

# PID Control and Auto-Tuning in the VentCon Pressure Control System

VENTREX

June 10, 2025

## Abstract

This document describes the implementation of the PID (Proportional-Integral-Derivative) control system and auto-tuning procedures in the VentCon pressure control system. The document details the core control algorithm, anti-windup mechanisms, hysteresis compensation, and the relay-based auto-tuning method with multiple tuning rules for different performance requirements.

## Contents

|          |                                      |          |
|----------|--------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                  | <b>2</b> |
| <b>2</b> | <b>PID Control Implementation</b>    | <b>2</b> |
| 2.1      | Basic PID Theory . . . . .           | 2        |
| 2.2      | Implementation in VentCon2 . . . . . | 2        |
| 2.3      | Anti-Windup Implementation . . . . . | 3        |
| 2.4      | Hysteresis Compensation . . . . .    | 3        |
| <b>3</b> | <b>Auto-Tuning Procedure</b>         | <b>4</b> |
| 3.1      | Relay Method Theory . . . . .        | 4        |
| 3.2      | Implementation in VentCon2 . . . . . | 4        |
| 3.3      | Multiple Tuning Rules . . . . .      | 5        |
| 3.4      | Adjustable Aggressiveness . . . . .  | 5        |
| 3.5      | Auto-Tuning Workflow . . . . .       | 6        |
| <b>4</b> | <b>Performance Considerations</b>    | <b>7</b> |
| 4.1      | Tuning Rule Selection . . . . .      | 7        |
| 4.2      | Comparative Step Responses . . . . . | 7        |
| 4.3      | Practical Recommendations . . . . .  | 8        |
| 4.4      | Limitations . . . . .                | 9        |
| <b>5</b> | <b>Conclusion</b>                    | <b>9</b> |

## 1 Introduction

The VentCon2 system is designed to provide precise pressure control using a solenoid valve operated via PWM (Pulse Width Modulation). The system employs a digital PID controller to maintain the desired pressure setpoint by dynamically adjusting the valve opening based on real-time pressure measurements. To simplify the tuning process, an automated self-tuning procedure based on the relay method is implemented.

## 2 PID Control Implementation

### 2.1 Basic PID Theory

The PID control algorithm calculates the control output  $u(t)$  based on the error  $e(t)$  between the desired setpoint and the measured process variable [7]:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

Where:

- $K_p$  is the proportional gain
- $K_i$  is the integral gain
- $K_d$  is the derivative gain
- $e(t) = \text{setpoint} - \text{measured value}$

### 2.2 Implementation in VentCon2

The VentCon2 system uses the PID\_v2 library for Arduino which implements a discrete form of the PID equation. The controller is configured with the following parameters:

```

1 // PID initialization
2 PID pid(&pressureInput, &pwmOutput, &settings.setpoint,
3         settings.Kp, settings.Ki, settings.Kd, DIRECT);
4 pid.SetMode(PID::Automatic);
5 pid.SetOutputLimits(0, pwm_max_value);
6 pid.SetSampleTime(SAMPLE_TIME); // 20ms (50Hz)

```

Listing 1: PID Configuration

The controller updates at a frequency of 50Hz, providing a good balance between responsiveness and processing overhead. The PID output is then mapped to the valve's effective operating range (50% to 90% duty cycle), as the valve exhibits non-linear behavior at lower duty cycles.

## 2.3 Anti-Windup Implementation

The system implements an advanced anti-windup mechanism that prevents integral windup in two key situations:

1. When the valve is below its minimum effective operating range (50% duty cycle)
2. When the valve is above its maximum effective operating range (90% duty cycle)

The anti-windup mechanism works by temporarily switching the PID controller to manual mode and back to automatic when these conditions are detected, effectively resetting the integral component:

```
1 if (settings.antiWindup)
2 {
3     float pidPercent = (pwmOutput / pwm_max_value) * 100.0;
4
5     if ((pidPercent < VALVE_MIN_DUTY && pwmOutput >
6         previousOutput) ||
7         (pidPercent > VALVE_MAX_DUTY && pwmOutput >
8         previousOutput))
9     {
10        // Reset the PID to prevent integral accumulation
11        pid.SetMode(PID::Manual);
12        pid.SetMode(PID::Automatic);
13    }
14 }
```

Listing 2: Anti-Windup Implementation

## 2.4 Hysteresis Compensation

Solenoid valves typically exhibit hysteresis behavior, where the opening and closing characteristics differ. To compensate for this, the system implements a direction-based hysteresis compensation mechanism:

```
1 if (settings.hysteresis)
2 {
3     bool currentlyIncreasing = pressureInput > lastPressure;
4
5     if (currentlyIncreasing != pressureIncreasing)
6     {
7         if (!currentlyIncreasing)
8         {
9             // When changing direction from increasing to decreasing
10            pidPercent += settings.hystAmount;
11        }
12        else
13        {
14            // When changing direction from decreasing to increasing
```

```

15     pidPercent -= settings.hystAmount;
16 }
17
18     pressureIncreasing = currentlyIncreasing;
19 }
20 }

```

Listing 3: Hysteresis Compensation

This approach helps overcome the "stiction" effect in solenoid valves, providing more consistent control regardless of whether the pressure is increasing or decreasing.

### 3 Auto-Tuning Procedure

#### 3.1 Relay Method Theory

The auto-tuning implementation uses the relay method developed by Åström and Hägglund [1], which is a variation of the Ziegler-Nichols frequency response method [2]. This approach is widely used in industrial applications due to its reliability and ease of implementation [8]. Instead of increasing the proportional gain until sustained oscillation occurs (which can be difficult and potentially unsafe), the relay method:

1. Forces the system into controlled oscillation using a relay (on/off) controller
2. Measures the resulting oscillation amplitude and period
3. Calculates the "ultimate gain" ( $K_u$ ) and "ultimate period" ( $T_u$ )
4. Applies tuning rules to determine the optimal PID parameters

The ultimate gain is calculated as:

$$K_u = \frac{4d}{\pi a} \quad (2)$$

Where:

- $d$  is the relay amplitude (AUTOTUNE\_RELAY\_STEP = 75%)
- $a$  is the resulting oscillation amplitude in the process variable

#### 3.2 Implementation in VentCon2

The auto-tuning process in VentCon2 works as follows:

1. The system sets the target pressure to a predefined setpoint (3.0 bar)

2. A relay controller alternates the valve between:
  - 0% (fully closed) when pressure exceeds the setpoint
  - 75% of effective valve opening when pressure falls below setpoint - noise band
3. The system records the cycle times and amplitudes for 10 complete cycles (or until timeout)
4. The ultimate gain and period are calculated from the averaged measurements
5. PID parameters are calculated using the selected tuning rule

An important implementation detail is the handling of the valve's non-linear behavior. The relay amplitude (75%) refers to the actual valve opening percentage rather than the raw PWM value. When applying the relay output in the high state, the system accounts for the valve's characteristics by mapping this percentage to the appropriate PWM value:

```

1 // When in high state:
2 float mappedDuty = VALVE_MIN_DUTY + (AUTOTUNE_RELAY_STEP_RAW /
   100.0) * (VALVE_MAX_DUTY - VALVE_MIN_DUTY);
3 uint32_t pwmValue = (uint32_t)((mappedDuty / 100.0) *
   pwm_max_value);
4 ledcWrite(PWM_CHANNEL_MOSFET, pwmValue);
5
6 // When in low state:
7 ledcWrite(PWM_CHANNEL_MOSFET, 0); // Completely closed

```

Listing 4: Auto-Tuning Relay Output

This ensures the actual valve behavior matches the theoretical model used for parameter calculation.

### 3.3 Multiple Tuning Rules

The VentCon2 system implements four different tuning rules to accommodate different control requirements [8]:

Where Z-N stands for Ziegler-Nichols, and "aggr" is the aggressiveness factor (default 1.2) that can be adjusted from 0.5 to 2.0.

The system defaults to the Ziegler-Nichols Aggressive rule with an aggressiveness factor of 1.2, as this provides a good balance between response speed and stability for most applications.

### 3.4 Adjustable Aggressiveness

A unique feature of the VentCon2 auto-tuning system is the adjustable aggressiveness factor, which provides an additional layer of control over the tuning outcome:

Table 1: PID Tuning Rules [12]

| Tuning Rule   | $K_p$                      | $K_i$         | $K_d$         |
|---|----------------------------|---------------|---------------|
| <b>Z-N Classic</b> [2]<br>Balanced response with moderate overshoot         | $0.6K_u$                   | $1.2K_u/T_u$  | $0.075K_uT_u$ |
| <b>Z-N Aggressive</b><br>Faster response with potential increased overshoot | $0.7K_u \cdot \text{aggr}$ | $1.75K_u/T_u$ | $0.085K_uT_u$ |
| <b>Tyreus-Luyben</b> [3]<br>Reduced overshoot, slower response              | $0.45K_u$                  | $0.36K_u/T_u$ | $0.14K_uT_u$  |
| <b>Pessen Integral</b> [4]<br>Fast setpoint tracking                        | $0.7K_u$                   | $1.75K_u/T_u$ | $0.105K_uT_u$ |

- Values from 0.5 to 1.0 provide more conservative control with less overshoot
- The default value of 1.2 provides faster response than classic Z-N
- Values from 1.5 to 2.0 generate very aggressive control for applications requiring minimal response time

This factor primarily affects the proportional gain but influences the overall system behavior, allowing fine-tuning without changing the fundamental tuning rule.

```

1 // Example of how aggressiveness affects parameter calculation
2 newKp = 0.7 * Ku * tuningAggressiveness; // Proportional gain
   scaled by aggressiveness

```

Listing 5: Aggressiveness Implementation

### 3.5 Auto-Tuning Workflow

The typical auto-tuning workflow for a user is:

1. Select the desired tuning rule (optional): `TUNE RULE n`
  - 0: Ziegler-Nichols Classic (balanced)
  - 1: Ziegler-Nichols Aggressive (faster response) - Default
  - 2: Tyreus-Luyben (minimal overshoot)
  - 3: Pessen Integral (fast setpoint tracking)
2. Adjust aggressiveness if needed (optional): `TUNE AGGR x`
3. Start the auto-tuning process: `TUNE START`
4. Wait for the process to complete (typically 1-3 minutes)

5. Review the calculated parameters
6. Accept or reject the parameters: `TUNE ACCEPT` or `TUNE REJECT`

Users can view available tuning rules and the current settings with the command `TUNE RULES`.

## 4 Performance Considerations

### 4.1 Tuning Rule Selection

- **Ziegler-Nichols Classic:** Provides a balanced response, but may exhibit significant overshoot
- **Ziegler-Nichols Aggressive:** Faster response time at the cost of potentially increased overshoot
- **Tyres-Luyben:** More conservative, with reduced overshoot but slower response
- **Pessen Integral:** Emphasizes setpoint tracking with fast recovery from disturbances

### 4.2 Comparative Step Responses

Figure 1 illustrates the typical step responses achieved with each tuning rule. These theoretical response curves demonstrate the relative differences in performance characteristics.

The plot illustrates several important characteristics:

- **Z-N Classic (blue):** Shows a moderate rise time with significant overshoot (approximately 32%) and some oscillation before settling.
- **Z-N Aggressive (red):** Demonstrates the fastest rise time with the highest overshoot (approximately 42%) but can take longer to fully settle due to more pronounced oscillations.
- **Tyres-Luyben (green):** Exhibits the slowest rise time but with minimal overshoot (approximately 5%), making it ideal for applications where overshoot must be minimized.
- **Pessen Integral (orange):** Provides a fast rise time with moderate overshoot (approximately 35%) but settles more quickly than the Ziegler-Nichols methods, making it well-suited for setpoint tracking applications.

The aggressiveness factor in the Z-N Aggressive method can shift its response curve between the Classic and Aggressive profiles shown here, providing flexible tuning options.

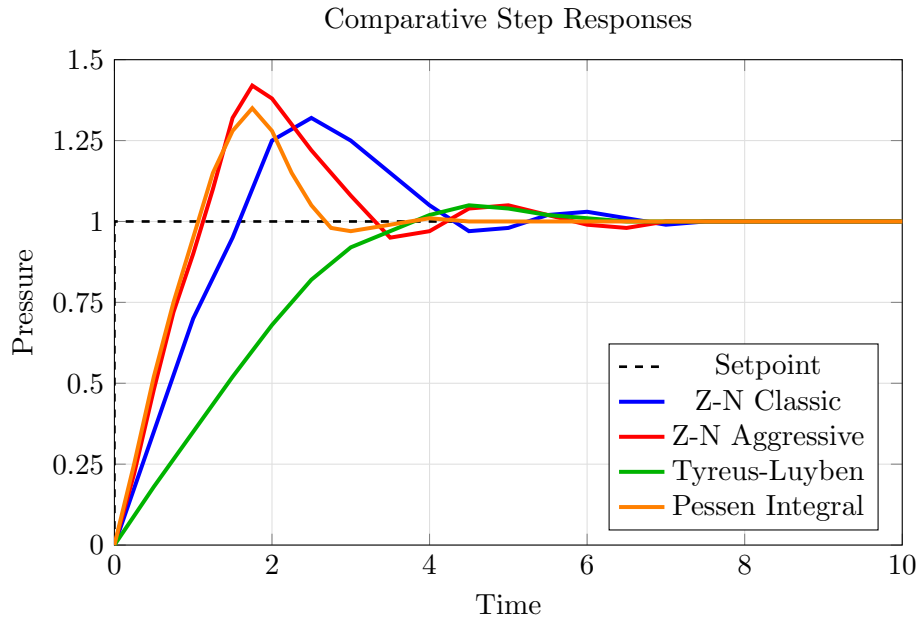


Figure 1: Typical step responses for different tuning rules applied to the pressure control system. The plot shows how each rule affects rise time, overshoot, and settling characteristics.

### 4.3 Practical Recommendations

1. Start with the default (Ziegler-Nichols Aggressive, aggressiveness=1.2)
2. If the response is too sluggish:
  - First try increasing aggressiveness: `TUNE AGGR 1.5` or `TUNE AGGR 2.0`
  - If still insufficient, try Pessen Integral rule: `TUNE RULE 3`
3. If overshoot is problematic:
  - First try reducing aggressiveness: `TUNE AGGR 0.8`
  - If still insufficient, switch to Tyreus-Luyben: `TUNE RULE 2`
4. Enable anti-windup (`AW ON`) for all tuning methods to improve recovery from saturation
5. For valves with significant stiction, enable hysteresis compensation (`HYST ON`)
6. After auto-tuning, fine-tune individual parameters if necessary:
  - To reduce overshoot: reduce  $K_p$  or increase  $K_d$



- To eliminate steady-state error: increase  $K_i$
- To reduce oscillation: increase  $K_d$  or reduce  $K_p$

#### 4.4 Limitations

The auto-tuning method has several limitations to be aware of:

- Works best for linear or approximately linear systems
- May not produce optimal results for highly non-linear systems
- Performance can be affected by noise (mitigated with `AUTOTUNE_NOISE_BAND`)
- Requires a stable starting point for best results
- External disturbances during tuning can affect accuracy

### 5 Conclusion

The VentCon2 system implements a sophisticated PID control system with advanced features like anti-windup and hysteresis compensation. The automated tuning procedure based on the relay method provides an accessible way to optimize the control parameters for specific applications without requiring in-depth knowledge of control theory [5].

The multiple tuning rules offer flexibility to balance between response speed, overshoot, and disturbance rejection based on the specific requirements of the application [6] [9] [10] [11].

### 6 References

#### References

- [1] Åström, K.J. and Hägglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5):645–651.
- [2] Ziegler, J.G. and Nichols, N.B. (1942). Optimum settings for automatic controllers. *Transactions of the ASME*, 64:759–768.
- [3] Tyreus, B.D. and Luyben, W.L. (1992). Tuning PI controllers for integrator/dead time processes. *Industrial & Engineering Chemistry Research*, 31(11):2625–2628.
- [4] Pessen, D.W. (1954). A new look at PID-controller tuning. *Journal of Basic Engineering*, 76:494–501.

- [5] Åström, K.J. and Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, NC, 2nd edition.
- [6] O'Dwyer, A. (2009). *Handbook of PI and PID controller tuning rules*. Imperial College Press, London, 3rd edition.
- [7] Wikipedia (2023). PID controller. [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller) [Online; accessed November 2023]
- [8] Wikipedia (2023). PID controller tuning. [https://en.wikipedia.org/wiki/PID\\_controller#Loop\\_tuning](https://en.wikipedia.org/wiki/PID_controller#Loop_tuning) [Online; accessed November 2023]
- [9] Control Guru (2023). PID control and auto-tuning resource site. <https://controlguru.com/> [Online; accessed November 2023]
- [10] MathWorks (2023). PID Tuning Algorithms. <https://www.mathworks.com/help/control/ug/pid-tuning-algorithms.html> [Online; accessed November 2023]
- [11] Control Engineering Practice (2023). PID Control Tuning: Principles and Applications. <https://blog.opticontrols.com/archives/344> [Online resource; accessed November 2023]
- [12] National Instruments (2023). PID theory explained. <https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html> [Online; accessed November 2023]