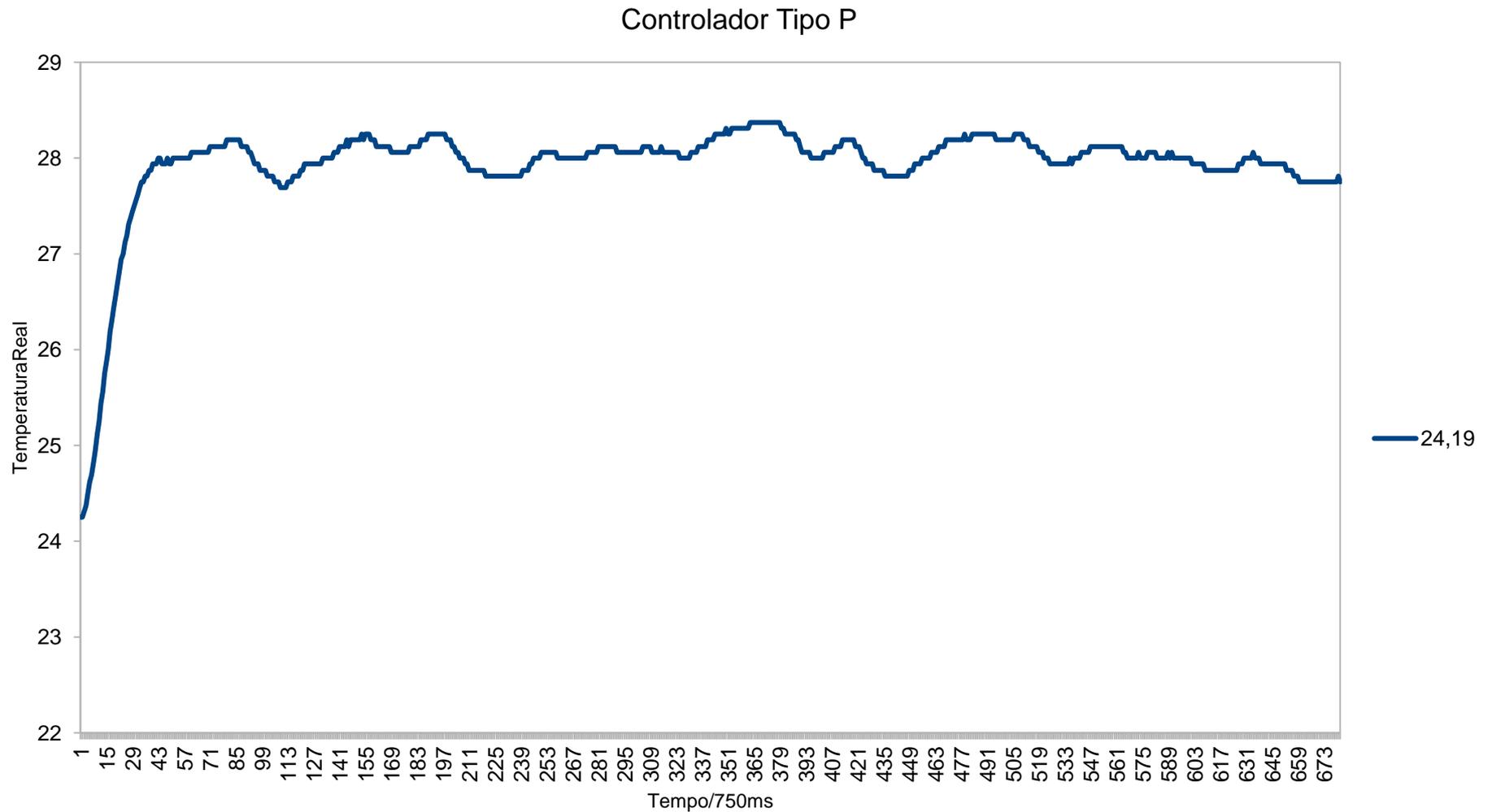
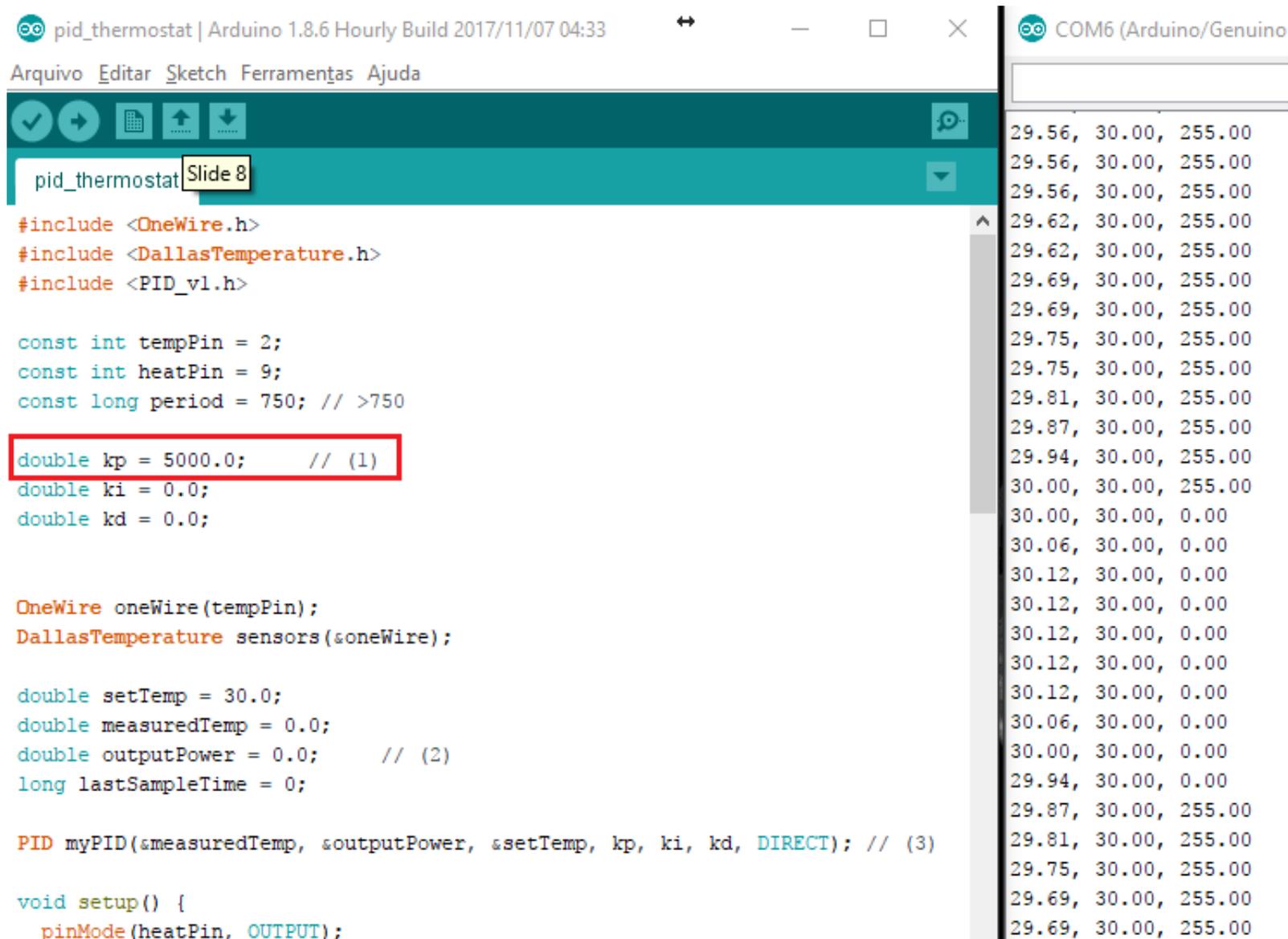


Figura 6 – Dados de saída (Output)
- Ajuste $t_{ref}=30$, $k_p=50$, k_i e $k_d=0$

3. Teste P – $t_{ref}=30$, $k_p=50$;

Análise:

4. Teste P – tref=30, kp=5000;



```
pid_thermostat | Arduino 1.8.6 Hourly Build 2017/11/07 04:33
Arquivo Editar Sketch Ferramentas Ajuda
pid_thermostat Slide 8
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>

const int tempPin = 2;
const int heatPin = 9;
const long period = 750; // >750

double kp = 5000.0; // (1)
double ki = 0.0;
double kd = 0.0;

OneWire oneWire(tempPin);
DallasTemperature sensors(&oneWire);

double setTemp = 30.0;
double measuredTemp = 0.0;
double outputPower = 0.0; // (2)
long lastSampleTime = 0;

PID myPID(&measuredTemp, &outputPower, &setTemp, kp, ki, kd, DIRECT); // (3)

void setup() {
  pinMode(heatPin, OUTPUT);
  // ...
}
```

Serial Output (COM6):

29.56	30.00	255.00
29.56	30.00	255.00
29.56	30.00	255.00
29.62	30.00	255.00
29.62	30.00	255.00
29.69	30.00	255.00
29.69	30.00	255.00
29.75	30.00	255.00
29.75	30.00	255.00
29.81	30.00	255.00
29.87	30.00	255.00
29.94	30.00	255.00
30.00	30.00	255.00
30.00	30.00	0.00
30.06	30.00	0.00
30.12	30.00	0.00
30.12	30.00	0.00
30.12	30.00	0.00
30.12	30.00	0.00
30.06	30.00	0.00
30.00	30.00	0.00
29.94	30.00	0.00
29.87	30.00	255.00
29.81	30.00	255.00
29.75	30.00	255.00
29.69	30.00	255.00
29.69	30.00	255.00

Figura 7 – Dados de entrada (Input)
- Ajuste tref=30, kp=5000, ki e kd=0

4. Teste P – $t_{ref}=30, k_p=5000;$

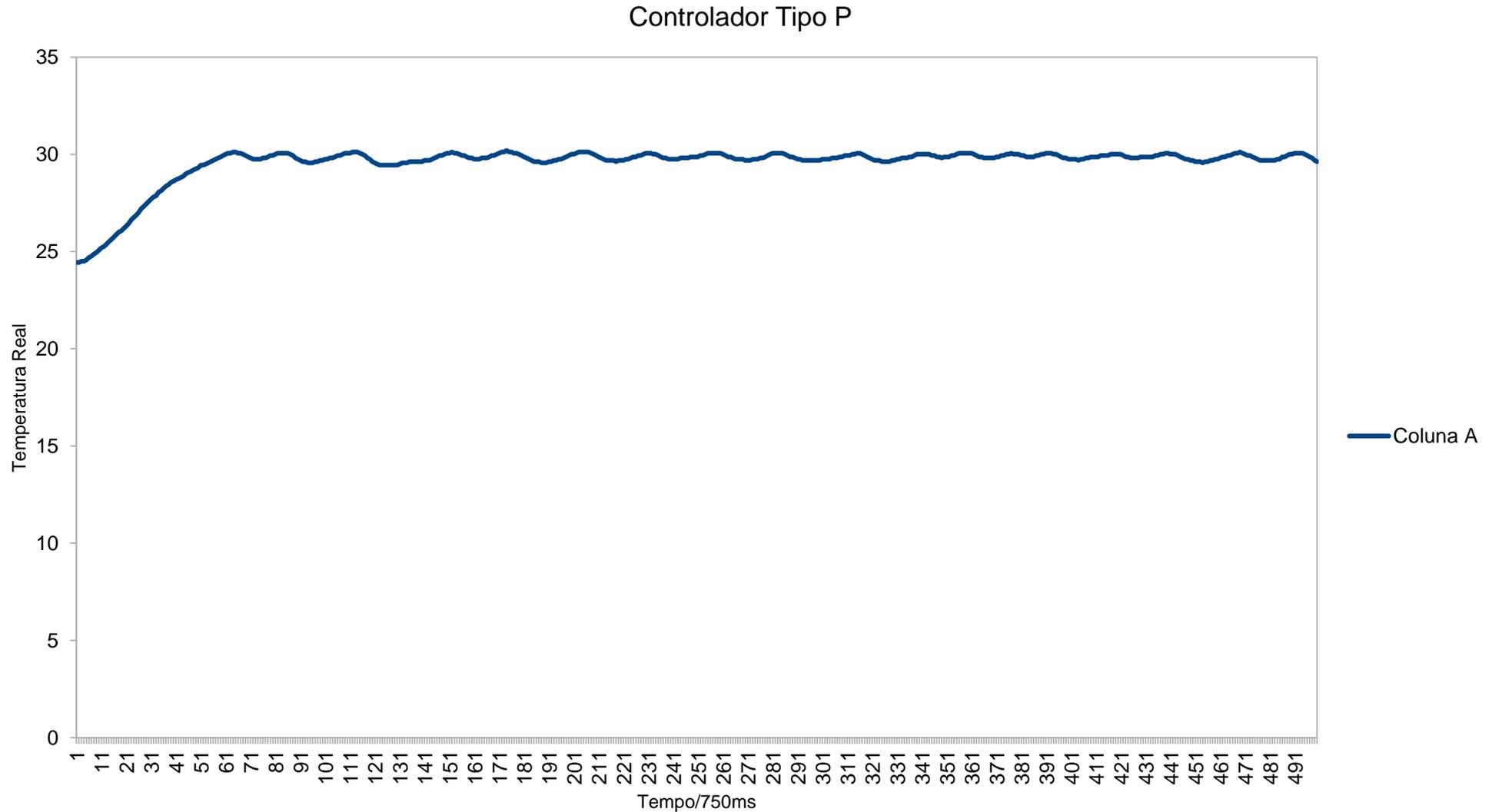


Figura 8 – Dados de saída (Output)
- Ajuste $t_{ref}=30, k_p=5000, k_i$ e $k_d=0$

4. Teste P – $t_{ref}=30, k_p=5000;$

Análise:

5. Teste PID – tref=30, kp=50, ki=20, kd=10

```
pid_thermostat | Arduino 1.8.6 Hourly Build 2017/11/07 04:33
Arquivo Editar Sketch Ferramentas Ajuda

pid_thermostat

#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>

const int tempPin = 2;
const int heatPin = 9;
const long period = 750; // >750

double kp = 50.0; // (1)
double ki = 20.0;
double kd = 10.0;

OneWire oneWire(tempPin);
DallasTemperature sensors(&oneWire);

double setTemp = 30.0;
double measuredTemp = 0.0;
double outputPower = 0.0; // (2)
long lastSampleTime = 0;

PID myPID(&measuredTemp, &outputPower, &setTemp, kp, ki, kd, DIRECT); // (3)
```

```
COM6 (Arduino/Genuino Mega or Mega 2560)
30.50, 30.00, 163.75
30.50, 30.00, 150.63
30.56, 30.00, 150.63
30.56, 30.00, 136.25
30.56, 30.00, 136.25
30.62, 30.00, 120.62
30.62, 30.00, 120.62
30.62, 30.00, 108.75
30.62, 30.00, 108.75
30.62, 30.00, 96.25
30.56, 30.00, 96.25
30.56, 30.00, 88.75
30.56, 30.00, 88.75
30.56, 30.00, 76.87
30.56, 30.00, 76.87
30.50, 30.00, 70.62
30.50, 30.00, 70.62
30.50, 30.00, 60.00
30.44, 30.00, 60.00
30.37, 30.00, 60.00
30.37, 30.00, 60.00
30.31, 30.00, 56.25
30.25, 30.00, 56.25
30.19, 30.00, 59.38
30.12, 30.00, 59.38
30.06, 30.00, 64.37
```

Figura 9 – Dados de entrada (Input)
- Ajuste tref=30, kp=50, ki=20 e kd=10.

5. Teste PID – $t_{ref}=30, k_p=50, k_i=20, k_d=10$

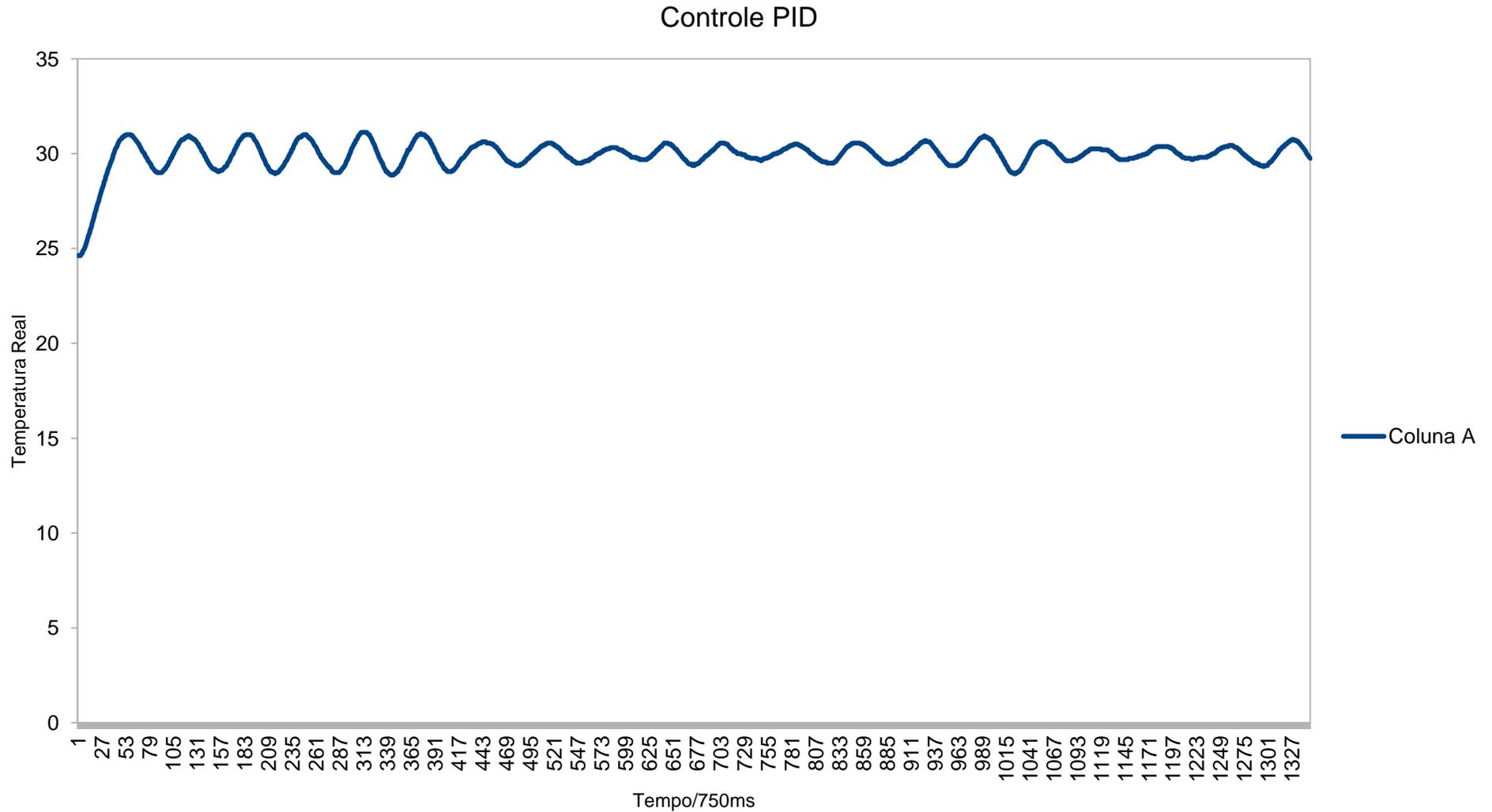


Figura 10 – Dados de saída (Output)
- Ajuste $t_{ref}=30, k_p=50, k_i=20$ e $k_d=10$.

5. Teste PID – $t_{ref}=30, k_p=50, k_i=20, k_d=10$

Análise: