Allen E. Paulson College of Engineering

Lab 9: Proportional Integral Derivative

Milton Tomas
Spring 2024, Academic Year (Sophomore)

Supervisor:
Dr. Soloiu

Georgia Southern University Department of Mechanical Engineering

MENG Class of 2026
Department of Mechanical Engineering

PROPORTIONAL INTEGRAL DERIVATIVE

Milton Tomas
Georgia Southern University
Statesboro, Georgia, US

ABSTRACT

In the following lab PID device was programmed to activate a cooling response if the thermocouple measured the temperature more than or equal to 30 degrees. In addition, an a led would help signal this change(red). If the thermocouple detected temperature below 30 the PID would cause the circuit board to stop and light up the other led (blue).

NOMENCLATURE

C	Celsius
Ω	Ohms (electrical resistance)
V	Voltage(J/C)

INTRODUCTION

PID, known as proportional integral derivative, is a controller to regulate temperature, flow, pressure, speed and other industrial variables. Proportional is the error term is multiplied by a constant known as K_p. Integral is the error term is accumulated over time. That accumulation is multiplied by a constant known as K_i. Derivative is the slope of the error term is calculated and multiplied by a constant K_d. When you add these terms together you will find that the following is the equation for a PID controller **Equation (1).**

With the PID, it is used to maintain the output of a process within a desired range. To get this achieved you need 4 basic variables. This includes the controlled variable, which is the parameter being maintain constant, the input variable that is a measuring tool for the controlled parameter, the error value and control signal that satisfies the setpoint variable. In other words, the value we choose.

Equation (1)

$$U(t)=K_p e(t) + Ki\int^{t}_{0} e(t)dt + K_d * de(t)/dt$$

EXPERIMENTAL METHODS

To replicate the following experiment, you will need a circuit board, a PID controller, a stripped AC power cable, two LEDs, a DC power supply, Thermocouple, 22 ohms resistor, jumper wires, and a computer fan.

Firstly, you will start by properly setting up the PID. Start by connecting the AC power cables to the proper ports (1 and 2). You will use the blue wire for port 1 and brown for the port 2. Connect three of the 6" wires in ports 3,4,5. Lastly, you will insert the wires from the thermocouple into ports 8 and 9.

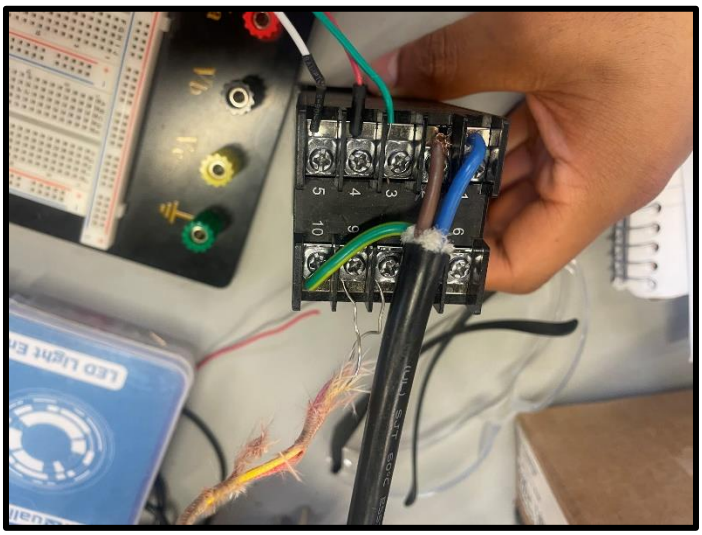


Figure 1

Next you will set parameters for the PID. You start by holding the setup key for two seconds to access the field parameter table. Change the AOP Alarm Output Assignment to 33 and ensure it is set to APID. You will then change the High Alarm and Low Alarm to 30 and 23.

Lastly, you will make a series circuit with the resistors and LED light in the materials needed in the first paragraph. Use the following figure to reference to as you build the circuit board. It is important to note that the DC Generator must not exceed 6.5v or else the fan will not run.

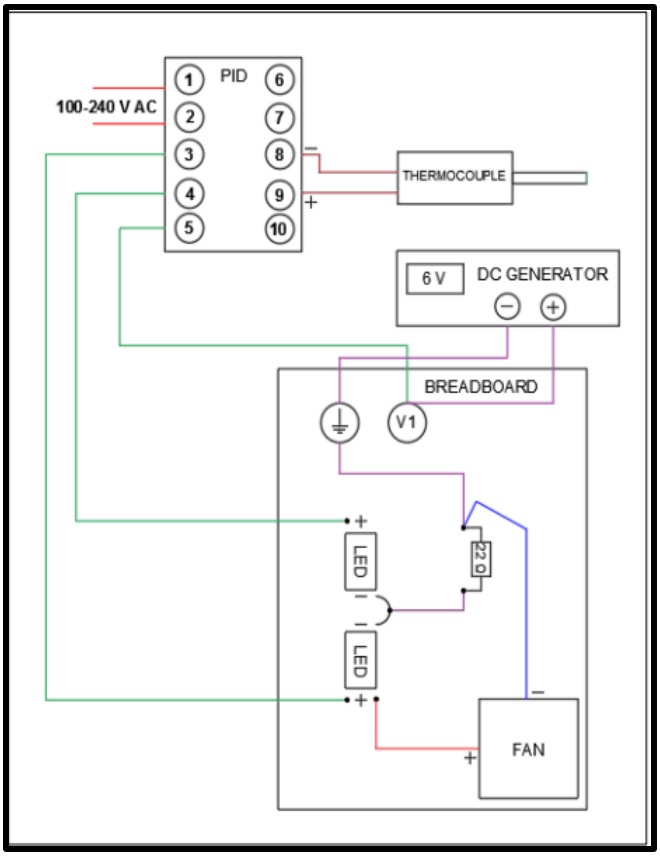


Figure 2 The completed Circuit Board Setup

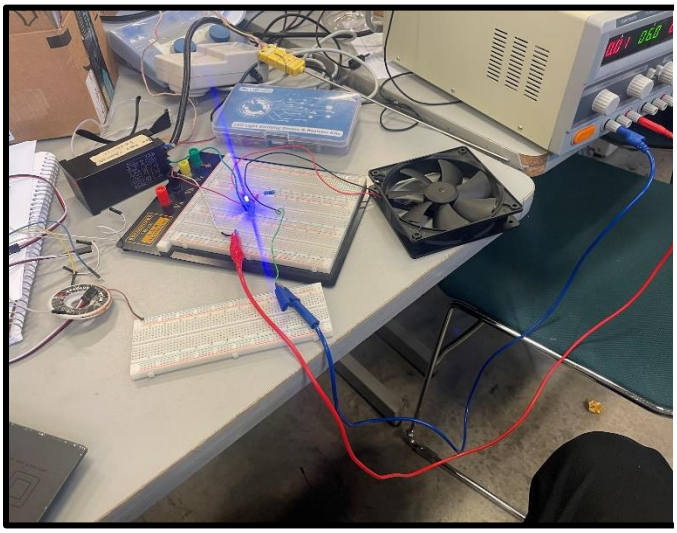


Figure 3 The following procedure is completed.

DATA

Based on the experiment, we find that there is two different scenarios that will play out depending on the temperature metered by the thermocouple. The blue LED light will be led if the temperature is below 30 Celsius. This is because we programmed the PID to that certain number. More specifically the range of the program was 22.8 to 30 Celsius. The other LED light would activate when the temperature exceeded 30 Celsius. Therefore, the cooling device is illuminated in response to the highest temperature it was programmed to.

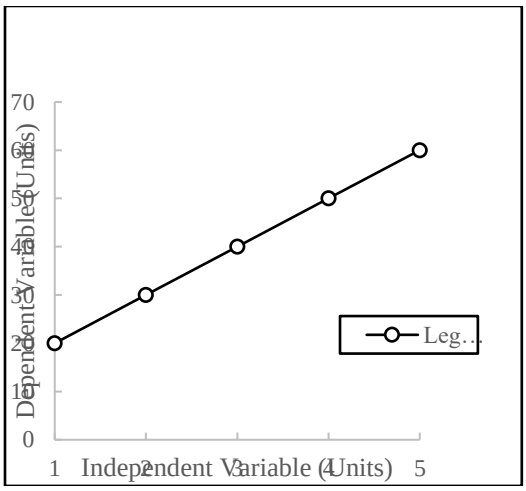
RESULTS

The results should contain: relevant equations used to perform calculations, sample calculations, quantitative results presented in tables and figures as appropriate to convey results. The equations and calculations performed should be introduced in paragraph form. Introduction to sample calculation.
Introduction to second sample calculation.

Equation (1)

$$U(t)=K_p e(t) + Ki\int_0^t e(t)dt + K_d * de(t)/dt$$

PID is built by computers or microcontrollers and programmable logic controllers (PLCs). Refer to the annex for the digitally coding for PID controller. It is important to point out that there is not actual calculus being done but rather you are adding the error values over time for the integral term. You find the difference between the current and previous error values for derivate term.



DISCUSSION

The discussion section should contain a discussion of the results and their physical significance, a comparison of experimental with theoretical/and or experimental data available in literature, and a statistical and experimental uncertainty analysis (standard deviation, equipment accuracy etc.). The minimum requirements for a statistical analysis are to show averages and standard deviations based off of a minimum of three trials. When describing experimental uncertainty the minimum requirements are to list equipment/measurement device accuracy. The results should not merely be stated but should be interpreted (i.e. what do the results mean?).

CONCLUSION

The conclusion should include a brief summary of what was done and significant findings. New material should not be incorporated in the conclusion. The conclusion should present quantitative findings not just qualitative descriptions of the results.

REFERENCES

[1] Soloui, Dr. V (2015) Engine Design and performance instruction manual. Georgia Southern department of mechanical engineering.

[2] Hymel, Shawn. “Introduction to Pid Controllers.” *DigiKey,* www.digikey.com/en/maker/projects/introduction-to-pid-controllers/763a6dca352b4f2ba00adde46445ddeb. Accessed 6 Apr. 2024.

ANNEX A

PUT ANNEX TITLE HERE

```
k_p = 1           # Tune this
k_i = 0           # Tune this
k_d = 0           # Tune this
interval = 0.001  # e.g. 1 ms

# Loop forever
error_prev = 0
integral = 0
while True:

    # Get value from sensor (feedback)
    val = read_from_sensor()

    # Calculate the PID terms
    error = setpoint - val
    integral = integral + (error * interval)
    derivative = (error - error_prev) / interval
    output = (k_p * error) + (k_i * integral) + (k_d * derivative)

    # Save value for next iteration
    error_prev = error

    # Wait for interval time
    sleep(interval)
```