Temperature Control Using PID

BY

NISHTHA PAREEK	2019B1A81044P
ANGEL MARIA BABY	2019B2A80997P
RAM SUKETU MEHTA	2019B3A80510P
PARTH MALU	2019B4A80683P

FOR THE COURSE

INSTR F343

INDUSTRIAL INSTRUMENTATION AND CONTROL

SUBMITTED TO

Prof. Puneet Mishra

And

Prof. Surekha Bahanot

Department of Electrical and Electronics Engineering

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,

PILANI

TABLE OF CONTENTS

INTRODUCTION

PID controller, which stands for Proportional Integral Derivative Controller, is a commonly used control algorithm in engineering. The PID controller automatically controls a process or system by modulating a control output to maintain a desired set point. The fundamental principle of the PID controller is to continuously measure the deviation between the desired set point and the actual process variable and then use this deviation to calculate the appropriate control output. The control output is calculated by adding proportional, integral, and derivative terms, each contributing differently to the overall control action.

$$u(t) = Kp * e(t) + Ki * \int e(t)dt + Kd * de(t)/dt$$

where u(t) represents the control output, e(t) represents the error between the set point and the process variable, and Kp, Ki, and Kd represent the proportional, integral, and derivative gain constants, respectively.

The proportional term is expressed as: $Kp * e(t)$
The integral term is expressed as: $Ki * \int e(t)dt$
The derivative term is expressed as: $Kd * de(t)/dt$

The PID controller adjusts the relative contributions of the proportional, integral, and derivative components to the overall control action via tuning parameters. These tuning parameters can be modified to optimise the efficacy of the controller for a specific application.
The PID controller is robust, which means it can maintain stable control even when disturbances and noise are present in the system. By modifying the control output based on the rate of change of the error, the PID controller can minimise system overshoot. The PID controller can minimise the system's settling time, which is the time required to reach a stable state after a change in the set point. By modifying the proportional, integral, and derivative gain constants, the PID controller's performance can be optimised for various applications.

TEMPERATURE CONTROL USING PID

Temperature control is essential to many industrial processes, and maintaining a specific temperature within a certain range is often necessary. A PID controller is widely used to control the temperature in such processes. A PID controller can maintain a constant temperature with high accuracy, reduce temperature oscillations, and increase energy efficiency. In addition, the PID controller can regulate the temperature in various industrial operations, such as chemical reactions, food processing, and HVAC systems.

PID controllers are typically used to regulate the temperature of a light fixture. The goal is to control the light bulb's temperature to prevent it from overheating and dying out. A thermistor, or temperature sensor, is used to measure the temperature of a light bulb. The temperature sensor is positioned in contact with the light bulb to measure its temperature and relay this data to the PID controller.

The proportional gain (Kp) of the PID controller determines the controller's temperature error response sensitivity. A greater value of Kp will cause the controller to respond more rapidly to variations in temperature, while a lower value will result in a slower response. The integral gain (Ki) is utilised to eradicate steady-state errors caused by a disparity between the desired and actual bulb temperature. A greater Ki value will expedite the elimination of steady-state errors but may also result in an overshoot. The derivative gain (Kd) attenuates temperature oscillations and reduces overshoot. A greater value of Kd will result in faster damping of oscillations but may also delay the response to changes in temperature.

In conclusion, controlling the temperature of a light bulb with a PID controller is an effective method for preventing the bulb from overheating. The ability of the PID controller to respond to temperature deviations, eradicate steady-state deviations, and dampen oscillations makes it an effective method for maintaining a constant temperature.

EXPERIMENT

The objective of this experiment is to use a proportional-integral-derivative (PID) controller to regulate the temperature of a conventional incandescent lightbulb. A 40 W bulb is chosen to ensure that the bulb's maximal temperature remains within the range of the employed temperature sensor.

The LM35 is the temperature sensor used in this experiment. This sensor measures temperature in Celsius and outputs a voltage proportional to the measured temperature. It has an acceptable temperature range and reasonable precision and requires no calibration. Using thermally conductive epoxy or adhesive metal tape, the LM35 sensor can adhere to the surface of the luminaire. The leftmost pin of the LM35 sensor is connected to the power supply (4 V to 30 V), the middle pin is the signal output, and the rightmost pin is ground. The Arduino board supplies the LM35 sensor with power and ground, and the signal output is received on one of the board's analog inputs.

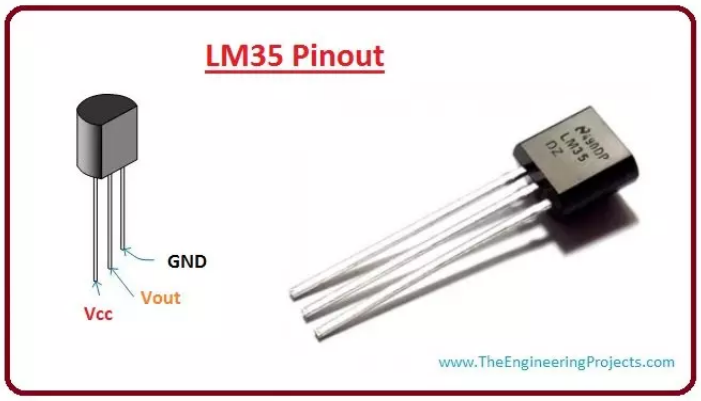


Figure 1: LM 35 Temperature Sensor

The bulb is controlled by a solid-state relay that is activated and deactivated by a digital output from the Arduino board. Using a low-power DC signal, the solid-state relay connects or disconnects a high-power device from an AC source. The relay is inserted into the neutral wire of the light bulb's socket.

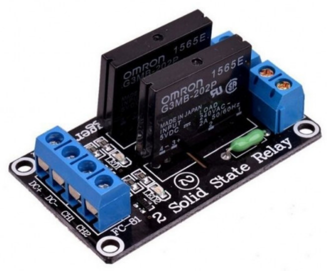


Figure 2: 5 V Solid-State Relay

Circuit Diagram

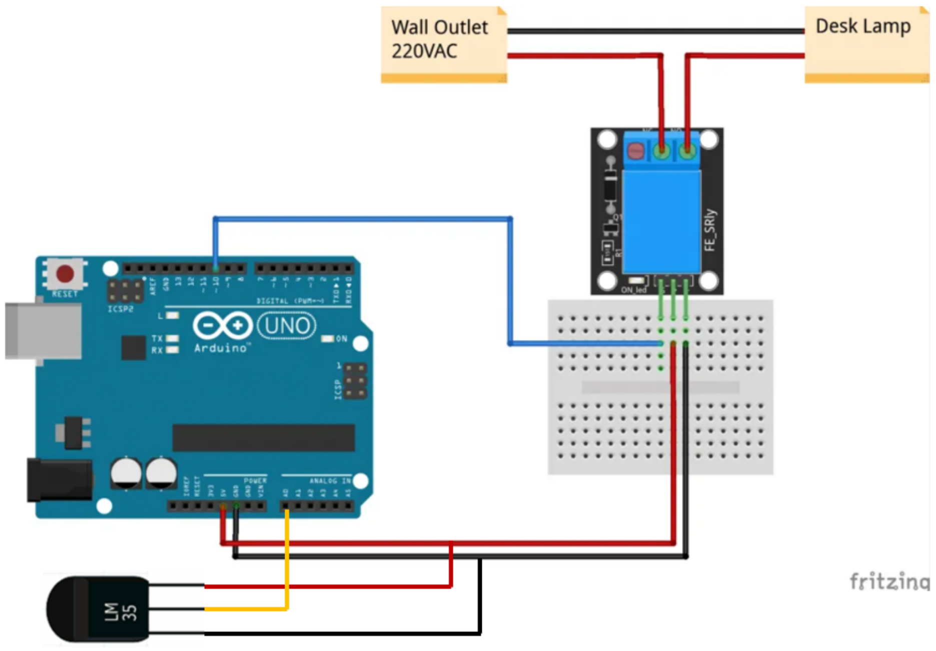


Fig3. Circuit Diagram

Experimental Setup

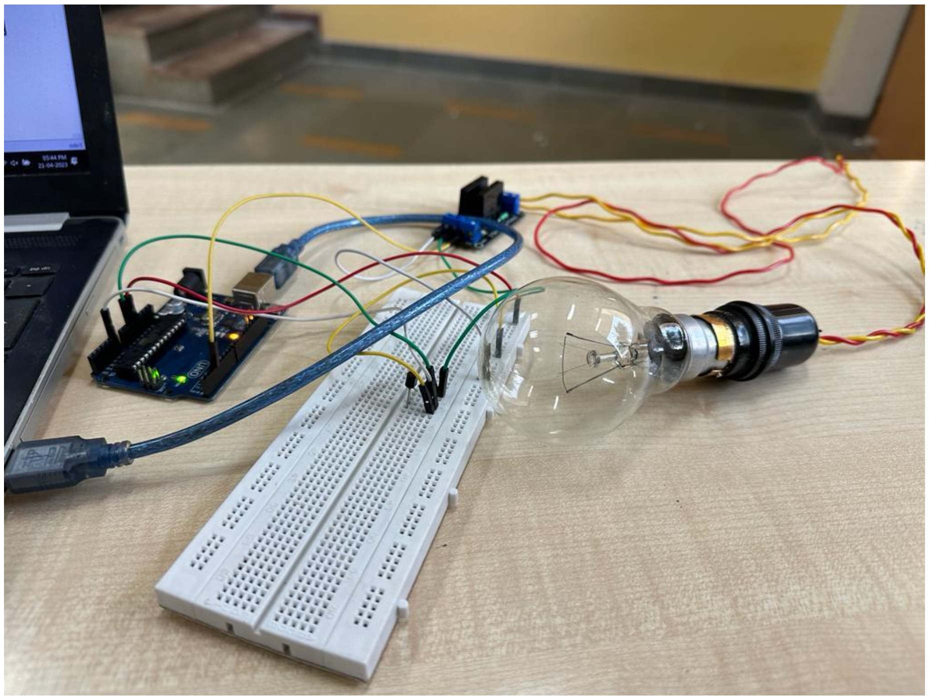


Fig4. Experiment Setup

In this experiment, we will use Simulink to control the relay, read data from a temperature sensor, and display the data in real time. The Simulink model for this experiment is displayed below.

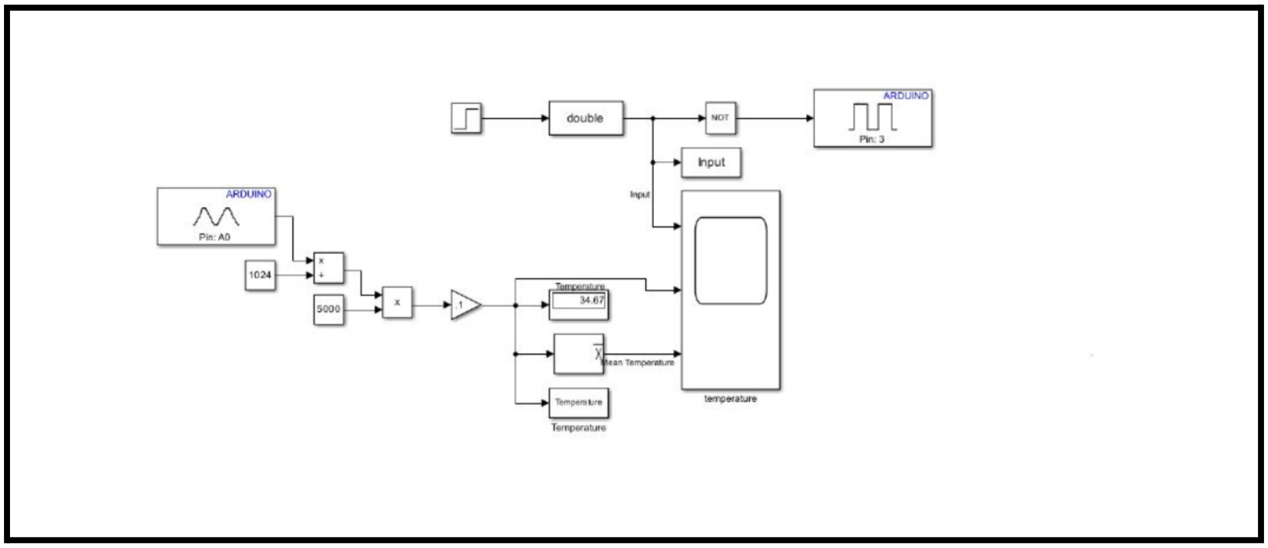


Fig5. Finding out process function for step input

The COM port in the IO Setup block is the location of the connected Arduino board. The model reads temperature data through an Analog Read on channel A0, converted from counts to degrees Celsius. The temperature data is in raw form and represented in bits. The Arduino board employs a 10-bit analog-to-digital converter, meaning that an analog input channel reads a voltage between 0 and 5 volts and splits that range into 1024 pieces. An output of 0 corresponds to 0 volts, and an output of 1023 corresponds to 5 volts. The data in bits are then converted to millivolts, assuming the default 5 volts reference, before being converted to temperature in degrees Celsius. The conversion is based on the information in the datasheet for the LM35.

The model also uses a Digital Write on channel 3 to command the relay to be open (1) or closed (0), corresponding to turning the lightbulb on and off. For this experiment, a step input will be used, turning on the lightbulb once, starting from the initial state where it is off.

MODELLING

A step input was given to the system, and the response was recorded. The response showed that the system was of the first order. From the plot of the recorded response, the value of static gain and time constant was determined.

The response of the system is shown below:

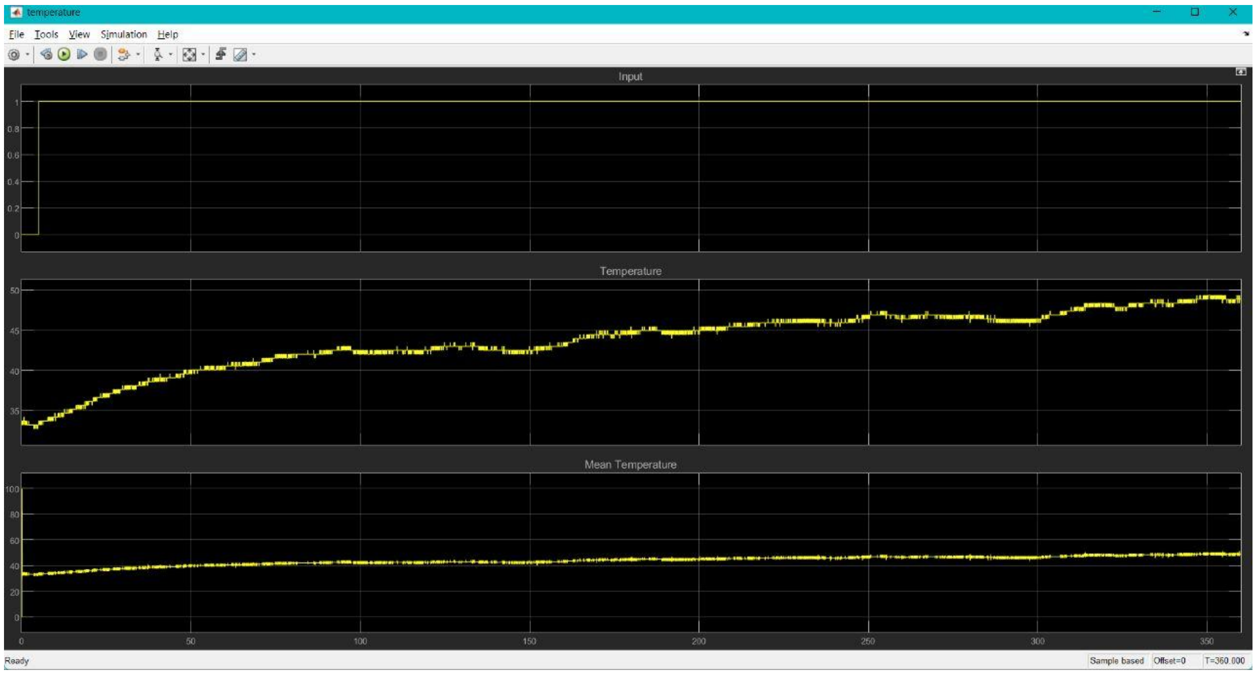


Fig6. Step input response

The steady state value of the system is extracted to find the static gain of a first-order system from a graph of its step response. The steady-state value is the value the response approaches as the time approaches infinity. Upon inspection of the provided step response data, the ambient temperature (initial lightbulb temperature) appears to be about 33.2 degrees Celsius. The steady-state light bulb temperature appears to be approximately 48.825 degrees Celsius. Given that the input is 1 (100 percent duty cycle) and the output is delta T, the system's DC gain K is approximately 15.625 degrees Celsius (48.825 - 33.2).

The time at which the response reaches 63 percent of its ultimate value corresponds to the constant(t). The time constant of the system is 119.150s.

After identifying the values them to calculate the transfer function of the system is calculated as

G(s) = K/(1 + s*t)

G(s) = 15.625/(1 + s*119.15)

Response of the system without controller

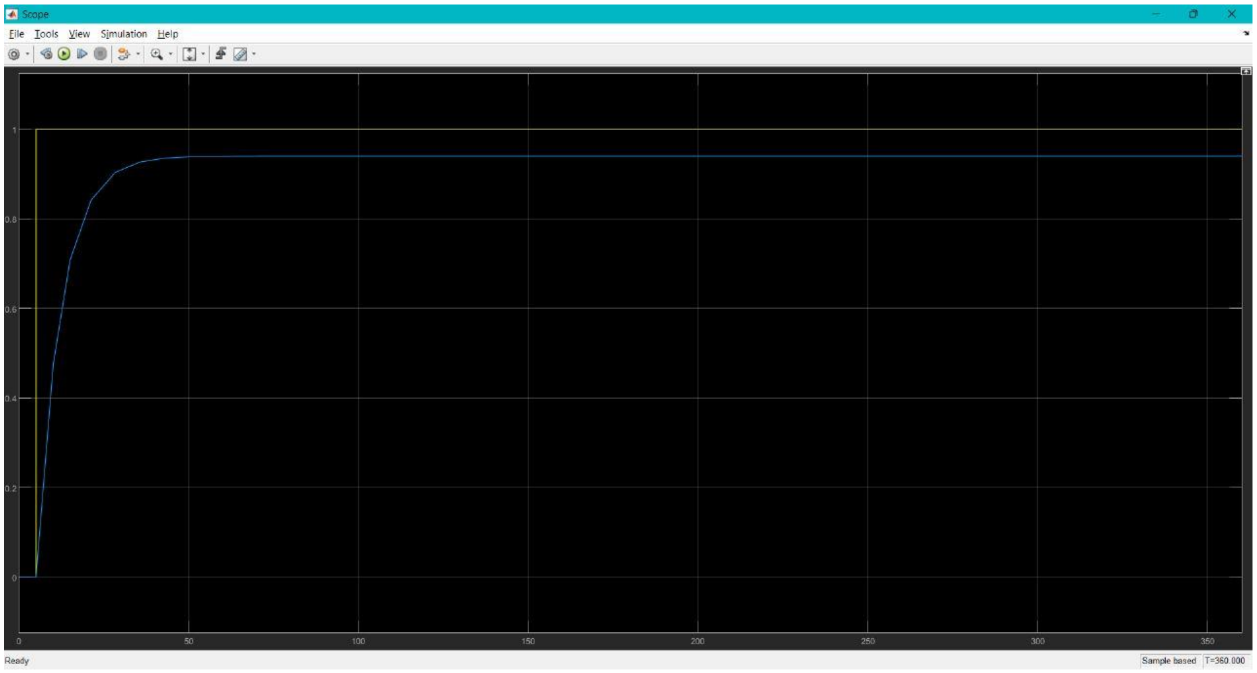


Fig7. Response of system without controller

This is the response obtained from the approximated model of system natural response when step input is given.

PID Controller

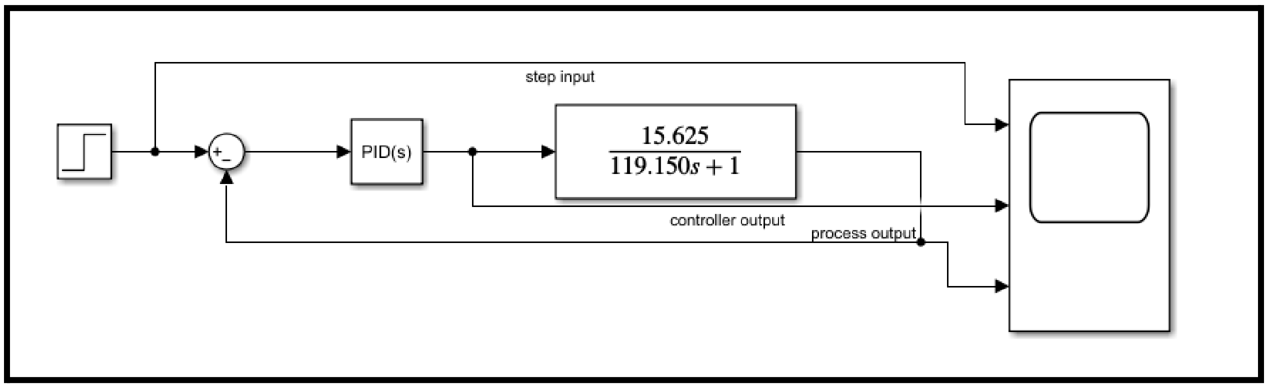


Fig8. PID tuning using transfer function

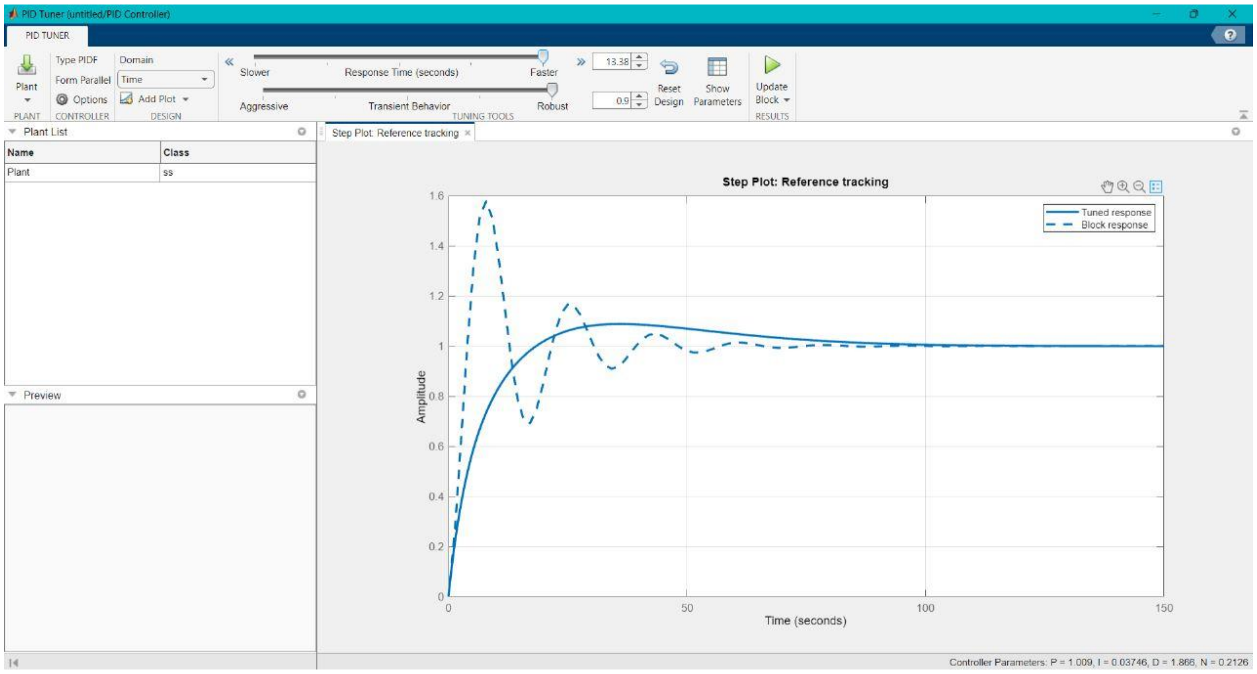
After obtaining the natural response of the system, we employed a PID Controller to regulate the temperature. The PID Controller was tuned to obtain maximum robustness and the fastest response.

<u>The obtained PID equation is</u>

1.008 + 0.037/4s +0.3955/(1+0.212/s)

P+I/s+D.N/(1+N/s)

PID Tuning



Type PIDF

Form Parallel

Domain Time

Response Time (seconds) 13.38

Transient Behavior 0.9

Name	Class
Plant	ss

Controller Parameters: P = 1.009, I = 0.03746, D = 1.866, N = 0.2126

Fig9. Tuned PID response

The dashed response was obtained by Kp = Ki = 1 and Kd = 0. We assumed it to be the initial condition used for tuning. And then, we changed the response time and transient behaviour to obtain the most robust and fastest response. The tuned response is shown in the solid curve. In the tuned response, the overshoot decreases because our Kd value in the tuned response has increased from the untuned case. Rise time has also decreased as Kp increased in the tuned case.

Tuned PID Response

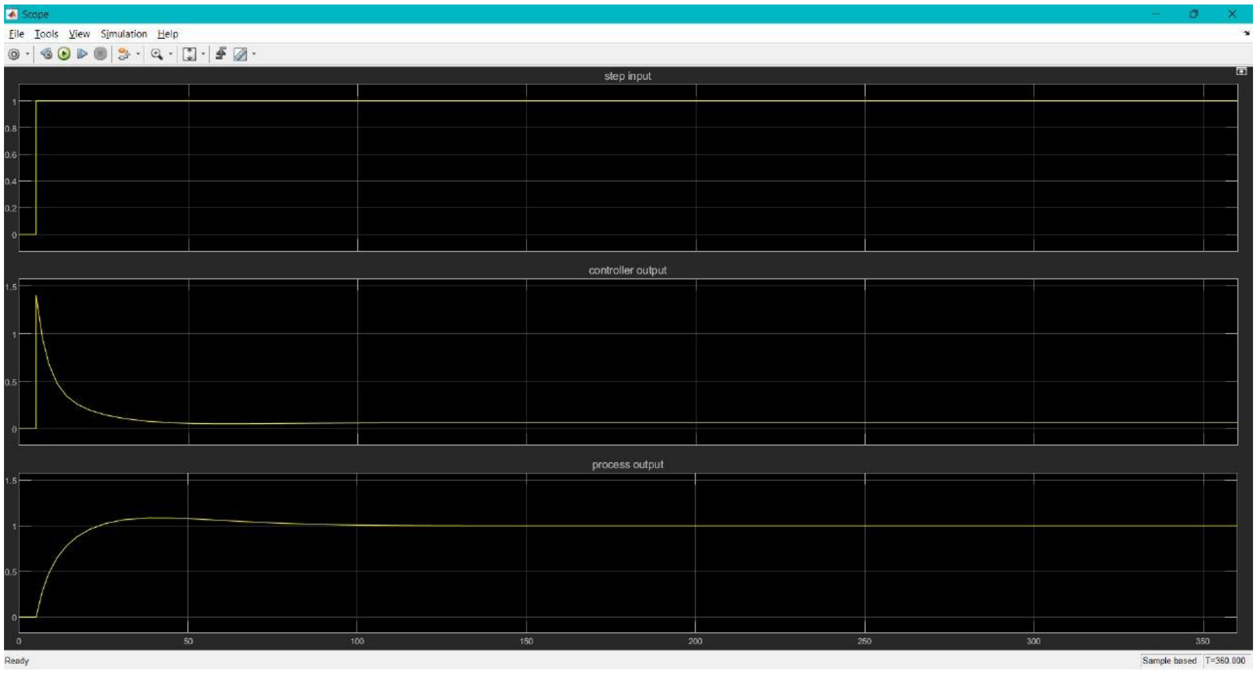


Fig10.Response of system with Tuned PID

Here, we can see that as in the beginning of the process, the controller output is maximum as error is maximum and controller tries to minimize it. As the error reaches to zero, the controller output diminishes showing an inverse response, which could also be seen from the response.

Once the PID parameters and the system transfer function were determined, the Simulink model of the controlled system was created. The PID controller was used in the feedback control loop, and the set temperature was 37.3 degrees.

RESULTS AND INTERPRETATION

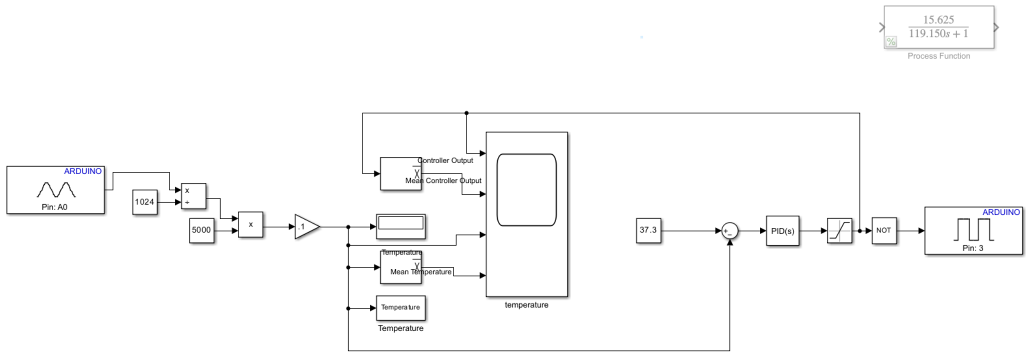


Fig11. System Model with tuned PID

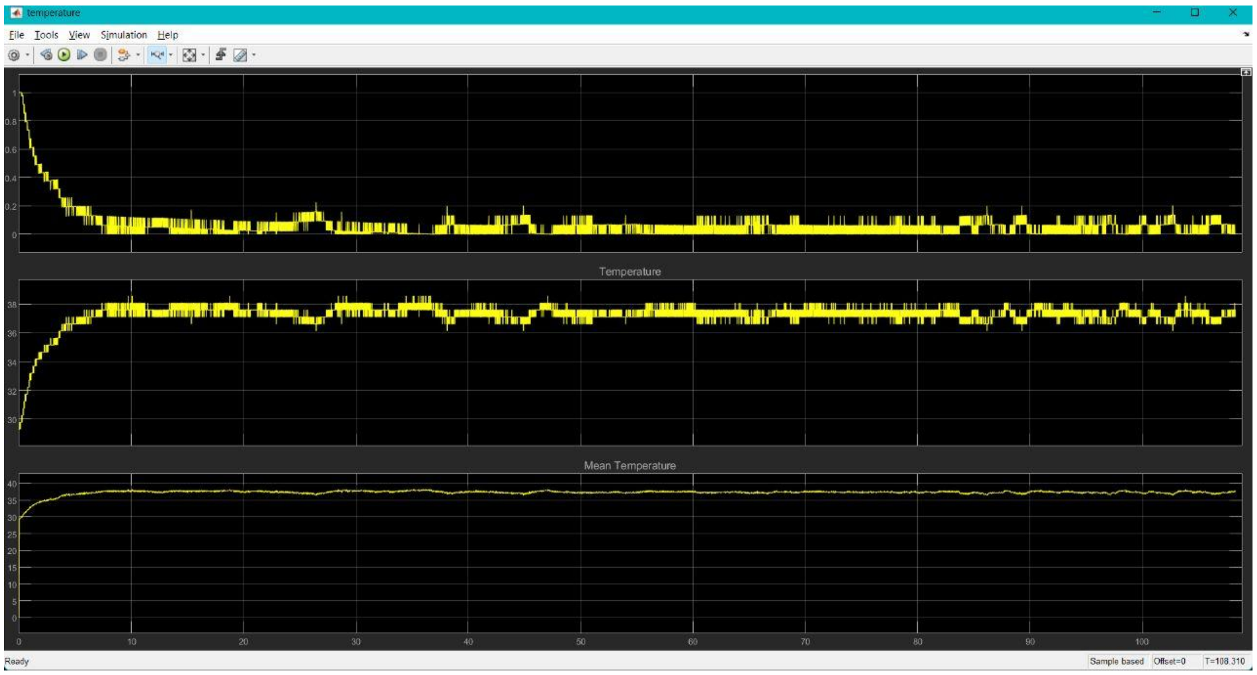


Fig12. System response with Tuned PID

As you run this model, observe the values of the control effort due to the proportional term of the controller and the integral term of the controller (as shown in the displays). Below is given the resulting temperature profile and control effort for this model. Initially, the error is large and gets smaller as the lightbulb heats up. Therefore, the proportional term of the control effort is initially large, decreasing as the error gets smaller. The integral portion of the control effort continues to grow even as the magnitude of the error decreases because the integral accumulates error, it is summing the area under the error "curve."

When the lightbulb's temperature reaches the desired level, the error and the proportional control effort are zero. The integral control effort, however, is still quite large because it has accumulated all of the positive errors since the control system was started. This is referred to as the integrator "winding up." Therefore, the lightbulb continues to heat up for a while and overshoots its desired temperature due to the integrator. At this point, the error is now negative, and the proportional control effort is negative.
Moreover, while the integral control effort is still positive, it is now starting to decrease. This process of unwinding the integrator takes time. Furthermore, the temperature can then undershoot the desired temperature as it decreases. This example illustrates why our closed-loop system is now second order, that is, why it can now have complex poles and oscillate.

CONCLUSIONS

1. The Natural response of the system is first order with transfer function:

   G(s) = K/(1 + s*t)

   G(s) = 15.625/(1 + s*119.15)

2. The transfer function of the PID Controller is:

   1.008 + 0.037/4s +0.3955/(1+0.212/s)

   (Form: P+I/s+D.N/(1+N/s))

   Thus we can conclude that the light bulb temperature is controlled by the above equations and we get a first order response for the designed feedback system.

REFERENCES

1. https://ctms.engin.umich.edu/CTMS/index.php?aux=Activities_Lightbulb
2. https://pidtuner.com/#/HJh0ZXer7P