ต้นแบบไซโลและระบบการจ่ายวัตถุดิบ :ส่วนโปรแกรมและระบบลำเลียง

PROTOTYPE OF SILO AND RAW MATERIAL DISPENSING SYSTEM: SOFTWARE PART AND TRANSPORTATION SYSTEM

จีรศักดิ์ ตันมีสุข

อภิสัฏ เอื้อไพโรจน์สาวร

ฮาริส มายนุ้ย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ต้นแบบไซโลและระบบการจ่ายวัตถุดิบ: ส่วนโปรแกรมและระบบลำเลียง

PROTOTYPE OF SILO AND RAW MATERIAL DISPENSING SYSTEM: SOFTWARE PART AND TRANSPORTATION SYSTEM

จีรศักดิ์ ตันมีสุข
อภิณัฐ เอื้อไพโรจน์ถาวร
ฮาริส สายนุ้ย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

PROTOTYPE OF SILO AND RAW MATERIAL DISPENSING SYSTEM: SOFTWARE PART AND TRANSPORTATION SYSTEM

JEERASAK TUNMEESUK
APINAT EUAPAIRODTAVORN
HARIS SAINUI

THIS THESIS IS SUBMITTED IN PARTIAL FULLFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2012

ปริญญานิพนธ์ปีการศึกษา 2555

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ต้นแบบไซโลและระบบการจ่ายวัตถุดิบ: ส่วนโปรแกรมและระบบลำเลียง
PROTOTYPE OF SILO AND RAW MATERIAL DISPENSING SYSTEM:
SOFTWARE PART AND TRANSPORTATION SYSTEM

ผู้จัดทำ	นายจีรศักดิ์	ตันมีสุข	52010172
	นายอภิณัฐ	เอื้อไพโรจน์ถาวร	52011394
	นายฮาริส	สายนุ้ย	52011478

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. ถาวร เบญจนราสุทธิ์)

ต้นแบบไซโลและระบบการจ่ายวัตถุดิบ: ส่วนโปรแกรมและระบบลำเลียง

โดย

นายจีรศักดิ์	ตันมีสุข	52010172
นายอภิณัฐ	เอื้อไพโรจน์ถาวร	52011394
นายฮาริส	สายนุ้ย	52011478

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. ถาวร เบญจนราสุทธิ์

ปีการศึกษา 2555

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการออกแบบและพัฒนาโปรแกรมสำหรับระบบไซโลต้นแบบและระบบลำเลียงวัตถุดิบ การศึกษาประกอบด้วยส่วนสำคัญ 2 ส่วน ได้แก่ โปรแกรมสำหรับระบบไซโลต้นแบบ และระบบลำเลียงวัตถุดิบ

โปรแกรมสำหรับระบบไซโลต้นแบบพัฒนาโดยโปรแกรมไมโครซอฟท์วิชวลซีชาร์ปให้สามารถใช้ในการเชื่อมต่อกับผู้ใช้งานเพื่อรับค่าน้ำหนักและจำนวนชุดที่ต้องการของวัตถุดิบแต่ละชนิด ตลอดจนประมวลผลเพื่อสั่งการไปยังส่วนระบบลำเลียง นอกจากนั้นยังสามารถบันทึกข้อมูลการผลิตเพื่อเก็บเป็นฐานข้อมูลสำหรับการตรวจสอบในภายหลัง

ระบบลำเลียงวัตถุดิบประกอบด้วย รถลำเลียงวัตถุดิบแบบเดินตามเส้นและส่วนติดต่อสื่อสาร รถลำเลียงวัตถุดิบควบคุมโดยใช้ไมโครคอนโทรลเลอร์โดยอาศัยทฤษฎีการควบคุมแบบพีไอดี และส่วนติดต่อสื่อสารใช้เทคโนโลยีซิกบีเพื่อสื่อสารระหว่างรถลำเลียงกับระบบจ่ายวัตถุดิบ และระหว่างรถลำเลียงแต่ละคัน

จากการทดลองพบว่า โปรแกรมสำหรับระบบไซโลที่พัฒนาขึ้นสามารถรับคำสั่งค่าน้ำหนักและจำนวนชุดของวัตถุดิบได้ตามต้องการ และระบบลำเลียงวัตถุดิบสามารถลำเลียงวัตถุดิบไปส่งตามจุดหมายได้ โดยรถลำเลียงแต่ละคันสามารถวางแผนเส้นทางและหลบเลี่ยงการชนกันของรถลำเลียงแต่ละคันได้

PROTOTYPE OF SILO AND RAW MATERIAL DISPENSING: SOFTWARE PART AND TRANSPORTATION SYSTEM

By

Mr. Jeerasak	Tunmeesuk	52010172
Mr. Apinat	Euapairodtavorn	52011394
Mr. Haris	Sainui	52011478

Advisor

Assoc. Prof. Dr. Taworn Benjanarasuth

Academic Year 2012

ABSTRACT

This thesis presents the design and implementation of a program for the prototype of a silo system and a raw material transportation system. The study can be divided into 2 major parts which are the program part for the prototype of the *silo system and the raw material transportation part.*

The program for the prototype of the silo system is developed by Microsoft visual c#. The Graphical User Interfaces (GUIs) of the developed program allow users to specify the desired weights and batches of raw materials. Moreover the program computes and commands the transportation system automatically. In addition the program can record production data in a database for further inspection.

The transportation system composes of trucks for transporting raw materials and communication modules for communicating with a microcontroller. The trucks are controlled by the microcontroller with PID control theory. The communication modules employ Zigbee technology for communicating between the truck and raw material dispensing system and between each truck.

The experimental results show that the developed program for the silo system can receive the commands of the number of weight and the set number of raw material required. The transportation system can transfer raw material to the target correctly. And each truck can plan a route and crashing avoidance between each truck.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา รศ.ดร.ถาวร เบญจนราสุทธิ์ เป็นอย่างสูง ที่ได้กรุณาให้คำปรึกษาแนวทางในการแก้ปัญหาต่าง ๆ ในโครงงานทั้งทางทฤษฎีและทางปฏิบัติ ทำให้ผู้จัดทำเข้าใจถึงที่มาของปัญหา และสามารถแก้ไขปัญหาต่าง ๆ ได้อย่างถูกวิธี รวมถึงเอื้อเฟื้ออุปกรณ์ที่จำเป็น ความเอาใจใส่ดูแลสอบถามถึงความก้าวหน้าอย่างสม่ำเสมอ ทำให้ผู้จัดทำได้ทำงานอย่างมีประสิทธิผลยิ่งขึ้น

ขอขอบพระคุณ อาจารย์ทุกท่านที่ได้ประสิทธิ์ประสาทความรู้ทั้งทางทฤษฎีและทางปฏิบัติแก่คณะผู้จัดทำ ทำให้เข้าใจและสามารถนำความรู้มาใช้ในปริญญานิพนธ์ฉบับนี้

ขอขอบคุณภาควิชาวิศวกรรมระบบควบคุม ที่ได้เอื้อเฟื้ออุปกรณ์ สถานที่ในการวิจัย และเครื่องมือต่าง ๆ ตลอดจนอำนวยความสะดวกให้แก่คณะผู้จัดทำจนปริญญานิพนธ์ฉบับนี้เสร็จสิ้น

ขอขอบคุณเพื่อน ๆ ทุก ๆ คนที่กำลังใจ สนับสนุนอุปกรณ์ที่ขาดเหลือ กระตุ้นเตือน และคอยถามไถ่ความคืบหน้าของโครงงานตลอดเวลา

และท้ายสุดขอขอบพระคุณ คุณพ่อ คุณแม่ และครอบครัว ที่คอยให้การสนับสนุนและโอกาสแก่คณะผู้จัดทำได้เล่าเรียนจนถึงวันนี้ ตลอดจนเพื่อน ๆ และ พี่ ๆ ในห้องวิจัยทุกคน ที่ให้คำปรึกษาแนะนำ และเป็นกำลังใจจนปริญญานิพนธ์ฉบับนี้เสร็จสิ้นสมบูรณ์

นายจีรศักดิ์ ตันมีสุข
นายอภิณัฐ เอื้อไพโรจน์ถาวร
นายฮาริส สายนุ้ย

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 กล่าวนำ

ในยุคปัจจุบันวงการอุตสาหกรรมได้ขยายตัวกว่าแต่ก่อนมากเพื่อตอบสนองความต้องการของผู้บริโภคและมีการแข่งขันสูง โรงงานอุตสาหกรรมทั้งขนาดใหญ่และขนาดเล็กทุกแห่งต่างพยายามพัฒนาผลิตภัณฑ์ของตนเองให้ตรงตามความต้องการของตลาด เป็นไปตามมาตรฐานผลิตภัณฑ์สากลเพื่อให้อยู่รอดท่ามกลางการแข่งขัน นอกจากตัวผลิตภัณฑ์แล้วระบบการจัดการในโรงงานก็มีผลโดยตรงต่อธุรกิจ หากธุรกิจใดมีการจัดการที่ไร้ประสิทธิภาพย่อมส่งผลเสียต่อธุรกิจอย่างแน่นอน ซึ่งความก้าวหน้าทางเทคโนโลยีในปัจจุบันถูกนำมาประยุกต์ใช้เพื่อพัฒนาระบบโรงงานให้มีประสิทธิภาพ มีการจัดการที่รวดเร็วและถูกต้องแม่นยำ เป็นการช่วยลดต้นทุนการผลิต เพิ่มผลกำไร และสร้างความน่าเชื่อถือให้กับโรงงาน

โครงงานนี้เริ่มมาจากการได้มีโอกาสไปศึกษาดูงาน ณ โรงงานอาหารสัตว์แห่งหนึ่งในส่วนของกระบวนการผลิตโดยเฉพาะการเตรียมวัตถุดิบตั้งต้น กลุ่มผู้จัดทำได้เล็งเห็นถึงความไม่ปลอดภัยของพนักงานขณะทำงานที่ต้องรับฝุ่นละอองในปริมาณมาก นอกจากนั้นยังขาดการเก็บบันทึกข้อมูลการผลิตในแต่ละรอบตามความต้องการของผู้สั่งผลิตภัณฑ์ ดังนั้นกลุ่มผู้จัดทำจึงมีแนวคิดในการพัฒนากระบวนการผลิตให้มีประสิทธิภาพมากขึ้น โดยใช้ระบบไซโลในการเก็บวัตถุดิบตั้งต้นและระบบจ่ายวัตถุดิบอัตโนมัติ ช่วยให้สามารถจ่ายวัตถุดิบได้ตามที่ต้องการและบันทึกข้อมูลปริมาณวัตถุดิบที่ใช้ในแต่ละรอบการผลิต และสามารถเรียกดูข้อมูลย้อนหลังได้ ทำให้ง่ายต่อการตรวจสอบเพื่อช่วยให้โรงงานสามารถผลิตสินค้าที่มีคุณภาพและปริมาณได้ตามเป้าหมาย อีกทั้งเป็นการยกระดับของโรงงานอุตสาหกรรมให้มีมาตรฐานที่ดีขึ้นเป็นไปตามความต้องการของผู้สั่งผลิตภัณฑ์ แต่เนื่องจากปัญหาในเรื่องราคาของอุปกรณ์ที่จะนำมาใช้กับโรงงานนั้นมีราคาสูง ทำให้การพัฒนาระบบจริงในโรงงานไม่สามารถดำเนินการได้ จึงลดขอบเขตการศึกษาเป็นการจำลองแทน โดยการจัดทำโครงงานแบ่งเป็น 2 โครงงาน คือ โครงงานส่วนโครงสร้าง ประกอบด้วยไซโลต้นแบบและระบบการจ่ายวัตถุดิบ และโครงงานส่วนโปรแกรมและระบบลำเลียงวัตถุดิบ ซึ่งเป็นโครงงานที่ศึกษาในปริญญานิพนธ์ฉบับนี้

1.2 วัตถุประสงค์ของการจัดทำโครงงาน

1. พัฒนาโปรแกรมสำหรับระบบไซโลต้นแบบให้สามารถสั่งการทำงานตามน้ำหนักและจำนวนชุดที่ต้องการ และสามารถจัดเก็บข้อมูลการทำงานเพื่อการตรวจสอบในภายหลัง
2. พัฒนาระบบลำเลียงวัตถุดิบโดยใช้รถลำเลียงแบบเดินตามเส้น ซึ่งรถลำเลียงสามารถวางแผนการลำเลียงเองได้โดยอัตโนมัติ และหลีกเลี่ยงปัญหาการชนกันของรถลำเลียง

1.3 ขอบเขตของโครงงาน

1. พัฒนาโปรแกรมสำหรับระบบไซโลต้นแบบ โดยอาศัยโปรแกรมไมโครซอฟท์วิชวลซีชาร์ป (Microsoft visual c#) ให้สามารถรับคำสั่งจากผู้ใช้งาน รับส่งข้อมูลกับไมโครคอนโทรลเลอร์ และบอกสถานะการทำงานของไซโลและรถลำเลียงวัตถุดิบในแต่ละช่วงการทำงานได้

2. ศึกษาและทดลองการใช้งานอุปกรณ์ต่าง ๆ ที่จำเป็นในระบบควบคุมทั้งทางอิเล็กทรอนิกส์และทางกล

3. ศึกษาทฤษฎีระบบควบคุมในกลุ่มตัวควบคุมแบบพีไอดี ให้สามารถออกแบบตัวควบคุมเพื่อควบคุมรถลำเลียงวัตถุดิบให้มีประสิทธิภาพ และสามารถนำไปใช้ได้จริง

4. ออกแบบระบบลำเลียงวัตถุดิบโดยสร้างรถลำเลียงแบบเดินตามเส้นจำนวน 2 คัน

5. ทดลองสั่งการทำงานผ่านโปรแกรมที่พัฒนาขึ้น และบันทึกผลการทดลอง

1.4 เนื้อหาในปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้แบ่งออกเป็น 5 บท ซึ่งรายละเอียดโดยคร่าวของแต่ละบทสามารถอธิบายได้ดังนี้

บทที่ 1 บทนำ กล่าวนำถึงวัตถุประสงค์ ขอบเขตของโครงงาน พร้อมทั้งรายละเอียดของปริญญานิพนธ์ของแต่ละบทในปริญญานิพนธ์

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง กล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการจัดทำโครงงาน ไมโครซอฟท์วิชวลชาร์ป ไมโครคอนโทรลเลอร์ การสื่อสารแบบอนุกรม การเชื่อมต่อพอร์ตอนุกรมมาตรฐาน RS-232 ยูอาร์ท ซิกบี มอเตอร์ไฟฟ้ากระแสตรง ออพติคัลเซนเซอร์ และตัวควบคุมในกลุ่มตัวควบคุมแบบพีไอดี

บทที่ 3 หลักการออกแบบ นำเสนอการออกแบบรถลำเลียงวัตถุดิบ วงจรอิเล็กทรอนิกส์ที่เกี่ยวข้อง โปรแกรมที่พัฒนาขึ้น รวมถึงแนวคิดในการออกแบบตัวควบคุม

บทที่ 4 ผลการทดลอง เป็นส่วนการทดสอบองค์ประกอบต่าง ๆ ในโครงงาน

บทที่ 5 บทวิจารณ์และสรุป ซึ่งจะสรุปผลการดำเนินงานในโครงงานทั้งหมด ปัญหาที่พบและแนวทางการปรับปรุงแก้ไขรวมไปถึงการพัฒนาโครงงานนี้ต่อไป

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

บทนี้กล่าวถึงทฤษฎีและความรู้ที่เกี่ยวข้องซึ่งจำเป็นในโครงงาน โดยแบ่งเป็น 3 ส่วน คือ ส่วนโปรแกรมการเชื่อมต่อกับผู้ใช้งานเพื่อสั่งการทำงานของระบบ พัฒนาโดยอาศัยโปรแกรมไมโครซอฟท์วิชวลซีชาร์ป ส่วนเชื่อมต่อกับอุปกรณ์ภายนอกโดยใช้ไมโครคอนโทรลเลอร์ในการรับและส่งข้อมูลผ่านพอร์ตอนุกรม และส่วนรถลำเลียงวัตถุดิบซึ่งเชื่อมต่อกับโปรแกรมคอมพิวเตอร์ที่พัฒนาขึ้นผ่านซิกบี โดยตัวรถอาศัยมอเตอร์ไฟฟ้ากระแสตรงในการขับเคลื่อน และใช้เซนเซอร์ชนิดใช้แสงเพื่อตรวจจับเส้นทางการวิ่ง และใช้ทฤษฎีการควบคุมแบบพีไอดีในการควบคุมรถลำเลียง รายละเอียดต่าง ๆ สามารถอธิบายได้ดังนี้

2.1 โปรแกรมการเชื่อมต่อกับผู้ใช้งาน

ในโครงงานนี้อาศัยโปรแกรมไมโครซอฟท์วิชวลซีชาร์ปในการพัฒนาโปรแกรม เพื่อนำมาใช้สั่งการทำงานของระบบ โดยไมโครซอฟท์วิชวลซีชาร์ปเป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมภาษาซีชาร์ป ซึ่งภาษาซีชาร์ปเป็นโปรแกรมยุคใหม่ที่ถูกสร้างขึ้นสำหรับการพัฒนาซอฟท์แวร์ภายใต้แนวคิดดอตเน็ตเฟรมเวิร์ก *(.NET Framework)* ซึ่งเป็นแนวคิดที่ได้รับความนิยมสูงที่สุดในปัจจุบัน

ดอตเน็ตเฟรมเวิร์ก คือ แพลตฟอร์มสำหรับพัฒนาซอฟท์แวร์สร้างขึ้นโดยไมโครซอฟท์ รองรับภาษาดอตเน็ตมากกว่า 40 ภาษา เช่น ภาษา C, Pascal, Java ซึ่งมีไลบรารีเป็นจำนวนมากสำหรับการเขียนโปรแกรม รวมถึงบริหารการดำเนินการของโปรแกรมบนดอตเน็ตเฟรมเวิร์ก โดยไลบรารีนั้นได้รวมถึงส่วนต่อประสานกับผู้ใช้ การเชื่อมต่อฐานข้อมูล วิทยาการเข้ารหัสลับ ขั้นตอนวิธี การเชื่อมต่อเครือข่ายคอมพิวเตอร์ และการพัฒนาเว็บแอปพลิเคชัน

โปรแกรมที่เขียนบนดอตเน็ตเฟรมเวิร์ก จะทำงานบนสภาพแวดล้อมที่บริหารโดย Common Language Runtime (CLR) ซึ่งเป็นส่วนหนึ่งในดอตเน็ตเฟรมเวิร์ก โดย CLR นั้นเตรียมสภาพแวดล้อมเสมือน ทำให้ผู้พัฒนาไม่ต้องคำนึงถึงความสามารถที่แตกต่างระหว่างหน่วยประมวลผลต่าง ๆ และ CLR ยังให้บริการด้านกลไกระบบความปลอดภัย การบริหารหน่วยความจำ และการจัดการความผิดปกติ (Exception handling) ดอตเน็ตเฟรมเวิร์กนั้นออกแบบมาเพื่อให้การพัฒนาซอฟท์แวร์ง่ายขึ้น รวดเร็วขึ้นและปลอดภัยขึ้นกว่าเดิม

ไมโครซอฟท์วิชวลซีชาร์ป สามารถสร้างฐานข้อมูลได้โดยอาศัยแอพพลิเคชัน OLEDB (Object Linking and Embedding Database) หรือ SQL server (Structured Query Language) ซึ่งมีความสามารถไม่ต่างกันมาก แต่การใช้ SQL server จำเป็นต้องมีไดร์ฟเวอร์ (Driver) ก่อนนำไปใช้ในการติดต่อกับโปรแกรมฐานข้อมูลต่าง ๆ ขณะที่ OLEDB สามารถติดต่อกับโปรแกรมฐานข้อมูลพื้นฐาน เช่น ไมโครซอฟท์แอคเซส (MS access) และ ไมโครซอฟท์เอคเซล (MS excel) ได้โดยตรง โครงงานนี้จึงเลือกใช้แอพพลิเคชัน OLEDB เพื่อง่ายต่อการทำงาน

2.2 การเชื่อมต่อกับอุปกรณ์ภายนอก

ในหัวข้อนี้กล่าวถึงการเชื่อมต่อกับอุปกรณ์ภายนอกที่ใช้ในโครงงาน ประกอบด้วย 4 ส่วน คือ ไมโครคอนโทรลเลอร์ การสื่อสารแบบอนุกรม การเชื่อมต่อพอร์ตอนุกรมมาตรฐาน RS-232 และยูอาร์ท (UART) รายละเอียดต่าง ๆ สามารถอธิบายได้ดังนี้

2.2.1 ไมโครคอนโทรลเลอร์

ในโครงงานนี้มีความจำเป็นที่จะต้องใช้การประมวลผลแบบเวลาจริง (Real time) เพราะว่าเป็นการทำงานที่เกี่ยวกับการคำนวณสมการทางคณิตศาสตร์อย่างต่อเนื่อง หากประมวลผลล่าช้าเกินไป จะทำให้ผลตอบสนองต่อสถานะนั้นผิดพลาดไปจากที่ต้องการ จึงต้องใช้หน่วยประมวลผลที่มีความเร็วสูง ในโครงงานนี้จึงเลือกใช้ไมโครคอนโทรลเลอร์ ARM STM32F103RET6 ดังรูปที่ 2.1 ซึ่งเป็นหน่วยประมวลผลแบบ 32 บิต ในตระกูล Cortex™-M3 ของบริษัท อาร์ม จำกัด มีหน่วยเชื่อมต่อและหน่วยฟังก์ชันพิเศษเพียงพอสำหรับความต้องการในโครงงานนี้

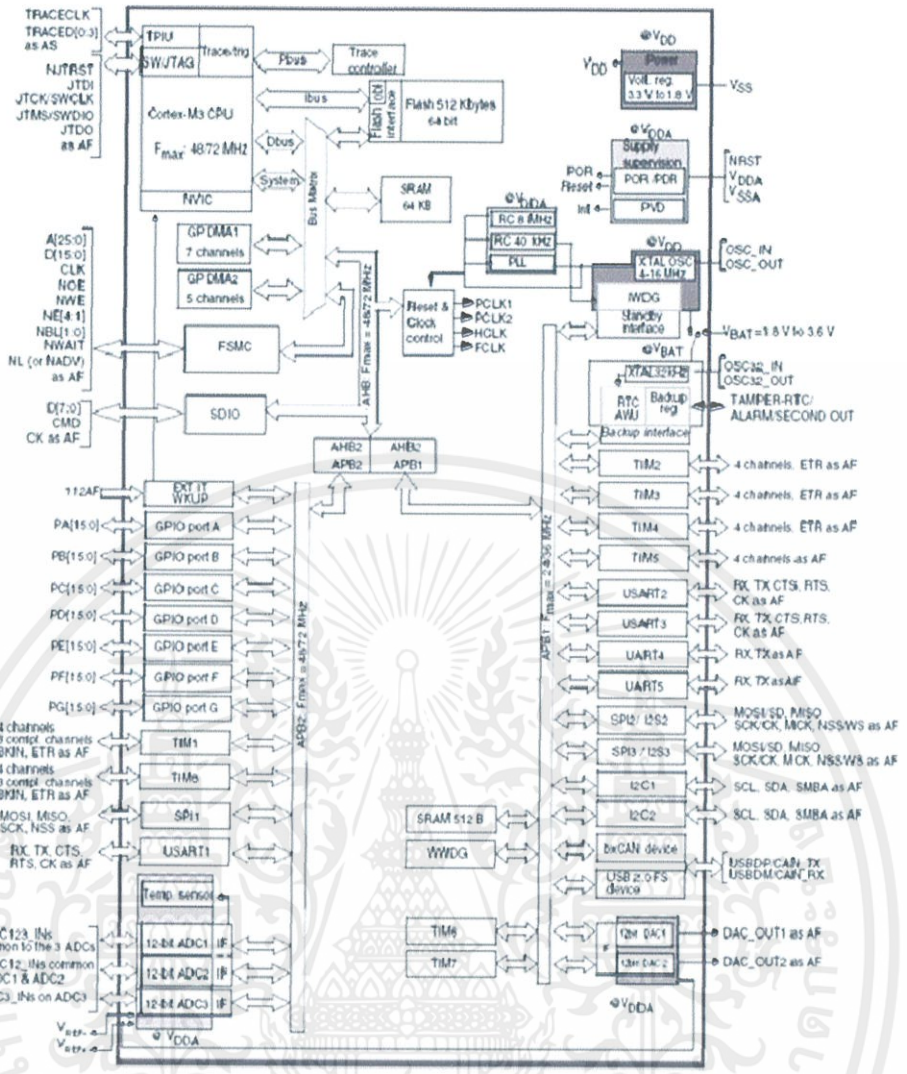
รูปที่ 2.1 ไมโครคอนโทรลเลอร์ ARM STM32F103RET6

<u>คุณสมบัติของ ARM STM32F103RET6</u>

ARM STM32F103RET6 เป็นไมโครคอนโทรลเลอร์ที่มีการประมวลผลข้อมูลแบบ 32 บิต ซึ่งมีจุดเด่นในความสามารถในการประมวลผลข้อมูลสัญญาณแบบดิจิตอลเนื่องจากไมโครคอนโทรลเลอร์มีโมดูล (Module) ที่จำเป็นในการใช้งานปกติหลายชนิดอยู่ภายในตัว ยกตัวอย่างเช่น USB, SPI, I^2C, *ADC, Timer/Counter, PMW, Capture* และ *UART* เป็นต้น จึงเหมาะแก่การใช้ในการทดลองและประยุกต์ใช้งานในหลากหลายรูปแบบ ซึ่งโครงสร้างของไมโครคอนโทรลเลอร์แสดงได้ดังรูปที่ 2.2

<u>คุณสมบัติด้านการประมวลผล</u>

- ซีพียู 32 บิต Cortex-M3 สามารถทำงานด้วยความถี่สัญญาณนาฬิกาสูงสุด 72 เมกะ-เฮิร์ทซ์ ประสิทธิภาพของความเร็วในการทำงาน 1.25 DMIPS/MHz เข้าถึงหน่วยความจำได้โดยทันที มีตัวหารเลขแบบฮาร์ดแวร์ และหน่วยประมวลผลคณิตศาสตร์ สามารถคูณเลข 32 บิต ได้ในเวลา 1 ไซเคิลนาฬิกา
- มีหน่วยความจำแบบแฟลช 512 กิโลไบต์ และสแตติกแรม 64 กิโลไบต์



รูปที่ 2.2 โครงสร้างของไมโครคอนโทรลเลอร์ ARM STM32F103RET6

- ไฟเลี้ยง 2.0 โวลต์ ถึง 3.6 โวลต์
- มีแหล่งจ่ายไฟ VBAT สำหรับโมดูลเวลานาฬิกาจริงภายในชิพและรีจิสเตอร์รักษาค่าเวลา
- รองรับการใช้คริสตอลออสซิลเลเตอร์ความถี่ 4 ถึง 16 เมกะเฮิร์ทซ์
- มีพาวเวอร์-ออนรีเซ็ต (POR) และพาวเวอร์-ดาวน์รีเซ็ต (PDR) มีวงจรตรวจจับไฟเลี้ยงแบบโปรแกรมได้ (*PVD: Programmable Voltage Detector*) รองรับการทำงานในโหมดประหยัดพลังงาน 3 โหมดคือ Sleep, Stop และ Standby
- ขาพอร์ตอินพุต-เอาต์พุตแบบความเร็วสูง 80 ขา รองรับระดับแรงดัน +5 โวลต์
- การดีบัก รองรับการดีบักแบบ Serial Wire Debug (SWD) และ JTAG

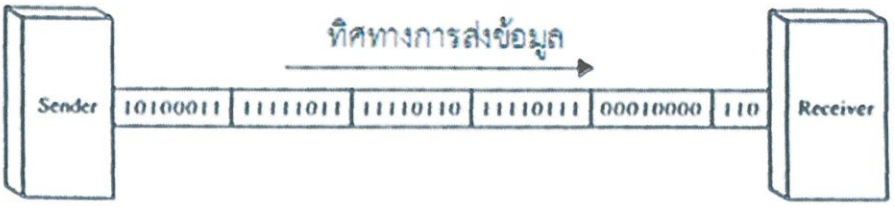
- โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล ความเร็ว 1 ไมโครวินาที ความละเอียด 12 บิต จำนวน 3 ชุด รับแรงดันได้ 0 ถึง 3.6 โวลต์ มีวงจรสุ่มและคงระดับสัญญาณ (Sample and hold) 2 ชุด และมีตัวตรวจจับอุณหภูมิอยู่ภายในชิพ เพื่อตรวจสอบอุณหภูมิภายในชิพขณะทำงาน
- DMA มีตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงหรือ DMA controller 12 ช่อง รองรับการทำงานของไทเมอร์, โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล, โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI, โมดูลเชื่อมต่อระบบบัส และโมดูลสื่อสารข้อมูลอนุกรม UART
- ไทเมอร์ 7 ชุด
  - ไทเมอร์ 16 บิต 4 ชุด รองรับการทำงานโหมดตรวจจับสัญญาณอินพุต (Input Capture: IC) โหมดเอาต์พุตเปรียบเทียบ (Output Compare: OC) โหมดกำเนิดสัญญาณ *PWM* และตัวนับพัลส์ สำหรับไทเมอร์ *TIM1* สามารถควบคุมโมดูลสัญญาณ PWM 6 ช่อง กำหนดค่าเวลาวิกฤต และการหยุดฉุกเฉินได้
  - วอทช์ดอกไทเมอร์ (Watchdog timer) 2 ชุด แบบอิสระและใช้ฐานเวลาร่วมกับฐานเวลาหลัก
  - ไทเมอร์หลักของระบบ (Systick) เป็นตัวนับแบบถอยหลัง 24 บิต
- โมดูลสื่อสารข้อมูล 9 แบบ
  - โมดูลติดต่อระบบบัส 2 ชุด รองรับ SMBus และ PMBus
  - โมดูลสื่อสารข้อมูลอนุกรม UART 3 ชุด รองรับการทำงาน ISO7816, LIN และ IrDA
  - โมดูลเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI 2 ชุด ความเร็ว 18 Mbit/s
  - โมดูลเชื่อมต่อบัส *CAN แบบ 2.0B Active*
  - โมดูลเชื่อมต่อพอร์ต USB 2.0 แบบความเร็วเต็มที่

2.2.2 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมแบ่งออกเป็น 2 แบบ คือ การสื่อสารแบบซิงโครนัส (Synchronous) และการสื่อสารแบบอะซิงโครนัส (Asynchronous)

2.2.2.1 การสื่อสารแบบซิงโครนัส

การส่งข้อมูลแบบซิงโครนัสมีการส่งสัญญาณนาฬิการ่วมกับสัญญาณข้อมูลที่ต้องการส่งและรับ การติดต่อแบบนี้ต้องใช้สายเชื่อมต่ออย่างน้อย 3 เส้น คือ สายสัญญาณนาฬิกา สายข้อมูลและสายกราวด์ ข้อดีของการส่งแบบนี้ คือ สามารถควบคุมเครื่องส่งและรับให้ทำงานได้พร้อมกันอย่างถูกต้อง การส่งข้อมูลจะส่งเป็นบิตที่ต่อเนื่องกันไป ผู้รับจะแยกบิตเหล่านี้เองออกมาเป็นไบต์หรือเป็นตัวอักษร รูปที่ 2.3 แสดงแนวคิดการส่งข้อมูลแบบซิงโครนัส โดยส่งเป็นบิตติดต่อกันยาว ๆ ถ้าผู้ส่งต้องการแบ่งช่วงกลุ่มข้อมูลก็ส่งกลุ่มบิต 0 หรือ 1 เพื่อแสดงสถานะว่าง เมื่อบิตมาถึงผู้รับ ผู้รับจะนับจำนวนบิตแล้วจับกลุ่มของบิตให้เป็นไบต์ที่มี 8 บิต การส่งข้อมูลแบบซิงโครนัสมีประสิทธิภาพมากกว่าการส่งข้อมูลแบบอะซิงโครนัสมาก และทำให้ใช้ความสามารถของสายสื่อสารได้เกือบทั้งหมด มีการส่งข้อมูลเร็วกว่าแบบอะซิงโครนัส เพราะไม่มีบิตพิเศษหรือช่องว่างที่ไม่ได้ใช้งานเมื่อถึงผู้รับ ด้วยเหตุนี้จึงนำไปใช้งานที่ต้องการความเร็วสูง เช่น การส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์

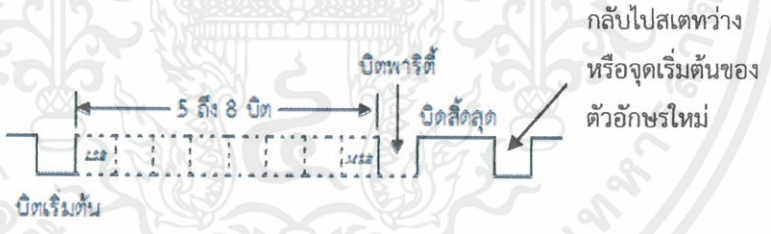


รูปที่ 2.3 รูปแบบการส่งข้อมูลแบบซิงโครนัส

2.2.2.2 การสื่อสารแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การรับส่งข้อมูลในสายโดยไม่จำเป็นต้องมีสายสัญญาณนาฬิการ่วมด้วย ซึ่งแตกต่างจากการสื่อสารข้อมูลแบบอะซิงโครนัสที่ต้องใช้สายสัญญาณนาฬิกาในการรับและส่งสัญญาณเพิ่มอีกหนึ่งเส้น สำหรับการสื่อสารข้อมูลแบบอะซิงโครนัส จะกำหนดสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือบอดเรท (Baudrate) มีหน่วยเป็นบิตต่อวินาที (Bit per second) รูปที่ 2.4 แสดงรูปแบบของข้อมูลที่ใช้ในการสื่อสารอนุกรมแบบอะซิงโครนัส มีส่วนประกอบด้วยกัน 4 ส่วน คือ

- บิตเริ่มต้น (Start bit) มีขนาด 1 บิต
- บิตข้อมูลแบบอนุกรม มีขนาด 5, 6, 7 หรือ 8 บิต
- บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มีก็ได้
- บิตสิ้นสุด (Stop bit) มีขนาด 1 ถึง 2 บิต

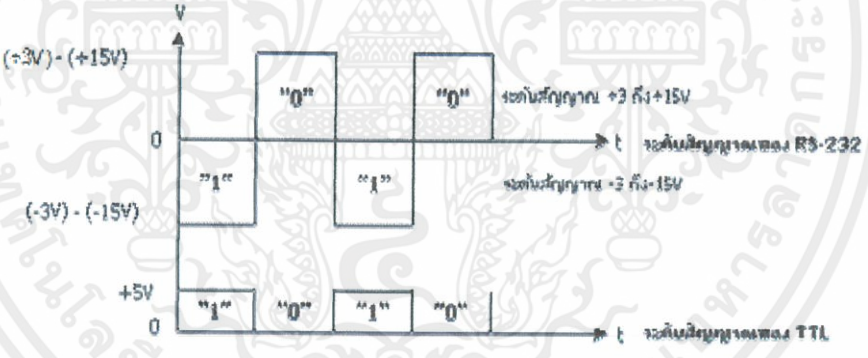


รูปที่ 2.4 รูปแบบการส่งข้อมูลแบบอะซิงโครนัส

หลักการทำงานของการส่งข้อมูลแบบอะซิงโครนัสในรูปที่ 2.4 สามารถอธิบายได้ว่าขณะที่ไม่มีข้อมูลส่งมา สัญญาณในสายข้อมูลจะมีลอจิกเป็น “1” (High) เรียกสถานะนี้ว่าสถานะหยุดรอ (Waiting stage) การเริ่มต้นส่งข้อมูลเริ่มจากให้สัญญาณในสายข้อมูลมีลอจิก “0” (Low) ด้วยช่วงขนาด 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น จากนั้นจะส่งบิตข้อมูลออกไป โดยเริ่มส่งจากบิตนัยสำคัญต่ำสุด (Least Significant Bit: LSB) ก่อน จนถึงบิตนัยสำคัญสูงสุด (Most Significant Bit: MSB) และตามด้วยบิตพาริตี้ ใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสิ้นสุดที่ส่งจะทำให้สัญญาณในสายข้อมูลมีสถานะลอจิก “1” อีกครั้ง เพื่อกลับเข้าสู่สถานะหยุดรอ แสดงว่าสิ้นสุดข้อมูล

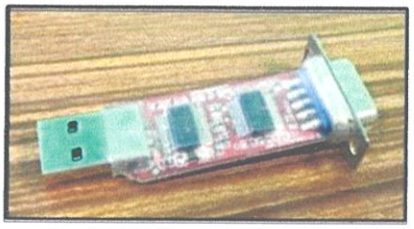
2.2.3 การเชื่อมต่อพอร์ตอนุกรมมาตรฐาน RS-232

RS-232 ย่อมาจาก Recommended Standard-232 เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรม กำหนดโดย EIA (Electronics Industry Association) หรือสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ใช้กับการสื่อสารแบบจุดต่อจุด โดยใช้สายเชื่อมต่อ DB แบบ 25 และ 9 เข็ม ออกแบบมาเพื่อใช้ส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยนำเอา RS-232 ไปประยุกต์ใช้ส่งข้อมูลระหว่างคอมพิวเตอร์กับอุปกรณ์ต่อพ่วงต่าง ๆ มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment: DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating: DCT) ได้ว่า อุปกรณ์ DTE ต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์ที่สามารถสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCT ทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับและส่งข้อมูลระหว่างอุปกรณ์ทั้งสองผ่านมาตรฐาน RS-232 ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อุปกรณ์ภายนอกจะเป็นแบบ DCT การรับส่งสัญญาณจะกำหนดความยาวสูงสุดไว้ที่ไม่เกิน 50 ฟุต สำหรับการส่งสัญญาณที่ความเร็ว 19,200 บิตต่อวินาที ในขณะที่ความยาวสายจะต้องสั้นลงถ้าต้องการสื่อสารที่ความเร็วสูงขึ้น โดยมีระดับสัญญาณตั้งแต่ +3 โวลต์ จนถึง +15 โวลต์ สำหรับลอจิก “0” และมีระดับแรงดันที่ -3 โวลต์ จนถึง -15 โวลต์ สำหรับลอจิก “1” ซึ่งไม่สามารถนำเอาต์พุตใด ๆ ต่อเข้ากับลอจิกเกตใช้งานได้โดยตรง ต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันก่อน โดยปกติใช้ไอซีจำพวก RS-232 เพื่อแปลงระดับแรงดัน ให้อยู่ในระดับทีทีแอล (TTL: Transistor-Transistor Logic) โดยลอจิก “0” จากเดิมมีระดับสัญญาณ +3 โวลต์ ถึง +15 โวลต์ จะถูกแปลงเป็น 0 โวลต์ ส่วนลอจิก “1” จากเดิมมีระดับสัญญาณ -3 โวลต์ ถึง -15 โวลต์ จะถูกแปลงเป็น 5 โวลต์ ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 ระดับแรงดันสัญญาณของพอร์ตอนุกรม RS-232 เทียบกับ TTL

ในยุคปัจจุบัน เครื่องคอมพิวเตอร์ที่นิยมใช้งานกันมากคือ เครื่องคอมพิวเตอร์แบบพกพาได้หรือโน้ตบุ๊ก (Notebook) ซึ่งโดยมากไม่มีพอร์ตอนุกรม RS-232 สำหรับเชื่อมต่อกับอุปกรณ์จากภายนอก ดังนั้นจึงต้องมีอุปกรณ์ที่ใช้สำหรับแปลงพอร์ตยูเอสบี (USB) ซึ่งเครื่องคอมพิวเตอร์ทุกเครื่องมีรองรับอยู่ แปลงเป็นพอร์ตอุนกรม RS-232 เพื่อใช้เชื่อมต่อกับอุปกรณ์ภายนอก ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 อุปกรณ์แปลงพอร์ต USB เป็นพอร์ตอนุกรม RS-232

2.2.4 ยูอาร์ท

ยูอาร์ท หรือที่นิยมเรียกว่า UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึง วงจรรวมในเครื่องคอมพิวเตอร์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรม หน้าที่หลักของ UART คือ ทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัสแล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าคอมพิวเตอร์ นอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังทำการแจ้งข้อมูลอื่น ๆ ให้คอมพิวเตอร์ทราบด้วย เช่น อัตราความเร็วในการรับส่งข้อมูล, รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

ภายใน UART มีส่วนของวงจรสร้างอัตราการถ่ายทอดข้อมูลแบบโปรแกรม โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART ตัวหารนี้มีขนาด 16 บิต นิยมกำหนดให้อยู่ในช่วง 10 – 65,535 โดย UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (Half duplex) และฟูลดูเพล็กซ์ (Full duplex) โดยการส่งแบบฮาล์ฟดูเพล็กซ์ เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน ชนิดของ UART ในเครื่องคอมพิวเตอร์ที่ใช้ทั่วไปมีอยู่ 2 เบอร์ คือ 8250 กับ 16450 โดยเบอร์ 8250 เป็น UART มาตรฐานที่มีใช้กันมายาวนาน UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและการส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ก็ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุก ๆ รุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้ กับอีกหนึ่งเบอร์ คือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 115, 200 บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO (First In First Out) ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5 โวลต์ และ +3 โวลต์ มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาทีเมื่อใช้สัญญาณนาฬิกา 16 เมกะเฮิร์ทซ์

การใช้งานโมดูล UART ด้วยคอมไพเลอร์ CCS ได้จัดเตรียมฟังก์ชันสำเร็จรูป (Build-in function) ไว้ให้เรียบร้อยแล้ว สามารถเรียกใช้งานฟังก์ชันได้โดยสามารถใช้งาน UART ได้ทั้งแบบฮาร์ดแวร์หรือแบบซอฟท์แวร์ก็ได้ โดยแบบฮาร์ดแวร์หมายถึงการใช้งานตัวโมดูล UART ที่มีอยู่ในไมโครคอนโทรลเลอร์ ซึ่งสามารถที่จะรับส่งข้อมูลผ่านทางอินเทอร์รัพท์ได้ ส่วนแบบซอฟท์แวร์นั้น

หมายถึงการใช้ซอฟท์แวร์ จำลองการทำงานของ UART ขึ้นมา ไม่สามารถรับและส่งข้อมูลด้วยวิธีอินเทอร์รัพท์ได้

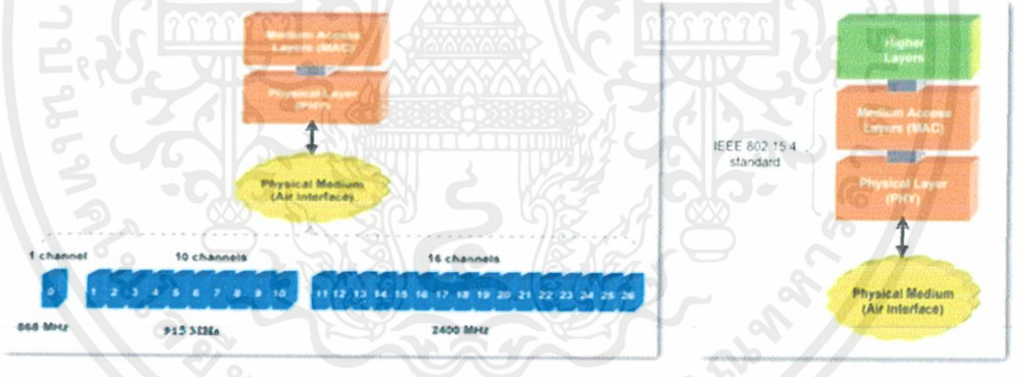
2.3 การลำเลียงวัตถุดิบ

ในหัวข้อนี้กล่าวถึงความรู้ซึ่งจำเป็นต้องใช้ในการพัฒนาส่วนการลำเลียงวัตถุดิบที่ใช้ในโครงงาน ประกอบด้วย 4 ส่วน คือ ซิกบี (Zigbee) มอเตอร์ไฟฟ้ากระแสตรง ออพติคัลเซนเซอร์ (Optical sensor) และตัวควบคุมในกลุ่มพีไอดี รายละเอียดต่าง ๆ สามารถอธิบายได้ดังนี้

2.3.1 ซิกบี

ซิกบี เป็นมาตรฐานการสื่อสารแบบไร้สายสากลประเภทหนึ่ง กำหนดโดย ZigBee Alliance โดยมีอัตราการรับส่งข้อมูลต่ำ ใช้พลังงานต่ำ ราคาถูก จุดประสงค์เบื้องต้นเพื่อประยุกต์ใช้สร้างระบบเครือข่ายเซนเซอร์แบบไร้สาย (Wireless sensor network) ซึ่งระบบนี้สามารถทำงานทั้งในร่มและกลางแจ้ง ทนแดด ทนฝน และสามารถทำงานได้โดยอาศัยแบตเตอรี่ขนาดเล็ก (เช่น ถ่าน AA) เป็นระยะเวลานานซึ่งอาจเกินหนึ่งปี เหมาะสมในการใช้งานด้านการติดตามผลต่าง ๆ

ซิกบีกำหนดย่านความถี่ใช้งานตามมาตรฐานไว้ 3 ย่านความถี่คือ ย่าน 2.4 กิกกะเฮิร์ทซ์ มีอัตราการรับส่งข้อมูลที่ 250 kbps มีช่องสัญญาณ 16 ช่อง แต่ละช่องอาศัยแบนด์วิดท์ (Bandwidth) ประมาณ 5 เมกะเฮิร์ทซ์, ย่าน 915 เมกะเฮิร์ทซ์ มีอัตราการรับส่งข้อมูลที่ 40 kbps มีช่องสัญญาณ 10 ช่อง แต่ละช่องอาศัยแบนด์วิดท์ประมาณ 2 เมกะเฮิร์ทซ์ และย่าน 868 เมกะเฮิร์ทซ์ มีอัตราการรับส่งข้อมูลที่ 20 kbps มีช่องสัญญาณ 1 ช่อง สามารถแสดงได้ดังรูปที่ 2.7 (ก)

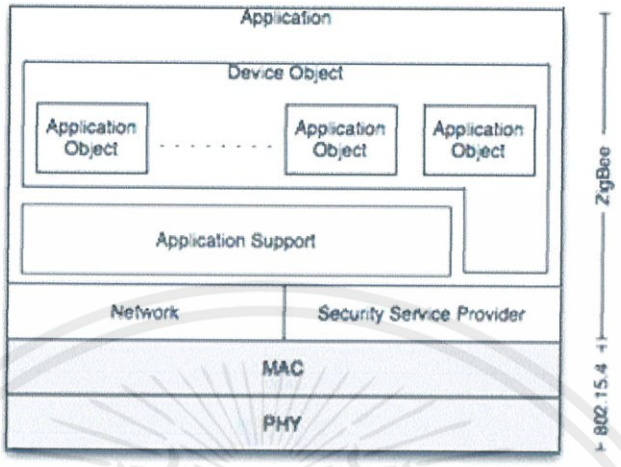


(ก) ย่านความถี่ใช้งานตามมาตรฐาน (ข) ซิกบีเลเยอร์

รูปที่ 2.7 มาตรฐานการสื่อสารซิกบี

ซิกบีนำ Physical layer และ MAC layer ของ IEEE 802.15.4 ซึ่งเป็นมาตรฐานการกำหนดการสื่อสารไร้สายแบบ WPAN (Wireless Personal Area Network) มาทำงานในเลเยอร์ที่ต่ำกว่า (2 เลเยอร์ล่างสุด) ดังรูปที่ 2.7 (ข) เช่น เรื่องของระดับกำลังสัญญาณ คุณภาพการเชื่อมต่อ การควบคุมการเข้าถึงข้อมูล การป้องกัน ฯลฯ ในเลเยอร์ถัดไปจากชั้น 802.15.4 จะเป็นรูปแบบของ

ZigBee higher layers เรียกว่า มาตรฐานซิกบี เลเยอร์นี้ประกอบไปด้วย แอพพลิเคชั่นโปรไฟล์ (Application Profile) ผู้ให้บริการรักษาความปลอดภัย (Security service provider) และโครงข่าย (Network) ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 โครงสร้างซอฟท์แวร์ของซิกบี

จากที่กล่าวมาซิกบีสามารถสร้างเป็นเครือข่ายได้เพราะอิงมาตรฐานตาม IEEE 802.15.4 และมีการจัดการในแบบของซิกบีในเลเยอร์ถัดไป ทั้งนี้ IEEE 802.15.4 แบ่งชนิดอุปกรณ์ในเครือข่ายออกเป็น 2 ประเภท คือ RFD (Reduced Function Device) และ FFD (Full Function Device) โดย RFD หมายถึงอุปกรณ์ที่ลดความสามารถลง จึงใช้หน่วยความจำน้อยกว่า การประมวลผลลดลง และใช้พลังงานในการดำเนินการต่ำลง เงื่อนไขคือ RFD สามารถคุยกับ FFD ได้เพียงตัวเดียวเท่านั้นในเครือข่ายและไม่สามารถคุยกันเองกับ RFD ได้ ส่วน FFD หมายถึงอุปกรณ์ที่สามารถทำงานร่วมกับเครือข่ายและอุปกรณ์อื่น ๆ ได้ ต้องการทรัพยากรต่าง ๆ มากกว่า RFD (เช่น หน่วยความจำ พลังงาน ความสามารถประมวลผล)

ซิกบีแบ่งได้ตามลักษณะการทำงาน 3 แบบ คือ

1. *Coordinator* มีหน้าที่สร้างการสื่อสาร เชื่อมโยงเครือข่ายระหว่าง End Device กับ Router หรือ Coordinator กับ Coordinator ด้วยกัน หรือ Coordinator กับ Router กำหนดแอดเดรส (address) ให้กับอุปกรณ์ที่อยู่ในวงเครือข่ายไม่ให้ซ้ำกัน ดูแลเรื่องการกำหนดเส้นทาง
2. End device เป็นอุปกรณ์ปลายทางสุด ซึ่งจะใช้รับสัญญาณจากเซนเซอร์ที่ปลายทาง โดยใช้พลังงานต่ำในการทำงาน
3. Router มีหน้าที่รับ-ส่งข้อมูล ในเส้นทางต่าง ๆ ของเครือข่าย

<u>คุณสมบัติโดยทั่วไปของซิกบี</u>

- อัตราการส่งข้อมูลมี 3 ระดับ คือ 250 kbps (ที่ความถี่ 2.4 กิกกะเฮิร์ทซ์) 40 kbps (ที่ความถี่ 915 เมกะเฮิร์ทซ์) และ 20 kbps (ที่ความถี่ 868 เมกะเฮิร์ทซ์)
- สายอากาศ: มีสายอากาศแบบ Whip

- ระยะทำการในร่ม: สูงสุดประมาณ 100 เมตร
- ระยะทำการกลางแจ้ง (แบบ line-of-sight): สูงสุดประมาณ 1,500 เมตร
- กำลังส่ง: 60 mW (18 dBm)
- ความไวในการรับสัญญาณ: -100 dBm (1% packet error rate)
- การทำงานของพอร์ต: สามารถกำหนดผ่านทางซอฟท์แวร์ X-CTU เพื่อให้การทำงานเป็นอินพุตอนาลอกสำหรับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ความละเอียด 10 บิต และอินพุตเอาต์พุตดิจิตอล
- ขนาด: 0.96x1.297 นิ้ว หรือ 2.438x3.294 เซนติเมตร
- ไฟเลี้ยง: 2.8 ถึง 3.4 โวลต์
- กระแสไฟฟ้า: เมื่อส่งข้อมูล 215 มิลลิแอมแปร์, รับข้อมูล 55 มิลลิแอมแปร์, น้อยกว่า 10 ไมโครแอมแปร์ ในโหมดลดพลังงานที่ไฟเลี้ยง +3.3 โวลต์
- อุณหภูมิใช้งาน: -40 ถึง 85 เซลเซียส

<u>คุณสมบัติการสื่อสารข้อมูล</u>

- สามารถทำงานเป็นอุปกรณ์มาสเตอร์และสเลฟได้
- อัตราถ่ายทอดข้อมูลผ่านคลื่นวิทยุ: 250,000 บิตต่อวินาที
- อัตราการถ่ายทอดข้อมูลอนุกรม (บอดเรท): 1,200 ถึง 115,200 บิตต่อวินาที
- รูปแบบโครงข่ายข้อมูลที่รองรับ: จุดต่อจุด (point-to-point), จุดต่อหลายจุด (point-to-multipoint) และเข้ากันได้กับอุปกรณ์ตามมาตรฐานรหัส 802.15.4
- ทางเลือกแอดเดรส: PAN ID, ช่อง และแอดเดรส สำหรับแอดเดรสสามารถกำหนดรหัสแอดเดรสได้มากถึง 65,000 รหัส
- เทคโนโลยีในการกระจายคลื่น: DSSS (Direct Sequence Spread Spectrum)
- รองรับการทำงานทั้งแบบ API และ AT command สามารถกำหนดได้ผ่านทางซอฟท์แวร์ X-CTU

<u>การรับรองมาตรฐาน</u>

- สหรัฐอเมริกา (FCC Part 15.247) OUR-XBEEPRO
- แคนาดา (IC) 4214A XBEEPRO
- ยุโรป (CE) ETSI (ที่กำลังส่งสูงสุด 10 dBm)
- ญี่ปุ่น 005NYCV0378 (ที่กำลังส่งสูงสุด 10 dBm)

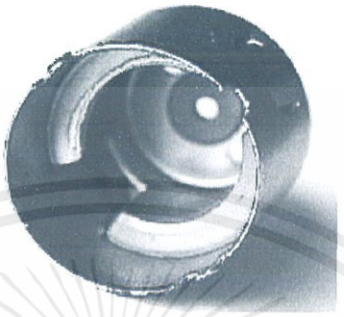
2.3.2 มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงเป็นต้นกำลังขับเคลื่อนที่สำคัญอย่างหนึ่งในทางอุตสาหกรรมเพราะมีคุณสมบัติที่เด่นในด้านการปรับความเร็วได้ นิยมใช้กันมากในโรงงานอุตสาหกรรม เช่น โรงทอผ้า โรงงานเส้นใยโพลีเอสเตอร์ โรงงานถลุงโลหะ หรือใช้เป็นต้นกำลังในการขับเคลื่อนรถไฟฟ้า เป็นต้น

2.3.2.1 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

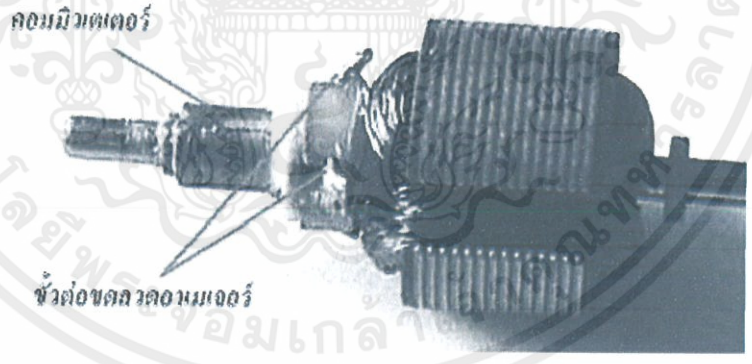
มอเตอร์ไฟฟ้ากระแสตรงมีส่วนประกอบที่สำคัญ 2 ส่วนดังนี้

1. ส่วนที่อยู่กับที่ หรือ สเตเตอร์ (Stator) ประกอบด้วย แม่เหล็กถาวร 2 ขั้ว วางอยู่ระหว่างขดลวดอาร์เมเจอร์ เมื่อป้อนแรงดันไฟฟ้าให้ขดลวดอาร์เมเจอร์หมุนทำให้เกิดสนามแม่เหล็ก 2 ชุด ตัดผ่านกันจึงเกิดแรงดูดและผลักกันระหว่างสนามแม่เหล็ก 2 ชุด แสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 สเตเตอร์ของมอเตอร์กระแสตรง

2. *ส่วนที่เคลื่อนที่ หรือ โรเตอร์ (Rotor) ประกอบด้วยขดลวดอาร์เมเจอร์ ที่พันอยู่บนแกนเหล็กอาร์เมเจอร์* และมีคอมมิวเตเตอร์ยึดติดอยู่ที่ปลายของขดลวดอาร์เมเจอร์ ดังรูปที่ 2.10 ซึ่งคอมมิวเตเตอร์ทำหน้าที่สัมผัสกับแปรงถ่านคาร์บอนที่อยู่ในโรเตอร์เพื่อให้กระแสไหลผ่านไปยังขดลวดอาร์เมเจอร์ ทำให้เกิดการสร้างสนามแม่เหล็กขึ้น เกิดการหักล้างหรือเสริมกันกับสนามแม่เหล็กที่เกิดในสเตเตอร์ จึงทำให้มอเตอร์หมุนได้



รูปที่ 2.10 โรเตอร์ของมอเตอร์ไฟฟ้ากระแสตรง

ส่วนประกอบที่สำคัญของโรเตอร์มีดังนี้

- แกนเพลา (Shaft) ใช้ยึดคอมมิวเตเตอร์และยึดแกนเหล็กอาร์เมเจอร์ประกอบกันเป็นตัวโรเตอร์ แกนเพลานี้จะวางอยู่บนแบริ่ง เพื่อบังคับให้หมุนอยู่ในแนวนิ่งไม่มีการสั่นสะเทือนได้

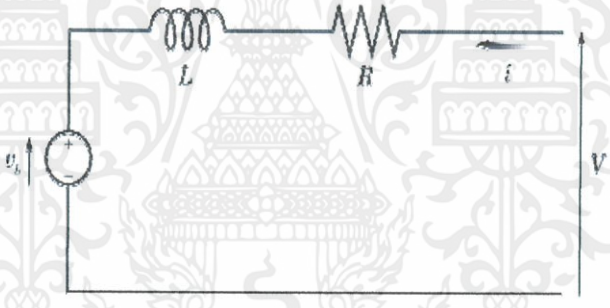
- แกนเหล็กอาร์เมเจอร์ทำด้วยแผ่นเหล็กบางอาบฉนวน (Laminated Sheet Steel) เป็นที่สำหรับใช้พันขดลวดอาร์เมเจอร์ซึ่งสร้างแรงบิด

- คอมมิวเตเตอร์ทำด้วยทองแดงทำการออกแบบเป็นซี่ แต่ละซี่มีฉนวนไมก้า (Mica) คั่นระหว่างซี่ของคอมมิวเตเตอร์ ส่วนหัวของคอมมิวเตเตอร์จะมีร่องสำหรับใส่ปลายสาย ของขดลวดอาร์เมเจอร์ ตัวคอมมิวเตเตอร์นี้อัดแน่นติดกับแกนเพลา เป็นรูปกลมทรงกระบอก มีหน้าที่สัมผัสกับแปรงถ่าน (Brushes) เพื่อรองรับกระแสจากสายป้อนเข้าไปยัง ขดลวดอาร์เมเจอร์เพื่อสร้างเส้นแรงแม่เหล็กอีกส่วนหนึ่งให้เกิดการหักล้างและเสริมกันกับเส้นแรงแม่เหล็กอีกส่วน

- ขดลวดอาร์เมเจอร์เป็นขดลวดพันอยู่ในร่องสลอต (Slot) ของแกนอาร์เมเจอร์ ขนาดของลวดจะเล็กหรือใหญ่และจำนวนรอบจะมากหรือน้อยนั้นขึ้นอยู่กับการออกแบบของตัวโรเตอร์ชนิดนั้น ๆ เพื่อให้เหมาะสมกับงานที่ต้องการ

2.3.2.2 คุณสมบัติเบื้องต้นของมอเตอร์ไฟฟ้ากระแสตรง

เมื่อจ่ายกระแสผ่านขั้วต่อที่เรียกว่าแปรงถ่าน ไปยังวงแหวนพิเศษที่เรียกว่าคอมมิวเตเตอร์ ซึ่งต่อเข้ากับวงรอบตัวนำ กระแสที่ไหลผ่านตัวนำจะทำให้เกิดเส้นแรงแม่เหล็กรอบตัวนำ โดยด้านหนึ่งเกิดเป็นแรงผลักขึ้น ส่วนอีกด้านเกิดแรงผลักตัวนำลงมา ทำให้วงรอบตัวนำมีการหมุน โดยแรงที่เกิดจะแปรผันตาม กระแสที่ไหลผ่าน ความยาวของตัวนำ และความเข้มของสนามแม่เหล็กสเตเตอร์ จากการทำงานจะพิจารณาจากวงจรสมมูลดังรูปที่ 2.11



รูปที่ 2.11 วงจรสมมูลของมอเตอร์ไฟฟ้ากระแสตรง

จากวงจรสมมูลของมอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรม ดังรูปที่ 2.11 แรงดันไฟฟ้าต้านกลับในมอเตอร์ไฟฟ้ากระแสตรงมีค่าเท่ากับ

$$E_a = V_t - I_a(R_a + R_{sr}) \qquad (2.1)$$

เมื่อ V_t คือ แรงดันไฟฟ้าที่จ่ายให้กับมอเตอร์ไฟฟ้ากระแสตรง หน่วย โวลต์

E_a คือ แรงดันไฟฟ้าต้านกลับที่เกิดขึ้นในอาร์เมเจอร์ หน่วย โวลต์

I_a คือ กระแสที่ไหลผ่านอาร์เมเจอร์ หน่วย แอมป์

R_a คือ ค่าความต้านของอาร์เมเจอร์ หน่วย โอห์ม

R_{sr} คือ ค่าความต้านทานของขดลวด หน่วย โอห์ม

2.3.2.3 การขับมอเตอร์ไฟฟ้ากระแสตรงขั้นพื้นฐาน

การควบคุมมอเตอร์ไฟฟ้ากระแสตรงพื้นฐาน สามารถแบ่งได้ดังนี้

- *การขับด้วยตัวต้านทานปรับค่าได้*

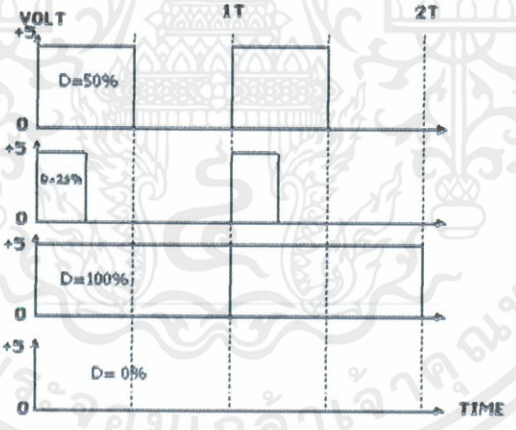
การขับด้วยตัวต้านทานปรับค่าได้เป็นรูปแบบพื้นฐานที่สุดของการควบคุมมอเตอร์ไฟฟ้ากระแสตรง คือใช้ตัวต้านทานปรับค่าได้ต่ออนุกรมกับมอเตอร์ โดยตัวต้านทานปรับค่าได้จะเป็นตัวกำหนดความเร็วในการหมุนของมอเตอร์ การบังคับแบบนี้ไม่มีประสิทธิภาพเพราะกำลังไฟสูญเสียไปในตัวต้านทาน มักนิยมใช้กับมอเตอร์ขนาดเล็ก การบังคับแบบนี้ให้คุณสมบัติการสตาร์ทดี (ให้แรงบิดสูงที่ความเร็วต่ำ) ดังนั้นการบังคับแบบนี้มีประโยชน์เฉพาะภาวะที่แรงต้านคงที่

- การขับด้วยวิธีเปลี่ยนค่าแรงดัน

วิธีการขับด้วยวิธีเปลี่ยนค่าแรงดันนี้ดีกว่าวิธีการแรกแต่จะซับซ้อนกว่าต้องใช้อุปกรณ์อิเล็กทรอนิกส์ที่อัตราขยายกำลังสูงและมอเตอร์จะถูกป้อนด้วยแรงดันที่เปลี่ยนแปลงค่าได้จากแหล่งจ่ายที่มีอิมพีแดนซ์ต่ำ

- การขับด้วยสัญญาณพีดับบลิวเอ็ม (PWM : Pulse Width Modulation)

สัญญาณพีดับบลิวเอ็ม คือ การปรับความกว้างของพัลส์ทำให้ได้ค่าเฉลี่ยของสัญญาณเปลี่ยนแปลง หากความกว้างของพัลส์น้อยจะได้ค่าเฉลี่ยต่ำในทางกลับกันหากความกว้างพัลส์มากจะได้ค่าเฉลี่ยมาก เพื่อใช้ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง ตัวอย่างสัญญาณพีดับบลิวเอ็มแสดงดังรูปที่ 2.12



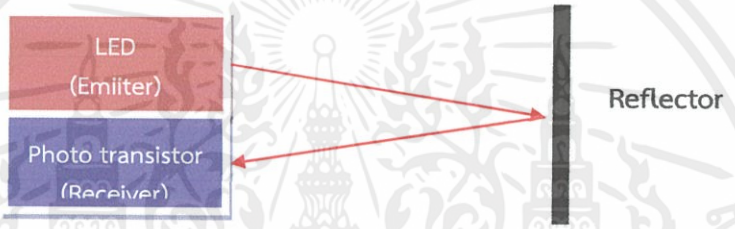
รูปที่ 2.12 สัญญาณพีดับบลิวเอ็ม

จากรูปที่ 2.12 จะพบว่าสัญญาณมีค่า Duty Cycle 3 ค่า ได้แก่ 25 50 และ100 เปอร์เซ็นต์ของคาบสัญญาณทั้งหมด คือ การขับมอเตอร์ด้วยระดับแรงดันไฟฟ้ากระแสตรง 25 เปอร์เซ็นต์ ของระดับแรงดันไฟฟ้าสูงสุดในวงจรขับ 50 เปอร์เซ็นต์ ของระดับแรงดันไฟฟ้าสูงสุดในวงจรขับ และ 100 เปอร์เซ็นต์ ของระดับแรงดันไฟฟ้าสูงสุดในวงจรขับ ตามลำดับ

2.3.3 ออพติคัลเซนเซอร์

ออพติคัลเซนเซอร์หรือเซนเซอร์ชนิดใช้แสง โดยทั่วไปใช้ในงานการตรวจจับการเคลื่อนไหว การตรวจจับวัตถุ และการตรวจสอบขนาดรูปร่างของวัตถุ เซนเซอร์ชนิดนี้ทำงานโดยอาศัยหลักการส่งและรับแสง มีส่วนประกอบสำคัญ 2 ส่วนคือ ตัวส่งแสง (Emitter) และตัวรับแสง (Receiver) ลักษณะการตรวจจับเกิดจากการที่ลำแสงจากตัวส่งแสง ส่งไปสะท้อนกับวัตถุหรือถูกขวางกั้นด้วยวัตถุ ส่งผลให้ตัวรับแสงรู้สภาวะที่เกิดขึ้นและเปลี่ยนแปลงสภาวะของสัญญาณทางด้านเอาต์พุตเพื่อนำไปใช้งานต่อไป

อุปกรณ์ที่ใช้เป็นตัวส่งแสงโดยทั่วไป คือ LED (Light Emitting Diode) และตัวรับแสงส่วนใหญ่นิยมใช้ คือ โฟโต้ไดโอด (Photo diode) หรือโฟโต้ทรานซิสเตอร์ (Photo transistor) ซึ่งมีคุณสมบัติในการทำงานเปลี่ยนแปลงตามแสงที่ตกกระทบตัวมัน ลำแสงที่ใช้เป็นตัวกลางในการส่งผ่านข้อมูลระหว่างแหล่งกำเนิดแสงกับตัวตรวจจับแสงหรือตัวรับ นิยมใช้แสงสีแดง (Visible red light) หรือแสงอินฟราเรด (Infrared light) โดยแหล่งกำเนิดแสงสีแดงสามารถติดตั้งและบำรุงรักษาได้ง่ายกว่า ในขณะที่แสงอินฟราเรดนั้นมีผลการรบกวนสัญญาณจากแหล่งกำเนิดแสงอื่นน้อยกว่า



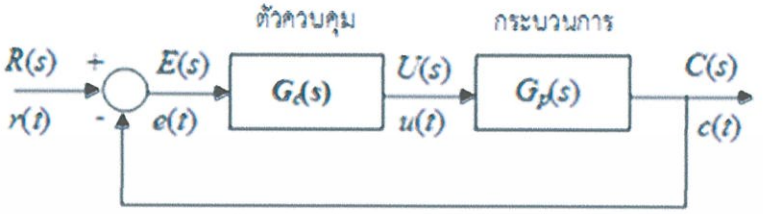
รูปที่ 2.13 หลักการทำงานของออพติคัลเซนเซอร์

รูปที่ 2.13 แสดงถึงหลักการทำงานของออพติคัลเซนเซอร์ สามารถอธิบายได้ดังนี้ แหล่งกำเนิดแสง LED สร้างแสงสีแดงหรือแสงอินฟราเรด เมื่อวัตถุเคลื่อนที่มาตรงกับตำแหน่งที่ได้ติดตั้งเซนเซอร์ไว้ จะทำให้ลำแสงดังกล่าวตกกระทบวัตถุนั้น แล้วสะท้อนกลับมายังโฟโตทรานซิสเตอร์ทำให้เกิดสัญญาณเอาต์พุตปรากฏให้ผู้ควบคุมทราบถึงตำแหน่งปัจจุบันของวัตถุ หรืออาจจะใช้เพื่อนับจำนวนวัตถุที่ได้ผ่านตำแหน่งนี้ไป ระยะห่างในการติดตั้งออพติคัลเซนเซอร์กับวัตถุที่ต้องการตรวจสอบโดยทั่วไปนั้นมีค่าประมาณ 1 มิลลิเมตร ถึงสูงสุด 5 เซนติเมตร ออพติคัลเซนเซอร์ที่ใช้ในโครงงานนี้ คือ TCRT5000 แสดงได้ดังรูปที่ 2.14

รูปที่ 2.14 ออพติคัลเซนเซอร์ที่ใช้ในโครงงาน

2.3.4 ตัวควบคุม

ตัวควบคุม เป็นส่วนประกอบหนึ่งในระบบควบคุมแบบป้อนกลับ ซึ่งมีสัญญาณความผิดพลาดเป็นอินพุต และมีเอาต์พุตเป็นอินพุตของอุปกรณ์ขับสำหรับระบบที่ต้องการควบคุม บล็อกไดอะแกรมของระบบควบคุมป้อนกลับแบบลบหนึ่งหน่วยซึ่งนิยมใช้งานโดยทั่วไป แสดงดังรูปที่ 2.15



รูปที่ 2.15 บล็อกไดอะแกรมของระบบควบคุมป้อนกลับแบบลบหนึ่งหน่วย

เมื่อ $r(t)$ และ $R(s)$ คือ อินพุตอ้างอิงของระบบบนโดเมนเวลาและบนโดเมนความถี่เชิงซ้อน

$e(t)$ และ $E(s)$ คือ ค่าความผิดพลาดบนโดเมนเวลาและบนโดเมนความถี่เชิงซ้อน

$u(t)$ และ $U(s)$ คือ เอาต์พุตของตัวควบคุมบนโดเมนเวลาและบนโดเมนความถี่เชิงซ้อน

$c(t)$ และ $C(s)$ คือ เอาต์พุตของระบบบนโดเมนเวลาและบนโดเมนความถี่เชิงซ้อน

$G_p(s)$ คือ ฟังก์ชันถ่ายโอนของพลานท์หรือกระบวนการที่ต้องการควบคุม

$G_c(s)$ คือ ฟังก์ชันถ่ายโอนของตัวควบคุม

ในหัวข้อนี้จะกล่าวถึงตัวควบคุมพื้นฐานในกลุ่มตัวควบคุมแบบพีไอดี ซึ่งเกิดจากการควบคุมพื้นฐานที่นิยม 3 รูปแบบ ได้แก่

1. การควบคุมแบบพี (P: Proportional control)
2. การควบคุมแบบไอ (I: Integral control)
3. การควบคุมแบบดี (D: Derivative control)

และการนำการควบคุม 3 ประเภทนี้มาใช้งานร่วมกัน โดยรายละเอียดระบบควบคุมที่นิยมใช้งานอธิบายได้ดังนี้

2.3.4.1 ระบบควบคุมแบบพี

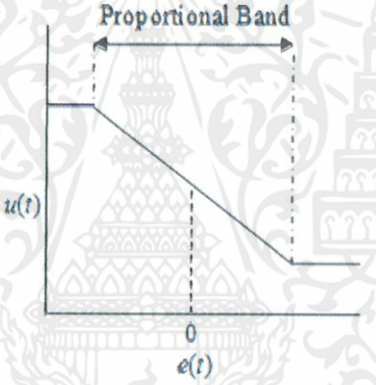
ในระบบควบคุมแบบพี เอาต์พุตของตัวควบคุม $u(t)$ จะเป็นสัดส่วนกับอินพุตของตัวควบคุม และถ้ากำหนดสัญญาณอินพุตของตัวควบคุมเป็นค่าความผิดพลาด $e(t)$ จะได้เอาต์พุตของตัวควบคุมดังสมการที่ (2.2)

(2.2)

เมื่อ K_p คือ อัตราขยายแบบสัดส่วน (Proportional gain) และมีฟังก์ชันถ่ายโอนของตัวควบคุมแบบพีเป็น

(2.3)

ดังนั้นการควบคุมด้วยตัวควบคุมแบบนี้จะเป็นเพียงการขยายสัญญาณความผิดพลาดเท่านั้น การที่ได้สัญญาณความผิดพลาดขนาดใหญ่ที่เวลาหนึ่ง ทำให้เกิดเอาต์พุตที่มีขนาดใหญ่จากตัวควบคุมในเวลานั้น อย่างไรก็ตาม การที่ให้อัตราขยายคงที่นั้น ในทางปฏิบัติอาจกำหนดไว้ในบางช่วงของสัญญาณความผิดพลาดเท่านั้น หรืออาจกำหนดให้ตัวควบคุมมีค่าเอาต์พุตไม่น้อยกว่าค่า ๆ หนึ่งและไม่มากเกินกว่าค่า ๆ หนึ่งก็ได้ ซึ่งการกำหนดช่วงจำกัดของเอาต์พุตจะมีลักษณะดังรูปที่ 2.16 ช่วงที่มีการกำหนดสัดส่วนนี้จะเรียกว่า Proportional Band

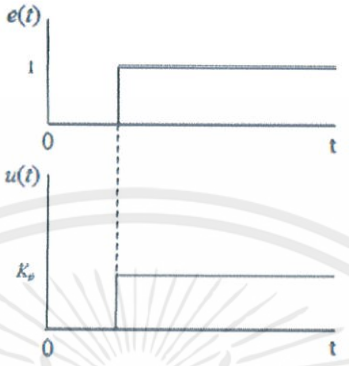


รูปที่ 2.16 การกำหนดช่วงจำกัดของเอาต์พุต

การกำหนด Proportional Band นี้จะช่วยให้สัญญาณเอาต์พุตมีค่าจำกัด ไม่ไปสู่ค่าอนันต์ทั้งทางด้านบวกและทางด้านลบ และเมื่อตัวควบคุมมีเอาต์พุตสูงสุดที่จะเป็นไปได้ค่าหนึ่ง ก็นิยมที่จะกำหนดเอาต์พุตค่าใด ๆ เป็นร้อยละของค่าสูงสุดที่จะเป็นไปได้ ดังนั้นการเปลี่ยนแปลงค่าเอาต์พุตของตัวควบคุม 100% หมายถึงเอาต์พุตจะเปลี่ยนค่าจากต่ำสุดที่เป็นไปได้ ไปสู่ค่าสูงสุดที่เป็นไปได้ ซึ่งจะได้ว่า

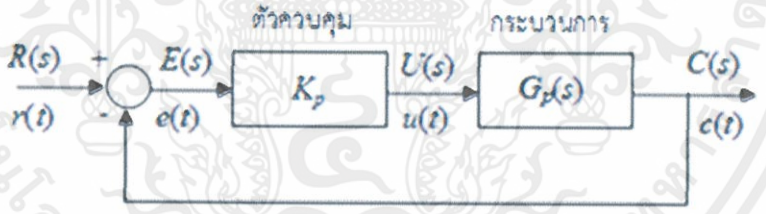
$$K_p = \frac{100}{Proportional\ Band} \tag{2.4}$$

เนื่องจากเอาต์พุตของตัวควบคุมจะเป็นสัดส่วนกับอินพุต ดังนั้นถ้าหากอินพุตมีลักษณะเป็นสัญญาณระดับ (Step signal) เอาต์พุตที่ได้ก็จะมีลักษณะเป็นสัญญาณระดับเช่นกัน โดยลักษณะของกราฟแสดงอินพุตและเอาต์พุตจะมีสัดส่วนที่แน่นอนค่าหนึ่งดังแสดงในรูปที่ 2.17 โดยรูปนี้แสดงถึงการตอบสนองของตัวควบคุม เมื่ออินพุตอยู่ในช่วง Proportional Band



รูปที่ 2.17 ผลการตอบสนองของตัวควบคุมแบบพี

ในทางปฏิบัติตัวควบคุมแบบพีนี้มีลักษณะเหมือนกับเครื่องขยายสัญญาณรูปแบบหนึ่งซึ่งอาจเป็นในลักษณะของอุปกรณ์ไฟฟ้า หรืออาจจะเป็นเครื่องขยายสัญญาณเชิงกล ลักษณะของระบบที่มีตัวควบคุมแบบพี แสดงได้ดังรูปที่ 2.18



รูปที่ 2.18 บล็อกไดอะแกรมของระบบควบคุมแบบพี

ข้อเสียของระบบที่มีตัวควบคุมแบบพีคือ ไม่เพิ่มชนิด (Type) ให้กับระบบซึ่งหมายความว่าถ้าระบบเป็นชนิด 0 (Type 0) ตัวควบคุมจะไม่เปลี่ยนแปลงชนิด ทำให้ระบบยังคงเป็นชนิดเดิมและไม่สามารถขจัดค่าความผิดพลาดในสภาวะคงตัว (Steady state error) ได้ แต่ก็สามารถทำให้ค่าความผิดพลาดดังกล่าวมีค่าน้อยลงได้ด้วยการปรับค่าเกนให้สูง ซึ่งในทางปฏิบัติแล้วการปรับค่าเกนให้สูงมากขนาดไหน เอาต์พุตที่ออกจริง ๆ จากตัวควบคุมมักมีค่าจำกัด และการปรับค่าเกนให้มีค่าสูงสำหรับระบบที่มีอันดับสูง อาจจะทำให้ได้ผลตอบสนองที่ไม่เป็นที่พึงประสงค์ เช่น การปรับค่าเกน

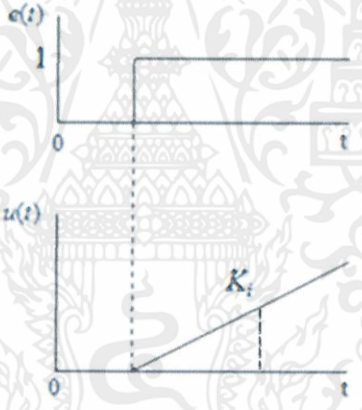
ให้สูงขึ้นสำหรับระบบอันดับสอง ผลที่ตามมาก็คือ ผลตอบสนองที่มีค่าพุ่งเกิน (Overshoot) ก็จะสูงขึ้นตามด้วย ซึ่งอาจจะเป็นอันตรายต่อระบบได้

2.3.4.2 ระบบควบคุมแบบไอ

ในระบบควบคุมแบบไอ เอาต์พุตของตัวควบคุมจะเป็นสัดส่วนกับการอินทิกรัลสัญญาณของค่าความผิดพลาดเทียบเวลา ดังสมการที่ (2.5)

$$u(t) = K_i \int_0^t e(t)\, dt \tag{2.5}$$

เมื่อ K_i คือ อัตราขยายแบบอินทิกรัล (Integral gain) ซึ่งจะมีหน่วยเป็น sec^{-1} รูปที่ 2.19 แสดงลักษณะการตอบสนองของตัวควบคุมแบบไอ เมื่อได้รับสัญญาณอินพุตแบบระดับ ค่า $u(t)$ ณ เวลา t ใด ๆ หมายถึงพื้นที่ใต้กราฟของสัญญาณอินพุต จากเวลา 0 ถึง t คูณด้วยอัตราขยายแบบอินทิกรัล ดังนั้น เมื่อเริ่มมีค่าอินพุตแบบระดับ เอาต์พุตที่ออกจากตัวควบคุมจึงมีค่ามากขึ้นเรื่อย ๆ ด้วยอัตราคงที่ ในลักษณะรูปสัญญาณความชัน (Ramp signal)

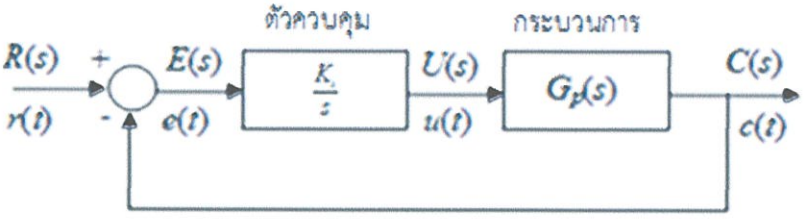


รูปที่ 2.19 ผลตอบสนองของตัวควบคุมแบบไอ

จากสมการที่ (2.5) จะได้ฟังก์ชันถ่ายโอนของตัวควบคุมแบบไอ เป็น

$$G_c(s) = \frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{2.6}$$

ลักษณะของระบบที่มีตัวควบคุมแบบไอ แสดงได้ดังรูปที่ 2.20



รูปที่ 2.20 บล็อกไดอะแกรมของระบบควบคุมแบบไอ

ทำให้มีฟังก์ชันถ่ายโอนของระบบเป็น $\frac{K_i}{s}G_p(s)$ และทำให้มีฟังก์ชันถ่ายโอนวงปิดเป็น

$$\frac{C(s)}{R(s)} = \frac{\frac{K_i}{s}G_p(s)}{1+\frac{K_i}{s}G_p(s)} \tag{2.7}$$

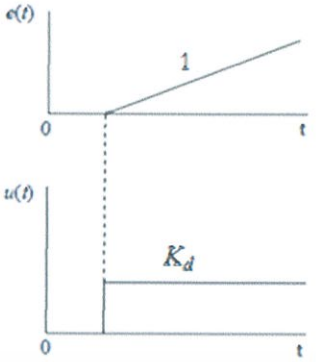
ดังนั้น สามารถพิจารณาถึงข้อได้เปรียบของการควบคุมแบบไอได้จากฟังก์ชันถ่ายโอน ซึ่งจะเห็นได้ว่าระบบควบคุมแบบไอ จะเพิ่มชนิดของระบบขึ้นมา 1 ระดับ ซึ่งทำให้ค่าความผิดพลาดที่สภาวะคงตัวเป็นศูนย์เทียบต่อสัญญาณอินพุตแบบระดับ อย่างไรก็ตามการเพิ่มโพลที่ $s=0$ ในระบบวงเปิดและไม่มีการเพิ่มซีโร่ให้ระบบควบคุม ทำให้ความแตกต่างระหว่างโพล (n) และซีโร่ (m) เพิ่มขึ้นอีก 1 มีผลให้มุมของอะซิมโทต (Asymptote angle) ของเส้นทางรากลดลง และจุดตัด (Centroid) เคลื่อนไปทางขวาของระนาบ s (s-plane) มากขึ้น ทำให้เสถียรภาพของระบบลดลง

2.3.4.3 ระบบควบคุมแบบดี

ระบบควบคุมแบบดีนี้ เอาต์พุตของตัวควบคุมจะเป็นสัดส่วนกับอัตราการเปลี่ยนแปลงอินพุตความผิดพลาดเทียบต่อเวลา หรือ

$$u(t) = K_d \frac{de(t)}{dt} \tag{2.8}$$

เมื่อ K_d คือ อัตราขยายแบบอนุพันธ์ (Derivative gain)



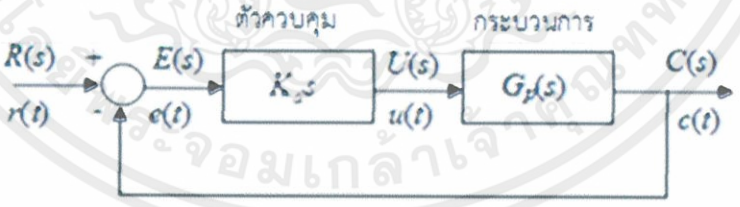
รูปที่ 2.21 ผลตอบสนองของตัวควบคุมแบบดี

รูปที่ 2.21 แสดงผลตอบสนองของระบบควบคุมแบบดี เมื่อสัญญาณอินพุตเป็นสัญญาณแบบความชัน จากสมการที่ (2.8) จะเห็นว่า ตัวควบคุมแบบดีจะให้สัญญาณออกจากตัวควบคุมมีค่ามาก เมื่อค่าอินพุตความผิดพลาดมีแนวโน้มเพิ่มขึ้นในอัตราที่สูงก่อนที่ความผิดพลาดจะเกิดขึ้นจริง ๆ

อย่างไรก็ตาม หากความผิดพลาดที่มีค่าคงที่ก็จะไม่มีการสะสมค่าความผิดพลาด แม้ว่าค่าความผิดพลาดจะมีมากก็ตาม ทำให้การควบคุมแบบดีนี้ ไม่เปลี่ยนแปลงต่อค่าความผิดพลาดที่คงที่ หรือมีการเปลี่ยนแปลงอย่างช้า ๆ ดังนั้นการควบคุมแบบนี้จึงไม่นิยมใช้ตามลำพัง แต่มักใช้ร่วมกับการควบคุมแบบอื่น โดยที่ฟังก์ชันถ่ายโอนตัวควบคุมแบบดี จะเป็น

$$G_c(s) = \frac{U(s)}{E(s)} = K_d s \qquad (2.9)$$

ลักษณะของระบบที่มีตัวควบคุมแบบดี แสดงได้ดังรูปที่ 2.22



รูปที่ 2.22 บล็อกไดอะแกรมของระบบควบคุมแบบดี

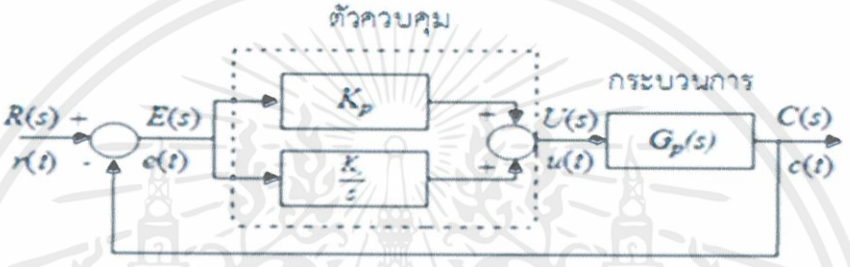
ถ้าหากว่าระบบเป็นแบบชนิด 1 หรือสูงกว่า การควบคุมแบบดีจะลดเทอม s ของพหุนาม ในส่วนของฟังก์ชันถ่ายโอนของฟอร์เวิร์ดพาร์ทลง ซึ่งจะไปลดชนิดของระบบลง 1 อย่างไรก็ตาม ดังที่ได้

กล่าวไว้ก่อนหน้านี้แล้วว่า การควบคุมแบบดีนี้จะไม่ใช้เพียงลำพังแต่จะใช้ร่วมกับตัวควบคุมแบบอื่น เพราะเมื่อใช้การควบคุมแบบดี ทำให้เพิ่มความเร็วในการตอบสนองของระบบต่อความผิดพลาดอื่น

ในทางปฏิบัติการนำการควบคุมดีไปใช้นั่นค่อนข้างจะลำบาก ดังนั้น ในทางปฏิบัติทั่วไปจะเป็นการประมาณการควบคุมดี โดยใช้ตัวชดเชยแบบมุมนำ (Lead compensator)

2.3.4.4 ระบบควบคุมแบบพีไอ

การที่ระบบควบคุมมีความเสถียรสัมพัทธ์ลดลง เมื่อใช้การควบคุมแบบไอ สามารถที่จะแก้ไขได้ในระดับหนึ่งโดยการใช้การควบคุมแบบพีร่วมกับไอ (PI: Proportional plus Integral) หรือเรียกว่าการควบคุมแบบพีไอ ซึ่งลักษณะของระบบควบคุมเป็นไปตามรูปที่ 2.23

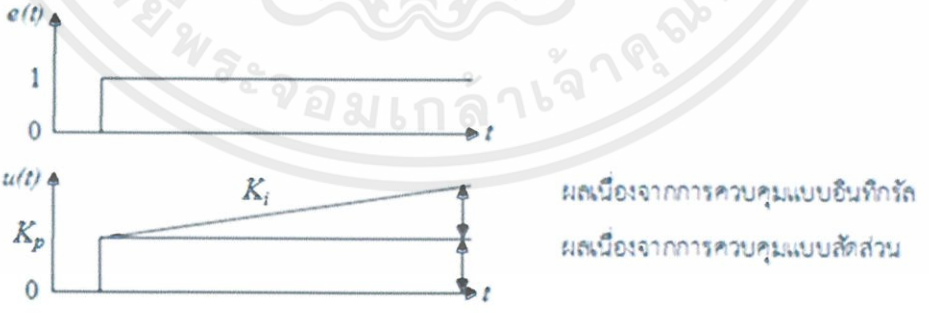


รูปที่ 2.23 บล็อกไดอะแกรมของระบบควบคุมแบบพีไอ

สำหรับระบบควบคุมดังกล่าวมีเอาต์พุตของตัวควบคุมเป็น

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\ \mathrm{dt} \qquad (2.10)$$

เอาต์พุตของตัวควบคุมแบบพีไอที่ได้รับเมื่อมีสัญญาณอินพุตเป็นแบบระดับแสดงได้ดังรูปที่ 2.24



รูปที่ 2.24 ผลตอบสนองของตัวควบคุมแบบพีไอ

จากสมการที่ (2.10) ฟังก์ชันถ่ายโอนของตัวควบคุมแบบพีไอ เป็น

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s}$$
$$= K_p \frac{(s + \frac{K_i}{K_p})}{s} \quad (2.11)$$

นิยามค่าเวลาคงตัวอินทิกรัล (Integral time constant) T_i เป็น

$$T_i = \frac{K_p}{K_i} \quad (2.12)$$

ดังนั้นจะได้

$$G_c(s) = \frac{K_p[s + (\frac{1}{T_i})]}{s} \quad (2.13)$$

และจะทำให้ฟังก์ชันถ่ายโอนระบบวงเปิดเป็น

$$G_0(s) = G_c(s)G_p(s)$$
$$= \frac{K_p[s + (\frac{1}{T_i})G_p(s)]}{s} \quad (2.14)$$

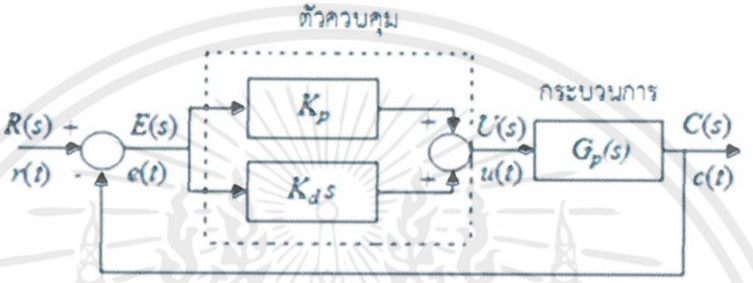
จะเห็นได้ว่ามีซีโร่ที่ $s = -\frac{1}{T_i}$ และโพลที่ $s = 0$ ซึ่งเป็นการเพิ่มโพลให้ฟังก์ชันถ่ายโอนวงเปิดของระบบเมื่อใช้การควบคุมแบบพีไอ การเพิ่มตัวประกอบ s เข้ากับพหุนามของฟังก์ชันถ่ายโอนวงเปิดเป็นการเพิ่มชนิดระบบขึ้นไป 1 จึงทำให้ระบบควบคุมแบบพีไอนี้ไม่มีความผิดพลาดที่สภาวะคงตัว นอกจาก นั้นการเพิ่มซีโร่ให้ระบบไปพร้อมๆ กันก็จะทำให้ความแตกต่างระหว่างโพล (n) กับซีโร่ (m) มีค่าคงที่ ดังนั้น มุมของเส้นอะซิมโทดสำหรับเส้นทางรากมีค่าคงเดิม แต่จุดตัดของเส้นอะซิมโทดบนแกนจริง จะเคลื่อนเข้าหาจุดกำเนิดมากขึ้นเนื่องจาก

จุดตัดของเส้นอะซิมโทดบนแกนจริง = (ผลรวมของโพล – ผลรวมของซีโร่) / *(n-m)*

ดังนั้นจุดตัดของเส้นอะซิมโทตจะเปลี่ยนไปเท่ากับ $-\frac{1}{T_i}/(n-m)$ ส่งผลให้ความเสถียรภาพของระบบลดลง อย่างไรก็ตาม การลดลงของความเสถียรสัมพัทธ์นี้จะน้อยกว่าการที่ใช้การควบคุมแบบไอเพียงอย่างเดียว

2.3.4.5 ระบบควบคุมแบบพีดี

การควบคุมแบบพีใช้ร่วมกับการควบคุมแบบดี (PD: Proportional plus Derivative Control) ดังที่แสดงในรูปที่ 2.25 จะมีฟังก์ชันถ่ายโอนระบบวงเปิดเป็น



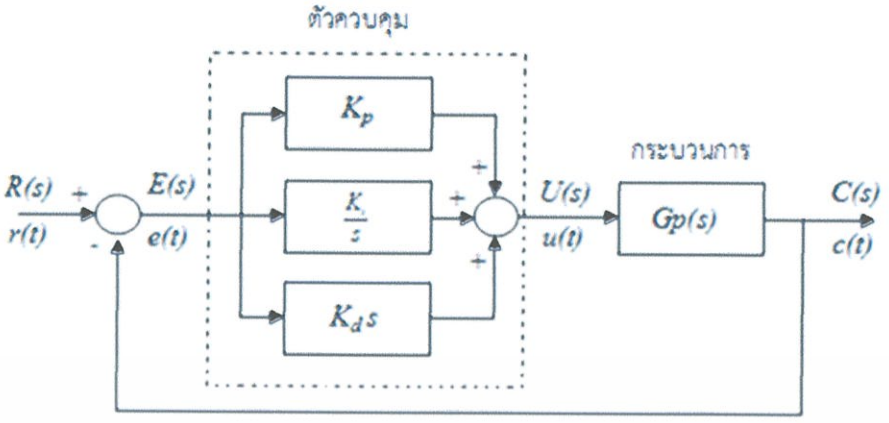
รูปที่ 2.25 บล็อกไดอะแกรมของระบบควบคุมแบบพีดี

$$G_0(s) = (K_p + K_d s)G_p(s) \qquad (2.15)$$
$$G_0(s) = K_p(1 + T_d s)G_p(s)$$

เมื่อ $T_d = \frac{K_d}{K_p}$ คือ ค่าเวลาคงตัวอนุพันธ์ (Derivative time constant) ซึ่งในการควบคุมแบบนี้ จะมีซีโร่เพิ่มขึ้นที่ $s = -T_d$ และจะเห็นว่าไม่มีการเปลี่ยนแปลงชนิดของระบบ ทำให้ไม่มีการเปลี่ยนแปลงค่าความผิดพลาดที่สภาวะคงตัว

2.3.4.6 ระบบควบคุมแบบพีไอดี

การควบคุมแบบพีไอดี (PID: Proportional plus Integral plus Derivative Control) อาศัยตัวควบคุมแบบพีร่วมกับแบบไอร่วมกับแบบดี ซึ่งอาจเรียกว่า การควบคุมแบบ 3 เทอม (Three-term control) โดยระบบควบคุมมีลักษณะดังรูปที่ 2.26



รูปที่ 2.26 บล็อกไดอะแกรมระบบควบคุมแบบพีไอดี

จะได้เอาต์พุตของตัวควบคุมเมื่อรับอินพุตเป็นความผิดพลาด $e(t)$ ดังสมการที่ (2.16)

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,\mathrm{dt} + K_d \frac{de(t)}{dt} \tag{2.16}$$

ซึ่งมีฟังก์ชันถ่ายโอนของตัวควบคุมแบบพีไอดี เป็น

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

$$G_c(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{2.17}$$

และมีฟังก์ชันถ่ายโอนระบบวงเปิดของระบบควบคุมพีไอดี เป็น

$$G_0(s) = G_c(s)G_p(s) = K_p(1 + \frac{1}{T_i s} + T_d s)G_p(s)$$

$$G_0(s) = \frac{K_p(T_i T_d s^2 + T_i s + 1)}{T_i s} G_p(s) \tag{2.18}$$

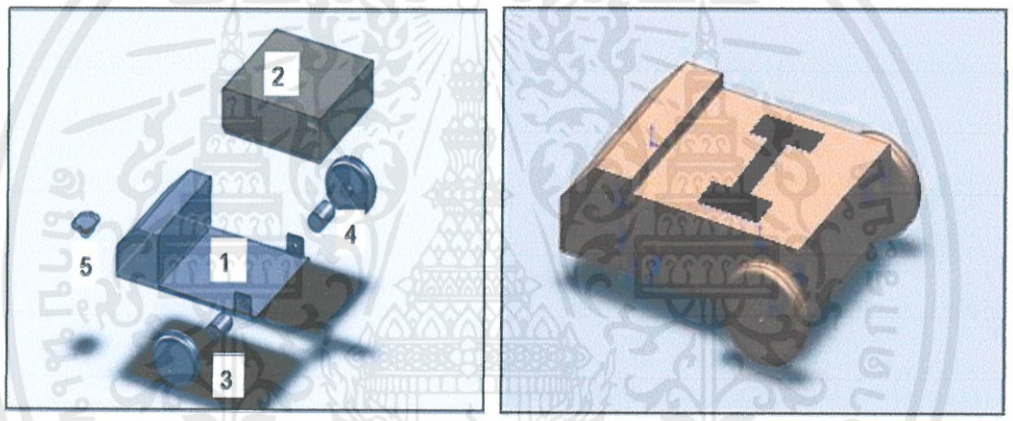
จากสมการที่ (2.18) เห็นว่าการควบคุมแบบพีไอดี จะเพิ่มซีโร่ให้กับระบบ 2 ตัว และเพิ่มโพล 1 ตัว โดยโพลที่เพิ่มนั้นอยู่ที่จุดกำเนิด ทำให้ชนิดของระบบเพิ่มขึ้น 1 ซึ่งทำให้ความผิดพลาดที่สภาวะคงตัวต่อสัญญาณขั้นบันไดมีค่าเป็นศูนย์

บทที่ 3
การออกแบบ

บทนี้กล่าวถึงการออกแบบโปรแกรมและอุปกรณ์ภายในโครงงานทั้งหมด ประกอบด้วย 3 ส่วน คือ ส่วนการออกแบบรถลำเลียงวัตถุดิบ ส่วนการออกแบบวงจร และส่วนการออกแบบโปรแกรม รายละเอียดต่าง ๆ สามารถอธิบายได้ดังนี้

3.1 การออกแบบรถลำเลียงวัตถุดิบ

ในโครงงานนี้ ใช้โปรแกรมโซลิดเวิร์ค (Solidworks) ในการออกแบบรถลำเลียงวัตถุดิบ โดยโปรแกรมโซลิดเวิร์ค เป็นโปรแกรมเขียนแบบและออกแบบที่ถูกพัฒนาขึ้นมาเพื่อนำไปใช้งานออกแบบผลิตภัณฑ์ เฟอร์นิเจอร์ และออกแบบชิ้นส่วนเครื่องกล 3 มิติ ซึ่งรถลำเลียงวัตถุดิบที่ใช้ในโครงงานนี้มีจำนวน 2 คัน โดยแบบรถลำเลียงวัตถุดิบที่ใช้ในโครงงานนี้สามารถแสดงได้ดังรูปที่ 3.1 โดยรูปที่ 3.1 (ก) เป็นแบบรถลำเลียงเมื่อมองแบบแยกส่วน และรูปที่ 3.1(ข) เป็นแบบรถลำเลียงเมื่อประกอบสำเร็จ

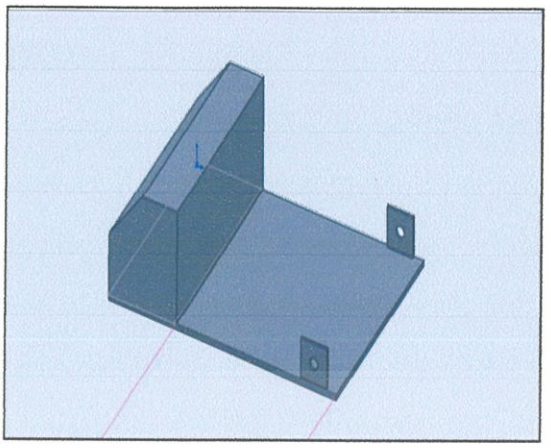


(ก) แบบรถเมื่อมองแยกส่วน (ข) แบบรถเมื่อประกอบสำเร็จ

รูปที่ 3.1 แบบรถลำเลียงวัตถุดิบ

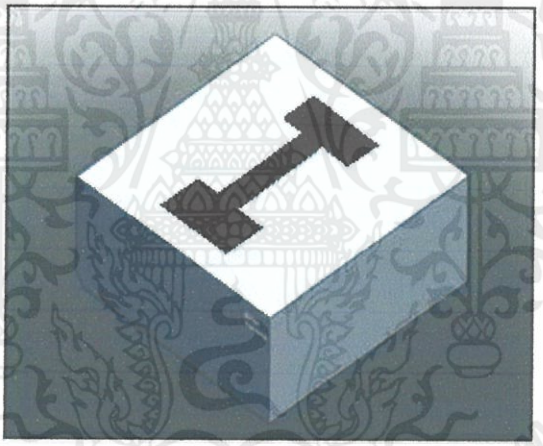
ส่วนประกอบของรถลำเลียงวัตถุดิบมี 5 ส่วน ได้แก่ โครงสร้างหลัก กระบะ มอเตอร์ ล้อหลัก และล้ออิสระ โดยแต่ละส่วนประกอบสามารถอธิบายได้ดังนี้

1. โครงสร้างหลักของรถลำเลียงวัตถุดิบ ดังรูปที่ 3.2 มีรายละเอียดดังนี้
   - ความยาวของฐานมีขนาดกว้าง 250 มิลลิเมตร ยาว 350 มิลลิเมตร
   - ขนาดของส่วนหัวของรถมีขนาดความกว้าง 100 มิลลิเมตร ยาว 250 มิลลิเมตร สูง 110 มิลลิเมตร จากนั้น ตัดมุมข้างหน้าให้ทำมุม 45 องศา จะได้ส่วนของหน้ารถตามต้องการ
   - หน้าแปลนยึดมอเตอร์ มีขนาดกว้าง 40 มิลลิเมตร ยาว 40 มิลลิเมตร เจาะรูตรงกลางของหน้าแปลนขนาด 60 มิลลิเมตร เพื่อใส่แกนมอเตอร์



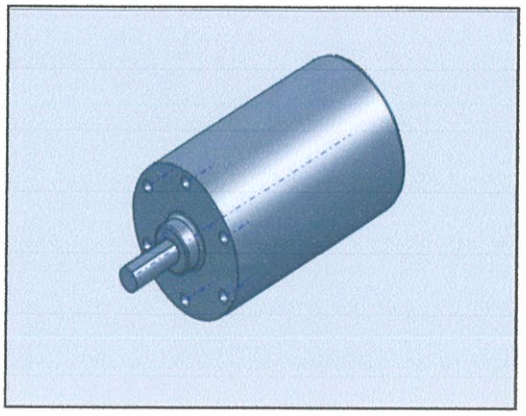
รูปที่ 3.2 โครงสร้างหลักของรถลำเลียงวัตถุดิบ

2. *กระบะ ดังรูปที่ 3.3 มีรายละเอียดดังนี้*
   - มีขนาดกว้าง 250 มิลลิเมตร ยาว 240 มิลลิเมตร และสูง 90 มิลลิเมตร
   - มีช่องว่างสำหรับติดตั้งกลไกสายพานเพื่อขนถ่ายวัตถุดิบไปส่งตามจุดต่าง ๆ (สำหรับรถลำเลียงคันที่ 2 จะไม่มีช่องว่างสำหรับติดตั้งกลไกสายพาน เป็นเพียงการจำลองการทำงานเท่านั้น)



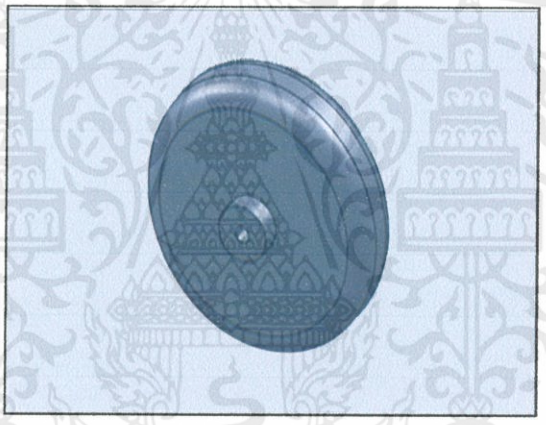
รูปที่ 3.3 กระบะของรถลำเลียงวัตถุดิบ

3. มอเตอร์ เป็นแบบตามมอเตอร์ซึ่งจัดเตรียมไว้แล้ว แสดงดังรูปที่ 3.4 มีรายละเอียดดังนี้
   - มอเตอร์กระแสตรงขนาด 12 โวลต์
   - เส้นผ่านศูนย์กลางของมอเตอร์มีขนาด 37 มิลลิเมตร ยาว 58 มิลลิเมตร
   - แกนมอเตอร์มีขนาดเส้นผ่านศูนย์กลาง 6 มิลลิเมตร ยาว 15 มิลลิเมตร
   - รูสำหรับใส่น็อตยึดหน้าแปลนจำนวน 6 รู มีขนาดเส้นผ่านศูนย์กลาง 3 มิลลิเมตร



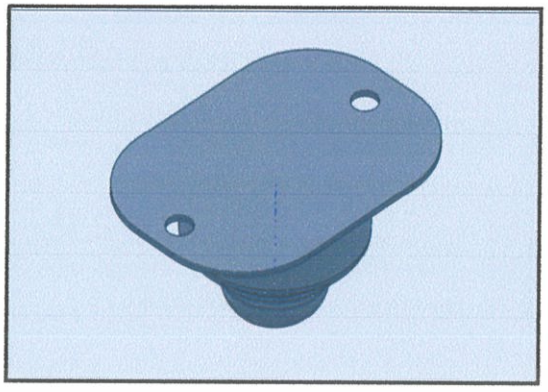
รูปที่ 3.4 มอเตอร์ไฟฟ้ากระแสตรง

4. ล้อหลัก เป็นแบบตามล้อสำเร็จลักษณะดังรูปที่ 3.5 มีรายละเอียดดังนี้
   - เส้นผ่านศูนย์กลางของวงล้อมีขนาด 120 มิลลิเมตร หนา 30 มิลลิเมตร
   - ช่องสำหรับสวมแกนมอเตอร์ขนาดเส้นผ่านศูนย์กลาง 7 มิลลิเมตร



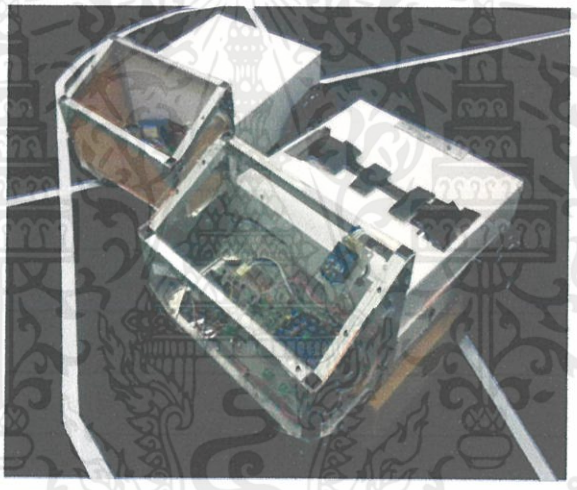
รูปที่ 3.5 ล้อหลักของรถลำเลียงวัตถุดิบ

5. ล้ออิสระ เป็นแบบตามล้อสำเร็จลักษณะดังรูปที่ 3.6 มีรายละเอียดดังนี้
   - แปลนยึด มีขนาดกว้าง 50 มิลลิเมตร ยาว 70 มิลลิเมตร และช่องใส่น็อต 2 ช่องขนาด 6 มิลลิเมตร
   - ลูกเหล็ก อยู่ในเบ้าขนาดเส้นผ่านศูนย์กลาง 14 มิลลิเมตร



รูปที่ 3.6 ล้ออิสระ

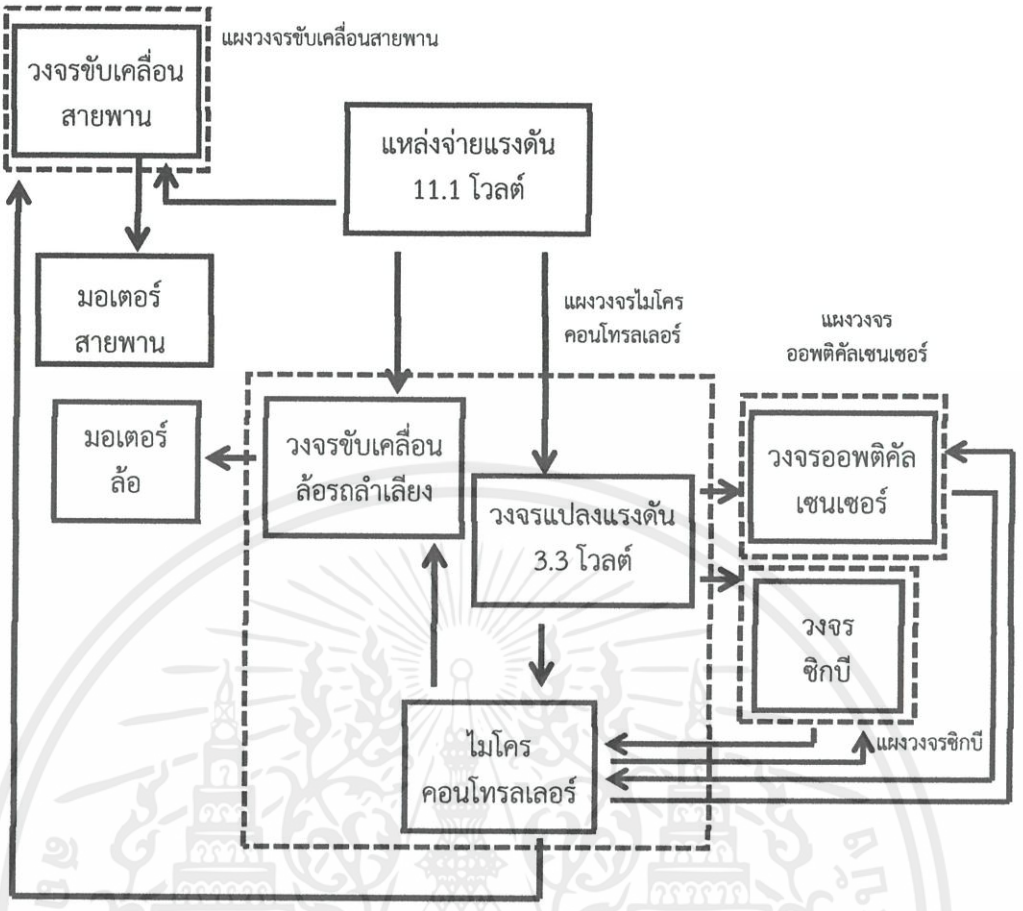
เมื่อทำการออกแบบรถลำเลียงเสร็จแล้ว จึงนำแบบไปสร้างขึ้นตามส่วนประกอบต่าง ๆ เพื่อประกอบรวมเป็นรถลำเลียงสำหรับนำมาใช้ลำเลียงวัตถุดิบ ซึ่งรถลำเลียงที่สร้างขึ้นสำเร็จแล้ว สามารถแสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 รถลำเลียงวัตถุดิบที่สร้างขึ้นเสร็จสมบูรณ์

3.2 การออกแบบวงจร

แผงวงจรที่ใช้ภายในโครงงานนี้มี 4 วงจร ได้แก่ แผงวงจรไมโครคอนโทรลเลอร์ แผงวงจรขับเคลื่อนสายพาน แผงวงจรออพติคัลเซนเซอร์ และแผงวงจรชิกบี สามารถอธิบายแผนภาพรวมของวงจรทั้งหมดได้ดังรูปที่ 3.8



รูปที่ 3.8 แผนภาพรวมของวงจรในรถลำเลียง

3.2.1 แผงวงจรไมโครคอนโทรลเลอร์

ในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์ของบริษัท อาร์ม จำกัด (ARM Ltd.) ซึ่งใช้บอร์ดสำเร็จรุ่น ET-ARM STAMP STAM32 ประกอบด้วย ARM STM32F103RET6 โดยภายในแผงวงจรไมโครคอนโทรลเลอร์ประกอบด้วย 6 ส่วน คือ

1. วงจรแปลงแรงดันไฟฟ้าจาก 11.1 โวลต์ให้เป็น 3.3 โวลต์ โดยใช้ไอซี LM1117T และคาปาซิสเตอร์ประเภทเทนทาลัม (Tantalum) 10 ไมโครฟารัด จำนวน 2 ตัว ต่อ 1 ชุด เป็นอุปกรณ์ในการแปลงแรงดัน โดยในโครงงานนี้ใช้วงจรแปลงแรงดัน 2 ชุด เนื่องจากมีโหลดจำนวนมาก หากใช้วงจรแปลงแรงดันเพียง 1 ชุดจะทำให้วงจรแปลงแรงดันทำงานหนักจนเกิดความเสียหายได้ นอกจากนั้นยังมีหลอดไฟแอลอีดีในการแสดงสถานะการทำงานของวงจรแปลงแรงดันไฟฟ้าอีกด้วย รวมถึงสวิตซ์เปิด – ปิด ที่ให้ผู้ใช้สามารถเปิดและปิดวงจรได้

2. วงจรขับเคลื่อนล้อรถลำเลียง ในโครงงานนี้เลือกใช้ไอซี L293D ซึ่งเป็นไอซีขับมอเตอร์โดยสัญญาณพีดับบลิวเอ็ม

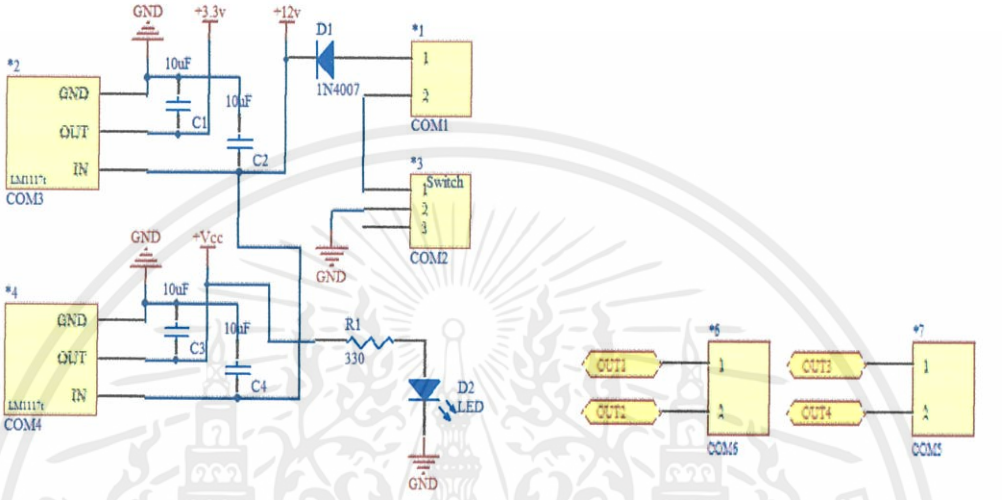
3. ส่วนการรองรับการเชื่อมต่อจากแผงวรจรออพติคัลเซนเซอร์

4. ส่วนการรองรับการเชื่อมต่อจากแผงวงจรซิกบี

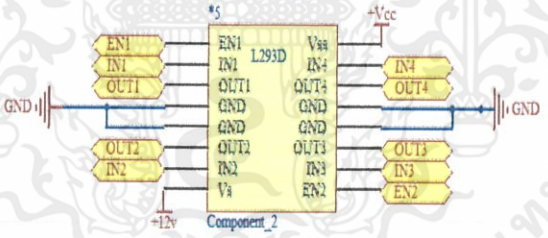
5. ส่วนการตรวจสอบระดับแรงดันไฟจากแหล่งจ่าย จะใช้ทรานซิสเตอร์เบอร์ BC547 จำนวน 2 ตัว ต่อตามวงจรในรูปที่ 3.9 และมีหลอดไฟแอลอีดีแสดงสถานะการแจ้งเตือนระดับแรงดันที่ต่ำกว่าการใช้งานที่เหมาะสม

6. ส่วนการรองรับการเชื่อมต่อจากแผงวงจรขับเคลื่อนสายพาน

ซึ่งวงจรไมโครคอนโทรลเลอร์สามารถแสดงได้ดังรูปที่ 3.9

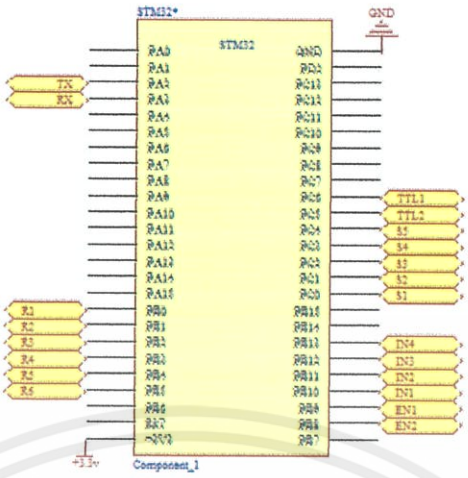


(ก) วงจรแปลงแรงดันไฟฟ้า (ข) ส่วนเชื่อมต่อระหว่างไอซี L293D กับมอเตอร์



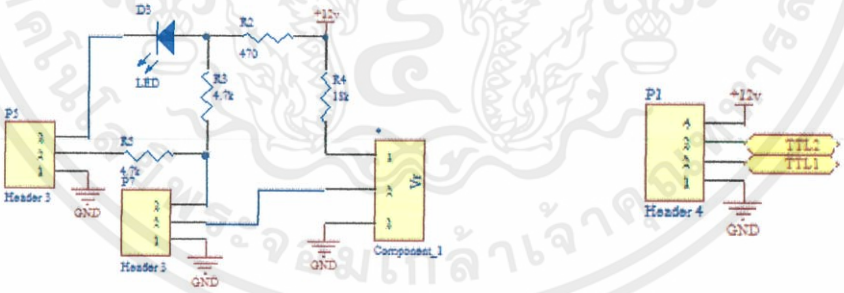
(ค) วงจรขับเคลื่อนล้อรถลำเลียง

รูปที่ 3.9 แผงวงจรไมโครคอนโทรลเลอร์



(ง) ไมโครคอนโทรลเลอร์

(จ) ส่วนรองรับการเชื่อมต่อจากซิกบี (ฉ) ส่วนรองรับการเชื่อมต่อจากแผงวงจรเซนเซอร์

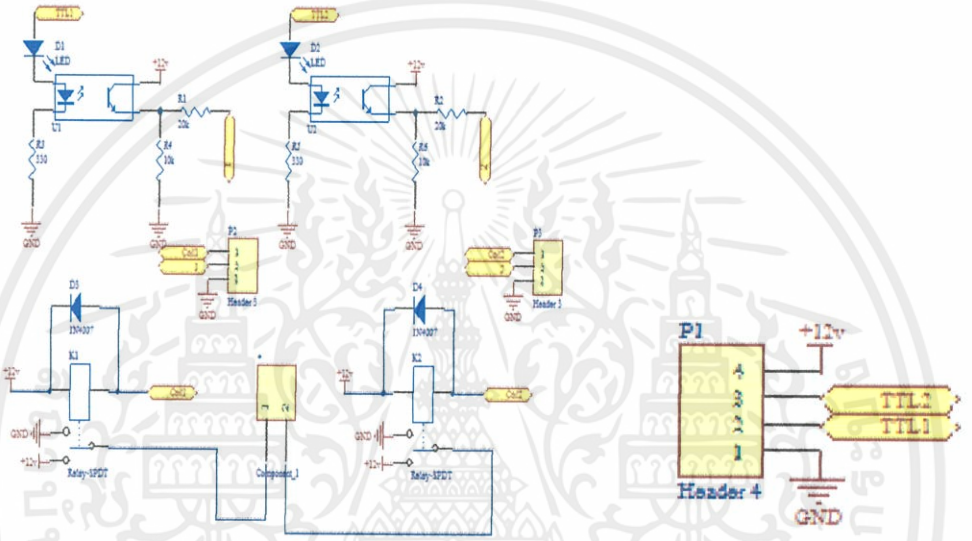


(ช) ส่วนการตรวจสอบระดับแรงดันจากแหล่งจ่าย (ซ) ส่วนรองรับการเชื่อมต่อกับแผงวงจรขับเคลื่อนสายพาน

รูปที่ 3.9 แผงวงจรไมโครคอนโทรลเลอร์ (ต่อ)

3.2.2 แผงวงจรขับเคลื่อนสายพาน

แผงวงจรขับเคลื่อนสายพาน นำไปใช้ควบคุมมอเตอร์ไฟฟ้ากระแสตรงในรถลำเลียงวัตถุดิบ โดยมีหลักการทำงานดังนี้ อุปกรณ์เชื่อมต่อทางแสง (Opto isolator) ในโครงงานนี้เลือกใช้เบอร์ PC814 จะทำการรับสัญญาณไฟฟ้าทีทีแอลจากไมโครคอนโทรเลอร์ซึ่งมีแรงดันตั้งแต่ 0-3.3 โวลต์ แล้วออพโต้ซึ่งทำหน้าที่เป็นตัวแยกสัญญาณกราวด์ของวงจรจะทำการเปิด-ปิดไฟเลี้ยง 0-12 โวลต์ ที่จ่ายให้กับรีเลย์ 12 โวลต์ โดยมีทรานซิสเตอร์ซึ่งเลือกใช้เบอร์ BC547 ช่วยในการขับกระแสไปเลี้ยงที่รีเลย์ เหตุผลที่ไม่ใช้การขับโดยตรงจากทรานซิสเตอร์เป็นเพราะการขับลักษณะนั้นจะมีสัญญาณรบกวนมาสู่ไมโครคอนโทรลเลอร์ได้ ซึ่งวงจรขับเคลื่อนสายพานแสดงได้ดังรูปที่ 3.10



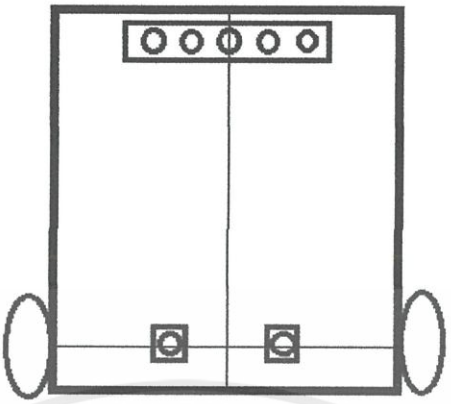
(ก) วงจรขับเคลื่อนสายพาน

(ข) ส่วนเชื่อมต่อกับแผงวงจรไมโครคอนโทรลเลอร์

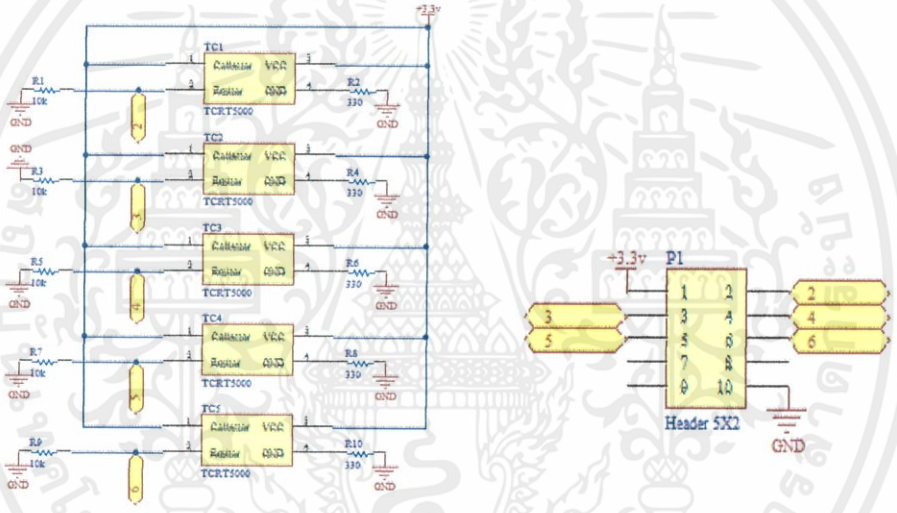
รูปที่ 3.10 แผงวงจรขับเคลื่อนสายพาน

3.2.3 แผงวงจรออพติคัลเซนเซอร์

ออพติคัลเซนเซอร์ที่ใช้คือ TCRT5000 เป็นออพติคัลสวิตช์แบบสะท้อนแสง ฝั่งเอาต์พุตมีลักษณะแบบโฟโต้ทรานซิสเตอร์ ซึ่งวงจรออพติคัลเซนเซอร์ที่ใช้ในโครงงานนี้ ใช้ออพติคัลเซนเซอร์จำนวน 5 ตัว เพื่อตรวจจับการเบี่ยงเบนของรถลำเลียงจากเส้นทางเดินได้ และมีเซนเซอร์อีก 2 ตัว ที่ติดอยู่ใกล้บริเวณล้อเพื่อช่วยในการเลี้ยวโค้งหักศอกได้อย่างแม่นยำขึ้น ดังรูปที่ 3.11 และแผงวงจรออพติคัลเซนเซอร์สามารถแสดงได้ดังรูปที่ 3.12



รูปที่ 3.11 ตำแหน่งการติดตั้งเซนเซอร์

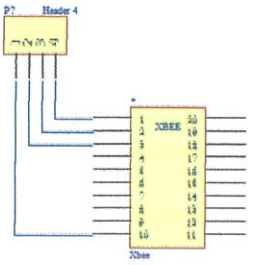


(ก) วงจรออพติคัลเซนเซอร์ (ข) ส่วนเชื่อมต่อกับแผงวงจรไมโครคอนโทรลเลอร์

รูปที่ 3.12 แผงวงจรออพติคัลเซนเซอร์

3.2.4 วงจรซิกบี

วงจรซิกบีนำมาใช้นั้นใช้รุ่น XB24-AWI-001 ซึ่งเป็น ZigBee รุ่นที่ 1 และเป็นโมดูลรับ-ส่งสัญญาณไร้สาย ย่านความถี่ 2.4 กิกกะเฮิร์ทซ์ ซึ่งเป็นย่านความถี่วิทยุ เป็นตามมาตรฐานโปรโตคอล 802.15.4 ใช้พลังงานต่ำ มีสายอากาศแบบ whip antenna เพื่อการสื่อสารระหว่างรถลำเลียงวัตถุดิบกับเครื่องคอมพิวเตอร์ และรถลำเลียงอีกหนึ่งคัน เพื่อเป็นการบอกสถานะล่าสุดส่งไปให้กับโปรแกรมที่พัฒนาขึ้น วงจรซิกบีจะเชื่อมต่อกับไมโครคอนโทรลเลอร์ผ่านขา Tx, Rx ดังแสดงได้ดังรูปที่ 3.13



(ก) โมดูลซิกบี (ข) ส่วนเชื่อมต่อกับแผงวงจรไมโครคอนโทรลเลอร์

รูปที่ 3.13 วงจรซิกบี

3.3 การออกแบบโปรแกรม

โปรแกรมที่ใช้ในโครงงานนี้ประกอบด้วยโปรแกรม 2 ส่วน ได้แก่ โปรแกรมส่วนคอมพิวเตอร์ และโปรแกรมส่วนไมโครคอนโทรลเลอร์ ซึ่งรายละเอียดของโปรแกรมแต่ละส่วน อธิบายได้ดังนี้

3.3.1 โปรแกรมส่วนคอมพิวเตอร์

ในโครงงานนี้อาศัยโปรแกรมวิชวลซีชาร์ป 2010 พัฒนาโปรแกรมเชื่อมต่อกับผู้ใช้งานเพื่อใช้สั่งการควบคุมการทำงาน การแสดงผล และการบันทึกผล มีแนวคิดในการพัฒนาโดยอาศัยแผนผังการทำงานดังรูปที่ 3.14 โดยโปรแกรมเชื่อมต่อกับผู้ใช้งานที่พัฒนาขึ้น ประกอบด้วยหน้าต่างสำคัญต่าง ๆ ได้แก่

- หน้าต่างตั้งค่าการเชื่อมต่อกับอุปกรณ์ภายนอกและการสร้างชื่อผู้ใช้ใหม่

  เป็นหน้าต่างสำหรับตั้งค่าอุปกรณ์ภายนอกให้ตรงกับคอมพิวเตอร์ก่อนเข้าสู่การทำงาน รวมทั้งการสร้างชื่อผู้ใช้ใหม่สำหรับบุคคลอื่น

- หน้าต่างล็อกอินเข้าสู่ระบบ

  เป็นหน้าต่างสำหรับป้อนชื่อผู้ใช้งานและรหัสผ่านเพื่อเป็นการป้องกันในเบื้องต้น

- หน้าต่างสั่งการทำงานของระบบ

  เป็นหน้าต่างสำหรับเลือกค่าน้ำหนัก จำนวนชุดวัตถุดิบ และรถลำเลียงที่ต้องการ

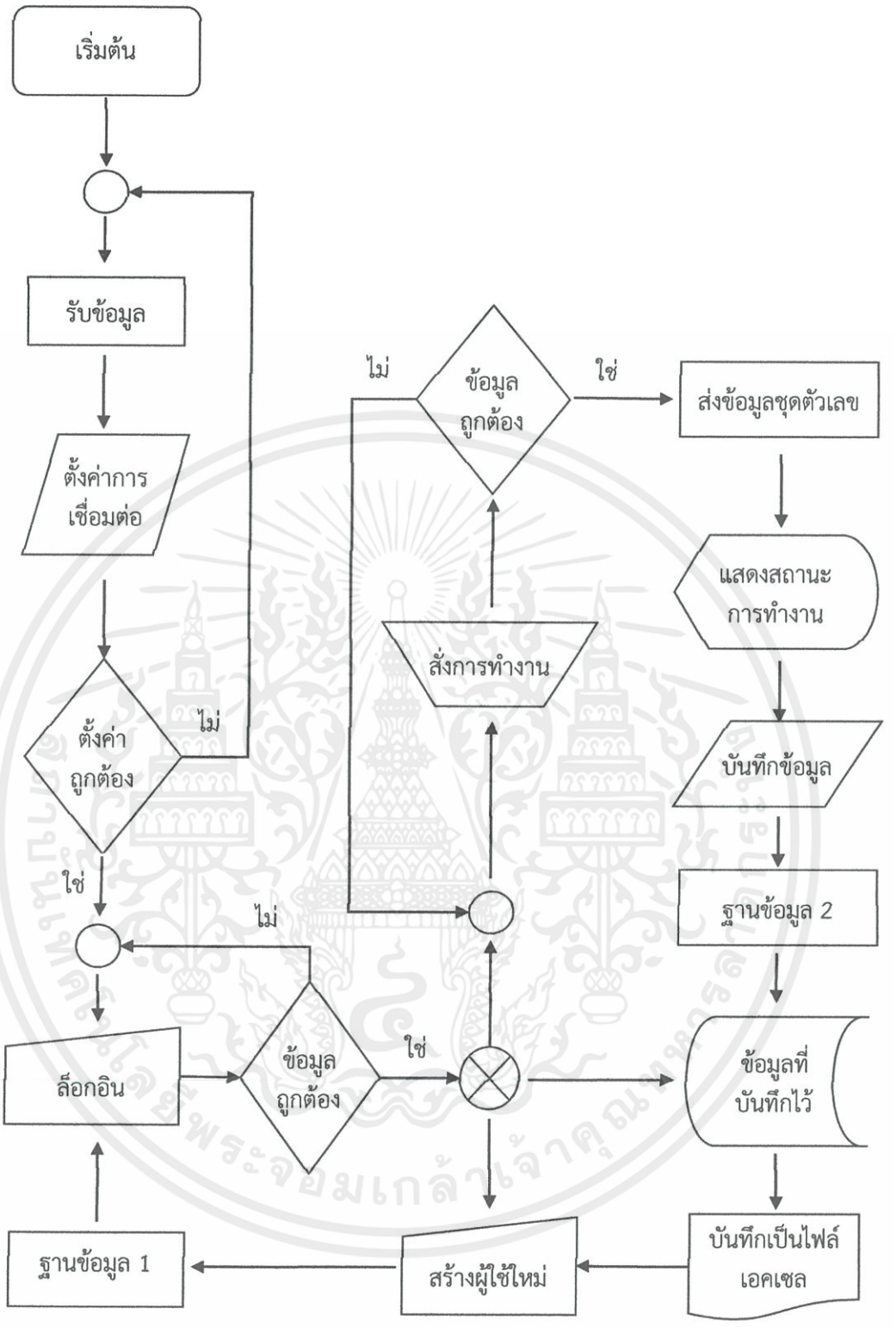
- หน้าต่างแสดงสถานะการทำงาน

  เป็นหน้าต่างสำหรับแสดงสถานะของทั้งรถลำเลียงและไซโล

- หน้าต่างบันทึกผลการทำงาน

  เป็นหน้าต่างสำหรับแสดงการบันทึกการทำงานของข้อมูลที่ได้เลือกไว้ รวมทั้งสามารถบันทึกข้อมูลให้เป็นไฟล์ของไมโครซอฟท์เอคเซล

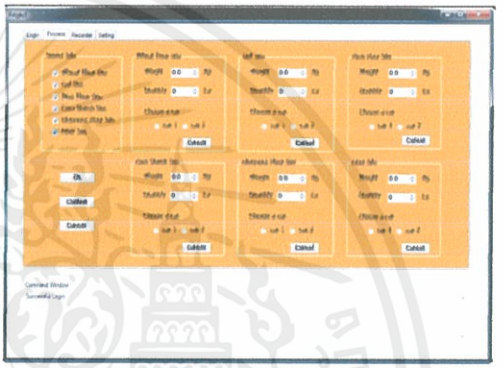
ลักษณะของโปรแกรมเชื่อมต่อกับผู้ใช้งานทั้งหมด สามารถแสดงได้ดังรูปที่ 3.15



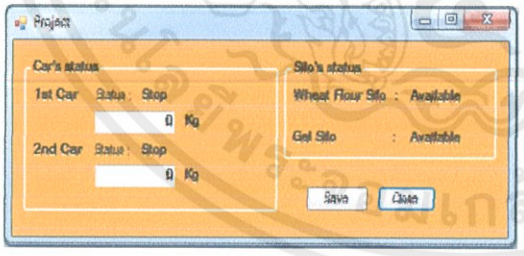
รูปที่ 3.14 แผนผังการทำงานของโปรแกรมที่พัฒนาขึ้น

(ก) หน้าต่างตั้งค่าการเชื่อมต่อกับอุปกรณ์ภายนอกและสร้างชื่อผู้ใช้ใหม่

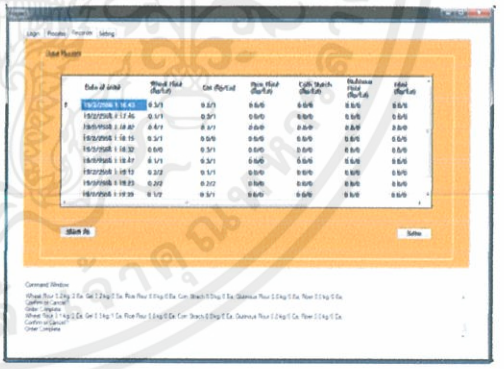
(ข) หน้าต่างล็อกอินเข้าสู่ระบบ



(ค) หน้าต่างสั่งการทำงานของระบบ



(ง) หน้าต่างแสดงสถานะการทำงาน



(จ) หน้าต่างบันทึกผลการทำงาน

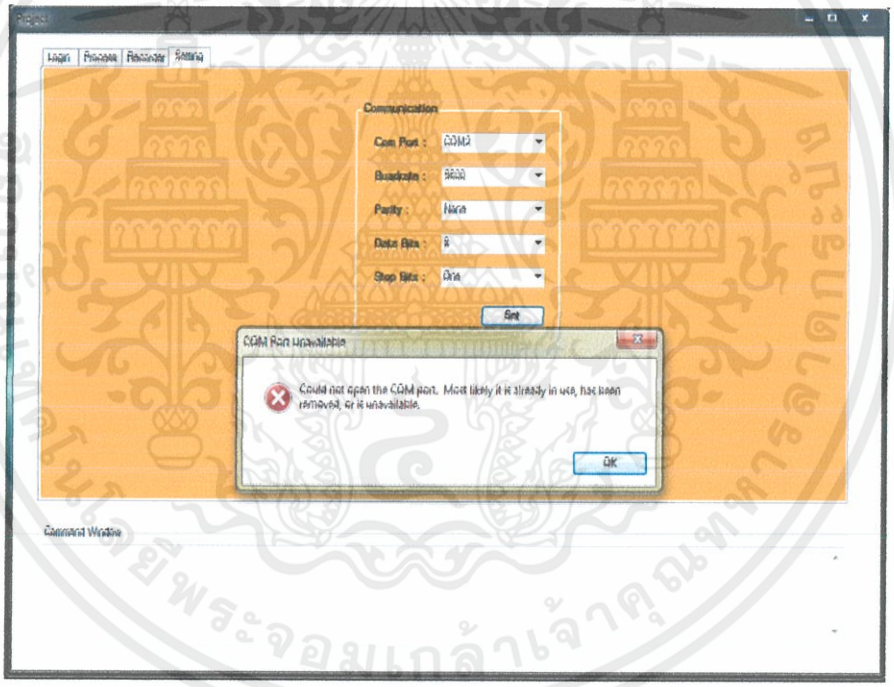
รูปที่ 3.15 ลักษณะของโปรแกรมเชื่อมต่อกับผู้ใช้งาน

อธิบายการทำงานของโปรแกรม

เริ่มต้น ผู้ใช้งานต้องตั้งค่าการเชื่อมต่อกับอุปกรณ์ภายนอกให้ตรงกับคอมพิวเตอร์ก่อน ผ่านหน้าต่างในรูป 3.15 (ก) ซึ่งในกรณียังไม่ได้ล็อกอิน จะแสดงเพียงส่วนการตั้งค่าการเชื่อมต่อเท่านั้น โดยมีรายละเอียดการตั้งค่าต่าง ๆ ดังนี้

- Com port คือ การเลือก port ให้ตรงกับอุปกรณ์ภายนอก โดยเครื่องคอมพิวเตอร์ จะกำหนดเลข Port ให้เองโดยอัตโนมัติ
- Buadrate คือ ความเร็วในการรับ – ส่งข้อมูลอนุกรมมีหน่วยเป็นบิตต่อวินาที โดยส่วนใหญ่จะเลือกที่ 9600 บิตต่อวินาที
- Parity คือ การตรวจสอบความถูกต้องในการสื่อสารหรือการรับส่งข้อมูลที่ส่งออกไป
- Data Bits คือ ความยาวของข้อมูล โดยส่วนใหญ่จะเลือกที่ 8 บิต
- Stop Bits คือ จำนวนบิตสัญญาณที่สั่งให้หยุดการทำงาน เมื่อการโอนข้อมูลสำเร็จแล้ว

หากการตั้งค่าไม่ถูกต้อง จะมีการเตือนเกิดขึ้นดังแสดงในรูปที่ 3.16 แต่หากตั้งค่าได้ถูกต้องแล้ว ผู้ใช้จึงจะสามารถเข้าสู่หน้าต่างล็อกอินได้ แต่หากยังไม่ได้ตั้งค่า จะปรากฏข้อความเตือนเมื่อเข้าสู่หน้าต่างล็อกอิน



รูปที่ 3.16 การแจ้งเตือนเมื่อตั้งค่าไม่ถูกต้อง

หลักจากตั้งค่าสำเร็จแล้ว ให้เลือกที่แถบ Login เพื่อทำการล็อกอินเข้าสู่ระบบดังรูปที่ 3.15 (ข) เมื่อล็อกอินสำเร็จแล้ว จะสามารถเข้าไปที่แถบ Process หรือแถบสั่งการทำงานได้ดังรูปที่ 3.15 (ค) โดยจากรูปดังกล่าวจะเห็นว่ามีหน้าต่างย่อย 6 หน้าต่าง เป็นหน้าต่างสั่งการทำงานของแต่ละไซโล ซึ่งในโครงงานนี้ใช้เพียง 2 ไซโล ในหน้าต่างย่อยนี้ประกอบด้วย น้ำหนักของวัตถุดิบที่ต้องการ จำนวนชุด

ของวัตถุดิบที่ต้องการ และรถลำเลียงวัตถุดิบที่ต้องการใช้งาน เมื่อเลือกค่าต่าง ๆ ตามที่ต้องการได้แล้ว ให้กดที่ปุ่ม Ok จะเป็นการทบทวนคำสั่งตามที่เลือกไว้ เมื่อตรวจสอบความถูกต้องแล้วให้กดที่ปุ่ม Confirm โปรแกรมจะส่งข้อมูลเป็นชุดตัวเลขให้กับโปรแกรมส่วนไมโครคอนโทรลเลอร์ จากนั้นจะมีหน้าต่างขึ้นมาใหม่แสดงดังรูปที่ 3.15 (ง) ซึ่งเป็นหน้าต่างแสดงสถานะการทำงานของรถลำเลียงวัตถุดิบและไซโล โดยสถานะของรถลำเลียงวัตถุดิบมีทั้งหมด 10 สถานะ ได้แก่

- Going to Wheat Flour Silo, Going to Gel Silo คือ กำลังไปที่ไซโล 1 หรือ ไซโล 2
- Stop at Wheat Flour Silo, Stop at Gel Silo คือ หยุดจอดที่ไซโล 1 หรือ ไซโล 2
- Feeding material คือ กำลังรอจ่ายวัตถุดิบจากไซโล
- Feeding completed คือ จ่ายวัตถุดิบเสร็จแล้ว
- *Going to storage* คือ *กำลังไปที่จุดเก็บวัตถุดิบ*
- Stop at storage คือ หยุดจอดที่จุดเก็บวัตถุดิบ
- Sending material to storage คือ กำลังส่งวัตถุดิบเข้าที่เก็บวัตถุดิบ
- Material Stocked คือ เก็บวัตถุดิบเสร็จแล้ว
- Going to stop point คือ กำลังไปที่หยุดจอดรถ
- Stop คือ หยุดการทำงาน

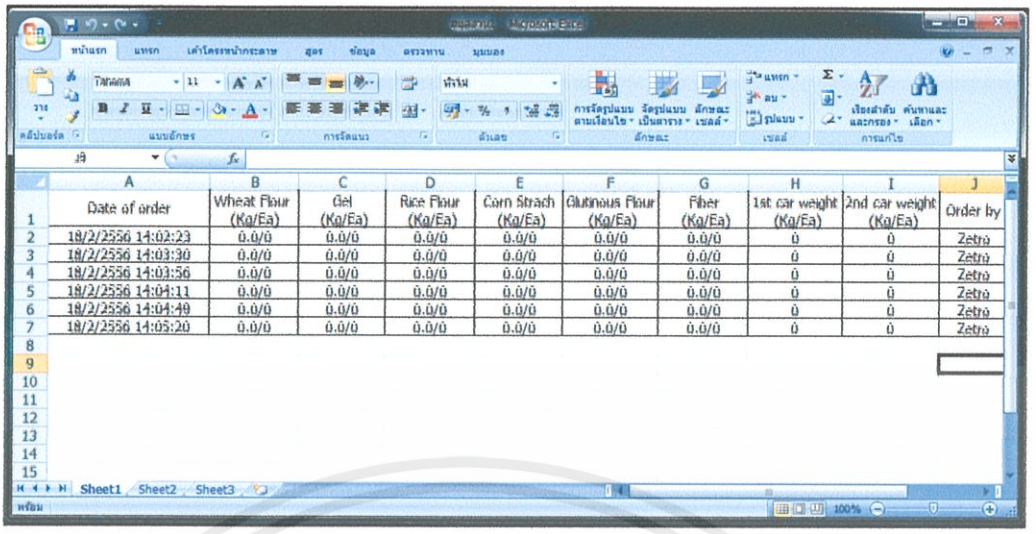
และสถานะของไซโลมี 2 สถานะได้แก่

- Available คือ มีวัตถุดิบเหลือสูงกว่าระดับที่กำหนด
- Unavailable คือ มีวัตถุดิบเหลือต่ำกว่าระดับที่กำหนด

หากสถานะของไซโลเปลี่ยนเป็น Unavailable จะมีหน้าต่างแจ้งเตือนขึ้นมาเพื่อให้ผู้ใช้งานเติมวัตถุดิบ ดังรูปที่ 3.17

รูปที่ 3.17 การแจ้งเตือนของไซโลเมื่อวัตถุดิบในไซโลเหลือต่ำกว่าระดับที่กำหนด

เมื่อรถลำเลียงวัตถุดิบกลับมาที่จุดเริ่มต้นแล้ว สามารถบันทึกการทำงานที่ผ่านมาได้โดยกดที่ปุ่ม Save ข้อมูลจะเก็บบันทึกลงในฐานข้อมูลโดยใช้ไมโครซอฟท์แอคเซส แล้วดึงข้อมูลออกมาแสดงเป็นตาราง ซึ่งอยู่ในแถบ Recorder ดังหน้าต่างในรูปที่ 3.15 (จ) ข้อมูลที่แสดงทั้งหมดสามารถนำไปบันทึกเก็บเป็นไฟล์ข้อมูลได้ในรูปแบบของไมโครซอฟต์เอคเซลได้ดังรูปที่ 3.18 เป็นการสิ้นสุดการทำงาน

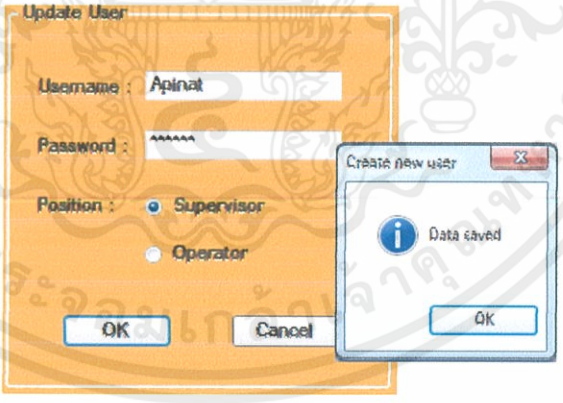


	A	B	C	D	E	F	G	H	I	J
1	Date of order	Wheat Flour (Kg/Ea)	Gel (Kg/Ea)	Rice Flour (Kg/Ea)	Corn Strach (Kg/Ea)	Glutinous Flour (Kg/Ea)	Fiber (Kg/Ea)	1st car weight (Kg/Ea)	2nd car weight (Kg/Ea)	Order by
2	18/2/2556 14:02:23	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
3	18/2/2556 14:03:30	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
4	18/2/2556 14:03:56	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
5	18/2/2556 14:04:11	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
6	18/2/2556 14:04:49	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
7	18/2/2556 14:05:20	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0	0	Zetro
8										
9										
10										
11										
12										
13										
14										
15										

Sheet1 Sheet2 Sheet3

รูปที่ 3.18 การบันทึกผลการทำงานในรูปแบบของไมโครซอฟท์เอคเซล

นอกจากนั้น โปรแกรมยังสามารถสร้างชื่อผู้ใช้งานใหม่ได้ ผ่านทางหน้าต่างในรูปที่ 3.15 (ก) โดยหน้าต่างย่อยนี้จะปรากฏขึ้นก็ต่อเมื่อทำการล็อกอินแล้ว ข้อมูลใหม่ที่สร้างขึ้นจะบันทึกลงในฐานข้อมูลอีกฐานข้อมูลหนึ่งโดยใช้ไมโครซอฟท์แอคเซสเช่นเดียวกัน จากหน้าต่างดังกล่าว หากเลือก Position เป็น Supervisor ผู้ใช้งานที่สร้างขึ้นใหม่นี้จะสามารถสั่งการทำงานทุกอย่างได้และสามารถสร้างชื่อผู้ใช้เพิ่มขึ้นอีกได้ แต่หากเลือก Position เป็น Operator จะไม่สามารถสร้างชื่อผู้ใช้ใหม่ได้ เมื่อสร้างชื่อผู้ใช้งานและเลือก Position เสร็จแล้วให้กดที่ปุ่ม Ok เป็นการบันทึกลงฐานข้อมูลดังกล่าว แสดงดังรูปที่ 3.19



รูปที่ 3.19 การสร้างชื่อผู้ใช้งานใหม่

3.3.2 โปรแกรมส่วนไมโครคอนโทรลเลอร์

ในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์รุ่น ET-ARM STAMP STAM32 โดยใช้งานโมดูล DMA ร่วมกับโมดูล USART และโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล และได้กำหนดขาสัญญาณขาเข้ากับสัญญาณขาออกตามคุณสมบัติพิเศษของไมโครคอนโทรลเลอร์แต่ละขา รายละเอียดการทำงานของแต่ละขามีดังนี้

ขา PB8, PB9 ซึ่งเป็นขาสัญญาณเอาต์พุตที่มีลักษณะสัญญาณเป็นสัญญาณพีดับบลิวเอ็ม ใช้เชื่อมต่อกับขา ENABLE ของไอซี L293D มีหน้าที่ในการขับมอเตอร์ล้อทั้ง 2 ข้าง

ขา PB10, PB11, PB12, PB13 ซึ่งเป็นขาสัญญาณเอาต์พุตที่มีลักษณะเป็นรูปแบบเปิด – ปิด เชื่อมต่อกับไอซี L293D มีหน้าที่ในการกำหนดทิศทางของการหมุนของล้อทั้ง 2 ข้าง

ขา PC5, PC6 เป็นขาสัญญาณเอาต์พุตที่มีลักษณะเป็นรูปแบบเปิด – ปิด มีไว้สำหรับเชื่อมต่อกับวงจรขับรีเลย์และควบคุมทิศทางการเคลื่อนที่ของสายพาน

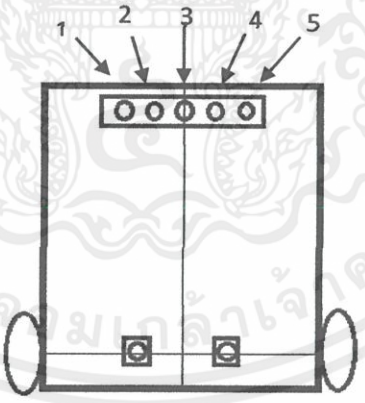
ขา PC0, PC1, PC2, PC3, PC4, PB0, PB1 เป็นขาสัญญาณอินพุตที่รับค่ามาจากเซนเซอร์ทั้ง 7 ตัว มีหน้าที่ช่วยตรวจสอบตำแหน่งของรถ

ขา PA9, PA10 เป็นช่องสัญญาณที่ใช้ในการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับเครื่องคอมพิวเตอร์

ขา PA2, PA3 เป็นช่องสัญญาณที่ใช้ในการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับซิกบี

3.3.2.1 การวัดการเบี่ยงเบนโดยออพติคัลเซนเซอร์

การจะควบคุมรถลำเลียงได้นั้นจำเป็นต้องรู้การเบี่ยงเบนจากแนวเส้นทางเดินปัจจุบันของรถลำเลียง ซึ่งในหัวข้อนี้จะอธิบายความสัมพันธ์ระหว่างค่าวัดจากออพติคัลเซนเซอร์ทั้ง 5 ตัว กับระดับความเบี่ยงเบนจากแนวเส้นทางเดิน โดยกำหนดลำดับออพติคัลเซนเซอร์ตามรูปที่ 3.20



รูปที่ 3.20 การกำหนดลำดับเซนเซอร์

เมื่อทราบถึงลำดับของออพติคัลเซนเซอร์แล้ว สามารถออกแบบตารางเพื่อตรวจสอบว่ารถลำเลียงเบี่ยงเบนออกจากเส้นมากน้อยเพียงใด โดยได้กำหนดระดับค่าเบี่ยงเบนที่วัดไว้ 9 ระดับ ซึ่งเป็นระดับที่นำไปคำนวณความเร็วของมอเตอร์ขับเคลื่อนล้อต่อไป แสดงได้ดังตารางที่ 3.1 เมื่อ

0 หมายถึงไม่พบเส้นสีขาว

1 หมายถึงพบเส้นสีขาว

ตารางที่ 3.1 ค่าออพติคัลเซนเซอร์ที่สอดคล้องกับระดับความเบี่ยงเบนจากแนวเส้นทางเดิน

สถานะการทำงาน	ลำดับของออพติคัลเซนเซอร์					มุม* (องศา)	ระดับค่าวัด
	1	2	3	4	5		
1	0	0	0	0	1	24 ถึง 28	4
2	0	0	0	1	1	17 ถึง 21	3
3	0	0	0	1	0	10 ถึง 14	2
4	0	0	1	1	0	3 ถึง 7	1
5	0	0	1	0	0	0	0
6	0	1	1	0	0	-3 ถึง -7	-1
7	0	1	0	0	0	-10 ถึง -14	-2
8	1	1	0	0	0	-17 ถึง -21	-3
9	1	0	0	0	0	-24 ถึง -28	-4

* ทิศทางของมุมกำหนดโดย มุมเป็นบวก หมายถึง รถลำเลียงเบี่ยงเบนจากเส้นทางไปทางซ้าย
มุมเป็นลบ หมายถึง รถลำเลียงเบี่ยงเบนจากเส้นทางไปทางขวา

นอกจากนั้น ออพติคัลเซนเซอร์ยังสามารถใช้ตรวจสอบจุดจอดรถลำเลียง จุดที่มีการเลี้ยวหักศอก และจุดที่รถลำเลียงกำลังจะหลุดออกจากเส้นทาง ดังแสดงได้ตามตารางที่ 3.2

ตารางที่ 3.2 ค่าออพติคัลเซนเซอร์ที่สอดคล้องกับสถานะการทำงานกรณีพิเศษ

สถานะการทำงาน	ลำดับออพติคัลเซนเซอร์					สถานะที่อ่านค่าได้
	1	2	3	4	5	
1	1	1	1	0	0	ถึงจุดเลี้ยวหักศอกทางซ้าย
2	0	0	1	1	1	ถึงจุดเลี้ยวหักศอกทางขวา
3	1	1	1	1	1	จุดจอดรถลำเลียง
4	1	0	0	0	0	รถลำเลียงเบี่ยงออกทางขวามากเกินไป
5	0	0	0	0	1	รถลำเลียงเบี่ยงออกทางซ้ายมากเกินไป

3.3.2.2 การทำงานของโปรแกรมหลัก

ในส่วนโปรแกรมหลักแบ่งออกเป็น 2 ส่วน สำหรับรถลำเลียงสองคันแต่หลักการทำงานนั้นจะคล้ายคลึงกัน ซึ่งมีลำดับการทำงานดังนี้

1. รถลำเลียงจอดรอ ณ จุดจอดรถ รอรับคำสั่งค่าไซโลและจำนวนรอบที่ต้องลำเลียงจากระบบต้นแบบไซโลซึ่งจะส่งข้อมูลโดยไมโครคอนโทรลเลอร์ผ่านโมดูลซิกบี โดยการรับข้อมูลของไมโครคอนโทรลเลอร์บนรถลำเลียงจะรับผ่านโมดูล USART และจะรอรับข้อมูลจนกว่าข้อมูลจะครบถ้วนและถูกต้อง

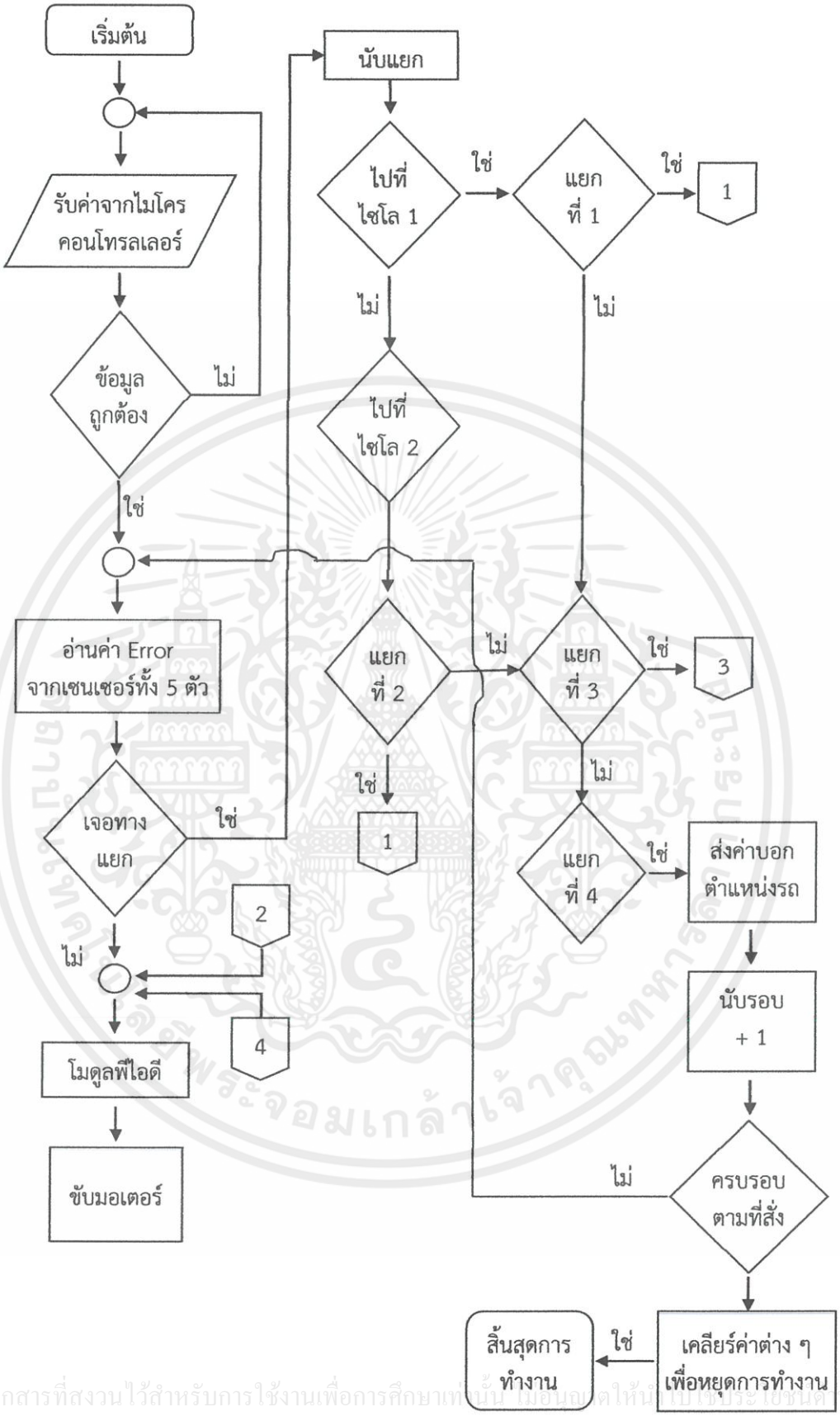
2. เมื่อรับข้อมูลเรียบร้อยแล้ว ไมโครคอนโทรลเลอร์บนรถลำเลียงจะอ่านค่าออพติคัลเซนเซอร์ทั้ง 5 ตัว ข้างหน้า เพื่อวัดมุมเบี่ยงเบนจากเส้นทางเดิน

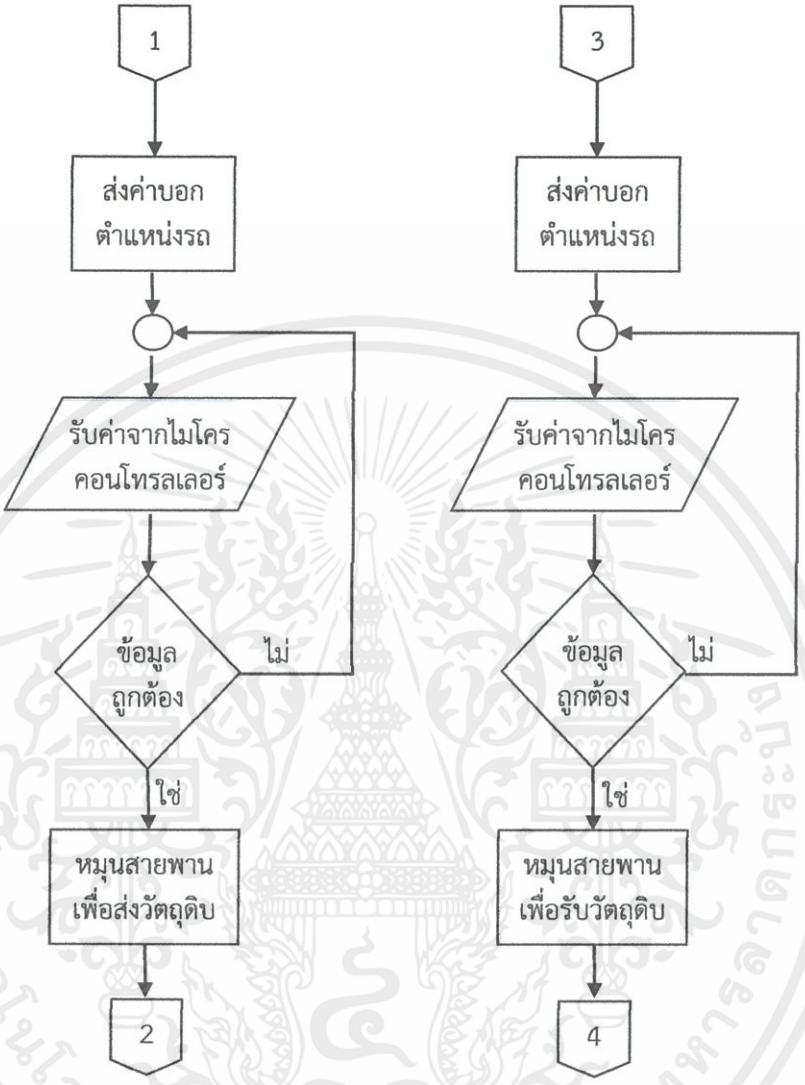
3. ตรวจสอบเงื่อนไขว่าเจอจุดแยกหรือไม่ ถ้าเจอจุดแยก ให้เพิ่มจุดแยกในฐานข้อมูล การนับจุดแยกเก็บในฐานข้อมูลนี้ เป็นค่าที่ใช้บอกตำแหน่งรถลำเลียงว่าอยู่ ณ จุดใด โดยหากจุดแยกนั้นเป็นจุดแยกที่ต้องการจอดรถตามคำสั่งที่ได้รับ ให้เข้าโมดูลทำงานตามคำสั่ง แต่ถ้าไม่ใช่จุดแยกที่ต้องการหรือถ้าไม่เจอจุดแยก ให้สั่งเข้าโมดูลการขับเคลื่อนรถลำเลียงเดินตามเส้นโดยอาศัยตัวควบคุมพีไอดี และทำอย่างนี้ทุกครั้งในทุกลูปการทำงาน

กรณีเจอจุดแยกที่ต้องการจอดรถตามคำสั่งที่ได้รับ อาทิ ได้รับคำสั่งให้ไปไซโลที่หนึ่ง และจำนวนแยกนับได้ตรงตามเส้นทางที่จะไปไซโลที่หนึ่งแล้ว โปรแกรมจะเข้าสู่โมดูลทำงานตามคำสั่ง โดยไมโครคอนโทรลเลอร์จะสั่งให้รถลำเลียงหยุดและส่งสัญญาณทางโมดูลซิกบีไปยังระบบไซโล เพื่อให้ระบบไซโลเลื่อนวัตถุดิบมาที่รถลำเลียงโดยอาศัยการหน่วงเวลาในการเลื่อนสายพานที่รถลำเลียง ซึ่งจากการทดลองหลาย ๆ ครั้งจะพบว่าใช้เวลาประมาณ 5 วินาที เมื่อระบบเลื่อนวัตถุดิบเข้าสู่รถลำเลียงเรียบร้อยแล้วจะกลับเข้าสู่โมดูลการขับเคลื่อนรถลำเลียงต่อไป โดยนับแยกจนถึงจุดแยกที่มีการส่งวัตถุดิบก็จะกลับมาสู่โมดูลการส่งวัตถุดิบโดยการเลื่อนสายพานบนรถลำเลียง หลังจากนั้นก็จะกลับเข้าสู่โมดูลการขับเคลื่อนรถลำเลียงต่อไป และจะนับแยกจนถึงจุดแยกที่มีไว้จอดรถลำเลียง ไมโครคอนโทรลเลอร์ก็จะสั่งให้รถลำเลียงหยุดและตรวจสอบว่ารถลำเลียงวิ่งครบรอบตามที่สั่งแล้วหรือไม่ ถ้าใช่ก็จะสั่งให้รถลำเลียงหยุด ถ้าไม่ก็จะสั่งให้กลับเข้าสู่โมดูลการขับเคลื่อนรถลำเลียงตามคำสั่งต่อไป

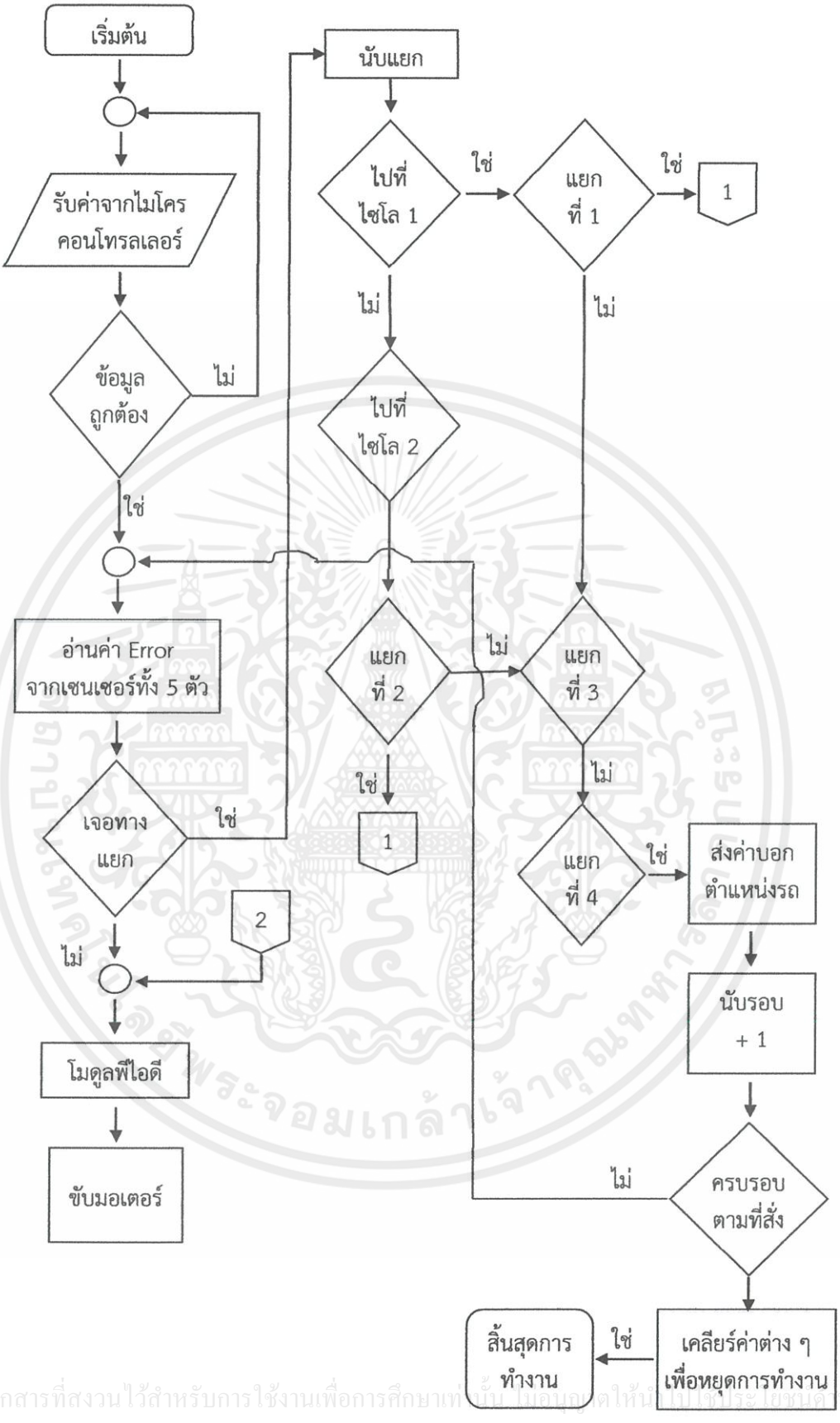
กรณียังไม่ถึงจุดแยกที่ต้องการหรือไม่ใช่จุดแยก โปรแกรมจะเข้าโมดูลการขับเคลื่อนรถลำเลียงเดินตามเส้นโดยอาศัยตัวควบคุมพีไอดี (โมดูลพีไอดี) โปรแกรมจะทำงานโดยการรับค่าป้อนกลับระดับความเบี่ยงเบนจากแนวเส้นทางเดินเข้าสู่สมการกฎการควบคุมของตัวควบคุมพีไอดี เพื่อคำนวณสัญญาณขับมอเตอร์ของล้อรถลำเลียงทั้งสองข้าง โดยรายละเอียดจะกล่าวในหัวข้อถัดไป

แผนผังการทำงานของโปรแกรมหลักของไมโครคอนโทรลเลอร์ที่ใช้สำหรับควบคุมรถลำเลียงวัตถุดิบทั้ง 2 คัน แสดงได้ดังรูปที่ 3.21 สำหรับรถลำเลียงคันที่ 1 และ รูปที่ 3.22 สำหรับรถลำเลียงคันที่ 2





รูปที่ 3.21 แผนผังการทำงานของรถลำเลียงคันที่ 1



รูปที่ 3.22 แผนผังการทำงานของรถลำเลียงคันที่ 2

3.3.2.3 กฎการควบคุมของตัวควบคุมแบบพีไอดีและการปรับแต่ง

จากที่ได้กล่าวไว้ในหัวข้อการทำงานของโปรแกรมหลักของไมโครคอนโทรลเลอร์สำหรับรถลำเลียงในหัวข้อ 3.3.2.2 ในกรณีรถลำเลียงยังไม่ถึงจุดแยกที่ต้องการหรือไม่ใช่จุดแยก โปรแกรมจะเข้าโมดูลการขับเคลื่อนรถลำเลียงเดินตามเส้นโดยอาศัยตัวควบคุมพีไอดี (โมดูลพีไอดี) โดยกฎการควบคุมแบบพีไอดีได้กล่าวไว้ในบทที่ 2

จากสมการที่ (2.16) พบว่ากฎการควบคุมสามารถคำนวณได้จากการดำเนินการ 3 ส่วน ได้แก่ ผลการควบคุมจากการแปรผันตรงกับความผิดพลาด ผลการควบคุมจากการแปรผันตรงกับการอินทิกรัลของความผิดพลาด และผลการควบคุมจากการแปรผันตรงกับการอนุพันธ์ของความผิดพลาด ดังอธิบายได้ตามสมการที่ (3.1)

$$U = K_p(error) + K_i(Integral) + K_d(Derivative) \tag{3.1}$$

อย่างไรก็ตามการดำเนินการตามสมการที่ (3.1) ต้องการการประมวลผลสัญญาณเวลาต่อเนื่อง โดยเฉพาะในการดำเนินการอินทิกรัล และการดำเนินการอนุพันธ์ แต่ในโครงงานนี้ได้อาศัยไมโครคอนโทรลเลอร์ซึ่งไม่สามารถประมวลผลสัญญาณเวลาต่อเนื่องได้โดยตรง ดังนั้นจึงจำเป็นต้องประมาณการดำเนินการอินทิกรัล และการดำเนินการอนุพันธ์ ด้วยวิธีการเชิงเลขเพื่อให้เป็นการประมวลผลสัญญาณดิจิตอล ที่สามารถดำเนินการได้โดยไมโครคอนโทรลเลอร์ โดยการประมวลผลจะใช้การชักตัวอย่างด้วยคาบการชักตัวอย่างคงที่ทุก T วินาที

การประมาณเทอมอินทิกรัลของความผิดพลาดอธิบายได้ดังนี้ เมื่อพิจารณาผลการอินทิกรัล ณ เวลา $t = kT$ เมื่อ k คือจุดการชักตัวอย่าง ณ เวลาปัจจุบัน เทอมอินทิกรัลของความผิดพลาดจะสามารถแบ่งได้ตามสมการที่ (3.2)

$$\int_0^{kT} e(t)dt = \int_0^{(k-1)T} e(t)dt + \int_{(k-1)T}^{kT} e(t)dt$$

$$Integral(k) = Integral(k-1) + \int_{(k-1)T}^{kT} e(t)dt \tag{3.2}$$

ซึ่งผลการอินทิกรัล ณ เวลาปัจจุบันเท่ากับผลการอินทิกรัลในอดีตจนถึงจุดการชักตัวอย่าง ณ เวลาก่อนหน้าที่ $k-1$ บวกกับผลการอินทิกรัลจากจุดการชักตัวอย่างที่ $k-1$ ถึงจุดการชักตัวอย่างปัจจุบันที่ k โดยผลการอินทิกรัลส่วนท้ายนี้สามารถประมาณได้หลายวิธี โดยในโครงงานนี้เลือกประมาณด้วยพื้นที่สี่เหลี่ยมผืนผ้า โดยนำค่าความผิดพลาดปัจจุบันคูณกับคาบการชักตัวอย่าง หรือเรียกวิธีการนี้ว่าแบ็คเวิร์ดออยเลอร์ (Backward Euler) แสดงได้ดังสมการที่ (3.3)

$$Integral_{now} = Integral_{previous} + T \cdot error_{now} \tag{3.3}$$

การประมาณเทอมอนุพันธ์ของความผิดพลาดอธิบายได้ดังนี้ เมื่อพิจารณาผลการอนุพันธ์ ณ เวลา $t = kT$ เทอมอนุพันธ์ของความผิดพลาดจะสามารถประมาณได้ด้วยผลต่างสองจุด คือ การนำค่าความผิดพลาดที่ตำแหน่งปัจจุบันลบกับค่าความผิดพลาดในอดีต แล้วหารด้วยคาบการชักตัวอย่าง ดังสมการที่ (3.4)

$$\frac{de(t)}{dt} \approx \frac{e(kT) - e((k-1)T)}{T} \tag{3.4}$$

ซึ่งในการเขียนโปรแกรมสามารถดำเนินการได้ตามสมการที่ (3.5)

$$Derivative = \frac{error_{now} - error_{previous}}{T} \tag{3.5}$$

จากแนวคิดการประมาณการเชิงเลขข้างต้น สามารถเขียนโปรแกรมประมาณกฎการควบคุมแบบพีไอดีด้วยไมโครคอนโทรลเลอร์ได้ดังแสดงในรูปที่ 3.23 ในกรณีนี้ได้กำหนดตัวแปรการชักตัวอย่างด้วย dt แทนตัวแปร T และในโปรแกรมที่ใช้จริงในโครงงานนี้แสดงไว้ในภาคผนวก ข.5 โดยเลือกใช้คาบการชักตัวอย่างที่ 0.011 วินาที

```
previous_error = setpoint - actual_position     // กำหนดค่าความผิดพลาดครั้งก่อนตั้งต้น
                                                โดยการนำตำแหน่งที่ต้องการลบกับตำแหน่ง
                                                ปัจจุบัน
integral = 0                                    // กำหนดให้ค่าการอินทิกรัลตั้งต้นเท่ากับ 0
start:
  error = setpoint - actual_position            // หาค่าความผิดพลาด โดยการนำตำแหน่ง
                                                ที่ต้องการลบกับตำแหน่งปัจจุบัน
  integral = integral + (error*dt)              // หาค่าอินทิกรัล โดยนำค่าอินทิกรัลเดิม
                                                บวกด้วยค่าความผิดพลาดที่เกิดขึ้นคูณกับ
                                                เวลา
  derivative = (error - previous_error)/dt      // หาค่าอนุพันธ์ โดย การนำค่าความ
                                                ผิดพลาดปัจจุบันลบกับค่าความผิดพลาดครั้ง
                                                ก่อน แล้วหารด้วยเวลา
  output = (Kp*error) + (Ki*integral) + (Kd*derivative)  // คำนวณสมการ โดยค่าคงที่ Kp คูณ
                                                กับค่าความผิดพลาดรวมกับค่า Ki คูณกับ
                                                เทอมอินทิกรัลรวมกับค่า Kd คูณกับเทอม
                                                อนุพันธ์
  previous_error = error                        // เก็บค่าความผิดพลาดครั้งก่อนไว้คำนวณ
  wait(dt)                                      //  รอเวลา
  goto start
```

รูปที่ 3.23 แนวคิดการเขียนโปรแกรมประมาณกฎการควบคุมแบบพีไอดีด้วยไมโครคอนโทรลเลอร์

การที่กฎการควบคุมแบบพีไอดีจะให้ผลการควบคุมที่น่าพึงพอใจ ก็ต่อเมื่อสามารถเลือกใช้ค่าพารามิเตอร์ K_p K_i และ K_d ที่เหมาะสม ซึ่งการปรับแต่งหากทราบแบบจำลองทางคณิตศาสตร์ของระบบที่ต้องการควบคุมสามารถปรับแต่งได้โดยการออกแบบเชิงวิเคราะห์ อย่างไรก็ตามในกรณีระบบควบคุมแบบพีไอดีของรถลำเลียงไม่สามารถหาแบบจำลองทางคณิตศาสตร์ได้ จึงจำเป็นต้องอาศัยการปรับแต่งจากการทดลองซึ่งสามารถทำได้โดยอาศัยกฎการปรับแต่ง (Tuning rule) อาทิกฎการปรับแต่งของซิกเกอร์-นิโคล (Ziegler-Nichols tuning rule) หรืออาศัยการปรับค่าแบบลองผิดลองถูก (Trial and error) โดยในโครงงานนี้ได้เลือกดำเนินการดังนี้ ในขั้นแรกให้ตั้งค่า K_i และ K_d เป็นศูนย์ แล้วเพิ่มค่า K_p จนกระทั่งสัญญาณเอาต์พุตเกิดการแกว่งคงที่ (Sustained oscillation) จากนั้นตั้งค่า K_p ให้เหลือครึ่งหนึ่งของค่าที่ทำให้เกิดการแกว่ง และค่อยปรับเพิ่มค่า K_i จนกระทั่งระบบสามารถติดตามเส้นทางลำเลียงได้ไม่มีค่าความผิดพลาดที่สภาวะคงตัว มีผลตอบสนองเร็วและมีค่าพุ่งเกิน (Overshoot) ที่ยอมรับได้ อย่างไรก็ตามไม่ควรปรับค่า K_i มากเกินไปเพราะจะทำให้ระบบไม่เสถียรได้ ขั้นตอนสุดท้ายถ้าต้องการลดค่าพุ่งเกินให้เพิ่มค่า K_d อย่างไรก็ตามไม่ควรใช้ค่าที่สูงเกินไป เพราะอาจขยายสัญญาณรบกวนความถี่สูงในระบบได้

บทที่ 4
ผลการทดลอง

ในบทนี้นำเสนอการทดลองของโครงงาน ซึ่งแบ่งออกเป็น 2 การทดลอง โดยการทดลองแรกเป็นการทดลองโปรแกรมที่พัฒนาขึ้น และการทดลองที่สอง เป็นการทดลองการทำงานของรถลำเลียงวัตถุดิบ ซึ่งแบ่งออกเป็น 2 การทดลอง คือ การทดลองวิ่งทางปกติ และการทดลองวิ่งทางโค้งหักศอก รายละเอียดของการทดลองเป็นดังนี้

4.1 การทดลองโปรแกรมที่พัฒนาขึ้น

ในการทดลองนี้ เป็นการทดลองของโปรแกรมที่ได้พัฒนาขึ้น โดยจะแสดงส่วนสั่งการทำงาน เมื่อเลือกน้ำหนัก จำนวนชุดที่ต้องการ และเลือกรถลำเลียงวัตถุดิบที่ใช้ การแสดงสถานะล่าสุดของรถลำเลียงวัตถุดิบแต่ละคัน การแจ้งเตือนเมื่อปริมาณวัตถุดิบในไซโลเหลือน้อย ตารางบันทึกผลการทำงานตามค่าที่ได้เลือกไว้ รวมไปถึงการสร้างชื่อผู้ใช้ใหม่ รายละเอียดต่าง ๆ แสดงได้ดังนี้

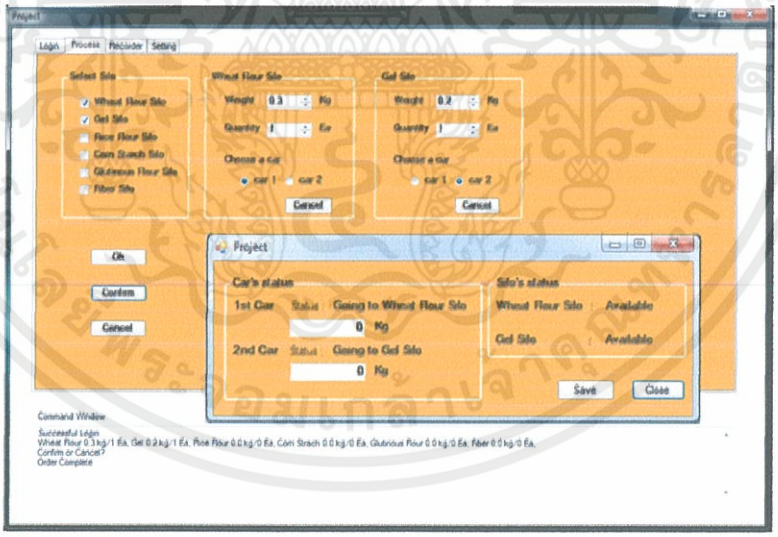
(ก) รอการป้อนชื่อและรหัสผู้ใช้งาน (ข) ชื่อและรหัสผู้ใช้งานถูกต้อง

รูปที่ 4.1 หน้าต่างล็อกอินเข้าสู่ระบบ

รูปที่ 4.1 แสดงหน้าต่างล็อกอินเข้าสู่ระบบ โดยรูปที่ 4.1 (ก) เป็นหน้าต่างเพื่อให้ผู้ใช้ป้อนชื่อผู้ใช้งานและรหัสผ่าน ส่วนรูปที่ 4.1 (ข) แสดงการล็อกอินเข้าสู่ระบบสำเร็จ ซึ่งก่อนทำการล็อกอินจะต้องทำการตั้งค่าการเชื่อมต่อกับอุปกรณ์ภายนอกกับคอมพิวเตอร์ให้ตรงกันก่อนในลักษณะดังรูปที่ 3.15 (ก) เมื่อล็อกอินได้แล้ว เลือกที่แถบ Process เพื่อเข้าสู่หน้าต่างสั่งการทำงานในลักษณะดังรูปที่ 3.15 (ค) จากนั้นเลือกวัตถุดิบที่ต้องการ, น้ำหนัก, จำนวนชุดและรถลำเลียงวัตถุดิบที่ต้องการ โดยในการทดลองนี้ เลือกน้ำหนัก 0.3 Kg จำนวน 1 ชุด รถลำเลียงคันที่ 1 สำหรับไซโลที่ 1 และเลือกน้ำหนัก 0.2 Kg จำนวน 1 ชุด รถลำเลียงคันที่ 2 สำหรับคันที่ 2 ดังรูปที่ 4.2

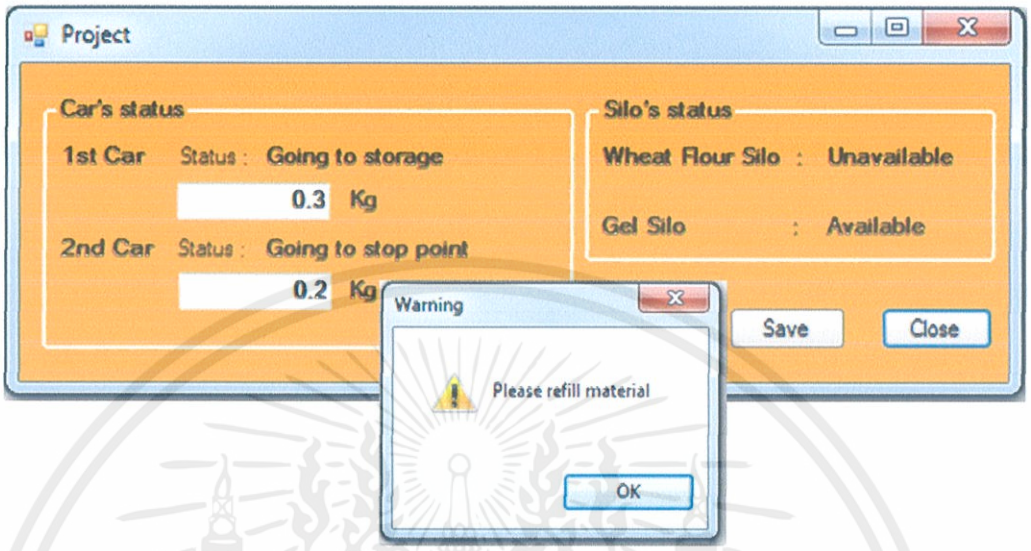
รูปที่ 4.2 หน้าต่างเลือกน้ำหนักและจำนวนชุดของวัตถุดิบ

จากนั้น เมื่อกดที่ปุ่ม Ok จะเป็นการทบทวนคำสั่งค่าต่าง ๆ ที่ได้เลือกไว้ก่อนนำข้อมูลข้างต้นส่งเป็นชุดข้อมูลให้กับโปรแกรมส่วนไมโครคอนโทรลเลอร์ เมื่อตรวจสอบว่าถูกต้องแล้วให้กดที่ปุ่ม Confirm โปรแกรมจะส่งข้อมูลออกไปและจะมีหน้าต่างแสดงสถานะขึ้นมาดังรูปที่ 4.3 ซึ่งสถานะของทั้งรถลำเลียงวัตถุดิบและไซโลทั้งหมดได้กล่าวไว้แล้วในบทที่ 3.3.1



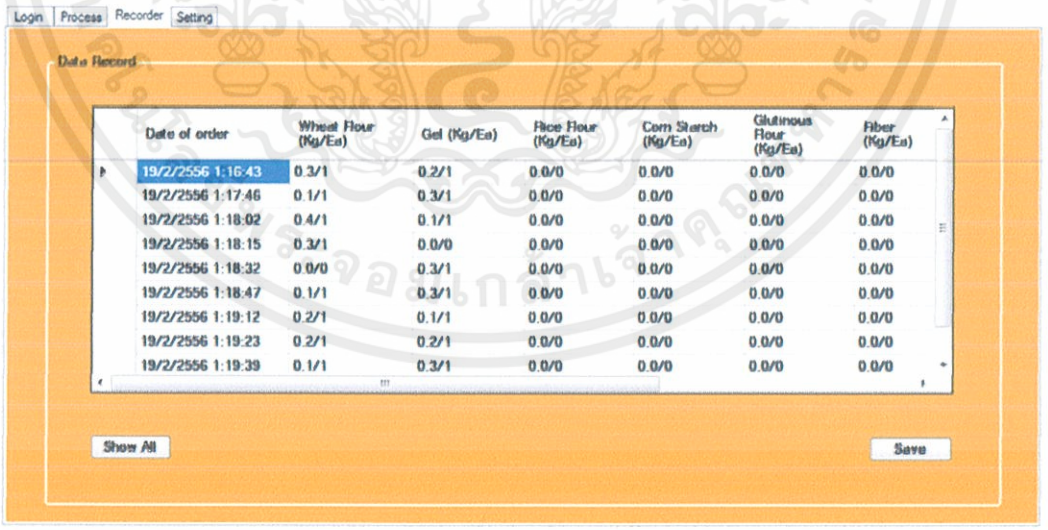
รูปที่ 4.3 หน้าต่างแสดงสถานะของรถลำเลียงวัตถุดิบและไซโล

หากปริมาณวัตถุดิบในไซโลเหลือน้อยลง โดยตรวจสอบจากอัลตราโซนิคเซนเซอร์ในระยะที่กำหนดไว้ โปรแกรมจะทำการแจ้งเตือนผู้ใช้งานเพื่อให้เติมวัตถุดิบลงในไซโลที่มีปริมาณวัตถุดิบเหลือน้อย ดังแสดงในรูปที่ 4.4



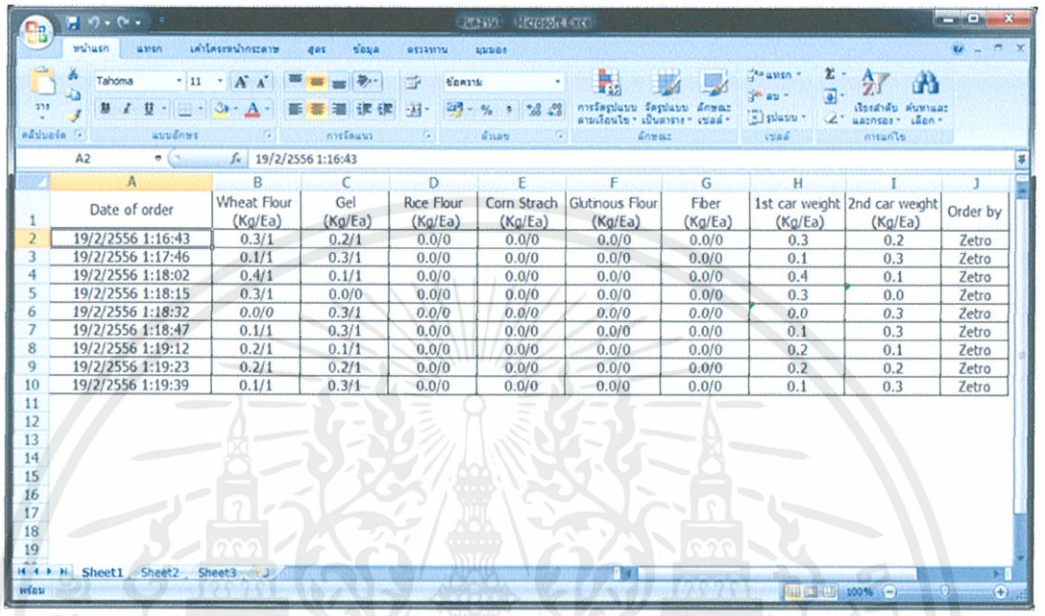
รูปที่ 4.4 การแจ้งเตือนเมื่อปริมาณวัตถุดิบในไซโลเหลือน้อย

เมื่อรถลำเลียงวัตถุดิบนำวัตถุดิบไปส่งที่รองรับและกลับมาที่จุดเริ่มต้นแล้ว จะสามารถทำการบันทึกข้อมูลที่ได้เลือกไว้ โดยกดที่ปุ่ม Save ข้อมูลจะถูกบันทึกลงในฐานข้อมูลโดยใช้โปรแกรมไมโครซอฟท์แอคเซส แล้วดึงข้อมูลออกมาแสดงเป็นตารางในแถบ Recorder ดังรูปที่ 4.5



รูปที่ 4.5 การแสดงข้อมูลที่บันทึก

รูปที่ 4.5 เห็นว่า ค่าที่ได้เลือกไว้ในข้างต้นคือ น้ำหนัก 0.3 Kg จำนวน 1 ชุด สำหรับไซโลที่ 1 และเลือกน้ำหนัก 0.2 Kg จำนวน 1 ชุด สำหรับคันที่ 2 มีข้อมูลตรงกันกับตารางข้างต้น จากนั้น เมื่อกดที่ปุ่ม Save ด้านล่าง จะมีโปรแกรมไมโครซอฟท์เอคเซลขึ้นมาและข้อมูลในตารางข้างต้นจะปรากฏโดยอัตโนมัติ ดังแสดงในรูปที่ 4.6 เมื่อข้อมูลทั้งหมดอยู่ในรูปแบบของไมโครซอฟท์เอคเซล จะสามารถบันทึกเก็บเป็นฐานข้อมูลการทำงานเพื่อสามารถตรวจสอบความถูกต้องของข้อมูลในภายหลังได้



	A	B	C	D	E	F	G	H	I	J
1	Date of order	Wheat Flour (Kg/Ea)	Gel (Kg/Ea)	Rice Flour (Kg/Ea)	Corn Strach (Kg/Ea)	Glutnous Flour (Kg/Ea)	Fiber (Kg/Ea)	1st car weight (Kg/Ea)	2nd car weight (Kg/Ea)	Order by
2	19/2/2556 1:16:43	0.3/1	0.2/1	0.0/0	0.0/0	0.0/0	0.0/0	0.3	0.2	Zetro
3	19/2/2556 1:17:46	0.1/1	0.3/1	0.0/0	0.0/0	0.0/0	0.0/0	0.1	0.3	Zetro
4	19/2/2556 1:18:02	0.4/1	0.1/1	0.0/0	0.0/0	0.0/0	0.0/0	0.4	0.1	Zetro
5	19/2/2556 1:18:15	0.3/1	0.0/0	0.0/0	0.0/0	0.0/0	0.0/0	0.3	0.0	Zetro
6	19/2/2556 1:18:32	0.0/0	0.3/1	0.0/0	0.0/0	0.0/0	0.0/0	0.0	0.3	Zetro
7	19/2/2556 1:18:47	0.1/1	0.3/1	0.0/0	0.0/0	0.0/0	0.0/0	0.1	0.3	Zetro
8	19/2/2556 1:19:12	0.2/1	0.1/1	0.0/0	0.0/0	0.0/0	0.0/0	0.2	0.1	Zetro
9	19/2/2556 1:19:23	0.2/1	0.2/1	0.0/0	0.0/0	0.0/0	0.0/0	0.2	0.2	Zetro
10	19/2/2556 1:19:39	0.1/1	0.3/1	0.0/0	0.0/0	0.0/0	0.0/0	0.1	0.3	Zetro

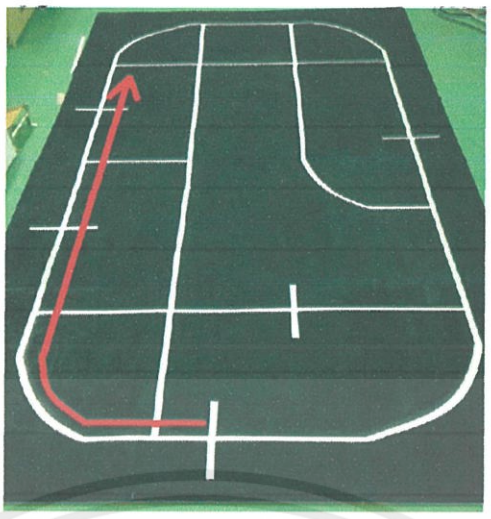
รูปที่ 4.6 การบันทึกผลการทำงานในรูปแบบของไมโครซอฟท์เอคเซล

4.2 การทดลองการทำงานของรถลำเลียงวัตถุดิบ

ในการทดลองการทำงานของรถลำเลียงวัตถุดิบแบ่งออกเป็น 2 การทดลอง คือ การทดลองแบบวิ่งทางปกติ และการทดลองแบบวิ่งเข้าโค้งหักศอก ซึ่งการทดลองแบบวิ่งทางปกติ ยังแบ่งออกเป็นอีก 2 การทดลอง คือ การทดลองโดยใช้การควบคุมแบบลำดับขั้น (If clause) และการทดลองโดยใช้การควบคุมแบบพีไอดี ส่วนการทดลองแบบวิ่งเข้าโค้งหักศอก แบ่งออกเป็นอีก 2 การทดลองเช่นเดียวกัน คือ การทดลองโดยใช้การควบคุมแบบพีไอดี และการทดลองโดยใช้การควบคุมแบบพีไอดีและใช้การควบคุมแบบลำดับขั้นเข้าช่วย รายละเอียดต่าง ๆ สามารถอธิบายได้ดังนี้

4.2.1 การทดลองการทำงานของรถลำเลียงวัตถุดิบแบบวิ่งทางปกติ

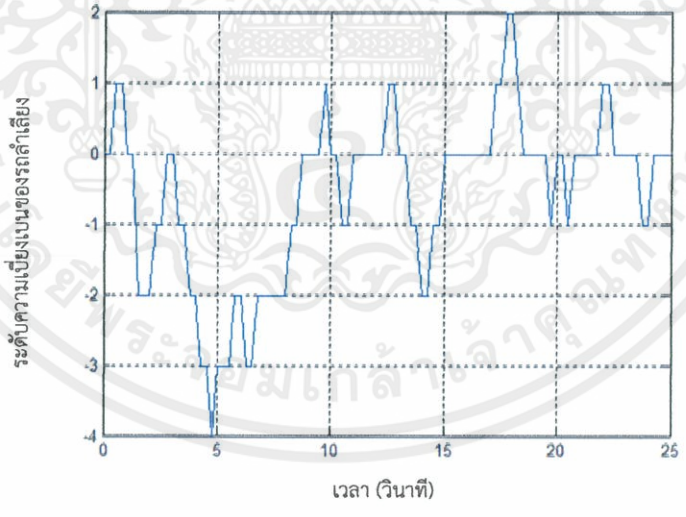
การทดลองการทำงานของรถลำเลียงวัตถุดิบแบบวิ่งทางปกติ เป็นการทดลองของรถลำเลียงให้วิ่งในทางตรง โดยมีลักษณะเส้นทางการวิ่งดังเส้นสีแดง ในรูปที่ 4.7 และมีรายละเอียดการทดลองดังนี้



รูปที่ 4.7 เส้นทางการวิ่งแบบทางปกติ

4.2.1.1 การทดลองแบบวิ่งทางปกติโดยใช้การควบคุมแบบลำดับขั้น

การทดลองนี้ เป็นการทดลองการวิ่งของรถลำเลียงวัตถุดิบโดยใช้การควบคุมแบบลำดับขั้น ซึ่งมีลำดับการทำงานดังนี้ เมื่อออพติคัลเซนเซอร์อ่านค่าได้ระดับความเบี่ยงเบนแล้วจึงตรวจสอบว่าตรงกับเงื่อนไขใดของโปรแกรม จากนั้นจึงสั่งให้มอเตอร์หมุนด้วยความเร็วต่าง ๆ โดยรายละเอียดของโปรแกรมแสดงไว้ในภาคผนวก ข.1 ซึ่งผลการทดลองได้ความสัมพันธ์ระหว่างระดับความเบี่ยงเบนของรถลำเลียงกับเวลาได้ดังรูปที่ 4.8



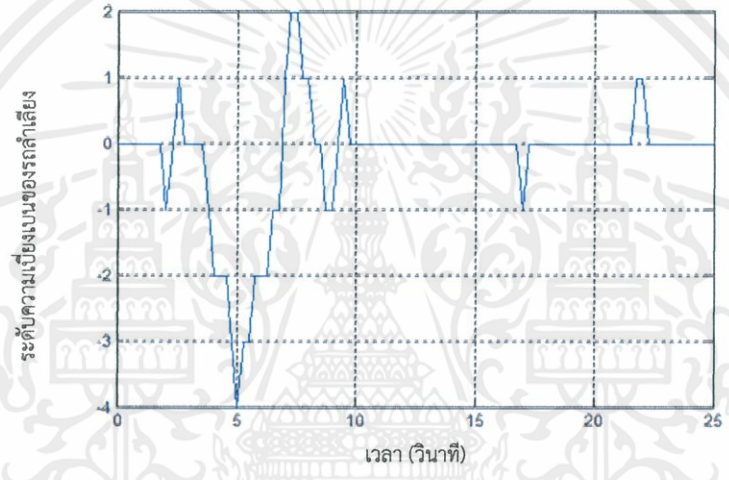
รูปที่ 4.8 ระดับความเบี่ยงเบนของรถลำเลียงเมื่อใช้การควบคุมแบบลำดับขั้น (ทางปกติ)

จากรูปที่ 4.8 จะพบว่าในช่วงเวลาที่รถลำเลียงเข้าโค้ง ซึ่งตรงกับช่วงนี้เวลาประมาณ 3 ถึง 9 วินาทีรถลำเลียงสามารถเข้าโค้งได้สำเร็จและไม่หลุดออกจากแนวเส้นทางเดิน แต่ในช่วงเวลาที่ 6 ถึง

7 วินาทีจะพบว่ารถลำเลียงเข้าโค้งแบบส่ายไปส่ายมา และในช่วงเวลาที่ 10 ถึง 25 เป็นช่วงเวลาที่รถลำเลียงจะวิ่งตรงตามแนวเส้นทางการลำเลียง จะเห็นว่ารถลำเลียงวิ่งส่าย อันเนื่องมาจากการควบคุมแบบลำดับขั้นยังไม่มีประสิทธิภาพเพียงพอในการควบคุมรถลำเลียงในการวิ่งตามเส้นทางนี้ จึงได้นำเสนอการควบคุมแบบพีไอดีขึ้นมา ซึ่งจะกล่าวในหัวข้อถัดไป

4.2.1.2 การทดลองแบบวิ่งทางปกติโดยใช้การควบคุมแบบพีไอดี

การทดลองนี้ เป็นการทดลองการวิ่งของรถลำเลียงวัตถุดิบโดยใช้การควบคุมแบบพีไอดีในการวิ่งทางปกติ ซึ่งเป็นการเขียนโปรแกรมคำนวณความเร็วของมอเตอร์แต่ละข้างของล้อรถลำเลียงโดยอาศัยระดับความเบี่ยงเบนที่ได้จากออพติคัลเซนเซอร์ จากนั้นจึงนำค่าระดับความเบี่ยงเบนที่ได้ป้อนให้กับตัวควบคุมพีไอดี เพื่อคำนวณสัญญาณควบคุมมอเตอร์ของรถลำเลียง และนำค่าดังกล่าวไปขับเคลื่อนล้อรถลำเลียงต่อไป โดยรายละเอียดของโปรแกรมแสดงไว้ในภาคผนวก ข.2 ซึ่งผลการทดลองได้ความสัมพันธ์ระหว่างระดับความเบี่ยงเบนของรถลำเลียงกับเวลาได้ดังรูปที่ 4.9

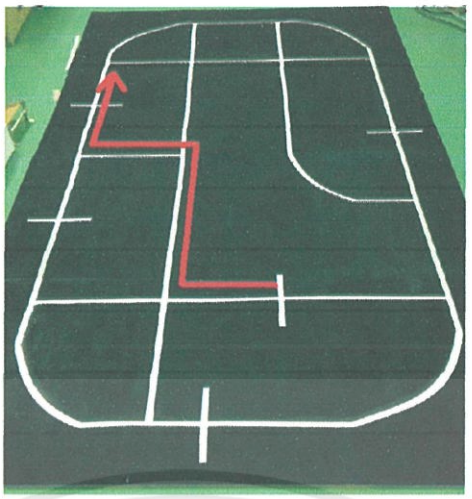


รูปที่ *4.9 ระดับความเบี่ยงเบนของรถลำเลียงเมื่อการควบคุมแบบพีไอดี (ทางปกติ)*

จากรูปที่ 4.9 จะพบว่าในช่วงเวลาที่รถลำเลียงเข้าโค้ง ซึ่งจะตรงกับเวลาประมาณ 3 ถึง 8 วินาทีนั้นสามารถเข้าโค้งได้เร็วกว่าการควบคุมแบบลำดับขั้น แต่ในช่วงเวลาที่ 8 ถึง 10 วินาที รถลำเลียงจะส่ายไปส่ายมา เนื่องจากต้องการปรับตัวรถลำเลียงให้เข้าสู่สภาวะสมดุล และในช่วงเวลาที่ 10 ถึง 25 วินาที รถลำเลียงสามารถวิ่งตามเส้นทางที่กำหนดไว้ได้ และไม่ค่อยมีการส่ายแบบการควบคุมแบบลำดับขั้น ดังนั้นในการทดลองวิ่งทางตรง การควบคุมแบบพีไอดีจึงมีประสิทธิภาพในการควบคุมมากกว่าการควบคุมแบบลำดับขั้น

4.2.2 การทดลองการทำงานของรถลำเลียงวัตถุดิบแบบวิ่งเข้าโค้งหักศอก

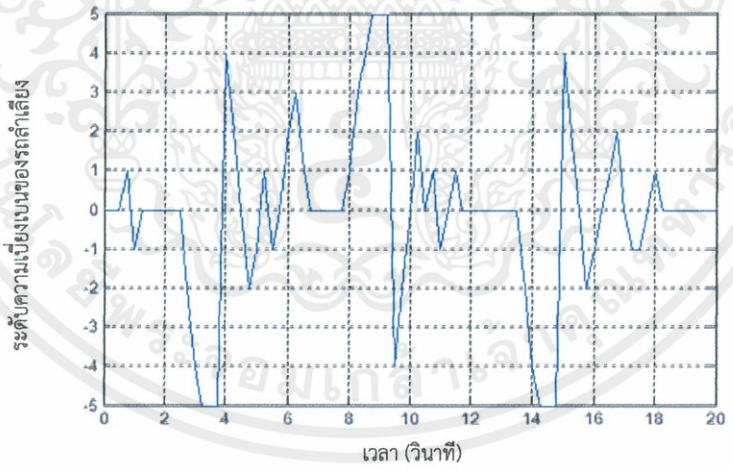
การทดลองการทำงานของรถลำเลียงวัตถุดิบแบบวิ่งเข้าโค้งหักศอก เป็นการทดลองของรถลำเลียงให้วิ่งในลักษณะเข้าโค้ง โดยมีลักษณะเส้นทางการวิ่งดังเส้นสีแดง ในรูปที่ 4.10 และมีรายละเอียดการทดลองดังนี้



รูปที่ 4.10 เส้นทางการวิ่งแบบเข้าโค้งหักศอก

4.2.2.1 การทดลองแบบวิ่งเข้าโค้งหักศอกโดยใช้การควบคุมแบบพีไอดี

การทดลองนี้ เป็นการทดลองการวิ่งของรถลำเลียงวัตถุดิบโดยใช้การควบคุมแบบพีไอดีในการวิ่งเข้าโค้งหักศอก ซึ่งเป็นการเขียนโปรแกรมคำนวณความเร็วของมอเตอร์แต่ละข้างของล้อรถลำเลียงโดยอาศัยระดับความเบี่ยงเบนที่ได้จากออพติคัลเซนเซอร์ จากนั้นจึงนำค่าระดับความเบี่ยงเบนที่ได้ป้อนเข้ากับตัวควบคุมพีไอดี เพื่อคำนวณสัญญาณควบคุมมอเตอร์ของรถลำเลียง และนำค่าดังกล่าวไปขับเคลื่อนล้อรถลำเลียงต่อไป โดยรายละเอียดของโปรแกรมแสดงไว้ในภาคผนวก ข.3 ซึ่งผลการทดลองได้ความสัมพันธ์ระหว่างระดับความเบี่ยงเบนของรถลำเลียงกับเวลาได้ดังรูปที่ 4.11



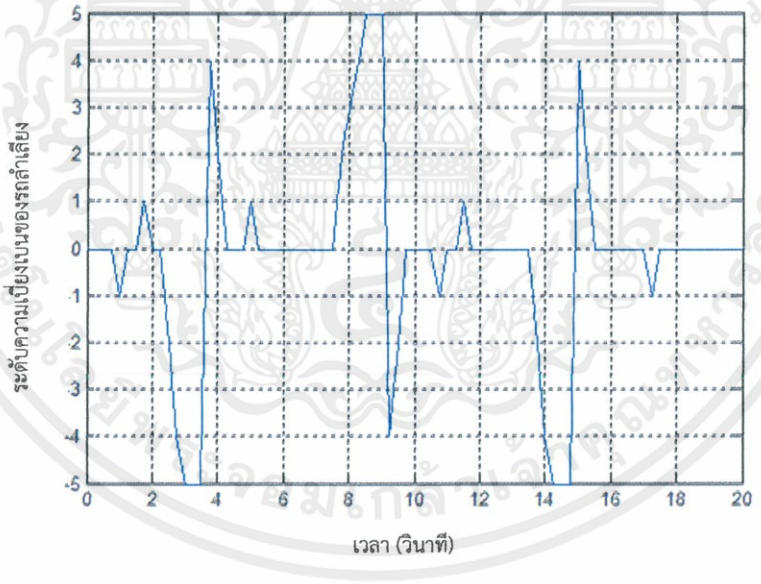
รูปที่ 4.11 ระดับความเบี่ยงเบนของรถลำเลียงเมื่อการควบคุมแบบพีไอดี (เข้าโค้งหักศอก)

จากรูปที่ 4.11 จะพบว่าในช่วงเวลาที่ 3 ถึง 4 วินาทีรถลำเลียงจะเลี้ยวโค้งหักศอกทางขวา ซึ่งเป็นโค้งหักศอกแรก หลังจากนั้นในช่วงเวลาที่ 4 ถึง 6 วินาที รถลำเลียงจะพยายามปรับตัวเองเพื่อให้เข้าสภาวะสมดุลและวิ่งต่อไปโดยที่ไม่เบี่ยงเบนออกจากแนวเส้นทางเดิน ในช่วงเวลาที่ 8 ถึง 9

วินาที รถลำเลียงจะเลี้ยวโค้งหักศอกทางซ้าย ซึ่งเป็นโค้งหักศอกที่สอง หลังจากนั้นในช่วงเวลาที่ 9 ถึง 11 วินาที ก็จะพยายามปรับตัวเองให้เข้ากับเส้นทางการวิ่ง ในช่วงเวลาที่ 14 ถึง 15 วินาที รถลำเลียงจะเลี้ยวโค้งหักศอกทางขวา ซึ่งเป็นโค้งหักศอกที่สาม จากนั้นก็จะพยายามปรับตัวให้เข้าสู่สภาวะสมดุลเพื่อเดินตรงต่อไป ในการควบคุมแบบพีไอดีในการวิ่งเข้าโค้งหักศอก จะพบว่ารถลำเลียงใช้เวลานานเกินไปในการกลับเข้าสู่เส้นทางการลำเลียง ซึ่งโดยประมาณ 2 วินาที จึงได้มีการนำเสนอการควบคุมแบบพีไอดีร่วมกับการควบคุมแบบลำดับขั้น ซึ่งจะกล่าวต่อไปในหัวข้อถัดไป

4.2.2.2 การทดลองแบบวิ่งเข้าโค้งหักศอกโดยใช้การควบคุมแบบพีไอดีร่วมกับการควบคุมแบบลำดับขั้น

การทดลองนี้ เป็นการทดลองการวิ่งของรถลำเลียงวัตถุดิบโดยใช้การควบคุมแบบพีไอดีร่วมกับการควบคุมแบบลำดับขั้นในการวิ่งเข้าโค้งหักศอก ซึ่งเป็นการเขียนโปรแกรมคำนวณความเร็วของมอเตอร์แต่ละข้างของล้อรถลำเลียงโดยอาศัยระดับความเบี่ยงเบนที่ได้จากออพติคัลเซนเซอร์ จากนั้นจึงนำค่าระดับความเบี่ยงเบนที่ได้ป้อนเข้ากับตัวควบคุมพีไอดี เพื่อคำนวณสัญญาณควบคุมมอเตอร์ของรถลำเลียง และนำค่าดังกล่าวไปขับเคลื่อนล้อรถลำเลียงต่อไป แต่ในการทดลองนี้จะแตกต่างจากการทดลองที่ 4.2.2.1 คือ ในช่วงเวลาที่รถลำเลียงกำลังเลี้ยวหักศอก จะตรวจสอบเงื่อนไขการเลี้ยวของรถลำเลียงจนกว่ารถลำเลียงจะเข้าสู่แนวเส้นทางการลำเลียง จึงจะขับเคลื่อนรถลำเลียงด้วยตัวควบคุมพีไอดีต่อไป โดยรายละเอียดของโปรแกรมแสดงไว้ในภาคผนวก ข.4 ซึ่งผลการทดลองได้ความสัมพันธ์ระหว่างระดับความเบี่ยงเบนของรถลำเลียงกับเวลาได้ดังรูปที่ 4.12



รูปที่ 4.12 ระดับความเบี่ยงเบนของรถลำเลียงเมื่อการควบคุมแบบพีไอดีร่วมกับแบบลำดับขั้น (เข้าโค้งหักศอก)

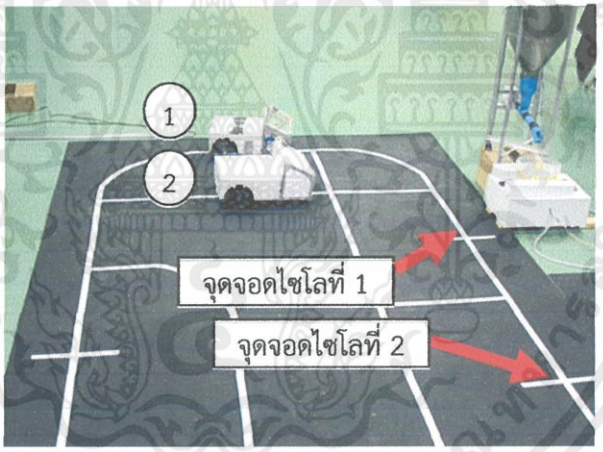
จากรูปที่ 4.12 จะพบว่าในช่วงเวลาที่ 3 ถึง 4 วินาทีรถลำเลียงจะเลี้ยวโค้งหักศอกทางขวา ซึ่งเป็นโค้งหักศอกแรก และมีการตรวจสอบเงื่อนไขจนกว่ารถลำเลียงจะเลี้ยวมาโดยที่ระดับความ

เบี่ยงเบนของรถลำเลียงเท่ากับศูนย์ จากนั้นจึงควบคุมรถลำเลียงด้วยตัวควบคุมพีไอดี ในช่วงเวลาที่ 8 ถึง 9 วินาที รถลำเลียงจะเลี้ยวโค้งหักศอกทางซ้าย ซึ่งเป็นโค้งหักศอกที่สอง และมีการตรวจสอบเงื่อนไขจนกว่ารถลำเลียงจะเลี้ยวมาโดยที่ระดับความเบี่ยงเบนของรถลำเลียงเท่ากับศูนย์ จากนั้นจึงควบคุมรถลำเลียงด้วยตัวควบคุมพีไอดี ในช่วงเวลาที่ 14 ถึง 15 วินาที รถลำเลียงจะเลี้ยวโค้งหักศอกทางขวา ซึ่งเป็นโค้งหักศอกที่สาม และมีการตรวจสอบเงื่อนไขจนกว่ารถลำเลียงจะเลี้ยวมาโดยที่ระดับความเบี่ยงเบนของรถลำเลียงเท่ากับศูนย์ จากนั้นจึงควบคุมรถลำเลียงด้วยตัวควบคุมพีไอดี ในการควบคุมแบบพีไอดีร่วมกับแบบลำดับขั้นในการวิ่งเข้าโค้งหักศอก พบว่ารถลำเลียงสามารถวิ่งตามเส้นแนวเส้นทางการเดินทางได้ และมีประสิทธิภาพในการเลี้ยวโค้งหักศอกมากกว่าการควบคุมแบบพีไอดีเพียงลำพัง

4.3 การทดลองการหลบเลี่ยงของรถลำเลียงวัตถุดิบ

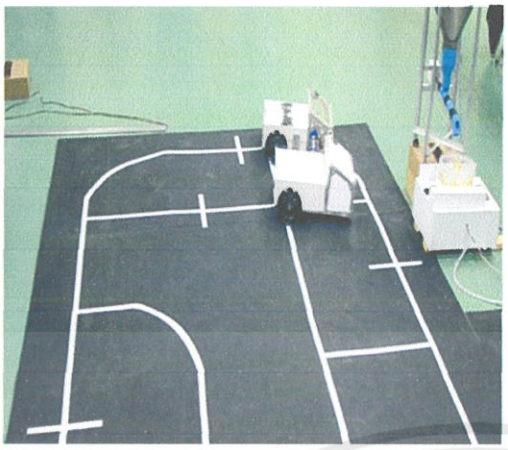
ในการทดลองนี้เป็นการทดลองการทำงานของรถลำเลียงวัตถุดิบจำนวน 2 คัน เพื่อให้เห็นถึงการหลบเลี่ยงกันระหว่างรถลำเลียงโดยใช้โมดูลซิกบีในการสื่อสาร ซึ่งสามารถแสดงผลการทดลองได้ดังนี้

เริ่มต้น กำหนดตำแหน่งรถลำเลียงวัตถุดิบทั้ง 2 คัน และกำหนดเป้าหมายที่ไซโลต่าง ๆ ตามต้องการ โดยในการทดลองนี้กำหนดให้รถลำเลียงคันที่ 1 ไปรับวัตถุดิบที่ไซโลที่ 1 และรถลำเลียงคันที่ 2 ไปรับวัตถุดิบที่ไซโลที่ 2 ซึ่งสามารถแสดงตำแหน่งเริ่มต้นได้ดังรูปที่ 4.13

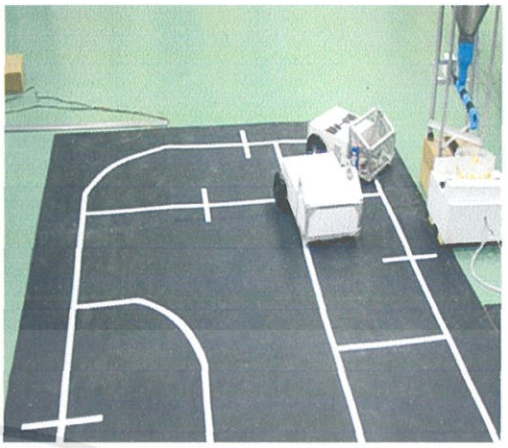


รูปที่ 4.13 ตำแหน่งเริ่มต้นของรถลำเลียงวัตถุดิบทั้ง 2 คัน

เมื่อได้รับคำสั่งจากโปรแกรมผู้ใช้งาน รถลำเลียงทั้ง 2 คัน ก็จะเริ่มวิ่งตามเส้นไปยังตำแหน่งที่กำหนดไว้ในข้างต้น และในขณะที่รถลำเลียงทั้ง 2 คัน กำลังวิ่งอยู่นั้น โมดูลซิกบีจะคอยสื่อสารระหว่างรถลำเลียงทั้ง 2 คัน เป็นการบอกตำแหน่งเพื่อไม่ให้เกิดการชนกันขึ้น สามารถแสดงได้ดังรูปที่ 4.14 (ก) และ 4.14 (ข)



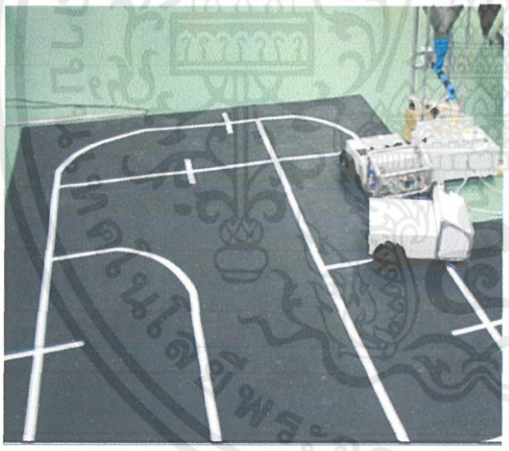
(ก) รถลำเลียงทั้ง 2 คัน วิ่งพร้อมกัน



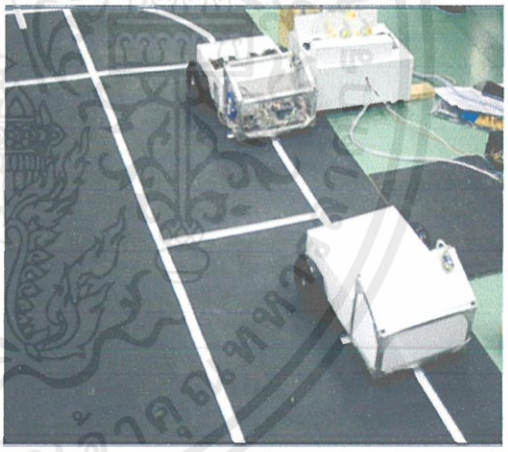
(ข) รถลำเลียงทั้ง 2 คัน วิ่งพร้อมกันและไม่ชนกัน

รูปที่ 4.14 การหลบเลี่ยงระหว่างรถลำเลียงวัตถุดิบทั้ง 2 คัน

เมื่อผ่านทางโค้งมาได้แล้ว รถลำเลียงทั้ง 2 คัน ก็จะวิ่งตามเส้นไปเรื่อย ๆ ถ้าเกิดเหตุการณ์ที่รถลำเลียงคันที่ 2 วิ่งอยู่นำหน้าใกล้ ๆ กับรถลำเลียงคันที่ 1 โมดูลซิกบีก็จะสื่อสารให้รถลำเลียงคันที่ 1 หยุดวิ่งก่อน ซึ่งสามารถแสดงได้ดังรูปที่ 4.15 (ก) และ 4.15 (ข)



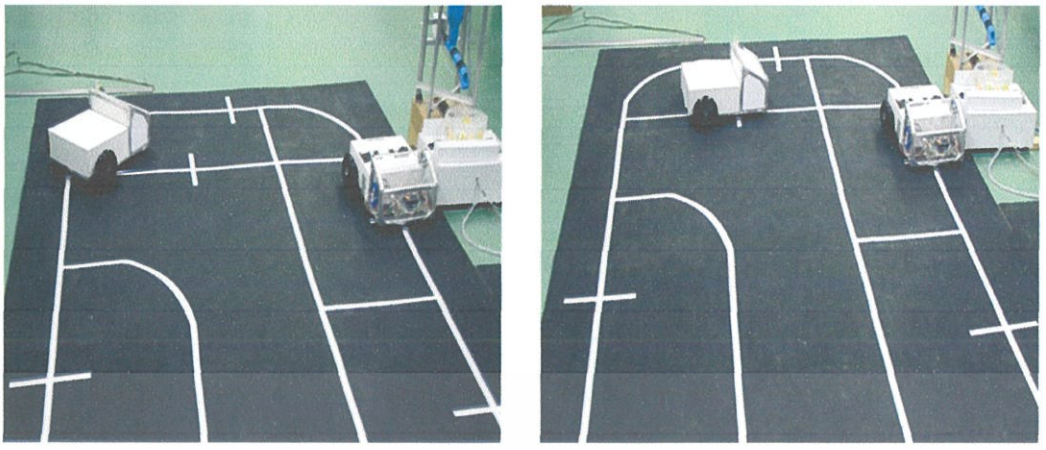
(ก) รถลำเลียงคันที่ 1 หยุดเพื่อให้รถลำเลียงคันที่ 2 ไปก่อน



(ข) รถลำเลียงคันที่ 2 วิ่งตามเส้นต่อไป

รูปที่ 4.15 การหลบเลี่ยงสำเร็จและการดำเนินงานขั้นต่อไป

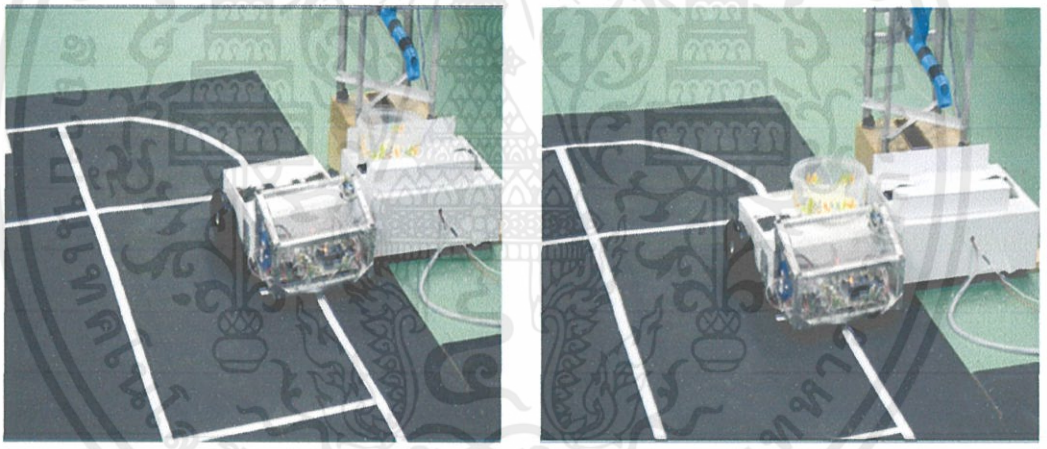
จากนั้นรถลำเลียงคันที่ 1 จะรอรับวัตถุดิบจากไซโลที่ 1 ส่วนรถลำเลียงคันที่ 2 ก็จะวิ่งจะตามเส้นไปเรื่อย ๆ จนกระทั่งกลับมาที่จุดจอดรถ ดังรูปที่ 4.16 (ก) และ 4.16 (ข)



(ก) รถลำเลียงคันที่ 2 กำลังวิ่งไปยังจุดจอดรถ (ข) รถลำเลียงคันที่ 2 หยุดจอดแล้ว

รูปที่ 4.16 รถลำเลียงคันที่ 2 กลับมายังจุดจอดรถ

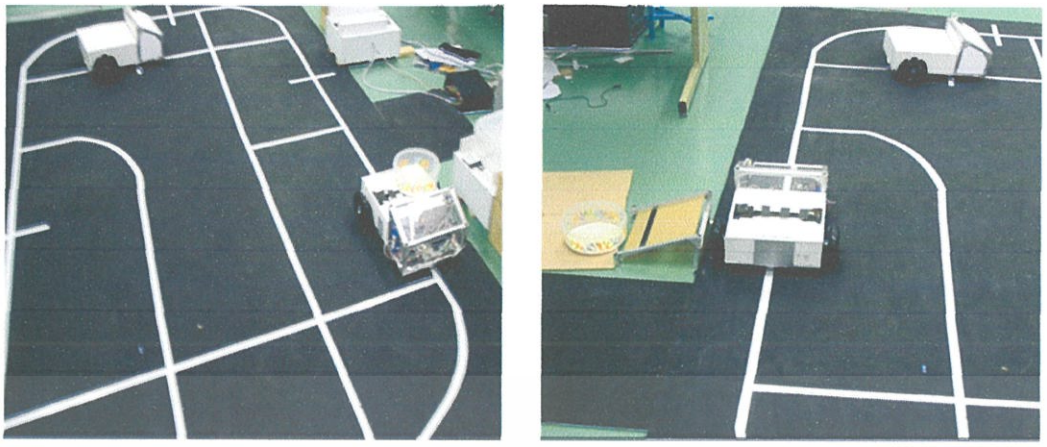
เมื่อวัตถุดิบที่จ่ายมายังภาชนะรองรับจากไซโลที่ 1 ถึงน้ำหนักที่กำหนดไว้แล้ว ส่วนลำเลียงจะส่งภาชนะรองรับมายังรถลำเลียงโดยใช้สายพานลำเลียง ดังรูปที่ 4.17 (ก) และ 4.17 (ข)



(ก) ส่วนลำเลียงกำลังส่งภาชนะรองรับวัตถุดิบ (ข) ภาชนะรองรับถูกส่งไปยังรถลำเลียงคันที่ 1

รูปที่ 4.17 การลำเลียงภาชนะรองรับจากส่วนลำเลียงมายังรถลำเลียงคันที่ 1

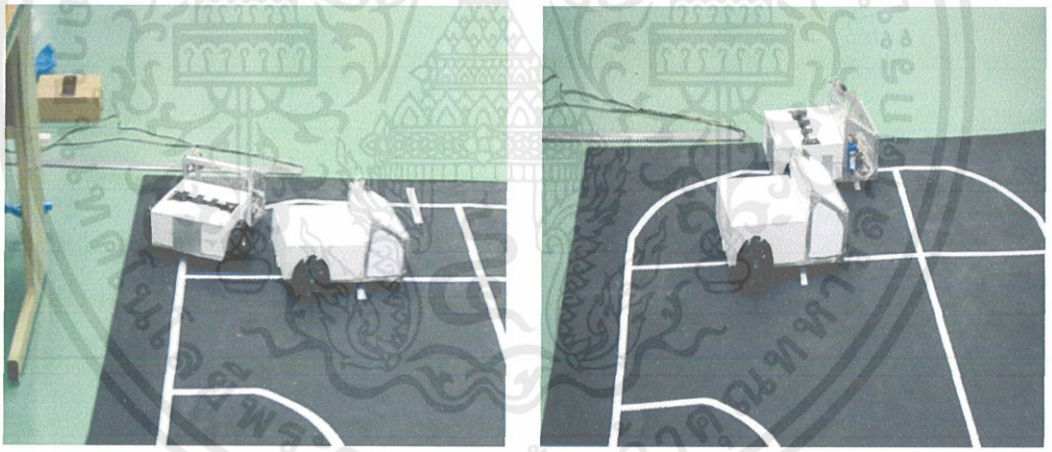
เมื่อรถลำเลียงคันที่ 1 ได้รับภาชนะรองรับแล้ว รถลำเลียงคันนี้ก็จะวิ่งตามเส้นต่อไปเพื่อไปยังจุดกักเก็บวัตถุดิบดังรูปที่ 4.18 (ก) เมื่อถึงตำแหน่งจุดกักเก็บวัตถุดิบแล้ว รถลำเลียงคันนี้จะหยุดจอดเพื่อลำเลียงภาชนะรองรับเข้าสู่จุดกักเก็บวัตถุดิบ จากนั้นสายพานที่อยู่บนรถลำเลียงจะทำงานขึ้นเพื่อส่งภาชนะรองรับเข้าจุดกักเก็บวัตถุดิบ ดังรูปที่ 4.18 (ข)



(ก) รถลำเลียงคันที่ 1 วิ่งไปยังจุดกักเก็บวัตถุดิบ (ข) รถลำเลียงคันที่ 1 หยุดจอดที่จุดกักเก็บวัตถุดิบ และส่งภาชนะเข้าจุดกักเก็บ

รูปที่ 4.18 รถลำเลียงคันที่ 1 ลำเลียงภาชนะรองรับไปยังจุดกักเก็บวัตถุดิบ

เมื่อภาชนะรองรับถูกส่งเข้าไปยังจุดกักเก็บเรียบร้อยแล้ว รถลำเลียงคันนี้ก็จะเดินทางกลับไปยังจุดจอดรถหรือจุดเริ่มต้นดังรูปที่ 4.19 (ก) และ 4.19 (ข) เป็นการสิ้นสุดการทำงาน



(ก) รถลำเลียงคันที่ 1 วิ่งไปยังจุดจอดรถ (ข) รถลำเลียงคันที่ 1 หยุดจอดแล้ว

รูปที่ 4.19 รถลำเลียงคันที่ 1 วิ่งกลับไปยังจุดจอดรถ

บทที่ 5
บทวิจารณ์และสรุปผล

5.1 สรุปการทดลอง

โครงงานนี้ ได้พัฒนาต้นแบบไซโลและระบบการจ่ายวัตถุดิบในส่วนโปรแกรมและระบบลำเลียง ซึ่งการออกแบบและทดลองแบ่งออกได้ 2 ส่วน คือ ในส่วนโปรแกรมเชื่อมต่อกับผู้ใช้งาน และในส่วนรถลำเลียงวัตถุดิบ

จากการทดลองในส่วนโปรแกรมเชื่อมต่อกับผู้ใช้งาน ซึ่งเป็นการสั่งปริมาณน้ำหนัก จำนวน และรถที่ใช้รับ-ส่งวัตถุดิบ พบว่า โปรแกรมที่พัฒนาขึ้น สามารถนำไปใช้งานได้จริง อีกทั้งยังมีการบันทึกผลจากการสั่งการในเบื้องต้น ทำให้สามารถตรวจสอบข้อมูลย้อนหลังได้หากพบข้อผิดพลาด ในส่วนของการบันทึกนั้นจะระบุ วันที่ เวลา ปริมาณน้ำหนัก จำนวน รถคันที่เลือกใช้ และชื่อของผู้ที่สั่งการ ซึ่งสามารถตรวจสอบการสั่งการย้อนหลังได้

จากการทดลองในส่วนรถลำเลียงวัตถุดิบ เบื้องต้นศึกษาเปรียบเทียบการวิ่งของรถลำเลียงในเส้นทางปกติโดยใช้การควบคุมแบบลำดับขั้นและการควบคุมแบบพีไอดี พบว่าการควบคุมแบบพีไอดีสามารถทำให้รถลำเลียงวิ่งได้โดยไม่หลุดแนวเส้นทางเดิน และมีการส่ายของรถลำเลียงน้อยกว่าการแบบควบคุมแบบลำดับขั้น และเมื่อนำโปรแกรมที่พัฒนาด้วยตัวควบคุมแบบพีไอดีร่วมกับการควบคุมแบบลำดับขั้นไปใช้ในการวิ่งเข้าโค้งหักศอกที่มีระยะทางน้อย พบว่าการควบคุมแบบดังกล่าว สามารถทำให้รถลำเลียงเลี้ยวหักศอกเข้าโค้งได้อย่างมีประสิทธิภาพ ไม่หลุดออกจากแนวเส้นทางการลำเลียง และไม่มีการส่ายของรถลำเลียงในการวิ่งในเส้นทางดังกล่าวมากนัก หลังจากนั้นจึงมีการทดลองวิ่งจริงของรถลำเลียงทั้งสองคัน พบว่ารถลำเลียงทั้งสองคันสามารถวิ่งตามเส้นทางที่ต้องการได้ โดยที่ไม่มีการชนกันเกิดขึ้น

5.2 ปัญหาที่พบและแนวทางแก้ไข

ในส่วนการพัฒนาโปรแกรมเชื่อมต่อกับผู้ใช้งาน เนื่องจากผู้พัฒนาโปรแกรมขาดประสบการณ์ในการออกแบบโปรแกรมเชื่อมต่อกับผู้ใช้งาน ซึ่งการออกแบบในช่วงแรกยังไม่สามารถตอบสนองความต้องการของผู้ใช้งานได้ จึงได้มีการปรับแต่งโปรแกรมและหน้าต่างเชื่อมต่อกับผู้ใช้งาน ซึ่งต้องใช้เวลาในการปรับแต่งมาก อีกทั้งในโปรแกรมเชื่อมต่อกับผู้ใช้งานจำเป็นต้องใช้ความรู้ในการเชื่อมต่อผ่านพอร์ตอนุกรม RS-232 การสร้างและเชื่อมต่อฐานข้อมูลของไมโครซอฟท์แอคเซส และการจัดเก็บไฟล์ข้อมูลในรูปแบบของไมโครซอฟท์เอคเซล ทำให้ต้องใช้เวลาในการศึกษาค้นคว้าพอสมควร

ในปัญหาส่วนการออกแบบรถลำเลียง เนื่องจากผู้จัดทำโครงงานขาดประสบการณ์ในการออกแบบเซนเซอร์รถลำเลียง ซึ่งในช่วงแรกได้ออกแบบออพติคัลเซนเซอร์ที่ใช้ในการตรวจจับการเบี่ยงเบนของรถลำเลียงไว้ 3 ตัว แต่เมื่อนำมาใช้งานจริง พบว่าการใช้งานออพติคัลเซนเซนเซอร์ 3 ตัวนั้นน้อยเกินไป ไม่เพียงพอต่อการควบคุมรถลำเลียงให้เลี้ยวหักศอกได้ และยังทำให้รถลำเลียงวิ่งหลุดเส้นทางที่มีลักษณะเป็นโค้งบ่อยครั้ง จึงได้มีการเพิ่มออพติคัลเซนเซอร์เป็น 5 ตัว ซึ่งสามารถแก้ไขปัญหานี้ได้เป็นอย่างดี แต่เมื่อได้ทดลองให้รถลำเลียงเลี้ยวหักศอกตามจุดต่าง ๆ ก็พบปัญหาใหม่ คือ ไม่มีเซนเซอร์ตรวจจับแยกบริเวณจุดหมุนของรถลำเลียง จึงได้ทดลองใช้การหน่วงเวลา และพบว่าวิธี

ดังกล่าวไม่มีความแม่นยำเลย จึงได้เพิ่มออพติคัลเซนเซอร์ที่บริเวณจุดหมุนของรถลำเลียงอีก 2 ตัว ซึ่งทำให้รถลำเลียงสามารถเลี้ยวเข้าโค้งหักศอกได้แม่นยำมากยิ่งขึ้น

5.3 ข้อเสนอแนะและแนวทางในการค้นคว้าพัฒนา

ในโครงงานนี้ได้มีการบันทึกการสั่งงานต่าง ๆ จากโปรแกรมเชื่อมต่อกับผู้ใช้งานหลายอย่าง แต่ไม่มีการบันทึกค่าน้ำหนักจริงที่ได้จากการชั่งจากสเตรนเกจ จึงทำให้ไม่สามารถทราบได้ว่า ในการทำงานแต่ละครั้ง ระบบผิดพลาดไปมากน้อยเพียงใด และไม่สามารถตรวจสอบข้อผิดพลาดจากการชั่งน้ำหนักย้อนหลังได้อีกด้วย จึงต้องมีการส่งค่าน้ำหนักจริงที่ชั่งได้จากสเตรนเกจให้เก็บบันทึกไว้ในฐานข้อมูล เพื่อให้ตรวจสอบย้อนหลังได้

ในโครงงานนี้ได้มีการใช้ออพติคัลเซนเซอร์ในการระบุการเบี่ยงเบนของรถลำเลียง ซึ่งมีข้อจำกัดในการใช้งาน คือ ต้องมีการสร้างเส้นทางการลำเลียงขึ้นมา และหากมีการขยายระบบไซโลให้มีมากกว่าที่ใช้ในโครงงาน ก็จำเป็นจะต้องสร้างเส้นทางลำเลียงเพิ่มเติม ซึ่งปัญหาดังกล่าวสามารถแก้ไขและพัฒนาได้โดยเปลี่ยนวิธีการโดยการไม่ใช้ออพติคัลเซนเซอร์ และเปลี่ยนมาใช้เซนเซอร์ระบุตำแหน่ง ซึ่งจะสามารถบอกได้ว่ารถลำเลียง ณ ตำแหน่งใดและมีการใช้งานเอนโค้ดเดอร์ (Encoder) ช่วยในการตรวจสอบการเคลื่อนที่ของรถลำเลียง และอาจใช้เซนเซอร์เข็มทิศ (Compass) ช่วยในการตรวจสอบทิศทางของรถลำเลียง ซึ่งจะช่วยให้รถลำเลียงสามารถเคลื่อนที่ไปได้ทุกที่ และมีการเพิ่มระบบสั่งการจากโปรแกรมเชื่อมต่อกับผู้ใช้งานให้สามารถวางแผนเส้นทางรถลำเลียงในโปรแกรมได้อีกด้วย

ภาคผนวก

ภาคผนวก ก
โปรแกรมเชื่อมต่อกับผู้ใช้งานที่พัฒนาขึ้น

โปรแกรมไมโครซอฟต์วิชวลซีชาร์ป ซึ่งเขียนขึ้นเพื่อใช้สั่งการควบคุมการทำงาน การแสดงผล และการบันทึกผล โดยมีโค้ดโปรแกรมดังต่อไปนี้

ก.1 Code in Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using System.IO.Ports;
using System.Data.OleDb;
using System.IO;
using System.Reflection;
using Excel = Microsoft.Office.Interop.Excel;

namespace WindowsFormsApplication1
{
    public partial class radgel1 : Form
    {
        private Form2 frm;                                                  //Declare function and variable
        private OleDbConnection conn;
        private OleDbDataAdapter adapt;
        private int randomnumber(int min, int max          {
            Random rdm = new Random();
            return rdm.Next(min, max);                     }
        DateTime DT = DateTime.Now;
        bool check = false;
        bool conncect = false;
        string a1,a2,b1,b2,c1,c2,d1,d2,e1,e2,f1,f2,a3,b3,c3,d3,e3,f3,send,str,buff;

    private void dateTimePicker1_ValueChanged(object sender, EventArgs e) {}        // Call Datatimepicker
    private void Form1_Load(object sender, EventArgs e)          {                  //Form1 loaded
            labelshowlogin.Hide();
            SerialPort1.Close();
            buttonlogout.Hide();
            buttonOk.Hide();
            buttoncancel.Hide();
            buttonConfirm.Hide();
            labeltime.Hide();
            labelshowlogin1.Text = "Please Login";
            labelshowlogin3.Text = "Please Login";
            GB1.Hide();
            GB2.Hide();
            GB3.Hide();
            GB4.Hide();
            GB5.Hide();
            GB6.Hide();
            GBSelectSilo.Hide();
            GBRecord.Hide();
            GBUpdate.Hide();                                    }

     private void buttonlogin_Click(object sender, EventArgs e)     {              //Button login clicked event
          if (conncect == true)                  {
                OleDbDataReader dr = null;
                OleDbConnection objConn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\Zetro\Desktop\Interface
VC# Updated\24-6-2555\Login2.accdb");                                           //Import data from Database
                OleDbCommand objCmd = new OleDbCommand("SELECT * FROM Login WHERE Username = '" + this.Tbname.Text + "' AND Password = '"
+ this.Tbpassword.Text + "' ", objConn);
            objConn.Open();
            dr = objCmd.ExecuteReader();
            objCmd.Dispose();
            if (dr.Read() == true)                          {
                txtbox1.Text += "Successful Login" + Environment.NewLine;
                txtbox1.SelectionLength = 0;
                txtbox1.SelectionStart = txtbox1.Text.Length;
                txtbox1.ScrollToCaret();
                gblogin.Hide();
                labelshowlogin.Text = "Welcome" + " " + Tbname.Text;
                labelshowlogin.Show();
                buttonlogout.Show();
                labelshowlogin1.Text = "";
                labelshowlogin3.Text = "";
                labeltime.Show();
                labeltime.Text = DT.ToString();
                GBSelectSilo.Show();
                GBUpdate.Show();
                GBRecord.Show();                    }
            Else                                    {
```

```
            txtbox1.Text += "Cannot login With This Username" + Environment.NewLine;
            txtbox1.SelectionLength = 0;
            txtbox1.SelectionStart = txtbox1.Text.Length;
            txtbox1.ScrollToCaret();                    }
          dr.Dispose();
          objConn.Close();                }
        else                              {
          MessageBox.Show(this, "Setting Connection at the Setting Page first.", "Please Setting Communication",          MessageBoxButtons.OK,
MessageBoxIcon.Stop);
          conncect = false;                                                    }
    private void buttonexit_Click(object sender, EventArgs e)      {                    //Button exit clicked event
      this.Close();                                                }
    private void buttonlogout_Click(object sender, EventArgs e)    {                    // Button logout clicked event
      DataSet ds = new DataSet();
      gblogin.Show();
      labelshowlogin.Hide();
      buttonlogout.Hide();
      labeltime.Hide();
      Tbname.Text = "";
      Tbpassword.Text = "";
      buttonOk.Hide();
      buttoncancel.Hide();
      buttonConfirm.Hide();
      labelshowlogin1.Text = "Please Login";
      labelshowlogin3.Text = "Please Login";
      random1.Text = "wait";
      random2.Text = "wait";
      random3.Text = "wait";
      random4.Text = "wait";
      random5.Text = "wait";
      random6.Text = "wait";
      GBSelectSilo.Hide();
      GBRecord.Hide();
      GBUpdate.Hide();
      txtbox1.Text = "";
      txtbox1.Text += "Successful Logout" + Environment.NewLine;
      CB1.Checked = false;
      CB2.Checked = false;
      CB3.Checked = false;
      CB4.Checked = false;
      CB5.Checked = false;
      CB6.Checked = false;
      WeightKg1.Value = 0;
      WeightKg2.Value = 0;
      WeightKg3.Value = 0;
      WeightKg4.Value = 0;
      WeightKg5.Value = 0;
      WeightKg6.Value = 0;
      QuantityEa1.Value = 0;
      QuantityEa2.Value = 0;
      QuantityEa3.Value = 0;
      QuantityEa4.Value = 0;
      QuantityEa5.Value = 0;
      QuantityEa6.Value = 0;
      dataGridView1.DataSource = ds.Tables["res"];              }
    public void buttonOk_Click(object sender, EventArgs e)                {          //Button Ok clicked event
      if ((Wheat1.Checked == false) && (Wheat2.Checked == false))          {
        Random rdm = new Random();                                                   //Call random function
        int irdm1;
        irdm1 = rdm.Next(8, 10);
        random1.Text = irdm1.ToString();                                  }
      if ((Gel1.Checked == false) && (Gel2.Checked == false))              {
        Random rdm1 = new Random();
        int irdm2;
        irdm2 = rdm1.Next(8, 10);
        random2.Text = irdm2.ToString();                                  }
      random3.Text = "0";
      random4.Text = "0";
      random5.Text = "0";
      random6.Text = "0";
      txtValue1.Text = "";
      txtValue2.Text = "";
      txtValue3.Text = "";
      txtValue4.Text = "";
      txtValue5.Text = "";
      txtValue6.Text = "";
      txtValue7.Text = "";
      txtValue8.Text = "";
      if (Wheat1.Checked == true)
        random1.Text = "8";
      if (Wheat2.Checked == true)
        random1.Text = "9";
      if(Gel1.Checked == true)
        random2.Text = "8";
      if (Gel2.Checked == true)
        random2.Text = "9";
      a1 = WeightKg1.Value.ToString();
      b1 = WeightKg2.Value.ToString();
      c1 = WeightKg3.Value.ToString();
      d1 = WeightKg4.Value.ToString();
      e1 = WeightKg5.Value.ToString();
      f1 = WeightKg6.Value.ToString();
      a2 = QuantityEa1.Value.ToString();
      b2 = QuantityEa2.Value.ToString();
      c2 = QuantityEa3.Value.ToString();
      d2 = QuantityEa4.Value.ToString();
      e2 = QuantityEa5.Value.ToString();
      f2 = QuantityEa6.Value.ToString();
      c3 = random3.Text;
      d3 = random4.Text;
      e3 = random5.Text;
```

```
            f3 = random6.Text;
            if (a1 == "0")              a1 = "0.0";
            if (b1 == "0")              b1 = "0.0";
            if (c1 == "0")              c1 = "0.0";
            if (d1 == "0")              d1 = "0.0";
            if (e1 == "0")              e1 = "0.0";
            if (f1  == "0")             f1 = "0.0";
            if (((a1=="0.0")&&(a2=="0")&&(b1=="0.0")&&(b2=="0")&&(c1=="0.0")&&(c2=="0")&&(d1=="0.0")&&(d2=="0") &&(e1=="0.0")&&(e2=="0")&&
                (f1=="0.0")&&(f2=="0"))||((a2=="0")&&(b2=="0")&&(c3=="0")&&(d3=="0")&&(e3=="0")&&(f3=="0")))                {
              txtbox1.Text +="Please order first" + Environment.NewLine;
              txtbox1.SelectionLength = 0;
              txtbox1.SelectionStart = txtbox1.Text.Length;
              txtbox1.ScrollToCaret();                                  }
            else                                                        {
              txtbox1.Text += "Wheat Flour " + a1 + " kg/" + a2 + " Ea, " + "Gel " + b1 + " kg/" + b2 + " Ea, " + "Rice Flour " + c1 + " kg/" + c2 + " Ea, " + "Corn
Strach " + d1 + " kg/" + d2 + " Ea, " + "Glutinous Flour " + e1 + " kg/" + e2 + " Ea, " + "Fiber " + f1 + " kg/" + f2 + " Ea," + Environment.NewLine + "Confirm or
Cancel?" + Environment.NewLine;
              txtbox1.SelectionLength =0;
              txtbox1.SelectionStart = txtbox1.Text.Length;
              txtbox1.ScrollToCaret();
              check = true;                                             }}
        private void buttonConfirm_Click(object sender, EventArgs e)           {           //Button Confirm clicked event
            if (check == true)                                          {
              a3 = random1.Text;
              b3 = random2.Text;
              c3 = random3.Text;
              d3 = random4.Text;
              e3 = random5.Text;
              f3 = random6.Text;
              if ((random1.Text == "8")&&(random2.Text == "8"))          {
                 random2.Text = "9";
                 b3 = random2.Text;                                     }
              if ((random1.Text == "9") && (random2.Text == "9"))       {
                 random2.Text = "8";
                 b3 = random2.Text;                                     }
              send = a1 + "," + a2 + "," + a3 + "," + b1 + "," + b2 + "," + b3 + "," + c1 + "," + c2 + "," + c3 + "," + d1 + "," + d2 + "," + d3 + "," + e1 + "," + e2 + "," +
                   e3 + "," + f1 + "," + f2 + "," + f3;
              SerialPort1.Write(send);                                  //Send data to serial port
              txtbox1.Text += "Order Complete"+ Environment.NewLine;
              txtbox1.SelectionLength = 0;
              txtbox1.SelectionStart = txtbox1.Text.Length;
              txtbox1.ScrollToCaret();
              DateTime DT = DateTime.Now;
              labelruntime.Text = DT.ToString();
              txtValue1.Text += a1 + "/" + a2;
              txtValue2.Text += b1 + "/" + b2;
              txtValue3.Text += c1 + "/" + c2;
              txtValue4.Text += d1 + "/" + d2;
              txtValue5.Text += e1 + "/" + e2;
              txtValue6.Text += f1 + "/" + f2;
              txtValue7.Text += labelruntime.Text;
              txtValue8.Text += Tbname.Text;
              frm = new Form2();
              frm.textBox1.Text = txtValue1.Text;
              frm.textBox2.Text = txtValue2.Text;
              frm.textBox3.Text = txtValue3.Text;
              frm.textBox4.Text = txtValue4.Text;
              frm.textBox5.Text = txtValue5.Text;
              frm.textBox6.Text = txtValue6.Text;
              frm.textBox7.Text = txtValue7.Text;
              frm.textBox8.Text = txtValue8.Text;
              frm.TTxx = this.test1.Text;
              frm.TTww = this.test2.Text;
              frm.ShowDialog();
              check = false;                                            }
            else                                                        {
              txtbox1.Text += "Please Order" + Environment.NewLine;
              txtbox1.SelectionLength =0;
              txtbox1.SelectionStart = txtbox1.Text.Length;
              txtbox1.ScrollToCaret();                                  }
            conn.Close();                                               }
        private void buttoncancel_Click(object sender, EventArgs e)            {           //Button Cancel clicked event
            txtbox1.Text +="Cancel Order" + Environment.NewLine;
            txtbox1.SelectionLength = 0;
            txtbox1.SelectionStart = txtbox1.Text.Length;
            txtbox1.ScrollToCaret();
            check = false;
            a1 = "0.0";
            a2 = "0";
            b1 = "0.0";
            b2 = "0";
            c1 = "0.0";
            c2 = "0";
            d1 = "0.0";
            d2 = "0";
            e1 = "0.0";
            e2 = "0";
            f1 = "0.0";
            f2 = "0";
            send = a1 + "," + a2 + "," + b1 + "," + b2 + "," + c1 + "," + c2 + "," + d1 + "," + d2 + "," + e1 + "," + e2 + "," + f1 + "," + f2 ;
            WeightKg1.Value = 0;
            WeightKg2.Value = 0;
            WeightKg3.Value = 0;
            WeightKg4.Value = 0;
            WeightKg5.Value = 0;
            WeightKg6.Value = 0;
            QuantityEa1.Value = 0;
            QuantityEa2.Value = 0;
            QuantityEa3.Value = 0;
            QuantityEa4.Value = 0;
            QuantityEa5.Value = 0;
```

```
        QuantityEa6.Value = 0;                                          }
    private void GbCommand_Enter(object sender, EventArgs e)  {}
    private void timer1_Tick(object sender, EventArgs e)          {}
    private void CB1_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 1 checked event
        if (CB1.Checked == true)                                    {
            GB1.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB1.Hide();
          if ((CB1.Checked == false) && (CB2.Checked == false) && (CB3.Checked == false) && (CB4.Checked == false) && (CB5.Checked == false) &&
(CB6.Checked == false))                                                 {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void CB2_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 2 checked event
        if (CB2.Checked == true)                                    {
            GB2.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB2.Hide();
          if ((CB1.Checked == false) && (CB2.Checked == false) && (CB3.Checked == false) && (CB4.Checked == false) && (CB5.Checked == false) &&
(CB6.Checked == false))                                                 {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void CB3_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 3 checked event
        if (CB3.Checked == true)                                    {
            GB3.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB3.Hide();
          if ((CB1.Checked == false) && (CB2.Checked == false) && (CB3.Checked == false) && (CB4.Checked == false) && (CB5.Checked == false) &&
(CB6.Checked == false))                                                 {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void CB4_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 4 checked event
        if (CB4.Checked == true)                                    {
            GB4.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB4.Hide();
          if ((CB1.Checked == false) && (CB2.Checked == false) && (CB3.Checked == false) && (CB4.Checked == false) && (CB5.Checked == false) &&
(CB6.Checked == false))                                                 {
            {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void CB5_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 5 checked event
        if (CB5.Checked == true)                                    {
            GB5.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB5.Hide();
          if ((CB1.Checked == false) && (CB2.Checked == false) && (CB3.Checked == false) && (CB4.Checked == false) && (CB5.Checked == false) &&
(CB6.Checked == false))                                                 {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void CB6_CheckedChanged(object sender, EventArgs e)          {          //Checkbox 6 checked event
        if (CB6.Checked == true)                                    {
            GB6.Show();
            buttonOk.Show();
            buttoncancel.Show();
            buttonConfirm.Show();                                   }
        else                                                        {
          GB6.Hide();
          if ((CB1.Checked == false)&&(CB2.Checked == false)&&(CB3.Checked == false)&&(CB4.Checked == false)&&(CB5.Checked ==
false)&&(CB6.Checked == false))                                         {
              buttonOk.Hide();
              buttoncancel.Hide();
              buttonConfirm.Hide();                                 }}}
    private void btnShowAll_Click(object sender, EventArgs e)     {                    //Button Showall clicked event
        string str1 = "select * from Table1";                                        //Show data in datagridview
        OleDbCommand cmd = new OleDbCommand(str1, conn);
        DataSet set = new DataSet();
        OleDbDataAdapter adapt = new OleDbDataAdapter(cmd);
        adapt.Fill(set, "res");
        dataGridView1.DataSource = set.Tables["res"];
        adapt.Dispose();
        cmd.Dispose();                                              }
    private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)  {}
    private void btnexcel_Click(object sender, EventArgs e)          {          //Button Save clicked event
        Excel.Application xlApp;                                                     //Export data to excel file
        Excel.Workbook xlWorkBook;
        Excel.Worksheet xlWorkSheet;
        object misValue = System.Reflection.Missing.Value;
        xlApp = new Excel.Application();
        xlWorkBook = xlApp.Workbooks.Add(misValue);
        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
        int i = 0;
```

```
        int j = 0;
        xlWorkSheet.Cells[1, 1] = "Date of order";
        xlWorkSheet.Cells[1, 2] = "Wheat Flour (Kg/Ea)";
        xlWorkSheet.Cells[1, 3] = "Gel (Kg/Ea)";
        xlWorkSheet.Cells[1, 4] = "Rice Flour (Kg/Ea)";
        xlWorkSheet.Cells[1, 5] = "Corn Starch (Kg/Ea)";
        xlWorkSheet.Cells[1, 6] = "Glutinous Flour (Kg/Ea)";
        xlWorkSheet.Cells[1, 7] = "Fiber (Kg/Ea)";
        xlWorkSheet.Cells[1, 8] = "1st car weight (Kg)";
        xlWorkSheet.Cells[1, 9] = "2nd car weight (Kg)";
        xlWorkSheet.Cells[1, 10] = "Order by";
        xlApp.Visible = true;
        for (i = 0; i <= dataGridView1.RowCount - 1; i++)            {
          for (j = 0; j <= dataGridView1.ColumnCount - 1; j++)       {
            DataGridViewCell cell = dataGridView1[j, i];
            xlWorkSheet.Cells[i + 2, j + 1] = cell.Value;            }}}
    private void buttonOkupdate_Click(object sender, EventArgs e)              {          //Button Ok clicked event
        OleDbConnection objConn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\Zetro\Desktop\Interface VC#
Updated\24-6-2555\Login2.accdb");
        objConn.Open();
        if (Tbupdatename.Text == string.Empty || Tbupdatepass.Text == string.Empty)          {
          MessageBox.Show("Please fillup complete");
          return;                                                                            }
        else                                                        {
          string str1 = string.Format("INSERT INTO Login VALUES ('{0}','{1}')", Tbupdatename.Text, Tbupdatepass.Text);
                    //Add new data to database
          OleDbCommand cmd = new OleDbCommand(str1, objConn);
          cmd.ExecuteNonQuery();
          cmd.Dispose();
          MessageBox.Show("Data stores successful");
          Tbupdatename.Text = "";
          Tbupdatepass.Text = "";
          radOperation.Checked = false;
          radSupervisor.Checked = false;                    }
        objConn.Close();                                            }
    private void buttoncancelUpdate_Click(object sender, EventArgs e)              {          //Button Cancel clicked event
        Tbupdatename.Text = "";
        Tbupdatepass.Text = "";
        radSupervisor.Checked = false;
        radOperation.Checked = false;                               }
    private void buttonSetport_Click(object sender, EventArgs e)                   {          //Button Setport clicked event
        bool error = false;
        if (SerialPort1.IsOpen) SerialPort1.Close();
        else                                                        {
          SerialPort1.BaudRate = int.Parse(CMBoxBuad.Text);                      // Set the port's settings
          SerialPort1.DataBits = int.Parse(CMBoxData.Text);
          SerialPort1.StopBits = (StopBits)Enum.Parse(typeof(StopBits),CMBoxStop.Text);
          SerialPort1.Parity = (Parity)Enum.Parse(typeof(Parity), CMBoxParity.Text);
          SerialPort1.PortName = CMBoxCOM.Text;
          Try                       {
            SerialPort1.Open();                                                          // Open the port
            conncect = true;
            MessageBox.Show(this,"Connect to RS232Successful","Connecting COMPORT",MessageBoxButtons.OK,
MessageBoxIcon.None);                   }
          catch (UnauthorizedAccessException) { error = true; }
          catch (IOException) { error = true; }
          catch (ArgumentException) { error = true; }
          if (error) MessageBox.Show(this, "Could not open the COM port.  Most likely it is already in use, has been removed, or is unavailable.", "COM Port
Unavailable", MessageBoxButtons.OK, MessageBoxIcon.Stop);     }}
    private void reset1_Click(object sender, EventArgs e)                          {          //Button Reset 1 clicked event
        WeightKg1.Value = 0;
        QuantityEa1.Value = 0;
        Wheat1.Checked = false;
        Wheat2.Checked = false;                                     }
    private void reset2_Click(object sender, EventArgs e)                          {          //Button Reset 2 clicked event
        WeightKg2.Value = 0;
        QuantityEa2.Value = 0;
        Gel1.Checked = false;
        Gel2.Checked = false;
        random2.Text = "0";                                         }
    private void reset3_Click(object sender, EventArgs e)                          {          //Button Reset 3 clicked event
        WeightKg3.Value = 0;
        QuantityEa3.Value = 0;
        Rice1.Checked = false;
        Rice2.Checked = false;                                      }
    private void reset4_Click(object sender, EventArgs e)                          {          //Button Reset 4 clicked event
        WeightKg4.Value = 0;
        QuantityEa4.Value = 0;
        Corn1.Checked = false;
        Corn2.Checked = false;                                      }
    private void reset5_Click(object sender, EventArgs e)                          {          //Button Reset 5 clicked event
        WeightKg5.Value = 0;
        QuantityEa5.Value = 0;
        Glu1.Checked = false;
        Glu2.Checked = false;                                       }
    private void reset6_Click(object sender, EventArgs e)                          {          //Button Reset 6 clicked event
        WeightKg6.Value = 0;
        QuantityEa6.Value = 0;
        Fiber1.Checked = false;
        Fiber2.Checked = false;                                     }
    private void Port1_DataReceived(object sender, SerialDataReceivedEventArgs e)    {      //Received data from serial port
        buff = SerialPort1.ReadExisting();
        string CAR;
        if (buff == 'a')
            CAR = "Going to Silo.";
        End if
        if (buff == 'b')
            CAR = "Stop at silo";
        End if
        if (buff == 'c')
            CAR = "Feeding material";
```

```
        End if
        if (buff == 'd')
            CAR = "Feed completed";
        End if
        if (buff == 'e')
            CAR = "Going to storage";
        End if
        if (buff == 'f')
            CAR = "Stop at storage";
        End if
        if (buff == 'g')
            CAR = "Sending material to storage";
        End if
        if (buff == 'h')
            CAR = "Going to stop point";
        End if
        if (buff == 'i')
            CAR = "Stop";
        End if
        this.test1.Text += CAR;             }
}}
```

ก.2 Code in Form2

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO.Ports;
using System.IO;
namespace WindowsFormsApplication1
{
    public partial class Form2 : Form
    {
        private OleDbConnection conn;
        string str;
        public Form2()                              {
            InitializeComponent();
            str = (@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\Zetro\Desktop\Interface VC# Updated\24-6-2555\ฐานข้อมูล41.accdb");
                                                    //Import data from database
            conn = new OleDbConnection(str);        }
        public string TTxx                          {
            get
                return this.car1status.Text;
            set
                this.car1status.Text = value;       }
        public string TTww                          {
            get
                return this.car1weight.Text;
            set
                this.car1weight.Text = value;       }
        private void Form2_Load(object sender, EventArgs e)     {}          //Form 2 loaded
        private void button1_Click(object sender, EventArgs e)  {           //Button Close clicked event
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            textBox8.Text = "";
            this.Close();                           }
        private void btnSave_Click(object sender, EventArgs e)  {           //Button Save clicked event
            string str1 = string.Format("INSERT INTO Table1 VALUES ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}')", textBox7.Text, textBox1.Text,
textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text, textBox6.Text, car1weight.Text, car2weight.Text,textBox8.Text );
            OleDbCommand cmd = new OleDbCommand(str1, conn);
            conn.Open();
            cmd.ExecuteNonQuery();
            MessageBox.Show("Data saved", "Save", MessageBoxButtons.OK,MessageBoxIcon.Information);
            cmd.Dispose();
            conn.Close();                           }}}
```

ภาคผนวก ข

โปรแกรมควบคุมการทำงานของรถลำเลียงวัตถุดิบ

โปรแกรมควบคุมการทำงานของรถลำเลียงวัตถุดิบ ใช้โปรแกรม Keil uVision 3 ในการเขียนโปรแกรม โดยมีโค้ดโปรแกรมดังต่อไปนี้

ข.1 โค้ดโปรแกรมของการทดลองที่ 4.2.1.1

```
int main{
while (1)     {
              if((ADC_Read[0]>1000)&&(ADC_Read[1]>1000)&&(ADC_Read[2]>1000)&&(ADC_Read[3]>1000)&&(ADC_Read[4]>1000))  {
                        Forward(1000,1000);
              else if((ADC_Read[2]>1000)&&(ADC_Read[3]>1000))
                        Forward(100,1000);
              else if((ADC_Read[3]>1000)&&(ADC_Read[0]>1000))
                        Forward(600,1000);
              else if((ADC_Read[0]>1000)&&(ADC_Read[1]>1000))
                        Forward(1000,600);
              else if((ADC_Read[1]>1000)&&(ADC_Read[4]>1000))
                        Forward(1000,100);
              else if(ADC_Read[2]>1000)
                        Forward(-200,1000);
              else if(ADC_Read[4]>1000)
                        Forward(1000,-200);
              else if(ADC_Read[3]>1000)
                        Forward(800,1000);
              else if(ADC_Read[1]>1000)
                        Forward(1000,800);
              else if(ADC_Read[0]>1000)
                        Forward(1000,1000);
              delay(10);                    }}
```

ข.2 โค้ดโปรแกรมของการทดลองที่ 4.2.1.2

```
void TIM2_IRQHandler(void)              {                                   // TIM2 Interupt PID Control
              if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)   {
                        TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
                        previous_error = setpoint - actual_position;
                        integral = 0;
                        errorr = setpoint - actual_position;
                        integral = integral + (errorr*11);
                        derivative = (errorr - previous_error)/11;
                        output = (Kp*errorr) + (Ki*integral) + (Kd*derivative);
                        previous_error = errorr;                    }}
int main(){
              actual_position = Check_Position();                           // Read Position of Optical Sensor
              if(output==0)                                                 // if Sensor 3 On
                        Forward(1000,1000);
              else if(output>0)                       {                     // Left
                        pwm1 = 1000 - output;
                        if(pwm1<201) {
                                  pwm1=-200; }
                        Forward(1000,pwm1);       }
              else if(output<0)                       {           // Right
                        pwm2 = 1000 + output;
                        if(pwm2<201)  {
                                  pwm2=-200; }
                        Forward(pwm2,1000);       }
                        delay(10);      }
```

ข.3 โค้ดโปรแกรมของการทดลองที่ 4.2.2.1

```
void TIM2_IRQHandler(void)              {                                   // TIM2 Interupt PID Control
              if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)   {
                        TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
                        previous_error = setpoint - actual_position;
                        integral = 0;
                        errorr = setpoint - actual_position;
                        integral = integral + (errorr*11);
                        derivative = (errorr - previous_error)/11;
                        output = (Kp*errorr) + (Ki*integral) + (Kd*derivative);
                        previous_error = errorr;                    }
int main(){
              actual_position = Check_Position();                           // Read Position of Optical Sensor
              if(cross == 1)                          {                     // Found first Cross
                        do                            {
```

```
                    Forward(700,700);                   // Forward 70% duty cycle
                    delay(10);                          // Delay 10 ms
                    }while(ADC_Read[2]<700);            // Wait untill Sensor right at wheel found the line
                    Clear();                            // Stop
                    delay(100);                         // Delay 100 ms
                    do          {
                                Forward(700,-700);  // Turn left with 70% duty cycle
                                delay(10);                  // Delay 10 ms
                                }while(ADC_Read[0]<700);    }           // Wait untill Sensor5 at the front of car found the line
else if(cross == 2)     {                               // Found Sencond Cross
          do            {
                    Forward(700,700);   // Forward 70% duty cycle
                    delay(10);                          // Delay 10 ms
                    }while(ADC_Read[1]<700);            // wait untill Sensor left at wheel found the line
                    Clear();                            // Stop
                    delay(100);                         // Delay 100 ms
                    do          {
                                Forward(-700,700);  // Turn Right with 70% duty cycle
                                delay(10);                  // Delay 10 ms
                                }while(ADC_Read[3]<700);  }  // wait untill Sensor1 at the front of car found the line
else if(cross == 3)     {
          do            {
                    Forward(700,700);                   // Forward 70% duty cycle
                    delay(10);                          // Delay 10 ms
                    }while(ADC_Read[2]<700);            // Wait untill Sensor right at wheel found the line
                    Clear();                            // Stop
                    delay(100);                         // Delay 100 ms
                    do          {
                                Forward(700,-700);          // Turn left with 70% duty cycle
                                delay(10);                  // Delay 10 ms
                                }while(ADC_Read[0]<700);  } // Wait untill Sensor5 at the front of car found the line
if(output==0)                                           // if Sensor 3 On
          Forward(1000,1000);
else if(output>0)       {                               // Left
          pwm1 = 1000 - output;
          if(pwm1<201)          {
                    pwm1=-200; }
          Forward(1000,pwm1);   }
else if(output<0)       {                               // Right
          pwm2 = 1000 + output;
          if(pwm2<201)          {
                    pwm2=-200; }
          Forward(pwm2,1000);   }
delay(10);  }
```

ข.4 โค้ดโปรแกรมของการทดลองที่ 4.2.2.2

```
void TIM2_IRQHandler(void)          {                   // TIM2 Interupt PID Control
          if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)  {
                    TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
                    previous_error = setpoint - actual_position;
                    integral = 0;
                    errorr = setpoint - actual_position;
                    integral = integral + (errorr*11);
                    derivative = (errorr - previous_error)/11;
                    output = (Kp*errorr) + (Ki*integral) + (Kd*derivative);
                    previous_error = errorr;                }
int main(){
          actual_position = Check_Position();               // Read Position of Optical Sensor
          if(cross == 1)            {                       // Found first Cross
                    do              {
                              Forward(700,700);             // Forward 70% duty cycle
                              delay(10);                    // Delay 10 ms
                              }while(ADC_Read[2]<700);      // Wait untill Sensor right at wheel found the line
                              Clear();                      // Stop
                              delay(100);                   // Delay 100 ms
                              do          {
                                        Forward(700,-700);          // Turn left with 70% duty cycle
                                        delay(10);                  // Delay 10 ms
                                        }while(ADC_Read[0]<700);    // Wait untill Sensor5 at the front of car found the line
                                        do          {
                                                  Forward(700,-700);  // Turn left with 70% duty cycle
                                                  delay(10);          // Delay 10 ms
                                        }while(ADC_Read[6]<700);    // Wait untill Sensor4 at the front of car found the line
                                        do
                                        {
                                                  Forward(700,-700);  // Turn left with 70% duty cycle
                                                  delay(10);          // Delay 10 ms
                                        }while(ADC_Read[5]<700);    }  // Wait untill Sensor3 at the front of car found the line
else if(cross == 2)     {                                 // Found Sencond Cross
          do            {
                    Forward(700,700);   // Forward 70% duty cycle
                    delay(10);          // Delay 10 ms
                    }while(ADC_Read[1]<700);    // wait untill Sensor left at wheel found the line
                    Clear();                    // Stop
                    delay(100);                 // Delay 100 ms
                    do          {
                                Forward(-700,700);  // Turn Right with 70% duty cycle
                                delay(10);          // Delay 10 ms
                                }while(ADC_Read[3]<700);    // wait untill Sensor1 at the front of car found the line
                    do          {
                                Forward(-700,700);  // Turn Right with 70% duty cycle
                                delay(10);          // Delay 10 ms
                                }while(ADC_Read[4]<700);    // wait untill Sensor2 at the front of car found the line
                                do          {
                                          Forward(-700,700);  // Turn Right with 70% duty cycle
                                          delay(10);          // Delay 10 ms
```

```
                                        }while(ADC_Read[4]<700);  }   // wait untill Sensor3 at the front of car found the line

        else if(cross == 3)         {
                        do          {
                                    Forward(700,700);                       // Forward 70% duty cycle
                                    delay(10);                              // Delay 10 ms
                                    }while(ADC_Read[2]<700);                // Wait untill Sensor right at wheel found the line
                                    Clear();                                // Stop
                                    delay(100);                             // Delay 100 ms
                                    do          {
                                                Forward(700,-700);  // Turn left with 70% duty cycle
                                                delay(10);          // Delay 10 ms
                                                }while(ADC_Read[0]<700);    // Wait untill Sensor5 at the front of car found the line
                                                do          {
                                                            Forward(700,-700);  // Turn left with 70% duty cycle
                                                            delay(10);          // Delay 10 ms
                                                }while(ADC_Read[6]<700);                // Wait untill Sensor4 at the front of car found the line
                                                do          {
                                                            Forward(700,-700);  // Turn left with 70% duty cycle
                                                delay(10);                              // Delay 10 ms
                                                }while(ADC_Read[5]<700);  }             // Wait untill Sensor3 at the front of car found the line
        if(output==0)                   // if Sensor 3 On
                        Forward(1000,1000);
        else if(output>0)                                                       // Left
        {
                        pwm1 = 1000 - output;
                        if(pwm1<201)                {
                                        pwm1=-200; }
                        Forward(1000,pwm1);         }
        else if(output<0)               {                                       // Right
                        pwm2 = 1000 + output;
                        if(pwm2<201)                {
                                        pwm2=-200; }
                        Forward(pwm2,1000);         }
        delay(10);                      }
```

ข.5 โค้ดโปรแกรมของรถลำเลียงวัตถุดิบคันที่ 1

```
#include "stm32f10x_lib.h"                                              // Include hex file
#include "string.h"
TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
TIM_OCInitTypeDef  TIM_OCInitStructure;
DMA_InitTypeDef DMA_InitStructure;
NVIC_InitTypeDef NVIC_InitStructure;
#define ADC1_DR_Address    ((u32)0x4001244C)                            // Address for connect DMA and ADC
vu16 ADC_Read[5];                                                       // Buffer ADC
u8 TxBuffer[100] = "ZZZZZ\r\n";
vu8 RxMessage[2] = {0};
unsigned int choice = 0;
unsigned int roundA = 0;
unsigned int silo = 0;
unsigned int ready = 0;
void RCC_Setup(void);
void GPIO_Setup(void);
void TIM4_Setup(void);
void TIM2_Setup(void);
void USART1_Setup(void);
void DMA_Setup(void);
void NVIC_Setup(void);
void USART2_Setup(void);
void ADC_Setup(void);
void TIM2_Setup(void);
void delay(unsigned long ms)            {                               // Delay 1 ms Function
        volatile unsigned long i,j;
        for(i=0; i<ms; i++)
        for(j=0; j<5525; j++);          }
int usart1_getc()                       {                               // Function receive charactor
        while(USART_GetFlagStatus(USART1,USART_FLAG_RXNE)==RESET);      // Wait untill receive data
        return(USART_ReceiveData(USART1));      }                       // Return charracter
void usart1_putc(unsigned char c)       {                               // Function transmit Charactor
        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);       // Wait untill transmittion ready
        USART_SendData(USART1,(int)c);          }                       // Send Charracter
void usart1_puts(unsigned char *s)      {               // Function USART1 transmit String
        while(*s)       {                                               // Check end of String
                        usart1_putc(*s++);          }}                  // Send charracter  1 time
int usart2_puts(unsigned char *s)       {                               // Function USART2 transmit String with DMA Module
        while(DMA_GetFlagStatus(DMA1_FLAG_TC7) == RESET);
        DMA_Cmd(DMA1_Channel7,DISABLE);
        strcpy(TxBuffer,s);
        DMA_DeInit(DMA1_Channel7);                      // Choose DMA1 to automatically transmits
        DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
        DMA_InitStructure.DMA_MemoryBaseAddr = (u32)TxBuffer;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
        DMA_InitStructure.DMA_BufferSize = strlen(TxBuffer);
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
        DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
        DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
        DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
        DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
        DMA_Init(DMA1_Channel7, &DMA_InitStructure);
        DMA_Cmd(DMA1_Channel7,ENABLE); }
int DMAChannel6_IRQHandler()            {                               // DMA Interupt Function
        if(DMA_GetFlagStatus(DMA1_IT_TC6)==SET)         {
                if(RxMessage[0]=='A')   {
                        choice =1;
                        delay(100);  }
```

```
                    if(RxMessage[0]=='B')     {
                              roundA = 1;
                              delay(100);   }
                    if(RxMessage[0]=='C')     {
                              roundA = 2;
                              delay(100);   }
                    if(RxMessage[0]=='D')     {
                              silo = 1;
                              delay(100);   }
                    if(RxMessage[0]=='E')     {
                              silo = 2;
                              delay(100);   }
                    if(RxMessage[0]=='G')     {
                              ready = 1;
                              delay(100);   }
                    DMA_ClearITPendingBit(DMA1_IT_TC6);  }}


void Forward(int a,int b)           {                       // This Function control speed wheel motor
          if((a>0)&&(b>0))                  {
                    GPIO_WriteBit(GPIOB,GPIO_Pin_10,0);                         // Direction Forward
                    GPIO_WriteBit(GPIOB,GPIO_Pin_11,1);                         // Direction Forward
                    GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);                         // Direction Forward
                    GPIO_WriteBit(GPIOB,GPIO_Pin_13,1);                         // Direction Forward
                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = a;                          // Speed Motor left
                    TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);

                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = b;                          // Speed Motor Right
                    TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);  }
          else if((a<0)&&(b>0))             {
                    a = -a;                                                     // Define direction is go left
                    GPIO_WriteBit(GPIOB,GPIO_Pin_10,0);                         // Direction Left
                    GPIO_WriteBit(GPIOB,GPIO_Pin_11,1);                         // Direction Left
                    GPIO_WriteBit(GPIOB,GPIO_Pin_12,1);                         // Direction Left
                    GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);                         // Direction Left
                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = a;                          // Speed Motor left
                    TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = b;                          // Speed Motor Right
                    TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);  }
          else if((a>0)&&(b<0))             {
                    b = -b;                                                     // Define direction is go right
                    GPIO_WriteBit(GPIOB,GPIO_Pin_10,1);                         // Direction Right
                    GPIO_WriteBit(GPIOB,GPIO_Pin_11,0);                         // Direction Right
                    GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);                         // Direction Right
                    GPIO_WriteBit(GPIOB,GPIO_Pin_13,1);                         // Direction Right
                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = a;                          // Speed Motor left
                    TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
                    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                    TIM_OCInitStructure.TIM_Pulse = b;                          // Speed Motor Right
                    TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                    TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);  }}
void Clear(void)                    {                                           // Function Stop Motion
          unsigned int a = 0;                                                   // Initial Stop
          unsigned int b = 0;                                                   // Initial Stop
          GPIO_WriteBit(GPIOB,GPIO_Pin_10,1);                                   // Direction Stop
          GPIO_WriteBit(GPIOB,GPIO_Pin_11,0);                                   // Direction Stop
          GPIO_WriteBit(GPIOB,GPIO_Pin_12,1);                                   // Direction Stop
          GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);                                   // Direction Stop
          TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
          TIM_OCInitStructure.TIM_Pulse = a;                                    // Speed Motor left
          TIM_OC3Init(TIM4, &TIM_OCInitStructure);
          TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
          TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
          TIM_OCInitStructure.TIM_Pulse = b;                                    // Speed Motor Right
          TIM_OC4Init(TIM4, &TIM_OCInitStructure);
          TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);  }
void Feed_In(void)                  {                                           // Feed in raw material
          delay(1000);                                                          // Delay 1 s
          GPIO_WriteBit(GPIOC,GPIO_Pin_5,0);
          GPIO_WriteBit(GPIOC,GPIO_Pin_6,1);
          delay(5000);                                                          // Delay 5 s
          GPIO_WriteBit(GPIOC,GPIO_Pin_5,0);
          GPIO_WriteBit(GPIOC,GPIO_Pin_6,0);  }
void Feed_Out(void)                 {                                           // Feed Out raw material
          delay(1000);                                                          // Delay 1 s
          GPIO_WriteBit(GPIOC,GPIO_Pin_5,1);
          GPIO_WriteBit(GPIOC,GPIO_Pin_6,0);
          delay(10000);                                                         // Delay 5 s
          GPIO_WriteBit(GPIOC,GPIO_Pin_5,0);
          GPIO_WriteBit(GPIOC,GPIO_Pin_6,0);  }
int Check_Position(void)            {                                           // Weight Optical Sensor function
          unsigned int i=0;
          if((ADC_Read[2]>900)&&(ADC_Read[3]>900)&&(ADC_Read[0]>900)&&(ADC_Read[1]>900)&&(ADC_Read[4]>900))          // 4Cross
                    i = 10;
          else if((ADC_Read[1]>900)&&(ADC_Read[2]>900)&&(ADC_Read[0]>900))                  // 3Cross
                    i = 0;
          else if((ADC_Read[0]>900)&&(ADC_Read[3]>900)&&(ADC_Read[4]>900))                  // 3Cross
                    i = 0;
          else if((ADC_Read[2]>900)&&(ADC_Read[3]>900))                                     // Sensor 1,2 On
                    i = 3;
          else if((ADC_Read[3]>900)&&(ADC_Read[0]>900))                                     // Sensor 2,3 On
```

```
                    i = 1;
            else if((ADC_Read[0]>900)&&(ADC_Read[1]>900))                    // Sensor 3,4 On
                    i = -1;
            else if((ADC_Read[1]>900)&&(ADC_Read[4]>900))                    // Sensor 4,5 On
                    i = -3;
            else if(ADC_Read[2]>900)                                         // Sensor 1 On
                    i = 4;
            else if(ADC_Read[3]>900)                                         // Sensor 2 On
                    i = 2;
            else if(ADC_Read[1]>900)                                         // Sensor 4 On
                    i = -2;
            else if(ADC_Read[4]>900)                                         // Sensor 5 On
                    i = -4;
            else if(ADC_Read[0]>900)                                         // Sensor 3 On
                    i = 0;
            return i;       }

int Transfer(unsigned int k)            {                                   // Function transfer hex data to dec data (0-9)
            if(k==0x30)
                    k = 0;
            if(k==0x31)
                    k = 1;
            if(k==0x32)
                    k = 2;
            if(k==0x33)
                    k = 3;
            if(k==0x34)
                    k = 4;
            if(k==0x35)
                    k = 5;
            if(k==0x36)
                    k = 6;
            if(k==0x37)
                    k = 7;
            if(k==0x38)
                    k = 8;
            if(k==0x39)
                    k = 9;
            else
                    k = k;
            return k;       }
int inTransfer(unsigned int k)          {                                   // Function transter dec data to hex data (0-9)
            if(k==0)
                    k = 0x30;
            if(k==1)
                    k = 0x31;
            if(k==2)
                    k = 0x32;
            if(k==3)
                    k = 0x33;
            if(k==4)
                    k = 0x34;
            if(k==5)
                    k = 0x35;
            if(k==6)
                    k = 0x36;
            if(k==7)
                    k = 0x37;
            if(k==8)
                    k = 0x38;
            if(k==9)
                    k = 0x39;
            else
                    k = k;
            return k;       }
int Show(unsigned long i)           }                                       // Function Show A2D sensor on USART1
            int a,b,c,d;
            if(i>=1000)     {
                    a = i/1000;
                    b = (i-(a*1000))/100;
                    c = (i-((a*1000)+(b*100)))/10;
                    d = (i-((a*1000)+(b*100)+(c*10)))/1;    }
            else if((1000>i)&&(i>=100))             {
                    a = 0;
                    b = i/100;
                    c = (i-(b*100))/10;
                    d = (i-((b*100)+(c*10)))/1;     }
            else if((100>i)&&(i>=10))               {
                    a = 0;
                    b = 0;
                    c = i/10;
                    d = (i-(c*10))/1;               }
            else if((10>i)&&(i>=0))                 {
                    a = 0;
                    b = 0;
                    c = 0;
                    d = i;                          }
            a = inTransfer(a);
            b = inTransfer(b);
            c = inTransfer(c);
            d = inTransfer(d);
            usart1_putc(a);
            usart1_putc(b);
            usart1_putc(c);
            usart1_putc(d);             }
void TIM2_IRQHandler(void)              }                                   // TIM2 Interupt PID Control
            if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)  {
                    TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
                    previous_error = setpoint - actual_position;
                    integral = 0;
                    errorr = setpoint - actual_position;
```

```
                integral = integral + (errorr*11);
                derivative = (errorr - previous_error)/11;
                output = (Kp*errorr) + (Ki*integral) + (Kd*derivative);
                previous_error = errorr;     }}
int main(void)          {                                                   // Main Function
        unsigned int CCR3_Val = 0;
        unsigned int CCR4_Val = 0;
        unsigned int cross = 0;
        unsigned int z=0;
        int setpoint = 0;
        int previous_error = 0;
        int actual_position = 0;
        int integral = 0;
        float derivative = 0;
        int errorr = 0;
        int Kp=100,Ki=10,Kd=0;                                              // Initial Kp,Ki,Kd
        float output = 0;
        int pwm1 = 0,pwm2 = 0;
        RCC_Setup();
        GPIO_Setup();
        USART1_Setup();
        NVIC_Setup();
        ADC_Setup();
        DMA_Setup();
        USART2_Setup();
        TIM4_Setup();
        TIM2_Setup();
/*      Show(ADC_Read[0]);                                                  // Show A2D Sensor
        usart1_puts("\r\n");
        Show(ADC_Read[1]);
        usart1_puts("\r\n");
        Show(ADC_Read[2]);
        usart1_puts("\r\n");
        Show(ADC_Read[3]);
        usart1_puts("\r\n");
        Show(ADC_Read[4]);
        usart1_puts("\r\n");
        usart1_puts("\r\n"); */
  while (1)     {                                                           // Loop Infinity
        if((choice!=0)&&(roundA!=0)&&(silo!=0)) {                           // Wait untill recieve data completely
                actual_position = Check_Position();                         // Read Position of Optical Sensor
                if(actual_position > 9)     {                               // When the truck find the cross
                        actual_position = 0;
                        cross = cross +1;                                   // Count Cross
                        Clear();                                            // Stop
                        delay(100);                                         // Delay 100 ms
                        if(silo==1)    {                                    // if go to silo 1
                                if(cross == 1) {                            // Cross = 1
                                        do {
                                            usart2_puts("F");               // Sent data 'F' to microcontroller at  silo dispensing system
                                            delay(100);                     // Delay 100 ms
                                        }while(ready!=1);                   // wait untill microcontoller at silo send Ready
                                            Feed_In();                      // Feed raw material in truck
                                        }
                                        else if(cross == 2)  {}             // Cross = 2
                                        else if(cross == 3)  {              // Cross = 3
                                            usart2_puts("H");               // Send  data 'H' to microsontroller at silo
                                            Feed_Out();    }                // Feed Out raw material
                                        else if(cross == 4)                 // Cross = 4
                                        {
                                            usart2_puts("I");               // Send data 'I' to microcontroller at silo
                                            z = z + 1;                      // Cont Round
                                            cross = 0;
                                            ready = 0;
                                            if(z==roundA)  {                // if Round = Command Stop
                                                    roundA = 0;
                                                    silo = 0;
                                                    choice = 0;
                                                    z = 0;  }}}
                        if(silo==2)    {                                    // Go to Silo 2
                                if(cross == 1)    {}                        // Cross = 1
                                else if(cross == 2)  {                      // Cross = 2
                                        do
                                        {
                                            usart2_puts("F");               // Send data 'F' to microcontroller
                                            at Silo dipensing system
                                            delay(100);                     // Delay 100 ms
                                        }while(ready!=1);                   // wait untill microcontroller at Silo send Ready
                                            Feed_In();  }                   // Feed in Raw material
                                        else if(cross == 3)  {              // Cross = 3
                                            usart2_puts("H");               // Send data 'H' to  microcontroller at Silo
                                            Feed_Out();   }                 // Feed out Raw material
                                        else if(cross == 4)  {              // Cross = 4
                                            usart2_puts("I");               // Send data 'I' to microcontroller at Silo
                                            z = z+1;                        // Count round
                                            cross = 0;
                                            ready = 0;
                                            if(z==roundA)  {                // If round = Command Stop
                                                    roundA  = 0;
                                                    choice = 0;
                                                    silo = 0;
                                                    z = 0;  }}}}
                if(output==0)                                               // if Sensor 3 On
                        Forward(1000,1000);
                else if(output>0)                   {                                   // Left
                        pwm1 = 1000 - output;
                        if(pwm1<201) {
                                pwm1=-200;  }
                        Forward(1000,pwm1);     }
                else if(output<0)                   {                       // Right
                        pwm2 = 1000 + output;
```

```
                                        if(pwm2<201)   {
                                                pwm2=-200;  }
                                        Forward(pwm2,1000);       }
                                delay(10);
                                Clear();  }}}
void RCC_Setup(void)            {                                                   // RCC Setup Function
        ErrorStatus HSEStartUpStatus;
        RCC_DeInit();
        RCC_HSEConfig(RCC_HSE_ON);
        HSEStartUpStatus = RCC_WaitForHSEStartUp();
        if (HSEStartUpStatus == SUCCESS)     {
                FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
                FLASH_SetLatency(FLASH_Latency_2);
                RCC_HCLKConfig(RCC_SYSCLK_Div1);
                RCC_PCLK2Config(RCC_HCLK_Div1);
                RCC_PCLK1Config(RCC_HCLK_Div4);
                RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
                RCC_PLLCmd(ENABLE);
                while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
                RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
                while (RCC_GetSYSCLKSource() != 0x08);  }
                RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_AFIO, ENABLE);     }
void GPIO_Setup(void)       {         // GPIO Setup Function
        GPIO_InitTypeDef GPIO_InitStructure;
        GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_8 | GPIO_Pin_9 ;                    // Enable Motor
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOB, &GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13 ;    // Direction Drive
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_Init(GPIOB,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6 ;                     // Direction Feed
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_Init(GPIOC,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 |GPIO_Pin_2 |GPIO_Pin_3 |GPIO_Pin_4 ;  // Optial Sensor
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
        GPIO_Init(GPIOC,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA,&GPIO_InitStructure);          }
void TIM2_Setup(void)     {                                                 // TIM2 Setup for PID Control
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
        TIM_TimeBaseStructure.TIM_Period = 9999;
        TIM_TimeBaseStructure.TIM_Prescaler = 72;
        TIM_TimeBaseStructure.TIM_ClockDivision =TIM_CKD_DIV1;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
        TIM_ARRPreloadConfig(TIM2, ENABLE);
        TIM_Cmd(TIM2, ENABLE);                  }
void TIM4_Setup(void)     {                                                 // TIM4 Setup for PWM motor
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
        TIM_TimeBaseStructure.TIM_Period = 999;
        TIM_TimeBaseStructure.TIM_Prescaler = 0;
        TIM_TimeBaseStructure.TIM_ClockDivision = 0;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
        TIM_ARRPreloadConfig(TIM4, ENABLE);
        TIM_Cmd(TIM4, ENABLE);                  }
void USART1_Setup(void)     {                                                   // USART 1 Setup
        USART_InitTypeDef USART_InitStructure;
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);                   // Enable USART1 Clock
        USART_InitStructure.USART_BaudRate = 9600;                              // Baudrate 9600
        USART_InitStructure.USART_WordLength = USART_WordLength_8b;             // Data 8 Bits
        USART_InitStructure.USART_StopBits = USART_StopBits_1;                  // 1 Bit Stop
        USART_InitStructure.USART_Parity = USART_Parity_No;                     // Parity none
        USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART_Init(USART1,&USART_InitStructure);                                // Configure the USART1
        USART_Cmd(USART1,ENABLE);       }
void NVIC_Setup()       {                                                       // Set TIM2 Interupt and DMA interupt
        NVIC_InitTypeDef NVIC_InitStructure;
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
        NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel6_IRQChannel;  // DMA interupt
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQChannel;                   // TIM2 interupt
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);         }
void USART2_Setup(void)     {                                                   // USART2 DMa function
        USART_InitTypeDef USART_InitStructure;
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);                   // Enable USART2 Clock
```

```
	USART_InitStructure.USART_BaudRate = 9600;                                  // USART2 configured
	USART_InitStructure.USART_WordLength = USART_WordLength_8b;
	USART_InitStructure.USART_StopBits = USART_StopBits_1;
	USART_InitStructure.USART_Parity = USART_Parity_No;
	USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
	USART_Init(USART2,&USART_InitStructure);                                    // Configure the USART2
	USART_DMACmd(USART2,USART_DMAReq_Rx | USART_DMAReq_Tx,ENABLE);
	USART_Cmd(USART2,ENABLE);       }                                           //  Enable USART2
void ADC_Setup(void)        {                                                   // ADC Setup 5 optical Sensor
	ADC_InitTypeDef ADC_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
	ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
	ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
	ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
	ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
	ADC_InitStructure.ADC_NbrOfChannel = 5;
	ADC_InitStructure.ADC_ScanConvMode = ENABLE;
	ADC_Init(ADC1, &ADC_InitStructure);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_13Cycles5);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 2, ADC_SampleTime_13Cycles5);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_12, 3, ADC_SampleTime_13Cycles5);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_13, 4, ADC_SampleTime_13Cycles5);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 5, ADC_SampleTime_13Cycles5);
	ADC_Cmd(ADC1, ENABLE);
	ADC_DMACmd(ADC1, ENABLE);
	ADC_ResetCalibration(ADC1);
	while(ADC_GetResetCalibrationStatus(ADC1));
	ADC_StartCalibration(ADC1);
	while(ADC_GetCalibrationStatus(ADC1));
	ADC_SoftwareStartConvCmd(ADC1,ENABLE);      }
void DMA_Setup(void)    {
	RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE);
	DMA_DeInit(DMA1_Channel1);                                                  // DMA Channel 1 for ADC
	DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&ADC1->DR;
	DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&ADC_Read;
	DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
	DMA_InitStructure.DMA_BufferSize = 5;
	DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
	DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
	DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
	DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
	DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
	DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
	DMA_Init(DMA1_Channel1, &DMA_InitStructure);
	DMA_DeInit(DMA1_Channel7);                                                  // DMA Channel 7 for USART2 Transmits
	DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
	DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&TxBuffer;
	DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
	DMA_InitStructure.DMA_BufferSize = strlen(TxBuffer);
	DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
	DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
	DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
	DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
	DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
	DMA_InitStructure.DMA_Priority = DMA_Priority_High;
	DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
	DMA_Init(DMA1_Channel7, &DMA_InitStructure);
	DMA_DeInit(DMA1_Channel6);                                                  // DMA Channel 6 for USART2 Recieve
	DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
	DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&RxMessage;
	DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
	DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
	DMA_InitStructure.DMA_BufferSize = 1;
	DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
	DMA_Init(DMA1_Channel6, &DMA_InitStructure);
	DMA_ITConfig(DMA1_Channel6,DMA_IT_TC,ENABLE);
	DMA_Cmd(DMA1_Channel1, ENABLE);
	DMA_Cmd(DMA1_Channel7, ENABLE);
	DMA_Cmd(DMA1_Channel6, ENABLE);             }
```

ข.6 โค้ดโปรแกรมของรถลำเลียงวัตถุดิบคันที่ 2

```
#include "stm32f10x_lib.h"                                                  // Include hex file
#include "string.h"
TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
TIM_OCInitTypeDef  TIM_OCInitStructure;
DMA_InitTypeDef DMA_InitStructure;
NVIC_InitTypeDef NVIC_InitStructure;
#define ADC1_DR_Address    ((u32)0x4001244C)                                // Address of ADC to connect DMA
vu16 ADC_Read[7];                                                           // ADC 7 ea
u8 TxBuffer[100] = "ZZZZZV\r\n";
vu8 RxMessage[2] = {0};
unsigned int choice = 0;
unsigned int roundA = 0;
unsigned int silo = 0;
unsigned int ready = 0;
void RCC_Setup(void);
void GPIO_Setup(void);
void TIM2_Setup(void);
void TIM4_Setup(void);
void USART1_Setup(void);
void DMA_Setup(void);
void NVIC_Setup(void);
void USART2_Setup(void);
void ADC_Setup(void);
void delay(unsigned long ms)        {                                       // Delay Function 1 ms for each loop
	volatile unsigned long i,j;
```

```
        for(i=0; i<ms; i++)
        for(j=0; j<5525; j++);      }
int usart1_getc()       {                                   // Function recieve character
        while(USART_GetFlagStatus(USART1,USART_FLAG_RXNE)==RESET);      // Wait untill receive data
        return(USART_ReceiveData(USART1));      }                       // Return charracter
void usart1_putc(unsigned char c)       {                               // Function transmit charactor
        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);       // Wait untill transmittion ready
        USART_SendData(USART1,(int)c);          }                       // Send Charracter
void usart1_puts(unsigned char *s)      {                               // Function transmit string
        while(*s)       {                                               // Check end of String
                usart1_putc(*s++);   }}                                 // Send charracter  1 time
int usart2_puts(unsigned char *s)       {                               // Function transmit charractor with DMA module
        while(DMA_GetFlagStatus(DMA1_FLAG_TC7) == RESET);
        DMA_Cmd(DMA1_Channel7,DISABLE);
        strcpy(TxBuffer,s);
        DMA_DeInit(DMA1_Channel7);                                      // DMA Channel 7
        DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
        DMA_InitStructure.DMA_MemoryBaseAddr = (u32)TxBuffer;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
        DMA_InitStructure.DMA_BufferSize = strlen(TxBuffer);
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
        DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
        DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
        DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
        DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
        DMA_Init(DMA1_Channel7, &DMA_InitStructure);
        DMA_Cmd(DMA1_Channel7,ENABLE);          }
int DMAChannel6_IRQHandler()    {                                       // Function recieve charractor interupt with DMA
        if(DMA_GetFlagStatus(DMA1_IT_TC6)==SET)         {
                if(RxMessage[0]=='a')       {
                        choice =1;
                        delay(100);   }
                if(RxMessage[0]=='b')       {
                        roundA = 1;
                        delay(100);   }
                if(RxMessage[0]=='c')       {
                        roundA = 2;
                        delay(100);   }
                if(RxMessage[0]=='d')       {
                        silo = 1;
                        delay(100);   }
                if(RxMessage[0]=='e')       {
                        silo = 2;
                        delay(100);   }
                if(RxMessage[0]=='g')       {
                        ready = 1;
                        delay(100);   }
                DMA_ClearITPendingBit(DMA1_IT_TC6);  }}
void Forward(int a,int b)       {                                       // Speed Motor Wheel in 2 direction
        if((a>0)&&(b>0))    {                                           // Forward
                GPIO_WriteBit(GPIOB,GPIO_Pin_10,0);
                GPIO_WriteBit(GPIOB,GPIO_Pin_11,1);
                GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);
                GPIO_WriteBit(GPIOB,GPIO_Pin_13,1);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = a;
                TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = b;
                TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);           }
        else if((a<0)&&(b>0))       {
                a = -a;                                                 // Left
                GPIO_WriteBit(GPIOB,GPIO_Pin_10,0);
                GPIO_WriteBit(GPIOB,GPIO_Pin_11,1);
                GPIO_WriteBit(GPIOB,GPIO_Pin_12,1);
                GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = a;
                TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = b;
                TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);           }
        else if((a>0)&&(b<0))       {
                b = -b;                                                 //Right
                GPIO_WriteBit(GPIOB,GPIO_Pin_10,1);
                GPIO_WriteBit(GPIOB,GPIO_Pin_11,0);
                GPIO_WriteBit(GPIOB,GPIO_Pin_12,0);
                GPIO_WriteBit(GPIOB,GPIO_Pin_13,1);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = a;
                TIM_OC3Init(TIM4, &TIM_OCInitStructure);
                TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
                TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                TIM_OCInitStructure.TIM_Pulse = b;
                TIM_OC4Init(TIM4, &TIM_OCInitStructure);
                TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);           }}
void Clear(void)    {                                                   // Stop motion
        unsigned int a = 0;
        unsigned int b = 0;
        GPIO_WriteBit(GPIOB,GPIO_Pin_10,1);
        GPIO_WriteBit(GPIOB,GPIO_Pin_11,0);
        GPIO_WriteBit(GPIOB,GPIO_Pin_12,1);
        GPIO_WriteBit(GPIOB,GPIO_Pin_13,0);
        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCInitStructure.TIM_Pulse = a;
```

```
	TIM_OC3Init(TIM4, &TIM_OCInitStructure);
	TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
	TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
	TIM_OCInitStructure.TIM_Pulse = b;
	TIM_OC4Init(TIM4, &TIM_OCInitStructure);
	TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);		}
int Check_Position(void)	{					// Check Position of 7 Optical Sensor
	unsigned int i=0;
	if((ADC_Read[3]>900)&&(ADC_Read[4]>900)&&(ADC_Read[5]>900)&&(ADC_Read[6]>900)&&(ADC_Read[0]>900))  // 4Cross
		i = 10;
	else if((ADC_Read[3]>900)&&(ADC_Read[4]>900)&&(ADC_Read[5]>900))		// 3Cross
		i = 10;
	else if((ADC_Read[5]>900)&&(ADC_Read[6]>900)&&(ADC_Read[0]>900))		// 3Cross
		i = 10;
	else if((ADC_Read[3]>900)&&(ADC_Read[4]>900))				// Sensor 1,2 On
		i = 3;
	else if((ADC_Read[0]>900)&&(ADC_Read[4]>900))				// Sensor 2,3 On
		i = 1;
	else if((ADC_Read[0]>900)&&(ADC_Read[6]>900))				// Sensor 3,4 On
		i = -1;
	else if((ADC_Read[0]>900)&&(ADC_Read[6]>900))				// Sensor 4,5 On
		i = -3;
	else if(ADC_Read[3]>900)						// Sensor 1 On
		i = 4;
	else if(ADC_Read[4]>900)						// Sensor 2 On
		i = 2;
	else if(ADC_Read[6]>900)						// Sensor 4 On
		i = -2;
	else if(ADC_Read[0]>900)						// Sensor 5 On
		i = -4;
	else if(ADC_Read[5]>900)						// Sensor 3 On
		i = 0;
	return i;	}
int Transfer(unsigned int k)	{					// Function transfer hex data to dec data (0-9)
	if(k==0x30)
		k = 0;
	if(k==0x31)
		k = 1;
	if(k==0x32)
		k = 2;
	if(k==0x33)
		k = 3;
	if(k==0x34)
		k = 4;
	if(k==0x35)
		k = 5;
	if(k==0x36)
		k = 6;
	if(k==0x37)
		k = 7;
	if(k==0x38)
		k = 8;
	if(k==0x39)
		k = 9;
	else
		k = k;
	return k;	}
int inTransfer(unsigned int k)	{					// Function transfer dec data to hex data (0-9)
	if(k==0)
		k = 0x30;
	if(k==1)
		k = 0x31;
	if(k==2)
		k = 0x32;
	if(k==3)
		k = 0x33;
	if(k==4)
		k = 0x34;
	if(k==5)
		k = 0x35;
	if(k==6)
		k = 0x36;
	if(k==7)
		k = 0x37;
	if(k==8)
		k = 0x38;
	if(k==9)
		k = 0x39;
	else
		k = k;
	return k;	}
int Show(unsigned long i)	{					// Function for read ADC with USART1
	int a,b,c,d;
	if(i>=1000)	{
		a = i/1000;
		b = (i-(a*1000))/100;
		c = (i-((a*1000)+(b*100)))/10;
		d = (i-((a*1000)+(b*100)+(c*10)))/1;	}
	else if((1000>i)&&(i>=100)) {
		a = 0;
		b = i/100;
		c = (i-(b*100))/10;
		d = (i-((b*100)+(c*10)))/1;  }
	else if((100>i)&&(i>=10))  {
		a = 0;
		b = 0;
		c = i/10;
		d = (i-(c*10))/1;
	else if((10>i)&&(i>=0))	{
		a = 0;
		b = 0;
```

```
                    c = 0;
                    d = i;          }
            a = inTransfer(a);
            b = inTransfer(b);
            c = inTransfer(c);
            d = inTransfer(d);
            usart1_putc(a);
            usart1_putc(b);
            usart1_putc(c);
            usart1_putc(d);             }
void TIM2_IRQHandler(void)              {                                   // TIM2 Interupt PID Control
            if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)  {
                    TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
                    previous_error = setpoint - actual_position;
                    integral = 0;
                    errorr = setpoint - actual_position;
                    integral = integral + (errorr*11);
                    derivative = (errorr - previous_error)/11;
                    output = (Kp*errorr) + (Ki*integral) + (Kd*derivative);
                    previous_error = errorr;    }}
int main(void)      {                                                       // Function Main
            unsigned int CCR3_Val = 0;
            unsigned int CCR4_Val = 0;
            unsigned int cross = 0;
            unsigned int z=0;
            int setpoint = 0;
            int previous_error = 0;
            int actual_position = 0;
            int integral = 0;
            float derivative = 0;
            int errorr = 0;
            int Kp=100,Ki=5,Kd=0;                                           // Initial Kp,Ki,Kd
            float output = 0;
            int pwm1 = 0,pwm2 = 0;
            RCC_Setup();
            GPIO_Setup();
            USART1_Setup();
            NVIC_Setup();
            ADC_Setup();
            DMA_Setup();
            USART2_Setup();
            TIM4_Setup();
            TIM2_Setup();
/*          Show(ADC_Read[0]);
            usart1_puts("\r\n");
            Show(ADC_Read[1]);
            usart1_puts("\r\n");
            Show(ADC_Read[2]);
            usart1_puts("\r\n");
            Show(ADC_Read[3]);
            usart1_puts("\r\n");
            Show(ADC_Read[4]);
            usart1_puts("\r\n");
            Show(ADC_Read[5]);
            usart1_puts("\r\n");
            Show(ADC_Read[6]);
            usart1_puts("\r\n");
            usart1_puts("\r\n");  */
while (1)   {                                                               // Infinity Loop
            if((choice!=0)&&(roundA!=0)&&(silo!=0))     {                   // Wait untill recieve data completely
                    actual_position = Check_Position();                     // Recieve Position of Optical Sensor
                    if(actual_position > 9)         {                       // if found 4Cross
                            actual_position = 0;
                            cross = cross +1;                               // Count Cross
                            Clear();                                        // Stop
                            delay(100);                                     // Delay 100 ms
                            if(silo==1)     {                               // If go to silo 1
                                    if(cross == 1)  {}
                                    else if(cross == 2)  {}
                                    else if(cross == 3)  {}
                                    else if(cross == 4)  {}}
                            if(silo==2)     {                               // If go to Silo 2
                                    if(cross == 1)     {                    // Found first Cross
                                            do  {
                                                Forward(700,700);           // Forward 70% duty cycle
                                                delay(10);                  // Delay 10 ms
                                            }while(ADC_Read[2]<700);        // Wait untill Sensor right at wheel found the line
                                                Clear();                    // Stop
                                                delay(100);                 // Delay 100 ms
                                            do  {
                                                Forward(700,-700);          // Turn left with 70% duty cycle
                                                delay(10);                  // Delay 10 ms
                                            }while(ADC_Read[0]<700);        // Wait untill Sensor5 at the front of car found the line
                                            do  {
                                                Forward(700,-700);          // Turn left with 70% duty cycle
                                                delay(10);                  // Delay 10 ms
                                            }while(ADC_Read[6]<700);        // Wait untill Sensor4 at the front of car found the line
                                            do  {
                                                Forward(700,-700);          // Turn left with 70% duty cycle
                                                delay(10);                  // Delay 10 ms
                                            }while(ADC_Read[5]<700);    }   // Wait untill Sensor3 at the front of car found the line
                                    else if(cross == 2)  {                  // Found Sencond Cross
                                            do  {
                                                Forward(700,700);           // Forward 70% duty cycle
                                                delay(10);                  // Delay 10 ms
                                            }while(ADC_Read[1]<700);        // wait untill Sensor left at wheel found the line
                                            Clear();                        // Stop
                                            delay(100);                     // Delay 100 ms
                                            do  {
                                                Forward(-700,700);          // Turn Right with 70% duty cycle
                                                delay(10);                  // Delay 10 ms
```

```
                    }while(ADC_Read[3]<700);          // wait untill Sensor1 at the front of car found the line
                    do {
                        Forward(-700,700);                  // Turn Right with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[4]<700);          // wait untill Sensor2 at the front of car found the line
                    do {
                        Forward(-700,700)                   // Turn Right with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[4]<700);    }     // wait untill Sensor3 at the front of car found the line
                else if(cross == 3) {
                    do {
                        Forward(700,700);                   // Forward 70% duty cycle
                              delay(10);                    // Delay 10 ms
                    }while(ADC_Read[2]<700);          // wait untill Sensor right  at wheel found the line
                    Clear();                                // Stop
                    delay(100);                             // Delay 100 ms
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[0]<700);          // Wait until Sensor5 at  the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[6]<700);         // Wait until Sensor4 at the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[5]<700);    }     // Wait untill Sensor3 at the front of car found the line
                else if(cross == 4) {
                    Clear();
                    delay(5000); }
                else if(cross == 5) {
                    do {
                        Forward(700,700);                   // Forward 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[2]<700);                // Wait until Sensor right  at wheel found the line
                    Clear();                        // Stop
                    delay(100);                     // Delay 100 ms
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);              // Delay 10 ms
                    }while(ADC_Read[0]<700);          // Wait until Sensor5 at the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                              delay(10);                    // Delay 10 ms
                    }while(ADC_Read[6]<700);          // Wait until Sensor4 at the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[5]<700);    }     // Wait until Sensor3 at  the front of car found the line
                else if(cross == 6) {}
                else if(cross == 7) {}
                else if(cross == 8) {
                    do {
                        Forward(700,700);                   // Forward 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[2]<700);                // Wait untill Sensor right at wheel found the line
                    Clear();                                // Stop
                    delay(100);                             // Delay 100 ms
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[0]<700);          // Wait untill Sensor5 at  the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[6]<700);          // Wait untill Sensor4 at the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[5]<700);    }     // Wait untill Sensor3 at the front of car found the line
                else if(cross == 9) {
                    Clear();
                    delay(5000); }
                else if(cross == 10) {}
                else if(cross == 11) {
                    do {
                        Forward(700,700);                   // Forward 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[2]<700);                // Wait untill Sensor right at wheel found the line
                    Clear();                                // Stop
                    delay(100);                             // Delay 100 ms
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[0]<700);          // Wait untill Sensor5 at the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[6]<700);          // Wait untill Sensor4 at  the front of car found the line
                    do {
                        Forward(700,-700);                  // Turn left with 70% duty cycle
                        delay(10);                          // Delay 10 ms
                    }while(ADC_Read[5]<700);    }     // Wait untill Sensor3 at the front of car found the line
                else if(cross == 12) {
                    while(1); }}

if(output==0)                                               // Forward
        Forward(1000,1000);
else if(output>0) {                                         // Left
        pwm1 = 1000 - output;
```

```
                                        if(pwm1<201)                {
                                                                    pwm1=-200; }
                                        Forward(1000,pwm1);         }
                                else if(output<0) {                                         // Right
                                        pwm2 = 1000 + output;
                                        if(pwm2<201)                {

                                                                    pwm2=-200; }
                                        Forward(pwm2,1000);         }
                                delay(10);
                                Clear();        }}}
void RCC_Setup(void)        {                                                               // RCC Setup
        ErrorStatus HSEStartUpStatus;
        RCC_DeInit();
        RCC_HSEConfig(RCC_HSE_ON);
        HSEStartUpStatus = RCC_WaitForHSEStartUp();
        if (HSEStartUpStatus == SUCCESS)    {
                FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
                FLASH_SetLatency(FLASH_Latency_2);
                RCC_HCLKConfig(RCC_SYSCLK_Div1);
                RCC_PCLK2Config(RCC_HCLK_Div1);
                RCC_PCLK1Config(RCC_HCLK_Div4);
                RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
                RCC_PLLCmd(ENABLE);
                while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
                RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
                while (RCC_GetSYSCLKSource() != 0x08); }
                RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_AFIO, ENABLE);    }
void GPIO_Setup(void)           {                                                           // GPIO Setup
        GPIO_InitTypeDef GPIO_InitStructure;
        GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_8 | GPIO_Pin_9 ;                            // Enable Motor
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOB, &GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13; // Direction  Drive
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_Init(GPIOB,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6  ;                            // Direction Feed
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_Init(GPIOC,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 |GPIO_Pin_2 |GPIO_Pin_3 |GPIO_Pin_4 ; // Optical Sensor
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
        GPIO_Init(GPIOC,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 ;                             // Sensor at the wheel
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
        GPIO_Init(GPIOB,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_Init(GPIOA,&GPIO_InitStructure);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA,&GPIO_InitStructure);               }
void TIM2_Setup(void)    {                                                                  // TIM2 Setup for PID Control
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
        TIM_TimeBaseStructure.TIM_Period = 9999;
        TIM_TimeBaseStructure.TIM_Prescaler = 72;
        TIM_TimeBaseStructure.TIM_ClockDivision =TIM_CKD_DIV1;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
        TIM_ARRPreloadConfig(TIM2, ENABLE);
        TIM_Cmd(TIM2, ENABLE);              }
void TIM4_Setup(void)    {                                                                  // TIM4 Setup for PWM motor
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
        TIM_TimeBaseStructure.TIM_Period = 999;
        TIM_TimeBaseStructure.TIM_Prescaler = 0;
        TIM_TimeBaseStructure.TIM_ClockDivision = 0;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
        TIM_ARRPreloadConfig(TIM4, ENABLE);
        TIM_Cmd(TIM4, ENABLE);              }
void USART1_Setup(void)    {                                        // USART 1 Setup
        USART_InitTypeDef USART_InitStructure;
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);                               // Enable USART1 Clock
        USART_InitStructure.USART_BaudRate = 9600;                                          // Baudrate 9600
        USART_InitStructure.USART_WordLength = USART_WordLength_8b;                         // Data 8 Bits
        USART_InitStructure.USART_StopBits = USART_StopBits_1;                              // 1 Bit Stop
        USART_InitStructure.USART_Parity = USART_Parity_No;                                 // Parity none
        USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART_Init(USART1,&USART_InitStructure);                                            // Configure the USART1
        USART_Cmd(USART1,ENABLE);          }
void NVIC_Setup()       {                                                                   // Set TIM2 Interupt and DMA interupt
        NVIC_InitTypeDef NVIC_InitStructure;
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
        NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel6_IRQChannel;   // DMA interupt
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
```

```
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQChannel;                    // TIM2 interupt
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);          }
void USART2_Setup(void)       {                                          // USART2 DMa function
        USART_InitTypeDef USART_InitStructure;
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);            // Enable USART2 Clock
        USART_InitStructure.USART_BaudRate = 9600;                       // USART2 configured
        USART_InitStructure.USART_WordLength = USART_WordLength_8b;
        USART_InitStructure.USART_StopBits = USART_StopBits_1;
        USART_InitStructure.USART_Parity = USART_Parity_No;
        USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART_Init(USART2,&USART_InitStructure);                         // Configure the USART2
        USART_DMACmd(USART2,USART_DMAReq_Rx | USART_DMAReq_Tx,ENABLE);
        USART_Cmd(USART2,ENABLE);        }                               //  Enable USART2

void ADC_Setup(void)        {                                            // ADC Setup for 7 sensor
        ADC_InitTypeDef ADC_InitStructure;
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
        ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
        ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
        ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
        ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
        ADC_InitStructure.ADC_NbrOfChannel = 7;
        ADC_InitStructure.ADC_ScanConvMode = ENABLE;
        ADC_Init(ADC1, &ADC_InitStructure);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 2, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_12, 3, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_13, 4, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 5, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 6, ADC_SampleTime_13Cycles5);
        ADC_RegularChannelConfig(ADC1, ADC_Channel_9, 7, ADC_SampleTime_13Cycles5);
        ADC_Cmd(ADC1, ENABLE);
        ADC_DMACmd(ADC1, ENABLE);
        ADC_ResetCalibration(ADC1);
        while(ADC_GetResetCalibrationStatus(ADC1));
        ADC_StartCalibration(ADC1);
        while(ADC_GetCalibrationStatus(ADC1));
        ADC_SoftwareStartConvCmd(ADC1,ENABLE);       }
void DMA_Setup(void)      {
        RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE);
        DMA_DeInit(DMA1_Channel1);                                       // DMA Channel 1 for ADC
        DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&ADC1->DR;
        DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&ADC_Read;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
        DMA_InitStructure.DMA_BufferSize = 5;
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
        DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
        DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
        DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
        DMA_Init(DMA1_Channel1, &DMA_InitStructure);
        DMA_DeInit(DMA1_Channel7);                                       // DMA Channel 7 for USART2 Transmits
        DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
        DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&TxBuffer;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
        DMA_InitStructure.DMA_BufferSize = strlen(TxBuffer);
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
        DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
        DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
        DMA_InitStructure.DMA_Priority = DMA_Priority_High;
        DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
        DMA_Init(DMA1_Channel7, &DMA_InitStructure);
        DMA_DeInit(DMA1_Channel6);                       // DMA Channel 6 for USART2 Recieve
        DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART2->DR;
        DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&RxMessage;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
        DMA_InitStructure.DMA_BufferSize = 1;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
        DMA_Init(DMA1_Channel6, &DMA_InitStructure);
        DMA_ITConfig(DMA1_Channel6,DMA_IT_TC,ENABLE);
        DMA_Cmd(DMA1_Channel1, ENABLE);
        DMA_Cmd(DMA1_Channel7, ENABLE);
        DMA_Cmd(DMA1_Channel6, ENABLE);          }
```

ภาคผนวก ค

คู่มือการใช้งานอุปกรณ์ในโครงงาน

ค.1 เอกสารคู่มือการใช้งานไมโครคอนโทรลเลอร์ ARM *STM32F103RET6*

STM32F103xC STM32F103xD STM32F103xE

High-density performance line ARM-based 32-bit MCU with 256 to 512KB Flash, USB, CAN, 11 timers, 3 ADCs, 13 communication interfaces

Features

- Core: ARM 32-bit Cortex™-M3 CPU
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 256 to 512 Kbytes of Flash memory
  - up to 64 Kbytes of SRAM
  - Flexible static memory controller with 4 Chip Select. Supports Compact Flash, SRAM, PSRAM, NOR and NAND memories
  - LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
  - 2.0 to 3.6 V application supply and I/Os
  - POR, PDR, and programmable voltage detector (PVD)
  - 4-to-16 MHz crystal oscillator
  - Internal 8 MHz factory-trimmed RC
  - Internal 40 kHz RC with calibration
  - 32 kHz oscillator for RTC with calibration
- Low power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC and backup registers
- 3 x 12-bit, 1 µs A/D converters (up to 21 channels)
  - Conversion range: 0 to 3.6 V
  - Triple-sample and hold capability
  - Temperature sensor
- 2 x 12-bit D/A converters
- DMA: 12-channel DMA controller
  - Supported peripherals: timers, ADCs, DAC, SDIO, I^2Ss, SPIs, I^2Cs and USARTs
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
  - Cortex-M3 Embedded Trace Macrocell™

- Up to 112 fast I/O ports
  - 51/80/112 I/Os, all mappable on 16 external interrupt vectors and almost all 5 V-tolerant
- Up to 11 timers
  - Up to four 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
  - 2 x 16-bit motor control PWM timers with dead-time generation and emergency stop
  - 2 x watchdog timers (Independent and Window)
  - SysTick timer: a 24-bit downcounter
  - 2 x 16-bit basic timers to drive the DAC
- Up to 13 communication interfaces
  - Up to 2 x I^2C interfaces (SMBus/PMBus)
  - Up to 5 USARTs (ISO 7816 interface, LIN, IrDA capability, modem control)
  - Up to 3 SPIs (18 Mbit/s), 2 with I^2S interface multiplexed
  - CAN interface (2.0B Active)
  - USB 2.0 full speed interface
  - SDIO interface
- CRC calculation unit, 96-bit unique ID
- ECOPACK® packages

Table 1. Device summary

Reference	Part number
STM32F103xC	STM32F103RC STM32F103VC STM32F103ZC
STM32F103xD	STM32F103RD STM32F103VD STM32F103ZD
STM32F103xE	STM32F103RE STM32F103ZE STM32F103VE

5.2 Absolute maximum ratings

Stresses above the absolute maximum ratings listed in *Table 7: Voltage characteristics*, *Table 8: Current characteristics*, and *Table 9: Thermal characteristics* may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 7. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}-V_{SS}	External main supply voltage (including V_{DDA} and V_{DD})(1)	-0.3	4.0	V
V_{IN}(2)	Input voltage on five volt tolerant pin	V_{SS} - 0.3	V_{DD} + 4.0	V
	Input voltage on any other pin	V_{SS} - 0.3	4.0	V
$\lvert\Delta V_{DDx}\rvert$	Variations between different V_{DD} power pins		50	mV
$\lvert V_{SSX} - V_{SS}\rvert$	Variations between all the different ground pins		50	mV
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 5.3.12: Absolute maximum ratings (electrical sensitivity)*		

1. All main power (V_{DD}, V_{DDA}) and ground (V_{SS}, V_{SSA}) pins must always be connected to the external power supply, in the permitted range.
2. V_{IN} maximum must always be respected. Refer to *Table 8: Current characteristics* for the maximum allowed injected current values.

Table 8. Current characteristics

Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD}/V_{DDA} power lines (source)(1)	150	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	150	mA
I_{IO}	Output current sunk by any I/O and control pin	25	mA
	Output current source by any I/Os and control pin	- 25	mA
$I_{INJ(PIN)}$(2)	Injected current on five volt tolerant pins(3)	-5/+0	mA
	Injected current on any other pin(4)	± 5	mA
$\Sigma I_{INJ(PIN)}$	Total injected current (sum of all I/O and control pins)(5)	± 25	mA

1. All main power (V_{DD}, V_{DDA}) and ground (V_{SS}, V_{SSA}) pins must always be connected to the external power supply, in the permitted range.
2. Negative injection disturbs the analog performance of the device. See note 3 below *Table 62 on page 105*.
3. Positive injection is not possible on these I/Os. A negative injection is induced by $V_{IN}<V_{SS}$. $I_{INJ(PIN)}$ must never be exceeded. Refer to *Table 7: Voltage characteristics* for the maximum allowed input voltage values.
4. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$. $I_{INJ(PIN)}$ must never be exceeded. Refer to *Table 7: Voltage characteristics* for the maximum allowed input voltage values.
5. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values).

ค.2 เอกสารคู่มือการใช้งานซิกบี

1.XBee/XBee PRO DigiMesh 2.4 OEM RF Modules

The XBee/XBee-PRO DigiMesh 2.4 OEM RF Modules were engineered to support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices.

Key Features

Long-range Data Integrity

XBee
- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100m)
- Transmit Power: 1 mW (0dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO
- Indoor/Urban: up to 300' (90 m)
- Outdoor line-of-sight: up to 1 mile (1600m)
- Transmit Power: 63mW (18dBm) EIRP
- Receiver Sensitivity: -100 dBm (1% packet error rate)
- RF Data Rate: 250,000 bps

Advanced Networking & Security
- Retries and Acknowledgements
- Self-routing, self-healing mesh networking
- DSSS (Direct Sequence Spread Spectrum)

Low Power

XBee
- TX Peak Current: 45mA
- RX Current: 50mA
- Sleep Current: <50uA

XBee-PRO
- TX Peak Current: 250mA (150mA for international variant)
- TX Peak Current (RPSMA module only): 340mA (180mA for international variant)
- RX Current: 55mA
- Sleep Current: <50uA

Easy-to-Use
- No configuration necessary for out-of box RF communications
- AT and API Command Modes for configuring module parameters
- Small form factor
- Extensive command set
- Free X-CTU Software (Testing and configuration software)

Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p63] for FCC Requirements.
Systems that contain XBee®/XBee-PRO RF Modules inherit Digi Certifications.
ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**
Manufactured under **ISO 9001:2000** registered standards
XBee®/XBee-PRO RF Modules aare optimized for use in the United States, Canada, Australia, Israel, Japan, and Europe. Contact Digi for complete list of government agency approvals.

Pin Signals

XBee/XBee-PRO DigiMesh 2.4 RF Module Pin Numbers

(top side shown, shields on bottom)

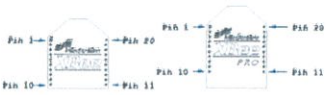


Table 3-01. Pin Assignments for the XBee/XBee-PRO 2.4 DigiMesh Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / $\overline{\text{CONFIG}}$	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	$\overline{\text{RESET}}$	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / $\overline{\text{SLEEP}}$	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	$\overline{\text{RTS}}$ / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 50k Ω pull-up resistor attached to $\overline{\text{RESET}}$
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

ค.3 เอกสารคู่มือการใช้งานออพติคัลเซนเซอร์ TCRT5000

TCRT5000(L)

Vishay Telefunken

Absolute Maximum Ratings

Input (Emitter)

Parameter	Test Conditions	Symbol	Value	Unit
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Forward surge current	t_p 10 µA	I_{FSM}	3	A
Power dissipation	T_{amb} 25°C	P_V	100	mW
Junction temperature		T_j	100	°C

Output (Detector)

Parameter	Test Conditions	Symbol	Value	Unit
Collector emitter voltage		V_{CEO}	70	V
Emitter collector voltage		V_{ECO}	5	V
Collector current		I_C	100	mA
Power dissipation	T_{amb} 55°C	P_V	100	mW
Junction temperature		T_j	100	°C

Sensor

Parameter	Test Conditions	Symbol	Value	Unit
Total power dissipation	T_{amb} 25°C	P_{tot}	200	mW
Operation temperature range		T_{amb}	−25 to +85	°C
Storage temperature range		T_{stg}	−25 to +100	°C
Soldering temperature	2 mm from case, t 10 s	T_{sd}	260	°C

ค.4 เอกสารคู่มือการใช้งานไอซี L293D

SGS-THOMSON MICROELECTRONICS

L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES

SO(12+4+4) Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

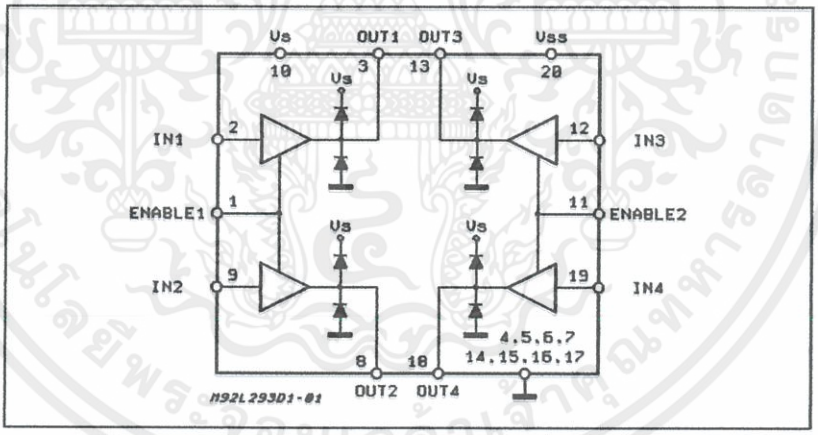
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

BLOCK DIAGRAM

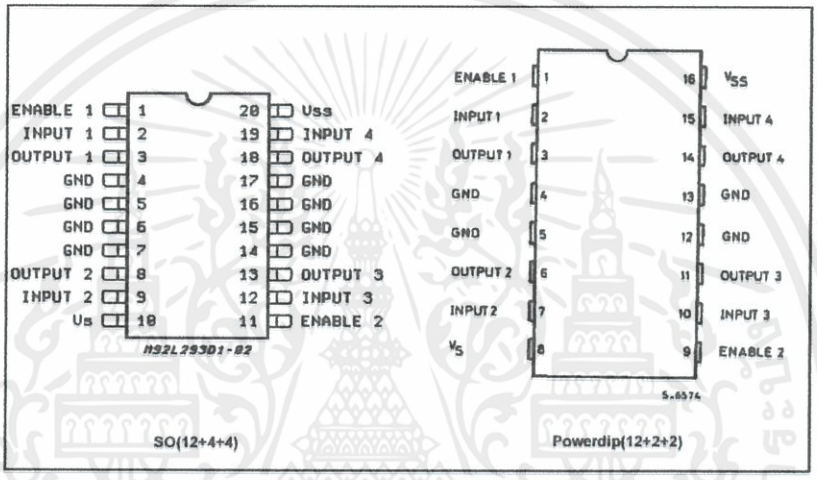


June 1996 1/7

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	- 40 to 150	°C

PIN CONNECTIONS (Top view)



THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max	-	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	-	

(*) With 6sq. cm on board heatsink.

ค.5 เอกสารคู่มือการใช้งานไอซี 7805

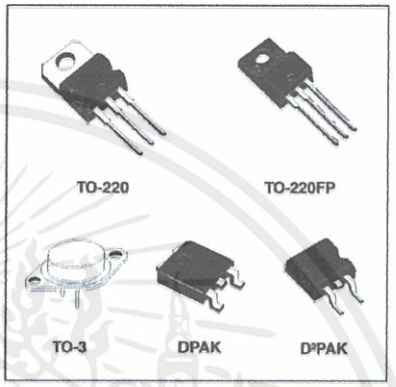
L78xx, L78xxC
L78xxAB, L78xxAC

Positive voltage regulator ICs

Datasheet – production data

Features

- Output current up to 1.5 A
- Output voltages of 5; 6; 8; 8.5; 9; 12; 15; 18; 24 V
- Thermal overload protection
- Short circuit protection
- Output transition SOA protection
- 2 % output voltage tolerance (A version)
- Guaranteed in extended temperature range (A version)



Description

The L78xx series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3, D²PAK and DPAK packages and several fixed output voltages, making it useful in a wide range of applications.

These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

Table 1. Device summary

Part numbers			
L7805	L7806AC	L7809AB	L7815AB
L7805C	L7808C	L7809AC	L7815AC
L7805AB	L7808AB	L7812C	L7818C
L7805AC	L7808AC	L7812AB	L7824C
L7806C	L7885C	L7812AC	L7824AB
L7806AB	L7809C	L7815C	L7824AC

This is information on a product in full production. www.st.com

ค.6 เอกสารคู่มือการใช้งานไอซี LM1117T

National Semiconductor

October 2002

LM1117/LM1117I
800mA Low-Dropout Linear Regulator

General Description

The LM1117 is a series of low dropout voltage regulators with a dropout of 1.2V at 800mA of load current. It has the same pin-out as National Semiconductor's industry standard LM317.

The LM1117 is available in an adjustable version, which can set the output voltage from 1.25V to 13.8V with only two external resistors. In addition, it is also available in five fixed voltages, 1.8V, 2.5V, 2.85V, 3.3V, and 5V.

The LM1117 offers current limiting and thermal shutdown. Its circuit includes a zener trimmed bandgap reference to assure output voltage accuracy to within ±1%.

The LM1117 series is available in LLP, TO-263, SOT-223, TO-220, and TO-252 D-PAK packages. A minimum of 10µF tantalum capacitor is required at the output to improve the transient response and stability.

Features

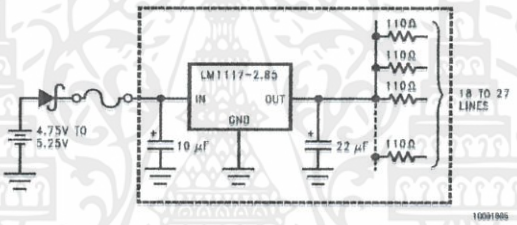
- Available in 1.8V, 2.5V, 2.85V, 3.3V, 5V, and Adjustable Versions
- Space Saving SOT-223 and LLP Packages
- Current Limiting and Thermal Protection
- Output Current 800mA
- Line Regulation 0.2% (Max)
- Load Regulation 0.4% (Max)
- Temperature Range
  - LM1117 0°C to 125°C
  - LM1117I −40°C to 125°C

Applications

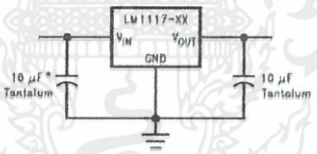
- 2.85V Model for SCSI-2 Active Termination
- Post Regulator for Switching DC/DC Converter
- High Efficiency Linear Regulators
- Battery Charger
- Battery Powered Instrumentation

Typical Application

Active Terminator for SCSI-2 Bus



Fixed Output Regulator



* Required if the regulator is located far from the power supply filter.

ค.7 เอกสารคู่มือการใช้งานไอซี PC814 X_E

PC814XJ0000F Series

[1]4-channel package type is also available.
(model No. **PC844XJ0000F Series**)

DIP 4pin
AC Input Photocoupler

■ Description

PC814XJ0000F Series contains an IRED optically coupled to a phototransistor.

It is packaged in a 4pin DIP, available in SMT gullwing lead-form option.

Input-output isolation voltage(rms) is 5.0kV.

Collector-emitter voltage is 80V and CTR is 20% to 300% at input current of ±1mA.

■ Features

1. 4pin DIP package
2. Double transfer mold package (Ideal for Flow Soldering)
3. AC input type
4. High collector-emitter voltage (V_{CEO} : 80V)
5. Current transfer ratio (CTR : MIN. 20% at I_F=±1mA, V_{CE}=5V)
6. High isolation voltage between input and output ($V_{iso(rms)}$: 5.0 kV)
7. Lead-free and RoHS directive compliant

■ Agency approvals/Compliance

1. Recognized by UL1577, file No. E64380 (as model No. PC814)
2. Approved by VDE, DIN EN60747-5-2[1] (as an option), file No. 40008087 (as model No. PC814)
3. Package resin : UL flammability grade (94V-0)

[1]DIN EN60747-5-2 : successor standard of DIN VDE0884

■ Applications

1. Programmable controllers
2. Telephone sets, telephone exchangers
3. System appliances
4. Signal transmission between circuits of different potentials and impedances

Sheet No.: D2-A03502EN
Date Jun. 30. 2005

ค.8 เอกสารคู่มือการใช้งานรีเลย์ 12 โวลต์

HRS4
Relay

No.1063016 E164730 CQC02001001299 50062840

1.COIL DATA

1-1.Nominal Voltage	3,5,6,9,12,24,48VDC
1-2.Coil Resistance	Refer to Coil Data Chart
1-3.Operate Voltage	Refer to Coil Data Chart
1-4.Release Voltage	Refer to Coil Data Chart
1-5.Nominal Power	360mW、450mW

2.CONTACT DATA

2-1.Contact Form	A-1 Form A,C-1 Form C
2-2.Contact Material	Ag Alloy
2-3.Contact Rating	1A:15A125VAC/24VDC，10A 250VAC 1C:10A 120VC/24VDC,10A/ 6A 250VAC TV-5
2-4.Max.Switching Voltage	250VAC/24VDC
2-5.Max.Switching Current	15A
2-6.Max.Switching Power	1,800VA,360W
2-7.Min.Switching Load	5VDC,100mA
2-8.Contact Resistance	Max 100mΩ (6VDC 1A)
2-9.Life	
Electrical	100,000次
Mechanical	10,000,000次

3.GENERAL DATA

3-1.Insulation Resistance	Min.1000M Ω 500VDC
3-2.Dielectric Strength	750VAC,1 Min between open contacts 1,500VAC,1 min between coil and contacts
3-4.Operate Time	Max.10ms
3-5.Release Time	Max.5ms
3-6.Operate Temperature	-30 ~ +85℃
3-7.Shock Resistance	
Endurance	1,000m/s^2
Misoperation	100m/s^2
3-8.Vibration Resistance	
Endurance	10~ 55Hz,1.5mm double amplitude
Misoperation	10 ~ 55Hz,1.5mm double amplitude
3-9.Humidity	35% ~ 95%RH,+40℃
3-10.Weight	Approximately 10g
3-11.Safety	UL NO.E164730 TUV NO.50062840 CSA NO.1063016 CQC NO.02001001299

ISO9001、ISO/TS16949、ISO14001

เอกสารอ้างอิง

[1] ฉัตราภรณ์ ศรีพิสุทธิ์, ณัฐชัย มาสกุลรัตน์, ธีร์พิสุทธิ์ เดชะไกรศรี. **"ระบบควบคุมตำแหน่งลูกบอลบนระนาบ."** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2553.

[2] นคร ภักดีชาติ. **ปฏิบัติการไมโครคอนโทรลเลอร์ ARM Cortex-M3 กับ STM32.** กรุงเทพมหานคร: บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. 2553.

[3] กิตินันท์ พลสวัสดิ์. **คู่มือเรียนและใช้งาน Visual C# 2010 ฉบับสมบูรณ์.** นนทบุรี: บริษัท ไอดีซี พรีเมียร์ จำกัด. 2553.

[4] CSharp.Net-Informations.com. **"How to export DataGridView to excel file."** [Online].Available: http://csharp.net-informations.com/excel/csharp-exceldatagridview.htm. 2012.

[5] CSharp.Net-Informations.com. **"C# OLEDB Connection."** [Online].Available: http://csharp.net-informations.com/excel/csharp-exceldatagridview.htm. 2012.

[6] Developer C Sharp. **"C# - Get value from another Form."** [Online].Available: http://www.youtube.com/watch?v=--dgs8FuR20. 2012.

[7] Noad coad. **"SerialPort (RS-232 Serial COM Port) in C# .NET."** [Online].Available: http://msmvps.com/blogs/coad/archive/2005/03/23/SerialPort-_2800_RS_2D00_232-Serial-COM-Port_2900_-in-C_2300_-.NET.aspx. 2005.