รถอินเวิร์ทเพนดูลัม

INVERTED PENDULUM ON CART

โดย

นายทรงกรด ธิราชัย
นางสาวทัตยา ปุคคละนันทน์
นายนพดล จันทร์คุณาภาส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมระบบควบคุม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2541

รถอินเวิร์ทเพนดูลัม

INVERTED PENDULUM ON CART

โดย

นายทรงกรด	ธิราชัย	38014171
นางสาวทัตยา	ปุคคละนันทน์	38014176
นายนพดล	จันทร์คุณาภาส	38014221

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. จงกล งามวิวิทย์

อาจารย์สุมิตร พนาอุดมทรัพย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง รถอินเวิร์ทเพนดูลัม

INVERTED PENDULUM ON CART

โดย	นายทรงกรด	ธิราชัย	38014171
	นางสาวทัตยา	ปุคคละนันทน์	38014176
	นายนพดล	จันทร์คุณาภาส	38014221

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. จงกล งามวิวิทย์)

.....อาจารย์ที่ปรึกษา

(อาจารย์สุมิตร พนาอุดมทรัพย์)

รถอินเวิร์ทเพนดูลัม

INVERTED PENDULUM ON CART

โดย

นายทรงกรด ธิราชัย รหัสประจำตัว 38014171

นางสาวทัตยา ปุคคลละนันทน์ รหัสประจำตัว 38014176

นายนพดล จันทร์คุณาภาส รหัสประจำตัว 38014221

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. จงกล งามวิวิทย์

อาจารย์สุมิตร พนาอุดมทรัพย์

บทคัดย่อ

รถอินเวิร์ทเพนดูลัมเป็นระบบที่ไม่มีเสถียรภาพในตัวเอง ถูกออกแบบเพื่อควบคุมให้ก้านเพนดูลัมบนรถมีตำแหน่งในแนวตั้งฉากกับพื้นตลอดเวลาถึงแม้จะได้รับการรบกวนจากภายนอกก็ตาม ตัวรถทำงานโดยใช้หลักการควบคุมโดยอาศัยสัญญาณป้อนกลับจากอุปกรณ์ตรวจจับ คือ โพเทนทิโอมิเตอร์ และเอ็นโคดเดอร์ ส่งไปยังคอมพิวเตอร์เพื่อทำการคำนวณการเคลื่อนที่ของรถและส่งสัญญาณออกไปควบคุมการเคลื่อนที่ของตัวรถต่อไป

ABSTRACT

The Inverted Pendulum on Cart is a naturally unstable system. Therefore, the goal of this project is to design and implement a state feedback controller which can keep the pendulum in the upright position even in the presence of disturbance. The control signal is generated to drive a cart by the computer's computation based on the designed control law using the measured outputs through sensor, potentiometer and encoder.

สารบัญ

สารบัญรูปภาพ

หน้า

หน้า

บทที่ 1
บทนำ

วัตถุประสงค์ของโครงงาน คือ

1. เพื่อศึกษาและสร้างต้นแบบรถอินเวิร์ทเพนดูลัม
2. ศึกษาทฤษฎีการควบคุมเสถียรภาพ และทดลองใช้ด้วยวิธีสเตทสเปซ
3. นำความรู้ที่ได้รับจากการศึกษาและทดลองในห้องปฏิบัติการ ไปเป็นพื้นฐานในการจัดทำและประยุกต์ใช้ในโครงงาน
4. ศึกษาถึงปัญหาที่เกิดขึ้นในโครงงาน และแนวทางแก้ไข รวมถึงการพัฒนาระบบต่อไป

การควบคุมเสถียรภาพของรถอินเวิร์ทเพนดูลัมในโครงงานนี้ ถูกออกแบบและสร้างให้มีการทำงานโดยอาศัยหลักการของระบบควบคุมที่มีสัญญาณป้อนกลับ

การทำงานของรถอินเวิร์ทเพนดูลัม คือ ตัวรถจะพยายามเลี้ยงแท่งอลูมิเนียมที่มีลูกตุ้มถ่วงอยู่ด้านบนให้แท่งอลูมิเนียมตั้งตรงตลอดเวลา โดยแท่งอลูมิเนียมจะเคลื่อนที่ได้ในทิศทางเดียวตามแนวยาวของรถ และตัวรถเคลื่อนที่ได้แกนเดียว คือเคลื่อนที่ไปข้างหน้าหรือเคลื่อนที่ถอยหลังเท่านั้น โดยจะมีตัวตรวจจับการเปลี่ยนแปลงองศาของแท่งอลูมิเนียม ว่าเบี่ยงเบนไปจากแนวตั้งฉากกับพื้นไปเป็นมุมเท่าไร และนำค่าที่ได้ไปแปลงเป็นสัญญาณดิจิตอล แล้วส่งให้ซีพียูประมวลผล และส่งสัญญาณไปควบคุมการหมุนของมอเตอร์ผ่านวงจรขับมอเตอร์ว่าจะให้รถเคลื่อนที่ไปในทิศทางใด

ในโครงงานนี้จะใช้คอมพิวเตอร์ในการประมวลผลและควบคุมการทำงานของรถ พร้อมแสดงผลตอบสนองของการเคลื่อนที่ของก้านเพนดูลัมและตัวรถ แล้วพล็อตกราฟผ่านทางหน้าจอคอมพิวเตอร์

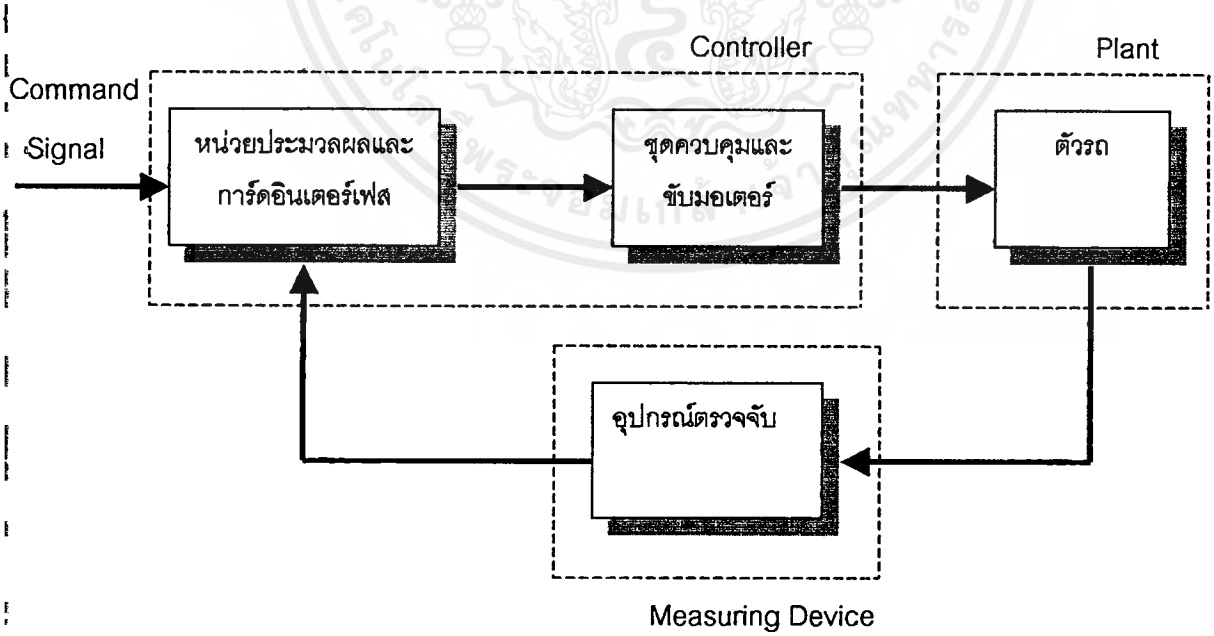
บทที่ 2
ทฤษฎีและหลักการ

2.1 หลักการในการควบคุม

ในการควบคุมระบบการเคลื่อนที่ของรถอินเวิร์ทเพนดูลัมให้ได้ตามต้องการนั้น การควบคุมที่เป็นสิ่งสำคัญที่สุด คือ การควบคุมการหมุนของมอเตอร์ซึ่งเป็นตัวการสำคัญที่ทำให้รถเคลื่อนที่ไปด้วยความเร็วและทิศทางที่ถูกต้องและสัมพันธ์กับทิศทางการแกว่งตัวของก้านเพนดูลัมเพื่อให้ก้านเพนดูลัมอยู่ในตำแหน่งตั้งฉากตลอดเวลา

การควบคุมการทำงานของมอเตอร์เป็นการควบคุมแบบดิจิตอล (Digital Control) ซึ่งได้จากการทำงานร่วมกันของ คอมพิวเตอร์ ชุดควบคุม และเซ็นเซอร์ (Sensor) โดยเซ็นเซอร์จะทำการวัดสัญญาณป้อนกลับ (Feedback Signal) ค่าตำแหน่งของรถ และค่ามุมการแกว่งของก้านอินเวิร์ทเพนดูลัม แล้วนำค่าที่ได้มาคำนวณหาค่าสเตท (State) ต่าง ๆ และค่าสัญญาณควบคุม (Conrtol Signal) หรือสัญญาณความเร็วอ้างอิง (Velocity Reference Command) เพื่อส่งไปยังชุดควบคุมการหมุนของมอเตอร์ต่อไป

บล็อกไดอะแกรมของระบบแสดงได้ดังรูป

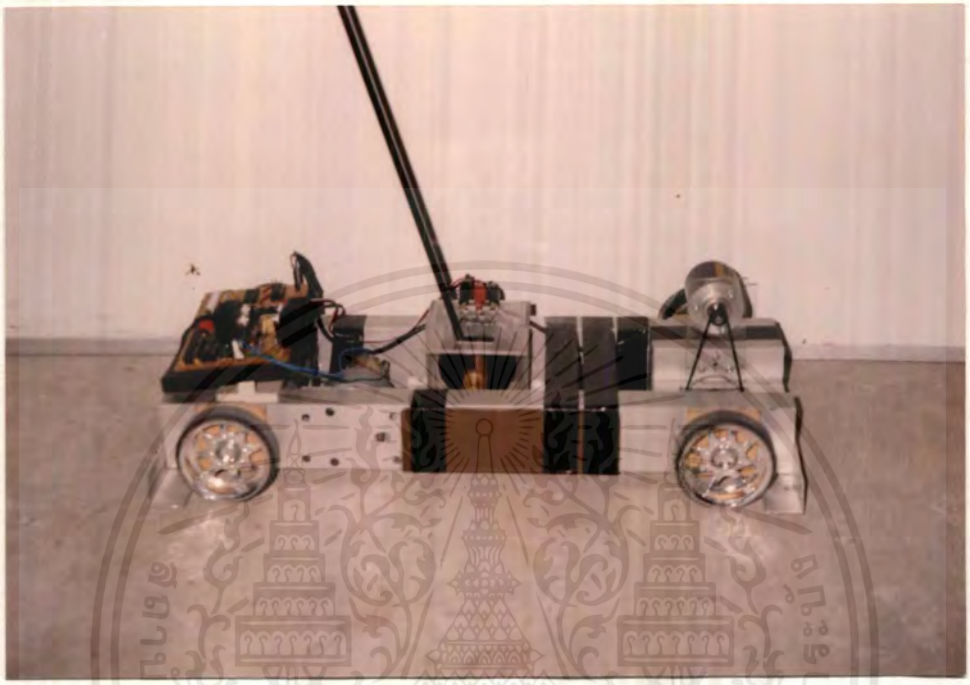


รูปที่ 2.1 แสดงบล็อกไดอะแกรมโดยรวมของระบบ

2.2 โครงสร้างของระบบ

2.2.1 โครงสร้างและส่วนประกอบรถอินเวิร์ทเพนดูลัม

ตัวรถถูกออกแบบให้มีลักษณะดังรูป



รูปที่ 2.2 รถอินเวิร์ทเพนดูลัม

2.2.2 ชุดควบคุมและเซ็นเซอร์

สามารถแบ่งได้เป็น 3 ส่วนคือ

ส่วนที่ 1 ส่วนตรวจจับ (Sensor)

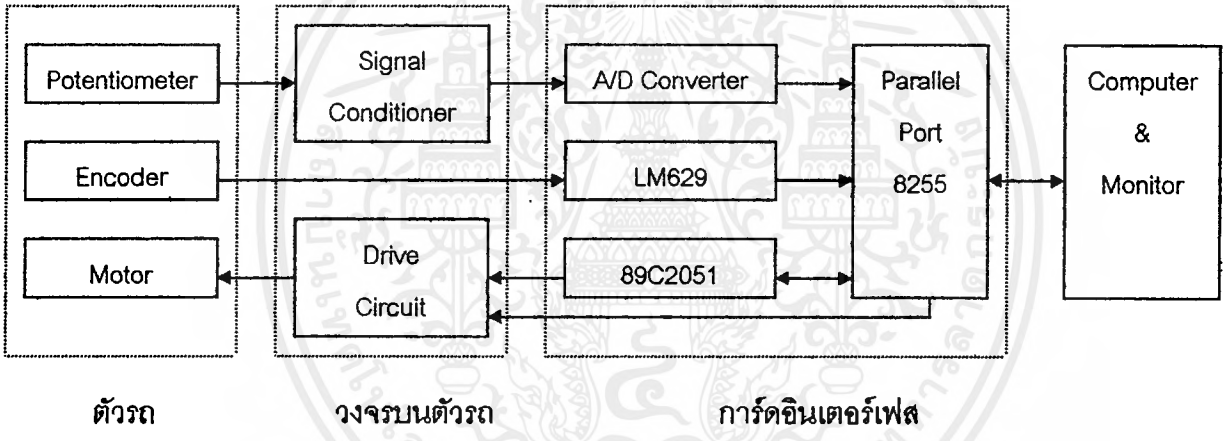
เซ็นเซอร์เป็นส่วนประกอบที่สำคัญในระบบคอนโทรลที่มีการป้อนกลับเพราะสัญญาณที่ป้อนกลับผ่านมาทางเซ็นเซอร์นั้นสามารถนำมาวิเคราะห์หาคุณสมบัติของระบบหรือสามารถบอกสถานะของระบบขณะนั้นได้ว่าเป็นอย่างไร ซึ่งจะได้นำมาหาสัญญาณควบคุมที่จะส่งไปควบคุมระบบ ดังนั้นถ้าสัญญาณป้อนกลับที่ได้มีความถูกต้องมากและมีสัญญาณรบกวนน้อย ก็จะได้ค่าสัญญาณควบคุมที่สามารถควบคุมระบบให้เกิดประสิทธิภาพสูงสุดได้ สำหรับโครงงานนี้ประกอบไปด้วยเซ็นเซอร์ 2 ชนิด คือ เอ็นโคดเดอร์ (Encoder) ที่ติดอยู่กับมอเตอร์ใช้ในการตรวจจับทิศทางการเคลื่อนที่และความเร็วของรถ และ โพเทนทิโอมิเตอร์ (Potentiometer) ใช้ในการวัดค่ามุมที่เบี่ยงเบนไปของก้านเพนดูลัม แล้วสัญญาณที่วัดได้จากทั้งเอ็นโคดเดอร์ และโพเทนทิโอมิเตอร์จะถูกส่งกลับไปยังหน่วยประมวลผลกลางและแสดงผล ซึ่งก็คือคอมพิวเตอร์ต่อไป

ส่วนที่ 2 หน่วยประมวลผลกลางและควบคุม

ได้แก่ คอมพิวเตอร์ส่วนบุคคล (Personal Computer , PC) และการ์ดอินเตอร์เฟส เลือกใช้ IBM PC เพราะสามารถนำมาต่อใช้งานร่วมกับอุปกรณ์ภายนอกต่าง ๆ ได้ง่าย มีความคล่องตัวสูง โดยจะใช้ไอซีเบอร์ LM629 เป็นตัวอ่านค่าพัลซ์จากเอ็นโคดเดอร์ทำให้รู้ตำแหน่งและความเร็วของรถ คอมพิวเตอร์จะแสดงค่าที่รับมาจากเซ็นเซอร์ผ่านทางหน้าจอและคอมพิวเตอร์จะประมวลผลแล้วส่งสัญญาณควบคุมไปยังวงจรขับมอเตอร์ การติดต่อกับคอมพิวเตอร์กระทำผ่านทางการ์ดอินเตอร์เฟส

ส่วนที่ 3 วงจรขับมอเตอร์

เป็นวงจรที่ควบคุมการหมุนมอเตอร์ซึ่งเป็นอุปกรณ์ที่ทำให้รถเคลื่อนที่ โดยรายละเอียดของวงจรขับมอเตอร์จะได้กล่าวถึงในบทต่อไป



รูปที่ 2.3 โครงสร้างของระบบรถอินเวิร์ทเพนดูลัม

2.3 มอเตอร์ไฟฟ้ากระแสตรง

2.3.1 หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงเป็นทรานสดิวเซอร์แรงบิด ซึ่งมีการออกแบบให้มีคุณลักษณะพิเศษ คือ แรงบิดของเพลามอเตอร์ไฟฟ้ากระแสตรงกับกระแสอาร์เมเจอร์ แรงบิดของเพลาของมอเตอร์จะได้จากผลระหว่างสนามแม่เหล็กและขดลวดตัวนำ หลักการในที่นี้กระแสที่ไหลในขดลวดตัวนำจะสร้างฟิลด์ที่ประกอบด้วยเส้นแรงแม่เหล็ก และขดลวดตัวนำเหล่านั้นอยู่ห่างจากศูนย์กลางการหมุนเท่ากับ r ความสัมพันธ์ระหว่างแรงบิดของเพลาและกระแสเท่ากับ

$$T = K\varnothing I$$

เมื่อ T คือ แรงบิดของเพลา มีหน่วยเป็น นิวตัน – เมตร

$\varnothing$ คือ เส้นแรงแม่เหล็ก มีหน่วยเป็น เวเบอร์

I คือ กระแส มีหน่วยเป็น แอมแปร์

K คือ ค่าคงที่

ดังนั้นแรงบิดของเพลาจะเป็นสัดส่วนโดยตรงกับผลคูณของเส้นแรงแม่เหล็ก และกระแสเมื่อขดลวดตัวนำเคลื่อนที่ในสนามแม่เหล็กก็จะทำให้เกิดโวลท์เตจตกคร่อมตัวมันเอง โวลท์เตจนี้จะเป็นสัดส่วนกับความเร็วของเพลาของมอเตอร์และต้านการไหลของกระแส ความสัมพันธ์ระหว่างโวลท์เตจย้อนกลับนี้และความเร็วของเพลามอเตอร์ คือ

$$E = K\varnothing W$$

เมื่อ E คือ โวลท์เตจย้อนกลับ มีหน่วยเป็น โวลท์

$\varnothing$ คือ เส้นแรงแม่เหล็ก มีหน่วยเป็น เวเบอร์

W คือ ความเร็วของมอเตอร์ มีหน่วยเป็น เรเดียน / วินาที

2.3.2 พื้นฐานของระบบควบคุมมอเตอร์กระแสตรง

ส่วนประกอบพื้นฐานของระบบควบคุมมอเตอร์กระแสตรงประกอบด้วยบล็อคที่สำคัญ 4 บล็อก คือ

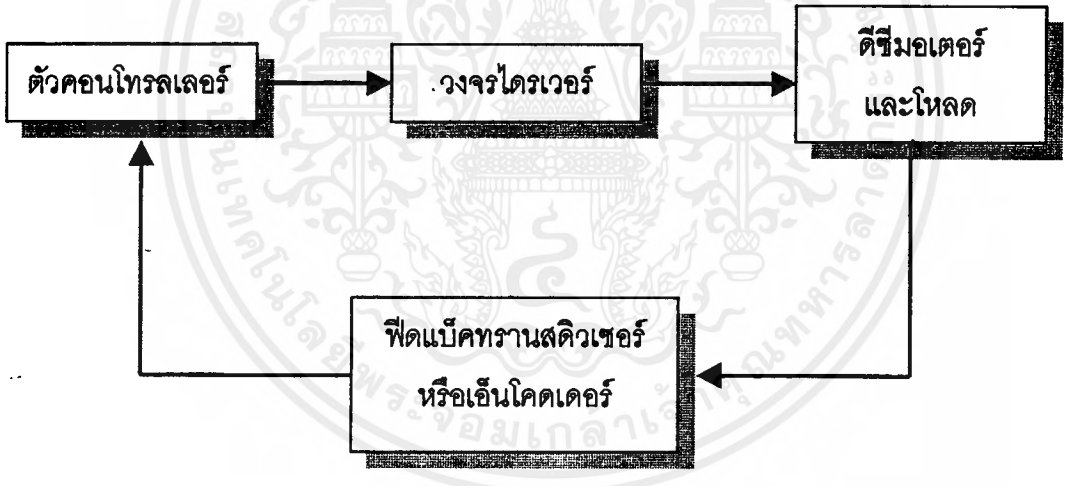
1. ตัวคอนโทรลเลอร์
2. วงจรไดรเวอร์หรือเพาเวอร์แอมปลิไฟล์
3. ฟีดแบ็คทรานสดิวเซอร์หรือเอ็นโคดเดอร์
4. ดีซีมอเตอร์และโหลด

ตัวคอนโทรลเลอร์ เป็นส่วนของระบบที่ทำให้เกิดสัญญาณคอนโทรลไปบังคับดีซีมอเตอร์และโหลด คอนโทรลเลอร์ที่ให้สัญญาณคอนโทรลเป็นสัญญาณอนาล็อก เราเรียกว่าอนาล็อกคอนโทรลเลอร์ ส่วนคอนโทรลเลอร์ที่ให้สัญญาณคอนโทรลเป็นสัญญาณดิจิตอล เราเรียกว่าดิจิตอลคอนโทรลเลอร์

วงจรไดรเวอร์ เป็นส่วนประกอบของระบบที่อยู่ระหว่างตัวคอนโทรลเลอร์กับดีซีมอเตอร์และโหลดมีหน้าที่ปรับรูปและขยายสัญญาณให้เหมาะสมก่อนที่จะป้อนเข้าไปขับดีซีมอเตอร์และโหลด วงจรไดรเวอร์ส่วนใหญ่ ได้แก่ เพาเวอร์แอมปลิไฟล์ซึ่งอาจแบ่งย่อยออกเป็นลิเนียร์เพาเวอร์แอมปลิไฟล์และพัลซ์วิดโมดูเลชั่นแอมปลิไฟล์

ฟีดแบ็คทรานสดิวเซอร์หรือเอ็นโคดเดอร์ เป็นสิ่งประดิษฐ์ที่ใช้รับรู้หรือดีเท็คสัญญาณเอาท์พุทที่ต้องการ โดยไม่มีผลของการโหลดดิ้ง (loading) สัญญาณที่ดีเท็คได้นี้จะป้อนกลับไปเปรียบเทียบกับสัญญาณอ้างอิงทำให้ได้สัญญาณผิดพลาด (error) ฟีดแบ็คทรานสดิวเซอร์แบ่งออกได้เป็น 2 แบบ คือ อนาล็อกทรานสดิวเซอร์ คือสิ่งประดิษฐ์ใช้เปลี่ยนพลังงานรูปหนึ่งให้เป็นสัญญาณอนาล็อก ได้แก่ พวกทาโคเจนเนอเรเตอร์ โพเทนทิโอมิเตอร์ และซิงโคร เป็นต้น ส่วนฟีดแบ็คทรานสดิวเซอร์อีกแบบหนึ่ง คือ ดิจิตอลทรานสดิวเซอร์ เป็นสิ่งประดิษฐ์ที่ใช้เปลี่ยนพลังงานรูปหนึ่งให้เป็นสัญญาณดิจิตอล ได้แก่ พวกอินครีเม้นท์เอ็นโคดเดอร์ รีโซลเวอร์ แมกเนติกพิคอัพ เป็นต้น

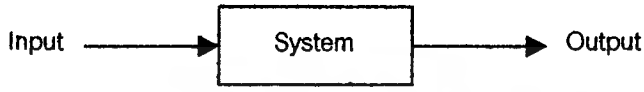
ดีซีมอเตอร์และโหลด คือระบบที่ถูกคอนโทรลหรือส่วนที่ออกแรงทำงานซึ่งจะเป็นเครื่องจักรกล (ดีซีมอเตอร์) หรืออะไรก็ตามที่ให้ตัวแปร ดีซีมอเตอร์ในที่นี้เป็นแบบแม่เหล็กถาวรที่มีคุณสมบัติการทำงานสูง มีอาร์เมเจอร์อินดักแตนซ์และแรงเฉื่อยของโรเตอร์ต่ำ



รูปที่ 2.4 ส่วนประกอบพื้นฐานของระบบควบคุมดีซีมอเตอร์

2.4 ทฤษฎีการวิเคราะห์ระบบด้วยวิธีสเตทสเปซ (State – Space Analysis)

การวิเคราะห์ระบบแบบ Classical Control นั้น ระบบจะประกอบด้วยอินพุทและเอาท์พุทอย่างละ 1 ตัวเท่านั้น



แต่สำหรับระบบแบบ Modern Control ระบบจะมีความซับซ้อนมากขึ้น โดยจะประกอบด้วยอินพุทและเอาท์พุตหลายตัว การวิเคราะห์ระบบแบบนี้นั้นจะอยู่บนพื้นฐานของหลักการวิเคราะห์แบบสเตท (The Concept of State)

Eigenvalues

จากสมการ Linear Differential Equations

$$y^{(n)} + a_1 y + ^{(n-1)} + \ldots + a_{n-1}\dot{y} + a_n y = u$$

ให้

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} \\ x_{n-1} &= y^{(n-2)} \\ &\vdots \\ x_n &= y^{(n-1)} \end{aligned}$$

จะได้

$$\begin{aligned} \dot{x}_1 &= \dot{y} = x_2 \\ \dot{x}_2 &= \ddot{y} = x_3 \\ &\vdots \\ \dot{x}_{n-1} &= y^{(n-1)} = x_n \\ \dot{x}_n &= y^{(n)} = -a_n x_1 - a_{n-2} x_2 - \ldots - a_1 x_n + u \end{aligned}$$

เขียนสมการให้อยู่ในรูปเมตริกซ์

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

จะได้สมการสถานะ (State Equation) ของระบบ คือ

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$

เมื่อ **A** คือ System Matrix

B คือ Input Vector

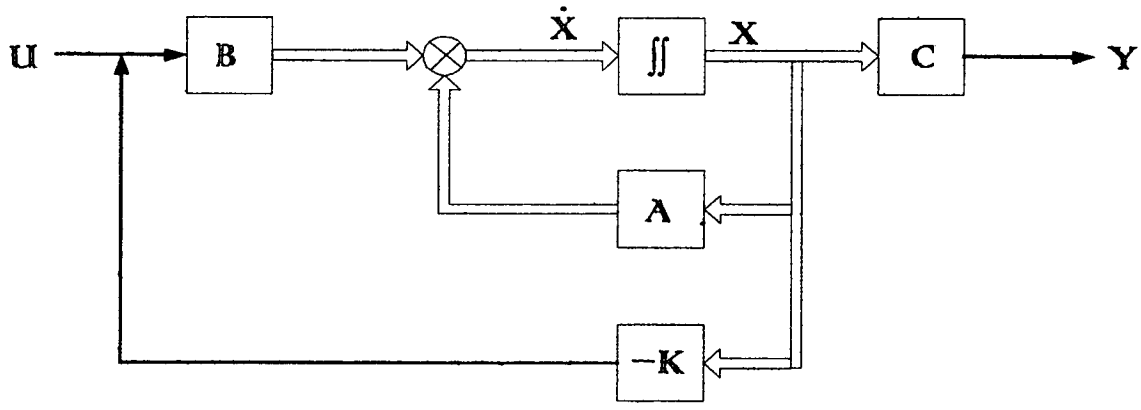
และ $$y = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

จะได้สมการเอาท์พุท (Output Equation) ของระบบคือ

$$\mathbf{Y} = \mathbf{CX}$$

เมื่อ **C** คือ Output Vector

เมื่อทำการป้อนกลับระบบด้วยค่าเกน (Gain) ค่าหนึ่ง



นั่นคือ $\mathbf{U} = -\mathbf{K}\mathbf{X}$ ====> Control Law

เมื่อ $\mathbf{K}$ คือ Feedback Gain

ทดสอบ Controllability ของระบบ

สร้างเมตริกซ์ Q_c จากเมตริกซ์ และ

$$Q_c = \left[\mathbf{B} : \mathbf{AB} : \mathbf{A}^2\mathbf{B} : \ldots : \mathbf{A}^{n-1}\mathbf{B}\right]_{n\times n}$$

เรียกเมตริกซ์นี้ว่า Controllability Matrix

แล้วทดสอบว่า Q_c มี Rank = n หรือไม่ ถ้า Rank(Q_c) = n หรือ det $Q_c \neq 0$ ระบบจะสามารถควบคุมได้ (Controllable)

จะต้องหา Transformation Matrix , T

เมื่อ $T = Q_cW$

$$w = \begin{bmatrix} a_{n-1} & a_{n-2} & \ldots & a_1 & 1 \\ a_{n-2} & a_{n-3} & & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_1 & 1 & & 0 & 0 \\ 1 & 0 & \ldots & 0 & 0 \end{bmatrix}$$

เมื่อ a_i เป็นสัมประสิทธ์ของสมการคุณลักษณะ (Characteristic Equation)

$$\left|s\mathbf{I} - \mathbf{A}\right| = s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n$$

ต้องการให้โพลวงปิด หรือ Eigenvalue ของระบบเป็น $\mu_1, \mu_2, \ldots, \mu_n$

$$(s-\mu_1)(s-\mu_2)\ldots(s-\mu_n) = 0$$

$$s^n + \alpha_1 s + \ldots + \alpha_{n-1}s + \alpha_n = 0$$

เพราะฉะนั้น

$$(s-\mu_1)(s-\mu_2)\ldots(s-\mu_n) = s^n + \alpha_1 s + \ldots + \alpha_{n-1}s + \alpha_n$$

จะได้

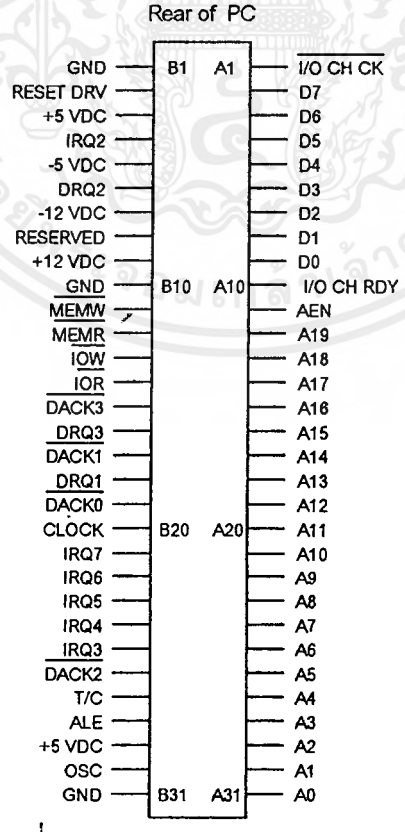
$$K = \left[\alpha_n - a_n \quad \alpha_{n-1} - a_{n-1} \quad \ldots \quad \alpha_2 - a_2 \quad \alpha_1 - a_1\right]T^{-1}$$

บทที่ 3
ส่วนประกอบและวงจรโดยรวมของระบบ

3.1 วงจรเชื่อมต่อระหว่างคอมพิวเตอร์และอุปกรณ์ภายนอก

การควบคุมอุปกรณ์ภายนอกโดยใช้เครื่องคอมพิวเตอร์นั้น จะต้องทำให้เครื่องคอมพิวเตอร์ติดต่อกับอุปกรณ์ภายนอกให้ได้เสียก่อน การติดต่อสื่อสารระหว่างตัวรถกับคอมพิวเตอร์ในโครงงานนี้จะทำผ่านการ์ดอินเตอร์เฟส (Interface Card) ซึ่งเป็นการ์ดที่ถูกออกแบบขึ้นเพื่อใช้ในการควบคุมและส่งผ่านข้อมูลโดยเฉพาะ และยังได้รวมชิปไอซีเบอร์ LM629 เพื่อใช้เป็นตัวอ่านค่าจากเอ็นโคดเดอร์ และไอซีที่ทำหน้าที่เปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล คือ ไอซีเบอร์ ADC0848 และไมโครคอนโทรลเลอร์เบอร์ 89C2051 เพื่อใช้เป็นตัวกำเนิด Pulse Width Modulation โดยวงจรเชื่อมต่อนี้ทำหน้าที่เป็นเหมือนพอร์ท รวมทั้งทำหน้าที่เป็นบัฟเฟอร์ของสัญญาณข้อมูล และสัญญาณควบคุมต่าง ๆ ด้วย

ภายในเครื่องคอมพิวเตอร์จะมีส่วนที่ใช้เชื่อมต่อกับอุปกรณ์ภายนอก โดยจะเป็นช่องเสียบที่เรียกว่า SLOT PC โดยตำแหน่งบัสใน IBM คอมพิวเตอร์ (IBM PC system bus) แสดงได้ดังนี้



รูปที่ 3.1 IBM PC system bus

สัญญาณที่ออกจาก IBM PC system bus ที่การ์ดอินเตอร์เฟสใช้ มีดังนี้

A0 – A9 เป็นแอดเดรสของระบบที่ใช้ติดต่อกับหน่วยความจำและอุปกรณ์อินพุท เอาท์พุท

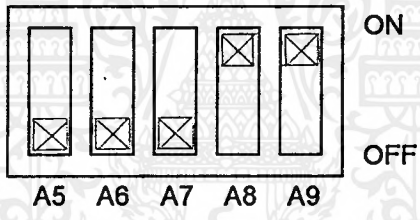
D0 - D9 เป็นสัญญาณข้อมูลขนาด 8 ใช้ติดต่อกับหน่วยความจำของไมโครโปรเซสเซอร์

IOW เป็นสัญญาณเขียนข้อมูลลงบนอุปกรณ์อินพุท เอาท์พุท สัญญาณนี้ควบคุมโดยไมโครโปรเซสเซอร์

IOR เป็นสัญญาณอ่านข้อมูลจากอุปกรณ์อินพุท เอาท์พุท สัญญาณนี้ควบคุมโดยไมโครโปรเซสเซอร์

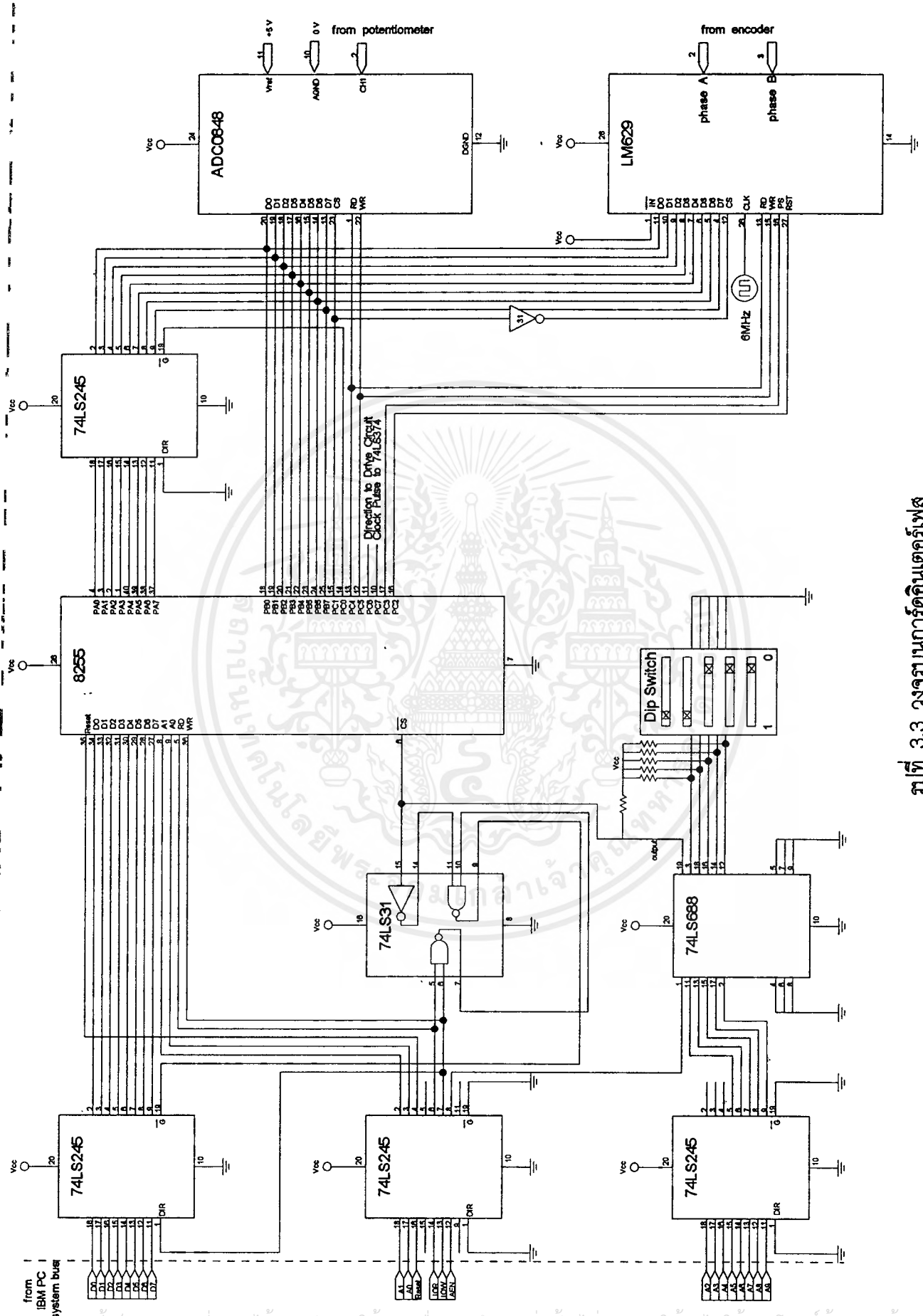
AEN คือสัญญาณ Enable Address เป็นสัญญาณเอาท์พุท

แอดเดรสที่คอมพิวเตอร์อนุญาตให้ใช้ในการพัฒนาเพื่อการติดต่อระหว่างอุปกรณ์ภายนอกกับคอมพิวเตอร์ อยู่ที่ตำแหน่ง 30F – 31F การนำแอดเดรสนี้ไปใช้สามารถทำได้โดยการเซ็ทตำแหน่งของ DIP Switch บนการ์ดอินเตอร์เฟส ให้ตรงตามตำแหน่งที่ต้องการใช้ การติดต่อกับเครื่องคอมพิวเตอร์นี้ใช้การติดต่อพอร์ทแบบตายตัว (Fix decode port) การปรับตั้งตำแหน่งของ DIP Switch สำหรับ Decode Address แสดงได้ดังรูป



รูปที่ 3.2 ตำแหน่งของ DIP Switch

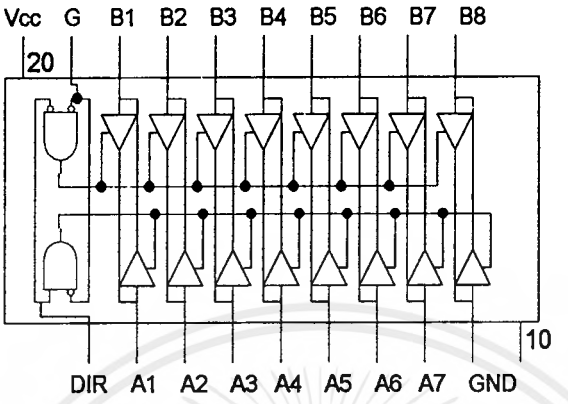
วงจรบนการ์ดอินเตอร์เฟส แสดงได้ดังรูป 3.3



ภาพที่ 3.3 วงจรบนการ์ดอินเตอร์เฟส

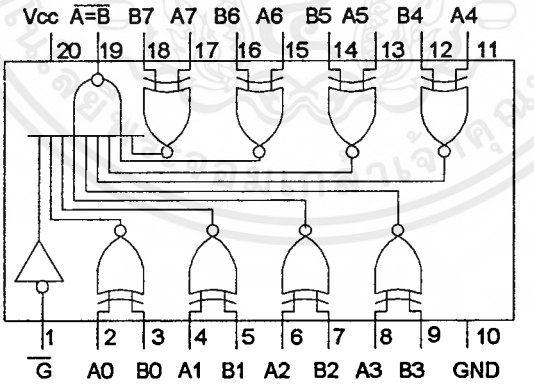
โดยอินเตอร์เฟสการ์ดนี้จะประกอบด้วยส่งต่าง ๆ คือ

- ชิปที่ทำหน้าที่เป็นตัวบัฟเฟอร์ข้อมูล เลือกใช้ไอซีเบอร์ 74LS245 ซึ่งเป็นบัฟเฟอร์สองทิศทางแต่ในโครงงานนี้กำหนดให้ทำงานเพียงทิศทางเดียว โดยขาของ 74LS245 แสดงได้ดังนี้



รูปที่ 3.4 ไอซีเบอร์ 74LS245

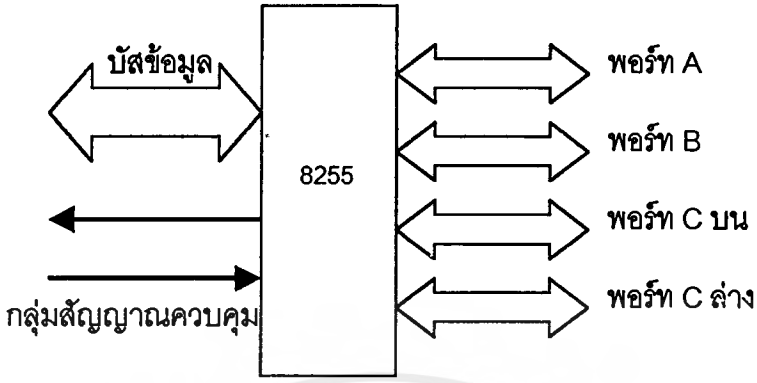
- Comparator เลือกใช้ไอซีเบอร์ 74LS688 เป็นตัวดีโค้ดแอดเดรส (Decode Address) กำหนดการทำงานของวงจรต่าง ๆ บนการ์ดว่าจะให้ทำงานเมื่อใด โดยเมื่อแอดเดรสของระบบบัสของคอมพิวเตอร์ตรงกับตำแหน่งที่ตั้งไว้โดย DIP Switch ก็จะมีสัญญาณเลือกชิปสั่งให้ชิปบนการ์ดอินเตอร์เฟสทำงานตามวงจรที่ต่อเอาไว้โดยผ่านเกทต่าง ๆ ในไอซีเบอร์ 74LS688 แสดงได้ดังนี้



รูปที่ 3.5 ไอซีเบอร์ 74LS688

- ชิปที่ทำหน้าที่เป็นพอร์ทแบบขนาน เลือกใช้ไอซีเบอร์ 8255 ที่เป็นไอซีในตระกูลของ 8080 ที่บริษัทอินเทลได้ออกแบบมาให้ใช้งานร่วมกับซีพียู 8080 อย่างไรก็ตามเราสามารถประยุกต์ใช้กับไมโครโปรเซสเซอร์ตัวอื่น ๆ ได้เช่นกัน เพราะใช้งานได้ง่ายและมีความคล่องตัวสูง การออกแบบที่ต้อง

ใช้พอร์ทขนานมักเลือกใช้ 8255 เป็นตัวอินเตอร์เฟสกับอุปกรณ์ภายนอก 8255 เป็นไอซีที่มี 40 ขา ใช้ต่อเป็นพอร์ทให้ไมโครโปรเซสเซอร์ได้ 3 พอร์ท โดยมีโครงสร้างพื้นฐานดังแสดงในรูป



รูปที่ 3.6 โครงสร้างพื้นฐานของ 8255

การเรียกพอร์ทของ 8255 จะเรียกพอร์ทต่าง ๆ ว่า พอร์ท A พอร์ท B และ พอร์ท C ที่พิเศษคือ พอร์ททุกพอร์ทสามารถเป็นได้ทั้งพอร์ทอินพุทและพอร์ทเอาท์พุท

การทำงานของ 8255 จะใช้สัญญาณควบคุมจากไมโครโปรเซสเซอร์มาควบคุมการทำงาน โดยไมโครโปรเซสเซอร์จะส่งคำสั่งมาโปรแกรมการทำงานหรือกำหนดรูปแบบของพอร์ทให้เป็นอินพุทหรือเอาท์พุทได้

การใช้งาน 8255 จะต้องส่งรหัสควบคุม (Control Code) เข้าไปยังพอร์ทข้อมูลควบคุมเพื่อควบคุมการทำงานของ 8255 ซึ่งมีหลายโหมดแต่ละโหมดจะแตกต่างกันออกไป การโปรแกรมให้ 8255 ทำงานจะทำได้ 3 โหมด คือ โหมด 0 โหมด 1 และโหมด 2

การโปรแกรม 8255 คือ การให้ค่ารหัสบิตต่างๆ เข้าไปในรหัสควบคุมแล้วส่งไปยังรีจิสเตอร์ของพอร์ทควบคุม ความหมายของบิตต่างๆ มีดังนี้

D7	D6	D5	D4	D3	D2	D1	D0

บิต D7 เป็นบิตที่แสดงรหัสคำสั่งควบคุม ถ้าเป็น 1 หมายถึงรหัสควบคุมนี้จะมีผลต่อการเปลี่ยนแปลงการเซตโหมดต่าง ๆ ของ8255

บิต D6 และ D5 เป็นการเลือกโหมดของพอร์ท A

บิต D4 ถ้าเป็น 0 หมายถึงพอร์ท A เป็นเอาท์พุท ถ้าเป็น 1 หมายถึงพอร์ท A เป็นอินพุท

บิต D3 ถ้าเป็น 0 หมายถึงพอร์ท C บนเป็นเอาท์พุท ถ้าเป็น 1 หมายถึงพอร์ท C บนเป็นอินพุท

บิต D2 เป็นการเลือกโหมดของพอร์ท B

บิต D1 ถ้าเป็น 0 หมายถึงพอร์ท B เป็นเอาท์พุท ถ้าเป็น 1 หมายถึงพอร์ท B เป็นอินพุท

บิต D0 ถ้าเป็น 0 หมายถึงพอร์ท C ล่างเป็นเอาท์พุท ถ้าเป็น 1 หมายถึงพอร์ท C ล่างเป็นอินพุท

การทำงานในโหมดต่าง ๆ ของ 8255 สามารถศึกษาได้เพิ่มเติมจากหนังสือไมโครโปรเซสเซอร์ทั่วๆไป

ในโครงงานนี้ เราต้องการใช้ 8255 เพื่อทำงานเป็นอินพุทพอร์ทและเอาท์พุทพอร์ท ดังนี้ พอร์ท A เป็นเอาท์พุท, พอร์ท B เป็นอินพุท, พอร์ท C เป็นเอาท์พุท และทุกพอร์ททำงานในโหมด 0 ฉะนั้นจะได้ Control Code คือ 10000010

- ตัวแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก เลือกใช้ไอซีเบอร์ ADC0848 ที่เป็น CMOS 8 - Bit Converter ที่สามารถรับสัญญาณอินพุทอนาล็อกได้หลายแบบ ในโครงงานนี้ ADC0848 จะถูกใช้ในโหมด Single – Ended โดยใช้ขาแนล 1 รับสัญญาณอนาล็อกจากวงจรของโพเทนทิโอมิเตอร์

โครงสร้างของ ADC0848 เป็นไอซีที่สามารถต่อใช้งานร่วมกับไมโครโปรเซสเซอร์ได้ง่าย มีสัญญาณนาฬิกาภายในตัวเอง ไม่ต้องปรับ Zero Adjust รับสัญญาณอินพุทได้ตั้งแต่ 0 ถึง 5 โวลท์ ใช้เพาเวอร์ซัพพลาย 5 โวลท์ ตัวชิปมี 24 ขา

- ชิพที่ทำหน้าที่รับสัญญาณจากเอ็นโคดเดอร์ เลือกใช้ไอซีเบอร์ LM629 ซึ่งเป็นชิพที่ใช้ควบคุมการเคลื่อนที่ของ DC Motor และ DC Servomotor ตัวชิพมี 28 ขา

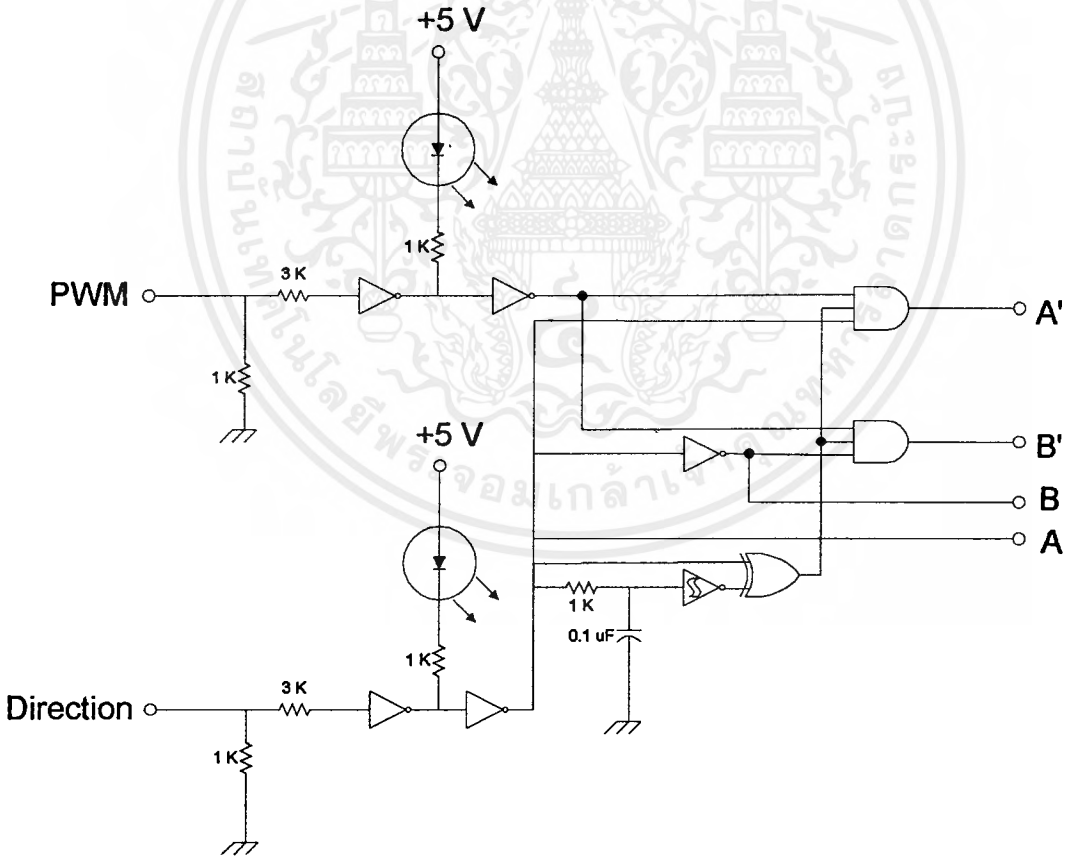
คุณสมบัติของ LM629 คือ

> เป็นฟิลเตอร์ดิจิตอลแบบ PID สามารถเขียนโปรแกรมควบคุมการทำงานเป็นแบบ PID ได้เลย
> เอาท์พุทเป็นสัญญาณ 8 Bit ที่มีทั้งขนาดและทิศทาง (8 – Bit Sign-Magnitude PWM)
> มีรีจิสเตอร์ตำแหน่ง ความเร็วและความเร่งขนาด 32 Bit
> สามารถกำหนดตำแหน่งได้
> สามารถทำงานได้ทั้งแบบโหมดการควบคุมตำแหน่งและโหมดการควบคุมความเร็ว
> สามารถเปลี่ยนแปลงความเร็วและทิศทางการเคลื่อนที่ได้ตลอดแม้ในขณะกำลังเคลื่อนที่อยู่
> ไอซีมีการชดเชยแรงเสียดทานที่พื้นให้ด้วยเมื่อใช้มอเตอร์ในการขับเคลื่อนรถอินเวิร์ทเพนดูลัม
> ใช้สัญญาณนาฬิกาจากภายนอก มีความถี่ได้สูงสุด 6 MHz
> มีหน่วยรับสัญญาณป้อนกลับจากเอนโคดเดอร์

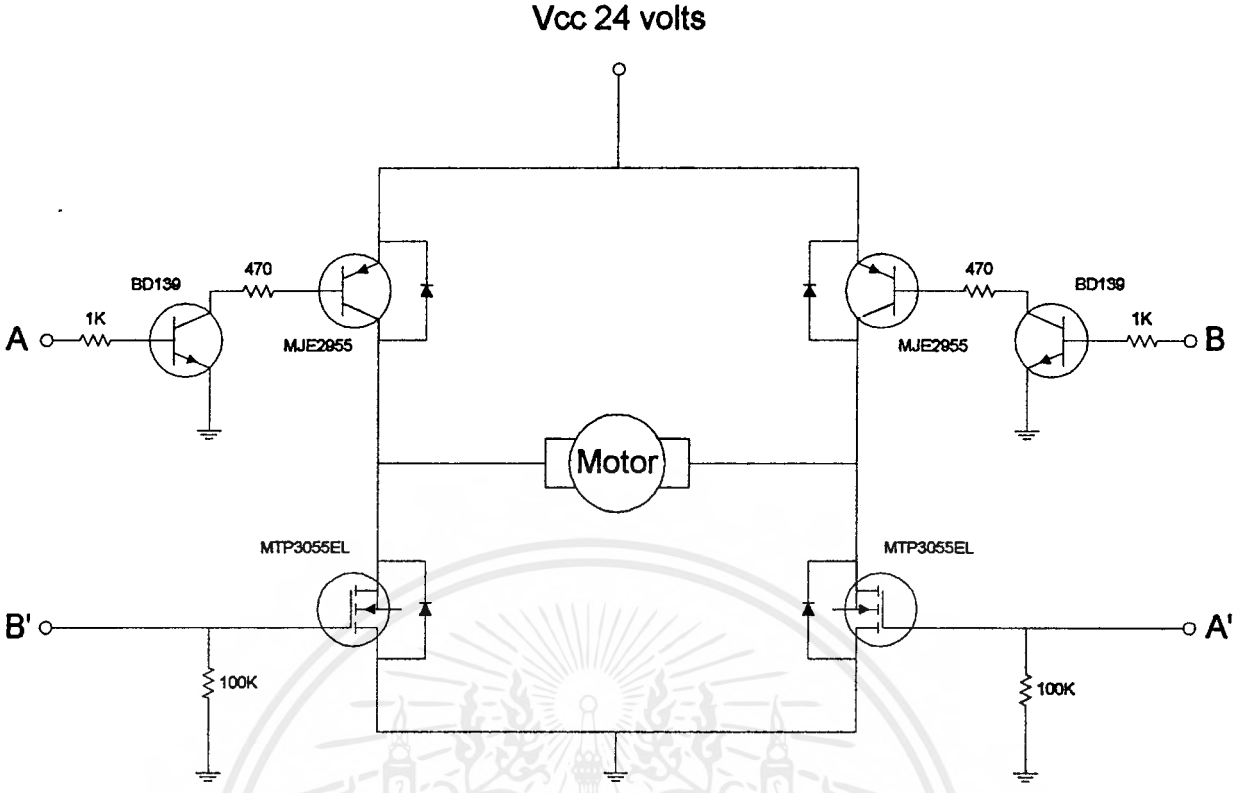
3.2 วงจรขับมอเตอร์

เป็นวงจรที่ควบคุมความเร็วและทิศทางมอเตอร์ ซึ่งเดิมการควบคุมการหมุนของมอเตอร์เราใช้ไอซีเบอร์ LM18201 ซึ่งเป็น Full Bridge ที่ออกแบบมาเพื่อควบคุมการเคลื่อนที่ของ DC Motor และ Stepper Motor โดยตรง แต่ปัญหาที่พบขณะใช้ไอซีตัวนี้คือ เมื่อจ่ายโวลท์เตจสูง ๆ ให้กับไอซี และส่งสัญญาณสั่งให้มอเตอร์เคลื่อนตัวกลับทิศไปมา ทำให้เกิดกระแสกระชากขึ้นสูง แต่ไอซี LM18201 สามารถทนกระแสกระชากได้เพียง 3 แอมแปร์ ตัวไอซีจะทนต่อภาวะเช่นนี้ไม่ได้ ทำให้ไอซีร้อนและเสียในที่สุด จึงเปลี่ยนมาใช้วงจรขับมอเตอร์แบบที่ใช้ทรานซิสเตอร์ร่วมกับมอสเฟ็ทแทน ซึ่งจะมีความทนทานมากกว่าและทำงานได้ดีกว่า

นอกจากนั้นวงจรขับมอเตอร์ยังต้องใช้ร่วมกับวงจรที่ช่วยเลือกว่าจะส่งค่า PWM และ Direction ไปในทิศทางใด และมีการต่อตัวต้านทานและบัฟเฟอร์เพื่อป้องกันการไหลย้อนกลับของกระแส ซึ่งอาจเป็นผลเสียต่อคอมพิวเตอร์และระบบได้



รูปที่ 3.7 วงจรป้องกันกระแสย้อนกลับและเลือกสัญญาณ



รูปที่ 3.8 วงจรขับมอเตอร์

3.3 วงจรของตัวตรวจจับ

ตัวตรวจจับ (Sensor) ที่ใช้ในรถอินเวิร์ทเพนดูลัมประกอบด้วย 2 ส่วนด้วยกัน คือ

1. ส่วนที่ใช้วัดการเปลี่ยนแปลงมุมของก้านอินเวิร์ทเพนดูลัม ว่าเบี่ยงเบนไปจากแนวตั้งฉากไปเป็นมุมเท่าไหร่
2. ส่วนที่ใช้วัดความเร็วและทิศทางการเคลื่อนที่ของรถ

โดยในโครงงานนี้มีการทดสอบเซ็นเซอร์ 3 ชนิดด้วยกัน ผลที่สังเกตและทดสอบได้มีดังนี้

- Hall Effect เป็นเซ็นเซอร์ที่ทำงานโดยอาศัยการเปลี่ยนแปลงของสนามแม่เหล็ก เมื่อเส้นฟลักซ์ตัดผ่านระนาบผิวของ Hall IC จะทำให้เกิดสัญญาณไฟฟ้าที่มีค่าแปรผันตรงกับความหนาแน่นของเส้นฟลักซ์นี้ แต่ปัญหาที่พบคือ อุปกรณ์ที่ใช้ในวงจร Hall Effect จะหาได้ยาก เช่น Permanent Magnet และ Hall IC สำหรับการทดลองใช้ Hall Effect เพื่อทดสอบการแกว่งตัวของก้านเพนดูลัมในโครงงานนี้พบว่า ค่าที่วัดได้จากตัว Hall Effect เมื่อก้านเพนดูลัมแกว่งตัวไปทางซ้ายและทางขวาจะไม่สมมาตรกันและคาดว่าปัญหานี้น่าจะเกิดมาจาก ความไม่สมดุลของแม่เหล็กที่ใช้

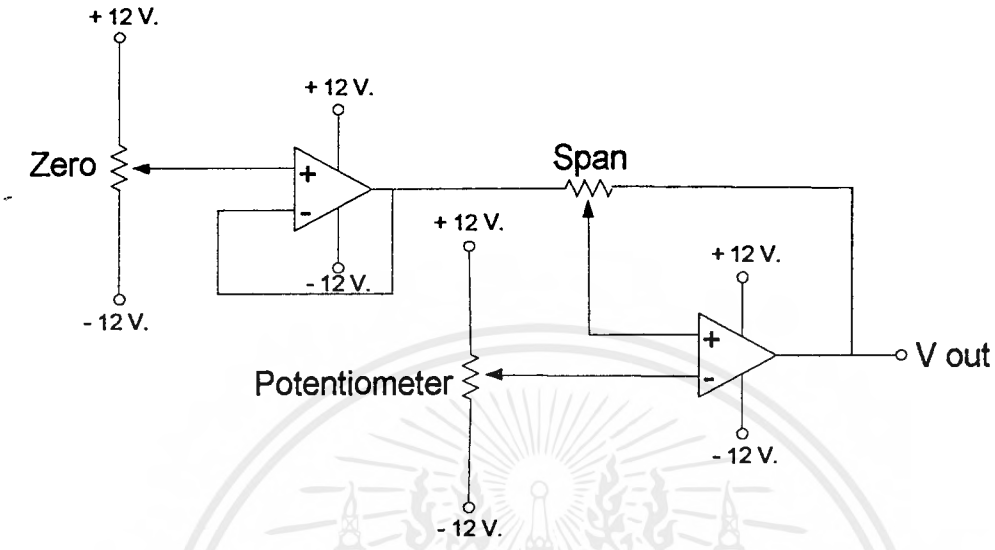
- Encoder จัดเป็นเซ็นเซอร์ที่ดีที่สุด คือมีความเที่ยงตรงสูง และค่าที่ได้เป็นค่าที่ถูกต้อง โดยเฉพาะอย่างยิ่งเอนโคดเดอร์ที่มีค่า Resolution สูง ๆ เช่น 4096 pulse/360° แต่เอนโคดเดอร์ก็มีข้อเสียคือ เมื่อตอนเริ่มต้นกระบวนการต้องหาจุด Equilibrium Position ให้ได้
- Potentiometer เป็นเครื่องมือที่เปลี่ยนพลังงานกลให้อยู่ในรูปพลังงานไฟฟ้า โดยอินพุทที่ป้อนให้แก่โพเทนทิโอมิเตอร์จะอยู่ในรูปของการเคลื่อนที่แบบเชิงกล และอาจเป็นการเคลื่อนที่แบบหมุนรอบหรือเชิงเส้นก็ได้ เป็นเซ็นเซอร์ที่เหมาะกับการวัดของระบบที่เคลื่อนที่ 2 ทิศทาง เช่น ใน joystick แต่ปัญหาที่พบคือ ระหรับช่วงมุมเล็ก ๆ สัญญาณเอาท์พุทของโพเทนทิโอมิเตอร์จะมี noise รบกวนระบบมาก ทำให้ค่าที่วัดได้ไม่ถูกต้อง

สำหรับในโครงงานนี้เลือกใช้เซ็นเซอร์ 2 แบบ คือ

- ใช้เอนโคดเดอร์สำหรับวัดทิศทางและความเร็วของรถ โดยต่อสายพานระหว่างแกนของเอนโคดเดอร์และแกนของล้อรถโดยตรง แล้วใช้ไอซี LM629 เป็นตัวรับพัลซ์ที่มีความถี่ต่าง ๆ กันและแปรค่าตรงกับการหมุนของเพลาของล้อ ซึ่งสามารถนำไปใช้ในการรับรู้ความเร็วของเพลาของมอเตอร์ในรูปของอัตราจำนวนพัลซ์ได้ โดยเอนโคดเดอร์ที่ใช้มีค่า Resolution 400 pulse/360° แล้วคอมพิวเตอร์จะทำการพล็อตกราฟการเคลื่อนที่ของรถทางหน้าจอ
- ใช้โพเทนทิโอมิเตอร์ เพื่อวัดมุมของก้านอินเวิร์ทเพนดูลัม โพเทนทิโอมิเตอร์จะทำงานคล้ายตัวต้านทานปรับค่าได้ที่มีแกนกลางหมุนได้ สำหรับโพเทนทิโอมิเตอร์ที่ใช้เป็นชนิดที่มีความต้านทาน 1 K ต่อการหมุนของแกน 1 รอบ

โพเทนทิโอมิเตอร์จะทำหน้าที่เปลี่ยนค่ามุมที่ได้มาเป็นสัญญาณโวลท์เตจ โดยการจ่ายค่าแรงดันคงที่ให้กับโพเทนทิโอมิเตอร์ ค่าแรงดันตกคร่อมที่ได้จะถูกนำมาเปรียบเทียบกับค่าแรงดันที่ได้จากค่าความต้านทานปรับค่าได้ ซึ่งใช้เป็นแรงดันอ้างอิง ผลที่ได้จะถูกปรับให้อยู่ในช่วงใช้งาน 0-5 โวลท์ โดยการปรับ Zero และ Span แล้วนำผลที่ได้ไปผ่านตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล (A/D Converter) บนการ์ดอินเตอร์เฟส แล้วส่งให้ซีพียูประมวลผลต่อไป

วงจรที่ใช้รับค่าโวลท์เตจจากโพเทนทิโอมิเตอร์ และปรับระดับแรงดันเอาท์พุทให้ตรงตามต้องการคือ 0 – 5 โวลท์ ซึ่งเป็นระดับที่ A/D Converter สามารถรับได้ แสดงดังรูป

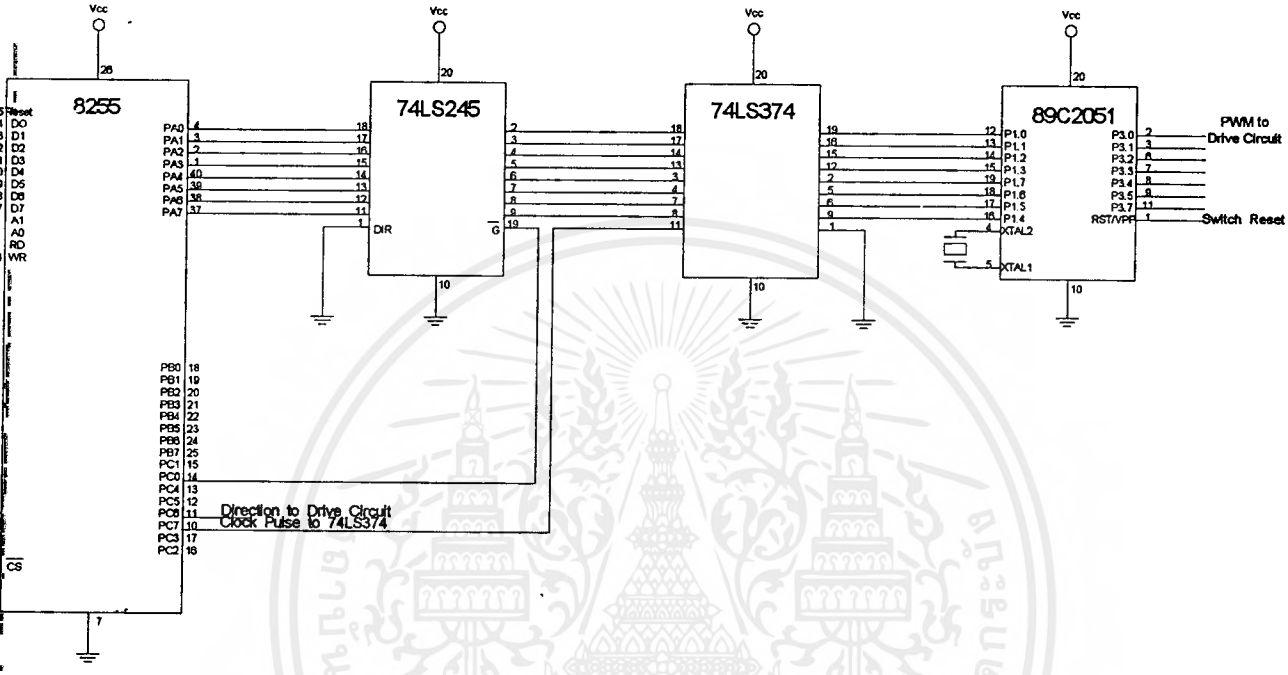


รูปที่ 3.9 วงจรรับและปรับค่าจากโพเทนทิโอมิเตอร์

3.4 วงจรกำเนิดพัลซ์วิดท์

เป็นส่วนหนึ่งของวงจรบนการ์ดอินเตอร์เฟสที่เพิ่มเติมมาเพื่อใช้เป็นตัวกำเนิด Pulse Width Modulation (PWM) ให้แก่วงจรขับมอเตอร์ ซึ่งแต่เดิมเราใช้ไอซีเบอร์ LM629 เป็นตัวควบคุมการเคลื่อนที่ของรถ โดยให้ LM629 เป็นตัวส่งทั้งค่า PWM และค่า Direction แต่ปัญหาที่พบคือ จากคุณสมบัติของ LM629 ที่จะพยายามรักษาตำแหน่งจริงให้เท่ากับตำแหน่งที่สร้างขึ้น ทำให้ไม่สามารถเปลี่ยนแปลงความเร็วแบบทันทีทันใดได้ จึงไม่เหมาะสมกับการควบคุมความเร็วมอเตอร์แบบที่ต้องการเปลี่ยนแปลงความเร็วอย่างรวดเร็ว เช่น ถ้าตั้งความเร็วไว้ที่ 3 เมตร/วินาที LM629 ก็จะสร้างกราฟการเคลื่อนที่ขึ้นมา เมื่อเวลาผ่านไป 1 วินาที และต้องการเปลี่ยนแปลงความเร็วเป็น -3 เมตร/วินาที ถ้าตำแหน่งจริงที่มอเตอร์ขับเคลื่อนได้ยังไม่ถึง 3 เมตร มอเตอร์จะไม่สามารถกลับทิศได้ทันที โดยจะวิ่งไปจนครบ 3 เมตรก่อนจึงจะเริ่มเปลี่ยนทิศทางการหมุนของมอเตอร์ได้ สรุปคือ การใช้ LM629 ในการควบคุมการเคลื่อนที่ของรถอินเวิร์ทเพนดูลัมไม่สามารถทำได้ เพราะรถไม่สามารถเปลี่ยนแปลงความเร็วได้ทันตามต้องการทั้งในการเคลื่อนที่ทิศทางเดียวกันและเคลื่อนที่กลับทิศกัน ทำให้ไม่สามารถรักษาตำแหน่งของก้านอินเวิร์ทเพนดูลัมไว้ได้

จากสาเหตุดังกล่าว จึงได้เปลี่ยนมาใช้ไมโครคอนโทรลเลอร์ 89C2051 เป็นตัวกำเนิด PWM และค่า Direction ถูกส่งจากคอมพิวเตอร์ผ่าน 8255 เข้าสู่วงจรขับมอเตอร์โดยตรง วงจรกำเนิด PWM นี้จะรวมอยู่บนการ์ดอินเตอร์เฟส ประกอบด้วยตัว Latch ค่า คือ ไอซีเบอร์ 74LS374 และไมโครคอนโทรลเลอร์ 89C2051 การต่อไอซีทั้ง 2 ตัวแสดงได้ดังรูป



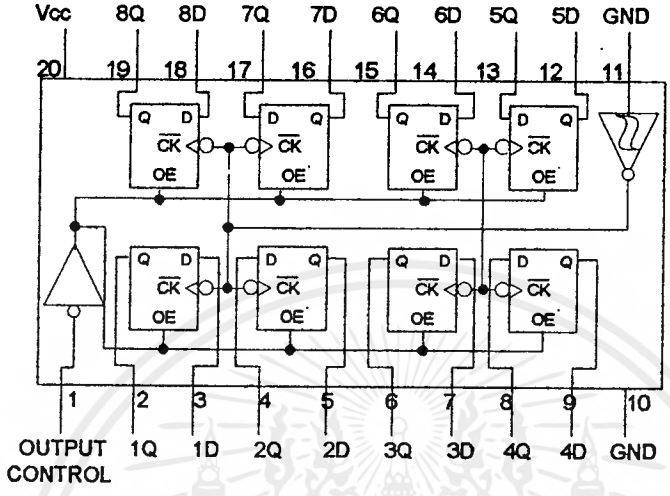
รูปที่ 3.10 วงจรกำเนิด PWM

จากวงจรการติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ 89C2051 จะทำผ่านไอซีที่ทำหน้าที่เป็นพอร์ทให้วงจรคือ 8255 โดยใช้ดาต้าบัสเดียวกันกับ ADC0848 และ LM629 มีไอซีเบอร์ 74LS374 ทำหน้าที่เป็นตัว Latch ค่าข้อมูลไว้ไม่ให้เปลี่ยนแปลงจนกว่าจะมีสัญญาณ Clock ขอบขาขึ้นที่สร้างจากโปรแกรมในคอมพิวเตอร์มาทริกให้มีการเปลี่ยนข้อมูล ตัวคอนโทรลเลอร์จะนำข้อมูลที่อ่านได้มาสร้างเป็นสัญญาณ PWM เพื่อส่งให้วงจรขับมอเตอร์ สำหรับโปรแกรมสร้าง PWM นี้ดูได้จากภาคผนวก ง

- ไมโครคอนโทรลเลอร์ 89C2051 เป็น 8 Bit ไมโครคอมพิวเตอร์ที่มีหน่วยความจำขนาด 2 กิโลไบท์ชนิด EPROM (Erasable & Programmable Read Only Memory) โครงสร้างภายในประกอบด้วยพอร์ท 2 พอร์ท คือ พอร์ท 1 และพอร์ท 3 สำหรับวงจรบนการ์ดอินเตอร์เฟสนี้ใช้พอร์ท 1 เป็นอินพุท และพอร์ท 3 เป็นเอาท์พุทส่งค่า PWM 89C2051 มีไทม์เมอร์ภายใน 1 ตัวคือ

ไทม์เมอร์ 0 ใช้เป็นตัวนับพัลซ์เพื่อกำหนดความกว้างของ PWM โดย Clock ที่จ่ายให้กับ 89C2051 มีความถี่ 6 MHz

- ไอซีเบอร์ 74LS374 ที่ทำหน้าที่ Latch ค่าข้อมูลมีโครงสร้างภายในดังนี้



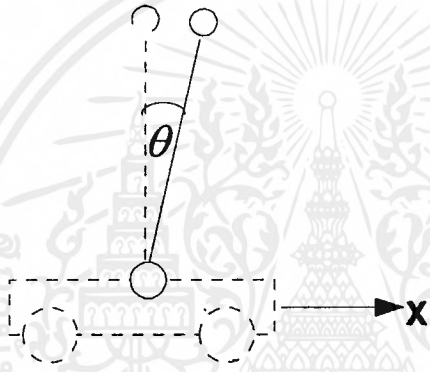
รูปที่ 3.11 ไอซีเบอร์ 74LS374

บทที่ 4
การวิเคราะห์ระบบ

4.1 Control Analysis

การพิจารณาระบบรถอินเวิร์ทเพนดูลัม แบ่งได้เป็น 2 ส่วนคือ

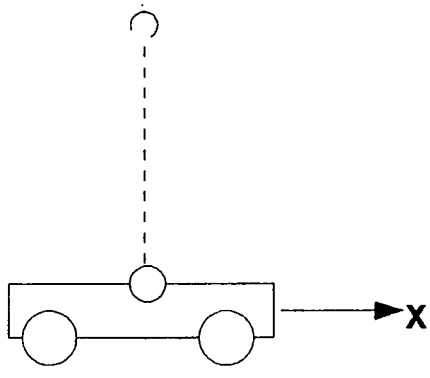
- ส่วนที่ 1 เป็นระบบที่มี อินพุท คือ ระยะจริงของรถ (X)
  เอาท์พุท คือ มุมของก้านลูกตุ้ม (θ)



ซึ่งเป็นส่วนของระบบที่ไม่สามารถควบคุมได้

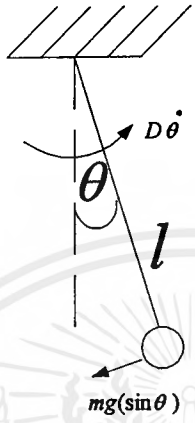
- ส่วนที่ 2 เป็นระบบที่มี อินพุท คือ ระยะทางเป้าหมาย (X_1)
  เอาท์พุท คือ ระยะจริงของรถ (X)

ซึ่งส่วนนี้เป็นส่วนที่สามารถควบคุมได้ โดยมีเป้าหมายที่จะควบคุมมุมของก้านเพนดูลัมในส่วนที่ 1 ต่อไป เราสามารถควบคุมการเคลื่อนที่ของรถได้ แต่ไม่สามารถควบคุมมุมของก้านเพนดูลัมได้โดยตรง จึงใช้การเคลื่อนที่ของรถนี้มาควบคุมมุมของก้านเพนดูลัมให้ตั้งตรงตลอดเวลา



4.2 การหาทรานส์เฟอร์ฟังก์ชันของระบบ

- ส่วนที่ 1 วิเคราะห์ก้านลูกตุ้มสำหรับกรณี Simple Pendulum เพื่อหาค่าพารามิเตอร์ของระบบ



รูปที่ 4.1 การแกว่งลูกตุ้มแบบ Simple Pendulum

เมื่อ $D\dot{\theta}$ คือ Damper เชิงมุม

จาก $\sum F = ma$

จะได้ $mg(\sin\theta) + D\dot{\theta} = -ml\ddot{\theta}$

เมื่อวิเคราะห์ที่มุม θ น้อย ๆ ($\sin\theta \approx \theta$)

$$ml\ddot{\theta} + D\dot{\theta} + mg\theta = 0$$

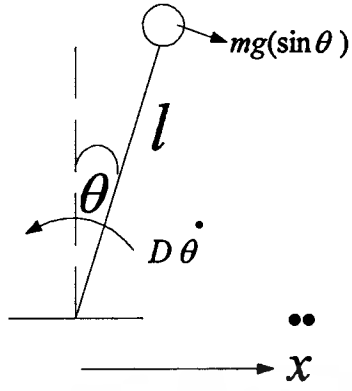
take Laplace ทั้งสมการได้ ($\dot{\theta}_0 = 0$)

$$ml[s^2\theta(s) - s\theta_0] + D[s\theta(s) - \theta_0] + mg\theta(s) = 0$$

$$(mls^2 + Ds + mg)\theta(s) = (mls + D)\theta_0$$

$$\theta(s) = \frac{(s + \frac{D}{ml})}{s^2 + (\frac{D}{ml})s + (\frac{g}{l})} \quad \theta_0$$

วิเคราะห์กรณี Inverted Pendulum โดยให้จุดหมุนของก้านลูกตุ้มเคลื่อนที่ด้วย



รูปที่ 4.2 การแกว่งลูกตุ้มแบบ Inverted Pendulum เมื่อมีการเปลี่ยนแปลงระยะทาง

จาก $\sum F = ma$

$$(mg\sin\theta) - D\dot{\theta} = m(\ddot{x} + l\ddot{\theta})$$

เมื่อวิเคราะห์ที่มุม θ น้อย ๆ ($\sin\theta \approx \theta$) มุมเริ่มต้นและความเร็วเชิงมุมเริ่มต้นเป็น 0

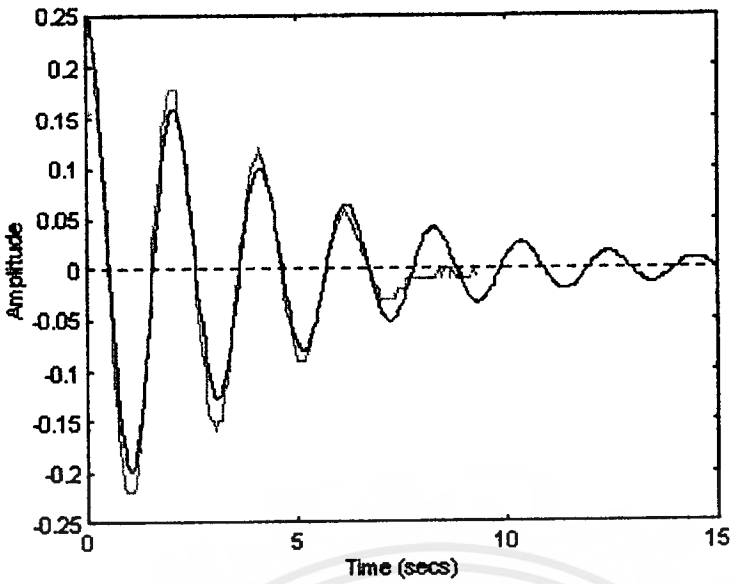
จะได้ $$\frac{\theta(s)}{X(s)} = -\frac{\frac{1}{l}s^2}{s^2 + (\frac{D}{ml})s - \frac{g}{l}}$$

จะเห็นว่าสมการคุณลักษณะ (Characteristic Equation) ที่ได้จากระบบ Simple Pendulum และ ระบบ Inverted Pendulum มีค่าตัวแปรเหมือนกัน แต่ต่างกันที่เครื่องหมายเท่านั้น

- ส่วนที่ 2 เป็นส่วนที่ได้จากการทดลองหาความสัมพันธ์ระหว่าง ระยะจริงของรถ (X) กับอินพุทคือค่า PWM รายละเอียดจะได้เสนอในหัวข้อต่อไป

4.3 การประมาณค่าโมเดลของระบบจากผลตอบสนองที่ได้จากการทดลอง

- ส่วนที่ 1 จากการทดลองแกว่งลูกตุ้มให้เคลื่อนที่อย่างอิสระแบบ Simple Pendulum จะได้ผลของค่ามุมเมื่อเวลาเปลี่ยนไป แสดงได้ดังรูป



รูปที่ 4.3 เปรียบเทียบกราฟของการแกว่งลูกตุ้ม

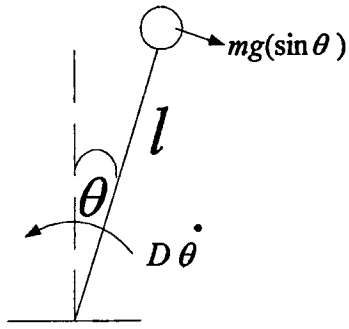
จากรูปที่ 4.3 แสดงกราฟของการแกว่งลูกตุ้มแบบ Simple Pendulum เปรียบเทียบกับกราฟของสมการ $\frac{\theta}{\theta_0} = \frac{s+0.44}{s^2+0.44s+9.1688}$ ซึ่งเป็นสมการที่หามาจากกราฟการแกว่งลูกตุ้มแบบ Simple Pendulum

เมื่อเปรียบเทียบกับสมการของระบบ Simple Pendulum $\frac{\theta(s)}{\theta_0(s)} = \frac{(s+\frac{D}{ml})}{s^2+(\frac{D}{ml})s+\frac{g}{l}}$

กับสมการของระบบอินเวิร์ทเพนดูลัม $\frac{\theta(s)}{X(s)} = -\frac{\frac{1}{l}s^2}{s^2+(\frac{D}{ml})s-\frac{g}{l}}$

จะได้สมการของก้านอินเวิร์ทเพนดูลัม คือ $\frac{\theta(s)}{X(s)} = -\frac{0.9346s^2}{s^2+0.44s-9.1688}$

ตรวจสอบสัมประสิทธ์ของสมการที่ได้โดยการทดลองปล่อยให้ลูกตุ้มตก เมื่อตั้งลูกตุ้มไว้ที่จุดสูงสุดแบบอินเวิร์ทเพนดูลัม



รูปที่ 4.4 การปล่อยให้ลูกตุ้มตกแบบอินเวิร์ทเพนดูลัม

จาก $\sum F = ma$

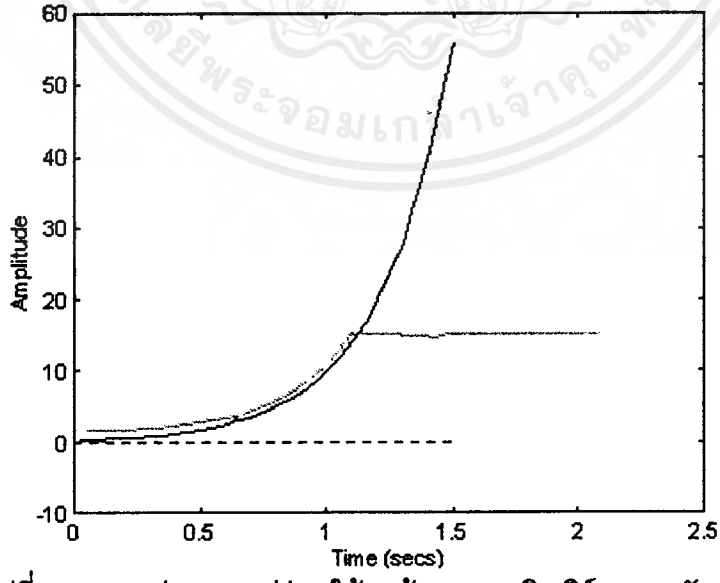
$$(mg\sin\theta) - D\dot{\theta} = ml\ddot{\theta}$$

จะได้

$$\frac{\theta(s)}{\theta_0(s)} = \frac{(s+\frac{D}{ml})}{s^2 + (\frac{D}{ml})s - \frac{g}{l}}$$

$$\therefore \quad \frac{\theta}{\theta_0} = \frac{s+0.44}{s^2+0.44s-9.1688}$$

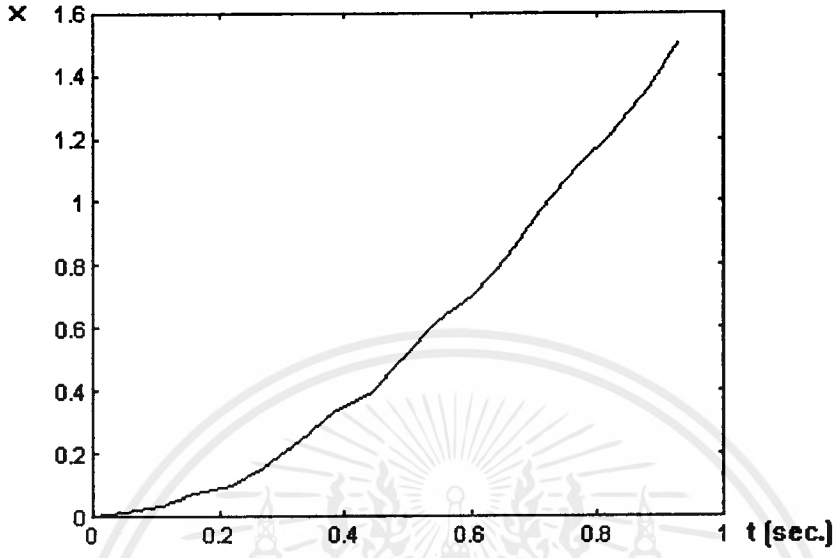
กราฟของการทดลองปล่อยลูกตุ้มแบบอินเวิร์ทเพนดูลัม กับสมการข้างต้น แสดงได้ดังรูป



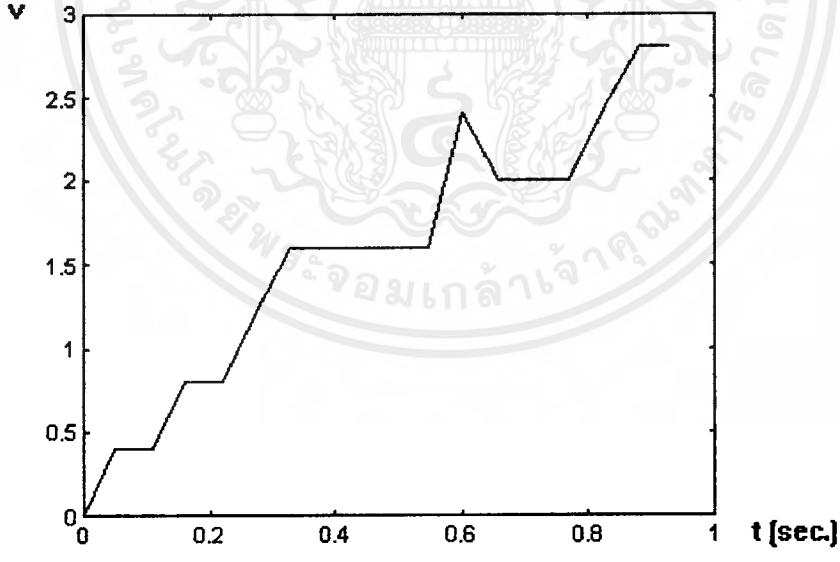
รูปที่ 4.5 กราฟของการปล่อยให้ลูกตุ้มตกแบบอินเวิร์ทเพนดูลัม

จากรูปจะเห็นได้ว่ากราฟที่ได้จากสมการมีความใกล้เคียงกับกราฟของการปล่อยลูกตุ้ม

- ส่วนที่ 2 เมื่อให้รถวิ่งด้วยความเร็วสูงสุด คือจ่ายค่า PWM เท่ากับ 255 ได้กราฟการเคลื่อนที่ของรถ คือ กราฟระยะทาง และกราฟความเร็ว ดังรูป



รูปที่ 4.6 กราฟระยะทางการเคลื่อนที่ของรถเมื่อ PWM=255



รูปที่ 4.7 กราฟความเร็วของรถเมื่อ PWM=255

จากกราฟของความเร็วสามารถประมาณกราฟเป็นแบบ First Order ได้

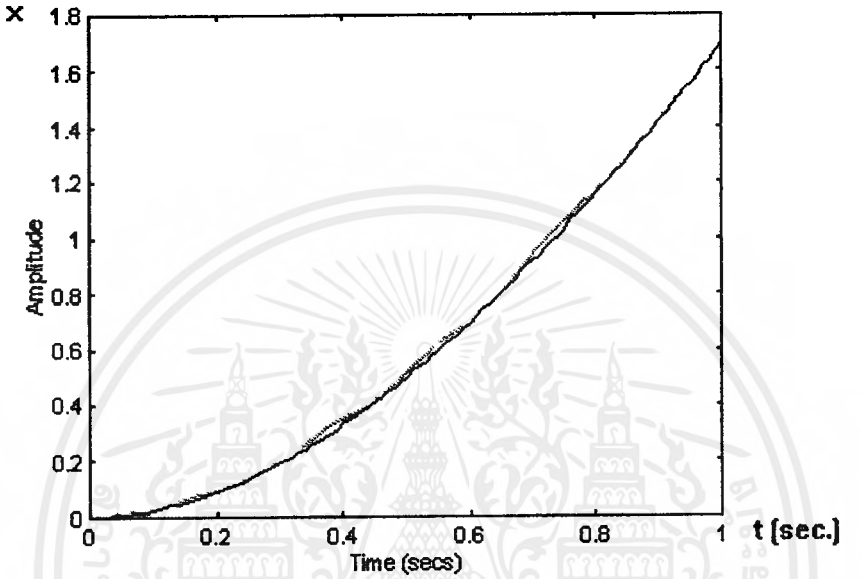
$$V(s) = \frac{K}{s+\alpha} \quad \bullet \quad PWM$$

จากความสัมพันธ์ของระยะทางและความเร็ว คือ $v = \dot{x}$

ได้ $X(s) = \frac{V(s)}{s}$

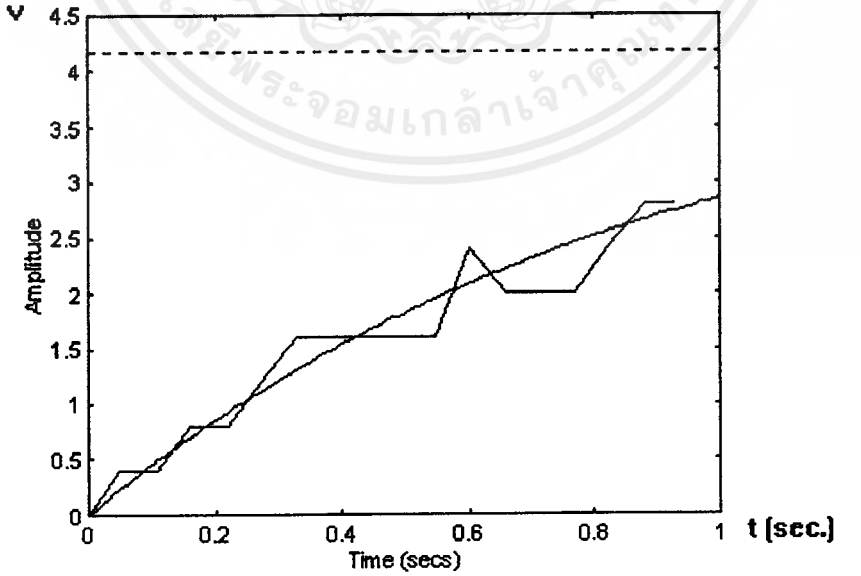
$X(s) = \frac{K}{s(s+\alpha)} \bullet PWM$ ซึ่งเป็นระบบ Second Order

จึงทำการประมาณกราฟของระยะทางข้างต้น ด้วยการแทนค่า K = 4.8 / 255 และ α = 1.15 ลงในสมการจะได้กราฟของระยะทาง ดังรูป



รูปที่ 4.8 กราฟระยะทางจากการแทนค่าตัวแปร

และทำการประมาณกราฟของความเร็วข้างต้น ด้วยการแทนค่า K = 4.8 / 255 และ α = 1.15 ลงในสมการจะได้กราฟของความเร็ว ดังรูป



รูปที่ 4.9 กราฟความเร็วจากการแทนค่าตัวแปร

เมื่อแทนค่าตัวแปรดังกล่าว จะได้ $\ddot{x} = \frac{4.8}{255} PWM - 1.5\dot{x}$

จากสมการของก้านอินเวิร์ทเพนดูลัม $\frac{\theta(s)}{X(s)} = -\frac{0.9346s^2}{s^2 + 0.44s - 9.1688}$

จะได้ $\ddot{\theta} + 0.44\dot{\theta} - 9.1688\theta = -0.9346\ddot{x}$

$$= -0.9346\left(\frac{4.8}{255} PWM - 1.5\dot{x}\right)$$

เขียนให้อยู่ในรูปเมตริกซ์ ได้ State Equation คือ

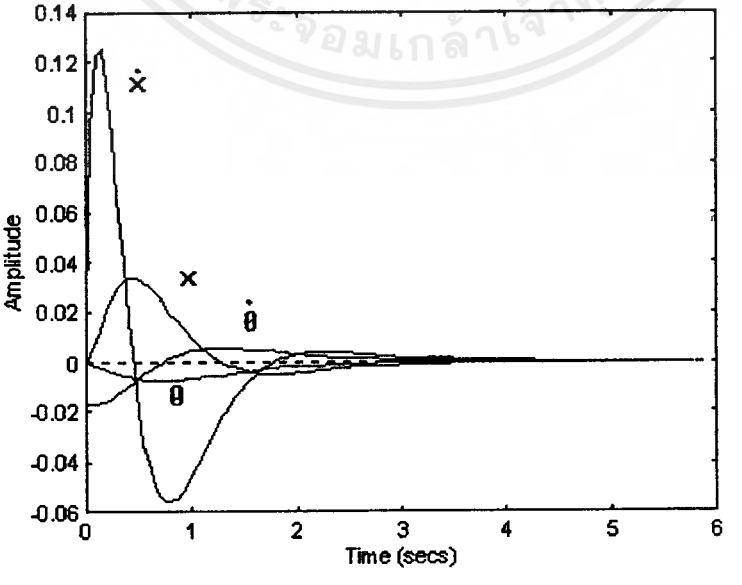
$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 9.1688 & -0.44 & 0 & 1.0748 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1.15 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -0.0176 \\ 0 \\ 0.0188 \end{bmatrix} PWM$$

นำค่าในเมตริกซ์ไปแทนในโปรแกรมหาค่าพารามิเตอร์ของระบบ ในภาคผนวก จ. โดยกำหนดโพลวงปิดเป้าหมายไว้ที่ s = -4 , -4 , -2.5 , -2.5 จะได้ค่า Feedback Gain (K) 4 ค่า คือ K = -4526.1 , -1509.3 , -579.4 , -781

จาก $\mathbf{U} = -\mathbf{KX}$

$$PWM = -\begin{bmatrix} -4526.1 & -1509.3 & -579.4 & -781 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix}$$

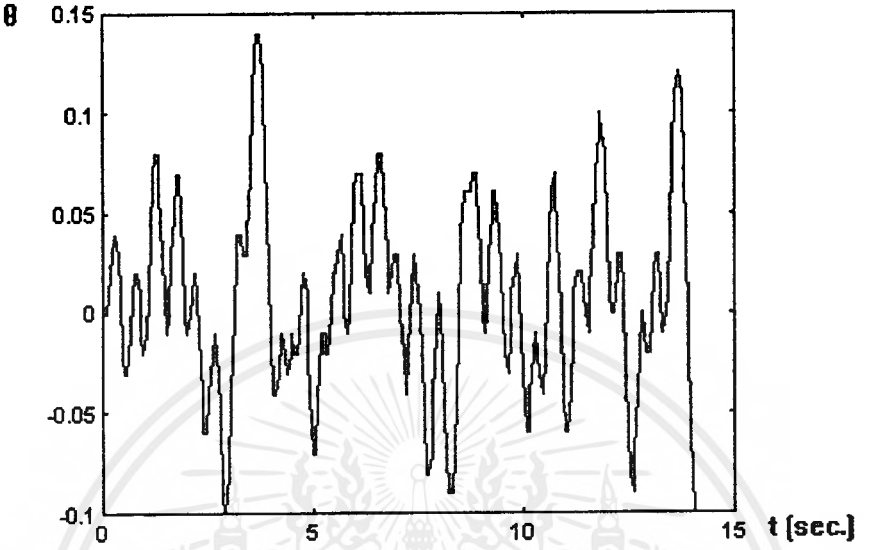
นำค่า PWM ไปแทนใน State Equation แล้วพล็อตกราฟผลตอบสนองของระบบได้ดังรูป



รูปที่ 4.10 กราฟผลตอบสนองของระบบ

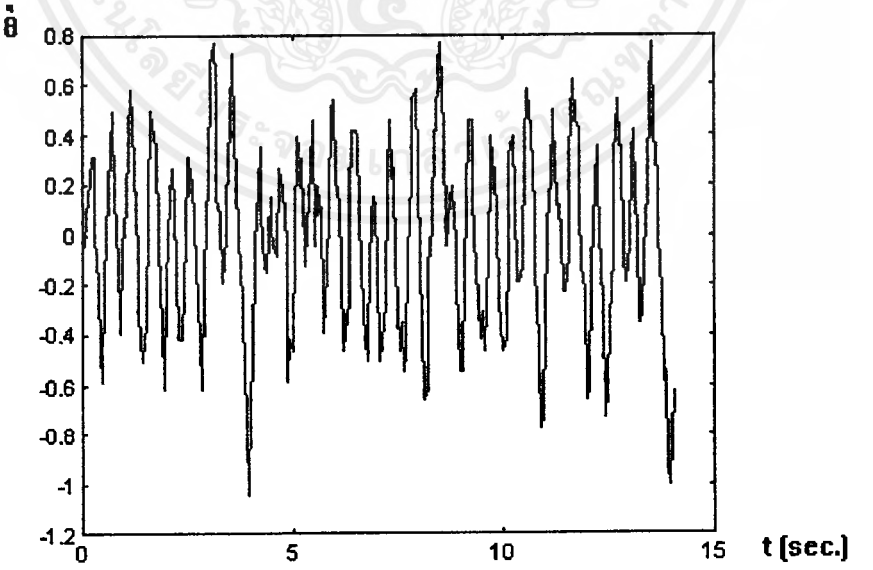
นำค่า K ที่ได้ไปใส่โปรแกรมควบคุมการเคลื่อนที่ของรถ ตามภาคผนวก ค. แล้วทดลองนำรถถมาวิ่งจริง ได้ผลการทดลองดังนี้

1. ค่ามุมของก้านเพนดูลัม (θ) เมื่อรถวิ่ง



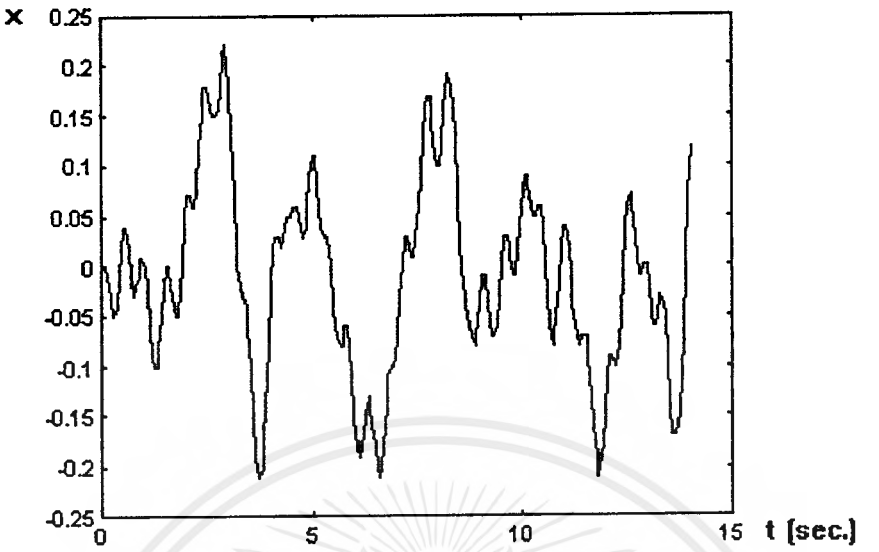
รูปที่ 4.11 ค่ามุมของก้านเพนดูลัมเมื่อรถวิ่ง

2. ค่าความเร็วเชิงมุมของก้านเพนดูลัม ($\dot{\theta}$) เมื่อรถวิ่ง



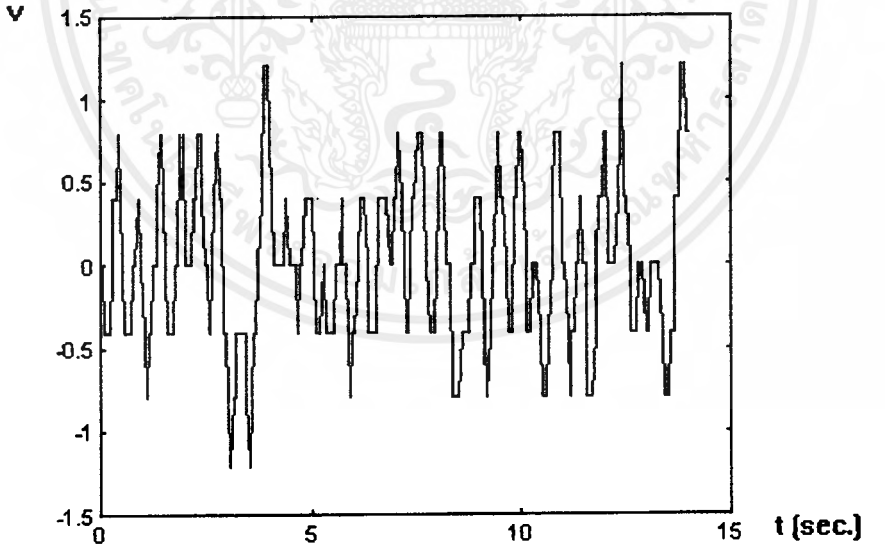
รูปที่ 4.12 ค่าความเร็วเชิงมุมของก้านเพนดูลัมเมื่อรถวิ่ง

3. ค่าตำแหน่งของรถ (x)



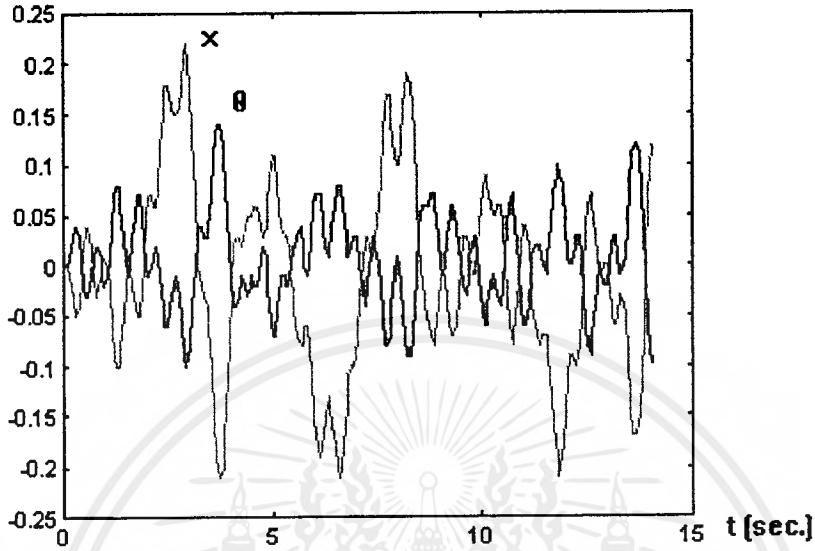
รูปที่ 4.13 ค่าตำแหน่งของรถ

4. ค่าความเร็วของรถ (v)



รูปที่ 4.14 ค่าความเร็วของรถ

5. แสดงเอาท์พุทของระบบ คือ ค่ามุมของก้านเพนดูลัม และตำแหน่งของรถ เมื่อเวลาเปลี่ยนไป



รูปที่ 4.15 เอาท์พุทของระบบ

บทที่ 5
สรุปผลและวิจารณ์

- **เป้าหมายที่ตั้งไว้และสิ่งที่ดำเนินการได้**

ตั้งเป้าหมายไว้ที่สามารถจะควบคุมทั้งตำแหน่งรถ และตำแหน่งของก้านลูกตุ้ม สิ่งที่สามารถทำได้คือ สามารถควบคุมก้านลูกตุ้มให้อยู่ในตำแหน่งสมดุลได้ แต่ ตำแหน่งรถยังเกิดการแกว่งในตำแหน่งเป้าหมายที่ตั้งไว้ ในขอบเขตค่าหนึ่งซึ่งสามารถยอมรับได้

- **ปัญหาและการแก้ไข**

- ปัญหาทาง Hardware
    - Potentiometer ยังมีค่าสัมประสิทธิ์ความฝืดมาก จึงส่งผลให้เกิดความไม่เป็นเชิงเส้นมากยิ่งขึ้นซึ่งผิดไปจากการประมาณให้เป็นเชิงเส้นที่กำหนดไว้ขั้นต้น
    - การ slip ระหว่างล้อกับพื้น และ ระหว่างล้อกับล้อต้นกำลัง จึงทำให้การชดเชยระบบของตัว Controller เป็นไปได้ช้า ทำให้เกิด Dead Band ขึ้นมา
    - Motor เป็น Motor ที่ไม่สมบูรณ์ คือเมื่อ apply voltage เข้าไปแล้วการหมุนทวนเข็มและตามเข็มความเร็วรอบไม่เท่ากัน จึงจำเป็นต้องชดเชย scale ของทั้งสองด้านให้เท่ากัน
- ปัญหาของตัว Controller
    - เดิมทีเราต้องการนำตัว Controller LM629 มาเป็นตัว Controller ในการควบคุมตำแหน่งขั้นต้น แต่เนื่องจากข้อจำกัดในการติดต่อกับตัว Controller หลัก (CPU) จึงทำให้ต้องตัดส่วน close loop ของตำแหน่งขั้นต้นตรงนี้ออกไป
- ปัญหาของตัว Software
    - sampling period ของตัว Controller เท่ากับ 18.2 Hz. ซึ่งช้ามากเพราะว่าการใช้สัญญาณ Interrupt จากภายนอกยังไม่สมบูรณ์ CPU ยังไม่บริหารทุกสัญญาณ จึงทำให้ต้องใช้สัญญาณ Interrupt จาก Timer control ภายในแทน
- ปัญหาของวงจร Driver
    - วงจร Drive มี Output Impedance ที่มีค่าไม่ต่ำเพียงพอ ทำให้เมื่อวงจร Drive มีการจ่ายกระแสขนาดสูง ๆ จะทำให้ voltage drop เกิดความไม่เป็นเชิงเส้นของวงจร

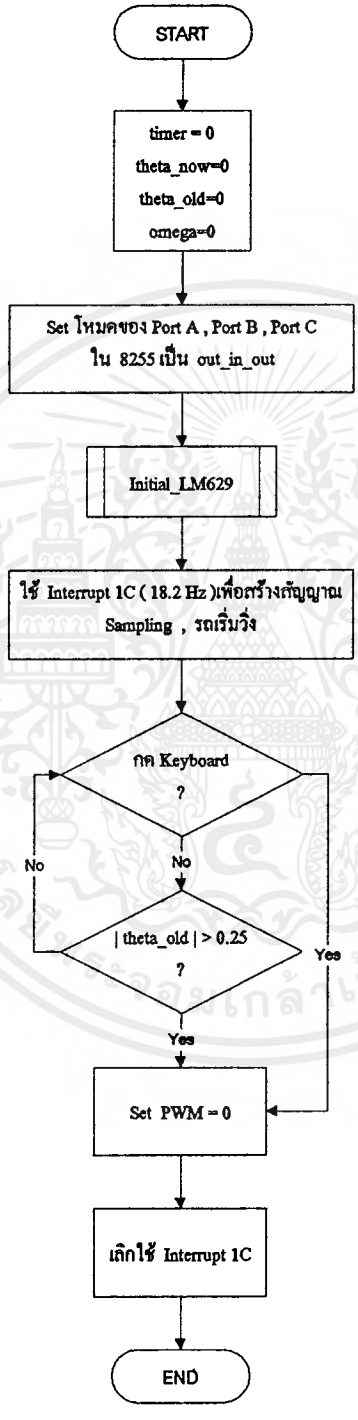
- วงจรจะเผชิญปัญหาการกลับขั้ว Motor อยู่ตลอดเวลาและมีความถี่สูง โดยคุณสมบัติของ Motor แล้ว เมื่อมีการกลับขั้วจะมีกระแสกระชากขนาดมาก และเมื่ออุปกรณ์ทำการ switching จะมี lost เกิดขึ้น ในช่วง on - off เมื่อรวมทั้งสองเหตุการณ์เข้าด้วยกัน จะทำให้เกิด lost มีค่ามากขึ้นเป็นทวีคูณ ส่งผลให้วงจร Driver ร้อนและพังได้

- **การพัฒนาในอนาคต**

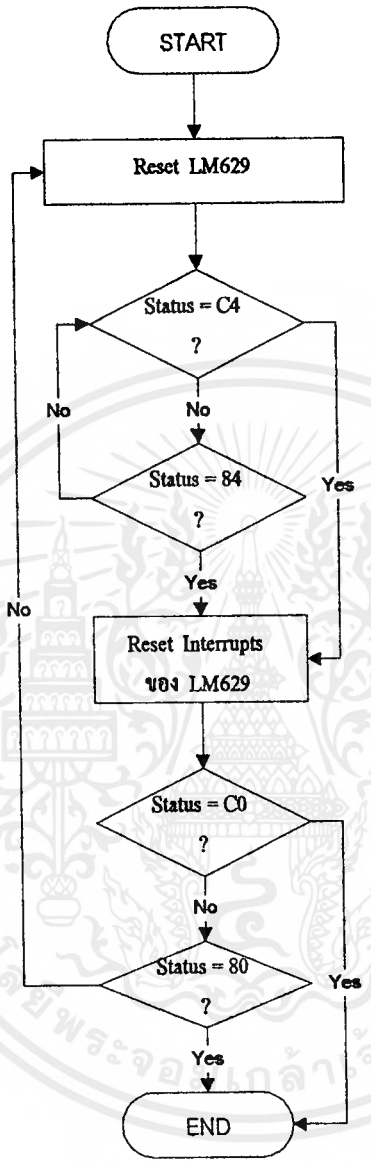
1. เพิ่ม sampling time ของระบบให้มากขึ้นจนเข้าใกล้ระบบ continuous
2. โดยลักษณะของวงจรมีลักษณะที่ Open คือสามารถขยายโดยเพิ่มก้านลูกตุ้มเข้าไปอีก
3. เปลี่ยนโพเทนทิโอมิเตอร์ที่ใช้อยู่ให้มีความผิดน้อยลง เพื่อการควบคุมระบบให้มีเสถียรภาพมากขึ้น
4. จัดทำตัวรถให้ไม่มีการสลิปของล้อและใช้มอเตอร์ที่มีคุณภาพสูงกว่าตัวที่ใช้อยู่

ภาคผนวก

ภาคผนวก ก
Flow Chart โปรแกรมควบคุมการเคลื่อนที่ของรถ



รูปที่ ก-1 Main Program



** หมายเหตุ Status : BIT 7 = Motor off

6 = Breakpoint Reached (Interrupt)

5 = Excessive Position Error (Interrupt)

4 = Wrap-around Occurred (Interrupt)

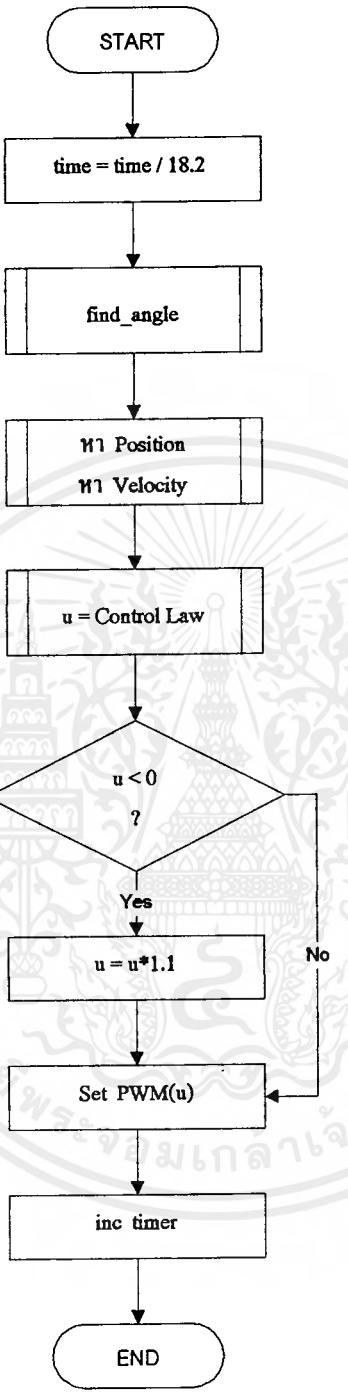
3 = Index Pulse Observed (Interrupt)

2 = Trajectory Complete (Interrupt)

1 = Command Error (Interrupt)

0 = Busy-bit

รูปที่ ก-2 Initial_LM629

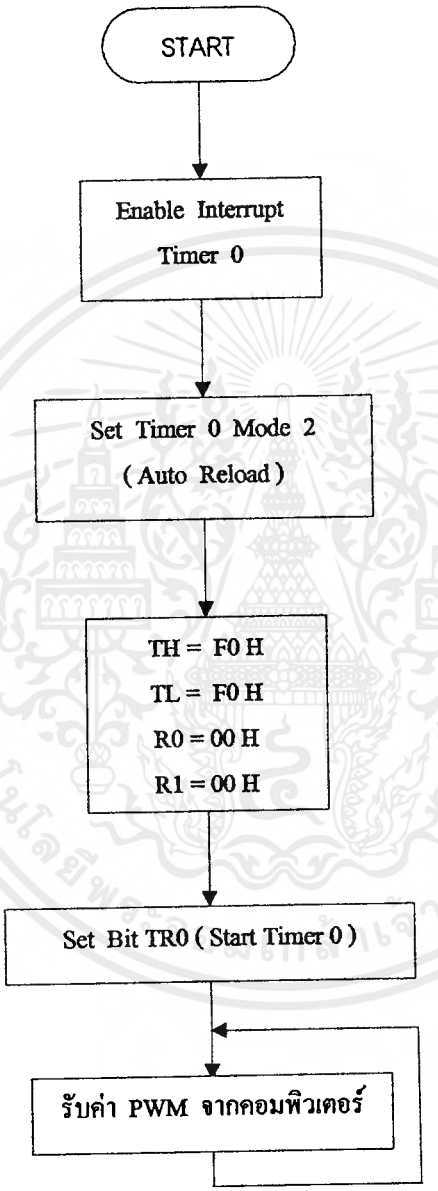


รูปที่ ก-3 Interrupt Service Routine

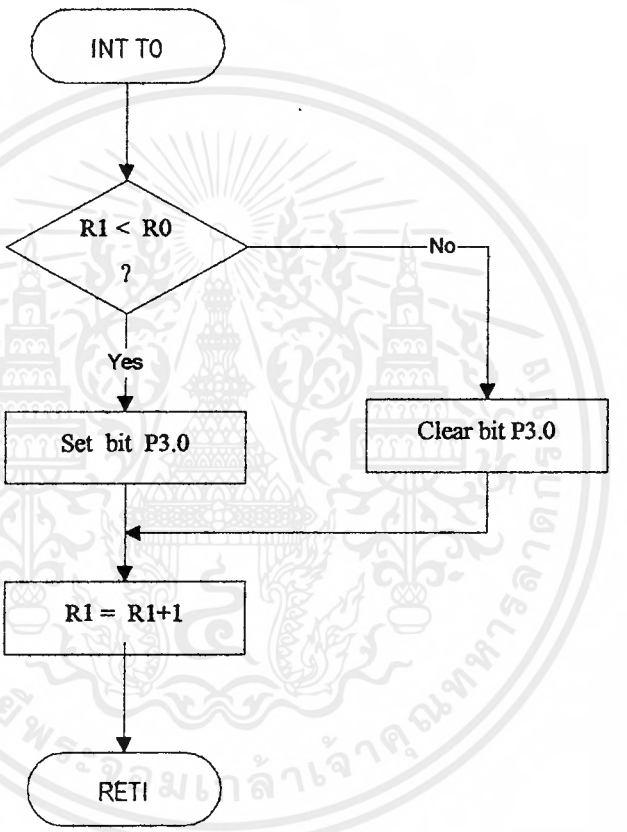
รูปที่ ก-4 Find_angle

ภาคผนวก ข

Flow Chart โปรแกรมกำเนิด PWM



รูปที่ ข-1 Main Program สร้าง PWM



รูปที่ ข-2 Interrupt Timer 0 Service Routine

ภาคผนวก ค
โปรแกรมควบคุมการเคลื่อนที่ของรถ

โปรแกรมการควบคุมการเคลื่อนที่ของรถสำหรับโครงงานนี้ใช้ภาษาปาสคาลในการเขียนโปรแกรม รายละเอียดของโปรแกรมแสดงได้ดังนี้

```
PROGRAM  Inverted_Pendulum_Cart;
uses   crt , dos , lm629_and_89C2051 , adc0848;
const
      k1      = 4526.1;
      k2      = 1509.3;
      k3      = 579.4;
      k4      = 781;
 var    timer     : longint;
        time , theta_old , theta_now , omega , position , velocity , zero , u : real;
        pwm       : integer;
        s         : string;
        f         : text;
        IntNo     : byte;
        Regs      : Registers;
        IntSave   : pointer;
       Chr        : char;
 procedure  find_angle;
       begin
          low ( CS );
          theta_now := -(read_ADC(1)*30.96/255)*3.14/180+zero;
          high ( CS );
          omega := ( theta_now-theta_old )*18.2;
          theta_old := theta_now;
       end;
 function     control_law ( x1 , x2 , x3 , x4 : real ) : real;
       begin
```

```
        control_law := k1*x1+k2*x2+k3*x3+k4*x4;
    end;
procedure  IRQ1c_handler ( regs : Registers ); interrupt;
    begin
        time := timer/18.2;
        find_angle;
        position := real_position;
        velocity := real_velocity;
        str ( time : 10 : 2 , s );          write ( f , s );
        str ( theta_now : 0 : 2 , s );      write ( f , s );
        str ( omega : 10 : 2 , s );         write ( f , s );
        str ( position : 10 : 2 , s );      write ( f , s );
        str ( velocity : 10 : 2 , s );      writeln ( f , s , ' ; ' );
        u := control_law ( theta_now , omega , position , velocity );
        if u < 0 then u := 1.1*u;
        pwm := round ( u );
        set_pwm ( pwm );
        gotoxy ( 1 , 1 );
        writeln ( '                    ');
        writeln ( '                    ');
        writeln ( '                    ');
        writeln ( '                    ');
        writeln ( '                    ');
        gotoxy ( 1 , 1 );
        writeln ( ' pwm       = ' , pwm );
        writeln ( ' control_law  =  ' , u : 5 : 3 );
        writeln ( ' theta          = ' , theta_now : 5 : 3 );
        writeln ( ' omega        = ' , omega : 5 : 3 );
        writeln ( ' x              = ' , position : 5 : 3 );
        writeln ( ' v              = ' , velocity : 5 : 3 );
        inc ( timer );
    end;
```

```
begin  {main}
    assign ( f , ' c:\matlab\cart.m ' );
    rewrite ( f );
    writeln ( f , ' m = [ ' );
    clrscr;
    timer      := 0;
    theta_now := 0;
    theta_old  := 0;
    omega      := 0;
    u          := 0;
    IntNo      := $1c;
    Port [ write_control_words ] := out_in_out;
    Port [ port_c ] := $ff;
    zero :=  (read_ADC(1)*30.96/255)*3.14/180
    GetIntVec ( IntNo , IntSave );
    Initial_LM629;
    SetIntVec ( IntNo , addr ( IRQ1c_handler ) );
    Repeat   until   keypressed   or ( abs ( theta_old ) > 0.25 );
    set_PWM ( 0 );
    SetIntVec ( IntNo , IntSave );
    Writeln ( f , ' ] ; ' );
    Writeln ( f , ' c1 = [ 1; 0; 0; 0; 0; ] ; ' );
    Writeln ( f , ' c2 = [ 0; 1; 0; 0; 0; ] ; ' );
    Writeln ( f , ' c3 = [0; 0; 1; 0; 0; ] ; ' );
    Writeln ( f , ' c4 = [0; 0; 0; 1; 0; ] ; ' );
    Writeln ( f , ' c5 = [ 0; 0; 0; 0; 1; ] ; ' );
    Writeln ( f , ' time = m*c1 ; ' );
    Writeln ( f , ' theta = m*c2 ; ' );
    Writeln ( f , ' omega = m*c3 ; ' );
    Writeln ( f , ' x = m*c4 ; ' );
    Writeln ( f , ' v = m*c5 ; ' );
    Close ( f );
END.
```

ส่วนของ UNIT ซึ่งเป็นส่วนที่โปรแกรมหลักเรียกใช้ UNIT ที่ใช้ติดต่อกับ ADC0848 มีรายละเอียดดังนี้

```
UNIT   ADC0848;

INTERFACE
uses   crt , dos;
const
    {address to select port of 8255 phototypeboard.}
    port_a = $300;
    port_b = $301;
    port_c = $302;
    write_control_words = $303;
    {control words to select port}
     {xxAxxBxxC}
    out_out_out = $80;
    out_in_out  = $82;      {use in this photocard}
    in_out_out  = $90;
    in_in_out   = $92;

var   channel : integer;

FUNCTION   read_ADC ( channel : integer ) : integer;
PROCEDURE   init_8255;

IMPLEMENTATION

        PROCEDURE   Low ( command : integer );
        BEGIN
        Port [ port_c ] := port [ port_c ] and not command;
        END;
```

```
    PROCEDURE   High ( command : integer );
    BEGIN
    Port [ port_c ] := port [ port_c ] or command;
    END;

function   read_ADC ( channel : integer ) : integer;
const   _wr  = $dc;          {write active + latch active}
        _rd  = $ed;          {read active + latch not active}
        _clr = $ff;          {non active}
begin
channel := channel+7;
port [ port_a ] := channel;
low ( $23 );          {port [ port_c ] := _wr;}
high ( $23 );         {port [ port_c ]:= _clr;}
delay ( 1 );
low ( $12 );          {port [ port_c ]:= _rd;}
read_ADC := port [ port_b ];
high ( $12 );         {port [ port_c ] := _clr;}
end;

procedure    init_8255;
begin
port [ write_control_words ] := out_in_out;
end;

END.
```

UNIT ที่ใช้ติดต่อกับ LM629 และ 89C2051 มีรายละเอียดดังนี้

```
UNIT   LM629_and_89C2051;

INTERFACE
uses   crt , dos , graph;
const
    {PulsePerMetre}
    PPM : real = 18.34{rev/m}*400;     {pulse/rev}
    {629 Sample Period}
    T629 : real    = 2048/6e6;
    coef_P : real  = 18.34*400;          {=PPM}
    coef_V : real  = 2048/6e6*65536*18.34*400;          {T629*65536*PPM}
    coef_A : real  = 2048/6e6*2048/6e6*65536*18.34*400;  {T629*T629*65536*PPM}
     {Control bit for lm629 from 8255}
    Direction   = $40;
    WR_         = $20;
    RD_         = $10;
    PS_         = $08;
    RST_        = $04;
    CS          = $02;
    dataout_    = $01;

    {address to select port of 8255 phototypeboard.}
    port_a = $300;
    port_b = $301;
    port_c = $302;
    write_control_words = $303;
    {control words to select port}
     {xxAxxBxxC}
    out_out_out = $80;
    out_in_out  = $82;
```

```
    in_out_out   = $90;
    in_in_out    = $92;

Var
      Highbyte , Lowbyte : integer;
      r_pos , r_vel      : longint;
      d_pos , d_vel      : longint;
      Kp , Ki , Kd       : longint;
      Acceleration , Velocity , Position : real;

FUNCTION   Dec2Hex ( dec : integer ) : string;
FUNCTION   Dec2Bin  ( dec : integer)  : string;
PROCEDURE Low ( command : integer );
PROCEDURE High ( command : integer );
FUNCTION   Status : integer;
      { BIT 7 = Motor off
            6 = Breakpoint Reached (Interrupt)
            5 = Excessive Position Error (Interrupt)
            4 = Wrap-around Occurred (Interrupt)
            3 = Index Pulse Observed (Interrupt)
            2 = Trajectory Complete (Interrupt)
            1 = Command Error (Interrupt)
            0 = Busy-bit  }
PROCEDURE  Busy_Check;
PROCEDURE  Write_Command ( command : integer );
PROCEDURE  Read_Data;
PROCEDURE  Write_Data ( Highbyte , Lowbyte : integer );
PROCEDURE  Reset_Interrupt ( Lowbyte : integer );
      { BIT  7 =  not  used
            6 = Breakpoint Reached Interrupt
            5 = Excessive Position Error Interrupt
            4 = Wrap-around Occurred Interrupt
            3 = Index Pulse Observed Interrupt
```

```
                2 = Trajectory Complete Interrupt
                1 = Command Error Interrupt
                0 = not used  }
 PROCEDURE  Initial_LM629;
 PROCEDURE  Filter_Module ( Kp , Ki , Kd : integer );
 PROCEDURE  Init_Acceleration ( acceleration : real );
 PROCEDURE  Set_Velocity ( velocity : real );
 PROCEDURE  Set_Velocity_Rel ( velocity : real );
 PROCEDURE  SetBreakPoint ( b : real );
 PROCEDURE  Trajectory_Module ( acceleration , velocity , position : real );
 PROCEDURE  Stop_Module;
 PROCEDURE  Start_Module;
 FUNCTION      Real_Position : real;
 FUNCTION      DESIRED_POSITION : real;
 FUNCTION      REAL_VELOCITY : real;
 FUNCTION      Desired_Velocity : real;
 PROCEDURE  OutText ( elapsed_time : real );
 PROCEDURE  Set_PWM ( duty : integer );


 IMPLEMENTATION
 FUNCTION     Dec2Hex ( dec : integer ) : string;
 var      hex2 , hex1 :  integer;
 const   a : array [ 0..15 ] of char
              = ( '0','1','2','3','4','5','6','7'
                 ,'8','9','A','B','C','D','E','F' );
 BEGIN
      hex1 := dec mod 16;
      hex2 := dec div 16;
      Dec2Hex := a [ hex2 ] + a [ hex1 ];
 END;

 FUNCTION     Dec2Bin ( dec : integer ) : string;
 var     bin2 , bin1 : integer;
```

```
const   a : array [ 0..15 ] of string
        = ( '0000','0001','0010','0011','0100','0101','0110','0111'
          ,'1000','1001','1010','1011','1100','1101','1110','1111' );
BEGIN
    bin1 := dec mod 16;
    bin2 := dec div 16;
    Dec2Bin := a [ bin2 ] + ' ' + a [ bin1 ];
END;


PROCEDURE   Low  ( command : integer );
BEGIN
    Port [ port_c ] := port [ port_c ] and not command;
END;

PROCEDURE High ( command : integer );
BEGIN
    Port [ port_c ] := port [ port_c ] or command;
END;

FUNCTION    Status : integer;
BEGIN
    Low ( PS_ );
    Low ( RD_ );
    Status := port [ port_b ];
    High ( RD_ );
END;

PROCEDURE   Busy_Check;
var    busy : boolean;
BEGIN
    Low ( PS_ );
    Low ( RD_ );
```

```
    repeat
        if status mod 2 =1 then busy := true else busy := false;
    until  busy = false;
    High ( RD_ );
END;

PROCEDURE   Write_Command ( command : integer );
BEGIN
    Low ( PS_ );
    Port [ port_a ] := command;
    Low ( dataout_ );
    Low ( WR_ );
    High ( WR_ );
    High ( dataout_ );
    Busy_Check;
END;

PROCEDURE   Read_Data;
BEGIN
    High ( PS_ );
    Low ( RD_ );
    Highbyte := port [ port_b ];
    High ( RD_ );
    Low ( RD_ );
    Lowbyte := port [ port_b ];
    High ( RD_ );
    Busy_Check;
END;

PROCEDURE   Write_Data ( Highbyte , Lowbyte : integer );
BEGIN
    High ( PS_ );
    Port [ port_a ] := Highbyte;
```

```
    Low ( dataout_ );
    Low ( WR_ );
    High ( WR_ );
    Port [ port_a ] := Lowbyte;
    Low ( WR_ );
    High ( WR_ );
    High ( dataout_ );
    Busy_Check;
END;

PROCEDURE   Reset_Interrupt ( Lowbyte : integer );
BEGIN
    Write_Command ( $1d );
    Write_Data ( $00 , Lowbyte );
END;

FUNCTION    Real_Position : real;
VAR    p : longint;
BEGIN
      Write_Command ( $0a );
      Read_Data;
      P := Highbyte*256+Lowbyte;
      Read_Data;
      P := ( p*256+Highbyte )*256+Lowbyte;
      Real_Position := p/coef_P;
END;

FUNCTION    DESIRED_POSITION : real;
VAR    p : longint;
BEGIN
      Write_Command ( $08 );
      Read_Data;
      P := Highbyte*256+Lowbyte;
```

```
    Read_Data;
    P := ( p*256+Highbyte )*256+Lowbyte;
    Desired_Position := p/coef_P;
END;

FUNCTION   REAL_VELOCITY : real;
VAR    v : integer;
BEGIN
    Write_Command ( $0B );
    Read_Data;
    V := Highbyte*256+Lowbyte;
    Real_Velocity := v/coef_v*65536;
END;

FUNCTION   Desired_Velocity : real;
VAR    v : longint;
BEGIN
    Write_Command ( $07 );
    Read_Data;
    V := Highbyte*256+Lowbyte;
    Read_Data;
    V := ( v*256+Highbyte )*256+Lowbyte;
    Desired_Velocity := v/coef_v;
END;

PROCEDURE Initial_LM629;
BEGIN
  repeat
    repeat
      Low ( RST_ );
      Delay ( 1 );
      High ( RST_ );
      Delay ( 2 );
```

```
        Status;
    until ( status = $c4 ) or ( status = $84 );
    Reset_Interrupt ( $00 );
  until ( status = $c0 ) or ( status = $80 );
END;

PROCEDURE Filter_Module ( Kp , Ki , Kd : integer );
BEGIN
  Write_Command ( $1e );      {LFIL Command}
  Write_Data ( $00 , $0e );
  Write_Data ( Kp and $FF00 , Kp and $FF );
  Write_Data ( Ki and $FF00 , Ki and $FF );
  Write_Data ( Kd and $FF00 , Kd and $FF );
  Write_Command ( $04 );      {UDF Command}
END;

PROCEDURE  Init_Acceleration ( acceleration : real );
var   A : longint;
BEGIN
  A := round ( acceleration*coef_a );
  Write_Command ( $1f );      {LTRJ Command}
  Write_Data ( $18 , $20 );
  Write_Data ( A and $ff000000 shr 24 , A and $ff0000 shr 16 );
  Write_Data ( A and $ff00 shr 8 , A and $ff );
  Write_Command ( $01 );
END;

PROCEDURE Set_Velocity ( velocity : real );
var   V : longint;
BEGIN
  V := round ( velocity*coef_v );
  Write_Command ( $1f );      {LTRJ Command}
  if  v>= 0
```

```
    then  Write_Data ( $18 , $08 )
    else  Write_Data ( $08 , $08 );
  V := abs ( V );
  Write_Data ( V and $ff000000 shr 24 , V and $ff0000 shr 16 );
  Write_Data ( V and $ff00 shr 8 , V and $ff );
  Write_Command ( $01 );
END;

PROCEDURE  Set_Velocity_Rel ( velocity : real );
var  V : longint;
BEGIN
  V := round ( velocity*coef_v );
  Write_Command ( $1f );    {LTRJ Command}
  Write_Data ( $18 , $0c );
  Write_Data ( V and $ff000000 shr 24 , V and $ff0000 shr 16 );
  Write_Data ( V and $ff00 shr 8 , V and $ff );
  Write_Command ( $01 );
END;

PROCEDURE  SetBreakPoint ( b:real );
var    bp : longint;
BEGIN
  Bp := round ( b*coef_p );
  Write_command ( $20 );          {SPBA}
  Write_Data ( bp and $FF000000 shr 24 , bp and $FF0000 shr 16 );
  Write_Data ( bp and $FF00 shr 8 , bp and $FF );
END;

PROCEDURE  Trajectory_Module ( acceleration , velocity , position : real );
var    A , V , P : longint;
BEGIN
  A := round ( Acceleration*coef_a );
  V := round ( Velocity*coef_v );
```

```
  P := round ( Position*coef_p );
  Write_Command ( $1f );          {LTRJ Command}
  Write_Data ( $00 , $2a );
  Write_Data ( A and $ff000000 shr 24 , A and $ff0000 shr 16 );
  Write_Data ( A and $ff00 shr 8 , A and $ff );
  Write_Data ( V and $ff000000 shr 24 , V and $ff0000 shr 16 );
  Write_Data ( V and $ff00 shr 8 , V and $ff );
  Write_Data ( P and $ff000000 shr 24 , P and $ff0000 shr 16 );
  Write_Data ( P and $ff00 shr 8 , P and $ff );
  Write_Command ( $01 );
END;

PROCEDURE  Stop_Module;
BEGIN
  Write_Command ( $1f );          {LTRJ Command}
  Write_Data ( $01 , $00 );
  Write_Command ( $01 );          {STT Command}
END;
PROCEDURE  Start_Module;
BEGIN
  Write_Command ( $01 );          {STT Command}
END;
PROCEDURE  Set_PWM ( duty : integer );
begin
    if  duty >= 0
      then  High ( Direction )
      else  Low ( Direction );
    if abs ( duty ) > 255 then duty := 255;
    port [ port_a ] := abs ( duty );
    Low ( $81 );
    High ( $81 );
end;
END.
```

ภาคผนวก ง
โปรแกรมกำเนิด PWM

โปรแกรมกำเนิด PWM เพื่อป้อนให้กับวงจรขับมอเตอร์ที่เขียนลงในไมโครคอนโทรลเลอร์ 89C2051 เป็นโปรแกรมภาษาแอสเซมบลี รายละเอียดของโปรแกรมแสดงได้ดังนี้

```
        ORG     0000H
        JMP     START
;**********************timer 0 interrupt*****************************
        ORG     000BH           ;timer 0 interrupt
SERVICE:                        ;port3.0 is PWM output
        MOV     A,R1
        CLR     C
        SUBB    A,R0
        JC      ON
OFF:    CLR     P3.0
        INC     R1
        RETI
ON:     SETB    P3.0
        INC     R1
        RETI
;
;**********************initial mcs51*****************************
;
START:  MOV     IE,#10000010B       ;initial timer0
        MOV     TMOD,#00000010B     ;timer0 mode 2
        MOV     TH0,#0F0H           ;autoreload 16 clock.
        MOV     TL0,#0F0H
        MOV     R0,#00H
```

```
        MOV     R1,#00H
        SETB    TR0                     ;start timer
;
;*******************wait command loop*************************
;
LOOP:   MOV     R0,P1                   ;R0 is Buffer from data bus
        JMP     LOOP
;
;*************************************************************
;
    END
```

ภาคผนวก จ
โปรแกรมหาค่าพารามิเตอร์

โปรแกรมหาค่าพารามิเตอร์นี้ใช้หาค่า Feedback Gain (K) ที่ป้อนกลับให้ระบบตามหลักการ Control Law โปรแกรมนี้ใช้ Run ใน MATLAB โดยสามารถกำหนดเลือกค่าโพลของระบบได้ตามต้องการ รายละเอียดของโปรแกรมแสดงได้ดังนี้

```
%*************************************************************************
%                PROGRAM TO DETERMINE  COEFFICIENT 'K'
%                        OF  CONTROL LAW
%*************************************************************************
% define coefficient of system
%       .
%       X = A X  +  B u
%
%       Y = C X  +  D u
%       u = - K X          ➔    Control Law
%
%*************************************************************************
A = [   0       1       0       0;
        9.1688  - 0.44  0       2.7938;
        0       0       0       1;
        0       0       0       -1.875];
B = [   0;      -0.0176;  0;     0.0188];
C = [   1       0       0       0;
        0       0       1       0];
D=[ 0 ];
%*************************************************************************
%determine eigen value
        eigen       = eig(A);
        first_eig   = [1 0 0 0]*eigen;
        second_eig  = [0 1 0 0]*eigen;
```

```
        third_eig      = [0 0 1 0]*eigen;
        forth_eig      = [0 0 0 1]*eigen;
%check eigen value be real or complex conjugate
        if  imag(first_eig)
                char_1 = [ 1 - ( first_eig+sen_eig )  (first_eig*second_eig') ];
        else
                char_1= conv ( [ 1 -first_eig ] , [ 1 -second_eig ] );      end;
        if  imag ( third_eig )
                char_2 = [ 1 - ( third_eig+forth_eig )  ( third_eig*forth_eig ) ];
        else
                char_2 = conv ( [ 1 -third_eig ] , [ 1 -forth_eig ] );      end;
        char = conv ( char_1 , char_2 );

%transfer to controllable canonical form
%       X = T Z
    %transfer matrix is T = Uc*W
        Uc      = [ B      A*B      A*A*B      A*A*A*B];
    %determine W
        row1    = char*   [0 0 0 1;
                           0 0 1 0;
                           0 1 0 0;
                           1 0 0 0;
                           0 0 0 0];
        row2    = char*   [0 0 1 0;
                           0 1 0 0;
                           1 0 0 0;
                           0 0 0 0;
                           0 0 0 0];
        row3    = char*   [0 1 0 0;
                           1 0 0 0;
                           0 0 0 0;
                           0 0 0 0;
                           0 0 0 0];
```

```
        row4    = char*   [1 0 0 0;
                           0 0 0 0;
                           0 0 0 0;
                           0 0 0 0;
                           0 0 0 0];
        W = [ row1;row2;row3;row4 ];
%confirm to find transform matrix ( T )
%       X = T Z
%It's will be
%       .   ~       ~
%       Z = A Z  +  B u
%           ~       ~
%       Y = C Z  +  D u

        T =  Uc*W;
        T1 = inv ( T );

                %test system  in controllable form
                        %test_A = T1*A*T;
                        %test_B = T1*B;

%*************************************************************************
%use pole placement to determine control law.
%declare the target system
        %if want to tune with target pole
target_pole             = [ -4 ; -4 ; -2.5 ; -2.5 ]; % <<========== Variable
first_target_pole       = [1 0 0 0]*target_pole;
second_target_pole = [0 1 0 0]*target_pole;
third_target_pole       = [0 0 1 0]*target_pole;
forth_target_pole       = [0 0 0 1]*target_pole;
    if  imag ( first_target_pole )
        char_1 = [ 1 - ( first_target_pole+second_target_pole ) (first_target_pole*second_target_pole ) ];
    else
```

```
        char_1 = conv ( [ 1 -first_target_pole ] , [ 1 -second_target_pole ] );    end;
    if  imag ( third_target_pole )
        char_2 = [1 - ( third_target_pole+forth_target_pole )  ( third_target_pole*forth_target_pole ) ];
    else
        char_2 = conv ( [ 1 - third_target_pole ] , [ 1 -forth_target_pole ] );     end;
    target_char = conv ( char_1 , char_2 );

%target_char = conv ( conv ( [ 1 -first_target_pole ] , [ 1 -second_target_pole ] ) , conv ( [ 1 -
third_target_pole ] , [ 1 -forth_target_pole ] ) );
target_A = [      0                1                0                0;
                  0                0                1                0;
                  0                0                0                1;
-target_char*[0;0;0;0;1] -target_char*[0;0;0;1;0] -target_char*[0;0;1;0;0] -target_char*[0;1;0;0;0]];

 %retransform in X parameter
        K_dat = [0 0 0 1]*( A-target_A );
        K     = K_dat*T1

%plot settling system
impulse ( target_A , B , [ 1 0 0 0 ] , D , 1 , 0 : 0.05 : 6 );  hold;
impulse ( target_A , B , [ 0 1 0 0 ] , D , 1 , 0 : 0.05 : 6 );
impulse ( target_A , B , [ 0 0 1 0 ] , D , 1 , 0 : 0.05 : 6 );
impulse ( target_A , B , [ 0 0 0 1 ] , D , 1 , 0 : 0.05 : 6 );
hold;
```

ภาคผนวก ฉ
คำสั่งแสดงกราฟเอาท์พุทของระบบ

เป็นชุดคำสั่งในโปรแกรม MATLAB เพื่อแสดงกราฟของมุมของก้านเพนดูลัม ความเร็วเชิงมุมของก้านเพนดูลัม ตำแหน่งของรถและความเร็วของรถ เมื่อเวลาเปลี่ยนไป โดยโปรแกรมหลักที่ควบคุมการเคลื่อนที่ของรถจะสร้างเมตริกซ์ m จากค่าตัวแปรในโปรแกรมคือ time , theta_now , omega , position และ velocity เรียงตามคอลัมน์ เมื่อรันโปรแกรมใน MATLAB จะได้กราฟ 5 กราฟตามที่แสดงไว้ในรูปที่ 4.11 - 4.15

```
m = [
 0.00   0.00   0.00   0.00   0.00;
 0.05  -0.00  -0.04   0.00   0.00;
 0.11   0.00   0.08  -0.00  -0.40;
 0.16   0.01   0.15  -0.01  -0.40;
 0.22   0.03   0.31  -0.03  -0.40;
 0.27   0.04   0.31  -0.05   0.00;
 0.33   0.04   0.00  -0.05   0.40;
 0.38   0.03  -0.19  -0.04   0.40;
 0.44   0.01  -0.46  -0.01   0.80;
 0.49  -0.02  -0.58   0.02   0.40;
 0.55  -0.03  -0.19   0.04   0.00;
 0.60  -0.03   0.00   0.04  -0.40;
 0.66  -0.01   0.35   0.02  -0.40;
 0.71   0.01   0.50  -0.01  -0.40;
 0.77   0.02   0.15  -0.03   0.00;
 0.82   0.02   0.04  -0.02   0.00;
 0.88   0.01  -0.31  -0.01   0.40;
 0.93  -0.01  -0.39   0.01   0.40;
 0.99  -0.02  -0.04   0.01   0.00;
 1.04  -0.01   0.12   0.00  -0.40;
 1.10   0.01   0.39  -0.02  -0.80;
```

1.15	0.04	0.58	-0.06	-0.40;
1.21	0.07	0.46	-0.09	-0.40;
1.26	0.08	0.27	-0.10	0.00;
1.32	0.08	-0.08	-0.10	0.00;
1.37	0.06	-0.31	-0.08	0.40;
1.43	0.03	-0.50	-0.05	0.80;
1.48	0.01	-0.42	-0.02	0.40;
1.54	-0.01	-0.35	0.00	0.00;
1.59	-0.00	0.12	0.00	-0.40;
1.65	0.03	0.50	-0.02	-0.40;
1.70	0.05	0.39	-0.04	-0.40;
1.76	0.07	0.35	-0.05	0.00;
1.81	0.07	0.00	-0.05	0.00;
1.87	0.05	-0.23	-0.02	0.80;
1.92	0.03	-0.35	0.00	0.40;
1.98	0.00	-0.62	0.05	0.80;
2.03	-0.01	-0.19	0.07	0.00;
2.09	-0.00	0.15	0.07	0.00;
2.14	0.01	0.27	0.06	0.00;
2.20	0.02	0.12	0.06	0.40;
2.25	0.01	-0.15	0.08	0.40;
2.31	-0.01	-0.42	0.12	0.80;
2.36	-0.04	-0.42	0.15	0.80;
2.42	-0.06	-0.42	0.18	0.40;
2.47	-0.06	0.00	0.18	0.00;
2.53	-0.04	0.31	0.17	0.00;
2.58	-0.03	0.23	0.16	-0.40;
2.64	-0.02	0.23	0.15	0.00;
2.69	-0.01	0.04	0.15	0.40;
2.75	-0.03	-0.27	0.16	0.80;
2.80	-0.05	-0.42	0.18	0.40;
2.86	-0.09	-0.62	0.21	0.40;
2.91	-0.10	-0.19	0.22	-0.40;

2.97	-0.09	0.08	0.20	-0.40;
3.02	-0.08	0.19	0.18	-0.80;
3.08	-0.04	0.69	0.13	-1.20;
3.13	-0.00	0.77	0.07	-0.80;
3.19	0.03	0.62	0.02	-0.80;
3.24	0.04	0.15	-0.00	-0.40;
3.30	0.04	0.08	-0.02	-0.40;
3.35	0.03	-0.19	-0.03	-0.40;
3.41	0.03	0.00	-0.04	-0.40;
3.46	0.04	0.12	-0.06	-0.80;
3.52	0.07	0.46	-0.10	-1.20;
3.57	0.11	0.73	-0.15	-0.80;
3.63	0.13	0.39	-0.19	-0.40;
3.68	0.14	0.23	-0.20	-0.40;
3.74	0.14	0.00	-0.21	0.00;
3.79	0.12	-0.27	-0.20	0.40;
3.85	0.11	-0.35	-0.17	0.40;
3.90	0.08	-0.50	-0.14	1.20;
3.96	0.02	-1.04	-0.08	1.20;
4.01	-0.01	-0.66	-0.02	0.80;
4.07	-0.04	-0.50	0.02	0.40;
4.12	-0.04	-0.04	0.03	0.00;
4.18	-0.03	0.19	0.03	0.00;
4.23	-0.01	0.35	0.02	0.00;
4.29	-0.01	0.08	0.02	0.00;
4.34	-0.02	-0.12	0.04	0.00;
4.40	-0.03	-0.15	0.05	0.40;
4.45	-0.02	0.15	0.05	0.00;
4.51	-0.01	0.04	0.05	0.00;
4.56	-0.02	-0.04	0.06	0.00;
4.62	-0.02	-0.08	0.06	0.00;
4.67	-0.01	0.27	0.05	-0.40;
4.73	0.01	0.23	0.04	0.00;

4.78	0.02	0.19	0.03	0.00;
4.84	0.01	-0.04	0.04	0.40;
4.89	-0.02	-0.58	0.06	0.40;
4.95	-0.04	-0.42	0.09	0.40;
5.00	-0.07	-0.46	0.11	0.40;
5.05	-0.07	0.00	0.11	0.00;
5.11	-0.04	0.39	0.08	-0.40;
5.16	-0.03	0.35	0.06	-0.40;
5.22	-0.01	0.27	0.04	-0.40;
5.27	-0.01	-0.04	0.03	0.00;
5.33	-0.02	-0.12	0.03	0.00;
5.38	-0.02	0.04	0.03	-0.40;
5.44	-0.00	0.27	-0.00	-0.40;
5.49	0.02	0.46	-0.04	-0.40;
5.55	0.02	-0.04	-0.06	-0.40;
5.60	0.03	0.19	-0.07	0.00;
5.66	0.04	0.08	-0.08	0.00;
5.71	0.02	-0.27	-0.08	0.40;
5.77	0.00	-0.39	-0.06	0.00;
5.82	-0.01	-0.15	-0.06	0.00;
5.88	-0.00	0.12	-0.08	-0.40;
5.93	0.03	0.50	-0.11	-0.80;
5.99	0.06	0.54	-0.15	-0.40;
6.04	0.07	0.23	-0.17	-0.40;
6.10	0.07	0.12	-0.19	0.00;
6.15	0.07	-0.12	-0.18	0.40;
6.21	0.04	-0.46	-0.17	0.40;
6.26	0.02	-0.39	-0.15	0.40;
6.32	0.01	-0.27	-0.13	0.00;
6.37	0.01	0.12	-0.14	-0.40;
6.43	0.04	0.42	-0.16	-0.40;
6.48	0.06	0.42	-0.18	-0.40;
6.54	0.08	0.39	-0.20	-0.40;

6.59	0.08	0.08	-0.21	0.40;
6.65	0.08	-0.12	-0.20	0.40;
6.70	0.06	-0.31	-0.18	0.40;
6.76	0.03	-0.50	-0.14	0.40;
6.81	0.01	-0.39	-0.11	0.40;
6.87	0.02	0.08	-0.10	0.00;
6.92	0.03	0.15	-0.10	0.00;
6.98	0.03	0.12	-0.09	0.40;
7.03	0.03	-0.12	-0.07	0.40;
7.09	-0.00	-0.50	-0.03	0.80;
7.14	-0.02	-0.35	-0.00	0.40;
7.20	-0.04	-0.27	0.03	0.00;
7.25	-0.02	0.23	0.03	-0.40;
7.31	0.00	0.46	0.02	-0.40;
7.36	0.01	0.23	0.01	0.00;
7.42	0.03	0.27	0.01	0.40;
7.47	0.02	-0.12	0.03	0.40;
7.53	-0.00	-0.46	0.06	0.80;
7.58	-0.02	-0.35	0.09	0.80;
7.64	-0.05	-0.54	0.13	0.80;
7.69	-0.08	-0.46	0.16	0.40;
7.75	-0.08	-0.08	0.17	0.00;
7.80	-0.07	0.23	0.17	-0.40;
7.86	-0.04	0.54	0.14	-0.40;
7.91	-0.01	0.58	0.11	-0.40;
7.97	0.01	0.27	0.10	0.00;
8.02	0.00	-0.12	0.10	0.40;
8.08	-0.02	-0.35	0.12	0.80;
8.13	-0.05	-0.66	0.16	0.80;
8.19	-0.09	-0.62	0.19	0.00;
8.24	-0.09	-0.12	0.19	0.00;
8.30	-0.09	0.04	0.18	0.00;
8.35	-0.07	0.42	0.16	-0.80;

8.41	-0.04	0.42	0.13	-0.80;
8.46	-0.01	0.69	0.07	-0.80;
8.52	0.04	0.77	0.02	-0.80;
8.57	0.06	0.39	-0.02	-0.40;
8.63	0.06	0.00	-0.03	-0.40;
8.68	0.06	-0.04	-0.04	-0.40;
8.74	0.06	0.15	-0.06	-0.40;
8.79	0.07	0.19	-0.07	0.00;
8.85	0.07	-0.04	-0.08	0.00;
8.90	0.06	-0.19	-0.07	0.40;
8.96	0.03	-0.54	-0.04	0.40;
9.01	0.00	-0.54	-0.01	0.40;
9.07	-0.01	-0.15	-0.01	0.00;
9.12	0.00	0.12	-0.01	-0.40;
9.18	0.03	0.46	-0.04	-0.80;
9.23	0.05	0.46	-0.07	-0.40;
9.29	0.06	0.15	-0.07	-0.40;
9.34	0.06	0.00	-0.07	0.00;
9.40	0.04	-0.27	-0.06	0.40;
9.45	0.02	-0.42	-0.03	0.80;
9.51	0.00	-0.31	-0.00	0.40;
9.56	-0.02	-0.46	0.03	0.40;
9.62	-0.03	-0.08	0.03	0.00;
9.67	-0.01	0.27	0.02	-0.40;
9.73	0.01	0.39	0.01	-0.40;
9.78	0.02	0.23	-0.01	-0.40;
9.84	0.03	0.08	-0.01	0.40;
9.89	0.01	-0.27	0.02	0.40;
9.95	-0.01	-0.35	0.04	0.80;
10.00	-0.03	-0.46	0.06	0.80;
10.05	-0.06	-0.42	0.09	0.40;
10.11	-0.06	-0.04	0.09	0.00;
10.16	-0.04	0.31	0.08	-0.40;

10.22	-0.02	0.39	0.06	-0.40;
10.27	-0.01	0.15	0.05	0.00;
10.33	-0.02	-0.19	0.05	0.00;
10.38	-0.03	-0.19	0.06	0.00;
10.44	-0.04	-0.12	0.06	-0.40;
10.49	-0.02	0.35	0.04	-0.80;
10.55	0.01	0.58	-0.00	-0.80;
10.60	0.04	0.50	-0.03	-0.80;
10.66	0.06	0.42	-0.06	-0.40;
10.71	0.07	0.08	-0.08	0.00;
10.77	0.06	-0.19	-0.07	0.40;
10.82	0.04	-0.23	-0.05	0.80;
10.88	0.00	-0.77	-0.02	0.80;
10.93	-0.04	-0.73	0.02	0.80;
10.99	-0.06	-0.35	0.04	0.00;
11.04	-0.06	0.04	0.04	0.00;
11.10	-0.04	0.23	0.02	-0.40;
11.15	-0.01	0.50	-0.01	-0.80;
11.21	0.01	0.46	-0.04	-0.40;
11.26	0.02	0.12	-0.06	-0.40;
11.32	0.02	0.12	-0.07	0.00;
11.37	0.02	0.00	-0.08	0.00;
11.43	0.01	-0.23	-0.07	0.40;
11.48	-0.00	-0.23	-0.07	0.00;
11.54	-0.01	-0.08	-0.07	0.00;
11.59	0.01	0.31	-0.09	-0.80;
11.65	0.04	0.62	-0.13	-0.80;
11.70	0.07	0.46	-0.16	-0.80;
11.76	0.09	0.42	-0.19	-0.40;
11.81	0.10	0.12	-0.21	0.00;
11.87	0.09	-0.12	-0.20	0.40;
11.92	0.08	-0.23	-0.18	0.40;
11.98	0.04	-0.66	-0.15	0.80;

12.03	0.01	-0.62	-0.11	0.80;
12.09	0.00	-0.15	-0.09	0.00;
12.14	0.00	0.04	-0.09	0.00;
12.20	0.02	0.35	-0.10	0.00;
12.25	0.03	0.19	-0.10	0.00;
12.31	0.03	-0.15	-0.08	0.40;
12.36	0.01	-0.27	-0.05	0.80;
12.42	-0.03	-0.73	-0.01	1.20;
12.47	-0.06	-0.62	0.04	0.40;
12.53	-0.08	-0.35	0.06	0.40;
12.58	-0.09	-0.08	0.07	0.00;
12.64	-0.07	0.31	0.06	-0.40;
12.69	-0.04	0.54	0.03	-0.40;
12.75	-0.02	0.35	0.01	-0.40;
12.80	-0.00	0.35	-0.01	0.00;
12.86	-0.01	-0.08	-0.01	0.00;
12.91	-0.02	-0.19	0.00	0.00;
12.97	-0.02	-0.04	0.00	-0.40;
13.02	-0.00	0.31	-0.02	-0.40;
13.08	0.02	0.42	-0.05	0.00;
13.13	0.03	0.15	-0.06	0.00;
13.19	0.03	0.00	-0.06	0.00;
13.24	0.01	-0.35	-0.05	0.00;
13.30	-0.01	-0.31	-0.03	0.00;
13.35	-0.00	0.08	-0.04	-0.40;
13.41	0.01	0.15	-0.05	-0.80;
13.46	0.04	0.62	-0.09	-0.80;
13.52	0.08	0.77	-0.14	-0.80;
13.57	0.11	0.42	-0.17	-0.40;
13.63	0.12	0.23	-0.17	-0.40;
13.68	0.12	0.00	-0.17	0.40;
13.74	0.11	-0.23	-0.15	0.40;
13.79	0.07	-0.58	-0.11	1.20;

```
   13.85    0.04    -0.54    -0.07    1.20;
   13.90    -0.01   -0.93    -0.00    1.20;
   13.96    -0.06   -1.00    0.07     0.80;
   14.01    -0.10   -0.62    0.12     0.80;
];
        c1=[1; 0; 0; 0; 0;];
        c2=[0; 1; 0; 0; 0;];
        c3=[0; 0; 1; 0; 0;];
        c4=[0; 0; 0; 1; 0;];
        c5=[0; 0; 0; 0; 1;];
        time=m*c1;
        theta=m*c2;
        omega=m*c3;
        x=m*c4;
        v=m*c5;
        plot(time,theta);figure;
        plot(time,omega);figure;
        plot(time,x);figure;
        plot(time,v);figure;
        plot(time,x,time,theta);
```

บรรณานุกรม

KATSUHIKO OGATA," Modern Control Engineering ". Englewood Cliffs, N.J.: Printice-Hall, Inc., 1990.

KATSUHIKO OGATA," Designing Linear Control Systems with MATLAB® ", Printice Hall, Inc., 1994.

Sprenger B., Kucera L., Mourad S., " Balancing of an Inverted Pendulum with a SCARA Robot " IEEE Trans., page 91-97, Volume 3, 1998.

ปรเมษฐ์ ประนยานันทน์, ปิยพงศ์ เผ่าวณิช, " คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 ", บริษัท ซีเอ็ดยูเคชั่น จำกัด, พ.ศ. 2521.

ยืน ภู่วรวรรณ, " ทฤษฎีและการประยุกต์ใช้งานไมโครโปรเซสเซอร์ Z-80 ", บริษัท ซีเอ็ดยูเคชั่น จำกัด, พ.ศ. 2521.

ผศ. โยธิน เปรมปราณีรัชต์, " วิเคราะห์และออกแบบระบบการควบคุมมอเตอร์ " , สจล., กันยายน พ.ศ. 2526.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลงได้ด้วยดี เพราะได้รับคำปรึกษาและความอนุเคราะห์ช่วยเหลือจาก รองศาสตราจารย์ ดร. จงกล งามวิวิทย์ และ อาจารย์สุมิตร พนาอุดมทรัพย์ อาจารย์ที่ปรึกษา คณะผู้จัดทำขอกราบของพระคุณเป็นอย่างสูงมา ณ โอกาสนี้ด้วย และต้องขอขอบพระคุณอาจารย์ประจำภาควิชาวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้ประสิทธ์ประสาทวิชาความรู้แก่คณะผู้จัดทำ

คณะผู้จัดทำ

นายทรงกรด ธิราชัย

นางสาวทัตยา ปุคคลนันทน์

นายนพดล จันทร์คุณาภาส