1) **ADC Reading Project with LEDs and LCD Display**

Project Concept

This project reads an **analog signal** from a potentiometer using an **ATmega16 microcontroller**, converts it to a **digital value (ADC)**, and displays it in two ways:

1. **LED bar** showing the voltage level.
2. **LCD screen** showing the numeric ADC value and its percentage.

2) **Components**

- **ATmega16 Microcontroller**
- **1 kΩ Potentiometer**
- **8 LEDs** connected to PORTB
- **16×2 LCD**
- 220 Ω resistors for LEDs
- Jumper wires and 5 V power supply

3) **Basic Circuit Connections**

Potentiometer:

- One end → VCC (5 V)
- Other end → GND
- Wiper (middle pin) → ADC0 input of ATmega16

LEDs:

- PORTB configured as output → each LED with 220 Ω resistor → GND

LCD:

- Data and control pins connected according to your library (4-bit or 8-bit mode)
- VCC/GND and contrast pin (V0) connected via a small resistor or potentiometer

4) Program Logic

1. **Initialize peripherals:**
    - Configure PORTB as output for LEDs
    - Initialize ADC for analog readings
    - Initialize LCD
2. **Read ADC:**
    - Read channel 0 → value range: 0–1023
3. **Calculate percentage:**
    - Use a safe formula to avoid overflow:
4. percentage = ((uint32_t)adc_value * 100 + 511) / 1023;
5. **Display on LEDs:**
    - Convert ADC value to the number of lit LEDs (0–8):
6. void LED_Display(uint16_t adc_value) {
7. uint8_t num_leds = (adc_value*8)/1023;
8. if(num_leds>8) num_leds=8;
9. PORTB = 0x00;
10. for(uint8_t i=0;i<num_leds;i++)
11. PORTB |= (1<<i);
12. }
13. **Display on LCD:**
    - First line → numeric ADC value
    - Second line → percentage
14. **Continuous update:**
    - Readings update every 300 ms using _delay_ms(300);

5) Notes

- Add a **100 nF capacitor** between ADC input and GND to filter noise if readings fluctuate.
- Ensure **Vref = 5 V** so maximum ADC value (1023) corresponds to 100%.
- The updated calculation formula prevents sudden drops in percentage at high ADC values.