ระบบปฏิบัติการรถโฟร์คลิฟท์อัจฉริยะ

INTELLIGENT FORKLIFT OPERATION SYSTEM

กำพล เกตุโรจน์

ดวิษ ตามสมัคร

ไทกิ นาคามูระ

สิทธิชัย กำยาน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมแมคคาทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบปฏิบัติการรถโฟร์คลิฟท์อัจฉริยะ

INTELLIGENT FORKLIFT OPERATION SYSTEM

กำพล เกตุโรจน์

ดวิษ ตามสมัคร

ไทกิ นาคามูระ

สิทธิชัย กำยาน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมแมคคาทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

INTELLIGENT FORKLIFT OPERATION SYSTEM

Kamphon Gaterojn
Dravid Tamsamak
Taiki Nakamura
Sittichai Kamyan

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN MECHATRONICS ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2013

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบปฏิบัติการรถโฟร์คลิฟท์อัจฉริยะ
INTELLIGENT FORKLIFT OPERATION SYSTEM

ผู้จัดทำ

นายกำพล	เกตุโรจน์	53010099
นายดวิษ	ตามสมัคร	53010542
นายไทกิ	นาคามูระ	53010604
นายสิทธิชัย	กำยาน	53011678

.....อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์)

.....อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ)

ระบบปฏิบัติการรถโฟร์คลิฟท์อัจฉริยะ

โดย

นายกำพล เกตุโรจน์ 53010099

นายดวิษ ตามสมัคร 53010542

นายไทกิ นาคามูระ 53010604

นายสิทธิชัย กำยาน 53011678

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์

รองศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ

ปีการศึกษา 2556

บทคัดย่อ

เนื่องจากในปัจจุบันการทำงานในโรงงานอุตสาหกรรมรม มีความจำเป็นต้องขนส่งสินค้าอยู่ตลอดเวลา เพื่อการขนย้ายสินค้าต่างๆต้องใช้รถยกในการช่วยขนย้าย ซึ่งการขับรถยกนั้นคนขับมีความสำคัญ โดยคนขับต้องมีทักษะในการควบคุมรถ มีความละเอียดรอบคอบในการทำงานสภาวะจิตต้องพร้อม ดังนั้นการใช้คนมีปัจจัยเสี่ยงหลายอย่าง ยกตัวอย่างเช่น ความประมาททำให้เกิดอุบัติเหตุ ความชำนาญในการขับขี่ไม่เท่ากันทำให้ยากต่อการบริหารจัดการ และสิ้นเปลืองค่าใช้จ่ายในระยะยาว

จากการสำรวจรวบรวมข้อมูล ทำให้เห็นปัญหาต่างๆ จากการทำงาน ดังนั้นทางคณะผู้จัดทำปริญญานิพนธ์จึงเล็งเห็นว่าสามารถสร้างระบบปฏิบัติการรถยกอัจฉริยะขึ้นมา ซึ่งเป็นไปตามความต้องการของตลาดในปัจจุบัน โดยระบบนี้จะช่วยลดเวลาลดค่าใช้จ่ายและมีความปลอดภัย ซึ่งทำให้การทำงานมีประสิทธิภาพสูงขึ้น มีความต่อเนื่องในการทำงานและลดความเสี่ยงต่างๆ ที่เกิดจากความผิดพลาดของคนขับ ทางคณะผู้จัดทำหวังเป็นอย่างยิ่งว่าการใช้ระบบรถยกอัจฉริยะจะมีความแพร่หลายในอนาคตและมีประโยชน์ต่อบุคคลที่สนใจไม่มากก็น้อย

INTELLIGENT FORKLIFT OPERATION SYSTEM

By

Mr. Kamphon Gaterojn 53010099

Mr. Dravid Tamsamak 53010542

Mr. Taiki Nakamura 53010604

Mr. Sittichai Kamyan 53011678

Advisors

Asst. Prof.Sumit Panaudomsup

Assoc. Prof.Kiattisak Kumwachasa

Academic Year 2013

ABSTRACT

Nowadays working in the industry is necessary to transport all the time. To transport goods must to use the forklift help to transport. For driving, the driver is important. The driver must have the skills and experienced to control the forklift, mind is in a good conditions. So the people are several risk factors. Example, negligence caused the accident, expertise in driving inequality makes it difficult to manage and wasteful spending in the long-term. The survey data showed issues from work. So the author of a thesis believe that the system can perform an intelligent forklift, which meets the demands of industry. This system will reduce time and costs. Moreover, this system is secure. That makes the operation more efficient and continuity in performance and reduce risks caused by driver faulty. The author of a thesis sincerely hope that the use of intelligent forklift systems are widespread in the future and benefit the interested more or less.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี เพราะด้วยความช่วยเหลือ คำแนะนำ คำปรึกษา และการดูแลจากหลายฝ่ายด้วยกัน โดยเฉพาะอย่างยิ่งจากอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์ และรองศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ ที่ให้ให้คำแนะนำเสมอมา รวมทั้งเอื้อเฟื้ออุปกรณ์ที่จำเป็นและยังคอยสอนประสบการณ์ในเรื่องการศึกษา การทำงาน และการใช้ชีวิตประจำวัน

อีกทั้งอาจารย์ในสถานศึกษา และอาจารย์ในสาขาวิชาวิศวกรรมการวัดและควบคุมทุกๆ ท่าน ที่อบรมสั่งสอนข้าพเจ้ามาโดยตลอด

ขอบคุณเพื่อนๆ ทุกคนที่ให้กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือ กระตุ้นเตือน รวมทั้งคอยถามไถ่ความคืบหน้าของโครงงานอยู่เสมอ

ขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตของข้าพเจ้า คือบิดามารดา ที่ช่วยเลี้ยงดูอบรมสั่งสอน ส่งเสียให้ข้าพเจ้าได้มีโอกาสศึกษาเล่าเรียน และยังคอยให้กำลังใจ ทำให้ข้าพเจ้ามีทุกวันนี้ ข้าพเจ้าขอขอบคุณมา ณ ที่นี้ด้วย

ผู้จัดทำ

นายกำพล เกตุโรจน์

นายดวิษ ตามสมัคร

นายไทกิ นาคามูระ

นายสิทธิชัย กำยาน

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

ในโรงงานอุตสาหกรรมมการเคลื่อนย้ายสินค้าจำเป็นต้องใช้รถยกในการเคลื่อนย้าย ซึ่งต้องใช้คนในการควบคุมรถยก ในบางครั้งมีอุบัติเหตุจากการทำงานโดยคนขับมีความประมาท อาจก่อให้เกิดอันตรายต่อชีวิตและทรัพย์สิน รวมไปถึงการบริหารจัดการเวลาที่ไม่แน่นอน เนื่องจากคนไม่สามารถทำงานต่อเนื่องได้ตลอดเวลา ส่งผลให้การควบคุมโดยใช้คนนั้นไม่มีเสถียรภาพ ดังนั้นทางคณะผู้จัดทำโครงงานจึงเล็งเห็นว่าสามารถสร้างระบบปฏิบัติการรถยกอัจฉริยะขึ้นมาได้ เพื่อตอบสนองตามความต้องการในการทำงาน ในปัจจุบันที่ต้องการความสะดวกความรวดเร็วมีความแม่นยำสูงสามารถบริหารจัดการเวลาได้ถูกต้อง ยังสามารถลดค่าใช้จ่ายในการฝึกอบรมพนักงานที่ขับรถได้ นอกจากนั้นยังลดโอกาสการเกิดอุบัติเหตุ ลดการสูญเสียชีวิตและทรัพย์สิน

1.2 ความมุ่งหมาย

1. ศึกษาการทำงานของระบบเครือข่ายไร้สาย (Wifi) ให้สามารถรับและส่งข้อมูลเพื่อส่งให้รถยกทำงานได้

2. ศึกษาการควบคุมอุปกรณ์อิเล็คทรอนิคส์โดยไมโครคอนโทรลเลอร์ ATMega32 ผ่านระบบเครือข่ายไร้สาย (Wifi)

3. ศึกษาการทำงานของระบบเลี้ยว

4. ศึกษาเซนเซอร์แสงที่ใช้ในการควบคุมทิศทางของรถ

1.3 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อให้เราสามารถควบคุมระบบจัดเก็บและขนส่งได้อย่างเที่ยงตรง

2. เพื่อลดค่าใช้จ่ายในการฝึกอบรมบุคลากร

3. เพื่อลดเวลาในการทำงาน

4. เพื่อลดแรงงานคน

5. เพื่อเพิ่มประสิทธิภาพในการทำงาน

6. เพื่อลดค่าใช้จ่ายจากการใช้พลังงาน

7. เพื่อลดอุบัติเหตุในการทำงาน

1.4 ขอบเขตการศึกษา

1. ศึกษาการทำงานของระบบเครือข่ายไร้สาย (Wifi) ให้สามารถรับและส่งข้อมูลเพื่อส่งให้รถยกทำงานได้

2. ศึกษาการควบคุมอุปกรณ์อิเล็คทรอนิคส์โดยไมโครคอนโทรลเลอร์ ATMega32 ผ่านระบบเครือข่ายไร้สาย (Wifi)

3. ศึกษาการทำงานของระบบเลี้ยว

4. ศึกษาเซนเซอร์แสงที่ใช้ในการควบคุมทิศทางของรถ

1.5 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวนำถึงวัตถุประสงค์ ขั้นตอนการศึกษา และการจัดทำโครงงาน พร้อมทั้งรายละเอียดของปริญญานิพนธ์

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง กล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการออกแบบรถโฟร์คลิฟท์และนำเอาความรู้ไปประยุกต์ใช้ในการจัดทำโครงงาน

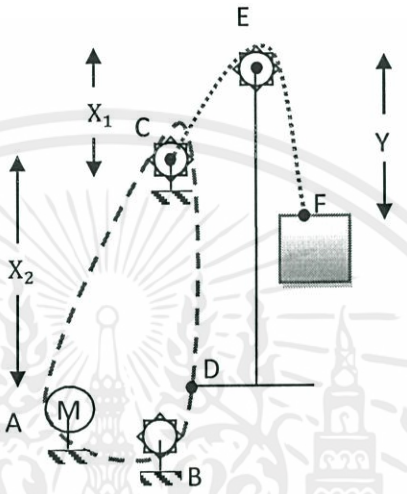
บทที่ 3 การคำนวณและการสร้างรถโฟร์คลิฟท์

บทที่ 4 การทดลองและผลการทดลอง

บทที่ 5 บทวิจารณ์และสรุป จะสรุปผลการดำเนินงานปัญหาที่เกิดขึ้น และแนวทางการปรับปรุงพัฒนาโครงงานนี้ต่อไป

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

2.1 ทฤษฎีระบบการเคลื่อนที่ของรอก



รูปที่ 2.1 แสดงระบบการเคลื่อนที่ของรอก

จากรูปที่ 2.1 สามารถสังเกตได้ว่า ระยะที่เปลี่ยนไปของ $CE + DC + EF$ เท่ากับค่าคงที่ค่าหนึ่ง เนื่องจากเส้นตรงเดียวกัน และระยะที่เปลี่ยนไปของ $CE = DC$ จึงทำให้ได้สมการที่ 1 ซึ่งนำมาหาอนุพันธ์จะได้ สมการความเร็วและสมการความเร่ง ดังสมการที่ 2 และสมการที่ 3

ระยะในแนวแกนดิ่ง; $CE + DC + EF = \text{constant}$

$X_1 + X_2 + Y = \text{constant}$

พิจารณาระยะที่เปลี่ยนไป $2X + Y2 = \text{constant}$ (2.1)

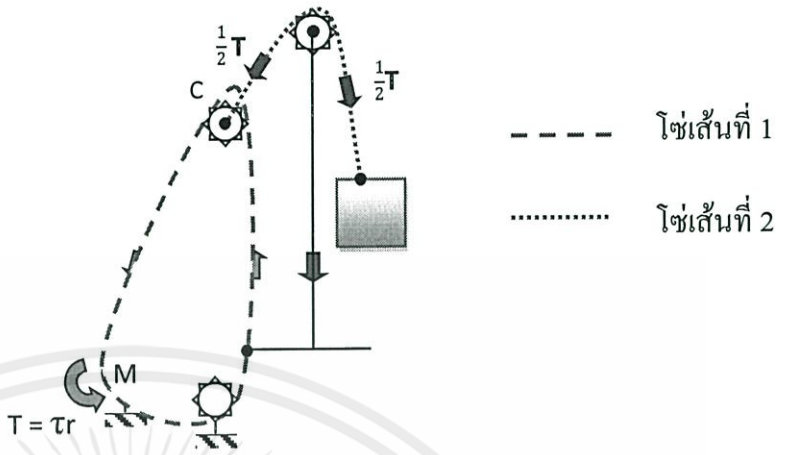
อนุพันธ์ลำดับที่ 1 $2v_x + v_y = 0$ (2.2)

อนุพันธ์ลำดับที่ 2 $2a_x + a_y = 0$ (2.3)

ดังนั้น Input คือระยะที่เปลี่ยนไป, ความเร็ว, ความเร่งของ $CE + DC$ มีค่าเท่ากับ $2X, 2v_x, 2a_x$

Output คือระยะที่เปลี่ยนไป, ความเร็ว, ความเร่งของ EF มีค่าเท่ากับ Y, v_y, a_y

จะได้อัตราส่วนระหว่าง Input และ Output คือ 1 : 2



รูปที่ 2.2 แสดงสมดุลแรงของโซ่ในระบบ

จากรูปที่ 2.2 กำหนดให้มอเตอร์ขับทอร์ก (τ) มีแรงตึงเชือกในโซ่เส้นที่ 1 เท่ากับ T=τr ในทิศขึ้น จะได้แรงปฏิกิริยาที่แกนยกเท่ากับ T ในทิศลง และเมื่อพิจารณาที่แกนยกจะเกิดแรงปฏิกิริยาในโซ่เส้นที่ 2 ที่จุด C และวัตถุจะมีแรงตึงเชือกเท่ากับ $\frac{1}{2}$T ในทิศลง ส่งผลให้เกิดผลรวมระหว่างแรงที่จุด C และแรงรวมวัตถุเท่ากับ T เพื่อนำไปต้านแรงตึงเชือกจากเส้นที่ 1 ดังนั้นจึงสรุปได้ว่า เมื่อมอเตอร์ขับทอร์กด้วยแรงตึงเชือกเส้นที่ 1 ทั้งหมด 1T จะทำให้สามารถยกวัตถุที่แรงตึงเชือกเส้นที่ 2 ได้เท่ากับ $\frac{1}{2}$T

จากผลรวมแรง ; $\sum F_y = 0$

แรงมอเตอร์ – แรงวัตถุ – แรงที่จุด C = 0

$$T - \frac{1}{2}T - \frac{1}{2}T = 0$$

$$T = \frac{1}{2}T + \frac{1}{2}T$$

ดังนั้น

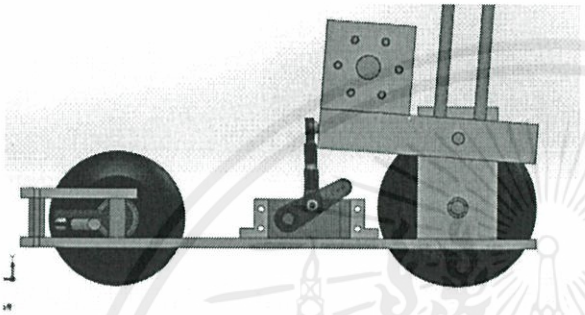
Input คือแรงมอเตอร์ มีค่าเท่ากับ T

Output คือแรงวัตถุ มีค่าเท่ากับ $\frac{1}{2}$T

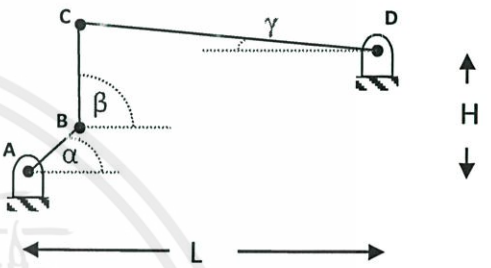
จะได้อัตราส่วนระหว่าง Input และ Output คือ 2 : 1

2.2 ทฤษฎี Four-bar linkage

ในการใช้งานโฟร์คลิฟท์นั้นเราจะต้องเอียงแกนยกไปข้างหน้าและข้างหลังเพื่อช่วยให้การยกของและวางของในที่สูงมีประสิทธิภาพ ในการเอียงแกนยกนั้นเราจะใช้เซอร์โวมอเตอร์ในการควบคุมองศาเอียง ซึ่งการควบคุมองศาเอียงอย่างมีประสิทธิภาพ เราจะต้องรู้ความสัมพันธ์ขององศาระหว่างองศาเซอร์โวมอเตอร์ที่เปลี่ยนไปกับองศาแกนยกที่เปลี่ยนไปและนำความสัมพันธ์นั้นไปใช้ในการคำนวณองศาการเอียงในโปรแกรมอย่างถูกต้อง



รูปที่ 2.3 แสดงรูปภาพของระบบเอียง



รูปที่ 2.4 แสดง Diagram ของระบบเอียง

แนวดิ่ง $$BC\cos(\beta) = L - AB\cos(\alpha) - CD\cos(\gamma) \quad (2.4)$$

แนวระนาบ $$BC\sin(\beta) = H - AB\sin(\alpha) + CD\sin(\gamma) \quad (2.5)$$

สมการที่ 2.5 หารด้วย สมการที่ 2.4

$$\frac{BC\sin(\beta)}{BC\cos(\beta)} = \tan(\beta) = \frac{L - AB\cos(\alpha) - CD\cos(\gamma)}{H - AB\sin(\alpha) + CD\sin(\gamma)} = X$$

จะได้ $$\beta = \tan^{-1}\left(\frac{L - Ab\cos(\alpha) - CD\cos(\gamma)}{H - AB\sin(\alpha) - CD\sin(\gamma)}\right) \quad (2.6)$$

สมการที่ 2.6 แทนใน สมการที่ 2.5

$$\sin(\tan^{-1}(X)) = \frac{H - AB\sin(\alpha) + CD\sin(\gamma)}{BC}$$

$$\sin\left(\sin^{-1}\left(\frac{X}{\sqrt{X^2+1}}\right)\right) = \frac{H - AB\sin(\alpha) + CD\sin(\gamma)}{BC} * \tan^{-1}X = \sin^{-1}\frac{X}{\sqrt{X^2+1}}$$

$$\frac{X}{\sqrt{X^2+1}} = \frac{H - AB\sin(\alpha) + CD\sin(\gamma)}{BC}$$

$$\frac{BC\,X}{\sqrt{X^2+1}} - H + AB\sin(\alpha) = CD\sin(\gamma)$$

$$\frac{BC\,X}{CD\sqrt{X^2+1}} - \frac{H}{CD} + \frac{AB\sin(\alpha)}{CD} = \sin(\gamma)$$

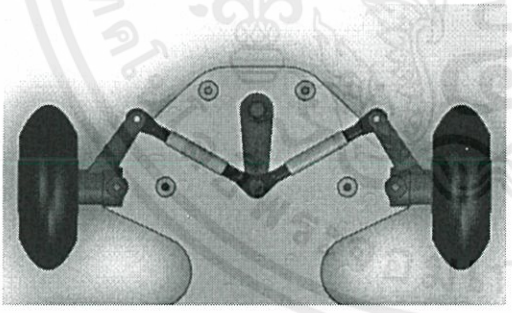
$$\gamma = \sin^{-1}\left(\frac{BC\,X}{CD\sqrt{X^2+1}} - \frac{H}{CD} + \frac{AB\sin(\alpha)}{CD}\right)$$

$$\gamma = \sin^{-1}\left(\frac{BC\left(\frac{L - AB\cos(\alpha) - CD\cos(\gamma)}{H - AB\sin(\alpha) + CD\sin(\gamma)}\right)}{CD\sqrt{\left(\frac{L - AB\cos(\alpha) - CD\cos(\gamma)}{H - AB\sin(\alpha) + CD\sin(\gamma)}\right)^2+1}} - \frac{H}{CD} + \frac{AB\sin(\alpha)}{CD}\right) \qquad (2.7)$$

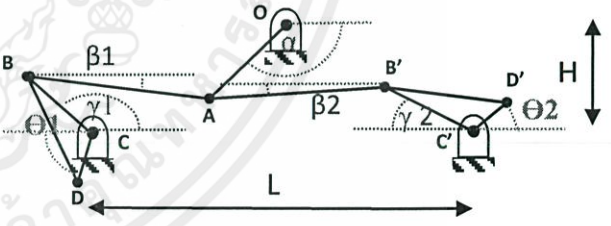
จากทฤษฎีของ Four-bar linkage สามารถหาความสัมพันธ์ระหว่างมุม α และ γ โดยใช้วิธีคณิตวิเคราะห์ (Analytical method) จะได้สมการที่ 2.4 และสมการที่ 2.5 จากระยะในแนวดิ่งและแนวระนาบตามลำดับ หลังจากนั้นนำสมการที่ 2.5 หารด้วย สมการที่ 2.4 จะได้ β ในเทอมของ α และ γ คือสมการที่ 2.6 จากนั้นนำมาแทนในสมการที่ 2.5 จัดรูปสมการจนได้ γ ในเทอมของ α คือ สมการที่ 2.7

เมื่อนำสมการที่ 2.7 มาหาอนุพันธ์ลำดับที่ 1 จะได้ความเร็วเชิงมุมของชิ้นส่วน CD เมื่อเทียบกับเชิงมุมของชิ้นส่วน AB และถ้านำมาหาอนุพันธ์ลำดับที่ 2.5 ก็จะได้ความเร่งเชิงมุมของชิ้นส่วน CD เมื่อเทียบกับความเร่งเชิงมุมของชิ้นส่วน AB

ในการออกแบบระบบเลี้ยงของล้อหลังดังในรูปที่ 2.5 ก็ใช้ทฤษฎีเดียวกันในการคำนวณหาความสัมพันธ์ของมุม ซึ่งจะต้องแยกคิดทีละล้อ



รูปที่ 2.5 แสดงรูปภาพที่วาดจากโปรแกรม CAD

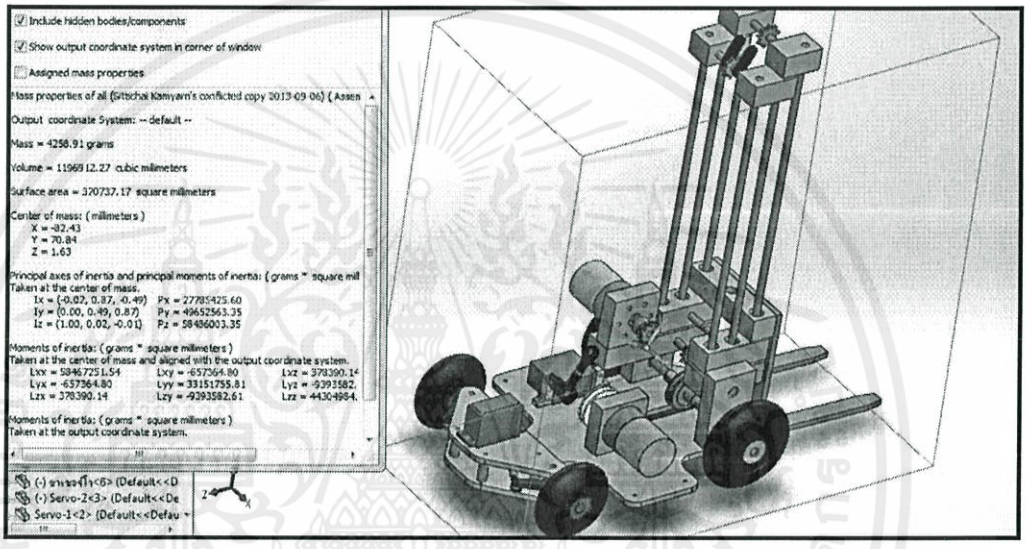


รูปที่ 2.6 แสดง Diagram ของระบบเลี้ยวล้อ

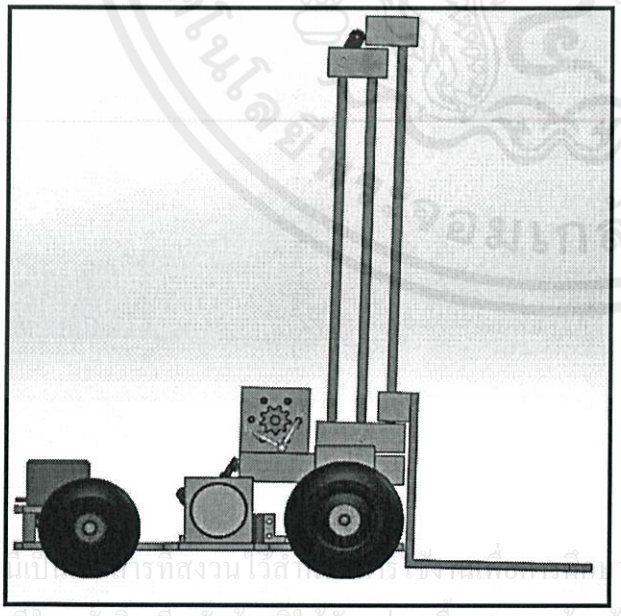
2.3 ทฤษฎี Center of Mass

ในการออกแบบตัวรถจำเป็นต้องคำนึกถึงเรื่องจุดศูนย์กลางมวล เพื่อให้ตัวรถยกนั้นสามารถยกของและเคลื่อนที่โดยที่ตัวรถอยู่ในสภาพสมดุลไม่พลิกคว่ำ มิเช่นนั้นอาจจะก่อให้ความเสียหายแก่ทรัพย์สินและอันตรายต่อบุคคลในการคำนวณจุดศูนย์กลางมวลนั้น เราจะใช้ทฤษฎี Center of Mass ดังนี้

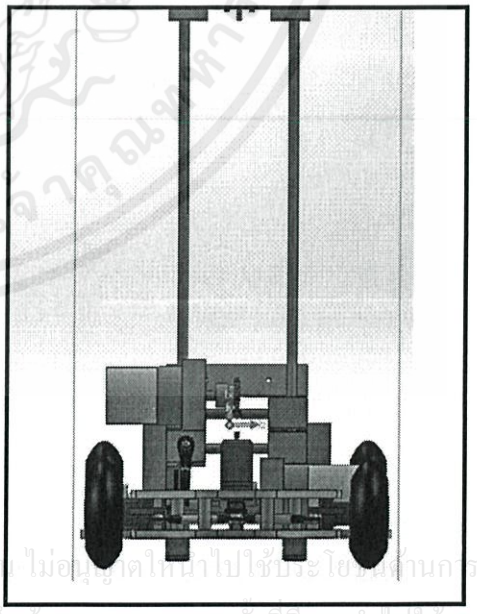
$$X_{cm} = \frac{\sum_{i=1}^{N} m_i x_i}{M}, Y_{cm} = \frac{\sum_{i=1}^{N} m_i y_i}{M}, Z_{cm} = \frac{\sum_{i=1}^{N} m_i z_i}{M}$$



รูปที่ 2.7 แสดง Center of Mass ในมุมมองต่างๆ



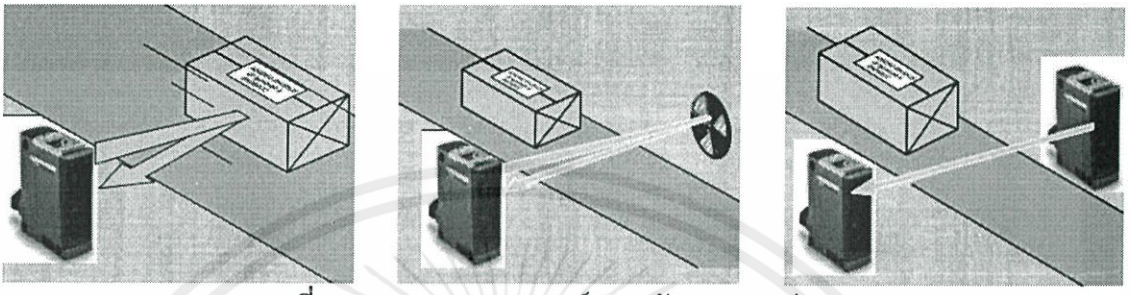
รูปที่ 2.8 แสดง Center of Mass ในมุมมองต่างๆ



รูปที่ 2.9 แสดง Center of Mass ในมุมมองต่างๆ

2.4 ทฤษฎีตรวจจับแบบใช้แสง

ใช้ตรวจจับวัตถุโดยสามารถทำงานได้ 3 ลักษณะคือ อุปกรณ์ตรวจจับแสงแบบใช้การสะท้อนแสงของวัตถุอุปกรณ์ตรวจจับแสงแบบใช้ตัวสะท้อนแสงและอุปกรณ์ตรวจจับแสงแบบใช้การกั้นแสงของวัตถุ



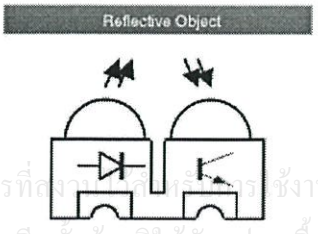
รูปที่ 2.10 ภาพแสดงอุปกรณ์ตรวจจับแสงแบบต่างๆ

ตัวเซนเซอร์ประกอบด้วย 2 ส่วน คือ

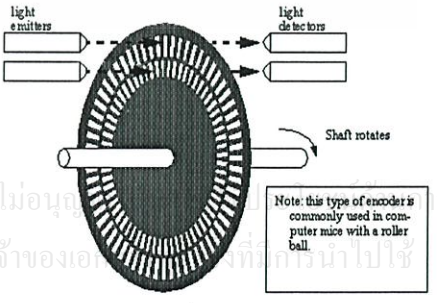
1. ตัวกระจายแสง (Emittingcomponent) ใช้ไดโอดเปล่งแสง (LED: Light Emitting Diode) ทำหน้าเปลี่ยนสัญญาณทางไฟฟ้าเป็นแสงส่วนใหญ่มักใช้คลื่นแสงที่ตาเราไม่สามารถมองเห็นได้ เช่น แสงอินฟราเรด

2.ตัวรับแสง (Receivingcomponent) ใช้อุปกรณ์จำพวก Phototransistor หรือ Photodiode
โดยสามารถเปลี่ยนค่าความเข้มแสง เป็นสัญญาณทางไฟฟ้าได้

โดยเซนเซอร์แสงที่เรานำมาใช้ตรวจจับเส้นเพื่อเป็นทางเดินของตัวรถจะเป็นแบบใช้การสะท้อนแสงของวัตถุเนื่องจากเราไม่สามารถนำตัวสะท้อนแสงหรือตัวเซนเซอร์ไปติดที่พื้นถนนได้ และเซนเซอร์ตัวนี้จะทำหน้าที่ตรวจจับวัสดุที่สามารถสะท้อนแสง และไม่สะท้อนแสงได้ ในที่นี้เราใช้ สีขาวที่สามารถสะท้อนแสงได้ดีและสีดำที่สะท้อนแสงได้น้อย และเซนเซอร์แสงที่เรานำมาใช้ตรวจวัดการหมุนของมอเตอร์จะเป็นแบบการกั้นแสงของวัตถุ โดยจะมีจานหมุนที่ต่อกับมอเตอร์อยู่ระหว่างตัวเซนเซอร์ซึ่งทำหน้าบังแสงจากตัวกระจายแสง



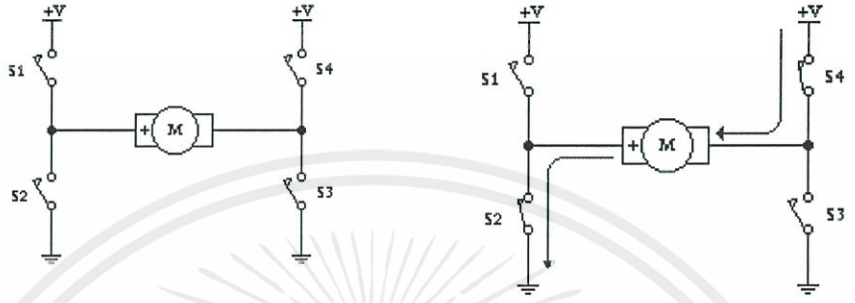
รูปที่ 2.11 แสดงเซนเซอร์ตรวจจับเส้น



รูปที่ 2.12 แสดงเซนเซอร์ตรวจวัดการหมุน

2.5 ทฤษฎี H-Bridge Switching

หลักการของวงจรนั้นจะประกอบไปด้วยสวิทช์ 4 ตัว นั่นก็คือ S1, S2, S3 และ S4 นั่นเอง ซึ่งในรูปตัวอย่างจะใช้ดีซีมอเตอร์ (DC-Motor) เป็นโหลด (Load) ของวงจร



รูปที่ 2.13 H-Bridge Switching

ในสภาวะเริ่มต้นสวิทช์ทุกตัวปิด (Off) อยู่จะไม่มีอะไรเกิดขึ้นทั้งสิ้นเพราะไม่มีกระแสไฟฟ้าไหลเข้าสู่มอเตอร์ และเมื่อเราทำการเปิด (On) สวิทช์ S1 และ S3 พร้อมกัน จะเป็นการเชื่อมวงจรทำให้มีกระแสไฟฟ้าไหลผ่านมอเตอร์จากขั้วบวกของมอเตอร์ไปยังขั้วลบของมอเตอร์จึงทำให้มอเตอร์สามารถหมุนได้ในทิศทางฟอร์เวิร์ด (Forward) (จะหมุนแบบตามเข็มนาฬิกา หรือทวนเข็มนาฬิกานั้นขึ้นอยู่กับลักษณะของการพันขดลวดภายในมอเตอร์) และในทางกลับกันถ้าหากเราทำการเปิด (On) สวิทช์ S2 และ S4 พร้อมกัน ก็จะเป็นการเชื่อมวงจร และทำให้เกิดกระแสไฟฟ้าไหลผ่านมอเตอร์ จากขั้วลบของมอเตอร์ไปยังขั้วบวกของมอเตอร์ จึงทำให้มอเตอร์สามารถหมุนได้และเป็นการหมุนในทิศทางรีเวิร์ด (Reverse) (กลับทิศทางกับกรณีแรก)ถ้าเกิดเราทำการเปิด (On) สวิทช์ S1 และ S4 หรือ S2 และ S3 คู่ใดคู่หนึ่งพร้อมกัน ก็จะเป็นการเชื่อมวงจร ทำให้มอเตอร์ทำตัวเป็นเจนเนอร์เรเตอร์ (Generator) จึงทำให้หยุดหมุนทันที

**** จากทฤษฎีข้างต้นจึงสามารถนำไปใช้ในการในวงจรควบคุมมอเตอร์เพื่อควบคุมทิศทางการหมุนของมอเตอร์ โดยผู้จัดทำได้เลือกทรานซิสเตอร์ (Transistor) มาเป็นสวิทช์ในการควบคุมทิศทางการหมุนของมอเตอร์โดยเลือกใช้ IC L298N ซึ่งสามารถควบคุมมอเตอร์ได้ทั้งทิศทางและความเร็วของมอเตอร์*

2.6 ทฤษฎี DC Motor

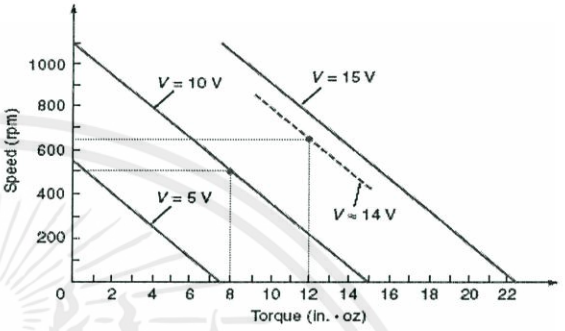
ในการควบคุมมอเตอร์กระแสตรง มีตัวแปรที่สำคัญที่ทำให้มอเตอร์หมุนไปตามที่เราต้องการ คือแรงดันไฟฟ้า (V) ความเร็วเชิงมุม (ω) และแรงบิดของมอเตอร์ (τ) ซึ่งมีความสัมพันธ์เป็นไปตามสมการข้างล่างนี้

$$V_t = I_a R_a + k\phi\omega$$

$$I_a = \frac{T}{k\phi}$$

$$V_t = \frac{R_a}{k\phi} T + k\phi\omega$$

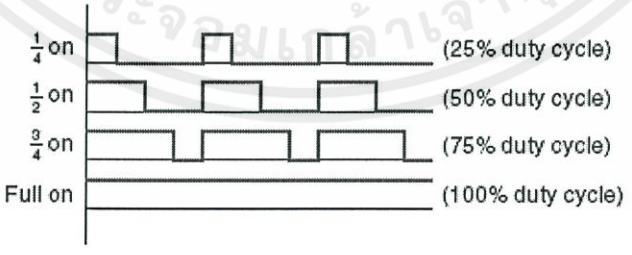
$$\omega = -\frac{R_a}{(k\phi)^2} T + \frac{V_t}{k\phi}$$



รูปที่ 2.14 กราฟแสดงความสัมพันธ์ของ V, ω และ τ

จะได้สมการเส้นตรง ที่มีแกนตั้งคือ ω แกนนอนคือ τ มีความชันคงที่คือ $-\frac{R_a}{(k\varphi)^2}$ โดย k, ϕ และ R_a เป็นค่าคงที่

ในรูปที่ 2.14 แสดงให้เห็น V ที่ค่าต่างๆ ทำให้เรานำไปใช้ในการคำนวณหาความเร็วเชิงมุมและแรงบิดมอเตอร์ได้เช่น มอเตอร์ขนาด 10V สามารถให้ความเร็วเชิงมุม 500 rpm เมื่อแรงบิดมีค่า 8 Nm ทั้งนี้เรายังสามารถควบคุมความเร็วเชิงมุมมอเตอร์ได้จากสัญญาณเพ้าส์ เรียกว่า PWM (Pulse-Wide Modulated) เพื่อใช้สั่งทรานซิสเตอร์ให้ทำงานเหมือนเป็นสวิทช์เปิด-ปิดมอเตอร์ ดังรูปที่ 2.15



รูปที่ 2.15 แสดง PWM ในแต่ละ Duty Cycle

เราสามารถคำนวณความเร็วเชิงมุมสุทธิได้จาก % ของ Duty Cycle เป็น % ของความเร็วเชิงมุมสูงสุดได้เลย

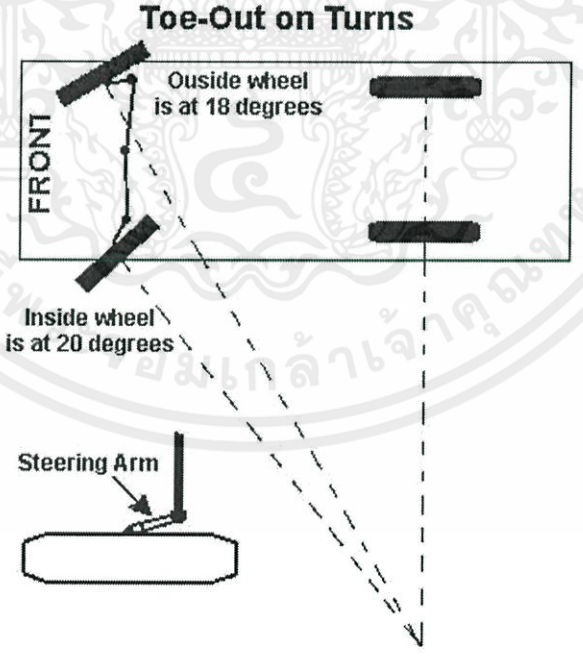
บทที่ 3

การคำนวณและการสร้าง

การออกแบบรถโฟร์คลิฟท์เพื่อขนส่งสินค้าเพื่อให้ได้ตามต้องการจำเป็นต้องคำนึงถึงประสิทธิ์ภาพในการขับเคลื่อนและความสามารถในการยกสินค้า เพื่อให้การเคลื่อนย้ายสินค้าเป็นไปอย่างมีประสิทธิ์ภาพสูงสุด และต้องคำนึงถึงความแข็งแรงของชิ้นส่วนต่างๆ หลักการออกแบบชิ้นส่วนต่างๆ มีหลักการดังนี้

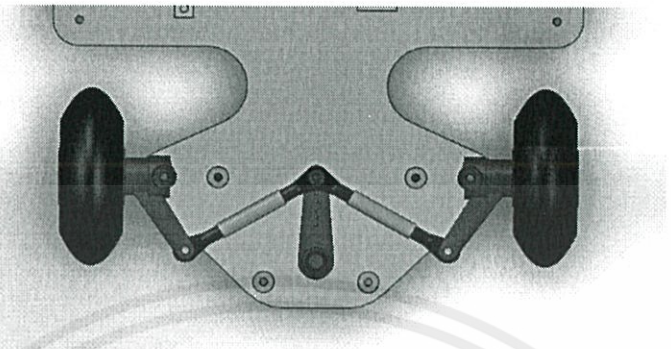
3.1 ระบบชักเลี้ยวของรถ

ในการออกแบบระบบเลี้ยวของรถสิ่งที่ต้องคำนึงคือวงเลี้ยวของรถ ดังนั้นเราจึงควรออกแบบระบบเลี้ยวล้อหน้าและล้อหลังให้มีองศาการเลี้ยวที่สัมพันธ์กันทั้ง 4 ล้อเพื่อให้เกิดจุดหมุนขณะเลี้ยวร่วมกันทั้ง 4 ล้อ ซึ่งจะทำให้ยางหมุนรอบจุดหมุนได้โดยไม่เกิดการไถล มิฉะนั้นจะทำให้ยางเกิดการสึกหลอและเกิดแรงเสียดทานระหว่างล้อกับพื้นถนน ที่เป็นเหตุให้สิ้นเปลืองกำลังโดยไม่จำเป็น ดังนั้นเราจึงต้องทำการออกแบบระบบให้ให้เกิด “Toe-Out on Turns” ดังรูปที่ 3.1

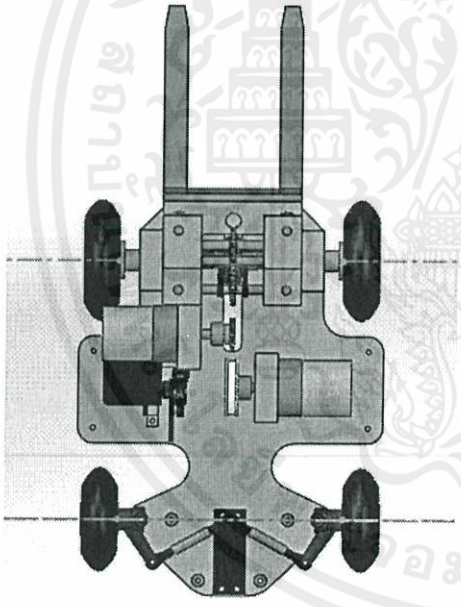


รูปที่ 3.1 แสดงการเกิดจุดหมุนของรถขณะเลี้ยว

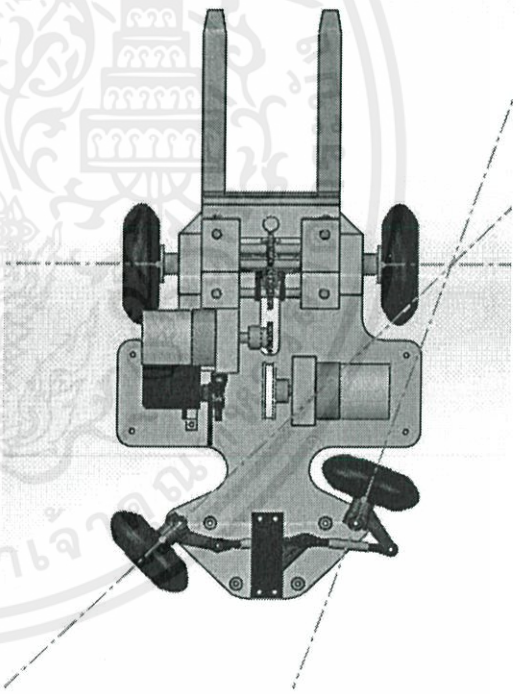
จากหลักการดังกล่าวทางคณะผู้จัดโครงงานทำจึงใช้โปรแกรม SolidWorksเพื่อจำลองดูการเคลื่อนที่ของล้อที่ทำให้เกิดจุดตัดร่วมกันทั้ง 4 ล้อขณะเลี้ยวดังรูปที่ 3.2



รูปที่ 3.2 แสดงระบบชักเลี้ยวของตัวรถ



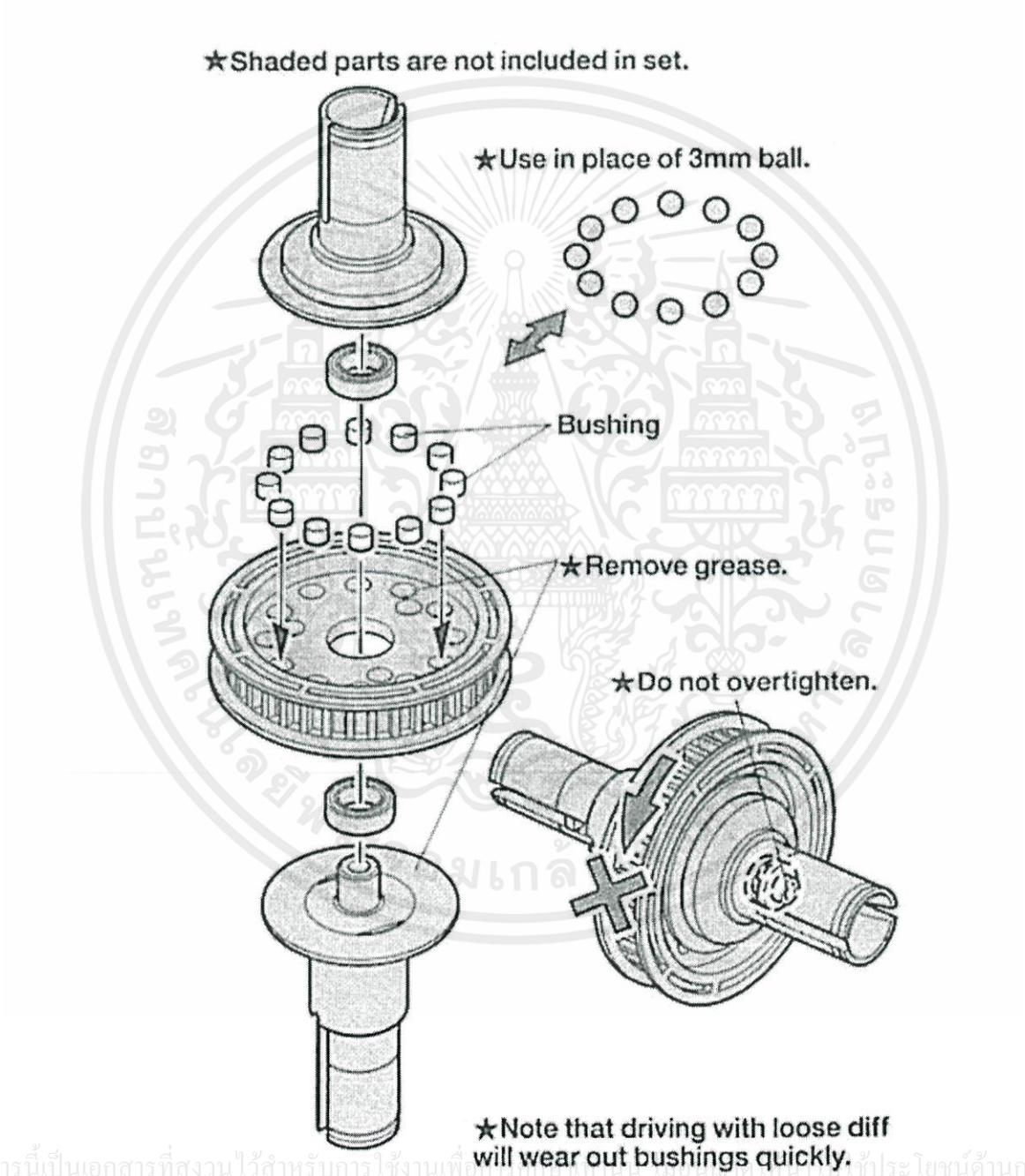
รูปที่ 3.3 แสดงจุดหมุนของรถ



รูปที่ 3.4 แสดงจุดหมุนของรถขณะเลี้ยว

3.2 ระบบ Differential

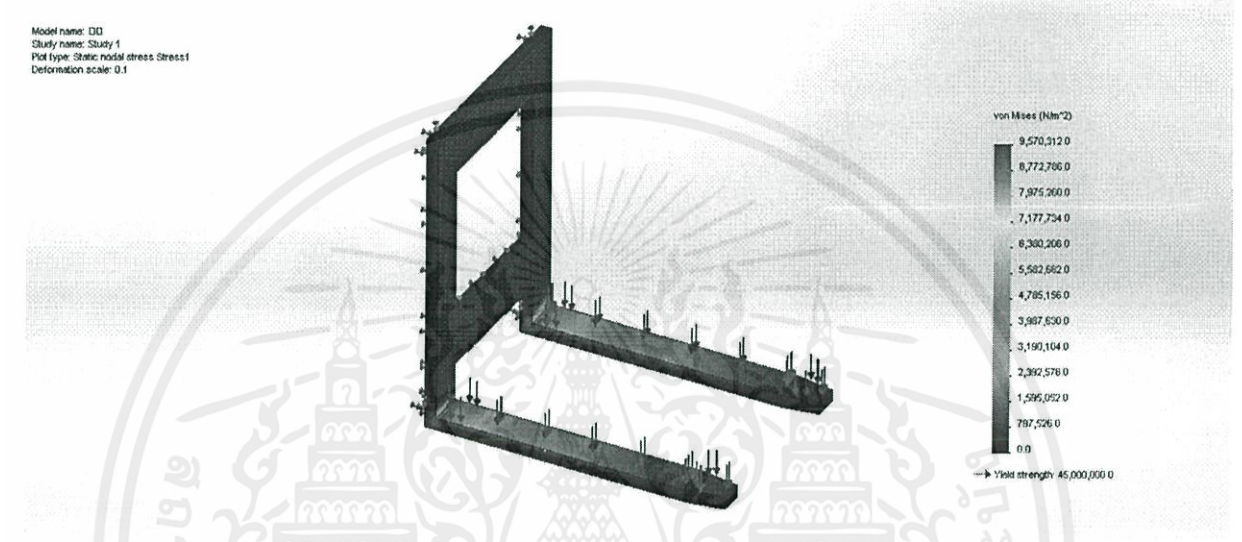
ในการออกแบบระบบ Differential สิ่งต้องคำนึงขณะเลี้ยวคือ ล้อหน้าซึ่งไม่ได้มีการชักเลี้ยว ความเร็วของทั้ง2ล้อจะมีความเร็วที่ไม่เท่ากันเพื่อที่จะป้องกันไม่ให้เกิดการไถลของยาง อันเป็นสาเหตุที่ทำให้รถเกิดอาการสะดุดขณะทำการเลี้ยว ดังนั้นเรามีความจำเป็นต้องต้องใช้ระบบ Differential ในการส่งกำลังจากมอเตอร์มายังล้อทั้งสองที่มีความเร็วในการเคลื่อนที่ที่ไม่เท่ากันได้



รูปที่ 3.5 แสดงการประกอบของชิ้นส่วนต่างๆของ ระบบ Differential

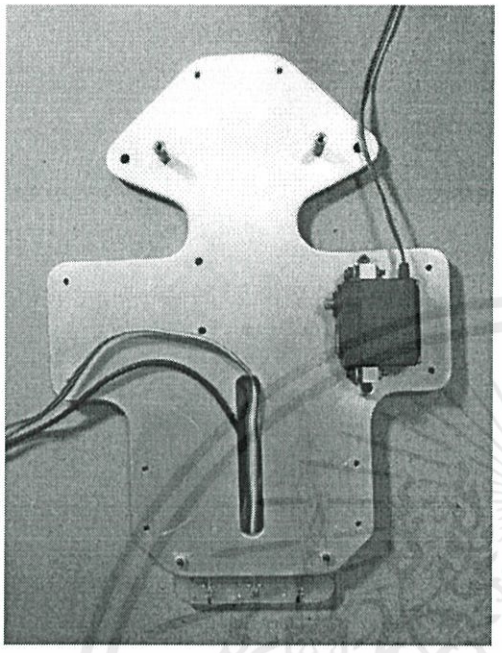
3.3 ความแข็งแรงของงา

ในการออกแบบงาสิ่งที่ต้องคำนึงถึงเป็นอันดับแรกคือความแข็งแรงของตัวงาเนื่องจากสินค้าที่เราต้องการจะยกนั้นอาจจะมีน้ำหนักมาก ดังนั้นเราจำเป็นต้องใช้โปรแกรม Solidworks ในการคำนวณความแข็งแรงดังรูปที่ 3.6

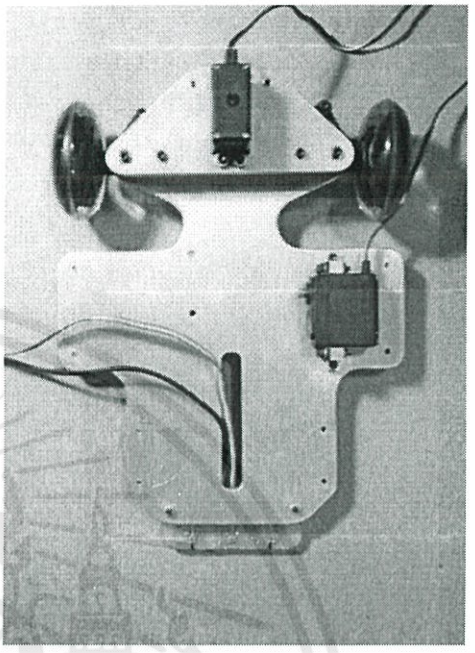


รูปที่ 3.6 การคำนวณแรงโดยใช้โปรแกรม Solidworks

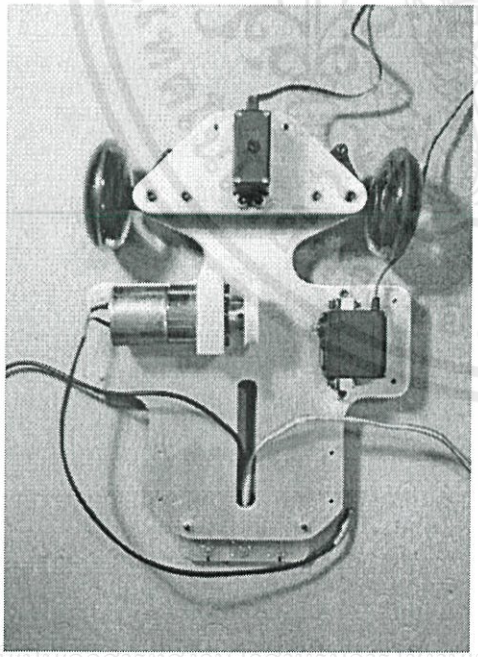
3.4 ขั้นตอนการประกอบ



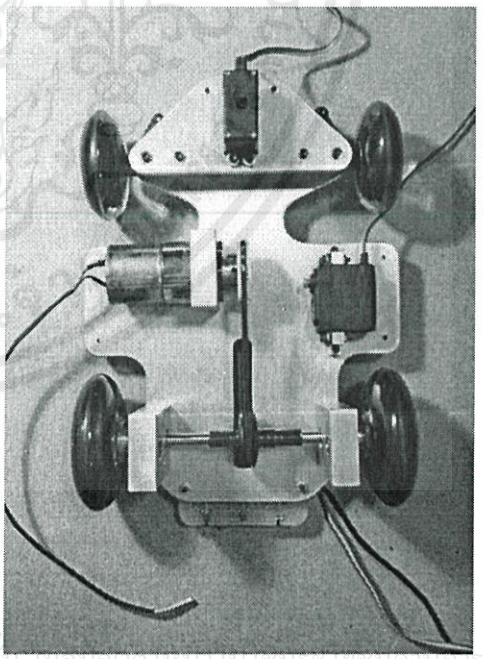
รูปที่ 3.7 ประกอบเซอร์โวและเซนเซอร์



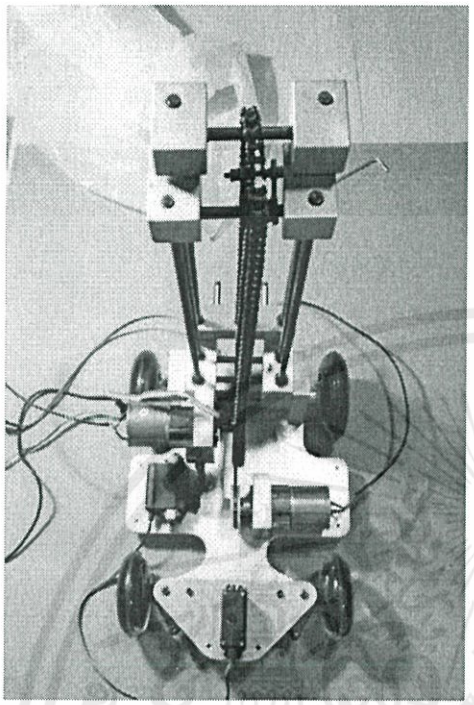
รูปที่ 3.8 ประกอบระบบชักเลี้ยว



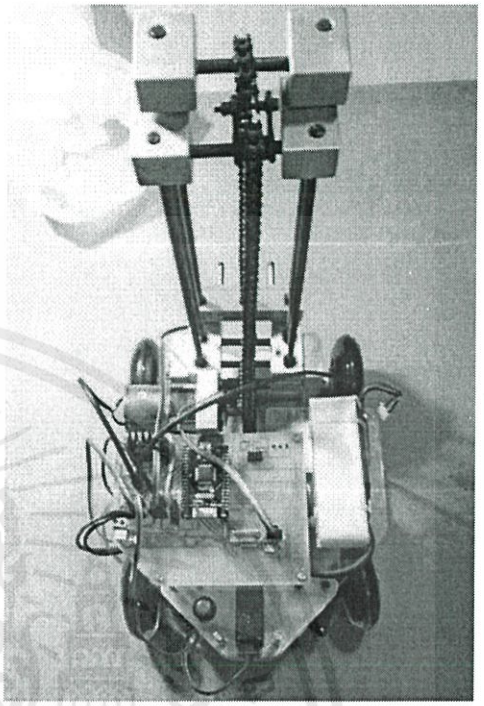
รูปที่ 3.9 ประกอบมอเตอร์และพูลเล่



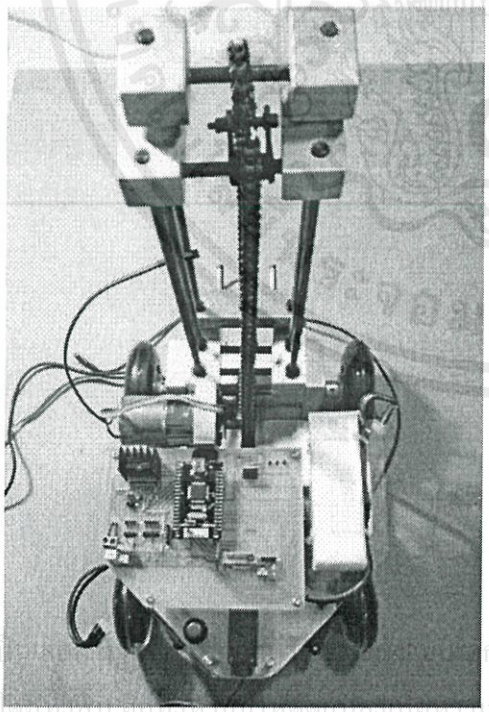
รูปที่ 3.10 ประกอบชุด Differential



รูปที่ 3.11 ประกอบชุดยกและมอเตอร์



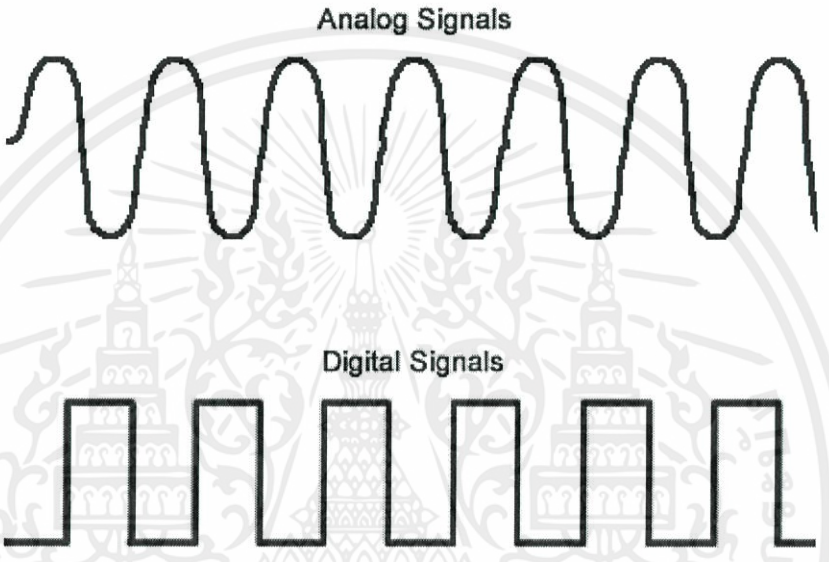
รูปที่ 3.12 ประกอบแผ่นวงจรและแบตเตอรี่



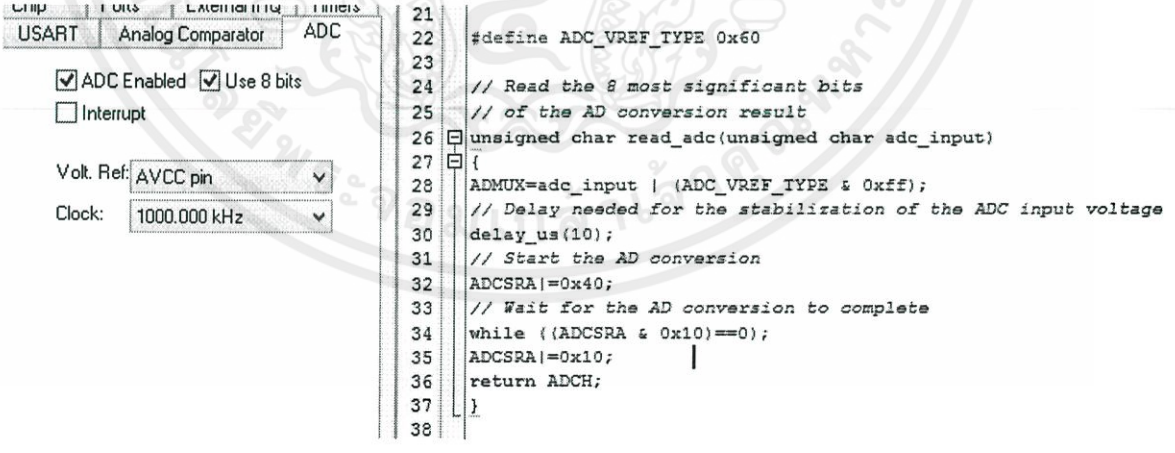
รูปที่ 3.13 เชื่อมต่อสายไฟมอเตอร์และเซนเซอร์

3.5 ตัวแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล Analog to Digital Converter (ADC)

เนื่องจากในการรับ Input จากอุปกรณ์บางชนิดที่ให้สัญญาณอนาล็อก จำเป็นต้องผ่าน Converter เพื่อเปลี่ยนให้เป็นสัญญาณแบบดิจิตอลซึ่งอุปกรณ์อนาล็อกที่ใช้ในโครงงานนี้มีทั้ง Optical Sensor, Encoder และ Variable Resistance ใน Microcontroller ตัวนี้ มีพอร์ตที่รองรับ ADC อยู่ คือพอร์ต A0 - A7



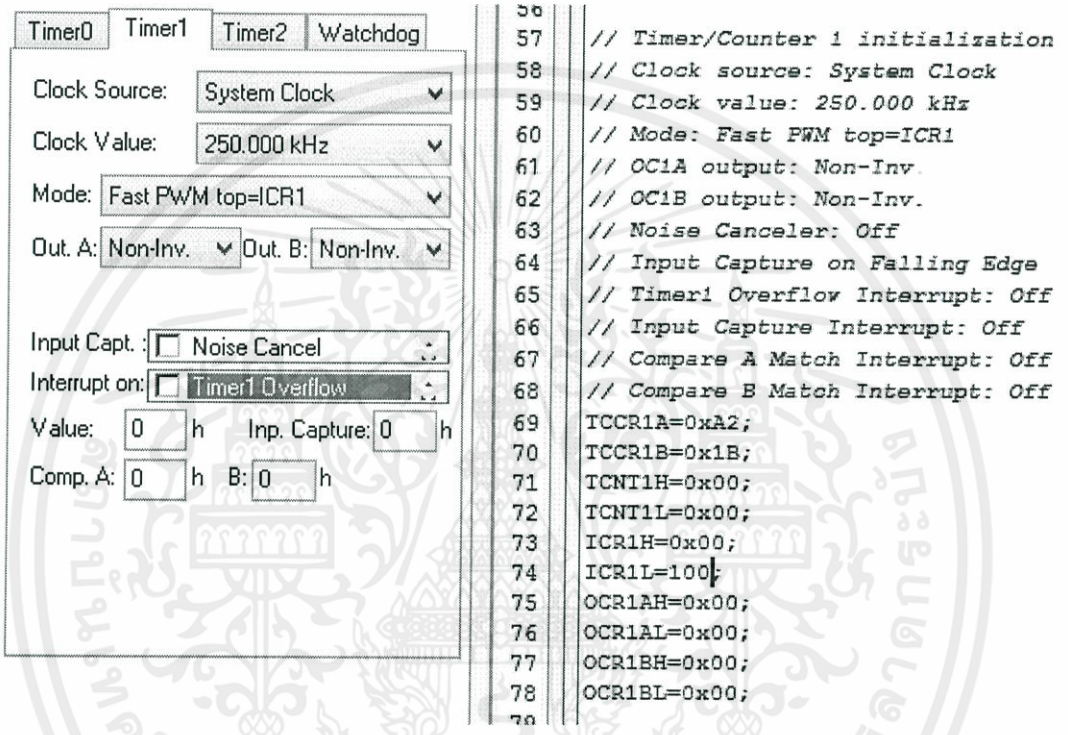
รูปที่ 3.14 แสดงสัญญาณชนิดต่างๆ



รูปที่ 3.15 การเลือกเซ็ตค่า Register สำหรับ ADC โดยใช้โปรแกรม CodeVisionAVR

3.6 การควบคุมความเร็วของมอเตอร์กระแสตรงหรือดีซีมอเตอร์ (DC Motor)

การเขียนโปรแกรมการจ่ายพัลส์ (Pulse) ให้แก่มอเตอร์กระแสตรงหรือดีซีมอเตอร์ (DC Motor) นั้น จะต้องมีไทม์เมอร์ (Timer) เข้ามาเกี่ยวข้องด้วย ซึ่งเราเลือกใช้ Timer/Counter1 เนื่องจาก Timer1 มี 2 ช่อง คือ ช่อง A และ ช่อง B นั่นคือเราสามารถขับมอเตอร์กระตรงทั้ง 2 ตัวได้ในเวลาเดียวกัน



รูปที่ 3.16 การเลือกเซ็ตค่า Register เริ่มต้นของ Timer1 โดยใช้โปรแกรม CodeVisionAVR

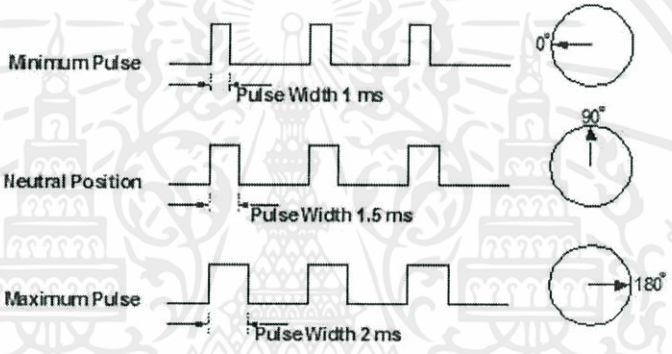
หลักการเขียนโปรแกรมควบคุมสัญญาณ Pulse Width Modulation (PWM) นี้มี Register มีความเกี่ยวข้อง ได้แก่ ICR1L, OCR1AL และ OCR1BL โดยที่เราจะตั้งค่า ICR1L เท่ากับ 100 ซึ่งเป็นเกณฑ์เอาไว้เทียบกับ OCR1AL และ OCR1BLเพื่อที่จะเอาไว้กำหนดความเร็วของมอเตอร์ เพราะฉะนั้นเราจึงสามารถปรับค่า OCR1AL และ OCR1BL ได้ตั้งแต่ 0 ถึง 100 เป็นสัญญาณ PWM จ่ายให้กับมอเตอร์ต่อไป

3.7 หลักการทำงานของเซอร์โวมอเตอร์

การหมุนของเซอร์โวมอเตอร์จะถูกควบคุมด้วยพัลส์หรือลอจิก 1 (5v) เป็นระเวลาที่กำหนด และหน่วงเวลาหรือส่งลอจิก 0 เป็นระยะเวลาคงที่ 20 มิลลิวินาทีสลับกันไป ส่งผลให้สามารถควบคุมการหมุนของเซอร์โวมอเตอร์ได้โดยทิศทางการหมุนเป็นดังนี้

เซอร์โวมอเตอร์จะหมุนไปทางขวา (ตามเข็มนาฬิกา) เมื่อพัลส์บวกมีความกว้าง 1 มิลลิวินาที (1 ms) และพัลส์ลบ 20 มิลลิวินาที

เซอร์โวมอเตอร์จะหมุนไปทางซ้าย (ทวนเข็มนาฬิกา) เมื่อพัลส์บวกมีความกว้าง 2 มิลลิวินาที และพัลส์ลบ 20 มิลลิวินาที



รูปที่ 3.17 อธิบายการทำงานของเซอร์โวมอเตอร์ตามพัลส์ที่ได้รับ

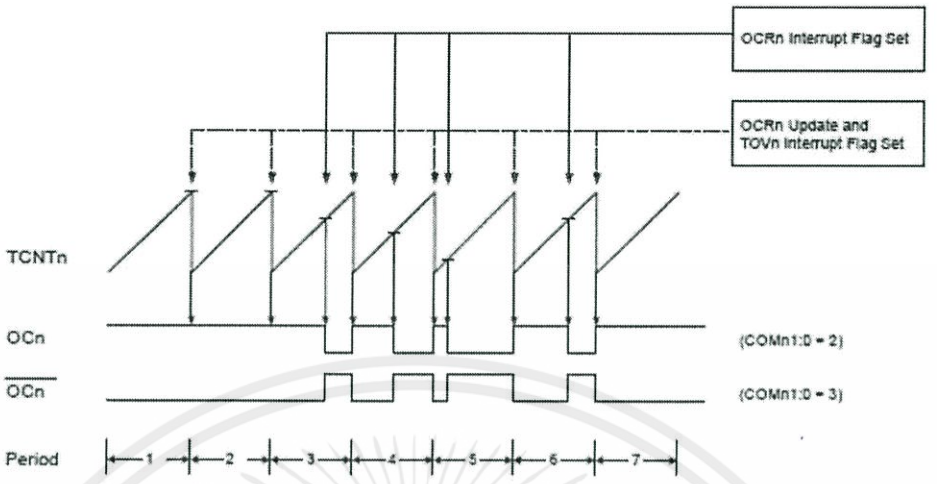
หลักการในการขับเซอร์โวมอเตอร์ คือ มีคาบการทำงาน 20 ms ช่วงเวลาที่อ่านค่าคือ 1-2 ms แรก

โดยที่ 1.0 ms เป็นมุมที่เซอร์โวมอเตอร์บิดซ้ายสุด

1.5 ms เป็นมุมที่เซอร์โวมอเตอร์บิดตำแหน่งกลาง

2.0 ms เป็นมุมที่เซอร์โวมอเตอร์บิดขวาสุด

Fast PWM mode



รูปที่ 3.18 อธิบายการเปรียบเทียบสัญญาณ

เป็นการเปรียบเทียบกันระหว่างสัญญาณ The Timer/Counter (TCNTn) ซึ่งเป็นสัญญาณรูปฟันเลื่อยกับ Output Compare Register (OCRn) ได้เป็น Output Compare (OCn) ค่า OCn จะเกิดการเปลี่ยนแปลงในช่วงเวลาที่ TCNTn และ OCRn มีการแตะกัน และจะกลับมาอยู่ในสถานะแรกเริ่มอีกครั้ง เมื่อถึงช่วงเวลาที่ TCNTn อยู่ในตำแหน่งสูงสุดในโครงงานนี้

เราใช้ Timer/Counter 0 ทำหน้าที่ในการขับเซอร์โวมอเตอร์ 2 ตัว เพื่อแยกโมดูลการทำงานระหว่างเซอร์โวมอเตอร์ และ DC มอเตอร์ เนื่องจาก Timer/Counter 0 จำกัด MAX ของ TCNT0 ที่ 256 ซึ่งยากต่อการเขียนโปรแกรมและการคำนวณ เพื่อให้ง่ายต่อการคำนวณ เราจึงสร้างฟังก์ชั่นการเปรียบเทียบสัญญาณใหม่ โดยตั้งค่า MAX ไว้ที่ 2000 และ OCR อยู่ในช่วง 100 – 200

เพื่อให้ได้คาบ 20 ms ที่ Servo สามารถทำงานได้ เราจึงใช้สูตรการคำนวณ เพื่อใช้ในการเขียนโปรแกรม

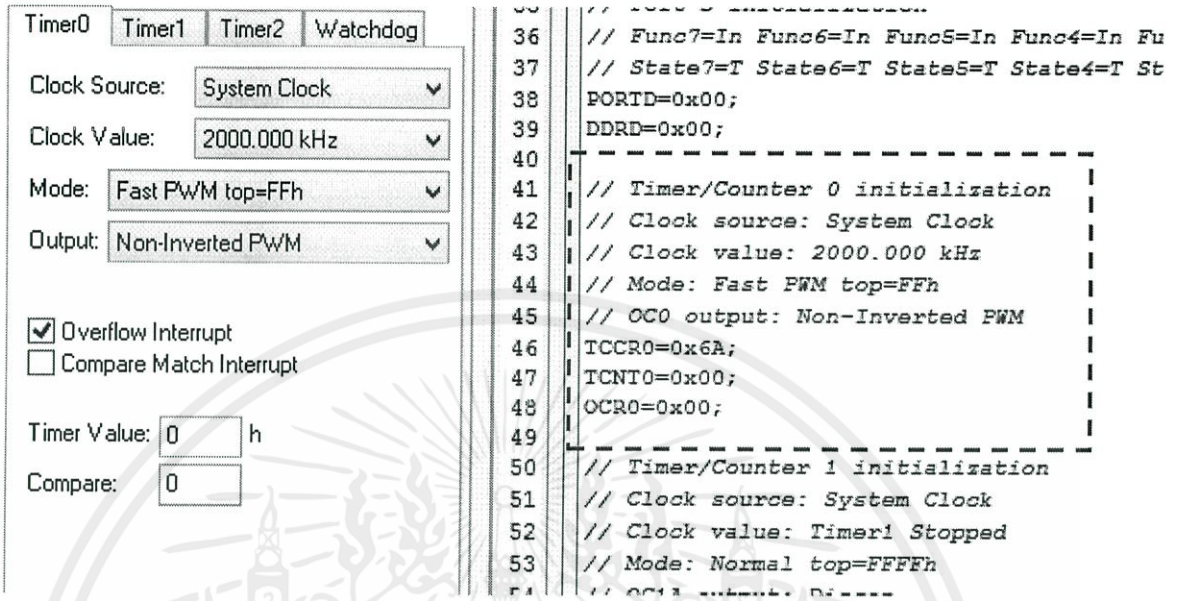
จากสูตรซึ่งดัดแปลงจากในดาต้าชีท $f_{OC0PWM} = \frac{fclk\_I/O}{N \cdot Max}$; N = Prescale Factor

ค่า N เท่ากับ 8 จากตาราง42 ในดาต้าชีทและ ความถี่ Clock 16 MHz

$$f_{OC0PWM} = \frac{fclk\_I/O}{N \cdot 2000} = \frac{16{,}000{,}000}{8 \cdot 2000} = 1 \text{ kHz}$$

Clock Value ที่ใช้คือ 2,000 kHz และ Max ของสัญญาณ TCNT0 เท่ากับ 2,000

ดังนั้นความถี่ใน 1 Cycle คือ f = 2 MHz/2000 = 1 kHz ซึ่งจะต้องตรงกับ f_{OC0PWM}



รูปที่ 3.19 การเลือกเซ็ตค่า Register เริ่มต้นของ Timer0 โดยใช้โปรแกรม CodeVisionAVR

3.8 วิธีการส่งข้อมูลผ่าน Uart

การส่งข้อมูลเชื่อมต่อกันระหว่างไมโครคอนโทรลเลอร์ (Microcontroller) กับคอมพิวเตอร์ (Computer) เราใช้วิธีส่งข้อมูล Serial Frame ผ่านสาย USB

ซึ่งเราสามารถ Set Register ได้ตามตัวอย่างในดาต้าชีท ดังนี้

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}

unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
      ;
  /* Get and return received data from buffer */
  return UDR;
}

void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
      ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

จากโปรแกรมด้านบน จะเห็นว่า ฟังก์ชั่นที่ใช้รับค่าคือ USART_Receive() เมื่อเราเรียกใช้งานฟังก์ชั่นนี้ โปรแกรมจะทำการรอรับค่า จนกว่าจะมีข้อมูล Serial Frame ที่มีเงื่อนไขตรงตามที่กำหนดเท่านั้น จึงจะดำเนินโปรแกรมต่อไป

สำหรับการส่งข้อมูล เราจะใช้ฟังก์ชั่น USART_Transmit(____) ซึ่งข้อมูลที่ถูกส่งคือตัวแปรที่อยู่ในวงเว็บ

การเซ็ตค่า Usart_Init (Baud) จะต้องใส่ไว้ใน Loop Main และ มีพารามิเตอร์คือ Baud ซึ่งแทน Register UBRR เราสามารถคำนวณค่า UBRR จากสูตรที่มีอยู่ในดาต้าชีท

ตาราง 3.1 แสดงการคำนวณค่าของ Register

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

จากสูตรในตารางที่ 3.1 สามารถเอาไปใช้คำนวณ Baud หรือ UBRR ได้

เนื่องจากเรา เลือก Baud Rate 9600 bps เราสามารถหา UBRR ได้จากสูตร เพื่อเอาไปใช้ในโปรแกรม

$$\text{UBRR} = \frac{\text{fosc}}{16\ \text{Baud}} = \frac{16{,}000{,}000}{16 \cdot 9600} = 103$$

หรือสามารถเทียบกับตาราง 3.2

ตาราง 3.2 แสดงค่าของ Register ต่างๆ

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%

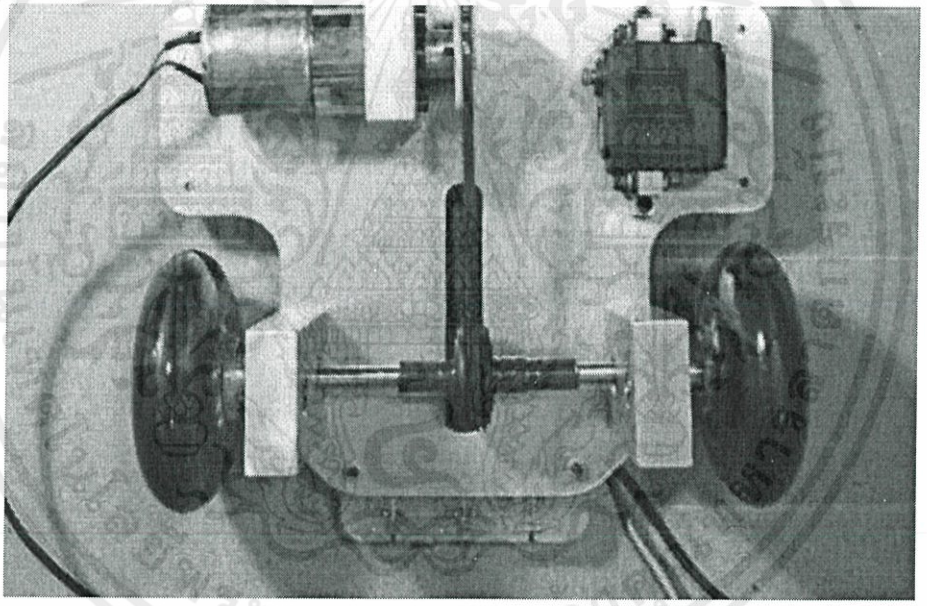
บทที่ 4

การทดลองและผลการทดลอง

4.1 ผลการทดลองการออกแบบ

4.1.1 ระบบดิฟเฟอร์เรนเชียว

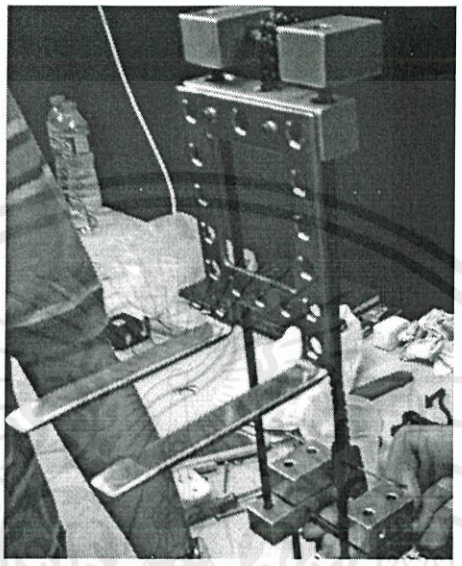
จากการเลือกใช้ระบบดิฟเฟอร์เรนเชียว (Differential) พบว่าสามารถทำให้รถเข้าโค้งได้อย่างนุ่มนวล ไม่เกิดการลื่นไถลของล้อจะทำให้การเข้าโค้งมีประสิทธิภาพสูงสุด แต่อย่างไรก็ตามก็ยังพบปัญหาที่เกิดจากตัวดิฟเฟอร์เรนเชียวเอง ถ้าเราทำการขับน็อตที่ยึดดิฟเฟอร์เรนเชียวแน่นมากเกินไปจะทำให้การหมุนของตัวดิฟเฟอร์เรนเชียวเกิดการสะดุดไม่ต่อเนื่อง แต่ถ้าขันหลวมมากเกินไปจะทำให้ระบบดิฟเฟอร์เรนเชียวไม่สามารถทำงานได้ฉะนั้นเราควรที่จะขันให้พอดีไม่แน่นหรือหลวมจนเกินไป



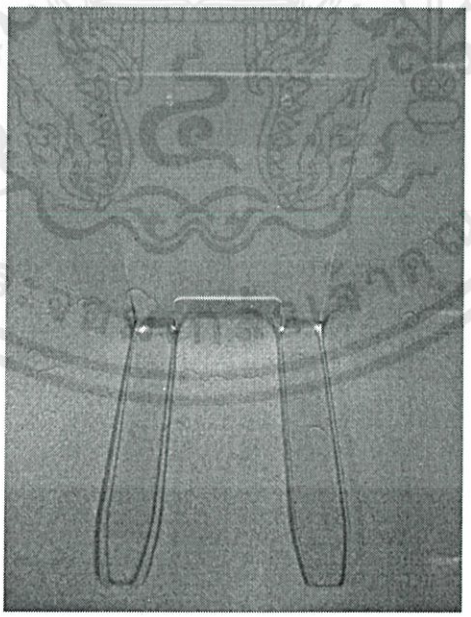
รูปที่ 4.1 แสดงระบบดิฟเฟอร์เรนเชียว

4.1.2 ความแข็งแรงของงา

จากที่ได้ใช้โปรแกรม Solidwork ในการออกแบบงาทางทีมงานได้เลือกใช้สแตนเลสสตีลซึ่งมีความแข็งแรงและคงทนสูง แต่ก็พบปัญหาในเรื่องของน้ำหนัก ทำให้การยกงาเกิดการสะดุด ดังนั้นทีมงานจึงเปลี่ยนวัสดุเป็นอะคลีลิกซึ่งมีน้ำหนักที่เบากว่า

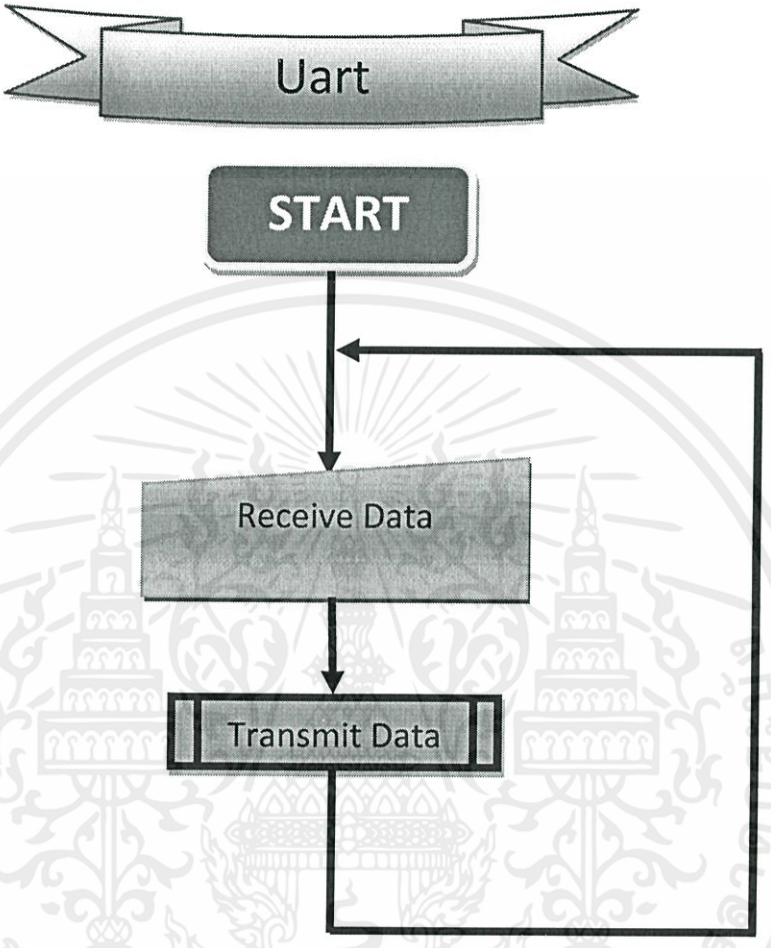


รูปที่ 4.2 งาสแตนเลสสตีล



รูปที่ 4.3 งาอะคลีลิก

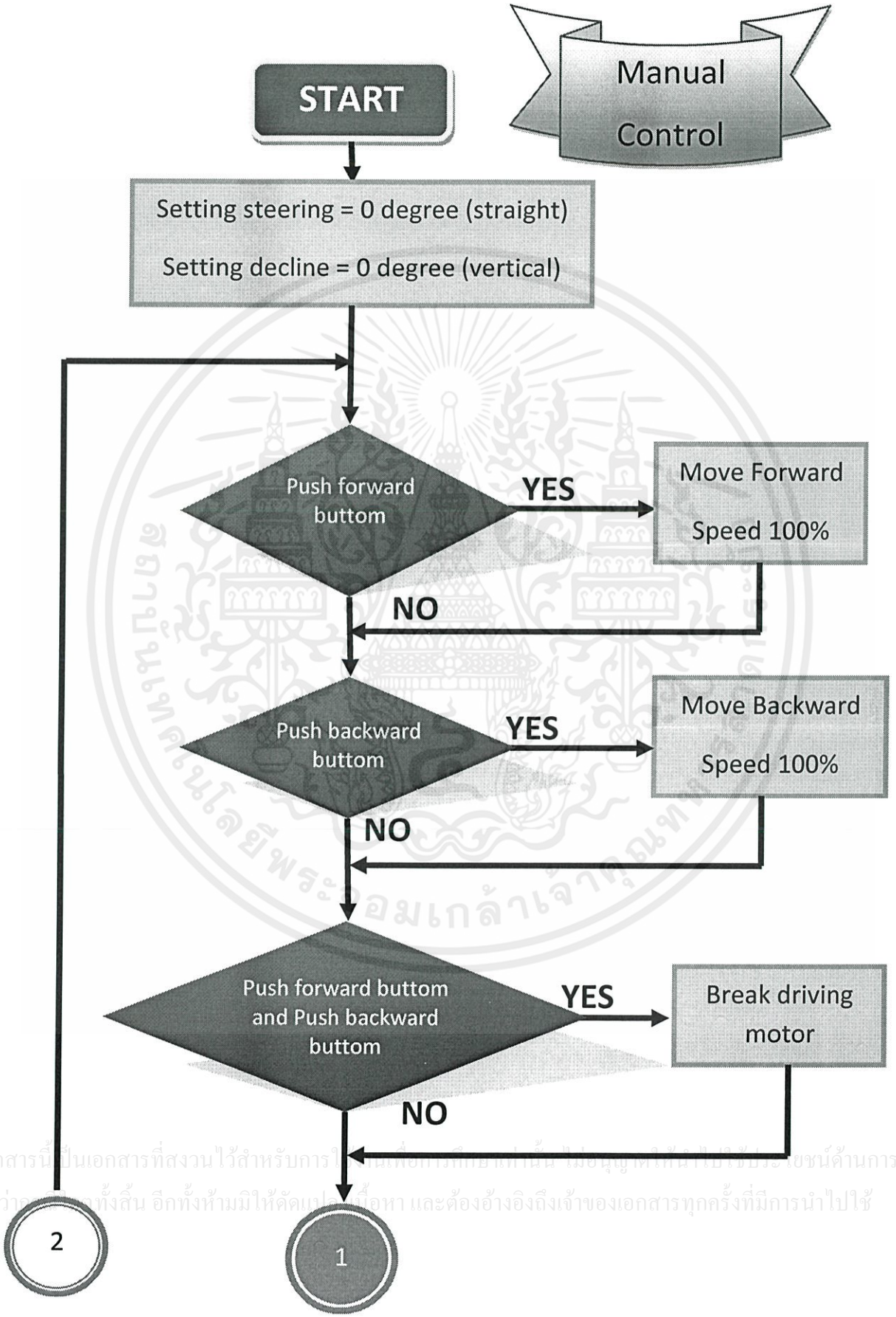
4.2 โปรแกรมทดสอบการรับส่งข้อมูลผ่าน Uart

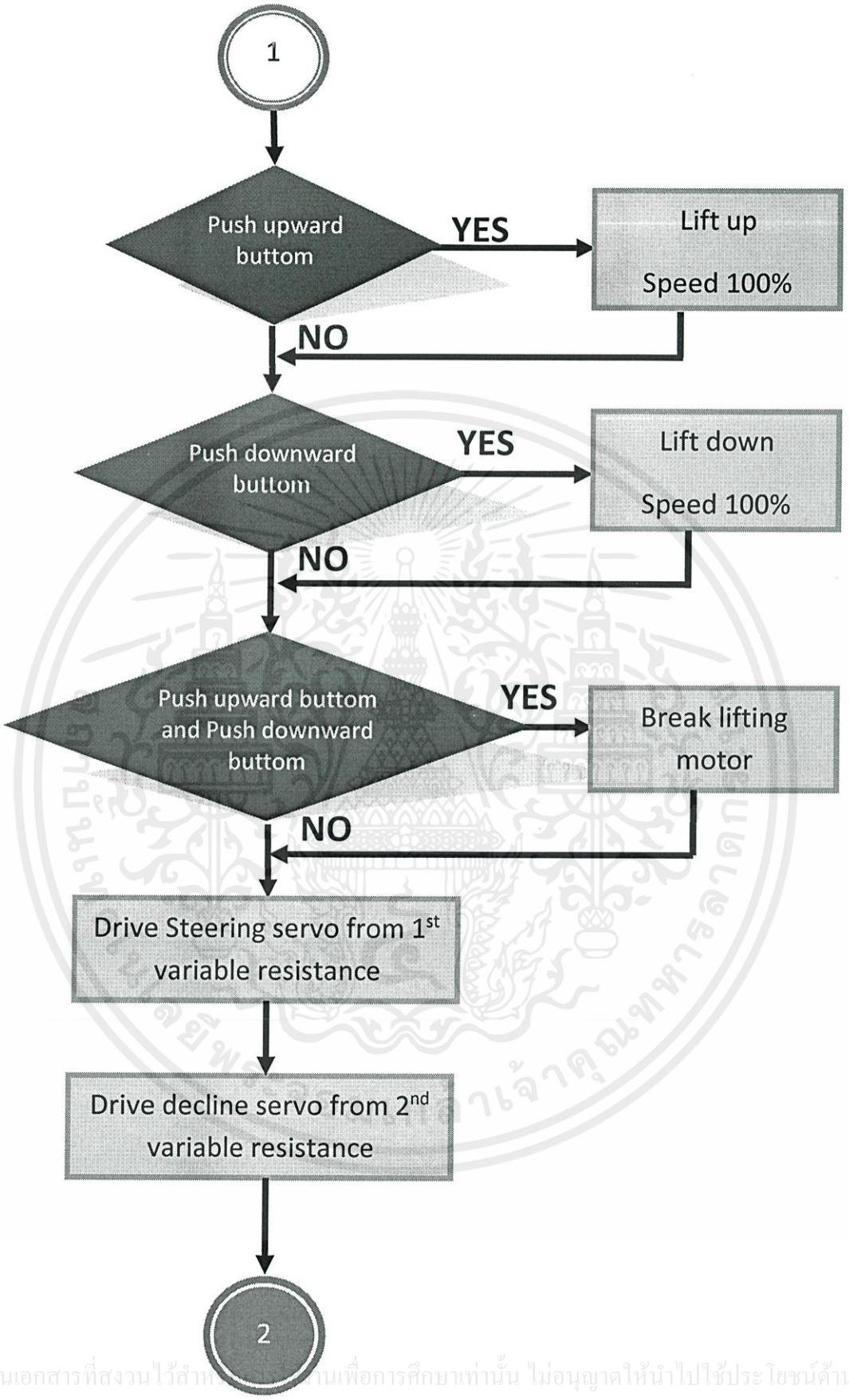


คำอธิบายโปรแกรม

จากโปรแกรมข้างต้น เมื่อรันโปรแกรมและเปิดซอฟแวร์ Interface เช่น X-CTU, Hercules ฯลฯ ขึ้นมา โปรแกรมจะรอรับข้อมูลจนกว่าเราจะป้อนอักขระหรือตัวเลขเข้าไป โปรแกรมจะทำการบันทึกค่าไว้ในตัวแปร Data หลังจากนั้นก็จะส่งข้อมูลเดิมกลับมาในหน้าซอฟแวร์ Interface

4.3 โปรแกรมบังคับด้วยมือ (Manual Control)

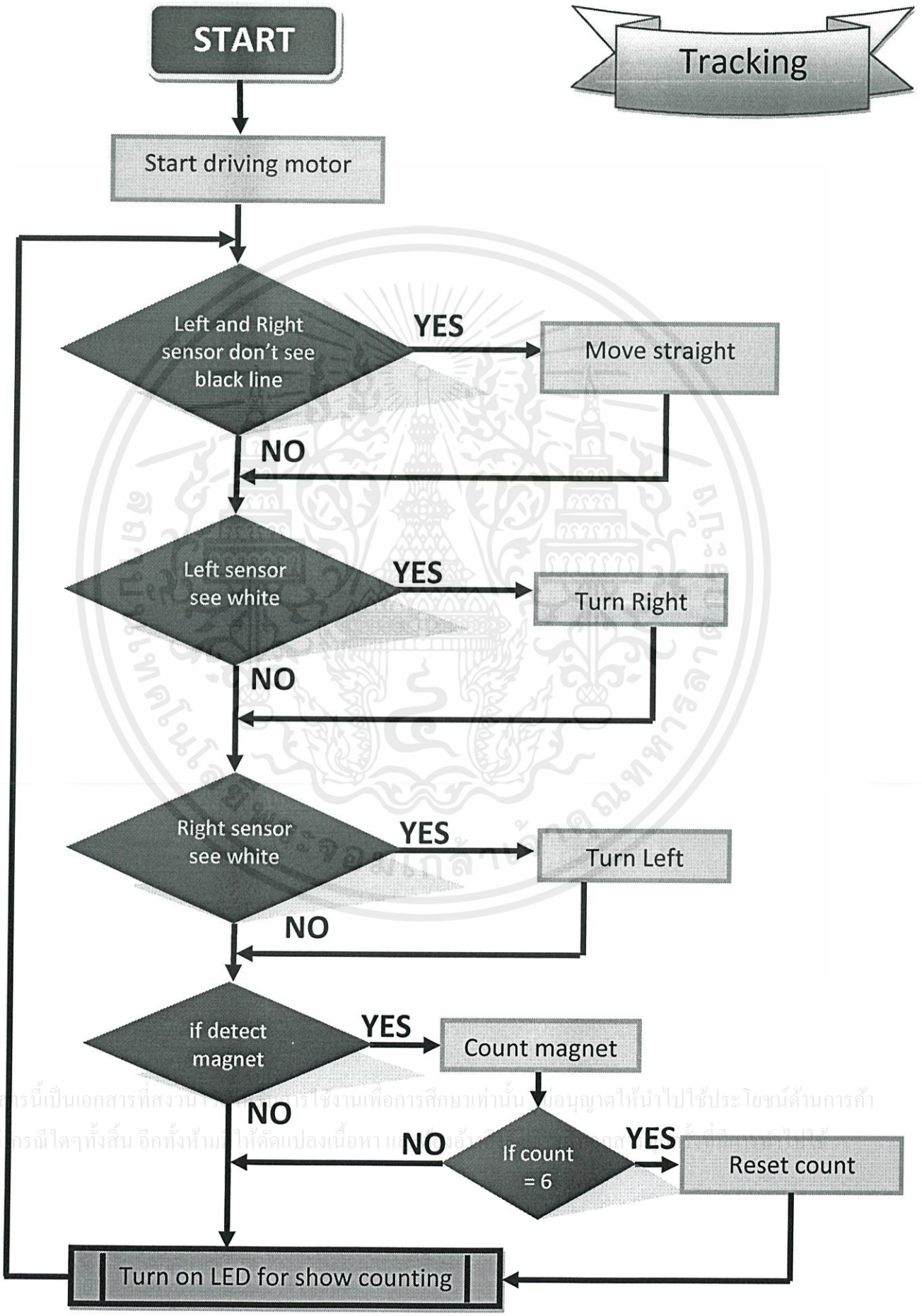




คำอธิบายโปรแกรม

โปรแกรมนี้ทำขึ้นเพื่อให้คนสามารถบังคับและควบคุมรถได้โดยตรง เมื่อเกิดเหตุฉุกเฉินจากการทำงานแบบอัตโนมัติ โดยใช้อุปกรณ์ที่มีปุ่มที่เป็นสวิทช์ซึ่งให้สัญญาณแบบดิจิตอลและปุ่มที่เป็น VR ซึ่งให้สัญญาณแบบอนาล็อก ฟังก์ชันการทำงานมีดังนี้ คือ เมื่อกดปุ่ม B3 รถจะเคลื่อนที่ไปข้างหน้า กดปุ่ม B4 รถจะถอยหลัง แต่เมื่อกดทั้ง B3 และ B4 ในเวลาเดียวกัน รถจะเบรคโดยการล็อคมอเตอร์ เช่นเดียวกันเมื่อกดปุ่ม B5 งารถจะเลื่อนขึ้น กดปุ่ม B6 งารถจะเลื่อนลง และเมื่อกดทั้ง B3 และ B4 ในเวลาเดียวกัน ส่วนปุ่ม VR จะไปบังคับเซอร์โวมอเตอร์ให้หมดตามที่เราบิด ซึ่งต้องผ่านการแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล (ADC) จากนั้น Microcontroller จะประมวลผลแปลงเป็นสัญญาณ Pulse จ่ายให้เซอร์โวมอเตอร์ต่อไป

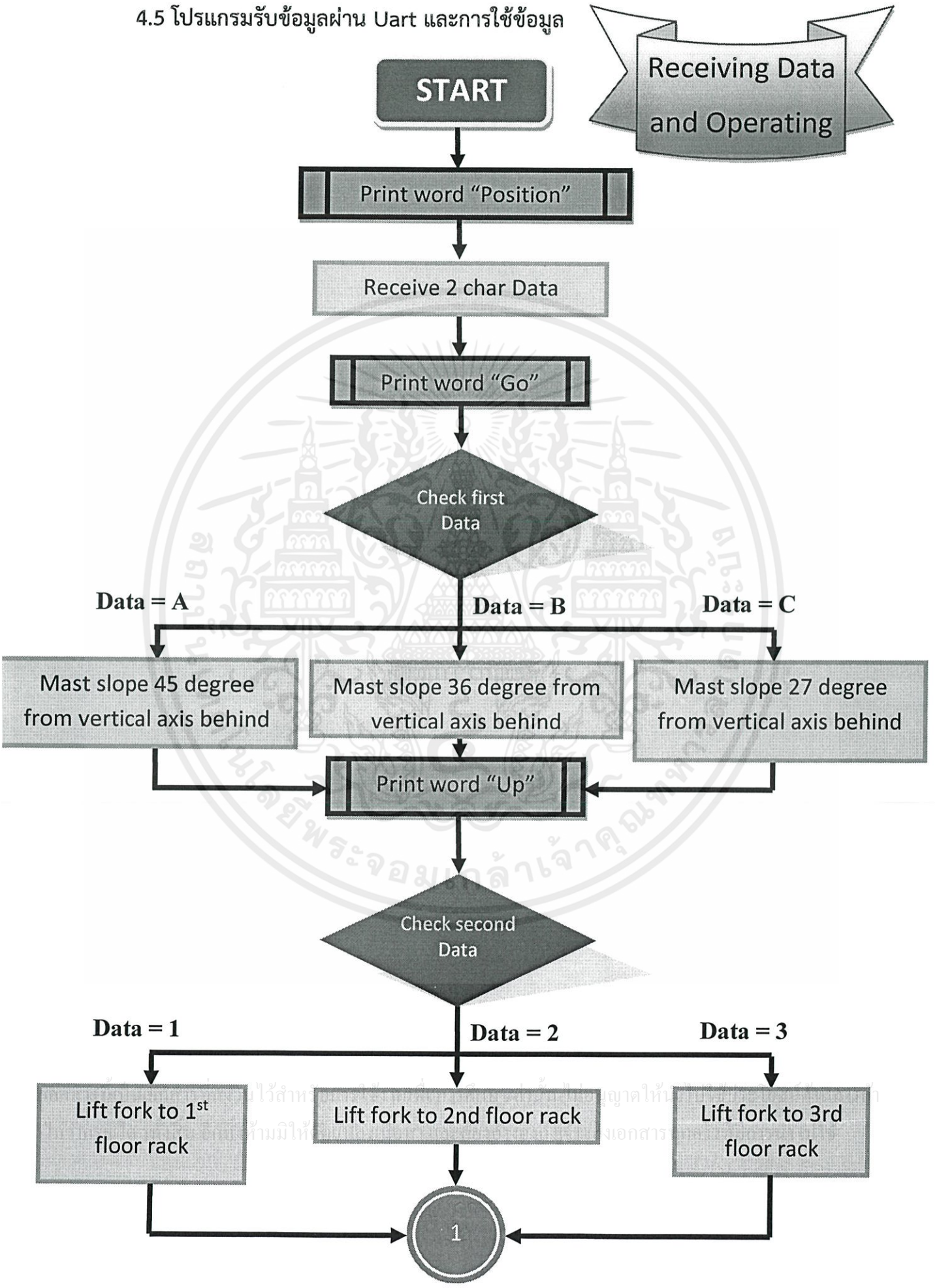
4.4 โปรแกรมเดินตามเส้น (Tracking)

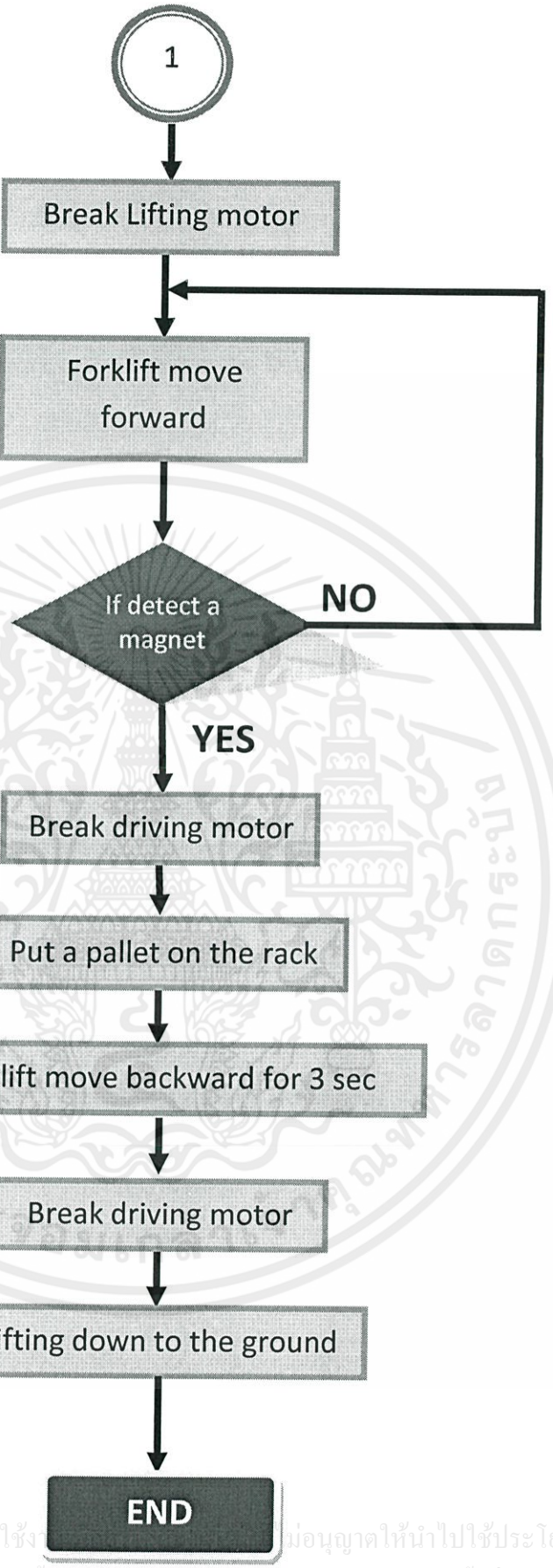


คำอธิบายโปรแกรม

โปรแกรมนี้จะเกี่ยวกับการเคลื่อนที่ของรถ โดยมีเส้นสีดำเป็นตัวกำหนดเส้นทางการเคลื่อนที่ ในการตรวจจับเส้นสีดำ เราใช้เซนเซอร์แสง (Optical Sensor) ทั้งหมด 3 ตัว โดยมีเงื่อนไขว่า เซนเซอร์ทั้งหมดต้องอยู่ในตำแหน่งเส้นสีดำเสมอ เมื่อมีเซนเซอร์ตัวซ้ายหรือตัวขวาตัวใดตัวหนึ่งหลุดเส้น Microcontroller จะไปสั่งเซอร์โวมอเตอร์ชักเลี้ยว ให้หักล้อกลับเป็นมุมใกล้เคียง 80 องศา เพื่อให้รถคงอยู่ในเส้นทางได้ นอกจากนี้โปรแกรมนี้ได้มีการตรวจจับเซนเซอร์แถบแม่เหล็กอยู่ด้วย เพื่อระบุตำแหน่งของรถ โดยมีการนับว่าได้ผ่านแม่เหล็กไปกี่จุดแล้ว

4.5 โปรแกรมรับข้อมูลผ่าน Uart และการใช้ข้อมูล

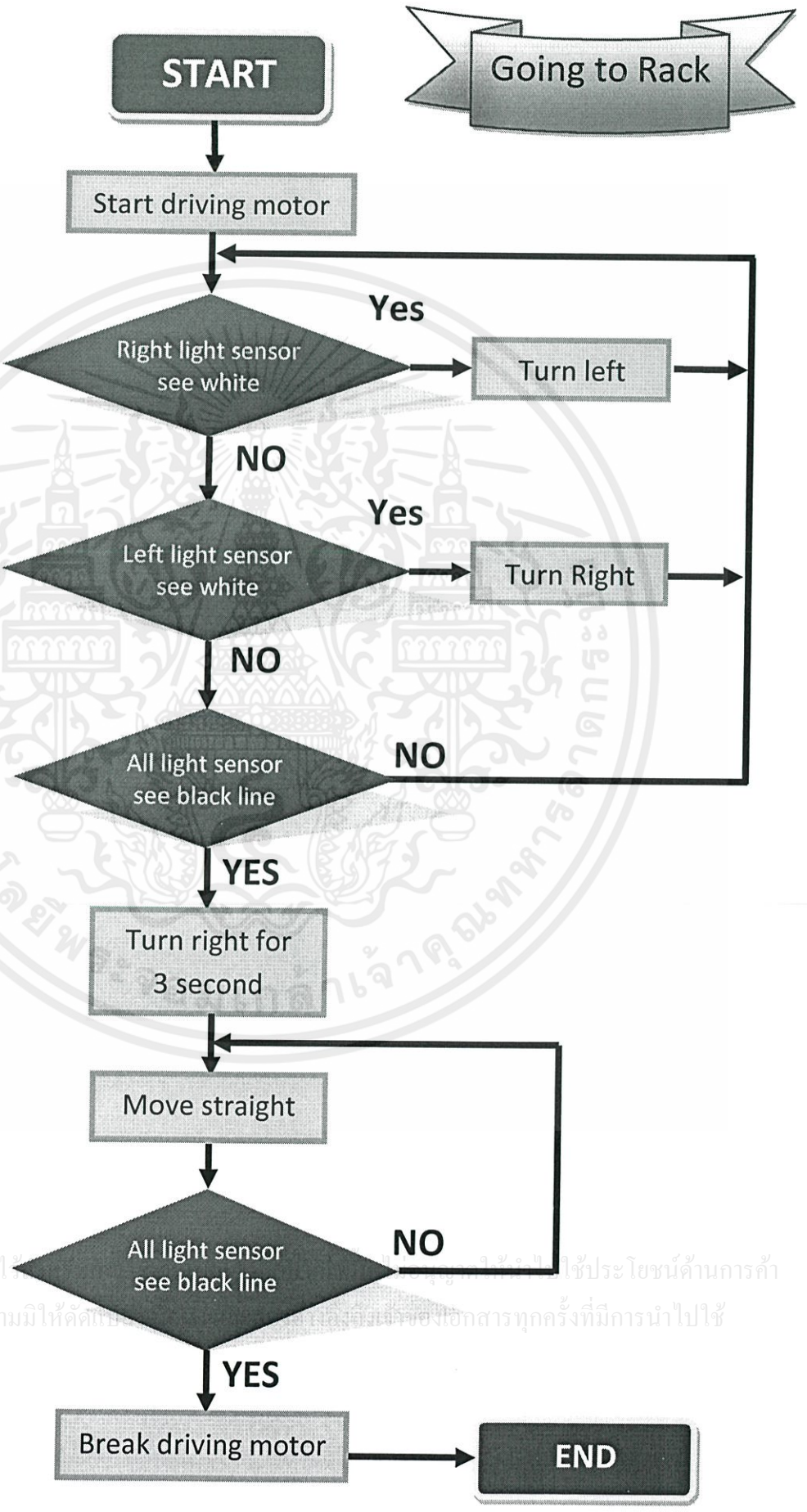




คำอธิบายโปรแกรม

โปรแกรมนี้เป็นการทดสอบการใช้ข้อมูลอักขระที่ได้จากคอมพิวเตอร์ผ่าน Uart โดยเริ่มโปรแกรมเชื่อมต่อกับซอฟแวร์อินเตอร์เฟสไมโครคอนโทรลเลอร์จะส่งอักขระ P, o, s, i, t, i, o, n หลังจากนั้นจะรอรับอักขระ 2 ตัว จากคอมพิวเตอร์ไปเก็บไว้ในตัวแปร data[0] และ data[1] ตามลำดับ เมื่อได้รับข้อมูลที่ตรงตามเงื่อนไข ไมโครคอนโทรลเลอร์จะประมวลผลและส่งข้อความว่า Go และตามได้ค่าที่อยู่ในตัวแปร data[0] ตามด้วย Up และตามได้ค่าที่อยู่ในตัวแปร data[1] จากนั้นไมโครคอนโทรลเลอร์จะทำการเช็คตัวแปร ที่อยู่ใน data[0] และ data[1] ว่าตรงตามเงื่อนไขที่ได้ตั้งโปรแกรมไว้หรือเปล่า คือ ถ้า data[0] เป็นอักขระ A เซอร์โวที่ควบคุมการเอียงของเสาจะบิดไป ค่าๆ หนึ่ง ตาม Pulse ของเซอร์โว อย่างที่ได้เคยอธิบายไว้ก่อนแล้ว อักขระ B และ C ก็เป็นไปตามเงื่อนไขของมันเช่นเดียวกัน นอกเหนือจากนี้ถือว่าเซอร์โวไม่มีการเปลี่ยนแปลง จากนั้นพิจารณา data[0] จะเกี่ยวกับการยกงาขึ้นไปในระดับชั้นต่างๆ ถ้าอักขระเป็น 1, 2 และ 3 จะถูกยกขึ้นไปชั้น 1 ชั้น 2 และชั้น 3 ตามลำดับ หลังจากนั้นรถจะเคลื่อนที่ไปข้างหน้าช้าๆ เมื่อเจอแถบแม่เหล็กก็จะหยุดช่วงขณะเพื่อวางพาเลท (Pallet) และถอยหลังกลับเป็นเวลา 48,000/16 = 3,000ms หรือ 3 วินาที จากนั้นงารถก็จะถูกเลื่อนลงเรื่อยๆ จนกว่าฐานของเสาจะไปแตะลิมิตสวิทช์ เพื่อสั่งให้มอเตอร์หยุด

4.6 โปรแกรมเมื่อรถเลี้ยวเข้ามาเก็บของ

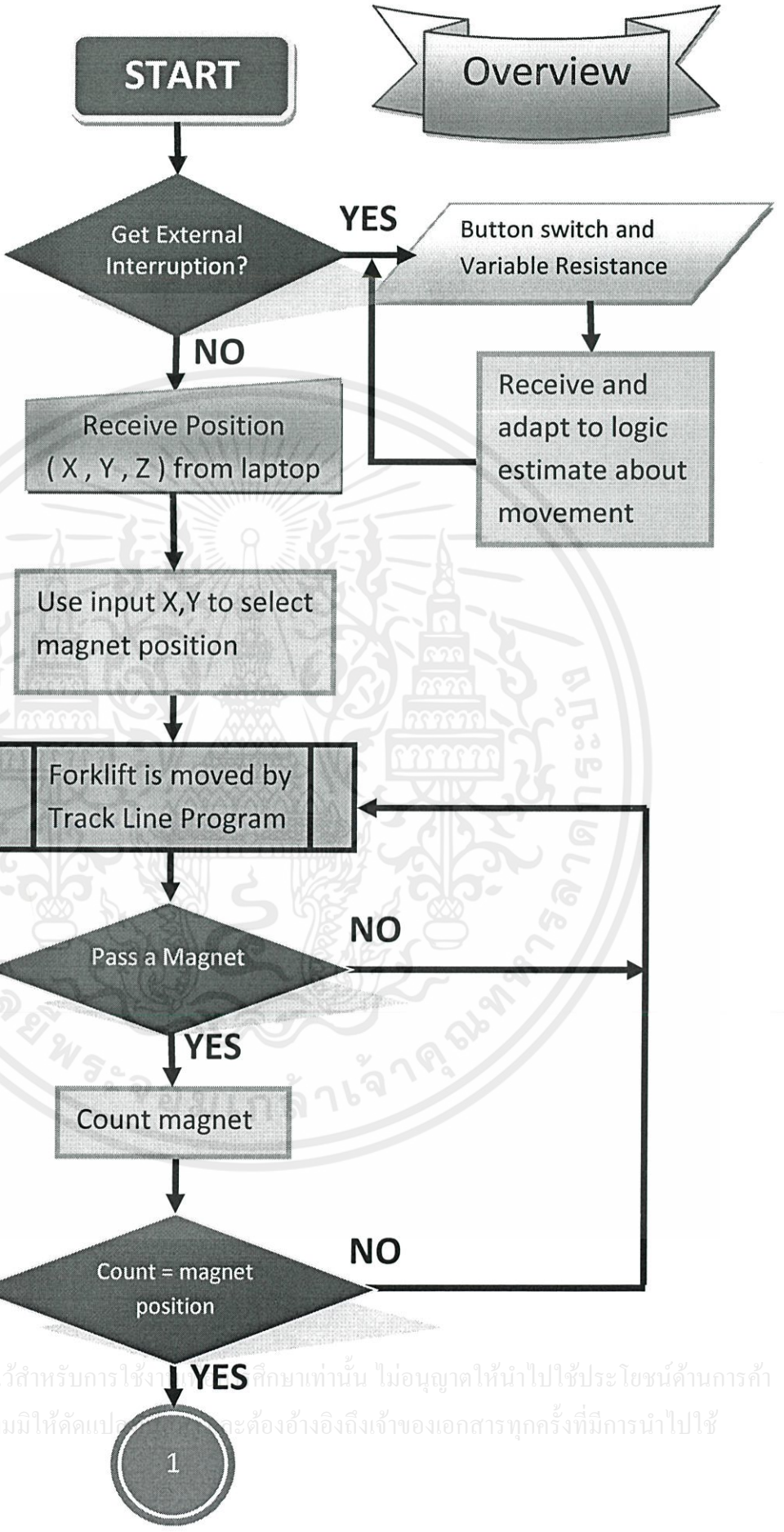


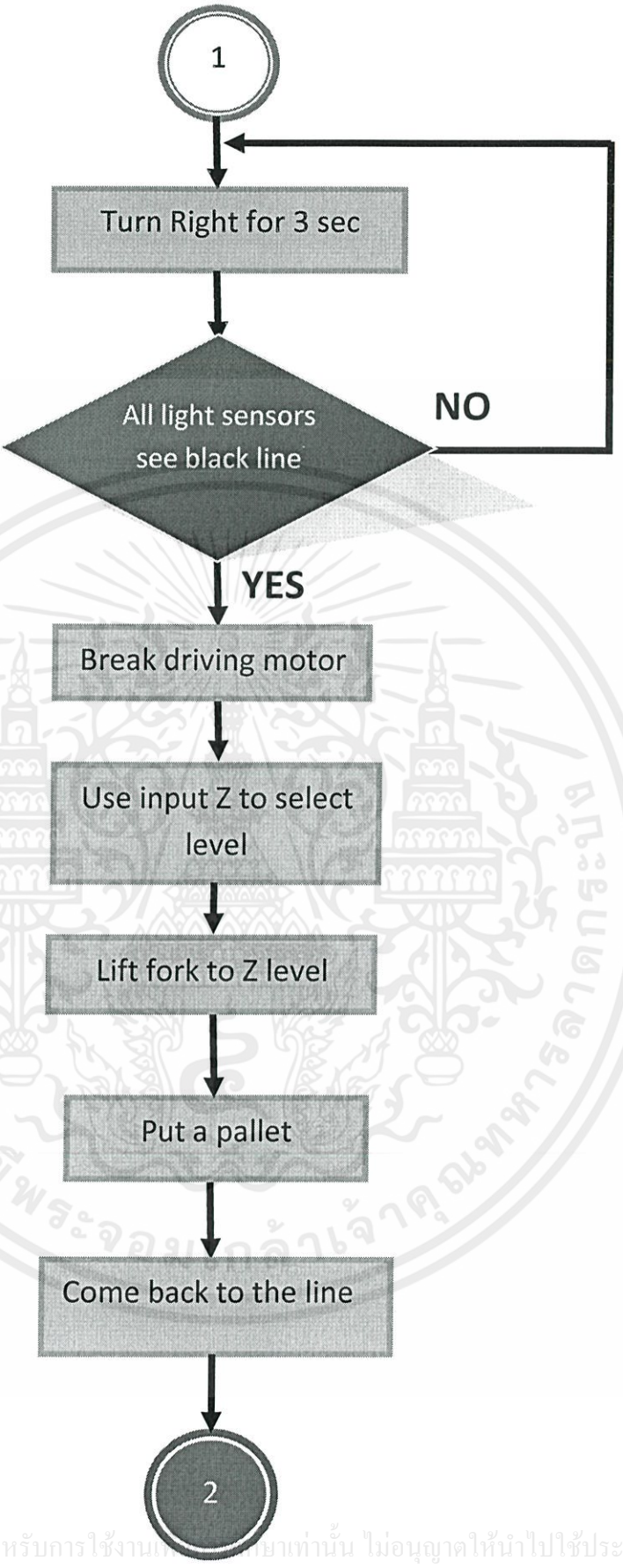
คำอธิบายโปรแกรม

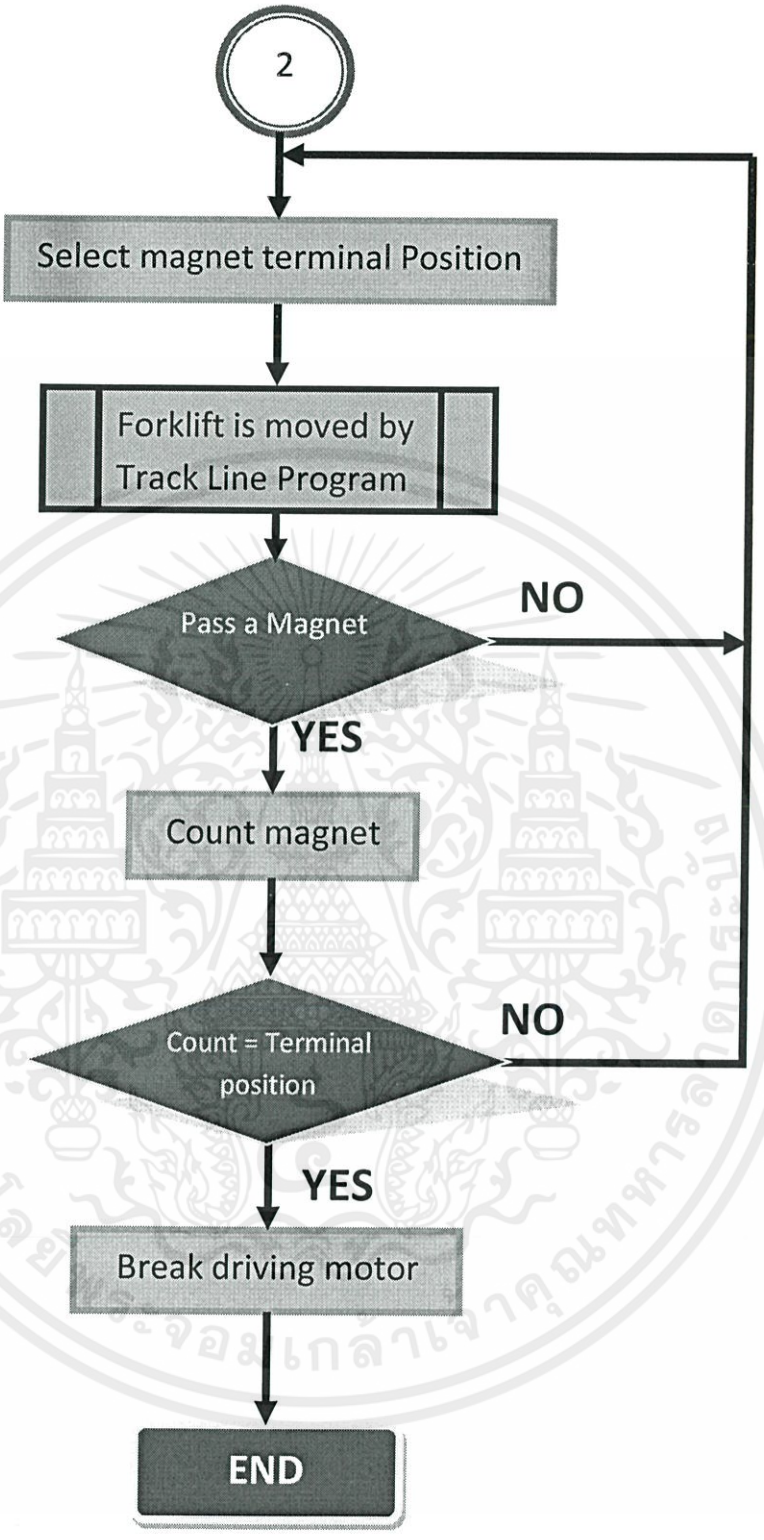
เริ่มต้นรถจะวิ่งตามเส้นเหมือนโปรแกรมเดินตามเส้นเช่นที่ได้เสนอไปแล้วนั้น แต่เมื่อเซนเซอร์แถบแม่เหล็กตรวจจับแม่เหล็กได้ โปรแกรมจะถูกเปลี่ยนให้รถเลี้ยวขวาจนกว่าเซนเซอร์แสงจะเจอเส้นสีดำพร้อมกันทั้ง 3 ตัว และมอเตอร์จะถูกสั่งให้หยุด

4.7 การทำงานโดยรวม

โหมดการทำงานของรถโฟร์คลิฟท์อัจฉริยะนี้ มีทั้งโหมดอัตโนมัติและโหมดบังคับด้วยมือ เราสามารถเลือกโหมดได้โดยการเสียบสายรีโมทคอนโทรลที่เราได้ทำขึ้นเองโดยเฉพาะ เพื่อเลือกเป็นโหมดบังคับด้วยมือ ซึ่งสามารถควบคุมรถได้อย่างอิสระ แต่ถ้าไม่เสียบหลังจากกดสวิทช์เปิดเครื่องโปรแกรมจะถือว่าเป็นโหมดอัตโนมัติทันที ซึ่งในโหมดนี้ไมโครคอนโทรลเลอร์จะรอรับพารามิเตอร์ตำแหน่ง 3 ตัว นั่นคือ X, Y และ Z ตามลำดับ เมื่อเราใส่อักขระพารามิเตอร์ทั้ง 3 ตัว เรียบร้อยแล้ว เราสามารถถอดสาย USB ที่ใช้เชื่อมต่อกันระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ได้เลย เนื่องจากข้อมูลได้ถูกส่งและบันทึกแล้ว พร้อมกันนี้ไมโครคอนโทรลเลอร์ประมวลผลข้อมูลที่ได้รับมาออกมาเป็นจำนวนแม่เหล็กที่รถต้องวิ่งผ่านและชั้นที่จะไปเก็บของ เริ่มต้นจากรถตักสินค้าจากสถานีแล้วเข้าสู่เส้นทางเดินหลัก จากนั้นจะเคลื่อนผ่านแถบแม่เหล็กไปเรื่อยๆ จนกว่าจะถึงตำแหน่งแม่เหล็กที่ถูกเลือกไว้ จากนั้นรถก็จะเลี้ยวเข้าไปหาชั้นวางของ ก่อนถึงชั้นวางของจะมีแถบเส้นสีดำแตะตามขวางอยู่ ถ้าเซนเซอร์แสงทั้ง 3 ตัว จับเส้นสีดำได้พร้อมกัน โปรแกรมจะสั่งให้รถหยุด เพื่อที่จะทำการยกงาให้ขึ้นไปยังตำแหน่งที่ถูกเก็บค่าไว้ในตอนแรก ในขณะที่ยกงาขึ้นนั้น จะมีเอ็นโค๊ดเดอร์ (Encoder) เป็นเสมือนตัวเช็คความสูงของงาในเวลานั้นๆ เมื่อถึงความสูงที่กำหนด มอเตอร์ที่ยกงาก็จะหยุด จากนั้นรถก็จะเดินหน้าเข้าไปวางของแล้วก็ถอยกลับมาตำแหน่งเดิม จากนั้นงาก็จะเลื่อนลงเรื่อยๆ จนกว่าจะไปแตะลิมิตสวิทช์ (Limit Switch) ที่อยู่ข้างล่างของเสา หลังจากที่งาหยุดรถก็จะถอยกลับมายังเส้นทางเดินรถตำแหน่งเดิมก่อนที่จะเลี้ยวเข้าชั้นวางของ จากนั้นรถจะวิ่งตามเส้นไปเรื่อยๆ ผ่านแถบแม่เหล็กเพื่อกลับไปยังสถานีเริ่มต้น จำนวนแม่เหล็กที่รถจะต้องผ่านในขากลับคำนวณได้จากการเอาจำนวนแม่เหล็กทั้งหมด (2X) ลบด้วยจำนวนแม่เหล็กที่ได้ผ่านมาก่อนแล้ว เมื่อถึงสถานีรถก็จะหยุดการทำงานชั่วคราว เพื่อรอรับคำสั่งต่อไป







บทที่ 5

บทวิจารณ์และสรุป

5.1 สรุปผลการทดลอง

จากการทดลองระบบปฏิบัติการโฟร์คลิฟท์อัจฉริยะ รถโฟร์คลิฟท์สามารถส่งสินค้าไปยังที่ต้องการได้ แต่ยังต้องการการพัฒนาระบบการสื่อสารเพื่อให้สามารถควบคุมรถได้ง่ายขึ้น ปัจจัยความไม่เสถียรในการตรวจจับการเคลื่อนที่ไปยังจุดหมายของเซนเซอร์แสงยังมีข้อจำกัด ในเรื่องของแสงสว่างในห้องซึ่ง ณ เวลาต่างๆ กันค่าที่ได้จากเซนเซอร์อาจมีค่าไม่เท่ากัน ทำให้การตรวจจับเส้นมีความผิดพลาดอยู่เล็กน้อย แต่เนื่องจากการแก้ไขเปลี่ยนไปใช้เซนเซอร์แม่เหล็กทำให้สามารถหยุด ณ จุดที่ต้องการได้ จึงทำให้รถโฟร์คลิฟท์สามารถส่งของไปยังจุดที่ต้องการได้แม่นยำมากขึ้น

5.2 ปัญหาที่พบและแนวทางแก้ไข

จากการใช้ระบบเซนเซอร์แสง ตรวจจับเส้นทางบนพื้นเพื่อเดินไปยังตำแหน่งเป้าหมาย พบว่าระบบสามารถเคลื่อนที่ไปตามเส้นทางที่กำหนดได้จริง แต่มีความผิดพลาดของตำแหน่งที่จะหยุดที่ตำแหน่งกำหนดอยู่บางครั้ง เนื่องจากปัจจัยทางความเร็ว และการตอบสนองทางตรรกะ ทำให้เซนเซอร์แสงจะสามารถอยู่นอกเหนือตรรกะที่กำหนดได้ ส่งผลให้ไม่หยุดในตำแหน่งที่ควรจะหยุด เนื่องจากสามารถเคลื่อนที่ไปตามเส้นทางที่กำหนดอย่างเสถียรแล้ว จึงใช้เซนเซอร์แม่เหล็กตรวจจับแม่เหล็กในตำแหน่งที่ควรจะหยุดแทน เพื่อลดปัญหาความซับซ้อนของโปรแกรม ลดข้อผิดพลาด แยกหน้าที่ของอุปกรณ์

ในส่วนปัญหาที่พบและยังไม่ได้ปรับปรุงแก้ไข ได้แก่ ปัญหาระบบไฟฟ้าของระบบการสื่อสารแบบไร้สาย (Wifi) เนื่องจากไฟเลี้ยงของระบบการสื่อสารไร้สายและไฟเลี้ยงคอนโทรลเลอร์มีค่าไม่เท่ากัน ทำให้เกิดปัญหาการส่งข้อมูลระหว่างคอนโทรลเลอร์และการสื่อสารไร้สาย ซึ่งการแก้ปัญหาดังกล่าวอาจจะต้องแก้ปัญหาที่ต้นเหตุ คือไฟเลี้ยงที่ตัวคอนโทรลเลอร์ จำเป็นต้องแก้ไขแผงวงจรและ IC บางตัว

5.3 ข้อเสนอแนะและแนวทางในการค้นคว้าพัฒนา

ในการควบคุมโฟร์คลิฟท์อัจฉริยะนั้น ควรจะสามารถสั่งการผ่านระบบการสื่อสารแบบไร้สาย (Wifi) ได้เพื่อให้ง่ายต่อการสั่งควบคุมส่งสินค้าไปเก็บที่ช่องไหน และสามารถควบคุมขณะอยู่หน้าจอสั่งการ และควรจะมีโปรแกรมจัดการสินค้าเพื่อตรวจจับว่าสินค้าที่ขนส่งโดยรถโฟร์คลิฟท์นั้นเก็บที่ช่องไหนชั้นไหนและเก็บสินค้า ณ วันไหนเพื่อง่ายต่อการค้นหาสินค้าที่ระบบโฟร์คลิฟท์เก็บสินค้าไว้

นอกจากนั้นโฟร์คลิฟท์ที่ทีมพัฒนาปัจจุบันนั้นมีแค่คันเดียว ไม่ได้วางแผนเพื่อให้ใช้ได้หลายคัน ฉะนั้นจึงควรศึกษาแนวทางให้การพัฒนาให้โฟร์คลิฟท์อัจฉริยะนั้นทำงานได้ในขณะที่มีหลายคัน โดยให้ระบบการสื่อสารไร้สาย (Wifi) สามารถควบคุมได้ทุกคันและมีโปรแกรมในการจัดการสินค้าของโฟร์คลิฟท์ที่ส่งสินค้าทุกคันเพื่อความสะดวกและรวดเร็วในการใช้งาน

เอกสารอ้างอิง

[1] ทฤษฎีระบบการเคลื่อนที่ของรอก :วิระศักดิ์กรัยวิเชียร, "กลศาสตร์ วิศวกรรม ภาคพลศาสตร์"

[2] ทฤษฎี Four-Bar Linkage : Arthur G.Erdman/George N.Sandon, "Mechanism Design Analysis and Synthesis Volume 1"

[3] ทฤษฎี Center of Mass

[4] ทฤษฎีตรวจจับแบบความถี่เหนือเสียงและทฤษฎีตรวจจับแบบใช้แสง
Available :

http://www.ind.cru.in.th/patanaphong/documents/วิชาการวัดและเครื่องมือวัด/บทที่3 เซนเซอร์และวงจรไฟฟ้า-2A.pdf

[5] ทฤษฎี H-Bridge Switching

Available :

http://www.cp.eng.chula.ac.th/~krerk/courses/2110262/2544/LabSheet/Lab5.htm

http://webcache.googleusercontent.com/search?q=cache:HiByrMyYYeEJ:www.ee.buu.ac.th/~acitl/project/2009/Mobility%2520Design%2520For%2520Control%2520Robotic%2520System/Report%26Presentation/edite/10%2520%25BA%25B7%25B7%25D5%25E82.docx+&cd=1&hl=th&ct=clnk&gl=th

[6] ทฤษฎี DC Motor : "PM DC Motor" Industrial Automation Control

[7] Datasheet ATMega32 by Atmel

ภาคผนวก

ภาคผนวก ก.

คุณสมบัติ Microcontroller

1. ATMega32 @ 16.00 MHz

2. 32Kbytes of Flash Program Memory

3. 1Kbytes of EEPROM 100000 write/erase

4. 2Kbytes Internal SRAM

5. 32 Programmable I/O lines

    - 8 channels of 10-bit ADC

    - 1 channels of SPI

    - 1 channels of I2C

    - 1 channels of USART

คุณสมบัติบอร์ด

1. เป็น MCU ตระกูล AVR เบอร์ ATMega32 ของ Atmel ซึ่งเป็น MCU ขนาด 8-bit.

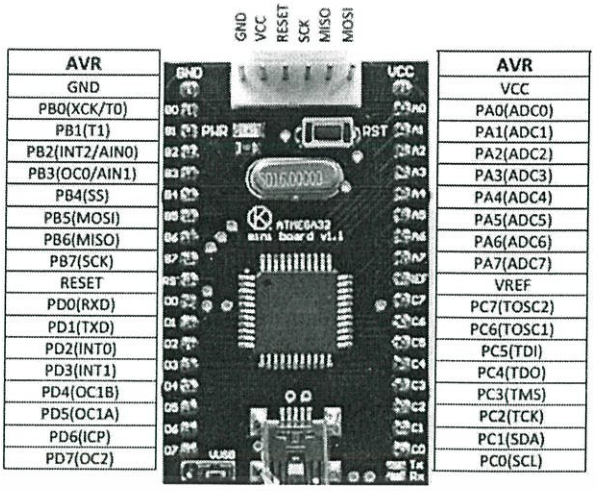
2. ใช้สัญญาณนาฬิกาจาก Crystal ความเร็ว 16MHz

3. สามารถใช้แรงดันได้ตั้งแต่ 4.5 – 5.1V

4. มีหน่วยความจำโปรแกรม 32KB ในโหมดปกติ หรือ 30KB ในโหมด Bootloader

5. สามารถเลือกใช้แรงดัน 5V จาก USB ได้

7. ในโหมด Bootloader สามารถโปรแกรมผ่านสาย USB จากโปรแกรม AVR Studio 4 ได้



พอร์ตต่างๆ ของ Microcontroller AVR ATMega32

PB0 = Available

PB1 = Available

PB2 = Available

PB3 = Forward Motor A Switch

PB4 = Backward Motor A Switch

PB5 = Upward Motor B Switch

PB6 = Downward Motor B Switch

PB7 = Available

PD0 = Uart Receive

PD1 = Uart Transmit

PD2 = Interrupt Manual System

PD3 = Right LED

PD4 = Pulse Motor B

PD5 = Pulse Motor A

PD6 = Center LED

PD7 = Left LED

PA0 = Right Optical Sensor

PA1 = Center Optical Sensor

PA2 = Left Optical Sensor

PA3 = Limit Switch

PA4 = Magnetic Sensor

PA5 = Control Servo C5 Variable Resistant

PA6 = Control Servo C4 Variable Resistant

PA7 = Encoder

PC7 = Available

PC6 = Available

PC5 = Swaying Servo Motor

PC4 = Wheeling Servo Motor

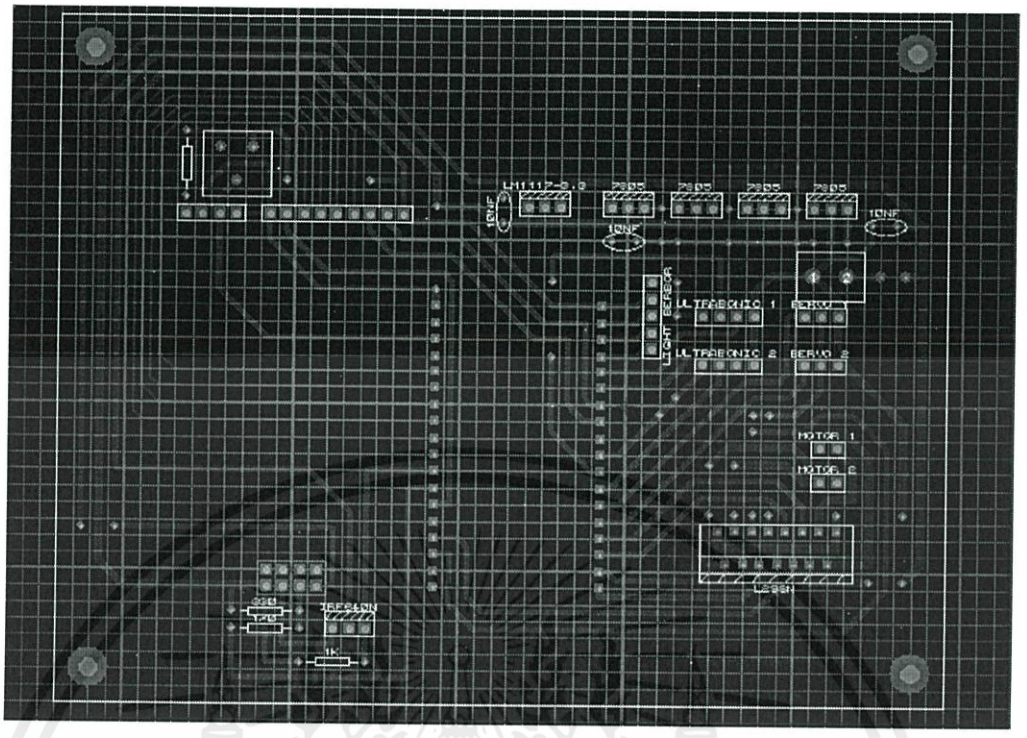
PC3 = Direction Motor B

PC2 = Direction Motor B

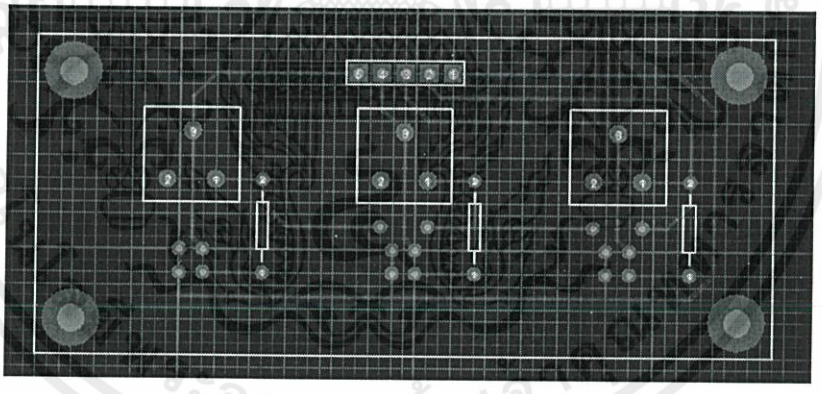
PC1 = Direction Motor A

PC0 = Direction Motor A

ระบุพอร์ตที่ใช้ต่อกับอุปกรณ์ต่างๆ



แผนผังบอร์ด PCB ของวงจรหลัก (เดิม) ที่ได้ออกแบบไว้ ด้วยโปรแกรม ARES Professional

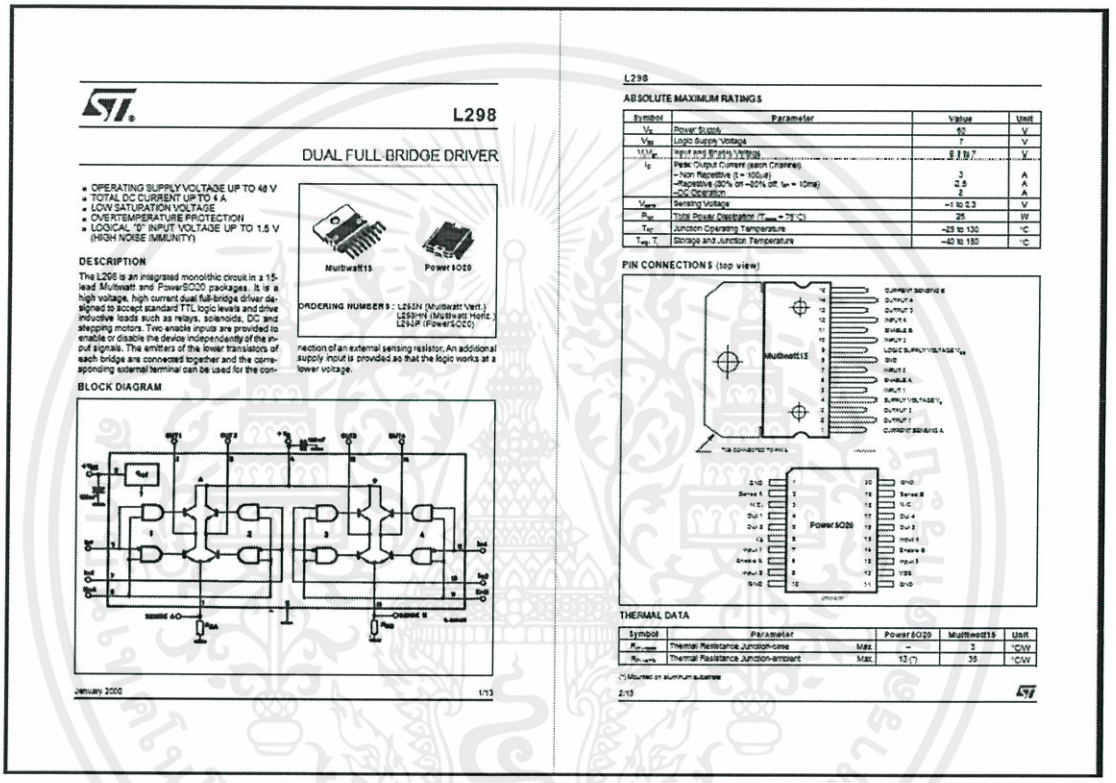


แผนผังบอร์ด PCB ของวงจรเซนเซอร์แสง ที่ได้ออกแบบไว้ ด้วยโปรแกรม ARES Professional

ภาคผนวก ข.

คู่มือการใช้งาน IC L298

IC L298 เป็นอุปกรณ์แปลงแรงดันไฟฟ้าจากไมโครคอนโทรลเลอร์เป็นแรงดันไฟฟ้าที่สามารถใช้ควบคุมมอเตอร์ได้ มีรายละเอียดต่างๆ ดังนี้



L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15 Power SO20

ORDERING NUMBERS : L298N (Multiwatt Vert.) L298HN (Multiwatt Horiz.) L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM

January 2000 1/13

L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	-0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) – Repetitive (80% on –20% off; ton = 10ms) – DC Operation	3 2.5 2	A A A
V_{sens}	Sensing Voltage	-1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	-25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	-40 to 150	°C

PIN CONNECTIONS (top view)

Multiwatt15

Power SO20

THERMAL DATA

Symbol	Parameter		Power SO20	Multiwatt15	Unit
$R_{th\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

2/13

PIN FUNCTIONS (refer to the block diagram)

MW.15	Power SO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	V_{SS}	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible inputs of the Bridge B.
13; 14	14;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
--	3;16	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V; T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; L = 0 V; = L V; = H		13 50	22 70	mA mA
		V_{en} = L V; = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; L = 0 V; = L V; = H		24 7	36 12	mA mA
		V_{en} = L V; = X			6	mA
V_{IL}	Input Low Voltage (pins 5, 7, 10, 12)		-0.3		1.5	V
V_{IH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		V_{SS}	V
I_{IL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			-10	µA
I_{IH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} -0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		-0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			-10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} -0.6V		30	100	µA
$V_{CE(sat)H}$	Source Saturation Voltage	L = 1A L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CE(sat)L}$	Sink Saturation Voltage	L = 1A (3) L = 2A (3)	0.85	1.2 1.7	1.6 2.3	V V
$V_{CE(sat)}$	Total Drop	L = 1A (3) L = 2A (3)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		-1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V)	Source Current Turn-off Delay	0.5 V; to 0.9 I_L (2); (4)		1.5		µs
T_2 (V)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V)	Source Current Turn-on Delay	0.5 V; to 0.1 I_L (2); (4)		2		µs
T_4 (V)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V)	Sink Current Turn-off Delay	0.5 V; to 0.9 I_L (3); (4)		0.7		µs
T_6 (V)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V)	Sink Current Turn-on Delay	0.5 V; to 0.9 I_L (3); (4)		1.6		µs
T_8 (V)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V)	Commutation Frequency	L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) Sustaining voltage can be -1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ -0.6 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

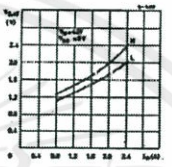
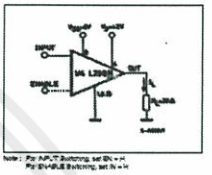


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

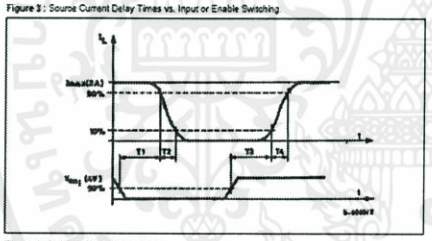
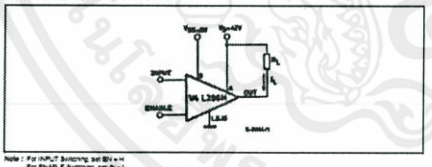


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input or Enable Switching.

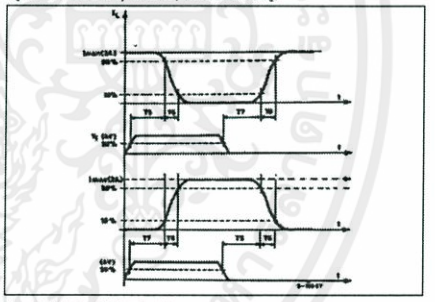
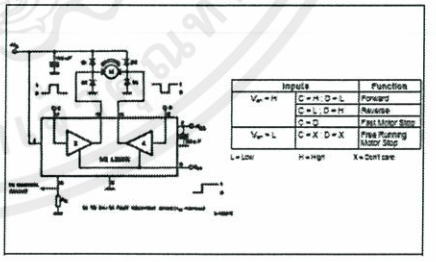


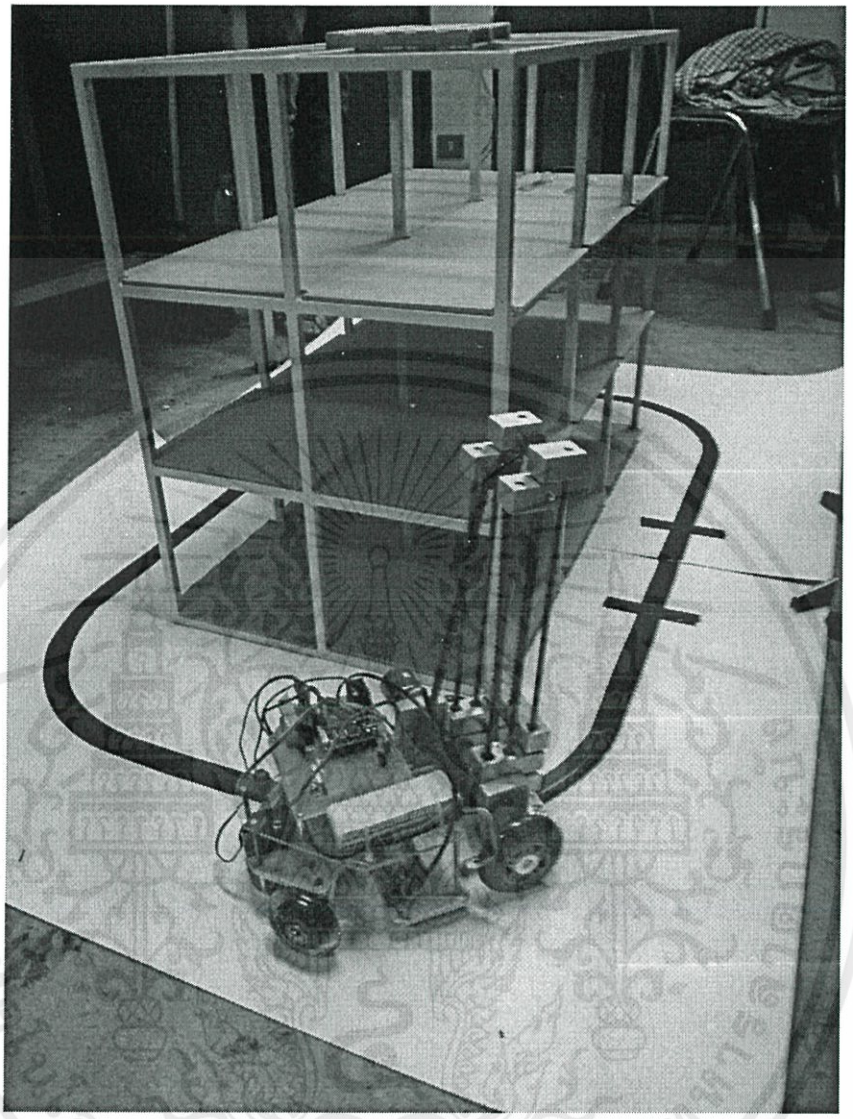
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

ภาคผนวก ค.



รูปสนามและชั้นวางของ

ภาคผนวก ง.

โปรแกรมทดสอบการรับส่งข้อมูลผ่าน Uart

```
#include<avr/io.h>
#include<util/delay.h>
void USART_Init( unsigned int baud )
{
    UBRRH = (unsigned char)(baud>>8);
    UBRRL = (unsigned char)baud;
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}

void USART_Transmit( unsigned char  data )
{
    /*  Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) );
    /*  Put data into buffer, sends the data  */
    UDR = data;
}

unsigned char  USART_Receive(  void )
{
    /*  Wait for data to be received  */
    while ( !(UCSRA & (1<<RXC)) );
    /*  Get and return received data from buffer */
    return  UDR;
}


void main()
{
    DDRD = 0x02;
    USART_Init(0x67);

    unsigned char data;

    while(1)
    {
        data = USART_Receive();
        _delay_ms(1);
        USART_Transmit(data);
    }

}
```

โปรแกรมบังคับด้วยมือ (Manual control)

```
#include<avr/io.h>
#include<util/delay.h>
#include<compat/deprecated.h>
#include<avr/interrupt.h>

#define TOP 2000
#define ADC_VREF_TYPE 0x60


volatile unsigned int tcnt=0;
volatile unsigned int ocr5=0;
volatile unsigned int ocr6=0;


// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}


void main()
{

    DDRA = 0b00000000;
    DDRB = 0b00000000;
    DDRC = 0b11111111;
    DDRD = 0b11111100;


    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: 2000.000 kHz
    // Mode: Fast PWM top=FFh
    // OC0 output: Non-Inverted PWM
    TCCR0=0x6A;
    TCNT0=0x00;
    OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA2;
TCCR1B=0x1B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=100;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

TIMSK=0x01;

sei();


// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;


   while(1)
   {
     //************** adc----->servo ***************
       ocr5 = (read_adc(5)>>1)+90;
       ocr6 = (read_adc(6)>>1)+90;
       _delay_ms(10);

     //************** button--->motorA *************

       //if button B3 is set motorA go forward
       if(bit_is_set(PINB,3) && bit_is_clear(PINB,4))
       {
           OCR1AL = 100;
           sbi(PORTC,0);
           cbi(PORTC,1);
       }

       //if button B4 is set motorA go backward
       else if(bit_is_clear(PINB,3) && bit_is_set(PINB,4))
       {
           OCR1AL = 100;
           cbi(PORTC,0);
           sbi(PORTC,1);
       }
```

```
            //if button B3&B4 is set motorA break
            else if(bit_is_set(PINB,3) && bit_is_set(PINB,4))
            {
                OCR1AL = 0;
                sbi(PORTC,0);
                sbi(PORTC,1);
            }

            else if(bit_is_clear(PINB,3) && bit_is_clear(PINB,4))
            {
                OCR1AL = 0;
                cbi(PORTC,0);
                cbi(PORTC,1);
            }

           //************** button--->motorB ************

            //if button B5 is set motorB go forward
            if(bit_is_set(PINB,5) && bit_is_clear(PINB,6))
            {
                OCR1BL = 100;
                sbi(PORTC,2);
                cbi(PORTC,3);
            }


            //if button B6 is set motorB go backward
            else if(bit_is_clear(PINB,5) && bit_is_set(PINB,6))
            {
                 OCR1BL = 100;
                 cbi(PORTC,2);
                 sbi(PORTC,3);
            }

            //if button B5&B6 is set motorB break
            else if(bit_is_set(PINB,5) && bit_is_set(PINB,6))
            {
                OCR1BL = 0;
                sbi(PORTC,2);
                sbi(PORTC,3);
            }

            //settling motorA,B
            else if(bit_is_clear(PINB,5) && bit_is_clear(PINB,6))
            {
                OCR1BL = 0;
                cbi(PORTC,2);
                cbi(PORTC,3);
            }


        }
    }

//************************ interrupt_servo1,2 ******************
ISR (TIMER0_OVF_vect)
{
    TCNT0 = 239;
    //if tcnt = TOP ,set C4 and C5 to 1,set tcnt = 0
    if(++tcnt > TOP)
    {
        tcnt = 0;
        sbi(PORTC,4);
        sbi(PORTC,5);

    }
    //if tcnt eqal ocr5 clear bit to 0
    if(tcnt >= ocr5)
    {
        cbi(PORTC,5);
    }

    //if tcnt eqal ocr6 clear bit to 0
    if(tcnt >= ocr6)
    {
        cbi(PORTC,4);
    }


}
```

โปรแกรมเดินตามเส้น (Tracking)

```
#include<avr/io.h>
#include<util/delay.h>
#include<compat/deprecated.h>
#include<avr/interrupt.h>

#define TOP 2000
#define ADC_VREF_TYPE 0x60

volatile unsigned int tcnt=0,x=0,t1=0,t[2],A[2],L,R;
volatile unsigned char a0=0,a1=0,a2=0;
volatile unsigned int ocr5=0,ocr4=0;

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

void Start_A()
{
    OCR1AL=80;
    cbi(PORTC,0);
    sbi(PORTC,1);
}

void Start_B()
{
    OCR1BL=100;
    cbi(PORTC,2);
    sbi(PORTC,3);
}


void Break_A()
{
    OCR1AL=0;
    sbi(PORTC,0);
    sbi(PORTC,1);
}

void read()
{
    a0=(read_adc(0)>>1)+90;
    a1=(read_adc(1)>>1)+90;
    a2=(read_adc(2)>>1)+90;
    if(a0<120)
        A[0]=0;
    else if(a0>=120)
        A[0]=1;
    if(a1<120)
        A[1]=0;
    else if(a1>=120)
        A[1]=1;
    if(a2<120)
        A[2]=0;
    else if(a2>=120)
        A[2]=1;
    _delay_ms(10);
}
```

```
void Show_counting()
{
    if(x==1)            //001
    {
        cbi(PORTD,3);
        cbi(PORTD,6);
        sbi(PORTD,7);
    }
    else if(x==2)       //010
    {
        cbi(PORTD,3);
        sbi(PORTD,6);
        cbi(PORTD,7);
    }
    else if(x==3)       //011
    {
        cbi(PORTD,3);
        sbi(PORTD,6);
        sbi(PORTD,7);
    }
    else if(x==4)       //100
    {
        sbi(PORTD,3);
        cbi(PORTD,6);
        cbi(PORTD,7);
    }

    else if(x==5)       //101
    {
        sbi(PORTD,3);
        cbi(PORTD,6);
        sbi(PORTD,7);
    }
    else if(x==6)       //110
    {
        sbi(PORTD,3);
        sbi(PORTD,6);
        cbi(PORTD,7);
    }
    else                //000
    {
        cbi(PORTD,3);
        cbi(PORTD,6);
        cbi(PORTD,7);
    }

    _delay_ms(10);
}
  ..   ...
void main()
{

    DDRA = 0b00000000;
    DDRB = 0b00000000;
    DDRC = 0b11111111;
    DDRD = 0b11111100;


    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: 2000.000 kHz
    // Mode: Fast PWM top=FFh
    // OC0 output: Non-Inverted PWM
    TCCR0=0x6A;
    TCNT0=0x00;
    OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA2;
TCCR1B=0x1B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=100;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

TIMSK=0x01;

sei();

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

ocr4=150;
ocr5=150;

read();

while(1)
{

   Start_A();



  //make sure middle_sensor(a1) run on the line
  while(A[1]==1) //detected black line
  {
      R=A[0];
      t[1]=A[1];
      L=A[2];

      read();

      //Check Left_sensor(a2) and Right_sensor(a0) detected white line
      //      * |*| *
      if(A[2]==0 && A[0]==0)
      {
          Start_A();
         ocr4=150;
      }
```

```
                //Check Left_sensor(a2) detected black line
                //           |**| *
                else if(A[2]==1 && A[0]==0)
                {
                    Start_A();
                    ocr4=110; //TurnRight
                }
                //Check Right_sensor(a0) detected black line
                //           * |**|
                else if(A[2]==0 && A[0]==1 )
                {
                    Start_A();
                    ocr4=190; //TurnLeft

                }
                //           |* * *|
                else if(A[2]==1 && A[0]==1) //Counter
                {


                if(bit_is_clear(PINA,4)) //Counter Position
                {
                    x++;

                    if(x>6)
                        x=0;

                }

                    _delay_ms(50);

                Show_counting();
            }

        }
    }


ISR (TIMER0_OVF_vect)
{
    TCNT0 = 239;
    //if tcnt = TOP ,set C4 and C5 to 1,set tcnt = 0
    if(++tcnt > TOP)
    {
        tcnt = 0;
        sbi(PORTC,4);
        sbi(PORTC,5);

    }
    //if tcnt eqal ocr5 clear bit to 0
    if(tcnt >= ocr5)
    {
        cbi(PORTC,5);
    }

    //if tcnt eqal ocr4 clear bit to 0
    if(tcnt >= ocr4)
    {
        cbi(PORTC,4);
    }


}
```

โปรแกรมรับข้อมูลผ่าน Uart และการใช้ข้อมูล

```
#include<avr/io.h>
#include<util/delay.h>
#include<compat/deprecated.h>
#include<avr/interrupt.h>

#define TOP 2000
#define ADC_VREF_TYPE 0x60

volatile unsigned int j=0,i,f,h=0,y=0,E=0,E1=0;
volatile unsigned int tcnt=0,ocr4=150,ocr5=0;
volatile unsigned char x,e=0;
volatile char ask[]="Position",go[]="Go",up[]="Up";

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}


void USART_Init( unsigned int baud )
{
    UBRRH = (unsigned char)(baud>>8);
    UBRRL = (unsigned char)baud;
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}

void USART_Transmit( unsigned char  data )
{
    //  Wait for empty transmit buffer
    while ( !( UCSRA & (1<<UDRE)) );
    //  Put data into buffer, sends the data
    UDR = data;
}

unsigned char  USART_Receive(  void )
{
    //  Wait for data to be received
    while ( !(UCSRA & (1<<RXC)) );
    //  Get and return received data from buffer
    return  UDR;
}
```

```
void Read_Encoder()
{
    e=(read_adc(7)>>1)+90;
    if(e<120)
        E=0;
    else if(e>=120)
        E=1;
}

void Start_A()
{
    OCR1AL=80;
    cbi(PORTC,0);
    sbi(PORTC,1);
}

void Backward_A()
{
    OCR1AL=80;
    sbi(PORTC,0);
    cbi(PORTC,1);
}

void Start_B()
{
    OCR1BL=80;
    cbi(PORTC,2);
    sbi(PORTC,3);
}


void Backward_B()
{
    OCR1BL=80;
    sbi(PORTC,2);
    cbi(PORTC,3);
}

void Break_A()
{
    OCR1AL=0;
    sbi(PORTC,0);
    sbi(PORTC,1);
}

void Break_B()
{
    OCR1BL=0;
    sbi(PORTC,2);
    sbi(PORTC,3);
}

void Limit_sw()
{
    while(bit_is_set(PINA,3))
    {

    }
    y=0;

    Break_B();
}


void main()
{
    unsigned char data[4];

    DDRA = 0b00000000;
    DDRC = 0b11111111;
    DDRD = 0b11111110;
    cbi(DDRB,1);

    USART_Init(0x67);

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: 2000.000 kHz
    // Mode: Fast PWM top=FFh
    // OC0 output: Non-Inverted PWM
    TCCR0=0x6A;
    TCNT0=0x00;
    OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA2;
TCCR1B=0x1B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=100;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

_delay_ms(20);

TIMSK=0x01;

cli(); //   sei();


// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;


USART_Transmit('P');            //print"Position"
USART_Transmit('o');
USART_Transmit('s');
USART_Transmit('i');
USART_Transmit('t');
USART_Transmit('i');
USART_Transmit('o');
USART_Transmit('n');

data[0] = USART_Receive();      //receive Rack_roll
_delay_ms(1);
data[1] = USART_Receive();      //receive Rack_floor
_delay_ms(1);


USART_Transmit('G');            //print"Go"
USART_Transmit('O');

switch(data[0])                 //choose the roll
{
    case 'A' :  ocr5=125;       //data[2]='1';
                break;
    case 'B' :  ocr5=130;       //data[2]='2';
                break;
    case 'C' :  ocr5=135;       //data[2]='3';
                break;
}
USART_Transmit(data[2]);

USART_Transmit('U');            //print"Up"
USART_Transmit('P');

switch(data[1])                 //choose the floor
{
    case '1' :  h=10;           //1st floor
                break;
    case '2' :  h=90;           //2nd floor
                break;
    case '3' :  h=185;          //3rd floor
                break;
    default: h=0;
}
USART_Transmit(data[1]);
USART_Transmit('\n');           //
sei();
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA2;
TCCR1B=0x1B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=100;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

_delay_ms(20);

TIMSK=0x01;

cli(); //   sei();


// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;


USART_Transmit('P');              //print"Position"
USART_Transmit('o');
USART_Transmit('s');
USART_Transmit('i');
USART_Transmit('t');
USART_Transmit('i');
USART_Transmit('o');
USART_Transmit('n');

data[0] = USART_Receive();        //receive Rack_roll
_delay_ms(1);
data[1] = USART_Receive();        //receive Rack_floor
_delay_ms(1);


USART_Transmit('G');              //print"Go"
USART_Transmit('O');

switch(data[0])                   //choose the roll
{
    case 'A' :  ocr5=125;         //data[2]='1';
                break;
    case 'B' :  ocr5=130;         //data[2]='2';
                break;
    case 'C' :  ocr5=135;         //data[2]='3';
                break;
}
USART_Transmit(data[2]);

USART_Transmit('U');              //print"Up"
USART_Transmit('P');

switch(data[1])                   //choose the floor
{
    case '1' :  h=10;             //1st floor
                break;
    case '2' :  h=90;             //2nd floor
                break;
    case '3' :  h=185;            //3rd floor
                break;
    default: h=0;
}
USART_Transmit(data[1]);
USART_Transmit('\n');             //
sei();
```

```
        Read_Encoder();
        Start_B();
        sbi(PORTD,3);
        for(y=0;y<h;y++)                    //Loop count Encoder
        {
            do
            {
                E1=E;
                Read_Encoder();
            }
            while(E1==E);
        }
        Break_B();
        cbi(PORTD,3);
        sbi(PORTD,6);
        do
        {
            Start_A();
            j++;
        }while(bit_is_set(PINA,4));
        cbi(PORTD,6);

            Break_A();
                _delay_ms(2000*8*2);

        sbi(PORTD,6);

            Backward_A();
                _delay_ms(40000);

        cbi(PORTD,6);

             Break_A();

        sbi(PORTD,7);
        Read_Encoder();
        Backward_B();

        Limit_sw();                         //check when lifting lowest
        cbi(PORTD,7);

}

ISR (TIMER0_OVF_vect)
{
    TCNT0 = 239;
    //if tcnt = TOP ,set C4 and C5 to 1,set tcnt = 0
    if(++tcnt > TOP)
    {
        tcnt = 0;
        sbi(PORTC,4);
        sbi(PORTC,5);

    }
    //if tcnt eqal ocr5 clear bit to 0
    if(tcnt >= ocr5)
    {
        cbi(PORTC,5);
    }

    //if tcnt eqal ocr4 clear bit to 0
    if(tcnt >= ocr4)
    {
        cbi(PORTC,4);
    }


}
```

โปรแกรมเมื่อรถเลี้ยวเข้ามาเก็บของ

```
#include<avr/io.h>
#include<util/delay.h>
#include<compat/deprecated.h>
#include<avr/interrupt.h>

#define TOP 2000
#define ADC_VREF_TYPE 0x60

volatile unsigned int tcnt=0,x=0,t1=0,t[2],A[2];
volatile unsigned char a0=0,a1=0,a2=0;
volatile unsigned int ocr5=0,ocr4=0;

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

void Start_A()
{
    OCR1AL=80;
    cbi(PORTC,0);
    sbi(PORTC,1);
}

void Start_B()
{
    OCR1BL=100;
    cbi(PORTC,3);
    sbi(PORTC,4);
}


void Break_A()
{
    OCR1AL=0;
    sbi(PORTC,0);
    sbi(PORTC,1);

}

void read()
{
    a0=(read_adc(0)>>1)+90;
    a1=(read_adc(1)>>1)+90;
    a2=(read_adc(2)>>1)+90;
    if(a0<120)
        A[0]=0;
    else if(a0>=120)
        A[0]=1;
    if(a1<120)
        A[1]=0;
    else if(a1>=120)
        A[1]=1;
    if(a2<120)
        A[2]=0;
    else if(a2>=120)
        A[2]=1;
//  _delay_ms(10);
}
```

```
void main()
{

    DDRA = 0b00000000;
    DDRB = 0b00000000;
    DDRC = 0b11111111;
    DDRD = 0b11111100;


    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: 2000.000 kHz
    // Mode: Fast PWM top=FFh
    // OC0 output: Non-Inverted PWM
    TCCR0=0x6A;
    TCNT0=0x00;
    OCR0=0x00;

    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: 250.000 kHz
    // Mode: Fast PWM top=ICR1
    // OC1A output: Non-Inv.
    // OC1B output: Non-Inv.
    // Noise Canceler: Off
    // Input Capture on Falling Edge
    // Timer1 Overflow Interrupt: Off
    // Input Capture Interrupt: Off
    // Compare A Match Interrupt: Off
    // Compare B Match Interrupt: Off
    TCCR1A=0xA2;
    TCCR1B=0x1B;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=100;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;

    TIMSK=0x01;

    sei();

    // ADC initialization
    // ADC Clock frequency: 1000.000 kHz
    // ADC Voltage Reference: AVCC pin
    // Only the 8 most significant bits of
    // the AD conversion result are used
    ADMUX=ADC_VREF_TYPE & 0xff;
    ADCSRA=0x84;

    ocr4=150;
    ocr5=150;

    read();

    while(1)
    {

        Start_A();
```

```
        //make sure middle_sensor(a1) run on the line
        while(A[1]==1) //detected black line
        {
            R=A[0];
            t[1]=A[1];
            L=A[2];

            read();

            //Check Left_sensor(a2) and Right_sensor(a0) detected white line
            //       * |*| *
            if(A[2]==0 && A[0]==0)
            {
                Start_A();
                ocr4=150;
            }



            //Check Left_sensor(a2) detected black line
            //             |**| *
            else if(A[2]==1 && A[0]==0)
            {
                Start_A();
                ocr4=110; //TurnRight
            }
            //Check Right_sensor(a0) detected black line
            //           * |**|
            else if(A[2]==0 && A[0]==1 )
            {
                Start_A();
                ocr4=190; //TurnLeft

            }
            //             |* * *|
            else if(A[2]==1 && A[0]==1) //Counter
            {
            //  detected magnetic sensor
            if(bit_is_clear(PINA,4))
            {
                while(x<=147)       //turn Right to the rack in x time
                {                   //we can adjust angle in x value
                    Start_A();
                    ocr4=195;
                    _delay_ms(x++);
                }
                x=0;

                do
                {
                    read();
                    ocr4=150;

                }while(!(A[2]==1 && A[1]==1 && A[0]==1));


                while(1)
                {
                    Break_A();
                }

            }
                _delay_ms(50);

        }

    }
```

```
ISR (TIMER0_OVF_vect)
{
    TCNT0 = 239;
    //if tcnt = TOP ,set C4 and C5 to 1,set tcnt = 0
    if(++tcnt > TOP)
    {
        tcnt = 0;
        sbi(PORTC,4);
        sbi(PORTC,5);

    }
    //if tcnt eqal ocr5 clear bit to 0
    if(tcnt >= ocr5)
    {
        cbi(PORTC,5);
    }

    //if tcnt eqal ocr4 clear bit to 0
    if(tcnt >= ocr4)
    {
        cbi(PORTC,4);
    }


}
```

การทำงานโดยรวม

```
//***********************************< Heading >*********************************
#include<avr/io.h>
#include<util/delay.h>
#include<compat/deprecated.h>
#include<avr/interrupt.h>

//*****************************< Define variable >*****************************

#define TOP 2000
#define ADC_VREF_TYPE 0x60

volatile unsigned int ocr5,ocr4,tcnt=0,x=0,A[2],z,h,e,E,E1,Point,Station,Count=0,T;
volatile unsigned char a0,a1,a2,X,Y,Z;


//*************************< Functions and Sub Programs >*********************

//                              *************[ADC]************
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
//                             ********[Reading sensor]******
void read()
{
    a0=(read_adc(0)>>1)+90;
    a1=(read_adc(1)>>1)+90;
    a2=(read_adc(2)>>1)+90;
    if(a0<120)
        A[0]=0;
    else if(a0>=120)
        A[0]=1;
    if(a1<120)                                      //<<<<<<<< [Optical sensor]
        A[1]=0;
    else if(a1>=120)
        A[1]=1;
    if(a2<120)
        A[2]=0;
    else if(a2>=120)
        A[2]=1;
    _delay_ms(10);
}

void Read_Encoder()
{
    e=(read_adc(7)>>1)+90;
    if(e<120)                                       //<<<<<<<< [Encoder sensor]
        E=0;
    else if(e>=120)
        E=1;
}

//                             ********[Functions Uart]*******
void USART_Init( unsigned int baud )
{
    UBRRH = (unsigned char)(baud>>8);
    UBRRL = (unsigned char)baud;
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}

void USART_Transmit( unsigned char  data )
{
    //  Wait for empty transmit buffer
    while ( !( UCSRA & (1<<UDRE)) );
    //  Put data into buffer, sends the data
    UDR = data;
}

unsigned char  USART_Receive(  void )
{
    //  Wait for data to be received
    while ( !(UCSRA & (1<<RXC)) );
    //  Get and return received data from buffer
    return  UDR;
}
```

```
//                          ********[Print "Position"]******
void Print(unsigned int T)
{
    if(T==0)
    {
        USART_Transmit('P');        //print "Position"
        USART_Transmit('o');
        USART_Transmit('s');
        USART_Transmit('i');
        USART_Transmit('t');
        USART_Transmit('i');
        USART_Transmit('o');
        USART_Transmit('n');
    }

    if(T==1)
    {   USART_Transmit('\t');       //print "Wrong Press again"
        USART_Transmit('W');
        USART_Transmit('r');
        USART_Transmit('o');
        USART_Transmit('n');
        USART_Transmit('g');
        USART_Transmit(' ');
        USART_Transmit('P');
        USART_Transmit('r');
        USART_Transmit('e');
        USART_Transmit('s');
        USART_Transmit('s');
        USART_Transmit(' ');
        USART_Transmit('a');
        USART_Transmit('g');
        USART_Transmit('a');
        USART_Transmit('i');
        USART_Transmit('n');
        USART_Transmit('\n');
    }

}

//                          ********[Receive Uart]*********
void Receive_XYZ()
{
    X = USART_Receive();        //receive X (1,2,3)          Z
    _delay_ms(1);               //                           |
    Y = USART_Receive();        //receive Y (1,2)           / \
    _delay_ms(1);               //                        X   Y
    Z = USART_Receive();        //receive Z (1,2,3)
    _delay_ms(1);
    USART_Transmit('G');
    USART_Transmit('o');
}
//                          ********[Control DC motor]******
void Start_A()
{
    OCR1AL=70;
    cbi(PORTC,0);
    sbi(PORTC,1);
}

void Backward_A()
{
    OCR1AL=70;
    sbi(PORTC,0);
    cbi(PORTC,1);
}

void Break_A()
{
    OCR1AL=0;
    sbi(PORTC,0);
    sbi(PORTC,1);
}

void Start_B()
{
    OCR1BL=100;
    cbi(PORTC,2);
    sbi(PORTC,3);
}

void Backward_B()
{
    OCR1BL=100;
    sbi(PORTC,2);
    cbi(PORTC,3);
}

void Break_B()
{
    OCR1BL=0;
    sbi(PORTC,2);
    sbi(PORTC,3);
}
```

```
//                          ********[Choose Point]*********
void Select_P()
{
    T=0;
    if(X=='1' && Y=='2' && Z=='1')
    {   Point=1;
        h=0;    }
    else if(X=='2' && Y=='2' && Z=='1')          //  _________ Y
    {   Point=2;                                 //  | 6 | 1 |
        h=0;    }                                //  |___|___|
    else if(X=='3' && Y=='2' && Z=='1')          //  | 5 | 2 |       Z=1 , h=10 times (Encoder)
    {   Point=3;                                 //  |___|___|
        h=0;    }                                //  | 4 | 3 |
    else if(X=='3' && Y=='1' && Z=='1')          //  |___|___|
    {   Point=4;                                 //  |
        h=0;    }                                //  X
    else if(X=='2' && Y=='1' && Z=='1')
    {   Point=5;
        h=0;    }
    else if(X=='1' && Y=='1' && Z=='1')
    {   Point=6;
        h=0;    }

    else if(X=='1' && Y=='2' && Z=='2')
    {   Point=1;
        h=92;   }
    else if(X=='2' && Y=='2' && Z=='2')          //  _________ Y
    {   Point=2;                                 //  | 6 | 1 |
        h=92;   }                                //  |___|___|
    else if(X=='3' && Y=='2' && Z=='2')          //  | 5 | 2 |       Z=2 , h=90 times (Encoder)
    {   Point=3;                                 //  |___|___|
        h=92;   }                                //  | 4 | 3 |
    else if(X=='3' && Y=='1' && Z=='2')          //  |___|___|
    {   Point=4;                                 //  |
        h=92;   }                                //  X
    else if(X=='2' && Y=='1' && Z=='2')
    {   Point=5;
        h=92;   }
    else if(X=='1' && Y=='1' && Z=='2')
    {   Point=6;
        h=92;   }

    else if(X=='1' && Y=='2' && Z=='3')
    {   Point=1;
        h=185;  }
    else if(X=='2' && Y=='2' && Z=='3')          //  _________ Y
    {   Point=2;                                 //  | 6 | 1 |
        h=185;  }                                //  |___|___|
    else if(X=='3' && Y=='2' && Z=='3')          //  | 5 | 2 |       Z=3 , h=185 times (Encoder)
    {   Point=3;                                 //  |___|___|
        h=185;  }                                //  | 4 | 3 |
    else if(X=='3' && Y=='1' && Z=='3')          //  |___|___|
    {   Point=4;                                 //  |
        h=185;  }                                //  X
    else if(X=='2' && Y=='1' && Z=='3')
    {   Point=5;
        h=185;  }
    else if(X=='1' && Y=='1' && Z=='3')
    {   Point=6;
        h=185;  }

    else
    {
       T=1;
       Print(1);

    }
    Station=7;
    Station=Station<<1;                  //multiply 2
    Point=Point<<1;
    Point++;
}

void Turn(unsigned char D)
{
    if(D=='L')
    ocr4=105;              //turn Left to pallet in x time
    else if(D=='R')
    ocr4=195;              //turn Right to pallet in x time
    while(x<=147)
    {                      //we can adjust angle in x value
        Start_A();

        _delay_ms(x++);
    }
    x=0;
}
```

```
void Raise_up(unsigned int H)
{
    Read_Encoder();
    Start_B();
    for(z=0;z<H;z++)                //Loop count Encoder
    {
        do
        {
            E1=E;
            Read_Encoder();
        }
        while(E1==E);
    }
    Break_B();
}

void Raise_down(unsigned int H)
{
    Read_Encoder();
    Backward_B();
    for(z=0;z<H;z++)                //Loop count Encoder
    {
        do
        {
            E1=E;
            Read_Encoder();
        }
        while(E1==E);
    }
    Break_B();
}

//                          ********[Limit Switch]********
void Limit_sw()
{
    while(bit_is_set(PINA,3))
    {
    }
    z=0;

    Break_B();
}

//                          *******[Remote Control]*******
void Manual_Control()
{
    while(1)
    {

        //************* adc------>servo **************
         ocr5 = (read_adc(5)>>1)+90;
         ocr4 = (read_adc(6)>>1)+90;
         _delay_ms(10);

        //************* button------>motorA ************

         //if button B3 is set motorA go forward
         if(bit_is_set(PINB,3) && bit_is_clear(PINB,4))
         {
             OCR1AL = 100;
             sbi(PORTC,0);
             cbi(PORTC,1);
         }

         //if button B4 is set motorA go backward
         else if(bit_is_clear(PINB,3) && bit_is_set(PINB,4))
         {
             OCR1AL = 100;
             cbi(PORTC,0);
             sbi(PORTC,1);
         }

         //if button B3&B4 is set motorA break
         else if(bit_is_set(PINB,3) && bit_is_set(PINB,4))
         {
             OCR1AL = 0;
             sbi(PORTC,0);
             sbi(PORTC,1);
         }

         else if(bit_is_clear(PINB,3) && bit_is_clear(PINB,4))
         {
             OCR1AL = 0;
             cbi(PORTC,0);
             cbi(PORTC,1);
         }

        //************* button---->motorB ************

         //if button B5 is set motorB go forward
         if(bit_is_set(PINB,5) && bit_is_clear(PINB,6))
         {
             OCR1BL = 100;
             sbi(PORTC,2);
             cbi(PORTC,3);
         }

         //if button B6 is set motorB go backward
         else if(bit_is_clear(PINB,5) && bit_is_set(PINB,6))
         {
             OCR1BL = 100;
             cbi(PORTC,2);
             sbi(PORTC,3);
         }
```

```
            //************* button------>motorB ************

             //if button B5 is set motorB go forvard
             if(bit_is_set(PINB,5) && bit_is_clear(PINB,6))
             {
                 OCR1BL = 100;
                 sbi(PORTC,2);
                 cbi(PORTC,3);
             }

             //if button B6 is set motorB go backward
             else if(bit_is_clear(PINB,5) && bit_is_set(PINB,6))
             {
                 OCR1BL = 100;
                 cbi(PORTC,2);
                 sbi(PORTC,3);
             }

             //if button B5&B6 is set motorB break
             else if(bit_is_set(PINB,5) && bit_is_set(PINB,6))
             {
                 OCR1BL = 0;
                 sbi(PORTC,2);
                 sbi(PORTC,3);
             }


             else if(bit_is_clear(PINB,5) && bit_is_clear(PINB,6))
             {
                 OCR1BL = 0;
                 cbi(PORTC,2);
                 cbi(PORTC,3);
             }
            //*****************************************

    }
}

//*******************************< Main Program >*********************************
void main()
{
//                          ********[Setting registers]*********
    DDRA = 0b00000000;
    DDRB = 0b00000000;                      //<<<<<<<<[set PORT]
    DDRC = 0b11111111;                      //Input=0, Output=1
    DDRD = 0b11111100;
    USART_Init(0x67);                       //Set Uart

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: 2000.000 kHz
    // Mode: Fast PWM top=FFh
    // OC0 output: Non-Inverted PWM
    TCCR0=0x6A;
    TCNT0=0x00;
    OCR0=0x00;

    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: 250.000 kHz
    // Mode: Fast PWM top=ICR1
    // OC1A output: Non-Inv.
    // OC1B output: Non-Inv.
    // Noise Canceler: Off
    // Input Capture on Falling Edge
    // Timer1 Overflow Interrupt: Off
    // Input Capture Interrupt: Off
    // Compare A Match Interrupt: Off
    // Compare B Match Interrupt: Off
    TCCR1A=0xA2;
    TCCR1B=0x1B;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=100;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;

    TIMSK=0x01;

    cli();              //<---------[Disable Interrupt Servo Function]

    // ADC initialization
    // ADC Clock frequency: 1000.000 kHz
    // ADC Voltage Reference: AVCC pin
    // Only the 8 most significant bits of
    // the AD conversion result are used
    ADMUX=ADC_VREF_TYPE & 0xff;
    ADCSRA=0x84;


    ocr4=150;           //<---------[set servo begining value]
    ocr5=130;

//              <<<<<<<<<<<<<<********[Start Program]********>>>>>>>>>>>>>>>>
    if(bit_is_set(PIND,2))      //Check Interrupt
        Manual_Control();

    do
    {
        Print(0);                   //Ask Position
        Receive_XYZ();              //Receive Position
        Select_P();                 //Select Point
    }while(T==1);                 //if wrong,it will go back to loop

    sei();              //<--------[Enable Interrupt Servo Function]

//  We know Point now. So, we can unplug USB
//  _delay_ms(150000);
    read();
    Backward_B();
    Limit_sw();
    Raise_up(2);
    Start_A();
```

```
    while(1)
    {

//                          *********[Hold a pallet]*********
        do
        {
            read();
        }while(bit_is_set(PINA,4));
    //  Raise_down(7);
        ocr4=150;
        ocr5=140;
        do
        {
            read();
            ocr4=150;
        }while(!(A[2]==1 && A[1]==1 && A[0]==1)); //out loop if |***|

        Break_A();

        ocr5=145;
        Raise_up(12);
        //---------------Go to pallet
        Backward_A();
        ocr4=150;                       //hold a pallet
        while(bit_is_set(PINA,4))
        {
        }
        ocr5=140;

//                          ********[Go to the line]********

           //make sure middle_sensor(a1) run on the line
           while(1) //detected black line
           {
               read();

               //Check Left_sensor(a2) and Right_sensor(a0) detected white line
               //          * |*| *
               if(A[2]==0 && A[0]==0)
               {
                   Start_A();
                   ocr4=150;
               }

               //Check Left_sensor(a2) detected black line
               //          |**| *
               else if(A[2]==1 && A[0]==0)
               {
                   Start_A();
                   ocr4=105; //TurnRight
               }
               //Check Right_sensor(a0) detected black line
               //          * |**|
               else if(A[2]==0 && A[0]==1 )
               {
                   Start_A();
                   ocr4=195; //TurnLeft

               }

               //
               if(bit_is_clear(PINA,4)) //Counter Position
               {
                   USART_Transmit('d');
                   while(!(bit_is_set(PINA,4)))
                   {
                       ocr4=150;
                   }

                   Count++;

                   if(Count==Point)
                   {
                       Turn('R');
                       ocr5=135;
                       do
                       {
                           read();
                           ocr4=150;
                       }while(!(A[2]==1 && A[1]==1 && A[0]==1)); //out loop if |***|
                       Break_A();

                       //-------------------------Raise a pallet
                       ocr5=140;

                       Raise_up(h);
```

```
                    Start_A();
                    OCR1AL=55;
                    read();
                    do
                    {
                        //          |***|
                        if(A[2]==1 && A[1]==1 && A[0]==1)
                            ocr4=150;

                        //          |**| *
                        else if(A[2]==1 && A[0]==0)
                            ocr4=120; //TurnRight

                        //          * |**|
                        else if(A[2]==0 && A[0]==1 )
                            ocr4=180; //TurnLeft
                        read();
                    }
                    while(!(A[2]==0 && A[1]==0 && A[0]==0)); //out loop if |   |
                    //_delay_ms(48000);

                    Break_A();
                    //-------------------------Put a pallet
                    ocr5=145;
                    Raise_down(10);
                    ocr5=155;
                    _delay_ms(5000);
                    _delay_ms(5000);
                    //-------------------
                    Backward_A();
                    OCR1AL=65;
                    ocr4=150;
                    do
                    {
                        read();
                        USART_Transmit('k');
                    }while(!(A[2]==1 && A[1]==1 && A[0]==1));

                    _delay_ms(4000);
                    _delay_ms(4000);
                    Break_A();

                    Backward_B();
                    Limit_sw();
                    //-----------------Go Back
                    Backward_A();
                    ocr4=150;

                    while(x<=160)       //Go back in x time
                    {                   //we can adjust angle in x value
                        Backward_A();
                        ocr4=195;
                        _delay_ms(x++);
                    }
                    x=0;
                    Break_A();
                }

                if(Count==Station)
                {

                    Break_A();
                    while(1);
                }

            }

            _delay_ms(50);

        } Break_A();

    }
}

//************************< Function interrupt servo with PWM >*************************
ISR (TIMER0_OVF_vect)
{
    TCNT0 = 239;
    //if tcnt = TOP ,set C4 and C5 to 1,set tcnt = 0
    if(++tcnt > TOP)
    {
        tcnt = 0;
        sbi(PORTC,4);
        sbi(PORTC,5);

    }
    //if tcnt eqal ocr5 clear bit to 0
    if(tcnt >= ocr5)
    {
        cbi(PORTC,5);
    }

    //if tcnt eqal ocr4 clear bit to 0
    if(tcnt >= ocr4)
    {
        cbi(PORTC,4);
    }
}
```