Project: CAN-controlled stepper with encoder feedback (STM32 Blue Pill)

1. Summary

This project implements a 2-node system using two STM32F103C8T6 ("Blue Pill") boards:

- **Transmitter (Node A)**: reads a rotary encoder (TIM1 in encoder mode) and transmits the 32-bit encoder count over CAN (ID 0x100) at 100 Hz.
- **Receiver (Node B)**: receives encoder counts over CAN, computes delta since last sample, decides motor direction and PWM duty, and drives a stepper motor via a motor driver (EM882S).

---

2. Hardware (used)

- 2 × STM32F103C8T6 (Blue Pill)
- 1 × ST-Link V2 (programming/debug)
- 2 × CAN transceiver (e.g., MCP2551)
- Rotary encoder: **KY-020**
- Level shifter (for interfacing 5V components)
- Motor driver: **EM882S**
- Stepper motor: **86CM85**
- CAN bus wiring: twisted pair (CAN_H/CAN_L) with 120 Ω termination resistors at both ends

---

3. Software architecture

- Transmitter
  - TIM1 configured as encoder input.
  - Reads encoder counter.
  - Transmit loop rate ≈ 100 Hz
- Receiver
  - CAN Rx: configured filter to accept MSG_ID << 5 (standard ID).
  - TIM3: base timer with interrupt every control tick.
    - Handle encoder overflow / wraparound for 16-bit counter.
    - Determine direction (CW/CCW) and set DIRCTION_PIN.
    - If direction changed since last tick → stop motor, update direction, return (skip further actions this tick).
    - Else compute duty and set PWM compare (TIM2 CH2).

---

4. Key code behaviours & rationale (concise)

- **Encoder overflow handling**
  - Because the encoder counter is 16-bit (0..65535), delta can overflow. We map delta into signed range by:
    - If delta > 32767 → delta -= 65536
    - If delta < -32768 → delta += 65536
  - This recovers the correct signed step delta between samples.
- **Direction & smoothing**
  - Direction is inferred from sign of delta. For CCW we invert delta to compute magnitude.
  - To avoid abrupt reversal and possible mechanical stress, we stop the motor for one tick whenever the direction changes and only update the direction pin then resume on next tick.
- **PWM duty computation**
  - duty = abs(delta) * SCALE; then clamped to timer period.
  - SCALE is used to tune how many encoder steps correspond to PWM duty. It can be adjusted for responsiveness or smoothness.
- **Why timer interrupt?**
  - At first, attempting to run motor updates alongside other code blocked the system or led to missed steps(before using timer interrupt).
  - Running the control loop inside a periodic timer interrupt guarantees consistent sampling & output regardless of main-loop activity.

---

5. Problems faced (observed) & fixes

1. **CAN termination**
   - Issue: unreliable CAN communication.
   - Fix: use **120 Ω** termination resistors at both physical ends of the CAN bus. (Not 100 Ω or 220 Ω.)
2. **Stepper pulses — manual pulse vs PWM**
   - Issue: generating manual pulses (GPIO toggling high-only) did not work reliably.
   - Fix: use hardware PWM from TIM2 (PWM on CH2). PWM provides stable, high-frequency pulses the driver expects.
3. **Parallel tasks causing interference**

   - Issue: motor control blocked other tasks when using naive loops.
   - Fix: move control logic to TIM interrupt (TIM3) to avoid blocking and to provide deterministic timing.

4. **Encoder wrap & noisy direction reversal**
   - Issue: noisy delta around wrap / immediate direction flips.
   - Fix: added overflow handling and logic to stop motor briefly before changing direction.

5. **Timer interrupt affecting motor speed**
   - Observation: interrupt frequency/prescaler & period determine control tick and thus perceived motor speed scaling. We used SCALE multiplier to tune duty relative to delta.

6. **Stability checklist**
   - Always check wiring before debugging code (loose pins, swapped CAN H/L, wrong termination).
   - Verify CAN transceiver Vcc/GND and Rx/Tx connections.
   - Check level shifting for encoder and CAN where logic level differs (5V device to 3.3V MCU).

---

6. Wiring & checklist (practical)

- **Level Shifter**
  - Connect LV to the input voltage and HV to the Output voltage you want for example 3.3v to 5v

  And connect the input signal via TXI and the output via TXO, the level shifter has multiple inputs and outputs (check datasheet)

- **Encoder**
  - KY-020: GND, VCC, A, B. If KY-020 uses 5V pull-ups, use level shifter or configure pull-ups to 3.3V to protect MCU, we also configured it in the code the timer to Encodermode_TI12 so it counts 4x the original rate

- **Motor driver EM882S**
  - Driver Vcc/Gnd rated for motor voltage.
  - We used 25v to power the driver
  - We configured the driver to 800 pulses/rev and 6.4A PEAK / 4.57 RMS
  - (SW1 OFF, SW2 ON, SW3 OFF, SW4 OFF, SW5 OFF, SW6 ON, SW7 ON, SW8 ON)
  - Pulse (PUL), Direction (DIR) pins wired to MCU PWM and GPIO.
  - Leave enable pins untouched

   - Verify logic voltage levels (use level shifter if driver expects 5V).
   - Motor coils correctly connected Via Datasheet, and driver current limit adjusted (if driver supports it).

---

7. Troubleshooting steps (practical flow)

1. Visual & power check: all wires seated; 3.3V and motor V supply present; common ground.
2. Confirm CAN wiring & termination resistors (120 Ω).
3. Use simple test code:
   - Blink LED from MCU to ensure basic program flow.
   - Transmit dummy CAN frames and sniff with other node or CAN analyzer.
   - Toggle PULSE from MCU in a simple loop to test driver/motor operation.
4. If motor moves but control is unstable:
   - Verify encoder readings frequency vs control tick.
   - Try increasing TIM3 rate and/or smoothing delta.
5. If CAN works intermittently:
   - Check bus length and node count; adjust termination (must be exactly at ends) and transceiver wiring.
   - Ensure identical bit timing (same Prescaler and TQ settings) on both nodes.

---

11. Resources & references

- Data sheet of the driver
- Data sheet of the motor
- Data sheet of Rotary Encoder
- 115. STM32CubeIDE MCP2551 CAN BUS with STM32 F103C8T6
- https://www.youtube.com/watch?v=iY_4YOlpqyI&list=PLWNDWPAClRVpz1kldq3PktejQH3D-xRaC&index=7
- http://youtube.com/watch?v=QMgckRoRy38
- https://youtu.be/nLV0fjUWI-g?si=t8spxstYzTB72QhY

---

12. Quick checklist to hand to lab partner

- Both Blue Pills programmed with correct binary (Tx / Rx).
- CAN transceivers powered and wired; twisted pair used for CAN_H/CAN_L.
- 120 Ω termination resistors at both ends of CAN bus.

- Common ground between modules.
- Encoder pull-ups safe for 3.3V MCU (or level shifter present).
- Motor driver current limit adjusted and motor coils wired correctly.
- Verify PWM present at PUL pin with scope before connecting motor.
- Watch for overheating or unusual vibrations.