**Pneumatic MotOR SPEED control**

**Objectives:**

In this lab your group will:

* Explore the fundamentals of pneumatic systems and their applications.
* Design and implement a closed-loop control system to regulate the speed of a pneumatic motor.
* Investigate how tuning **PID controller gains** affects system performance characteristics such as **rise time, settling time, and overshoot.**

**Introduction**

In this experiment, you’ll develop and implement a closed-loop speed control system for a pneumatic motor. The setup uses a **solenoid-operated binary valve** controlled in real time via **Simulink** and an **Arduino-based control platform.**

This hands-on activity provides practical experience with:

* Pneumatic actuation and control hardware
* Real-time system interfacing through Simulink
* PID controller tuning and its impact on dynamic system response

By the end of the lab, you will be able to analyze and interpret how adjustments to control parameters influence the transient and steady-state behavior of a pneumatic motor system.

**Background**

**Pneumatic Motors**

A pneumatic motor (also called an air motor) is a type of actuator that converts the energy of compressed air into rotary mechanical motion. These motors are widely used in applications where lightweight, simple, and explosion-safe actuation is required — such as in assembly lines, hand tools, packaging systems, and automation setups.

**How It Works:**  
Compressed air enters the motor and acts on vanes, pistons, or turbines inside a housing, producing rotational motion. The speed of a pneumatic motor is typically controlled by adjusting the airflow rate into the motor. Higher airflow increases torque and speed, while restricting airflow reduces them.

**Types of Pneumatic Motors:**

* **Vane Motors:** Use sliding vanes inside an eccentric rotor to produce motion.
* **Piston Motors:** Use reciprocating pistons arranged radially or axially.
* **Turbine Motors:** Use high-speed air jets impinging on turbine blades.

In this lab, you’ll work with a **vane-type pneumatic motor**, which is known for:

* Simple design and construction
* High-speed capability
* Smooth operation
* Rapid response to control inputs

**Speed Control in Pneumatic Motors:**  
Traditionally, pneumatic motor speed is controlled by modulating the airflow using a proportional flow control valve. In this experiment, however, a binary (on/off) solenoid valve is used. By rapidly opening and closing the valve using pulse-width modulation (PWM), an average airflow can be controlled over time, thus regulating motor speed.

This setup highlights both the versatility and limitations of pneumatic systems, especially when precise control is needed in applications without proportional valves.

**5-Port, 3-Position, Closed-Center Solenoid-Operated Valve (Configured for One-Direction Motor Control)**

In this lab, the pneumatic motor is actuated using a 5-port, 3-position, closed-center, solenoid-operated valve. While such valves are often used for double-acting cylinders, here it is configured to control airflow for one-direction motor operation. Here’s how it works in this setup:

**5 Ports:**

* **Port 1 (Pressure Inlet, P)**: Connected to the compressed air supply.
* **Port 2 (Motor Inlet, B)**: Connected to the pneumatic motor.
* **Port 3-5** (The remaining three ports are unused in this configuration.)
* **Motor Exhaust**: Releases air from the motor outlet to the atmosphere.

1. **Positions:**
2. **Open (Forward Flow) Position**: Connects the compressed air supply to the motor inlet (Port 1 to Port 2), while exhausting the motor outlet through the motor exhaust port.
3. **Closed-Center Position**: All ports are blocked, stopping airflow to the motor and holding its speed steady.
4. **(Reverse Position is unused)**: In this experiment, the reverse flow position is not utilized since the motor runs in a single direction.

**Solenoid-Operated Binary Control:**  
The valve is actuated by electrically controlled solenoids, functioning as digital (on/off) actuators. Control signals from an Arduino running a Simulink-based program rapidly switch the valve state. By adjusting the pulse width (the ratio of time the valve is open to the total cycle time), the system effectively modulates the average airflow to the motor.

A closed-loop feedback system using a fiber optic speed sensor and PID controller continuously adjusts the motor’s speed by modulating the PWM signal, enabling controlled, real-time regulation of motor performance.

**Pneumatic Motor System**

**Figure 1** illustrates the configuration of the pneumatic motor system. Compressed air enters the system through the pressure inlet (Port 1) of a 5-port, 3-position, closed-center solenoid-operated valve. When actuated, the valve connects Port 1 to the motor inlet (Port 2) to supply air, while the motor exhaust exits through the motor exhaust port. The motor runs in a single direction. A speed sensor measures motor speed, feeding data to a Simulink-based control program on an Arduino, which adjusts the valve’s pulse-width modulation (PWM) signal to regulate airflow and control motor speed.



Figure 1: Schematic of pneumatic motor system with 5-Port, 3-Position, Closed-Center Solenoid-Operated Valve

**Procedure**

1. **A close up of a machine

   AI-generated content may be incorrect.Power Up the System**
   * Connect the power cord for the Pneumatic Educational Platform and turn on the **Main Switch** located at the back.
   * Turn on the **Compressor Toggle Switch** on the front of the platform to start filling the air tank.
2. **A close-up of a machine

   AI-generated content may be incorrect.Verify Pneumatic Connections**
   * Check that the pneumatic circuit is connected correctly according to the provided schematic diagram (see **Figure 1**).
   * Use the **Filter-Pressure Regulator** to set the system pressure to **75 psi**.
3. A close-up of a circuit board

   AI-generated content may be incorrect.**Select the Speed Sensor**
   * On the **Electronics Bay**, to select the speed sensor slide the **latching Switch** so that the **LCD screen** shows RPM on the screen, if not selected already.
   * Latching Switch is labeled as 2 on the figure to the right.
4. **Open and Run the Simulink Control Model**
   * Open the provided starting **Simulink model** designed for open-loop control of the pneumatic motor.
   * In Simulink, select the **“HARDWARE” tab** from the command ribbon.
   * Click the **“Monitor & Tune”** icon to compile and download the control code to the Arduino.  
     *Note: Simulink will compile the model into C-code, load it onto the Arduino, and initiate communication.*
5. **Verify Model Execution**
   * When prompted by a pop-up window summarizing the model build and deployment process, select **“OK”** to proceed.
   * A screenshot of a computer

     AI-generated content may be incorrect.The system will then begin controlling the pneumatic motor in open-loop mode.
6. **Control the Solenoid Valve Manually**
   * Familiarize yourself with controlling the solenoid valve by changing the speed of the motor by changing the duty cycle of the PWM signal through the Simulink interface.
7. **Develop and Implement Closed-Loop Control**
   * Develop a closed-loop control system that causes the pneumatic motor to cycle between **300 RPM and 700 RPM mm**, then back to **300 RPM** over a **30-second cycle** using both a square and sine wave.
   * Tune your PID gains to track the input reference
   * Use the **minimum control gain values necessary** to satisfy these requirements.
8. **Evaluate System Performance**
   * Use the **cursor measurement tools** in Simulink to analyze system response characteristics (rise time, overshoot, steady-state error, and settling time).
   * Make detailed observations about the dynamic behavior and control performance of the system.
   * Hold the wheel with your had momentarily and describe what happens because of the controller.

**Results and Discussion**

**1. Open-Loop Control Observations**

* Describe the system’s behavior when manually controlling the solenoid valve in open-loop mode.
* What happens as you increase the duty cycle for the PWM signal?
* Are there any limits on the speed sensor? Can it keep up with the motor?

**2. Closed-Loop System Response**

**a. PID Controller Tuning**

* Record the final PID gain values you selected (Kp, Ki, Kd).
* Briefly explain your tuning approach (e.g., trial and error, increasing proportional gain first, etc.).
* Describe how changes in each gain affected system performance (rise time, overshoot, steady-state error, settling time).

**b. Measured Performance**

* Provide a summary of the key performance metrics for your final controller when following the square wave:
  + **Steady-State Error (RPM):**
  + **Percent Overshoot (%):**
  + **±2% Settling Time (s):**
* Include screenshots or plots of the system meting the performance criteria.
* Discuss how the system differed when following the sine-wave

**4. Sources of Error and System Limitations**

* Identify possible sources of error in your experiment:
  + Sensor calibration inaccuracies
  + Air leaks or pressure fluctuations
  + Valve switching delays
  + Air compressibility effects
* Discuss how these might have affected your results and performance measurements.

**5. Recommendations for Improvement**

* Suggest at least two ways to improve system performance or experimental setup.
* Examples might include:
  + Using proportional valves for finer control
  + Improving sensor resolution
  + Adjusting cycle timing
  + Enhancing the control algorithm (e.g., implementing feedforward control)
* Discuss how the motor can be driven in both directions and draw the revised pneumatic circuit to accomplish this.