LAB1.2: Incremental Encoder

การทดลองที่ 1 Incremental Encoder

จุดประสงค์

1. เพื่อศึกษาหลักการทำงานของ Incremental Encoder
2. เพื่อให้สามารถจำแนกลักษณะของสัญญาณ ทิศทาง และสามารถอ่านค่าสัญญาณได้ถูกต้อง
3. เพื่อให้สามารถประมวลผลสัญญาณ และแปลงสัญญาณเป็นค่าที่ต้องการได้
4. เพื่อให้สามารถวิเคราะห์คุณสมบัติของ Encoder ได้
5. ออกแบบ wrap-around และ homing sequence

สมมติฐาน

1. หากเพลาหมุนครบ 1 รอบจะได้จำนวนพัลส์ตามค่า PPR ที่ระบุไว้ในสเปกอุปกรณ์
2. หากสัญญาณ A และ B มีเฟสต่างกัน 90° จะสามารถระบุทิศทางการหมุนได้
3. หากความเร็วในการหมุนเพิ่มขึ้น สัญญาณจะมีความถี่สูงขึ้น
4. หากอ่านค่าแบบ X4 จะได้ความละเอียดของการระบุตำแหน่งสูงกว่าแบบ X1 หรือ X2
5. หากการประมวลผลสัญญาณ และการนับพัลส์มีความถูกต้อง จำนวนพัลส์จะสัมพันธ์เชิงเส้นกับ ตำแหน่งเชิงมุม และความเร็วเชิงมุม

ตัวแปร

1. ตัวแปรต้น :

- ทิศทางการหมุนของเพลา (CW / CWW)
- วิธีการอ่านสัญญาณ Quadrature (Mode X1, X2, X4)
- ความเร็วการหมุนเพลา

2. ตัวแปรตาม :

- ตำแหน่งเชิงมุม (Angular Position)
- ความเร็วเชิงมุม (Angular Velocity)
- คุณภาพของสัญญาณ
- Raw count จาก Encoder

3. ตัวแปรควบคุม :

- แรงดันไฟเลี้ยงของ Encoder และวงจรอ่านสัญญาณ
- ค่า Pulses per Revolution (PPR) ของ Encoder

เอกสารและงานวิจัยที่เกี่ยวข้อง

Incremental encoder คือ เซนเซอร์วัดการเคลื่อนที่เชิงมุม (Angular motion) หรือการเคลื่อนที่เชิงเส้นแล้วเปลี่ยนเป็นสัญญาณพัลส์ เพื่อวัดตำแหน่ง และความเร็ว โดยทั่วไปจะมีช่องสัญญาณ 2 คลื่น (A,B) ที่อยู่ในสถานะ Quadrature มีลักษณะของสัญญาณเป็นคลื่นรูปสี่เหลี่ยม (Square wave) มีเฟสต่างกันประมาณ 90° ซึ่งทำให้สามารถระบุทิศทางการหมุนได้โดยเปรียบเทียบว่าสัญญาณ A ขึ้นก่อน หรือตามหลัง สัญญาณ B (A ขึ้นก่อน หมุนแบบ CW)

ค่าความละเอียด (Resolution) ของ Encoder คือ ค่าที่ใช้บอกระดับความละเอียดของสัญญาณที่ Encoder สามารถแยกแยะได้ในหนึ่งรอบการหมุน สามารถแสดงได้ 2 แบบหลัก ๆ คือ

1. Pulses Per Revolution (PPR) คือ จำนวนพัลส์ที่ออกมาครบหนึ่งรอบของเพลา เป็นค่าคงที่ของ Encoder
2. Cycles Per Revolution (CPR) คือ จำนวนขอบที่ระบบสามารถนับได้จริงต่อรอบ การนับพัลส์ทุกของขึ้น/ลงของสัญญาณ A และ B ทำให้ระบบสามารถอ่านตำแหน่งได้ละเอียดขึ้น โหมดการอ่านแบบ Quadrature (A/B) มี 3 แบบหลัก คือ

1. X1 นับเฉพาะขอบขาขึ้นหรือขอบขาหรือลงของสัญญาณ A (CPR = PPR)
2. X2 นับขอบขึ้น และขอบลงของสัญญาณ A (CPR = 2*PPR)
3. X4 นับขอบขึ้น และขอบลงของสัญญาณ A และ B (CPR = 4*PPR)

สูตรการหาความละเอียดเชิงมุมของ Encoder (Angular Resolution)

$$\mathrm{res}_\theta = \frac{2\pi}{\mathrm{CPR}} \left[\frac{\mathrm{rad}}{\mathrm{pulse}}\right]$$

ตัวแปร

res_θ (Resolution) [rad/pulse] คือ ขนาดของมุมที่เพลาหมุนต่อ 1 พัลส์

CPR คือ จำนวนขอบที่ระบบสามารถนับได้จริงต่อรอบ

สูตรการหาตำแหน่งเชิงมุม (Angular Position)

$$\theta = \mathrm{res}_\theta \cdot \mathrm{n}_{\mathrm{pulse}} \,[\mathrm{rad}]$$

ตัวแปร

$\mathrm{n}_{\mathrm{pulse}}$ คือ จำนวนพัลส์ที่นับได้ระหว่างจุดอ้างอิงเวลา 2 จุด

สูตรการหาความเร็วเชิงมุม (Angular Velocity)

$$\omega = \mathrm{res}_\theta \cdot \frac{\mathrm{d}}{\mathrm{dt}} \mathrm{n}_{\mathrm{pulse}} \left[\frac{\mathrm{rad}}{\mathrm{s}}\right]$$

ตัวแปร

$\frac{\mathrm{d}}{\mathrm{dt}} \mathrm{n}_{\mathrm{pulse}}$ คือ จำนวนพัลส์ที่นับได้ส่วนด้วยความต่างของเวลาระหว่างจุดอ้างอิง 2 จุด

ขั้นตอนการดำเนินงาน

1. การตั้งค่า IOC

- การเลือก Timers ในการทดลองเลือกใช้ TIM3 เนื่องจากเป็น General purpose timer ใช้งานง่ายและไม่รบกวน Timer ของระบบหลักที่ STM32 ใช้อยู่แล้ว สามารถสร้างสัญญาณ PWM ได้ 4 ช่อง ซึ่ง Timer อื่น ๆ สามารถเลือกใช้ได้แต่จะมีข้อจำกัดในการใช้มากกว่า และมีการทำงานที่ซับซ้อนกว่า การเลือกใช้ขึ้นอยู่กับวัตถุประสงค์ของงาน
- ตั้ง Combined Channel เป็น Encoder mode เพื่อนอ่านค่า A และ B จาก encoder
- การตั้งค่า Counter period คือ ค่าที่กำหนดจุดที่ Timer จะรีเซ็ตกลับเป็นศูนย์ การตั้งค่าอยู่ในช่วง 61439 เพื่อให้ค่าใหญ่พอที่จะรองรับค่าที่นับจาก encoder และป้องกันการ Overflow เร็วเกินไป และสามารถตรวจจับการหมุนแบบไป-กลับได้โดยไม่พลาดช่วงที่เปลี่ยนค่า
- ตั้งค่า Encoder mode เพื่อกำหนดการอ่านค่า Quadrature (X1,X2 และ X4) และสามารถกำหนดได้ว่าจะให้ TI1 หรือ TI2 จะเป็นตัวนับหรือตัวกำหนดทิศทาง เช่น กำหนดให้ Encoder mode TI_1 X1 หมายถึงให้นับขาขึ้นหรือขาลงของขา A หรือ B ที่เสียบอยู่กับ pin ที่กำหนดให้เป็น TI1 และ ขาที่เสียบอยู่กับ TI2 จะกลายเป็นตัวกำหนดทิศทางแทน เป็นต้น (โดยมาตรฐาน TI1 อยู่ที่ขา A และ TI2 อยู่ที่ขา B) และในการทดลองจะตั้งการนับเป็นแบบ x4

2. การเตรียมอุปกรณ์และการต่อวงจร

- เชื่อม NUCLEO-G474RE เข้าแล็ปท็อป
- TIM3_CH1 (TI1 เซ็ตเป็น PA6) ต่อที่ขา A และ TIM3_CH2 (TI2 เซ็ตเป็น PA4) ต่อที่ขา B , ต่อ Vcc 3.3 V และ Ground ให้ encoder

3. Simulink Setup

- ตั้งค่าตำแหน่งที่ไฟล์ IOC อยู่และตำแหน่งของ PORT ปัจจุบันที่บอร์ดเสียบอยู่
- Encoder อ่านค่า QEI แบบ x4 และ counter ของ encoderและส่ง raw count ออกมา
- แปลงรูปแบบของข้อมูลเป็น double ด้วยบล็อก Data Type Conversion
- เขียนฟังก์ชั่น Wrap-Around เพื่อรองรับการ Overflow ของค่าที่นับจาก encoder และ ออกแบบ Homing Sequence เพื่อกำหนดให้ค่าที่กำลังนับอยู่นั้นรีเซ็ตกลับไปอยู่ที่จุดอ้างอิงเริ่มต้น (ในการทดลองนี้คือตำแหน่งที่ 0) และทำการทดสอบ
- กด Run on Hardware เพื่อเริ่มเก็บสัญญาณจากอุปกรณ์เข้า Simulink

4. การตรวจสอบการอ่านค่าพื้นฐาน

- ทดสอบหมุนแกน encoder ช้าๆ เพื่อยืนยันว่าค่าจำนวนครั้งนับ (raw count) เปลี่ยนแปลงตามทิศการหมุน (CW/CCW) อย่างถูกต้อง จากนั้นบันทึกพฤติกรรมการเพิ่มหรือลดของค่า raw count เพื่อใช้ประกอบผลการทดลอง

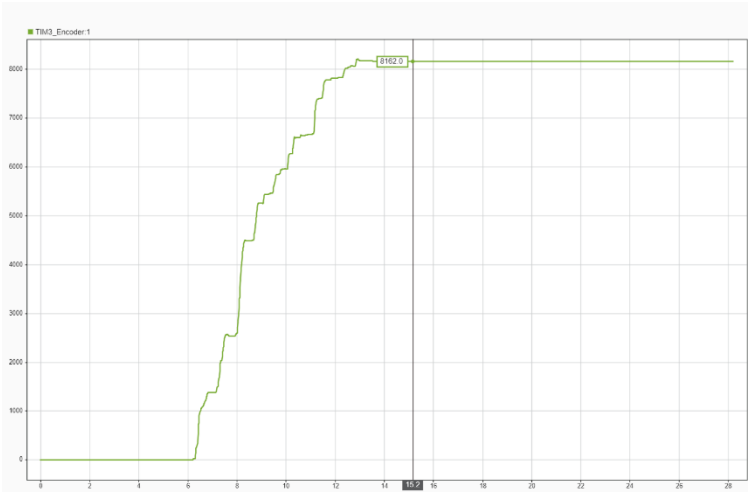
5. การเก็บข้อมูล

- วัดและหาค่า PPR ของอุปกรณ์และแปลงเป็นค่า Angular Resolution
- วัดสัญญาณทิศทางของการหมุน CW และ CCW
- เก็บค่า raw count และแปลงเป็น Relative Position, Angular Position และ Angular Velocity
- ใช้ Data Inspector เพื่อตรวจสอบกราฟสัญญาณ และส่งออกข้อมูลในรูปแบบไฟล์รูปภาพ

ผลการทดลอง

1. หาค่า PPR และคำนวณ Angular Resolution

การหา PPR จะเริ่มจากหมุน encoder 1 รอบและดูจำนวน raw count ที่นับได้จากกราฟ เนื่องจาก encoder ถูกกำหนดรูปแบบการนับมาเป็นแบบ x4 เราจึงต้องหาร 4 เพื่อหาค่า PPR จริงของอุปกรณ์



รูปที่ 1 กราฟตัวอย่างจากการวัด raw count ด้วยการหมุน encoder 1 รอบ

ค่า raw count ที่เก็บได้จากกราฟการทดลอง 5 ครั้งมีดังนี้

ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	Average
8162	8262	8250	8186	8164	8204.8

ตารางที่ 1 ค่าที่วัดได้จากการวัดโดยการหมุน encoder 1 รอบ

เนื่องจากค่าที่ได้จากการทดลองนั้นมีความแตกต่างกันน้อยมากจึงทำการทดลองเพียง 5 ครั้ง จากการทดลองนี้จึงได้ค่าเฉลี่ยที่ encoder อ่านได้เท่ากับ 8204.8 พัลส์/รอบ และนำเลขนี้ไปหาร 4 จึงมีค่าออกมาดังนี้

$$8204.8 \div 4 = 2051.2$$

เมื่อได้ค่ามาจึงนำไปเทียบกับ datasheet ของ encoder คือ AMT103-V incremental encoder และเจอว่าในส่วนของ output resolution (PPR) นั้นมีรุ่นที่ทำ PPR ได้เท่ากับ 2048 พัลส์/รอบ ซึ่งมีความใกล้เคียงกับเลขที่คำนวณมาได้มากที่สุด กลุ่มเราจึงสรุปว่า encoder นี้มีค่า PPR เท่ากับ 2048 พัลส์/รอบ

ต่อมาหา Angular Resolution หาได้จากสมการ

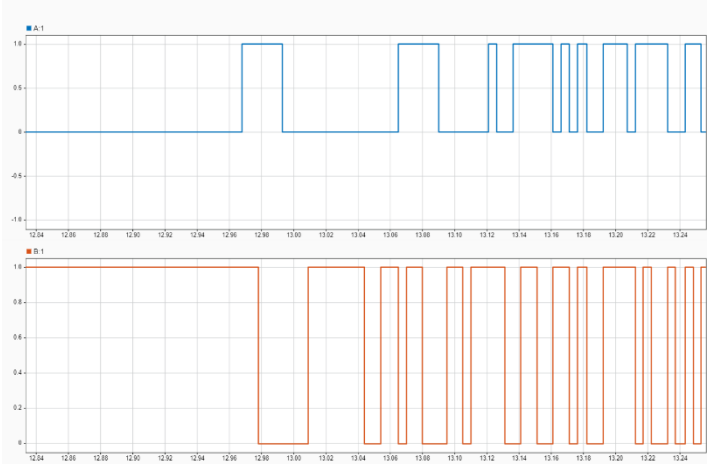
$$\text{res}_\theta = \frac{2\pi}{\text{CPR}} \left[\frac{\text{rad}}{\text{pulse}}\right]$$

ซึ่งเมื่อนำค่า PPR ที่หาได้มาใส่สมการจึงได้ค่าเท่ากับ

$$\text{res}_\theta = \frac{2\pi}{4 \cdot 2048} \approx 0.00077 \left[\frac{\text{rad}}{\text{pulse}}\right]$$

ดังนั้นค่า Angular Resolution เมื่ออ่านแบบ x4 ของ encoder ตัวนี้จึงมีค่าประมาณ**เท่ากับ 0.00077 rad/pulse**

2. Phase relationship ระหว่างสัญญาณ A และ B

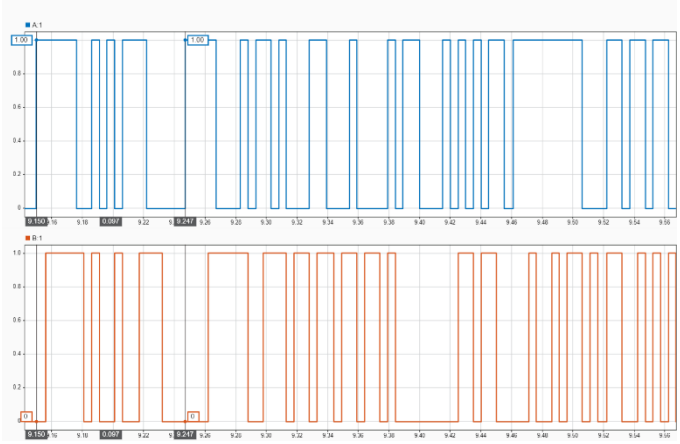


รูปที่ 2 กราฟความสัมพันธ์ระหว่างสัญญาณ A และ B

จากรูปจะเห็นเมื่อเริ่มหมุน สัญญาณ A เริ่มที่สถานะ Low (0) และสัญญาณ B เท่ากับ High (1) ซึ่งทำให้สามารถสรุปได้ว่าระหว่าง 2 สัญญาณนี้ไม่ได้เริ่มที่เดียวกันและมี Phase Shift อยู่**ที่ 90°**

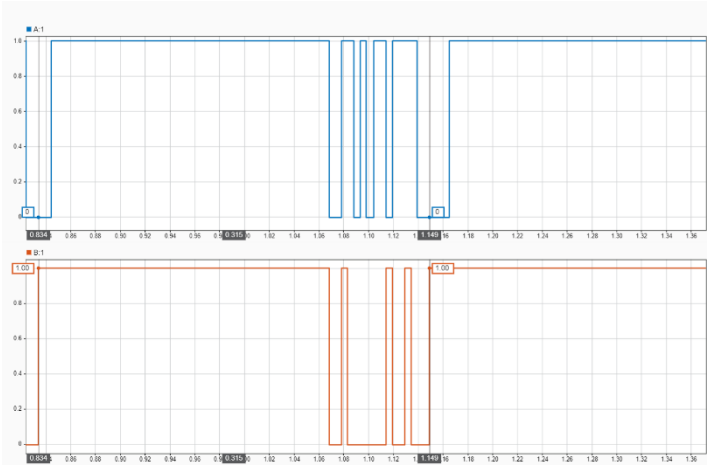
3. ทิศทางสัญญาณเมื่อหมุน encoder ตามเข็ม (CW) และทวนเข็ม (CCW)

การหาทิศทางที่ได้จากกราฟเราจะดูที่เมื่อเริ่มหมุนนั้นสัญญาณ A หรือ B นั้นสัญญาณไหนนำหรือตาม จากการดูขาขึ้นของทั้ง 2 สัญญาณเทียบกัน



รูปที่ 3 กราฟสัญญาณเมื่อทำการทำการหมุน encoder ตามเข็ม

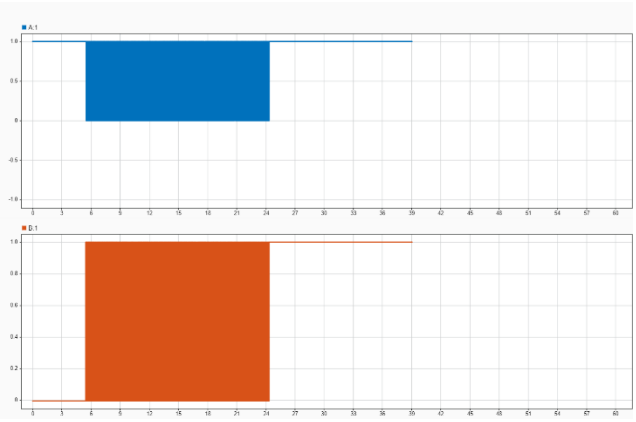
จากภาพนี้เมื่อทำการหมุนในทิศตามเข็ม (CW) ผลที่ได้คือสัญญาณ A นำสัญญาณ B

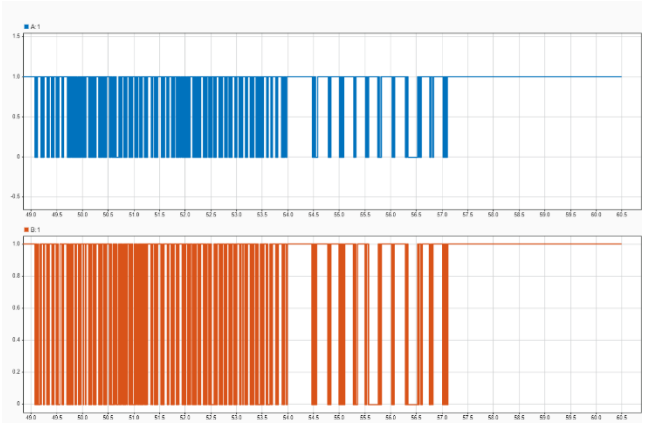


รูปที่ 4 กราฟสัญญาณเมื่อทำการทำการหมุน encoder ทวนเข็ม

จากภาพนี้เมื่อทำการหมุนในทิศตามเข็มสัญญาณ (CWW) ผลที่ได้คือสัญญาณ B นำสัญญาณ A

4. คุณภาพของสัญญาณเมื่อหมุนด้วยความเร็วระหว่าง encoder AMT103-V และ encoder รุ่นหนึ่ง

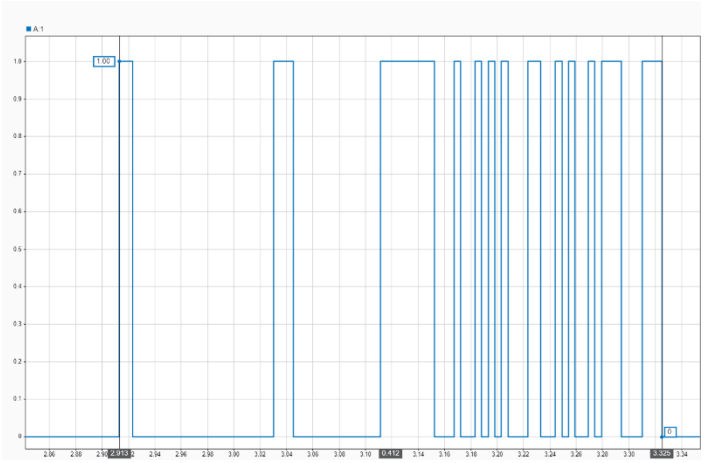




รูปที่ 5 และ 6 กราฟสัญญาณ A และ B ของ encoder รูปทางซ้ายคือกราฟของ encoder AMT103-V และทางขวาคือกราฟของ encoder รุ่นหนึ่งและทำการหมุน encoder ทั้ง 2 ด้วยความเร็ว

จากรูปทั้ง 2 สัญญาณ A และ B มีความต่างที่ชัดเจนในเรื่องของความห่างระหว่างพัลส์เมื่อหมุนด้วยความเร็วด้าน encoder รุ่นหนึ่งเมื่อหมุนด้วยความเร็วมากเท่าไร พัลส์จะยิ่งห่างมากเท่านั้น ซึ่งสามารถบอกได้เป็นนัยว่าความละเอียดของ encoder AMT103-V นั้นมีความละเอียดและคุณภาพมากกว่า encoder รุ่นหนึ่ง

5. การแปลง raw count จากสัญญาณแปลงเป็น Relative Position, Angular Position และ Angular Velocity



รูปที่ 7 กราฟของสัญญาณ A ที่ได้จาก encoder ช่วงหนึ่ง

หา Relative Position ได้จากการกำหนดกรอบอ้างอิงของเวลาช่วงนึงเพื่อหาจำนวนพัลส์ที่ได้ในช่วงเวลาหนึ่ง จากภาพเวลาเริ่มต้นคือ 2.913 วินาที และเวลาตอนปลายเท่ากับ 3.325 วินาที และเมื่อนับจำนวนพัลส์จากกราฟสามารถสรุปได้ทันทีว่า Relative Position นั้นมีค่า**เท่ากับ 13 พัลส์**

หา Angular Position หาได้จากการนำจำนวนพัลส์ในช่วงเวลาหนึ่งมาคูณด้วย Angular Resolution จากข้อ 1. ตามสูตร

$$\theta = \mathrm{res}_{\theta} \cdot \mathrm{n}_{\mathrm{pulse}} \,[\mathrm{rad}]$$

นำค่าที่คำนวณได้มาใส่ในสูตรได้ค่าเท่ากับ

$$\theta = 0.00077 \cdot 13 \approx 0.01001 \,\mathrm{rad}$$

จากค่าที่ได้มาสามารถสรุปได้ว่า Encoder ได้หมุนไปแล้วมีค่าประมาณ**เท่ากับ 0.01001 rad**

หา Angular Velocity ได้จากการนำ Angular Resolution มาคูณกับจำนวนพัลส์ของช่วงเวลาอ้างอิงหารด้วยความต่างของเวลาในช่วงที่อ้างอิงนั้นคือเวลาปลายลบด้วยเวลาต้นตามสูตร

$$\omega = \mathrm{res}_{\theta} \cdot \frac{\mathrm{d}}{\mathrm{dt}} \mathrm{n}_{\mathrm{pulse}} \,[\frac{\mathrm{rad}}{\mathrm{s}}]$$

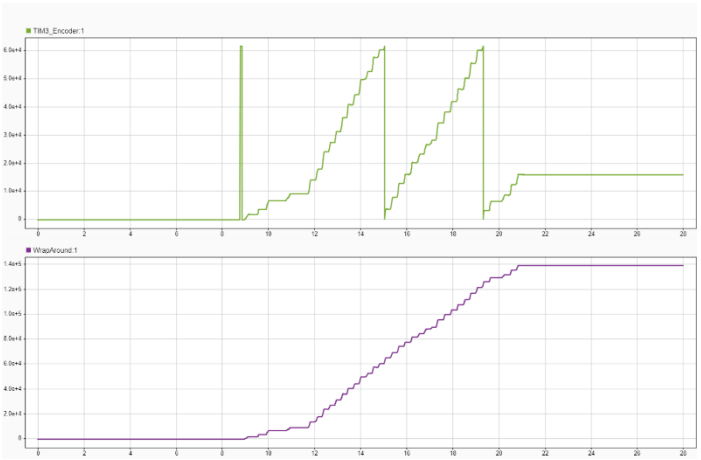
นำค่าที่คำนวณได้มาใส่ในสูตรได้ค่าเท่ากับ

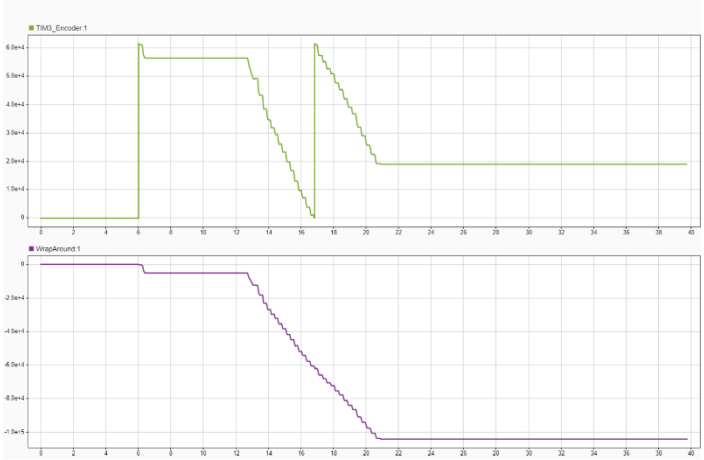
$$\omega = 0.00077 \cdot \frac{13}{3.325 - 2.913} \approx 0.0243 \,[\frac{\mathrm{rad}}{\mathrm{s}}]$$

Angular Velocity ของ encoder ในเวลาระหว่าง 2.913 และ 3.325 จึงมีค่าประมาณ**เท่ากับ 0.0243 rad/s**

6. Wrap-Around

Wrap-Around คือวิธีแก้ปัญหาค่าล้นช่วงนับของตัวนับ encoder เมื่อเคาน์เตอร์ที่นับแบบวนรอบกระโดดกลับจากค่าบนสุดไปเป็น 0 หรือจาก 0 ไปเป็นค่าบนสุด ทำให้ค่าความต่างของจำนวนพัลส์ระหว่างสองตัวอย่างถูกตีความผิดเป็นการเปลี่ยนแปลงขนาดใหญ่มาก ทั้งที่ความจริงเปลี่ยนไปเพียง 1 เคาน์ต์เท่านั้น



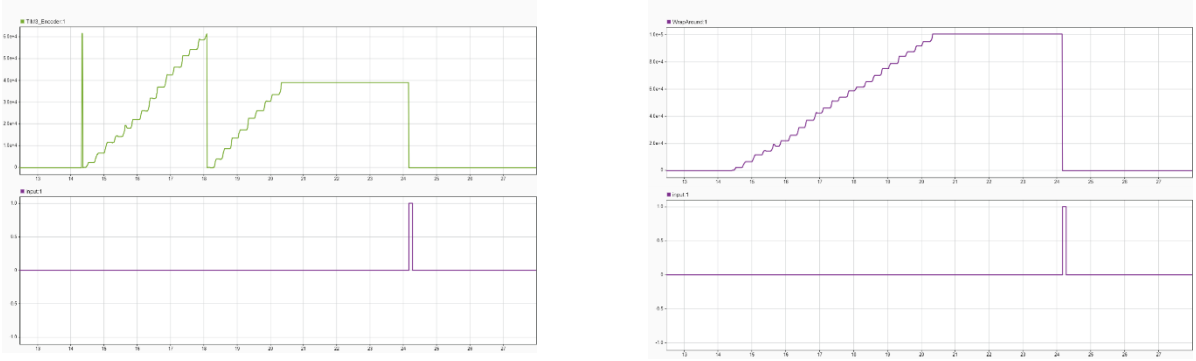


รูปที่ 8 และ 9 กราฟเปรียบเทียบค่า raw count ก่อนและหลังเข้า Wrap-Around ภาพทางซ้ายคือหมุนเพิ่มค่าและทางขวาคือลดค่า raw count

เมื่อหมุนเพื่อเพิ่มหรือลดจำนวนไปเรื่อยๆ ในส่วนของ raw count เมื่ออยู่จุดสูงสุดหรือต่ำสุดตัว raw count จะทำการวนกลับไปที่ตรงกันข้ามทันที แต่ Wrap-Around จะทำให้การนับเพิ่มหรือลดนั้นไม่วนกลับไปที่ตำแหน่งตรงข้ามจะนับไปเรื่อยๆตามรูป 8 และ 9

7. Homing Sequence

Homing Sequence คือกระบวนการกำหนดตำแหน่งอ้างอิงเริ่มต้นของระบบ (Home Position) เพื่อให้ค่าตำแหน่งจาก encoder กลับไปที่ตำแหน่งอ้างอิง (ตำแหน่งอ้างอิงในการทดลองนี้คือ 0) โดยในการทดลองนี้จะใช้ปุ่มสีน้ำเงินที่อยู่บนบอร์ด Microcontroller โดยตั้ง pin PC13 ให้เป็น input ในการส่งสัญญาณ High (1) เพื่อทำการรีเซ็ตค่า raw count จาก encoder และ Wrap-Around ให้กลับมาอยู่ที่จุดอ้างอิง



รูปที่ 10 และ 11 กราฟการเปลี่ยนแปลงค่า count เมื่อ Homing Sequence ทำงาน
โดยภาพทางซ้ายคือ raw count จาก encoder และทางขวาคือจากฟังก์ชั่น Wrap-Around

รูปที่ 12 รูปของบอร์ด STM32 Microcontroller NUCLEO-G474RE

สรุปผลการทดลอง

- ระบบอ่านสัญญาณจาก Incremental Encoder ได้ถูกต้อง เห็นรูปแบบ Quadrature ของช่อง A และ B และใช้ลำดับ phase ยืนยันทิศ CW/CCW ได้
- ทดสอบในโหมดนับ X4 เพียงโหมดเดียว พบจำนวน count ต่อรอบสอดคล้องกับสเปกของอุปกรณ์ (PPR) และคำนวณความละเอียดเชิงมุมได้ถูกต้อง (ความละเอียดเชิงมุมของ X4 เท่ากับ 360 องศาหารด้วยจำนวน count ต่อรอบที่วัดได้)
- การแปลงค่าจาก raw count เป็นจำนวนพัลส์,ตำแหน่งและความเร็วให้แนวโน้มสอดคล้องกับการหมุนจริง
- การแก้ค่าล้นของตัวนับ (wrap-around) ทำให้ข้อมูลตำแหน่งต่อเนื่อง ไม่เกิดการกระโดดของค่า raw count
- กระบวนการกำหนดตำแหน่งอ้างอิงเริ่มต้น (homing sequence) ทำงานได้ เมื่อถึงเงื่อนไขโฮม ระบบตั้งตำแหน่งเป็นศูนย์และกลับสู่การวัดปกติ

อภิปรายผล

- โหมด X4 ให้ความละเอียดสูงสุด จึงประเมินมุมและความเร็วได้ละเอียด แต่ไวต่อสัญญาณรบกวนและความไม่สม่ำเสมอของขอบสัญญาณมากกว่าโหมดที่ละเอียดน้อยกว่า
- จำนวน count ต่อรอบที่วัดได้สอดคล้องกับสเปก ความคลาดเคลื่อนที่พบอธิบายได้จากการหมุนด้วยมือและความละเอียดของเวลาเก็บข้อมูล
- การเลือกค่า Counter Period มีผลต่อโอกาสเกิดการล้น ต้องมีขั้นตอน wrap-around ก่อนคำนวณมุมและความเร็วเพื่อหลีกเลี่ยงค่ากระโดด
- แม้ทดลองจริงเฉพาะ X4 แต่สามารถยืนยันเชิงตรรกะความสัมพันธ์ของ X1 และ X2 ได้จากอัตราส่วนของ count ต่อรอบ (X4 มากกว่า X2 สองเท่า และมากกว่า X1 สี่เท่า)
- วิธีโฮมที่ใช้สวิตช์/เงื่อนไขบนบอร์ดใช้งานได้สะดวก แต่ความทำซ้ำและความแม่นยำต่ำกว่าวิธีอ้างอิงตำแหน่งทางกลเฉพาะ เช่น ช่อง Z (ช่อง index) หรือสวิตช์ลิมิต

ข้อเสนอแนะ

- ใช้แหล่งหมุนความเร็วคงที่แทนการหมุนด้วยมือ และระบุช่วงเวลาเก็บข้อมูลให้ชัดเจน
- เพิ่มการกรองเชิงตัวเลขอย่างง่ายในช่องความเร็ว เช่น ค่าเฉลี่ยเคลื่อนที่ เพื่อลดการสั่นของค่าที่คำนวณ
- ปรับ Counter Period ให้เหมาะกับย่านความเร็วที่ใช้งาน ลดโอกาสล้นโดยไม่เสียความละเอียดเกินจำเป็น
- ทำขั้นตอน wrap-around ก่อนการคำนวณตำแหน่งและความเร็วทุกครั้ง
- หากต้องการความแม่นยำของจุดอ้างอิงสูงขึ้น ให้ใช้ช่อง Z ของ encoder หรือสวิตช์ลิมิตสำหรับโฮม

อ้างอิง

Incremental Encoder Signals 101 (สัญญาณที่ได้จาก Encoder และค่าความละเอียด)

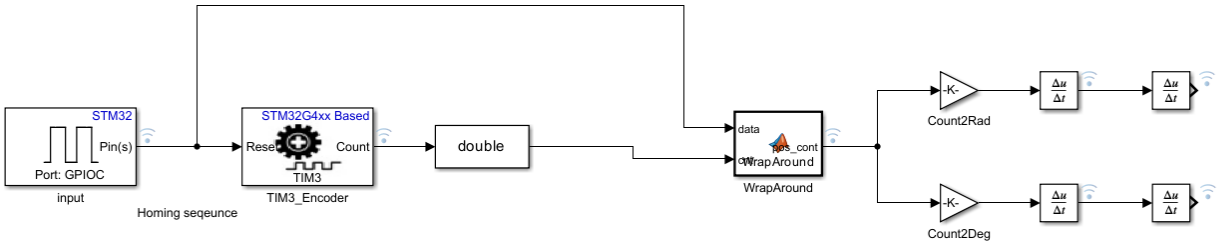
https://drive.google.com/file/d/1mFoqoDe_3F_oH_7Rjq2fEOqsgK6uGTP4/view?usp=sharing (หลักการทำงานของ Encoder)

https://drive.google.com/file/d/1O-dHPAbwDYKtX--1FQ_iwLBVp6TBAVcP/view?usp=sharing (Overflow)

https://drive.google.com/file/d/1MvFKP1fLEzK535CkfOCROg3cMpON89WR/view?usp=sharing (AMT103-V Incremental encoder datasheet)

ภาคผนวก

1. โครงสร้างของ Homing Sequence และ Wrap-Around



รูปที่ 1 โครงสร้างการทำงานใน Simulink

```
WrapAround
1  function pos_cont = WrapAround(data,cnt)
2  persistent last_cnt acc
3  if isempty(last_cnt)
4      last_cnt = int32(cnt);
5      acc = int64(0);
6  end
7  %homing sequence for wrap around
8  if data == 1
9      last_cnt = int32(0);
10     acc = int64(0);
11 end
12
13 MAX_COUNT = int32(61439);
14 RANGE = int32(MAX_COUNT) + int32(1);   % 61440
15 HALF = RANGE/2;
16
17 delta = int32(cnt) - last_cnt;
18
19 % detect wrap
20 if delta > HALF
21     delta = delta - RANGE;   % clockwise large positive -> negative jump (wrap backward)
22 elseif delta < -HALF
23     delta = delta + RANGE;   % counterclockwise large negative -> positive jump (wrap forward)
24 end
25
26 acc = acc + int64(delta);   % accumulate corrected change
27 pos_cont = double(acc);
28 last_cnt = int32(cnt);
29 end
30
```

รูปที่ 2 โค้ดในการสร้างฟังก์ชั่น Wrap-Around

- Wrap-Around

  โค้ดมีทั้งหมด 29 บรรทัดและมีการทำงานดังนี้

  บรรทัดที่ 1: ประกาศ output ของฟังก์ชั่นชื่อ pos_cout และ กำหนด input จำนวน 2 ตัวแปรในฟังก์ชั่นคือ data รับมาจากการกดปุ่มสีน้ำเงินที่อยู่บนบอร์ด Microcontroller และ cnt คือค่า raw count ที่ได้มาจาก encoder

  บรรทัดที่ 2: ประกาศตัวแปรแบบไม่เปลี่ยนแปลงเมื่อฟังก์ชั่นจบที่คล้ายกับการใช้ static ใน C/C++ ชื่อ persistent โดยตั้งชื่อตัวแปรคือ last_cnt เพื่อเก็บค่าที่อ่านได้ก่อนหน้าของ raw count และ acc คือ

accumulate หรือตัวสะสมเพื่อส่งออกข้อมูลที่ไม่เกิดการ overflow จาก raw count ของ encoder ไปยัง output

บรรทัดที่ 3-6: คือการกำหนดค่าให้ตัวแปร last_cnt และ acc หาก last_cnt ไม่มีการกำหนดค่าไว้หรือ NULL ให้กำหนดทั้ง 2 ตัวเป็นค่า 0 โดยมีการใช้ built-in function อย่าง int32 และ int64 เนื่องจากใน Matlab function ไม่สามารถเก็บค่า persistent ในรูปแบบเป็น double ได้ จึงต้องใช้ฟังก์ชั้นเพื่อกำหนดประเภท โดย

Int32() คือประเภท int แบบ Signed 32-bit ที่เก็บค่าได้ตั้งแต่ $[-2^{16},2^{16}-1]$

Int64() คือประเภท int แบบ Signed 64-bit ที่เก็บค่าได้ตั้งแต่ $[-2^{32},2^{32}-1]$

บรรทัดที่ 8-11: เมื่อมีการกดปุ่มสีน้ำเงินบนบอร์ด microcontroller จะมี pulse ขาขึ้นหรือสัญญาณ High (1) ขึ้นมา เมื่อเจอสัญญาณนี้ให้ฟังก์ชั่น Wrap-Around รีเซ็ตค่าที่นับได้กลับไปที่จุดอ้างอิงหรือตำแหน่งที่ 0

บรรทัดที่ 13-17: กำหนด 4 ตัวแปรคือ MAX_COUNT คือค่าที่ encoder จะสามารถส่งข้อมูลได้ตาม counter period ที่ตั้งไว้, RANGE คือระยะห่างรวมระหว่างจุดเริ่มต้นและจุดปลายการกำหนดเขตนำไปหาค่ากลางของ counter period ต่อไป มีค่าเท่ากับ 61440, HALF คือค่ากลางเพื่อนำไปเช็คค่าที่วนซ้ำหรือ overflow ต่อไป มีค่าเท่ากับ 30720 และ delta เพื่อตรวจจับค่ากระโดดเมื่อ raw count จาก encoder เกิด overflow

บรรทัดที่ 18-24: เป็นการตรวจสอบค่ากระโดดหรือ overflow ของ encoder โดยการทำงานคือหาก delta ซึ่งสมมุติกำหนดค่าให้เป็น 61439 หมายความว่าเป็นการหมุนลดจำนวนจาก 0 มาเป็น 61439 และ Half คือ 30720 ซึ่งจะเข้าเงื่อนไขแรกและ delta จะเปลี่ยนค่ากลายเป็น delta – RANGE หรือ 61439 – 61440 นั้นเท่ากับ -1 ซึ่งจะทำให้เลขนับต่อไปได้โดยไม่เกิดการกระโดดของค่าอีก ในเงื่อนไขอีกตัวทำงานคล้ายกันแต่ delta จะกลายเป็น +1 แทน

บรรทัดที่ 25-29: ช่วงสุดท้ายของโค้ดเป็นการกำหนดค่าของ Wrap-Around ที่ถูกต้องและนำข้อมูลออกเป็น output โดยตัวแปร acc ทำการสะสมค่าเพิ่มหรือลดจาก delta และให้ส่งออก acc เป็น output โดยมีการเปลี่ยนประเภทของตัวแปรให้เป็น double เพื่อนำไปแสดงผลในกราฟอีกทีและเก็บค่า raw count ปัจจุบันไว้ใน last_cnt เพื่อนำไปเปรียบเทียบกับการนับตัวอื่นๆต่อไป

- Homing sequence

  สร้างบล๊อก digital read เพื่อรับค่าจากการกดปุ่มสีน้ำเงินบนบอร์ด microcontroller และตั้งค่าบล็อกที่ใช้ในการอ่านค่า raw count จาก encoder ให้สามารถรีเซ็ตได้เมื่อมีสัญญาณ High เข้ามา จากนั้นลากเส้นบล๊อกนี้เข้าไปที่ encoder และ Wrap-Around เพื่อให้แต่ล่ะบล็อกสามารถอ่านค่าของปุ่มได้และทำการรีเซ็ตค่าทุกครั้งที่กดปุ่ม (การรีเซ็ต Wrap-Around ถูกอธิบายไว้ในบรรทัดที่ 8-11 ของโค้ดไว้แล้ว)