THINKERING LAB

EXPERIMENT: CNC TURNING OPERATION

🎯 AIM:

To study and perform a turning operation on a CNC lathe and evaluate machining parameters such as cutting speed, feed rate, and material removal rate (MRR).

□ APPARATUS / TOOLS REQUIRED:

- CNC Turning Machine (e.g., FANUC CNC Lathe controller-based system)
- Workpiece material (Mild Steel / Aluminum rod)
- Cutting tool (Single-point carbide tool)
- Tool holder
- Vernier caliper / Micrometer
- Coolant system
- Chuck and tailstock
- Computer interface for program input

📘 THEORY

Turning is a machining process in which a single-point cutting tool removes material from a rotating workpiece to produce a cylindrical shape.

Key parameters:

- **Cutting Speed (V):** Speed at which the workpiece surface moves past the cutting tool
- **Feed (f):** Distance the tool advances per revolution
- **Depth of Cut (d):** Thickness of material removed in one pass

Types of Turning:

- Plain turning
- Step turning
- Taper turning
- Facing

The CNC machine automates these operations using programmed instructions (G-code).

⚙ FORMULAS

Cutting Speed (V)

$$V = \frac{\pi D N}{1000}$$

Where:

- V = Cutting speed (m/min)

- DDD = Diameter of workpiece (mm)
- NNN = Spindle speed (rpm)

---

2. Spindle Speed (N)

N=1000VπDN = \frac{1000V}{\pi D}N=πD1000V

Used to calculate required rpm for a given cutting speed.

---

3. Feed Rate (F)

F=f×NF = f \times NF=f×N

Where:

- FFF = Feed rate (mm/min)
- fff = Feed per revolution (mm/rev)
- NNN = Spindle speed (rpm)

---

4. Material Removal Rate (MRR)

MRR=πDfdNMRR = \pi D f d NMRR=πDfdN

Where:

- DDD = Initial diameter (mm)
- fff = Feed (mm/rev)
- ddd = Depth of cut (mm)
- NNN = Spindle speed (rpm)

---

5. Machining Time (T)

T=Lf×NT = \frac{L}{f \times N}T=f×NL

Where:

- TTT = Machining time (min)
- LLL = Length of cut (mm)
- fff = Feed per revolution (mm/rev)
- NNN = Spindle speed (rpm)

---

6. Depth of Cut (d)

d=D1−D22d = \frac{D_1 - D_2}{2}d=2D1−D2

Where:

- D1D_1D1 = Initial diameter
- D2D_2D2 = Final diameter

---

7. Tool Engagement Ratio (optional advanced)

TER=dDTER = \frac{d}{D}TER=Dd

---

8. Power Requirement (approx.)

P=MRR×K60P = \frac{MRR \times K}{60}P=60MRR×K

Where:

- PPP = Power (kW)
- KKK = Specific cutting force constant

🔄 SEQUENCE OF OPERATION

1. Fix the workpiece in the chuck
2. Mount the cutting tool on the tool post
3. Set tool offsets and zero reference
4. Load CNC program
5. Perform dry run (without cutting)
6. Start spindle and coolant
7. Execute turning operation
8. Measure final dimensions

💻 CNC PROGRAM (Sample G-Code)

```
%
O1001;
G21; (Metric units)
G90; (Absolute programming)
G97 S1200 M03; (Spindle ON clockwise)

G00 X50 Z5; (Rapid move to start position)
G01 Z0 F0.2; (Facing)
G01 X0;

G00 X50 Z5;
G01 Z-40 F0.25; (Turning operation)
```

THINKERING LAB

```
G01 X30;

G00 X100 Z100; (Tool retract)
M05; (Spindle stop)
M30; (End program)
%
```

PROCEDURE

1. Select suitable workpiece material and cutting tool
2. Mount the workpiece securely in the chuck
3. Set tool offsets and work coordinate system
4. Input or upload CNC program into the machine
5. Perform a dry run to check tool path
6. Start the machining process
7. Apply coolant during cutting
8. After completion, stop the machine
9. Measure final dimensions using precision tools
10. Record machining parameters and results

📊 OBSERVATIONS

- Initial diameter of workpiece = ______ mm
- Final diameter after turning = ______ mm
- Length of cut = ______ mm
- Spindle speed = ______ rpm
- Feed rate = ______ mm/rev

Calculated:

- Cutting speed = ______ m/min
- MRR = ______ mm^3/min
- Machining time = ______ min

PRECAUTIONS

- Ensure proper clamping of the workpiece
- Use correct spindle speed and feed rate
- Always perform a dry run before machining
- Maintain proper coolant flow
- Check tool alignment before operation
- Do not touch rotating parts
- Wear safety equipment (gloves, goggles)

✅ **RESULT**

The CNC turning operation was successfully performed. The relationship between machining parameters (cutting speed, feed, and depth of cut) and output responses (surface finish, MRR, machining time) was studied and verified

THINKERING LAB

EXPERIMENT: CNC THREADING OPERATION

🎯 AIM

To perform an external threading operation on a CNC lathe and study the relationship between pitch, spindle speed, and feed rate.

□ APPARATUS / TOOLS REQUIRED

- CNC Turning Machine (e.g., FANUC CNC Lathe or equivalent)
- Workpiece (Mild Steel / Aluminum rod)
- Threading tool (single-point carbide threading insert)
- Tool holder
- Vernier caliper / Screw gauge / Thread pitch gauge
- Coolant system
- Chuck and tailstock
- CNC control panel / programming interface

📘 THEORY

Threading is a machining process used to create **helical grooves** on a cylindrical workpiece. In CNC turning, threading is carried out using synchronized movement between the spindle rotation and tool feed.

The most common CNC threading command is the **G76 threading cycle**, which performs multiple passes automatically to achieve the required thread profile.

Key Parameters:

- **Pitch (P):** Distance between threads
- **Major Diameter:** Outer diameter of thread
- **Minor Diameter:** Root diameter
- **Lead:** Distance tool advances in one revolution (equal to pitch for single-start threads)

⚙ FORMULAS

1. **Feed Rate in Threading:**

Feed=Pitch(P)Feed = Pitch (P)Feed=Pitch(P)

2. **Cutting Speed (V):**

V=πDN1000V = \frac{\pi D N}{1000}V=1000πDN

Where:

- DDD = Diameter (mm)
- NNN = Spindle speed (rpm)

3. **Thread Depth (for metric threads):**

Depth=0.613×PitchDepth = 0.613 \times PitchDepth=0.613×Pitch

4. **Minor Diameter:**

dmin=dmaj−2×Depthd_{min} = d_{maj} - 2 \times Depthdmin=dmaj−2×Depth

🔄 SEQUENCE OF OPERATION

1. Mount the workpiece in the chuck
2. Fix threading tool in tool holder
3. Set tool offsets and zero reference
4. Define thread parameters (pitch, depth, length)
5. Load CNC program
6. Perform dry run
7. Start spindle and coolant
8. Execute threading cycle
9. Inspect thread using pitch gauge

💻 CNC PROGRAM (Sample G-Code using G76)

```
%
O2001;
G21; (Metric units)
G90; (Absolute programming)
G97 S600 M03; (Spindle ON)

G00 X30 Z5; (Start position)

G76 P020060 Q100 R0.05;
G76 X24.5 Z-30 P1200 Q200 F1.5;

G00 X50 Z50;
M05;
M30;
%
```

Explanation:

- **P020060** → Finishing passes, thread angle
- **Q100** → Minimum depth of cut
- **R0.05** → Finishing allowance
- **X24.5** → Minor diameter
- **Z-30** → Thread length

- **P1200** → Thread depth (in microns)
- **F1.5** → Pitch (1.5 mm)

PROCEDURE

1. Select suitable workpiece and threading tool
2. Mount workpiece securely in chuck
3. Set tool offsets carefully (critical for threading)
4. Input CNC threading program
5. Perform dry run without cutting
6. Run machine with coolant ON
7. Allow threading cycle to complete
8. Stop the machine and remove workpiece
9. Check thread using thread gauge and micrometer
10. Record all parameters

-OBSERVATIONS

- Workpiece diameter (initial) = ______ mm
- Major diameter = ______ mm
- Minor diameter = ______ mm
- Pitch = ______ mm
- Spindle speed = ______ rpm

Calculated:

- Cutting speed = ______ m/min
- Thread depth = ______ mm

⚠ PRECAUTIONS

- Ensure proper clamping of the workpiece
- Use correct spindle speed and feed rate
- Always perform a dry run before machining
- Maintain proper coolant flow
- Check tool alignment before operation
- Do not touch rotating parts
- Wear safety equipment (gloves, goggles)

✅ **RESULT**

The CNC threading operation was successfully performed. The required thread profile was obtained, and the relationship between pitch, spindle speed, and feed rate was verified.

THINKERING LAB

EXPERIMENT: CNC DRILLING OPERATION

🎯 AIM

To perform a drilling operation on a CNC machine and study the machining parameters involved.

🎯 OBJECTIVE

- To understand CNC drilling using canned cycles
- To determine cutting speed, feed rate, and material removal rate
- To produce accurate holes with required dimensions
- To study the effect of machining parameters on hole quality

APPARATUS / TOOLS REQUIRED

- CNC Vertical Machining Center (e.g., FANUC CNC Milling Machine)
- Workpiece (Mild Steel / Aluminum block)
- Twist drill (HSS / Carbide)
- Drill chuck or collet holder
- Machine vice / fixture
- Vernier caliper / Depth gauge
- Coolant system
- CNC controller interface

📘 THEORY

Drilling is a machining process used to create **cylindrical holes** using a rotating multi-point cutting tool.

In CNC drilling:

- The drill rotates at high speed (spindle motion)
- The tool advances linearly along the Z-axis
- Canned cycles such as **G81 (simple drilling)** and **G83 (peck drilling)** automate the process

Important Concepts:

- **Cutting speed** depends on spindle speed and drill diameter
- **Feed rate** determines how fast the drill penetrates the material
- **Peck drilling** is used for deep holes to avoid chip clogging

THINKERING LAB

⚙ FORMULAS

1. Cutting Speed

V=πDN1000V = \frac{\pi D N}{1000}V=1000πDN

2. Feed Rate

FR=f×NFR = f \times NFR=f×N

3. Material Removal Rate (MRR)

MRR=πD24×f×NMRR = \frac{\pi D^2}{4} \times f \times NMRR=4πD2×f×N

4. Machining Time

T=Lf×NT = \frac{L}{f \times N}T=f×NL

Where:

- DDD = Drill diameter (mm)
- NNN = Spindle speed (rpm)
- fff = Feed per revolution (mm/rev)
- LLL = Depth of hole (mm)

🔄 SEQUENCE OF OPERATION

1. Clamp the workpiece securely on the machine table
2. Mount the drill tool in the spindle
3. Set work coordinate system (zero reference)
4. Set tool offsets
5. Input CNC program
6. Perform dry run
7. Start spindle and coolant
8. Execute drilling cycle
9. Retract tool after drilling
10. Measure hole dimensions

💻 CNC PROGRAM (Sample G-Code – G81)

```
%
O3001;
G21; (Metric units)
G90; (Absolute programming)
G17; (XY plane)

G00 X25 Y25 Z5; (Position above hole)
```

THINKERING LAB

```
G81 R2 Z-20 F120; (Drilling cycle)
G80; (Cancel cycle)

G00 Z50;
M05;
M30;
%
```

PROCEDURE

1. Select suitable drill size and workpiece material
2. Fix the workpiece properly using a vice or fixture
3. Mount the drill tool and ensure proper alignment
4. Set tool offsets and work zero accurately
5. Enter the CNC drilling program into the machine
6. Perform a dry run without cutting
7. Switch ON coolant supply
8. Run the program and observe drilling operation
9. Stop the machine after completion
10. Measure hole diameter and depth
11. Record observations

📊 OBSERVATIONS

- Drill diameter = ______ mm
- Hole depth = ______ mm
- Spindle speed = ______ rpm
- Feed rate = ______ mm/min

Calculated Values:

- Cutting speed = ______ m/min
- MRR = ______ mm^3/min
- Machining time = ______ min

⚠ PRECAUTIONS

- Ensure proper clamping of the workpiece
- Use correct spindle speed and feed rate
- Always perform a dry run before machining
- Maintain proper coolant flow
- Check tool alignment before operation
- Do not touch rotating parts
- Wear safety equipment (gloves, goggles)

✓ RESULT

The CNC drilling operation was successfully carried out. The hole was produced with acceptable accuracy, and the machining parameters such as cutting speed, feed rate, and MRR were studied and verified.

THINKERING LAB

EXPERIMENT: 3D PRINTING OF A CUBE

🎯 AIM

To fabricate a cube using a 3D printing machine and study the process parameters involved in additive manufacturing.

🎯 OBJECTIVE

- To understand the working principle of 3D printing (FDM process)
- To generate and execute G-code for a simple geometry
- To study the influence of printing parameters on accuracy and surface finish
- To measure dimensional accuracy of the printed cube

APPARATUS / TOOLS REQUIRED

- 3D Printer (e.g., Creality Ender 3 or similar FDM printer)
- Filament material (PLA / ABS)
- Computer with slicing software (e.g., Ultimaker Cura)
- CAD model of cube (e.g., 20 mm × 20 mm × 20 mm)
- Vernier caliper
- Build plate (heated bed)
- USB/SD card for file transfer

📘 THEORY

3D printing is an **additive manufacturing process** in which objects are created layer by layer from a digital model.

In **Fused Deposition Modeling (FDM)**:

- Thermoplastic filament is heated and extruded through a nozzle
- Material is deposited layer by layer along predefined paths
- Each layer solidifies to form a 3D object

Key Parameters:

- **Layer height** (affects resolution)
- **Infill density** (affects strength)
- **Print speed**
- **Nozzle temperature**
- **Bed temperature**

⚙ FORMULAS

1. **Print Time Estimation (approx.):**

T=Total path lengthPrint speedT = \frac{\text{Total path length}}{\text{Print speed}}T=Print speedTotal path length

2. **Material Volume of Cube:**

V=a3V = a^3V=a3

Where aaa = side of cube

3. **Material Consumption (with infill):**

Vactual=V×Infill fractionV_{actual} = V \times \text{Infill fraction}Vactual=V×Infill fraction

4. **Dimensional Error:**

Error=Measured value−Actual value\text{Error} = \text{Measured value} - \text{Actual value}Error=Measured value−Actual value

🔄 SEQUENCE OF OPERATION

1. Create cube model using CAD software
2. Import model into slicing software
3. Set printing parameters (layer height, infill, speed)
4. Generate G-code
5. Transfer G-code to printer
6. Load filament into printer
7. Level the build plate
8. Start printing process
9. Allow print to complete
10. Remove printed cube and inspect

💻 SAMPLE G-CODE (Simplified)

```
; Start G-code
G21 ; Set units to mm
G90 ; Absolute positioning
M104 S200 ; Set nozzle temp
M140 S60 ; Set bed temp
M109 S200 ; Wait for nozzle temp
M190 S60 ; Wait for bed temp

G28 ; Home all axes
G1 Z0.2 F3000

; Begin printing layers
G1 X10 Y10 F1500
G1 Z0.2
G1 X30 Y10 E1
G1 X30 Y30 E2
```

```
G1 X10 Y30 E3
G1 X10 Y10 E4

; (Layer-by-layer continuation...)

M104 S0 ; Turn off nozzle
M140 S0 ; Turn off bed
M84 ; Disable motors
```

(Note: Actual slicer-generated code will be much longer and optimized.)

PROCEDURE

1. Design a cube (e.g., 20×20×20 mm) using CAD software
2. Export the model as STL file
3. Open STL file in slicing software (Ultimaker Cura)
4. Set parameters:
   - Layer height (e.g., 0.2 mm)
   - Infill (e.g., 20%)
   - Print speed (e.g., 50 mm/s)
5. Generate G-code
6. Transfer file to 3D printer
7. Load filament and preheat nozzle
8. Level the bed properly
9. Start the print
10. Observe printing layer by layer
11. After completion, remove the cube carefully
12. Measure dimensions using vernier caliper
13. Record observations

📊 OBSERVATIONS

- Designed cube size = ______ mm
- Measured length = ______ mm
- Measured width = ______ mm
- Measured height = ______ mm
- Layer height = ______ mm
- Infill percentage = ______ %
- Printing time = ______ min

Calculated:

- Volume of cube = ______ mm^3
- Dimensional error = ______ mm

⚠ PRECAUTIONS

- Ensure proper bed leveling before printing
- Use correct temperature settings for filament
- Avoid touching the nozzle during operation (very hot)
- Ensure proper adhesion to build plate
- Do not interrupt printing midway
- Keep printer in a stable environment
- Use proper ventilation for ABS printing

✅ RESULT

The cube was successfully fabricated using a 3D printing machine. The dimensional accuracy and surface finish were evaluated, and the effect of process parameters such as layer height and infill density was studied.

THINKERING LAB

EXPERIMENT: 3D PRINTING OF A BOLT

🎯 AIM

To design and fabricate a bolt using a 3D printing machine and evaluate dimensional accuracy and thread quality.

🎯 OBJECTIVE

- To understand additive manufacturing for threaded components
- To generate and execute G-code for a bolt model
- To study the effect of printing parameters on thread formation
- To compare printed dimensions with standard bolt dimensions

□ APPARATUS / TOOLS REQUIRED

- 3D Printer (e.g., Creality Ender 3 or similar FDM printer)
- Filament (PLA / ABS)
- CAD software (e.g., Fusion 360 or SolidWorks)
- Slicing software (e.g., Ultimaker Cura)
- Vernier caliper / Thread pitch gauge
- STL file of bolt
- Build plate and nozzle assembly

📕 THEORY

3D printing of a bolt is an application of **additive manufacturing**, where a threaded component is built layer by layer.

In **Fused Deposition Modeling (FDM)**:

- Molten filament is extruded through a nozzle
- Layers are deposited in a helical pattern to form threads
- Thread quality depends on layer resolution and printer precision

Important Concepts:

- **Pitch** → Distance between threads
- **Major Diameter** → Outer diameter of bolt
- **Minor Diameter** → Core diameter
- **Tolerance** → Important for fitting with nuts

THINKERING LAB

⚙ FORMULAS

1. **Volume of Bolt (approx.):**

V=πd24×LV = \pi \frac{d^2}{4} \times LV=π4d2×L

2. **Thread Depth (metric):**

Depth=0.613×PitchDepth = 0.613 \times PitchDepth=0.613×Pitch

3. **Material Usage:**

Vactual=V×Infill fractionV_{actual} = V \times \text{Infill fraction}Vactual=V×Infill fraction

4. **Dimensional Error:**

Error=Measured−Designed\text{Error} = \text{Measured} - \text{Designed}Error=Measured−Designed

🔄 SEQUENCE OF OPERATION

1. Design bolt model with threads in CAD software
2. Export model as STL file
3. Import STL into slicing software
4. Set printing parameters (layer height, infill, speed)
5. Generate G-code
6. Transfer file to 3D printer
7. Load filament and preheat nozzle
8. Level the bed
9. Start printing
10. Remove and inspect printed bolt

💻 SAMPLE G-CODE (Simplified)

```
; Start G-code
G21 ; Units in mm
G90 ; Absolute positioning
M104 S200 ; Nozzle temp
M140 S60 ; Bed temp
M109 S200
M190 S60

G28 ; Home axes
G1 Z0.2 F3000

; Printing base
G1 X10 Y10 F1500
G1 X30 Y10 E1
G1 X30 Y30 E2
```

```
G1 X10 Y30 E3

; (Thread layers generated by slicer)

M104 S0
M140 S0
M84
```

(Note: Actual bolt G-code from slicer will include detailed helical paths for threads.)

PROCEDURE

1. Create a bolt model (e.g., M10 × 50 mm) using CAD software (Fusion 360)
2. Apply thread feature with correct pitch and dimensions
3. Export model as STL file
4. Open STL in slicing software (Ultimaker Cura)
5. Set parameters:
   - Layer height (e.g., 0.1–0.2 mm for better threads)
   - Infill (30–50%)
   - Print speed (40–50 mm/s)
6. Generate G-code
7. Transfer file to printer
8. Load filament and level the bed
9. Start printing and monitor progress
10. After printing, remove bolt carefully
11. Measure dimensions and check thread fit
12. Record observations

📊 **OBSERVATIONS**

- Designed major diameter = ______ mm
- Measured diameter = ______ mm
- Pitch = ______ mm
- Length of bolt = ______ mm
- Layer height = ______ mm
- Printing time = ______ min

Calculated:

- Dimensional error = ______ mm
- Volume of bolt = ______ mm^3

⚠ **PRECAUTIONS**

- Use small layer height for better thread accuracy
- Ensure proper bed leveling
- Maintain correct nozzle temperature
- Avoid high print speed (affects threads)

- Ensure proper cooling for fine details
- Do not force-fit nut (may damage threads)
- Keep nozzle clean to avoid defects

✅ **RESULT**

The bolt was successfully fabricated using a 3D printing machine. The thread profile and dimensions were evaluated, and the influence of process parameters on print quality and accuracy was observed.

THINKERING LAB

EXPERIMENT: 3D PRINTING OF A SQUARE PYRAMID

🎯 AIM

To design and fabricate a square pyramid using a 3D printing machine and study dimensional accuracy and surface characteristics.

🎯 OBJECTIVE

- To understand the FDM-based 3D printing process
- To design a square pyramid using CAD software
- To generate and execute G-code
- To analyze dimensional accuracy and surface finish

APPARATUS / TOOLS REQUIRED

- 3D Printer (e.g., Creality Ender 3)
- Filament material (PLA / ABS)
- CAD software (e.g., Fusion 360)
- Slicing software (e.g., Ultimaker Cura)
- Vernier caliper
- STL file
- Build plate and nozzle assembly

📘 THEORY

3D printing is an **additive manufacturing process** where objects are built layer by layer from a digital model.

In **Fused Deposition Modeling (FDM)**:

- Filament is melted and extruded through a heated nozzle
- Material is deposited layer-by-layer
- Each layer adheres to the previous one

For a **square pyramid**:

- The base is square
- Each layer reduces in size progressively
- Inclined faces are approximated using thin layers (staircase effect)

Key Parameters:

- Layer height
- Print speed
- Nozzle temperature

- Infill density

⚙ DESIGN PROCEDURE

1. Open CAD software (Fusion 360)
2. Create a square sketch (base side = *a*)
3. Draw a vertical axis and define pyramid height (*h*)
4. Use “Loft” or “Extrude with taper” feature to form pyramid
5. Ensure all faces converge to a single apex point
6. Apply dimensions (e.g., base = 30 mm, height = 40 mm)
7. Check model for errors (solid body)
8. Export file as STL format

⚙ FORMULAS

1. **Volume of Square Pyramid:**

V=13a2hV = \frac{1}{3} a^2 hV=31a2h

2. **Slant Height (l):**

l=(a2)2+h2l = \sqrt{\left(\frac{a}{2}\right)^2 + h^2}l=(2a)2+h2

3. **Surface Area:**

A=a2+2alA = a^2 + 2 a lA=a2+2al

4. **Material Consumption:**

Vactual=V×Infill fractionV_{actual} = V \times \text{Infill fraction}Vactual=V×Infill fraction

5. **Dimensional Error:**

Error=Measured−Designed\text{Error} = \text{Measured} - \text{Designed}Error=Measured−Designed

🔄 SEQUENCE OF OPERATION

1. Design the square pyramid in CAD
2. Export model as STL file
3. Import STL into slicing software
4. Set printing parameters
5. Generate G-code
6. Transfer file to printer
7. Load filament and preheat nozzle
8. Level build plate
9. Start printing process
10. Remove printed model and inspect

💻 SAMPLE G-CODE (Simplified)

```
; Start G-code
G21 ; Units in mm
G90 ; Absolute positioning
M104 S200 ; Nozzle temperature
M140 S60 ; Bed temperature
M109 S200
M190 S60

G28 ; Home axes
G1 Z0.2 F3000

; Base layer
G1 X10 Y10 F1500
G1 X40 Y10 E1
G1 X40 Y40 E2
G1 X10 Y40 E3
G1 X10 Y10 E4

; Gradual reduction layers (handled by slicer)

M104 S0
M140 S0
M84
```

PROCEDURE

1. Create pyramid model using CAD (Fusion 360)
2. Export the design as STL file
3. Open STL in slicing software (Ultimaker Cura)
4. Set parameters:
   - Layer height: 0.1–0.2 mm
   - Infill: 15–25%
   - Print speed: 40–60 mm/s
5. Generate G-code
6. Transfer file to the printer
7. Load filament and set temperatures
8. Level the bed
9. Start printing and monitor layers
10. After completion, allow cooling
11. Remove printed pyramid
12. Measure dimensions and record

📊 OBSERVATIONS

- Designed base side = ______ mm
- Measured base side = ______ mm
- Designed height = ______ mm
- Measured height = ______ mm

- Layer height = ______ mm
- Print time = ______ min

Calculated:

- Volume = ______ mm^3
- Slant height = ______ mm
- Dimensional error = ______ mm

⚠ PRECAUTIONS

- Ensure proper bed leveling before printing
- Use smaller layer height for smooth inclined faces
- Avoid excessive print speed
- Maintain correct temperature settings
- Ensure proper adhesion to build plate
- Do not disturb printer during operation
- Allow model to cool before removal

✅ RESULT

The square pyramid was successfully designed and fabricated using a 3D printer. The dimensional accuracy and surface quality were evaluated, and the influence of process parameters on inclined surfaces was observed.

THINKERING LAB

EXPERIMENT: SMART DUSTBIN (AUTOMATIC LID OPENING SYSTEM)

🎯 AIM

To design and develop a smart dustbin that automatically opens and closes its lid using a sensor-based system.

🎯 OBJECTIVE

- To understand automation using sensors and microcontrollers
- To implement a contactless waste disposal system
- To study the working of ultrasonic sensors and servo motors
- To improve hygiene using embedded systems

APPARATUS / TOOLS REQUIRED

- Microcontroller (e.g., Arduino Uno)
- Ultrasonic sensor (HC-SR04)
- Servo motor (SG90 or equivalent)
- Dustbin with lid
- Jumper wires
- Breadboard
- Power supply (battery/adapter)
- Connecting wires and mounting accessories

📘 THEORY

A smart dustbin uses **sensor-based automation** to detect the presence of an object (like a hand) and open the lid automatically.

Working Principle:

- The ultrasonic sensor emits sound waves and measures the time taken for the echo to return
- Distance is calculated using the **Speed of Sound principle
- If the detected distance is below a threshold, the microcontroller activates the servo motor
- The servo rotates to open the lid and closes after a delay

⚙ DESIGN PROCEDURE

1. Select suitable dustbin and components
2. Mount the ultrasonic sensor on the front/top of the dustbin
3. Attach the servo motor to the lid using a linkage mechanism
4. Connect components to Arduino Uno
5. Write and upload control program

6. Set threshold distance (e.g., 15 cm)
7. Test opening and closing mechanism
8. Adjust angles and timing for smooth operation

⚙ FORMULAS

1. **Distance Measurement:**

Distance=Speed×Time2Distance = \frac{Speed \times Time}{2}Distance=2Speed×Time

Where:

- Speed of sound ≈ 343 m/s

2. **Time Calculation:**

Time=2×DistanceSpeedTime = \frac{2 \times Distance}{Speed}Time=Speed2×Distance

3. **Servo Angle Control:**

- Open position ≈ 90°
- Closed position ≈ 0°

🔄 SEQUENCE OF OPERATION

1. Power ON the system
2. Ultrasonic sensor emits signal
3. Object (hand) reflects the signal
4. Sensor calculates distance
5. If distance < threshold:
   - Servo rotates → lid opens
6. After delay:
   - Servo returns → lid closes

💻 PROGRAM CODE (Arduino

```
#include <Servo.h>

Servo lidServo;

int trigPin = 9;
int echoPin = 10;
int servoPin = 6;

long duration;
int distance;

void setup() {
```

```
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  lidServo.attach(servoPin);
  lidServo.write(0); // Lid closed
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  Serial.println(distance);

  if (distance < 15) {
    lidServo.write(90); // Open lid
    delay(3000);
    lidServo.write(0); // Close lid
  }

  delay(500);
}
```

PROCEDURE

1. Assemble all components as per circuit design
2. Connect ultrasonic sensor and servo to Arduino Uno
3. Upload the program to Arduino
4. Power the system
5. Place hand near sensor
6. Observe lid opening and closing
7. Adjust threshold distance if needed
8. Record observations

OBSERVATIONS

- Threshold distance set = ______ cm
- Response time = ______ sec
- Servo angle (open) = ______°
- Servo angle (close) = ______°

Performance:

- Detection accuracy = ______
- Smoothness of operation = ______

⚠ PRECAUTIONS

- Ensure proper wiring connections
- Do not expose electronics to moisture
- Use stable power supply
- Mount servo firmly to avoid misalignment
- Avoid obstacles blocking sensor
- Calibrate sensor distance properly

✅ RESULT

The smart dustbin was successfully designed and implemented. The system automatically opened and closed the lid based on object detection, demonstrating effective use of sensors and microcontroller-based automation.

THINKERING LAB

EXPERIMENT: SMALL WALKING ROBOT

🎯 AIM

To design and develop a small walking robot using servo motors and a microcontroller and study its basic locomotion mechanism.

🎯 OBJECTIVE

- To understand robotic locomotion and gait mechanisms
- To design and implement a basic walking robot (biped/quadruped)
- To control servo motors using a microcontroller
- To analyze stability and motion of the robot

APPARATUS / TOOLS REQUIRED

- Microcontroller (e.g., Arduino Uno)
- Servo motors (SG90 or MG995 – 4 to 8 units)
- Robot chassis (acrylic / 3D printed frame)
- Battery pack (5V–6V supply)
- Jumper wires
- Breadboard (optional)
- Screws, nuts, and mechanical linkages
- Computer with Arduino IDE

📘 THEORY

A walking robot mimics human/animal locomotion using coordinated joint movements.

Types of walking robots:

- Biped (2 legs)
- Quadruped (4 legs – more stable)
- Hexapod (6 legs – highly stable)

Working Principle:

- Servo motors control leg joints (hip/knee)
- Microcontroller sends PWM signals to control angle
- Coordinated movement produces walking motion

The robot's stability depends on its **center of gravity** and **gait pattern**.

⚙ DESIGN PROCEDURE

1. Select robot type (biped or quadruped)

2. Design chassis using lightweight material
3. Attach servo motors to joints
4. Fix legs using mechanical linkages
5. Connect all servos to Arduino Uno
6. Provide external power supply to servos
7. Program walking sequence
8. Test and adjust servo angles
9. Optimize gait for balance and smooth motion

🔌 Circuit Setup

- Connect servo motors to PWM pins on Arduino
- Use **external power supply** for servos (important!)
- Common ground between Arduino and battery

Basic layout:

```
Battery → Servo Power
Arduino → Control Signals → Servos
```

Control Logic

Start simple:

- Move legs in sequence (predefined gait)
- Example:
  - Lift leg → move forward → place down
  - Repeat in cycle

You can program:

- Crawl gait (stable, slow)
- Trot gait (faster, less stable)

Experimental Variables

Test different parameters:

- Step length
- Speed of servo movement
- Timing between legs
- Weight distribution

Measure:

- Distance traveled

- Stability (falls or not)
- Energy usage

Data Collection

You can:

- Record video and analyze motion
- Log sensor data via Arduino serial monitor
- Measure time vs distance

Test Environment

- Flat surface (start here)
- Slight incline (advanced test)
- Obstacle course (later stage)

Iteration Plan

1. Build basic walking motion
2. Fix balance issues
3. Add sensors
4. Improve gait efficiency

⚠ Common Mistakes to Avoid

- Powering servos directly from Arduino (causes resets)
- Using too heavy materials
- Not calibrating servo angles
- Trying complex walking before basic movement works

🔄 SEQUENCE OF OPERATION

1. Power ON the robot
2. Microcontroller initializes servo positions
3. Servo motors move legs in sequence
4. One leg lifts while others support body
5. Robot shifts center of gravity forward
6. Cycle repeats to achieve walking motion
7. Adjust gait for stability

💻 PROGRAM CODE (Arduino)

```
#include <Servo.h>

Servo leg1;
Servo leg2;
```

```
Servo leg3;
Servo leg4;

void setup() {
  leg1.attach(3);
  leg2.attach(5);
  leg3.attach(6);
  leg4.attach(9);
}

void loop() {
  // Step 1: Move forward left side
  leg1.write(60);
  leg3.write(120);
  delay(500);

  // Step 2: Move forward right side
  leg2.write(60);
  leg4.write(120);
  delay(500);

  // Reset position
  leg1.write(90);
  leg2.write(90);
  leg3.write(90);
  leg4.write(90);
  delay(500);
}
```

□ PROCEDURE

1. Assemble robot frame and attach servo motors
2. Connect servos to Arduino Uno
3. Provide external power supply for servos
4. Upload Arduino code
5. Place robot on flat surface
6. Switch ON power supply
7. Observe walking motion
8. Adjust servo angles for better stability
9. Test on different surfaces

📊 OBSERVATIONS

- Type of gait used = _______
- Step length = _______ cm
- Walking speed = _______ cm/s
- Servo angle range = _______°

Performance:

- Stability: Good / Average / Poor

- Motion smoothness: ______
- Balance efficiency: ______

⚠ PRECAUTIONS

- Ensure proper servo alignment before powering ON
- Use external power supply for servos
- Avoid uneven surfaces during testing
- Do not overload robot frame
- Secure all mechanical joints tightly
- Calibrate servo angles before operation

✓ RESULT

The small walking robot was successfully designed and operated. The robot demonstrated basic locomotion using coordinated servo control, and the influence of gait and balance on stability was studied.

THINKERING LAB

EXPERIMENT: AI CONTROLLED STREET LIGHT OPERATOR

🎯 AIM

To design and implement an AI-based street light system that automatically controls street lights based on environmental conditions and human/vehicle presence to improve energy efficiency.

PARTS (APPARATUS)

- Microcontroller (Arduino Uno / ESP32)
- LDR (Light Dependent Resistor)
- PIR motion sensor
- LED street light module
- Relay module
- Power supply (5V/12V)
- Breadboard and jumper wires
- Optional AI module (ESP32-CAM / edge AI device)

⚙ DESIGN

The system is designed as a **smart lighting network** using sensors and AI logic.

Design Components:

- **LDR sensor** → Detects day/night conditions
- **PIR sensor** → Detects motion of humans/vehicles
- **Microcontroller** → Processes inputs
- **Relay module** → Switches street lights ON/OFF
- **LED lights** → Simulated street lights

System Layout:

- Sensors placed on roadside poles
- Controller connected centrally
- Lights distributed along road section

🔄 WORKING

- During daytime, LDR detects high light intensity → lights OFF
- At night, LDR activates system
- PIR sensor detects motion (vehicle/person)
- AI logic predicts movement pattern and activates lights only in required zones
- Lights turn OFF automatically after no motion is detected

AI Functionality:

- Predicts traffic flow
- Adjusts brightness dynamically
- Reduces energy consumption

PROCEDURE

1. Assemble circuit using LDR, PIR sensor, and Arduino Uno
2. Connect LED street light module through relay
3. Upload control program
4. Place LDR in ambient light location
5. Install PIR sensor facing motion path
6. Power ON system
7. Test system during day and night conditions
8. Observe automatic switching behavior
9. Adjust sensitivity settings if required

💻 PROGRAM (Arduino Code)

```
int ldrPin = A0;
int pirPin = 2;
int relayPin = 8;

int ldrValue = 0;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(relayPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  ldrValue = analogRead(ldrPin);
  int motion = digitalRead(pirPin);

  Serial.println(ldrValue);

  // Night condition
  if (ldrValue < 500) {
    if (motion == HIGH) {
      digitalWrite(relayPin, HIGH); // Light ON
    } else {
      digitalWrite(relayPin, LOW);  // Light OFF
    }
  } else {
    digitalWrite(relayPin, LOW); // Daytime OFF
  }

  delay(500);
}
```

THINKERING LAB

📊 PERFORMANCE

- Detection accuracy: High / Medium / Low
- Response time: ~1–2 seconds
- Power saving efficiency: Up to 60–80%
- Sensor reliability: Stable under normal conditions

👁 OBSERVATIONS

- Light turns OFF in daylight automatically
- Light turns ON when motion is detected at night
- System responds within seconds
- Energy usage reduced compared to normal street lights

🌍 APPLICABILITY

- Smart cities
- Highway lighting systems
- Residential streets
- Parking areas
- Industrial zones
- Campus lighting systems

🔢 TYPES OF SMART STREET LIGHT SYSTEMS

1. LDR-based automatic street light
2. Motion sensor-based system
3. AI prediction-based adaptive lighting
4. IoT cloud-controlled street lighting
5. Hybrid (LDR + PIR + AI system)

📌 METHOD

- Sensor-based detection (LDR + PIR)
- Microcontroller decision-making
- Relay switching control
- AI-based prediction (optional enhancement)
- Energy optimization algorithm

📊 RESULTS

- System successfully controlled street lights automatically
- Lights operated only when required
- Significant energy savings achieved
- Improved efficiency compared to conventional lighting systems
- Real-time response to environmental changes

⚠ CONCLUSION

The AI-controlled street light operator was successfully designed and implemented. The system demonstrated automatic switching, motion detection, and energy-efficient lighting control, making it suitable for smart city applications.

PROCESS OF OPERATING CNC AND DNC MACHINES

🖥 1. CNC MACHINE OPERATION (Computer Numerical Control)

🔄 Operating Process

1. Job Design

- Component is designed using CAD software (e.g., Fusion 360)
- Final geometry is created digitally

2. Program Generation

- CAD model is converted into G-code using CAM software
- Tool paths, feed, speed, and depth are defined

3. Machine Setup

- Workpiece is clamped on machine bed/chuck
- Cutting tools are installed in turret/tool holder
- Tool offsets and work coordinates are set

4. Program Input

- G-code is loaded into CNC controller via USB, memory card, or network

5. Dry Run

- Machine is run without cutting to check tool path safety

6. Machining Operation

- Spindle starts rotating
- Tool follows programmed path
- Material is removed layer by layer

7. Inspection

- Finished part is measured using precision tools
- Corrections are made if required

🌐 2. DNC SYSTEM OPERATION (Direct Numerical Control)

🔄 Operating Process

1. Central Computer Programming

- Programs are stored in a central server or host computer

2. Data Transfer

- CNC machines receive instructions directly from the central system via network (LAN/RS-232)

3. Machine Execution

- Each CNC machine executes program in real-time or buffered mode

4. Monitoring

- Operator monitors multiple machines from central station

5. Updates & Control

- Programs can be edited and updated centrally
- Reduces manual file transfer errors

🏭 APPLICATIONS

🔧 CNC Applications

- Automotive components (shafts, gears)
- Aerospace parts
- Mold and die making
- Medical implants
- Precision machining

🌐 DNC Applications

- Large manufacturing industries
- Production lines with multiple CNC machines
- Aerospace manufacturing plants
- Automated factories (smart manufacturing systems)

⋆ ADVANTAGES

✓ CNC Advantages

- High accuracy and precision
- Reduced human error
- Complex shapes can be produced
- High production speed
- Repeatability of parts

✓ DNC Advantages

- Centralized control of multiple machines
- Easy program management and updates
- Reduced storage requirements in machines
- Improved production efficiency
- Real-time monitoring

🚀 FUTURE DEVELOPMENT PATHS

1. AI-Integrated CNC Systems

- Self-optimizing tool paths
- Predictive maintenance using machine learning
- Automatic parameter selection

🌐 2. IoT-Based Smart Manufacturing

- Machines connected to cloud systems
- Real-time production monitoring
- Remote control via mobile apps

3. Digital Twin Technology

- Virtual replica of CNC machines
- Simulation before actual machining
- Error prediction and correction

⚡ 4. Fully Autonomous Manufacturing

- Minimal human intervention
- Robots handling loading/unloading
- Lights-out manufacturing plants

⚮ 5. Industry 4.0 Integration

- Smart factories
- Connected supply chains
- Data-driven production optimization

□ SUMMARY

- CNC machines operate using programmed instructions executed locally
- DNC systems control multiple CNC machines from a central computer
- Both are widely used in precision manufacturing industries
- Future trends focus on **AI, IoT, automation, and smart factories**

THINKERING LAB

RESPONSE OF EVs AND EXISTING FUEL VEHICLES (AUTOMOTIVES)

🚗 1. RESPONSE OF EXISTING FUEL VEHICLES (ICE VEHICLES)

🔥 Working Response

Internal Combustion Engine (ICE) vehicles run on petrol, diesel, or CNG. Their response includes:

- Fuel combustion generates mechanical power
- Engine response depends on throttle input and fuel-air mixture
- Power delivery is gradual due to mechanical and thermal processes

⚙ Characteristics

- Slower torque response compared to EVs
- Gear shifting required for performance control
- Engine noise and vibration present
- Efficiency limited (~20–30%)

🌍 Environmental Response

- High CO_2 emissions
- Air pollution (NOx, CO, particulate matter)
- Heat loss through exhaust

⚡ 2. RESPONSE OF ELECTRIC VEHICLES (EVs)

🔋 Working Response

EVs use electric motors powered by batteries:

- Instant torque delivery when accelerator is pressed
- Direct conversion of electrical energy to mechanical motion
- Controlled by power electronics (inverters, controllers)

⚙ Characteristics

- Instant acceleration (0 delay torque response)
- Smooth and silent operation
- No gear shifting in most EVs
- High efficiency (~85–95%)

🌍 Environmental Response

- Zero tailpipe emissions
- Reduced air pollution
- Lower noise pollution

🔄 COMPARISON: EV vs ICE VEHICLES

Feature	ICE Vehicles	EVs
Energy Source	Fuel (petrol/diesel)	Electricity
Torque Response	Slow	Instant
Efficiency	Low (20–30%)	High (85–95%)
Emissions	High	Zero
Maintenance	High	Low
Noise	High	Very low

🚀 FUTURE DEVELOPMENT OF EVs

🔋 1. Battery Technology Improvement

- Solid-state batteries
- Faster charging (5–10 minutes charging goal)
- Higher energy density and longer range

⚡ 2. Fast Charging Infrastructure

- Ultra-fast charging stations
- Wireless charging roads
- Smart grid integration

3. AI and Autonomous EVs

- Self-driving electric cars
- AI-based traffic optimization
- Predictive maintenance systems

🌐 4. Vehicle-to-Grid (V2G) Technology

- EVs supply electricity back to grid
- Energy storage during low demand
- Grid stabilization support

5. Smart Mobility Systems

- Connected EV ecosystems
- App-based vehicle control
- Real-time navigation optimization

🏭 APPLICATIONS OF EVs

🚗 Transportation Sector

- Personal cars
- Electric buses
- Two-wheelers and scooters

🚚 Commercial Use

- Delivery vehicles
- Logistics fleets
- Last-mile transportation

🏙 Urban Mobility

- Smart city transport systems
- Ride-sharing EV fleets

✈ Specialized Applications

- Airport vehicles
- Industrial transport systems
- Autonomous robots and carts

📌 SUMMARY

- ICE vehicles rely on fuel combustion with slower response and higher emissions
- EVs provide instant torque, high efficiency, and zero emissions
- EV technology is rapidly advancing toward **AI integration, fast charging, and autonomous driving**
- EVs are becoming essential for **smart cities and sustainable transportation**

FUTURE ENERGY AND INDUSTRIAL DEVELOPMENT (TOWARDS A BETTER PATH)

🎯 INTRODUCTION

Future development in energy and industry is focused on creating a **sustainable, clean, efficient, and technology-driven system**. The goal is to reduce environmental damage while increasing productivity and energy security.

⚡ 1. FUTURE ENERGY DEVELOPMENT

🔋 A. Renewable Energy Expansion

Future energy systems will rely heavily on:

- Solar energy (photovoltaic and thermal)
- Wind energy (onshore and offshore farms)
- Hydropower and tidal energy
- Biomass energy

👉 These reduce dependence on fossil fuels and lower pollution.

🔥 B. Hydrogen Energy

- Green hydrogen produced using electrolysis
- Clean fuel for industries and transport
- Zero carbon emission energy carrier

⚡ C. Energy Storage Systems

- Advanced lithium-ion and solid-state batteries
- Grid-scale energy storage
- Pumped hydro storage systems

👉 Helps manage renewable energy fluctuations.

🌐 D. Smart Energy Grids

- AI-based electricity distribution
- Real-time demand management
- Integration of renewable sources

☀ E. Decentralized Energy Systems

- Rooftop solar systems
- Microgrids in villages and industries
- Local energy generation and consumption

🏭 2. FUTURE INDUSTRIAL DEVELOPMENT

A. Industry 4.0 and Smart Factories

- Automation using robotics
- AI-based production control
- IoT-enabled machines

👉 Improves efficiency and reduces human error.

B. Artificial Intelligence in Industry

- Predictive maintenance
- Quality control automation
- Supply chain optimization

🌐 C. Digital Twin Technology

- Virtual replica of factories and machines
- Simulation before real production
- Reduces cost and errors

⚙ D. Advanced Manufacturing

- 3D printing (additive manufacturing)
- CNC automation systems
- Nano-manufacturing techniques

🌱 E. Green Industrialization

- Zero-emission factories
- Waste recycling systems
- Carbon-neutral production processes

🌍 3. SUSTAINABLE DEVELOPMENT PATH

♻ Key Principles:

- Reduce carbon emissions
- Reuse industrial waste
- Recycle materials efficiently
- Use renewable energy sources

🚗 4. ROLE OF TRANSPORT IN FUTURE DEVELOPMENT

- Electric vehicles replacing fossil fuel vehicles
- Hydrogen-powered transport systems
- Smart traffic and logistics systems

🚀 5. FUTURE TECHNOLOGIES DRIVING DEVELOPMENT

- Artificial Intelligence (AI)
- Internet of Things (IoT)
- Machine Learning systems
- Robotics and automation
- Quantum computing (future stage)

📊 6. BENEFITS OF FUTURE ENERGY AND INDUSTRIAL DEVELOPMENT

- Reduced pollution and climate change
- High energy efficiency
- Economic growth and job creation
- Sustainable resource utilization
- Improved quality of life

⚠ 7. CHALLENGES

- High initial investment
- Technology adaptation issues
- Energy storage limitations
- Skilled workforce requirement
- Infrastructure development needs

📌 CONCLUSION

The future of energy and industrial development is moving towards a **clean, smart, and sustainable ecosystem**. Integration of renewable energy, AI, automation, and digital technologies will create a balanced system that supports economic growth while protecting the environment.