Closed-Loop Stepper Motor Test Guide

Hardware Requirements

Components Needed:

- **Stepper Motor**: 1 revolution = 4 revolutions of small wheel
- **Optical Encoder**: 20 pulses per revolution (adjustable in code)
- **Arduino/ESP32**: With pins 2, 4, 17, 18, 19 available
- **Limit Switches**: Home switch (left) and safety switch (right)
- **Stepper Driver**: Compatible with step/direction control
- **Linear Rail System**: 80cm total travel distance

Pin Connections:

```
Pin 19 → Stepper STEP signal
Pin 17 → Stepper DIRECTION signal
Pin 18 → Home/Left limit switch (active low with pullup)
Pin 4  → Right safety switch (active low with pullup)
Pin 2  → Optical encoder signal (interrupt capable)
```

System Specifications

- **Motor Driver**: TB6560 with 1/32 microstepping
- **Motor**: 200 steps/rev standard stepper (6400 steps/rev with microstepping)
- **Total Travel**: 80cm in 5 stepper motor revolutions
- **Distance per Motor Revolution**: 16cm (0.16m)
- **Resolution**: 40,000 steps per meter
- **Gear Ratio**: 4:1 (stepper to wheel) - 1 motor rev = 4 wheel revs
- **Start Position**: 2.5cm offset from home position
- **Encoder**: 20 PPR optical encoder on small wheel
- **Encoder Resolution**: 0.002m (2mm) per pulse

Getting Started

1. Initial Setup

1. Upload the code to your Arduino/ESP32
2. Connect hardware according to pin diagram
3. Open Serial Monitor at 115200 baud

4. System will display the main menu

2. First-Time Calibration

Always start with calibration mode before testing other functions:

```
Select option: 3 (Calibration/Homing mode)
Type: start
```

What happens:

- Motor moves toward home switch at slow speed (0.05 m/s)
- When home switch triggers, motor stops and sets position as zero
- Encoder count resets to zero
- Motor moves to start position (2.5cm from home)
- System is now calibrated

Testing Modes

Mode 1: Acceleration Mode

Purpose: Test velocity and acceleration profiles with real-time feedback

Steps:

1. Select option `1`
2. Enter velocity (e.g., `0.05` for 5cm/s)
3. Enter acceleration (e.g., `0.02` for 2cm/s^2)
4. Watch real-time data display
5. Type `return` when motor reaches end

What to observe:

- Position feedback from encoder vs. stepper position
- Velocity ramping up according to acceleration
- Position error correction via PID control
- Smooth acceleration curve

Mode 2: Constant Velocity Mode

Purpose: Test steady-state velocity control

Steps:

1. Select option `2`
2. Enter velocity (e.g., `0.08` for 8cm/s)
3. Motor moves at constant speed
4. Type `return` to return to start

What to observe:

- Steady velocity after initial acceleration
- Consistent encoder feedback
- Position tracking accuracy

Mode 3: Calibration Mode

Purpose: Home the system and set reference position

Use this mode:

- At startup
- After any positioning errors
- When system loses position reference
- To verify limit switch operation

Mode 4: Trapezoidal Motion

Purpose: Test complex motion profiles with acceleration, cruise, and deceleration phases

Steps:

1. Select option `4`
2. Enter acceleration distance (e.g., `0.1` for 10cm)
3. Enter deceleration distance (e.g., `0.15` for 15cm)
4. Enter acceleration rate (e.g., `0.05` for 5cm/s^2)
5. Enter cruise speed (e.g., `0.1` for 10cm/s)
6. Review calculated parameters
7. Type `start` to begin motion

What to observe:

- Three distinct phases: accelerating, cruising, decelerating
- Smooth transitions between phases
- Encoder tracking throughout motion profile

Real-Time Data Display

During any motion, the system displays:

```
Pos: 25.43cm | Vel: 8.21cm/s | Acc: 2.15cm/s² | Steps: -10172 | Target: -32000 | Error: 0.8mm
```

Data explanation:

- **Pos**: Current position from encoder (cm)
- **Vel**: Current velocity from encoder (cm/s)
- **Acc**: Current acceleration from encoder (cm/s^2)
- **Steps**: Current stepper position (steps)
- **Target**: Target stepper position (steps)
- **Error**: Position error between stepper and encoder (mm)

PID Control Testing

PID Tuning Mode

1. Select option `5`
2. Adjust parameters:
   - `kp 2.0` (proportional gain)
   - `ki 0.1` (integral gain)
   - `kd 0.05` (derivative gain)
3. Type `done` when finished

PID Control Commands

- `pid on` - Enable PID control (default)
- `pid off` - Disable PID control for testing

Safety Features

Limit Switches

- **Home Switch**: Stops motion and triggers homing
- **Right Switch**: Emergency stop at far end
- Both switches are active-low with internal pullups

Emergency Commands

- `stop` - Immediate motor stop and mode reset

- `return` - Return to start position (when at end)

Testing Procedures

1. Basic Functionality Test

1. Calibrate system (Mode 3)
2. Test acceleration mode with conservative values:
   - Velocity: 0.02 m/s (2cm/s)
   - Acceleration: 0.01 m/s² (1cm/s²)
3. Verify motor travels exactly 80cm (32,000 steps)
4. Check encoder feedback: should show ~400 total pulses
5. Verify PID error correction keeps error under 2mm

2. Distance Verification Test

1. Run constant velocity mode at 0.05 m/s (5cm/s)
2. Measure actual distance traveled with ruler
3. Should be exactly 80cm for full travel
4. Encoder should show 400 pulses total (5 motor revs × 4 wheel revs × 20 pulses)
5. Check position accuracy: encoder position should match actual distance

3. High Resolution Performance Test

1. Use trapezoidal mode with moderate parameters:
   - Acceleration distance: 0.1m (10cm)
   - Deceleration distance: 0.1m (10cm)
   - Acceleration: 0.05 m/s² (5cm/s²)
   - Cruise speed: 0.1 m/s (10cm/s)
2. Monitor for smooth motion with 1/32 microstepping
3. Check position accuracy: should maintain <2mm error
4. Verify total distance is exactly 80cm

4. PID Tuning Test

1. Disable PID control: `pid off`
2. Run any motion mode
3. Note position errors
4. Enable PID control: `pid on`
5. Repeat same motion
6. Compare error reduction

Troubleshooting

Common Issues:

Large Position Errors (>5mm)

- Check encoder wiring and connection to Pin 2
- Verify TB6560 driver connections (Step=Pin19, Dir=Pin17)
- Ensure encoder wheel rotates 4 times per motor revolution
- Check for mechanical backlash or belt slippage
- Verify 1/32 microstepping setting on TB6560

Motor Travels Wrong Distance

- Confirm TB6560 is set to 1/32 microstepping
- Check SW5-SW8 DIP switches on TB6560:
  - All ON = 1/32 microstepping
  - Verify current setting matches code constants
- Measure actual distance with ruler to verify 80cm travel

Encoder Count Issues

- Should get exactly 400 pulses for full 80cm travel
- Check encoder signal quality on Pin 2
- Verify optical encoder alignment
- Ensure encoder wheel diameter gives 4 revolutions per motor revolution

Motor Not Moving

- Check TB6560 driver power supply (12-36V)
- Verify step/direction signals on pins 19 and 17
- Check motor coil connections to TB6560
- Verify TB6560 current setting (adjust potentiometer)
- Check enable signal if using one

Oscillations with 1/32 Microstepping

- Reduce PID gains (start with Kp=1.0, Ki=0.05, Kd=0.02)
- Check for loose mechanical connections
- Verify encoder signal quality and debouncing
- Ensure adequate power supply for high resolution stepping

TB6560 Driver Issues

- Check DIP switch settings (SW5-SW8 all ON for 1/32)
- Verify input voltage (12-36V recommended)
- Adjust current limit potentiometer for your motor
- Check for overheating (add cooling if needed)

Parameter Adjustment

Key Constants to Modify:

cpp

```cpp
// Motor specifications for TB6560 with 1/32 microstepping
const int MOTOR_STEPS_PER_REV = 200;       // Standard stepper motor
const int MICROSTEP_MULTIPLIER = 32;       // 1/32 microstepping
const int DRIVER_STEPS_PER_REV = 6400;     // 200 × 32 = 6400 steps/rev

// System mechanics
const float TOTAL_DISTANCE_M = 0.8;        // 80cm total travel
const int TOTAL_MOTOR_REVS = 5;            // 5 motor revolutions for 80cm
const float DISTANCE_PER_MOTOR_REV = 0.16; // 16cm per motor revolution

// Calculated system constants
const float STEPS_PER_METER = 40000.0;     // 6400 ÷ 0.16 = 40,000 steps/m
const float MAX_TRAVEL_M = 0.8;            // Maximum travel distance (80cm)
const long TARGET_POSITION = 32000;        // 0.8 × 40,000 = 32,000 steps

// Encoder constants
const int ENCODER_PPR = 20;                // Pulses per revolution of encoder wheel
const float GEAR_RATIO = 4.0;              // 1 motor rev = 4 wheel revs
const float WHEEL_CIRCUMFERENCE = 0.04;    // 0.16m ÷ 4 = 0.04m wheel circumference
const float ENCODER_RESOLUTION = 0.002;    // 0.04m ÷ 20 = 0.002m per pulse
```

PID Tuning Guidelines:

- Start with Kp=2.0, Ki=0.1, Kd=0.05
- Increase Kp for faster response
- Increase Ki to eliminate steady-state error
- Increase Kd to reduce overshoot
- Monitor for oscillations and reduce gains if needed

Expected Performance

With proper tuning and TB6560 setup, expect:

- **Distance Accuracy**: Exactly 80cm travel (measured with ruler)

- **Position Accuracy**: ±1mm with PID control
- **Velocity Accuracy**: ±2% with encoder feedback
- **Encoder Resolution**: 2mm per pulse (400 pulses for 80cm)
- **Step Resolution**: 0.025mm per step (40,000 steps/meter)
- **Smooth Motion**: 1/32 microstepping provides very smooth operation
- **Consistent Repeatability**: Should return to same position within 1mm

TB6560 Specific Performance:

- Quiet operation with 1/32 microstepping
- Good torque retention at low speeds
- Suitable for precision positioning applications
- Works well with encoder feedback for closed-loop control

This system provides comprehensive closed-loop control with real-time feedback, making it ideal for precision positioning applications and motion control research.