Stage 2 Checklist: Distributed Motor Intelligence (XIAO RP2040)

Stage 2 Checklist: Distributed Motor Intelligence (XIAO RP2040)

■ Objective

Enable per-wheel distributed control using XIAO RP2040 microcontrollers. Each wheel node will handle encoder reading, PID motor control, and report real-time odometry data to the Arduino Nano 33 IoT. The Nano acts as a coordinator between the wheel nodes and future high-level control via Raspberry Pi.

■ Preparation & Hardware Inventory

- [ ] XIAO RP2040 (x4) received and functional
- [ ] Headers soldered on each XIAO RP2040
- [ ] Buck converters (5V or 3.3V output) for clean MCU power
- [ ] Dupont wires, breakout cables, connectors ready
- [ ] Logic-level tools: USB-UART adapter (for testing/debug)
- [ ] Confirm availability of PWM-capable pins on each board

■ Wiring & Mounting Plan

- [ ] Design per-wheel perfboard/bracket for XIAO + TB6612FNG + AS5600
- [ ] Wire motor outputs to TB6612FNG and to motor terminals
- [ ] Connect AS5600 (SDA/SCL) to XIAO (3.3V logic)
- [ ] Connect XIAO to Arduino Nano via I2C (shared SDA/SCL)
- [ ] Common GND for all power and communication lines
- [ ] Mount each wheel module securely on chassis

■■■ Firmware per XIAO Node

- [ ] Initialize I2C communication to receive velocity commands
- [ ] Read encoder data (AS5600 over I2C)
- [ ] Convert encoder angle to velocity and distance
- [ ] Implement PID loop to maintain target speed
- [ ] Output PWM to TB6612FNG motor driver
- [ ] Send feedback (speed, ticks, error) to Arduino Nano
- [ ] Optional: Add basic safety timeout if command not received in X ms

■ Firmware for Arduino Nano 33 IoT

- [ ] Master I2C interface to communicate with all 4 wheel MCUs
- [ ] Send velocity commands periodically (based on joystick, teleop, or plan)
- [ ] Receive speed and tick feedback from each wheel
- [ ] Aggregate odometry into robot-centric position estimate
- [ ] Provide heartbeat signal to detect node failures (optional)
- [ ] Log odometry or forward it to Raspberry Pi (future integration)

■ Communication Protocol (Initial)

- [ ] Define I2C address for each wheel node (e.g., 0x10 - 0x13)
- [ ] Message structure:
- Command to node: [VEL_L, VEL_R] as bytes or int16
- Response from node: [Speed, TickCount, Status]
- [ ] Include basic CRC/checksum (optional)
- [ ] Timing: Commands sent at ~20Hz, feedback received at ~20Hz

■ Testing & Validation

- [ ] Test each wheel node independently
- [ ] Validate encoder reading stability and accuracy
- [ ] Tune PID control loop for smooth response
- [ ] Validate communication (correct addressing, timely response)
- [ ] Compare odometry with manual measurements
- [ ] Stress test all 4 wheels operating together

■ Documentation

- [ ] Document wiring diagram per wheel node
- [ ] Save firmware with version tags (XIAO + Nano)
- [ ] Record PID tuning parameters and logic
- [ ] Save schematic and logic as PDF/image
- [ ] Capture short video demos (optional for portfolio)

■ Future Upgrades & Add-ons (Post Stage 2)

- [ ] Upgrade to CAN bus communication (requires replacing Nano or adding CAN shield)
- [ ] Add current sensing (e.g., INA219 or ACS712) per wheel
- [ ] Implement wheel slip or stall detection logic
- [ ] Add IMU sensor per wheel (shock, tilt, or ground detection)
- [ ] Enable flash logging of encoder ticks and performance per wheel
- [ ] Replace Nano 33 IoT with central CAN-capable controller (Teensy 4.1 or STM32)
- [ ] Integrate with Raspberry Pi (Stage 3) for ROS2-based navigation, SLAM, vision, and teleoperation

■ Milestone

Completion of Stage 2 means the robot is ready for autonomous behavior layering and high-level control in Stage 3.