Project Requirements

1. CANopen Communication Simulation (CiA301)

- **Protocol Structure:** Implement the base CANopen protocol over a virtual CAN interface (e.g., SocketCAN).
- **NMT (Network Management):** Handle NMT commands to transition the simulator between states (Pre-operational, Operational, Stopped).
- **SDO (Service Data Object):** Implement SDO read/write interactions to allow a master controller to configure the simulated Object Dictionary.
- **PDO (Process Data Object):** Handle Transmit (TPDO) and Receive (RPDO) message mapping and triggering (event-driven or timer-triggered) to simulate real-time data exchange.
- **Heartbeat:** Generate periodic heartbeat messages to indicate the node's presence and current NMT state.
- **Node-ID Simulation:** Mimic the addressing architecture of the ZLAC8015D based on its specific Node-ID.

2. CiA402 State Machine Implementation

- **Object Dictionary:** Create an internal data structure to store the required CiA402 registers (e.g., indexes `0x6040`, `0x6041`, `0x6060`, `0x6061`, `0x607A`, `0x60FF`, etc.).
- **State Machine Transitions:** Model the strict CiA402 state machine logic: *Switch On Disabled -> Ready to Switch On -> Switched On -> Operation Enable -> Fault*.
- **Control/Status Word Logic:** Process the Control Word (`0x6040`) commands from the master and update the internal state and Status Word (`0x6041`) accordingly.

3. Motor Physics & Operation Modes

- **Dual Motor Emulation:** Simulate both the left and right wheel motors (typically mapped to sub-indexes 01 and 02 in the ZLAC8015D).
- **Profile Position Mode (`0x6060 = 1`):** Emulate moving to a target position using specified profile velocity, acceleration, and deceleration (S-curve simulation).
- **Profile Velocity Mode (`0x6060 = 3`):** Simulate accelerating to and maintaining a target speed.
- **Profile Torque Mode (`0x6060 = 4`):** Emulate torque output based on target torque and torque slope commands.
- **Telemetry Generation:** Continuously calculate simulated values for actual position, velocity, torque, and system status.

4. Graphical User Interface (GUI)

- **Dashboard Framework:** Build an interactive UI using Python/C++/Qt (PyQt/PySide) or a Web UI.
- **Visualization:** Display the current motor state, operating mode, speed, torque, and position.
- **Register Monitoring:** Show live values of the Control Word and Status Word.
- **Interaction:** Provide user input fields to manually send commands (start, stop, enable operation, modify target values) without needing an external CAN master.
- **CAN Logger:** Include a live, scrolling panel showing the raw CAN messages being sent and received.

5. Testing & Deliverables

- **Validation:** Test the simulator against standard CANopen tools or a robotic software stack to ensure message format and timing behavior match the real hardware.
- **Documentation:** Provide documentation detailing how the object dictionary is used and explaining the simulation architecture.
- **Demonstration:** Create a demo showcasing the interaction between the simulator and external robotics control software.