**Project Idea: Distributed Smart Energy Metering and Monitoring System**

**Objective:**

Design a system that monitors and manages power consumption across different parts of a building or lab. Each Arduino will handle the power usage of a specific zone and send data to a central unit that aggregates information and provides alerts when usage crosses certain thresholds.

**Components and Equipment Needed:**

1. **4 Arduino Boards** (Arduino Uno or any similar board)
2. **Current and Voltage Sensors** (e.g., ACS712 for current, ZMPT101B for voltage)
3. **LCD or OLED Display** (for displaying aggregate data on the central unit)
4. **Push Buttons** (for reset functions and input in each zone)
5. **LED Indicators** (to show power status and threshold alerts)
6. **EEPROM Storage** (on the Arduinos for storing usage data in case of power failure)
7. **Wires, Resistors, and Breadboards** (for setting up the circuit)

**System Design and Setup:**

1. **Arduino Board Roles**:
   * **3 "Zone" Arduinos** (Zone 1, Zone 2, and Zone 3):
     + Each board monitors power usage in a specific zone.
     + Measure current and voltage via sensors and calculate power consumption.
     + Store data in EEPROM periodically to retain usage information during power loss.
     + Send data to the central board using serial communication.
   * **1 "Central" Arduino**:
     + Receives and aggregates data from each of the zone boards.
     + Displays total consumption on an LCD or OLED.
     + Provides alerts if total power consumption crosses a pre-defined limit.
     + Optionally, send aggregated data to a PC or other external device for logging and analysis.
2. **Functional Requirements Mapping**:
   * **ADC (Analog-to-Digital Conversion)**:
     + The zone boards use ADC to read analog values from current and voltage sensors.
   * **GPIOs (General Purpose Input/Output)**:
     + Use GPIOs to manage LED indicators and push buttons for each board.
   * **Interrupts**:
     + Use hardware interrupts to handle button presses (for debouncing) and to detect power threshold alerts.
   * **Counters/Timers**:
     + Use timers for periodic sampling of voltage and current values.
     + Configure counters to manage time intervals for logging data to EEPROM.
   * **EEPROM**:
     + Each zone board uses EEPROM to store cumulative power usage, so data is preserved even if power is lost.
   * **Serial Communication**:
     + Use serial communication between the zone boards and the central board to transmit data.
3. **Switch Debouncing**:
   * Implement switch debouncing for the reset button on each board. This can be done using hardware interrupts and a simple debouncing algorithm with counters.
4. **Direct Register Access**:
   * Configure ADC, GPIOs, and timer registers directly via pointers and bitwise manipulation.
   * Implement custom functions for ADC reading, GPIO control, and timer configuration without relying on digital Write or pin Mode.

**Detailed Implementation Outline:**

1. **Analog-to-Digital Conversion (ADC)**:
   * Set up the ADC by directly accessing the ADC control and status registers.
   * Configure the necessary pins as ADC inputs for reading current and voltage sensor data.
2. **GPIO Control**:
   * Use direct register access for controlling LEDs and push buttons. This involves setting data direction and output registers directly for each GPIO pin used.
3. **Timers and Counters**:
   * Configure timers to periodically sample data at specific intervals.
   * Use counters to manage data logging intervals and timing for EEPROM writes.
4. **Interrupts**:
   * Set up interrupts to detect button presses and handle threshold alerts.
   * Use hardware debouncing techniques by employing counters or timed intervals to stabilize input from buttons.
5. **EEPROM Usage**:
   * Use EEPROM for non-volatile storage of power consumption data. Write data periodically to preserve it across power cycles.
   * Directly access EEPROM control registers to handle data reads and writes.
6. **Serial Communication**:
   * Set up UART for serial communication between each zone board and the central board.
   * Ensure efficient data transfer by using specific start and end bytes in the serial data packets to frame messages.