รายงานการวิจัย

โครงการวิจัยร่วมภาครัฐและเอกชน

การพัฒนาชุดทดลองระบบควบคุมอัตโนมัติเพื่อการวิจัย และฝึกอบรมด้านเทคโนโลยีควบคุมชั้นสูง ปีที่ 3

คณะผู้วิจัย

ผศ.ดร.มานพ วงศ์สายสุวรรณ

รศ.ดร.วราภรณ์ เชาว์วิศิษฐ

รศ.ดร.วัชรพงษ์ โขวิฑูรกิจ

รศ.ดร.เดวิด บรรเจิดพงศ์ชัย

อ.ดร.สุชิน อรุณสวัสดิ์วงศ์

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

หน้า

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญภาพ

ภาพประกอบ **หน้า**

ภาพประกอบ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

อุตสาหกรรมของประเทศไทยยังต้องการบุคลากรในสายเทคโนโลยีอีกเป็นจำนวนมาก แต่การ ฝึกอบรมบุคลากรในสายงานนี้ นอกจากจะต้องใช้อาจารย์ผู้สอนที่มีความรู้ความชำนาญแล้ว ยัง จำเป็นต้องใช้อุปกรณ์ชุดทดลองที่มีคุณภาพอีกเป็นจำนวนมาก ปัจจุบันประเทศไทยยังนำเข้า อุปกรณ์เหล่านี้จากต่างประเทศเป็นจำนวนมาก ซึ่งส่วนใหญ่มีราคาค่อนข้างสูง โดยเฉพาะ อุปกรณ์ทางด้านวิศวกรรมระบบควบคุม

วัตถุประสงค์ของโครงการวิจัยนี้คือ การพัฒนาต้นแบบชุดทดลองสำหรับการฝึกอบรมด้าน ระบบควบคุมอัตโนมัติโดยเฉพาะ เพื่อให้เหมาะสำหรับการเรียนการสอน และยังใช้ใน งานวิจัยได้อีกด้วย ในบรรดาชุดทดลองมากมายที่มีใช้กันอยู่ในปัจจุบัน ชุดทดลองระบบ เพนดูลัมผกผันเป็นหนึ่งในชุดทดลองที่ได้รับความสนใจและได้รับความนิยมอยู่เป็นเวลานาน เนื่องจากมีโครงสร้าง เชิงกลที่ไม่ซับซ้อน แต่ก็มีพฤติกรรมที่ไม่เป็นเชิงเส้น (non-linearity) สูง ทำให้เป็น โจทย์ที่ท้าทายสำหรับการควบคุม อีกทั้งยังสามารถออกแบบกลยุทธ์ในการควบคุมได้หลาก หลายรูปแบบ

ถึงแม้ว่าชุดทดลองเหล่านี้ เราสามารถซื้อหาได้จากต่างประเทศ แม้ว่าจะมีราคาสูงอยู่บ้าง แต่ ชุดทดลองที่นำเข้ามักจะใช้ซอฟต์แวร์ที่มีลิขสิทธิ์และมีข้อจำกัดในการดัดแปลงแก้ไข ทำให้ สามารถใช้งานหรือทดสอบตัวควบคุมแบบต่าง ๆ ได้เฉพาะที่มีเตรียมไว้ให้ในคู่มือเท่านั้น เพราะ ไม่มีการเปิดเผยรหัสต้นฉบับ (source code)

เพื่อแก้ไขอุปสรรคดังกล่าว คณะวิจัยจึงศึกษาการทำชุดทดลองต้นฉบับขึ้นใช้เอง โดยติดต่อกับ บริษัทเอกชนที่มีศักยภาพในการพัฒนาชุดทดลองที่ใช้มอเตอร์ เพื่อให้เราสามารถสร้างฮาร์ดแวร์ ได้จากวัสดุภายในประเทศและร่วมพัฒนาซอฟต์แวร์และคอร์สแวร์สำหรับใช้ในการเรียน การสอนต่อไป โดยเริ่มจากชุดทดลองเพนดูลัมเป็นกรณีแรก

บทที่ 2
แบบจำลองทางคณิตศาสตร์ของระบบเพนดูลัมผกผันแบบหมุน

ในบทนี้จะนำเสนอการหาสมการพลวัต ของ ระบบเพนดูลัมผกผันแบบหมุน เพื่อนำสมการที่ได้ช่วยในการพัฒนาตัวควบคุมที่จะใช้ในการควบคุมระบบต่อไป ในบทนี้จะเริ่มด้วยการใช้ สมการลากรองจ์ ในการวิเคราะห์ ซึ่งจะได้แบบจำลองของระบบที่ไม่เป็นเชิงเส้น จากนั้นจะทำให้เป็นเชิงเส้น เพื่อความสะดวกในการใช้ออกแบบตัวควบคุมในภายหลัง

2.1 แบบจำลองของระบบแบบสัญญาขาเข้าเป็นแรงบิดและไม่เป็นเชิงเส้น

จากรูปที่ 2.1 ระบบมีค่าต่าง ๆ ซึ่งมีนิยามดังตาราง 2.1 ต่อจากนี้เราจะใช้สมการลากรองจ์ หาแบบจำลองไม่เชิงเส้นของระบบ โดยเริ่มจากการพิจารณาพลังงานศักย์และพลังงานจลน์ของระบบดังนี้

พลังงานจลน์ของระบบ

พลังงานจลน์ของระบบแบ่งเป็นสามส่วน คือ

1. พลังงานจลน์ของเพนดูลัมในการหมุนรอบจุดศูนย์กลางของจานหมุน
2. พลังงานจลน์ของเพนดูลัมในการหมุนรอบจุดศูนย์กลางมวลของแท่งเพนดูลัมเอง
3. พลังงานจลน์จากการเคลื่อนที่ของจุดศูนย์กลางมวลของแท่งเพนดูลัม

จากการวิเคราะห์ข้างต้นจะได้ว่า

$$K_e = \frac{1}{2}J_0\dot{\alpha}^2 + \frac{1}{2}J_1\dot{\beta}^2 + \frac{1}{2}mv^2 \tag{2.1}$$

โดย v เป็นความเร็วเชิงเส้นของก้านเพนดูลัม เมื่อพิจารณาหาความเร็วของจุดศูนย์กลางมวลชองแท่งเพนดูลัม จากรูปที่ 2.1 ให้จุดศูนย์กลางมวลของแท่งเพนดูลัมอยู่ที่ $l\hat{a}_{\acute{r}}$ โดยที่

$$l\hat{a}_{\acute{r}} = \hat{a}_z\cos\beta - \hat{a}_\alpha\sin\beta \tag{2.2}$$

พิจารณาจุด C.M. เทียบกับจุด O จะได้ว่าตำแหน่งของจุด C.M. มีค่าดังนี้

$$P_{\text{C.M.}} = \hat{a}_z\cos\beta - \hat{a}_\alpha\sin\beta + L\hat{a}_r \tag{2.3}$$

โดยที่

$$\hat{a}_\alpha = -\hat{a}_x\sin\alpha + \hat{a}_y\cos\alpha \tag{2.4}$$

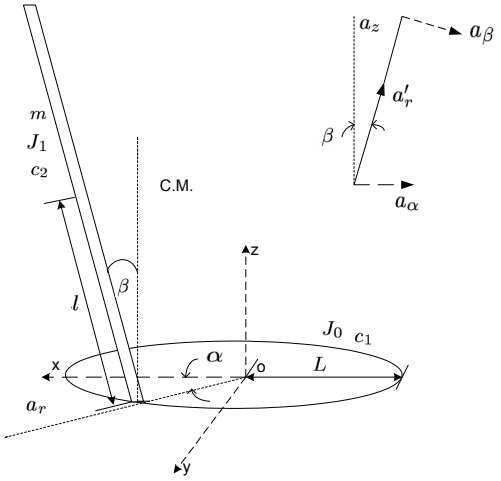
$$\hat{a}_r = \hat{a}_z\cos\alpha + \hat{a}_y\sin\alpha \tag{2.5}$$

แทนค่า $\hat{a}_\alpha$ และ $\hat{a}_r$ ลงใน (2.1) จะได้พิกัดของจุดศูนย์กลางมวลของเพนดูลัม ในพิกัดคาร์ทีเชียน คือ

$$P_{\text{C.M.}} = (l\sin\beta\sin\alpha + L\cos\alpha, L\sin\alpha - l\sin\beta\cos\alpha, l\cos\beta) \tag{2.6}$$

จากตำแหน่งของจุดศูนย์กลางมวลของเพนดูลัมดังสมการ (2.6) เราสามารถหาความเร็วได้ดังต่อไปนี้

$$v_{\text{x}} = C\dot{M}_{\text{x}} = l\sin\beta\cos\alpha\dot{\alpha} + \left(l\cos\beta\dot{\beta} - L\dot{\alpha}\right)\sin\alpha \tag{2.7}$$



รูปที่ 2.1: แท่งเพนดูลัมและจานหมุน

สัญลักษณ์	ชื่อของค่าคงตัว	หน่วย
α	มุมของจานหมุนที่เปลี่ยนไป	rad
β	มุมที่เปลี่ยนไปของเพนดูลัมทำกับแกนในแนวดิ่ง	rad
J_0	โมเมนต์ความเฉื่อยรอบจุดศูนย์กลางมวลของจานหมุน	$\mathrm{kg \cdot m^2}$
J_1	โมเมนต์ความเฉื่อยรอบจุดศูนย์กลางมวลของเพนดูลัม	$\mathrm{kg \cdot m^2}$
c_1	แรงเสียดทานของจานหมุน	$\mathrm{N - m \cdot s}$
c_2	แรงเสียดทานของก้านเพนดูลัม	$\mathrm{N - m \cdot s}$
m	มวลของเพนดูลัม	kg
l	ระยะห่างจากจานหมุนถึงจุดศูนย์กลางมวลของเพนดูลัม	m
L	รัศมีของจานหมุน	m
g	ค่านิจแรงโน้มถ่วงของโลก	$\mathrm{m/s^2}$

ตารางที่ 2.1: ตัวแปรของระบบเพนดูลัมผกผันแบบหมุน

$$v_{\mathrm{y}} = C\dot{M}_{\mathrm{y}} = l\sin\beta\sin\alpha\dot{\alpha} - \Big(l\cos\beta\dot{\beta} - L\dot{\alpha}\Big)\cos\alpha \tag{2.8}$$

$$v_{\mathrm{z}} = C\dot{M}_{\mathrm{z}} = -l\dot{\beta}\sin\beta \tag{2.9}$$

$$v_{\mathrm{x}}^2 + v_{\mathrm{y}}^2 + v_{\mathrm{z}}^2 = l^2\sin^2\beta\dot{\alpha}^2 - 2lL\cos\beta\dot{\beta}\dot{\alpha} + L^2\dot{\alpha}^2 + l^2\dot{\beta}^2 \tag{2.10}$$

ดังนั้น

$$\frac{1}{2}mv^2 = \frac{1}{2}ml^2\sin^2\beta\dot{\alpha}^2 + \frac{1}{2}mL^2\dot{\alpha}^2 + \frac{1}{2}ml^2\dot{\beta}^2 - mlL\cos\beta\dot{\beta}\dot{\alpha} \tag{2.11}$$

พลังงานศักย์ของระบบ

พลังงานศักย์ของระบบ

$$K_{\mathrm{p}} = U = mgl\cos\beta \tag{2.12}$$

จากพลังงานจลน์และพลังงานศักย์ที่ได้ ดังสมการ(2.1) และ (2.12) เราสามารถเขียนสมการลากรองจ์ ได้ดังนี้

$$L = K_{\mathrm{e}} - U \tag{2.13}$$

$$L = \frac{1}{2}J_0\dot{\alpha}^2 + \frac{1}{2}J_1\dot{\beta}^2 + \frac{1}{2}ml^2\sin^2\beta\dot{\alpha}^2 + \frac{1}{2}mL^2\dot{\alpha}^2 + \frac{1}{2}ml^2\dot{\beta}^2 - mlL\cos\beta\dot{\beta}\dot{\alpha} - mgl\cos\beta \tag{2.14}$$

จาก

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}_i} - \frac{\partial L}{\partial\theta_i} + \frac{\partial R}{\partial\dot{\theta}_i} = Q \tag{2.15}$$

โดย R คือพลังงานที่สูญเสียไปเนื่องจากสัมประสิทธิ์แรงเสียดทานของระบบ

$$R = \frac{1}{2}c_1\dot{\alpha}^2 + \frac{1}{2}c_2\dot{\beta}^2 \tag{2.16}$$

พิจารณา $\theta_i = \alpha$ คำนวนหา $\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}_i}$ และ $\frac{\partial L}{\partial\theta_i}$ โดยที่ แรงบิดมีผลให้เกิดงานเมื่อ α เปลี่ยนไป แทนลงในสมการลากรองจ์ คือ สมการที่ (2.15) จะได้สมการ (2.17) จากนั้นพิจารณา $\theta_i = \beta$ คำนวนหา $\frac{d}{dt}\frac{\partial L}{\partial\beta}$ และ $\frac{\partial L}{\partial\beta}$ แทนลงในสมการลากรองจ์ (2.18) แบบจำลองของระบบเป็นดังนี้

$$p_{11}\ddot{\alpha} + p_{12}\ddot{\beta} + q_1 = \tau \tag{2.17}$$

$$p_{12}\ddot{\alpha} + p_{22}\ddot{\beta} + q_2 = 0 \tag{2.18}$$

โดยที่

$$p_{11} = J_0 + mL^2 + \frac{1}{2}ml^2\left(1 - \cos(2\beta)\right) \tag{2.19}$$

$$p_{12} = -mLl\cos\beta \tag{2.20}$$

$$p_{22} = J_1 + ml^2 \tag{2.21}$$

$$q_1 = c_1\dot{\alpha} + ml^2\sin(2\beta)\dot{\alpha}\dot{\beta} + mlL\dot{\beta}^2\sin\beta \tag{2.22}$$

$$q_2 = c_2\dot{\beta} - \frac{1}{2}ml^2\sin(2\beta)\dot{\alpha}^2 - mgl\sin\beta \tag{2.23}$$

2.2 ของระบบแบบสัญญาขาเข้าเป็นแรงบิดและเป็นแบบเชิงเส้น

จากสมการ (2.17) และ (2.17) จะทำการแปลงแบบจำลองไม่เชิงเส้นให้เป็นแบบเชิงเส้นเพื่อความสะดวกในการออกแบบตัวควบคุมต่อไป โดยในที่นี้จะแปลงให้เป็นเชิงเส้นรอบจุดที่แท่งเพนดูลัมนั้นตั้งได้ นั้นคือจุดที่ $\beta =$

0 โดยการประมาณให้ $\dot{\alpha}^2 \approx 0, \cos\beta \approx 1, \sin\beta \approx \beta$ จะได้สมการ

$$(J_0 + mL^2)\ddot{\alpha} - mlL\ddot{\beta} + c_1\dot{\alpha} = \tau \tag{2.24}$$

$$-mlL\ddot{\alpha} + (J_1 + ml^2)\ddot{\beta} + c_2\dot{\beta} - mlg\beta = 0 \tag{2.25}$$

กำหนดตัวแปรสถานะ $x = \begin{bmatrix}\alpha & \beta & \dot{\alpha} & \dot{\beta}\end{bmatrix}^{\mathrm{T}}$ จากสมการ (2.24) และ (2.25) เขียนให้อยู่ในรูปสมการสถานะ จะได้ว่า

$$E\dot{x} = Fx + Gu \tag{2.26}$$

โดยที่

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & J_0 + mL^2 & -mlL \\ 0 & 0 & -mlL & J_1 + ml^2 \end{bmatrix} \tag{2.27}$$

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -c_1 & 0 \\ 0 & mgl & 0 & -c_2 \end{bmatrix} \tag{2.28}$$

$$G = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.29}$$

จะได้สมการสถานะของระบบ คือ

$$\dot{x} = E^{-1}Fx + E^{-1}Gu \tag{2.30}$$

โดยเมตริกซ์ E จะต้องมีดีเทอร์มิแนนท์ไม่เท่ากับ 0 เพื่อให้สามารถหาเมตริกซ์ผกผันของเมตริกซ์ E ได้ โดยที่

$$A = E^{-1}F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m^2l^2L}{g(J_0J_1+J_0ml^2+mL^2J_1)} & \frac{-(J_1+ml^2)}{c_1(J_0J_1+J_0ml^2+mL^2J_1)} & \frac{mlL}{c_2(J_0J_1+J_0ml^2+mL^2J_1)} \\ 0 & \frac{J_0+mL^2}{mgl(J_0J_1+J_0ml^2+mL^2J_1)} & \frac{mlL}{c_1(J_0J_1+J_0ml^2+mL^2J_1)} & \frac{-(J_0+mL^2)}{c_2(J_0J_1+J_0ml^2+mL^2J_1)} \end{bmatrix} \tag{2.31}$$

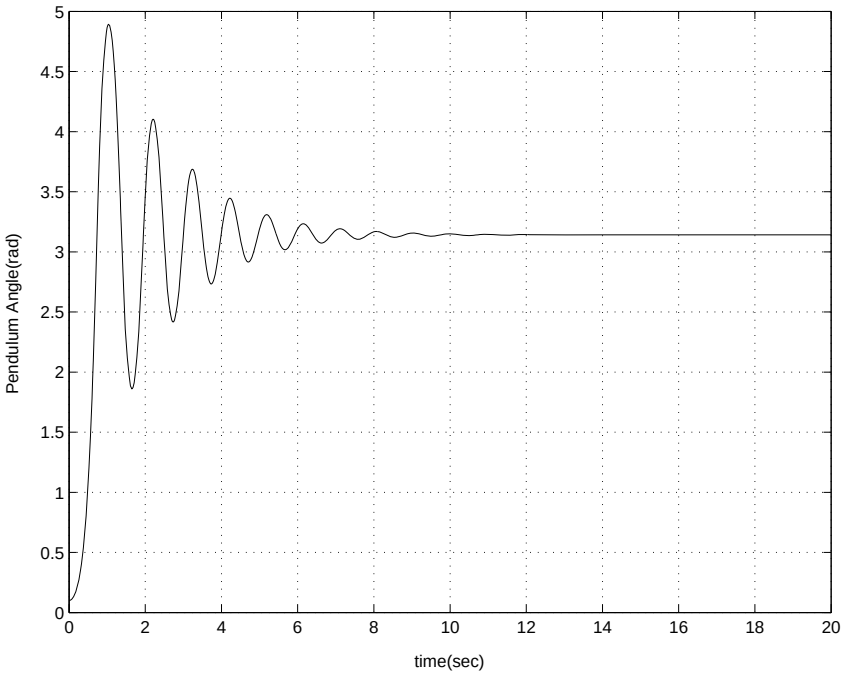
$$B = E^{-1}G = \begin{bmatrix} 0 \\ 0 \\ \frac{(J_1+ml^2)}{(J_0J_1+J_0ml^2+mL^2J_1)} \\ \frac{mlL}{(J_0J_1+J_0ml^2+mL^2J_1)} \end{bmatrix} \tag{2.32}$$

เมื่อทำการจำลองระบบแบบไม่เชิงเส้นในโปรแกรม MATLAB จะได้ผลตอบสนอง ดังรูปที่ 2.2(ก) ซึ่ง จะเห็นว่าผลการจำลองระบบสอดคล้องกับความเป็นจริง กล่าวคือ เมื่อกระตุ้นระบบด้วยแรงบิด ตัวเพนดูลัมจะแกว่งไปมาจนเข้าสู่สภาวะอยู่ตัวที่ค่า π rad ซึ่งคือตำแหน่งแนวดิ่งนั้นเอง และเมื่อทำการจำลองแบบจำลองแบบเชิงเส้นจะได้ผลตามรูปที่ 2.2(ข) ซึ่งจะเห็นได้ว่าเมื่อเวลาผ่านไป มุมของแท่งเพนดูลัมจะมีค่าเพิ่มเข้าสู่อนันต์ นั้นแสดงว่าการควบคุมระบบที่ประมาณให้เป็นเชิงเส้นนั้นจะทำได้แค่เพียงมุมเล็กๆค่าหนึ่งเท่านั้น

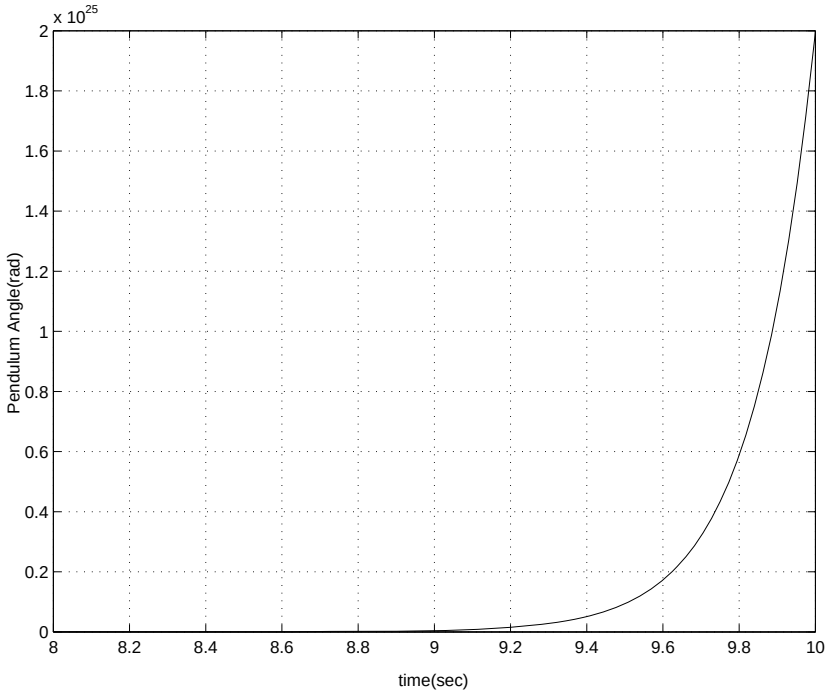
2.3 เพนดูลัมผกผันคู่แบบหมุน

ลักษณะและพลวัตของระบบ

เพนดูลัมผกผัน (inverted pendulum) เป็นอุปกรณ์ที่ใช้อย่างแพร่หลายในการสาธิตการประยุกต์ใช้ทฤษฎีควบคุม จุดประสงค์ในการ ออกแบบตัวควบคุมคือรักษาสมดุลของแท่งเพนดูลัมให้ตั้งในแนวดิ่ง โดยอาศัย

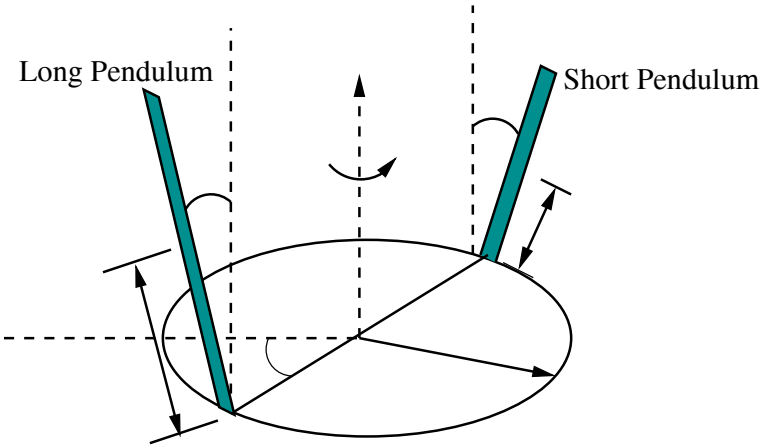


(ก) กราฟแสดงผลตอบสนองทางเวลาสำหรับระบบไม่เชิงเส้น



(ข) กราฟแสดงผลตอบสนองทางเวลาสำหรับระบบเชิงเส้น

รูปที่ 2.2: กราฟแสดงผลตอบสนองทางเวลาสำหรับระบบ (ก) ไม่เชิงเส้น; (ข) เชิงเส้น



รูปที่ 2.3: ระบบเพนดูลัมผกผันคู่แบบหมุน

การสร้างแรงที่บริเวณฐานของแท่งเพนดูลัม เพนดูลัมผกผันที่ใช้ในช่วงแรก คือเพนดูลัมผกผันบนรถ (cart pendulum) ซึ่งฐานของเพนดูลัมจะติดอยู่บนรถและอาศัยการเคลื่อนที่ของรถในการรักษาสมดุลของเพนดูลัม อย่างไรก็ตาม เพนดูลัมรูปแบบนี้มีข้อจำกัดในทางปฏิบัติเกี่ยวกับระยะทางที่รถต้องใช้ในการเคลื่อนที่ ด้วยเหตุผลดังกล่าวจึงมีการพัฒนารูปแบบเป็นเพนดูลัมผกผันแบบ หมุน (rotary inverted pendulum) ขึ้น สำหรับเพนดูลัมผกผันแบบหมุนฐานของแท่งเพนดูลัมจะติดบนจานหมุนและอาศัยการหมุนของจานหมุนในการรักษา สมดุลของแท่งเพนดูลัม ดังนั้นเพนดูลัมรูปแบบนี้จึงไม่มีข้อจำกัดเกี่ยวกับระยะการเคลื่อนที่

ในงานวิจัยนี้เราพิจารณาเพนดูลัมผกผันคู่แบบหมุน (rotary double inverted pendulum) ซึ่งประกอบด้วยแท่งเพนดูลัมสองแท่งที่มีความยาวแตกต่างกัน และตั้งอยู่ในตำแหน่งตรงกันข้ามบนจานหมุนดังแสดงในรูปที่ 2.3 เพนดูลัมผกผันรูปแบบนี้จะมีความยากในการควบคุม มากกว่าเพนดูลัมผกผันแบบเดี่ยว เนื่องจากจานหมุนต้องหมุนเพื่อรักษาสมดุลของแท่งเพนดูลัมสองแท่งพร้อมๆกัน ขณะที่เพนดูลัมผกผันแบบเดี่ยวจะรักษา สมดุลของแท่งเพนดูลัมเพียงแท่งเดียวเท่านั้น

จากรูปที่ 2.3 กำหนดให้ α คือการกระจัดเชิงมุมของจานหมุน (หน่วยเป็น rad), β_1 และ β_2 คือมุมของเพนดูลัมแท่งยาวและแท่งสั้นเทียบกับแกนแนวดิ่ง (rad) และ τ คือ แรงบิด (torque) ที่กระทำต่อจานหมุน (N·m) ในการควบคุมเราใช้มอเตอร์กระแสตรง เป็นตัวขับเร้า (actuator) สำหรับสร้างแรงบิดให้กับจานหมุน จุดประสงค์ของเราคือการควบคุมการเคลื่อนที่จานหมุนเพื่อรักษาสมดุลของ แท่งเพนดูลัมทั้งสองแท่ง เมื่อมุมเบี่ยงเบนเริ่มต้นมีค่าไม่เท่ากับศูนย์ ส่วนค่าพารามิเตอร์ต่างๆของระบบแสดงในตารางที่ 2.2

สมการการเคลื่อนที่ (equation of motion) ของระบบสามารถอธิบายได้ดังนี้

$$\begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix} \begin{pmatrix} \ddot{\alpha} \\ \ddot{\beta}_1 \\ \ddot{\beta}_2 \end{pmatrix} + \begin{pmatrix} p'_1 \\ p'_2 \\ p'_3 \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \\ 0 \end{pmatrix} \tag{2.33}$$

ตารางที่ 2.2: ค่าพารามิเตอร์ของเพนดูลัมผกผันคู่แบบหมุน

พารามิเตอร์	สัญลักษณ์	ค่าที่ใช้	หน่วย
โมเมนต์ความเฉื่อยของจานหมุน	J_0	0.06	kg-m^2
โมเมนต์ความเฉื่อยของเพนดูลัมแท่งยาว	J_1	0.1	kg-m^2
โมเมนต์ความเฉื่อยของเพนดูลัมแท่งสั้น	J_2	1.915×10^{-3}	kg-m^2
สัมประสิทธิ์ความเสียดทานของจานหมุน	c_0	0.004	N-m·s
สัมประสิทธิ์ความเสียดทานของเพนดูลัมแท่งยาว	c_1	0.003	N-m·s
สัมประสิทธิ์ความเสียดทานของเพนดูลัมแท่งสั้น	c_2	0.0009	N-m·s
มวลของเพนดูลัมแท่งยาว	m_1	1	kg
มวลของเพนดูลัมแท่งสั้น	m_2	0.13	kg
ระยะจากจุดหมุนถึงจุดศูนย์กลางมวลของเพนดูลัมแท่งยาว	l_1	0.3	m
ระยะจากจุดหมุนถึงจุดศูนย์กลางมวลของเพนดูลัมแท่งสั้น	l_2	0.19	m
รัศมีจานหมุน	L	0.5	m
ค่าคงที่แรงโน้มถ่วง	g	9.781	m/s

โดยที่

$$\begin{aligned}
P_{11} &= J_0 + m_1 l_1^2 \sin^2\beta_1 + m_1 L^2 + m_2 l_2^2 \sin^2\beta_2 + m_2 L^2 \\
P_{12} &= -m_1 l_1 L \cos\beta_1, \qquad P_{13} = -m_2 l_2 L \cos\beta_2 \\
P_{21} &= -m_1 l_1 L \cos\beta_1, \qquad P_{22} = J_1 + m_1 l_1^2 \\
P_{23} &= 0, \qquad P_{31} = -m_2 l_2 L \cos\beta_2 \\
P_{32} &= 0, \qquad P_{33} = J_2 + m_2 l_2^2
\end{aligned} \tag{2.34}$$

$$\begin{aligned}
p_1' &= m_1 l_1^2 \dot\beta_1 \dot\alpha \sin(2\beta_1) + m_2 l_2^2 \dot\beta_2 \dot\alpha \sin(2\beta_2) + m_1 l_1 L \dot\beta_1^2 \sin\beta_1 + m_2 l_2 L \dot\beta_2^2 \sin\beta_2 + c_0 \dot\alpha \\
p_2' &= -m_1 l_1^2 \dot\alpha^2 \sin\beta_1 \cos\beta_1 - m_1 g l_1 \sin\beta_1 + c_1 \dot\beta_1 \\
p_3' &= -m_2 l_2^2 \dot\alpha^2 \sin\beta_2 \cos\beta_2 - m_2 g l_2 \sin\beta_2 + c_2 \dot\beta_2
\end{aligned} \tag{2.35}$$

แบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์

ในการสร้างแบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ เราจะอาศัยแนวคิดของระบบกึ่งเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ (quasi linear parameter varying, quasi-LPV) อย่างไรก็ตามเทอมที่ไม่เป็นเชิงเส้นบางส่วนใน (2.34) และ (2.35) จะถูกประมาณเป็นเชิงเส้นรอบๆจุด $\beta_1 = \beta_2 = 0$ เพื่อทำการลดจำนวนเทอมไม่เป็นเชิงเส้น ที่ต้องนิยามเป็นพารามิเตอร์แปรตามเวลาไม่ให้มีจำนวนมากเกินไป ในที่นี้เรากำหนดให้

$$\begin{aligned}
\delta_1 &= m_1 l_1 L \dot\beta_1 \sin\beta_1, \qquad \delta_2 = m_2 l_2 L \dot\beta_2 \sin\beta_2 \\
\delta_3 &= \frac{1}{2} m_1 l_1^2 \dot\alpha \sin(2\beta_1), \qquad \delta_4 = \frac{1}{2} m_2 l_2^2 \dot\alpha \sin(2\beta_2)
\end{aligned} \tag{2.36}$$

เมื่อแทนกลับเข้าไปใน (2.35) เราได้ว่า

$$\begin{aligned}
p_1' &= 2\delta_3 \dot\beta_1 + 2\delta_4 \dot\beta_2 + \delta_1 \dot\beta_1 + \delta_2 \dot\beta_2 + c_0 \dot\alpha \\
p_2' &= -\delta_3 \dot\alpha - m_1 g l_1 \sin\beta_1 + c_1 \dot\beta_1 \\
p_3' &= -\delta_4 \dot\alpha - m_2 g l_2 \sin\beta_2 + c_2 \dot\beta_2
\end{aligned}$$

ทำการประมาณเทอม $\sin\beta_1$ และ $\sin\beta_2$ รอบจุด $\beta_1 = \beta_2 = 0$ เราได้ว่า

$$p_2' \approx -\delta_3\dot{\alpha} - m_1 g l_1 \beta_1 + c_1\dot{\beta}_1$$
$$p_3' \approx -\delta_4\dot{\alpha} - m_2 g l_2 \beta_2 + c_2\dot{\beta}_2$$

พิจารณาเทอม P_{11}, P_{12} และ P_{13} ใน (2.34) เมื่อทำการประมาณเป็นเชิงเส้นรอบ $\beta_1 = \beta_2 = 0$ เราได้ว่า

$$\begin{aligned} P_{11} &\approx J_0 + m_1 L^2 + m_2 L^2 \\ P_{12} &\approx -m_1 l_1 L, \qquad P_{13} \approx -m_2 l_2 L \end{aligned} \tag{2.37}$$

เมื่อกำหนดให้ตัวแปรสถานะของระบบ $x^T = \begin{pmatrix} \alpha & \beta_1 & \beta_2 & \dot{\alpha} & \dot{\beta}_1 & \dot{\beta}_2 \end{pmatrix}$ เราได้แบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ $\mathcal{M}_1$ คือ

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&J_0+m_1L^2+m_2L^2&-m_1l_1L&-m_2l_2L \\ 0&0&0&-m_1l_1L&J_1+m_1l_1^2&0 \\ 0&0&0&-m_2l_2L&0&J_2+m_2l_2^2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} =$$
$$\begin{pmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&-c_0&-(\delta_1+2\delta_3)&-(\delta_2+2\delta_4) \\ 0&m_1gl_1&0&\delta_3&-c_1&0 \\ 0&0&m_2gl_2&\delta_4&0&-c_2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}\tau \tag{2.38}$$

อย่างไรก็ตามเราสามารถสร้างแบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ในรูปแบบอื่นได้อีก โดยนิยามพารามิเตอร์แปรตามเวลาในรูปแบบที่แตกต่างกัน ในกรณีที่เรากำหนดให้พารามิเตอร์แปรตามเวลา δ_5 มีค่าดังนี้

$$\delta_5 = m_1 l_1^2 \dot{\beta}_1 \sin(2\beta_1) + m_2 l_2^2 \dot{\beta}_2 \sin(2\beta_2) \tag{2.39}$$

เราได้ว่าเทอม p_1' สามารถแสดงได้ในอีกรูปแบบหนึ่งคือ

$$p_1' = \delta_5\dot{\alpha} + \delta_1\dot{\beta}_1 + \delta_2\dot{\beta}_2 + c_0\dot{\alpha} \tag{2.40}$$

จากการนิยามพารามิเตอร์ในรูปแบบดังกล่าว เราได้แบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ $\mathcal{M}_2$ คือ

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} 1&0&0 & 0 & 0 & 0 \\ 0&1&0 & 0 & 0 & 0 \\ 0&0&1 & 0 & 0 & 0 \\ 0&0&0 & J_0+m_1L^2+m_2L^2 & -m_1l_1L & -m_2l_2L \\ 0&0&0 & -m_1l_1L & J_1+m_1l_1^2 & 0 \\ 0&0&0 & -m_2l_2L & 0 & J_2+m_2l_2^2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} = \begin{pmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&-(c_0+\delta_5)&-\delta_1&-\delta_2 \\ 0&m_1gl_1&0&\delta_3&-c_1&0 \\ 0&0&m_2gl_2&\delta_4&0&-c_2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}\tau \quad (2.41)$$

ถ้าเราแยกเทอมที่บวกกันในพารามิเตอร์ δ_5 ออกเป็น $\delta_5 = \delta_6 + \delta_7$ โดยที่

$$\delta_6 = m_1l_1^2\dot{\beta}_1\sin(2\beta_1), \qquad \delta_7 = m_2l_2^2\dot{\beta}_2\sin(2\beta_2) \quad (2.42)$$

เราได้แบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ $\mathcal{M}_3$ คือ

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} 1&0&0 & 0 & 0 & 0 \\ 0&1&0 & 0 & 0 & 0 \\ 0&0&1 & 0 & 0 & 0 \\ 0&0&0 & J_0+m_1L^2+m_2L^2 & -m_1l_1L & -m_2l_2L \\ 0&0&0 & -m_1l_1L & J_1+m_1l_1^2 & 0 \\ 0&0&0 & -m_2l_2L & 0 & J_2+m_2l_2^2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} = \begin{pmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&-(c_0+\delta_6+\delta_7)&-\delta_1&-\delta_2 \\ 0&m_1gl_1&0&\delta_3&-c_1&0 \\ 0&0&m_2gl_2&\delta_4&0&-c_2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dot{\alpha} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}\tau \quad (2.43)$$

จากแนวทางที่ได้แสดงมาเห็นได้ว่าแบบจำลองเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ สามารถมีได้มากกว่าหนึ่งรูปแบบขึ้นอยู่กับการนิยามพารามิเตอร์แปรตามเวลาของระบบ เนื่องจากพารามิเตอร์แปรตามเวลาจะทำหน้าที่ในการปรับพลวัตของตัวควบคุมเชิงเส้นที่เปลี่ยนตามพารามิเตอร์ การนิยามพารามิเตอร์แปรตามเวลาที่แตกต่างกันจึงมักจะให้ผลการควบคุมที่แตกต่างกันด้วย ในอนาคตเราจะแสดงผลของการเลือกใช้แบบจำลองในการออกแบบตัวควบคุมที่มีต่อลักษณะผลตอบของระบบ

บทที่ 3
การออกแบบระบบควบคุม

ระบบเพนดูลัมผกผันแบบหมุนเป็นระบบที่มีความไม่เป็นเชิงเส้นมากและมีตัวแปรที่เราต้องการควบคุมและสัญญาณขาเข้าหลายตัว การควบคุมระบบด้วยวิธี PID จึงให้ผลตอบสนองที่ไม่ค่อยดีนัก ในที่นี้จึงได้นำเสนอวิธีการควบุมที่เรียกว่า Optimal Control โดยจากแบบจำลองของระบบที่ได้จากที่แสดงไปแล้วข้างต้น และพารามิเตอร์ของระบบที่ได้จากการหาเอกลักษณ์ของระบบ พารามิเตอร์ที่ได้จากการหาเอกลักษณ์ของระบบมีค่าดังตาราง 2.1 ในที่นี้แสดงการออกแบบระบบควบคุมของสองแบบจำลอง ได้แก่ แบบจำลองที่มีแรงบิดเป็นสัญญาณขาเข้าก่อน และแบบจำลองที่มีความเร็วเชิงเส้นของจานหมุนเป็นสัญญาณขาเข้า ดังต่อไปนี้

3.1 แบบจำลองที่มีแรงบิดเป็นสัญญาณขาเข้า

Optimal Control

ระบบเพนดูลัมแบบผกผันมีแบบจำลองคณิตศษสตร์ที่แปลงให้อยู่ในรูปเชิงเส้นแล้วดังนี้

$$\dot{x} = Ax + Bu \tag{3.1}$$

$$y = Cx + Du \tag{3.2}$$

โดยที่

$$x = \begin{bmatrix} \alpha, & \beta, & \dot{\alpha}, & \dot{\beta}, \end{bmatrix}^{\mathrm{T}} \tag{3.3}$$

$$u = \tau \tag{3.4}$$

และ

$$A = \begin{bmatrix} 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \\ 0.0000 & 0.0000 & -3.5019 & 14.4011 \\ -0.0222 & 0.0061 & 0.0248 & -0.1020 \end{bmatrix} \tag{3.5}$$

$$B = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 55.4589 \\ -15.2122 \end{bmatrix} \tag{3.6}$$

$$C = I_{4\times 4} \tag{3.7}$$

$$D = \begin{bmatrix} 0.0000 \end{bmatrix} \tag{3.8}$$

ตำแหน่งขั้วของระบบ$(P) = \begin{bmatrix} 0 \\ -3.8470 \\ 3.7435 \\ -0.0207 \end{bmatrix}$ (3.9)

จากระบบเป็นระบบอันดับสี่ เราสามารถหาเมตริกค่าคงที่ในการป้อนกลับ (K) ได้หลายวิธี ในที่นี้จะขอเสนอวิธีการ LQR (linear quadratic regulator) โดยกำหนดเมตริก Q และ R แล้วใช้คำสั่ง LQR ในโปรแกรม MATLAB เพื่อหาเมตริก ค่าคงตัวในการป้อนกลับ (K) ที่ดีที่สุดที่สอดคล้องกับเมตริก Q และนำผลที่ได้ไปจำลองโดยใช้คำสั่งใน M-File ดังนี้

Matlab Code

```
clear;
clc;
%PARAMETER
syms J1 J0 m g l L J1 c1 c2 real;
m=0.058;
g=9.8;
l=0.405;
L=0.2;
J1=0.007614;
J0=0.017;
c1=0.0004;
c2=0.001631;
%LQR
E =[1 0 0 0;0 1 0 0;0 0 J0+m*(L2) m*l*L;0 0 m*l*L J1+m*(l2)];
F =[0 0 1 0;0 0 0 1;0 0 -c1 0;0 m*g*l 0 -c2];
G =[0;0;1;0];
A = inv(E)*F;
B = inv(E)*G;
C = [1 0 0 0;
0 1 0 0;
0 0 1 0;
0 0 0 1];
D = [0];
x=100000000;
y=100000;
Q=[x 0 0 0;
0 y 0 0;
0 0 0 0;
0 0 0 0];
R=10;
K = lqr(A,B,Q,R);
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
%PLOT
T=0:0.01:100;
```

```
U=1*ones(size(T));
[Y, X] = lsim(Ac,Bc,Cc,Dc,U,T);

subplot(2,2,1); plot(T,Y(:,1))
title('Plate Angle')
axis([0 5 -1e-3 1e-3])

subplot(2,2,2); plot(T,Y(:,2))
title('Pendulum Angle')
axis([0 5 -1e-3 1e-3])

subplot(2,2,3); plot(T,Y(:,3))
title('Plate Angular Velocity')
axis([0 5 -1e-3 1e-3])

subplot(2,2,4); plot(T,Y(:,4))
title('Pendulum Angular Velocity')
axis([0 5 -1e-3 1e-3])
```

จากคำสั่งใน M-File เราได้ปรับค่า Q และ R ให้ได้ผลตอบตอบสนองดีที่สุด โดยเราเลือกให้

$$Q = \begin{bmatrix} 10^8 & 0 & 0 & 0 \\ 0 & 10^5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad R = 10 \tag{3.10}$$

จะได้ $$K = [-3162 \ -23956 \ -1758 \ 6450] \tag{3.11}$$

ผลตอบสนองหลังจากควบคุม

หลังจากควบคุมด้วยวิธี LQR แล้วจะได้ระบบใหม่ที่มีค่าคงตัวเป็นดังนี้

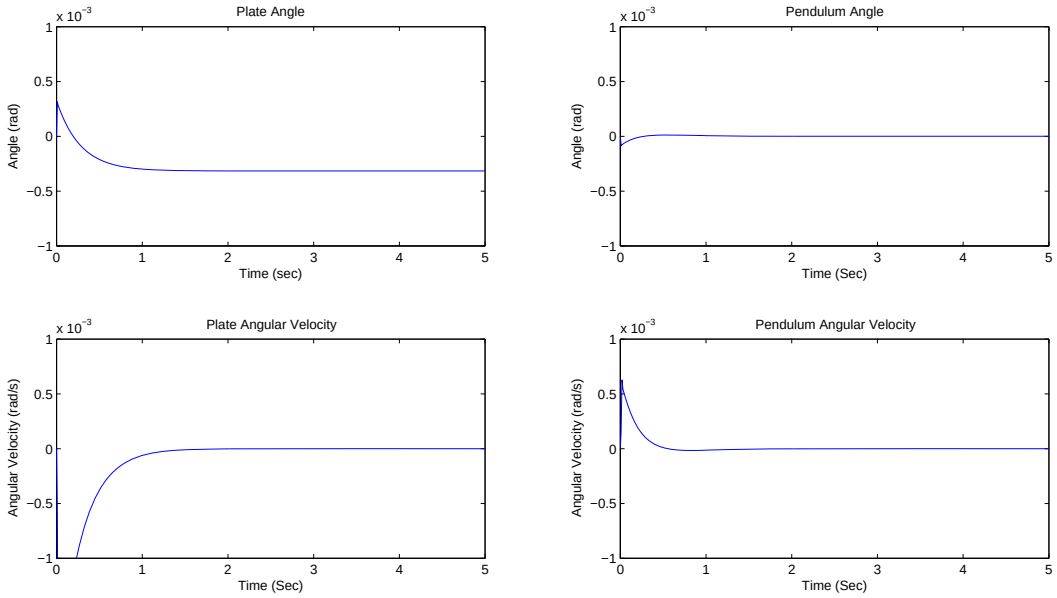
$$A = 10^6 \times \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.1754 & -0.0481 & 1.3285 & -0.3644 \\ -0.0975 & -0.0267 & 0.3577 & -0.0981 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 55.4589 \\ -15.2122 \end{bmatrix} \tag{3.12}$$

$$C = I_{4\times 4}, \qquad D = [0] \tag{3.13}$$

ซึ่งมีขั้วของระบบเป็น

$$P = [-2.9613 + 2.9613i, \ -2.9613 - 2.9613i, \ -0.0371, \ -0.0362]^{\mathrm{T}} \tag{3.14}$$

และได้ผลตอบสนองต่อฟังก์ชั่น ขั้นบันไดขนาด 1 rad เริ่มที่เวลา 0 s ดังนี้



รูปที่ 3.1: แสดงผลตอบเชิงเวลาของระบบที่มีสัญญาขาเข้าเป็นแรงบิดภายหลังการควบคุมแบบ LQR ต่อฟังก์ชั่นขั้นบันไดขนาด1 rad เริ่มที่เวลา 0 s

3.2 แบบจำลองที่มีความเร็วเชิงมุมของจานหมุนเป็นสัญญาณขาเข้า

มีพารามิเตอร์ของระบบที่ได้จากการหาเอกลักษณ์ของระบบดังนี้

$$A = \begin{bmatrix} 0 & 1 \\ 13.4405 & -0.0952 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0.2743 \end{bmatrix} \tag{3.15}$$

$$C = I_{4\times 4} \qquad D = [0] \tag{3.16}$$

จากระบบเป็นระบบอันดับสองเราสามารถหาเมตริกค่าคงที่ในการป้อนกลับ K ได้โดยวิธีการ LQR ดังที่ได้นำเสนอไปก่อนหน้านี้แล้ว โดยใช้คำสั่งใน M-File ดังนี้

Matlab Code

```
clear;
clc;
%PARAMETER
syms J1 J0 m g l L J1 c1 c2 real;
m=0.058;
g=9.8;
l=0.405;
L=0.2;
J1=0.007614;
J0=0.017;
c1=0.0004;
c2=0.001631;
%LQR
```

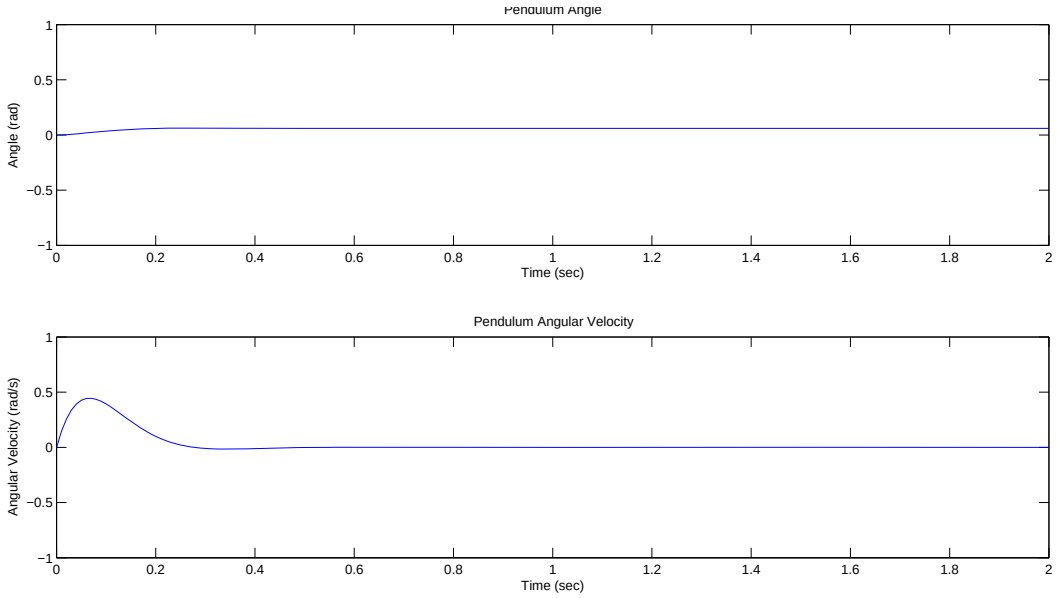
```
A = [0 1; (m*g*l)/(J1+m*(l2)) -c2/(J1+m*(l2))];
B = [0; (m*l*L)/(J1+m*(l2))];
C = [1 0;
0 1];
D = [0];

x=10000000;
y=1000; Q=[x 0;
0 y];
R=10;
K = lqr(A,B,Q,R);
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
%PLOT
T=0:0.01:100;
U=1*ones(size(T));
[Y, X]=lsim(Ac,Bc,Cc,Dc,U,T);

subplot(2,1,1); plot(T,Y(:,1))
title('Pendulum Angle')
axis([0 2 -1 1])

subplot(2,1,2); plot(T,Y(:,2))
title('Pendulum Angular Velocity')
axis([0 2 -1 1])
```

จากคำสั่งใน M-File เราได้ปรับค่า Q และ R ให้ได้ผลตอบตอบสนองดีที่สุด จากการเลือกค่า Q และ R ได้ค่า K ดังนี้

$$Q = \begin{bmatrix} 10^7 & 0 \\ 0 & 10^3 \end{bmatrix}, \qquad R = 10 \tag{3.17}$$

$$K = [1050.2,\ 87.7] \tag{3.18}$$

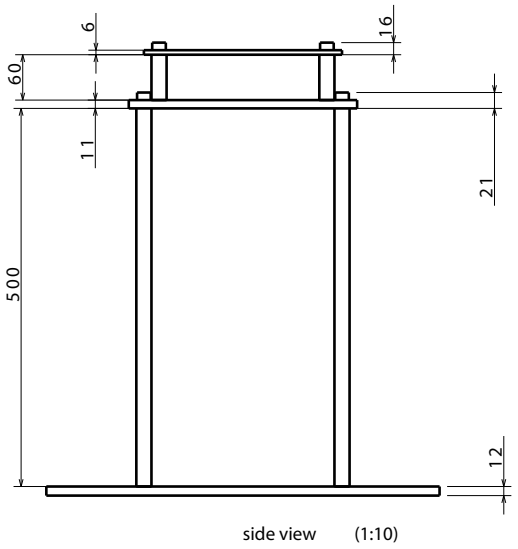
ผลตอบสนองจากการควบคุม



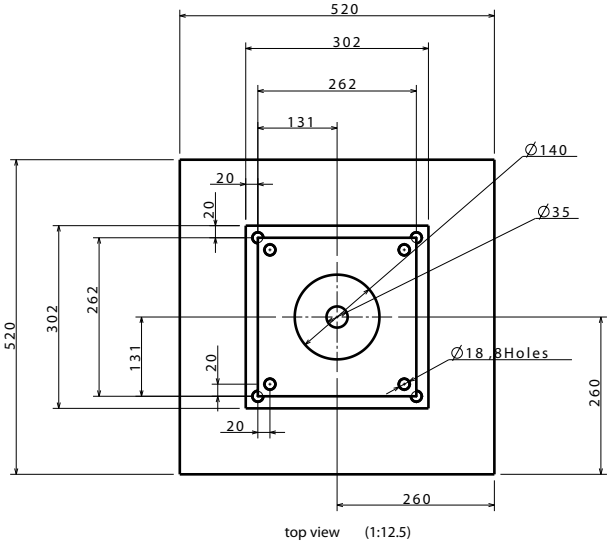
รูปที่ 3.2: แสดงผลตอบเชิงเวลาของระบบที่มีสัญญาขาเข้าเป็นความเร็วเชิงมุมของจานหมุนภายหลังการควบคุมแบบ LQR ต่อฟังก์ชั่นขั้นบันไดขนาด1 rad เริ่มที่เวลา 0 s

บทที่ 4
วงจรประยุกต์ของระบบเพนดูลัมผกผันแบบหมุน

4.1 โครงสร้าง



รูปที่ 4.1: มิติด้านข้างของแท่งวางของระบบเพนดูลัมผกผันแบบหมุน

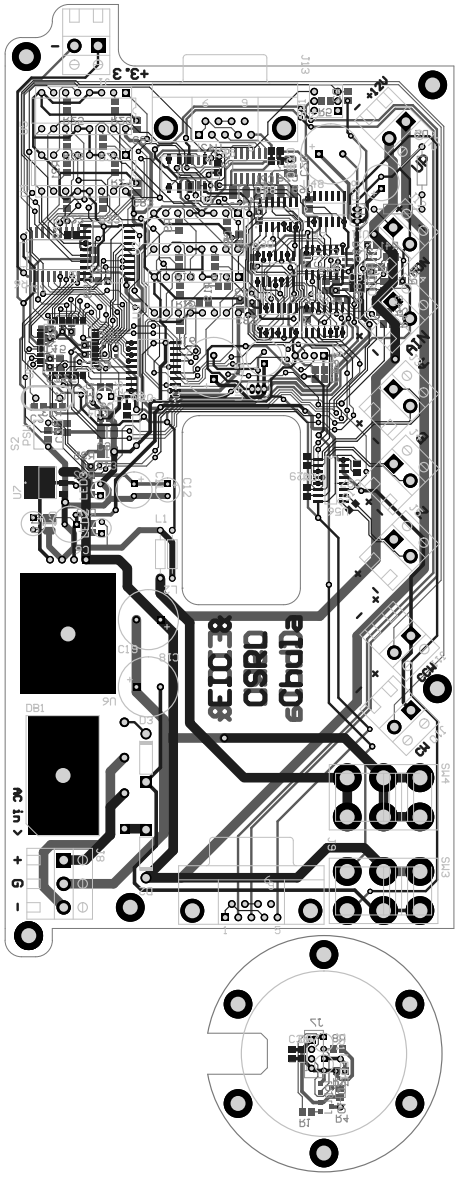


รูปที่ 4.2: มิติด้านบนของแท่งวางของระบบเพนดูลัมผกผันแบบหมุน

4.2 ส่วนควบคุม

ส่วนควบคุมประกอบด้วยไปด้วยสองส่วนหลักได้แก่

- **มอเตอร์กระแสสลับและวงจรขับเคลื่อนมอเตอร์** NHK มอเตอร์รุ่น M-JS1003FN001 สำหรับการควบคุมระบบเพนดูลัมผกผันแบบหมุน โดยใช้ ตัวขับเคลื่อน (Driver) รุ่น ESA-J1003B25-31

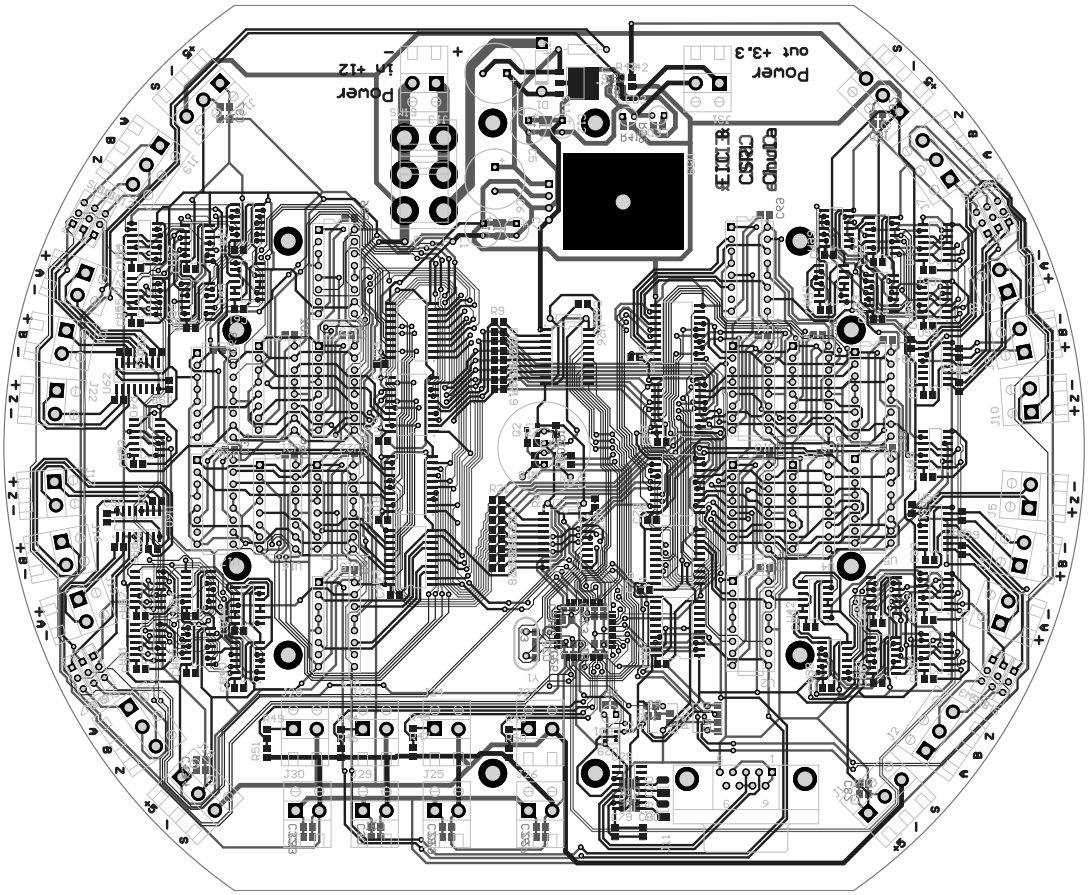


รูปที่ 4.3: ภาพแผงวงจรควบคุมมอเตอร์ของระบบเพนดูลัมผกผันแบบหมุน scale 1:1.6

- **วงจรควบคุม** ซึ่งประกอบด้วยวงจรที่สั่งให้มอเตอร์ทำงานและวงจรรับสัญญาณขาออกของระบบ วงจรทั้งสองมีแผนภาพวงจรดังรูปที่ 4.5 และ 4.6 และมีตำแหน่งของอุปกรณ์ดังรูปที่ 4.7 และ 4.8

4.3 การทำงานของวงจร

วงจรทำงานด้วยการต่อไฟเลี้ยง +12 Volt AC โดยใช้หม้อแปลงแปลงไฟจากไฟกระแสสลับ 220 Volt แล้วจ่ายผ่านทางช่องต่อ J8 ซึ่งวางอยู่ตรงมุมซ้ายบนของแผงวงจร (ดังรูปที่ 4.7) และมีลักษณะการเรียงตัวจากขวาไปซ้าย คือ + GND – นอกจากนี้ ยังสามารถจ่ายไฟออกจากแผงวงจรได้อีก 2 จุด ได้แก่ 3.3 Volt DC/DGND และ 12 Volt DC/DGND โดยต่อออกทางช่องต่อ J1 และ J2 ตามลำดับ สำหรับช่องต่อ J1 นั้นสามารถนำไปใช้ได้อเนกประสงค์ ส่วนช่องต่อ J2 ใช้สำหรับจ่ายไฟเลี้ยงให้แก่แผงวงจรเซนเซอร์ที่อยู่ทางด้านบนผ่านทางวงแหวนทองแดง (Slip Ring) ที่ติดตั้งอยู่กับแท่นยึดมอเตอร์



รูปที่ 4.4: ภาพด้านบนแผงวงจรเซนเซอร์ของระบบเพนดูลัมผกผันแบบหมุน scale 1:1.6

วงจรควบคุมมอเตอร์

วงจรควบคุมใช้ไมโครคอนโทรลเลอร์ LPC2148 (ARM7) ในการประมวลผลเพื่อควบคุมระบบเพนดูลัมผกผันแบบหมุน โดยวงจรนี้สามารถรับสัญญาณขาเข้า และให้สัญญาณขาออกได้หลายทาง ดังนี้

1. ช่องต่อ J6 ใช้สำหรับต่อกับแผงวงจรอินฟราเรดที่ติดตั้งเพิ่มไว้ใต้มอเตอร์เพื่อสื่อสารกับแผงวงจรเซนเซอร์ผ่านทาง RS-232 โดยจะรับข้อมูลจากแผงวงจรเซนเซอร์ทั้งข้อมูล แอนะล็อก และ ดิจิทัล ซึ่งวัดได้จากตัวเข้ารหัส (Encoder) ที่ติดอยู่กับก้านเพนดูลัม
2. ช่องต่อ AIN± ใช้สำหรับรับสัญญาณขาออกแอนะล็อก ±10 โวลต์ จาก ตัวขับ (รุ่น ESA-J1003B25-31) เพื่อบ่งบอกความเร็วปัจจุบันของมอเตอร์
3. ช่องต่อ MON± ใช้สำหรับส่งสัญญาณขาออก ±10 Volt ที่ขยายมาจากสัญญาณขาออก 0-3.3 Volt ของไมโครคอนโทรลเลอร์ ARM7 โดยจะมีหน้าที่ควบคุมความเร็วและแรงบิดของมอเตอร์
4. ช่องต่อ A±, B±, Z± ใช้สำหรับรับสัญญาณ Differential line จากตัวเข้ารหัสเพื่อระบุตำแหน่งปัจจุบันของมอเตอร์
5. ช่องต่อ CW±, CCW± ใช้สำหรับส่งสัญญาณขาออกเพื่อควบคุมตำแหน่งของมอเตอร์ ซึ่งควบคุมด้วยสัญญาณที่มีลักษณะเป็นขบวนของพัลส์ (Pulse train) และทิศทางที่ต้องการของมอเตอร์ ตามลำดับ
6. ช่อง RS232 ได้แก่ ช่องต่อ J9 และ J13 จะต่ออยู่กับ พอร์ตอนุกรมช่อง 0 ของ ARM7 ซึ่งเราสามารถเลือกได้ว่าจะให้ช่องสัญญาณนี้ต่อกับช่องต่อ J9 หรือ J13 ได้จาก SW2 และ P0.6 ของไมโครคอนโทรลเลอร์

ARM7 โดยช่องต่อ J9 จะติดต่อกับชุดขับมอเตอร์ และช่องต่อ J13 จะใช้สำหรับโปรแกรมตัวควบคุมเพนดุลัมหรือใช้สื่อสารกับคอมพิวเตอร์

วิธีโปรแกรมไมโครคอนโทรลเลอร์ ARM7 บนวงจรควบคุมมอเตอร์

1. กดสวิตซ์ SW2 ลง
2. เชื่อมสายอนุกรมระหว่างช่องต่อ J13 และคอมพิวเตอร์
3. โปรแกรมชุดคำสั่งของไมโครคอนโทรลเลอร์ผ่านโปรแกรม Phillips (LPC210X_ISP.exe) ทางพอร์ตอนุกรม
4. เมื่อป้อนโปรแกรม เสร็จแล้วกด SW2 (SW2 จะกลับขึ้นมา)
5. กดปุ่ม Reset (S2) 1 ครั้ง โปรแกรมจะเริ่มทำงาน

หลังจากไมโครคอนโทรลเลอร์ ARM7 เริ่มทำงานแล้ว เราจะสามารถเลือกช่องlyPPklของพอร์ตอนุกรมช่อง 0 ได้ 2 ช่อง

1. หากต้องการส่งข้อมูลไปยังคอมพิวเตอร์ผ่านช่องต่อ J13 สามารถทำได้โดยสั่ง P0.6 เป็น low
2. หากต้องการติดต่อกับชุดขับมอเตอร์ เพื่อปรับแต่งค่าตัวขับ สามารถทำได้โดยสั่ง P0.6 เป็น high การทำงานของตัวขับ

สำหรับชุดขับมอเตอร์รุ่น ESA-J1003B25-31 นั้นสามารถควบคุมมอเตอร์ได้ 3 แบบ (วิธี initialize และเลือกรูปแบบการควบคุมสามารถดูได้จากคู่มือ ESA25 ในภาคผนวก) ดังนี้

1. Position control mode สามารถกำหนดตำแหน่งที่ต้องการได้ 2 วิธี คือ
   (a) สัญญาณขาเข้าดิจิทัลผ่านคำสั่ง CW± และ CCW± ทางพอร์ต P0.16 และ P0.17 ของ ARM7 ตามลำดับ
   (b) ข้อมูลอนุกรม RS-232 ผ่านทางช่องต่อ J9
2. Velocity control mode สามารถกำหนดความเร็วที่ต้องการได้ 2 วิธี คือ
   (a) สัญญาณขาเข้าแอนะล็อกผ่านคำสั่ง AIN± ทางขา AOUT ของ ARM7 ซึ่งสัญญาณที่ออกมาจาก AOUT นั้นจะมีค่าตั้งแต่ 0 ถึง 3.3 Volt แล้วนำไปผ่านวงจรขยายเพื่อแปลงให้สัญญาณมีค่าตั้งแต่ -10 ถึง +10 Volt
   (b) ข้อมูลอนุกรม RS-232 ผ่านทางช่องต่อ J9
3. Torque control mode สามารถกำหนดแรงบิดที่ต้องการได้ 2 วิธี คือ
   (a) ใช้สัญญาณขาเข้าแอนะล็อกผ่านคำสั่ง AIN± ทางพอร์ต AOUT ของไมโครคอนโทรลเลอร์ ARM7
   (b) ใช้ RS-232 ผ่านทางช่องต่อ J9

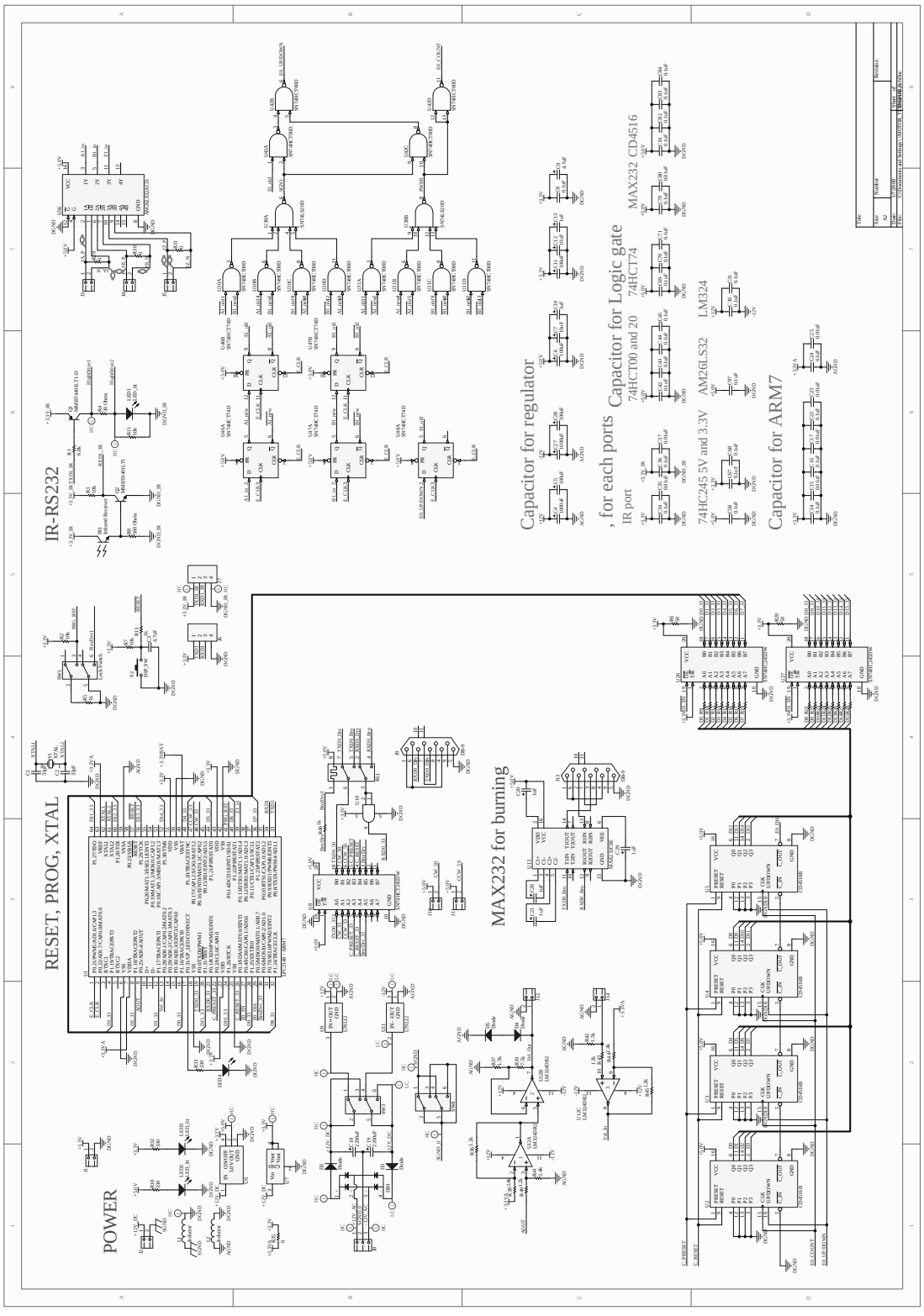
หมายเหตุ : สำหรับการเลือกสั่ง Position control command, Velocity control command หรือ Torque control command ผ่านทาง ARM7 นั้นจะต้องเลือกผ่านทางพอร์ตอนุกรม J9 การอ่านสัญญาณขาออกของมอเตอร์ทำได้โดยติดต่อขอข้อมูลผ่านทางพอร์ตอนุกรม J9 หรือ

- Feedback Position เป็นสัญญาณขาออกดิจิทัลจากช่องต่อ A±, B±, Z± ซึ่งถูกนับโดยวงจรนับ 16 บิตจะอ่านข้อมูลเข้าไมโครคอนโทรลเลอร์ผ่านทางพอร์ต P1.16 ถึง P1.31
- Feedback Velocity เป็นสัญญาณขาออกแอนะล็อกผ่านทางช่องต่อ MON± ซึ่งจะมีสัญญาณอยู่ในช่วง ±10 Volt แล้วจะถูกแปลงให้เป็นสัญญาณในช่วง 0 ถึง 3.3 Volt ด้วยวงจรลดทอนแรงดัน สัญญาณนี้จะถูกอ่านด้วย A/D ผ่านทางขา AN0.0 ของไมโครคอนโทรลเลอร์

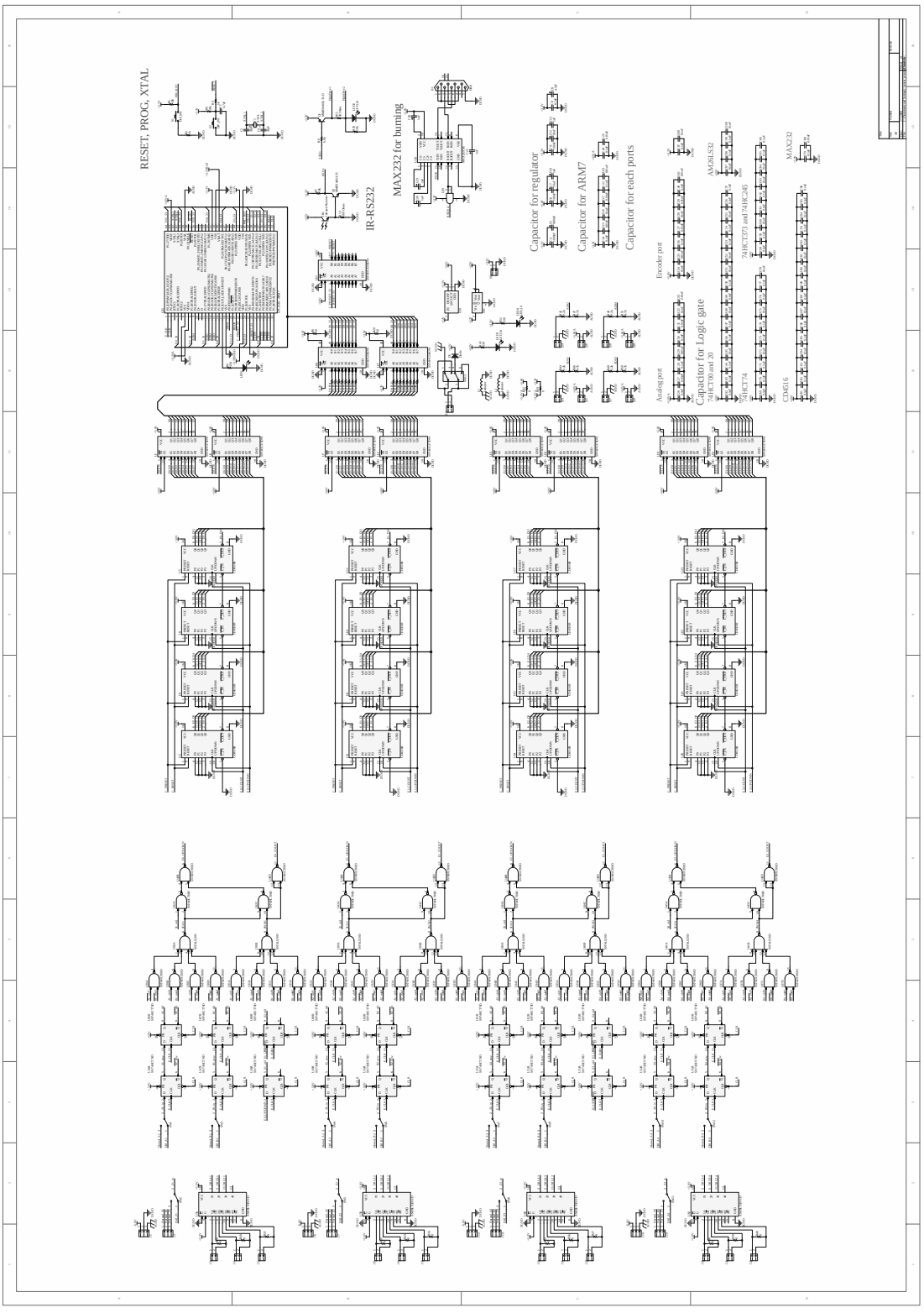
วงจรเซนเซอร์

วงจรเซนเซอร์ทำงานด้วยไฟเลี้ยงวงจรจากแผงวงจรด้านล่าง (วงจรควบคุมมอเตอร์ผ่านช่องต่อ J19) โดยจ่ายผ่านมาทางวงแหวนทองแดง ไฟเลี้ยงที่จ่ายมานี้เป็นไฟฟ้ากระแสตรงมีค่าประมาณ 12 Volt ซึ่งการจ่ายไฟเลี้ยงผ่านทางวงแหวนทองแดงนั้น ช่วยให้แผงวงจรเซนเซอร์สามารถหมุนได้อย่างอิสระ เนื่องจากไม่ต้องต่อสายไฟจากภายนอกเข้าแผงวงจร และวงจรจะแปลงไฟเลี้ยงวงจร 12 Volt ไปแปลงเป็นไฟเลี้ยงวงจร 3.3 Volt และ 5 Volt เพื่อใช้เป็นแหล่งจ่ายไฟฟ้าให้กับไมโครคอนโทรลเลอร์ (LPC2148 - ARM7)และอุปกรณ์ต่าง ๆ ในวงจร วงจรนี้ทำหน้าที่อ่านข้อมูลจากเซนเซอร์ต่าง ๆ และส่งข้อมูลนั้นกลับไปยังวงจรควบคุมมอเตอร์ผ่านทางวงจรส่งสัญญาณอินฟราเรดที่ติดตั้งไว้ช่วงกลางของแผงวงจรเซนเซอร์ โดยแผงวงจรอินฟราเรดนี้จะสามารถรับส่งสัญญาณได้ที่ อัตราบอด (Baud rate) สูงสุด 57,600 บิตต่อวินาที

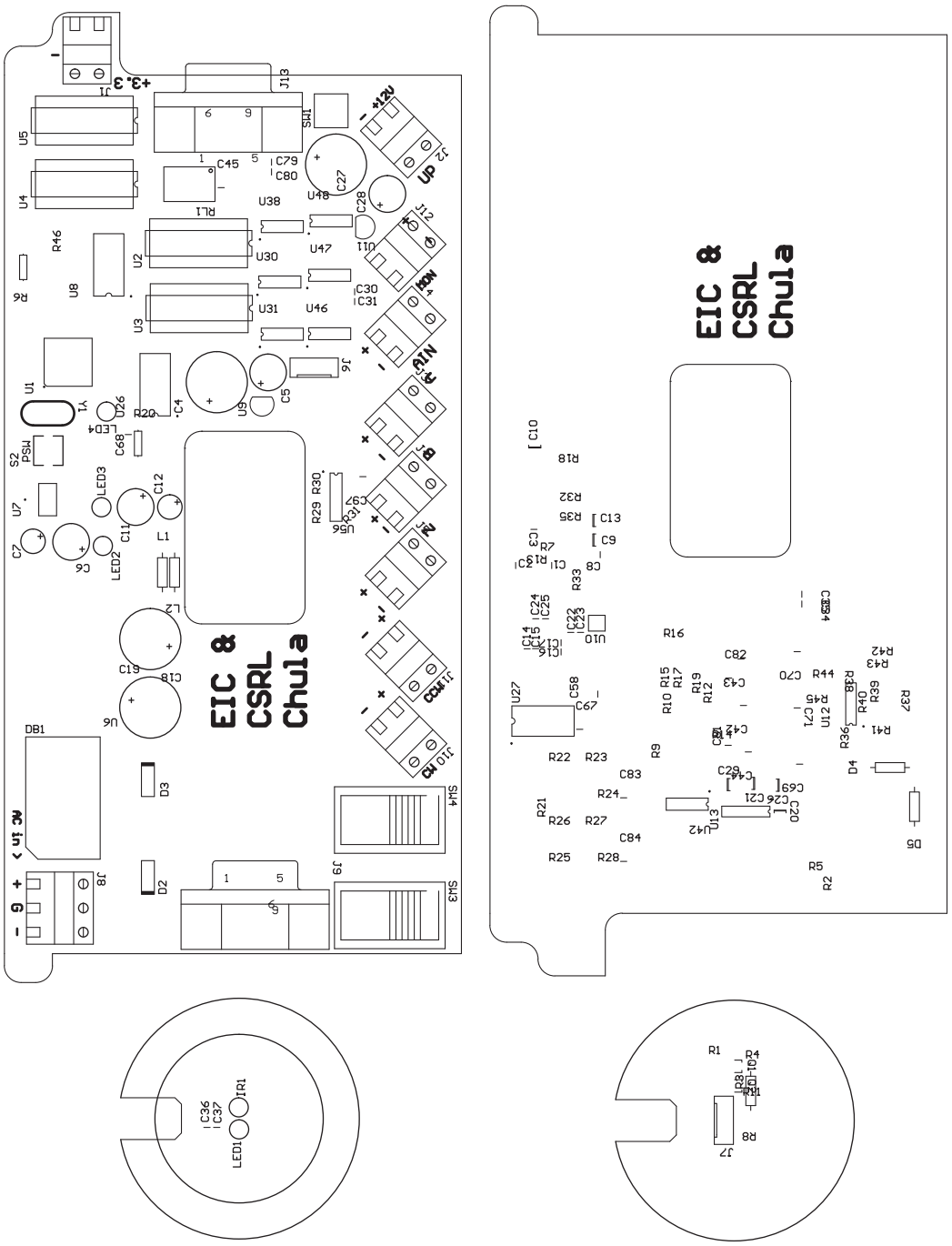
ภายในแผงวงจรเซนเซอร์มีวงจรนับตัวเข้ารหัสทั้งหมดสี่ชุดสามารถนับตัวเข้ารหัสแบบ quadrature ด้วยความละเอียดถึง 16 บิต โดยตัวเข้ารหัสที่ใช้มีความละเอียดของช่องสัญญาณ A หรือ B เท่ากับ 1250 พัลส์ต่อรอบ สัญญาณขาเข้าของวงจรนับนั้นเป็นได้ทั้งแบบ Differential line และแบบที่ไม่ใช่ Differential line โดยสามารถเลือกได้ทางสวิตซ์ที่วางอยู่บริเวณวงจรนับนั้น ๆ นอกจากวงจรนับแล้วยังสามารถอ่านสัญญาณขาเข้าที่เป็นสัญญาณแอนะล็อกได้อีกสี่ช่องทาง ผ่านทางช่องต่อ J23, J24, J27 และJ28 พร้อมช่องจ่ายไฟ 5 Volt สำหรับการติดตั้งเซนเซอร์แอนะล็อกชนิดอื่น อาทิ มาตรวัดความเร่งและไจโรสโคป ในอนาคต



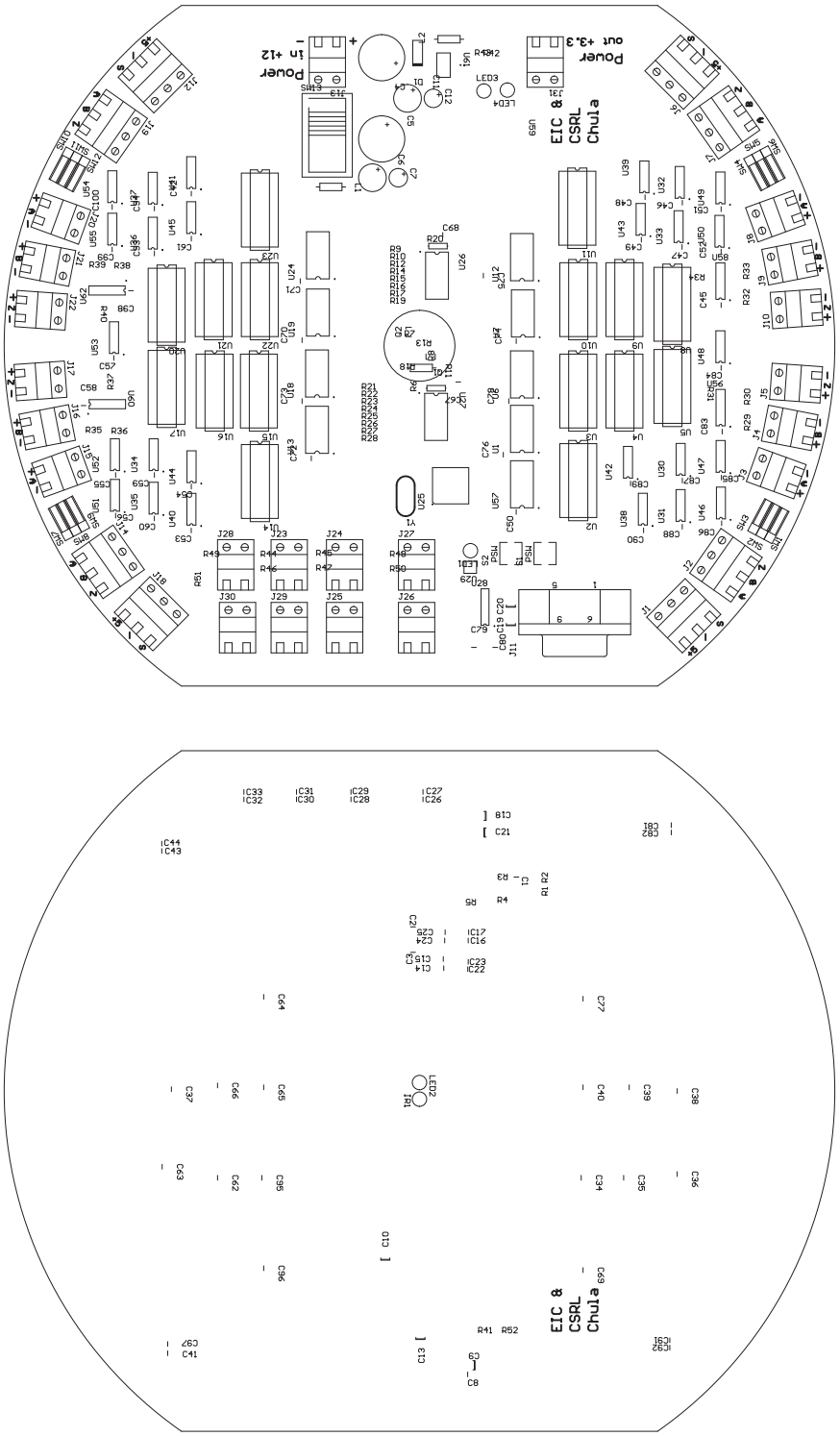
รูปที่ 4.5: แผนภาพวงจรควบคุมของระบบเพนดูลัมผกผันแบบหมุน



รูปที่ 4.6: แผนภาพวงจรเซนเซอร์ของระบบเพนดูลัมผกผันแบบหมุน



รูปที่ 4.7: ภาพตำแหน่งของอุปกรณ์ด้านบนและล่างในวงจรควบคุมของระบบเพนดูลัมผกผันแบบหมุน



รูปที่ 4.8: ภาพตำแหน่งของอุปกรณ์ด้านบนและล่างในวงจรเซนเซอร์ของระบบเพนดูลัมผกผันแบบหมุน

บทที่ 5
การหาค่าเอกลักษณ์ของระบบ

การหาค่าเอกลักษณ์ของระบบแบ่งเป็นสองส่วนคือ การหาเอกลักษณ์ของจานหมุน และการหาเอกลักษณ์ของแท่งเพนดูลัม

5.1 การหาเอกลักษณ์ของจานหมุน

การหาเอกลักษณ์ของจานหมุนด้วย Parametric method : ARX

แท่งเพนดูลัมถูกถอดออกจากระบบ จากนั้นสัญญาณควบคุมแรงบิดแบบสุ่มถูกจ่ายให้กับมอเตอร์ ความเร็วของมอเตอร์ถูกวัดผ่าน encoder ที่ติดอยู่กับมอเตอร์ จากผลการทดลองสุ่มการใช้แบบจำลองระบบแบบต่าง ๆ พบว่า การหาเอกลักษณ์ของระบบให้ผลการประมาณที่ดีที่สุดเมื่อ ใช้แบบจำลองของระบบ ดังนี้

$$v(k) = a_1 v(k-1) + b_1 \tau(k - n_c) \tag{5.1}$$

จากรูป 5.1 และ 5.2 และตาราง 5.1 คอรีเลชั่นของทอร์คและความเร่งเชิงมุมมีค่ามากที่สุดเมื่อมีการประวิง

การประวิงเวลา	0	1	2	3	4	5	6
คอรีเลชั่น	0.0357	0.6837	0.6772	0.1145	-0.1597	-0.0236	-0.1092

ตารางที่ 5.1: คอรีเลชั่นระหว่างทอร์คและความเร่งเชิงมุมที่มีการชดเชยการประวิงเวลาค่าต่าง ๆ

เวลาของสัญญาณขาเข้าเราจึงเลือก $n_c = 1$ และจากการหาเอกลักษณ์ของระบบจากข้อมูลชุดนี้มีค่า $a_1 = 0.9734, b_1 = 0.4467,$ Loss function $= 0.366402$ เมื่อระบุว่าอัตราสุ่มของการหาเอกลักษณ์คือ $f = 75\text{Hz}$ จะได้แบบจำลองของระบบเป็น

$$\omega(k) = 0.9734\omega(k-1) + 0.4467\tau(k-1) \tag{5.2}$$

เมื่อเปรียบเทียบกับระบบต่อเนื่องซึ่งมีลักษณะดังนี้

$$\dot{\omega}(t) = A\omega(t) + B\tau(t) \tag{5.3}$$

แปลงระบบนี้เป็นระบบแบบไม่ต่อเนื่องจะได้

$$\omega(t+T) = e^{AT}\omega(t) + \frac{B}{A}\left(e^{AT} - 1\right)\tau(t) \tag{5.4ก}$$

$$\omega(k) = e^{AT}\omega(k-1) + \frac{B}{A}\left(e^{AT} - 1\right)\tau(k-1) \tag{5.4ข}$$

จะได้ว่า

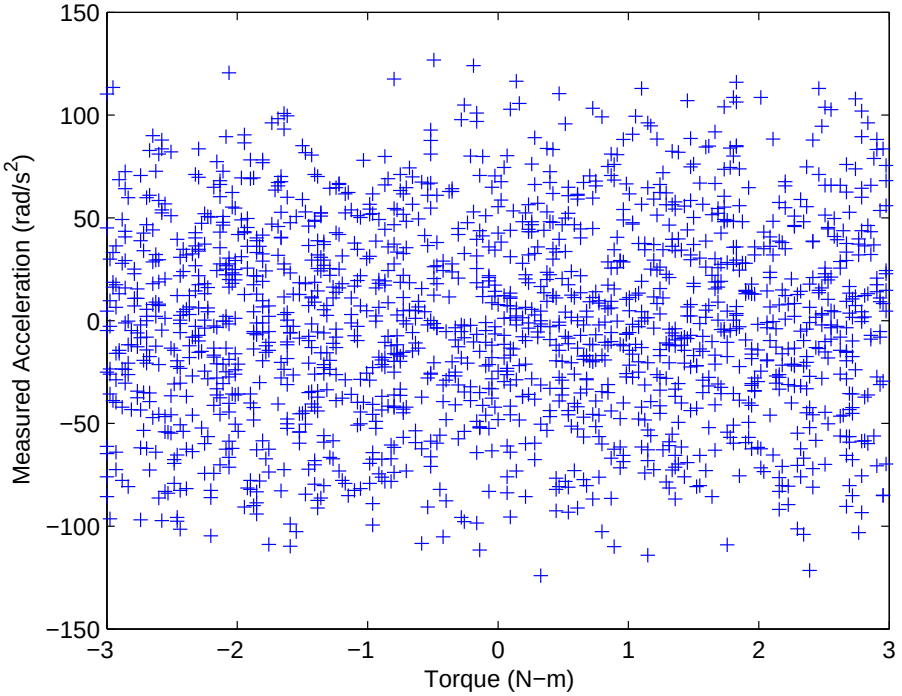
$$a_1 = e^{AT}, \qquad b_1 = \frac{B}{A}\left(e^{AT} - 1\right), \tag{5.5ก}$$

$$A = \frac{\log(a_1)}{T}, \qquad B = \frac{b_1 \log(a_1)}{T(a_1 - 1)} \tag{5.5ข}$$

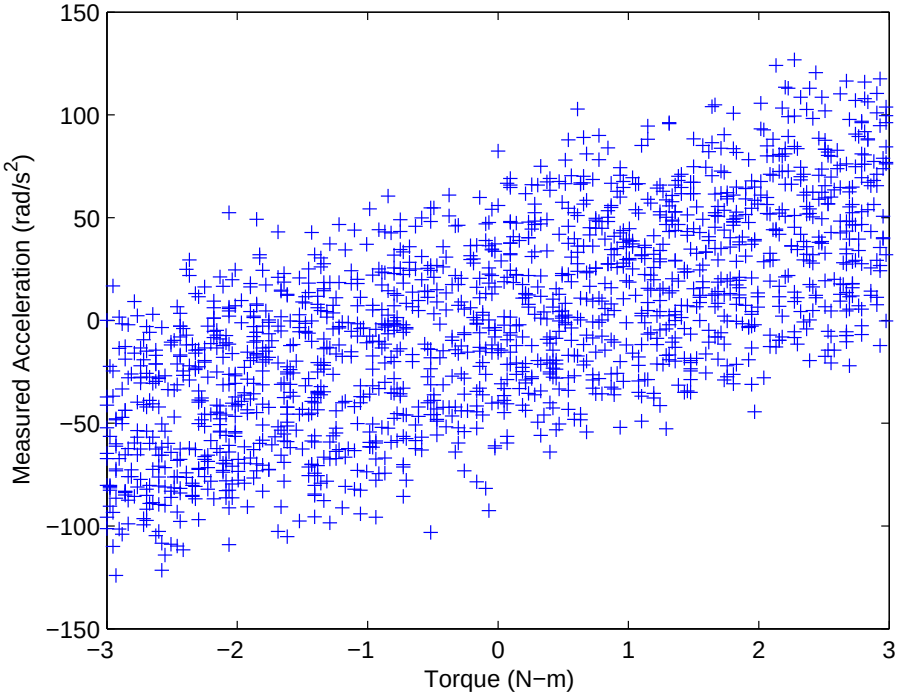
$$A = -2.0220, \qquad B = 33.9561 \tag{5.5ค}$$

เมื่อเขียนให้อยู่ในรูปสมการเชิงกล

$$\tau = J_0\dot{\omega} - c_0\omega = \frac{1}{B}\dot{\omega} - \frac{A}{B}\omega \tag{5.6}$$



รูปที่ 5.1: ความสัมพันธ์ระหว่างทอร์คและความเร่งเชิงมุมเมื่อไม่มีการชดเชยการประวิงเวลา



รูปที่ 5.2: ความสัมพันธ์ระหว่างทอร์คและความเร่งเชิงมุมเมื่อมีการชดเชยการประวิงเวลา 2 คาบการสุ่ม

ซึ่งจะได้ว่า

$$J_0 = 0.0294 \quad \mathrm{kg \cdot m^2}, \qquad c_0 = -0.0595 \quad {}^{\mathrm{N \cdot m}}/_{\mathrm{rad/s}} \tag{5.7}$$

โดยที่ J_0 คือโมเมนต์ความเฉื่อยของจานหมุนและ c_0 คือสัมประสิทธิ์แรงหนืดตามการหมุนของจานหมุน

เมื่อนำค่าที่ได้จากการหาเอกลักษณ์มาทดสอบกับการทดสอบผลตอบสนองขั้นของระบบจะมีลักษณะกราฟสีเขียวดังรูป 5.3 ซึ่งค่อนข้างแตกต่างจากความเร่งที่วัดได้จากระบบจริง จึงคะเนโครงสร้างของแบบจำลองคณิตศาสตร์โดยเพิ่มผลของแรงเสียดทานที่แปรผันตามความเร็วแบบไม่เชิงเส้นอย่างแรง จาก (5.2) จึงถูกประมาณเป็นแบบจำลองที่เป็นสมการเชิงอนุพันธ์ดัง (5.8)

$$\dot{\omega} = 47\tau - 0.47\omega - f(\omega) \tag{5.8}$$

โดยที่ $f(\omega)$ เป็นฟังก์ชันของ ω ฟังก์ชันอิ่มตัวที่มีความชันเป็น 4 เมื่อ $-5 \leq \omega \leq 5$ และมีค่าคงที่ -20 และ 20 ในช่วงอื่น มีลักษณะดังรูป 5.4 ผลการจำลองระบบโดยใช้แบบจำลอง (5.8) ให้ผลการจำลองที่ดีกว่าดังเส้นสีแดงในรูป 5.3 การเปลี่ยนแปลงแบบจำลองทำให้ค่า J_0 เปลี่ยนไปจากเดิมและ c_0 มีค่าไม่สม่ำเสมอและมีแนวโน้มดังรูป 5.4 จาก (5.8) สามารถเขียนให้อยู่ในรูปสมการเชิงกลได้ดังนี้

$$J_0\dot{\omega} = \tau - F(\omega) \tag{5.9}$$

โดย J_0 มีค่า $1/47$ และ F คือแรงที่เป็นฟังก์ชันไม่เชิงเส้นของ ω มีความชันเปลี่ยนแปลงตามค่า ω

5.2 การหาเอกลักษณ์ของแท่งเพนดูลัม

ในรายงานฉบับนี้ใช้ผลตอบสนองจากการแกว่งของแท่งเพนดูลัมเพื่อหาพารามิเตอร์จากการแกว่งของแท่งเพนดูลัม ผลตอบสนองการแกว่งของแท่งเพนดูลัมแท่งยาวมีลักษณะดังแสดงในรูป 5.5 ซึ่งจะเห็นได้ว่า เพนดูลัมมีคาบของการแกว่ง T เท่ากับ 1.0043 วินาที มีความถี่ของการแกว่งเท่ากับ $\frac{2\pi}{T} = 6.0462$ ${}^{\mathrm{rad}}/_{\mathrm{s}}$ จากข้อมูลในรูป 5.5 จะคำนวณได้ว่ามุมของแท่งเพนดูลัมลดลงด้วยอัตรา $e^{-0.0282t}$ และลู่เข้าสู่ค่า -0.1413 และมีค่าเริ่มต้นที่ 0.695 ขั้นตอนการคำนวณมีดังนี้ ขั้นแรกกำหนดสมการของเส้นที่ลากผ่านจุดยอดทุกจุดของกราฟเป็น

$$\beta(t) = \beta_i e^{at} - c \tag{5.10}$$

ค่าพารามิเตอร์ β_i, a และ c สามารถหาได้โดยใช้วิธี least square กับเซตของพิกัดของจุดยอดแต่ละจุดของกราฟ แต่เนื่องจากสมการ 5.10 ไม่เชิงเส้นจึงต้องมีกระบวนการแปลงดังนี้

- เก็บข้อมูลสองชุดที่มีค่ามุมเริ่มต้นไม่เท่ากันแล้วนำข้อมูลทั้งสองมา ลบกันเพื่อตัดตัวแปร c ออกจากสมการจะได้

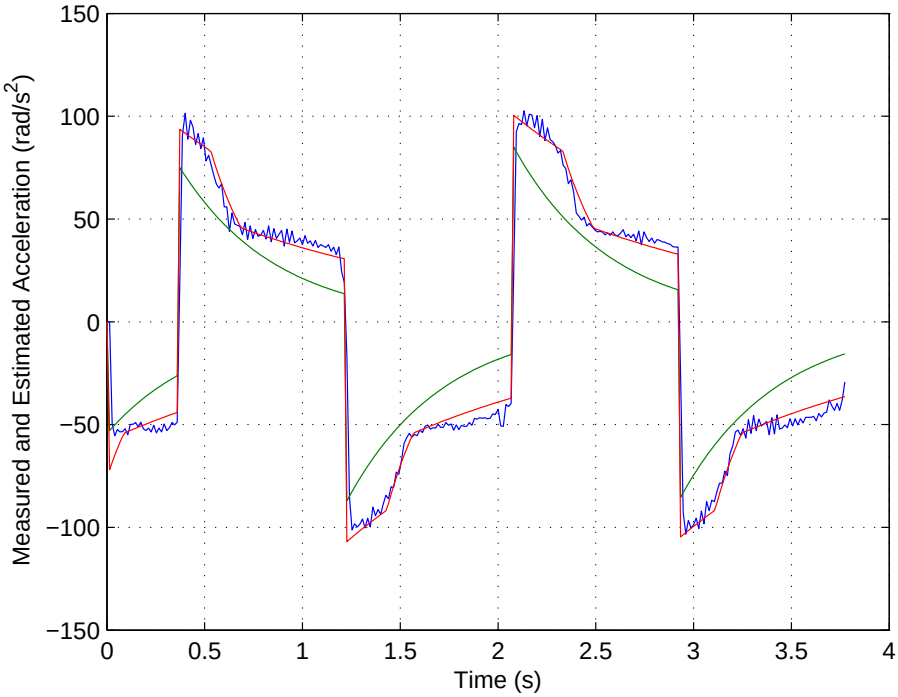
$$\beta_1(t) - \beta_2(t) = (\beta_{1i} - \beta_{2i})e^{at} \tag{5.11ก}$$

$$\frac{\beta_1(t) - \beta_2(t)}{\beta_{1i} - \beta_{2i}} = e^{at} \tag{5.11ข}$$

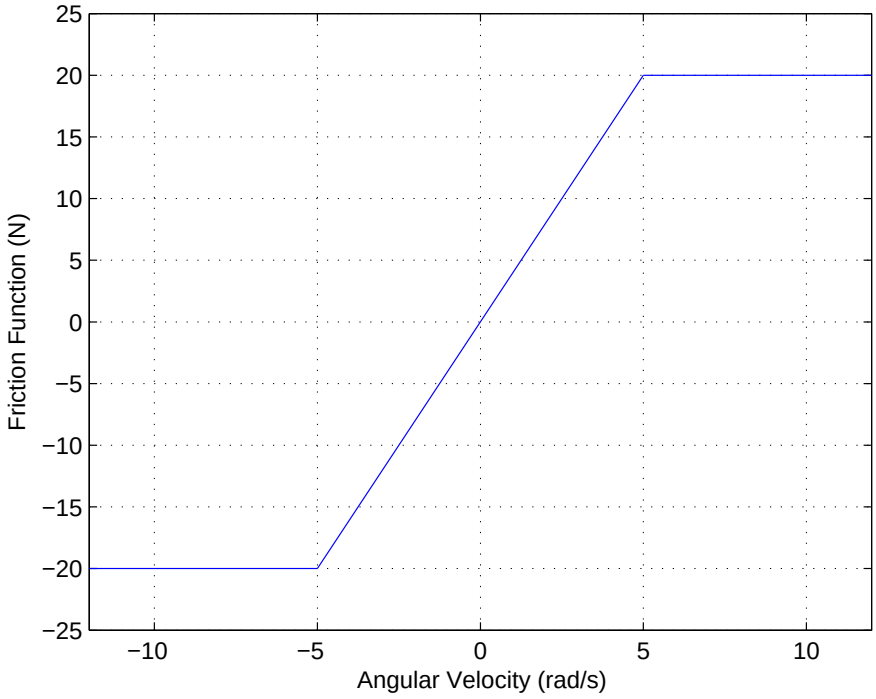
$$\log\left(\frac{\beta_1(t) - \beta_2(t)}{\beta_{1i} - \beta_{2i}}\right) = at \tag{5.11ค}$$

โดย $\beta_{1i} - \beta_{2i}$ เป็นค่าที่ทราบแน่นอนซึ่งสามารถใช้หลักการ least square เพื่อหา a ได้

- จากนั้นเลือกข้อมูลมาชุดหนึ่ง แทนค่า a ใน (5.10) จากนั้นใช้หลักการ least square เพื่อหาค่า c และหาค่า β_i ที่แม่นยำยิ่งขึ้น ซึ่งได้ค่า $a = -0.0282$, $\beta_i = 0.8363$ และ $c = 0.1413$



รูปที่ 5.3: การเปรียบเทียบผลตอบสนองสัญญาณทอร์คเป็นขั้นและความเร่งของจานเพนดูลัมทั้งที่วัดได้และที่ได้จากการจำลองโดยใช้แบบจำลองทั้งสองแบบ



รูปที่ 5.4: แบบจำลองความสัมพันธ์ระหว่างความเร็วและแรงเสียดทานที่เกิดขึ้นในจานหมุน

เส้นสีเขียวและแดงในรูป 5.5 เป็นกราฟของสมการที่ได้มาจากการหาค่า β_i, a และ c ด้วยวิธีนี้ ค่าพารามิเตอร์เหล่านี้จะถูกนำไปเปรียบเทียบกับสมการพลวัตของการแกว่งของแท่งเพนดูลัม พิจารณาสมการพลวัตต่อไปนี้

$$\left(ml^2 + J_1\right)\ddot{\beta} = -mgl\sin(\beta) - \eta_2\dot{\beta} - \eta_3\text{sgn}\left(\dot{\beta}\right) \qquad \sin(\beta) \approx \beta \tag{5.12ก}$$

$$\ddot{\beta} = c_1\beta + c_2\dot{\beta} + c_3\text{sgn}\left(\dot{\beta}\right) \tag{5.12ข}$$

โดย η_2 และ η_3 คือสัมประสิทธิ์แรงหนืดการหมุนและแรงเสียดทานคงตัวของแกนหมุนแท่งเพนดูลัม และ c_1, c_2 และ c_3 เป็นค่าคงตัวที่สมมติขึ้นและเป็นค่าคงตัวที่ต้องการทราบค่า เขียนสมการ 5.12ข ให้อยู่ในรูปเมทริกซ์

$$\begin{bmatrix} \ddot{\beta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} c_2 & c_1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\beta} \\ \beta \end{bmatrix} + \begin{bmatrix} c_3\text{sgn}\left(\dot{\beta}\right) \\ 0 \end{bmatrix} \tag{5.13}$$

ให้

$$A = \begin{bmatrix} c_2 & c_1 \\ 1 & 0 \end{bmatrix}, \qquad X = \begin{bmatrix} \dot{\beta} \\ \beta \end{bmatrix} \qquad B = \begin{bmatrix} c_3 \\ 0 \end{bmatrix} \tag{5.14}$$

ใช้การแปลงลาปลาซช่วยโดยสมมติให้ $\text{sgn}\left(\dot{\beta}\right)$ มีค่าเป็นบวกมีความเร็วในทิศทางเดียวจะได้

$$\dot{X} = AX + B \tag{5.15}$$

$$s\mathbb{X} - X(0) = A\mathbb{X} + \frac{1}{s}B \tag{5.16}$$

$$\mathbb{X} = \frac{sX(0) + B}{s(s - A)} \tag{5.17}$$

และ

$$X = e^{At}\left(X(0) + A^{-1}B\right) - A^{-1}B \tag{5.18}$$

ดังนั้นผลเฉลยของ X จึงมีผลตอบขึ้นกับ pole หรือ eigenvalue ของระบบ ซึ่งจากผลตอบของระบบจริงข้างต้นระบุได้ว่า pole ของระบบมีค่าเท่ากับ $-0.0282 \pm 6.0462i$

$$s^2 - c_2 s - c_1 = 0 \tag{5.19}$$

$$s^2 + 0.0564s + 39.1408 = 0 \tag{5.20}$$

$$c_1 = -36.5557, \qquad c_2 = -0.0564 \tag{5.21}$$

จากนั้นเพื่อหา c_3 จึงต้องมีการคำนวณการสูญเสียพลังงานจากพลังงานตั้งต้นตามสมการนี้

$$\text{P.E.}(0) + \text{K.E.}(0) = E_{(\text{loss}-\text{C2})} - E_{(\text{loss}-\text{C3})} \tag{5.22}$$

โดยที่พลังงานศักย์ตั้งต้น $\text{P.E.}(0)$ นั้นคำนวณมาจากสมการ

$$\text{P.E.}(0) = mgl(1 - (1 - \frac{(\beta_i - c)^2}{2})) \tag{5.23}$$

พลังงานศักย์ตาม (5.23) เป็นพลังงานศักย์ของระบบที่ประมาณ $\sin(\beta) \approx \beta$ และพลังงานจลน์ตั้งต้น $\text{K.E.}(0)$ นั้นคำนวณมาจากสมการ

$$\text{K.E.}(0) = \frac{1}{2}(ml^2 + J_1)\left(a(\beta_i - c)\right)^2 \tag{5.24}$$

จากนั้นพิจารณา $E_{(\text{loss}-\text{C2})}$ จะได้

$$E_{(\text{loss}-\text{C2})} = \eta_2 \int_0^t \dot{\beta} \mathrm{d}\beta = \eta_2 \int_0^t \dot{\beta}^2 \mathrm{d}t \tag{5.25}$$

พิจารณา $E_{(\text{loss}-\text{C3})}$ จะได้

$$E_{(\text{loss}-\text{C3})} = \eta_3 \int_0^t \text{sgn}(\dot{\beta}) \mathrm{d}\beta \tag{5.26}$$

เมื่อนำ $ml^2 + J_1$ หารทั้งสองข้างของสมการจะได้

$$\frac{mgl}{ml^2 + J_1} \frac{(\beta_i - c)^2}{2} + \frac{1}{2} \left(a(\beta_i - c)\right)^2 = \frac{\eta_2}{ml^2 + J_1} \int_0^t \dot{\beta}^2 \mathrm{d}t + \frac{\eta_3}{ml^2 + J_1} \int_0^t \text{sgn}(\dot{\beta}) \mathrm{d}\beta \tag{5.27ก}$$

$$c_1 \frac{(\beta_i - c)^2}{2} + \frac{1}{2} \left(a(\beta_i - c)\right)^2 = c_2 \int_0^t \dot{\beta}^2 \mathrm{d}t + c_3 \int_0^t \text{sgn}(\dot{\beta}) \mathrm{d}\beta \tag{5.27ข}$$

โดยการสมมติให้

$$\beta(t) = \left(\beta_i e^{at} - c\right) \cos(\omega t), \qquad \dot{\beta}(t) = \beta_i e^{at} \left(acos(\omega t) - \omega sin(\omega t)\right) + c\omega \sin(\omega t) \tag{5.28}$$

ซึ่งสามารถคำนวณได้ว่า $(c_2 = 2a)$

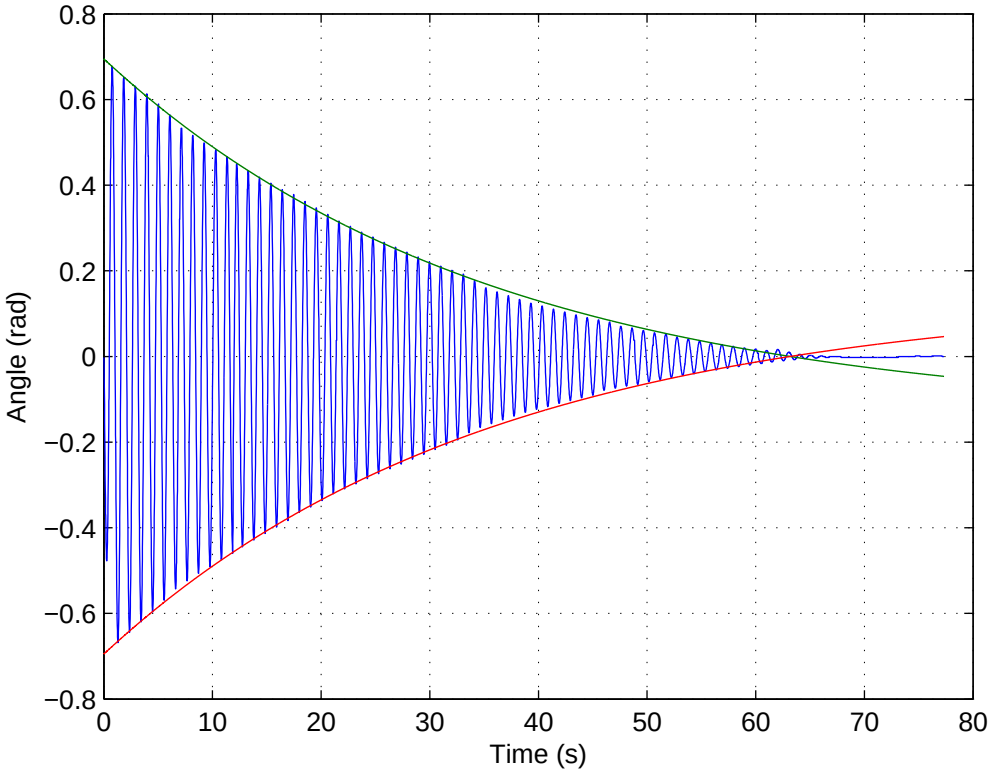
$$\begin{aligned}
c_2 \int_0^t \dot{\beta}^2 \mathrm{d}t &= c_2 \beta_i^2 \int_0^t e^{2at} \left(a \cos(\omega t) - \omega \sin(\omega t)\right)^2 dt \\
&\quad + 2c_2 \beta_i c\omega \int_0^t e^{at} \left(a \cos(\omega t) \sin(\omega t) - \omega \sin^2(\omega t)\right) \mathrm{d}t \\
&\quad + c_2 c^2 \omega^2 \int_0^t \sin^2(\omega t) \mathrm{d}t \\
&= -\frac{\beta_i^2}{2} \left(a^2 \cos(2\omega t) + a^2 - a\omega \sin(2\omega t) + \omega^2\right) e^{2at} \Big|_0^{t_f} \\
&\quad + \frac{2c\omega\beta_i}{(4\omega^2 + a^2)} \left(4\omega^3 + a^2\omega - (2a\omega^2 + a^3) \sin(2\omega t) + a^2 \omega \cos(2\omega t)\right) e^{at} \Big|_0^{t_f} \\
&\quad - \frac{c^2 a\omega}{2} \left(2\omega t_f - \sin(2\omega t_f)\right)
\end{aligned} \tag{5.29}$$

หลังจากแทนค่า $\beta_i, c, a, \omega, c_1, c_2$ และ t_f ลงใน (5.27) จะได้

$$8.8289 + 1.9139 \times 10^{-4} = 6.5373 + c_3 \int_0^t \text{sgn}(\dot{\beta}) \mathrm{d}\beta \tag{5.30ก}$$

$$c_3 = \frac{8.8289 + 1.9139 \times 10^{-4} - 6.5373}{60.6795} = 0.0378 \tag{5.30ข}$$

ซึ่งปริพันธ์พจน์สุดท้ายคือระยะทางที่เกิดแรงเสียดทานคงที่ทั้งหมดหาได้จากผลรวมของขนาดของมุมที่เคลื่อนที่ในความเร็วทั้งสองทิศ เมื่อนำ c_1, c_2 และ c_3 ที่คำนวณได้แทนในสมการ 5.12ข จะได้แบบจำลองคณิตศาสตร์ของการหมุนแท่งเพนดูลัมที่สมบูรณ์



รูปที่ 5.5: ผลตอบสนองการแกว่งของแท่งเพนดูลัมจากระบบจริง

บทที่ 6
การนำไปใช้กับระบบจริง

ในบทนี้แสดงการนำแบบจำลองที่ได้มาเพื่อออกแบบตัวควบคุมเพื่อนำไปใช้กับระบบจริงที่เป็นเพนดูลัมผกผันแบบแท่งเดียว

6.1 ประมาณเป็นเชิงเส้นเพื่อหาอัตราขยายการป้อนกลับโดย LQR

จากการหาเอกลักษณ์ของระบบพบว่าระบบมีความไม่เชิงเส้นอยู่ เราจึงละเลยผลความไม่เป็นเชิงเส้นนั้น เหลือเพียงแบบจำลองที่เป็นเชิงเส้นเท่านั้น เพื่อให้สามารถเขียนสมการพลวัตให้อยู่ในรูปของปริภูมิสถานะได้ (5.8) จึงเหลือเพียง

$$J_0\dot{\omega} = \tau - c_1\omega \tag{6.1}$$

โดยที่ J_0 มีค่า $1/47$ และ c_1 มีค่า 0.03 จากการเลือกค่า c_1 ให้ใกล้เคียงกับแบบจำลองที่ไม่เชิงเส้น และจากการละเลย c_3 ใน (5.12ข) จะได้สมการ

$$\left(ml^2 + J_1\right)\ddot{\beta} = -mgl\beta - \eta_2\dot{\beta} \tag{6.2}$$

เราทราบค่า m, l และ g ที่ค่อนข้างแน่นอน ดังนั้นจากค่าคงตัวที่ได้จากการหาหาเอกลักษณ์ของระบบ จึงสามารถคำนวณ J_1 ในสมการ (2.25) ได้เป็น $J_1 = 0.0019$ และเลือกค่า c_2 ให้ผลตอบสนองใกล้เคียงกับระบบที่ไม่เชิงเส้นได้ $c_2 = -4.626 \times 10^{-4}$ นำค่าคงตัวต่าง ๆ เหล่านี้ไปแทนใน (2.27) - (2.28) เพื่อหาแบบจำลองคณิตศาสตร์ที่เป็นเชิงเส้น ได้

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 61.24 & -0.536 \\ 0 & 0 & -0.535 & 0.0066 \end{bmatrix} \tag{6.3}$$

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -0.03 & 0 \\ 0 & 0.2415 & 0 & -4.626 \times 10^{-4} \end{bmatrix} \tag{6.4}$$

จากนั้นในกระบวนการทำ LQR กำหนดให้เมทริกซ์ Q มีค่าดังนี้

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 2000 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix} \tag{6.5}$$

และ R มีค่าเท่ากับ 1 สุดท้ายแล้ว จะได้อัตราขยายการป้อนกลับ K เป็น

$$K = \begin{bmatrix} -0.1 & 55.5 & -3.56 & 4.94 \end{bmatrix} \tag{6.6}$$

ถัดไปเป็นการทดลองใช้ค่าอัตราขยายการป้อนกลับที่คำนวณได้กับระบบจำลอง

6.2 ทดลองใช้กับระบบจำลองที่รวมความไม่เชิงเส้น

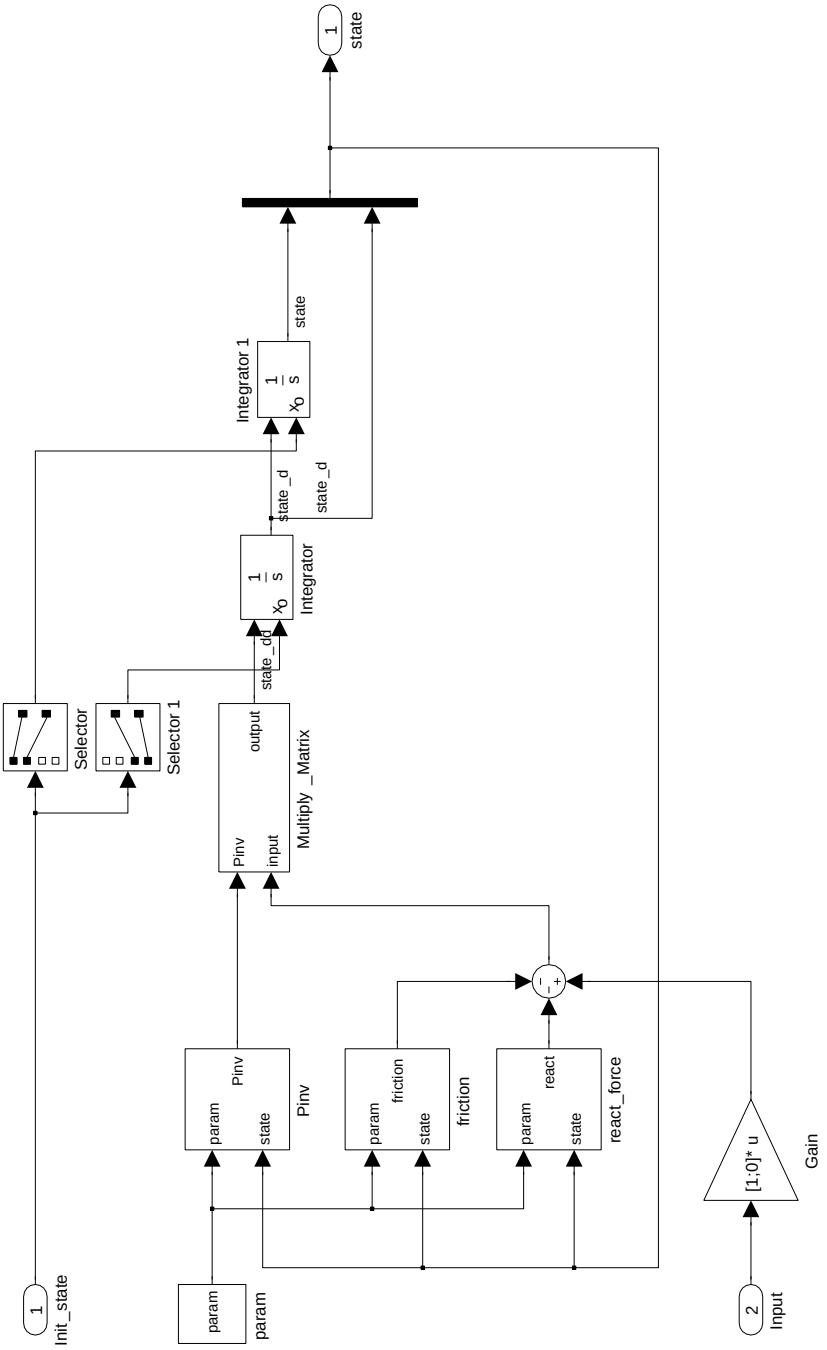
ระบบจำลองที่ไม่เชิงเส้นถูกเขียนให้อยู่ในรูปแผนภาพดังรูป 6.1 - 6.7 ซึ่งเป็นแบบจำลองใน Simulink Matlab ที่รวมความไม่เชิงเส้นที่มาจากแรงเสียดทานด้วย จากนั้นจึงทดลองควบคุมด้วยตัวควบคุมที่ออกแบบไว้แล้วโดยที่ตัวควบคุมมีแผนภาพบล็อกดังรูป 6.8 ผลการจำลองมีค่าดังรูป 6.9 และ 6.10

ผลตอบสนองที่ได้ในรูป 6.9 และ 6.10 แสดงให้เห็นว่าตัวควบคุมที่ออกแบบมาสามารถใช้ควบคุมระบบเพนดูลัมแท่งเดี่ยวแบบผกผันที่ไม่เชิงเส้นได้

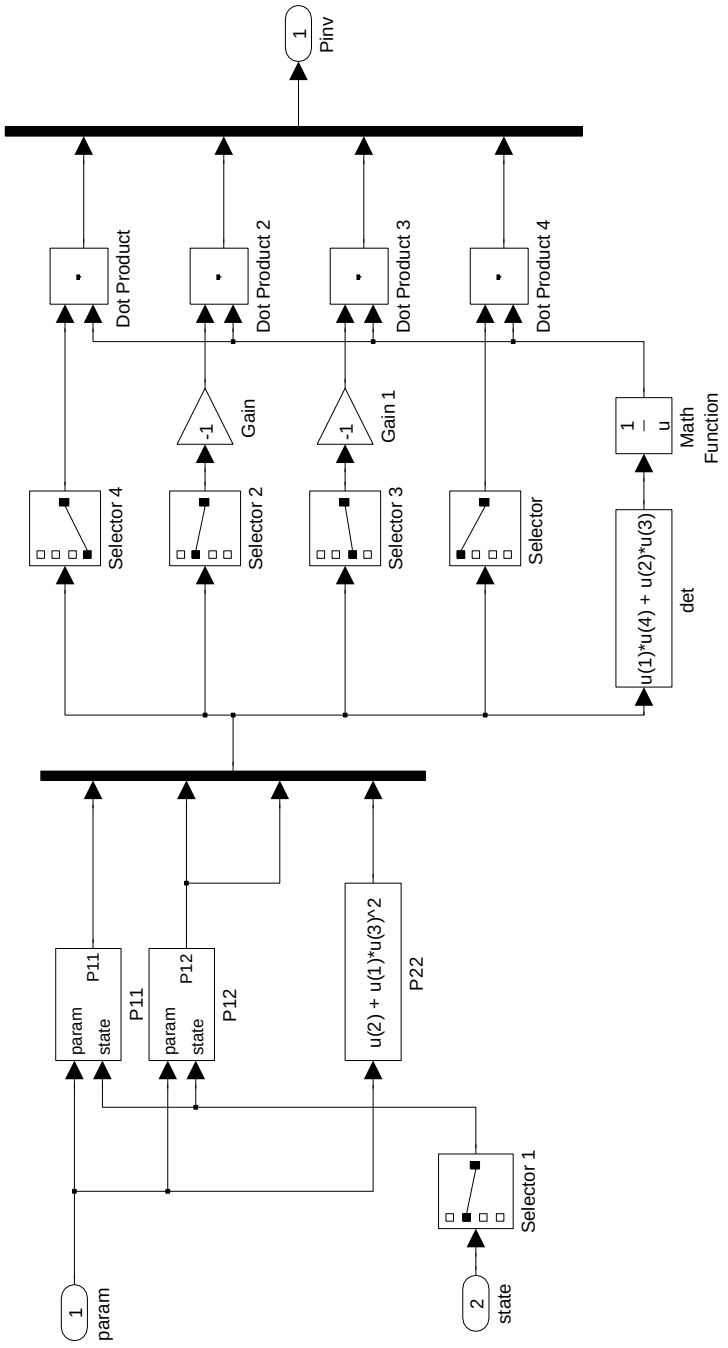
6.3 ทดลองใช้กับระบบจริง

จากนั้นเราจึงนำตัวควบคุมดังกล่าวมาใช้กับระบบจริง โดยจากรูป 6.8 นั้นแสดงการควบคุมในระบบจริงไว้อย่างชัดเจน คือ มีอัตราสุ่มของการควบคุมและการป้อนกลับเท่ากับ 75Hz สัญญาณควบคุมมีค่าเป็นขั้นไม่ต่อเนื่อง 256 ระดับมีค่าตั้งแต่ -3 ถึง $3\ \text{N}\cdot\text{m}$ และสัญญาณควบคุมที่มีค่ามากกว่า $3\ \text{N}\cdot\text{m}$ หรือน้อยกว่า $-3\ \text{N}\cdot\text{m}$ จะถูกจำกัดขอบเขตไม่ให้เกิน 3 และ $-3\ \text{N}\cdot\text{m}$ ตามลำดับ ในการทดลองควบคุมระบบนั้นทดลองทำควบคู่กับการแกว่งเพนดูลัมขึ้นอัตโนมัติโดยใช้วิธีอย่างง่ายคือ จากเพนดูลัมที่อยู่ ณ สถานะพักที่จุดต่ำสุด เพิ่มพลังงานให้แท่งเพนดูลัมทีละน้อยด้วยการแกว่งจานหมุนเพิ่มเมื่อแท่งเพนดูลัมมีพลังงานจลน์สูงสุด (ประมาณว่าเป็นจุดต่ำสุดของการแกว่งแท่งเพนดูลัม) จนกระทั่งแท่งเพนดูลัมมีพลังงานศักย์มากพอที่จะควบคุมแท่งเพนดูลัมให้อยู่ตำแหน่งศูนย์ (ด้านบนสุด) จึงเริ่มกระบวนการควบคุมแท่งเพนดูลัมโดยใช้ค่าอัตราขยายตัวควบคุมที่ออกแบบไว้ ในที่นี้กำหนดให้มุมบนสุดของการแกว่งแท่งเพนดูลัมมีค่าเบี่ยงเบนไม่เกิน 5 องศา

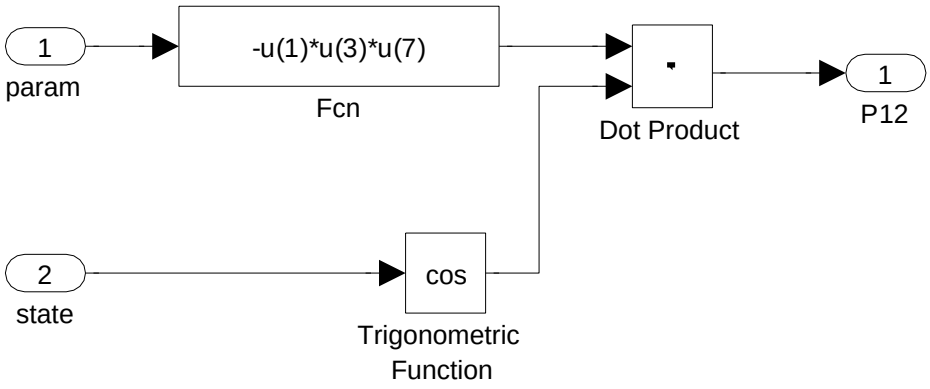
รูป 6.11 และ 6.13 แสดงถึงมุมของแท่งเพนดูลัมและจานหมุนในขณะที่มีการแกว่งแท่งเพนดูลัมขึ้นอัตโนมัติตามวิธีข้างต้น หลังจากนั้นเมื่อแท่งเพนดูลัมแกว่งขึ้นได้มุมที่เหมาะสมระบบควบคุมจึงเปลี่ยนมาเป็นการควบคุมแท่งเพนดูลัมให้อยู่ในตำแหน่งสูงสุดด้วยวิธี LQR ซึ่งมีผลตอบสนองดังรูป 6.12 และ 6.14 ตามลำดับ สังเกตได้ว่าระบบจริงกับระบบจำลองนั้นมีผลตอบสนองทางเวลาของการควบคุมไม่เท่ากัน ระบบจริงให้ผลตอบสนองที่ช้ากว่าเนื่องจากการจำลองระบบมีการตัดแรงเสียดทานบางส่วนออกไปจึงทำให้ระบบมีความหน่วงจริงมากกว่าระบบที่จำลองขึ้น



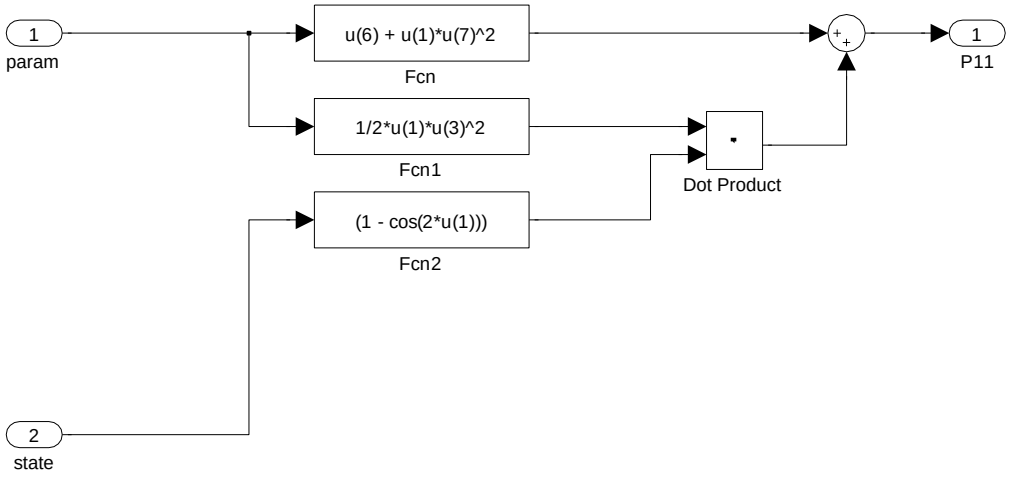
รูปที่ 6.1: แผนภาพแสดงแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น



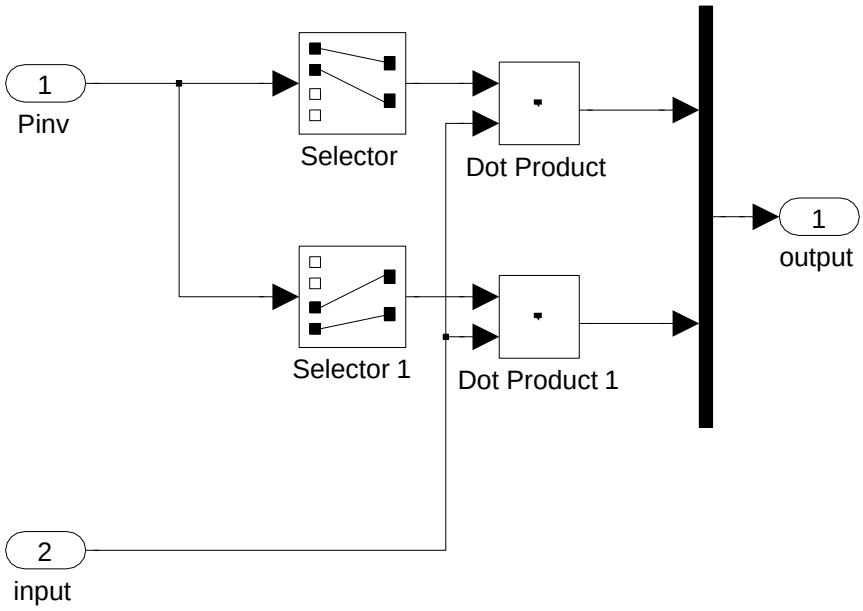
รูปที่ 6.2: แผนภาพย่อยที่ 1 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (Pinv)



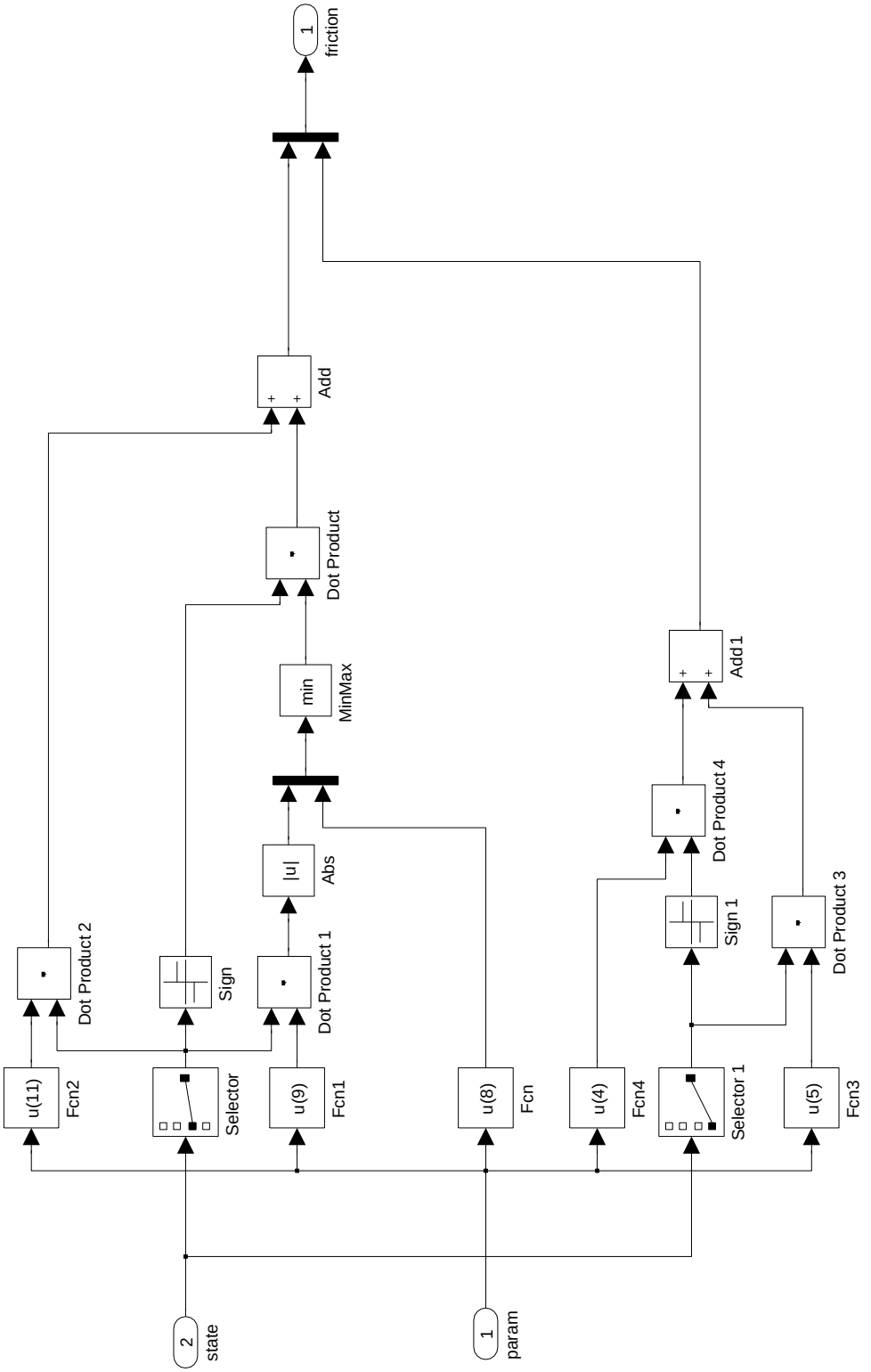
รูปที่ 6.3: แผนภาพย่อย 1-2 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (P12)



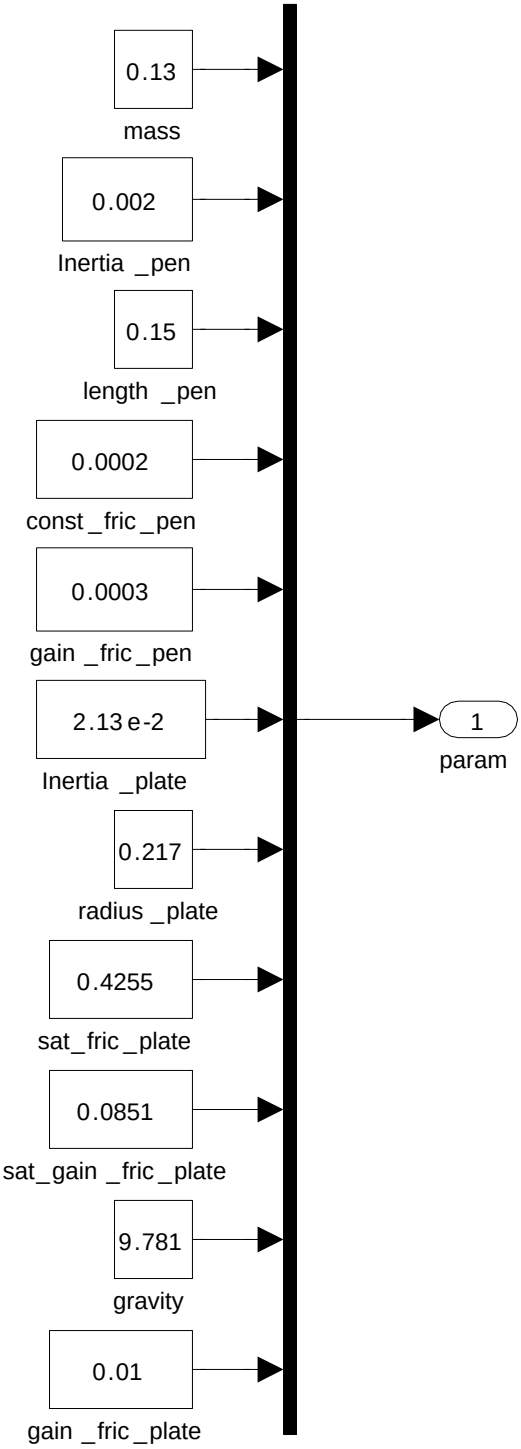
รูปที่ 6.4: แผนภาพย่อย 1-1 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (P11)



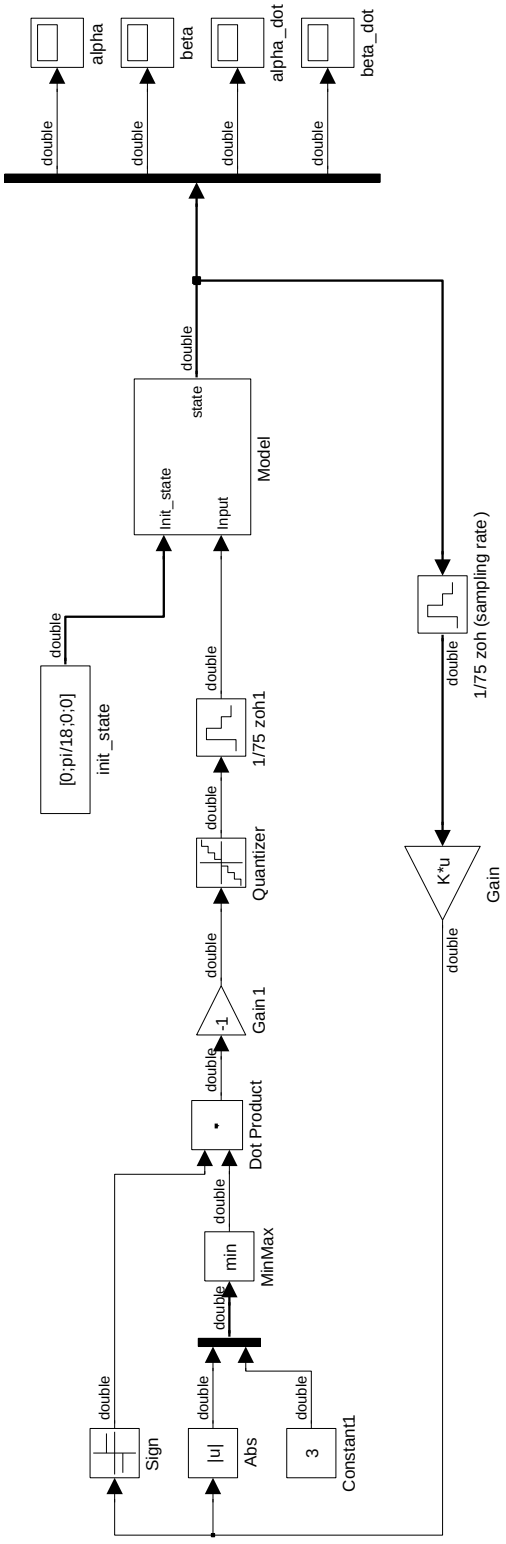
รูปที่ 6.5: แผนภาพย่อย 2 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (MultiplyMatrix)



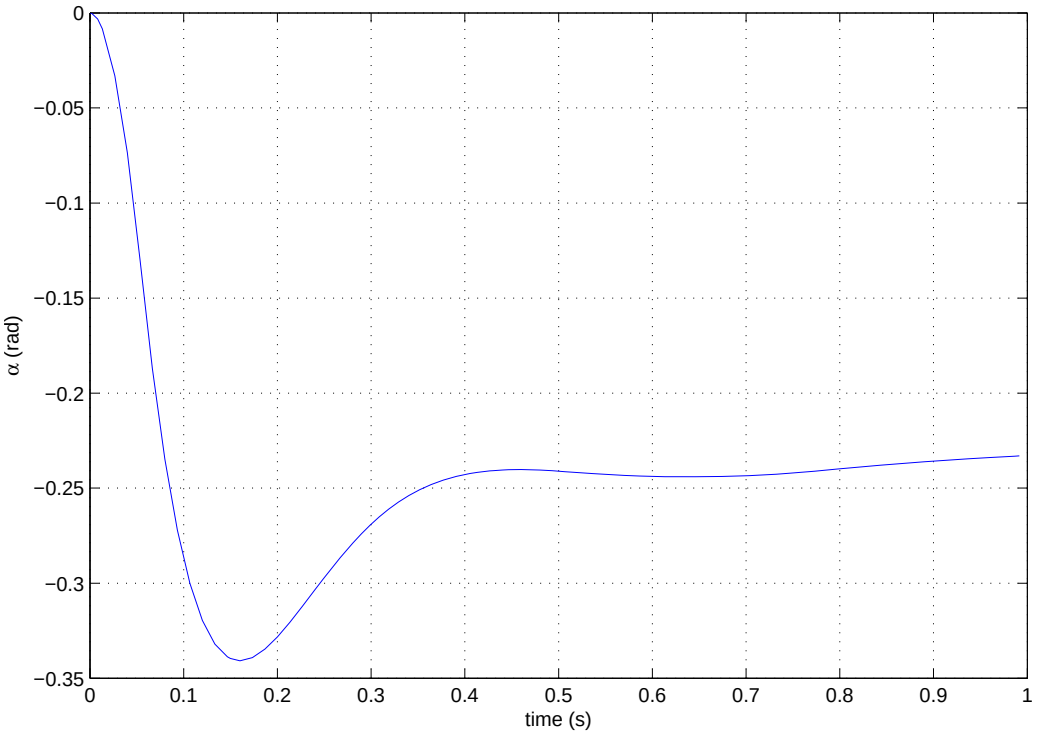
รูปที่ 6.6: แผนภาพย่อย 3 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (Friction)



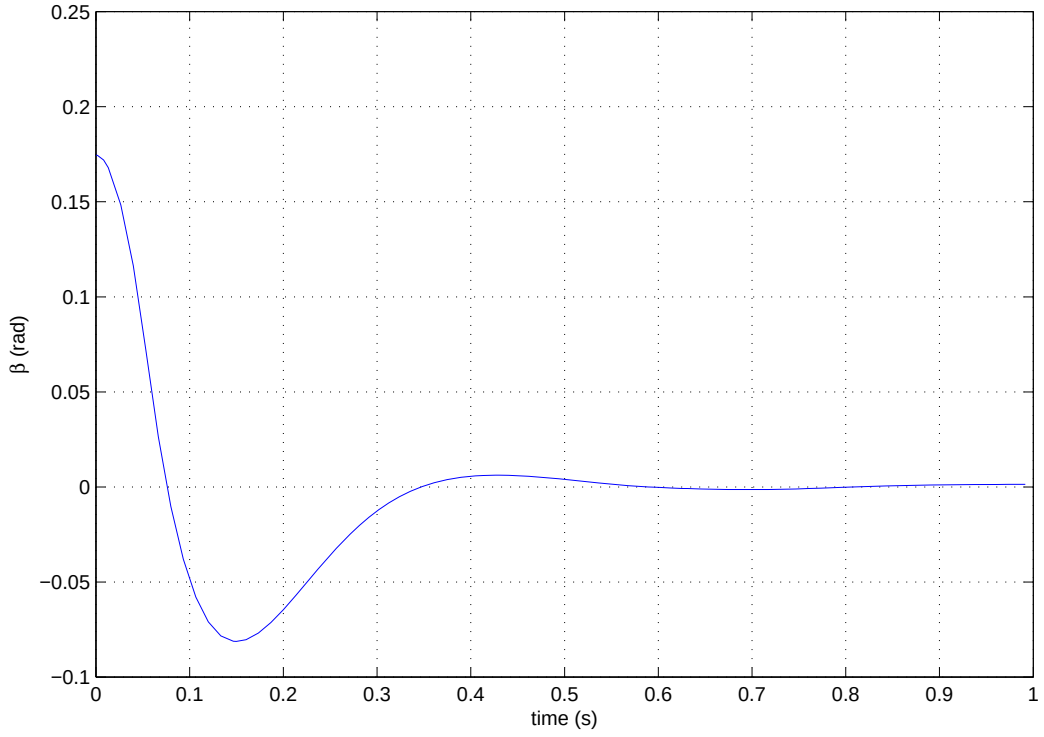
รูปที่ 6.7: แผนภาพย่อย 4 ของแบบจำลองของระบบเพนดูลัมไม่เชิงเส้น (Param)



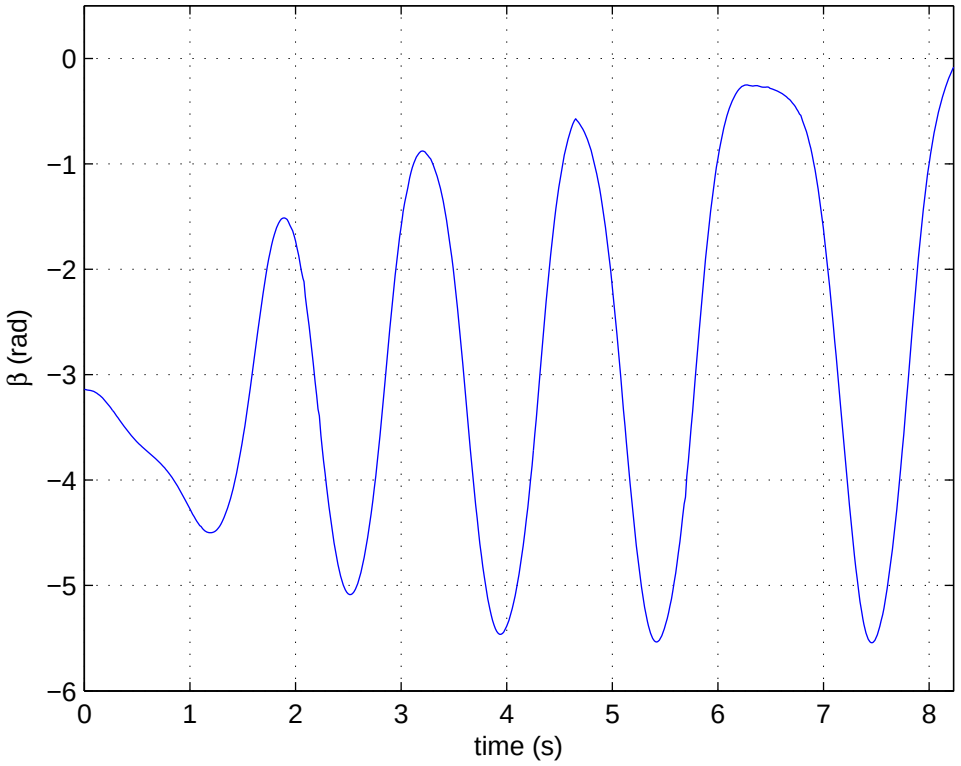
รูปที่ 6.8: แผนภาพแสดงการจำลองการควบคุมระบบเพนดูลัมไม่เชิงเส้น



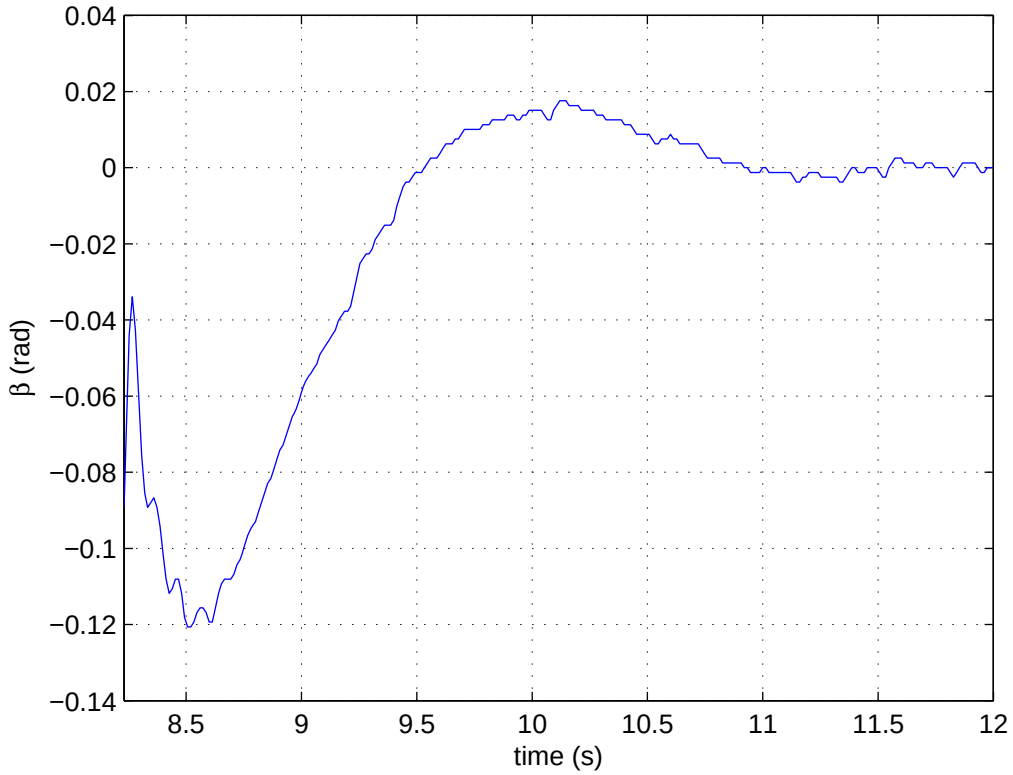
รูปที่ 6.9: การจำลองผลตอบสนองมุมจานหมุนของการควบคุมแท่งเพนดูลัมแบบผกผันที่มีมุมเบี่ยงเบนเริ่มต้น $\pi/18$



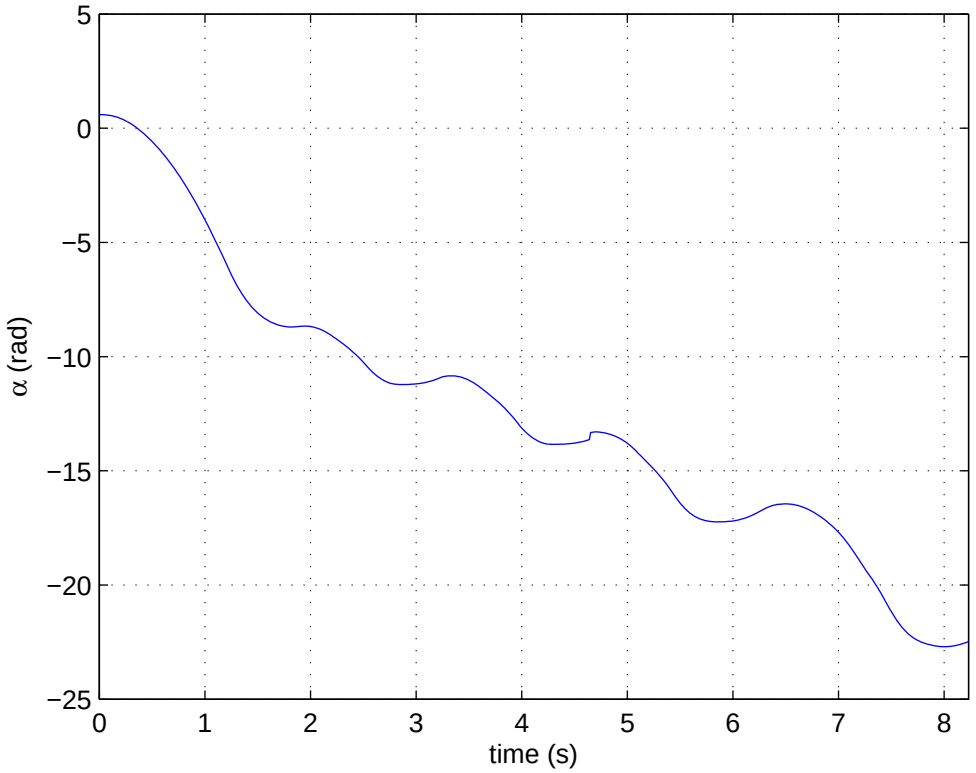
รูปที่ 6.10: การจำลองผลตอบสนองมุมแท่งเพนดูลัมของการควบคุมแท่งเพนดูลัมแบบผกผันที่มีมุมเบี่ยงเบนเริ่มต้น $\pi/18$



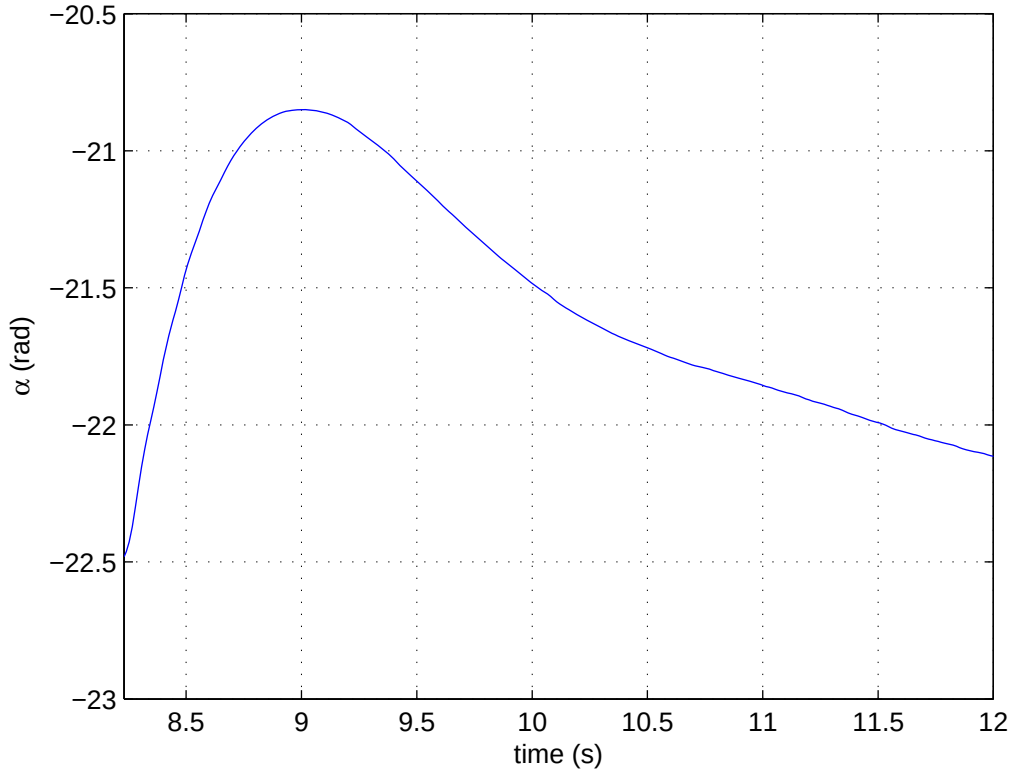
รูปที่ 6.11: ผลตอบสนองมุมแท่งเพนดูลัมช่วงการแกว่งแท่งเพนดูลัมขึ้นด้วยการเพิ่มพลังงานทีละน้อย



รูปที่ 6.12: ผลตอบสนองมุมแท่งเพนดูลัมช่วงการควบคุมแท่งเพนดูลัมขึ้นด้วยอัตราการป้อนกลับที่คำนวณได้



รูปที่ 6.13: ผลตอบสนองมุมของจานหมุนเพนดูลัมช่วงการแกว่งแท่งเพนดูลัมขึ้นด้วยการเพิ่มพลังงานทีละน้อย



รูปที่ 6.14: ผลตอบสนองมุมของจานหมุนเพนดูลัมช่วงการควบคุมแท่งเพนดูลัมขึ้นด้วยอัตราการป้อนกลับที่คำนวณได้

บทที่ 7
บทสรุป

งานวิจัยนี้ได้นำเสนอชุดทดลองเพนดูลัมผกผันคู่แบบหมุน (rotary double inverted pendulum) ซึ่งมีความซับซ้อนและมีความไม่เชิงเส้นสูง (high nonlinearity) เหมาะสำหรับการทดสอบ ทฤษฎีระบบควบคุมชั้นสูง โดยได้นำเสนอส่วนของการสร้างฮาร์ดแวร์ วัสดุ อุปกรณ์ที่ใช้ ซึ่งหาได้ใน ประเทศและมีราคาไม่แพง นอกจากนี้ ยังได้นำเสนอซอฟต์แวร์สำหรับจำลอง พฤติกรรมการทำงานของระบบ ที่มีส่วนติดต่อกับผู้ใช้แบบกราฟิก (graphical user interface) และการออกแบบตัวควบคุมโดยใช้วิธีควบคุมแบบเหมาะสมที่สุด (linear optimal quadratic regulator)

7.1 สิ่งที่ได้ทำแล้ว

- ออกแบบ และทำอุปกรณ์ภายในระบบที่มีเสถียรภาพและเหมาะสมกับการทดลองมากกว่าเดิม และรองรับการประยุกต์ได้หลายแบบ
- การจำลองการควบคุมระบบ โดยใช้การควบคุมแบบ Optimal Control
- ออกแบบระบบควบคุมที่สามารถควบคุมเพนดูลัมผกผันแบบหมุนแท่งเดี่ยวได้ และสามารถแกว่งแท่งเพนดูลัมขึ้นเองได้ด้วยวิธีอย่างง่าย

ในรายงานฉบับนี้มีความก้าวหน้าที่สำคัญ คือ การออกแบบระบบควบคุมของเพนดูลัมผกผันแบบหมุนที่มีความมั่นคง และมีกำลังขับสูง เนื่องจากการเปลี่ยนตัวขับเคลื่อนของระบบมาเป็นมอเตอร์ขนาดใหญ่มีกำลังสูงถึง $3\ \mathrm{N-m}$ นอกจากนั้นยังใช้ตัวควบคุมของระบบเป็นระบบฝังตัว คือเป็นไมโครคอนโทรลเลอร์ ARM7 ซึ่งทำงานได้สูงถึง 60 MHz ระบบที่ออกแบบนี้มีความคล่องตัวของการใช้งานสูง เนื่องจากแยกวงจรเซนเซอร์และวงจรควบคุมออกจากกัน และติดต่อกันผ่านทางสัญญาณอินฟราเรดทำให้ไม่มีปัญหาการบิดของสายสัญญาณจากการหมุนของตัวขับเคลื่อนเพนดูลัมอย่างที่เคยมีในระบบเก่า ชุดทดลองควบคุมนี้สามารถใช้ควบคุมเพนดูลัมแบบผกผันตั้งแต่ 1 แท่งถึง 2 แท่งได้ โดยการโปรแกรมตัวควบคุมที่เหมาะสม และสามารถอ่านค่าสถานะของการควบคุมที่เวลาต่างๆกลับมายังคอมพิวเตอร์ได้ทางพอร์ตอนุกรม RS-232 ระบบเพนดูลัมผกผันแบบหมุนนี้เป็นส่วนหนึ่งในงานจัดแสดงงานจุฬาฯวิชาการครั้งที่ 15 ปี 2551

7.2 สิ่งที่จะต้องทำในอนาคต

หาพารามิเตอร์ของระบบและการออกแบบตัวควบคุมสำหรับเพนดูลัมแบบคู่ เพื่อปรับปรุงประสิทธิภาพของตัวควบคุมให้ดียิ่งขึ้นและพัฒนาคอร์สแวร์ที่มีเนื้อหาครอบคลุมการหา เอกลักษณ์ของระบบ (system identification) การทดสอบแบบจำลอง (model validation) การออกแบบตัวควบคุมชั้นสูง สำหรับใช้ในการฝึกอบรมบุคลากรด้านเทคโนโลยีระบบควบคุมได้ต่อไป

รายการอ้างอิง

[1] S. Mori, H. Nishihara, and K. Furuta. Control of unstable mechanical system control of pendulum. International Journal of Control 23, 5(1976): 673–692.

[2] K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on, pp. 2193–2198 vol.3, 1991.

[3] K. Astrom. Swinging up a pendulum by energy control. Automatica 36, 2February 2000: 287–295.

ภาคผนวก

ภาคผนวก ก

C-ARM Program

ก.1 ไฟล์ ‘Main.c’

```
#include <LPC214x.h>
#include <stdio.h>
#include <stdlib.h>
#include "init.h"
#include "uart/uart.h"
#include "adc/adc.h"
#include "nsk/nsk.h"
#include "uart/command.h"
#include "control/control.h"

#define AD0GSR        (*((volatile unsigned long *) 0xE0034008))

unsigned char init_dis = 0;
unsigned short enc0, temp;
unsigned long enc_m, start_alpha;
unsigned char status = 0x00;
// bit
// 0 - printf enable
// 1 - | 0 = idle,                1 = iden friction plate
// 2 | 2 = iden plate,     3 = iden pendulum
// 3 | 4 = operate                  5 = step response
// 4 | 6 = control crane

extern unsigned long timeval; // encoder position
extern unsigned short enc0_h;
extern signed char vel_ana;  // velocity read form analog channel

void control_loop (void);
void check_sum (char datc);
void read_enc (void);

int main (void)
{
        unsigned char i;
        unsigned char timechar = 0x00;
        unsigned int timetemp = 0x0000;
        unsigned char input;
        signed int preinput, old_enc0, err, enc0d, enc_md = 0;
        signed long old_enc_m = 0, temp;
        unsigned char stop_count;
        signed short regulate_point = 2500;
```

```
        init();

45      clr_rstflp();
        for (i=0;i<255;i++);
        set_rstflp();
        clr_precnt();
        set_rstcnt();
50      for (i=0;i<255;i++);
        clr_rstcnt();

        cnt0_en();
        uart0_com();
55      NSK_init();
        printf(menu);
        upd_dac(0x80);
        while(1){
                // check IR every loop
60              if (rx1_avail()) {
                        check_sum(U1RBR);
                }

                // check for command every loop
65              check_cmd();
                // operate mode
                if ((status & 0x0E) == 0x08){
                        if (timechar == 0x00)
                        {
70                              sendchar('.');
                        }
                }
                // iden friction plate
                if ((status & 0x0E) == 0x02){
75                      if ((timetemp & 0x01F) == 0x00){
                                if (init_dis == 0){
                                        init_iden_plate();
                                        init_dis = 1;
                                }
80                              vel_ana = ((AD0GDR & 0x0000FF00) >> 8);
                                vel_ana -= 128;
                                input = (unsigned char)(rand() >> 23);
                                upd_dac(input);
                                read_enc();
85                              conversion_adc();
                                if ((status & 0x01) != 0)
                                        printf("%d %d\n",input,enc_m);
```

```
                }
        }
90      // step response mode
        if ((status & 0x0E) == 0x0A){
                if ((timetemp & 0x07FF) == 0x0000){
                        input = 0x68;
                        upd_dac(input);
95              }
                if ((timetemp & 0x07FF) == 0x03FF){
                        input = 0xB0;
                        upd_dac(input);
                }
100             if ((timetemp & 0x01F) == 0x00){
                        if (init_dis == 0){
                                init_iden_plate();
                                input = 0x50;
                                upd_dac(input);
105                             init_dis = 1;
                        }
                        vel_ana = ((AD0GDR & 0x0000FF00) >> 8);
                        vel_ana -= 128;
                        read_enc();
110                     conversion_adc();
                        if ((status & 0x01) != 0)
                                printf("%d %d\n",input,enc_m);
                }
        }
115     // control crane mode
        if ((status & 0x0E) == 0x0C){
                if ((timetemp & 0x01F) == 0x00){
                        if (init_dis == 0){
                                read_enc();
120                             init_iden_plate();
                                input = 0x84;
                                upd_dac(input);
                                init_dis = 1;
                                start_alpha = enc_m;
125                             old_enc_m = (signed long)*(&enc_m);
                        }
                        read_enc();
                        temp = (signed long)*(&enc_m);
                        err = -(signed short)*(&enc0);
130                     enc0d = (enc0 - old_enc0);
                        old_enc0 = enc0;
                        enc_md = (signed short)(temp - old_enc_m);
                        old_enc_m = temp;
```

```
                preinput = (signed short)(-( -2*err - 0.25*enc_md + 5*enc0d)/8);

                if (preinput < 0)
                        preinput -= 7;
                else {
                        if (preinput > 0)
                                preinput += 7;
                }

                if (preinput > 127)
                        preinput = 127;
                else {
                        if (preinput < -128)
                                preinput = -128;
                }
                input = (unsigned char)(preinput + 128);
                upd_dac(input);
                if ((status & 0x01) != 0)
                        printf("%d\n",enc0);
                if   (((signed short)*(&enc0) > 1250)
                    || ((signed short)*(&enc0) < -1250)){
                        stop_count++;
                        if (stop_count > 3) {
                                status = 0x00;
                                init_dis = 0;           // initialize parameter
                                for (i=0;i<MAX_CMD;i++)
                                        cmd_line[i] = 0;
                                i = 0;
                                sendchar(0x0D);sendchar(0x0A);sendchar(':');
                                stop_count = 0;
                        }
                }
                else {
                        stop_count = 0;
                }
        }
}
// control pendulum mode
if ((status & 0x0E) == 0x0E){
        if ((timetemp & 0x01F) == 0x00){
                if (init_dis == 0){
                        read_enc();
                        init_iden_plate();
                        input = 0x84;
                        upd_dac(input);
                        init_dis = 1;
```

```
180                     start_alpha = enc_m;
                        old_enc_m = (signed long)*(&enc_m);
                }
                read_enc();
                temp = (signed long)*(&enc_m);
185             err = regulate_point -(signed short)*(&enc0);
                enc0d = (enc0 - old_enc0);
                old_enc0 = enc0;
                enc_md = (signed short)(temp - old_enc_m);
                old_enc_m = temp;
190             // -0.0011    0.7743   -0.1188    9.7468
                //preinput = (signed short)( -0.25*err - 0.03*enc_md + 5*enc0d);
                preinput = (signed short)( -0.27*err - 0.03*enc_md + 6*enc0d);
                //preinput = (signed short)(( -2.5*err - 0.25*enc_md + 15*enc0d)/8);
                //preinput = (signed short)(( -5.2*err + 2*enc_md + 75*enc0d));
195             /*
                if (preinput < 0)
                        preinput -= 7;
                else {
                        if (preinput > 0)
200                             preinput += 7;
                }
                */
                if (preinput > 127)
                        preinput = 127;
205             else {
                        if (preinput < -128)
                                preinput = -128;
                }
                input = (unsigned char)(preinput + 128);
210             upd_dac(input);
                if ((status & 0x01) != 0)
                        printf("%d %d %d\n",preinput,enc0,enc_m);
                if   ((enc0 > 3800) || (enc0 < 1200)){
                        stop_count++;
215                     if (stop_count > 3) {
                                status = 0x00;
                                init_dis = 0;                   // initialize parameter
                                for (i=0;i<MAX_CMD;i++)
                                        cmd_line[i] = 0;
220                             i = 0;
                                sendchar(0x0D);sendchar(0x0A);sendchar(':');
                                stop_count = 0;
                        }
                }
225             else {
```

```
                        stop_count = 0;
                    }
                }
            }
            // unstable pendulum mode
            if ((status & 0x10) == 0x10){
                if ((timetemp & 0x01F) == 0x00){
                    if (init_dis == 0){
                        read_enc();
                        init_iden_plate();
                        input = 0x84;
                        upd_dac(input);
                        init_dis = 1;
                        start_alpha = enc_m;
                        old_enc_m = (signed long)*(&enc_m);
                    }
                    read_enc();
                    enc0d = (enc0 - old_enc0);
                    old_enc0 = enc0;
                    if (((signed short)*(&enc0) < 30)
                        || ((signed short)*(&enc0) > -30))
                    {
                        if (enc0d > 0)
                            preinput = 10;
                        else{
                            if (enc0d < 0)
                                preinput = -10;
                        }

                    }
                    if ((signed short)*(&enc0) > 2400){
                        regulate_point = 2500;
                        status = 0x0F;
                        printf("begin control +\n");
                    }
                    if ((signed short)*(&enc0) < -2400){
                        regulate_point = -2500;
                        status = 0x0E;
                        printf("begin control -\n");
                    }

                    if (preinput > 127)
                        preinput = 127;
                    else {
                        if (preinput < -128)
                            preinput = -128;
```

```
                    }
                    input = (unsigned char)(preinput + 128);
                    upd_dac(input);
                    if ((status & 0x01) != 0)
                        printf("%d %d %d\n",preinput,enc0,enc_m);
                }
            }
            // iden pendulum mode
            if ((status & 0x0E) == 0x06){
                read_enc();
                if ((timetemp & 0x01F) == 0x00){
                    if (init_dis == 0){
                        init_iden_plate();
                        input = 0x40;
                        upd_dac(input);
                        init_dis++;
                    }
                    else {
                        if (init_dis < 6)
                            init_dis++;
                        else {
                            input = 0x80;
                            upd_dac(input);
                        }
                    }
                    vel_ana = ((AD0GDR & 0x0000FF00) >> 8);
                    vel_ana -= 128;
                    read_enc();
                    conversion_adc();
                    if ((status & 0x01) != 0)
                        printf("%d %d %.8x\n",input,enc0,enc_m);
                }
            }
            if (status == 0x00)
            {
                upd_dac(128);
            }
            /*
            // operate mode
            if ((status & 0x06) == 0x06){
                // Do control once in 4 loop
                if (timechar == 0x00)
                {
                    set_led();
                    read_enc();
                    vel_ana = (AD0GDR & 0x0000FF00) >> 8;
```

```
                        control_loop();
                        conversion_adc();
320                     if ((status & 0x01) != 0){
                                printf(``%X %X %X\n'',enc0,enc_m,vel_ana);
                        }
                }
        }
325     */
        if (timechar == 0x00)
                set_led();
        if (timechar == 0x02)
                clr_led();
330     while(timeval == timetemp);
        timetemp = timeval;
        timechar = timetemp & 0x03;
    }
}
335
void check_sum (char datc)
{
    static char* p;
    static unsigned char intcnt = 0;
340 if (intcnt == 0)
    {
        if (datc == 0x55)
        {
            p = (char *)(&temp);
345         p++;
            intcnt++;
        }
    }
    else
350 {
        if (intcnt < 3) // store data
        {
            *(p) = datc;
            sendchar1(datc);
355         sendchar1(' ');
            p--;
            intcnt++;
        }
        else
360     {
            if (((*(p+2)) ^ (*(p+1))) == datc) {
                enc0 = temp;
                //printf(``%X\n'',enc0);
```

```
                        }
                        intcnt = 0;
                }
        }
}

void read_enc (void)
{
        short* p;
        p = (short *)(&enc_m);
        enc_m = FIO1PIN >> 16;
        *(p+1) = enc0_h;
}

void control_loop (void)
{
        upd_dac(0x88);
}
```

ก.2 ไฟล์ 'Init.c'

```
#include <LPC214x.h>
#include "init.h"
#include "timer/timer.h"
#include "pwm/pwm.h"
#include "ext/ext.h"
#include "uart/uart.h"
#include "adc/adc.h"

void PLL_init (void)          // OSC = 11.0592 (change to 55.296)
{
        PLL0CFG = 0x24;             // M = 5, CCLK = 55.296
                                            // P = 3, 156 < Fcco = 55.296*2*2 = 221.184 < 320
        PLL0CON = 0x01;             // Lock Frequency by enable PLLE
        PLL0FEED = 0xAA;
        PLL0FEED = 0x55;
        while((PLL0STAT & 0x0400) == 0x00); // Check bit 10 (Lock?)
        PLL0CON = 0x03;             // Connect to CCLK
        PLL0FEED = 0xAA;
        PLL0FEED = 0x55;
        VPBDIV = 0x02;              // PCLK = 1/2 of CCLK (27.648)
}

void init_io (void)
{
        SCS = 0x00000003;           // Enable Fast GPIO
        // P0.0, P0.1 = UART0
        PINSEL0 &= 0xFFFFFFF0;
        PINSEL0 |= 0x00000005;
        // P0.5 = MAT0.1
        PINSEL0 &= 0xFFFFF3FF;
        PINSEL0 |= 0x00000800;
        // P0.7 = EINT2
        PINSEL0 &= 0xFFFF3FFF;
        PINSEL0 |= 0x0000C000;
        // P0.8, P0.9 = UART1
        PINSEL0 &= 0xFFF0FFFF;
        PINSEL0 |= 0x00050000;
        // P0.15 = EINT2
        PINSEL0 &= 0x3FFFFFFF;  // Note that in edge-detecting mode, we cannot use multiple EINT pin.
        //PINSEL0 |= 0x80000000;  // Then, we disable EINT2 on P0.15.
        // P0.25 = AOUT
        PINSEL1 &= 0xFFF3FFFF;
        PINSEL1 |= 0x00080000;
        // P0.28 = AD0.1
        PINSEL1 &= 0xFCFFFFFF;
```

```
        PINSEL1 |= 0x01000000;
        // set IO direction output for P0.2,3,4,6,16,17,22,29,30,31
        FIO0DIR = 0xE043005C;
        FIO1DIR = 0x00000000;
        PINSEL2 = 0x00000000;
}

void init (void)
{
        PLL_init();
        // pre-relay
        open_relay();
        init_io();
        init_timer();
        init_pwm();                    // init value is 50% at 864k Hz
        init_eint();
        init_uart0();
        init_uart1();
        init_adc();
        init_dac();
}
```

ก.3 ไฟล์ 'Init.h'

```
#ifndef _INIT_H_
#define _INIT_H_

// Port1 = Data From Encoder
// Counter and Flip-Flip control pin
#define clr_rstflp() FIO0CLR = 0x00400000     // clear P0.22
#define set_rstflp() FIO0SET = 0x00400000     // set P0.22
#define clr_precnt() FIO0CLR = 0x00000004     // clear P0.2
#define set_precnt() FIO0SET = 0x00000004     // set P0.2
#define clr_rstcnt() FIO0CLR = 0x00000008     // clear P0.3
#define set_rstcnt() FIO0SET = 0x00000008     // set P0.3

#define cnt0_en()     FIO0CLR = 0x00000010 // clear P0.4 : Counter output enable
#define cnt_dis()     FIO0SET = 0x00000010 // set P0.4

#define clr_cw()      FIO0CLR = 0x00010000 // clear P0.16 : CW
#define set_cw()      FIO0SET = 0x00010000 // set P0.16

#define clr_ccw()     FIO0CLR = 0x00020000 // clear P0.17 : CCW
#define set_ccw()     FIO0SET = 0x00020000 // set P0.17

#define clr_svon()    FIO0CLR = 0x40000000 // clear P0.30 : SVON
#define set_svon() FIO0SET = 0x40000000     // set P0.30

#define open_relay() FIO0SET = 0x20000000 // clear P0.29 : p-transistor gate close, -12V to N-fet gate close
#define close_relay() FIO0CLR = 0x20000000 // (close-circuit)

#define clr_led()   FIO0CLR = 0x80000000      // clear P0.31 : Auxilary LED
#define set_led()   FIO0SET = 0x80000000      // set P0.31

#define read_carry() (IO0PIN & 0x00000080) == 0x00000080 //: P0.7
#define read_dir()       (FIO0PIN & 0x00000400) == 0x00000400 //: P0.10
#define read_z()     (IO0PIN & 0x00002000) == 0x00008000 //: P0.13

void init (void);

#endif
```

ก.4 ไฟล์ '/uart/uart.c'

```
#include <LPC214X.h>
#include <STDIO.H>
#include "uart.h"
#include "command.h"

extern unsigned char init_dis;
extern unsigned char status;

void init_uart0 (void)
{
	U0IER = 0x00;			// disable transmit and receive interrupt
	U0LCR = 0x03;			// set length to 8 bit
	U0LCR |= 0x80;
	U0DLM = 0x00;
	U0DLL = 0x0F;				// not use U1FDR (115200)
	U0LCR &= 0x07;
	U0TER = 0x80;			// transmit enable
	U0FCR = 0x07;			// enable FIFO, TX FIFO Reset, RX FIFO Reset
}

void uart0_115200 (void)
{
	U0LCR |= 0x80;
	U0DLL = 0x0F;			// Not devided by 12 (115200)
	U0LCR &= 0x07;
}

void uart0_9600 (void)
{
	U0LCR |= 0x80;
	U0DLL = 0xB4;			// devided by 12 (9600)
	U0LCR &= 0x07;
}

void init_uart1 (void)
{
	U1IER = 0x00;			// disable transmit and receive interrupt
	U1LCR |= 0x80;
	U1DLM = 0x00;
	U1DLL = 0x1E;				// not use U1FDR (57600)
	//U1DLL = 0x5A;				// not use U1FDR (19200)
	U1LCR &= 0x07;
	U1LCR = 0x03;			// set length to 8 bit
	U1TER = 0x80;			// transmit enable
	U1FCR = 0x07;			// enable FIFO, TX FIFO Reset, RX FIFO Reset
```

```
}

int sendchar (int ch)
{
  if (ch == '\n')
  {
    while ((U0LSR & 0x20) == 0x00);                    // Wait TXD Buffer Empty
    U0THR = CR;                                        // Write CR
  }
  while ((U0LSR & 0x20) == 0x00);                      // Wait TXD Buffer Empty
  return (U0THR = ch);                                 // Write Character
}

int getkey (void)
{
  while ((U0LSR & 0x01) == 0x00);                      // Wait RXD Receive Data Ready
  return (U0RBR);                                      // Get Receice Data & Return
}

int sendchar1 (int ch)
{
  if (ch == '\n')
  {
    while ((U1LSR & 0x20) == 0x00);                    // Wait TXD Buffer Empty
    U1THR = CR;                                        // Write CR
  }
  while ((U1LSR & 0x20) == 0x00);                      // Wait TXD Buffer Empty
  return (U1THR = ch);                                 // Write Character
}

int getkey1 (void)
{
  while ((U1LSR & 0x01) == 0x00);                      // Wait RXD Receive Data Ready
  return (U1RBR);                                      // Get Receice Data & Return
}

int readint1 (void)
{
        int dat;
        char* p;
        p = (char *)(&dat);
        *(p+1) = getkey1();
        *(p) = getkey1();
        return(dat);
}
```

```
void intwrite1 (int dat)
{
        char* p;
        p = (char *)(&dat);
        sendchar1(*(p+1));
        sendchar1(*(p));
}

void printline (unsigned char *p, unsigned char length){
        unsigned char i = 0;
        unsigned char temp;
        do{
                temp = *(p+i);
                sendchar(temp);
                i++;
        } while ( (i < length) && (temp != LF) );
}

void readline (unsigned char *p, unsigned char length){
        unsigned char i = 0;
        unsigned char temp;
        do{
                temp = getkey();
                *(p+i) = temp;
                i++;
        } while ( (i < length) && (temp != LF) );
}

unsigned char uart0_com (void) {
        unsigned char temp;
        //waiting for the shift register is empty
        while((U0LSR & 0x40) == 0);
        //clear all buffer
        while((U0LSR & 0x01) != 0){
                temp = U0RBR;
        }
        // select port connect to com by set P0.6
        FIO0SET = 0x00000040;
        uart0_115200();
        return(temp);
}

unsigned char uart0_drv (void){
        unsigned char temp;
        //waiting for the shift register is empty
        while((U0LSR & 0x40) == 0);
```

```
        //clear all buffer
        while((U0LSR & 0x01) != 0){
140             temp = U0RBR;
        }
        // select port connect to drv by clr P0.6
        FIO0CLR = 0x00000040;
        uart0_9600();
145     return(temp);
}

void check_cmd (void) {
        static unsigned char esc_line;
150     unsigned char temp;
        static unsigned char i;
        // check if data available
        if ((U0LSR & 0x01) != 0){
                temp = U0RBR;
155             if (temp == 0x0D){              // Enter
                        if (i!=0){
                                process_cmd(cmd_line);
                        }
                        i = 0;
160                     printf("\n:");
                }else{
                        if (i < MAX_CMD){
                                if (temp == 0x1B){          // Esc up, dwn, rgt, lft
                                        if (esc_line == 1){
165                                             // return to idle mode
                                                status = 0;
                                                esc_line = 0;
                                                init_dis = 0;                   // initialize parameter
                                                for (i=0;i<MAX_CMD;i++)
170                                                     cmd_line[i] = 0;
                                                i = 0;
                                                sendchar(0x0D);sendchar(0x0A);sendchar(':');
                                        }else{
                                                esc_line = 1;
175                                     }
                                } else {
                                        if ((temp == 0x08)){ // Back Space
                                                if (i != 0){
                                                        i--;
180                                                     cmd_line[i] = 0;
                                                        sendchar(0x08);sendchar(0x20);sendchar(0x08);
                                                }
                                        } else {
```

```
                                        if (esc_line == 1){
185                                             if (temp == 0x5B)
                                                        esc_line = 2;
                                                else
                                                        esc_line = 0;
                                        } else {
190                                             if (esc_line == 2){
                                                        switch (temp) {
                                                                case 'C':
                                                                        if (i != (MAX_CMD-1)){
                                                                                if (cmd_line[i] == 0)
195                                                                                     cmd_line[i] = 0x20;
                                                                                i++;
                                                                                sendchar(0x1B);sendchar(0x5B);sendchar('C');
                                                                        }
                                                                        break;
200                                                             case 'D':
                                                                        if (i != 0){
                                                                                if (cmd_line[i] == 0)
                                                                                        cmd_line[i] = 0x20;
                                                                                i--;
205                                                                             sendchar(0x1B);sendchar(0x5B);sendchar('D');
                                                                        }
                                                                        break;
                                                        }
                                                        esc_line = 0;
210                                             } else {
                                                        cmd_line[i] = temp;
                                                        sendchar(temp);
                                                        i++;
                                                }
215                                     }
                                }
                        }
                } else {
                        printf("\ncommand too long\n:");
220                     for (i=0;i<MAX_CMD;i++)
                                cmd_line[i] = 0;
                        i = 0;
                }
        }
225 }

void process_cmd (unsigned char *p){
        unsigned char temp,i,j;
```

```
230         unsigned char k[CMD_CNT];
            for (i=0;i<CMD_CNT;i++)
                    k[i] = i;
            for (i=0;i<MAX_CMD;i++){
                    temp = *(p+i);
235                 *(p+i) = 0;
                    if (temp == 0)
                            temp = SPC;
                    for (j=0;j<CMD_CNT;j++){
                            if (!((temp == cmd_table[(j<<3)+i])||(temp == (cmd_table[(j<<3)+i]+0x20))))
240                                 k[j] = 0xFF;
                    }
                    if (temp == SPC)
                            break;
            }
245         temp = num_cmd(k);
            switch (temp){
                    case 0:
                            command_0 (cmd_line,i);
                            break;
250                 case 1:
                            printf("\n");
                            printf("2nd cmd\n");
                            break;
                    case 2:
255                         printf("\n");
                            printf("3rd cmd\n");
                            break;
                    case 3:
                            command_3 (cmd_line,i);
260                         break;
                    case 4:
                            command_4 (cmd_line,i);
                            break;
                    case 255:
265                         printf("\n");
                            printf("unknown command\n");
                            break;
            }
    }
270
    unsigned char num_cmd (unsigned char *p){
            unsigned char i;
            unsigned char temp;
            temp = 0xFF;
275         for (i=0;i<CMD_CNT;i++){
```

```
            if (*(p+i) != 0xFF)
                temp = *(p+i);
        }
        return temp;
280 }
```

ก.5 ไฟล์ '/uart/uart.h'

```
#ifndef _UART_H_
#define _UART_H_

#define CR 0x0D       // Carriage Return
#define LF 0x0A       // Line Feed
#define SPC 0x20      // Space

#define wait_uart1() while((U1LSR & 0x01) == 0x00)
#define wait_uart0() while((U0LSR & 0x01) == 0x00)
#define rx1_avail() (U1LSR & 0x01) == 0x01

extern const char menu[];

void init_uart0 (void);
void uart0_115200 (void);
void uart0_9600 (void);
void init_uart1 (void);
int sendchar (int ch);
int getkey (void);
int sendchar1 (int ch);
int getkey1 (void);
int readint1 (void);
void intwrite1 (int dat);
void printline (unsigned char *p, unsigned char length);
void readline (unsigned char *p, unsigned char length);
unsigned char uart0_com (void);
unsigned char uart0_drv (void);
void check_cmd (void);
void process_cmd (unsigned char *p);
unsigned char num_cmd (unsigned char *p);

#endif
```

ก.6 ไฟล์ '/uart/command.c'

```
#include <LPC214X.H>
#include <STDIO.H>
#include "command.h"
#include "uart.h"

extern unsigned char status;

unsigned char cmd_line[MAX_CMD] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

const char menu[] =
  "\n"
  "+-----------------------------------------------------------------+\n"
  "| This program is used for oprating pendulum           |\n"
  "+ command -+ --- syntax ----+--------------- function ------------+\n"
  "| Print   | P <Y/N>      | print data enable/disable     |\n"
  "| Set     | S <name> <val> | set parameter name and its value |\n"
  "| Get     | G <name> <val> | set parameter name and its value |\n"
  "| DRV     | D <cmd>      | send command directly to driver |\n"
  "| MODE    | M <0,1,2,3,4> | 0 idle,  1 iden friction plate, |\n"
  "|         |  <5,6,7>     | 2 iden plate,3 iden pendulum, |\n"
  "|         |              | 4 operate, 5 step response,   |\n"
  "|         |              | 6 control crane, 7 control pendulum |\n"
  "+----------+----------------+------------------------------------+\n\n:";

const char cmd_table[] =   "PRINT  "
                           "SET    "
                           "GET    "
                           "DRV    "
                           "MODE   ";

unsigned char command_0 (unsigned char *p,unsigned char j){
        unsigned char temp,tempp,i;
        do {
                temp = *(p+j);
                *(p+j) = 0;
                j++;
        }while((temp == 0)||(temp == SPC));

        for (i=j;i<MAX_CMD;i++){
                tempp = *(p+i);
                *(p+i) = 0;
                if (!((tempp == 0) || (tempp == SPC))) {
                        temp = 0xFF;
                }
        }
```

```
        switch (temp) {
                case 'Y': case 'y':
                        status |= 0x01;
                        printf(" > enabled\n");
                        return 1;
                case 'N': case 'n':
                        status &= ~0x01;
                        printf(" > disabled\n");
                        return 1;
                default :
                        printf("\nPRINT : invalid param, only Y/N is allowed\n");
                        return 0;
        }
}

unsigned char command_3 (unsigned char *p,unsigned char j){
        unsigned char temp,k;
        unsigned int i;

        uart0_drv();

        do {
                temp = *(p+j);
                *(p+j) = 0;
                j++;
        }while((temp == 0)||(temp == SPC));
        if (temp > 0x60)
                temp -= 0x20;
        sendchar(temp);
        getkey();

        for (i=j;i<MAX_CMD;i++){
                temp = *(p+i);
                if (!((temp == 0) || (temp == SPC))) {
                        k = i;
                }
        }

        for (i=j;i<(k+1);i++){
                temp = *(p+i);
                *(p+i) = 0;
                if (temp > 0x60)
                        temp -= 0x20;
                sendchar(temp);
                getkey();
        }
```

```
        sendchar(0x0D);
        i = 0;
        do{
                temp = getkey();
                i++;
        }while ((temp != ':') && (i < 1000));

        if (i == 1000){
                i = 0;
                sendchar(0x08);
                do{
                        temp = getkey();
                        i++;
                }while ((temp != ':') && (i < 1000));
        }
        for (i = 0; i < 65535; i++);
        uart0_com();
        if (i == 1000){
                printf(" : fail\n");
                return 0;
        } else {
                printf(" : succeed\n");
                return 1;
        }
}

unsigned char command_4 (unsigned char *p,unsigned char j){
        unsigned char temp,tempp,i;
        do {
                temp = *(p+j);
                *(p+j) = 0;
                j++;
        }while((temp == 0)||(temp == SPC));

        for (i=j;i<MAX_CMD;i++){
                tempp = *(p+i);
                *(p+i) = 0;
                if (!((tempp == 0) || (tempp == SPC))) {
                        temp = 0xFF;
                }
        }

        switch (temp) {
                case '0':
                        status &= 0xF1;
                        printf(" : idle\n");
```

```
                return 1;
            case '1':
                status &= 0xF1;
                status |= 1 << 1;
                printf(" : iden friction plate\n");
                return 1;
            case '2':
                status &= 0xF1;
                status |= 2 << 1;
                printf(" : iden plate\n");
                return 1;
            case '3':
                status &= 0xF1;
                status |= 3 << 1;
                printf(" : iden pendulum\n");
                return 1;
            case '4':
                status &= 0xF1;
                status |= 4 << 1;
                printf(" : operate\n");
                return 1;
            case '5':
                status &= 0xF1;
                status |= 5 << 1;
                printf(" : step response\n");
                return 1;
            case '6':
                status &= 0xF1;
                status |= 6 << 1;
                printf(" : control crane\n");
                return 1;
            case '7':
                status &= 0xF1;
                status |= 7 << 1;
                printf(" : control pendulum\n");
                return 1;
            case '8':
                status &= 0xF1;
                status |= 8 << 1;
                printf(" : unstable pendulum\n");
                return 1;
            default :
                printf("\nMODE : invalid mode\n");
                return 0;
        }
    }
```

ก.7 ไฟล์ '/uart/command.h'

```
#ifndef _COMMAND_H_
#define _COMMAND_H_

#define MAX_CMD 64
#define CMD_Len 8
#define CMD_CNT 5

extern unsigned char cmd_line[];
extern const char menu[];
extern const char cmd_table[];

unsigned char command_0 (unsigned char *p,unsigned char j);
unsigned char command_3 (unsigned char *p,unsigned char j);
unsigned char command_4 (unsigned char *p,unsigned char j);

#endif
```

ก.8 ไฟล์ '/timer/timer.c'

```
#include <LPC214x.h>                    // LPC21XX Peripheral Registers
#include "timer.h"

unsigned long timeval;

/* Timer Counter 0 Interrupt executes each 10ms */
void tc0 (void) __irq
{
  ++timeval;
  T0EMR &= 0xFFF0;                                  // Clear MAT0.n on T0MR0 matched
  T0IR       = 1;                 // Clear interrupt flag
  VICVectAddr = 0;                // Acknowledge Interrupt
}

/* Setup the Timer Counter 0 Interrupt */
void init_timer (void)
{
  T0MR0 = 11519;                   // 2400Hz = 11520 - 1 counts
  T0MR1 = 10000;                   // same as MR0 used in adc.
  T0EMR = 0x0080;                                   // enable to set high ON MAT0.1 when MR1 match
  T0MCR = 3;                      // Interrupt and Reset on MR0
  T0TCR = 1;                      // Timer0 Enable
  VICVectAddr0 = (unsigned long)tc0;  // set interrupt vector in 0
  VICVectCntl0 = 0x20 | 4;        // use it for Timer 0 Interrupt
  VICIntEnable |= 0x00000010;     // Enable Timer0 Interrupt
}
```

ก.9 ไฟล์ ‘/timer/timer.h’

```
#ifndef _TIMER_H_
#define _TIMER_H_

void init_timer (void);

#endif
```

ก.10 ไฟล์ '/pwm/pwm.c'

```
#include <LPC214x.h>

void init_pwm (void)
{
    PINSEL1 |= 0x00000400;          // select P0.21 as a PWM5 output
    PWMPCR = 0x2000;                // enable PWM5 and select single edge control for PWM5
    PWMTC = 0x00000000;             // count register set = 0 at the begining
    PWMPR = 0x00000000;             // count every (n+1)*PCLK cycles (every PCLK cycle)
    PWMMCR = 0x00000002;            // set PWMTC to reset if PWMMR0 is matched
    PWMMR0 = 0x00000008;            // set period of the PWM to 128 count of PWMTC
    PWMMR5 = 0x00000004;            // set low-time of the PWM to 64 count of PWMTC
    PWMLER = 0x21;                  // update changing of PWMMR5 and PWMMR0
    PWMTCR = 0x09;                  // Enable PWM and run counter
}

void change_duty_cycle (unsigned char DC)
{
    PWMMR5 = (DC >> 1);             // devide value by 2
    PWMLER = 0x20;                  // update changing of PWMMR5
}
```

ก.11 ไฟล์ '/pwm/pwm.h'

```
#ifndef _PWM_H_
#define _PWM_H_

void init_pwm (void);
void change_duty_cycle (unsigned char DC);

#endif
```

ก.12 ไฟล์ '/NSK/nsk.c'

```
#include <LPC214X.H>
#include <STDIO.H>
#include "../uart/uart.h"
#include "../init.h"

unsigned char LINE_TEMP_1[16];
unsigned char LINE_TEMP_2[16];

void NSK_init(void){
	unsigned long i;
	unsigned char INIT_NAME[16] = {'N','S','K',' ','M','E','G','A','T','O','R','Q','U','E',CR,LF};
	unsigned char temp;

	printf("NSK-ESA25 INIT : \n");
	uart0_drv();

	// control relay of ESA25 here
	// delay
	for (i=1;i<27648000;i++);
	close_relay();
	// wait for NSK init value
	while(1){
		temp = getkey();
		if (temp == SPC){
			set_led();
			for (i=0;i<16;i++){
				temp = getkey();
				if (temp != INIT_NAME[i]){
					break;
				}
			}
			if (i == 16) {
				readline(LINE_TEMP_1,16);
				readline(LINE_TEMP_2,16);
				temp = getkey();
				if (temp == ':') {
					break;
				}
			}

		}
	}
	clr_led();
	temp = i;

```

```
        uart0_com();
        printline(LINE_TEMP_1,16);
        printline(LINE_TEMP_2,16);
        printf("ESA Init : Succeed %d\n",temp);
50      sendchar(':');
        set_svon();
    }
```

ก.13 ไฟล์ ‘/NSK/nsk.h’

```
#ifndef _NSK_H_
#define _NSK_H_

void NSK_init(void);

#endif
```

ก.14 ไฟล์ '/ext/ext.c'

```
#include <LPC214x.h>
#include "../init.h"

unsigned short enc0_h = 0;
signed long abs_pos = 0;

void ext2 (void) __irq
{
        unsigned char current_dir;
        current_dir = read_dir();
                if(current_dir)
                        enc0_h++;
                else
                        enc0_h--;
        EXTINT     = 0x04;                  // Clear interrupt flag by writing '1' to it
        VICVectAddr = 0;                    // Acknowledge Interrupt
}

void init_eint (void)
{
        EXTMODE = 0x04;                                 // set to edge-sensitive
        EXTPOLAR = 0x04;                                // set to rising edge
        VICVectAddr1 = (unsigned long)ext2;  // set Interrupt slot1 to ext2 function
        VICVectCntl1 = 0x20 | 16;                       // enable slot and be choosen for EINT2
        VICIntEnable |= 0x00010000;                     // enable EINT2 interrupt
}
```

ก.15 ไฟล์ ‘/ext/ext.h’

```
#ifndef _EXT_H_
#define _EXT_H_

void init_eint (void);

#endif
```

ก.16 ไฟล์ ‘/control/control.c’

```
#include <LPC214X.H>
#include <STDIO.H>
#include "../uart/uart.h"

void init_iden_plate (void){
        uart0_drv();
        printf("/NSK ON\n");
        printf("FR0\n");              // set Feedback Resolution to 10 bit
        uart0_com();
}
```

ก.17 ไฟล์ ‘/control/control.h’

```
#ifndef _CONTROL_H_
#define _CONTROL_H_

void init_iden_plate (void);

#endif
```

ก.18 ไฟล์ '/adc/adc.c'

```
#include <LPC214x.h>
#include "adc.h"

signed char vel_ana;

void adc_routine (void) __irq
{
        vel_ana = (AD0GDR & 0x0000FF00) >> 8;
        VICVectAddr = 0;
}

void init_adc (void)
{
        AD0CR = 0x00241102;                                     // Select AD0.1.
        // Converting freq. is devided by (17+1).
        // Non-Burst mode, (8-bit accuracy if burst) (but now is 10-bit).
        // Operation.
        // Convert on MAT0.1 and rising edge.

        //AD0INTEN = 0x00000002;                                // enable channel 1 interrupt
        //VICVectAddr2 = (unsigned long)adc_routine;  // set Interrupt slot1 to adc_routine function
        //VICVectCntl2 = 0x20 | 18;                             // enable slot and be choosen for AD0
        //VICIntEnable |= 0x00040000;                           // enable AD0 interrupt
}

void convert_adc (void)
{
        AD0CR |= 0x01000000;                                    // start conversion
}

void upd_dac (unsigned char val)
{
        DACR_DATA = val;                                        // see definition of DACR_DATA in adc.h
}

void init_dac (void)
{
        DACR |= 0x00010000;                                     // enable maximum power mode
}
```

ก.19 ไฟล์ ‘/adc/adc.h’

```
#ifndef _ADC_H_
#define _ADC_H_

#define DACR_DATA       (*((volatile unsigned char *) 0xE006C001))

#define conversion_adc()    AD0CR |= 0x01000000

void init_adc (void);
void convert_adc (void);
void init_dac (void);
void upd_dac (unsigned char val);

#endif
```