RMXplorer

LAB 2: Actuators

Name

- นายจักพรรณ ศรีรอดบาง 67340500005
- นางสาวณภัทร บัวพา 67340500011
- นายทัดชัย สุขสราญ 67340500016
- นายธนดล พูลสระคู 67340500017

Objectives

- เพื่อให้เข้าใจลักษณะทางไฟฟ้า–กลของมอเตอร์แต่ละประเภท
- เพื่อให้การควบคุมมอเตอร์ด้วยวิธีต่าง ๆ
- เพื่อให้ความเสถียรและข้อจำกัดของมอเตอร์
- เพื่อให้ฝึกการวัดจริง วิเคราะห์ข้อมูล และเชื่อมโยงกับทฤษฎี
- เพื่อให้รู้วิธีเลือกมอเตอร์ที่เหมาะสมกับงาน

1. DC Motor

การทดลองที่ 1 Motor Characteristic

จุดประสงค์

1. เพื่อหา K_t (Torque Constant) จากความสัมพันธ์ระหว่างแรงบิด และกระแส
2. เพื่อสร้างกราฟ Motor Characteristic จากมอเตอร์บนบอร์ดการทดลองได้
3. เพื่อหาค่า Stall Torque

สมมติฐาน

หากวัดแรงบิดจากโหลด และกระแสจากเซนเซอร์ได้อย่างถูกต้อง กราฟ T เทียบกับ I จะได้เส้นตรงผ่านจุดกำเนิด และความชันของเส้นตรงนั้นคือค่า K_t

ตัวแปร

1. ตัวแปรต้น*:*

- *Voltage*
- *Duty cycle (0%, 20%, 40%, 60%, 80%, 100%)*

2. ตัวแปรตาม*:*

- *Torque*
- *Current*
- *Power*
- *Efficiency*

3. ตัวแปรควบคุม*:*

- *แรงดันไฟฟ้าขาเข้า*
- *กระแสไฟ*
- *Load Cell*
- *ความถี่ PWM 2000 Hz*

เอกสารและงานวิจัยที่เกี่ยวข้อง

มอเตอร์ไฟฟ้ากระแสตรง หรือ DC Motor นั้นประกอบไปด้วย 2 ส่วน ดังนี้ส่วนที่อยู่กับที่ จะเรียกว่า Stator ที่มีขดลวดสนาม (Field Coil) และส่วนที่เคลื่อนที่ จะเรียกว่า Rotor โดยส่วนนี้จะประกอบไปด้วยขดลวดอาร์เมเจอร์ (Armature) และแปรงถ่าน (Brush) หลักการพื้นฐานประกอบด้วยขดลวด 2 ชุด ซึ่งขดลวดชุดหนึ่งอยู่ที่ Stator เรียกว่าขดลวดสนาม (Field winding) ที่ทำหน้าที่สร้างสนามแม่เหล็กถาวร ซึ่งแหล่งจ่ายไฟฟ้ากระแสตรงที่จ่ายมานั้นจะมาจากแหล่งเดียวกันกับขดลวดอาร์เมเจอร์ แต่ในบางครั้งสำหรับมอเตอร์ขนาดเล็กนั้นจะใช้แม่เหล็กถาวรแทนการใช้ขดลวดเพื่อสร้างสนามแม่เหล็กถาวร และขดลวดชุดที่สองที่อยู่ในส่วนของ Rotor จะเรียกว่าขดลวดอาร์เมเจอร์ (Armature winding) ซึ่งจะจ่ายไฟฟ้ากระแสตรงเข้าขดลวดอาร์เมจอร์

ผ่านแปรงถ่าน (Brush) และชุด Commutator ซึ่งตัวขดลวดนั้นจะทำให้เกิด Torque ในการหมุนของ Rotor ที่เกิดมาจากการกระทำระหว่างขั้วแม่เหล็กของขดลวดใน Stator และ Rotor ที่ต่างขั้วกัน และผลักกันทำให้เกิดการหมุนขึ้นได้ หากกลับขั้วของการจ่ายไฟมอเตอร์ก็จะหมุนกลับทิศทาง สมการพื้นฐานของมอเตอร์กระแสตรงสามารถนิยามได้ดังนี้

$$\mathrm{V} = \mathrm{E} + \mathrm{I_a R_a}$$

$\mathrm{V} =$ แรงดันจ่าย (V)

$\mathrm{E} =$ แรงดันไฟฟ้าย้อนกลับ (Back-EMF) (V)

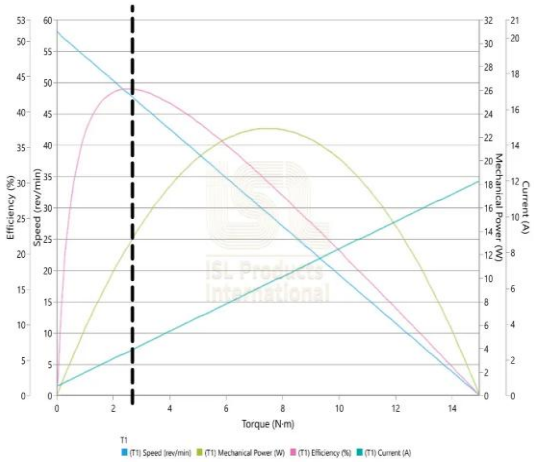
$\mathrm{I}_a =$ กระแสที่ไหลในอาร์มาเจอร์ (A)

$\mathrm{R}_a =$ ความต้านทานของขดลวดอาร์มาเจอร์ (Ω)

แรงบิดของมอเตอร์กระแสตรงจะแปรผันโดยตรงกับกระแสที่ไหลในขดลวดอาร์มาเจอร์ เมื่อเพิ่มกระแสมอเตอร์จะสร้างแรงบิดเพิ่มขึ้นในอัตราส่วนคงที่ ซึ่งสามารถหาค่าได้จากความชันของกราฟแรงบิดต่อกระแส

$$\mathrm{T} = \mathrm{K_t} \cdot \mathrm{I_a}$$

d คือ แรงบิดสูงสุดที่มอเตอร์สามารถสร้างได้เมื่อเพลาหยุดนิ่ง (ความเร็ว = 0) สมการคือ $\tau_{st} = \mathrm{mgL}$ เป็นค่าที่ใช้บอกความสามารถสูงสุดของมอเตอร์ในการเริ่มหมุนหรือเอาชนะแรงต้านสูงมากๆ



รูปที่ 1 แสดลักณะกราฟ Motor Characteristic

สมการ DC Motor ที่ใช้ในการสร้าง Motor Characteristic Curve ดังนี้

สมการหา ω (Speed) = $\left(\frac{0-\omega_{NL}}{\tau_{ST}}\right)\tau_L + \omega_{NL}$

สมการหา I (Current) = $\left(\frac{i_{ST}-i_{NL}}{\tau_{ST}}\right)\tau_L + i_{NL}$

สมการหา P (Power) = $-\frac{\omega_{NL}}{\tau_{ST}}\tau_L^2 + \omega_{NL}\tau_L$

สมการหา η (Efficiency) = $\frac{P_{out}}{P_{in}} = \frac{-\frac{\omega_{NL}}{\tau_{ST}}\tau_L^2+\omega_{NL}\tau_L}{\left(\frac{i_{ST}-i_{NL}}{\tau_{ST}}\right)\tau_L v_{in}+i_{NL}v_{in}}$

ตารางที่ 1 ความสัมพันธ์ของกราฟ Motor Characteristic

กราฟ	ความสัมพันธ์	คำอธิบาย
Speed vs Torque	ความเร็วลดลงเชิงเส้นเมื่อแรงบิดเพิ่มขึ้น	โหลดมากขึ้น ⟶ ความเร็วลดลง เพราะแรงดันตกที่ขดลวดมากขึ้น
Current vs Torque	กระแสเพิ่มตามแรงบิด	แรงบิดมากขึ้นต้องใช้กระแสมากขึ้น (T = Kt·I)
Power vs Torque	กำลังกลเพิ่มขึ้นจนถึงจุดสูงสุดกลางกราฟ	กำลังสูงสุดที่ ~50% ของ Stall Torque
Efficiency vs Torque	โค้งนูน มีค่าสูงสุดช่วงแรงบิดกลาง ๆ	ที่โหลดต่ำหรือสูงเกินไป ประสิทธิภาพจะลดลง

ขั้นตอนการดำเนินงาน

การหา Stall torque

ปรับค่า Duty Cycle ที่ 100% จากนั้นทำการเคาะก้านลงบน Load cell อ่านค่าที่ได้เมื่อมอเตอร์หยุดทำงาน และอ่านค่ากระแส จากนั้นแทนค่าในสมการการหา Stall torque

การหา K_t

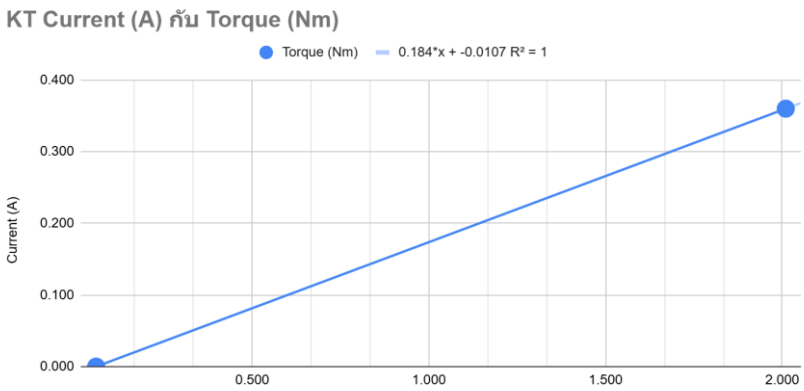
ทำเช่นเดียวกับการหาค่า Stall torque แต่วางก้านลงไปแล้วค่อยใส่แรงให้กับมัน เพิ่มการอ่านค่ากระแส ทำการทดลองซ้ำจำนวน 3 ครั้ง เพื่อหาค่าเฉลี่ยของกระแส จากนั้นนำไปสร้างกราฟเส้นตรงแรงบิดเทียบกับกระแส ค่าความชันที่ได้จากกราฟคือ K_t

การหากราฟ Motor Characteristic

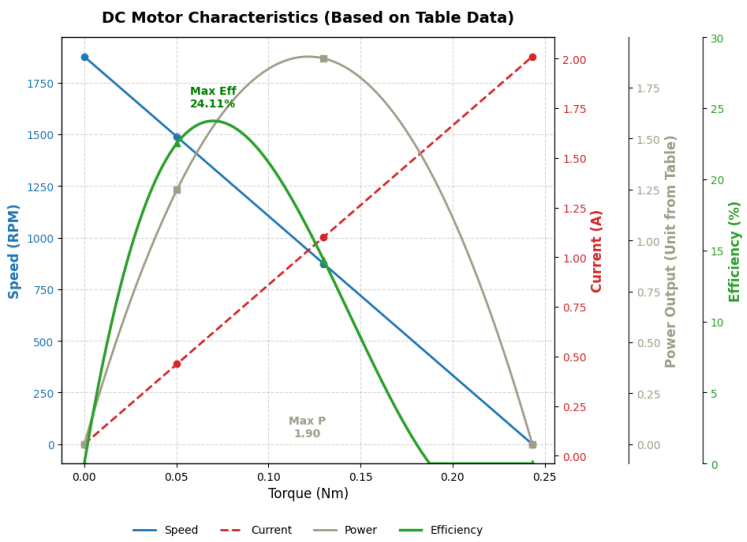
เริ่มจากการติดตั้งบอร์ด และปรับค่าตามตัวแปรควบคุม จากนั้นทำการเก็บค่าที่เป็นค่าคงที่จากสมการ Motor Characteristic Curve สำหรับการหาค่า τ_L จากการปรับ Duty cycle แต่ละค่า ทำการทดลองซ้ำ 3 ครั้ง และหาค่าเฉลี่ย นำค่าที่ได้มาแทนลงในสมการ เพื่อสร้างกราฟ

ผลการทดลอง

การหา Stall torque: L มีความยาว 111.27 mm. m อ่านค่าได้ 0.337 kg และ g มีค่า 9.81 นำไปแทนในสมการ $\tau_{st} = \mathrm{mgL}$ เพราะฉะนั้นจะได้ Stall torque = 0.368 Nm



รูปที่ 2 กราฟแสดงความสัมพันธ์ระหว่างแรงบิดและกระแส



รูปที่ 3 กราฟ Motor Characteristic

สรุปผลการทดลอง

ค่า Torque Constant จะได้ K_t = 0.184 Nm/A ซึ่งมาจากค่าความชันของกราฟเส้นตรง พบว่าความสัมพันธ์ระหว่างแรงบิดและกระแสเป็นเส้นตรงตลอดช่วงการทดลอง ซึ่งสอดคล้องกับสมการ $T = K_t I$ ทำให้สามารถคำนวณค่า K_t ได้อย่างแม่นยำ

ค่า Stall Torque จากการเพิ่มโหลดทีละระดับจนความเร็วลดลงเหลือศูนย์ พบแรงบิดสูงสุดที่มอเตอร์สามารถพัฒนาได้ โดยจุดนี้มีกระแสสูงสุดตามลักษณะของมอเตอร์ DC ค่า Stall Torque ที่ได้สามารถใช้เป็นข้อมูลอ้างอิงสำหรับการประเมินความสามารถด้านโหลดของมอเตอร์ในงานจริง

Motor Characteristic Curve สามารถสรุปค่าที่ทำให้มอเตอร์ทำงานได้มีประสิทธิภาพสูงสุดที่แรงดันไฟฟ้า 12V ได้

Max Speed: 1876.67 RPM (ที่ Torque 0 Nm หรือ No-Load)

Max Current: 2.01 A (ที่ Torque 0.2433 Nm หรือ Stall)

Max Power Output: 1.90 W (ที่ Torque ประมาณ 0.1215 Nm)

Max Efficiency: 24.11 % (ที่ Torque ประมาณ 0.0702 Nm)

กราฟ Speed–Torque มีลักษณะเป็นเส้นตรงลดลงตามทฤษฎี คือ ความเร็วลดลงเมื่อแรงบิดเพิ่มขึ้นจนถึงจุดหยุดหมุน ขณะที่กราฟ Current–Torque เพิ่มขึ้นแบบเส้นตรง และกราฟ Power–Torque มีลักษณะโค้งพาราโบลาซึ่งสูงสุดที่ประมาณครึ่งหนึ่งของแรงบิดสูงสุด กราฟ Efficiency–Torque แสดงว่าประสิทธิภาพสูงสุดอยู่ที่โหลดเบาถึงระดับปานกลางตามทฤษฎีมอเตอร์ DC

โดยรวม ผลการทดลองสามารถใช้พัฒนาแบบจำลองเชิงคณิตศาสตร์ของมอเตอร์ และเป็นข้อมูลสำหรับการออกแบบระบบควบคุม ความเร็ว และโหลดของเครื่องกลในงานประยุกต์ต่าง ๆ ได้อย่างมีความน่าเชื่อถือ

อภิปรายผล

Stall Torque อาจได้รับผลจากข้อจำกัดของอุปกรณ์วัด เนื่องจากการวัด Stall Torque ต้องการกระแสสูงมาก การจำกัดกระแสของแหล่งจ่ายหรือวงจรป้องกันอาจทำให้ค่า Stall Torque ต่ำกว่าความเป็นจริง นอกจากนี้ความร้อนที่เกิดขึ้นระหว่างการทดลองอาจทำให้แรงบิดลดลงเล็กน้อย

แรงบิดของมอเตอร์กระแสตรงจะแปรผันโดยตรงกับกระแสที่ไหลในขดลวดอาร์มาเจอร์ เมื่อเพิ่มกระแสเข้าไปมอเตอร์จะสร้างแรงบิดเพิ่มขึ้นในอัตราส่วนคงที่ K_t ซึ่งสามารถหาค่าได้จากความชันของกราฟแรงบิดต่อกระแส

กราฟ Speed–Torque ความเร็วจะสูงสุดที่โหลดเป็นศูนย์ เมื่อแรงบิดเพิ่มขึ้น ความเร็วลดลงแบบเส้นตรง และที่แรงบิดสูงสุด (Stall Torque) ความเร็ว = 0 สรุปคือ ยิ่งโหลดมาก ยิ่งหมุนช้าลง

กราฟ Current–Torque มอเตอร์กินกระแส น้อยที่สุดตอนหมุนฟรี แรงบิดมอเตอร์แปรผันเชิงเส้น กับกระแส ยิ่งต้องสร้างแรงบิดมาก จะยิ่งกินกระแสมาก สรุปคือ กระแสเพิ่มตามแรงบิดอย่างเชิงเส้น 100%

กราฟ Power–Torque จากความสัมพันธ์ Speed–Torque ที่เป็นเส้นตรง Power จะเป็นเส้นโค้งพาราโบลา (โค้งพุ่งขึ้นแล้วลง) ที่ Torque = 0 จะได้ Power = 0 (เพราะไม่มีโหลด) ที่ Stall จะมี Power = 0 (เพราะความเร็ว = 0) Power สูงสุดอยู่ตรงกลางของกราฟ (ประมาณครึ่งหนึ่งของ Stall Torque) กล่าวคือ มอเตอร์ให้กำลังสูงสุดที่ประมาณ 40–60% ของ Stall Torque

กราฟ Efficiency–Torque ประสิทธิภาพต่ำมากที่โหลดต่ำ (เพราะกระแสสูญเสียคงที่) เพิ่มขึ้นเมื่อโหลดมากขึ้นเล็กน้อย สูงสุดที่ประมาณ แรงบิด 20–40% ของ Stall Torque ลดลงเมื่อใกล้ Stall เพราะกระแสสูงมากทำให้เกิดความร้อนและ Loss Efficiency สูงสุดจะไม่ใช่ตรงที่ Power สูงสุด แต่จะต่ำกว่าเล็กน้อย

ข้อเสนอแนะ

- *ตรวจสอบสายไฟทุกครั้งก่อนทำการทดลอง เพื่อให้ได้ค่าที่ถูกต้อง*
- ค่าความร้อนที่เกิดขึ้นอาจจะส่งผลต่อผลการทดลอง ทำให้คลาดเคลื่อนจากความเป็นจริงเล็กน้อย

อ้างอิง

https://naichangmashare.com/2021/05/28/electric-motor-ep-1/

https://beingelectricalengineer.blogspot.com/2019/08/characteristics-of-dc-shunt-motor.html

การทดลองที่ 2 ผลของ PWM (Duty Cycle และ Frequency) ต่อพฤติกรรมมอเตอร์

จุดประสงค์

1. เพื่อศึกษาผลของการเปลี่ยนค่า Duty Cycle ต่อความเร็วและกระแสของมอเตอร์กระแสตรง
2. เพื่อศึกษาผลของการเปลี่ยนค่า ความถี่ของสัญญาณ PWM ต่อความเรียบของความเร็ว และการตอบสนองของมอเตอร์
3. เพื่อหาความสัมพันธ์ระหว่างแรงดันเฉลี่ยที่เกิดจาก PWM กับ ความเร็วเชิงมุมของ มอเตอร์
4. เพื่อทำความเข้าใจหลักการควบคุมความเร็วของมอเตอร์ด้วย Pulse Width Modulation (PWM) และพฤติกรรมของมอเตอร์ในเชิงไดนามิก

สมมติฐาน

1. เมื่อเพิ่ม *Duty Cycle* ความเร็วและกระแสของมอเตอร์จะเพิ่มขึ้นอย่างสัดส่วน
2. ความถี่ *PWM* ที่สูงขึ้นจะทำให้ความเร็วของมอเตอร์มีความเรียบมากขึ้นและลด *ripple*
3. ค่าแรงดันเฉลี่ยจาก *PWM* จะสัมพันธ์เชิงเส้นกับความเร็วของมอเตอร์

ตัวแปร

1. ตัวแปรต้น:

- *Duty Cycle (0, 20%, 40%, 60%, 80%, 100%)*
- *ความถี่ PWM (0, 5000, 10000, 15000, 20000Hz)*

2. ตัวแปรตาม:

- *ความเร็วมอเตอร์*
- *กระแสไฟฟ้า*
- *ประสิทธิภาพ*
- *Torque*

3. ตัวแปรควบคุม:

- *แรงดันไฟฟ้าที่จ่ายเข้า*
- *Load Cell*
- *ความถี่ PWM (2000 Hz)*
- *Duty Cycle (100%)*

เอกสารและงานวิจัยที่เกี่ยวข้อง

PWM หรือ Pulse Width Modulation คือวิธีการควบคุมแรงดันเฉลี่ยที่จ่ายให้มอเตอร์ โดยการเปิด – ปิดสวิตช์ (เช่นทรานซิสเตอร์ใน H-Bridge) ด้วยความถี่คงที่ แต่เปลี่ยนระยะเวลาที่เปิด ในแต่ละรอบ หลักการพื้นฐานคือ สัญญาณ PWM มีรูปเป็นสัญญาณสี่เหลี่ยม มีคาบเวลา (T) และ ความถี่ (f = 1/T) คงที่สิ่งที่เปลี่ยนคือระยะเวลาที่สัญญาณอยู่ในสถานะ HIGH หรือเรียกว่า Duty Cycle แรงดันเฉลี่ยที่ มอเตอร์ได้รับคือ

$$V_{avg} = D \times V_{\sup ply}$$

โดยที่

D = Duty Cycle (ส่วนของเวลาที่สัญญาณเปิด)

$V_{\sup ply}$ = แรงดันที่จ่ายเต็ม (เช่น 12 V)

Duty Cycle (%) คือ อัตราส่วนของช่วงเวลาที่สัญญาณอยู่ในสถานะ HIGH ต่อคาบทั้งหมด

$$D = \frac{t_{on}}{T} \times 100\%$$

โดยที่

D = Duty Cycle

t_{on} = เวลาที่เปิดใช้งาน

T = คาบ

ผลของ PWM ต่อมอเตอร์ DC คือควบคุมความเร็วได้แม่นยำ เพราะแรงดันเฉลี่ยเปลี่ยนตาม Duty Cycle แบบเชิงเส้น รักษาแรงบิดได้ดีที่รอบต่ำ เนื่องจากมอเตอร์ได้รับแรงดันเต็มช่วงที่เปิด ทำให้แรงบิดเริ่มต้นสูง และลดการสูญเสียพลังงาน ทรานซิสเตอร์สวิตช์ทำงานแบบเปิด- ปิด ไม่ทำงานในช่วงกลางเหมือนการลดแรงดันแบบเส้นตรง

ข้อดีของการใช้ PWM

1. ประสิทธิภาพสูง: เมื่อสวิตช์ (ทรานซิสเตอร์) ทำงานในโหมด PWM มันจะทำงานเป็นสวิตช์เปิด-ปิดเท่านั้น ทำให้สูญเสียพลังงานเป็นความร้อนน้อยมาก เมื่อเทียบกับการใช้ตัวต้านทานปรับค่าได้ เพื่อลดแรงดันไฟฟ้า ซึ่งจะทำให้เกิดความร้อนสูงและสิ้นเปลืองพลังงาน
2. การควบคุมเชิงดิจิทัลที่ง่าย: สามารถสร้างสัญญาณ PWM ได้ง่ายด้วยไมโครคอนโทรลเลอร์ หรือชิปเฉพาะทาง ทำให้การควบคุมมอเตอร์มีความยืดหยุ่นและแม่นยำสูง

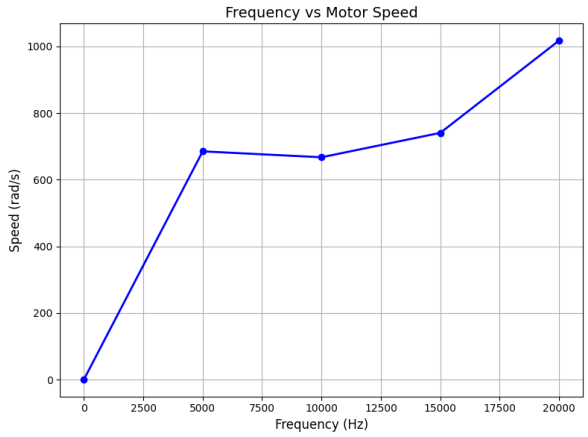
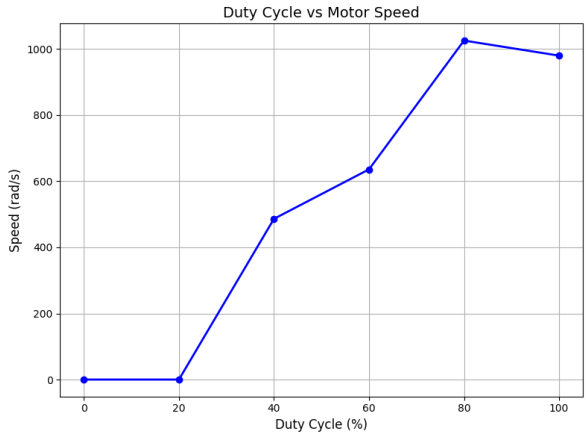
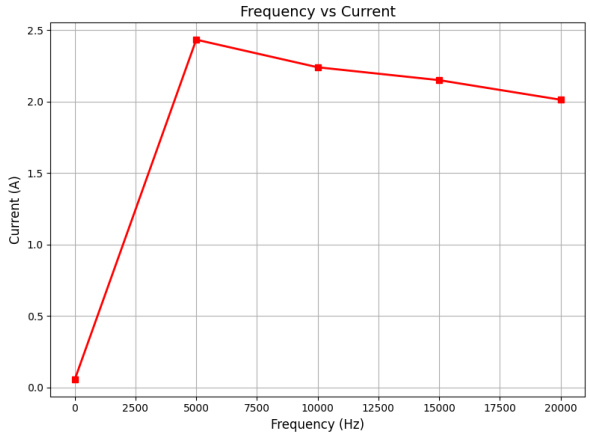
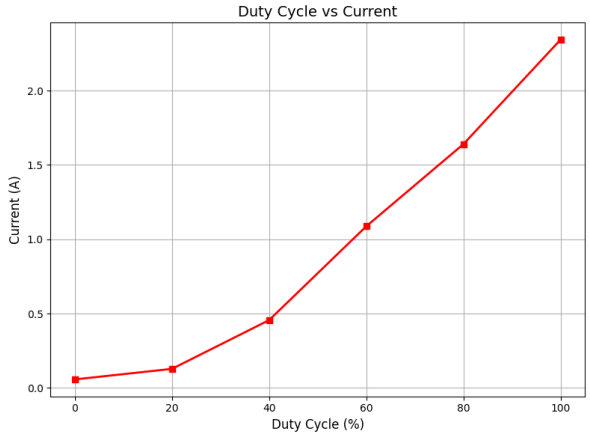
ขั้นตอนการดำเนินงาน

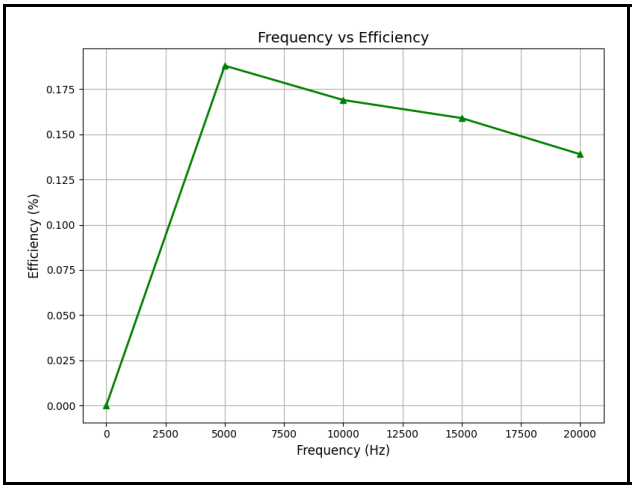
การทดลองหาความสัมพันธ์ระหว่างความเร็วมอเตอร์ กระแสไฟฟ้า ประสิทธิภาพมอเตอร์ และ PWM ทำการควบคุม Duty Cycle ไว้ที่ 100% จากนั้นทำการเก็บค่า Torque และกระแสไฟฟ้า เปลี่ยน PWM ที่ค่าต่างๆ ทำซ้ำทั้งหมด 3 ครั้ง สำหรับความเร็วจะแทนค่าในสมการของการทดลองข้างต้น และทำการสร้างกราฟเปรียบเทียบ

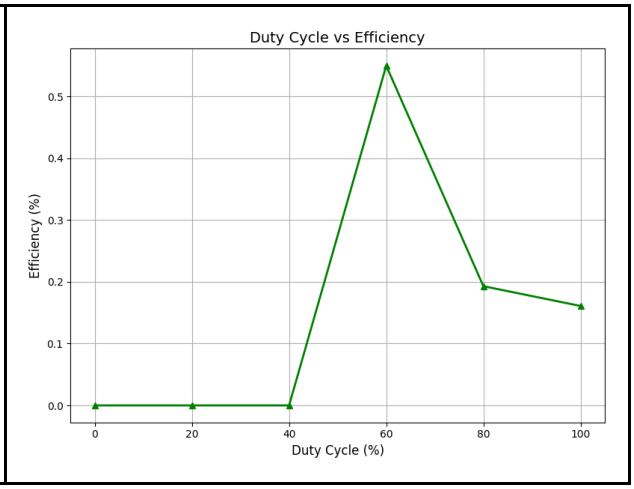
การหาทดลองหาความสัมพันธ์ระหว่างความเร็วมอเตอร์ กระแสไฟฟ้า ประสิทธิภาพมอเตอร์ และ Duty Cycle ทำการควบคุมความถี่ PWM ที่ 2000 Hz จากนั้นทำการ ปรับค่า Duty Cycle ไปที่ค่าต่างๆ จากนั้นทำการเก็บค่า Torque และกระแสไฟฟ้า เปลี่ยน PWM ที่ค่าต่างๆ ทำซ้ำทั้งหมด 3 ครั้ง สำหรับความเร็วจะแทนค่าในสมการของการทดลองข้างต้น และทำการสร้างกราฟเปรียบเทียบ

ผลการทดลอง

ตารางที่ 2 แสดงการเปรียบเทียบระหว่าง Frequency คงที่ และ Duty Cycle คงที่

ที่ Duty Cycle 100% เปลี่ยน PWM Frequency เป็นค่าต่าง ๆ (0 - 20000 Hz)	**ที่ PWM Frequency 2000 Hz เปลี่ยน Duty Cycle เป็นค่าต่าง ๆ (0-100%)**
กราฟแสดงความสัมพันธ์ระหว่าง PWM Frequency และความเร็วมอเตอร์	กราฟแสดงความสัมพันธ์ระหว่าง Duty Cycle และความเร็วมอเตอร์
	
กราฟแสดงความสัมพันธ์ระหว่าง PWM Frequency และกระแสไฟฟ้า	กราฟแสดงความสัมพันธ์ระหว่าง Duty Cycle และกระแสไฟฟ้า
	
กราฟแสดงความสัมพันธ์ระหว่าง PWM Frequency และประสิทธิภาพของมอเตอร์	กราฟแสดงความสัมพันธ์ระหว่าง Duty Cycle และประสิทธิภาพของมอเตอร์





สรุปผลการทดลอง

1. ผลการทดลองเมื่อเปลี่ยนค่า PWM Frequency (Duty Cycle คงที่ 100%)

กราฟความสัมพันธ์ระหว่าง Frequency กับ ความเร็ว, กระแส และ ประสิทธิภาพ:

ด้านความเร็ว (Speed): พบว่าความเร็วมีแนวโน้มเพิ่มขึ้นเมื่อความถี่เพิ่มขึ้น แต่ไม่ได้เป็นเส้นตรง (Linear) ตลอดช่วง สังเกตได้ว่าที่ช่วง 5,000 - 10,000 Hz ความเร็วค่อนข้างคงที่ ก่อนจะเพิ่มขึ้นอีกครั้งที่ความถี่สูง (20,000 Hz)

ด้านกระแสไฟฟ้า (Current): กระแสไฟฟ้าขึ้นไปแตะจุดสูงสุดที่ความถี่ 5,000 Hz (~2.4A) จากนั้นมีแนวโน้มลดลง เล็กน้อยเมื่อความถี่สูงขึ้น (เหลือ ~2.0A ที่ 20,000 Hz) ซึ่งเป็นผลมาจากค่าความต้านทานเชิงเหนี่ยวนำ (Inductive Reactance) ของขดลวดมอเตอร์ที่เพิ่มขึ้นตามความถี่ ทำให้กระแสไหลผ่านได้ยากขึ้น

ด้านประสิทธิภาพ (Efficiency): ประสิทธิภาพสูงสุดเกิดขึ้นที่ความถี่ 5,000 Hz (~0.19%) และค่อยๆ ลดลงเมื่อความถี่เพิ่มขึ้น ซึ่งสอดคล้องกับกราฟกระแสไฟฟ้าที่ลดลง

สรุป การเพิ่มความถี่ PWM ไม่ได้ทำให้ประสิทธิภาพดีขึ้นเสมอไป ในการทดลองนี้จุดทำงานที่เหมาะสมที่สุดคือ 5,000 Hz ซึ่งให้ประสิทธิภาพสูงสุด แม้ความเร็วจะเพิ่มขึ้นที่ความถี่สูงมากๆ แต่ประสิทธิภาพกลับลดลง เนื่องจากผลของ Inductive Reactance ที่ขัดขวางการไหลของกระแส

2. ผลการทดลองเมื่อเปลี่ยนค่า Duty Cycle (PWM Frequency คงที่ 2,000 Hz)

กราฟความสัมพันธ์ระหว่าง Duty Cycle กับ ความเร็ว, กระแส และ ประสิทธิภาพ:

ด้านความเร็ว:

- *ที่ช่วง 0-20% มอเตอร์ยังไม่หมุน (Dead zone) เนื่องจากแรงบิดไม่ชนะแรงเสียดทานสถิต*
- *ที่ช่วง 20-80% ความเร็วเพิ่มขึ้นเป็นเส้นตรง (Linear) ตามทฤษฎีการควบคุมแบบ PWM*
- *ที่ช่วง 80-100% ความเร็วเริ่มอิ่มตัวและตกลงเล็กน้อย*

ด้านกระแสไฟฟ้า (Current): กระแสไฟฟ้าเพิ่มขึ้นตามค่า Duty Cycle โดยช่วงแรกเพิ่มขึ้นช้าๆ และเริ่มพุ่งสูงขึ้นอย่างชัดเจนเมื่อ Duty Cycle เกิน 40% ไปจนถึงจุดสูงสุดที่ 100%

ด้านประสิทธิภาพ (Efficiency): กราฟมีลักษณะเป็นระฆังคว่ำ (Bell Shape)

- *ประสิทธิภาพเป็น 0 ในช่วงแรกที่มอเตอร์ไม่หมุน*
- *ประสิทธิภาพพุ่งขึ้นสูงสุดอย่างชัดเจนที่ Duty Cycle 60% (ค่าประมาณ 0.55%)*
- ประสิทธิภาพลดฮวบลงเมื่อ *Duty Cycle* เกิน *80%* แม้กระแสจะสูงมาก แต่ความเร็วไม่ได้เพิ่มขึ้นตามสัดส่วน ทำให้เกิดการสูญเสียในรูปความร้อน แทนที่จะเป็นงานทางกล

สรุป Duty Cycle สามารถควบคุมความเร็วได้ดีในช่วง Linear (20-80%) แต่ช่วงที่มอเตอร์ทำงานได้ มีประสิทธิภาพทางพลังงานดีที่สุดคือ Duty Cycle 60% การจ่าย Duty Cycle สูงถึง 100% แม้จะได้แรงบิด/กระแสสูง แต่ประสิทธิภาพทางกลกลับต่ำลงเนื่องจากการสูญเสียพลังงานในระบบที่สูงเกินความจำเป็น

อภิปรายผล

การควบคุมความเร็วด้วย PWM ให้ผลสอดคล้องกับทฤษฎี กราฟแสดงให้เห็นว่าความเร็วเพิ่มอย่างเกือบเชิงเส้นกับ Duty Cycle การเบี่ยงเบนเล็กน้อยในช่วง Duty ต่ำอาจเกิดจาก Friction เริ่มต้น, Back-EMF ต่ำ และ Ripple จาก PWM ที่ความถี่ไม่สูงพอ

กระแสเพิ่มขึ้นตาม Duty Cycle เพราะแรงบิดสูงขึ้น เมื่อความเร็วสูงขึ้น มอเตอร์ต้องเอาชนะแรงเสียดทาน และโหลดไดนามิกมากขึ้น ทำให้กระแสเพิ่มตามสมการ $T = K_t I$ สัญญาณกระแสบางครั้งมี ripple เนื่องจากมอเตอร์มี Inductance ไม่เพียงพอในการกรอง PWM หรือความถี่ PWM ต่ำเกินไป

ประสิทธิภาพสูงสุดไม่ใช่ที่ Duty สูงสุด ผลทดลองชี้ว่าประสิทธิภาพจะสูงที่สุดในช่วง Duty กลางเพราะ Loss ภายในต่ำ, กระแสไม่สูงเกินไป (ลด I^2R Loss) และ Back-EMF ของมอเตอร์อยู่ในช่วงทำงานที่เหมาะสม เมื่อ Duty ใกล้ 100% แม้มอเตอร์หมุนเร็วขึ้น แต่กระแสสูงทำให้ความสูญเสียเพิ่ม ส่งผลให้ประสิทธิภาพลดลง

ผลของ PWM Frequency จากการทดลองพบว่าความถี่ PWM มีผลต่อความเรียบของความเร็ว โดยความถี่ต่ำ จะเกิดความเร็วสั่น, Torque ripple มาก, เกิดเสียง ถ้าความถี่สูง จะเกิดความเร็วเรียบ และกระแสเนียนขึ้น แต่ทำให้ไดรเวอร์มี switching loss สูงขึ้น สิ่งนี้สอดคล้องกับทฤษฎี H-Bridge และ PWM ในงานควบคุมมอเตอร์ทั่วไป

ข้อเสนอแนะ

- หากโหลดไม่คงที่ จะมีผลต่อการวัดกระแสและประสิทธิภาพ
- ค่าเซนเซอร์หรือสัญญาณ *ADC* อาจมี *noise* ทำให้ต้องใช้ *filtering*

อ้างอิง

https://resources.pcb.cadence.com/blog/2020-pulse-width-modulation-characteristics-and-the-effects-of-frequency-and-duty-cycle?utm_source=chatgpt.com

การทดลองที่ 3 การเปรียบเทียบโหมด H - Bridge (Sign-Magnitude vs Locked Anti Phase)

จุดประสงค์

1. เพื่อเปรียบเทียบนิสัยการทำงานของมอเตอร์ภายใต้โหมด Sign-Magnitude และ Locked Anti-Phase โดยวัดพฤติกรรมของความเร็ว (RPM), กระแส (A) และ ripple ของความเร็ว/แรงบิด
2. เพื่อประเมินผลกระทบของแต่ละโหมดต่อประสิทธิภาพการใช้พลังงาน (power loss) และการเกิดความร้อนที่ไดรเวอร์ และมอเตอร์
3. เพื่อสรุปข้อดี–ข้อเสีย และแนะนำโหมดการขับที่เหมาะสมสำหรับงานใช้งานต่าง ๆ

สมมติฐาน

Locked Anti-Phase จะให้การตอบสนองที่เรียบและลด speed ripple ได้ดีกว่า Sign-Magnitude โดยเฉพาะที่ความเร็วต่ำและ duty ใกล้จุดเปลี่ยนทิศทาง แต่จะมี switching loss และความร้อนที่สูงกว่า

ตัวแปร

1. ตัวแปรต้น:
- โหมด *H - Bridge*
2. *ตัวแปรตาม:*
- ลักษณะกระแสระหว่างทำงาน
- ความเร็วเชิงมุม
3. *ตัวแปรควบคุม:*
- แรงดันไฟฟ้าที่จ่ายเข้ามา
- กระแส
- *PWM 2000 Hz*

เอกสารและงานวิจัยที่เกี่ยวข้อง

หลักการทำงานของ H-Bridge คือ วงจรอิเล็กทรอนิกส์ที่ทำหน้าที่เป็นสวิตช์ควบคุมการไหลของกระแสไฟฟ้าไปยังอุปกรณ์โหลด (เช่น มอเตอร์กระแสตรง หรือ DC Motor) ให้สามารถ สลับขั้วแรงดัน ได้อย่างง่าย ทำให้สามารถควบคุม ทิศทางการหมุน และ ความเร็ว ของมอเตอร์ได้

คำว่า H-Bridge มาจากลักษณะการจัดเรียงของสวิตช์ 4 ตัว ที่เชื่อมต่อกันในรูปตัว H โดยมีมอเตอร์อยู่ที่กึ่งกลาง การควบคุมความเร็วของมอเตอร์กระแสตรงที่ต่อกับ H-Bridge ทำได้โดยใช้ PWM

โหมดควบคุมมอเตอร์ (Sign-Magnitude และ Locked Anti-Phase) โหมดเหล่านี้กำหนดรูปแบบการเปิด-ปิดสวิตช์ทรานซิสเตอร์ ทั้งสี่ตัวในวงจร H-Bridge เพื่อควบคุมแรงดันเฉลี่ยที่จ่ายให้กับมอเตอร์ ซึ่งส่งผลต่อความเร็วและทิศทางของมอเตอร์

1. โหมด Sign-Magnitude

โหมดนี้ใช้สัญญาณควบคุม 2 สัญญาณ แยกกัน คือ Sign (สัญญาณทิศทาง): สัญญาณดิจิทัล (HIGH/LOW) สำหรับกำหนดทิศทางการหมุน (เดินหน้า/ถอยหลัง) และMagnitude (ขนาด): สัญญาณ PWM สำหรับกำหนดขนาดของแรงดันเฉลี่ย (ความเร็ว) หลักการทำงาน คือ

1. การกำหนดทิศทาง: วงจร *H-Bridge* จะเปิดสวิตช์คู่หนึ่งไว้ คงที่ เพื่อกำหนดทิศทางการไหลของกระแส
2. การกำหนดความเร็ว: ใช้สัญญาณ *PWM* ไปขับสวิตช์อีกคู่หนึ่งให้ สลับเปิด-ปิด (*switching)*

ช่วง On-time (PWM HIGH): สวิตช์ทั้งสี่ตัวอยู่ในสถานะที่ทำให้กระแสไหลผ่านมอเตอร์ไปยังทิศทางที่ต้องการ
ช่วง Off-time (PWM LOW): มอเตอร์จะถูก ลัดวงจร (short-circuited) โดยการเปิดสวิตช์ฝั่งล่างหรือฝั่งบนคู่หนึ่ง ทำให้กระแสไหลวนกลับผ่านมอเตอร์ ซึ่งเป็นการเบรกแบบ Passive Braking หรือ Coast
ข้อดี และข้อสังเกต

- ความซับซ้อนในการควบคุม: ต้องใช้สัญญาณควบคุม 2 เส้น (PWM และ DIR)
- การสูญเสียกำลังงานจากการสวิตช์ (Switching Loss) ต่ำกว่า Locked Anti-Phase มาก เนื่องจากมีสวิตช์เพียง 1-2 ตัวเท่านั้นที่สลับเปิด-ปิดด้วยความถี่ PWM

2. โหมด Locked Anti-Phase
โหมดนี้เป็นเทคนิคที่ใช้สัญญาณควบคุม เพียง 1 สัญญาณ คือ สัญญาณ PWM เพียงเส้นเดียวในการควบคุมทั้งทิศทางและความเร็วพร้อมกัน หลักการทำงาน คือ

1. การควบคุมด้วย PWM 1 เส้น: สัญญาณ PWM จะถูกส่งไปควบคุมสวิตช์คู่หนึ่ง และสัญญาณ PWM ตรงกันข้าม จะถูกส่งไปควบคุมสวิตช์อีกคู่หนึ่งพร้อมกัน
2. จุดศูนย์กลาง (Neutral Point):
    - Duty Cycle 50%: แรงดันเฉลี่ยที่มอเตอร์ได้เท่ากับ 0 โวลต์ มอเตอร์จะหยุด
    - Duty Cycle > 50%: มอเตอร์หมุนไปทิศทางหนึ่ง (เช่น เดินหน้า)
    - Duty Cycle < 50%: มอเตอร์หมุนไปทิศทางตรงกันข้าม (เช่น ถอยหลัง)
3. การเบรก: โหมดนี้สามารถทำ Regenerative Braking (เบรกแบบสร้างพลังงานกลับ) ได้โดยธรรมชาติ เนื่องจากวงจรจะสลับขั้วอย่างรวดเร็วตลอดเวลา

ข้อดีและข้อสังเกต

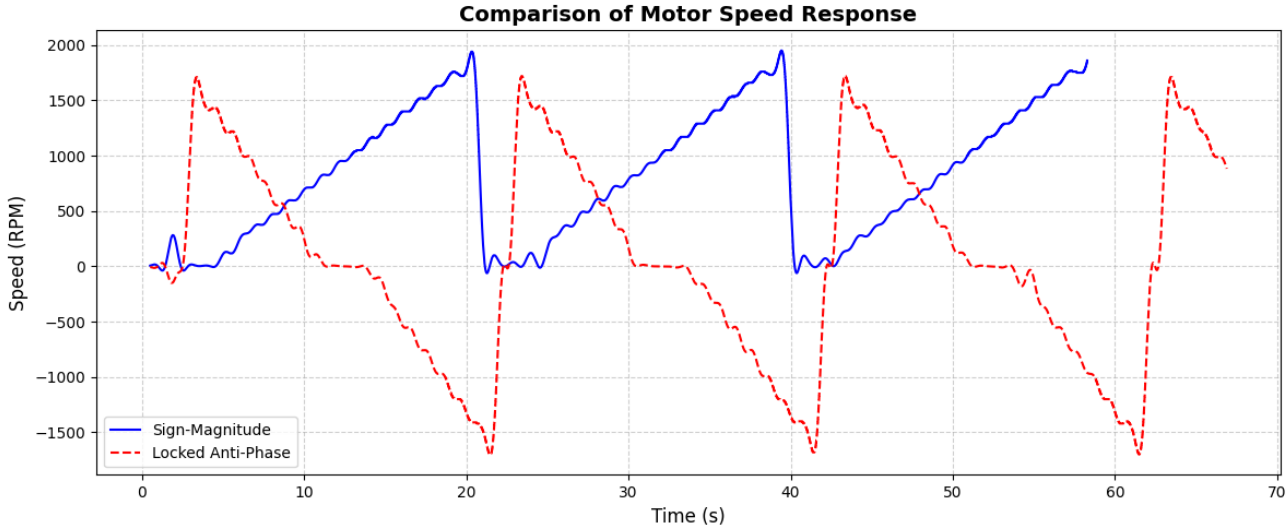
- ความซับซ้อนในการควบคุม: ใช้สัญญาณควบคุมเพียง 1 เส้น ทำให้การเดินสายง่ายขึ้น
- การสูญเสียกำลังงานจากการสวิตช์: สูงกว่า Sign-Magnitude มาก เนื่องจาก สวิตช์ทั้งสี่ตัว จะสลับเปิด-ปิดด้วยความถี่ PWM ทุกรอบการทำงาน

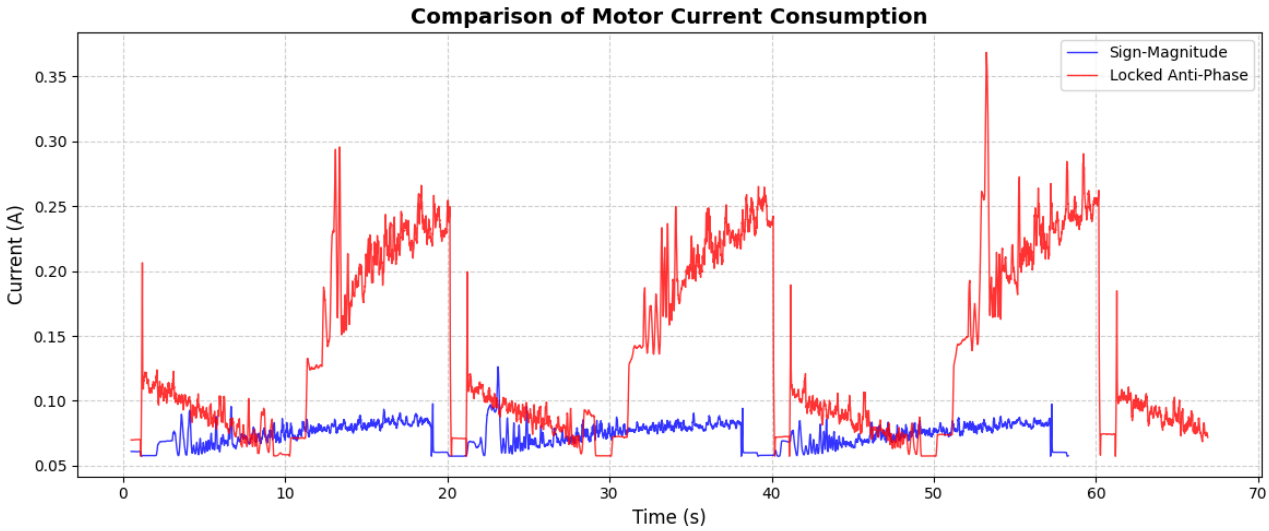
ขั้นตอนการดำเนินงาน

การทดลองหาความสัมพันธ์ระหว่างโหมด H – Bridge กับ RPM และ ทิศทางที่จะเกิดขึ้น โดยทำการเปลี่ยน Duty Cycle ที่ค่าต่างๆทำซ้ำทั้งหมด 3ครั้ง โดยวิธีการปรับเปลี่ยนโหมดจะต้องทำสลับสายไฟ ที่ PWM กับ DIR ของ Motor drive

ผลการทดลอง

รูปที่ 4 แสดงกราฟความสัมพันธ์ RPM และ Duty Cycle และแสดงกราฟ Current





สรุปผลการทดลอง

จากการทดลองขับมอเตอร์กระแสตรงด้วย H-Bridge แบบสองโหมด คือ Sign-Magnitude และ Locked Anti-Phase โดยใช้ค่า PWM เดียวกัน พบผลสรุปดังนี้

1. ความเร็วเชิงมุม (Angular Speed)

โหมด Sign-Magnitude ให้ความเร็วเฉลี่ยสูงกว่าในทุกระดับ Duty Cycle เนื่องจากแรงดันเฉลี่ย ที่ส่งให้มอเตอร์มีขนาดใกล้เคียงกับ $V_{avg} = D \cdot V_s$

โหมด Locked Anti-Phase ให้ความเร็วต่ำกว่า เนื่องจากแรงดันที่มอเตอร์ได้รับสลับขั้วที่ความถี่ PWM ทำให้แรงดันเฉลี่ยเป็น $V_{avg} = (2D - 1)V_s$ ส่งผลให้ที่ Duty เท่ากัน มอเตอร์เห็นแรงดันเฉลี่ยที่น้อยกว่า

2. กระแสไฟฟ้า (Current)

กระแสเฉลี่ยในโหมด Locked Anti-Phase สูงกว่าในระดับ Duty ต่ำถึงปานกลาง เนื่องจากการสลับขั้ว ±Vs ทำให้เกิด torque ripple และ current ripple สูงกว่า

โหมด Sign-Magnitude ใช้กระแสน้อยกว่าในช่วงเดียวกัน เพราะการสลับเพียง 0 ↔ +Vs ทำให้มอเตอร์รับแรงดันเป็นทิศเดียว

3. ความเรียบของการหมุน (Smoothness)

ความเร็วและแรงบิดของโหมด Sign-Magnitude มีความเรียบกว่าโหมด Locked Anti-Phase มี ripple สูง ทั้งกระแสและความเร็ว โดยสังเกตได้จากการแกว่งของค่า RPM

4. ประสิทธิภาพโดยรวม

การทดลองพบว่าที่ Duty เดียวกัน โหมด Sign-Magnitude มีประสิทธิภาพดีกว่า โหมด Locked Anti-Phase สูญเสียเพิ่มจาก switching loss และ torque ripple

อภิปรายผล

ความเร็ว และแรงดันเฉลี่ยสัมพันธ์ตามทฤษฎีทั้งสองโหมด ความเร็วที่แตกต่างกันชัดเจนของสองโหมดสอดคล้องกับสมการแรงดันเฉลี่ย จึงทำให้ที่ Duty เดียวกัน ความเร็วของโหมด Locked Anti-Phase ต่ำกว่าอย่างเป็นระบบ

Ripple สูงใน Locked Anti-Phase ทำให้กินกระแสมากกว่า เนื่องจากสัญญาณที่มอเตอร์รับเป็นรูปแบบ +Vs, -Vs ที่ความถี่ PWM ทำให้กระแสเปลี่ยนทิศทางตลอดเวลา ส่งผลให้ I_{ripple} สูง และ I_{RMS} สูงกว่าของ Sign-Magnitude แม้ I_{avg} ใกล้เคียงกัน แต่ I_{RMS} สูงขึ้นทำให้เกิดความสูญเสีย I^2R มากกว่า จึงอธิบายได้ว่าทำไมโหมดนี้ใช้กระแสสูงกว่าแม้ความเร็วน้อยกว่า

Sign-Magnitude ให้พฤติกรรมการควบคุมความเร็วที่เสถียรกว่า ผลการทดลองพบว่า ความเร็วของ Sign-Magnitude เพิ่มขึ้นเป็นเส้นตรงตาม Duty, Locked Anti-Phase เพิ่มขึ้นไม่เป็นเส้นตรงในช่วง Duty 0–50% และทิศทางแรงดันถูกสลับเร็วเกิน ทำให้ประสิทธิภาพแรงบิดเฉลี่ยลดลง สอดคล้องกับทฤษฎีว่ามอเตอร์ DC ไม่ตอบสนองทันต่อการสลับทิศทางความถี่สูงของแรงดันใน Anti-Phase

ความสูญเสีย (Losses) ใน Anti-Phase สูงกว่า มี 2 สาเหตุหลักได้แก่ Switching Loss เพิ่มขึ้น เพราะ MOSFET ต้องสลับทิศทางเต็ม +Vs และ -Vs ตลอดเวลา และ Torque Ripple เพิ่มขึ้น ทำให้ต้องใช้กระแสมากขึ้นเพื่อรักษาความเร็วเฉลี่ย จึงทำให้ประสิทธิภาพโดยรวมต่ำกว่า

ข้อเสนอแนะ

- ความร้อนของ *H-Bridge* อาจส่งผลต่อการสูญเสียภายใน
- ถ้าความถี่ *PWM* ต่ำเกิน รูปคลื่นกระแสจะมี *ripple* สูงกว่าที่ควร

อ้างอิง

https://www.ti.com/lit/an/slva504a/slva504a.pdf?ts=1763737261008&ref_url=https%253A%252F%252Fwww.google.com%252F

https://forum.arduino.cc/t/loss-of-power-from-h-bridge/96694

2. Stepper Motor

การทดลองที่ 1 การหาความสัมพันธ์ระหว่างความถี่ และโหมด (Frequency vs. Speed & Drive Modes)

จุดประสงค์

1. เพื่อศึกษาความสัมพันธ์ระหว่างความถี่สัญญาณอินพุต กับความเร็วรอบของ Stepper Motor โดยวัดและเปรียบเทียบค่าความเร็วที่ได้จากการทดลองจริง
2. เพื่อวิเคราะห์ผลของการเปลี่ยน Drive Mode
3. เพื่อเปรียบเทียบความละเอียดเชิงมุมของแต่ละ Drive Mode และผลที่เกิดกับความต่อเนื่อง และแรงสั่นสะเทือนของมอเตอร์

สมมติฐาน

ที่ความถี่ Input เท่ากัน โหมดที่มีความละเอียดสูง (Microstep) จะหมุนช้ากว่าโหมดความละเอียดต่ำ (Full Step) ตามสัดส่วนของการแบ่ง Step และความเร็วรอบมอเตอร์ (RPM) จะแปรผันตรงกับความถี่ Input (Pulse Frequency) จนถึงจุดหนึ่งที่จะเกิด Loss Step

ตัวแปร

1. ตัวแปรต้น:
- ความถี่สัญญาณ Pulse
- โหมดการทำงาน (Full, Half, 1/4Step)
2. ตัวแปรตาม:
- ความเร็วรอบจริงของมอเตอร์
- Accelelation
- Loss step
3. ตัวแปรควบคุม:
- แรงดันไฟเลี้ยง
- PWM

เอกสารและงานวิจัยที่เกี่ยวข้อง

Stepper Motor เป็นมอเตอร์กระแสตรงที่ออกแบบให้หมุนแบบเป็นขั้น (step) โดยแต่ละพัลส์สัญญาณไฟฟ้าที่ส่งเข้าไปยังขดลวดของ Stator จะทำให้ Rotor หมุนไปเป็นมุมคงที่ตามค่า Step Angle ที่ถูกกำหนดไว้ เช่น 1.8° ต่อ 1 สเต็ป หรือ 0.9° ต่อ 1 สเต็ป การหมุนแบบเป็นขั้นเช่นนี้ช่วยให้สามารถกำหนดตำแหน่งได้แม่นยำ โดยไม่จำเป็นต้องใช้ระบบป้อนกลับ (open-loop control) ในงานทั่วไป

การทำงานของ Stepper Motor เกิดจากการสร้างสนามแม่เหล็กหมุนภายใน Stator ตามลำดับของสัญญาณขับ (step sequence) ซึ่งดึงให้ Rotor เคลื่อนที่ตามสนามอย่างเป็นจังหวะ หากต้องการมุมละเอียด

ขึ้นสามารถใช้เทคนิค Microstepping ที่กำหนดกระแสในแต่ละเฟสเป็นสัดส่วนเชิงไซน์–โคไซน์ ทำให้ Rotor เคลื่อนที่อย่างต่อเนื่องและลดการสั่นสะเทือนลง แรงบิดของ Stepper Motor มีความสัมพันธ์โดยตรงกับกระแสของขดลวด และลดลงเมื่อความเร็วรอบเพิ่มสูงขึ้น ทำให้ Stepper Motor มีช่วงความเร็วที่เหมาะสม ซึ่งต้องออกแบบให้เหมาะกับการใช้งานเพื่อหลีกเลี่ยงปัญหา Loss Step ที่เกิดเมื่อมอเตอร์รับภาระหรือความเร่งสูงเกินขีดความสามารถ Stepper Motor แบ่งได้เป็น 3 ประเภทหลัก ได้แก่:

ตารางที่ 4 แสดงลักษณะ การทำงาน และคุณสมบัติของ Stepper Motor แต่ละประเภท

ประเภท	ลักษณะ	การทำงาน	คุณสมบัติ
Permanent Magnet Stepper	โรเตอร์ทำจากวัสดุแม่เหล็กถาวรที่มีขั้วแม่เหล็กชัดเจน	เมื่อสเตเตอร์มีพลังงาน โรเตอร์จะเรียงตัวตามสนามแม่เหล็กที่สร้างขึ้น	ให้แรงบิด (Torque) สูง และมีมุมสเต็ปที่ค่อนข้างใหญ่ (เช่น 7.5° หรือ 15°)
Variable Reluctance Stepper	โรเตอร์ทำจากเหล็กอ่อน (Soft Iron) ที่มีฟัน (Teeth) และไม่มีแม่เหล็กถาวร	การหมุนเกิดขึ้นจากหลักการที่โรเตอร์จะพยายามหมุนไปยังตำแหน่งที่ให้ Reluctance หรือความต้านทานต่อฟลักซ์แม่เหล็กน้อยที่สุด	โครงสร้างง่าย และมีความเฉื่อยต่ำ แต่มีแรงบิดต่ำ และต้องมีพลังงานไฟฟ้าจ่ายค้างไว้เพื่อคงตำแหน่ง
Hybrid Stepper	เป็นการรวมกันของคุณสมบัติของสองประเภทข้างต้น โดยโรเตอร์มีแม่เหล็กถาวร และมีฟันเฟืองขนาดเล็ก	ใช้หลักการดึงดูดของแม่เหล็กถาวรร่วมกับหลักการรีลักแทนซ์	เนื่องจากให้แรงบิดสูง และมีมุมสเต็ปที่เล็กมาก (เช่น 1.8° หรือ 0.9°) ซึ่งหมายถึงความละเอียดในการควบคุมตำแหน่งสูง

หลักการขับเคลื่อนในโหมดต่าง ๆ จะถูกขับโดยการเปิด–ปิดขดลวดแต่ละเฟสตามลำดับ ซึ่งเกิดรูปแบบการหมุนที่แตกต่างกัน โดยโหมดที่ใช้บ่อยมีดังนี้:

ตารางที่ 5 แสดงการทำงาน และลักษณะเด่นของโหมดการขับเคลื่อ Stepper Motor แต่ละประเภท

โหมด	การทำงาน	ลักษณะเด่น
Wave Drive	เป็นการขับเคลื่อนที่ง่ายที่สุด โดยมีการจ่ายพลังงานให้ ขดลวดเพียง 1 ชุดต่อครั้ง (One Phase ON) และสลับไปยังชุดถัดไป ตัวอย่างลำดับการขับ: $A \longrightarrow B \longrightarrow \bar{A} \longrightarrow \bar{B} \longrightarrow \ldots$	-ใช้พลังงานน้อยที่สุด -แรงบิดต่ำที่สุด -ความละเอียดเท่ากับจำนวนสเต็ปพื้นฐานของมอเตอร์ -เหมาะกับงานที่ไม่ต้องการแรงบิดมาก
Full Step Drive	มีการจ่ายพลังงานให้ ขดลวด 2 ชุดพร้อมกันต่อครั้ง ตัวอย่างลำดับการขับ: $AB \longrightarrow B\bar{A} \longrightarrow \bar{A}\bar{B} \longrightarrow A\bar{B} \longrightarrow \ldots$	-แรงบิดสูงกว่า Wave Drive (ประมาณ 1.4 เท่า) -ความแม่นยำสูงขึ้นเนื่องจากมีแรงดึงโรเตอร์มากกว่า -เป็นโหมดที่นิยมมากในงานทั่วไป
Half Step Drive	เป็นการรวมโหมด Wave Drive และ Full Step เข้าด้วยกัน โดยมีการสลับระหว่างการเปิด 1 ขดลวด และการเปิด 2 ขดลวด ตัวอย่างลำดับ:	-ความละเอียดเพิ่มเป็น 2 เท่า เช่น 1.8° ⟶ 0.9° -การเคลื่อนที่นุ่มนวลกว่าการขับแบบ Full Step

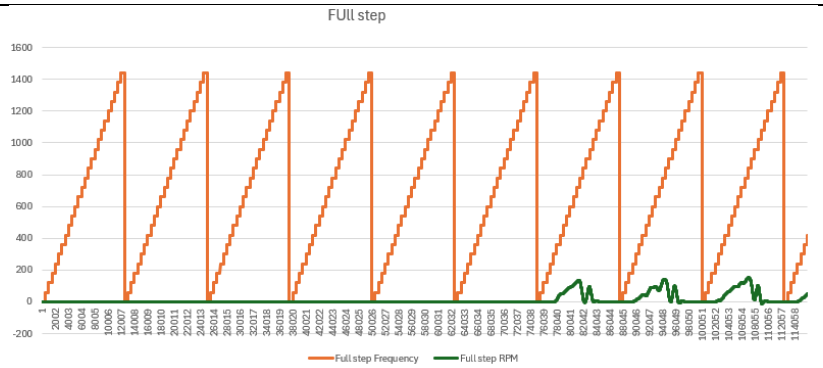
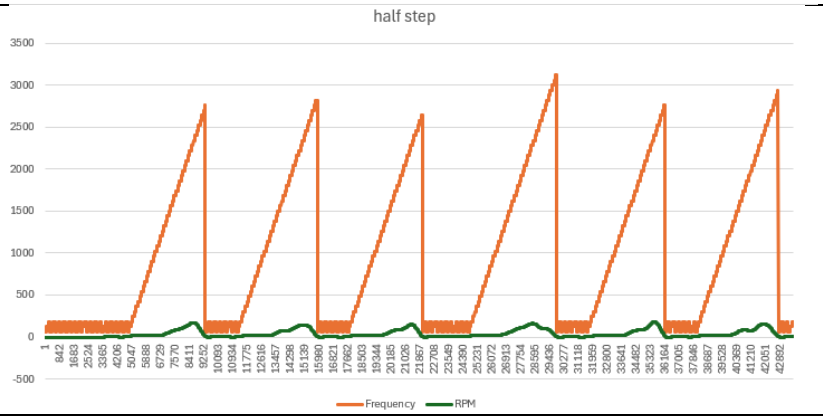
	$A \longrightarrow AB \longrightarrow B \longrightarrow B\bar{A} \longrightarrow \bar{A} \longrightarrow \bar{A}\bar{B} \longrightarrow \bar{B} \longrightarrow \bar{B}A \longrightarrow \ldots$	-แต่แรงบิดในสเตปที่เปิด 1 เฟสจะต่ำกว่า จึงมี torque ripple บ้าง
Microstepping	เป็นการขับเคลื่อนที่ซับซ้อนที่สุด โดยใช้เทคนิคการควบคุมกระแสไฟฟ้าแบบ PWM หรือการควบคุมแรงดันไฟฟ้าแบบอนาล็อก เพื่อปรับระดับกระแสในขดลวดทั้งสองอย่างต่อเนื่อง ทำให้สนามแม่เหล็กหมุนอย่างราบรื่น ตัวอย่างกระแส: เฟส A = $I\sin(\theta)$ เฟส B = $I\cos(\theta)$	-ความละเอียดสูงมาก: 1/4, 1/8, 1/16, 1/32 step -การเคลื่อนที่เรียบที่สุด และเสียงเงียบที่สุด -ลดการสั่น และresonance ของ stepper ได้ดี -แต่แรงบิดสูงสุดลดลงเล็กน้อย เพราะไม่มีจังหวะที่กระแสสูงสุดทั้งสองเฟสพร้อมกัน

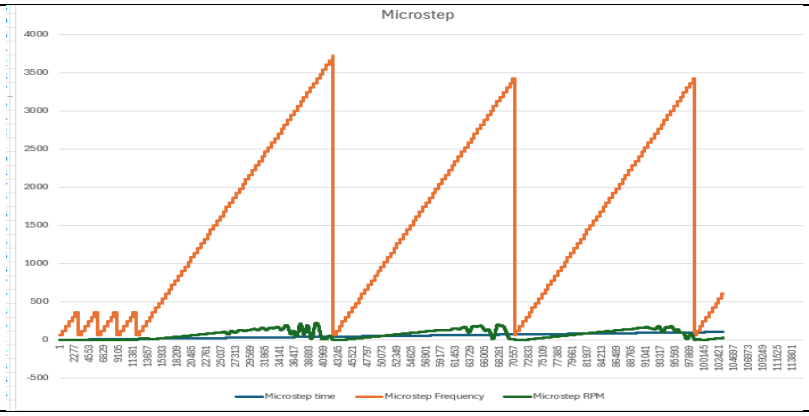
ขั้นตอนการดำเนินงาน

การทดลองหาความสัมพันธ์ระหว่างโหมดการทำงานของ Stepper Motor และ ความเร็วที่ได้จาก Stepper motor ทำการควบคุมโหมดการทำงานไว้ที่โหมดต่าง หลังจากนั้นทำการเพิ่ม Frequency ไปเรื่อยๆ และทำการเก็บค่า RPM และ Frequency ทำซ้ำ 3 ครั้ง

ผลการทดลอง

ตารางที่ 5 แสดงความสัมพันธ์ระหว่าง Frequency และ RPM

โหมด	กราฟความสัมพันธ์
Full step	
Half step	

Microstepping	

สรุปผลการทดลอง

ตารางที่ 6 ผลการทดลองของคุณสมบัติ แต่ละโหมด

คุณสมบัติ	Full Step Drive	Half Step Drive	Microstep Drive
การจ่ายไฟ	จ่ายไฟให้ขดลวด 2 ชุด พร้อมกัน (สลับขั้ว)	สลับระหว่างการจ่ายไฟ 1 ชุด และ 2 ชุด	จ่ายไฟให้ขดลวด 2 ชุดพร้อมกัน แต่ปรับ ระดับกระแส อย่างต่อเนื่อง
แรงบิดสูงสุด (Pull-Out Torque)	สูงสุด	สูงกว่า Microstep แต่ ต่ำกว่า Full Step เล็กน้อย	ต่ำที่สุด เมื่อเทียบกับ Full Step และ Half Step
ความราบรื่น (Smoothness)	ต่ำสุด (การหมุนเป็นจังหวะชัดเจน)	ปานกลาง (การสั่นสะเทือนลดลง)	สูงสุด (การหมุนราบรื่นมากที่สุด)
การนำไปใช้งาน	ต้องการ แรงบิดสูง และความเร็วสูงสุด (เช่น งานขนย้ายที่รวดเร็ว)	ต้องการ ความละเอียดปานกลาง และการทำงานที่ราบรื่นกว่า Full Step	ต้องการ ความแม่นยำสูง ในการกำหนดตำแหน่ง และการทำงานที่เงียบ (เช่น เครื่องพิมพ์ 3D)

อภิปรายผล

ตารางที่ 7 สรุปข้อดี-ข้อเสีย และการใช้งาน แต่ละโหมด

โหมด	ข้อดี	ข้อเสีย	การใช้งาน
Full step	- ให้แรงบิดสูงที่สุดในบรรดาโหมดพื้นฐาน - ความถี่ตอบสนองสูง รองรับความเร็วได้ดี - ควบคุมง่าย สัญญาณไม่ซับซ้อน - ประสิทธิภาพแรงบิดดี	- การเคลื่อนที่ไม่เรียบมาก เกิด vibration หรือ resonance ได้ - ตำแหน่งละเอียดเท่า Step Base (1.8° หรือ 0.9°) ไม่ละเอียดเท่า Microstep	ระบบที่ต้องการแรงบิดสูง เช่น CNC axis ขนาดเล็ก, หุ่นยนต์, ระบบยก โหลดทั่วไป

Half step	- ความละเอียดเพิ่มขึ้น 2 เท่า - การเคลื่อนที่นุ่มกว่า Full Step - แรงบิดยังคงระดับปานกลาง–สูง	- แรงบิดไม่คงที่ทุกสเตป (สลับ 1-phase กับ 2-phase) ทำให้เกิด torque ripple เล็กน้อย - ความเร็วสูงสุดต่ำกว่า Full Step เล็กน้อย	งานที่ต้องการความละเอียดเพิ่มขึ้นโดยยังต้องการแรงบิดดี เช่น robot arm, positioning system
Microstepping	- ความละเอียดสูงมาก (1/4, 1/8, 1/16, 1/32 step ฯลฯ) - การเคลื่อนที่นิ่งที่สุด และเสียงเงียบที่สุด - ลด resonance ได้ดีมาก - ควบคุมความเร่งได้ละเอียด	- แรงบิดสูงสุดลดลง (โดยเฉพาะเมื่อแบ่งไมโครสเตปมากเกิน) - ความเร็วสูงสุดต่ำลงเพราะต้องใช้ความถี่สเตปสูงขึ้นมาก - ต้องใช้ไดรเวอร์ที่ซับซ้อนและราคาสูง	งานที่ต้องการความละเอียดและความนุ่มนวลสูง เช่น เครื่องพิมพ์ 3D, กล้อง PTZ, ระบบ precise positioning

ความเร็วรอบกับโหมดการขับ: ผลการทดลองพบว่า Microstepping ทำความเร็วรอบได้สูงสุด (226.86 RPM) รองลงมาคือ Half Step และ Full Step ตามลำดับ สาเหตุที่ Full Step ทำความเร็วได้ต่ำสุดเพราะเกิดการสั่นรุนแรงจากการเปลี่ยนสเต็ปที่กระชาก ทำให้มอเตอร์หลุดสเต็ป (Stall) ตั้งแต่ความเร็วต่ำ ต่างจาก Microstepping ที่มีความ Smooth กว่าจึงไต่ความเร็วได้สูงกว่า

ขีดจำกัดแรงบิด: กราฟ Half Step แสดงลักษณะความเร็วเพิ่มขึ้นแล้วตกลง เกิดจากธรรมชาติของมอเตอร์ที่แรงบิดลดลงเมื่อความถี่สูงขึ้น เมื่อแรงบิดลดต่ำกว่าแรงเสียดทาน มอเตอร์จึงหยุดหมุนทันที

ข้อเสนอแนะ

- ควรมีการติดตั้งมอเตอร์อย่างแน่นหนาหรือใช้วัสดุดูดซับการสั่นสะเทือน เพื่อลดการสั่นรุนแรงที่เกิดขึ้นในโหมด Full Step ซึ่งอาจส่งผลให้เกิด Loss Step ได้ตั้งแต่ความเร็วต่ำ
- ควรกำหนดจุด Maximum Start/Stop Frequency ของแต่ละโหมดอย่างชัดเจน โดยวัดความถี่สูงสุดที่มอเตอร์สามารถเริ่มหมุนและหยุดหมุนได้ทันทีโดยไม่เกิด Loss Step

อ้างอิง

https://www.monolithicpower.com/en/learning/resources/stepper-motors-basics-types-uses?srsltid=AfmBOoogagNpoVLQTfHSeP0FsNwKfFsXKj13nqv4ydz0s_ZUnuO2LlUI

https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-basics.html

https://www.omega.co.uk/prodinfo/stepper_motors.html

การทดลองที่ 2 การวิเคราะห์ Loss Step (Loss Step Analysis)

จุดประสงค์

1. เพื่อหาจุดสูงสุดที่มอเตอร์สามารถทำงานได้โดยไม่เกิดความผิดพลาด (Maximum Start/Stop Frequency) และวิเคราะห์สภาวะที่เกิด Loss Step

สมมติฐาน

การป้อนความถี่ที่สูงเกินกว่าแรงบิดของมอเตอร์จะเอาชนะความเฉื่อยได้ จะทำให้เกิดการ Loss Step (ตำแหน่งจริงไม่ตรงกับตำแหน่งสั่งการ)

ตัวแปร

1. ตัวแปรต้น:
- ความถี่สัญญาณ Pulse
- โหมดการทำงาน (Full, Half, 1/4Step)
2. ตัวแปรตาม:
- ความคลาดเคลื่อนของตำแหน่ง
3. ตัวแปรควบคุม:
- แรงดันไฟเลี้ยง
- Drive Mode (เลือกใช้ Full Step เพื่อให้เห็นผลชัดเจนที่สุด)

เอกสารและงานวิจัยที่เกี่ยวข้อง

Loss Step คือปรากฏการณ์ที่ Stepper Motor ไม่สามารถหมุนตามจำนวน Step ที่สั่งได้ ทำให้ตำแหน่งจริงของมอเตอร์คลาดเคลื่อนจากตำแหน่งที่ควรจะเป็น ความคลาดเคลื่อนนี้เกิดขึ้นเมื่อแรงบิดที่มอเตอร์สร้างขึ้นไม่เพียงพอต่อการเอาชนะแรงเฉื่อยหรือแรงต้านของโหลด ส่งผลให้โรเตอร์ไม่สามารถขยับไปตำแหน่งถัดไปตามสัญญาณ Pulse ที่ได้รับ แม้ว่าสัญญาณยังคงถูกส่งเข้ามอเตอร์อย่างต่อเนื่อง มอเตอร์ก็จะตามไม่ทันเส้นคำสั่ง

การอ่าน Loss Step ทำโดยเปรียบเทียบตำแหน่งสั่ง กับตำแหน่งจริงที่ได้จาก encoder โดย Loss Steps = N_cmd – N_meas

เงื่อนไขที่ทำให้เกิด Loss Step

1. ความถี่สัญญาณ Pulse สูงเกินไป เมื่อความถี่สูงกว่าแรงบิดที่มอเตอร์สามารถจ่ายได้ในช่วงความเร็วนั้น มอเตอร์จะไม่สามารถก้าวตามสัญญาณที่สั่งได้ทัน
2. โหลดมีแรงเฉื่อยสูงหรือมีแรงเสียดทานมาก โหลดที่หนักหรือมีความเฉื่อยมากต้องการแรงบิดสูงขึ้นในการเร่ง หากแรงบิดไม่พอจะเกิด Loss Step ง่ายขึ้น
3. เริ่มต้นหมุนที่ความถี่สูงโดยไม่มีการเร่ง เริ่มสั่งความถี่สูงทันทีจากสถานะหยุด (static) ทำให้กระแสในขดลวดไม่มีเวลาสร้างแรงบิดเต็มที่

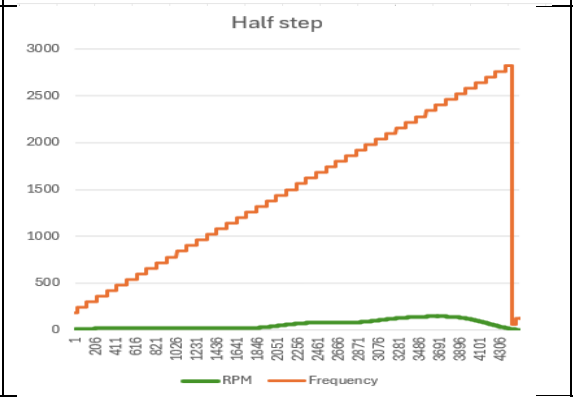
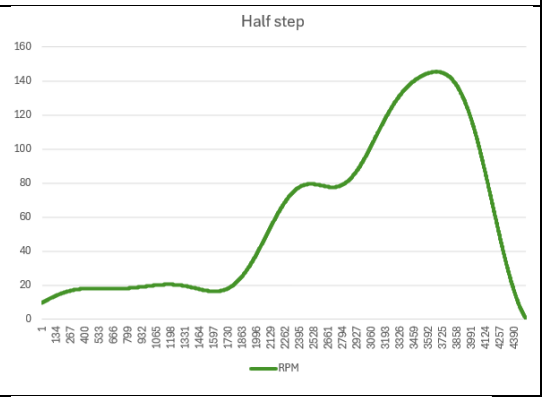
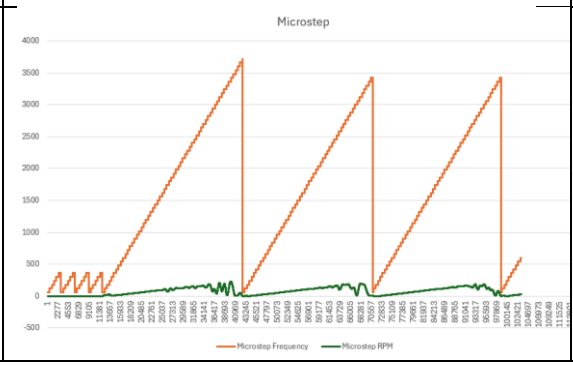
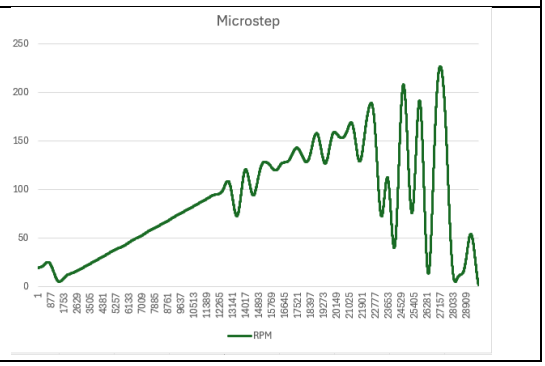
4. แรงบิดของมอเตอร์ลดลงจากข้อจำกัดเชิงไฟฟ้า เช่น กระแสขึ้นไม่ทัน ขดลวดมีอิมพีแดนซ์สูง หรือ Back-EMF สูงเมื่อความเร็วเพิ่มขึ้น
5. การสั่นในบางความถี่ทำให้แรงบิดลดลง และเกิดการสูญเสียตำแหน่งได้ แม้โหลดไม่มาก

ขั้นตอนการดำเนินงาน

การทดลองหาความสัมพันธ์ระหว่างโหมดการทำงานของ Stepper Motor และ ความเร็วที่ได้จาก Stepper motor ทำการควบคุมโหมดการทำงานไว้ที่โหมดต่าง หลังจากนั้นทำการเพิ่ม Frequency ไปเรื่อยๆ และทำการเก็บค่า RPM และ Frequency ทำซ้ำ 3 ครั้ง

ผลการทดลอง

ตารางที่ 8 แสดงความสัมพันธ์ และแนวโน้มของแต่ละโหมด

โหมด	กราฟความสัมพันธ์	กราฟแสดงแนวโน้ม RPM
Full step		
Half step		
Microstepping		

สรุปผลการทดลอง

จากการทดลองหาความสัมพันธ์ระหว่างความถี่สัญญาณอินพุต และความเร็วรอบของ Stepper Motor ในโหมด Full Step, Half Step และ Microstepping พบว่า

1. **ความสัมพันธ์เชิงเส้น:** ในช่วงความถี่เริ่มต้น ความเร็วรอบของมอเตอร์ (RPM) แปรผันตรงกับความถี่อินพุตจริงตามสมมติฐาน โดยความเร็วรอบ มีลักษณะเพิ่มขึ้นเป็นเส้นตรงตามการเพิ่มขึ้นของความถี่

2. **การเกิด Loss Step:** เมื่อความถี่อินพุตเพิ่มสูงเกินขีดจำกัดหนึ่ง มอเตอร์จะไม่สามารถสร้างแรงบิดเอาชนะความเฉื่อยได้ทัน ส่งผลให้เกิดการ Loss Step หรือ Stall (หยุดหมุน) โดยค่าความเร็วรอบจริงจะตกลงสู่ศูนย์ทันที แม้ว่าจะยังมีความถี่อินพุตเพิ่มขึ้นต่อไปก็ตาม ซึ่งสอดคล้องกับสมมติฐานที่ตั้งไว้

3. **ประสิทธิภาพ:** โหมดที่มีความละเอียดสูงกว่าสามารถทำความเร็วรอบสูงสุด (Max RPM) ได้มากกว่า โดย Microstepping ทำความเร็วได้สูงสุด (226.86 RPM) รองลงมาคือ Half Step (177.12 RPM) และ Full Step ทำได้น้อยที่สุด (153.68 RPM)

อภิปรายผล

1. ความสัมพันธ์ระหว่างความถี่ และ Loss Step จากกราฟ Half Step แสดงให้เห็นพฤติกรรมที่ชัดเจนที่สุด คือความเร็วรอบเพิ่มขึ้นในช่วงแรกจนถึงจุดยอด แล้วตกลงอย่างรวดเร็ว

สอดคล้องกับสมมติฐาน การป้อนความถี่ที่สูงเกินกว่าแรงบิดของมอเตอร์จะเอาชนะความเฉื่อยได้ จะทำให้เกิดการ Loss Step

เมื่อความถี่สูงขึ้น ค่าความเหนี่ยวนำของขดลวดทำให้กระแสไหลเข้าไม่เต็มที่ แรงบิดจึงลดลง เมื่อแรงบิดต่ำกว่าแรงเสียดทาน มอเตอร์จึงหยุดหมุน (Stall) ทำให้กราฟ RPM ร่วงลงแม้ความถี่อินพุต จะยังสูงขึ้น

2. ความเร็ว แม้สมมติฐานจะระบุว่า ที่ความถี่ Input เท่ากัน โหมดละเอียดสูง (Microstep) จะหมุนช้ากว่า ซึ่งเป็นจริงในเชิงทฤษฎีระยะทางต่อพัลส์ แต่ผลการทดลองพบว่าในแง่ของ เสถียรภาพ และความเร็วสูงสุดที่ทำได้จริง Microstepping กลับทำได้ดีที่สุด

สาเหตุเพราะ การขับแบบ Full Step กราฟความเร็วรอบราบเรียบติดศูนย์ แสดงถึงการเกิดการสั่น อย่างรุนแรงจากการเปลี่ยนสเต็ปที่กระชาก ทำให้มอเตอร์หลุดสเต็ปตั้งแต่ออกตัว ไม่สามารถไต่ความเร็วได้ ในขณะที่ Microstepping และ Half Step มีการเคลื่อนที่ที่ Smooth กว่า จึงลดแรงสั่นสะเทือน และสามารถไต่ความถี่ขึ้นไปได้สูงกว่าโดยไม่เกิด Loss Step เร็วเท่ากับ Full Step

ข้อเสนอแนะ

- แนะนำให้เขียนโปรแกรมทำ *Ramp* สำหรับปรับ *Frequency* และใช้ การหน่วงเวลาเพื่อให้เห็น *RPM* ที่เปลี่ยนไปอย่างชัดเจน

อ้างอิง

https://www.faulhaber.com/en/know-how/tutorials/stepper-motor-tutorial-how-to-prevent-step-losses-with-stepper-motors/

https://www.gian-transmission.com/what-causes-stepper-motors-to-lose-steps/

https://www.xfoyomotor.com/blog/what-is-the-step-loss-in-a-gearbox-stepper-motor-528149.html

การทดลองที่ 3 ผลของความเร่ง (Effect of Acceleration)

จุดประสงค์

1. เพื่อศึกษาว่าการใช้ Acceleration (Ramp up) ช่วยลดปัญหา Loss Step ที่ความเร็วสูงได้อย่างไร

สมมติฐาน

การค่อยๆเพิ่มความถี่ จะช่วยให้มอเตอร์สามารถทำความเร็วได้สูงกว่า Start/Stop Frequency โดยไม่เกิด Loss Step

ตัวแปร

1. ตัวแปรต้น:
- อัตราเร่ง (Slope ของ Ramp)
2. ตัวแปรตาม:
- RPM
- *ความต่อเนื่องของ* RPM
3. ตัวแปรควบคุม:
- แรงดันไฟเลี้ยง
- Drive Mode (เลือกใช้ Micro Step เพื่อให้เห็นผลชัดเจนที่สุด)

เอกสารและงานวิจัยที่เกี่ยวข้อง

ความสัมพันธ์ของแรงบิด (Torque) กับความเร็วรอบ ของ Stepper Motor ที่มักแสดงเป็น Pull-in และ Pull-out Torque Curve โดยแรงบิดของมอเตอร์จะลดลงเมื่อความถี่ของ Pulse สูงขึ้น ซึ่งหมายความว่าเมื่อสั่งให้มอเตอร์หมุนเร็ว แต่แรงบิดไม่เพียงพอที่จะเอาชนะแรงเฉื่อยหรือแรงเสียดทาน มอเตอร์จะไม่สามารถหมุนก้าวตามสัญญาณได้และเกิด Loss Step ขึ้นทันที การเข้าใจเส้นลักษณะนี้ช่วยให้สามารถคาดการณ์จุดที่จะเกิดปัญหาได้ก่อนการทดลองจริง

Acceleration Profile การสั่งให้มอเตอร์เริ่มหมุนที่ความถี่สูงทันทีโดยไม่มีการไล่ความเร็ว (Ramp) ทำให้กระแสในขดลวดไม่ทันสร้างแรงบิดเต็มที่ ส่งผลให้มอเตอร์หลุด step ได้ง่าย การออกแบบ Acceleration ที่เหมาะสม เช่น Linear Ramp หรือ S-curve Ramp สามารถช่วยลดโอกาสการเกิด Loss Step ได้อย่างมาก และเป็นหัวใจของการควบคุมมอเตอร์ในระบบจริง เช่น เครื่อง CNC และเครื่องพิมพ์สามมิติ

ในด้านการวัด จำเป็นต้องมีความเข้าใจพื้นฐานเรื่อง **Encoder และการประเมินตำแหน่ง** เพื่อใช้ตรวจสอบว่ามอเตอร์เคลื่อนที่ตามคำสั่งหรือไม่ โดยการเปรียบเทียบตำแหน่งที่สั่ง (Commanded Steps) กับตำแหน่งจริงที่วัดได้ (Measured Steps) จะทำให้สามารถตรวจจับการสูญเสียสเต็ปได้อย่างแม่นยำ รวมถึงการวิเคราะห์พฤติกรรมกระแสของมอเตอร์ ซึ่งสะท้อนถึงความสามารถในการสร้างแรงบิดในช่วงเร่งหรือช่วงที่มีโหลดสูง

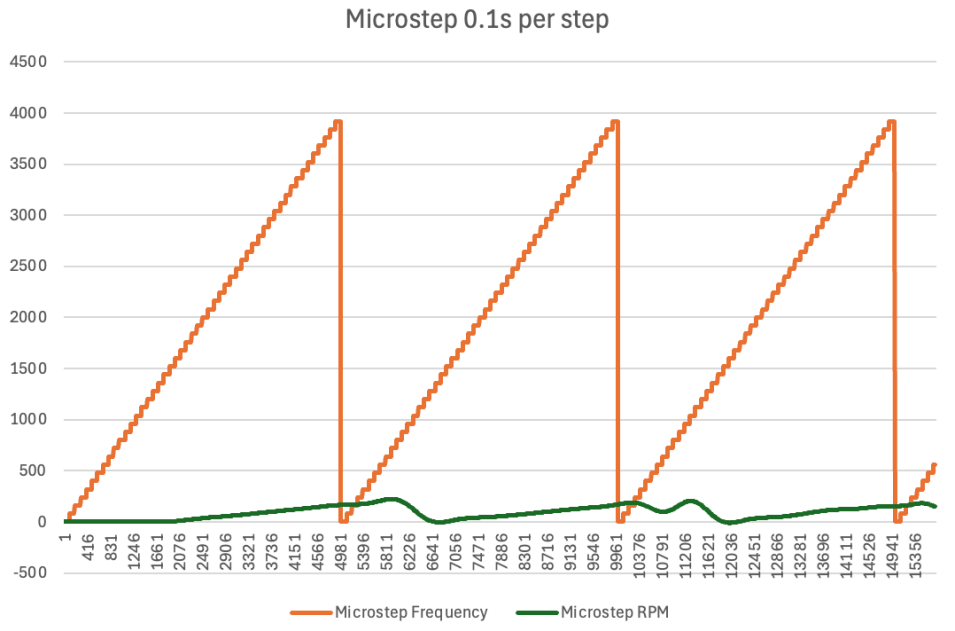
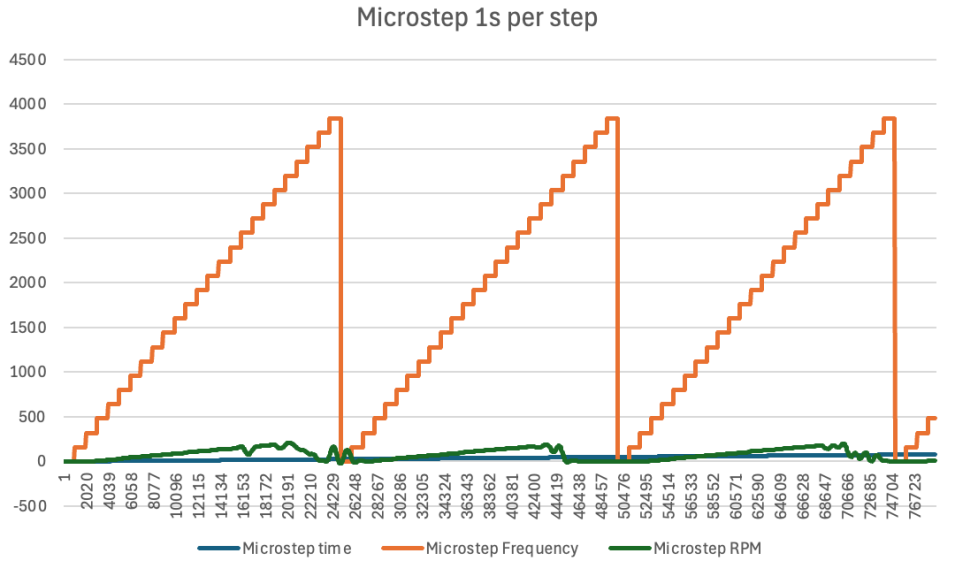
Rotational Dynamics เช่นแรงเฉื่อย (Inertia) และแรงต้านทางกล (Mechanical Load) เนื่องจากระบบที่มีแรงเฉื่อยสูงจะต้องการแรงบิดมากขึ้นในการเร่งความเร็ว ซึ่งมีผลโดยตรงต่อการเกิด Loss Step

ขั้นตอนการดำเนินงาน

การทดลองหาความสัมพันธ์ระหว่างความเร่งของ Frequency ต่อ Stepper Motor และ ความเร็วที่ได้จาก Stepper motor และทำการเปลี่ยน Duration ต่อ 1 step เป็นค่าต่างๆ หลังจากนั้นทำการเพิ่ม Frequency ไปเรื่อยๆ และทำการเก็บค่า RPM และ Frequency ทำซ้ำ 3 ครั้ง

ผลการทดลอง

ตารางที่ 9 แสดงลักษณะการเปรียบเทียบระหว่าง Frequency และ RPM ของการตั้งค่าที่แตกต่างกัน

การตั้งค่า	กราฟความสัมพันธ์
0.1s Per Step	
1s Per Step	

สรุปผลการทดลอง

1. **การไต่ระดับความเร็ว:** การเร่งแบบละเอียด (*0.1s/step)* ช่วยให้มอเตอร์ไต่ระดับความเร็วได้จริงและลดปัญหา *Loss Step* ได้ตามสมมติฐาน
2. **การหยุดหมุน (*Stall):*** การเร่งแบบหยาบหรือทิ้งช่วงนาน (*1s/step)* ทำให้มอเตอร์หยุดหมุนทันทีและไม่สามารถทำความเร็วได้ เนื่องจากเกิดการกระชากจนแรงบิดไม่พอเอาชนะความเฉื่อย

อภิปรายผล

กราฟ Microstep ที่แสดงความสัมพันธ์ระหว่าง Microstep Frequency (เส้นสีส้ม) กับ Microstep RPM (เส้นสีเขียว) แสดงให้เห็นถึงการใช้อัตราเร่ง (Ramp Up)

- Ramp Up (อัตราเร่ง): เส้นสีส้มแสดงการเพิ่มความถี่พัลส์ที่สั่งให้มอเตอร์หมุนอย่างต่อเนื่อง และเป็นขั้นบันได (การเร่ง)
- ช่วงทำงานปกติ: ในช่วงความถี่ต่ำถึงปานกลาง เส้นสีเขียว (RPM) จะเกาะติดกับเส้นสีส้ม หรือคงที่อย่างเหมาะสม ซึ่งแสดงว่ามอเตอร์สามารถ ติดตามสัญญาณ ได้อย่างถูกต้อง (ไม่เกิด Loss Step)
- จุดเกิด Loss Step / Stall: เมื่อความถี่ถึงจุดสูงสุด (จุดกระชากของเส้นสีส้ม) หรือเมื่อมอเตอร์ถึงขีดจำกัด แรงบิดไม่พอเอาชนะแรงเฉื่อย เส้นสีเขียว (RPM) จะตกลงอย่างรวดเร็ว หรือกลับไปใกล้ศูนย์ ซึ่งเป็นหลักฐานที่แสดงว่ามอเตอร์ เกิด Loss Step หรือ หยุดนิ่ง (Stall)

ความต่อเนื่องช่วงลด Loss Step กราฟ 0.1s พิสูจน์ว่าการเปลี่ยนความถี่ทีละน้อย ช่วยรักษาแรงบิดให้ต่อเนื่องพอที่จะลากโหลดผ่านแรงเสียดทาน และความเฉื่อยได้ ต่างจากแบบ 1s ที่เปรียบเสมือนการกระชากรุนแรงจนมอเตอร์หลุดสเต็ปตั้งแต่เริ่ม

เสถียรภาพของการหมุน: แม้การใช้ Ramp up (0.1s) จะทำให้มอเตอร์หมุนได้ แต่กราฟความเร็วรอบจริง ยังมีการแกว่งตัวไม่เป็นเส้นตรงเป๊ะ แสดงให้เห็นว่าแม้อัตราเร่งจะช่วยได้ แต่ยังต้องระวังเรื่องการสั่น ในบางช่วงความถี่ที่ทำให้ความเร็วไม่นิ่ง

ข้อเสนอแนะ

- ควรกำหนดจุด ***Maximum Start/Stop Frequency*** ของแต่ละโหมดอย่างชัดเจน โดยวัดความถี่สูงสุดที่มอเตอร์สามารถเริ่มหมุนและหยุดหมุนได้ทันทีโดยไม่เกิด *Loss Step*
- ควรมีการติดตั้งมอเตอร์อย่างแน่นหนาหรือใช้วัสดุดูดซับการสั่นสะเทือน เพื่อลดการสั่นรุนแรงที่เกิดขึ้นในโหมด *Full Step* ซึ่งอาจส่งผลให้เกิด *Loss Step* ได้ตั้งแต่ความเร็วต่ำ

อ้างอิง

https://www.faulhaber.com/fileadmin/Import/Media/AN002_EN.pdf

https://www.portescap.com/en/newsroom/whitepapers/2022/10/understanding-and-measuring-the-pull-out-and-pull-in-torque-of-a-stepper-motor

3. Brushless DC motor

การทดลองที่ 1 การทำงานจริง Brushless DC motorแบบ 6-Step โดยใช้ Back EMF Sensing

จุดประสงค์

1. เพื่อศึกษาหลักการทำงานและการขับเคลื่อนมอเตอร์ *BLDC* แบบ *6-Step Commutation* โดยใช้วิธีตรวจจับสัญญาณ *Back EMF (Sensorless)*
2. เพื่อแสดงการเกิด *Phase Shift* ระหว่างสัญญาณแต่ละ *Phase* ของ *BMF*
3. เพื่อศึกษาความสัมพันธ์ระหว่างความถี่ของสัญญาณ *Back EMF* กับความเร็วรอบของมอเตอร์ และคำนวณหาความเร็วรอบจากการทดลองและทิศทางการหมุน

สมมติฐาน

ในการทดลองขับมอเตอร์ Brushless DC แบบ 6-Step Commutation โดยใช้เทคนิคการตรวจจับสัญญาณ Back EMF สัญญาณแรงดันย้อนกลับที่วัดได้ของแต่ละเฟสจะมีลักษณะเป็นรูปคลื่นสี่เหลี่ยมคางหมู และมีมุมต่างเฟสกัน 120 องศาทางไฟฟ้า โดย ความถี่ของสัญญาณ Back EMF จะมีความสัมพันธ์แบบแปรผันตรงกับความเร็วรอบของมอเตอร์ หากความถี่ของสัญญาณ Back EMF เพิ่มขึ้น ความเร็วรอบของมอเตอร์ก็จะสูงขึ้นตามไปด้วย นอกจากนี้ลักษณะทิศทางการหมุนจะแตกต่างกันโดยที่ขึ้นอยู่กับลักษณะของคลื่นที่เรียงกัน

ตัวแปร

1. ตัวแปรต้น:
- Reference Speed (0,1876,4942,5916,7890,9864 RPM)
- ทิศทางการหมุนของมอเตอร์ (CW)
2. ตัวแปรตาม:
- Mechanical speed
- ความถี่สัญญาณ Back EMF
3. ตัวแปรควบคุม:
- Driver Supply Voltage 24v
- Duration (1000 ms)
- No Load

เอกสารและงานวิจัยที่เกี่ยวข้อง

หลักการทำงานพื้นฐานของระบบขับเคลื่อน Brushless DC Motor มีความแตกต่างจากมอเตอร์ไฟฟ้ากระแสตรงแบบดั้งเดิม กล่าวคือ โครงสร้างภายในได้ถูกออกแบบให้ขดลวดตัวนำอยู่ที่ส่วนอยู่กับที่ Stator และใช้แม่เหล็กถาวรติดตั้งที่ส่วนเคลื่อนที่ Rotor โดยตัดอุปกรณ์เปลี่ยนทิศทางกระแสกลหรือแปรงถ่านออก ดังนั้น การควบคุมการหมุนจึงจำเป็นต้องอาศัยวงจรอิเล็กทรอนิกส์ทำหน้าที่เปลี่ยนทิศทางกระแสไฟฟ้า เพื่อสร้างสนามแม่เหล็กหมุนที่สัมพันธ์กับตำแหน่งของโรเตอร์อย่างแม่นยำ กลไกการสร้างสัญญาณขับเคลื่อน 3 เฟส อาศัยการทำงานของวงจรอินเวอร์เตอร์ซึ่งประกอบด้วยอุปกรณ์สวิตช์อิเล็กทรอนิกส์ เช่น MOSFET หรือ IGBT จำนวน 6 ตัว เชื่อมต่อกันในลักษณะวงจรบริดจ์ 3 เฟส (3-Phase Bridge Configuration) โดยอุปกรณ์สวิตช์เหล่านี้จะถูกควบคุมให้เปิดและปิดวงจรตามลำดับขั้นตอนที่กำหนดไว้ เพื่อจ่ายกระแสไฟฟ้า เข้าสู่ขดลวดสเตเตอร์แต่ละเฟส (U, V, และ W) ทำให้เกิดความต่างศักย์และกระแสไฟฟ้าไหลผ่านขดลวดในทิศทางที่กำหนด ส่งผลให้เกิดแรงบิดแม่เหล็กไฟฟ้าผลักดันให้โรเตอร์หมุนไปในทิศทางที่ต้องการ

เพื่อให้การสลับเฟสสัญญาณมีความถูกต้องแม่นยำ ระบบควบคุมจำเป็นต้องทราบข้อมูลตำแหน่งของโรเตอร์แบบเรียลไทม์ ซึ่งสามารถกระทำได้โดยการใช้ Hall Effect Sensors ติดตั้งภายในตัวมอเตอร์ หรือการใช้เทคนิคไร้เซ็นเซอร์ โดยการตรวจวัดแรงเคลื่อนไฟฟ้าต้านกลับ Back Electromotive ForceBack-EMF จากขดลวดเฟสที่ไม่ได้นำกระแส นอกจากนี้ การควบคุมขนาดของแรงดันไฟฟ้าและความเร็วรอบมอเตอร์ จะใช้เทคนิค PWM ในการปรับค่าแรงดันเฉลี่ยที่จ่ายให้กับขดลวด เพื่อควบคุมกระแสและแรงบิดให้เป็นไปตามค่าเป้าหมาย

ในการขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงไร้แปรงถ่าน นั้นประสิทธิภาพและพฤติกรรมตอบสนองของมอเตอร์ขึ้นอยู่กับอัลกอริทึมในการควบคุมสัญญาณไฟฟ้าเป็นสำคัญ ซึ่งในทางวิศวกรรมสามารถจำแนกการควบคุมออกเป็น 2 รูปแบบหลักที่มีความแตกต่างกันในเชิงหลักการและผลลัพธ์การทำงาน ดังนี้

1. การควบคุมแบบสลับเฟส 6 ขั้นตอน (6-Step Commutation)

การควบคุมแบบ 6-Step Commutation หรือที่รู้จักในชื่อ Trapezoidal Control เป็นวิธีการพื้นฐานที่นิยมใช้กับมอเตอร์ที่มีรูปคลื่นแรงดันต้านกลับ (Back-EMF) เป็นรูปสี่เหลี่ยมคางหมู หลักการทำงานอาศัยการแบ่งรอบการหมุนทางไฟฟ้าออกเป็น 6 ส่วน (Sectors) แต่ละส่วนมีมุมกว้าง 60 องศา โดยวงจรควบคุมจะทำการจ่ายกระแสไฟฟ้าเข้าสู่ขดลวดสเตเตอร์ตามลำดับขั้นตอนที่สัมพันธ์กับตำแหน่งของโรเตอร์

กระบวนการทำงานในแต่ละขั้นตอน (Step) จะมีลักษณะเฉพาะคือ มีขดลวดเพียง 2 เฟสที่ทำหน้าที่นำกระแส (Active Phases) โดยเฟสหนึ่งทำหน้าที่รับกระแสไหลเข้าและอีกเฟสหนึ่งรับกระแสไหลออก ในขณะที่เฟสที่สามจะถูกตัดวงจรและอยู่ในสภาวะลอยตัว (Floating Phase) การสลับเปลี่ยนคู่เฟสจะเกิดขึ้นเมื่อโรเตอร์หมุนผ่านตำแหน่งที่กำหนด ซึ่งตรวจจับได้โดยเซ็นเซอร์ฮอลล์ (Hall Effect Sensors) หรือการวัดสัญญาณ Back-EMF ผ่านเฟสที่ลอยตัวโดนเราสามารถคำนวณความเร็วกลับได้จากสมการ

$$n = \frac{120f}{p}$$

โดย

n = ความเร็ว (RPM)

f = ความถี่ที่วัดได้ (Hz)

p = จำนวน Poles

รูปที่ 5 แสดงมอเตอร์ที่ใช้ทดลอง รุ่น A221213T

ตารางที่ 11 แสดงรายละเอียดของมอเตอร์ รุ่น A221213T

No. of Cells:	2-3 Li-Poly 6-10 NiCd/NiMH
KV:	1000 RPM/V
Max Efficiency:	80%
No Load Current	0.5A@10v
Resistance:	0.090 ohms
Max Current:	13A for 60S
Max Watts:	150W
Weight	52.7 g / 1.86 oz
Size:	28mm dia x 28 mm bell length
Shaft Diameter	3.2 mm
Poles:	14
Model Weight	300-800g / 10.5 – 28.2 oz

ซึ่งจำนวน Poles = 14

แม้ว่าวิธีการนี้จะมีจุดเด่นในด้านความเรียบง่ายของโครงสร้างวงจรและการประมวลผลที่ไม่ซับซ้อน แต่มีข้อจำกัดทางกายภาพคือ การจ่ายกระแสสลับขั้วในทันทีที่เปลี่ยนเซกเตอร์ ทำให้เกิด Torque Ripple สูง ส่งผลให้มอเตอร์มีการสั่นสะเทือนและเกิดเสียงรบกวนในขณะหมุน โดยเฉพาะอย่างยิ่งในช่วงความเร็วรอบต่ำ

2. การควบคุมแบบวางแนวสนามแม่เหล็ก (Field Oriented Control: FOC)

การควบคุมแบบ FOC หรือ Vector Control เป็นเทคนิคชั้นสูงที่นิยมใช้กับมอเตอร์ PMSM หรือ BLDC ที่ต้องการประสิทธิภาพสูง และความนิ่มในการขับเคลื่อน หลักการสำคัญคือการควบคุมเวกเตอร์ของกระแสไฟฟ้า และสนามแม่เหล็กให้มีความสัมพันธ์ตั้งฉากกันตลอดเวลา ซึ่งเป็นสภาวะที่ก่อให้เกิดแรงบิดสูงสุดต่อหน่วยกระแส

กระบวนการของ FOC อาศัยแบบจำลองทางคณิตศาสตร์ในการแปลงแกนอ้างอิงของระบบไฟฟ้ากระแสสลับ 3 เฟส (I_a, I_b, I_c)ซึ่งเป็นค่าที่เปลี่ยนแปลงตามเวลา ให้กลายเป็นระบบไฟฟ้ากระแสตรง 2 แกนบนกรอบอ้างอิงที่หมุนไปพร้อมกับโรเตอร์ (d - q Reference Frame) ผ่านกระบวนการแปลงทางคณิตศาสตร์ 2 ขั้นตอน ได้แก่:

1. การแปลงคลาร์ก (Clarke Transformation): แปลงระบบ 3 เฟส เป็นแกนคงที่ 2 มิติ (α, β)
2. การแปลงปาร์ค (Park Transformation): แปลงแกนคงที่ เป็นแกนหมุน (d, q)

ผลลัพธ์ที่ได้คือกระแสไฟฟ้า 2 องค์ประกอบ ได้แก่ กระแสแกน d (I_d) ซึ่งควบคุมฟลักซ์แม่เหล็ก และกระแสแกน $q(I_q)$ ซึ่งควบคุมแรงบิด ระบบควบคุมจะทำการปรับค่ากระแสทั้งสองนี้ผ่านตัวควบคุมแบบ PI (Proportional-Integral Controller) เพื่อให้ได้ค่าแรงบิดตามต้องการ จากนั้นจึงทำการ Inverse Transformation และสร้างสัญญาณขับผ่านเทคนิค Space Vector Pulse Width Modulation (SVPWM)

การควบคุมด้วยวิธี FOC ช่วยขจัดปัญหาแรงบิดกระเพื่อม ทำให้มอเตอร์หมุนได้อย่างราบรื่น เสียงเงียบ และมีประสิทธิภาพการใช้พลังงานสูงสุด อีกทั้งยังสามารถตอบสนองต่อการเปลี่ยนแปลงความเร็ว และโหลดได้อย่างรวดเร็ว (High Dynamic Response)

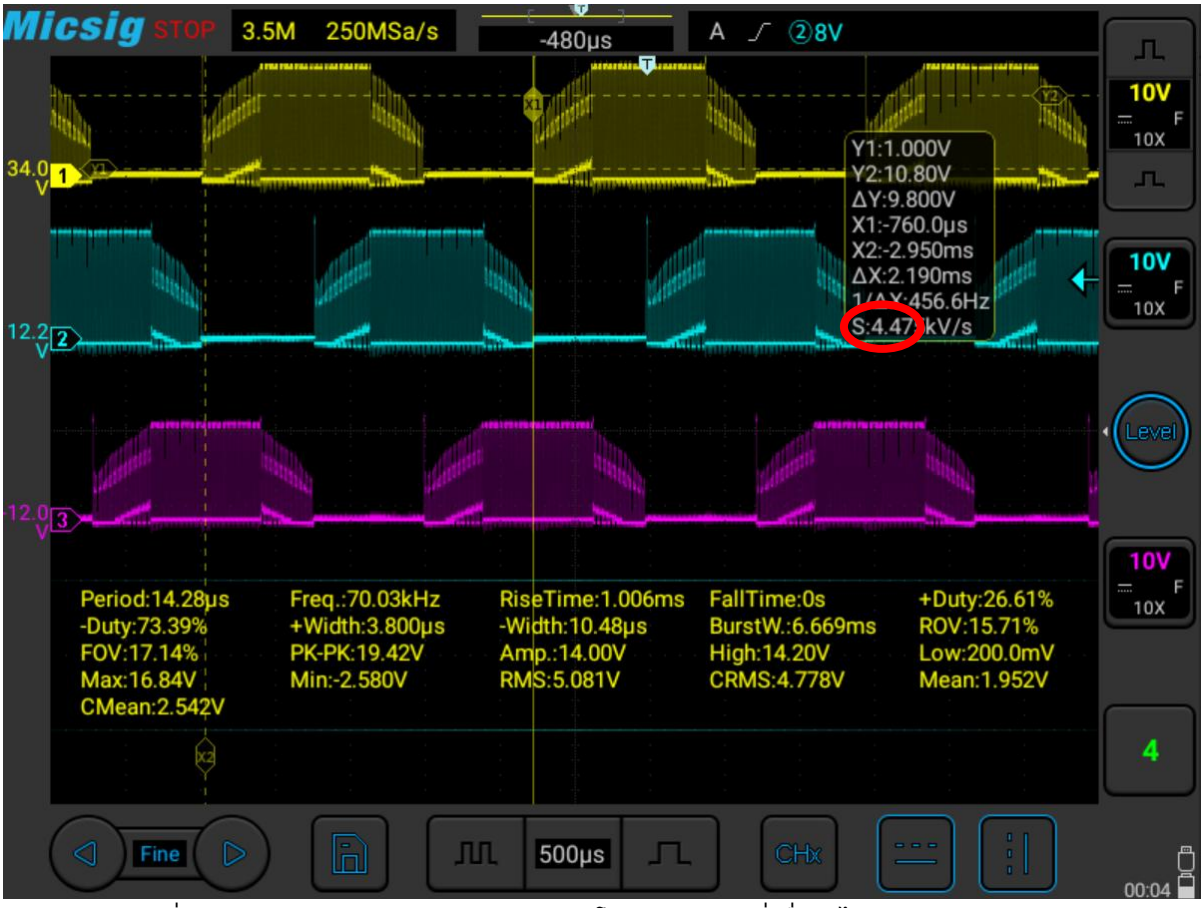
ขั้นตอนการดำเนินงาน

เชื่อมต่อบอร์ดการทดลองเข้ากับ Oscilloscope โดยใช้ทั้งหมด 3 channel และเชื่อม GND เข้าด้วยกัน จากนั้นตั้ง Duration 500 ms เนื่องจากเป็นค่าที่เหมาะสมไม่เร็วเกินไปและช้าเกินไป จากนั้นจ่าย Speed Reference ตั้งแต่ (0,1876,4942,5916,7890,9864 RPM) ตามลำดับ ในแต่ละรอบให้กดปุ่ม Cursor บนเครื่อง Oscilloscope หมุนปรับเส้น Cur A (X1) ไปวางไว้ที่ "ขอบขาขึ้น" ของลูกคลื่นสีเหลืองลูกแรกหมุนปรับเส้น Cur B (X2) ไปวางไว้ที่ "ขอบขาขึ้น" ของลูกคลื่นสีเหลืองลูกถัดไปเพื่อดูความถี่ (Frequency) ของสัญญาณ Back EMF ที่ใช้คำนวณความเร็วรอบจากนั้นถ่ายภาพหน้าจอเอาไว้ ทำอย่างนี้ซ้ำทั้งหมด 3 รอบ

ผลการทดลอง

ตารางที่ 11 แสดงตัวอย่างจากการทดลอง

ตัวอย่างรูปจากการเก็บผลการทดลองทิศทาง (CW)
ตัวอย่างรูปจากการเก็บผลการทดลองทิศทาง (CCW)



รูปที่ 56ตัวอย่างรูปจากการเก็บผลการทดลองโดนการดูความถี่เพื่อนำไปคำนวณหาความเร็ว

ตารางที่ 12 แสดงการเก็บผลในแต่ละรอบการทดลอง

รอบที่ 1				
Reference Speed (RPM)	ความถี่ BEMF ที่วัดได้ (Hz)	Reference Speed (RPM)	ความเร็วรอบจากการคำนวณ (RPM)	ความคลาดเคลื่อน (%)
0	0	0	0	0
1876	213.7	1850	1831.71	2.36
3942	456.6	3984	3913.71	0.71
5916	654.5	5982	5610	5.17
7890	871.1	7988	7466.57	5.36
9864	1000	9798	8571.42	13.10
รอบที่ 2				
Reference Speed (RPM)	ความถี่ BEMF ที่วัดได้ (Hz)	Reference Speed (RPM)	ความเร็วรอบจากการคำนวณ (RPM)	ความคลาดเคลื่อน (%)
0	0	0	0	0
1876	222.2	1860	1904.57	1.52
3942	452.5	3870	3878.57	1.60
5916	684.9	5770	5870.57	0.76
7890	902.5	7968	7735.71	1.95

9864	1000	9450	8571.42	13.10
รอบที่ 3				
Reference Speed (RPM)	ความถี่ BEMF ที่วัดได้ (Hz)	Reference Speed (RPM)	ความเร็วรอบจากการคำนวณ (RPM)	ความคลาดเคลื่อน (%)
0	0	0	0	0
1876	217.4	1870	1863.42857	0.67
3942	460.8	3882	3949.71429	0.19
5916	684.3	5944	5865.42857	0.85
7890	902.5	7906	7735.71429	1.95
9864	1000	9888	8571.42857	13.10

สรุปผลการทดลอง

จากการทดลองเรื่องการทำงานจริงของ Brushless DC Motor แบบ 6-Step โดยใช้การตรวจจับสัญญาณแรงดันย้อนกลับสามารถสรุปผลได้ดังนี้ ลักษณะของสัญญาณ Back EMF จากการตรวจวัดสัญญาณด้วยออสซิลโลสโคป พบว่าสัญญาณแรงดันย้อนกลับที่วัดได้จากขดลวดสเตเตอร์ทั้ง 3 เฟส มีลักษณะเป็นรูปคลื่นสี่เหลี่ยมคางหมู และมีการเลื่อนเฟส ต่างกัน 120 องศาทางไฟฟ้า ซึ่งสอดคล้องกับทฤษฎีการทำงานของมอเตอร์แบบ 6-Step Commutation และเป็นไปตามสมมติฐานที่ตั้งไว้ความสัมพันธ์ระหว่างความถี่ และความเร็วรอบและทิศทางผลการทดลองแสดงให้เห็นว่า ความถี่ของสัญญาณ Back EMF (f) มีความสัมพันธ์แบบแปรผันตรงกับความเร็วรอบของมอเตอร์ (n) ตามสมการ

$$\mathrm{n} = \frac{120\mathrm{f}}{\mathrm{p}}$$

อภิปรายผล

การวิเคราะห์ค่าความคลาดเคลื่อนจากการทดลองซ้ำจำนวน 3 รอบ พบผลการวิเคราะห์ความคลาดเคลื่อนดังนี้ช่วงความเร็วรอบต่ำถึงปานกลาง (1,876 - 7,890 RPM) การคำนวณความเร็วรอบจากสัญญาณ Back EMF มีความแม่นยำสูง โดยมีค่าความคลาดเคลื่อนเฉลี่ยอยู่ในเกณฑ์ต่ำ (ประมาณ 0.19% ถึง 5.36%) ตัวอย่างเช่น ในการทดลองรอบที่ 3 ที่ความเร็วอ้างอิง 1,876 RPM วัดความถี่ได้ 217.4 Hz คำนวณความเร็วได้ 1,863.43 RPM คิดเป็นความคลาดเคลื่อนเพียง 0.67% ช่วงความเร็วรอบสูง (9,864 RPM) พบค่าความคลาดเคลื่อนคงที่ที่ระดับสูงถึง 13.10% ในทุกรอบการทดลอง ซึ่งแสดงให้เห็นถึงขีดจำกัดของระบบที่ความเร็วรอบสูงสุด

ข้อเสนอแนะ

- ในการทดลองครั้งต่อไป ควรมีการติดตั้งตัววัดความเร็วรอบภายนอกเพื่อตรวจสอบความถูกต้องของข้อมูล
- ควรตรวจสอบขีดจำกัดความถี่สวิตชิ่งของชุดขับเคลื่อนก่อนการทดลอง เพื่อป้องกันความผิดพลาดในการเก็บข้อมูลที่ความเร็วรอบสูง

อ้างอิง

https://www.jkongmotor.com/th/comprehensive-introduction-to-3-phase-bldc-motors.html

https://th.vsdmotor.com/info/detailed-explanation-of-brushless-motor-contro-17361180357272576.html

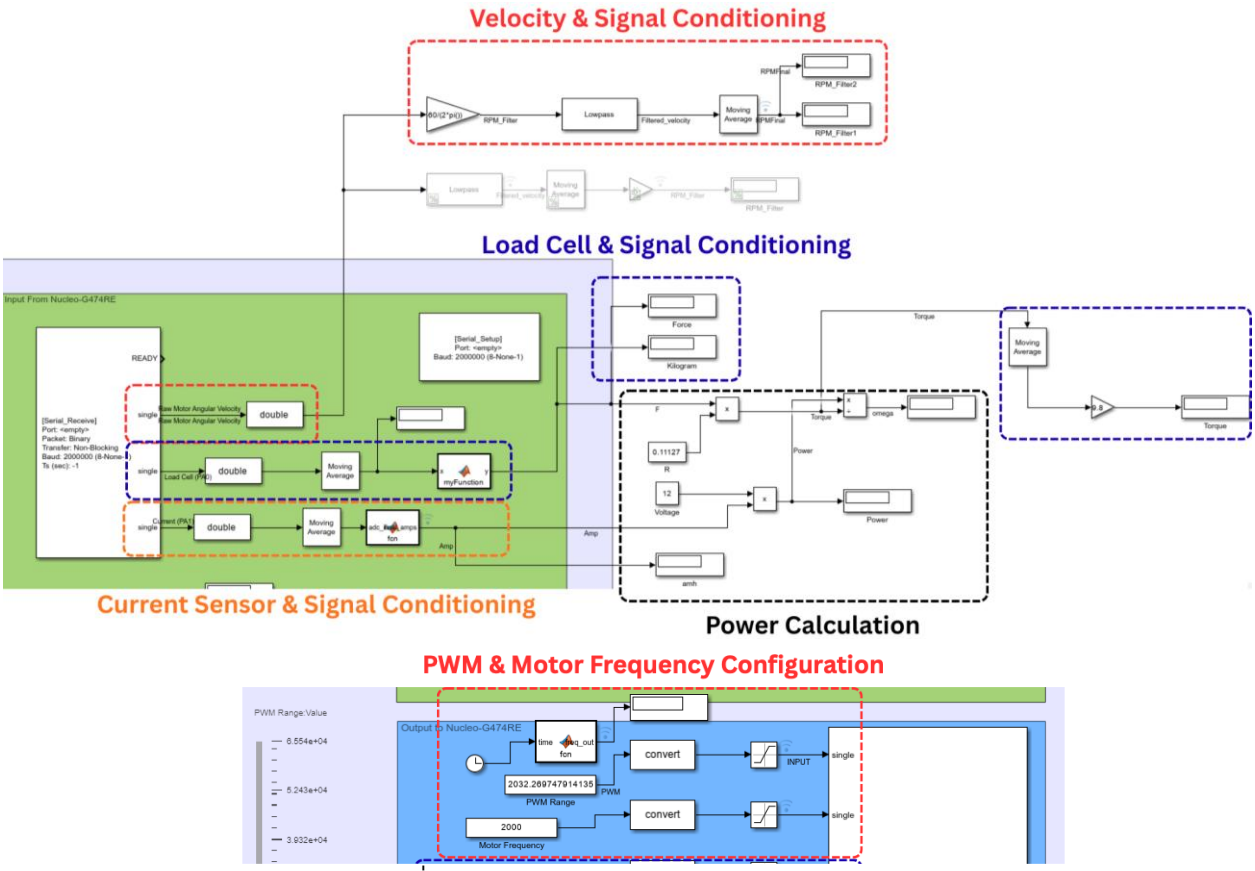
https://www.pmdcorp.com/resources/type/articles/get/field-oriented-control-foc-a-deep-dive-article

https://www.rhydolabz.com/documents/26/BLDC_A2212_13T.pdf?srsltid=AfmBOorus2J4qC9Tl5CnHqWQti95k_ykNUufb5wFfd0TBgxvhEELxWXa

ภาคผนวก

Lab 2.1 DC Motor

ภาคผนวก ก.1 โครงสร้างไฟล์ Simulink (MotorXplorer)



รูปที่ ก-1 ภาพรวมของ Model Simulink (Main View)

คำอธิบาย Block Diagram การทำงานของระบบ (Simulink Model Description)

1. Velocity & Signal Conditioning (ส่วนประมวลผลสัญญาณความเร็ว)

ส่วนนี้ทำหน้าที่รับค่าความเร็วเชิงมุม (Angular Velocity) ที่อ่านได้จาก Encoder ผ่านบอร์ด STM32 และนำมาปรับสภาพสัญญาณเพื่อให้เหมาะสมกับการวิเคราะห์และแสดงผล

- **Gain Block** ($60/2\pi$) ทำหน้าที่แปลงหน่วยความเร็วจาก เรเดียนต่อวินาที (rad/s) ให้เป็น รอบต่อนาที (RPM) ตามสมการ $\mathrm{RPM} = \omega_{\mathrm{rad/s}} \times \frac{60}{2\pi}$
- **Lowpass Filter:** ทำหน้าที่กรองสัญญาณรบกวนความถี่สูง (High-frequency noise) ที่เกิดจากการสั่นสะเทือนหรือสัญญาณรบกวนทางไฟฟ้า เพื่อให้ได้กราฟความเร็วที่เหมาะสม
- **Moving Average:** ใช้สำหรับการหาค่าเฉลี่ยเคลื่อนที่ของสัญญาณความเร็ว ช่วยลดความผันผวน ของข้อมูล ก่อนนำค่าไปแสดงผลหรือใช้ในการคำนวณต่อไป

2. Load Cell & Signal Conditioning (ส่วนประมวลผลสัญญาณแรง)

ส่วนนี้ทำหน้าที่อ่านค่าสัญญาณ Analog จาก Load Cell (ผ่านขา PA0) เพื่อนำมาคำนวณหาแรงบิด (Torque)

- **Moving Average:** เนื่องจากสัญญาณจาก Load Cell มี Noise รบกวนสูง การใช้ Moving Average จะช่วยเกลี่ยค่าแรงให้อ่านได้นิ่งและแม่นยำขึ้น
- **MATLAB Function (Load Cell Processing):** ทำหน้าที่แปลงค่าดิบ (ADC) เป็นค่าแรง (Force) ในหน่วยนิวตัน (N) โดยภายใน Block นี้บรรจุอัลกอริทึม Auto-Tare เพื่อปรับค่าศูนย์อัตโนมัติเมื่อไม่มีโหลด ช่วยกำจัดค่าความคลาดเคลื่อน (Zero Drift) และมีการใช้สมการคำนวณแบบ Piecewise Linear (สมการแยกช่วง) เพื่อชดเชยความไม่เป็นเชิงเส้นของ Load Cell ทำให้ได้ค่าแรงที่ถูกต้องแม่นยำตลอด ย่านการวัดก่อนส่งค่าแรงนี้ไปคูณกับรัศมีก้าน (r = 0.11127 m) เพื่อหาค่าแรงบิด (Torque)

3. Current Sensor & Signal Conditioning (ส่วนประมวลผลสัญญาณกระแสไฟฟ้า)

ส่วนนี้ทำหน้าที่อ่านค่ากระแสไฟฟ้าที่มอเตอร์ใช้งานจริงผ่าน Current Sensor (ผ่านขา PA1)

- **Moving Average:** ใช้กรองสัญญาณรบกวนในกระแสไฟฟ้าที่อาจเกิดจากการสวิตช์ของ PWM (Switching Noise)
- **MATLAB Function (adc_to_amps_fcn):** ทำหน้าที่แปลงค่าดิบจาก ADC ให้เป็นค่ากระแสไฟฟ้าในหน่วยแอมแปร์ (Ampere) โดยอ้างอิงจาก Sensitivity และ Offset ของเซนเซอร์ WCS1700 จากการทดลอง Calibrate

4. Power Calculation (ส่วนคำนวณกำลังงาน)

ส่วนนี้ทำหน้าที่คำนวณพารามิเตอร์ทางไฟฟ้าและทางกลแบบ Real-time เพื่อใช้ในการวิเคราะห์ประสิทธิภาพ

- **Torque Calculation:** คำนวณแรงบิดจากสมการ $\tau = F \times r$ โดยนำค่าแรง F (จาก Load Cell) คูณกับรัศมีก้าน r (ค่าคงที่ 0.11127 m)
- **Electrical Power (P_{in}):** คำนวณกำลังไฟฟ้าขาเข้าจากสมการ $P_{in} = V \times I$ โดยใช้แรงดันคงที่ 12V คูณกับกระแสที่วัดได้จริง
- **Mechanical Power (P_{in}):** คำนวณกำลังงานกลขาออกจากสมการ $P_{out} = \tau \times \omega$

5. PWM & Motor Frequency Configuration (ส่วนกำหนดค่าควบคุมมอเตอร์)

ส่วนนี้เป็นส่วน User Interface สำหรับกำหนดค่าการทำงานของมอเตอร์ส่งไปยังบอร์ด STM32

- **PWM Range:** รับค่า Duty Cycle (0-100%) เพื่อกำหนดแรงดันเฉลี่ยที่จะจ่ายให้มอเตอร์
- **Motor Frequency:** กำหนดความถี่ของสัญญาณ PWM ที่ใช้ขับ H-Bridge (เช่น 2000 Hz) ซึ่งมีผลต่อความเรียบของกระแสและการตอบสนองของมอเตอร์

ภาคผนวก ก.2: โปรแกรมและผลการวิเคราะห์สเปกตรัมความถี่ (FFT Analysis)

ส่วนนี้แสดง Source Code ภาษา MATLAB ที่ใช้ในการประมวลผลสัญญาณความเร็วรอบมอเตอร์ (Raw Speed Data) เพื่อแปลงจากโดเมนเวลา (Time Domain) เป็นโดเมนความถี่ (Frequency Domain) ช่วยให้สามารถระบุย่านความถี่ของสัญญาณรบกวน (Noise) ได้อย่างชัดเจน

1. MATLAB Script สำหรับการวิเคราะห์ FFT โค้ดด้านล่างใช้สำหรับอ่านค่าตัวแปรจาก Workspace (ที่ได้จาก Simulink) และคำนวณหา Single-Sided Amplitude Spectrum P1(f)

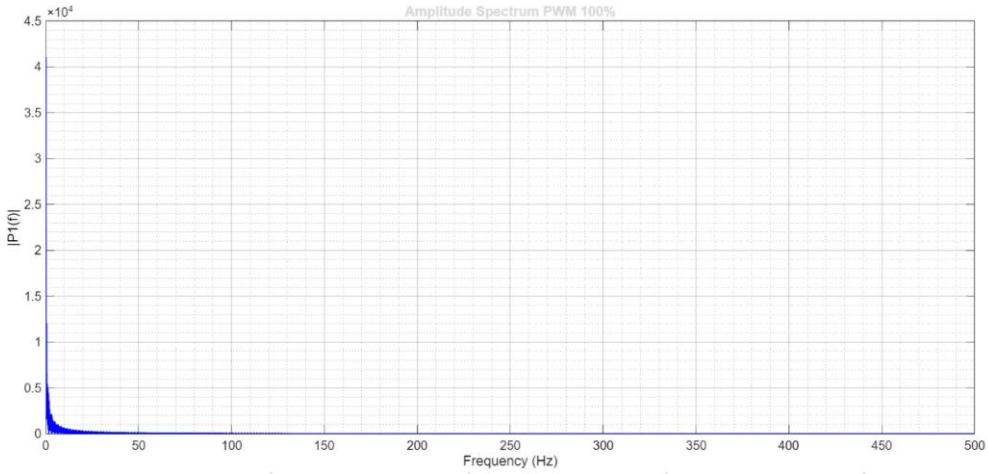
```
% --- 1. การคำนวณ FFT ---
% หมายเหตุ: PWMstep5{4} คือข้อมูลความเร็วรอบมอเตอร์ (Velocity Data) จาก Simulink

% คำนวณ Sampling Time (dt) และ Sampling Frequency (Fs) จากข้อมูลจริง
dt = PWMstep5{4}.Values.Time(2) - PWMstep5{4}.Values.Time(1);
Fs = 1/dt;

% เตรียมข้อมูลสำหรับ FFT
L = length(PWMstep5{4}.Values.Time);        % จำนวน Sample ทั้งหมด
x = squeeze(PWMstep5{4}.Values.Data);       % ดึงข้อมูลความเร็ว (Data) ออกมา

% คำนวณ Fast Fourier Transform
Y = fft(x);

% แปลงเป็น Single-Sided Spectrum
P2 = abs(Y/L);        % Two-sided amplitude spectrum
P1 = P2(1:L/2+1);     % Single-sided spectrum
P1(2:end-1) = 2*P1(2:end-1);

% สร้างแกนความถี่ (Frequency Axis)
f = Fs*(0:(L/2))/L;

% --- 2. การ Plot กราฟ (Format สำหรับรายงาน) ---
figure('Color', 'w');                   % กำหนดพื้นหลังหน้าต่างเป็นสีขาว

plot(f, P1, 'b', 'LineWidth', 1.2);    % พล็อตเส้นสีน้ำเงิน ความหนา 1.2
title('Single-Sided Amplitude Spectrum of Motor Speed');
xlabel('Frequency (Hz)');
ylabel('|P1(f)| (Magnitude)');

% ปรับแต่งแกนและฟอนต์ให้ชัดเจน (Excel/Publication Style)
set(gca, ...
    'Color', 'w', ...            % พื้นหลังภายในกราฟสีขาว
    'XColor', 'k', ...           % เส้นแกน X สีดำ
    'YColor', 'k', ...           % เส้นแกน Y สีดำ
    'FontSize', 12, ...          % ขนาดตัวอักษร 12 pt
    'FontName', 'TH Sarabun New'); % ใช้ฟอนต์เดียวกับรายงาน (หรือ Arial ตามเดิม
ก็ได้)
grid on;                         % เปิดเส้นตารางหลัก
grid minor;                      % เปิดเส้นตารางย่อยเพื่อให้อ่านค่าง่ายขึ้น

% ปรับช่วงแกน X เพื่อโฟกัสย่านความถี่ต่ำ (Optional)
xlim([0 500]);
```

คำอธิบายการทำงานของโปรแกรมการวิเคราะห์ FFT (Code Description)

โปรแกรม MATLAB ที่พัฒนาขึ้นนี้มีวัตถุประสงค์เพื่อวิเคราะห์องค์ประกอบความถี่ของสัญญาณความเร็วรอบมอเตอร์ (Motor Speed) ที่ได้จากการจำลองหรือการทดลองผ่าน Simulink โดยมีขั้นตอนการทำงานหลักดังนี้:

1. **การคำนวณค่าพารามิเตอร์การสุ่มตัวอย่าง (Sampling Parameters Calculation):** โปรแกรมเริ่มต้นด้วยการเข้าถึงข้อมูลดิบจากตัวแปร PWMstep5 (โครงสร้างข้อมูลจาก Simulink) โดยทำการคำนวณคาบเวลาในการสุ่มตัวอย่าง (**dt**) จากผลต่างเวลาระหว่างจุดข้อมูลที่ 1 และ 2 เพื่อนำไปหาความถี่ในการสุ่มตัวอย่าง (**Fs**) ที่แท้จริง ซึ่งเป็นขั้นตอนสำคัญเพื่อให้แกนความถี่ในกราฟมีความถูกต้องแม่นยำ
2. **การประมวลผล Fast Fourier Transform (FFT):** นำข้อมูลความเร็ว (x) มาผ่านฟังก์ชัน fft() เพื่อแปลงสัญญาณจากโดเมนเวลา (Time Domain) เป็นโดเมนความถี่ (Frequency Domain) ผลลัพธ์ที่ได้ (Y) จะเป็นค่าจำนวนเชิงซ้อนที่ประกอบด้วยทั้งขนาดและเฟส
3. **การแปลงเป็นสเปกตรัมด้านเดียว (Single-Sided Spectrum Conversion):** เนื่องจากผลลัพธ์จาก FFT จะได้สเปกตรัมแบบสองด้าน (Two-sided) ที่มีความสมมาตร โปรแกรมจึงทำการคำนวณหาขนาดของสัญญาณ (P2) โดยหารด้วยความยาวข้อมูล (L) เพื่อ Normalize ค่า จากนั้นทำการตัดเลือกเฉพาะช่วงความถี่ที่เป็นบวก (P1) และคูณด้วย 2 (ยกเว้นที่ความถี่ DC และ Nyquist) เพื่อชดเชยพลังงานที่หายไปจากการตัดสเปกตรัมด้านลบออก ทำให้ได้ค่า Amplitude Spectrum ที่ถูกต้องทางทฤษฎี
4. **การแสดงผลกราฟ (Visualization):** ส่วนสุดท้ายของโปรแกรมเป็นการตั้งค่ากราฟเพื่อให้เหมาะสมกับการนำเสนอในรายงาน โดยกำหนดพื้นหลังเป็นสีขาว (Color, w) ใช้เส้นกราฟสีน้ำเงิน พร้อมทั้งเปิดใช้งานเส้นตาราง (Grid/Minor Grid) เพื่อให้อ่านค่าได้ง่าย และปรับแต่งฟอนต์ให้เป็นมาตรฐานเดียวกับเอกสาร (TH Sarabun New) รวมถึงจำกัดแกน X ไว้ที่ช่วง 0-500 Hz เพื่อโฟกัสเฉพาะย่านความถี่ต่ำที่สนใจศึกษา

2. ผลการวิเคราะห์สเปกตรัม (FFT Results) จากการรันโปรแกรมข้างต้นกับชุดข้อมูลความเร็วที่ Duty Cycle 100% ได้ผลลัพธ์ดังแสดงในรูปที่ ก-5



รูปที่ ก-2 กราฟแสดงสเปกตรัมขนาดของสัญญาณความเร็ว (Amplitude Spectrum) ที่ PWM 100%

การวิเคราะห์ผลจากกราฟ:

- **DC Component (0 Hz):** จะเห็นแท่งกราฟสูงที่สุดที่ความถี่ 0 Hz ซึ่งตัวแทนของ "ความเร็วรอบเฉลี่ย" ของมอเตอร์ (เป็นค่าที่เราต้องการ)
- **Noise Component (> 1 Hz):** สังเกตเห็นสัญญาณขนาดเล็กกระจายตัวตั้งแต่ความถี่ประมาณ 1 Hz เป็นต้นไป ซึ่งเกิดจากการสั่นสะเทือนทางกล (Mechanical Vibration) และสัญญาณรบกวนในระบบ
- **ข้อสรุป:** จากกราฟนี้ จึงเป็นที่มาของการเลือกค่า **Passband Edge Frequency = 1 Hz** ในการออกแบบ Low-Pass Filter (ในภาคผนวก ก.3) เพื่อตัดสัญญาณรบกวนเหล่านี้ออก โดยยังคงค่าความเร็วเฉลี่ยไว้

ภาคผนวก ก.3: การออกแบบและวิเคราะห์ Digital Low-Pass Filter

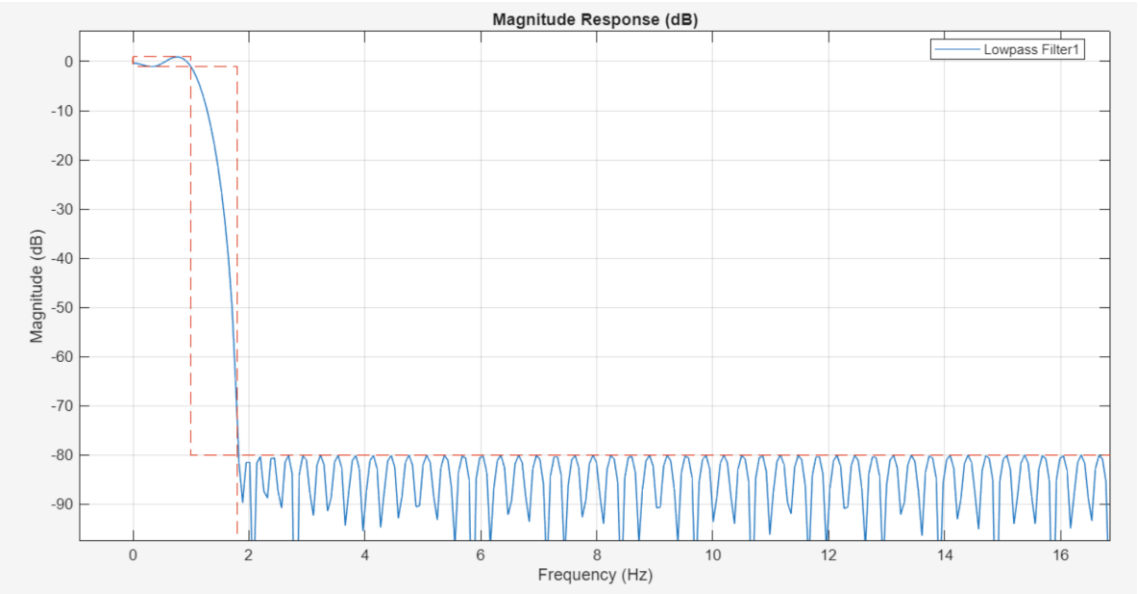
ในการทดลองนี้ ได้มีการออกแบบวงจรกรองความถี่ต่ำผ่าน (Low-Pass Filter) เพื่อกำจัดสัญญาณรบกวน (Noise) จาก Encoder โดยใช้ Block Digital Filter Design ใน Simulink โดยมีการกำหนดค่าพารามิเตอร์และผลการตอบสนองความถี่ดังนี้

1. การตั้งค่าพารามิเตอร์ (Filter Configuration) จากการวิเคราะห์ลักษณะของสัญญาณรบกวนและความต้องการของระบบ ได้เลือกใช้ Filter ชนิด **FIR (Finite Impulse Response)** เนื่องจากมีคุณสมบัติ Linear Phase ทำให้รูปร่างของสัญญาณไม่ผิดเพี้ยน โดยกำหนดค่าดังนี้:

- **Passband Edge Frequency:** 1 Hz (ยอมให้สัญญาณความถี่ต่ำกว่า 1 Hz ผ่าน)
- **Stopband Edge Frequency:** 1.8 Hz (ตัดสัญญาณความถี่สูงกว่า 1.8 Hz ทิ้งทันที)
- **Minimum Stopband Attenuation:** 80 dB (ลดทอนสัญญาณรบกวนลงถึง 80 เดซิเบล)
- **Input Sample Rate:** 1000 Hz (สอดคล้องกับความถี่ในการสุ่มตัวอย่างของระบบ)

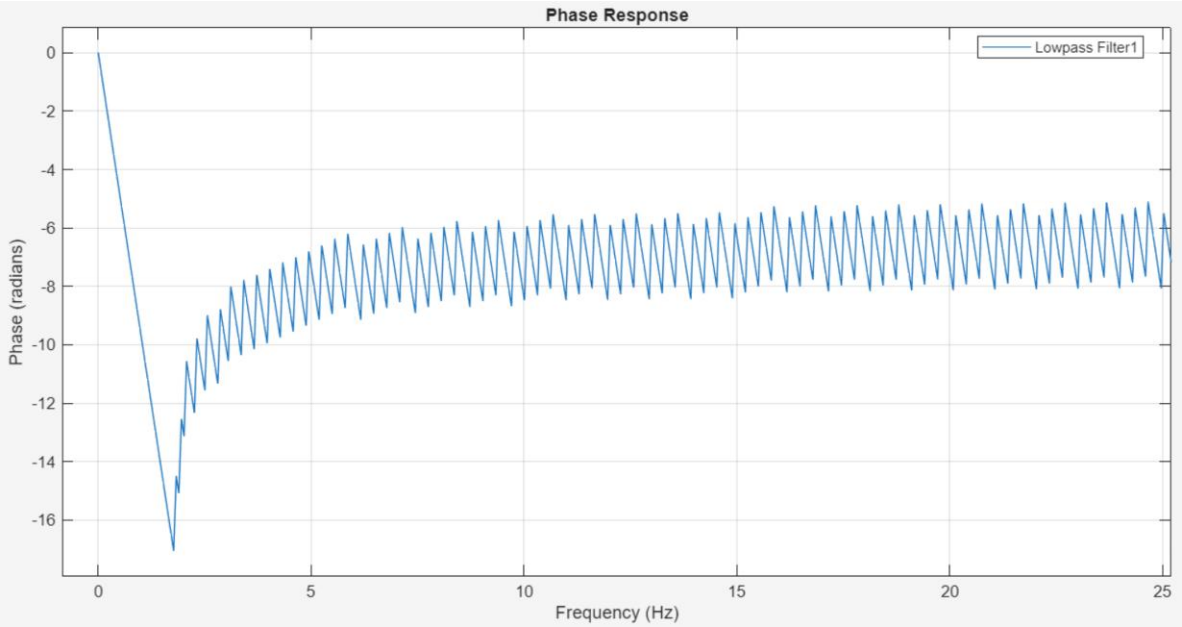
รูปที่ ก-3 การตั้งค่าพารามิเตอร์ของ Digital Low-Pass Filter ใน Simulink

2. การวิเคราะห์ผลตอบสนองความถี่ (Bode Plot Analysis) เพื่อให้มั่นใจว่า Filter ทำงานได้ตามต้องการ จึงทำการวิเคราะห์ Bode Plot ซึ่งประกอบด้วย Magnitude Response และ Phase Response



รูปที่ ก-4 กราฟผลตอบสนองขนาด (Magnitude Response)

- **วิเคราะห์ผล:** จากรูปที่ ก-4 จะเห็นว่ากราฟมีอัตราขยาย (Magnitude) ที่ 0 dB ในช่วงความถี่ 0-1 Hz (Passband) และลดลงอย่างรวดเร็ว (Sharp Cut-off) ในช่วง Transition Band (1 - 1.8 Hz) จนกระทั่งถึงระดับ -80 dB ที่ความถี่ Stopband ซึ่งแสดงให้เห็นว่า Filter สามารถกำจัดสัญญาณรบกวนความถี่สูงได้อย่างมีประสิทธิภาพตามที่ออกแบบไว้



รูปที่ ก-5 กราฟผลตอบสนองเฟส (Phase Response)

- **วิเคราะห์ผล:** จากรูปที่ ก-5 ลักษณะของเฟสมีความเป็นเชิงเส้น (Linear Phase) ตลอดช่วงความถี่ใช้งาน ซึ่งเป็นคุณสมบัติเด่นของ FIR Filter ข้อดีคือสัญญาณ Output จะมีความล่าช้า (Delay) คงที่ และรูปร่างของสัญญาณไม่เกิดการบิดเบี้ยว (Distortion) เมื่อเทียบกับ IIR Filter
- ความชันของกราฟเฟส (Phase Slope) สื่อถึงค่าความล่าช้าของกลุ่มสัญญาณ (Group Delay) การที่กราฟเฟสเป็นเส้นตรง (Linear) หมายความว่าความชันคงที่ ส่งผลให้ทุกย่านความถี่ถูกหน่วงเวลาไปเท่ากันหมด (Constant Delay)

ภาคผนวก ก.4 : การประมวลผลสัญญาณ Load Cell

ก.4 โปรแกรมคำนวณแรงบิดและระบบปรับค่าศูนย์อัตโนมัติ (Load Cell Calibration & Auto-Tare) โค้ดนี้ถูกบรรจุอยู่ใน MATLAB Function Block ภายในส่วน Load Cell Signal Conditioning ทำหน้าที่แปลงค่าดิบ (Raw ADC) ให้เป็นค่า แรง (Force) ที่แม่นยำ เพื่อส่งต่อไปคำนวณหาแรงบิด (Torque)

```
function y = myFunction(x)
% --- MATLAB Function: Load Cell Calculation with Auto-Tare ---
% Input (x): ค่า Raw Analog (ADC/Voltage) จาก Load Cell
% Output (y): ค่าแรงบิด (Torque) หรือน้ำหนักที่คำนวณได้

    % --- 1. ตัวแปร Persistent (Memory) ---
    % ตัวแปรเหล่านี้จะจดจำค่าไว้ใช้ในรอบการคำนวณถัดไป (State Preservation)
    persistent tare_offset;      % เก็บค่า Offset ปัจจุบัน (ค่าเมื่อไม่มีโหลด)
    persistent stable_counter;   % ตัวนับจำนวนรอบที่ค่า Input นิ่งต่อเนื่อง

    % --- 2. การตั้งค่าพารามิเตอร์ (Configuration) ---
    TARE_STABLE_COUNT = 50;      % จำนวนรอบที่ต้องนิ่งก่อนทำการ Tare (50 รอบ @ 1kHz =
0.05IDLE_RANGE_LOW = 60;         % ขอบเขตล่างของช่วง Idle
    IDLE_RANGE_HIGH = 85;        % ขอบเขตบนของช่วง Idle

    % --- 3. การกำหนดค่าเริ่มต้น (Initialization) ---
    if isempty(tare_offset)
        tare_offset = 71;        % ค่า Default Offset (Calibration Point แรก)
        stable_counter = 0;
    end

    % --- 4. ตรรกะการปรับค่าศูนย์อัตโนมัติ (Auto-Tare Logic) ---
    % ตรวจสอบว่าค่า Input อยู่ในช่วงพัก (Idle) หรือไม่
    if (x >= IDLE_RANGE_LOW) && (x <= IDLE_RANGE_HIGH)
        stable_counter = stable_counter + 1; % ถ้านิ่ง ให้เริ่มนับ

        % ถ้านิ่งนานเกินกำหนด ให้ถือว่าเป็นค่าศูนย์ใหม่ (Re-Zeroing)
        if stable_counter >= TARE_STABLE_COUNT
            tare_offset = x;    % อัปเดตค่า Offset
        end
    else
        stable_counter = 0;     % ถ้าค่านอกช่วง (มีโหลด) ให้รีเซ็ตตัวนับ
    end

    % --- 5. การคำนวณแรงบิด (Calibration Equation) ---
    % ใช้สมการ Linear Regression แบบ 2 ช่วง (Piecewise Linear) เพื่อความแม่นยำ
    if x < 630
        % ช่วง Low Range: ใช้ Tare Offset ที่อัปเดตแล้ว
        result = (x - tare_offset) / 1038;
    else
        % ช่วง High Range: ใช้สมการสอบเทียบช่วงสูง
        result = (x + 255) / 1740;
    end

    % --- 6. Output Scaling ---
    y = result * 8;  % ขยายค่าสัญญาณตามอัตราส่วนของแขนโมเมนต์หรือ Gain
end
```

คำอธิบายการทำงานของโปรแกรม (Code Description):

- **1. Auto-Tare Mechanism (ระบบปรับค่าศูนย์อัตโนมัติ):**
  ทำหน้าที่กำจัดค่าความคลาดเคลื่อนเริ่มต้น (Zero Drift) ที่เกิดจากอุณหภูมิหรือแรงเสียดทานตกค้าง
  การทำงาน: ระบบจะตรวจสอบค่า Input ว่านิ่งอยู่ในช่วง ไร้โหลด (No-load range: ADC 60-85) ติดต่อกันครบ 50 รอบการทำงาน หรือไม่ หากเงื่อนไขเป็นจริง ระบบจะบันทึกค่า ADC ปัจจุบันเป็นค่า Tare Offset ใหม่ทันทีเพื่อใช้เป็นจุดอ้างอิงศูนย์ (Zero Point)

- **2. Piecewise Linearization (การชดเชยความเป็นเชิงเส้น):**
  จากการทดสอบสอบเทียบ (Calibration) พบว่า Load Cell มีการตอบสนองที่ไม่เป็นเชิงเส้น (Non-linear Behavior) ตลอดช่วงการวัด จึงได้ออกแบบอัลกอริทึมการคำนวณแบบแยกช่วง (Piecewise Linear) โดยใช้จุดตัดที่ค่า ADC = 630 เพื่อให้ได้ค่าแรงที่ถูกต้องแม่นยำที่สุดตามสมการดังนี้:

  ช่วงโหลดต่ำ (Low Range, ADC < 630): ใช้อัตราขยายชุดที่ 1 (High Sensitivity)

- $F_{calculated} = \frac{ADC - Offset}{1038}$

  ช่วงโหลดสูง (High Range, ADC > 630): ใช้อัตราขยายชุดที่ 2 (Low Sensitivity) พร้อมชดเชยจุดตัดแกน

- $F_{calculated} = \frac{ADC + 255}{1740}$

- **3. Mechanical Advantage Scaling (การชดเชยอัตราทดทางกล):**
  ในขั้นตอนสุดท้าย โปรแกรมจะนำค่า $F_{calculated}$ มาคูณด้วยตัวประกอบ 8

- $F_{output} = F_{calculated} \times 8$

-

  เหตุผล: เนื่องจากชุดทดลองมีการติดตั้ง *Load Cell* ผ่านระบบคานงัด ซึ่งมีอัตราทดระยะทาง **1 : 8** (แรงที่เซนเซอร์รับได้จริงน้อยกว่าแรงที่มอเตอร์ทำได้ 8 เท่า) จึงต้องคูณค่านี้กลับเข้าไปเพื่อให้ได้ค่า **แรงจริง** ที่กระทำต่อระบบ ก่อนส่งค่าออกไปคำนวณ Torque ต่อไป

- **4. Force Calculation:** ผลลัพธ์สุดท้ายจากการคำนวณคือค่า แรง (Force) ในหน่วยนิวตัน (N) ซึ่งจะถูกส่งต่อไปยัง Block คำนวณเพื่อคูณกับรัศมีก้าน (r = 0.11127 m) จนได้ออกมาเป็นค่า แรงบิด (Torque) ที่สมบูรณ์

ภาคผนวก ก.5 : การสร้างสัญญาณควบคุม (Control Signal Generation)

ก.5 โปรแกรมสร้างสัญญาณทดสอบแบบขั้นบันได (Step Input Generation) โค้ดนี้ (MATLAB Function: fcn) ถูกนำมาใช้แทนการปรับค่าด้วยมือ (Manual Slider) เพื่อสร้างสัญญาณควบคุมรูปแบบ Square Wave หรือ

Step Input ที่มีความแม่นยำสูงทางเวลา สำหรับใช้ทดสอบผลตอบสนองของระบบ (Step Response) โดยมีหลักการทำงานดังนี้:

```
function freq_out = fcn(time)
% --- MATLAB Function: Step Signal Generator ---
% Input (time): เวลาจำลองปัจจุบัน (Current Simulation Time)
% Output (freq_out): ค่าสัญญาณความถี่หรือ PWM ที่ส่งออกไป

    % --- 1. ตัวแปร Persistent (Memory) ---
    persistent state            % สถานะปัจจุบัน (0 หรือ 1)
    persistent last_step_time   % เวลาล่าสุดที่มีการสลับสถานะ

    % --- 2. พารามิเตอร์สัญญาณ (Signal Parameters) ---
    val_low       = 0;          % ค่าต่ำสุด (Low Level)
    val_high      = 65540;      % ค่าสูงสุด (High Level) - ค่า Timer
ARR step_duration = 3;          % ระยะเวลาในการคงสถานะ (วินาที)

    % --- 3. การกำหนดค่าเริ่มต้น (Initialization) ---
    if isempty(state)
        state = 0;
        last_step_time = time;
    end

    % --- 4. ตรรกะการสลับสถานะ (Toggle Logic) ---
    % ตรวจสอบว่าเวลาผ่านไปเกินระยะเวลา Step หรือยัง
    if (time - last_step_time) >= step_duration
        state = 1 - state;      % สลับสถานะ (Toggle): 0->1 หรือ 1->0
        last_step_time = time;  % บันทึกเวลาล่าสุดที่สลับ
    end

    % --- 5. การกำหนดค่า Output ---
    if state == 1
        freq_out = val_high;
    else
        freq_out = val_low;
    end
end
```

คำอธิบายหลักการทำงานของโปรแกรม (Detailed Code Logic):

- **1. State Preservation (การจำค่าข้ามรอบการทำงาน):**
  - **Code:** persistent state last_step_time
  - **หลักการ:** โดยปกติฟังก์ชันใน Simulink จะถูกรีเซ็ตค่าตัวแปรทุกครั้งที่จบ Time Step แต่การใช้คำสั่ง persistent ทำให้ตัวแปร state (สถานะปัจจุบัน) และ last_step_time (เวลาที่สลับสวิตช์ครั้งล่าสุด) ยังคงค่าเดิมไว้ได้ ในรอบการทำงานถัดไป (Memory Retention) ซึ่งจำเป็นมากสำหรับระบบที่ต้องทำงานต่อเนื่องตามเวลา (Time-dependent System)
- **2. Non-blocking Timer (ระบบจับเวลาแบบไม่อาศัย Loop):**

- **Code:** if (time - last_step_time) >= step_duration
- **หลักการ:** โปรแกรมใช้วิธีเปรียบเทียบ "ผลต่างเวลา" (Time Delta) ระหว่างเวลาปัจจุบัน (time) กับเวลาล่าสุดที่เกิดเหตุการณ์ (last_step_time) แทนการใช้คำสั่ง delay หรือ wait การทำเช่นนี้ทำให้ระบบทำงานได้แบบ Non-blocking (ไม่หยุดชะงัก) ซึ่งเป็นหัวใจสำคัญของการจำลองระบบ Real-time Simulation

- **3. Mathematical Toggling (การสลับสถานะด้วยสมการคณิตศาสตร์):**
  - **Code:** state = 1 – state;
  - **หลักการ:** เป็นเทคนิคทางคณิตศาสตร์ในการสลับค่า Binary (Toggle) ระหว่าง 0 และ 1
    - หาก state เดิมคือ 0 -> 1 - 0 = 1 (ON)
    - หาก state เดิมคือ 1 -> 1 - 1 = 0 (OFF)
- **4. PWM Output Mapping (การแปลงเป็นสัญญาณควบคุม):**
  - **Code:** ส่งค่า val_high (65540) หรือ val_low (0)
  - **ความหมาย:** ค่า 65540 ที่กำหนดในตัวแปร val_high คือค่าสูงสุดของ Counter (Auto-Reload Register: ARR) ในบอร์ด STM32 ซึ่งเทียบเท่ากับการจ่าย Duty Cycle 100% (แรงดันเต็มพิกัด) การสลับค่าระหว่าง 0 และ 65540 ทันทีจึงเป็นการสร้างสัญญาณ Step Input ที่สมบูรณ์

วัตถุประสงค์การใช้งาน (Application & Analysis):

การสร้างสัญญาณแบบนี้มีวัตถุประสงค์เพื่อวิเคราะห์ ผลตอบสนองชั่วครู่ (Transient Response) ของมอเตอร์ ได้แก่:

1. **Rise Time (t_r):** เพื่อดูว่ามอเตอร์ใช้นานแค่ไหนในการเร่งจากหยุดนิ่งไปสู่ความเร็วสูงสุด (บอกถึงความเฉื่อยและแรงบิดของระบบ)
2. **Overshoot (M_p):** เพื่อดูว่าความเร็วพุ่งเกินค่าเป้าหมายหรือไม่เมื่อได้รับแรงดันเต็มที่ทันที

ภาคผนวก ก.6 : การประมวลผลและชดเชยค่ากระแสไฟฟ้า (Current Sensor Processing)

ก.6 โปรแกรมแปลงค่า ADC และปรับแก้ความถูกต้อง (ADC to Ampere Conversion & Correction)

โค้ดนี้ถูกบรรจุอยู่ใน MATLAB Function Block (adc_to_amps_fcn) ภายในส่วน Current Sensor Signal Conditioning ทำหน้าที่แปลงค่าดิบ (Raw ADC) จากเซนเซอร์ WCS1700 ให้เป็นค่ากระแสไฟฟ้า (Ampere) ที่มีความแม่นยำสูง โดยแบ่งกระบวนการทำงานออกเป็น 3 ขั้นตอนหลัก ดังนี้:

```
function final_amps = fcn(adc_in)

    % --- ส่วนที่ 1: คำนวณค่าเดิม (ตามที่คุณใช้อยู่ปัจจุบัน) ---
    adc_zero = 2084;
    adc_drop = adc_zero - adc_in;

    A = 2.296e-8;
    B = 2.53e-5;
    C = 0.00013;
    D = 0.058;

    % ค่าที่อ่านได้ปัจจุบัน
    current_reading = A*(adc_drop^3) + B*(adc_drop^2) + C*adc_drop + D;

    % --- ส่วนที่ 2: ปรับจูนค่าใหม่ (Correction Stage) ---
    % ใช้สมการนี้เพื่อกดกราฟลง ให้ตรงกับ PSU

    % สมการ: y = -0.0575*x^2 + 0.995*x
    final_amps = -0.0575 * (current_reading^2) + 0.995 *
current_reading;
    % ป้องกันค่าติดลบเล็กน้อยตอนไม่มีกระแส
    if final_amps < 0
        final_amps = 0;
    end

end
```

คำอธิบายการทำงานของโปรแกรม (Code Description):

- **1. Baseline Calculation (การคำนวณค่ากระแสเบื้องต้น):**
  ขั้นตอนแรกเป็นการแปลงค่าความต่างศักย์ (Voltage Drop) ที่อ่านได้จาก ADC ให้เป็นค่ากระแสประมาณการ โดยใช้สมการพหุนามกำลังสาม (Cubic Polynomial) เพื่อรองรับความไม่เป็นเชิงเส้น (Non-linearity) ของเซนเซอร์ WCS1700 ที่ย่านกระแสต่ำ
  - **การหาค่าความต่าง:** คำนวณ adc_drop จากจุดอ้างอิงศูนย์ (adc_zero = 2084)
  - สมการคำนวณ: ใช้สัมประสิทธิ์ A, B, C, D จากการทดลองช่วงแรก
  - $\mathrm{I_{raw}} = \mathrm{A(drop)^3} + \mathrm{B(drop)^2} + \mathrm{C(drop)} + \mathrm{D}$

- **2. Secondary Correction (การปรับแก้ค่าความผิดพลาดเทียบกับ PSU):**
  จากการทดสอบพบว่าค่า I_{raw} ที่ได้ยังมีความคลาดเคลื่อนเมื่อเทียบกับค่าจริงที่อ่านได้จาก Power Supply (เช่น อ่านได้ 3.1A แต่ค่าจริงคือ 2.5A) จึงได้นำผลการเปรียบเทียบมาทำ Curve Fitting อีกครั้งด้วยสมการพหุนามกำลังสอง (Quadratic Equation) เพื่อกดกราฟให้ทับซ้อนกับค่าจริงมากที่สุด
  - สมการปรับแก้ (Correction Formula):
  - $I_{final} = -0.0575(I_{raw})^2 + 0.995(I_{raw})$
- **3. Zero Clamping (การกำจัดสัญญาณรบกวนช่วงเดินเครื่องเปล่า):**
  เนื่องจากสัญญาณ ADC อาจมี Noise เล็กน้อยที่ทำให้ผลการคำนวณติดลบเมื่อมอเตอร์หยุดหมุน โปรแกรมจึงมีเงื่อนไขตรวจสอบว่า:
  - **Logic:** หาก final_amps < 0 ให้บังคับค่าเป็น 0 ทันที
  - **ประโยชน์:** ช่วยให้กราฟกระแสไฟฟ้าเริ่มต้นที่ 0 สนิท (Clean Zero) ไม่เกิดค่าติดลบที่ผิดธรรมชาติเมื่อแสดงผลบนกราฟ

ภาคผนวก ก.7 : การสร้างกราฟคุณลักษณะสมบัติของมอเตอร์ (Motor Characteristics Plotting)

ก.7 โปรแกรม Python สำหรับการวิเคราะห์และแสดงผลกราฟ (Python Script for Data Visualization)

เนื่องจากข้อมูลดิบที่ได้จากการทดลองมีจำนวนจุดจำกัด (Discrete Data Points) การนำเสนอข้อมูลให้เห็นแนวโน้มที่ชัดเจนจึงต้องใช้กระบวนการทางคณิตศาสตร์เข้ามาช่วย โปรแกรมด้านล่างนี้เขียนด้วยภาษา Python โดยใช้ไลบรารี `Matplotlib` และ `SciPy` เพื่อสร้างกราฟ Complete DC Motor Characteristics ที่มีความละเอียดสูงและอ่านค่าได้ง่าย

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import make_interp_spline

# 1. ป้อนข้อมูลดิบ (Raw Data Points)
# แปลงเป็น NumPy array เพื่อให้ง่ายต่อการคำนวณ
torque = np.array([0, 0.05033333333, 0.13, 0.2433333333])
speed = np.array([1876.666667, 1488.479452, 874.0639269, 0])
current = np.array([0.058, 0.461769863, 1.100849315, 2.01])
power = np.array([0, 1.248668874, 1.893805175, 0])
efficiency = np.array([0, 22.53411229, 14.33594006, 0])

# 2. สร้างข้อมูลให้ Smooth (Spline Interpolation)
# สร้างแกน X ใหม่ที่มีความละเอียดสูง (เช่น 300 จุด)
t_smooth = np.linspace(torque.min(), torque.max(), 300)

# สร้างฟังก์ชัน Spline สำหรับข้อมูลแต่ละชุด (k=3 คือ cubic spline ที่ให้ความโค้งสวยงาม)
spl_speed = make_interp_spline(torque, speed, k=3)
spl_current = make_interp_spline(torque, current, k=3)
spl_power = make_interp_spline(torque, power, k=3)
spl_eff = make_interp_spline(torque, efficiency, k=3)

# คำนวณค่า Y ใหม่จากฟังก์ชัน Spline บนแกน X ที่ละเอียดขึ้น
speed_smooth = spl_speed(t_smooth)
current_smooth = spl_current(t_smooth)
power_smooth = spl_power(t_smooth)
eff_smooth = spl_eff(t_smooth)

# ป้องกันค่าติดลบ (Clip negative values) สำหรับ Power และ Efficiency เพื่อความถูกต้องทางฟิสิกส์
# เนื่องจาก Spline อาจมีการแกว่งตัวไปในค่าลบได้ในบางกรณี
power_smooth = np.clip(power_smooth, 0, None)
eff_smooth = np.clip(eff_smooth, 0, None)

# 3. ตั้งค่าการพล็อต (Plotting Setup)
fig, ax1 = plt.subplots(figsize=(10, 7))
plt.subplots_adjust(right=0.75)

plt.title('Complete DC Motor Characteristics (Smooth Curve Experimental Data)', fontsize=14,
weight='bold', y=1.02)
plt.grid(True, which='major', linestyle='--', alpha=0.5)

# --- แกนที่ 1: Speed (สีน้ำเงิน) ---
color1 = 'tab:blue'
ax1.set_xlabel('Torque (Nm)', fontsize=12)
ax1.set_ylabel('Speed (RPM)', color=color1, fontsize=12, weight='bold')
# พล็อตเส้น Smooth
ax1.plot(t_smooth, speed_smooth, linestyle='-', color=color1, linewidth=2, label='Speed')
# พล็อตจุดข้อมูลจริง (Markers) เพื่อแสดงที่มาของข้อมูล
ax1.scatter(torque, speed, color=color1, marker='o', s=30, zorder=5)
ax1.tick_params(axis='y', labelcolor=color1)

# --- แกนที่ 2: Current (สีแดง) ---
ax2 = ax1.twinx()
color2 = 'tab:red'
ax2.set_ylabel('Current (A)', color=color2, fontsize=12, weight='bold')
ax2.plot(t_smooth, current_smooth, linestyle='--', color=color2, linewidth=2, label='Current')
ax2.scatter(torque, current, color=color2, marker='o', s=30, zorder=5)
ax2.tick_params(axis='y', labelcolor=color2)

# --- แกนที่ 3: Power Output (สีน้ำตาล/ทอง) ---
ax3 = ax1.twinx()
color3 = '#A09E88'
ax3.spines["right"].set_position(("axes", 1.15))
ax3.set_ylabel('Power Output (W)', color=color3, fontsize=12, weight='bold')
ax3.plot(t_smooth, power_smooth, linestyle='-', color=color3, linewidth=2, label='Power')
ax3.scatter(torque, power, color=color3, marker='s', s=30, zorder=5)
ax3.tick_params(axis='y', labelcolor=color3)

# --- แกนที่ 4: Efficiency (สีเขียว) ---
ax4 = ax1.twinx()
color4 = 'tab:green'
ax4.spines["right"].set_position(("axes", 1.3))
ax4.set_ylabel('Efficiency (%)', color=color4, fontsize=12, weight='bold')
ax4.plot(t_smooth, eff_smooth, linestyle='-', color=color4, linewidth=2.5, label='Efficiency')
ax4.scatter(torque, efficiency, color=color4, marker='^', s=50, zorder=5)
ax4.tick_params(axis='y', labelcolor=color4)
ax4.set_ylim(0, 30)

# แสดงค่าตัวเลขตรงจุด Max Efficiency (คำนวณจากเส้น Smooth)
max_eff_idx = np.argmax(eff_smooth)
max_eff_val = eff_smooth[max_eff_idx]
max_eff_torque = t_smooth[max_eff_idx]
plt.text(max_eff_torque, max_eff_val + 1,
         f'Max Eff\n~{max_eff_val:.2f}%',
         color='green', ha='center', weight='bold')

# รวม Legend
lines = [ax1.get_lines()[0], ax2.get_lines()[0], ax3.get_lines()[0], ax4.get_lines()[0]]
labels = [l.get_label() for l in lines]
ax1.legend(lines, labels, loc='upper center', bbox_to_anchor=(0.5, -0.12), ncol=4, frameon=False)

# แสดงกราฟ
plt.show()
```

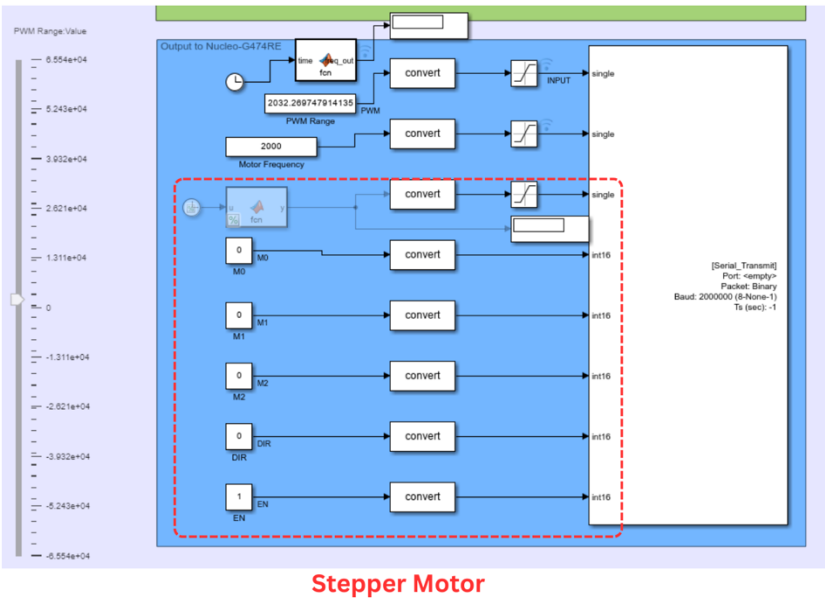
คำอธิบายหลักการทำงานของโปรแกรม (Code Description):

- **1. Cubic Spline Interpolation (การสร้างเส้นแนวโน้มแบบต่อเนื่อง):** ข้อมูลที่ได้จากการทดลองมีลักษณะเป็นจุดไม่ต่อเนื่อง (Discrete) โปรแกรมจึงใช้ฟังก์ชัน make_interp_spline จากไลบรารี SciPy มาสร้างสมการพหุนามกำลังสาม (Cubic Polynomial) เพื่อเชื่อมจุดข้อมูลแต่ละจุดเข้าด้วยกัน
    - **วัตถุประสงค์:** เพื่อสร้างเส้นกราฟที่ราบเรียบ (Smooth Curve) ซึ่งช่วยให้สามารถประมาณค่าแนวโน้มของระบบ (Trend Estimation) ในช่วงระหว่างจุดข้อมูลที่ไม่ได้ทำการวัดจริงได้ และทำให้สามารถระบุจุดยอด (Peak) ของกราฟประสิทธิภาพได้แม่นยำขึ้น
- **2. Multi-Axis Visualization (การแสดงผลกราฟหลายแกน):** เนื่องจากตัวแปรทั้ง 4 ตัว (ความเร็ว, กระแส, กำลัง, ประสิทธิภาพ) มีหน่วยและขนาดข้อมูล (Scale) ที่แตกต่างกันอย่างมาก การใช้แกน Y เพียงแกนเดียวจะทำให้กราฟดูยาก
    - **เทคนิค:** ใช้คำสั่ง ax.twinx() เพื่อสร้างแกน Y เพิ่มเติมซ้อนทับบนกราฟเดิม และใช้ spines ในการขยับตำแหน่งแกนที่ 3 และ 4 ออกไปด้านขวา ทำให้สามารถเปรียบเทียบความสัมพันธ์ของตัวแปรทั้งหมดได้ในภาพเดียว (Comprehensive View)
- **3. Physical Constraint Clipping (การจำกัดค่าตามความเป็นจริงทางฟิสิกส์):**
    - **Logic:** np.clip(data, 0, None)
    - **เหตุผล:** คณิตศาสตร์ของ Spline Interpolation อาจเกิดการแกว่งตัว (Oscillation) ไปสู่ค่าติดลบในช่วงที่ข้อมูลมีการเปลี่ยนแปลงรวดเร็ว ซึ่งขัดแย้งกับความเป็นจริงทางฟิสิกส์ (เช่น ประสิทธิภาพติดลบไม่ได้) จึงต้องมีการบังคับค่าต่ำสุดให้เป็น 0 เพื่อความถูกต้องของข้อมูล

Lab 2.2 Stepper Motor

ภาคผนวก ข: การสร้างสัญญาณควบคุม Stepper Motor

ข.1 โครงสร้างไฟล์ Simulink สำหรับการควบคุม Stepper Motor ส่วนนี้แสดงภาพรวมของ Model Simulink ที่ใช้ในการทดลองควบคุม Stepper Motor ผ่านบอร์ด STM32G474RE โดยมีการออกแบบระบบควบคุมแบบ Open-Loop ที่แบ่งส่วนการทำงานหลักออกเป็น 2 ส่วน ดังแสดงในรูปที่ ข-1



รูปที่ ข-1 ภาพรวมของ Model Simulink (Main View) สำหรับการควบคุม Stepper Motor

คำอธิบาย Block Diagram การทำงานของระบบ

1. Driver Configuration & Logic Control (ส่วนกำหนดค่าไดรเวอร์และลอจิก) ส่วนนี้ทำหน้าที่กำหนดสถานะลอจิก (0 หรือ 1) ไปยังขา GPIO ของบอร์ด STM32 เพื่อตั้งค่าการทำงานของ Stepper Driver (DRV8825)

- **Stepping Setup (M0, M1, M2):**
- ทำหน้าที่กำหนด ความละเอียดในการหมุน (Step Resolution) ของมอเตอร์ ว่าต้องการให้หมุนทีละกี่องศาต่อพัลส์ (เช่น Full Step, Half Step, 1/4 Step, ไปจนถึง 1/32 Step)
- **Direction (DIR):** กำหนดทิศทางการหมุนของมอเตอร์ (0 = ตามเข็ม, 1 = ทวนเข็ม หรือตาม Datasheet ของ Driver)
- **Enable (EN):**

- ค่า **1 (High):** สั่งเปิดการทำงานของภาคขยายกำลัง (Energize Coils) ทำให้มอเตอร์มีแรงบิดล็อกแกน (Holding Torque)
- ค่า **0 (Low):** ปล่อยลอยมอเตอร์ (Free Run)

3. Serial Transmission (ส่วนส่งข้อมูลกลับ)

- **Serial Transmit Block:** ทางด้านขวาสุดของ Model ทำหน้าที่ส่งข้อมูลสถานะการทำงาน (เช่น ความถี่ปัจจุบัน) กลับมายังคอมพิวเตอร์ผ่านพอร์ต Serial เพื่อใช้ในการเก็บข้อมูล (Data Logging) และวิเคราะห์ผลต่อไป

ข.2 โปรแกรมสร้างสัญญาณความถี่แบบขั้นบันได (Stepped Frequency Generator)

ในการทดลองเพื่อหาความสัมพันธ์ระหว่างความถี่และความเร็วรอบ (Frequency vs Speed) หรือการทดสอบหาจุด Loss Step ได้มีการเขียนโปรแกรม MATLAB Function เพื่อสร้างสัญญาณความถี่ที่เพิ่มขึ้นเป็นขั้นบันไดโดยอัตโนมัติตามเวลาที่ผ่านไป ดังแสดงในโค้ดด้านล่าง

```
function y = fcn(u)
% --- MATLAB Function: Stepped Frequency Generator ---
% Input (u): เวลาจำลองปัจจุบัน (Simulation Time)
% Output (y): ค่าความถี่ (Frequency) ที่ส่งไปยัง Stepper Motor Driver

    % --- 1. ตัวแปร Persistent ---
    persistent freq_index    % ตัวแปรสำหรับเก็บลำดับขั้นความถี่ปัจจุบัน

    % --- 2. การกำหนดค่าเริ่มต้น (Initialization) ---
    if isempty(freq_index)
        freq_index = 0;
    end

    % --- 3. การตั้งค่าพารามิเตอร์ (Configuration) ---
    step_duration = 0.5;     % ระยะเวลาในการคงความถี่ไว้ในแต่ละขั้น (0.5
วินาที)steps_per_cycle = 25;    % จำนวนขั้นความถี่ทั้งหมดใน 1 รอบ
    freq_multiplier = 60;    % ขนาดของความถี่ที่เพิ่มขึ้นต่อ 1 ขั้น (Hz)

    % --- 4. การคำนวณลำดับขั้น (Step Calculation) ---
    % คำนวณว่า ณ เวลา u วินาที ควรอยู่ที่ขั้นที่เท่าไหร่
    step_number = floor(u / step_duration);

    % --- 5. การวนรอบสัญญาณ (Looping) ---
    % ใช้ Modulo เพื่อให้ค่า index วนกลับมาที่ 0 เมื่อครบ 25 ขั้น (0 ถึง 24)
    freq_index = mod(step_number, steps_per_cycle);

    % --- 6. การกำหนดค่า Output ---
    % คำนวณความถี่จากลำดับขั้น (เช่น ขั้นที่ 10 * 60 = 600 Hz)
    y = freq_index * freq_multiplier;
end
```

คำอธิบายการทำงาน:

1. **Time-Based Step Generation:** โปรแกรมรับค่าเวลา (u) จาก Simulink Clock และนำมาหารด้วยระยะเวลาต่อขั้น (step_duration = 0.5 วินาที) จากนั้นใช้ฟังก์ชัน floor ปัดเศษลงเพื่อให้ได้ค่าเป็นจำนวนเต็ม ซึ่งเปรียบเสมือน "หมายเลขขั้น" (Step Number) ณ เวลานั้นๆ
2. **Frequency Ramp-up Pattern:** ค่าความถี่ขาออก (y) จะแปรผันตรงกับหมายเลขขั้น โดยเพิ่มขึ้นทีละ 60 Hz ในทุกๆ 0.5 วินาที (ตามตัวคูณ freq_multiplier)
3. **Cycle Reset:** เมื่อทำงานครบ 25 ขั้น (หรือประมาณ 12.5 วินาที) คำสั่ง mod จะทำการรีเซ็ตค่ากลับมาเริ่มต้นที่ 0 Hz ใหม่อีกครั้ง ทำให้สามารถทดสอบซ้ำได้อย่างต่อเนื่องโดยไม่ต้องหยุด Simulation

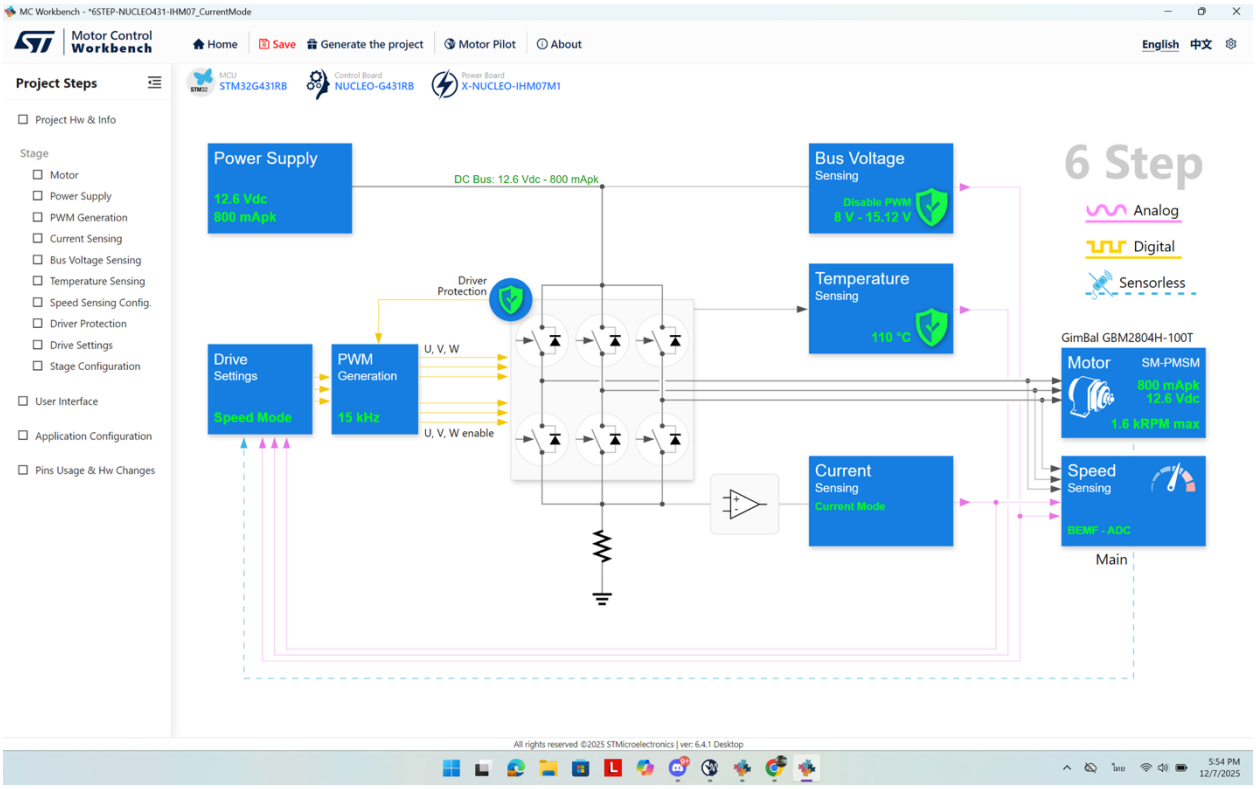
วัตถุประสงค์การใช้งาน:

Code นี้ช่วยให้สามารถกวาดช่วงความถี่ (Frequency Sweep) จากต่ำไปสูงแบบอัตโนมัติ เพื่อสังเกตพฤติกรรมของ Stepper Motor ว่าที่ความถี่ใดเริ่มเกิดอาการสั่น คราง หรือเกิดการ Loss Step โดยไม่ต้องมานั่งปรับค่าความถี่ด้วยมือทีละค่า

Lab 2.3 Brushless DC Motor

ภาคผนวก ค: การตั้งค่าพารามิเตอร์และการใช้งาน ST MC Workbench

ค.1 การกำหนดค่าระบบและการตั้งค่าพารามิเตอร์ (Project System Configuration) ส่วนนี้แสดงภาพรวมการตั้งค่าระบบ (System View) ในโปรแกรม ST MC Workbench สำหรับควบคุมมอเตอร์ BLDC (GimBal GBM2804H-100T) โดยใช้บอร์ดขับ X-NUCLEO-IHM07M1 ร่วมกับ STM32G431RB ดังปรากฏในรูปที่ ค-1



รูปที่ ค-1 หน้าต่าง Project Overview แสดงผังการเชื่อมต่อและการตั้งค่าพารามิเตอร์หลักของระบบ

คำอธิบายการกำหนดค่าพารามิเตอร์ (Parameter Configuration Description): เพื่อให้ระบบควบคุมสามารถขับเคลื่อนมอเตอร์ Gimbal ได้อย่างแม่นยำและปลอดภัย ได้มีการกำหนดค่าพารามิเตอร์สำคัญตามที่ปรากฏในรูปที่ ค-1 ดังนี้:

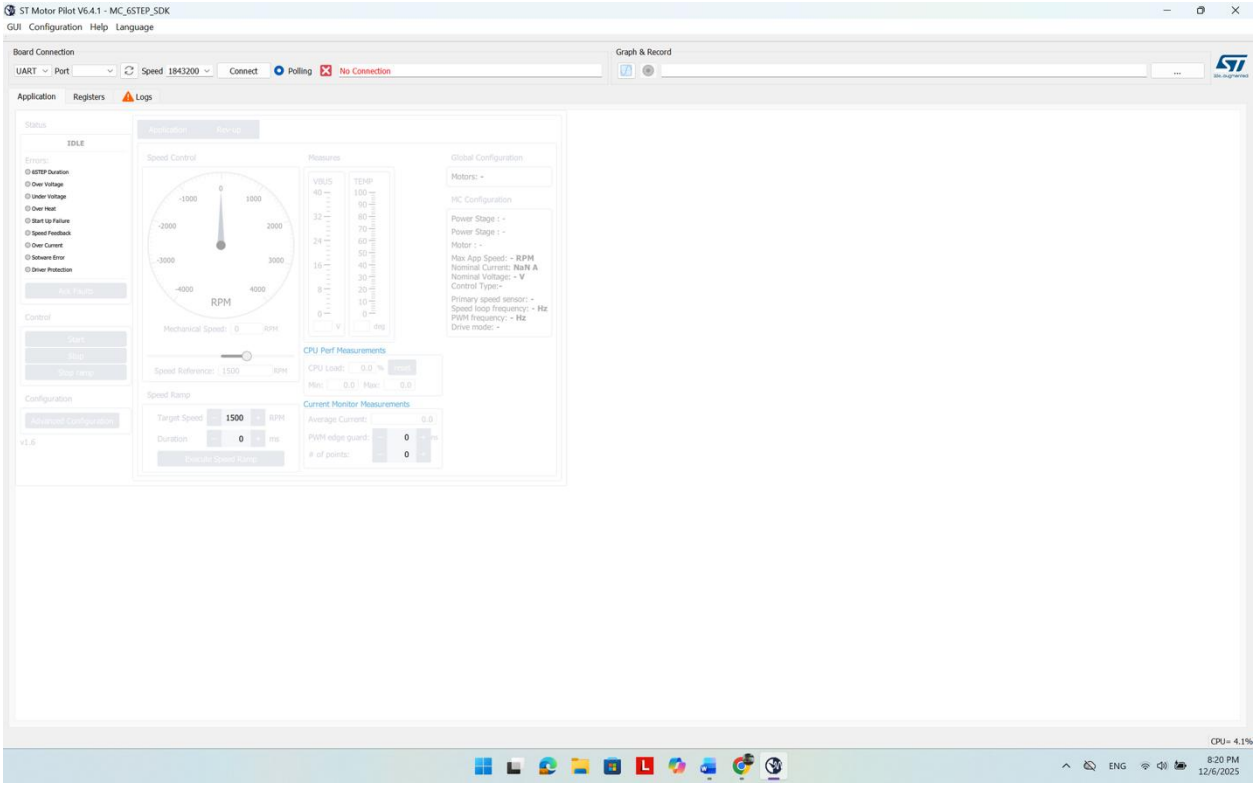
- **Motor Parameters (คุณสมบัติมอเตอร์):**
  - **Motor Type:** กำหนดเป็น **SM-PMSM** (Surface-Mounted PMSM)
  - **Current & Speed Limits:** ตั้งค่ากระแสสูงสุด (Nominal Current) ไว้ที่ 800 mApk (0.8 A) และความเร็วสูงสุดที่ 1.6 kRPM
  - **เหตุผล:** เพื่อให้สอดคล้องกับสเปกของมอเตอร์ GimBal GBM2804H-100T ซึ่งเป็นมอเตอร์สำหรับงานความเร็วต่ำแต่ต้องการความละเอียดสูง และป้องกันขดลวดเสียหายจากกระแสเกิน
- **PWM Configuration (การสร้างสัญญาณควบคุม):**

- **Switching Frequency:** กำหนดความถี่ PWM ไว้ที่ 15 kHz
- **เหตุผล:** เป็นความถี่มาตรฐานสำหรับมอเตอร์ชนิดนี้ ซึ่งเพียงพอต่อการสร้างรูปคลื่นกระแสที่เสถียร โดยไม่สร้างความร้อนสะสมที่ตัวขับ (Driver) มากเกินไป และหลีกเลี่ยงย่านความถี่เสียงรบกวนบางช่วง

- **Driving Topology (รูปแบบการขับเคลื่อน):**
  - **Control Algorithm:** เลือกโหมด 6-Step (Trapezoidal Control)
  - **เหตุผล:** เหมาะสมกับการควบคุมเบื้องต้นสำหรับมอเตอร์ BLDC ซึ่งให้การตอบสนองแรงบิดที่ดีและประมวลผลได้รวดเร็วบนไมโครคอนโทรลเลอร์
- **Sensorless Position Sensing (การระบุตำแหน่งโรเตอร์):**
  - **Method:** เลือกใช้ BEMF Sensing via ADC
  - **เหตุผล:** ใช้วงจรแบ่งแรงดัน (Voltage Divider) และ ADC ของไมโครคอนโทรลเลอร์ในการตรวจวัดแรงดันย้อนกลับ (Back-EMF) เพื่อหาจุด Zero-Crossing สำหรับการสลับเฟสการทำงานโดยไม่ต้องใช้เซนเซอร์ภายนอก
- **Power Stage Protection (ระบบป้องกันภาคกำลัง):**
  - **Bus Voltage Protection:** ตั้งค่าแรงดันบัส 12.6 Vdc และกำหนดจุดตัดการทำงาน (Over-Voltage Threshold) ที่ 15.12 V
  - **Temperature Protection:** กำหนดจุดตัดการทำงานเมื่ออุณหภูมิบอร์ดสูงเกิน 110 °C
  - **เหตุผล:** ค่า 12.6 V สอดคล้องกับแหล่งจ่ายไฟ (หรือแบตเตอรี่ 3S) ที่ใช้ ส่วนค่า Threshold 15.12 V (เผื่อ 20%) และอุณหภูมิ 110 °C ถูกตั้งไว้เพื่อป้องกันวงจรภาคกำลัง (Power Stage) เสียหายหากเกิดสภาวะผิดปกติ

ค.2 หน้าต่างควบคุมและสั่งงานมอเตอร์ (Control & Monitoring Interface)

ส่วนนี้แสดงหน้าต่าง Speed Control ในโปรแกรม Motor Pilot ซึ่งทำหน้าที่รับ-ส่งคำสั่งควบคุมมอเตอร์แบบ Real-time ผ่านพอร์ต UART โดยในการทดลองนี้ได้เน้นการใช้งานฟังก์ชัน Speed Ramp เพื่อควบคุมอัตราเร่งของมอเตอร์



รูปที่ ค-2 หน้าต่าง Speed Control สำหรับกำหนดค่าความเร็วและระยะเวลาในการเร่งรอบ (Ramp)

ขั้นตอนและหลักการกำหนดค่าสั่งงาน (Operation Description):

- **Speed Ramp Configuration (การตั้งค่าการไต่ระดับความเร็ว):** ในการสั่งให้มอเตอร์หมุนไปยังความเร็วเป้าหมาย ได้กำหนดพารามิเตอร์ดังนี้:
    - **Target Speed:** ระบุความเร็วรอบที่ต้องการทดสอบ (เช่น 1500 RPM หรือค่าอื่นๆ ตามเงื่อนไขการทดลอง)
    - **Duration:** กำหนดค่าระยะเวลาในการไต่ระดับเท่ากับ **1000 ms (1 วินาที)**
    - **Execute Speed Ramp:** ปุ่มสำหรับส่งคำสั่งเริ่มทำงาน
- **เหตุผลทางวิศวกรรม (Engineering Justification):**
    - **Limiting Inrush Current:** การกำหนด Duration ไว้ที่ 1000 ms แทนการสั่งงานแบบทันทีทันใด (Step Change, 0 ms) เป็นการจำกัดกระแสกระชากเริ่มต้น (Inrush Current) ป้องกันไม่ให้กระแสสูงเกินพิกัด (Overcurrent Trip) จนระบบตัดการทำงาน
    - **Sensorless Stability:** สำหรับมอเตอร์ BLDC แบบ Sensorless ที่ต้องอาศัยการวัด Back-EMF ในการระบุตำแหน่งโรเตอร์ การเปลี่ยนความเร็วรอบอย่างรุนแรงทันทีอาจทำให้ Observer คำนวณตำแหน่งผิดพลาดและเกิดการหลุดซิงค์ (Loss of Synchronism) หรือมอเตอร์หยุดหมุน (Stall) ได้ การใช้ Ramp 1000 ms จึงช่วยรักษาเสถียรภาพของระบบควบคุมในขณะเปลี่ยนความเร็ว